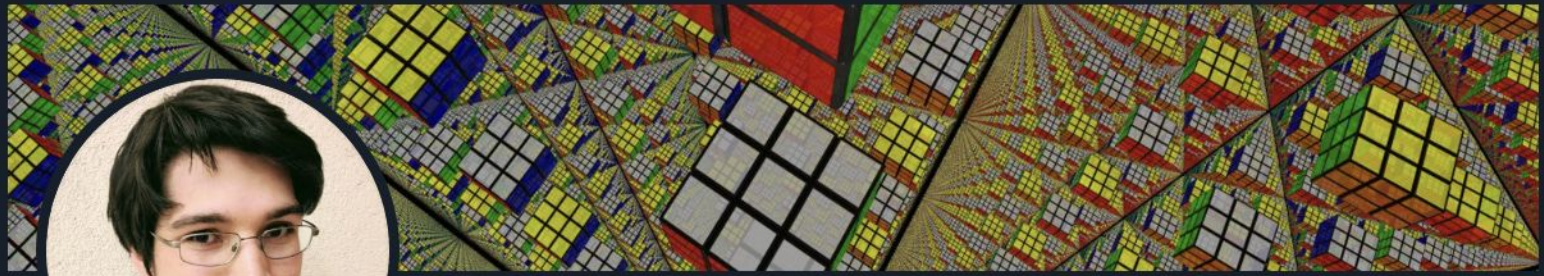
WebAuthn

The future of user authentication on the web 🤞

Lucas Garron
CS 253 Guest Talk
2019-11-06



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) 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

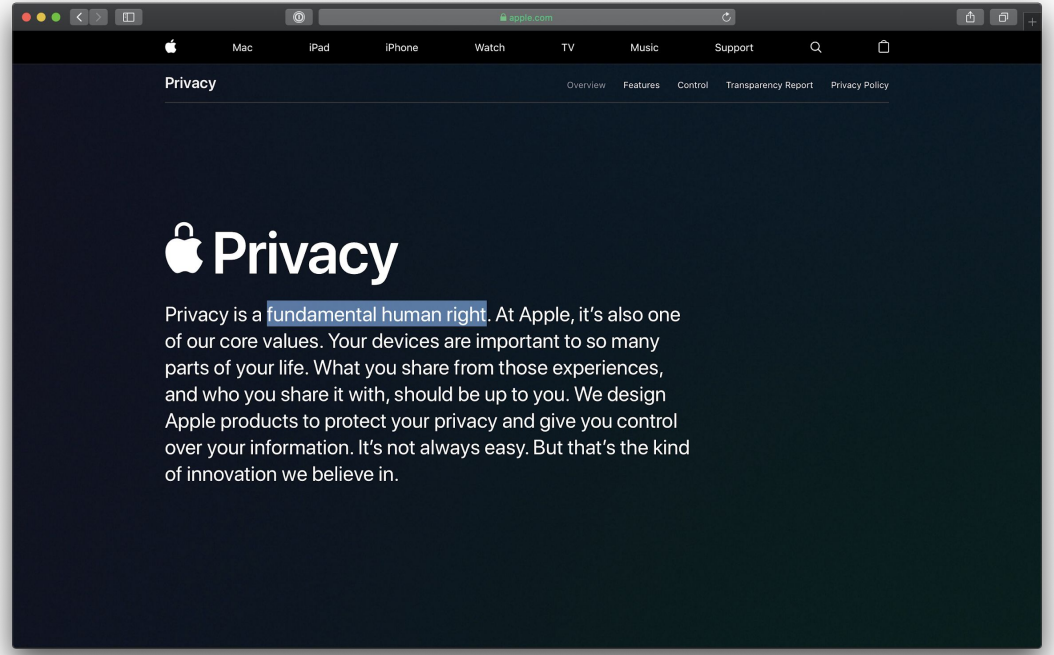
Previously, GitHub supported physical security keys using the experimental U2F API for Chrome. WebAuthn is the standards-based successor. You can now use physical security keys on GitHub with:

- 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


DROP
ICE

Security and Privacy are not “add-on features”



Passwords (Redux)

“Use `bcrypt`”

Terribly phishable

HaveIBeenPwned.com

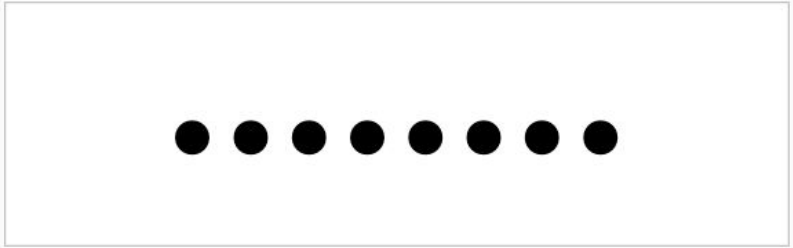
Authentication Factors

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=Window]
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=Window]
interface AuthenticatorResponse {
    [SameObject] readonly attribute ArrayBuffer      clientDataJSON;
};

[SecureContext, Exposed=Window]
interface AuthenticatorAttestationResponse : AuthenticatorResponse {
    [SameObject] readonly attribute ArrayBuffer      attestationObject;
};

[SecureContext, Exposed=Window]
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<PublicKeyCredentialParameters>  pubKeyCredParams;

    unsigned long                               timeout;
    sequence<PublicKeyCredentialDescriptor>     excludeCredentials = [];
    AuthenticatorSelectionCriteria              authenticatorSelection;
    AttestationConveyancePreference             attestation = "none";
    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",
    "ble",
    "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 uvi;
};

partial dictionary AuthenticationExtensionsClientOutputs {
  ArrayBuffer uvi;
};

partial dictionary AuthenticationExtensionsClientInputs {
  boolean loc;
};

partial dictionary AuthenticationExtensionsClientOutputs {
  Coordinates loc;
};

partial dictionary AuthenticationExtensionsClientInputs {
  boolean uvm;
};

typedef sequence<unsigned long> UvmEntry;
typedef sequence<UvmEntry> UvmEntries;

partial dictionary AuthenticationExtensionsClientOutputs {
  UvmEntries uvm;
};

dictionary authenticatorBiometricPerfBounds{
    float FAR;
    float FRR;
    };
```

https://www.w3.org/TR/webauthn/#idl-index

Demo Time!

webauthn.io

webauthntest.azurewebsites.net

Try it yourself!

Windows Hello

Fingerprint (Android)

Touch ID (Chrome macOS)

Stop thinking about factors

A tour of factors

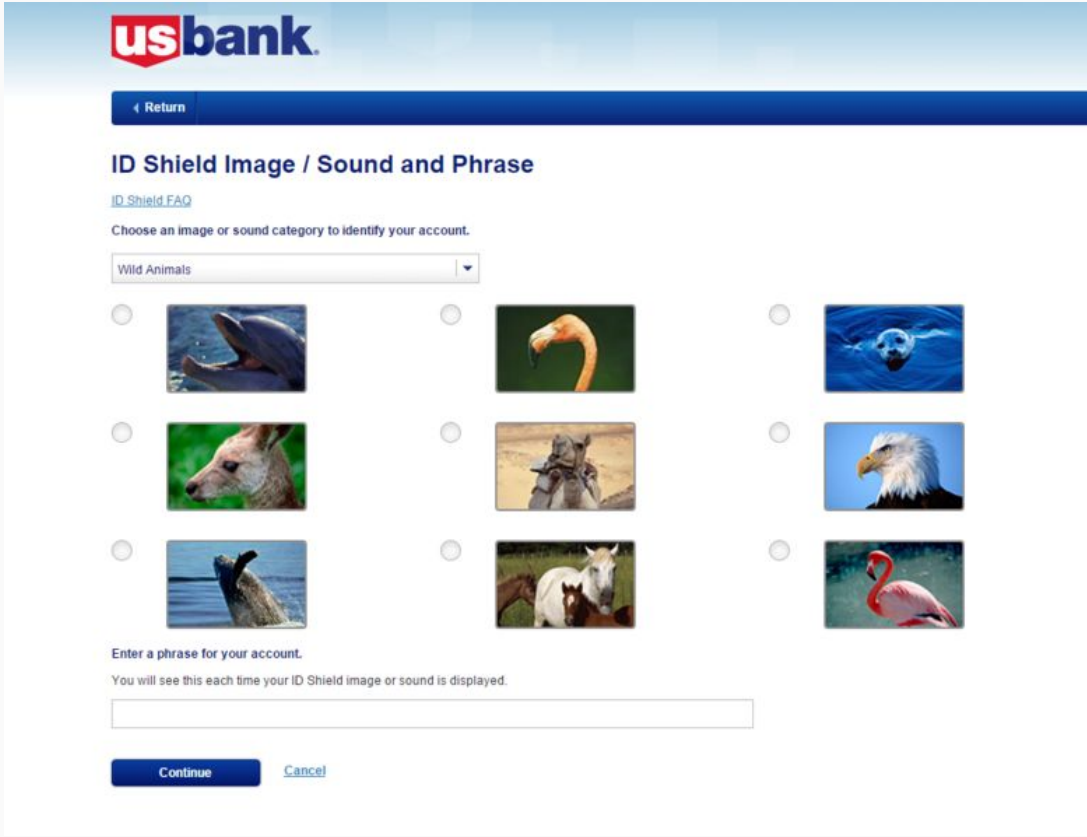
Email

“We’ve emailed
You a login link”.

Security Images

Not a user auth factor.

Useless against “Meddler in the Middle” attacks



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.

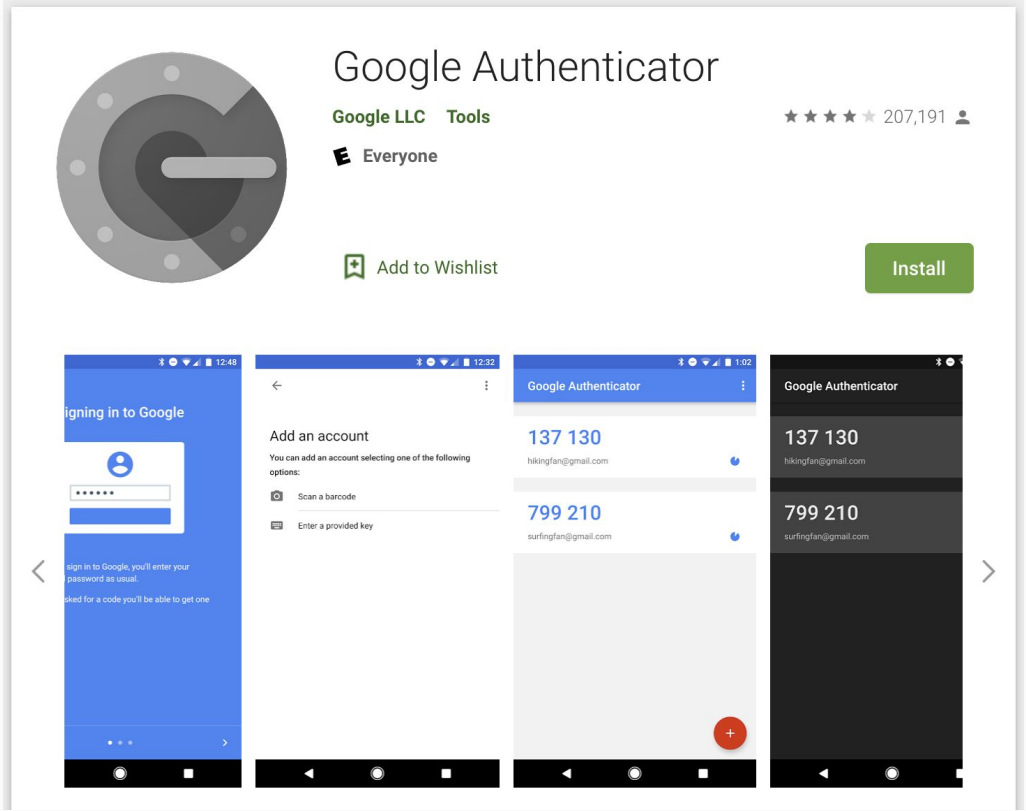
Why you are at risk if you use SMS for two-step verification

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)

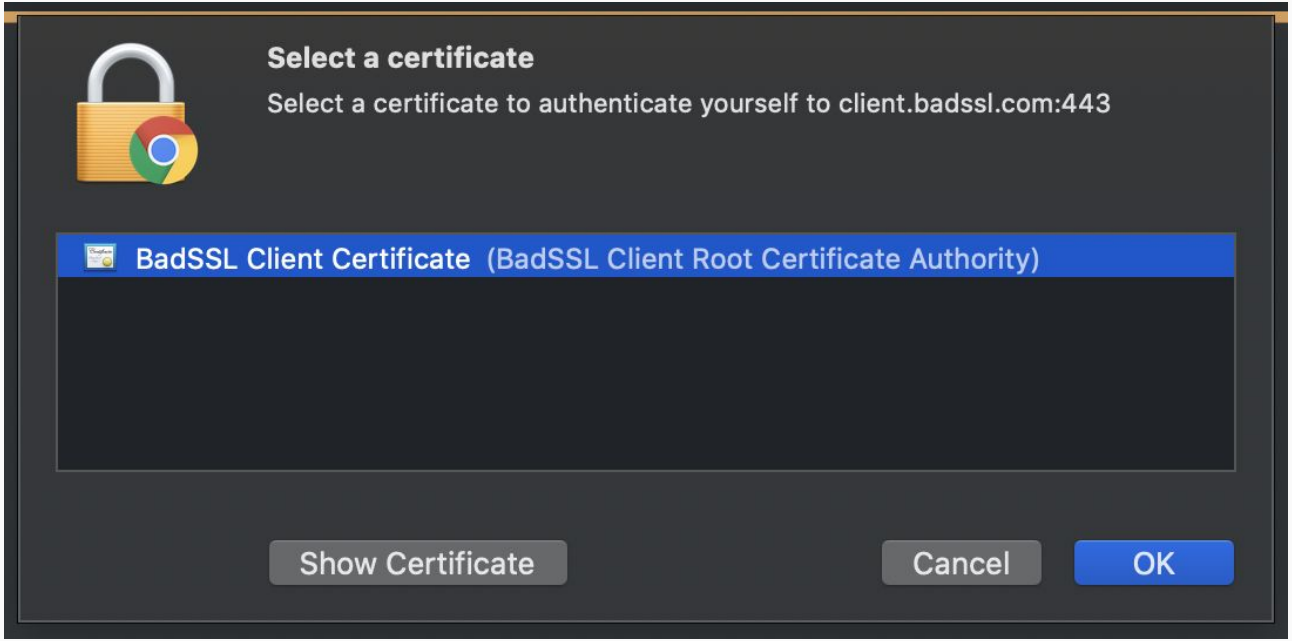
PAKE

Password Authenticated Key Exchange

(uncommon on the web)

Different security strengths

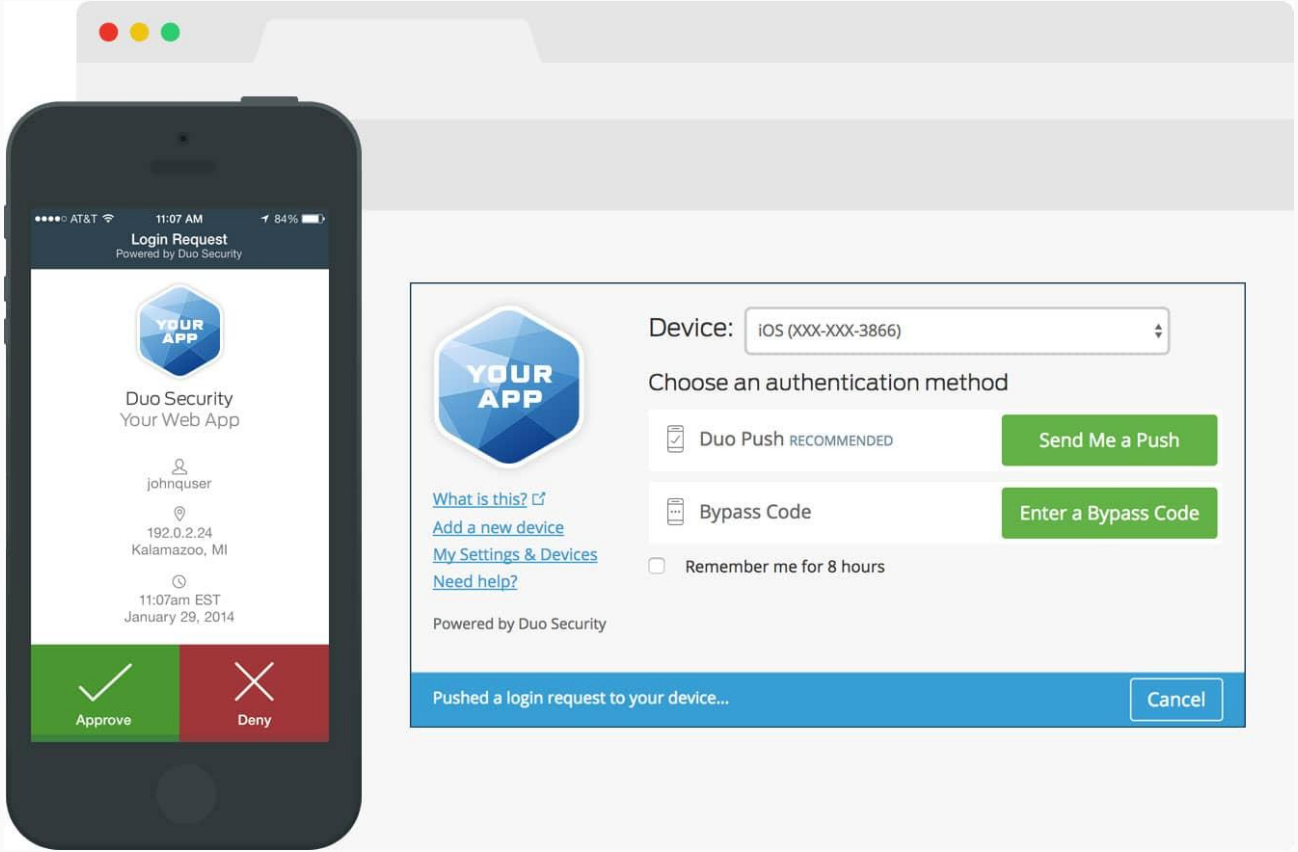
Client Certificates



SSH Key

```
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAACFwAAAAdzc2gtcn
NhAAAAAwEAAQAAAgEAyRuISDnGIyVmRVG4x2RdH4A7Z4tfjcRFpUIlZBwQjWTWBTjXllHi
RAW71sPej/9kfgJDzk9SNxU1CjWtsJTcllYcgdjlRrvbbm/KtSbOWXazZ3SsI+Mg07jGcv
rwekfBbo50NiLaYOc3OBNbSOOagXCNoeWtCDzCFHL+SzuzDJqmQ2FT//oIbWxR8NCTBiKY
/k8Ol/x0y6tA84LL3r9XEqpLDRDUV+7VYWT7kLmn6PxaSy5tHaQyBpEZlqrbIfq+2vyHYE
ZfdfVmqHGfkknqrY9DEfLv9MBhPmNrrFsqVu/TemsVbEcx9/LtIsOqAwCPJrfir3Ua1SQt
i7WqbDDpaU7tVQDSzuh3OV5h206fODkT/HIhUsSXEKGd2waStDMuWIDz2iVoXtByf8kXFN
gswNYYQexKnrxerRckLniGd2fOJKzEcg7I2y9CKk9neTwoMcXLuhMjN9adhtMXi1v04x4M
57dCiW/SRlfXnaRMh94zpCasMvnQWd9Ekut8yDcRGYhlsQmkOFQ1Zv0kH4DUKkCRn1wiIQ
7xF2kirfpzTCy18k33o5VWOwJ7zYYYbxhvd1n/i2x2uacb/Lenci7MerX87EcdnAvKxAFx
aLbWwLipnT7DGlzp9e7zKFe9VG0+JEhY1LcirNhPTQTX6h/xrEKSDPrcevNlq9UwG+Qqmy
```

Push notifications



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)

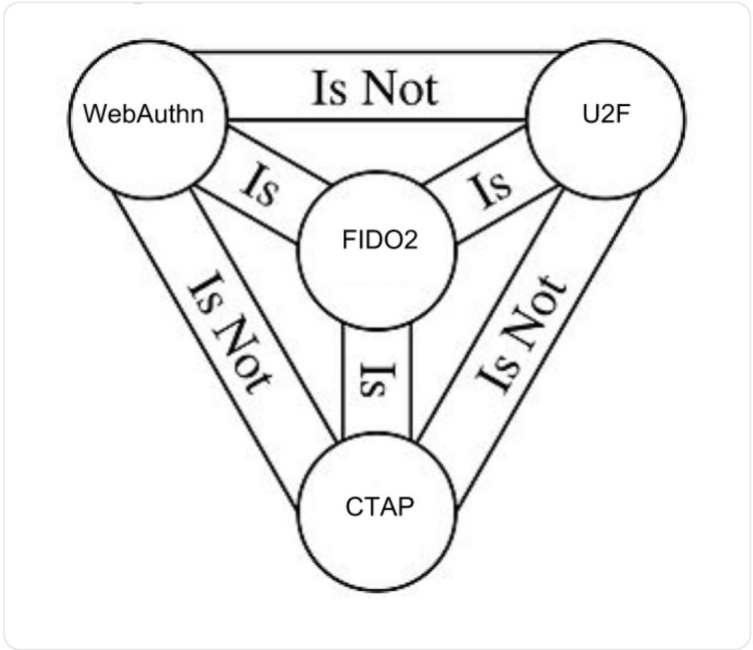
Under the hood

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🙏



2:17 PM - 1 Oct 2019

U2F

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

CTAP2

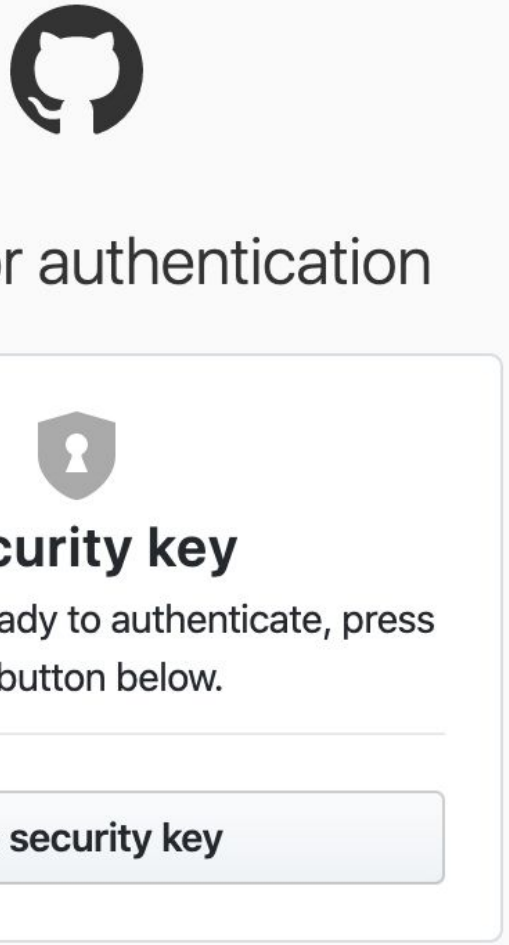
Used by your browser/OS to communicate with security keys

FIDO2

≈ WebAuthn + CTAP2

Implementing WebAuthn

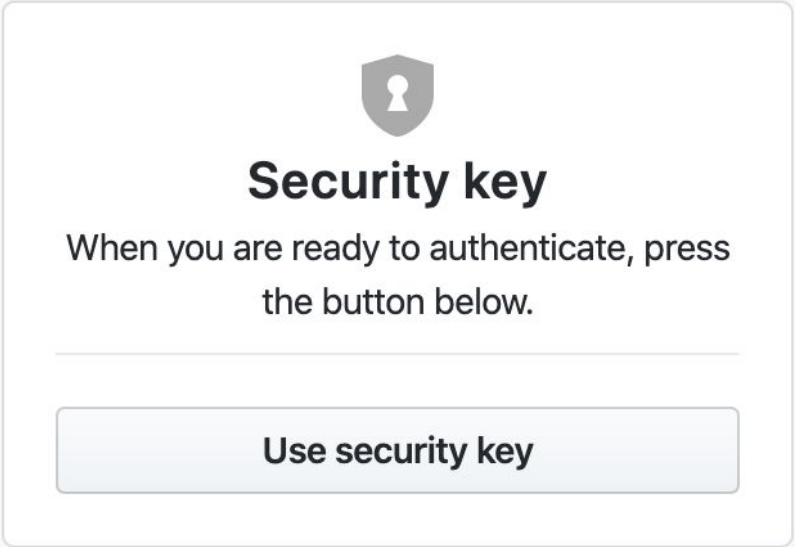
User-Facing Terminology



User-Facing Terminology

For now:

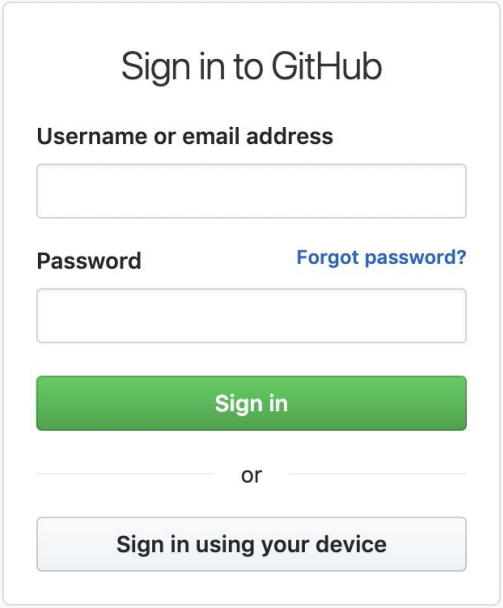
“security key”



User-Facing Terminology

In the future:

“using your 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

Account Recovery

A big **unsolved** problem.

WebAuthn: A Journey

Worth adopting, but

there’s a long way to go.