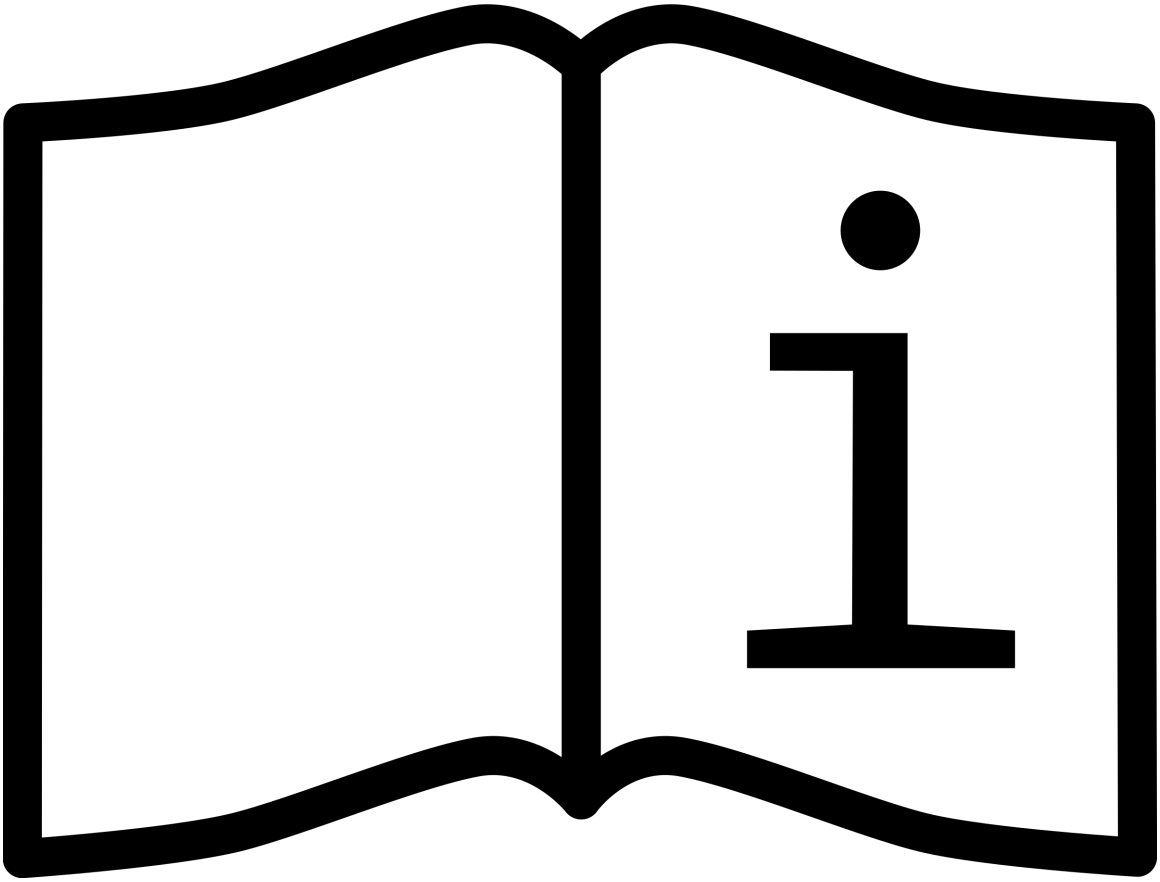
Bern University of Applied Sciences

Project 1

Module: BTI 7301



RPiHSM - Instruction manual

Students

Noli Manzoni, Sandro Tiago Carlao

Tutor

Dr. **Simon Kramer**

Contents

1 Introduction

This document describes how to install and use the RPiHSM. The section 2 explains how to configure the different applications, the Raspberry Pi 3 model B and how to set up the electrical components.

2 Configuration

To configure the different application some software are required. First download and install the Java Development Kit 1.8 or above for you system, then download and install Apache Maven 3.3.9 or above (`https://maven.apache.org/`). Once the two software are installed and correctly configured download the source code from `https://github.com/nolithepain/RPiHSM.git`.

2.1 RPiHSM - IoT

2.1.1 Google Keyczar configuration

To use the Google Keyczar library the source code must be downloaded from `https://github.com/google/keyczar`. Once the code is downloaded a new branch with the last library version as name must be created so that in the compile process errors do not appear (e.g. git checkout -b Java_release_0.71j). To compile the source code so that it can be installed on Maven the following command must be executed in the folder java/code of the Keyczar project. To complete the process, the generated jar must be installed on Maven.

```
Mvn install:install-file -Dfile=keyczar-0.71j-031417.jar -
    DgroupId=org.keyczar -DartifactId=keyczar -Dversion=0.71j
    -Dpackaging=jar
```

Now you can compile the IoT application by using the command *mvn clean package* in the IoT application directory. Once the command is done the jar file can be found in the target folder.

2.1.2 Raspberry Pi 3 model B configuration

OS installation If the Raspberry Pi have a valid operation system installed please go to the next step. First download the Raspbian Jessie with Pixel OS and create a bootable SD card. Once the SD card is ready, to be able to use the serial port at the maximum speed, open it and modify the cmdline.txt. The following line must be replaced by the second one.

```
bash dwc_otg.lpm_enable=0 console=serial0,115200 console=tty1
    root=PARTUUID=402e4a57-02 rootfstype=ext4 elevator=deadline
     fsck.repair=yes rootwait quiet init=/usr/lib/raspi-config/
    init_resize.sh splash plymouth.ignore-serial-consoles

bash dwc_otg.lpm_enable=0 console=tty1,115200 kgdboc=tty1
    ,115200 console=tty1,115200 root=PARTUUID=402e4a57-02
    rootfstype=ext4 elevator=deadline fsck.repair=yes rootwait
    quiet init=/usr/lib/raspi-config/init_resize.sh splash
    plymouth.ignore-serial-consoles
```

Default user The Pi user is a well know name so please change its password. To have higher security (the Pi user have a lot of permissions) a new user with less persmissions must be created. The goal of this user is to be automatically logged in when the Raspberry Pi starts.

```
$ sudo adduser hsm
```

No password for sudo So that the hsm user can execute the sudo command, to start the RPiHSM-IoT application, without the password,the file /etc/sudoers.d/010_pi-nopassword must be modified by adding this line in the end of the file.

```
hsm ALL=(ALL) NOPASSWD: ALL
```

Key sets folder The application needs a specific directory to store the key sets. To have a new keyset directory on every new user a directory must be created in skel folder.

```
$ sudo mkdir sudo mkdir /etc/skel/keyset
```

Application users To be able to use the application with multiple users they must be first added on the Raspberry Pi OS. To each new user the permission on the previously created hsm directory must be changed so that only the owner and the hsm user can write and the others users not.

```
$ sudo adduser $user
$ sudo chgrp -R hsm /home/$user
$ sudo chmod -R 770 /home/$user
```

The variable $user is the new user name.

Auto login for HSM user To auto login the hsm user so that the IoT application can be executed when the Raspberry Pi start up some operations must be done.

```
$ sudo cp /lib/systemd/system/getty@.service
/etc/systemd/system/autologinhsm@.service
$ sudo ln -s /etc/systemd/system/autologinhsm@.service
/etc/systemd/system/getty.target.wants/getty@tty8.service
$ sudo nano /etc/systemd/system/getty.target.wants/
getty@tty8.service
```

Now the line that start with ExecStart= must be modified in:

```
ExecStart=-/sbin/agetty --autologin hsm %I $TERM
```

Then, to execute a correct login the following line must be added in the end of the file.

```
Alias=getty.target.wants/getty@tty8.service
```

To apply the above commands, the system must be restarted (sudo reboot).

Ask username and password for login In the default configuration of the OS the auto login is enable, so to have higher security it must be disabled by following these steps.

```
$ sudo raspi-config
```

Then go to *Boot Options -> Desktop/CLI -> Desktop (Desktop GUI, requiring user to login).* Now the Raspberry Pi will always ask for the username and password when the system start up.

Execute the IoT application when the RPi starts To compile the application please look the section 2.1.1. Once the application jar is generated, it must be moved in the HSM user home directory. Then to start the application when the OS start up, the following line must be added at the end of the file */home/hsm/.profile.*

```
$ sudo java -jar RPiHSM-IoT-0.1.jar &
```

Enable Serial Interface To enable the serial interface the following command must be executed and then you should go in *Interfacing Options -> Serial -> Disable Serial Login Sheel -> Enable Serial Interface.*

```
$ sudo raspi-config
```

This command will also disable the possibility to login with the serial cable.

Test the application Before continue with this configuration make sure that the application work correctly. If under linux problems appear, some updates must be executed.

```
$  sudo apt-get install libpam-runtime
$  sudo apt-get install libpam-modules
$  sudo apt-get install libpam-modules-bin
$  sudo apt-get install libpam0g-dev
```

Remember to delete the WiFi password from the */etc/wpa$_s$upplicant/wpa$_s$upplicant.conf* file and turn off it when the updates are done.

Disable GUI To save energy, resources and to have more security the Raspberry Pi GUI must be disabled with the following steps.

```
sudo raspi-config
```

Then go to *Boot Options -> Desktop/CLI -> Console (Text console, requiring user to login).*

Disable other services To have higher security other services can be disable. To disable the WiFi and Bluetooth services these lines must be added in the */etc/modprobe.d/raspi-blacklist.conf* file.

```
#Disable WiFi
blacklist brcmfmac
blacklist brcmutil
#Disable Bluetooth
blacklist btbcm
blacklist hci_uart
```

To disable the HDMI interface, USB ports and the RJ45 port these lines must be added at the end of the */etc/rc.local* file.

```
#Disable HDMI
/opt/vc/bin/tvservice -o
#Disable USB and RJ45
echo 0 > /sys/devices/platform/soc/3f980000.usb/buspower
exit 0
```

2.1.3 Electrical components

To use the application an USB serial cable is needed. To connect the cable to the Raspberry Pi use the following diagram (the leds are optional).

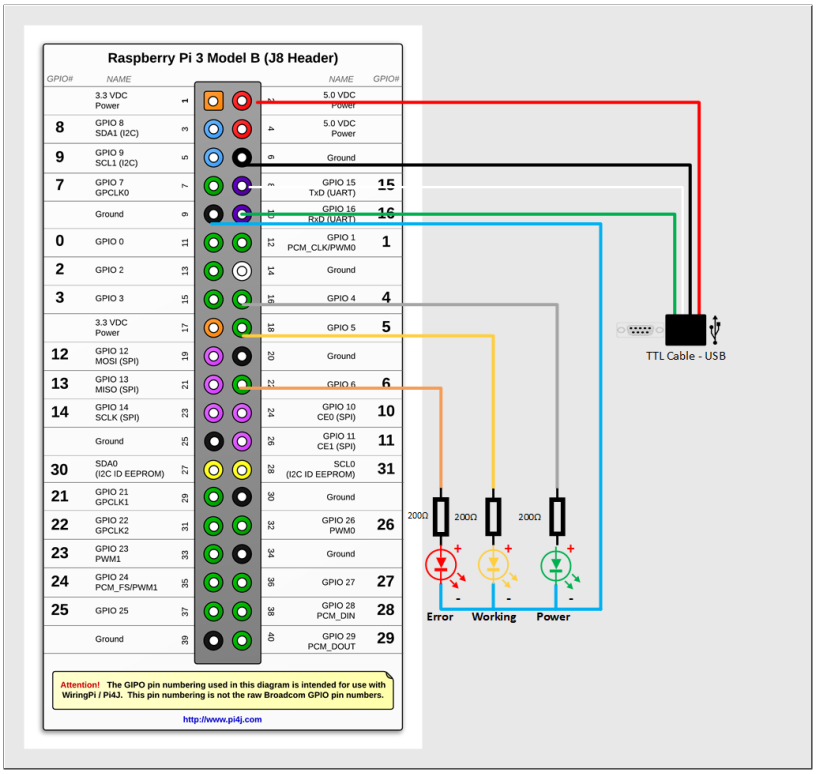


Figure 1: Raspberry Pi electrical components

2.2 RPiHSM - Pc

To be able to use the Command Line application and the GUI, the API must be installed. First the API must be compiled by executing *mvn clean package* in the API application directory. Once the jar is generated it must be installed in maven by using this command (change the version if it is not the same).

```
mvn install:install-file -Dfile=RPiHSM-API-1.0.jar -DgroupId=
    ch.bfh.ti.project1.RPiHSM-API -DartifactId=RPiHSM-API -
    Dversion=1.0 -Dpackaging=jar
```

Once the API are installed in the system the right driver must be installed. The instruction for the installation are found in the driver directory in the API project.

2.2.1 Command Line

To use the application it must be just compiled. To do it the *mvn clean package* command must be executed in the directory of the Command Line project. Once the jar is generated in can be found in the target directory (for the use instruction look at the section 3.2).

2.2.2 Graphical Interface

To use the application it must be just compiled. To do it the *mvn clean package* command must be executed in the directory of the GUI project. Once the jar is generated in can be found in the target directory (for the use instruction look at the section 3.3).

3 Instruction

3.1 IoT

The Raspberry Pi comes with three leds (green, yellow and red). These leds represent different state of the application.

(a) ON (b) BUSY (c) ERROR

Figure 2: Raspberry Pi states

ON When the green led is on means that the application is ready and it can be used.

BUSY When the yellow led is blinking means that the application is busy so no other command can be handle (wait until it turn of).

ERROR When the red led is blinking means that an error is occurred (it is shown in the client application). When the red led is on it means that the application has a failure. To resolve this problem the Raspberry Pi must be disconnected and then re-connected.

3.2 Command Line

This application can be used through the command line of your operating system (only expert user). To use the application you should be located in the directory where the jar file is located.

Figure 3: Jar file location

Each command can be executed by calling the application through Java with the command *java -jar RPiHSM-CommandLine-0.1.jar*. The user must insert its username and password every time.

3.2.1 Commands

help The help command shows all the available command.

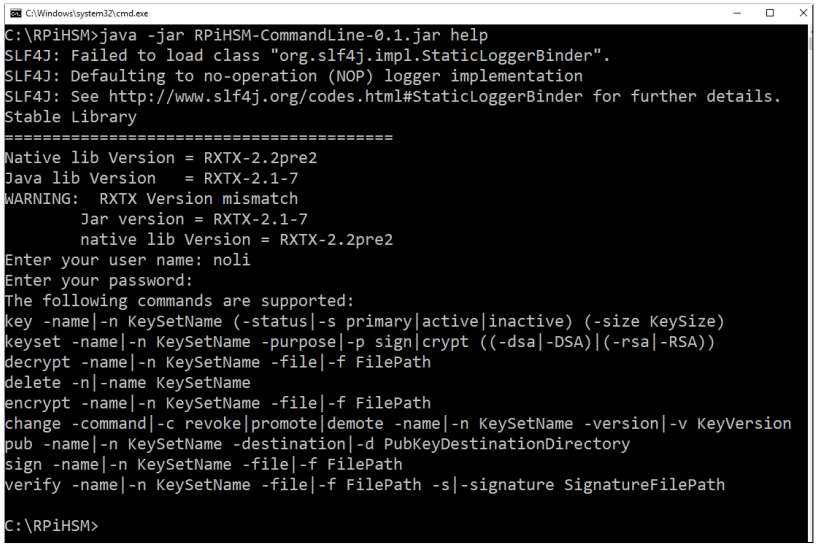


Figure 4: help command

keyset This command allows the user to create a box for the keys. The key set must have a name *-name or -n*, a purpose *-purpose or -p* that can be crypt or sign and optionally a algorithm *-dsa or -DSA* or *-rsa or -RSA*.

```
C:\RPiHSM>java -jar RPiHSM-CommandLine-0.1.jar keyset -n test
    -p crypt -rsa
C:\RPiHSM>java -jar RPiHSM-CommandLine-0.1.jar keyset -name
    test_sign -purpose sign
```

key This command allow the user to add a key in a given key set. The command must have the name of the key set *-name or -n* and optionally the start status of the key *-status or -s* that must be primary, active or inactive, and the key size *-size* that must be a multiple of 2.

```
C:\RPiHSM>java -jar RPiHSM-CommandLine-0.1.jar key -name test
    -s active -size 1024
C:\RPiHSM>java -jar RPiHSM-CommandLine-0.1.jar key -name test
```

encrypt This command allows the user to encrypt a given file with the primary key of a given key set. The command must have the name of the key set *-name or -n* and the file path *-file or -f*. If not algorithm is given aes is used.

```
C:\RPiHSM>java -jar RPiHSM-CommandLine-0.1.jar encrypt -n test
    -file C:\RPIHSM\image.jpg
```

decrypt This command allows the user to decrypt a given file with the primary key of a given key set. The command must have the name of the key set *-name or -n* and the file path *-file or -f*.

```
C:\RPiHSM>java -jar RPiHSM-CommandLine-0.1.jar decrypt -n test
    -file C:\RPIHSM\image.jpg
```

delete This command allows the user to delete a key set and its keys. The command need the key set name *-name or -n*.

```
C:\RPiHSM>java -jar RPiHSM-CommandLine-0.1.jar delete -name
    test
```

change This command allows the user to change the status of a given key of a given key set. The command need the desired operation *-c or -command* that must be revoke (delete a key), promote (increase status) or demote (decrease status), the key set name *-name or -n* and the version of the key *-version or -v* (the key name -> the keys are named with their creation order number).

```
C:\RPiHSM>java -jar RPiHSM-CommandLine-0.1.jar change -c
    demote -n test -v 2
C:\RPiHSM>java -jar RPiHSM-CommandLine-0.1.jar change -c
    promote -n test -v 1
```

pub This command allows the user to export the public keys of a given key set. The command must have the key set name *-name or -n* and the destination folder *-destination or -d* (the destination folder contents will be erased).

```
C:\RPiHSM>java -jar RPiHSM-CommandLine-0.1.jar pub -n test -d
    C:\RPiHSM\pub\
```

sign This command allow the user to create a signature of a given file with a given key set. The command must have the key set name *-name or -n* and the file path *-file or -f* (give to the command a copy of the file because it will be overwrote).

```
C:\RPiHSM>java -jar RPiHSM-CommandLine-0.1.jar sign -n
    test_sign -f C:\RPIHSM\image.jpg
```

verify This command allow the user to verify a signature of a given file with a given key set. The command must have the key set name *-name or -n*, the file path *-file or -f* and the signature path *-signature or -s*

```
C:\RPiHSM>java -jar RPiHSM-CommandLine-0.1.jar verify -n
    test_sign -f C:\RPIHSM\image.jpg -signature  C:\RPIHSM\
    signature.bin
```

3.2.2 Errors

In this section are described the errors that can appear in the command line application.

Illegal Argument in the given command This error can appear when:

1. The arguments are wrong or are missing.
2. The key size is less than 0 or is not a multiple of 2.
3. The key status is neither primary, nor active and nor inactive.
4. The change command is neither demote, nor promote and nor revoke.

Wrong user or password The user information are wrong.

The serial port is user by another process There are another another application that is using the serial port (close other instance of the application).

Something has gone wrong with the connection The serial port have some problem (try to disconnect and re-connect the Raspberry Pi).

The serial port has some problems The application cannot close the connection or the Raspberry Pi is connected with a invalid serial cable.

Something has gone wrong with the command execution The program cannot write on the serial cable (try to disconnect and re-connect the Raspberry Pi).

The key creation has not been successful The IoT application has returned an error, check if the given key set exist otherwise try to disconnect and re-connect the Raspberry Pi.

DSA cannot be used with RSA These two algorithm cannot be used together.

DSA can be use only with sign key set The dsa algorithm can be use only with sign purpose.

The key set creation has not been successful The IoT application has returned an error. Maybe the key set already exist (delete the key set with the delete command or add the key to it with the key command).

File not found The given file path point to a invalid or a non existent file.

The decryption has not been successful The IoT application has returned an error, check if the given key set exist otherwise try to disconnect and re-connect the Raspberry Pi. Be sure to use the same key set and key that was used for the encryption.

The key set deletion has not been successful The IoT application has returned an error, check if the given key set exist otherwise try to disconnect and re-connect the Raspberry Pi.

The encryption has not been successful The IoT application has returned an error, check if the given key set exist otherwise try to disconnect and re-connect the Raspberry Pi. If you are using a rsa key set your file must be smaller than 470 bytes (this limit is forced by Google Keyczar [1]). Be sure that the used key set was created with crypt purpose and that it contains at least a key.

The demotion has not been successful The IoT application has returned an error, check if the given key set exist otherwise try to disconnect and re-connect the Raspberry Pi. Be sure that the demoted key is not already inactive and that the given key number exists.

The promotion has not been successful The IoT application has returned an error, check if the given key set exist otherwise try to disconnect and re-connect the Raspberry Pi. Be sure that the promoted key is not already primary and that the given key number exists.

The revocation has not been successful The IoT application has returned an error, check if the given key set exist otherwise try to disconnect and re-connect the Raspberry Pi. Be sure that the revoked key is not active or primary (it must be inactive to be revoked) and that the given key number exists.

The public key exportation has not been successful The IoT application has returned an error, check if the given key set exist otherwise try to disconnect and re-connect the Raspberry Pi.

The key set is empty The key exportation has not been successful because the given key set contains any keys. Use the command key to add new keys to it.

The key set is not asymmetric The given key set was created with crypt purpose and with the default algorithm (aes) hence the public keys cannot be exported.

The signing operation has not been successful The IoT application has returned an error, check if the given key set exist otherwise try to disconnect and re-connect the Raspberry Pi. Be sure that the used key set was created with sign purpose and that it contains at least a key.

The verification has not been successful The IoT application has returned an error, check if the given key set exist otherwise try to disconnect and re-connect the Raspberry Pi. Be sure to use the same key set and key that was used for the signature generation.

3.3 Graphical Interface

To open the graphical user interface you should double click the generated jar. Once the GUI is opened you can change the program language and login with a valid user.

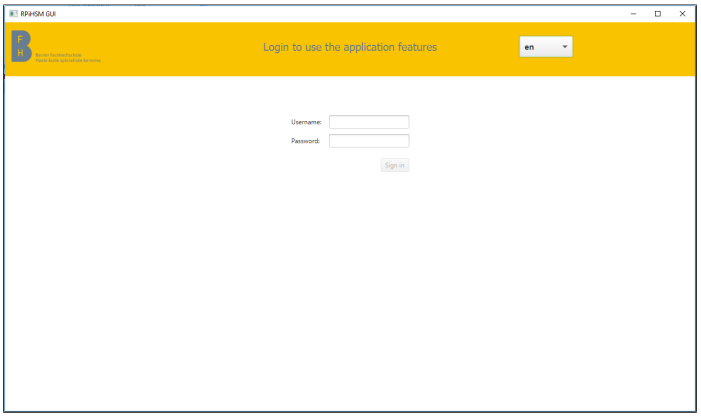


Figure 5: Welcome page

3.3.1 Commands

Login To login an user name (longer than three characters) and a password must be inserted otherwise the sign in button is disabled.

(a) Sign in disabled (b) Sign in enabled

Once the login has success the main menu is opened.

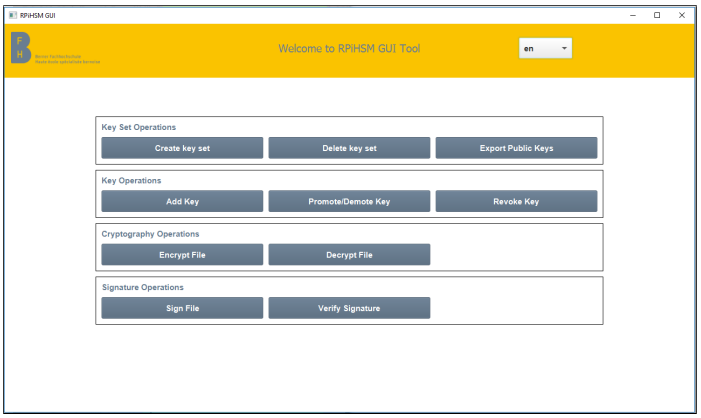
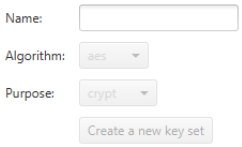
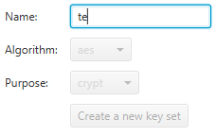


Figure 7: Application menu

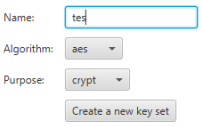
Create key set To enable the other buttons a name longer than three characters is needed.



(a) Buttons disabled

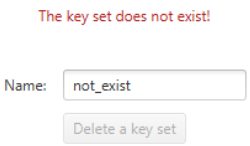


(b) Name to small

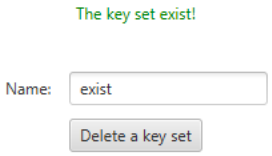


(c) Buttons enabled

Delete key set To enable the delete button a valid key set name must be inserted. It is enabled only if the key set exist.

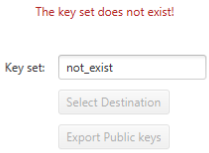


(a) Key set doesn't exist

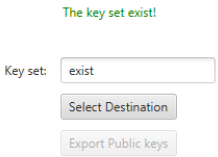


(b) Key set exists

Export public keys To enable the destination button an existing key set name must be inserted.

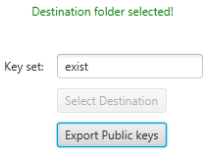


(a) Key set doesn't exist



(b) Key set exists

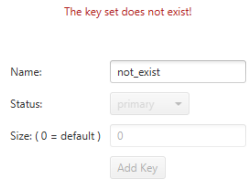
Once the key set is verified a destination directory must be chosen to be able to export the public keys.



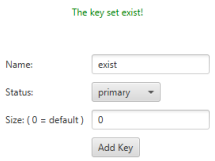
(a) Destination folder

Once the operation has terminated the public key can be found in the destination folder. Each key have its own file.

Add key To enable the status button and the size text field an existing key set name must be inserted.

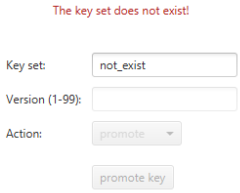


(a) Key set doesn't exist

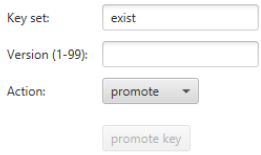


(b) Key set exists

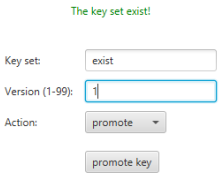
Promote/Demote key To enable the version text field an existing key set name must be inserted. Then a valid key number must be inserted to be able to promote or to demote the key.



(a) Key set doesn't exist

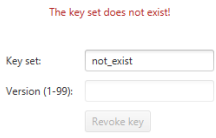


(b) Key set exists

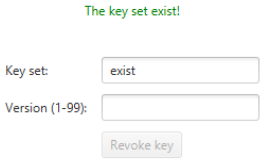


(c) Valid key number

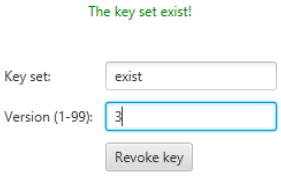
Demote key To enable the version text field an existing key set name must be inserted. Then a valid key number must be inserted to be able to revoke the key.



(a) Key set doesn't exist

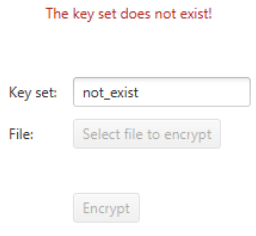


(b) Key set exists

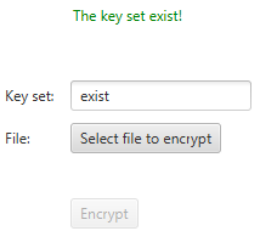


(c) Valid key number

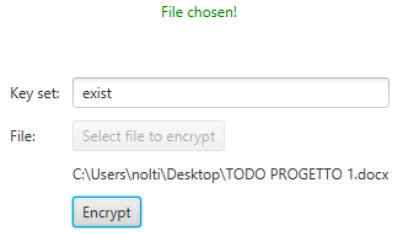
Encrypt file To enable the file chooser an existing key set name must be inserted. Then to enable the encrypt button a valid file must be chosen. If the key set use the rsa algorithm the file cannot be larger than 470 bytes (limited by Google Keyczar [1]). Be sure to use a key set created with crypt purpose.



(a) Key set doesn't exist

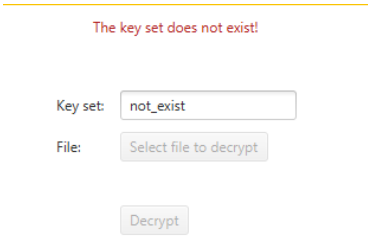


(b) Key set exists

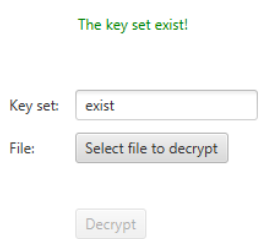


(c) File chosen

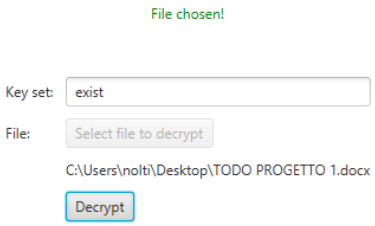
Decrypt To enable the file chooser an existing key set name must be inserted. Then to enable the decrypt button a valid file must be chosen. Be sure to use the same key set used in the encryption.



(a) Key set doesn't exist

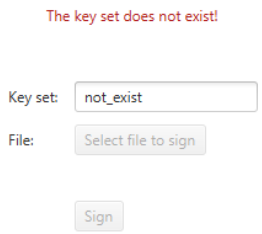


(b) Key set exists

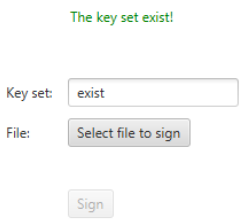


(c) File chosen

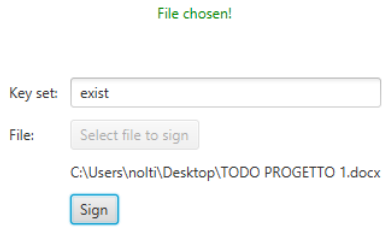
Sign file To enable the file chooser an existing key set name must be inserted. Then to enable the sign button a valid file must be chosen. Be sure to use a key set created with sign purpose.



(a) Key set doesn't exist

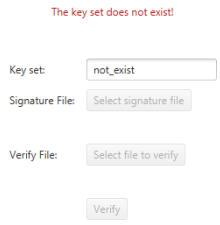


(b) Key set exists

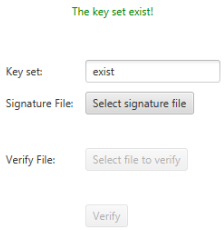


(c) File chosen

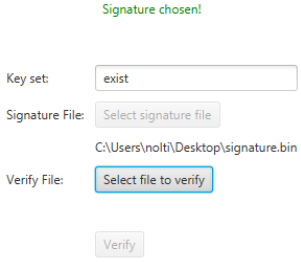
Verify signature To enable the signature file chooser an existing key set name must be inserted. Then to enable the file chooser a valid signature must be uploaded. Once the file is selected the verify button will be enabled. Be sure to use the same key set used in the signing operation.



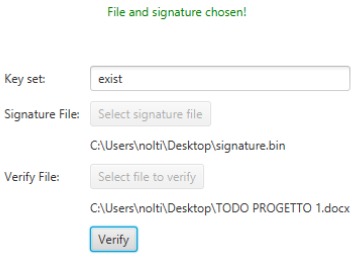
(a) Key set doesn't exist



(b) Key set exists



(c) Signature chosen



(d) File chosen

3.3.2 Errors

The USB port is not connected The serial cable is not connected. Please attach the cable.

Wrong credentials The user information are wrong.

The serial port is user by another process There are another another application that is using the serial port (close other instance of the application).

Something has gone wrong with the connection The serial port have some problem (try to disconnect and re-connect the Raspberry Pi).

The serial port has some problems The application cannot close the connection or the Raspberry Pi is connected with a invalid serial cable.

Something has gone wrong with the command execution The program cannot write on the serial cable (try to disconnect and re-connect the Raspberry Pi).

The key creation has not been successful The IoT application has returned an error, check if the given key set exist otherwise try to disconnect and re-connect the Raspberry Pi.

DSA cannot be used with RSA These two algorithm cannot be used together.

DSA can be use only with sign key set The dsa algorithm can be use only with sign purpose.

The key set creation has not been successful The IoT application has returned an error. Maybe the key set already exist (delete the key set with the delete command or add the key to it with the key command).

File not found The given file path point to a invalid or a non existent file.

The decryption has not been successful The IoT application has returned an error, check if the given key set exist otherwise try to disconnect and re-connect the Raspberry Pi. Be sure to use the same key set and key that was used for the encryption.

The key set deletion has not been successful The IoT application has returned an error, check if the given key set exist otherwise try to disconnect and re-connect the Raspberry Pi.

The encryption has not been successful The IoT application has returned an error, check if the given key set exist otherwise try to disconnect and re-connect the Raspberry Pi. If you are using a rsa key set your file must be smaller than 470 bytes (this limit is forced by Google Keyczar [1]). Be sure that the used key set was created with crypt purpose and that it contains at least a key.

The demotion has not been successful The IoT application has returned an error, check if the given key set exist otherwise try to disconnect and re-connect the Raspberry Pi. Be sure that the demoted key is not already inactive and that the given key number exists.

The promotion has not been successful The IoT application has returned an error, check if the given key set exist otherwise try to disconnect and re-connect the Raspberry Pi. Be sure that the promoted key is not already primary and that the given key number exists.

The revocation has not been successful The IoT application has returned an error, check if the given key set exist otherwise try to disconnect and re-connect the Raspberry Pi. Be sure that the revoked key is not active or primary (it must be inactive to be revoked) and that the given key number exists.

The public key exportation has not been successful The IoT application has returned an error, check if the given key set exist otherwise try to disconnect and re-connect the Raspberry Pi.

The key set is empty The key exportation has not been successful because the given key set contains any keys. Use the command key to add new keys to it.

The key set is not asymmetric The given key set was created with crypt purpose and with the default algorithm (aes) hence the public keys cannot be exported.

The signing operation has not been successful The IoT application has returned an error, check if the given key set exist otherwise try to disconnect and re-connect the Raspberry Pi. Be sure that the used key set was created with sign purpose and that it contains at least a key.

The verification has not been successful The IoT application has returned an error, check if the given key set exist otherwise try to disconnect and re-connect the Raspberry Pi. Be sure to use the same key set and key that was used for the signature generation.

4 Contacts

If you have any questions or problems please contact us at noli.manzoni@students.bfh.ch or sandro.tiagocarlao@students.bfh.ch. If you want more information about the case you can contact the designer at kevinalexander.thomas@students.bfh.ch.

Berner Fachhochschule
Haute école spécialisée bernoise
Bern University of Applied Sciences

This project was realised at the **Bern University Of Applied Sciences** [2].

References

[1] *Toolkit made by Google designed to make it easier and safer for developers to use cryptography in their applications.*
`https://github.com/google/keyczar`

[2] *Bern University Of Applied Sciences in Biel/Bienne Switzerland*
`https://www.bfh.ch`