
Ecsecure User Guide

EigenCOMM Wireless Microcontroller

Document Description

This document mainly introduces the features of Ecsecure and how it runs in GccBuild.bat.

Contents

1.	General Description.....	3
1.1	Generate key pair.....	3
1.2	Generate header Bin	3
1.3	How to Support third-party BIN signatures.....	4
1.4	Sign Bootloader and System image with the same key	5
1.5	Sign Agentboot	5
1.6	The same key used to sign Bootloader1, Bootloader2 and System image	5
1.7	Generate key pair for FOTA package	5
1.8	Sign the FOTA package	5
1.9	Make public key from Bootloader's header	6
1.10	Use public key to verify the signature in the System header.....	6
2.	Version.....	6

1. General Description

Ecsecure is a tool for generate ECDSA key pair, header Bin of Bootloader and System image for Secure Boot. It also can generate customized header Bin for customers' extended Bin.

There are two versions of Ecsecure that can run in Windows and Ubuntu, the executable files are Ecsecure.exe and ecsecure respectively.

For the most common design, it mainly runs in the Gccbuild.bat (Windows environment) or the build.sh (Linux environment) of SDK. Before compiling and generating the bootloader, it generates the ECDSA key pair for the bootloader and system image (AP+CP). After the system image is compiled, it creates the security header for the bootloader and system image. For details, please refer to AN0015-EC CAT1 Secure Boot.docx.

In order to meet the needs of different customers, Ecsecure features are constantly expanding.

The following sections will introduce the features of this tool and how to use it. *Italic font indicates that it will change according to the actual situation.*

1.1 Generate key pair

The command lines for generating two pairs of the ECDSA key pairs for bootloader and system image is as follows:

Windows:

```
ecsecure.exe PUBKEY=pubkey.c BLPEN=pub_key_bl.pem SYSPEN=pub_key_sys.pem
```

Ubuntu:

```
./ecsecure PUBKEY=pubkey.c BLPEN=pub_key_bl.pem SYSPEN=pub_key_sys.pem
```

The command lines for generating a pair of the ECDSA key pair for bootloader and system image is as follows:

```
ecsecure.exe PUBKEY=pubkey.c BLPEN=pub_key_bl.pem
```

The default is to generate 224bit public key (for EC618/QCX216).

For compressed 256 bit bootloader public key (for new chip) should add parameter: CURV_256_COMP=1.

The PUBKEY is needed by SDK, it will generate the source code compiled into the bootloader.

1.2 Generate header Bin

The command lines for generate the header Bins for bootloader and system image is as follows:

Windows:

```
ecsecure.exe APIMAGE=ap_at_command.bin CPIMAGE=cp-demo-flash.bin BLIMAGE=ap_bootloader.bin  
HEAD1=bl_sec_header.bin HEAD2=sys_sec_header.bin BLPEN=pub_key_bl.pem  
SYSPEN=pub_key_sys.pem SYSBNADR=7e000
```

Linux:

```
./ecsecure APIMAGE=ap_at_command.bin CPIMAGE=cp-demo-flash.bin BLIMAGE=ap_bootloader.bin
HEAD1=bl_sec_header.bin HEAD2=sys_sec_header.bin BLPEM=pub_key_bl.pem
SYSPERM=pub_key_sys.pem SYSBNADR=7e000
```

For compressed 256 bit bootloader public key (for new chip) should add parameter: CURV_256_COMP=1.

If you need to sign third-party bin files, you can specify the JSON configuration file, such as:

CFGJSON=custom.json. if use this parameter the above parameters cannot be defaulted. Please refer to section 1.3 for specific instructions.

1.3 How to Support third-party BIN signatures

If the customer does not have additional BIN files that require security verification, the current section can be ignored. The JSON format configuration can help customers configure BIN files that require additional security verification, sign the BIN file with system private key, and generate the corresponding header file.

The following is an example of a JSON configuration file. “AdditionalBin” corresponds to a set of BIN file, currently supporting a maximum of ten BIN files. “binfile” is additional BIN file’s path and filename. “binheader” is the name of the security header of the BIN file. “identifier”, “imageId”, “imageAddr” is 32-bit digit in decimal representation of BIN file, which can be defaulted if not need set. They are parameters that the flash tool may need to use, as additional bin is still being designed, so the settings for these parameters are reserved.

```
{
  "AdditionalBin": [
    {
      "binfile": "custom_example1.bin",
      "binheader": "custom_sec_header1.bin",
      "identifier": xxxxxx,
      "imageId": yyyyyyy,
      "imageAddr": zzzzzzz
    },
    {
      "binfile": "custom_example2.bin",
      "binheader": "custom_sec_header2.bin",
      "identifier": xxxxxx,
      "imageId": yyyyyyy,
      "imageAddr": zzzzzzz
    }
  ]
}
```

The following is a command line example:

```
ecsecure.exe APIMAGE=ap_at_command.bin CPIMAGE=cp-demo-flash.bin
BLIMAGE=ap_bootloader.bin HEAD1=bl_sec_header.bin HEAD2=sys_sec_header.bin
BLPEM=pub_key_bl.pem SYSPEM=pub_key_sys.pem CFGJSON=custom.json
```

1.4 Sign Bootloader and System image with the same key

This feature is for a customer who wished to sign the Bootloader and system using the same set of keys, and the public key stored in Bootloader's header. If there is no key input, generate it and sign Bootloader and System image, if there is key input, use it to sign Bootloader and System image.

```
ecsecure.exe APIMAGE=ap_at_command.bin CPIMAGE=cp-demo-flash.bin BLIMAGE=ap_bootloader.bin
HEAD1=bl_sec_header.bin HEAD2=sys_sec_header.bin UNIPEM=uni_key_pair.pem
ADRBASE=AP_FLASH_XIP_ADDR APADR=AP_FLASH_LOAD_ADDR CFGDEF=mem_map_718p.txt
```

1.5 Sign Agentboot

```
ecsecure.exe AGENTBT=agentboot.bin HEAD1=agentbt_header.bin BLPEM=pub_key_bl.pem
```

1.6 The same key used to sign Bootloader1, Bootloader2 and System image

This feature is use same set of keys to sign the Bootloader1, Bootloader2 and System image, generating their respective headers. BL2BNADR set the Bootloader2's address in flash.

```
ecsecure.exe APIMAGE=ap_at_command.bin CPIMAGE=cp-demo-flash.bin BLIMAGE=ap_bootloader.bin
BLEXT=ap_bootloader2.bin HEAD1=bl_sec_header.bin HEAD2=sys_sec_header.bin
HEAD3=bl2_sec_header.bin BLPEM=uni_key_pair.pem BL2BNADR=c000 SYSBNADR=24000
```

1.7 Generate key pair for FOTA package

This feature is to generate the ECDSA key pair for verifying FOTA package. The command line is embedded in the Makefile of SDK, and the generated public key will be compiled into Bootloader2 and System image. The default is secp256r1 keys, if there is CODE=1, it is secp224r1 keys.

```
ecsecure.exe FOTAKEY=fota_key.c BLPEM=fota_keys.pem
ecsecure.exe FOTAKEY=fota_key.c BLPEM=fota_keys.pem CODE=1
```

1.8 Sign the FOTA package

This feature is used in the FOTA tool to generate an authentication header containing a signature for the FOTA package. default is secp256r1 keys, if there is CODE=1, it is secp224r1 keys.

```
ecsecure.exe FOTASIGN=sign_sys.bin FOTAHASH=hash_sys.bin SYSPEM=fota_keys.pem
ecsecure.exe FOTASIGN=sign_sys.bin FOTAHASH=hash_sys.bin SYSPEM=fota_keys.pem CODE=1
```

1.9 Make public key from Bootloader's header

This feature is only applicable to the case where the public key is stored in the header of the Bootloader, as described in section 1.4

```
ecsecure.exe PUBKEY=publickey.bin HEAD1=bl_sec_header.yy.bin
```

1.10 Use public key to verify the signature in the System header

This feature is used by the FOTA tool to verify the validity of the signature in the System header before upgrading, in order to avoid the UE boot failure due to invalid signature. Its output file has three results: success(verification success), failure(verification failure) and keyInvalid(key invalid).

```
ecsecure.exe PUBKEY=publickey.bin HEAD2=sys_sec_header.bin VRESULT=result.txt
```

2. Version

Version	Date	Author	Comments
1.0	2023-8-24	Yuanjing.Zhong	Initiated
2.0	2024-05-30	Yuanjing.Zhong	Add 1.4-1.9 section
2.1	2024-09-24	Yuanjing.Zhong	Delete the original section 1.4, add 1.9 2.0 section
2.1	2024-11-08	Yuanjing.Zhong	Update 1.7 1.8
2.1	2025-02-26	Yuanjing Zhong	Update 1.1, remove BLHASH=1 SYSHASH, Delete 1.9 2.0
2.2	2025-08-09	Yuanjing Zhong	Add 1.9 1.10