Sessão 8: Autenticação, autorização e certificação digital

1) Uso de criptografia simétrica em arquivos

Esta atividade será realizada nas máquinas virtuais *FWGW1-G* e *LinServer-G*.

1. Na máquina *FWGW1-G*, descubra quais cifras simétricas são suportadas pelo programa `gpg` (*GNU Privacy Guard*).

```
$ hostname
FWGW1-A
```

```
$ gpg --version | grep Cipher -A1
Cipher: IDEA, 3DES, CAST5, BLOWFISH, AES, AES192, AES256, TWOFISH,
        CAMELLIA128, CAMELLIA192, CAMELLIA256
```

2. Crie um arquivo `teste.txt` com qualquer conteúdo. Criptografe-o usando a cifra simétrica AES256, com senha `rnpesr`. Em seguida, copie o arquivo cifrado resultante para o diretório *home* do usuário `aluno`, na máquina *LinServer-G*, usando o comando `scp`.

```
$ echo 'teste de cifragem' > teste.txt
```

```
$ gpg --symmetric --cipher-algo AES256 teste.txt
```

```
$ ls teste.txt*
teste.txt  teste.txt.gpg
```

```
$ scp teste.txt.gpg aluno@172.16.1.10:~
teste.txt.gpg                                                 100%   94
0.1KB/s   00:00
```

3. Na máquina *LinServer-G*, tente descriptografar o arquivo copiado. Seu conteúdo permanece o mesmo?

```
$ hostname
LinServer-A
```

```
$ ls teste.txt*
teste.txt.gpg
```

```
$ gpg -o teste.txt.out -d teste.txt.gpg
gpg: AES256 encrypted data
gpg: encrypted with 1 passphrase
```

```
$ cat teste.txt.out
teste de cifragem
```

2) Uso de criptografia assimétrica em arquivos

Esta atividade será realizada nas máquinas virtuais *FWGW1-G* e *LinServer-G*.

1. Na máquina *FWGW1-G*, descubra quais cifras assimétricas são suportadas pelo programa `gpg` (*GNU Privacy Guard*).

```
$ hostname
FWGW1-A
```

```
$ gpg --version | grep Pubkey
Pubkey: RSA, RSA-E, RSA-S, ELG-E, DSA
```

2. Vamos fazer um exercício de criptografia usando chaves assimétricas entre dois usuários fictícios, Alice (operando na máquina *FWGW1-G*) e Bobby (operando na máquina *LinServer-G*). Vamos começar por Alice — gere um par de chaves assimétricas RSA padrão, com 4096 bits e sem data de expiração para ela, usando o programa `gpg`. O e-mail de Alice será alice@seg12.esr.rnp.br , e a senha de acesso à chave será `rnpesr123`.

```
$ gpg --gen-key
```

```
gpg (GnuPG) 1.4.18; Copyright (C) 2014 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Please select what kind of key you want:
   (1) RSA and RSA (default)
   (2) DSA and Elgamal
   (3) DSA (sign only)
   (4) RSA (sign only)
Your selection? 1
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (2048) 4096
Requested keysize is 4096 bits
Please specify how long the key should be valid.
         0 = key does not expire
      <n>  = key expires in n days
      <n>w = key expires in n weeks
      <n>m = key expires in n months
      <n>y = key expires in n years
Key is valid for? (0) 0
Key does not expire at all
Is this correct? (y/N) y

You need a user ID to identify your key; the software constructs the user ID
from the Real Name, Comment and Email Address in this form:
    "Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"

Real name: Alice
Email address: alice@seg12.esr.rnp.br
Comment:
You selected this USER-ID:
    "Alice <alice@seg12.esr.rnp.br>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? O
You need a Passphrase to protect your secret key.
```

Nesse momento o gpg irá informar que precisa de um grande número de bytes aleatórios para ter entropia na geração de números primos usada no algoritmo RSA. Aperte teclas quaisquer no teclado até que a chave seja gerada, como mostrado abaixo:

```
gpg: /home/aluno/.gnupg/trustdb.gpg: trustdb created
gpg: key 209411F7 marked as ultimately trusted
public and secret key created and signed.

gpg: checking the trustdb
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 1u
pub   4096R/209411F7 2018-09-06
      Key fingerprint = 2D01 2274 8A9A 180C E269  B387 113A A4ED 2094 11F7
uid                  Alice <alice@seg12.esr.rnp.br>
sub   4096R/B2CCF948 2018-09-06
```

3. No caso da máquina *LinServer-G*, a entropia mesmo após digitar um grande número de teclas é baixa, pois há menor número de fontes de aleatoriedade (como o fato de não estar conectado à uma rede pública via eth0, por exemplo). Ao invés de "cansar o braço" digitando caracteres no passo de geração de chaves, instale o pacote `rng-tools` e rode o comando `rngd -r /dev/urandom`:

```
# hostname
LinServer-A
```

```
# apt-get install rng-tools
```

```
# rngd -r /dev/urandom
```

4. Agora sim, vamos agora gerar a chave de Bobby, na máquina *LinServer-G*. Repita o procedimento do passo (2), alterando o nome de usuário para Bobby e o email para bobby@seg12.esr.rnp.br .

```
$ hostname
LinServer-A
```

```
$ gpg --gen-key
(...)
```

```
gpg: /home/aluno/.gnupg/trustdb.gpg: trustdb created
gpg: key EAD0CF1F marked as ultimately trusted
public and secret key created and signed.

gpg: checking the trustdb
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 1u
pub   4096R/EAD0CF1F 2018-09-06
      Key fingerprint = 23CF A392 6118 B50A 9115  B1D2 D42E AF5D EAD0 CF1F
uid                  Bobby <bobby@seg12.esr.rnp.br>
sub   4096R/4A677FB6 2018-09-06
```

5. Temos que exportar as chaves públicas de ambos os usuários, copiá-las para a máquina remota, e importá-las. Comece pela chave de Alice, exportando-a em formato *ASCII armored*; em seguida, copie-a para a máquina *LinServer-G* usando o `scp`, importe-a usando `gpg --import` e assine a chave.

   Na máquina *FWGW1-G*, execute:

   ```
   $ hostname
   FWGW1-A
   ```

   ```
   $ gpg --armor --export Alice > alice_public.asc
   ```

   ```
   $ scp alice_public.asc aluno@172.16.1.10:~
   alice_public.asc                                              100% 3083
   3.0KB/s   00:00
   ```

   Agora, na máquina *LinServer-G*, execute:

   ```
   $ hostname
   LinServer-A
   ```

   ```
   $ gpg --import alice_public.asc
   gpg: key 209411F7: public key "Alice <alice@seg12.esr.rnp.br>" imported
   gpg: Total number processed: 1
   gpg:               imported: 1  (RSA: 1)
   ```

   Valide a chave recebida com o remetente (por exemplo, verficando que o *fingerprint* está de fato correto), e posteriormente assine-a como mostrado a seguir.

```
$ gpg --edit-key Alice
gpg (GnuPG) 1.4.18; Copyright (C) 2014 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.


pub  4096R/209411F7  created: 2018-09-06  expires: never       usage: SC
                     trust: unknown       validity: unknown
sub  4096R/B2CCF948  created: 2018-09-06  expires: never       usage: E
[ unknown] (1). Alice <alice@seg12.esr.rnp.br>
```

```
gpg> fpr
pub   4096R/209411F7 2018-09-06 Alice <alice@seg12.esr.rnp.br>
 Primary key fingerprint: 2D01 2274 8A9A 180C E269  B387 113A A4ED 2094 11F7
```

```
gpg> sign

pub  4096R/209411F7  created: 2018-09-06  expires: never       usage: SC
                     trust: unknown       validity: unknown
 Primary key fingerprint: 2D01 2274 8A9A 180C E269  B387 113A A4ED 2094 11F7

     Alice <alice@seg12.esr.rnp.br>

Are you sure that you want to sign this key with your
key "Bobby <bobby@seg12.esr.rnp.br>" (EAD0CF1F)

Really sign? (y/N) y

You need a passphrase to unlock the secret key for
user: "Bobby <bobby@seg12.esr.rnp.br>"
4096-bit RSA key, ID EAD0CF1F, created 2018-09-06
```

```
gpg> check
uid  Alice <alice@seg12.esr.rnp.br>
sig!3        209411F7 2018-09-06  [self-signature]
sig!         EAD0CF1F 2018-09-06  Bobby <bobby@seg12.esr.rnp.br>
```

```
gpg> quit
Save changes? (y/N) y
```

6. Faça o procedimento reverso, exportando/copiando/importando e assinando a chave de Bobby na máquina de Alice. Lembre-se que o `ssh` para a máquina *FWGW1-G* é permitido apenas a partir da Intranet, então pode ser mais interessante iniciar o procedimento de cópia a partir do firewall, e não da máquina *LinServer-G*.

Primeiro, na máquina *LinServer-G*, vamos exportar a chave:

```
$ hostname
LinServer-A
```

```
$ gpg --armor --export Bobby > bobby_public.asc
```

Como não há regra que permita `ssh` da DMZ para a máquina *FWGW1-G*, vamos fazer a cópia no sentido inverso:

```
$ hostname
FWGW1-A
```

```
$ scp aluno@172.16.1.10:~/bobby_public.asc ~
bobby_public.asc                                              100% 3083
3.0KB/s   00:00
```

Agora, basta importar e assinar a chave:

```
$ gpg --import bobby_public.asc
gpg: key EAD0CF1F: public key "Bobby <bobby@seg12.esr.rnp.br>" imported
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
```

```
$ gpg --edit-key Bobby
gpg (GnuPG) 1.4.18; Copyright (C) 2014 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.


pub  4096R/EAD0CF1F  created: 2018-09-06  expires: never       usage: SC
                     trust: unknown       validity: unknown
sub  4096R/4A677FB6  created: 2018-09-06  expires: never       usage: E
[ unknown] (1). Bobby <bobby@seg12.esr.rnp.br>
```

```
gpg> fpr
pub   4096R/EAD0CF1F 2018-09-06 Bobby <bobby@seg12.esr.rnp.br>
 Primary key fingerprint: 23CF A392 6118 B50A 9115  B1D2 D42E AF5D EAD0 CF1F
```

```
gpg> sign

pub  4096R/EAD0CF1F  created: 2018-09-06  expires: never       usage: SC
                     trust: unknown       validity: unknown
 Primary key fingerprint: 23CF A392 6118 B50A 9115  B1D2 D42E AF5D EAD0 CF1F

     Bobby <bobby@seg12.esr.rnp.br>

Are you sure that you want to sign this key with your
key "Alice <alice@seg12.esr.rnp.br>" (209411F7)

Really sign? (y/N) y

You need a passphrase to unlock the secret key for
user: "Alice <alice@seg12.esr.rnp.br>"
4096-bit RSA key, ID 209411F7, created 2018-09-06
```

```
gpg> check
uid  Bobby <bobby@seg12.esr.rnp.br>
sig!3        EAD0CF1F 2018-09-06  [self-signature]
sig!         209411F7 2018-09-06  Alice <alice@seg12.esr.rnp.br>
```

```
gpg> quit
Save changes? (y/N) y
```

7. Agora, vamos fazer o teste de criptografia assimétrica propriamente dito. Na máquina *FWGW1-G*, verifique que as chaves estão de fato disponíveis. Em seguida, criptografe um documento de texto com conteúdo qualquer com a chave pública de Bobby, envie para a máquina *LinServer-G*, e tente decriptá-lo usando a chave privada de Bobby.

   Na máquina *FWGW1-G*, vamos verificar se as chaves estão disponíveis:

```
$ hostname
FWGW1-A
```

```
$ gpg --list-keys
gpg: checking the trustdb
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0  valid:   1  signed:   1  trust: 0-, 0q, 0n, 0m, 0f, 1u
gpg: depth: 1  valid:   1  signed:   0  trust: 1-, 0q, 0n, 0m, 0f, 0u
/home/aluno/.gnupg/pubring.gpg
------------------------------
pub   4096R/209411F7 2018-09-06
uid                  Alice <alice@seg12.esr.rnp.br>
sub   4096R/B2CCF948 2018-09-06

pub   4096R/EAD0CF1F 2018-09-06
uid                  Bobby <bobby@seg12.esr.rnp.br>
sub   4096R/4A677FB6 2018-09-06
```

Perfeito. Vamos criar um documento `asym.txt` com conteúdo qualquer e criptografá-lo com a chave pública de Bobby, e finalmente copiá-lo para a máquina *LinServer-G*:

```
$ echo 'teste assimetrico' > asym.txt
```

```
$ gpg -e -r Bobby asym.txt
```

```
$ ls asym.txt*
asym.txt  asym.txt.gpg
```

```
$ scp asym.txt.gpg aluno@172.16.1.10:~
asym.txt.gpg                                                  100%  614
0.6KB/s   00:00
```

Na máquina *LinServer-G*, vamos tentar decriptar o arquivo com a chave privada de Bobby:

```
$ hostname
LinServer-A
```

```
$ ls asym.txt*
asym.txt.gpg
```

```
$ gpg -o asym.txt -d asym.txt.gpg

You need a passphrase to unlock the secret key for
user: "Bobby <bobby@seg12.esr.rnp.br>"
4096-bit RSA key, ID 4A677FB6, created 2018-09-06 (main key ID EAD0CF1F)

gpg: encrypted with 4096-bit RSA key, ID 4A677FB6, created 2018-09-06
      "Bobby <bobby@seg12.esr.rnp.br>"
```

```
$ cat asym.txt
teste assimetrico
```

8. Vamos agora testar a assinatura digital de arquivos. Começando a partir da máquina *LinServer-G*, crie um arquivo texto com conteúdo qualquer. Assine-o com a chave privada de Bobby, e copie o arquivo para a máquina *FWGW1-G*. Finalmente, verifique a assinatura usando o *keyring* de Alice.

   Na máquina *LinServer-G*, vamos criar um arquivo com conteúdo qualquer e assiná-lo usando a chave privada de Bobby, em texto claro (opção `--clearsign`):

   ```
   $ hostname
   LinServer-A
   ```

   ```
   $ echo 'teste assinatura' > sign.txt
   ```

   ```
   $ gpg --clearsign sign.txt

   You need a passphrase to unlock the secret key for
   user: "Bobby <bobby@seg12.esr.rnp.br>"
   4096-bit RSA key, ID EAD0CF1F, created 2018-09-06
   ```

   ```
   $ ls sign.txt*
   sign.txt      sign.txt.asc
   ```

   Note que tanto o conteúdo do arquivo quanto a assinatura estão em texto claro, uma vez que a mensagem foi apenas assinada (e não criptografada).

```
$ cat sign.txt.asc
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

teste assinatura
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1

iQIcBAEBAgAGBQJbkQzSAAoJENQur13q0M8fuDIP/Rhid0LBw1jzir/gqNHHk4wy
1gfubA7Rc8KG1DO7vK/KHbk3PCP+Z15xKNm5A3WV02XRWZYsy6rTolrJ8AkqBP9O
k0sgmDeBOxIQwztfWiVXF/Nm5jPzmbczVvTloCY+nHWKVjnP4ryWWi9pAmXjFNG1
7+ThOQbQAmLMBKxA8kivr2SF7DjcC8oC2HDpzc3+VIBi4TgPRLcu3caEyI94zqHH
X5AKivyVi+G6/KywG3WNIYcg1VPvg7s8I0a6fdQF25bj/DUyN1lwfelPmST2A0Ap
1u4vkWsRV7yMeNANtQ6+0DL/Vwv/JeZOJJOWTUwSJq/2QyFKPzDhAFgOkO6MIgCx
caOd0tqeCElWolGfvBP+1j3Zo2dxR2BUSbelleKEn6n1D3uIsB325SPQzcm7FdDj
9F77QjIPquKlGJzHVIjVv/GQoWYO5BWpGIhwUXW3SEnZjQi3UDD1IJJH/8GIxYFY
TpMSi6DL9Q4SBYBeWwV/dZqebkNII4Ire56bxcKT3G9qko7ISv27WmlZ1TSUsKud
S2Zprb7wMXgpJgXvFFw/+XrhNGTbPAzv9/I1khy1KmuxgQzpD7xXMJOXVEfkdQhK
mbRf1EKVob9X+urzcmfbn/3FLtG6kPFa0Xxwv0OKhIPSpwgA2mhL0tMKqOmmO+bb
mk0WjV3ZzHIWi7sZ2LHH
=63dA
-----END PGP SIGNATURE-----
```

Vamos copiar o arquivo para a máquina *FWGW1-G* (lembrando que a cópia deve ser iniciada no sentido inverso, devido às regras de firewall), e verificar a assinatura.

```
$ hostname
FWGW1-A
```

```
$ scp aluno@172.16.1.10:~/sign.txt.asc ~
sign.txt.asc                                                  100%  883
0.9KB/s   00:00
```

```
$ gpg --verify sign.txt.asc
gpg: Signature made Thu 06 Sep 2018 07:17:38 AM EDT using RSA key ID EAD0CF1F
gpg: Good signature from "Bobby <bobby@seg12.esr.rnp.br>"
```

9. Finalmente, vamos "juntar tudo". Da máquina *FWGW1-G*, crie um arquivo texto com conteúdo qualquer e (1) assine-o com a **chave privada de Alice**, e (2) criptografe-o com a **chave pública de Bobby**. Copie o arquivo para a máquina *LinServer-G*, decripte-o e verifique sua assinatura.

   Na máquina *FWGW1-G*, vamos criar um arquivo com conteúdo qualquer, assiná-lo, criptografá-lo e enviar para a máquina *LinServer-G*:

```
$ hostname
FWGW1-A
```

```
$ echo 'teste assinatura e criptografia assimetrica' > sign-asym.txt
```

```
$ gpg -s -e -r Bobby sign-asym.txt

You need a passphrase to unlock the secret key for
user: "Alice <alice@seg12.esr.rnp.br>"
4096-bit RSA key, ID 209411F7, created 2018-09-06
```

```
$ ls sign-asym.txt*
sign-asym.txt  sign-asym.txt.gpg
```

```
$ scp sign-asym.txt.gpg aluno@172.16.1.10:~
sign-asym.txt.gpg                                             100% 1207
1.2KB/s   00:00
```

Agora, na máquina *LinServer-G*, basta invocar a opção `-d` do `gpg`. Além de decriptar o arquivo, sua assinatura será verificada.

```
$ hostname
LinServer-A
```

```
$ gpg -o sign-asym.txt -d sign-asym.txt.gpg

You need a passphrase to unlock the secret key for
user: "Bobby <bobby@seg12.esr.rnp.br>"
4096-bit RSA key, ID 4A677FB6, created 2018-09-06 (main key ID EAD0CF1F)

gpg: encrypted with 4096-bit RSA key, ID 4A677FB6, created 2018-09-06
      "Bobby <bobby@seg12.esr.rnp.br>"
gpg: Signature made Thu 06 Sep 2018 07:24:59 AM EDT using RSA key ID 209411F7
gpg: Good signature from "Alice <alice@seg12.esr.rnp.br>"
```

```
$ cat sign-asym.txt
teste assinatura e criptografia assimetrica
```

3) Uso de criptografia assimétrica em e-mails

Esta atividade será realizada em sua máquina física.

Vamos agora testar o procedimento de criptografia assimétrica usado na atividade (2) em um cenário mais prático: no envio e recebimento de e-mails.

1. Crie uma conta de e-mail gratuita no serviço GMail, do Google.
2. Em sua máquina física, instale o programa *gpg4win* (que pode ser baixado em https://www.gpg4win.org/download.html). Durante a instalação, aceite todas as opções padrão, e desmarque a caixa *Executar Kleopatra* ao final do processo de instalação.
3. Em sua máquina física, instale o cliente de e-mail *Mozilla Thunderbird* (que pode ser baixado em https://www.thunderbird.net/pt-BR/thunderbird/all/). Durante a instalação, aceite todas as opções padrão.
4. Ao abrir o Thunderbird, adicione a conta de e-mail criada no passo (1), como mostra a imagem a seguir:

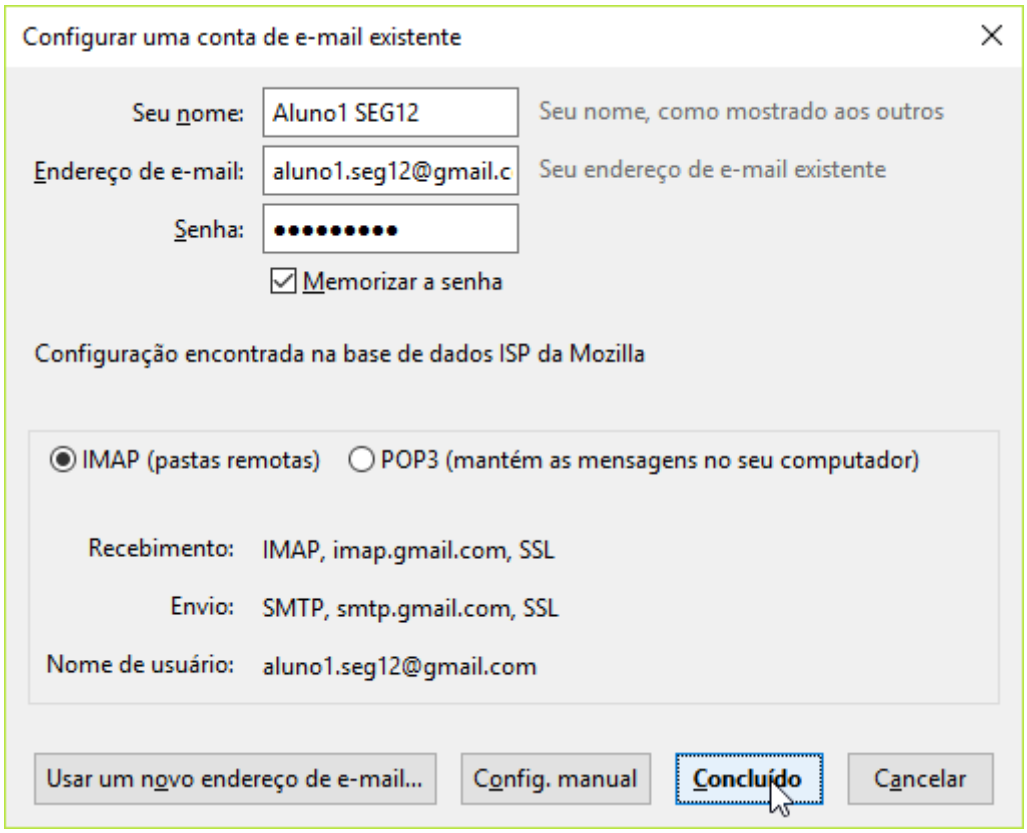

Figura 49: Adicionando uma conta de e-mail ao Thunderbird

5. No Thunderbird, navegue no menu localizado no canto superior direito. Clique em *Extensões* > *Extensões*. No canto superior da janela, pesquise por `enigmail` e pressione ENTER. O primeiro resultado, a extensão *Enigmail*, é o que queremos: clique no botão *Adicionar ao Thunderbird* > *Instalar agora*.
6. Desde a versão 2.0.0 do `Enigmail`, lançada em março de 2018, o modo padrão de operação é o *Enigmail/PeP*. O *PeP* (*pretty Easy privacy* cujo website é https://www.pep.security/) é uma implementação de segurança para e-mails com o objetivo expresso de ser simples e de baixa configuração. Para o nosso cenário, isso significa:

- Geração automática de pares de chaves assimétricas
- Distribuição automática de chaves públicas via anexo ou *upload* para servidores de chaves (*keyservers*)
- Criptografia e assinatura automática de mensagens

7. Vamos testar esses conceitos. Envie uma mensagem para o seu colega usando o Thunderbird. Caso o *Enigmail/PeP* esteja funcionando corretamente, o botão *Habilitar a Proteção* deverá estar marcado no centro da tela:

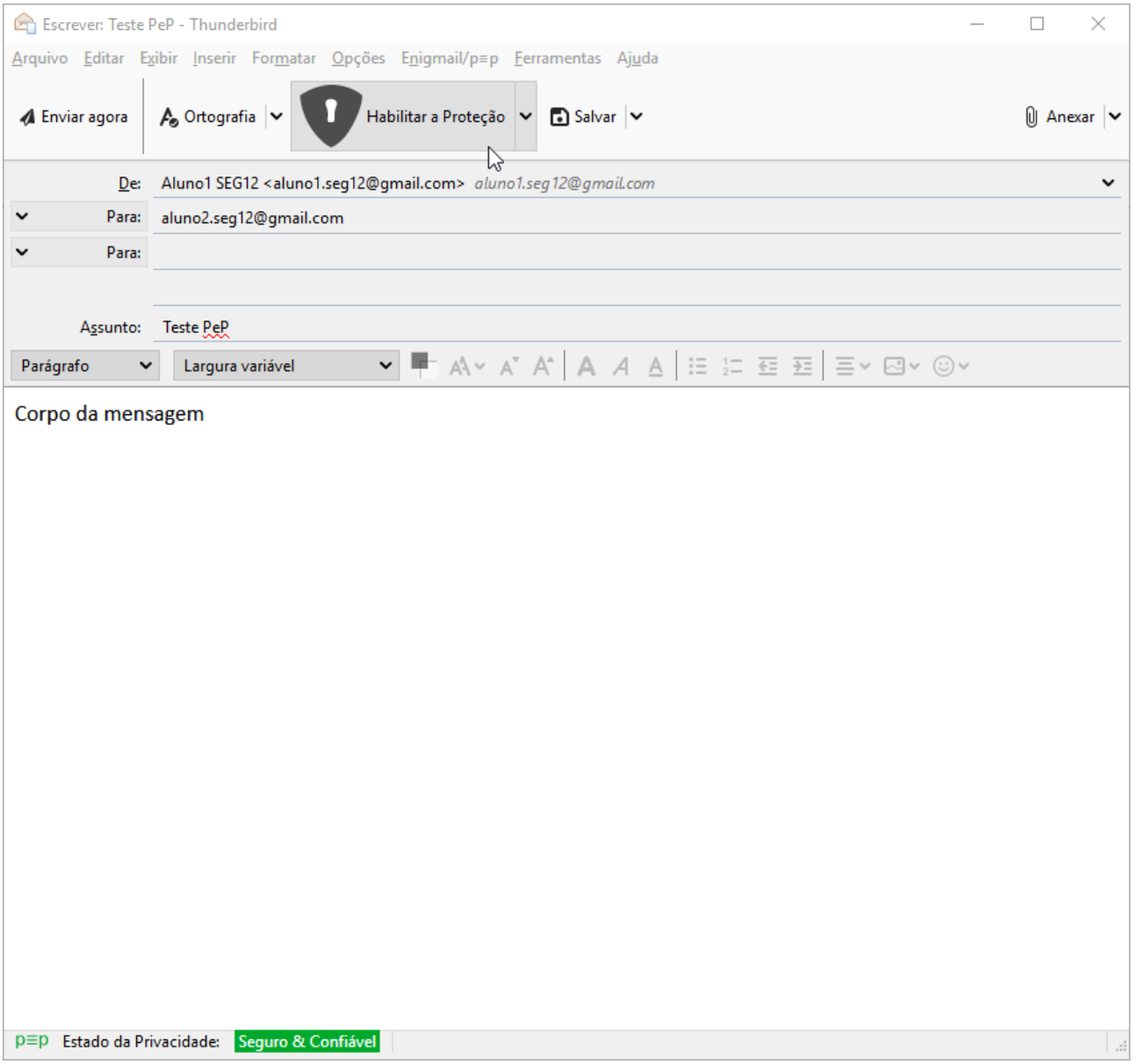

Figura 50: PeP habilitado no Thunderbird

Clicando no botão *Seguro & Confiável* na base da janela, o *Enigmail/PeP* mostra que o envio de mensagens será feito de forma segura (i.e. criptografada) e confiável (i.e. assinada), como mostra a imagem a seguir:

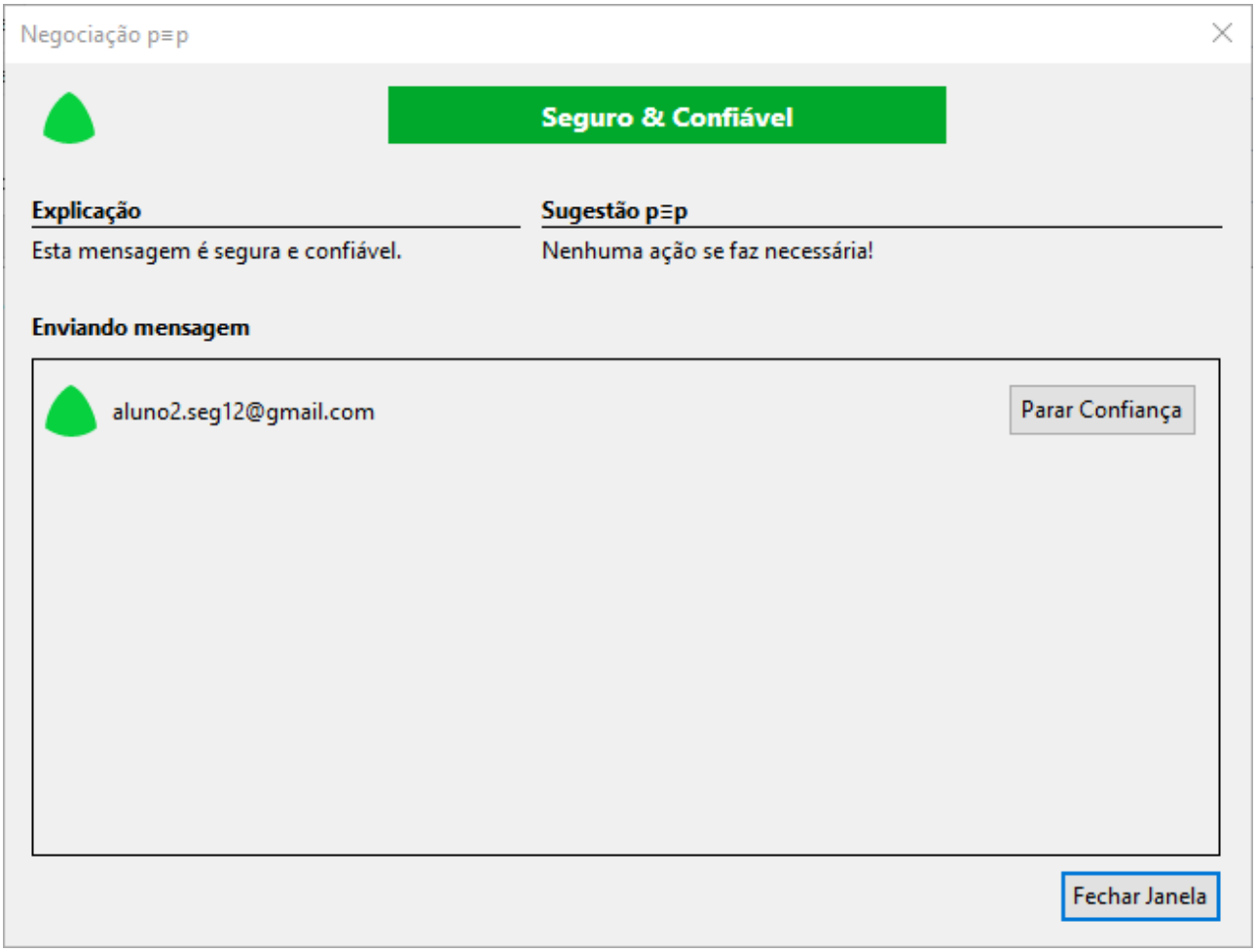

Figura 51: Relação de confiança Enigmail/PeP no Thunderbird

8. Teste o envio de mensagens entre você e seu colega. O *Enigmail/PeP* está funcionando corretamente? O que você achou desse esquema facilitado de criptografia assimétrica?

4) Criptografia de partições e volumes

Esta atividade será realizada em sua máquina física.

1. Instale o *VeraCrypt* (que pode ser baixado em https://www.veracrypt.fr/en/Downloads.html) em sua máquina física. Durante a instalação, aceite todas as opções padrão.
2. O VeraCrypt pode criptografar partições inteiras ou apenas criar um contêiner seguro. Com isso, podemos gravar arquivos sigilosos no contêiner e transportá-lo através de mídia física ou meio não confiável de forma bastante conveniente. Na tela principal do VeraCrypt, clique em *Create Volume*.

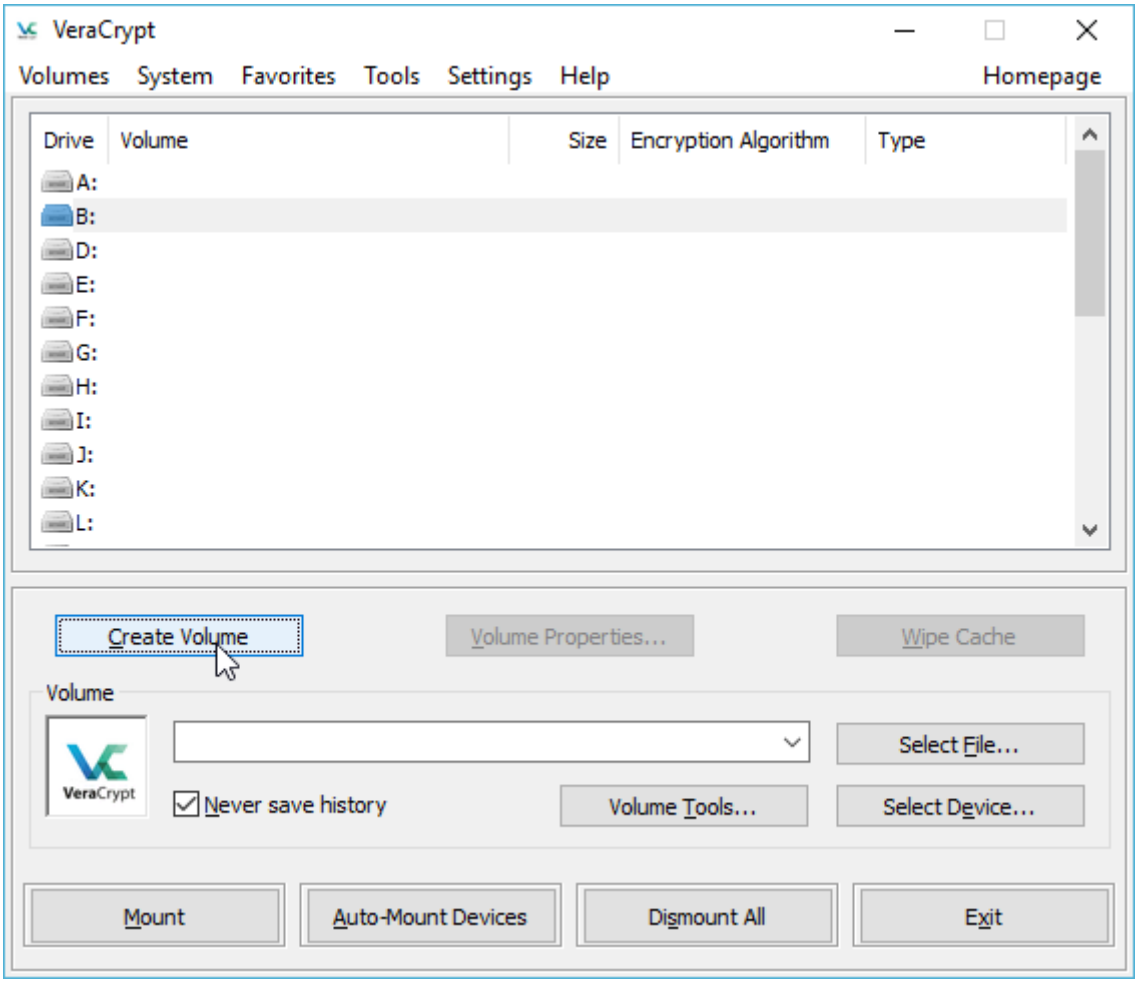

Figura 52: Criação de volumes no VeraCrypt, parte 1

3. Na tela seguinte, mantenha marcada a opção *Create an encrypted file container* e clique em *Next*.

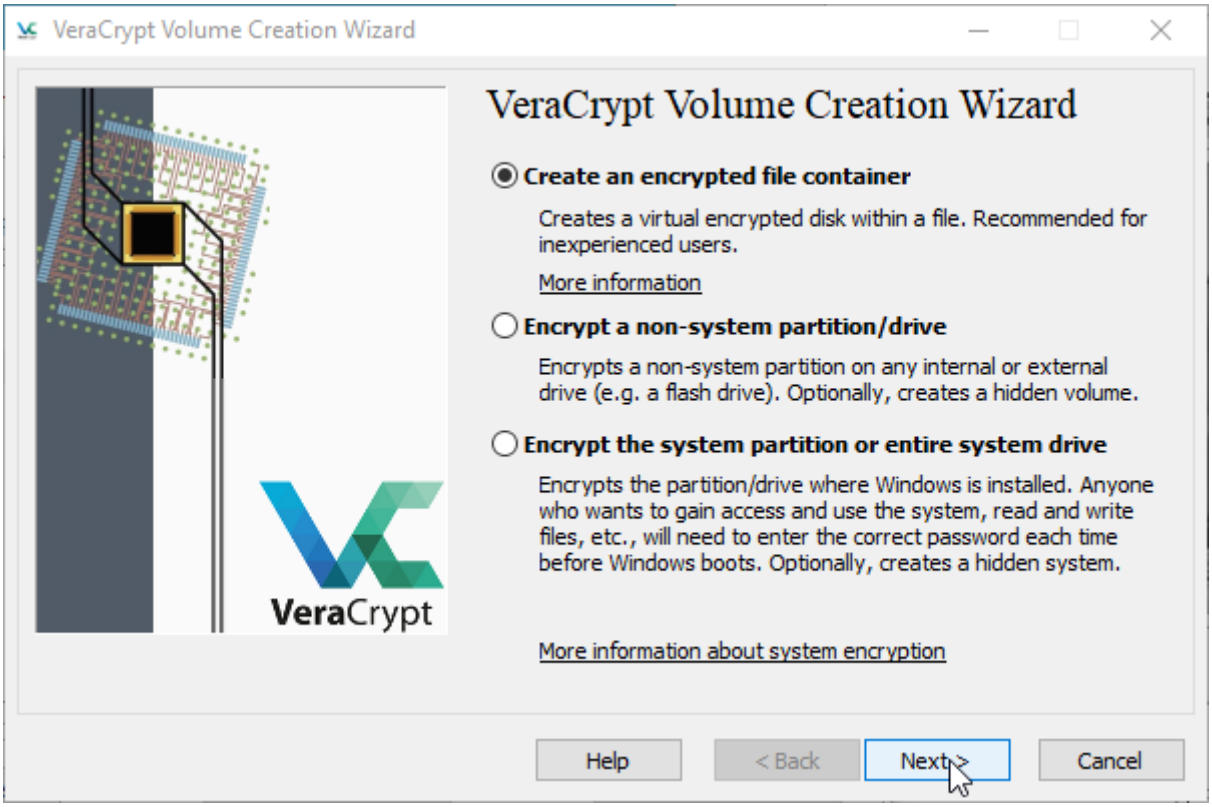

Figura 53: Criação de volumes no VeraCrypt, parte 2

4. Na tela subsequente, mantenha marcada a opção *Standard VeraCrypt volume* e clique em *Next*.
5. Em *Volume Location*, selecione uma pasta/arquivo destino para o contêiner e clique em *Next*.

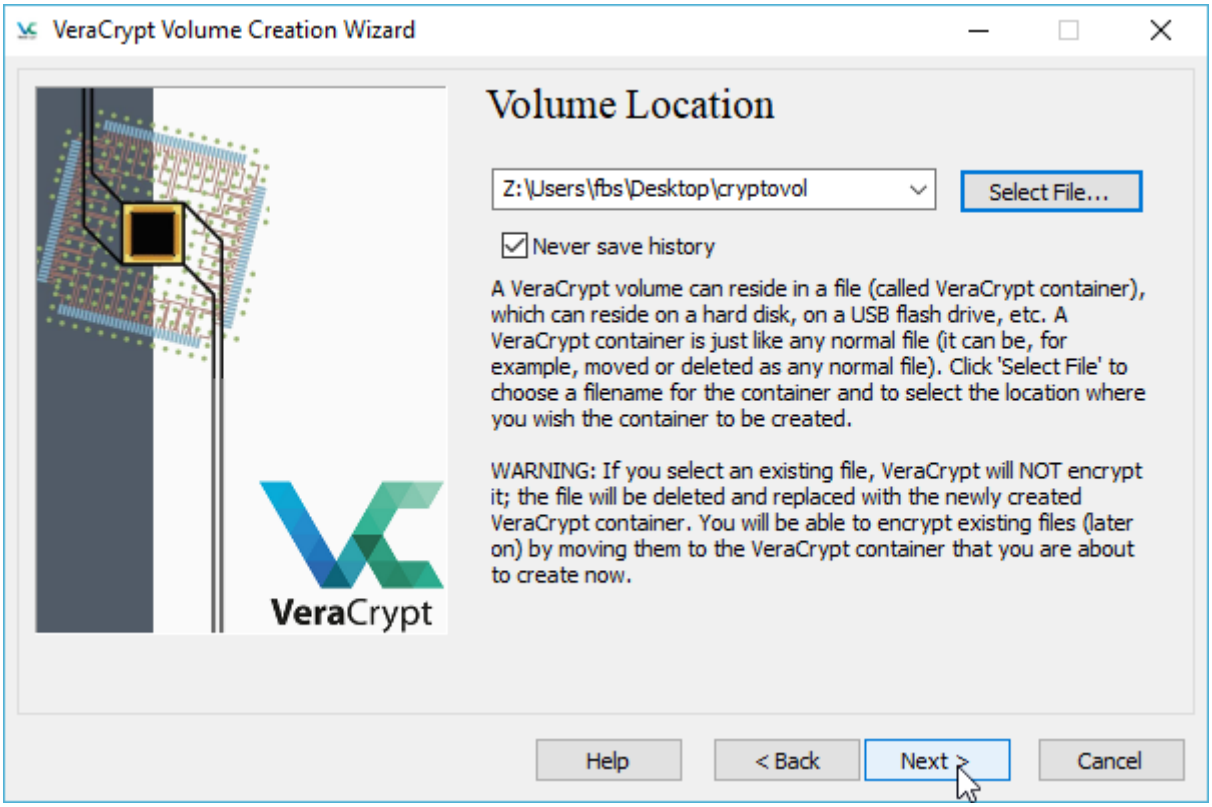

Figura 54: Criação de volumes no VeraCrypt, parte 3

6. Para as opções de criptografia, mantenha o algoritmo AES e hash SHA-512, e clique em *Next*.
7. Para o tamanho do volume, escolha 50MB, e clique em *Next*.
8. Para a senha do contêiner, é importante escolher uma senha forte que não seja facilmente descoberta. Para fins de teste, usaremos `rnpesr123`. Clique em *Next*.
9. Mantenha o *filesystem* em FAT, e mova o mouse para gerar entropia. Finalmente, clique em *Format*.
10. Para montar o volume, selecione uma letra vazia no seu sistema. A seguir, no quadro *Volume* da tela principal do VeraCrypt, clique em *Select File...* e selecione o arquivo indicado no passo (5). Depois, clique em *Mount* e digite a senha informada no passo (8).

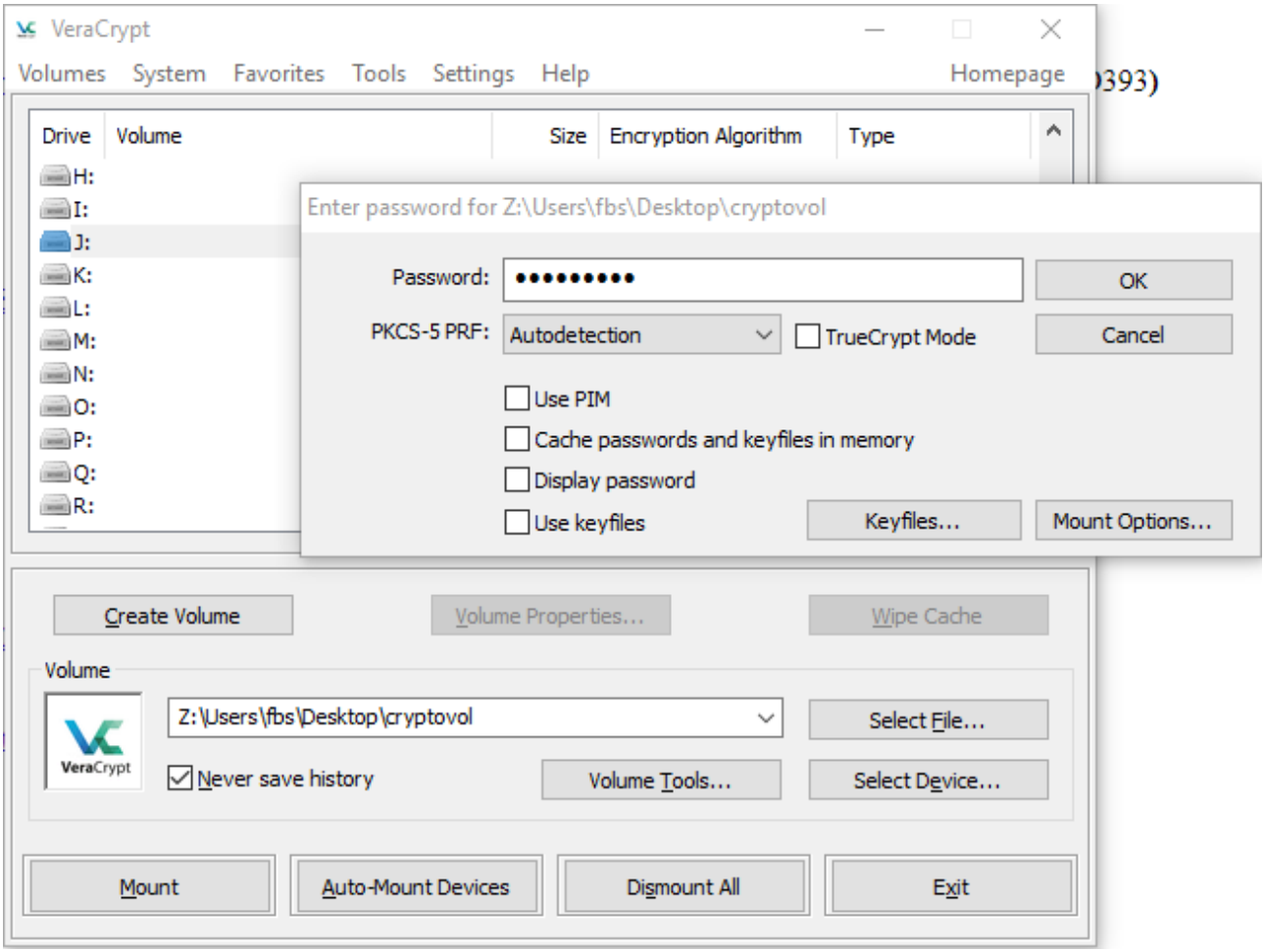

Figura 55: Criação de volumes no VeraCrypt, parte 4

11. Pronto, o volume criptografado está montado. Basta escrever arquivos como desejado e, ao final do processo, clicar em *Dismount* na janela principal do VeraCrypt. Caso queira mover o volume criptografado para outro local, copie-o em um *pendrive*, mídia removível ou mesmo através da Internet, e remonte-o no local de destino.

5) Autenticação usando sistema OTP

Esta atividade será realizada na máquina *LinServer-G*.

Nesta atividade iremos instalar e configurar um sistema TOTP (*time-based one-time password*) usando a ferramenta *Google Authenticator* na máquina *LinServer-G*. Essa autenticação de duplo fator irá prover mais segurança durante logins SSH na máquina-alvo.

1. Instale **em seu celular** o aplicativo *Google Authenticator*:
   - Sistemas Android: https://play.google.com/store/apps/details?id=com.google.android.apps.authenticator2&hl=en
   - Sistemas Apple: https://itunes.apple.com/us/app/google-authenticator/id388497605?mt=8
2. Para conseguir ler o *QR code* na tela, será necessário ter uma tela maior do que a console padrão do Virtualbox — faça login via ssh na máquina *LinServer-G* usando o PuTTY ou Cygwin e vire superusuário usando o comando su.

```
fbs@FBS-DESKTOP ~
$ hostname
FBS-DESKTOP
```

```
fbs@FBS-DESKTOP ~
$ ssh aluno@172.16.1.10
Password:
Last login: Thu Sep  6 09:31:40 2018 from 172.16.1.254
aluno@LinServer-A:~$
```

```
aluno@LinServer-A:~$ su -
Password:
root@LinServer-A:~#
```

3. Instale o pacote que implementa suporte ao Google Authenticator na biblioteca PAM:

```
# hostname
LinServer-A
```

```
# apt-get install libpam-google-authenticator
```

4. Depois, insira a linha `auth required pam_google_authenticator.so` imediatamente após a linha 4, `@include common-auth`, no arquivo `/etc/pam.d/sshd`:

```
# nano /etc/pam.d/sshd
(...)
```

```
# head -n5 /etc/pam.d/sshd | grep -v '^#' | sed '/^$/d'
@include common-auth
auth  required  pam_google_authenticator.so
```

5. Configure o `ssh` para permitir autenticação via *challenge-response*, alterando a diretiva `ChallengeResponseAuthentication` no arquivo `/etc/ssh/sshd_config` (linha 49). Feito isso, não esqueça de reiniciar o daemon do `ssh`.

```
# nano /etc/ssh/sshd_config
(...)
```

```
# grep '^ChallengeResponseAuthentication' /etc/ssh/sshd_config
ChallengeResponseAuthentication yes
```

```
# systemctl restart ssh
```

6. Agora, na máquina *LinServer-G*, execute **como um usuário não-privilegiado** (como o usuário `aluno`) o comando `google-authenticator`.

   Tabela 1. Opções do google-authenticator

Pergunta	Opção
Do you want authentication tokens to be time-based?	y
Do you want me to update your "/home/aluno/.google_authenticator" file?	y
Do you want to disallow multiple uses of the same authentication token?	y
Increase token window from default size of 1:30min to about 4min?	y
Do you want to enable rate-limiting?	y

7. Abra o aplicativo *Google Authenticator* em seu celular e clique no `+` vermelho no canto inferior direito da tela. Em seguida, clique em *Scan a barcode* e leia o *QR code* gerado no passo (6). Na tela principal, deverá surgir uma nova linha com seis dígitos (que serão re-gerados a cada 30s) e o identificador `aluno@LinServer-G`.

8. Verifique que a hora atual do servidor está correta. Como configuramos o NTP na sessão 6, é provável que esteja tudo correto, mas a *timezone* pode estar desconfigurada, como mostrado abaixo:

   ```
   $ date
   Thu Sep  6 09:40:05 EDT 2018
   ```

   Se esse for o caso, rode o comando `dpkg-reconfigure tzdata` como usuário `root`. Escolha *America* > *Sao_Paulo* (ou outra *timezone*, se for esse o caso). Verifique que o relógio foi corrigido:

   ```
   # dpkg-reconfigure tzdata

   Current default time zone: 'America/Sao_Paulo'
   Local time is now:      Thu Sep  6 10:42:27 BRT 2018.
   Universal Time is now:  Thu Sep  6 13:42:27 UTC 2018.
   ```

```
# date
Thu Sep  6 10:42:36 BRT 2018
```

9. Perfeito, tudo pronto. **NÃO** feche a sessão `ssh` atual, pois em caso de erros poderá ser necessário verificar alguns arquivos. Em lugar disso, abra uma nova sessão `ssh`, como usuário `aluno`, para a máquina *LinServer-G*. No *prompt Verification code*, informe o código temporizado indicado pelo aplicativo instalado em seu celular.

```
fbs@FBS-DESKTOP ~
$ hostname
FBS-DESKTOP
```

```
fbs@FBS-DESKTOP ~
$ ssh aluno@172.16.1.10
Password:
Verification code:
You have mail.
Last login: Thu Sep  6 10:32:40 2018 from 172.16.1.254
aluno@LinServer-A:~$
```

```
aluno@LinServer-A:~$ hostname
LinServer-A
```

```
aluno@LinServer-A:~$ whoami
aluno
```