Apêndice B

gpg: Manual de sobrevivência.

O gpg (*GNU Privacy Guard*) é a implementação livre do *OpenPGP* como está definido no RFC 2440[1]. Na maioria das vezes compatível com o original pgp[2] constitui hoje o programa de referência que todos usam para codificar e assinar documentos . Sobre a utilização deste `software` existem hoje disponíveis diversos textos que tentam simplificar a sua aprendizagem [Gar94, Luc06, Opp00].

B.1 Como criar a nossa identidade?

Para começar a utilizar o gpg seja para assinar ou cifrar mensagens é necessário primeiro criar um par de chaves (chave-pública,chave-privada). Para tal basta usar a opção `-gen-key`. Segue-se a transcrição da criação de um par inicial de chaves. Para além de responder a perguntas triviais (como qual o nome verdadeiro do utilizador ou o seu endereço electrónico) as restantes opções podem ser respondidas com os valores por omissão (que neste caso fazem bastante sentido!).

Segue-se a transcrição de uma sessão de criação de um novo par de chaves com o gpg:

```
foo@khurt:~$ gpg --gen-key
gpg (GnuPG) 1.2.3; Copyright (C) 2003 Free Software Foundation, Inc.
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it
under certain conditions. See the file COPYING for details.

gpg: /home/foo/.gnupg: directory created
gpg: new configuration file ‘/home/foo/.gnupg/gpg.conf’ created
gpg: WARNING: options in ‘/home/foo/.gnupg/gpg.conf’ are not yet
active during this run
gpg: keyring ‘/home/foo/.gnupg/secring.gpg’ created
gpg: keyring ‘/home/foo/.gnupg/pubring.gpg’ created
Please select what kind of key you want:
   (1) DSA and ElGamal (default)
   (2) DSA (sign only)
   (5) RSA (sign only)
Your selection? 1
```

Vamos querer uma chave que nos permita tanto assinar como cifrar e decifrar mensagens, portanto a opção é a **1** (*DSA and ElGamal*)!

[1] Pode ser encontrado em ftp://ftp.rfc-editor.org/in-notes/rfc2440.txt

[2] Que começou por ser um programa de distribuição gratuita (mas nunca de distribuição livre!) e que hoje é um produto comercial (http://www.pgp.com/).

```
DSA keypair will have 1024 bits.
About to generate a new ELG-E keypair.
              minimum keysize is  768 bits
              default keysize is 1024 bits
    highest suggested keysize is 2048 bits
What keysize do you want? (1024)
```

O programa sugere sempre um tamanho de chave que é o razoável, apesar desse valor ter evoluido ao longo do tempo. Para aquilo que vamos aqui fazer 1024 bits é mais do que suficiente!

```
Requested keysize is 1024 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)
```

Não vale a pena criar uma chave provisória... Entretanto podemos começar a usar a chave para outros fins e para tal, uma chave sem limite de validade é muito mais cómoda!

```
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 Real Name, Comment and Email Address in this form:
    "Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"

Real name: Foo Bar
```

O nome real, aquele pelo qual nos reconhecemos no mundo "analógico" (que alguns insistem em chamar "mundo real"). É uma boa ideia colocar aqui o vosso verdadeiro nome. Depois de uma chave validada não é possível mudar o nome a ela associado... mas também a mudança legal de nome não é um operação assim tão frequente.

```
Email address: foo@khurt.ncc.up.pt
```

O nosso endereço electrónico. Mais tarde vai ser possível adicionar mais endereços (especialmente para aqueles com mais tendências esquizofrénicas) mas por agora usamos o endereço "principal".

```
Comment:
You selected this USER-ID:
    "Foo Bar <foo@khurt.ncc.up.pt>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? O
```

Última oportunidade de corrigir alguma asneira!...

```
You need a Passphrase to protect your secret key.
```

A chave privada vai estar guardada num ficheiro com permissões 0x0600, mas para além disso vai estar protegida por uma senha (*password*). Queremos proteger a utilização da nossa identidade a todo o custo, pelo que o programa não pede uma simples *password*, mas sim uma *passphrase*, ou seja algo mais longo e portanto mais difícil de "adivinhar". Mesmo assim não é aconselhavel guardar a nossa identidade gpg num computador em que suspeitemos que é provável que alguém possa copiar a nossa chave, e assim poder desencadear um ataque noutra(s) máquina(s).

```
We need to generate a lot of random bytes. It is a good idea to
perform some other action (type on the keyboard, move the mouse,
utilize the disks) during the prime generation; this gives the
random number generator a better chance to gain enough entropy.
++++++++++++++++++++....++++++++++++++++++++.+++++.+++++.+++++++
++++++++++++++++++..++++++++++.++++++++++++++++++++++++++++++++
+++++++....+++++>..++++++++++.................................
.............................+++++
```

Numa máquina *Unix* ligada à rede, com diversos processos a "correr", este pedido não faz tanto sentido...

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

pub  1024D/E8B20E42 2003-09-25 Foo Bar <foo@khurt.ncc.up.pt>
     Key fingerprint =
               A525 2CCF 8DF1 B791 1EA7  B6CE F338 0644 E8B2 0E42
sub  1024g/7D1349ED 2003-09-25
```

...e pronto! a nossa identidade está criada, ainda que ninguém nos conheça (por enquanto!). De qualquer forma, já existimos!

Toda a informação foi guardada no directório `/.gnupg`. Como não é possível recuperar a nossa identidade sem esta informação é mais do que aconselhável guardar uma cópia de segurança deste directório num dispositivo externo (como por exemplo num CD ou numa *"pendrive"*).

B.2 Dar a conhecer a nossa identidade

OK! Agora temos uma identidade digital... mas ninguém a conhece! Pior ninguém nos reconhece! O que há a fazer é distribuir a nossa chave **pública**.

Podemos fazer isso de duas maneiras: enviando a nossa chave a quem esteja interessado, publicando na nossa página *WWW*, etc..., ou enviando a nossa chave publica para um **servidor de chaves**.

Para usar o primeiro método temos que criar um ficheiro que contenha a nossa chave pública. O comando a invocar é:

```
foo@khurt:~$ gpg --export foo@khurt.ncc.up.pt > foo-key
```

Ou melhor ainda, para que o ficheiro esteja num formato "legível":

```
foo@khurt:~$ gpg --armor --export foo@khurt.ncc.up.pt > foo-key.txt
```

Se, pelo contrário, optarmos por colocar a nossa chave num servidor público o que temos que fazer é:

```
foo@kurt:~$ gpg --keyserver pgp.keyserver.nl --send-keys foo
```

Há que ter algum cuidado antes de optar por este segundo método: é muito fácil introduzir uma chave num servidor público, mas não é possível retirar uma chave uma vez lá colocada! Enquanto estamos a experimentar as facilidades do gpg é melhor deixar em paz os servidores públicos de chaves!

B.3 Passar a conhecer os outros

Para que todo o esquema de assinaturas e cifras faça algum sentido, para além de ter uma identidade digital tenho que conhecer as identidades (as chaves públicas) de outros interlocutores. Suponhamos que temos a chave pública de rvr@ncc.up.pt no ficheiro `rvr.txt`. Para incorporar esta chave no nosso rol de conhecidos basta fazer

```
foo@khurt:~$ gpg --import rvr.txt
gpg: key 18EA3457: public key "Rogerio Reis <rvr@debian.org>" imported
gpg: Total number processed: 1
gpg:               imported: 1
```

Agora se verificarmos que chaves conhecemos, obtemos:

```
foo@khurt:~$ gpg --list-keys
/home/foo/.gnupg/pubring.gpg
----------------------------
pub  1024D/E8B20E42 2003-09-25 Foo Bar <foo@khurt.ncc.up.pt>
sub  1024g/7D1349ED 2003-09-25

pub  1024D/18EA3457 2001-07-10 Rogerio Reis <rvr@debian.org>
uid                            Rogerio Reis <rvr@ncc.up.pt>
uid                            Rogerio Reis (gpg) <rvr@ncc.up.pt>
sub  1024g/B0191BB5 2001-07-10
sub  4096g/72A4A44D 2003-06-16
sub  4096R/F75B22BE 2003-06-16
```

De outra forma, e se não tivermos a chave pública já num ficheiro, podemos obter a chave pretendida de um servidor público. A forma mais simples é utilizar a sua interface WWW. Servidores com esse tipo de facilidades são por exemplo:

- http://math-www.uni-paderborn.de/pgp/
- http://pgp.mit.edu/
- http://pgp.zdv.uni-mainz.de/keyserver/

B.4 Assinar um documento

Estamos agora em condições de fazer a primeira coisa útil com este sistema: assinar um documento. Suponhamos que temos um texto `exemplo.tex` e que queremos assinar este documento, isto é, dar alguma garantia que fomos nós que assinámos o documento (como não é aconselhável assinar algo que não se leu) que tomamos conhecimento do seu conteúdo, e que este não foi entretanto modificado em nada. Para isso basta fazer:

```
foo@khurt:~$ gpg --sign -b exemplo.tex
```

A opção `-b` é para criar a assinatura num ficheiro separado (`exemplo.tex.sig`) deixando o ficheiro original inalterado. Se fosse pretendido que a assinatura fosse "legível", ou seja ASCII, bastaria usar a opção `-armor` ou `-clearsign` em vez de `-sign`. O gpg obriga a introduzir a *password* antes de assinar, evidentemente!

Para verificar que a assinatura é "boa" basta (supondo que conhecemos a chave pública do assinante) usar:

```
foo@khurt:~$ gpg --verify exemplo.tex.sig exemplo.tex
gpg: Signature made Sun Sep 28 23:22:30 2003 BST using
DSA key ID E8B20E42
gpg: Good signature from "Foo Bar <foo@khurt.ncc.up.pt>"
```

B.5 Cifrar e decifrar documentos

Se em vez de assinar, quisermos cifrar o ficheiro `exemplo.tex` por forma que somente `rvr@ncc.up.pt` o possa ler então basta

```
foo@khurt:~$ gpg --recipient rvr@ncc.up.pt --encrypt exemplo.tex
gpg: F75B22BE: There is no indication that this key really belongs
to the owner
4096R/F75B22BE 2003-06-16 "Rogerio Reis <rvr@ncc.up.pt>"
 Primary key fingerprint:
               BEB4 3CCC 0725 E024 F531  D1DD 4314 8E6D 18EA 3457
      Subkey fingerprint:
               084F 9F53 8BAD 6737 2C39  E972 6FD0 7C52 F75B 22BE

It is NOT certain that the key belongs to the person named
in the user ID.  If you *really* know what you are doing,
you may answer the next question with yes

Use this key anyway? yes
```

O resultado é um novo ficheiro `exemplo.tex.gpg` que somente `rvr@ncc.up.pt` conseguirá decifrar (nem `foo` o consegue).

Todas as dúvidas sobre a autenticidade da chave usada prendem-se com o que é explicado em B.6.

Para decifrar o ficheiro, `rvr` necessita somente executar

```
% gpg exemplo.tex.gpg

You need a passphrase to unlock the secret key for
user: "Rogerio Reis <rvr@debian.org>"
4096-bit RSA key, ID F75B22BE, created 2003-06-16
                                  (main key ID 18EA3457)

gpg: encrypted with 4096-bit RSA key, ID F75B22BE,
                                       created 2003-06-16
      "Rogerio Reis <rvr@debian.org>"
```

Isto decifra o `exemplo.tex.gpg` recuperando o ficheiro `exemplo.tex`.

Quando se cifra um ficheiro nada impede de o fazer indicando diversos nomes de destinatários, passando o resultado a poder ser decifrado por qualquer um deles.

Também é possivel assinar ao mesmo tempo que se cifra (usando a opção `-sign`) ou obter um resultado em ASCII (usando `-armor`).

B.6 Assinar identidades de outros

Que garantias temos da "qualidade" das chaves públicas que conhecemos de outras pessoas? Se as incluímos no nosso "molho-de-chaves" vindas de um ficheiro que vimos gerar, então podemos garantir que correspondem a quem dizem pertencer, mas se vierem de um qualquer servidor público, nada podemos garantir.

Para permitir ter um pouco mais de confiança nas chaves que usamos existe no gpg um mecanismo de assinatura de chaves e de "transitividade de confianças". Que quer isto dizer? Que se eu posso garantir que uma chave pertence realmente a alguém (por exemplo a `foo`) posso assinar a sua chave... e devolver a chave a `foo`, ou coloca-la num servidor público. Alguém que conheça a minha assinatura (e confie em mim), pode passar a confiar que aquela chave de facto pertence a `foo`.

O mecanismo de transitividade de confiança é muito frágil e depende da consciência dos utilizadores do gpg que não devem, em circunstância alguma, assinar uma chave que não têm certeza que pertença a quem declara pertencer. Para facilitar esta tarefa a cada chave pública está associada uma `string` (o valor de uma função de *hash*) a que chamamos impressão digital (*fingerprint*) e que é costume distribuir em papel (portanto presencialmente) aqueles que queremos que assinem a nossa chave. Mais tarde quando estes forem assinar a nossa chave devem verificar que a impressão digital que receberam impressa corresponde à da chave que estão prestes a assinar. Grandes comunidades que nunca tiveram encontros presenciais, conseguem identificar-se à distância, e assim confiar uns nos outros usando este mecanismo[3]

Para se obter a impressão digital de uma chave basta

```
foo@khurt:~$ gpg --fingerprint foo
pub  1024D/E8B20E42 2003-09-25 Foo Bar <foo@khurt.ncc.up.pt>
     Key fingerprint =
                A525 2CCF 8DF1 B791 1EA7  B6CE F338 0644 E8B2 0E42
sub  1024g/7D1349ED 2003-09-25
```

Para se assinar uma chave, o gpg com a opção `-edit-key` entra num modo interactivo cuja interacção se ilustra aqui:

```
foo@khurt:~$ gpg --edit-key rvr
gpg (GnuPG) 1.2.3; Copyright (C) 2003 Free Software Foundation, Inc.
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it
under certain conditions. See the file COPYING for details.


pub  1024D/18EA3457  created: 2001-07-10 expires: never  trust: -/-
sub  1024g/B0191BB5  created: 2001-07-10 expires: never
sub  4096g/72A4A44D  created: 2003-06-16 expires: never
sub  4096R/F75B22BE  created: 2003-06-16 expires: never
(1). Rogerio Reis <rvr@debian.org>
(2)  Rogerio Reis <rvr@ncc.up.pt>
(3)  Rogerio Reis (gpg) <rvr@ncc.up.pt>

Command> sign 1
Really sign all user IDs? yes

pub  1024D/18EA3457  created: 2001-07-10 expires: never   trust: -/-
 Primary key fingerprint:
                BEB4 3CCC 0725 E024 F531  D1DD 4314 8E6D 18EA 3457

     Rogerio Reis <rvr@debian.org>
     Rogerio Reis <rogerio.reis@netcabo.pt>
     Rogerio Reis <rvr@ncc.up.pt>
     Rogerio Reis (gpg) <rvr@ncc.up.pt>
```

[3] O exemplo clássico é a comunidade que mantém a distribuição *debian* (http://www.debian.org) que vota "legislação" interna, aprova uma "constituição", aprova contas e administra sistemas, elege representantes e dirigentes, sempre sem necessidade de reuniões presenciais.

```
How carefully have you verified the key you are about to sign
actually belongs to the person named above?  If you don't know what
to answer, enter "0".

   (0) I will not answer. (default)
   (1) I have not checked at all.
   (2) I have done casual checking.
   (3) I have done very careful checking.

Your selection? (enter '?' for more information):?
When you sign a user ID on a key, you should first verify that the
key belongs to the person named in the user ID.  It is useful for
others to know how carefully you verified this.

"0" means you make no particular claim as to how carefully you
verified the key.

"1" means you believe the key is owned by the person who claims to
    own it but you could not, or did not verify the key at all.  This
    is useful for a "persona" verification, where you sign the key of
    a pseudonymous user.

"2" means you did casual verification of the key.  For example, this
    could mean that you verified the key fingerprint and checked the
    user ID on the key against a photo ID.

"3" means you did extensive verification of the key.  For example,
    this could mean that you verified the key fingerprint with the
    owner of the key in person, and that you checked, by means of a
    hard to forge document with a photo ID (such as a passport) that
    the name of the key owner matches the name in the user ID on the
    key, and finally that you verified (by exchange of email) that the
    email address on the key belongs to the key owner.

Note that the examples given above for levels 2 and 3 are *only*
examples. In the end, it is up to you to decide just what "casual"
and "extensive" mean to you when you sign other keys.

If you don't know what the right answer is, answer "0".
Your selection? (enter '?' for more information): 2
Are you really sure that you want to sign this key
with your key: "Foo Bar <foo@khurt.ncc.up.pt>" (E8B20E42)

I have checked this key casually.

Really sign? yes

You need a passphrase to unlock the secret key for
user: "Foo Bar <foo@khurt.ncc.up.pt>"
1024-bit DSA key, ID E8B20E42, created 2003-09-25


Command> save
```

Recomenda-se vivamente ler muito bem as diversas alternativas de cada resposta, em vez de responder sim a tudo como habitualmente fazemos a todas as caixas de diálogo dos programas da Microsoft!!!

B.7 Acautelar o futuro: os certificados de revogação

Porque podemos querer invalidar uma chave, quer porque esta tenha sido comprometida, quer porque perdemos a parte privada, ou porque nos esquecemos da senha, ou simplesmente porque o contexto da sua utilização deixou de fazer sentido, existe um mecanismo no gpg que permite revogar tal chave. Mas porque algumas das razões que nos podem levar a tar decisão pressupõem que não temos mais acesso à chave privada, o que devemos fazer é gerar antecipadamente o certificado de revogação de cada chave que possuímos e quardar o resultado em lugar seguro. Os certificados de revogação são suficientemente curtos para se poderem copiar à mão, e portanto uma das formas de os preservar pode mesmo ser imprimindo-os.

Para gerar o certificado de revogação de uma chave, neste caso da nossa chave principal, basta fazer:

```
foo@kh:~$ gpg --gen-revoke foo

sec  1024D/17471D33 2007-10-17 Foo Bar <foo@khurt.ncc.up.pt>

Create a revocation certificate for this key? (y/N) y
Please select the reason for the revocation:
  0 = No reason specified
  1 = Key has been compromised
  2 = Key is superseded
  3 = Key is no longer used
  Q = Cancel
(Probably you want to select 1 here)
Your decision? 0
```

Neste caso, como estamos a criar um certificado de revogação, para o caso de perdermos o acesso à chave privada, o melhor é optar por não indicar nenhuma razão particular.

```
Enter an optional description; end it with an empty line:
>
```

Pela mesma razão colocamos nenhum comentário.

```
Reason for revocation: No reason specified
(No description given)
Is this okay? (y/N) y
```

Mais uma hipótese para nos podermos arrepender...

```
You need a passphrase to unlock the secret key for
user: "Foo Bar <foo@khurt.ncc.up.pt>"
1024-bit DSA key, ID 17471D33, created 2007-10-17

ASCII armored output forced.
Revocation certificate created.

Please move it to a medium which you can hide away; if Mallory gets
access to this certificate he can use it to make your key unusable.
It is smart to print this certificate and store it away, just in case
```

```
your media become unreadable.  But have some caution:  The print system of
your machine might store the data and make it available to others!
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.4.6 (GNU/Linux)
Comment: A revocation certificate should follow

iEkEIBECAAkFAkcWBeECHQAACgkQ+j1u5RdHHTN6tgCdF5m7QJPPx1ZfQzPoU+2O
B8Xc7rEAn2Uf7Ci5U4n1LN5Mu8Cf3wQb19SX
=+TIp
-----END PGP PUBLIC KEY BLOCK-----
```

Já temos produzida o certificado de revogação da chave. As recomendações feitas pelo programa fazem todo o sentido. Se por acaso este certificado cair nas mão erradas antes de nós querermos de facto revogar a chave, pode tornar a nossa identidade inválida, e fazer perder a confiança que os outros já depositam na nossa chave.

B.8 Programas que simplificam tudo isto

Existem programas que nos simplificam a vida tanto na administração do nosso "molho-de-chaves" como na cifra/decifração de mensagens (expecialmente se estas forem enviadas por *email*. Enquanto os segundos são quase imprescidíveis, para quem usa regularmente o gpg nas suas mensagens, para assinar e tornar confidencial o correio, as primeiras, ainda que práticas para as acções mais correntes não dispensam da utilização do gpg directamente para tarefas mais delicadas...

Somente alguns exemplos:

B.8.1 Leitores de *email* com suporte para gpg

sylpheed Um leitor de *email* com interface *GTK*, bastante completo e relativamente compacto. Assenta em primitivas equilaventes ao `mh` mas não necessita da sua prévia instalação. Pode automáticamente consultar servidores públicos de chaves para obter chaves de interlocutores desconhecidos. Existe um ramo de desenvolvimento mais sofisticado chamado `sylpheed-claws`. Plugins anti-*spam* e anti-vírus.

- http://sylpheed.good-day.net/
- http://sylpheed-claws.sourceforge.net/

exmh Leitor de *email* escrito completamente em *Tcl/Tk* assenta sobre as primitivas do `mh` (ou `nmh`).

- http://www.beedub.com/exmh/

mew Leitor de *email* para o *Gnu-Emacs*, assenta sobre o `mh`.

- http://www.mew.org/

enigmail *Plugin* para o *mozilla*.

- http://enigmail.mozdev.org/

evolution Um pouco a transposição do ignominioso *Outlook* para o *Gnome*... Não usa `mh` nem formatos compatíveis... Para quem goste!

- http://ximian.com/products/evolution/

kmail O leitor de mail do *KDE*.

- http://kmail.kde.org/

pine Um leitor de News e gestor de correio baseado que é executado em modo texto. Pode parecer primitivo, mas é muito mais poderoso e configurável do que parece.

- http://www.washington.edu/pine/

B.8.2 Gestores de chaves para o gpg

gpa *Gnu Prricvacy Assistent*. Interface gráfico sobre o *GTK*.

- http://www.gnupg.org/(en)/related_software/gpa/index.html

seahorse Aplicação *Gnome*

- http://seahorse.sourceforge.net/

GnomePGP O aplicação gráfica de suporte do gpg para o `Gnome`

- http://freshmeat.net/projects/gnomepgp

Kgpg Um interface ao gpg para o `KDE`

- http://devel-home.kde.org/~kgpg/

B.8.3 Onde encontrar mais informação?

- A página `man` do gpg:
- gpg mini howto
  http://webber.dewinter.com/gnupg_howto/english/GPGMiniHowto.html
- GnuPG Keysigning Party HOWTO
  http://www.cryptnet.net/fdp/crypto/gpg-party.html
- The GNU Privacy Handbook
  http://www.gnupg.org/gph/en/manual.html
- GnuPG FAQ
  http://www.gnupg.org/(en)/documentation/faqs.html
- *PGP and GPG: Email for the Practical Paranoid*, M Lucas, No Starch Press.