Último fascículo do tutorial sobre criptografia

Criptografia, teoria e prática: parte 5

No artigo final desta série, consolidam-se as práticas para envio e recepção de arquivos e mensagens criptografados e assinados.

por Marcio Barbado Jr. e Tiago Tognozi

Finalmente, chegamos ao último artigo desta série. Para finalizar, veremos como aplicar todos os conceitos e práticas explicados até aqui, com o intuito de assinar e cifrar mensagens para enviá-las com privacidade e garantia de autenticidade, assim como decifrar essas mesmas mensagens e conferir sua assinatura no recebimento.

Encriptar para enviar

Lembram-se do exemplo de Manuel e Luiza? Um já possui a chave pública do outro, pois estão sempre trocando arquivos confidenciais.

As redes da matriz e da filial são interligadas por uma VPN, formando uma só rede. E eis que Manuel precisa disponibilizar a Luiza uma importante planilha.

Como de costume, ele irá encriptar tal arquivo e então colocá-lo no servidor de arquivos da rede, acessível a ambos. Antes de enviar a planilha para o servidor, a primeira ação a ser tomada por Manuel é encriptá-la com a chave pública de Luiza. Via linha de comando, isso seria feito assim:

```
$ gpg --recipient “Luiza” \
   --output \
   “Faturamento-Julho09.ods.gpg” \
   --encrypt \
   “Faturamento-Julho09.ods”
```

Ou seja, Manuel diria ao GnuPG para utilizar a chave de Luiza para gerar o arquivo `Faturamento-Julho09.ods.gpg`, encriptado, resultante do original, chamado `Faturamento-Julho09.ods`.

Recomenda-se que o arquivo a ser gerado possua o nome completo do original, incluindo extensão, que no caso é `.ods`, acrescido da nova extensão, `.gpg`.

Manuel acabou de criar o arquivo `Faturamento-Julho09.ods.gpg`, que é um arquivo encriptado pronto para ser levado ao servidor de arquivos. A única pessoa capaz de decriptar facilmente a planilha é Luiza, pois pertence a ela a chave pública utilizada na encriptação.

Múltiplos destinatários

E se Manuel precisar disponibilizar a planilha protegida para outros funcionários além de Luiza? É certo que ele deve possuir cadastradas as chaves públicas de cada um dos destinatários, mas o GnuPG torna desnecessária a repetição de comandos.

Manuel fornece um único comando (longo, é verdade) especificando a lista de pessoas que poderão decriptar o arquivo. E a única diferença com relação ao comando apresentado anteriormente é que, agora, cada destinatário exige uma opção `--recipient` adicional, por exemplo:

```
$ gpg --recipient “Luiza” \
   --recipient “Marcio” \
   --recipient  “Tiago” \
   --recipient “Manuel” --output \
   “Faturamento-Julho09.ods.gpg” \
   --encrypt Faturamento-Julho09.ods
```

Pode parecer estranho o fato de Manuel ter especificado a si mesmo nesse comando. Porém, cabe desta-

car que, ao receber o comando com mais de uma opção `--recipient`, o algoritmo de encriptação empregado pelo GPG utiliza as chaves públicas de todos os destinatários envolvidos. Com isso, ele gera um arquivo encriptado cujo tamanho sofre pequenos acréscimos, de forma proporcional à quantidade de pessoas listadas, e apenas as chaves privadas daquelas pessoas conseguirão decifrá-lo.

Portanto, já que irá possibilitar a decriptação da planilha a mais de um destinatário, e ainda considerando que esse procedimento não produz grandes aumentos no tamanho do arquivo encriptado, Manuel inclui a si mesmo na lista, possibilitando que sua chave privada também consiga decifrar o arquivo.

Mas atenção: na verdade, mesmo nos casos em que precisar disponibilizar o arquivo a um único destinatário, é interessante acrescentar uma opção `--recipient` adicional para si mesmo; trata-se de uma medida de segurança que permitirá a Manuel acessar o conteúdo do arquivo facilmente caso o original (decriptado) não esteja disponível.

Decriptar ao receber

Esta seção descreve os procedimentos necessários para se decriptar arquivos assimetricamente, ou seja, utilizando uma chave privada.

Continuando com o exemplo de Manuel e Luiza, ela deseja restabelecer o conteúdo original do arquivo `Faturamento-Julho09.ods.gpg` que Manuel deixou no servidor de arquivos.

Então, ela copia o arquivo encriptado para um diretório local, no qual fornece o seguinte comando:

```
$ gpg --decrypt-files \
     "Faturamento-Julho09.ods.gpg"
```

que solicitará a *passphrase* para utilizar sua chave privada e então decriptar o arquivo, transformando-o de volta numa planilha que possa ser aberta e utilizada por Luiza: `Faturamento-Julho09.ods`.

Listagem 1: Arquivo mara.txt

```
Luiza, ouvi rumores de que você será promovida.

  Ass.: Manuel dos Santos
```

Listagem 2: Arquivo mara.txt.asc

```
---BEGIN PGP SIGNED MESSAGE---
Hash: SHA1

Luiza, ouvi rumores de que você será promovida.

  Ass.: Manuel dos Santos
---BEGIN PGP SIGNATURE---
Version: GnuPG v1.4.7 (MingW32)

iD8DBQFKiz6VFTizx1Qd664RApsZAKCR87GzrdUhi4vnUaJD6kMzOBU5NQCgkmcq
LGD5Jn3u4qbqJfcM2KGjp7g=
=8RLi
---END PGP SIGNATURE---
```

Considerando arquivos de texto simples, encriptados, é possível fornecer um comando via terminal que permita a visualização de seus conteúdos decriptados sem que essa ação gere um arquivo decriptado. Então, supondo o arquivo encriptado `faturamento.txt.gpg`, o comando para a visualização de seu conteúdo no terminal seria:

```
$ gpg --decrypt "faturamento.txt"
```

Assinar para enviar

Esta seção apresenta o comando `--clearsign`, que atribui autenticidade a texto simples, utilizando a chave privada do remetente. O destinatário poderá testar o caráter genuíno do texto recebido somente se possuir a chave pública do remetente.

Suponha que Manuel crie um arquivo texto chamado `mara.txt` cujo conteúdo é mostrado na **listagem 1**.

Oras, qualquer um poderia se passar por Manuel e criar o arquivo acima. O trecho *Ass.: Manuel dos Santos* não é realmente uma assinatura. Não prova que o autor daquele arquivo seja ele de fato.

Manuel, ciente disso, decide que, antes de enviar o arquivo, irá também assiná-lo digitalmente com o comando:

```
$ gpg --local-user [KEY_ID] \
     --clearsign "mara.txt"
```

ou simplesmente:

```
$ gpg --clearsign "mara.txt"
```

A primeira opção de comando é útil em situações nas quais se possui mais de um par de chaves - por exemplo, um para o email profissional e outro para o pessoal.

Com o comando fornecido, o GnuPG irá utilizar a chave privada de Manuel e, portanto, solicitará sua passphrase.

O conteúdo do arquivo `mara.txt` será então digitalmente assinado e o resultado será escrito em um novo arquivo, `mara.txt.asc`, criado automaticamente, que ficará com o conteúdo mostrado na **listagem 2**.

O arquivo de extensão `.asc` está pronto para ser enviado. Não está encriptado, apenas assinado (não há privacidade, apenas autenticidade). Seu conteúdo pode ser acessado (lido) normalmente. É interessante destacar as três formas tradicionais que Manuel pode utilizar para fazer com que a mensagem digitalmente assinada chegue a Luiza. Ele pode:

- disponibilizar o arquivo `mara.txt.asc` no servidor de arquivos;
- enviar o arquivo anexado em uma mensagem eletrônica; ou
- copiar e colar o conteúdo do arquivo no corpo de uma mensagem eletrônica a enviar.

A última opção pode gerar erros, dependendo dos clientes de email envolvidos na comunicação. Muitos desses erros se manifestam no cliente do destinatário, durante a verificação da assinatura digital. Tais falhas podem ser evitadas, configurando-se devidamente a codificação da saída do cliente remetente. Por exemplo: uma conta do Gmail, utilizada via Web, exigirá mensagens enviadas no formato UTF-8 (Unicode).

Verificar assinatura ao receber

Esta seção pressupõe que o destinatário já possua a chave pública do remetente (confira a seção *Cadastrar chaves públicas* do quarto artigo desta série [1]).

O comando `--verify` permite que o lado receptor de um arquivo digitalmente assinado possa conferir a autenticidade desse arquivo utilizando a chave pública da entidade que o enviou.

Luiza recebe então o arquivo `mara.txt.asc`. Para saber se este veio mesmo de Manuel, verifica a assinatura digital com o comando:

```
$ gpg --verify "mara.txt.asc"
```

Com isso, o GnuPG irá utilizar a chave pública de Manuel para testar a assinatura e relatar, na saída padrão do terminal, se é válida ou não (**listagem 3**).

Caso a mensagem tenha chegado a Luiza no corpo de um email, tudo o que ela tem a fazer é abrir um editor de texto com um novo arquivo texto em branco, copiar e colar o conteúdo do corpo da mensagem recebida no arquivo e salvá-lo com um nome, que pode ser `mara.txt.asc`. Daí em diante, basta utilizar o comando `--verify` mostrado nesta seção.

Emails

A criptografia assimétrica aplicada à troca de mensagens eletrônicas resguarda a estabilidade do relacionamento entre parceiros comerciais e a privacidade das pessoas por trás de tudo isso. Ela é particularmente interessante pois envolve a automação das ações de encriptação, decriptação, assinatura e verificação.

Um email pode basicamente possuir dois pedaços a proteger:

- o corpo: basicamente texto; e/ou
- o(s) arquivo(s) anexado(s).

Alguns programas clientes de email, configurados devidamente, podem automatizar todo o processo criptográfico tanto no envio quanto no recebimento. E aqui cabe menção à necessidade de se utilizar codificação Unicode nos referidos programas.

O KMail e o Evolution já possuem o GnuPG integrado. Note, entretanto, que os exemplos de Manuel e Luiza também servem para ilustrar situações nas quais o primeiro deseja enviar uma mensagem eletrônica contendo um arquivo encriptado, anexado. E nessa situação, o processo envolve pouca automação, já que a encriptação e a decriptação devem ser realizadas manualmente pelos usuários, respectivamente antes de enviar e depois de receber o email.

Envio

O tenebroso e inseguro panorama que a Internet vem revelando ao mundo tem forçado usuários a adotarem práticas mais responsáveis no envio de mensagens eletrônicas.

Recomenda-se sempre redigir emails **sem** formatação. Esta medida, além de contribuir com a comunicação criptografada, combate o terror causado por mensagens eletrônicas formatadas em HTML. Trata-se de uma criteriosa "bandeira", hoje defendida formalmente pela campanha "ASCII ribbon campaign", que toca uma questão de segurança bastante atual e latente, dadas as novas ameaças da Web, representadas em parte por textos com formatação HTML que certas vezes, ao ser carregados, fazem chamadas a conteúdos remotos.

A formatação HTML é perigosa e desrespeitosa no sentido de abusar da confiança e do desconhecimento técnico do destinatário. Muitas organizações já estão bloqueando emails dessa natureza.

Devidamente configurado, o cliente de email que Manuel usa emprega texto simples com codificação Uni-

Listagem 3: Confirmação da assinatura do remetente

```
gpg: Signature made 08/18/09 20:51:49 using DSA key ID 541DEBAE
gpg: Good signature from "Manuel dos Santos (GnuPG test.) <msantos@
bdslabs.com.br>"
```

code para o corpo das mensagens. Suponhamos que ele queira enviar a Luiza um email cujo corpo possui um simples aviso, que deve ser digitalmente assinado.

O programa de envio infelizmente não faz isso automaticamente durante o envio. Então, Manuel deverá redigir o conteúdo do email em um editor de texto. Escrita a mensagem, ele a salva com formato ASCII no arquivo `reuniao.txt` (**listagem 4**).

O texto do arquivo ainda não está digitalmente assinado. Convém observar que o trecho:

```
Ass.: Manuel dos Santos
```

não é realmente uma assinatura. Não prova que o autor daquele texto seja de fato Manuel, que deve então assinar digitalmente o arquivo para que Luiza possa verificar (utilizando a chave pública do colega) se aquele aviso partiu da suposta fonte.

Manuel então fornece um comando em seu computador, que atribui autenticidade ao texto, assinando-o digitalmente:

```
$ gpg --local-user [KEY_ID] \
    --clearsign "reuniao.txt"
```

ou simplesmente:

```
$ gpg --clearsign "reuniao.txt"
```

Com qualquer um desses comandos, o GnuPG irá utilizar a chave privada de Manuel, e portanto solicitará sua *passphrase*. Manuel deve fornecê-la para que o conteúdo do arquivo `reuniao.txt` seja digitalmente assinado e o resultado seja escrito em um novo arquivo texto, `reuniao.txt.asc`, criado automaticamente, que terá o conteúdo mostrado na **listagem 5**.

Manuel deve então copiar e colar o conteúdo do arquivo texto de extensão `.asc` no corpo da mensagem eletrônica a ser enviada. Feito isso, basta efetivamente enviá-la.

Listagem 4: Arquivo reuniao.txt

```
Luiza, amanhã ocorrerá uma reunião com o Bráulio, aqui na matriz às
18:30.

  Ass.: Manuel dos Santos
```

Listagem 5: Arquivo reuniao.txt.asc

```
---BEGIN PGP SIGNED MESSAGE---
Hash: SHA1

Luiza, amanhã ocorrerá uma reunião com o Bráulio, aqui na matriz às
➥ 18:30.

  Ass.: Manuel dos Santos
---BEGIN PGP SIGNATURE---
Version: GnuPG v1.4.7 (MingW32)

iD8DBQFKkrGfFTizx1Qd664RAmJIAJ9twL6gZHF9nCgVx1hafY6H9GaPxQCfaHao
ZrgOebcxjkS82NTUW3G2qRw=
=H4Wj
---END PGP SIGNATURE---
```

Recepção

Caso o corpo da mensagem esteja encriptado e o programa de email que Luiza utiliza não possa decriptá-lo automaticamente durante o recebimento da mensagem, ela deverá:

- abrir um editor de texto e criar um novo arquivo, inicialmente em branco; e
- copiar o conteúdo encriptado do corpo do email para o novo arquivo texto recém-criado e salvá-lo com final `.txt.asc`, como em `recadinho.txt.asc`;

Luiza salva o arquivo (ainda encriptado) no diretório `meus_recadinhos/`. Então, dentro desse diretório, ela deve fornecer o seguinte comando:

```
$ gpg --decrypt-files \
    "recadinho.txt.asc"
```

e o GnuPG irá solicitar a ela sua passphrase para então decriptar o arquivo, transformando-o em `recadinho.txt`.

Claro que, se Manuel tiver utilizado outra chave pública que não a de Luiza, ela não conseguirá decifrar o tal "recadinho".

Tudo transcorre normalmente e Luiza decifra o recado, contendo um lembrete de Manuel para que ela não se esqueça de enviar uma foto em formato digital que havia prometido.

Então, Manuel recebe no dia seguinte um email de Luiza com um arquivo anexado, `biquini.jpg.gpg` , que está encriptado.

Ele salva o arquivo localmente e, com o seguinte comando, decripta-o:

```
$ gpg --decrypt-files \
    "biquini.jpg.gpg"
```

Agora, Manuel pode visualizar a foto, pois o comando que forneceu gerou o arquivo `biquini.jpg`, que não está mais protegido.

Solução de problemas

O interessante modelo colaborativo proposto pelo movimento FOSS, seja ele da linha "catedral" ou "bazar", permite, por exemplo, que seções como esta, as boas e velhas "soluções de problemas", possam evoluir.

Colabore com o desenvolvimento do GnuPG. Caso você encontre algum problema não abordado por este texto, existem ainda outros meios através dos quais é possível interagir com as comunidades ligadas às tecnologias aqui abordadas.

Mais do que apresentarem problemas e soluções, o referido modelo possibilita ao usuário a participação na busca por soluções de novos problemas que não sejam contemplados por qualquer outra documentação.

Caso você encontre algum problema interessante, estude-o. Caso obtenha uma boa solução, coloque-a "no papel", elabore-a com a maior didática possível em inglês e encaminhe-a para o FAQ do GnuPG.

Esse FAQ é mantido por David D. Scribner e recebe sugestões em inglês por meio do email faq@gnupg.org.

Eventualmente, caso você não se consiga resolver um determinado problema, consulte os arquivos da lista de discussão do GnuPG [2][3][4].

Se, mesmo recorrendo às listas de discussão, o bug persistir, envie um relatório ao BTS (*bug tracking system*) do GnuPG [5] e também um email para gnupg-bugs@gnu.org.

Conclusões

A criptografia sofreu notável evolução ao assimilar conceitos da teoria quântica. A então chamada "criptografia quântica" passou, portanto, a unir a matemática dos conceitos clássicos às leis da física.

Lembre-se: chaves públicas podem ser de conhecimento público.

Os paradigmas vigentes, referentes ao tratamento digital inadvertido de informações, passam uma falsa ideia de segurança pelo fato de que eventuais acessos não autorizados a estas são muitas vezes praticados sem que as vítimas sequer percebam a quebra de privacidade.

É possível abrir mão da selvageria e insegurança generalizadas que acometem o tráfego e o armazenamento de informações digitais. O risco de sua exposição será tão grande quanto a negligência de seus responsáveis.

Assim, uma vez mais, o conhecimento, livremente disseminado, é capaz de preencher lacunas sociais, civilizar e lapidar costumes, aprimorando-os.

Quanto aos servidores de chaves, é ilusória a percepção de que proporcionam alguma praticidade ao usuário. A praticidade que almeja produtividade é contra a segurança.

E ainda, referindo-se ao modelo colaborativo de desenvolvimento, proposto pelo movimento do Software Livre; as múltiplas instâncias disponibilizadas para solucionar problemas criam uma grande comunidade de colaboradores e fazem, nitidamente, as tecnologias de domínio público evoluírem mais rapidamente que as tecnologias fechadas. ■

Mais informações

[1] Marcio Barbado Jr. e Tiago Tognozi, "Criptografia: teoria e prática, parte 4": http://lnm.com.br/article/3405

[2] Listas de discussão do GnuPG: http://www.gnupg.org/documentation/mailing-lists.html

[3] Histórico da lista dos usuários do GnuPG: http://marc.theaimsgroup.com/?l=gnupg-users&r=1&w=2

[4] Histórico da lista dos desenvolvedores do GnuPG: http://marc.theaimsgroup.com/?l=gnupg-devel&r=1&w=2

[5] Rastreamento de bugs do GnuPG: https://bugs.g10code.com/gnupg/index

Sobre os autores

Marcio Barbado Jr. (marcio.barbado@bdslabs.com.br) e **Tiago Tognozi** (tiago.tognozi@bdslabs.com.br) são especialistas em segurança na BDS Labs (www.bdslabs.com.br).

Nota de licenciamento

Gostou do artigo?

Queremos ouvir sua opinião. Fale conosco em cartas@linuxmagazine.com.br

Este artigo no nosso site: http://lnm.com.br/article/3475