Lab Handout 6: SSH and GPG

Secure Shell (SSH)

SSH or Secure Shell is used to login to a computer enables it communicate and share data with another device. An inherent feature of ssh is that it allows the user to control the computer s/he is accessing.

Below is the usual format of an SSH command:

ssh <username>@<IP address>

where ssh is the command to invoke the secure shell program,
username is the username of the account you want to log-in to, and
IP address is the IP address of the host where the username resides.

Example:

ssh dcspclab1@10.0.3.234

GPG

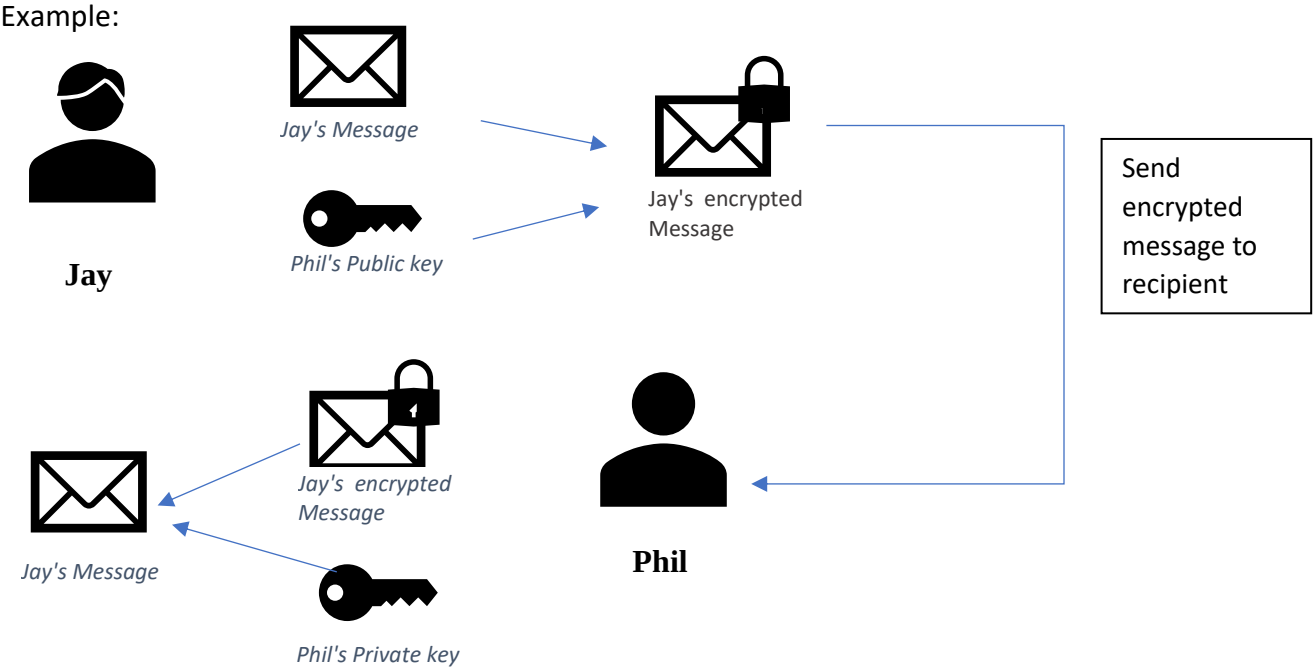
GPG is a tool for secure communication which allows you to encrypt and/or decrypt data given a key. If you did not protect your data, the information is susceptible to security problems such as man-in-the-middle attacks and other security threats such as sniffing.

Public Key Cryptography

- This is a method of encrypting data with two different keys and making one of the keys, the public key, available for anyone to use. The other key is known as the private key.
- Data encrypted with the public key can only be decrypted with the private key, and data encrypted with the private key can only be decrypted with the public key.

1. **Public Key**
   This key may be given to anyone with whom the user wants to communicate. The sender of data needs the public key of the recipient so that s/he can encrypt the data.
2. **Private Key**
   This key must be kept secret. This will be used to decrypt messages encrypted using your public key along with a passphrase.

Example:



Jay wants to send Phil a message over the network (they are using the internet now cause the telephone is too old school).

The message Jay wants to send to Phil is very confidential ("Hi"). So Jay decided to encrypt the message using Phil's public key. Once the message is encrypted, he sent it to Phil.

Phil then received the message and saw that it is just gibberish. He then realized that maybe it is encrypted. So he used his private key and decrypted Jay's encrypted message. After the decryption, Phil was now able to see Jay's message. It was just "Hi."

HOW TO USE GPG

Generate new keypair	gpg --gen-key It will ask for the following information: 1. What **kind** and **size of key** you want; the defaults are probably good enough. 2. How long the key should be valid (**expiration**). You can safely choose a non-expiring key for your own use. If you plan to use

	a key for public signing, you might want to consider a yearly expiration. 3. Your **real name** and **e-mail address**; these are necessary for identifying your key in a larger set of keys. 4. A **comment** for your key, perhaps to distinguish a key used for special tasks like signing software releases. The comment can be empty. 5. A **passphrase**. Whatever you do, *don't forget it!* Your key, and all your encrypted files, will be useless if you do.
Exporting a public key *(Generate an ASCII version of your public key)*	gpg --armor --output **<username>-pubkey.txt** --export **'User Name'**
Importing a public key *(Include a public key to your ring)*	gpg --import **<username>-pubkey.txt**
List public keys in your ring	gpg --list-keys
List private keys in your ring	gpg --list-secret-keys
Encrypt a file *(Public key of the recipient should be in your ring)*	**# the long version** gpg --encrypt --recipient **'User Name' message.txt** **# using terse options** gpg -e -r **'User Name' message.txt**
Decrypt a file *(This will make use of the **private key** and **passphrase**)*	gpg --output **message.txt** --decrypt **message.txt.gpg**
Shows the fingerprint of the public key	gpg --fingerprint **'User Name'**
Delete public key	gpg --delete-key **'User Name'**
Delete private key	gpg --delete-secret-key **'User Name'**