Setting Up Two YubiKeys: A Primary and a True Working Spare

Why an Unregistered Spare Isn't a Spare at All

Development Guide Summary

August 19, 2026

Contents

1 The One Assumption Worth Checking First

Two YubiKeys almost always means one primary and one spare – that part of the assumption is right. The part worth checking is what "spare" actually means in practice. Yubico's own guidance on this is blunt: *a spare key that isn't registered to your accounts is of no help to you if you lose the primary.* A second key sitting in a drawer, never enrolled anywhere, is not a backup – it's just a second unused device that happens to be identical to the one you lost.

The correct model is: **both keys get registered to every account, at the same time, during initial setup** – not "register the primary now, add the spare later." "Later" is exactly the moment that tends not to arrive before the primary goes missing.

2 What a YubiKey Actually Does

A YubiKey is not one authentication method – it is several, bundled into one USB device, and which one gets used depends entirely on what the account you're registering it with supports:

Table 1: Protocols a modern YubiKey (5 series) speaks

Protocol	**Where it's used**
FIDO2 / WebAuthn	Passwordless sign-in and passkeys (Google, Microsoft, GitHub, most modern "security key" 2FA prompts). Touch required; a PIN is required for resident/discoverable credentials.
FIDO U2F	Older-style second-factor prompts ("insert your security key and tap it"), the predecessor to WebAuthn, still widely supported as a fallback.
OTP (Yubico OTP / HOTP)	One-touch one-time-password generation, used by some enterprise VPNs and Yubico's own services.
TOTP	Standard 30-second rotating codes, stored on the key and read through the Yubico Authenticator app instead of a phone app.
PIV (smart card)	Certificate-based login – Windows Hello for Business, SSH key storage, code signing.
OpenPGP	GPG key storage for encryption/signing, including signed git commits.

For the purposes of this guide, the two protocols that matter for "day-to-day account logins" are FIDO2/WebAuthn and, where a site hasn't caught up yet, U2F. Everything below assumes those, with PIV and OpenPGP flagged as natural next steps at the end.

3 Tooling: Yubico Authenticator and `ykman`

Yubico ships two overlapping tools, and it's worth knowing which does what before starting:

- **Yubico Authenticator** – the GUI app. Shows TOTP codes stored on the key, manages the FIDO2 PIN, and displays basic device info. This is the one most people install first.
- `ykman` – the command-line tool underneath it, installed separately. Everything the GUI can do, plus scriptable device inspection – useful here specifically because you have two keys and want to confirm they actually match before relying on both equally.

Listing 1: Confirm both keys are visible and check their firmware

```
ykman list
ykman info
```

Run `ykman info` with only one key plugged in at a time (plugging in both simultaneously makes `ykman` ambiguous about which one it's talking to on some commands). **Compare the firmware version reported for each key.** If they were bought at different times, it is entirely possible for the "spare" to be an older hardware revision with a different FIDO2 feature set than the primary – worth knowing about now, not the day you're relying on it.

4 Step 1: Set a PIN on Both Keys

FIDO2 resident credentials (what most services now call "passkeys") require a PIN on the key itself – this is separate from, and in addition to, any account password. Set it identically-in-spirit (not necessarily the same literal PIN) on both keys before registering either with any account:

```
ykman fido access change-pin
```

or, in Yubico Authenticator: *the key icon → FIDO PINs → Set PIN.*

> **Warning:** FIDO2 keys lock out and wipe their resident credentials after too many incorrect PIN attempts (typically 8, counting down, with the retry counter resetting on a correct entry). This is a hardware security feature, not a bug, and it cannot be undone by anyone – not you, not Yubico. Write the PIN down somewhere durable (a password manager entry, or physically alongside where the spare key itself is stored) before you need it under pressure.

5 Step 2: Register Both Keys, in the Same Sitting, to Every Account

This is the step the "spare in a drawer" approach skips, and it is the entire point of owning two keys. For each account that supports security keys:

1. Go to the account's security/2FA settings and choose *Add security key* (wording varies: "Add security key," "Set up a passkey," "Add a new way to sign in").
2. Insert the **primary** key, touch it when prompted, and give it a label the account will show you later – something literal like `YubiKey Primary` rather than the default.
3. **Immediately, in the same session**, repeat the process for the **spare** key, labeled `YubiKey Spare`.
4. Confirm the account's security page now lists two distinct security keys, not one.

Table 2: Where to start – accounts most worth doing first

Account	**Notes**
Primary email (Google, etc.)	Highest priority – almost every other account's password-reset flow routes through email.
Password manager (Bitwarden, 1Password, etc.)	Your recovery codes for everything else likely live here, so this account itself cannot be allowed to become a single point of failure.
Microsoft account / Windows Hello for Business	Covers both the cloud account and, separately, Windows sign-in – see Section 8.
GitHub / GitLab	Supports both WebAuthn 2FA and, separately, PIV/OpenPGP-based signed commits and SSH (Section 10).
Any VPN / work SSO	Often the account with the most painful account-recovery process if you're locked out remotely.

6 Step 3: Store the Two Keys Somewhere Genuinely Different

Registering both keys everywhere solves the "lost one key" scenario. It does nothing for the "lost the bag with both keys in it" scenario, which is a real failure mode if both live on the same keychain. Yubico's own recommendation is to keep the spare "somewhere safe but accessible" – a

home safe, a fireproof box, a bank safe-deposit box, or with a trusted person – specifically *not* wherever the primary key already lives day to day (on your main keychain, in your everyday bag).

7 Step 4: Recovery Codes Are a Third Layer, Not a Backup for the Keys

Most services that support security keys also generate one-time **recovery/backup codes** when you first enable 2FA. Save these (in the password manager, or printed and stored with the spare key) as the true last resort – the scenario where the primary is lost *and* the spare isn't reachable (e.g., you're traveling without it). They are not a substitute for registering both physical keys; they are what's left if a moment arrives where neither key is available at all.

8 Windows-Specific Setup

Windows 10/11 supports YubiKeys directly for machine sign-in, separate from any browser-based account login:

```
Settings -> Accounts -> Sign-in options -> Security key ->
    Manage
```

This registers a key for *logging into this specific PC*, which is a different registration from any web account's 2FA – register both keys here too, on any machine where losing sign-in access would actually be a problem. In an organization using Windows Hello for Business, this typically also involves the PIV/smart-card side of the key rather than pure FIDO2; check with IT if the sign-in prompt looks like a certificate PIN rather than the usual security-key touch prompt.

9 Testing the Spare – Don't Assume, Verify

The only real test of a spare key is using *only* the spare to sign into something, with the primary deliberately out of reach. Doing this once at setup time, and again roughly annually, catches the two failure modes that "registered once and forgotten" setups are prone to: a service silently dropping an old registration (rare, but it happens during account security overhauls), and firmware/feature drift between two keys of different ages (Section 4).

10 Where This Goes Next

Two natural follow-ons once basic account registration is solid, both relevant to a development workflow already built around Geany and command-line

Table 3: Quick `ykman` reference

Command	Purpose
`ykman list`	List connected keys and serial numbers.
`ykman info`	Firmware version, enabled interfaces/applications.
`ykman fido info`	FIDO2 PIN status, credential count, retry counter.
`ykman fido credentials list`	List resident credentials (passkeys) stored on this key.
`ykman fido access change-pin`	Set or change the FIDO2 PIN.
`ykman piv info`	PIV/smart-card certificate status.

tooling:

- **Hardware-backed SSH keys**, generated on and never leaving the YubiKey's PIV or OpenPGP applet, for signing git commits and authenticating to GitHub/GitLab without a private key file sitting on disk at all.
- **A documented recovery drill**, written down once and re-run occasionally: which accounts have both keys registered, where the recovery codes live, and where the spare physically is – the same instinct already applied to this site's toolchain guides, aimed at a security setup instead of a build system.