passbolt

Fosdem 2024

Beyond passwords: secure authentication with passkeys

Identity and Access Management devroom

Hello, I’m Remy,

Co-founder of passbolt

$ whoami

/passbolt
@stripthis

@passbolt@mastodon.social

passbolt,
FIDO CP-SIG

fido ALLIANCE MEMBER

What is authentication?

Asserting a user identity using something they:

know (passphrase, password, pin)
have (token, certificate, key)
are (biometric)
or do (typing pattern, gait)

Password based authentication

Security issues:

- Credential stuffing.
- Phishing.
- Password loss.
- Bruteforce (online)
- Bruteforce (offline, in case of leak).

~ Adversary in the middle (network)

~ Password logging.

~ User enumeration

Implementation considerations:

+ Checking against breaches & entropy

~ User training

+ Account recovery
+ Captcha (+GDPR) / WAF / Alerts
+ "Costly" hashing mechanism (bcrypt)
+ HTTPs pinned and well configured
+ Additional client side hashing?

~ Vague error messages & constant time?

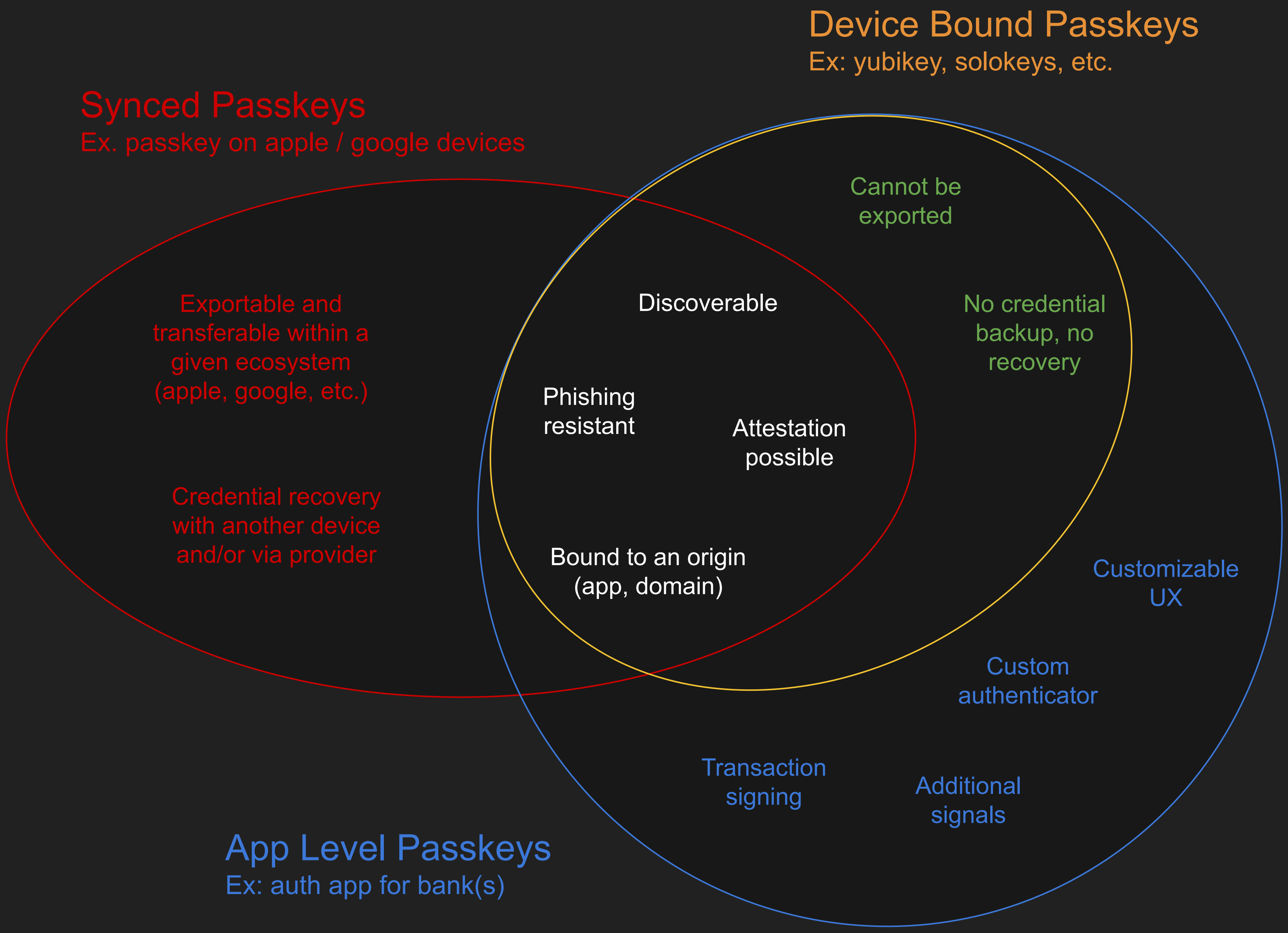
Who has setup passkeys as a user?
As a developer?

What is a passkey?

Passkeys are passwords replacements. They are public/private key pairs used for user authentication using cryptographic signatures.

Passkeys are user credentials that are discoverable (by the browser, websites, apps).

They are stored within applications or security keys. They may be synced across devices.



Device Bound Passkeys

Ex: yubikey, solokeys, etc.

Synced Passkeys

Ex. passkey on apple / google devices

Exportable and transferable within a given ecosystem (apple, google, etc.)

Credential recovery with another device and/or via provider

Discoverable

Phishing resistant

Attestation possible

Bound to an origin (app, domain)

Cannot be exported

No credential backup, no recovery

Customizable UX

Custom authenticator

Transaction signing

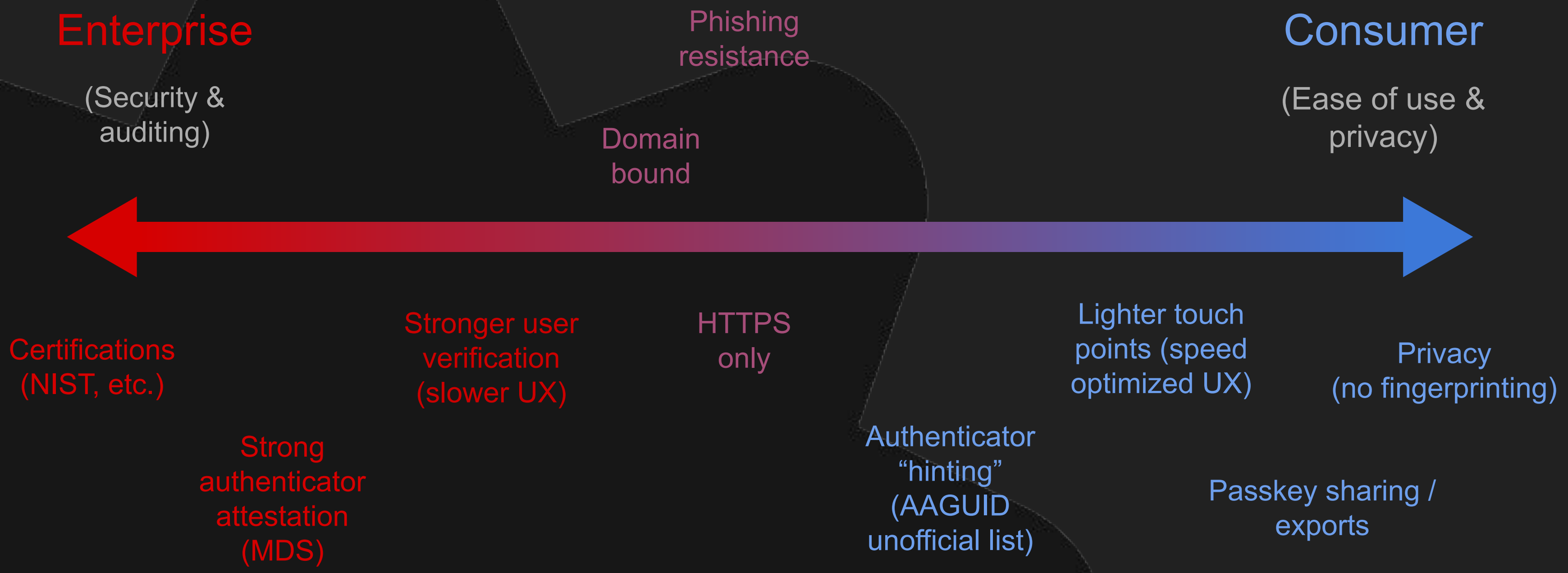
Additional signals

App Level Passkeys

Ex: auth app for bank(s)

Conflicting requirements? More options!

The complex art of balancing the standards to cater for different audiences...



Enterprise

(Security & auditing)

Phishing resistance

Consumer

(Ease of use & privacy)

Domain bound

Certifications (NIST, etc.)

Stronger user verification (slower UX)

HTTPS only

Lighter touch points (speed optimized UX)

Privacy (no fingerprinting)

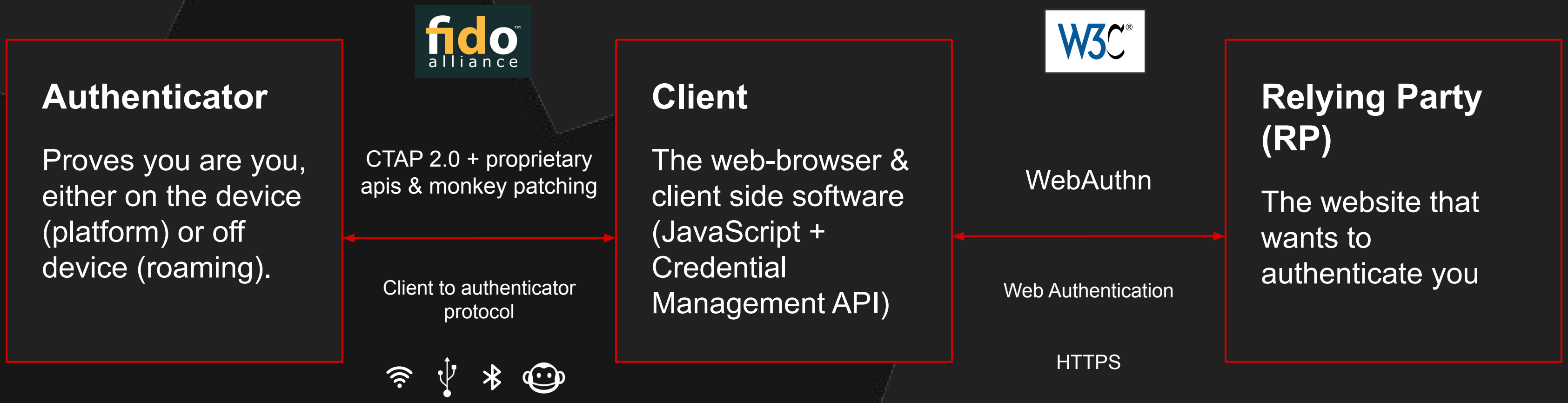
Strong authenticator attestation (MDS)

Authenticator “hinting” (AAGUID unofficial list)

Passkey sharing / exports

FIDO2 Project

A joint effort between the FIDO Alliance and the W3C.



Authenticator

Proves you are you, either on the device (platform) or off device (roaming).

CTAP 2.0 + proprietary apis & monkey patching

Client to authenticator protocol

Client

The web-browser & client side software (JavaScript + Credential Management API)

WebAuthn

Web Authentication

HTTPS

Relying Party (RP)

The website that wants to authenticate you

Examples:
iOS Keychain
Microsoft Hello
Yubikey
Dashlane
Etc.

Examples:
Chrome
Firefox
Safari
Etc.

Examples:
Google
Mastodon
etc.

Which ceremonies are supported?

Attestation (Registration)

When an authenticator registers a new key pair with a service. Either first one or as an alternative for recovery.

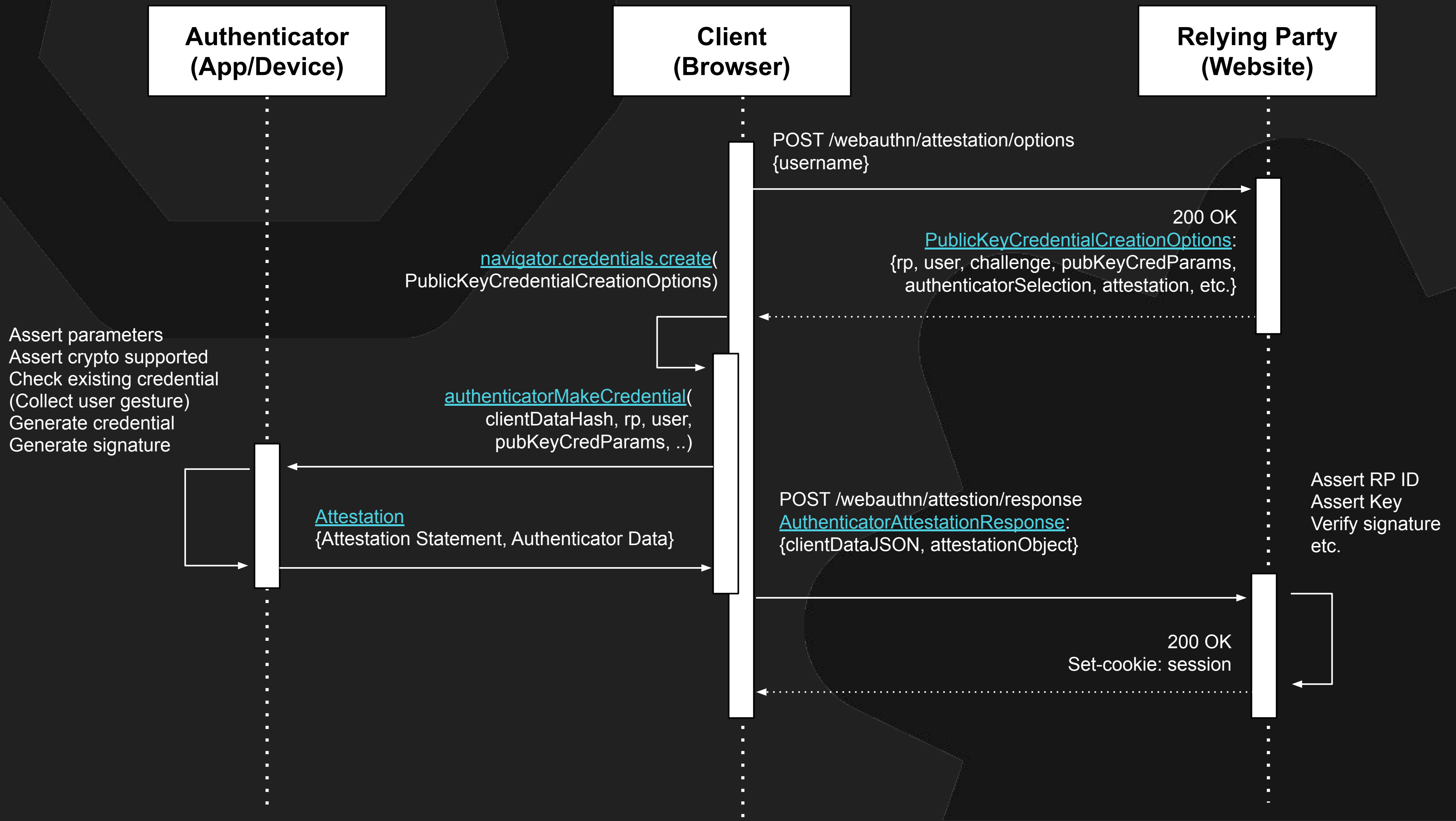
Assertion (Login)

When a user chooses to log into a service.

Not supported: listing and deletion of passkeys. RPs are in charge of this (potentially leading to accessibility / security issues).

Attestation ceremony

e.g. a client sends a registration request



Authenticator (App/Device)

Client (Browser)

Relying Party (Website)

POST /webauthn/attestation/options
{username}

200 OK
PublicKeyCredentialCreationOptions:
{rp, user, challenge, pubKeyCredParams,
authenticatorSelection, attestation, etc.}

navigator.credentials.create(
PublicKeyCredentialCreationOptions)

Assert parameters
Assert crypto supported
Check existing credential
(Collect user gesture)
Generate credential
Generate signature

authenticatorMakeCredential(
clientDataHash, rp, user,
pubKeyCredParams, ..)

Attestation
{Attestation Statement, Authenticator Data}

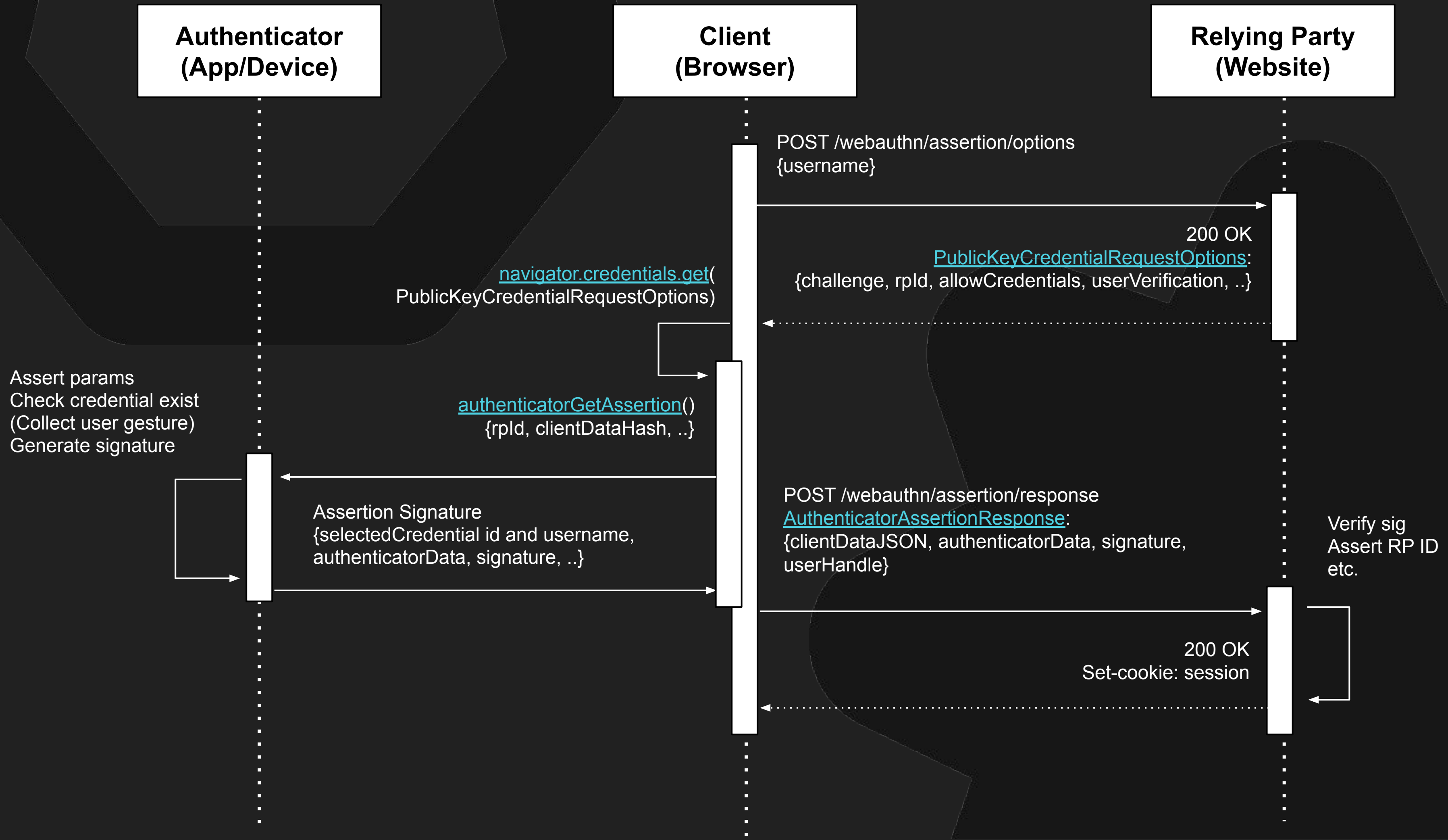
POST /webauthn/attestion/response
AuthenticatorAttestationResponse:
{clientDataJSON, attestationObject}

Assert RP ID
Assert Key
Verify signature
etc.

200 OK
Set-cookie: session

Assertion ceremony

e.g. authentication flow (login flow)



Authenticator (App/Device)

Client (Browser)

Relying Party (Website)

POST /webauthn/assertion/options
{username}

200 OK
PublicKeyCredentialRequestOptions:
{challenge, rpId, allowCredentials, userVerification, ..}

navigator.credentials.get(
PublicKeyCredentialRequestOptions)

authenticatorGetAssertion()
{rpId, clientDataHash, ..}

Assert params
Check credential exist
(Collect user gesture)
Generate signature

Assertion Signature
{selectedCredential id and username,
authenticatorData, signature, ..}

POST /webauthn/assertion/response
AuthenticatorAssertionResponse:
{clientDataJSON, authenticatorData, signature,
userHandle}

Verify sig
Assert RP ID
etc.

200 OK
Set-cookie: session

What about account recovery?

RPs

- ✓ More than one passkeys
- ✓ Password
- ✓ Magic Link

Authenticators / Platforms

- ✓ Another device
- ~ Recovery contact
- ~ Custom procedure

Example for iCloud

“Passkeys can be recovered through iCloud keychain escrow, which is also protected against brute-force attacks, even by Apple. [...]

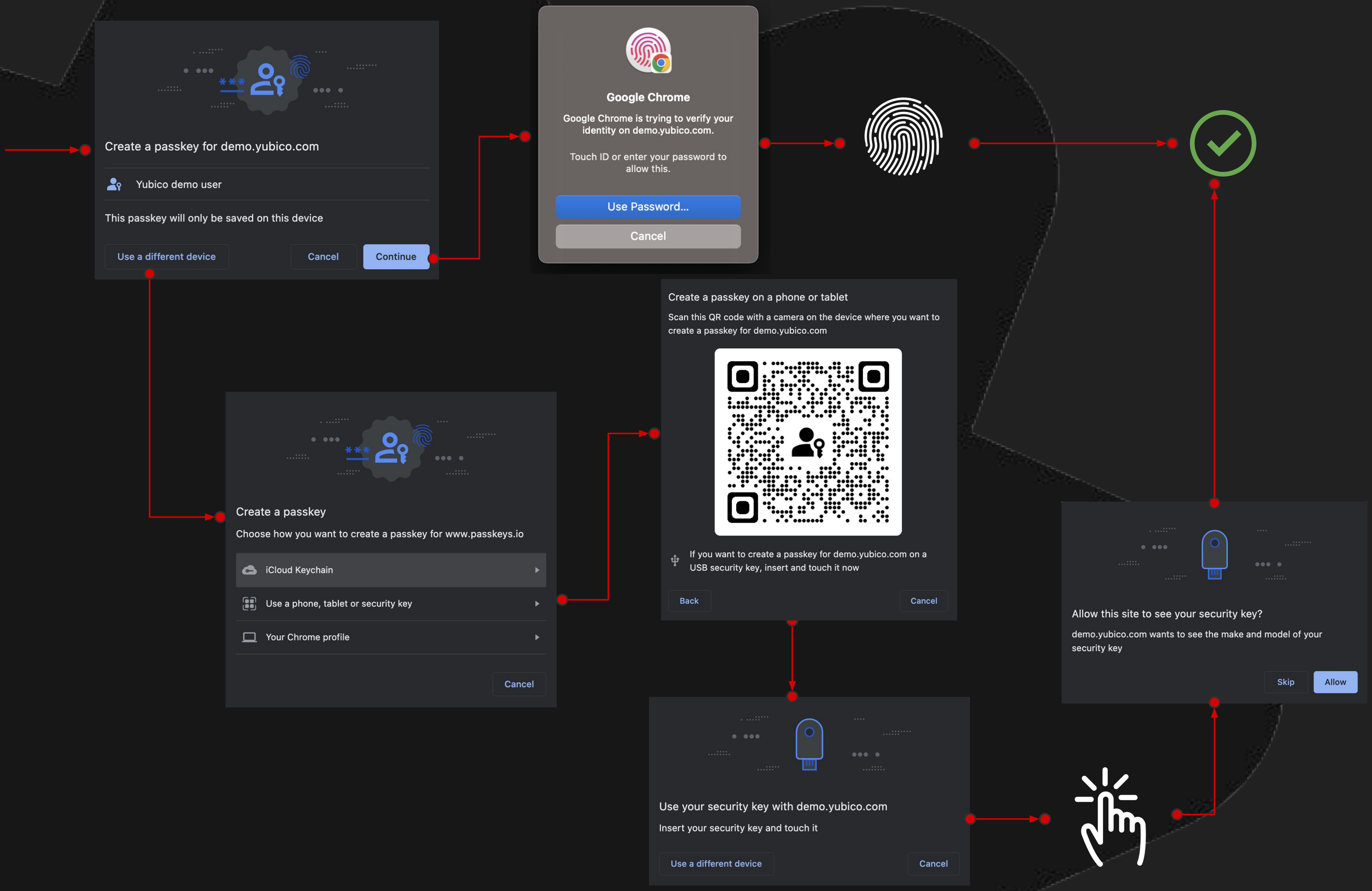
To recover a keychain, a user must authenticate with their iCloud account and password and respond to an SMS sent to their registered phone number. After they authenticate and respond, the user must enter their [lost] device passcode. iOS, iPadOS, and macOS allow only 10 attempts to authenticate. After several failed attempts, the record is locked and the user must call Apple Support to be granted more attempts. After the tenth failed attempt, the escrow record is destroyed.

Optionally, a user can set up an account recovery contact [...].”

Ref. https://support.apple.com/en-gb/guide/security/sec3e341e75d/web

How does it look like?

Registration on MacOS / Chrome (01/24)



Create a passkey for demo.yubico.com

Yubico demo user

This passkey will only be saved on this device

Google Chrome

Google Chrome is trying to verify your identity on demo.yubico.com.

Touch ID or enter your password to allow this.

Create a passkey

Choose how you want to create a passkey for www.passkeys.io

Create a passkey on a phone or tablet

Scan this QR code with a camera on the device where you want to create a passkey for demo.yubico.com

If you want to create a passkey for demo.yubico.com on a USB security key, insert and touch it now

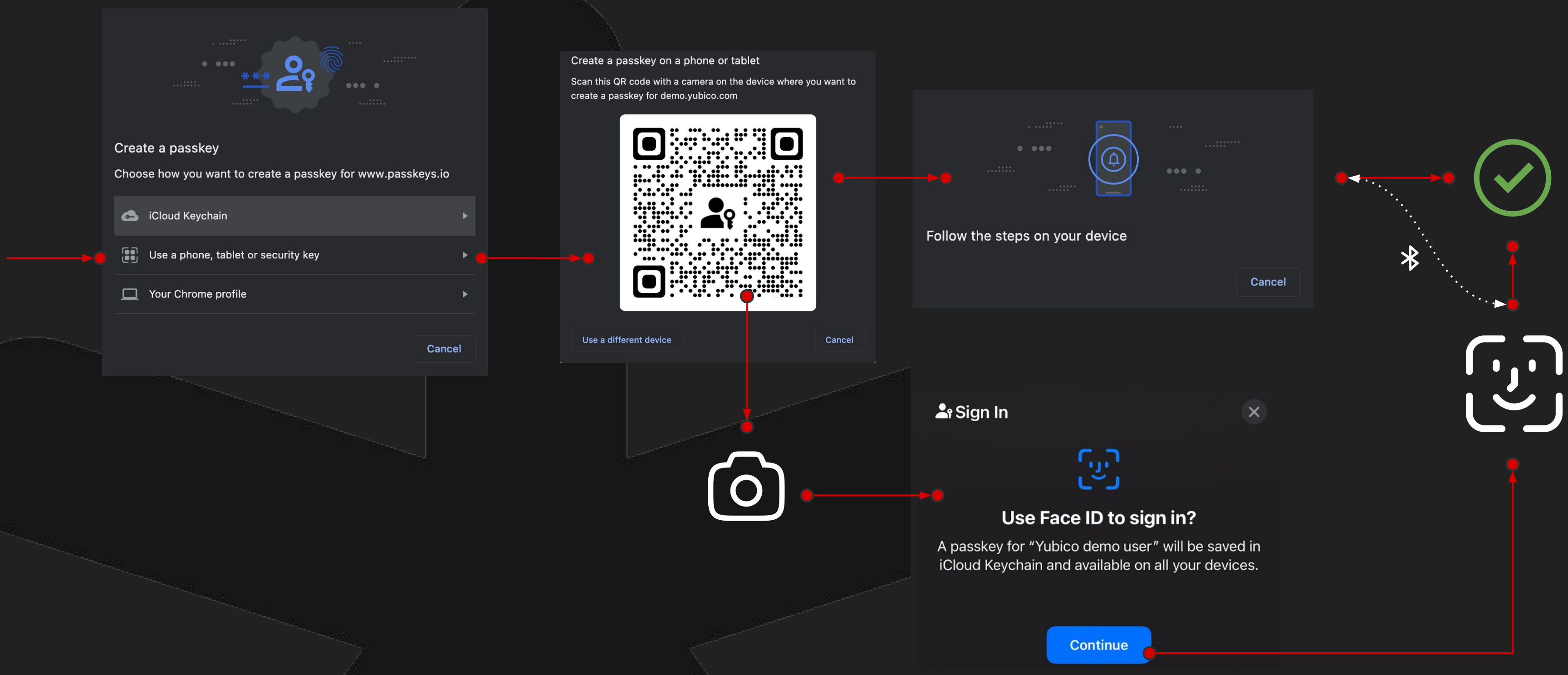
Use your security key with demo.yubico.com

Insert your security key and touch it

Allow this site to see your security key?

demo.yubico.com wants to see the make and model of your security key

Registration on MacOS/Chrome/iOS (01/24)



Create a passkey

Choose how you want to create a passkey for www.passkeys.io

Create a passkey on a phone or tablet

Scan this QR code with a camera on the device where you want to create a passkey for demo.yubico.com

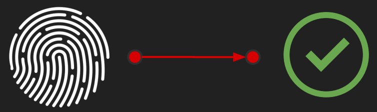
Follow the steps on your device

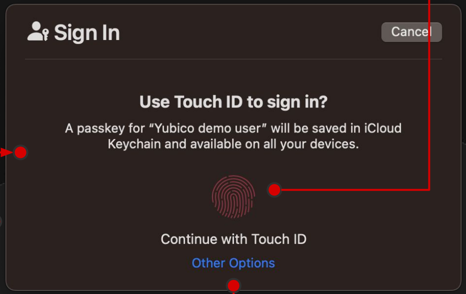
Sign In

Use Face ID to sign in?

A passkey for “Yubico demo user” will be saved in iCloud Keychain and available on all your devices.

Registration on MacOS/Safari or Firefox (01/24)





Sign In

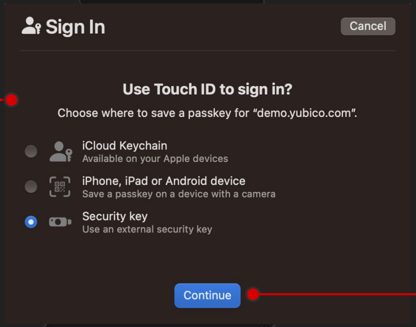
Cancel

Use Touch ID to sign in?

A passkey for "Yubico demo user" will be saved in iCloud Keychain and available on all your devices.

Continue with Touch ID

Other Options



Sign In

Cancel

Use Touch ID to sign in?

Choose where to save a passkey for "demo.yubico.com".

○ iCloud Keychain
Available on your Apple devices

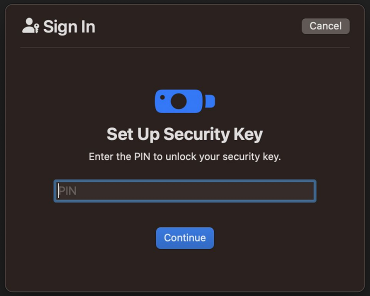
○ iPhone, iPad or Android device
Save a passkey on a device with a camera

● Security key
Use an external security key

Continue

"Currently, YubiKeys can store a maximum of 25 passkeys." (if you've never entered a PIN when you registered your Yubikey you are not using resident keys).

🤔



Sign In

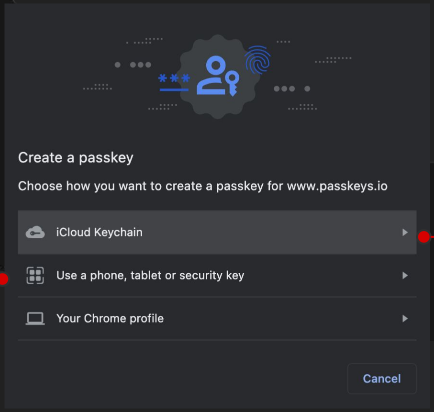
Cancel

Set Up Security Key

Enter the PIN to unlock your security key.

PIN

Continue



Create a passkey

Choose how you want to create a passkey for www.passkeys.io

iCloud Keychain

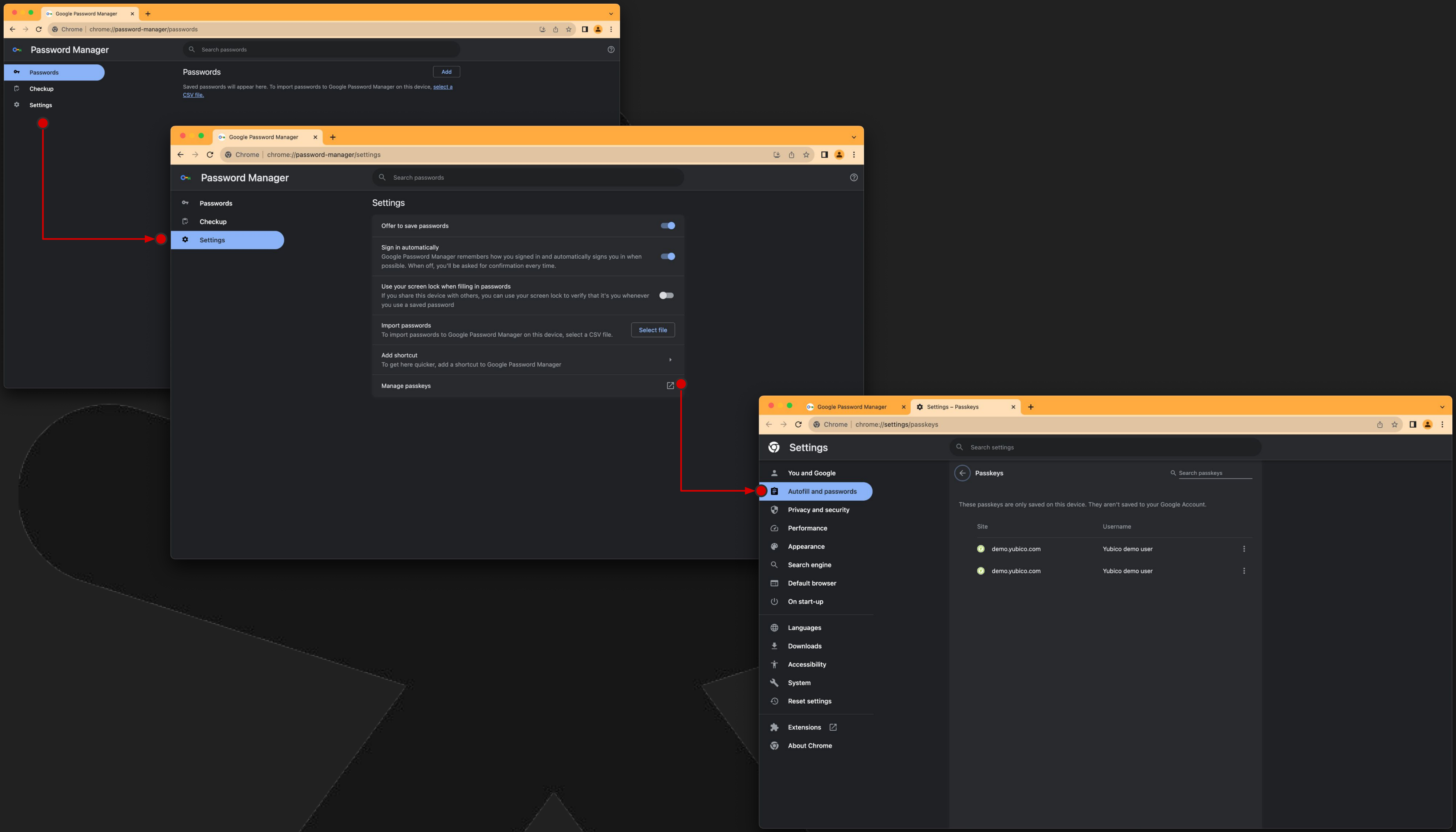
Use a phone, tablet or security key

Your Chrome profile

Cancel

Or from Chrome...

Managing passkeys on MacOS/Chrome



Passwords

Saved passwords will appear here. To import passwords to Google Password Manager on this device, select a CSV file.

Settings

Offer to save passwords

Sign in automatically
Google Password Manager remembers how you signed in and automatically signs you in when possible. When off, you'll be asked for confirmation every time.

Use your screen lock when filling in passwords
If you share this device with others, you can use your screen lock to verify that it's you whenever you use a saved password

Import passwords
To import passwords to Google Password Manager on this device, select a CSV file.

Select file

Add shortcut
To get here quicker, add a shortcut to Google Password Manager

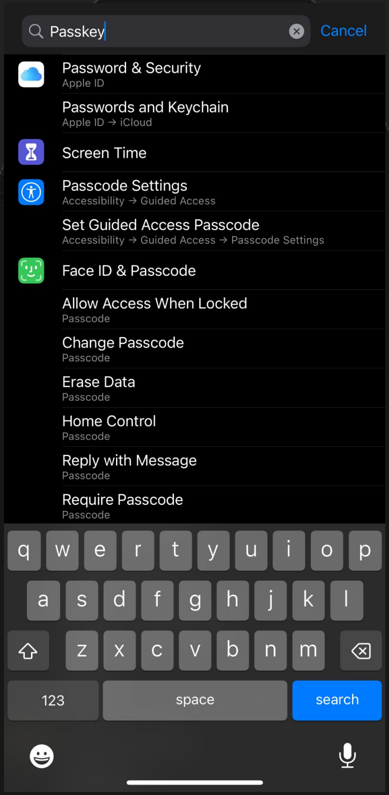
Manage passkeys

Passkeys

These passkeys are only saved on this device. They aren't saved to your Google Account.

Site	Username
demo.yubico.com	Yubico demo user
demo.yubico.com	Yubico demo user

Managing Pass~~words~~keys on iOS



Passkey

Password & Security
Apple ID

Passwords and Keychain
Apple ID → iCloud

Screen Time

Passcode Settings
Accessibility → Guided Access

Set Guided Access Passcode
Accessibility → Guided Access → Passcode Settings

Face ID & Passcode

Allow Access When Locked
Passcode

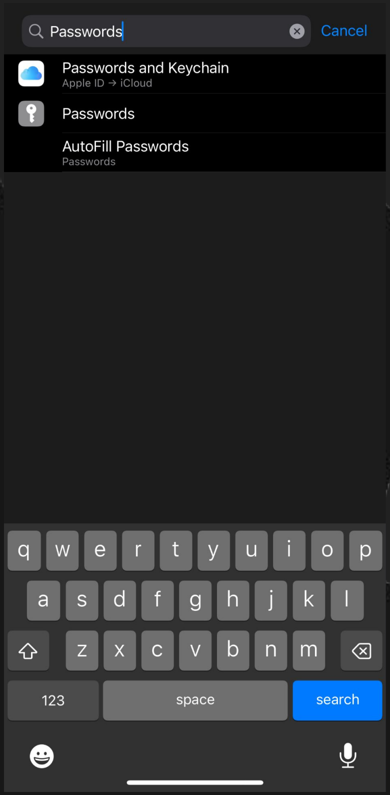
Change Passcode
Passcode

Erase Data
Passcode

Home Control
Passcode

Reply with Message
Passcode

Require Passcode
Passcode

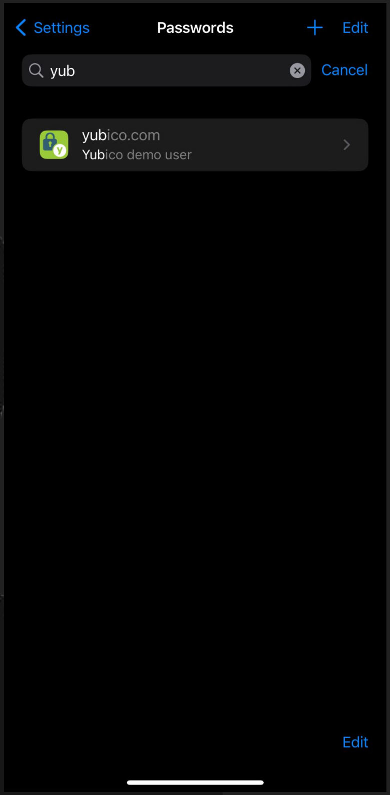


Passwords

Passwords and Keychain
Apple ID → iCloud

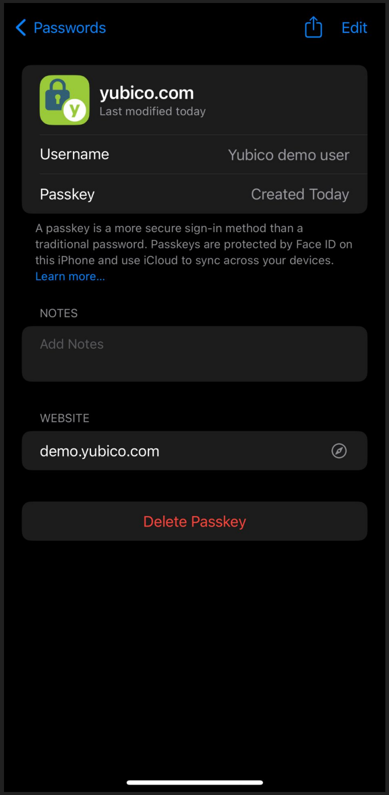
Passwords

AutoFill Passwords
Passwords



yub

yubico.com
Yubico demo user



yubico.com
Last modified today

Username	Yubico demo user
Passkey	Created Today

A passkey is a more secure sign-in method than a traditional password. Passkeys are protected by Face ID on this iPhone and use iCloud to sync across your devices. Learn more...

NOTES

Add Notes

WEBSITE

demo.yubico.com

Passkeys security issues

Security issues:

- Device / platform account loss
- Passkey management & review

~ Passkeys transfer/sharing

- User enumeration

~ CA revocation

~ Quantum computers? Weak PQC?

~ & more (UI redressing, proximity)

Implementation considerations:

~ Account recovery? More passkeys?

~ User training? Better UX? Alerts?

~ Better signalization of sharing props?

~ Random username / fake credential ids?

~ Forced rotations? Devices exclusion?

- Crypto agility?

~ RTFM?

Passkeys other issues

Other issues:

- Fragmented end user experience
- Specs depth & stability
- Entry barrier for authenticators
- Pay to play

Other considerations:

~ UX Working group

~ Passkeys “the good parts”? RP Guidelines?

~ Monkey patching? EU Fines?

~ Pooling of resources for open source actors

Questions? 🍅?

Thank you Fosdem

🍻 ❤️