ssh and lab

APRICOT 2020

Melbourne Australia

2020.02.12-16

Acknowledgement

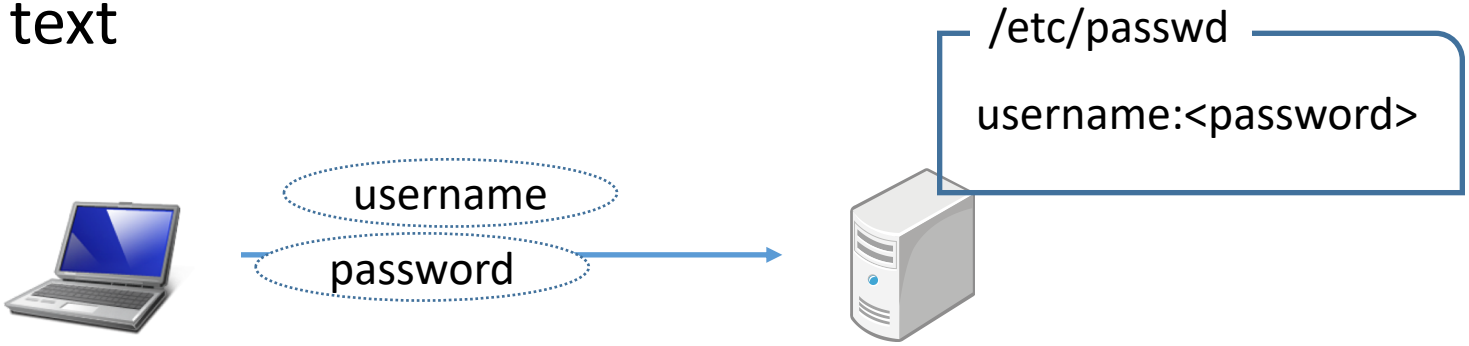
- Original slides made by
  - Yoshinobu ‘maz’ Matsuzaki <maz@iij.ad.jp>

Secure Shell (ssh)

- Replacement for unsecure tools/protocols
  - rsh and telnet
- Usually listen on tcp/22
- Whole communication is encrypted
- Ability to check server’s signature
- Multiple ways to authenticate users
  - public key
  - password

telnet – how insecure?

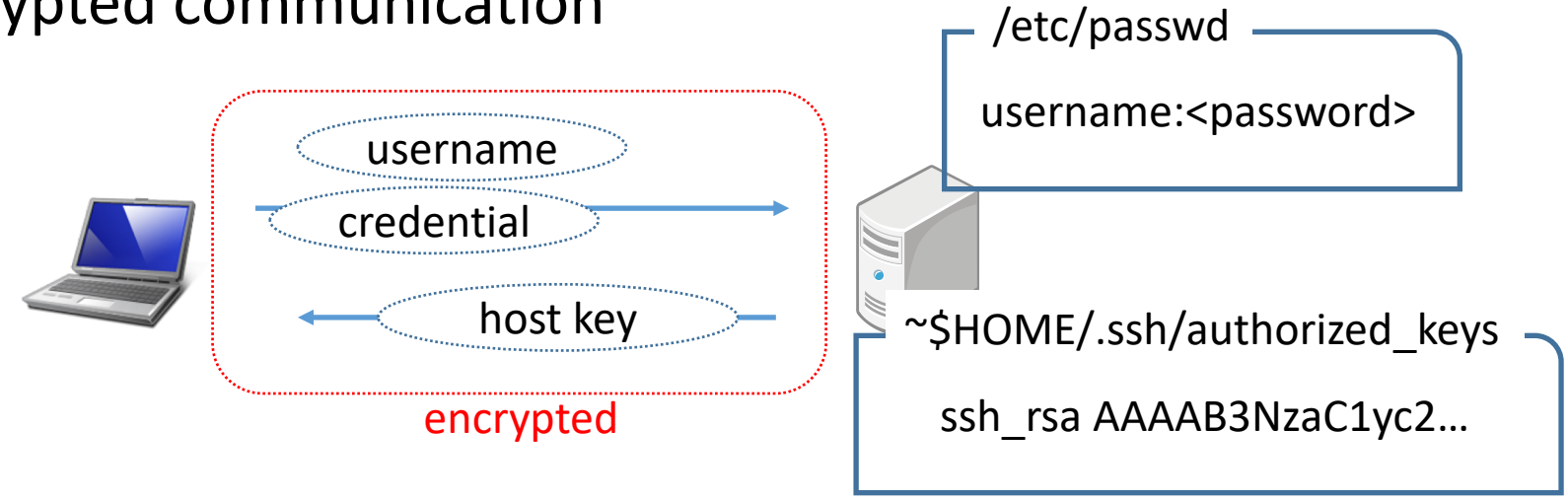
- Checks ‘username’ + ‘password’
- Plain text



- Anyone on the wire can monitor the communication
- Password leakage / guess
- A fake server can steal username & password

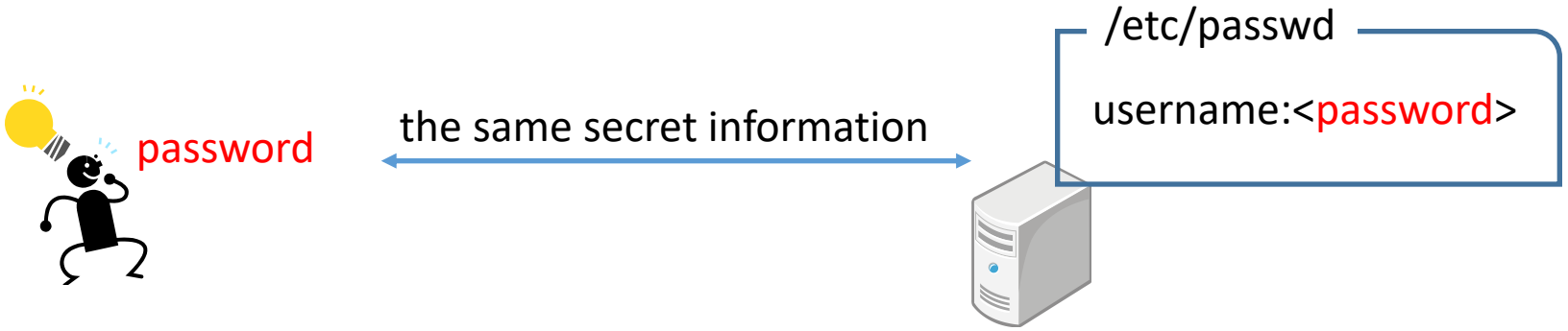
ssh

- ‘username’ + (‘password’ or ‘public key’)
- encrypted communication



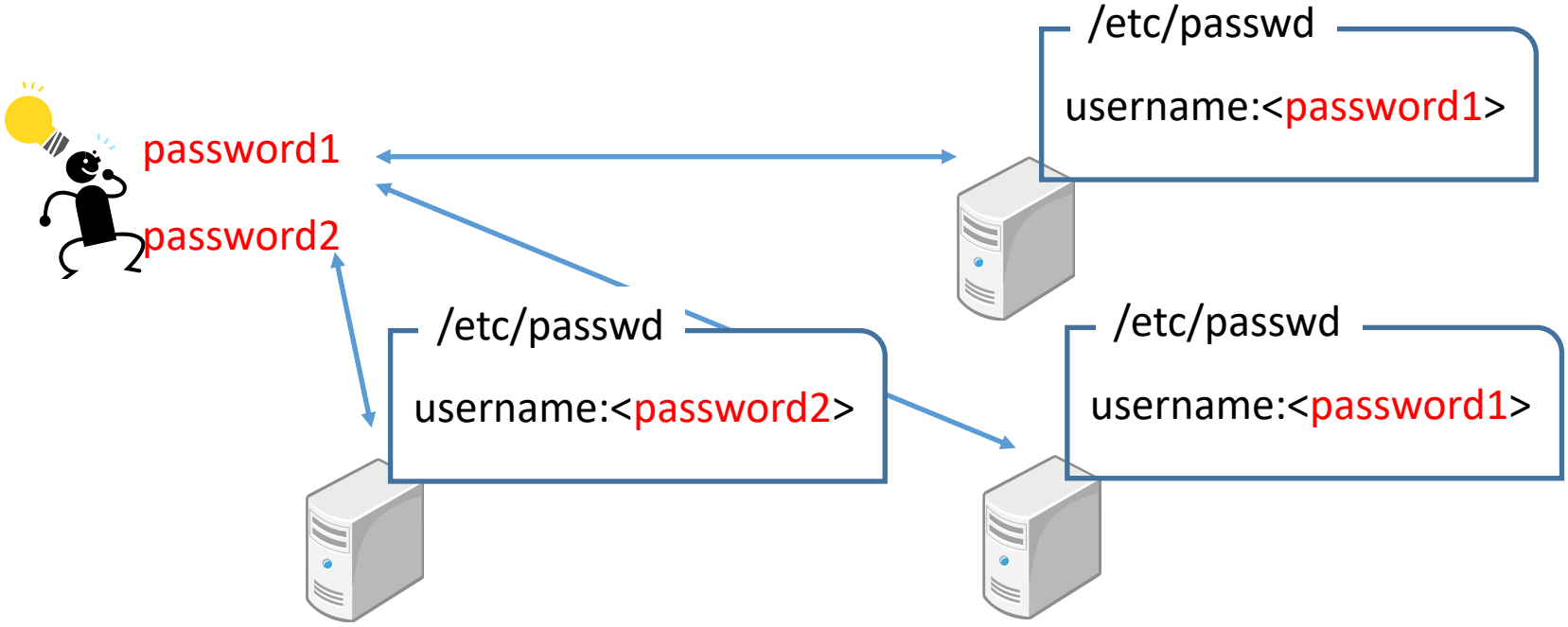
- Anyone can monitor the traffic, it’s encrypted though
- Still there is a risk of password leakage / guess for password authentication

Password authentication: setup



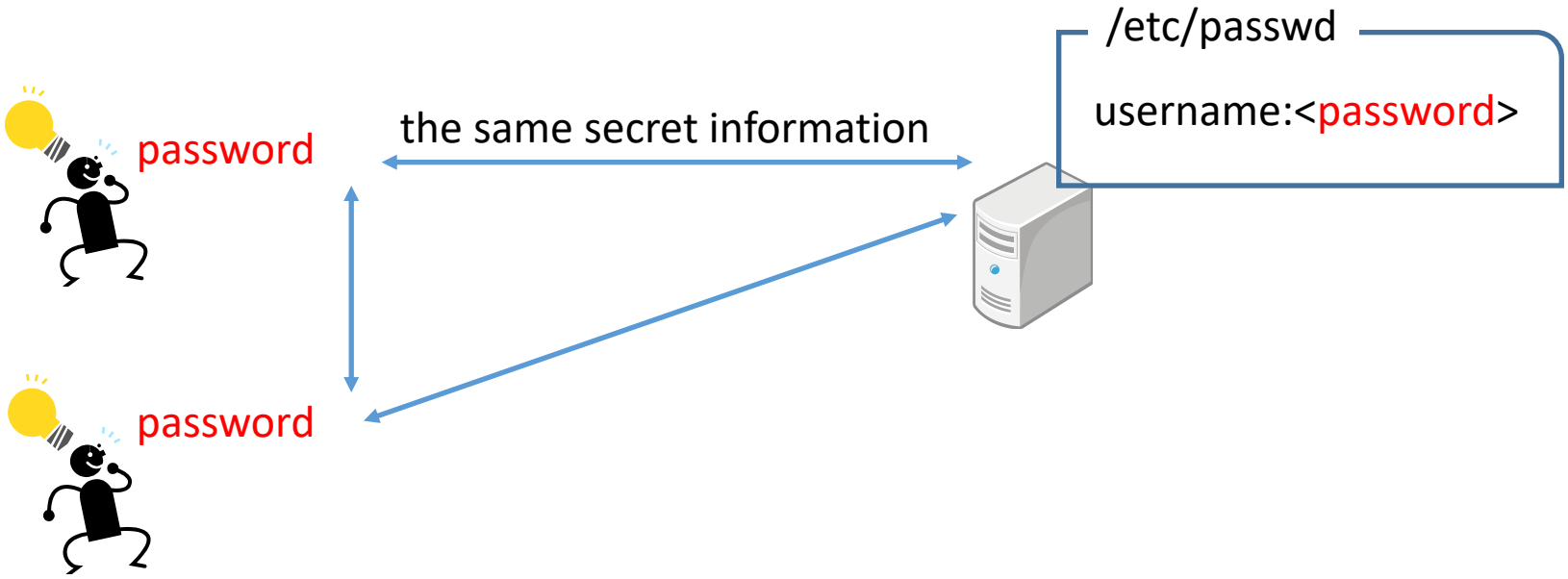
- Agree on a secret information called password per user

Passwords for multiple hosts



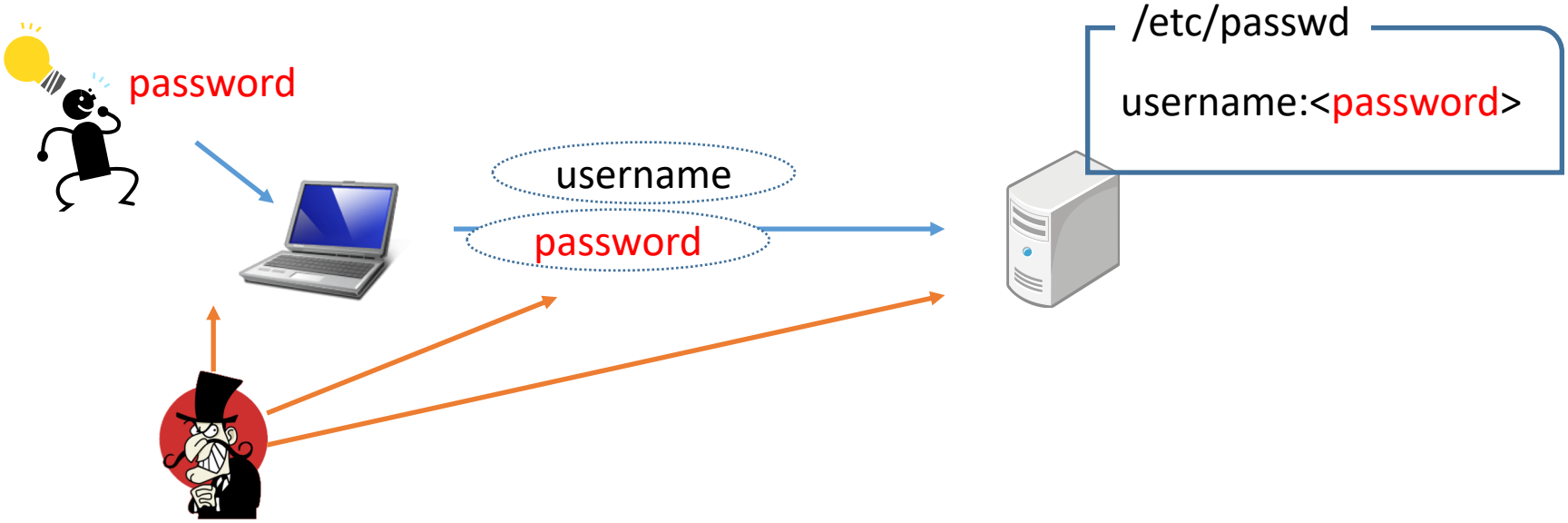
- User can use the same password or different ones per host
- User must remember combinations of host and password

Password for a shared account



- Users need to share the secret information

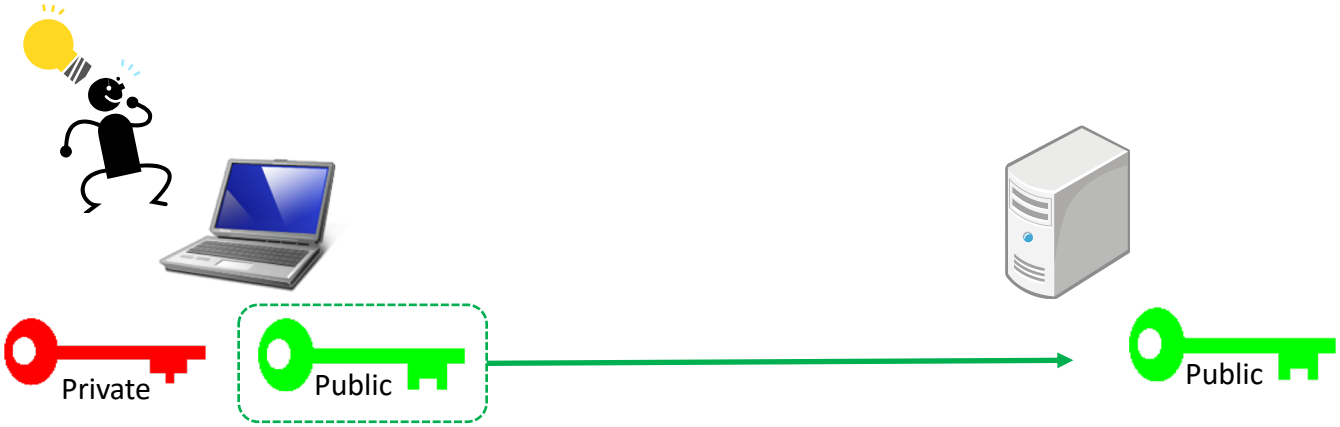
Password authentication is danger



- Users should:
  - remember it
  - type it
  - share it with remote hosts

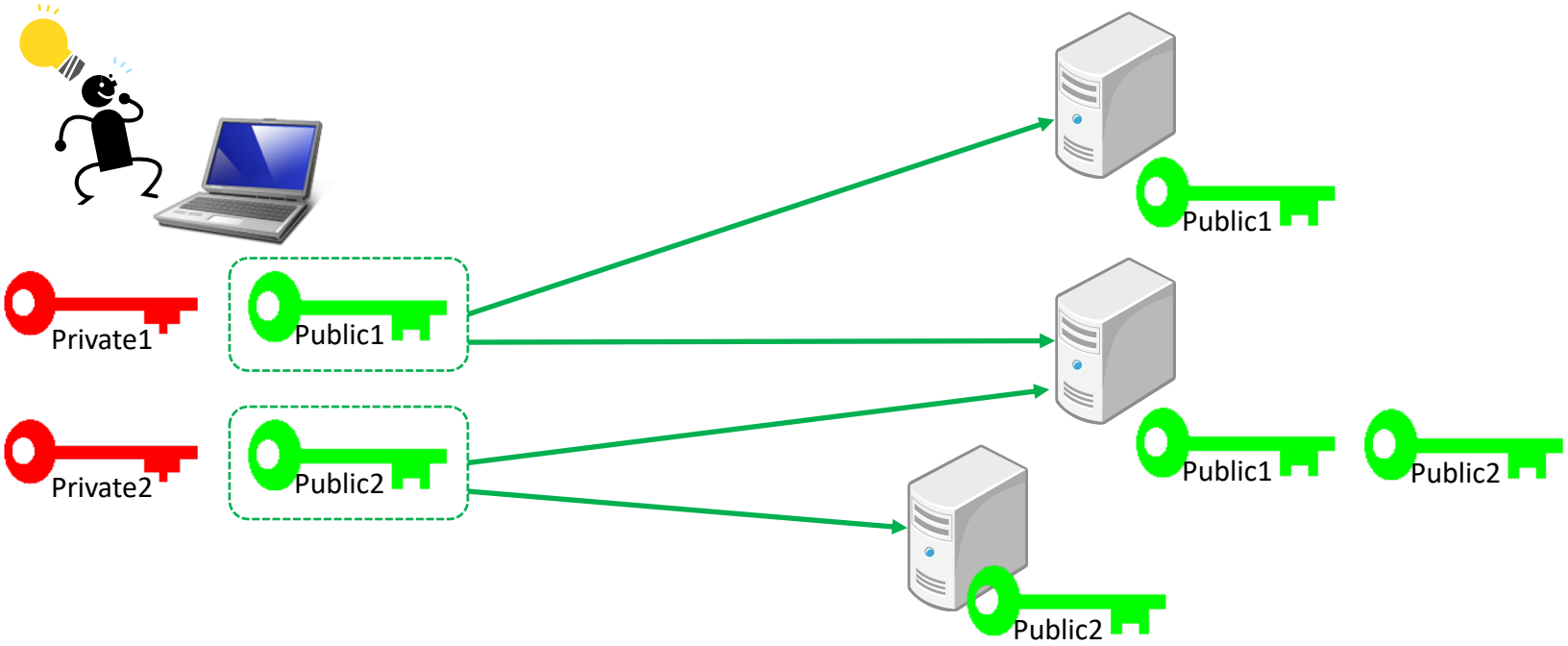
- a password tends to be short
- using the same one on multiple hosts
- risks of shoulder hacking
- it's leaky

Public key authentication: setup



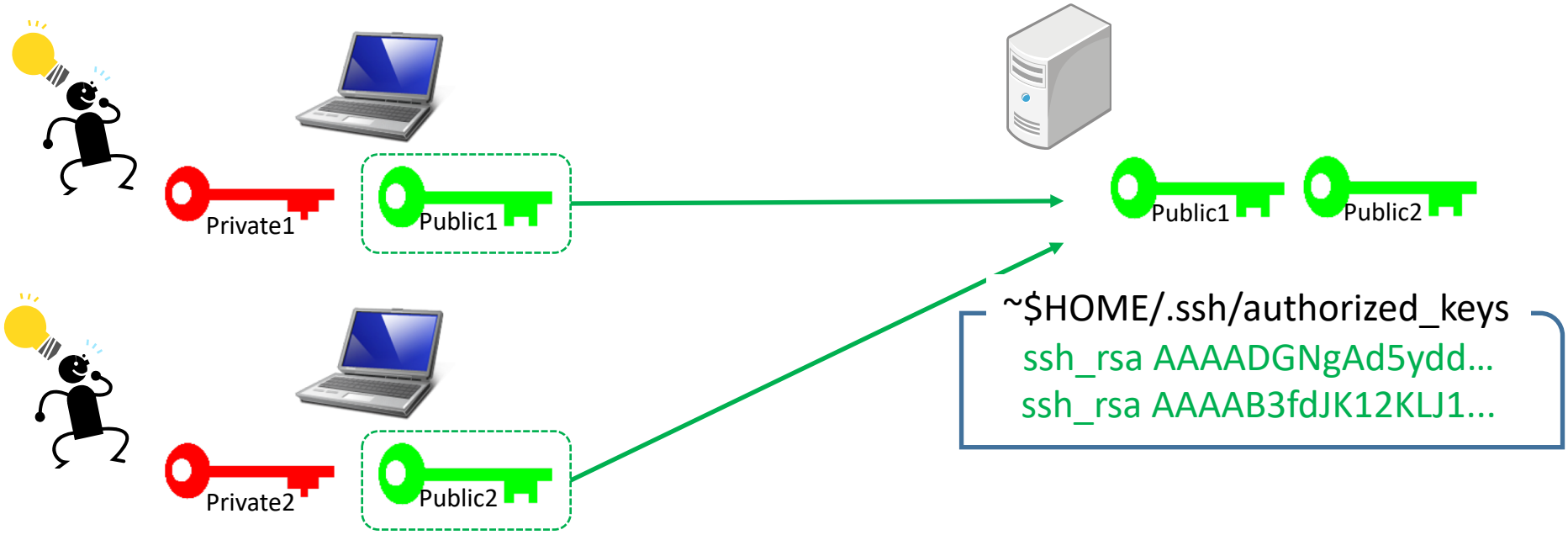
- Generate a key pair
- Send the public key to a remote host
  - On an UNIX host, authorized public keys for the user should be stored in '$HOME/.ssh/authorized_keys'
  - Other devices have own configuration formats to store authorized public keys

Keys for multiple host



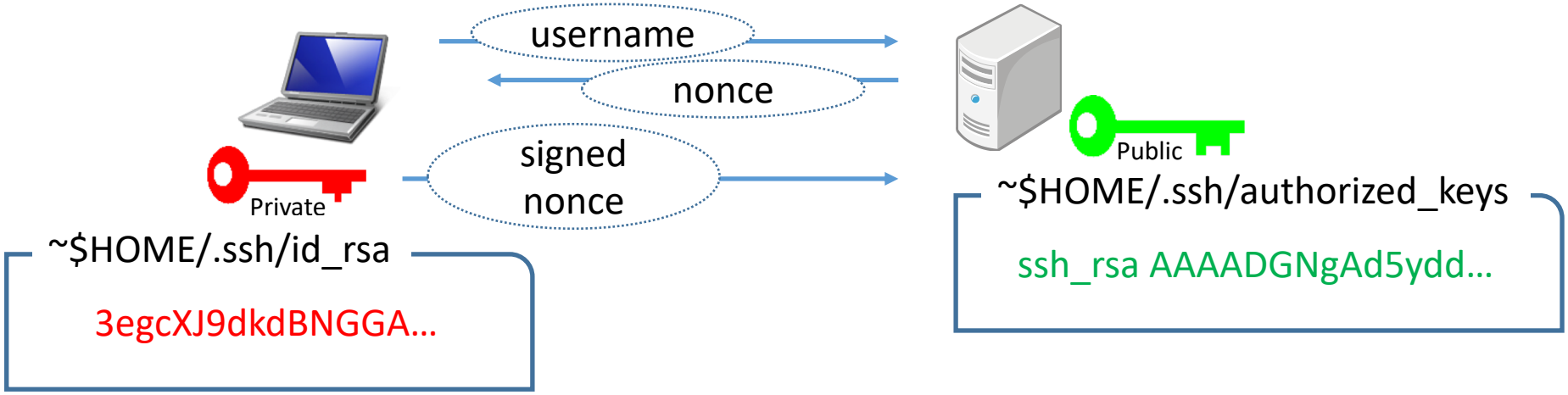
- User can use the same key pair or a different key pair, and a host can store multiple public keys per user
- Modern software automatically chooses an appropriate private key during authentication

Key for a shared account



- Each user can have own key pair
  - Or you can share a private key among users (not recommended)

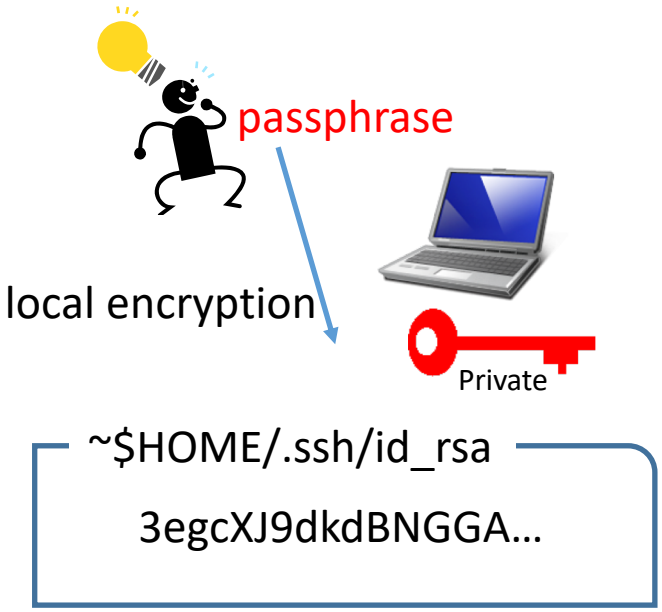
Public key authentication



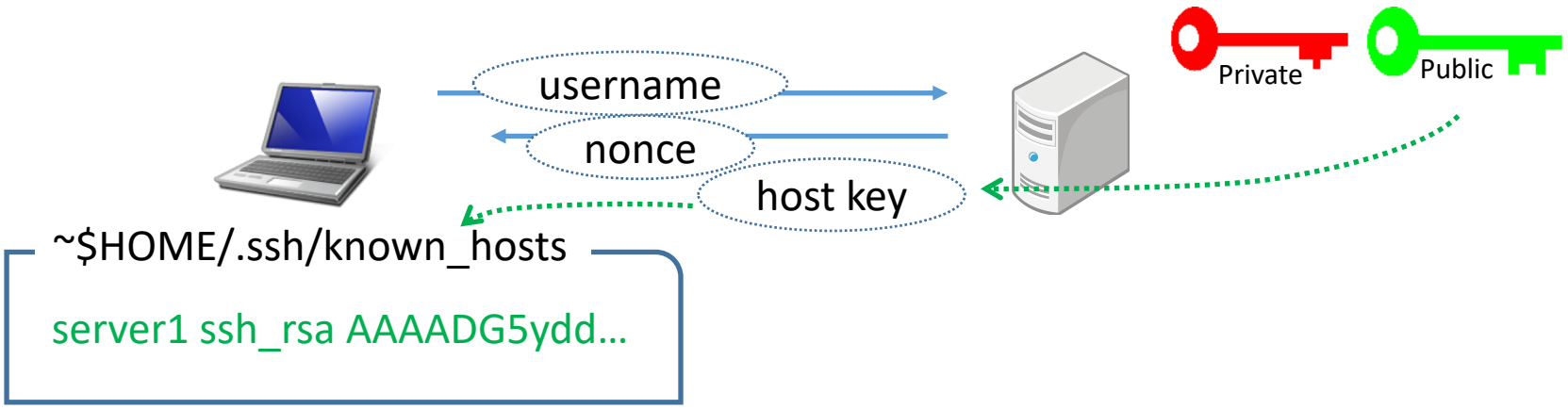
- A digital signature signed by private key can be verified by corresponding public key
  - It proves the private key holder is trying to login

Private key

- Key authentication is highly relying on the secrecy of a private key
- Keep it secure and secret
  - Store it in a secure host only
- Set a passphrase to encrypt the private key file locally
  - Decrypt and use it when needed
  - You can change the passphrase anytime, and still the public key is the same and unchanged



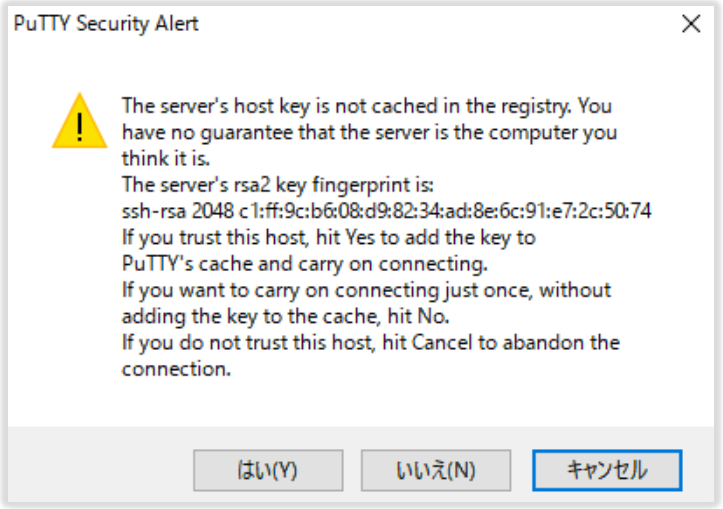
Host authentication



- A ssh server has own key pair (host key)
  - Sends the public key during session initialization
- A client stores the public keys in a file, and verifies and uses it during session setup
  - On an UNIX, the file is '$HOME/.ssh/known_hosts'
  - Used to decrypt information from the host

During the initial connection

```
$ ssh 10.0.0.1
The authenticity of host '10.0.0.1 (10.0.0.1)' can't be established.
ECDSA key fingerprint is SHA256:WrHnt6dnAlhEZvBU5H5WGQUqIMrFFbL18LBGM3u/NrI.
Are you sure you want to continue connecting (yes/no)?
```



- If you don’t have the host key, clients ask you whether you trust the key or not
  - ’yes’ if you are comfortable
- Or you can put the key into the file manually in advance

When the host key doesn't match

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that a host key has just been changed.
The fingerprint for the ECDSA key sent by the remote host is
SHA256:wqfhRI5/APqcB7Fl+aqB1Np3fkuBI6fVtD4Dms2sOu4.
Please contact your system administrator.
Add correct host key in /home/workshop/.ssh/known_hosts to get rid of this message.
Offending ECDSA key in /home/workshop/.ssh/known_hosts:16
  remove with:
  ssh-keygen -f "/home/workshop/.ssh/known_hosts" -R 10.0.0.1
ECDSA host key for 10.0.0.1 has changed and you have requested strict checking.
Host key verification failed.
```

- It could be just reinstalled/replaced server
- But pay attention just in case...

ssh-agent

- Holds decrypted private key in the process and use it for authentication
  - You do not need to type passphrase every time when logging in to a remote host
  - During the startup process, you will be asked your passphrase to decrypt and store your private key
  - ssh clients work with the agent nicely
- Use the agent on a trusted host only
  - like your own local pc

ssh implementations

- OpenSSH
  - https://www.openssh.com/
  - build in most UNIX systems, MacOS and Windows
- PuTTY
  - http://www.chiark.greenend.org.uk/~sgtatham/putty/
  - For Windows
- and more!
  - For androids, iOS devices

Key algorithms

- rsa2048 is pretty common
  - some routers support rsa1024 only
- A paranoid might use
  - rsa4098
  - ed25519

- These should match with your server side capabilities

Generating a key pair (OpenSSH)

```
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/workshop/.ssh/id_rsa): <enter>
Created directory '/home/workshop/.ssh'.
Enter passphrase (empty for no passphrase): <your passphrase>
Enter same passphrase again: <your passphrase again>
Your identification has been saved in /home/workshop/.ssh/id_rsa.
Your public key has been saved in /home/workshop/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:Ew4VveDGVRoQLm6H4SDT1O3NwIg6drb+YNhw7m4Jg0I workshop@ws
The key's randomart image is:
+---[RSA 2048]----+
|     .o +==...    |
|    o. oo= oo     |
|   o.o ++o*..     |
| E+oo+ *=.+       |
|.o.+..=.S         |
|o o*.. . .        |
|. .+=.            |
|   o+.            |
|   oo..           |
+----[SHA256]-----+
```

You have a key pair (OpenSSH)

```
$ cd .ssh/
$ ls -l
total 8
-rw------- 1 workshop workshop 1675 Nov 13 09:44 id_rsa      <- your private key
-rw-r--r-- 1 workshop workshop  392 Nov 13 09:44 id_rsa.pub  <- your public key
$ cat id_rsa.pub
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQCyMGJvtGry4Pgh9mvyRbuAUX961yR0YTmFc34LqKBUM0CYuZqu/uIyP4
J78rqshUMVWj15zTKlP+IRonYJS7idLKDuqvkm1oJXqYim+2TjeNY3rDSPchkt6xHTxKnmLglShBITrQr+h3jp
NeRLaxlMStTx86opE4kPd2LF0Dv4w0RDQEz8A6yHS0d12ZNG4SPNomeuwPiZuRB6CPkFPwxR9PpEoYg0kq90Zl
fx00zIaJbkVd/u/G9T+ctRTnS+3hSMHaAZ6c04q6+CAw6PR27t0nE+51rM7HRSlwC0FwACBpF2tX8+weRe8u0h
h4BJnm132LZVDYO99Td4zx5pzwgv workshop@ws
```

Putting the key on the target host (OpenSSH)

- login the target host first, and edit the file
  - $ mkdir -p ~/.ssh
  - $ chmod 0700 ~/.ssh
  - $ vi ~/.ssh/authorized_keys
  - copy and paste your public key there

- Note: each public key should be one line in the file
  - without CR/LF

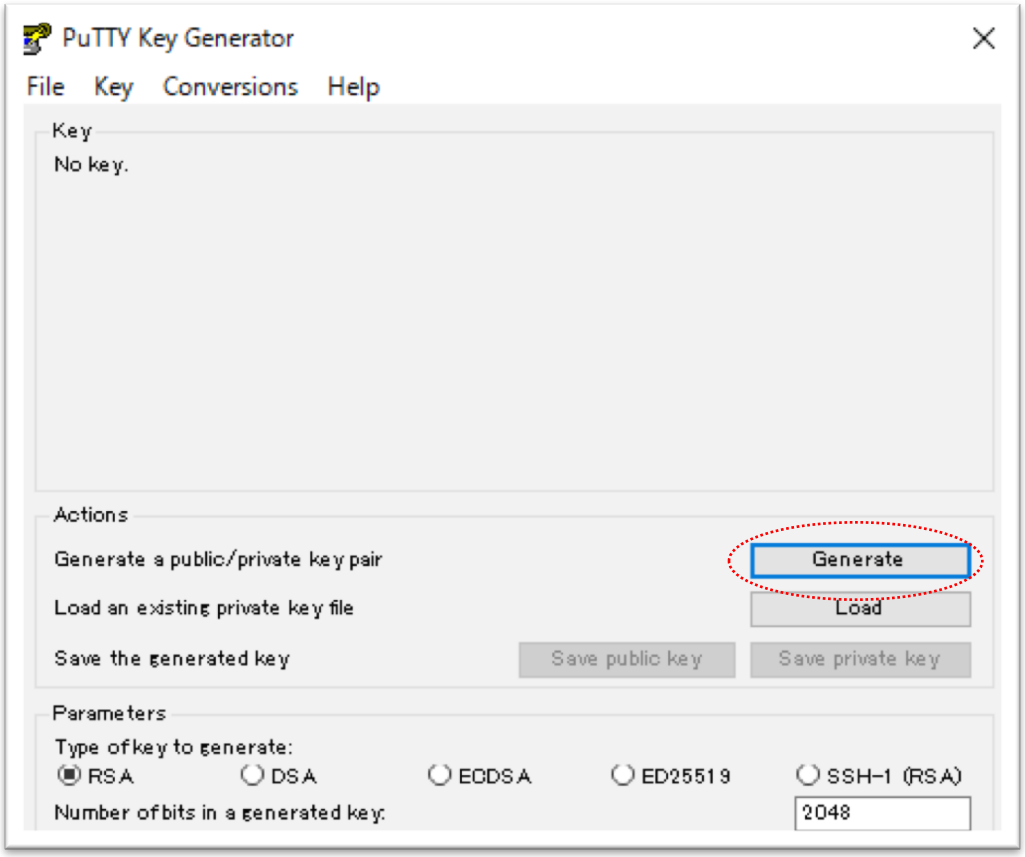
ssh key authentication (OpenSSH)

- $ ssh <username>@<target_host>

- OpenSSH client automatically use keys those have default notation in the ~/.ssh folder
  - id_rsa, id_id_ecdsa, and so on

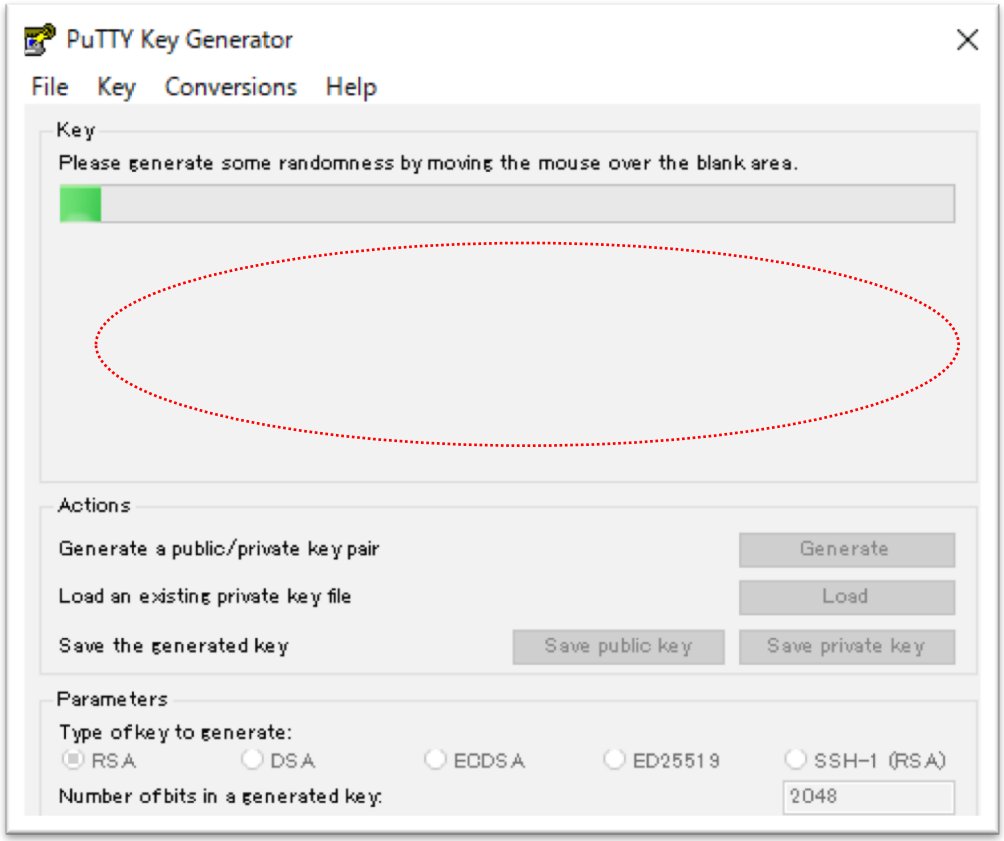
Generating a key pair (Putty)

1. Download ‘puttygen.exe’ and execute it
2. Pick parameters as you like (default setting is RSA2048 now), and ’Generate’



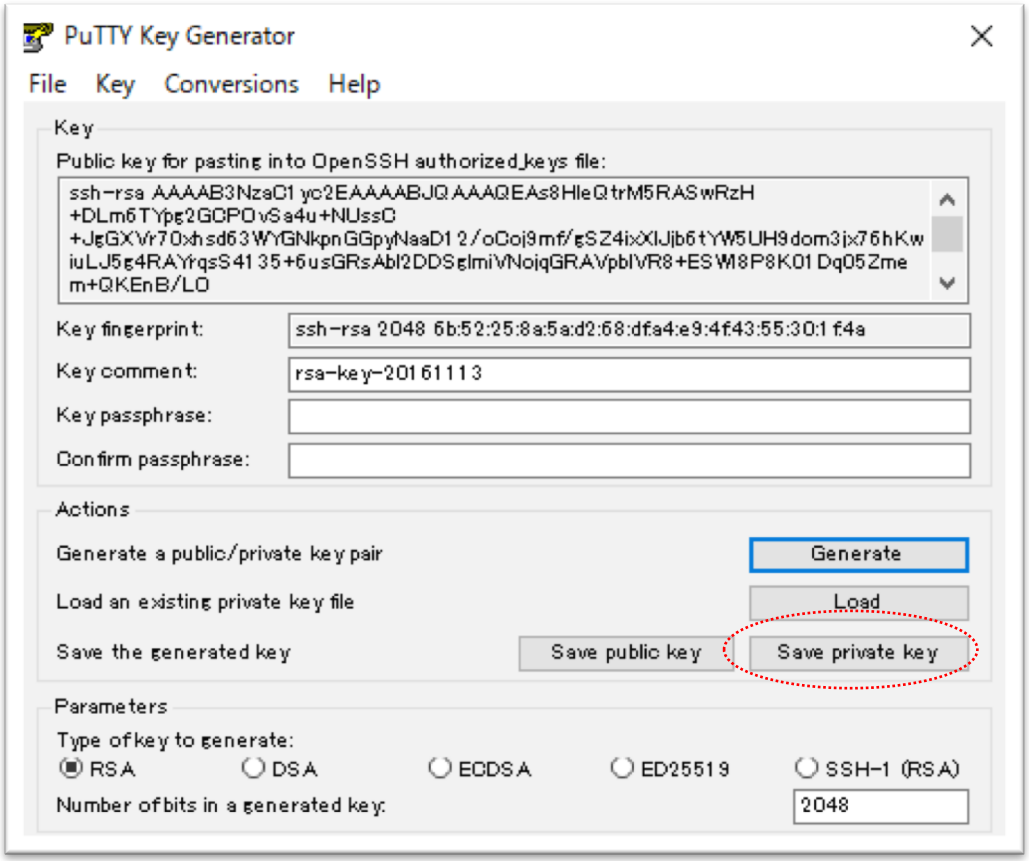
Generating a key pair (Putty)

3. Move your mouse in the blank area as the application says until it gets finished



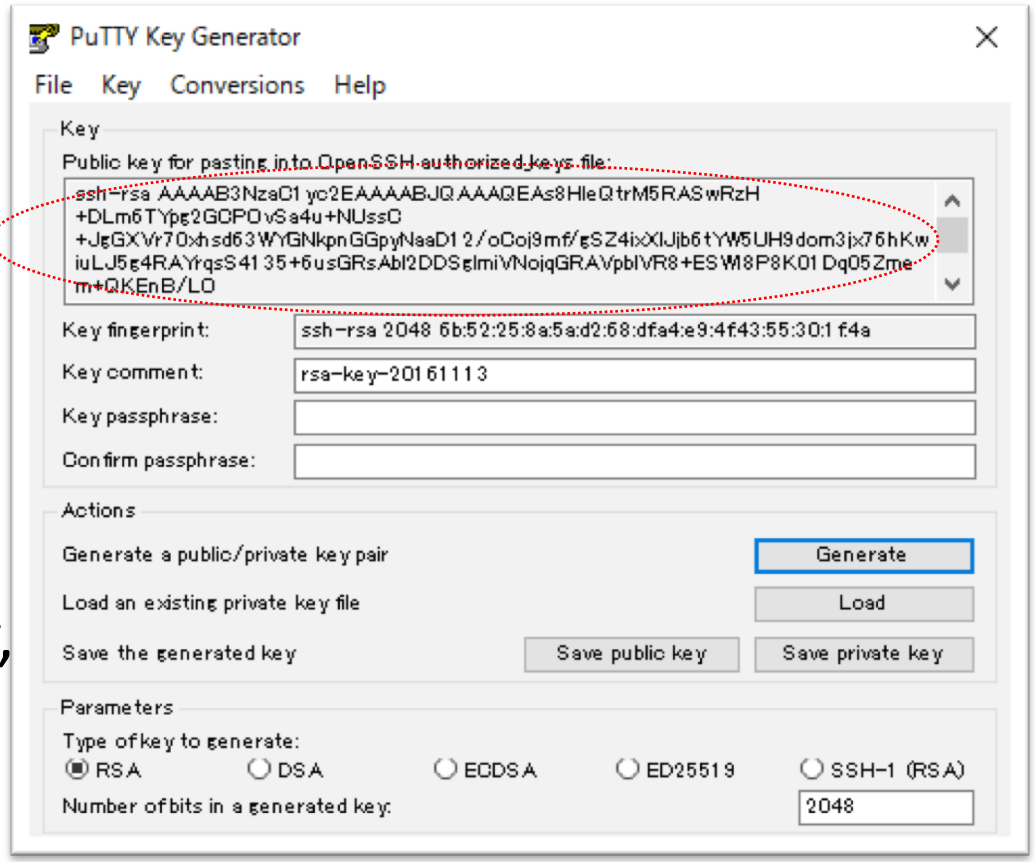
Generating a key pair (Putty)

4. Name and save your private key somewhere in your folder



Generating a key pair (Putty)

5. Right-click in the text field labeled ’Public key for pasting into OpenSSH authorized_keys file’ and choose “Select All” and “copy” the key
6. Open ‘notepad’, paste your public key, then save as a text file



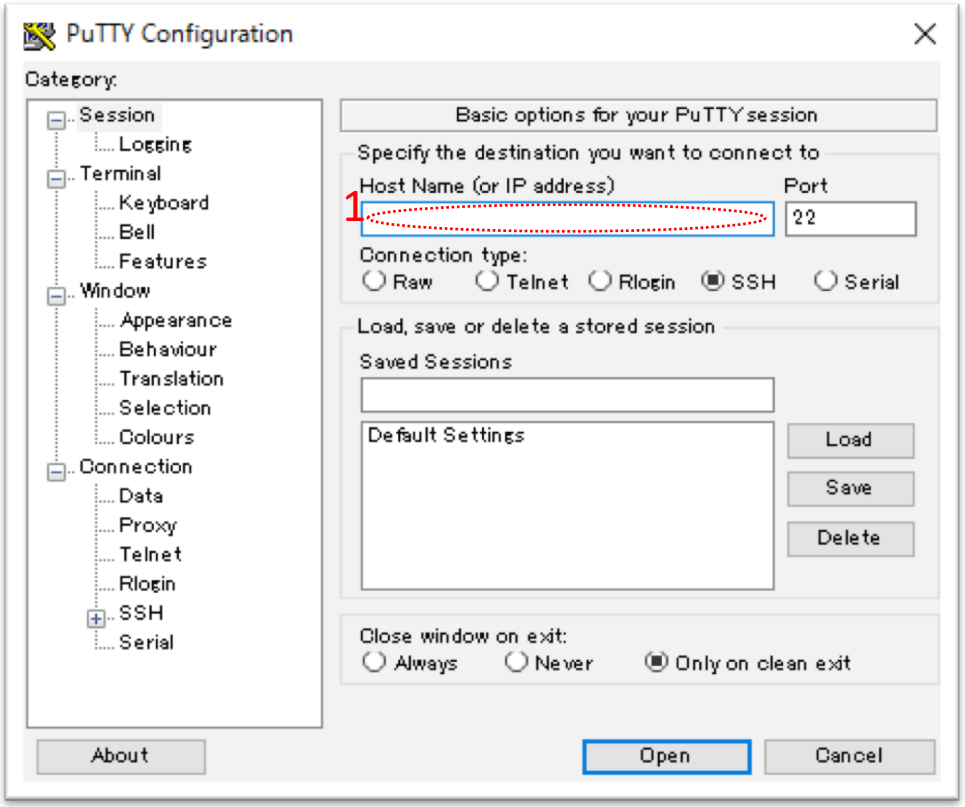
Putting the key on the target host (Putty)

- Login the target host first, and edit the file
  - $ mkdir -p ~/.ssh
  - $ chmod 0700 ~/.ssh
  - $ vi ~/.ssh/authorized_keys
  - copy your key from the public key file
  - type i on the ssh session window to insert new text in the file
  - right click your mouse to paste your public key
  - press Esc and type :wq then <enter> to overwrite the file and quit vi

- Note: each public key should be one line in the file
  - without CR/LF

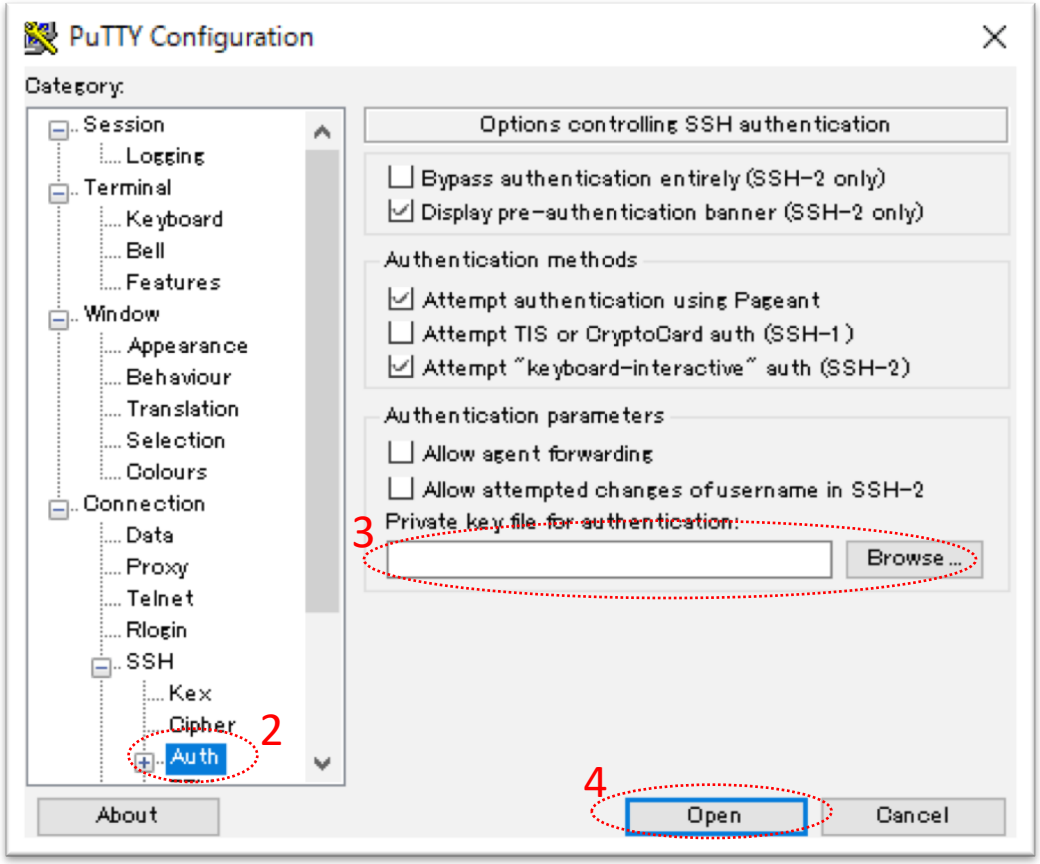
ssh key authentication (Putty)

1. Set ‘<username>@<target_host>’ in the Host Name field



ssh key authentication (Putty)

2. Go to ‘Connction’ -> ‘ssh’ -> ‘Auth’
3. ‘Browse’ and find your saved private key and set the file there
4. ‘Open’



hands on

Hands on overview

- Download software if necessary
  - Required only for Windows 10 earlier than 1803 and Windows 8 and before
  - Latest Windows 10 (1803 and later) has a built-in OpenSSH
- Exercise 1: SSH login using the password authentication
- Exercise 2: Generating a keypair and login using the key
- Exercise 3: Disable the password authentication
- Exercise 4: Using agent
- Exercise 5: Secure file copy
- Exercise 6: Allow other users