Web Services Option Pack

nShield Web Services PKCS 11 Provider v3.3.0 User Guide

12 June 2024

Table of Contents

1. Overview: Web Services PKCS #11 library

The Web Services PKCS #11 Library allows you to run PKCS #11 applications from a Web Services client.

The client machine that the PKCS #11 library is installed to does not require any existing Web Services or Security World Software to be present.

2. Install the Web Services PKCS #11 library

2.1. Linux

To install the nShield PKCS #11 library:

1. Sign in as a user with root privileges.
2. Change to the root directory.
3. Extract the PKCS #11 tar to the root directory. This installs all the files required by PKCS #11 to `/opt/nfast/webservices/pkcs11`.

   ```
   tar -xzf /path/to/nShield-WebServicesClient-Linux-1.2.0.tar.gz
   ```

4. If you are installing the PKCS #11 library for the first time, go to the `/opt/nfast/webservices/pkcs11/conf` folder and copy the `example_pkcs11webservices.cfg` file to `pkcs11webservices.cfg`.

   If you are reinstalling or updating the PKCS #11 library and do not want to overwrite your existing `pkcs11webservices.cfg` file, you can skip this step.

5. If required, change the log file destination or grant permission to the appropriate user to write to the default log folder: `/opt/nfast/webservices/pkcs11/log`. For help changing the logging path, see Log file path on Linux.

2.2. Windows

To install the nShield PKCS #11 library:

1. Ensure that `Microsoft Visual C++ 2015-2022 Redistributable (x64)` has been installed. A supported version is included in the installation media or can be downloaded from https://learn.microsoft.com/en-us/cpp/windows/latest-supported-vc-redist?view=msvc-170.
2. Sign in as Administrator or as a user with local administrator rights.
3. Using the provided installation media, launch `setup.msi` manually.
4. Follow the onscreen instructions.
5. Accept the license terms and select **Next** to continue.

6. Specify the installation directory and select **Next** to continue.
7. If you are installing the PKCS #11 library for the first time, go to the `C:\ProgramData\nCipher\webservices\pkcs11\conf` folder and copy the `example_pkcs11webservices.cfg` file to `pkcs11webservices.cfg`.

   If you are reinstalling or updating the PKCS #11 library and do not want to overwrite your existing `pkcs11webservices.cfg` file, you can skip this step.
8. If required, change the log file destination or grant permission to the appropriate user to write to the default log folder: `C:\ProgramData\nCipher\webservices\pkcs11\log`. For help changing the logging path, see Log file path on Windows.

3. Configure the Web Services PKCS #11 library

The `pkcs11webservices.cfg` file contains an example PKCS #11 configuration. To use this file, you need to alter it to point to the correct certificate locations.

3.1. Default PKCS #11 configuration

The PKCS #11 library is installed with a default configuration. Entrust recommends reviewing and updating the initial configuration before the library is called to ensure that all configuration settings are appropriate for the deployment environment.

Pay special attention to the TLS connection and logging to ensure that the system is used securely.

Ensure that the configuration file and TLS authentication files have restrictive access control, so that only the application using the PKCS #11 client library has access to these files.

3.2. Configuration parameters for both Windows and Linux

The following configuration options are shared on both Linux and Windows.

3.2.1. Web Services server hostname

```
HOST <host name of server>
```

Set this to the Web Services server host name.

3.2.2. Web Services server port number

```
PORT 18001
```

Set this to the Web Services server port number.

3.2.3. Log level

The `LOGLEVEL` field in the configuration file controls the PKCS #11 library logging. The available log levels are:

- `0(None)`
- `1(Fatal)`
- `2(Error)`
- `3(Warning)`
- `4(Debug 1)`
- `5(Debug 3)`
- `6(Debug 9)`

By default, the PKCS #11 logging level is set at 3 (Warning).

Higher log levels include all logs from lower levels. If the logging level is set to `4(Debug 1)`, you only get log events with Debug 1, Warning, Error, and Fatal. If the logging level is set to `6(Debug 9)` it enables all the log levels. To turn off logging set `LOGLEVEL` to `0(None)`.

```
LOGLEVEL 3
```

Logs that are not at DEBUG9 level have the message prefixed with standard fields:

- `Timestamp with format [yyyy-mm-dd hh:mm:ss]`
- `Process ID`
- `Level (one of DEBUG3 DEBUG1 WARNING ERROR FATAL)`
- `PKCS #11 session handle (zero if not used)`
- `PKCS #11 library name`

Logs at DEBUG9 level add field for thread ID in prefix:

- `Timestamp with format [yyyy-mm-dd hh:mm:ss]`
- `Process ID`
- `Thread ID`
- `Level (DEBUG9)`
- `PKCS #11 session handle (zero if not used)`
- `PKCS #11 library name`

3.2.4. Enable logging to console

By default, PKCS #11 has console logging disabled. In order to enable this, set `LOGSTDOUT` to `1` in the configuration file.

```
LOGSTDOUT 1
```

3.2.5. Retry Web Services communication

Since the PKCS #11 library operates over a network connection, it is possible that network fluctuations could cause an internal request to the Web Services server to fail. To ensure the reliability and robustness of PKCS #11 applications, the request can be retried.

You can configure the maximum number of retries for each request and the delay (in seconds) between each retry. For example:

```
MAXRETRIES 5
RETRYDELAY 10
```

If you don't define either of these entries, the PKCS #11 library uses the defaults.

- For `MAXRETRIES`, the default is `5` and the maximum number is `256`.

  To disable retries, set `MAXRETRIES` to `0`.

- For `RETRYDELAY`, the default is `10` seconds. There is no maximum value but higher values will reduce performance on unstable systems. Only integer values are supported.

  To disable the retry delay, set `RETRYDELAY` to `0`. This will cause retries to occur consecutively without delay.

3.3. Configuration parameters for Linux only

For details on creating certificates, see Server/client mutual authentication with the Web Services PKCS #11 library.

3.3.1. TLS certificate

```
CERT <path to TLS certificate>
```

Set this to the file path of the client TLS certificate.

3.3.2. TLS private key

```
KEY <path to private key>
```

Set this to the file path of the client TLS private key.

3.3.3. TLS client authentication

```
AUTH <path to TLS CA Certificate for Mutual Authentication>
```

Set this to the server root certificate, which forms the trust anchor for authenticating the server's certificates received during TLS handshake.

3.3.4. Log file path on Linux

By default, PKCS #11 outputs the logs to `/opt/nfast/webservices/pkcs11/log/pkcs11webservices.log`. To change the default path set `LOGFILEPATH` to any valid existing path.

```
LOGFILEPATH /opt/nfast/webservices/pkcs11/log/pkcs11webservices.log
```

To enable logging for a user, change the path to a Linux user path so that the user has write permission to the folder.

```
LOGFILEPATH /home/<username>/log/pkcs11webservices.log
```

3.3.5. Enable logging to syslog

You can direct PKCS #11 logs to a syslog server. By default this configuration is disabled. In order to enable this, set `LOGSYS` to `1` in the configuration file.

```
LOGSYS 1
```

3.4. Configuration parameters for Windows only

For details on creating certificates, see Server/client mutual authentication with the Web Services PKCS #11 library.

3.4.1. TLS thumbprint

```
THUMBPRINT <storetype>\<store>\<thumbprint>
```

Certificate thumbprint.

storetype is `LocalMachine` or `CurrentUser`

Example:

```
THUMBPRINT LocalMachine\My\F46DFD2118E64D0A8E0D5FCD0E0987D5C21BB2D5
```

For details on finding the thumbprint of a certificate see Server/client mutual authentication with the Web Services PKCS #11 library.

If certificates are stored on `LocalMachine`, non-administrator users may lose access to their private key.

If a certificate is updated, a new thumbprint will be generated and will need to be copied over to the THUMBPRINT line.

3.4.2. Log file path on Windows

By default, PKCS #11 outputs the logs to `C:\ProgramData\nCipher\WebServices\pkcs11\log\pkcs11webservices.log`. To change the default path set `LOGFILEPATH` to any valid existing path.

```
LOGFILEPATH C:\ProgramData\nCipher\WebServices\pkcs11\log\pkcs11webservices.log
```

To enable logging for a user, change the path to a Windows user path so that the user has write permission to the folder.

```
LOGFILEPATH C:\Users\<username>\log\pkcs11webservices.log
```

3.5. Enable logging to event viewer

You can direct PKCS #11 logs to event viewer. By default this configuration is

disabled. In order to enable this, set EVENTLOG to 1 in the configuration file.

```
EVENTLOG 1
```

Messages in event viewer are limited to a log level of 2(Error) and lower. Higher values will not produce additional events.

3.6. Enable fake session public keys

Entrust Certificate Authority generates key pairs where the public key is a session key with CKA_TOKEN set to false. In normal operation, the PKCS #11 library does not support generation of session keys so this special mode is required.

You can direct PKCS #11 to allow the generation of public keys in a key pair that appear like session keys while the session remains open. By default this feature is disabled. In order to enable this, set ENTRUSTCASUPPORT to 1 in the configuration file.

When this feature is enabled, a fake session public key can be generated with C_GenerateKeyPair if CKA_TOKEN is set to false in the public key template.

```
ENTRUSTCASUPPORT 1
```

The fake session keys generated are stored in the database like any other token public key with CKA_TOKEN true. They will be seen as token keys in any other sessions or any future sessions.

4. Server/client mutual authentication with the Web Services PKCS #11 library

4.1. Overview

The Web Services PKCS #11 Library can only communicate securely with a WSOP Server if the following certificates are installed:

- The WSOP Server's CA certificate.
- An appropriate client certificate (with each PKCS #11 client using its own client certificate).
- Any intermediate CA certificates that are to be used to form a complete chain to verify the client certificate on the WSOP Server.

The following sections in the *nShield Web Services Option Pack User Guide* contain information concerning the PKCS #11 Library's client certificates:

- *Web Services client authentication* describes the Web Services Option Pack client authentication, client certificate based virtual partitioning, and the client certificate revocation.
- *TLS client authentication* describes how the client 's CA (root) certificate for the PKCS #11 Library's client's certificate hierarchy is configured in the WSOP server.
- *Appendix A: Security guidance for the Web Services Option Pack* describes some important security considerations associated with the PKCS #11 Library's client's certificate hierarchy.

The 'extended key usage' extension of a PKCS #11 Library's client (end-entity) certificate should be set to 'TLS Web client authentication'.

4.2. Linux

The following guidance should be followed when installing the PKCS #11 Library's certificates in a Linux platform.

- If intermediate certificates are used in the PKCS #11 Library's client certificate hierarchy, they should be included in the same file as the client certificate. The root client certificate should not be included in this file.
- The recommended format for the client certificate file is PEM.

- The recommended format for the client certificate key file is PEM.

For details on the Linux config file entries, see Configuration parameters for Linux only.

4.3. Windows

The following guidance should be followed when installing the PKCS #11 Library's certificates in a Windows platform.

1. Install the WSOP Server's CA certificate into the `Root` store.
   Below is an example of how you can do this using `certutil.exe`.
   a. Add a CA certificate to the Root store

   ```
   certutil.exe -addstore Root <ca_certificate.pem>
   ```

   b. Check that the certificate has been installed:

   ```
   certutil.exe -store Root
   ```

2. Install any intermediate CA certificates for the client certificate.
   Below is an example of how you can do this by using `certutil.exe` to load the client certificate's intermediate certificates into the 'CA' certificate store:

   ```
   certutil.exe -addstore CA <intermediate_ca_certificate.pem>
   ```

3. Install the client certificate and its private key. This should be a PFX file that contains a single certificate and the associated private key.

   

   The PFX must not contain the full certificate chain.

   For example, to install the PFX file in the Local Machine's certificate store

   ```
   certutil.exe -p <password> -importPFX [certificatestorename] <client-cert.pfx>
   ```

   or to install the PFX file into the Current User's certificate store

   ```
   certutil.exe -p <password> -importPFX -user [certificatestorename] <client-cert.pfx>
   ```

To find the thumbprint of a certificate, open Certificate view dialog box with the command below, select the details tab and scroll down to the thumbprint field.

For example, to view thumbprints in `My` store.

```
certutil.exe -viewstore My
```

PowerShell can also be used to see the thumbprint on a list. In this example store is `My` and store type is `LocalMachine`.

```
PS Get-ChildItem -Path Cert:LocalMachine\My
```

If necessary, update `pkcs11webservices.cfg` with the changed thumbprint.

For details on the Windows config file entries, see Configuration parameters for Windows only

5. Utilities in the Web Services PKCS #11 library

The following four utility programs are provided:

`ckcheckinst`	Checks basic functionality.
`ckinfo-dynamic`	Prints version information.
`cklist-dynamic`	Lists objects created on the Softcard.
`ckmechinfo`	Lists supported mechanisms.

Run these programs with the following commands:

Linux:

```
/opt/nfast/webservices/pkcs11/bin/ckcheckinst
/opt/nfast/webservices/pkcs11/bin/ckinfo-dynamic --library
/opt/nfast/webservices/pkcs11/lib/libpkcs11webservices.so
/opt/nfast/webservices/pkcs11/bin/cklist-dynamic --library
/opt/nfast/webservices/pkcs11/lib/libpkcs11webservices.so
/opt/nfast/webservices/pkcs11/bin/ckmechinfo
```

Windows:

```
C:\Program Files\nCipher\WebServices\pkcs11\bin\ckcheckinst.exe
C:\Program Files\nCipher\WebServices\pkcs11\bin\ckinfo-dynamic.exe --library "C:\Program
Files\nCipher\WebServices\pkcs11\lib\libpkcs11webservices.so"
C:\Program Files\nCipher\WebServices\pkcs11\bin\cklist-dynamic.exe --library "C:\Program
Files\nCipher\WebServices\pkcs11\lib\libpkcs11webservices.so"
C:\Program Files\nCipher\WebServices\pkcs11\bin\ckmechinfo.exe
```

5.1. Softcard generation tool

Because PKCS #11 does not directly support Softcard generation, a command line tool is provided.

The Softcard tool uses the same configuration file as the PKCS #11 library for the Web Services server secure connection. It does not support any logging. For more information, see Configure the Web Services PKCS #11 library.

Both single and double hyphen arguments are supported, for example, `-g` | `--generate`. You can also combine compatible single hyphen arguments, for example, you could use `-vg` instead of `-v -g`.

To generate a new Softcard run the following command:

Linux:

```
/opt/nfast/webservices/pkcs11/bin/softcardtool -g --name=<new-softcard-name>
```

Windows:

```
C:\Program Files\nCipher\WebServices\pkcs11\bin\softcardtool.exe -g --name=<new-softcard-name>
```

When prompted, enter a new passphrase for the Softcard.

Special characters for name and passphrase are not supported.

To verify the Web Services server connection, run the tool with the verbose and list options:

Linux:

```
/opt/nfast/webservices/pkcs11/bin/softcardtool -vl
```

Windows:

```
C:\Program Files\nCipher\WebServices\pkcs11\bin\softcardtool.exe -vl
```

To delete a Softcard, remove all keys associated with the Softcard and use the following command:

Linux:

```
/opt/nfast/webservices/pkcs11/bin/softcardtool -d --ID=<deleted-softcard-ID>
```

Windows:

```
C:\Program Files\nCipher\WebServices\pkcs11\bin\softcardtool.exe -d --ID=<deleted-softcard-ID>
```

To see all the available options, run

Linux:

```
/opt/nfast/webservices/pkcs11/bin/softcardtool --help

softcardtool, 1.3.0

Usage:
       softcardtool [options]
```

Windows:

```
C:\Program Files\nCipher\WebServices\pkcs11\bin\softcardtool.exe --help

softcardtool, 1.3.0

Usage:
       softcardtool.exe [options]
```

```
Options:

Help options:
  -h, --help                Display help for `softcardtool'.
  -V, --version             Display the version number of `softcardtool'.
  -u, --usage               Display a brief usage summary for `softcardtool'.

Command options:
  -v, --verbose             Modify another command to be verbose.
  -a, --audit="LEVEL"       Displays rest client logs with a level lower than
                              or equal to the one provided. Select one of:
                              (FATAL < ERROR < WARNING < INFO < DEBUG)
  -l, --list                List softcards.
  -g, --generate            Generate a new softcard.
  -n, --name="NAME"         Name of softcard to generate, delete or retrieve.
  -d, --delete              Delete softcard by ID or name.
  -r, --retrieve            Retrieve a softcard's information by ID or name.
  -i, --id="ID"             ID of softcard to delete or retrieve. Ignored if
                              name is provided.

Generates, deletes, lists, and retrieves the details of softcards.

A softcard name may have leading/trailing spaces. If a name is supplied in the
initial prompt and contains spaces, it must be enclosed in double quotation
marks, which will be removed. For example:
      --generate --name "softcard name"         outputs: softcard name
      --generate, "Enter name:" softcard name   outputs: softcard name
Aside from the space character, names and passphrases including special
characters are not supported.
```

You cannot use `--delete` in conjuction with `--generate` or `--retrieve`. All other combinations are permitted and operate in the order of generate, delete, list, retrieve.

6. Web Services PKCS #11 library compliance with the PKCS #11 specification

6.1. Supported functions

The following sections list the PKCS #11 functions supported by the PKCS #11 library. For a list of supported mechanisms, see Mechanisms.

6.1.1. General purpose functions

The following functions perform as described in the PKCS #11 specification:

- `C_Finalize`
- `C_GetInfo`
- `C_GetFunctionList`
- `C_Initialize`.

6.1.2. Slot and token management functions

The following functions perform as described in the PKCS #11 specification:

- `C_GetSlotInfo`
- `C_GetTokenInfo`
- `C_GetMechanismList`
- `C_GetMechanismInfo`
- `C_GetSlotList`.

`C_GetSlotList` returns a list of slot IDs. You cannot make any assumptions about the values of these handles. These handles are not equivalent to the slot numbers returned by the Web Services server.

6.1.3. Standard session management functions

The following functions perform as described in the PKCS #11 specification:

- `C_OpenSession`

- `C_CloseSession`
- `C_CloseAllSessions`
- `C_Login`
- `C_Logout`
- `C_GetSessionInfo`

6.1.4. Object management functions

The following functions perform as described in the PKCS #11 specification:

- `C_CreateObject`
- `C_DestroyObject`
- `C_GetAttributeValue`
- `C_FindObjectsInit`
- `C_FindObjects`
- `C_FindObjectsFinal`
- `C_SetAttributeValue`

6.1.5. Encryption functions

The following functions perform as described in the PKCS #11 specification:

- `C_EncryptInit`
- `C_Encrypt`

6.1.6. Decryption functions

The following functions perform as described in the PKCS #11 specification:

- `C_DecryptInit`
- `C_Decrypt`

6.1.7. Sign functions

The following functions perform as described in the PKCS #11 specification:

- `C_SignInit`

- `C_Sign`

6.1.8. Verify functions

The following functions perform as described in the PKCS #11 specification:

- `C_VerifyInit`
- `C_Verify`

6.1.9. Key-management functions

The following function performs as described in the PKCS #11 specification:

- `C_GenerateKey`
- `C_GenerateKeyPair`

The Web Services PKCS #11 library does not currently support creating key objects with `C_CreateObject`. Use `C_GenerateKey` to generate a secret key object and `C_GenerateKeyPair` to generate a public/private key pair.

`C_GenerateKey` will only generate key types supported by the PKCS #11 library.

String fields such as CKA_LABEL and CKA_APPLICATION should not contain any HTML character for example <, > or & as this can result in those values becoming corrupted when they are retrieved later.

6.1.10. Message digesting functions

The following functions perform as described in the PKCS #11 specification:

- `C_DigestInit`
- `C_Digest`
- `C_DigestUpdate`
- `C_DigestFinal`

6.1.11. Random number generation function

The following function performs as described in the PKCS #11 specification:

- C_GenerateRandom

6.2. Objects

The following table lists the objects currently supported by the PKCS #11 library.

Object	Supported Types
CKO_DATA	
CKO_CERTIFICATE	CKC_X_509
CKO_SECRET_KEY	CKK_AES CKK_SHA256_HMAC CKK_SHA384_HMAC CKK_SHA512_HMAC
CKO_PRIVATE_KEY	CKK_RSA CKK_EC
CKO_PUBLIC_KEY	CKK_RSA CKK_EC
CKO_PROFILE	CKP_BASELINE_PROVIDER

The PKCS #11 provider implementation conforms to the *Baseline Provider Clause* defined in section 3.3 of *PKCS #11 Cryptographic Token Interface Profiles Version 3.0*.

6.3. Mechanisms

The following table lists the mechanisms currently supported by the PKCS #11 library and the functions available to each one.

Mechanism	Encrypt & Decrypt	Sign & Verify	SR & VR	Digest	Gen. Key/Key Pair	Wrap & Unwrap	Derive Key
CKM_AES_CBC_PAD	Y	—	—	—	—	—	—
CKM_AES_CBC	Y	—	—	—	—	—	—
CKM_AES_GCM	Y	—	—	—	—	—	—
CKM_AES_KEY_GEN	—	—	—	—	Y	—	—
CKM_SHA256_HMAC	—	Y	—	—	—	—	—

Mechanism	Encrypt & Decrypt	Sign & Verify	SR & VR	Digest	Gen. Key/Key Pair	Wrap & Unwrap	Derive Key
CKM_SHA256_HMAC_GENERAL	—	Y	—	—	—	—	—
CKM_SHA384_HMAC	—	Y	—	—	—	—	—
CKM_SHA384_HMAC_GENERAL	—	Y	—	—	—	—	—
CKM_SHA512_HMAC	—	Y	—	—	—	—	—
CKM_SHA512_HMAC_GENERAL	—	Y	—	—	—	—	—
CKM_SHA256_KEY_GEN	—	—	—	—	Y	—	—
CKM_SHA384_KEY_GEN	—	—	—	—	Y	—	—
CKM_SHA512_KEY_GEN	—	—	—	—	Y	—	—
CKM_SHA_1	—	—	—	Y	—	—	—
CKM_SHA224	—	—	—	Y	—	—	—
CKM_SHA256	—	—	—	Y	—	—	—
CKM_SHA384	—	—	—	Y	—	—	—
CKM_SHA512	—	—	—	Y	—	—	—
CKM_SHA3_224	—	—	—	Y	—	—	—
CKM_SHA3_256	—	—	—	Y	—	—	—
CKM_SHA3_384	—	—	—	Y	—	—	—
CKM_SHA3_512	—	—	—	Y	—	—	—
CKM_SHA256_RSA_PKCS_PSS	—	Y	—	—	—	—	—
CKM_SHA256_RSA_PKCS	—	Y	—	—	—	—	—
CKM_SHA384_RSA_PKCS_PSS	—	Y	—	—	—	—	—
CKM_SHA384_RSA_PKCS	—	Y	—	—	—	—	—
CKM_SHA512_RSA_PKCS_PSS	—	Y	—	—	—	—	—
CKM_SHA512_RSA_PKCS	—	Y	—	—	—	—	—

Mechanism	Encrypt & Decrypt	Sign & Verify	SR & VR	Digest	Gen. Key/Key Pair	Wrap & Unwrap	Derive Key
CKM_RSA_PKCS_KEY_PAIR_GEN	—	—	—	—	Y	—	—
CKM_RSA_PKCS_OAEP	Y	—	—	—	—	—	—
CKM_RSA_PKCS	Y	—	—	—	—	—	—
CKM_ECDSA_KEY_PAIR_GEN	—	—	—	—	Y	—	—
CKM_ECDSA_SHA256	—	Y	—	—	—	—	—
CKM_ECDSA_SHA384	—	Y	—	—	—	—	—
CKM_ECDSA_SHA512	—	Y	—	—	—	—	—

In this table:

- Y indicates that the function is supported by the mechanism.
- — indicates that the function is not supported by the mechanism.

6.3.1. Notes on Mechanisms

6.3.1.1. AES Mechanisms

The AES mechanisms support three different key sizes: 16, 24, and 32 bytes. For non-padded AES mechanisms the plaintext size must be a multiple of the block size (16 bytes).

For CKM_AES_GCM, the following restrictions apply to the CKM_GCM_PARAMS structure:

- ulIvLen must be 12 bytes.
- ulTagBits must be 128 bits.

CKM_AES_GCM is not supported with Security Worlds conforming to FIPS 140 Level 3.

6.3.1.2. HMAC Mechanisms

HMAC_GENERAL mechanisms support signature lengths ranging from half the

output size to the full output size. For example CKM_SHA256_HMAC_GENERAL supports outputs in the range 16-32 bytes.

6.3.1.3. RSA Mechanisms

For the CKM_RSA_PKCS_OAEP mechanism the hashAlg and the mgf values specified by CK_RSA_PKCS_OAEP_PARAMS must have the same SHA hash size. The supported pairs of values are as follows:

hashAlg	mgf
CKM_SHA_1	CKG_MGF1_SHA1
CKM_SHA256	CKG_MGF1_SHA256
CKM_SHA384	CKG_MGF1_SHA384
CKM_SHA512	CKG_MGF1_SHA512

For a hash length h and RSA modulus length k in bytes, the longest message that can be encrypted is k-2h-2 bytes long.

For CKM_SHA*_RSA_PKCS_PSS mechanisms the hashAlg and the mgf values specified by CK_RSA_PKCS_PSS_PARAMS must have the same SHA hash size.The sLen value is expected to be the length of the message hash in bytes.

The supported sets of values for hashAlg, mgf and sLen are as follows:

hashAlg	mgf	sLen
CKM_SHA256	CKG_MGF1_SHA256	32
CKM_SHA384	CKG_MGF1_SHA384	48
CKM_SHA512	CKG_MGF1_SHA512	64

6.3.1.4. ECDSA Mechanisms

For ECDSA mechanisms, the CKA_EC_PARAMS attribute must be supplied for key generation. Its value is that of the object identifier format (oId), supplied as a byte array. Only the curves below are supported.

Curve	oId
P-256	{ 0x06, 0x08, 0x2a, 0x86, 0x48, 0xce, 0x3d, 0x03, 0x01, 0x07 }

Curve	oId
P-384	{ 0x06, 0x05, 0x2b, 0x81, 0x04, 0x00, 0x22 }
P-521	{ 0x06, 0x05, 0x2b, 0x81, 0x04, 0x00, 0x23 }

For all supported ECDSA mechanisms, the field size has a minimum of 256 bits and a maximum of 512.

For more information, including minimum and maximum key sizes, run `ckmechinfo` as described in Utilities in the Web Services PKCS #11 library.

6.4. Attributes

If providing these attributes when creating an object or generating a key, they must be set to `CK_FALSE`:

- `CKA_DERIVE`
- `CKA_EXTRACTABLE`
- `CKA_COPYABLE`
- `CKA_TRUSTED`
- `CKA_UNWRAP`
- `CKA_WRAP`
- `CKA_WRAP_WITH_TRUSTED`

If providing these attributes when creating an object or generating a key, they must be set to `CK_TRUE`:

- `CKA_SENSITIVE`
- `CKA_DESTROYABLE`

If providing these attributes when creating an object, they must be set to `CK_TRUE`:

- `CKA_ALWAYS_SENSITIVE`
- `CKA_NEVER_EXTRACTABLE`
- `CKA_LOCAL`

This attribute must be set to `CK_TRUE` when creating an object or generating a key:

- `CKA_TOKEN`

This read-only attribute can be included in a search template provided to

`C_FindObjectsInit` and can be retrieved using `C_GetAttributeValue`:

- `CKA_UNIQUE_ID`

These attributes can be set using `C_SetAttributeValue`:

- `CKA_EXTRACTABLE` (only from `CK_FALSE` to `CK_FALSE`)
- `CKA_LABEL`

CKA_PUBLIC_EXPONENT value, if provided, must be an odd number greater than 2 and less than 0x7FFFFFFF and should be prime. The bit length of CKA_PUBLIC_EXPONENT must be less than half the key length. The PKCS #11 library only supports token objects, not session objects.