YubiHSM 2 Commands

Yubico

May 11, 2022
# CONTENTS

1 Introduction 1

2 AUTHENTICATE SESSION Command 3
  2.1 Description ................................................................. 3
  2.2 Shell Example ............................................................... 3
  2.3 Protocol Details ............................................................ 3

3 BLINK DEVICE Command 5
  3.1 Description ................................................................. 5
  3.2 Shell Example ............................................................... 5
  3.3 Protocol Details ............................................................ 5

4 CHANGE AUTHENTICATION KEY Command 7
  4.1 Description ................................................................. 7
  4.2 Shell Example ............................................................... 7
  4.3 Protocol Details ............................................................ 7

5 CLOSE SESSION Command 9
  5.1 Description ................................................................. 9
  5.2 Shell Example ............................................................... 9
  5.3 Protocol Details ............................................................ 9

6 CREATE OTP AEAD Command 11
  6.1 Description ................................................................. 11
  6.2 Shell Example ............................................................... 11
  6.3 Protocol Details ............................................................ 11

7 CREATE SESSION Command 13
  7.1 Description ................................................................. 13
  7.2 Shell Example ............................................................... 13
  7.3 Protocol Details ............................................................ 13

8 DECRYPT OAEP Command 15
  8.1 Description ................................................................. 15
  8.2 Shell Example ............................................................... 15
  8.3 Protocol Details ............................................................ 15

9 DECRYPT OTP Command 17
  9.1 Description ................................................................. 17
  9.2 Shell Example ............................................................... 17
  9.3 Protocol Details ............................................................ 17
<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>43 SIGN ATTESTATION CERTIFICATE Command</td>
<td>85</td>
</tr>
<tr>
<td>43.1 Description</td>
<td>85</td>
</tr>
<tr>
<td>43.2 Shell Example</td>
<td>85</td>
</tr>
<tr>
<td>43.3 Protocol Details</td>
<td>86</td>
</tr>
<tr>
<td>44 SIGN ECDSA Command</td>
<td>87</td>
</tr>
<tr>
<td>44.1 Description</td>
<td>87</td>
</tr>
<tr>
<td>44.2 Shell Example</td>
<td>87</td>
</tr>
<tr>
<td>44.3 Protocol Details</td>
<td>87</td>
</tr>
<tr>
<td>45 SIGN EDDSA Command</td>
<td>89</td>
</tr>
<tr>
<td>45.1 Description</td>
<td>89</td>
</tr>
<tr>
<td>45.2 Shell Example</td>
<td>89</td>
</tr>
<tr>
<td>45.3 Protocol Details</td>
<td>89</td>
</tr>
<tr>
<td>46 SIGN HMAC Command</td>
<td>91</td>
</tr>
<tr>
<td>46.1 Description</td>
<td>91</td>
</tr>
<tr>
<td>46.2 Shell Example</td>
<td>91</td>
</tr>
<tr>
<td>46.3 Protocol Details</td>
<td>91</td>
</tr>
<tr>
<td>47 SIGN PKCS1 Command</td>
<td>93</td>
</tr>
<tr>
<td>47.1 Description</td>
<td>93</td>
</tr>
<tr>
<td>47.2 Shell Example</td>
<td>93</td>
</tr>
<tr>
<td>47.3 Protocol Details</td>
<td>93</td>
</tr>
<tr>
<td>48 SIGN PSS</td>
<td>95</td>
</tr>
<tr>
<td>48.1 Description</td>
<td>95</td>
</tr>
<tr>
<td>48.2 Shell Example</td>
<td>95</td>
</tr>
<tr>
<td>48.3 Protocol Details</td>
<td>95</td>
</tr>
<tr>
<td>49 SIGN SSH CERTIFICATE Command</td>
<td>97</td>
</tr>
<tr>
<td>49.1 Description</td>
<td>97</td>
</tr>
<tr>
<td>49.2 Shell Example</td>
<td>97</td>
</tr>
<tr>
<td>49.3 Protocol Details</td>
<td>97</td>
</tr>
<tr>
<td>50 UNWRAP DATA Command</td>
<td>99</td>
</tr>
<tr>
<td>50.1 Description</td>
<td>99</td>
</tr>
<tr>
<td>50.2 Shell Example</td>
<td>99</td>
</tr>
<tr>
<td>50.3 Protocol Details</td>
<td>99</td>
</tr>
<tr>
<td>51 VERIFY HMAC Command</td>
<td>101</td>
</tr>
<tr>
<td>51.1 Description</td>
<td>101</td>
</tr>
<tr>
<td>51.2 Shell Example</td>
<td>101</td>
</tr>
<tr>
<td>51.3 Protocol Details</td>
<td>101</td>
</tr>
<tr>
<td>52 WRAP DATA Command</td>
<td>103</td>
</tr>
<tr>
<td>52.1 Description</td>
<td>103</td>
</tr>
<tr>
<td>52.2 Shell Example</td>
<td>103</td>
</tr>
<tr>
<td>52.3 Protocol Details</td>
<td>103</td>
</tr>
<tr>
<td>53 Copyright</td>
<td>105</td>
</tr>
</tbody>
</table>
This section contains a list of the commands supported by the YubiHSM 2.

The low-level format for each command message and the relative response is provided, together with an example of how that command can be used within the yubiHSM-shell.

The numerical codes corresponding to each command are provided below:

<table>
<thead>
<tr>
<th>Command Name</th>
<th>Hex Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Echo</td>
<td>0x01</td>
</tr>
<tr>
<td>Create Session</td>
<td>0x03</td>
</tr>
<tr>
<td>Authenticate Session</td>
<td>0x04</td>
</tr>
<tr>
<td>Session Message</td>
<td>0x05</td>
</tr>
<tr>
<td>Get Device Info</td>
<td>0x06</td>
</tr>
<tr>
<td>Reset Device</td>
<td>0x08</td>
</tr>
<tr>
<td>Close Session</td>
<td>0x40</td>
</tr>
<tr>
<td>Get Storage Info</td>
<td>0x41</td>
</tr>
<tr>
<td>Put Opaque</td>
<td>0x42</td>
</tr>
<tr>
<td>Get Opaque</td>
<td>0x43</td>
</tr>
<tr>
<td>Put Authentication Key</td>
<td>0x44</td>
</tr>
<tr>
<td>Put Asymmetric Key</td>
<td>0x45</td>
</tr>
<tr>
<td>Generate Asymmetric Key</td>
<td>0x46</td>
</tr>
<tr>
<td>Sign Pkcs1</td>
<td>0x47</td>
</tr>
<tr>
<td>List Objects</td>
<td>0x48</td>
</tr>
<tr>
<td>Decrypt Pkcs1</td>
<td>0x49</td>
</tr>
<tr>
<td>Export Wrapped</td>
<td>0x4a</td>
</tr>
<tr>
<td>Import Wrapped</td>
<td>0x4b</td>
</tr>
<tr>
<td>Put Wrap Key</td>
<td>0x4c</td>
</tr>
<tr>
<td>Get Log Entries</td>
<td>0x4d</td>
</tr>
<tr>
<td>Get Object Info</td>
<td>0x4e</td>
</tr>
<tr>
<td>Set Option</td>
<td>0x4f</td>
</tr>
<tr>
<td>Get Option</td>
<td>0x50</td>
</tr>
<tr>
<td>Get Pseudo Random</td>
<td>0x51</td>
</tr>
<tr>
<td>Put Hmac Key</td>
<td>0x52</td>
</tr>
<tr>
<td>Sign Hmac</td>
<td>0x53</td>
</tr>
<tr>
<td>Get Public Key</td>
<td>0x54</td>
</tr>
<tr>
<td>Sign Pss</td>
<td>0x55</td>
</tr>
<tr>
<td>Sign Ecdsa</td>
<td>0x56</td>
</tr>
<tr>
<td>Derive Ecdh</td>
<td>0x57</td>
</tr>
<tr>
<td>Delete Object</td>
<td>0x58</td>
</tr>
<tr>
<td>Decrypt Oaep</td>
<td>0x59</td>
</tr>
</tbody>
</table>

continues on next page
Table 1 – continued from previous page

<table>
<thead>
<tr>
<th>Command Name</th>
<th>Hex Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generate Hmac Key</td>
<td>0x5a</td>
</tr>
<tr>
<td>Generate Wrap Key</td>
<td>0x5b</td>
</tr>
<tr>
<td>Verify Hmac</td>
<td>0x5c</td>
</tr>
<tr>
<td>Sign Ssh Certificate</td>
<td>0x5d</td>
</tr>
<tr>
<td>Put Template</td>
<td>0x5e</td>
</tr>
<tr>
<td>Get Template</td>
<td>0x5f</td>
</tr>
<tr>
<td>Decrypt Otp</td>
<td>0x60</td>
</tr>
<tr>
<td>Create Otp Aead</td>
<td>0x61</td>
</tr>
<tr>
<td>Randomize Otp Aead</td>
<td>0x62</td>
</tr>
<tr>
<td>Rewrap Otp Aead</td>
<td>0x63</td>
</tr>
<tr>
<td>Sign Attestation Certificate</td>
<td>0x64</td>
</tr>
<tr>
<td>Put Otp Aead Key</td>
<td>0x65</td>
</tr>
<tr>
<td>Generate Otp Aead Key</td>
<td>0x66</td>
</tr>
<tr>
<td>Set Log Index</td>
<td>0x67</td>
</tr>
<tr>
<td>Wrap Data</td>
<td>0x68</td>
</tr>
<tr>
<td>Unwrap Data</td>
<td>0x69</td>
</tr>
<tr>
<td>Sign Eddsa</td>
<td>0x6a</td>
</tr>
<tr>
<td>Blink Device</td>
<td>0x6b</td>
</tr>
<tr>
<td>Change Authentication Key</td>
<td>0x6c</td>
</tr>
</tbody>
</table>
AUTHENTICATE SESSION COMMAND

Complete the mutual authentication process started with \textit{CREATE SESSION Command}. 

2.1 Description

Finish the Session negotiation and authenticate the Session to the device. After this command completes successfully the Session is authenticated and can be used.

2.2 Shell Example

Create a new Session with Authentication Key 1 using the password \texttt{password}, this performs both the creation and authentication steps:

```
yubihs\> session open 1 password
Created session 0
```

2.3 Protocol Details

2.3.1 Command

<table>
<thead>
<tr>
<th>Tc</th>
<th>0x04</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lc</td>
<td>17</td>
</tr>
<tr>
<td>Vc</td>
<td>S</td>
</tr>
</tbody>
</table>

Parameters:

\begin{align*}
S & := \text{Session ID (1 byte)} \\
B & := \text{Host Cryptogram (8 bytes)} \\
M & := \text{CMAC}(S-MAC, 016 || T || Lc + 8 || S || B) \text{ (8 bytes)}
\end{align*}

This is the first authenticated message in the chain. The device verifies $M$ and $B$, both using $S$-MAC.
### 2.3.2 Response

<table>
<thead>
<tr>
<th>Tr</th>
<th>0x84</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lr</td>
<td>0</td>
</tr>
<tr>
<td>Vr</td>
<td>Ø</td>
</tr>
</tbody>
</table>
3.1 Description

Blink the LED of the device to identify it.

3.2 Shell Example

Blink the device for 15 seconds:

```
yubihsmd> blink @ 15
```

3.3 Protocol Details

3.3.1 Command

<table>
<thead>
<tr>
<th>Tc</th>
<th>0x6b</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lc</td>
<td>1</td>
</tr>
<tr>
<td>Vc</td>
<td>S</td>
</tr>
</tbody>
</table>

Parameters:

S := Seconds to blink for (1 byte)

3.3.2 Response

<table>
<thead>
<tr>
<th>Tr</th>
<th>0xeb</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lr</td>
<td>0</td>
</tr>
<tr>
<td>Vr</td>
<td>Ø</td>
</tr>
</tbody>
</table>
Chapter Four

Change Authentication Key Command

Change an Authentication Key.

4.1 Description

Replace the Authentication Key used to establish the current Session. It is not possible to modify any of the metadata connected to the Object such as Domains or Capabilities. Only the payload data of the Object (i.e., the long-lived symmetric keys) will be modified.

The same PBKDF2 derivation scheme described in Session is available.

4.2 Shell Example

Change the current Authentication Key deriving it from the password newpassword:

```bash
yubihsm> change authkey 0 1 newpassword
Changed Authentication key 0x0001
```

4.3 Protocol Details

4.3.1 Command

<table>
<thead>
<tr>
<th>Tc</th>
<th>Lc</th>
<th>Vc</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x6c</td>
<td>2 + 1 + 16 + 16</td>
<td>I</td>
</tr>
</tbody>
</table>

Replace the currently used Authentication Key with a new set of keys.

Parameters:

- I := Object ID of the Authentication Key (2 bytes)
- A := Algorithm (1 byte)
- Ke := Encryption Key (16 bytes)
- Km := Mac Key (16 bytes)
4.3.2 Response

<table>
<thead>
<tr>
<th>Tr</th>
<th>0xec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lr</td>
<td>2</td>
</tr>
<tr>
<td>Vr</td>
<td>I</td>
</tr>
</tbody>
</table>

Parameters:

I := Object ID of the changed Object (2 bytes)
CLOSE SESSION COMMAND

Close session.

5.1 Description

Close the current Session and release it for re-use.

5.2 Shell Example

Close Session 0:

```
yubihsn> session close 0
```

5.3 Protocol Details

5.3.1 Command

\[
\begin{array}{|c|}
\hline
Tc \ = \ 0x40 \\
Lc \ = \ 0 \\
Vc \ = \ Ø \\
\hline
\end{array}
\]

5.3.2 Response

\[
\begin{array}{|c|}
\hline
Tr \ = \ 0xc0 \\
Lr \ = \ 0 \\
Vr \ = \ Ø \\
\hline
\end{array}
\]

CHAPTER SIX

CREATE OTP AEAD COMMAND

Create a Yubico OTP AEAD.

6.1 Description

Create a Yubico OTP AEAD using the provided data.

6.2 Shell Example

Create a new AEAD using Otp-aead Key 0x027c with the key 000102030405060708090a0b0c0d0e0f and private ID 010203040506. Store the result in the file aead:

```
yubihsn> otp aead_create 0 0x027c 000102030405060708090a0b0c0d0e0f 010203040506 aead
```

6.3 Protocol Details

6.3.1 Command

| Tc = 0x61 |
| Lc = 24 |
| Vc = I || K || P |

Parameters:

I := Object ID of the OTP AEAD Key (2 bytes)

K := OTP Key (16 bytes)

P := OTP Private ID (6 bytes)
6.3.2 Response

\[
\begin{array}{|l|}
\hline
T_r &= 0xe1 \\
L_r &= L_A \\
V_r &= A \\
\hline
\end{array}
\]

Parameters:

\[A := \text{Nonce concatenated with AEAD (36 bytes)}\]
CHAPTER
SEVEN

CREATE SESSION COMMAND

Begin the mutual authentication process for establishing a Session.

7.1 Description

Start negotiating a Session with the device. This command tells the device which Authentication Key to use and sends the host challenge part. The response will contain the device challenge and device authentication part. To establish the session continue with AUTHENTICATE SESSION Command.

7.2 Shell Example

Create a new session with Authentication Key 1 using the password password. This does both the session creation and authentication steps:

```
yubihs> session open 1 password
Created session 0
```

7.3 Protocol Details

7.3.1 Command

<table>
<thead>
<tr>
<th>Tc</th>
<th>Lc</th>
<th>Vc</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x03</td>
<td>10</td>
<td>I</td>
</tr>
</tbody>
</table>

Parameters:

I := Key set ID (2 bytes)

H := Host Challenge (8 bytes)

The device generates a random Card Challenge C (8 bytes).

The device derives three Session Keys (S-ENC, S-MAC and S-RMAC) starting from the set of two static keys identified by I (K-ENC and K-MAC) and the two challenges H and C, using the same procedure described in SCP03.

The device uses S-MAC together with H and C to compute the Card Cryptogram A. The host will compute the Host Cryptogram B after having received C and derived S-MAC.
On success the device generates a Session ID $S$ (1 byte) and sets the message counter for the current Session to 1.

### 7.3.2 Response

<table>
<thead>
<tr>
<th>$Tr$</th>
<th>0x83</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Lr$</td>
<td>17</td>
</tr>
<tr>
<td>$Vr$</td>
<td>$S \mid C \mid A$</td>
</tr>
</tbody>
</table>
8.1 Description

Decrypt data encrypted with RSA-OAEP

8.2 Shell Example

Decrypt data stored in file enc using key 0x79c3:

```bash
yubihsm> decrypt oaep 0 0x79c3 rsa-oaep-sha1 enc
xlwIc7yQf/KkV5v4Y87Q9ZSqiLReoNAXlCmmMPA4W08U=
```

8.3 Protocol Details

8.3.1 Command

| TC = 0x59 |
| Lc = 2 + 1 + LD + LH |
| Vc = I || M || D || HL |

Parameters:

- **I** := Object ID of the Asymmetric Key (2 bytes)
- **M** := Hash Algorithm to use for MGF1 (1 byte)
- **D** := Decryption data (256, 384 or 512 bytes)
- **HL** := Hash of OAEP Label (20, 32, 48 or 64 bytes)
8.3.2 Response

\[
\begin{align*}
T_r &= 0xc9 \\
L_r &= LR \\
V_r &= R
\end{align*}
\]

Parameters:

\( R := \) Decrypted data with OAEP padding removed
Decrypted a Yubico OTP.

9.1 Description

Decrypt a Yubico OTP and return counters and timer information.

9.2 Shell Example

Decrypt a (hex encoded) Yubico OTP using key ID 0x027c:

```
yubihs> otp decrypt 0 0x027c 2f5d71a4915dec304aa13ccf97bb0d0bb aead
OTP decoded, useCtr:1, sessionCtr:1, tstph:1, tstpl:1
```

9.3 Protocol Details

9.3.1 Command

<table>
<thead>
<tr>
<th>Tc</th>
<th>0x60</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lc</td>
<td>2 + 36 + 16</td>
</tr>
<tr>
<td>Vc</td>
<td>K</td>
</tr>
</tbody>
</table>

Parameters:

- I := Object ID of the OTP AEAD Key (2 bytes)
- A := Nonce concatenated with AEAD (36 bytes)
- O := OTP (16 bytes)
9.3.2 Response

<table>
<thead>
<tr>
<th>Tr</th>
<th>0xe0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lr</td>
<td>6</td>
</tr>
<tr>
<td>Vr</td>
<td>S</td>
</tr>
</tbody>
</table>

Parameters:
- S := Session counter (2 bytes)
- U := Usage counter (1 byte)
- Th := Timestamp high (1 byte)
- Tl := Timestamp low (2 bytes)
Decipher data that was encrypted using RSA-PKCS#1v1.5.

10.1 Description

Decrypt data encrypted with RSA-PKCS#1v1.5

10.2 Shell Example

Decrypt the file enc using key 0xa930:

```bash
yubihsm> decrypt pkcs1v1_5 0 0xa930 enc
xlwIc7yQf/KkV5v4Y87Q9ZSqqLReoNAx1CmmMPA4W08U=
```

10.3 Protocol Details

10.3.1 Command

<table>
<thead>
<tr>
<th>Tc</th>
<th>0x49</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lc</td>
<td>2 + LD</td>
</tr>
<tr>
<td>Vc</td>
<td>I</td>
</tr>
</tbody>
</table>

Parameters:

- I := Object ID of the Asymmetric Key (2 bytes)
- D := Decryption data (256, 384 or 512 bytes)

The data is padded using the PKCS#1v1.5 scheme with Block Type 2. The data is decrypted and conformance to the padding scheme must be checked. Padding is then removed and the contained message is returned.
10.3.2 Response

<table>
<thead>
<tr>
<th>Tr</th>
<th>0xc9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lr</td>
<td>LR</td>
</tr>
<tr>
<td>Vr</td>
<td>R</td>
</tr>
</tbody>
</table>

Parameters:

R := Decrypted data with padding removed
DELETE OBJECT COMMAND

Delete an Object.

11.1 Description

Delete an Object in the device.

11.2 Shell Example

Delete Asymmetric Key 0x52b6:

```
yubihsms> delete 0 0x52b6 asymmetric-key
```

11.3 Protocol Details

11.3.1 Command

```
Tc = 0x58
Lc = 2 + 1
Vc = I || T
```

Parameters:

- \( I \): Object ID (2 bytes)
- \( T \): Type (1 byte)
11.3.2 Response

<table>
<thead>
<tr>
<th>Tr</th>
<th>0xd8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lr</td>
<td>0</td>
</tr>
<tr>
<td>Vr</td>
<td>Ø</td>
</tr>
</tbody>
</table>
DERIVE ECDH COMMAND

Perform an ECDH operation.

12.1 Description

Perform an ECDH key exchange with the private key in the device.

12.2 Shell Example

Perform an ECDH operation with key \(0x52b6\) and a public key in the file pubkey.pem:

```
yubihsm> derive ecdh 0 0x52b6 pubkey.pem
5898516bcb0cb3db89d53471137c2d1c741b8ba6ebf2bb01f4a62d97342e97b2
```

12.3 Protocol Details

12.3.1 Command

\[
\begin{align*}
T_c &= 0x57 \\
L_c &= 2 + LD \\
V_c &= K || D
\end{align*}
\]

Parameters:

- \(I\) := Object ID of the Asymmetric Key (2 bytes)
- \(D\) := Uncompressed public key to perform the exchange with (57, 65, 97, 129 or 133 bytes)
12.3.2 Response

<table>
<thead>
<tr>
<th>Tc</th>
<th>0xd7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lc</td>
<td>LX</td>
</tr>
<tr>
<td>Vc</td>
<td>X</td>
</tr>
</tbody>
</table>

Parameters:

\[ X := \text{X coordinate of the completed key exchange} \]
Get device metadata.

### 13.1 Description

Gets device version, device serial, supported Algorithms and available log entries.

### 13.2 Shell Example

Fetch device info for currently connected device:

```
yubihs> get deviceinfo
Version number: 2.0.0
Serial number: 2000000
Log used: 2/62
```
13.3 Protocol Details

13.3.1 Command

\[
\begin{array}{c}
\text{Tc} = 0x06 \\
\text{Lc} = 0 \\
\text{Vc} = \emptyset
\end{array}
\]

13.3.2 Response

\[
\begin{array}{c}
\text{Tr} = 0x86 \\
\text{Lr} = 9 + \text{algorithms} \\
\text{Vr} = \text{VMajor} || \text{VMinor} || \text{VBuild} || \text{S} || \text{Ltotal} || \text{Lused} || \text{A}
\end{array}
\]

Parameters:

- \text{VMajor} := Major version number (1 byte)
- \text{VMinor} := Minor version number (1 byte)
- \text{VBuild} := Build version number (1 byte)
- \text{S} := Serial number (4 bytes)
- \text{Ltotal} := Log Store size expressed in number of log entries (1 byte)
- \text{Lused} := Log lines used (1 byte)
- \text{A} := List of supported Algorithms
Echo data back from the device.

### 14.1 Description

Return the byte sequence present within the data field, without any modification. Can be sent over an encrypted Session or as a bare command.

### 14.2 Shell Example

Plain echo:

```
yubihsn> plain echo 0x3c 10
Response (10 bytes):
3c3c3c3c3c3c3c 3c3c
```

Echo over session 0:

```
yubihsn> echo 0 0x3c 10
Response (10 bytes):
3c3c3c3c3c3c3c 3c3c
```

### 14.3 Protocol Details

#### 14.3.1 Command

<table>
<thead>
<tr>
<th>Tc</th>
<th>0x01</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lc</td>
<td>LE</td>
</tr>
<tr>
<td>Vc</td>
<td>E</td>
</tr>
</tbody>
</table>

Parameters:

E: Data to echo (1-2021 bytes)
14.3.2 Response

<table>
<thead>
<tr>
<th>Tr</th>
<th>0x81</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lr</td>
<td>LE</td>
</tr>
<tr>
<td>Vr</td>
<td>E</td>
</tr>
</tbody>
</table>

Parameters:

E: Data to echo (1-2021 bytes)
Get an Object in encrypted form.

15.1 Description

Retrieves an Object under wrap from the device. The Object is encrypted using AES-CCM with a 16 bytes MAC and a 13 bytes nonce.

15.2 Shell Example

Fetch the Asymmetric Key 0x997e encrypted with Wrap Key 0xcf94 and store the result in the file key.enc:

```
yubihsn> get wrapped 0 0xcf94 asymmetric 0x997e key.enc
```

15.3 Protocol Details

15.3.1 Command

<table>
<thead>
<tr>
<th>Tc</th>
<th>0x4a</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lc</td>
<td>2 + 1 + 2</td>
</tr>
<tr>
<td>Vc</td>
<td>Iw</td>
</tr>
</tbody>
</table>

Parameters:

- \( Iw \) := Object ID of Wrap Key to use (2 bytes)
- \( T \) := Type of Object to wrap (1 byte)
- \( Io \) := Object ID of Object to wrap (2 bytes)
15.3.2 Response

<table>
<thead>
<tr>
<th>Tr</th>
<th>(0x\text{ca})</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lr</td>
<td>(13 + \text{LR})</td>
</tr>
<tr>
<td>Vr</td>
<td>(N \parallel R)</td>
</tr>
</tbody>
</table>

Parameters:

- \(N\) := Nonce used for this wrap (13 bytes)
- \(R\) := Wrapped data (Length dependent on object)
CHAPTER
SIXTEEN

GENERATE ASYMMETRIC KEY COMMAND

Generate an Asymmetric Key.

16.1 Description

Generate an Asymmetric Key in the device.

16.2 Shell Example

Generate a new key using secp256r1 in the device:

```
yubihsmdump> generate asymmetric 0 0 eckey 1 sign-ecdsa ecp256
Generated Asymmetric key 0x2846
```

16.3 Protocol Details

16.3.1 Command

| Tc = 0x46 |
| Lc = 2 + 40 + 2 + 8 + 1 |
| Vc = I || L || D || C || A |

Generate an Asymmetric key-pair with a given ID. Each parameter has a fixed length and the order is compulsory.

Parameters:

- **I**: Object ID of the Asymmetric Key (2 bytes)
- **L**: Label (40 bytes)
- **D**: Domains (2 bytes)
- **C**: Capabilities (8 bytes)
- **A**: Algorithm (1 byte)
16.3.2 Response

<table>
<thead>
<tr>
<th>Tr</th>
<th>0xc6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lr</td>
<td>2</td>
</tr>
<tr>
<td>Vr</td>
<td>I</td>
</tr>
</tbody>
</table>

Parameters:

\[ I := \text{Object ID of the created Asymmetric Key (2 bytes)} \]
CHAPTER
SEVENTEEN

GENERATE HMAC KEY COMMAND

Generate an HMAC Key.

17.1 Description

Generate an HMAC Key in the device.

17.2 Shell Example

Generate an HMAC-SHA512 key:

```
yubihs> generate hmackey 0 0 hmackey 1 sign-hmac:verify-hmac hmac-sha512
Generated HMAC key 0xa9bf
```

17.3 Protocol Details

17.3.1 Command

```
Tc = 0x5a
Lc = 2 + 40 + 2 + 8 + 1
Vr = I || L || D || C || A
```

Parameters:

- I := Object ID of the HMAC Key (2 bytes)
- L := Label (40 bytes)
- D := Domains (2 bytes)
- C := Capabilities (8 bytes)
- A := Algorithm (1 byte)
17.3.2 Response

<table>
<thead>
<tr>
<th>Tr</th>
<th>0xda</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lr</td>
<td>2</td>
</tr>
<tr>
<td>Vr</td>
<td>I</td>
</tr>
</tbody>
</table>

Parameters:

I:= Object ID of the created HMAC Key (2 bytes)
GENERATE OTP AEAD KEY COMMAND

Generate an OTP AEAD Key.

18.1 Description

Generate an OTP AEAD Key for Yubico OTP decryption.

18.2 Shell Example

Generate a new AES-256 OTP AEAD Key that can decrypt Yubico OTPs and create new AEADs:

```
yubihsn> generate otpaeadkey 0 0 otpaeadkey 1 decrypt-otp,
create-otp-aead aes256-yubico-otp 0x01020304
Generated OTP AEAD key 0x027c
```

18.3 Protocol Details

18.3.1 Command

```
Tc = 0x66
Lc = 2 + 40 + 2 + 8 + 1 + 4
Vc = I || L || D || C || A || N
```

Parameters:

- \( I \) := Object ID of the OTP AEAD Key (2 bytes)
- \( L \) := Label (40 bytes)
- \( D \) := Domains (2 bytes)
- \( C \) := Capabilities (8 bytes)
- \( A \) := Algorithm (1 byte)
- \( N \) := Nonce ID (4 bytes)
18.3.2 Response

\[
\begin{array}{|c|}
\hline
\text{Tr} = 0xe6 \\
\text{Lr} = 2 \\
\text{Vr} = I \\
\hline
\end{array}
\]

Parameters:

\( I := \text{Object ID of the created OTP AEAD Key (2 bytes)} \)
GENERATE WRAP KEY COMMAND

Generate a Wrap Key.

19.1 Description

Generate a Wrap Key that can be used for export, import, wrap data and unwrap data.

19.2 Shell Example

Generate a new Wrap Key that can be used for wrap and unwrap:

```
$ yubihsm> generate wrapkey 0 0 wrapkey 1 wrap-data:unwrap-data none
    aes256-ccm-wrap
Generated Wrap key 0x5b3a
```

19.3 Protocol Details

19.3.1 Command

<table>
<thead>
<tr>
<th>Tc</th>
<th>Lc</th>
<th>Vc</th>
</tr>
</thead>
<tbody>
<tr>
<td>@x5b</td>
<td>2 + 40 + 2 + 8 + 1 + 8</td>
<td>I</td>
</tr>
</tbody>
</table>

Parameters:

- I := Object ID of the Wrap Key (2 bytes)
- L := Label (40 bytes)
- D := Domains (2 bytes)
- C := Capabilities (8 bytes)
- A := Algorithm (1 byte)
- DC := Delegated Capabilities (8 bytes)
19.3.2 Response

<table>
<thead>
<tr>
<th>Tr</th>
<th>0xdb</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lr</td>
<td>2</td>
</tr>
<tr>
<td>Vr</td>
<td>I</td>
</tr>
</tbody>
</table>

Parameters:

\[ I := \text{Object ID of created Wrap Key (2 bytes)} \]
CHAPTER TWENTY

GET LOG ENTRIES COMMAND

Fetch device audit log.

20.1 Description

Fetch all current entries from the device Log Store.

20.2 Shell Example

```
yubihs-> audit get 0
0 unlogged boots found
0 unlogged authentications found
Found 6 items
item: 46 -- cmd: 0x4b -- length: 234 -- session key: 0x0001
  -- target
key: 0xcf94 -- second key: 0x997e -- result: 0xcb -- tick: 335725
  -- hash: 415f51f1f035a1b713e730e4464e4033
item: 47 -- cmd: 0x4c -- length: 77 -- session key: 0x0001
  -- target
key: 0xaff7 -- second key: 0xffffff -- result: 0xcc -- tick: 351714
  -- hash: 5496a60d478c2b9c801d8d32ca66b554
item: 48 -- cmd: 0x00 -- length: 0 -- session key: 0xffffff
  -- target
key: 0xffffff -- second key: 0x0000 -- result: 0x00 -- tick: 0 -- hash:
    14ac7747ba9bb243cfc70befe5349b
item: 49 -- cmd: 0x03 -- length: 10 -- session key: 0xffffff
  -- target
key: 0x0001 -- second key: 0xffffff -- result: 0x83 -- tick: 139 -- hash:
    b20a8f25c025e693a8e869b433294a20
item: 50 -- cmd: 0x04 -- length: 17 -- session key: 0xffffff
  -- target
key: 0x0001 -- second key: 0xffffff -- result: 0x84 -- tick: 139 -- hash:
    ebfae425c319ac7a0afbb8b92597de7c
item: 51 -- cmd: 0x67 -- length: 2 -- session key: 0x0001
  -- target
key: 0xffffff -- second key: 0xffffff -- result: 0xe7 -- tick: 697 -- hash:
    2e395d1b706668737e1d2215813db47e
```
20.3 Protocol Details

20.3.1 Command

\[
\begin{array}{c}
T_c = 0x4d \\
L_c = 0 \\
V_c = \emptyset
\end{array}
\]

20.3.2 Response

\[
\begin{array}{c}
T_r = 0xcd \\
L_r = 2 + 2 + 1 + (N \times 32) \\
V_r = B || O || N || E_1 || E_2 || \ldots || E_N
\end{array}
\]

Parameters:

- \(B\) := Number of unlogged boot events (if the log buffer is full and audit enforce is set) (2 bytes)
- \(O\) := Number of unlogged authentication events (if the log buffer is full and audit enforce is set) (2 bytes)
- \(N\) := Number of elements in the list (1 byte)
- \(E_i\) := Generic log entry composed of
  - Command number (two bytes)
  - Command ID (one byte)
  - Command length (two bytes)
  - ID of the originating session’s authentication key (two bytes)
  - Target key affected by the command (two bytes)
  - Secondary key if the command affected more than one key (two bytes)
  - Result of the command on success or an error code if unsuccessful (one byte)
  - Systick when the command was processed (4 bytes)
  - Digest (16 bytes)

The digest is computed as \(\text{trunc}(16, \text{SHA256}(E_i.\text{Data} || \text{trunc}(16, E_{i-1}.\text{Digest})))\). For the initial log entry, a random string of 32 bytes is used, instead of the digest of the previous message.

When the device initializes after a reset, a log entry with all fields set to \(0xff\) is logged.
When the device boots up, a log entry with all fields set to \(0x00\) is logged.
GET OBJECT INFO COMMAND

Get Object metadata.

21.1 Description

Fetch all metadata about an Object

21.2 Shell Example

Get Object info for Asymmetric Key with ID 0x1e15:

```
yubihs> get objectinfo 0 0x1e15 asymmetric-key
id: 0x1e15, type: asymmetric-key, algorithm: rsa2048, label: "rsakey",
    length: 896, domains: 1, sequence: 0, origin: imported, capabilities:
        sign-pkcs
```

21.3 Protocol Details

21.3.1 Command

```
Tc = 0x4e
Lc = 2 + 1
Vc = I || T
```

Parameters:

- I := Object ID (2 bytes)
- T := Type (1 byte)
21.3.2 Response

<table>
<thead>
<tr>
<th>Tr = 0xce</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lr = 8 + 2 + 2 + 2 + 1 + 1 + 1 + 1 + 40 + 8</td>
</tr>
<tr>
<td>Vr = C</td>
</tr>
</tbody>
</table>

Parameters:

- C := Capabilities (8 bytes)
- I := Object ID (2 bytes)
- N := Object Length (2 bytes)
- D := Domains (2 bytes)
- T := Type (1 byte)
- A := Algorithm (1 byte)
- S := Sequence (1 byte)
- O := Origin (1 byte)
- L := Label (40 bytes)
- DC := Delegated Capabilities (8 bytes)
Fetch an Opaque Object from device.

### 22.1 Description

Retrieve an Opaque Object (like an X.509 certificate) from the device.

### 22.2 Shell Example

Fetch Opaque Object 0xe255 and store in the file cert.der:

```
yubihsn> get opaque 0 0xe255 cert.der
```

### 22.3 Protocol Details

#### 22.3.1 Command

<table>
<thead>
<tr>
<th>Tc</th>
<th>0x43</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lc</td>
<td>2</td>
</tr>
<tr>
<td>Vc</td>
<td>1</td>
</tr>
</tbody>
</table>

Parameters:
- \( I := \text{Object ID (2 bytes)} \)

#### 22.3.2 Response

<table>
<thead>
<tr>
<th>Tr</th>
<th>0xc3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lr</td>
<td>LD</td>
</tr>
<tr>
<td>Vr</td>
<td>D</td>
</tr>
</tbody>
</table>

Parameters:
- \( D := \text{Data} \)
GET OPTION COMMAND

Fetch a device-global option.

23.1 Description

Get device-global Options. Each invocation of this command retrieves a single Option, which is selected by its represented TAG (see SET OPTION Command).

23.2 Shell Example

```bash
yubihsm> get option 0 force-audit
Option value is: 00
```

23.3 Protocol Details

23.3.1 Command

```
Tc = 0x50
Lc = 1
Vc = T
```

Parameters:

T := The tag of the selected option (1 byte)

23.3.2 Response

```
Tr = 0xd0
Lr = LO
Vr = 0
```

Parameters:

O := The option-specific value (LO bytes)
CHAPTER TWENTYFOUR

GET PSEUDO RANDOM COMMAND

Get pseudo-random data from device.

24.1 Description

Extract a fixed number of pseudo-random bytes from the device, using the internal PRNG.

24.2 Shell Example

```
yubihsn> get random 0 16
bd50979da2dbca13d8d735abf419556
```

24.3 Protocol Details

24.3.1 Command

<table>
<thead>
<tr>
<th>$Tc$</th>
<th>$0x51$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Lc$</td>
<td>$2$</td>
</tr>
<tr>
<td>$Vc$</td>
<td>$B$</td>
</tr>
</tbody>
</table>

Parameters:

$B$ := Number of pseudo-random bytes to extract (2 bytes)

24.3.2 Response

<table>
<thead>
<tr>
<th>$Tr$</th>
<th>$0xd1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Lr$</td>
<td>$B$</td>
</tr>
<tr>
<td>$Vr$</td>
<td>$R$</td>
</tr>
</tbody>
</table>

Parameters:

$R$ := Random data ($B$ bytes)
CHAPTER TWENTYFIVE

GET PUBLIC KEY COMMAND

Fetch a public key from device.

25.1 Description

Fetch the public key of an Asymmetric Key.

25.2 Shell Example

Fetch the public key of Asymmetric Key 0x2846:

```
yubihsn> get pubkey 0x2846
-----BEGIN PUBLIC KEY-----
MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAE85fayPHTvCrv0RRcyCshv0hTKAM7
xHiU2InG06lTRQumGDeBnQZIITykK/0PWKLGDAMfBVrmKkWwxB47ze9A==
-----END PUBLIC KEY-----
```

25.3 Protocol Details

25.3.1 Command

<table>
<thead>
<tr>
<th>Tc</th>
<th>Lc</th>
<th>Vc</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x54</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

Parameters:

I := Object ID of the Asymmetric Key (2 bytes)
25.3.2 Response

\[
\begin{array}{ll}
T_r &= \text{0xd4} \\
L_r &= 1 + L_{P1} \{ + L_{P2} \} \\
V_r &= A \ || \ P_{1} \{ \ || \ P_{2} \}
\end{array}
\]

Parameters:

- **A** := Algorithm (1 byte)
- **P1** :=
  - For RSA: Public modulus N (256, 384 or 512 bytes)
  - For ECC: Public point X (32, 48, 64 or 66 bytes)
  - For EDC: Public point A, compressed (32 bytes)
- **P2** :=
  - For RSA: NOT DEFINED
  - For ECC: Public point Y (32, 48, 64 or 66 bytes)
  - For EDC: NOT DEFINED
GET STORAGE INFO COMMAND

Fetch storage information.

26.1 Description

Report currently free storage. This is reported as currently free records, free pages and page size. Each object takes a record slot and will use as many pages as needed.

26.2 Shell Example

```
yubihsn> get storage 0
free records: 255/256, free pages: 1023/1024 page size: 126 bytes
```

26.3 Protocol Details

26.3.1 Command

```
Tc = 0x41
Lc = 0
Vc = Ø
```

26.3.2 Response

```
Tr = 0xc1
Lr = 10
Vr = Rtotal || Rfree || Ptotal || Pfree || S
```

Parameters:

- **Rtotal**: Total number of records (2 bytes)
- **Rfree**: Currently free storage records (2 bytes)
- **Ptotal**: Total number of pages (2 bytes)
Pfree := Currently free storage pages (2 bytes)
S := Page size in bytes (2 bytes)
GET TEMPLATE COMMAND

Fetch a Template Object from the device.

27.1 Description

Retrieve a Template Object from the device.

27.2 Shell Example

Fetch Template Object \(0x7b19\) and store in the file template.dat:

```
yubihs> get template 0 0x7b19 template.dat
```

27.3 Protocol Details

27.3.1 Command

\[
\begin{aligned}
Tc &= 0x5f \\
Lc &= 2 \\
Vc &= I
\end{aligned}
\]

Parameters:

\(I := \) Object ID of the Template to retrieve (2 bytes)

27.3.2 Response

\[
\begin{aligned}
Tr &= 0xdf \\
Lr &= LD \\
Nr &= D
\end{aligned}
\]

Parameters:

\(D := \) Data
**CHAPTER TWENTYEIGHT**

**IMPORT WRAPPED COMMAND**

Import an wrapped/encrypted object into the device.

### 28.1 Description

Import a wrapped/encrypted Object that was previously exported by an YubiHSM 2 device. The imported object will retain its metadata (Object ID, Domains, Capabilities ...etc), however, the object’s origin will be marked as *imported* instead of *generated*.

### 28.2 Shell Example

Import the Object stored in `key.enc` and unwrap it using Wrap Key `0xcf94`

```
yubiHsm> put wrapped 0 0xcf94 key.enc
Object imported as 0x997e of type asymmetric
```

### 28.3 Protocol Details

#### 28.3.1 Command

<table>
<thead>
<tr>
<th>Tc</th>
<th>Lc</th>
<th>Vc</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x4b</td>
<td>2 + 13 + LO</td>
<td>I</td>
</tr>
</tbody>
</table>

Parameters:

- **I**: Object ID of the Wrap Key (2 bytes)
- **N**: Nonce associated with this wrapped Object (13 bytes)
- **O**: Wrapped Object (Length dependant on Object)
28.3.2 Response

\[
\begin{array}{|c|}
\hline
T_c = 0xcb \\
L_c = 3 \\
V_c = T \, || \, I \\
\hline
\end{array}
\]

Parameters:

- \( T \) := Type of imported Object (1 byte)
- \( I \) := Object ID of imported Object (2 bytes)
LIST OBJECTS COMMAND

List Objects in device.

29.1 Description

Get a filtered list of Objects from the device.

29.2 Shell Example

Get a list of all Asymmetric Keys for Session 0:

```
yubihsn> list objects 0 0 asymmetric-key
Found 4 object(s)
id: 0x3479, type: asymmetric-key, sequence: 0
id: 0x7df6, type: asymmetric-key, sequence: 0
id: 0x9602, type: asymmetric-key, sequence: 0
id: 0xd6cd, type: asymmetric-key, sequence: 0
```

29.3 Protocol Details

29.3.1 Command

\[
\begin{array}{c}
\text{Tc} = \text{0x48} \\
\text{Lc} = \text{LF} \\
\text{Vc} = \text{F}
\end{array}
\]

Parameters:

\( F \) := List of Tag-Value pairs describing a filter to apply. Possible tags to use for filtering are described in the table below.
YubiHSM 2 Commands

<table>
<thead>
<tr>
<th>Name</th>
<th>Identifier</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>0x01</td>
<td>2 bytes</td>
</tr>
<tr>
<td>TYPE</td>
<td>0x02</td>
<td>1 byte</td>
</tr>
<tr>
<td>DOMAINS</td>
<td>0x03</td>
<td>2 bytes</td>
</tr>
<tr>
<td>CAPABILITIES</td>
<td>0x04</td>
<td>8 bytes</td>
</tr>
<tr>
<td>ALGORITHM</td>
<td>0x05</td>
<td>1 byte</td>
</tr>
<tr>
<td>LABEL</td>
<td>0x06</td>
<td>40 bytes</td>
</tr>
</tbody>
</table>

29.3.2 Response

<table>
<thead>
<tr>
<th>Tr = 0xc8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lr = 4 * N</td>
</tr>
<tr>
<td>Vr = R1</td>
</tr>
</tbody>
</table>

Parameters:

Ri := Object ID (2 bytes), Type (1 byte) and Sequence (1 byte).
CHAPTER THIRTY

PUT ASYMMETRIC KEY COMMAND

Import an Asymmetric Key.

30.1 Description

Import an Asymmetric Key into the device.

30.2 Shell Example

Store an RSA key from key.pem into the device:

```
yubiHSM> put asymmetric 0 0 rsakey 1 sign-pkcs key.pem
Stored Asymmetric key 0x1e15
```

30.3 Protocol Details

30.3.1 Command

```
Tc = 0x45
Lc = 2 + 40 + 2 + 8 + 1 + LP1 { + LP2 }
Vc = I || L || D || C || A || P1 { || P2 }
```

The key parameters vary according to the chosen algorithm. Each parameter has a fixed length and the order is compulsory.

Parameters:

- I := Object ID of the Asymmetric Key (2 bytes)
- L := Label (40 bytes)
- D := Domains (2 bytes)
- C := Capabilities (8 bytes)
- A := Algorithm (1 byte)
- P1 :=
YubiHSM 2 Commands

For RSA: secret prime p (128, 192 or 256 bytes)
For ECC: private key integer d (32, 48, 64 or 66 bytes)
For EDC: private key integer k (32 bytes)

P2 :=
For RSA: secret prime q (128, 192 or 256 bytes)
For ECC: NOT DEFINED
For EDC: NOT DEFINED

30.3.2 Response

\[
\begin{array}{|l|}
\hline
Tr = 0xc5 \\
Lr = 2 \\
Vr = 1 \\
\hline
\end{array}
\]

Parameters:

I := ID of created Object (2 bytes)
CHAPTER THIRTYONE

PUT AUTHENTICATION KEY COMMAND

Store a new Authentication Key.

31.1 Description

Store an Authentication Key in the device.

31.2 Shell Example

Store a new Authentication Key derived from the password newpassword:

```plaintext
yubihsm> put authkey 0 0 authkey 1 generate-asymmetric-key,sign-pkcs
   sign-pkcs newpassword
Stored Authentication key 0xbb72
```

31.3 Protocol Details

31.3.1 Command

```
Tc = 0x44
Lc = 2 + 40 + 2 + 8 + 1 + 8 + 16 + 16
Vc = I || L || D || C || A || DC || Ke || Km
```

Parameters:

- **I**: Object ID of the Authentication Key (2 bytes)
- **L**: Label (40 bytes)
- **D**: Domains (2 bytes)
- **C**: Capabilities (8 bytes)
- **A**: Algorithm (1 byte)
- **DC**: Delegated Capabilities (8 bytes)
- **Ke**: Encryption Key (16 bytes)
- **Km**: Mac Key (16 bytes)
31.3.2 Response

<table>
<thead>
<tr>
<th>Tr</th>
<th>0xc4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lr</td>
<td>2</td>
</tr>
<tr>
<td>Vr</td>
<td>I</td>
</tr>
</tbody>
</table>

Parameters:

I := Object ID of created Authentication Key (2 bytes)
CHAPTER
THIRTYTWO

PUT HMAC KEY COMMAND

Import an HMAC Key.

32.1 Description

Store an HMAC Key in the device.

32.2 Shell Example

Store an HMAC Key with the binary value 666f6f in the device:

```plaintext
yubihsm> put hmackey 0 0 hmackey 1 sign-hmac,verify-hmac hmac-sha256 666f6f
Stored HMAC key 0x7cf2
```

32.3 Protocol Details

32.3.1 Command

<table>
<thead>
<tr>
<th>Tc = 0x52</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lc = 2 + 40 + 2 + 8 + 1 + LP</td>
</tr>
<tr>
<td>Vc = I</td>
</tr>
</tbody>
</table>

Parameters:

I := Object ID of the HMAC Key (2 bytes)
L := Label (40 bytes)
D := Domains (2 bytes)
C := Capabilities (8 bytes)
A := Algorithm (1 byte)
P := Key (Minimum 1 byte)

For HMAC-SHA1 and HMAC-SHA256: maximum 64 bytes
For HMAC-SHA384 and HMAC-SHA512: maximum 128 bytes
32.3.2 Response

<table>
<thead>
<tr>
<th>Tr</th>
<th>0xd2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lr</td>
<td>2</td>
</tr>
<tr>
<td>Vr</td>
<td>I</td>
</tr>
</tbody>
</table>

Parameters:

I := Object ID of created HMAC Key (2 bytes)
PUT OPAQUE COMMAND

Store an Opaque Object.

33.1 Description

Stores Opaque data (like an X.509 certificate) in the device. The size of the object is currently limited to what will fit into one message to the YubiHSM 2 (2028 bytes, including the headers).

33.2 Shell Example

Store the certificate in file cert.der in the device:

```
yubihsms> put opaque 0 0 certificate 1 none opaque-x509-certificate cert.der
Stored Opaque object 0xe255
```

33.3 Protocol Details

33.3.1 Command

<table>
<thead>
<tr>
<th>$T_c$</th>
<th>$0x42$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_c$</td>
<td>$2 + 40 + 2 + 8 + 1 + L_0$</td>
</tr>
<tr>
<td>$V_c$</td>
<td>$I</td>
</tr>
</tbody>
</table>

Parameters:

- $I$ := Object ID (2 bytes)
- $L$ := Label (40 bytes)
- $D$ := Domains (2 bytes)
- $C$ := Capabilities (8 bytes)
- $A$ := Algorithm (1 byte)
- $O$ := Opaque data
33.3.2 Response

<table>
<thead>
<tr>
<th>Tr</th>
<th>0xc2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lr</td>
<td>2</td>
</tr>
<tr>
<td>Vr</td>
<td>I</td>
</tr>
</tbody>
</table>

Parameters:

I := Object ID of created Opaque Object (2 bytes)
PUT OTP AEAD KEY COMMAND

Import an OTP AEAD Key.

34.1 Description

Import an OTP AEAD Key used for Yubico OTP Decryption.

34.2 Shell Example

Import OTP AEAD Key with Nonce ID \texttt{0x01020304} and key value \texttt{000102030405060708090a0b0c0d0e0f} (AES-128):

\begin{verbatim}
yubihsn> put otpaeadkey 0 0 otpaeadkey 1 decrypt-otp 0x01020304␣ → 000102030405060708090a0b0c0d0e0f
Stored OTP AEAD key \texttt{0xe34f}
\end{verbatim}

34.3 Protocol Details

34.3.1 Command

\begin{center}
\begin{tabular}{|c|}
\hline
\textbf{Tc} = \texttt{0x65} \\
\hline
\textbf{Lc} = 2 + 40 + 2 + 8 + 1 + 4 + \text{LK} \\
\hline
\textbf{Vc} = I \mid | \mid L \mid | | D \mid | | C \mid | | A \mid | | N \mid | | K \\
\hline
\end{tabular}
\end{center}

Parameters:

- \textbf{I} := Object ID (2 bytes)
- \textbf{L} := Label (40 bytes)
- \textbf{D} := Domains (2 bytes)
- \textbf{C} := Capabilities (8 bytes)
- \textbf{A} := Algorithm (1 byte)
- \textbf{N} := Nonce ID (4 bytes)
- \textbf{K} := Key (16, 24 or 32 bytes depending on algorithm)
34.3.2 Response

<table>
<thead>
<tr>
<th>Tr</th>
<th>0xe5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lr</td>
<td>2</td>
</tr>
<tr>
<td>Vr</td>
<td>I</td>
</tr>
</tbody>
</table>

Parameters:

I := ID of created OTP AEAD Key (2 bytes)
PUT TEMPLATE COMMAND

Store a Template.

35.1 Description

Stores a Template in the device. The size of the object is currently limited to what will fit into one message to the YubiHSM (2021 bytes, including the headers).

35.2 Shell Example

Store the SSH Template in file `template.dat` in the device:

```
  yubihsn> put template 0 0 ssh_template 1 none template-ssh template.dat
  Stored Template object 0x7b19
```

35.3 Protocol Details

35.3.1 Command

```
\begin{align*}
  T_c &= 0x5e \\
  L_c &= 2 + \text{40} + 2 + 8 + 1 + \text{LD} \\
  V_c &= I || L || D || C || A || D
\end{align*}
```

Parameters:

- \( I \) := Object ID of the Template (2 bytes)
- \( L \) := Label (40 bytes)
- \( D \) := Domains (2 bytes)
- \( C \) := Capabilities (8 bytes)
- \( A \) := Algorithm (1 byte)
- \( D \) := Template data
35.3.2 Response

<table>
<thead>
<tr>
<th>Tr</th>
<th>0xde</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lr</td>
<td>2</td>
</tr>
<tr>
<td>Vr</td>
<td>I</td>
</tr>
</tbody>
</table>

Parameters:

I := Object ID of created Template (2 bytes)
CHAPTER
THIRTYSEVEN

PUT WRAP KEY COMMAND

Import a Wrap Key.

36.1 Description

Import a key for wrapping into the device.

36.2 Shell Example

Import an AES-128 Wrap Key able to export and import, with some Delegated Capabilities set:

```bash
yubihsm> put wrapkey 0 0 wrapkey 1 export-wrapped,import-wrapped
exportable-under-wrap,sign-pkcs,sign-pss 000102030405060708090a0b0c0d0e0f
Stored Wrap key 0xaaff7
```

36.3 Protocol Details

36.3.1 Command

<table>
<thead>
<tr>
<th>Tc</th>
<th>Lc</th>
<th>Vc</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x4c</td>
<td>(2 + 40 + 2 + 8 + 1 + 8 + LW)</td>
<td>(I</td>
</tr>
</tbody>
</table>

Parameters:

- \(I\) := Object ID (2 bytes)
- \(L\) := Label (40 bytes)
- \(D\) := Domains (2 bytes)
- \(C\) := Capabilities (8 bytes)
- \(A\) := Algorithm (1 byte)
- \(DC\) := Delegated Capabilities (8 bytes)
- \(W\) := Wrap Key (16, 24 or 32 bytes)
YubiHSM 2 Commands

For AES128_CCM_WRAP: 16 bytes
For AES192_CCM_WRAP: 24 bytes
For AES256_CCM_WRAP: 32 bytes

36.3.2 Response

<table>
<thead>
<tr>
<th>Tc</th>
<th>0xcc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lc</td>
<td>2</td>
</tr>
<tr>
<td>Vc</td>
<td>I</td>
</tr>
</tbody>
</table>

Parameters:

I := ID of created Wrap Key (2 bytes)
RANDOMIZE OTP AEAD COMMAND

Create an OTP AEAD from random data.

37.1 Description

Create a new OTP AEAD using random data for key and private ID.

37.2 Shell Example

Generate a new OTP AEAD using OTP AEAD Key 0xc5f4 and put the result in file aead:

```
yubihsn> otp aead_random 0 0xc5f4 aead
```

37.3 Protocol Details

37.3.1 Command

<table>
<thead>
<tr>
<th>Tc</th>
<th>0x62</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lc</td>
<td>2</td>
</tr>
<tr>
<td>Vc</td>
<td>1</td>
</tr>
</tbody>
</table>

Parameters:

I := Object ID for the OTP AEAD Key (2 bytes)

37.3.2 Response

<table>
<thead>
<tr>
<th>Tr</th>
<th>0xe2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lr</td>
<td>36</td>
</tr>
<tr>
<td>Vr</td>
<td>A</td>
</tr>
</tbody>
</table>

Parameters:

A := Nonce concatenated with AEAD (36 bytes)
CHAPTER THIRTYEIGHT

RESET DEVICE COMMAND

Factory reset a device.

38.1 Description

Resets and reboots the device, deletes all Objects and restores the default Options and Authentication Key.

38.2 Shell Example

Send reset over Session 0:

```bash
yubihsn> reset 0
Device successfully reset
```

38.3 Protocol Details

38.3.1 Command

| Tc = 0x08 |
| Lc = 0    |
| Vc = Ø    |

38.3.2 Response

| Tr = 0x88 |
| Lr = 0    |
| Vr = Ø    |
REWARP OTP AEAD COMMAND

Rewrap an OTP AEAD.

39.1 Description

Re-encrypt a Yubico OTP AEAD from one OTP AEAD Key to another OTP AEAD Key.

39.2 Shell Example

N/A

39.3 Protocol Details

39.3.1 Command

<table>
<thead>
<tr>
<th>Tc</th>
<th>0x63</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lc</td>
<td>2 + 2 + 36</td>
</tr>
<tr>
<td>Vc</td>
<td>I1</td>
</tr>
</tbody>
</table>

Parameters:

- I1 := Key ID from (2 bytes)
- I2 := Key ID to (2 bytes)
- A := Nonce concatenated with AEAD (36 bytes)
39.3.2 Response

<table>
<thead>
<tr>
<th>Tr</th>
<th>0xe3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lr</td>
<td>36</td>
</tr>
<tr>
<td>Vr</td>
<td>A</td>
</tr>
</tbody>
</table>

Parameters:

\[ A := \text{Nonce concatenated with AEAD (36 bytes)} \]
SESSION MESSAGE COMMAND

Send a command over an established session.

40.1 Description

Sends a wrapped command for a previously established session. The command is encrypted and authenticated.

40.2 Shell Example

Send an echo over Session 0:

```
yubihs> echo 0 0xff 1
Response (1 bytes):
 ff
```

40.3 Protocol Details

40.3.1 Command

```
Tc = 0x05  
Lc = 1 + Linner_c + 8  
Vc = S || Ic || Mc
```

Parameters:

- S := Session ID (1 byte)
- Linner_c/inner_r := Length of the encrypted inner command / response (2 bytes)
- Mc/r := CMAC of the outer command / response (8 bytes)
40.3.2 Response

\[
\begin{align*}
T_r &= 0x85 \\
L_r &= 1 + \text{Linner}_r + 8 \\
V_r &= S || T_r || M_r
\end{align*}
\]
Set the last extracted log entry.

### 41.1 Description

Inform the device what the last extracted log entry is so logs can be reused. Mostly of practical use when forced auditing is enabled.

### 41.2 Shell Example

Set log index 41 as the last extracted entry:

```
yubihsn> audit set 0 41
``` 

### 41.3 Protocol Details

#### 41.3.1 Command

<table>
<thead>
<tr>
<th>Tc</th>
<th>Lc</th>
<th>Vc</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x67</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

Parameters:

- I := Index to set as last read log (2 bytes)
41.3.2 Response

<table>
<thead>
<tr>
<th>Tr</th>
<th>0xe7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lr</td>
<td>0</td>
</tr>
<tr>
<td>Vr</td>
<td>\varnothing</td>
</tr>
</tbody>
</table>
CHAPTER FORTYTWO

SET OPTION COMMAND

Set a device-global option.

42.1 Description

Set device-global options that affect general behavior. Each invocation of this command sets a single option, which is represented as a TAG-LENGTH-VALUE (TLV).

42.2 Shell Example

Turn off audit logging for Sign HMAC (command 53) and Verify HMAC (command 5c):

```
yubihsmd> put option 0 command_audit 53005c00
```

42.3 Protocol Details

42.3.1 Command

```
Tc = 0x4f
Lc = 3 + Lo
Vc = To
```

Parameters:

- \( To \) := The TLV encoding of the selected option
- \( Lo \) := The option-specific length in bytes

The options currently supported are the following:

- **TAG** is 1 byte
- **LENGTH** is 2 bytes
- **VALUE** is \( Lo \) bytes
Tags:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Force audit</td>
<td>0x01</td>
<td></td>
</tr>
<tr>
<td>Command audit</td>
<td>0x03</td>
<td></td>
</tr>
</tbody>
</table>

Values:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>OFF</td>
<td>0x00</td>
<td>Disabled</td>
</tr>
<tr>
<td>ON</td>
<td>0x01</td>
<td>Enabled</td>
</tr>
<tr>
<td>FIX</td>
<td>0x02</td>
<td>Enabled, only possible to turn off through factory reset</td>
</tr>
</tbody>
</table>

There are two supported options, **Force audit** and **Command audit**.

With **Force audit** set, the device will refuse operations as long as the Log Store is full. It takes a 1 byte value option. **Command audit** can be used to toggle whether a specific command should be logged, this takes tuples of command number and option value.

### 42.3.2 Response

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Tr</td>
<td>0xcf</td>
<td></td>
</tr>
<tr>
<td>Lr</td>
<td>@</td>
<td></td>
</tr>
<tr>
<td>Vr</td>
<td>@</td>
<td></td>
</tr>
</tbody>
</table>
SIGN ATTESTATION CERTIFICATE COMMAND

Attest an Asymmetric Key.

43.1 Description

Get attestation of an Asymmetric Key, output is an X.509 certificate.

43.2 Shell Example

Attest Asymmetric Key 0x79c3 using attestation key 0 (builtin):

```
yubihsm> attest asymmetric 0 0x79c3 0
-----BEGIN CERTIFICATE-----
MIIDeTCCAmGgAwIBAgIQaa8FkvRhqntp5HjyyCfilzANBgkqhkiG9w0BAQsFADAn
MSUwIwYDVQQDDBxzdWJpSFNNIEF0dGVzdGF0aW9uICgxMjM0NTYpMCAXDTE3MDEw
MTAwMCMwF0dIzIwNzExMCMaMDAwNjA0c3JvbGl0aW9ucyBzZXcgZGlkOjB4NzljMzCC
ASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAMYpAzHar0syanQEiRqWy8WDO5q
ETjDuI02txNBWyMCNgeEYzo/uglUXLEmZj6Dd8EcdY9upHqVpLduB+GIt+UEqSDeMNS
rzj2QZ/1QMELMdadaD90dc707aPvKT/oAujiaZ89vfq7jEWWBTPquyFxaCBoz8WWta9j
5XxRppQrR7ub43950fXwpWbvl1NLx0QAQdDqEm2V3THhnbu6T5XsgC780dOikyJw2TP
62rQXS7GRuXob/QaINsJRxbbydQUXDFNq8CnSkl8dHsNd7b0SdAV6Vw30JFbJ2uoW2E
GMF9qYWnTEVvyPMMQWf90r9HVPf83TBAy0MCnEAAAoBnTCBmmjATBgqrBgE4EY1ECq
QDBAAUEAwIAAADATBggorBgEEAYLCEqQCBARUCx6EgDASBgorBgEEAYLCEqQDBAQDgA
RMRMGcisGAQQBgsQKBAEQBMDAAAABMBkgCisGAQQBgsQKBAUgCMJAAAAAAAARABMIG
cisGAQQBgsQKBAYEBAIcMwFgYKKwYBBAGCxAoECQIDAZyc2FrZXkwDQQJKoZIhcv
NACQELBQADggEBABRReYze+KRfevrgyI3C2aLAWSiQRj36vvaP1Fh4b0w4X2HC
rLAI150h405eH/axXNv+368FWLQhY68jJKDoeckrlt9thFxphasd/Wt1Pbqzj
trnEiillyjP6rrdCR1iytmnQ3nssk3w1mTE/AtzmoD0i7V/wNymilB790FDGmB6P
d1VI7zGUDtL1j1qeyY4/ETqKuPzYZ5RUPYr08/iPzy64AdtDxtl3e39npTc0hp2
PSQq63gU7vt9+5SeEj0CF/qTk317L1R4TfeHFS1gBTHSWuVcDORNJxDHTco
bI+wE2dCknjyLU9dr5tkNsd3k5pscuTmpBgFDlge=
-----END CERTIFICATE-----
```
43.3 Protocol Details

43.3.1 Command

| Tc = 0x64 |
| Lc = 2 + 2 |
| Vc = I || A |

Parameters:

I := Object ID of the Asymmetric Key to attest (2 bytes)
A := Object ID of the Asymmetric Key used for attestation (2 bytes)

If A is 0 the internal attestation key is used.

43.3.2 Response

| Tr = 0xe4 |
| Lr = LX |
| Vr = X |

Parameters:

X := DER encoded X.509 attestation
SIGN ECDSA COMMAND

Sign data with ECDSA.

44.1 Description

Computes a digital signature using ECDSA on the provided data.

44.2 Shell Example

Sign data in file data using key 0x52b6 and put the result in file sig:

```
yubihs> sign ecdsa 0 0x52b6 ecdsa-sha256 data sig
```

44.3 Protocol Details

44.3.1 Command

```
Tc = 0x56
Lc = 2 + LD
Vc = I || D
```

Parameters:

- **I**: Object ID of the Asymmetric Key (2 bytes)
- **D**: H

The DSI for ECDSA is a possibly zero-left-padded hash of the data, H.
## 44.3.2 Response

<table>
<thead>
<tr>
<th>Tr</th>
<th>0xd6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lr</td>
<td>LDS</td>
</tr>
<tr>
<td>Vr</td>
<td>DS</td>
</tr>
</tbody>
</table>

Parameters:

\[ DS := \text{Resulting signature} \]

The length of DS, LDS, depends on the Algorithm used and equals the length of the signature plus its DER encoding.
Sign with EdDSA.

### 45.1 Description

Computes a digital signature using EdDSA on the provided data.

### 45.2 Shell Example

Perform an EdDSA signature with key 0xddf6 of the content of file data:

```
yubihsm> sign eddsa 0 0xddf6 ed25519 data
    wZljrOstOLPuMHGrXDnpAb5Wxo79+wX/vQkb/6K34tOd8se/QfLNRVtonfErttkWUAz/
    →...UINtaG4XJyY8vabCQ==
```

### 45.3 Protocol Details

#### 45.3.1 Command

<table>
<thead>
<tr>
<th>Tc</th>
<th>Lc = 2 + LD</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0x6a$</td>
<td>$Vc = I</td>
</tr>
</tbody>
</table>

Parameters:

$I := \text{Object ID of the Asymmetric Key (2 bytes)}$

The DSI for EdDSA is the raw data D.

$DSI := D$

For a given DSI, the command will generate a digital signature DS. The length of DS, LDS, depends on the Algorithm used. At this time only Ed25519 is implemented.

$DS := \text{EdDSA(DSI). Key is omitted}$

$DS := 0x0040$ bytes
45.3.2 Response

\[
\begin{array}{|c|}
\hline
Tr = 0xea \\
Lr = LDS \\
Vr = DS \\
\hline
\end{array}
\]

Parameters:

\[DS := \text{Resulting signature}\]
SIGN HMAC COMMAND

HMAC data.

46.1 Description

Perform an HMAC operation in device and return the result.

46.2 Shell Example

Perform an HMAC operation using the HMAC Key 0x7cf2:

```
yubihs> hmac 0 0x7cf2 666f6f6261
   →4c17e17300a51a3f8aeeba131e9c680e4e40b429aa1d547807efd8e3d95ccd39
```

46.3 Protocol Details

46.3.1 Command

\[
\begin{align*}
Tc &= 0x53 \\
Lc &= 2 + LD \\
Vc &= I || D
\end{align*}
\]

Parameters:

\( I \) := Object ID of the HMAC Key (2 bytes)

\( D \) := Data to HMAC
46.3.2 Response

\[
\begin{array}{l}
T_r = 0xd3 \\
L_r = LR \\
V_r = R
\end{array}
\]

Parameters:

\( R := \text{HMAC Response, 20, 32, 48 or 64 bytes depending on the Algorithm.} \)
Sign with RSA-PKCS#1v1.5.

### 47.1 Description

Computes a digital signature using RSA-PKCS1v1.5 on the provided data.

### 47.2 Shell Example

Sign the data in the file `test` using rsa-pkcs1-sha256:

```
yubihsn> sign pkcs1v1_5 0 0x1e15 rsa-pkcs1-sha256 test
```

### 47.3 Protocol Details

#### 47.3.1 Command

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$Tc$</td>
<td>0x047</td>
</tr>
<tr>
<td>$Lc$</td>
<td>$2 + LD$</td>
</tr>
<tr>
<td>$Vc$</td>
<td>$I | D$</td>
</tr>
</tbody>
</table>

Parameters:

- $I$ := Object ID of the Asymmetric Key (2 bytes)
- $D$ := Digest

The Digest can be either a raw hash of data, where DigestInfo will be applied in the device, or DigestInfo + hash. Hashes supported are SHA-1, SHA-256, SHA-384 and SHA-512.
47.3.2 Response

<table>
<thead>
<tr>
<th>Tr</th>
<th>0xc7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lr</td>
<td>LDS</td>
</tr>
<tr>
<td>Vr</td>
<td>DS</td>
</tr>
</tbody>
</table>

Parameters:

DS := Resulting signature
SIGN PSS

Sign data using RSA-PSS.

48.1 Description

Computes a digital signature using RSA-PSS on the provided data.

48.2 Shell Example

Sign what is in file data using key 0x79c3 and put the resulting signature in sig:

```
yubihs> sign pss 0 0x79c3 rsa-pss-sha256 data sig
```

48.3 Protocol Details

48.3.1 Command

\[
\begin{align*}
T_c &= 0x55 \\
L_c &= 2 + 1 + 2 + LD \\
V_c &= I || M || S || D
\end{align*}
\]

Parameters:

- \(I\) := Object ID of the Asymmetric Key (2 bytes)
- \(M\) := Hash Algorithm to use for MGF1
- \(S\) := Salt len (2 bytes)
- \(D\) := Hashed data (20, 32, 48 or 64 bytes)

The DSI of EMSA-PSS is as defined in RFC 3447:

\[
\text{DSI} := \text{EMSA-PSS-ENCODE}(M, \text{emBits}, \text{Hash}, \text{MGF}, sLen).
\]

- \(\text{Hash}\) is a supported hash Algorithm
- \(\text{MGF}\) is a supported masking function
- \(sLen\) is the length of the Salt
The DSI is generated internally and only the Hash of the data and the Salt length are provided.

### 48.3.2 Response

<table>
<thead>
<tr>
<th>Tr</th>
<th>0xd5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lr</td>
<td>LDS</td>
</tr>
<tr>
<td>Vr</td>
<td>DS</td>
</tr>
</tbody>
</table>

Parameters:

DS := Resulting signature
SIGN SSH CERTIFICATE COMMAND

Sign an SSH Certificate request.

49.1 Description

Produce an SSH Certificate signature. The certificate can then be used to login to hosts.

49.2 Shell Example

Produce a new SSH Certificate.

```
yubihsn> certify 0 0xabcd 0x1234 rsa-pkcs-sha256 req.dat cert.dat
```

49.3 Protocol Details

49.3.1 Command

\[
\begin{align*}
T_c &= 0x5d \\
L_c &= 2 + 2 + 1 + 4 + 256 + LR \\
V_c &= I || T || A || N || S || R
\end{align*}
\]

Sign and SSH Certificate by using the given Asymmetric Key and SSH Template.

Parameters:

- \(I\) := Object ID of the Asymmetric Key (2 bytes)
- \(T\) := Object ID of the SSH Template (2 bytes)
- \(A\) := Algorithm (1 byte)
- \(N\) := Timestamp with the definition of \(\text{Now}\) (4 bytes)
- \(S\) := Signature over the request and timestamp (256 bytes)
- \(R\) := Request (LR bytes)
49.3.2 Response

\[
\begin{array}{|c|}
\hline
\text{Tr} = 0\text{xd6} \\
\text{Lr} = \text{LS} \\
\text{Vr} = \text{S} \\
\hline
\end{array}
\]

Parameters:

\( S := \text{Certificate Signature (LS bytes)} \)
UNWRAP DATA COMMAND

Decrypt data.

50.1 Description

Decrypt (unwrap) data using a Wrap Key.

50.2 Shell Example

```
yubiHsm> decrypt aesccm 0 0x5b3a MRkj6B0AAAAAAAAAo04dkIeAYoPvwTV/M/JX1dwKnLqnER01hSW4wPS
Hello world!
```

50.3 Protocol Details

50.3.1 Command

<table>
<thead>
<tr>
<th>( Tc )</th>
<th>( 0x69 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( Lc )</td>
<td>( 2 + 13 + LD + 16 )</td>
</tr>
<tr>
<td>( Vc )</td>
<td>( I</td>
</tr>
</tbody>
</table>

Parameters:

- \( I \) := Object ID of a Wrap Key (2 bytes)
- \( N \) := Nonce (13 bytes)
- \( D \) := Data to be unwrapped
- \( M \) := Mac (16 bytes)
50.3.2 Response

<table>
<thead>
<tr>
<th>Tr</th>
<th>0xe9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lr</td>
<td>LD</td>
</tr>
<tr>
<td>Vr</td>
<td>D</td>
</tr>
</tbody>
</table>

Parameters:

\[ D := \text{Unwrapped data} \]
Verify an HMAC.

51.1 Description

Verify a generated HMAC.

51.2 Shell Example

N/A

51.3 Protocol Details

51.3.1 Command

<table>
<thead>
<tr>
<th>Tc</th>
<th>0x5c</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lc</td>
<td>2 + LH + LD</td>
</tr>
<tr>
<td>Vc</td>
<td>I</td>
</tr>
</tbody>
</table>

Parameters:

I := Object ID of the HMAC Key (2 bytes)
H := HMAC (20, 32, 48 or 64 bytes)
D := Data
51.3.2 Response

<table>
<thead>
<tr>
<th>Tr</th>
<th>0x0dc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lr</td>
<td>1</td>
</tr>
<tr>
<td>Vr</td>
<td>V</td>
</tr>
</tbody>
</table>

Parameters:

\[
V := \text{Verified (1 byte)}
\]

\[
V \text{ will have the value } 1 \text{ if verification succeeded and } 0 \text{ otherwise.}
\]
CHAPTER
FIFTYTWO

WRAP DATA COMMAND

Encrypt data.

52.1 Description

Encrypt (wrap) data using a Wrap Key.

52.2 Shell Example

Using Wrap Key 0x5b3a encrypt the string “Hello world!”:

```
$ yubihsmd encrypt aesccm 0x5b3a "Hello world!"
MRkj6B0AAAAAAAAAAoO4dkIeAYoPvwTV/M/JX1dwKnLqnER01hSW4wPS
```

52.3 Protocol Details

52.3.1 Command

<table>
<thead>
<tr>
<th>Tc</th>
<th>0x68</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lc</td>
<td>2 + LD</td>
</tr>
<tr>
<td>Vc</td>
<td>I</td>
</tr>
</tbody>
</table>

Parameters:

- **I** := Object ID of the Wrap Key (2 bytes)
- **D** := Data to be wrapped
52.3.2 Response

| Tr = 0xe8  |
| Lr = 13 + LD + 16 |
| Vr = N || D || M |

Parameters:

N := Nonce (13 bytes)
D := Wrapped data
M := Mac (16 bytes)
© 2022 Yubico AB. All rights reserved.

**Trademarks**

Yubico and YubiKey are registered trademarks of Yubico AB. All other trademarks are the property of their respective owners.

**Disclaimer**

The contents of this document are subject to revision without notice due to continued progress in methodology, design, and manufacturing. Yubico shall have no liability for any error or damages of any kind resulting from the use of this document.

The Yubico Software referenced in this document is licensed to you under the terms and conditions accompanying the software or as otherwise agreed between you or the company that you are representing.

**Contact Information**

Yubico Inc.
530 Lytton Street
Suite 301
Palo Alto, CA 94301
USA

Click the links to:

- Submit a support request
- Send a Contact Me request
- See additional contact options for getting touch with us

**Document Updated**

2022-05-11 23:17:23 UTC