YubiKey PIV Tool User Guide

Yubico

Sep 09, 2025

CONTENTS

CHAPTER

ONE

INTRODUCTION

Use the Yubico PIV tool for interacting with the Personal Identity Verification (PIV) application on a YubiKey.

Through the Yubico PIV tool, you can generate keys on the device, import keys and certificates, and create certificate requests, and other operations. A shared library and a command-line tool, `yubico-piv-tool`, is included.

The Yubico PIV tool was designed to interact with and manage the PIV functions alone. Built on the C ykpiv library, the PIV tool provides a CLI to access all of the functionality supported on the PIV function of the YubiKey. While the PIV tool allows for the CLI to be used as part of a scripted process, the lack of support beyond the PIV functions means that it is less script-friendly than `ykman`. However, as a purpose built interface on just the ykpiv library, the PIV tool is an excellent reference architecture for supporting the YubiKey as a PIV smart card natively.

The PIV tool also provides a PKCS#11 module, called YKCS11, that can be used to expose the YubiKey's smart card functionality to applications that communicate with hard tokens through the PKCS#11 API. For example, OpenSSL, OpenSSH, JAVA, FireFox and the like.

Use the PIV tool when `ykman` does not have a specific command, or when testing the PIV functionality of the YubiKey. On POSIX platforms, PIV tool requires `pcscd` to be pre-installed.

- See the Yubico PIV Tool Release Notes for PIV Tool versions.
- See the PIV Introduction on developers.yubico.com for information about performing RSA or ECC sign/decrypt operations using a private key stored on the YubiKey smartcard, through common interfaces like PKCS#11.

1.1 PIV Standard

PIV, or FIPS 201, is a US government standard. It enables RSA or ECC sign/encrypt operations using a private key stored on a smartcard (such as the YubiKey), through common interfaces like PKCS#11.

YubiKeys support the PIV card interface specified in NIST SP 800-73 document Cryptographic Algorithms and Key Sizes for Personal Identity Verification. PIV enables you to perform RSA or ECC sign/decrypt operations using a private key stored on the smartcard, through common interfaces like PKCS#11. This document contain the library, tools and PKCS#11 module to interact with the hardware functionality.

You can read more about the PIV standards here: PIV Standards

PIV is primarily used for non-web applications. It has built-in support under Windows, and can be used on macOS and Linux via the OpenSC project.

1.2 PIV Tool Design

The Yubico PIV tool was designed to interact with and manage the PIV functions alone. Built on the C `ykpiv` library, the PIV tool provides a CLI to access all of the functionality supported on the PIV function of the YubiKey. While PIV tool allows for the CLI to be used as part of a scripted process, the lack of support beyond the PIV functions means that it is less script-friendly than `ykman`. However, as a purpose built interface on just the `ykpiv` library, the PIV tool is an excellent reference architecture for supporting the YubiKey as a PIV smart card natively.

The PIV tool also provides a PKCS#11 module, called YKCS11, that can be used to expose the YubiKey's smart card functionality to applications that communicate with hard tokens through the PKCS#11 API; applications like OpenSSL, OpenSSH, JAVA, FireFox and the like.

Use the PIV tool when `ykman` does not have a specific command, or when testing the PIV functionality of the YubiKey. On POSIX platforms, PIV tool requires `pcscd` to be pre-installed.

1.3 PIV Usage Guides

For information and examples on what you can do with a PIV enabled YubiKey, see https://developers.yubico.com/PIV/.

1.4 General Information

The default PIN code is `123456`. The default PUK code is `12345678`.

For firmware 5.7 and above: The default AES-192 management key (9B) is `010203040506070801020304050607080102030405060708`.

For firmware 5.4 and below: The default 3DES management key (9B) is `010203040506070801020304050607080102030405060708`.

The following key slots exists:

- 9A, 9C, 9D, 9E: RSA 1024, RSA 2048, ECC secp256r1 or ECC secp384r1 keys (algorithms 6, 7, 11 respectively).
- 9B: Triple-DES key (algorithm 3) for PIV management.

The maximum size of stored objects is 2025/3049 bytes for current versions of YubiKey NEO and YubiKey 4, respectively.

Currently all functionality are available over both contact and contactless interfaces (contrary to what the specifications mandate).

1.5 Software

Card management has been tested with the tools from the OpenSC project, specifically piv-tool, and Yubico PIV software. Basic features should work with any PIV compliant middleware.

https://github.com/OpenSC/OpenSC/wiki

https://developers.yubico.com/yubico-piv-tool/

https://developers.yubico.com/yubikey-piv-manager/

https://github.com/OpenSC/OpenSC/wiki/US-PIV

https://github.com/OpenSC/OpenSC/wiki/PivTool

1.6 License

In general the project is covered by the following BSD license. The file ykcs11/pkcs11.h has additional copyright and licensing information, please see it for more information.

CHAPTER

TWO

SET UP PIV TOOL

2.1 Preparing a YubiKey for Real Use

Change the management key to make sure nobody but you can modify the state of the PIV application on the YubiKey. **Make sure to keep a copy of the key around for later use.**

All of these invocations leaves traces of keys and pins in the command line history. To avoid this, do not include the argument, then the software prompts for the key/pin to be input, and that response is not included in the command line history. For example, when changing the the management key, simply do not include the management key string. Use `-k` in the command, rather than `-k <string>`.

```
key=$(export LC_CTYPE=C; dd if=/dev/urandom 2>/dev/null | tr -d '[:lower:]' | tr -cd
↪'[:xdigit:]' | fold -w48 | head -1)
echo ${key}
yubico-piv-tool -aset-mgm-key -n${key}
```

Change the PIN and PUK, as well.

```
pin=$(export LC_CTYPE=C; dd if=/dev/urandom 2>/dev/null | tr -cd '[:digit:]' | fold -w6␣
↪| head -1)
echo ${pin}
```

```
puk=$(export LC_CTYPE=C; dd if=/dev/urandom 2>/dev/null | tr -cd '[:digit:]' | fold -w8␣
↪| head -1)
echo ${puk}
```

```
yubico-piv-tool -achange-pin -P123456 -N${pin}
yubico-piv-tool -achange-puk -P12345678 -N${puk}
```

2.2 Secure Channel Communication

Some YubiKeys with firmware version 5.7.0 and higher have support for encrypted communicating over a secure channel. Encryption in Yubico PIV Tool is based on the SCP11b protocol. To activate Yubico PIV Tool encryption, add the `--enc` flag in the command line tool or open a connection with the encryption boolean set to `true` when using `libykpiv` during development. After encryption is activated, all subsequent communication over the same session is encrypted. Note that the SCP11b based implementation entails that the communication is encrypted, but the On-Card certificate (`CERT.SD.ECKA`) used to setup the encrypted session is not validated. The feature is used mainly to protect sensitive communication between the host and the YubiKey against monitoring over a medium which could potentially be eavesdropped on, such as NFC.

2.3 Building on POSIX platforms

Either clone from Git or download and unpackage the tarball, then make sure you have the pre-requisites installed and build following the steps below from the `yubico-piv-tool` directory.

Please make sure to have recent versions of the following packages installed on your system.

```
cmake libtool libssl-dev pkg-config check libpcsclite-dev gengetopt
help2man zlib-devel
```

Help2man is used to generate the manpages. Gengetopt version 2.22.6 or later is needed for command line parameter handling. The Vagrant VM has all these dependencies preinstalled.

Please note that these package names are debian based. Other POSIX platforms might have different names. For example, `libssl-dev` can probably be replaced with `openssl-devel` and `libpcsclite-dev` can probably be replaced by `pcsc-lite-devel` on a Redhat platform. Also note that Gengetopt might not be available on all platforms and might need to be built from source. See Gengetop Installation.

After installation of all dependencies, run the following:

```
cd yubico-piv-tool
mkdir build; cd build
cmake ..
make
sudo make install
```

On macOS, you might need to point out homebrew openssl version when running `pkg-config`.

```
PKG_CONFIG_PATH="/usr/local/opt/openssl@1.1/lib/pkgconfig" cmake ..
```

To statically link to OpenSSL (the `libcrypto` library), use the `cmake` option `-DOPENSSL_STATIC_LINK=ON`.

Do not forget you might need to be root for the last command. On Linux it might be needed to update your linked libraries after install.

```
sudo ldconfig
```

The backend to use is decided at compile time, see the summary at the end of the `cmake` output. Use `--with-backend=foo` to choose backend, replacing `foo` with the backend you want to use. The backends available are "pcsc", "macscard" and "winscard" using the PCSC interface, with slightly different shared library linkage and header file names: "pcsc" is used under GNU-like systems, "macscard" under MacOS X, and "winscard" is used under Windows. In most situations, running `cmake` should automatically find the proper backend to use.

2.4 Building on Windows

Building on Windows requires MSBuild or Visual Studio and the MSVC compiler. It also requires building the binaries from the source release package and not from the source checked out from the repository on GitHub. This is because some files that are part of the command line shell are generated but they cannot, currently, be generated on Windows. Those files are, however, included in the source release package.

On Windows, `getopt` is needed to read command line arguments. The easiest way to install `getopt` is with the vcpkg package manager The path to *getopt* DLL library and include file need to be specified as a command line argument to `cmake`. Also the path to OpenSSL needs to be specified either as a command line argument to `cmake` or by setting the environment variable `OPENSSL_ROOT_DIR`.

The command line examples bellow are for `PowerShell` and the prerequisites were installed from source (using `vcpkg`).

```
env:OPENSSL_ROOT_DIR ="PATH/TO/OPENSSL_DIR"
mkdir build; cd build
cmake -A <ARCH> -DGETOPT_LIB_DIR="PATH/TO/GETOPT_DIR/lib" -DGETOPT_INCLUDE_DIR="PATH/TO/
↪GETOPT_DIR/include ..
cmake --build .
```

To run the tests, `check` is used. The path to the `check` directory needs to be specified as a command line argument to `cmake`. Also the path to `check` binaries, `OpenSSL` binaries, `libykpiv.dll` and `libykcs11.dll ``need to be in the ``PATH`.

```
env:OPENSSL_ROOT_DIR ="PATH/TO/OPENSSL_DIR"
mkdir build; cd build
cmake -A <ARCH> -DGETOPT_LIB_DIR="PATH/TO/GETOPT_DIR/lib" -DGETOPT_INCLUDE_DIR="PATH/TO/
↪GETOPT_DIR/include -DCHECK_PATH="PATH/TO/CHECK_DIR" ..
cmake --build .
$env:Path +=";PATH/TO//CHECK_DIR/bin;PATH/TO/OPENSSL_DIR/bin;PATH/TO/build\lib\Debug;
↪PATH/TO/build\ykcs11\Debug"
ctest.exe -C Debug
```

For building on 32 bits system, use `Win32` as ARCH. For building on 64 bits systems, use `x64` as ARCH.

2.5 Coverage

Code coverage is provided courtesy of `lcov` and CMake-codecov This currently only works with `make`.

Enable coverage with

```
cmake -DENABLE_COVERAGE=1 ..
```

You can then build the project normally and run some executables (for example running the tests with `make test`).

At this point coverage evaluation can be generated with `gcov` or `lcov` related targets. For example, this `make` command generates a single HTML report in `./lcov/html/all_targets/index.html`.

```
make lcov
```

2.6 Portability

The main development platform is Debian GNU/Linux. The project compiles on Windows using MSVC and the PCSC backend. It can also be built for MacOS X, also using the PCSC backend.

2.7 Card Holder Unique Identifier

For the application to be usable in windows the object CHUID (Card Holder Unique Identifier) has to be set and unique. The card contents are also aggressively cached so the CHUID has to be changed if the card contents change.

CHAPTER

THREE

PIV TOOL COMMON TASKS

For a list of all available options `--help` can be given. For more information about what's happening `--verbose` can be added to any command. For much more information `--verbose=2` may be used.

3.1 YubiKey Management Related Tasks

3.1.1 Change the management key

Change the management key used for administrative authentication:

```
yubico-piv-tool -aset-mgm-key
```

See *set-mgm-key*.

3.1.2 Display PIV tool version

Display PIV tool version running on the YubiKey:

```
yubico-piv-tool -aversion
```

See *version*.

3.1.3 Reset PIN/PUK retry counter and codes

Default pin 123456, puk 12345678.

```
yubico-piv-tool -k${key} -averify -P${pin} -apin-retries --pin-retries=3 --puk-retries=3
```

See *verify-pin*.

3.1.4 Reset the application after PIN/PUK modified

PIN/PUK need to be blocked hence trying a couple of times — you need to modify this if you have changed the default number of PIN/PUK retries.

```
yubico-piv-tool -averify-pin -P471112
yubico-piv-tool -averify-pin -P471112
yubico-piv-tool -averify-pin -P471112
yubico-piv-tool -averify-pin -P471112
yubico-piv-tool -achange-puk -P471112 -N6756789
yubico-piv-tool -achange-puk -P471112 -N6756789
yubico-piv-tool -achange-puk -P471112 -N6756789
yubico-piv-tool -achange-puk -P471112 -N6756789
yubico-piv-tool -areset
```

See *verify-pin*, *change-puk*, and *reset*.

3.1.5 Set a random chuid

Set a random `chuid`, import a key, and import a certificate from a PKCS12 file, into slot 9c:

```
yubico-piv-tool -s9c -itest.pfx -KPKCS12 -aset-chuid -aimport-key -aimport-cert
```

See *set-chuid*, *import-key*, and *import-certificate*.

3.2 Key Related Tasks

3.2.1 Generate a new private key

```
yubico-piv-tool -k${key} -agenerate -s9c
```

See *generate*.

3.2.2 Generate a new ECC-P256 key

Generate a new ECC-P256 key on device in slot 9a and print the public key on stdout:

```
yubico-piv-tool -s9a -AECCP256 -agenerate
```

See *generate*.

3.2.3 Import a key into slot 85

Import a key into slot 85 and set the touch policy: Both options only available on YubiKey 4 and 5:

```
yubico-piv-tool -aimport-key -s85 --touch-policy=always -ikey.pem
```

See *import-key*.

3.2.4 Run a signature test

Read out the certificate from a slot and then run a signature test:

```
yubico-piv-tool -aread-cert -s9a
yubico-piv-tool -averify-pin -atest-signature -s9a
```

See *read-certificate* and *verify-pin*.

3.2.5 Set the touch policy

Import a key into slot 85 and set the touch policy: Both options only available on YubiKey 4 and 5 or newer:

```
yubico-piv-tool -aimport-key -s85 --touch-policy=always -ikey.pem
```

See *import-key* and *PIV Tool Options*.

3.3 Certificate Related Tasks

3.3.1 Compress a large certificate

Import a large certificate and use `yubico-piv-tool` to apply GZIP compression. Compression is required for certificates larger than 2048 bytes in order to have fit:

```
yubico-piv-tool -s9c -icert.pem --compress -aimport-cert
```

See *import-certificate* and *generate*.

3.3.2 Delete a certificate

Delete a certificate in slot 9a, with management key being asked for:

```
yubico-piv-tool -adelete-certificate -s9a -k
```

See *delete-certificate*.

3.3.3 Generate a certificate request

Generate a certificate request with public key from stdin and print the resulting request on stdout:

```
yubico-piv-tool -s9a -S'/CN=foo/OU=test/O=example.com/' -averify -arequest
```

See *generate*.

3.3.4 Generate a self-signed certificate

Generate a self-signed certificate with public key from stdin and print the certificate, for later import on stdout:

```
yubico-piv-tool -s9a -S'/CN=bar/OU=test/O=example.com/' -averify -aselfsign
```

See *generate*.

3.3.5 Import a certificate

Import a certificate from stdin:

```
yubico-piv-tool -s9a -aimport-certificate
```

See *import-certificate* and *generate*.

3.3.6 Import a large certificate

Import a large certificate that requires compression. Certificates larger than 2048 bytes require compression in order to fit:

```
openssl x509 -in cert.pem -outform DER | gzip -9 > der.gz
yubico-piv-tool -s9c -ider.gz -KGZIP -aimport-cert
```

See *import-certificate*.

3.3.7 Import a large certificate

Import a certificate which is larger than 2048 bytes and have the yubico-piv-tool do the GZIP compression in order to fit:

```
yubico-piv-tool -s9c -icert.pem --compress -aimport-cert
```

See *import-certificate*.

3.3.8 Read out the certificate

Read out the certificate from a slot and then run a signature test:

```
yubico-piv-tool -aread-cert -s9a
yubico-piv-tool -averify-pin -atest-signature -s9a
```

See *read-certificate*, *verify-pin*, and *test-signature*.

3.3.9 Show certificate information

Show some certificate information and some other data:

```
yubico-piv-tool -astatus
```

See *status*.

CHAPTER

FOUR

PIV YKCS11 MODULE

Yubico YKCS11 is a PKCS#11 module that allows external applications to communicate with the PIV application running on a YubiKey.

This module is based on version 2.40 of the PKCS#11 (Cryptoki) specifications. The complete specifications are available at oasis-open.org > PKCS #11 Cryptographic Token Interface Base Specification Version 2.40.

4.1 Building

YKCS11 is automatically built as part of `yubico-piv-tool`. For example:

```
$ mkdir build; cd build
$ cmake ..
$ make
$ sudo make install
```

For additional information about building `yubico-piv-tool`, *Set up PIV Tool*.

After the yuvico-piv-tool is installed, the default location for the ykcs11 module is: `/usr/local/lib/libykcs11.so`. Optionally, you can be built it locally in `yubico-piv-tool/build/ykcs11/libykcs11.so`.

4.2 Portability

The module has been developed and tested using Ubuntu Linux, MacOS, and Windows. Both MacOS and Windows use PCSC as a backend.

4.3 YKCS11 on Windows

After installing yubico-piv-tool using the Windows installer, add the `Yubico PIV Tool\bin` directory to the system path, so other applications can load it. This is also recommended, because the `libykcs11.dll` is dynamically linked to `libykpiv.dll` and `libcrypto-1_1.dll`, and setting the system path enables `ykcs11` to access both of them.

On Windows 10, to set the system path:

1. Go to **Control Panel > System and Security > System > Advanced system setting**.
2. Click **Environment Variables**.
3. Under System Variables, highlight **Path** and click **Edit**.
4. Click **New** and add the absolute path to `Yubico PIV Tool\bin`.

Alternatively, if you do not want to set the system path, copy `libykpiv.dll` and `libcrypto-1_1.dll` into the same directory as the application that needs to access the `ykcs11` module.

4.3.1 A Note for Developers

If `LoadLibrary` is called with an absolute path, it does **not** look for dependencies of the specified DLL in that directory, but rather in the startup directory of the application that calls `LoadLibrary`. The solution is to either:

- Call `LoadLibraryEx` with the flag `LOAD_WITH_ALTERED_SEARCH_PATH` for absolute paths.
- Add the directory where ykcs11 is located to the system PATH.
- Copy the dependencies into the application directory.

Note: Calling `LoadLibraryEx` with the `LOAD_WITH_ALTERED_SEARCH_PATH` flag for a non-absolute path is undefined behavior according to MicroSoft documentation.

For example, Pkcs11Interop sets a variable to `LOAD_WITH_ALTERED_SEARCH_PATH` if the path looks absolute, and `0` otherwise. After that, it always calls `LoadLibraryEx`. This means, if the flag is `0` then `LoadLibraryEx` behaves exactly like `LoadLibrary`.

4.4 Key Mapping

The `ykcs11` module provides access to all 25 keys that can be stored on the YubiKey PIV application. These keys correspond to the keys in the PIV slots as described in PIV Certificate Slots and are accessible through `yubico-piv-tool`.

The mapping is as follows:

ykcs11 id	PIV
1	9a
2	9c
3	9d
4	9e
5 - 24	82 - 95
25	f9

4.5 Key Generation

Key pair generation is a particular operation, in the sense that within PIV this is the only moment where the newly created public key is given back to the user. To prevent the key from being lost, it is automatically stored within the YubiKey by wrapping it in an X.509 certificate.

4.6 Attestation Certificates

Attestation certificates are also accessible with the YKCS11 module. An attestation certificate is a regular X509 Certificates that has the same CKA_ID and public key as the key it is attesting. Attestation certificates, however, are not stored in the YubiKey (CKA_TOKEN is FALSE) and are generated when accessed.

For more details about attestation, see *PIV Tool Attestation* and `Action`, *attest*.

4.7 User Types

YKCS11 defines two types of users: a regular user and a security officer (SO). These are mapped to perform regular tasks for the private key material (PIN-associated operations) and device management (management-key associated operations).

4.8 PINs and Management Key

- The default user PIN for the YubiKey is `123456`.
- The default management key is `010203040506070801020304050607080102030405060708`.

To perform operations involving the private keys, a regular user must be logged in (using the PIN or fingerprint). However, given the different PIN policies for different keys, subsequent operations might require a new login. This is supported by the module through the `CONTEXT_SPECIFIC` user in accordance with the specifications.

4.9 Keys with PIN policy "never"

It is possible to skip PIN verification when using keys with PIN policy "never" by using `VERIFY_NONE` as a PIN.

4.10 Fingerprint Authentication with YubiKey Bio

It is also possible to use the fingerprint reader on the YubiKey Bio to login. Attempting to login with an empty PIN triggers a bio verification. The user is then expected to scan their fingerprint. Note: there might not be an indication on the screen that a fingerprint scan is expected, but the YubiKey blinks while it waits for a fingerprint scan. Bio verification can also be triggered by using `VERIFY_BIO` as a PIN.

4.11 OpenSSL

The YubiKey only supports functions that require an asymmetric private key. Functions that do not, like encryption, signature verification, hashing and generation of a random number, are done by OpenSSL.

Additionally, the YubiKey only performs raw decryption and signature. When padding is used, for example OAEP and PSS padding, applying and removing the padding is handled using OpenSSL functions.

4.12 Testing

Apart from the internal tests, the YKCS11 has also been tested with the Pkcs11Interop .NET library.

4.13 Debugging

By default, the ykcs11 module has debugging disabled. Debugging is *highly* verbose and might be confusing.

To enable debugging of the ykcs11 module, set the variable `YKCS11_DBG` to a numerical value `1` to `9`. The value `0` indicates disabled debugging.

Set `YKCS11_DBG` for debugging, using one of the methods:

- Set the environment variable `YKCS11_DBG`.
- Rebuild the project as follows. The value `2` here is an example:

```
$ mkdir build; cd build
$ cmake .. -DYKCS11_DBG=2
$ make
$ sudo make install
```

Alternatively, use PKCS#11 Spy as provided by OpenSC, to inspect the PKCS#11 communication.

4.14 YKCS11 Functions and Objects

See the following tables of YKCS11 functions and objects.

4.14.1 Supported PKCS#11 Functions

PKCS#11 Function	Mechanism	Comment
C_Initialize		
C_Finalize		
C_GetInfo		
C_GetFunctionList		
C_GetSlotList		
C_GetSlotInfo		
C_GetTokenInfo		
C_GetMechanismList		
C_GetMechanismInfo		
C_InitToken		
C_SetPIN		
C_OpenSession		
C_CloseSession		
C_CloseAllSessions		
C_GetSessionInfo		
C_Login		
C_Logout		

continues on next page

Table 1 – continued from previous page

PKCS#11 Function	Mechanism	Comment
C_CreateObject		With CKO_PRIVATE_KEY or CKO_CERTIFICATE
C_DestroyObject		
C_GetObjectSize		
C_GetAttributeValue		
C_FindObjectsInit		
C_FindObjects		
C_FindObjectsFinal		
C_EncryptInit	CKM_RSA_X_509, CKM_RSA_PKCS, CKM_RSA_PKCS_OAEP	With RSA keys only. Uses OpenSSL encryption functions
C_Encrypt		With RSA keys only. Uses OpenSSL encryption functions
C_EncryptUpdate		With RSA keys only. Uses OpenSSL encryption functions
C_EncryptFinal		With RSA keys only. Uses OpenSSL encryption functions
C_DecryptInit	CKM_RSA_X_509, CKM_RSA_PKCS, CKM_RSA_PKCS_OAEP	With RSA keys only.
C_Decrypt		With RSA keys only.
C_DecryptUpdate		With RSA keys only.
C_DecryptFinal		With RSA keys only.
C_DigestInit	CKM_SHA_1, CKM_SHA256, CKM_SHA384, CKM_SHA512	Uses OpenSSL digest functions
C_Digest		Uses OpenSSL digest functions

continues on next page

Table 1 – continued from previous page

PKCS#11 Function	Mechanism	Comment
C_DigestUpdate		Uses OpenSSL digest functions
C_DigestFinal		Uses OpenSSL digest functions
C_SignInit	CKM_RSA_X_509, CKM_RSA_PKCS, CKM_SHA1_RSA_PKCS, CKM_SHA256_RSA_PKCS, CKM_SHA384_RSA_PKCS, CKM_SHA512_RSA_PKCS, CKM_RSA_PKCS_PSS, CKM_SHA1_RSA_PKCS_PSS, CKM_SHA256_RSA_PKCS_PSS, CKM_SHA384_RSA_PKCS_PSS, CKM_SHA512_RSA_PKCS_PSS, CKM_ECDSA, CKM_ECDSA_SHA1, CKM_ECDSA_SHA224, CKM_ECDSA_SHA256, CKM_ECDSA_SHA384 CKM_EDDSA	
C_Sign		
C_SignUpdate		
C_SignFinal		

continues on next page

Table 1 – continued from previous page

PKCS#11 Function	Mechanism	Comment
C_VerifyInit	CKM_RSA_X_509, CKM_RSA_PKCS, CKM_SHA1_RSA_PKCS, CKM_SHA256_RSA_PKCS, CKM_SHA384_RSA_PKCS, CKM_SHA512_RSA_PKCS, CKM_RSA_PKCS_PSS, CKM_SHA1_RSA_PKCS_PSS, CKM_SHA256_RSA_PKCS_PSS, CKM_SHA384_RSA_PKCS_PSS, CKM_SHA512_RSA_PKCS_PSS, CKM_ECDSA, CKM_ECDSA_SHA1, CKM_ECDSA_SHA224, CKM_ECDSA_SHA256, CKM_ECDSA_SHA384 CKM_EDDSA	Uses OpenSSL verification functions
C_Verify		Uses OpenSSL verification functions
C_VerifyUpdate		Uses OpenSSL verification functions
C_VerifyFinal		Uses OpenSSL verification functions
C_GenerateKeyPair	CKM_RSA_PKCS_KEY_PAIR_GEI CKM_EC_KEY_PAIR_GEN CKM_EC_EDWARDS_KEY_PAIR_ CKM_EC_MONTGOMERY_KEY_I	

4.14.2 Supported PKCS#11 Objects

Not all PKCS#11 Object types are implemented. This is a list of what is implemented and what it maps to.

PKCS#11	Supported CKK	Comment
CKO_PRIVATE_KEY		
	CKK_RSA, CKK_EC, CKK_ED_EDWARDS, CKK_EC_MONTGOMERY	YubiKey 5.7 or later required for: RSA 1024, 2048, 3072, RSA 4096 with e=0x10001, EC with secp256r1 and secp384r1, ED2559, and X25519
CKO_PUBLIC_KEY		
		Stored in X509 Certificates
CKO_CERTIFICATE		
		X509 Certificates containing either the public key or the attestation certificate
CKO_DATA		

4.14.3 Supported Attributes per Object Type

Attribute	Private key object	Public key object	Certificate object	Data object
CKA_CLASS	X	X	X	X
CKA_ID	X	X	X	X
CKA_TOKEN	X	X	X	X
CKA_PRIVATE	X	X	X	X
CKA_LABEL	X	X	X	X
CKA_APPLICATION				X
CKA_OBJECT_ID				X
CKA_MODIFIABLE	X	X	X	X
CKA_COPYABLE	X	X	X	X
CKA_DESTROYABL	X	X	X	X
CKA_VALUE			X	X
CKA_SUBJECT			X	
CKA_ISSUER			X	
CKA_SERIALNUMB			X	
CKA_CERTIFICATE			X	
CKA_TRUSTED		X	X	
CKA_KEY_TYPE	X	X		
CKA_SENSITIVE	X	X		

continues on next page

Table 2 – continued from previous page

Attribute	Private key object	Public key object	Certificate object	Data object
CKA_ALWAYS_SEN	X	X		
CKA_EXTRACTABI	X	X		
CKA_NEVER_EXTF	X	X		
CKA_LOCAL	X	X		
CKA_ENCRYPT	X	X		
CKA_DECRYPT	X	X		
CKA_WRAP	X	X		
CKA_WRAP_WITH_	X	X		
CKA_UNWRAP	X	X		
CKA_SIGN	X	X		
CKA_SIGN_RECOV	X	X		
CKA_VERIFY	X	X		
CKA_VERIFY_REC(	X	X		
CKA_DERIVE	X	X		
CKA_MODULUS	X	X		
CKA_EC_POINT	X	X		
CKA_EC_PARAMS	X	X		
CKA_MODULUS_B	X	X		
CKA_PUBLIC_EXP(	X	X		
CKA_ALWAYS_AUT	X	X		

4.14.4 Key Alias per Slot and Object Type

Some applications, mainly Java, specify the keys to use by their key alias, which is referred to as a key's label by PKCS#11. Objects' labels as access by YKCS11 are fixed values and are unmodifiable. Following is the list of object labels according to their object type and the slot they reside in (See PIV Certificate Slots for the slot usage).

Slot	Private key	Public key	Certificate	Attestation certificate	Data object
9a	Private key for PIV Authentication	Public key for PIV Authentication	X.509 Certificate for PIV Authentication	X.509 Certificate for PIV Attestation 9a	X.509 Certificate for PIV Authentication
9c	Private key for Digital Signature	Public key for Digital Signature	X.509 Certificate for Digital Signature	X.509 Certificate for PIV Attestation 9c	X.509 Certificate for Digital Signature

continues on next page

Table 3 – continued from previous page

Slot	Private key	Public key	Certificate	Attestation certificate	Data object
9d	Private key for Key Management	Public key for Key Management	X.509 Certificate for Key Management	X.509 Certificate for PIV Attestation 9d	X.509 Certificate for Key Management
9e	Private key for Card Authentication	Public key for Card Authentication	X.509 Certificate for Card Authentication	X.509 Certificate for PIV Attestation 9a	X.509 Certificate for Card Authentication
82	Private key for Retired Key 1	Public key for Retired Key 1	X.509 Certificate for Retired Key 1	X.509 Certificate for PIV Attestation 82	X.509 Certificate for Retired Key 1
83	Private key for Retired Key 2	Public key for Retired Key 2	X.509 Certificate for Retired Key 2	X.509 Certificate for PIV Attestation 83	X.509 Certificate for Retired Key 2
84	Private key for Retired Key 3	Public key for Retired Key 3	X.509 Certificate for Retired Key 3	X.509 Certificate for PIV Attestation 84	X.509 Certificate for Retired Key 3
85	Private key for Retired Key 4	Public key for Retired Key 4	X.509 Certificate for Retired Key 4	X.509 Certificate for PIV Attestation 85	X.509 Certificate for Retired Key 4
86	Private key for Retired Key 5	Public key for Retired Key 5	X.509 Certificate for Retired Key 5	X.509 Certificate for PIV Attestation 86	X.509 Certificate for Retired Key 5

continues on next page

Table 3 – continued from previous page

Slot	Private key	Public key	Certificate	Attestation certificate	Data object
87	Private key for Retired Key 6	Public key for Retired Key 6	X.509 Certificate for Retired Key 6	X.509 Certificate for PIV Attestation 87	X.509 Certificate for Retired Key 6
88	Private key for Retired Key 7	Public key for Retired Key 7	X.509 Certificate for Retired Key 7	X.509 Certificate for PIV Attestation 88	X.509 Certificate for Retired Key 7
89	Private key for Retired Key 8	Public key for Retired Key 8	X.509 Certificate for Retired Key 8	X.509 Certificate for PIV Attestation 89	X.509 Certificate for Retired Key 8
8a	Private key for Retired Key 9	Public key for Retired Key 9	X.509 Certificate for Retired Key 9	X.509 Certificate for PIV Attestation 8a	X.509 Certificate for Retired Key 9
8b	Private key for Retired Key 10	Public key for Retired Key 10	X.509 Certificate for Retired Key 10	X.509 Certificate for PIV Attestation 8b	X.509 Certificate for Retired Key 10
8c	Private key for Retired Key 11	Public key for Retired Key 11	X.509 Certificate for Retired Key 11	X.509 Certificate for PIV Attestation 8c	X.509 Certificate for Retired Key 11
8d	Private key for Retired Key 12	Public key for Retired Key 12	X.509 Certificate for Retired Key 12	X.509 Certificate for PIV Attestation 8d	X.509 Certificate for Retired Key 12

continues on next page

Table 3 – continued from previous page

Slot	Private key	Public key	Certificate	Attestation certificate	Data object
8e	Private key for Retired Key 13	Public key for Retired Key 13	X.509 Certificate for Retired Key 13	X.509 Certificate for PIV Attestation 8e	X.509 Certificate for Retired Key 13
8f	Private key for Retired Key 14	Public key for Retired Key 14	X.509 Certificate for Retired Key 14	X.509 Certificate for PIV Attestation 8f	X.509 Certificate for Retired Key 14
90	Private key for Retired Key 15	Public key for Retired Key 15	X.509 Certificate for Retired Key 15	X.509 Certificate for PIV Attestation 90	X.509 Certificate for Retired Key 15
91	Private key for Retired Key 16	Public key for Retired Key 16	X.509 Certificate for Retired Key 16	X.509 Certificate for PIV Attestation 91	X.509 Certificate for Retired Key 16
92	Private key for Retired Key 17	Public key for Retired Key 17	X.509 Certificate for Retired Key 17	X.509 Certificate for PIV Attestation 92	X.509 Certificate for Retired Key 17
93	Private key for Retired Key 18	Public key for Retired Key 18	X.509 Certificate for Retired Key 18	X.509 Certificate for PIV Attestation 93	X.509 Certificate for Retired Key 18
94	Private key for Retired Key 19	Public key for Retired Key 19	X.509 Certificate for Retired Key 19	X.509 Certificate for PIV Attestation 94	X.509 Certificate for Retired Key 19

continues on next page

Table 3 – continued from previous page

Slot	Private key	Public key	Certificate	Attestation certificate	Data object
95	Private key for Retired Key 20	Public key for Retired Key 20	X.509 Certificate for Retired Key 20	X.509 Certificate for PIV Attestation 95	X.509 Certificate for Retired Key 20
f9	Private key for PIV Attestation	Public key for PIV Attestation	X.509 Certificate for PIV Attestation	X.509 Certificate for PIV Attestation f9	X.509 Certificate for PIV Attestation

4.15 YKCS11 Supported Applications

The YKCS11 module was tested with the following applications:

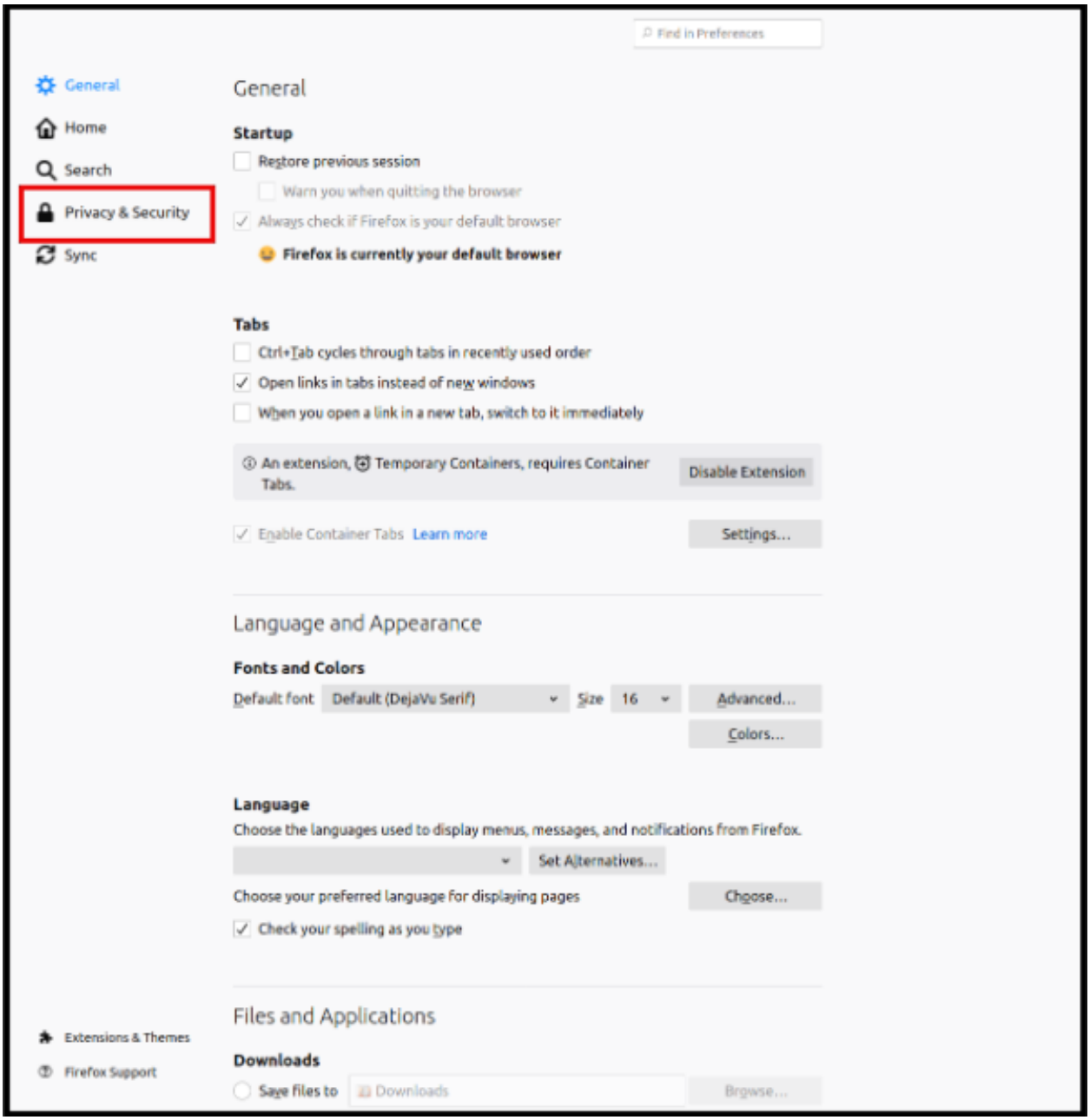
- *FireFox*
- *Fortify*
- *Java Keytool*
- *OpenSC pkcs11-tool*
- *OpenSSH*
- *OpenSSL via PKCS11 Engine*

4.15.1 FireFox

With FireFox, it is possible to authenticate to websites and other web services with certificates stored on a smartcard and accessed through a PKCS#11 module. In order to do that, the PKCS#11 module needs to be added to FireFox as a Security Device. However, in order for FireFox to recognize the certificate in the smartcard, it needs to have previously imported its issuer certificate (typically a CA certificate) as a trusted authority.

4.15.1.1 Importing CA Certificate on FireFox

1. Open FireFox Certificate Manager. Go to **Preferences** → **Privacy & Security** → **View Certificates**.



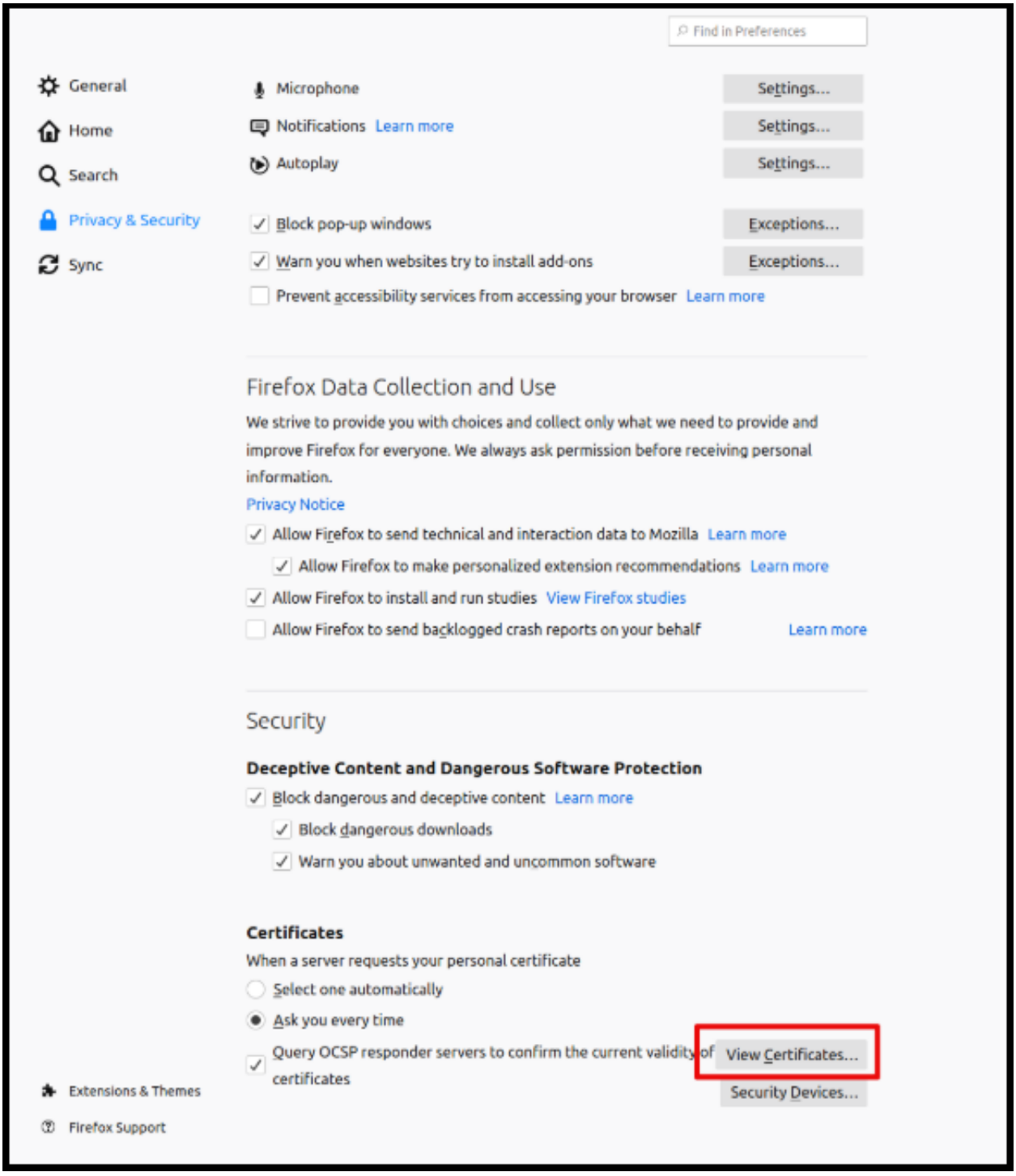
Select Privacy & Security

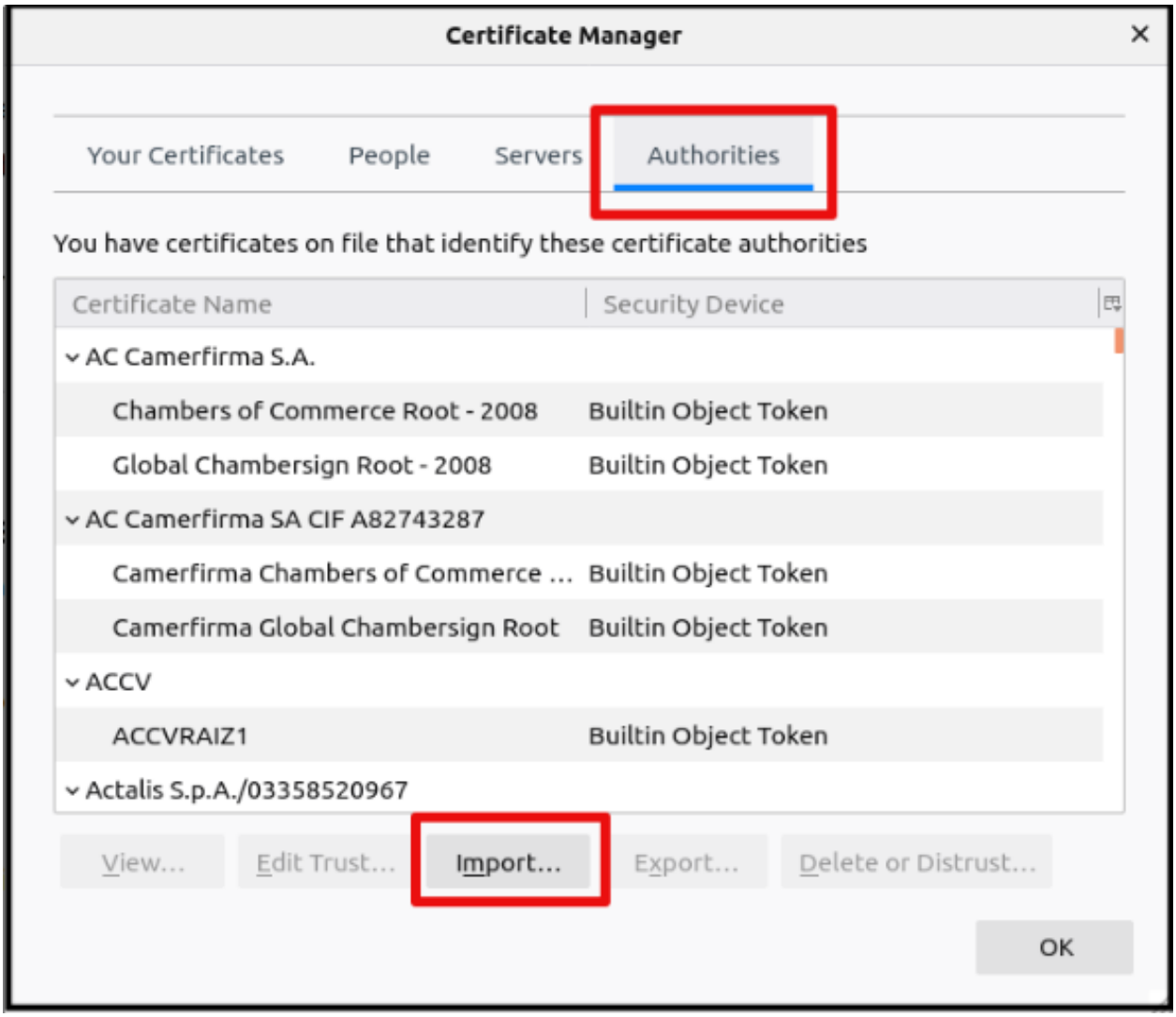
View Certificates

2. Go to **Authorities** and click **Import**.

   Import Authority

3. Navigate to the issuer certificate and choose it.


Find in Preferences
General
Home
Search
Privacy & Security
Sync
Extensions & Themes
Firefox Support
Microphone
Settings...
Notifications Learn more
Settings...
Autoplay
Settings...
Block pop-up windows
Exceptions...
Warn you when websites try to install add-ons
Exceptions...
Prevent accessibility services from accessing your browser Learn more
Firefox Data Collection and Use
We strive to provide you with choices and collect only what we need to provide and improve Firefox for everyone. We always ask permission before receiving personal information.
Privacy Notice
Allow Firefox to send technical and interaction data to Mozilla Learn more
Allow Firefox to make personalized extension recommendations Learn more
Allow Firefox to install and run studies View Firefox studies
Allow Firefox to send backlogged crash reports on your behalf Learn more
Security
Deceptive Content and Dangerous Software Protection
Block dangerous and deceptive content Learn more
Block dangerous downloads
Warn you about unwanted and uncommon software
Certificates
When a server requests your personal certificate
Select one automatically
Ask you every time
Query OCSP responder servers to confirm the current validity of certificates
View Certificates...
Security Devices...


Certificate Manager
Your Certificates
People
Servers
Authorities
You have certificates on file that identify these certificate authorities
Certificate Name
Security Device
AC Camerfirma S.A.
Chambers of Commerce Root - 2008
Builtin Object Token
Global Chambersign Root - 2008
Builtin Object Token
AC Camerfirma SA CIF A82743287
Camerfirma Chambers of Commerce ...
Builtin Object Token
Camerfirma Global Chambersign Root
Builtin Object Token
ACCV
ACCVRAIZ1
Builtin Object Token
Actalis S.p.A./03358520967
View...
Edit Trust...
Import...
Export...
Delete or Distrust...
OK

4.15.1.2 Adding YKCS11 Security Device on FireFox

1. Open FireFox Device Manager. Go to **Preferences** → **Privacy & Security** → **Security Devices**.

   View Security Devices

2. Click **Load**.

   Load Security Devices

3. Choose a name for the YKCS11 module, navigate to libykcs11.so, and choose it, then click OK.

   Security Devices Load Driver

   The YKCS11 module should now appear in the Security devices column on the left.

   Device Manager Listing for Yubico Information

4.15.1.3 Viewing and Downloading a Certificate on FireFox

Caution: If the certificate issuer is not trusted (aka not imported) by FireFox, the certificate details only show an error message.

1. Open FireFox Certificate Manager. Go to **Preferences** → **Privacy & Security** → **View Certificates**.
2. Highlight the certificate and click **View**. The certificate details are displayed in a FireFox tab.

   Certificate Manager View

3. To download the certificate, find the download link in the certificate details tab. It is possible to download only the certificate or the entire certificate chain in PEM format.

   Certificate Information Download

4.15.1.4 Other Functionality on FireFox

With FireFox, it is possible to access and download the certificate. It is also possible to change the PIV user pin with the **Change Password** button in the Certificate Manager.

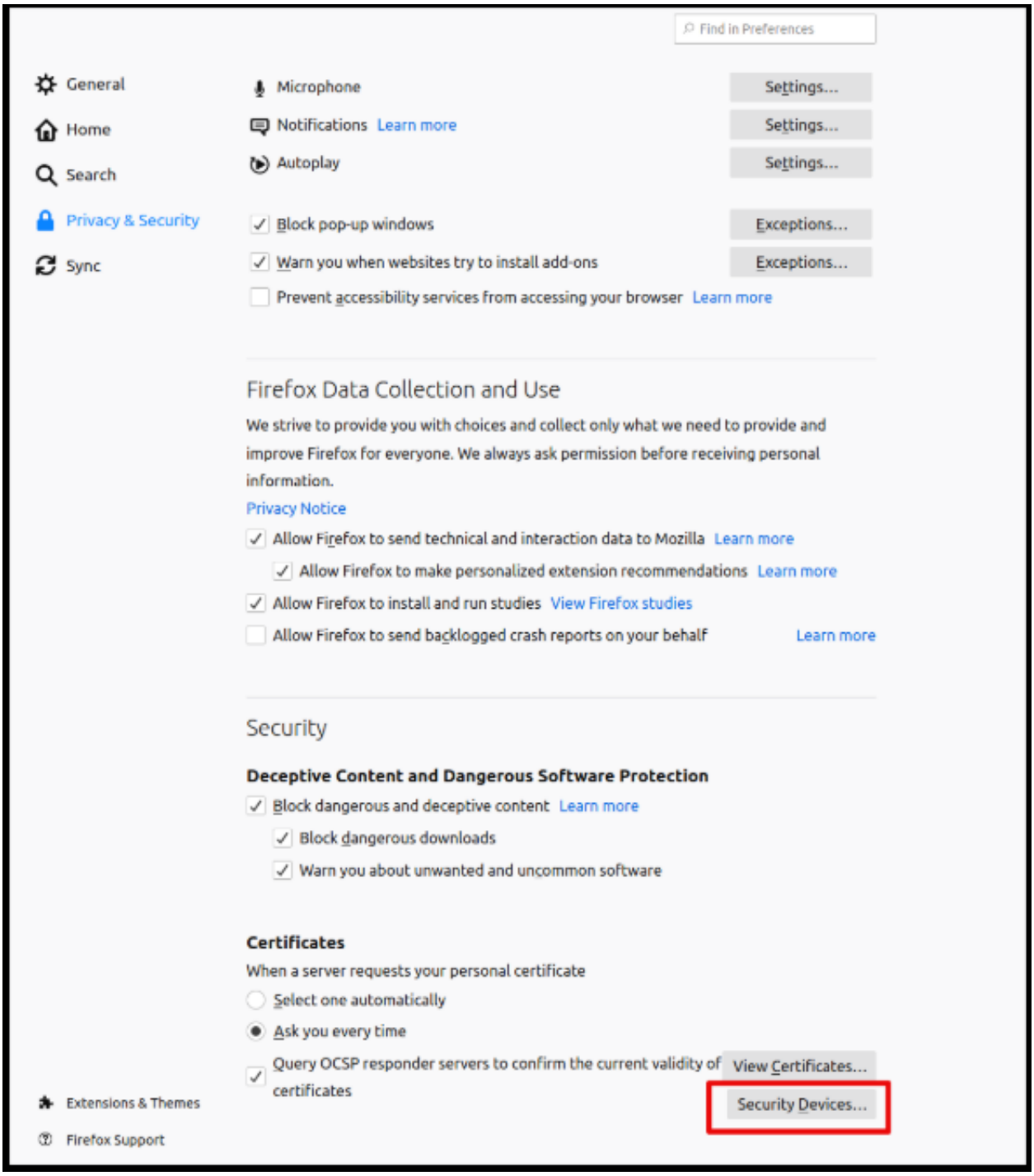
Generally, FireFox has functionality to export the private key (with the **Backup** button in the Certificate Manager). This, however, fails because the YubiKey does not allow the private key to leave it.

Deleting a key through FireFox does not work either unless the user is logged in as an SO user.

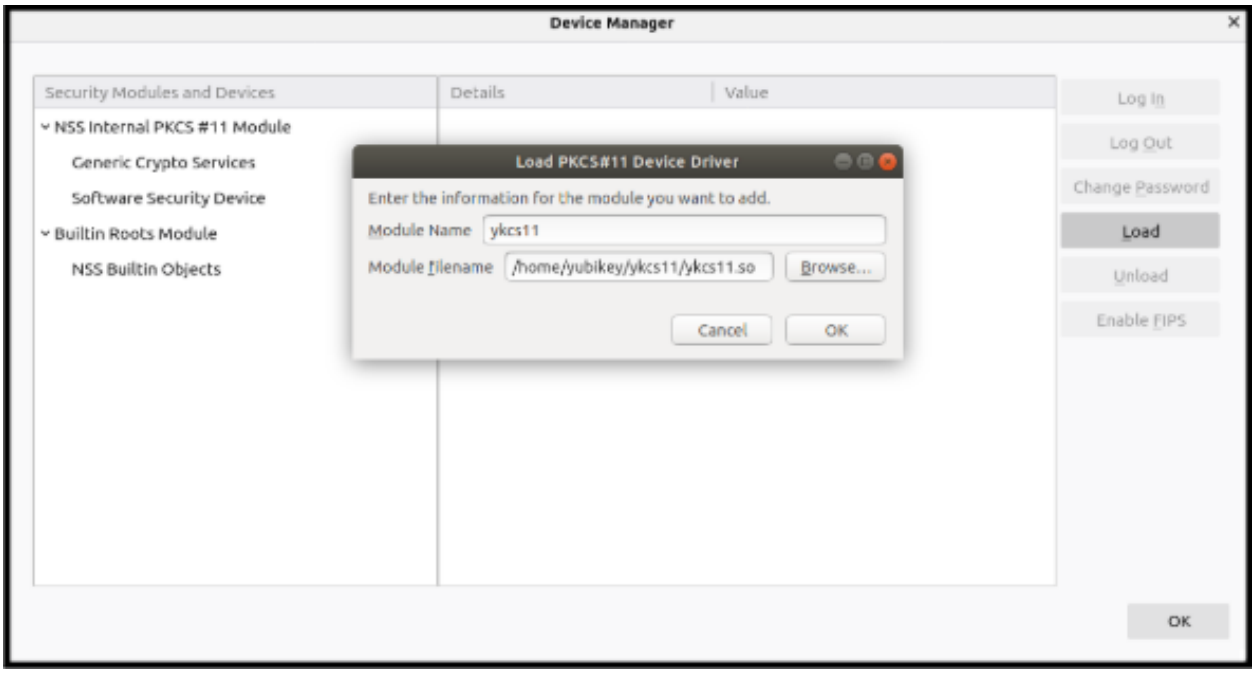
4.15.2 Fortify

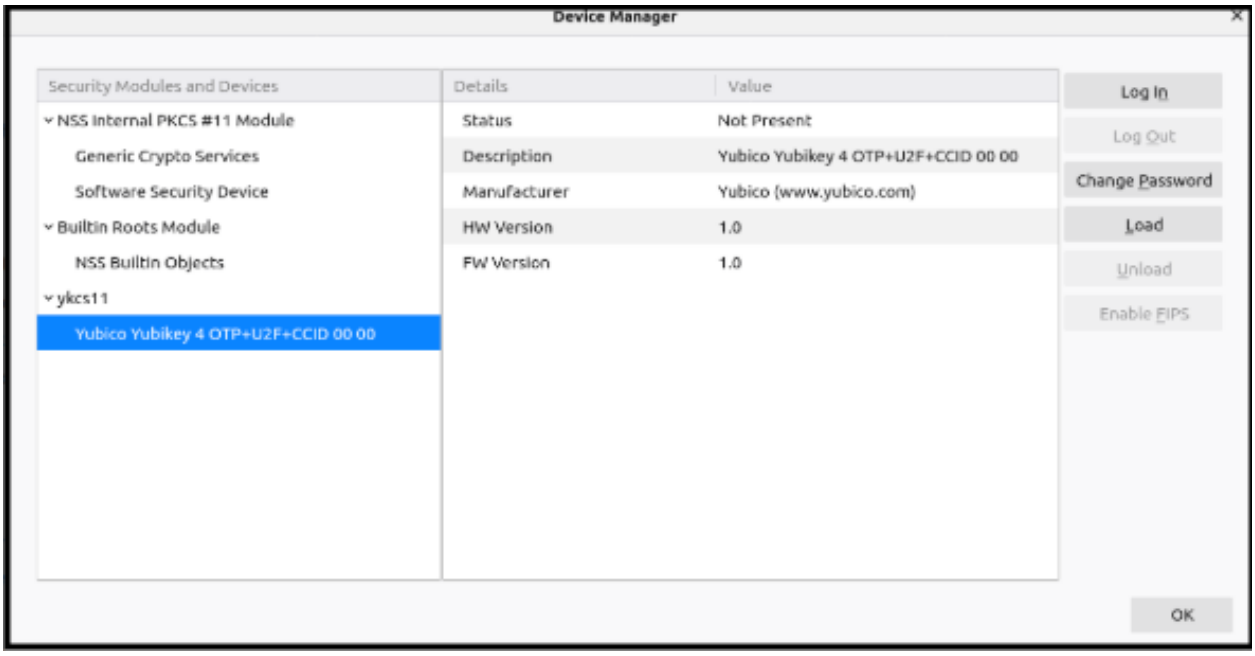
Fortify is a locally installed application that listens on a known TCP port and enables web applications to use smart cards, security tokens, and locally installed certificates. Fortify has native support for YubiKey's PIV interface through the YKCS11 module.

1. Go to fortifyapp.com to download and then install the application specific to your platform.
2. When running the application, a small shield-shaped icon close to the **start menu** should appear.
3. Verify matching codes.


Find in Preferences
General
Home
Search
Privacy & Security
Sync
Microphone
Settings...
Notifications Learn more
Settings...
Autoplay
Settings...
Block pop-up windows
Exceptions...
Warn you when websites try to install add-ons
Exceptions...
Prevent accessibility services from accessing your browser Learn more
Firefox Data Collection and Use
We strive to provide you with choices and collect only what we need to provide and improve Firefox for everyone. We always ask permission before receiving personal information.
Privacy Notice
Allow Firefox to send technical and interaction data to Mozilla Learn more
Allow Firefox to make personalized extension recommendations Learn more
Allow Firefox to install and run studies View Firefox studies
Allow Firefox to send backlogged crash reports on your behalf Learn more
Security
Deceptive Content and Dangerous Software Protection
Block dangerous and deceptive content Learn more
Block dangerous downloads
Warn you about unwanted and uncommon software
Certificates
When a server requests your personal certificate
Select one automatically
Ask you every time
Query OCSP responder servers to confirm the current validity of certificates
View Certificates...
Security Devices...
Extensions & Themes
Firefox Support


Device Manager
Security Modules and Devices
Details
Value
NSS Internal PKCS #11 Module
Generic Crypto Services
Software Security Device
Builtin Roots Module
NSS Builtin Objects
Log In
Log Out
Change Password
Load
Unload
Enable FIPS
OK


Device Manager
Security Modules and Devices
Details
Value
NSS Internal PKCS #11 Module
Generic Crypto Services
Software Security Device
Builtin Roots Module
NSS Builtin Objects
Load PKCS#11 Device Driver
Enter the information for the module you want to add.
Module Name
ykcs11
Module filename
/home/yubikey/ykcs11/ykcs11.so
Browse...
Cancel
OK
Log In
Log Out
Change Password
Load
Unload
Enable FIPS
OK



a. Click the Fortify icon and go to **Tools**. A browser opens with the Fortify webapp interface.

   Fortify Tools Menu

b. You are presented with a code inside the webpage and a code outside of it. If the codes match, click **Approve**.

   Fortify Interface Verification

4. View stored YubiKey certificates. In the Fortify web interface:

   a. Select the Yubico provider.

      Fortify Select Provider

   b. At the prompt, enter the PIN of the PIV user.

      Fortify PIN

   c. In the panel, view the list of certificates stored in the YubiKey.

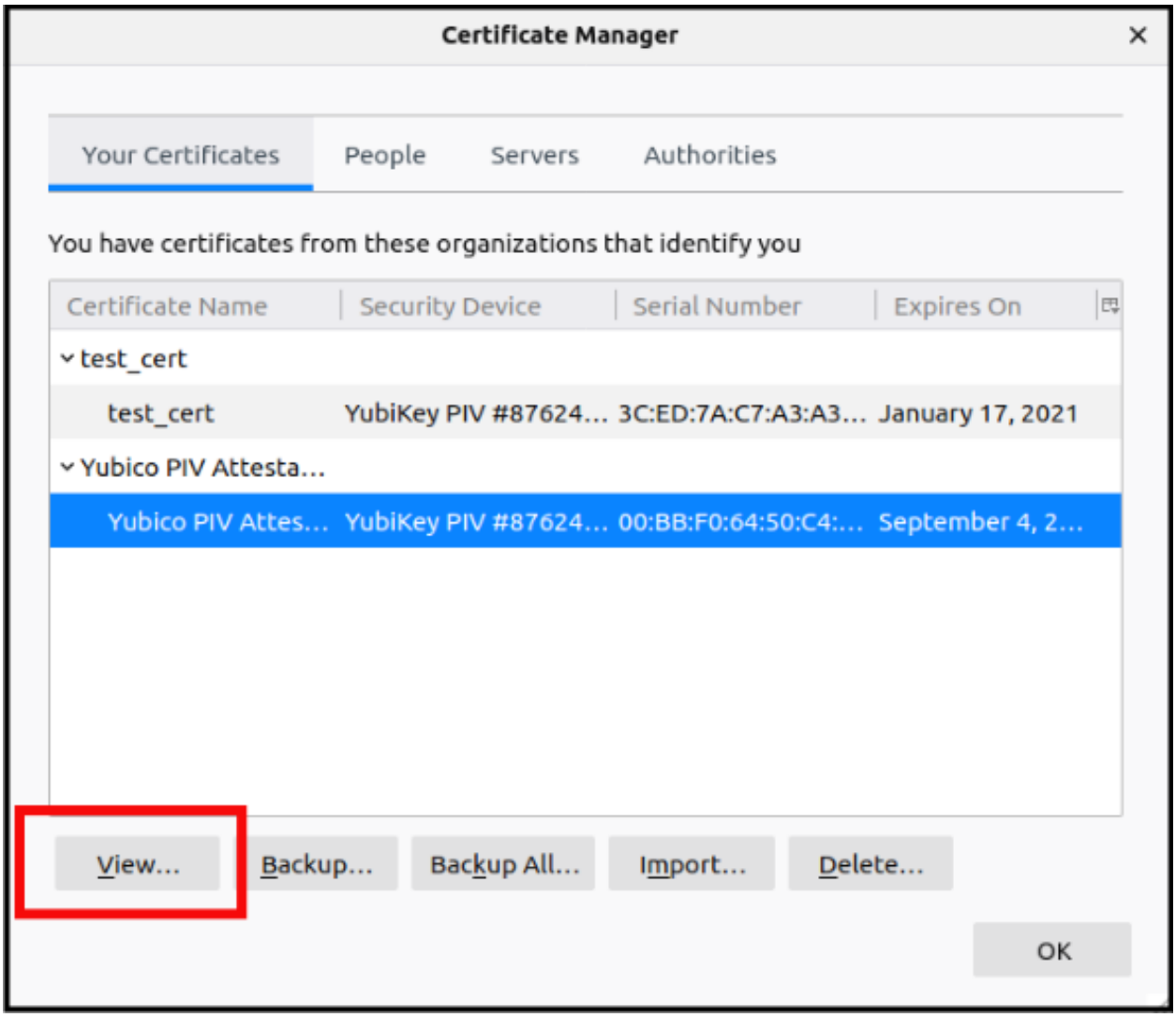
      Fortify View Certificates

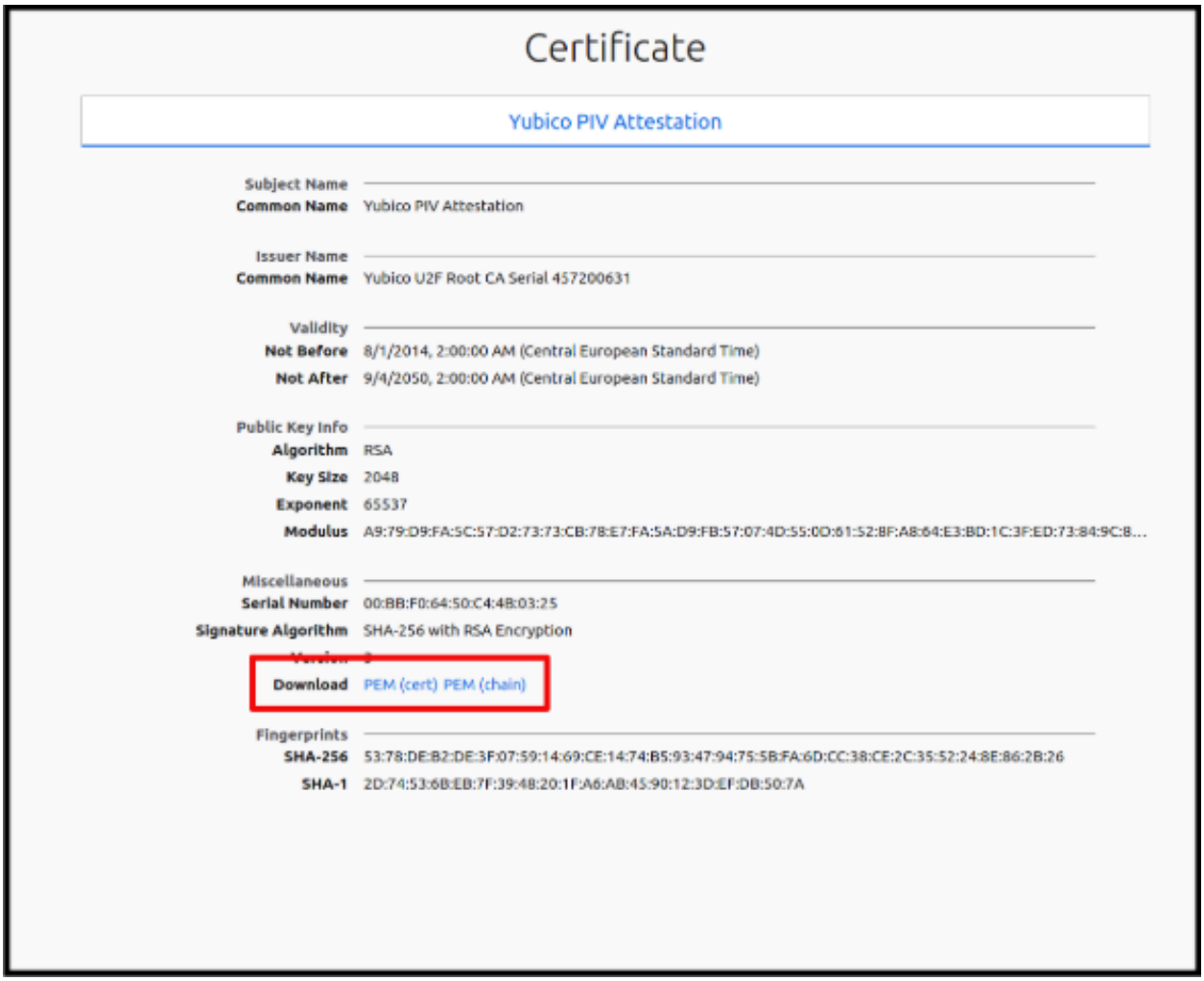
4.15.2.1 Tips for using Fortify

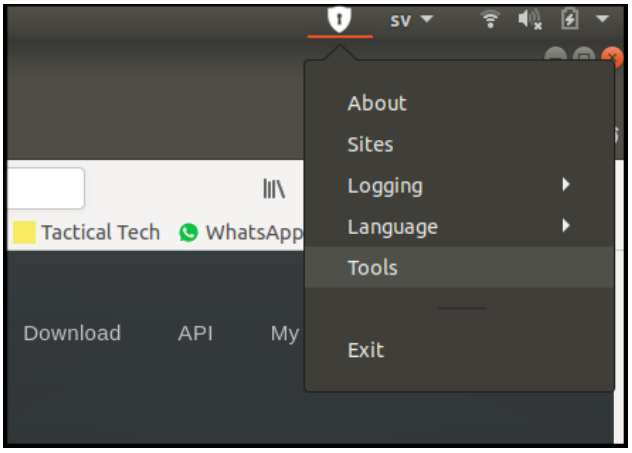
Fortify expects to find the YKCS11 module in the following locations:

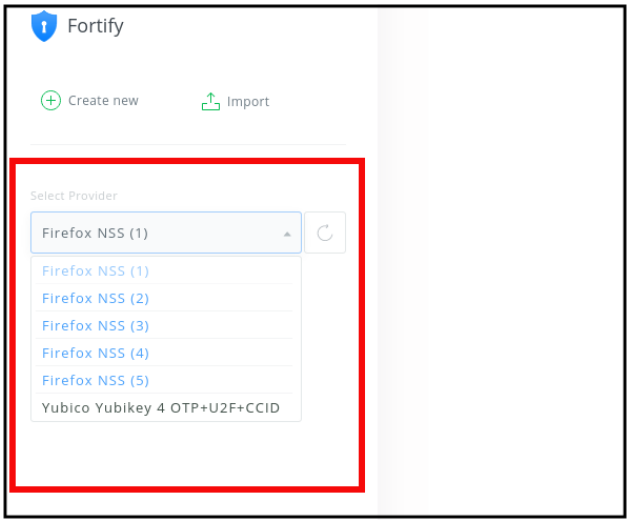
- MacOS: `/usr/local/lib/libykcs11.dylib`
- Linux: `/usr/local/lib/libykcs11.so`
- Windows: `%WINDIR/System32/libykcs11-1.dll`

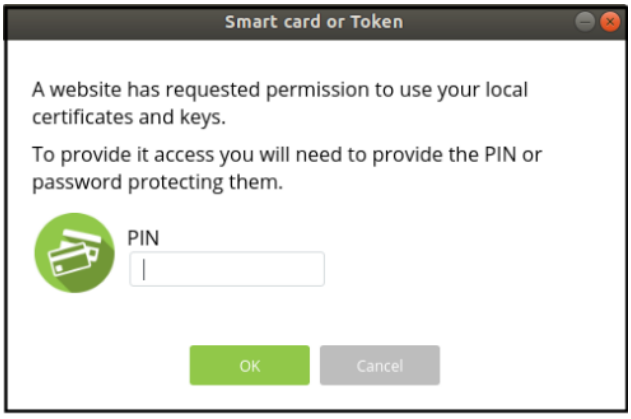
The paths are specified in a file called `card.json`. On a Linux system, this file might reside in `$HOME/.fortify/card.json`.

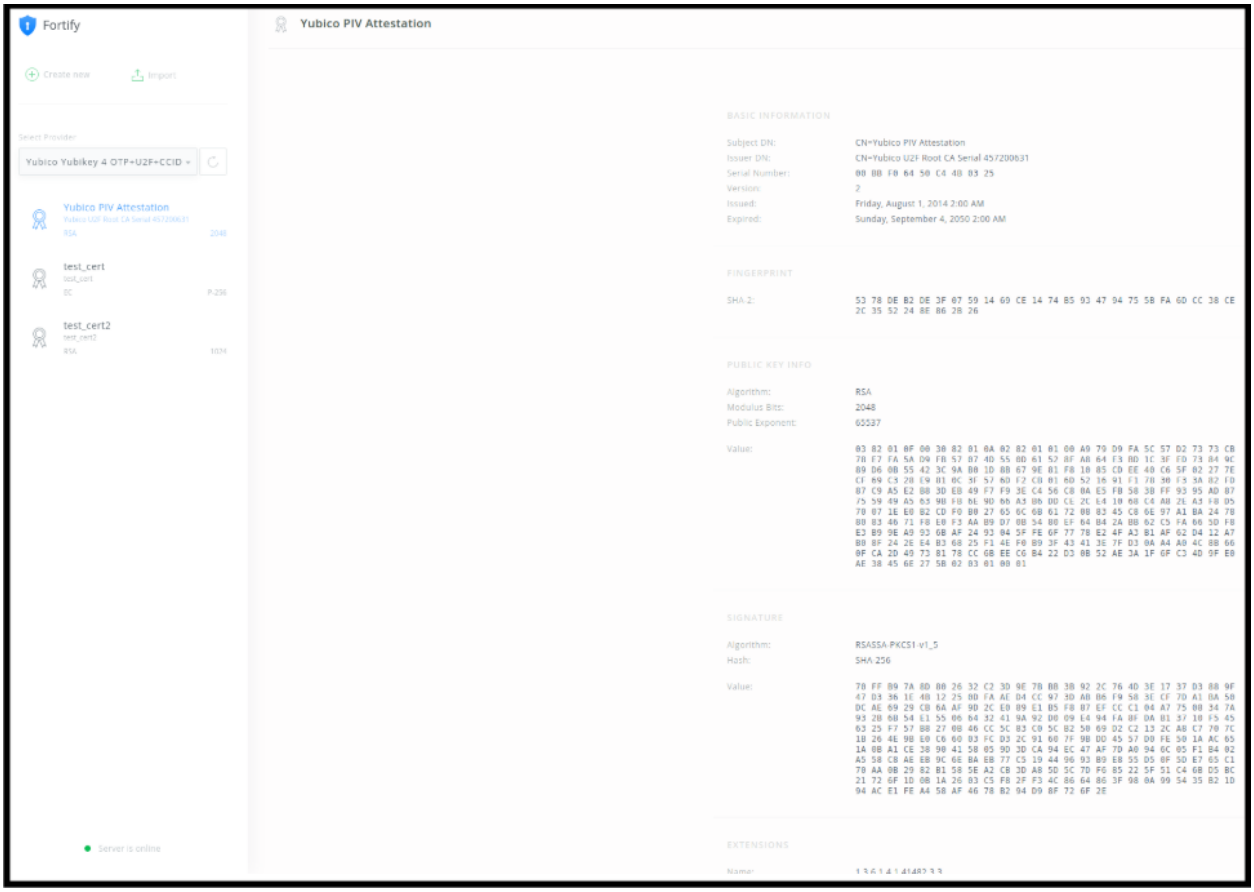

Certificate Manager
Your Certificates
People
Servers
Authorities
You have certificates from these organizations that identify you
Certificate Name
Security Device
Serial Number
Expires On
test_cert
test_cert
YubiKey PIV #87624...
3C:ED:7A:C7:A3:A3...
January 17, 2021
Yubico PIV Attesta...
Yubico PIV Attes...
YubiKey PIV #87624...
00:BB:F0:64:50:C4:...
September 4, 2...
View...
Backup...
Backup All...
Import...
Delete...
OK


Certificate
Yubico PIV Attestation
Subject Name
Common Name Yubico PIV Attestation
Issuer Name
Common Name Yubico U2F Root CA Serial 457200631
Validity
Not Before 8/1/2014, 2:00:00 AM (Central European Standard Time)
Not After 9/4/2050, 2:00:00 AM (Central European Standard Time)
Public Key Info
Algorithm RSA
Key Size 2048
Exponent 65537
Modulus A9:79:D9:FA:5C:57:D2:73:73:CB:78:E7:FA:5A:D9:FB:57:07:4D:55:0D:61:52:8F:A8:64:E3:BD:1C:3F:ED:73:84:9C:8...
Miscellaneous
Serial Number 00:BB:F0:64:50:C4:4B:03:25
Signature Algorithm SHA-256 with RSA Encryption
Download PEM (cert) PEM (chain)
Fingerprints
SHA-256 53:78:DE:B2:DE:3F:07:59:14:69:CE:14:74:B5:93:47:94:75:5B:FA:6D:CC:38:CE:2C:35:52:24:8E:86:2B:26
SHA-1 2D:74:53:6B:EB:7F:39:48:20:1F:A6:AB:45:90:12:3D:EF:DB:50:7A


sv
About
Sites
Logging
Language
Tools
Exit
Tactical Tech
WhatsApp
Download
API
My


Fortify
Create new
Import
Select Provider
Firefox NSS (1)
Firefox NSS (1)
Firefox NSS (2)
Firefox NSS (3)
Firefox NSS (4)
Firefox NSS (5)
Yubico Yubikey 4 OTP+U2F+CCID


Smart card or Token
A website has requested permission to use your local certificates and keys.
To provide it access you will need to provide the PIN or password protecting them.
PIN
OK
Cancel


Fortify
Create new
Import
Select Provider:
Yubico Yubikey 4 OTP+U2F+CCID
Yubico PIV Attestation
test_cert1
test_cert2
Server is online
Yubico PIV Attestation
BASIC INFORMATION
Subject DN:
CN=Yubico PIV Attestation
Issuer DN:
CN=Yubico U2F Root CA Serial 457200631
Serial Number:
Version:
Issued:
Friday, August 1, 2014 2:00 AM
Expired:
Sunday, September 4, 2050 2:00 AM
FINGERPRINT
SHA-2:
PUBLIC KEY INFO
Algorithm:
RSA
Modulus Bits:
2048
Public Exponent:
65537
Value:
SIGNATURE
Algorithm:
RSASSA-PKCS1-v1_5
Hash:
SHA-256
Value:
EXTENSIONS
Name:

4.15.3 Java Keytool

The YKCS11 module can be used with Java keytool through the SunPKCS11 provider that is shipped with Java by default. To configure the SunPKCS11 provider to use the YKCS11, create a configuration file, let's call it `sun_ykcs11.conf`, with the following content:

```
name = ykcs11
library = /path/to/libykcs11.so
```

The name is an arbitrary string.

This configuration file should then be used as the provider's argument parameter in the command line.

4.15.3.1 List Content

```
$ keytool -list -keystore NONE -storetype PKCS11
-providerClass sun.security.pkcs11.SunPKCS11 -providerArg /path/to/sun_ykcs11.conf
```

Where –

`-keystore NONE` indicates that the keys are not stored in a soft token, aka not a file.

`-storetype PKCS11` indicates that the keys are accessible through the PKCS#11 interface.

4.15.3.2 Signing a JAR File

To sign a jar file using `jarsigner`, specify the alias of the signing key. The aliases of the keys stored on the YubiKey PIV are fixed and unmodifiable. To view the list of key aliases:

- Use the YubiKey command, `keytool -list`, shown in Example 1 above.
- See *YKCS11 Functions and Objects*.

Example of signing a jar file with the key on slot 9c:

```
$ jarsigner -keystore NONE -storetype PKCS11
-providerClass sun.security.pkcs11.SunPKCS11
-providerArg /path/to/sun_ykcs11.conf lib.jar "X.509 Certificate for Digital Signature"
```

To display PKCS11 debug messages, add the parameter `-J-Djava.security.debug=sunpkcs11`:

```
$ jarsigner -verify -keystore NONE -storetype PKCS11
-providerClass sun.security.pkcs11.SunPKCS11
-providerArg /path/to/sun_ykcs11.conf lib.jar "X.509 Certificate for Digital Signature"
-J-Djava.security.debug=sunpkcs11\
```

The jarsigner command returns two files appearing under META-INF inside the jar file. One of them has the ending `.SF` and the other has the ending `.EC` or `.RSA` depending on the key type of the signing key.

To verify the signature of a jar file, run:

```
$ jarsigner -verify -keystore NONE -storetype PKCS11
-providerClass sun.security.pkcs11.SunPKCS11
-providerArg /path/to/sun_ykcs11.conf lib.jar "X.509 Certificate for Digital Signature"
```

4.15.4 OpenSC pkcs11-tool

The YKCS11 module works well with `pkcs11-tool`. Be aware though that older versions of OpenSC (like the ones available on Linux distributions) might produce errors when running some commands. If so, try again after installing a newer version.

4.15.4.1 Display Device Info

```
$ pkcs11-tool --module /path/to/libykcs11.so --show-info
```

4.15.4.2 Key Generation

Only the SO user is permitted to generate keys. The SO's PIN is the PIV management key.

- Generate an EC key in slot 9A using curve `secp384r1`

```
$ pkcs11-tool --module /path/to/libykcs11.so --login
--login-type so --keypairgen --id 1 --key-type EC:secp384r1
```

- Generate an EC key in slot 9E using curve `prime256v1`

```
$ pkcs11-tool --module /path/to/libykcs11.so --login
--login-type so --keypairgen --id 4 --key-type EC:prime256v1
```

- Generate an RSA key in slot 9C

```
$ pkcs11-tool --module /path/to/libykcs11.so --login
--login-type so --keypairgen --id 2 --key-type rsa:2048
```

4.15.4.3 Signing

Signatures generated using `pkcs11-tool` can be verified using, for example, `openssl`.

To verify the signature with `openssl`, the public key needs to be extracted from the certificate. To do that, complete the steps:

1. Export the certificate from the YubiKey using the YubiKey Manager, `ykman`, `yubico-piv-tool`, FireFox or any other available tool.
2. If the certificate is not in `PEM` format, convert it into `PEM` format.
3. Extract the public key from the certificate. Run the following command:

```
$ openssl x509 -in cert.pem -pubkey -noout > pubkey.pem
```

The following are a few command line examples of signing data with `pkcs11-tool` and verifying the signature with `openssl`.

- Sign data with an RSA key in slot 9E.

```
$ pkcs11-tool --module /path/to/libykcs11.so --sign --id 4 -i data.txt -o data.sig
$ openssl rsautl -verify -in data.sig -inkey 9e_pubkey.pem -pubin
```

- Sign data with an RSA key in slot 9C and SHA256.

```
$ pkcs11-tool --module /path/to/libykcs11.so --sign
-m RSA-SHA256 --id 2 -i data.txt -o data.sig
$ openssl dgst -sha256 -verify 9c_pubkey.pem
-signature data.sig data.txt
```

- Signing data with an EC key in slot 9A and SHA1.

```
$ pkcs11-tool --module /path/to/libykcs11.so --sign --id 1
-m ECDSA-SHA1 --signature-format openssl -i data.txt
-o data.sig
$ openssl dgst -sha1 -verify 9a_pubkey.pem -signature data.sig data.txt
```

4.15.4.4 Testing RSA Keys

At least one RSA key needs to already exist in the YubiKey for this test to work.

```
$ pkcs11-tool --module /path/to/libykcs11.so --login  --test
```

4.15.4.5 Testing EC Keys

With the default installation of the YubiKey's PIV, testing EC keys works only on slot 9C. This is because `pkcs11-tool --test-ec` assumes that the same user can both generate a keypair and sign data. This, however, is not allowed by the YubiKey, which implements separation of duty more strictly. By default, however, the key that resides on slot 9C has its `CKA_ALWAYS_AUTHENTICATE` attribute set to True, which prompts the user for the PIN during the different operations, and so the right PIN can be entered at the right time.

```
$ pkcs11-tool --module /path/to/libykcs11.so --login
--login-type so --test-ec --id 2 --key-type EC:secp256r1
```

4.15.5 OpenSSH

Keys stored on the YubiKey can be used to login to remote SSH servers. This can be done either with the YubiKey PIV application using, for example, OpenSSH, or with the YubiKey PGP application.

Following are some command line examples of using OpenSSH with the Yubikey PIV application through YKCS11. For guides on how to use the YubiKey PGP application with SSH, see SSH authentication.

To direct OpenSSH to use keys stored on the YubiKey, specify the YKCS11 module with the `-I` option in the command.

```
$ ssh -I /path/to/libykcs11.so git@github.com
```

To download the public key accessible through the YKCS11 module:

```
$ ssh-keygen -D /path/to/libykcs11.so
```

4.15.6 OpenSSL via PKCS11 Engine

The easiest way to get OpenSSL to work with YKCS11 via `engine_pkcs11` is by using the `pll-kit` proxy module.

To get the OpenSSL PKCS11 engine to use the Yubico YKCS11 module specifically, set the environment variable `PKCS11_MODULE_PATH` to point to `libykcs11.so` module.

For more details on PKCS11 engine, see OpenSC libp11.

For more details on how to configure OpenSSL PKCS11 engine for Yubico supported modules, see OpenSSL with YubiHSM 2 via engine_pkcs11 and yubihsm_pkcs11.

4.15.6.1 Data Signing with OpenSSl

```
$ PKCS11_MODULE_PATH=/path/to/libykcs11.so openssl rsautl -engine pkcs11 -keyform engine␣
↪-inkey "pkcs11:object=Private key for PIVAuthentication;type=private" -sign -in data.
↪txt -out data.sig
```

Where – `object=Private key for PIVAuthentication` specifies the alias (or label) of the private key to be used. The aliases of the keys stored on the YubiKey PIV are fixed and unmodifiable. See *YKCS11 Functions and Objects*.

```
type=private specifies that the type of the object is a private key (CKA_CLASS = CKO_
↪PRIVATE_KEY)
```

For more parameters to specify keys see RFC7512.

Caution: Be aware that the order of the parameters in the command line can be important.

CHAPTER

FIVE

PIV TOOL ATTESTATION

This feature is only available in YubiKey 4.3 and above.

5.1 What is Attestation

The YubiKey is able to create an attestation statement in the form of an X.509 certificate. This provides evidence that a certain key was generated on a YubiKey. The certificate can be validated up to Yubico Root CA to prove authenticity and validity. The returned attestation statement is in the form of a PEM encoded X.509 certificate, signed by a key stored in PIV slot f9 on the YubiKey.

For more information on attestation, see Yubico content:

- Developer documentation, PIV attestation
- .NET YubiKey SDK: User's Manual, PIV attestation statements

5.2 Getting and Verifying Attestation Certificates

Each YubiKey comes with a pre-loaded key and certificate. The certificate is signed by Yubico Root Certificate Authority (CA). The pre-loaded key and certificate can be replaced by overriding the content of the slot.

Important: If the pre-loaded Yubico factory-issued key or certificate is overwritten, it cannot be restored - even a factory reset does not recover the data.

The OpenSSL method listed below is an example for verifying the certificate and validating with the Yubico Root CA. This method is for testing the concept of attestation verification only.

Important: Yubico recommends using production level evaluation for production verification.

The following steps assume that the pre-loaded certificate and key in PIV slot f9 have not been overwritten. If they have been overwritten, you need to replace the certificate chain with your own - the one you used for the new key and certificate.

To get and verify an attestation statement:

1. Get and install Yubico PIV Tool for your platform.

   Make note of the folder/directory where you install the PIV tool. For example, on Windows that folder might be: `C:\Program Files\Yubico\Yubico PIV Tool\bin`.

The Yubico PIV tool launches automatically launches when your run a `yubico-piv-tool` command. For example:

```
yubico-piv-tool --version
```

When you run the `yubico-piv-tool`, make sure the your platform can access and understand the command:

- Add the install directory to PATH or navigate to the installation folder/directory.
- You might need to add the `./` prefix to the command. This tells the system to run the command from the current folder/directory.
- If you are using PowerShell, add the file extension, `.exe`, to the PIV tool executable, `yubico-piv-tool.exe`.

2. Get an attestation statement (X.509 certificate) for a slot.

```
yubico-piv-tool --action=attest --slot=9a --out <PATH-attestation>\
↪Slot9aAttestation.pem
```

Where –

<PATH-attestation> is where the command stores the attestation statement.

9a is an example for the slot on the YubiKey that contains the key you generated and want to attest. See `--slot` command in *PIV Tool Options* for possible slots.

Note: The attestation fails when there is no key in the designated slot (slot 9a) or if the key in the slot was imported. Error message: `Failed to attest data`.

---

3. Get the intermediate CA from slot `f9` of the YubiKey.

```
yubico-piv-tool --action=read-certificate --slot=f9 --out  <PATH-intermediate>\
↪SlotF9Intermediate.pem
```

Where –

<PATH-intermediate> is where the command stores the intermediate CA.

f9 is the slot on the YubiKey with the pre-loaded intermediate certificate. See `--slot` command in *PIV Tool Options* for possible slots.

4. Determine the firmware version of your YubiKey.

```
yubico-piv-tool --action=version
```

5. Download the certificate(s) appropriate for your YubiKey firmware version.

- For pre-5.7.4 firmware, download the root certificate:

  Yubico PIV Root CA Serial 263751

- For 5.7.4 or newer firmware, download the certificate chain:

  Yubico Attestation Root 1 and Intermediate Certificates

Note:

- Record where the files are stored.
- Make sure the files are saved with the correct file extension `.pem` (and not .pem.txt or .txt)!

6. Download and install openssl for your platform.

   See the openssl documentation for details.

Note: These commands don't work with OpenSSL 1.1.0 on YubiKey 4 series products. To verify certificate chains for such devices, see PIV Attestation Verification Fails with OpenSSL 1.1.0.

7. Verify the attestation certificate using the command appropriate for your YubiKey firmware version.

   - For pre-5.7.4 firmware

```
openssl verify -CAfile <PATH-certs>\yubico-piv-ca-1.pem -untrusted <PATH-
↪intermediate>\SlotF9Intermediate.pem <PATH-attestation>\Slot9aAttestation.pem
```

   - For 5.7.4 and newer firmware

```
openssl verify -CAfile <PATH-certs>\yubico-ca-1.pem -untrusted <PATH-
↪intermediate>\yubico-intermediate.pem -untrusted <PATH-intermediate>\
↪SlotF9Intermediate.pem <PATH-attestation>\Slot9aAttestation.pem
```

   Where – <PATH-certs>, <PATH-intermediate>, and <PATH-attestation> are the locations of the stored Yubico CA, intermediate, and attestation .pem files identified in Step 1 to Step 5.

   Expected result:

```
Slot9Aattestation.pem: OK
```

CHAPTER

SIX

PIV TOOL COMMAND, OPTIONS AND ACTIONS

6.1 yubico-piv-tool [Option] ...

Use the PIV tool command options.

6.1.1 PIV Tool Options

Option	Description	Possible Values and/or Default
`-h`, `--help`	Print help and exit	

`-a`, `--action ENUM`	Action to take.
	
	See *PIV Tool action Command Parameters* for descriptions of actions.
		`attest`, `change-pin`, `change-puk`, `delete-certificate`, `delete-key`, `generate`, `import-certificate`, `import-key`, `list-readers`, `move-key`, `pin-retries`, `read-certificate`, `read-object`, `read-public-key`, `request-certificate`, `reset`, `selfsign-certificate`, `set-ccc`, `set-chuid`, `set-mgm-key`, `sign-data`, `status`, `test-decipher`, `test-signature`, `unblock-pin`, `verify-bio`, `verify-pin`, `version`, `write-object`
		
		Multiple actions may be given at once and are executed in order, for example:
		`--action=verify-pin`
		`--action=request-certificate`

continues on next page

Table 1 – continued from previous page

Option	Description	Possible Values and/or Default
-A, --algorithm ENUM	The algorithm to use to generate the key pair.	RSA1024, RSA2048, ECCP256, ECCP384 Values that require YubiKey 5.7 or newer: RSA3072, RSA4096, ED25519, X25519 Default: RSA2048
--attestation	Add attestation cross-signature.	Default: off
--compress	Compress a large certificate using GZIP before import.	Default: off
--enc	Communication with the YubiKey is done over an encrypted channel.	Default: off
-f, --format ENUM	Format of data for write/read object.	hex, base64, binary Default: hex
-full-help	Print help, including hidden options, and exit.	
-global	Reset the whole device over all applications, including the PIV application.	Default: off
-H, --hash ENUM	Hash to use for signatures.	SHA1, SHA256, SHA384, SHA512. Default: SHA256

continues on next page

Table 1 – continued from previous page

Option	Description	Possible Values and/or Default
`-i`, `--input STRING`	Filename to use as input. If left out, input is read from `stdin`.	None for `stdin` or filename. Default: `-` for stdin The only supported format for public key is PEM.
`--id INT`	The ID of the object to write/read according to PIV Specifications	
`-k`, `--key STRING`	Management key to use. If no value is specified, PIV tool prompts for value	Default: See Note **2**
`-K`, `--key-format ENUM`	Format of the key being read/written.	`PEM`, `PKCS12`, `GZIP`, `DER`, `SSH` Default: `PEM`
`-m`, `--new-key-algo ENUM`	New management key algorithm to use for the action, `set-mgm-key`.	`AES128`, `AES192`, `AES256`, `TDES` Default: `TDES`
`-n`, `--new-key STRING`	New management key to use for the action, `set-mgm-key`. If omitted, PIV tool prompts for key.	
`-N`, `--new-pin STRING`	New PIN/PUK code changing to. If omitted, PIV tool prompts for PIN/PUK.	

continues on next page

Table 1 – continued from previous page

Option	Description	Possible Values and/or Default
`-o, --output STRING`	Filename to use as output or certificate file. If not specified, output is printed to `stdout`.	None for `stdout` or filename. Default: `-` for `stdout`
`-p, --password STRING`	Password for decryption of private key file. If omitted, PIV tool prompts for password	
`-P, --pin STRING`	PIN/PUK code for verification. If omitted, PIV tool prompts for PIN/PUK	
`--pin-policy ENUM`	Set pin policy for actions: `generate` or `import-key`. Only available on YubiKey 4 or newer.	Values PIN key verification: `never`, `once`, `always` Value BIO key verification: `matchonce`, `matchalways` Default: slot 9c, `always` slot 9a, 9d and 9e, `once` remaining slots, `never`
`--pin-retries INT`	Number of retries before the PIN code is blocked.	
`--puk-retries INT`	Number of retries before the PUK code is blocked.	
`-r, --reader STRING`	Only use a matching reader.	Default: `Yubikey`

continues on next page

Table 1 – continued from previous page

Option	Description	Possible Values and/or Default
-s, --slot ENUM	The key slot to operate on. See Note **1**	9a, 9c, 9d, 9e, 82, 83, 84, 85, 86, 87, 88, 89, 8a, 8b, 8c, 8d, 8e, 8f, 90, 91, 92, 93, 94, 95, f9 where - 9a for PIV Authentication. 9c for Digital Signature (PIN always checked). 9d for Key Management. 9e for Card Authentication (PIN never checked). 82-95 for Retired Key Management. f9 for Attestation.
-S, --subject STRING	The subject to use for certificate request.	The string must be written as: /CN=host.example.com/OU=test/O=example.com/
--scp11	Use encrypted communication in accordance with SCP11b. DEPRECATED as of yubico-piv-tool v2.7.2.	Use the --enc flag.
--serial INT	Serial number of the self-signed certificate	
--to-slot ENUM	The slot to move an existing key to. See Note **1**	9a, 9c, 9d, 9e, 82, 83, 84, 85, 86, 87, 88, 89, 8a, 8b, 8c, 8d, 8e, 8f, 90, 91, 92, 93, 94, 95, f9 where - 9a for PIV Authentication. 9c for Digital Signature (PIN always checked). 9d for Key Management. 9e for Card Authentication (PIN never checked). 82-95 for Retired Key Management. f9 for Attestation.

continues on next page

Table 1 – continued from previous page

Option	Description	Possible Values and/or Default
`--touch-policy ENUM`	Set touch policy for the slot containing the key. Applies to the actions: `generate`, `import-key` or `set-mgm-key`. Requires YubiKey 4 or newer.	`never`, `always`, `caches` Default: `never`
`-v, --verbose INT`	Print more information.	Default: `0`
`-V, --version`	Print version and exit.	
`--valid-days INT`	Time (in days) until the self-signed certificate expires.	Default: `365`

(1) For additional information on slot values, see PIV Certificate Slots.

(2) `--key` value default: `010203040506070801020304050607080102030405060708`.

6.2 PIV Tool action Command Parameters

6.2.1 Syntax

```
yubico-piv-tool --action ENUM <options>...

yubico-piv-tool -aENUM <options>...
```

6.2.2 Description

The tables lists the possible actions for the PIV tool command option `--action ENUM`. Where `ENUM` is replaced with options from the table. See the balance of this chapter for additional usage information.

6.2.3 Parameters

Action	Description
attest	Generate an X509 certificate for an asymmetric key that was generated inside the YubiKey.
change-pin	Change the PIN code required to access the PIV interface.
change-puk	Change the PUK.
`delete-cert`, *delete-certificate*	Delete a certificate from a specific slot.
delete-key	Delete a key from a specific slot.
generate	Generate an RSA or an EC key on a specific slot.
`import-cert`, *import-certificate*	Import an X509 certificate into a specific slot.
import-key	Import a private key into a specific slot.
list-readers	List the accessible smart card readers.
move-key	Move a key between slots.
pin-retries	Change the number of retries allowed before the PIN or the PUK are blocked.
`read-cert`, *read-certificate*	Return the X509 certificate stored on a specific slot.

continues on next page

Table 2 – continued from previous page

Action	Description
read-object	Return the content of a slot.
read-public-key	Return the public key stored on a specific slot.
`request`, `request-certificate`	Generate a certification request for an asymmetric key stored on a specific slot. See *generate*.
reset	Reset the YubiKey PIV interface.
`selfsign`, `selfsign-certificate`	Generate a self signed X509 certificate for an asymmetric key stored on a specific slot. See *generate*.
set-ccc	Set a new CCC.
set-chuid	Set or change the Card Holder Unique Identifier.
set-mgm-key	Set the management key required to perform administrative actions on the PIV interface.
`sign` *sign-data*	Sign input data.
status	Return the device metadata and content.
test-decipher	Test the decryption function.
test-signature	Test the digital signing function.
unblock-pin	Set a new PIN code after entered incorrectly too many times.
`verify-bio`	Verify the PIN code required to access the PIV interface on a bio Yubikey. See *generate*.

continues on next page

Table 2 – continued from previous page

Action	Description
`verify`, `verify-pin`	Verify the PIN code required to access the PIV interface. See *generate*.
version	Return the device firmware version.
`write-object`	Store an object in a slot. See *read-object*.

6.3 attest

6.3.1 Syntax

```
yubico-piv-tool --action=attest --slot ENUM --output=[STRING]

yubico-piv-tool -a attest
```

6.3.2 Description

The attestation, `attest`, feature is only available in YubiKey 4.3 and above.

Generate an X509 certificate for an asymmetric key that was generated inside the YubiKey.

- See attestation in this guide, *PIV Tool Attestation*.
- See attestation with a developer's product, PIV Attestation.

6.3.3 Examples

```
yubico-piv-tool --action=attest --slot=f9 --out SlotF9Intermediate.pem
```

6.3.4 Parameters

Parameter	Required Optional	Description	Possible values, Default
`-s`, `--slot ENUM`	Required	The key slot to operate on. See Note **1**	`9a`, `9c`, `9d`, `9e`, `82`, `83`, `84`, `85`, `86`, `87`, `88`, `89`, `8a`, `8b`, `8c`, `8d`, `8e`, `8f`, `90`, `91`, `92`, `93`, `94`, `95`, `f9` where - `9a` for PIV Authentication. `9c` for Digital Signature (PIN always checked). `9d` for Key Management. `9e` for Card Authentication (PIN never checked). `82-95` for Retired Key Management. `f9` for Attestation.
`-o`, `--output STRING`	Required	Filename to use as output or certificate file. If not specified, output is printed to `stdout`.	None for `stdout` or filename. Default: `-` for `stdout`

(1) For additional information on slot values, see PIV Certificate Slots.

6.4 change-pin

6.4.1 Syntax

```
yubico-piv-tool --action=change-pin --new-pin STRING

yubico-piv-tool -a change-pin -N <string>
```

6.4.2 Description

Change the Personal Identification Number (PIN) code required to access the PIV interface.

6.4.3 Parameters

Parameter	Required Optional	Description	Possible values, Default
`-N, --new-pin STRING`	Required	New PIN/PUK code changing to. If omitted, PIV tool prompts for PIN/PUK.	

6.5 change-puk

6.5.1 Syntax

```
yubico-piv-tool --action=change-puk --new-pin STRING

yubico-piv-tool -a change-puk -N <string>
```

6.5.2 Description

Change the Personal Unblocking Key (PUK).

6.5.3 Parameters

Parameter	Required Optional	Description	Possible values, Default
`-N,` `--new-pin STRING`	Required	New PIN/PUK code changing to. If omitted, PIV tool prompts for PIN/PUK.	

6.6 delete-certificate

6.6.1 Syntax

```
yubico-piv-tool --action=delete-certificate --slot ENUM --key [STRING]

yubico-piv-tool -a delete-certificate -s ENUM -k [STRING]
```

6.6.2 Description

Deletes a certificate from the specified slot. The corresponding private key is not deleted unless it is overwritten.

Deleting a certificate requires authentication by providing the management key. If no management key is provided, the PIV tool attempts authentication using the default management key.

Important: It is strongly recommended you change the Yubikey PIN, PUK, and management key before you start using the Yubikey.

6.6.3 Examples

```
$ yubico-piv-tool -a delete-certificate -s <slot> -k
```

6.6.4 Parameters

Parameter	Required Optional	Description	Possible values, Default
`-s`, `--slot ENUM`	Required	The key slot to operate on. See Note **1**	`9a`, `9c`, `9d`, `9e`, `82`, `83`, `84`, `85`, `86`, `87`, `88`, `89`, `8a`, `8b`, `8c`, `8d`, `8e`, `8f`, `90`, `91`, `92`, `93`, `94`, `95`, `f9` where - `9a` for PIV Authentication. `9c` for Digital Signature (PIN always checked). `9d` for Key Management. `9e` for Card Authentication (PIN never checked). `82-95` for Retired Key Management. `f9` for Attestation.
`-k`, `--key STRING`	Required	Management key to use. If no value is specified, PIV tool prompts for value	Default: See Note **2**

(1) For additional information on slot values, see PIV Certificate Slots.

(2) `--key` value default: `010203040506070801020304050607080102030405060708`.

6.7 delete-key

6.7.1 Syntax

```
$ yubico-piv-tool -a delete-key -s <slot> -k
```

6.7.2 Description

Deletes a key from the specified PIV slot. The function requires YubiKey 5.7 or higher.

Note: This actions deletes only the key, not the certificate. So if the slot already stores a certificate, it might still look populated even if the key is no longer there.

Deleting a key is an action that requires authentication, which is done by providing the management key. If no management key is provided, the tool tries to authenticate using the default management key.

Important: It is strongly recommended you change the Yubikey PIN, PUK, and management key before you start using the Yubikey.

6.7.3 Examples

```
$ yubico-piv-tool -a delete-key -s 9c -k
Enter Password:
Enter management key:
Successfully deleted key.
```

6.7.4 Parameters

Parameter	Required Optional	Description	Possible values, Default
`-s`, `--slot ENUM`	Required	The key slot to operate on. See Note **1**	`9a`, `9c`, `9d`, `9e`, `82`, `83`, `84`, `85`, `86`, `87`, `88`, `89`, `8a`, `8b`, `8c`, `8d`, `8e`, `8f`, `90`, `91`, `92`, `93`, `94`, `95`, `f9` where - `9a` for PIV Authentication. `9c` for Digital Signature (PIN always checked). `9d` for Key Management. `9e` for Card Authentication (PIN never checked). `82-95` for Retired Key Management. `f9` for Attestation.
`-k`, `--key [STRING]`	Required	Management key to use. If no value is specified, PIV tool prompts for value	Default: `010203040506070801020304050607080102030405060708010`

(1) For additional information on slot values, see PIV Certificate Slots.

6.8 generate

6.8.1 Syntax

```
$ yubico-piv-tool -a generate -s <slot> -k [ -A <key algorithm> -o <public key file> ]

$ yubico-piv-tool -a verify-pin -a selfsign -s <slot> -S <subject dn> [ -P <PIN code> --
↪pin-policy <never|once|always|matchonce|matchalways> --touch-policy
↪<never|always|cached> -i <public key file> --serial <cert serial number> --valid-days␣
```

(continues on next page)

(continued from previous page)

```
↪DAYS -o <cert file> ]

$ yubico-piv-tool -a verify-pin -a request-certificate -s <slot> -S <subject dn> [ -P
↪<PIN> -i <public key file> -o <cert request file> ]

$ yubico-piv-tool -a import-certificate -s <slot> -k [ -o <cert file> ]
```

6.8.2 Description

Generate an RSA or an EC key on a specific slot. This requires a sequence of action commands. Completed key generation can include `generate`, `selfsign`, `request-certificate`, `verify-pin` or `verify-bio`, and `import-certificate`.

An occupied slot on the Yubikey PIV interface usually contains a private key, a public key and an X509 certificate. The key pair generate, the certificate generation and the certificate import are done using different actions in the right order.

Generating a key pair sets the public key as an output (action `generate`). The public key is used to either generate a self signed certificate (action `selfsign`) or a certificate request (action `request-certificate`). The resulting certificate should then be imported into the same slot (action `import-certificate`).

Generating the key pair and importing the certificate are both actions that require authentication, which is done by providing the management key. If no management key is provided, the tool will try to authenticate using the default management key.

Important: It is strongly recommended to change the Yubikey's PIN, PUK and management key before start using it

---

While generating the certificate/certificate request does not require authentication, the signing operation does require verifying the PIN code or the fingerprint if the YubiKey supports Bio verification, which has to be done in an action that must take place before the signing action, otherwise the operation fails. Use `-a verify-pin` to verify the PIN and `-a verify-bio` for fingerprint verification.

6.8.3 Examples

6.8.3.1 Example 1: Self signed certificate on slot 9a

```
$ yubico-piv-tool -a generate -s 9a -A ECCP256 -k
-----BEGIN PUBLIC KEY-----
MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEwyLPuYF7xF4iQ+5VWUnDQsMSf9O7
Jc1gBDHQJ0kfYnZ8tV2OFk3JFyfZDL9g9g3eFaH00dzstxH7te64DtYepw==
-----END PUBLIC KEY-----
Successfully generated a new private key.
```

```
$ yubico-piv-tool -a verify-pin -a selfsign -s 9a -S '/CN=piv_auth/OU=test/O=example.com/
↪'
Enter PIN:
Successfully verified PIN.
Please paste the public key...
-----BEGIN PUBLIC KEY-----
MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEwyLPuYF7xF4iQ+5VWUnDQsMSf9O7
```

(continues on next page)

(continued from previous page)

```
Jc1gBDHQJ0kfYnZ8tV2OFk3JFyfZDL9g9g3eFaH00dzstxH7te64DtYepw==
-----END PUBLIC KEY-----
-----BEGIN CERTIFICATE-----
MIIBujCCAWCgAwIBAgIJAJKWdUFfuvqiMAoGCCqGSM49BAMCMDgxETAPBgNVBAMM
CHBpdl9hdXRoMQ0wCwYDVQQLDAR0ZXN0MRQwEgYDVQQKDAtleGFtcGxlLmNvbTAe
Fw0xOTA4MTIxMzM0NTdaFw0yMDA4MTExMzM0NTdaMDgxETAPBgNVBAMMCHBpdl9h
dXRoMQ0wCwYDVQQLDAR0ZXN0MRQwEgYDVQQKDAtleGFtcGxlLmNvbTBZMBMGByqG
SM49AgEGCCqGSM49AwEHA0IABMMiz7mBe8ReIkPuVVlJw0LDEn/TuyXNYAQx0CdJ
H2J2fLVdjhZNyRcn2Qy/YPYN3hWh9NHc7LcR+7XuuA7WHqejUzBRMB0GA1UdDgQW
BBQS0iNbyP8W817uCk/2lPd19ZvNRDAfBgNVHSMEGDAWgBQS0iNbyP8W817uCk/2
lPd19ZvNRDAPBgNVHRMBAf8EBTADAQH/MAoGCCqGSM49BAMCA0gAMEUCIQC5CTvl
LE0htwa89LBRRSL2BWHqciSLvqx9azjJfd63JAIgcAJSIhWpiXeBcGZdcTbnmkqU
kWu4LDU2ymBRp8pp4Iw=
-----END CERTIFICATE-----
Successfully generated a new self signed certificate.
```

```
$ yubico-piv-tool -a import-certificate -s 9a -k
Please paste the certificate...
-----BEGIN CERTIFICATE-----
MIIBujCCAWCgAwIBAgIJAJKWdUFfuvqiMAoGCCqGSM49BAMCMDgxETAPBgNVBAMM
CHBpdl9hdXRoMQ0wCwYDVQQLDAR0ZXN0MRQwEgYDVQQKDAtleGFtcGxlLmNvbTAe
Fw0xOTA4MTIxMzM0NTdaFw0yMDA4MTExMzM0NTdaMDgxETAPBgNVBAMMCHBpdl9h
dXRoMQ0wCwYDVQQLDAR0ZXN0MRQwEgYDVQQKDAtleGFtcGxlLmNvbTBZMBMGByqG
SM49AgEGCCqGSM49AwEHA0IABMMiz7mBe8ReIkPuVVlJw0LDEn/TuyXNYAQx0CdJ
H2J2fLVdjhZNyRcn2Qy/YPYN3hWh9NHc7LcR+7XuuA7WHqejUzBRMB0GA1UdDgQW
BBQS0iNbyP8W817uCk/2lPd19ZvNRDAfBgNVHSMEGDAWgBQS0iNbyP8W817uCk/2
lPd19ZvNRDAPBgNVHRMBAf8EBTADAQH/MAoGCCqGSM49BAMCA0gAMEUCIQC5CTvl
LE0htwa89LBRRSL2BWHqciSLvqx9azjJfd63JAIgcAJSIhWpiXeBcGZdcTbnmkqU
kWu4LDU2ymBRp8pp4Iw=
----END CERTIFICATE-----
Successfully imported a new certificate.
```

It is also possible to combine all these commands above into one single command (notice the order of the actions):

```
$ yubico-piv-tool -a generate -a verify-pin -a selfsign -a import-certificate -s 9a -k -
↪A ECCP256 -S '/CN=piv_auth/OU=test/O=example.com/'
```

6.8.3.2 Example 2: generate Signed certificate on slot 9c

```
$ yubico-piv-tool -a generate -s 9c -A RSA2048 -o pub.key
Successfully generated a new private key.
```

```
$ yubico-piv-tool -a verify-pin -a request-certificate -s 9c -S '/CN=digi_sign/OU=test/
↪O=example.com/' -i pub.key -o csr.pem
Enter PIN:
Successfully verified PIN.
Successfully generated a certificate request.
```

After sending the certificate request to the CA and getting a signed certificate:

```
$ yubico-piv-tool -a import-certificate -s 9c -i cert.pem
Successfully imported a new certificate.
```

6.8.4 Parameters

Parameter	Required Optional	Description	Possible values, Default
-s, --slot ENUM	Required	The key slot to operate on. See Note **1**	9a, 9c, 9d, 9e, 82, 83, 84, 85, 86, 87, 88, 89, 8a, 8b, 8c, 8d, 8e, 8f, 90, 91, 92, 93, 94, 95, f9 where - 9a for PIV Authentication. 9c for Digital Signature (PIN always checked). 9d for Key Management. 9e for Card Authentication (PIN never checked). 82-95 for Retired Key Management. f9 for Attestation.
-k, --key [STRING]	Required	Management key to use. If no value is specified, PIV tool prompts for value	Default: 010203040506070801020304050607080102030
-S, --subject STRING	Required	The subject to use for certificate request.	The string must be written as: /CN=host.example.com/ OU=test/O=example.com/

continues on next page

Table 3 – continued from previous page

Parameter	Required Optional	Description	Possible values, Default
`-A`, `--algorithm ENUM`	Optional	The algorithm to use to generate the key pair.	`RSA1024`, `RSA2048`, `ECCP256`, `ECCP384` Values that require YubiKey 5.7 or newer: `RSA3072`, `RSA4096`, `ED25519`, `X25519` Default: `RSA2048`
`-o`, `--output STRING`	Optional	Filename to use as output or certificate file. If not specified, output is printed to `stdout`.	None for `stdout` or filename. Default: `-` for `stdout`
`-P`, `--pin STRING`	Optional	PIN/PUK code for verification. If omitted, PIV tool prompts for PIN/PUK	
`--pin-policy ENUM`	Optional	Set pin policy for actions: `generate` or `import-key`. Only available on YubiKey 4 or newer.	Values PIN key verification: `never`, `once`, `always` Value BIO key verification: `matchonce`, `matchalways` Default: slot 9c, `always` slot 9a, 9d and 9e, `once` remaining slots, `never`
`--touch-policy ENUM`	Optional	Set touch policy for the slot containing the key. Applies to the actions: `generate`, `import-key` or `set-mgm-key`. Requires YubiKey 4 or newer.	`never`, `always`, `caches` Default: `never`

continues on next page

Table 3 – continued from previous page

Parameter	Required Optional	Description	Possible values, Default
`-i`, `--input STRING`	Optional	Filename to use as input. If left out, input is read from `stdin`.	None for `stdin` or filename. Default: `-` for stdin The only supported format for public key is PEM.
`--serial INT`	Optional	Serial number of the self-signed certificate	
`--valid-days INT`	Optional	Time (in days) until the self-signed certificate expires.	Default: `365`

(1) For additional information on slot values, see PIV Certificate Slots.

6.9 import-certificate

6.9.1 Syntax

```
$ yubico-piv-tool -a import-certificate -s <slot> -k [ -i <input file> -K <input file␣
↪format> ]
```

6.9.2 Description

Import an X509 certificate into a specific slot.

Part of generating an RSA or an EC key on a specific slot. This requires a sequence of action commands. Completed key generation can include `generate`, `selfsign`, `request-certificate`, `verify-pin` or `verify-bio`, and `import-certificate`. See *generate*.

The `import-key` command option precedes `import-certificate`. See *import-key*.

6.9.3 Examples

```
$ yubico-piv-tool -a import-certificate -s <slot> -k [ -o <cert file> ]
```

6.9.4 Parameters

Parameter	Required Optional	Description	Possible values, Default
`-s`, `--slot ENUM`	Required	The key slot to operate on. See Note **1**	`9a`, `9c`, `9d`, `9e`, `82`, `83`, `84`, `85`, `86`, `87`, `88`, `89`, `8a`, `8b`, `8c`, `8d`, `8e`, `8f`, `90`, `91`, `92`, `93`, `94`, `95`, `f9` where - `9a` for PIV Authentication. `9c` for Digital Signature (PIN always checked). `9d` for Key Management. `9e` for Card Authentication (PIN never checked). `82-95` for Retired Key Management. `f9` for Attestation.
`-k`, `--key [STRING]`	Required	Management key to use. If no value is specified, PIV tool prompts for value	Default: `010203040506070801020304050607080102030`
`-o`, `--output STRING`	Optional	Filename to use as output or certificate file. If not specified, output is printed to `stdout`.	None for `stdout` or filename. Default: `-` for `stdout`

(1) For additional information on slot values, see PIV Certificate Slots.

6.10 import-key

6.10.1 Syntax

```
$ yubico-piv-tool -a import-key -s <slot> -k [options]
```

6.10.2 Description

Imports a key, a certificate, or both into the Yubikey PIV interface for a specific slot. The largest accepted keys are of size 2025/3049 bytes for current versions of YubiKey NEO and YubiKey 5, respectively. It is possible to import larger certificates, but that requires compression in order for it to fit (see examples bellow).

This action is also used to import decryption keys (aka. key management keys typically found in slot 9d) into the retired slots (slots 82-95)

Importing either a key or a certificate is an action that requires authentication, which is done by providing the management key. If no management key is provided, the tool will try to authenticate using the default management key.

Important: It is strongly recommended to change the Yubikey's PIN, PUK and management key before start using it.

6.10.3 Examples

```
$ yubico-piv-tool -a import-key -s <slot> -k [ -P <PIN code> --pin-policy
↪<never|once|always|matchonce|matchalways> --touch-policy <never|always|cached> -i
↪<input file> -p <input file password> -K <input file format> ]

$ yubico-piv-tool -a import-certificate -s <slot> -k [ -i <input file> -K <input file␣
↪format> ]

$ yubico-piv-tool -a import-key -a import-certificate -s <slot> -k [ -P <PIN code> --pin-
↪policy <never|once|always|matchonce|matchalways> --touch-policy <never|always|cached> -
↪i <input file> -p <input file password> -K <input file format> ]
```

```
$ yubico-piv-tool -a import-key -a import-certificate -s 9c -k -i key.pfx -K PKCS12
Enter Password:
Enter management key:
Successfully imported a new private key.
Successfully imported a new certificate.

$ yubico-piv-tool -a import-certificate -s 9c -k -i cert_large.gz -K GZIP
Successfully imported a new certificate.
```

6.10.4 Parameters

Parameter	Required Optional	Description	Possible values, Default
`-s`, `--slot ENUM`	Required	The key slot to operate on. See Note **1**	`9a`, `9c`, `9d`, `9e`, `82`, `83`, `84`, `85`, `86`, `87`, `88`, `89`, `8a`, `8b`, `8c`, `8d`, `8e`, `8f`, `90`, `91`, `92`, `93`, `94`, `95`, `f9` where - `9a` for PIV Authentication. `9c` for Digital Signature (PIN always checked). `9d` for Key Management. `9e` for Card Authentication (PIN never checked). `82-95` for Retired Key Management. `f9` for Attestation.
`-k`, `--key [STRING]`	Required	Management key to use. If no value is specified, PIV tool prompts for value	Default: `010203040506070801020304050607080102030`
`-P`, `--pin STRING`	Optional	PIN/PUK code for verification. If omitted, PIV tool prompts for PIN/PUK	
`--pin-policy ENUM`	Optional	Set pin policy for actions: `generate` or `import-key`. Only available on YubiKey 4 or newer.	Values PIN key verification: `never`, `once`, `always` Value BIO key verification: `matchonce`, `matchalways` Default: slot 9c, `always` slot 9a, 9d and 9e, `once` remaining slots, `never`

continues on next page

Table 5 – continued from previous page

Parameter	Required Optional	Description	Possible values, Default
`--touch-policy ENUM`	Optional	Set touch policy for the slot containing the key. Applies to the actions: `generate`, `import-key` or `set-mgm-key`. Requires YubiKey 4 or newer.	`never`, `always`, `caches` Default: `never`
`-i, --input STRING`	Optional	Filename to use as input. If left out, input is read from `stdin`.	None for `stdin` or filename. Default: `-` for stdin The only supported format for public key is PEM.
`-p, --password STRING`	Optional	Password for decryption of private key file. If omitted, PIV tool prompts for password	
`-K, --key-format ENUM`	Optional	Format of the key being read/written.	`PEM`, `PKCS12`, `GZIP`, `DER`, `SSH` Default: `PEM`

(1) For additional information on slot values, see PIV Certificate Slots.

6.11 list-readers

No sample available.

6.12 move-key

6.12.1 Syntax

```
$ yubico-piv-tool -a move-key -s <slot> --to-slot <slot> -k
```

6.12.2 Description

Moves a key from one PIV slot to another. The function requires YubiKey 5.7 or higher.

Note: This actions moves only the key, not the certificate. So if the slot already stores a certificate, it might still look populated even if the key is no longer there.

Moving a key is an action that requires authentication, which is done by providing the management key. If no management key is provided, the tool will try to authenticate using the default management key.

Important: It is strongly recommended to change the Yubikey's PIN, PUK and management key before start using it.

6.12.3 Examples

```
$ yubico-piv-tool -a move-key -s 9c --to-slot 84 -k
Enter Password:
Enter management key:
Successfully moved key.
```

6.12.4 Parameters

Parameter	Required Optional	Description	Possible values, Default
`-s`, `--slot ENUM`	Required	The key slot to operate on. See Note **1**	`9a`, `9c`, `9d`, `9e`, `82`, `83`, `84`, `85`, `86`, `87`, `88`, `89`, `8a`, `8b`, `8c`, `8d`, `8e`, `8f`, `90`, `91`, `92`, `93`, `94`, `95`, `f9` where - `9a` for PIV Authentication. `9c` for Digital Signature (PIN always checked). `9d` for Key Management. `9e` for Card Authentication (PIN never checked). `82-95` for Retired Key Management. `f9` for Attestation.
`-k`, `--key [STRING]`	Required	Management key to use. If no value is specified, PIV tool prompts for value	Default: `010203040506070801020304050607080102030`
`--to-slot ENUM`	Required	The slot to move an existing key to. See Note **1**	`9a`, `9c`, `9d`, `9e`, `82`, `83`, `84`, `85`, `86`, `87`, `88`, `89`, `8a`, `8b`, `8c`, `8d`, `8e`, `8f`, `90`, `91`, `92`, `93`, `94`, `95`, `f9` where - `9a` for PIV Authentication. `9c` for Digital Signature (PIN always checked). `9d` for Key Management. `9e` for Card Authentication (PIN never checked). `82-95` for Retired Key Management. `f9` for Attestation.

(1) For additional information on slot values, see PIV Certificate Slots.

6.13 pin-retries

No sample available.

6.14 read-certificate

6.14.1 Syntax

```
$ yubico-piv-tool -a read-certificate -s <slot> [ -o <cert.pem> -K <cert file format> ]
```

6.14.2 Description

Returns the X509 certificate stored on a certain slot.

6.14.3 Examples

```
$ yubico-piv-tool -a read-cert -s 9a
-----BEGIN CERTIFICATE-----
MIIBuTCCAWCgAwIBAgIJAMOZXtijzEepMAoGCCqGSM49BAMCMDgxETAPBgNVBAMM
CHBpdl9hdXRoMQ0wCwYDVQQLDAR0ZXN0MRQwEgYDVQQKDAtleGFtcGxlLmNvbTAe
Fw0xOTA4MTMwODEwNDVaFw0yMDA4MTIwODEwNDVaMDgxETAPBgNVBAMMCHBpdl9h
dXRoMQ0wCwYDVQQLDAR0ZXN0MRQwEgYDVQQKDAtleGFtcGxlLmNvbTBZMBMGByqG
SM49AgEGCCqGSM49AwEHA0IABKPfSKeNY204JiHsSUwDAV8GuYqZOHfJJxrT4E0q
VWsKdC5zwRc7xvb2YgbMonPW5BfIUi766/VwWN54UsqWVuWjUzBRMB0GA1UdDgQW
BBR/bpCmGr+ark0VbGX5UvYWy9dM9DAfBgNVHSMEGDAWgBR/bpCmGr+ark0VbGX5
UvYWy9dM9DAPBgNVHRMBAf8EBTADAQH/MAoGCCqGSM49BAMCA0cAMEQCIHZZe7Xm
s6y8LKEBqGnbr1cbniHgMrvM1ST6GpL27HuaAiB+UwjI21GxIsd5r2avmwvT5LeZ
gQBns9KNCIgkwx+/Iw==
-----END CERTIFICATE-----
```

6.14.4 Parameters

Parameter	Required Optional	Description	Possible values, Default
`-s`, `--slot ENUM`	Required	The key slot to operate on. See Note **1**	`9a`, `9c`, `9d`, `9e`, `82`, `83`, `84`, `85`, `86`, `87`, `88`, `89`, `8a`, `8b`, `8c`, `8d`, `8e`, `8f`, `90`, `91`, `92`, `93`, `94`, `95`, `f9` where - `9a` for PIV Authentication. `9c` for Digital Signature (PIN always checked). `9d` for Key Management. `9e` for Card Authentication (PIN never checked). `82-95` for Retired Key Management. `f9` for Attestation.
`-o`, `--output STRING`	Optional	Filename to use as output or certificate file. If not specified, output is printed to `stdout`.	None for `stdout` or filename. Default: `-` for `stdout`
`-K`, `--key-format ENUM`	Optional	Format of the key being read/written.	`PEM`, `PKCS12`, `GZIP`, `DER`, `SSH` Default: `PEM`

(1) For additional information on slot values, see PIV Certificate Slots.

6.15 read-object

6.15.1 Syntax

```
$ yubico-piv-tool -a read-object --id <object ID> [ -o <output file> -f <file format> ]

$ yubico-piv-tool -a write-object --id <object ID> -k [ -i <input file>  -f <file format>
↪]
```

6.15.2 Description

The `read-object` syntax includes `write-object` syntax.

Reads and stores raw data into a PIV slot. The form and ID of the data are detailed in section 4.3 of the PIV Specification SP 800-73-4.

Writing an object is an action that requires authentication, which is done by providing the management key. If no management key is provided, the tool tries to authenticate using the default management key.

Important: It is strongly recommended to change the Yubikey's PIN, PUK and management key before start using it.

6.15.3 Examples

```
$ yubico-piv-tool -a read-object --id 0x5fc10d
708202b2308202ae30820196a003020102020832b1fd4fd258f9bd300d06092a864886f70d01010b0500
303b311530130603550403 0c0c4d616e6167656d656e7443413115301306035504 0a0c0c454a42434120
59756269636f310b3009060355040613025345301e170d3139303830383134333034325a170d32313038
3037313433034325a30203111300f0603550403 0c08757365725f333834310b3009060355040613025 3
453076301006072a8648ce3d020106052b8104002203620004564443 20b440fe49f312b023aa571da565
e9bc966dc928aef49c87e45d95cccf5b07fbe9e6620d2bb9d3c268671b2eed0e912c1dfae34f1e8f61a2
4565cb6498129618b96b7e3f38962796aa67382878cbe2cc1a8c369a55cecbd31b7a5cb032a37f307d30
0c0603551d130101ff04023000301f0603551d23041830168014 0c6d2aca0fe3aef788b50479477aba8a
87b08ad4301d0603551d250416301406082b0601050507030206082b0601050507030 4301d0603551d0e
04160414a508f3007b5344dc8efe08d87dfdbcb53191c7f3300e0603551d0f0101ff0404030205e0300d
06092a864886f70d01010b05000382010100 3993c325a5396ae1455e94d31dc6eda702b3e17b0f82de6d
1c22e994de13124022d7b127dff25a082c6f8a4ff74e0a965cb619bbc62787072b5d1ecb5a06e4b9d245
23534b1c4e6ac8265e8debb8111c62afbf8e1952e5ebd3ac81f6cf1900497719cb1ab60c1e92be9032db
1f69bf04d5def4fe2788de04452f2b01ced25fb186ce1b67c830dbbcc5e9d857951e347047c75f7456d4
2e9519694a7361f0b892d9acec10a55e5a61c483942543b13bd2c345b08ed1adc043647505a8d3ce2152
c4dfb8dc005e0fedc3d94aaf1e7e63b0c720c16481207451dd800e9cf7750c9bec580ce97aa540366ff1
f1ad5366fc3aac5563db73b6f44574968e3922e9e9fb710100fe00
```

6.15.4 Supported PIV Object IDs for read- and write-object

Type of Object Data	ASN.1 OID	ID
Card Capability Container	2.16.840.1.101.3.7.1.219.0	0x5fc107
Card Holder Unique Identifier	2.16.840.1.101.3.7.2.48.0	0x5fc102
X.509 Certificate for PIV Authentication	2.16.840.1.101.3.7.2.1.1	0x5fc105
Cardholder Fingerprints	2.16.840.1.101.3.7.2.96.16	0x5fc103
Security Object	2.16.840.1.101.3.7.2.144.0	0x5fc106
Cardholder Facial Image	2.16.840.1.101.3.7.2.96.48	0x5fc108
X.509 Certificate for Card Authentication	2.16.840.1.101.3.7.2.5.0	0x5fc101
X.509 Certificate for Digital Signature	2.16.840.1.101.3.7.2.1.0	0x5fc10a
X.509 Certificate for Key Management	2.16.840.1.101.3.7.2.1.2	0x5fc10b
Printed Information	2.16.840.1.101.3.7.2.48.1	0x5fc109
Discovery Object	2.16.840.1.101.3.7.2.96.80	0x7e
Key History Object	2.16.840.1.101.3.7.2.96.96	0x5fc10c
Retired X.509 Certificate for Key Management	2.16.840.1.101.3.7.2.16.1	0x5fc10d
Retired X.509 Certificate for Key Management	2.16.840.1.101.3.7.2.16.2	0x5fc10e
Retired X.509 Certificate for Key Management	2.16.840.1.101.3.7.2.16.3	0x5fc10f
Retired X.509 Certificate for Key Management	2.16.840.1.101.3.7.2.16.4	0x5fc110
Retired X.509 Certificate for Key Management	2.16.840.1.101.3.7.2.16.5	0x5fc111
Retired X.509 Certificate for Key Management	2.16.840.1.101.3.7.2.16.6	0x5fc112
Retired X.509 Certificate for Key Management	2.16.840.1.101.3.7.2.16.7	0x5fc113
Retired X.509 Certificate for Key Management	2.16.840.1.101.3.7.2.16.8	0x5fc114
Retired X.509 Certificate for Key Management	2.16.840.1.101.3.7.2.16.9	0x5fc115
Retired X.509 Certificate for Key Management	2.16.840.1.101.3.7.2.16.10	0x5fc116
Retired X.509 Certificate for Key Management	2.16.840.1.101.3.7.2.16.11	0x5fc117
Retired X.509 Certificate for Key Management	2.16.840.1.101.3.7.2.16.12	0x5fc118
Retired X.509 Certificate for Key Management	2.16.840.1.101.3.7.2.16.13	0x5fc119
Retired X.509 Certificate for Key Management	2.16.840.1.101.3.7.2.16.14	0x5fc11a
Retired X.509 Certificate for Key Management	2.16.840.1.101.3.7.2.16.15	0x5fc11b
Retired X.509 Certificate for Key Management	2.16.840.1.101.3.7.2.16.16	0x5fc11c
Retired X.509 Certificate for Key Management	2.16.840.1.101.3.7.2.16.17	0x5fc11d
Retired X.509 Certificate for Key Management	2.16.840.1.101.3.7.2.16.18	0x5fc11e
Retired X.509 Certificate for Key Management	2.16.840.1.101.3.7.2.16.19	0x5fc11f
Retired X.509 Certificate for Key Management	2.16.840.1.101.3.7.2.16.20	0x5fc120
Cardholder Iris Images	2.16.840.1.101.3.7.2.16.21	0x5fc121
Biometric Information Templates Group Templates	2.16.840.1.101.3.7.2.16.21	0x7f61
Secure Messaging Certificate Signer	2.16.840.1.101.3.7.2.16.21	0x5fc122
Pairing Code Reference Data Container	2.16.840.1.101.3.7.2.16.21	0x5fc123

6.15.5 Parameters

Parameter	Required Optional	Description	Possible values, Default
`--id INT`	Required	The ID of the object to write/read according to PIV Specifications	
`-k, --key [STRING]`	Required	Management key to use. If no value is specified, PIV tool prompts for value	Default: `010203040506070801020304050607080102030`
`-i, --input STRING`	Optional	Filename to use as input. If left out, input is read from `stdin`.	None for `stdin` or filename. Default: `-` for stdin The only supported format for public key is PEM.
`-o, --output STRING`	Optional	Filename to use as output or certificate file. If not specified, output is printed to `stdout`.	None for `stdout` or filename. Default: `-` for `stdout`
`-f, --format ENUM`	Optional	Format of data for write/read object.	`hex`, `base64`, `binary` Default: `hex`

6.16 read-public-key

6.16.1 Syntax

```
$ yubico-piv-tool -a read-public-key -s <slot> [ -o <cert.pem> -K <cert file format> ]
```

6.16.2 Description

Returns the X509 public key stored on a certain slot.

6.16.3 Examples

```
$ yubico-piv-tool -a read-public-key -s 9a
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAntRh/Q1ILx5n3KJIUJCM
vW1aNGa5jjlEwMBBtFWOrgEmmHUK4BvyMIVZyL5kYZr9aJZdrRW0+ltzGWWDZ0ET
nZrYIqHuJZuCaLQNk6kN+KJfW0/QGgV6WxMwniBIDL924miUlTjt8FvnuiW3oAuC
xLVktNp9cPlzXlWKvHqZzwprhX1SQ9AApuKiABxxiPmVdo2qSFflKMTH3wL+DRCO
Nbc/YRiJqEjqub0p67TMkgoBUfpCLYFiMFaHj4cv/RsTho/A0osnql6JSesGkDJJ
YhHs5RCYytvgqpx8BQp1iEawSw15Fq1eJxUyFbyeHoUkwVfTNso39KnhgDhGt2Xf
IQIDAQAB
-----END PUBLIC KEY-----
```

6.16.4 Parameters

Parameter	Required Optional	Description	Possible values, Default
`-s`, `--slot ENUM`	Required	The key slot to operate on. See Note **1**	`9a`, `9c`, `9d`, `9e`, `82`, `83`, `84`, `85`, `86`, `87`, `88`, `89`, `8a`, `8b`, `8c`, `8d`, `8e`, `8f`, `90`, `91`, `92`, `93`, `94`, `95`, `f9` where - `9a` for PIV Authentication. `9c` for Digital Signature (PIN always checked). `9d` for Key Management. `9e` for Card Authentication (PIN never checked). `82-95` for Retired Key Management. `f9` for Attestation.
`-o`, `--output STRING`	Optional	Filename to use as output or certificate file. If not specified, output is printed to `stdout`.	None for `stdout` or filename. Default: `-` for `stdout`
`-K`, `--key-format ENUM`	Optional	Format of the key being read/written.	`PEM`, `PKCS12`, `GZIP`, `DER`, `SSH` Default: `PEM`

(1) For additional information on slot values, see PIV Certificate Slots.

6.17 request-certificate

6.17.1 Description

Generate a certification request for an asymmetric key stored on a specific slot.

Part of generating an RSA or an EC key on a specific slot. This requires a sequence of action commands. Completed key generation can include `generate`, `selfsign`, `request-certificate`, `verify-pin` or `verify-bio`, and `import-certificate`.

See *generate*.

6.17.2 Examples

```
$ yubico-piv-tool -a verify-pin -a request-certificate -s <slot> -S <subject dn> [ -P
↪<PIN> -i <public key file> -o <cert request file> ]
```

6.18 reset

6.18.1 Syntax

```
$ yubico-piv-tool -a reset
```

6.18.2 Description

Erases all keys and certificates stored on the device and sets it to the default PIN, PUK and management key. This only affects the PIV application on the YubiKey, so any non-PIV configuration remains intact. Resetting the device does not erase the attestation key and certificate (slot f9) either, though they can be overwritten.

To reset the device, the PIN and the PUK need to be blocked. This happens when the wrong PIN and PUK is entered more than the number of their retries.

Global Reset

Some YubiKeys with firmware version 5.7.0 or higher have support for a global support option. This option erases all data on the YubiKey and is not restricted to the PIV application. It also does not require that the PIN and PUK to be blocked.

Note: The global reset option cannot be used over an encrypted session.

6.18.3 Examples

```
$ yubico-piv-tool -averify-pin -P471112
$ yubico-piv-tool -averify-pin -P471112
$ yubico-piv-tool -averify-pin -P471112
$ yubico-piv-tool -averify-pin -P471112
$ yubico-piv-tool -achange-puk -P471112 -N6756789
$ yubico-piv-tool -achange-puk -P471112 -N6756789
$ yubico-piv-tool -achange-puk -P471112 -N6756789
$ yubico-piv-tool -achange-puk -P471112 -N6756789
$ yubico-piv-tool -areset
```

```
$ yubico-piv-tool -areset --global
```

6.18.4 Parameters

Parameter	Required Optional	Description	Possible values, Default
`-global`	Optional	Reset the whole device over all applications, including the PIV application.	Default: `off`

6.19 selfsign-certificate

6.19.1 Description

Generate a self signed X509 certificate for an asymmetric key stored on a specific slot.

Part of generating an RSA or an EC key on a specific slot. This requires a sequence of action commands. Completed key generation can include `generate`, `selfsign`, `request-certificate`, `verify-pin` or `verify-bio`, and `import-certificate`.

See *generate*.

6.19.2 Examples

```
$ yubico-piv-tool -a verify-pin -a selfsign -s <slot> -S <subject dn> [ -P <PIN code> --
↪pin-policy <never|once|always|matchonce|matchalways> --touch-policy
↪<never|always|cached> -i <public key file> --serial <cert serial number> --valid-days␣
↪DAYS -o <cert file> ]
```

6.20 set-ccc

No sample available.

6.21 set-chuid

No sample available.

6.22 set-mgm-key

No sample available.

6.23 sign-data

6.23.1 Syntax

```
$ yubico-piv-tool -a verify-pin --sign -s <slot> [ -H <hash algorithm> -A <key algorithm>
↪ -P <PIN code> -i <input data file> -o <signature file> ]
```

6.23.2 Description

Signs input data.

The signing operation requires verifying the PIN code or the fingerprint if the YubiKey supports Bio verification. Use `-a verify-pin` to verify the PIN and `-a verify-bio` for fingerprint verification.

6.23.3 Examples

```
$ yubico-piv-tool -a verify-pin --sign -s 9c -H SHA512 -A RSA2048 -i data.txt -o data.sig
Enter PIN:
Successfully verified PIN.
Signature successful!
```

```
$ openssl dgst -sha512 -verify pubkey.pem -signature data.sig data.txt
Verified OK
```

6.23.4 Parameters

Parameter	Required Optional	Description	Possible values, Default
`-s`, `--slot ENUM`	Required	The key slot to operate on. See Note **1**	`9a`, `9c`, `9d`, `9e`, `82`, `83`, `84`, `85`, `86`, `87`, `88`, `89`, `8a`, `8b`, `8c`, `8d`, `8e`, `8f`, `90`, `91`, `92`, `93`, `94`, `95`, `f9` where - `9a` for PIV Authentication. `9c` for Digital Signature (PIN always checked). `9d` for Key Management. `9e` for Card Authentication (PIN never checked). `82-95` for Retired Key Management. `f9` for Attestation.

`-A`, `--algorithm ENUM`	Optional	The algorithm to use to generate the key pair.	`RSA1024`, `RSA2048`, `ECCP256`, `ECCP384`
			
			Values that require YubiKey 5.7 or newer:
			`RSA3072`, `RSA4096`, `ED25519`, `X25519`
			
			Default: `RSA2048`

`-H`, `--hash ENUM`	Optional	Hash to use for signatures.	`SHA1`, `SHA256`, `SHA384`, `SHA512`. Default: `SHA256`
`-P`, `--pin STRING`	Optional	PIN/PUK code for verification. If omitted, PIV tool prompts for PIN/PUK	
`-i`, `--input STRING`	Optional	Filename to use as input. If left out, input is read from `stdin`.	None for `stdin` or filename. Default: `-` for stdin The only supported format for public key is PEM.

(1) For additional information on slot values, see PIV Certificate Slots.

6.24 status

6.24.1 Syntax

```
$ yubico-piv-tool -a status [ -s <slot> ]
```

6.24.2 Description

Lists the device's meta data and the content of slots 9a, 9c, 9d and 9e. The content of slot f9 is listed if the slot is specified as an argument. This action, however, does **not** list the content of the retired slots (slots 82-95).

6.24.3 Examples

Example 1:

```
$ yubico-piv-tool -a status
Version:    4.4.0
Serial Number:      12345678
CHUID:      No data available
CCC:        No data available
Slot 9a:
        Private Key Algorithm:      RSA2048
        Public Key Algorithm:       RSA2048
        Subject DN:     CN=piv_auth, C=SE
        Issuer DN:      CN=TestCA, O=Yubico, C=SE
        Fingerprint:    4a1416fce853b29eaf520174bf8639d72ff30bd84e4586f81ac2a19eda43fdf1
        Not Before:     Aug  8 14:29:23 2019 GMT
        Not After:      Aug  7 14:29:23 2021 GMT
Slot 9c:
        Private Key Algorithm:      ECCP384
            Public Key Algorithm:   RSA2048
        Subject DN:     CN=sign, C=SE
        Issuer DN:      CN=TestCA, O=Yubico, C=SE
        Fingerprint:    803a89d5e196835d4a7e5e600e413fec1d3014712fcfd9e31fe15010829226dd
        Not Before:     Aug  8 14:29:50 2019 GMT
        Not After:      Aug  7 14:29:50 2021 GMT
        WARNING: Slot private key and certificate do not match
Slot 9d:
        Private Key Algorithm:      RSA2048
        Public Key Algorithm:       RSA2048
        Subject DN:     CN=key_mgm, C=SE
        Issuer DN:      CN=TestCA, O=Yubico, C=SE
        Fingerprint:    4a1416fce853429eaf420074bf8d39d72ff30bd84e4586f81ac2a19eda43fdf1
        Not Before:     Aug  8 14:29:23 2019 GMT
        Not After:      Aug  7 14:29:23 2021 GMT
```

(continues on next page)

(continued from previous page)

```
        WARNING: Slot private key and certificate do not match
Slot 9e:
        Private Key Algorithm:      RSA2048
        Public Key Algorithm:       RSA2048
        Subject DN:     CN=card_auth, C=SE
        Issuer DN:      CN=TestCA, O=Yubico, C=SE
        Fingerprint:    803a89d5e196845d4a7e5e6006413fec1d30157128cfd9e3afe15010829226dd
        Not Before:     Aug  8 14:29:50 2019 GMT
        Not After:      Aug  7 14:29:50 2021 GMT
PIN tries left:     3
```

Example 2:

```
$ yubico-piv-tool -a status -s 9a
Version:    4.4.0
Serial Number:      12345678
CHUID:      No data available
CCC:        No data available
Slot 9a:
        Private Key Algorithm:      RSA2048
        Public Key Algorithm:       RSA2048
        Subject DN:     CN=piv_auth, C=SE
        Issuer DN:      CN=TestCA, O=Yubico, C=SE
        Fingerprint:    4a1416fce853b29eaf520174bf8639d72ff30bd84e4586f81ac2a19eda43fdf1
        Not Before:     Aug  8 14:29:23 2019 GMT
        Not After:      Aug  7 14:29:23 2021 GMT
PIN tries left:     3
```

Example 3:

```
$ yubico-piv-tool -a status -s f9
Version:    4.4.0
Serial Number:      12345678
CHUID:      ␣
↪3019d4e739da739ced39ce739d836858210842108421c84210c3eb3410461c7c766122b38b2edf05183c3d0
   41a350832303330303130313e00fe00
CCC:        ␣
↪f015a000000116ff02f9a5b5f5fc5cd67c63a147ddf405f10121f20121f300f40100f50110f600f700fa00f
   b00fc00fd00fe00
Slot f9:
    Private Key Algorithm:  RSA2048
    Public Key Algorithm:   RSA2048
    Subject DN: CN=Test Attestation Certificate
    Issuer DN:  CN=Test Attestation Certificate
    Fingerprint:        8dbc03bea80282748f0403de0922c93751fe498d376b6ae1aa87d1b8af74c7a3
    Not Before: Jan 22 09:47:58 2018 GMT
    Not After:  Jan 24 09:47:58 2018 GMT
PIN tries left:     3
```

6.24.4 Parameters

Parameter	Required Optional	Description	Possible values, Default
`-s`, `--slot ENUM`	Required	The key slot to operate on. See Note **1**	`9a`, `9c`, `9d`, `9e`, `82`, `83`, `84`, `85`, `86`, `87`, `88`, `89`, `8a`, `8b`, `8c`, `8d`, `8e`, `8f`, `90`, `91`, `92`, `93`, `94`, `95`, `f9` where - `9a` for PIV Authentication. `9c` for Digital Signature (PIN always checked). `9d` for Key Management. `9e` for Card Authentication (PIN never checked). `82-95` for Retired Key Management. `f9` for Attestation.

(1) For additional information on slot values, see PIV Certificate Slots.

6.25 test-decipher

6.25.1 Syntax

```
$ yubico-piv-tool -a read-certificate -s <slot> [ -o cert.pem ]
$ yubico-piv-tool -a verify-pin -a test-decipher -s <slot> [ -P <PIN code> -i cert.pem ]
```

6.25.2 Description

Test the decryption function. This applies to both `test-signature` and `test-decipher`.

To test decryption:

1. Make sure there is a certificate stored on the slot being tested. To get the certificate, use the `read-certificate` action.
2. Verify the PIN code or the fingerprint, (for YubiKeys that support Bio verification). If the PIN code or fingerprint is not completed before a generation action, the tests fail.
   - To verify the PIN, use `-a verify-pin`
   - To verify the fingerprint, use `-a verify-bio`

Important: Run the test-decypher action before you run a generate action. If test is run out of order the test-decypher action fails.

6.25.3 Examples

Example 1:

```
$ yubico-piv-tool -a read-certificate -s 9a
-----BEGIN CERTIFICATE-----
MIIBuTCCAWCgAwIBAgIJAMOZXtijzEepMAoGCCqGSM49BAMCMDgxETAPBgNVBAMM
CHBpdl9hdXRoMQ0wCwYDVQQLDAR0ZXN0MRQwEgYDVQQKDAtleGFtcGxlLmNvbTAe
Fw0xOTA4MTMwODEwNDVaFw0yMDA4MTIwODEwNDVaMDgxETAPBgNVBAMMCHBpdl9h
dXRoMQ0wCwYDVQQLDAR0ZXN0MRQwEgYDVQQKDAtleGFtcGxlLmNvbTBZMBMGByqG
SM49AgEGCCqGSM49AwEHA0IABKPfSKeNY204JiHsSUwDAV8GuYqZOHfJJxrT4E0q
VWsKdC5zwRc7xvb2YgbMonPW5BfIUi766/VwWN54UsqWVuWjUzBRMB0GA1UdDgQW
BBR/bpCmGr+ark0VbGX5UvYWy9dM9DAfBgNVHSMEGDAWgBR/bpCmGr+ark0VbGX5
UvYWy9dM9DAPBgNVHRMBAf8EBTADAQH/MAoGCCqGSM49BAMCA0cAMEQCIHZZe7Xm
s6y8LKEBqGnbr1cbniHgMrvM1ST6GpL27HuaAiB+UwjI21GxIsd5r2avmwvT5LeZ
gQBns9KNCIgkwx+/Iw==
-----END CERTIFICATE-----
```

Example 2:

```
$ yubico-piv-tool -a verify-pin -a test-decipher -s 9a
Enter PIN:
Successfully verified PIN.
Please paste the certificate to encrypt for...
-----BEGIN CERTIFICATE-----
MIIBuTCCAWCgAwIBAgIJAMOZXtijzEepMAoGCCqGSM49BAMCMDgxETAPBgNVBAMM
CHBpdl9hdXRoMQ0wCwYDVQQLDAR0ZXN0MRQwEgYDVQQKDAtleGFtcGxlLmNvbTAe
Fw0xOTA4MTMwODEwNDVaFw0yMDA4MTIwODEwNDVaMDgxETAPBgNVBAMMCHBpdl9h
dXRoMQ0wCwYDVQQLDAR0ZXN0MRQwEgYDVQQKDAtleGFtcGxlLmNvbTBZMBMGByqG
SM49AgEGCCqGSM49AwEHA0IABKPfSKeNY204JiHsSUwDAV8GuYqZOHfJJxrT4E0q
VWsKdC5zwRc7xvb2YgbMonPW5BfIUi766/VwWN54UsqWVuWjUzBRMB0GA1UdDgQW
BBR/bpCmGr+ark0VbGX5UvYWy9dM9DAfBgNVHSMEGDAWgBR/bpCmGr+ark0VbGX5
UvYWy9dM9DAPBgNVHRMBAf8EBTADAQH/MAoGCCqGSM49BAMCA0cAMEQCIHZZe7Xm
s6y8LKEBqGnbr1cbniHgMrvM1ST6GpL27HuaAiB+UwjI21GxIsd5r2avmwvT5LeZ
gQBns9KNCIgkwx+/Iw==
```

(continues on next page)

(continued from previous page)

```
-----END CERTIFICATE-----
Successfully performed ECDH exchange with card.
```

Example 3:

It is also possible to combine the commands above into one single command. Be sure to use the correct actions order:

```
$ yubico-piv-tool -a read-certificate -a verify-pin -a test-decipher -s 9a -o cert.pem -
↪i cert.pem
Enter PIN:
Successfully verified PIN.
Successfully performed ECDH exchange with card.
```

6.25.4 Parameters

Parameter	Required Optional	Description	Possible values, Default
`-s`, `--slot ENUM`	Required	The key slot to operate on. See Note **1**	`9a`, `9c`, `9d`, `9e`, `82`, `83`, `84`, `85`, `86`, `87`, `88`, `89`, `8a`, `8b`, `8c`, `8d`, `8e`, `8f`, `90`, `91`, `92`, `93`, `94`, `95`, `f9` where - `9a` for PIV Authentication. `9c` for Digital Signature (PIN always checked). `9d` for Key Management. `9e` for Card Authentication (PIN never checked). `82-95` for Retired Key Management. `f9` for Attestation.
`-P`, `--pin STRING`	Optional	PIN/PUK code for verification. If omitted, PIV tool prompts for PIN/PUK	
`-i`, `--input STRING`	Optional	Filename to use as input. If left out, input is read from `stdin`.	None for `stdin` or filename. Default: `-` for stdin The only supported format for public key is PEM.
`-o`, `--output STRING`	Optional	Filename to use as output or certificate file. If not specified, output is printed to `stdout`.	None for `stdout` or filename. Default: `-` for `stdout`

(1) For additional information on slot values, see PIV Certificate Slots.

6.26 test-signature

6.26.1 Syntax

```
$ yubico-piv-tool -a read-certificate -s <slot> [ -o cert.pem ]
$ yubico-piv-tool -a verify-pin -a test-signature -s <slot> [ -P <PIN code> -i cert.pem ]
```

6.26.2 Description

Test the signature function. This applies to both `test-signature` and `test-decipher`.

To test signing:

1. Make sure there is a certificate stored on the slot being tested. To get the certificate, use the `read-certificate` action.
2. Verify the PIN code or the fingerprint, (for YubiKeys that support Bio verification). If the PIN code or fingerprint is not completed before a generation action, the tests fail.
   - To verify the PIN, use `-a verify-pin`
   - To verify the fingerprint, use `-a verify-bio`

Important: Run the test-decypher action before you run a generate action. If test is run out of order the test-signature action fails.

6.26.3 Examples

Example 1:

```
$ yubico-piv-tool -a read-certificate -s 9a
-----BEGIN CERTIFICATE-----
MIIBuTCCAWCgAwIBAgIJAMOZXtijzEepMAoGCCqGSM49BAMCMDgxETAPBgNVBAMM
CHBpdl9hdXRoMQ0wCwYDVQQLDAR0ZXN0MRQwEgYDVQQKDAtleGFtcGxlLmNvbTAe
Fw0xOTA4MTMwODEwNDVaFw0yMDA4MTIwODEwNDVaMDgxETAPBgNVBAMMCHBpdl9h
dXRoMQ0wCwYDVQQLDAR0ZXN0MRQwEgYDVQQKDAtleGFtcGxlLmNvbTBZMBMGByqG
SM49AgEGCCqGSM49AwEHA0IABKPfSKeNY204JiHsSUwDAV8GuYqZOHfJJxrT4E0q
VWsKdC5zwRc7xvb2YgbMonPW5BfIUi766/VwWN54UsqWVuWjUzBRMB0GA1UdDgQW
BBR/bpCmGr+ark0VbGX5UvYWy9dM9DAfBgNVHSMEGDAWgBR/bpCmGr+ark0VbGX5
UvYWy9dM9DAPBgNVHRMBAf8EBTADAQH/MAoGCCqGSM49BAMCA0cAMEQCIHZZe7Xm
s6y8LKEBqGnbr1cbniHgMrvM1ST6GpL27HuaAiB+UwjI21GxIsd5r2avmwvT5LeZ
gQBns9KNCIgkwx+/Iw==
-----END CERTIFICATE-----
```

Example 2:

```
$ yubico-piv-tool -a verify-pin -a test-signature -s 9a
Enter PIN:
Successfully verified PIN.
Please paste the certificate to verify against...
-----BEGIN CERTIFICATE-----
MIIBuTCCAWCgAwIBAgIJAMOZXtijzEepMAoGCCqGSM49BAMCMDgxETAPBgNVBAMM
CHBpdl9hdXRoMQ0wCwYDVQQLDAR0ZXN0MRQwEgYDVQQKDAtleGFtcGxlLmNvbTAe
Fw0xOTA4MTMwODEwNDVaFw0yMDA4MTIwODEwNDVaMDgxETAPBgNVBAMMCHBpdl9h
dXRoMQ0wCwYDVQQLDAR0ZXN0MRQwEgYDVQQKDAtleGFtcGxlLmNvbTBZMBMGByqG
SM49AgEGCCqGSM49AwEHA0IABKPfSKeNY204JiHsSUwDAV8GuYqZOHfJJxrT4E0q
VWsKdC5zwRc7xvb2YgbMonPW5BfIUi766/VwWN54UsqWVuWjUzBRMB0GA1UdDgQW
BBR/bpCmGr+ark0VbGX5UvYWy9dM9DAfBgNVHSMEGDAWgBR/bpCmGr+ark0VbGX5
UvYWy9dM9DAPBgNVHRMBAf8EBTADAQH/MAoGCCqGSM49BAMCA0cAMEQCIHZZe7Xm
s6y8LKEBqGnbr1cbniHgMrvM1ST6GpL27HuaAiB+UwjI21GxIsd5r2avmwvT5LeZ
gQBns9KNCIgkwx+/Iw==
-----END CERTIFICATE-----
Successful ECDSA verification.
```

Example 3:

It is also possible to combine the commands above into one single command. Be sure to use the correct actions order:

```
$ yubico-piv-tool -a read-certificate -a verify-pin -a test-signature -s 9a -o cert.pem -
↪i cert.pem
Enter PIN:
Successfully verified PIN.
Successful ECDSA verification.
```

6.26.4 Parameters

Parameter	Required Optional	Description	Possible values, Default
`-s`, `--slot ENUM`	Required	The key slot to operate on. See Note **1**	`9a`, `9c`, `9d`, `9e`, `82`, `83`, `84`, `85`, `86`, `87`, `88`, `89`, `8a`, `8b`, `8c`, `8d`, `8e`, `8f`, `90`, `91`, `92`, `93`, `94`, `95`, `f9` where - `9a` for PIV Authentication. `9c` for Digital Signature (PIN always checked). `9d` for Key Management. `9e` for Card Authentication (PIN never checked). `82-95` for Retired Key Management. `f9` for Attestation.
`-P`, `--pin STRING`	Optional	PIN/PUK code for verification. If omitted, PIV tool prompts for PIN/PUK	
`-i`, `--input STRING`	Optional	Filename to use as input. If left out, input is read from `stdin`.	None for `stdin` or filename. Default: `-` for stdin The only supported format for public key is PEM.
`-o`, `--output STRING`	Optional	Filename to use as output or certificate file. If not specified, output is printed to `stdout`.	None for `stdout` or filename. Default: `-` for `stdout`

(1) For additional information on slot values, see PIV Certificate Slots.

6.27 unblock-pin

No sample available.

6.28 verify-bio

6.28.1 Description

Use `-a verify-pin` to verify the PIN and `-a verify-bio` for fingerprint verification.

See *generate*, *test-signature*, *test-decipher*, or *sign-data*.

6.28.2 Examples

```
$ yubico-piv-tool -a verify-bio -a selfsign -s <slot> -S <subject dn> [ -P <PIN code> --
↪pin-policy <never|once|always|matchonce|matchalways> --touch-policy
↪<never|always|cached> -i <public key file> --serial <cert serial number> --valid-days␣
↪DAYS -o <cert file> ]
```

```
$ yubico-piv-tool -a verify-bio -a request-certificate -s <slot> -S <subject dn> [ -P
↪<PIN> -i <public key file> -o <cert request file> ]
```

6.29 verify-pin

6.29.1 Description

Use `-a verify-pin` to verify the PIN and `-a verify-bio` for fingerprint verification.

See *generate*, *test-signature*, *test-decipher*, or *sign-data*.

6.29.2 Examples

```
$ yubico-piv-tool -a verify-pin -a selfsign -s <slot> -S <subject dn> [ -P <PIN code> --
↪pin-policy <never|once|always|matchonce|matchalways> --touch-policy
↪<never|always|cached> -i <public key file> --serial <cert serial number> --valid-days␣
↪DAYS -o <cert file> ]
```

```
$ yubico-piv-tool -a verify-pin -a request-certificate -s <slot> -S <subject dn> [ -P
↪<PIN> -i <public key file> -o <cert request file> ]
```

6.30 version

6.30.1 Syntax

```
$ yubico-piv-tool -a version
```

6.30.2 Description

Displays the application version.

6.30.3 Examples

```
$ yubico-piv-tool -a version
Application version 4.4.0 found.
```

6.31 write-object

6.31.1 Syntax

```
$ yubico-piv-tool -a write-object --id <object ID> -k [ -i <input file>  -f <file format>
↪]
```

6.31.2 Description

Writing an object is an action that requires authentication, which is done by providing the management key. If no management key is provided, the tool tries to authenticate using the default management key.

Important: It is strongly recommended to change the Yubikey's PIN, PUK and management key before start using it.

See *read-object* for *Supported PIV Object IDs for read- and write-object* and parameters.

CHAPTER

SEVEN

COPYRIGHT

7.1 Trademarks

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

7.2 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.

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.

7.3 Contact Information

Yubico AB
Kungsgatan 44
111 35 Stockholm
Sweden

More options for getting touch with us are available on the Contact page of Yubico's website.

7.4 Document Updated

2025-09-09 18:24:24 UTC