AN13720

PN7642 Secure Key Mode demo application

Rev. 1.1 — 13 March 2023 Application note

Document information

Information	Content
Keywords	PN7642, open controller, Secure Key Mode, demo application
Abstract	This document describes a demo application to showcase the Secure Key Mode commands of PN7642 frontend controller. The scope of this document is to describe the way of working with the demo application on PN7642 FAMA board.

Revision history

Revision history

Rev	Date	Description
1.1	20230313	Security status changed into public
1.0	20230110	Initial release

1 Introduction

This document describes the demo application, to showcase the PN7642 Secure Key Mode, to perform different key store operations. Using the PN76 family development board connected to the LPC55S16 evaluation board, or with a standalone PN76 family development board by using system services as interface.

For more information on the operation of PN7642 NFC frontend controller, refer to the data sheet and other available documents.

The document describes only the application demo options to work with Secure Key Mode commands with sample keys.

2 Software and hardware requirements

2.1 Software requirements

- MCUXpresso IDE v11.6.1 (onwards)
- Python v3.8 onwards
  - cryptodome (installed by using command "*`pip install pycryptodome`*")
- Debugger: SEGGER J-Link v7.70d (onwards) or MCU-Link

2.2 Hardware requirements

- PN76 development board hosting a PN7642 (C100 onwards)

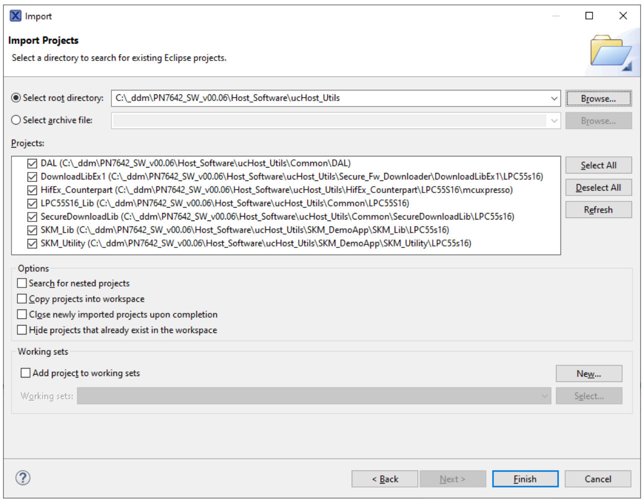
3 Steps to load projects in MCUXpresso, build, execute

The Secure Key Mode application can either work with an external host (LPC55s16) or within PN7642 application space.

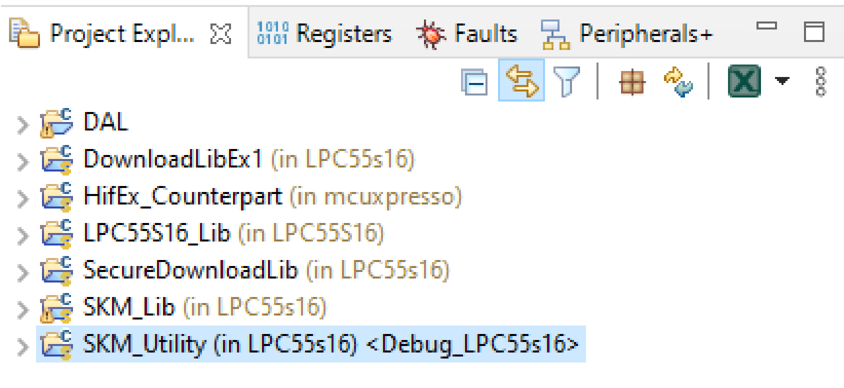
In this section, we explain about how to load the Secure Key Mode demo application in MCUXpresso IDE.

3.1 Steps to load projects in MCUXpresso to work with external host (LPC55s16), build, execute

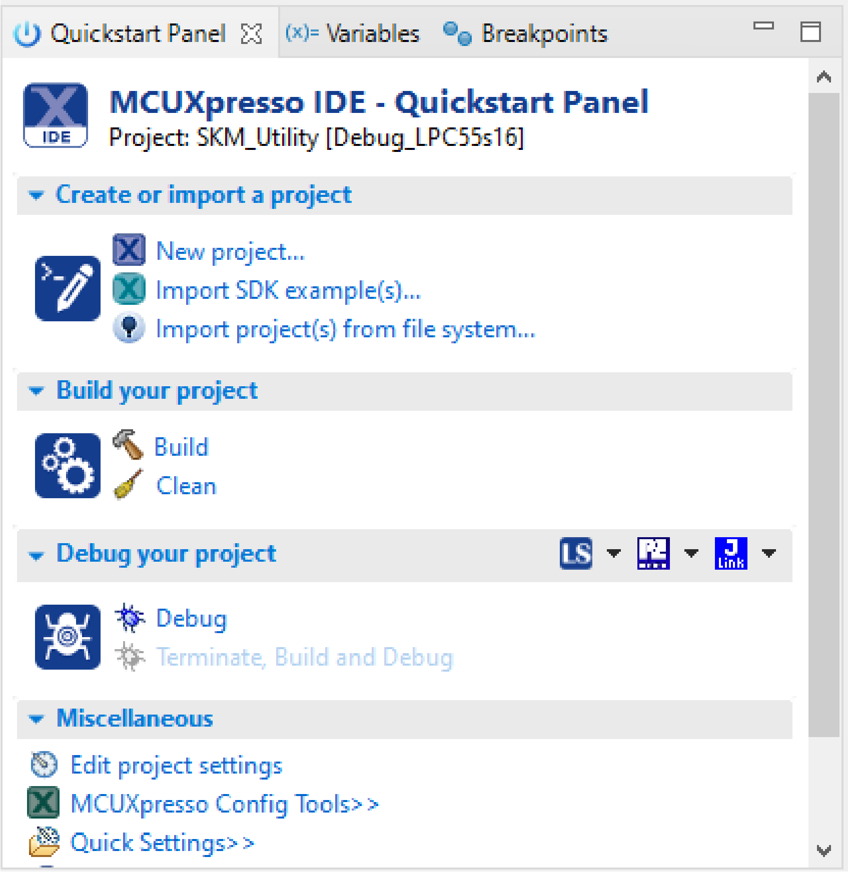
1. Open the MCUXpresso tool and select a project workspace
2. Import projects by selecting menu options “File➔Import” then select “Existing projects into workspace”.
3. Then select the **<PN7642_SW_Extracted_Directory>\Host_Software\ucHost_Utils** to import. The following projects will be displayed and select as shown in below diagram:



4. Now the “Project window” should look as below:



5. Now select the project “SKM_Utility” and “build” the application using links provided in the “QuickStart Panel”.



6. Once the project is built, make sure to connect USB port in LPC55s16 board that enumerates as LPC-Link port.
7. Now download the binary by using the **Debug** links provided in the **QuickStart Panel** and select the corresponding LPCLink that was connected to LPC55s16 board.
8. After binary is downloaded, now select the menu option Run➔Resume to execute the application binary.
9. Once the application is executed, you will see the following menu in the **Console** window-tab in MCUXpresso.

Figure 1. Initial Main menu on Console window of MCUXpresso IDE

```
******** Secure Key Mode ********
Select the Option
- Enter 00 for Entering into Secure Key Mode (SKM).
- Enter 01 for Get DieID.
- Enter 02 for Open Session.
- Enter 03 for Provisioning APP_ROOT_KEY.
- Enter 04 for Provisioning APP_MASTER_KEY.
- Enter 05 for Provisioning APP_FIXED_KEY.
- Enter 06 for Deleting APP_MASTER_KEY.
- Enter 07 for Deleting APP_FIXED_KEY.
- Enter 08 for Updating APP_MASTER_KEY.
- Enter 09 for Updating APP_FIXED_KEY.
- Enter 10 for Locking Provisioning of APP_ROOT_KEY.
- Enter 11 for Getting the SKM State.
- Enter 12 for Provisioning a APP_ASYMM_KEY auto generated.
```

```
- Enter 13 for Provisioning a APP_ASYMM_KEY provided as plain.
- Enter 14 for Provisioning a APP_ASYMM_KEY provided as encrypted
- Enter 15 for Deleting a APP_ASYMM_KEY.
- Enter 16 for Setting the domain parameters for the CUSTOM ASYMM_KEY.
- Enter 17 for Getting the Public key of an APP_ASYMM_KEY.
- Enter 18 for Purging the APP ROOT KEY.
- Enter 19 to SKM RESET.
- Enter 20 to perform SOFT RESET.
Select Option (Hit Enter after Input) :
```

10. The menu is self-explanatory. The user can execute the menu options for different functionalities provided by the Secure Key Mode (SKM).

3.2 Steps to load projects in MCUXpresso to work with application space in PN7642 IC itself

In this option, a Secure Key Mode demo application is provided as part of SDK itself.

Select demo application **\Demo_Apps\pn_skm** when **Import SDK example(s)** option is selected in **Quickstart Panel**.

4 Working with crypto scripts

In this section, we explain the python scripts, used to generate the encrypted key data, derive a key from the input key and so on.

These scripts may be used to generate the data, which shall be given as input to the many secure key mode commands. The data must be in a particular format, which is explained in further chapters.

These scripts are present at **<PN7642_SW_Extracted_Directory>\Host_Software\Scripts** directory.

4.1 Deriving a key

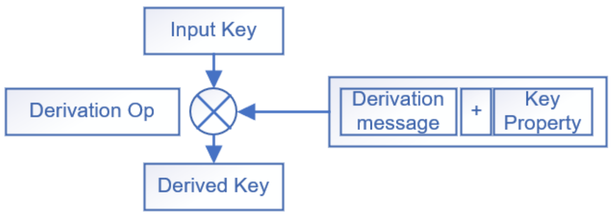


Figure 2. Deriving a key

In this operation, an input key is derived using a derivation message with the key property. This derived key can be used only for the corresponding operations as provided in the key property.

4.2 Generating an encrypted key data

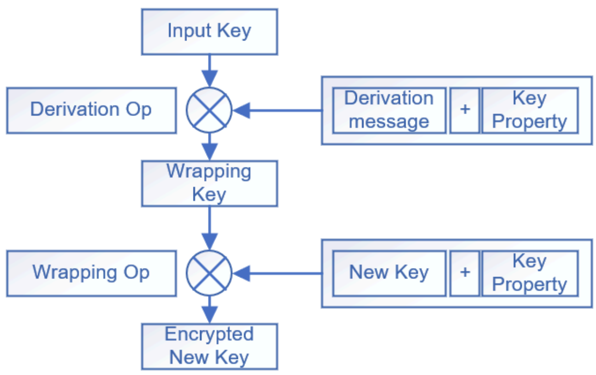


Figure 3. Generating an encrypted key data for a given key

In this operation, a wrapping key is derived from an input key with a derivation message. Using this wrapping key, a new key is wrapped to generate the encrypted key data.

This encrypted key data and the derivation message for wrapping keys, is therefore provided as input to the different secure key mode commands, to perform the unwrapping of the key to store or refer.

Table 1. Note about Key operation

!	Use either 128-bit or 256-bit keys only for a given operation. Mix of 128-bit and 256-bit keys is not allowed in the same operation as input key, wrapping key and so on.

4.3 Script options to be used

4.3.1 Type of keys

Following table shows the options considered for different key types.

Table 2. Type of Keys options

Tool options	Description
-k	A new key
-rk	Input key
-wk	Wrapping key
-wiv	Wrapping init vector

4.3.2 Key operation

Following table shows the options considered for different key operations.

Table 3. Key operations

Tool options	Description
-w	Wrapping operation
-d	Key derivation operation

4.3.3 Key properties

Following options are provided for different key properties. Different key properties can be combined for the desired operation.

Following table shows the options considered for different key operations.

Table 4. Key properties

Tool options	Description
-lock	Indicating the key must be locked after loading into SGI
-wrapen	key can be used only for wrapping/unwrapping
-encen	key must be used only for encryption
-decen	key must be used only for decryption
-expen	generated key shall be exported to store in secure key store
-deren	Generated key must be used only for further derivation and not anything else

4.4 Providing the derivation message to script and providing it to secure key mode commands

The key derivation message shall be provided with an option **"-dd"**

Input to the script: The derivation message shall be of 32 bytes of data to be given in the following format:

```
  <dwData0> <dwData1> <dwData2> <dwData3> <dwData4> <dwData5> <dwData6>
<dwData7>
```

The dwData2 and dwData3 must be 0’s.

Input to the secure key mode commands: The above derivation message you have to provide in following format:

```
<dwData4> <dwData5> <dwData6> <dwData7> <dwData0> <dwData1>
```

For e.g., if the derivation data is:

```
dwData0 = 0xCBD963C6, dwData1 = 0xA8C4B0AC,
dwData4 = 0x6493F33F, dwData5 = 0xF06D625F,
dwData6 = 0x9322F8A2, dwData7 = 0xAEA1B95B
```

Then message provided to the script is:

```
CBD963C6A8C4B0AC00000000000000006493F33FF06D625F9322F8A2AEA1B95B
```

Then derivation message to secure key mode command is:

```
6493F33FF06D625F9322F8A2AEA1B95BCBD963C6A8C4B0AC
```

4.4.1 Generating a wrapping key from an input key

Key property: -lock -wrapen

Table 5. Generating a wrapping key from an input key

Tool options	Description
command	> python cryptoWrapperReference.py -d -rk 68029E29E29FB77F6AE9F0A9D1F0EE0F -dd CBD963C6A8C4B0AC00000000000000006493F33FF06D625F9322F8A2AEA1B95B -lock -wrapen
output data	Root key: 68029e29e29fb77f6ae9f0a9d1f0ee0f Derivation data: cbd963c6a8c4b0ac00000000000002106493f33ff06d625f9322f8a2aea1b95b Derived key: 19e1c19f4d5a71efcac008317518bd1a
conclusion	In the above example, -d ➔ derivation operation. -rk ➔ Input Key.68029E29E29FB77F6AE9F0A9D1F0EE0F -dd ➔ Derivation message.CBD963C6A8C4B0AC00000000000000006493F33FF06D625F9322F8A2AEA1B95B -lock -wrapen ➔ Key property for generating wrapping key Therefore the derived key (wrapping key is): 19E1C19F4D5A71EFCAC008317518BD1A

4.4.2 Generating an encryption and decryption key from an input key

Key property: `-lock -encen -decen`

Table 6. Generating an encryption and decryption key from an input key

Tool Options	Description
command	`> python cryptoWrapperReference.py -d -rk 68029E29E29FB77F6AE9F0A9D1F0 EE0F -dd CBD963C6A8C4B0AC00000000000000006493F33FF06D625F9322F8A2AEA1 B95B -lock -encen -decen`

output data	`Root key: 68029e29e29fb77f6ae9f0a9d1f0ee0f`
	`Derivation data: cbd963c6a8c4b0ac00000000000002606493f33ff06d6 25f9322f8a2aea1b95b`
	`Derived key: a64088e305114923c354343073cad48f`

conclusion	In the above example,
	`-d` ➔ derivation operation.
	`-rk` ➔ Input Key.`68029E29E29FB77F6AE9F0A9D1F0EE0F`
	`-dd` ➔ Derivation message.`CBD963C6A8C4B0AC00000000000000006493F33FF06D625 F9322F8A2AEA1B95B`
	`-lock -encen -decen` ➔ Key property for generating encryption/decryption key
	Therefore the derived key (encryption/decryption key is): `A64088E305114923C354343073 CAD48F`

4.5 Generating encrypted key data (wrapping) and providing data to demo app

By using the script, you can encrypt a key with a wrapping key and a wrapping IV (init vector). This encrypted data is provided to the secure key mode along with derivation message to generate the wrapping key, to unwrap and further store in the secure key store of PN7642.

Key properties used:

1. `-lock -secen -expen` (for storing into secure key storage)
2. `-lock -decen -encen` (for encrypting and decrypting operations)

Encrypted key data therefore generated from the script contains:

- `Ciphertext`, `Tag` and `IV`

You must provide the encrypted data in the following format:

- `<Ciphertext><Tag><IV` in reverse order of 4 bytes`>`

Table 7. Generating an encryption and decryption key and data format to be sent to SKM Demo app

Tool Options	Description
command	`> python cryptoWrapperReference.py -w -wk 19e1c19f4d5a71efcac008317518bd1a -k 46f3d11130d88c3c96f2f598fb9c0f51 -wiv 11111111222222223333333344444444 -lock -secen -expen`

output data	`Key to wrap:` **`46f3d11130d88c3c96f2f598fb9c0f510000000000000 0000000000000000000`**
	`Config: 00000000000000000000000000000341`
	`Wrapping key: 19e1c19f4d5a71efcac008317518bd1a`
	`CMAC key: e61e3e60b2a58e10353ff7ce8ae742e5`
	`Iv:` **`11111111222222223333333344444444`**
	`Ciphertext:` **`55fac73fd84c94356252e69059ade2e4948b715bd995b38b5259fd1c8d e6728f14fa9fedad0cba9f14ad93b047bc5123`**

Table 7. Generating an encryption and decryption key and data format to be sent to SKM Demo app*...continued*

Tool Options	Description
	Tag: **e59ac95098fcbc4392ee7be1cd9eb8ee**
conclusion	In the above example, -w ➔ Wrapping operation. -wk ➔ Wrapping Key.19e1c19f4d5a71efcac008317518bd1a -k ➔ ➔ New key to be wrapped.46f3d11130d88c3c96f2f598fb9c0f51 -wiv ➔ ➔ Wrapping Init VectorKey.111111112222222233333333444444444 -lock -secen -expen ➔ Key property of New key Therefore the wrapped data to be sent to DemoApp would be: <ciphertext><tag><iv in reverse order of 4 bytes> and that is **55fac73fd84c94356252e69059ade2e4948b715bd995b38b5259fd1c8de6728f14fa9fedad0cba9f14ad93b047bc5123e59ac95098fcbc4392ee7be1cd9eb8ee44444444333333332222222211111111**

4.6 Providing the derivation message, encrypted key data to Secure Key Mode commands

The host when sending the derivation message, encrypted key data to the PN7642 secure key mode commands, should be adhered to the following convention.

The data must be divided into 32-bit values and which are stored in the memory as little-endian format.

For e.g., the following input data (derivation message) to demo application provided as:

```
6493f33ff06d625f9322f8a2aea1b95bcbd963c6a8c4b0ac
```

To actual secure key mode command it shall be provided as below:

Table 8. Actual Secure Key Mode command

Location	Value	Value	Value	Value
Addr	0x3f	0xf3	0x93	0x64
Addr+4	0x5f	0x62	0x6d	0xf0
Addr+8	0xa2	0xf8	0x22	0x93
Addr+12	0x5b	0xb9	0xa1	0xae
Addr+16	0xc6	0x63	0xd9	0xcb
Addr+20	0xac	0xb0	0xc4	0xa8

4.7 Generating the authentication data

In order to open a session to work on the application keys, the host has to provide an encrypted data as a challenge. The user can encrypt a sample data of 16 bytes using ECB method and provide the encrypted data along with the expected data of 16 bytes.

The Secure Key Mode of PN7642 will generate an encryption/decryption key from the **NXP_TPT_KEY** or **APP_ROOT_KEY** (either 128-bit or 256-bit) from the provided derivation message and then decrypt the encrypted data to compare against the response (expected data).

Refer to 4.1 above and 4.4.2 above for more information on the derivation message and generating the encryption/decryption key.

Refer to (1) below for Secure Key Mode commands.

Following steps as listed in (1) below for generating the authentication data.

1. Send the **SKM_GET_SKM_INFO** command as depicted in (1) below
2. From the response of **SKM_GET_SKM_INFO**, note down the value of the **dwCounter** field.
3. If **APP_ROOT_KEY** is not provisioned, then **NXP_TPT_KEY** would be considered or **APP_ROOT_KEY** is considered as “input key” in 4.4.2 above.
4. Host can provide a derivation message. As from the **SKM_OPEN_SESSION**, the PN7642 considers only 20 bytes of 24 bytes of the derivation message and rest 4 bytes are the counter value what was provided as part of **SKM_GET_SKM_INFO** response.
5. So, if the **dwCounter** value is say **0x12345678**, then the derivation message would be following.

Derivation message from host is (20 bytes):

```
6493f33ff06d625f9322f8a2aea1b95bcbd963c612345678
```

dwCounter value as read from **SKM_GET_SKM_INFO** response: **0x12345678**

Resultant derivation message for generation of the encryption key is:

```
6493f33ff06d625f9322f8a2aea1b95bcbd963c612345678
```

So, the generated encryption key from input key would be as below:

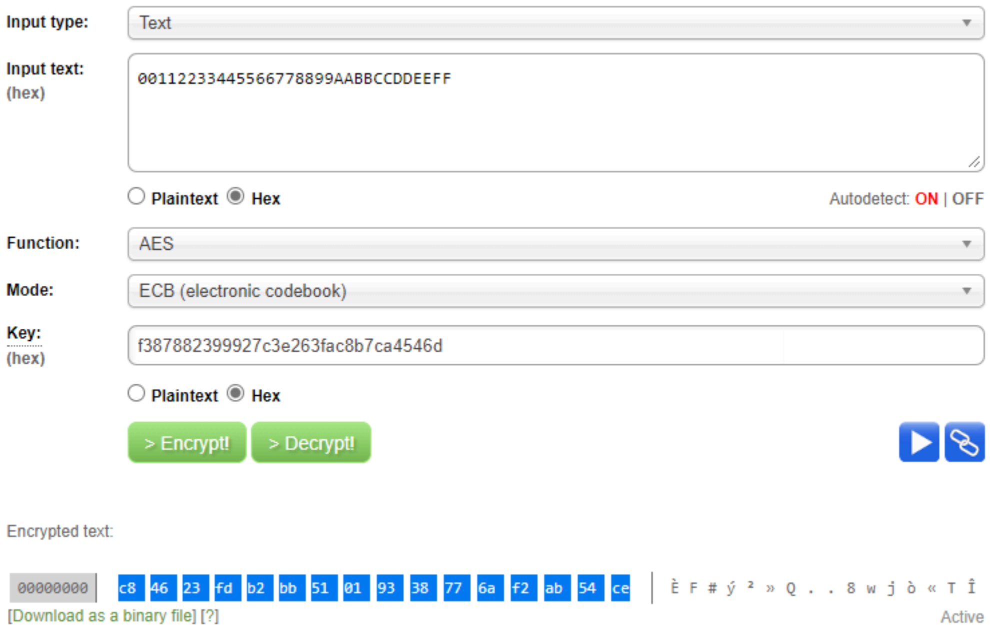
Table 9. Generated encryption key

Tool Options	Description
command	`> python cryptoWrapperReference.py -d -rk 68029e29e29fb77f6ae9f0a9d1f0ee0f -dd cbd963c6` **12345678** `0000000000000006493f33ff06d625f9322f8a2aea1b95b -lock -encen -decen`
output data	`Root key: 68029e29e29fb77f6ae9f0a9d1f0ee0f` `Derivation data:` **cbd963c6123456780000000000000002606493f33ff06d625f9322f8a2aea1b95b** `Derived key:` **f387882399927c3e263fac8b7ca4546d**
conclusion	In the above example, `-d` ➔ derivation operation. `-rk` ➔ Input key **19e1c19f4d5a71ef68029e29e29fb77f6ae9f0a9d1f0ee0fcac008317518bd1a** `-dd` ➔ Derivation message.`cbd963c6`**12345678** `0000000000000006493f33ff06d625f9322f8a2aea1b95b` `-lock -encen -decen` ➔ Key property of New key for encryption and decryption Therefore the generated encryption/decryption key would be: `f387882399927c3e263fac8b7ca4546d` And Derivation message to be provided as part of SKM_OPEN_SESSION command is: `6493f33ff06d625f9322f8a2aea1b95bcbd963c6`**12345678**

1. With therefore generated encryption/decryption key, user has to encrypt a sample data of 16 bytes.

Please refer to website: [AES Encryption – Easily encrypt or decrypt strings or files (online-domain-tools.com)]

You have to encrypt the sample data (16 bytes) using ECB method.



So for a sample data of **`00112233445566778899AABBCCDDEEFF`** (16 bytes), the encrypted data using the encryption/decryption key: **`f387882399927c3e263fac8b7ca4546d`** would be: **`c84623fdb2bb51019338776af2ab54ce`**

This encrypted data (**`c84623fdb2bb51019338776af2ab54ce`**) and expected data (**`00112233445566778899AABBCCDDEEFF`**) shall be provided as values to the **`SKM_OPEN_SESSION`** command.

4.8 Sample data for OpenSession commands with various keys such as NXP_TPT_KEYs and APP_ROOT_KEYs

Referring to Section 4.6 the authentication data for different counter values is generated as below:

4.8.1 Sample data for OpenSession command with NXP_TPT_KEY 128-bit

A sample of first 20 challenge generated with the different encryption/decryption key is as given below:

The challenge and response data can be generated by using the script provided in **`PN7642_SW_Extracted_Dir\Host_Software\Scripts`** directory.

Assumptions:

NXP Transport Key 128-bit: 4B3CEAED37CB6C03DB322BB483888474

DER_MSG_For EncryKey: 0123456709ABCDEF112233445566778899AABBCCDDEEFF00

Response Data for Challenge: 00112233445566778899AABBCCDDEEFF

Table 10. Sample data for SKM_OPEN_SESSION command with NXP_TPT_KEY_128

Counter	Encryption/Decryption Key	Challenge
00000000	8E8EA2584496465834EE1AC3BE3D11CB	B251BDBFFA64120BE5058AD658C7734D

Table 10. Sample data for SKM_OPEN_SESSION command with NXP_TPT_KEY_128*...continued*

Counter	Encryption/Decryption Key	Challenge
00000001	536CF577F105342DB7F45B43BAE9F9D9	FEE41205047E06638337EB548B9DE010
00000002	B1703816E7370373E1990463E4B7AB0C	1135A73AF14ABFB52E9E7E0BCC1811EA
00000003	79C454F284B73D006C4ECD1917F53BDF	5B3D4A6AFE18BBECEE4B798D6E0AA2A6
00000004	5B8CEAC13D067D28AD3E7872C63715BE	5B3B765357ED6B0560B267FBE0F4CEA2
00000005	31962FB8FC225A0089780D2DE8E456C5	9926FBB0938009B90611D0206C67E974
00000006	E1144B8293DB03318578A1F4DA921243	A07AE0DBE47E72B84F7E4939B7FA4873
00000007	CEB101899D73D7DA93945871878548E5	DB592828FF60CA1944F9B102E8B7C4A0
00000008	12E9C6A456657E3077DD94EAF1CEB5AB	791AE675E277DED25CFBF2A2C8EBD0C7
00000009	DED1433E63EE69F79D00CF95CE44D992	7C50CB6920DCC3218BAEFD54EFAEF3D9
0000000a	9BBEA980AFB63F167192860A989A0CA5	C323AA5EA65D591BB5D4ECDCB00F65DE
0000000b	08434159B8FC8A0DE766717930D529CF	910702A06E490D3E6D06983673CEAB06
0000000c	8664256ECCC1AD81916412E64DCAD2B4	517A657B61FB672FA22104479943B0C4
0000000d	F3F0CCD3067D54111E04C58E7B88C514	BC6A8A1389810A3C35B81E19CE479CF3
0000000e	9D29685615BB3A7108D892D48E3384BE	2FB2FEC8BE96F8404DF1F025BFAAC773
0000000f	1315C7864177BD9C293F38C987F5C7A3	CFB620FCB195017A947E76285FB2C8CD
00000010	E4EDA2C0D35BD9FE0E69B2329AD94A45	8373698A3BD54ABA6FBE3B3CC04C9B06
00000011	11075150887F65097B996C81DB5979EB	6ED4330D77923EFA381AFEE42D3FB3B4
00000012	DB88FE4DB71660483B26291CD6469AA3	5A38BE719B2C7C9AA47FE055F9CA79BC
00000013	E703F6DDF22F0B4406FBD893ECF33E78	A4E6015FD6BFC2F074F260C93F2FB279
00000014	AEC69C5398313B900A0A23C02698321A	4973D79171C4D6582218EDCFBE0993CB

4.8.2 Sample data for OpenSession command with NXP_TPT_KEY 256-bit

A sample of first 20 challenge generated with the different encryption/decryption key is as given below:

The challenge and response data can be generated by using the script provided in `PN7642_SW_Extracted_Dir\Host_Software\Scripts` directory.

Assumptions:

NXP Transport Key 256-bit: 7B986646E11B4DC55BBF1D35F2B00CACBA0AE0E822D70E89EAB95825BA843B82

DER_MSG_For EncryKey: 0123456709ABCDEF112233445566778899AABBCCDDEEFF00

Response Data for Challenge: 00112233445566778899AABBCCDDEEFF

Table 11. Sample data for SKM_OPEN_SESSION command with NXP_TPT_KEY_256

Counter	Encryption/Decryption Key	Challenge
00000000	CB440BB9EB0463258CE3A91301EE7A310BAEB6A53D75736F1143BCC5934FEECD	D59FAD0FF236CE669CF788DB422A6E5B
00000001	80ECF905FB14EBBEBEB47C60C52ED84CCF2BF801D68AAB090C78205438373EC3	C7364215825330C43089CC10E852127A

Table 11. Sample data for SKM_OPEN_SESSION command with NXP_TPT_KEY_256*...continued*

Counter	Encryption/Decryption Key	Challenge
00000002	E93260E87F2059CF282CB6A46F4C605CED7554 D44F1A2029200A329A65D5C2DC	743CDC8732609554085F3B012CFE4770
00000003	FCB40F77B33F90A5A517D8D791A30F712F3386 CAA2F84EBAE5E1B6AD4A7B59F4	47184455B461329A93D169323EE6A7C9
00000004	6C836D5FD8D8B097C31A730D4FA0701797F80F7 B7B3DE7C47EE31D1F1BC87664	B4BC7F524F9D90C984E085DFF11D47D1
00000005	03126E779BD0983FEE8322533CC1EA666E12CB C2B172BF494B1E0A1741DD226E	FF0032830AA5751271B19CB15C1F4DB5
00000006	36FBAB45543E958A12C3528B1FB9B7D54D590 B633D42D8B8707BEBF9B4498620	1A759B3278C4D0AACF1BE0DF7EDBEA37
00000007	091BFEA960DA7C1A4F54CB6D26B8684458010 BF6157470D806F1FABF67C5AEBD	EE5840BC45A938E51880389962A7CA84
00000008	B58768EA67FFD17401296A4F108AE3CDE510E0 51FDCB55C4C612BDC20108568C	0EEDA4AF641F4B2F9A958E40FA15AF32
00000009	23674A0B2EB53F8FFE5D9EE575204437DC5D49 F78135AB2819D197134F77DA3E	2C5AF4EDADCB2224AD6F0E104C0DC915
0000000a	B1DB0805CE22B0DAD4698A4714CB3B1246B97 E8E7383007907B3673AF15B79CD	FF28224AA827E115108E3713AAA5054C
0000000b	10783E9F18AA8C59DB33E3C5D36E486A5129 D860F42AE3DB484BF9301FDBD5D3	3423BFEB87B2DC041691813CC8808180
0000000c	D12D9B90D184E5807E0E155D2E357C00B2 E05382BF31E852411462E3E047061B	4C1EA461631CE045AE4F865280F239D5
0000000d	D68735DB297B11F379030A4DF8C7518D73A0 EAAD3379F12BC73B83E71A54CF76	6E8EDF2EF9652D335E23D0BAA1DCED82
0000000e	456780FED7DFDCC0F83640603B39E7F07AB7E D3121FC5413E379BB73E1957274	6AD3D0B65BB312176CCDF7EC98F431B5
0000000f	E1DCC0444EB24C4E3940D9CCE09D93749AF87 E80E5D9946A3CB180B1C91B85D9	13E78E4AA5936DE23A7C9CCD75F52035
00000010	3C1B6AFA7C33028B1711839DCCE732668EAB59 15F9A8804BF6113757F5B7000C	8C0D379D40F63773F26506905CB8E120
00000011	E0B9FC288A23DF9BCC5D1932AC2E0F3214BD6 C9E78063D7886BF25368F9A1D81	7FD5BC0B41205B261C1E9E73BD63E525
00000012	0A24245E2F2775D37FDCAC6161B008F80A59 DBF730664923D70712CBC75CEF29	83FF63F5F84A0DCFEDCE0ACD96F45763
00000013	27E22F3E21B10970EE620C7CDC41B7D54A5458 BCE28B60DEE86CB18D4853481D	39ECCB938FAE82B96E5D0519CA745622
00000014	9AAFA5ED5BFA0BB24569199FF9786B133AFF94 14C9CD2E2FDC1E0820261D7A67	9E76285F34285B639D152EC0E2BED5DA

4.8.3 Sample data for OpenSession command with APP_ROOT_KEY 128-bit

A sample of first 20 challenge generated with the different encryption/decryption key is as given below:

The challenge and response data can be generated by using the script provided in `PN7642_SW_Extracted_Dir\Host_Software\Scripts` directory.

Assumptions:

APP_ROOT_KEY_128: 46F3D11130D88C3C96F2F598FB9C0F51

DER_MSG_For EncryKey: 0123456709ABCDEF112233445566778899AABBCCDDEEFF00

Response Data for Challenge: 00112233445566778899AABBCCDDEEFF

Table 12. Sample data for SKM_OPEN_SESSION command with a sample APP_ROOT_KEY_128

Counter	Encryption/Decryption Key	Challenge
00000000	8F2E19A6D68820CDD97CE78344868BC9	18658BD08FDB2A65B9F0FB0F04DEAD03
00000001	E119329B8F0EF53D4DAA3E026E7911AA	AB5276F104FC5648EB8072F419C4A9E9
00000002	BD61924AD9A7D1F793C58006E3EBCBEC	CB17EC77ECC8ABD6E36FFFACA6442C9A
00000003	72536927B43ADC22159394D01EB85084	676F07EA105760FE249C5E06D35AAC20
00000004	3C4184B44C951E27E08DC9690CA157BE	A603F33EA75B14984E002AAD2F45E654
00000005	F5F9BFB39C58C3D3DE0C1EC9071BFEF0	ED0DA3F0C13C001837B18551C3F56788
00000006	9541486E3E96B0CE72DC796E10182DE0	8F173C78214FFB2859ED5D7011DEC9EC
00000007	2FAC47564AC6D3B193F0C82A3003BFD1	331F260518B6E10C13B5BFA4D750826C
00000008	924C21576CC04B7BCD34516E9F556AFE	5E91FA11384460A599F579569DB62FF2
00000009	B0E21F580BEC7D577DB1BEDFE2765964	7B42B98021DE23007F866FC0E944FD4E
0000000a	685DF802E47315F852E7ACE5D4728B56	DDA5B5E81352AB930B93E790C195F6F2
0000000b	0305A16D6A99B914A0FC3F9EBC081899	E05B3D448EF793D798508BEF4C9DDC4C
0000000c	137D80B49DD9882729209896254D5EEA	638F0FC470CDCECE3C15AC4EB7114A41
0000000d	3B730C98CD16E797176E1068E14BBBA0	2F5F1149E61C89378F1D23C7A9143A00
0000000e	961854ACEE8D4121D3121DD8A0B8F5CD	F650194FCD32E3B3C7AB8620FF5739F6
0000000f	BC68120A26EB0620A320FA22FF407A83	0E225E0BE2F77F99C34110DE69A1FDAF
00000010	A3604369919DCB76D36414A070CE3105	07635336259850E174FE5BBE9BFE80CD
00000011	0F09E933A51188EF54B18157FD2420FB	3A155E56021D43DBB1F8F77E10DF8174
00000012	8940EFC6338C3C3EBFF1099F0AE49DE2	45A4083BD8F6DADA46844C95C38D0C00
00000013	187C3AD2EF5F3427B4D7CD982FDE776C	5C5BA70F5BE6F6E44DD3C3A2D265A960
00000014	9CA0871BD0A0A273CDEBE6974464E8CA	49567AB5E305EE009065833B8F14F514

4.8.4 Sample data for OpenSession command with APP_ROOT_KEY 256-bit

A sample of first 20 challenge generated with the different encryption/decryption key is as given below:

The challenge and response data can be generated by using the script provided in `PN7642_SW_Extracted_Dir\Host_Software\Scripts` directory.

Assumptions:

APP_ROOT_KEY_256: 207d74cf3eed13ae1373d61e134592f226ae1112590461623cf76eb27ef9b55c

DER_MSG_For EncryKey: 0123456709ABCDEF112233445566778899AABBCCDDEEFF00

Response Data for Challenge: 00112233445566778899AABBCCDDEEFF

Table 13. Sample data for SKM_OPEN_SESSION command with a sample APP_ROOT_KEY_256

Counter	Encryption/Decryption Key	Challenge
00000000	B6B1E585850807DD1D107816081F4BE9E9EA3 4879E6DDF67C149875DDF4ABE5B	3513940D6F11B67B30FBB6FD6DC20000
00000001	D003A897077E0D2F98848DBDB3D435F0D61F C87631445F370709E07CF898E250	F80145DD2B5EA8686FB113954D67CDB8
00000002	BFA016ADEF3AA11D4EFD6BFF2CFBC1A7BA49 E3336958862174CE4F83F33E966F	DA3B5D4485070686EB76AFA37D9D540F
00000003	380637A0B39664705DA9119CD19F259265B9002 D7F7C6233502CB817F7221E10	0E243C12D1BBD342B7BD4684B5C6A054
00000004	4C24A468288E79A7AA8B9AA5105DBB7740BB5A7 C8766A3217166ACB848745195	1C90426691B05ED92F2BCFAAA2C81079
00000005	7E54A6C5A9F87A2C5786F250582F19F6A288CF6 EE5A781726CE500A6D54577C7	8380C7842072ED9D2FB919761A843266
00000006	48E49193C39E1709FB386BB17E754425CA0335 BB7D96E4CD98046E44C4BC908F	1918825A5D5CBCD7B59F246163187919
00000007	FDC586B63B495E0007DE93140C9D8E3B6BACD0 F1E0555F3B6ADD52D1CCFA1DF5	370A77B306BE07283385F7F2E02C9CCF
00000008	EB8361BBC9F31A15024ECD94E38482A0A4989C0 BC9CAC97E3E46846568B161B2	4A70EBA0E4378B90E376026526513D6C
00000009	03B034A0A8FBE890FF050C3EEEE8D7F4871 CCD3F3CCD59018AE3D6DEC5BBD177	4ECF6577601CD7834BCCCAC1C7A2CE31
0000000a	1662A684D88D1A2FBE1CEE4DFFFFAAA8C70C7 F284E7DB6F079E2426BEC33B185	CC87F494E27A4B51B25B210FBFAE585C
0000000b	B2158BFBE40EF7FA5BACA2BD77B56AB063DA2 C155C12F3D6D4E590BA2A6C8FEB	09B5060D954AF914654A71EF39DBBA78
0000000c	6977115E8B0618CBC1CCC4E3CA195333CCE745 7CDDA864A1BEC25E87C1543486	26B25831B6DC1A8DE642C4FA2E8871D2
0000000d	C5E37A2CDCCD93ED69E8CA1599CE3C446261 F5F3C565FDEE4E2487F8364C12F1	2F933F4800457EDFF246D9B0D00BDC8B
0000000e	6FFC3D7E7B959901EB2CF8459F3DE79ABD92 182EC3F8140F4A0AF8B1ED03CED2	709512B6642C458233C0070CF3AE97C1
0000000f	0FC1A31020F67DB87D9B083F24C7E7296AB 74027DD7EB686A633DC7D284CBA72	93C3182F6DA44959B7DDFFB7CD447D98
00000010	1E694DC3A0172505135BD4E3228F44EB8D85B12 A1528B712567D54F1C9BAC68B	1CEDD5901EE4D1BE44FC0965AFBA482C
00000011	650CDB74EE279471D110746703ABF6 CDC370610272CBE3463DF73481EDEA95C2	7A70F7F15ED7B351A4774D36DE4C827C
00000012	143075B3EAB3C41FDE8DB43E37C846D38B5 BCFB576C0D7538742329F498E17F5	CD2D64C80171B373FA2046D29E801137
00000013	14F8679E2C542C0C8BD72B737C10287EC30 CCE52908F2CCC650A859CC58F2A00	F52EBF9F771156C91EA48DAF5B313CF5
00000014	9209C5200B42EA3E0183ED4EF5424E10B7F4 239536466C190931BAE4150E88CF	09023E89A13896E08CFCA584ACF3CA0D

5 Working with Secure Key Mode demo application

After the secure key mode demo application is built and downloaded onto the target board, and executes the following menu will appear in the “Console” tab.

Interaction in console window:

Figure 4. Initial Main menu on Console window of MCUXpresso IDE

```
******** Secure Key Mode ********
Select the Option
- Enter 00 for Entering into Secure Key Mode (SKM).
- Enter 01 for Get DieID.
- Enter 02 for Open Session.
- Enter 03 for Provisioning APP_ROOT_KEY.
- Enter 04 for Provisioning APP_MASTER_KEY.
- Enter 05 for Provisioning APP_FIXED_KEY.
- Enter 06 for Deleting APP_MASTER_KEY.
- Enter 07 for Deleting APP_FIXED_KEY.
- Enter 08 for Updating APP_MASTER_KEY.
- Enter 09 for Updating APP_FIXED_KEY.
- Enter 10 for Locking Provisioning of APP_ROOT_KEY.
- Enter 11 for Getting the SKM State.
- Enter 12 for Provisioning a APP_ASYMM_KEY auto generated.
- Enter 13 for Provisioning a APP_ASYMM_KEY provided as plain.
- Enter 14 for Provisioning a APP_ASYMM_KEY provided as encrypted
- Enter 15 for Deleting a APP_ASYMM_KEY.
- Enter 16 for Setting the domain parameters for the CUSTOM ASYMM_KEY.
- Enter 17 for Getting the Public key of an APP_ASYMM_KEY.
- Enter 18 for Purging the APP ROOT KEY.
- Enter 19 to SKM RESET.
- Enter 20 to perform SOFT RESET.
Select Option (Hit Enter after Input) :
```

Note: To exercise options `1 - 19`, you should be in the Secure Key Mode. To enter into Secure Key Mode, select option 0.

Note: If you perform SOFT RESET (selecting option 20) or after downloading the application, then then you must enter Secure Key Mode to exercise other options.

Note: Whenever you provide inputs to the options or parameters, make sure that there should not be any space present in the input. If there is any space present in the input, restart the application to work with.

5.1 Entering into the Secure Key Mode of PN7642

Select the option `00` to enter into the Secure Key Mode. Sample contents would be as below (check the status of the command execution as highlighted):

Interaction in console window:

Figure 5. Entering Secure Key Mode

```
Select Option (Hit Enter after Input) : 00
Option 0 selected
SKM_ENTERMODE : SUCCESS
```

5.2 Getting the Die-ID of the PN7642 IC

Select the option “1” to enter into the Secure Key Mode. Sample contents would be as below (check the status of the command execution as highlighted):

Interaction in console window:

Figure 6. Getting Die-ID of the IC

```
Select Option (Hit Enter after Input) : 01
Option 1 selected
SKM_GET_DIE_ID :
SUCCESS
DieID: 00000000000000000000000000000000
```

5.3 Open a session for working on different keys

Select the option **`02`**” to open a session for provisioning the APP_ROOT_KEY.

Let us assume that APP_ROOT_KEY was not provisioned earlier.

So, the NXP_TPT_KEY is present in the PN7642 IC.

We first authenticate using the NXP_TPT_KEY_128-bit.

Assuming that: NXP_TPT_KEY_128-bit: **`4B3CEAED37CB6C03DB322BB483888474`**

Assuming that dwCounter = `0x00000000`,

Refer to Section 4.8.4 for the derivation message, corresponding challenge, and response data for authentication using the various keys (NXP_TPT_KEY (128/256), APP_ROOT_KEY (128/256)).

Interaction in console window:

Figure 7. Opening a session to provision APP_ROOT_KEY with NXP_TPT_KEY_128

```
Select Option (Hit Enter after Input) : 02
Option 2 selected
SKM_GET_SKM_STATE : SUCCESS
==========================================
APP_ROOT_KEY_SESSION_STATUS : CLOSED
```

```
APP_MASTER_KEY/APP_FIXED_KEY Session Status : CLOSED
APP_ASYMM_KEY_SESSION_STATUS : CLOSED
SKM_LOCK_STATUS : NOT LOCKED
SKM Intigrity check STATUS : 0C
SKM Asymmetric key domain params status : Valid.
CurveType Set is: SECP384R1
COUNTER_FOR_AUTH : 00000000
==========================================
For working on which type of key (00 -> APP_ROOT_KEY, 01 -> MASTER/FIXED_KEY, 02 -
> ASYMM_KEY 03-> PURGE_KEY) : 00
Authenticating with which key length (00 -> 128-bit, 01 -> 256-bit) : 00
Derivation message for encryption/decryption Key (24 bytes): 0123456709
ABCDEF112233445566778899AABBCCDDEEFF00
Encrypted data: (16bytes) B251BDBFFA64120BE5058AD658C7734D
Expected Decryption data: (16 bytes) : 00112233445566778899AABBCCDDEEFF
SKM_OPENSESSION : SUCCESS
```

5.4 Provisioning APP_ROOT_KEY

First let us generate the encrypted key data required for provisioning the APP_ROOT_KEY_128-bit.

Refer to Generating a wrapping key from an input key and Generating encrypted key data (wrapping) and providing data to demo app

Assuming that: NXP_TPT_KEY_128-bit: **4B3CEAED37CB6C03DB322BB483888474**

Assuming that: APP_ROOT_KEY_128-bit: **46F3D11130D88C3C96F2F598FB9C0F51**

Assuming that the derivation message to generate a encryption/decryption key from APP_ROOT_KEY_128 is:

0123456709ABCDEF112233445566778899AABBCCDDEEFF00

Table 14. Generating encrypted APP_ROOT_KEY_128 bit data

Tool Options	Description
command to generate wrapping key	`> python cryptoWrapperReference.py -d -rk 4B3CEAED37CB6C03DB322BB483888474 -dd 99AABBCCDDEEFF000000000000000000000123456709ABCDEF1122334455667788 -wrapen -lock`

output data: wrapping key	`Root key: 4b3ceaed37cb6c03db322bb483888474`
	`Derivation data:` **99aabbccddeeff00000000000000002100123456709abcdef1122334455667788**
	`Derived key: c241b6e2347e727ef871827e8419ba6a`
	`This generated wrapping key:` **c241b6e2347e727ef871827e8419ba6a**

command to generate encrypted key data	`> python cryptoWrapperReference.py -w -wk c241b6e2347e727ef871827e8419ba6a -k 46f3d11130d88c3c96f2f598fb9c0f51 -wiv 11111111222222223333333344444444 -lock -secen -expen`
output data: wrapped key data	`Key to wrap:` **46f3d11130d88c3c96f2f598fb9c0f51000000000000000000000000000000000**

Table 14. Generating encrypted APP_ROOT_KEY_128 bit data*...continued*

Tool Options	Description

	`Config: 00000000000000000000000000000341`
	`Wrapping key: c241b6e2347e727ef871827e8419ba6a`
	`CMAC key: 3dbe491dcb818d81078e7d817be64595`
	`Iv: 11111111222222223333333344444444`
	`Ciphertext:` **`180cd359f8dcb492ee48fd431ea239b026d002601b60c7b9507628b36f 546c79bca44740792355e31a48c62fbc07bc13`**
	`Tag: f7784c0251e2c0b520d529df67d2c636`

Data to be provided to application	**`180CD359F8DCB492EE48FD431EA239B026D002601B60C7B9507628B36F546C79BCA447 40792355E31A48C62FBC07BC13F7784C0251E2C0B520D529DF67D2C636444444443333 3333222222221111111`**

First open the session to work on APP_ROOT_KEY by referring to Section 5.3.

Interaction in console window:

Figure 8. Provisioning a APP_ROOT_KEY_128-bit

```
Select Option (Hit Enter after Input) : 03

Option 3 selected

Authenticating with which key length (00 -> 128-bit, 01 -> 256-bit) : 00

Option 0 selected

Derivation message for wrapping Key (24 bytes): 0123456709ABCDEF1122334455667788 99
AABBCCDDEEFF00

Encrypted key data: (80bytes)180cd359f8dcb492ee48fd431ea239b026d002601b60c7b950762
8b36f546c79bca44740792355e31a48c62fbc07bc13f7784c0251e2c0b520d529df67d2c6364444444
43333333322222222111111111

SKM_PROV_APP_ROOT_KEY :SUCCESS
```

5.5 Provisioning APP_MASTER_KEY

First open the session to work on APP_MASTER_KEY/APP_FIXED_KEY by referring to Section 5.3.

APP_MASTER_KEY is derived from either NXP_TPT_KEY or APP_ROOT_KEY.

Assuming that APP_ROOT_KEY (**`46F3D11130D88C3C96F2F598FB9C0F51)`** is already provisioned earlier, a new APP_MASTER_KEY is derived with derivation message as: 000000000000000000000000000000000000000000000

Where the derived master key is: `903b6bc7bbc909ca4130f1ddf356f161`

Table 15. Generating derived key from APP_ROOT_KEY_128-bit for APP_MASTER_KEY

Tool Options	Description
command to derive a key	`> python cryptoWrapperReference.py -d -rk 46F3D11130D88C3C96F2F598FB9 C0F51 -dd 0000000000000000000000000000000000000000000000000000 0000 -lock -expen`

output data: derived key	`Root key: 46f3d11130d88c3c96f2f598fb9c0f51`
	`Derivation data:` **`00000000000000000000000000000204000000000000000000000 00000000000`**
	`Derived key: 903b6bc7bbc909ca4130f1ddf356f161`

Table 15. Generating derived key from APP_ROOT_KEY_128-bit for APP_MASTER_KEY*...continued*

Tool Options	Description
	Therefore generated APP_MASTER_KEY is: `903b6bc7bbc909ca4130f1ddf356f161`
Conclusion: Data to be provided to application	`000000000000000000000000000000000000000000000000000`

Interaction in console window:

Figure 9. Provisioning a APP_MASTER_KEY_128-bit

```
Select Option (Hit Enter after Input) : 04
Option 4 selected
KeyIndex where Master Key to be stored (01, 04 - 15) : 01
Master key length (00 -> 128-bit, 01 -> 256-bit) : 00
Derivation message for Master key from APP_ROOT_KEY Key (24 bytes): 00000000000000
0000000000000000000000000000000000
SKM_PROV_APP_MASTER_KEY :SUCCESS
```

5.6 Provisioning APP_FIXED_KEY

First let us generate the encrypted key data required for provisioning the APP_ROOT_KEY_128-bit.

Refer to Generating a wrapping key from an input key and Generating encrypted key data (wrapping) and providing data to demo app.

Assuming that: APP_ROOT_KEY_128-bit: **46F3D11130D88C3C96F2F598FB9C0F51**

Assuming that the derivation message to derive a wrapping key from APP_ROOT_KEY_128 is:

0123456709ABCDEF112233445566778899AABBCCDDEEFF00

Assuming that the APP_FIXED_KEY_128-bit with key property for encryption/decryption is: **7EDA2BD5D7093F353EE6D2993E4BA348**

Table 16. Generating encrypted APP_FIXED_KEY_128 bit data

Tool Options	Description
command to generate APP_FIXED_KEY data	`> python cryptoWrapperReference.py -d -rk 46F3D11130D88C3C96 F2F598FB9C0F51 -dd 99AABBCCDDEEFF00000000000000000000123456709A BCDEF1122334455667788 -wrapen -lock`

output data: wrapping key	`Root key: 46f3d11130d88c3c96f2f598fb9c0f51`
	`Derivation data:` **`99aabbccddeeff00000000000000002100123456709abc def1122334455667788`**
	`Derived key: 9001fe02b1a18f6f7ab8902e313dfede`
	`This generated wrapping key:` **`9001fe02b1a18f6f7ab8902e313dfede`**

command to generate encrypted key data	`> python cryptoWrapperReference.py -w -wk 9001FE02B1A18F6F7AB8902E313 DFEDE -k 7EDA2BD5D7093F353EE6D2993E4BA348 -wiv 11111111222222223333333 344444444 -lock -encen -decen`

output data: wrapped key data	`Key to wrap: 7eda2bd5d7093f353ee6d2993e4ba3480000000000000000 0000000000000000`
	`Config: 000000000000000000000000000014d`
	`Wrapping key: 9001fe02b1a18f6f7ab8902e313dfede`

Table 16. Generating encrypted APP_FIXED_KEY_128 bit data*...continued*

Tool Options	Description

	`CMAC key: 6ffe01fd4e5e709085476fd1cec20121`
	`Iv: 11111111222222223333333344444444`
	`Ciphertext:` **`75d8ed84b2c1910e8ee6ea15b9ff48c89acadaec336566d07209639d53 bac7bf9ec907e7dbfe19183b4c4d5b22bed4e3`**
	`Tag: e0ef6539a8470018075dbb6fbf1df631`

Data to be provided to application	**`75D8ED84B2C1910E8EE6EA15B9FF48C89ACADAEC336566D07209639D53BAC7BF9EC907 E7DBFE19183B4C4D5B22BED4E3E0EF6539A8470018075DBB6FBF1DF63144444444443333 33332222222211111111`**

First open the session to work on APP_FIXED_KEY by referring to Section 5.3.

Interaction in console window:

Figure 10. Provisioning a APP_FIXED_KEY_128-bit

```
Select Option (Hit Enter after Input) : 05

Option 5 selected

KeyIndex where Fixed Key to be stored (16-26) : 16

Fixed key length (00 -> 128-bit, 01 -> 256-bit) : 00

Derivation message for wrapping Key (24 bytes): 0123456709ABCDEF112233445566778899
AABBCCDDEEFF00

Encrypted key data: 75d8ed84b2c1910e8ee6ea15b9ff48c89acadaec336566d07209639d53bac7
bf9ec907e7dbfe19183b4c4d5b22bed4e3e0ef6539a8470018075dbb6fbf1df6314444444443333333
2222222211111111 (80bytes)

SKM_PROV_APP_FIXED_KEY :SUCCESS
```

5.7 Updating an APP_MASTER_KEY

First open the session to work on APP_MASTER_KEY/APP_FIXED_KEY by referring to Section 5.3.

APP_MASTER_KEY is derived from APP_ROOT_KEY.

Assuming that APP_ROOT_KEY (**`46F3D11130D88C3C96F2F598FB9C0F51`**) is already provisioned earlier, a new APP_MASTER_KEY is derived with derivation message as: 0123456709ABCDEF112233445566778899 AABBCCDDEEFF00

Table 17. Generating a APP_MASTER_KEY from APP_ROOT_KEY_128-bit

Tool Options	Description
command to derive a key	`> python cryptoWrapperReference.py -d -rk 46F3D11130D88C3C96 F2F598FB9C0F51 -dd 99AABBCCDDEEFF000000000000000000000123456709A BCDEF1122334455667788 -lock -expen`

output data: derived key	`Root key: 46f3d11130d88c3c96f2f598fb9c0f51`
	`Derivation data: 99aabbccddeeff00000000000000002000123456709abc def1122334455667788`
	`Derived key: 36c39408d5d43e69533a87ff0d1886f8`
	Therefore generated APP_MASTER_KEY is :`36c39408d5d43e69533a87ff0d1886f8`

ConclusionData to be provided to application	`0123456709ABCDEF112233445566778899AABBCCDDEEFF00`

Interaction in console window:

Figure 11. Updating a APP_MASTER_KEY_128-bit

```
Select Option (Hit Enter after Input) : 08

Option 8 selected

KeyIndex where existing Master Key to be updated (01, 04 - 15) : 01

Derivation message for Master key from APP_ROOT_KEY Key (24 bytes): 0123456709
ABCDEF112233445566778899AABBCCDDEEFF00

SKM_UPDATE_APP_MASTER_KEY : SUCCESS
```

5.8 Updating an APP_FIXED_KEY

First let us generate the encrypted key data required for provisioning the APP_ROOT_KEY_128-bit.

Refer to Generating a wrapping key from an input key and Generating encrypted key data (wrapping) and providing data to demo app.

Assuming that: APP_ROOT_KEY_128-bit: **46F3D11130D88C3C96F2F598FB9C0F51**

Assuming that the derivation message to derive a wrapping key from APP_ROOT_KEY_128 is:

0123456709ABCDEF112233445566778899AABBCCDDEEFF00

Assuming that the APP_FIXED_KEY_128-bit with key property for encryption/decryption is: **F3EE6343C2F17E724B8E9C55F447E88F**

Table 18. Generating encrypted APP_FIXED_KEY_128 bit data

Tool Options	Description
command to generate APP_FIXED_KEY data	`> python cryptoWrapperReference.py -d -rk 46F3D11130D88C3C96F2F598FB9C0F51 -dd 99AABBCCDDEEFF0000000000000000000123456709ABCDEF1122334455667788 -wrapen -lock`

output data: wrapping key	`Root key: 46f3d11130d88c3c96f2f598fb9c0f51`
	`Derivation data: 99aabbccddeeff000000000000002100123456709abcdef1122334455667788`
	`Derived key: 9001fe02b1a18f6f7ab8902e313dfede`
	`This generated wrapping key:` **`9001fe02b1a18f6f7ab8902e313dfede`**

command to generate encrypted key data	`> python cryptoWrapperReference.py -w -wk 9001FE02B1A18F6F7AB8902E313DFEDE -k F3EE6343C2F17E724B8E9C55F447E88F -wiv 111111112222222233333333344444444 -lock -encen -decen`

output data: wrapped key data	`Key to wrap: f3ee6343c2f17e724b8e9c55f447e88f00000000000000000000000000000000`
	`Config: 0000000000000000000000000000014d`
	`Wrapping key: 9001fe02b1a18f6f7ab8902e313dfede`
	`CMAC key: 6ffe01fd4e5e709085476fd1cec20121`
	`Iv: 11111111222222223333333344444444`
	`Ciphertext:` **`00573f3e822c94ee2f998e6a04b057e9e097b914ab75e81cfc23c79404216e3c2f55d2843f85cf53eeda6d9cb86415e8`**
	`Tag: 770dd6f3fe124654e1264e9653da613a`

Data to be provided to application	**`00573F3E822C94EE2F998E6A04B057E9E097B914AB75E81CFC23C79404216E3C2F55D2843F85CF53EEDA6D9CB86415E8770DD6F3FE124654E1264E9653DA613A4444444433333333222222221111111`**

First open the session to work on APP_FIXED_KEY by referring to Section 5.3.

Interaction in console window:

Figure 12. Updating APP_FIXED_KEY_128-bit

```
Select Option (Hit Enter after Input) : 09

Option 9 selected

KeyIndex where existing Fixed Key to be updated (16-26) : 16

Derivation message for wrapping Key (24 bytes): 0123456709ABCDEF112233445566778899
AABBCCDDEEFF00

Encrypted key data: (80bytes) 00573F3E822C94EE2F998E6A04B057E9E097B914AB75E81CFC23
C79404216E3C2F55D2843F85CF53EEDA6D9CB86415E8770DD6F3FE124654E1264E9653DA613A444444
44333333332222222211111111

SKM_UPDATE_APP_FIXED_KEY :SUCCESS
```

5.9 Deleting a APP_MASTER_KEY

First open the session to work on APP_MASTER_KEY/APP_FIXED_KEY by referring to Section 5.3.

Assuming that APP_MASTER_KEY is already provisioned at KeyId 1. This KeyId can be deleted.

Interaction in console window:

Figure 13. Deleting APP_MASTER_KEY_128-bit

```
Select Option (Hit Enter after Input) : 06

Option 6 selected

KeyIndex of Master Key to be deleted (01, 04 - 15) : 01

SKM_DELETE_APP_MASTER_KEY :SUCCESS
```

5.10 Deleting a APP_FIXED_KEY

First open the session to work on APP_MASTER_KEY/APP_FIXED_KEY by referring to Section 5.3.

Now assume that APP_FIXED_KEY is already provisioned at KeyId: 16, let us delete that key.

Interaction in console window:

Figure 14. Deleting APP_FIXED_KEY_128-bit

```
Select Option (Hit Enter after Input) : 07

Option 7 selected

KeyIndex of Fixed Key to be deleted (16-26) : 16

SKM_DELETE_APP_FIXED_KEY :SUCCESS
```

5.11 Locking further provisioning of APP_ROOT_KEY (either 128/256 bit)

First open the session to work on APP_ROOT_KEY by referring to Section 5.3.

Now assume that APP_ROOT_KEY is already provisioned, let us lock the APP_ROOT_KEY from further provisioning.

Interaction in console window:

Figure 15. Locking APP_ROOT_KEY_128-bit

```
Select Option (Hit Enter after Input) : 10

Option 10 selected

Which APP_ROOT_KEY of key length need to be locked from further provisioning (00 -
> 128-bit, 01 -> 256-bit) : 00

Option 0 selected

SKM_LOCK_APP_ROOT_KEY : SUCCESS
```

5.12 Getting the SKM information

Interaction in console window:

Figure 16. Getting SKM Information

```
Select Option (Hit Enter after Input) : 11

Option 11 selected

SKM_GET_SKM_STATE : SUCCESS

==========================================

APP_ROOT_KEY_SESSION_STATUS : CLOSED

APP_MASTER_KEY/APP_FIXED_KEY Session Status : CLOSED

APP_ASYMM_KEY_SESSION_STATUS : CLOSED

SKM_LOCK_STATUS : NOT LOCKED

SKM Intigrity check STATUS : 0C

SKM Asymmetric key domain params status : Valid.

CurveType Set is: SECP384R1

COUNTER_FOR_AUTH : 0000000B

==========================================
```

5.13 Provisioning the APP_ASYMM_KEY which is auto-generated inside PN7642

First open the session to work on APP_ASYMM_KEY by referring to Section 5.3.

Then set up the domain parameters to work with APP_ASYMM_KEY.

Interaction in console window:

Figure 17. Provisioning auto-generated APP_ASYMM_KEY

```
Select Option (Hit Enter after Input) : 12

Option 12 selected

KeyIndex where Generated Asymm Key to be provisioned (27-33) : 27
```

```
Key Properties (01 -> SECP256R1, 02 -> SECP384R1, 03 -> BP256R1, 04 ->
BP384R1, 05 -> CUSTOM_DP_256, 69 -> CUSTOM_DP_384, 06 -> EDDSA_256, 07 ->
EDDSA_MONTDH25519_256) : 01

Return PUB key associated with this PVT key (00 -> No, 01 -> Yes) : 01

SKM_PROV_APP_ASYMM_KEY_GEN :SUCCESS

======PUB_KEY_DATA========================

Key Id : 1B

Key Prop : 01

Public Key :  6A14D8F8FF449305106C3A80E1AA6DAFA8DBD984782163293CBE3F2DF3B839200B1E
AB08CDFCD07714B21D15E228DB1E7BF5F4151B39A52FE5BB3D074742EF3E

==========================================
```

5.14 Provisioning the APP_ASYMM_KEY which is provided in plain format

First open the session to work on APP_ASYMM_KEY by referring to Section 5.3.

Then set up the domain parameters to work with APP_ASYMM_KEY by referring to 5.13 above (If custom curve is to be used).

Interaction in console window:

```
Select Option (Hit Enter after Input) : 13
```

Figure 18. Provisioning APP_ASYMM_KEY provided in plain format

```
Select Option (Hit Enter after Input) : 13

Option 13 selected

KeyIndex where plain Asymm Key to be provisioned (27-33) : 28

Key Properties (01 -> SECP256R1, 02 -> SECP384R1, 03 -> BP256R1, 04 -> BP384R1, 05
-> CUSTOM_DP_256, 69 -> CUSTOM_DP_384, 06 -> EDDSA_256) : 01

Return PUB key associated with this PVT key (00 -> No, 01 -> Yes) : 01

Plain pvt key data (32/48 bytes) : 390BA5453390DF090658776AABA12E867787CC6162766
A70B78CBD016B74B0E2

SKM_PROV_APP_ASYMM_KEY_PLAIN :SUCCESS

======PUB_KEY_DATA========================

Key Id : 1C

Key Prop : 01

Public Key : 57C4C08265E9839C37EADB131835107712FBFF5EAC22112CA84492E523F16514E2023
5CB725676C3D03C77E406C55436BB2184D925F5D1FD48DE551CF8162687

==========================================
```

5.15 Provisioning the APP_ASYMM_KEY which is provided in encrypted format

First open the session to work on APP_ASYMM_KEY by referring to Section 5.3.

Then set up the domain parameters to work with APP_ASYMM_KEY by referring to 5.13 above (If custom curve is to be used).

For the encryption of the asymmetric key in a generic way, refer to Section 4.7.

The Secure Key Mode in PN7642 IC, uses the APP_ROOT_KEY_128 to generate a encryption/decryption key internally, and decrypt the encrypted APP_ASYMM_KEY and then stores in the extended key store area.

Let us generate the encrypted APP_ASYMM_KEY.

Assuming that: APP_ROOT_KEY_128-bit: **46F3D11130D88C3C96F2F598FB9C0F51**

Assuming that the derivation message to generate a encryption/decryption key from APP_ROOT_KEY_128 is:

886DD0D58CDF9346ABE32A2B80E8EBFE554B9EF9544B2338.

Note that this derivation message must be stored in EEPROM area at address: 0x568, with E_PN76_EEPROM_SECURE_LIB_CONFIG in PN76_WriteEeprom() System service API.

Assuming that the APP_ASYMM_KEY for curve SECP256R1 is: **390BA5453390DF090658776AABA12 E867787CC6162766A70B78CBD016B74B0E2**

And generated PUBLIC_KEY part would be: **57C4C08265E9839C37EADB131835107712FBFF5EAC22112 CA84492E523F16514E20235CB725676C3D03C77E406C55436BB2184D925F5D1FD48DE551CF8162687**

Table 19. Generating a encrypted APP_ASYMM_KEY data to provision

Tool Options	Description
To generate Encryption/ Decryption key from APP_ROOT_KEY	`> python cryptoWrapperReference.py -d -rk 46F3D11130D88C3C96F2F598FB9 C0F51 -dd 554B9EF9544B2338000000000000000000886DD0D58CDF9346ABE32A2B80E8 EBFE -lock -encen -decen`

Encryption/Decryption key	`Root key: 46f3d11130d88c3c96f2f598fb9c0f51`
	`Derivation data: 554b9ef9544b233800000000000000260886dd0d58cdf9 346abe32a2b80e8ebfe`
	`Derived key: b5dbe7fbdf2a31ebc3b9f90300a61b3c`
	`Thus generated encyption/decryption key:` **B5DBE7FBDF2A31EBC3B9F90300A61 B3C**

Encrypted APP_ ASYMM_KEY	Using the above encryption/decryption key, the APP_ASYMM_KEY will be encrypted using AES encryption with ECB mode. Input ASYM_KEY to be encrypted: **390BA5453390DF090658776AABA12E867787CC61 62766A70B78CBD016B74B0E2** Therefore encrypted APP_ASYMM_KEY data to be provisioned: **66A84560F07A016E9 BE76070186E1B492479E61E5D35818C5BEA65AC1A437295**
Data to be provided to Secure Key Mode application	Derivation message: **886DD0D58CDF9346ABE32A2B80E8EBFE554B9EF9544B2338** encrypted APP_ASYMM_KEY data: **66A84560F07A016E9BE76070186E1B492479E61E5 D35818C5BEA65AC1A437295**

Interaction in console window:

Figure 19. Provisioning APP_ASYMM_KEY provided in encrypted format

```
Select Option (Hit Enter after Input) : 14

Option 14 selected

KeyIndex where encrypted Asymm Key to be provisioned (27-33) : 29

Key Properties (01 -> SECP256R1, 02 -> SECP384R1, 03 -> BP256R1, 04 -> BP384R1, 05
-> CUSTOM_DP_256, 69 -> CUSTOM_DP_384, 06 -> EDDSA_256) : 01
```

```
Return PUB key associated with this PVT key (00 -> No, 01 -> Yes) : 01
Derivation message for encryption key from APP_ROOT_KEY_128 Key (24 bytes) : 886
DD0D58CDF9346ABE32A2B80E8EBFE554B9EF9544B2338
Encrypted pvt key data (32/48 bytes) :
66a84560f07a016e9be76070186e1b492479e61e5d35818c5bea65ac1a437295
SKM_PROV_APP_ASYMM_KEY_PLAIN :SUCCESS
======PUB_KEY_DATA========================
Key Id : 1D
Key Prop : 01
Public Key : 57C4C08265E9839C37EADB131835107712FBFF5EAC22112CA84492E523F16514E2023
5CB725676C3D03C77E406C55436BB2184D925F5D1FD48DE551CF8162687
==========================================
```

5.16 Setting up the domain parameters for asymmetric keys

First open the session to work on APP_ASYMM_KEY by referring to Section 5.3.

The Secure Key Mode application sets a custom curve which is same as that of **SECP256R1**.

Interaction in console window:

Figure 20. Setting up the APP_ASYMM_KEY

```
Select Option (Hit Enter after Input) : 16
Option 16 selected
Custom Curve Length (05 -> CUSTOM_DP_256, 69 -> CUSTOM_DP_384) : 05
Key Properties (01 -> SECP256R1, 02 -> SECP384R1, 03 -> BP256R1, 04 -> BP384R1, 05
-> ECC256K1) : 01
Key Domain Params are set to the Key Property :
SKM_PROV_APP_ASYMM_KEY_SETDP :SUCCESS
```

5.17 Getting the public key to the corresponding APP_ASYMM_KEY private key

First open the session to work on APP_ASYMM_KEY by referring to Section 5.3.

Then set up the domain parameters to work with APP_ASYMM_KEY by referring to 5.13 above (If custom curve is to be used).

Interaction in console window:

Figure 21. Provisioning APP_ASYMM_KEY provided in encrypted format

```
Select Option (Hit Enter after Input) : 17
Option 17 selected
KeyIndex from where public key to be retrieved from PVT key (27-33) : 29
SKM_PROV_APP_ASYMM_KEY_GET_PUB_KEY :SUCCESS
======PUB_KEY_DATA========================
```

```
Key Id : 1D
Key Prop : 01
Public Key : 57C4C08265E9839C37EADB131835107712FBFF5EAC22112CA84492E523F16514E2023
5CB725676C3D03C77E406C55436BB2184D925F5D1FD48DE551CF8162687
=========================================
```

5.18 Deleting the APP_ASYMM_KEY private key

First open the session to work on APP_ASYMM_KEY by referring to Section 5.3.

Interaction in console window:

Figure 22. Deleting the APP_ASYMM_KEY

```
Select Option (Hit Enter after Input) : 15
Option 15 selected
KeyIndex from where Asymm Key to be deleted (27-33) : 27
SKM_PROV_APP_ASYMM_KEY_DELETE :SUCCESS
```

5.19 Purging the application keys

First open the session to work on PURGE_KEY by referring to Section 5.3.

Interaction in console window:

Figure 23. Purging the application keys

```
Select Option (Hit Enter after Input) : 18
Option 18 selected
PN76_Sys_SKM_Purge_AppKeys : SUCCESS
```

6 References

1. AN13719 PN76 instruction manual, available on https://www.nxp.com/doc/AN13719
2. PN7642 Product data sheet, available on https://www.nxp.com/doc/PN7642

7 Abbreviations

Table 20. Abbreviations

Acronym	Description
CLK	Clock
DH	Device Host
EEPROM	Electrically Erasable Programmable Read Only Memory
FW	Firmware
GND	Ground
GPIO	General Purpose Input Output
HW	Hardware
I²C	Inter-Integrated Circuit (serial data bus)
IRQ	Interrupt Request
ISO/IEC	International Standard Organization / International Electrotechnical Community
NFC	Near Field Communication
OS	Operating System
PCD	Proximity Coupling Device (Contactless reader)
PICC	Proximity Integrated Circuit Card (Contactless card)
PMU	Power Management unit
POR	Power-on reset
RF	Radiofrequency
RST	Reset
SPI	Serial Peripheral Interface
VEN	V Enable pin

8 Legal information

8.1 Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

8.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Evaluation products — This product is provided on an "as is" and "with all faults" basis for evaluation purposes only. NXP Semiconductors, its affiliates and their suppliers expressly disclaim all warranties, whether express, implied or statutory, including but not limited to the implied warranties of non-infringement, merchantability and fitness for a particular purpose. The entire risk as to the quality, or arising out of the use or performance, of this product remains with customer.

In no event shall NXP Semiconductors, its affiliates or their suppliers be liable to customer for any special, indirect, consequential, punitive or incidental damages (including without limitation damages for loss of business, business interruption, loss of use, loss of data or information, and the like) arising out the use of or inability to use the product, whether or not based on tort (including negligence), strict liability, breach of contract, breach of warranty or any other theory, even if advised of the possibility of such damages.

Notwithstanding any damages that customer might incur for any reason whatsoever (including without limitation, all damages referenced above and all direct or general damages), the entire liability of NXP Semiconductors, its affiliates and their suppliers and customer's exclusive remedy for all of the foregoing shall be limited to actual damages incurred by customer based on reasonable reliance up to the greater of the amount actually paid by customer for the product or five dollars (US$5.00). The foregoing limitations, exclusions and disclaimers shall apply to the maximum extent permitted by applicable law, even if any remedy fails of its essential purpose.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

8.3 Licenses

Purchase of NXP ICs with NFC technology — Purchase of an NXP Semiconductors IC that complies with one of the Near Field Communication (NFC) standards ISO/IEC 18092 and ISO/IEC 21481 does not convey an implied license under any patent right infringed by implementation of any of those standards. Purchase of NXP Semiconductors IC does not include a license to any NXP patent (or other IP right) covering combinations of those products with other products, whether hardware or software.

8.4 Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

Tables

Figures

Contents

Date of release: 13 March 2023
Document identifier: AN13720