OpenSSL 3

nShield® HSM Integration Guide

2026-01-28

© Entrust Corporation. All rights reserved.

Table of Contents

1. Introduction . . . 1
    1.1. Product configurations . . . 1
    1.2. Test cases in this guide . . . 1
    1.3. Requirements . . . 3
    1.4. More information . . . 4
2. Procedures . . . 5
    2.1. Install the HSM . . . 5
    2.2. Install the Security World Software and create the security world . . . 5
    2.3. Install OpenSSL and required packages . . . 5
    2.4. Set up the PKCS #11 engine . . . 5
    2.5. Test cases . . . 8
    2.6. FIPS Level 3 remarks and recommendations . . . 18
3. Additional resources and related products . . . 19
    3.1. nShield HSMs . . . 19
    3.2. nShield as a Service . . . 19
    3.3. Entrust products . . . 19
    3.4. nShield product documentation . . . 19

Chapter 1. Introduction

This guide describes how to use the nShield PKCS #11 library to integrate an Entrust nShield® Hardware Security Module (HSM) with OpenSSL 3 to provide secure solutions.

Some of the scenarios where OpenSSL can be used with an HSM:

- Generating key pairs protected by the HSM
- Generating certificates protected by keys protected by the HSM
- Importing of certificates so they can be protected by the HSM
- Encrypting and decrypting files and documents using keys protected by the HSM
- Signing applications and files with keys protected by the HSM

1.1. Product configurations

We have successfully tested nShield HSM integration with OpenSSL 3 on a server running Red Hat Enterprise Linux 9 (x86-64-v2) in the following configurations:

OpenSSL	Security World	HSM	Firmware	Netimage	FIPS Level 3
3.0.7	13.3.2	Connect XC	12.50.11 (FIPS 140-2 certified)	12.80.4	No
3.0.7	13.3.2	Connect 5C	13.2.2	13.4.5	Yes
3.0.7	13.4.5	Connect XC	12.50.11 (FIPS 140-2 certified)	12.80.4	No
3.0.7	13.4.5	Connect 5C	13.2.2	13.4.5	Yes
3.0.7	13.4.5	Connect XC	12.72.1 (FIPS 140-2 certified)	13.4.5	Yes

1.2. Test cases in this guide

The following table summarizes the test cases in this guide and the test result in regards to the type of protection used: module, softcard, and OCS.

Test case	All protection methods	Notes
Generate RSA key pair - protected by HSM - `generatekey`	Passed	
Generate RSA key pair - protected by HSM - `pkcs11-tool`	Passed	
Generate DSA key pair - protected by HSM - `generatekey`	Passed	
Generate ECC key pair - protected by HSM - `generatekey`	Passed	
Generate AES key pair - protected by HSM - `generatekey`	Passed	
Generate AES key pair - protected by HSM - `pkcs11-tool`	Passed	Warnings when using the `pkcs11-tool` were observed but the keys were created. Warning: PKCS11 function C_GetAttributeValue(VALUE) failed: rv = CKR_ATTRIBUTE_SENSITIVE (0x11)
Generate CSR from RSA Key	Passed	
Generate a Self Signed Certificate	Passed	
Import the Signed Certificate	Passed	The signed certificate is imported but is always protected by the module, even if you use a softcard or OCS card.
Encrypt text file with RSA key	Passed	

Test case	All protection methods	Notes
Decrypt text file with RSA key	Passed	Need to use -pkeyopt rsa_padding_mode:oaep on FIPS Level 3
Sign text file with RSA key	Passed	
Encrypt/Decrypt text file with AES key	Not tested	Not Supported currently by OpenSSL 3
Sign text file with ECC key	Tested with module and softcard protection	Passed

1.3. Requirements

Ensure that you have supported versions of the nShield, OpenSSL, and third-party products. See Product configurations.

To perform the integration tasks, you must have:

- `root` access on the operating system.
- Access to `nfast` accounts.

Before starting the integration process, familiarize yourself with:

- The documentation for the HSM.
- The documentation and setup process for OpenSSL on server.

Before using the nShield software, you need to know:

- The number and quorum of Administrator Cards in the Administrator Card Set (ACS), and the policy for managing these cards.
- Whether the application keys are protected by the module or an Operator Card Set (OCS) with or without a pass phrase.
- The number and quorum of Operator Cards in the OCS, and the policy for managing these cards.
- Whether the security world should be compliant with FIPS 140 Level 3.

Entrust recommends that you allow only unprivileged connections unless you are performing administrative tasks.

For more information, see the *User Guide* and *Installation Guide* for the HSM.

1.4. More information

OpenSSL is a registered trademark owned by the OpenSSL Software Foundation, see https://openssl.org/.

For more information about OS support, contact your Entrust nShield Support, https://nshieldsupport.entrust.com.

Access to the Entrust nShield Support Portal is available to customers under maintenance. To request an account, contact nshield.support@entrust.com.

Chapter 2. Procedures

This guide describes several scenarios using module, softcard and OCS protection.

2.1. Install the HSM

Install the HSM by following the instructions in the *Installation Guide* for the HSM.

We recommend that you install the HSM before configuring the Security World Software in the OpenSSL server.

2.2. Install the Security World Software and create the security world

To install the Security World Software and create the security world:

1. On the computer that you want to use OpenSSL, install the latest version of the Security World Software as described in the *Installation Guide* for the HSM.

   Entrust recommends that you uninstall all existing nShield software before you install new nShield software.

2. Create the security world as described in the *User Guide*, creating the ACS and OCS that you require.

2.3. Install OpenSSL and required packages

The installation of OpenSSL varies depending on the operating system.

Installation on RedHat Linux 9:

```
sudo yum install -y openssl-pkcs11 openssl-libs mod_ssl opensc expect
```

2.4. Set up the PKCS #11 engine

To avoid problems associated with the Entrust supplied OpenSSL, which is used internally by `generatekey` to make certificates, ensure that `/opt/nfast/bin` is NOT at the front of your `$PATH`.

You can confirm that the right binary is being run with the following command:

```
% which openssl

/usr/bin/openssl
```

If this command returns anything inside `/opt/nfast`, check your `$PATH` variable.

2.4.1. Configure OpenSSL

1. Locate the OpenSSL configuration file:

   ```
   % openssl version -d

   OPENSSLDIR: "/etc/pki/tls"
   ```

   The minimum configuration is similar to the following:

   ```
   #
   # OpenSSL example configuration file.
   # This is mostly being used for generation of certificate requests.
   #

   # Note that you can include other files from the main configuration
   # file using the .include directive.
   #.include filename
   .include = /etc/pki/tls/openssl.original.cnf

   # This definition stops the following lines choking if HOME isn't
   # defined.
   HOME = .
   RANDFILE = $ENV::HOME/.rnd

   #OJW: nShield PKCS11
   openssl_conf = openssl_def
   [openssl_def]
   engines = engine_section
   [engine_section]
   pkcs11 = pkcs11_section
   [pkcs11_section]
   engine_id = pkcs11
   #dynamic_path = /usr/lib/ssl/engines/libpkcs11.so #Ubuntu16.04
   #dynamic_path = /usr/lib/x86_64-linux-gnu/engines-1.1/pkcs11.so #Debian9&10, Ubuntu18.04
   #dynamic_path = /usr/lib64/openssl/engines/pkcs11.so #RHEL/CentOS7
   #dynamic_path = /usr/lib64/engines-1.1/pkcs11.so #RHEL/CentOS8
   dynamic_path = /usr/lib64/engines-3/pkcs11.so #RHEL 9
   MODULE_PATH = /opt/nfast/toolkits/pkcs11/libcknfast.so
   init = 0
   #!
   ```

   The following message can appear when you are creating certificates:

   ```
   unable to find 'distinguished_name' in config
   problems making Certificate Request
   140493626791824:error:0E06D06C:configuration file routines:NCONF_get_string:no
   value:conf_lib.c:324:group=req name=distinguished_name
   ```

If it does, you need to add the following to your OpenSSL configuration, adjusted to your organization's values:

```
#
[req]
distinguished_name = req_distinguished_name
req_extensions = v3_req
prompt = no
[req_distinguished_name]
C = US
ST = FL
L = Sunrise
O = Entrust
OU = nShield
CN = www.entrust.com
[v3_req]
subjectAltName = @alt_names
[alt_names]
DNS.1 = www.entrust.com
DNS.2 = entrust.com
```

Make sure the server's host name matches the CN in the certificate.

2. Create or edit the `/etc/pki/tls/openssl.pkcs11.cnf` file:

```
% sudo vi /etc/pki/tls/openssl.pkcs11.cnf
```

3. Enter the settings from above, and save the file.

2.4.2. Set up /opt/nfast/cknfastrc

You might have to add the following variables to the `/opt/nfast/cknfastrc` file. They are referenced in this guide to address specific situations and their use depends on your current environment.

```
CKNFAST_DEBUG=10
CKNFAST_DEBUGFILE=/path/to/debug/file
CKNFAST_FAKE_ACCELERATOR_LOGIN=1
CKNFAST_LOADSHARING=1
```

Turn debug off in a production environment.

2.4.3. Test the configuration

1. Export the `OPENSSL_CONF` environment variable:

```
% export OPENSSL_CONF=/etc/pki/tls/openssl.pkcs11.cnf
```

2. Test the configuration:

```
% openssl engine -tt -c -v

(pkcs11) pkcs11 engine
[RSA, rsaEncryption, id-ecPublicKey]
[ available ]
SO_PATH, MODULE_PATH, PIN, VERBOSE, QUIET, INIT_ARGS, FORCE_LOGIN
```

Debugging notes

Security world permissions

The following message can appear:

```
Unable to load module /opt/nfast/toolkits/pkcs11/libcknfast.so
```

If it does, it indicates that there is no security world. Make sure you create a security world first.

Missing PKCS #11 engine in the output

If you don't see the PKCS #11 engine in the output, check the `dynamic_path` line in the `openssl.pkcs11.cnf` configuration file. This may vary on other platforms and other operating system versions.

```
dynamic_path = /usr/lib64/engines-3/pkcs11.so #RHEL 9
```

2.5. Test cases

The test cases cover module, softcard and OCS protection. Create an OCS card and a softcard so they can be used with the test cases.

1. Create an OCS card called `testOCS`:

```
% createocs -m1 -s2 --persist -Q 1/1 -N testOCS
```

2. Create a softcard called `testSC`:

```
% ppmk -n testSC
```

Test case execution

1. Generate an RSA key pair where private key is protected with HSM using `pkcs11-tool`:

Module protection

```
% pkcs11-tool --module /opt/nfast/toolkits/pkcs11/libcknfast.so --slot-index 0 -l \
  --pin ncipher --keypairgen --key-type rsa:2048 --label rsa_key_1
```

Softcard protection

```
% pkcs11-tool --module /opt/nfast/toolkits/pkcs11/libcknfast.so --token-label testSC \
  -l --pin ncipher --keypairgen --key-type rsa:2048 --label rsa_key_1
```

OCS protection

```
% pkcs11-tool --module /opt/nfast/toolkits/pkcs11/libcknfast.so --token-label testOCS \
  -l --pin ncipher --keypairgen --key-type rsa:2048 --label rsa_key_2
```

2. Generate an RSA key pair where private key is protected with HSM using `generatekey`:

Module protection

```
% generatekey -b -g -m1 pkcs11 plainname=rsa_key_2 type=rsa \
  protect=module size=2048
```

Softcard protection

```
% generatekey -b -g -m1 pkcs11 plainname=rsa_key_2 type=rsa \
  protect=softcard size=2048 softcard=testSC
```

OCS protection

```
% generatekey -b -g -m1 pkcs11 plainname=rsa_key_2 type=rsa \
  protect=token size=2048 cardset=testOCS
```

3. List the keys that have been generated:

```
% nfkminfo -l
```

4. Generate a DSA key pair where private key is protected with HSM using `generatekey`:

Module protection

```
% generatekey -b -g -m1 pkcs11 plainname=dsa_key_1 type=dsa \
  protect=module size=2048
```

Softcard protection

```
% generatekey -b -g -m1 pkcs11 plainname=dsa_key_1 type=dsa \
  protect=softcard size=2048 softcard=testSC
```

OCS protection

```
% generatekey -b -g -m1 pkcs11 plainname=dsa_key_1 type=dsa \
  protect=token size=2048 cardset=testOCS
```

5. List the keys that have been generated:

```
% nfkminfo -l
```

6. Generate an ECC key pair where private key is protected with HSM using `generatekey`:

Module protection

```
% generatekey -b -g -m1 pkcs11 plainname=ecc_key_1 \
  type=ECDSA curve=NISTP224 protect=module
```

Softcard protection

```
% generatekey -b -g -m1 pkcs11 plainname=ecc_key_1 \
  type=ECDSA curve=NISTP224 protect=softcard softcard=testSC
```

OCS protection

```
% generatekey -b -g -m1 pkcs11 plainname=ecc_key_1 \
  type=ECDSA curve=NISTP224 protect=token cardset=testOCS
```

7. List the keys that have been generated:

```
% nfkminfo -l
```

8. Generate a symmetric AES key protected with HSM using `pkcs11-tool`:

   The following warning appears during execution, for module, softcard and OCS protection but it does not affect key creation.

```
warning: PKCS11 function C_GetAttributeValue(VALUE) failed: rv = CKR_ATTRIBUTE_SENSITIVE (0x11)
```

   If you don't supply the `--private` option, for softcard and OCS protection, the following error appears:

```
error: PKCS11 function C_GenerateKey failed: rv = CKR_TEMPLATE_INCONSISTENT (0xd1)
Aborting.
```

Module protection

```
% pkcs11-tool --module /opt/nfast/toolkits/pkcs11/libcknfast.so --slot-index 0 \
  -l --pin ncipher --keygen --key-type AES:32 --label aes_key_1
```

Softcard protection

```
% pkcs11-tool --module /opt/nfast/toolkits/pkcs11/libcknfast.so --token-label testSC \
  -l --pin ncipher --private --keygen --key-type AES:32 --label aes_key_1
```

OCS protection

```
% pkcs11-tool --module /opt/nfast/toolkits/pkcs11/libcknfast.so --token-label testOCS \
  -l --pin ncipher --private --keygen --key-type AES:32 --label aes_key_1
```

9. Generate a symmetric AES key protected with HSM using `generatekey`:

Module protection

```
% generatekey -b -g -m1 pkcs11 plainname=aes_key_2 type=aes \
  protect=module size=256
```

Softcard protection

```
% generatekey -b -g -m1 pkcs11 plainname=aes_key_2 type=aes \
  protect=softcard size=256 softcard=testSC
```

OCS protection

```
% generatekey -b -g -m1 pkcs11 plainname=aes_key_2 type=aes \
  size=256 protect=token cardset=testOCS
```

10. List the keys that have been generated:

```
% nfkminfo -l
```

11. Generate CSR from RSA Key:

Module protection:

```
% openssl req -engine pkcs11 -keyform engine \
  -key pkcs11:token=loadshared%20accelerator;pin-value=ncipher;type=private;object=rsa_key_2 \
```

```
  -new -out /tmp/csrfile.csr
```

Softcard protection

```
% openssl req -engine pkcs11 -keyform engine \
  -key pkcs11:token=testSC;pin-value=ncipher;type=private;object=rsa_key_2 \
  -new -out /tmp/csrfile.csr
```

OCS protection

```
% openssl req -engine pkcs11 -keyform engine \
  -key pkcs11:token=testOCS;pin-value=ncipher;type=private;object=rsa_key_2 \
  -new -out /tmp/csrfile.csr
```

12. Check the CSR file:

```
% ls -al /tmp/csrfile.csr
-rw-r--r--. 1 cccc cccc 1086 Apr 17 11:17 /tmp/csrfile.csr

% file /tmp/csrfile.csr
/tmp/csrfile.csr: PEM certificate request
```

13. Generate a self-signed certificate:

Module protection

```
% openssl req -x509 -new -engine pkcs11 -keyform engine \
  -key pkcs11:token=loadshared%20accelerator;pin-value=ncipher;object=rsa_key_2 \
  -out /tmp/certificate.pem -days 365
```

Softcard protection

```
% openssl req -x509 -new -engine pkcs11 -keyform engine \
  -key pkcs11:token=testSC;pin-value=ncipher;type=private;object=rsa_key_2 \
  -out /tmp/certificate.pem -days 365
```

OCS protection

```
% openssl req -x509 -new -engine pkcs11 -keyform engine \
  -key pkcs11:token=testOCS;pin-value=ncipher;type=private;object=rsa_key_2 \
  -out /tmp/certificate.pem -days 365
```

14. Check the self-signed certificate:

```
% ls -al /tmp/certificate.pem
-rw-r--r--. 1 cccc cccc 1224 Apr 17 11:26 /tmp/certificate.pem

% file /tmp/certificate.pem
```

```
/tmp/csrfile.csr: PEM certificate request
```

15. Before you import the self-signed certificate, list the available keys:

```
% nfkminfo -l
```

16. Import the certificate:

    When you import the certificate, the key and the certificate will have the same label. The signed certificate is also imported but it will be protected by the module.

    Module protection

```
% ckcerttool -n -f /tmp/module.pem -L rsa_key_2 \
  -k uaa48299ba93d20591d8b914f187de24afec50775b
```

    Softcard protection

```
% ckcerttool -c testSC -f /tmp/softcard.pem -L rsa_key_2 \
  -k ucb0d00b55bc6a5fe7aa1caf88bc9c8e70a26f051a-34668473dab910e0497ae44bb9e0b1fb5c8d2d3e
```

    OCS protection

```
% ckcerttool -c testOCS -f /tmp/ocs.pem -L rsa_key_2 \
  -k uc063c0d5280a8c3046acc6b8dce312387ce0e3855-166c4e5fedb0995db1bbe0c1ffecf4b537bab85e
```

17. List the available keys again:

```
% nfkminfo -l
```

18. Look for the imported certificate by running `cklist` that lists certificates and the keys:

```
% cklist -pncipher -n -l rsa_key_2
```

    Look for: `CKA_CLASS CKO_CERTIFICATE` and `CKA_LABEL "rsa_key_2"`.

19. Encrypt and decrypt a text file with the RSA key.

    During FIPS Level 3 testing the following error was found at the decryption step:

```
Public Key operation error
409C08B1AC7F0000:error:41800070:PKCS#11 module:ERR_CKR_error:Mechanism invalid:p11_rsa.c:150:
```

    This happens because `CKM_RSA_PKCS` is not allowed in FIPS mode. You have to use

OAEP, which is more secure. Add the `oaep` flag to `openssl pkeyutl`:

```
-pkeyopt rsa_padding_mode:oaep
```

Create a file to be encrypted with the RSA key:

```
This file was created to be encrypted.
Just a regular text file.
```

20. Encrypt the file with the RSA key:

Module protection

```
% openssl pkeyutl -encrypt -engine pkcs11 -keyform engine -in /tmp/filetoencrypt.txt \
  -inkey pkcs11:token=loadshared%20accelerator;pin-value=ncipher;object=rsa_key_2 \
  -out /tmp/encryptedfile.txt -pkeyopt rsa_padding_mode:oaep
```

Softcard protection

```
% openssl pkeyutl -encrypt -engine pkcs11 -keyform engine -in /tmp/filetoencrypt.txt \
  -inkey pkcs11:token=testSC;pin-value=ncipher;object=rsa_key_2 \
  -out /tmp/encryptedfile.txt -pkeyopt rsa_padding_mode:oaep
```

OCS protection

```
% openssl pkeyutl -encrypt -engine pkcs11 -keyform engine -in /tmp/filetoencrypt.txt \
  -inkey pkcs11:token=testOCS;pin-value=ncipher;object=rsa_key_2 \
  -out /tmp/encryptedfile.txt -pkeyopt rsa_padding_mode:oaep
```

21. Verify the encryption by decrypting the file:

Module protection

```
% openssl pkeyutl -decrypt -engine pkcs11 -keyform engine -in /tmp/encryptedfile.txt \
  -inkey pkcs11:token=loadshared%20accelerator;pin-value=ncipher;object=rsa_key_2 \
  -out /tmp/decryptedfile.txt -pkeyopt rsa_padding_mode:oaep
```

Softcard protection

```
% openssl pkeyutl -decrypt -engine pkcs11 -keyform engine -in /tmp/encryptedfile.txt \
  -inkey pkcs11:token=testSC;pin-value=ncipher;object=rsa_key_2 \
  -out /tmp/decryptedfile.txt -pkeyopt rsa_padding_mode:oaep
```

OCS protection

```
% openssl pkeyutl -decrypt -engine pkcs11 -keyform engine -in /tmp/encryptedfile.txt \
```

```
  -inkey pkcs11:token=testOCS;pin-value=ncipher;object=rsa_key_2 \
  -out /tmp/decryptedfile.txt -pkeyopt rsa_padding_mode:oaep
```

22. List the contents of the decrypted file:

```
% cat /tmp/decryptedfile.txt

This file was created to be encrypted.
Just a regular text file.
```

23. Sign a file with existing RSA key.

    Create a file to encrypt:

```
This file was created to be encrypted.
Just a regular text file.
```

24. Perform the signing:

    Module protection

```
% openssl dgst -sha256 -engine pkcs11 -keyform engine \
  -sign pkcs11:token=loadshared%20accelerator;pin-value=ncipher;object=rsa_key_2 \
  -out /tmp/filetoencrypt.sig /tmp/filetoencrypt.txt
```

    Softcard protection

```
% openssl dgst -sha256 -engine pkcs11 -keyform engine \
  -sign pkcs11:token=testSC;pin-value=ncipher;object=rsa_key_2 \
  -out /tmp/filetoencrypt.sig /tmp/filetoencrypt.txt
```

    OCS protection

```
% openssl dgst -sha256 -engine pkcs11 -keyform engine \
  -sign pkcs11:token=testOCS;pin-value=ncipher;object=rsa_key_2 \
  -out /tmp/filetoencrypt.sig /tmp/filetoencrypt.txt
```

25. Extract the public key:

    Module protection

```
% openssl rsa -engine pkcs11 -inform ENGINE -pubout -text \
  -in pkcs11:token=loadshared%20accelerator;pin-value=ncipher;object=rsa_key_2 \
  -out /tmp/pub.signature.key
```

    Softcard protection

```
% openssl rsa -engine pkcs11 -inform ENGINE -pubout -text \
  -in pkcs11:token=testSC;pin-value=ncipher;object=rsa_key_2 \
  -out /tmp/pub.signature.key
```

OCS protection

```
% openssl rsa -engine pkcs11 -inform ENGINE -pubout -text \
  -in pkcs11:token=testOCS;pin-value=ncipher;object=rsa_key_2 \
  -out /tmp/pub.signature.key
```

26. Verify the signature:

```
% openssl dgst -verify /tmp/pub.signature.key -signature /tmp/filetoencrypt.sig /tmp/filetoencrypt.txt

Verified OK
```

27. Encrypt and decrypt a text file with the AES key.

    Create a file to encrypt:

```
This file was created to be encrypted.
Just a regular text file.
```

28. Perform the encryption:

Module protection

```
% openssl pkeyutl -encrypt -engine pkcs11 -keyform engine -in /tmp/filetoencrypt.txt \
  -inkey pkcs11:token=loadshared%20accelerator;pin-value=ncipher;object=aes_key_2 \
  -out /tmp/encryptedfile.txt

Engine "pkcs11" set.
No private keys found.
Key not found.
PKCS11_get_private_key returned NULL
Could not read private key from org.openssl.engine:pkcs11:pkcs11:token=loadshared%20accelerator;pin-
value=ncipher;object=aes_key_2
408CDDF3FB7F0000:error:40000065:pkcs11 engine:ERR_ENG_error:object not found:eng_back.c:974:
408CDDF3FB7F0000:error:13000080:engine routines:ENGINE_load_private_key:failed loading private
key:crypto/engine/eng_pkey.c:79:
pkeyutl: Error initializing context
```

Softcard protection

```
% openssl pkeyutl -encrypt -engine pkcs11 -keyform engine -in /tmp/filetoencrypt.txt \
  -inkey pkcs11:token=testSC;pin-value=ncipher;object=aes_key_2 \
  -out /tmp/encryptedfile.txt

Engine "pkcs11" set.
No private keys found.
Key not found.
PKCS11_get_private_key returned NULL
```

```
Could not read private key from org.openssl.engine:pkcs11:pkcs11:token=testSC;pin-
value=ncipher;object=aes_key_2
40AC8394D97F0000:error:40000065:pkcs11 engine:ERR_ENG_error:object not found:eng_back.c:974:
40AC8394D97F0000:error:13000080:engine routines:ENGINE_load_private_key:failed loading private
key:crypto/engine/eng_pkey.c:79:
pkeyutl: Error initializing context
```

OCS protection

```
% openssl pkeyutl -encrypt -engine pkcs11 -keyform engine -in /tmp/filetoencrypt.txt \
  -inkey pkcs11:token=testOCS;pin-value=ncipher;object=aes_key_2 -out /tmp/encryptedfile.txt

Engine "pkcs11" set.
No private keys found.
Key not found.
PKCS11_get_private_key returned NULL
Could not read private key from org.openssl.engine:pkcs11:pkcs11:token=testOCS;pin-
value=ncipher;object=aes_key_2
403CF2A61C7F0000:error:40000065:pkcs11 engine:ERR_ENG_error:object not found:eng_back.c:974:
403CF2A61C7F0000:error:13000080:engine routines:ENGINE_load_private_key:failed loading private
key:crypto/engine/eng_pkey.c:79:
pkeyutl: Error initializing context
```

It seems it is not possible to encrypt a file with an AES key. More information:

- https://github.com/openssl/openssl/issues/8719
- https://github.com/openssl/openssl/issues/10265

29. Sign a file with an existing ECC key.

    Create a file to encrypt:

```
This file was created to be encrypted.
Just a regular text file.
```

30. Perform the signing:

Module protection

```
% openssl dgst -sha256 -engine pkcs11 -keyform engine \
  -sign pkcs11:token=loadshared%20accelerator;pin-value=ncipher;object=ecc_key_1 \
  -out /tmp/filetoencrypt.sig /tmp/filetoencrypt.txt

Engine "pkcs11" set.
2024-04-17 15:10:15 [106907] t40ac5c9f757f0000: pkcs11: 000008CB Error:  pData: length 32 but NULL buffer
```

Softcard protection

```
% openssl dgst -sha256 -engine pkcs11 -keyform engine \
  -sign pkcs11:token=testSC;pin-value=ncipher;object=ecc_key_1 \
  -out /tmp/filetoencrypt.sig /tmp/filetoencrypt.txt

Engine "pkcs11" set.
```

```
2024-04-17 15:12:59 [107022] t405c6f45a57f0000: pkcs11: 000008CB Error:  pData: length 32 but NULL buffer
```

2.6. FIPS Level 3 remarks and recommendations

Recommendations for when a FIPS Level 3 world file is used for the HSM configuration:

- Create an OCS card 1/N where N is at least the number of HSMs being used in the configuration.
- Leave the OCS card inserted on HSM used in the configuration.
- The OCS card must be present any time new key material is created (FIPS authorization).

Chapter 3. Additional resources and related products

3.1. nShield HSMs

3.2. nShield as a Service

3.3. Entrust products

3.4. nShield product documentation