Cryptography Application : SSH

PacNOG19

28th November - 2nd December 2016

Nadi, Fiji

Issue Date: [31-12-2015]

Revision: [v.1]

APNIC

What is “Safely”

- **Authentication** – I am Assured of Which Host I am Talking With
- **Authentication** - The Host Knows Who I Am
- The Traffic is **Encrypted**

Traditional (Telnet)

Servers

clear text

Terminal

clear text

Routers

Encrypted (SSH)

Servers

encrypted text

Terminal

encrypted text

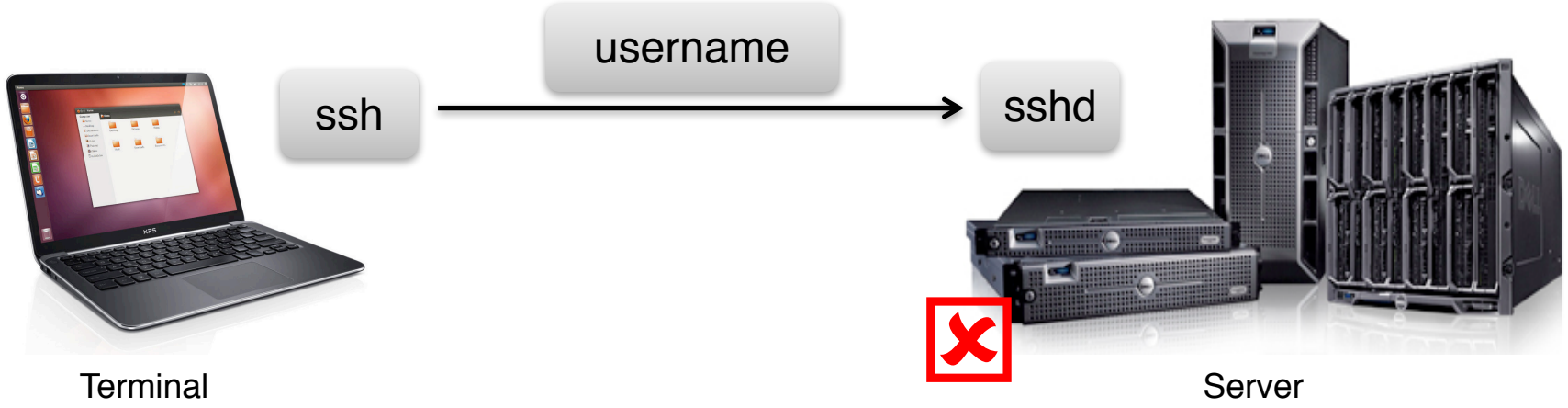
Routers

Secure Shell (SSH)

- Provides authenticated and encrypted shell access to a remote host
- It's not only a secure shell; it is much more
  - Transport protocol (eg. SCP, SFTP, SVN)
  - Connection forwarder. You can use it to build custom tunnels

SSH (Ordinary Password Authentication)

1. The user makes an initial TCP connection and sends a username.

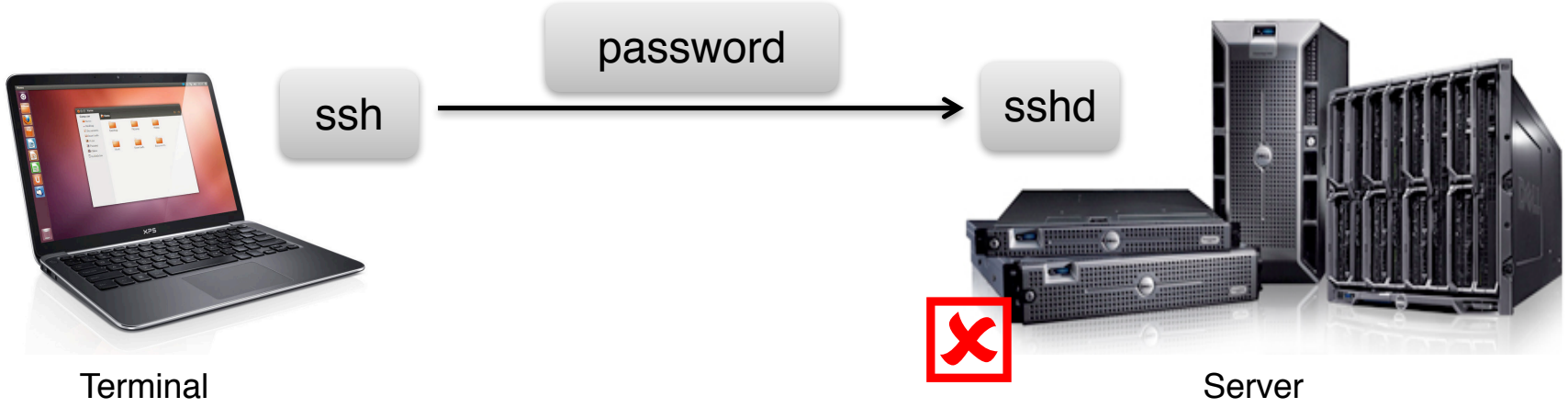


SSH (Ordinary Password Authentication)

2. The ssh daemon on the server responds with a demand for a password, and access to the system has not yet been granted in any way.

SSH (Ordinary Password Authentication)

3. The ssh client prompts the user for a password, which is relayed through the encrypted connection to the server where it is compared against the local user base.



SSH (Ordinary Password Authentication)

4. If the user's password matches the local credential, access to the system is granted and a two-way communications path is established, usually to a login shell.

Password Authentication

- Password Authentication is that it's simple to set up - usually the default - and is easy to understand.
- Allows brute-force password guessing.
- Passwords must be remembered and entered separately upon every login.

Public Key Access

- User creates a pair of public and private keys.
- The **public key** - nonsensitive information.
- The **private key** - is protected on the local machine by a strong passphrase.
- Installs the public key in his **`$HOME/.ssh/authorized_keys`** file on the target server.
- This key must be installed on the target system - one time.

Public Key Access

1. The user makes an initial connection and sends a username along with a request to use a key.

2. The ssh daemon on the server looks in the user's authorized_keys file, constructs a challenge based on the public key found there, and sends this challenge back to the user's ssh client.

3. The ssh client receives the key challenge. It finds the user's private key on the local system, but it's protected by an encrypting passphrase.

4. The user is prompted for the passphrase to unlock the private key.

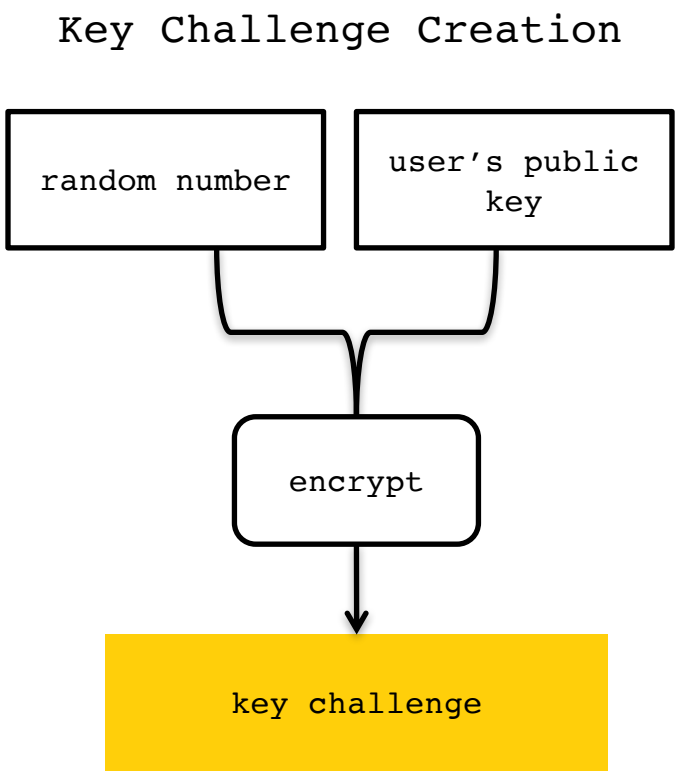
5. ssh uses the private key to construct a key response, and sends it to the waiting sshd on the other end of the connection. **It does not send the private key itself!**

6. sshd validates the key response, and if valid, grants access to the system.

How key challenge work (Under the hood)

1. User ssh to server, he presents his username to the server with a request to set up a key session.

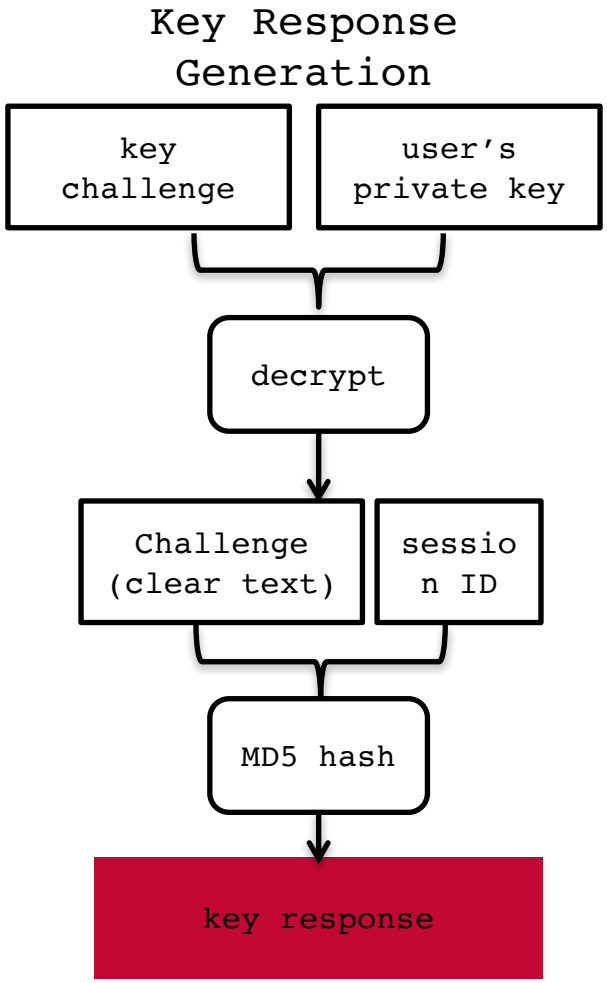
2. The server creates a "challenge". It creates and remembers a large random number, then encrypts it with the user's public key.



How key challenge work (Under the hood)

3. Agent decrypts it with the private key and get the random number generated by the server.

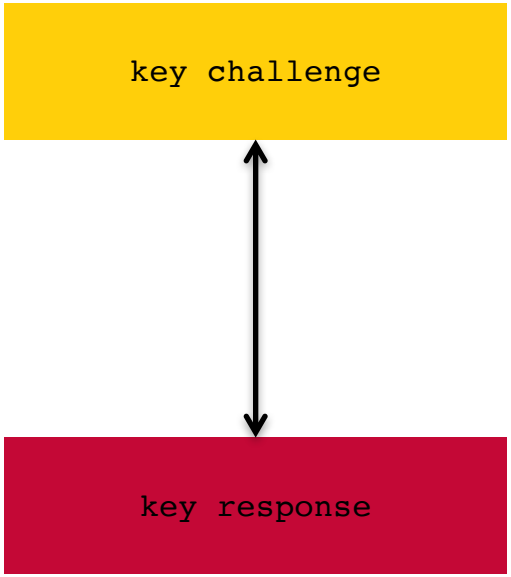
4. The agent takes this random number, appends the previously negotiated SSH session ID and creates an MD5 hash value of the resultant string: this result is sent back to the server as the key response.



How key challenge work (Under the hood)

5. The server computes the same MD5 hash (random number + session ID) and compares it with the key response from the agent.

6. If they match, the user must have been in possession of the private key, and access is granted.



Public Key Access

- Public keys cannot be easily brute-forced.
- The same private key (with passphrase) can be used to access multiple systems: no need to remember many passwords.
- Requires one-time setup of public key on target system.
- Requires unlocking private key with secret passphrase upon each connection.

Public Key Access

- Never store Private Key on a multi-user host.
- Store Private Key ONLY on your laptop and protect your laptop (Encrypt Disk!).
- It is OK to use SSH_AGENT to remember your key ONLY if your laptop/computer locks very quickly.

Private Key on Unix / MacOSX

- SSH is Built In
  - UNIX
  - Linux
  - MacOS X

Generate Key (Unix / MacOSX)

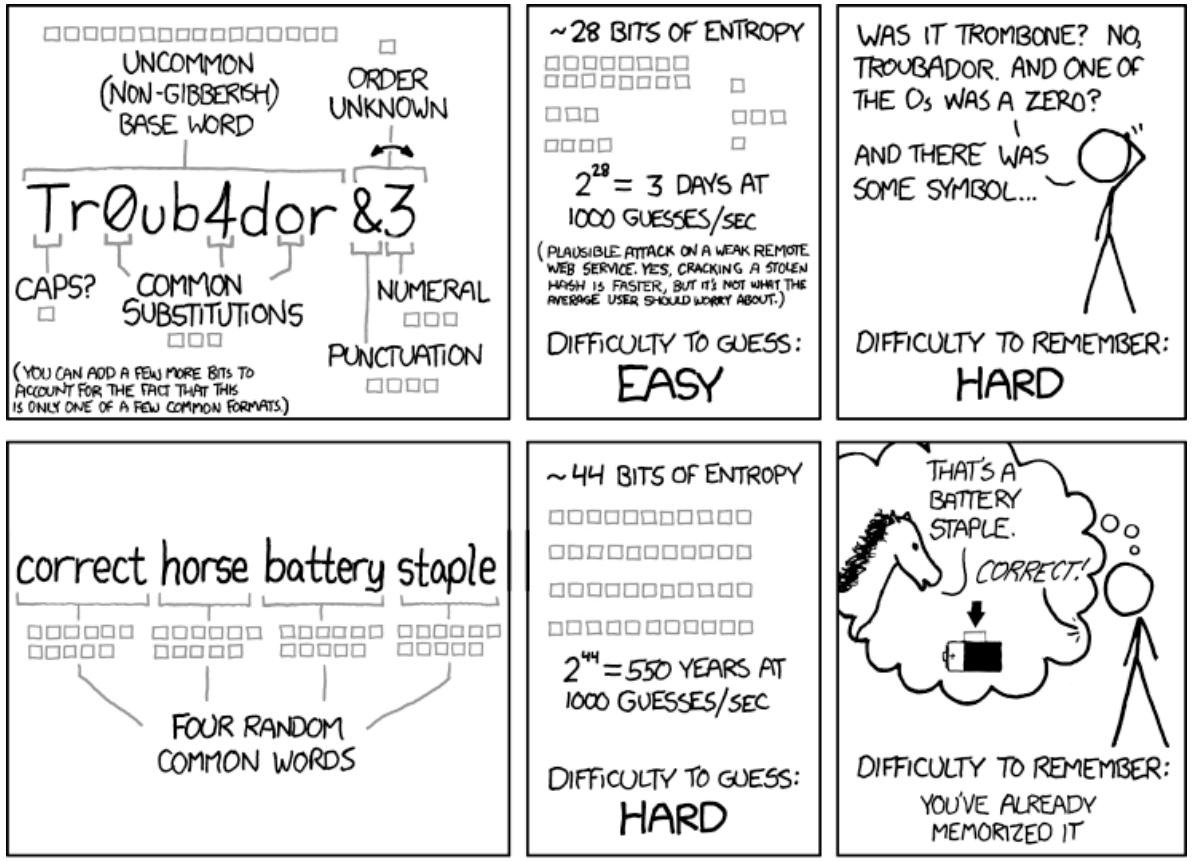
```
$/usr/home/foo> ssh-keygen -t rsa -b 4096 -C your_email@example.com
Generating public/private rsa key pair.
Enter file in which to save the key (/usr/home/foo/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /usr/home/foo/.ssh/id_rsa.
Your public key has been saved in /usr/home/foo/.ssh/id_rsa.pub.
The key fingerprint is:
27:99:35:e4:ab:9b:d8:50:6a:8b:27:08:2f:44:d4:20 foo@bdnog.org
```

Generate Key (Unix / MacOSX)

~/.ssh/id_rsa: The private key. DO NOT SHARE THIS FILE!

~/.ssh/id_rsa.pub: The associated public key. This can be shared freely without consequence.

Password vs Passphrase



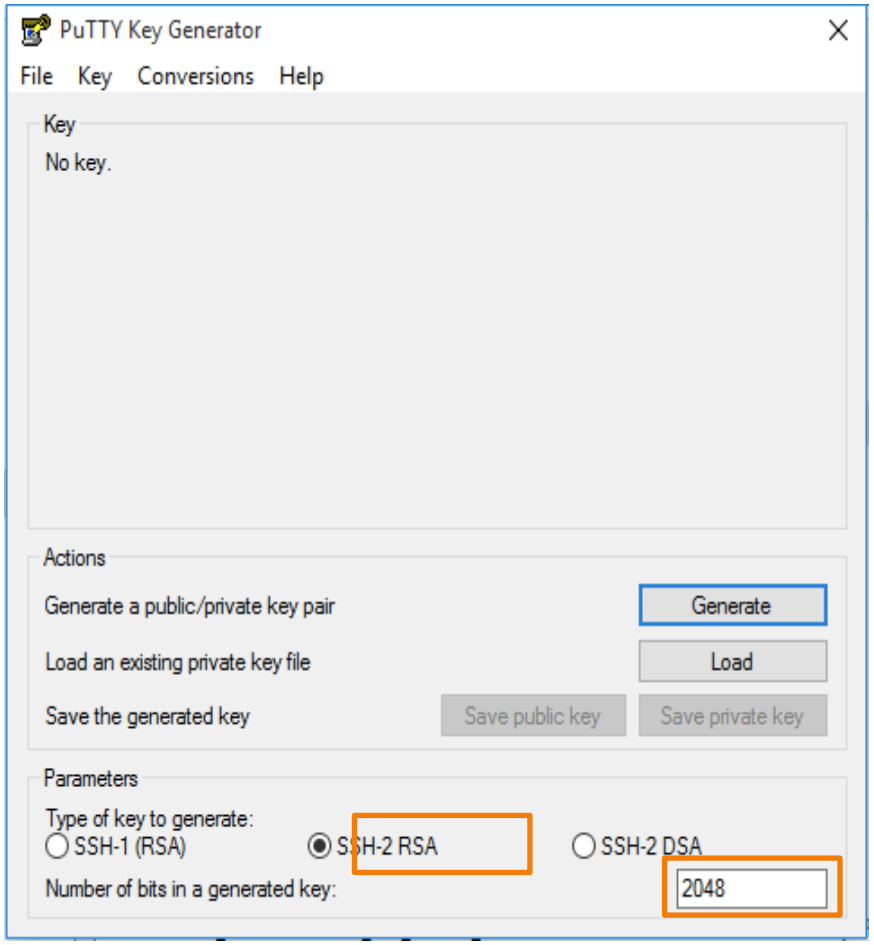
source : http://xkcd.com/936/

Private Key on Windows

- http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html
  - PuTTY (the Telnet and SSH client itself)
  - PuTTYgen (an RSA and DSA key generation utility).
  - Pageant (an SSH authentication agent for PuTTY, PSCP, PSFTP, and Plink)

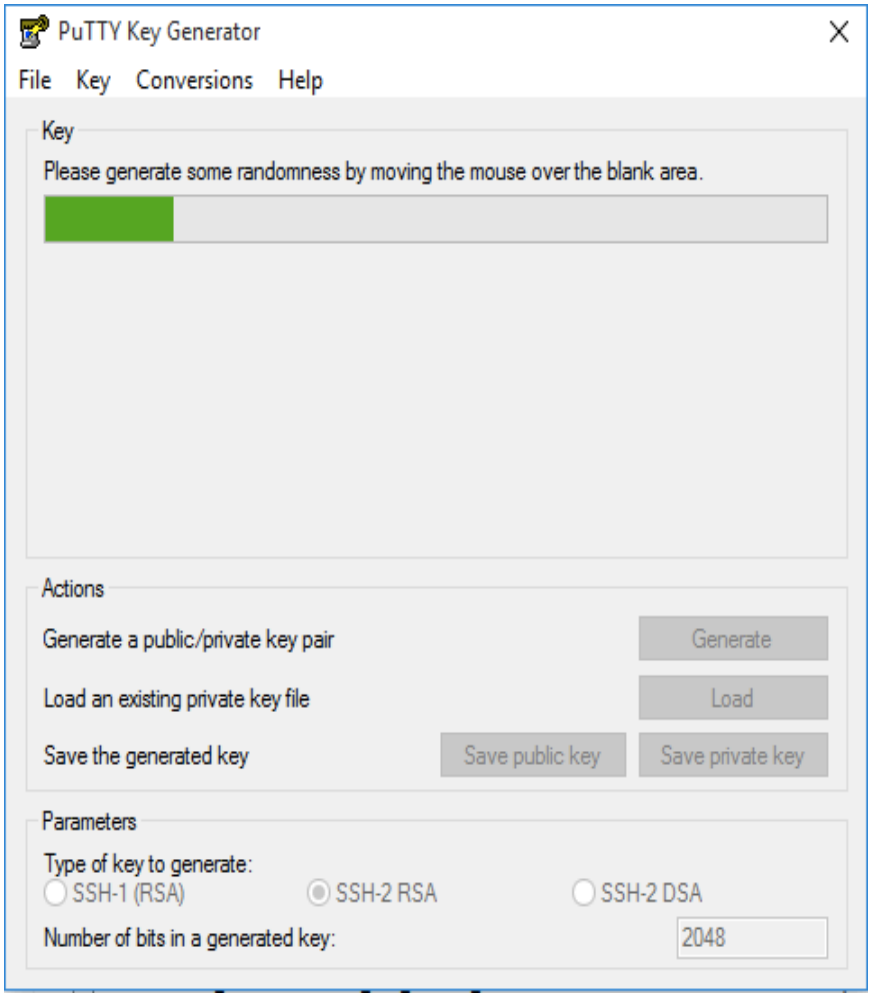
Generate Key (Windows)

1. Run PuttyGen



Generate Key (Windows)

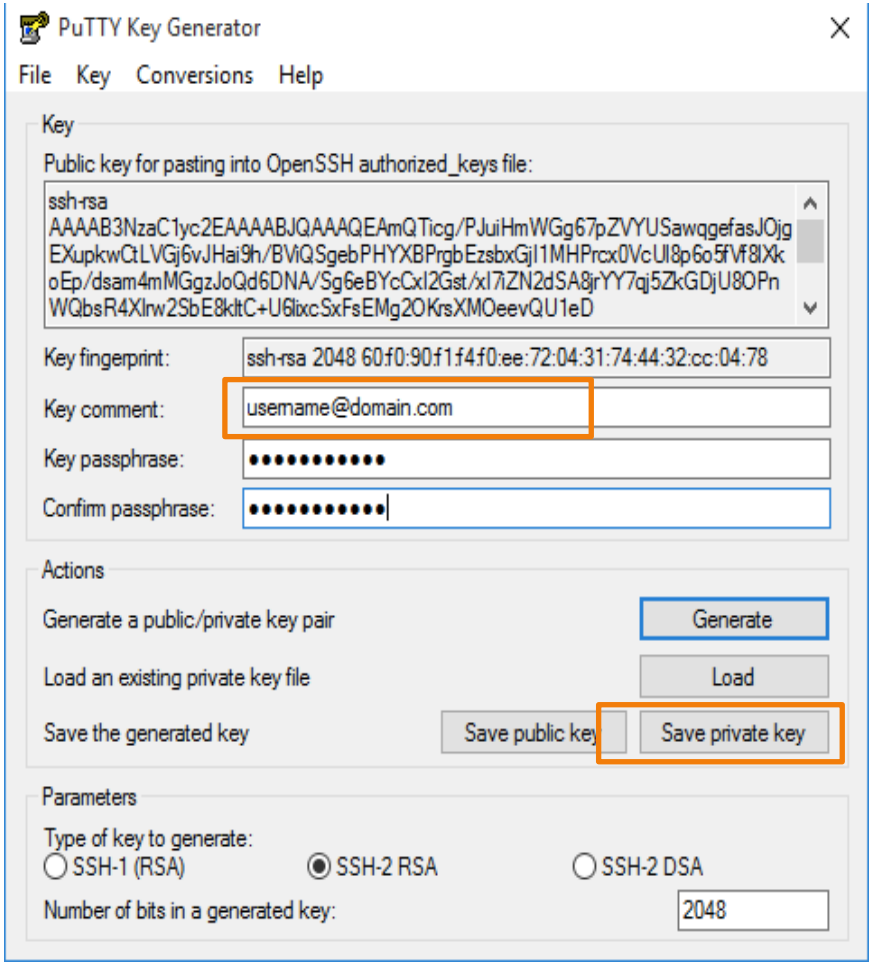
2. Generate Key



Generate Key (Windows)

3. Enter Passphrase & Save Private Key

4. Right-click in the text field labeled Public key for pasting into OpenSSH authorized_keys file and choose Select All and copy the key



Putting the Key on the Target Host

- You can copy the public key into the new machine's authorized_keys file with the ssh-copy-id command

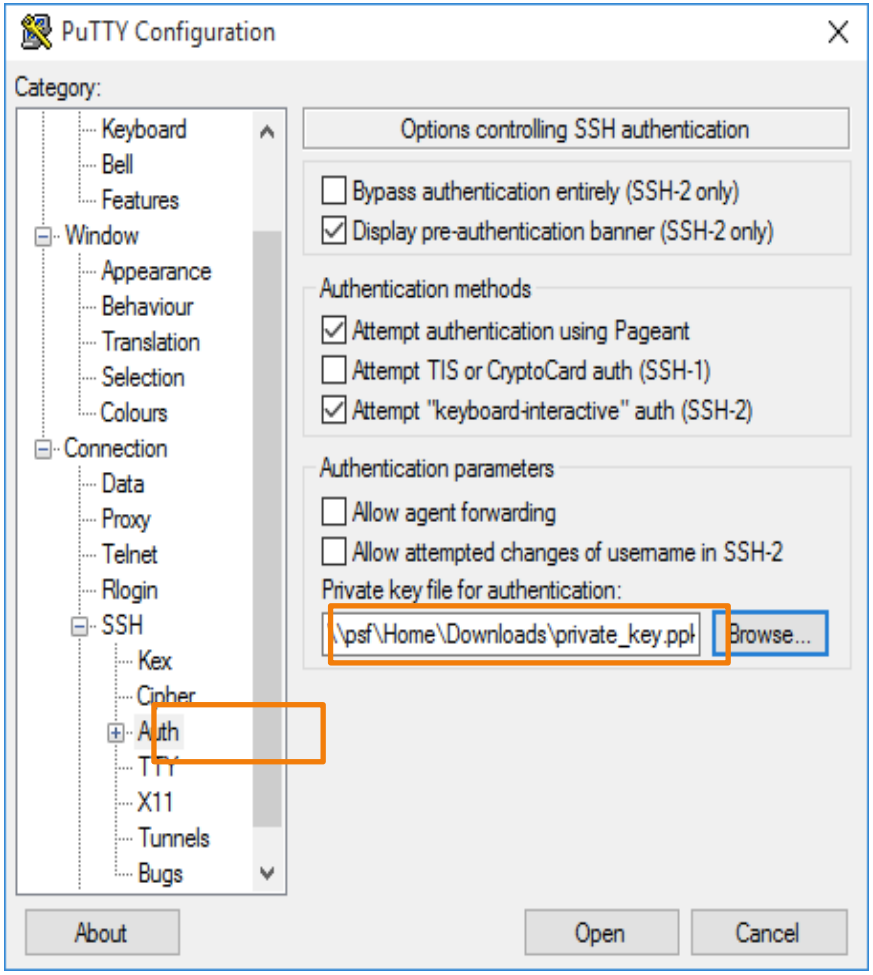
```
ssh-copy-id user@serverip
```

- Alternatively, you can paste in the keys using SSH:

```
cat ~/.ssh/id_rsa.pub | ssh user@serverip "mkdir -p
~/.ssh && cat >> ~/.ssh/authorized_keys”
```

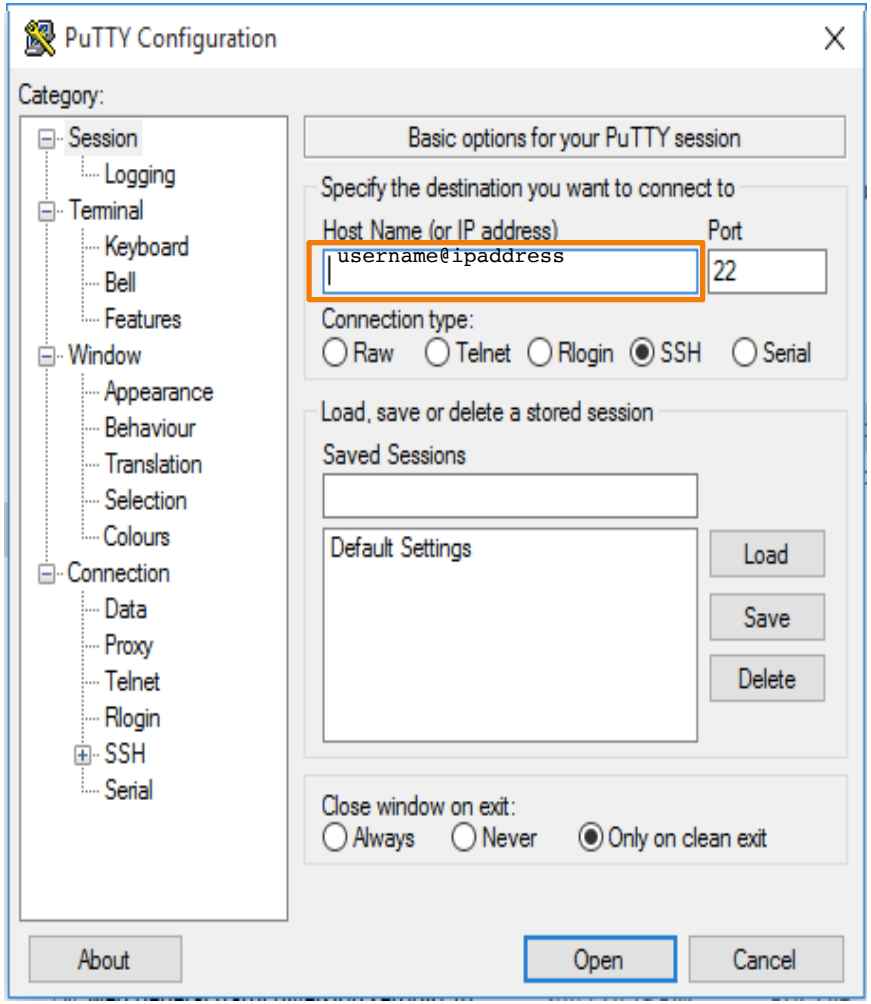
Generate Key (Windows)

4. Load Key in Putty



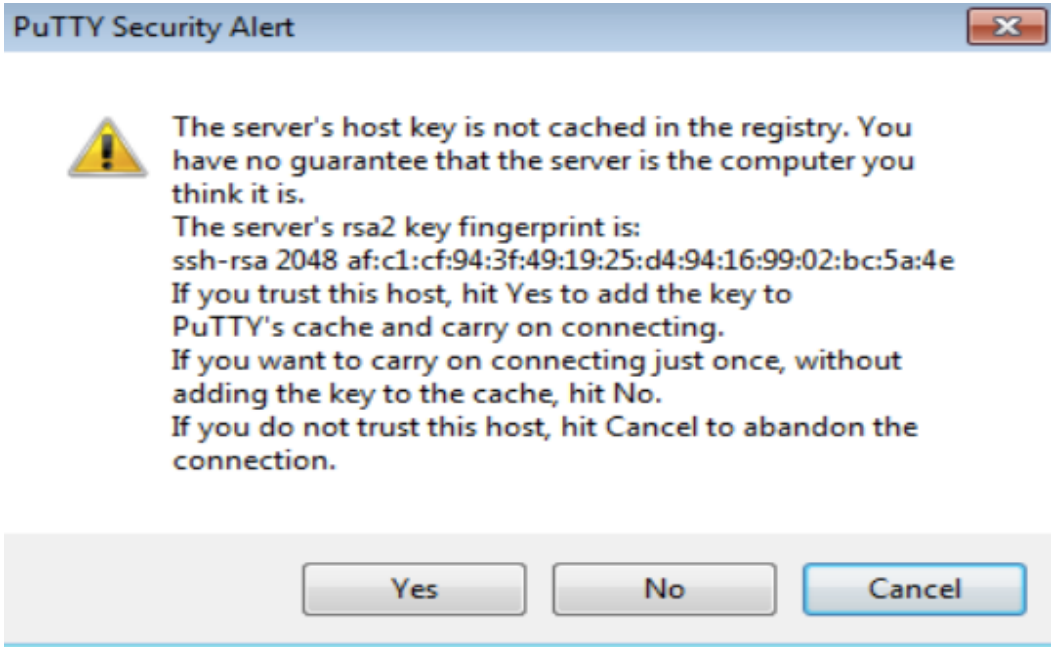
Generate Key (Windows)

5. SSH to host



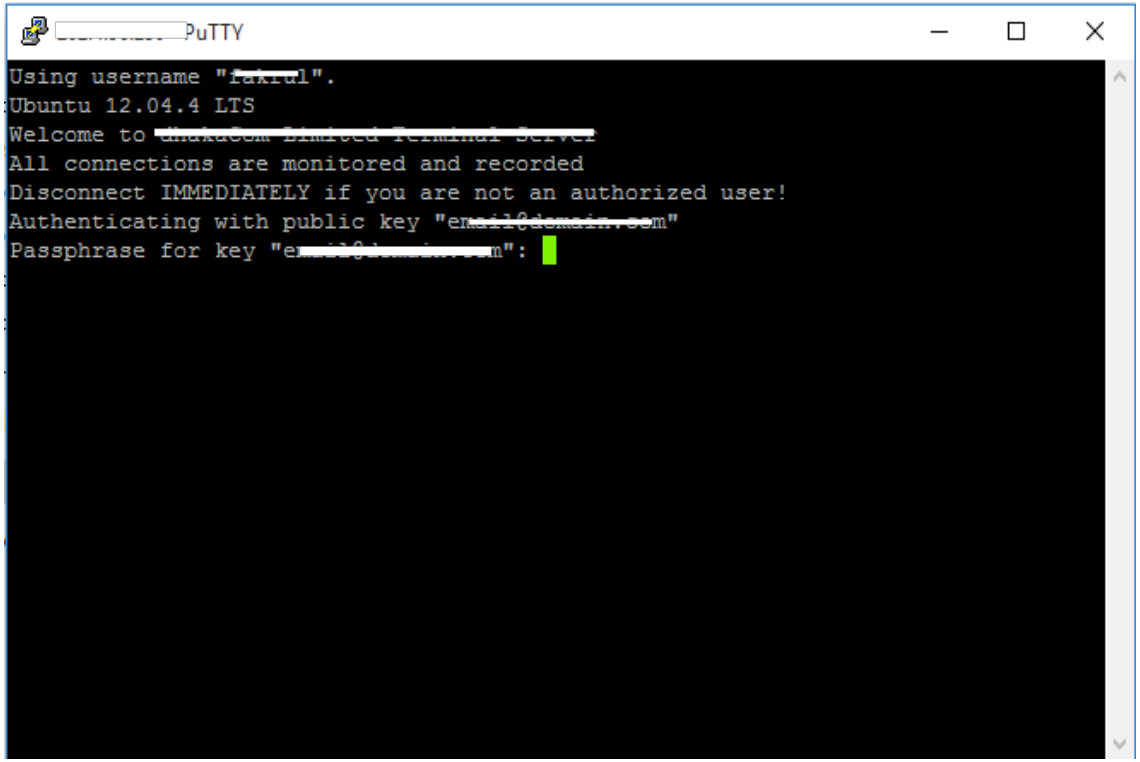
Generate Key (Windows)

6. Accept Host's Key



Generate Key (Windows)

7. passphrase for Key



PuTTY Agent: Pageant

- Select Add Key, browse to your key, select, enter passphrase
- Enter passphrase again. Eventually you'll get it right.
- SSH to your server
- PuTTY enable/disable agent: `Connection -> SSH -> Auth, "Attempt Authentication using Pageant"` checkbox

Exercise

- Create your key
- Follow the lab manual ssh-lab.pdf