HID OmniKey 5427

Secure Key Loading Example

Version : 1.0

April 2023

By M. Walker

Contents

Overview

The aim of this document is to provide a sample back-and-forth transaction between a Key Loading Application and Reader for the Secure Channel key loading procedure. Where applicable, commands and responses are decoded to aid in comprehension. I have used colors to link data from one section to the next. Anything not highlighted will either be fixed bytes in the APDU that are not directly linked to the data, or not used in the next section.

The example data set will hopefully prove useful to others whom are trying to implement a secure channel key loading application. Note that the actual sending and receiving of APDUs are not covered within the scope of this document.

The procedures listed herein are based on the code examples provided on the HID public GitHub page. (Note: one line/url, wrapping due to long URL)

https://github.com/hidglobal/HID-OMNIKEY-Sample-Codes/blob/master/HidGlobal.OK.Readers/SecureSession/SamSecureSession/SamSecureSession.cs

All keys used in this example have been made up as the actual secure session keys are not public. As such, while you can use this to prove your code works correctly to create, encode and decode the packets, it is expected it will fail if used to send to an actual reader. Please ensure you have access to the correct reader keys for live key loading operations.

Extracted from the HID GitHub source code examples we can see there are 6 possible keys.

EndUser	0x80 <-- This is the one I am using in this example
EndUserAdmin	0x81
OemUser	0x82
OemUserAdmin	0x83
HidUser	0x84
HidUserAdmin	0x85

Example data used (remember, example only)

```
End User Key (0x80)         : 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
Set iClass key 33 (0x21)    : A0 A1 A2 A3 A4 A5 A6 A7
```

The system uses AES 128 Bit encryption, some in ECB and some in CBC mode.
The IV for the AES can be either all 00’s or based on a previous operation. At each step, I have attempted to ensure all keys, IVs, data etc has been listed and show where it come from along with the operations performed on said data. While this has the effect of increasing verbosity, it is the author's hope that it will avoid confusion as to exactly what data has been used. If the IV (for example) is not 100% clear, look back to the previous steps/actions and see if you can find where it came from.

Authentication - Phase 1

Initial packet and response
The initial packet sent to the reader is a clear APDU. User supplied information is Version (Ver), Secure Session Key Number used (KN) and the client nonce (nonce).

Note: For the purpose of this document "complement" means the inverse of.
e.g. data XOR with FFFFFFFFFFFFFFFFFFFFFFFF

The initial packet send from the key loading application to the reader only needs 3 pieces of data

Ver:	0x01	This is fixed, may change over implementations.
KN:	0x80	The Key ID for the End User Key
Nonce:	5F F8 0F 01 72 93 F4 AE	8 Byte nonce (random data)

```
                                                Ver       KN
TX :     FF 70 07 6B 14 A1 12 A0 10 80 01 01 81 01 80 82
         08 5F F8 0F 01 72 93 F4 AE 00
              nonce

RX :     9D 20 13 51 22 C7 4D 0A 1F 0B 8E FC 8E 23 D1 81
         1E E6 53 85 34 58 61 EB 98 A2 7D 69 E4 B8 F0 7A
         B5 63 90 00
```

Information needed: RX bytes 2..34 (32 bytes Data)

```
13 51 22 C7 4D 0A 1F 0B 8E FC 8E 23 D1 81 1E E6
53 85 34 58 61 EB 98 A2 7D 69 E4 B8 F0 7A B5 63
```

Client nonce appended to end (this is not really needed as long as you have the client nonce)

13 51 22 C7 4D 0A 1F 0B 8E FC 8E 23 D1 81 1E E6	Server UID + Server Nonce
53 85 34 58 61 EB 98 A2 7D 69 E4 B8 F0 7A B5 63	Server Cryptogram
5F F8 0F 01 72 93 F4 AE	Client Nonce

Extract Fields:

Server UID	: 13 51 22 C7 4D 0A 1F 0B
Server Nonce	: 8E FC 8E 23 D1 81 1E E6
Server Cryptogram	: 53 85 34 58 61 EB 98 A2 7D 69 E4 B8 F0 7A B5 63
Client Nonce	: 5F F8 0F 01 72 93 F4 AE

Create Secure Channel base key

Key - EndUser	: 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
Server UID	: 13 51 22 C7 4D 0A 1F 0B
Compliment Server UID	: EC AE DD 38 B2 F5 E0 F4
Data - Server UID + Compliment	: 13 51 22 C7 4D 0A 1F 0B EC AE DD 38 B2 F5 E0 F4
IV	: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Result - AES Encrypt (ECB)	: A3 4F 5A 39 8B EC C2 02 6B ED F8 FF B4 2C E4 EE

Create Session keys

We now use the Secure Channel Base Key and Server Nonce to create the three session keys

Secure Channel Base Key	: A3 4F 5A 39 8B EC C2 02 6B ED F8 FF B4 2C E4 EE
Server Nonce	: 8E FC 8E 23 D1 81 1E E6

Session key Initial data

SN is the first 2 bytes from the serverNonce 8E FC

		SN	
smk1	: 01 01	8E FC	00 00 00 00 00 00 00 00 00 00 00 00
smk2	: 01 02	8E FC	00 00 00 00 00 00 00 00 00 00 00 00
emk1	: 01 82	8E FC	00 00 00 00 00 00 00 00 00 00 00 00

Now encrypted these three with the SecureChannelBaseKey

Data - smk1 initial data	: 01 01 8E FC 00 00 00 00 00 00 00 00 00 00 00 00
Key – SecureChannelBaseKey	: A3 4F 5A 39 8B EC C2 02 6B ED F8 FF B4 2C E4 EE
IV	: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Result - AES Encrypt (ECB)	: **CD 40 05 71 B4 28 D4 85 FB EF 6B 6A 63 5A 37 51**

Data - smk2 initial data	: 01 02 8E FC 00 00 00 00 00 00 00 00 00 00 00 00
Key - SecureChannelBaseKey	: A3 4F 5A 39 8B EC C2 02 6B ED F8 FF B4 2C E4 EE
IV	: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Result – AES Encrypt (ECB)	: **51 ED 86 8B 61 0C CA 4C 97 7F 29 45 BE 7B 0A F0**

Data - emk1 initial data	: 01 82 8E FC 00 00 00 00 00 00 00 00 00 00 00 00
Key - SecureChannelBaseKey	: A3 4F 5A 39 8B EC C2 02 6B ED F8 FF B4 2C E4 EE
IV	: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Result – AES Encrypt (ECB)	: **A0 F6 8B FC C5 13 28 DD 94 74 7C 38 EF CE 40 DE**

Session Keys

smk1	: **CD 40 05 71 B4 28 D4 85 FB EF 6B 6A 63 5A 37 51**
smk2	: **51 ED 86 8B 61 0C CA 4C 97 7F 29 45 BE 7B 0A F0**
emk1	: **A0 F6 8B FC C5 13 28 DD 94 74 7C 38 EF CE 40 DE**

To test that we have created the correct session keys, we can now calculate the server cryptogram and compare to the actual cryptogram we received from the reader.

Data – clientNonce + serverNonce	: 5F F8 0F 01 72 93 F4 AE 8E FC 8E 23 D1 81 1E E6
Key - emk1	: A0 F6 8B FC C5 13 28 DD 94 74 7C 38 EF CE 40 DE
IV	: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Result - AES Encrypt (ECB)	: **53 85 34 58 61 EB 98 A2 7D 69 E4 B8 F0 7A B5 63**
Should match serverCryptogram	: 53 85 34 58 61 EB 98 A2 7D 69 E4 B8 F0 7A B5 63

We have a match, so our emk1 must be correct.

Authentication - Phase 2

Now that we have our session keys, we can build our response to the device to complete the mutual authentication. We should note that the server (device) cryptogram was clientNonce + serverNonce, our response should be the opposite, serverNonce + clientNonce.

Build Next Packet

	Server Nonce	Client Nonce
Data - Client Crypto Data	: 8E FC 8E 23 D1 81 1E E6	5F F8 0F 01 72 93 F4 AE
Key - emk1	: A0 F6 8B FC C5 13 28 DD	94 74 7C 38 EF CE 40 DE
IV	: 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
Result - AES Encrypt (ECB)	: 17 D3 1E 81 33 45 C3 4D	02 E2 1E 4C DD CE 18 2D

Compute cMAC

Data - Client Crypto Block	: 17 D3 1E 81 33 45 C3 4D 02 E2 1E 4C DD CE 18 2D
Key – smk2	: 51 ED 86 8B 61 0C CA 4C 97 7F 29 45 BE 7B 0A F0
IV	: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Result - AES Encrpyt (CBC)	: 1C 58 02 95 83 91 07 49 DE A2 BA 3E 30 D5 F3 24

Build APDU from results above.

FF 70 07 6B 28 A1 26 A1 24	<- Fixed
80 10	<- Fixed
17 D3 1E 81 33 45 C3 4D 02 E2 1E 4C DD CE 18 2D	<- Encrypted Client Crypto
81 10	<- Fixed
1C 58 02 95 83 91 07 49 DE A2 BA 3E 30 D5 F3 24	<- Client Crypto Mac
00	<- Fixed

```
TX:    FF 70 07 6B 28 A1 26 A1 24 80 10 17 D3 1E 81 33
       45 C3 4D 02 E2 1E 4C DD CE 18 2D 81 10 1C 58 02
       95 83 91 07 49 DE A2 BA 3E 30 D5 F3 24 00

RX:    9D 10 00 40 B4 F1 87 67 50 06 69 7F 08 4B CC 88
       16 F9 90 00
```

Correct response should start with 9D 10 and end with 90 00

Response Data : 00 40 B4 F1 87 67 50 06 69 7F 08 4B CC 88 16 F9

This response is the MAC of a null/padding : 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Note: From here the IV is based on previous data. As such it will be flagged as IVBx (IV Base)

Compute MAC

Data - Client Crypto Block	: 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
Key – smk2	: 51 ED 86 8B 61 0C CA 4C 97 7F 29 45 BE 7B 0A F0	
IV	: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
Result - AES Encrypt (CBC)	: 00 40 B4 F1 87 67 50 06 69 7F 08 4B CC 88 16 F9	<- IVB1
Should match data	: 00 40 B4 F1 87 67 50 06 69 7F 08 4B CC 88 16 F9	

We have a match so the device has accepted the last step in the authentication.

Send Key over the Secure Channel

We have now proven each side knows the master key chosen and correctly created the session keys

From here on we use the last MAC as the next IV

smk1	:	CD 40 05 71 B4 28 D4 85 FB EF 6B 6A 63 5A 37 51
smk2	:	51 ED 86 8B 61 0C CA 4C 97 7F 29 45 BE 7B 0A F0
emk1	:	A0 F6 8B FC C5 13 28 DD 94 74 7C 38 EF CE 40 DE
IVB1	:	00 40 B4 F1 87 67 50 06 69 7F 08 4B CC 88 16 F9

<u>Build and send the change key APDU</u>

Build change key packet (this may change for different key types/sizes, more testing needed to confirm)

Encrypt Payload

		KN LN LN key bytes Padding
Data - Clear Change Key Packet	:	FF 82 20 21 08 A0 A1 A2 A3 A4 A5 A6 A7 80 00 00
Key emk1	:	A0 F6 8B FC C5 13 28 DD 94 74 7C 38 EF CE 40 DE
IVB1	:	00 40 B4 F1 87 67 50 06 69 7F 08 4B CC 88 16 F9
IV - Compliment IVB1	:	FF BF 4B 0E 78 98 AF F9 96 80 F7 B4 33 77 E9 06
Result - AES Encrypt CBC	:	8F 91 A4 3F C6 82 9F 2E 54 F6 B6 2E CA 44 E0 34

Compute cMAC

Data - Crypto Block	:	8F 91 A4 3F C6 82 9F 2E 54 F6 B6 2E CA 44 E0 34	
Key - smk2	:	51 ED 86 8B 61 0C CA 4C 97 7F 29 45 BE 7B 0A F0	
IV - IVB1	:	00 40 B4 F1 87 67 50 06 69 7F 08 4B CC 88 16 F9	
Result - AES Encrypt (CBC)	:	66 F8 EA 6E CB C9 95 38 06 65 16 29 0B AD F0 A6	<- IVB2

```
TX:     FF 70 07 6B 20 8F 91 A4 3F C6 82 9F 2E 54 F6 B6
        2E CA 44 E0 34 66 F8 EA 6E CB C9 95 38 06 65 16
        29 0B AD F0 A6 00

RX:     9D 20 A3 83 51 76 F7 EA E1 31 2C 41 87 77 9F 24
        61 F5 41 D4 D7 CF 66 5C 6F D4 E3 14 02 9D 21 A3
        41 A4 90 00
```

Check response

Starts and ends with 9D 20 and ends with 90 00

Cryptogram	:	A3 83 51 76 F7 EA E1 31 2C 41 87 77 9F 24 61 F5	
MAC	:	41 D4 D7 CF 66 5C 6F D4 E3 14 02 9D 21 A3 41 A4	<- IVB3

Data - Cryptogram	:	A3 83 51 76 F7 EA E1 31 2C 41 87 77 9F 24 61 F5
Previous MAC/IVB2	:	66 F8 EA 6E CB C9 95 38 06 65 16 29 0B AD F0 A6
IV – Complement IVB2	:	99 07 15 91 34 36 6A C7 F9 9A E9 D6 F4 52 0F 59
Key - emk1	:	A0 F6 8B FC C5 13 28 DD 94 74 7C 38 EF CE 40 DE
Result - AES (CBC) Decrypt	:	**90 00 80 00 00 00 00 00 00 00 00 00 00 00 00 00**

OK (90 00) and Padding Null (80 00…)

Close Session – Send Null packet

Encrypt Payload

Data - Clear Change Key Packet	:	91 00 80 00 00 00 00 00 00 00 00 00 00 00 00 00
Key - emk1	:	A0 F6 8B FC C5 13 28 DD 94 74 7C 38 EF CE 40 DE
IVB3	:	41 D4 D7 CF 66 5C 6F D4 E3 14 02 9D 21 A3 41 A4
IV - Compliment IVB3	:	BE 2B 28 30 99 A3 90 2B 1C EB FD 62 DE 5C BE 5B
Result - AES Encrypt CBC	:	5F 43 8A EB BE 18 F4 A1 D8 64 0B 12 77 2F FF D1

Compute MAC

Data - Crypto Block	:	5F 43 8A EB BE 18 F4 A1 D8 64 0B 12 77 2F FF D1
Key - smk2	:	51 ED 86 8B 61 0C CA 4C 97 7F 29 45 BE 7B 0A F0
IV - IVB3	:	41 D4 D7 CF 66 5C 6F D4 E3 14 02 9D 21 A3 41 A4
Result - AES Encrypt (CBC)	:	21 D4 01 34 98 34 85 21 6A 28 63 52 B7 F1 EC 4A

TX :
```
FF 70 07 6B 20 5F 43 8A EB BE 18 F4 A1 D8 64 0B
12 77 2F FF D1 21 D4 01 34 98 34 85 21 6A 28 63
52 B7 F1 EC 4A 00
```

RX :
```
9D 00 90 00
```

Response to close secure session: 90 00 OK