

Relatório Sistemas Operacionais B

Projeto 1: Crypto Device Driver

Gabriel Gonçalves Mattos Santini RA: 18189084,

Luiz Vinícius dos Santos Ruoso RA: 18233486,

Marcelo Germani Olmos RA: 18048298,

Victor Felipe dos Santos RA: 18117820,

Victor Luiz Fraga Soldera RA: 18045674.

1

***Resumo.** Em sistemas baseados no modelo de kernel proposto por Linus Torvalds, o Linux, é evidenciado sua modularização, construção na linguagem C, e sua comunicação entre kernel e driver através de uma API (Interface de Programação de Aplicativos, em português). Neste projeto é proposto a implementação de um módulo de kernel (v 4.15.0, disponível kernel.org) para Criptografia (em AES - CBC), Descritografia (em AES - CBC) e Resumo Criptográfico (HASH, em SHA1). Toda a comunicação é baseada a partir do conceito de "CHARACTER DEVICE".*

1. Introdução

Nesse projeto, é proposta a implementação de um módulo de Kernel, o qual deve criar um *CHARACTER DEVICE*, onde podemos realizar a comunicação de um programa a nível de usuário com um driver, a nível de kernel. Nesse caso, o presente desenvolvimento foca na criação desse módulo de kernel para realizar: **Encriptação, Decriptação e Hash** (resumo criptográfico), respectivamente em AES - CBC (128 bits), para os dois primeiros e, SHA1 para o último.

2. Desenvolvimento

2.1. Base

Para iniciar o projeto, foi criado um programa base e definida uma arquitetura para o projeto, onde cada parte, i.e. Encriptação, Decriptação e Hash, foi criada como um arquivo separado e todos eles eram ligados como um só módulo utilizando o Makefile.

O módulo de base cria o dispositivo que serve como interface com programas externos, além de definir as formas com que o mesmo reagiria as chamadas de open, close, read e write, seguindo como base o programa introduzido em [Malloy 2015] e criando as chamadas dos módulos baseadas em [Mueller and Vasut 2020].

2.1.1. init

Esta função é chamada quando o módulo é inicializado. Ela cria o dispositivo de interface e prepara tudo para as funções posteriores.

2.1.2. exit

Chamada quando o módulo é retirado do Kernel, esta função libera a memória alocada para o módulo e retira o dispositivo de interface.

2.1.3. open

Quando um programa abre o arquivo do dispositivo, este tranca um mutex, impedindo que o dispositivo seja acessado por mais de um programa ao mesmo tempo, o que elimina race conditions.

2.1.4. release

Quando um programa fecha o arquivo, o mutex é destrancado, permitindo que outro programa acesse o dispositivo.

2.1.5. write

No momento em que um programa escreve no dispositivo, esta função recebe a mensagem escrita no dispositivo e copia ela para um vetor de caracteres e salva o tamanho da mensagem.

2.1.6. read

Ao ser requisitada a leitura do dispositivo, este define qual função deve ser chamada, segundo o primeiro caractere da string recebida, e chama a função, enviando como parâmetros a string que deve ser utilizada, o tamanho da string, a chave criptográfica e o vetor de inicialização.

2.2. Encriptação

Acessando o site: kernel.org, na seção de exemplos de código, há base para criação da cifra de forma assíncrona, o usual para comunicação entre o módulo de kernel e o kernel através da API.

2.2.1. *void encrypt(char * string, int size_of_string , char * localKey, char * iv)*

Nessa função recebemos a partir da base, com o uso do conceito de *CHARACTER DEVICE* e modularização de software adotada na idealização do projeto, aqui é recebido a *string* que será criptografada, o tamanho da mesma em *size_of_string*, a chave em caracteres em *localKey* e o *iv*, que para o método AES - CBC é requisito base.

2.2.2. *static int test_skcipher(char msgToEncrypt[], char keyFromUser[], char ivFromUser[])*

Em *test_skcipher*, é iniciado o gatilho e variáveis, nos tipos especificados pela API para encriptação assíncrona. Nessa função recebemos novamente os dados chegados pela *Base*, e depois é copiado para *scrachpad*, a frase para encriptação, essa mesma é passada para a função *sg_init_one*, para inicialização com 16, que é o *AES BLOCK SIZE*. Esse conteúdo inicializado e alocado, é passado para *sk.sg* onde é encriptado e retornado para *resultdata* e o mesmo é copiado para o endereço recebido através de *msgToEncrypt*.

2.2.3. *static unsigned int test_skcipher_encdec(struct skcipher_def *sk, int enc)*

Em *test_skcipher_encdec* é acionado o gatilho assíncrono, que será passado para a função anteriormente descrita. A mesma recebe variáveis de tipos específicos para criação da requisição e espera do retorno.

2.3. Decriptação

Usando a mesma base da encriptação, recebe-se a string encriptada no formato hexadecimal, e, para poder haver a devida decriptação, deve-se converter cada byte hexadecimal (2 caracteres Hexadecimais, no estilo "a8", por exemplo) em um char da tabela ASCII. Para isso, criou-se uma função chamada *hex_to_ascii*. Em seguida, alterou-se a chamada de função *test_skcipher_encdec(&sk, 0)*; onde o segundo parâmetro passado (0) é a chamada da função de decriptação da API Crypto do Kernel. Então a string resultante é a string decriptada, mas em formato hexadecimal. Para haver a conversão, converte-se cada caracteres hexadecimal em ASCII novamente.

2.4. Hash

Depois de encriptar e decriptar, foi necessário criar o Resumo Criptográfico(HASH) dos dados fornecidos pelo usuário e formato escolhido para criar o resumo foi o SHA1(Secure Hash Algorithm). Para realizar o resumo, foi criada a função *hash*, que recebe uma string e após isto copia a o resumo criptográfico para vetor de caracteres.

2.4.1. *crypto_shash_init*

A função *crypto_shash_init* foi utilizada para configurar o tipo de resumo que seria feito, através da variável *desc*.

2.4.2. *crypto_shash_update*

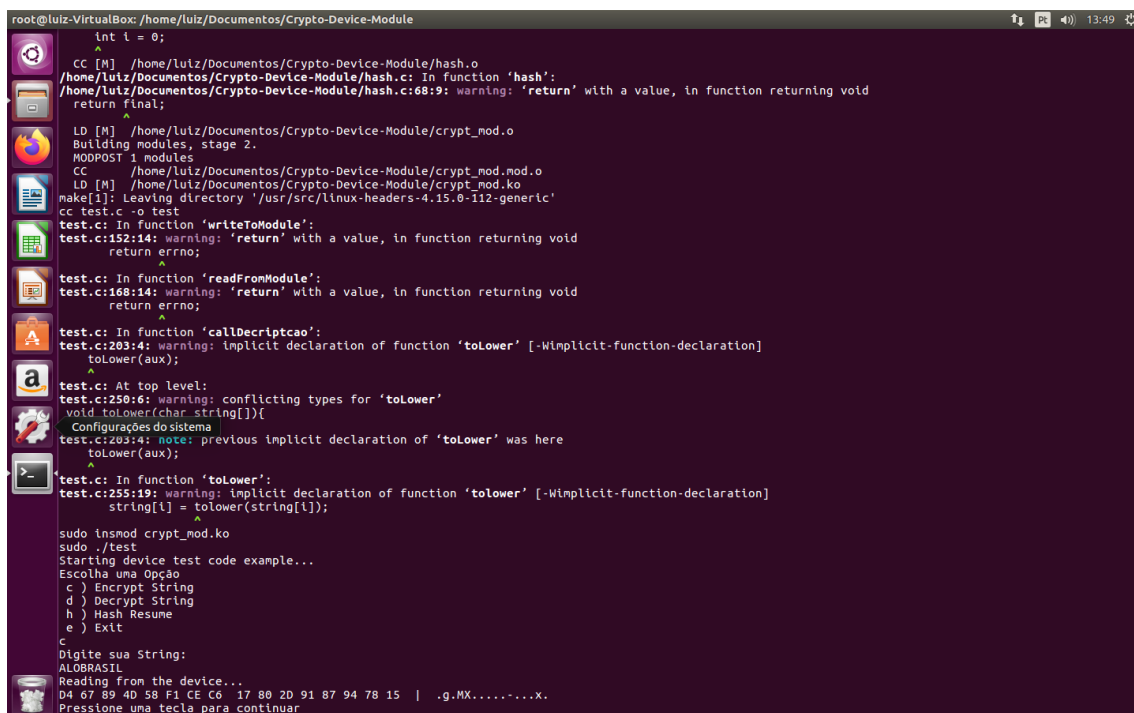
A função *crypto_shash_update* realiza o resumo criptográfico na string.

2.4.3. *crypto_shash_final*

A função *crypto_shash_final* envia o resumo criptográfico para a variável *hashval*.

3. Testes

Após a criação dos módulos, foram realizados testes, de forma a encontrar problemas e solucioná-los. Para isso, decidimos uma string, "ALOBRAASIL", e a submetemos à função de encriptação. Durante a construção da deciptação, percebeu-se a não deciptação da string apenas usando a função *testskcipherencdec()*, então, para fazer a verificação, utilizamos o site <http://www.cryptogrium.com/aes-cbc.html> para confirmar que a string era enviada em formato hexadecimal, e, então, para haver a deciptação tem-se de se converter cada byte hexadecimal em uma caractere da tabela ASCII. Realizados esses testes conceituais de deciptação e encriptação, foi desenvolvido o módulo *test.c* que realizada a comunicação entre o usuário o módulo de *CHARACTER DEVICE* e os arquivos que contém as funções *Encrypt*, *Decrypt* e *Hash*. Das Figuras de número 1 a 4, de forma exemplificada há os resultados para cada função.



```
root@luiz-VirtualBox: /home/luiz/Documentos/Crypto-Device-Module
int i = 0;
^
CC [M] /home/luiz/Documentos/Crypto-Device-Module/hash.o
/home/luiz/Documentos/Crypto-Device-Module/hash.c: In function 'hash':
/home/luiz/Documentos/Crypto-Device-Module/hash.c:68:9: warning: 'return' with a value, in function returning void
return final;
^
LD [M] /home/luiz/Documentos/Crypto-Device-Module/crypt_mod.o
Building modules, stage 2.
MODPOST 1 modules
CC [M] /home/luiz/Documentos/Crypto-Device-Module/crypt_mod.mod.o
LD [M] /home/luiz/Documentos/Crypto-Device-Module/crypt_mod.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.15.0-112-generic'
cc test.c -o test
test.c: In function 'writeToModule':
test.c:152:14: warning: 'return' with a value, in function returning void
return errno;
^
test.c: In function 'readFromModule':
test.c:168:14: warning: 'return' with a value, in function returning void
return errno;
^
test.c: In function 'callDecryptao':
test.c:203:4: warning: implicit declaration of function 'toLower' [-Wimplicit-function-declaration]
toLower(aux);
^
test.c: At top level:
test.c:250:6: warning: conflicting types for 'toLower'
void toLower(char string[]){
^
Configurações do sistema
test.c:203:4: note: previous implicit declaration of 'toLower' was here
toLower(aux);
^
test.c: In function 'toLower':
test.c:255:19: warning: implicit declaration of function 'toLower' [-Wimplicit-function-declaration]
string[i] = toLower(string[i]);
^
sudo insmod crypt_mod.ko
sudo ./test
Starting device test code example...
Escolha uma Opção
c ) Encrypt String
d ) Decrypt String
h ) Hash Resume
e ) Exit
c
Digite sua String:
ALOBRAASIL
Reading from the device...
D4 67 89 4D 58 F1 CE C6 17 80 2D 91 87 94 78 15 | .g.MX.....x.
Pressione uma tecla para continuar
```

Figura 1. Encriptação

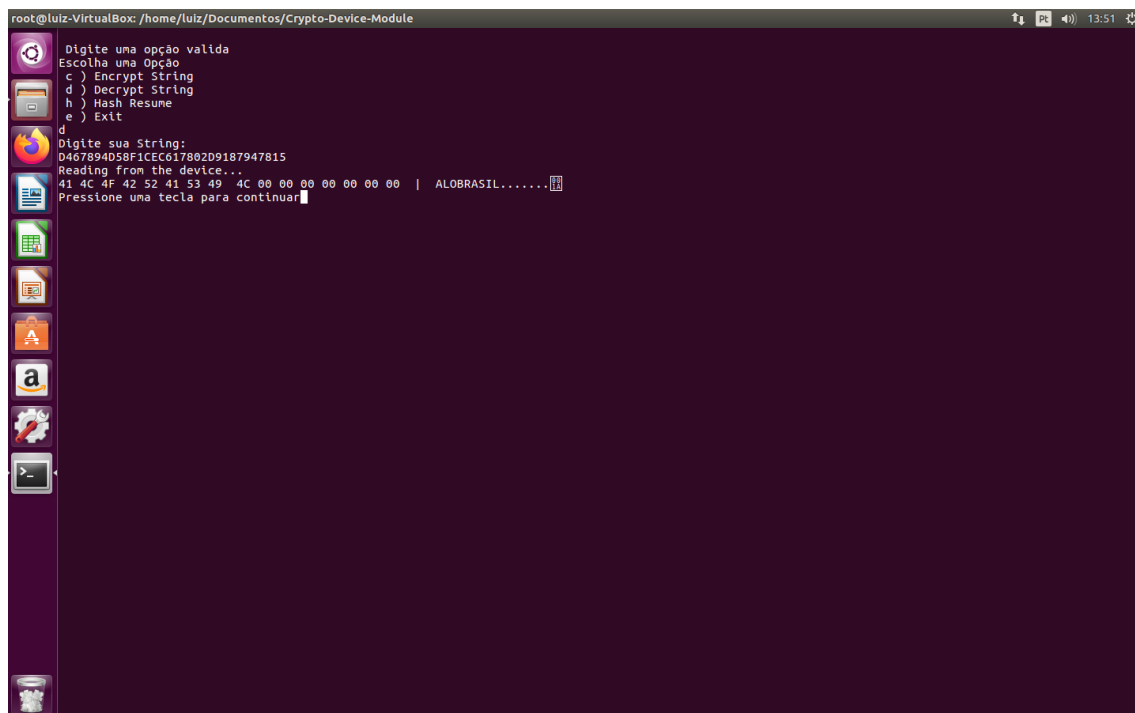


Figura 2. Decriptação

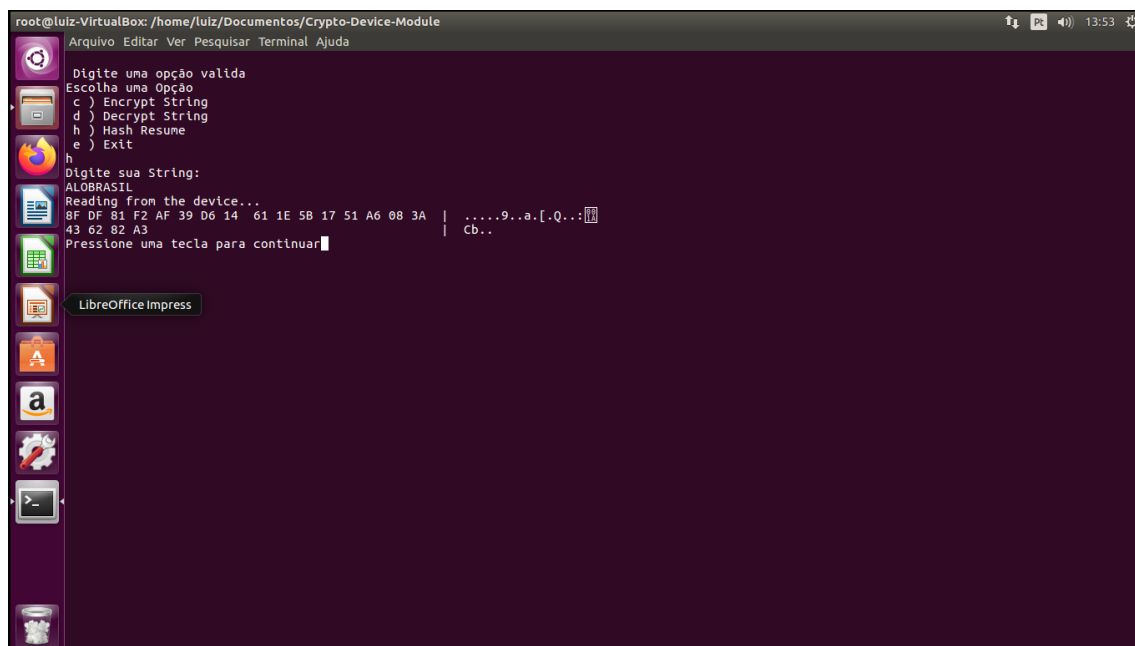


Figura 3. Hash

```
root@luiz-VirtualBox: /home/luiz/Documentos/Crypto-Device-Module
Arquivo Editor Ver Pesquisar Terminal Ajuda
CC /home/luiz/Documentos/Crypto-Device-Module/crypt_mod.mod.o
LD [M] /home/luiz/Documentos/Crypto-Device-Module/crypt_mod.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.15.0-112-generic'
cc test.c -o test
test.c: In function 'writeToModule':
test.c:152:14: warning: 'return' with a value, in function returning void
    return errno;
               ^
test.c: In function 'readFromModule':
test.c:168:14: warning: 'return' with a value, in function returning void
    return errno;
               ^
test.c: In function 'callDecryptcao':
test.c:203:4: warning: implicit declaration of function 'toLower' [-Wimplicit-function-declaration]
    toLower(aux);
    ^
test.c: At top level:
test.c:250:6: warning: conflicting types for 'toLower'
void toLower(char string[]){
    ^
test.c:203:4: note: previous implicit declaration of 'toLower' was here
    toLower(aux);
    ^
test.c: In function 'toLower':
test.c:255:19: warning: implicit declaration of function 'tolower' [-Wimplicit-function-declaration]
    string[i] = tolower(string[i]);
                  ^
Configurações do sistema
Starting device test code example...
Escolha uma Opção
c ) Encrypt String
d ) Decrypt String
h ) Hash Resume
e ) Exit
c
Digite sua String:
ALOBRASIL
Reading from the device...
D7 4B E4 A0 C6 AB EE BC 02 A3 A7 0D D2 F2 B3 DD | .K.....
Pressione uma tecla para continuar
```

Figura 4. JOURNALCTL exemplificando IV e KEY usados

Alterando a KEY e IV na instalação do módulo podemos alterar os resultados de encriptação e decrptação, das Figuras de número 5 a 8:

```
root@luiz-VirtualBox: /home/luiz/Documentos/Crypto-Device-Module
Arquivo Editor Ver Pesquisar Terminal Ajuda
CC /home/luiz/Documentos/Crypto-Device-Module/crypt_mod.mod.o
LD [M] /home/luiz/Documentos/Crypto-Device-Module/crypt_mod.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.15.0-112-generic'
cc test.c -o test
test.c: In function 'writeToModule':
test.c:152:14: warning: 'return' with a value, in function returning void
    return errno;
               ^
test.c: In function 'readFromModule':
test.c:168:14: warning: 'return' with a value, in function returning void
    return errno;
               ^
test.c: In function 'callDecryptcao':
test.c:203:4: warning: implicit declaration of function 'toLower' [-Wimplicit-function-declaration]
    toLower(aux);
    ^
test.c: At top level:
test.c:250:6: warning: conflicting types for 'toLower'
void toLower(char string[]){
    ^
test.c:203:4: note: previous implicit declaration of 'toLower' was here
    toLower(aux);
    ^
test.c: In function 'toLower':
test.c:255:19: warning: implicit declaration of function 'tolower' [-Wimplicit-function-declaration]
    string[i] = tolower(string[i]);
                  ^
Configurações do sistema
Starting device test code example...
Escolha uma Opção
c ) Encrypt String
d ) Decrypt String
h ) Hash Resume
e ) Exit
c
Digite sua String:
ALOBRASIL
Reading from the device...
D7 4B E4 A0 C6 AB EE BC 02 A3 A7 0D D2 F2 B3 DD | .K.....
Pressione uma tecla para continuar
```

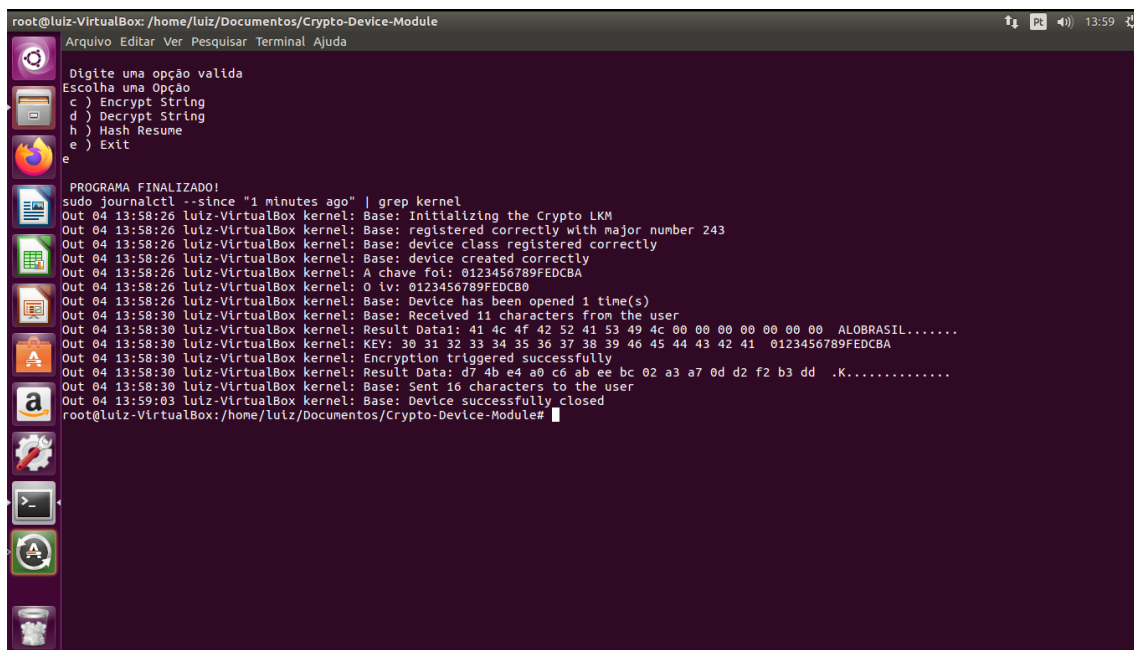
Figura 5. Encriptação com IV e KEY diferentes

```
root@luiz-VirtualBox: /home/luiz/Documentos/Crypto-Device-Module
Arquivo Editar Ver Pesquisar Terminal Ajuda
MODPOST 1 modules
CC      /home/luiz/Documentos/Crypto-Device-Module/crypt_mod.mod.o
LD [M]  /home/luiz/Documentos/Crypto-Device-Module/crypt_mod.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.15.0-112-generic'
cc test.c -o test
test.c: In function 'writeToModule':
test.c:152:14: warning: 'return' with a value, in function returning void
return errno;
      ^
test.c: In function 'readFromModule':
test.c:168:14: warning: 'return' with a value, in function returning void
return errno;
      ^
test.c: In function 'callDecryptcao':
test.c:203:4: warning: implicit declaration of function 'toLower' [-Wimplicit-function-declaration]
toLower(aux);
^
test.c: At top level:
test.c:250:6: warning: conflicting types for 'toLower'
void toLower(char string[]){
^
test.c:203:4: note: previous implicit declaration of 'toLower' was here
toLower(aux);
^
test.c: In function 'toLower':
test.c:255:19: warning: implicit declaration of function 'tolower' [-Wimplicit-function-declaration]
string[i] = tolower(string[i]);
               ^
sudo insmod crypt_mod.ko key="0123456789FEDCBA" iv="0123456789FEDCB0"
sudo ./test
Starting device test code example...
Escolha uma Opção
c ) Encrypt String
d ) Decrypt String
h ) Hash Resume
e ) Exit
d
Digite sua String:
d74b4a0c6abebc02a3a70dd2f2b3dd
Reading from the device...
41 4C 4F 42 52 41 53 49 4C 00 00 00 00 00 00 | ALOBRASIL.....
Pressione uma tecla para continuar
```

Figura 6. Decriptação com IV e KEY diferentes

```
root@luiz-VirtualBox: /home/luiz/Documentos/Crypto-Device-Module
Arquivo Editar Ver Pesquisar Terminal Ajuda
Digite uma opção valida
Escolha uma Opção
c ) Encrypt String
d ) Decrypt String
h ) Hash Resume
e ) Exit
h
Digite sua String:
aa
Reading from the device...
E0 C9 03 58 98 DD 52 FC 65 C4 14 54 CE C9 C4 D2 | ...X..R.e..T....
61 1B FB 37 | a..7
Pressione uma tecla para continuar
```

Figura 7. HASH com IV e KEY diferentes



```
root@luiz-VirtualBox: /home/luiz/Documentos/Crypto-Device-Module
Arquivo Editar Ver Pesquisar Terminal Ajuda

Digite uma opção válida
Escolha uma Opção
c ) Encrypt String
d ) Decrypt String
h ) Hash Resume
e ) Exit
e

PROGRAMA FINALIZADO!
sudo journalctl --since "1 minutes ago" | grep kernel
Out 04 13:58:26 luiz-VirtualBox kernel: Base: Initializing the Crypto LKM
Out 04 13:58:26 luiz-VirtualBox kernel: Base: registered correctly with major number 243
Out 04 13:58:26 luiz-VirtualBox kernel: Base: device class registered correctly
Out 04 13:58:26 luiz-VirtualBox kernel: Base: device created correctly
Out 04 13:58:26 luiz-VirtualBox kernel: A chave foi: 0123456789FEDCBA
Out 04 13:58:26 luiz-VirtualBox kernel: 0 iv: 0123456789FEDCB0
Out 04 13:58:26 luiz-VirtualBox kernel: Base: Device has been opened 1 time(s)
Out 04 13:58:30 luiz-VirtualBox kernel: Base: Received 11 characters from the user
Out 04 13:58:30 luiz-VirtualBox kernel: Result Data1: 41 4c 4f 42 52 41 53 49 4c 00 00 00 00 00 00 ALOBRASIL.....
Out 04 13:58:30 luiz-VirtualBox kernel: KEY: 30 31 32 33 34 35 36 37 38 39 46 45 44 43 42 41 0123456789FEDCBA
Out 04 13:58:30 luiz-VirtualBox kernel: Encryption triggered successfully
Out 04 13:58:30 luiz-VirtualBox kernel: Result Data: d7 4b e4 a0 c6 ab ee bc 02 a3 a7 0d d2 f2 b3 dd .K.....
Out 04 13:58:30 luiz-VirtualBox kernel: Base: Sent 16 characters to the user
Out 04 13:59:03 luiz-VirtualBox kernel: Base: Device successfully closed
root@luiz-VirtualBox: /home/luiz/Documentos/Crypto-Device-Module#
```

Figura 8. JOURNALCTL com IV e KEY diferentes

3.1. Problemas Encontrados

Durante o desenvolvimento do projeto, houveram diversos problemas referentes ao Acesso Direto a Memória(DMA), sendo assim, confirmou-se que o erro era causado pelo uso da função `kmalloc()`, inclusa na biblioteca `linux/slab.h` ou pelo próprio sistema operacional. A função `kmalloc` tem controle direto na memória física, e por isso causava erros referentes a compilação e até na interface de usuário. Desse modo, trocou-se o `kmalloc` pelo `vmalloc`, função inclusa na biblioteca `linux/vmalloc.h`, onde há o acesso direto na memória, mas apenas na virtual. Outro problema que vale salientar, é a diferença em tabela ASCII para letras maiúsculas e minúsculas, para manter uma consistência nos resultados sempre convertemos entradas em Hexadecimal para minúsculo.

4. Conclusão

Durante o trabalho utilizamos informações obtidas através de sites que foram indicados para auxiliar na hora de realizar as funções de cifrar, decifrar e calcular o resumo criptográfico (hash) dos dados fornecidos pelo usuário. Tivemos que realizar algumas alterações nos programas base que achamos por conta da diferença de versão que acaba mudando algumas características no código. Por fim obtivemos sucesso ao utilizar da API criptográfica do kernel Linux para realizar todas as funções propostas.

Referências

- [Malloy 2015] Malloy, D. ((acessado em Setembro, 2020),2015). *Writing a Linux Kernel Module — Part 2: A Character Device*. <http://derekmalloy.ie/writing-a-linux-kernel-module-part-2-a-character-device>.
- [Mueller and Vasut 2020] Mueller, S. and Vasut, M. ((acessado em Setembro, 2020)). *Linux Kernel Crypto API*. <https://www.kernel.org/doc/html/v4.12/crypto/index.html>.