

# Cryptography - SSH

**bdNOG7**

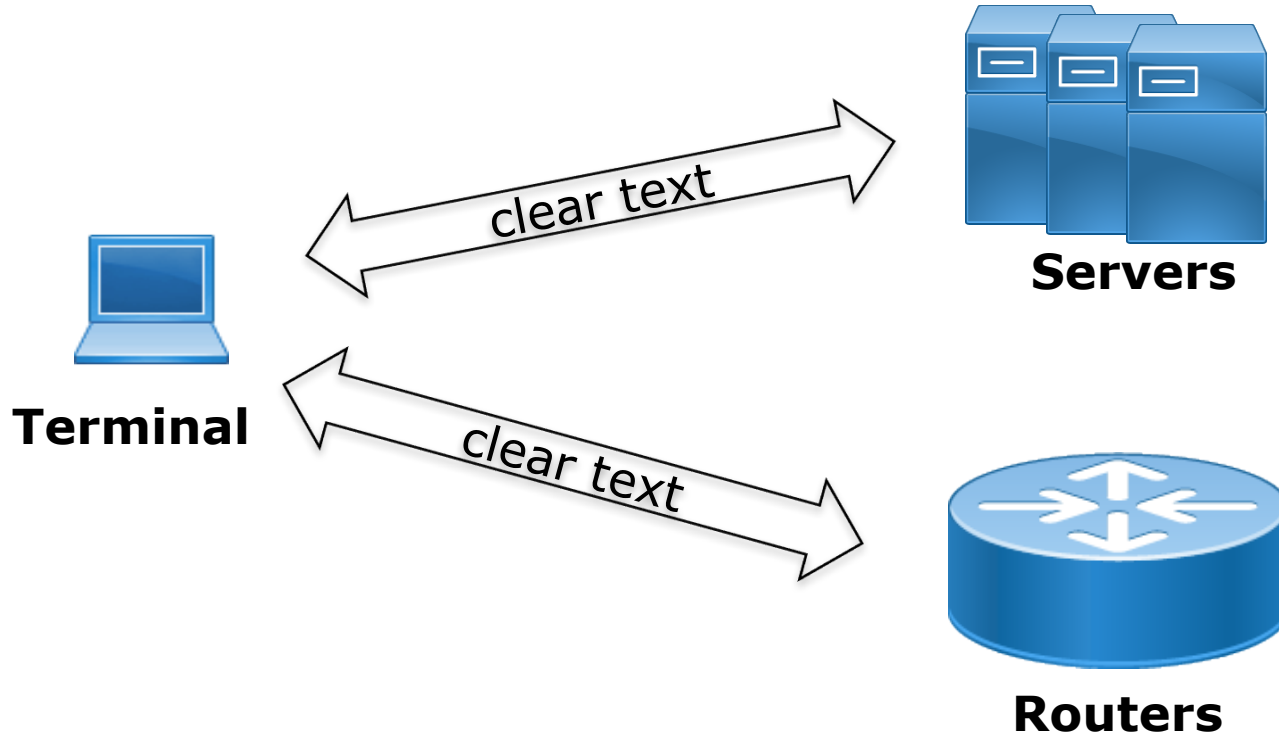
19-22 November 2017

Dhaka, Bangladesh

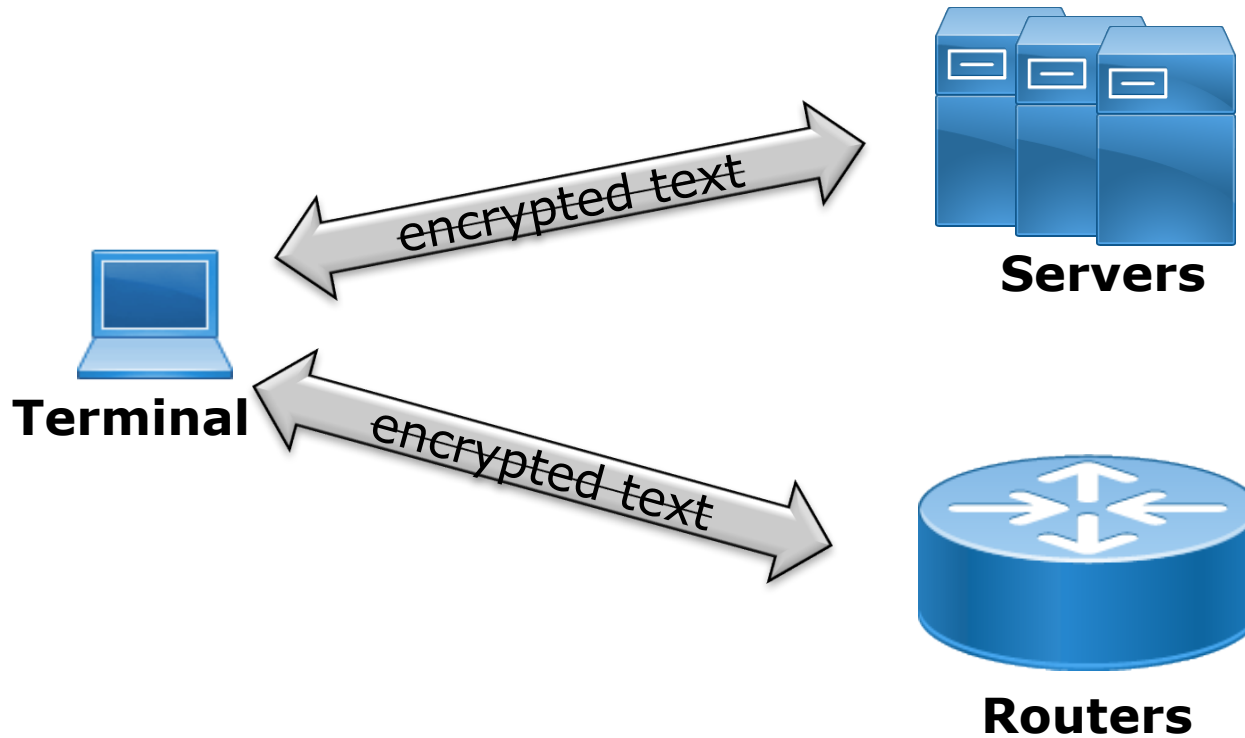
# What is “Secure”

- **Authentication** – I know who I am talking to
- Our communication is **Encrypted**

# Telnet



# SSH – encrypted channel



# Secure Shell (SSH)

- Authenticated and encrypted shell access to a remote host
- Client-server model
- TCP 22
- It is much more than a secure shell
  - Transport protocol (eg. SCP, SFTP)
  - Connection forwarder
    - You can use it to build custom tunnels

# Secure Shell (SSH) process

- Client-server crypto handshake
- Generate a symmetric key to secure the transport
- Client authenticates to server securely
- Secure communication can begin

# SSH – under the hood



**Client generates random**

- Client decrypts signed hash
- Compute and compare hashes (validate S-random)

- Compute Master key using S-random and its own random

## CipherSuite Negotiation

The highest supported by both

- Encryption algorithm
- Hashing algorithm
- Key-exchange algorithm



**Server generates random**

- Hash [C-random and S-random]
- Sign hash with its private-key

- Compute Master key using C-random and its own random

**C-random**

**S-random, signed hash, public-key (host-key)**

**Master Key**

- Master key (shared-secret) used to encrypt all messages!
- The hash is used as SSH session ID

# SSH Authentication

- Client sends its username to server over the secure channel
  - Encrypted using the “**shared master key**”
- Server checks if the username exists in the local database
  - If username not valid, tear down the SSH session!
  - If valid, the server sends back authentication method
    - Password based
    - Public-key based



# SSH Authentication – Password

- Client sends its password
  - encrypted using the shared master secret
- Server decrypts, and checks the password
  - If match found, access is granted (shell access)

# Password Authentication

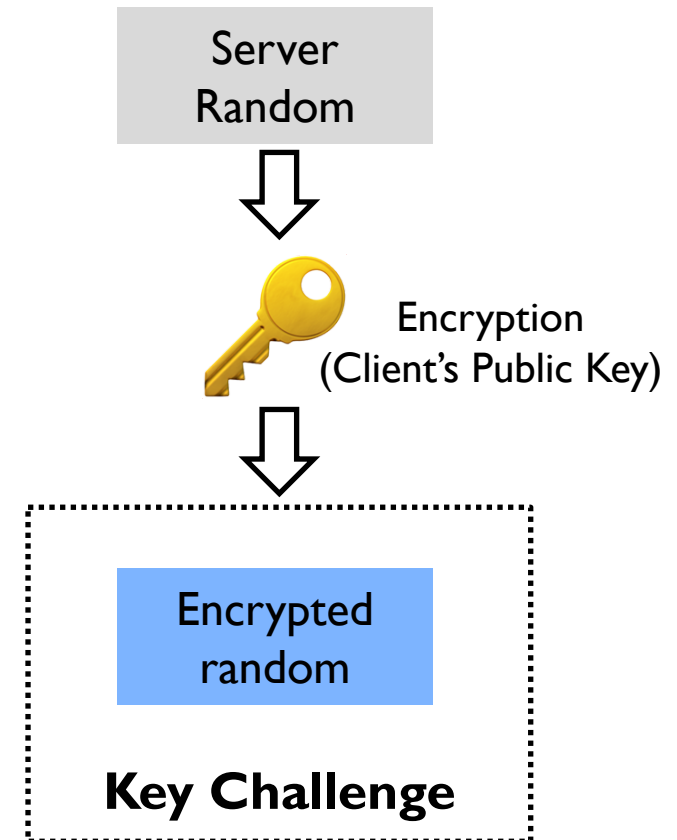
- Password Authentication is simple to set up
  - usually the default
- But allows **brute-force guessing** 😞

# SSH Authentication - Public Key

- User creates a key pair
  - public and private
    - **public key** - nonsensitive information
    - **private key** - is protected on the local machine by a strong passphrase
- Installs the public key in `$HOME/.ssh/authorized_keys` file on the target server.
  - one time installation

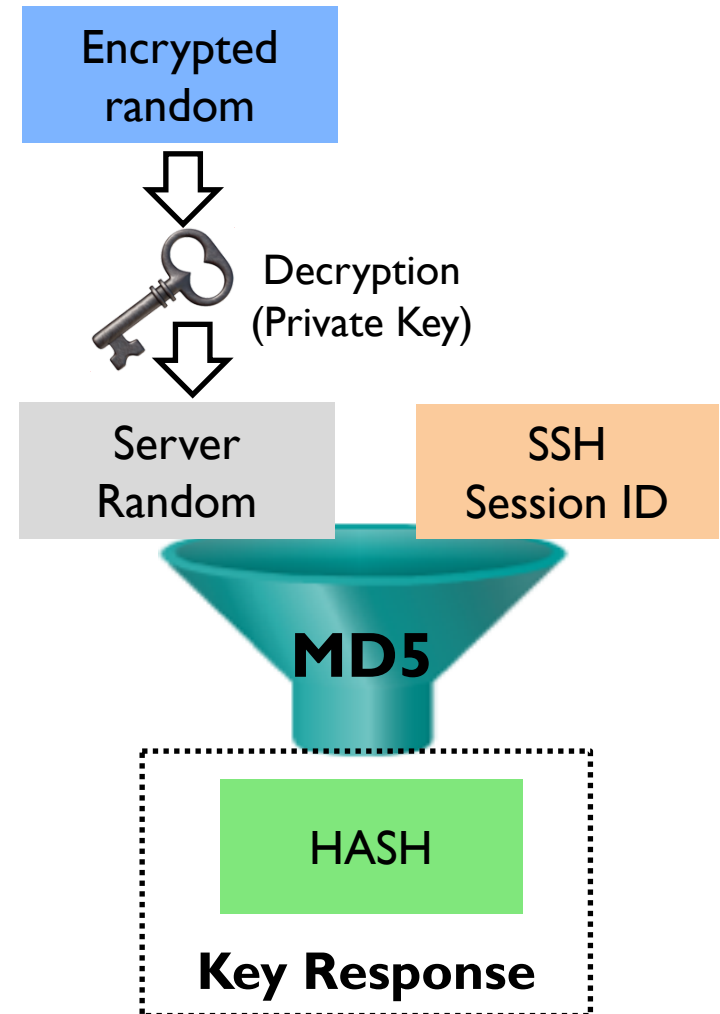
# How it works - Key Challenge

1. Client connects to server with a request to set up a key session
  - Sends KeyID for the key-pair it wants to use, and
  - Username/account-name
2. If there is a public key in the **authorized\_keys**
  - server generates a random number
  - encrypts the random using client's public
  - sends the encrypted random as a key-challenge to client



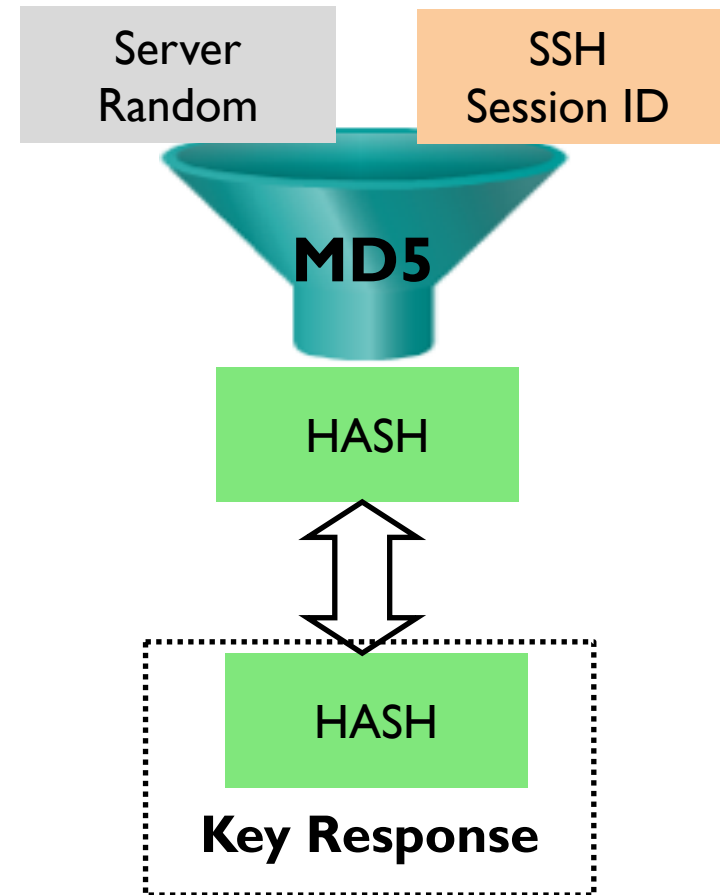
# How it works - Key Response

3. Client decrypts the random number with its private key
4. Creates an MD5 hash of the random and the session ID
  - sends back to the server as the key response
    - encrypted with shared Master key



# How it works - Access

5. The server computes its own MD5 hash and compares it with the received hash
  - **random number + session ID**
6. If the hashes match, the user must be in possession of the private key
  - **access is granted!**



# Public Key Authentication

- Cannot derive private key from public key!
  - Cannot brute force either
- Requires one-time setup of public key on target system
- Requires unlocking private key with secret passphrase upon each connection
  - If you have setup one

# Public Key Access

- Never store Private Key on a multi-user host
- Store Private Key **ONLY** on your machine and protect
  - **Encrypt Disk!**
- It is OK to use SSH\_AGENT to remember your key **ONLY** if your laptop/computer locks very quickly



# SSH Keys on Unix / MacOS

- SSH is built-in
  - UNIX
  - Linux
  - MacOS X

# Generate Key (Unix / MacOS)

```
$/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
```

```
your_email@example.com
```

# SSH Keys (Unix / MacOS)

`~/.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

<p>UNCOMMON (NON-GIBBERISH) BASE WORD</p> <p>ORDER UNKNOWN</p> <p>Tr0ub4dor &amp;3</p> <p>CAPS? COMMON SUBSTITUTIONS NUMERAL PUNCTUATION</p> <p>(YOU CAN ADD A FEW MORE BITS TO ACCOUNT FOR THE FACT THAT THIS IS ONLY ONE OF A FEW COMMON FORMATS.)</p>	<p>~28 BITS OF ENTROPY</p> <p><math>2^{28} = 3 \text{ DAYS AT } 1000 \text{ GUESSES/SEC}</math></p> <p>(PLAUSIBLE ATTACK ON A WEAK REMOTE WEB SERVICE. YES, CRACKING A STOLEN HASH IS FASTER, BUT IT'S NOT WHAT THE AVERAGE USER SHOULD WORRY ABOUT.)</p> <p>DIFFICULTY TO GUESS: <b>EASY</b></p>	<p>WAS IT TROMBONE? NO, TROUBADOR. AND ONE OF THE 0s WAS A ZERO?</p> <p>AND THERE WAS SOME SYMBOL...</p> <p>DIFFICULTY TO REMEMBER: <b>HARD</b></p>
<p>correct horse battery staple</p> <p>FOUR RANDOM COMMON WORDS</p>	<p>~44 BITS OF ENTROPY</p> <p><math>2^{44} = 550 \text{ YEARS AT } 1000 \text{ GUESSES/SEC}</math></p> <p>DIFFICULTY TO GUESS: <b>HARD</b></p>	<p>THAT'S A BATTERY STAPLE.</p> <p>CORRECT!</p> <p>DIFFICULTY TO REMEMBER: YOU'VE ALREADY MEMORIZED IT</p>

THROUGH 20 YEARS OF EFFORT, WE'VE SUCCESSFULLY TRAINED EVERYONE TO USE PASSWORDS THAT ARE HARD FOR HUMANS TO REMEMBER, BUT EASY FOR COMPUTERS TO GUESS.

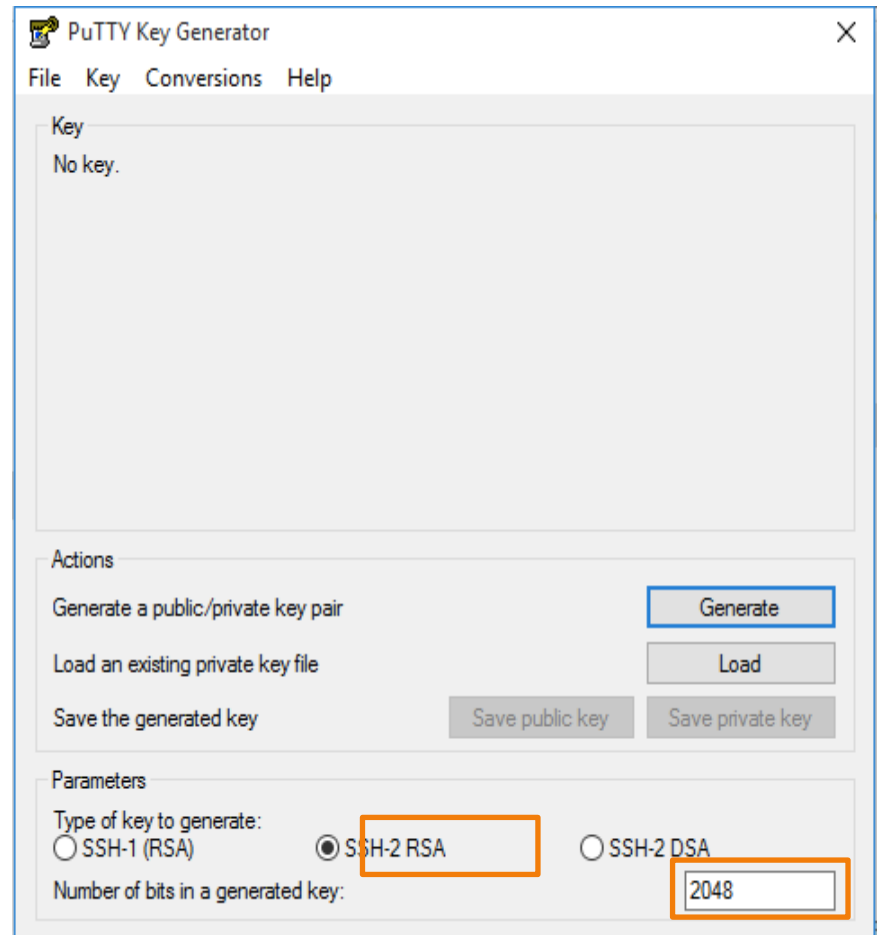
source : <http://xkcd.com/936/>

# Private Key on Windows

- <https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.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)

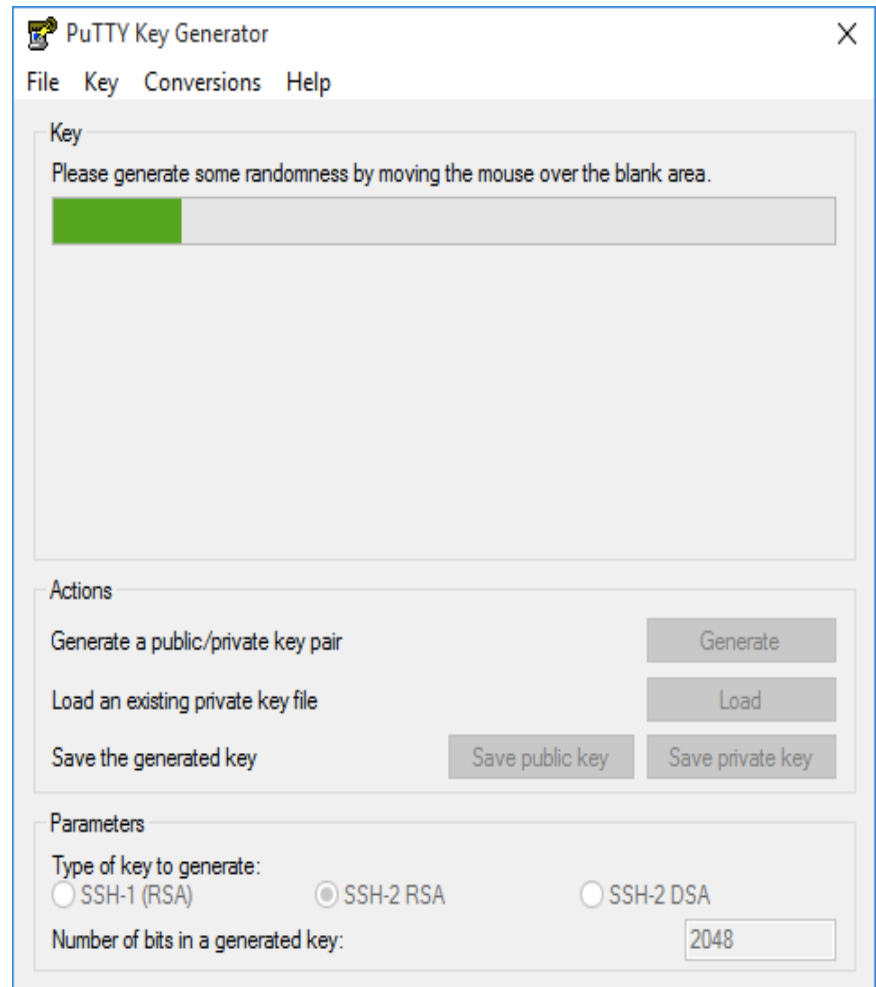
# Generate Key (Windows)

## 1. Run PuttyGen



# Generate Key (Windows)

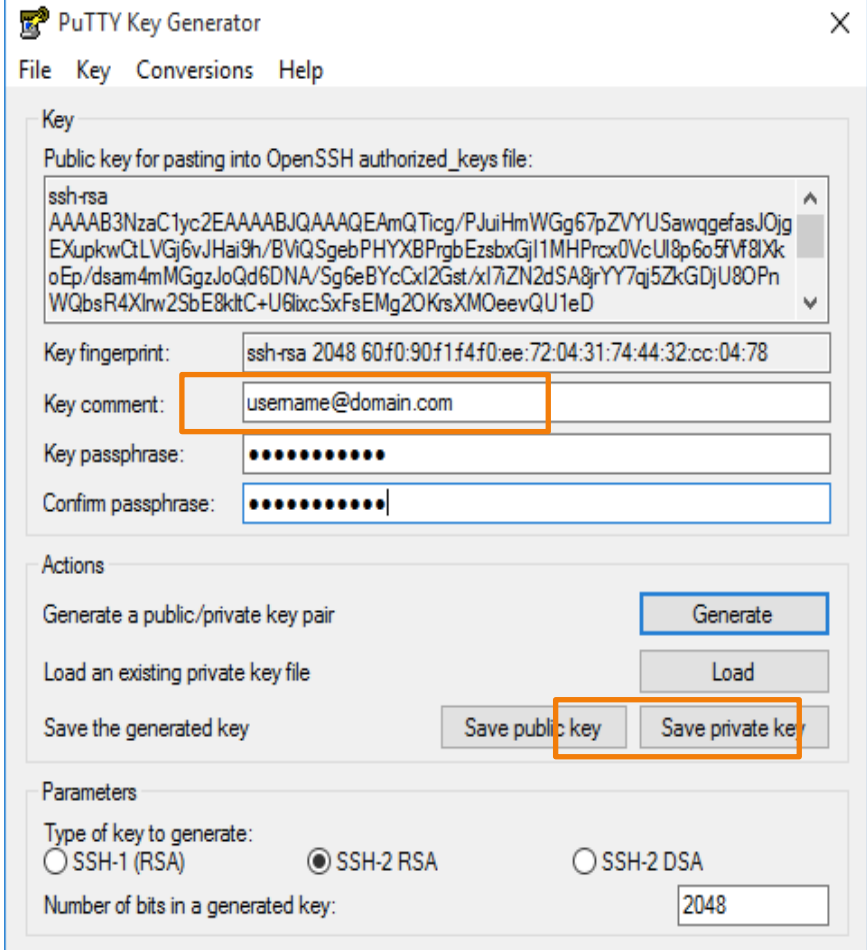
## 2. Generate Key



# 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



The screenshot shows the PuTTY Key Generator application window. The 'Key' section displays a generated public key for an 'ssh-rsa' type. The key fingerprint is 'ssh-rsa 2048 60:f0:90:f1:f4:f0:ee:72:04:31:74:44:32:cc:04:78'. The key comment is 'username@domain.com'. The key passphrase and confirm passphrase fields are filled with dots. The 'Actions' section includes buttons for 'Generate', 'Load', 'Save public key', and 'Save private key'. The 'Parameters' section shows 'SSH-2 RSA' selected as the key type and '2048' bits for the key length.

PuTTY Key Generator

File Key Conversions Help

Key

Public key for pasting into OpenSSH authorized\_keys file:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAABJQAAAQEAmQTicg/PJuiHmWGg67pZVYUSawqgefajJOjg
EXupkwCtLVGj6vJHai9h/BViQSgebPHYXBPrgbEzsbxGj11MHPrcx0VcUl8p6o5Vf8Xk
oEp/dsam4mMGgzJoQd6DNA/Sg6eBYcCx12Gst/xl7iZN2dSA8jrYY7qj5ZkGDjU8OPn
WQbsR4Xlrv2SbE8ktC+U6lxcSxFsEMg20KrsXMOeevQU1eD
```

Key fingerprint: ssh-rsa 2048 60:f0:90:f1:f4:f0:ee:72:04:31:74:44:32:cc:04:78

Key comment: username@domain.com

Key passphrase: .....

Confirm passphrase: .....

Actions

Generate a public/private key pair

Load an existing private key file

Save the generated key

Parameters

Type of key to generate:

SSH-1 (RSA)  SSH-2 RSA  SSH-2 DSA

Number of bits in a generated key: 2048



# Saving 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

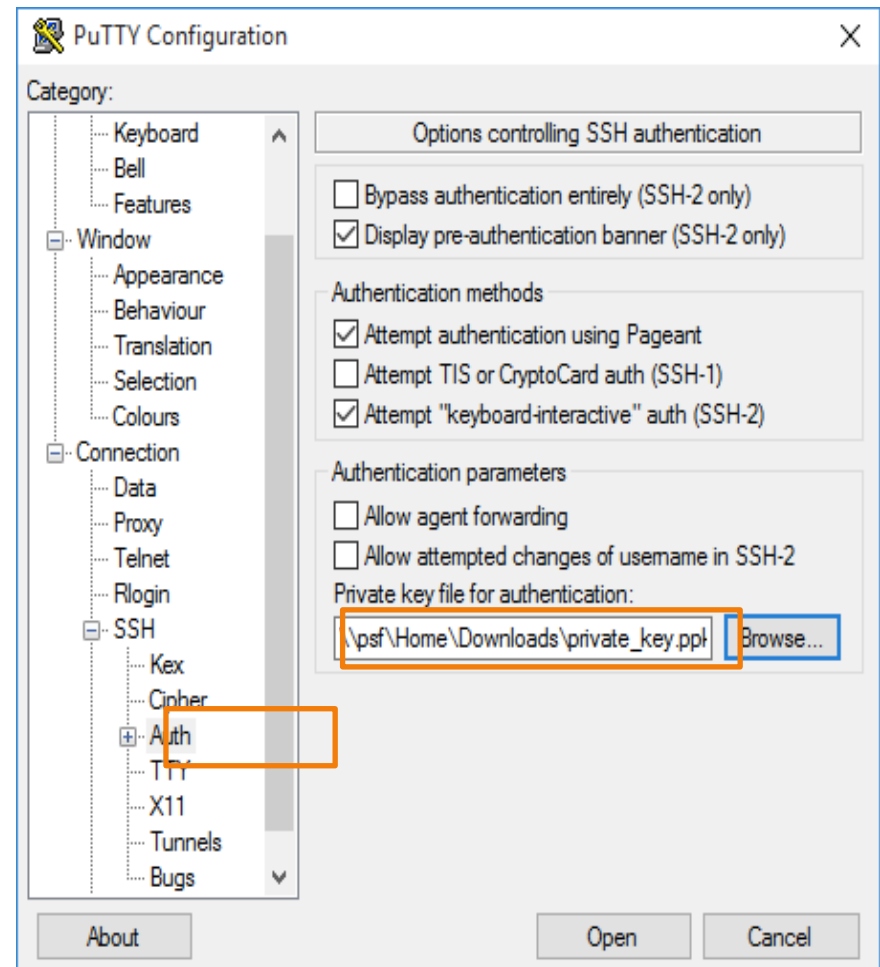
```
ssh-copy-id user@serverip
```

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

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

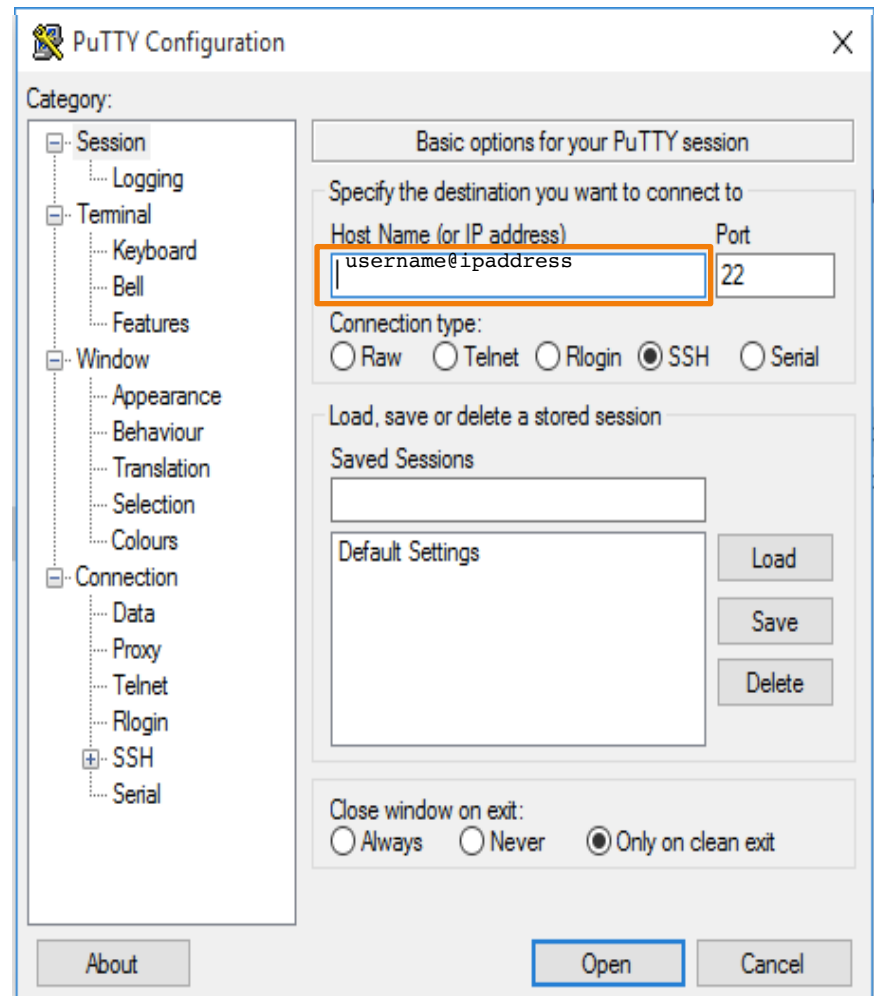
# Saving Key on the Target Host - Windows

## 4. Load Key in Putty



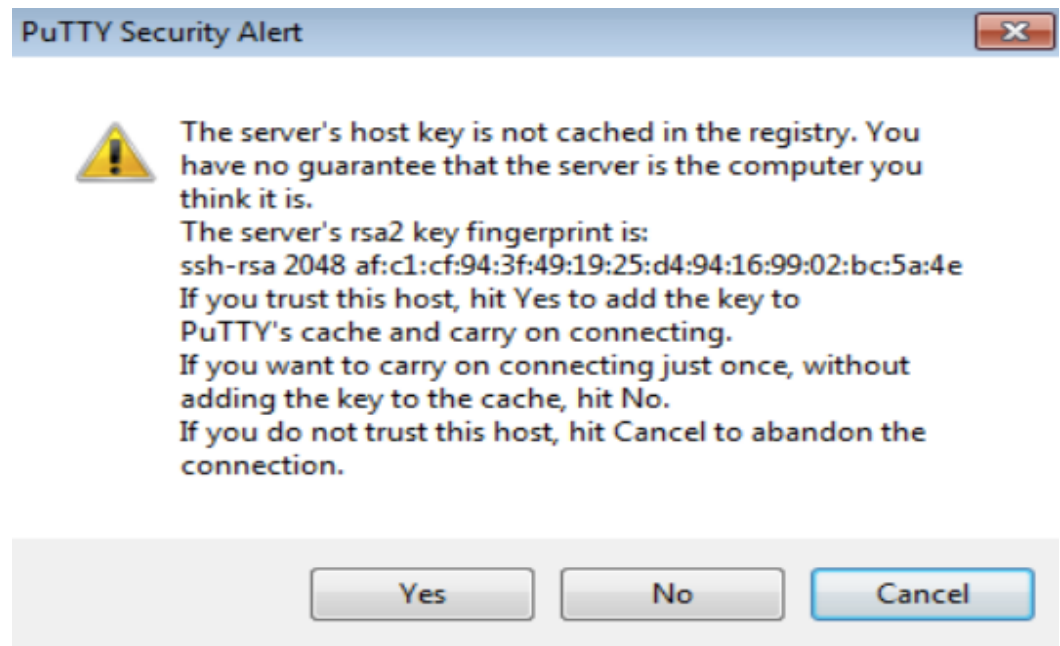
# Saving Key on the Target Host - Windows

## 5. SSH to host



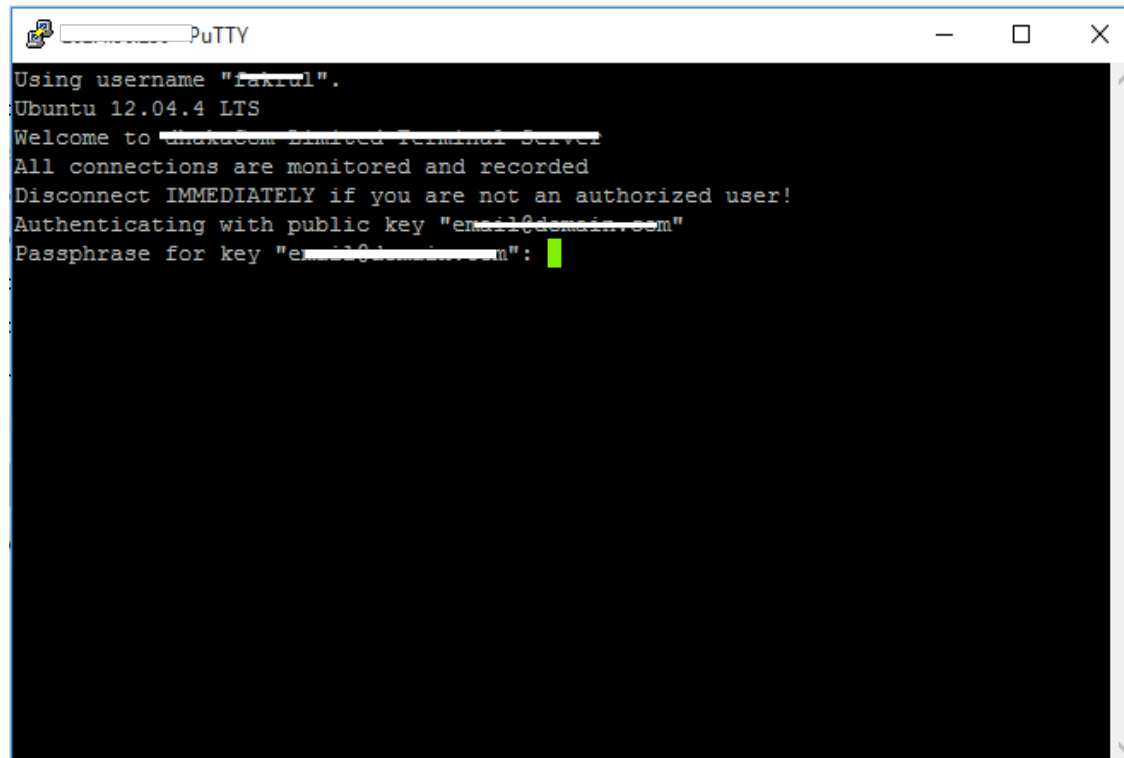
# Saving Key on the Target Host - Windows

## 6. Accept Host's Key



# Saving Key on the Target Host - Windows

## 7. passphrase for Key



```
Using username "fakrul".
Ubuntu 12.04.4 LTS
Welcome to unltd.com Limited Terminal Server
All connections are monitored and recorded
Disconnect IMMEDIATELY if you are not an authorized user!
Authenticating with public key "email@domain.com"
Passphrase for key "email@domain.com": █
```

# Lab Exercise

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