Getting Started with GPG on OS X

The GPG Suite for OS X gives you a simple GUI for creating and managing your GPG keys, as well as plugins for integrating PGP with your OS X programs. The following instructions take you through installing GPG Suite and generating a keypair. The settings suggested here will increase the security of your key and are strongly recommended!

If in doubt, come on `#opensourcecornell` on Freenode and ask for help!

Install GPG Suite and Setup Keypair

1. Go to `https://gpgtools.org/` and download the *GPG Suite*.
2. Run the installer.
3. The application "GPG Keychain Access" will pop up after the installation is complete. This will allow you to generate a new keypair. Keep "Upload key after generation." unchecked. Enter your name and email address, click "Advanced Options," and increase the key length to 4096. You don't need (or want) a comment. **Take note of when your key expires! You can always change the expiration date later!**
4. Generate the keypair. You'll need to enter a passphrase and wait for the keypair to be generated. This passphrase is the last line of defense if your private key is compromised, and you will need to enter it when you sign or decrypt messages. **Do not forget this passphrase!**
5. Once your keypair has been generated, select it, go to the menu, and select "Key>Generate Revoke Certificate…". Don't give a reason (this is in case anything ever happens). Your revocation certificate will let you revoke this key if it ever gets stolen. Keep it in a secure place (not, for instance, on your desktop).
6. Go to the menu and select "GPG Keychain Access>Preferences…". Change the keyserver to `hkps://hkps.pool.sks-keyservers.net`. The HKPS protocol is more secure, because it sends the keys over an encrypted connection.
7. If you have more email addresses and names that you plan to use this key under, double click on the key, go to the User IDs tab, and click the first `+` button. You will be prompted to add another name and email address (and comment, which again, you probably don't need or want). By selecting a User ID, you can click the "primary" button to make it the name and email that others will see by default.
8. When you are done setting up your key, select it, go to the menu and select "Key>Send public key to Keyserver".

Integration with Finder

You can now sign, encrypt, and decrypt files in Finder using your new key.

1. Open the directory containing the file you want to sign, encrypt, or decrypt.
2. Select the file in Finder.
3. **To sign the file:** Go to "Finder>Services>OpenPGP: Sign File".
4. **To encrypt the file:** Go to "Finder>Services>OpenPGP: Encrypt File".
5. **To decrypt the file:** Go to "Finder>Services>OpenPGP: Decrypt File".

Integration with Apple Mail

You should be able to sign messages with your new key, and encrypt messages to anyone whose public key you have.

1. Start composing a message. You'll see two extra buttons in the window: a lock and a star.
2. **To sign your message:** (a good idea for all your messages) make sure there is a check in the star button by clicking it.

3. **To encrypt your message:** (if you have the recipient's public key) make sure the lock is closed by clicking it.

You will also be able to receive and decrypt encrypted messages to your key, and verify signatures of signed messages if you have the sender's public key. To do this, just open the message from your inbox.

Signing Someone Else's Key

When you sign someone else's key, you are saying that you have verified that the information in the key is correct and that they own the key. To do this, you will need to see photo identification. **Never, never sign a key if you haven't verified all of its information!**

Next, generally after the keysigning party, you do the actually keysigning. You need to make sure that the person owns all the email addresses they say they do. The way you verify this is a little convoluted: for each User ID, you sign it separately, export it, and send them encrypted to the email address you signed. If the key has three User IDs, and the person only controls two of them, only those two emails can be encrypted, and so you will effectively only have signed those two!

Unfortunately, the GPG Suite doesn't give a good way to do this. You will need to do the following:

1. Once you have verified their name, download their public key by going to "Key>Retrieve from Keyserver...".
2. Export the original key by selecting it and clicking "export".
3. Double click the key and go to the "User IDs" tab.
4. For each User ID, do the following exactly:
   (a) Select the User ID.
   (b) Click the + button below the "Signatures" field.
   (c) Select how carefully you checked this key. This should almost always be either "I will not answer" (which is perfectly fine) or "I have done casual checking."
   (d) You may uncheck the box that says "Signature expires" if you want. Most signatures do not expire. It is slightly more secure if your signature expires, but you will need to keep track of this and go to another keysigning party with the person and repeat this entire section again when resigning.
   (e) After generating the signature, close the "Key Inspector" window and export the new key as a separate file.
   (f) Encrypt this file, and send it by email to the email address you just signed.
   (g) Delete the key from "GPG Keychain Access" by going to "Key>Delete," and import it again from the original file. If you do not do this, the next exported key will have more than one User ID signed, which defeats the purpose of this! Go back to the "Key Inspector" window as above.
5. When the owner receives all the signatures, they will upload the new key to a keyserver, which you can then download.

Having Your Own Key Signed

If you are on the receiving end of the above process, be sure you bring a photo ID (or more!) and your key information with you to the keysigning. Be sure to have written copies of your key id, fingerprint, and all your user IDs to give to the keysigner, so they can sign your key after the party. You can find this by double-clicking on your key.

Following the party, check your email addresses for encrypted messages with the signatures. Decrypt the messages, and import the attachments into "GPG Keychain Access". This will apply the signatures to your key. Then, select the key and go to "Key>Send public key to Keyserver" so others can download the newly signed key.

Getting Started with GPG on GNU/Linux or UNIX

Most GNU or UNIX systems come with GPG, either installed by default or in their package repositories. This page assumes that you have installed GPG. (As a hint, look for a package like `gpg`, `gnupg`, or (in the case of Debian-derived systems) `gpg2`.)

Most of the following are commands to be entered in the terminal. (As a note, Debian-derived systems should use the `gpg2` command in place of the `gpg` command.)

If in doubt, come on `#opensourcecornell` on Freenode and ask for help!

Setup GPG

Although the default settings for GPG are mostly acceptible, it is recommended that you change some of the settings for better security. To do so, open up the file `~/.gnupg/gpg.conf` in your favorite editor and do the following:

1. Search for a line that looks like `#keyserver hkp://keys.gnupg.net`. Comment all such lines out, and add the following line below it:

   ```
   keyserver hkps://hkps.pool.sks-keyservers.net
   ```

2. At the end of the file, add the following lines:

   ```
   personal-cipher-preferences AES256 TWOFISH AES192 AES
   personal-digest-preferences SHA512 SHA384 SHA256
   personal-compress-preferences ZLIB BZIP2 ZIP
   cert-digest-algo SHA256
   ```

Setup Keypair

1. Open a terminal.
2. Run the command `gpg --gen-key`
3. When prompted, select option 1, "RSA and RSA (default)".
4. Use 4096 bits instead of the default.
5. Give an expiration date for the key (2y or 3y is a good choice). This can always be changed later, but it's good if you lose the key at any point in the future. **Take note of when your key expires! You can always change the expiration date later!**
6. Enter your name, enter your email address, but do not enter a comment.
7. Enter a passphrase and wait for the key to be generated. This passphrase is the last line of defense if your private key is compromised, and you will need to enter it when you sign or decrypt messages. **Do not forget this passphrase!**
8. Once your keypair has been generated, take note of the key ID that was printed to the terminal (it will look like `pub 4096R/KEYID`). Next, run the command `gpg -o revcert.asc --gen-revoke {KEYID}`. This will generate a revocation certificate, which allows you revoke this key if it ever gets stolen. Don't give a reason (this is in case anything ever happens). Keep it in a secure place (not, for instance, right in your home directory).
9. If you have more email addresses and names that you plan to use this key under, run the command `gpg --edit-key {KEYID}`. At the prompt that appears, type `adduid`. Again, enter your name and email (and no comment). Do this for each email you want to add. When you are done, select the primary user ID by running the commands `uid {UID NUMBER}` and `primary`. Exit by typing `quit`, and be sure to save your changes.
10. When you are done setting up your key, run the command `gpg --send-key {KEYID}` to submit your public key to the keyserver we set up.

Signing, Verifying, Encrypting, and Decrypting Files

You can now use the `gpg` command to sign, verify, encrypt, and decrypt files:

```
$ gpg -s filename                # Sign a file, makes filename.gpg
$ gpg --clearsign -a filename    # Sign file in same file, makes filename.asc
$ gpg --verify filename.asc      # Verify a signature or clearsigned message
$ gpg -e filename                # Encrypt a file, makes filename.gpg
$ gpg -d filename.gpg            # Decrypt a file and print its contents
```

Integration with Thunderbird, Evolution, etc.

Thunderbird, Mutt, Evolution, and KMail all integrate with GPG very easily, but there isn't enough space here to describe how. Either Google it (Thunderbird uses something called Enigmail, the others do so natively) or ask one of us to help.

Signing Someone Else's Key

When you sign someone else's key, you are saying that you have verified that the information in the key is correct and that they own the key. To do this, you will need to see photo identification. **Never, never sign a key if you haven't verified all of its information!**

Next, generally after the keysigning party, you do the actually keysigning. You need to make sure that the person owns all the email addresses they say they do. The way you verify this is a little convoluted: for each User ID, you sign it separately, export it, and send them encrypted to the email address you signed. If the key has three User IDs, and the person only controls two of them, only those two emails can be encrypted, and so you will effectively only have signed those two!

Follow these steps:

1. Once you have verified their name, download their public key running `gpg --recv-keys` {`THEIRKEYID`}
2. Export the original key by running `gpg --armor -o key.orig --export` {`THEIRKEYID`}
3. For each User ID, do the following exactly:
   (a) Run `gpg --edit-key` {`THEIRKEYID`}, and at the prompt, type `uid` {`UIDNUMBERTOSIGN`}.
   (b) Type `sign` to sign the user ID, and then `quit` to exit this prompt.
   (c) Export this key to a different file: `gpg --armor -o key.uid`*n*`signed --export` {`THEIRKEYID`}.
   (d) Encrypt this file, and send it by email to the email address you just signed.
   (e) Delete the signed key (`gpg --delete-keys` {`THEIRKEYID`}) and reimport the original (`gpg --import key.orig`). If you don't do this, the next exported key will have more than one User ID signed, which defeats the purpose of this! Repeat for the remaining User IDs.
4. When the owner receives all the signatures, they will upload the new key to a keyserver, which you can then download.

Having Your Own Key Signed

If you are on the receiving end of the above process, be sure you bring a photo ID (or more!) and your key information with you to the keysigning. Be sure to have written copies of your key id, fingerprint, and all your user IDs to give to the keysigner, so they can sign your key after the party.

Following the party, check your email addresses for encrypted messages with the signatures. Decrypt the messages, and import the attachments using `gpg --import filename`. This will apply the signatures to your key. Then, send the newly signed key to a keyserver so others can download it by running the command `gpg --send-keys` {`KEYID`}.

Getting Started with GPG on Windows

To use GPG on Windows, you'll be using the GPG4Win distribution. Most of the following are commands to be entered in the command prompt. The full distribution of GPG4Win has a GUI application to generate keys, but it is not reliable and is rather buggy, so we suggest you just use the command line version for generating keys.

If in doubt, come on `#opensourcecornell` on Freenode and ask for help!

Install and Setup GPG4Win

1. In your browser, navigate to `http://www.gpg4win.org/` and click on the download link. You can download GPG4Win Lite, which does not contain the graphical key manager program (which is what we recommend, because it is very buggy on Windows) or the full GPG4Win version. Either way, you will be creating your key from the command line, to ensure the key is made properly.
2. Follow the steps through the installation, and use the default steps.

Although the default settings for GPG are mostly acceptible, it is recommended that you change some of the settings for better security. To do so, open up the file `C:\Users\{User}\AppData\Roaming\gnupg\gpg.conf` in your favorite editor and do the following:

1. Search for a line that looks like `#keyserver hkp://keys.gnupg.net`. Comment all such lines out, and add the following line below it:

   ```
   keyserver hkps://hkps.pool.sks-keyservers.net
   ```

2. At the end of the file, add the following lines:

   ```
   personal-cipher-preferences AES256 TWOFISH AES192 AES
   personal-digest-preferences SHA512 SHA384 SHA256
   personal-compress-preferences ZLIB BZIP2 ZIP
   cert-digest-algo SHA256
   ```

Setup Keypair

1. Open a command prompt and run the command `gpg --gen-key`
2. When prompted, select option 1, "RSA and RSA (default)".
3. Use 4096 bits instead of the default.
4. Give an expiration date for the key (2y or 3y is a good choice). This can always be changed later, but it's good if you lose the key at any point in the future. **Take note of when your key expires! You can always change the expiration date later!**
5. Enter your name, enter your email address, but do not enter a comment.
6. Enter a passphrase and wait for the key to be generated. This passphrase is the last line of defense if your private key is compromised, and you will need to enter it when you sign or decrypt messages. **Do not forget this passphrase!**
7. Once your keypair has been generated, take note of the key ID that was printed to the terminal (it will look like `pub 4096R/KEYID`). Next, run the command `gpg -o revcert.asc --gen-revoke {KEYID}`. This will generate a revocation certificate, which allows you revoke this key if it ever gets stolen. Don't give a reason (this is in case anything ever happens). Keep it in a secure place (not, for instance, right in your home directory).
8. If you have more email addresses and names that you plan to use this key under, run the command `gpg --edit-key {KEYID}`. At the prompt that appears, type `adduid`. Again, enter your name and email (and no comment). Do this for each email you want to add. When you are done, select the primary user ID by running the commands `uid {UID NUMBER}` and `primary`. Exit by typing `quit`, and be sure to save your changes.
9. When you are done setting up your key, run the command `gpg --send-key {KEYID}` to submit your public key to the keyserver we set up.

Signing, Verifying, Encrypting, and Decrypting Files

You can now use the `gpg` command to sign, verify, encrypt, and decrypt files:

```
$ gpg -s filename              # Sign a file, makes filename.gpg
$ gpg --clearsign -a filename  # Sign file in same file, makes filename.asc
$ gpg --verify filename.asc    # Verify a signature or clearsigned message
$ gpg -e filename              # Encrypt a file, makes filename.gpg
$ gpg -d filename.gpg          # Decrypt a file and print its contents
```

Additionally, if you navigate to any of the files in Windows Explorer and right-click, you should see options to do all of the above.

Integration with Outlook, Thunderbird, etc.

GPG4Win comes with an Outlook plugin to allow you to use GPG when sending emails. This should be installed automatically if you have Outlook installed. Thunderbird integrates with GPG very easily, but there isn't enough space here to describe how. Either Google it (Thunderbird uses something called Enigmail) or ask one of us to help.

Signing Someone Else's Key

When you sign someone else's key, you are saying that you have verified that the information in the key is correct and that they own the key. To do this, you will need to see photo identification. **Never, never sign a key if you haven't verified all of its information!**

Next, generally after the keysigning party, you do the actually keysigning. You need to make sure that the person owns all the email addresses they say they do. The way you verify this is a little convoluted: for each User ID, you sign it separately, export it, and send them encrypted to the email address you signed. If the key has three User IDs, and the person only controls two of them, only those two emails can be encrypted, and so you will effectively only have signed those two!

Follow these steps:

1. Once you have verified their name, download their public key running `gpg --recv-keys {THEIRKEYID}`
2. Export the original key by running `gpg --armor -o key.orig --export {THEIRKEYID}`
3. For each User ID, do the following exactly:
   (a) Run `gpg --edit-key {THEIRKEYID}`, and at the prompt, type `uid {UIDNUMBERTOSIGN}`.
   (b) Type `sign` to sign the user ID, and then `quit` to exit this prompt.
   (c) Export this key to a different file: `gpg --armor -o key.uid`*n*`signed --export {THEIRKEYID}`.
   (d) Encrypt this file, and send it by email to the email address you just signed.
   (e) Delete the signed key (`gpg --delete-keys {THEIRKEYID}`) and reimport the original (`gpg --import key.orig`). If you don't do this, the next exported key will have more than one User ID signed, which defeats the purpose of this! Repeat for the remaining User IDs.
4. When the owner receives all the signatures, they will upload the new key to a keyserver, which you can then download.

Having Your Own Key Signed

If you are on the receiving end of the above process, be sure you bring a photo ID (or more!) and your key information with you to the keysigning. Be sure to have written copies of your key id, fingerprint, and all your user IDs to give to the keysigner, so they can sign your key after the party.

Following the party, check your email addresses for encrypted messages with the signatures. Decrypt the messages, and import the attachments using `gpg --import filename`. This will apply the signatures to your key. Then, send the newly signed key to a keyserver so others can download it by running the command `gpg --send-keys {KEYID}`.