How and Why to Use an SSH Key

Stop typing passwords, and make the connection safer at the same time

SSH series, part 2 of 3. Assumes you have read "What SSH Is and Why You'd Use It." Next: SSH Shortcuts.

In part 1 you connected to a server by typing a password. That works, and it is also the weakest link in the whole setup. This tip replaces the password with an SSH key, which is both more secure and less typing. It is, as the professionals say, how you are supposed to do this.

Why a key beats a password

A password is a short secret you type, and it crosses the connection every time you log in. It can be guessed, brute-forced by a machine trying millions of combinations, phished, or reused from some other site that already got breached.

An SSH key is different. It is a pair of very long cryptographic files, far too large to guess, and you prove who you are by *holding* one of them rather than by sending a secret across the wire. Set it up once and the server stops asking for a password at all. More secure and more convenient at the same time, which is rare.

What a key pair actually is

When you make an SSH key, you get two files that work only as a matched set:

- The **private key** (named `id_ed25519`) stays on your laptop. You never share it, never copy it anywhere, never let it leave your machine. This file is the thing that proves the connection is really you.
- The **public key** (named `id_ed25519.pub`, the same name with `.pub` on the end) is safe to hand out. You place a copy of it on every server you want to log into.

Think of the public key as a padlock you can make endless copies of and bolt onto other people's doors, and the private key as the single key in your pocket that opens all of them. Handing out the padlock gives nothing away; the private key is the whole game.

Step 1: make the key

Open a terminal (Terminal on a Mac, PowerShell on Windows) and run:

Educational only. Not legal advice. No attorney-client relationship. ©2026 Chariton Media.

```
ssh-keygen -t ed25519 -C "your name or email"
```

- `-t ed25519` chooses a modern, strong type of key. Use it and do not overthink it.
- `-C "your name or email"` adds a label so you can tell your keys apart later. Anything readable works.

It asks where to save the key; press Enter to accept the default (`~/.ssh/id_ed25519`). Then it asks for a passphrase. **Set one.** The passphrase encrypts the private key on your disk, so if your laptop is lost or stolen, the thief has an encrypted file they cannot use rather than a working master key to your servers. For a lawyer, that is not optional.

When it finishes, you have two new files. Confirm them:

```
ls ~/.ssh          # you should see id_ed25519 and id_ed25519.pub
```

Step 2: put the public key on the server

Now copy the public key up to a server so it will recognize you.

On **Mac and Linux** there is a command for exactly this:

```
ssh-copy-id you@198.51.100.23
```

It asks for your server password this one last time, then adds your public key to the server's list of allowed keys.

Windows has no `ssh-copy-id`. Send the public key up by hand instead:

```
type $env:USERPROFILE\.ssh\id_ed25519.pub | ssh you@198.51.100.23 "cat >>
~/.ssh/authorized_keys"
```

Either way, you are appending your public key to a file on the server named `authorized_keys`, which is simply the server's list of "these keys are allowed in."

Step 3: connect, no password

Try it:

```
ssh you@198.51.100.23
```

The server no longer asks for its password. If you set a passphrase (you did), your own machine asks for *that* instead, to unlock your private key. That difference is the whole point: the passphrase unlocks a file that never leaves your laptop, while the old password had to travel to the server every single time.

Educational only. Not legal advice. No attorney-client relationship. ©2026 Chariton Media.

The part the tutorials skip: the private key is the credential

Here is the lawyer-specific catch. Once you copy your public key onto several servers, your one private key is the thing that opens all of them. It is a master key to your practice's infrastructure, and it deserves that level of care.

- **Never let the private key travel.** Do not copy `id_ed25519` onto a server, into a shared drive, into a git repository, or an email. Only the `.pub` file ever moves.
- **Keep the passphrase on it**, so a lost laptop is a lost encrypted file, not a lost master key.
- **Back it up somewhere encrypted.** If you lose the private key, you lose key-based access to every server that trusts it, and you are back to passwords until you generate and distribute a new one.
- **If a device is lost or stolen**, remove that public key from your servers' `authorized_keys` files and generate a fresh pair. The public key you handed out is easy to revoke; that is by design.

Where to go next

You can now connect to a server with a single keystroke and no password. The last tip in this series, "SSH Shortcuts," gives that connection a short name, so `ssh you@198.51.100.23` becomes `ssh main`.

The playbook

```
ssh-keygen -t ed25519 -C "you@firm"      # make the key pair, and set a
passphrase
ssh-copy-id you@server-address            # put the public key on the server
(Mac / Linux)
ssh you@server-address                    # connect with the key, no server
password
```

Make the key once, copy the public half to each server, and you never type a server password again.

Educational only. Not legal advice. No attorney-client relationship. ©2026 Chariton Media.