THALES

GnuPG: Integration Guide

THALES LUNA HSM AND LUNA CLOUD HSM

Document Information

Document Part Number	007-013996-001
Revision	C
Release Date	19 October 2021

Trademarks, Copyrights, and Third-Party Software

CONTENTS

Overview

GnuPG is an implementation of PGP (Pretty Good Privacy), which is a form of public key/private key encryption. The strength of the encryption comes from the fact that a file can be encrypted for a given recipient using only the public key, yet both keys are needed in order to decrypt the file. So the idea is to give your public key to your friends and colleagues, but keep the private key closely guarded. The private key is additionally stored on Thales Luna HSM that adds an additional level of security to prevent someone using your private key if they gain physical access to both your computer and account.

This guide describes the steps involved in integrating Thales Luna HSM and Thales Luna Cloud HSM with GnuPG. The benefits of securing the private key with Luna HSM includes:

- Secure generation, storage, and protection of the signing private key on FIPS 140-2 level 3 validated hardware.
- Full life cycle management of the keys.
- Access to the HSM audit trail*.
- The advantage of cloud services with confidence.

*Luna Cloud HSM services do not have access to the secure audit trail.

Certified Platforms

Certified platforms for Luna HSM
Certified platforms for Luna Cloud HSM

Certified platforms for Luna HSM

The following platforms are certified for integrating GnuPG with Luna HSM:

HSM Type	GnuPG Version	PKCS11-SCD-Daemon	Platforms Certified
Luna HSM	2.2.31	0.9.2	Red Hat Enterprise Linux 8.2 (64 bit) Ubuntu 16.04
Luna HSM	2.0.22	0.9.1	Red Hat Enterprise Linux 7.7 (64 bit) Red Hat Enterprise Linux 7.0 (64 bit)
Luna HSM	2.0.14	0.9.1	Red Hat Enterprise Linux 6.5 (64 bit)

NOTE: GnuPG Integration is tested in HA as well as FIPS mode.

Luna HSM: Luna HSM appliances are purposefully designed to provide a balance of security, high performance, and usability that makes them an ideal choice for enterprise, financial, and government organizations. Luna HSMs physically and logically secure cryptographic keys and accelerate cryptographic processing. Luna HSM on premise offerings include the Luna Network HSM, Luna PCIe HSM, and Luna USB HSMs. Luna HSMs are also available for access as an offering from cloud service providers such as IBM cloud HSM and AWS cloud HSM classic.

Certified platforms for Luna Cloud HSM

The following platforms are certified for integrating GnuPG with Luna Cloud HSM:

HSM Type	GnuPG Version	PKCS11-SCD-Daemon	Platforms Certified
Luna Cloud HSM	2.2.31	0.9.2	Red Hat Enterprise Linux 8.2 (64 bit)
Luna Cloud HSM	2.0.22	0.9.1	Red Hat Enterprise Linux 7.7 (64 bit) Red Hat Enterprise Linux 7.0 (64 bit)

Luna Cloud HSM: Luna Cloud HSM provides on-demand HSM and Key Management services through a simple graphical user interface. With Luna Cloud HSM, security is simple, cost effective and easy to manage because there is no hardware to buy, deploy and maintain. As an Application Owner, you click and deploy services, generate usage reports and maintain just the services you need.

Prerequisites

Before you proceed with the integration, complete the following tasks:

Configure Luna HSM

Configure Luna Cloud HSM service

Installing GPG-dependent packages

Installing Pinentry package

Installing GnuPG packages

Installing gnupg-pkcs11-scd smart-card daemon and pkcs11-helper

Configure Luna HSM

To configure Luna HSM:

1. Ensure that the HSM is set up, initialized, provisioned, and ready for deployment. Refer to Luna HSM documentation for help.
2. Create a partition that will be later used by GnuPG.
3. Create and exchange certificate between the Luna Network HSM and Client system. Register client and assign partition to create an NTLS connection. Initialize Crypto Officer and Crypto User roles for the registered partition.
4. Ensure that the partition is successfully registered and configured. The command to see the registered partitions is:

```
# /usr/safenet/lunaclient/bin/lunacm

lunacm (64-bit) v10.3.0-275. Copyright (c) 2020 SafeNet. All rights
reserved.

Available HSMs:

Slot Id ->              0

Label ->                INTG_01
```

```
Serial Number ->          1213475834492
Model ->                  LunaSA 7.7.0
Firmware Version ->       7.7.0
Bootloader Version ->     1.1.2
Configuration ->          Luna User Partition With SO (PW) Signing With
Cloning Mode
Slot Description ->       Net Token Slot
FM HW Status ->           FM Ready
```

5. For PED-authenticated HSM, enable partition policies 22 and 23 to allow activation and auto-activation.

NOTE: Refer to Luna HSM documentation for detailed steps about creating NTLS connection, initializing the partitions, and assigning various user roles.

NOTE: For PED-based Luna HSM ,ensure that ProtectedAuthenticationPathFlagStatus is set to '1' in the Misc Section of Chrystoki.conf file.

Set up Luna HSM High-Availability

Refer to Luna HSM documentation for HA steps and details regarding configuring and setting up two or more HSM boxes on host systems. You must enable the HAOnly setting in HA for failover to work so that if the primary goes down due to any reason all calls automatically route to the secondary until the primary recovers and starts up.

Set up Luna HSM in FIPS Mode

Under FIPS 186-3/4, the RSA methods permitted for generating keys are 186-3 with primes and 186-3 with aux primes. This means that RSA PKCS and X9.31 key generation is no longer approved for operation in a FIPS-compliant HSM. If you are using the Luna HSM in FIPS mode, you have to make the following change in the configuration file:

```
[Misc]
RSAKeyGenMechRemap=1
```

The above setting redirects the older calling mechanism to a new approved mechanism when Luna HSM is in FIPS mode.

NOTE: The above setting is not required for Universal Client. This setting is applicable only for Luna Client 7.x.

Control User Access to the HSM

By default, only the root user has access to the HSM. You can specify a set of non-root users that are permitted to access the HSM by adding them to the hsmusers group. The client software installation automatically creates the hsmusers group. The hsmusers group is retained when you uninstall the client software. This allows you to upgrade your client software while retaining your hsmusers group configuration.

To add users to hsmusers group

To allow non-root users or applications access to the HSM, assign the users to the hsmusers group. The users you have assigned to the hsmusers group must exist on the client workstation. The HSM can be accessed only by the users whom you have added to the hsmusers group. To add a user to the hsmusers group:

a. Ensure that you have sudo privileges on the client workstation.

b. Add a user to the hsmusers group.

```
sudo gpasswd --add <username> hsmusers
```

where <username> is the name of the user you want to add to the hsmusers group.

To remove users from hsmusers group

To revoke a user's access to the HSM, you can remove them from the hsmusers group. To remove a user from the hsmusers group:

a. Ensure that you have sudo privileges on the client workstation.

b. Remove a user from the hsmusers group.

```
sudo gpasswd -d <username> hsmusers
```

where `<username>` is the name of the user you want to remove from the hsmusers group. To see the change, you need to log in again.

> **NOTE:** The user you delete will continue to have access to the HSM until you reboot the client workstation.

Configure Luna Cloud HSM service

You can configure Luna Cloud HSM Service in the following ways:

> Standalone Cloud HSM service using minimum client package

> Standalone Cloud HSM service using full Luna client package

> Luna HSM and Luna Cloud HSM service in hybrid mode

> **NOTE:** Luna Client v10.x or higher is required for configuring Luna HSM device and Luna Cloud HSM service in hybrid mode.

Standalone Cloud HSM service using minimum client package

To configure Luna Cloud HSM service using minimum client package:

1. Transfer the downloaded .zip file to your client workstation using pscp, scp, or other secure means.
2. Extract the .zip file into a directory on your client workstation.
3. Extract or untar the appropriate client package for your operating system. Do not extract to a new subdirectory; place the files in the client install directory.

   `[Linux]`

```
cvclient-min.tar
# tar -xvf cvclient-min.tar
```

4. Run the setenv script to create a new configuration file containing information required by the Luna Cloud HSM service.

```
[Linux]
Source the setenv script.
# source ./setenv
```

5. Run the LunaCM utility and verify the Cloud HSM service is listed.

Standalone Cloud HSM service using full Luna client package

To configure Luna Cloud HSM service using full Luna client package:

1. Transfer the downloaded .zip file to your Client workstation using pscp, scp, or other secure means.
2. Extract the .zip file into a directory on your client workstation.
3. Extract or untar the appropriate client package for your operating system. Do not extract to a new subdirectory; place the files in the client install directory.

```
[Linux]
cvclient-min.tar
# tar -xvf cvclient-min.tar
```

4. Run the setenv script to create a new configuration file containing information required by the Luna Cloud HSM service.

```
[Linux]
Source the setenv script.
# source ./setenv
```

5. Copy the server and partition certificates from the Cloud HSM service client directory to Luna client certificates directory:

> **NOTE:** Skip this step for Luna Client v10.2 or higher.

Cloud HSM Certificates:

```
server-certificate.pem
partition-ca-certificate.pem
partition-certificate.pem
```

LunaClient Certificate Directory:

```
[Linux default location for Luna Client]
/usr/safenet/lunaclient/cert/
```

6. Open the configuration file from the Cloud HSM service client directory and copy the XTC and REST section.

```
[Linux]
Chrystoki.conf
```

7. Edit the Luna Client configuration file and add the XTC and REST sections copied from Cloud HSM service client configuration file.
8. Change server and partition certificates path from step 5 in XTC and REST sections. Do not change any other entries provided in these sections.

   NOTE: Skip this step for Luna Client v10.2 or higher.

   [XTC]

```
. . .
PartitionCAPath=<LunaClient_cert_directory>\partition-ca-certificate.pem
PartitionCertPath00=<LunaClient_cert_directory>\partition-certificate.pem
. . .

[REST]
. . .
SSLClientSideVerifyFile=<LunaClient_cert_directory>\server-certificate.pem
. . .
```

9. Edit the following entry from the Misc section and update the correct path for the plugins directory:

```
Misc]
PluginModuleDir=<LunaClient_plugins_directory>

[Linux Default]
/usr/safenet/lunaclient/plugins/
```

   Save the configuration file. If you wish, you can now safely delete the extracted Cloud HSM service client directory.

10. Reset the ChrystokiConfigurationPath environment variable and point back to the location of the Luna Client configuration file.

    Either open a new shell session, or export the environment variable for the current session pointing to the location of the Chrystoki.conf file:

```
# export ChrystokiConfigurationPath=/etc/
```

11. Run the LunaCM utility and verify that the Cloud HSM service is listed. In hybrid mode, both Luna and Cloud HSM service will be listed.

    NOTE: Follow the Luna Cloud HSM documentation for detailed steps for creating service, client, and initializing various user roles.

Luna HSM and Luna Cloud HSM service in hybrid mode

To configure Luna HSM and Luna Cloud HSM service in hybrid mode, follow the steps mentioned under the Standalone Cloud HSM service using full Luna client package section above.

NOTE: Luna Client v10.x or higher is required for configuring Luna HSM device and Luna Cloud HSM service in hybrid mode.

To use Luna Cloud HSM Service in FIPS mode

Cloud HSM service operates in both FIPS and non-FIPS mode. If your organization requires non-FIPS algorithms for your operations, enable the Allow non-FIPS approved algorithms check box when configuring your Cloud HSM service. The FIPS mode is enabled by default. Refer to the Mechanism List in the SDK Reference Guide for more information about available FIPS and non-FIPS algorithms.

Installing GPG-dependent packages

Install the following GPG-dependent packages from https://www.gnupg.org/download/index.html before you begin the integration process:

- npth
- libgpg-error
- libgcrypt
- libksba
- libassuan

Installing Pinentry package

To authenticate partition access for GPG, you can use the **salogin** utility, which gets automatically installed along with the Thales Luna Client software. However, if you do not want to use the **salogin** utility, you can install the **Pinentry** package available at https://www.gnupg.org/download/index.html.

Installing GnuPG package

After building and installing the above packages, you need to install the **GPG** package available at https://www.gnupg.org/download/index.html

Installing gnupg-pkcs11-scd smart-card daemon and pkcs11-helper

After installing GPG, you need to install **pkcs11-helper** and **gnupg-pkcs11-scd** and libraries.

> **NOTE:** While building the gnupg-pkcs11-scd daemon, the development packages associated with these libraries are subsequently used at runtime. To keep the new library versions separate from the versions that are already installed, run the **export LD_LIBRARY_PATH=/usr/local/lib** command**.**

- pkcs11-helper (https://github.com/OpenSC/pkcs11-helper/releases)
- gnupg-pkcs11-scd (https://github.com/alonbl/gnupg-pkcs11-scd/releases/)

Integrating GnuPG with Luna HSM

This chapter covers the following topics:

Accessing Luna HSM

You can use either of the following methods to access Luna HSM for GPG:

Using salogin utility

The persistent session allows the GPG to access the HSM object without prompting the password every time.

> **NOTE:** Persistent Session does not supported for Luna Cloud HSM. For Luna Cloud HSM, refer Using Pinentry section.

To open the persistent session using **salogin** utility, perform the following steps:

1. Add the following text in the **/etc/Chrystoki.conf** file:

```
Misc = {
   AppIdMajor=1;
   AppIdMinor=1;
}
```

2. Run the following command to open the authenticated persistent session to access the HSM object:

```
# ./salogin -o -s 0 -i 1:1 -p <partition_password>
```

   Where `-s` represent the slot_id and `-i` represent the AppId set in the **Chrystoki.conf** file.

Using Pinentry

To use Pinentry, you need to add the following text in the **~/.gnupg/gpg-agent.conf** file.

```
pinentry-program /usr/local/bin/pinentry
```

> **NOTE:** If **gpg-agent.conf file** doesn't exist, you need to create it at **~/.gnupg/** directory.

Configuring the gnupg-pkcs11-scd.conf file

> **NOTE:** This step is required only when using Pinentry with GPG v2.0.x. Skip this step for GPG v2.2.x

To configure the gnupg-pkcs11-scd.conf file, add/modify/uncomment the following lines to **~/.gnupg/gnupg-pkcs11-scd.conf** file:

```
provider-p1-allow-protected-auth

provider-p1-cert-private

provider-p1-private-mask 0
```

> **NOTE:** If **gnupg-pkcs11-scd.conf** file doesn't exist, copy **/usr/local/etc/gnupg-pkcs11-scd.conf.example** or **/usr/local/share/doc/gnupg-pkcs11-scd/gnupg-pkcs11-scd.conf.example** file in **~/.gnupg** directory and rename it.

Generating keys and certificates

After creating the NTLS connection with HSM, follow the below steps to generate the RSA key pair on HSM. GPG uses several asymmetric key pairs as part of the keychain configuration. These are signing, encryption, and authentication key pairs. You can use the same key pair for all three functions. To generate keys and certificates:

1. Generate the RSA key pair on Luna HSM using the CMU utility provided with Luna Client in /usr/safenet/lunaclient/bin directory. Provide the partition password when prompted and select RSA Mechanism Type.

   > **NOTE:** CMU command options might be slightly differ in other versions of Luna Client, kindly refer to the Luna SA documentation for exact options.

   ```
   # /usr/safenet/lunaclient/bin/cmu generatekeypair -modulusBits=2048
   -publicExponent=65537 -labelPublic=GPG-Sign-Pub –labelPrivate=GPG-Sign-Priv
   –sign=1 –verify=1 -encrypt=1 -decrypt=1 -wrap=1 -unwrap=1
   -id=c50f7b86372b441ba77cb6f8598f1e35

   Please enter password for token in slot 0 : ********

   Select RSA Mechanism Type -

   [1] PKCS [2] FIPS 186-3 Only Primes [3] FIPS 186-3 Auxiliary Primes : 1

   Select RSA Mechanism Type PKCS
   ```

NOTE: You can use one of the following Linux commands to generate 32 byte Hex value and use it as key ID:

```
# head -c16 </dev/urandom|xxd -p -u
# xxd -len 16 -plain /dev/urandom
```

2. List the contents generated on HSM partition and note down the handle of public/private key. Provide the partition password when prompted.

```
# /usr/safenet/lunaclient/bin/cmu list
Please enter password for token in slot 0 : ********
handle=34       label=GPG-Sign-Pub
handle=35       label=GPG-Sign-Priv
```

3. Generate the self-signed certificate from the generated public/private key. Provide the partition password and certificate attributes when prompted.

NOTE: For Luna Cloud HSM, replace publichandle and privatehandle with publicouid and privateouid, respectively.

```
# /usr/safenet/lunaclient/bin/cmu selfsigncertificate -label=GPG-Sign
-publichandle=34 -privatehandle=35 -startDate=20210925 -endDate=20220925
-serialNumber=0133337A -keyusage=digitalsignature,keyencipherment
-id=c50f7b86372b441ba77cb6f8598f1e35
Please enter password for token in slot 0 : ********
Enter Subject 2-letter Country Code (C) : IN
Enter Subject State or Province Name (S) : UPST
Enter Subject Locality Name (L) : NOIDA
Enter Subject Organization Name (O) : THALES
Enter Subject Organization Unit Name (OU) : HSM Integration
Enter Subject Common Name (CN) : GPG-Signing
Enter EMAIL Address (E) :
```

NOTE: Ensure that the id must be same for both keys and certificate. Self-signed certificate is used for test purposes. For production environment, create the certificate request and signed it by the Trusted Certificate Authority.

4. If you want to use different keys and certificate for Encryption and Authentication, repeat steps 1-3.

NOTE: Ensure that the id and label for every key/certificate are different.

Configuring GPG to use the PKCS#11 smart card daemon

Perform the following steps to configure the gpg-agent that uses the smart card daemon to access the keys on HSM:

1. Add the following line to **~/.gnupg/gpg-agent.conf** file.

   ```
   scdaemon-program /usr/local/bin/gnupg-pkcs11-scd
   ```

   > **NOTE:** if ~**/.gnupg/gpg-agent.conf** file is not present, then create the file and add the scdaemon-program.

2. Add/modify the following lines to **~/.gnupg/gnupg-pkcs11-scd.conf** file.

   ```
   providers p1

   provider-p1-library /usr/safenet/lunaclient/lib/libCryptoki2_64.so
   ```

   > **NOTE:** If **gnupg-pkcs11-scd.conf** file doesn't exist, copy **/usr/local/etc/gnupg-pkcs11-scd.conf.example** or **/usr/local/share/doc/gnupg-pkcs11-scd/gnupg-pkcs11-scd.conf.example** file in **~/.gnupg** directory and rename it. Ensure that the library path is correct where you extracted the client, when using Luna Cloud HSM.

3. Set the following environment variables to make sure you are using the installed GPG version.

   ```
   # export LD_LIBRARY_PATH=/usr/local/lib:$LD_LIBRARY_PATH

   # export PATH=/usr/local/bin:$PATH
   ```

For GPG v2.2.x

1. Use the following command to enable GPG to discover all useful information of the card (or HSM partition in this case).

   ```
   # gpg --card-status
   ```

2. Execute the following command to connect the agent to HSM and get the keys from HSM.

   ```
   # gpg-agent --server gpg-connect-agent
   ```

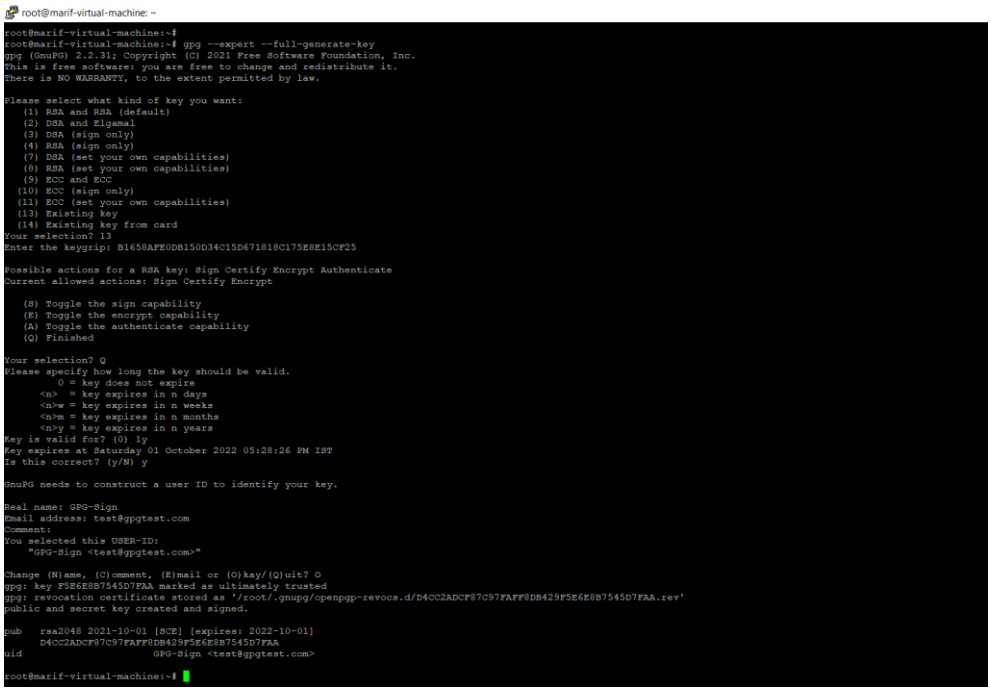
3. At the prompt, enter SCD LEARN. The Pinentry program pop ups and prompts for the partition password. The output of the command will be similar to what is depicted in the image below.

Look for the line **S KEY-FRIENDLY**, and note the 20 byte hash that you need to provide in next steps as **keygrip** when prompted.

4. Execute the following command to generate the GPG virtual keys. Note that the keys are not actually generated on the localhost but only a reference to the HSM keys is generated by the GPG.

```
# gpg --expert --full-generate-key
```

When executed, you'll be prompted to select the few options to generate the key. Follow the prompt and select Existing Keys for key type to be generated, purpose of the key and other values like Real Name, Email Address, and validity as per your choice.
The Real Name provided here will be used to refer the key during RPM signing.



5. After generating the keys, you can list the keys using the following command.

```
# gpg --list-keys
```

```
root@marif-virtual-machine:~# gpg --list-keys
gpg: checking the trustdb
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: depth: 0  valid:   2  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 2u
gpg: next trustdb check due at 2022-09-29
/root/.gnupg/pubring.kbx
------------------------
pub   rsa2048 2021-09-29 [SCE] [expires: 2022-09-29]
      635964843EB2357AB7B96BD1AD05DBB378832D9B
uid           [ultimate] marif <marif@localhost.localdomain>

pub   rsa2048 2021-10-01 [SCE] [expires: 2022-10-01]
      D4CC2ADCF87C97FAFF8DB429F5E6E8B7545D7FAA
uid           [ultimate] GPG-Sign <test@gpgtest.com>

root@marif-virtual-machine:~#
```

For GPG v2.0.x

1. Execute the following command to connect the agent to HSM and get the keys from HSM:

   ```
   # gpg-agent --server gpg-connect-agent
   ```

2. At the prompt, enter **SCD LEARN**. The pinentry program pop ups and prompts for the partition password. The output of the command will be similar to what the following image demonstrates:

> **NOTE:** If you open the persistent session via **salogin,** the password prompt will not appear.

3. Look for the line **S KEY-FRIENDLY**, identify the signing/encryption/authentication certificate by the appropriate Common name (CN), and copy the 20 byte SHA-1 hash in the **gnupg-pkcs11-scd.conf** file as follows:

   ```
   openpgp-sign 8C5CE31F726FE84CBB0891E0E2816F2EF07F0000

   openpgp-encr 7990A0D320B59A0DA525CE39D15398743762EFBB

   openpgp-auth 8B91705A7B3ED221AAFF5E78B95C89DD4EB0DDCD
   ```

> **NOTE:** 20 byte Hash would be the same if you are using the same key for each function.

4. Use the following command to enable GPG to discover all useful information of the card (or HSM partition in this case):

```
# gpg --card-status
```

```
[root@localhost ~]# /usr/bin/gpg --card-status
Application ID ...: D2760001240111503131798 8A0061111
Version ..........: 11.50
Manufacturer .....: unknown
Serial number ....: 7988A006
Name of cardholder: [not set]
Language prefs ...: [not set]
Sex ..............: unspecified
URL of public key : [not set]
Login data .......: [not set]
Signature PIN ....: forced
Key attributes ...: 1R 1R 1R
Max. PIN lengths .: 0 0 0
PIN retry counter : 0 0 0
Signature counter : 0
Signature key ....: 8C5C E31F 726F E84C BB08  91E0 E281 6F2E F07F 0000
Encryption key....: 7990 A0D3 20B5 9A0D A525  CE39 D153 9874 3762 EFBB
Authentication key: 8B91 705A 7B3E D221 AAFF  5E78 B95C 89DD 4EB0 DDCD
General key info..: [none]
[root@localhost ~]#
```

5. Execute the following commands to generate the GPG virtual keys. Note that the keys are not actually generated on the localhost but only a reference to the HSM keys is generated by the GPG.

```
# gpg --card-edit
```

```
# Command> admin
```

```
# Command> generate
```

You need to provide the following inputs:

a. Respond “y” to replace the existing keys?

b. Do not back up keys if prompted.

c. Set the expiry parameters.

d. Provide the key name when prompted for real name.

> **Note**: The value provided in Real name will be used to refer the key for RPM signing going forward.

Testing GPG with Luna HSM

You can test GPG in the following use-cases:

> File Signing and Verification

> RPM Signing and Verification

File Signing and Verification

To sign and verify a file, perform the following steps:

1. Run the following command and replace <Your key name> with real name of the key and somefile with actual file name.

   ```
   # gpg --sign --default-key <Your key name> somefile
   ```

   Provide the partition password when prompted. After signing is completed, a file **somefile.gpg** will be created containing the original file contents and signature.

   > **NOTE:** If you open persistent session via **salogin**, you will not be asked to provide the password.

2. The original file can be verified and recovered by running the following command.

   ```
   # gpg somefile.gpg
   ```

   Contents of the original file and recovered file will be the same.

RPM signing and verification

RPM Signing uses GPG under the hood for key management and crypto functions. Some GPG options can be invoked via the RPM command line, but some experimentation may be required in order to get the desired signature options and attributes. Follow the below sections sequentially to sign and verify the rpms.

Environment Settings

1. At a minimum, the following environment setting may be required in order for RPM (via GPG) to call the Pinentry program:

   > **NOTE:** This step is required only when using Pinentry with GPG v2.0.x. Skip this step for GPG v2.2.x

   ```
   # GPG_TTY=$(tty)
   # export GPG_TTY
   ```

2. Create/Update the file **`~/.rpmmacros`** in order to utilize the key. The contents of the file would be similar to what the following image depicts:

   ```
   %_signature gpg
   %_gpg_path /root/.gnupg
   %_gpg_name GPG-Sign
   ```

```
%__gpg /usr/local/bin/gpg

%__gpg_sign_cmd %{__gpg} gpg --force-v3-sigs --batch --verbose --no-armor
--no-secmem-warning -u "%{_gpg_name}" -sbo %{__signature_filename} --
digest-algo sha256 %{__plaintext_filename}'
```

Where:

- signature must be gpg
- gpg_path refers the gnupg directory path
- gpg_name refer to the key Real Name
- gpg refers the gpg binary/executable
- gpg_sign_cmd value should be same as provided that is used to force the SHA256 signature

Signing an RPM

You can sign the RPM files using the rpm utility or you can use the script provided below to automate the signing process.

Using RPM Command:

```
# rpm --addsign example.rpm
```

Where example.rpm is the RPM file to be signed. You can specify --resign flag, if RPM file need to be re-signed.

```
# rpm --resign example.rpm
```

Using Script:

> **NOTE:** Ensure that your system has expect installed, if not, install the expect using the command # **yum install expect expect**. Expect is required to run the script.

A script is required to demonstrate some of the common options that contribute to the signing operation and signature format. Create the script from the source listing, modify the script to reference the appropriate key name, and then execute the following command:

```
./<scriptname> <your_rpm>
```

An example of the script is provided below. Copy and paste the below snippet in a file called **rpmsigntest**:

```
#!/usr/bin/expect --
spawn rpm --define "_gpg_name  GPG-Sign" --resign {*}$argv
expect {
"Enter pass phrase:" { send "\r" ; exp_continue }
eof
}
```

Where GPG-Sign is your signing key Real name you set earlier.

Now run the script to sign the RPM, for example:

```
# ./rpmsigntest example.rpm
spawn rpm --define _gpg_name  GPG-Sign --resign example.rpm
Enter pass phrase:
Pass phrase is good.
example.rpm:
```

The Pinentry program pop ups and prompts for the partition password. However, if the persistent session is opened via SALOGIN or Pin is already cached, then there will be no prompt for password. Each RPM signing involves three separate signing operations and you receive prompt for the partition password each time.

Verify a signed RPM

In order to verify the signature on the RPM file, the public key associated with the signing key needs to be imported into the GPG keychain.

1. Export the public key using the command:

   ```
   # gpg --export --armor <your_keyname> > <your_keyfile>
   ```

   For example:

   ```
   # gpg --export --armor GPG-Sign > gpg.key
   ```

2. Import the public key using the command:

   ```
   # rpm --import <your_keyfile>
   ```

   For example:

   ```
   # rpm --import gpg.key
   ```

3. Verify the signed RPM

   ```
   # rpm --checksig <your_rpm>
   ```

   For example:

   ```
   # rpm --checksig example.rpm
   ```

   ```
   example.rpm: rsa sha1 (md5) pgp md5 OK
   ```

   `rpm` still shows generic output because `SHA1` (and `MD5`) digests are an integral part of `rpm` packages. Use below command to confirm the sign algorithm.

   ```
   # rpm -q --qf '%{SIGPGP:pgpsig} %{SIGGPG:pgpsig}\n' -p example.rpm
   ```

   ```
   RSA/SHA256, Thursday 30 September 2021 01:50:59 AM IST, Key ID
   ad05dbb378832d9b (none)
   ```

Unattended RPM signing

You may need to perform the RPM signing as part of the product build and RPM creation processes. Thus it would be impractical for someone to enter the partition password for each RPM signature. To perform unattended rpm signing:

1. Start the gpg-agent daemon using the following command:

   ```
   # gpg-agent --daemon
   ```

   NOTE: For additional options, please refer to gpg-agent man pages. If the gpg-agent is started in daemon mode, it will remain in the background and the partition password will be cached for subsequent use.

2. Create a simple expect script. Create a file called rpm-sign.exp containing the following snippet.

   ```
   #!/usr/bin/expect --
   spawn rpm --define "_gpg_name <your keyname here>" --resign
      {*}$argv
   expect {
   "Enter pass phrase:" { send "\r" ; exp_continue }
   eof
   }
   ```

   NOTE: Expect is an extension to the tcl scripting language and is a program used to automate interactions with other applications that expose a text terminal interface. The easiest way to install it on a RedHat/CentOS system is with yum, as follows:

   # yum install expect expectk

3. Sign the rpm using the created script file.

   ```
   # ./rpm-sign.exp <rpm_name>
   ```

This completes the integration of Luna HSM with GnuPG (GPG), which secures the crypto keys on FIPS 140-3 compliant hardware device. Crypto keys secured on HSM partition are available for signing when required through GnuPG.

Contacting Customer Support

If you encounter a problem during this integration, contact your supplier or Thales Customer Support. Thales Customer Support operates 24 hours a day, 7 days a week. Your level of access to this service is governed by the support plan arrangements made between Thales and your organization. Please consult this support plan for further information about your entitlements, including the hours when telephone support is available to you.

Customer Support Portal

The Customer Support Portal, at https://supportportal.thalesgroup.com, is a database where you can find solutions for most common problems. The Customer Support Portal is a comprehensive, fully searchable repository of support resources, including software and firmware downloads, release notes listing known problems and workarounds, a knowledge base, FAQs, product documentation, technical notes, and more. You can also use the portal to create and manage support cases.

> **NOTE:** You require an account to access the Customer Support Portal. To create a new account, go to the portal and click on the **REGISTER** link.

Telephone Support

If you have an urgent problem, or cannot access the Customer Support Portal, you can contact Thales Customer Support by telephone at +1 410-931-7520. Additional local telephone support numbers are listed on the support portal.

Email Support

You can also contact technical support by email at technical.support.DIS@thalesgroup.com.