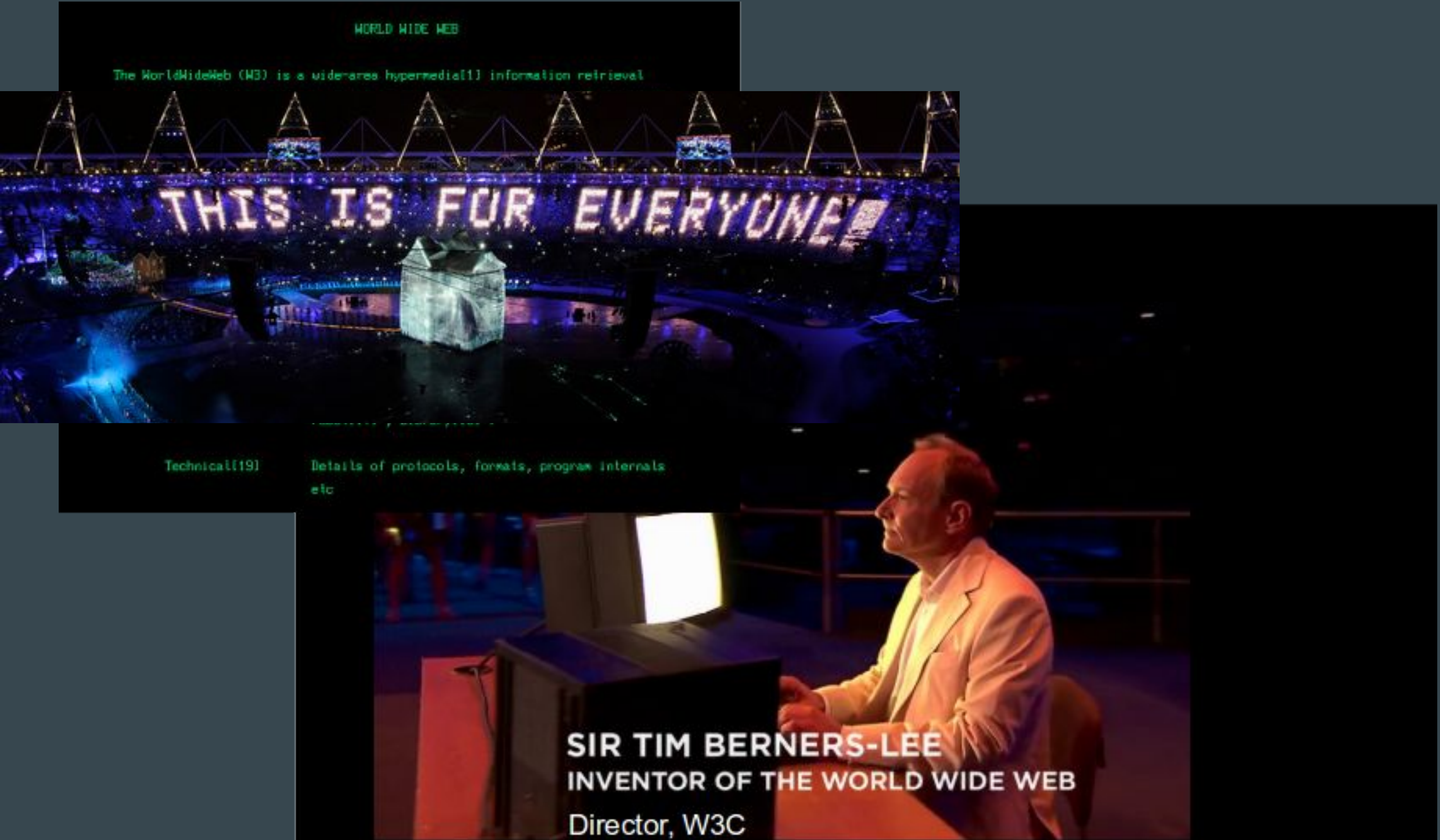
# WebAuthn: Beyond the Password

Web Security @W3C/MIT CSAIL
Wendy Seltzer, wseltzer@w3.org



```
                         WORLD WIDE WEB

   The WorldWideWeb (W3) is a wide-area hypermedia[1] information retrieval

          Technical[19]      Details of protocols, formats, program internals
                             etc
```

SIR TIM BERNERS-LEE
INVENTOR OF THE WORLD WIDE WEB
Director, W3C

# World Wide Web Consortium (W3C)

Voluntary standard-setting. Stewards of the Open Web Platform.

Addressing the collective action challenge of Web security

## Modular security

- Component by component (end-to-end)
- Foundations for trust and secure communication

## Incentives: keep the platform working jointly, compete on top

# Why Web Authentication? @#&!?%)??

Passwords annoy users:

- Prompts interrupt the flow of activity (Web purchase, posting, reading, or interaction)
- Entry is even more annoying on mobile
- Passwords are forgettable. Password-generation rules make management harder.
- Some sites block password manager auto-fill.

Passwords are insecure:

- Reuse across sites can mean break-once-break-anywhere
- Vulnerable to interception (phishing) that can compromise accounts
- Trade-off between memorable/enterable and vulnerable to brute-force guessing

# Better authentication improves user experience and security

- Faster log-in means faster check-out
- Happier users return more frequently
- Strong authentication leads to greater accountability

Consequences of passwords

- 2013 Yahoo! breach compromised all 3 Billion user accounts (passwords were weakly encrypted)
- 2018 Twitter warned all users to change their passwords because they were stored in plaintext

# Web Authentication

Security

- Strong cryptography
- Unphishable
- Resists data-breach and brute force attacks
- Test of user presence
- Attestation

Usability

- Passwordless
- One- or two-factor
- In-device, biometric

# Web Authentication

fido™ ALLIANCE
simpler
stronger
authentication

- Member Submission of FIDO2 work to W3C
- Continued work on CTAP (Client to Authenticator Protocol)

W3C®

- Web API: Enable the browser to mediate between client-side authenticator and web applications

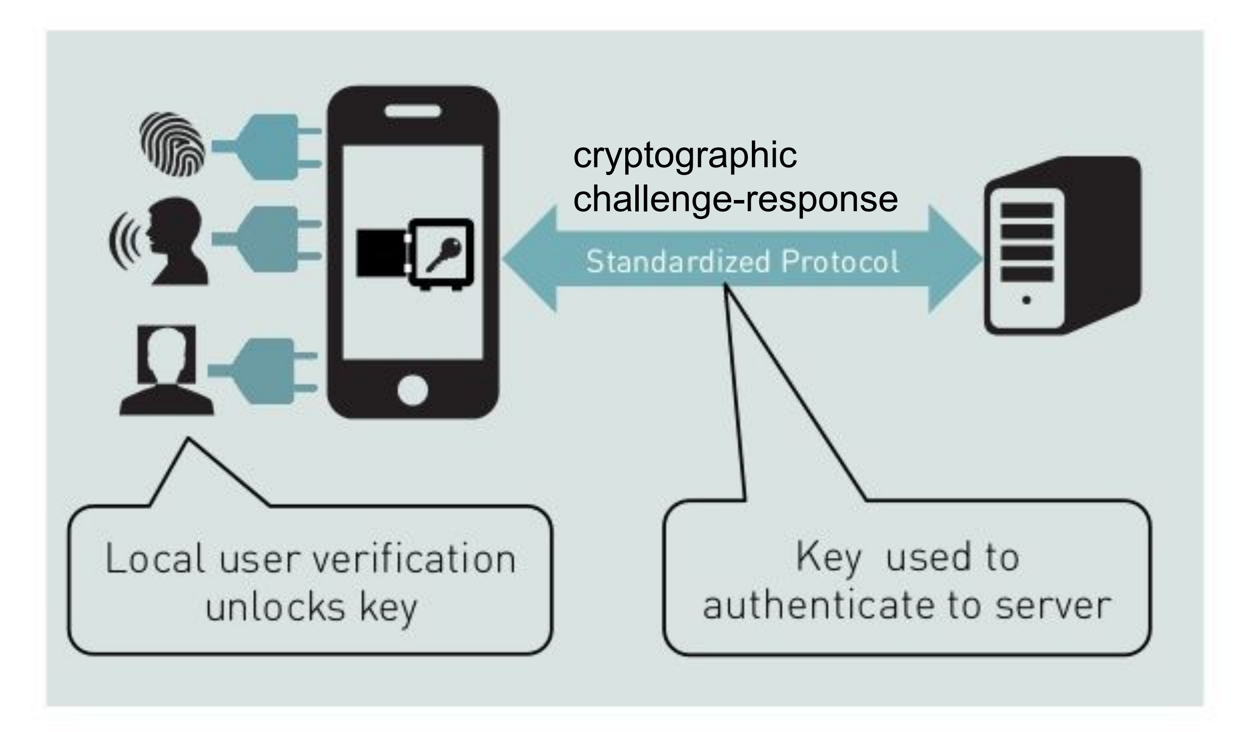
WebAuthn

# Web Authentication: How it works

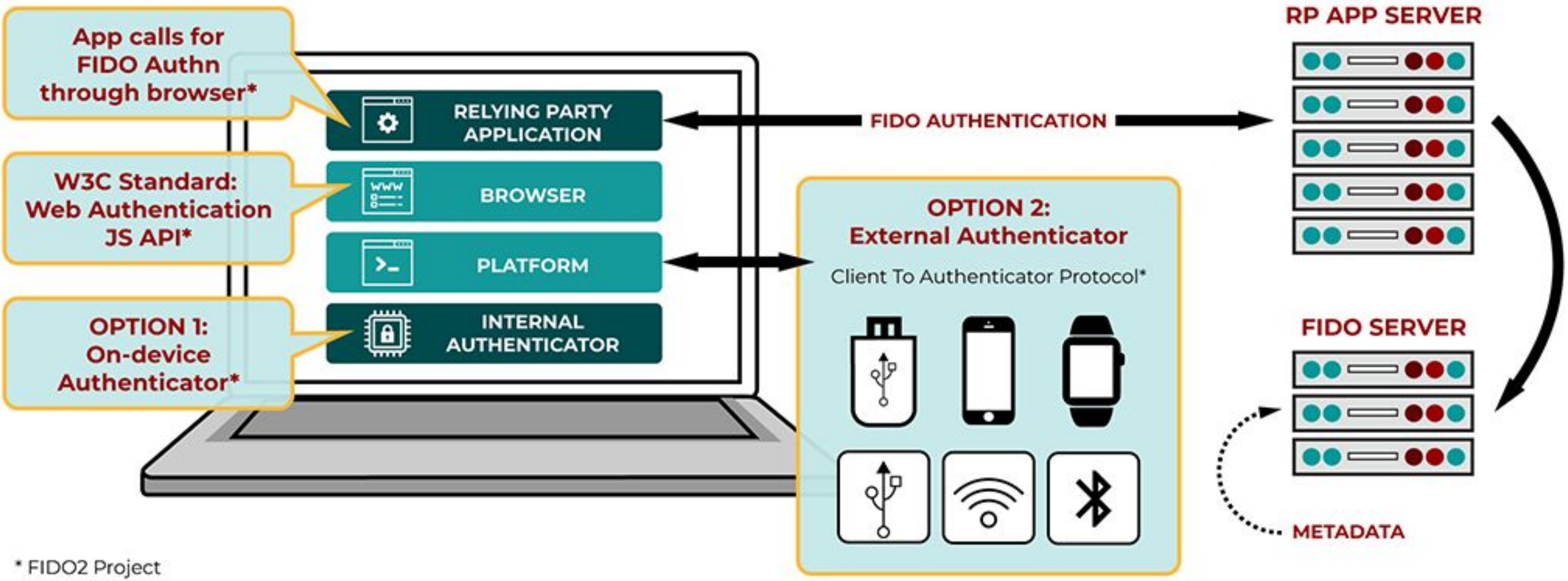
WebAuthn enables a cryptographic challenge **unique** to each website and **bound** to its origin.

Local authentication such as biometrics never leaves the device.

https://www.w3.org/TR/webauthn/


cryptographic
challenge-response
Standardized Protocol
Local user verification unlocks key
Key used to authenticate to server

# W3C WebAuthn with FIDO


App calls for FIDO Authn through browser*
W3C Standard: Web Authentication JS API*
OPTION 1: On-device Authenticator*
RELYING PARTY APPLICATION
BROWSER
PLATFORM
INTERNAL AUTHENTICATOR
FIDO AUTHENTICATION
RP APP SERVER
OPTION 2: External Authenticator
Client To Authenticator Protocol*
FIDO SERVER
METADATA

* FIDO2 Project

# Demos

https://webauthn.org

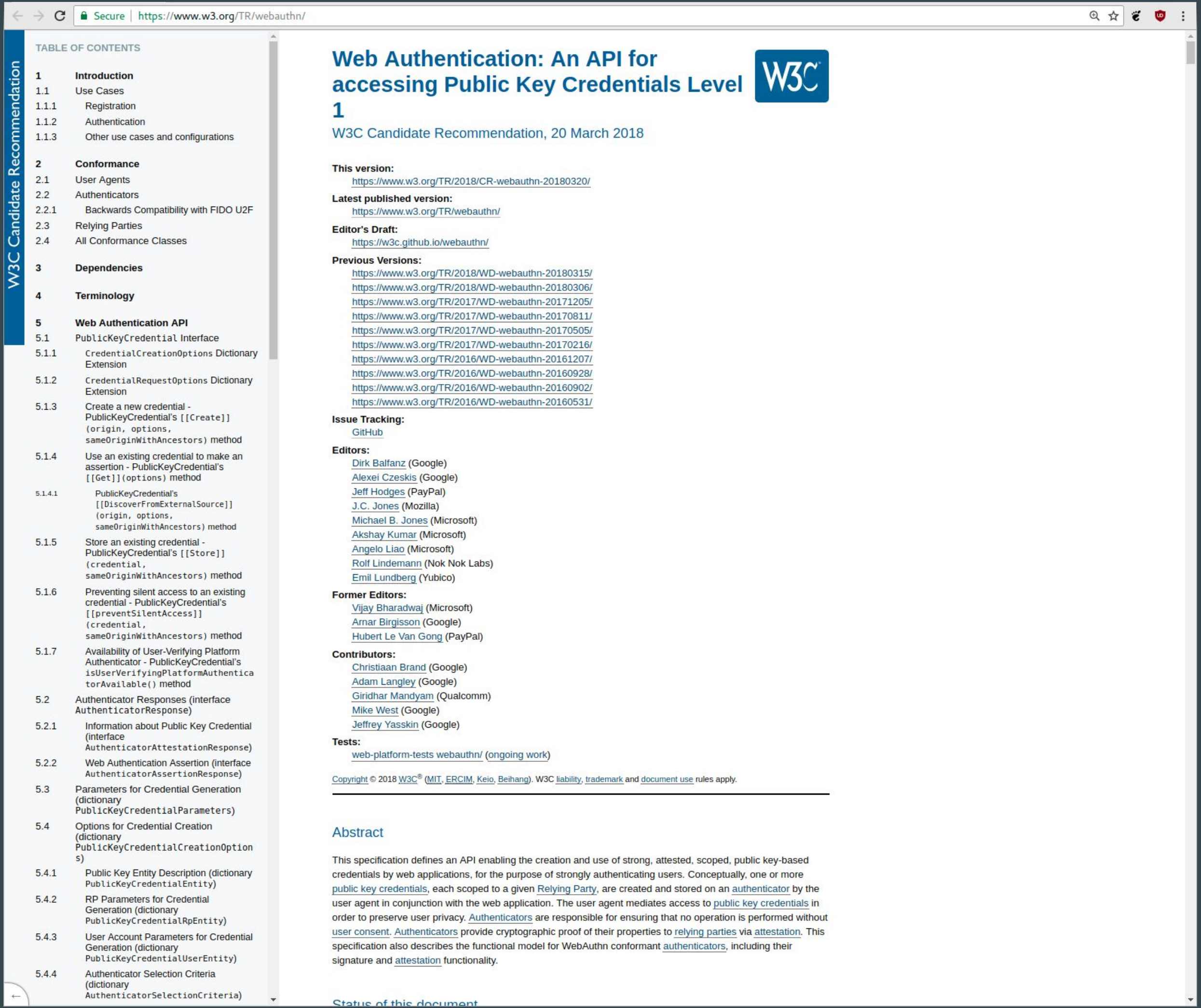
https://webauthn.io

WebAuthn

# WebAuthn at Candidate Rec.

W3C Working Group Chairs: Tony Nadalin, Microsoft, and John Fontana, Yubico

https://github.com/w3c/webauthn



Secure | https://www.w3.org/TR/webauthn/

W3C Candidate Recommendation

TABLE OF CONTENTS

1 **Introduction**
1.1 Use Cases
1.1.1 Registration
1.1.2 Authentication
1.1.3 Other use cases and configurations

2 **Conformance**
2.1 User Agents
2.2 Authenticators
2.2.1 Backwards Compatibility with FIDO U2F
2.3 Relying Parties
2.4 All Conformance Classes

3 **Dependencies**

4 **Terminology**

5 **Web Authentication API**
5.1 `PublicKeyCredential` Interface
5.1.1 `CredentialCreationOptions` Dictionary Extension
5.1.2 `CredentialRequestOptions` Dictionary Extension
5.1.3 Create a new credential - PublicKeyCredential's `[[Create]](origin, options, sameOriginWithAncestors)` method
5.1.4 Use an existing credential to make an assertion - PublicKeyCredential's `[[Get]](options)` method
5.1.4.1 PublicKeyCredential's `[[DiscoverFromExternalSource]](origin, options, sameOriginWithAncestors)` method
5.1.5 Store an existing credential - PublicKeyCredential's `[[Store]](credential, sameOriginWithAncestors)` method
5.1.6 Preventing silent access to an existing credential - PublicKeyCredential's `[[preventSilentAccess]](credential, sameOriginWithAncestors)` method
5.1.7 Availability of User-Verifying Platform Authenticator - PublicKeyCredential's `isUserVerifyingPlatformAuthenticatorAvailable()` method
5.2 Authenticator Responses (interface `AuthenticatorResponse`)
5.2.1 Information about Public Key Credential (interface `AuthenticatorAttestationResponse`)
5.2.2 Web Authentication Assertion (interface `AuthenticatorAssertionResponse`)
5.3 Parameters for Credential Generation (dictionary `PublicKeyCredentialParameters`)
5.4 Options for Credential Creation (dictionary `PublicKeyCredentialCreationOptions`)
5.4.1 Public Key Entity Description (dictionary `PublicKeyCredentialEntity`)
5.4.2 RP Parameters for Credential Generation (dictionary `PublicKeyCredentialRpEntity`)
5.4.3 User Account Parameters for Credential Generation (dictionary `PublicKeyCredentialUserEntity`)
5.4.4 Authenticator Selection Criteria (dictionary `AuthenticatorSelectionCriteria`)

## Web Authentication: An API for accessing Public Key Credentials Level 1

W3C Candidate Recommendation, 20 March 2018

**This version:**
https://www.w3.org/TR/2018/CR-webauthn-20180320/

**Latest published version:**
https://www.w3.org/TR/webauthn/

**Editor's Draft:**
https://w3c.github.io/webauthn/

**Previous Versions:**
https://www.w3.org/TR/2018/WD-webauthn-20180315/
https://www.w3.org/TR/2018/WD-webauthn-20180306/
https://www.w3.org/TR/2017/WD-webauthn-20171205/
https://www.w3.org/TR/2017/WD-webauthn-20170811/
https://www.w3.org/TR/2017/WD-webauthn-20170505/
https://www.w3.org/TR/2017/WD-webauthn-20170216/
https://www.w3.org/TR/2016/WD-webauthn-20161207/
https://www.w3.org/TR/2016/WD-webauthn-20160928/
https://www.w3.org/TR/2016/WD-webauthn-20160902/
https://www.w3.org/TR/2016/WD-webauthn-20160531/

**Issue Tracking:**
GitHub

**Editors:**
Dirk Balfanz (Google)
Alexei Czeskis (Google)
Jeff Hodges (PayPal)
J.C. Jones (Mozilla)
Michael B. Jones (Microsoft)
Akshay Kumar (Microsoft)
Angelo Liao (Microsoft)
Rolf Lindemann (Nok Nok Labs)
Emil Lundberg (Yubico)

**Former Editors:**
Vijay Bharadwaj (Microsoft)
Arnar Birgisson (Google)
Hubert Le Van Gong (PayPal)

**Contributors:**
Christiaan Brand (Google)
Adam Langley (Google)
Giridhar Mandyam (Qualcomm)
Mike West (Google)
Jeffrey Yasskin (Google)

**Tests:**
web-platform-tests webauthn/ (ongoing work)

### Abstract

This specification defines an API enabling the creation and use of strong, attested, scoped, public key-based credentials by web applications, for the purpose of strongly authenticating users. Conceptually, one or more public key credentials, each scoped to a given Relying Party, are created and stored on an authenticator by the user agent in conjunction with the web application. The user agent mediates access to public key credentials in order to preserve user privacy. Authenticators are responsible for ensuring that no operation is performed without user consent. Authenticators provide cryptographic proof of their properties to relying parties via attestation. This specification also describes the functional model for WebAuthn conformant authenticators, including their signature and attestation functionality.

### Status of this document

# WebAuthn Implementations

Browser implementations include:

- Chrome 67
- Firefox 60
- Edge development version
- Safari participating in the Working Group

**W3C WG Participants:** Airbnb, Alibaba Group, Apple, Bloomberg, Consensus, Deutsche Telekom, ETRI, Federal Reserve Bank of Minneapolis, Google, HM Government, IBM, Intel, Intuit, Microsoft, Mozilla, NIST, New Zealand Government, Nok Nok Labs, Opera Software AS, Orange, PayGate, PayPal, Qualcomm Innovation Center, SoftBank Corp., Tencent, Thomson Reuters, Trust1Team, Wiley, Yubico

# WebAppSec: Encryption Everywhere

## Standardizing and Enabling HTTPS for confidentiality, integrity, and authentication

- Secure Contexts
- Upgrade Insecure Requests
- Mixed Content
- Referrer Policy
- Subresource Integrity

## HTTPS Work Elsewhere

- Let’s Encrypt
- Certificate Transparency
- HSTS

# WebAppSec: Enlisting the User Agent in Cooperative Policy Enforcement

- Content Security Policy
  - Level 2 is Recommendation; Level 3 in development (Editor's Draft)
- Secure Contexts
- Subresource Integrity (Rec), Mixed Content

**Security Related APIs**

- Permissions API
- Credential Management
- Clear Site Data

**Experiments in the Web Security Model / Same Origin Policy**

- Confinement with Origin Web Labels (COWL)
- Suborigin Namespaces

# Build a toolbox for trust among users

End-to-End = local self-determination

Modularize

Encrypt everywhere

Build for Open

Enlist and enable the user

# Links

Web Authentication: https://www.w3.org/webauthn

WebAuthn spec: https://www.w3.org/TR/webauthn/

WebAppSec: https://www.w3.org/2011/webappsec/

Web Payments: https://www.w3.org/Payments/WG/

- Securing the Web. W3C TAG Finding, January 2015: https://www.w3.org/2001/tag/doc/web-https
- End-to-End Encryption and the Web. W3C TAG Finding, July 2015: https://www.w3.org/2001/tag/doc/encryption-finding/

# Thanks!

Wendy Seltzer
wseltzer@w3.org https://wendy.seltzer.org/
@wseltzer +1.617.715.4883