# OpenPGP Card Application
# User Guide


*Cédric Mesnil (cedric@ledger.fr)*

December 11, 2019

# Contents



## 0.1 License


Author: Cedric Mesnil <cedric@ledger.fr>

## 0.2 Introduction

GnuPG application for Ledger Blue and Nano S

This application implements "The OpenPGP card" specification revision 3.3. This specification is available in doc directory and at https://g10code.com/p-card.html .

The application supports:

- RSA with key up to 4096 bits
- ECDSA with secp256k1, secp256r1, brainpool 256r1 and brainpool 256t1 curves
- EDDSA with Ed25519 curve
- ECDH with secp256k1, secp256r1, brainpool 256r1, brainpool 256t1 and curve25519 curves

This release has known missing parts (see also [GPGADD]) :

- Ledger Blue support
- Seed mode ON/OFF via apdu

## 0.3 How to install GPG Application

### 0.3.1 Nano S / Blue

For both, source and binary installation, use the most recent tag.

#### 0.3.1.1 From Binary

Use the "Ledger Manager" Chrome App. See https://www.ledgerwallet.com/apps/manager for details.

As the "OpenPGP card" application is not fully compliant with UI and documentation guidelines, the application is in developer section: click on "Show delevopers items" on the bottom right corner to see it.

- Launch the Ledger Manager. See Ledger Manager and https://ledger.groovehq.com/knowledge_base/topics/ledger-manager for details about installing and using the manager;
- Connect your Nano S or your Blue, enter your PIN, and stay on the dashboard;
- Click on *show developer items* on the bottom right corner;
- Click on the green bottom arrow icon near the Ledger *Open PGP* logo;
- Confirm the installation when required on your device by pressing the right button above the checkmark;
- Quit the Ledger Manager

The application is ready to use!

#### 0.3.1.2 From source

Building from sources requires the the Nano S SDK 1.4.2.1 on firmware 1.4.2. See https://github.com/LedgerHQ/nanos-secure-sdk

Refer to the SDK documentation for the compiling/loading...

### 0.3.2 System Configuration

For Linux and MAC, until version 1.4.27, Ledger CCID interface is not supported by default by pcscd and must be manually added

For windows....

#### 0.3.2.1 Linux

You have to have to add the NanoS to /etc/libccid_Info.plist

- In <key>ifdVendorID</key> add the entry <string>0x2C97</string>
- In <key>ifdProductID</key> add the entry <string>0x0001</string>
- In <key>ifdFriendlyName</key> add the entry <string>Ledger Token</string>

These 3 entries must be added at the end of each list.

#### 0.3.2.2 MAC

1. First it is necessary to [disable SIP](https://developer.apple.com/library/mac/documentation/Security/Conceptual/System_Integrity_Protection_Guide/ConfiguringSystemIntegrityProtection/ConfiguringSystemIntegrityProtection.html) That doesn't allow the editing of files in /usr/.
2. You have to add the Nano S to /usr/libexec/SmartCardServices/drivers/ifd-ccid.bundle/Contents/Info.plist
    - In <key>ifdVendorID</key> add the entry <string>0x2C97</string>
    - In <key>ifdProductID</key> add the entry <string>0x0001</string>
    - In <key>ifdFriendlyName</key> add the entry <string>Ledger Token</string>

This 3 entries must be added at the end of each list.

3. [Enable SIP](https://developer.apple.com/library/content/documentation/Security/Conceptual/System_Integrity_Protection_Guide/ConfiguringSystemIntegrityProtection/ConfiguringSystemIntegrityProtection.html)

#### 0.3.2.3 Windows

TODO

## 0.4 Nano S OpenPGP Card application explained

### 0.4.1 Menu Overview

The full menu layout is :

*Device Info*
Select Slot
    *Choose:*
    Slot 1 #+
    Slot 2
    Slot 3
    Set Default
Settings
    Key Template
        Choose Key...
            Signature
            Decryption
            Authentication
        Choose Type...
            RSA 2048
            RSA 3072

RSA 4096
NIST P256
ED25519
Set Template
Seed mode
<ON|OFF>
Set on
Set off
PIN mode
UIF mode
*Choose:*
Host
On Screen
Confirm only #+
Trust
Reset
About
*OpenPGP Card*
*(c) Ledger SAS*
*Spec 3.0*
*App 1.0.1*

Emphasis entries are not selectable and just provide information.
A "#" after the entry label means default value on reset.
A "+" after the entry label means current value.

### 0.4.2 Device Info

The *Device Info* provides current user and slot information. The format is:

<User: **name**/ Slot: **n** / Serial: **s** >

with:

- **name** is the one provided to `gpg --card-edit`. See [GPGSC].
- **n** is the current slot, see below.
- **s** is the 32 bits card serial number. Note that the last three bits always

encode the current slot value.

### 0.4.3 Select Slot

This menu is only available on `XL` version

A Slot is a set of three key pairs *Signature, Decryption, Authentication* as defined by gnupg specification.

Usually a GPG card application only manages a single set. Ledger version enhances this and allows you to manage three key sets.

The *Select Slot* menu allows you to select the slot you want to play with, and to set the default slot when the application start.

To change the current slot, display the slot you want and select it

To change the default slot, first select it, and then select the *Set Default* entry.

### 0.4.4 Settings

#### 0.4.4.1 Key Template

A key template is defined by the OpenGPG card application specification. It describes the key to be generated with the `generate` command in `gpg --card-edit`

To set up a new ECC template you have three choices: the NanoS menu, the `gpg-connect-agent` tool and last, the `gpg --edit-card` interactive setup.

**gpg --card-edit** (recommended)

This method suppose you have a recent GnuPG tool and that you correctly configured it. See the dedicated section for that.

In a terminal launch :

```
$ gpg --card-edit
gpg/card> admin
Admin commands are allowed

gpg/card> set-key
Changing card key attribute for: Signature key
Please select what kind of key you want:
  (1) RSA
  (2) ECC
Your selection? 2
Please select which elliptic curve you want:
  (1) Curve 25519
  (4) NIST P-384
Your selection? 1
The card will now be re-configured to generate a key of type: ed25519
Note: There is no guarantee that the card supports the requested size.
    If the key generation does not succeed, please check the
    documentation of your card to see what sizes are allowed.
Changing card key attribute for: Encryption key
Please select what kind of key you want:
  (1) RSA
  (2) ECC
Your selection? 2
Please select which elliptic curve you want:
  (1) Curve 25519
  (4) NIST P-384
Your selection? 1
The card will now be re-configured to generate a key of type: cv25519
Changing card key attribute for: Authentication key
Please select what kind of key you want:
  (1) RSA
  (2) ECC
Your selection? 2
Please select which elliptic curve you want:
  (1) Curve 25519
  (4) NIST P-384
Your selection? 1
The card will now be re-configured to generate a key of type: ed25519
```

To show the current template use the `gpg --card-status` command.

**gpg-connect-agent**

This method suppose you have correctly configured your GnuPG tool. See the dedicated section for that.

In a terminal launch :

```
gpg-connect-agent "SCD SETATTR KEY-ATTR --force 1 <tag> <curvename>" /bye
gpg-connect-agent "SCD SETATTR KEY-ATTR --force 2 18    <curvename>" /bye
gpg-connect-agent "SCD SETATTR KEY-ATTR --force 3 <tag> <curvename>" /bye
```

This 3 commands fix, in that order, the template for Signature, Decryption, Authentication keys.

Supported curve name are:

- secp256k1 with tag 19
- nistp256 with tag 19

- cv25519 (only for key 2)
- ed25519 with tag 22 (only for key 1 and 3)

To show the current template use the `gpg --card-status` command.

**NanoS menu**

First under *Choose Key* menu, select the one of three keys for which you want to modify the template. Then under "Choose Type", select the desired key template. Finally select "Set Template" entry to set it.

To show the current template use the `gpg --card-status` command.

#### 0.4.4.2 Seed mode

**WARNING** : SEED MODE IS EXPERIMENTAL

When generating new keys on NanoS, those keys can be generated randomly or in a deterministic way. The deterministic way is specified in [GPGADD]. The current mode is displayed in the first sub menu. To activate the seeded mode select *ON*, to deactivate the seeded mode select *OFF*.

When the application starts, the seeded mode is always set to *OFF*

**WARNING** : SEED MODE IS EXPERIMENTAL

#### 0.4.4.3 PIN mode

Some operations require the user to enter his PIN code. The default PIN values are:

- user: `123456`
- admin: `12345678`

The PIN entry can be done using four methods, named "*Host*", "*On Screen*", "*Confirm only*", "*Trust*".

After each mode a + or # symbol may appear to tell which mode is the current one and which one is the default when the application starts. The default mode can be changed by first selecting the desired mode and then selecting the *Set default" menu. Note that*Trust* can not be set as default mode.

Before you can change the PIN mode, you need to verify the PIN on the client. To do this, run `gpg --card-edit`, then `admin` and finally `verify` on you PC. You will then be asked to enter the current PIN. After doing so, you can change the PIN mode on your device.

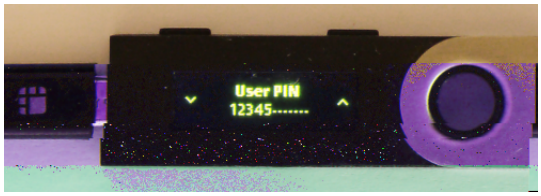
Note that *On Screen*", "*Confirm only*" and "*Trust*" may not work if the client application does not support it. In that case the "*Host*" should be automatically used by the client in a transparent way.

**Host**

The PIN is entered on the external computer.

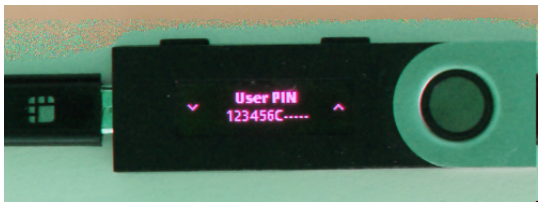
**On Screen**

The PIN is entered on the Nano S or Blue screen. For entering the PIN choose the next digit by using the left or right button. When the digit you expect is displayed select it by pressing both buttons ar the same time

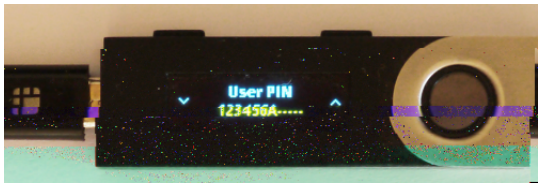



Once all digits are selected, validate the PIN by selecting the **'V'** (Validate) letter


If you want to change the previous digit select the **'C'** (Cancel) letter.




Finnaly if you want to abort the PIN entry, select the **'A'** (Abort) letter.




**Confirm only**

The user is requested, on the NanoS or Blue screen, to confirm the PIN validation. The PIN value is not required, the user just has to push the *REJECT* or *OK* button on the device.

This is the default mode after application installation.


**Trust**

Act as if the PIN is always validated. This is a dangerous mode which should only be used in a highly secure environment.

#### 0.4.4.4 UIF mode

By activating UIF mode for either signature, decryption or authentication, a user validation will be ask by the device each time the related operation is performed.

To activate or deactivate the UIF, select the operation to protect and press both button. When activated, a '+' symbol appears after the operation name.

#### 0.4.4.5 Reset

Selecting the menu will erase all OpenPGP Card Application data and will reset the application in its '*just installed*' state.

## 0.5 Nano S OpenPGP Card application usage

### 0.5.1 GPG

The OpenGPG Card application need at least version 2.1.19 for full support. A version prior to 2.1.19 will fail when using ECC.

You should test with a test key and make a backup of your keyring before starting, except if your are sure about what you do.

#### 0.5.1.1 Configuration

In order to use a Ledger device with gpg it is needed to explicitly setup the reader and the delegated PIN support. Edit the file ~/.gnupg/scdaemon.conf and add the following lines:

```
reader-port "Ledger Token [Nano S] (0001) 01 00"
allow-admin
enable-pinpad-varlen
```

If you do not set the `enable-pinpad-varlen` option, even if Nano S is configured in *On Screen* mode, gpg will keep requesting the PIN on the host.

You can check `the reader-port` value by running the command line `pcsc_scan`:

```
$ pcsc_scan
PC/SC device scanner
V 1.4.27 (c) 2001-2011, Ludovic Rousseau <ludovic.rousseau@free.fr>
Compiled with PC/SC lite version: 1.8.14
Using reader plug'n play mechanism
Scanning present readers...
0: Alcor Micro AU9560 00 00
1: Ledger Token [Nano S] (0001) 01 00
Reader 0: Alcor Micro AU9560 00 00
Card state: Card removed,
Reader 1: Ledger Token [Nano S] (0001) 01 00
Card state: Card inserted,
ATR: 3B 00
+ TS = 3B --> Direct Convention
+ T0 = 00, Y(1): 0000, K: 0 (historical bytes)
```

#### 0.5.1.2 Get/Set basic information

The `gpg --card-status` command provides default card information. Just after installation it should look like this:

```
$ gpg --card-status
Reader ...........: Ledger Token [Nano S] (0001) 01 00
Application ID ...: D2760001240103002C97AFB114290000
Version ..........: 3.0
Manufacturer .....: unknown
Serial number ....: AFB11429
Name of cardholder: [not set]
Language prefs ...: [not set]
Sex ..............: unspecified
URL of public key : [not set]
Login data .......: [not set]
Signature PIN ....: not forced
Key attributes ...: rsa2048 rsa2048 rsa2048
Max. PIN lengths .: 12 12 12
PIN retry counter : 3 0 3
Signature counter : 0
Signature key ....: [none]
Encryption key....: [none]
Authentication key: [none]
General key info..: [none]
```

You can set the user information with the `gpg --card-edit` command and `name`, `url`, `login`, `lang`, `sex` subcommands. For example if you want to set up your name:

```
$ gpg --card-edit
gpg/card> admin
Admin commands are allowed

gpg/card> name
Cardholder's surname: Mesnil
Cardholder's given name: Cedric

gpg/card> sex
Sex ((M)ale, (F)emale or space): M

gpg/card> list

Reader ...........: Ledger Token [Nano S] (0001) 01 00
Application ID ...: D2760001240103002C97AFB114290000
Version ..........: 3.0
Manufacturer .....: unknown
Serial number ....: AFB11429
Name of cardholder: Cedric Mesnil
Language prefs ...: [not set]
Sex ..............: unspecified
URL of public key : [not set]
Login data .......: [not set]
Signature PIN ....: not forced
Key attributes ...: rsa2048 rsa2048 rsa2048
Max. PIN lengths .: 12 12 12
PIN retry counter : 3 0 3
Signature counter : 0
Signature key ....: [none]
Encryption key....: [none]
Authentication key: [none]
General key info..: [none]
```

Notes:

- Modifying the user information will prompt you to enter `User PIN`.
- Setting user information is not required for using gpg client.

#### 0.5.1.3 Generate new key pair

For generating a new key pair follow those steps:

- Select the desired NanoS OpenPGP Card application slot
- Setup the desired key template for this slot
- Generate the new key set

**Step 1**

Starting from main menu:

- Select *Select slot* menu
- Scroll to desired slot
- Select it
- Optionally set it as default by selecting *Set Default* menu
- Select *Back* to return to main menu.

**Step 2**

The default template for each three keys (*signature*, *decryption*, *authentication*) is `RSA 2048`. If you want another kind of key you have to set the template before generating keys.

!WARNING!: changing the current template of a key automatically erases the associated key.

Starting from main menu:

- Select *Settings* menu
- Select *Key template* menu
- Select *Choose Key...* menu (a)
- Scroll and select which key you want to set the new template for
- Select *Choose type...* menu
- Scroll and select among the supported key types and sizes
- Select *Set template*
- Repeat this process from (a) if you want to modify another key template
- Select *Back* to return to main menu.

**Step 3**

Once the template has been set, it's possible to generate new key pairs with `gpg`.

!WARNING!: gpg will generate the three key pairs and will overwrite any key already present in the selected slot.

Here after is a detailed log of key generation of ECC keys, assuming the three key templates are `NIST P256`.

**Edit Card**

```
$ gpg2 --edit-card
Reader ...........: Ledger Token [Nano S] (0001) 01 00
Application ID ...: D2760001240103002C97AFB1142B0000
Version ..........: 3.0
Manufacturer .....: unknown
Serial number ....: AFB1142B
Name of cardholder: Cedric Mesnil
Language prefs ...: [not set]
Sex ..............: male
URL of public key : [not set]
Login data .......: [not set]
Signature PIN ....: not forced
Key attributes ...: nistp256 nistp256 nistp256
Max. PIN lengths .: 12 12 12
PIN retry counter : 3 0 3
Signature counter : 0
Signature key ....: [none]
Encryption key....: [none]
Authentication key: [none]
General key info..: [none]
```

**Switch to admin mode:**

```
gpg/card> admin
Admin commands are allowed
```

**Request new key generation without backup**

```
gpg/card> generate
Make off-card backup of encryption key? (Y/n) n
```

**Unlock user level "81"**

```
Please unlock the card

Number: 2C97 AFB1142B
Holder: Cedric Mesnil

Use the reader's pinpad for input.
OK
Press any key to continue.
```

**Set key validity**

```
Please specify how long the key should be valid.
0 = key does not expire
<n>  = key expires in n days
<n>w = key expires in n weeks
<n>m = key expires in n months
<n>y = key expires in n years
Key is valid for? (0) 0
Key does not expire at all
Is this correct? (y/N) y
```

**Set user ID**

```
GnuPG needs to construct a user ID to identify your key.

Real name: Cedric Mesnil
Email address: cedric@ledger.fr
Comment:
You selected this USER-ID:
"Cedric Mesnil <cedric@ledger.fr>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? O

You selected this USER-ID:
"Cedric Mesnil <cedric@ledger.fr>"
```

**Unlock admin level "83"**

```
Please enter the Admin PIN

Number: 2C97 AFB1142B
Holder: Cedric Mesnil

Use the reader's pinpad for input.
OK
Press any key to continue.
```

**Unlock user level "82"**

```
Please unlock the card

Number: 2C97 AFB1142B
Holder: Cedric Mesnil
Counter: 8

Use the reader's pinpad for input.
OK
Press any key to continue.
```

**Final confirmation**

```
Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? O
gpg: key DF3FA4A33EF00E47 marked as ultimately trusted
gpg: revocation certificate stored as '/home/gnuk/.gnupg/openpgp-revocs.d/89F772243C9A3E583CB59A
public and secret key created and signed.
```

**Get information after key generation**

```
gpg/card> list

Reader ...........: Ledger Token [Nano S] (0001) 01 00
Application ID ...: D2760001240103002C97AFB1142B0000
Version ..........: 3.0
```

```
Manufacturer .....: unknown
Serial number ....: AFB1142B
Name of cardholder: Cedric Mesnil
Language prefs ...: [not set]
Sex ..............: male
URL of public key : [not set]
Login data .......: [not set]
Signature PIN ....: not forced
Key attributes ...: nistp256 nistp256 nistp256
Max. PIN lengths .: 12 12 12
PIN retry counter : 3 0 3
Signature counter : 12
Signature key ....: F844 38BB CA87 F9A7 6830  F002 F8A4 A353 3CBF CAA5
created ....: 2017-08-22 15:59:36
Encryption key....: B1D3 C9F2 C3C5 87CA 36A7  F02E E137 28E9 13B8 77E1
created ....: 2017-08-22 15:59:36
Authentication key: F87D EF02 9C38 C43D 41F0  6872 2345 A677 CE9D 8223
created ....: 2017-08-22 15:59:36
General key info..: pub  nistp256/F8A4A3533CBFCAA5 2017-08-22 cedric mesnilCedric
Mesnil <cedric@ledger>
sec>  nistp256/F8A4A3533CBFCAA5  created: 2017-08-22  expires: never
card-no: 2C97 AFB1142B
ssb>  nistp256/2345A677CE9D8223  created: 2017-08-22  expires: never
card-no: 2C97 AFB1142B
ssb>  nistp256/E13728E913B877E1  created: 2017-08-22  expires: never
card-no: 2C97 AFB1142B
```

**Say goodbye

```
gpg/card> quit**
```

At this point it's possible to check that the key has been generated on card with the following command:

```
$ gpg2 --list-secret-keys cedric@ledger
gpg: checking the trustdb

sec>  nistp256 2017-08-22 [SC]
F84438BBCA87F9A76830F002F8A4A3533CBFCAA5
Card serial no. = 2C97 AFB1142B
uid           [ultimate] cedric mesnilCedric Mesnil <cedric@ledger>
ssb>  nistp256 2017-08-22 [A]
ssb>  nistp256 2017-08-22 [E]
```

#### 0.5.1.4 Moving existing key pair

This section shows how to move an existing key onto the Ledger device.

The key to transfer here is a RSA 4096 bits key:

```
$ gpg2 --list-secret-keys "RSA 4096"
sec   rsa4096 2017-04-26 [SC]
FB6C6C75FB016635872ED3E49B93CB47F954FB53
uid           [ultimate] RSA 4096
ssb   rsa4096 2017-04-26 [E]
```

In case of transfer it is not necessary to previously set the template. It will be automatically changed. When generating a new key, the three keys (*signature*, *decryption*, *authentication*)) are automatically generated. When transferring existing ones, it is possible to choose which one will be moved.

**Edit Key**

```
$ gpg2 --edit-key "RSA 4096"
gpg (GnuPG) 2.1.19; Copyright (C) 2017 Free Software Foundation, Inc.
```

```
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Secret key is available.

sec  rsa4096/9B93CB47F954FB53
created: 2017-04-26  expires: never       usage: SC
trust: ultimate      validity: ultimate
ssb  rsa4096/49EE12B0F5CBDF26
created: 2017-04-26  expires: never       usage: E
[ultimate] (1). RSA 4096
```

**Select the key to move, here the *encryption* one.**

```
gpg> key 1

sec  rsa4096/9B93CB47F954FB53
created: 2017-04-26  expires: never       usage: SC
trust: ultimate      validity: ultimate
ssb* rsa4096/49EE12B0F5CBDF26
created: 2017-04-26  expires: never       usage: E
[ultimate] (1). RSA 4096
```

**Move**

```
gpg> keytocard
Please select where to store the key:
(2) Encryption key
Your selection? 2
```

**Unlock admin level "83"**

```
Please enter the Admin PIN

Number: 2C97 1D49B409
Holder:

Use the reader's pinpad for input.
OK
Press any key to continue.
```

**Unlock admin level "83" (maybe twice....)**

```
Please enter the Admin PIN

Number: 2C97 1D49B409
Holder:

Use the reader's pinpad for input.
OK
Press any key to continue.
```

```
sec  rsa4096/9B93CB47F954FB53
created: 2017-04-26  expires: never       usage: SC
trust: ultimate      validity: ultimate
ssb* rsa4096/49EE12B0F5CBDF26
created: 2017-04-26  expires: never       usage: E
[ultimate] (1). RSA 4096
```

**Say goobye with saving!**

```
gpg> save
```

**check**

```
$ gpg2 --edit-keys cedric
gpg: error reading key: No public key
gnuk@Lulu:~$ /opt/gnupg2.1.19/bin/gpg2 --edit-key "RSA 4096"
gpg (GnuPG) 2.1.19; Copyright (C) 2017 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Secret key is available.

sec  rsa4096/9B93CB47F954FB53
created: 2017-04-26  expires: never       usage: SC
trust: ultimate      validity: ultimate
ssb  rsa4096/49EE12B0F5CBDF26
created: 2017-04-26  expires: never       usage: E
card-no: 2C97 7BB895B9
[ultimate] (1). RSA 4096

gpg> quit
```

gpg will PINdt42 1 passphrase.

```
gpg (GnuPG) 2.1.15; Copyright (C) 2016 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
created: 2017-08-25  expires: never       usage: SC
trust: ultimate      validity: ultimate
ssb  rsa2048/8E95F2999EEC38C4
created: 2017-08-25  expires: never       usage: E
[ultimate] (1). cedric
```

**Add sub key**

```
gpg> *addkey*

Please select what kind of key you want:
(3) DSA (sign only)
(4) RSA (sign only)
(5) Elgamal (encrypt only)
(6) RSA (encrypt only)
(7) DSA (set your own capabilities)
(8) RSA (set your own capabilities)
(10) ECC (sign only)
(11) ECC (set your own capabilities)
(12) ECC (encrypt only)
(13) Existing key
Your selection? 8
```

**Toggle sign/encrypt OFF, Toggle authentication ON**

```
Possible actions for a RSA key: Sign Encrypt Authenticate
Current allowed actions: Sign Encrypt

(S) Toggle the sign capability
(E) Toggle the encrypt capability
(A) Toggle the authenticate capability
(Q) Finished

Your selection? S

Possible actions for a RSA key: Sign Encrypt Authenticate
Current allowed actions: Encrypt

(S) Toggle the sign capability
(E) Toggle the encrypt capability
(A) Toggle the authenticate capability
(Q) Finished

Your selection? E

Possible actions for a RSA key: Sign Encrypt Authenticate
Current allowed actions:

(S) Toggle the sign capability
(E) Toggle the encrypt capability
(A) Toggle the authenticate capability
(Q) Finished

Your selection? A

Possible actions for a RSA key: Sign Encrypt Authenticate
Current allowed actions: Authenticate

(S) Toggle the sign capability
(E) Toggle the encrypt capability
(A) Toggle the authenticate capability
(Q) Finished

Your selection? Q
```

**Set key options**

```
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (2048) 2048
Requested keysize is 2048 bits
Please specify how long the key should be valid.
0 = key does not expire
<n>  = key expires in n days
```

```
<n>w = key expires in n weeks
<n>m = key expires in n months
<n>y = key expires in n years
Key is valid for? (0) 0
Key does not expire at all
Is this correct? (y/N) y
Really create? (y/N) y
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.

sec  rsa2048/831415DA94A9A15C
created: 2017-08-25  expires: never       usage: SC
trust: ultimate      validity: ultimate
ssb  rsa2048/8E95F2999EEC38C4
created: 2017-08-25  expires: never       usage: E
ssb  rsa2048/C20B90E12F68F035
created: 2017-08-28  expires: never       usage: A
[ultimate] (1). cedric
```

**Select the key and move it**

```
gpg> key 2

sec  rsa2048/831415DA94A9A15C
created: 2017-08-25  expires: never       usage: SC
trust: ultimate      validity: ultimate
ssb  rsa2048/8E95F2999EEC38C4
created: 2017-08-25  expires: never       usage: E
ssb* rsa2048/C20B90E12F68F035
created: 2017-08-28  expires: never       usage: A
[ultimate] (1). cedric

gpg> keytocard
Please select where to store the key:
(3) Authentication key
Your selection? 3

sec  rsa2048/831415DA94A9A15C
created: 2017-08-25  expires: never       usage: SC
trust: ultimate      validity: ultimate
ssb  rsa2048/8E95F2999EEC38C4
created: 2017-08-25  expires: never       usage: E
ssb* rsa2048/C20B90E12F68F035
created: 2017-08-28  expires: never       usage: A
[ultimate] (1). cedric
```

**Save and Quit**

```
gpg> save
$
```

#### 0.5.2.4 Configure SSH and GPG

First, tell gpg-agent to enable ssh-auth feature by adding the following line to your .gpg-agent.conf:

```
enable-ssh-support
```

Starting with gpg2 it necessary to add some configuration options to make the *pinentry* work properly. Add the following line to ~/.bashrc file:

```
export SSH_AUTH_SOCK=`gpgconf --list-dirs agent-ssh-socket`
export GPG_TTY=`tty`
gpgconf --launch gpg-agent
```

It may be also necessary to setup the loopback pinentry options.

Add the following line to your ~/.gnupg/gpg-agent.conf:

```
allow-loopback-pinentry
```

And add the following line to your ~/.gnupg/gpg.conf:

```
pinentry-mode loopback
```

Then export your authentication public key. First execute the `gpg -k --with-subkey-fingerprint --with-keygrip cedric` command.

```
pub   rsa2048 2017-08-25 [SC]
7886147C4C2E5CE2A4B1546C831415DA94A9A15C
Keygrip = DE2B63C13AB92EBD2D05C1021A9DAA2D40ECB564
uid           [ultimate] cedric
sub   rsa2048 2017-08-25 [E]
789E56872A0D9A5AC8AF9C2F8E95F2999EEC38C4
Keygrip = 9D7C2EF8D84E3B31371A09DFD9A4B3EF72AB4ACE
sub   rsa2048 2017-08-28 [A]
2D0E4FFFAA448AA2770C7F02C20B90E12F68F035
Keygrip = 6D60CB58D9D66EE09804E7FE460E865A91F5E41A
```

Add the keygrip of the authentication key, the one identified by `[A]`, to .gnupg/sshcontrol file:

```
$ echo 6D60CB58D9D66EE09804E7FE460E865A91F5E41A > .gnupg/sshcontrol
```

Export your authentication key, identifier by its fingerprint, in a SSH compliant format.

```
$ gpg --export-ssh-key 2D0E4FFFAA448AA2770C7F02C20B90E12F68F035
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDCIARKh0IZTHld+I6oA8nwrgnCUQE8f
7X3pmI4ZwryT52fKhpcsQJsd3krodXrM//LiK8+m2ZRMneJ9iGlqqE7SCyZkNBj1GUm9s
rK3Q5eoR6nU0s+sq17b/FAtQWHBJTqqaOtyA33hFj5twUtWZ6rokX9cNZrD1ne8kRVHDe
3uEBsaY5PR1Tuko/GwywLyZu0SwfEobl/RPjL7P8rUSc7DTHpQMw8fjJFb4BNvIHAlaVC
5FwZwkuogygaJdN/44MayHFmOZmzx9CAgYgLpTzen35+CcyhlqCqi+HjNlnHL2DDWd4iR
d3Y6pY8LjS3xQkECc3Bhedptp17D+H9AVJt openpgp:0x2F68F035
```

Finaly copy the above export (`ssh-rsa AAAAB...Jt openpgp:0x2F68F035`) into the ~/.ssh/authorized_keys file on your remote server.

Now, if everything is correctly setup and running, an `ssh-add -l` should show your key:

```
$ ssh-add -l
2048 SHA256:sLCzsoi5GAG2kJkG6hSp8gTLPxSvo/zNtsks2kQ7vTU cardno:2C979421A9E1
(RSA)
2048 SHA256:sLCzsoi5GAG2kJkG6hSp8gTLPxSvo/zNtsks2kQ7vTU (none) (RSA)
```

And you should be able to ssh to your remote server with your gpg key!

### 0.5.3 Backup and Restore

#### 0.5.3.1 Introduction

"The OpenPGP card" specification does not provide any mechanism for backuping you key. Thus if you generate your keys on device and loose it, you definitively loose you private key.

In order to avoid such extreme panic situation, a backup/restore mechanism is provided. At any time you can backup a snapshot of your device data, including your private keys. All public data are retrieve in clear form. The private key are stored encrypted with a key derived from your seed, i.e. from your 24 BIP words.

The backup/restore tool is located in `pytools` directory:

```
usage: gpgcli.py [-h] [--adm-pin PIN] [--backup] [--backup-keys] [--file
FILE]
[--pinpad] [--reader READER] [--reset] [--restore]
[--set-serial SERIAL] [--set-fp SIG:DEC:AUT] [--seed-key]
[--user-pin PIN]

optional arguments:
-h, --help            show this help message and exit
--adm-pin PIN         Administrative PIN, if pinpad not used
--backup              Perfom a full backup except the key
--backup-keys         Perfom keys encrypted backup
--file FILE           basckup/restore file
--pinpad              PIN validation will be deledated to pinpad
--reader READER       PCSC reader
--reset               Reset the application. All data are erased
--restore             Perfom a full restore except the key
--set-serial SERIAL   set the four serial bytes
--set-fp SIG:DEC:AUT  sig:dec:aut fingerprints, 20 bytes each in hexa
--seed-key            Regenerate all keys, based on seed mode
--slot SLOT           slot to backup
--user-pin PIN        User PIN, if pinpad not used
```

First you must either provide your pin codes or use the pinpad (onscreen pin). This is done by giving either `--adm-pin` AND `--user-pin` or `--pinpad`. Note that using `--xx-pin` may compromise your pin codes.

Then you must precise if you want a backup or a restore with `--backup` or `--restore`

By default backup is performed without saving keys, assuming you use the seed mode. If you also want to backup keys you have to pass the `--backup-keys` option. In a general manner it is better to also save your keys with `--backup-keys` option.

Note that backup and restore works on current slot, so you have to perform a backup per slot even if some data are shared. You can precise the slot/backup to restore with `--slot`

#### 0.5.3.2 Backup and Restore example

First you must have the path of the ledger-app-openpgp-card/pytools in your PYTHONPATH.

full backup command:

```
python3 -m gpgcard.gpgcli --backup --pinpad --backup-keys --file my_bck_file_name.pickle
```

backup command without private keys:

```
python3 -m gpgcard.gpgcli --backup --pinpad --file my_bck_file_name.pickle
```

full restore command:

```
python3 -m gpgcard.gpgcli --backup --pinpad --file my_bck_file_name.pickle
```

full restore command with seed key generation:

```
python3 -m gpgcard.gpgcli --backup --pinpad --seed --file my_bck_file_name.pickle
```

#### 0.5.3.3 Restore without backup

If you have seeded key but do not have done a backup and still have your keyring, there is a solution to restore at least the key and their related information: serial and fingerprints. All other information such as name, url, ... shall be set manually with `gpg --card-edit`.

**Step 1: retrieve information**

Run the command `gpg --edit-key John`, replace John by your own key id.

```
$ gpg --edit-key John
gpg: WARNING: unsafe permissions on homedir './test/ring'
gpg (GnuPG) 2.2.4; Copyright (C) 2017 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Secret key is available.

sec  ed25519/8451AAF7D43D1095
created: 2018-10-10  expires: never       usage: SC
card-no: 2C97 FD6C11BE
trust: ultimate      validity: ultimate
ssb  ed25519/C5A8FB078520ABBB
created: 2018-10-10  expires: never       usage: A
card-no: 2C97 FD6C11BE
ssb  cv25519/0953D871FC4B9EA4
created: 2018-10-10  expires: never       usage: E
card-no: 2C97 FD6C11BE
[ultimate] (1). John Doe

gpg>
```

The `usage` field tells you each key purpose: `SC` or `S` for signature, `A` for authentication, `E` for encryption.

The `card-no`field provides you with the serial number of the card on which the key are stored. You should have three or less keys with the same serial. These are the keys we want to restore.

For each key you also have the key template (rsa2048, rsa3072, rsa4096, ed2559, cv25519) followed by the short fingerprint, e.g. `ed25519/8451AAF7D43D1095`

Note the serial and the three key template names: `FD6C11BE` , `ed25519:cv25519:ed25519`. Take care of the order: `SC:E:A`.

Now type the `quit` command.

To get the full fingerprint of each key, run (yes twice `--fingerprint`):

`gpg --fingerprint --fingerprint John`,

```
$ gpg --fingerprint --fingerprint John
gpg: WARNING: unsafe permissions on homedir './test/ring'
pub   ed25519 2018-10-10 [SC]
2C68 8345 BDDA 0EDF B24D  B4FB 8451 AAF7 D43D 1095
uid           [ultimate] John Doe
sub   ed25519 2018-10-10 [A]
CEC5 9AE6 A766 14BC 3C6D  37D9 C5A8 FB07 8520 ABBB
sub   cv25519 2018-10-10 [E]
DF15 7BD4 AC3B D1EE 9910  99C8 0953 D871 FC4B 9EA4
```

Assemble the three full fingerprint, corresponding to the one identified previously, in the the following order `SC:E:A` :

`2C688345BDDA0EDFB24DB4FB8451AAF7D43D1095:DF157BD4AC3BD1EE991099C80953D871FC4B9EA4:CEC59AE6A76614BC3C6D37D9C5A8FB078520ABBB`.

**Step 1: restore**

Plug you Nano S and run the OpenPGP application.

Finally run the following command :

```
python3 -m gpgcard.gpgcli --pinpad  --set-template ed255519:cv25519:ed255519
--set-fingerprints
```

```
'2C688345BDDA0EDFB24DB4FB8451AAF7D43D1095:DF157BD4AC3BD1EE991099C80953D871FC4B9EA4:CEC59AE6A766
--set-serial 'FD6C11BE' --seed
```

### 0.5.4 Trouble/FAQ

**Q:** pinentry failed with a strange canceled message:

**R:** there is some problem with gpg2 and pinentry-gnome3. You may update your system to use pinentry-gtk-2. Under Ubuntu-like OS, use `update-alternatives --config pinentry`

**Q:** gpg-connection agent failed

**R:** check that you don't have multiple running agents. After setting-up all SSH stuff, try to fully logout/login

**Q:** It does not work at all, HELP ME!!!

**R** Please keep calm and do not cry. Add the following option to ~/.gnupg/gpg-agent.conf

```
debug-level guru
log-file /tmp/gpgagent.log
```

Add the follwing option to ~/.gnupg/scdaemon.conf

```
log-file /tmp/scd.log
debug-level guru
debug-all
```

Make a nice issue report under github providing log and and command line you run.

**!*WARNING*!** : this may reveal confidential information such as key values. Do your log with a test key.

**Q:** I'm having issue when using SSH, there is no pinpad prompt either on my host nor my Nano (`sign_and_send_pubkey: signing failed: agent refused operation`)

**R:** You might need to add this command to your .bashrc/.zshrc :

```
gpg-connect-agent updatestartuptty /bye >/dev/null
```

Be aware that when using **Host** PIN mode, you will have to enter your PIN directly on your computer and if you use a ncurses-like PIN entry program. In some cases, you will be prompted to the first shell that uses the above command (at least on Mac).

**Q:** My mac is not able to see my `Ledger Token`

**R:** For some reason, SC communication on Mac takes some times or mess it up sometimes.

To troubleshot those issues, you can try to reload the `scdaemon` using this command :

```
gpgconf --reload scdaemon
gpgconf --reload gpg-agent
```

If not successful, you can try to trigger daemons to restart by sending a **SIGTERM** like so :

```
kill -TERM $(pgrep gpg-agent) $(pgrep scdaemon).
```

Changing USB port might also help sometimes. Do not hesitate.

**Q:** My mac is **STILL* not able to see my `Ledger Token`

**R:** This might be related to your CCID drivers. Mojave comes with the version `1.4.27` pre-installed. You can manually install a more recent version from this website<https://ccid.apdu.fr/files/> and install it this way :

```
CCID_VERSION=1.4.30
wget https://ccid.apdu.fr/files/ccid-${CCID_VERSION}.tar.bz2
tar xzvf ccid-${CCID_VERSION}.tar.bz2
cd ccid-${CCID_VERSION}
./MacOSX/configure
make
make install
```

Installing the driver depends on `libusb` which can be installed using the following `brew install libusb`. It also requires static linking against it, if you use dynamic linking you will have the following output when using the `./MacOSX/configure` step :

```
/usr/local/Cellar/libusb/1.0.23/lib/libusb-1.0.0.dylib
/usr/local/Cellar/libusb/1.0.23/lib/libusb-1.0.dylib
*****************************
Dynamic library libusb found in /usr/local/Cellar/libusb/1.0.23/lib
*****************************
Rename it to force a static link
```

You can use the following :

```
LIBUSB_VERSION=1.0.23

for f in /usr/local/Cellar/libusb/${LIBUSB_VERSION}/lib/*.dylib; do
  mv $f $f.fake
done

./MacOSX/configure

for f in /usr/local/Cellar/libusb/${LIBUSB_VERSION}/lib/*.dylib.fake; do
  ORIG="$( echo $f | sed 's#.fake##g' )"
  mv $f ${ORIG}
done
```

Once installed, you should see the new driver installed using this command :

```
SmartCards:

  Readers:

  Reader Drivers:

   #01: org.debian.alioth.pcsclite.smartcardccid:1.4.27
    (/usr/libexec/SmartCardServices/drivers/ifd-ccid.bundle)
   #02: org.debian.alioth.pcsclite.smartcardccid:1.4.30
    (/usr/local/libexec/SmartCardServices/drivers/ifd-ccid.bundle)

  Tokend Drivers:

  SmartCard Drivers:

   #01: com.apple.CryptoTokenKit.pivtoken:1.0
    (/System/Library/Frameworks/CryptoTokenKit.framework/PlugIns/pivtoken.appex)

  Available SmartCards (keychain):
```

```
Available SmartCards (token):
```

## 0.6 Annexes

### 0.6.1 References


- [GPG] *The GNU Privacy Guard*, https://gnupg.org/
- [GPGSC] *The GnuPG Smartcard HOWTO*, https://gnupg.org/howtos/card-howto/en/smartcard-howto.html
- [G10CODE] *The OpenPGP card application*, https://g10code.com/p-card.html
- [GPGADD] *The OpenPGP card application add-on*, https://github.com/LedgerHQ/blue-app-openpgp-card/blob/master/doc/gpgcard3.0-addon.rst