SSH and keys

Network Startup Resource Center
www.nsrc.org

Passwords are bad!

- A large proportion of security failures are due to passwords
  - Users choose poor passwords
  - Users write them down or share them
  - Passwords can be guessed or brute-forced
  - Passwords can be sniffed or key-logged
  - People hate forced password changes and password complexity tests, and will work around them

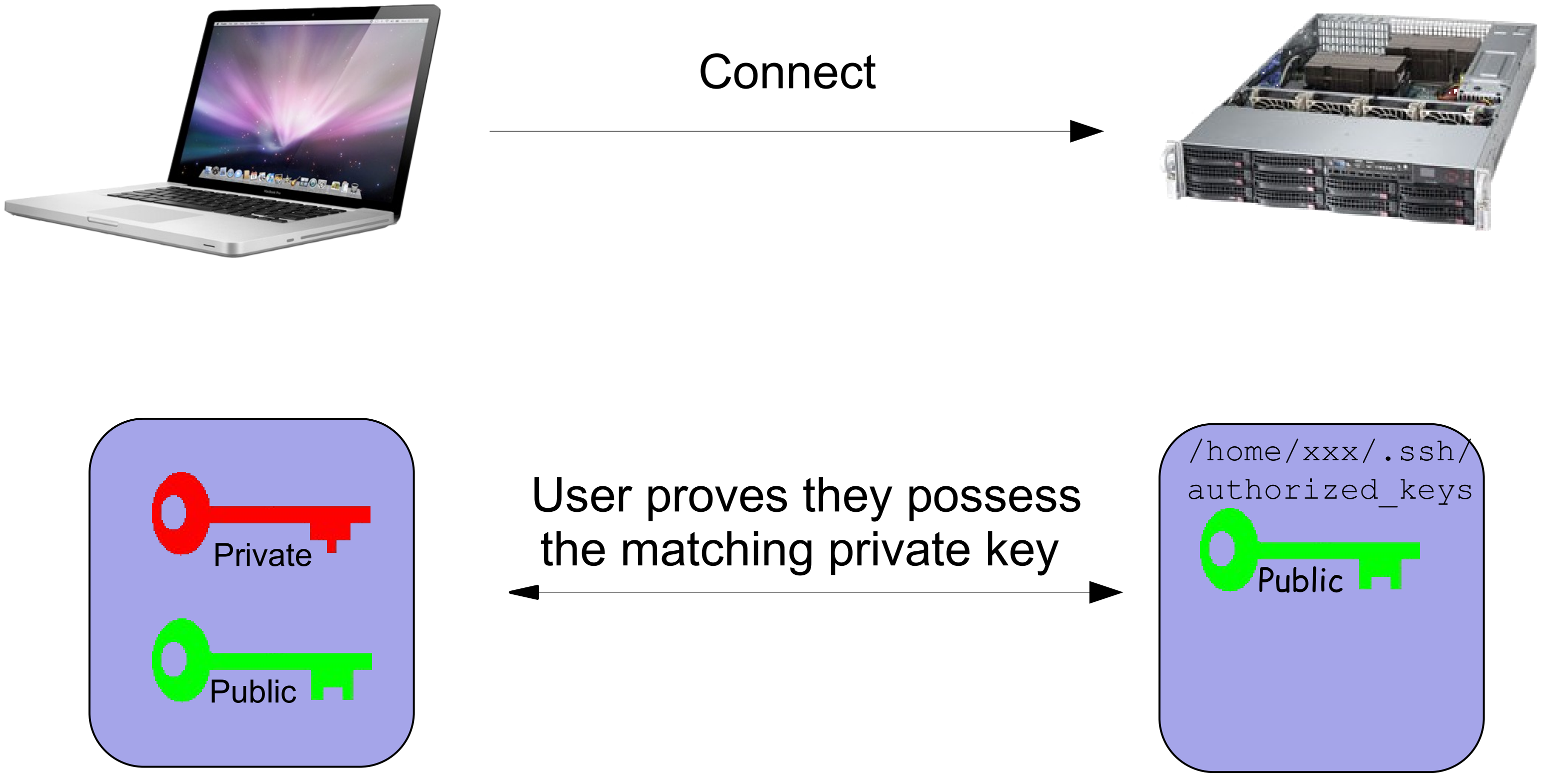
SSH and system administration

- SSH gives you remote command-line access to systems
- Therefore a very attractive target for attackers
- Traffic is **encrypted,** which at least makes it hard to sniff passwords off the network
  - Much better than telnet
- But in addition, SSH allows you to use **cryptographic keys** instead of passwords

Using crypto keys with SSH

1. Generate a Private/Public key pair
2. Copy the public key onto each of the systems you want to be able to log into
   - It goes into `$HOME/.ssh/authorized_keys`
3. Log in with ssh, using your private key to prove your identity to the other system, instead of a password

User authentication with keys

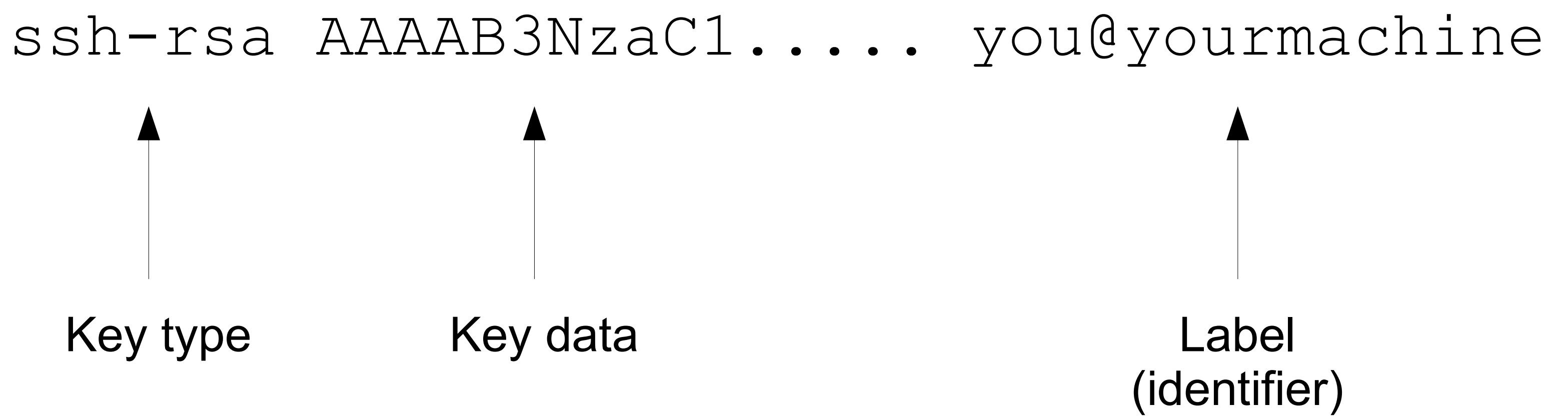

Connect
Private
Public
User proves they possess the matching private key
/home/xxx/.ssh/authorized_keys
Public

Generating a key pair

- This is a one-time operation
- For Windows/putty: use **`puttygen.exe`**
- For Linux and OSX: use **`ssh-keygen`**
- There are three different key types currently: rsa, ecdsa, ed25519
  - ecdsa and ed25519 are newest and fastest, rsa is more widely compatible
  - If you need to use RSA, choose a key length of 2048 or 3072 bits (e.g. -t rsa -b 2048)
- You get a private key and a related public key

OpenSSH public key looks like this

- One very long line of text



```
ssh-rsa AAAAB3NzaC1..... you@yourmachine
```

- Safe for copy-paste (but beware line wrap)
- puttygen has a different native format but can also export the above format

Understand the difference!

- Your **private key** is like the Crown Jewels
- Your **public key** is like a photograph of the Crown Jewels
- Which of these would you be happy to send via the postal service? :-)
- <u>Never</u> give your private key to anyone else
- <u>Never</u> send your private key via E-mail
  - Should you need to transfer it, do so via a secure channel like scp or sftp

Keeping your private key safe

- Keep it on the machine where it was generated
  - usually your laptop
  - plus a secure backup, e.g. USB key in a safe
- Protect it with a strong **passphrase**
- The key is actually stored encrypted on your hard disk; the passphrase decrypts it
- So an attacker would need both to steal the key file **and** know your passphrase
  - "2-factor authentication": something you <u>have</u>, and something you <u>know</u>

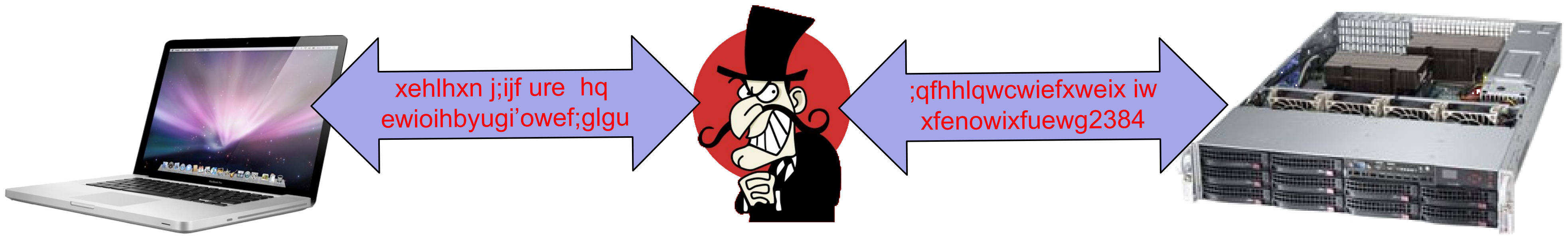
Disabling passwords over SSH

- Once you have key authentication working, you can disable fallback to password auth

```
# editor /etc/ssh/sshd_config

PasswordAuthentication no
ChallengeResponseAuthentication no

PermitRootLogin without-password
-- or --
PermitRootLogin no

# service ssh restart
```

Man-in-the-middle attacks

- How do you know you did not actually connect to someone else, who is decrypting your traffic and re-encrypting it to the remote host?



Host keys

- Solution: the host you are connecting to, also has its own key
- The host proves its own identity to you each time you connect
- The first time you connect, you will be shown the host's "fingerprint" (hash of public key)
  - If you've ever used SSH, even with passwords, you will have seen this prompt
- Future connections will check that the <u>same</u> host key is seen

Host key verification

- If later there is a man-in-the-middle, on connection your ssh client will see the MITM's key instead of the host's key
- It won't match, you will get an error and the connection is dropped
- Questions:
  - What happens if you reinstall the host's OS?
  - What effect might this have on your users?
  - How are you going to deal with it?

Questions?

- Now do the exercise: SSH with public key authentication

SSH Agent

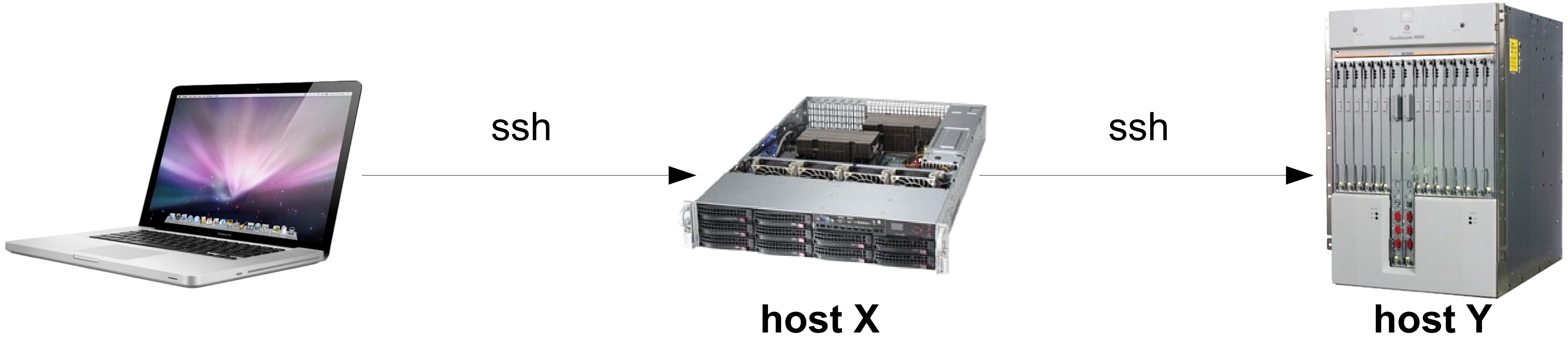
- Having to enter your passphrase every time you log in is tedious
- However there is a simple solution to this: the SSH Agent
- Once you have decrypted your private key once with your passphrase, the Agent keeps the decrypted key in RAM
- Subsequent logins don't prompt you at all
- This makes SSH + keys **very convenient!**

Installing SSH agent

- For Windows/putty: download **`pageant.exe`**
  - Start it
  - Select your private key file
  - Enter your passphrase
- OSX: already has it
- Linux with Unity/Gnome/KDE: already has it

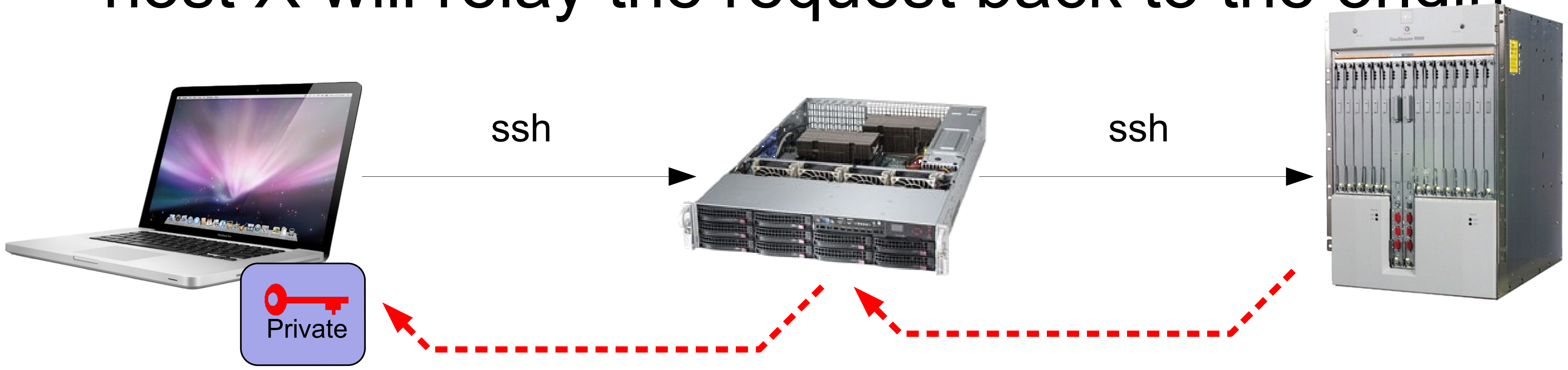
Multi-hop authentication

- Sometimes it is necessary to ssh into host X, and then ssh from host X to host Y
  - e.g. due to network ACLs
  - or because host Y is on a private IP address
  - or because you are running some sysadmin tool on host X which needs to log in to host Y



Agent forwarding

- You may be tempted to copy your private key from your laptop to host X, but DON'T!
- There is a better way: turn on Agent Forwarding when you connect to host X
- Host Y will try to authenticate from host X, and host X will relay the request back to the origin



Summary

- SSH + key is **very secure**
  - Disable password authentication to get max benefit
- SSH + key + agent is **very convenient**
  - Type passphrase just once at start of day
  - No need to type passwords each time you login
  - No need to regularly change passwords across many hosts
  - Agent forwarding permits multi-hop logins
- *You need to deploy this!*

Questions?

- Now do the exercise: SSH with agent