Pretty Good Talk on Pretty Good Privacy (PGP)

Krishna Vedati

<vedati@PACorp.com>

KeyID: 283C0ECD Key fingerprint: BB F2 4D 85 B8 67 AE 5D ED F4 FF CF DC BB 63 EA

Trail Map

* **Introduction.**
* **History of PGP.**
* **Getting and Installing PGP.**
* **Simple User Setup.**
* **Key Generation Operations.**
* **Encryption and Decryption of Files.**
  - How to encrypt a file.
  - Encrypting Email Messages.
  - Encrypting Messages for Multiple Users.
* **Signing or Authentication.**
  - How Signing Works?
  - Signing Email messages/keys/documents.
* **Key Manipulation Operations.**
  - Adding/Deleting Keys.
  - Extracting User's public key for publishing.
  - Changing Key properties.

Trail Map (Continued)

- Revoking Keys.

* **The Web of Trust.**
* **Key Server Operations.**
  - Publishing your public key on the internet.
  - Obtaining somebody's public key.
* **Configuring PGP Defaults.**
* **Books and on-line documents on PGP.**

Introduction

* **What is PGP?**

  PGP is a program that uses public and conventional cryptography techniques to protect the privacy of your electronic mail and the files you store on your computer. You can also use PGP as a tamper proof digital signature system, allowing you to prove that files or electronic documents have not been modified.

* **What can PGP do?**

  - Encrypt files.
  - Create secret and public keys.
  - Manage keys.
  - Can encrypt and decrypt electronic mail.
  - Can electronically sign documents.
  - Can certify other people's keys
  - Can revoke, disable, and escrow keys.

* **Why PGP?**

  - Using PGP users can reliably/securely exchange messages even when using insecure channels of communication.
  - Once the message is encrypted only the intended recipient can decrypt it; nobody else can, not even the sender.
  - Fast, practical, efficient and COOL!

History of PGP

* **Single Key or Conventional ciphers.**
  - One single key for sender and recipient.
  - Messages can be decrypted by anyone who possesses the key.
  - Key distribution problems.
* **Public Key Cryptography (RSA)**
  - 2 keys, an encryption or public key and a decryption or secret key
  - The recipient creates both keys.
  - The decryption key is kept secret and the encryption key is published
  - RSA algorithm is based on the mathematics of exponentiation.
* **PGP is a hybrid program**
  - Creates a random conventional encryption key (session key)
  - It sends the session key to the recipient using a header block that is encrypted using the RSA public key encryption. RSA serves as the secure channel.
  - The bulk of the message is encrypted with the session key using a conventional cipher, IDEA.
  - PGP uses IDEA because RSA encryption is too slow to encrypt the whole message.
  - IDEA is thought of as a stronger encryption method than RSA, so this scheme does not weaken PGP.
  - PGP uses conventional encryption to protect your secret keys.

Getting & Installing PGP

* **Getting PGP:**
  - <ftp://net-dist.mit.edu/pub/PGP/README> and follow the instructions.
  - You can also obtain PGP through <http://web.mit.edu/network/pgp-form.html>
  - Please read all the instructions before downloading the distribution.
* **Installing PGP on a UNIX system:**
  - `gzip -d pgp262s.tar.gz | tar xvf -`
  - Build RSAREF library.
  - Build pgp.
* **Verify Your Copy of PGP:**
  - **`src/pgp -ka keys.asc distkeys.pgp`**: PGP will create a distkeys.pgp key ring using the keys in keys.asc key file that comes with the distribution.
  - **`src/pgp pgp262si.tar.asc`** : PGP will prompt for a keyring file < enter distkeys.pgp>; PGP will verify the checksum informing you whether your PGP distribution has been tampered with or not.
  - Run the sample MIT tests in setup.doc file before installing.
* **Read setup.doc on how to install PGP for system-wide use.**

Simple User Setup

* **Create a $(HOME)/.pgp (PGPPATH environment) directory.**

```
unix% mkdir ~/.pgp; chmod 700 ~/.pgp
```

* **The path specified by PGPPATH environment variable is the default location for public and secret key rings and PGP configuration file.**

* **Copy PGP on-line documentation files to $PGPPATH directory.**

```
unix% cp /usr/local/lib/pgp/pgpdoc*.txt $PGPPATH
```

* **Copy PGP configuration file to $PGPPATH directory.**

```
unix% cp /usr/local/lib/pgp/config.txt $PGPPATH
```

* **Make sure that pgp executable is in your path and export the PGPPATH environment variable.**

```
unix(ksh)% PGPPATH=$HOME/.pgp; export PGPPATH
unix(ksh)% PATH=$PATH:/usr/local/bin;export PATH
```

Note:PGP 2.6. will not generate keys for you until it finds the PGP documentation (pgpdoc*.txt) files in the directories it searches. (Check PGP_SYSTEM_DIR variable while building PGP to set the default location for PGP documentation).*

Key Generation Operations.

* **To Create PGP keys:**

```
unix% pgp -kg
```

* **PGP asks for RSA key size**
  - 512 bits- Low commercial grade, fast but less secure.
  - 768 bits- High commercial grade, medium speed, good security.
  - 1024 bits- “military” grade, slow, high security.

  Key Size is not changeable once keys are generated.

* **PGP asks for your User ID**

  XEmacs is Great!!! <xemacs@pacorp.com>

  - You can change your User ID or add a new one after keys are created.

* **PGP asks for a pass phrase (sort of password)**
  - Pass phrase can have many words, spaces, punctuation and any printable characters. ~~*(no one knows my :) pass phrase)*~~
  - Don’t write your pass phrase anywhere. Don’t reveal it to any person.

Key Generation Operations (Continued)

* **PGP generates 3 files.**
  - **pubring.pgp –** file that contains all the public keys including the user's.
  - **secring.pgp –** file that contains all the private keys of the user.

    Note: a user can have multiple PGP private keys under multiple user ids.
  - **randseed.bin –** file that contains the random number generation information, which PGP uses internally.

Encryption/Decryption Operations.

* **To encrypt a plain text file (foo.txt) to send in email to user1:**

```
unix% pgp -eat foo.txt user1

unix% pgp -eat <file name> <users list>
```

  - This operation needs user1's public key.
  - -e options tells PGP to encrypt file foo.txt.
  - -a options tells PGP to generate an ascii armor (mailable file) after encryption.
  - -t option suggests translate characters between PCs Macs and Unix work stations for message portability.

* **To encrypt for multiple users:**

```
unix% pgp -eat foo.txt tom vedati
```

* **To encrypt and mail file foo.txt using one command:**

```
unix% pgp -feat tom < foo.txt | mail tom@pacorp.com
```

* **Combining typing, encrypting and sending mail:**

```
unix% pgp -feat tom | mail laura@pacorp.com
```

end the message with ^D.

Encryption/Decryption Operations (Continued)

* **To decrypt an encrypted file:**

```
unix% pgp foo.txt.asc -o newfile
```

- -o outputs the decrypted file to ‘newfile’.
- You can not decrypt messages that are encrypted for other people.

PGP will ask for your pass phrase before decrypting the file. Be wary that the decrypted form is now stored in ‘newfile’ where others can possibly read it.

* **PGP can be used to do conventional encryption also:**

```
unix% pgp -c foo.txt
```

- Uses conventional encryption
- Avoids RSA algorithm and avoids the necessity to muck around with keyrings.

Encryption/Decryption with Mailers.

* **Packages like elm-2.4, VM(XEmacs), MH have built in support for automatic encryption and decryption of messages using PGP while composing and reading mail.**

* **These mailers will also support all the PGP functionality like:**

  - Message encryption/decryption.
  - Snarfing keys from messages.
  - Signing messages before sending.

* **VM needs mailcrypt and MH users need to set some configuration options (refer to MH FAQ)**

* **For other PGP related utilities check the URL**

  - **<http://draco.centerline.com:8080/~franl/pgp/utilities.html>**

Signing or Authentication

* **PGP digital signature**
  - A special number that is cryptographically produced and digitally verified which makes it possible for you to mathematically verify the name of the person who signed the message.It also shows that the person who signed the document had access to the PGP secret key and pass phrase for the key indicated by the signature and that the document has not been modified since it was signed.

* **How does signing work:**
  - When the sender creates the digital signature, PGP first scrunches the message to an 128 bit cryptographic checksum, using MD5. Cryptographic checksum algorithms are constructed so that they are difficult to reverse.That is, given a given cryptographic checksum, it is difficult to find a message that will map to it. RSA encryption is applied to the cryptographic checksum using the user's secret key yielding the digital signature. The digital signature is combined with the original message to create a signed message.
  - To check a digital signature, PGP applies RSA decryption using the public key to the signature. It applies MD5 to the message to yield a cryptographic checksum. Since public key decryption is the inverse of secret key encryption, these two values should be equal. If so, the digital signature checks.

Signing (Continued)

* **To sign a file:**

```
unix% pgp -sat foo
```

  - Will sign the file foo with your signature and creates a foo.asc ascii file.

* **To verify the signed text do:**

```
unix% pgp foo
```

  - If the file foo.asc is modified by any one (including the user) PGP will complain.
  - *Note: The signer's public key must exist on your public key ring to verify files signed by that person.*

* **To sign a user's public key:**

```
unix% pgp -ks tom
```

  - Will sign tom's public key certifying that you believe and verified it is really his public key.
  - *Note: Before certifying someone's public key you should have them read the finger print for their public key (obtained via 'pgp -kvc [user_id]') and verify that it matches the finger print on the public key you were given.*

Key Manipulation Operations

* **To List all the key manipulation functions:**

```
unix% pgp -k
```

* **To add a key file's contents to your pubic or private key ring:**

```
unix% pgp -ka keyfile [keyring]
```

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

```
unix% pgp -kr userid [keyring]
```

* **To edit your user ID or pass phrase:**

```
unix% pgp -ke your_userid [keyring]
```

* **To extract a key from your public or secret key ring:**

```
unix% pgp -kx your_userid keyfile [keyring]
```

* **To extract a key from your public or secret key ring into an ascii mailable format:**

```
unix% pgp -kxa your_userid keyfile [keyring]
```

Key Manipulation Operations (Continued)

* **To view the contents of your public key ring:**

```
unix% pgp -kv [userid] [keyring]
```

* **To view the contents of your public key ring in verbose format (shows who signed the keys):**

```
unix% pgp -kvv [userid] [keyring]
```

* **To check signatures on your public key ring:**

```
unix% pgp -kc [userid] [keyring]
```

Will show you all the keys on your key ring and then, for each key, shows you who is attesting to its validity.

* **To check all keys finger prints:**

```
unix% pgp -kvc [userid] [keyring]
```

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

```
unix% pgp -ks her_uid [-u youruid] [keyring]
```

* **To remove selected signatures from a userid on a key ring:**

```
unix% pgp -krs userid [keyring]
```

Key Manipulation Operations (Continued)

* **What is a key Revocation Certificate?**
  - Contains a copy of your public key
  - Is signed by your secret key
  - Is incorporated into a person's key ring
  - Prevents a person who receives it from using your public key

* **To make your Revocation Certificate:**

```
unix% pgp -kd your_userid
```

PGP asks whether to generate secret key compromise certificate, answer yes and then send you public key to all the people, whom you do not want them to user you public key any more.

You can not create a key revocation certificate if you forget your pass phrase.

* **To Disable a public key:**

```
unix% pgp -kd userid
```

The Web of Trust

* **Key Certification (trust):**
  - When PGP uses a key for encryption or signing, it determines if in PGP's opinion, the key can be trusted. If PGP does not trust the key, it will print an message warning you that the key is not to be trusted. You can tell PGP to use the key anyway. PGP determines trust on the basis of signatures from trusted keys. When you add a key to your public keyring you are asked if the key can be trusted to introduce other keys. If a PGP notes a signature from a trusted key, it tends to trust the key bearing the signature. You can change the trust parameters on a key using the -ke option. Contains a copy of your public key
* **Trust is not transitive.**
  - If you trust a person, you do not necessarily trust everyone that that person trusts! You can believe that the person you trust is naive and that therefore the persons that he trusts is not necessarily trustworthy. Thus the PGP configuration option CERT_DEPTH should be set to 2.
* **You should not expect other people to trust your inferences. If you infer that a given key is good on the basis of key signatures, you should not sign that key.**
* **You should only sign a key when you know of your own personal knowledge that a key is valid. Make sure that the finger prints match.**

PGP Key Servers

* **The user can publicize his public key on the internet**

* **Can get other people's public keys from the internet**

* **For PGP key server operations send mail to:**

pgp-public-keys@pgp.mit.edu

With one of the following **subject** headers as described below:

- **help**

  Sends you instructions.

- **add**

  Adds the key to the database. The content of the mail message should contain your public key in the ascii format (pgp -kxa).

- **index**

  List all of the PGP keys on the server (-kv).

- **verbose index**

  List all of the PGP keys with the verbose format (-kvv).

PGP Key Servers (Continued)

* **`For PGP key server operations send mail to:`**

  pgp-public-keys@pgp.mit.edu

  With one of the following **`subject`** header as described below:

  - **`get userid`**

    Gets just one person's key (-kxfa ***userid***).

  - **`get`**

    Gets the whole public key database.

  - **`mget regexp`**

    Gets all the keys that match ***regexp.***

  - **`last days`**

    get all the keys updated within last ***days*** **.**

Configuring PGP Defaults

* **Edit the file config.txt in $PGPPATH and change the variables to set up personal defaults.**

* **Do not delete any of the configuration variables because you may find it had to reconstruct them later.**

* **Make a copy of the your configuration file before you edit it because if you screw up it PGP might not work.**

* **Specifying a Configuration Variable on the Command Line:**

```
unix% pgp -eat bigfile Sam +ARMORLINES = 1000
```

Will split the encrypted message into chunks 1000 lines.

Books and on-line documents on PGP

* *PGP Pretty Good Privacy* **by *Simson Garfinkel***

* *Protect Your Privacy A Guide for PGP users* **by William Stallings**

* *E-MAIL SECURITY How to keep Your Electronic Messages Private* **by Bruce Schneier**

* *The Computer Privacy Handbook* **by Andre Bacard**

* *The Official PGP User's Guide* **by Philip R. Zimmermann**

* *PGP Companion for Windows The official Guide to WinPGP* **by Peter Kent**

* Check out ***<http://www.efh.org/pgp/pgpwork.html>*** for an excellent introduction on PGP.

Thanks To

* *Laura Rose <laura@PACorp.com>*

* *Subu Darbha* <sdarbha@PACorp.com>