Git Practical

What is SSH?

- SSH, also known as Secure Shell or Secure Socket Shell, is a network protocol that gives users, particularly system administrators, a secure way to access a computer over an unsecured network. It provides a safe and secure way of executing commands, making changes, and configuring services remotely.

- SSH uses the client-server model, connecting a Secure Shell client application, which is the end where the session is displayed, with an SSH server, which is the end where the session runs. SSH implementations often include support for application protocols used for terminal emulation or file transfers.

- SSH can also be used to create secure tunnels for other application protocols, for example, to securely run X Window System graphical sessions remotely. An SSH server, by default, listens on the standard Transmission Control Protocol (TCP) port 22.

How SSH Works?

- When you connect through SSH, you will be dropped into a shell session, which is a text-based interface where you can interact with your server. For the duration of your SSH session, any commands that you type into your local terminal are sent through an encrypted SSH tunnel and executed on your server.

- The SSH connection is implemented using a client-server model. This means that for an SSH connection to be established, the remote machine must be running a piece of software called an SSH daemon.

- This software listens for connections on a specific network port, authenticates connection requests, and spawns the appropriate environment if the user provides the correct credentials.

- The user's computer must have an SSH client. This is a piece of software that knows how to communicate using the SSH protocol and can be given information about the remote host to connect to, the username to use, and the credentials that should be passed to authenticate.

How SSH authenticates users?

- Clients generally authenticate either using passwords (less secure and not recommended) or SSH keys, which are very secure. Password logins are encrypted and are easy to understand for new users.

- SSH keys are a matching set of cryptographic keys which can be used for authentication. Each set contains a public and a private key. The public key can be shared freely without concern, while the private key must be vigilantly guarded and never exposed to anyone.

- To authenticate using SSH keys, a user must have an SSH key pair on their local computer. On the remote server, the public key must be copied to a file within the user's home directory at ~/.ssh/authorized_keys. This file contains a list of public keys, one-per-line, that are authorized to log into this account.

- When a client connects to the host, wishing to use SSH key authentication, it will inform the server of this intent and will tell the server which public key to use. The server then checks its authorized_keys file for the public key, generates a random string, and encrypts it using the public key. This encrypted message can only be decrypted with the associated private key. The server will send this encrypted message to the client to test whether they actually have the associated private key.

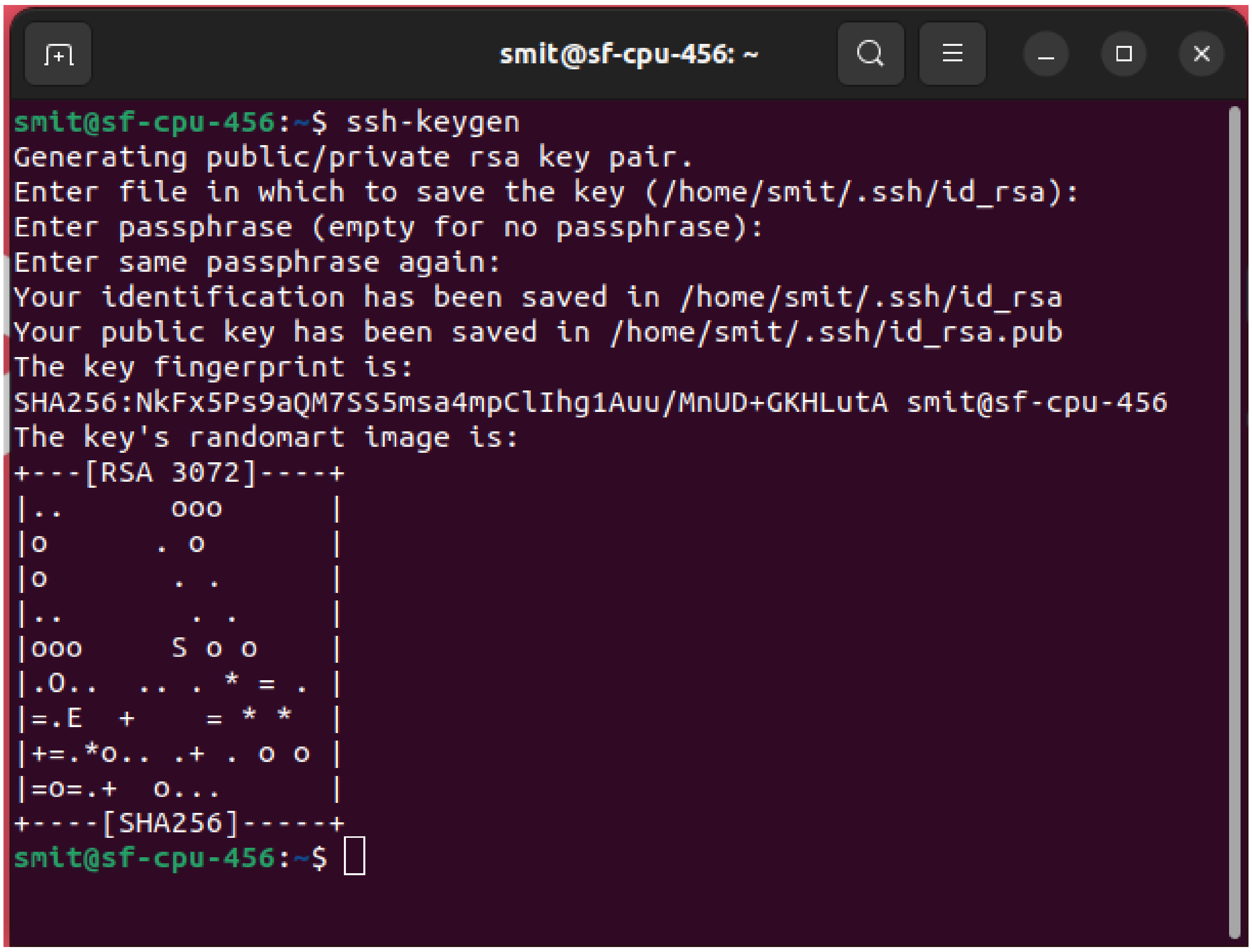
- Upon receipt of this message, the client will decrypt it using the private key and combine the random string that is revealed with a previously negotiated session ID. It then generates an MD5 hash of this value and transmits it back to the server. The server already had the original message and the session ID, so it can compare an MD5 hash generated by those values and determine that the client must have the private key.

Generating SSH key Pairs

- Generating a new SSH public and private key pair on your local computer is the first step towards authenticating with a remote server without a password. Unless there is a good reason not to, you should always authenticate using SSH keys.

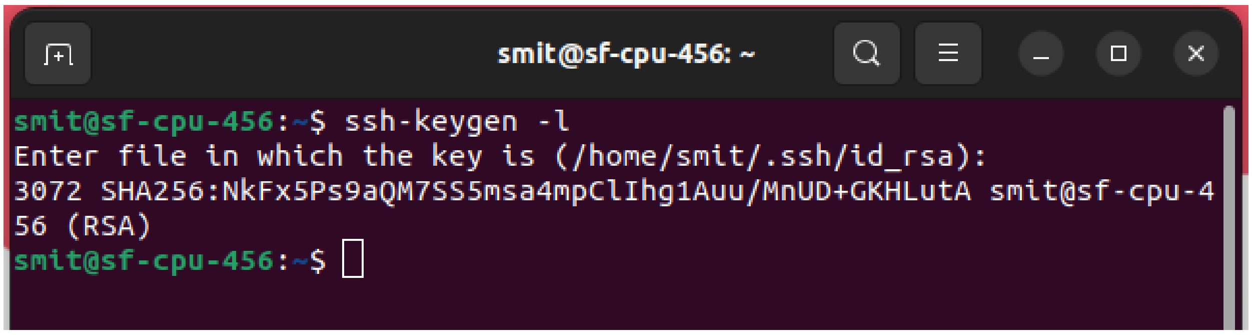
- A number of cryptographic algorithms can be used to generate SSH keys, including RSA, DSA, and ECDSA. RSA keys are generally preferred and are the default key type.

- Command to generate an RSA key pair on local computer is “ssh-keygen”.



```
smit@sf-cpu-456:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/smit/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/smit/.ssh/id_rsa
Your public key has been saved in /home/smit/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:NkFx5Ps9aQM7SS5msa4mpClIhg1Auu/MnUD+GKHLutA smit@sf-cpu-456
The key's randomart image is:
+---[RSA 3072]----+
|..      ooo      |
|o      . o       |
|o       . .      |
|..       . .     |
|ooo     S o o    |
|.O..  .. . * = . |
|=.E  +    = * *  |
|+=.*o.. .+ . o o |
|=o=.+  o...      |
+----[SHA256]-----+
smit@sf-cpu-456:~$
```

- Each SSH key pair share a single cryptographic “fingerprint” which can be used to uniquely identify the keys. This can be useful in a variety of situations. We can get the fingerprint for an ssh key using command “ssh-keygen -l”.



```
smit@sf-cpu-456:~$ ssh-keygen -l
Enter file in which the key is (/home/smit/.ssh/id_rsa):
3072 SHA256:NkFx5Ps9aQM7SS5msa4mpClIhg1Auu/MnUD+GKHLutA smit@sf-cpu-4
56 (RSA)
smit@sf-cpu-456:~$
```