Project #2: Secure Vault Design as a Linux Device Driver

Istanbul Technical University,
Faculty of Computer and Informatics
BLG413E- System Programming
2020-2021 Fall

You should develop a Linux kernel module that implements secure vault as Linux device driver and the driver will control a vault device (**/dev/vault**).

In addition, you should also implement *read* and *write* system calls for your module. ***write()*** call basically takes an input text and stores it to the buffer on a device after encrypting it by using the active permutation key stored on the module. Meanwhile, ***read()*** call will return the content of the buffer on a device, on which it is called, after decrypting it. Both of these calls use the same permutation key defined on the module for encryption and decryption. This permutation key is set and updated by a specific ***ioctl()*** call, which its related details are given in the next section.

This vault device should follow the permutation cipher method[1] to encrypt and to decrypt the data. An encryption example is given as follows:

- Assume that your encryption alphabet is "*abcdefghijklmnopqrstuvwxyz*", and your encryption key is "*ceayf*". According to order of key letters in your encryption alphabet, your permutation function is determined as follows:

$$\left.\begin{array}{rccccc} characters: & c & e & a & y & f \\ orders\ in\ alphabet: & 3 & 5 & 1 & 25 & 6 \\ labels\ based\ on\ alphabetical\ order: & 2 & 3 & 1 & 5 & 4 \end{array}\right\} \Longrightarrow (2,3,1,5,4) \Longrightarrow C = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 1 & 5 & 4 \end{pmatrix}$$

- The following steps are going to be taken, if the given text "*oneringtorulethemall*" is encrypted using permutation function above:

  i) Dissect the text into substrings by the size of the permutation function's domain set.

$$\begin{pmatrix} o & n & e & r & i \\ n & g & t & o & r \\ u & l & e & t & h \\ e & m & a & l & l \end{pmatrix}$$

  ii) Apply permutation function to each substring/line.

$$\begin{pmatrix} n & e & o & i & r \\ g & t & n & r & o \\ l & e & u & h & t \\ m & a & e & l & l \end{pmatrix}$$

  iii) Concatenate the substrings in order to get the final encrypted text "*neoirgthnroleuhtmaell*".

Decryption will follow the same steps only with the inverse of the permutation key (C^{-1})

[1] https://crypto.interactive-maths.com/permutation-cipher.html

Implementation Details

- Your alphabet is small English lower case letters: "*abcdefghijklmnopqrstuvwxyz*".
- In addition, your initial key value should be "*abcd*".
- Write operation on a device will take the information, stores encrypted text, and encryption is performed using the current key. Write operation on a **device must return** an error code if another write operation is performed before reading (you cannot write without reading previous input).
- Read operation returns the decrypted text, and decryption is performed using the current key. Read operation on a device **must return an error code** if there is not a write operation is performed before (you cannot read from an empty device). If key length is not an exact divisor of the input string length, fill the remaining letters with "0".

- Your module needs to have two *ioctl* commands:
    1. Command to change key (All devices share the same key, so changing operation from a device affects all others).
    2. Command to clear the content (stored string) inside vault/device.

Important Note: If you start coding from a provided scull module (or some other module code), make sure you remove parts unnecessary for this assignment. Also for storing device-specific data, follow the same practices as the scull module.

Submission Details

Deadline: Jan 4th, 2021

- Every group member is required to submit source code file(s) through the Ninova system as a zip file.
- Any form of cheating or plagiarism will not be tolerated. This includes actions such as, but not limited to, submitting the work of others as one's own (even if in part and even with modifications) and copy/pasting from other resources (even when attributed). Serious offenses will be reported to the administration for disciplinary measures.
- You must write a small test program that showcases how your ioctl commands are working.