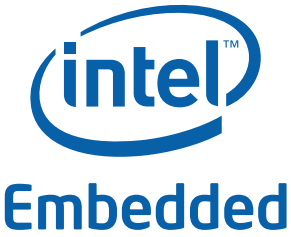

Linux* Kernel Cryptographic Driver for Intel® QuickAssist Technology

Application Note

April 2010

Order Number: 323643-001US

Contents

Tables

Revision History

Date	Revision	Description
April 2010	001	Initial release of this document.

§ §

1.0 Introduction

1.1 About this Manual

This document discusses the following topics about the Linux* Kernel Cryptographic Driver for Intel® QuickAssist Technology:

- features and limitations
- build and installation

1.2 Software Overview

This section lists the features and limitations of the Linux* Kernel Cryptographic Driver for Intel® QuickAssist Technology.

1.2.1 Features

This software generates a GNU/Linux kernel module that accelerates some of the IPSec algorithms using the Intel® EP80579 Software on Intel® QuickAssist Technology.

- Supported algorithms are:
  - aes-128-cbc-hmac-sha1
  - 3des-cbc-hmac-sha1
  - aes-128-gcm
- Supported IPSec mode is Encapsulating Security Payload (ESP). Authentication Header (AH) mode is NOT supported.
- Supported IPSec protocol is IPv4 protocol. IPSec for IPv6 is NOT supported.
- Supports both tunnel and transport mode.

1.2.2 Limitations

The Linux* Kernel Cryptographic Driver for Intel® QuickAssist Technology has the following limitations:

- Authentication Header (AH) mode is not supported
- IPSec for IPv6 protocol is not supported

1.3 Documentation

1.3.1 Where to Find Current Software and Documentation

The software release and associated collateral can be found on the Hardware Design resource center.

1. In a web browser, go to http://www.intel.com/go/soc
2. For Software and pre-boot firmware: Click on “Tools & Software” tab.

3. For Documentation: Click on “Technical Documents” tab.

The EP80579 security software release package contains encryption software and is subject to export requirements defined by the U.S Department of Commerce. To satisfy these requirements, the End User Certification Form must be filled out and submitted for review/approval. Instructions on this process are included during the download process. Please note that this process may take up to two business days to complete.

1.3.2 Product Documentation

This release includes:

- Linux* Kernel Cryptographic Driver for Intel® QuickAssist Technology Application Note (this document)

The documents listed in Table 1 are also required and may be accessed as described in Section 1.3.1.

Table 1. Security Documents

Document Name	Number
Intel® EP80579 Software for Security Applications on Intel® QuickAssist Technology for Linux* Getting Started Guide	320182
Intel® EP80579 Software for Security Applications on Intel® QuickAssist Technology Programmer’s Guide	320183
Intel® EP80579 Software for Security Applications on Intel® QuickAssist Technology Cryptographic API Reference Manual	320184

1.3.3 Documentation Conventions

The following conventions are used in this manual:

- `Courier font` - commands and code examples

1.4 Software Requirements

- Kernel: GNU*/Linux* 2.6.28-10
- Software: Intel® EP80579 Software for Security Applications on Intel® QuickAssist Technology Package Version 1.0.3
- Operating system: CentOS* 5.2 32-bit version
- IPSec stack for validation: strongSwan* 4.3.6 http://www.strongswan.org/

Note: Before installing the Linux* Kernel Crypto Driver, you must first install and configure the EP80579 security software, Package Version 1.0.3. For detailed instructions, see the Intel® EP80579 Software for Security Applications on Intel® QuickAssist Technology for Linux* Getting Started Guide.

2.0 Unpacking and Building the Software

This chapter provides instructions for unpacking and building the Linux* Kernel Cryptographic Driver for Intel® QuickAssist Technology.

Note: System commands given in this chapter assume that the user is issuing commands from a bash shell. This is the default shell. Use the `echo $0` command to verify use of the bash shell or run `/bin/bash` to switch to the bash shell.

2.1 Environment Setup

The Linux* Kernel Crypto Driver comes in the form of a tarball. The package can be unpacked at any location on the system, but for the purposes of this User Guide, a recommendation is provided.

Create a directory to unpack the Linux* Kernel Crypto Driver and create an environment variable to refer to the directory:

```
mkdir /QAT_SOFTWARE
cd /QAT_SOFTWARE
export ICP_ROOT=$PWD
```

Make sure you have a symlink named linux to the right Linux kernel source tree:

```
cd /usr/src/kernels
ls -l
```

After entering the above commands, the following is displayed:
(date and permission settings may differ):

```
lrwxrwxrwx  1 root root 32 Jul 14  2009 linux -> /usr/src/kernels/linux-2.6.28.10
```

If the symlink is missing or the symlink is pointing to a different Linux kernel source tree, (re)create the symlink with the below command:

```
ln -s /usr/src/kernels/linux-2.6.28.10 linux
```

Note: If the symbolic link name "linux" already exists, you may have to remove it before creating a new one.

To be able to build the module, the Linux kernel source tree should be properly configured. Refer to the Linux kernel documentation for details, specifically documentation found in /usr/src/kernels/linux/README

2.2 Unpack Linux* Kernel Crypto Driver

The Linux* Kernel Cryptographic Driver for Intel® QuickAssist Technology comes in the form of a tarball. The package can be unpacked at any location on the system, but for the purposes of this Guide, a recommendation is provided. Transfer the tarball to the directory $ICP_ROOT using any preferred method, for example, a USB flash drive, CDROM or network transfer.

Unpack the tarball using the following commands:

```
cd $ICP_ROOT
tar -zxvf QAT_CRYPTO_DRIVER.L.0.1.0-14.tar.gz
export ICP_ENV_DIR=$ICP_ROOT/qat_sw_lib/sw/mk
```

Before building the Linux* Kernel Crypto Driver, you need to copy the Module.symvers file. (For details, refer to /usr/src/kernels/linux/Documentation/kbuild/modules.txt)

```
cd $ICP_ROOT
cp /EP805XX_release/Acceleration/library/icp_crypto/look_aside_crypto/src/
Module.symvers qat_sw_lib/sw/qat/shims/netkey/
```

Note: These commands assume that the EP805XX_release directory was configured as described in the Intel® EP80579 Software for Security Applications on Intel® QuickAssist Technology for Linux* Getting Started Guide.

2.3 Build Linux* Kernel Crypto Driver

The commands to build the module are:

```
cd qat_sw_lib/sw/qat/shims/netkey
make
```

The kernel module is called: qat_icp_netkey.ko

When the build is complete, the module is located at:

```
$ICP_ROOT/qat_sw_lib/sw/qat/shims/netkey/qat_icp_netkey.ko
```

2.4 Load Linux* Kernel Crypto Driver

The command to insert the module is:

```
insmod $ICP_ROOT/qat_sw_lib/sw/qat/shims/netkey/qat_icp_netkey.ko
```

Note: The order of operation is very important. You must load the Linux* Kernel Crypto Driver module before you start the IPSec tunnel, otherwise the default kernel crypto module will be used by the system.

All log messages from the module are dumped into the file /var/log/messages

When the module is loaded, the following is displayed:

```
3DES-HMAC_SHA1 loaded
AES-CBC-HMAC_SHA1 loaded
AES_GCM loaded
netkey_init: Intel ICP NetKey Loaded.
```

The Cryptographic API (also called LAC) module must be started first or errors will result. If the LAC module is not present, the following is displayed:

```
qat_icp_netkey: Unknown symbol cpaCySymSessionCtxGetSize
qat_icp_netkey: Unknown symbol cpaCySymPerformOp
qat_icp_netkey: Unknown symbol cpaCySymInitSession
qat_icp_netkey: Unknown symbol cpaCyBufferListGetMetaSize
qat_icp_netkey: Unknown symbol cpaCySymRemoveSession
qat_icp_netkey: Unknown symbol cpaCyStopInstance
qat_icp_netkey: Unknown symbol cpaCyStartInstance
```

When the Linux* Kernel Crypto Driver module is unloaded, the following is displayed:

```
netkey_exit: crypto_unregister_alg()
netkey_exit: Intel ICP NetKey Unloaded.
```

2.5 Exercise Linux* Kernel Crypto Driver

The Linux* Kernel Crypto Driver works below the IPSec stack. To validate that the installation completed successfully, you must install strongSwan on two machines, create an IPSec tunnel between the machines, and route some traffic through the tunnel. At least one side of the tunnel must load the Linux* Kernel Crypto Driver module into the kernel.

To verify that the Linux* Kernel Crypto Driver is called, one option is to build the module with the __DEBUG compile flag. This will enable all the debugging output to the file in /var/log/messages.

Note: Enabling debugging output will decrease the system performance, so this option should be used with caution.

You can also use the `lsmod` command to verify the Linux* Kernel Crypto Driver is called. When the module is not used, the number of processes using it is 0; when the module is used, the number of processes using the module will be incremented.

§ §