GnuPG: Integration Guide

THALES LUNA HSM AND LUNA CLOUD HSM

Document Information

Document Part Number	007-013996-001
Revision	D
Release Date	29 May 2023

Trademarks, Copyrights, and Third-Party Software

CONTENTS

Overview

Thales Luna HSM plays a crucial role in enhancing the security of GnuPG, an implementation of PGP (Pretty Good Privacy), which utilizes public key/private key encryption. The encryption strength lies in the fact that a file can be encrypted using only the recipient's public key, while both keys are required for decryption. This approach involves distributing the public key to friends and colleagues, while safeguarding the private key.

To further fortify the protection of the private key, Thales Luna HSM is employed. By storing the private key within the Thales Luna HSM, an additional layer of security is added. This ensures that even if an adversary gains physical access to both the computer and the account, they would be unable to utilize the private key without the appropriate access controls provided by the HSM. The benefits that can be attained by securing the private key with Luna HSM include:

> Ensuring secure key generation, storage, and protection through FIPS 140-2 level 3 validated hardware.

> Providing full life cycle management of the keys.

> Maintaining an audit trail through HSM.

> Achieving significant performance enhancements by offloading cryptographic operations from application servers.

Note: The Luna Cloud HSM service does 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 Luna HSM 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: The 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 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 Entrust 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

> Install GPG-dependent packages

> Install Pinentry package

> Install GnuPG packages

> Install gnupg-pkcs11-scd smart-card daemon and pkcs11-helper

Configure Luna HSM

To configure Luna HSM with GnuPG:

1. Set up and initialize the Luna HSM according to the instructions provided in the Luna HSM documentation. Ensure that the HSM is provisioned and ready for deployment.
2. Create a partition within the Luna HSM that will be used by GnuPG. This partition will contain the private key associated with GnuPG.
3. Generate a certificate for the Luna Network HSM and exchange it between the Luna HSM and the client system. This certificate is required to establish a secure connection (NTLS) between the Luna HSM and the client system. During this process, register the client system and assign the previously created partition to establish a connection. Assign the Crypto Officer and Crypto User roles to the registered partition. These roles define the access privileges and permissions for managing cryptographic operations within the partition.

4. Ensure that the registered partition and its associated configuration are correctly set up by using the following command to view the registered partitions:

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

   Example Output:

   ```
   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. Enable partition policies 22 and 23 for PED-authenticated HSM. These policies allow activation and auto-activation within the designated partition.

   NOTE: Please refer to the Luna HSM documentation for comprehensive instructions on creating an NTLS connection, initializing partitions, and assigning different user roles.

   NOTE: For PED-based Luna HSM, it is essential to ensure that the `ProtectedAuthenticationPathFlagStatus` is set to '1' in the Misc section of the Chrystoki.conf file. This setting helps maintain a secure authentication path for enhanced security.

Set up Luna HSM High-Availability

Follow the instructions provided in the Luna HSM documentation to configure and set up two or more HSM boxes on host systems for high availability. Ensure that the HAOnly setting is enabled to enable failover functionality. In the event of the primary HSM going down, all calls will automatically route to the secondary HSM until the primary recovers and restarts.

Set up Luna HSM in FIPS Mode

To configure Luna HSM in FIPS Mode, update the configuration file by adding or modifying the following setting within the `[Misc]` section:

```
RSAKeyGenMechRemap=1
```

This setting ensures that older calling mechanisms are redirected to the approved RSA key generation methods (186-3 with primes and 186-3 with aux primes) required for FIPS compliance. By making this configuration change, Luna HSM will be properly set up to operate in FIPS mode, adhering to the approved RSA key generation standards.

NOTE: The configuration setting mentioned above, `RSAKeyGenMechRemap=1`, is not required for the Universal Client. It is specifically applicable only for Luna Client 7.x.

Control User Access to the HSM

By default, only the root user has access to the HSM. To grant access to non-root users or applications, add them to the hsmusers group. The hsmusers group is automatically created during client software installation and persists even after uninstallation, allowing for seamless upgrades while retaining the hsmusers group configuration.

To add users to hsmusers group

To add users to the hsmusers group and allow non-root users or applications access to the HSM, follow these steps:

1. Ensure that you have sudo privileges on the client workstation.
2. Use the following command to add a user to the hsmusers group:

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

   Replace `<username>` with the name of the user you want to add to the hsmusers group.

   NOTE: The users you add to the hsmusers group must already exist on the client workstation. Only the users added to the hsmusers group will have access to the HSM.

To remove users from hsmusers group

To remove users from the hsmusers group and revoke their access to the HSM, follow these steps:

1. Ensure that you have sudo privileges on the client workstation.
2. Use the following command to remove a user from the hsmusers group:

   Replace `<username>` with the name of the user you want to remove from the hsmusers group.

   NOTE: After removing the user from the hsmusers group, please note that their access to the HSM will persist until the client workstation is rebooted. To see the changes take effect, it is necessary to log in again.

Configure Luna Cloud HSM service

Follow these steps to set up your Luna Cloud HSM:

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 using the following command:

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

   NOTE: Do not extract to a new subdirectory. Place the files in the client install directory.

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

   ```
   source ./setenv
   ```

   > **NOTE:** To add the configuration to an already installed UC client, use the -addcloudhsm option when running the setenv script.

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

   > **NOTE:** If your organization requires non-FIPS algorithms for your operations, ensure that the Allow non-FIPS approved algorithms check box is checked. For more information, refer to Supported Mechanisms.

Install GPG-dependent packages

Before beginning the integration, it is necessary to install the following GPG-dependent packages from the official GnuPG website:

> npth

> libgpg-error

> libgcrypt

> libksba

> libassuan

Install Pinentry package

To enable partition access authentication for GPG, you have the option to use the salogin utility, which comes bundled with the Thales Luna Client software. However, if you prefer not to use salogin, you can install the Pinentry package from the GnuPG website.

Install GnuPG package

After successfully building and installing the aforementioned packages, proceed to install the GnuPG package from the GnuPG website.

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

Once GnuPG is installed, proceed to install the pkcs11-helper and gnupg-pkcs11-scd components along with their libraries. Ensure the correct version of these libraries is used at runtime by executing the following command:

```
export LD_LIBRARY_PATH=/usr/local/lib
```

> **NOTE:** During the build process of the gnupg-pkcs11-scd component, the required files associated with it are utilized. To maintain separation between any existing files and the new ones, it is recommended to set the LD_LIBRARY_PATH environment variable accordingly.

Integrating GnuPG with Luna HSM

This chapter covers the following topics:

Access Luna HSM

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

> Using salogin utility

> Using Pinentry

Using salogin utility

The salogin utility allows you to establish a persistent session, eliminating the need to enter the password every time GPG accesses the HSM object.

> **NOTE:** Please note that the persistent session is not supported for Luna Cloud HSM. For Luna Cloud HSM, refer to the Using Pinentry section below.

To open a persistent session using the salogin utility, follow these steps:

1. Add the following text to the /etc/Chrystoki.conf file: To open the persistent session using **salogin** utility, perform the following steps:

   ```
   Misc = {

         AppIdMajor=1;

         AppIdMinor=1;

   }
   ```

2. Run the following command to open the authenticated persistent session for accessing the HSM object: 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>
   ```

   Here, “-s” represents the slot ID, and “-I” represents the AppId set in the Chrystoki.conf file.

Using Pinentry

Pinentry is an alternative method for authenticating partition access in GPG. To use Pinentry, you need to add the following text to 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 in the ~/.gnupg/ directory.

Configure 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 in the **~/.gnupg/gnupg-pkcs11-scd.conf** file:

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

provider-p1-cert-private

provider-p1-private-mask 0
```

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

Generate keys and certificates

To generate the RSA key pair on the Luna HSM for GPG, follow these steps:

1. Open the CMU utility provided with Luna Client: The CMU utility is located in the /usr/safenet/lunaclient/bin directory. Execute the following command:

   ```
   # /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
   ```

2. Enter the partition password and select the appropriate RSA Mechanism Type when prompted. For example, select "PKCS" by entering "1" when prompted.

NOTE: The CMU command options may vary slightly in different versions of Luna Client, Please refer to the Luna HSM documentation for exact options.

3. **Generate a 32-byte Hex Value for the Key ID:** You can use one of the following Linux commands to generate a 32-byte Hex value, which can be used as the key ID:

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

   Or,

   ```
   # xxd -len 16 -plain /dev/urandom
   ```

   This generated Hex value will serve as the key ID for your GPG key.

4. **View the contents generated on the HSM partition:** To view the contents generated on the HSM partition, use the following command. Make a note of the assigned handle for the public and private keys.

   ```
   # /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
   ```

5. **Generate a self-signed certificate:** Follow these steps to generate a self-signed certificate using the previously generated public/private key. When prompted, enter the partition password and provide the required certificate attributes.

   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” remains the same for both the keys and the certificate. The self-signed certificate is intended for test purposes. In a production environment, create the certificate request and have it signed by a trusted Certificate Authority.

6. If you need to use different keys and certificates for Encryption and Authentication, repeat steps 1-3.

   NOTE: Make sure that the “id” and “label” for each key and certificate are unique.

Configure GPG to use the PKCS#11 smart card daemon

To configure the gpg-agent and enable it to utilize the smart card daemon for accessing keys on the HSM, follow these steps:

1. Open the **~/.gnupg/gpg-agent.conf** file and add the following line:

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

   > **NOTE:** if ~**/.gnupg/gpg-agent.conf** file does not exist, then create the file and add the `scdaemon-program` line.

2. Add or modify the following lines in the **~/.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 does not exist, copy the **gnupg-pkcs11-scd.conf.example** file from either **/usr/local/etc/gnupg-pkcs11-scd.conf.example** or **/usr/local/share/doc/gnupg-pkcs11-scd/gnupg-pkcs11-scd.conf.example** file to the **~/.gnupg** directory and rename it. Ensure that the library path is correct when using Luna Cloud HSM.

3. Set the following environment variables to ensure that 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

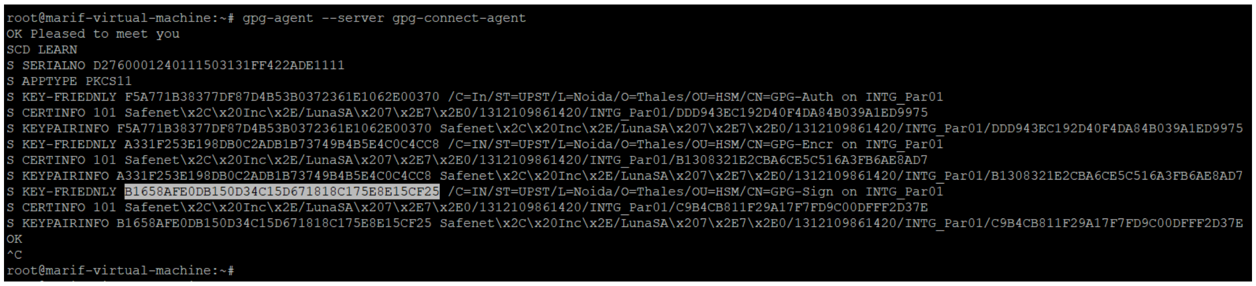
1. Use the following command to enable GPG to discover all useful information from the card or HSM partition:

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

2. Execute the following command to connect the agent to HSM and retrieve the keys from it:

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

3. At the prompt, enter SCD LEARN. The Pinentry program will pop up and prompt you for the partition password. The output of the command will be similar to the image below:



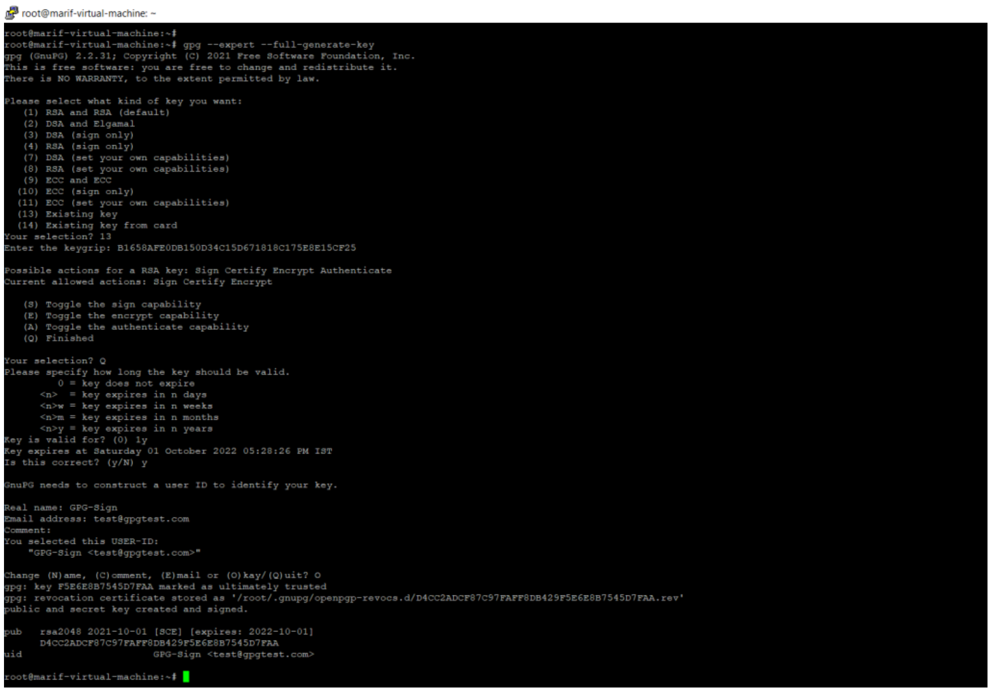
```
root@marif-virtual-machine:~# gpg-agent --server gpg-connect-agent
OK Pleased to meet you
SCD LEARN
S SERIALNO D27600012401115031 31FF422ADE1111
S APPTYPE PKCS11
S KEY-FRIEDNLY F5A771B38377DF87D4B53B0372361E1062E00370 /C=In/ST=UPST/L=Noida/O=Thales/OU=HSM/CN=GPG-Auth on INTG_Par01
S CERTINFO 101 Safenet\x2C\x20Inc\x2E/LunaSA\x207\x2E7\x2E0/1312109861420/INTG_Par01/DDD943EC192D40F4DA84B039A1ED9975
S KEYPAIRINFO F5A771B38377DF87D4B53B0372361E1062E00370 Safenet\x2C\x20Inc\x2E/LunaSA\x207\x2E7\x2E0/1312109861420/INTG_Par01/DDD943EC192D40F4DA84B039A1ED9975
S KEY-FRIEDNLY A331F253E198DB0C2ADB1B73749B4B5E4C0C4CC8 /C=IN/ST=UPST/L=Noida/O=Thales/OU=HSM/CN=GPG-Encr on INTG_Par01
S CERTINFO 101 Safenet\x2C\x20Inc\x2E/LunaSA\x207\x2E7\x2E0/1312109861420/INTG_Par01/B1308321E2CBA6CE5C516A3FB6AE8AD7
S KEYPAIRINFO A331F253E198DB0C2ADB1B73749B4B5E4C0C4CC8 Safenet\x2C\x20Inc\x2E/LunaSA\x207\x2E7\x2E0/1312109861420/INTG_Par01/B1308321E2CBA6CE5C516A3FB6AE8AD7
S KEY-FRIEDNLY B1658AFE0DB150D34C15D671818C175E8E15CF25 /C=IN/ST=UPST/L=Noida/O=Thales/OU=HSM/CN=GPG-Sign on INTG_Par01
S CERTINFO 101 Safenet\x2C\x20Inc\x2E/LunaSA\x207\x2E7\x2E0/1312109861420/INTG_Par01/C9B4CB811F29A17F7FD9C00DFFF2D37E
S KEYPAIRINFO B1658AFE0DB150D34C15D671818C175E8E15CF25 Safenet\x2C\x20Inc\x2E/LunaSA\x207\x2E7\x2E0/1312109861420/INTG_Par01/C9B4CB811F29A17F7FD9C00DFFF2D37E
OK
^C
root@marif-virtual-machine:~#
```

   Look for the line starting with **S KEY-FRIENDLY** and note the 20-byte hash. You will need to provide this hash as the **keygrip** in the next steps, when prompted.

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

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

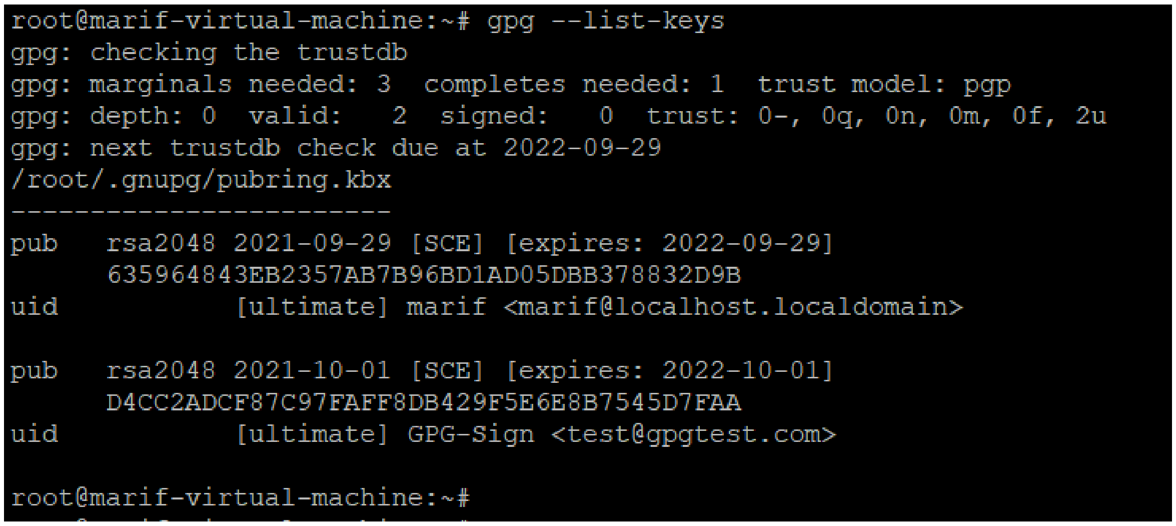
   Follow the prompts to select the key type, specify the purpose or usage of the key, enter your real name that'll be used as a reference to RPM signing, provide an email address associated with the key, and choose the validity period for the key.



```
root@marif-virtual-machine: ~
root@marif-virtual-machine:~#
root@marif-virtual-machine:~# gpg --expert --full-generate-key
gpg (GnuPG) 2.2.31; Copyright (C) 2021 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.

Please select what kind of key you want:
   (1) RSA and RSA (default)
   (2) DSA and Elgamal
   (3) DSA (sign only)
   (4) RSA (sign only)
   (7) DSA (set your own capabilities)
   (8) RSA (set your own capabilities)
   (9) ECC and ECC
  (10) ECC (sign only)
  (11) ECC (set your own capabilities)
  (13) Existing key
  (14) Existing key from card
Your selection? 13
Enter the keygrip: B1658AFE0DB150D34C15D671818C175E8E15CF25

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

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

Your selection? Q
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) 1y
Key expires at Saturday 01 October 2022 05:28:26 PM IST
Is this correct? (y/N) y

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

Real name: GPG-Sign
Email address: test@gpgtest.com
Comment:
You selected this USER-ID:
    "GPG-Sign <test@gpgtest.com>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? O
gpg: key F5E6E8B7545D7FAA marked as ultimately trusted
gpg: revocation certificate stored as '/root/.gnupg/openpgp-revocs.d/D4CC2ADCF87C97FAFF8DB429F5E6E8B7545D7FAA.rev'
public and secret key created and signed.

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

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

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

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



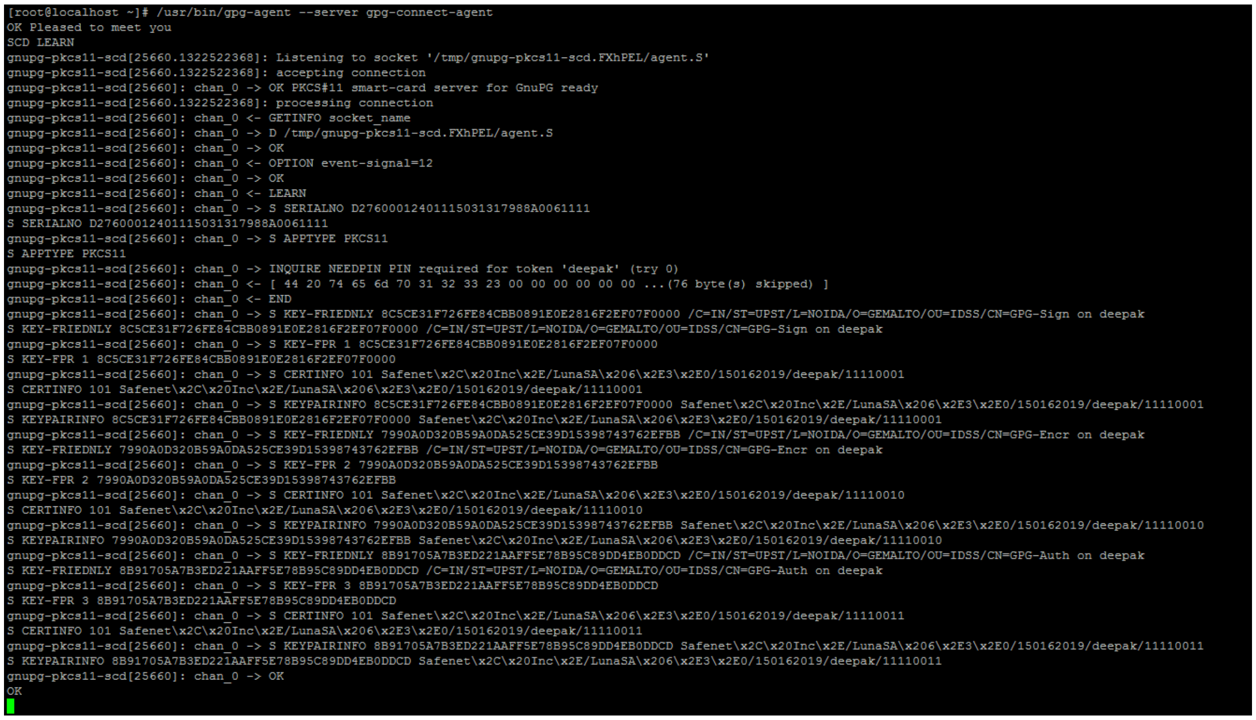
```
root@marif-virtual-machine:~# gpg --list-keys
gpg: checking the trustdb
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: depth: 0  valid:   2  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 2u
gpg: next trustdb check due at 2022-09-29
/root/.gnupg/pubring.kbx
------------------------
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. Connect the agent to HSM and retrieve the keys by executing the following command:

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

2. At the prompt, enter **SCD LEARN**. The pinentry program will appear and prompt you for the partition password. The output of the command will be similar to the image below:



```
[root@localhost ~]# /usr/bin/gpg-agent --server gpg-connect-agent
OK Pleased to meet you
SCD LEARN
gnupg-pkcs11-scd[25660.1322522368]: Listening to socket '/tmp/gnupg-pkcs11-scd.FXhPEL/agent.S'
gnupg-pkcs11-scd[25660.1322522368]: accepting connection
gnupg-pkcs11-scd[25660]: chan_0 -> OK PKCS#11 smart-card server for GnuPG ready
gnupg-pkcs11-scd[25660.1322522368]: processing connection
gnupg-pkcs11-scd[25660]: chan_0 <- GETINFO socket_name
gnupg-pkcs11-scd[25660]: chan_0 -> D /tmp/gnupg-pkcs11-scd.FXhPEL/agent.S
gnupg-pkcs11-scd[25660]: chan_0 -> OK
gnupg-pkcs11-scd[25660]: chan_0 <- OPTION event-signal=12
gnupg-pkcs11-scd[25660]: chan_0 -> OK
gnupg-pkcs11-scd[25660]: chan_0 <- LEARN
gnupg-pkcs11-scd[25660]: chan_0 -> S SERIALNO D27600012401115031317988A0061111
S SERIALNO D27600012401115031317988A0061111
gnupg-pkcs11-scd[25660]: chan_0 -> S APPTYPE PKCS11
S APPTYPE PKCS11
gnupg-pkcs11-scd[25660]: chan_0 -> INQUIRE NEEDPIN PIN required for token 'deepak' (try 0)
gnupg-pkcs11-scd[25660]: chan_0 <- [ 44 20 74 65 6d 70 31 32 33 23 00 00 00 00 00 00 ...(76 byte(s) skipped) ]
gnupg-pkcs11-scd[25660]: chan_0 <- END
gnupg-pkcs11-scd[25660]: chan_0 -> S KEY-FRIEDNLY 8C5CE31F726FE84CBB0891E0E2816F2EF07F0000 /C=IN/ST=UPST/L=NOIDA/O=GEMALTO/OU=IDSS/CN=GPG-Sign on deepak
S KEY-FRIEDNLY 8C5CE31F726FE84CBB0891E0E2816F2EF07F0000 /C=IN/ST=UPST/L=NOIDA/O=GEMALTO/OU=IDSS/CN=GPG-Sign on deepak
gnupg-pkcs11-scd[25660]: chan_0 -> S KEY-FPR 1 8C5CE31F726FE84CBB0891E0E2816F2EF07F0000
S KEY-FPR 1 8C5CE31F726FE84CBB0891E0E2816F2EF07F0000
gnupg-pkcs11-scd[25660]: chan_0 -> S CERTINFO 101 Safenet\x2C\x20Inc\x2E/LunaSA\x206\x2E3\x2E0/150162019/deepak/11110001
S CERTINFO 101 Safenet\x2C\x20Inc\x2E/LunaSA\x206\x2E3\x2E0/150162019/deepak/11110001
gnupg-pkcs11-scd[25660]: chan_0 -> S KEYPAIRINFO 8C5CE31F726FE84CBB0891E0E2816F2EF07F0000 Safenet\x2C\x20Inc\x2E/LunaSA\x206\x2E3\x2E0/150162019/deepak/11110001
S KEYPAIRINFO 8C5CE31F726FE84CBB0891E0E2816F2EF07F0000 Safenet\x2C\x20Inc\x2E/LunaSA\x206\x2E3\x2E0/150162019/deepak/11110001
gnupg-pkcs11-scd[25660]: chan_0 -> S KEY-FRIEDNLY 7990A0D320B59A0DA525CE39D15398743762EFBB /C=IN/ST=UPST/L=NOIDA/O=GEMALTO/OU=IDSS/CN=GPG-Encr on deepak
S KEY-FRIEDNLY 7990A0D320B59A0DA525CE39D15398743762EFBB /C=IN/ST=UPST/L=NOIDA/O=GEMALTO/OU=IDSS/CN=GPG-Encr on deepak
gnupg-pkcs11-scd[25660]: chan_0 -> S KEY-FPR 2 7990A0D320B59A0DA525CE39D15398743762EFBB
S KEY-FPR 2 7990A0D320B59A0DA525CE39D15398743762EFBB
gnupg-pkcs11-scd[25660]: chan_0 -> S CERTINFO 101 Safenet\x2C\x20Inc\x2E/LunaSA\x206\x2E3\x2E0/150162019/deepak/11110010
S CERTINFO 101 Safenet\x2C\x20Inc\x2E/LunaSA\x206\x2E3\x2E0/150162019/deepak/11110010
gnupg-pkcs11-scd[25660]: chan_0 -> S KEYPAIRINFO 7990A0D320B59A0DA525CE39D15398743762EFBB Safenet\x2C\x20Inc\x2E/LunaSA\x206\x2E3\x2E0/150162019/deepak/11110010
S KEYPAIRINFO 7990A0D320B59A0DA525CE39D15398743762EFBB Safenet\x2C\x20Inc\x2E/LunaSA\x206\x2E3\x2E0/150162019/deepak/11110010
gnupg-pkcs11-scd[25660]: chan_0 -> S KEY-FRIEDNLY 8B91705A7B3ED221AAFF5E78B95C89DD4EB0DDCD /C=IN/ST=UPST/L=NOIDA/O=GEMALTO/OU=IDSS/CN=GPG-Auth on deepak
S KEY-FRIEDNLY 8B91705A7B3ED221AAFF5E78B95C89DD4EB0DDCD /C=IN/ST=UPST/L=NOIDA/O=GEMALTO/OU=IDSS/CN=GPG-Auth on deepak
gnupg-pkcs11-scd[25660]: chan_0 -> S KEY-FPR 3 8B91705A7B3ED221AAFF5E78B95C89DD4EB0DDCD
S KEY-FPR 3 8B91705A7B3ED221AAFF5E78B95C89DD4EB0DDCD
gnupg-pkcs11-scd[25660]: chan_0 -> S CERTINFO 101 Safenet\x2C\x20Inc\x2E/LunaSA\x206\x2E3\x2E0/150162019/deepak/11110011
S CERTINFO 101 Safenet\x2C\x20Inc\x2E/LunaSA\x206\x2E3\x2E0/150162019/deepak/11110011
gnupg-pkcs11-scd[25660]: chan_0 -> S KEYPAIRINFO 8B91705A7B3ED221AAFF5E78B95C89DD4EB0DDCD Safenet\x2C\x20Inc\x2E/LunaSA\x206\x2E3\x2E0/150162019/deepak/11110011
S KEYPAIRINFO 8B91705A7B3ED221AAFF5E78B95C89DD4EB0DDCD Safenet\x2C\x20Inc\x2E/LunaSA\x206\x2E3\x2E0/150162019/deepak/11110011
gnupg-pkcs11-scd[25660]: chan_0 -> OK
OK
```

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

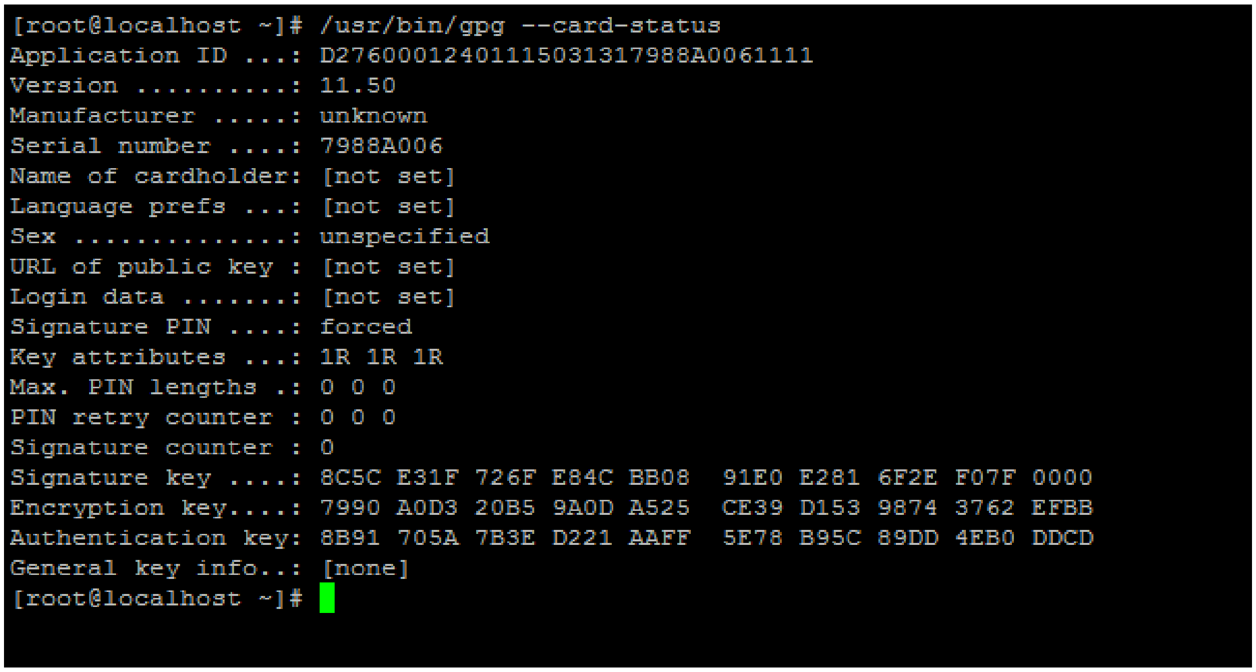
3. Look for the line starting with **S KEY-FRIENDLY**. Identify the signing/encryption/authentication certificate by the appropriate Common Name (CN), and copy the 20-byte SHA-1 hash. Add it to the **gnupg-pkcs11-scd.conf** file as follows:

   ```
   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. Enable GPG to discover useful information about the card or HSM partition by executing the following command:

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



```
[root@localhost ~]# /usr/bin/gpg --card-status
Application ID ...: D27600012401115031317988A0061111
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. Generate the GPG virtual keys by executing the following commands. Note that the keys are not actually generated on the localhost and only a reference to the HSM keys is created by GPG.

```
# gpg --card-edit

# Command> admin

# Command> generate
```

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 the real name.

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

Test 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, replacing <Your key name> with the real name of the key and “somefile” with the 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 have opened a persistent session via **salogin**, you will not be asked to provide the password.

2. Verify the original file by running the following command:

   ```
   # gpg somefile.gpg
   ```

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

RPM signing and verification

RPM Signing uses GPG for key management and cryptographic functions. Follow the steps below to sign and verify RPMs effectively.

Set up the Environment

Before proceeding with RPM signing, set up the environment as follows:

1. Run the following command:

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

   NOTE: This step is only required for GPG v2.0.x. Skip this step for GPG v2.2.x

2. Create or update the file `~/.rpmmacros` in order to utilize the key. The contents of the file should be similar to the following:

   ```
   %_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 set to gpg
- `gpg_path` should point to the gnupg directory path
- `gpg_name` should match the Real Name of the key
- `gpg` should point to the gpg binary/executable
- `gpg_sign_cmd` should match the provided value for forcing the SHA256 signature

Sign an RPM

To sign an RPM, you can choose between using the RPM command and using an automated script.

Using the RPM command:

1. Sign an RPM file. Replace example.rpm with the actual RPM file you want to sign.

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

2. Use the --resign flag if you need to re-sign the RPM file.

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

Use an automated script:

1. Ensure that the **expect** tool is installed. If not, run the following command to install it:

   ```
   # yum install expect expect.
   ```

2. Create the script from the source listing below, modifying it to reference the appropriate key name:

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

3. Execute the following command to run the script and sign the RPM:

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

   Replace `<scriptname>` with the name of the script file and `<your_rpm>` with the actual RPM file.

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

Verify a signed RPM

To verify the signature on an RPM file, follow these steps:

1. Export the public key associated with the signing 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 into your GPG keychain by using the command:

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

   For example:

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

3. Use the following command to verify the signature on the RPM file:

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

   For example:

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

The output will indicate whether the signature is valid. If the signature is valid, you will see something like:

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

NOTE: RPM output may appear generic because SHA1 (and MD5) digests are an integral part of RPM packages. To confirm the signing algorithm used, you can run the following command:

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

The output will display the signing algorithm used and other related information. For example, you may see:

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

Perform unattended RPM signing

In certain scenarios, you may need to automate the RPM signing process during product builds and RPM creation. Manually entering the partition password for each RPM signature is not practical. To enable unattended RPM signing, follow these steps:

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

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

   NOTE: The GPG Agent will run in the background and cache the partition password for future use. For more options, refer to the GPG Agent's manual pages.

2. Create a simple script using the Expect tool. Create a file called rpm-sign.exp and add 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 a tool that automated interactions with applications having a text terminal interface. To install Expect on a RedHat/CentOS system, you can use yum:

```
# yum install expect expectk
```

3. Sign the RPM using the created script file.

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

By following these steps, you have completed the integration of Luna HSM with GnuPG (GPG), which secures the cryptographic keys on a FIPS 140-3 compliant hardware device. The crypto keys secured on the HSM partition are available for signing as required through GnuPG, ensuring enhanced security for your RPM signing process.

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.