Kiuru SAM Integration with Thales Luna 7 HSM

This document outlines the steps needed to be taken on the HSM side in preparation for integration with Kiuru SAM (used with Kiuru MSSP).

Thales Luna 7 provides two option to select between two distinct mechanisms for securing the end-user keys:

1. Scalable Key Storage (SKS)
2. Key Wrap Key (KWK)

For ease of use, we suggest opting for SKS instead of KWK.

The *Kiuru SAM* server expects following Common Criteria Protection Profiles to be fulfilled:

- The Cryptographic Module is required to be certified against EN 419221-5 to obtain a QSCD.
- The SAM is required to be certified against CEN EN 419241-2 (to obtain a QSCD.)

The *Kiuru SAM* server can operate also with classical HSMs (the KWK mode), although obtaining a certification as QSCD will be more difficult in that case.

1. HSM Initialization

High-availability clustering of HSMs is frequently a prerequisite.

This means that every HSM participating in the cluster must have identical configuration, and master keys must be cloned in between devices. Also when creating a partition, it is done on one HSM instance, and then cloned to other instances. The *master SMK* of each joining partition is changed when it joins to a ha-group. As every key on V1 partition is encrypted with the partition SMK, they become unusable after joining.

With Thales Luna7 this is known as "High-Availability Group"

> https://thalesdocs.com/gphsm/luna/7/docs/network/Content/admin_partition/ha/ha.htm

The best mode here is SKS.

> https://thalesdocs.com/gphsm/luna/7/docs/network/Content/admin_partition/ha/setup.htm

There is important note at the start of that page:

> **V1 partitions**: If you add an application partition with an existing SMK to an HA group, the primary member's SMK overwrites the existing SMK of the joining partition. If a partition's SMK has ever been used to encrypt important SKS objects, save a backup of the SMK before adding that partition to any HA group.
>
> ("SMK" = Scalable Master Key)

Policy settings must be as documented at every participating HSM, and also at every partition of the HSMs!

Setup partitions and ha-groups at one HSM instance, then add more external

members to this initial node. It is not clear if the concept of "group" is entirely managed at the client side, or happens also at the HSM itself. Network connectivity control is definitively on client side, which means that every client node must define the cluster communication locally. Initially add every HSM instance, then mark remote nodes to be on standby -list.

2. Scalable Key Storage

SKS requires Thales Luna 7 firmware version of 7.7.0 or higher.

To initialize SKS, run the following commands on the HSM:

2.1. Setup partition

The partition needs to support SKS feature, and be V1 type

The *partition create* command gets parameters:

- Partition name
- Storage size = 1 M
- version = 1

Example commands for creating the partition and setting PSO PIN, CO PIN and CU PIN

```
$ lunash:>hsm login
$ lunash:>par create -p KIURUSAM -s 1000000 -v 1

$ lunash:> partition init -partition KIURUSAM -label KIURUSAM \
$          -pass password -domain Test

$ lunash:> partition init co -partition KIURUSAM -psopin password \
$     -copin password

$ lunash:> partition init cu -partition KIURUSAM -copin password \
$     -cupin password
```

2.2. Disable Per-Key Authorization Data

We need to disable the "Require Per Key Authorization Data" setting:

```
$ /usr/safenet/lunaclient/bin/lunacm ...
lunacm:> slot set -slot 0
lunacm:> partition showpolicies
lunacm:> role login -name po
  (Give Partition SO's password)
lunacm:> partition changepolicy -policy 40 -value 0
```

3. Configure SAM

Provide the specified information to Methics for MSSP configuration:

1. HSM IP address and port

2. Partition name
3. Slot number
4. PSO PIN, CO PIN, CU PIN
5. LCO PIN if using SKS+PKA

4. Key Wrap Key

4.1. Create KWK

We want to initialize an AES key-wrap key called Kiuru-Sam-Wrap-1.

To initialize the KWK, run the following command and follow the flow on the HSM:

```
# /opt/safenet/bin/64/ckdemo
...
Enter your choice: 1
Status: Doing great, no errors (CKR_OK)

Enter your choice: 3
Partition SO    [0]
Crypto Officer  [1]
Crypto User     [2]: 1
Enter PIN: ********************
Status: Doing great, no errors (CKR_OK)

Enter your choice: 45
Select type of key to generate
...
> 16
Enter Key Length in bytes (16, 24, 32): 32    <<-- 32 bytes = 256 bits
Enter Is Token Attribute [0-1]: 1             <<-- indicates HSM residency
Enter Is Sensitive Attribute [0-1]: 1
Enter Is Private Attribute [0-1]: 1
Enter Is Modifiable Attribute [0-1]: 1        <<-- could be 0
Enter Encrypt Attribute [0-1]: 0
Enter Decrypt Attribute [0-1]: 0
Enter Sign Attribute [0-1]: 0
Enter Verify Attribute [0-1]: 0
Enter Wrap Attribute [0-1]: 1                 <<-- used only for wrap+unwrap!
Enter Unwrap Attribute [0-1]: 1
Enter Derive Attribute [0-1]: 0
Enter Extractable Attribute [0-1]: 0          <<-- never extractable
Generated AES Key: 78c20000180000015c610900   <<-- use below!

Status: Doing great, no errors (CKR_OK)

Enter your choice: 25
Which object do you want to modify (0 to list available objects) :
78c20000180000015c610900                      <<--- Copy from above

Edit template for set attribute operation.
(1) Add Attribute (2) Remove Attribute (0) Accept Template: 1
...
Select which one: 3
Enter string value: Kiuru-Sam-Wrap-1
```

```
CKA_LABEL=Kiuru-Sam-Wrap-1

(1) Add Attribute (2) Remove Attribute (0) Accept Template: 0
Status: Doing great, no errors (CKR_OK)
```

5. Generic HSM keys for all modes

5.1 Configure SAM

Deliver the following information to Methics for MSSP configuration:

1. HSM IP address and port
2. Key Wrap Key label
3. Slot number
4. CO PIN

5.2 Create symmetric DB master key

The HSM has master key, and actual operational keys are derived from it by encrypting static data.
This lessens the number of HSM calls by factor 20 to 30 avoiding HSM overload.

```
# /opt/safenet/bin/64/ckdemo
… (login as above)
Enter your choice: 45
Select type of key to generate
. . .
> 16
Enter Key Length in bytes (16, 24, 32): 32    <<-- 32 bytes = 256 bits
Enter Is Token Attribute [0-1]: 1             <<-- indicates HSM residency
Enter Is Sensitive Attribute [0-1]: 1
Enter Is Private Attribute [0-1]: 1
Enter Is Modifiable Attribute [0-1]: 1        <<-- could be 0
Enter Encrypt Attribute [0-1]: 1              <<-- used for encrypt
Enter Decrypt Attribute [0-1]: 0              <<-- not used for decrypt
Enter Sign Attribute [0-1]: 0                 <<-- not used for HMAC sign
Enter Verify Attribute [0-1]: 0               <<-- not used for HMAC verify
Enter Wrap Attribute [0-1]: 0                 <<-- not used for wrap+unwrap!
Enter Unwrap Attribute [0-1]: 0
Enter Derive Attribute [0-1]: 0               <<-- not used for key derivation
Enter Extractable Attribute [0-1]: 0          <<-- never extractable
Generated AES Key: 79c20000180000015c610900   <<-- use below!

Status: Doing great, no errors (CKR_OK)


Enter your choice: 25
Which object do you want to modify (0 to list available objects):
79c20000180000015c610900                      <<--- Copy from above
Edit template for set attribute operation.
(1) Add Attribute (2) Remove Attribute (0) Accept Template: 1
...
```

```
Select which one: 3
Enter string value: Kiuru-Samdb-aeskey-1
CKA_LABEL=Kiuru-Samdb-aeskey-1

(1) Add Attribute (2) Remove Attribute (0) Accept Template: 0
Status: Doing great, no errors (CKR_OK)

Enter your choice: 0
```

5.3. Create DB PKI signing key

Generating the Java self-signed PKI key needs to use Kiuru keytool. With the SAM's hsm.conf configured, we can use the Kiuru Keytool to generate the PKI key:

Content of the *kiuru-samdb-luna-slot3-cryptoofficer.keystore* file is:

```
slot:3
usertype:CKU_CRYPTO_OFFICER
```

Note that here we use `usertype:CKU_CRYPTO_OFFICER` as we create permanent keys in the HSM. Operational access to these keys by the SAM will be using `usertype:CKU_CRYPTO_USER` role, which is the default of the `usertype` parameter, and therefore is not indicated in the "keystore" file used by the SAM server configuration.

There is need for one PKI key, which is used to sign the SAM auditlog records. We suggest using the ECDSA keys as that has smaller signature size.

Execute following command as `kiuru' user (not as root!)

Generating ECDSA key on curve NIST-P384r1 (secp384r1):

```
$ ## Execute as 'kiuru' user

$ /opt/sam/bin/kiuru.sh keytool -genkeypair -alias Kiuru-Samdb-Auditlog-
Signkey-1 -provider com.safenetinc.luna.provider.LunaProvider -storetype Luna
-keystore /opt/sam/security/kiuru-samdb-luna-slot3-cryptoofficer.keystore -
keyalg EC -keysize 384 -groupname secp384r1 -dname 'cn=samdb-auditlog-signkey-
1' -validity 9000  -storepass *************
. . .
Enter key password for <Kiuru-SamDb-Auditlog-Signkey-1>
        (RETURN if same as keystore password):   <enter>
$
```

Alternative for generating a 2048 bit RSA key:

```
$ ## Execute as 'kiuru' user

$ /opt/sam/bin/kiuru.sh keytool -genkeypair -alias Kiuru-Samdb-Auditlog-
Signkey-1 -provider com.safenetinc.luna.provider.LunaProvider -storetype Luna
-keystore /opt/sam/security/kiuru-samdb-luna-slot3-cryptoofficer.keystore -
keyalg RSA -keysize 2048  -dname 'cn=samdb-auditlog-signkey-1' -validity 9000
-storepass *************
. . .
Enter key password for <Kiuru-SamDb-Auditlog-Signkey-1>
        (RETURN if same as keystore password):   <enter>
$
```

This key is used for signing auditlog blocks generated by specific Kiuru SAM instance. In clustered setup each Kiuru SAM generates their own chain of blocks. External application can verify the auditlog continuity and immutability by computing SHA-256 checksums + verifying signatures.

This key is used at every 100 events, or every 30 seconds (which ever happens sooner), or every 5 minutes if no events have happened since previous block signature.

6. HSM Clustering

A clustered HSM setup can be achieved with the steps described in the following sections.

6.1. Initialize HSMs

Initialize all of the cluster HSMs. See chapter 1. Make sure that all HSMs can access each other over network.

Also make sure that:

- All partitions must have same SO/CU/CO PINs (passwords)

- All partitions must have identical cloning domain labels (`lunacm> par domainlist`)

Set up HSM client config from each SAM to each HSM:

```
lunacm> clientconfig deploy
```

6.1. Create HA Group

Create a new HA group and add members to it by running the following commands on each SAM node (first on the primary site):

```
lunacm> ha createGroup -serialNumber [sn1] -label KIURUSKS -password [pwd]
lunacm> ha addMember -serialNumber [sn2] -group KIURUSKS -password [pwd]
lunacm> ha addMember -serialNumber [sn3] -group KIURUSKS -password [pwd]
```