About SSH Keys

SSH keys provide a more secure way of logging into a virtual private server with SSH than using a password alone. While a password can eventually be cracked with a brute force attack, SSH keys are nearly impossible to decipher by brute force alone. Generating a key pair provides you with two long strings of characters: a public and a private key. You can place the public key on any server, and then unlock it by connecting to it with a client that already has the private key. When the two match up, the system unlocks without the need for a password.

Step One: Create the ed25519 Key Pair

The first step is to create the key pair on your machine:

```
ssh-keygen -t ed25519 -C "your_email@example.com"
```

The preceding command creates an ed25519 type key, which, for various reasons, is preferred over a RSA type key. If you have a legacy system, feel free to use RSA. The email acts as an identifier and is optional.

Step Two: Store the Keys and Passphrase

Once you have entered the ssh-keygen command, you will get a few more questions, such as:

```
Enter file in which to save the key (/home/yourusername/.ssh/id_rsa):
```

The path in parentheses will be the full path of your home directory, followed by ".ssh/id_rsa". You can press *enter* here, saving the file to the default location (in this case /home/yourusername/.ssh/id_rsa) or enter a different file name (not recommended unless you know what you're doing). Then you will be prompted to enter a passphrase:

```
Enter passphrase (empty for no passphrase):
```

It's up to you whether you want to use a passphrase. If you choose to use a passphrase, you'll have an extra level of security at the cost of having to enter the passphrase every time you log in. To use no passphrase, just hit *enter*.
The entire key generation process looks like this:

```
ssh-keygen -t ed25519 -C "your_email@example.com"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/yourusername/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/yourusername/.ssh/id_ed25519
Your public key has been saved in /home/yourusername/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:h2/JA/EqOonH/d2TogyVKehWQTUtCKmKsNu3VylFnh0 your_email@example.com
```

```
The key's randomart image is:
+--[ED25519 256]--+
|    .o.oo.       |
|    ... o.E      |
|   .  .o.+ .     |
|. .  . .+*.      |
|oo  . o.S.o      |
|+  . ..oo* .     |
| o ooooo. *  .   |
|. o.*.o+ o.oo    |
|   oo+ .+.....   |
+----[SHA256]-----+
```

Step Three: Copy the Public Key

Once the key pair is generated, it is time to place the public key on the server that you want to use. You can copy the public key into the new machine's authorized_keys file with the ssh-copy-id command if you are using MacOS/Linux, or using scp on Windows. In this example, we will use eniac however adjust to fit your needs.

If ssh-copy-id is on your system:

```
ssh-copy-id username@eniac.cs.hunter.cuny.edu
```

Alternatively, manual copy:

```
cat .ssh/id_ed25519.pub | ssh username@eniac.cs.hunter.cuny.edu 'cat >> .ssh/authorized_keys'
```

No matter which command you chose, you should see something like:

```
The authenticity of host 'eniac.cs.hunter.cuny.edu (146.95.214.71)' can't be established.
ED25519 key fingerprint is b1:2d:33:67:ce:35:4d:5f:f3:a8:cd:c0:c4:48:86:12.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '146.95.214.71' (RSA) to the list of known hosts.
```

If you are still prompted for a password after going through all the steps, edit file permissions on the remote server by typing the following command:

```
ssh username@eniac.cs.hunter.cuny.edu "chmod 700 .ssh; chmod 640 .ssh/authorized_keys"
```

Now you can go ahead and log into the server and you will not be prompted for a password.

Step Four (optional): Make your own config

Lastly, we can reduce the amount of typing needed to log in by setting up our own ssh config. This will require a text editor to set up, in this example we will be using nano.

On your host machine, type in *nano ~/.ssh/config*, and inside this file enter the following:

```
Host eniac
    HostName eniac.cs.hunter.cuny.edu
    User username
    IdentityFile ~/.ssh/id_ed25519
```

On the second line, replace Username with your given account name.

Once you are done, hit Control + O to save your changes, then Control + X to close nano.
Additionally if jump from eniac to another machine(proxy jump), you can add the following to the file.

```
Host cslab10
    User username
    ProxyJump eniac
```

In this case we used cslab10, but replace that with the name of the machine you are using. If you'd like the name and HostName to be different, specify the HostName parameter.

Now instead of typing *ssh username@eniac.cs.hunter.cuny.edu* everytime you want to connect, you can simply type *ssh eniac* and it will connect you without the need for a password.

If you have set up the ProxyJump, you can also now just directly connect to that without the need to connect to eniac.