YubiKey 5 Series Technical Manual

Yubico

Sep 14, 2021

CONTENTS

CHAPTER

ONE

INTRODUCTION TO THE YUBIKEY 5 SERIES

The YubiKey 5 Series security keys offer strong authentication with support for multiple protocols, including FIDO2, which is the new standard that enables the replacement of password-based authentication. The YubiKey strengthens security by replacing passwords with strong hardware-based authentication using public key cryptography.

- For those who just want to use a YubiKey without programming anything, the most useful part of this guide will be *Understanding the USB Interfaces*, which describes how the YubiKey connects, and indicates what it can connect to.

  For an overview on setting up two-step verification in a typical case, see Google on using a security key for 2-step verification.

- The full list of the services that work with YubiKeys is on Yubico's Works With YubiKey page.
- Most of the rest of this guide targets systems integrators, IT teams, or developers who expect to integrate support for YubiKeys into their environment.

All the YubiKeys in the YubiKey 5 Series have the basic functionalities and capabilities described in this guide. However, it is the firmware version that determines which of the more specialized functionalities and capabilities are available to your YubiKey.

CHAPTER

TWO

WHAT'S NEW?

The capabilities of the YubiKey 5 Series are dependent on the different combinations of firmware + connector type + protocol. This section covers connector types (form factors). Capabilities brought to the YubiKey 5 Series by the firmware are covered in *5.4.X Firmware: Overview of Features & Capabilities* and in *Protocols and Applications*.

2.1 YubiKey 5Ci

The YubiKey 5Ci is the first hardware authenticator of its kind enabled with both USB-C and Lightning® connectors on a single security key. With multi-protocol capabilities that support OTP, U2F, *FIDO2*/WebAuthn, and Smart Card requirements, the YubiKey 5Ci provides a unified solution for secure logins on mobile and computing devices.

2.1.1 Lightning® Connector

The YubiKey 5Ci introduced support for Apple's Lightning® connector. All features of the YubiKey 5 are supported over Lightning®, including FIDO2, PIV, OpenPGP, OATH and OTP. The YubiKey 5Ci is the first YubiKey to roll out new feature enhancements to FIDO2 and OpenPGP. Details on the new functionality can be found in the Enhancements to FIDO 2 Support and in the Enhancements to OpenPGP Support.

Like the USB interface, the YubiKey 5Ci's Lightning® interface also uses a variety of channels for communication between the YubiKey and iOS.

The YubiKey 5Ci presents itself as an Apple iOS peripheral. It is able to interact with:

- Any iOS app using the Yubico YubiKey iOS SDK
- Any app input data field via touch-triggered OTP.
- Any WebAuthn-compliant application (starting in iOS 13). This includes the Safari browser.

When connecting the YubiKey 5Ci via Lightning®, the **interfaces enabled** setting is common to both USB-C and Lightning®. Enabling or disabling an interface will apply to both connections.

Note: **Developers**: for apps within iOS to be able to use advanced protocols that send and receive information from the YubiKey 5Ci, the Yubico SDK is required (the Yubico iOS SDK can be accessed at https://github.com/YubicoLabs/yubikit-ios) and the app registered with Yubico. This can be done via the Yubico iOS SDK App submission page.

The USB and iProduct strings that show up when connecting via Lightning® or USB are specific to the connection type. They are described in the YubiKey USB ID Values guide.

2.1.2 iPad and iPad Pro

For users of keys in the YubiKey 5 Series, because the iPad Pro does not have a Lightning port, support depends on what you want to do. The second part of this article covers all those aspects: Can I use my YubiKey with iPads?

From the developer perspective, support for the iPad Pro has some limitations. Consult Supporting U2F or FIDO2 Security Keys on iOS or iPadOS | Security Key Compatibility for detailed instructions on working around those limitations.

2.2 NFC

Expanding the options for quick tap-n-go authentication across desktops, laptops, and mobile devices, all of the applications - including *FIDO2* - are available over NFC.

The YubiKey 5 NFC and YubiKey 5C NFC support the iPhone 7 and newer.

Background tag reading is supported in the iPhone XS and newer.

The YubiKey 5 NFC and YubiKey 5C NFC provide an NFC wireless interface In addition to USB. The YubiKey 5 NFC and YubiKey 5C NFC include the RFID standard specific to the ISO/IEC 14443-A and ISO/IEC 14443-4 NFC format; RFID implementations not included in the listed ISO standards are not supported.

The NDEF URI has been updated to a new format; an example of the new format is provided below. The `<OTP>` value is replaced with the OTP generated by the YubiKey.

https://demo.yubico.com/yk/

For operations that require a touch, all touch requests within the first 20 seconds of the operation will succeed. After a period of inactivity, a YubiKey placed on a desktop NFC reader may power down to help prevent unintended access. To regain connectivity with an NFC reader, remove the YubiKey from the reader and reposition it on the reader. Some NFC readers may power-cycle and in doing so, prevent the YubiKey from powering down.

2.3 USB

All of the models in the YubiKey 5 Series provide a USB 2.0 interface, regardless of the form factor of the USB connector. The YubiKey will present itself as a USB composite device in addition to each individual USB interface.

The USB PID and iProduct string changes depending on which of the USB interfaces are enabled. They are described in the YubiKey USB ID Values guide.

See also *Understanding the USB Interfaces*.

CHAPTER

THREE

5.4.X FIRMWARE: OVERVIEW OF FEATURES & CAPABILITIES

The YubiKey firmware is separate from the YubiKey itself in the sense that it is put onto each YubiKey in a process separate from the manufacture of the physical key. Nonetheless, it can be neither removed nor altered. Yubico periodically updates the YubiKey firmware to take advantage of features and capabilities introduced into operating systems such as Windows, MacOS, and Ubuntu, etc., as well as to enable new YubiKey features.

The firmware version on a YubiKey or an HSM therefore determines whether or not a feature or a capability is available to that device. The quickest and most convenient way to determine your device's firmware version is to use the YubiKey Manager tool (ykman), a lightweight software package installable on any OS. The YubiKey Manager has both a graphical user interface (GUI) and a command line interface (CLI).

- Download the YubiKey Manager tool: https://www.yubico.com/products/services-software/download/yubikey-manager/
- YubiKey Manager (ykman) CLI & GUI Guide

Depending on when they were manufactured, the YubiKey 5 Series keys have the 5.2.3 firmware or the 5.4.X firmware. The capabilities of the firmware versions 5.2.3 and earlier are included in the general descriptions in *Protocols and Applications*.

The features, capabilities, and enhancements brought to the YubiKey 5 Series by the 5.4.X firmware are **summarized** below, with the full details given in the technical description sections in this manual.

3.1 Secure Channel

Secure channel is used for establishing an authenticated and encrypted communication channel over CCID between a host and the secure element on the YubiKey. The YubiKey security domain can store three concurrent long-lived transport key sets.

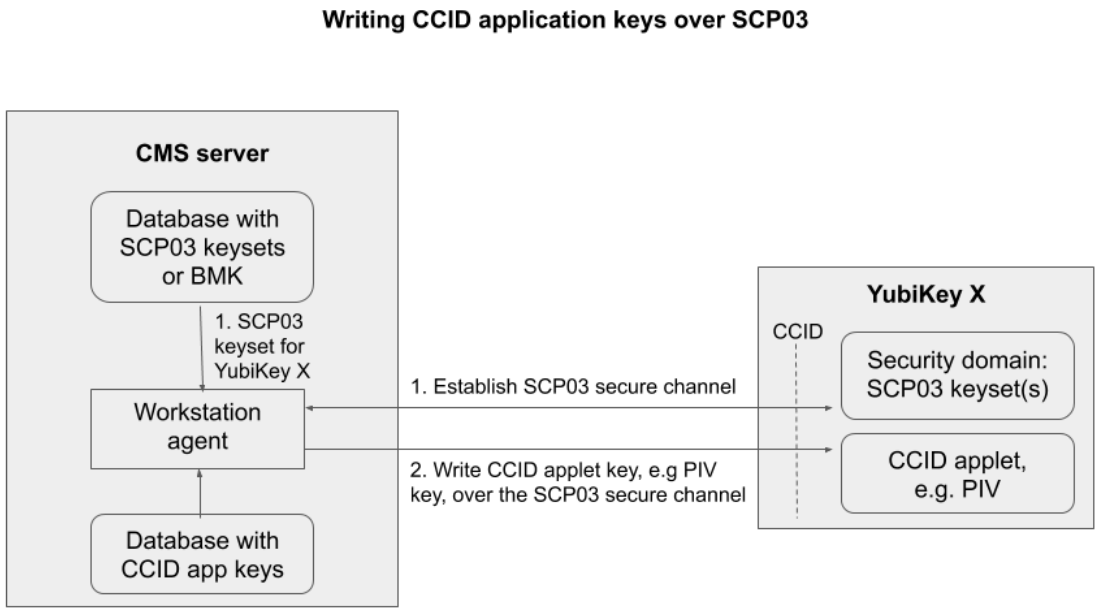
SCP03 (Secure Channel Protocol 03), which is part of the GlobalPlatform standard, is a framework for mutual authentication and encrypted transport between hosts and secure elements in smart cards. With the 5.4.X firmware, Yubico has implemented a secure channel based on that specification.

For an **overview** of this implementation, see *Secure Channel*, while **detailed descriptions** of the secure channel feature are to be found in *Yubico Secure Channel Technical Description*, *Secure Channel Key Diversification and Programming*, and *Yubico SCP03 Developer Guidance*.

Note: Applications based on PKCS #11 or Microsoft CNG do not usually use the secure channel.

3.1.1 Security Domains & Key Diversification

The authenticated and encrypted communication channel takes place over the CCID interface between a host and the secure element on the YubiKey. This includes programming or communication from CCID functions. The secure channel feature can therefore be used to load application keys onto the YubiKey to be used with the CCID applications OATH, OpenPGP, or PIV.


Writing CCID application keys over SCP03
CMS server
Database with SCP03 keysets or BMK
1. SCP03 keyset for YubiKey X
Workstation agent
Database with CCID app keys
1. Establish SCP03 secure channel
2. Write CCID applet key, e.g PIV key, over the SCP03 secure channel
YubiKey X
CCID
Security domain: SCP03 keyset(s)
CCID applet, e.g. PIV

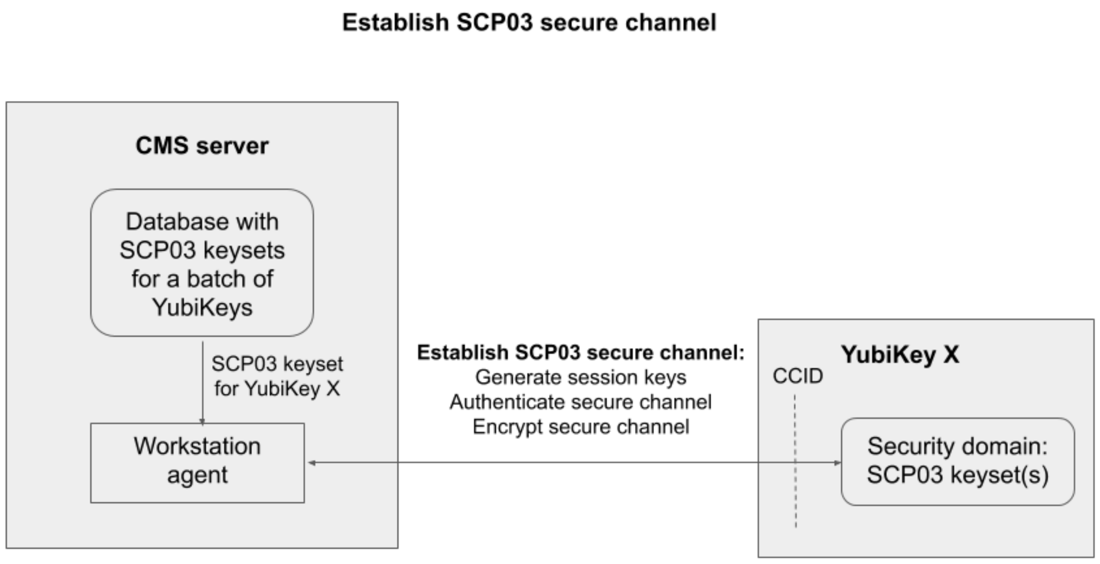
Writing CCID Application Keys over SCP03

The YubiKey security domain can store three concurrent transport key sets. A transport key set contains three long-lived AES keys. When a session is established, the session AES keys are derived from the long-lived transport key set.

Key diversification is the process of deriving a secure channel static transport key set from a Batch Master Key (BMK), the YubiKey identifier (part of serial number), and additional metadata. Key diversification therefore facilitates secure distribution of key sets over a secure channel. To derive the YubiKey transport key sets, the Batch Master Key (BMK) is shared with the CMS system. If the CMS vendor gives Yubico access to its BMK, Yubico can preprogram the secure channel transport key sets for the YubiKey 5 batches. The BMK could be protected by the YubiHSM2.

In order to import new transport key sets, a secure channel must be established with the security domain. This has to be done with a previously loaded transport key set or the default transport key set. Each secure channel transport key set is protected by being written to the YubiKey security domain in the secure element and stored in a server, typically a CMS system. The host that is accessing the YubiKey has an agent that connects to the CMS system to retrieve the secure channel key set. Based on the secure channel key set, both at the host and the YubiKey, a secure session is established.

Establish SCP03 Secure Channel


Establish SCP03 secure channel
CMS server
Database with SCP03 keysets for a batch of YubiKeys
SCP03 keyset for YubiKey X
Workstation agent
Establish SCP03 secure channel:
Generate session keys
Authenticate secure channel
Encrypt secure channel
YubiKey X
CCID
Security domain: SCP03 keyset(s)

3.1.2 Benefits and Usage

- Encryption application keys can be stored on the CMS server as well as on the YubiKey. If the YubiKey is lost or compromised, the encryption key can be recovered and loaded onto a replacement YubiKey.
- Key diversification enables simplified and secured distribution of secure channel transport key sets as only the BMK must be shared with the CMS system to derive the YubiKey transport key sets.
- The secure channel transport key sets can be preprogrammed at the YubiKey batches by Yubico, if the Yubico supply chain has access to the BMK of the CMS vendor.
- The CMS system can generate the secure channel transport key sets based on the YubiKey serial numbers, the BMK, and additional metadata. The CMS can then use the initial secure channel transport key set for writing additional secure channel transport key sets to the YubiKeys.

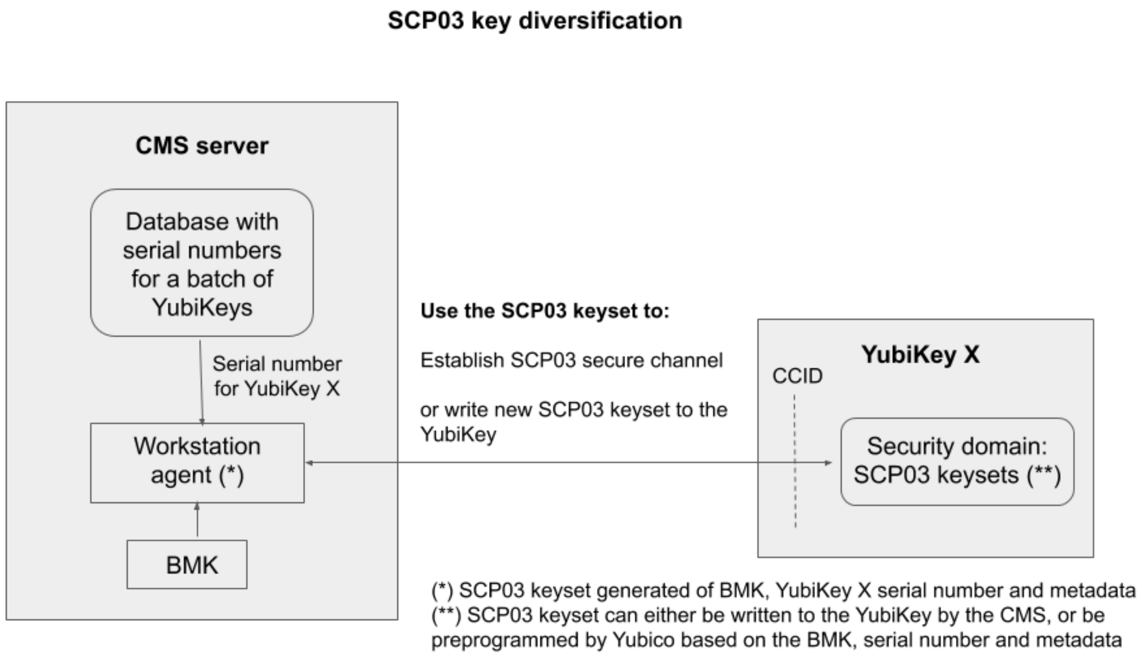
SCP03 Key Diversification

For more technical information, see *Secure Channel Key Diversification and Programming*.

3.1.3 Secure Channel CPLC Data

The Card Production Life Cycle (CPLC) data object is a random dataset that is stored on each YubiKey to assure unique identification of the YubiKeys in CMS. Although it is officially deprecated from the SCP03 standard, it is still widely used to hold card data specific to CMS services or to uniquely identify smart cards. Therefore Yubico has implemented the CPLC data object to provide unique identification of YubiKeys for CMS vendors.

For a more detailed description of CPLC data object, see *Secure Channel CPLC Data*.


SCP03 key diversification
CMS server
Database with serial numbers for a batch of YubiKeys
Serial number for YubiKey X
Workstation agent (*)
BMK
Use the SCP03 keyset to:
Establish SCP03 secure channel
or write new SCP03 keyset to the YubiKey
YubiKey X
CCID
Security domain: SCP03 keysets (**)
(*) SCP03 keyset generated of BMK, YubiKey X serial number and metadata
(**) SCP03 keyset can either be written to the YubiKey by the CMS, or be preprogrammed by Yubico based on the BMK, serial number and metadata

3.2 PIV Enhancements

3.2.1 YubiKey PIV Metadata

YubiKey 5 PIV metadata enables services and client software to obtain information about PIV keys from a central location, obviating the need to query PIV attestation. This enables the YubiKey PIV application to report on characteristics of cryptographic keys in the specified PIV slot. YubiKey PIV metadata thereby facilitates integration with CMS vendors.

PIV metadata was introduced with the YubiKey 5.3.0 firmware. For details, see the Get Metadata section of the PIV extensions on developers.yubico.com.

3.2.2 PIV Management Key (AES)

Prior to the release of the 5.4.x firmware, the PIV management key was a **3DES** key. This feature expands the management key type held in PIV slot 9b to include AES keys (128, 192 and 256) as defined in SP 800-78-4 Cryptographic Algorithms and Key Sizes for Personal Identity Verification <SP800-78-4, section 5). The PIV management key in AES format enables current and future FIPS-compliant CMS services.

For more technical information, see *PIV Enhancements Technical Description*.

3.3 NFC ID: Calculation Changed

Crucial to vendors of physical access control systems and door protection systems utilizing NFC readers, the modification of the YubiKey NFC ID calculation enables NFC readers and access management systems (door locks) using the NFC ID tag to identify NFC-enabled YubiKeys, including those without serial numbers. It is now calculated so that a unique string is returned in the first part of the NFC ID. Until the release of the 5.4.X firmware, the fact that some access control systems truncate the YubiKey NFC ID meant that YubiKey 5 NFC IDs appeared to be non-unique.

For more technical information on this, see *NFC ID Calculation Technical Description*.

3.4 YubiHSM Auth

YubiHSM Auth is a YubiKey CCID application that stores the long-lived credentials used to establish secure sessions to a YubiHSM 2. The secure session protocol is based on Secure Channel Protocol 3 (SCP03). YubiHSM Auth is supported by YubiKey v5.4.0 and higher.

For more details, see *YubiHSM Auth*.

YubiHSM Auth is a CCID application that can store long-lived credentials (AES keys) that are used to establish secure sessions to a YubiHSM 2. By providing an external challenge, a derivation scheme that yields three session keys is executed. The session keys are not stored on the YubiKey but simply output as a result. The session keys can be used for authentication to the YubiHSM 2. The authentication scheme is based on SCP03 (see *Secure Channel* above). Each long-lived YubiHSM Auth credential is protected by a user access code that has to be provided to authenticate each session. Storing and deleting credentials requires a separate admin access code.

3.4.1 Benefits and Usage

YubiHSM Auth enables the secure storage of the long-lived credentials for accessing a YubiHSM 2. The existing authentication solution for the YubiHSM 2 is based on software credentials derived from the Password-Based Key Derivation Function 2 (PBKDF2) algorithm with a password as input.

Generating keys using PBKDF2 is just for convenience. It is preferable - and recommended - to provide AES keys directly to avoid exposing them to attack. Not only is it important to avoid losing the derivation password or the keys themselves (as those are basically the same thing), but those credentials also

- Exist outside a secure element and
- Need to be given in clear text to the program that uses them loads them into memory.

With YubiHSM Auth only the ephemeral session keys exist outside a secure environment.

For more details, see *YubiHSM Auth*.

CHAPTER

FOUR

PHYSICAL ATTRIBUTES

The serial number is printed on the back of every YubiKey in the 5 Series. The 2D barcode (QR code) of the serial number is also printed on the back of every YubiKey in the 5 Series except the YubiKey 5C Nano, which is too small to accommodate the 2D barcode.

4.1 YubiKey 5 NFC

- Dimensions: 18mm x 45mm x 3.3mm
- Weight: 3g
- Physical Interfaces: USB, NFC
- Operating Temperatures: 0 °C - 40 °C (32 °F - 104 °F)
- Storage Temperatures: -20 °C - 85 °C (-4 °F - 185 °F)

4.2 YubiKey 5 Nano

- Dimensions: 12mm x 13mm x 3.1mm
- Weight: 1g
- Physical Interfaces: USB

- Operating Temperatures: 0 °C - 40 °C (32 °F - 104 °F)
- Storage Temperatures: -20 °C - 85 °C (-4 °F - 185 °F)

4.3 YubiKey 5C

- Dimensions: 12.5mm x 29.5mm x 5mm
- Weight: 2g
- Physical Interfaces: USB
- Operating Temperatures: 0 °C - 40 °C (32 °F - 104 °F)
- Storage Temperatures: -20 °C - 85 °C (-4 °F - 185 °F)

4.4 YubiKey 5C Nano

- Dimensions: 12mm x 10.1mm x 7mm
- Weight: 1g
- Physical Interfaces: USB
- Operating Temperatures: 0 °C - 40 °C (32 °F - 104 °F)
- Storage Temperatures: -20 °C - 85 °C (-4 °F - 185 °F)

4.5 YubiKey 5Ci

- Dimensions: 12mm x 40.3mm x 5mm
- Weight: 2.9g
- Physical Interfaces: USB, Lightning®
- Operating Temperatures: 0 °C - 40 °C (32 °F - 104 °F)
- Storage Temperatures: -20 °C - 85 °C (-4 °F - 185 °F)

4.6 YubiKey 5C NFC

- Dimensions: 18mm x 45mm x 3.7mm
- Weight: 4g
- Physical Interfaces: USB, NFC
- Operating Temperatures: 0 °C - 40 °C (32 °F - 104 °F)
- Storage Temperatures: -20 °C - 85 °C (-4 °F - 185 °F)

CHAPTER

FIVE

UNDERSTANDING THE USB INTERFACES

USB interfaces are the different channels that software can use to communicate with the YubiKey when it is connected via USB. Each interface enables a specific set of applications on the YubiKey; if an interface is disabled, none of the applications that use that interface will be available.

Note: With previous YubiKeys and older Yubico firmware, the USB interfaces were referred to as "modes" and the FIDO interface was called the "U2F mode".

5.1 OTP

The OTP interface presents itself to the operating system as a USB keyboard. The OTP application is accessible over this interface. Output is sent as a series of keystrokes from a virtual keyboard. This allows for OTP to be used in any environment which can accept standard keyboard input.

The OTP interface is supported natively across all desktop OS environments (macOS, Windows, Linux) as well as on mobile OS platforms (iOS, Android). Output is sent as a series of keystrokes from a virtual keyboard, allowing the OTP application to work with any environment that supports USB keyboard input.

5.2 FIDO

The FIDO interface provides access to the FIDO2 and U2F applications.

The FIDO interface presents itself as a generic human interface device (HID). The FIDO interface is supported on all desktop platforms running WebAuthn-compatible browsers or applications, as well as on Android and iOS (starting in iOS 13).

5.3 CCID

The CCID interface provides communication for the PIV / Smart Card, OATH (HOTP and TOTP), and OpenPGP applications.

The YubiKey presents itself to the operating system as a USB smart card reader. Each of the applications presents itself as a separate smart card attached to that reader. The CCID interface is supported on Windows and MacOS, and on Linux with the PC/SC package. CCID is also supported on Android.

Note: Developers: to access the CCID interface on iOS, the Yubico iOS SDK is required.

CHAPTER

SIX

PROTOCOLS AND APPLICATIONS

The YubiKey 5 Series provides applications for a wide variety of authentication options: FIDO2, OATH, OpenPGP, OTP, Smart Card, U2F. The applications are all separate from each other, with separate storage for keys and credentials. This section references enhancements provided by the YubiKey 5.4.X firmware.

For information on managing these applications, see *Tools and Troubleshooting*.

6.1 FIDO2

The FIDO2 standard offers the same high level of security as FIDO U2F, since it is based on public key cryptography. In addition to providing phishing resistant two-factor authentication, the FIDO2 application on the YubiKey allows for the storage of resident credentials. As the resident credentials can accommodate the username and other data, this enables truly passwordless authentication on sites and applications that support the WebAuthn protocol. YubiKeys in the 5 Series can hold up to 25 resident keys.

FIDO2 support is available to the iPad Pro via the USB-C or Lightning® connectors of the *YubiKey 5Ci*. FIDO2/WebAuthn can be achieved over USB-C using any of the following options:

- `ASWebAuthenticationSession`
- `SFSafariViewController`
- Redirect to Safari browser

For more details on support for the iPad Pro, see *iPad and iPad Pro* below, and to see which U2F/FIDO2 security keys currently work with iOS/iPadOS 13.3+ devices using the Safari browser in combination with apps using `SFSafariViewController` or `ASWebAuthenticationSession` - see Supporting U2F or FIDO2 Security Keys on iOS or iPadOS | Security Key Compatibility.

6.1.1 Locking FIDO2 Credentials

The resident credentials can be left unlocked and used for strong single-factor authentication, or they can be protected by a PIN for two-factor authentication.

- The FIDO2 PIN must be between 4 and 128 characters in length.
- Once a FIDO2 PIN is set, it can be changed but it cannot be removed without resetting the FIDO2 application.
- If the PIN is entered incorrectly 8 times in a row, the FIDO2 application will be locked. In order to restore this functionality, the FIDO2 application must be reset.

Note: Resetting the FIDO2 application will also reset the U2F key, so the YubiKey must be re-registered not only with all the FIDO2 sites, but also with all U2F sites.

Note: The YubiKey 5Ci supports Credential Management to allow for selective deletion of resident keys. See the guide to the Enhancements to FIDO 2 Support for details.

6.1.2 Default Values

PIN: None set.

6.1.3 AAGUID Values

See *FIDO2 AAGUIDs* for the AAGUIDs of all YubiKeys for the more recent firmware releases.

6.1.4 Supported Extensions

The YubiKey 5 Series supports only the AppID extension (`appid`) as defined by the W3C Web Authentication API specification. This extension allows U2F credentials registered using the legacy FIDO JavaScript APIs to be used with WebAuthn. That means that if you register a YubiKey in the 5 Series on a website that used U2F at that time and later upgrades to FIDO2, your U2F credentials will continue to work on the website.

6.2 FIDO U2F

FIDO U2F is an open standard that provides strong, phishing-resistant two-factor authentication for web services using public key cryptography. U2F does not require any special drivers or configuration to use, just a compatible web browser. The U2F application on the YubiKey can be associated with an unlimited number of U2F sites.

6.3 OATH

The OATH application can store up to 32 OATH credentials, either OATH-TOTP (time based) or OATH-HOTP (counter based). These credentials are separate from those stored in the OTP application, and can only be accessed via the CCID channel. In order to manage these credentials and read the OTPs generated by the YubiKey, the Yubico Authenticator is needed.

Access to the OTPs can be restricted by setting a password for this application.

Note: Developers: using the OATH application functions on iOS requires the Yubico iOS SDK.

6.3.1 OATH-HOTP

When an OATH-HOTP credential is programmed, the OTP is generated using the standard RFC 4226 HOTP algorithm and the YubiKey will automatically type the OTP. Optionally, the OTP can be prefixed by a public identity, conforming to the openauthentication.org Token Identifier Specification.

6.4 OpenPGP

The OpenPGP application provides an OpenPGP-compatible smart card in compliance with version 3.4 of the specification if the YubiKey firmware is 5.2.3 or later. If the firmware is an earlier version, the OpenPGP-compatible smart card is in compliance with version 2.0 of the specification.

OpenPGP-compatible smart card can be used with compatible PGP software such as GnuPG (GPG) and can store one PGP key each for authentication, signing, and encryption. Similar to the PIV / Smart Card touch policy, the OpenPGP application can also be set to require the YubiKey's metal contact be touched to authorize an operation.

Note: Developers: using the OpenPGP functions on iOS requires the Yubico iOS SDK.

YubiKey firmware 5.2.3 and above in combination with OpenPGP 3.4:

- Extends existing RSA support for OpenPGP operations to ECC algorithms
- Provides the Yubico Attestation feature for verifying keys generated on a YubiKey device
- Utilizes separate x.509 cardholder certificates alongside the existing OpenPGP certificates for authentication, signature and encryption/decipher
- Bring attestation functionality to OpenPGP keys and certificates generated on a YubiKey
- Improves security by supporting Key Derivation Function (KDF) PINs. With KDF enabled, the PIN is stored as a hash on the YubiKey. The OpenPGP client will only pass the hashed value, never the PIN directly.

6.4.1 Elliptic Curve Cryptographic (ECC) Algorithms

The YubiKey 5.2.3 firmware added support for ECC algorithms. These can be used for Signature, Authentication and Decipher keys. The full list of curves supported by OpenPGP 3.4 can be found in section 4.4.3.10 of the OpenPGP Smart Card 3.4 spec (page 35).

In addition to *RSA Algorithms*, YubiKeys support the following ECC algorithms:

- secp256r1
- secp256k1
- secp384r1
- secp521r1
- brainpoolP256r1
- brainpoolP384r1
- brainpoolP512r1
- curve25519
  - x25519 (decipher only)
  - ed25519 (sign / auth only)

For further details on the new features, including key attestation, expanded encryption algorithms and additional cardholder certificates, refer to Enhancements to OpenPGP Support.

6.4.2 RSA Algorithms

- RSA 1024
- RSA 2048
- RSA 3072 (requires GnuPG version 2.0 or higher)
- RSA 4096 (requires GnuPG version 2.0 or higher)

6.4.3 Default Values

- PIN: 123456
- Admin PIN: 12345678

6.5 OTP

The OTP application provides two programmable slots, each of which can hold one of the types of credentials listed below. A Yubico OTP credential is programmed to slot 1 during manufacturing. Output is sent as a series of keystrokes from a virtual keyboard.

- Trigger the YubiKey to produce the credential in the first slot by briefly touching the metal contact of the YubiKey.
- If a credential has been programmed to the second slot, trigger the YubiKey to produce it by touching the contact for 3 seconds.

6.5.1 Yubico OTP

Yubico OTP is a strong authentication mechanism that is supported by the YubiKey 5 Series. Yubico OTP can be used as the second factor in a two-factor authentication (2FA) scheme or on its own, providing single-factor authentication.

The OTP generated by the YubiKey has two parts, with the first 12 characters being the public identity which a validation server can link to a user, while the remaining 32 characters are the unique passcode that is changed each time an OTP is generated.

The character representation of the Yubico OTP is designed to handle a variety of keyboard layouts. It is crucial that the same code is generated if a YubiKey is inserted into a German computer with a QWERTZ layout, a French one with an AZERTY layout, or a US one with a QWERTY layout. The “Modhex”, or Modified Hexadecimal coding, was invented by Yubico to use only specific characters to ensure that the YubiKey works with the maximum number of keyboard layouts. (USB keyboards send their keystrokes by means of “scan codes” rather than the actual character. The translation to keystrokes is done by the device to which the YubiKey is connected).

6.5.2 Static Password

A static password can be programmed to the YubiKey so that it will type the password for you when you touch the metal contact.

For managing multiple passwords, see the password managers that the YubiKey can secure with two-factor authentication (2FA).

6.5.3 HMAC-SHA1 Challenge-Response

This type of credential is most often used for offline authentication, as it does not require contacting a server for validation.

An HMAC-SHA1 Challenge-Response credential enables software to send a challenge to the YubiKey and verify that an expected, predetermined response is returned. This credential can also be set to require a touch on the metal contact before the response is sent to the requesting software. This type of credential must be activated by the software sending the challenge; it cannot be activated by touching the metal contact on the YubiKey.

Note: Developers: as the Challenge-Response function requires two-way communication with the YubiKey, using this feature on iOS requires the Yubico iOS SDK.

6.6 Smart Card (PIV Compatible)

The YubiKey 5 Series provides a PIV-compatible smart card application. PIV, or FIPS 201, is a US government standard. It enables RSA or ECC sign/encrypt operations using a private key stored on a smart card through common interfaces like PKCS#11.

On Windows, the smart card functionality can be extended with the YubiKey Smart Card Minidriver.

Note: The YubiKey Smart Card Minidriver is not available for Android, Linux, macOS or iOS.

The YubiKey 5 Series supports extended APDUs, extended `Answer To Reset (ATR)`, and `Answer To Select (ATS)`. Using the PIV APDUs on iOS requires the Yubico iOS SDK.

6.6.1 Default Values

- PIN: 123456
- PUK: 12345678
- Management Key (3DES): `010203040506070801020304050607080102030405060708`

6.6.2 Supported Algorithms

The YubiKey 5 Series supports the following algorithms on the PIV smart card application.

- RSA 1024
- RSA 2048
- ECC P-256
- ECC P-384

6.6.3 Policies

PIN Policy

To specify how often the PIN needs to be entered for access to the credential in a given slot, set a PIN policy for that slot. This policy must be set upon key generation or import; it cannot be changed later.

Touch Policy

In addition to requiring the PIN, the YubiKey can require a physical touch on the metal contact. Similar to the PIN policy, the touch policy must be set upon key generation or import.

6.6.4 Slot Information

The keys and certificates for the smart card application are stored in slots, which are described below. The PIN policies described below are the defaults, before they are overridden with a custom PIN policy. **These slots are separate from the programmable slots in the OTP application.**

Slot 9a: PIV Authentication

This certificate and its associated private key is used to authenticate the card and the cardholder. This slot is used for system login, etc. To perform any private key operations, the end user PIN is required. Once the correct PIN has been provided, multiple private key operations may be performed without additional cardholder consent.

Slot 9c: Digital Signature

This certificate and its associated private key is used for digital signatures for the purpose of document, email, file, and executable signing. To perform any private key operations, the end user PIN is required. The PIN must be submitted immediately before each sign operation to ensure cardholder participation for every digital signature generated.

Slot 9d: Key Management

This certificate and its associated private key is used for encryption to assure confidentiality. This slot is used for encrypting emails or files. The end user PIN is required to perform any private key operations. Once the correct PIN has been provided, multiple private key operations may be performed without additional cardholder consent.

Slot 9e: Card Authentication

This certificate and its associated private key is used to support additional physical access applications, such as providing physical access to buildings via PIV-enabled door locks. The end user PIN is NOT required to perform private key operations for this slot.

Slots 82-95: Retired Key Management

These slots are meant for previously used Key Management keys to be able to decrypt earlier encrypted documents or emails.

Slot f9: Attestation

This slot is only used for attestation of other keys generated on device with instruction f9. This slot is not cleared on reset, but can be overwritten.

6.6.5 Attestation

Attestation enables you to verify that a key on the smart card application was generated on the YubiKey and was not imported. An X.509 certificate for the key to be attested is created if the key has been generated on the YubiKey. Included in the certificate are the following extensions that provide information about the YubiKey.

- `1.3.6.1.4.1.41482.3.3`: Firmware version, encoded as three bytes. For example, 050100 indicates firmware version 5.1.0.
- `1.3.6.1.4.1.41482.3.7`: Serial number of the YubiKey, encoded as an integer.
- `1.3.6.1.4.1.41482.3.8`: Two bytes, the first encoding the PIN policy and the second encoding the touch policy.
- PIN policy:
  - 01 - never require PIN
  - 02 - require PIN once per session
  - 03 - always require PIN.
- Touch policy:
  - 01 - never require touch
  - 02 - always require touch
  - 03 - cache touch for 15 seconds.
- `1.3.6.1.4.1.41482.3.9`: YubiKey's form factor, encoded as a one-byte octet-string.
- USB-A Keychain: 0x01
- USB-A Nano: 0x02

- USB-C Keychain: 0x03
- USB-C Nano: 0x04
- USB-C and Lightning®: 0x05
- Undefined: 0x00

6.6.6 Changes

Answer to Reset (ATR) and Answer to Select (ATS)

The ATR has been changed from “Yubikey 4” to “YubiKey” and adds support for ATS.

PIV Attestation Root CA

YubiKeys in the 5 Series have a PIV attestation root certificate authority different from the one previous YubiKeys had. You can download the certificate of the new root certificate authority on the PIV attestation page.

6.6.7 Easier Identification

The YubiKey 5 Series devices can report their form factor via the PIV application whether or not they have an NFC interface. This enables easier, programmatic identification of the physical attributes of the YubiKey. For more information about how to query this information, see the YubiKey 5 Series Configuration Reference Guide.

CHAPTER

SEVEN

TOOLS AND TROUBLESHOOTING

7.1 Managing Applications

7.1.1 Enabling/Disabling

The YubiKey Manager can be used to find out which applications are enabled on which interface and to enable or disable each application on each physical interface.

To find out which applications are enabled, select the **Interfaces** tab. A checkbox with a tick is shown next to each enabled applications. To change which applications are enabled, use the checkboxes to select the ones you want enabled and click Save Interfaces.

Note: For the YubiKey 5Ci, any modifications made to the applications over the USB interface will also apply to the applications over Lightning®.

7.1.2 Locking

Once the desired applications have been selected, a lock code can be set to prevent changes to the set of enabled applications. This is done using the YubiKey Manager command line interface command `ykman config set-lock-code`. The lock code is 16 bytes presented as 32 hex characters. For more information, see the *YubiKey Manager (ykman) CLI & GUI Guide <https://docs.yubico.com/software/yubikey/tools/ykman/>*`_.

7.2 YubiKey Manager (ykman)

The YubiKey Manager is a tool for configuring all aspects of YubiKeys in the 5 series and for determining the model of key and the firmware it runs. It has both a graphical interface and a command line interface. Being cross-platform, it runs on Windows, macOS, and Linux. Some of the more advanced options are only available through the command line. See the YubiKey Manager (ykman) CLI Guide.

7.2.1 Graphical User Interface (GUI)

The graphical user interface of the YubiKey Manager provides an easy-to-use method of performing basic configuration tasks of the YubiKey 5 Series, including:

- Displaying information about the YubiKey(s) connected to the computer.
- Enabling or disabling applications per physical interface.
- Setting or changing the FIDO2 PIN, as well as resetting the FIDO application.
- Managing the credentials in the OTP application.

7.2.2 Command Line Interface (CLI)

Using ykman's CLI, you can do everything that the GUI can and more. This includes, but is not limited to:

- Enabling or disabling applications and prevent unauthorized changes by setting a lock code.
- Managing the credentials in the PIV / Smart Card application, including resetting them.
- Managing and generating OTPs from the credentials in the OATH application, including resetting the application.
- Resetting the OpenPGP application and setting the OpenPGP touch policy.

For usage information and examples for ykman, see the YubiKey Manager (ykman) CLI Guide.

7.3 Yubico Authenticator

Yubico Authenticator is used to manage credentials on the OATH application and display the OTPs generated by the YubiKey. Yubico Authenticator is required in order to generate OTPs for OATH-TOTP credentials as the YubiKey does not contain a battery and thus cannot track time. It is open source, cross-platform, and runs on Windows, macOS, Linux, and Android. The Android version of Yubico Authenticator can communicate with YubiKeys over NFC or USB.

7.4 YubiKey Smart Card Minidriver

The YubiKey Smart Card Minidriver extends the PIV / Smart Card application on the YubiKey on Windows, facilitating deployment and management. Key benefits include:

- Enrollment of the YubiKey using standard Windows utilities.
- Auto-enrollment, enabling user self-provisioning of a YubiKey and automatic renewal.
- Multiple authentication certificates on one YubiKey.
- Changing of the PIN from the Ctrl+Alt+Del menu.
- Unblocking of the PIN using the PUK at the Windows logon screen.

To get started with the YubiKey Smart Card Minidriver, see the deployment guide

Note: For use with YubiKeys in the 5 Series, version 4.0 or later of the minidriver is required.

7.5 Troubleshooting

If you run into any issues with a key from the YubiKey 5 Series, refer to the Knowledge Base and search for your issue. If your issue is not listed in the Knowledge Base, or if you have any technical questions, you can open a ticket with our Technical Support team: https://www.yubico.com/support/contact/.

CHAPTER EIGHT

FIDO2 AAGUIDS

The FIDO2 specification states that an Authenticator Attestation GUID (AAGUID) must be provided during attestation. An AAGUID is a 128-bit identifier indicating the type of the authenticator.

New AAGUIDs will be issued for new YubiKey products which support FIDO2, or when existing YubiKey products have FIDO2 features added or removed.

For the complete list of AAGUIDs, see https://support.yubico.com/hc/en-us/articles/360016648959-YubiKey-Hardware-FIDO2-AAGUIDs.

CHAPTER

NINE

YUBICO SECURE CHANNEL TECHNICAL DESCRIPTION

9.1 Introduction to Yubico Secure Channel

Yubico has implemented a subset of the (GlobalPlatform Secure Channel Protocol 03) Secure channel specification: specifically, only the most secure implementation including command and response message authentication code (MAC) and encryption.

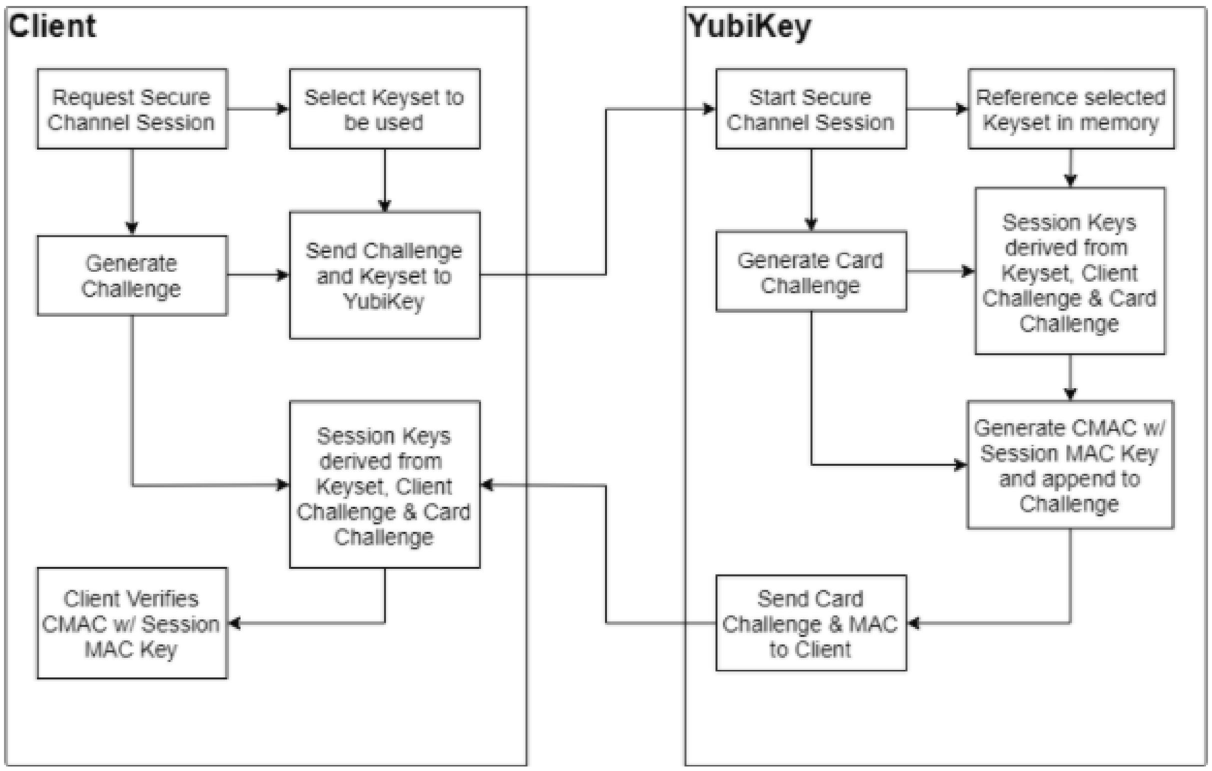
At the highest level, implementing a secure channel consists of providing overhearing and tampering resistance to information being sent between an external service, like a Credential Management Solution (CMS) and a smart card. Overhearing resistance is accomplished by the AES encryption of all commands being sent and received by use of a unique, private symmetric AES key. Tamper resistance is included by sending a securely encrypted MAC of both the commands and associated responses using an AES key unique to each session. Since these protections are applied to the data at the endpoints of the communication channel, a standard CCID interface can be used without modification, supporting native flows in Windows, Linux and other systems.

On the YubiKey, all of the secure channel operations occur within the secure cryptographic processor, with the plain text of the communication never exposed to outside observers.

Flow when Initializing a Secure Channel on a YubiKey

9.2 YubiKey Secure Channel Support

The YubiKey Secure Channel implementation is separate from the rest of the functionality on the YubiKey. It is only active when a secure channel is established and sits between the input of APDU commands sent into the secure element and the applications on the YubiKey. As such, any command which can be sent as an APDU over CCID can use secure channel, regardless if it is for PIV, OTP or other supported functions. The only exceptions are the FIDO protocols (U2F/WebAuthn), as they do not support communication over the CCID channel.


Client
Request Secure Channel Session
Select Keyset to be used
Generate Challenge
Send Challenge and Keyset to YubiKey
Session Keys derived from Keyset, Client Challenge & Card Challenge
Client Verifies CMAC w/ Session MAC Key
YubiKey
Start Secure Channel Session
Reference selected Keyset in memory
Generate Card Challenge
Session Keys derived from Keyset, Client Challenge & Card Challenge
Generate CMAC w/ Session MAC Key and append to Challenge
Send Card Challenge & MAC to Client

9.3 Transport Keys and Session Keys

Key	Usage	Creation
Static Secure Channel Encryption Key (Key-ENC)	Generate session key for Decryption/ Encryption (AES)	Imported from Trusted source
Static Secure Channel Message Authentication Code Key (Key-MAC)	Generate session key for Secure Channel authentication and Secure Channel MAC Verification and Generation (AES)	Imported from Trusted source
Data Encryption Key (Key-DEK)	Sensitive Data Decryption (AES) used for encryption other Transport key sets on import	Imported from Trusted source
Session Secure Channel Encryption Key (S-ENC)	Used for data confidentiality	Dynamically Created Per Session
Secure Channel Message Authentication Code Key for Command (S-MAC)	Used for data and protocol integrity	Dynamically Created Per Session
Secure Channel Message Authentication Code Key for Response (S-RMAC)	User for data and protocol integrity	Dynamically Created Per Session

The Yubico Secure Channel uses two types of AES keys as defined in the SCP03 specifications; these are organized in the static, externally sourced and imported transport keys, and the dynamic, internally generated session keys. The YubiKey can hold up to 3 transport key sets, and generates unique session keys for each session.

9.4 Transport Keys

A Transport Key set is made of 3 AES keys:

- Secure Channel Encryption Key **(KEY-ENC)**
- Secure Channel Message Authentication Code Key **(Key-MAC)**
- Data Encryption Key **(Key-DEK)**

The transport key sets used for establishing the secure channels are protected in the SCP03 security domain in the secure element. A transport key set contains three long-lived keys, imported from an external source. When a session is established, the session keys are derived from the long-lived transport key set.

The YubiKey security domain can store three concurrent long-lived transport key sets. In order to import new transport key sets, a secure channel must be established with the security domain. This has to be done with a previously loaded transport key set or the default transport key set.

The Secure Channel Encryption Key is used when initializing a session to generate the Session Secure Channel Encryption Key to be used during that session. Likewise, the Secure Channel MAC Key is used to generate the Session Secure Channel MAC key for Command and Session Secure Channel MAC Key for Response. The Data Encryption Key is only used when importing new transport key sets; the keys to be imported must be encrypted with a known Data Encryption key.

The Transport keys are imported from a CMS or HSM over an established secure channel. YubiKeys are shipped with either default values for the transport keys, or values derived from a Batch Master Key set at programming. Transport keys can and should be rotated on a regular basis depending on the threat model for the organization. Once overwritten on a YubiKey, Transport keys cannot be restored, so they should be archived on the CMS if necessary.

9.5 Session Keys

The Session Key set is made of 3 AES keys:

- Session Secure Channel Encryption Key **(S-ENC)**
- Secure Channel Message Authentication Code Key for Command **(S-MAC)**
- Secure Channel Message Authentication Code Key for Response **(S-RMAC)**

Session keys are all dynamically generated at the start of each session, using the Secure Channel Encryption and MAC Transport Keys, as well as the challenge sent from the client at the session start. For more details, refer to the GP SCP03 spec, section 4.1.5.

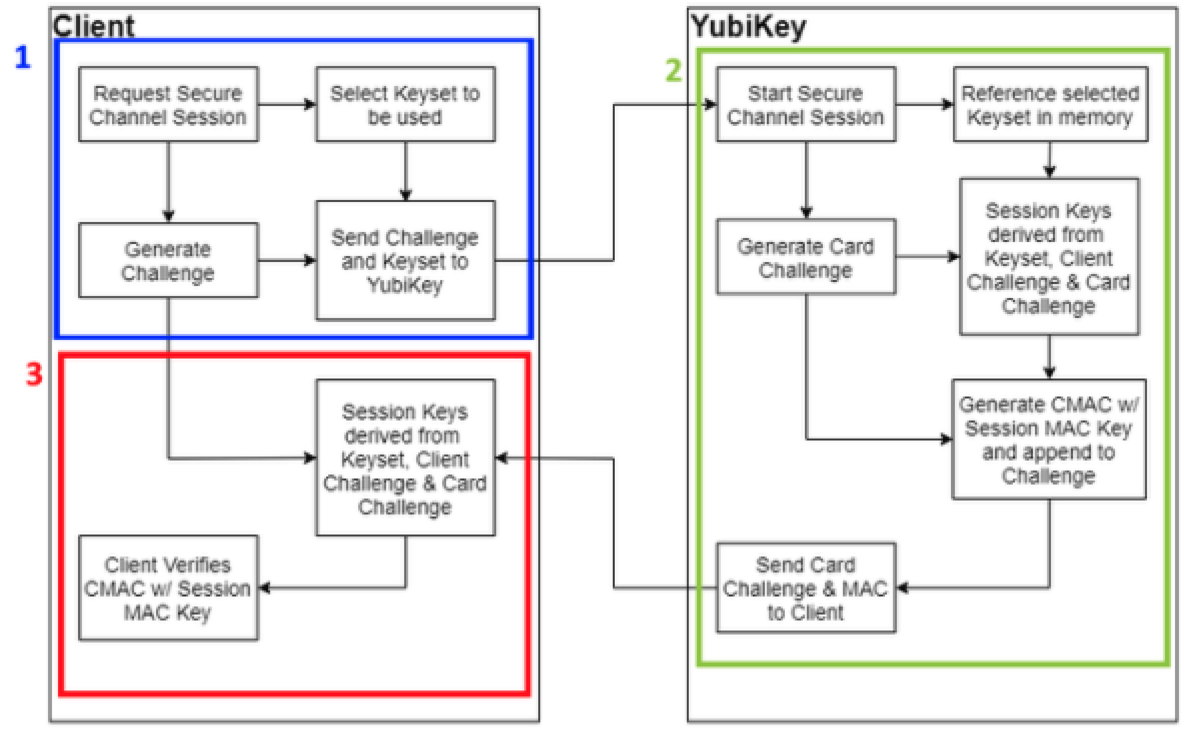
Every command sent over a secure channel between a client or CMS and a YubiKey is encrypted with the Session Secure Channel Encryption Key. Further, each command from the client has a MAC generated from the contents of the encrypted command APDU and the Session MAC key for Command. This MAC value is used to verify the authenticity of the command sent. Each command MAC value is based off the previous command MAC, enabling a chain which can be verified to ensure the data was not tampered in transit, nor is a command being replayed.

The Response MAC is generated using the encrypted response APDU from the YubiKey and the Session MAC key for Response. Each Response MAC also includes a value derived from the original command MAC sent from the client, providing a verification that the data included in the response corresponds to the last command sent.

9.6 Establishing a Secure Channel

A client connecting to any CCID function on the YubiKey, can establish a secure channel at the start of a session. Once a session has been started with a secure channel, all communication to and from the YubiKey over that session must be encrypted; sending a command in plain text will not be accepted and immediately end the session, removing any authorizations granted previously.

To begin, the client will identify the function they wish to communicate with and send the **Initialize Update** command.


Client
1
Request Secure Channel Session
Select Keyset to be used
Generate Challenge
Send Challenge and Keyset to YubiKey
3
Session Keys derived from Keyset, Client Challenge & Card Challenge
Client Verifies CMAC w/ Session MAC Key
YubiKey
2
Start Secure Channel Session
Reference selected Keyset in memory
Generate Card Challenge
Session Keys derived from Keyset, Client Challenge & Card Challenge
Generate CMAC w/ Session MAC Key and append to Challenge
Send Card Challenge & MAC to Client

YubiKey Secure Channel Initialize Update Flow

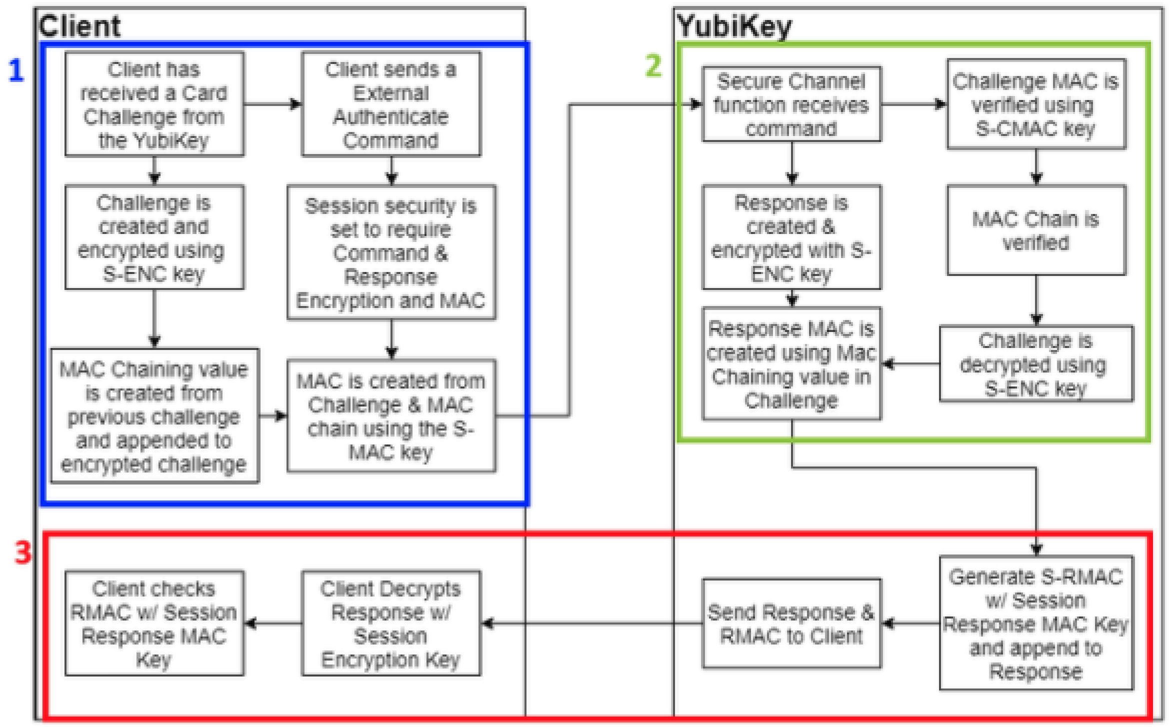
Step 1 When a client starts the process of establishing a secure channel with the YubiKey, it will select the Transport key set on the YubiKey to be used, as well as generating a unique challenge. This challenge will be used by the client to derive the session keys which will be utilized going forward. The Initialize Update command is sent from the client, which includes the challenge and Transport Key set identifier, to the YubiKey's CCID function they want to establish a secure channel with.

Step 2 The YubiKey, upon receiving an Initialize Update command when a client is starting a session with any CCID facing function, will direct communication between a client and the YubiKey to the Secure Channel function for the remainder of the session. The Secure Channel Function will use the Transport Key Identifier to select the Transport Key set in memory to use. The YubiKey will generate a card challenge and use it, along with the challenge provided from the client, to derive the Session keys for Encryption and MAC using the selected Transport Key set.

The YubiKey will then generate a command MAC from the previously internally generated card challenge using the Session Command MAC key (S-MAC). The card challenge from the YubiKey and associated MAC are sent back to the client.

Step 3 The client, upon receiving the response from the YubiKey, derives the session keys using the transport key set, the original challenge and the card challenge from the YubiKey. The client then verifies the MAC using the session keys it had generated. Upon a successful verification, the client can be confident that the YubiKey has generated matching session keys. However, at this point, the YubiKey does not know if the client has the correct key set.

To authorize the YubiKey to accept commands from the client, the **External Authenticate** command must be run next.


Client
YubiKey
1
Client has received a Card Challenge from the YubiKey
Client sends a External Authenticate Command
Challenge is created and encrypted using S-ENC key
Session security is set to require Command & Response Encryption and MAC
MAC Chaining value is created from previous challenge and appended to encrypted challenge
MAC is created from Challenge & MAC chain using the S-MAC key
2
Secure Channel function receives command
Challenge MAC is verified using S-CMAC key
Response is created & encrypted with S-ENC key
MAC Chain is verified
Response MAC is created using Mac Chaining value in Challenge
Challenge is decrypted using S-ENC key
3
Client checks RMAC w/ Session Response MAC Key
Client Decrypts Response w/ Session Encryption Key
Send Response & RMAC to Client
Generate S-RMAC w/ Session Response MAC Key and append to Response

YubiKey Secure Channel External Authenticate flow

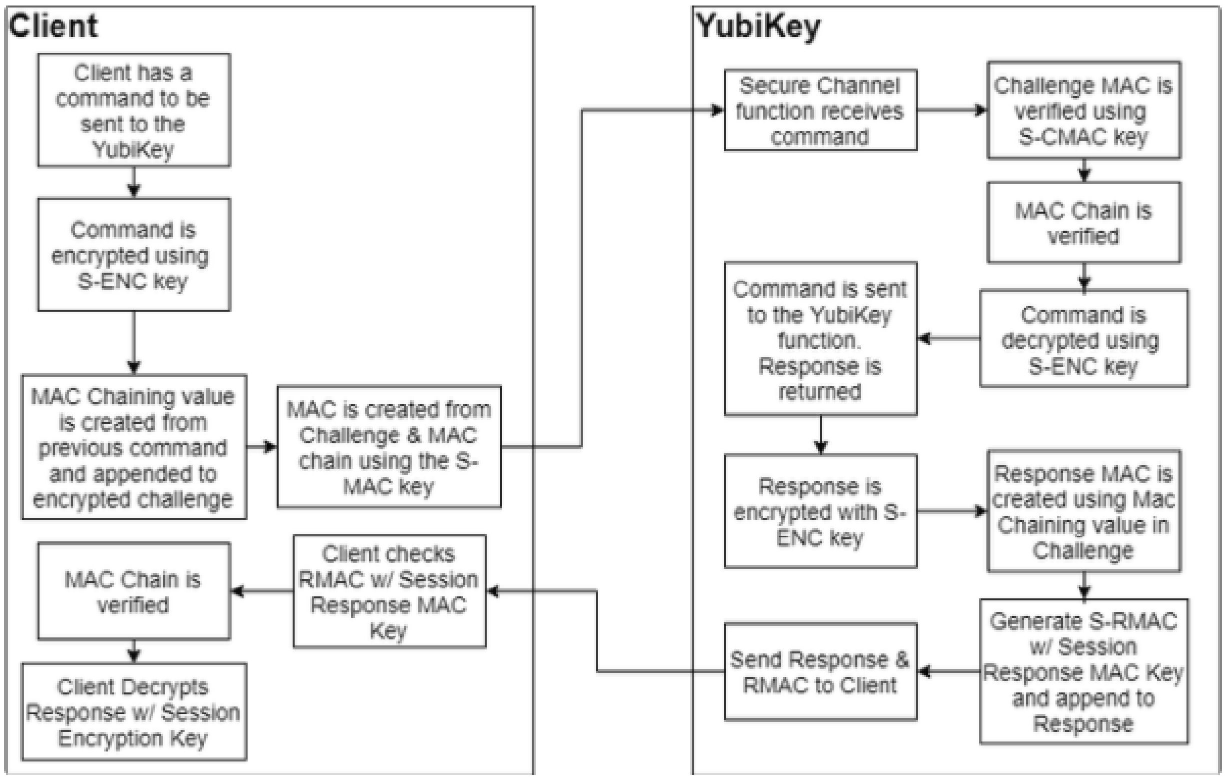
Step 1 The External Authenticate flow starts with the client receiving the card challenge from the YubiKey created during the Initialize Update command. From that point, the client will define the session security settings - the YubiKey only supports the strictest option, with both commands and responses encrypted and have associated MACs generated. As with the Initialize Update flow, the client creates a challenge and encrypts it with the session encryption key. However, a MAC value from the previous challenge is also created and appended to the challenge, creating a chain of commands to be tracked. The Challenge and MAC chain value are then used to create a command MAC using the S-MAC key, and both are sent to the YubiKey.

Step 2 The YubiKey receives the External Authenticate command, and verifies the Challenge using the MAC value and S-MAC key. The MAC Chain is then verified, confirming that the client has the same session keys and a secure channel has been created. At that point, the challenge is decrypted using the S-ENC key, and a response is created. In addition, the challenge from the client is used to create a new MAC chain value, which is appended to the response.

Step 3 The response and MAC Chain value are used to generate a response MAC using the S-RMAC key. Then the response and associated MAC are sent from the YubiKey back to the client. At this point, the response MAC is verified using the S-RMAC key, the MAC Chain value is verified against the command sent previously, and the response is decrypted.

At this point, a secure channel has been established between the client and the YubiKey.

9.7 Communicating Over Secure Channel


Client
Client has a command to be sent to the YubiKey
Command is encrypted using S-ENC key
MAC Chaining value is created from previous command and appended to encrypted challenge
MAC is created from Challenge & MAC chain using the S-MAC key
MAC Chain is verified
Client checks RMAC w/ Session Response MAC Key
Client Decrypts Response w/ Session Encryption Key
YubiKey
Secure Channel function receives command
Challenge MAC is verified using S-CMAC key
MAC Chain is verified
Command is sent to the YubiKey function. Response is returned
Command is decrypted using S-ENC key
Response is encrypted with S-ENC key
Response MAC is created using Mac Chaining value in Challenge
Send Response & RMAC to Client
Generate S-RMAC w/ Session Response MAC Key and append to Response

Communicating Over Secure Channel Flow

When Command APDU is sent over an established secure channel, the Yubico Secure Channel follows an encrypt then MAC approach.

Step 1 The command APDU is first encrypted using the Session Encryption Key (S-ENC). It is important to note that any APDU delivering instructions or data (such as a key or certificate) to a YubiKey is considered a Command APDU. Sending an unencrypted command will end the current session and will remove any authorizations.

Step 2 A command MAC is created using the Session MAC key (S-MAC) using the encrypted APDU along with a MAC chain value created from the previous Command MAC. This is sent to the YubiKey.

Step 3 The YubiKey verifies the command MAC, then verifies the MAC Chain links to the previous command sent.

Step 4 With the MAC values verified, the command is decrypted and passed to the YubiKey functionality. A response from the function being communicated with is returned.

Step 5 The response is encrypted with the S-ENC key, then has a MAC chain value derived from the command appended to it. The response and chain MAC value are used with the Session Response MAC key to generate a response MAC. The Response and Response MAC are sent back to the client.

Step 6 The client performs the same operations, verifying the response MAC, verifying the MAC chain, then decrypting the response. The Response APDU are passed to the client.

CHAPTER

TEN

SECURE CHANNEL KEY DIVERSIFICATION AND PROGRAMMING

10.1 Introduction

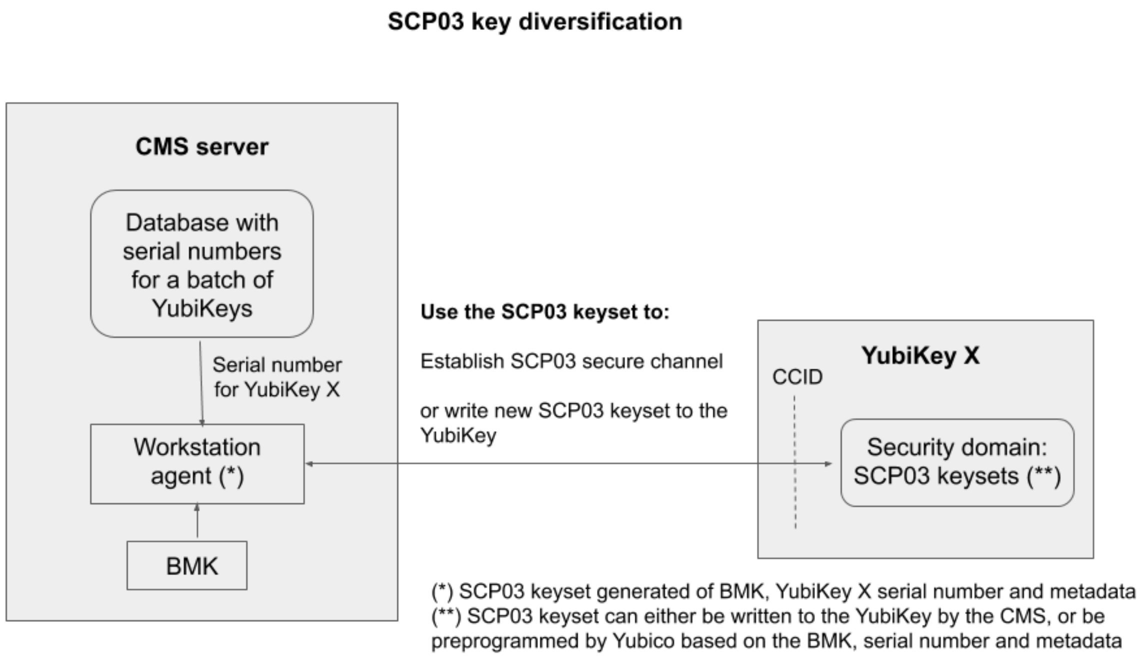
The term “key diversification” refers to the process of deriving a secure channel static transport key set from a Batch Master Key (BMK), the YubiKey identifier (part of serial number), and additional metadata.

10.1.1 Benefits and Usage

Key diversification enables simplified and secured distribution of secure channel transport key sets as only the Batch Master Key must be shared with the CMS system to derive the YubiKey transport key sets.

Hence, the secure channel transport key sets can be pre-programmed by Yubico, assuming that Yubico has access to the BMK of the CMS vendor.

Another option is for the CMS system to generate the secure channel transport key sets based on the YubiKey serial number, the BMK, and additional metadata. The CMS can then use the initial secure channel transport key set for writing additional secure channel transport key sets to the YubiKeys.


SCP03 key diversification
CMS server
Database with serial numbers for a batch of YubiKeys
Serial number for YubiKey X
Workstation agent (*)
BMK
Use the SCP03 keyset to:
Establish SCP03 secure channel
or write new SCP03 keyset to the YubiKey
YubiKey X
CCID
Security domain: SCP03 keysets (**)
(*) SCP03 keyset generated of BMK, YubiKey X serial number and metadata
(**) SCP03 keyset can either be written to the YubiKey by the CMS, or be preprogrammed by Yubico based on the BMK, serial number and metadata

SCP03 Key Diversification

10.2 Secure Channel and Security Domains

The YubiKey supports up to 3 secure channel transport key sets. This is to enable more granular control over the establishment of a secure channel to a specific device. The keys in each of these key sets can be overwritten once connected to the YubiKey, allowing for a YubiKeys to be shipped with a default key set, then have the key set be changed to a random set of keys at initialization, ensuring only the CMS server has the actual keys.

10.3 Key Diversification Option

When purchasing YubiKeys from Yubico, there is an option to custom-configure the transport keys from the default to values derived from details specific to the hardware and a BMK. This means these keys can be distributed with locked down key sets, ensuring they cannot be connected to remotely by third parties. The BMK is generated and owned by the customer, who in turn can provide it to Yubico and their CMS deployment. The CMS can then use the BMK to establish a secure channel to a customer's YubiKeys, and set new transport keys during initialization, limiting access to just that CMS. As with other custom configuration options, these keys can be overwritten or deleted by the customer; they are not "baked into" the YubiKey firmware.

10.3.1 Batch Master Key (BMK) Generation

Each custom order with diversified keys has a unique BMK used when Yubico programs them. A BMK is a 32-bit AES key - for more details on this function, see: https://www.yubico.com/wp-content/uploads/2015/04/YubiHSM-Manual_1_5_0.pdf, section 4.8.

Before every order with Key Diversification the customer will generate and provide the BMK to Yubico by secure means. After the YubiKeys have been programmed using the BMK provided, the BMK on the Yubico programming station is destroyed, leaving the customer with the only extant BMK. The customer must maintain and securely archive their BMK if required for future orders.

10.3.2 Key Diversification Function (KDF)

The diversification function used is the AES-CMAC-KDF Counter Mode derivation algorithm specified in NIST SP800-108.

Scroll horizontally to see the diversified key on the next line:

```
AES CMAC of [ Counter (1 byte) || Label (4 bytes) || 00 || Context (10 bytes) || Key␣
↪Length in bits (2 Bytes) ]
```

Note: AES256 and 3DES keys require two rounds of KDF to generate a 32-byte key value and 24-byte key value respectively, the first one with a counter value set to `01` and the second one with a counter value set to `02`.

The Key Length in the KDF input filed is expressed in hexadecimal value. It is:

- `0100` for a 256-bit key,
- `00C0` for a 192-bit Key (e.g. PIV Admin Key),
- `0080` for a 128-bit key (e.g. ISD Keys & Interfaces Management Key) and
- `0040` for a 64-bit code (e.g. the PUK).

10.3.3 Labels for Key Diversification

The KDF function of separating keys uses the following labels as input. Note that these are example values, keeping Yubikey 5 Series implementation with ISD Keys as 16-byte values, YubiKey Interfaces Management Key as a 16-byte value, , the PIV Admin Key as a 24-byte value and the PUK as an 8-byte value.

Factory Key Codes	Key Length in bits	KDF Label
Issuer Security Domain DAK (Authentication Key)	'0080'	'00000001'
Issuer Security Domain DMK (MAC Key)	'0080'	'00000002'
Issuer Security Domain DEK (Encryption Key)	'0080'	'00000003'
PIV Application Administrative Key	'00C0'	'00000004'
PIV Application PUK	'0040'	'00000007'
Capabilities Lock Code (YubiKey Interfaces Management Key)	'0080'	'00000010'

In general the Key Length should be derived from the table below.

Key Size	Key Length in Bits
32 Bytes	'0100'
24 Bytes	'00C0'
20 Bytes	'00A0'
16 Bytes	'0080'
8 Bytes	'0040'

10.3.4 Context for Key Diversification

The value of the **Context** field in the KDF input data is the **Issuer Context** and is equal to the first 10 bytes of the value returned from the Global Platform `INITIALIZE UPDATE` command.

10.3.5 PUK Generation from Diversified Value

We use the trailing 8 bytes of the Diversified Value and generate the PUK using the pseudocode below. In the HTML version of this guide, you may need to scroll horizontally to see the full line.

```
for (int i = 0; i < 8; i++) { diversifiedVal[i] = (byte) (0x30 + ((diversifiedVal[i] &␣
↪0x7F) % 10));}
```

10.4 Global Platform: CPLC Data

10.4.1 Description

Although this format is officially deprecated and not part of the GlobalPlatform specification, some organizations need support for the information stored in the so-called CPLC (Card Production Life Cycle).

This consists of a static set of bytes can can be retrieved with a GET DATA command (INS 0xca) and TAG 0x9f7f after selecting the SD application.

The response is 42 bytes that can be parsed into different fields with different meanings. However, Yubico elected not to attribute any specific meaning to 40 of those bytes. Only the first two bytes are meaningful.

Example Command

To retrieve the value (scroll horizontally if necessary):

```
opensc-tool -c default -s '00a4040008a000000151000000' -s '00ca9f7f'
```

Relevant Output

```
40 90 73 F9 53 94 C0 01 23 D8 E9 F0 68 3A 48 9A @.s.S...#...h:H.
76 30 4C D8 F6 CC 41 66 61 0F C4 F5 8C DE D6 93 v0L...Afa.......
77 32 09 82 1B EA 0C 78 3D 8B                   w2.....x=.
```

Of those 42 bytes, only the first two (*40 90*) are meant to signify an Infineon SLE 78 chipset, the rest are random bytes generated when the SD application is (re)initialized.

CHAPTER

ELEVEN

YUBIHSM AUTH

11.1 Introduction

YubiHSM Auth is a YubiKey CCID application that stores the long-lived credentials used to establish secure sessions to a YubiHSM 2. The secure session protocol is based on Secure Channel Protocol 3 (SCP03). YubiHSM Auth is supported by YubiKey v5.4.0 and higher.

YubiHSM Auth uses hardware to protect the long-lived credentials for accessing a YubiHSM 2. This increases the security of the authentication credentials, as compared to the authentication solution for the YubiHSM 2 based on software credentials derived from the Password-Based Key Derivation Function 2 (PBKDF2) algorithm with a password as input.

11.2 Credentials and PIN Codes

Each YubiHSM Auth credential is comprised of two AES-128 keys which are used to derive the three session-specific AES-128 keys. The YubiHSM Auth application can store up to 32 YubiHSM Auth credentials in the YubiKey.

Each YubiHSM Auth credential is protected by a 16-byte user access code provided to the YubiKey for each YubiHSM Auth operation. The access code is used to access the YubiHSM Auth Credential to derive the session-specific AES-128 keys.

Storing or deleting YubiHSM Auth credentials requires a separate 16-byte admin access code.

Each access code has a limit of eight retries and optionally, verification of user presence (touch).

11.3 YubiHSM 2 Secure Channel

Use the YubiKey YubiHSM Auth application to establish an encrypted and authenticated session to a YubiHSM 2. Although the YubiHSM 2 secure channel is based on the protocol Global Platform Secure Channel Protocol ‘03’ (SCP03), there are two important differences:

- The YubiHSM 2 secure channel protocol does not use APDUs, so the commands and possible options are not those of the complete SCP03 specification.
- SCP03 uses key sets with three long-lived AES keys, while the YubiHSM 2 secure channel uses key sets with two long-lived AES keys.

The YubiHSM 2 authentication protocol uses a set of static credentials called a long-lived key set. This consists of two AES-128 keys:

- ENC: Used for deriving keys for command and response encryption, as specified in SCP03.

- MAC: Used for deriving keys for command and response authentication, as specified in SCP03.

The identical long-lived keyset is protected in the YubiHSM 2 and in the YubiKey YubiHSM Auth application.

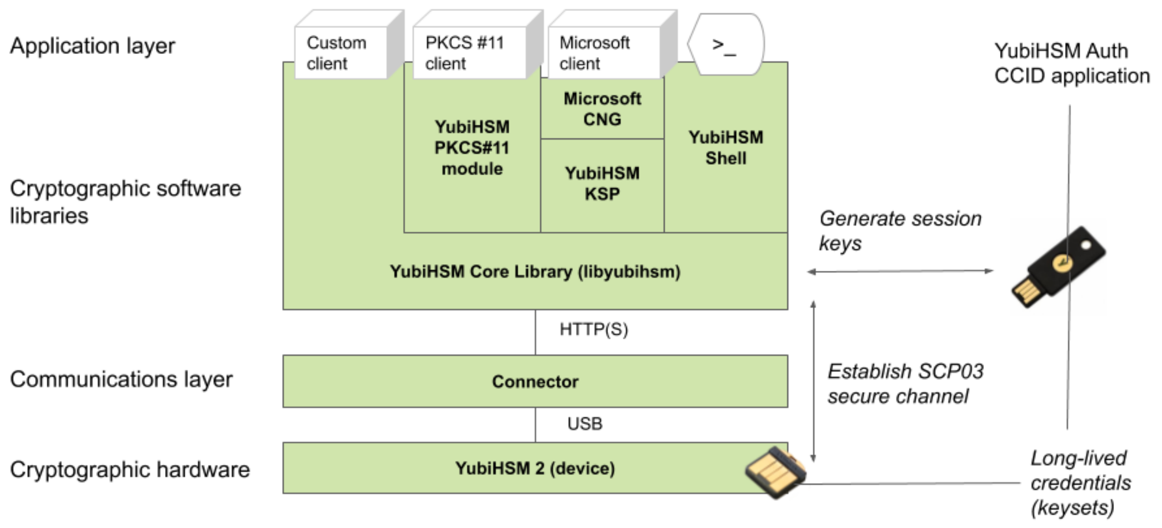
Those long-lived key sets are used by the YubiHSM Auth application to derive a set of three session-specific AES-128 keys using the challenge-response protocol as defined in SCP03:

- Session Secure Channel Encryption Key (S-ENC): Used for data confidentiality.
- Secure Channel Message Authentication Code Key for Command (S-MAC): Used for data and protocol integrity.
- Secure Channel Message Authentication Code Key for Response (S-RMAC): Used for data and protocol integrity.

The YubiHSM Auth session-specific keys are output from the YubiKey to the calling library, which uses the session keys to encrypt and authenticate commands and responses during a single session. The session keys are discarded afterwards.

11.4 Architecture Overview

The figure below shows how the YubiHSM Auth application fits in to the YubiHSM 2 architecture.



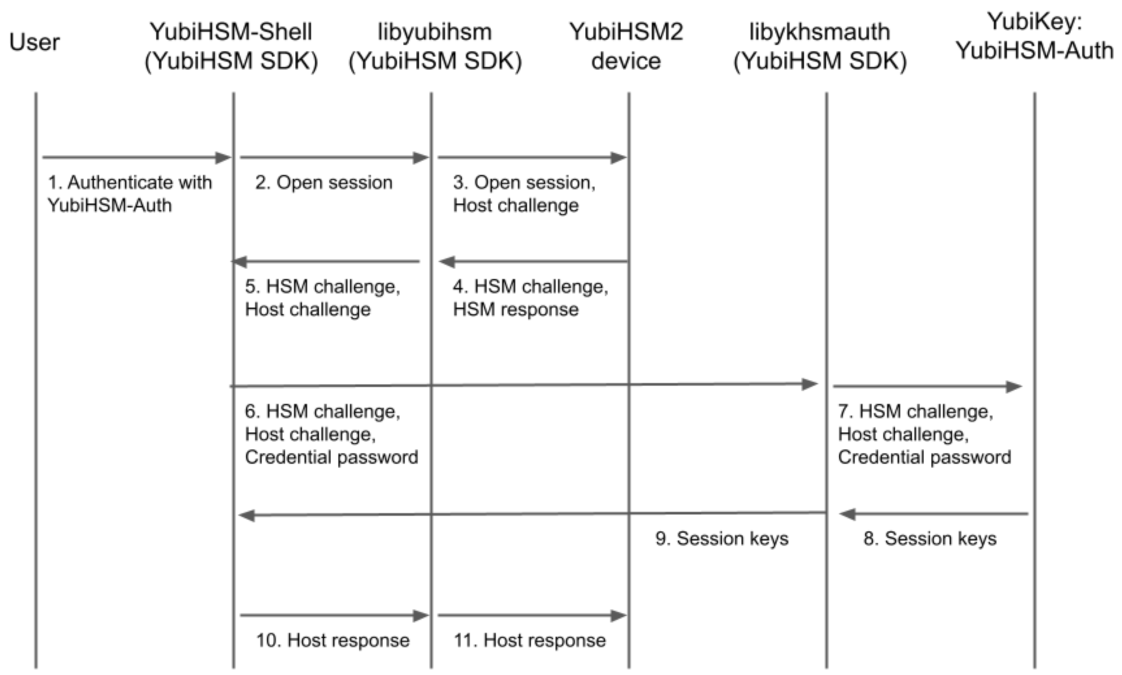
The identical long-lived credentials (key sets) are protected in both the YubiKey YubiHSM Auth application and in the YubiHSM 2. The YubiHSM-Shell software tool can be used for generating the key sets in the YubiHSM 2, and the YubiHSM-Auth software tool can be used for importing the same key sets to the YubiKey YubiHSM Auth application.

At the client, the YubiHSM authentication protocol is implemented in the `libykhsmauth` library, which derives the three session AES-keys by calling the YubiKey YubiHSM Auth CCID application. The session objects that are created can be used by the `libyubihsm` in the communication with YubiHSM.

The YubiHSM session keys are therefore generated on the basis of the long-lived credentials that are protected in the YubiHSM 2 and YubiKey YubiHSM Auth in conjunction with the SCP03 derivation scheme.

11.5 YubiHSM Auth Flowchart

The flowchart below illustrates the authentication protocol communication with YubiHSM using the static keys on YubiHSM Auth. It is assumed that the YubiHSM and YubiHSM Auth application share the same static keyset. The steps are explained below.


User
YubiHSM-Shell (YubiHSM SDK)
libyubihsm (YubiHSM SDK)
YubiHSM2 device
libykhsmauth (YubiHSM SDK)
YubiKey: YubiHSM-Auth
1. Authenticate with YubiHSM-Auth
2. Open session
3. Open session, Host challenge
4. HSM challenge, HSM response
5. HSM challenge, Host challenge
6. HSM challenge, Host challenge, Credential password
7. HSM challenge, Host challenge, Credential password
8. Session keys
9. Session keys
10. Host response
11. Host response

1. The user launches YubiHSM-Shell and enters the commands `connect` and `session open`, with the flag `ykopen` that indicates that the YubiKey with YubiHSM Auth shall be used.
2. The YubiHSM-Shell invokes the `libyubihsm` library, with a request to open a session to the YubiHSM 2.
3. The `libyubihsm` library generates a host challenge, and opens a session to the YubiHSM 2 device.
4. The YubiHSM 2 device generates an HSM challenge, and generates the session keys based on the HSM challenge, the host challenge, and the static key set in the YubiHSM 2 device. The YubiHSM 2 returns the HSM challenge in an HSM response to the `libyubihsm` library.
5. The `libyubihsm` library propagates the host challenge and HSM challenge to the YubiHSM Shell.
6. The user enters the Credential password for unlocking the static keyset in the YubiHSM Auth application in the YubiKey. The YubiHSM Shell invokes the `libykhsmauth` library, with a request to generate session keys.
7. The `libykhsmauth` library invokes the YubiHSM Auth application in the YubiKey with the Credential password, the HSM challenge and host challenge are used as input parameters.
8. The Credential password unlocks the static keyset in the YubiHSM Auth application, and the YubiHSM Auth application generates the session keys based on the static keys, HSM challenge, and host challenge.
9. The `libykhsmauth` library returns the session keys to YubiHSM Shell.
10. The YubiHSM Shell acknowledges the protocol handshake to `libyubihsm`.
11. The `libyubihsm` sends the host response to the YubiHSM 2 device. The session keys can now be used for secure channel communication between `YubiHSM-Shell/libyubihsm` in the host and the YubiHSM device.

11.6 Software and Tools

11.6.1 YubiHSM-Auth Software Tool

The YubiHSM-Auth software tool is part of the YubiHSM Shell, which is installed with the YubiHSM SDK. YubiHSM-Auth tool can be used for:

- Storing the YubiHSM Auth credentials on a YubiKey
- Deleting the YubiHSM Auth credentials on a YubiKey
- Listing the YubiHSM Auth credentials on a YubiKey
- Changing the YubiHSM Auth management key on a YubiKey
- Checking the number of retries of the YubiHSM Auth credential password
- Checking the version of the YubiHSM Auth application
- Calculating session keys, mainly for debugging and test purposes
- Resetting the YubiHSM Auth application on a YubiKey

First, the YubiHSM 2 device needs to be configured with an authentication key. The default authentication key password on KeyID=1 is set to "password", and this should be changed or replaced with other authentication keys. For the examples in this section, however, it is assumed that the default authentication key is still present on the YubiHSM 2.

In order to generate and store the equivalent YubiHSM Auth credentials on the YubiKey, the `yubihsm-auth` command line tool can be used. To invoke YubiHSM-Auth simply run `yubihsm-auth` with the required commands and parameters.

To get a list of available commands, parameters and their syntax, run: `yubihsm-auth --help`.

An example of how to use `yubihsm-auth` for storing YubiHSM Auth credentials on a YubiKey is shown below:

```
$ yubihsm-auth -a put --label="default key" --derivation-password="password" --credpwd=
↪"MyPassword" --touch=on --mgmkey="00000000000000000000000000000000" --verbose=5
Credential successfully stored
```

Where:

- `-a put` is the action to insert a YubiHSM Auth credential on the YubiKey
- `--label` is the label of the YubiHSM Auth credential on the YubiKey
- `--derivation-password` is used as input to the PBKDF2 algorithm, which is used for generating the two AES-128 keys that constitute the YubiHSM Auth credentials to be stored on the YubiKey
- `--credpwd` is the password protecting the YubiHSM Auth credentials on the YubiKey
- `--touch` is set to 'on', which requires the user to touch the YubiKey when accessing the YubiHSM Auth credential
- `--mgmkey` is the management key that is needed for writing the YubiHSM Auth credentials on the YubiKey
- `--verbose` is used to print more information as output

Note: We recommend using an offline air-gapped computer when storing the YubiHSM Auth credentials on the YubiKey.

Now, the YubiKey YubiHSM Auth application can be used with YubiHSM Shell for authentication to the YubiHSM 2.

11.6.2 Using YubiHSM-Auth with YubiHSM Shell

It is now possible to authenticate to the YubiHSM 2 device with static credentials that are protected in the YubiKey application called YubiHSM Auth. For more information on this YubiKey feature and how to configure it, see Using YubiHSM Auth.

The YubiHSM Shell tool supports authentication with YubiHSM Auth credentials in both interactive mode and command-line mode.

In order to use yubihsm-shell with the YubiHSM Auth-enabled YubiKey in interactive mode, open a session by executing the following yubihsm-shell command: `yubihsm> session ykopen <authkey> <label> <password>` where, in the context of using YubiHSM-Shell with the YubiHSM Auth application, the following parameters are used:

- `authkey` is the identifier of the authentication key in the YubiHSM 2
- `label` is the label of the YubiHSM-Auth credentials stored in the YubiKey
- `password` is the password that protects the YubiHSM-Auth credentials stored in the YubiKey.

Below is an example of an interactive command with YubiHSM Shell:

```
yubihsm> session ykopen 1 "default key" "MyPassword"
trying to connect to reader 'Yubico YubiKey OTP+FIDO+CCID 0'
Created session 0
```

To use yubihsm-shell with YubiHSM Auth in command-line mode, add the parameter `--ykhsmauth-label` that implicitly invokes the YubiHSM Auth application at the YubiKey. Below is an example of how to use YubiHSM Shell in command-line mode:

```
$ yubihsm-shell --ykhsmauth-label "default key" -p "MyPassword" -a generate-asymmetric -
↪A rsa2048 -i 11 -c sign-pss -l Signature_Key
```

If the YubiKey is configured to require touch when accessing the YubiHSM-Auth credentials, the user needs to touch the YubiKey sensor in addition to entering the credential password.

Once the user is authenticated with YubiHSM Auth, all YubiHSM-Shell commands can be used.

CHAPTER

TWELVE

NFC ID CALCULATION TECHNICAL DESCRIPTION

12.1 Background to Door Access

The YubiKey 5 NFC can be used for physical access to doors. Essentially, the physical access system reads out the NFC ID from the YubiKey, truncates and parses the NFC ID in different ways, and checks if there is a match to a registered value in a database. If there is a match, the door is opened.

12.2 Calculation of NFC ID

For YubiKey 5.2.x and lower versions, the NFC ID was calculated as follows:

`0x88 0x27 0 0 serial_3 serial_2 serial_1 serial_0`

where `serial_0`, `serial_1`, `serial_2` and `serial_3` are the four bytes containing information about the YubiKey's serial number. In other words, `serial_x` is a byte that contains some of the digits of the serial number, however not a digit in itself.

`serial_0` is the most significant digit, ranging to `serial_3` which is the least significant digit. The least significant digit (`serial_3`) changes most frequently, while the most significant digit (`serial_0`) changes with the lowest frequency.

When a door access system reads out the NFC ID from the YubiKey, the NFC ID may be truncated and reversed in different ways before it is matched to the registered IDs in a database. In some cases, the most significant digits are parsed out and placed first, while the rest of the NFC ID is truncated. Such processing has in some cases resulted in parsed NFC ID values that consist of the most significant digits such as `serial_0` and `serial_1`, which may not be unique for a batch of YubiKeys. In other cases, only `0x27 0 0` are used, resulting in non-unique values.

12.3 NFC ID Calculation for YubiKey v5.3.0 and Above

For YubiKeys with firmware of 5.3.0 and above, the NFC ID calculation has been changed such that the NFC ID is now derived as:

`0x88 0x27 serial_3 serial_2 serial_1 serial_0 serial_2 serial_3`

Note that two of the four bytes in the serial number are repeated both at the beginning and at the end of the sequence.

(For the Security Key by Yubico, which does not have a serial number, the NFC ID is calculated as follows:

`0x08 AA BB CC` where `AA`, `BB` and `CC` are random bytes.)

This updated calculation of the NFC ID ensures unique values, regardless of the parsing direction of the NFC ID, whether from left to right or right to left.

CHAPTER

THIRTEEN

PIV ENHANCEMENTS TECHNICAL DESCRIPTION

13.1 PIV Metadata

13.1.1 Background: How PIV Attestation Works

A technical description of YubiKey PIV attestation is available at the Yubico developer website.

Attestation is performed on a public key that has been *generated on the YubiKey*. For example, consider an asymmetric key-pair that is generated on the YubiKey with the following ykman (YubiKey Manager) command:

```
ykman piv generate-key 9c -
```

This command generates an asymmetric key-pair, and stores the private key in the specified slot (9c in this example). The public key that has been generated is returned as output.

The ykman attestation command can be executed for the key-pair at the slot (9c):

```
ykman piv attest 9c C:\Test\attestation-cert-9c.cer
```

The generated certificate is generated in real time at the YubiKey. The attestation certificate and private key, which are stored in slot f9, are used for signing the generated certificate for the slot (9c). The attestation certificate is used as template when creating the generated certificate for the slot (9c). In addition to the template attestation certificate, the extensions and subject details are appended to the generated certificate.

However, the generated certificate is not the same as the X.509 certificate that may be issued by an external CA or self-signed on the YubiKey. For example, the X.509 certificate could be issued by the Microsoft ADCS and written to the YubiKey. The YubiKey Manager GUI can be used to generate a key-pair and self-sign the public key at the YubiKey.

The public key at slot 9a can be attested (signed in real time by the CA attestation certificate) with the same ykman command as above:

```
ykman piv attest 9a C:\Test\attestation-cert-9a.cer
```

And the X.509 self-signed certificate can be exported from the YubiKey with the following ykman command:

```
ykman piv export-certificate 9a C:\Test\self-signed-9a.cer
```

13.1.2 The Shortcomings of PIV Attestation

PIV attestation only works for asymmetric keys that have been *generated on* the YubiKey; it does not work for asymmetric keys that have been *imported into* the YubiKey.

For example, the following ykman command imports a PKCS #12 file into the YubiKey at slot 9e:

```
ykman piv import-key 9e C:\\Test\\TestUser1.p12 -P 123456
```

```
ykman piv import-certificate 9e C:\\Test\\TestUser1.p12 -P 123456
```

These ykman commands unpack the PKCS #12 file, store the private key in the private key slot (9e), and store the X.509 certificate in the corresponding certificate slot.

Now, if one tries to attest the public key at slot 9e with the YkMan attestation command, the operation will fail:

```
ykman piv attest 9e C:\\Test\\attestation-9e.cer
```

```
Error:  Attestation failed.
```

One more drawback with PIV attestation is performance, since generation of multiple PIV attestation certificates can be time-consuming.

13.2 When To Use PIV Metadata

PIV metadata should be used for the following cases:

- If PIV attestation cannot be used (for imported keys),
- If an attestation certificate is not required, PIV metadata can be used for achieving higher performance.

13.3 Yubico PIV Library and Metadata API

PIV metadata is supported by YubiKey v5.3.0 and above. YubiKey PIV metadata can be accessed by using the *libykpiv* library.

The Yubico PIV Tool contains the library

- `libykpiv.so` (for Linux),
- `libykpiv.dylib` (for MacOS),
- `libykpiv.dll` (for Windows).

The source code of the `libykpiv` library is published at the Yubico GitHub repo.

13.3.1 libykpiv

The *libykpiv* library exposes a C API in the header file ykpiv.h, which includes the functions `ykpiv_get_metadata()` and `ykpiv_util_parse_metadata()`. The source code of these functions is available in the file ykpiv.c.

In particular, the function `ykpiv_get_metadata()` calls the underlying function `_ykpiv_transfer_data()`, which transfers APDUs to the YubiKey PIV applet over the CCID interface.

The function `_ykpiv_transfer_data()` takes the input parameter `templ`, which is populated with the APDUs (*CLA*, *INS*, *P1*, *P2*) that are specified at the Yubico developer website for PIV extensions under the section **GET METADATA**. The YubiKey returns the tag length values (TLVs) (*Algorithm*, *Policy*, *Origin*, etc) that are specified in the same section, and the TLV-encoded output is returned in the `ykpiv_get_metadata()` parameter data.

Table 1: TLVs Returned

Key	TLV	Description
Algorithm	0x01	Algorithm/type of the key
Policy	0x02	PIN and Touch policy of the key (keys only)
Origin	0x03	Origin of the key: imported or generated
Public key	0x04	Public key associated with the private key
Default value	0x05	Whether the PIN/key has a default value PIN and PUK and Mgmt key only)
Retries	0x06	Number of retries left (PIN and PUK only)

It is even possible to invoke the function `ykpiv_transfer_data()` directly for low-level APDU communication with the YubiKey's PIV applet.

The function `ykpiv_util_parse_metadata()` can be used for parsing the returned TLV-encoded object.

Therefore, the developer can integrate the `libykpiv` library for low level programming with YubiKey PIV metadata.

13.4 Using PIV Metadata with YKCS11

The YKCS11 library is also part of the Yubico PIV Tool. YKCS11 is a PKCS#11 module that allows external applications to communicate with the PIV application running on a YubiKey.

When the PKCS #11 function `C_OpenSession()` is called for a YubiKey PKCS #11 slot (which is a YubiKey PIV application in a PC/SC reader), then the YKCS11 library will parse out the public keys for all PIV key slots. If PIV attestation is supported, the PIV attestation certificate will be used for parsing out the public key.

If PIV attestation is not supported, i.e., if the key-pair has been imported into a YubiKey, then the YKCS11 library calls the functions `ykpiv_get_metadata()` and `ykpiv_util_parse_metadata()` to parse out the requested public key.

If both attestation and PIV metadata fail, in that order, YKCS11 will fall back to parse the public key from the X.509 certificate.

Note: The X.509 certificate's public key may not match the private key in the YubiKey PIV slot.

13.5 PIV AES Management Key

Historically, the YubiKey PIV management key is a 3DES key. With the release of the YubiKey v5.4.0 firmware, the YubiKey PIV management key can also be an AES key.

Technically speaking, this feature expands the management key type held in PIV slot 9b to include AES keys (128, 192 and 256) as defined in the PIV specification (SP800-78-4, section 5). PIV management key in AES format allows for the YubiKey to be compatible with current or future FIPS compliant CMS services.

CHAPTER

FOURTEEN

YUBICO SCP03 DEVELOPER GUIDANCE

This section describes how Secure Channel Protocol 3 works in the YubiKey for developers integrating support for it.

14.1 Introduction

SCP03 is a protocol from Global Platform for mutual authentication and encrypted transport using smart cards. The protocol allows for the following modes of encryption and authentication of data:

- C-MAC,
- C-ENC,
- R-MAC, and
- R-ENC.

The YubiKey implements this with all of them turned on; turning anything off is not an option.

Authenticating with SCP03 does not assign any specific permissions in the YubiKey; what it does is set up a mutually authenticated and encrypted channel between the YubiKey and the host. Unencrypted commands sent over the secure channel will end the session, revoking any previously issued authorizations.

For more details on SCP03, refer to the Global Platform SCP03 specifications.

14.2 Key Sets

A key set contains three long-lived keys, the encryption key (**Key-ENC**), the mac key (**Key-MAC**), and the data encryption key (**Key-DEK**). When a session is established, the session encryption key is derived from the ENC key, while the session mac keys are derived from the MAC key. Any new key sets transported over the session are encrypted with the DEK.

The YubiKey only allows putting or deleting a whole key set at a time, not manipulating the individual keys within the set.

Each key set is identified by the key version defined when the set is imported into the YubiKey. Each individual key also has an id, but that serves solely to identify the specific key within the set - ENC, MAC or DEK. The key version number is required for addressing the correct set. 255 is the factory default version and therefore that version number is reserved. When importing a key set, the version is set to a value in the range 1-254.

The YubiKey can store up to three key sets at a time. By default there is one key set installed with key version 255 having the value `404142434445464748494a4b4c4d4e4f` for all three keys, which are known as the test keys. When a new key set is installed, it replaces the default key set. The YubiKey supports only AES-128 for all three keys.

When authentication with a key set fails repeatedly (i.e., 32 times in a row) that key set is deleted. When the last key set is deleted, the security domain is automatically reset with the default key set installed. To delete the last key set on purpose and force a reset, the delete instruction is sent with `p2=1`.

14.3 Sessions

The session is established only within the scope of the currently selected applet. When a new applet is selected, the session is terminated. To manage SCP03 keys, a session needs to be established with the AID `a000000151000000` - the Issuer Security Domain.

When a large amount of data is to be transported over the session, it is encrypted and mac'ed in its entirety. If the data exceeds the capacity of a single message, it is chunked for transport.

14.4 CPLC

The security domain contains an entry called CPLC which identifies a specific device. On a YubiKey, this entry is filled with random data on first boot. No significance is to be ascribed to any of the fields.

14.5 Software

Yubico has conformed to the Global Platform Open Standard, and as such, has developed the SCP03 support on the YubiKey to be compatible with open source offerings.

14.6 GlobalPlatformPro

GlobalPlatformPro is a Java library and tool for interacting with smartcards supporting the GlobalPlatform secure channel protocols.

14.6.1 Examples

Some of the following examples are long lines of code; for those, you might have to scroll horizontally.

Open a channel with the security domain and print information

```
$ java -jar tool/target/gp.jar --mode mac --mode enc --mode rmac --mode renc --debug --
↪info
```

Open a channel with the security domain and install a new key set

```
$ java -jar tool/target/gp.jar --mode mac --mode enc --mode rmac --mode renc --debug --
↪lock 000102030405060708090a0b0c0d0e0f
```

Open a channel with the PIV applet and verify the PIN over the channel

```
$ java -jar tool/target/gp.jar --mode mac --mode enc --mode rmac --mode renc --debug --
↪sdaid a000000308000010000100 -s 00200080083132333435 36ffff
```

14.7 gpshell

Gpshell is a C library and tool for interacting with the secure channel protocols.

14.7.1 Examples

Gpshell works with scripts; here is an example of opening a channel with the YubiKey. To see the last line of code in its entirety, scroll horizontally.

```
enable_trace
mode_211
establish_context
card_connect
select -AID a000000151
open_sc -enc_key 404142434445464748494a4b4c4d4e4f -mac_key␣
↪404142434445464748494a4b4c4d4e4f -kek_key 404142434445464748494a4b4c4d4e4f -security␣
↪51 -scp 3 -scpimpl 96
```

14.8 References

- https://globalplatform.org/specs-library/card-specification-v2-3-1/
- https://globalplatform.org/specs-library/secure-channel-protocol-03-amendment-d-v1-2/
- https://github.com/martinpaljak/GlobalPlatformPro/
- https://sourceforge.net/projects/globalplatform/
- https://sourceforge.net/p/globalplatform/wiki/GPShell/

CHAPTER

FIFTEEN

ACRONYMS

3DES Triple Data Encryption Algorithm

AES Advanced Encryption Standard

CCC Card Capability Container

CCID Chip card interface device, a USB protocol for a smartcard.

CHUID Card Holder Unique ID

CMS Credential Management System

CN Common name

CSR Certificate Signing Request

ECC Elliptic curve cryptography

FIDO Fast Identity Online

FIPS Federal Information Processing Standards (US government) covering codes and encryption standards.

HMAC Hash-based message authentication code

HOTP HMAC-based One-Time Password algorithm

KDF Key Derivation Function

OATH The Initiative for Open Authentication is an organization that specifies two open authentication standards, TOTP and HOTP.

OTP One-Time Password

PBKDF2 Password-Based Key Derivation Function 2

PKCS #11 This is number eleven of the Public Key Cryptography Standards; it is also the API for creating and manipulating cryptographic tokens.

PUK PIN Unlock Key

stdin standard input - usually keyboard or CLI instructions

stdout standard output - usually print to screen

TOTP Time-based One-Time Password algorithm

X.509 The standard defining the format of a public key certificate

CHAPTER SIXTEEN

COPYRIGHT

16.1 Trademarks

16.1.1 Disclaimer

16.1.2 Contact Information

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

https://www.yubico.com/support/contact/

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

16.1.3 Document Updated

2021-09-14 01:00:21 UTC