

STM32MPx Series Signing Tool Software User Manual

Contents

- [1 STM32MPx Series Signing Tool Software](#)
- [2 Product Information](#)
- [3 Product Usage Instructions](#)
- [4 Frequently Asked Questions \(FAQ\)](#)
- [5 Introduction](#)
- [6 Install STM32MP-SignTool](#)
- [7 STM32MP-SignTool command-line interface](#)
 - [7.1 PKCS#11 solution](#)
- [8 Revision history](#)
- [9 IMPORTANT NOTICE – READ CAREFULLY](#)
- [10 Documents / Resources](#)
 - [10.1 References](#)
- [11 Related Posts](#)



STM32MPx Series Signing Tool Software

Product Information

Specifications:

- **Product Name:** STM32MP-SignTool
- **Version:** UM2543 – Rev 4
- **Release Date:** June 2024

Product Usage Instructions

Installation:

To install STM32MP-SignTool, follow the steps provided in the user manual.

Command-Line Interface:

The following commands are available when using STM32MP-SignTool from the command line:

- `-binary-image(-bin), -input(-in)`
- `-image-version (-iv)`
- `-private-key (-prvk)`

- `--public-key -pubk`

Examples:

Refer to the examples below to understand how to use STM32MP-SignTool effectively:

1. Example 1: Default algorithm selection and output file creation.
2. Example 2: Signing a binary file with header version 2 and multiple public keys.

Standalone Mode:

When using STM32MP-SignTool in Standalone mode, follow these steps:

1. Enter the absolute path first.
2. Provide the required information like algorithm selection, image version, entry point, and load address.

Frequently Asked Questions (FAQ)

• How do I verify the output image file?

You can verify the resulted image by parsing the output file and checking each header field. Use the command:

```
./STM32MP_SigningTool_CLI.exe -dump /home/user/output.stm32
```

• What is the difference between header versions for signing?

The header version determines the number of public keys required for authentication. For example, header version 1 requires one key path for STM32MP15xx products, while header version 2 and greater require eight key paths for others.

Introduction

- The STM32MPx series signing tool software (named STM32MP-SignTool in this document) is integrated in the STM32CubeProgrammer (STM32CubeProg).
- STM32MP-SignTool is a key tool that guarantees a secure platform and ensures the signing of binary images using ECC keys generated by STM32MP-KeyGen software (refer to the user manual STM32MPx series key generator software description (UM2542) for more details).
- The signed binary images are used during the STM32MPx series MPU secure boot sequence that supports a trusted boot chain. This action ensures an authentication and integrity check of the loaded images.
- STM32MP-SignTool generates a binary image file, a public key file, and a private key file.
- The binary image file contains the binary data to be programmed for the device.
- The public key file contains the ECC public key in PEM format, generated with STM32MP-KeyGen.
- The private key file contains the encrypted ECC private key in PEM format, generated with STM32MP-KeyGen.
- A signed binary file can also be generated from an already signed file with the batch file mode. In this case, the following parameters are not mandatory: the image entry point, the image load address, and the image version parameters.

Install STM32MP-SignTool

- This tool is installed with the STM32CubeProgrammer package (STM32CubeProg). For more information about the set-up procedure, refer to section 1.2 of the user manual STM32CubeProgrammer software

description (UM2237).

- This software applies to the STM32MPx series Arm®-based MPUs.

Note: Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

STM32MP-SignTool command-line interface

The following sections describe how to use STM32MP-SignTool from the command line.

Commands

The available commands are listed below:

- **–binary-image(-bin), –input(-in)**
 - **Description:** binary image file path (.bin extension)
 - **Syntax:** 1 -bin /home/User/binaryFile.bin
 - **Syntax :**2 -in /home/User/binaryFile.bin
- **–image-version (-iv)**
 - **Description:** enters the image version of the signed image file
 - **Syntax:** -iv <version_number>
- **–private-key (-prvk)**
 - **Description:** private key file path (.pem extension)
 - **Syntax:** -prvk <private_key_file_path>
 - **Example:** -prvk ../privateKey.pem
- **–public-key -pubk**
 - **Description:** public key file paths
 - **Syntax:** -pubk <File_Path{1..8}>
 - For header v1: use just one key path for STM32MP15xx products
 - For header v2 and greater: use eight key paths for others
- **–password (-pwd)**
 - **Description:** password of the private key (this password must contain at least four characters)
 - **Example:** -pwd azerty
- **–load-address (-la)**
 - **Description:** image load address
 - **Example:** -la <load_address>
- **–entry-point (-ep)**
 - **Description:** image entry point
 - **Example:** -ep <entry_point>
- **–option-flags (-of)**
 - **Description:** image option flags (default value = 0)
 - **Example:** -of <option_flags>
- **–algorithm (-a)**
 - **Description:** specifies one of the prime256v1 (value 1, default) or brainpoolP256t1 (value 2)
 - **Example:** -a <2>
- **–output (-o)**
 - **Description:** output file path. This parameter is optional. If not specified, the output file is generated at

the same source file path (for example, the binary image file is C:\BinaryFile.bin). The signed binary file is C:\BinaryFile_Signed.bin.

- **Syntax:** -o <Output_File_Path>
- **-type (-t)**
 - **Description:** binary type. Possible values are ssbl, fsbl, teeh, teed, teex and copro
 - **Syntax:** -t <type>
- **-silent (-s)**

Description: no message displayed for replacing an existing output file
- **-help (-h and -?)**

Description: shows help
- **-version (-v)**

Description: displays the tool version
- **-enc-dc (-encdc)**
 - **Description:** encryption derivation constant for FSBL encryption [header v2]
 - **Syntax:** -encdc <Deriv_hexVal>
- **-enc-key (-enck)**
 - **Description:** OEM secret file for FSBL encryption [header v2]
 - **Syntax:** -enck <Key_Path>
- **-dump-header (-dump)**
 - **Description:** parse and dump image header
 - **Syntax:** -dump <File_Path>
- **-header-version (-hv)**
 - **Description:** signing header version, possible values: 1, 2, 2.1, 2.2
 - **Example** for STM32MP15: -hv 2
 - **Example** for STM32MP25: -hv 2.2
- **-no-keys (-nk)**
 - **Description:** adding empty header without key options
 - **Notice:** need to disable authentication option with Option flags command.

Examples for STM32MP-SignTool

The following examples show how to use STM32MP-SignTool:

- **Example 1**

```
-bin /home/User/BinaryFile.bin -pubk /home/user/publicKey.pem -prvk  
/home/user/privateKey.pem -iv 5 -pwd azerty -la 0x20000000 -ep 0x08000000
```

The default algorithm (prime256v1) is selected and the option flag value is 0 (default value). The signed output binary file (BinaryFile_Signed.bin) is created in the /home/user/ folder

- **Example 2**

```
-bin /home/User/Folder1/BinaryFile.bin -pubk /home/user/publicKey.pem -prvk  
/home/user/privateKey.pem -iv 5 -pwd azerty -s -la 0x20000000 -ep 0x08000000 -a 2 -o  
/home/user/Folder2/Folder3/signedFile.bin
```

The BrainpoolP256t1 algorithm is selected in this case. Even if Folder2 and Folder3 does not exist, they are created. With the `-s` command, even if a file exists with the same specified name, it is automatically replaced without any message.

- **Example 3**

Sign a binary file using header version 2 that includes eight public keys for the authentication flow.

```
./STM32MP_SigningTool_CLI.exe -bin /home/user/input.bin -pubk publicKey00.pem  
publicKey01.pem publicKey02.pem publicKey03.pem publicKey04.pem publicKey05.pem  
publicKey06.pem publicKey07.pem -prvk privateKey00.pem -pwd azerty -t fsbl -iv  
0x00000000 -la 0x20000000 -ep 0x08000000 -of 0x80000001 -o /home/user/output.stm32
```

- **Example 4**

Sign a binary file using header version 2 that includes eight public keys for authentication plus encryption flow.

```
./STM32MP_SigningTool_CLI.exe -bin /home/user/input.bin -pubk publicKey00.pem  
publicKey01.pem publicKey02.pem publicKey03.pem publicKey04.pem publicKey05.pem  
publicKey06.pem publicKey07.pem -prvk privateKey00.pem -iv 0x00000000 -pwd azerty -la  
0x20000000 -ep 0x08000000 -t fsbl -of 0x00000003 -encdc 0x25205f0e -enck  
/home/user/OEM_SECRET.bin -o /home/user/output.stm32
```

- **Example 5**

Verify the resulted image by parsing the output file and check each header field.

```
./STM32MP_SigningTool_CLI.exe -dump /home/user/output.stm32
```

- **Example 6**

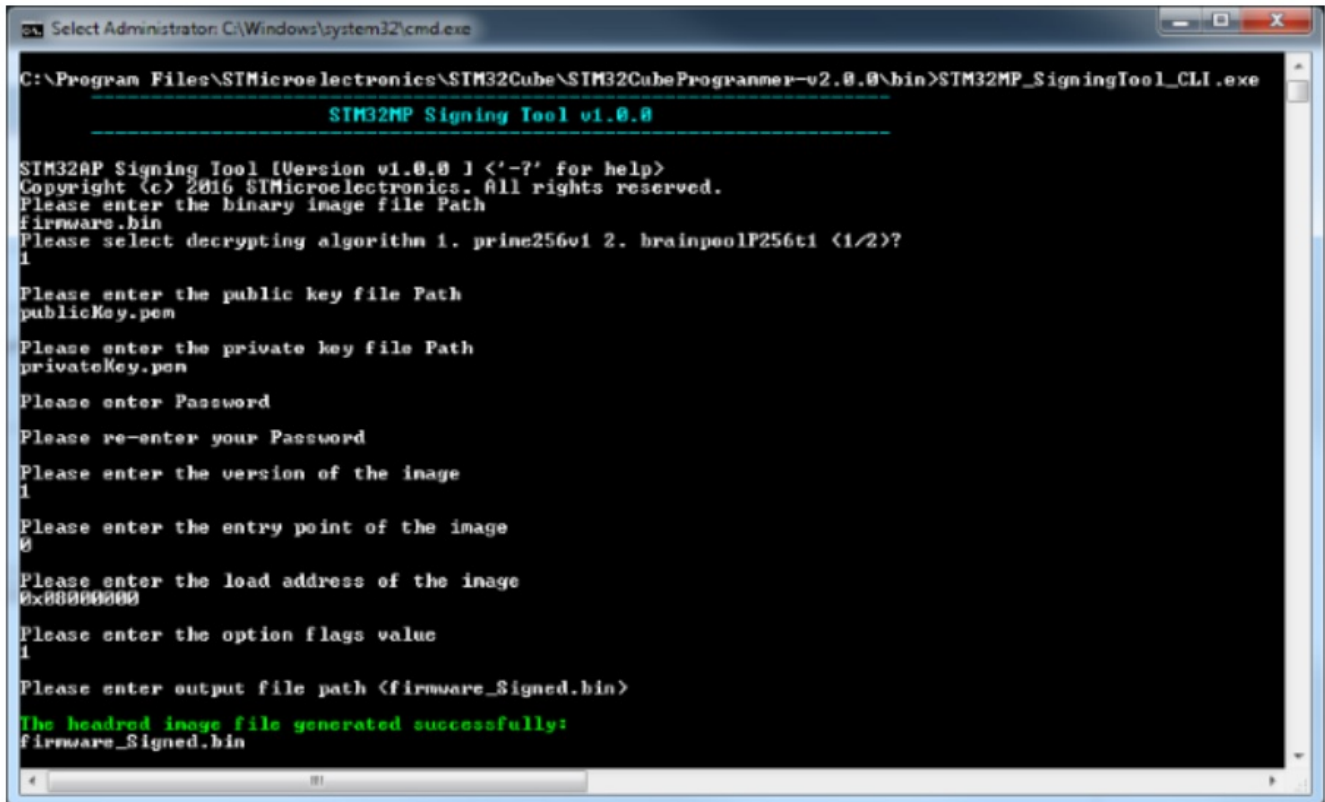
Add a header without signing and without deploying keys.

```
STM32MP_SigningTool_CLI.exe -in input.bin -nk -of 0x0 -iv 1 -hv 2.2 -o output.stm32
```

Standalone mode

When executing STM32MP-SignTool in Standalone mode, an absolute path must be entered first. A password is then requested twice for confirmation, as shown in the figure below.

Figure 1. STM32MP-SignTool in Standalone mode



```
Select Administrator: C:\Windows\system32\cmd.exe

C:\Program Files\STMicroelectronics\STM32Cube\STM32CubeProgrammer-v2.0.0\bin>STM32MP_SigningTool_CLI.exe

STM32MP Signing Tool v1.0.0

STM32AP Signing Tool [Version v1.0.0] <'-' for help>
Copyright (c) 2016 STMicroelectronics. All rights reserved.
Please enter the binary image file Path
firmware.bin
Please select decrypting algorithm 1. prime256v1 2. brainpoolP256t1 <1/2>?
1
Please enter the public key file Path
publicKey.pem
Please enter the private key file Path
privateKey.pem
Please enter Password
Please re-enter your Password
Please enter the version of the image
1
Please enter the entry point of the image
0
Please enter the load address of the image
0x00000000
Please enter the option flags value
1
Please enter output file path <firmware_Signed.bin>
The headred image file generated successfully:
firmware_Signed.bin
```

The next steps are the following:

- Select one of the two algorithms.
- Enter the image version, the image entry point, and the image load address.
- Enter the option flag value.

Another output file path can be specified if needed, or press enter to continue with the existing one.

PKCS#11 solution

- The signed binary images are used during the STM32MP secure boot sequence that supports a trusted boot chain. This action ensures an authentication and integrity check of the loaded images.
- The classic signing command requests that all public and private keys be provided as input files. These are directly accessible by any person who is allowed to execute the signing service. Ultimately, this can be considered to be a security leak. There are several solutions to protect keys against any attempts to steal key data. In this context, the PKCS#11 solution has been adopted.
- The PKCS#11 API can be used to handle and store cryptographic keys. This interface specifies how to communicate with cryptographic devices such as HSMs (hardware security modules) and smartcards. The purpose of these devices is to generate cryptographic keys and sign information without revealing private-key material to the outside world.
- Software applications can call the API to use these objects for:
 - Generate symmetric/asymmetric keys
 - Encryption and decryption
 - Computing and verifying the digital signature
- PKCS #11 presents to applications a common, logical view of the device that is called a cryptographic token

and it assigns a slot ID to each token. An application identifies the token that it wants to access by specifying the appropriate slot ID.

- The STM32SigningTool is used to manage the key objects stored on smartcards and similar PKCS#11 security tokens where sensitive private keys never leave the device.
- The STM32SigningTool uses the PKCS#11 interface to manipulate and sign input binaries based on ECDSA public/private keys. These keys are stored in security tokens (hardware or software).

Additional PKCS#11 commands

- **-module (-m)**
 - **Description:** specify a PKCS#11 module/Library path to load (dll, so)
 - **Syntax:** -m <Module_Path>
- **-key-index (-ki)**
 - -key-index (-ki)
 - **Description:** list of used keys indexes in hex format
Use one index for header v1 and eight indexes for header v2 (separated by space)
 - **Syntax:** -ki <values>
- **-slot-index (-si)**
 - **Description:** specify the index of the slot to use (default 0x0)
 - **Syntax:** -si <hexValue>
- **-active-keyIndex (-aki)**
 - **Description:** specify the actual active key index (default 0)
 - **Syntax:** -aki < hexValue >

PKH/PKTH file generation

After the processing of the signing operation, the tool systematically generates the PKH files to use after for OTP fuse.

- PKH file named pkcsHashPublicKey0x{active_key_index}.bin for header v1
- PKTH file named pkcsPublicKeysHashHashes.bin for header v2

Examples

The tool can sign input files for both header v1 and header v2, with a minimal difference in the command line.

• Header v1

```
-bin input.bin -iv <value> -pwd <value> -la <value> -ep <value> -t <type> -of <value> -  
-key-index <value> -aki 0 --module <module_path> --slot-index <index> -o output.stm32
```

• Header v2

```
-bin input.bin -iv <value> -pwd <value> -la <value> -ep <value> -t <type> -of <value> -  
-key-index <value0> <value1> <value2> <value3> <value4> <value5> <value6> <value7> -aki  
<active_index> --module <module_path> --slot-index <index> -o output.stm32
```

- An error on the command line, or an inability of the tool to identify the key objects that match, causes an error message to be displayed. This indicates the source of the problem.
- The SigningTool is able only to use preconfigured HSMs, and it is not designed to manage or create new

security objects. Therefore, it is necessary to install free software to set up a suitable environment. The keys can then be generated, and information about objects obtained.

Error examples:

- Invalid slot index

Figure 2. HSM TOKEN_NOT_RECOGNIZED

```
Token Info:
Manufacturer ID : SoftHSM project
Label          :
Model          : SoftHSM v2
Serial number   :
PIN min lenght  : 4
PIN max lenght  : 255
Hardware version : 2.5
Firmware version : 2.5
Error: CKR_TOKEN_NOT_RECOGNIZED
Error: CKR_CANCEL
```

- Unknown key object that is mentioned in –key-index command

Figure 3. HSM OBJECT_HANDLE_INVALID

```
Public key search object :
ID      : 0x1200
Error: CKR_OBJECT_HANDLE_INVALID
Error: Cannot extract public key from pkcs11 module !
```

The tool treats the objects sequentially. If it cannot identify the matching key objects on the first try, the signing operation stops the process. An error message is then displayed to indicate the source of the problem.

Revision history


Date	Version	Changes
14-Feb-2019	1	Initial release.
26-Nov-2021	2	Updated: <ul style="list-style-type: none"> Section 2.1: Commands Section 2.2: Examples for STM32MP-SignTool Added Section 2.4: PKCS#11 solution
27-Jun-2022	3	Updated Section 2.1: Commands
26-Jun-2024	4	Replaced in the whole document: <ul style="list-style-type: none"> STM32MP1 series by STM32MPx series STM32MP1-SignTool by STM32MP-SignTool STM32MP1-KeyGen by STM32MP-KeyGen Updated –public-key -pubk and added –header-version (-hv) and –no-keys (- nk) in Section 2.1: Commands. Added “Exemple 6” in Section 2.2: Examples for STM32MP-SignTool .

IMPORTANT NOTICE – READ CAREFULLY

- STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgment.
- Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of purchasers’ products.
- No license, express or implied, to any intellectual property right is granted by ST herein.
- Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.
- ST and the ST logo are trademarks of ST. For additional information about ST trademarks, refer to www.st.com/trademarks. All other product or service names are the property of their respective owners.
- Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2024 STMicroelectronics – All rights reserved

Documents / Resources

	<p>ST STM32MPx Series Signing Tool Software [pdf] User Manual</p> <p>STM32MPx Series Signing Tool Software, STM32MPx Series, Signing Tool Software, Tool Software, Software</p>
--	---

References

- [User Manual](#)

Manuals+, Privacy Policy

This website is an independent publication and is neither affiliated with nor endorsed by any of the trademark owners. The "Bluetooth®" word mark and logos are registered trademarks owned by Bluetooth SIG, Inc. The "Wi-Fi®" word mark and logos are registered trademarks owned by the Wi-Fi Alliance. Any use of these marks on this website does not imply any affiliation with or endorsement.