Operating Systems

Assignment 4

Aarushi Agarwal 2016216
Surabhi S Nath 2016271

Encryption Decryption Devices

Description of your code and how you implemented the function – the logical and implementation details

Our task was to build an encryption and a decryption device which given a cryptographic key, encodes an input into encrypted form and subsequently decrypts it back into original form.

We have 2 device drivers one for Encryption and one for Decryption. These were created using mknod. Hence we have 2 files- Encrypt.c and Decrypt.c wherein each file contains module initialization which registers the device by calling register_chrdev(), module cleanup which unregisters the device by calling unregister_chrdev(). 4 file operations are supported for both devices: open(), read(), write() and release() or close().

We also have a Userspace.c which calls open() of both files and creates a random cryptographic key. The key is generated by opening the file /dev/urandom and reading 16 random chars using fgets from that file.
We also take in a user input which we need to encrypt. The input is split in chunks of 16 bytes or 128 bits and transferred from Userspace.c to the write function of Encrypt.c

The write() function for the encryption device takes is initially sent the key which is stored in the first row of a 2D array. Next it is sent input data in chunks of 128 bits which is xor-ed with the key and stored in the second row of the 2D array. Subsequently, more chunks are sent across which are xor-ed with the last row and stored in the next row. This way, the entire message is encrypted. The read() function reads the encrypted input.

Decrypt.c works in a similar manner and does the reverse operation. Given the encrypted input to write() of decrypt, it uses the same key and the 2D array to stepwise

recreate the original input and stores it in a 2D array. Read() of decrypt reads the original string from the array. This way we re-obtain our input string.

Description of how to compile and test the program

We use kernel makefile to compile Linux modules. Kbuild mechanism is used for compiling the kernel. The result of the compilation process is a .ko file. Thus 2 .ko files are obtained after make - Encrypt.ko and Decrypt.ko. The Userspace.c is compiled normally using gcc. The makefile contains the compilation of all these 3 files.

To test the program, the user can give any string input in the file. The encrypted form of the input and the decrypted form will be displayed. The original input would the same as the decrypted form. The decrypted version is also written into another file. This verifies that our devices are working correctly.

The inputs the user should give

The user needs to give in any string input which he wished to encrypt. However, here, the input is read from a file - Encryptfile.txt and written into a file - Decryptfile.txt.

Expected output (and how to interpret it)

We print the following information:

- The key generated
- The input string
- The encrypted string
- The decrypted string

The random key would contain random characters. The decrypted string should be the input string.

An example is shown below:

Error values and how to interpret them

We have handled the following errors:

- If the file we are trying to open does not exist, we have displayed an error. ENOENT is displayed and 2 is returned.
- If read() function returns -1, we have displayed an error
- We have printed a kernel alert if the device registration fails