Generating an `ed25519-sk` SSH Key with a YubiKey

Workflow: Generating an `ed25519-sk` SSH Key with a YubiKey

FIDO2-backed SSH key. The private key never leaves the token; what lands on disk is only a *key handle* that is useless without the YubiKey.

Covers key generation on **Linux/macOS** (§2) and **Windows 11 Pro** (§3).

---

0. Prerequisites

Component	Requirement	Check
OpenSSH client	≥ 8.2 (`-sk` types), ≥ 8.4 for `verify-required`; on Windows ≥ 8.9.0.0	`ssh -V`
OpenSSH server	≥ 8.2 to accept, ≥ 8.4 to enforce `verify-required`	`sshd -V` / `ssh -Q key`
FIDO middleware	libfido2 + `ssh-sk-helper` (Unix), `webauthn.dll` (Windows)	`ssh -Q key | grep sk-`
YubiKey firmware	**≥ 5.2.3** for Ed25519 over FIDO2 (older → only `ecdsa-sk`)	`ykman info`
FIDO2 application	enabled, PIN set	`ykman fido info`

Debian/Ubuntu:

```
apt install openssh-client libfido2-1 yubikey-manager
```

Confirm the algorithm is actually available in your build:

```
ssh -Q key | grep sk-
# expect: sk-ssh-ed25519@openssh.com
#         sk-ssh-ed25519-cert-v01@openssh.com
```

If `sk-ssh-ed25519@openssh.com` is missing, the client was built without FIDO support — no amount of flags will help.

---

1. Set a FIDO2 PIN

Required for `verify-required`. The FIDO2 PIN is independent of the PIV PIN and the OpenPGP PIN.

```
ykman fido info                 # shows whether a PIN is already set
ykman fido access change-pin    # sets or changes it
```

On Windows this can also be done natively: **Settings → Accounts → Sign-in options → Security key → Manage**.

- PIN length: 4–63 characters (UTF-8 code points on firmware ≥ 5.7).
- 3 consecutive wrong entries → token must be re-inserted.
- 8 wrong entries total → FIDO2 application locks; recovery requires `ykman fido reset`, which **destroys every FIDO2 credential on the token**, including WebAuthn passkeys.

---

2. Generate the key (Linux / macOS)

2a. Non-resident (default; recommended)

The key handle lives in `~/.ssh/`, the secret in the token.

```
ssh-keygen -t ed25519-sk \
  -O verify-required \
```

```
  -O application=ssh:karger-oliver-yk1 \
  -C "oliver@karger.lan yk1 $(date +%F)" \
  -f ~/.ssh/id_ed25519_sk_yk1
```

Touch the YubiKey when it blinks. You will also be prompted for the FIDO2 PIN, and afterwards for a passphrase on the handle file — set one; it costs nothing and adds a layer if the handle is stolen alongside the token.

2b. Resident / discoverable

The credential is stored *on* the token and can be recovered onto any machine. Convenient for roaming admins, but it consumes a discoverable-credential slot and means the token alone is sufficient to reconstruct the handle (PIN still required).

```
ssh-keygen -t ed25519-sk \
  -O resident \
  -O verify-required \
  -O application=ssh:karger-oliver-yk1 \
  -O user=oliver \
  -C "oliver@karger.lan yk1 resident" \
  -f ~/.ssh/id_ed25519_sk_yk1
```

Slot budget: **25** discoverable credentials on firmware 5.2.3–5.6, **100** on firmware 5.7+.

`-O application=` must start with `ssh:` and is what distinguishes multiple resident keys on the same token. Pick a scheme and stick to it, e.g. `ssh:karger-<user>-<token-serial>`.

Option reference

Option	Effect
`-O resident`	store discoverable credential on the token
`-O verify-required`	PIN **and** touch on every authentication
`-O no-touch-required`	touch not required — only for unattended automation, and the server must allow it
`-O application=ssh:NAME`	credential label / RP ID
`-O user=NAME`	FIDO2 user handle, shown in `ykman fido credentials list`
`-O challenge=FILE` / `-O write-attestation=FILE`	produce an attestation statement (see §8)

Inspect the result

```
ssh-keygen -l -f ~/.ssh/id_ed25519_sk_yk1.pub
ykman fido credentials list          # resident keys only
```

---

3. Generate the key (Windows 11 Pro)

The commands are the same, but the FIDO plumbing is different: Win32-OpenSSH gained FIDO support in **8.9.0.0** and, in a non-elevated session, routes everything through Windows' `webauthn.dll`. That changes where prompts appear and which subcommands work.

3a. Install / verify the OpenSSH client

The client ships as a Feature-on-Demand, not preinstalled. In an **elevated** PowerShell:

```
Get-WindowsCapability -Online | Where-Object Name -like 'OpenSSH*'
Add-WindowsCapability -Online -Name OpenSSH.Client~~~~0.0.1.0
```

Then check the version and algorithm support:

```
ssh -V
# Windows 11 24H2 ships OpenSSH_for_Windows_9.5p2 — well above the 8.9 floor


ssh -Q key | Select-String sk-
```

If the build is older than 8.9.0.0 (older Windows 11 releases, or a locked-down image), install the current MSI from the Win32-OpenSSH releases instead:

```
msiexec /i OpenSSH-Win64-v9.x.x.x.msi ADDLOCAL=Client
```

Note that the FoD binaries in `C:\Windows\System32\OpenSSH\` and an MSI install in `C:\Program Files\OpenSSH\` can coexist — check `where.exe ssh` to see which one you are actually invoking.

3b. YubiKey tooling

```
winget install Yubico.YubiKeyManagerCLI     # ykman
winget install Yubico.Authenticator         # GUI, incl. FIDO2 PIN + credential management
```

Set the FIDO2 PIN before generating (see §1) — either with Yubico Authenticator or via Windows Settings.

3c. Generate

Non-resident, PowerShell (backtick = line continuation):

```
ssh-keygen -t ed25519-sk `
  -O verify-required `
  -O application=ssh:karger-oliver-yk1 `
  -C "oliver@karger.lan yk1" `
  -f "$env:USERPROFILE\.ssh\id_ed25519_sk_yk1"
```

Resident:

```
ssh-keygen -t ed25519-sk `
  -O resident `
  -O verify-required `
  -O application=ssh:karger-oliver-yk1 `
  -O user=oliver `
  -f "$env:USERPROFILE\.ssh\id_ed25519_sk_yk1"
```

What to expect that differs from Linux:

- In a normal (non-elevated) session, `ssh-keygen` does **not** print `Enter PIN for authenticator`. Instead the **Windows Security** dialog pops up and handles PIN entry and the touch prompt. If you are running inside a session without an interactive desktop (SSH-in, remote PowerShell, a service), that dialog cannot appear and the operation fails.
- In an **elevated** session, Win32-OpenSSH talks to the token directly rather than through `webauthn.dll`, and you get the classic console PIN prompt.
- Creating a resident key through `webauthn.dll` sometimes produces a misleading warning that a credential already exists. Verify the actual state with `ykman fido credentials list` before overwriting anything.

3d. Fix file permissions

OpenSSH on Windows refuses to use a private key file that is readable by other principals:

```
icacls "$env:USERPROFILE\.ssh\id_ed25519_sk_yk1" /inheritance:r
icacls "$env:USERPROFILE\.ssh\id_ed25519_sk_yk1" /grant:r "$($env:USERNAME):(R)"
```

3e. Agent (optional)

```
Set-Service ssh-agent -StartupType Automatic
Start-Service ssh-agent
ssh-add "$env:USERPROFILE\.ssh\id_ed25519_sk_yk1"
```

The Windows agent stores keys in the registry under the user profile — for an `sk` key that is only the handle, which is fine. `ssh-add -K` is **not** implemented on Windows (see §7).

3f. WSL

WSL2 has no USB-HID passthrough by default, so `ssh-keygen -t ed25519-sk` inside WSL will not see the token. Three options, in order of sanity:

1. Generate and authenticate with the **Windows** `ssh.exe`, and only use WSL for the rest.
2. Forward the Windows `ssh-agent` into WSL via `npiperelay` + `socat`, and let Windows handle the token.
3. Attach the device with `usbipd-win` — works, but takes the YubiKey away from Windows while attached.

---

4. Deploy the public key

```
ssh-copy-id -i ~/.ssh/id_ed25519_sk_yk1.pub user@host
```

`ssh-copy-id` does not exist on Windows; use:

```
type "$env:USERPROFILE\.ssh\id_ed25519_sk_yk1.pub" | ssh user@host "cat >> ~/.ssh/authorized_keys"
```

Or write `~/.ssh/authorized_keys` directly, with per-key enforcement:

```
restrict,pty,verify-required sk-ssh-ed25519@openssh.com AAAAGnNr... oliver@karger.lan yk1
```

`verify-required` in `authorized_keys` requires sshd ≥ 8.4. Without it, the server accepts the signature even if the client skipped user verification — the client-side flag alone is not an access control.

5. Server configuration

```
PubkeyAuthentication yes
PubkeyAcceptedAlgorithms sk-ssh-ed25519@openssh.com,ssh-ed25519,rsa-sha2-512
PubkeyAuthOptions verify-required
PasswordAuthentication no
KbdInteractiveAuthentication no
```

Notes:

- `PubkeyAuthOptions verify-required` enforces PIN globally; it will lock out any non-`sk` key, so either scope it in a `Match` block or make sure every account has migrated first.
- If you already ship a hardened `PubkeyAcceptedAlgorithms` list via Ansible, `sk-ssh-ed25519@openssh.com` must be added explicitly — a restrictive list silently rejects the new key type.
- Reload and test in a *second* session before dropping the first:

```
sshd -t && systemctl reload ssh
```

6. Verify

```
ssh -o IdentitiesOnly=yes -i ~/.ssh/id_ed25519_sk_yk1 -v user@host
```

Expected flow: PIN prompt (console on Unix / elevated Windows, Windows Security dialog otherwise) → LED blinks → touch → session. In the verbose log you should see `Offering public key: ... ED25519-SK` and `Server accepts key`.

Pin it down in the client config — `~/.ssh/config` on Unix, `%USERPROFILE%\.ssh\config` on Windows:

```
Host *.karger.lan
    IdentityFile ~/.ssh/id_ed25519_sk_yk1
    IdentitiesOnly yes
    AddKeysToAgent no
```

7. Recovery on a new machine

Only works for **resident** keys.

Unix:

```
mkdir -m 700 ~/.ssh/rk && cd ~/.ssh/rk
ssh-keygen -K          # writes id_ed25519_sk_rk_<application> + .pub
# or, without writing the handle to disk:
ssh-add -K
```

Windows 11 Pro:

```
# MUST be an elevated PowerShell — ssh-keygen -K needs direct token access,
# which the non-elevated webauthn.dll path cannot provide.
mkdir "$env:USERPROFILE\.ssh\rk"; cd "$env:USERPROFILE\.ssh\rk"
ssh-keygen -K
```

`ssh-add -K` is not implemented in Win32-OpenSSH — `ssh-keygen -K` followed by `ssh-add <file>` is the only path.

For non-resident keys there is no recovery path — the handle file *is* the only copy. Back it up (it is not secret in isolation) or accept that losing it means re-enrolling.

---

8. Attestation (optional, useful for NIS2 evidence)

Proves the key was generated inside genuine Yubico hardware rather than a software authenticator.

```
dd if=/dev/urandom of=challenge.bin bs=32 count=1
ssh-keygen -t ed25519-sk \
  -O challenge=challenge.bin \
  -O write-attestation=attest.bin \
  -O verify-required \
  -f ~/.ssh/id_ed25519_sk_yk1
```

`attest.bin` contains the attestation certificate and signature; verification means chaining it to the Yubico U2F root CA and checking the signature over challenge + credential. There is no ready-made `ssh-keygen` verb for this — expect a small Python/`libfido2` helper. Archive `challenge.bin`, `attest.bin`, the public key and the token serial together as the enrolment record.

On Windows, generate attestation from an **elevated** prompt; the `webauthn.dll` path may strip or alter the attestation statement.

---

9. Rollout considerations

- **Two tokens per user, always.** The secret is non-exportable, so redundancy means a *second, independent* key pair on a backup token, enrolled in `authorized_keys` at the same time. Generate both during provisioning; a spare token issued later requires a second visit to every target system.
- **Enrol from one platform.** Mixing Windows- and Linux-generated credentials on the same token works, but the Windows quirks (dialog-based PIN, elevation requirement, resident-key warnings) make a single provisioning workstation the cheaper option. If enrolment happens at the user's own Windows 11 Pro machine, script it and require an elevated prompt so behaviour is consistent.
- **Naming.** `ssh:karger-<user>-<serial>` in `application=`, and the token serial in the key comment. That makes revocation on token loss a grep instead of an investigation.
- **Inventory.** Public key + token serial + issue date belong in NetBox (or wherever the token asset record lives), so that a lost-token report maps directly to the keys that need pulling.
- **Distribution.** Push `authorized_keys` from a single source of truth via Ansible (`ansible.posix.authorized_key` with `exclusive: yes`), otherwise revocation is unreliable.
- **Automation accounts** should not use `sk` keys at all — a touch requirement in a cron job just means someone will add `no-touch-required` and forget. Use a separate, non-`sk` key with `restrict` + `command=` instead.

---

10. Known friction

Area	Note
Windows, no PIN prompt	Non-elevated sessions delegate to `webauthn.dll`; the prompt is the Windows Security dialog, not the console. No interactive desktop → no key generation.
Windows, `ssh-add -K`	Not implemented. Use `ssh-keygen -K` from an elevated prompt.
Windows, resident keys	Spurious "credential already exists" warnings via `webauthn.dll`. Confirm with `ykman fido credentials list`.

Area	Note
Windows, two OpenSSH installs	FoD in `System32\OpenSSH\` vs MSI in `Program Files\OpenSSH\`. Check `where.exe ssh`.
WSL	No USB-HID passthrough by default; use the Windows client, agent relay, or `usbipd-win`.
Agent forwarding	Works, but every hop triggers PIN + touch on the *origin* machine.
Forgejo / Gitea	Accepts `sk-ssh-ed25519@openssh.com`, but check `[ssh]` minimum-key-size settings — some hardened configs reject unknown types outright.
`ykman fido reset`	Wipes all FIDO2 credentials, not just SSH ones. Never the first troubleshooting step.
Older YubiKeys	Firmware < 5.2.3 → `ed25519-sk` fails with a vague error. Fall back to `-t ecdsa-sk`.
Multiple tokens plugged in	`ssh-keygen` picks the first FIDO device it finds; unplug the others or pass `-O device=/dev/hidrawN` (Unix only).

Revision #2
Created 7 August 2026 17:35:42 by Oliver Karger
Updated 7 August 2026 17:41:21 by Oliver Karger