Cosmian KMS

Integration Guide

CryptoServer

utimaco®

Imprint

Copyright 2020	Utimaco IS GmbH
	Germanusstr. 4
	D-52080 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	06/10/2025
Status	**PUBLISHED**
Document No.	IG-2025-0019

All rights reserved

No part of this documentation may be reproduced in any form (printing, photocopy or according to any other process) without the written approval of Utimaco IS GmbH or be processed, reproduced or distributed using electronic systems.

Utimaco IS GmbH reserves the right to modify or amend the documentation at any time without prior notice. Utimaco IS GmbH assumes no liability for typographical errors and damages incurred due to them.

All trademarks and registered trademarks are the property of their respective owners.

Table of Contents

1 Introduction

Thank you for purchasing our CryptoServer security system. We hope you are satisfied with our product. Please do not hesitate to contact us if you have any complaints or comments.

1.1 About This Manual

This manual describes how to integrate the Cosmian KMS with CryptoServer.

1.1.1 Target Audience for This Manual

This manual is intended for security officers, deploying or operating security devices such as HSMs and Key Management Systems.

1.1.2 Contents of This Manual

This manual describes the motivations for integrating the Cosmian KMS into Utimaco CryptoServer and the technical steps required to perform that integration.

Chapter 2 provides an overview of the motivations and use cases

Chapter 3 describes the necessary configuration steps for the integration

Chapter 4 shows how to manipulate keys and perform operations via the KMIP interface

1.1.3 Document Conventions

We use the following conventions in this manual:

Convention	Use	Example
Bold	Items of the Graphical User Interface (GUI), e.g., menu options	Press the **OK** button.
`Monospaced`	File names, folder and directory names, commands, file outputs, programming code samples	You will find the file `example.conf` in the `/exmp/demo/` directory.

Convention	Use	Example
Italic	References and important terms	See Chapter 3, "Sample Chapter", in the *CryptoServer - csadm Manual* or [CSADMIN].

Table 1: Document conventions

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

Here you find important safety information that should be followed.

Here you find additional notes or supplementary information.

1.1.4 Abbreviations

We use the following abbreviations in this manual:

Abbreviation	Meaning
HSM	Hardware Security Module
KMS	Key Management System
PKCS#11	Public-Key Cryptography Standards that defines a C programming interface

Table 2: List of Abbreviations

2 Motivation and use cases

Integrating Cosmian KMS and Utimaco CryptoServer is most useful for demanding client-side and application-level encryption scenarios where key material security is paramount.

2.1 Cosmian KMS

Cosmian KMS (https://docs.cosmian.com/key_management_system/) is an open-source key management system that delivers exceptional performance by enabling massive client-side encryption on the fly.

Cosmian KMS offers a KMIP 2.1 interface, simple horizontal and vertical scaling, and ready-made connectors for various existing applications.

The KMS is available in multiple packages and multiple operating systems. It has also been designed to run confidentially in the cloud by leveraging Cosmian VM, a hardened OS that runs on confidential computing machines that use encrypted disk and memory. Pre-packaged cloud KMS are available on the major cloud providers' marketplaces (https://cosmian.com/data-protection-suite/cosmian-kms/)

2.2 Use Cases

The integration is most useful in the following scenarios:

- When high volume, highly concurrent on-the-fly encryption/decryption is required on client-side or application-level encryption scenarios and maximum security for keys is needed. This use case includes:
  - Big Data applications in the cloud (Snowflake, Databricks, etc..). where processing of big data during querying may require on-the-fly decryption of millions of rows to fulfill the query.
  - Workplace applications (Google Workspace, Office 365) when they are protected by client-side encryption (respectively Google CSE and Microsoft DKE). In this scenario, a large number of user machines may concurrently access the KMS at any time.
- The software providing the encryption/decryption oracle (Cosmian KMS) cannot be co-located with the hardware security module. Such a scenario includes running the Cosmian KMS in confidential computing in the cloud for reduced latency while keeping the CryptoServer in a different location for better security.

The Cosmian KMS is a scalable software solution that offers large-scale, real-time encryption and decryption. The Utimaco CryptoServer delivers robust security for all key materials the KMS handles.

2.3 Operating principles

The integration of Cosmian KMS and Utimaco CryptoServer is performed via a PKCS#11 interface. Multiple instances of KMS servers may be connected via this interface to the same CryptoServer.

The integration offers the possibility to:

- Create operating keys in the KMS wrapped (encrypted) by master keys stored in the HSM. At rest, master keys in the HSM are protected by certified hardware, while the operating keys in the KMS in the KMS database are stored encrypted by the master keys. At runtime, when an encryption or decryption request is processed by the KMS, the KMS will first request the HSM to unwrap the operating key and then perform data operation using the operating key. The latter key is kept in the KMS memory for the subsequent requests.
- Perform various KMIP operations (see chapter 4) on the HSM, directly via the KMS KMIP interface.

3 Configuration

This solution works on Linux (x64_86) and has been validated against the Utimaco libcs_pkcs11_R3.so library version 6.0.

3.1 Utimaco library setup

On every machine running the Cosmian KMS, the server application expects:

- the Utimaco `cs_pkcs11_R3` library to be installed in /lib/libcs_pkcs11_R3.so
- the Utimaco configuration file `cs_pkcs11_R3.cfg` to be in /etc/utimaco and
- and the environment variable `CS_PKCS11_R3_CF` to point to it, i.e.,

›_ Console

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

Please make sure the `cs_pkcs11_R3.cfg` is set with the correct parameter, and validate your installation with the `p11tool2` utility by running, for instance,

›_ Console

```
./p11tool2 Slot=0 GetSlotInfo
```

3.2 KMS configuration

At least one slot and its corresponding password must be configured. Any slot and any number of slots may be used.

When using the TOML configuration file, the HSM support is enabled by configuring these 4 parameters:

kms.toml

```
hsm_model = "utimaco"

hsm_admin = "<HSM_ADMIN_USERNAME>" # defaults to "admin"

hsm_slot = [0, 0, ] # example [0,4] for slots 0 and 4

hsm_password = ["<password>", "<password>", ] # example ["pass0", "pass4"] for slots 0 and 4
```

Even if only one slot is used, the hsm_slot and hsm_password parameters must be arrays.

When the KMS is started from the command line, the HSM support can be enabled by using the following arguments:

```
›_ Console

--hsm-model "utimaco" \

--hsm-admin "<HSM_ADMIN_USERNAME>" \

--hsm-slot <number_of_1st_slot> --hsm-password <password_of_1st_slot> \

--hsm-slot <number_of_2and_slot> --hsm-password <password_of_2and_slot>
```

The `hsm-model` parameter is the HSM model to be used; use utimaco.

The `hsm-admin` parameter is the username of the HSM administrator. The HSM administrator is the only user that can create objects on the HSM via the KMIP Create operation the delegate other operations to other users. (see below)

The `hsm-slot` and `hsm-password` parameters are the slot number and user password of the HSM slots to be used by the KMS. These arguments can be repeated multiple times to specify multiple slots.

3.3 HSM keys

HSM keys are prefixed keys. They are created with a unique identifier that is pre-fixed by the hsm keyword and the slot number in the form: `hsm::<slot_number>::<key_identifier>`

For instance, the key `hsm::1::mykey` is a key stored in the HSM slot 1 with the identifier `mykey`. Technically, this identifier is stored in the `LABEL` field of the key object in the HSM.

Non-prefixed keys are considered KMS keys and are stored in the KMS database.

3.4 KMIP operations

Some KMIP operations can be performed via the KMS server API on HSM keys.

First, ensure the HSM is configured, and the Cosmian CLI is installed and configured.

3.4.1 Create

Create a new key in the HSM. The key unique identifier must be provided on the request and follow the `hsm::<slot_number>::<key_identifier>` format described above. Only the user identified by the `hsm-admin` argument can create keys in the HSM.

RSA and AES keys are supported. When creating an RSA key, the `key_identifier` will be that of the private key. The corresponding public key will be automatically created and stored in the HSM with the same `key_identifier`, but with the `_pk` suffix; for example, the public key of the `hsm::1::mykey` private key will be created with unique identifier `hsm::1::mykey_pk`.

Create an RSA 4096-bit key with the Cosmiian CLI:

```
❯ cosmian kms rsa keys create --size_in_bits 4096 --sensitive hsm::4::my_rsa_key

The RSA key pair has been created.

      Public key unique identifier: hsm::4::my_rsa_key_pk

      Private key unique identifier: hsm::4::my_rsa_key
```

Create an AES 256-bit key with the Cosmiian CLI:

```
❯ cosmian kms sym keys create --algorithm aes --number-of-bits 256 --sensitive
hsm::4::my_aes_key

The symmetric key was successfully generated.

   Unique identifier: hsm::4::my_aes_key
```

Keys should be flagged as sensitive when created in the HSM, so that the private key or symmetric key cannot be exported (see below Get and Export).

HSM keys do not support object tagging in this release.

3.4.2 Destroy

Contrarily to the KMS keys, HSM keys must not be Revoked before being Destroyed. The Destroy operation will remove the key from the HSM.

Only the user identified by the `hsm-admin` argument or a user granted the Destroy operation (by the HSM admin) can destroy keys in the HSM.

To destroy the key `hsm::4::my_rsa_key`, the following command can be used:

```
❯ cosmian kms rsa keys destroy --key-id hsm::4::my_rsa_key

Successfully destroyed the key.

      Unique identifier: hsm::4::mykey
```

To destroy the corresponding public `key hsm::4::my_rsa_key_pk`, the following command can be used:

```
❯ cosmian kms rsa keys destroy --key-id hsm::4::my_rsa_key_pk

Successfully destroyed the object.

   Unique identifier: hsm::4::my_rsa_key_pk
```

3.4.3 Get & Export

The Get and Export operations retrieve the key material from the HSM. Only the user identified by the `hsm-admin` argument or a user granted the Get operation (by the HSM admin) can retrieve keys from the HSM.

Private keys or symmetric keys marked as sensitive cannot be retrieved from the HSM. The public key of a keypair can always be retrieved.

To export the public key `hsm::4::my_rsa_key_pk` in PKCS#8 PEM format, the following command can be used:

```
❯ cosmian kms rsa keys export --key-id hsm::4::my_rsa_key_pk --key-format pkcs8-
pem /tmp/pubkey.pem

The key hsm::4::my_rsa_key_pk of type PublicKey was exported to "/tmp/pubkey.pem"

   Unique identifier: hsm::4::my_rsa_key_pk
```

To export the private key `hsm::4::mykey` in PKCS#8 PEM format, the following command can be used:

```
❯ cosmian kms rsa keys export --key-id hsm::4::my_rsa_key --key-format pkcs8-pem /
tmp/privkey.pem

The key hsm::4::my_rsa_key of type PrivateKey was exported to "/tmp/privkey.pem"

   Unique identifier: hsm::4::my_rsa_key
```

To export the symmetric key `hsm::4::my_aes_key` in raw format (i.e. raw bytes), the following command can be used:

```
❯ cosmian kms sym keys export --key-id hsm::4::my_aes_key --key-format raw /tmp/
symkey.raw

The key hsm::4::my_aes_key of type SymmetricKey was exported to "/tmp/symkey.raw"

   Unique identifier: hsm::4::my_aes_key
```

3.4.4 Encrypt

Symmetric keys and public keys can be used to encrypt data. Only the user identified by the `--hsm-admin` argument or a user granted the Encrypt operation (by the HSM admin) can encrypt data with keys stored in the HSM.

For symmetric keys, only AES GCM is supported. CKM_RSA_PKCS_OAEP and the now deprecated but still widely used CKM_RSA_PKCS (v1.5) are supported for RSA keys. The hashing algorithm is fixed to SHA256.

When using RSA, the maximum message size in bytes is:

- PKCS#1 v1.5: (key size in bits / 8) - 11
- OAEP: (key size in bits / 8) - 66

To encrypt a message with the public `key hsm::4::my_rsa_key_pk` and the CKM RSA PKCS OAEP algorithm, the following command can be used:

```
❯ cosmian kms rsa encrypt --key-id hsm::4::my_rsa_key_pk --encryption-algorithm
ckm-rsa-pkcs-oaep \

/tmp/secret.txt
```

The encrypted file is available at `/tmp/secret.enc`

To encrypt a message using AES GCM with the symmetric `key hsm::4::my_aes_key`, the following command can be used:

```
❯ cosmian kms sym encrypt --key-id hsm::4::my_aes_key --data-encryption-algorithm
aes-gcm /tmp/secret.txt
```

The encrypted file is available at `/tmp/secret.enc`

3.4.5 Decrypt

Symmetric keys and private keys can be used to decrypt data. Only the user identified by the `hsm-admin` argument or a user granted the Decrypt operation (by the HSM admin) can decrypt data with keys stored in the HSM.

For symmetric keys, only AES GCM is supported. CKM_RSA_PKCS_OAEP and the now deprecated but still widely used CKM_RSA_PKCS (v1.5) are supported for RSA keys. The hashing algorithm is fixed to SHA256.

To decrypt a message with the private key h`sm::4::hsm::4::my_rsa_key` and the CKM RSA PKCS OAEP algorithm, the following command can be used:

```
❯ cosmian kms rsa decrypt --key-id hsm::4::my_rsa_key --encryption-algorithm ckm-
rsa-pkcs-oaep \

  --output-file /tmp/secret.recovered.txt /tmp/secret.enc
```

The decrypted file is available at `/tmp/secret.plain`

To decrypt a message using AES GCM with the symmetric key `hsm::4::my_aes_key`, the following command can be used:

```
> cosmian kms sym decrypt --key-id hsm::4::my_aes_key --data-encryption-algorithm
aes-gcm

  --output-file /tmp/secret.recovered.txt /tmp/secret.enc
```

The decrypted file is available at "/tmp/secret.recoverd.txt"

3.5 Creating a KMS key wrapped by an HSM key

The `–wrapping-key-id` argument must specify the unique identifier of the HSM key to create a KMS key wrapped by an HSM key.

The user creating the key must be the HSM admin (see above) or have been granted the Encrypt operation on the HSM key.

For instance, the following command creates a 256-bit AES key wrapped by the HSM RSA (public) key `hsm::4::my_rsa_key_pk`:

```
> cosmian kms sym keys create --algorithm aes --number-of-bits 256 --sensitive \

  --wrapping-key-id hsm::4::my_rsa_key_pk my_sym_key

The symmetric key was successfully generated.

     Unique identifier: my_sym_key
```

The symmetric key is now stored in the database, encrypted (wrapped) by the HSM key. The encryption happened in the HSM.

The symmetric key can now be used to encrypt and decrypt data, and the KMS will transparently unwrap the key using the HSM key.

This unwrapping will happen once, and the unwrapped symmetric key will be cached in memory for later operations; no clear text symmetric key will be stored in the KMS database.

3.5.1 Small data: encrypting server side

For example, to encrypt a message with the key `my_sym_key` server side, the following command can be used:

```
> cosmian kms sym encrypt --key-id my_sym_key /tmp/secret.txt
```

```
The encrypted file is available at "/tmp/secret.enc"
```

To decrypt a message with the key my_sym_key, the following command can be used:

```
> cosmian kms sym decrypt --key-id my_sym_key --output-file /tmp/
secret.recovered.txt /tmp/secret.enc
```

The decrypted file is available at `/tmp/secret.recovered.txt`

3.5.2 Large data: encrypting client side with key wrapping

To encrypt a large file with the key `my_sym_key` client side, the following command can be used:

```
>cosmian kms sym encrypt --key-id my_sym_key_2 --data-encryption-algorithm aes-gcm
\

--key-encryption-algorithm rfc5649 /tmp/large.bin
```

The encrypted file is available at `/tmp/large.enc`

In this case an ephemeral symmetric key (the Data Encryption Key, DEK) is generated and used to encrypt the data. The DEK is then encrypted/wrapped with RFC4659 (a.k.a NIST AES Key Wrap) with the key `my_sym_key`, called the Key Encryption Key, KEK. The wrapping of the DEK by the KEK is stored at the beginning of the encrypted file. At rest, in the KMS database, `my_sym_key` is stored encrypted/wrapped with the HSM key `hsm::4::my_rsa_key_pk`.

To decrypt a large file with the KEK `my_sym_key` client side, the following command can be used:

```
> cosmian kms sym decrypt --key-id my_sym_key_2 --data-encryption-algorithm aes-
gcm \

  --key-encryption-algorithm rfc5649 --output-file /tmp/large.recoverd.bin /tmp/
large.enc
```

The decrypted file is available at `/tmp/large.recoverd.bin`

4 Further Information

This document forms a part of the information and support which is provided by the Utimaco IS GmbH. Additional documentation can be found on the product CD in the Documentation directory.

All CryptoServer product documentation is also available at the Utimaco IS GmbH website:

http://hsm.utimaco.com