SSH keys

Dr Peadar Grant

February 21, 2024

1 SSH authentication

Available methods:

Password typed on demand (as you've used so far).

Key pair using cryptographic key files (today's focus!)

- Some hardware crypto devices use a hardware private key.

2-Factor Authentication using authenticator app.

One-Time Password from a printed list.

Kerberos / **GSSAPI** to pass through credentials (advanced!)

Which methods are *required* and *sufficient* will depend on how the SSH server and the operating system on the remote host are configured!

2 Key-based authentication

SSH key pairs are an alternative or addition to a username/password. They consist of:

Private key kept on the client and securely stored.

Public key on the server(s) you want to log in to. (The public key can be freely shared around, even put up in public.)

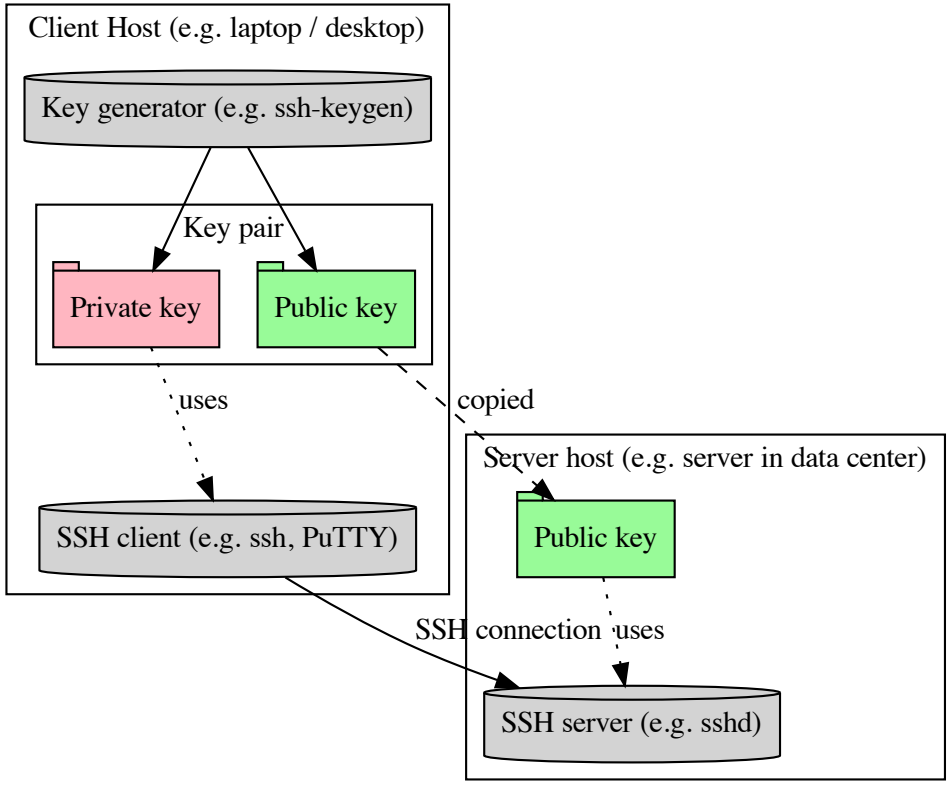

Figure 1: Key-based authentication

2.1 Motivation

Main reasons to use a public / private key pair:

- Avoids repetitive typing of password when logging into a host.
- Provides security against *brute force* password attacks.

There are also some reported security downsides to SSH Key Pairs!

- No universal means to revoke keys if a private key is stolen.

2.2 Encryption algorithms

There are a number of encryption algorithms:

RSA: at least 4096-bits

ED25519: highly recommended!

DSA: not recommended anymore

ECDSA: not recommended anymore

You don't need to worry too much about their mechanics. I generally choose `ed25519`.

2.3 Creating key pair (mac, linux, windows 10 onwards)

Key pairs are normally and ideally created on your own local client computer. Key pairs only need to be generated once.

To create an ED25519 key pair, in Powershell/Bash type:

```
ssh-keygen -t ed25519
```

Just press enter for now at each prompt.

2.4 Default key locations

The **key pair** is stored in **two** files (same for Mac, Linux, Windows). You can find them by changing into the `.ssh` direcxstory and listing the contents of it:

```
cd .ssh
ls
```

From the directory listing:

```
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        16/10/2020     15:19           3243 id_ed25519 <-- PRIVATE
-a----        16/10/2020     15:19            749 id_ed25519.pub <-- public
```

You can of course copy these files to/from a memory stick or online storage. Remember though that if your private key is compromised, anybody can use it.

2.5 Connecting over SSH with keys

In PowerShell/Bash we can use the SSH command to connect to the SSH server on a remote host:

- This will then present us with a new shell on the remote computer (Bash for Linux/UNIX, PowerShell for Windows).
- By default, SSH will try all private keys in `.ssh` so we don't need to specify which.

```
ssh student@$publicIp
```

All you will notice is that you're not asked for a password.

2.6 Using specific non-default key

We can force the use of a specific key using the `-i` option:

```
ssh -i private_key_file_name_here username@host
```

This is useful if a key is kept on a USB stick or network device.

3 Encrypted private keys

You can optionally encrypt the private key file with a passphrase (aka password) when creating it.

The passphrase will be needed any time the private key file is used.

3.1 SSH agent

The SSH agent is a program that allows a private key to be decrypted once and used a number of times while logged in.

4 Private key security

Your private key is valuable!

Important points:

- If an unauthorized person has your private key, they can log into any server that your public key is on.
- Best to protect your private key using a passphrase (where possible!)
  - Alternatives like private key on full disk encrypted laptop ok too!

4.1 Risk factors

Your private key is more at risk of compromise when the following risk factors apply:

- The private key file has **no passphrase** to protect it if it falls into the wrong hands.
- There are copies of your private key in a large number of locations.
- You use your private key (and passphrase) on client systems that you don’t trust.