yubico

The YubiKey Manual

Usage, configuration and introduction of basic concepts

Version: 3.4
Date: 27 March, 2015

Disclaimer

The contents of this document are subject to revision without notice due to continued progress in methodology, design, and manufacturing. Yubico shall have no liability for any error or damages of any kind resulting from the use of this document.

The Yubico Software referenced in this document is licensed to you under the terms and conditions accompanying the software or as otherwise agreed between you or the company that you are representing.

Trademarks

Yubico and YubiKey are trademarks of Yubico AB.

Contact Information

Yubico AB
Kungsgatan 37, 8 floor
111 56 Stockholm
Sweden
info@yubico.com

Contents

1 Document Information

1.1 Purpose

The purpose of this documentation is to provide an detailed understanding of the Yubico YubiKey concepts, operation and configuration.

The document does not cover a "systems perspective", but rather focuses on technical and usage aspects of the YubiKey device itself.

1.2 Audience

This document is intended primarily for readers with a technical/IT background. The document assumes knowledge of basic security concepts and terminology.

1.3 Related documentation

- YubiKey Configuration Utility – User's guide
- YubiKey Configuration COM API – Describes the configuration API
- YubiKey Client COM API – Describes the client-side API
- YubiKey Server COM API – Describes the server-side API
- YubiKey low-level Interface description – Describes the HID API
- RFC 2104 – HMAC: Keyed-Hashing for Message Authentication
- RFC 4226 – HOTP: An HMAC-Based One-Time Password Algorithm
- OATH Token Identifier Specification from openauthentication.org
- Type 4 Tag Operation Specification (and related documents) from nfc-forum.org
- Specification for Integrated Circuit(s) Cards Interface Devices from usb.org
- Yubico online forum – http://forum.yubico.com

1.4 Document History

Date	Version	Author	Activity
2009-09-09	2.0	JE	New release
2009-12-03	2.1	JE	Added OATH-HOTP
2010-06-15	2.2	JE	Updated for 2.2 features
2013-04-11	3.1	JE	Updated for 2.4 and NEO features
2014-09-17	3.3	JE	Updated for 3.3 firmware
2015-03-27	3.4	TM	Updated logo and other images

1.5 Definitions

Term	Definition
YubiKey device	Yubico's authentication device for connection to the USB port
USB	Universal Serial Bus
HID	Human Interface Device. A specification of typical USB devices used for human interaction, such as keyboards, mice, joysticks etc.
AES	Advanced Encryption Standard, FIPS-197
UID	Unit IDentity, a.k.a. Secret Id
Ticket	A general term for an access code generated by the YubiKey, a.k.a. OTP.
Modhex	Modified Hexadecimal coding
OTP	One Time Password
OATH	Initiative for open authentication (RFC 4226)

HOTP	Hashed One Time Password
HMAC-SHA1	Secure message hashing (RFC 2104)
EMC	Electromagnetic Compatibility
FCC	Federal Communications Commission
CE	Conformité Européenne (European Conformity)
FIDO	Fast IDentity Online
U2F	Universal Second Factor, specified by FIDO alliance

2 Introduction and basic concepts

The Yubico YubiKey is an authentication device capable of generating One Time Passwords (OTPs). The YubiKey connects to a USB port and identifies itself as a standard USB HID keyboard, which allows it to be used in most computer environments using the system's native drivers.



The YubiKey comprises an integrated touch-button that triggers the OTP generation.

Generated OTPs are sent as emulated keystrokes via the keyboard input path, thereby allowing the OTPs to be received by any text input field or command prompt.

The YubiKey operation and output is configurable, but the basic OTP generation scheme can be conceptually described as:

1. The YubiKey is inserted into the USB port. The computer detects it as an external USB HID keyboard
2. The user touches the YubiKey OTP generation button
3. Internally, a byte string is formed by concatenation of various internally stored and calculated fields, including as a non-volatile counter, a timer and a random number.
4. The byte string is encrypted with a 128-bit AES key
5. The encrypted string is converted to a series of characters that are outputted as keystrokes via the keyboard port

The generated string of keystrokes is then typically sent via an input dialog or a web form to a server or host application for verification. The basic steps for verification can be conceptually described as:

6. The received string is converted back to a byte string
7. The byte string is decrypted using the same (symmetric) 128-bit AES key
8. The string's checksum is verified. If not valid, the OTP is rejected
9. Additional fields are verified. If not valid, the OTP is rejected
10. The non-volatile counter is compared with the previously received value. If lower than or equal to the stored value, the received OTP is rejected as a replay.
11. If greater than the stored value, the received value is stored and the OTP is accepted as valid.

2.1 Basic concepts and terms

The basic function of the YubiKey is to generate One Time Passwords (OTPs), where two base modes exist – Yubico OTP and OATH-HOTP mode.

However, in order to support multiple modes of usage, several parameters can be configured to match the requirements of a particular application.

2.1.1 YubiKey standard vs. YubiKey NEO disambiguation

With the introduction of the YubiKey NEO, additional concepts beyond the capabilities of the original YubiKey have been introduced. In the following text, the original YubiKey functionality is referenced as 'YubiKey functionality', whereas the enhanced YubiKey NEO functionality is described separately. See sections 2.6 and 0 for more information.

2.1.2 The OTP string and the public identity

The full Yubico OTP string comprises an optional public id string identifying the particular device followed with the actual dynamic OTP part.

A sample output from a YubiKey may look like

```
fifjgjgkhchb irdrfdnlnghhfgrtnnlgedjlftrbdeut
fifjgjgkhchb gefdkbbditfjrlniggevfhenublfnrev
fifjgjgkhchb lechfkfhiiuunbtnvgihdfiktncvlhck
```

Here, the YubiKey button has been pressed three times in a row. As seen, the first part is static where the second changes each time. The fixed public id is used to identify the particular device when the OTP string is received so the right AES key can be retrieved to decrypt the dynamic OTP part. The public id part is optional and can be up to 128 bits in length.

The default settings for YubiKeys programmed to use the Yubico authentication server uses a 6 byte = 48 bits public id.

2.1.3 Modified Hexadecimal (Modhex) encoding

The character representation may look a bit strange at first sight but is designed to cope with various keyboard layouts causing potential ambiguities when decoded. USB keyboards send their keystrokes by the means of "scan codes" rather than the actual character representation. The translation to keystrokes is done by the computer. For the YubiKey, it is critical that the same code is generated if it is inserted in a German computer having a QWERTZ, a French with an AZERTY or an US one with a QWERTY layout. The "Modhex", or Modified Hexadecimal coding was invented by Yubico to just use the specific characters that don't create any ambiguities. The Modhex coding packs four bits of information in each keystroke. This gives that a 128-bit OTP string requires 128 / 4 = 32 characters.

A deeper description of the Modhex encoding scheme can be found in section 6.2.

2.1.4 The Yubico OTP part

The OTP part comprises 128 bits AES-128 encrypted information encoded into 32 Modhex characters. The following fields make up the OTP

- Private identity. This is a 6-byte "secret" field that is used as a part of the OTP verification. When not used as a private id, it is typically set to all zeroes.

- Counter fields. Each time a new OTP is yielded, a counter is incremented by one. The counter fields are made up of a non-volatile and a volatile part. The first is incremented by one the first time after power up, the second counter increments every time. This combination guarantees the OTPs to be truly unique.
- Timer field. In order to add entropy and to add additional means for Phishing protection, an 8 Hz timer field is inserted. Once the YubiKey is inserted, this 24-bit field is loaded with a random number and then counts up with approximately 8 Hz.
- Random number – a 16-bit random number is inserted for increased entropy.
- A closing CRC16 checksum of all fields

A more detailed description of the OTP generation algorithm can be found in section 6.1.

2.1.5 OATH-HOTP

In OATH-HOTP mode, the OTP is generated using the standard RFC 4226 HOTP algorithm. The public identity, if used, conforms to the openauthentication.org Token Identifier Specification

2.1.6 Challenge-response mode

With introduction of the Challenge-Response mode in YubiKey 2.2, there is support for programmatic interaction with the YubiKey. By using a client-side interface API, an OTP request can be sent to the YubiKey where the calculated OTP is read via an API call rather than by the means of recording keystrokes. Furthermore, additional security can be provided by a server emitted factor is inserted into the OTP generation algorithm.

The challenge-response mode is further described in section 2.5.

2.1.7 Static mode

Although it somewhat invalidates the idea with an OTP generation device, the YubiKey further supports a "static mode". As the name implies, the static mode forces the OTP part to be static.

The rationale behind the static mode is to support a medium-security scenario where compatibility with legacy systems is the key. Although static, the yielded OTP comprises a password of a length and complexity that is resistant to password guessing which is further impractical to write down or tell to someone over the telephone.

The YubiKey 2 further comprises a function to allow the user to change its static output without the need for client software. This allows seamless integration into existing password structures without any need for modification or server side software.

2.2 Functional blocks

The YubiKey comprises the following high-level functional blocks

2.2.1 USB interface

The YubiKey is designed to attach to a standard "Type A" port. The YubiKey is a "low speed" USB device (1.5 MBit/s) which conforms to the USB 2.0 specification. The YubiKey emulates an USB HID keyboard and also works in pre-boot settings.

The YubiKey is powered from the USB port and powers down according to the USB specification when the computer enters suspend mode. The YubiKey does not have an internal battery or other backup power source.

The YubiKey is not certified to work with an A-A extension cable as such cables are discouraged and not allowed by the USB specification. Although it "typically works just fine", electrical (CE/FCC) and/or USB limits may be violated.

2.2.2 OTP generation engine

The heart of the YubiKey is the microcontroller with the OTP generation algorithms implemented. The microcontroller firmware is stored in ROM and cannot be upgraded.

2.2.3 Configuration interface

The YubiKey comprises a configuration interface which allows OTP generation data and parameters to be set via the USB interface. Apart from status information, the configuration interface is "write only", i.e. once written, sensitive information cannot be read out.

2.2.4 Non-volatile memory

The YubiKey comprises a non-volatile memory that keeps settings and counter values when the device is unplugged. The memory data retention is guaranteed to be 10 years minimum.

2.2.5 Touch button

The YubiKey has an integrated touch-button used to trigger generation of an OTP. The touch button has no moving parts and operates by the means of capacitive loading introduced by a finger touching it. This means that the button cannot be pressed with an insulating device, such as a pen, a bandaged finger or a hand with a glove on.

2.2.6 Indicator light

Surrounding the touch button is a green indicator light, signaling the current state of the YubiKey. A steady green light means that the YubiKey is ready to generate an OTP where a rapidly flashing light signals some form of error condition.

2.3 Security rationale

A common question is *how secure the YubiKey is compared to method X, system Y or device Z*. Fully answering this is somewhat beyond the scope of this document as it further depends on the actual system implementation. However, given a few assumptions, the following pillars are the fundament of the YubiKey security when using Yubico OTP mode.

2.3.1 Intended usage

The YubiKey is designed as a device to be used in two-factor authentication. This means that the user should use the YubiKey together with a second factor, such as a secret password or a PIN. This prevents unauthorized usage if the device is lost or stolen.

2.3.2 Prevention of replay

The YubiKey OTP algorithm yields a 32 character dynamic string that by design is guaranteed to be unique. The OTP contains linear counters that allow the instance verifying it to determine in which particular order a set of OTPs have been generated.

The security of the YubiKey assumes that the verifying party keeps track of the last sequence number received from a particular device. If an OTP is received where the sequence number is less than or equal to the last number received, this should be rejected as a replay of an earlier generated OTP.

2.3.3 OTP lifetime

A potential issue with OTPs not including a battery-backed real-time clock is that the last OTP has an "unlimited lifetime". A scenario involving "Phishing", i.e. where a rogue user asks the legitimate user for an OTP, which is later used to access a protected service. Given a reasonably frequent usage by the legitimate user, all previously stored OTPs will by their nature be invalidated at each use. However, if this scenario is still of concern, the system shall be designed to ask for more than one OTPs during a session.

The YubiKey comprises an 8 Hz timer which starts to count when the device is powered via the USB port. This timer value is inserted in the OTP which allows the verifying party to determine the time elapsed between two subsequently received OTPs. An attacker would then have to predict the actual time elapsed for a legitimate user and convince the victim to yield OTPs in that order. This makes the attack much harder and less practical to conduct.

2.3.4 Cryptographic strength

The sent OTP is the cipher text output from an AES 128-bit encryption stage. Assuming the integrity of the AES-128 algorithm, a key space of 2^{128} bits gives about 3×10^{38} combinations. Given that there is no known cryptanalysis vector for the AES algorithm, the only possible attack involves an exhaustive search.

Just as an illustration, trying 3×10^{38} combinations would take 1000 computers working in parallel, each testing 10 billion keys each second some 10^{18} years. Even given the predictable growth in computing power, an exhaustive search is simply not practical over a foreseeable future.

2.3.5 Protection of the key and configuration data

Given the symmetric nature of the AES encryption algorithm, the security of the YubiKey relies that the AES key is logically and physically protected both in the key and in the server that verifies the OTP.

The configuration data is updated via a configuration API, accessible via the USB interface. To prevent unauthorized update, the configuration can be protected by a 48-bit access code. If used, an exhaustive search of all combinations would typically take some 100,000 years to perform. Furthermore, the YubiKey configuration data is write-only, i.e. configuration data and the key can only be written but not be read. This means that unauthorized update of the configuration is an act of sabotage rather than a security threat.

The configuration data is stored in a non-volatile storage integral to the microcontroller. A potential attack is to physically probe the silicon or analyze the hardware behavior to potentially gain full or partial knowledge of the stored secrets. However, such an attack would require a complete break-up of the YubiKey, involving dissolving the microcontroller chip encapsulation. Furthermore, very advanced equipment is needed to probe the chip internals. Given the effort and costs involved for such an attack, this is not considered a threat given that just a single device will be broken.

It has been proven that certain forms of side-channel attacks can be performed against the standard YubiKey, potentially leaking information that allows for a partial or full key recovery. Additional hardware- and software countermeasures have been added to address such concerns.

With the introduction of the YubiKey NEO, a far stronger protection against various forms of invasive and non-invasive attacks is achieved.

2.4 OATH-HOTP mode

From firmware version 2.1, the YubiKey supports the OATH-HOTP standard as outlined by RFC 4226. OTP generation is event based where the moving factor is stored in non-volatile memory of the YubiKey. The HOTP output can be truncated to 6 or 8 digits.

In OATH mode, the YubiKey further supports the OpenAuthentication.org Token Identifier Specification, where each token can be uniquely identified in an OATH ecosystem. The Token Identifier can be configured to be automatically outputted together with the HOTP.

From firmware version 2.2, the OATH-HOTP mode supports configuration of a preset moving factor value. A configuration tool can then assign a random "seed" to avoid having a predictable moving factor at the time of deployment.

The OATH mode is set per configuration which allows one YubiKey to generate both YubiKey OTPs and OATH HOTPs in the same physical device.

2.5 Challenge-response mode

For settings where automatic interaction with a client-side application is required or a server factor in the OTP generation is desired, challenge-response mode has been added from firmware release 2.2. The challenge allows cryptographic processing of a server generated challenge to create a response that can be verified by the server. The challenge-response mode is of course not limited to a strict client-server like setting and can be used for applications like a software protection "dongle", workstation lock etc.

Two basic modes of operations are available:

- **Yubico OTP mode**
  The Yubico OTP mode takes a 6 byte challenge and creates a response using the Yubico OTP algorithm, where variable fields generated by the device creates different responses even if the challenge is the same.
- **HMAC-SHA1 mode**
  The HMAC-SHA1 mode creates a HMAC on a 0-64 byte (0-512 bits) data block using a 20 byte (160 bits) fixed secret. As there is no fields generated by the device, the response is identical if a second identical challenge is issued.

Optionally, the challenge-response mode can be configured to require user interaction by the means of the user actively pressing the YubiKey button in order for the response to be sent.

Challenge-response mode cannot be used with normal one-way OTP or static modes. When configured in challenge-response mode, only API access is available.

2.6 YubiKey NEO

The YubiKey NEO features additional capabilities beyond the YubiKey standard feature set.

The additional features are:

- Fully compatible with the YubiKey standard
- Common Criteria EAL5+ secure element storing cryptographic data and performing secure operations
- Asymmetric cryptographic capabilities
- Near Field Communication (NFC) interface allowing contact-less data exchange in NFC- and RFID environments
- JavaCard execution environment
- USB CCID compliant Smartcard interface
- Full-speed USB interface

See section 0 for more information regarding the YubiKey NEO

2.7 YubiKey versions and parametric data

The YubiKey has like any product undergone a process of evolution over the years. Apart from various firmware revisions, two major versions have been released to date. The YubiKey 2 is backwards compatible with version 1, both functional and from a configuration point of view.

Firmware updates are designed to be backwards compatible. It is an explicit policy to only maintain one firmware version for each YubiKey version.

The firmware is stored in ROM and cannot be upgraded. Firmware upgrades implies replacement of existing keys.

			
Introduced	2008	2009	2012
Availability	Discontinued	Yes	Yes
Weight	1.8 g (0.06 oz)	3.3 g (0.12 oz)	3 g (0.1 oz)
Dimensions	45 x 18 x 2.3 mm (1.8 x 0.7 x 0.1")	45 x 18 x 3 mm (1.8 x 0.7 x 0.12")	45 x 18 x 3 mm (1.8 x 0.7 x 0.12")
Color	Black only	Black and White standard.	Black standard. Other colors on custom order
USB	2.0 Low-speed	2.0 Low-speed	2.0 Full-speed
Configurations	1	2	2
Static password mode	Basic from firmware revision 1.3	Enhanced	Enhanced
Static password update by user	No	Yes	No
OATH-HOTP	No	From firmware revision 2.1	Yes
Challenge-response mode	No	From firmware revision 2.2	Yes
Secure element	No	No	Yes
Javacard execution environment	No	No	Yes
USB CCID	No	No	Yes
NFC / ISO 14443A	No	No	Yes
Construction	Two piece + resin	Mono-block mold, hermetical	Mono-block mold, hermetical
Protection class (non-certified)	IP 51	IP 67	IP 67
Max bending force	5 N	25 N	25 N
EMC	CE 89/336/EEC FCC 47 CFR Part 15	CE 89/336/EEC FCC 47 CFR Part 15	CE 89/336/EEC FCC 47 CFR Part 15

2.8 YubiKey Nano

The YubiKey is also available in a very compact form-factor, allowing a flush installation into a USB type A receptacle. The functionality is fully compatible with the standard YubiKey.

		
Weight	3 g (0.1 oz)	1.8 g (0.06 oz)
Dimensions	45 x 18 x 3 mm (1.8 x 0.7 x 0.12")	13 x 12 x 2.4 mm (0.51 x 0.48 x 0.09")

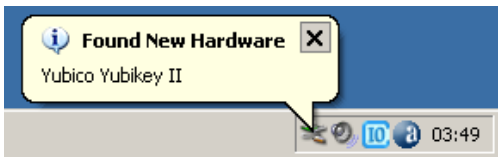
3 Installing the YubiKey

The YubiKey can be used on computer environments supporting USB HID keyboards. Although any system can be used, the following description shows it on a XP and MacOS X systems. Although there are small differences between the Windows flavors, the same concept is used from Windows 98 SE and onwards.

3.1 Inserting the YubiKey for the first time (Windows XP)

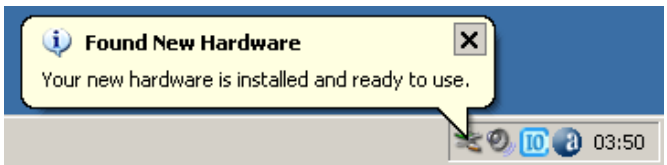
The touch button and gold contacts shall be facing up when inserting the key.

When inserted, the operating system (in this case Windows XP) recognizes the new device. The installation progress appears as a pop-up balloon in the Windows tray



The device is recognized as a Human Interface Device (HID), and the operating system installs the built-in drivers for it

When the driver installation is complete, the device is ready to use

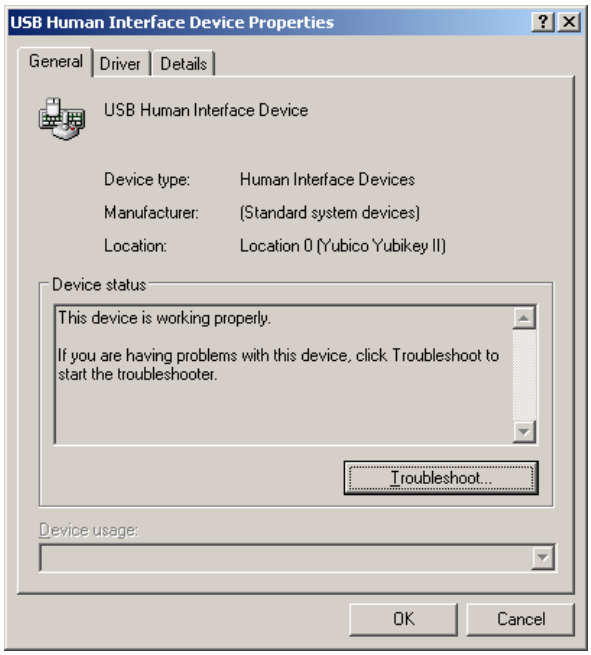


3.2 Verifying the installation (Windows XP)

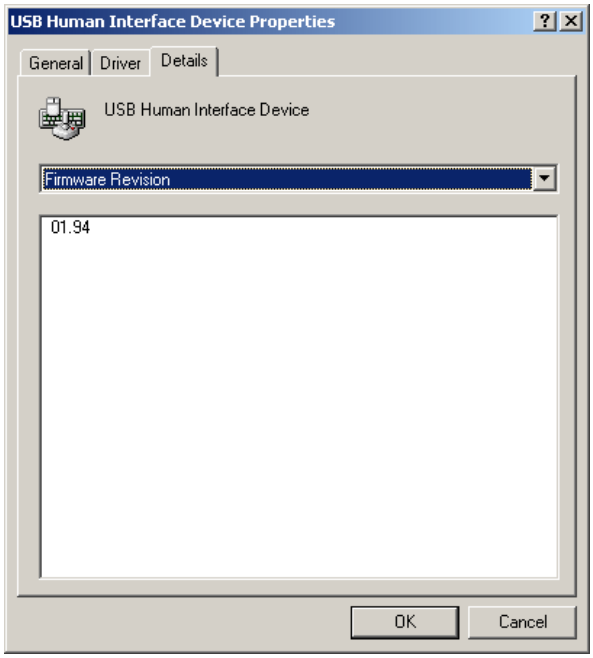
The device is ready to use and end-users only needs to be assured that the "Your new hardware is installed and ready to use" confirmation appears. If needed, the installation can be verified.

When the device is installed, it appears under the list of HID devices in the Windows device manager.

Double-clicking the selected entry brings up the properties for it



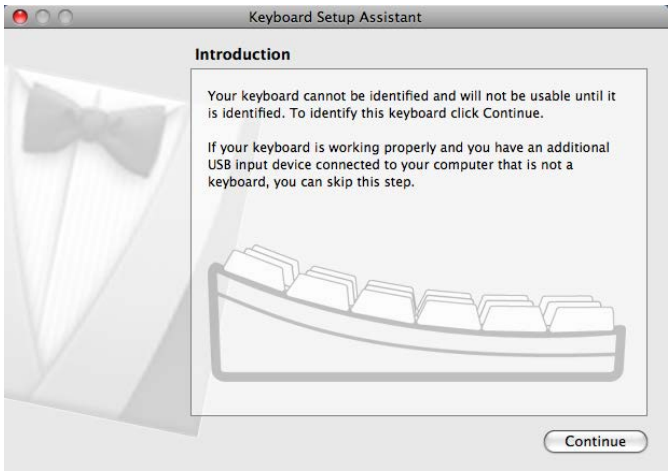
The device firmware version can be verified by selecting "Firmware version" under the "Details" tab



In this case, the firmware version is 1.9.4.

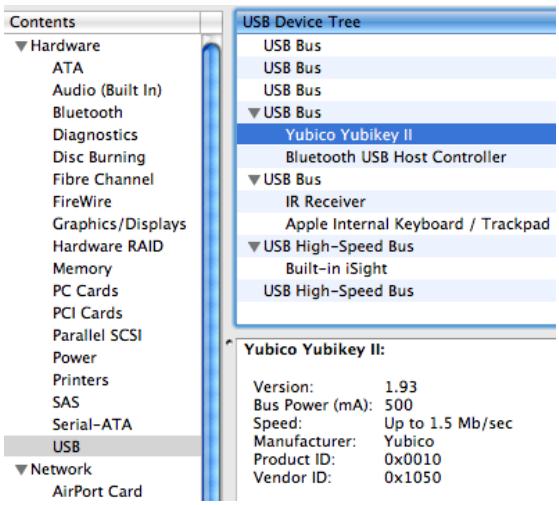
3.3 Installing the key under Mac OS X

When running Apple Mac OS X, inserting a non-Apple keyboard, like the YubiKey will bring up the following dialog



Simply discard the screen by pressing the close button. The YubiKey now installs as a default ANSI keyboard.

Verifying the installation can be done by bringing up the "About this Mac" dialog. Choose "More info…" and click "USB". The attached USB devices now appear. Click on the "Yubico YubiKey" and the properties appear



The Vendor ID 0x1050 and Product ID 0x0010 uniquely identify the YubiKey. The parameter "Bus Power (mA): 500" does not specify the power consumption of the YubiKey (which is max 30 mA) but rather what this specific port can supply.

3.4 Installing the YubiKey on other platforms

The YubiKey is used on a wide variety of platforms and similar straight-forward principles of identifying the USB HID device and pairing it with the appropriate standard keyboard drivers typically apply. If any

operating system specific questions arise, please check out on the Yubico developer's forum on-line or ask for support from Yubico.

3.5 Understanding the LED indicator

The LED indicator shows the status of the YubiKey. For YubiKey NEO, please refer to section 7.8

3.5.1 Error or no power – constant off

If the ring does not light up at all, the device does not work properly. Ensure that the device is inserted with the ring facing up and it is properly seated in the USB contact. If connected to a hub, ensure that the hub has power if needed.

3.5.2 Power down – occasional blips (YubiKey 1 only)

If the host computer enters power-down mode (hibernation or suspend) and stops polling the USB interface, the YubiKey also enters power-down mode. A short green "blip" is then yielded approximately every 8 seconds.

3.5.3 Device enumeration or error condition – rapid flashing

During USB device enumeration process, the LED flashes rapidly with a rate of approximately 4 Hz. The quick flashing also occurs when an invalid operation is triggered, such as trying to trigger an un-configured OTP configuration.

3.5.4 Non-configured – flash(-es) every 2 seconds

If the YubiKey does not have a valid configuration written to it, one or two short flashes are yielded approximately every 2 seconds. Without a valid configuration the device won't emit OTPs. Trying to emit a code in this mode will cause a short burst of flashes to indicate that the code cannot be generated.

3.5.5 Ready – Constant on

When a valid configuration is present and the device is ready to emit an OTP, the indicator shows a steady green light.

3.5.6 Ready to update or challenge trigger – slow flashing (YubiKey 2/2.2 only)

When the device is ready to update a parameter the indicator flashes slowly (approx 2 Hz). Pressing the key again commits the update. Waiting for 8 seconds automatically terminates the update operation.

The same flashing apply when the YubiKey is configured to require an user confirmation when receiving a challenge request. The user must then respond to the challenge by touching the YubiKey button within 15 seconds

or the request will be terminated. The challenge-response feature is available from YubiKey version 2.2.

3.6 Testing the installation

The quickest way to test the YubiKey that it works correctly is to open up a text editor, word processor or command prompt. Simply touch the button and an OTP string shall appear like

```
fifjgjgkhchbbvdjvfblveechbhdklchbjhcvluvlcfk
```

If the YubiKey is configured to work with Yubico's validation server, there is a test page where the actual output can be tested.

3.7 Installation troubleshooting

The fundamental principle of the YubiKey is that the installation is quick, automatic and painless. If however something fails during the installation, please verify the following.

3.7.1 The key is inserted and the indicator light does not light up

This probably means that the YubiKey does not have power

- Verify that the YubiKey is properly seated in the USB port
- Verify that the YubiKey is not inserted upside-down.
- If attached to an external hub, check that the hub has power
- Verify that another USB device works in the same port

3.7.2 The key is inserted, the indicator light flashes shortly and then goes out after a few seconds

This probably means that the YubiKey has entered power down. This is the normal behavior to conserve power when the computer enters suspend/hibernation.

- Verify that the YubiKey is properly seated in the USB port
- Verify that another USB device works in the same port

3.7.3 The key is inserted and the indicator just flashes rapidly

This means that the YubiKey has not yet been recognized by the computer and its operating system.

- Verify that the YubiKey is properly seated in the USB port
- Verify that another USB device works in the same port.
- Verify that there is no computer policy/setting that prevents attachment of external devices.

3.7.4 The key is inserted and the indicator flashes every 2 seconds

This means that the YubiKey has not been properly configured and is therefore unable to create an OTP. Check with the YubiKey issuer for a replacement.

3.7.5 Nothing happens when the trigger button is pressed

- There must be at least one valid configuration present. Ensure the green light is shining steadily.

- Hold the button steady for about 0.5 seconds and the OTP is triggered.
- Touch with a naked finger and not a pen, pointer, eraser etc. Gloves and bandage won't work.
- For YubiKey 2, if there are multiple configurations touch the button shortly and release.
- Check if the YubiKey works on another computer

3.7.6 It appears like the light goes out when trigger button is pressed but nothing appears on the screen

- Verify that the cursor is placed in a valid input field
- Verify that the YubiKey is properly seated in the USB port

If the above does not resolve the issue, check out the Yubico forum on-line or send a problem description to support@yubico.com

4 Using the YubiKey

From a user perspective, there are just a few things to learn and understand in order to use the YubiKey. First, insert the YubiKey in the USB port with the button and gold contact facing up. When a steady green light is on, the YubiKey is ready to emit an OTP via the keyboard interface.

If the green light does not go on steadily, check the troubleshooting guide in section 3.7.

Under the gold button is a solid-state capacitive touch sensor that reacts on touch by a finger. There are no moving parts and unlike traditional mechanical or membrane touch buttons, no explicit force is necessary.

Ensure that the cursor is placed in a valid input field and touch the button with a fingertip and hold steady for approximately 0.5 seconds and the OTP string is emitted. The indicator will then be turned off for approximately 2 seconds where the touch button is disabled to prevent multiple triggers.

The sensor is designed not to react just when slightly touched or when a finger is swiped over it. The delay and an algorithm are used to prevent accidental triggering.

Touching with a pen or similar won't work. Furthermore, wearing gloves or having tape or bandage on the finger won't trigger the sensor.

4.1 Using multiple configurations (from version 2.0)

YubiKey 2 supports an optional second configuration. This allows the YubiKey to be used for multiple services where both configurations are completely separated from each other. A typical usage is that one configuration is used for a remote service and one for a local service in static mode.

If both configurations are set, the trigger behavior is slightly different as the user must select which OTP configuration that is desired:

- Short press 0.3 – 1.5 seconds) and release – OTP from configuration #1 is yielded
- Long press (2.5 – 5 seconds) and release – OTP from configuration #2 is yielded

4.2 Updating a static password (from version 2.0)

YubiKey 2 supports user-initiated update of a static password. If configured, the user presses and holds the key for 8-15 seconds. When the button is

released, the indicator light flashes. By pressing shortly, the change is confirmed and the new OTP is yielded.

4.3 Responding to a challenge (from version 2.2)

YubiKey 2.2 supports challenge-response, where an OTP trigger can be issued by a client-side application. If user interaction (permission) is required, the green light flashes slowly. Shortly press the YubiKey button and the response is generated. If the request is not acknowledged within 15 seconds, the challenge request is terminated with an error.

5 Configuring the YubiKey

The YubiKey behavior and output can be configured to precisely fit the desired mode of operation. Configuration data is written via the configuration interface, accessible via the USB port. The configuration data is stored in the non-volatile memory where it is kept even when the YubiKey is unplugged.

Configuring the YubiKey is typically made via the configuration API where a high-level interface is provided. The following sections describe the settings in general terms rather than from an application-, binary-level or API-level point of view.

The generalized format of the OTP output string looks like

`ref_string <TAB> fixed_string <TAB> OTP_string <TAB> <CR>`

5.1 The TKTFLAG_xx format flags

The output format is controlled by the **`TKTFLAG_xx`** settings. These are binary flags that can be turned either on or off. As the YubiKey functionality has been extended, the usage of flags have become interleaved to allow full backwards completely.

5.1.1 TKTFLAG_TAB_FIRST

When set, an initial TAB is sent before the fixed string

5.1.2 TKTFLAG_APPEND_TAB1

When set, a TAB is sent after the fixed string

5.1.3 TKTFLAG_APPEND_TAB2

When set, a TAB is sent after the OTP string

5.1.4 TKTFLAG_APPEND_DELAY1

When set, a 0.5 second delay is inserted after the fixed string

5.1.5 TKTFLAG_APPEND_DELAY2

When set, a 0.5 second delay is inserted after the OTP string

5.1.6 TKTFLAG_APPEND_CR

When set, an ENTER / Carriage Return character is sent last

5.1.7 TKTFLAG_PROTECT_CFG2 (from version 2.0 only)

This flag is not a format flag but is included here for backwards compatibility. See section 5.5.11 for a description of this flag.

5.1.8 TKTFLAG_OATH_HOTP (from version 2.1 only)

This flag is not a format flag but is included here for backwards compatibility. When set, the configuration is set to OATH-HOTP mode

5.1.9 TKTFLAG_CHAL_RESP (from version 2.2 only)

This flag is not a format flag but is included here for backwards compatibility. When set together with any of the **CFGFLAG_CHAL_YUBICO** (5.5.15) or **CFGFLAG_CHAL_HMAC** (5.5.16) flags, the configuration is set to Yubico OTP challenge-response mode.

5.2 The reference string

When set, a reference string of the Modhex characters 0..15 are outputted first. This can be used by the verifying application to verify the mapping of the Modhex characters.

5.3 The fixed string

The fixed string is used to identify a particular YubiKey device. The length of the fixed string can be set between 0 and 16 bytes. There is no internal logic for the fixed string and it is completely independent of the OTP part, i.e. no information from the fixed string is used in the OTP algorithm.

The fixed string is referred to as the "Token Identifier" in OATH-HOTP mode (see section 5.3.4)

5.3.1 Normal Modhex mode

The normal case is that the fixed string specifies a 1-16 byte (8 – 128 bits) binary string. The output is encoded in Modhex format, yielding 2 to 32 characters output as each Modhex character represents 4 bits of information

For example, a fixed string of 6 bytes in this mode with the following data:

```
0x84 0x05 0x06 0x1e 0x1f 0x20
```

This input in this mode yields the fixed string **jfcgchbubvdc**

More on Modhex encoding can be found in section 6.1

5.3.2 No fixed string

The fixed string is optional and may not need to be used in certain use cases.

- All YubiKeys in a collection share the same AES key. Each individual YubiKey then uses the private (secret) identity field to identify the individual device.
- The YubiKey is used in static mode and 32 or 16 characters is enough for the desired password strength.

5.3.3 Extended scan code mode (from version 2.0 only)

The YubiKey 2 also supports output by keyboard scan codes rather than default Modhex coding. When configured, each byte in the fixed string is treated as a keyboard scan code rather than a binary byte. Using this mode rise the potential caveat that it may give undesirable output depending on the keyboard national setting. For example, keyboard scan code 0x1c will result in the character Y on a computer configured for a North-American keyboard whereas it will result in the character Z on a computer configured for a German keyboard.

The specified string is treated as a collection of scan codes. Setting the most significant bit (0x80) in a byte specifies that it shall be preceded with a SHIFT.

For example, a fixed string of 6 bytes in this mode with the following data:

```
0x84 0x05 0x06 0x1e 0x1f 0x20
```

This input yields the fixed string **Abc123** on a computer set for a North-American keyboard.

There are several on-line resources available how scan codes map to specific characters. One can be found at

http://download.microsoft.com/download/1/6/1/161ba512-40e2-4cc9-843a-923143f3456c/scancode.doc

This mode is enabled by a combination of the flags **CFGFLAG_SHORT_TICKET** being set and the **CFGFLAG_STATIC_TICKET** being cleared. When this combination is set, the OTP part is not sent. This allows full backwards-compatibility with YubiKey 1 which does not support this feature.

YubiKey 2.1 supports 1-16 characters scan code strings whereas YubiKey 2.2 supports up to 38 characters

5.3.4 OATH-HOTP Token Identifier (from version 2.1 only)

The YubiKey supports the Class A Token Identifier Specification as outlined by openauthentication.org.

The general format of the 12 character Token Identifier is as follows:

OO	OMP	OATH Manufacturer Prefix. A two character prefix identifying the manufacturer. Yubico has applied for manufacturer prefix 'ub' to allow Modhex compatibility
TT	TT	Token Type. A two character token type identifier.
UUUUUUUU	MUI	Manufacturer Unique Identifier. An 8 character string that uniquely identifies the token.

The Token Identifier can be configured to be all numeric, OMP Modhex, OMP + TT Modhex or all Modhex.

5.4 Public identity / token identifier interoperability

As the public identity and token identifiers may be configured freely, there is a certain risk that tokens configured by different users independently may clash with the same identity. Although the secret cryptographic parts most certainly would never clash, interoperability issues may arise in an infrastructure where the *authenticator* or *identity provider* of a specific token is to be found.

In order to allow several different customers to assign their own private "namespaces", the concept of a customer prefix for the fixed string has been introduced. The customer prefix is a unique 16-bit number that is assigned by Yubico. The customer prefix database will be accessible as a global repository that can link a specific prefix to a specific authentication site.

5.4.1 Interoperability guidelines in Yubico OTP mode

The public identity may be 0 to 16 bytes in length but the following guidelines are to be followed in Yubico OTP mode in order to ensure interoperability

- IDs below 6 bytes in length are considered to be used in a private context and will therefore not be considered interoperable.
- IDs above 6 bytes in length are reserved for future extensions and should not be used
- IDs with 6 bytes in length with the first byte not being 0x28 are assigned by Yubico and shall not be used.
- IDs with 6 bytes in length are considered interoperable if the first byte is 0x28
- A 16-bit unique prefix stored in byte 1 (high) and byte 2 (low) identifies a particular user.
- Customer prefixes 0x0000 – 0x0009 are available freely for testing purposes and are not considered interoperable
- Bytes 3-5 identify the individual key according to the user's context. The three bytes form a 24 bit range which equals to 2^{24} or approximately 16.8 million combinations.

5.4.2 Interoperability guidelines in OATH-HOTP mode

The interoperability guidelines for OATH-HOTP mode is less strict defined by Yubico as users may apply for their own manufacturer at openauthentication.org. Yubico however provides a method to use the Yubico assigned OMP 'ub' with a Yubico assigned customer prefix.

- OMP must be set to 'ub'
- The YubiKey must be configured with the **CFGFLAG_OATH_FIXED_MODHEX2** flag set.
- TT must be set to 190 + customer_prefix / 1000 (Modhex encoded)
- First MUI digit is (customer_prefix MOD 1000) / 100
- Second MUI digit is (customer_prefix MOD 100) / 10
- Third MUI digit is customer_prefix MOD 10
- The remaining 5 MUI digits are the device identity according to the user's context. The 5 digits form a range with 100,000 unique combinations.

5.5 The OTP string and the CFGFLAG_xx flags

5.5.1 The CFGFLAG_xx configuration flags

Functional parameters are controlled by the **CFGFLAG_xx** settings. These are binary flags that can be turned either on or off.

5.5.2 CFGFLAG_SEND_REF

When set, a reference string of the modhex characters 0..15 are outputted first. This can be used by the verifying application to verify the mapping of the Modhex characters.

For YubiKey 2, this flag in combination with the flag **CFGFLAG_STRONG_PW2** replaces this string by the shifted character 1, typically mapped to a '!'. This can be used to meet strong password requirements where at least one character is required to be a "special chacter".

5.5.3 CFGFLAG_PACING_10MS

When set, an intra-character pacing time of 10 ms is added between each sent keystroke.

5.5.4 CFGFLAG_PACING_20MS

When set, an intra-character pacing time of 20 ms is added between each sent keystroke. Combined with the **CFGFLAG_PACING_10MS** flag, the total delay is 30 ms.

5.5.5 CFGFLAG_STATIC_TICKET

Setting this bit causes the OTP generation to be forced into static mode, i.e. the term OTP becomes somewhat misleading.

In static mode, the OTP generation algorithm is the same, but all dynamic fields are set to fixed values

The static mode is implemented to allow integration with legacy systems without the need for additional server-side software. See section 2.1.3 for more information about the static mode.

5.5.6 CFGFLAG_TICKET_FIRST (YubiKey 1 only)

YubiKey 1 supports swapping the order of the fixed string and the OTP string. When set, the OTP part is sent first and fixed part last.

Usage of this function is discouraged as it is not implemented in YubiKey 2.

5.5.7 CFGFLAG_ALLOW_HIDTRIG (YubiKey 1 only)

YubiKey 1 supports trigger from an external keyboard as well as by the trigger button. The function only works properly in Windows systems and reacts when the caps-lock, num-lock and scroll-lock update messages are sent out to all keyboards in the system. Quickly "double-tapping" on any of these keys on one attached keyboard will trigger an OTP on the YubiKey if this bit is set.

The function is not portable and usage of this function is discouraged as it is not implemented in YubiKey 2.

5.5.8 CFGFLAG_SHORT_TICKET (from version 2.0)

Setting this flag truncates the OTP part to 16 characters. This function is only meaningful in static mode as a truncated non-static OTP cannot be successfully decoded.

In order to maintain YubiKey 1 compatibility, the non-applicable combination of this flag being set in non-static mode enables the "Extended scan code mode" described in section 5.3.3.

5.5.9 CFGFLAG_STRONG_PW1 (from version 2.0)

Setting this flag enables generation of mixed-case characters required by password policy settings in some legacy systems.

Although a 128-bit password can be considered strong enough, if there is a specific requirement for a mix between uppercase- and lowercase characters, even a long OTP will fail.

Setting this flag causes the first two characters to be shifted. This means that an OTP string like

grjndvjfluejrjtlijukvgrrdhljjjgi

will be changed to

GRjndvjfluejrjtlijukvgrrdhljjjgi

5.5.10 CFGFLAG_STRONG_PW2 (from version 2.0)

Setting this flag enables generation of mixed character and digits required by password policy settings in some legacy systems.

Although a 128-bit password can be considered strong enough, if there is a specific requirement for a mix between characters and digits, even a long OTP will fail.

Setting this flag causes the first two digits in the range 0..7 to be shifted to numbers 1..8. This means that an OTP string like

grjndvjfluejrjtlijukvgrrdhljjjgi

will be changed to

6rjn3vjfluejrjtlijukvgrrdhljjjgi

If this flag is set together with the flag CFGFLAG_STRONG_PW1, the output will be

6RJn3vjfluejrjtlijukvgrrdhljjjgi

If this flag is set together with the flag CFGFLAG_SEND_REF, the reference string will be replaced with a shifted 1. The output will then be

!6rjn3vjfluejrjtlijukvgrrdhljjjgi

5.5.11 CFGFLAG_MAN_UPDATE (from version 2.0)

In order to support legacy password systems, the YubiKey 2 supports user-triggered static password change. The function is designed for the specific use case of a traditional login system with a stricter password policy where the user is asked to change their password on a regular basis.

The intended use case is like the following:

1. The user is asked to update their password.
12. The user enters their secret password. The user presses the YubiKey button and the current fixed password is yielded
13. The user is asked to enter the new password.
14. The user enters their secret password. The user presses and holds the YubiKey button for 10 seconds.
15. When released, a short tap updates the internal password with a new randomized one. The new OTP is sent.

16. The user is asked to confirm the new password.
17. The user enters their secret password. The user presses the YubiKey button again and the new password is sent.
18. The user completes the password change.

As the change function has no protection against unauthorized usage, there is a danger that an unauthorized person can sabotage a user's YubiKey by triggering this function.

5.5.12 TKTFLAG_PROTECT_CFG2 (from version 2.0)

As the name implies, this is actually a ticket format flag, but for compatibility reasons, this configuration parameter is stored in this fields.

The "protect configuration 2" bit is used to lock and/or protect the second configuration in a YubiKey. If the issuer of the key wants to prevent the user from assigning something to configuration 2, setting this flag will reject all attempts to write configuration 2.

However, given a scenario with a shared ownership of the YubiKey, the issuer of configuration #2 can protect the issuer of configuration #1 to block it. As long as the configuration #1 does not have this bit set, the configuration #2 can be updated. If the configuration #2 is successfully written with this bit set, writing a configuration with this bit set to configuration #1 has no effect.

5.5.13 CFGFLAG_OATH_HOTP8 (from version 2.1)

Together with the TKTFLAG_OATH_HOTP flag, this flag selects the length of the HOTP output. When set, the HOTP output is truncated to 8 digits, otherwise the HOTP output is truncated to 6 digits.

5.5.14 CFGFLAG_OATH_FIXED_MODHEXx (from version 2.1)

These flags control the format of the Token Identifier string. It can either be all numeric, the OMP Modhex, the OMP + TT Modhex or all Modhex.

CFGFLAG_OATH_FIXED_MODHEX1	First byte is Modhex
CFGFLAG_OATH_FIXED_MODHEX2	First two bytes are Modhex
CFGFLAG_OATH_FIXED_MODHEX	All bytes are Modhex

5.5.15 CFGFLAG_CHAL_YUBICO (from version 2.2)

This flag set together with **TKTFLAG_CHAL_RESP** (5.1.9) enables Yubico OTP challenge-response mode. When set, the configuration does not work in normal OTP mode

5.5.16 CFGFLAG_CHALRESP_HMAC (from version 2.2)

This flag set together with **TKTFLAG_CHAL_RESP** (5.1.9) enables HMAC-SHA1 challenge-response mode. When set, the configuration does not work in normal OTP mode

5.5.17 CFGFLAG_CHALRESP_BTN_TRIG (from version 2.2)

This flag is not a format flag but is included here for backwards compatibility. When set together with the **TKTFLAG_CHAL_RESP** flag, the challenge-response configuration requires user acceptance by touching the YubiKey button.

5.6 The EXTFLAG_xx extended mode flags

Additional extended properties are controlled by the **EXTFLAG_xx** settings. These are binary flags that can be turned either on or off.

Extended flags that are device specific rather than configuration specific (SERIAL_BTN_VISIBLE, SERIAL_USB_VISIBLE, SERIAL_API_VISIBLE) are merged, i.e. if a flag is set in any of the configuration, the device consider the flag to be set.

5.6.1 EXTFLAG_SERIAL_BTN_VISIBLE (from version 2.2)

This flag allows the serial number to be retrieved by holding the touch button while inserting the device into the USB port. Once the LED starts to flash, release the button and the serial number will then be sent as a string of digits.

5.6.2 EXTFLAG_SERIAL_USB_VISIBLE (from version 2.2)

This flag makes the serial number to appear in the USB descriptor's iSerialNumber field. Note that this makes each device unique from the host computer's view.

5.6.3 EXTFLAG_SERIAL_API_VISIBLE (from version 2.2)

This flag allows the serial number to be read via proprietary API calls

5.6.4 EXTFLAG_USE_NUMERIC_KEYPAD (from version 2.3)

In OATH-HOTP mode, numeric digits are sent rather than Modhex characters. This may cause problems with certain keyboard layouts. Setting the USE_NUMERIC_KEYPAD flag causes numeric character to be sent as keystrokes from the numeric keypad rather than the normal numeric keys on a 84-key keyboard.

5.6.5 EXTFLAG_FAST_TRIG (from version 2.3)

Setting this flag causes the trigger action to become faster. It only applies when one configuration is written. If both configurations are set, the flag has no effect.

5.6.6 EXTFLAG_ALLOW_UPDATE (from version 2.3)

Normally, a configuration has to be entirely re-written if anything is to be changed. The ALLOW_UPDATE flag allows certain non-security related flags to be modified after the configuration has been written. These flags are:

TKTFLAG_TAB_FIRST
TKTFLAG_APPEND_TAB1
TKTFLAG_APPEND_TAB2
TKTFLAG_APPEND_DELAY1
TKTFLAG_APPEND_DELAY2
TKTFLAG_APPEND_CR
CFGFLAG_PACING_10MS
CFGFLAG_PACING_20MS
EXTFLAG_SERIAL_BTN_VISIBLE
EXTFLAG_SERIAL_USB_VISIBLE
EXTFLAG_SERIAL_API_VISIBLE
EXTFLAG_USE_NUMERIC_KEYPAD
EXTFLAG_FAST_TRIG
EXTFLAG_ALLOW_UPDATE

EXTFLAG_DORMANT
EXTFLAG_LED_INV

It is important to keep the EXTFLAG_DORMANT flag set when updating the configuration if the desire is to keep it on. Once cleared, the flag cannot be set without a complete re-write of the configuration.

The flag can be combined with a write protection (see section 5.7)

5.6.7 EXTFLAG_DORMANT (from version 2.3)

This flags allow a configuration to be stored without being accessible. This is useful in deployments where a post-issuance activation of a configuration is desired. This option can be used together with write protection (see section 5.7), requiring users who want to activate a configuration to supply a valid configuration password.

5.6.8 EXTFLAG_LED_INV (from version 2.4)

This flag inverts the configured state of the LED. The default state is that the LED is constantly on when the device is configured. Setting this flag causes the LED to be off.

5.7 Write protection

In order to protect a configuration from being modified by an unauthorized instance, an optional access code can be specified at the time when a new configuration is written.

If an access code is configured for a configuration, this password must be supplied at each subsequent update attempt. If the supplied password does not match the stored password, the update is rejected.

For YubiKey 2 devices, each configuration has its own configuration access code.

5.8 Automatic navigation

In YubiKey 1, prior to version 1.3.5, a function was provided to allow automatic navigation when the device is inserted, where an URL string was outputted. This function is discouraged and has been removed in recent versions as it implies potential security and compatibility issues.

6 Implementation details

6.1 The Yubico OTP generation algorithm

The YubiKey OTP generation is made up of the following fields

uid	0	6	Private (secret) id
useCtr	6	2	Usage counter
tstp	8	3	Timestamp
sessionCtr	11	1	Session usage counter
rnd	12	2	Random number
crc	14	2	CRC16 checksum

6.1.1 Private ID

The private id field comprises 6 bytes copied from the private id field configuration value. This field can be used to store a private identity if shared encryption keys are used. Otherwise, this field can be set to all zeroes.

The verifying instance should verify this field against the expected value. If an OTP is encrypted with a non-matching AES key, this field will be invalid and the OTP shall in this case be rejected.

Alternatively, this field can be initiated with a random number, adding additional secret information in the plaintext.

When using the YubiKey in challenge-response mode, the private id is XORed with the challenge prior to OTP generation. Therefore, validation of a decrypted OTP response involves XORing the private id with the original value. The result shall then match the issued challenge if the OTP is valid.

6.1.2 Usage counter

The usage counter is a non-volatile counter which value is preserved even when the device is unplugged. The first time the device is used after a power-up or reset, this value is incremented by 1 and the session counter is set to zero

Bit 15 of this field is used by the YubiKey 1 to indicate that a trigger was initiated by an external (keyboard) trigger rather than by the integrated button. The verifying instance shall mask this bit before verifying the result. For the YubiKey 2, this bit is always zero.

For compatibility reasons, this means that the field is only 15 bits wide, giving a usable range of 1 – 0x7fff. When this counter reaches 0x7fff it stops there. One could think that this could lead to a YubiKey being practically useless during its lifetime if this occurs. However, considering a YubiKey being used five times a day, 365 days per year, it will take 18 years for the counter to get stuck. Furthermore, as this counter only increment the first time after power up / reset, the practical lifetime is even longer.

If for some strange reason the counter would ever reach the final value, it is probably so worn out that a replacement would be necessary. If it still looks

fine, the device can still be re-configured which would cause the counter to be reset.

Note that this finite nature of the counter makes Yubico OTP mode less practical to use in challenge-response settings where the interaction occurs very often. Consider HMAC-SHA1 mode instead for such settings.

The field is stored in little-endian format, i.e. the least significant byte being stored first.

6.1.3 Timestamp

The timestamp is a 24-bit field incremented with a rate of approximately 8 Hz. The timestamp value is set to a random value after startup from the internal random number generator.

This field may be used by the verifying party to determine the time elapsed between two subsequent OTPs received during a session. See section 2.3.3for further information about this topic.

This field wraps from 0xffffff to 0 without any further action. If used by the verifying party, this condition must be taken into account. Given an 8 Hz rate, the timer will wrap approximately every 24 days.

The field is stored in little-endian format, i.e. the least significant byte being stored first.

6.1.4 Session usage counter

At power up, the session usage counter is initiated to zero. After each new OTP has been generated, this field is incremented by one. If this field wraps from 0xff to 0, the usage counter field is automatically incremented.

6.1.5 Random number

A 16-bit random number is picked from the internal random number generator to add some additional entropy to the final result. One can always argue if this adds any additional security, but it surely does not hurt.

6.1.6 Checksum

A 16-bit ISO13239 1st complement checksum is added to the end. The checksum spans all bytes except the checksum itself. The checksum is verified by calculating the checksum of all bytes, including the checksum field. This shall give a fixed residual of 0xf0b8 if the checksum is valid. If the checksum is invalid, the OTP shall be rejected.

The field is stored in little-endian format, i.e. the least significant byte being stored first.

6.2 Modified Hexadecimal (Modhex) encoding

The Modhex encoding scheme was invented to cope with potential keyboard mapping ambiguities. See section 2.1.3for background information.

The Modhex mapping done like with hexadecimal coding but the output is mapped in the following way:

0	c	4	f	8	j	c	r
1	b	5	g	9	k	d	t
2	d	6	h	a	l	e	u

3	e	7	i	b	n	f	v

Examples:

- The hexadecimal byte **0x47** is represented as **fi**
- The hexadecimal string **0xba 0xad 0xf0 0x0d** is represented as **nlltvcct**

6.3 CRC16 calculation and verification

The CRC16 algorithm used follows the ISO13239 standard. The schoolbook implementation can be done as:

```
static unsigned short crc;

void initCrc(void)
{
    crc = 0xffff;
}

void updCrc(unsigned char val)
{
    int i, j;

    crc ^= val;
    for (i = 0; i < 8; i++) {
        j = crc & 1;
        crc >>= 1;
        if (j) crc ^= 0x8408;
    }
}

unsigned short getCrc(const unsigned char *bp, int bcnt)
{
    initCrc();
    while (bcnt--) updCrc(*bp++);

    return crc;
}

unsigned char verifyCrc(const unsigned char *bp, int bcnt)
{
    initCrc();
    while (bcnt--) updCrc(*bp++);

    return crc == 0xf0b8;
}
```

6.4 Random number generator

The standard YubiKey has a built-in random number generator that involves a Linear Feedback Shift Register (LFSR) that is fed from analog output of the touch sensor as well as asynchronous data from USB traffic.

Although not directly critical to the security of the YubiKey or the OTP generation algorithm, the random number generation yields reasonably high quality numbers given these unrelated sources.

The YubiKey NEO features a high-quality, cryptographically secure random number generator.

6.5 USB programming interface

Configuration of the YubiKey is done via the USB interface. Since the keyboard interface is basically a one-way function, i.e. sending keystrokes, writing configuration data is done by the means of writing HID feature reports.

A HID feature report has a usable payload of 8 bytes where the last byte is used to identify the sequence number, leaving 7 bytes for configuration data. Writing a full configuration frame involve writing of 10 feature reports = 70 bytes.

When the final feature report has been received, the frame checksum is verified over the first 64 bytes. If this matches the expected value, the configuration frame is considered valid.

The access code for the particular configuration is verified to match the supplied one. If they do not match, the update request is rejected. Otherwise the configuration record is written and the status program sequence number is incremented.

The programming application shall read the sequence number via a status query prior to performing an update operation. If the sequence number has not been incremented after the update operation, the operation has failed.

6.5.1 USB status query

The YubiKey status can be read by the means of a feature report. Apart from verifying configuration operations as described above, the status query is used by factory testing to verify the functionality of the touch sensor.

6.5.2 Serial number readout (YubiKey 2.2 only)

The Device serial number is read by writing a serial number read command. The serial number is then instantly read using a single feature report.

6.5.3 Challenge-response mode (YubiKey 2.2 only)

In challenge-response mode, the response spans multiple feature reports. Furthermore, as the response may not be available instantly (processing time and user button accept), a completion poll bit is implemented. Once set, the result is read as a sequence of feature repors

6.5.4 Additional resources

Please refer to the YubiKey *low-level Interface description* for a detailed explanation of the USB low-level interface.

Detailed examples and information of the implementation can be found in published source code libraries, accessible via the Yubico developer's web page.

7 The YubiKey NEO

The YubiKey NEO is an enhanced version of the standard YubiKey, featuring the following additional features

- Backwards-compatible with standard YubiKey
- Comprises a Common Criteria EAL5+ compliant secure element for storage and processing of cryptographic information
- Features a USB CCID Smartcard functionality.
- FIDO U2F support from firmware version 3.3
- JavaCard 3.0 / JCOP 2.4.2 R1 execution environment
- ISO14443A RFID/NFC interface
- NDEF applet installed for interaction with YubiKey functionality
- Mifare Classic emulation

7.1 Standard YubiKey compatibility

The NEO is backwards compatible with the standard YubiKey, excluding the following functions:

- Static password manual update (4.2)
- Intra-packet delays (5.1.4, 5.1.5)

The USB product ID is different, but the command set is identical. This allows the NEO to be used with existing configuration- and API tools.

7.2 NDEF messages

The NEO supports NDEF (NFC Data Exchange Format) messages, which can be configured through the YubiKey configuration interface and then be used through the NFC interface.

An NDEF message is configured for an existing OTP configuration. The NEO supports all specified record types, with the most common ones being the URI- and TEXT types. The resulting NDEF message is constructed as a concatenation between a configured URI and a generated OTP.

Example:

Configured URI: `http://www.testsite.com/?otp=`

Generated OTP: `niljijfcnfdbjeduvuthuugnvuuvgrnh`

Result: `http://www.testsite.com/?otp=niljijfcnfdbjeduvuthuugnvuuvgrnh`

The NEO emulates a "Type 4" tag and NFC interrogators that supports this type can get a "tap-and-go" experience.

7.3 Modes of operation

The NEO is shipped in YubiKey standard mode, where it is compatible with a standard YubiKey.

The NEO can be configured to operate in one of three modes:

- **Mode 0:** YubiKey standard keyboard emulation (OTP) only (default)
  The Smartcard mode is disabled
- **Mode 1:** Smartcard (CCID) mode only
  The YubiKey mode is disabled
- **Mode 2:** Composite YubiKey standard + CCID mode

From firmware version 3.3, additional modes have been added with the introduction of U2F support:

- **Mode 3:** Universal Second Factor (U2F) mode only.
- **Mode 4:** Composite OTP- and U2F mode.
- **Mode 5:** Composite U2F- and CCID mode.
- **Mode 6:** Composite OTP-, U2F- and CCID mode.

Each mode has its separate USB device- and interface descriptors, including a unique vendor id (VID). Once the mode has been changed, the device has to be power-cycled for the USB interface changes to take effect.

7.4 U2F mode of operation (version 3.3 and later)

The YubiKey U2F mode complies with the FIDO alliance U2F version 2 standard for U2F messages and U2FHID transport.The YubiKey button is used for test of user presence, where the user physically acknowledges registration- and authentication messages.

For more information about U2F, please refer to relevant FIDO U2F documentation.

7.5 CCID mode of operation

When configured in CCID mode, the NEO exposes a USB interface compliant to the Integrated Circuit(s) Cards Interface Device (CCID) specification. A host application can then send commands directly to the secure element of the NEO, just like if the NEO was attached as a smartcard in a CCID smartcard reader.

In CCID mode, the NEO communicates using the T=1 smartcard protocol.

In CCID-only mode, the NEO can be configured to have card presence states, emulating having a smartcard reader where a smartcard is inserted and removed. The configurations are:

- **Always present**
  The NEO reports that a card is permanently present. Touching the YubiKey button will cause the LED to toggle and the state of this flip-flop can be read through the secure element.
- **Insert- and removal enabled**
  After insertion of the NEO, it reports that a smartcard reader is present, but no smartcard is inserted. By touching the YubiKey button, the NEO reports that a smartcard has been inserted. Touching the button again causes the NEO to report that the card has been removed.

7.6 Auto eject enabled

The NEO will automatically report that the card has been removed after a configured time of inactivity.USB composite modes and concurrency.

When more than one function is enabled simultaneous (modes 2, 4, 5 and 6), the device becomes what is known as a "USB composite device". With this, one physical device (the YubiKey NEO) can expose more than one interface, which in this case is a combination of OTP-, U2F- and CCID modes.

In the case of a composite devices, some usability- and functional limitations arise as all interfaces share a single resource.

- **Any CCID mode:** When enabled, this function becomes default and the LED signals the same behavior as when in CCID-only mode. When either of the other functions (YubiKey standard or U2F) are accessed, these will use the secure element, hence interfering with the CCID mode of operation. In order to handle this, the CCID interface then sends a "card eject" command to the host, simulating that the card has been ejected and cannot be accessed. When the other function has been completed, the CCID interface then sends a "card insert" command to the host after a 3 second dwell time, telling it that the CCID functionality is available again. The CCID button functionality is disabled in composite modes.
- **OTP+U2F mode:** In this mode, the OTP mode is the default and the button and LED works just like in OTP-only mode. The U2F mode is activated by a request via the U2F interface, disabling the OTP functionality. When the U2F interface has been idle for more than 3 seconds, the OTP functionality becomes enabled again.

7.7 Contact-less mode (NFC) of operation

The contact-less (NFC) mode of operation is automatically enabled when the device is not plugged into the USB port. In NFC mode, the device exposes an ISO14443A interface, supporting the ISO14443-4 (T=CL) protocol. The command set is identical to the CCID mode of operation.

The YubiKey button and LED are not enabled in contactless mode.

7.8 Device status LED

The status LED reflects the current device state, which in the case of a composite device also depends on which interface is active.

Enumeration in progress
A burst of three flashes every 1.5 second

7.8.1 YubiKey OTP mode

The behavior in YubiKey OTP mode is the same as for the standard YubiKey (see section 3.5)

7.8.2 CCID mode

Interface not acquired, button not touched
Constant off with a short flash on every 1.5 second

Interface acquired, button not touched
Constant on with a short flash off every 1.5 second

CCID activity
Every time an APDU is exchanged, the LED flashes

7.8.3 U2F mode

Idle
The LED is constant off

Touch (test of user presence) pending
The LED flashes 50:50 at a rate of about once a second

Button touched
The LED is constantly off

7.9 Javacard execution enviroment

The NEO is built around a secure element, featuring Javacard 3.0 and JCOP 2.4.2 R1, which complies with the Global Platform specification version 2.1. With this, Javacard applets designed for traditional smartcards can be loaded into the NEO and accessed using tools and middleware that works with Javacard and CCID, such as PC/SC.

The YubiKey- and NDEF functionality is provided though pre-loaded Javacard applets. In YubiKey mode, all YubiKey functionality is provided through the USB controller, which makes the necessary translations, transparent to the user.

Access to the Global Platform manager requires the card manager keys, which are generated and kept secret by Yubico at time of manufacturing. Yubico provides a developer program to access this restricted information. Contact Yubico sales for additional information.