
# Email Security That's Good Enough

by Æleen Frisch

Most systems administrators are well aware of the insecurity of electronic mail. In fact, one of our tasks is to educate and remind users not to send credit card numbers or other valuable information via email. Sometimes, however, this limitation is inconvenient. For example, recently a new user from Europe needed a quick way to send bank account numbers and related information to a friend back home, and telephoning and faxing were not practical.

For the variety of similar situations where email is the best method for sending sensitive information, there is a solution: encrypted email messages. The Pretty Good Privacy (PGP) system, written by Phil Zimmermann, provides an easy-to-use facility for protecting email messages (and other files) from access by anyone other than their intended recipient.

PGP serves two distinct functions when used to protect email messages or files:

- It ensures privacy: The information in the message or file will not be accessible by anyone other than the person to whom it is being sent.
- It provides authentication: PGP's digital signatures provide a reliable way for a recipient to verify that a message was *really* sent by the person it says it's from.

You can choose to use one or both of them on any given occasion.

## How PGP Works

In general, encryption methods require a cryptographic *key* to encrypt a file. A key is a data string something like a password that can lock (encode) and/or unlock (decode) a file. It is used by the mathematical algorithms that compose the encryption program.

Simple encoding schemes use only a single key. This means the same

*Æleen Frisch is Systems Wrangler for a very heterogeneous network of computers and workstations. She is also the author of the book* Essential System Administration *(O'Reilly & Associates Inc., now in its second edition). In her spare time, she enjoys painting and lounging around with her cats, Daphne and Sarah.* Email: aefrisch@lorentzian.com.

key is used by the encoding algorithms to transform the original data into its encrypted form and to translate the encrypted message back into its original form. Such a system is called a *single-key encryption system*, a *private key encryption system*, or, in PGP parlance, *conventional cryptography*.

In contrast, *public key encryption systems* use two mathematically-related keys. One key, called the *public key*, is used to encrypt the file but cannot be used to decrypt it. Rather, the message can only be decrypted with the other key, which is known as the *private* or *secret key*.

To use such a system, someone who wanted to send you a private message would encrypt it with your public key, which you would have previously sent him. On receiving it, you would decrypt it with your private key, which you keep secret from everyone else.

The advantage of a two-key system is that public keys can be published and otherwise disseminated without any compromise in security because these keys can be used only to encode messages and not to decode them. The one aspect of the message that is slightly counterintuitive is that you must use the recipient's public key to encode a message to her and not your own public key.

Public key encryption systems are significantly more computationally intensive than single-key encryption schemes. For this reason, PGP uses a third, randomly generated key in its actual operation known as the *session key*. The outgoing message is encrypted with the session key, via a single-key encryption method, then the session key itself is encrypted with the public key and bundled together with the message. At the other end, the private key is used to decrypt the session key, and then the session key decrypts the message.

Because the session key is much shorter than the message itself, but still long enough to make any attempt to break it unfeasible, this approach substantially reduces the computational requirement without significantly affecting the system security.

## Setting Up and Using PGP

Here are the steps needed to set up and use PGP:

- Build and install the PGP software, usually into `/usr/local/bin`.
- Users who want to use PGP must create their public/private key pairs.
- Send public keys to those people from whom you wish to receive encrypted mail.
- Obtain the public keys from those people to whom you wish to send encrypted mail.
- Create a mail message, encrypt it with PGP, and optionally add your digital signature as authentication.

Building and installing PGP is straightforward. The `pgp` command is used to access all of PGP's functions. Its `-h` and `-k` options may be used to obtain brief help on the general and key management-specific functions, respectively. The PGP man page and manual are also excellent sources of information.

Once PGP is installed, the first step for anyone wanting to use it is to create a public/private key pair. This is accomplished via the `pgp -kg` command, which will prompt you for the information that it requires. The high points of the process are illustrated in the sample session shown in Listing 1.

The pass phrase acts like a password, and you will need to enter it when performing most PGP functions. The security of your encrypted messages relies on your choosing a phrase that cannot be broken. Choose something that is several words long.

After you have entered the pass phrase, the key generation process asks you to enter some random keystrokes which are used to generate random bits needed to create the keys. Just type away until it tells you to stop.

### Listing 1. Creating a Public/Private Key Pair

```
$ mkdir ~/.pgp; pgp -kg
Pretty Good Privacy(tm) 2.6.2 - Public-key encryption for the
masses.
(c) 1990-1994 Philip Zimmermann, Phil's Pretty Good Software.
...

Pick your RSA key size:
    1) 512 bits - Low commercial grade, fast but less secure.
    2) 768 bits - High commercial grade, medium speed, good securi-
ty.
    3) 1024 bits - "Military" grade, slow, highest security.
Choose 1, 2, or 3, or enter desired number of bits: 3
Always select the 1024-bit key size.
...
Enter a user ID for your public key: Rachel Chavez <chavez@fun-
fun.com>
The format is: Full Name <email address>
...
You need a pass phrase to protect your RSA secret key. Your pass
phrase can be any sentence or phrase and may have many words,
spaces, punctu-ation, or any other printable characters.

Enter pass phrase ...will not echo...
Enter same pass phrase again: ...will not echo...
```

Once your keys have been created, several files will be created in your ~/.pgp subdirectory, including pubring.pgp, your public key ring, where your public key and other public keys you accumulate are stored, and secring.pgp, which holds your private key. The files in this subdirectory should be protected against access from anyone but their owner (i.e., mode 600 = -rw-------).

## Adding Keys to your Public Key Ring

When someone sends you their public key, add it to your public key ring with the pgp -ka command, specifying the file containing the public key (often a saved email message) as the command's parameter:

```
$ pgp -ka new_key
Pretty Good Privacy(tm) 2.6.2
...
Looking for new keys...
pub     512/2F7A2A49 1996/12/10
        Sonya Jones <jones@erewhon.com>

Checking signatures...

One or more of the new keys are not fully certi-
fied. Do you want to certify any of these keys
yourself (y/N)? y

Key for user ID: Sonya Jones
<jones@erewhon.com>
1024-bit key, Key ID 2F7A2A49, created
1996/12/10
Key fingerprint = F4 BA 18 40 23 9D 65 0A 05 9E 77
77
AA AF 88 DF
This key/user ID association is not certified.

Do you want to certify this key yourself (y/N)? y

READ CAREFULLY: Based on your own direct first-
hand knowledge, are you absolutely certain that
you are prepared to solemnly certify that the
above public key actually belongs to the user
specified by the above user ID (y/N)? y
```

The PGP system finds one public key within the specified file that belongs to Sonya Jones. PGP requires that public keys be *certified* before they are added to your public key ring. This can happen in one of two ways.

- You can certify the key yourself by answering yes to the prompts, as illustrated in the preceding example. You should do this only if you are sure that the public key has in fact come from the person indicated. The key fingerprint printed out by PGP (a series of hexadecimal digit pairs) can be used to verify the authenticity of a key by telephone or other method.

Following these prompts, the PGP package asks you to specify the extent to which you trust this person to introduce other public keys to you in the future. This leads to the second certification method.

- The key can include the signature of someone that you already trust who thereby certifies it for you. When a key has such a signature, the certification prompts are not given by the pgp command.

You can view the keys on your public key ring and their signatures with the pgp -kvv command; the key part of its output is shown in Listing 2.

You can sign a key explicitly with the pgp -ks command. For example, Rachel Chavez has signed her own key using this command:

```
$ pgp -ks chavez -u chavez
```

Note that it is common practice to sign your own key. Rachel could sign Tim's key using this command:

```
$ pgp -ks txw -u chavez
```

Now, if Rachel sends Tim's public key to someone else who already trusts Rachel, then Tim's key will be automatically certified when the third person adds it to his public key ring. Of course, Rachel would have to send

### Listing 2. Viewing your Public Key Ring

```
$ pgp -kvv
...
Type bits/key  ID       Date           User ID
pub 1024/BE8B1759     1995/06/20      Rachel Chavez <chavez@funfun.com>
sig    14187BD5                         Rachel Chavez <chavez@funfun.com>
                                       Rachel Chavez <rachav@workwork.com>

pub 1024/2F7A2A49     1996/12/10      Sonya Jones <jones@erewhon.com>
sig    14187BD                          Rachel Chavez <chavez@funfun.com>
pub 1024/XD083766     1996/04/11      Tim Wrench <txw@somewhere.org>
```

## Useful PGP Commands

*Note that option order is often significant for the* `pgp` *command.*

| | |
|---|---|
| `pgp -h` | General command help. |
| `pgp -k` | Help for key management functions. |
| `pgp -kg` | Generate public/private key pair. |
| `pgp -ks` *user* `-u` *user* | Certify *user*'s public key (can be your own). |
| `pgp -ka` *file* | Add public keys in *file* to your public key ring. |
| `pgp -kv` | View keys on your public key ring. Use `-kvv` to see the signatures as well. |
| `pgp -kxa` *user file* | Extract public key for *user* to *file*. |
| `pgp -seat` *file user(s)* | Encrypt and sign *file* for transmission to specified *users*. Add `-w` to delete the cleartext file after encryption. |
| `pgp -seatf` *user* `\|` `mail` *user* | Encrypt, sign and send a message entered on-the-fly. |
| `pgp` *file* | Decrypt *file* using your private key (prompts for new file name). Add `-m` to view the decoded message via `more`. |

her signed version of Tim's key back to Tim himself if Tim wanted to distribute a version of his public key that includes Rachel's signature.

### Sending Public Keys to Other Users

To extract a public key to a file, use a command of the form:

`pgp -kxa` *username filename*

where *username* is the username for the key that you want to extract. You can add the `-f` option to send the key to standard output instead of to a file. Thus, Rachel could use this command to send her key public key to user `phil@supercomp.edu`:

```
$ pgp -kxaf chavez | mail phil@supercomp.edu
```

As always, the `pgp` command prompts you for your pass phrase and any other required input.

If we look at Listing 2, we notice that Rachel Chavez's public key has a second username associated with it: `rachav@workwork.com`. PGP is designed so that each person has one pair of keys (not one pair per user ID).

You can add a username to your public key with the `pgp -ke` command (add your username as the command's parameter to avoid the first prompt). You can also use this command to change your pass phrase.

### Using PGP to Send, Read Secure Email

Once all of the setup work is done and keys are all in place, it is very simple to send and receive secure email. If the message you want to send is contained in the file `my_mess`, then the following command will send the message in encrypted form to user `jones`:

```
$ pgp -ea my_mess jones
```

Adding the `-w` option will cause PGP to delete the original cleartext message file once the encrypted file is created.

You can also include your *digital signature* within the message. A digital signature is PGP's method for authenticating the sender (creator) of a message, and the recipient will need your public key in order to verify your digital signature. Thus, both sender and recipient need to have the other person's public key for secure, authenticated electronic communication.

The following one-line command will send the message in the file `my_mess` to user `jones`, encrypting and authenticating it, and erasing the original file afterwards:

```
$ pgp -seatw my_mess jones; mail
jones@erewhon.com < jones.asc
```

Of course, you could use any mailer to send the resulting file to user `jones`. If you want to create an encrypted mail message on-the-fly, you can use a command like this one:

```
$ pgp -seatf jones | mail jones@erewhon.com
 . . .
Enter the message here, terminated with …
^D
```

Once the message is terminated with Control-D, the `pgp` command will go on to ask for your pass phrase as usual.

To read an encoded message, save it to a file and then execute the command `pgp` *file*. You will be prompted for a file name in which to save the decrypted message. Any digital signature included within the file is also authenticated as part of the same process.

If you would prefer to view the decoded message, use

the `-m` option to display it one page at a time (you will be able to optionally save it to a file at the end).

My users do not routinely use PGP for email messages, so we have not bothered to integrate the package into any of the mailer programs that are used at our site. However, it has been incorporated into most popular mailer programs. Information about integrating PGP into various email programs can be found at `http://world.std.com/~franl/pgp/utilities.htm`.

## More Information

The official PGP distribution point is at Massachusetts Institute of Technology in Cambridge, MA. U.S. and Canadian users can obtain PGP via the Web at `http://web.mit.edu/network/pgp-form.html`. Users without WWW access can obtain it by anonymous FTP to the site `net-dist.mit.edu`; retrieve the file `pub/PGP/ README` and follow the directions.

Both methods require you to attest to your U.S. or Canadian citizenship and to agree to the license agreements and other restrictions pertaining to PGP. Because U.S. law prohibits transfer of the software to noncitizens or outside of the United States and Canada, this site will not transfer the software to users coming from elsewhere in the world.

PGP is also widely available on the Internet at major WWW and FTP sites, both in North America and elsewhere. PGP versions outside the United States and Canada are variations modified to use alternate encryption algorithms to those prohibited for export outside the United States. One easily accessible European site on the WWW is `http://www.funet.fi/pub/crypt/cryptography/pgp/unix/README.html`, or by anonymous FTP to `nic.funet.fi` from the directory `/pub/crypt/cryptography/pgp/unix`.

PGP versions outside the United States and Canada are variations modified to use alternate encryption algorithms to those prohibited for export outside the United States.

Finally, the following books may be of interest to PGP users and systems administrators responsible for installing and maintaining the package:

- *The Official PGP User's Guide* by Phil Zimmermann (MIT Press, 1995, ISBN 0-262-74017-6).
- *PGP: Pretty Good Privacy*, by Simson L. Garfinkel (O'Reilly & Associates, 1995, ISBN 1-56592-098-8). ▲