Getting Started With PGP

by *Kevin Henry*

Phil Zimmermann created the original PGP (Pretty Good Privacy) security system in 1991 as a way of protecting digital information. PGP uses both public key cryptography and secret key cryptography to protect information against eavesdropping and forgery. Every user has a pair of keys, one public and one private. PGP uses the public key to encrypt messages and verify signatures. The private key facilitates message decryption and signature creation.

Cryptosystem keys are very large integers with particular mathematical properties. A discussion of these properties is beyond the scope of this tutorial. I suggest reading the RSA Cryptography FAQ for a rigorous introduction to all modern methods of cryptography; especially the sections devoted to public key technology and PGP, which are especially relevant to this tutorial [1].

The following is a general introduction to public key cryptosystems. This tutorial walks through the basics of PGP by demonstrating a few simple steps, which will greatly enhance your electronic security. Having read this article, you will be able to sign, verify, encrypt, and decrypt files using your own key pair.

Graphical PGP plug in modules are available for integrating message security into the very popular Windows mail applications including Microsoft Outlook, Qualcomm Eudora, and Claris Emailer. For users of Unix systems, command line PGP is supported by Emacs, `elm`, and `exmh`. These support modules greatly simplify sending and receiving encrypting mail by invoking PGP in the background of the mail program, encrypting messages before sending and decrypting them when you want to read. However, you must still supply the pass phrase for your secret key.

Overview Of PGP

Public key cryptosystems operate on the assumption that any user's public key is readily available, but an individual user has the only copy of his or her private key. Like all other systems of this type, PGP provides **privacy** and **authentication**.

Privacy

PGP encryption ensures the privacy of a message, so no one but the intended recipient can decipher it. Eavesdroppers can capture coded messages in transit over insecure networks, but will find them incomprehensible.

You can ensure the privacy of your messages by encrypting them for their proper recipients. PGP uses a mathematically based encryption algorithm of exponents and modular arithmetic [2] to combine your message with your intended recipient's public key. Only your recipient, using his or her private key, can decipher the code and read your message.

Authentication

PGP also allows a recipient to verify that messages are actually from the senders they claim, and have not been altered in transit, whether accidentally or maliciously. Files are first passed through a secure hash function. A secure hash function produces a fixed-length message from an arbitrarily long input message. It has two special properties. First, it is difficult to find two different inputs that result in the same hash. Second, given a hash it is difficult to determine the original input message. PGP signs a hash rather than the entire file for both security and efficiency reasons. Although forging email on the Internet is relatively simple to accomplish, using authentication frustrates individuals who do so.

You can use PGP to digitally sign your messages, proving to your recipients that your messages are really from you. PGP creates a signature of your message using your private key, which other people can check against your public key. No one can forge the signature without your private key.

PGP Key Rings

Your **key ring**s are files where PGP stores your keys; so you alone should have access to your key ring files. I've removed quite a few private keys insecurely stored on computers in public labs during busy seasons of student projects. Remember that anyone who obtains your private key can both read messages encrypted to you and forge your signature. Protect your private key with great care!

To automatically avoid those situations entirely, use PGP on a computer you own personally, or a highly secure multi-user system. The examples in this tutorial are from my Linux system. In any case, you must either trust your system administrator, or keep your private key on a floppy disk.

Now that we have covered the background of computer message security, let's get started using PGP.

Configuring Your Computer

Notice that command line options are not always clear. Next to each possibly confusing sample input, find italicized descriptions of the options employed. Think of them as comments in code, but don't type them on the command line!

Obtaining PGP

Because of legal restrictions limiting the use and distribution of strong cryptographic technology, as well as various patent issues; domestic and international versions of PGP have evolved along similar but separate lines.

If you are a U.S. citizen in the United States, or a Canadian citizen in Canada, you may download PGP Freeware from MIT [3]. If you not a citizen of the U.S. or Canada, or you intend to use PGP worldwide, get your software from the International PGP Home Page [4]. The version numbering scheme between domestic and international versions is not synchronized and slightly confusing, see below for a note about compatibility.

No matter where you live, you may only use the free versions of PGP for personal, academic, or volunteer work; that is to say you may not make any money from your use. For-profit businesses must buy commercial licenses for PGP from Network Associates [5]. For example, a company that exchanges proprietary information about its business plans using freeware versions of PGP technology would probably be in violation.

To ensure compatibility, make sure you have the latest version of PGP. Although there is no problem with compatibility between platforms, there is a problem with using DSS/DH keys rather than RSA keys between the "domestic" and "international" versions [6]. Meanwhile, the patented RSA algorithm (United States Patent: 4,405,829) becomes part of the public domain on September 21st, which should greatly simplify many of the legal issues surrounding cryptography.

I went to the MIT site, agreed to the terms, and downloaded the United States, non-commercial version 6.5.2; executable binary distribution for RedHat Linux 5.0 and higher. No extra software is needed to use this command line freeware. If you're using a Microsoft Windows computer, download the appropriate file. You will probably need WinZip [7] to extract the program.

Installing PGP

The RPM package manager inherent to RedHat Linux [8] makes installation trivial. As the superuser, uncompress the archive with:

```
# tar xfvz PGP-version-whatever.rpm.tar.gz
```

The "xfvz" meands *extract, from file, verbose, unzip first*. Now install the package with:

```
# rpm -i PGP-version-whatever.rpm
```

where the -i flag is for *install*

If you don't trust a binary from the MIT site, you may choose to download a distribution of the PGP source code. Examine the source code, then simply follow the README file instructions to compile and install it yourself. Unless you have to buy a compiler, this is the most reliable way to go, and it's good to examine the source code of any program you install. For simplicity here, I used a binary.

Establishing Your Identity

You will need to create a private and public key pair to use with PGP. It's appropriate to publish your public key somewhere public, for example on a web page, where other PGP users can get it.

Creating Your Keys

On a UNIX or compatible system, run PGP with the options to generate a key pair:

```
$ pgp -kg        keys, generate
Pretty Good Privacy(tm) Version 6.5.2
(c) 1999 Network Associates Inc.
Uses the RSAREF(tm) Toolkit, which is copyright RSA Data Security, Inc.
Export of this software may be restricted by the U.S. government.
Choose the public-key algorithm to use with your new key
1) DSS/DH (a.k.a. DSA/ElGamal) (default)
2) RSAChoose 1 or 2:
```

You are asked to choose an algorithm for your keys to support. The default is Digital Signature Standard, Diffie-Hellman; also known as Digital Signature Algorithm, El Gamal. You may override the default by choosing Rivest, Shamir, Adleman; but this may have patent law implications. Go with the default.

```
Choose the type of key you want to generate
1) Generate a new signing key (default)
2) Generate an encryption key for an existing signing keyChoose 1 or 2:
```

You don't have any signing keys yet. Choose the default again.

```
Pick your DSS "master key" size:
1) 1024 bits- Maximum size (Recommended)
Choose 1 or enter desired number of bits:
```

Again, choose the default; unless for some reason you want a weak key.

```
Generating a 1024-bit DSS key.
You need a user ID for your public key. The desired format for this
user ID is your name, followed by your E-mail address enclosed in
<angle brackets>, if you have an E-mail address.
For example:  John Q. Smith <jqsmith@nai.com>
Enter a user ID for your public key: Kevin Henry <khenry@acm.org>
```

I entered my name and one of my email addresses. Enter your name and email address.

```
Enter the validity period of your signing key in days from 0 - 10950
0 is forever (the default is 0):
```

Press Enter here to accept the default, if you want your key to never expire.

There are situations where keys should expire after a certain period of time. For example, if PGP were being embedded in a system to secure communications between two bank computers for a single session, creating a key pair for that session would be very secure. Even if the key pair was cracked, it was only used for one session then discarded, minimizing the potential damage done.

```
You need a pass phrase to protect your DSS secret key.
Your pass phrase can be any sentence or phrase and may have many
words, spaces, punctuation, or any other printable characters.

Enter pass phrase:
Enter same pass phrase again:
```

Your signing key is stored in code on your private key ring. Remember the pass phrase you enter here, because you will need it to unlock the private key for use later. It's important to choose a pass phrase the same way you would choose an important account password. Choose something you can remember without writing down, but hard for other people to guess.

```
PGP will generate a signing key. Do you also require an
encryption key? (Y/n)
```

Yes, you do. Press Enter to accept Y, the default.

```
Pick your DH key size:
1) 1024 bits- High commercial grade, secure for many years
2) 2048 bits- "Military" grade, secure for foreseeable future
3) 3072 bits- Archival grade, slow, highest security
Choose 1, 2, 3, or enter desired number of bits: 4096
```

Obviously, archival grade isn't the highest security allowed. Feel free to make your own judgment call here, but remember that computers get faster all the time.

```
Enter the validity period of your encryption key in days from 0 - 10950
0 is forever (the default is 0):
```

Unless you have a reason not to, accept the default option again here.

```
Note that key generation is a lengthy process.
PGP needs to generate some random data. This is done by measuring
the time intervals between your keystrokes. Please enter some
random text on your keyboard until the indicator reaches 100%.
Press ^D to cancel
44% of required data
```

Computers cannot generate random numbers. The best they can do is measure natural phenomena and apply the resulting data to complicated formulae, producing close to random numbers. So we play on the keys for a while until PGP has gathered enough data.

```
100% of required dataEnough, thank you.
******* ..............................................................
..........................................******* .
Make this the default signing key? (Y/n)
```

Sure! Just hit Enter (again).

```
Key generation complete.
```

Congratulations! You've created a public and private key for yourself.

Publishing Your Public Key

For other people to encrypt mail for you or check your signatures, they must have access to your public key. PGP includes functionality to export your public key from your public key ring to a text file. This process is officially called extracting a key.

First, let's see what public keys are available for use to extract.

```
$ pgp -kv     keys, view list
...
Type bits      keyID      Date       User ID
DSS  4096/1024 0xEA5F4D84 2000/05/08 *** DEFAULT SIGNING KEY ***
                                     Kevin Henry <khenry@acm.org>
1 matching key found.
```

Now let's extract (copy) that key to a text file.

```
$ pgp -kx khenry pgp.txt        keys, extract
...
Extracting from keyring '/home/khenry/.pgp/pubring.pkr', userid "khenry".

Key extracted to file 'pgp.txt'.
```

PGP needed me to specify just enough of my user name for it to have uniqueness from the other keys on the ring. I gave it more than enough, so it extracted my public key to a text file.

You can distribute this file of your PGP public key amongst your friends, or post it to your web page where anyone can get it. My real key (not the one I created for this tutorial) is part of my web site [9].

If you think your key will be in high demand, or just for fun; you might also consider submitting your key to the Public PGP Key Server. The server is provided as a free service by MIT [10].

Message Authentication With PGP

Signing A Message

To prove to someone else that a message is authentic and has not been tampered with, use PGP to create a digital signature on the message. First create a text file of the message, then use PGP to sign it.

```
$ cat message.txt
This is a test message. I am going to sign it.
$ pgp -sta message.txt        sign, text mode, armor
...
A secret key is required to make a signature.

You need a pass phrase to unlock your secret key.
Key for user ID Kevin Henry <khenry@acm.org>

Enter pass phrase:
Passphrase is good

Clear signature file: message.txt.asc
```

Without option "a" for ASCII, PGP will create a binary file of the signature. The results look like this:

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

This is a test message. I am going to sign it.

-----BEGIN PGP SIGNATURE-----Version: PGP 6.5.2
iQA/AwUBORbXuzZ63grqX02EEQIczgCgiTGGq/N8Dt+mm0++8B6dQJdS4doAoP5L
JRsUwjFboBitfT/P0C3IScmY
=f/qi
-----END PGP SIGNATURE-----
```

Use the options shown above to produce this text file format. Files like this are ready to pass through email systems.

Getting Another Person's Key

As a recipient, I want to make sure that messages are authentic and have not been tampered with on their way to me. I do this by verifying a digital signature. First, I need the public key of the sender in a file.

```
$ pgp -ka publickey.txt        keys, add to ring
...
Looking for new keys...
```

```
DSS  4096/1024 0xEA5F4D84 2000/05/08 Kevin Henry <khenry@acm.org>
sig?           0xEA5F4D84             (unknown signator, can't be checked)

Keyfile contains 1 new keys. Add these keys to keyring? (Y/n)
```

Putting the other person's public key on my ring is a necessary step before I can check a signature from him or her.

```
Keyfile contains:
   1 new key(s)

Summary of changes :

New userid: "Kevin Henry<khenry@acm.org>".
New signature from keyID 0xEA5F4D84 on userid Kevin Henry <khenry@acm.org>

Added :
   1 new key(s)
   1 new signatures(s)
   1 new user ID(s)
```

PGP tells me exactly what it did to my key rings, but it does go a little off the deep end with the plurals. Look carefully.

Now I'm ready to check the signature of the message.

Verifying A Signature

```
$ pgp message.txt.asc
...
File is signed.  Good signature from user "Kevin Henry <khenry@acm.org>".

Signature was made 2000/05/08 15:06 GMT

WARNING:  Because this public key is not certified with a trusted
signature, it is not known with high confidence that this public key
actually belongs to: "Kevin Henry <khenry@acm.org>"

Plaintext filename: message.txt
```

The message was authentic and in the same original condition as when the sender made the signature. But there's something else going on here. I never explicitly told PGP to trust this public key, but I can do that.

Certifying A Key

PGP uses a somewhat anarchic model called the "web of trust" to certify trusted keys; there is no central authority. I can sign or certify someone's public key to indicate that I trust him or her as the owner of that key. I'll specify the key as a serial number for this example:

```
$ pgp -ks 0xEA5F4D84        keys, sign
...
A secret key is required to make a signature.
```

```
You specified no user ID to select your secret key,
using default signing key if set, otherwise the default user ID
and key will be the most recently added key on your secret keyring.

Key for user ID: Kevin Henry <khenry@acm.org>
1024-bit DSS key, Key ID 0xEA5F4D84, created 2000/05/08
          Key fingerprint =  7C 0F B3 57 55 98 90 D8  AE 8E A5 C7 36 7A DE 0A  EA 5F
4D 84

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 dire warning is to discourage forging or carelessly certifying public keys. Nothing really stops someone from creating keys for another person and soliciting signatures on that key from unsuspecting or other unscrupulous users.

Ideally, you will verify the fingerprint over a telephone with the owner of the key. In this case, I made the key myself: I know it's mine.

```
You need a pass phrase to unlock your secret key.
Key for user ID "Kevin Henry <khenry@ece.villanova.edu>"

Enter pass phrase:
```

I am signing this example key with my real personal secret key.

```
Passphrase is good

Attach a regular expression to this signature, or
press enter for none:
```

I don't need to do this, so I press Enter, which completes the process. I now view the new key in verbose mode:

```
$ pgp -kvv khenry@acm.org        keys, view, verbose
...
Looking for user ID "khenry@acm.org".
Type bits      keyID      Date       User ID
DSS  1024      0xEA5F4D84 2000/05/08
 DH  4096      0xEA5F4D84 2000/05/08 Kevin Henry <khenry@acm.org>
sig            0xEA5F4D84             Kevin Henry <khenry@acm.org>
sig            0xDF5CEF90             Kevin Henry <khenry@ece.villanova.edu>
1 matching key found.
```

It would be courteous of me to extract this public key, with my newly created signature, and return it to its owner, who adds my signature to his or her copy of his or her public key. That way the network of trust is encouraged and maintained. Some organizations also offer PGP signing services [11].

PGP users often participate in PGP keysigning parties as a popular and social way of certifying public keys in the "web of trust." You can learn how to conduct your own key signing party by reading http://www.pgp.net/pgpnet/pgp-faq/faq-06.html

Message Privacy With PGP

Encrypting A Message

I can encrypt a message for anyone whose public key I have:

```
$ cat message2.txt
I will use PGP to encrypt this message. Eavesdroppers will be foiled.
$ pgp -esa message2.txt khenry@acm.org        encrypt, sign, armor
...
A secret key is required to make a signature.

Recipients' public key(s) will be used to encrypt.

You need a pass phrase to unlock your secret key.
Key for user ID "Kevin Henry <khenry@ece.villanova.edu>"

Enter pass phrase:

Passphrase is good

Key for user ID: Kevin Henry <khenry@acm.org>
1024-bit DSS key, Key ID 0xEA5F4D84, created 2000/05/08

Transport armor file: message2.txt.asc
```

I (as the sender) can now send this file to myself (as the recipient) without fear of anyone reading it while in transit over the Internet.

Decrypting A Message

I want to read a message that has been sent to me encrypted with my public PGP key.

```
$ pgp message2.txt.asc
...
File is encrypted. Secret key is required to read it.

Key for user ID: Kevin Henry <khenry@acm.org>
1024-bit DSS key, Key ID 0xEA5F4D84, created 2000/05/08
Key can sign.
You need a pass phrase to unlock your secret key.

Enter pass phrase:
signature not checked.
Signature made 2000/05/08 17:56 GMT
key does not meet validity threshold.

WARNING:  Because this public key is not certified with a trusted
signature, it is not known with high confidence that this public key
actually belongs to: "(KeyID: 0xDF5CEF90)".
```

```
Plaintext filename: message2.txt
```

This dire sounding warning from PGP just means there's no copy of my real public key available in this user's key ring. However, the decryption was successful.

```
$ cat message2.txt
I will use PGP to encrypt this message. Eavesdroppers will be foiled.
```

Congratulations!

Caveats

PGP is not without its problems. Although PGP is relatively easy to use, on an absolute scale people find both the command line and graphical user interface difficult to use. A study by Whitten shows that the average technically-aware person will make mistakes such as neglecting to encrypt a message or not understanding how to decrypt a message [12]. Moreover, the implementation of PGP has its share of security flaws. For instance, recent version of PGP for Linux generated non-random keys. This vulnerability allows an adversary to easily guess a user's private key. The lesson to learn is that no security system is designed or implemented perfectly. Therefore, you should always be on the lookout for security vulnerability announcements.

Conclusions

In this tutorial, you received a quick introduction to Pretty Good Privacy and its underlying public key technology. You watched me download and install PGP, create keys for myself and exchange those keys with my other identity, sign, verify, encrypt and decrypt electronic mail messages. I encourage you to create your own PGP keys, and enter the world of enhanced computer communications security.

References

1
RSA Laboratories, Cryptography FAQ -- http://www.rsalabs.com/faq/

2
PGPi Documentation, How PGP Works -- http://www.pgpi.org/doc/pgpintro/#p10

3
MIT, PGP Freeware Download -- http://web.mit.edu/network/pgp.html

4
PGPI: The International PGP Home Page -- http://www.pgpi.org/

5
Network Associates, Inc., PGP Home -- http://www.pgp.com/

6
PGPi Documentation, Compatibility -- http://www.pgpi.org/doc/faq/pgpi/en/#Compatibility

7
WinZip Computing, Inc., WinZip Home Page -- http://www.winzip.com/

8
Red Hat, Inc., REDHAT.COM -- http://www.redhat.com/

9
Kevin Henry, Public PGP Key -- http://www.ece.villanova.edu/~khenry/pgp.txt

10

MIT, PGP Certificate Server -- http://pgpkeys.mit.edu:11371/

11

Thawte, A VeriSign Company, and PGP -- http://www.thawte.com/certs/personal/pgp.html

12

Whitten, Alma. Why Johnny Can't Encrypt. 8th USENIX Security Symposium, http://www.usenix.org/