XRADIO Security Boot Developer Guide

Revision 1.0

Nov 21, 2019

Declaration

THIS DOCUMENTATION IS THE ORIGINAL WORK AND COPYRIGHTED PROPERTY OF XRADIO TECHNOLOGY ("XRADIO"). REPRODUCTION IN WHOLE OR IN PART MUST OBTAIN THE WRITTEN APPROVAL OF XRADIO AND GIVE CLEAR ACKNOWLEDGEMENT TO THE COPYRIGHT OWNER.

THE PURCHASED PRODUCTS, SERVICES AND FEATURES ARE STIPULATED BY THE CONTRACT MADE BETWEEN XRADIO AND THE CUSTOMER. PLEASE READ THE TERMS AND CONDITIONS OF THE CONTRACT AND RELEVANT INSTRUCTIONS CAREFULLY BEFORE USING, AND FOLLOW THE INSTRUCTIONS IN THIS DOCUMENTATION STRICTLY. XRADIO ASSUMES NO RESPONSIBILITY FOR THE CONSEQUENCES OF IMPROPER USE (INCLUDING BUT NOT LIMITED TO OVERVOLTAGE, OVERCLOCK, OR EXCESSIVE TEMPERATURE).

THE INFORMATION FURNISHED BY XRADIO IS PROVIDED JUST AS A REFERENCE OR TYPICAL APPLICATIONS, ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS DOCUMENT DO NOT CONSTITUTE A WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. XRADIO RESERVES THE RIGHT TO MAKE CHANGES IN CIRCUIT DESIGN AND/OR SPECIFICATIONS AT ANY TIME WITHOUT NOTICE.

NOR FOR ANY INFRINGEMENTS OF PATENTS OR OTHER RIGHTS OF THE THIRD PARTIES WHICH MAY RESULT FROM ITS USE. NO LICENSE IS GRANTED BY IMPLICATION OR OTHERWISE UNDER ANY PATENT OR PATENT RIGHTS OF XRADIO.THIRD PARTY LICENCES MAY BE REQUIRED TO IMPLEMENT THE SOLUTION/PRODUCT. CUSTOMERS SHALL BE SOLELY RESPONSIBLE TO OBTAIN ALL APPROPRIATELY REQUIRED THIRD PARTY LICENCES. XRADIO SHALL NOT BE LIABLE FOR ANY LICENCE FEE OR ROYALTY DUE IN RESPECT OF ANY REQUIRED THIRD PARTY LICENCE. XRADIO SHALL HAVE NO WARRANTY, INDEMNITY OR OTHER OBLIGATIONS WITH RESPECT TO MATTERS COVERED UNDER ANY REQUIRED THIRD PARTY LICENCE.

Revision History

Version	Date	Summary of Changes
1.0	2019-11-21	Initial Version

Table 1- 1 Revision History

Contents

Figures

1 概述

1.1 Security boot 简述

Security boot，即安全启动，是软硬件相结合的安全保护机制，只有合法固件才可以运行。所有合法固件均需要经过唯一的密钥签名，否则，都是非法固件。

Security boot 需要 Bootloader 的支持。支持安全启动和不支持安全启动的 Bootloader 不能相互替换，即：

- 安全启动方案，必须使用支持安全启动的 Bootloader
- 非安全启动方案，必须使用不支持安全启动的 Bootloader

2 使用说明

2.1 配置密钥文件

密钥文件的生成工具位于“ca”目录内。“ca”目录下的文件说明如下：

1）openssl.cnf ：证书信息配置文件，可设置国家、地区、组织等信息。如图 2-1 所示，一般只需要修改标黄的条目。

```
[ req_distinguished_name ]
countryName                 = Country Name (2 letter code)
countryName_default         = CN
countryName_min             = 2
countryName_max             = 2

stateOrProvinceName         = State or Province Name (full name)
stateOrProvinceName_default= GuangDong

localityName                = Locality Name (eg, city)
localityName_default        = Zhuhai

0.organizationName          = Organization Name (eg, company)
0.organizationName_default = your_company

# we can do this but it is not needed normally :-)
#1.organizationName         = Second Organization Name (eg, company)
#1.organizationName_default= World Wide Web Pty Ltd

#organizationalUnitName         = Organizational Unit Name (eg, section)
#organizationalUnitName_default =

commonName              = Common Name (e.g. server FQDN or YOUR name)
commonName_max          = 64

emailAddress            = Email Address
emailAddress_max        = 64
emailAddress_default    = your_email@allwinnertech.com
```

图 2-1 openssl.cnf 文件配置

2）gen_rsa_key.sh ：用于生成密钥文件的脚本。每次运行该脚本都会随机生成一组密钥文件（每次运行生成的密钥文件都是不一样的）。运行该脚本生成的密钥文件说明如下：

- cakey.pem：1024-bit RSA private key
- cakey_pub.pem：“cakey.pem”对应的 1024-bit RSA public key，pem 格式
- cakey_pub.dcr：“cakey.pem”对应的 1024-bit RSA public key， dcr 格式
- cakey_pub_dcr_hash.txt：存储了“cakey_pub.dcr”文件的 SHA256 哈希值，文件内容举例如下：

```
SHA256(cakey_pub.dcr)= 38390faeeceda2e8685680bb579071e7a2d2b3d7357688f9038f6ba23e7890ef
```

其中，蓝色的字符串即为 SHA256 哈希值对应的字符串表示。SHA256 哈希值为 32 字节 16 进制数据，此处转换成 64 个可读字符进行显示。该哈希值将用于的 EFUSE 中 Secure Boot 字段的烧录，烧录方法在 2.4 节进行介绍。

2.2 编译 bootloader

实现安全启动的功能需要编译支持安全启动的 Bootloader，步骤如下：

1）在“project\bootloader\gcc\localconfig.mk”中设置“export __CONFIG_SECURE_BOOT := y”。

2）在“project\bootloader\gcc\”目录下执行“make build”重新编译生产“boot.bin”。

2.3 编译 image

安全启动方案需要对“boot.bin”和“app.bin”进行签名后打包生成 image，编译步骤如下：

1）在“project\<your_project>\gcc\localconfig.mk”中设置“export __CONFIG_SECURE_BOOT := y”。

2）修改工程对应的 image config 文件，向“boot.bin”和“app.bin”添加证书名字，并将“attr”属性修改为“0x5”，如图 2-2 标黄内容所示。

```
{
    "magic"   :"AWIH",
    "version" :"0.3",
    "OTA"     :{"addr": "1024K", "size": "32K"},
    "count"   :8,
    "section" :[
        {"id": "0xa5ff5a00", "bin": "boot.bin",     "cert": "boot.crt", "flash_offs": "0K",    "sram_offs": "0x00067000", "ep": "0x00067101", "attr": "0x5"},
        {"id": "0xa5fe5a01", "bin": "app.bin",      "cert": "app.crt", "flash_offs": "32K",   "sram_offs": "0x00010000", "ep": "0x00010101", "attr": "0x5"},
        {"id": "0xa5fd5a02", "bin": "app_xip.bin",  "cert": "null", "flash_offs": "140K",  "sram_offs": "0xffffffff", "ep": "0xffffffff", "attr": "0x2"},
        {"id": "0xa5fc5a03", "bin": "net.bin",      "cert": "null", "flash_offs": "396K",  "sram_offs": "0x60000000", "ep": "0xffffffff", "attr": "0x1"},
        {"id": "0xa5fb5a04", "bin": "net_ap.bin",   "cert": "null", "flash_offs": "624K",  "sram_offs": "0x60000000", "ep": "0xffffffff", "attr": "0x1"},
        {"id": "0xa5fa5a05", "bin": "wlan_bl.bin",  "cert": "null", "flash_offs": "884K",  "sram_offs": "0xffffffff", "ep": "0xffffffff", "attr": "0x1"},
        {"id": "0xa5f95a06", "bin": "wlan_fw.bin",  "cert": "null", "flash_offs": "887K",  "sram_offs": "0xffffffff", "ep": "0xffffffff", "attr": "0x1"},
        {"id": "0xa5f85a07", "bin": "wlan_sdd.bin", "cert": "null", "flash_offs": "1015K", "sram_offs": "0xffffffff", "ep": "0xffffffff", "attr": "0x1"}
    ]
}
```

图 2-2 image config

3）在“project\<your_project>\gcc\”目录下执行“make build”重新编译生成支持安全启动的镜像。

由于“ca”目录下用于签名的密钥属于机密信息，所以，开发和量产可能需要分别使用两套不同的密钥，即：

1）开发过程中，使用临时密钥进行功能开发和验证。

2）量产时使用保密的量产密钥进行签名，生成量产镜像。

使用量产密钥生成镜像的方法如下：

1）在开发完成并确认最终软件可量产后，在“project\<your_project>\gcc\”目录下执行“make build &&make sign”。该命令将创建“project\<your_project>\image\xr872\signature”目录，该目录包含了生成安全启动镜像所需的文件。

2）将“signature”目录交给拥有量产密钥的负责人，该负责人将量产密钥的“ca”目录复制到“signature”目录中，目录内容大概如图 2-3 所示。执行脚本“./signpack.sh <image_cfg_file>”，即可生成量产镜像。

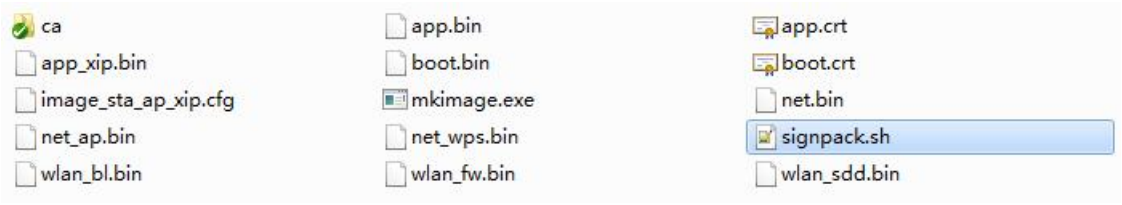

图 2-3 signature 目录内容

2.4 烧录 EFUSE Secure Boot 字段

要使完整的安全验证链生效，必须把正确的 RSA Public Key 的 SHA256 哈希值（2.1 节提到的保存在文件“cakey_pub_dcr_hash.txt”中的哈希值）烧录到芯片的 EFUSE 中，具体方法如下：

1）打开“efuse_tool.exe”工具，选择“Secure boot”选择框，并填入需要写入的 RSA Public Key 的 SHA256 哈希值（64 字符表示），如图 2-4 所示。

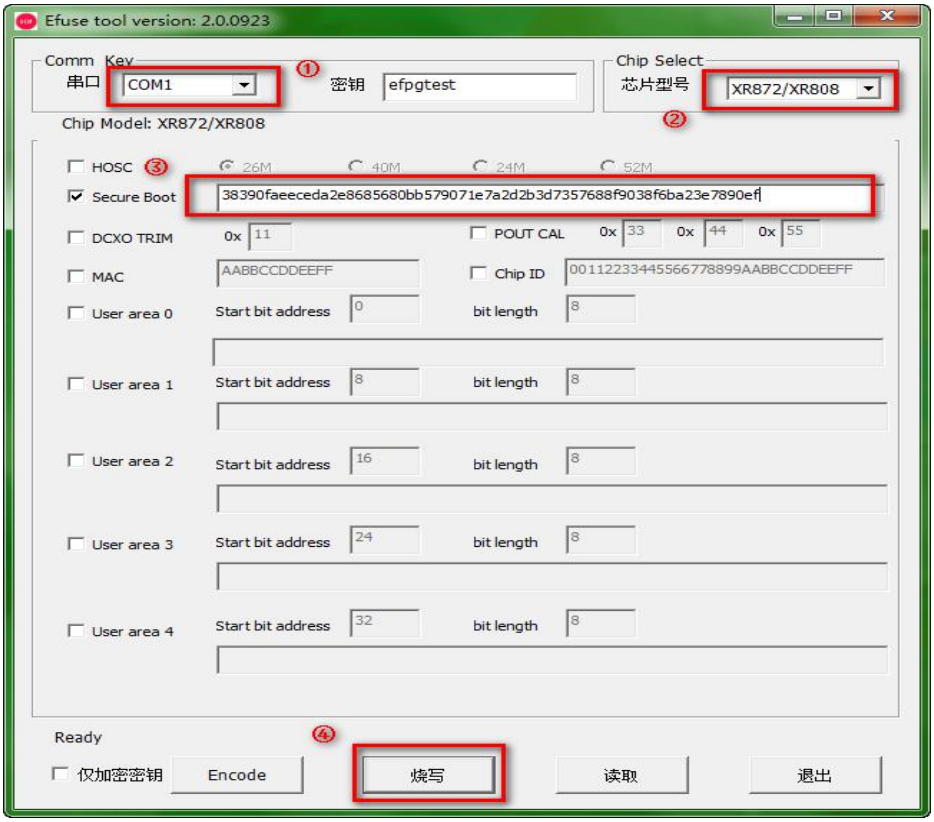

图 2-4 烧录 EFUSE Secure Boot 字段

2）点击“烧写”按钮进行 EFUSE Secure Boot 字段烧写。

注意：待烧写的工程板应保证其运行固件支持 efuse 的测试命令。