Secure Files

# Introduction

The secure\_files example application demonstrates reading and writing encrypted files from/to the filesystem. To use secure\_files, the secureboot mode should be enabled.

For more details on secureboot mode, refer Application\_for\_using\_SSBL.pdf (*sdk\_x.y\_alpha\apps\ssbl\doc*).

**Note**: x and y in sdk\_x.y refer to the SDK package release version.

# Description of Operation

1. Reconstruct secureboot secret and get the firmware cipher key from flash
2. Mount the filesystem
3. Write the encrypted message to a file using firmware cipher key
4. Read the encrypted message as plain text (should be garbled)
5. Read and decrypt encrypted message (this should show the original message contents)

# Building and Flashing Process

## Building Components

1. Build secure SSBL (refer section: 7.2.1 in Application for\_using\_SSBL.pdf (*sdk\_x.y\_alpha\apps\ssbl\doc)*
2. Build the secure\_files example:

|  |
| --- |
| cd <sdk>/examples/secure\_files  make clean  make KEY=../../apps/ssbl/enroll.json |

## Flashing Components

1. Enroll keys & flash SSBL components in secureboot mode (refer steps 1 to 4 of section: 7.2.2 in Application for\_using\_SSBL.pdf (*sdk\_x.y\_alpha\apps\ssbl\doc)*
2. Build the secure filesystem (data\_secure.img):

|  |
| --- |
| cd <sdk>  python ./script/build\_rootfs\_generic.py --folder\_path examples/secure\_files/ --secure True --keyfile ./apps/ssbl/enroll.json |

1. Create Signed and encrypted ELF (secure\_files.elf.enc)

|  |
| --- |
| cd <sdk>/examples/secure\_files  make clean  make KEY=<sdk>/apps/ssbl/enroll.json |

1. Flash application at 0x20000

|  |
| --- |
| cd <sdk>/examples/secure\_files$  ../../script/flash.py write 0x20000 out/secure\_files.elf.enc |

1. Flash the filesystem

|  |
| --- |
| cd <sdk>/examples/secure\_files$  ../../script/flash.py write 0x180000 root\_secure.img |

## Expected Output

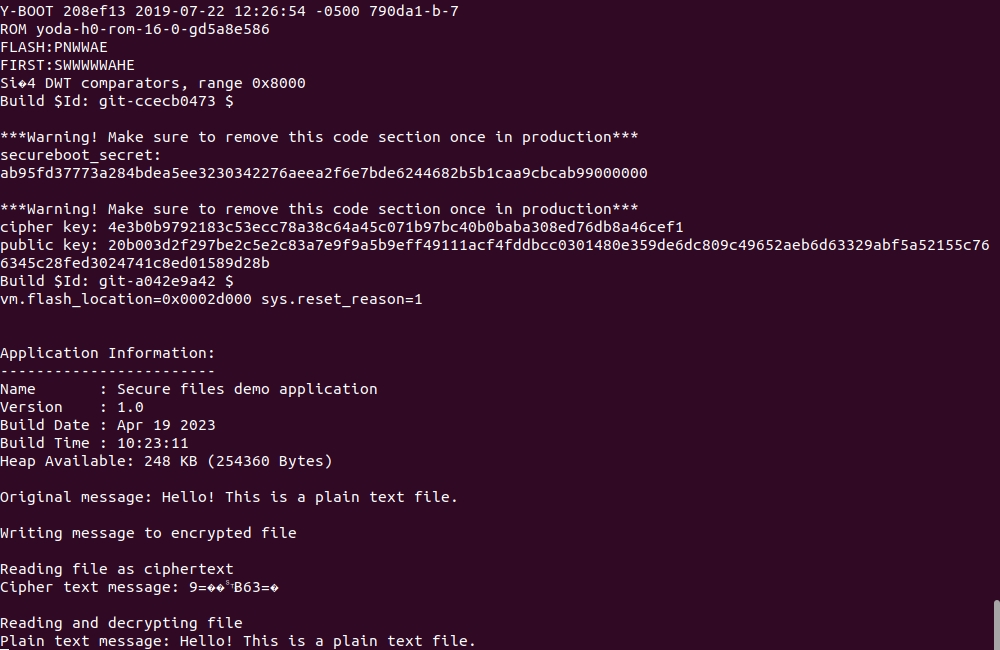


Figure : Expected output