Assignment #2: Encrypted “Pipe” Pseudo-Device Driver

Ilya Samoylov, Matt Skrobola, Joe Fela

**Overview**

* cryptctl.c acts as the control device driver
* it holds an array of structs stored in kernal memory where each struct represents an encryption/decryption character driver.
* the main control driver has ioctl functions that add/remove/ect
* user app is interface.c

**How to setup**

* make
* ./interface create -key
* ./interface delete -index
* ./interface encrypt -index -input
* ./interface decrypt -index -input
* ./interface change\_key -newkey -index
* ./interface getkey -index
* run cat /proc/devices to see if cryptctrl mapped to the right major number
* you can check if the character drivers are in /dev after creating/deleting
* You can check if that module is loaded by running lsmod
* Major number is hardcoded for the control driver to 240

**Difficulties**

* Accessing VM’s
  + An initial problem some of use encountered were logging into the VM. Once we all realized we needed to be on Rutgers vpn in order to access the class vm, this issue was resolved.
* What to do when vm crashes
  + Before we found out we needed to contact the cs help line, we were confused as to how we reset our crashed vm.
* Crashing the vm
  + We encountered a bug where when trying to free, it would crash the vm. We fixed this by realizing we didn’t write the key for kernel storage, which was causing the crash.
* Command Line Utility
  + Figuring out how to implement the command line utility caused some setbacks. We were confused since the program would be run on a different virtual machine how we would be able to edit the directories for that virtual machine’s path. In the end, we discovered that just creating a file sperate from our cyptctl file, we could use this file to implement aspects of cryptctl in the command line by just calling “./interface” then the command we wished to call, and that the path directory did not need to be edited.
* Using Vigenere Cipher
  + We had never worked with Vigenere’s Cipher before, we have learned about it, but never implemented it before, so this caused a little learning curve. We also found that our implementation only worked for uppercase letters, which lead to our next issue.
* Upper and Lowercase letters
  + Some confusion lead to us having difficulties with encrypting and decrypting upper vs lowercase letters. At first we thought we would just set everything to upper or lowercase, but we realized this would result in wrong decrypting. We solved this issue by modifying how we encrypted, by allowing it to accept both lower and uppercase letters.
* Deciding to not include symbols
  + We realized we ran into an issue when trying to pass symbols through the encryption algorithm. We decided it would be easier to just skip symbols and only focus on upper and lower case letters.

**Flow**

* We created a “control device” file which is located in /dev/cryptctl. When this cryptctl file creates from ioctl(), it first creates an encrypt/decrypt fil pair, named cryptEncryptXX and cryptDecrytpXX, where “XX” stands for the ID for the pair. This is done in the function construct\_crypt\_pair in cryptctl.c
* The Vignere cipher is applied to text which is part of the input for the encryption aspect by combining the key with the text phrase to be encrypted, and decrypted in the same manor. This aspect can be seen fully in the functions encryptString and decryptString in the interface.c file.
* Once the file is encrypted, it is written to the cryptEncryptXX file created in step 1.
* In order for the control device to work, it needs to be registered. This process is done in the function init\_module, where the major and minor number are assigned for the device number.
* In order to meet the command line utility requirement, as discussed in the “Difficulties” section above, we needed to create another file, which is our interface.c file. This file makes the ioctl calls to the control device in order to create the encryption decryption pairs, remove pairs, get the key, change the key, as well as encrypt and decrypt strings in the command line.