PGP

Trabalho 3 - Segurança em Computação (INE5429)

Universidade Federal de Santa Catarina

PGP

Trabalho 3 - Segurança em Computação (INE5429)

Universidade Federal de Santa Catarina

Felipe de Campos Santos

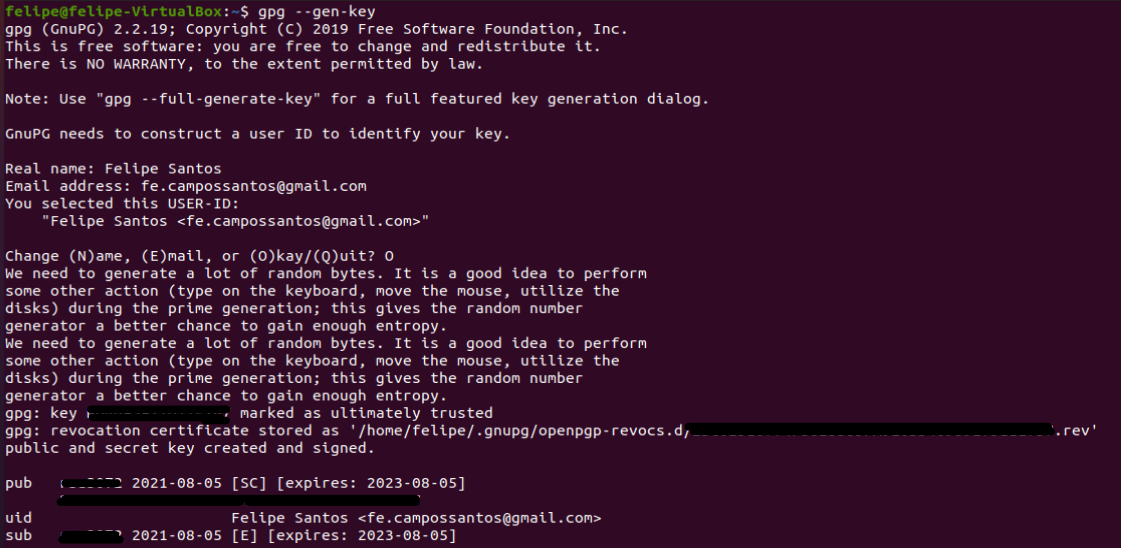
17200441

06/08/2021

# 1~3) Criar certificado GPG

Para criar o certificado GPG, usaremos o gpg no Ubuntu 20.04.2 LTS em VM (usando Oracle VirtualBox Manager) e usando como guia o [documento da Oracle.](https://blogs.oracle.com/wssfc/how-to-generate-pgp-keys-using-gpg-145-on-linux)

## Criando as chaves



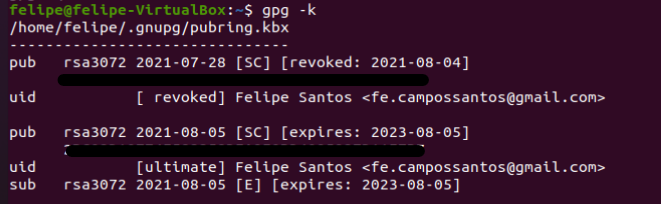
Depois de gerar as chaves, usando o comando

| gpg --gen-key |
| --- |

vamos adicionar uma subchave para podermos encriptar documentos:

| gpg --edit-key 'Felipe Santos' |
| --- |

Agora, se listarmos as chaves que tenho, aparecem duas. A primeira, criei quando estava aprendendo a mexer com o gpg, e foi revogada antes da escrita deste trabalho. A segunda é a atual:



KeyID da chave é *0x7D11E7B7*

## Colocando a chave no servidor e revogando-a

Antes de colocar a chave criada acima, deixo aqui um print do keyserver do RNP contando com uma chave anterior minha já revogada:

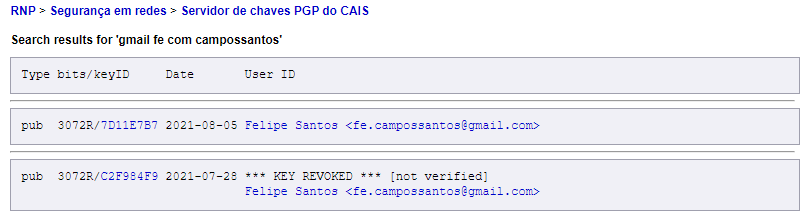


Agora, em frente.

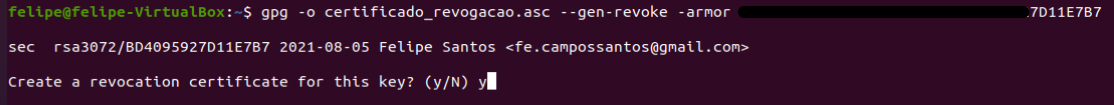
Vamos enviar a chave gerada para o servidor:



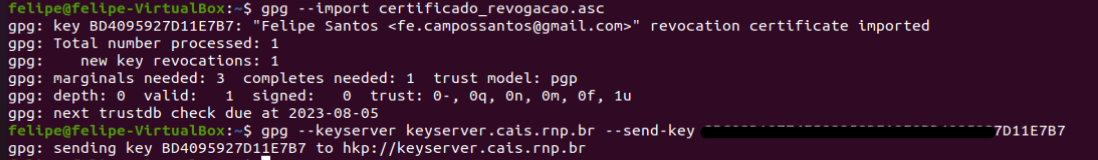
feito isso, a chave foi enviada para o servidor e podemos checar ela aqui, junto com a já revogada (possível distingui-las pela data):



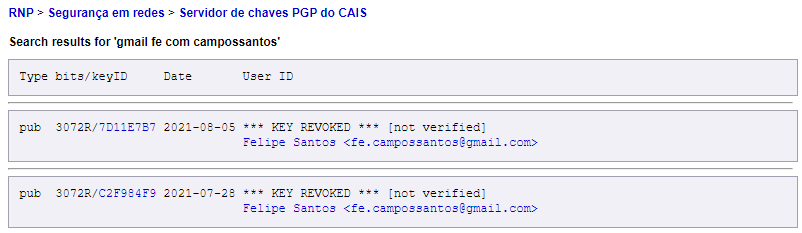
Agora, vamos revoga-lo. Para isso, no CLI, vamos gerar o certificado de revogação:



tendo o certificado gerado, vamos importa-lo no GnuPG e envia-lo ao keyserver do RNP novamente:

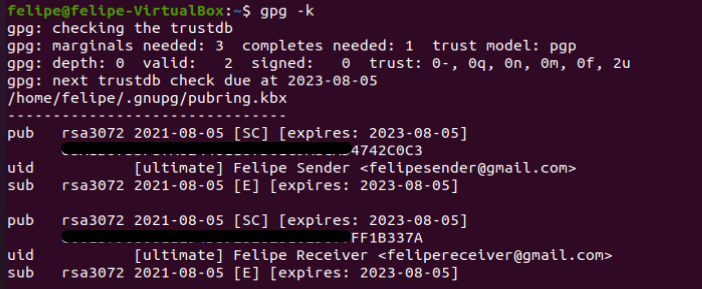


e se checarmos o keyserver do RNP agora, minha key deve aparecer como revoked (revogada):

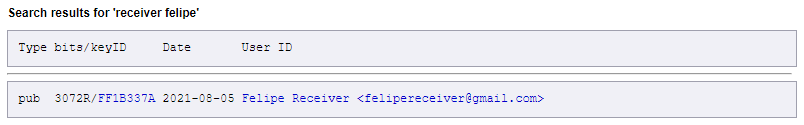


## Assinando um certificado GPG

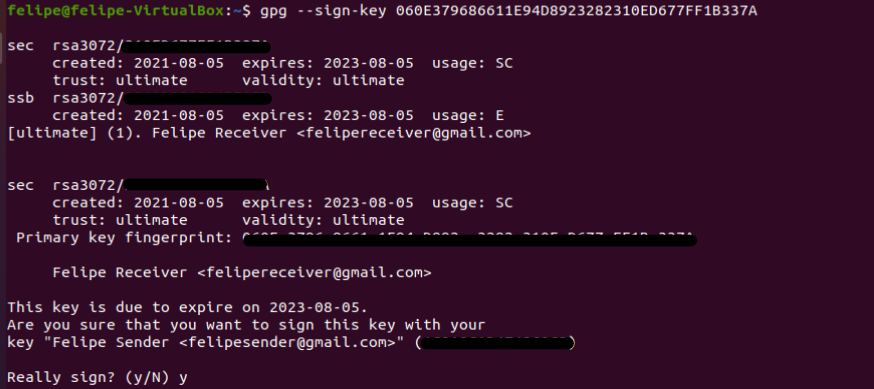
Para esse passo, foram criadas duas chaves no meu nome: Felipe Sender e Felipe Receiver, exemplificando a entidade que enviará uma mensagem e a que receberá.



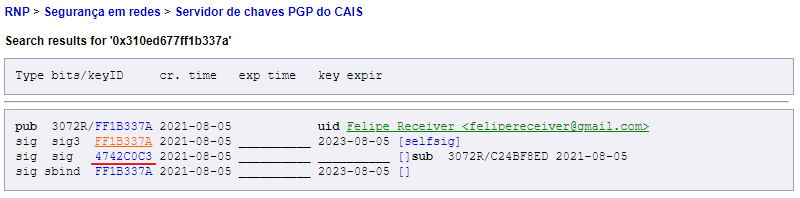
Sender assinará o certificado do Receiver. Para isso, primeiro colocaremos a chave de Receiver no keyserver do RNP, seguindo os mesmos passos usados anteriormente.



E agora, usando a chave Felipe Sender, assinamos a chave acima:



e enviamos para o RNP novamente. Agora, o RNP vai mesclar a chave adicionada anteriormente com essa que assinamos e enviamos. Agora, ao examinarmos a chave, vemos que nela está presente tambem a assinatura que fizemos:



E para revogarmos, vamos editar a chave de Felipe Receiver que temos localmente, e revogar a assinatura que fizemos à ela com a chave Felipe Sender. Para isso, entramos com o comando

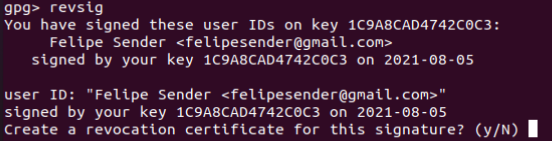
| gpg --edit-key Felipe Receiver |
| --- |

Aqui pode ficar confuso, pelo fato de que eu criei as duas chaves localmente, mas vamos lembrar que Receiver representa uma pessoa terceira (quem vai receber minhas informações). Vamos editar a chave *dela* que temos localmente, e revogar da chave *dela* a *nossa* assinatura.

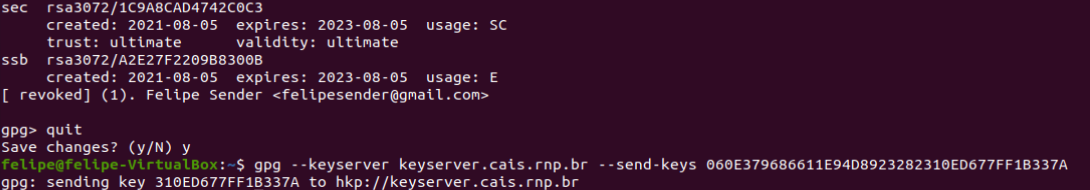
O comando acima abre o terminal gpg, e nele devemos entrar o comando *revsig*

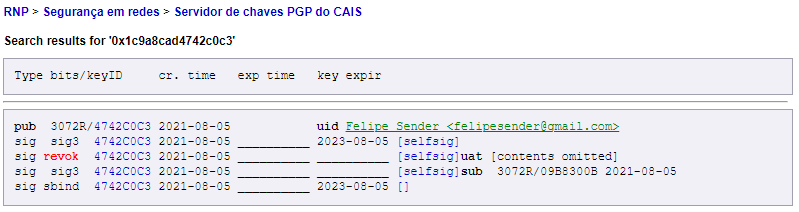
| gpg revsig |
| --- |

e temos:



agora basta criarmos a revogação e enviar novamente ao keyserver, que se atualizará.



****

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**4) O que é o anel de chaves privadas? Como este está estruturado? Na sua aplicação GPG onde este anel de chaves é armazenado? Quem pode ser acesso a esse porta chaves?**

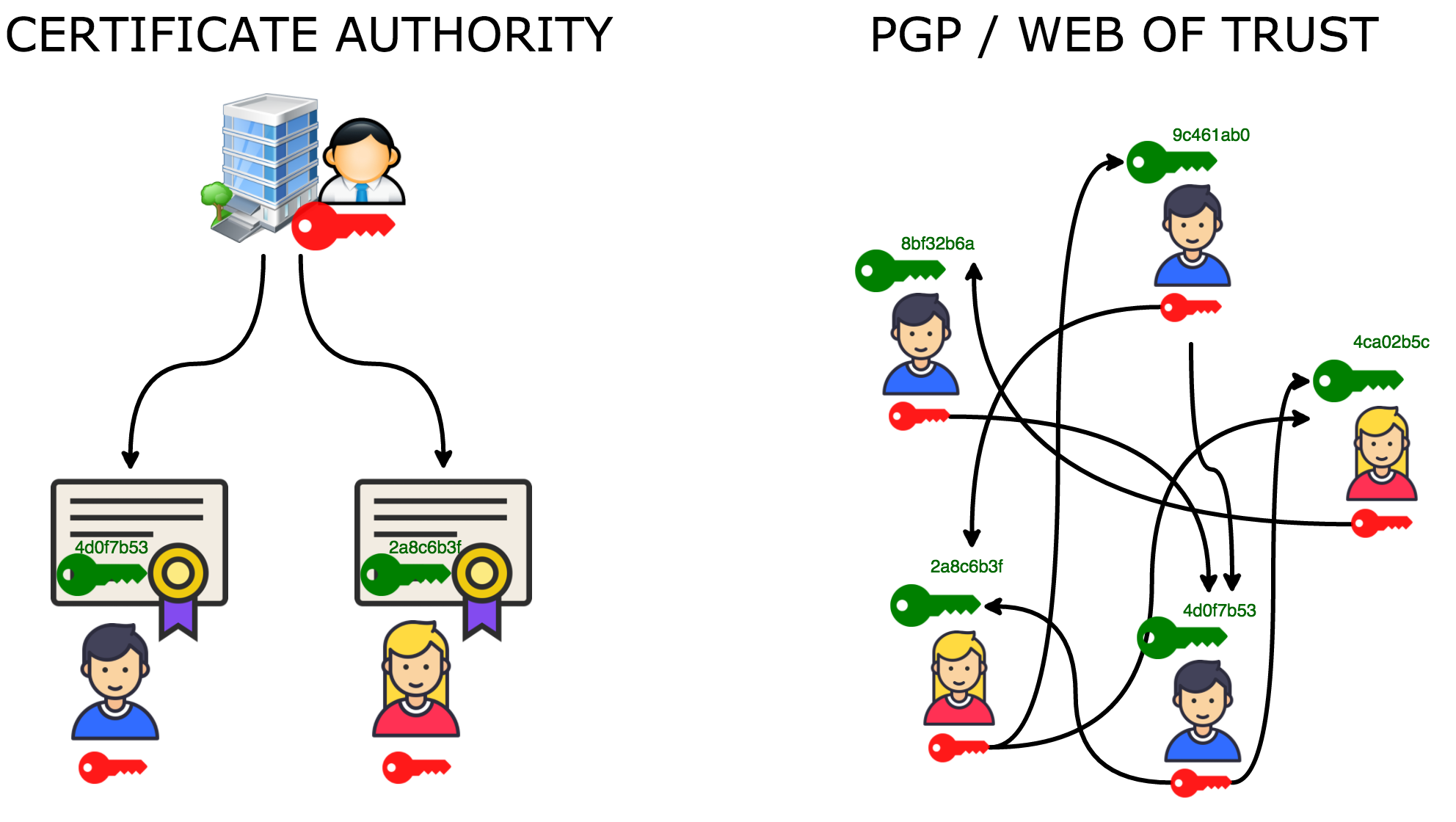
O anel de chaves privadas é uma estrutura de dados que contem um ou mais pares de chave pública e privada. O anel aceita mais que um par para que os usuários possam alterar suas chaves públicas sem que isso invalide as mensagens prévias, sendo construidas ou enviadas.

A aplicação PGP salva as chaves em dois arquivos no disco rígido (o GnuPG salva as chaves na pasta “.gnupg” na raiz /home do usuário.), de forma criptografada, e apenas o usuário dono da chave privada tem acesso ao anel, pois é nele que ficam todas as chaves privadas desse usuário. Ou seja, se o usuário perder esse anel, fica impossível descriptografar qualquer documento criptografado para aquelas chaves.

**5) Qual a diferença entre assinar uma chave local e assinar no servidor?**

Uma chave assinada localmente não pode ser exportada para um keyserver ou enviada a outra pessoa, ou seja, é um jeito de confiar na chave apenas localmente, sem colocar ela como confiável por ti num keyserver.

**6) O que é e como é organizado o banco de dados de confiabilidade?**

****

retirado de joenio.me/

Diferente de uma autoridade certificadora, que é uma maneira centralizada de garantir confiabilidade, o PGP utiliza a rede de confiabilidade, que tem suas qualidades e defeitos.

A autoridade certificadora funciona como uma pessoa que dizem que é confiável, e você também confia nela, ou seja, se ela te diz que a Ana é confiável, você passa a confiar na Ana. Isso traz uma facilidade na relação de confiança de terceiros, porém isso também pode ser conveniente para terceiros com interesses diferentes dos seus (hackers, governo, grandes empresas).

Já na rede de confiabilidade, você escolhe em quem confiar, e a partir do momento que você confia na Ana, você também confia nas pessoas em quem a Ana confia, como por exemplo o Bob, e se você confia no Bob, você também confia nas pessoas em que o Bob confia, e assim em diante, além de poder diferencias o “nível” da sua confiança nesses terceiros

**7) O que são e para que servem as sub-chaves?**

Quando existia apenas uma chave por usuário, o termo “chave” fazia sentido, pois só havia aquela. A partir do momento que foi possível ter mais de uma chave por usuário, surgiu a noção de anel de chaves, junto com o termo subkey, que nada mais é que uma chave comum, ou seja, todas as “chaves” de um anel de chaves na verdade são sub-chaves.

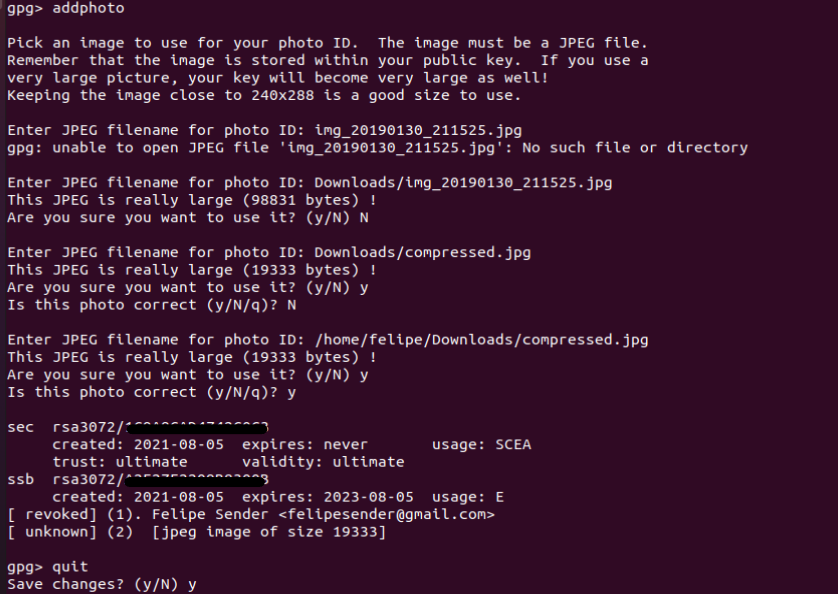
Exemplificando, eu tenho 4 chaves no meu anel de chaves: C1, C2, C3, C4.

Para o PGP (GnuPG), a minha chave pública é a minha chave mais antiga, ou seja, o conjunto de chaves vai ser chamado de chave C1, apesar delas todas estarem presentes (junto com outros dados, como certificados, assinaturas e etc)

Logo, as sub-chaves server para o mesmo fim que as chaves, apesar de serem usadas para fins a mais curto prazo. Exemplo, ter uma chave principal que é guardada com mais segurança offline e com validade longa, e subchaves para uso do dia a dia, que são mais fáceis de serem revogadas, logo fica melhor dessas subchaves serem mais abrangentemente usadas.

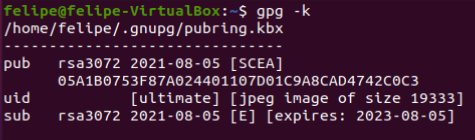
**8) Coloque sua foto ( ou uma figura qualquer ) que represente você em seu certificado GPG.**

Foi adicionada uma foto, porém não foi possível visualiza-la no RNP



Porém o tamanho da chave (em ASCII) ficou muito maior, possivelmente porque a imagem esta sendo guardada junto.

Pelo terminal é possível ver que a imagem está associada à chave:



**9) O que é preciso para criar e manter um servidor de chaves GPG, sincronizado com os demais servidores existentes?**

Para criar um servidor de chaves “é fácil”, exitem tutoriais na internet de como criar um usando, por exemplo, [security/pks](https://www.freshports.org/security/pks), que basicamente cria um banco de dados local que sincroniza com keyservers de preferência. Isso geralmente é feito por quem tem um interesse específico em ter um servidor de chaves privado, por qualquer motivo que seja (uso interno, maior controle, etc).

Já a sincronização entre os servidores pode ser feita de duas maneiras:

1. A maneira antiga - por email
2. A maneira atual - protocolo SKS

**- Por email:**

Cada servidor tinha sua lista de sincronização, formando uma rede:

* O servidor A conectava com o B e o C
* O servidor B conectava com D e E

Com um pouco de grafos em mente, podemos ver que o servidor A tinha uma conexão com o servidor E por meio de B. Claro que na prática a escala era bem maior.

Dessa maneira, cada vez que um servidor recebia um informação (chave) nova, ele a repassava para a sua lista de sincronização via email, então os servidores que recebiam esse email repassavam para a lista de sincronização deles, e assim em diante, mantendo a rede atualizada.

Com o crescimento no número de servidores de chave, esse método deixou de ser escalável, e assim migraram para o

**- Protocolo SKS**

A cada período de tempo, os servidores “fofocam” entre sí, de uma maneira mais eficiente: é usada uma *arvore de partição* (partition tree) que realiza uma busca em suas chaves e faz uma troca de informação entre os servidores apenas das informações que foram alteradas, assim diminuindo o payload de informação repassado (e aumentando a eficiência).

**10) Dê um exemplo de como tornar sigiloso um arquivo usando o GPG. Envie esse arquivo para um colega e que enviar para você outro arquivo cifrado. Você deve decifrar e recuperar o conteúdo original.**

É possível encriptar arquivos com o gpg pelo CLI usando o parâmetro *--encrypt*. Assim os dados do documento só poderão ser lidos por quem puder decriptar (usando *--decrypt*), ou seja, quem for dono da chave privada correspondente à chave pública usada.

Para exemplificar, usaremos as chaves já criadas (Sender e Receiver), por isso não faremos os passos de exportação e importação das chaves para um keyserver ou envio delas por alguma mídia, já que temos as duas localmente. Em um uso real, seria necessário que a pessoa que vai receber a mensagem me passasse a chave pública dela. Então, nosso processo será:

Sender quer enviar uma mensagem secreta para Receiver. Logo, Sender usará a chave pública de receiver para criptografar a mensage, e Receiver usará sua chave privada para descriptografar a mensagem e poder lê-la.

A mensagem secreta será um arquivo de texto:



para cifrar a mensagem, usaremos o comando abaico

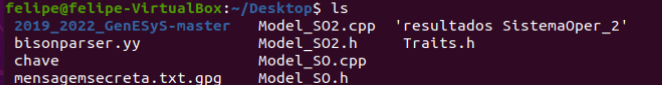
| gpg -e -r "Felipe Receiver" mensagemsecreta.txt |
| --- |

aonde *-e* é o comando para “encypt”, *-r* define quem sera o “recipient” da mensagem, seguido de seu user ID (nesse caso, Receiver) e então seguido da mensagem que queremos cifrar.

Com isso, temos um arquivo novo (mensagemsecreta.txt.gpg):



E podemos ver que ela ficou ilegível, e agora vamos apagar o arquivo original:

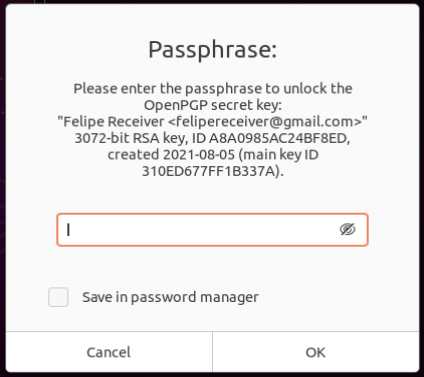


Apesar da presença de arquivos de outras matérias, podemos ver que o arquivo original foi apagado.

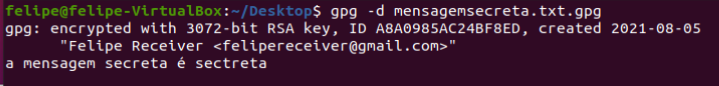
Com isso, podemos realizar a decifragem do arquivo. Usaremos o seguinte comando:

| gpg -d mensagemsecreta.txt.gpg |
| --- |

Onde *-d* é para “decrypt”, seguido do arquivo que se deseja decifrar. Isso pedirá a senha que usamos para criar a chave:

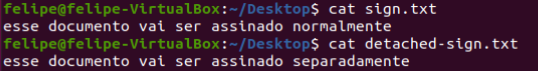


E como é possível ver na imagem, está pedindo a senha para a chave de Receiver, que era nossa intenção desde o começo. Entrando com a senha, o gpg decifra a mensagem usando a chave privada de Receiver e nos mostra qual era a mensagem:



**11) Mostre um exemplo de como assinar um arquivo ( assinatura anexada e outro com assinatura separada ), usando o GPG. Envie uma mensagem assinada para um colega. Esse colega deve enviar para você outra mensagem assinada. Verifique se a assinatura está correta.**

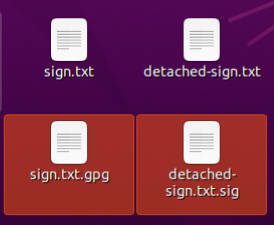
Para esse exercício, usaremos dois documentos diferentes, cada um representando o tipo de assinatura que será feita



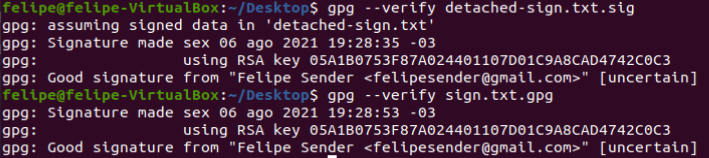
Agora, assinamos os dois documentos, cada um com seu respectivo tipo de assinatura



Com isso, temos mais dois documentos novos:



E verificamos as duas assinaturas, tendo como retorno a válidez e algumas informações sobre elas:

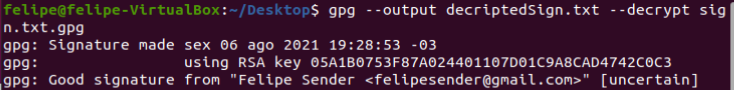


Vale lembrar que, para a assinatura separada, a validez só foi confirmada pois o documento original esta ali junto. O que aconteceria se tentassemos checar a validez sem ter o documento original?

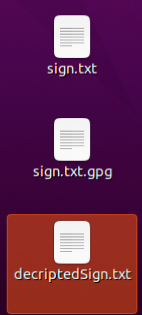


O GPG não tem como checar a validez pois não tem o documento original. A assinatura separada serve principalmente para quando você precisa atestar a veracidade de algum documento, assinando-o, mas sem impedir que a pessoa que vai receber o documento possa ler ele sem precisar decifrar.

Podemos ainda, rapidamente, checar a assinatura do documento assinado e decifra-lo, obtendo assim não só a validez da assinatura mas também o dado:



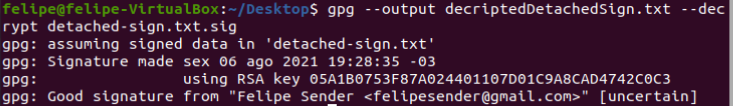
Com isso, obtemos mais um documento, resultado da decifragem:



e, para confirmar, o conteúdo dele:



Mas, o que aconteceria se tentassemos a mesma coisa com a assinatura separada?



O GPG apenas checou a assinatura, retornando válida, mas não houve criação de nenhum novo documento. O que faz todo sentido, pois a assinatura estava *separada* dos dados, então podemos apenas verifica-la, considerando que o caso de uso para esse tipo de assinatura é que o documento original seja enviado junto dela, assim não precisando decifrar ele.

# 

# Referências

https://users.ece.cmu.edu/~adrian/630-f04/PGP-intro.html

<https://www.gnupg.org/gph/en/manual.html>

<http://slides.com/mricon/pgp-web-of-trust>

<https://joenio.me/gnupg>

<https://www.enigmail.net/forum/viewtopic.php?f=3&t=375>

<https://security.stackexchange.com/questions/76940/what-exactly-is-a-subkey>

https://blogs.oracle.com/wssfc/how-to-generate-pgp-keys-using-gpg-145-on-linux

<http://eriberto.pro.br/wiki/index.php?title=Como_preparar-se_para_uma_festa_de_assinatura_de_chaves_GPG>

<https://www.inf.ufsc.br/~bosco.sobral/extensao/material-cripto-seg/gris-2004-t-001.pdf>

<https://www.inf.ufsc.br/~bosco.sobral/ensino/ine5630/cripto-seg.html>

<https://www.inf.ufsc.br/~bosco.sobral/ensino/ine5630/>

<https://slideplayer.com.br/slide/3381426/#:~:text=17%20Anel%20de%20Chave%20Privada,sendo%20preparadas%20ou%20em%20transito>

<https://gist.github.com/F21/b0e8c62c49dfab267ff1d0c6af39ab84>

<https://security.ias.edu/how-revoke-gnupgpgp-signature-key>

<https://forums.freebsd.org/threads/setting-up-your-own-openpgp-keyserver-why-how.59920/>

<https://superuser.com/questions/861723/are-gpg-keyservers-synchronised-with-each-other-pgp-mit-server-too>

<https://sks-keyservers.net/>

<https://security.stackexchange.com/questions/82045/how-do-pgp-keyservers-synchronize-the-keys>

<https://www.gnupg.org/gph/en/manual/x457.html>

https://stackpointer.dev/?p=108