ssh

Secure Shell

Keying, Key Exchange, Session Setup, ssh-agent, and Bastion/Jump Hosts

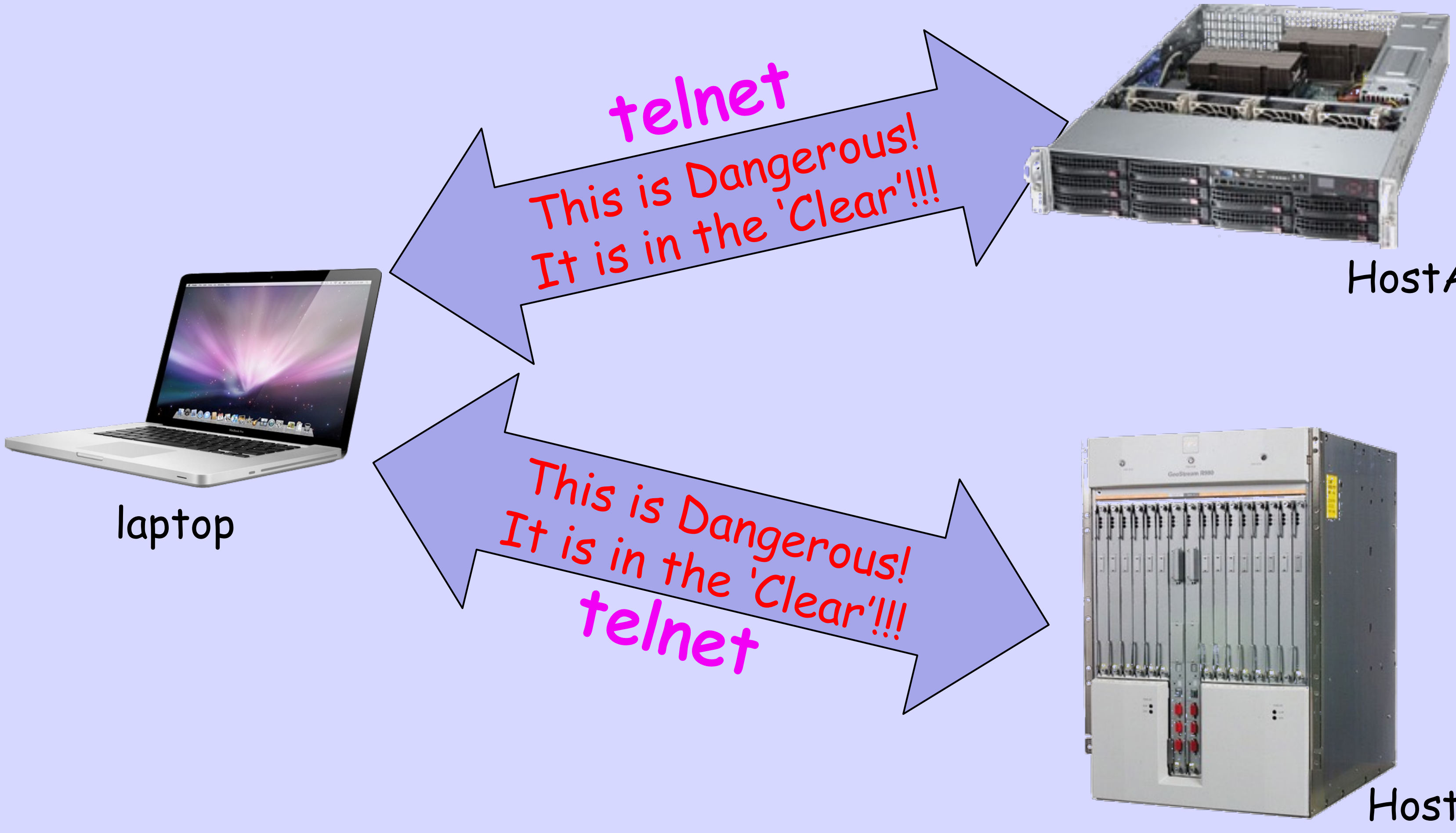
randy, rob, and hans

Communicate Safely with Remote Systems

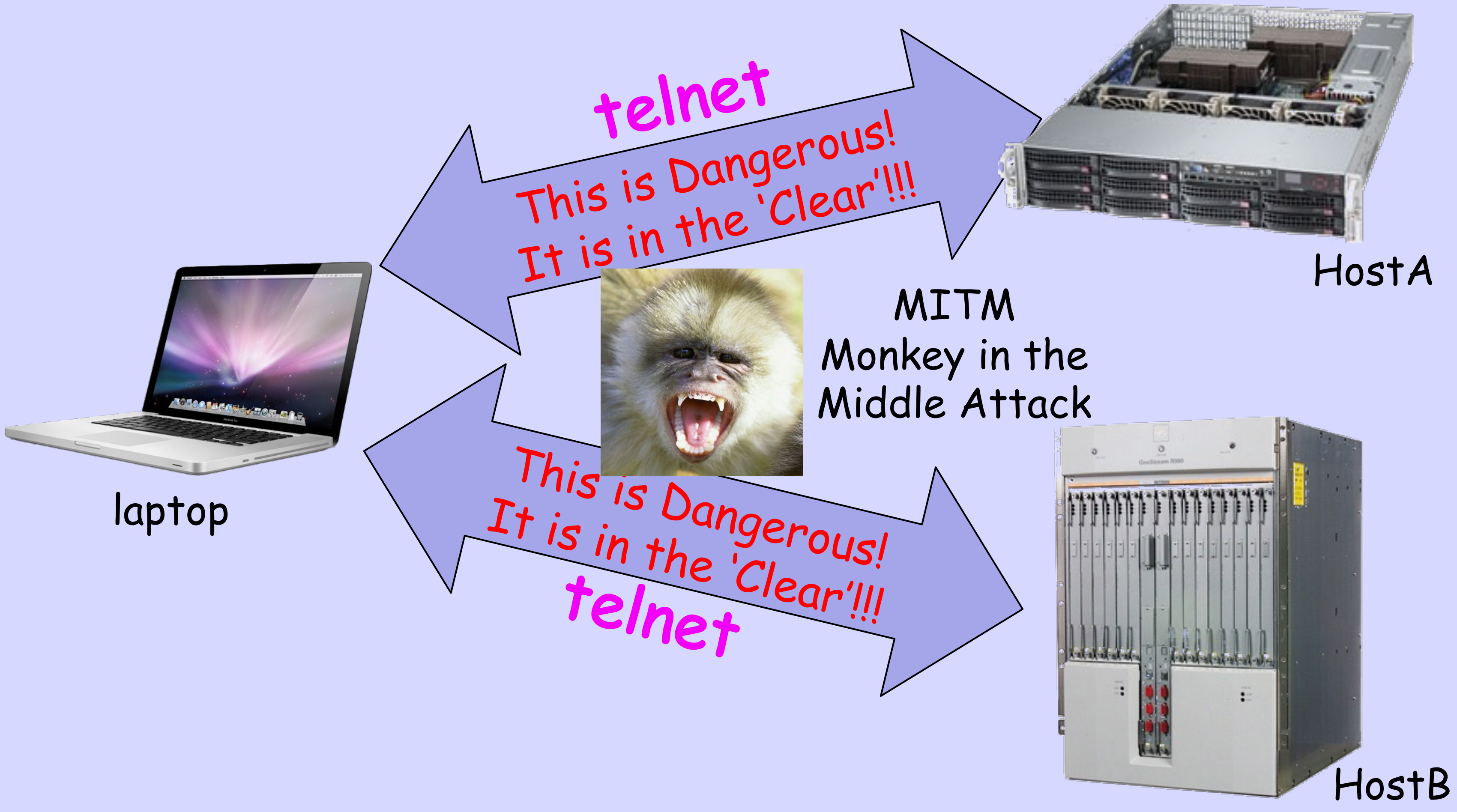
What is *Safely*?

- Authentication – I am assured of which host I am talking with
- Authentication - The host knows who I am
- The traffic is encrypted

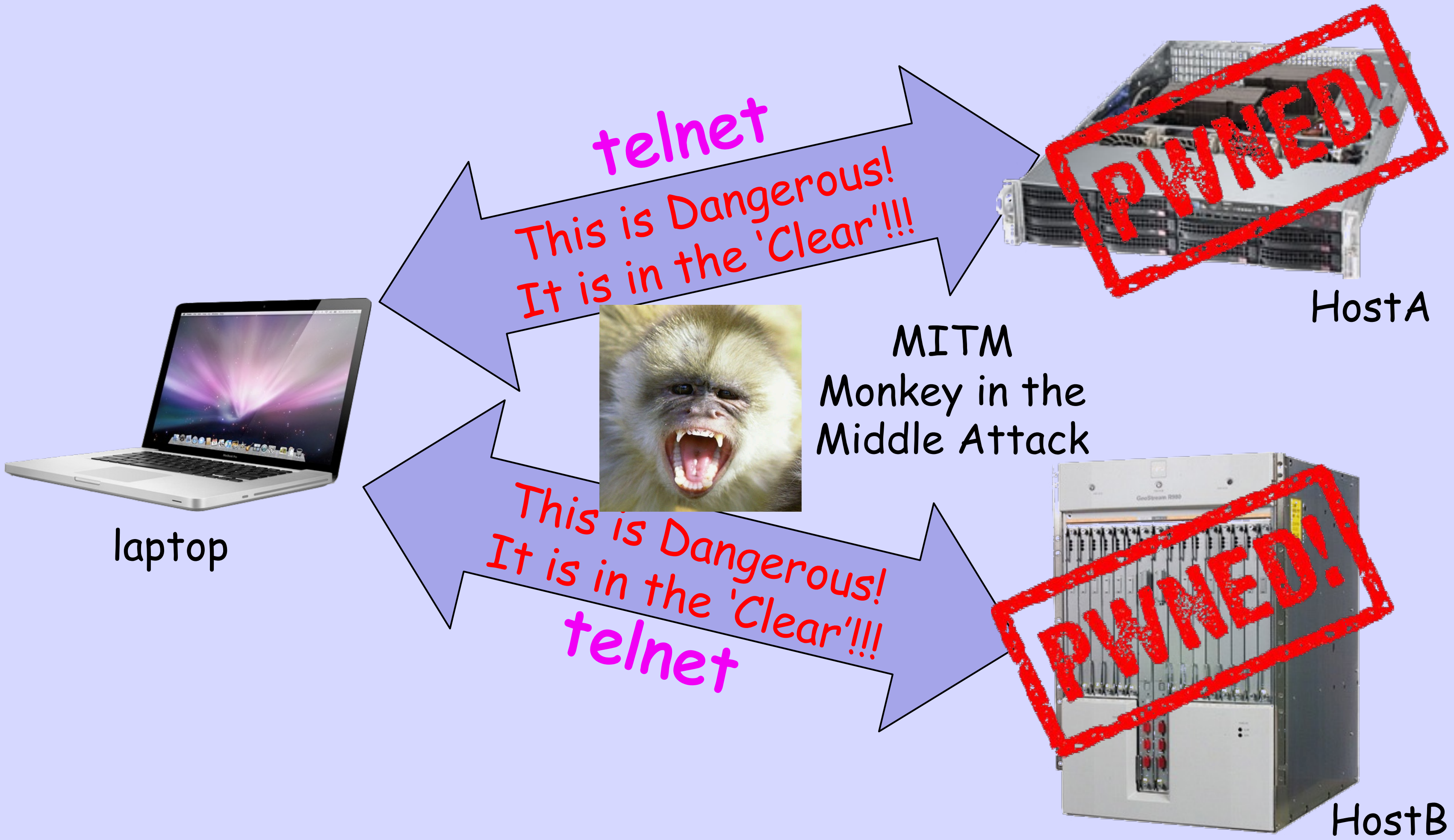
Traditional



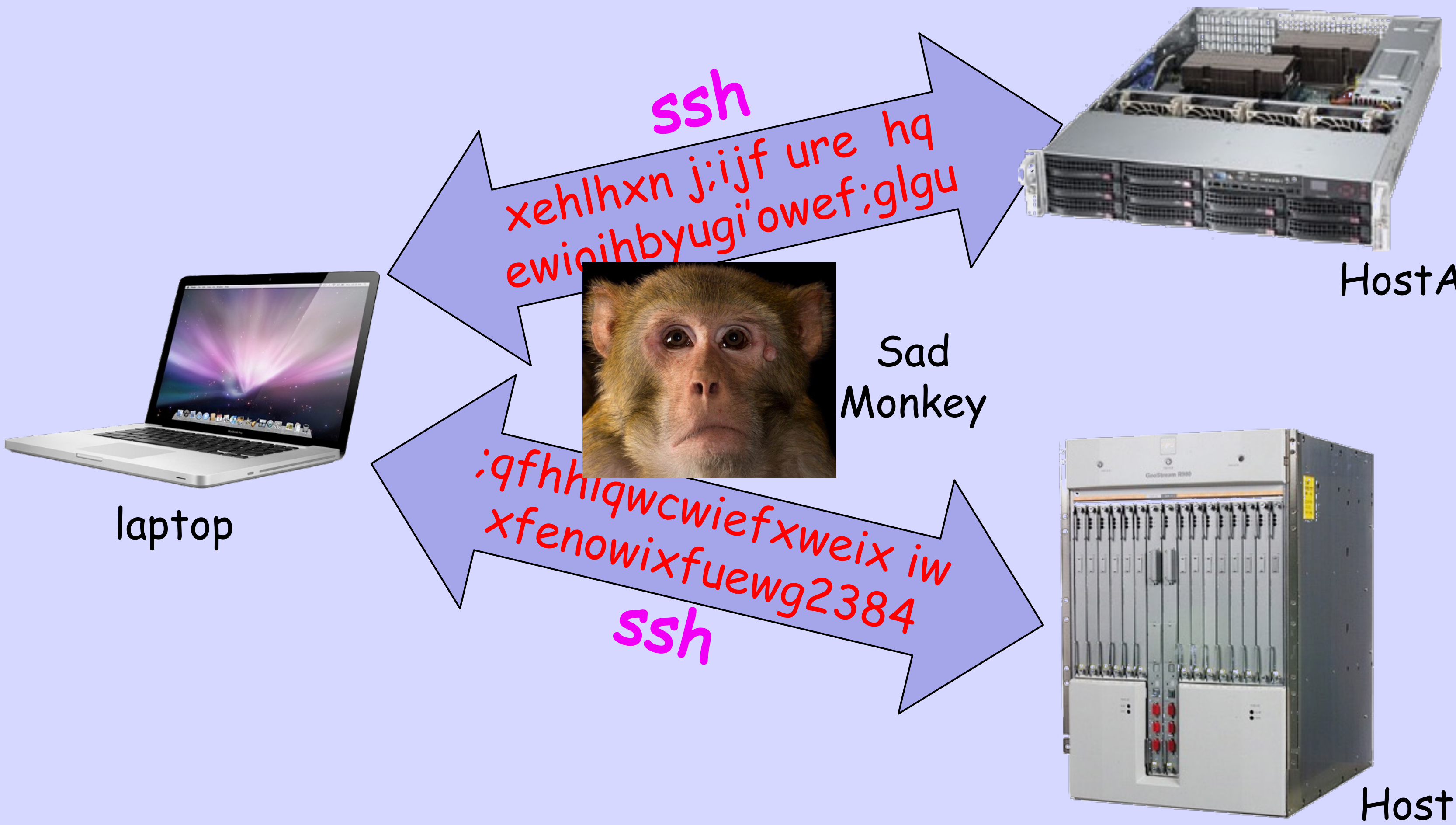
Traditional



Traditional



ssh Encrypts & Authenticates



Secure Shell - ssh

- Provides authenticated and encrypted shell access to a remote host
- But it is much more
- It is used by other protocols, sftp, scp, rsync, …
- Authenticate through Jump Hosts
- Build custom tunnels

Secure Shell - ssh

- Provides authenticated and encrypted shell access to a remote host
- But it is much more
- It is used by other protocols, sftp, scp, rsync, …
- Authenticate through Jump Hosts
- Build custom tunnels

Think of SSH as
a bit like
PGP where the other
end is a computer,
not a human

But PGP is

Object Security

SSH is

Channel/Transport

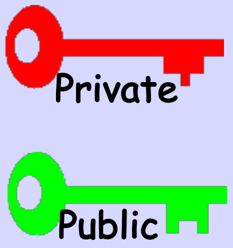
Security

A router or host on the public Internet which uses passwords has a lifetime of approximately 5 minutes

Use Only
Keyed ssh

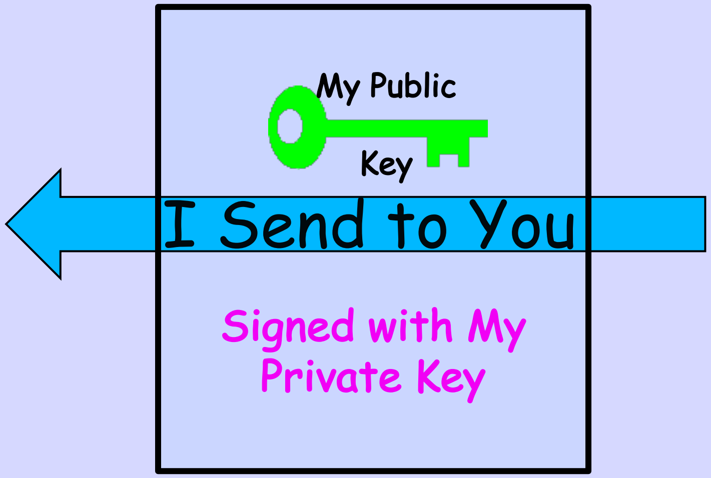
For Authentication, ssh uses *asymmetric*, i.e. public/private key cryptography

If I have a key pair



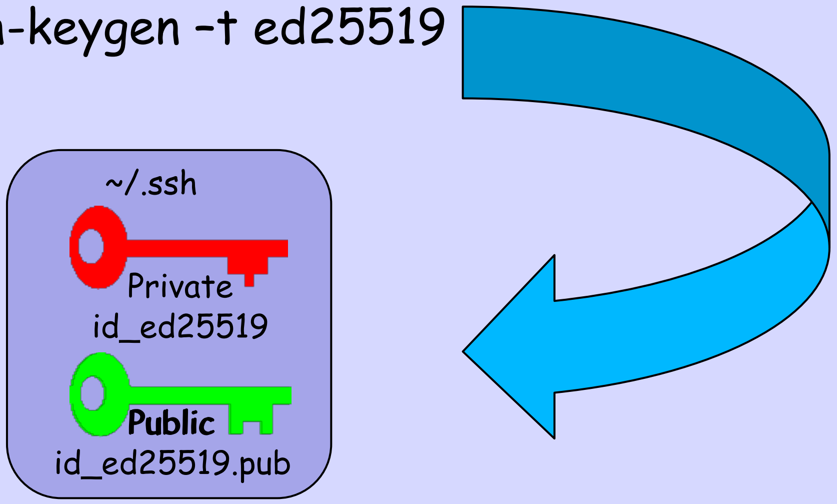
How do I convince you that I have both private and public keys over the public Internet?

Proof of Possession



- Verify signature using my public key
- If it verifies, you know that I must have the matching private key
- And you now have my public key

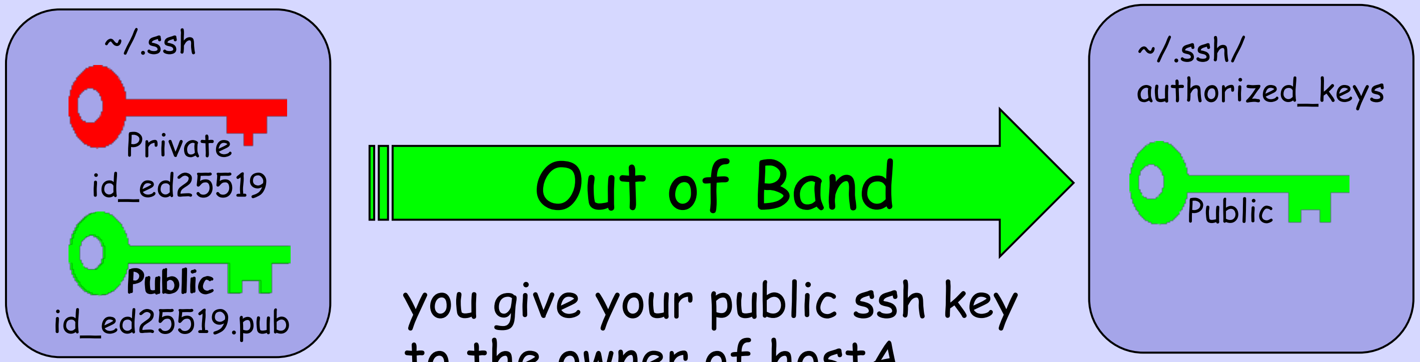
ssh - Key Generation



Public Key on Destination

laptop:

hostA:



you give your public ssh key
to the owner of hostA

Destination has *Host Keys*

laptop:

~/.ssh

Private

id_ed25519

Public

id_ed25519.pub

hostA:

Host keys identify the host

/etc/ssh/…

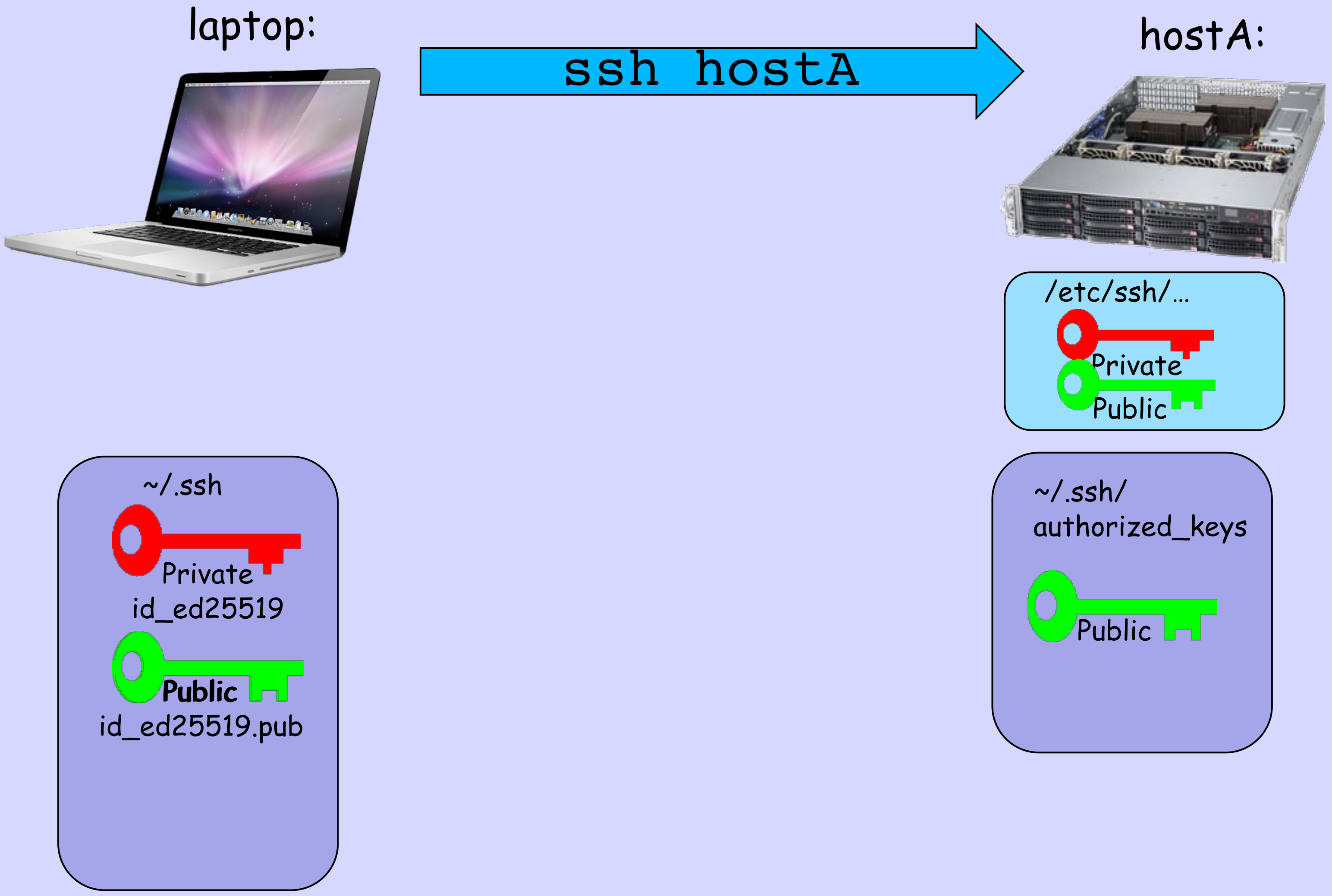
Private

Public

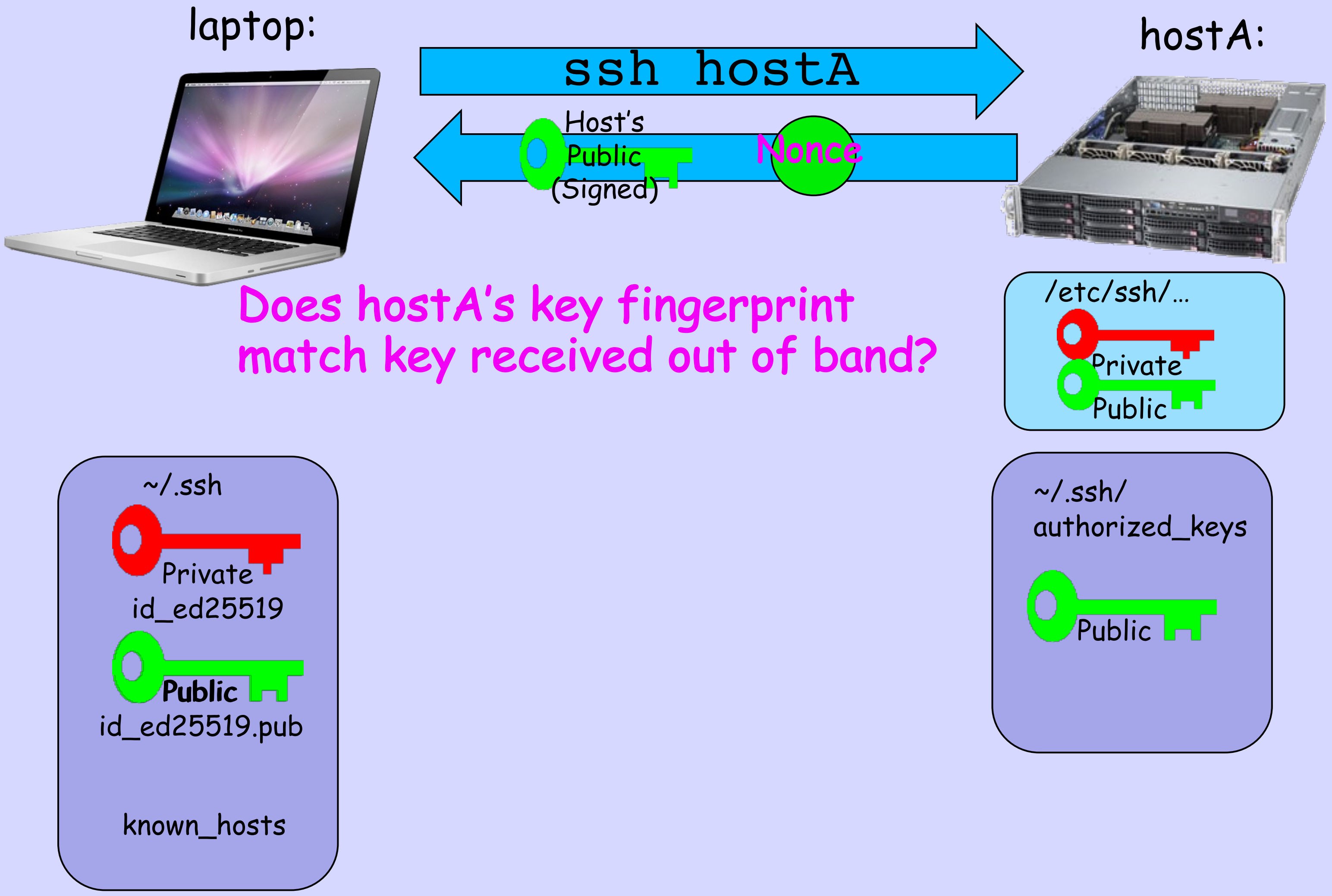
~/.ssh/
authorized_keys

Public

2-Way Authentication



Host's Identity



Does hostA's key fingerprint match key received out of band?

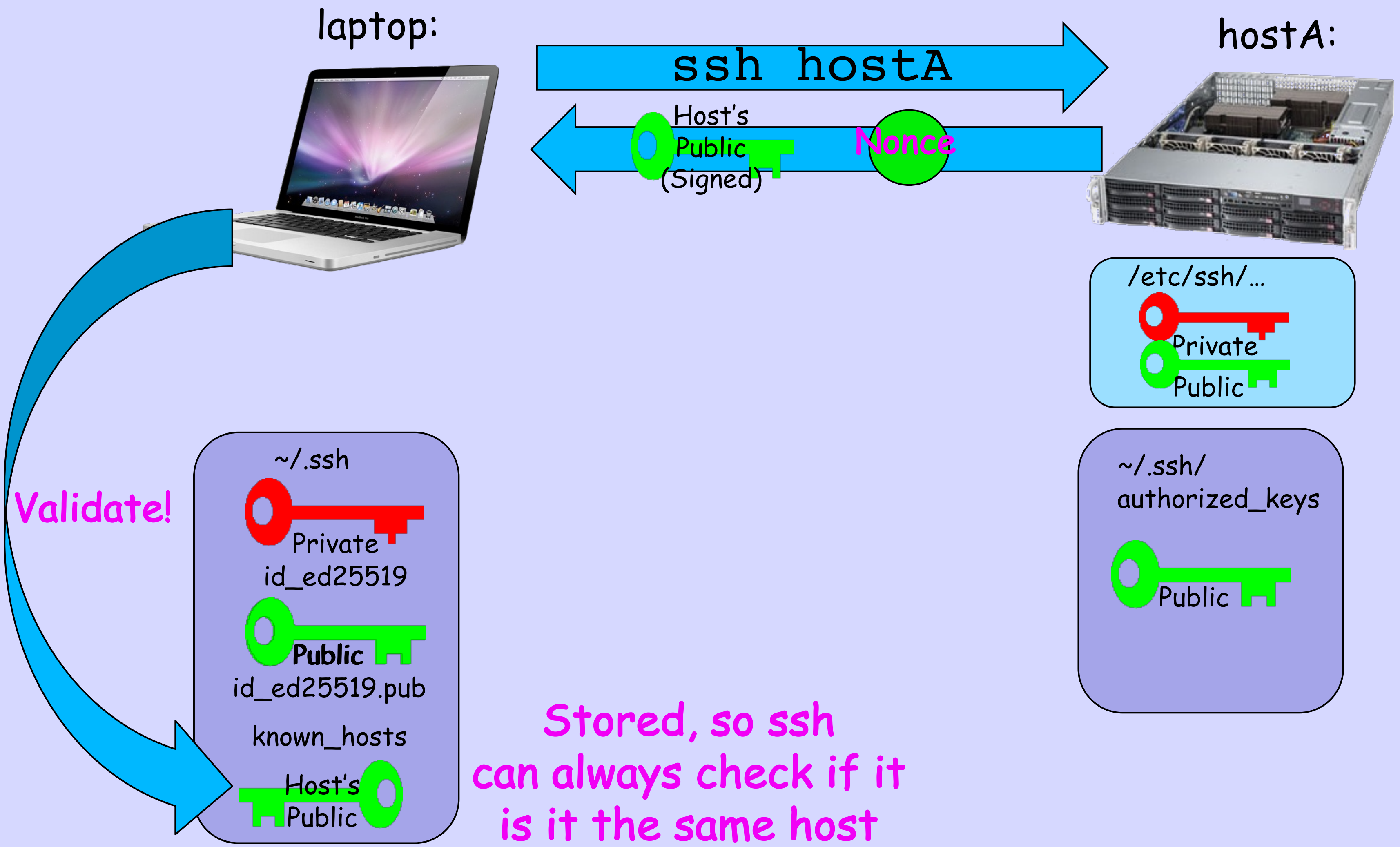
Checking Host's Keys

```
ryuu.rg.net:/Users/randy> ssh adrilankha.hactrn.net
The authenticity of host 'adrilankha.hactrn.net
(2001:418:1::19)' can't be established.
ED25519 key fingerprint is
SHA256:hx5isFS0gpMDFsaZoTs0g6Qo8pgvECmu/c9pDEVS0dM.
This key is not known by any other names
```

And you should check it against what you got out of band

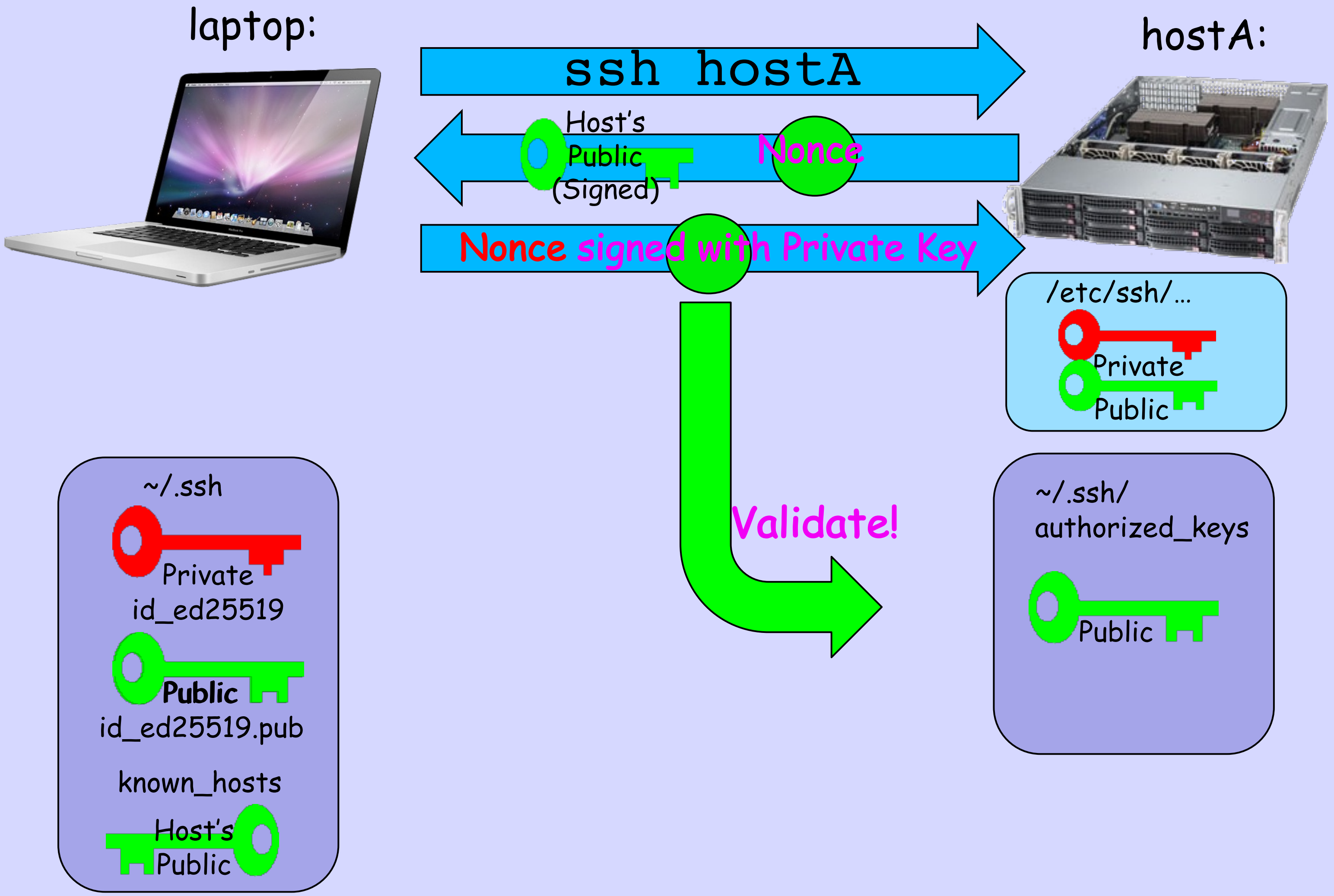
There is a certificate-based ssh system which solves the whole known host identity issue. Next episode ☺

Validate Host's Identity

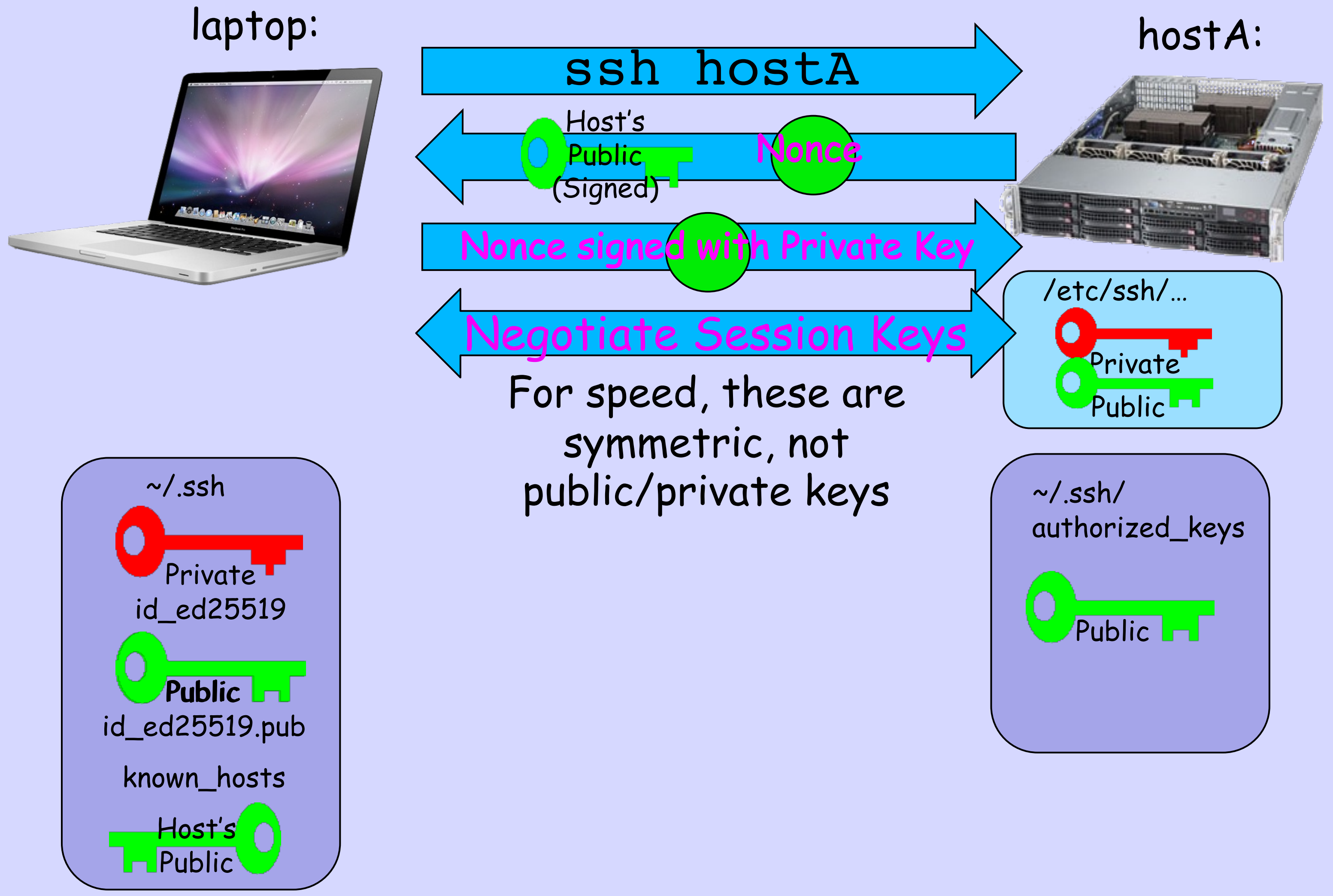


Stored, so ssh can always check if it is it the same host

User Validation

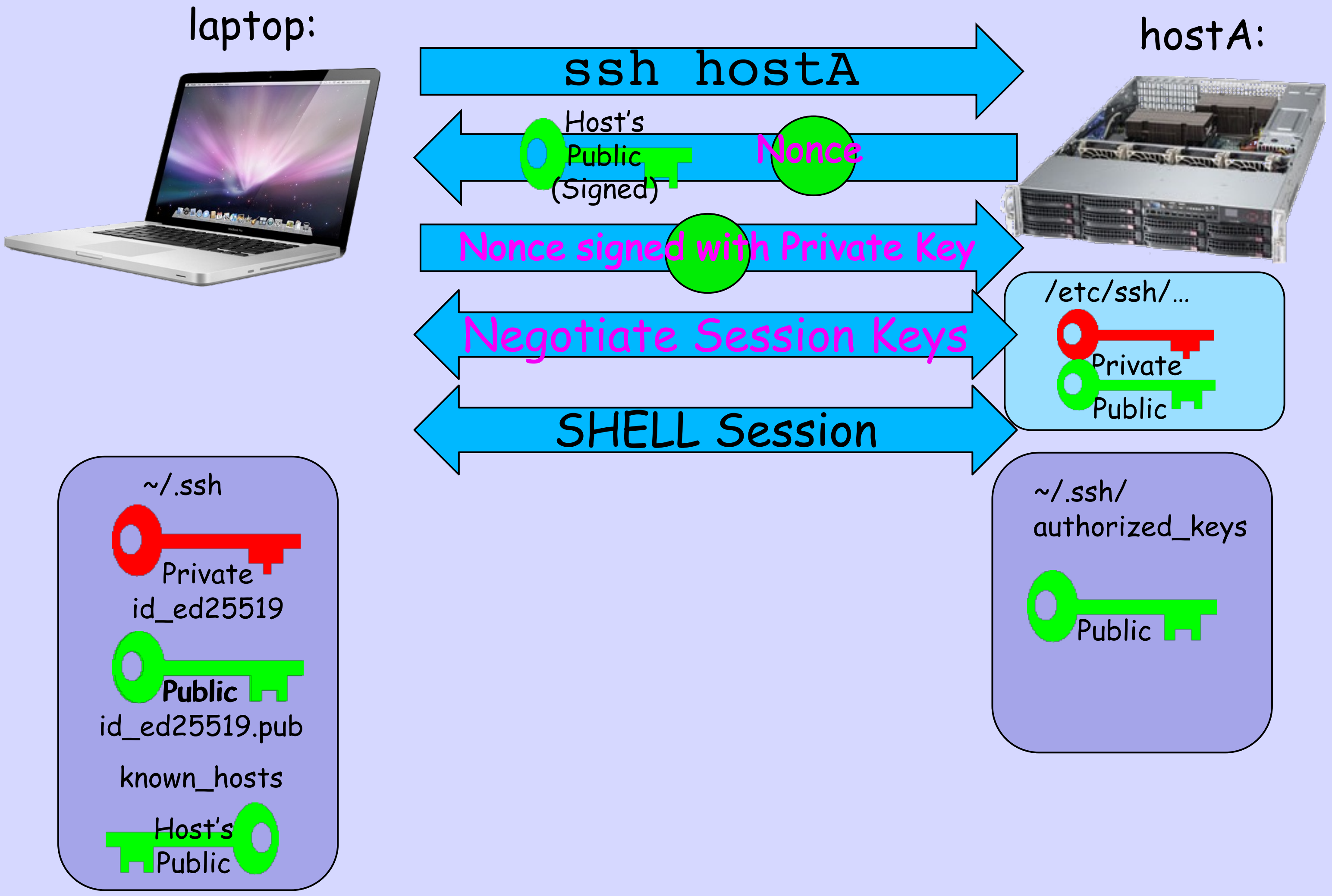


Create Session Key(s)

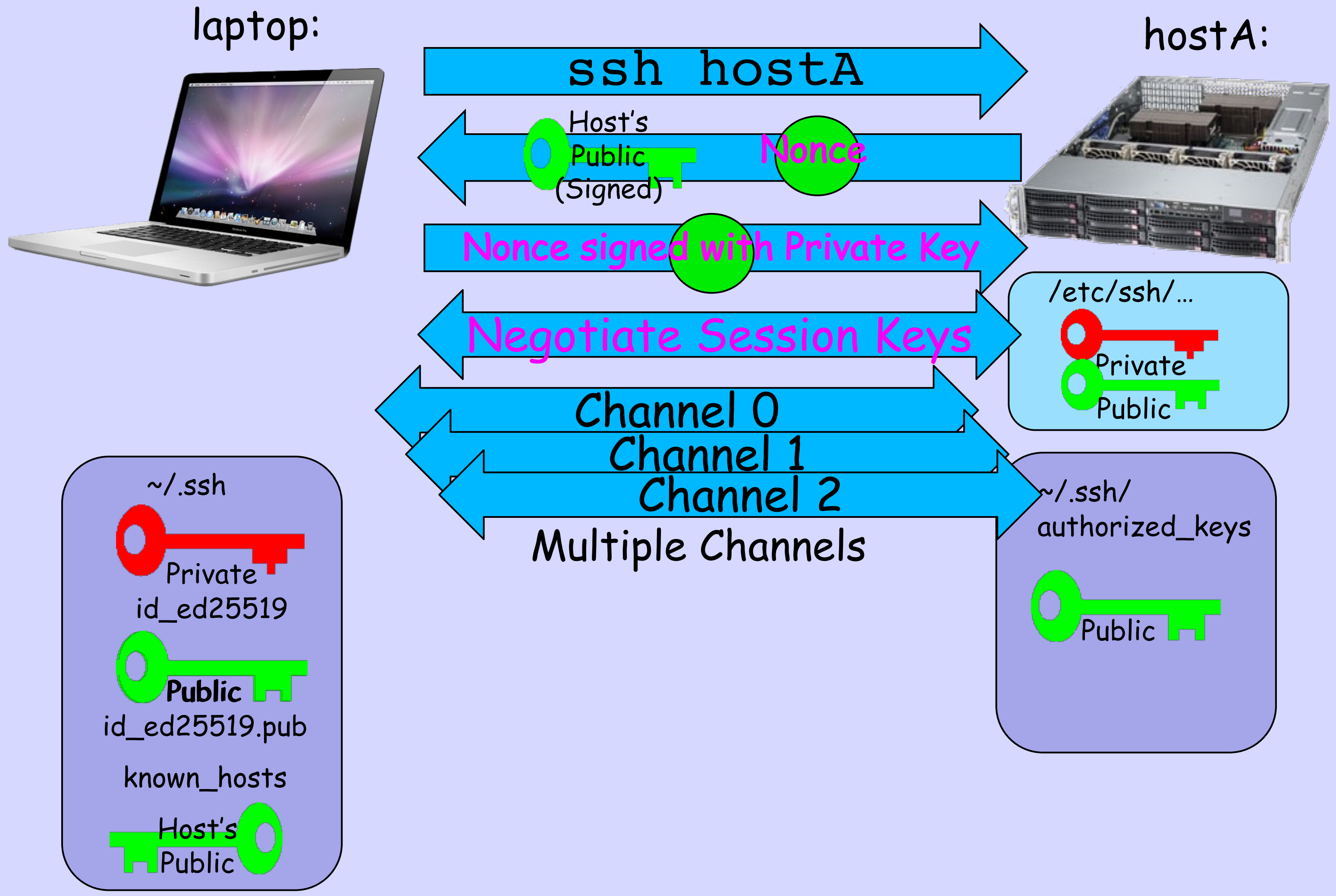


For speed, these are symmetric, not public/private keys

Open Session



Sessions Have Channel(s)



Use Keys Not Passwords

- In `/etc/ssh/sshd_config`
  ```
  PermitRootLogin without-password
  PasswordAuthentication no
  UsePAM no
  ```
- Never store private keys on a multi-user host
- Protect private key with a good passphrase or a hardware token
- Store private key ONLY on your laptop and protect your laptop (encrypt disk!)

The only compromise we have had to our infrastructure was a researcher who stored their private key on a university server

Private Key Protection

FreeBSD repository compromise some years ago (think supply chain)

"The compromise is believed to have occurred due to the leak of an SSH key from a developer who legitimately had access to the machines in question, and was not due to any vulnerability or code exploit within FreeBSD."

ssh-agent

- ssh-agent remembers your decoded key

- When you log in to your laptop

```
ryuu.rg.net:/Users/randy> ssh-add ~/.ssh/id_ed25519
Enter passphrase for .ssh/id_ed25519:
Identity added: .ssh/id_ed25519 (randy@ryuu.psg.com)

ryuu.rg.net:/Users/randy> ssh-add ~/.ssh/id_rsa
Enter passphrase for .ssh/id_rsa:
Identity added: .ssh/id_rsa (randy@ryuu.psg.com)
```

Dangerous Mac Hacks

```
ryuu.rg.net:/Users/randy> head .ssh/config
#
Host *
   ForwardAgent    yes
   AddKeysToAgent  yes
   UseKeychain     yes        # Dangerous!
```

```
ryuu.rg.net:/Users/randy> ssh-add -K .ssh/id_ed25519 # -K is a Macism
Enter passphrase for .ssh/id_ed25519:
Identity added: .ssh/id_ed25519 (randy@ryuu.psg.com)

ryuu.rg.net:/Users/randy> ssh-add -A   # -A is a Macism
Identity added: /Users/randy/.ssh/id_ed25519 (randy@ryuu.psg.com)
Identity added: /Users/randy/.ssh/id_rsa (randy@ryuu.psg.com)
```

OK, that's Deprecated Because It's a Macism

```
WARNING: The -K and -A flags are deprecated and have been replaced
         by the --apple-use-keychain and --apple-load-keychain
         flags, respectively.  To suppress this warning, set the
         environment variable APPLE_SSH_ADD_BEHAVIOR as described in
         the ssh-add(1) manual page.
```

I promise to update my habits ☺

And we really should not use KeyChain

Agent Forwarding

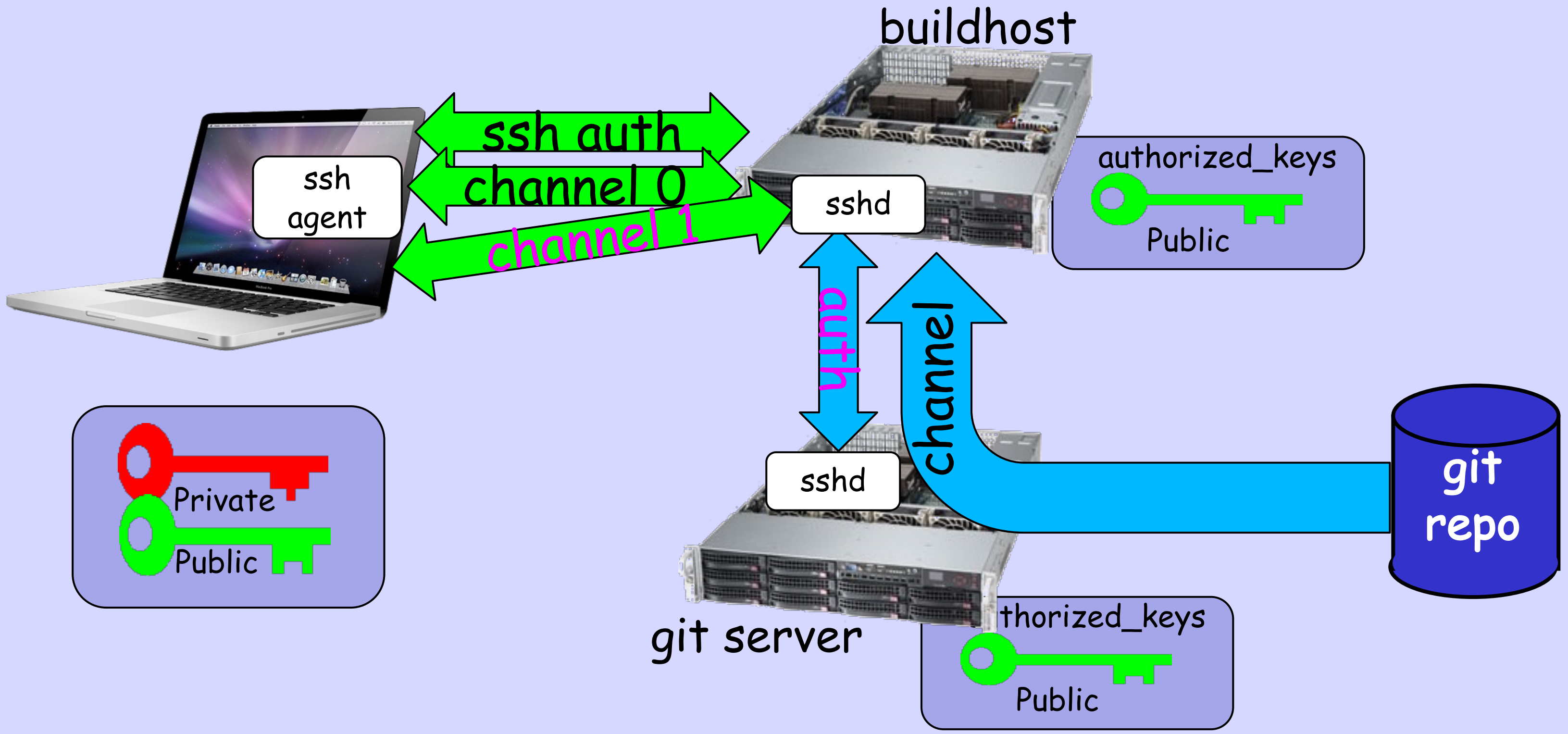
```
laptop:~$ ssh buildhost

buildhost:~$ git clone git@github
```

What is actually happening here?

Agent Forwarding

allows your local ssh-agent to reach through a channel of an SSH connection and transparently authenticate on a remote server



Jump/Bastion Host

- Sometimes you can not reach the host you want directly
- I.e., you need to go through hostA to get to hostB
- hostA is called a *Bastion* or *Jump* host
- But you never want your private keys to leave your laptop!

no ssh-agent forwarding

```
laptop:~$ ssh -A jumphost.com
jumphost:~$ ssh targethost.com
targethost:~$
```

-A   for when client does not have ssh agent forwarding enabled. I.e.

in laptop:~/.ssh/config

```
ForwardAgent no    # often default!
```

With ssh-agent Forwarding

```
laptop:~$ ssh jumphost.com
jumphost:~$ ssh targethost.com
targethost:~$
```

- ssh-agent forwarding handles all the authentication work
- But use ssh-agent forwarding only through trusted jump hosts

Authentication

```
laptop:~$ ssh jumphost.com
jumphost:~$ ssh targethost.com
targethost:~$
```

laptop:~/.ssh/known_hosts must know jumphost's key

jumphost:~/.ssh/known_hosts must know targethost's key

jumphost:~/.ssh/authorized_keys must have my public key

targethost:~/.ssh/authorized_keys must have my public key

Shorthand

```
laptop:~$ ssh -J jumphost.com targethost.com

targethost:~$
```

jumphost must be able to resolve targethost.com in the DNS

and ssh-agent forwarding is still your friend

Multi-Hop

```
laptop:~$ ssh -J jump0,jump1 targethost.com

targethost:~$
```

and this can get as long as you want
if ssh forwarding is enabled

Make it Automagic

```
laptop:~$ cat ~/.ssh/config

Host *
  AddKeysToAgent yes

Host targethost.com
  JumpHost jumphost.com

Host *.internal.com
  JumpHost otherjumphost.com
```

Magic

So now one can

```
ssh targethost.com
```

and

```
ssh foo.internal.com
```

Take-Aways

- ssh-agent is a very valuable friend
- ~/.ssh/config makes it easy
- Never send your *private* key in email. They need your *public* key
- Your private key(s) MUST NEVER LEAVE YOUR LAPTOP!