OpenStack

Barbican

Flamingo (v21.0.0)

Integration Guide

u.trust GP HSM Se-Series

v6.3.0

utimaco®

Imprint

Copyright 2026	Utimaco IS GmbH
	Krefelder Straße 220
	52070 Aachen
	Germany
Phone	AMERICAS +1-844-UTIMACO (+1 844-884-6226)
	EMEA +49 800-627-3081
	APAC +81 800-919-1301
Internet	https://support.hsm.utimaco.com/
e-mail	support@utimaco.com
Document Version	1.0.0
Date	2026-03-13
Status	PUBLISHED
Document No.	IG-2025-0057

Table of Contents

1 Introduction

This guide is part of the information and support provided by Utimaco. Additional documentation produced to support your Utimaco SecurityServer product can be found in the document directory of the Utimaco SecurityServer product bundle. All Utimaco SecurityServer product documentation is available from Utimaco's website at https://utimaco.com/.

1.1 About This Guide

This guide describes how to enable HSM integration with OpenStack Barbican (Flamingo release). Utimaco HSM securely stores the MKEK and HMAC used by Barbican.

1.2 Target Audience for This Guide

This guide is intended for Partner Product and Utimaco u.trust GP HSM Se-Series administrators.

1.3 Document Conventions

The following conventions are used in this guide:

Convention	Use	Example
Bold	Items of the Graphical User Interface (GUI), e.g., menu options	Press **OK**
`Monospaced`	Code that is given for explanation or as an example, file paths	`chsm-create`
Italic	References and important terms	See *Sample Chapter* in the *CryptoServer - Sample Manual*

Table 1: Document conventions

We use special icons to highlight the most important notes and information.

Here you will find important safety information.

Here you will find additional notes or supplementary information.

This message indicates the expected result after the successful execution of an instruction.

1.4 Abbreviations

The following abbreviations are used in this guide:

Abbreviation	Meaning
API	Application Programming Interface
CSADM	CryptoServer Command-line Administration Tool
CSP	Cryptographic Service Provider
CXI	Cryptographic eXtended Services Interface
DB	Database
GUI	Graphical User Interface
HMAC	Hash-based message authentication code
HSM	Hardware Security Module

IP	Internet Protocol
LAN	Local Area Network
MBK	Master Backup Key
MKEK	Master Key Encryption Key
PCIe	PCI Express Interface
PKCS	Public Key Cryptography Standards
PKCS#11	PKCS Part 11: The Cryptographic Token Interface Standard
SO	The PKCS#11 cryptographic slot Security Officer
URL	Uniform Resource Locator

Table 2: Abbreviations

2 Overview

2.1 OpenStack Barbican

Barbican is the OpenStack Key Manager service. It provides secure storage, provisioning, and management of secret data. This includes keying material such as Symmetric Keys, Asymmetric Keys, Certificates, and raw binary data.

2.2 Utimaco u.trust General Purpose HSM Se-Series

Utimaco u.trust General Purpose HSM Se-Series is a hardware security module developed by Utimaco IS GmbH. It is a physically protected, specialized computer unit designed to perform sensitive cryptographic tasks and securely manage and store cryptographic keys and data. It can be used as a universal, independent security component for heterogeneous computer systems.

3 Integration Requirements and Prerequisites

Ensure that the system environment you will be using meets the following hardware and software requirements.

This guide assumes that the user has already installed and configured the required Software.

3.1 Tested Versions

The integrations that have been successfully tested with the Utimaco HSM and OpenStack Barbican:

Operating System	OpenStack Version	Utimaco Security Server Version	Utimaco HSM
Ubuntu 24.04	OpenStack Flamingo	SecurityServer 6.3.0	u.trust General Purpose HSM Se-Series

Table 3: List of tested versions

3.2 Software Requirements

Software	Software Requirements
HSM Interface	CryptoServer CP5 PKCS#11 Provider
HSM Utility	CryptoServer CP5 PKCS#11 Tool (p11tool2)

Table 4: List of software requirements

3.3 Hardware Requirements

Hardware	Hardware Requirements
Utimaco LAN HSM	u.trust General Purpose HSM Se-Series LAN
Utimaco PCI-e HSM	u.trust General Purpose HSM Se-Series PCIe card

Table 5: List of hardware requirements

3.4 Prerequisites

Before you begin, please ensure that you have:

- Installed and set up the operating system listed in Tested Versions.
- Installed and set up the HSM listed in Tested Versions.
- Replaced the HSM default admin with a new admin user.
- Created and stored the MBK on each HSM. Refer to the SecurityServer documentation to set up the MBK.
- Set up and configured the SecurityServer. Refer to the SecurityServer documentation to set up the HSM.
- An admin user, which is required to install software.

4 Installation and Configuration

4.1 Download and Install Utimaco SecurityServer Software

If you have not already done so, create and request an Utimaco Support Portal Account at Support - Utimaco Portal. This will allow you to download the software components needed for this installation.

1. Copy the downloaded software to the appropriate location on the OpenStack Barbican server.
2. Create `/utimaco/bin` and `/utimaco/lib` directories under `/opt`.

`›_ Console`

```
# mkdir -p  /opt/utimaco/bin

# mkdir /opt/utimaco/lib
```

3. Copy the PKCS#11 library file `libcs_pkcs11_R3.so` from the Utimaco SecurityServer software package to the `/opt/utimaco/lib` directory and make the file executable.

`›_ Console`

```
 # cp ~/u.trust-GP-HSM-Product-Bundle_v6.3.0.0/Software/Linux/Crypto_APIs/
PKCS11_R3/lib/libcs_pkcs11_R3.so /opt/utimaco/lib

 # chmod +x /opt/utimaco/lib/cs_pkcs11_R3.so
```

4. Copy the csadm and p11tool2 files from the Utimaco SecurityServer software package to `/opt/utimaco/bin` directory and make both files executable.

`›_ Console`

```
# cd ~/u.trust-GP-HSM-Product-Bundle_v6.3.0.0/Software/Linux/
Administration/

# cp csadm p11tool2 /opt/utimaco/bin

# chmod +x /opt/utimaco/bin/csadm /opt/utimaco/bin/p11tool2

# chmod +x /opt/utimaco/lib/libcs_pkcs11_R3.so
```

4.2 SecurityServer PKCS#11 Configuration

1. Create the directory `/etc/utimaco`. Locate the Utimaco PKCS#11 configuration file in your SecurityServer software directory (`Software/Linux/Crypto_APIs/PKCS11_R3/sample`). Copy the Utimaco PKCS#11 configuration file `cs_pkcs11_R3.cfg` into `/etc/utimaco` directory.

`›_ Console`

```
# mkdir /etc/utimaco

# cd ~/u.trust-GP-HSM-Product-Bundle_v6.3.0.0/Software/Linux/Crypto_APIs/
PKCS11_R3/sample/

# cp cs_pkcs11_R3.cfg /etc/utimaco

# cd /etc/utimaco
```

2. Edit the `cs_pkcs11_R3.cfg` file using your preferred text editor and make the appropriate changes to the file.

`cs_pkcs11_R3.cfg`

```
[Global]

# For unix:
Logpath = /tmp

# Loglevel (0 = NONE; 1 = ERROR; 2 = WARNING; 3 = INFO; 4 = TRACE)
Logging = 4

# Prevents expiring session after inactivity of 15 minutes
KeepAlive = true

# Set the Device to connect with
[HSMCluster]
# Device specifier
Devices = <HSM_IP>
```

For detailed guidance on commands and their parameters, please refer to the Utimaco SecurityServer documentation. The device could be a SecurityServer GP HSM, available in either PCIe or LAN form factors. Depending on the type, the device configuration line will follow one of these formats:

- **LAN-based HSM:** Device = 288@ipaddress
- **PCIe-based HSM:** Device = /dev/cs2.0

Make sure to select the appropriate format based on your specific hardware setup.

To simplify your testing process, it's recommended that you enable the PKCS#11 log file by adjusting the logging settings. Specifically:

- Set the `LogPath` to a writable directory (not a specific file).
- Set the `Logging` level to 1 for basic logging. Increase it to 4 for more detailed output during testing.

This will generate a log file named `cs_pkcs11_R3.log` within the specified `LogPath` directory. Reviewing this log can help with troubleshooting if you encounter issues. Once testing is complete, it's advisable to reduce `Logging` level to 1 or 2 to limit output to only critical or important messages.

4.3 (Optional) Installing OpenStack Barbican

Skip this section if OpenStack Barbican is already installed. The below steps are only for demonstration purposes and will change based on actual requirements.

Use the steps below to install OpenStack Flamingo with Barbican via DevStack on a single node. DevStack is a series of extensible scripts used to quickly bring up a complete OpenStack environment either based on the latest versions of everything from git master or a selected release (Dalmatian, Flamingo, etc.).

1. Create `stack` user with passwordless sudo privileges.

`›_ Console`

```
# useradd -s /bin/bash -d /opt/stack -m stack

# chmod +x /opt/stack

# echo "stack ALL=(ALL) NOPASSWD: ALL" | tee /etc/sudoers.d/stack

# su stack
```

2. Clone the DevStack Repository and switch to stable Flamingo 2025.2 release branch.

›_ Console

```
# git clone https://opendev.org/openstack/devstack /opt/stack/devstack

# cd /opt/stack/devstack

# git checkout stable/2025.2
```

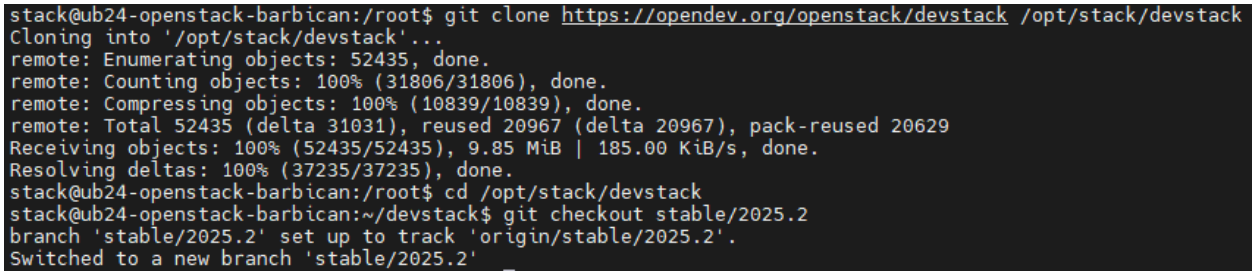

Figure 1 : Cloning DevStack

3. Copy the sample DevStack configuration file `local.conf` from `samples` folder to the DevStack directory.

›_ Console

```
# cp samples/local.conf /opt/stack/devstack
```

4. Edit the configuration file using your preferred text editor and make the following changes to the file:

- Set the passwords (`ADMIN_PASSWORD` , `DATABASE_PASSWORD` , `RABBIT_PASSWORD` , `SERVICE_PASSWORD`) to preferred values.
- Adjust `HOST_IP` to your machine's IP address if accessing from other systems. For single-machine testing, `127.0.0.1` is sufficient.

- Add the lines below to install and enable Barbican.

`local.conf`

```
# Enable Barbican plugin
enable_plugin barbican https://opendev.org/openstack/barbican stable/
2025.2
enable_service barbican
```

The sample configuration file explains each option and includes a link to more detailed settings documentation.

6. Run DevStack installation.

`›_ Console`

```
# ./stack.sh
```

This process typically takes 15-30 minutes depending on your network speed and system resources. The script will download, configure, and start all necessary OpenStack services including Barbican.

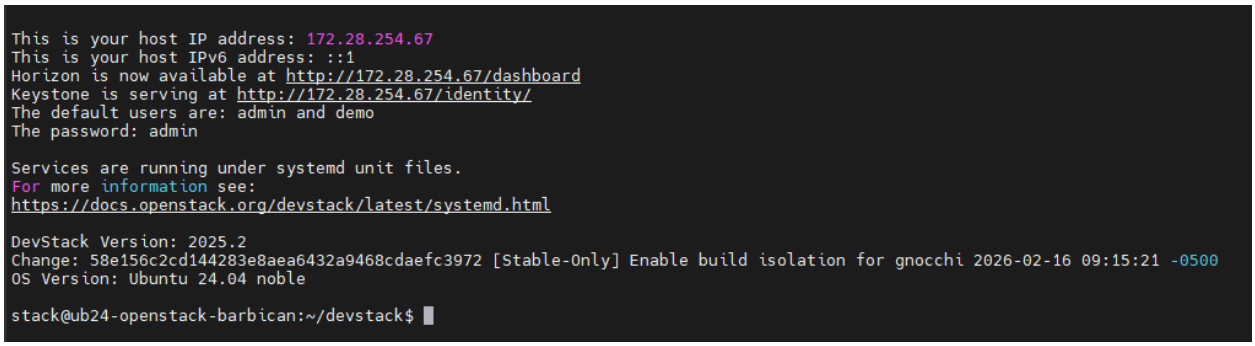

Figure 2 : OpenStack installation finished

7. Verify the installation by sourcing the OpenStack credentials and then listing the Barbican secrets, which should return empty initially and check the OpenStack services status. All services (including Barbican) should be active and running.

›_ Console

```
# source ~/devstack/openrc admin admin

# openstack secret list

# sudo systemctl list-units 'devstack@*' --no-pager
```

```
stack@ub24-openstack-barbican:/root$ sudo systemctl list-units 'devstack@*' --no-pager
  UNIT                                          LOAD   ACTIVE SUB     DESCRIPTION
  devstack@barbican-keystone-listener.service   loaded active running Devstack devstack@barbican-keystone-listener.service
  devstack@barbican-retry.service               loaded active running Devstack devstack@barbican-retry.service
  devstack@barbican-svc.service                 loaded active running Devstack devstack@barbican-svc.service
  devstack@c-api.service                        loaded active running Devstack devstack@c-api.service
  devstack@c-sch.service                        loaded active running Devstack devstack@c-sch.service
  devstack@c-vol.service                        loaded active running Devstack devstack@c-vol.service
  devstack@dstat.service                        loaded active running Devstack devstack@dstat.service
  devstack@etcd.service                         loaded active running Devstack devstack@etcd.service
  devstack@g-api.service                        loaded active running Devstack devstack@g-api.service
  devstack@keystone.service                     loaded active running Devstack devstack@keystone.service
  devstack@n-api-meta.service                   loaded active running Devstack devstack@n-api-meta.service
  devstack@n-api.service                        loaded active running Devstack devstack@n-api.service
  devstack@n-cond-cell1.service                 loaded active running Devstack devstack@n-cond-cell1.service
  devstack@n-cpu.service                        loaded active running Devstack devstack@n-cpu.service
  devstack@n-novnc-cell1.service                loaded active running Devstack devstack@n-novnc-cell1.service
  devstack@n-sch.service                        loaded active running Devstack devstack@n-sch.service
  devstack@n-super-cond.service                 loaded active running Devstack devstack@n-super-cond.service
  devstack@neutron-api.service                  loaded active running Devstack devstack@neutron-api.service
  devstack@neutron-ovn-maintenance-worker.service loaded active running Devstack devstack@neutron-ovn-maintenance-worker.service
  devstack@neutron-periodic-workers.service     loaded active running Devstack devstack@neutron-periodic-workers.service
  devstack@neutron-rpc-server.service           loaded active running Devstack devstack@neutron-rpc-server.service
  devstack@placement-api.service                loaded active running Devstack devstack@placement-api.service
  devstack@q-ovn-agent.service                  loaded active running Devstack devstack@q-ovn-agent.service

Legend: LOAD   → Reflects whether the unit definition was properly loaded.
        ACTIVE → The high-level unit activation state, i.e. generalization of SUB.
        SUB    → The low-level unit activation state, values depend on unit type.

23 loaded units listed. Pass --all to see loaded but inactive units, too.
To show all installed unit files use 'systemctl list-unit-files'.
```

Figure 3 : OpenStack services status after installation

5 Integrating OpenStack Barbican with Utimaco SecurityServer

5.1 Configuration on Utimaco SecurityServer

1. Set the environment variable to point to the PKCS#11 configuration file.

`›_ Console`

```
# export CS_PKCS11_R3_CFG="/etc/utimaco/cs_pkcs11_R3.cfg"
```

2. Create users SO (Security Officer) and USR (the Crypto user) and initialize a slot. This is done using p11tool2. We first create the SO using the `InitToken` command and change his initial PIN with `SetPin` command. Then we create the cryptographic user using the `InitPin` command and change his initial PIN as well.

`›_ Console`

```
# cd /opt/utimaco/bin

# ./p11tool2 Slot=<slot_id> Label=<label>
Login=<admin_name>,<admin_auth_token> InitToken=<Initial_SO_PIN>

# ./p11tool2 Slot=<slot_id> LoginSO=<Initial_SO_PIN>
SetPin=<Initial_SO_PIN>,<SO_PIN>

# ./p11tool2 Slot=<slot_id> LoginSO=<SO_pin>
InitPin=<Initial_CryptoUser_PIN>

# ./p11tool2 Slot=<slot_id> Login=<Initial_CryptoUser_PIN>
SetPin=<Initial_CryptoUser_PIN>,<CryptoUser_PIN>
```

3. Run the `csadm ListUsers` command to see the created users.

```
›_ Console
```

```
# ./csadm Dev=<HSM_IP> ListUsers
```

```
stack@ub24-openstack-barbican:/opt/utimaco/bin$ ./csadm Dev=3001@localhost ListUsers
Name      Permission   Mechanism      Attributes
ADMIN      22000000    RSA sign       Z[0]I[0]
SO_0000    00000200    HMAC passwd    Z[1]I[0]A[CXI_GROUP=SLOT_0000]L[barbicanToken                ]
USR_0000   00000002    HMAC passwd    Z[0]I[0]A[CXI_GROUP=SLOT_0000]
```

Figure 4 : Security officer and cryptographic user created

5.2 Configure Barbican to Use Utimaco HSM

Edit the `barbican.conf` file located in `/etc/barbican` using your preferred text editor and add the following configuration to the file. Make sure to replace the placeholders in <> with relevant values used when initializing the PKCS#11 slot in previous steps. Additional information on the configurations can be found here.

barbican.conf

```
[secretstore]
enabled_secretstore_plugins = store_crypto

[crypto]
enabled_crypto_plugins = p11_crypto

[p11_crypto_plugin]
# Path to vendor PKCS11 library (string value)
library_path = /opt/utimaco/lib/libcs_pkcs11_R3.so

# Token label used to identify the token to be used.
token_label = <label>

# Password to login to PKCS11 session (string value)
login = <CryptoUser_PIN>

# Master KEK and HMAC key labels (as stored in the HSM) (string value)
mkek_label = mkek_utimaco
hmac_label = hmac_utimaco

# (Optional) HSM Slot ID that contains the token device to be used.
(integer value)
slot_id = <slot_id>

# Secret encryption mechanism (string value) - can be changed with secrets
present in the project:
# CKM_AES_CBC (default) or CKM_AES_GCM
encryption_mechanism = CKM_AES_CBC

# Key wrapping mechanism used to wrap the PKEK with the MKEK (string value)
- can not be changed after secrets are present in the project:
# CKM_AES_CBC_PAD (default) or CKM_AES_KEY_WRAP_PAD or CKM_AES_KEY_WRAP_KWP
(recommended by OpenStack for new deployments)
key_wrap_mechanism = CKM_AES_KEY_WRAP_KWP
```

`barbican.conf`

```
# Whether to generate and pass an IV to the key wrap operation (boolean
value)
# Should be True (default) for CKM_AES_CBC_PAD and False for
CKM_AES_KEY_WRAP_PAD/KWP
key_wrap_generate_iv = False
```

As mentioned in the comment above the parameter, `key_wrap_mechanism` must be defined before any secrets are stored in the OpenStack project, as it cannot be changed afterward.

In addition to the default `CKM_AES_CBC_PAD`, the following key wrapping mechanisms are available:

- CKM_AES_KEY_WRAP_PAD
- CKM_AES_KEY_WRAP_KWP

For new deployments, OpenStack recommends using `CKM_AES_KEY_WRAP_KWP`, as it provides a standards-compliant implementation of AES Key Wrap with Padding (RFC 5649) and better interoperability across HSM vendors.

`mkek_utimaco` and `hmac_utimaco` keys will be generated on the Utimaco HSM in slot 0 in the next section of this document.

5.3 Generating MKEK and HMAC Key on Utimaco HSM

1. Generate the MKEK using the `barbican-manage hsm gen_mkek` command.

`›_ Console`

```
# sudo -u stack -i barbican-manage hsm gen_mkek --library-path '/opt/
utimaco/lib/libcs_pkcs11_R3.so' --passphrase <CryptoUser_PIN>
--slot-id <slot_id> --label 'mkek_utimaco' --length 32
```

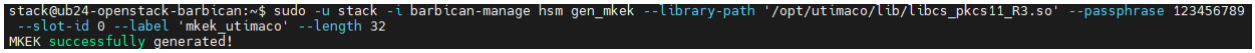

Figure 5 : MKEK generation

2. Generate the HMAC using the `barbican-manage hsm gen_hmac` command.

```
›_ Console
```

```
# sudo -u stack -i barbican-manage hsm gen_hmac --library-path '/opt/
utimaco/lib/libcs_pkcs11_R3.so' --passphrase <CryptoUser_PIN>
--slot-id <slot_id> --label 'hmac_utimaco' --length 32
```

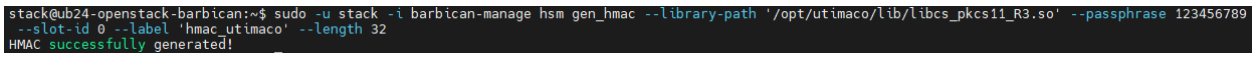

Figure 6 : HMAC generation

3. Verify that the keys are generated on the Utimaco HSM using the `p11tool2 ListObjects` command.

```
›_ Console
```

```
#./p11tool2 slot=<slot_id> LoginUser=<Crypto_User_PIN> ListObjects
```

```
stack@ub24-openstack-barbican:/opt/utimaco/bin$ ./p11tool2 Slot=0 LoginUser=123456789 ListObjects


CKO_SECRET_KEY:

+ 1.1
  CKA_KEY_TYPE                  = CKK_AES
  CKA_UNIQUE_ID                 = AFD1DA99-7671-4A26-BD8B-2CE7962D8E70
  CKA_SENSITIVE                 = CK_TRUE
  CKA_EXTRACTABLE               = CK_FALSE
  CKA_LABEL                     = mkek_utimaco
  CKA_ID                        =


+ 1.2
  CKA_KEY_TYPE                  = CKK_AES
  CKA_UNIQUE_ID                 = 9BCDC5EB-1FDD-45AD-8B40-538612A7AEEB
  CKA_SENSITIVE                 = CK_TRUE
  CKA_EXTRACTABLE               = CK_FALSE
  CKA_LABEL                     = hmac_utimaco
  CKA_ID                        =
```

Figure 7 : Listing MKEK and HMAC with p11tool2

4. Restart the OpenStack Barbican service.

›_ Console

```
# sudo systemctl restart devstack@barbican-svc.service
```

6 Verification and Testing

6.1 Encrypting and Decrypting Secrets

1. Authenticate the current shell with OpenStack admin credentials.

```
›_ Console
```

```
# source ~/devstack/openrc admin admin
```

2. Create a secret or password.

```
›_ Console
```

```
# openstack secret store --name Utimaco123 --payload FlamingoPassword
```

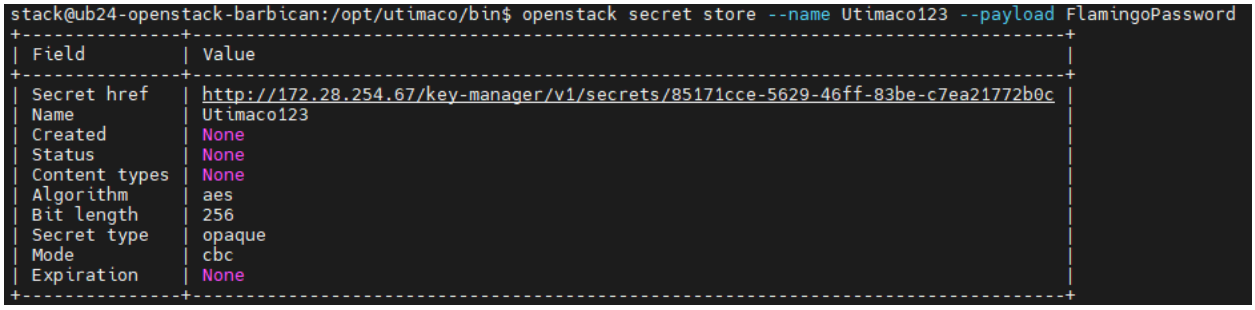

Figure 8 : Creating a secret in OpenStack Barbican

Here `Utimaco123` is the secret name and its value is `FlamingoPassword`. This secret is stored in an encrypted form in OpenStack Barbican.

3. You can also verify the encryption operation logging in PKCS11 log file `cs_pkcs11_R3.log` during secret generation as shown below.

```
stack@ub24-openstack-barbican:/opt/utimaco/bin$ tail -50 /tmp/cs_pkcs11_R3.log
24.02.2026 14:02:42.550 | [01784646:01784646] C_OpenSession                | T: enter C_OpenSession(slotID: 0x00000000, flags: 0x00000006, pApplication: 0,
24.02.2026 14:02:42.550 | [01784646:01784646] C_OpenSession                | T: leave C_OpenSession()
24.02.2026 14:02:42.550 | [01784646:01784646] C_GetSessionInfo             | T: enter C_GetSessionInfo(hSession: 0x00000002)
24.02.2026 14:02:42.550 | [01784646:01784646] C_GetSessionInfo             | T: leave C_GetSessionInfo()
24.02.2026 14:02:42.550 | [01784646:01784646] C_Login                      | T: enter C_Login(hSession: 0x00000002, userType: 0x00000001)
24.02.2026 14:02:42.599 | [01784646:01784646] C_Login                      | T: leave C_Login()
24.02.2026 14:02:42.610 | [01784646:01784646] C_FindObjectsInit            | T: enter C_FindObjectsInit(hSession: 0x00000002, pTemplate: CK_ATTRIBUTE[3])
24.02.2026 14:02:42.610 | [01784646:01784646] C_FindObjectsInit            | T: Attributes in: 3
                                                                               CKA_CLASS:
                                                                               CKA_KEY_TYPE:
                                                                               CKA_LABEL:
24.02.2026 14:02:42.611 | [01784646:01784646] C_FindObjectsInit            | T: leave C_FindObjectsInit()
24.02.2026 14:02:42.614 | [01784646:01784646] C_FindObjects                | T: enter C_FindObjects(hSession: 0x00000002, ulMaxObjectCount: 0x00000002)
24.02.2026 14:02:42.614 | [01784646:01784646] C_FindObjects                | T: leave C_FindObjects()
24.02.2026 14:02:42.614 | [01784646:01784646] C_FindObjectsFinal           | T: enter C_FindObjectsFinal(hSession: 0x00000002)
24.02.2026 14:02:42.614 | [01784646:01784646] C_FindObjectsFinal           | T: leave C_FindObjectsFinal()
24.02.2026 14:02:42.614 | [01784646:01784646] C_FindObjectsInit            | T: enter C_FindObjectsInit(hSession: 0x00000002, pTemplate: CK_ATTRIBUTE[3])
24.02.2026 14:02:42.614 | [01784646:01784646] C_FindObjectsInit            | T: Attributes in: 3
                                                                               CKA_CLASS:
                                                                               CKA_KEY_TYPE:
                                                                               CKA_LABEL:
24.02.2026 14:02:42.615 | [01784646:01784646] C_FindObjectsInit            | T: leave C_FindObjectsInit()
24.02.2026 14:02:42.615 | [01784646:01784646] C_FindObjects                | T: enter C_FindObjects(hSession: 0x00000002, ulMaxObjectCount: 0x00000002)
24.02.2026 14:02:42.615 | [01784646:01784646] C_FindObjects                | T: leave C_FindObjects()
24.02.2026 14:02:42.615 | [01784646:01784646] C_FindObjectsFinal           | T: enter C_FindObjectsFinal(hSession: 0x00000002)
24.02.2026 14:02:42.615 | [01784646:01784646] C_FindObjectsFinal           | T: leave C_FindObjectsFinal()
24.02.2026 14:02:42.636 | [01784646:01784646] C_GenerateKey                | T: enter C_GenerateKey(hSession: 0x00000002, pMechanism: CKM_AES_KEY_GEN (0x108
te: CK_ATTRIBUTE[d], phKey: 0x56e1f5af6e90)
24.02.2026 14:02:42.637 | [01784646:01784646] C_GenerateKey                | T: leave C_GenerateKey()
24.02.2026 14:02:42.641 | [01784646:01784646] C_GenerateRandom             | T: enter C_GenerateRandom(hSession: 0x00000002, pRandomData: CK_BYTE[10])
24.02.2026 14:02:42.641 | [01784646:01784646] C_GenerateRandom             | T: leave C_GenerateRandom()
24.02.2026 14:02:42.641 | [01784646:01784646] C_WrapKey                    | T: enter C_WrapKey(hSession: 0x00000002, pMechanism: CKM_AES_CBC_PAD (0x1085),
y: 0x00000001, hKey: 0x00000003)
24.02.2026 14:02:42.641 | [01784646:01784646] C_WrapKey                    | T: leave C_WrapKey()
24.02.2026 14:02:42.645 | [01784646:01784646] C_WrapKey                    | T: enter C_WrapKey(hSession: 0x00000002, pMechanism: CKM_AES_CBC_PAD (0x1085),
y: 0x00000001, hKey: 0x00000003)
24.02.2026 14:02:42.645 | [01784646:01784646] C_WrapKey                    | T: leave C_WrapKey()
24.02.2026 14:02:42.645 | [01784646:01784646] C_SignInit                   | T: enter C_SignInit(hSession: 0x00000002, pMechanism: CKM_SHA256_HMAC (0x251),
00002)
24.02.2026 14:02:42.645 | [01784646:01784646] C_SignInit                   | T: leave C_SignInit()
24.02.2026 14:02:42.653 | [01784646:01784646] C_Sign                       | T: enter C_Sign(hSession: 0x00000002)
24.02.2026 14:02:42.653 | [01784646:01784646] C_Sign                       | T: leave C_Sign()
24.02.2026 14:02:42.657 | [01784646:01784646] C_OpenSession                | T: enter C_OpenSession(slotID: 0x00000000, flags: 0x00000006, pApplication: 0,
24.02.2026 14:02:42.657 | [01784646:01784646] C_OpenSession                | T: leave C_OpenSession()
24.02.2026 14:02:42.657 | [01784646:01784646] C_GetSessionInfo             | T: enter C_GetSessionInfo(hSession: 0x00000003)
24.02.2026 14:02:42.657 | [01784646:01784646] C_GetSessionInfo             | T: leave C_GetSessionInfo()
24.02.2026 14:02:42.657 | [01784646:01784646] C_GenerateRandom             | T: enter C_GenerateRandom(hSession: 0x00000003, pRandomData: CK_BYTE[10])
24.02.2026 14:02:42.658 | [01784646:01784646] C_GenerateRandom             | T: leave C_GenerateRandom()
24.02.2026 14:02:42.658 | [01784646:01784646] C_EncryptInit                | T: enter C_EncryptInit(hSession: 0x00000003, pMechanism: CKM_AES_CBC (0x1082),
00003)
24.02.2026 14:02:42.658 | [01784646:01784646] C_EncryptInit                | T: leave C_EncryptInit()
24.02.2026 14:02:42.658 | [01784646:01784646] C_Encrypt                    | T: enter C_Encrypt(hSession: 0x00000003)
24.02.2026 14:02:42.658 | [01784646:01784646] C_Encrypt                    | T: leave C_Encrypt()
24.02.2026 14:02:42.658 | [01784646:01784646] C_CloseSession               | T: enter C_CloseSession(hSession: 0x00000003)
24.02.2026 14:02:42.658 | [01784646:01784646] C_CloseSession               | T: leave C_CloseSession()
```

Figure 9 : Key wrapping and secret encryption in PKCS#11 logs

The PKCS#11 logs show how Barbican uses the HSM to create and store secrets, including session setup, key generation, key wrapping, encryption, and HMAC operations, with the active cryptographic mechanisms (key wrapping, encryption, etc.) visible in the log entries. The initial secret creation includes one-time project setup (PKEK generation and wrapping under the MKEK), while subsequent secrets reuse this setup and primarily perform encryption and storage operations, resulting in shorter log sequences.

4. Retrieve the newly created secret metadata.

›_ Console

```
# openstack secret get <secret_href>
```

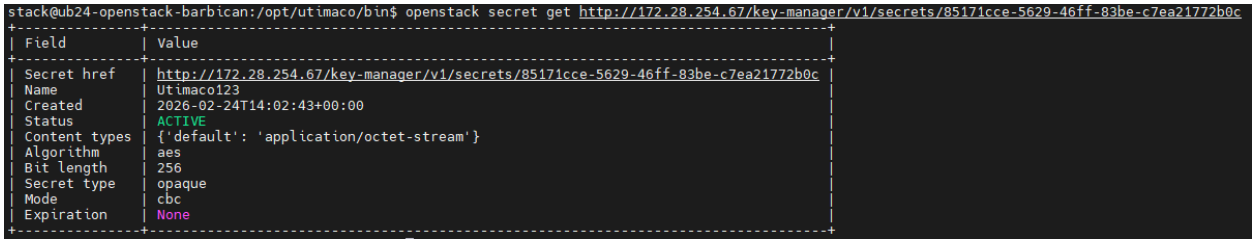

Figure 10 : Retrieving the newly created secret metadata

5. Retrieve the newly created secret payload.

›_ Console

```
# openstack secret get <secret_href> --payload
```

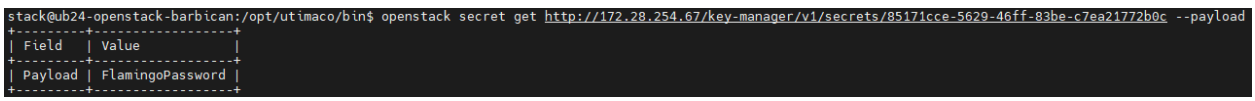

Figure 11 : Retrieving the newly created secret value

The secret is first decrypted and then displayed.

6. You can also verify the decryption operation logging in PKCS11 log file `cs_pkcs11_R3.log` during secret retrieval as shown below.

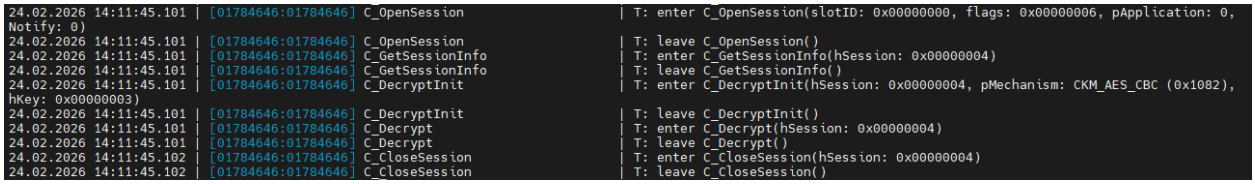

Figure 12 : Secret decryption in PKCS#11 logs

6.2 Generating a Symmetric Key in Barbican

1. Generate a new 256-bit key and store it in Barbican using `openstack order create` command.

`›_ Console`

```
# openstack secret order create --name app_key_flamingo --algorithm aes --mode ctr --bit-length 256 --payload-content-type=application/octet-stream key
```

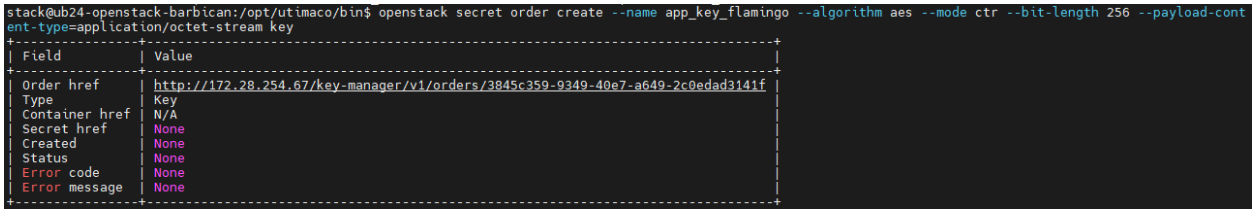

Figure 13 : Symmetric key created and stored on Barbican

2. You can also verify the encryption operation logging in the PKCS#11 log file `cs_pkcs11_R3.log` during secret generation, as shown below.

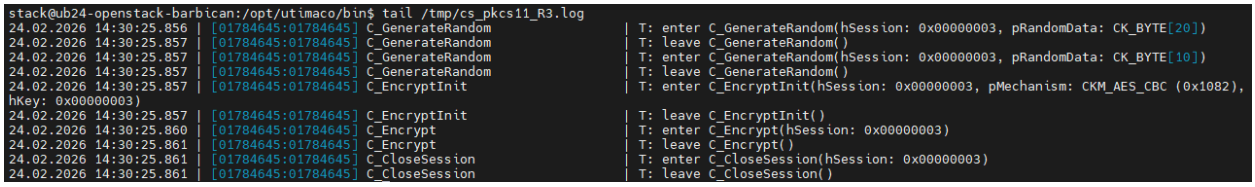

Figure 14 : Key encryption in PKCS#11 logs

3. Retrieve the metadata of the order to identify the location of the generated key, shown here as the `Secret href` value.

`›_ Console`

```
# openstack secret order get <order_href>
```

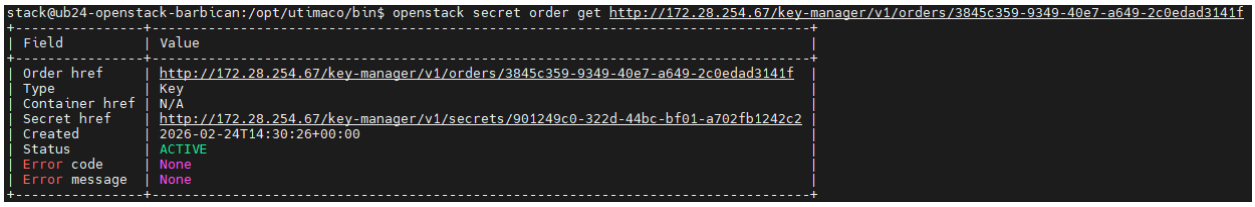

Figure 15 : Retrieving order metadata

4. Retrieve the secret metadata.

`›_ Console`

```
# openstack secret get <secret_href>
```

```
stack@ub24-openstack-barbican:/opt/utimaco/bin$ openstack secret get http://172.28.254.67/key-manager/v1/secrets/901249c0-322d-44bc-bf01-a702fb1242c2
+---------------+-----------------------------------------------------------------------------------+
| Field         | Value                                                                             |
+---------------+-----------------------------------------------------------------------------------+
| Secret href   | http://172.28.254.67/key-manager/v1/secrets/901249c0-322d-44bc-bf01-a702fb1242c2 |
| Name          | app_key_flamingo                                                                  |
| Created       | 2026-02-24T14:30:26+00:00                                                         |
| Status        | ACTIVE                                                                            |
| Content types | {'default': 'application/octet-stream'}                                           |
| Algorithm     | aes                                                                               |
| Bit length    | 256                                                                               |
| Secret type   | symmetric                                                                         |
| Mode          | ctr                                                                               |
| Expiration    | None                                                                              |
+---------------+-----------------------------------------------------------------------------------+
```

Figure 16 : Retrieving secret metadata

5. Alternatively, you can list the symmetric key that has been generated by using the `openstack secret list` command.

`›_ Console`

```
# openstack secret list
```

```
stack@ub24-openstack-barbican:/opt/utimaco/bin$ openstack secret list
+---------------------+-----------------+---------------------+--------+-----------------------+-----------+------------+-------------+------+------------+
| Secret href         | Name            | Created             | Status | Content types         | Algorithm | Bit length | Secret type | Mode | Expiration |
+---------------------+-----------------+---------------------+--------+-----------------------+-----------+------------+-------------+------+------------+
| http://172.28.254.6 | Utimaco123      | 2026-02-            | ACTIVE | {'default':           | aes       |        256 | opaque      | cbc  | None       |
| 7/key-manager/v1/se |                 | 24T14:02:43+00:00   |        | 'application/octet-   |           |            |             |      |            |
| crets/85171cce-     |                 |                     |        | stream'}              |           |            |             |      |            |
| 5629-46ff-83be-     |                 |                     |        |                       |           |            |             |      |            |
| c7ea21772b0c        |                 |                     |        |                       |           |            |             |      |            |
| http://172.28.254.6 | app_key_flamingo | 2026-02-           | ACTIVE | {'default':           | aes       |        256 | symmetric   | ctr  | None       |
| 7/key-manager/v1/se |                 | 24T14:30:26+00:00   |        | 'application/octet-   |           |            |             |      |            |
| crets/901249c0-     |                 |                     |        | stream'}              |           |            |             |      |            |
| 322d-44bc-          |                 |                     |        |                       |           |            |             |      |            |
| bf01-a702fb1242c2   |                 |                     |        |                       |           |            |             |      |            |
+---------------------+-----------------+---------------------+--------+-----------------------+-----------+------------+-------------+------+------------+
```

Figure 17 : Symmetric key in secret list

6.3 Storing Public Key, Private Key and Certificate in OpenStackBarbican

1. Create a new directory in the `stack` user folder. It will be used for storing the keys and certificate created in the next steps.

```
›_ Console

# mkdir ~/ssl

# cd ~/ssl
```

2. Create a self-signed certificate and private key using the command below.

```
›_ Console

# openssl req -x509 -newkey rsa:4096 -keyout privateFlamingo.pem -out
certFlamingo.pem -sha256 -days 365 -nodes
```

You can generate a key and a certificate by using other utilities as well.

3. Verify that the private key and certificate file are generated.

```
›_ Console

# ll
```

```
stack@ub24-openstack-barbican:~/ssl$ ll
total 16
drwxrwxr-x  2 stack stack 4096 Feb 24 14:41 ./
drwxr-x--x 23 stack stack 4096 Feb 24 14:39 ../
-rw-rw-r--  1 stack stack 2134 Feb 24 14:41 certFlamingo.pem
-rw-------  1 stack stack 3272 Feb 24 14:40 privateFlamingo.pem
```

Figure 18 : Certificate and private key created

4. Generate a public key from the private key.

›_ Console

```
# openssl rsa -in privateFlamingo.pem -pubout -out publicFlamingo.pem
```

```
stack@ub24-openstack-barbican:~/ssl$ openssl rsa -in privateFlamingo.pem -pubout -out publicFlamingo.pem
writing RSA key
stack@ub24-openstack-barbican:~/ssl$ ll
total 20
drwxrwxr-x  2 stack stack 4096 Feb 24 14:43 ./
drwxr-x--x 23 stack stack 4096 Feb 24 14:39 ../
-rw-rw-r--  1 stack stack 2134 Feb 24 14:41 certFlamingo.pem
-rw-------  1 stack stack 3272 Feb 24 14:40 privateFlamingo.pem
-rw-rw-r--  1 stack stack  800 Feb 24 14:43 publicFlamingo.pem
```

Figure 19 : Public key created from private key

5. Store the public key in OpenStack Barbican.

›_ Console

```
# openstack secret store --algorithm rsa --secret-type public --payload-content-type application/octet-stream --payload-content-encoding base64 --payload "$(base64 < publicFlamingo.pem)" --bit-length 2048 --name PubFlamingoTest
```

```
stack@ub24-openstack-barbican:~/ssl$ openstack secret store --algorithm rsa --secret-type public --payload-content-type application/octet-stream --payload-conte
nt-encoding base64 --payload "$(base64 < publicFlamingo.pem)" --bit-length 2048 --name PubFlamingoTest
+---------------+------------------------------------------------------------------------------------+
| Field         | Value                                                                              |
+---------------+------------------------------------------------------------------------------------+
| Secret href   | http://172.28.254.67/key-manager/v1/secrets/b0ba734c-f70b-4f9f-baf2-5560f6943593 |
| Name          | PubFlamingoTest                                                                    |
| Created       | None                                                                               |
| Status        | None                                                                               |
| Content types | {'default': 'application/octet-stream'}                                            |
| Algorithm     | rsa                                                                                |
| Bit length    | 2048                                                                               |
| Secret type   | public                                                                             |
| Mode          | cbc                                                                                |
| Expiration    | None                                                                               |
+---------------+------------------------------------------------------------------------------------+
```

Figure 20 : Public key stored in Barbican

6. You can also verify the encryption operation logging in the PKCS#11 log file `cs_pkcs11_R3.log` during public secret generation, as shown below.

```
24.02.2026 14:45:01.289 | [01784645:01784645] C_OpenSession                | T: enter C_OpenSession(slotID: 0x00000000, flags: 0x00000006, pApplication: 0,
Notify: 0)
24.02.2026 14:45:01.289 | [01784645:01784645] C_OpenSession                | T: leave C_OpenSession()
24.02.2026 14:45:01.289 | [01784645:01784645] C_GetSessionInfo             | T: enter C_GetSessionInfo(hSession: 0x00000004)
24.02.2026 14:45:01.289 | [01784645:01784645] C_GetSessionInfo             | T: leave C_GetSessionInfo()
24.02.2026 14:45:01.289 | [01784645:01784645] C_GenerateRandom             | T: enter C_GenerateRandom(hSession: 0x00000004, pRandomData: CK_BYTE[10])
24.02.2026 14:45:01.289 | [01784645:01784645] C_GenerateRandom             | T: leave C_GenerateRandom()
24.02.2026 14:45:01.289 | [01784645:01784645] C_EncryptInit                | T: enter C_EncryptInit(hSession: 0x00000004, pMechanism: CKM_AES_CBC (0x1082),
hKey: 0x00000003)
24.02.2026 14:45:01.289 | [01784645:01784645] C_EncryptInit                | T: leave C_EncryptInit()
24.02.2026 14:45:01.299 | [01784645:01784645] C_Encrypt                    | T: enter C_Encrypt(hSession: 0x00000004)
24.02.2026 14:45:01.300 | [01784645:01784645] C_Encrypt                    | T: leave C_Encrypt()
24.02.2026 14:45:01.300 | [01784645:01784645] C_CloseSession               | T: enter C_CloseSession(hSession: 0x00000004)
24.02.2026 14:45:01.300 | [01784645:01784645] C_CloseSession               | T: leave C_CloseSession()
```

Figure 21 : Secret encryption in PKCS#11 logs

7. Retrieve the value of the public key.

`›_ Console`

```
# openstack secret get -p -c Payload -f value <secret_href>
```

```
stack@ub24-openstack-barbican:~/ssl$ openstack secret get -p -c Payload -f value http://172.28.254.67/key-manager/v1/secrets/b0ba734c-f70b-4f9f-baf2-5560f69435
3
-----BEGIN PUBLIC KEY-----
MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAshlnYlLpeCGw2TPdrr7w
tXbB3WmEZzx2DXsnTzcu1fXrmUPyx5Q5e0aH/Fr56ppCGiqhCaNGTtyiilwdRoLz
/tP+JE57yfD6OJqYDIk9obzjhTrgMQzrzb7n0PCO2q/sMDQXXdDAAjARQ8jyUIvC
ii6t2me3zLEFrsBMm00hGN0QRcyTqhW6HEVZFvTpBHgVlAPVWXa/GLzZnQmfCQRB
aa+cFVYD8OwLFssmEj8aavGwkLRR+kqrzklhDd9N0Wa+JP2dBOzCO5wTuottMSSp
oZxg4wMb9P4FmXFgMFhq82+pP2OqG9rAj1HzmLVfApTewWAeukTPoCMa2YvxMGvU
CS97G0erxP7TbNhfnTdvq5E5XznPXjVpPXwWWLpm5/7vlII2wuv/lXtuenUAuG7+
QNeAbAiCX9jpsb6zihhEs5gpKIJBKIPCFSVVE5IhNw2aCVPzNcoge962QYsKacgG
zhN5kbp6ko9lYiErYUqRBGmf1KpZFq/ScjfxMH9BjYeUDmcrDyNLRTGDFx7UMKFv
aq65z9Fgf5v0Xypw1ebv3VRHX2xTWvYs6bozh9HhoqFBc/VILKoiIMHHBMXjLgS1
UkCEZwiu4vhNI6aEBLPahIPYrTN6vcc5MrRl7fwY5R7GaQSEKBM8WLweUPEhAD8j
sT7YZlKVu+8Gm9lExvXlVaECAwEAAQ==
-----END PUBLIC KEY-----
```

Figure 22 : Retrieving public key value from Barbican

8. Store the private key in OpenStack Barbican.

›_ Console

```
# openstack secret store --algorithm rsa --secret-type private --payload-content-type application/octet-stream --payload-content-encoding base64 --payload "$(base64 < privateFlamingo.pem)" --bit-length 2048 --name PrivFlamingoTest
```

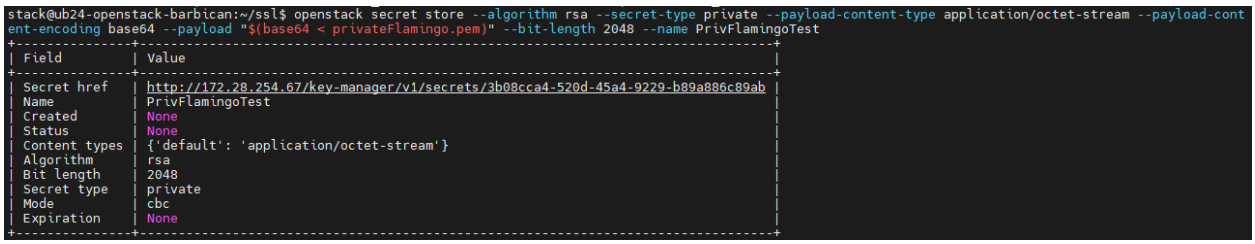

Figure 23 : Private key stored in Barbican

9. Retrieve the value of the private key.

›_ Console

```
# openstack secret get -p -c Payload -f value <secret_href>
```

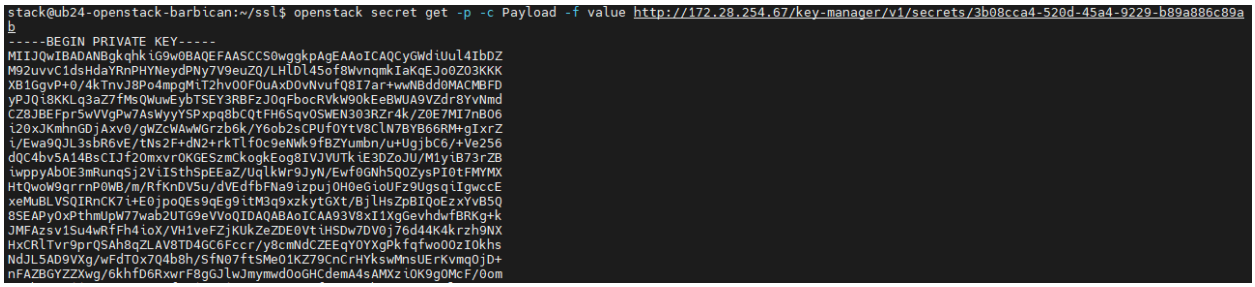

Figure 24 : Retrieving private key value from Barbican

10. Store the certificate in OpenStack Barbican.

›_ Console

```
# openstack secret store --algorithm rsa --secret-type certificate --payload-content-type application/octet-stream --payload-content-encoding base64 --payload "$(base64 < certFlamingo.pem)" --bit-length 2048 --name CertificateFlamingoTest
```

```
stack@ub24-openstack-barbican:~/ssl$ openstack secret store --algorithm rsa --secret-type certificate --payload-content-type application/octet-stream --payload-content-encoding base64 --payload "$(base64 < certFlamingo.pem)" --bit-length 2048 --name CertificateFlamingoTest
+---------------+------------------------------------------------------------------------------------+
| Field         | Value                                                                              |
+---------------+------------------------------------------------------------------------------------+
| Secret href   | http://172.28.254.67/key-manager/v1/secrets/1d0d67b3-fcfd-49f3-8800-57a06cd7703e |
| Name          | CertificateFlamingoTest                                                            |
| Created       | None                                                                               |
| Status        | None                                                                               |
| Content types | {'default': 'application/octet-stream'}                                            |
| Algorithm     | rsa                                                                                |
| Bit length    | 2048                                                                               |
| Secret type   | certificate                                                                        |
| Mode          | cbc                                                                                |
| Expiration    | None                                                                               |
+---------------+------------------------------------------------------------------------------------+
```

Figure 25 : Certificate stored in Barbican

11. Retrieve the value of the certificate from OpenStack Barbican.

›_ Console

```
# openstack secret get -p -c Payload -f value <secret_href>
```

```
stack@ub24-openstack-barbican:~/ssl$ openstack secret get -p -c Payload -f value http://172.28.254.67/key-manager/v1/secrets/1d0d67b3-fcfd-49f3-8800-57a06cd7703e
-----BEGIN CERTIFICATE-----
MIIF+zCCA+OgAwIBAgIUC0rDy1qH/y/oKpouB758uyImu4IwDQYJKoZIhvcNAQEL
BQAwgYwxCzAJBgNVBAYTAkRFMRMwEQYDVQQIDApTb21lLVN0YXRlMQ8wDQYDVQQH
DAZBYWNoZW4xEDAOBgNVBAoMB1V0aW1hY28xCzAJBgNVBAsMAklHMQwwCgYDVQQD
DANJR1UxKjAoBgkqhkiG9w0BCQEWG3Rlc3R1dGltYWNvQHRlc3R1dGltYWNvLmNv
bTAeFw0yNjAyMjQxMzhaFw0yNzAyMjQxNDQxMzhaMIGMMQswCQYDVQQGEwJE
RTETMBEGA1UECAwKU29tZS1TdGF0ZTEPMA0GA1UEBwwGQWFjaGVuMRAwDgYDVQQK
DAdVdGltYWNvMQswCQYDVQQLDAJJRzEMMAoGA1UEAwwDSUdVMSowKAYJKoZIhvcN
AQkBFht0ZXN0dXRpbWFjb0B0ZXN0dXRpbWFjby5jb20wggIiMA0GCSqGSIb3DQEB
AQUAA4ICDwAwggIKAoICAQCyGWdiUul4IbDZM92uvvC1dsHdaYRnPHYNeydPNy7V
9euZQ/LHlDl45of8WvnqmkIaKqEJo0ZO3KKKXB1GgvP+0/4kTnvJ8Po4mpgMiT2h
vOOFOuAxDOvNvufQ8I7ar+wwNBdd0MACMBFDyPJQi8KKLq3aZ7fMsQWuwEybTSEY
3RBFzJOqFbocRVkW90kEeBWUA9VZdr8YvNmdCZ8JBEFpr5wVVgPw7AsWyyYSPxpq
8bCQtFH6SqvOSWEN303RZr4k/Z0E7MI7nBO6i20xJKmhnGDjAxv0/gWZcWAwWGrz
b6k/Y6ob2sCPUfOYtV8ClN7BYB66RM+gIxrZi/Ewa9QJL3sbR6vE/tNs2F+dN2+r
kTlfOc9eNWk9fBZYumbn/u+UgjbC6/+Ve256dQC4bv5A14BsCIJf20mxvrOKGESz
mCkogkEog8IVJVUTkiE3DZoJU/M1yiB73rZBiwppyAbOE3mRunqSj2ViISthSpEE
```

Figure 26 : Retrieving certificate value from Barbican

12. List all created secrets.

```
›_ Console
```

```
# openstack secret list
```

```
stack@ub24-openstack-barbican:~/ssl$ openstack secret list
+----------------------+----------------------+----------------------+--------+------------------------+-----------+------------+-------------+------+------------+
| Secret href          | Name                 | Created              | Status | Content types          | Algorithm | Bit length | Secret type | Mode | Expiration |
+----------------------+----------------------+----------------------+--------+------------------------+-----------+------------+-------------+------+------------+
| http://172.28.254.6  | CertificateFlamingo  | 2026-02-             | ACTIVE | {'default':            | rsa       |       2048 | certificate | cbc  | None       |
| 7/key-manager/v1/se  | Test                 | 24T14:55:16+00:00    |        | 'application/octet-    |           |            |             |      |            |
| crets/1d0d67b3-      |                      |                      |        | stream'}               |           |            |             |      |            |
| fcfd-49f3-8800-      |                      |                      |        |                        |           |            |             |      |            |
| 57a06cd7703e         |                      |                      |        |                        |           |            |             |      |            |
| http://172.28.254.6  | PrivFlamingoTest     | 2026-02-             | ACTIVE | {'default':            | rsa       |       2048 | private     | cbc  | None       |
| 7/key-manager/v1/se  |                      | 24T14:50:53+00:00    |        | 'application/octet-    |           |            |             |      |            |
| crets/3b08cca4-      |                      |                      |        | stream'}               |           |            |             |      |            |
| 520d-45a4-9229-      |                      |                      |        |                        |           |            |             |      |            |
| b89a886c89ab         |                      |                      |        |                        |           |            |             |      |            |
| http://172.28.254.6  | Utimaco123           | 2026-02-             | ACTIVE | {'default':            | aes       |        256 | opaque      | cbc  | None       |
| 7/key-manager/v1/se  |                      | 24T14:02:43+00:00    |        | 'application/octet-    |           |            |             |      |            |
| crets/85171cce-      |                      |                      |        | stream'}               |           |            |             |      |            |
| 5629-46ff-83be-      |                      |                      |        |                        |           |            |             |      |            |
| c7ea21772b0c         |                      |                      |        |                        |           |            |             |      |            |
| http://172.28.254.6  | app_key_flamingo     | 2026-02-             | ACTIVE | {'default':            | aes       |        256 | symmetric   | ctr  | None       |
| 7/key-manager/v1/se  |                      | 24T14:30:26+00:00    |        | 'application/octet-    |           |            |             |      |            |
| crets/901249c0-      |                      |                      |        | stream'}               |           |            |             |      |            |
| 322d-44bc-           |                      |                      |        |                        |           |            |             |      |            |
| bf01-a702fb1242c2    |                      |                      |        |                        |           |            |             |      |            |
| http://172.28.254.6  | PubFlamingoTest      | 2026-02-             | ACTIVE | {'default':            | rsa       |       2048 | public      | cbc  | None       |
| 7/key-manager/v1/se  |                      | 24T14:45:01+00:00    |        | 'application/octet-    |           |            |             |      |            |
| crets/b0ba734c-      |                      |                      |        | stream'}               |           |            |             |      |            |
| f70b-4f9f-baf2-      |                      |                      |        |                        |           |            |             |      |            |
| 5560f6943593         |                      |                      |        |                        |           |            |             |      |            |
+----------------------+----------------------+----------------------+--------+------------------------+-----------+------------+-------------+------+------------+
```

Figure 27 : Listing all secrets

6.4 Key Rotation/Migration

KEK Rewrap Compatibility

In OpenStack Barbican Flamingo, the PKCS#11 `unwrap_key()` interface requires an explicit key wrap mechanism parameter, while the `rewrap_pkek` utility still uses an older call signature. Without alignment, PKEK rewrapping fails. A small compatibility fix is required to pass the configured key wrapping mechanism mechanism to `unwrap_key()` .

The compatibility fix is performed with the following steps:

1. Locate the `pkcs11_kek_rewrap.py` script (default location is `/opt/stack/barbican/barbican/cmd/pkcs11_kek_rewrap.py`).

2. Find the original code.

`pkcs11_kek_rewrap.py`

```
current_kek = self.pkcs11.unwrap_key(kek_mkek, iv, wrapped_key, session)
```

3. Update the code.

`pkcs11_kek_rewrap.py`

```
current_kek = self.pkcs11.unwrap_key(self.pkcs11.key_wrap_mechanism,
kek_mkek, iv, wrapped_key, session)
```

4. Save the changes.

Perform the key rotation using the following steps:

1. Create a new MKEK using the `barbican-manage hsm gen_mkek` command.

`›_ Console`

```
# sudo -u stack -i barbican-manage hsm gen_mkek --library-path '/opt/
utimaco/lib/libcs_pkcs11_R3.so' --passphrase <CryptoUser_PIN>
--slot-id <slot_id> --label 'mkek_utimaco_rewrap' --length 32
```

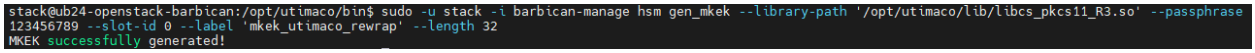

Figure 28 : Rewrap MKEK generated

2. Create a new HMAC using the `barbican-manage hsm gen_hmac` command.

`›_ Console`

```
# sudo -u stack -i barbican-manage hsm gen_mkek --library-path '/opt/
utimaco/lib/libcs_pkcs11_R3.so' --passphrase <CryptoUser_PIN>
--slot-id <slot_id> --label 'hmac_utimaco_rewrap' --length 32
```

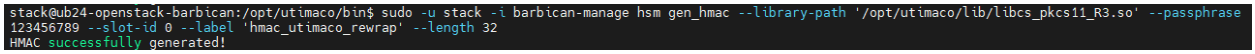

Figure 29 : Rewrap HMAC generated

3. Verify that the keys are generated on the Utimaco HSM using the `p11tool2 ListObjects` command.

`›_ Console`

```
# ./p11tool2 slot=<slot_id> LoginUser=<CryptoUser_PIN> ListObjects
```

```
stack@ub24-openstack-barbican:/opt/utimaco/bin$ ./p11tool2 Slot=0 LoginUser=123456789 ListObjects

CKO_SECRET_KEY:

+ 1.1
  CKA_KEY_TYPE                  = CKK_AES
  CKA_UNIQUE_ID                 = AFD1DA99-7671-4A26-BD8B-2CE7962D8E70
  CKA_SENSITIVE                 = CK_TRUE
  CKA_EXTRACTABLE               = CK_FALSE
  CKA_LABEL                     = mkek_utimaco
  CKA_ID                        =

+ 1.2
  CKA_KEY_TYPE                  = CKK_AES
  CKA_UNIQUE_ID                 = 3293A71D-11E9-416C-ABB2-472E76CDD81A
  CKA_SENSITIVE                 = CK_TRUE
  CKA_EXTRACTABLE               = CK_FALSE
  CKA_LABEL                     = hmac_utimaco_rewrap
  CKA_ID                        =

+ 1.3
  CKA_KEY_TYPE                  = CKK_AES
  CKA_UNIQUE_ID                 = 9BCDC5EB-1FDD-45AD-8B40-538612A7AEEB
  CKA_SENSITIVE                 = CK_TRUE
  CKA_EXTRACTABLE               = CK_FALSE
  CKA_LABEL                     = hmac_utimaco
  CKA_ID                        =

+ 1.4
  CKA_KEY_TYPE                  = CKK_AES
  CKA_UNIQUE_ID                 = 178D5B1A-2046-41CC-94A3-1F0013E0594C
  CKA_SENSITIVE                 = CK_TRUE
  CKA_EXTRACTABLE               = CK_FALSE
  CKA_LABEL                     = mkek_utimaco_rewrap
  CKA_ID                        =
```

Figure 30 : Listing rewrap keys with p11tool2

4. Edit the `barbican.conf` file located in `/etc/barbican` using your preferred text editor and change the `mkek_label` and `hmac_label` parameter values to the labels used when creating the new MKEK and HMAC.

`barbican.conf`

```
# Master KEK and HMAC key labels (as stored in the HSM) (string value)
mkek_label = mkek_utimaco_rewrap
hmac_label = hmac_utimaco_rewrap
```

5. Restart OpenStack Barbican service.

›_ Console

```
# sudo systemctl restart devstack@barbican-svc.service
```

6. Optional: Run the `rewrap_pkek` command with `--dry-run` option to display the changes that will be made.

›_ Console

```
# sudo -u stack -i barbican-manage hsm rewrap_pkek --dry-run
```

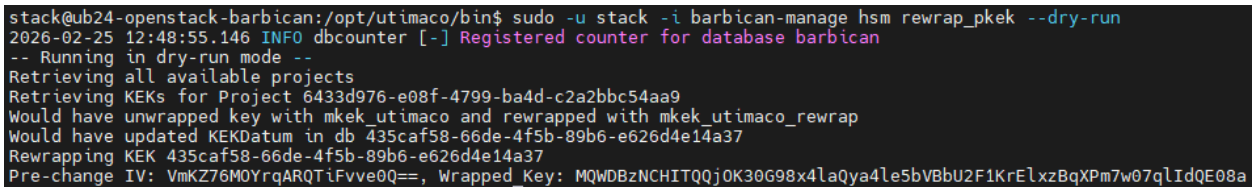

Figure 31 : Rewrapping PKEK (dry run)

7. Run the `rewrap_pkek` command to rewrap PKEK with the new MKEK.

›_ Console

```
# sudo -u stack -i barbican-manage hsm rewrap_pkek
```

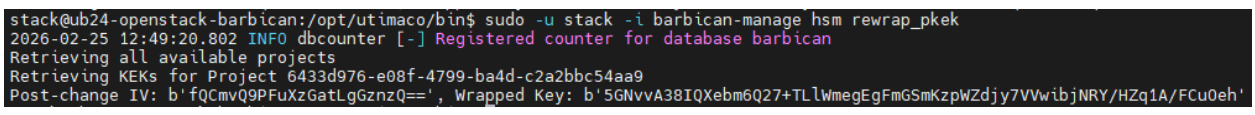

Figure 32 : Rewrapping PKEK

8. You can verify the rewrap operation logging in the PKCS#11 log file `cs_pkcs11_R3.log`.

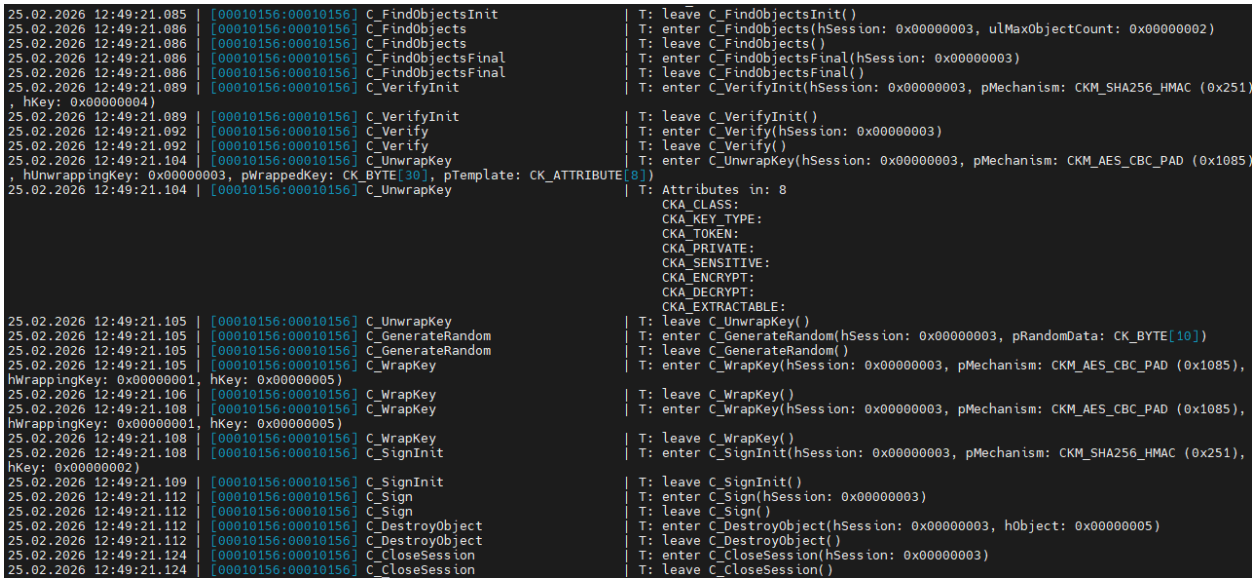

Figure 33 : PKEK rewrapping in PKCS#11 logs

9. Verify that you can retrieve the payloads of the secrets generated earlier.

`›_ Console`

```
# openstack secret get <secret_href> --payload
```

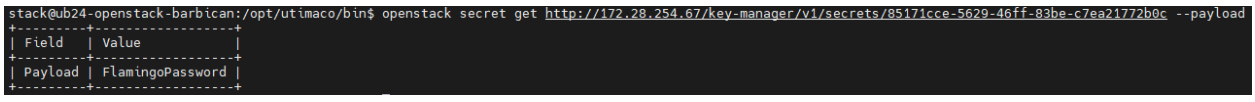

Figure 34 : Retrieving secret value after PKEK rewrap

6.5 Using Different Encryption Mechanisms

The encryption algorithm used to encrypt secret payloads before they are stored in the database is configurable and depends on the PKCS#11 mechanism supported by the connected HSM.

By default, AES in CBC mode (CKM_AES_CBC) is used to encrypt the payloads. However, Barbican and u.Trust GP HSM also support additional encryption mechanisms, such as AES in GCM mode (CKM_AES_GCM).

Follow these steps to change the encryption mechanism, store a secret, and verify its use:

1. Edit the Barbican configuration file `barbican.conf` using your preferred text editor and update the `encryption_mechanism` parameter under `p11_crypto_plugin]`.

`barbican.conf`

```
# Secret encryption mechanism (string value) - can be changed with secrets
present in the project:
# CKM_AES_CBC (default) or CKM_AES_GCM
encryption_mechanism = CKM_AES_GCM
```

Save the file after making the change.

It is important to note that changing the encryption mechanism affects only newly created secrets. Existing secrets remain encrypted using the mechanism that was active at the time of their creation and can not be decrypted with a different mechanism.

2. Restart the OpenStack Barbican service.

`›_ Console`

```
# sudo systemctl restart devstack@barbican-svc.service
```

3. Generate a new secret.

›_ Console

```
# openstack secret store --name Utimaco123_GCM --payload
FlamingoPassword_GCM
```

```
stack@ub24-openstack-barbican:/opt/utimaco/bin$ openstack secret store --name Utimaco123_GCM --payload FlamingoPassword_GCM
+---------------+------------------------------------------------------------------------------------+
| Field         | Value                                                                              |
+---------------+------------------------------------------------------------------------------------+
| Secret href   | http://172.28.254.67/key-manager/v1/secrets/9ab7f307-f282-43f8-8534-4f7c3b1c69c0 |
| Name          | Utimaco123_GCM                                                                     |
| Created       | None                                                                               |
| Status        | None                                                                               |
| Content types | None                                                                               |
| Algorithm     | aes                                                                                |
| Bit length    | 256                                                                                |
| Secret type   | opaque                                                                             |
| Mode          | cbc                                                                                |
| Expiration    | None                                                                               |
+---------------+------------------------------------------------------------------------------------+
```

Figure 35 : Storing a secret after changing the encryption mechanism

4. Verify the encryption mechanism used via the PKCS#11 log file `cs_pkcs11_R3.log`.

```
stack@ub24-openstack-barbican:/opt/utimaco/bin$ tail /tmp/cs_pkcs11_R3.log
25.02.2026 12:55:15.735 | [00011256:00011256] C_GetSessionInfo               | T: enter C_GetSessionInfo(hSession: 0x00000003)
25.02.2026 12:55:15.735 | [00011256:00011256] C_GetSessionInfo               | T: leave C_GetSessionInfo()
25.02.2026 12:55:15.738 | [00011256:00011256] C_GenerateRandom               | T: enter C_GenerateRandom(hSession: 0x00000003, pRandomData: CK_BYTE[c])
25.02.2026 12:55:15.739 | [00011256:00011256] C_GenerateRandom               | T: leave C_GenerateRandom()
25.02.2026 12:55:15.742 | [00011256:00011256] C_EncryptInit                  | T: enter C_EncryptInit(hSession: 0x00000003, pMechanism: CKM_AES_GCM (0x1087),
hKey: 0x00000003)
25.02.2026 12:55:15.742 | [00011256:00011256] C_EncryptInit                  | T: leave C_EncryptInit()
25.02.2026 12:55:15.745 | [00011256:00011256] C_Encrypt                      | T: enter C_Encrypt(hSession: 0x00000003)
25.02.2026 12:55:15.745 | [00011256:00011256] C_Encrypt                      | T: leave C_Encrypt()
25.02.2026 12:55:15.745 | [00011256:00011256] C_CloseSession                 | T: enter C_CloseSession(hSession: 0x00000003)
25.02.2026 12:55:15.745 | [00011256:00011256] C_CloseSession                 | T: leave C_CloseSession()
```

5. Verify the encryption mechanism used via Barbican database.

›_ Console

```
# mysql
```

›_ mysql console

```
# USE barbican;

# SELECT secret_id, kek_meta_extended FROM encrypted_data ORDER BY
created_at DESC;
```

The newly created secret will appear at the top of the table.

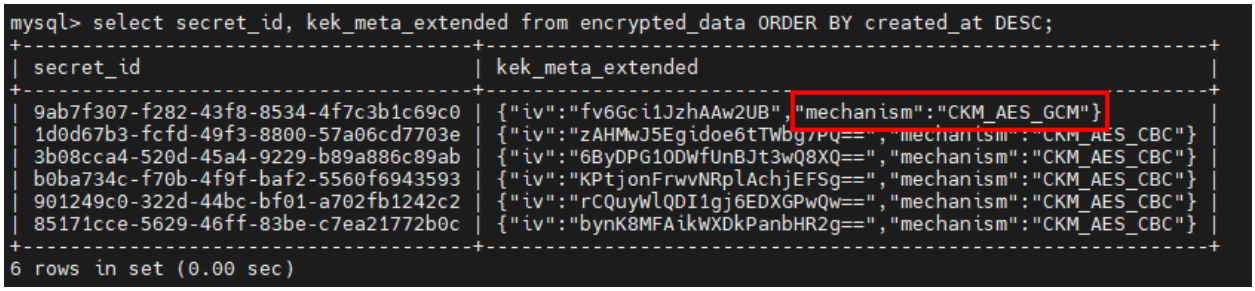

Figure 36 : Table showing encryption mechanism used

6. Confirm that the secret payload can be retrieved.

›_ Console

```
# openstack secret get <secret_href> --payload
```

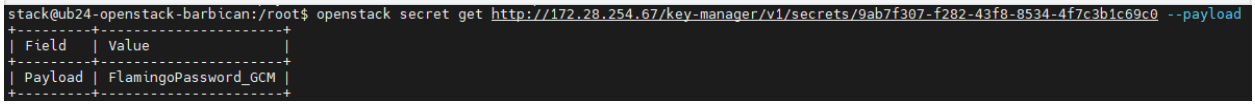

Figure 37 : Retrieving secret value with correct encryption mechanism

7. Change the `encryption_mechanism` parameter in the `barbican.conf` back to `CKM_AES_CBC` and restart the Barbican service.
8. Attempting to retrieve the secret payload again results in an internal server error.

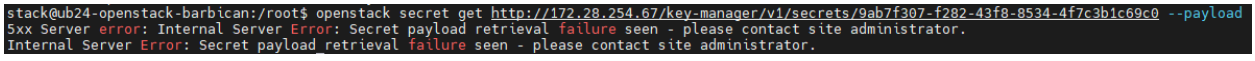

Figure 38 : Internal server error after attempting to retrieve payload with wrong mechanism

9. An error also appears in the PKCS#11 logs that indicates that the decryption operation was attempted using parameters that do not match the original encryption mechanism. In this case the parameter is the initialization vector (IV) format and length as CKM_AES_CBC requires a 16-byte IV and CKM_AES_GCM requires a GCM parameter structure containing an IV (typically 12 bytes). If an IV does not match the requirements of the selected mechanism, the HSM rejects the operation.

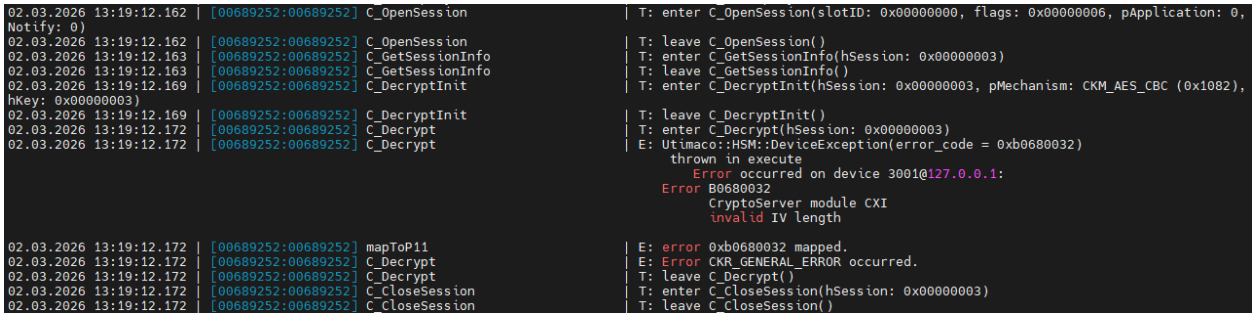

Figure 39 : Invalid IV length error in PKCS#11 logs

7 Troubleshooting

7.1 Common Issues and How to Resolve Them

Error	Diagnosis
C_Login [type=1] returned Error 0x00000102 (CKR_USER_PIN_NOT_INITIALIZED)	PKCS#11 slot is not initialized.
Error:Slot 0000 0000: p11cat.P11.getAuthState(Native Method): CS_GetSessionInfo returned Error 0x00000030 (CKR_DEVICE_ERROR)	1. Verify HSM services are up and running. 2. Check the IP entry is correct in `cs_pkcs11_R3.cfg` file.
Internal Server Error: Secret payload retrieval failure seen - please contact site administrator.	1. Check `barbican.log` if HMAC and MKEK labels are correct. 2. Restart Barbican service
Missing value auth-url required for auth plugin password	OpenStack admin credentials are not loaded. Resolve by running `# source ~/devstack/openrc admin admin`

Table 6: List of hardware requirements

7.2 SecurityServer PKCS#11 Logs

Enabling PKCS#11 logging to facilitate easier testing and troubleshooting is recommended. This can be done by configuring the `Loglevel` and `LogPath` parameters in the configuration file.

- `LogPath` should point to a writable directory (not a specific file) where log files can be stored.
- `Loglevel` controls the verbosity of the logs:
- Set it to `1` for basic logging.
- For detailed testing and debugging, increase the level to `4`.

The generated log file will be named `cs_pkcs11_R3.log` and located in the directory specified by `LogPath`. Reviewing this log file can help identify and resolve issues that arise during testing.

Once testing is complete, it is advisable to reduce the `Loglevel` to `1` or `2` to limit logging to only critical or important messages, thereby optimizing performance and reducing unnecessary log data.

7.3 OpenStack Barbican Logs

To facilitate testing and troubleshooting during integration, it is recommended to enable verbose logging in Barbican. Logging behavior is controlled through the `barbican.conf` configuration file. During initial setup and debugging, the global log level can be set to `debug = true` to provide detailed insight into request handling, PKCS#11 interactions, and cryptographic operations. Barbican logs are written to the DevStack log directory (typically `/opt/stack/logs/`), with the primary log file named `barbican.log`.

The following configuration options are commonly used:

- `debug = true`
  Enables debug-level logging.
- `use_syslog = false`
  Ensures logs are written to file rather than the system journal.

Once testing is complete, it is advisable to set `debug = false` to limit log volume and reduce operational overhead.

8 Contact and Support Information

If a chapter titled "**Contact and Support Information**" does not yet exist, create it as the last top-level chapter and insert the following content:

You can reach us from Monday to Friday, 09.00 a.m. to 05.00 p.m., Central European Time (CET).

Utimaco IS GmbH
Krefelder Str. 220
52070 Aachen
Germany

RMA Query

If you need to send the device back to Utimaco IS GmbH, please open a new RMA case (Return Merchandise Authorization). We request that you use the following web address. RMA cases cannot be opened by email or phone.

https://support.hsm.utimaco.com/support/rma/new

Other Support Queries

- Mail (preferred contact method)
  support@utimaco.com
  Attach the diagnostic information to your email.
- Web portal
  https://support.hsm.utimaco.com/support/cases/new/
  The diagnostic information will be requested in our response if necessary.
- By phone
  AMERICAS +1-844-UTIMACO (+1 844-884-6226)
  EMEA +49 800-627-3081
  APAC +81 800-919-1301
  The diagnostic information will be requested in our response if necessary.