ECSE 2920 Outline

The first course of action we took onto this project is creating a new Git repository as a place for us to archive all documentations and pictures. Andrew clicked the "+" dropdown menu, then clicked New repository, then named it "Group-5". All group members created individual directory hierarchies using the terminal to store our workspace for all deliverables and documentations. This is done using mkdir [directory_name] to make a new directory, and cd [directory_name] to change the current directory to work on. Because we cannot have the repository owner to only submit progress for this project, the rest of us needed to have remote access to the Group-5 repository so that we can publish updates to any contained file if necessary.

We gain this access by executing this Git command:

```
$ git clone [SSH Address]
```

This way of cloning the Git repository using an SSH key is more secure than HTTPS because the data is encrypted by a key pair; a private key is handled by a group member to sign, and a public key is handled by GitHub, but kept in the group member's Git account. In order to execute the command, however, each group member needs to create an individual private SSH key on the computer.

The group member generates an SSH key by opening the terminal and running this command:

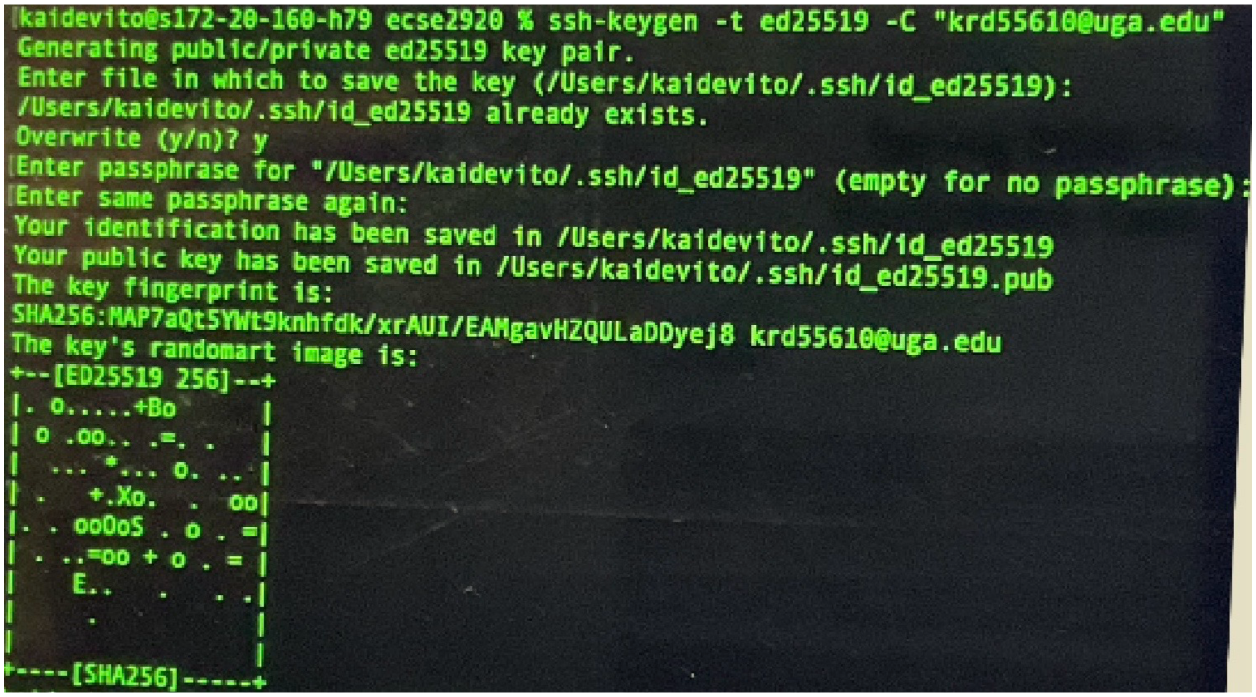
```
$ ssh-keygen -t ed25519 -C "[student_email@uga.edu]"
```

When the terminal displays the following messages, the enter key is pressed to save the key to the default file location:

```
> Generating public/private ALGORITHM key pair.
> Enter a file in which to save the key (/Users/YOU/.ssh/id_ALGORITHM): [Press enter]
```

Including the passphrase for the SSH key is optional.

```
> Enter passphrase (empty for no passphrase): [Type a passphrase]
```



```
kaidevito@s172-20-160-h79 ecse2920 % ssh-keygen -t ed25519 -C "krd55610@uga.edu"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/Users/kaidevito/.ssh/id_ed25519):
/Users/kaidevito/.ssh/id_ed25519 already exists.
Overwrite (y/n)? y
Enter passphrase for "/Users/kaidevito/.ssh/id_ed25519" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /Users/kaidevito/.ssh/id_ed25519
Your public key has been saved in /Users/kaidevito/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:MAP7aQt5YWt9knhfdk/xrAUI/EAMgavHZQULaDDyej8 krd55610@uga.edu
The key's randomart image is:
+--[ED25519 256]--+
|. o.....+Bo      |
| o .oo.. .=. .   |
|  ... *... o. .. |
| .   +.Xo.  .  oo|
|. . ooOoS . o . =|
| . ..=oo + o . = |
|   E..   .    . .|
|    .            |
|                 |
+----[SHA256]-----+
```

Figure 1. The image of the terminal showing the successfully created public SSH key.

Once the public key is saved into the .pub file, a group member copies and displays its contents like so:

$ pbcopy < ~/.ssh/id_ed25519.pub
$ pbpaste

The result from $ pbpaste is then pasted into the textbox under GitHub Profile Picture -> Settings -> SSH and GPG keys -> New SSH key.

All group members had to create a Python file that prints out our name using the print() function to then upload to our repository. Before executing the commands, we reviewed the Git Workflow and its corresponding commands to traverse each member's saves from our computers to the Git repository.

There are four areas where saves are kept: the working repository on the computer, then the staging area for organizing the files for pushing to the Git repository, then the local repository storing a copy of the Git repository, and finally the remote (Git) repository itself.

This is the sequence of Git commands we had to run:

$ git add . - moves saves from working repository to staging area.
$ git commit -m "message" - moves saves from staging area to local repository
$ git push - moves saves from local repository to remote (Git) repository
$ git status - verify current states of working directory and staging area