What is public key authentication?

As general computer users, we all know what is password authentication. It is where we just require a password (usually 8–20 characters long) to login to a computer. But, with the computational power of computers being exponentially increasing and with the number of attacks on servers being vastly increasing, can a password of length 8–20 characters prevent your server from being breached through a SSH login?

SSH (Secure Shell) is the method we use to login to a remote servers and do whatever the task we want through the server's terminal. If you keep using password authentication and the common users and usernames for login (like **'root'**) there is a possibility that your server can be breached into if the used password is not lengthy enough and strong enough. For example, if you have a VPS, and using any common user name like **root** to login, check your logs located at /var/log/secure. Most probably, you will see something like this in it. It doesn't mean that you have been hacked, but it means that there have been attempts to breach into the system. Therefore, it is better to take measures to make your system more secure.

```
Apr 10 06:39:27 echo sshd[22297]: reverse mapping checking getaddrinfo
for 222-237-78-139.tongkni.co.kr failed - POSSIBLE BREAK-IN ATTEMPT!
Apr 10 13:39:27 echo sshd[22298]: Received disconnect from
222.237.78.139: 11: Bye Bye
Apr 10 06:39:31 echo sshd[22324]: Invalid user edu1 from 222.237.78.139
Apr 10 06:39:31 echo sshd[22324]: reverse mapping checking getaddrinfo
for 222-237-78-139.tongkni.co.kr failed - POSSIBLE BREAK-IN ATTEMPT!
Apr 10 13:39:31 echo sshd[22330]: input_userauth_request: invalid user
edu1
Apr 10 13:39:31 echo sshd[22330]: Received disconnect from
222.237.78.139: 11: Bye Bye
Apr 10 06:39:35 echo sshd[22336]: Invalid user test1 from 222.237.78.139
Apr 10 06:39:35 echo sshd[22336]: reverse mapping checking getaddrinfo
for 222-237-78-139.tongkni.co.kr failed - POSSIBLE BREAK-IN ATTEMPT!
Apr 10 13:39:35 echo sshd[22338]: input_userauth_request: invalid user
test1
Apr 10 13:39:35 echo sshd[22338]: Received disconnect from
222.237.78.139: 11: Bye Bye
Apr 10 06:39:39 echo sshd[22377]: Invalid user test from 222.237.78.139
Apr 10 06:39:39 echo sshd[22377]: reverse mapping checking getaddrinfo
for 222-237-78-139.tongkni.co.kr failed - POSSIBLE BREAK-IN ATTEMPT!
Apr 10 13:39:39 echo sshd[22378]: input_userauth_request: invalid user
test
Apr 10 13:39:39 echo sshd[22378]: Received disconnect from
222.237.78.139: 11: Bye Bye
```

From this Ubuntu Community Post,

```
With public key authentication, the authenticating entity has a public
key and a private key. Each key is a large number (1024,2048 or 4096
bits long) with special mathematical properties. The private key is kept
on the computer you log in from, while the public key is stored on the
.ssh/authorized_keys file on all the computers you want to log in to.
When you log in to a computer, the SSH server uses the public key to
"lock" messages in a way that can only be "unlocked" by your private key
-- this means that even the most resourceful attacker can't snoop on, or
interfere with, your session (because predicting the private key is
impossible under currently available computational power due to its
length) . As an extra security measure, most SSH programs store the
private key in a pass-phrase-protected format, so that if your computer
is stolen or broken in to, you should have enough time to disable your
old public key before they break the pass-phrase and start using your
key.
```

Simply, you generate a key pair (a private key and a public key) where any text encrypted using the private key can only be decrypted by the corresponding public key and vice versa. In order to make your SSH login more secure, using public key authentication is recommended over password authentication.

How to enable public key authentication?

Step 1 : Generating Key Pair

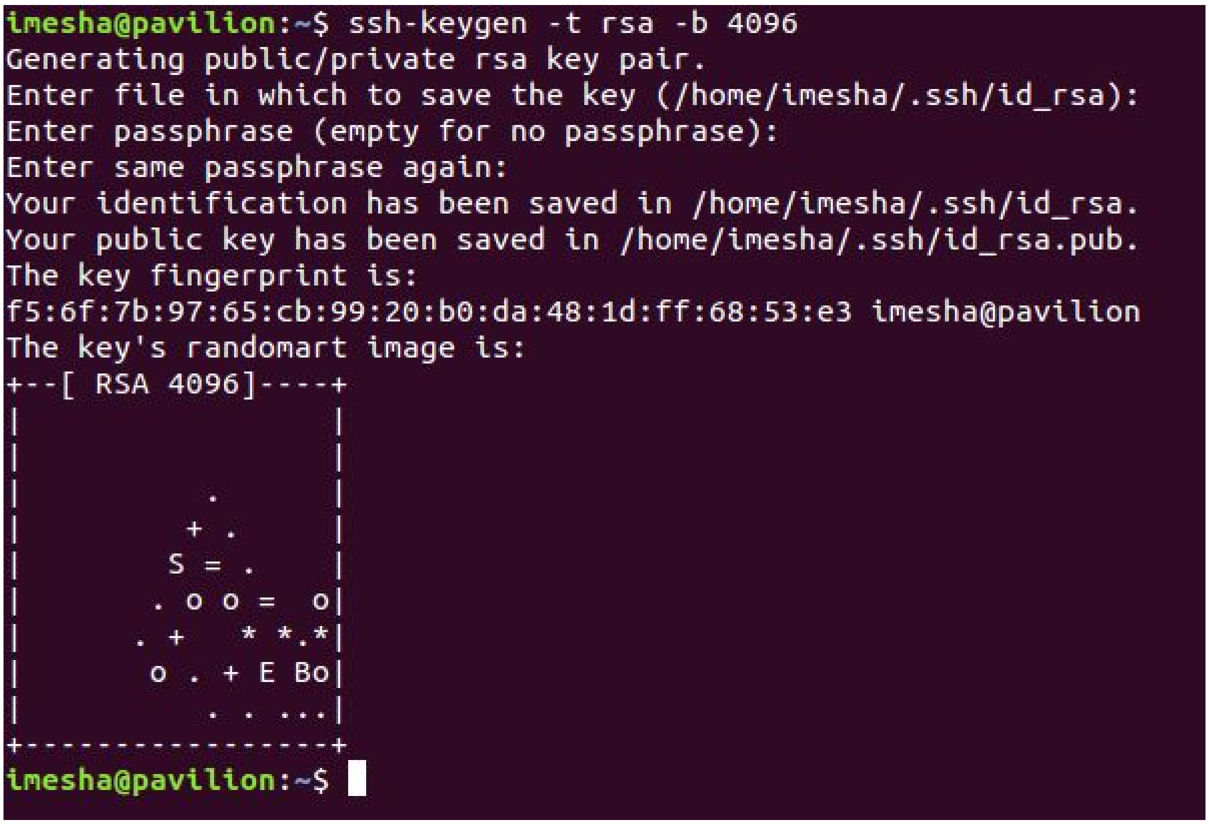
As mentioned above, SSH keys come in pairs (private key and public key). Usually the keys are stored in the ***~/.ssh*** directory. You should make sure that directory exists and its permissions are 7***00 (To prevent other users from accessing your keys)***.
If the directory is not there, create it.

```
mkdir ~/.ssh
chmod 700 ~/.ssh
```

Then, you have to generate the key pair.

```
ssh-keygen -t rsa -b 4096
```

Here, **-t rsa** means that we are creating a RSA key pair. **-b 4096** means that we are creating a key pair of length 4096 bits. Other common key lengths are 1024 and 2048. But, for increased security, using 4096 bit length key pair is better. You will be asked several questions like:



```
imesha@pavilion:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/imesha/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/imesha/.ssh/id_rsa.
Your public key has been saved in /home/imesha/.ssh/id_rsa.pub.
The key fingerprint is:
f5:6f:7b:97:65:cb:99:20:b0:da:48:1d:ff:68:53:e3 imesha@pavilion
The key's randomart image is:
+--[ RSA 4096]----+
|                 |
|                 |
|          .      |
|         + .     |
|        S = .    |
|       . o o =  o|
|      . +   * *.*|
|       o . + E Bo|
|          . . ...|
+-----------------+
imesha@pavilion:~$
```

When asked for the location to which the key pair to be saved, press **enter** to keep the default **~/.ssh/id_rsa**. Else specify the file. Then give a password to make sure the locally stored private key cannot be easily stolen and used. Now you will see 2 files in your **~/.ssh** directory, **id_rsa** which contains the private key and the **id_rsa.pub** which includes the public key. (Note that if you specify another path other than the default file path to store the key pair, you will find these files in that directory. Now our first step is done.

Step 2 : Uploading the public key to the remote server

Then login to the remote server using SSH. You have to use password authentication as usual for this step in order to login. Then, create the **~/.ssh** directory if it is not there.

```
cd ~
mkdir .ssh
```

Then, you can use **secure copy (SCP)** to copy the public key to the remote server as follows. Remember, this command should be run from your local computer. Not from the terminal in which we logged into the remote server.

```
scp ~/.ssh/id_rsa.pub user@example.com:~/.ssh/uploaded_key.pub
```

The **scp** command take two arguments here. First one is the location of the public key file in your local computer. The second argument include the remote user name, remote server's ip/address, location to which the public key should be uploaded.

```
scp <local location of the public key> <remote user>@<remote server
address>:<destination file>
```

Then, we have to move the uploaded public key to the **authorized_keys** file in the **.ssh** directory. To move that, switch to the terminal in which we logged in to the remote server using SSH. Then go to the **~/.ssh** directory. You will see the **uploaded_key.pub** has been copied to this location. Then, use the following command to put the public key to **authorized_keys** file.

```
cat uploaded_key.pub >> authorized_keys
```

One more thing! You have to check the **/etc/ssh/sshd_config** file for the following lines.

```
RSAAuthentication yes
PubkeyAuthentication yes
AuthorizedKeysFile .ssh/authorized_keys
```

If those lines are commented with **#**, remove those. Then save the file. Now you have to restart the ssh service.

```
sudo service ssh restart
sudo service sshd restart (on CentOS)
```

Now we are good to go ….

you can now login to the remote server with SSH using public key authentication using the following command,

```
ssh -i <path to your private key> <user>@<remote server address>
Ex : ssh -i ~/.ssh/id_rsa root@192.168.56.166
```

Now you are logging in using public key authentication. This doesn't mean your server is secured. That is because you are still using password authentication also.

Step 3 : Disable password authentication

Go to **/etc/ssh/sshd_config** file and do the following change.

```
PasswordAuthentication no
```

Just set the password authentication to no. Then restart the ssh service again. Now you cannot login using the password. Just using the public key authentication.

Now your VPS is more secured !

Further Improvements

You can change the port from which the SSH server is listening to incoming SSH connections by changing the following line to your preferred port.

```
#Port 22
```

Usually it is 22. That is why you usually get this much attack attempts since you are using the default port. You can change this and prevent attacks coming from default ports. Still an attacker can determine your port by port scanning, but this will drastically reduce the automated bot attacks coming to your VPS.

Then you can disable the **root** login also by changing the following line to **no** in the same file.

```
#PermitRootLogin yes
```

This will allow no one to login as root. Therefore, you can login as a general user and switch to the super user mode later in order to be the **root**. This will be an advantage since the attackers have to now guess your username apart from the difficulty of finding your private key.