ssd20x ui_verify

zkgui工程分为不加密版本和加密版本，目前demo app默认使用加密版本。

不加密的版本：

demo地址：https://github.com/aaron201912/SSD_PLAYER

app运行半小时会自动重启，仅用来demo演示，在SDK0312版本后不再更新；

加密版本：

demo地址：https://github.com/aaron201912/UuidSSDPlayer

app运行无时间限制，但要求芯片带有uuid，且在芯片进行加密认证后才能正常运行app。

加密方式：

- **手动加密：**由用户读取设备的uuid号，提供给中科生成加密文件，然后使用烧写工具写入加密文件到KEY_CUST分区，完成加密；

获取设备uuid

依次通过读取 bank20 offset 16,17,18 获取 uuid 的值

```
/ # /config/riu_r 20 16
BANK:0x20 16bit-offset 0x16
0x47B4
/ # /config/riu_r 20 17
BANK:0x20 16bit-offset 0x17
0xE4F6
/ # /config/riu_r 20 18
BANK:0x20 16bit-offset 0x18
0x0535
```

如上图读出来的uuid为 0x0535E4F647B4，对应的字节数组为: B4 47 F6 E4 35 05。

根据uuid字节数组生成加密文件

由中科本地服务器生成.

烧写加密数据

烧写工具secutil和加密文件放在ftp中/SSD20X/Demo_Release/security_SSDUI目录下，根据设备的uuid烧写对应的加密文件，现在固定烧写至KEY_CUST分区。

如本机的UUID为0x0535E4F647B4，对应加密文件为sec_535E4F647B4.dat。

将烧写工具和加密文件拷贝到板子 /customer目录下，执行 ./secutil sec_535E4F647B4.dat KEY_CUST

```
/customer # ./secutil sec.dat KEY_CUST
fileLen 2048
read len: 456
mtd: /dev/mtd7
flash type: 4,size: 20000,erasesize: 20000
write len: 2048
```

烧写成功。烧写完成后重启设备，若此时app运行正常，则表示加密文件已经生效。

注：该方式为较早使用的加密方式，SDK008版本之后不再支持。现在使用PC端加密的方式替代这种方式来加密设备。

- **设备加密：**用户向中科申请授权设备，使用中科授权设备加密。

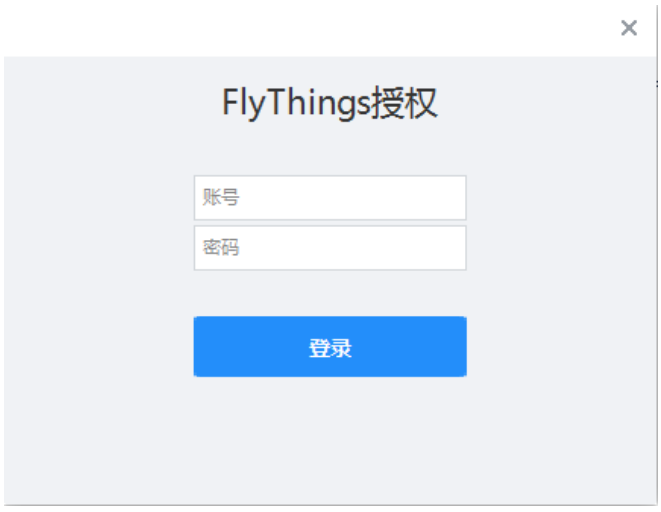
将授权设备接入以太网，确保网络畅通，并连接待授权设备的ttyS0串口与授权设备的串口，TX→RX, RX→TX，GND->GND。

使用账号密码登录。进入主页面后点击“选项”按钮，按需选择授权项目。选择完毕后，点击“启动授权服务”，烧录服务准备就绪（再次点击该按钮可暂停授权服务）。

授权机会自动对待授权设备进行授权。授权设备会显示具体的授权过程。当日志显示“授权成功”时，被授权设备正常启动，即可断开设备连接，然后更换下一台设备进行授权。

- **PC端加密：**使用PC端授权工具加密。

  电脑连接以太网，确保网络正常。待授权设备ttyS0串口连接电脑，断开其它应用中该串口的连接。启动授权程序，输入账号密码登录。



选择授权的项目，点击“启动授权”按钮开始授权。带界面上出现“授权成功”的日志时，表示授权成功，待授权设备正常开机后可更换下一台设备进行授权。

清除加密分区：

获取KEY_CUST分区设备节点：

cat /proc/mtd

```
/ # cat /proc/mtd
dev:    size   erasesize  name
mtd0: 00060000 00020000 "IPL0"
mtd1: 00060000 00020000 "IPL1"
mtd2: 00060000 00020000 "IPL_CUST0"
mtd3: 00060000 00020000 "IPL_CUST1"
mtd4: 000c0000 00020000 "UBOOT0"
mtd5: 000c0000 00020000 "UBOOT1"
mtd6: 00060000 00020000 "ENV0"
mtd7: 00020000 00020000 "KEY_CUST"
mtd8: 00060000 00020000 "LOGO"
mtd9: 00500000 00020000 "KERNEL"
mtd10: 00500000 00020000 "RECOVERY"
mtd11: 070e0000 00020000 "UBI"
```

清除KEY_CUST分区：如上图，KEY_CUST分区对应mtd7

dd if=/dev/zero of=/dev/mtd7 count=1 bs=4K && echo > /dev/mtd7

授权常见问题：

- **被授权设备运行zkgui时出现“ekey not match“的打印信息**

```
[MI_PANEL_Init][331]LCD environment is Invalid
readXXXFile /mnt/extsd/EasyUI.cfg fail errno: No such file or directory !!!
readXXXFile /res/etc/EasyUI.cfg fail errno: No such file or directory !!!
initEasyUICfg ok!
readXXXFile /data/preferences.json fail errno: No such file or directory !!!
readXXXFile /system/res/internal/lang/en_US.json[MI ERR ]: MI_SYS_IMPL_Exit[3707]: gSysInitCount:1
 fail errno: No such file or directory !!!
readXXXFile /data/preferences.json fail errno: No suclient [795] disconnected, module:panel
client [795] disconnected, module:disp
ch file or directory !!!
writeXXXFile /data/preferences.json fail errno: No such file or directory !!!
ExecuteThread start...
crc32: 738e5f7a, ait.crc32: 738e5f7a
ekey not match!
Contact www.zkswe.com!
sstar_disp_Deinit...
client [795] disconnected, module:sys
```

出现这个问题是因为使用的libeasyui.so的版本与当前烧写的key不兼容。比较可能的原因是：

1. 之前有通过手动方式加密写入过key，更换为新版本的libeasyui.so时再次授权时，未先擦除KEY_CUST分区，导致授权失败；
2. 之前有基于新版本的libeasyui.so成功进行授权，后面又人为替换其它版本的libeasyui.so，导致key与libeasyui.so出现不兼容的情况。

- **授权时提示串口超时**

```
[MI_PANEL_Init][331]LCD environment is Invalid
readXXXFile /mnt/extsd/EasyUI.cfg fail errno: No such file or directory !!!
readXXXFile /res/etc/EasyUI.cfg fail errno: No such file or directory !!!
initEasyUICfg ok!
readXXXFile /data/preferences.json fail errno: No such file or directory !!!
readXXXFile /system/res/internal/lang/en_US.json fail errno: No such file or directory !!!
readXXXFile /data/preferences.json fail errno: No such file or directory !!!
writeXXXFile /data/preferences.json fail errno: No such file or directory !!!
ExecuteThread start...
crc32: 2dd41b51, ait.crc32: ffffffff
uart authorize time out!
ExecuteThread end...
uart authorize time out!
uart authorize time out!
```

出现这个问题是串口连接出现异常，需要检查串口RX，TX是否接反，连接线接触是否正常。

不加密版本的工程要变更成支持加密的版本需要对现有的工程项目和app的运行环境做更新，具体变更如下：

1. 更新项目工程jni下的include目录和libeasyui.so。编译libzkgui.so时先清空工程，然后选择编译flything。

2. 修改项目代码，项目中使用MPPOINT结构类型的需要替换为SZKPoint

3. 更新zkgui bin目录库，同步github上app目录下Makefile和app/minigui_dir/lib下lib。

4. 使用sdk/verify/application/zk_full_sercurity下的lib作为基本的运行环境。当修改app时，仅需替换生成的libzkgui.so和ui目录和添加app所依赖的第三方库即可。