FEITIAN EPASS NFC

Feitian ePass NFC

smartcard feitian epass ssh Google Titan

SSH Authentication with a Feitian ePass NFC/FIDO/U2F Security Key

Feitian ePass NFC FIDO U2F Security Key can work as a Generic Identity Device Specification (GIDS) smart card. There also are many other manufacturers and card models to which these instructions can be applied, but the specific tools to initialize the card can be different.

In this how-to we will prepare a USB/NFC hardware key for SSH authentication and register the device in WebADM. It is assumed you have a working WebADM and Spankey servers already, although you can also deploy the public key manually to a destination server.

For Yubikeys and other PIV devices, please refer to Smart Card -PIV

1. Enabling CCID Mode

Feitian ePass keys can work in three different modes:

-U2F/FIDO

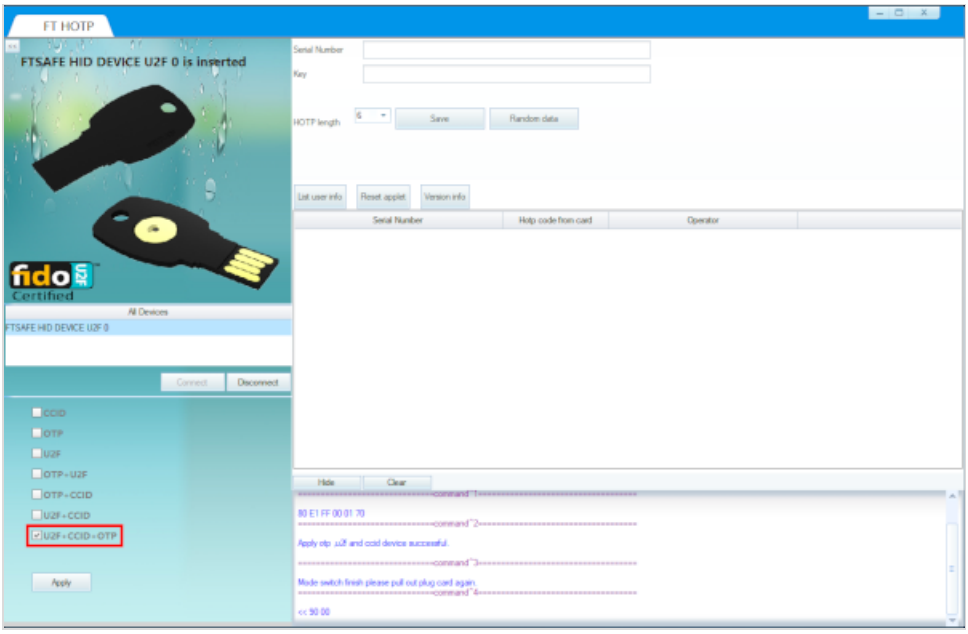
-OATH HOTP (Event-based)

-CCID (chip card interface device)

All these modes can be used with OpenOTP, but in this guide, we focus on the CCID which is useful for SSH authentication.

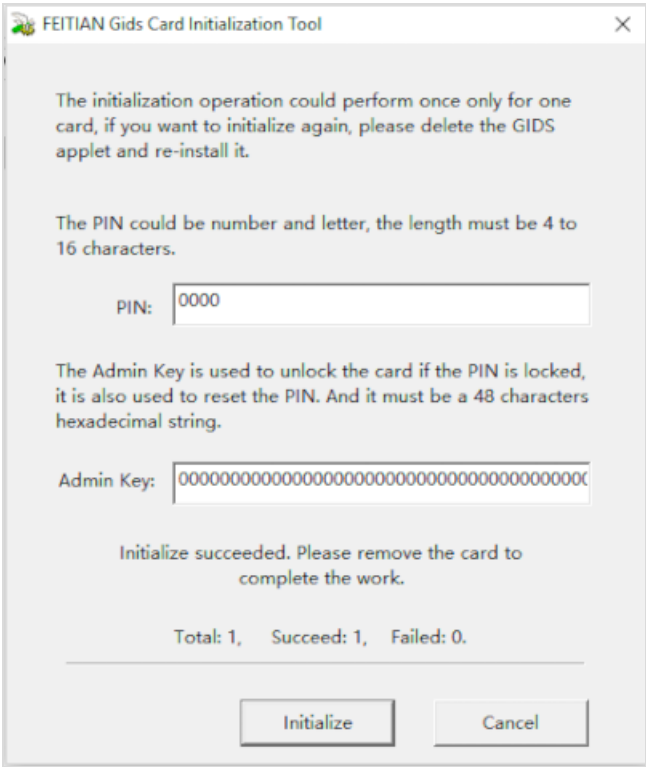
By default the Feitian keys ship only have U2F/FIDO mode enabled, thus the first step is to change the operating mode of the key. This is done with a Feitian tool (ePassFIDO-NFC OTP Tool 3.7), which can be downloaded from Feitian website and which works in Windows.

Connect the key to a computer with the Feitian tool, start the tool and select an operating mode which includes CCID. The key can work simultaneously on all three modes. If the new mode includes U2F, you can continue to use the key for FIDO authentication simultaneously with SSH key authentication.



After the CCID mode is enabled, the smart card function must be initialized using other Feitian software (GIDS Initialization Tool). This tool can also be downloaded from the Feitian website.

Start the tool in windows, set the desired PIN and Admin key and click Initialize. Please take care in selecting and storing the PIN and Admin key.



2. Generating SSH Keys

Once the key is initialized, we can generate SSH keypair and extract the public key. For this, we need to connect it to a computer with OpenSC (version 0.18 or later).

First, we verify that the key is connected and recognized correctly:

```
[john@Mac-mini ~]$ opensc-tool --list-readers
# Detected readers (pcsc)
Nr.  Card  Features  Name
0    Yes             FT U2F CCID KB
```

Next we can dump the contents of the key:

```
[john@Mac-mini ~]$ pkcs15-tool -D
Using reader with a card: FT U2F CCID KB
PKCS#15 Card [GIDS card]:
 Version        : 2
 Serial number  : dd2de1de707dbd44ba70a1cdc89296
 Manufacturer ID: www.mysmartlogon.com
 Flags          :

PIN [UserPIN]
 Object Flags   : [0x3], private, modifiable
 ID             : 80
 Flags          : [0x12], local, initialized
 Length         : min_len:4, max_len:15, stored_len:0
 Pad char       : 0x00
 Reference      : 128 (0x80)
 Type           : ascii-numeric
 Tries left     : 3
```

Please note the ID number of the PIN, as this is used in the next command as –auth-id parameter, when we generate the public-private keypair:

```
[john@Mac-mini ~]$ pkcs15-init  --verify-pin --auth-id 80 --generate-key rsa/2048 --
key-usage sign,decrypt --label "RSA"
```

Once the key pair is generated, we can list the contents of the device and extract the public key.

```
[john@Mac-mini ~]$ pkcs15-tool --list-key
Using reader with a card: FT U2F CCID KB
Private RSA Key [RSA]
 Object Flags   : [0x1], private
 Usage          : [0x6], decrypt, sign
 Access Flags   : [0x1D], sensitive, alwaysSensitive, neverExtract, local
 ModLength      : 2048
 Key ref        : 129 (0x81)
 Native         : yes
 Auth ID        : 80
 ID             : 00
 MD:guid        : 8d41c334-4ef0-805d-464c-5e8881e5e754

[john@Mac-mini ~]$ pkcs15-tool --read-public-key 00
Using reader with a card: FT U2F CCID KB
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAjFiuFwpwKytBH+igZ9MM
nFbcN+M2Cdz4+jpUNuGpDqaKt+bdGyIdqtdkoEws9+G53lBvjHTjWJ9gy09/ck7a
a+wwoBbHwDfN8MQ9fyZYkIglWDY3nNOytsSTCzf8xWp67J2rtCiM4cMbcYXYtYDL
CNqACDHvSFk4jir/Jpl9Ai8dYX2Y9L9aN8eZlwKVTwbWahkIYsWpa6jrFwxUe9NW
nLhdKsG3YLJd6H2Jwe2PWUXul3WHL3NFfkmijTg2tcbyHSX+l4KxFoFOEfzTrVWu
zLNQRi4z8O/WzXv46Ra2DXOg3WOOLPM35DGNDE3VK1Wy9WHhlxIPaX5lPQSBIJTM
nQIDAQAB
-----END PUBLIC KEY-----
```

With this information we can create an inventory file in .csv format with the right public key. The serial number can be decided by you, as it is only used for keeping track of the devices in WebADM inventory. You can for example user the number printed on the physical device:

```
"Type","Reference","Description","DN","Data","Status"
"PIV Pubkey","1234","FT
ePass","","PublicKey=MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAjFiuFwpwKytBH+igZ9MMnFbc
```

Next, we can import this file into the WebADM inventory, assign it to a user and test authentication with ssh. These steps are described in Smart Card -PIV starting at “We import the file. Under the import tab, we click on Import Inventory File:”

3. Using with a Contactless Reader

The Feitian ePass key supports NFC communication. Thus it is possible to use the GIDS smart card for authentication also with a contactless reader, resulting a very easy workflow:

```
[root@fedora28 ~]#  pkcs15-tool --read-public-key 00
Using reader with a card: Broadcom Corp 5880 [Contactless SmartCard] (0123456789ABCD)
01 00
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAjFiuFwpwKytBH+igZ9MM
nFbcN+M2Cdz4+jpUNuGpDqaKt+bdGyIdqtdkoEws9+G53lBvjHTjWJ9gy09/ck7a
a+wwoBbHwDfN8MQ9fyZYkIglWDY3nNOytsSTCzf8xWp67J2rtCiM4cMbcYXYtYDL
CNqACDHvSFk4jir/Jpl9Ai8dYX2Y9L9aN8eZlwKVTwbWahkIYsWpa6jrFwxUe9NW
nLhdKsG3YLJd6H2Jwe2PWUXul3WHL3NFfkmijTg2tcbyHSX+l4KxFoFOEfzTrVWu
zLNQRi4z8O/WzXv46Ra2DXOg3WOOLPM35DGNDE3VK1Wy9WHhlxIPaX5lPQSBIJTM
nQIDAQAB
-----END PUBLIC KEY-----
```

Or in SSH authorized key format, which can also be directly copied to the authorized_keys file in the destination server:

```
[john@Mac-mini ~]$ pkcs15-tool --read-ssh-key 00
#  Using reader with a card: Broadcom Corp 5880 [Contactless SmartCard]
(0123456789ABCD) 01 00
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQCMWK4XCnArK0Ef6KBn0wycVtw34zYJ3Pj6OlQ24akOpoq35t0bIh2q12Sg1
RSA
```

```
[john@Mac-mini ~]$ ssh -I opensc-pkcs11.so user@123.123.35.12
Enter PIN for 'UserPIN (GIDS card)':
Last login: Thu Sep 20 15:49:46 2018 from 123.188.35.12
user@host:~$
```

If you want to use the smart card authentication without providing the -I flag in the command line, you can configure OpenSC library in ~/.ssh/config (or to /etc/ssh/ssh_config for all users). The correct path the library depends on your system.

```
PKCS11Provider /usr/lib/opensc-pkcs11.so
```