Crypto af_alg blackkey demo

Biyong SUN

26, NOV 2021

Introduction

Since LF_v5.10.52-2.1.0 crypto_af_alg blackkey demo “caam-decrypt” becomes default in release. You can try it with binary demo release image.

The demo is using black key to decrypt data.

This document goes more detail based on BSP release document

i.MX Linux® User's Guide, Rev. LF5.10.52_2.1.0, 15 October 2021
10.6 crypto_af_alg application support

Demo environment

HW: i.MX8MM EVK
SW: LF_v5.10.52-2.1.0_images_IMX8MMEVK binary demo image

caam-decrypt source code:
https://source.codeaurora.org/external/imx/crypto_af_alg/

Note：
caam-decrypt already in binary demo image

PC side

1. generate key and iv by openssl

echo 12345 | openssl enc -aes-256-cbc -k - -P -md sha1 -pbkdf2
salt=1982686A7BACEE4D
key=D84041EC14BB28543E8545BEB094FE643B5BC1345C31CD576BC708A1559FBD2D
iv =F950CACE80F76F0AC00D9C8762B3A5C9

2. encrption by openssl

echo "**For test caam-decrypt**" | openssl enc -e -aes-256-cbc -in - -out test.txt.enc
-K D84041EC14BB28543E8545BEB094FE643B5BC1345C31CD576BC708A1559FBD2D
-iv F950CACE80F76F0AC00D9C8762B3A5C9

PC side（Cont.）

3. convert key and iv to plaint txt for caam-decrypt

```
echo F950CACE80F76F0AC00D9C8762B3A5C9| xxd -r -p > fromopenssl.iv.txt
echo D84041EC14BB28543E8545BEB094FE643B5BC1345C31CD576BC708A1559FBD2D| xxd -r -p > fromopenssl.key.txt
```

why need this step?
Because openssl uses hex key.
But caam-keygen and caam-decrypt need plaint text key and iv.

For Openssl
F950CACE80F76F0AC00D9C8762B3A5C9
means
0xF9,0x50,0xCA,0xCE,0x80,0xF7,0x6F,0x0A,0xC0,0x0D,0x9C,0x87,0x62,0xB3,0xA5,0xC9

hexdump -C fromopenssl.iv.txt
00000000 f9 50 ca ce 80 f7 6f 0a c0 0d 9c 87 62 b3 a5 c9 |.P....o.....b...|
00000010

PC side（Cont.）

4. prepare data for caam-decrypt
   cat fromopenssl.iv.txt test.txt.enc > data.caam-decrypt.enc

   note:
   the format for caam-decrypt

   AES Encrypted file format
   16 Octets - Initialization Vector (IV) is an input to encryption algorithm.
   nn Octets - Encrypted message (for AES-256-CBC, it must be multiple of 16)

Now you can send fromopenssl.key.txt and data.caam-decrypt.enc to the board.

on i.MX8MM evk board

1. generate blackkey blob
   caam-keygen create blackkey ecb -t $(cat fromopenssl.key.txt)

Now, you will find /data/caam/{blackkey, blackkey.bb}

Original openssl output key:
D84041EC14BB28543E8545BEB094FE643B5BC1345C31CD576BC708A1559FBD2D

Plaint text key:

```
00000000  d8 40 41 ec 14 bb 28 54  3e 85 45 be b0 94 fe 64  |.@A...(T>.E....d|
00000010  3b 5b c1 34 5c 31 cd 57  6b c7 08 a1 55 9f bd 2d  |;[.4¥1.Wk...U..-|
```

Black key:

```
00000000  4f 67 61 54 00 00 00 00  01 00 00 00 20 00 00 00  |OgaT........ ...|
00000010  5c 00 00 00 c1 9e 96 67  71 de 44 13 11 db d3 45  |¥......gq.D....E|
00000020  33 e4 37 0e d5 2b 24 f8  c4 ba 98 99 6e 9a ca 19  |3.7..+$.....n...|
00000030  73 c2 5c a1 80 77 2b 4c  c9 ed 85 03 5a 89 cf c8  |s.¥..w+L....Z...|
00000040  fe 5c 18 a7 27 c0 b7 de  f2 c0 c5 c5 f9 b4 78 fd  |.¥..'.........x.|
00000050  fd 6f 15 cc d9 8f 06 69  2e 0c 00 a1 34 96 39 c9  |.o.....i....4.9.|
00000060  39 9a 01 ec 00 00 00 00  00 00 00 00 00 00 00 00  |9...............|
```

on i.MX8MM evk board (Cont.)

2. delete fromopenssl.key.txt
   if you want to, you can also delete the blackkey only leave blackkey.bb

3. test decryption by caam-decrypt with blackkey

caam-decrypt /data/caam/blackkey.bb AES-256-CBC data.caam-decrypt.enc data.caam-decrypt.dec
root@imx8mmevk:/# cat data.caam-decrypt.dec
For test caam-decrypt

on i.MX8MM evk board(cont.)

COM10:115200baud - Tera Term VT

File Edit Setup Control Window Help

```
root@imx8mmevk:/#
root@imx8mmevk:/#
root@imx8mmevk:/#
root@imx8mmevk:/#
root@imx8mmevk:/# caam-decrypt  /data/caam/blackkey.bb AES-256-CBC data.caam-decrypt.enc    data.caam-decrypt.dec
root@imx8mmevk:/# cat data.caam-decrypt.dec
For test caam-decrypt




root@imx8mmevk:/#
```

on i.MX8MM evk board(Cont.)

Note: If the chip is not closed, the black key blob can use on any chip on any board.
Because the black key is generated by a test key.
The test key is same on every chip.

Once it is closed, the black key is generated by master key.
Each one of the chip, has one unique master key.
So only the chip generate the black key can do decryption.

A chip/board generates black key blob “A.bb”, then move to the B chip/blob.
B chip/board can not use A.bb to do decryption.

The key code in caam-decrypt

caam-decrypt is using AF_ALG socket.
It is Linux layer programming. It is common.

```
 int main(int argc, char *argv[])
{
        struct sockaddr_alg sa = {
                .salg_family = AF_ALG,
                .salg_type = "skcipher",        /* selects the symmetric cipher */
                .salg_name = "tk(cbc(aes))"   /* this is the cipher name */
        };
```

The key code in caam-decrypt(cont.)

caam-decrypt is calling caam-keygen to import black key.

caam-decrypt.h

```
#define CAAM_KEYGEN_APP                 "/usr/bin/caam-keygen"
#define CAAM_KEYGEN_IMPORT              "import"
#define KEY_LOCATION                    "/data/caam/"
#define KEY_NAME                "black_key"
#define IV_LEN                          16

int caam_import_black_key(char *blob_name)
{
        pid_t cpid, w;
        int status;
        char *argv[] = {CAAM_KEYGEN_APP, CAAM_KEYGEN_IMPORT, NULL, KEY_NAME, NULL};

        argv[2] = blob_name;
        /*
        * Command to be execute, to create a black key is:
        * /usr/bin/caam-keygen import <blob_name> <key_name>
```

uuu for caam-keygen

fsl-image-mfgtool-initramfs-imx_mfgtools.cpio.gz.u-boot for uuu have no caam modules need to collect the caam modules

In this demo, use LF_v5.10.52-2.1.0_images_IMX8MMEVK.zip
imx-image-multimedia-imx8mmevk.wic

This is to use one build.

You can build all modules as build-in, so you can skip this step.

uuu for caam-keygen(cont.)

sudo kpartx -av imx-image-multimedia-imx8mmevk.wic
sudo mount /dev/mapper/loop0 /mnt

Copy following files in /mnt mounted

/lib/modules/5.10.52-lts-5.10.y+ga11753a89ec6/kernel/drivers/crypto/caam/error.ko
/lib/modules/5.10.52-lts-5.10.y+ga11753a89ec6/kernel/lib/crypto/libdes.ko
/lib/modules/5.10.52-lts-5.10.y+ga11753a89ec6/kernel/crypto/authenc.ko
/lib/modules/5.10.52-lts-5.10.y+ga11753a89ec6kernel/drivers/char/hw_random/rng-core.ko
/lib/modules/5.10.52-lts-5.10.y+ga11753a89ec6.bak/kernel/crypto/crypto_engine.ko
/lib/modules/5.10.52-lts-5.10.y+ga11753a89ec6.bak/kernel/drivers/crypto/caam/caamalg_desc.ko
/lib/modules/5.10.52-lts-5.10.y+ga11753a89ec6.bak/kernel/drivers/crypto/caam/caamhash_desc.ko
/lib/modules/5.10.52-lts-5.10.y+ga11753a89ec6.bak/kernel/drivers/crypto/caam/caamkeyblob_desc.ko
/lib/modules/5.10.52-lts-5.10.y+ga11753a89ec6.bak/kernel/drivers/crypto/caam/caam.ko
/lib/modules/5.10.52-lts-5.10.y+ga11753a89ec6.bak/kernel/drivers/crypto/caam/caam_jr.ko

uuu for caam-keygen(cont.)

In this demo all the caam modules are in the caam_module.tar

```
caam_module/
|-- authenc.ko
|-- caamalg_desc.ko
|-- caamhash_desc.ko
|-- caam_jr.ko
|-- caamkeyblob_desc.ko
|-- caam.ko
|-- crypto_engine.ko
|-- error.ko
|-- ins_caam_modules
|-- libdes.ko
`-- rng-core.
```

```
#!/bin/bash

#ins_caam_modules

modules="error libdes authenc  \
        rng-core crypto_engine  \
        caamalg_desc caamhash_desc  \
        caamkeyblob_desc caam caam_jr"

for module in ${modules}
do
    echo insmod ${module}
    insmod ${module}.ko
done

echo
echo
ls -l  /dev/caam-keygen
echo
```

uuu for caam-keygen

fsl-image-mfgtool-initramfs-imx_mfgtools.cpio.gz.u-boot for uuu have no
xxd and caam-keygen

You can get the caam-keygen in imx-image-multimedia-imx8mmevk.wic or in your build.

In this demo build arm static link from
https://source.codeaurora.org/external/imx/keyctl_caam latest tag lf-5.10.35-2.0.0

xxd is built from vim source code as arm static link

tar fvt caam-keygen_xdd.tar
-rwxrwxr-x caam-keygen
-rwxrwxr-x xxd

uuu for caam-keygen(cont.)

caam-keygen.uuu

.....

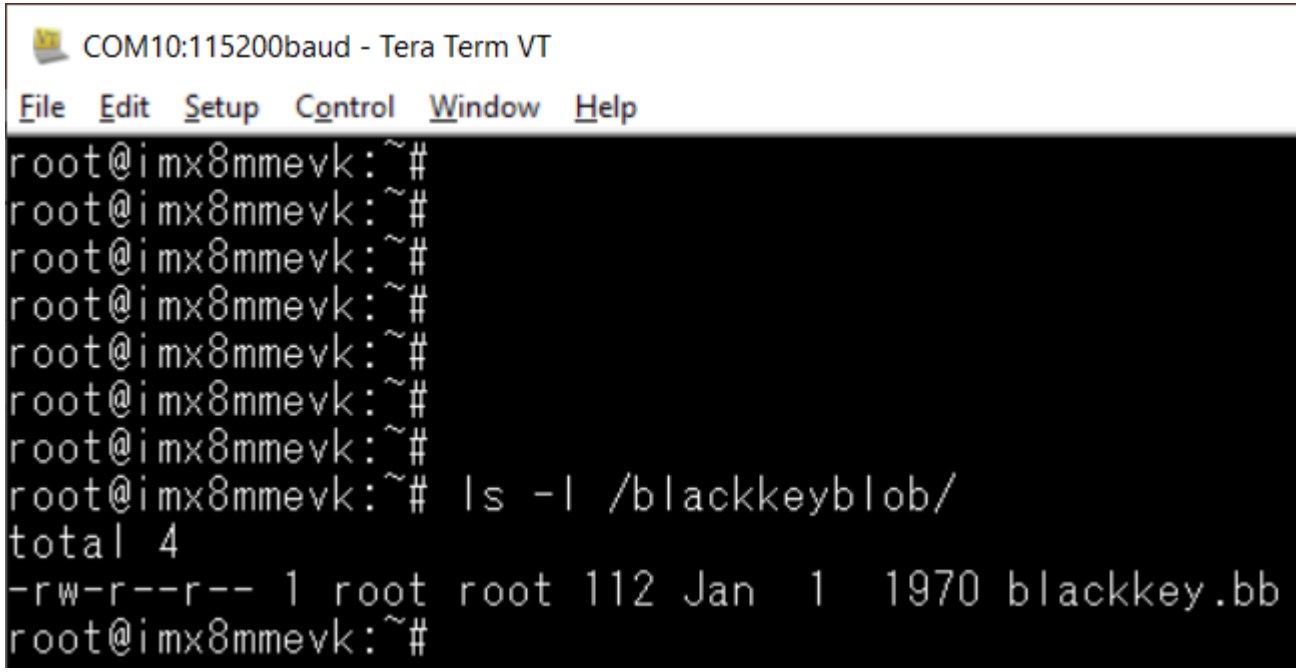
……

FBK: ucp caam_module.tar t:/tmp/caam_module.tar
FBK: ucmd tar -xf /tmp/caam_module.tar -C /tmp/
FBK: ucmd cd   /tmp/caam_module/ && ./ins_caam_modules
FBK: ucp caam-keygen_xdd.tar t:/tmp/caam-keygen_xdd.tar
FBK: ucmd tar -xf /tmp/caam-keygen_xdd.tar -C /bin/
FBK: ucmd caam-keygen create blackkey ecb -t "$(echo D84041EC14BB28543E8545BEB094FE643B5BC1345C31CD576BC708A1559FBD2D| xxd -r -p)"
FBK: ucmd ls -l /data/caam/
FBK: ucmd mkdir -p /mnt/ext3
FBK: ucmd mmc=`cat /tmp/mmcdev`; mount /dev/mmcblk${mmc}p2 /mnt/ext3
FBK: ucmd mkdir -p /mnt/ext3/blackkeyblob
FBK: acmd export EXTRACT_UNSAFE_SYMLINKS=1
FBK: ucmd cp -f /data/caam/blackkey.bb /mnt/ext3/blackkeyblob/
FBK: Sync
FBK: ucmd umount /mnt/ext3
FBK: DONE

uuu for caam-keygen(cont.)

uuu caam-keygen.uuu

After flash, you will find blackkey.bb in /blackkeyblob/



Note:
caam-keygen.uuu assume you
have done flash os system.

caam-keygen.uuu only the script to
generate black key blob
and store in /blackkeyblob/ of rootfs.

If you want do all together, please integrate
caam-keygen.uuu script to your uuu script of flash os system

SECURE CONNECTIONS
FOR A SMARTER WORLD