Openloop

Security Key Guide

PIV/PKCS#11 & FIDO2/CTAP2 Passkeys

Haudi Crypto, Inc.

Version 1.0.0
2026-08-08

Contents

---

1. Introduction

Openloop works not only as a crypto-asset hardware wallet but also as a **security key**. A single device provides the following two security functions.

Function	Protocol	Main uses
Passkey	FIDO2/CTAP2	Passwordless authentication for websites, two-factor authentication
PIV/PKCS#11	PIV (NIST SP 800-73)	SSH authentication, TLS client certificates, code signing

These two functions do not overlap in purpose; they complement each other.

Use case	Passkey (CTAP2)	PIV/PKCS#11
Browser WebAuthn authentication	✓	—
SSH public-key authentication	—	✓
TLS client authentication	—	✓
Code signing / git signing	—	✓

Prerequisites

Openloop passkeys can be used over **two transports**. Each has different prerequisites.

Transport	Prerequisites	When to use
USB CTAPHID (desktop)	No software installation required, browser only	Windows / macOS / Linux PC
BLE (via Openloop Connect)	Requires the **Openloop Connect mobile app** plus device pairing	iOS / Android smartphone

For PIV/PKCS#11, you need **Openloop Connect** (the desktop app) running. The PKCS#11 library is bundled with Connect.

---

2. Passkeys (FIDO2/CTAP2)

2.1 Overview

Openloop operates as a FIDO2/CTAP2-compliant security key. As a secure alternative to passwords, you can use it on any supporting website or service.

> ⚠ **Certification status**: At this time, Openloop has not obtained the FIDO Alliance's **FIDO Certified Authenticator** certification (FIDO2 Certified). While it is a spec-compliant CTAP2/WebAuthn implementation, official FIDO Alliance certification and AAGUID registration (FIDO Metadata Service) may be pursued in the future.

Item	Specification
Protocol	CTAP2 (FIDO_2_0)
Backward compatibility	U2F (FIDO U2F V2)
Transport	USB HID (CTAPHID) / BLE (via Openloop Connect, see below)
Signature algorithms	ES256 (P-256) / EdDSA (Ed25519)
Discoverable Credential	✓ Supported (Resident Key)
User Verification	✓ Device PIN + physical touch
Maximum credentials	100
Attestation	Self-attestation

⚠️ **About factory reset**: Performing a factory reset permanently deletes all FIDO2 passkeys and PIV keys. Passkeys cannot be restored from the recovery phrase. You will need to re-register the security key with each service.

2.2 The two transports (USB / BLE) — important

Openloop passkeys work over entirely different paths for USB and BLE. Because this is easy to confuse, we clarify it up front.

🔌 USB CTAPHID path (desktop)

```
[PC browser] ——— USB HID (CTAPHID) ——— [Openloop device]
              ↑
              Works through the OS's built-in security-key support
              Openloop Connect is not required
```

- Uses the security-key support **provided by the OS itself** (Windows: webauthn.dll; macOS / Linux: built into the browser)
- The browser's picker/prompt shows something like "**Insert your security key**" or "Use a security key"
- No Openloop Connect installation required
- Authenticate by connecting the USB cable and pressing the device button

📶 BLE (via Openloop Connect) path (mobile)

```
[iOS/Android browser]
   ↓ navigator.credentials API
[OS Credential Manager]
   ↓ Provider selection (user action)
[Openloop Connect app's Credential Provider Extension/Service]
   ↓ BLE
[Openloop device]
```

- **Implemented independently by Openloop Connect** as an iOS Credential Provider Extension / Android Credential Provider Service
- Appears in the browser's passkey picker as follows:
  - **iOS**: "**Openloop Connect**" (per iOS spec, the containing-app name is displayed; it actually operates over BLE)
  - **Android**: "**Openloop Connect**" (over BLE)
- Required: the Openloop Connect mobile app + an Openloop device paired over BLE
- No USB cable needed; the device only has to be within Bluetooth range

> ⚠ **iOS note**: The picker shows "Openloop Connect," but **communication happens over BLE, not USB**. Because iOS fixes an Extension's display name to the parent app name (Openloop Connect), the fact that it operates over BLE is not obvious from the UI. Keep this in mind.

2.3 Per-transport RP behavior — important (Google, etc.)

For most RPs (Relying Parties, i.e. websites), **a single registration can be used over either transport (USB/BLE)**. However, some RPs (a notable example being **Google**) apply strict `transports` hint filtering, so a **USB-registered credential works only over USB, and a BLE-registered credential works only over BLE**.

Concrete examples affected

RP	Using a USB registration over BLE	Using a BLE registration over USB
Most sites (Microsoft, GitHub, JAL, etc.)	✓ Works	✓ Works
Google	✗ Not visible in the iOS BLE picker	△ Works (Mac/Chrome's transport filter is lenient)

If you want to use both transports (Google, etc.)

By **registering the passkey twice — once over USB and once over BLE — on the same Openloop device**, you can make it usable over both transports. Because Openloop generates a different credential ID per transport even on the same hardware, the two are registered on the server as two independent credentials.

```
Google account settings:
  ├─ Security keys section: the USB registration (used on PC/Mac)
  └─ Passkeys section: the BLE registration (used on iPhone/Android)
```

2.4 Enabling it on Openloop

The passkey function is disabled by default. Whether you use USB or BLE, enable it with the steps below.

Common: enable the passkey function

1. On the Openloop device, open **Settings** > **Passkey**
2. Set **Passkey** to **ON**

If using the USB CTAPHID path

1. Open **Settings** > **USB settings**
2. Set **USB HID** to **ON**

If using the BLE (Openloop Connect) path

1. Open **Settings** > **Bluetooth**
2. Set **Bluetooth** to **ON**
3. On iOS / Android, launch the Openloop Connect app and pair/select the device

2.5 Resident Key and Non-Resident Key

There are two kinds of passkey credentials.

Kind	Description	Characteristics
Resident Key (Discoverable Credential)	Stores the credential information inside the device. Authentication is possible without entering a username.	Uses the device's storage. Openloop can store **up to 100**.
Non-Resident Key	Embeds information wrapped with a device-specific cipher in the credential ID. Holds no state inside the device.	No limit on the number stored. However, the server must present the credentialId at authentication time.

Which one a website requires is determined by the `residentKey` parameter at registration. Recent passkey-capable services (Google, Microsoft, GitHub, etc.) commonly require a Resident Key.

> You can view and delete the list of Resident Keys stored on the device under **Settings** > **Passkey** > **Credential list**. Non-Resident Keys hold no state on the device, so they do not appear in the list.

2.6 Registering a passkey — USB CTAPHID path (desktop)

Register Openloop as a USB security key from a PC browser.

1. In the website's security settings, choose "Add a security key" or similar
2. The browser shows a dialog asking you to insert the security key
3. Connect Openloop over USB (or it may already be connected)
4. A registration request appears on the Openloop device screen
5. Review the details and touch **Approve**
6. You may be prompted to enter the device PIN (depending on the User Verification setting)

2.7 Authenticating with a passkey — USB CTAPHID path

Use this when logging in to a registered website from a PC browser.

1. On the website's login screen, choose "Sign in with a security key" or similar
2. Connect Openloop over USB
3. An authentication request appears on the Openloop device screen
4. Touch **Approve**

2.8 Registering a passkey — BLE (via Openloop Connect) (mobile)

From an iPhone / Android browser, register a passkey on the Openloop connected over BLE via Openloop Connect.

Preparation

1. Install the **Openloop Connect** mobile app (App Store / Google Play)
2. Launch the app and **pair over BLE** with the Openloop device, then select it (in Connect's device management screen)
3. On the Openloop device, confirm that **Settings** > **Bluetooth** is ON

> **When using multiple devices**: If you have registered multiple Openloop devices in Connect, passkey registration and authentication are performed against **the device currently selected in Connect** (the OS picker shows only "Openloop Connect" and does not let you choose which device). To use a different device, **switch the selection** in Connect's device management screen before registering or authenticating.

Registration steps

1. Open the target site in iOS Safari / Android Chrome
2. Choose the passkey registration menu (e.g. Google: "Create a passkey")
3. The OS Credential Manager picker appears:
   - **iOS**: Select "Openloop Connect" (it operates over BLE even though the picker shows "Connect")
   - **Android**: Select "Openloop Connect"
4. Connect begins communicating with Openloop over BLE, and a registration request appears on the device screen
5. Press the **Approve** button on the device
6. Registration complete

2.9 Authenticating with a passkey — BLE (via Openloop Connect)

1. On the target site's login screen in the browser, choose passkey authentication
2. Select **Openloop Connect** in the OS picker
3. Connect communicates with Openloop over BLE, and an authentication request appears on the device screen
4. Press the **Approve** button on the device
5. Login complete

> ℹ️ **Transport icon display**: On the Openloop device's passkey list screen, a 🔌 (USB) or 📶 (Bluetooth) icon appears to the left of each credential, so you can tell which transport it was registered over.

2.10 Managing credentials

You can manage registered passkeys on the Openloop device.

- **View list**: Settings > Passkey > Credential list
  - Each credential shows a transport icon (🔌 USB / 📶 Bluetooth)
- **Delete individually**: Select a credential and delete it
- **Delete all (on the device)**: Settings > Passkey > Passkey reset
- **Delete all (from the host / CTAP2 Reset)**: You can also initialize FIDO2 from the PC/browser side (**over USB only**; a confirmation appears on the device screen at execution, so approve it). All credentials and the FIDO PIN are erased.
  - **Chrome / Edge**: Settings > Privacy and security > Manage security keys > Reset (`chrome://settings/securityKeys`)
  - **CLI (libfido2)**: `fido2-token -R <device>`

> ⚠️ Performing a passkey reset (either on the device or from the host) deletes all credentials. You will need to re-register with each website.

2.11 Supported browsers & platforms

USB CTAPHID path

OS	Chrome	Edge	Safari	Firefox
Windows	✓	✓	-	✓

OS	Chrome	Edge	Safari	Firefox
macOS	✓	-	△	△

- **Windows**: WebAuthn registration and authentication work normally in all browsers (via the OS's built-in webauthn.dll)
- **macOS Chrome**: Registration and authentication work normally
- **macOS Safari/Firefox**: Registration and authentication work normally. If the user does not approve on the device side, the browser may keep waiting for a response (a platform-side limitation). You can recover with the browser's "Cancel" button.

BLE (via Openloop Connect) path

OS	Browser	Openloop Connect required
iOS 17+	Safari (via Credential Provider Extension)	✓
Android 14+	Chrome / Edge (via Credential Provider Service)	✓

> ⚠ On iOS, the browser's passkey picker shows "Openloop Connect," but this is because iOS fixes an Extension's display name to the parent app name (ignoring CFBundleDisplayName); **the actual communication happens over BLE**.

2.12 Examples of supported services

Major websites and services that support passkeys:

- **Google** account
- **Microsoft** account
- **GitHub**
- **Cloudflare**
- **AWS** (IAM)
- **1Password**
- and many other FIDO2/WebAuthn-capable services

> ℹ You can find the latest list of supporting services at passkeys.directory.

3. PIV/PKCS#11

3.1 Overview

Openloop implements the PIV (Personal Identity Verification, NIST SP 800-73) smart card interface. Through the PKCS#11 shared library, it covers use cases that passkeys cannot, such as SSH authentication and TLS client authentication.

3.2 Architecture

```
SSH / Firefox / pkcs11-tool
  ↓ PKCS#11 C API
libopenloop-pkcs11 (.dylib / .dll / .so)
  ↓ PIV APDU → WebSocket (ws://127.0.0.1:21320)
Openloop Connect (transparent APDU forwarding)
  ↓ USB HID
Openloop device (PIV handler → signing with SE050)
```

Because the PKCS#11 library communicates with the device via Openloop Connect, **Openloop Connect must be running** when you use the PIV/PKCS#11 functions.

3.3 Enabling PIV/PKCS#11

Enable PIV under Settings > Passkey & PIV > PIV ON. Turning it OFF hides the PKCS#11 slots.

Also check the following settings:

1. Install and launch **Openloop Connect**
2. On the Openloop device, set **Settings** > **USB settings** > **USB HID** to **ON**
3. Connect Openloop over USB and confirm that Connect shows the device as Device Connected

3.4 PIV PIN setup

The PIV PIN is **optional**. Setting a PIN enables PIN authentication over USB, and when the PIN is sent, signing happens without a confirmation screen (blind signing). If you do not set a PIN, on-device confirmation is required as before.

Dual mode

Openloop's PIV implementation offers two operating modes depending on whether the PIN is sent.

Mode	Behavior	Use
PIN sent	Blind signing (no confirmation screen, automatic signing)	Automation / CI/CD / scripts
PIN not sent	Traditional confirmation flow (approve on the device screen)	Interactive use

> You can use both modes with a single device. Whether the PIN is sent is controlled on the client side (SSH configuration, etc.).

How to set the PIN

```
# Initial PIN setup
pkcs11-tool --module "$MODULE" --login --init-pin --new-pin 123456

# Change the PIN
pkcs11-tool --module "$MODULE" --login --change-pin --pin 123456 --new-pin 654321
```

Deleting the PIN

To delete the PIN, use one of the following methods:

- **Device UI**: Settings > Passkey & PIV > touch the 🗑 button next to PIV PIN
- **USB APDU**: Send the RESET RETRY COUNTER command

If the PIN is locked

Entering the wrong PIN **8 times in a row** locks it. If it becomes locked, delete the PIN with the 🗑 button in the device UI and set it again.

SSH usage example (with PIN)

```
# SSH connection with a PIN
ssh -o "PKCS11Provider=$MODULE" -o "PKCS11Pin=123456" user@host
```

> ⚠ Writing the PIN directly on the command line may leave it in the shell history or the process list. In security-critical environments, consider using ssh-agent or the PIN-not-sent mode (the on-device confirmation flow).

---

4. PKCS#11 library paths

The PKCS#11 library is bundled with Openloop Connect. It is in the `pkcs11/` folder under the app installation path:

OS	Relative to the app path
macOS	`<app>/Contents/Resources/pkcs11/libopenloop-pkcs11.dylib`
Windows	`<app>/resources/pkcs11/libopenloop-pkcs11.dll`
Linux	`<app>/resources/pkcs11/libopenloop-pkcs11.so`

The app's install location `<app>` varies by installation method:

Installation method	Install location
macOS (DMG)	`/Applications/Openloop Connect.app`
Windows (official installer)	`C:\Program Files\Openloop Connect`
Windows (Microsoft Store)	In PowerShell: `(Get-AppxPackage 7CA75049.OpenloopConnect).InstallLocation`
Linux (deb)	`/opt/Openloop Connect`

The examples that follow set the full path of the macOS DMG build in a shell variable:

```
MODULE="/Applications/Openloop Connect.app/Contents/Resources/pkcs11/libopenloop-pkcs11.dylib"
```

---

5. SSH public-key authentication

Use Openloop as an SSH hardware key via PKCS#11. The private key is kept inside the device's SE050 secure element and can never be extracted.

5.1 Preparing a key

Generate a key pair in a PIV slot (default: 9A Authentication). Generate the key by either of the following:

- **Demo page**: Generate it from the web interface at PIV Demo
- **pkcs11-tool**: Generate it from the command line (see Chapter 7)

5.2 Retrieving the public key

```
# Retrieve the SSH public key from the device
ssh-keygen -D "$MODULE"

# Save it to a file
ssh-keygen -D "$MODULE" > ~/openloop_key.pub
```

Add the retrieved public key to the remote server's `~/.ssh/authorized_keys`.

5.3 SSH connection

```
# A one-off connection
ssh -I "$MODULE" user@hostname
```

5.4 Persistent configuration (~/.ssh/config)

Instead of specifying the `-I` option every time, you can write the configuration in `~/.ssh/config`.

```
# Apply to a specific host
Host myserver
    HostName 192.168.1.100
    User ubuntu
    PKCS11Provider /Applications/Openloop Connect.app/Contents/Resources/pkcs11/libopenloop-pkcs

# Apply to all hosts
Host *
    PKCS11Provider /Applications/Openloop Connect.app/Contents/Resources/pkcs11/libopenloop-pkcs
```

5.5 SSH implementation compatibility and the OpenSSH recommendation

Openloop's PIV/PKCS#11 supports three algorithms: P-256, Ed25519, and RSA-2048 (RSA-2048 supported). **The algorithms usable over PKCS#11 differ by SSH implementation.**

SSH implementation	RSA-2048	P-256 (ECDSA)	Ed25519 (EdDSA)	Notes
OpenSSH (8.5+)	✓	✓	✓	Recommended. Full support for all algorithms.
macOS system SSH (`/usr/bin/ssh`)	✓	△	✗	Only RSA is stable over PKCS#11.
PuTTY (Windows)	✓	△	✗	Limited PKCS#11 support. RSA-centric.
Dropbear	✗	✗	✗	No PKCS#11 support.

> ⚠ **We strongly recommend using OpenSSH (the Homebrew build).** The macOS system SSH (the Apple build) has incomplete ECDSA/EdDSA support over PKCS#11. Using an RSA-2048 key avoids the compatibility issues.

```
# Install Homebrew OpenSSH
brew install openssh

# Use the Homebrew build (specify the full path)
/opt/homebrew/bin/ssh -I "$MODULE" user@hostname

# Check the version (confirm 8.5 or later)
/opt/homebrew/bin/ssh -V

# Add IgnoreUnknown UseKeychain in ~/.ssh/config
# (Suppresses warnings about the macOS-specific option that Homebrew SSH does not recognize)
```

5.6 Debugging

```
# Enable debug output from the PKCS#11 library
OPENLOOP_PKCS11_DEBUG=1 ssh-keygen -D "$MODULE"

# Verbose SSH connection log
ssh -vvv -I "$MODULE" user@hostname
```

---

6. Firefox TLS client authentication

Why Firefox

Among the major browsers, **Firefox is the only one that supports loading an external PKCS#11 module**. Chrome, Edge, and Safari use only the OS's built-in keychain/certificate store and provide no way to load a third-party PKCS#11 library directly. That is why TLS client authentication using Openloop's PKCS#11 library is done in Firefox.

Register the PKCS#11 module in Firefox to use TLS client-certificate authentication (mTLS).

6.1 Registering the PKCS#11 module

1. Open Firefox **Settings**
2. Go to **Privacy & Security**
3. In the **Certificates** section, click **Security Devices…**
4. Click **Load**
5. **Module Name**: enter `Openloop`
6. **Module filename**: browse to and select the library path above
7. Click **OK**

6.2 How to verify

1. In the Security Devices dialog, expand the **Openloop** module
2. Confirm the slot that shows token information
3. Click **View Certificates** (Settings > Certificates > View Certificates)
4. The Openloop certificate appears on the **Your Certificates** tab

6.3 mTLS authentication flow

When you access a website that requests a client certificate, Firefox automatically prompts you to select the certificate from the Openloop device. The signing operation requires User Presence (a physical touch on the device).

6.4 Automatic registration (macOS)

On macOS, Openloop Connect can automatically register the PKCS#11 module in Firefox by placing a manifest at the following path:

```
~/Library/Application Support/Mozilla/PKCS11Modules/openloop_pkcs11.json
```

When automatic registration is active, manual registration is unnecessary.

---

7. pkcs11-tool reference

You can use OpenSC's `pkcs11-tool` command to work with the device's keys and objects.

Installation

```
# macOS
brew install opensc

# Ubuntu/Debian
sudo apt install opensc

# Windows
# Install from the OpenSC installer: https://github.com/OpenSC/OpenSC/releases
```

List slots and tokens

```
pkcs11-tool --module "$MODULE" -T
```

List objects

```
pkcs11-tool --module "$MODULE" -O
```

Generate a key pair

```
# Generate P-256 in slot 9A (Authentication)
pkcs11-tool --module "$MODULE" --keypairgen \
  --key-type EC:prime256v1 \
  --id 01 --label "PIV AUTH"

# Generate Ed25519 in slot 9A
pkcs11-tool --module "$MODULE" --keypairgen \
  --key-type EC:edwards25519 \
  --id 01 --label "PIV AUTH"
```

Signing test

```
# Create test data and sign it with the key in slot 9A
echo "test data" | openssl dgst -sha256 -binary > /tmp/hash.bin
pkcs11-tool --module "$MODULE" --sign \
  --mechanism ECDSA \
  --id 01 \
  --input-file /tmp/hash.bin \
  --output-file /tmp/sig.bin
```

Deleting a key

```
pkcs11-tool --module "$MODULE" --delete-object \
  --type privkey --id 01
```

---

8. PIV slots & algorithms

8.1 PIV slots

Slot	Name	Use
`9A`	PIV Authentication	SSH authentication, general authentication. The most frequently used.
`9C`	Digital Signature	Document signing, S/MIME. Always requires a PIN.
`9D`	Key Management	Encryption/decryption, key agreement.
`9E`	Card Authentication	Physical access, contactless authentication. No PIN required.

8.2 Supported algorithms

Algorithm	PIV ID	Notes
P-256 (secp256r1)	0x11	ECDSA over NIST P-256. Widely supported.
Ed25519	0x22	EdDSA over Curve 25519. SSH requires OpenSSH 8.5 or later.
RSA-2048	0x07	PKCS#1 v1.5 signatures. Supports SSH/GPG/PDF signing. Key generation takes about 10 seconds.

8.3 PKCS#11 object-ID mapping

PIV slot	PKCS#11 ID	CKA_LABEL	Use
9A	01	PIV AUTH	Authentication
9C	02	SIGN	Digital Signature
9D	03	KEY MGMT	Key Management

PIV slot	PKCS#11 ID	CKA_LABEL	Use
9E	04	CARD AUTH	Card Authentication

9. Troubleshooting

The passkey isn't recognized

- Check that **Settings** > **USB settings** > **USB HID** is ON on the device
- Check that **Settings** > **Passkey** is ON on the device
- Check that the USB cable supports data (charge-only cables won't work)
- Try reconnecting the device

The PKCS#11 library can't be found

- Check that Openloop Connect is installed
- Check that the library path is correct (see Chapter 4)

`ssh-keygen -D` shows no keys

- Check that Openloop Connect is running
- Check that the device is connected to Connect (Device Connected shown)
- Check that a key has been generated in the PIV slot (verify with pkcs11-tool -O)
- On macOS, check that you are using the Homebrew build of OpenSSH

Firefox can't load the module

- Check that Openloop Connect is running
- Check that the library file path is correct
- Try restarting Firefox

Signing times out

- Check whether a signing-confirmation dialog is showing on the device screen
- Operations that require User Presence (a physical touch) need you to touch the device

Collecting debug logs

You can see the details of the communication by enabling debug output from the PKCS#11 library:

```
OPENLOOP_PKCS11_DEBUG=1 ssh-keygen -D "$MODULE"
```

10. References

FIDO2 / WebAuthn

Resource	URL
CTAP2 specification (FIDO Alliance)	https://fidoalliance.org/specs/fido-v2.1-ps-20210615/fido-client-to-authenticator-protocol-v2.1-ps-20210615.html
WebAuthn specification (W3C)	https://www.w3.org/TR/webauthn-2/
U2F specification (FIDO Alliance)	https://fidoalliance.org/specs/fido-u2f-v1.2-ps-20170411/
CTAPHID — USB transport	https://fidoalliance.org/specs/fido-v2.1-ps-20210615/fido-client-to-authenticator-protocol-v2.1-ps-20210615.html#usb
List of passkey-capable services	https://passkeys.directory

PIV / PKCS#11

Resource	URL
NIST SP 800-73 (PIV specification)	https://csrc.nist.gov/pubs/sp/800/73/4/final
PKCS#11 specification (OASIS)	https://docs.oasis-open.org/pkcs11/pkcs11-base/v3.0/pkcs11-base-v3.0.html
OpenSC (pkcs11-tool)	https://github.com/OpenSC/OpenSC
OpenSSH PKCS#11 documentation	https://man.openbsd.org/ssh-keygen#D

Openloop-related

Resource	URL
Openloop PIV/PKCS#11 demo	https://crypto.haudi.jp/openloop/demo/piv/
Openloop passkey demo	https://crypto.haudi.jp/openloop/demo/ctap2/
Openloop Connect	https://crypto.haudi.jp/openloop/