OS ASSIGNMENT 4

Shagun Uppal - 2016088
Niket Singh - 2016060

Encryption Device

In this part of the assignment, we need to make an encryption device which takes in input and encrypt it using an encryption key. The encryption key is generated by using Random Bytes from /dev/urandom. Once the 16 byte encryption key is generated, the file to be encrypted is read in chunks of 16 bytes. Firstly, the first 16 bytes of the encrypted text are the randomly generated key itself. Further, the first 16 bytes read from the file to be encrypted are XORed with the 16 byte key giving the first 16 bytes of the encrypted text. This 16 byte encrypted text becomes the key for the next 16 bytes (128 bits) to be encrypted, which is further encrypted with its corresponding key and similarly, further each time the encrypted text for the previous block becomes the key for the next block until we reach the end of the file. In case, the end block read is not complete 16 bytes, then it is padded with a special character '@' in order to uniformly obtain 16 byte block from it. This leads to an encrypted text of the given file.

Decryption Device

In this part of the assignment, we need to make a decryption device which takes in the encrypted file, the one obtained by the encryption device in the above part and then decrypts it using the same key. For this, it reads the encrypted file again in chunks of 16 bytes at a time. The first 16 bytes read are XORed with the original shared key. Further, each chunk of 16 bytes read from the encrypted file acts as a key for the next 16 bytes read from the encrypted file in order to decrypt it and retain the original contents of the initial unencrypted file.

Commands Used

In order to make a new device on the system, we used the following commands :

```
# mknod /dev/device_name c major_number minor_number
# chmod a+r+w /dev/device_name
# make
# insmod /dev/deivce_name
```

This will create a new device in the system with the given device name and the respective major number, also it gives the required permissions to our device. Further, once the compilation is done, the compiled module is loaded onto the kernel.

Compiling the test program

For compiling the test program for the following devices :

Encryption Device and Decryption Device :
gcc test.c some.txt

Inputs and Expected Outputs

Given any input text file, the encdev first encrypts the given text file and then the decryption device decrypts the encrypted data and writes it back to the user space.

Error Values Handled

The 2 errors handled are as follows :

1) **Device Registration :**

In case, if the desired device registration fails, the kernel prints an alert :
Device Registration Failed

2) **Copy From User :**

While copying the data of the encrypted file from buffer to kernel memory, we use copy_from_user command. In case, if it throws an error due to invalid buffer or kernel memory locations, it throws an error.

3) Opening / Closing the device :

Opening and Closing the device throws an error, if device not created properly.