PA-1 CS730

Name: Chinmay Goyal (180206)

Design

- The Cryptcard device is recognised using the product id and the vendor id
- For multiprocess implementation of cryptocard, a spin lock was taken in the kernel space which allows only 1 process to access the device at a time.
- Functions like set_config and set_key cannot be allowed to also require lock because the configuration of the key and the mode of operation is stored in the private field of the file pointer which can be shared by multiple processes.
- A sysfs variable has been created in the `/sys/kernel/cryptcard/` folder with the name `command`.
- The sysfs variable is write only and can be used to set the default global configuration
  - The structure of the variable is as follows:
    - a - first 8 bits
    - b - next 8 bits
    - interrupt value - next bit
    - mmio - next bit
  - In total 18 bits can be used in the command file.

Implementation

Crypter.C

initialize_crypt

- Function to initialize crypt_data struct.

create_handle

- The character device initialized in cryptocard_mod is opened as a file, and the correpsonding file descriptor is returned

close_handle

- The file descriptor of the file is closed

encrypt

- `write` syscall is called through this method.
- crypt_data struct has the information regarding the operation type and the address of the data that needs to be encrypted.
- The pointer to the crypt_data struct is passed as the buffer of the write syscall.

decrypt

- Similar to `encrypt`, only the operation value is changed in crypt_data struct

set_key

- The value corresponding to the key is stored in the file_pvt struct in the kernel code.
- In the library implementation, crypt_data struct is used to pass data to kernel space

set_config

- Similar to `set_key`

map_card

- This function calls mmap and provides a chunk of virtual memory to the process.
- A factor of `0xa8` is used which is for the offset of the data region in the device.

unmap_card

- This function calls munmap and unmaps the mapped memory region.

cryptocard_mod.c

struct crypt_data

- This data structure is used to transfer all the operational data from user space to kernel space
- Members of this struct are:
    - address: The virtual address pointing to the data to encrypt
    - length: Size of data to encrypt
    - operation: All the library space functions call the syscall `write`. This variable distinguishes between the different types of operation
    - a: Value of key a
    - b: Value of key b
    - interrupt: Interrupt mode is on/off
    - mmio: Mode is MMIO or DMA

struct file_pvt

- This data structure is used to store the configuration of an open file pointer
- Members of this struct are:
    - a: Value of key a
    - b: Value of key b
    - interrupt: Interrupt mode is on/off
    - mmio: Mode is MMIO or DMA

ioread64

- Function to read 64 bits, similar to `ioread32`

iowrite64

- Function to write 64 bits, similar to `iowrite32`

set_a_b

- Function to set the key

__encrypt

- This function encrypts/decrypts the required data.
- The mode of encryption/decryption is decided through the values stored in struct file_pvt associated with the corresponding Device Handle, which is passed as an argument to the function.
- Firstly, the value of the status register of the corresponding mode is checked to ensure that the device is free
- The required values are written to the corresponding registers
- The trigger bit is set, if interrupt mode is on, then the current process is put in the wait queue which waits for the interrupt to become active
- If the interrupt mode is of then a while loop continuosly checks for the status register.

private_data_init

- Function to initialize struct file_pvt to default values.

private_data_release

- Free the memory held by struct file_pvt of the file

demo_open, demo_release and demo_read

- These functions are taken from Kernel Workshop demo code provided in the class

demo_write

- Function to perform the required operation.
- In case of encryption/decryption this function handles the cases when the size of data to encrypt/decrypt is more than the I/O capacity of the device in each of the two modes.

demo_mmap

- Function to remap the user space virtual address to the device memory.

struct file_operations fops

- Taken from Kernel Workshop with minor modifications

memtrack_command_show and memtrack_command_set

- These functions are taken from Kernel Workshop
- The command variable is Write Only, hence `memtrack_command_show` is just an empty function and does not do anything
- `memtrack_command_set` sets the value of the default configuration of the device.

init_char_dev and cleanup_char_dev

- Functions to initialize and cleanup the char_dev
- Taken from Kernel Workshop code with minor modifications

irq_handler

- Handle the interrupt generated by the device.
- Writes value present in ISR to interrupt ACK register in order to stop generating interrupts.

set_interrupts

- Function to set the interrupt handler

crypt_probe

- This is the function called when the driver is matched with the device.
- This inspiration for this function was taken from Oleg Kutkov Blog

crypt_remove

- Function called when the device being pointed to by the device is disconnected from the driver
- Insipration taken from Oleg Kutkov Blog

Benchmarking Result

- Benchmark results have been added in the folder `bechmark-results`
- The benchmarking was done with file size of 10MB

command	%usr	%system	%wait	%CPU
MMAP	46.75	4.75	0.42	51.50
MMAP_Interrupt	75.17	6	0.17	81.17
MMIO	0.06	81.97	0	82.93
MMIO_Interrupt	0.04	94.65	0.42	94.69
DMA	0.01	6.7	0.24	6.71
DMA_Interrupt	0.01	0.21	0	0.22

Testing

- Testing was done through the eval-tests given.
- Some intensive testing was done by modifying the benchmark programs to check for correctness as well.

Acknowledgements

- The code for the pci driver was inspired a lot from Oleg Kutkov Blog
- A lot of theoritical doubts were cleared from LDD Book
- DMAs' inspiration was taken from here
- Existing linux functions and structures were understood from Elixir Bootlin Website
- The character device setup was inspired from the class code used during the Kernel Workshop