

# YubiKey Configuration Manual

---

The Configuration Tool for the Yubikey

**Version: 2.0**

**Date: 9<sup>th</sup> September 2009**

---

### **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 62  
111 22 Stockholm  
Sweden  
[info@yubico.com](mailto:info@yubico.com)

# Contents

<b>1</b>	<b>Document Information</b>	<b>4</b>
1.1	Purpose	4
1.2	Audience	4
1.3	Related documentation	4
1.4	Document History	4
1.5	Definitions	4
<b>2</b>	<b>Introduction</b>	<b>5</b>
2.1	System requirements	5
2.2	Random numbers	5
2.3	Security and cryptographic practices	5
<b>3</b>	<b>Installing the application</b>	<b>7</b>
<b>4</b>	<b>Using the application</b>	<b>8</b>
4.1	The Start page	8
4.2	The Task page	9
4.3	Common configuration items	10
4.3.1	Binary strings	10
4.3.2	Remember settings	10
<b>5</b>	<b>Create a dynamic configuration (OTP)</b>	<b>11</b>
5.1	Specify public identity	11
5.2	Specify private identity	12
5.3	Specify cryptographic key	13
5.4	Specify output parameters	13
5.5	Specify configuration protection	16
5.6	Writing the configuration to the Yubikey	17
5.6.1	Lock / protect configuration 2	17
5.6.2	The programming sequence	18
<b>6</b>	<b>Create a static configuration (password)</b>	<b>20</b>
6.1	Specifying output parameters	21
6.2	Specifying configuration protection	21
6.3	Writing the configuration to the Yubikey	21
<b>7</b>	<b>Remove an existing configuration</b>	<b>22</b>
7.1	Specifying configuration protection	22
7.2	Writing the configuration to the Yubikey	22
<b>8</b>	<b>Check the Yubikey type and version</b>	<b>23</b>
<b>9</b>	<b>Test the OTP output of a Yubikey</b>	<b>24</b>
<b>10</b>	<b>Convert between different formats</b>	<b>26</b>
<b>11</b>	<b>Program settings</b>	<b>28</b>
11.1	General settings	28
11.2	Programming output	28
11.3	Sounds at programming	29
<b>12</b>	<b>Specify a text file for configuration input</b>	<b>30</b>

# 1 Document Information

---

## 1.1 Purpose

The purpose of this documentation is to provide an in-detail understanding of the Yubikey configuration process using the Yubikey Configuration Utility.

The document does not cover a “systems perspective”, but rather focuses on the process of configuring the Yubikey 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.

Furthermore, basic knowledge of Yubikey concepts is assumed. More information about this topic can be found in the “Related documentation section”

## 1.3 Related documentation

- YubiKey Integrators' Guide – describes the configuration component
- The YubiKey Manual – Usage, configuration and introduction of basic concepts
- Yubico online forum – <http://forum.yubico.com>

## 1.4 Document History

Date	Version	Author	Activity
2009-09-09	2.0	JE	New release

## 1.5 Definitions

Term	Definition
YubiKey device	Yubico’s authentication device for connection to the USB port
YubiKey 1 YubiKey 2	The two labels for the two versions of the YubiKey marketed to date
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. Private Id or Secret Id
Ticket	A general term for an access code generated by the Yubikey, a.k.a. OTP.
OTP	One Time Password
Modhex	Modified Hexadecimal coding

## 2 Introduction

---

The Yubikey Configuration Utility, YubikeyConfig.exe, is a Microsoft Windows application designed to configure and verify a Yubikey authentication device. The application follows a step-by-step approach to make configuration easy to follow and understand, while still being powerful enough to exploit all functionality both of the Yubikey 1 and Yubikey 2.

The Yubikey Configuration Utility provides the following main functions:

- Programming a Yubikey in dynamic “OTP” mode
- Programming a Yubikey in static “password” mode
- Removing (invalidating) an existing configuration from a Yubikey
- Checking the type and firmware version of a Yubikey
- Test/verify the OTP output of a programmed Yubikey

The Yubikey Configuration Utility provides basic means of batch processing where a larger number of keys can be configured sequentially. Configuration input can be automatic and/or read from a text file. Output of the configuration process can be written to a text file which later can be imported into a third-party verification tool.

The Yubikey Configuration Utility is intended to be a stand-alone application. For integration of configuration functionality into a custom application in the Microsoft Windows environment, a configuration component is provided. Please refer to appropriate documentation in section 1.3.

Basic Yubikey concepts and terms are not covered by this document. For related information, please refer to section 1.3.

### 2.1 System requirements

The Yubikey Configuration Utility is designed to run on all Microsoft Windows Win32 environments from Windows 2000 and onwards. The configuration utility is a single-file executable that runs without any other dependencies.

One free USB port is required.

### 2.2 Random numbers

Where random number generation is used in the application, the random values are generated using the Win32 Crypto API function **CryptGenRandom**, which should satisfy most needs. There is no special seeding or additional obfuscation added.

### 2.3 Security and cryptographic practices

The user must be aware of the appropriate security- and cryptographic practices needed to maintain the integrity of the generated configurations.

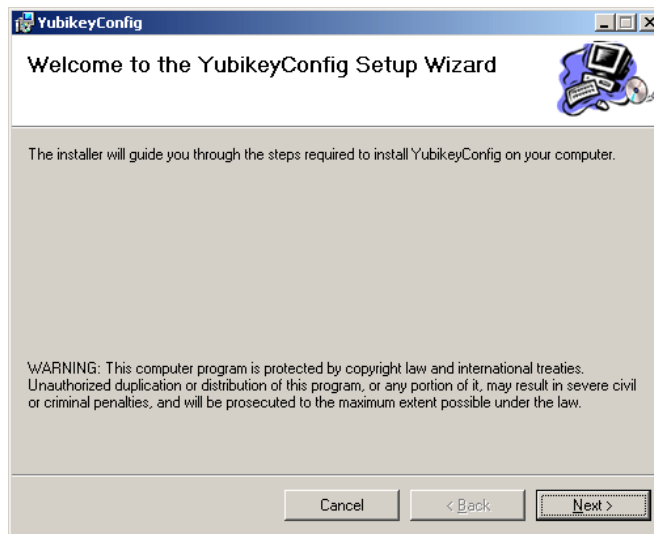
There is absolutely “no black magic” with the application in this respect, but given that cryptographically sensitive information is handled and

potentially read from and/or stored on persistent local store, security aspects need to be fully understood.

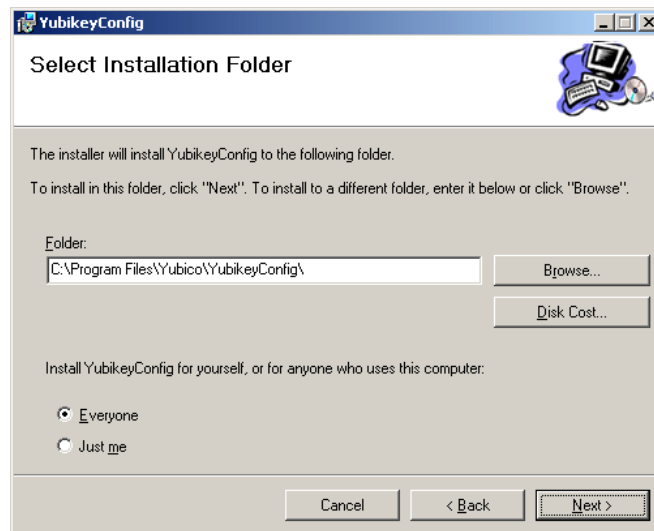
### 3 Installing the application

The Yubikey Configuration utility is a stand-alone application that runs without any other dependencies. This means that the application file **YubikeyConfig.exe** alone can simply be distributed to a second computer without an installer being run.

However, a basic Windows installer is provided for convenience of the first-time user:



Press Next to select installation location



Press Next > two times and then Close and the application and documentation (this document) should show up under the Start menu / All programs like:

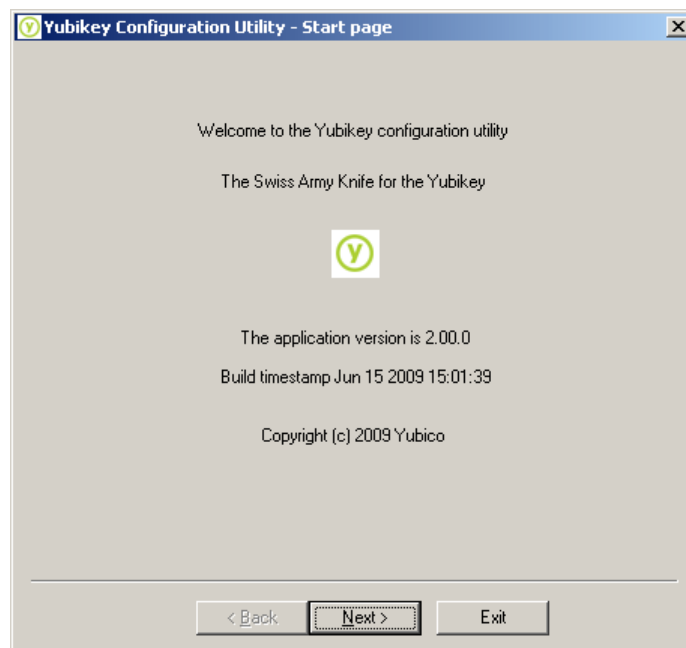


## 4 Using the application

Start the Yubikey Configuration Utility by launching the **YubikeyConfig.exe** application.

The Yubikey Configuration Utility is designed as a series of “pages”, following a task-oriented, step-by step process. Pages have “Next” and “Previous” buttons and when the final step in a task has been reached, the “Next” button is replaced with a “Finish” button.

### 4.1 The Start page

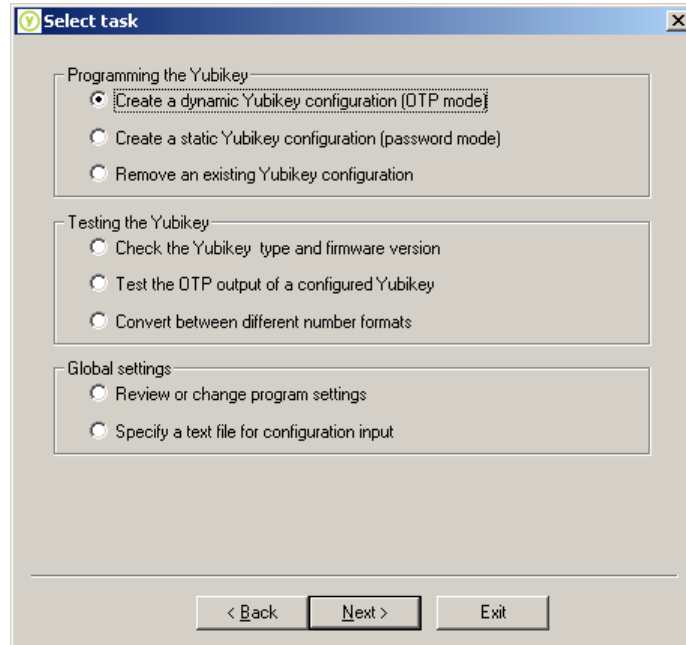


The start page displays the typical “About” information, including the application version number and build timestamp.

Press “Next >” to proceed to the task page

## 4.2 The Task page

The “main menu” of the Yubikey Configuration Utility is the Task page where the desired task is selected:



Select one of the following tasks and press Next >

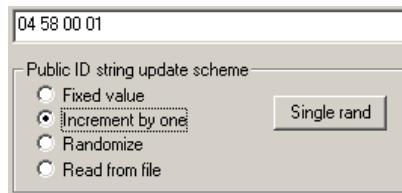
- Create a dynamic Yubikey configuration (OTP mode)  
A series of pages follows where input data is collected to create a configuration for a dynamic Yubikey configuration.
- Create a static Yubikey configuration (password mode)  
A few steps are needed to create a static Yubikey, typically used to generate a “hard to remember, impossible to guess” password.
- Remove an existing Yubikey configuration  
This invalidates (removes) an existing configuration from a Yubikey
- Check the Yubikey type and firmware version  
This selection provides a quick way to check the type and firmware of a Yubikey.
- Test the OTP output of a configured Yubikey  
This feature is primarily intended for “lab”- and tutorial purposes, where the OTP output of a newly generated Yubikey can be verified, dissected and “debugged”.
- Convert between different number formats  
This utility allows quick conversion between decimal-, hex- and Modhex formats
- Review or change program settings  
Here, a few persistent settings for the application can be set or reviewed.
- Specify a text file for configuration input  
Here, a text file holding pre-generated configuration input data can be selected.

## 4.3 Common configuration items

There are a few common items found on more than one page

### 4.3.1 Binary strings

In the case where binary strings are to be supplied, there is a common entry is similar at all instances



One of four options can be selected:

- Fixed value: A single fixed value is used for all devices being configured
- Increment by one: After each configuration has been written successfully, the string is incremented by one. The increment is done from right-to-left.
- Randomize: Prior to each program operation, a new randomized value is generated.
- Read from file: The input for this field is taken from the text file specified from the Task page. If no input file is specified (see section 12), this option is grayed out. Items will be taken from columns in the order they are referenced. First item is taken from column 1, second item from column 2 etc.

**Single rand** generates a single random number and sets this as the current fixed value.

### 4.3.2 Remember settings

For most users, some settings are seldom changed or the default settings are sufficient. In this case, the settings can be remembered and the page will be skipped next time.



The selection is cleared each time the application starts and if the "< Back" button is used.

## 5 Create a dynamic configuration (OTP)

Creating a dynamic Yubikey configuration is the task that requires the most input although most fields have appropriate default values.

The process requires the following sequence of steps:

1. Specify public identity
2. Specify private identity
3. Specify cryptographic key
4. Specify output parameters
5. Specify configuration protection
6. Write the configuration to the Yubikey

### 5.1 Specify public identity

The public identity is the first optional fixed part of the OTP string, used to identify a Yubikey. This field is sent in clear text.

Specify a public identity

Before the OTP string, a fixed identity can be used as a prefix which identifies a particular Yubikey. The fixed part is sent in clear text as opposite to the OTP part

Do not use a public identity

Use a public identity

Desired length:  (1 - 16 bytes)

Public ID string:

Public ID string update scheme

Fixed value

Increment by one

Randomize

Read from file

Remember these settings and don't ask next time

< Back   Next >   Exit

The public identity is optional. If there is no requirement for it, check the "Do not use a public identity".

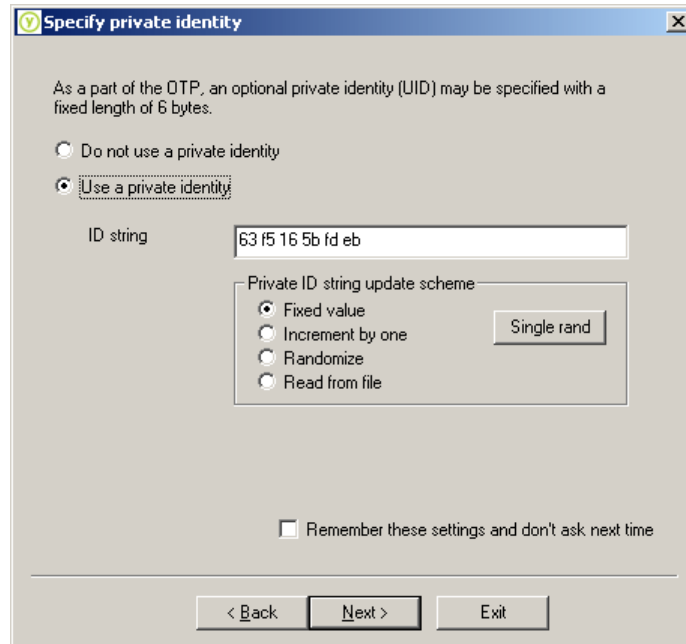
If used, a length between 1 and 16 bytes has to be specified. Any length between 1 and 5 bytes is considered a "private scope" and won't create any interoperability issues. A public ID length of 6 bytes or more is for use with the Yubico validation server architecture or for future extensions. We strongly disregard usage of a 6-byte or longer unless a 2 byte unique namespace prefix is acquired from Yubico.

The example above shows a public id string of 4 bytes. Converted to Modhex coding, this will be sent as **jbdvjccb**

The public identity string generation is performed as described in section 4.3.1.

## 5.2 Specify private identity

The private identity is a secret field, included as an input parameter in the OTP generation algorithm.

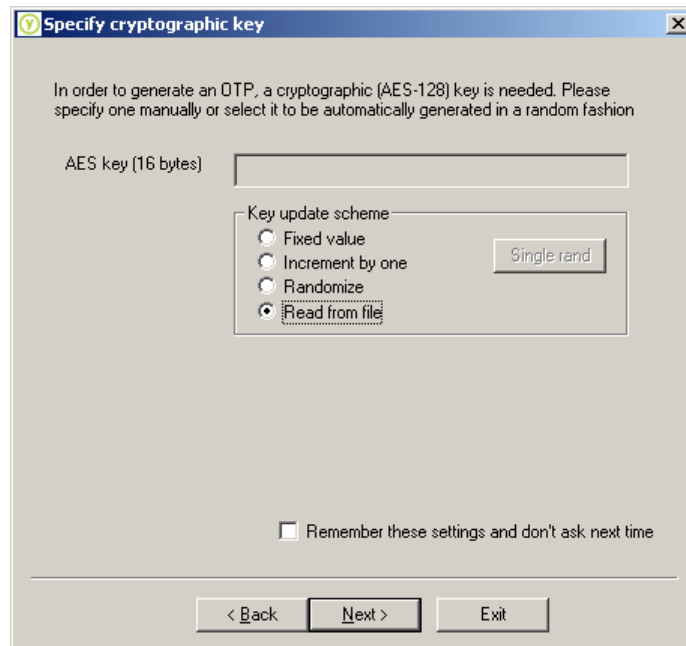


Utilizing the private identity field is optional. If there is no requirement for it, check the “Do not use a private identity” and the field will be forced to all zeroes.

If used, the private identity string generation is performed as described in section 4.3.1. The example shows a fixed random value that was generated by pressing the **Single rand** button.

### 5.3 Specify cryptographic key

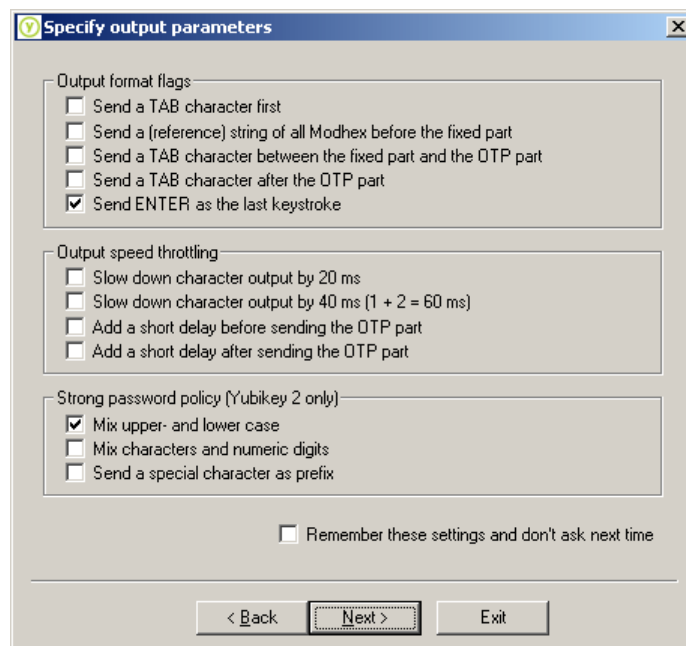
The cryptographic key is used to encrypt the OTP.



The key generation is performed as described in section 4.3.1. The example shows a key value that that will be read from the input file at the time of programming.

### 5.4 Specify output parameters

A collection of binary flags are provided to set various format- and behavioural parameters to either on or off.



The flags are organized in three groups

## Output format flags

### Send a TAB character first

The first character will be a TAB, typically used to move the cursor to the next input field

### Send a (reference) string of all Modhex characters before the fixed part

If there is an issue where keyboard mappings cause ambiguities, a reference string of all 16 Modhex characters can be sent first. This function cannot be used if "Mix characters and numeric digits" is set.

### Send a TAB character between the fixed part and the OTP part

In order to separate the fixed (public identity) part and the OTP part, a TAB can be inserted to move the cursor to the next input field

### Send a TAB character after the OTP part

A final TAB can be sent after the OTP part, typically used to move the cursor to the next input field after the password.

### Send ENTER as the last keystroke

This selection sends an ENTER as the final keystroke, typically used to trigger a default OK button or to complete input from a command prompt.

## Output speed throttling

Normally, the USB host polls the IN interrupt endpoint at the rate it can receive characters. The default poll rate is 10 ms which means that at full speed, a key entry is sent every 10ms. Each complete keystroke comprises a key-down and a key-up entry, which means that about 50 characters per second can be output.

If there are issues with lost characters due to a too high character output rate, the output can be slowed down.

### Slow down character output by 20 ms

This selection adds a 20 ms additional delay for each keystroke (10 at down and 10 at up). Given a default endpoint poll rate of 10 ms, this throttles the rate to about 25 characters per second.

### Slow down character output by 40 ms

To further slow down the output, an additional 40 ms delay can be added. Combining the 20 ms and 40 ms flags give an overall additional delay of 60 ms.

### Add a short delay before sending the OTP part

If there is some parsing - or GUI rendering delay for a particular application, an additional 500 ms delay can be inserted before sending the OTP.

### Add a short delay after sending the OTP part

If there is some parsing - or GUI rendering delay for a particular application, an additional 500 ms delay can be inserted after the OTP has been sent.

## Strong password policy

Although passwords with a 32 character length can be seen as strong enough, certain legacy systems enforce a stricter policy where a combination of uppercase and lowercase characters or a combination of numeric and alpha characters are required. The goal with this function is not to add additional bits of entropy but rather to meet legacy standards.

### Mix upper- and lower case

This flag enforces the two first Modhex characters to become shifted. Consider the OTP string:

**jfdkibjkegielgbgjkbejktuhurirnjt**

Setting this bit would create the following output

**JFdkibjkegielgbgjkbejktuhurirnjt**

### Mix characters and numeric digits

This flag enforces the two first Modhex characters in the range 0..7 to be replaced with digits 1...8. As this is not a part of the Modhex alphabet, it may cause incompatibilities with existing validation servers.

Setting this bit would create the following output:

**j53kibjkegielgbgjkbejktuhurirnjt**

Setting both bits would create the following output:

**J53Kibjkegielgbgjkbejktuhurirnjt**

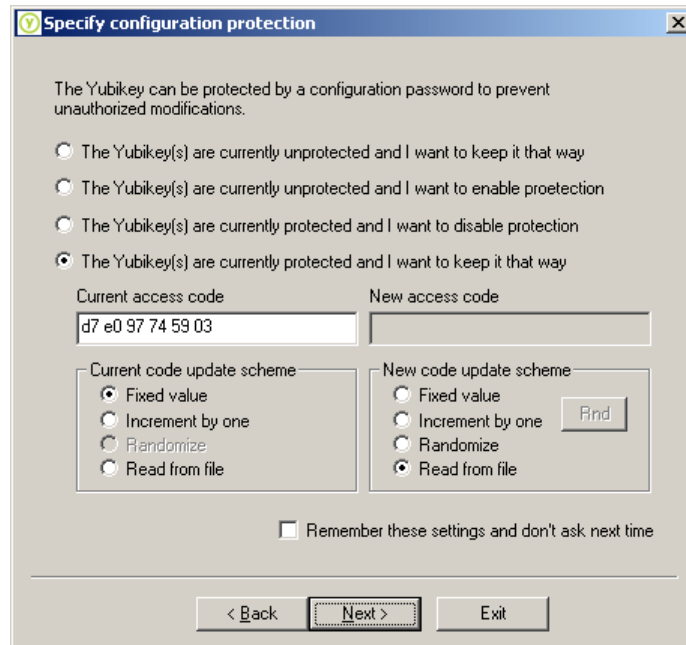
### Send a special character as prefix

Setting this flag will add a SHIFT-1 prefix, typically resulting in a '!' character. However, using this bit should be done with caution as SHIFT-1 has different mappings for different keyboard layouts. Furthermore, As this is not a part of the Modhex alphabet, it may cause incompatibilities with existing validation servers.

This function requires the "Mix upper- and lower case" flag to be set.

## 5.5 Specify configuration protection

To protect against unauthorized update of a specific configuration, a protection access code can be added. Then, in order to update or remove this configuration, the corresponding access code must be used, otherwise the request is rejected.

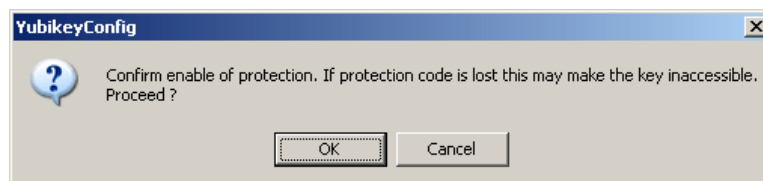


Four combinations are available:

- Unprotected -> Unprotected
- Unprotected -> Protected
- Protected -> Unprotected
- Protected -> Protected

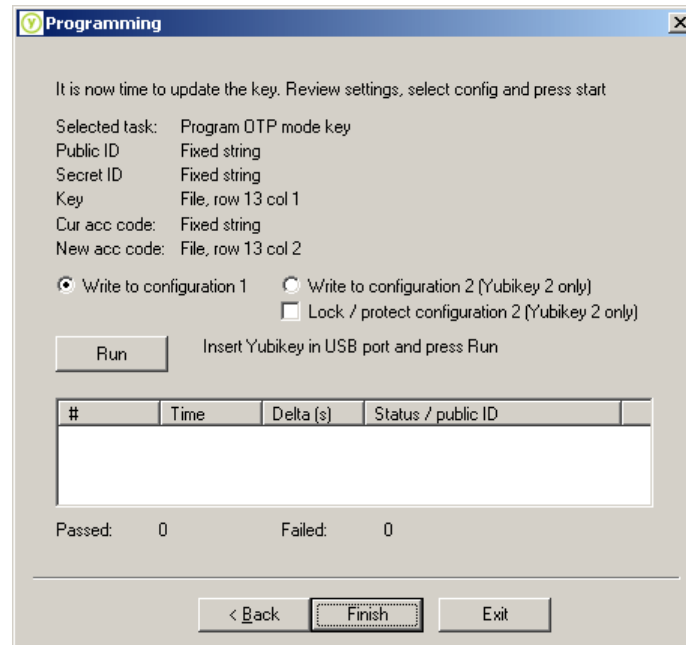
Generation of access codes is performed as described in section 4.3.1. In the example above, a key with a fixed access code **d7e097745903** will be changed to values read from the input file.

**Caution:** When protection is enabled the key can practically become “read-only” if the access code is lost. In order to enable protection, a confirmation dialog has to be confirmed first.



## 5.6 Writing the configuration to the Yubikey

The final step is to write the collected data to the configuration memory of the Yubikey.



Depending on the configuration settings (section 11.2) data about the programming is shown brief or verbose.

Before starting the programming, configuration 1 or 2 must be selected. Configuration 2 is valid for Yubikey 2 only.

### 5.6.1 Lock / protect configuration 2

This function allows the second configuration to be locked or prevented from being locked in a scenario where two different “owners” or “holders” control the first and second configuration respectively.

- When set, writing to configuration 1 prevents configuration 2 from being written, i.e. the “holder” of configuration 1 can block configuration 2.
- Writing to configuration 2 with this bit set will prevent the “holder” of configuration 1 from setting its lock bit.

Together with the configuration protection (see section 5), the Yubikey can be protected from unauthorized updates.

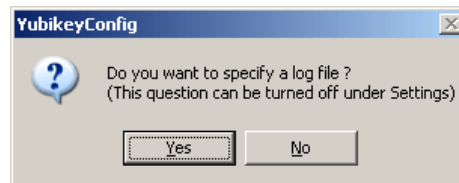
**Example 1:** A Yubikey is issued by a bank that uses configuration #1. The configuration is protected by an access password. The bank does not want the holder of the Yubikey to change the behaviour of the Yubikey and therefore sets the lock bit for protection #2. A casual user can then not write to configuration #2. Configuration #1 cannot be changed as there is an access password for it.

**Example 2:** Assume a user writing to configuration #2. This user does not know about the “owner” of configuration #1 and wants to protect itself from that “owner” from setting its lock bit. If the user set the protect bit

when writing to configuration 2, the owner of configuration #1 cannot lock update of configuration #2.

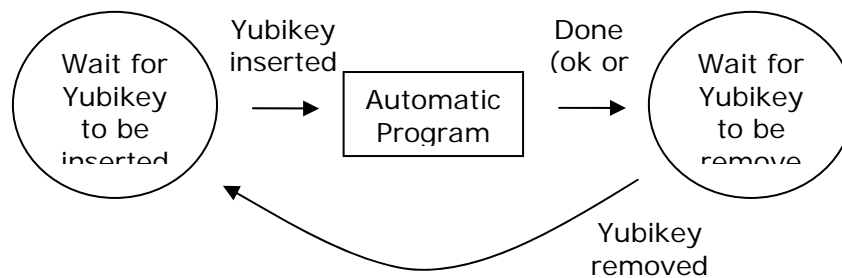
### 5.6.2 The programming sequence

When ready to start the programming, press Run. If the configuration is set to prompt for a log file (see section 11.2), the following question will appear:

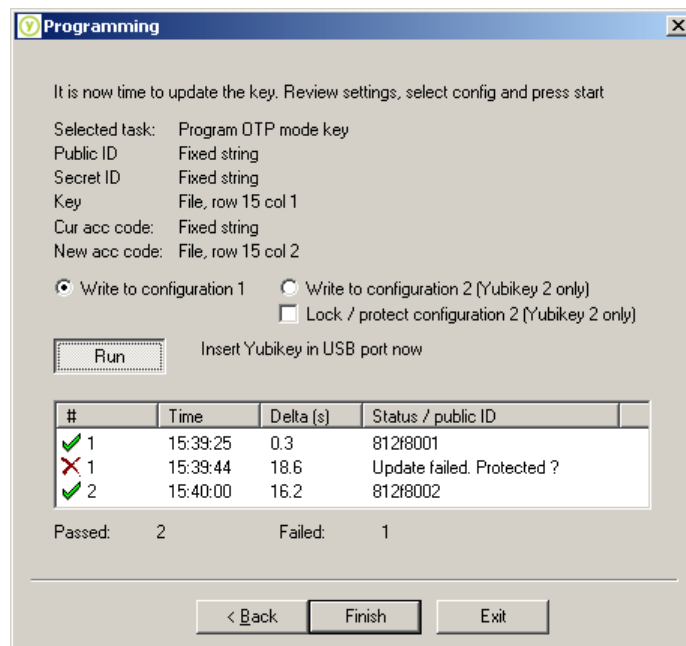


If Yes is pressed, a common "Save as..." dialog appears where the name and location of the log file is specified. Three different output formats are available – comma delimited, semicolon delimited or tab delimited.

Now, when the Run button is pressed, Yubikeys can be inserted, programmed and removed sequentially in a batch-like fashion.



The status output gets updated as programming progresses



Basic statistics show the number of passed and failed keys. In the example above, the second update failed where a Yubikey with "access protection code" set didn't pass. Just remove the failing Yubikey and insert the next one and the programming resumes.

The column **Delta (s)** shows the number of seconds elapsed since the previous programming operation. This gives a hint of the programming throughput, i.e. number of keys that can be programmed per time unit. For example, an average delta of 15 seconds gives a theoretical throughput of  $3600 / 15 = 240$  programmed keys per hour.

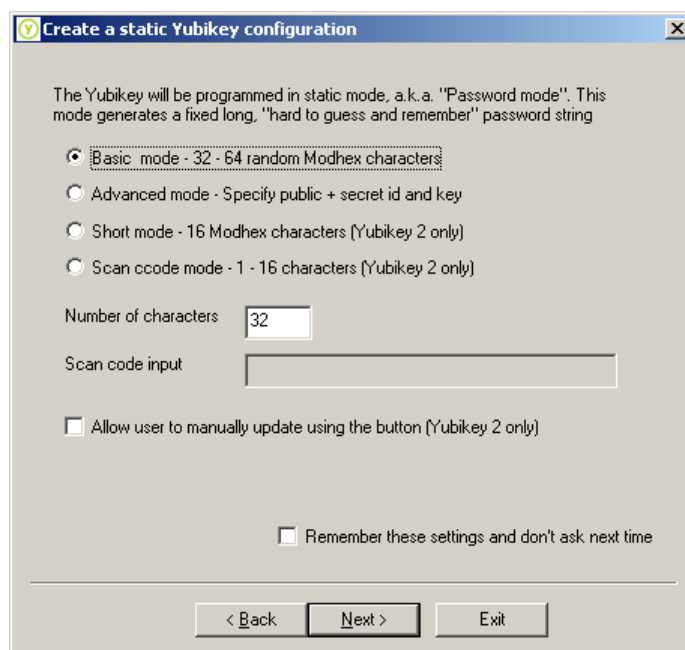
If set, the log file is opened for each successfully updated Yubikey and a new entry is added. Depending on the output settings (see section 11.2) either the public id is written or both (public + secret) fields are written.

In order to support efficient batch programming, sounds can be enabled under sound settings (see section 11.3). If set, two different tunes can be played depending if the programming operation passed or failed.

## 6 Create a static configuration (password)

The static mode is provided to create “hard to guess and remember” password. The process requires the following sequence of steps:

1. Select mode
2. Specify output parameters
3. Specify configuration protection
4. Write the configuration to the Yubikey



Depending on the requirements, four different modes are available.

### Basic mode

This is the simplest mode which applies default values to some underlying fields

### Advanced mode

When there is a desire to have partial compatibility with the dynamic / OTP mode, all underlying parameters can be specified. This adds additional steps, like programming the key for dynamic mode. See sections 5.1 to 5.3 for a description of the dynamic mode parameters.

### Short mode

This is like the basic mode but with the difference that the string is truncated to 16 digits. This feature is available for Yubikey 2 only.

### Scan code mode

The scan code mode provides a mechanism to generate a string based on any arbitrary keyboard scan code. This mode may create incompatibilities if different national keyboard layouts are used as the mapping varies between countries. This function available for Yubikey 2 only.

Simply place the cursor in the Scan code input field and type the desired string. The keystrokes are converted to scan codes.

### 6.1 Specifying output parameters

This is the same options as for dynamic configuration. Follow the instructions in 5.4.

### 6.2 Specifying configuration protection

This is the same options as for dynamic configuration. Follow the instructions in 5.5.

### 6.3 Writing the configuration to the Yubikey

This is the same options as for dynamic configuration. Follow the instructions in 5.6.

## 7 Remove an existing configuration

---

This task removes (invalidates) an existing configuration. The process requires the following sequence of steps:

1. Specify configuration protection
2. Write the configuration to the Yubikey

### 7.1 Specifying configuration protection

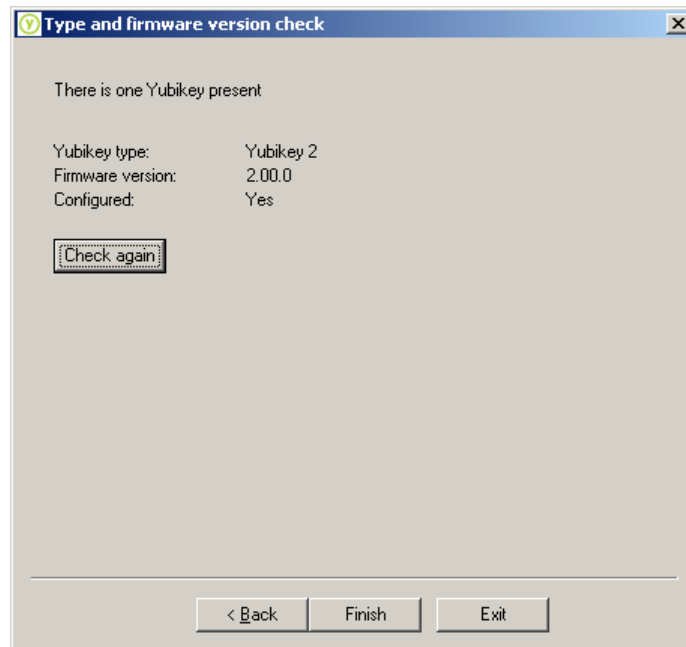
This is the same options as for dynamic configuration. Follow the instructions in 5.5.

### 7.2 Writing the configuration to the Yubikey

This is the same options as for dynamic configuration. Follow the instructions in 5.6.

## 8 Check the Yubikey type and version

This task provides a quick way to read out the Yubikey firmware revision and type.



## 9 Test the OTP output of a Yubikey

This task is for advanced users to provide a “lab” tool, primary for testing and tutorial purposes. Here, the OTP output is decrypted based on the most recently written configuration and the individual output fields are displayed.

Number of detected triggers:	6	Trigger by ENTER:	Yes	
Field 1			0	
Field 2	jbdvjccf		8	
Field 3	ltvfdtuftrbnjgttkitkndhfnudfcl		32	
Private id	63 f5 16 5b fd eb	Rnd	03f1	
Timestamp	13953534	52	PC tstp	6547
Counters	2	3	Ratio	7.94

The cursor defaults to the first of the three input fields. Press the Yubikey trigger button and the output is captured and then decoded.

The application automatically restores focus to the first input field but moving around the cursor and then pressing the Yubikey trigger button may lead to invalid results as the output may not appear where intended.

How to interpret the meaning of the decoded output is out of the scope of this document please refer to related documentation is listed in section 1.3.

**Field 1, 2 and 3** show the captured data entered into the input fields. The **Trigger by ENTER** shows if the output was finished with an ENTER keystroke or not.

**Private id** shows the decoded private id if successful. Otherwise the text “Invalid” is shown here. The distinction between a valid and an invalid OTP is determined only by verifying the decoded OTP timestamp.

**Timestamp** shows the current Yubikey timestamp output and the difference from the last received. The later field shall be related to the field **PC tstp**, which shows the delta in milliseconds taken from the PC clock.

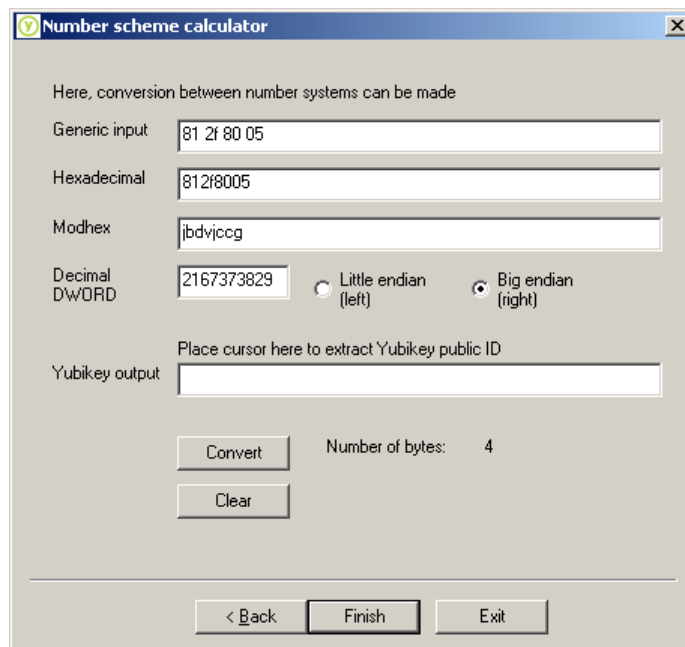
**Ratio** shows the ratio between the Yubikey timestamp and the PC timestamp. This value should be around 8 as this is the Yubikey timestamp frequency (in Hertz).

**Counters** shows the use counter and the session counter values

**Rnd** shows the random number value

## 10 Convert between different formats

A simple “calculator” is provided to allow quick conversion between different numeric representation formats



### Generic input

This field represents the common input field used throughout the application. This field type will parse the input string to determine if it is a delimited string, a packed hexadecimal or a Modhex one.

### Hexadecimal

This field shows and accepts only packed (non-delimited) hexadecimal strings

### Modhex

This field shows and accepts only packed (non-delimited) Modhex strings

### Decimal (DWORD)

This field uses the first four bytes to represent a 32-bit unsigned long integer (DWORD). Byte ordering can either be Little Endian (LSB leftmost) or Big Endian (MSB leftmost)

### Yubikey output

This fields allow automatic extraction and calculation of the public ID from a Yubikey output string. The general format is to extract the first 32 – string-length characters.

Consider the Yubikey output string

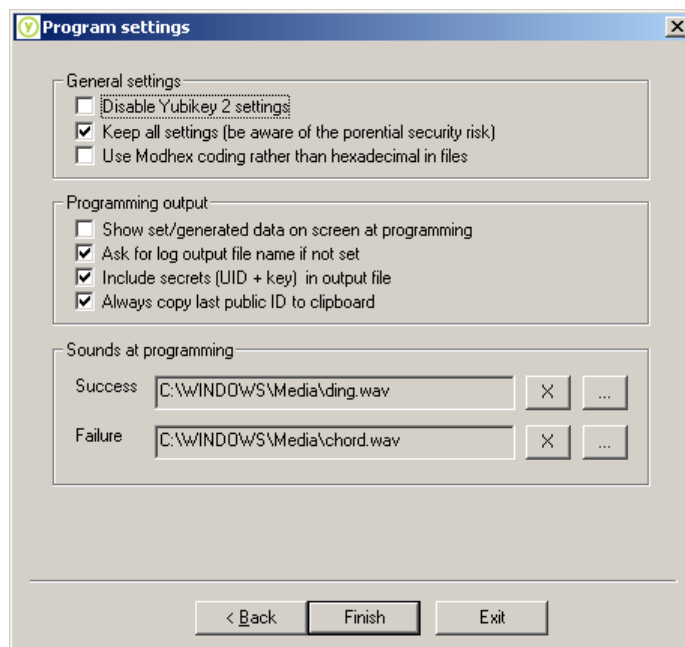
```
cccccccftkbufkhleivdrntelblbgkriigbjkugevb
```

Here, the first 12 bytes are the public id. Placing the cursor in this field and pressing the Yubikey button causes the first 12 bytes

(**cccccccf**tkb****) to be extracted and converted. The numeric ID 19857 is then displayed.

## 11 Program settings

Global persistent settings for the program are collected on a single page



### 11.1 General settings

#### **Disable Yubikey 2 settings**

When set, all Yubikey 2 specific options are disabled

#### **Keep all settings**

Normally, all selections for binary strings and schemes are reset when the program is exited so no security related information is stored persistently. Setting this flag causes all settings to be kept.

#### **Use Modhex coding rather than hexadecimal in files**

The default format in input and output files is hexadecimal. Setting this flag uses Modhex coding instead.

### 11.2 Programming output

#### **Show/set generated data on screen at programming**

For debugging and tutorial purposes, the generated fields (public id, secret id, key and access codes) can be displayed as they are to be written to the Yubikey.

#### **Ask for log output file name if not set**

Data written to the Yubikey(s) can be written to a sequential log text file. Setting this flag causes the application to prompt for a valid text file the first time a programming operation is initiated.

#### **Include secrets (UID + key) in output file**

Setting this flag included the secrets (private id and cryptographic key) to be included in the output log file.

**Always copy last public ID to clipboard**

When set, the last generated public ID is copied to the clipboard after a programming operation has been completed

## 11.3 Sounds at programming

To (potentially) make batch programming of keys more efficient, sounds can be added to programming operations. X clears the sound, ... browses for a file.

**Success sound**

When a programming operation is completed successfully, the sound specified here is played.

**Failure sound**

When a programming operation ends in a failure, the sound specified here is played.

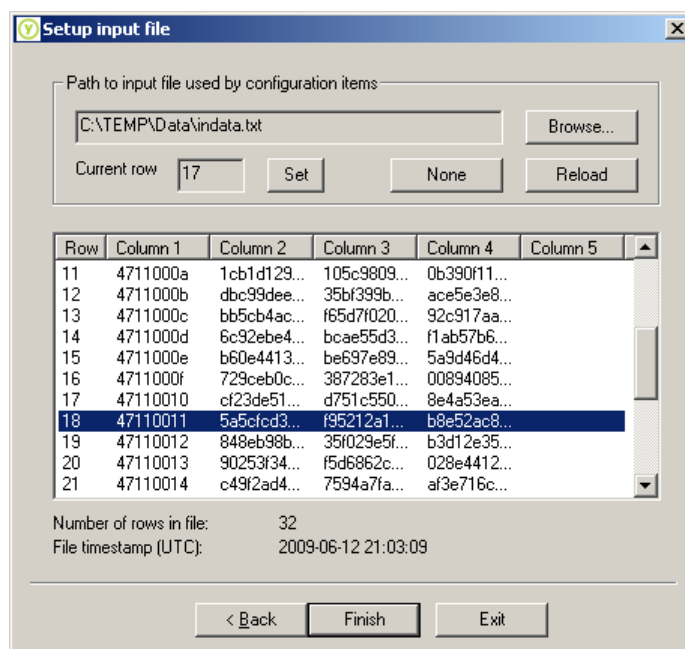
## 12 Specify a text file for configuration input

Input to data fields can be read from a file rather than being generated by the application. Any text file will do where hexadecimal- or Modhex items are separated by any non-valid character as delimiter

A sample input text file **indata.txt** comprising the data...

```
47110000,d92c3cb03cf7,b6e18a9ef958ba331958177d586a1cd0,2e69054a195a
47110001,bdfffb816243a,0ce571abb39e87df473fb7165e343854,3f007f4efc96
47110002,a70bafb07afc,a5a94934d0beed9feb1d0cb1682ec750,fa945da85285
... following lines cut
```

... gives the following result:



The application then uses the columns in the order they are referenced. In the example above, the text file contains four columns targeted for public id, private id, key and access code.

### Path to input file used by configuration items

#### Browse...

Browses for a file. Any delimited text file will do.

#### Reload

If the file has been modified while the application is running, pressing Reload will read in the newly updated data.

#### None

Clears the current file and flushes any buffered data.

#### Current row

This item shows the current row offset in the file. Each time a programming operation is been completed successfully, the row offset is incremented by 1 and the next row is read from the file.

**Set**

The file offset can be changed at any time by marking the appropriate row and then click "Set". Double-clicking a row yields the same result.

**Number of rows in file**

This field represents the number of rows read from the file.

**File timestamp (UTC)**

This field represents the file's OS timestamp, represented in Universal Time Coordinates (UTC) format.