

WebAuthn

The future of user authentication on the web 🤝

Lucas Garron
CS 253 Guest Talk
2021-11-11



Lucas Garron

@lgarron

Mathematician, cuber, dancer, coder. I want the web to win. @GitHub websec, formerly @GoogleChrome usable security. Immigrant. He/him.

⌚ Mountain View ⚡ garron.net

You may know me from:

Chrome DevTools Security

badssl.com, hstspreload.org

Speedcubing, Dancing

WebAuthn at GitHub

Ben Toews
([@mastahyeti](https://twitter.com/mastahyeti))
implemented U2F.

I wrote most of the
WebAuthn
implementation.

August 21, 2019 — Product, Security

GitHub supports Web Authentication (WebAuthn) for security keys



Lucas Garron

GitHub now supports [Web Authentication \(WebAuthn\)](#) for security keys—the new standard for secure authentication on the web. Starting today, you can use security keys for two-factor authentication on GitHub with even more browsers and devices. And, since many browsers are actively working on WebAuthn features, we're excited about the potential for strong and easy-to-use authentication options for the entire GitHub community in the future.

[Register a new security key in your GitHub settings](#)

More browsers, devices, and biometric options

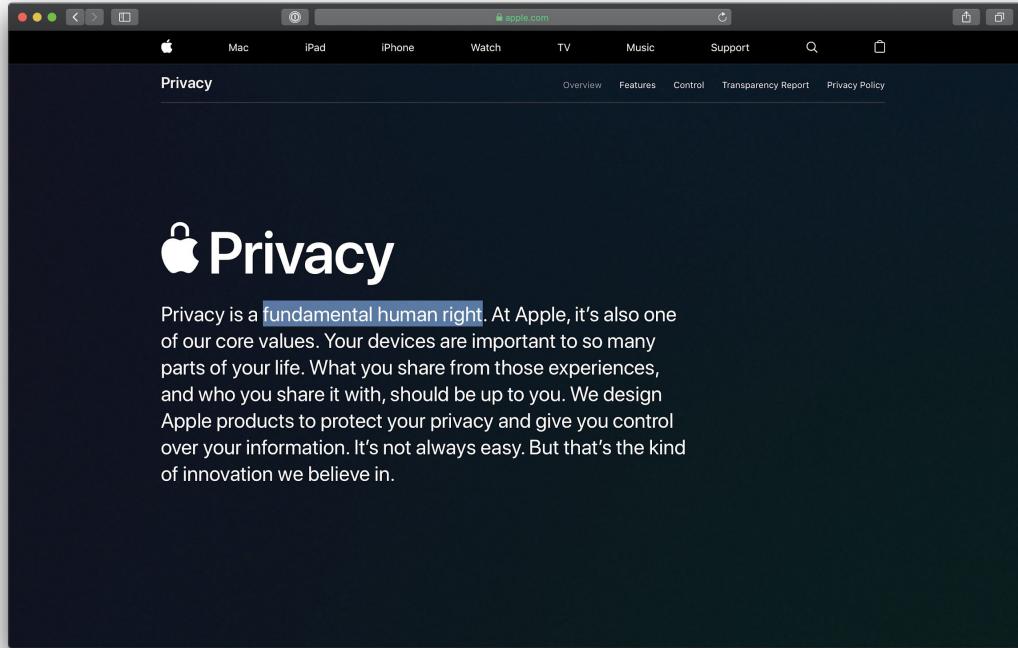
Previously, GitHub supported physical security keys [using the experimental U2F API for Chrome](#). WebAuthn is the standards-based successor. You can now use physical security keys on GitHub with:

- Windows, macOS, Linux, and Android: Firefox and Chrome-based browsers
- Windows: Edge
- macOS: Safari, currently in [Technology Preview](#) but coming soon to everyone
- iOS: [Brave](#), using the new [YubiKey 5Ci](#)

A few words on Responsibility



Security and Privacy are not “add-on features”



Privacy is a **fundamental human right**. At Apple, it's also one of our core values. Your devices are important to so many parts of your life. What you share from those experiences, and who you share it with, should be up to you. We design Apple products to protect your privacy and give you control over your information. It's not always easy. But that's the kind of innovation we believe in.

Passwords (Redux)

“Use bcrypt”

Terribly phishable

HavelBeenPwned.com

Authentication Factors

Factor

Something
you ____.



Factor

Something
you know.



Example:
Password

Factor

Something
you have.



Example:
Security Key

Factor

Something
you are.



Example:
Fingerprint

Classical “Factors”

.....



Stop thinking about factors

WebAuthn is supposed to help you...
Stop thinking about factors

WebAuthn

WebAuthn

A browser API
for many authentication factors.

WebAuthn

`navigator.credentials.create(...)`

`navigator.credentials.get(...)`

WebAuthn

§ IDL Index

```
(SecureContext, Exposed=>idc)
interface PublicKeyCredential : Credential {
    [SameObject] readonly attribute ArrayBuffer     rawId;
    [SameObject] readonly attribute AuthenticatorResponse  response;
    AuthenticationExtensionsClientOutputs getClientExtensionResults();
};

partial dictionary CredentialCreationOptions {
    PublicKeyCredentialCreationOptions  publicKey;
};

partial dictionary CredentialRequestOptions {
    PublicKeyCredentialRequestOptions  publicKey;
};

partial interface PublicKeyCredential {
    static Promise<boolean> isUserVerifyingPlatformAuthenticatorAvailable();
};

(SecureContext, Exposed=>idc)
interface AuthenticatorAttestationResponse : AuthenticatorResponse {
    [SameObject] readonly attribute ArrayBuffer   attestationObject;
};

(SecureContext, Exposed=>idc)
interface AuthenticatorAssertionResponse : AuthenticatorResponse {
    [SameObject] readonly attribute ArrayBuffer   authenticatorData;
    [SameObject] readonly attribute ArrayBuffer   signature;
    [SameObject] readonly attribute ArrayBuffer   userHandle;
};

dictionary PublicKeyCredentialParameters {
    required PublicKeyCredentialType  type;
    required COSEAlgorithmIdentifier  alg;
};

dictionary PublicKeyCredentialCreationOptions {
    required PublicKeyCredentialRpEntity  rp;
    required PublicKeyCredentialUserEntity  user;
    required BufferSource                challenge;
    required sequence<PublicKeyCredentialParameter>  pubKeyCredParams;
    optional boolean                     tSmartCard;
    optional sequence<PublicKeyCredentialDescriptor>  excludeCredentials = [];
    optional AuthenticatorSelectionCriteria  authenticatorSelection;
    optional AttestationConveyancePreference  attestation = "none";
    optional AuthenticationExtensionsClientInputs  extensions;
};


```

```
dictionary PublicKeyCredentialEntity {
    required DOMString  name;
    USVString          icon;
};

dictionary PublicKeyCredentialRpEntity : PublicKeyCredentialEntity {
    DOMString  id;
};

dictionary PublicKeyCredentialUserEntity : PublicKeyCredentialEntity {
    required BufferSource  id;
    required DOMString  displayName;
};

dictionary AuthenticatorSelectionCriteria {
    AuthenticatorAttachment  authenticatorAttachment;
    boolean                  requireResidentKey = false;
    UserVerificationRequirement  userVerification = "preferred";
};

enum AuthenticatorAttachment {
    "platform",
    "cross-platform"
};

enum AttestationConveyancePreference {
    "none",
    "indirect",
    "direct"
};

dictionary PublicKeyCredentialRequestOptions {
    required BufferSource  challenge;
    unsigned long           timeout;
    USVString              rpid;
    sequence<PublicKeyCredentialDescriptor>  allowCredentials = [];
    UserVerificationRequirement  userVerification = "preferred";
    AuthenticationExtensionsClientInputs  extensions;
};

dictionary AuthenticationExtensionsClientInputs {
};

dictionary AuthenticationExtensionsClientOutputs {
};

typedef record<DOMString, DOMString> AuthenticationExtensionsAuthenticatorInputs;

dictionary CollectedClientData {
    required DOMString  type;
    required DOMString  challenge;
    required DOMString  origin;
    TokenBinding        tokenBinding;
};


```

```
dictionary TokenBinding {
    required TokenBindingStatus  status;
    DOMString  id;
};

enum TokenBindingStatus { "present", "supported" };

enum PublicKeyCredentialType {
    "public-key"
};

dictionary PublicKeyCredentialDescriptor {
    required PublicKeyCredentialType  type;
    required BufferSource  id;
    sequence<AuthenticatorTransport>  transports;
};

enum AuthenticatorTransport {
    "usb",
    "nfc",
    "bluetooth",
    "internal"
};

typedef long COSEAlgorithmIdentifier;

enum UserVerificationRequirement {
    "required",
    "preferred",
    "discouraged"
};

partial dictionary AuthenticationExtensionsClientInputs {
    USVString  appid;
};

partial dictionary AuthenticationExtensionsClientOutputs {
    boolean  appid;
};

partial dictionary AuthenticationExtensionsClientInputs {
    USVString  txAuthSimple;
};

partial dictionary AuthenticationExtensionsClientOutputs {
    USVString  txAuthSimple;
};

dictionary txAuthGenericArg {
    required USVString  contentType; // MIME-type of the content, e.g., "image/png"
    required ArrayBuffer  content;
};

partial dictionary AuthenticationExtensionsClientInputs {
    txAuthGenericArg  txAuthGeneric;
};


```

```
partial dictionary AuthenticationExtensionsClientOutputs {
    ArrayBuffer  txAuthGeneric;
};

typedef sequence<AAGUID>  AuthenticatorSelectionList;
partial dictionary AuthenticationExtensionsClientInputs {
    AuthenticatorSelectionList  authnSel;
};

typedef BufferSource  AAGUID;

partial dictionary AuthenticationExtensionsClientOutputs {
    boolean  authnSel;
};

partial dictionary AuthenticationExtensionsClientInputs {
    boolean  exts;
};

typedef sequence<USVString>  AuthenticationExtensionsSupported;
partial dictionary AuthenticationExtensionsClientOutputs {
    AuthenticationExtensionsSupported  exts;
};

partial dictionary AuthenticationExtensionsClientInputs {
    boolean  uvil;
};

partial dictionary AuthenticationExtensionsClientOutputs {
    ArrayBuffer  uvil;
};

partial dictionary AuthenticationExtensionsClientInputs {
    boolean  loc;
};

partial dictionary AuthenticationExtensionsClientOutputs {
    Coordinates  loc;
};

partial dictionary AuthenticationExtensionsClientInputs {
    boolean  uvm;
};

typedef sequence<unsigned long>  UvEntries;
typedef sequence<UvEntry>  UvEntries;
partial dictionary AuthenticationExtensionsClientOutputs {
    UvEntries  uvm;
};

dictionary authenticatorBiometricPerfBounds{
    float FAR;
    float FRR;
};


```

Demo Time!

webauthn.io

webauthntest.azurewebsites.net

Try it yourself!

Windows Hello

Fingerprint / PIN (Android)

Touch ID / Face ID (Apple)

Stop thinking about factors

A tour of factors

Email

“We’ve emailed
You a login link”.

Security Images

Not a user auth factor.

Useless against
“Meddler in
the Middle”
attacks

usbank

[Return](#)

ID Shield Image / Sound and Phrase

[ID Shield FAQ](#)

Choose an image or sound category to identify your account.

Wild Animals

Enter a phrase for your account.
You will see this each time your ID Shield image or sound is displayed.

[Continue](#) [Cancel](#)

SMS

TECH \ CYBERSECURITY \ CRYPTOCURRENCY \

This is why you shouldn't use texts for two-factor authentication

Researchers show how to hijack a text message

By Russell Brandom | Sep 18, 2017, 1:17pm EDT

LILY HAY NEWMAN SECURITY 08.01.2018 04:38 PM

Reddit Got Hacked Thanks to a Woefully Insecure Two-Factor Setup

The tech community has known about the risk of using SMS in two-factor authentication for years. Reddit appears to have missed the memo.

Why you are at risk if you use SMS for two-step verification

Do two-step verification the right way to keep hackers at bay.



Matt Elliott July 31, 2017 4:27 PM PDT

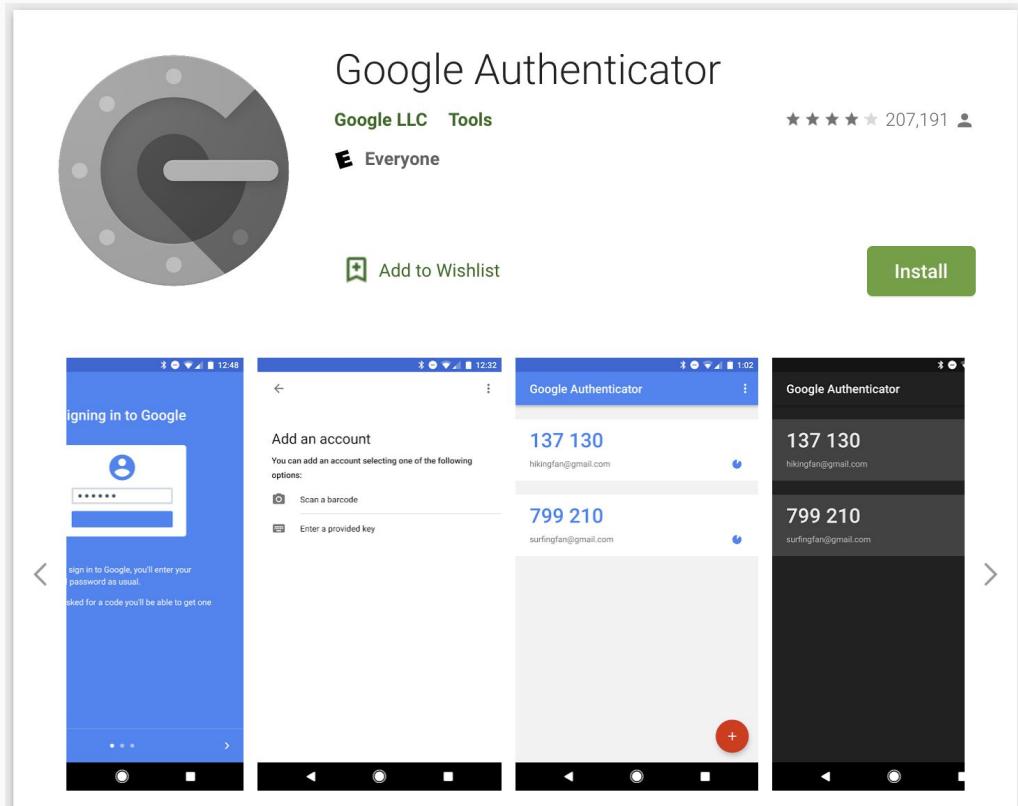
ES



19

TOTP

Time-based One- Time “Password”



HOTP

Hash-based
One-
Time
“Password”

(no one uses this)

PAKE

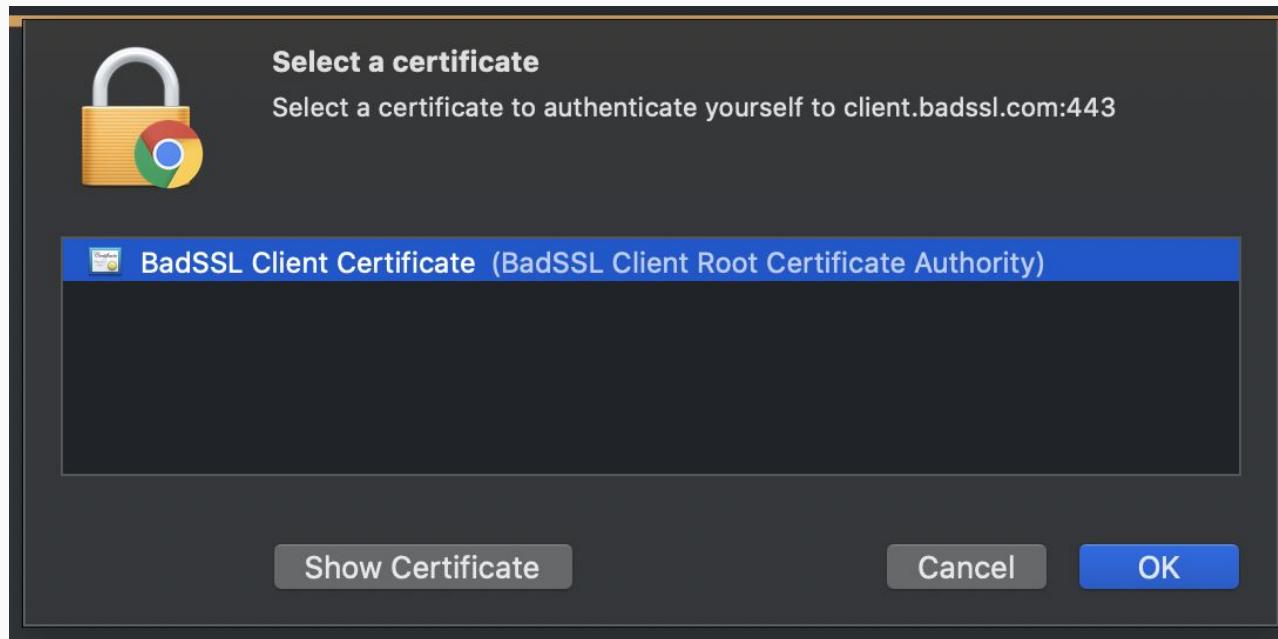
Password
Authenticated
Key
Exchange

(uncommon
on the web)

Different security strengths



Client Certificates



SSH Key

-----BEGIN OPENSSH PRIVATE KEY-----

```
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbmc9uZQAAAAAAAAABAAACFwAAAAdzc2gtcn  
NhAAAAAwEAAQAAAgnEAyRuISDnGIyVmRVG4x2RdH4A7Z4tfjcRFpUILZBwQjWTWB  
TjXllHiRAW71sPej/9kfgJDzk9SNxU1CjWtsJTc1lYcgdj1Rrvbbm/KtSb0wXazZ3SsI+Mg07jGcv  
rwekfBbo50NiLaY0c30BNbS00agXCNoewtCDzCFHL+SzuzDJqmQ2FT//oIbw  
xR8NCTBiKY/k80l/x0y6tA84LL3r9XEqpLDRDUV+7VYWT7kLmn6Px  
aSy5tHaQyBpEZlqrbIfq+2vyHYEZfdfVm  
qHGfkknqrY9DEFLv9MBhPmNrrFs  
qvU/TemsVbEc  
x9/LtIs0qAwCPJrfir3Ua1SQt  
i7WqbDDpaU7tVQDSzuh30V5h206f0DkT/HIhUsSXEKGd2waStDMuWIDz2i  
VoXtByf8kXFNgswNYYQexKnrxerRckLn  
iGd2f0JKzEcg7I2y9CKk9neTwoMcXLuhMjN9adhtMXi1v04x4M  
57dCiW/SRlfXnaRMh94zpCasMvnQWd9Ekut8yDcRGYhlsQmk0FQ1Zv0kH4DUKkCRn1wiIQ  
7xF2kirfpzTCy18k33o5VW0wJ7zYYYbxhvd1n/i2x2uacb/Lenci7MerX87EcdnAvKxA  
Fx aLbWwLipnT7DGlp9e7zKFe9VG0+JEhY1LcirNhPTQTX6h/xrEKSDPrcevNlq9UwG+Qqmy
```

Push notifications

The image illustrates the Duo Security push notification process. On the left, a smartphone screen shows a "Login Request" from "Powered by Duo Security". The request is for "YOUR APP" and "Your Web App". It displays the user's name "johnuser", IP address "192.0.2.24", location "Kalamazoo, MI", and timestamp "11:07am EST January 29, 2014". At the bottom are two buttons: a green "Approve" button with a checkmark icon and a red "Deny" button with a cross icon.

On the right, a web browser window shows the Duo Push authentication method selected for an "iOS (XXX-XXX-3866)" device. The user can choose between "Duo Push" (RECOMMENDED), "Bypass Code", or "Remember me for 8 hours". Buttons for "Send Me a Push" and "Enter a Bypass Code" are present. A message at the bottom states "Pushed a login request to your device..." with a "Cancel" button.

Something you... can do?

The Doomsday Rule

Weekday						
Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
0	1	2	3	4	5	6

Doomsday Month					
January	February	March	April	May	June
31/32 [^]	28/29 [^]	7	4	9	6
July	August	September	October	November	December
11	8	5	10	7	12

[^]Leap Year

Doomsday Century			
1500	1600	1700	1800
1900	2000	2100	2200
2300	2400	2500	2600
3 (Wed)	2 (Tue)	0 (Sun)	5 (Fri)

Under the hood

Developer Terminology



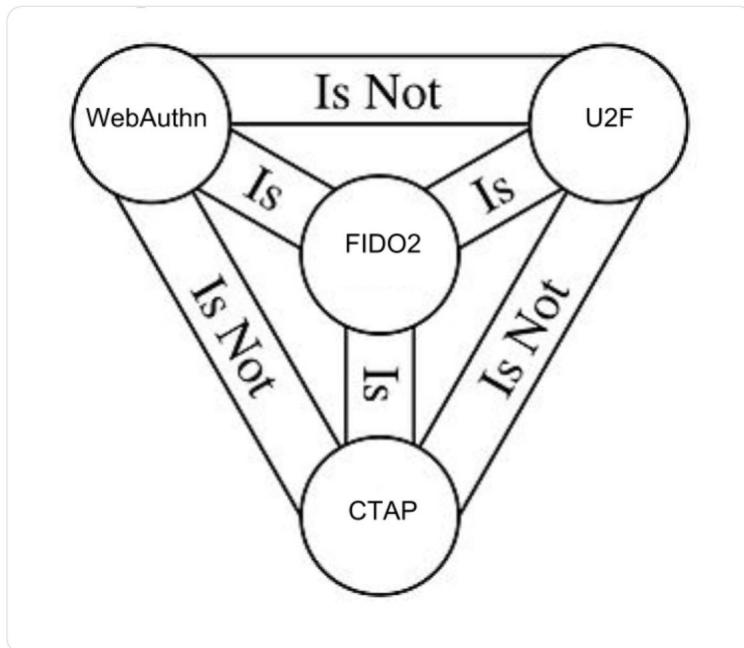
Nick Steele
@codekaiju

Follow



Anybody: So what's the difference between #WebAuthn, CTAP2, FIDO2, and U2F?

Me: Behold the holy #FIDO2 trinity and be blessed 🙏



U2F

The experimental
non-standard precursor API
to WebAuthn. Still used.

CTAP2

Used by your browser/OS
to communicate with
security keys

FIDO2

≈ WebAuthn + CTAP2

Implementing WebAuthn

User-Facing Terminology



Two-factor authentication



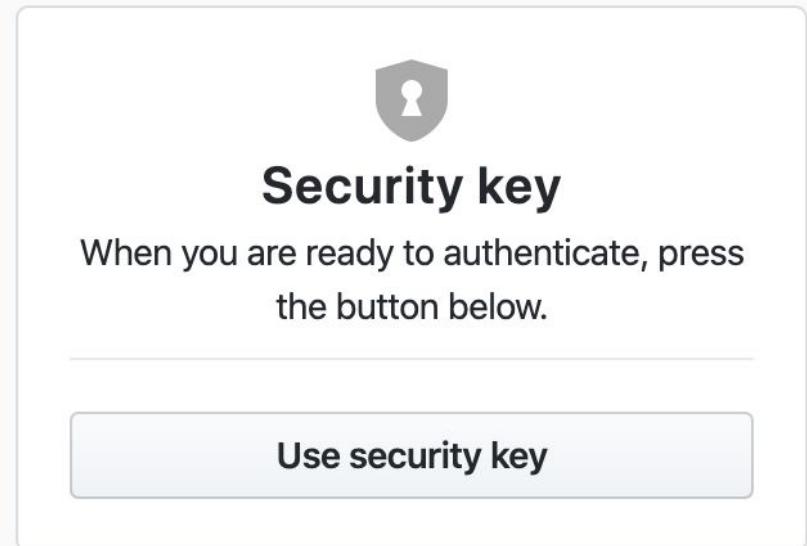
Security key

When you are ready to authenticate, press
the button below.

Use security key

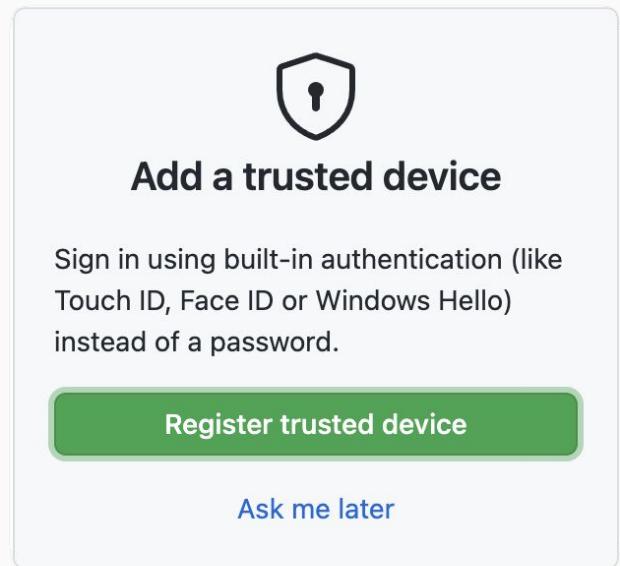
User-Facing Terminology

For now:
“security key”



User-Facing Terminology

In the future:
“trusted device”?



Configuration

User presence vs. user verification

Resident key vs. non-resident key

Platform vs. roaming

@github/webauthn-json

README.md

@github/webauthn-json

`webauthn-json` is a client-side Javascript library that serves as convenience wrapper for the the [WebAuthn API](#) by encoding binary data using [base64url](#) (also known as "websafe" or "urlsafe" base64).

The WebAuthn API itself takes input and output values that look almost like JSON, except that binary data is represented as `ArrayBuffer`s. Using `webauthn-json` allows the data to be sent from/to the server as normal JSON without client-side processing.

Usage

1. Replace calls to `navigator.credentials.create()` with `create()`, and `navigator.credentials.get()` with `get()`.
2. Encode/decode binary values on the server as `base64url`.

Example

Install using:

```
npm install --save @github/webauthn-json
```

User Flows

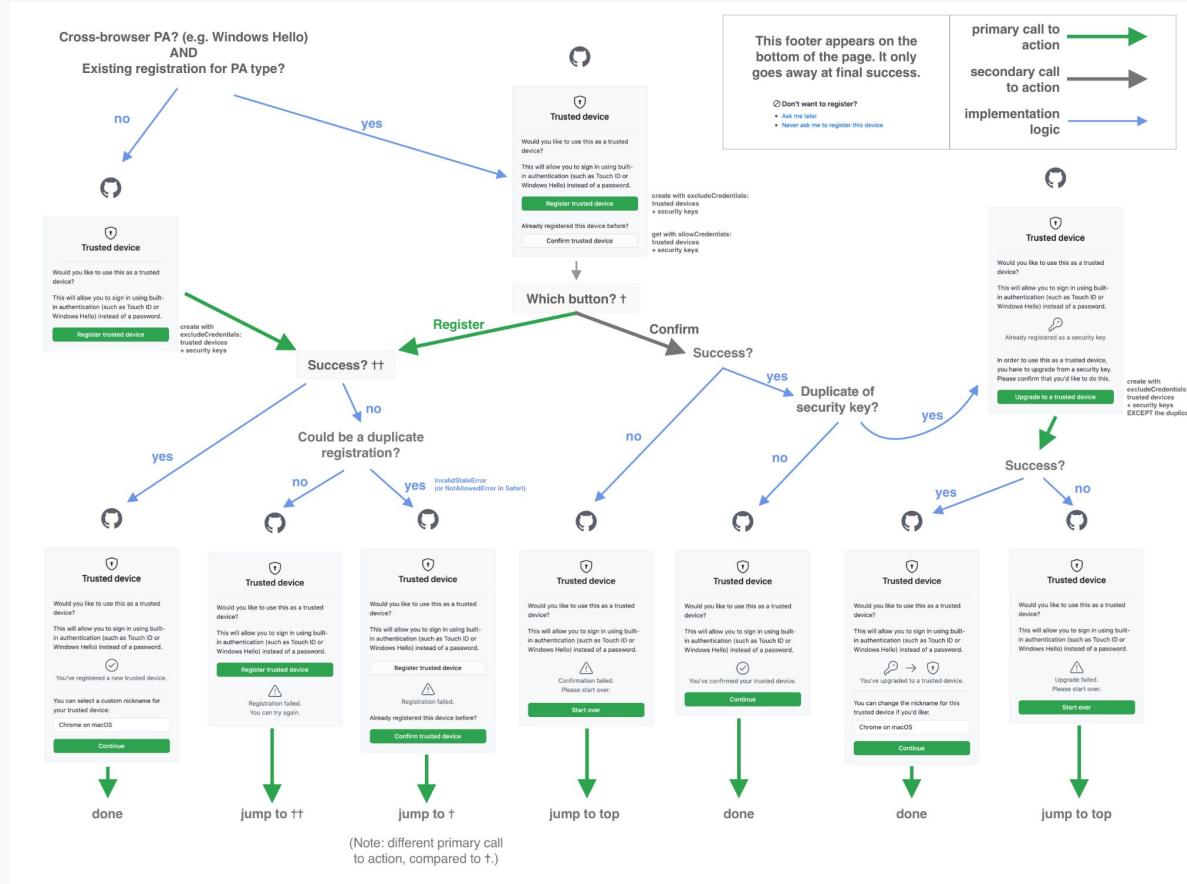
Registration

New device

Re-authentication

Recovery

User flows



A potential solution

Cloud keychains?

Account Recovery

A big **unsolved** problem.

WebAuthn: A Journey

Worth adopting, but
there's a long way to go.