nShield Post-Quantum Software Development Kit

PQSDK v1.2.1 Release Notes

19 February 2025

Table of Contents

1. Introduction . . . 1
    1.1. Versions of these Release Notes . . . 1
2. Purpose of this release . . . 2
3. Features of nShield Post-Quantum Software Development Kit v1.2 . . . 3
    3.1. Sectorization . . . 3
    3.2. Chunked Merkle Tree Generation . . . 4
    3.3. Merkle Tree Caching . . . 5
    3.4. SEE Machine Versions . . . 5
    3.5. LMS Statistics . . . 5
4. Compatibility . . . 8
    4.1. Supported hardware . . . 8
    4.2. Supported operating systems . . . 8
    4.3. Supported Security World Versions . . . 8
        4.3.1. nShield PCIe HSMS . . . 8
        4.3.2. Rack-mounted nShield HSMs . . . 8
5. Required licenses . . . 10
    5.1. nShield XC . . . 10

1. Introduction

These release notes apply to version v1.2.1 of the nShield Post-Quantum Software Development Kit (PQSDK). They contain information specific to this release such as new features, defect fixes, and known issues.

The Release Notes may be updated with issues that have become known after this release has been made available. Please check https://nshieldsupport.entrust.com/hc/en-us/sections/360001115837-Release-Notes for the latest version of this document.

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

1.1. Versions of these Release Notes

Revision	Date	Description
1.1	2025-02-19	Version for v1.2.1. Fix in `liboqs`.
1.0	2025-01-28	Version for v1.2.0.

2. Purpose of this release

PQSDK v1.2 extends the support for Leighton-Micali hash based signatures (LMS RFC 8554) with three important features:

- Sectorization
- Chunked Merkle Tree Generation
- Merkle Tree Caching

In addition, PQSDK v1.2 offers full support for nShield 5c hardware security modules (HSMs).

The host C library and shared library have been tested using single-threaded applications and are not recommended for multi-threaded use at this time.

3. Features of nShield Post-Quantum Software Development Kit v1.2

To use LMS keys, either in this or previous releases, it is necessary to create a NVRAM delegation certificate so that the SEE machine has permission to create NVRAM files. To create this, use the `userdatatool.py` script supplied with PQSDK. An obsolete tool named `UserDataTool` is still available for the moment but is deprecated and should not be used.

3.1. Sectorization

Sectorization allows LMS keys to be partitioned across a number of Remote Administrator ready smart cards. These cards can be used to provision one or more nShield hardware security modules with a block of one-time signature keys. When these signing keys are exhausted no more signatures may be created until additional provisioning is done.

You MUST NOT use smart cards that are not Remote Administrator ready for LMS sectorization. These cards lack sufficient protection.

Sectorization can be done from the command line using the example program `pipsqueak` (previously named `pqsdk-test`). To use `pipsqueak`, ensure that `pqsdk.dll` or `libpqsdk.so` are available, either in `%path%` or in the `LD_LIBRARY_PATH` environment variable.

Here is an example of how `pipsqueak` can be used to create an LMS key pair:

```
$ ./pipsqueak generate algorithm=lms alias=lms-test \
  lmscode=sha256_m32_h10 lmotscode=sha256_n32_w8 \
  expcards=1 explease=2
```

In this example the LMS key will have a Merkle tree height of 10 and therefore can create 1024 (2^10) signatures over its lifetime. LMS keys with a height of up to 25 are supported, but these will take many hours to generate.

Note that expcards is one and explease is two. The pipsqueak program will prompt the user to insert two smart cards if necessary. Remote Administrator allows configuration of dynamic slots. If there are enough blank cards in the physical and dynamic slots to write all necessary cards pipsqueak will do so without prompting.

When this command is complete there will be no nShield HSMs that can sign with this LMS key. All blocks of one-time signature keys are stored on smart cards. In this example, we provision an nShield HSM:

```
$ ./pipsqueak provision
```

If there are more than one LMS card inserted into a slot of some nShield HSM this command will prompt the user to select one. To avoid prompting, specify the electronic serial number and slot using the "provesn" and "slot" options:

```
$ ./pipsqueak provesn=F0C3-4E26-48A2 slot=2
```

Once this is done there should be signatures available. To see how many signatures are available, use the `showinfo` option:

```
$ ./pipsqueak showinfo alias=lms-test
PQSDK: 1.2 (F0C3-4E26-48A2: 1.2, 5016-03E0-D947: 1.2)
  F0C3-4E26-48A2
  - Slot 2: ic=23 lms
  7F53-7852-7240
  5016-03E0-D947
Alias: lms-test
Algorithm: LMS sha256_m32_h10 sha256_n32_w8
  Remaining: 128 signatures
Public key (alias: lms-test):
  00000006 00000004 f240df42 dc7de7c1 ec26d453 38658885 5315919c
  68fe0b2b 3117da26 0ffa3e14 12f9e3b6 534bc356 0b36444f fb68fa0b
```

Notice that a single provision made 128 signatures available. That comes from the height (10) the card exponent (1) and lease exponent (2): subtracting the card and lease exponents from the height gives seven and 2^7 is 128.

A single nShield HSM can be provisioned up to twice. It is not necessary for the leases to be contiguous. In other words, a card with four provisions can be used on HSM A, then on HSM B and then on HSM A again. This is intended to help avoid running out of signatures. When one lease is running low the user can provision a second one without waiting for it to be exhausted. However, attempting to add a third lease will fail until the first lease has been used up.

3.2. Chunked Merkle Tree Generation

In previous PQSDK releases, an LMS key pair was generated using a single call to the SEE machine. This is still possible if no sectorization is requested, but in this release key generation can be separated into many steps. This avoids problems with time outs for LMS keys with larger heights, such as 20 or 25. In addition, this allows client programs to display progress to users.

The new generateLMS command returns the number of chunks and cards remaining. It is necessary to call the continueLMS command until both of these are zero, at which point the

LMS key will be saved to the Security World key management data directory. The number of chunks is determined by the chunk exponent argument to generateLMS. If that exponent is n, there will be 2^n + 1 chunks necessary.

```
$ ./pipsqueak generate algorithm=lms alias=lms-test \
    lmscode=sha256_m32_h10 lmotscode=sha256_n32_w8 \
    expchunk=2
PQSDK: 1.2 (5016-03E0-D947: 1.2)
Generate LMS key started: lms-test (0.106 seconds)
  chunk: lms-test (4.394 seconds) 4 remaining
  chunk: lms-test (4.375 seconds) 3 remaining
  chunk: lms-test (4.373 seconds) 2 remaining
  chunk: lms-test (4.373 seconds) 1 remaining
  chunk: lms-test (0.091 seconds) 0 remaining
Public key (alias: lms-test):
  00000006 00000004 e3d540d8 591b7ed5 7b84a0b3 fd7b9208 12901560
  2fefd704 5820de5a 8bd9ad88 61b7e857 d8e766a3 c5043783 ed95e91b
```

3.3. Merkle Tree Caching

In previous releases, the Merkle tree had to be completely reconstructed each time an LMS key was loaded after the SEE machine had been restarted. This was a problem for larger tree heights. To address this, PQSDK now caches the top portion of the tree in the Security World key management data directory. This means only a portion of the Merkle tree needs to be generated again when an existing key is loaded.

3.4. SEE Machine Versions

In previous releases, a command named see_getVersion was used to fetch the version num ber of a single SEE machine. This command is now internal and should not be used. Instead, use getSEEVersions which will respond with an array with version numbers for each SEE machine available.

all commands that begin with the `see_` prefix are internal. They are meant to be called by host commands which usually handle additional tasks that must be done to make the command convenient.

3.5. LMS Statistics

The following are measurements of LMS signature sizes and key generation times. These are intended to assist in planning. Key generation times vary depending on many factors, including the chunk size used. They have been measured in laboratory conditions with no external load and in some cases (when marked with a ~ character) are estimated by extrapo lation the timing of partial chunks of a key that was not completely generated.

```
sha256/w1/h5
  signature: 8684 bytes
  XC: 0.7 seconds
  n5: 0.1 seconds
sha256/w8/h5
  signature: 1292 bytes
  XC: 0.7 seconds
  n5: 0.5 seconds
shake/w1/h5
  signature: 8684 bytes
  XC: 0.8 seconds
  n5: 0.2 seconds
shake/w8/h5
  signature: 1292 bytes
  XC: 5.4 seconds
  n5: 1.8 seconds
sha256/w1/h10
  signature: 8844 bytes
  XC: 3.5 seconds
  n5: 2.0 seconds
sha256/w8/h10
  signature: 1452 bytes
  XC: 17.4 seconds
  n5: 16.7 seconds
shake/w1/h10
  signature: 8844
  XC: 18.9 seconds
  n5:  6.0 seconds
shake/w8/h10
  signature: 1452 bytes
  XC: 171.5 seconds
  n5:  58.5 seconds
sha256/w1/h15
  signature: 9004 bytes
  XC: 75.7 seconds
  n5: 64.4 seconds
sha256/w8/h15
  signature: 1612 bytes
  XC: 542.2 seconds
  n5: 538.0 seconds
shake/w1/h15
  signature: 9004 bytes
  XC: 557.9 seconds
  n5: 192.4 seconds
shake/w8/h15
  signature: 1612 bytes
  XC: 5552.4 seconds
  n5: 1870.6 seconds
sha256/w1/h20
  signature: 9164 bytes
  XC: ~8720 seconds
  n5: ~2060 seconds
sha256/w8/h20
  XC: ~2451 seconds
  n5: ~2058 seconds
shake/w1/h20
  XC: ~22704 seconds
  n5:  ~6162 seconds
shake/w8/h20
  XC: ~178057 seconds
  n5:  ~59859 seconds
sha256/w1/h25
  XC: ~77626 seconds
  n5: ~73141 seconds
sha256/w8/h25
  XC: ~564396 seconds
```

```
  n5: ~551054 seconds
shake/w1/h25
  XC: ~745144 seconds
  n5: ~196975 seconds
shake/w8/h25
  XC: ~5511086 seconds
  n5: ~1914601 seconds
```

4. Compatibility

4.1. Supported hardware

This release is targeted at deployments with any combination of the following nShield HSMs:

- nShield Solo XC (Base, Mid, High)
- nShield Connect XC (Base, Mid, High, Serial Console)

4.2. Supported operating systems

This release has been tested for compatibility with the following operating systems:

- Red Hat Enterprise Linux 8 x64
- Additional mainstream x64 based Linux distributions other than the one listed above may be compatible, however Entrust cannot guarantee this compatibility.

4.3. Supported Security World Versions

The following tables show the supported configuration of firmware and software installations.

4.3.1. nShield PCIe HSMS

HSM Type	Firmware Version	Security World Version	CodeSafe Version
nShield Solo XC	v12.72.1	v12.70.4	v12.70.4
nShield Solo XC	v12.72.1	v13.4.3	v13.4.3

4.3.2. Rack-mounted nShield HSMs

HSM Type	Image Version	Firmware Version	Security World Version	CodeSafe Version
nShield Connect XC	v12.80.5	v12.72.1	v12.70.4	v12.70.4
nShield Connect XC	v12.80.5	v12.72.1	v13.4.3	v13.4.3
nShield 5c	v13.6.5	v13.5.1	v13.6.5	v13.6.5

Other combinations of Firmware, Security World software, and CodeSafe software may work but are not supported at this time.

5. Required licenses

Please use the Feature Enable Tool (FET) to set and view enabled features.

5.1. nShield XC

The Unrestricted SEE (`SEE Activation (EU+10)`) feature must be enabled on your HSM. Restricted SEE (`SEE Activation (Restricted)`) is not currently supported.