Filter

Passkeys
Overview
Supported environments
Use cases
FAQ

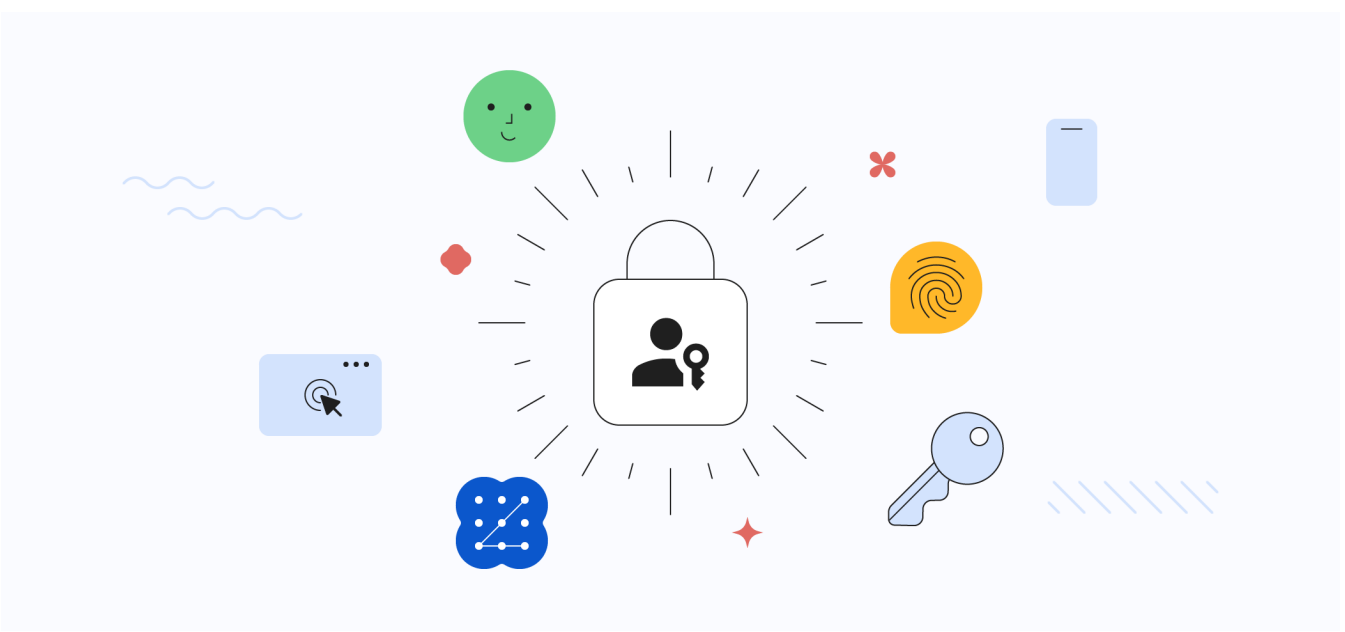
Developer guides
Overview
Server-side introduction
Server-side registration
Server-side authentication
User verification deep dive
Discoverable credentials deep dive
Prevent duplicate credentials
Determine the passkey provider
Passkeys upgrades

Integration guide for the web
Create a passkey on the web
Sign in with a passkey on the web

Integration guide for Android apps

Passwordless login with passkeys

Send feedback



Introduction

Passkeys are a safer and easier alternative to passwords. With passkeys, users can sign in to apps and websites with a biometric sensor (such as a fingerprint or facial recognition), PIN, or pattern, freeing them from having to remember and manage passwords.

Developers and users both hate passwords: they give a poor user experience, they add conversion friction, and they create security liability for both users and developers. Google Password Manager in Android and Chrome reduces the friction through autofill; for developers looking for even further improvements in conversion and security, passkeys and identity federation are the industry's modern approaches.

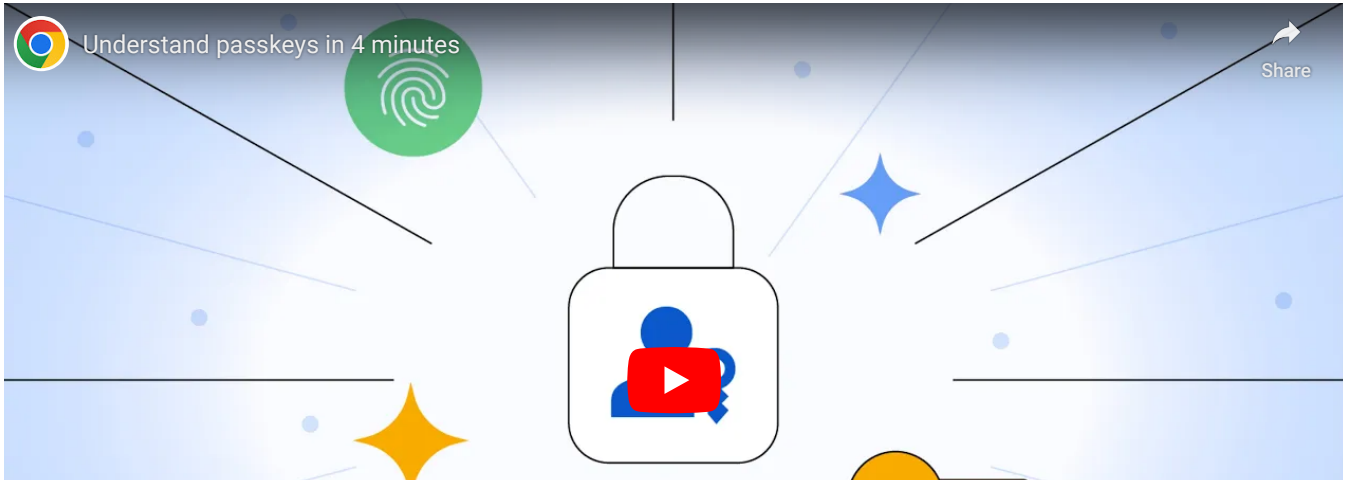
A passkey can meet multifactor authentication requirements in a single step, replacing both a password and OTP (e.g. 6-digit SMS code) to deliver robust protection against phishing attacks and avoids the UX pain of SMS or app-based one-time passwords. Since passkeys are standardized, a single implementation enables a passwordless experience across all of a users' devices, across different browsers and operating systems.

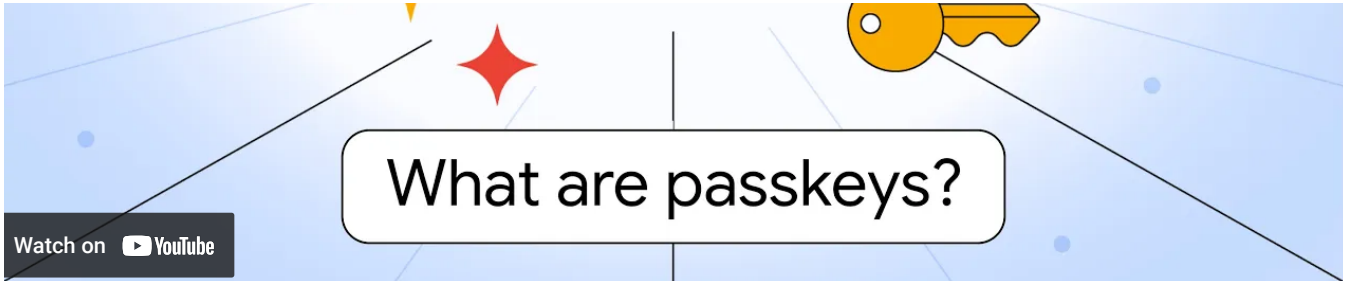
Passkeys are easier:

- Users can select an account to sign in with. Typing the username is not required.
- Users can authenticate using device's screen lock such as a fingerprint sensor, facial recognition or PIN.
- Once a passkey is created and registered, the user can seamlessly switch to a new device and immediately use it without needing to re-enroll (unlike traditional biometric auth, which requires setup on each device).

Passkeys are safer:

- Developers only save a public key to the server instead of a password, meaning there's far less value for a bad actor to hack into servers, and far less cleanup to do in the event of a breach.
- Passkeys protect users from phishing attacks. Passkeys work only on their registered websites and apps; a user cannot be tricked into authenticating on a deceptive site because the browser or OS handles verification.
- Passkeys reduce costs for sending SMS, making them a safer and more cost-effective means for two-factor authentication.





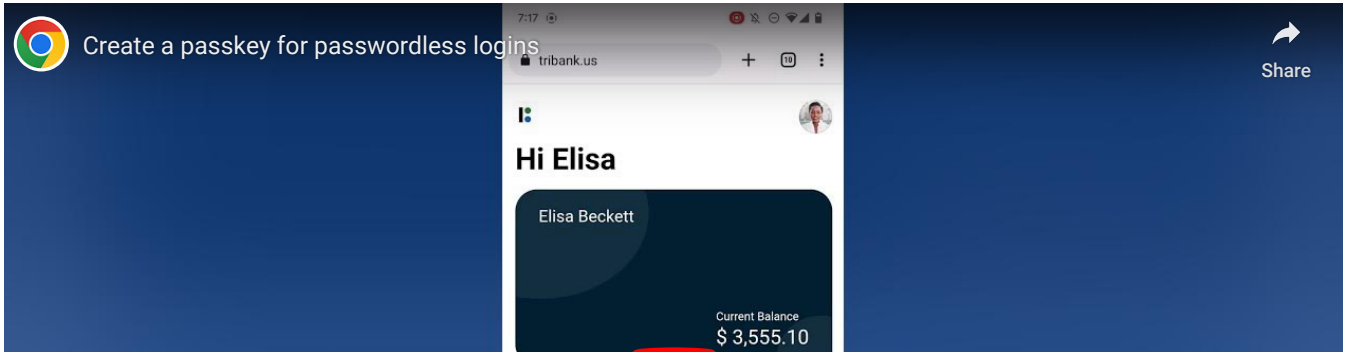
What are passkeys?

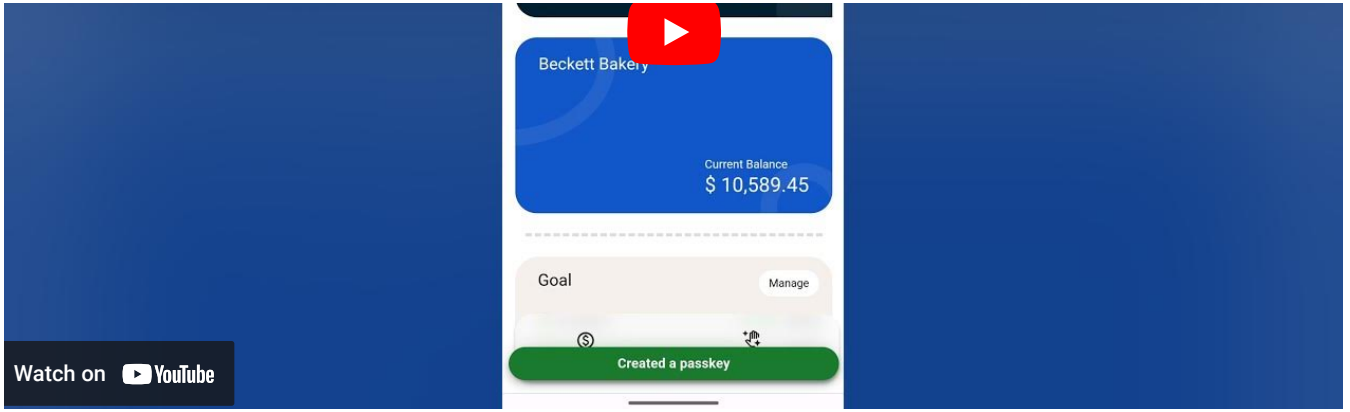
A *passkey* is a digital credential, tied to a user account and a website or application. Passkeys allow users to authenticate without having to enter a username or password, or provide any additional authentication factor. This technology aims to *replace legacy authentication mechanisms such as passwords*.

When a user wants to sign in to a service that uses passkeys, their browser or operating system will help them select and use the right passkey. The experience is similar to how saved passwords work today. To make sure only the rightful owner can use a passkey, the system will ask them to unlock their device. This may be performed with a biometric sensor (such as a fingerprint or facial recognition), PIN, or pattern.

To create a passkey for a website or application, a user first must register with that website or application.

1. Go to the application and sign in using the existing sign-in method.
2. Click **Create a passkey** button.
3. Check the information stored with the new passkey.
4. Use the device screen unlock to create the passkey.

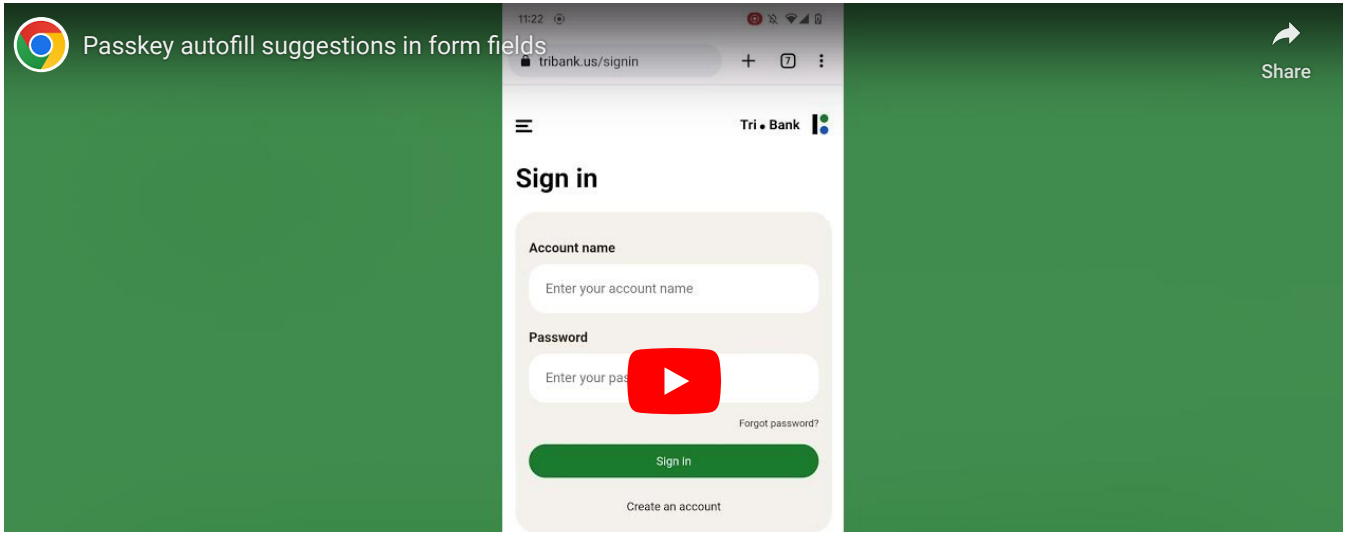


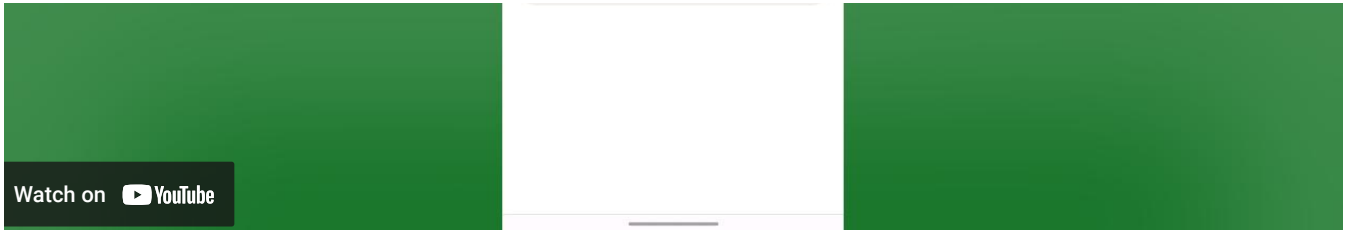


When they return to this website or app to sign in, they can take the following steps:

1. Go to the application.
2. Tap on the account name field to show a list of passkeys in an autofill dialog.
3. Select their passkey.
4. Use the device screen unlock to complete the login.

The user's device generates a signature based on the passkey. This signature is used to verify the login credential between the origin and the passkey.





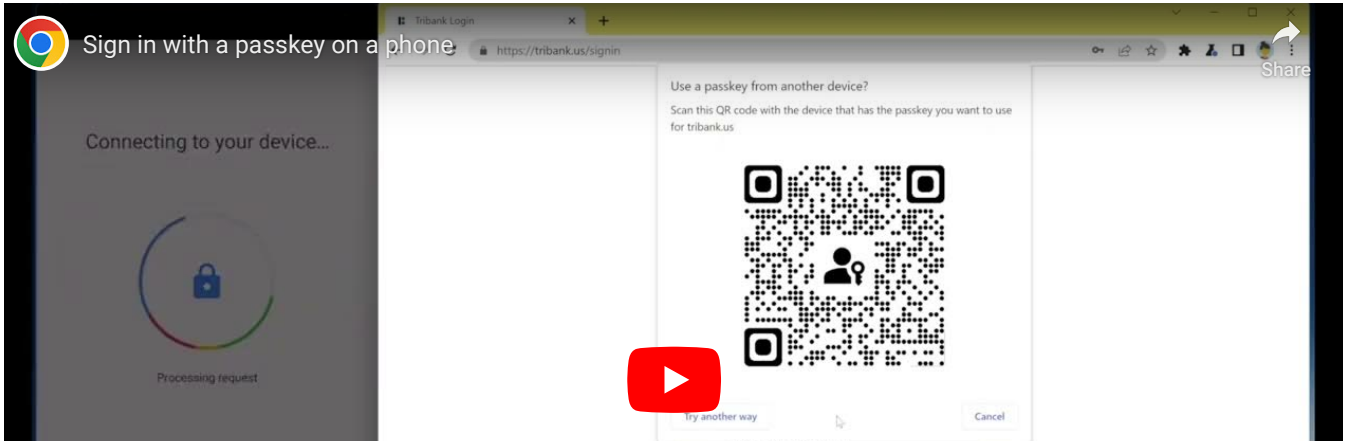
A user can sign into services on any device using a passkey, regardless of where the passkey is stored. For example, a passkey created on a mobile phone can be used to sign in to a website on a separate laptop.

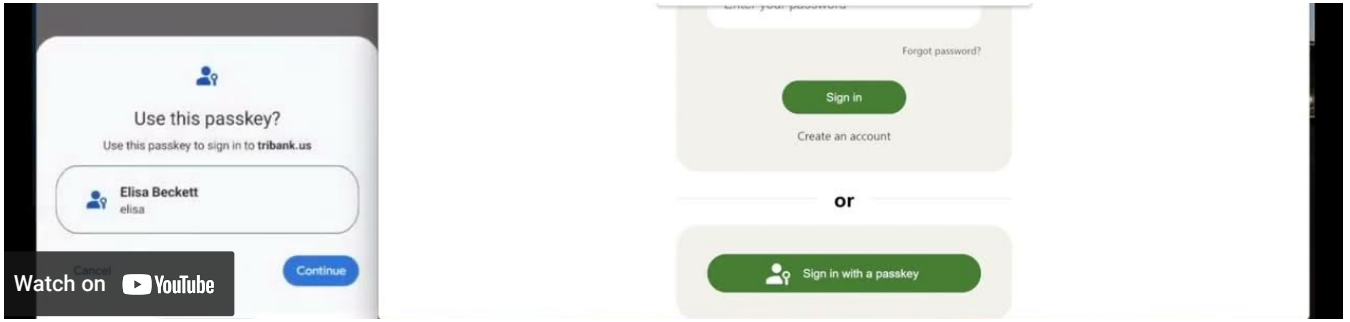
How do passkeys work?

Passkeys are intended to be used through operating system infrastructure that allows passkey managers to create, backup, and make passkeys available to the applications running on that operating system. On Android, passkeys can be stored in the Google Password Manager, which synchronizes passkeys between the user's Android devices that are signed into the same Google account. Passkeys are securely encrypted on-device before being synced, and requires decrypting them on new devices. Users with Android OS 14 or later can opt to store their passkeys in a compatible third-party password manager.

Note: See Passkey support on Android and Chrome to learn how Android and Chrome behave on each operating system.

Users aren't restricted to using the passkeys only on the device where they're available—passkeys available on phones can be used when logging into a laptop, even if the passkey isn't synchronized to the laptop, as long as the phone is near the laptop and the user approves the sign-in on the phone. As passkeys are built on FIDO standards, all browsers can adopt them.





For example, a user visits `example.com` on the Chrome browser on their Windows machine. This user has previously logged into `example.com` on their Android device and generated a passkey. On the Windows machine, the user chooses to sign in with a passkey from another device. The two devices will connect and the user will be prompted to approve the use of their passkey on the Android device, for example, with a fingerprint sensor. After doing so, they're signed in on the Windows machine. Note that the passkey itself isn't transferred to the Windows machine, so typically `example.com` will offer to create a new passkey there. That way, the phone isn't required next time the user wants to sign in. Read Sign-in with a phone to learn more.

Who is using passkeys?

A number of services are already using passkeys in their systems.

- DocuSign
- Google
  - The beginning of the end of the password
  - Google Online Security Blog: Making authentication faster than ever: passkeys vs. passwords
  - Designing the user experience of passkeys on Google accounts
- Kayak
- Mercari
- NTT Docomo
- PayPal
- Shopify
  - How Passkeys Reduce Friction in the Ecommerce Shopping Experience

    - Supporting Passkeys in Shop's Authentication Flows
- Yahoo! JAPAN
    - Yahoo! JAPAN's password-free authentication reduced inquiries by 25%, sped up sign-in time by 2.6x

Try it yourself

You can try passkeys in this demo: https://passkeys-demo.appspot.com/

Privacy considerations

★ **Important:** Passkeys have been designed with user privacy in mind. Several concerns that end users may raise appear below; to reassure your users, developers should add a reassuring message to the UI (e.g. "With passkeys, the user's biometric information is never revealed to the website or the app. Biometric material never leaves the user's personal device") and create an FAQ or support article explaining more.

- Because signing in with biometric might give users a false impression that this is sending sensitive information to the server. In reality, biometric material never leaves the user's personal device.
- Passkeys on their own don't allow tracking users or devices between sites. The same passkey is never used with more than one site. Passkey protocols are carefully designed so that no information shared with sites can be used as a tracking vector.
- Passkey managers protect passkeys from unauthorized access and use. For example, the Google Password Manager encrypts passkey secrets end-to-end. Only the user can access and use them, and even though they're backed up to Google's servers, Google can't use them to impersonate users.

Security considerations

- Passkeys use public key cryptography. Public key cryptography reduces the threat from potential data breaches. When a user creates a passkey with a site or application, this generates a public–private key pair on the user's device. Only the public key is stored by the site, but this alone is useless to an attacker. An attacker can't derive the user's private key from the data stored on the server, which is required to complete authentication.
- Because passkeys are bound to a website or app's identity, they're safe from phishing attacks. The browser and operating system ensure that a passkey can only be used with the website or app that created them. This frees users from being responsible for signing in to the genuine website or app.

Get notified

Subscribe to the Google passkeys developer newsletter to get notified about passkey updates.

Next steps

- Passkey support on Android and Chrome
- Frequently asked questions (FAQ)
- Use cases
- Passkeys developer guide for relying parties
- Learn how to sign-in to an Android app using the Credential Manager

Was this helpful?

Send feedback

Last updated 2024-05-03 UTC.

GitHub
Fork our samples and try them yourself

Stack Overflow
Ask a question under the google-signin tag

Blog

Chromium Blog

The latest news on the Google Developers blog

The latest news on the Chromium blog.

Product Info

Terms of Service

Branding Guidelines

Help

Sign In With Google

Google Identity

Developer consoles

Google API Console

Google Cloud Platform Console

Google Play Console

Firebase Console

Actions on Google Console

Cast SDK Developer Console

Chrome Web Store Dashboard

Android

Chrome

Firebase

Google Cloud Platform

All products

Terms | Privacy

Sign up for the Google for Developers newsletter

Subscribe

Language