STM32MP1 Series Key Generator software description

Introduction

The STM32MP1 Series Key Generator software (named STM32MP1-KeyGen in this document) is integrated in the STM32CubeProgrammer (STM32CubeProg).

STM32MP1-KeyGen is a tool that generates the ECC keys pair needed for signing binary images. The generated keys are used by the STM32 Signing tool for signing process.

STM32MP1-KeyGen generates a public key file, a private key file and a hash public key file.

The public key file contains the generated ECC public key in PEM format.

The private key file contains the encrypted ECC private key in PEM format. The encryption can be done using the aes 128 cbc or aes 256 cbc ciphers. The cipher selection is done using the --prvkey-enc option.

The hash public key file contains the SHA-256 hash of the public key in binary format. The SHA-256 hash is calculated based on the public key without any encoding format. The first byte of the public key is present just to indicate whether the public key is in compressed or uncompressed format. Since only uncompressed format is supported, this byte is removed.

1 Install STM32MP1-KeyGen

This tool is installed with the STM32CubeProgrammer package (STM32CubeProg). For more information about the set-up procedure, refer to the section 1.2 of the user manual *STM32CubeProgrammer software description* (UM2237).

This software applies to the STM32MP1 Series Arm®-based MPUs.

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

2 STM32MP1-KeyGen command line interface

The following sections describe how to use STM32MP1-KeyGen from command line.

2.1 Commands

The available commands are listed below:

- `--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 path (.pem extension)
  - Syntax: `-pubk <public_key_file_path>`
  - Example: `-pubk C:\publicKey.pem`
- `--public-key-hash (-hash)`
  - Description: Hash image file path (.bin extension)
  - Syntax: `-hash <hash_file_path>`
- `--absolute-path (-abs)`
  - Description: Absolute path for output files
  - Syntax: `-abs <absolue_path_folder_path>`
  - Example: `-abs C:\KeyFolder\`
- `--password (-pwd)`
  - Description: Password of the private key (this password must contain at least four characters)
  - Example: `-pwd azerty`
- `--prvkey-enc (-pe)`
  - Description: Encrypting private key algorithm (aes128/aes256) (aes256 algorithm is the default algorithm)
  - Syntax: `-pe aes128`
- `--ecc-algo (-ecc)`
  - Description: ECC algorithm for keys generation (prime256v1/brainpoolP256t1) (prime256v1 is the default algorithm)
  - Syntax: `-ecc prime256v1`
- `--help (-h and -?)`
  - Description: Shows help.
- `--version (-v)`
  - Description: Displays the tool version.

2.2 Examples

The following examples show how to use STM32MP1-KeyGen:

- **Example 1**

```
-abs /home/user/KeyFolder/ -pwd azerty
```

All files (publicKey.pem, privateKey.pem and publicKeyhash.bin) are created in the */home/user/KeyFolder/* folder. The private key is encrypted with the aes256 default algorithm.

- **Example 2**

```
-abs /home/user/keyFolder/ -pwd azerty -pe aes128
```

All files (publicKey.pem, privateKey.pem and publicKeyhash.bin) are created in the */home/user/KeyFolder/* folder. The private key is encrypted with the aes128 algorithm.

- **Example 3**

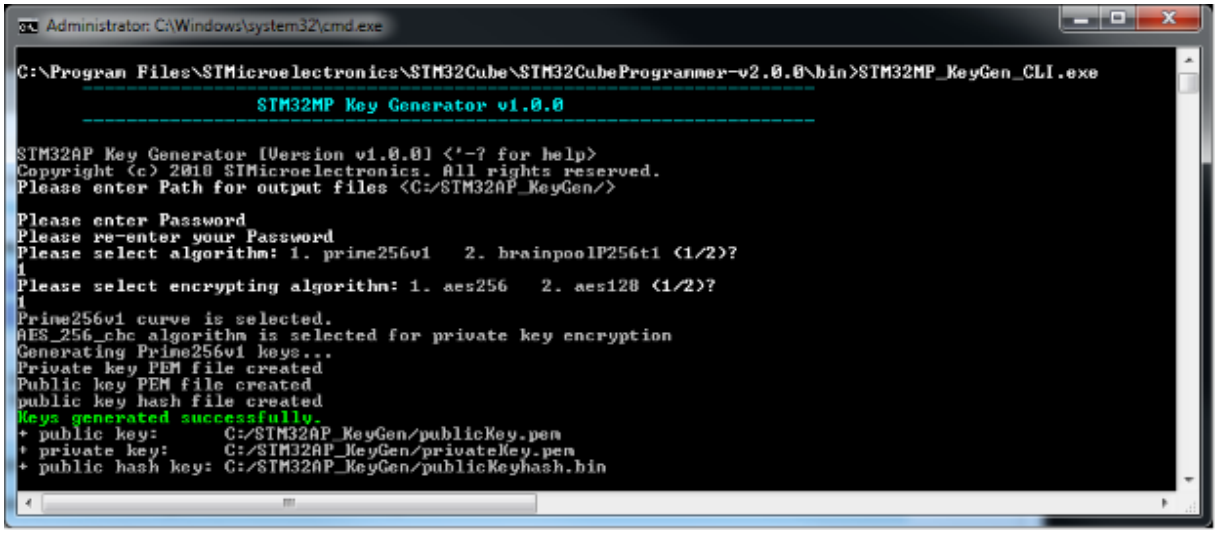
```
-pubk /home/user/public.pem -prvk /home/user/Folder1/Folder2/private.pem -hash /home/use
r/pubKeyHash.bin -pwd azerty
```

Even if the Folder1 and Folder2 does not exist, they are created.

2.3 Standalone mode

When executing STM32MP1-KeyGen in Standalone mode, an absolute path and a password are requested as shown in the figure below.

Figure 1. STM32MP-KeyGen in Standalone mode



When the user press <Enter>, the files are generated in the *<C:\Users\User_Name\.STM32AP_KeyGen/>* folder.

Then enter the password twice and select one of the two algorithms (prime256v1 or brainpoolP256t1) by pressing the respective key (1 or 2).

Finally select an encrypting algorithm (aes256 or aes128) by pressing the respective key (1 or 2).

Revision history

Table 1. **Document revision history**

Date	Version	Changes
14-Feb-2019	1	Initial release.

Contents