
PGP: a Nutshell Overview

by Jeremiah S.Junken
(jjunken@nations.ucs.indiana.edu)

Editorial Crew
Gary Kline
(kline@tao.thought.org)

Peter Simons
(simons@peti.gun.de)

<><> Forward <><>

Nutshell, text Version 2.7 (Not related to PGP Version number)

This document could easily be titled "PGP for dummies", but I like doing things differently. Nonetheless, this document should get the "dummy" from point A to the finish line without much trouble. Even if you're one of the guru's who entire computing experience borders making you an acolyte to the computer god, I think you would still be well served to read this.

This document is in no way meant to supplant the documentation provided by Mr. Zimmerman, but rather as a plain-English quick- reference for those who would rather get down to business than screw around with intricacies. My belief is that you should learn the essential basics and dive in straight away. The finer points will grow on you, and you learn them as you go.

Remember that security as is only as good as the people who know the codes, so under no circumstances tell ANYONE your password or write it down where it might be found. Be aware of shoulder-surfers (people who can't keep their eyes off the keyboard when you're entering passcodes they're not intended to know)..

You probably got this because you wanted to avoid the BS associated with huge, detailed manuals. After all, you didn't get PGP for an education, you got it to DO something. With this in mind, I've written this to be easy to understand so you can get started quickly. However, you should remember

that if you do not use PGP correctly and bypass steps, you risk not only your own security, but anyone who communicates with you and possibly more. For that reason, you should see the recommended reading order immediately below the table of contents and ultimately go through the whole guide!

It would be a VERY good idea to print up the whole thing so you can reference it while actually using the software!

For the sake of Simson Garfinkle and his little publisher, O'Reilly, who was subtly hinting lawsuit due the use of the name "PGP in a Nutshell", which Mr. Garfinkle suggested might be a trademark violation... I hereby declare that neither myself, nor any of the editing crew of the document have anything whatsoever to do with O'Reilly, or their "Nutshell" series, or Mr. Garfinkle and his efforts. (Where does WIRED get these cretins, anyway?)

(At this point, I don't think I'd want to, either... :-))

I suggest that you read this tutorial in the following order:
Getting Started, Environment, Advanced PGP, PGP Applied.

<><> Table of Contents <><>

Master PGP and make it REALLY work for you!

Advanced PGP:
Authentication
Advanced PGP: Certification
Advanced PGP: Key Editing
Advanced PGP: Copying Secret Keys

- **Advanced Security Considerations**

Unlikely surveillance possibilities

Advanced Security Considerations:

Electromagnetic Inference Interception

Advanced Security Considerations:

Hard Disk Reading

Advanced Security Considerations: Remote Video Monitoring Advanced Security Considerations:

linetap Advanced Security Considerations:

modifications.

- **Correspondence Information**

How to reach me

- **PGP Distribution Information**

Where to get PGP

- **Notes General Notes on text, etc.**

- **PGP Supplementary Stuff PGP related programs, etc.**

<><> Environment <><>

When you sit down in front of your terminal and use PGP, you are doing so to ensure your security and privacy. You can't cover all bases; obviously, you're only human, but you can minimize the chance for security leaks and thusly, the compromise of your privacy.

Environment, in this text, is a reference to your computer, and the physical area in which your computer exists.

- **Environment: PLAINTEXTS**

When you use PGP, you first write a document containing the message you want to encrypt. This document you are writing is not encrypted, and while it's not encrypted, it's vulnerable to being read.So you encrypt it and send it to your destinations. Pretty simple. Of course, this document can still be read! How? DID YOU DELETE THE ORIGINAL? You must. Or, use PGP's Conventional Cryptology option and encrypt the original with a password so you read it when you need it.

- **Environment: RESIDUAL DATA**

If you use MS-DOS 5.0 or better, you should be familiar with the UNDELETE command. If you were to write a plaintext, encrypt it, delete the original plaintext and mail it, the PLAINTEXT CAN STILL BE UNDELETED. Even if you don't have an undelete command, you should be aware that there are some out there, and if someone REALLY WANTED TO, they could recover the plaintext. This is true of ANY platform that uses any sort of disk technology, everything from Macintosh or an IBM to the largest mainframes. Not just hard disks, but floppies too!

There are several ways to avoid this. For MS-DOS, there exists a NUKE command. This command writes over the file with 1's, 0's, 1's, THEN deletes it. In that way, it's not possible to recover it. This is Department of Defense Standard.

Another way is to run a Disk-Defragmenter after the file is deleted. This will also overwrite the residual data.

- **Environment: ENVIRONMENTAL VARIABLES**

In PGP, there is an option to set your passphrase as an Environmental Variable. In MS-DOS, this could be in your AUTOEXEC.BAT. In AmigaDOS, your startup-sequence or one if it's children. In UNIX C-shell, your .login. This is a TERRIBLE mistake to set it in a batch file, because anyone could read that file and your passphrase would be plainly visible, unencrypted, and therefore available to the intruder. However, if you set it in a batch file, it's already in memory, and PGP could be used by whoever was sitting in front of the computer without any problems.

Solution: Do not set a PGPPASS Environmental variable. Ever. For ANY reason.

- **Environment: PASSWORD ECHO**

There is an option to echo the password when you type it in. By default, PGP does not show what you type so that someone looking over your shoulder could not see it. They could still watch your fingers on the keyboard.

Solution: DO NOT set Password echo on. It's not rude to ask someonenot to watch when you enter a password.

- **Environment: SHARED SYSTEMS**

Shared Systems are bad news for security, Period. I've been on the hacking side, and the side of the Hacked, and quite simply, a shared system = Privacy Risk. Remember that the operator (ROOT on a Unix System) or a clever hacker could easily see a dump of system memory and hence, your passphrase while PGP is decrypting.

The TEMP directory (/tmp in UNIX) is another problem. PGP could store it's swap files there. You would be best off to edit the PGP Config.txt (PGP.config on Amiga) and define the TEMP directory to be your home directory. Also Make sure you have the privileges set so that others cannot read your home directory

(In Unix: chmod . og-rwx ; chmod . u+rwx. If this command syntax is incorrect, see the NOTES sections immedietly below the Table of Contents.)

Type the following to make your home directory private:

chmod 700 /home/users/smith,

if Smith was your home path. An easier way to do this is to type: chmod 700 ~

~ means your home directory.

There's also the umask command, which you should look up with the "man" command.

If you use a Mailsystem such as ELM that creates a tempfile in the /tmp before mailing, it's better to write your message with Emacs before you start in mail, emacs newmessage.txt then encrypt it (pgp -ea newmessage.txt) then mail it with pipe redirect:

mail user < newmessage.txt.asc

and that way, any temp files created would be encrypted, and hence, useless to the peeping intruder!

If you would like to see some more security concerns, see the Advanced Security Considerations section at the tail-end of this file!

- **Environment: YOUR PASSWORD**

When you select your password, you should not use anything easy to guess, like the name of a spouse, a nickname, a favorite sports team, or something even worse like your last name backwards. The technique I use for generating passwords which are easy to remember but next to impossible to guess is to think back to your elementary school years, think of the best time you had during that period in your life, or something you had then that you really loved, and use that as a passphrase.

For example's purposes, we'll use my favorite toy of that time, which was ROBOTIX robot toy construction set. (That, or my Commodore 64 :-))

The passphrase 'robotix' might be in a dictionary or something that someone might try. So, you might add a few random characters: 'ro_b&ot|><', and maybe the year of your birth. '19ro_b&ot|><75'. That would be damned near impossible to guess, whereis 'robotix' is unlikely to be guessed, it's still possible.

Of course, it's a passPHRASE not a passWORD, so it could be: "When in the course of human events..."you could change that to: "\\/h3|\| |n th3 k0urS3 uhv H\//.A|\| 3\/3|\|T5..," or something.

Another HUGE mistake a lot of people make is mumbling their passwords, especially in efforts to remember them. THAT is a critical mistake. So, if you think what you're typing, make sure your mouth isn't doing the thinking!

The point of this thing is that unless your password is only in your head and encrypted on your secret ring, it'ss vunerable. Put yourself in the position of someone who wants to learn your password, and think of every possible, even ridiculous thing you would try.

This section is a jump start from ignorance into competent usage of PGP.
Read carefully and follow instructions step-by-step!

<><> Getting Started <><>

- **Getting Started: GENERATING YOUR KEYSET**

Okay. In order for someone to send you mail, they'll need your Public Key. You have to create that yourself. When you create it, it creates your Secret Key (which is password protected) and a Public Key. The Public Key is used by others to encrypt data to you. Once encrypted with your Public Key, YOUR _SECRET_ Key, and ONLY YOUR SECRET key can decode the information.

So Let's do it!

type: pgp -kg

It will prompt you for several things. One is your ID line, or what people will see that identifies the key as yours from the human perspective.

Your name <your email address>

That's the general convention, but some people like to use a witty comment instead of an Email Address. It's entirely up to you.

For example, Peter Simons is Peter Simons <simons@peti.GUN.de

Another option will be Key Size. Pick the largest option (1024-bitkey). It might take a while (As long as 5 minutes) to generate the key on most modern machines, but this is YOUR SECURITY we're talking about, not waiting on a laundry dryer. (On older machines, it could be as long as an hour, but it will never take that long to decrypt a message. Usually no more than 5 minutes. I use a very old computer, and it doesn't take more than 40 seconds to encrypt or decrypt for a 1024bit key.. but then again, I use an AMIGA!)

It will ask for a passphrase. A "passphrase" is password, but it's longer. It can be a whole sentence, or just a few letters. Remember to make it something you can remember easily, but not something easily guessed. When I've helped friends generate passwords, I usually tell them to try and remember a really fun time they had with a friend, and pick a word that describes the situation, then the friend's name, and use either.

For a good password, you might want to look at the section in the very beginning on passwords!

The most secure passwords are random strings of both letters and numbers like: az193095=-evce2 or something. Whatever you choose make sure YOU can ALWAYS remember it, and that no one is likely to guess it.

It will ask for random keystrokes, and indicate a number showing the number remaining for you to enter at the bottom of the screen. Why? Nothing is more unique than the timing between sets of keystrokes from one person to another.

A computer could not possibly generate a set of numbers as random and haphazard as these timing values. Since it's been established that PGP is effective and it knows what it's doing, humor it. Type reasonably slowly. PGP will indicate that you've entered enough with a Beep and a message saying "-Enough, thank you."

A series of periods and pluses will show up at the bottom of the screen. These are of no concern to you, they're just progress indicators.

They look like this:

.....++++ .....++++ .....++++

When it's finished, you need to "extract" your public key from the public key ring in ASCII format so that you can mail it to the people who will use it (or pass it on diskette, or however you transmit it.) This is accomplished by typing:

pgp -kxa Your_ID keyfile pubring.pgp

So, For Gary Kline to get his key into a mailable file called "mykey.asc", he would do:

pgp -kxa kline mykey.asc pubring.pgp

As a side note, there are Environmental variables, specificly, PGPPATH, where you define the location of your Secret and Public Keys.

Is MS-DOS: SET PGPPATH="C:\PGP" (assuming PGP and it's files are in C:\PGP)

In AmigaDOS: setenv PGPPATH SYS:PGPAmiga

In UNIX C-Shell, setenv PGPPATH $HOME/pgp

In UNIX BASH, export PGPPATH="~/pgp"

Otherwise, you have to specifiy full path names in the commands, so in Gary's case, if he didn't set those variables, it would look like:

pgp -kxa kline@tao.thought.org mykey /home/kline/.pgp/pubring.pgp

A file called "mykey.asc" will be created, and voila! Your friends will add that keyfile to their own public ring and be able to mail you messages securely!

- **Getting Started: DECRYPTING MESSAGES**

Once your friends have your key and mail you a message with PGP encryption, you will need to save that message to a file. Assuming you've done that, and the PGP encrypted message is in a file called 'newmsg1.txt', we'll go through the motions.

pgp -d newmsg1.txt

PGP will ask for your secret passphrase. If entered correctly, PGP will decrypt it. It may ask you a few questions, answer them appropriately (ie: DO you want to overwrite file with file, etc.) Just answer them according to your wishes.

Now, using an editor or text viewer, you can read the message. If there is extraneous garble at the top, it means the person that sent the message signed it with the PGP key. Nothing is wrong, just ignore the garble. (This rarely occurs.)

Now, after reading the message, you should delete it. There's no security in the message once it's decrypted.. anyone could read it just as you did. You can keep the encrypted version if you tell PGP not to overwrite it in the decryption process, and decrypt it when you need to refer to it.

Here's a small excercise you can try right now if you've generated you keyset:

EDIT newmsg1.txt (replace 'edit' with the name of your editor, and write a short message to yourself.)

pgp -ea newmsg1.txt (encrypt the message. When PGP asks for the userID, specifiy your own.)

TYPE newmsg1.asc (You can use "cat" or "more" in lieu of "type" depending on what kind of computer you have.)

DELETE newmsg1.txt(delete the original plaintext. In Unix, replace 'delete' with 'rm', in MS-DOS, with "del")

pgp -d newmsg1.asc(PGP will prompt you for your password.)

TYPE newmsg1.txt (Or whatever filename you and PGP give the decrypted message.)

- **Getting Started:** ENCRYPTING MESSAGES, USING OTHER PEOPLE'S KEYS TO DO SO.

The first step is to obtain the public key of the person you intend to mail.. PGP is a two-way street and requires both people to have the software and have exchanged keys in order to communicate properly.

Once you have isolated their public key in a file, type:

pgp -ka keyfile [keyring]

where "keyfile" is the file containing their key.

(Remember: Once you add their key, you'll not need to do it again!)

PGP will ask you if you want to certify the key. Only if youare absolutely certain this key came from who it says it's fromthen YES, you want to certify it. (If you don't certify it, PGP will always ask you if you're SURE you want to use it each time you do!)

It will prompt you again, for verification, then ask for your secret passphrase. This is so no one but you can certify which keys you can trust for you. (There is a way to transfer trust, read the full documentation for more information on that.)

Once it's entered, the key is added to your public keyring and you'll never need to add it again.

Now, assuming you've just added Jane Doe's Public key to your keyfile and would like to mail her a message, you would type:

pgp -ea filename User_Id

Where filename is the message file, and User_Id is that of Ms. Doe, so something like:

pgp -ea doemsg.txt Jane

If there's more than one Jane in your public key file, but only one Doe, you would type:

pgp -ea doemsg.txt Doe

and pgp would produce a file called 'doemsg.asc' (or 'doemsg.txt.asc' on UNIX systems.)

Done! You would simply mail doemsg.asc to Jane Doe, and she would decrypt it with her secret key.

- **Getting Started: PGP QUICK REFERENCE**

Below are all the basic commands for PGP. Once you're familiar with basic use, read through the manual and use what's below as a reference, like a cheat-sheet.

Remember to add the 'a' option to anything producing an out-file, or it will output a BINARY that you cannot directly mail.

Ie: rather than pgp -e, use pgp -ea

The "a" means PRODUCE ASCII OUTPUT, which you can mail straight away.

- **Mailing files:**

In UNIX systems, you would type: mail username < "file" were "file" contains the output from pgp (usually file.asc).

- To encrypt a plaintext file with recipient's public key, type:

  pgp -e textfile her_userid [Other_Ids] (produces textfile.pgp)

- To sign a plaintext file with your secret key:

  pgp -s textfile [-u Your_Id] (produces textfile.pgp)

- To sign a plaintext file with your secret key, and then encrypt it with recipient's public key, producing a .pgp file:

  pgp -es textfile Recipient_Id [Other_Ids] [-u Your_Id]

- To encrypt with conventional encryption only:

  pgp -c plaintextfile

- To decrypt or check a signature for a ciphertext (.pgp) file:

  pgp ciphertextfile [plaintextfile]

The following command string will produce an encrypted ASCII file ("file".asc), signed with your secret key, with the recipient's public key, ready for mailing:

pgp -esa "file" Recipient_Id [Other_Ids] [-u Your_Id]"

For example, sending a file to Gary and Peter would be done like this:

The "Other_Ids" would be other recipients, so you could encrypt to more than one person at a time, making seperate files encrypted to each of them.

pgp -esa NewInfo.txt simons@peti gary@tao -u Jeremiah

So PGP would write two outfiles, one for Gary, one for Peter, signed with my key.

To generate your own unique public/secret key pair: pgp -kg

REMEMBER: When making any sort of outfile that you intend to mail (ie: creating encrypted mail messages) remember to add the -a extension.pgp -kx should be pgp -kxa, and pgp -e should ALWAYS be pgp -ea, otherwise, the output is unmailable binary data which cannot be viewed or otherwise used on most systems!

Key management functions:

- **Note that:**

pubring.pgp = Contains your & other's public files secring.pgp = contains your secret keys

[keyring] by default is PUBRING.PGP unless you specify otherwise. FYI: A "Keyring" is a file where PGP keeps keys in a format that it can quickly read while decrypting.

[[i'm thinking that you might want to explainn each term (like ``keyring") just after you use it for the first time... and then, having a Glossary of Terminology at the bottom of the tutorial would be a way of re-inforcing what the reader has just learned. this way, when he reads thru a 2nd time, he say, "Oh, yeah!"....]]

To generate your own unique public/secret key pair:

pgp -kg

To add a key file's contents to your public or secret key ring:

pgp -ka keyfile [keyring]

To remove a key or a user ID from your public or secret key ring:

pgp -kr User_Id [keyring]

To edit your user ID or pass phrase:

pgp -ke Your_Id [keyring]

To extract (copy) a key from your public or secret key ring:

pgp -kx User_Id keyfile [keyring]

To view the contents of your public key ring:

pgp -kv[v] [User_Id] [keyring]

To view the "fingerprint" of a given key:

pgp -kvc [User_Id] [keyring]

To check signatures on your public key ring:

pgp -kc [User_Id] [keyring]

To sign someone else's public key on your public key ring:

pgp -ks her_userid [-u Your_Id] [keyring]

To remove selected signatures from a User_Id on a keyring:

pgp -krs User_Id [keyring]

If you want to extract your public key to mail to someone:

pgp -kxa User_Id mykey [keyring]

Where User_Id = the first unique pattern of letters in your ID signature (ie: If you signature is Joe Blow <blowj@big.u.edu>, [[need to rewrite this below:]] then myid = joe) the result will be a file called mykey.asc, which you can mail to people:

mail user@host < mykey.asc

- **Getting Started: "STUPID" QUESTIONS**

I say "stupid" in quotes because the only stupid question is the one you didn't ask! If you knew everything, you wouldn't be reading this, and it's here to be helpful, not confusing!

Statement: pgp -ea file User_Id

Explaination: The file is the message to encrypt. The User_Id is the person you intend to send it to, in this example. "-e" means encrypt. "a" means ASCII output, presumably for mailable text.

When you specify a "user ID", you don't have to type the whole ID. In fact, most systems won't let you. PGP only needs a non-ambiguous clue.

(Peter Simons' ID is <simons@peti.GUN.de)

For example, if I write a message to Peter Simons, I can say "Peter", "Pete", "Simons", "Simon", "Simo", "peti.gun" or anything in his ID that isn't in another ID. If there a Simon Jackson in my public keyring, I should say Peter, because there are two occurances of "Simon". If there's a Peter Jennings in my public ring also, I should say "peti.gun", since that's unique to Peter Simon's ID.

Statement: Why isn't all output just ASCII in the first place?

Because sometimes you would want _BINARY_ output for one reason or another. Binary output produces a smaller file, so if you were putting the file onto a disk rather than mailing it, it would be a good idea to just use the binary mode. It's also used for things like STEALTH and stenography, which we won't go into here.

Statement: Why can't I encrypt with a secret key?

Because if you did, then ANYONE with your public key could decode it, assuming it were possible at all.

Statement: IF someone has my public key, can they figure out my password or hack my mail?

Absolutely not! That's the whole point to PGP in the first place!!

<><> PGP APPLIED <><>

As the Environment section implies, there's more to using PGP than enciphering Email. PGP is a way of doing things, not just a program. As I've stated before, and feel a need to emphasize, IF YOU DON'T USE PGP CORRECTLY, YOU RISK MORE THAN YOUR PRIVACY.

- **Environment:**

Make sure your workstation is free of shoulder-surfers, Password Echo, PGPPASS Environmental variables, Scripts containing your password in an unprotected mode, or any programs that might be intercepting keyboard input.

Residual Information, such as your original unencrypted documents, decrypted mail files, and UNDELETABLE files can be as much a compromise as no PGP at all.

See the Environment section of this document

- **Authentication:**

Failure to verify your keys with their supposed corresponding users is risking TOO MUCH to fail to justify even a long distance phone call. What's 30 cents against the compromise of your privacy?

See the section on Authentication in this document

- **Secure Password:**

Don't be a dummy. A secure password is multifaceted, but rotates around one thing: You're the only one that knows it. If it's written down, easy to guess, or possible to elicit from your computer, it's not a secure password!

See the Environment section of this document

Every element of PGP exists for a reason, and some parts that may seem irrelevent are actually important, maybe CRITICAL to certain privacy purposes.

Phil and his crew did not spend as much time as they did and PGP itself did not become as popular with everything from grass roots radicals to major conservatives to cryptographic experts for it's health.. it became that way because PGP offers all these features.

Keep this all in mind when you begin to think something in here is trite or tedious. It's there for a reason, and that reason is YOUR PRIVACY!

- **Advanced PGP**

This part of this text assumes you're now familiar with PGP, and comfortable with using it routinely. Whether you are or are not, you should read it. If it seems to complicated, don't worry about!

- **Advanced PGP: AUTHENTICATION**

So now you're comfortable with using PGP. You can encrypt, decrypt, make and add keys and all that good stuff. So now I throw you a curve-ball. How do you KNOW that YOU are encrypting a message to ME instead of someone else?

Roleplaying time! You have this key, which came in a message from me to you, and when you add it to your keyset, my name came up in the ID with my Email address. Okay. So you're ready to send me a message. WAIT RIGHT THERE!

As you should know, it's quite possible to forge Email, or get someone's account password! So, then, it's possible that you have a charlitan key! Someone could have easily generated a key with the same ID tag I have, broke into my account and mailed it to you, then all my incoming mail from you they would divert--read re-encrypt--with my real key and send it to me. And neither of us would know!

PGP solves that, too. Easily. Go back to key generation, briefly, and recall the keystrokes PGP asked for. There is no way anyone could do it like you did, and it's doubtful you could, either! When you generated that key, a part of it was the "fingerprint", which is totally unique to your key. Even if you lost your key and generated another one that looked the same, it's fingerprint would be totally different.

So, before you encrypt things to someone, you should compare the fingerprint on the key you have with the one they have over the phone or in person. THEN you would know you have the proper key and not a charlitan!

The fingerprint is also referred to as a fingerprint, and can been seen by invoking the command:

pgp -kvc User_Id keyringExample (pgp -kvc peter pubring.pgp)

There is also a way to "sign" a file. With this done,you can send an encrypted file, such as a letter containing technical data, sign it, and if ANYTHING is changed, PGP will know it and caution you.

This is done with pgp -sb filename.

Example (pgp -sb technote.txt) [[Q: how do i append my PGP sig or fingerprint in ASCII to a doc after i've encrypted it?]]

[[consistancy in this type of doc is a sound practice;why not made every ``Example" in this fmt?]]

This can come in handy for making sure no one changes instruction manuals to PGP itself, and more.

- **Advanced PGP: CERTIFICATION**

When you add a key to your keyset, PGP asks you if you want to certify the key.. do you KNOW that the key belongs to who you say it does? Do you trust that person to give you keys thatare authenticated? This is certification.

If Joe Blow hands you a floppy disc you watched him copy his key onto, you can be reasonably sure it's Joe Blow's key. So yes, you'll certify that. Of course. BUT.. If Joe Blow hands you a disk with other people's keys on it, do you trust that he checked those keys out reasonably well to make sure they're authentic? In other words, you can trust Joe Blow with his own key, but do you trust him to give you keys? If yes, how much? Always? Sometimes? Maybe? Never?

These are levels of trust. If you trust Joe Blow,and Joe Blow trusts John Doe, then it's possible that also John Doe is giving you keys, indirectly.

It's always best to get the key from the person themselves, check it out with them and do it that way, but it's not always possible, either for reasons of time quantity of work, and this is where Certification comes in. It's generally a wise idea to think things through as if it were a chess game, or setting up dominos.

Examples:

Joe Blow you trust. John Doe you don't. Therefore, you SOMETIMES trust Joe Blow.

Joe Blow you don't trust. Therefore, you NEVER trust anything he certifies.

Joe Blow you trust, John Doe you trust, Therefore, you USUALLY trust Joe Blow.

The only person PGP should understand you to trust fully is YOURSELF, and when you generate a key, that's the default setting.

- **Advanced PGP: KEY EDITING**

Okay. You're Peter Simons. You key reads:

Peter Simons <simons@peti.GUN.de>

But, you moved. Now you're Peter Simons, root@k-rad.elite.org.

You CAN edit your key's ID line without it messing up encryption. It's quite simple. You can use this function to also change your password should you feel the desire to do so.

pgp -ke simon

PGP would prompt you on editing options, first being the ID line and then being the password.

You should note that once you change it and lock in the change, PGP will remember the old ID and refer to it as an ALIAS. This way, it's more clear that it is the same key to other users.

People can always use the new or old key to encrypt to you, whether you change the ID and/or the password, however, they'll see the old ID unless you give them the copy of the new public key (pgp -kxa yourname mykey pubring) as if it were new.

- **Advanced PGP: COPYING SECRET KEYS, USING PGP IN TWO PLACES**

Let's say you're like me. You go to a university, and you use PGP offline most of the time, but.. once in a while, you use PGP online. In order to use the same key, you'll need to copy your SECRET keyring, secring.pgp, and put a copy of it where you intend to use it. If it's avoidable, you shouldn't do it, but sometimes it's not. Keyrings are inter-compatible. That is, they work on different computers regardless of whether it's a NeXT, an Amiga, a Mac or an IBM, or anything else.

In some cases, you might need to DISTRIBUTE a secret key, such as in a political organization or something. It's generally best to have a "data treasurer" for that sort of thing, but if you HAVE to do it, then it's done the same way a public key is, except for the keyring specified.

pgp -kxa User_ID SecretKeyFile secring.pgp

Remember that if you distribute it over mail, you would be foolish to distribute it in the same message as it's password, and even more foolish if you didn't encrypt the mail to the user you intended to send it to!

<><> Advanced Security Considerations <><>

Warning: The things presented in this segment of the document are surveillance techniques employed by various government, private and espionage organizations around the world. These are not likely to be employed to read your mail to your best friend, unless you happen to be conspiring to launch a nuclear missile.

Don't lose any sleep over this.

- **Advanced Security Considerations:**

Electromagnetic Interference Interception

Every electrical device, from digital wristwatches and toasters to televisions and mainframe computers generate electromagnetic interference. There are devices that measure this energy, and in some circumstances can interpret it into being able to tell what a given device is doing.

A computer's monitor is controlled by a signal send from the video card to the monitor (electromagnetic interference.) A remote device, carefully tuned in on this signal, could reproduce the image on your monitor remotely for the purpose of taping or monitoring.

The same is true with a computer keyboard. Whenever you press a key, a certain signal is sent to the computer, different from other signals sent by other keys. A device like the one described above could essentially carbon copy all of your key presses into a recorder and everything you type could be reproduced.

If you want a working example of this concept, look at a typewriter ribbon (especially those found in IBM Selectric series typewriters.) If you look carefully and fill in the spaces mentally, you can see everything the unwary typist has typed. On the Selectric, spaces aren't shown on the ribbon, since the space prints nothing and would be a waste of ribbon to advance the ribbon when you hit it. (Same with Tab, Return, etc.)

- **Advanced Security Considerations:**

Hard disk reading

If you format your hard drive so that there is no data on it at all, it is still possible to pick up trace magnetic signals where readable data and the previous formatting existed. With special equipment, the contents of your hard drive could be totally reconstructed, despite the formatting.

The solution is straight forward: Department of Defense standard Data Deletion, which was described in the beginning. It overwrites the file 3 times with 1's and 0's before deleting, so the residual data is not usable in any scheme.

- **Advanced Security Considerations:**

Remote Video Monitoring

Obviously it's possible for someone to videotape your computer screen and/or your fingers on the keyboard. This is a standard tactic. This is avoided somewhat by positioning the computer where neither the keyboard or the monitor is visible through a window, and that there is no reflection visible either, as could be seen in the user's glasses, a mirror, a glossy poster, chrome on furniture, etc.

- **Advanced Security Considerations:**

Linetap

If you were to use PGP on a remote system, your modem line could be compromised by buffering the signal transparently into another computer and thusly reproducing the entire terminal session. For that reason, it's better to use PGP offline and upload encrypted texts.

- **Advanced Security Considerations:**

Modifications

There is no way to tell if PGP has been modified unless you get the distribution package from it's creators, or get the source code, carefully examine it, and compile it yourself. Even then, it's possible to have a compiler that recognizes security applications and creates a "backdoor".

Although there are lots of ways to lessen the likelyhood of tampering, it's a game of Better Mousetrap, Smarter Mouse.

The more common scenario is straight-forward: Someone modifies the source on a shared system and gets a dump of everything you've done with PGP on that system. The chance of this is somewhat eliminated by compiling your own copy on the system, or better, simply use your own copy offline!

Correspondence:

This was written by Jeremiah S.Junken, with a few additions by Gary Kline and the key-reference chart which was taken from PGP 2.2 as compiled on the Indian NeXT cluster, UCS of IU Bloomington.

In the event of address change or the like, I refer correspondence to Peter Simons, the author of PGPAmiga and the maintainer of the PGPAmiga mailing list.

Please address correspondence related to this to me. Although Peter is a great guy who knows PGP intimately and loves helping people out with it, he's also extremely busy, so keep that in mind before you mail him specificly!

Always read alt.security.pgp if you need more information, and/or subscribe to the PGPAmiga mailing list! (contact Peter Simons)

I would like to thank:

Peter Simons Especially, the man who ported PGP to the Amiga and isresponsible for a zillion programs of merit for the Amiga networking.

Gary Kline Who's enthusiasm, assistance, and insightful editorial commentary have made this a much better document.

Phillip Zimmerman For coding PGP in the first place.

And the many people who sent back a lot of positive feedback on this beast, and offered help and suggestions on bringing this to full fruition.

PGP Distribution Information:

PGP is found in compiled form for Amiga, MS-DOS, Mac, and, (I THINK..) Atari ST. The C language source code is also available.

FTP to SODA.BERKELEY.EDU (don't be a hoser)...

/pub/cypherpunks/pgp

there you will find several versions, and compiled versions for Amiga, Macintrash, MS-DOS, etc., as well as other cool things.

net-dist.mit.edu (Source, MS-DOS executables) src.doc.ic.ac.uk (Source, Amiga, MS-DOS, Macintrash) ftp.luth.se(Amiga /pub/aminet/util/crypt) wuarchive.wustl.edu (Everything)

Notes:

This revision contains extra information on authentication and certification that I skimmed in the previous version out of laziness.

I've had two comments regarding my mention of UNIX commands. When I say UNIX, I'm referring to the command syntaxes used in the version of UNIX running on NeXT machines running NeXTstep 3.1 or higher. This is a BSD derivative that is similar to anything running BSD4.2 or higher, NetBSD, Linux, etc. If you don't know what you're system's running, you should be able to get information on any standard command's syntax with the 'man' command. For example: 'man ls' will give information on the arguments and syntax for the directory listing command.

PGP Supplementary Stuff

Stealth - A "Stenography" program which strips RSA headers and indentifying crypto signatures so PGP encrypted material may be passed off as GIF images and/or sounds, or imbedded within such files.

PGPSendMail - A SendMail replacement for Amiga (being ported to UNIX as this document is being written) that provides streamlined ciphering features fully integrated with any mail system. (THANKS PETER!!)

iSpell - A spell-checking program anyone using PGP should have to check texts before they're enciphered. Versions exist for MS-DOS, Amiga, UNIX (as Source code), etc.

PGP FrontEnds - There are tons of little utilities for Windows, etc. that add a GUI to PGP. I've seen half a dozen for MS-DOS, so if you like point'n'click conviencience, you might considering trying to find one of these.

Remember that PGP needs your help and support to continue to exist and be able to be used. At this point, the United States and other world governments are opposed to secure cryptography and are trying to make it's use illegal. With this in mind, I urge you to FTP to ftp.eff.org, and check out the /alerts directory, and grab some information on the Electronic Frontieer Foundation. The EFF is, essentially, the first activist/civil liberties group that deals only with the electronic world, (the Internet, etc..).

Also, the author of the original version, Phillip Zimmerman has accumulated some expenses keeping himself out of jail due to the legal entanglements and contraversy surrounding PGP. If you would care to help this man who has put his freedom on the line to help insure yours, you are encourages to make a donation, ANY donation to his legal defense fund. Mailings go to his lawyer.

Phillip Zimmerman Legal Defense Fund
c/o Phillip Dubois, Attorney at law.
2305 Broadway
Boulder, Colorado 80304