



# Projeto 2

Sistemas Operacionais B – Data: 14/11/2017

Integrantes: Armando Dalla Costa Neto,	15118029
Mateus Talzzia Diogo,	15147861
Matheus Augusto Cremonez Guimarães,	15004336
Leonardo Borges Bergamo,	15251275
Paulo Vinicius Martimiano de Oliveira,	15149313
Rafael Mont'Alverne de Souza,	15078371

## 1. Introdução

Com a realização do projeto anterior, foi possível obter uma primeira aproximação com a encriptação de dados, de forma a criptografar, descriptografar e realizar o processo de hash no dado, tudo de acordo com uma chave pré-estabelecida na instanciação do módulo criado.

Com base nos resultados obtidos com o projeto anterior e no conhecimento adquirido, neste projeto o objetivo é a criação de um sistema de arquivos minix, utilizando o sistema operacional Linux, de forma que todo arquivo inserido neste sistema de arquivos, deverá ser criptografado. Ao ler algum arquivo desta partição, o mesmo deverá ser descriptografado e exibir o conteúdo original para o usuário. Todas essas modificações serão feitas através da alteração do módulo original minix, de forma a remodelar suas funcionalidades para a realização do projeto.

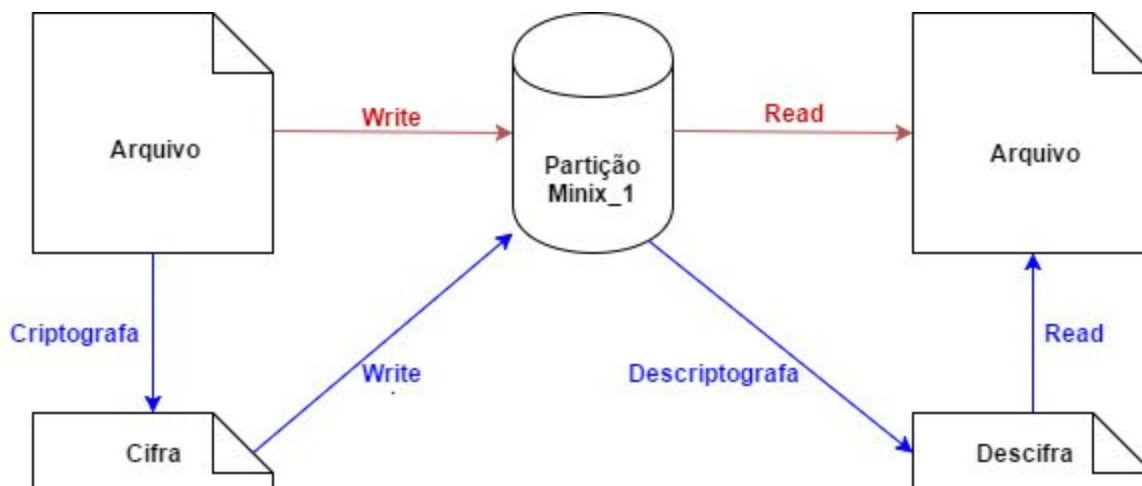
O sistema de arquivos minix é caracterizado por, encontrar-se fora do kernel no espaço de usuário, ser implementado todo em linguagem C e evitar características complexas, além de, ter copiado a estrutura básica do sistema de arquivos do UNIX. Pelo fato de encontrar-se fora do kernel no espaço do usuário, o sistema de arquivos minix pode ser usado como servidor de arquivos de rede independente. Esta independência do sistema de arquivos proporciona vantagens como: poder sofrer modificações praticamente de forma independente do restante do MINIX e poder ser inteiramente removido, recompilado e usado como servidor remoto.

## 2. Detalhes de Projeto

O módulo minix alterado deve ser capaz de chamar funções de encriptação em suas funcionalidades, de forma a utilizá-las de acordo com a ação realizada pelo usuário na partição. Ao carregar o módulo de kernel, deve-se informar no parâmetro “key” a chave simétrica que será utilizada para cifrar e decifrar os dados, como no exemplo a seguir: `insmod cryptomodule.ko key=”secretkey”`.

Os arquivos armazenados na partição criada, deverão ser cifrados pelo módulo minix no momento de sua criação ou atualização, sendo utilizado o algoritmo AES neste processo. Para a leitura, deve ser feito o processo de descryptografia pelo módulo, utilizando a chave fornecida na instanciação.

O diagrama abaixo ilustra resumidamente a abordagem utilizada:



Os fluxos em vermelho representam o padrão de funcionamento, e os fluxos em azul representam os novos fluxos que são utilizados.

## 3. Implementação realizada

A versão do kernel utilizada neste trabalho foi a 4.8.0-36.

As modificações foram realizadas no arquivo `file.c` presente na pasta `minix`.

Primeiro foi realizado a criação da partição do sistema de arquivos Minix através dos comandos abaixo:

```
$ dd if=/dev/zero of=file.img bs=1k count=10000  
$ losetup /dev/loop0 file.img  
$ mkfs.minix -c /dev/loop0 10000
```

Após a criação da partição, foram realizadas as seguintes alterações no arquivo *file.c*: Primeiramente, foram trocadas, dentro da struct, as funções `generic_file_read_iter` e `generic_file_write_iter`. De maneira que, a função de criptografia é chamada no `generic_file_write_iter` e a de descriptografia é chamada no `READ`.

Inicialmente o código chamava a função `generic_file_writer_iter()` quando era realizado a escrita no arquivo e a função `generic_file_read_iter()` quando era realizada a leitura em um arquivo. Foram criadas 2 funções (criptografa e descriptografa) de maneira que invés de serem realizadas as chamadas para as funções padrões de `read` e `write` (`generic_file_writer_iter` e `generic_file_read_iter`) seriam chamadas as novas funções, permitindo que pudessemos alterar as variáveis que são utilizadas por essas variáveis como parâmetro e alterar seus valores, cifrando ou decifrando os dados.

Para criptografar, antes de ser feito um `write`, armazena-se o conteúdo do arquivo em uma variável `buffer` e chama-se a função `retornaEncriptado`, função esta que recebe o `buffer` e realiza criptografia do mesmo.

Para descriptografar, após feito um `read`, armazena-se o conteúdo do arquivo em uma variável `buffer` (de maneira análoga ao procedimento de criptografia) e chama-se a função `retornaEncriptado`.

Essa função recebe 2 parâmetros, o primeiro é o `buffer` com os dados (possuirá o valor de “retorno” da função também) e o segundo uma variável que se possui valor 1 cifra os dados e se possuir o valor 0 decifrar os dados enviados.

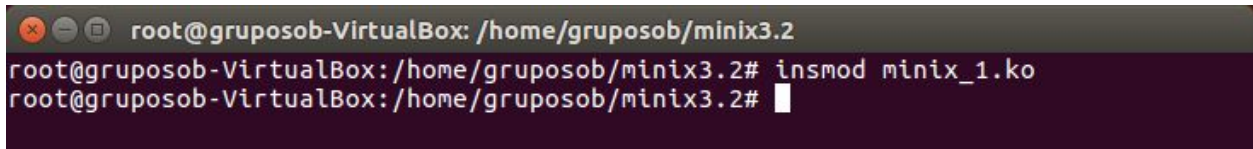
#### 4. Resultados obtidos

De acordo com o projeto realizado pelo grupo, obtivemos resultados aqui descritos em relação ao que nos foi proposto.

Podemos mostrar através dos prints realizados que tanto a criptografia quanto a descriptografia dos arquivos na nossa partição, estão coerentes, visto que acessando eles sem que a partição esteja montada, o conteúdo dos arquivos permanece criptografado.

A execução do módulo pode ser exemplificada com o teste a seguir:

Inicialmente, instanciamos nosso módulo modificado (*minix\_1.ko*) e montamos nossa partição minix (*disk*) no sistema, onde será nossa “ambiente de trabalho”:



```
root@gruposob-VirtualBox: /home/gruposob/minix3.2
root@gruposob-VirtualBox:/home/gruposob/minix3.2# insmod minix_1.ko
root@gruposob-VirtualBox:/home/gruposob/minix3.2#
```

Imagem 1: instancição do módulo modificado (*minix\_1.ko*).

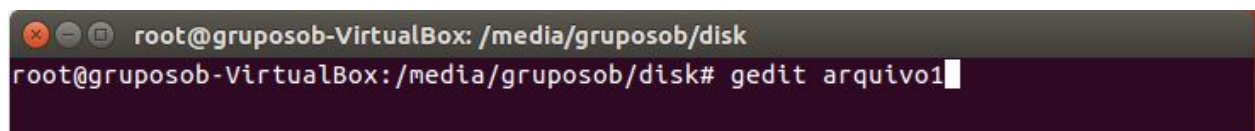
Nosso módulo de criptografia automaticamente considera a chave como “aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa” (32 caracteres ‘a’) como padrão caso ela não

seja passada na instanciação do módulo.

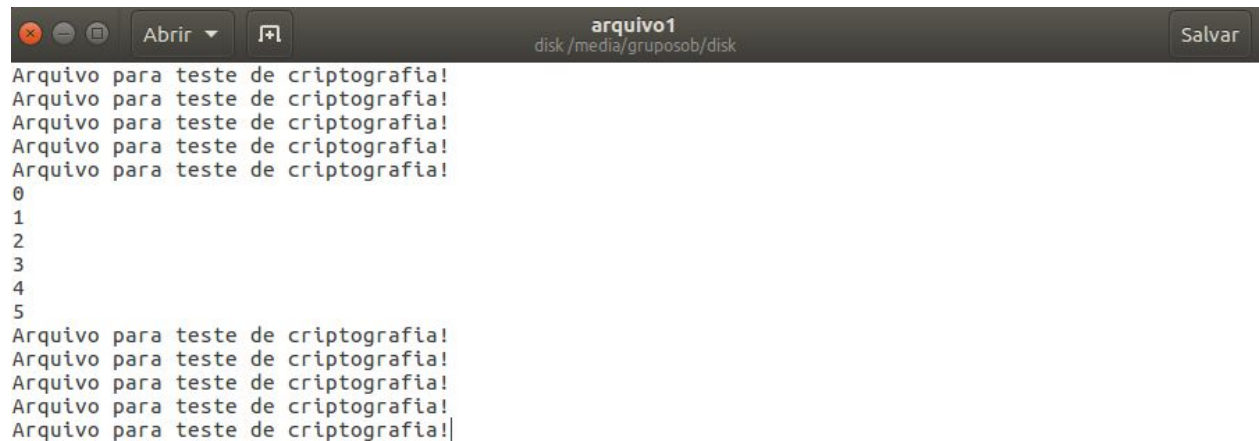


**Imagem 2: partição minix utilizada como “ambiente de trabalho”.**

Após isso, criamos um arquivo de texto, e escrevemos no mesmo, para verificar o funcionamento do módulo:

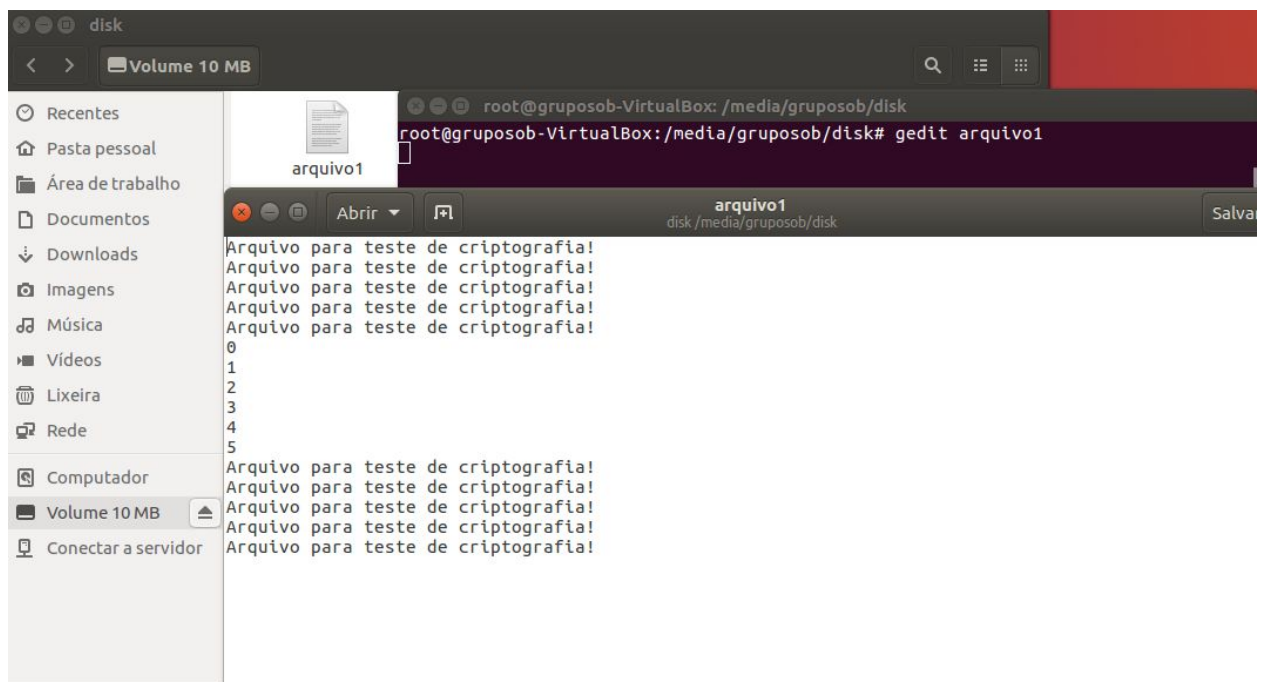


**Imagem 3: criação do arquivo de texto para testes (*arquivo1*).**



**Imagem 4: conteúdo do arquivo1.**

Ainda com nosso módulo modificado instanciado, após escrever, tentamos abrir o arquivo criado para leitura, e verificamos que foi descriptografado com sucesso:



**Imagem 5: abertura, com módulo modificado, do *arquivo1* para leitura.**

Na imagem seguinte, foi instanciado o módulo normal para testar a criptografia,

e podemos observar o funcionamento da mesma:

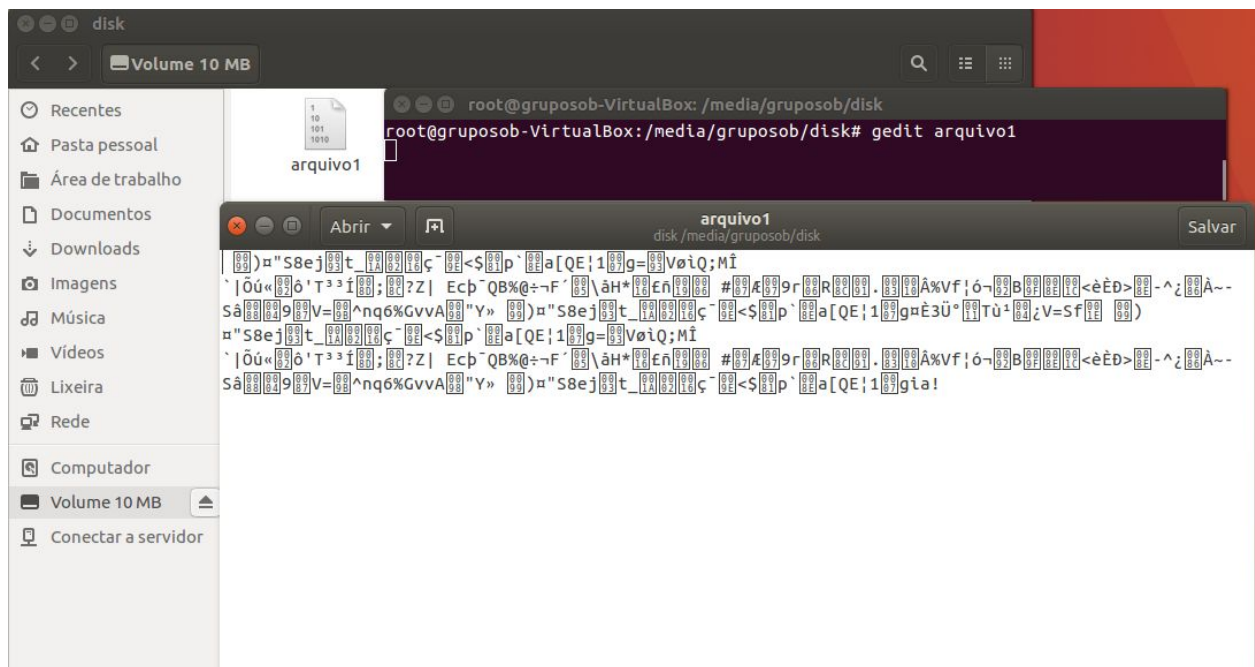


Imagem 6: abertura, com módulo normal, do *arquivo1* para leitura.

Agora, vamos instanciar nosso módulo com uma key diferente, para testar a criptografia:

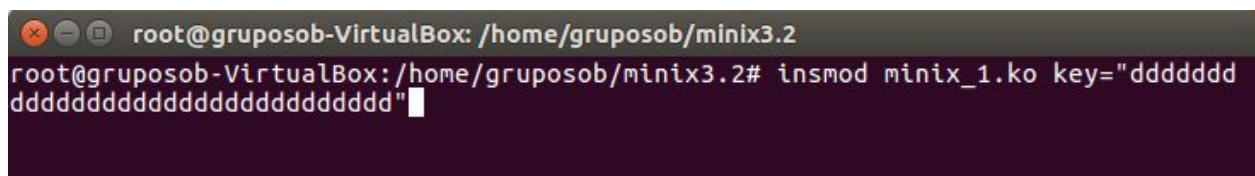


Imagem 7: instanciação do módulo modificado com uma nova chave criptográfica.

É então criado um arquivo de texto (*teste5*) com o mesmo conteúdo do teste anterior. E então instanciamos o módulo padrão e vemos sua criptografia:



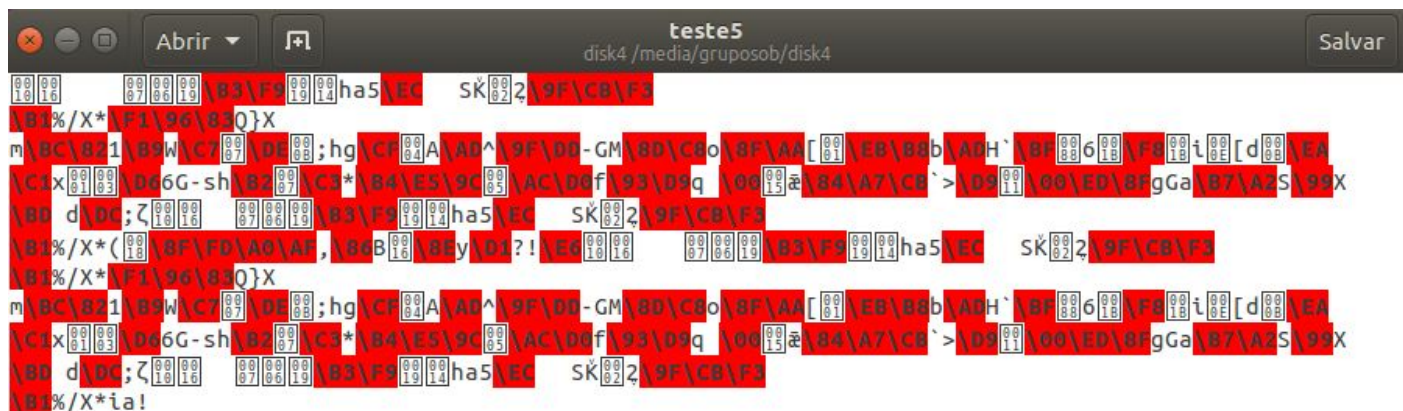


Imagem 8: conteúdo do arquivo criptografado com o módulo padrão instanciado.

É importante observar que a criptografia está diferente do primeiro teste, já que agora estamos com uma chave criptográfica diferente.

Porém, se abrirmos o mesmo arquivo com nosso módulo modificado e com a chave criptográfica correta:



Imagem : conteúdo do arquivo criptografado com o módulo modificado instanciado.

## **5. Observações**

A cifragem e decifragem dos dados funcionam com blocos de 16 caracteres; os últimos caracteres do texto, caso não formem um bloco de 16 caracteres completo, não são utilizados para a cifragem ou decifragem, são mantidos como originais. Essa abordagem foi utilizada pois a maneira como implementamos o projeto, existe uma variável que conta quantos caracteres existem no texto e caso tentemos completar o bloco final, ainda não foi possível alterar o valor dessa variável, ou seja, o arquivo era salvo ignorando os caracteres que foram utilizados para completar o bloco final.

O projeto implementado possui um BUG: após um tempo utilizando-o para escrever e ler arquivos, o mesmo vem a apresentar mal funcionamento e acaba travando a máquina que opera. Foi realizada a re-implementação da cifragem e decifragem dos dados, de maneira minimalista, foram retiradas as bibliotecas importadas que não estavam sendo utilizadas, foram limitados os caracteres que são cifrados, foram reescritos todos os arquivos que compõem a pasta minix e novamente alterado o arquivo file.c e diversas outras abordagens, mas nem uma obteve sucesso em resolver o problema.

## **6. Conclusão**

Com a realização deste projeto foi possível uma maior familiarização com os detalhes de implementação de um sistema de arquivos minix, bem como, implementar, compilar, instalar e testar módulos de kernel, fazendo assim as devidas modificações necessárias para a conclusão deste trabalho.

## 7. Referências

*Sistemas de arquivos minix. Disponível em:*

<<http://www.ic.unicamp.br/~islene/2s2007-mo806/slides/Minix.pdf>> Acesso em 10 nov. 2017

*Anatomy of the Linux File System. Disponível em:*

<<https://www.ibm.com/developerworks/library/l-linuxfilesystem/index.html>> Acesso em 10 nov. 2017

*Linux Kernel Crypto API. Disponível em:*

<<https://www.kernel.org/doc/html/v4.12/crypto/index.html>> Acesso em 10 nov. 2017