CSCE 3402 Spring 2021
Exercise 9 – The Cipher Character Device
Kernel Module as a Cipher Box

Assigned: Tuesday, April 6th in Lab

Due: Tuesday, April 20th at Lab time
Delayed submission with penalty until Thursday, April 22nd at 11:55pm.

Goals
The goal of this lab exercise is to build a new kernel module that you can utilize in it all what you have learned so far and to make use of the Linux Kernel Character device interface to build an encrypted device that can store a message in an encrypted representation whose original representation should not be retrieved without a special secret key. What you will be building in this lab exercise is a simple device driver for a virtual device, virtual encrypted box.

Details
This lab exercise is an individual exercise that you need to carry out on your own. You are required to create two-character devices with the same major number; the major number is used to identify that the devices sharing it are managed by the same driver and hence the same kernel module. You should detect an available major number by utilizing the Linux Kernel Character Device interface APIs conveniently. You should assign the two-character devices different minor numbers for your own identification within your kernel module if needed. You should also create the corresponding character devices filesystem entry point nodes under **"/dev"** using the shell command line program **"mknod"** and give them the right ownerships and privilege modes; please man it and learn how to use it :) The names of your filesystem character device nodes should be **"/dev/cipher"** and **"/dev/cipher_key"**.

The role of those two-character devices is to allow the user to store a message in an encrypted representation inside the kernel. For the sake of simplicity, the size of the message should not exceed 4096 bytes; 4 KB, and the key should not exceed 128 Bytes. You should build the correct file operations data structures for both devices and implement the essential 4 filesystem routines: open, read, write, and release. Allocate a character device range, register your driver devices, and add them to the Linux Kernel Character device backbone through using the cdev APIs: cdev_init, cdev_add, and cdev_del, … etc. Essentially, the user should start by storing her/his key into the **"/dev/cipher_key"** device. Upon writing a message to the **"/dev/cipher"** device it should be encrypted by the key in **"/dev/cipher_key"** and stored in an encrypted format. Upon reading from **"/dev/cipher"** the return should be gibberish as it should return the encrypted representation. Of course, your module should be very keen not to give back to

anyone the stored key in **"/dev/cipher_key"** so upon reading from this device file your read routine should return to the reading process a very offending message:)

For example, if we execute the following sequence it should look like that:

```
root@csce-345-lab:~# echo "THISISMYCIPHERKEY" > /dev/cipher_key
root@csce-345-lab:~# cat /dev/cipher_key
Go away silly one, you cannot see my key >-:
root@csce-345-lab:~# cat /etc/hosts > /dev/cipher
root@csce-345-lab:~# cat /dev/cipher
?D????.
??>a?/?}4v?}KK?????6Qs8m?????^?S.?v?$????j?L?56\:X???1??ZE?.??`?fk?&Ѓ
???]???(*?N?
????l]s??:????\??:?g`3G???UsTS?A???/?P?Y??
?TƆ??`??{{?a?8s?"?W?<??E??9???1Z??k?(?=??Ru????X9??+g??`
root@csce-345-lab:~#
```

In your module, you should encrypt you message with the **RC4 stream cipher**, and you will be provided with its source code to use it.

Now, we stored an encrypted message into the character device, and it is stored in an encrypted format, what shall we do with it? Of course, we store data to be able to retrieve it later. For that you will create 2 **procfs** entries, with the same names as the character devices you created above but located under the **procfs**: **"/proc/cipher"** and **"/proc/cipher_key"**. A user who wants to read the stored encrypted message should have the valid key. The user should write the key into **"/proc/cipher_key"** and should be able to retrieve the original message upon reading from **"/proc/cipher"**. Of course, if the user provided a wrong key she/he should see gibberish stuff. It is very important that the user cannot read back the key from **"/proc/cipher_key"** and write should be disabled to **"/proc/cipher"**.

Finally, you should do all kinds of validations and housekeeping, including destroying all the stored data, messages and keys, and removing all the character devices gracefully upon exiting your module.

Bonus

You are eligible for 2% bonus (of the overall course grade) if you can perform encryption and decryption using the famous AES algorithm with the X86 ISA supported extension. I will upload a sample code for you, but you need to make it work :) **If you decide to do the bonus, you still have to submit the original version of the kernel module with RC4 or else you will lose marks.**

What to submit

1. All the C code you wrote for the version kernel module.
2. You Makefile.

3. A small readme file explaining how to use your make files to compile the programs.

How to submit:

Compress all your work: source code, report, readme file, and any extra information into a zip archive. You should name your archive in the specific format <Student_ID>_<Name>_Lab9.zip. Finally, upload your code to blackboard.

Grade

This Lab exercise is worth 20 % of the overall course grade. The exercise will be graded on a 100% grade scale, and then will be scaled down to the 20% its worth. The grading of the assignment will be broken down as follows:

1. 10 % for just submitting a meaningful assignment before or on the due date. This 10% does not account for the correctness of your assignment but submitting an empty assignment without code will definitely results in loosing this 10% and consequently the whole grade of this assignment.
2. 80 % for the correctness and the quality of the submitted code and make files.
3. 10 % readme file.

Delays

You have up to 2 working days of delay, after which the assignment will not be accepted and your grade in that case will be ZERO. For every day (of the 2 allowed days), a penalty of 10% will be deducted from the grade. And of course you will lose the 10% mentioned in point 1 above under the “Grade” section.