Overview of YubiKey

June 19, 2017

1 Introduction

YubiKey is a USB device that provides second factor authentication, private key storage, and decryption. It has one physical button that is a capacitive sensor. It is manufactured by Yubico. As of June 2014, the YubiKey is at the fourth generation.

The YubiKey supports two methods as a second factor authentication device:

- One-time password (OTP) which is a ASCII string that is typed into the computer by YubiKey and sent to web service; it contains a passcode encrypted under a key known to Yubico that cannot be extracted from device; there are two slots for that can be used for the key, accessed through short and long press of the button,
- Universal second factor (U2F) developed by FIDO Alliance that is a standardised challenge-response system between server, browser, and the device.

The smartcard functionality for private key and certificate storage is accessed through chip card interface device (CCID) protocol as described by two standards:

- Personal identity verification (PIV) developed by NIST; provides 24 private key and certificate slots,
- OpenPGP card that has three slots.

The other functionality included in YubiKey is:

- static password, shares two slots with OTP and OATH,
- HMAC-based one-time password (OATH-HOTP), and
- time-based one-time password (OATH-TOTP), that is derived from HOTP and needs client program to generate because Yubikey does not contain a reliable clock.

The OATH secrets can be stored and functionality provided either using two slots or through CCID interface.

There are five different models of YubiKey sold by Yubico. These are:

- YubiKey 4 that has all features as outlined above,
- YubiKey 4 Nano that has small form factor that does not extrude out of USB port,
- YubiKey 4C that has USB-C connector as opposed to USB-A,
- YubiKey NEO which is from previous generation but supports NFC (near-field communication) in addition to USB connectivity; also does not have RSA key size 4096 and ECC curve p384 key generation and decryption support,
- FIDO U2F security key that is only a U2F device.

In YubiKey 4 the components running on the device itself became closed source, while previously they were open. The client-side tools and libraries remain open source. The tools are:

- yubikey-manager (ykman) that toggles which features of YubiKey are enabled, manages the slot occupancy, resets OpenPGP functionality,
- yubico-piv-tool that has mostly management commands such as key import and generation for PIV,
- yubikey-personalization (ykpersonalize) that writes to the two slots,
- yubioath-desktop that generates OATH codes and saves keys,

- libu2f-host, python-u2flib-host, python-u2flib-server, u2fval, php-u2flib-server libraries that are C, Python, and PHP implementations of U2F protocol running on host (client's computer) and server,
- yubico-c-client (libykclient) validates OTP against Yubico server,
- yubico-pam and pam-u2f are PAM modules for OTP and U2F, respectively, allows, for example, logging into Linux account using YubiKey.

The PIV and OpenPGP functionality can be accessed using OpenSC and OpenGPG tools, respectively.

At USB level YubiKey publishes three interfaces:

- HID keyboard used for OTP transfer to computer,
- HID device, used for setup and U2F, and
- CCID chip/smartcard used for OpenGPG, PIV, and OATH.

2 One-time password

One-time password (OTP) is a 44 character string in modhex (modified hexadecimal) encoding that consists of ASCII lowercase letters that have same positions in most Latin keyboard layouts. OTP is transferred from YubiKey to computer through HID (human interface device) keyboard USB connection, which means that computer thinks the device is a regular keyboard and interprets the sent key codes under its currently enabled keyboard layout. It is a base16 encoding, which means that two characters correspond to one byte.

Example: `ccccccgdjdglvhulbflrfgvllhvhhhekleeluehieknk`

First 12 characters is the public ID of the YubiKey. The remaining 32 characters are concatenation of passcode and usage counter encrypted under secret AES key stored on the device and known to Yubico authentication servers by default. A thorough explanation of passcode as used in Yubikey 2 can be found in Oswald et al. [2013].

3 Universal second factor

Universal second factor (U2F) is a challenge-response protocol between server, browser, and device. Server generates a challenge, sends it to browser, browser adds more data about visited website to defend against phishing, sends the request to the device, device calculates the public and private key based on the website identifier, signs the challenge, and returns the signature to the service. U2F is a standard by FIDO Alliance which Yubico is a member. It needs support from web browsers and currently is implemented in Chrome browser.

The protocol also uses commands in APDU format, although through a different non-smartcard compatible (CCID) USB channel. There are three messages that the device supports: register, authenticate, and version, so the attack surface is small.

3.1 Key generation

AppID SHA256 hash of the URL (protocol, host name, port) to the service or to a canonical JSON file containing all URLs of that service.

Client data SHA256 hash of JSON containing random challenge, origin host name, and TLS Channel ID, providing traceability.

N Random nonce generated by the device during key generation.

MAC Calculated on device during key generation and authentication as $\text{MAC} = \text{HMAC}_S(\text{AppID} \| \text{HMAC}_S(\text{AppID} \| N))$.

S Secret key used for HMACs, preloaded during manufacturing onto the device that cannot be extracted.

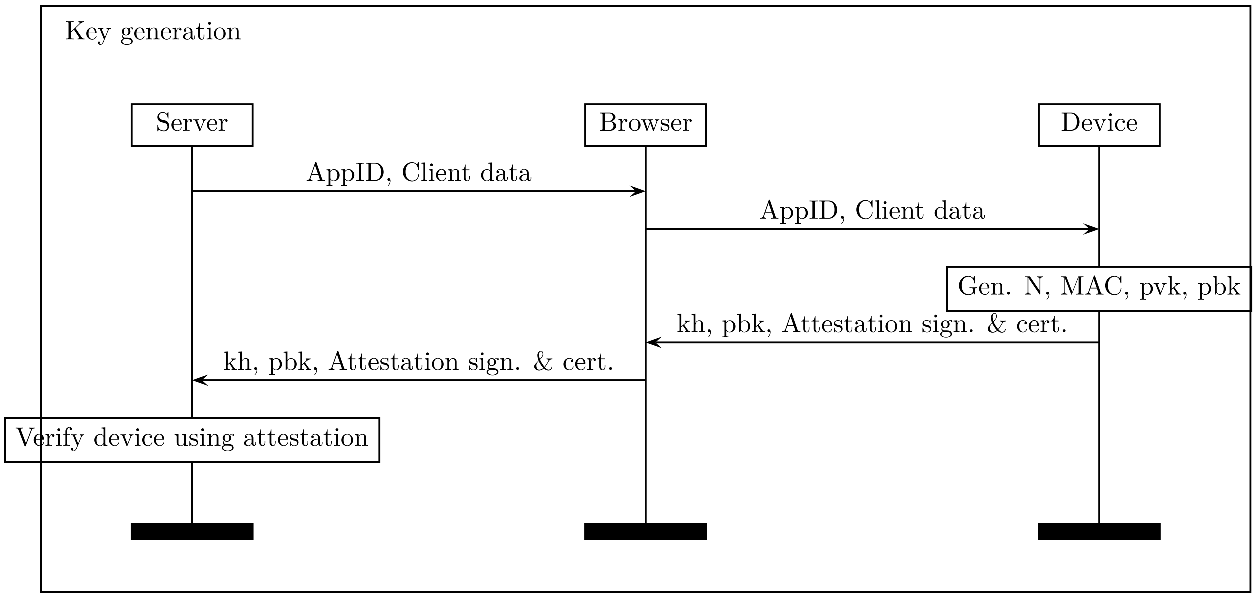
Key handle kh N and MAC together.

Private key pvk Private key is different for every service and computed on device during key generation and authentication as $\text{pvk} = \text{HMAC}_S(\text{AppID} \parallel N)$.

Public key pbk Public key is different for every service and computed on device from private key using P-256 NIST EC.

Attestation signature Computed on device using ECDSA signing public key, AppID, challenge, and key handle under attestation signing key preloaded on the device.

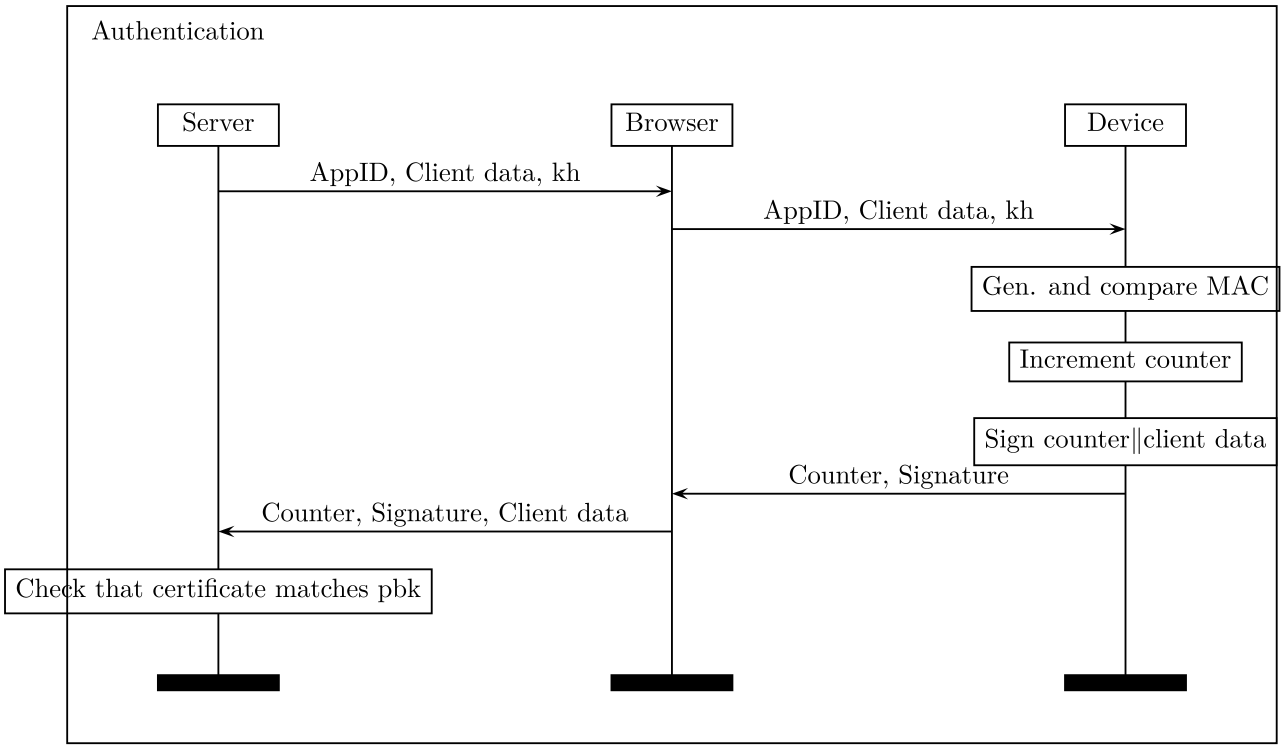
Attestation certificate Preloaded on the device, using it and signature the service can check the version and model of the device.


Key generation
Server
Browser
Device
AppID, Client data
AppID, Client data
Gen. N, MAC, pvk, pbk
kh, pbk, Attestation sign. & cert.
kh, pbk, Attestation sign. & cert.
Verify device using attestation

3.2 Authentication

Counter A global counter is stored on the device and is incremented on every authentication with any service.

Signature The device signs counter||client data under the recomputed private key for that service.



4 PIV card

Yubico states that both PGP and PIV "enables RSA or ECC sign/encrypt operations using a private key stored on a smartcard". YubiKey only supports asymmetric encryption and RSA and ECC private key storage.

The relevant standards for the smartcard mode are the following.

PKCS#11 defines high-level C API [Gleeson et al., 2016] and OpenSC is the implementation which Yubico suggests for its YubiKey PIV mode.

PKCS#15 defines data organisation inside the smartcard (dedicated file, elementary file, etc.). PIV cards are not PKCS#15 cards, and OpenSC just emulates as if PIV card was PKCS#15 card.

ISO/IEC 7816 defines standard APDU messages and structure, which can be used in many ways by vendors.

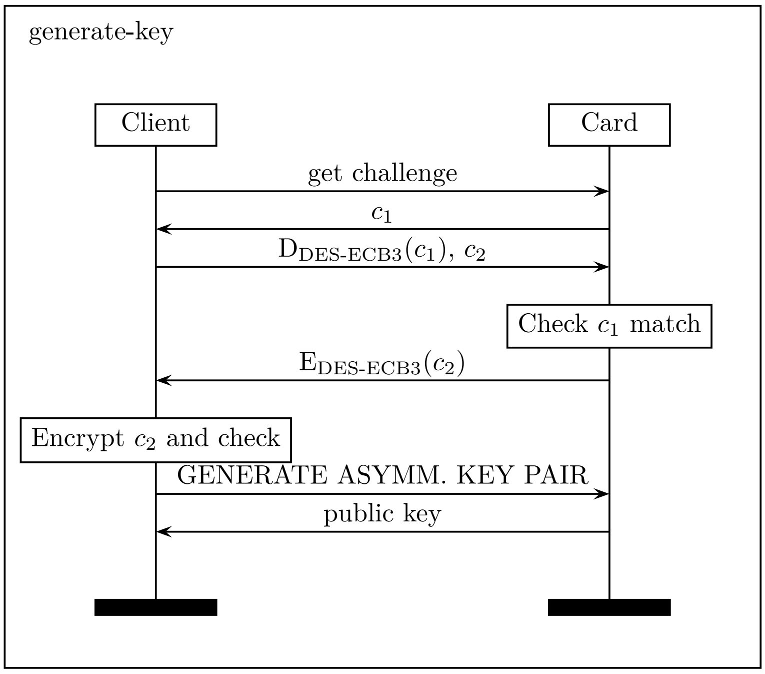
NIST SP 800-73 defines these US government PIV cards, which have 24 private key and certificate slots, specifies precise APDU messages, and their structure [Cooper et al., 2015]. YubiKey is a PIV-II card.

There are 24 private key slots that have these purposes: authentication of the card and cardholder (login), digital signature, key management (encrypting files), card authentication (door entry), attestation, and the rest are retired key management keys.

It is simple to execute an attack when user asks to generate an asymmetric key pair on the device and instead putting the attacker's private key in the specified slot and returning his/her public key to the client tool in an expected format. This works by intercepting GENERATE ASYMMETRIC KEY PAIR message and instead sending a chain of IMPORT ASYMMETRIC KEY messages to the card. Afterwards, the attacker responds to the users with a response to GENERATE ASYMMETRIC KEY PAIR.

After such attack, the attestation command ATTEST does not succeed because the key has not been generated on the device. If it is, then attestation command returns a certificate signed by a private key found in attestation key slot, which YubiKey has preloaded. There is also attestation key certificate in the certificate slot that is signed by Yubico root key so the user could check the certification chain of the generated attestation certificate.

The following is the communication trace when generating key pair using yubico-piv-tool. There is a 24 byte management key saved on the device that is used to encrypt and decrypt challenges with three key triple DES before doing management commands such as generating or importing keys. Importing a key has the same trace, except the command is IMPORT ASYMMETRIC KEY and response returns nothing.



To execute a wrap-and-decrypt attack, there is no concept of attributes (sensitive, wrap, unwrap, etc.) in the PIV standard and in the APDU messages, and no wrap function. When OpenSC tools show the attributes, they are hard-coded into OpenSC's PIV emulator, and no APDU messages are sent ot the card when you use, for example, PKCS#11 function `C_SetAttributeValue`.

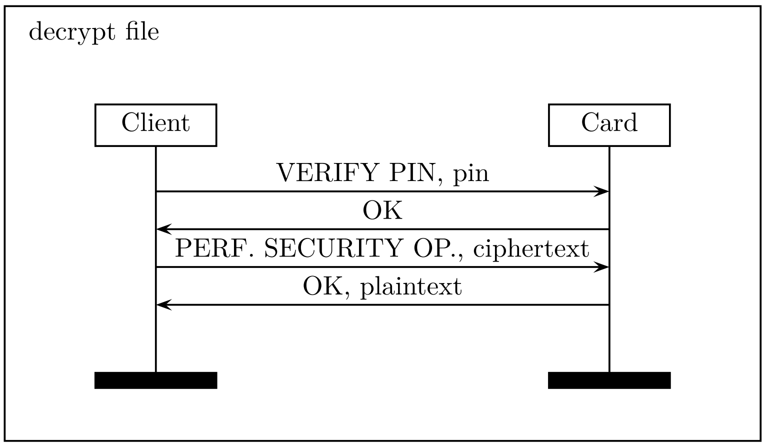
Both PIV and OpenPGP card specifications specify protocol for secure messaging but it is optional in both specifications and Yubikey does not implement it. Thus, the sensitive information, such as private keys being imported and plaintext sent to and from the device, are not encrypted and can be sniffed. Moreover, during authentication before cryptographic commands, a VERIFY command with user's PIN is sent in plaintext.

Non-management operations such as decrypting files are not provided in yubico-piv-tool. Instead, user uses generic tool from OpenSC pkcs11-tool which supports PIV. The decrypt command uses the RSA private key in first slot (9a) to decrypt a file of size up to 245 bytes. The decryption is returned in response as plaintext.

5 OpenPGP card

The APDU messages in OpenPGP mode are different from PIV mode. They are defined in the OpenPGP Card specification [Pietig, 2015]. YubiKey implements OpenPGP card version 2.1. There are no flags for wrap and unwrap in the messages and there is no such functionality in the GnuPG client program.

The following is a trace using GnuPG tool when decrypting a text file. OpenGPG uses hybrid ciphers, where cipher text is symmetric session key is encrypted under public key and plaintext encrypted under session key. Only the encryption of session key is decrypted on the card.



References

David Cooper, Hildegard Ferraiolo, Ketan Mehta, Salvatore Francomacaro, Ramaswamy Chandramouli, and Jason Mohler. Interfaces for personal identity verification—part 1: PIV card application namespace, data model and representation. *NIST Special Publication*, 800:73–4, 2015. URL http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-73-4.pdf.

Susan Gleeson, Chris Zimman, Robert Griffin, and Tim Hudson. PKCS #11 cryptographic token interface base specification version 2.40 plus errata 01. 2016. URL http://docs.oasis-open.org/pkcs11/pkcs11-base/v2.40/pkcs11-base-v2.40.html.

David Oswald, Bastian Richter, and Christof Paar. Side-channel attacks on the Yubikey 2 one-time password generator. In *International Workshop on Recent Advances in Intrusion Detection*, pages 204–222. Springer, 2013.

Achim Pietig. Functional specification of the OpenPGP application on ISO smart card operating systems. 2015. URL https://www.g10code.com/docs/openpgp-card-3.0.pdf.