Rockchip Anti Copy Board Developer's Guide

ID：RK-KF-YF-877

Release Version：V1.0.0

Release Date：2023-12-01

Security Level: □Top-Secret □Secret □Internal ■Public

Rockchip Electronics Co., Ltd.

No.18 Building, A District, No.89, software Boulevard Fuzhou, Fujian,PRC

Website: www.rock-chips.com

Customer service Tel: +86-4007-700-590

Customer service Fax: +86-591-83951833

Customer service e-Mail: [fae@rock-chips.com]

Preface

Summary

This document mainly introduces the principle and usage steps of Rockchip anti copying board technology.

Readers

This document is mainly applicable to the following engineers:

Technical Support Engineer

Software Development Engineer

History

Revision	**Author**	**Date**	**Description**
V1.0.0	hisping	2023-12-01	Original document

Contents

1. Summary

Rockchip provides anti copying board technology to protect customers' firmware, private data, and core code.

Anti copying board technology is mainly used to prevent customers' firmware and private data from being illegally copied and used by unauthorized users, and to avoid commercial losses caused by copying.

This document introduces three solutions for anti copying board technology:

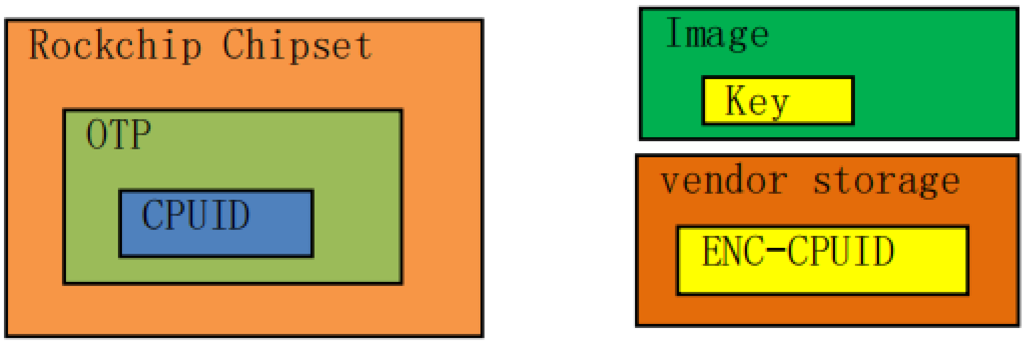
- Low-level solution
- Mid-level solution
- High-level solution

Customers are free to choose one of the options to use.

2. Low-level solution

2.1 Principle

During production, Rockchip chips will burn CPUID to OTP, and each chip has a unique CPUID that customers can read, Encrypt using symmetric key to obtain ENC-CPUID, Customers can store ENC-CPUID in the vendor storage partition, After system startup, read ENC-CPUID from the vendor storage partition and decrypt it to obtain DEC-CPUID. Compare DEC-CPUID and CPUID to see if they match, If the matching fails, It is considered illegal and the device is restarted to achieve the purpose of preventing board copying.



The security of this solution depends on the protection of the key, and customers should avoid directly fix the plaintext key in the code, suggest to obfuscate the key before fix it into the code, prevent illegal users from obtaining plaintext keys through disassembly.

In addition, ENC-CPUID is stored in the vendor storage partition, erasing the flash will clear the vendor storage partition, resulting in the loss of ENC-CPUID. Therefore, after erasing the flash, it is necessary to rewrite the ENC-CPUID.

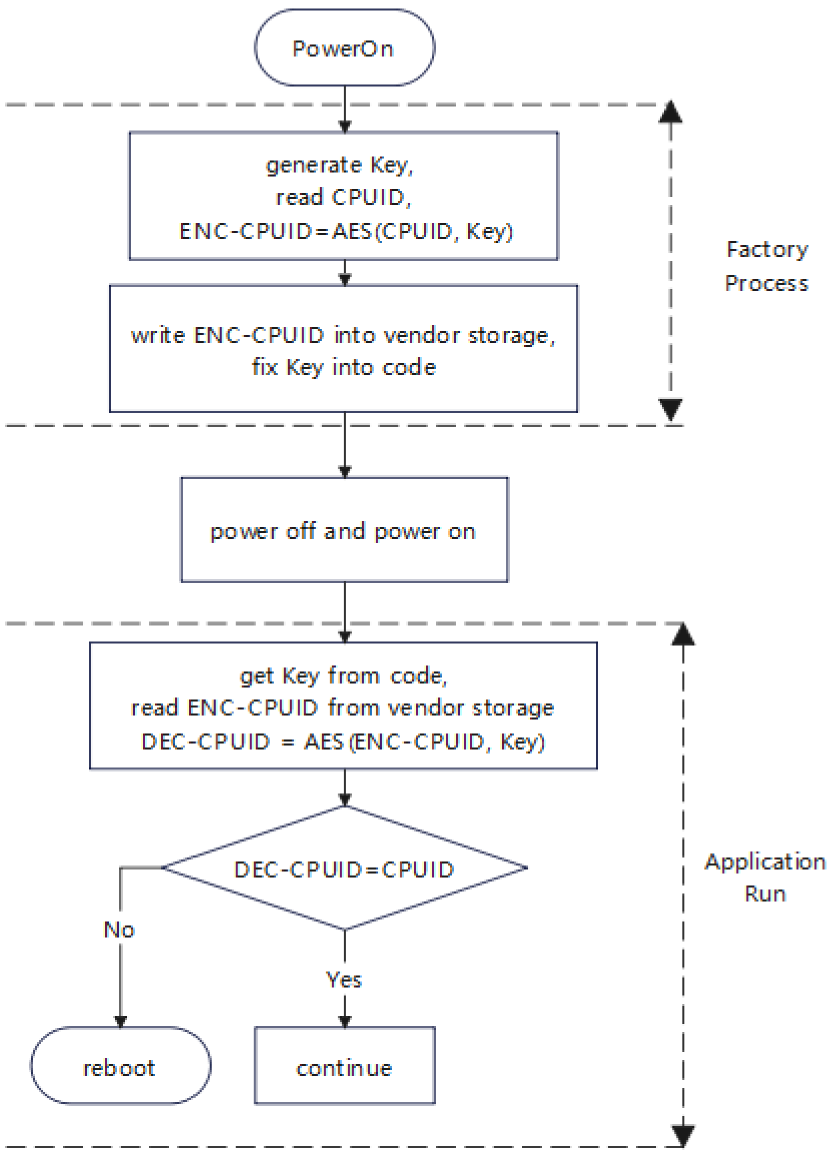
2.2 Usage steps

The application steps of the Low-level solution are as follows:

1. The customer generates a custom private symmetric key, reads the CPUID from the OTP, and encrypts it using the symmetric key (AES algorithm) to obtain the ENC-CPUID.
2. Write ENC-CPUID to the vendor storage partition, obfuscate the key to increase security, and fix the obfuscated key in the application code.
3. After the app runs, it reads the ENC-CPUID from the vendor storage partition, decrypts the ENC-CPUID using the key, and obtains the DEC-CPUID.
4. Compare DEC-CPUID and CPUID to see if they match, and the app will choose to run normally or restart immediately based on the matching results.

2.3 Flow chart

The complete flowchart of the Low-level solution is as follows:



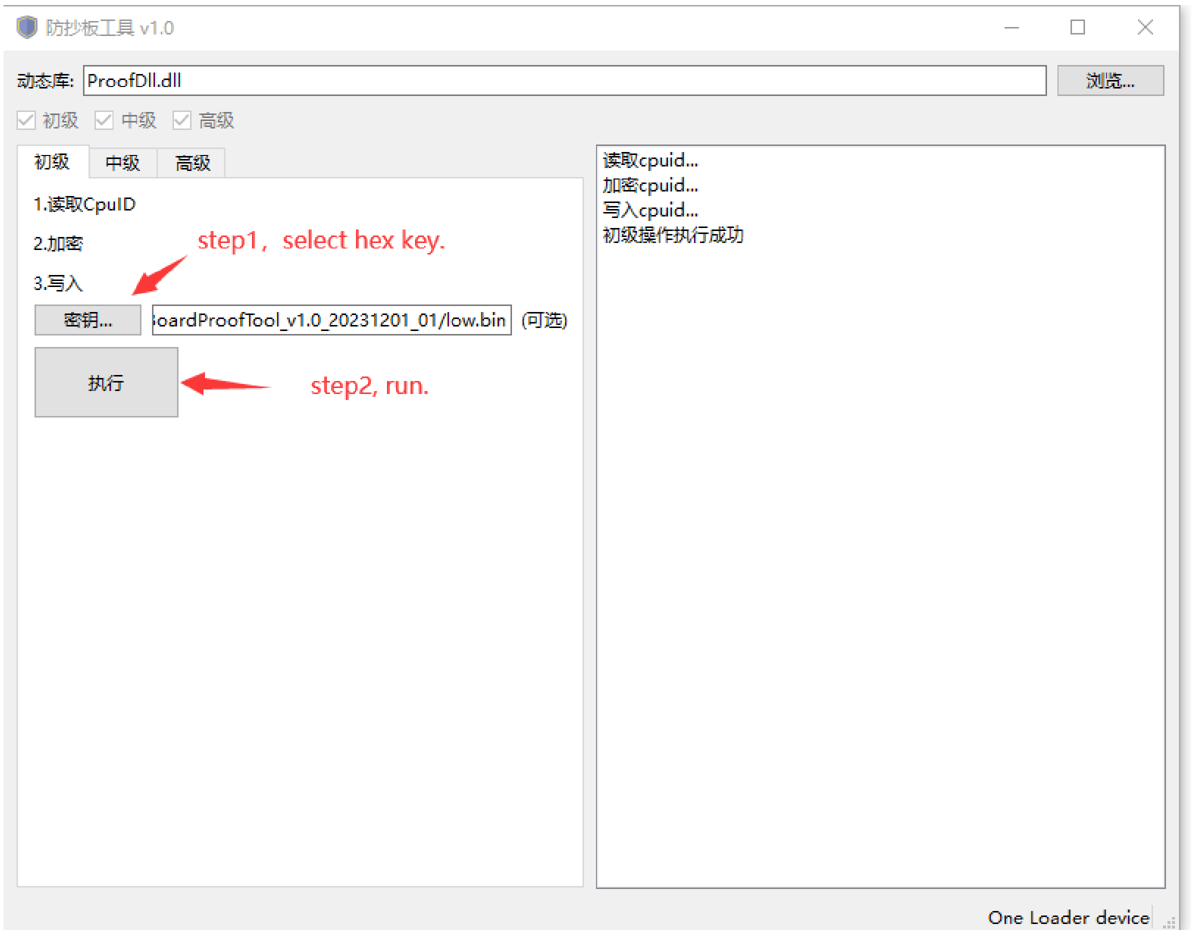
2.4 Example

1. Write ENC-CPUID using the BoardProofTool tool.

Press and hold the recovery key when the device is turned on to enter the Loader state.

The customer create a low. bin file, then edits it in hexadecimal format and writes a 32 byte key.

Open the BoardProofTool.exe tool, click on "密钥" to select the low. bin file, then click "执行" to complete the process.



动态库: ProofDll.dll

☑ 初级 ☑ 中级 ☑ 高级

密钥... :oardProofTool_v1.0_20231201_01/low.bin (可选)

```
读取cpuid...
加密cpuid...
写入cpuid...
初级操作执行成功
```

BoardProofTool tool calls rkusb_do_read_otp function in U-Boot to read CPUID, The tool automatically encrypts CPUID with a key to obtain ENC-CPUID, The tool calls vendor_storage_write function in U-Boot to write ENC-CPUID.

2. Write ENC-CPUID using the rk_anti_copy_board.

The source code of rk_anti_copy_board is located in the rk_tee_user/v2/host/rk_anti_copy_board directory.

Android compilation command:

```
cd /home1/xxxx/rk_android_13
source build/envsetup.sh
lunch rk3568_t-userdebug
cd /home1/xxxx/rk_android_13/external/rk_tee_user/v2
./build.sh ta
mm

//copy to device
adb push
Y:\rk_android_13\out\target\product\rk3568_t\vendor\bin\rk_anti_copy_board
/vendor/bin
```

Linux compilation command：

```
cd /home1/xxxx/rk_px30_linux/external/security/rk_tee_user/v2
rm -rf out/
./build.sh 6432

//copy to device
adb push
Y:\rk_px30_linux\external\security\rk_tee_user\v2\out\rk_anti_copy_board\rk_anti_
copy_board /usr/bin
```

Test command：

```
root@RK3588:/# rk_anti_copy_board low_gen
generate enc_cpuid success!
```

3. Verifies the legitimacy of the device.

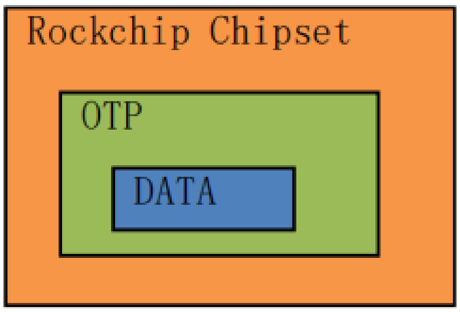
Test command：

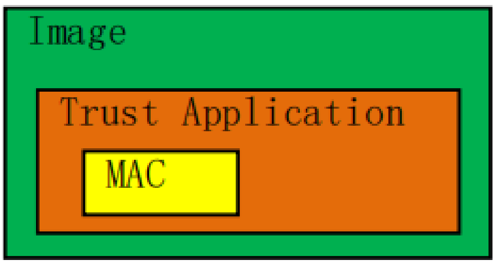
```
root@RK3588:/# rk_anti_copy_board low_verify
verify success!
```

3. Mid-level solution

3.1 Principle

The Rockchip chip has reserved space in OTP for customers to store anti copying board private data，Customer generated private data DATA and MAC, customer customized specific function func, Where DATA and MAC satisfy specific function conversion relationships, such as MAC=func (DATA), Customers can write DATA to the OTP and solidify the MAC in the TA (Trust Application) code. After the system starts, the TA is called to read the private data DATA from the OTP. The TA judges whether there is a given function conversion relationship between DATA and MAC. If the matching fails, it is considered illegal. Restart the device to achieve the purpose of anti copying board.





The security of this solution depends on the protection of the private data DATA, which cannot be read by the kernel, Only TA (Trust Application) can read private data DATA. Customers should refer to the "TA Signature" section in the document "Rockchip_Developer_Guide_TEE_SDK_EN.md", use your own key to sign the TA (Trust Application) to avoid unauthorized TA run and prevent the leakage of private data DATA.

3.2 Usage steps

The usage steps of the Mid-level solution are as follows:

1. Customer generated custom private data DATA， Select a specific conversion function， Calculate MAC=func (DATA).
2. Write the private data DATA into OTP, develop a trusted application TA, and solidify the MAC in TA code.
3. After the APP runs, it calls the trusted application TA, which reads the private data DATA from the OTP. The trusted application TA calculates whether the DATA matches the MAC and returns the matching result.
4. The APP selects normal operation or immediate restart based on the matching results.

3.3 Flow chart

The flowchart of the Mid-level solution is as follows：

3.4 Example

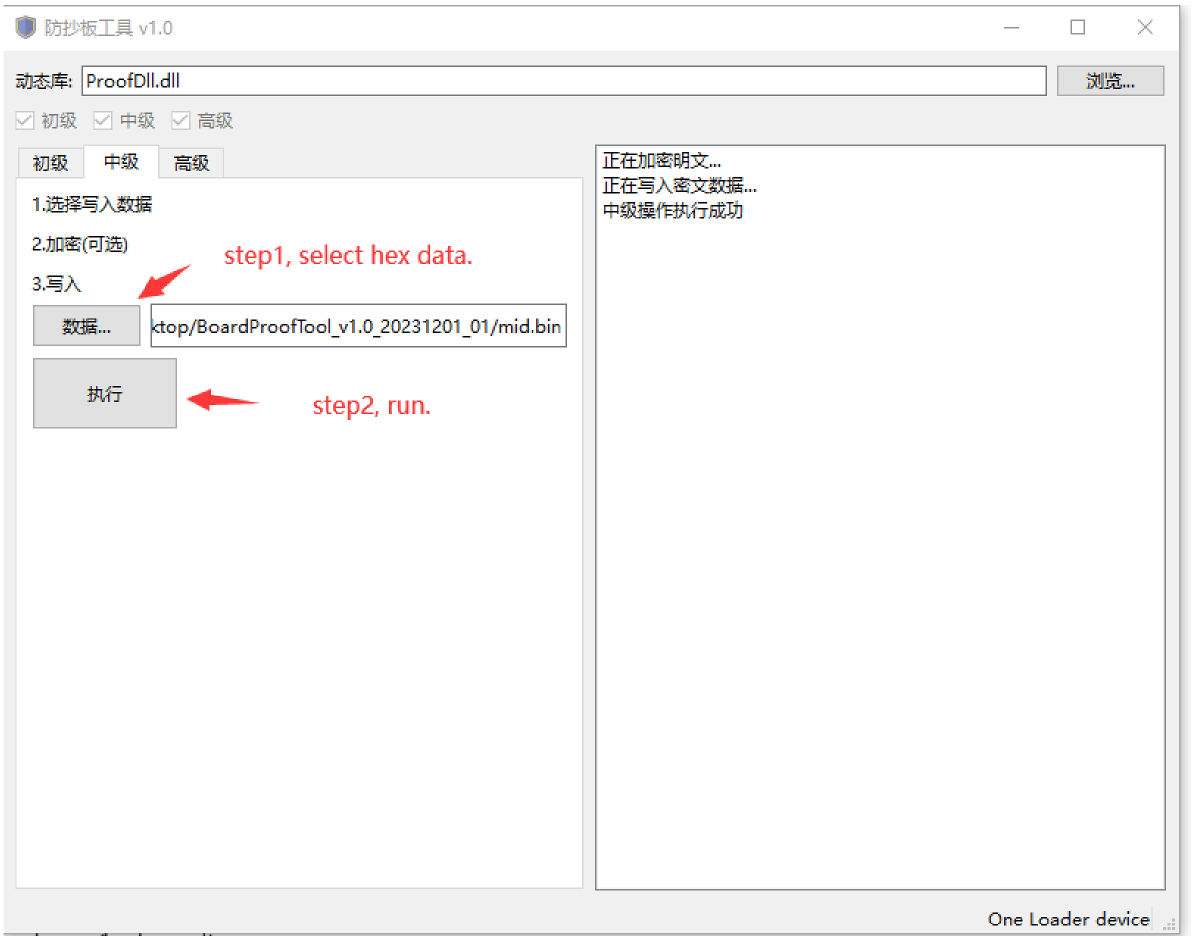
1. Write custom private data using the BoardProofTool tool.

Press and hold the recovery key when the device is turned on to enter the Loader state.

The customer create a mid. bin file, then edits it in hexadecimal format and writes a 16 byte custom private data.

Open the BoardProofTool.exe tool, click on "数据" to select the mid. bin file, then click "执行" to complete the process.

(Note: OTP is a one-time programmable storage, Each device can only write private data once.)



正在加密明文...
正在写入密文数据...
中级操作执行成功

2. Write custom private data using the rk_anti_copy_board.

The source code of rk_anti_copy_board is located in the rk_tee_user/v2/host/rk_anti_copy_board directory.

Android compilation command:

```
cd /home1/xxxx/rk_android_13
source build/envsetup.sh
lunch rk3568_t-userdebug
cd /home1/xxxx/rk_android_13/external/rk_tee_user/v2
./build.sh ta
mm

//copy to device
adb push Y:\rk_android_13\hardware\rockchip\optee\v2\arm64\libteec.so
/vendor/lib64
adb push Y:\rk_android_13\hardware\rockchip\optee\v2\arm64\tee-supplicant
/vendor/bin
adb push
Y:\rk_android_13\external\rk_tee_user\v2\out\ta\rk_anti_copy_board\3d4fc699-2065-
7bb9-33c7-b6529b43c91a.ta /vendor/lib/optee_armtz
adb push
Y:\rk_android_13\out\target\product\rk3568_t\vendor\bin\rk_anti_copy_board
/vendor/bin
```

Linux compilation command:

```
cd /home1/xxxx/rk_px30_linux/external/security/rk_tee_user/v2
rm -rf out/
./build.sh 6432

//copy to device
adb push Y:\rk_px30_linux\external\security\bin\optee_v2\lib\arm64\libteec.so.1
/lib64
adb push Y:\rk_px30_linux\external\security\bin\optee_v2\lib\arm64\tee-supplicant
/usr/bin
adb push
Y:\rk_px30_linux\external\security\rk_tee_user\v2\out\ta\rk_anti_copy_board\3d4fc
699-2065-7bb9-33c7-b6529b43c91a.ta /lib/optee_armtz
adb push
Y:\rk_px30_linux\external\security\rk_tee_user\v2\out\rk_anti_copy_board\rk_anti_
copy_board /usr/bin
```

Test command：

```
root@RK3588:/# rk_anti_copy_board mid_gen
save data into OTP success!
```

3. Verifies the legitimacy of the device.

Test command：

```
root@RK3588:/# tee-supplicant &
root@RK3588:/# rk_anti_copy_board mid_verify
I/TA: Hello!
I/TA: verify success!
I/TA: Goodbye!
```

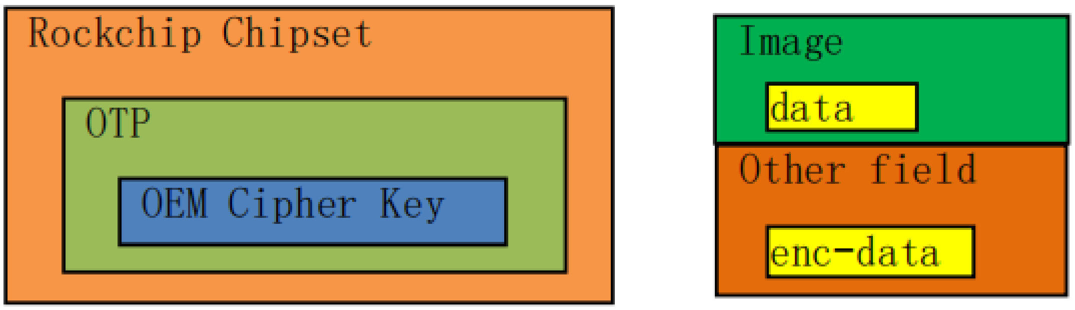
4. High-level solution

4.1 Principle

Rockchip chip has reserved space in OTP for customers to store custom private keys， For usage methods, please refer to the "OEM Cipher Key" section in the《Rockchip_Developer_Guide_OTP_CN.md》 document.

Rockchip anti copying board technology relies on user private key OEM Cipher Key stored in OTP， This key is written during the factory process， To ensure that the key is not leaked, the system only provides a burn write interface and no read interface, Some platforms support Hardware Read for locking keys, keys cannot be read by the CPU after the key is locked, resulting in higher security.

Customers can customize their own plaintext data, use a private key OEM Cipher Key, and encrypt the data using AES algorithm， Obtain enc-data and store it in flash or fix in code， Setting plaintext data in application code. After system startup， use the private key OEM Cipher Key to decrypt the enc-data and obtain the dec-data， Compare the dec-data with the plaintext data in the application code. If the plaintext data or enc-data is tampered， Or the flash firmware is illegally copied to other unauthorized chip platforms， Then there will be a mismatch between dec-data and plaintext data， It is considered illegal and restart device if match fails, In order to achieve the purpose of preventing board copying.

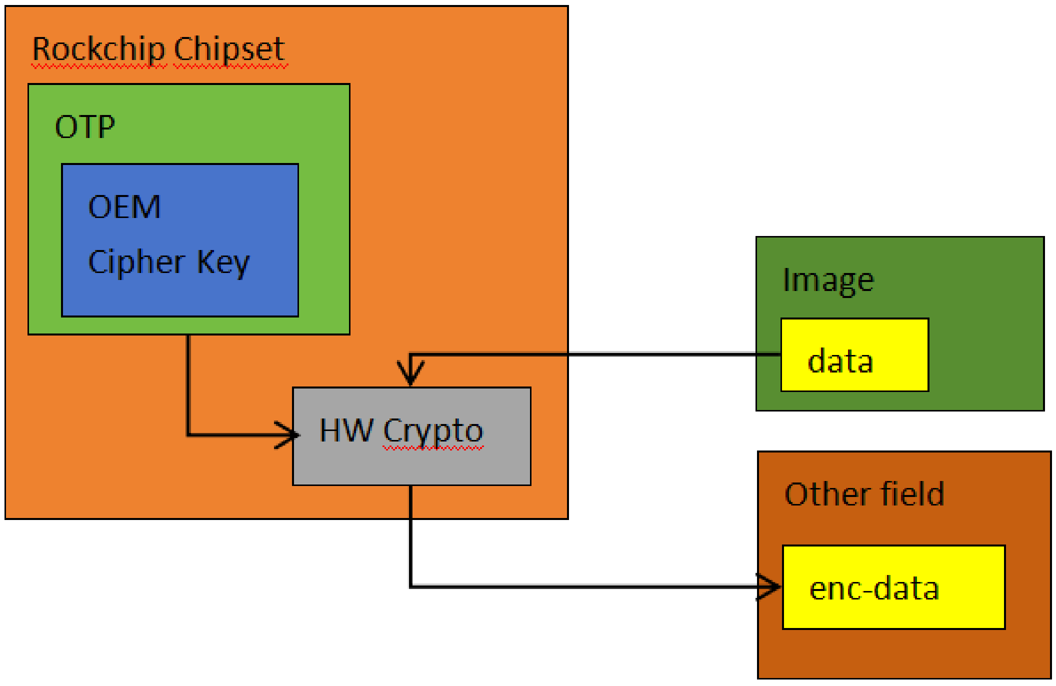


Advantages of this solution: After the user's private key OEM Cipher Key is locked, it cannot be read by the CPU, and has higher security features such as confidentiality, integrity, and tamper resistance.

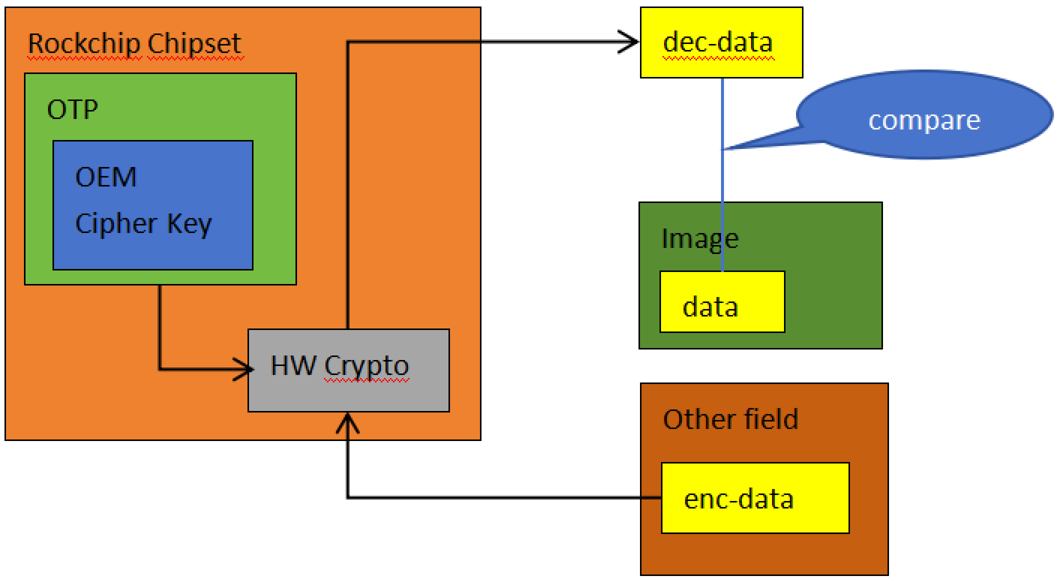
4.2 Usage steps

The usage steps of the High-level solution are as follows:

1. The customer generates a custom private key OEM Cipher Key, writes the OEM Cipher Key to the OTP, and locks the key.
2. Set plaintext data in the APP code.
3. Encrypt plaintext data using a private key OEM Cipher Key in the APP code， Obtain enc-data and store it in flash or fix in code.

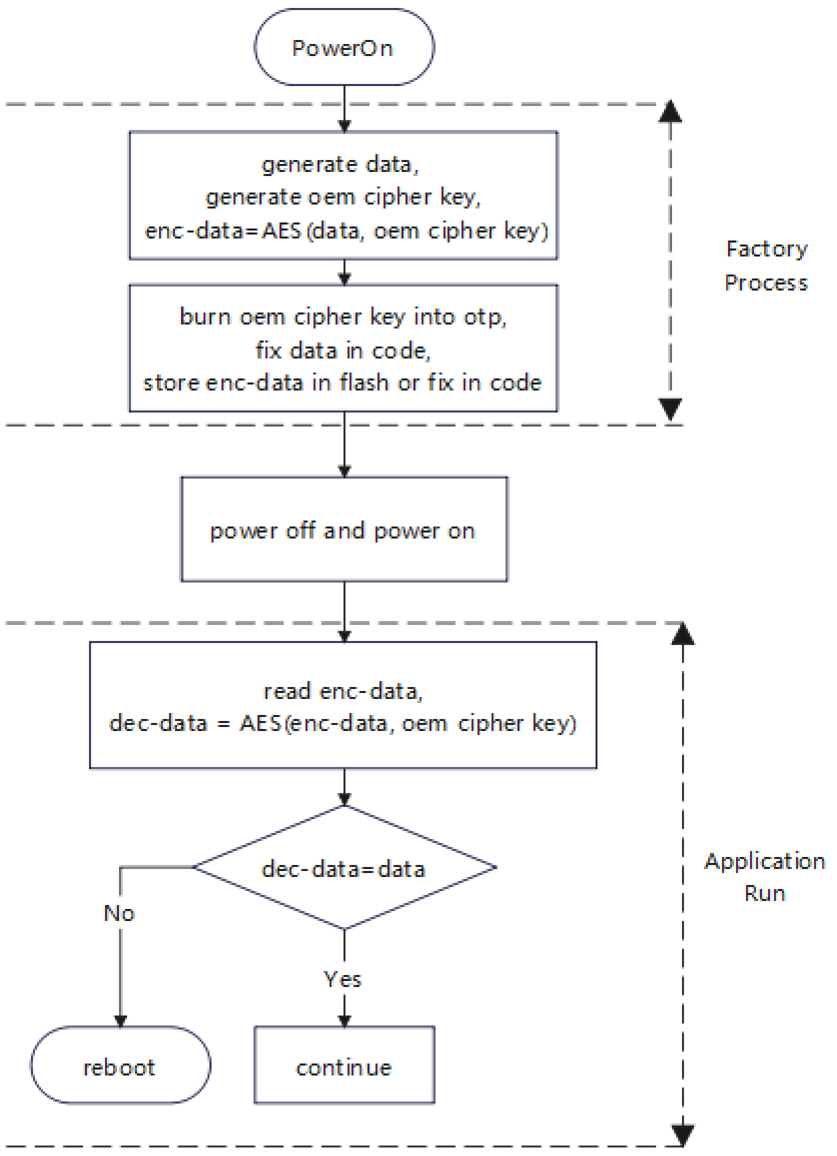


4. After system startup， APP uses private key OEM Cipher Key to decrypt enc-data and obtaining dec-data，Compare the dec-data with the plaintext data in the application code. Select normal operation or immediate restart based on the matching results.



4.3 Flow chart

The flowchart of the High-level solution is as follows:



4.4 Example

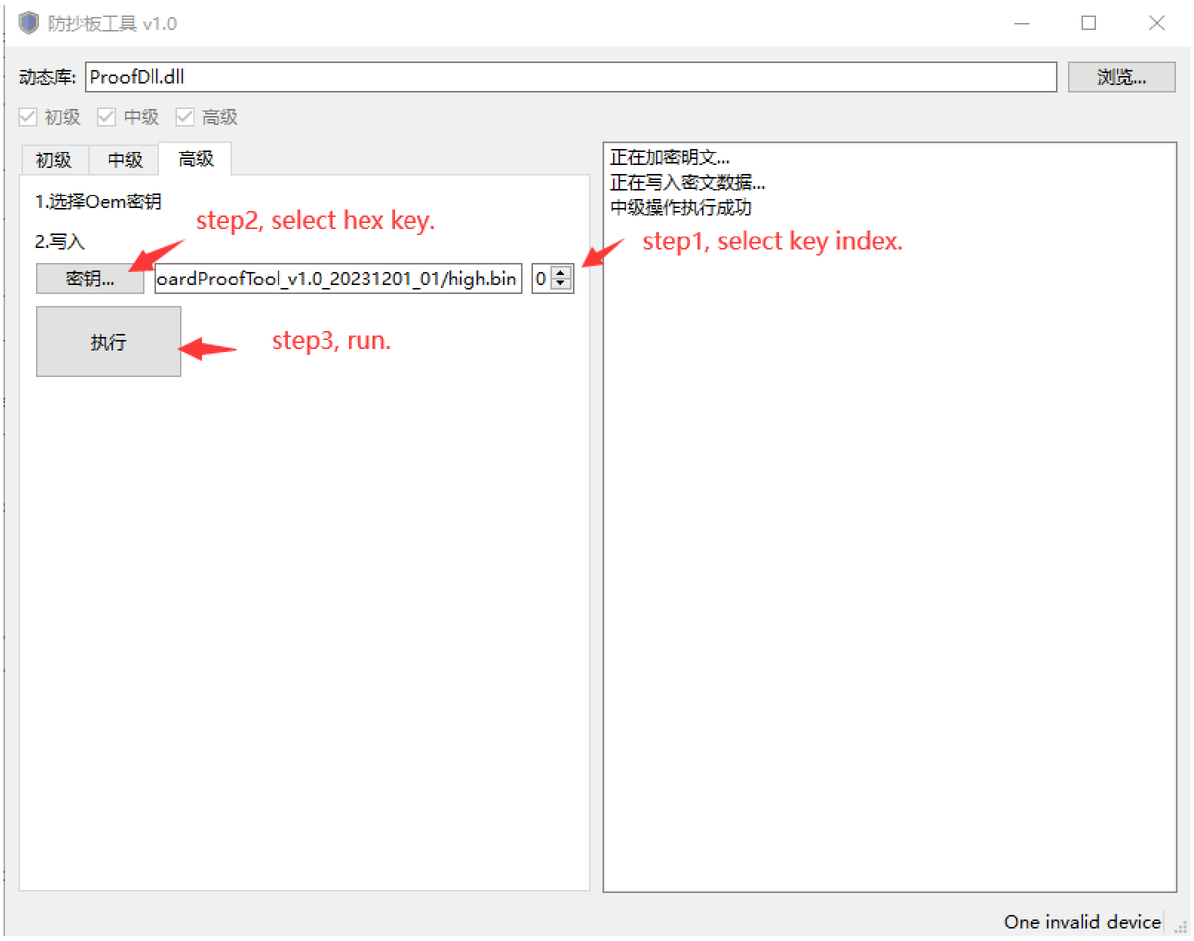
1. Write OEM Cipher Key using the BoardProofTool tool.

Press and hold the recovery key when the device is turned on to enter the Loader state.

The customer create a high.bin file, then edits it in hexadecimal format and writes a 16 byte OEM Cipher Key (Also supports 24 or 32 byte lengths).

Open the BoardProofTool.exe tool, select key index, click on "密钥" to select the high.bin file, then click "执行" to complete the process.

(Note: OTP is a one-time programmable storage, Each device can only write OEM Cipher Key once on same key index.)



```
正在加密明文...
正在写入密文数据...
中级操作执行成功
```

2. Write OEM Cipher Key using the rk_anti_copy_board.

The source code of rk_anti_copy_board is located in the rk_tee_user/v2/host/rk_anti_copy_board directory.

The compilation method is consistent with the Low-level solution, and will not be elaborated here.

Test command:

```
root@RK3588:/# rk_anti_copy_board high_gen
```

3. Verifies the legitimacy of the device.

Test command:

```
root@RK3588:/# rk_anti_copy_board high_verify
```

4.5 Solution extension

Customers can extend the usage of the private key OEM Cipher Key.

For example, using OEM Cipher Key to encrypt and store critical data, and decrypting critical data when needed after system startup.

OEM Cipher Key can also be used to encrypt the core code. After system startup, when the core code needs to be run, it should be decrypted before execution.

5. Extended Security

Customers can combine anti copying board technology with the following security technologies to further reduce the risk of being copied and improve system security.

- Secure Boot. Verify the legality of the firmware during startup, Ensure that the running firmware has not been tampered with, Prevent illegal program operation.
- Firmware Encryption. Encrypt and then burn the firmware, Decrypt and run only at startup, Prevent unauthorized users from obtaining plaintext firmware on flash and prevent firmware from being disassembled for analysis.