Linux Core Crypto User Guide

Introduction

The Crypto API Driver is a set of Linux drivers that provide access to the hardware cryptographic accelerators available on AM335x/AM437x/AM57x/DRA7 devices. These drivers are available built-in in the kernel in the current SDK release.

Following are the Hardware accelerators supported on the following devices:

```
* AM335X     : MD5, SHA1, SHA224, SHA256, AES, DES
* AM437X     : MD5, SHA1, SAH224, SHA256, SHA384, SHA512, AES, DES, DES3DES
* AM57x/DRA7 : AES, DES, DES3DES
```

Building the Driver

For devices with available cryptographic hardware accelerators, a Linux driver and additionally an Cryptodev (or OCF on AMSDK v6.0 or older) kernel module (for OpenSSL) is needed to access them. Other devices use the pure software implementation of OpenSSL for the crypto demos.

AM335x, AM43xx - AES, DES, SHA/MD5 Drivers

Starting with **AMSDK 5.05.00.00**, the driver is completely integrated into the kernel source. The pre-built kernel that comes with the SDK already has the AES, DES and SHA/MD5 drivers built-in to the kernel. The kernel configuration has already been set up in the SDK and no further configuration is needed for the drivers to be built-in to the kernel. The configuration of the random number generator does require an extra step and this is detailed in the next section.

For reference, the configuration details are shown below. The configuration of the AES, DES and SHA/MD5 driver is done under the Hardware crypto devices sub-menu of the Cryptographic API menu in the kernel configuration.

```
--- Cryptographic API
    [*] Hardware crypto devices --->
        --- Hardware crypto devices
            <*> Support for OMAP MD5/SHA1/SHA2 hw accelerator
            <*> Support for OMAP AES hw engine
            <*> Support for OMAP DES3DES hw engine
```

Messages printed during bootup will indicate that initialization of the crypto modules has taken place.

```
[    2.120565] omap-sham 53100000.sham: hw accel on OMAP rev 4.3
[    2.160584] mmc1: BKOPS_EN bit is not set
[    2.173466] omap-aes 53500000.aes: OMAP AES hw accel rev: 3.2
[    2.180241] edma-dma-engine edma-dma-engine.0: allocated channel for 0:5
[    2.187808] edma-dma-engine edma-dma-engine.0: allocated channel for 0:6
```

Build the Cryptodev kernel module using SDK

For using OpenSSL to access the Crypto Hardware Accelerator Drivers above, the Cryptodev is required (can be built as module). The framework is not officially in the kernel and was ported to Linux under the name "cryptodev".

Using Cryptographic Hardware Accelerators

Using the TRNG Hardware Accelerator

The pre built kernel that come with the SDK already has the TRNG driver built into the kernel. No further configuration is required.

For reference, the configuration details are shown below.

In the configuration menu, scroll down to Device Drivers and hit enter. Now scroll to Character devices and hit enter.

```
Device Drivers --->
   Character devices --->
       < > Hardware Random Number Generator Core support
           < > OMAP Random Number Generator support
```

```
[    1.660514] omap_rng 48310000.rng: OMAP Random Number Generator ver. 20
```

Once the system is booted up, the hwrng device should now show up in the filesystem.

```
root@am335x-evm:~# ls -l /dev/hwrng
crw------- 1 root root 10, 183 Jan 1 2000 /dev/hwrng
root@am335x-evm:~#
```

Use cat on this device to generate random numbers.

```
root@am335x-evm:~# cat /dev/hwrng | od -x
0000000 b2bd ae08 4477 be48 4836 bf64 5d92 01c9
0000020 0cb6 7ac5 16f9 8616 a483 7dfd 6bf4 3aa5
0000040 d693 db24 d917 5ee7 feb7 34c3 34e9 e7a5
0000060 36b7 ea85 fc17 0e66 555c 0934 7a0c 4c69
0000100 523b 9f21 1546 fddb d58b e5ed 142a 6712
0000120 8d76 8f80 a6d2 30d8 d107 32bc 7f45 f997
0000140 9d5d 0d0c f1f0 64f9 a77f 408f b0c1 f5a0
0000160 39c6 f0ae 4b59 1a76 84a7 a364 8964 f557
root@am335x-evm:~#
```

Support tools for the hardware random number generator can be loaded from rng-tools on Sourceforge [1]. The latest version at the time of this write-up is version 3.0 [2], dated 2010-07-04.

1. We're still in the Linux-devkit environment. Download the file rng-tools-3.tar.gz, and untar in a suitable location.

2. Change to the directory that contains the rng-tools distribution, and configure the package:

```
host $ ./configure --prefix=/home/user/targetfs/TI814x-targetfs_5_03_01/usr \
 --exec-prefix=/home/user/targetfs/TI814x-targetfs_5_03_01/usr \
 --host --target=arm-linux
```

3. Next make the **rngd** and **rngtest** executables.

```
host $ make
```

4. Install the generated executables in the target filesystem.

5. Test the random number generator on the target.

```
root@am335x-evm:~# cat /dev/hwrng | rngtest -c 1000
rngtest 3
Copyright (c) 2004 by Henrique de Moraes Holschuh
This is free software; see the source for copying conditions.  There is
 NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR
PURPOSE.

rngtest: starting FIPS tests...
rngtest: bits received from input: 20000032
rngtest: FIPS 140-2 successes: 999
rngtest: FIPS 140-2 failures: 1
rngtest: FIPS 140-2(2001-10-10) Monobit: 0
rngtest: FIPS 140-2(2001-10-10) Poker: 0
rngtest: FIPS 140-2(2001-10-10) Runs: 1
rngtest: FIPS 140-2(2001-10-10) Long run: 0
rngtest: FIPS 140-2(2001-10-10) Continuous run: 0
rngtest: input channel speed: (min=788.218; avg=4070.983;
max=2790178.571)Kibits/s
rngtest: FIPS tests speed: (min=846.755; avg=15388.376;
max=21920.595)Kibits/s
rngtest: Program run time: 6072670 microseconds
```

Note that the results may be slightly different on your system, since, after all, we're dealing with a random number generator. Any appreciable number of errors typically indicates a bad random number generator.

If you're satisfied the random number generator is working correctly, you can use **rngd** (the random number generator daemon) to feed the /dev/random entropy pool.

AES, DES, SHA Hardware Accelerators using Cryptodev

The device drivers for AES, DES and SHA/MD5 hardware acceleration is configured and built into the kernel by default. No other special setup is needed for OpenSSL to access the crypto modules.

First, the kernel from the SDK must be configured and built according to the SDK User's Guide.

The General Purpose (GP) EVMs on TI SoCs allows access to built in cryptographic accelerators. Inorder to use these drivers from OpenSSL, the drivers on their own have no contact with userspace. For this, a special driver is available which abstracts the access to these accelerators through Cryprodev module.

The demo application under the crypto menu of Matrix will load and use the Cryptodev driver kernel modules automatically to perform hardware accelerated crypto functions. The process of manually loading the kernel modules and using the driver is explained below.

Cryptodev is itself a special device driver which provides a general interface for higher level applications such as OpenSSL to access hardware accelerators.

The filesystem which comes with the SDK comes built with the Cryptodev kernel modules and the TI driver which directly accesses the hardware accelerators is built into the kernel.

From the target boards perspective the drivers are located in the following directories:

```
/lib/modules/`uname -r`/extra/cryptodev.ko
```

To use the drivers they must first be installed. Use the modprobe command to install the drivers. The following log shows the commands used to install the modules and query the system for the state of all system modules.

```
root@am335x-evm:~# lsmod
Module                  Size  Used by
cryptodev              11962  0
root@am335x-evm:~#
```

After the modules are installed, OpenSSL commands may be executed which take advantage of the hardware accelerators through the Cryptodev driver. The following example demonstrates the OpenSSL built-in speed test to demonstrate performance. The addition of the parameter **-engine cryptodev** tells OpenSSL to use the Cryptodev driver if it exists.

```
root@am335x-evm:~# openssl speed -evp aes-128-cbc -engine cryptodev
engine "cryptodev" set.
Doing aes-128-cbc for 3s on 16 size blocks: 108107 aes-128-cbc's in 0.16s
Doing aes-128-cbc for 3s on 64 size blocks: 103730 aes-128-cbc's in 0.20s
Doing aes-128-cbc for 3s on 256 size blocks: 15181 aes-128-cbc's in 0.03s
Doing aes-128-cbc for 3s on 1024 size blocks: 15879 aes-128-cbc's in 0.03s
Doing aes-128-cbc for 3s on 8192 size blocks: 4879 aes-128-cbc's in 0.02s
OpenSSL 1.0.0b 16 Nov 2010
built on: Thu Jan 20 10:23:44 CST 2011
options:bn(64,32) rc4(ptr,int) des(idx,risc1,2,long) aes(partial) idea(int) blowfish(idx)
compiler: arm-none-linux-gnueabi-gcc -march=armv7-a -mtune=cortex-a8 -mfpu=neon -mfloat-abi=softfp -mthumb-interwork -mno-thumb -fPS
The 'numbers' are in 1000s of bytes per second processed.
type 16 bytes 64 bytes 256 bytes 1024 bytes 8192 bytes
aes-128-cbc 10810.70k 33193.60k 129544.53k 542003.20k 1998438.40k
root@am335x-evm:~#
root@am335x-evm:~#
root@am335x-evm:~#
```

Using the Linux time -v function gives more information about CPU usage during the test.

```
root@am335x-evm:~# time -v openssl speed -evp aes-128-cbc -engine cryptodev
engine "cryptodev" set.
Doing aes-128-cbc for 3s on 16 size blocks: 108799 aes-128-cbc's in 0.17s
Doing aes-128-cbc for 3s on 64 size blocks: 102699 aes-128-cbc's in 0.18s
Doing aes-128-cbc for 3s on 256 size blocks: 16166 aes-128-cbc's in 0.03s
Doing aes-128-cbc for 3s on 1024 size blocks: 15080 aes-128-cbc's in 0.03s
Doing aes-128-cbc for 3s on 8192 size blocks: 4838 aes-128-cbc's in 0.03s
OpenSSL 1.0.0b 16 Nov 2010
built on: Thu Jan 20 10:23:44 CST 2011
options:bn(64,32) rc4(ptr,int) des(idx,risc1,2,long) aes(partial) idea(int) blowfish(idx)
compiler: arm-none-linux-gnueabi-gcc -march=armv7-a -mtune=cortex-a8 -mfpu=neon -mfloat-abi=softfp -mthumb-interwork -mno-thumb -fPS
The 'numbers' are in 1000s of bytes per second processed.
type 16 bytes 64 bytes 256 bytes 1024 bytes 8192 bytes
aes-128-cbc 10239.91k 36515.20k 137949.87k 514730.67k 1321096.53k
Command being timed: "openssl speed -evp aes-128-cbc -engine cryptodev"
User time (seconds): 0.46
System time (seconds): 5.89
```

```
Percent of CPU this job got: 42%
Elapsed (wall clock) time (h:mm:ss or m:ss): 0m 15.06s
Average shared text size (kbytes): 0
Average unshared data size (kbytes): 0
Average stack size (kbytes): 0
Average total size (kbytes): 0
Maximum resident set size (kbytes): 7104
Average resident set size (kbytes): 0
Major (requiring I/O) page faults: 0
Minor (reclaiming a frame) page faults: 479
Voluntary context switches: 36143
Involuntary context switches: 211570
Swaps: 0
File system inputs: 0
File system outputs: 0
Socket messages sent: 0
Socket messages received: 0
Signals delivered: 0
Page size (bytes): 4096
Exit status: 0
```

When the cryptodev driver is removed, OpenSSL reverts to the software implementation of the crypto algorithm. The performance using the software only implementation can be compared to the previous test.

```
root@am335x-evm:~# modprobe -r cryptodev
root@am335x-evm:~# time -v openssl speed -evp aes-128-cbc
Doing aes-128-cbc for 3s on 16 size blocks: 697674 aes-128-cbc's in 2.99s
Doing aes-128-cbc for 3s on 64 size blocks: 187556 aes-128-cbc's in 3.00s
Doing aes-128-cbc for 3s on 256 size blocks: 47922 aes-128-cbc's in 3.00s
Doing aes-128-cbc for 3s on 1024 size blocks: 12049 aes-128-cbc's in 3.00s
Doing aes-128-cbc for 3s on 8192 size blocks: 1509 aes-128-cbc's in 3.00s
OpenSSL 1.0.0b 16 Nov 2010
built on: Thu Jan 20 10:23:44 CST 2011
options:bn(64,32) rc4(ptr,int) des(idx,risc1,2,long) aes(partial) idea(int) blowfish(idx)
compiler: arm-none-linux-gnueabi-gcc -march=armv7-a -mtune=cortex-a8 -mfpu=neon -mfloat-abi=softfp -mthumb-interwork -mno-thumb -fPS
The 'numbers' are in 1000s of bytes per second processed.
type 16 bytes 64 bytes 256 bytes 1024 bytes 8192 bytes
aes-128-cbc 3733.37k 4001.19k 4089.34k 4112.73k 4120.58k
Command being timed: "openssl speed -evp aes-128-cbc"
User time (seconds): 15.03
System time (seconds): 0.00
Percent of CPU this job got: 99%
Elapsed (wall clock) time (h:mm:ss or m:ss): 0m 15.07s
Average shared text size (kbytes): 0
Average unshared data size (kbytes): 0
Average stack size (kbytes): 0
Average total size (kbytes): 0
Maximum resident set size (kbytes): 7216
```

```
Average resident set size (kbytes): 0

Major (requiring I/O) page faults: 1

Minor (reclaiming a frame) page faults: 484

Voluntary context switches: 13

Involuntary context switches: 35

Swaps: 0

File system inputs: 0

File system outputs: 0

Socket messages sent: 0

Socket messages received: 0

Signals delivered: 0

Page size (bytes): 4096

Exit status: 0
```

References

[1] http://sourceforge.net/projects/gkernel/files/rng-tools/

[2] http://sourceforge.net/projects/gkernel/files/rng-tools/3/

Article Sources and Contributors

Linux Core Crypto User Guide *Source*: http://processors.wiki.ti.com/index.php?oldid=234129 *Contributors*: HongmeiGou, Joelagnel, Liubin