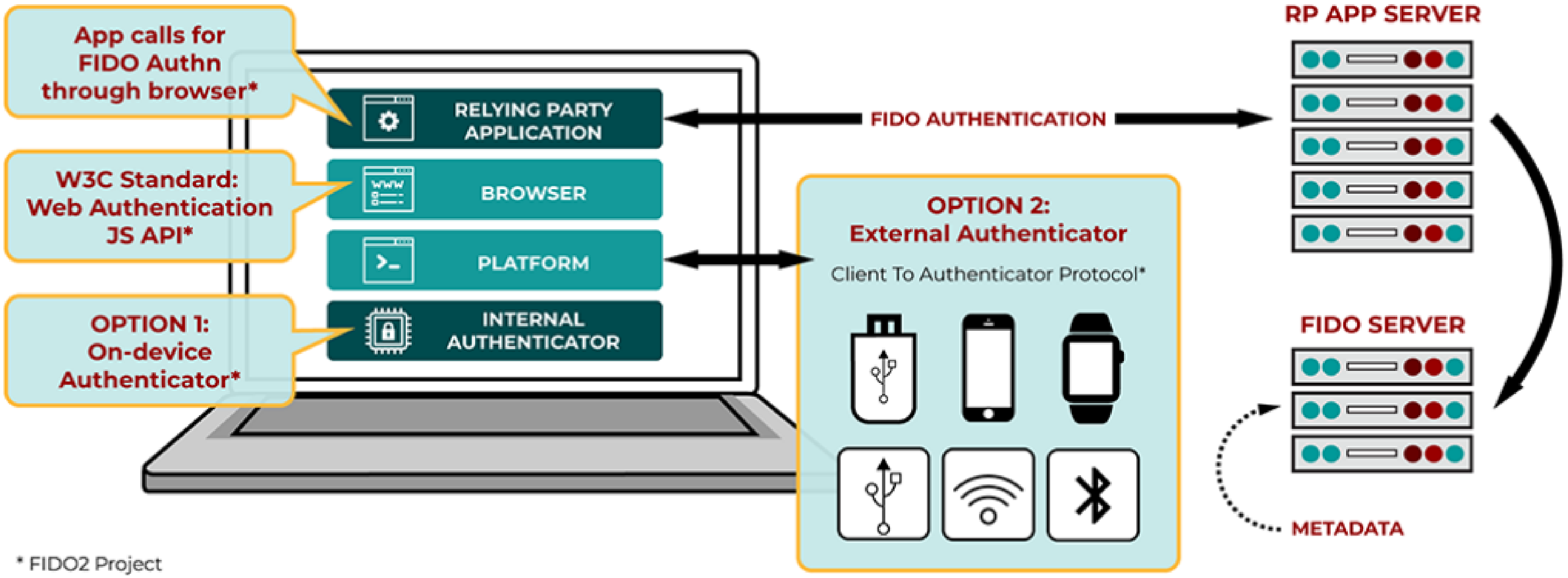
FIDO and FIDO2 framework



FIDO (Fast Identity Online) Alliance

- Open industry association

- Mission
  - Develop open authentication standards and promote their adoption to reduce the use of passwords

- Approach
  - Strong authentication based on public keys
  - Phishing resistance
  - Good usability

FIDO token-based authentication

- Authentication key pairs are stored in tokens
  - Thus we need protocol to interact with them
- Authentication is based on signatures
  - But these are too long to be copied by people
- Enrolment of devices in users' profiles is left to the authenticators
  - Plus the recovery procedure upon loosing a token

FIDO certification

- ▷ Validation of the quality of FIDO products

- ▷ Certification programs
  - Functional
    - Compliance and interoperability
  - Authenticator
    - Protection of secrets (L1 up to L3+)
  - Biometric
    - FAR, FRR
    - IAPMR (Impostor Attack Presentation Match Rate)

Universal 2nd Factor (U2F) protocol

- The user has a U2F device
  - The device creates a unique key pair per service
    - URL based
  - The service registers the public key on the user account
    - Different services get different keys
    - No user tracking
  - The service requests a user’s device signature for their authentication

- Interface with a U2F device
  - JavaScript API (within browsers)
  - Native OS APIs

U2F devices

- USB devices
  - With a distinctive, recognizable HID interface
- NFC devices
- Bluetooth LE devices
- Software applications
  - Possibly backed up by hardware security devices

- Devices must have a "test of user presence"
  - To prevent accessible devices to be used without user consent
  - Devices cannot provide responses without such consent
  - Consent usually involves touching a button (may involve fingerprint or pin code)

U2F protocols

- Upper layer
  - Core cryptographic protocol
  - Defines the semantics and contents of the data items exchanged and produced
  - Defines the cryptographic operations involved in the processing of those data items
- Lower layer
  - Host-device transport protocol
  - CTAP (Client To Authenticator Protocol)

U2F upper layer protocol: User registration

- The U2F device is asked to generate a service-specific key pair
  - Service is identified with a hash of the service identity
    - protocol, hostname, port
- The U2F device generates a key pair
  - And returns a Key Handle and the public key
  - These elements are provided to the service
  - The Key Handle encodes the service identity

U2F upper layer protocol: User authentication (1)

- The user provides their identifier within the service
  - e.g. a user name
  - The service returns the user Key Handle and a random challenge

- The user's client application contacts a locally accessible device to perform a signature, providing
  - The Key Handle
  - A hash of the service identity
  - A hash of client data, which include
    - The random challenge
    - The service hostname
    - And an optional TLS ChannelID extension

U2F upper layer protocol: User authentication (2)

- The device checks if the service identity hash is valid for the Key Handle
  - On success looks up for the corresponding private key
  - And uses it to sign the hashed client data
- The signature is returned to the caller
  - That forwards it to the service for validation
  - Together with the client data
- The service validates the client data
  - And if valid, validates its signature with the user's public key

Certification of U2F devices

- Service providers need to be sure about the quality of U2F devices
  - They need a certification
- U2F have an attestation key pair
  - With a public key certificate issued by the manufacturer
  - And manufacturers need to be FIFO certified
- Public key produced by the device are signed with the attestation private key
  - To prove they were produced by a certified device

Anonymity of attestation key pairs

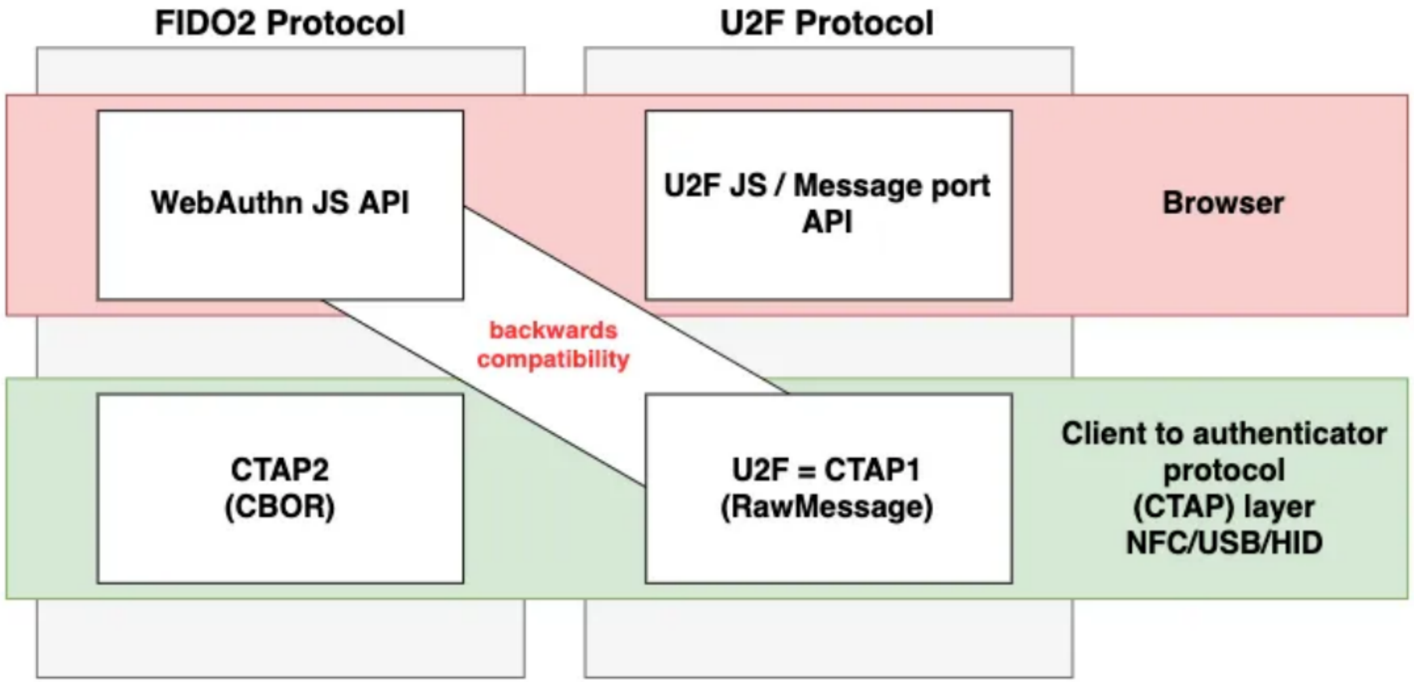
- U2F devices cannot have unique attestation key pairs
  - They would not be anonymous any more
  - Different services could track a user by their attestation public key
- Attestation key pairs are shared by batches of attestation key pairs
  - And thus, users' U2F devices cannot be tracked

Uncertified U2F devices

- They can exist and still being used
  - It all depends on the service
- But in this case, services have to have their own trust chain for those devices

FIDO2 and U2F

▷ FIDO2 is backward compatible with U2F devices



Source: https://medium.com/webauthnworks/sorting-fido-ctap-webauthn-terminology-7d32067c0b01

U2F JS / MessagePort API

- JavaScript interface used by services Web pages to interact with U2F devices
  - Using a MessagePort API
  - https://fidoalliance.org/specs/u2f-specs-master/fido-u2f-javascript-api.html

WebAuthn

- Part of the FIDO2 framework
  - Web Authentication API
    - An evolution of the U2F API
  - Specification written by the W3C and FIDO
    - With the participation of Google, Mozilla, Microsoft, Yubico, and others
- Web API
  - Service API for dealing with the registration and authentication of U2F devices
- JavaScript API
  - Used by Web pages to interact with local U2F devices
  - Implemented by browsers

Client to Authenticator Protocol (CTAP)

- Standard interoperation between a user platform (e.g. a laptop) and a user-controlled cryptographic authenticator
  - ITU-T Recommendation X.1278

- Based in the Universal 2nd Factor (U2F) authentication standard

CTAP variants

- CTAP1/U2F
  - Aka FIDO U2F
  - Raw message format
- CTAP2
  - For FIDO2 authenticators (aka WebAuthn authenticators)
  - CBOR (Concise Binary Object Representation) data serialization format
    - Loosely based on JSON but in a binary format

Use case: Passkeys

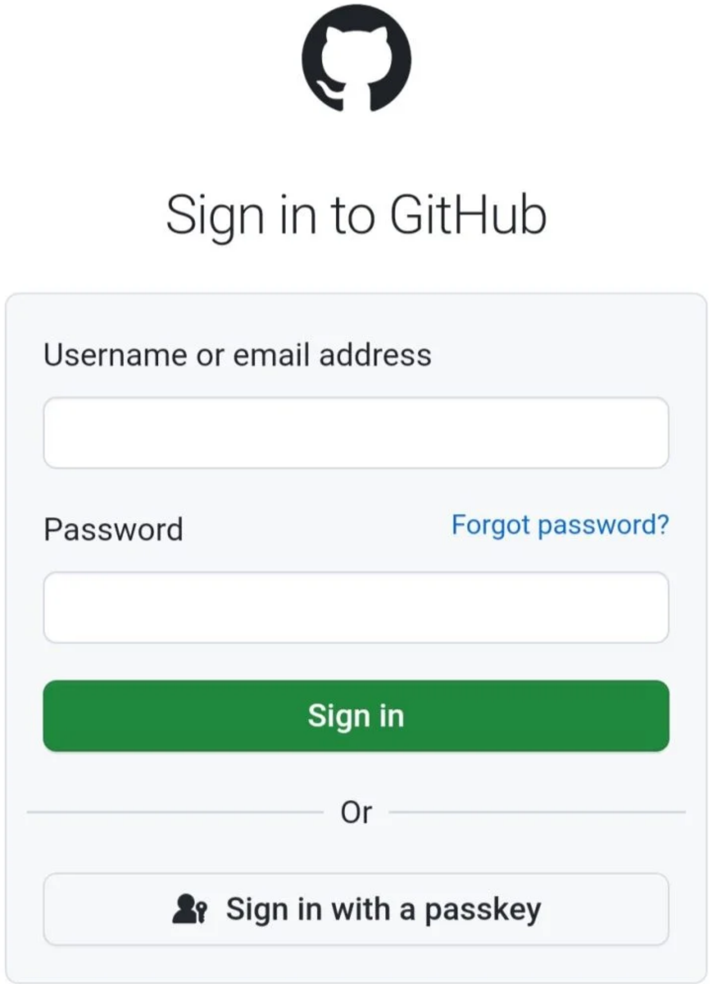
- Passkeys appeared as a way to avoid common auth issues
  - Weak passwords
  - Phishing
  - Password/cookie theft
  - Lack of a second factor
  - MITM or Leak
  - Cost with 2nd factor

- They promote better usability
  - No need to generate/memorize/manage hundreds of passwords

Use case: Passkeys

- How:
  - Using auth material from the user directly in the device
    - This will never be exposed to others
    - Face, Fingerprint, PIN code (PIN can be alphanumeric)
    - Auth material enables the process but it is not sent
  - Generating a keypair, whose public key is stored at the servisse
    - Compromise of the service will only allow access to the **public** key
  - Authentication considers the service, device, keys and user
    - Implicit use of 2FA and external HSM may be used
- Why: No secret is exposed to third parties
  - Also: domain is matched by browser, blocking phishing and typos

Use case: Passkeys



Use case: Passkeys
Functionality

- Device Bound Passkeys: device specific keys that may never leave it
  - Such as typical FIDO 2 keys

- Attestation: capability to ensure the provenance of the authenticator
  - Ensures that the authenticator is actually providing the auth data
    - Public key is packed into na attestation object, signed by a private key
    - Very flexible, as long as relying party can verify the attestation

- Synced Passkeys: capability to keep passkeys available
  - Passkeys are backed up and used when required

Use case: Passkeys
Limitations

- Device support: It's still a new technology

- Device dependency: Passkeys are rapidly device specific
  - Cross Device Authentication allows linking devices but authenticators must support it
  - Different ecossystems may still not be fully interoperable

- Biometrics are not that safe against local attacks
  - But most attacks are not local…
  - At it's better than only passwords

Use case: Passkeys

Capability	Android	Chrome OS	iOS/iPad OS	macOS	Ubuntu	Windows
Synced Passkeys	✓ v9+	Planned [1]	✓ v16+	✓ v13+ [2]	✗ Not Supported	Planned [1]
Browser Autofill UI	✓ Chrome 108+ Edge 122+ ✗ Firefox	Planned	✓ Safari Chrome Edge Firefox	✓ Safari Chrome 108+ Firefox 122+ Edge 122+	✗ Not Supported	✓ Chrome 108+ [3] Firefox 122+ [3] Edge 122+ [3]
Cross-Device Authentication ***Authenticator***	✓ v9+	n/a	✓ v16+	n/a	n/a	n/a
Cross-Device Authentication ***Client***	Planned	✓ v108+	✓ v16+	✓ v13+	✓ Chrome Edge	✓ v23H2+
Third-Party Passkey Providers	✓ v14+	Browser Extensions	✓ v17+	✓ v14+	Browser Extensions	Browser Extensions Native Planned

https://passkeys.dev/device-support/ **as in April 2024**

Use case: Passkeys

Capability	Android	Chrome OS	iOS/iPad OS	macOS	Ubuntu	Windows
***Device-bound* Passkeys**	✗ Not Supported	✗ Not Supported	on security keys	on security keys	on security keys	✓
Client Hints	✗ Not Supported	Chrome [4]	✗ Not Supported	Chrome [4] Edge [4] ✗ Safari Firefox	Chrome [4] Edge [4] ✗ Firefox	Chrome [4] [5] Edge [4] [5] ✗ Firefox
Device-bound Passkey Attestation	n/a	n/a	n/a	n/a	n/a	✓
Synced Passkey Attestation	✗ Not Supported	n/a	✗ Not Supported	✗ Not Supported	n/a	n/a

https://passkeys.dev/device-support/ as in April 2024