nFinity Partner

KMIP Server Gateway (KSG)

nCipher Integration Guide

Version 1.0

DATE: JUNE 1, 2020

Introduction

The KSG is a special purpose KMIP Server application that sits in front of existing HSMs (Hardware Security Modules). It provides an OASIS KMIP protocol interface to existing HSMs by translating KMIP protocol requests into PKCS#11 calls and then translating the PKCS#11 responses back into KMIP. The possible KMIP operations that are supported via KSG depend upon the limitations of the backend HSMs. KSG supports most commercially available HSMs. KSG also supports a local PKCS#11 software token.

Integration configurations

The integration between the nShield HSM and the P6R KMIP Server has been tested for the following integration combinations:

Operating system	***Partner product name***	**nShield HSM**	**nShield firmware version**	**nShield Security World version**	**Certification support**
Linux/UNIX/Windows	KMIP Server	Connect XC, Edge, nSaaS	12.50.11	12.60.5	FIPS 140-2

Supported cryptographic algorithms

KSG version 2020.1.23851 supports the following cryptographic algorithms. The PKCS#11 mechanisms are mapped from the KMIP equivalent.

```
Encrypt / Decrypt: 3DES, AES, and RSA with OAEP.   PKCS#11 mechanisms:
CKM_DES_OFB64, CKM_DES_CFB64, CKM_AES_CBC_PAD, CKM_AES_CBC, CKM_AES_ECB.
CKM_AES_OFB, CKM_AES_CFB64, CKM_RSA_PKCS_OAEP.

Sign / Verify: RSA, EC, and HMAC. PKCS#11 mechanisms:
CKM_SHA1_RSA_PKCS, CKM_SHA224_RSA_PKCS, CKM_SHA256_RSA_PKCS,
CKM_SHA384_RSA_PKCS, CKM_SHA512_RSA_PKCS,
CKM_ECDSA_SHA1, CKM_SHA_1_HMAC, CKM_SHA224_HMAC, CKM_SHA256_HMAC,
CKM_SHA384_HMAC, CKM_SHA512_HMAC,
CKM_ECDSA_SHA1, CKM_SHA1_RSA_PKCS, CKM_SHA224_RSA_PKC, CKM_SHA256_RSA_PKCS,
CKM_SHA384_RSA_PKCS,
and CKM_SHA512_RSA_PKC.

Key Generation: CKM_NC_SHA_1_HMAC_KEY_GEN, CKM_NC_SHA224_HMAC_KEY_GEN,
CKM_NC_SHA256_HMAC_KEY_GEN, CKM_NC_SHA384_HMAC_KEY_GEN,
CKM_NC_SHA512_HMAC_KEY_GEN,
 CKM_DES3_KEY_GEN, CKM_AES_KEY_GEN, CKM_RSA_PKCS_KEY_PAIR_GEN,
CKM_EC_KEY_PAIR_GEN, CKM_SHA1_KEY_DERIVATION,
 CKM_SHA224_KEY_DERIVATION, CKM_SHA256_KEY_DERIVATION,
CKM_SHA384_KEY_DERIVATION, CKM_SHA512_KEY_DERIVATION,
 and CKM_GENERIC_SECRET_KEY_GEN.

EC Curve names:
   c2pnb163v1 1.2.840.10045.3.0.1
   c2pnb163v2 1.2.840.10045.3.0.2
   c2pnb163v3 1.2.840.10045.3.0.3
    c2tnb191v1 1.2.840.10045.3.0.5
    c2tnb191v2 1.2.840.10045.3.0.6
    c2tnb191v3 1.2.840.10045.3.0.7
    c2pnb208w1 1.2.840.10045.3.0.10
    c2tnb239v1 1.2.840.10045.3.0.11
    c2tnb239v2 1.2.840.10045.3.0.12
    c2tnb239v3 1.2.840.10045.3.0.13
    c2pnb272w1 1.2.840.10045.3.0.16
    c2pnb304w1 1.2.840.10045.3.0.17
    c2tnb359v1 1.2.840.10045.3.0.18
    c2pnb368w1 1.2.840.10045.3.0.19
    c2tnb431r1 1.2.840.10045.3.0.20
    prime192v1 1.2.840.10045.3.1.1
    prime192v2 1.2.840.10045.3.1.2
    prime192v3 1.2.840.10045.3.1.3
    prime239v1 1.2.840.10045.3.1.4
    prime239v2 1.2.840.10045.3.1.5
    prime239v3 1.2.840.10045.3.1.6
    prime256v1 1.2.840.10045.3.1.7
```

Supported nCipher features

KSG supports KMIP versions 1.0, 1.1, 1.2, 1.3, 1.4, and 2.0. KSG supports all KMIP message formats: TTLV, TTLV over HTTPS, JSON, and XML. KSG is however limited in what KMIP operations it can support due to limitations enforced by the backend HSM in use (e.g., if the HSM is in FIPS mode many restrictions exist on its use).

Currently, the following KMIP operations are supported:

- Activate
- Add Attribute
- Delete Attribute
- Destroy
- Discover Versions
- Create (both Symmetric Key and Secret Data Object)
- Create Key Pair
- Get (Symmetric Key, Public Key, Certificate, Secret Data)
- Get Attributes
- Get Attribute List
- Get Usage Allocation
- Locate
- Modify Attribute
- Obtain Lease
- Query
- Register
- Re-Key
- RNG Retrieve
- RNG Seed
- Revoke
- Decrypt
- Encrypt
- Sign
- Signature Verify
- MAC
- MAC Verify
- Derive Key (only HASH derivation method for SHA-2 keys are currently supported)
- Interop (for KMIP 2.0 and greater when enabled)

The following limitations exist:

1. Only a single instance of any KMIP attribute is currently supported. Support for mulit-instance attributes is on the road map. Note that many applications only use a single instance of an attribute.

2. KMIP Get operation for a Symmetric Key works on AWS CloudHSM and Cavium HSMs because both support a customer supplied Key Encrypting Key (KEK), which allows KSG to extract the Symmetric Key in wrapped form from the HSM and then locally decrypt it.
3. KMIP RNG Seed operation works if the HSM supports it.
4. Currently, a Secret Data Object created on a call to the KMIP Create operation is saved in the KSG database. In the future, a configuration item will allow this object to be saved on an HSM as a CKO_DATA object if the HSM supports CKO_DATA objects.
5. The KMIP Digest attribute can only be calculated if the backend HSM supports the PKCS#11 C_DigestKey() API call.
6. KMIP Encrypt, Decrypt, Sign, Signature Verify, MAC and MAC Verify operations are the non-streaming versions (as introduced in KMIP 1.2).
7. KMIP Register only allows a Certificate object to be loaded into the backend HSM.
   Using the KMIP "Batch Error Continuation Option" set to "Undo" will not work properly with a KMIP "Destroy" operation. KSG's current undo implementation uses a database transaction of the KSG internal database to roll back the effect of operations in a failed KMIP request. However, when KSG is connected to an HSM this transaction does not include the HSM's state. So, if a KMIP Destroy operation is performed, thus removing key material from the HSM, the current undo implementation will not restore the deleted key material. Thus, KMIP Destroy operations are not supported in failed KMIP requests with an "Undo" error continuation.

KSG can run with the P6R PKCS#11 software token that does not run in FIPS mode. In this configuration, many of the limitations above do not exist, however, no HSM is in use.

Requirements and prerequisites

The P6R customer will require a P6R KSG production license file.

Terminology

Terms and abbreviations related to the nFinity Partner Product or to nCipher product(s).

Term	Description
ACS	Administrator Card Set - reference nCipher product documentation
OCS	Operator Card Set - reference nCipher product documentation
Softcard	Passcode - reference nCipher product documentation
Module	Module - - reference nCipher product documentation
KMIP	Key Management Interoperability Protocol

Related documents and support

Documents for nFinity Partner Product; nCipher hardware, software.

- nShield Installation Guide
- User Guide for the nShield HSM
- User Guide for the nFinity Partner Product

nFinity Partner support pages: https://ncipher.zendesk.com/hc/en-us.

Integration procedure

For this version of KSG the user must configure KSG's PKCS#11 module to talk to the user's HSM. In addition, the user must set up a TLS proxy to take incoming TLS requests and forward them to KSG (e.g., Stunnel) [in a future release KSG will have its own TLS server protocol handling as an option]. Here are the KSG set up steps.

1. Install KSG in the directory you will run it out of.
2. Run the p6pkcs11tool to configure the P6R PKCS#11 module so it can talk to your HSM. (See documentation for the p6pkcs11tool that comes with KSG.) When PKCS#11 is properly configured the following files should appear in the ..<install directory>/data directory: PKCS11, PKCS11.sig, and pkcs11baseKey.txt. Now KSG is ready to talk to your HSM.
3. Modify the ..<install directory>/confs/ksg.conf file (see options below under the section "Gateway Configuration Parameters").
4. Copy the P6R license file into the ..<install directory>/licenses/ directory.
5. Install and configure Stunnel (or another TLS proxy) to sit in front of KSG (see the section "KSG and Stunnel" below).

Note: To run the p6pkcs11tool on Linux currently requires setting LD_LIBRARY_PATH the to "<installed path>../components" directory so it can dynamically load libp6pkcs11.so, which is P6R's PKCS#11 library implementation. Otherwise, an error that the shared library cannot be found will be returned.

After installing KSG make sure to copy the cknfast-64.dll (for Windows) or libcknfast.so (for Linux) file into the KSG "components" directory. This allows KSG to find and load the nCipher PKCS#11 library and use it to talk to the HSM.

To modify the behavior of the nShield Connect\Edge a user can set either environment variables or create a "cknfastrc" file for the HSM to read. The cknfastrc file that we used for this integration is as follows:

CKNFAST_OVERRIDE_SECURITY_ASSURANCES=explicitness;tokenkeys;longterm
CKNFAST_FAKE_ACCELERATOR_LOGIN=1

Note that in addition to the parameters set above, we routinely set CKNFAST_DEBUG=9 and CKNFAST_DEBUGFILE to log the actions taken by the HSM and there results. Please refer to nCipher documentation for a description of the settings above.

The nCipher HSM uses Vendor Defined mechanisms for HMAC key generation. Instead of the PKCS#11 V2.40 standard where HMACs are created via key generation nCipher uses the GenerateKey() API. P6R's KSG handles this difference but requires the calling KMIP application to use the KMIP Create operation instead of the KMIP Derive Key operation to create a HMAC key when using an nCipher nShield HSM. Note, that this also requires that the "HSM_type" KSG configuration parameter must be set with the nCipher nShield defined value (see above).

Unlike the AWS CloudHSM and Cavium HSMs, the integration with the nCipher nShield Connect\Edge does not support the KMIP Get operation since we did not find any way to inject a customer provided KEK into the HSM. Thus, keys created on the nShield Connect\Edge could only be extracted from the HSM using wrapping keys created on the HSM.

We performed integration testing with the nShield Edge (which provides the same API as the Connect). Specifics of the HSM we used where discovered by calling the PKCS#11 API function C_GetTokenInfo():

...bin>p6pkcs11tool -lt 0
p6pkcs11Tool v2019.1.23062 - Copyright 2004-2019 P6R Inc - All Rights Reserved

------- Token Details -------
Slot Id: 0
Label: [accelerator]
Manufacturer Id: [nCipher Corp. Ltd]
Serial Number: [2D77-39B3-D9FE]
Flags: 209
Flag set: CKF_RNG
Flag set: CKF_USER_PIN_INITIALIZED
Flag set: CKF_DUAL_CRYPTO_OPERATIONS
Max PIN length: 256
Min PIN length: 0
Hardware Version: 0.11
Firmware Version: 2.61

The P6R PKCS#11 library uses two different tokens to load nCipher HSMs. The nCipher "{7DFB0F94-811D-409D-9062-73E4AD9FE1B6}" token is used for HSMs that support Security World Software Versions 12.60. Note that both tokens have been tested to also work with the nShield Edge product.

A nCipher nShield Connect HSM can be configured to look like another token. Actually, we map one slot of the nCipher HSM into one slot of our P6R PKCS#11 library. In the example below, P6R slot number 6 is mapped to nCipher HSM slot number 492971158 (0x1d622496) (see the vendorSlotId parameter).

This is easily done by the following two steps. First, define the nCipher slot in the p6pkcs11.conf file. An example of how to do this is above on see the @ref p6pkcs11_configuration Page and is repeated below:

[p6pkcs11-tokens]
nCipher = "{7DFB0F94-811D-409D-9062-73E4AD9FE1B6}"

[p6pkcs11-slot-6]
slotType="nCipherVersion201"
slotFlags=5
vendorSlotId=492971158
vendorVerifyPIN=true
vendorPreInitToken=true
vendorCompat=3

Any slot number will work so the section header of "[p6pkcs11-slot-6]" above is arbitrary. The value under the "[p6pkcs11-tokens]" section above defines the GUID of the P6R plugin DLL (.so file for Linux) for nCipher HSMs.

The second step is to copy the nCipher provided files: cknfast-64.dll (64 bit) or cknfast-32.dll (32 bit) (on Windows), and libcknfast.so (on Linux) into the P6R "components" directory so that they can be found and loaded. Or when using Security World Software Versions 12.60 copy the nCipher provided files:

cknfast.dll (64 bit) or cknfast32.dll (32 bit) (on Windows). Then set up and configure the nCipher nShield Connect HSM as defined in the nCipher Quick Start Guide.

However, the nShield does not seem to start it slot numbers at zero so to discover the HSMs starting slot numbers run the following command:

C:\ksg-2019.1.23062\windows\WINNT6.1_x86_OPT_64bit_vs2010.OBJ\bin>p6pkcs11tool -lp 0
p6pkcs11Tool v2019.1.23062 - Copyright 2004-2019 P6R Inc - All Rights Reserved

----- Plugin Slot Id List -----
Slot Id: 0
plugin Slot Id: 492971157
plugin Slot Id: 492971158

Note, when the nShield is in strict FIPS mode a call to C_GenerateKey() needs the attributes CKA_SENSITIVE and CKA_PRIVATE has to be set to CKA_TRUE. In addition, calls to C_CreateObject() where the calling application is trying to place a key onto the HSM will fail since strict FIPS mode does not allow keys of unknown origin from being loaded into the HSM.

Contact Information

To request technical support for P6R KMIP Server reference:

https://support.p6r.com/

To request technical support for nCipher nShield Products reference:

https://www.ncipher.com/services/support/contact-support for technical support contact details.