Modifying OCF for Efficient Load Sharing Between H/W & S/W Crypto Engine

Chauhan Avinashsingh S1 Arjav A.Bavarva2
1, 2 VLSI & Embedded Systems Design Department
1, 2 Gujarat Technological University, Ahmedabad, India

Abstract— The project aims at using cryptographic service framework mainly the OCF(OpenBSD cryptographic framework) for the Linux kernel to make efficient use of crypto hardware and Software. As we know that cryptographic mechanisms are an important security component of any operating system in securing the system itself and its communication paths. So I will introspect this in detail by carefully studying OCF and Cryptographic API which is a main part of Linux Kernel and has been initiated to deliver cryptographic functionality to the whole Kernel. Next, we will see the hardware support and the Cryptographic API, that provides an inside kernel API to cryptographic resources and a device interface for user-level access to hardware-accelerated cryptographic operations. Thus by knowing the architecture of cryptographic API a device driver can be implemented. The study involves the great understanding of Linux kernel its various APIs, services, system calls, device management and finally writing the block and character device driver. Finally, we consider load balancing between multiple crypto hardware and software implemented algorithms and a priority mechanism for selecting which implementations to use first i.e. in a efficient manner for this we will make us of OCF-Linux which is a port of OpenBSD's Cryptographic Framework (OCF) to Linux.

Keywords: OCF , linux , kernel, API, Device driver, load balancing, hardware accelerator

I. INTRODUCTION

With the increase in demand of embedded devices being connected to each other and to the internet services, the need for secure transmissions is increasing steadily. Not only industrial, telecom and network, but also consumer and automotive embedded devices get connected to networks. It is widely recognized that data security will play a central role in the design of future IT systems. Many of those IT applications will be realized as embedded systems which rely heavily on security mechanisms.[1] Examples include security for wireless phones, wireless computing, pay-TV, and copy protection schemes for audio/video consumer products and digital cinemas.

Establishing encrypted VPN connections with the help of IPsec, for example, puts huge loads on current embedded processors.[2] Also, technologies such as DRM (Digital Rights Management) cause the need for cryptographic operations.

Cryptographic transformations are a fundamental building block in many security applications and protocols. To improve performance, several vendors market hardware accelerator cards. However, until now no operating system provided a mechanism that allowed both uniform and efficient use of this new type of resource[3].

The problem with encryption algorithms on embedded processors is that nowadays processors are scalar and thus only perform poorly processing these algorithms. This certainly demands cryptography hardware. Cryptographic operations should be offloaded to this piece of hardware by the operating system.[2]

Today software implementation[4] of algorithms intended for efficient hardware implementation. To address this issue, vendors have been marketing hardware cryptographic accelerators that implement several cryptographic algorithms used by security protocols and applications. However, modern operating systems lack the necessary support to provide efficient access to such functionality to applications and the operating system itself through a uniform API that abstracts away device details. As a result, accelerators are often used directly through libraries linked with applications, typically requiring device-specific knowledge by the applications, and preventing the operating system itself from easily utilizing such hardware[2].

We will carefully study the OpenBSD Cryptographic Framework (OCF), a service virtualization layer[5] implemented inside the kernel, that provides uniform access to accelerator functionality by hiding device-specific details behind a carefully-designed API. The abstraction introduced allows us to easily support new hardware accelerators and enable applications to use any such accelerator without device-specific knowledge. Furthermore, this intermediate layer does not unduly impact performance, as is common when such abstractions are introduced. It offers great features such as load balancing across multiple accelerators, session migration, and algorithm chaining.

In this project work we used the specialized security framework[5] available open source OpenBSD Cryptographic Framework(OCF), a service virtualization layer is implemented inside the linux kernel.

II. OBJECTIVES

The objective of the proposed work is to study and analyze the OCF its services features and its support that its offer to different hardware crypto devices at present. Detail scope of this work includes:

- Study the integration of hardware and software crypto devices through OCF inside kernel.
- Writing driver for hardware crypto device.
- Modifying OCF to have efficient load-balancing across multiple accelerators including hardware and software both.

III. PROJECT BLOCK DIAGRAM

The Project implementation block diagram is shown which shows user space and kernel space inside the kernel. OCF
The framework is implemented inside kernel space which coordinated the devices (crypto h/w) in the hardware abstraction layer & s/w crypto engine is in kernel space.

Fig. 1: Project Block Diagram

IV. TOOLS USED FOR DEVELOPMENT

A. Hardware

1. Am335x series Board.
2. PC with Linux supported OS (my case ubuntu 12.0.4).
3. Sd card 4gb with reader.

B. Software

1. VI, gedit and GCC.
2. Source Insight.

V. API’S & MODULE USED IN PROJECT DEVELOPMENT

A. Crypto API (Linux)

It is a cryptography framework in the Linux kernel, for various parts of the kernel that deal with cryptography, such as IPsec and dm-crypt. It was introduced in kernel version 2.5.45 and has since expanded to include essentially all popular block ciphers and hash functions.

B. Cryptodev-linux module

Cryptodev-linux is a device that allows access to Linux kernel cryptographic drivers; thus allowing of userspace applications to take advantage of hardware accelerators. Cryptodev-linux is implemented as a standalone module that requires no dependencies other than a stock linux kernel. Its API is compatible with OpenBSD's cryptodev userspace API (/dev/crypto).

C. OCF-Linux

OCF-Linux is a Linux port of the OpenBSD/FreeBSD Cryptographic Framework (OCF). This port aims to bring full asynchronous HW/SW crypto acceleration to the Linux kernel and applications running under Linux. Results have shown improvements of up to 7 times that of software crypto for bulk crypto throughput using OpenSSL.

At this point in time OCF-Linux provides acceleration for OpenSwan, OpenSSL, OpenSSH (scp, ssh, ...) and also supports the BSD crypto testing applications. It can accelerate DES, 3DES, AES, MD5, SHA, and Public Key operations and adds randomness to the kernels /dev/random by utilising the RNG hardware. This project is being actively developed as a high performance crypto solution for embedded devices but also applies equally well to any linux based server or desktop.

OCF-Linux was based directly on the FreeBSD port of the OCF framework. The FreeBSD versions contained several performance improvements that were desired. There is some good reference material on the changes and why they were needed that is worth reading for a background.

The current version supports most 2.4 and 2.6 kernels up to and including 2.4.35 and 2.6.39. It also supports all 3.x kernels.

VI. IMPLEMENTATION

OCF driver implementation is done by modifying the crypto code by adding specific hardware units and then writing the device driver.

Finally Verifying the driver with the help of am335x board or simply in pc via dumping the driver as an loadable module and seeing its performance fig 2 shows the flowchart sequential steps for implementation of the project.

Figure shows the basic implementation of Driver in this project. In this diagram,12C subsystem has the connection with rest of the all driver and subsystem. Each of the subsystem has the driver, this will help to communicate any of the data transfer to the user space and hardware level. All of them creates interface to communicate between the user space and kernel space[9].

There is very thick boundary between this two level i.e. User space and kernel space shown in figure. No one can enter from the user space to kernel space we have some interface which helps to data transfer from user space to kernel space. System call APIs read, write, open etc. All are used to build up the setup for this two level.

For writing any of the character drivers the following points are to be considered.

- Define the functions used in writing driver.
- Fill up the structure in file operation.
- Register the driver.

VII. RESULT AND DISCUSSION

A. Driver compilation and verification

![Driver compilation and verification](image)

Fig. 3: Driver compilation and verification

B. Final generated u-boot file after successful

![Final generated u-boot file after successful](image)

Fig. 4: Final generated u-boot file after successful

C. U-boot and x-loader final source

![U-boot and x-loader final source](image)

Fig. 5: U-boot and x-loader final source

D. Hardware setup and cross compilation

E. Generating config file and ocf setup

![Generating config file and ocf setup](image)

Fig. 6: Hardware setup and cross compilation

Fig. 7: Generating config file and ocf setup

VIII. CONCLUSIONS

Today generally the software crypto algorithm is used but by the result it is shown that hardware crypto accelerators are more fast, efficient, secure and low cpu power consuming, so designing Device driver and modifying in ocf makes a significant change in crypto processing task and load distribution mechanism between hardware and software.

IX. FUTURE WORK

There are plenty of directions that future work could take. First and foremost, it can come with new embedded devices emerging in market with additional crypto hardware instead of using software crypto alone.

Secondly from security point of view several new hardware crypto accelerators cards may emerge from different vendors as compare to very little vendor available presently.

REFERENCES


[8] Nikos Mavrogiannopoulos, Miloslav Trmac and Bart Preneel, "A Linux kernel cryptographic framework Decoupling cryptographic keys from applications"
SAC’12 Riva del Garda, Italy March, 2012.