FIDO2, WebAuthn and passwordless

Tobiasz Heller

Agenda

Why passwords only is not enough

What is

- FIDO2,
- WebAuthn,
- CTAP
- Passwordless

Live coding

81% of all hacking-related breaches leverage stolen or weak passwords.

Verizon 2017 Data Breach Investigations report

Users have more than 90 online accounts

https://fidoalliance.org/what-is-fido/

Up to 51% of passwords are reused

https://fidoalliance.org/what-is-fido/

28% of users use 2nd factor authentication

Duo Labs 2017, State of the Auth

FIDO2

FIDO2

In simple words

- **Specification**
- allows developers leverage either
  - **hardware keys** (e.g., YubiKeys) - roaming authenticators
  - or **secure hardware on** the **device** (e.g., secure elements on your phone, TPMs on your laptop) - platform authenticators
- gated by biometric sensors or pin (alphanumeric)
- to **authenticate** users **without passwords**
- by using the Javascript API in browser

Demo

FIDO2

FIDO alliance https://fidoalliance.org/

- Design authentication standards to help reduce the world's over-reliance on passwords
- Many members: Google, Microsoft, 1Password, Amazon, Apple, Paypal, Lenovo, Intel, Yubico etc
- They designed:
    - Universal Authentication Framework - **UAF**
    - Universal 2nd Factor - **U2F** (now renamed to CTAP1)
    - **FIDO2** successor of UAF and U2F

FIDO2

Goals

- Authentication standards based on **public key cryptography**
- **More secure** than passwords SMS and OTP
- **Simpler** for consumers to use
- **Easier** for service providers to deploy and manage

Public key cryptography

Asymmetric cryptography - Short recap

- Uses the concept of a keypair. Each key pair consists of a **public key** and a corresponding **private key**
- These "keys" are **long, random numbers** that have a mathematical relationship with each other.
- In **encryption anyone** with **public key** can **encrypt** message but only one with **private key** can **decrypt** ciphertext to obtain original message
- In **digital signature** sender **signs** message with **private key**. Anyone with **public key** can **verify** message signature, but forger who does not know private key, cannot pass message verification step

FIDO2

Security

- Cryptographic login credentials are **unique** across **every website**
- Private key **never leave** the user's **device** and are never stored on a server
- **Unphishable** - there are no codes/passwords that user need to enter on website
- Protection to replay attacks - **new challenge** for **every** authentication ceremony
- Server only stores public key and randomly generated credential_id, it means that **servers no longer store secrets**

FIDO2

Use cases

PASSWORDLESS
authentication

SECOND-FACTOR
authentication

MULTI-FACTOR
authentication

SECURITY KEY

FACIAL RECOGNITION

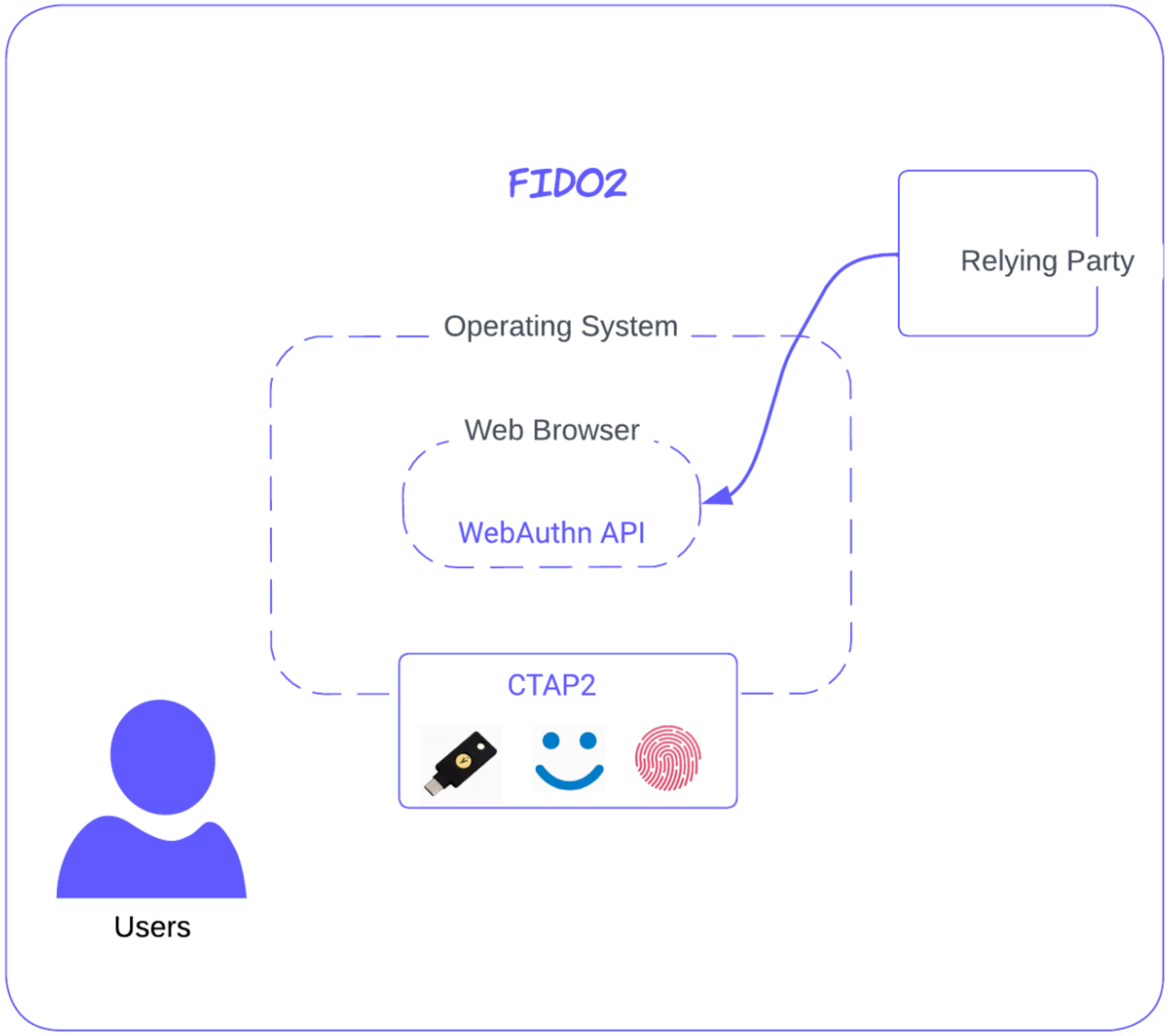
FINGERPRINT

VOICE

https://fidoalliance.org/what-is-fido/

FIDO2

CTAP + WebAuthn



https://goteleport.com/blog/webauthn-explained/

Relying Party

- Software application that wants to authenticate a user
- Can be websites, web applications, desktop applications etc

CTAP

Client-to-Authenticator Protocol

- **protocol** that is used for communication between a **client** or platform, and an **external authenticator**
- CTAP2 - allows passwordless login
- CTAP1 (U2F) - can be used as 2nd factor only

WebAuthn

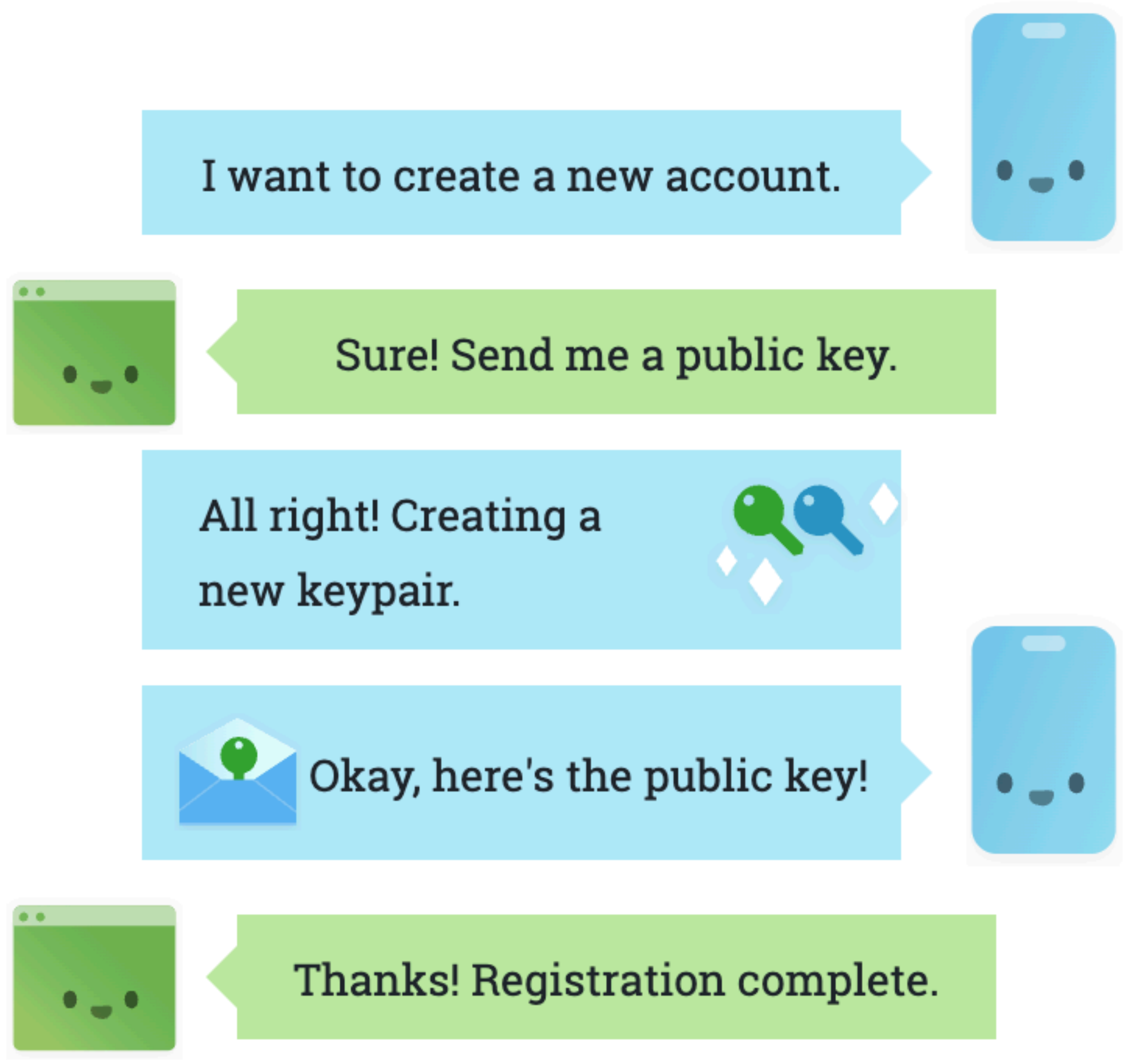
WebAuthn

The Web Authentication API (WebAuthn) is **specification** that allows servers to **register** and **authenticate** users using **public key cryptography** instead of a password.

- users can easily setup different authenticators: security keys and built-in platform biometric sensors
- https://www.w3.org/TR/webauthn-2/
- Written by W3C and FIDO Alliance, with participation of Google, Mozilla, Microsoft, Yubico and more
- Supported by all leading browsers and platforms

WebAuthn

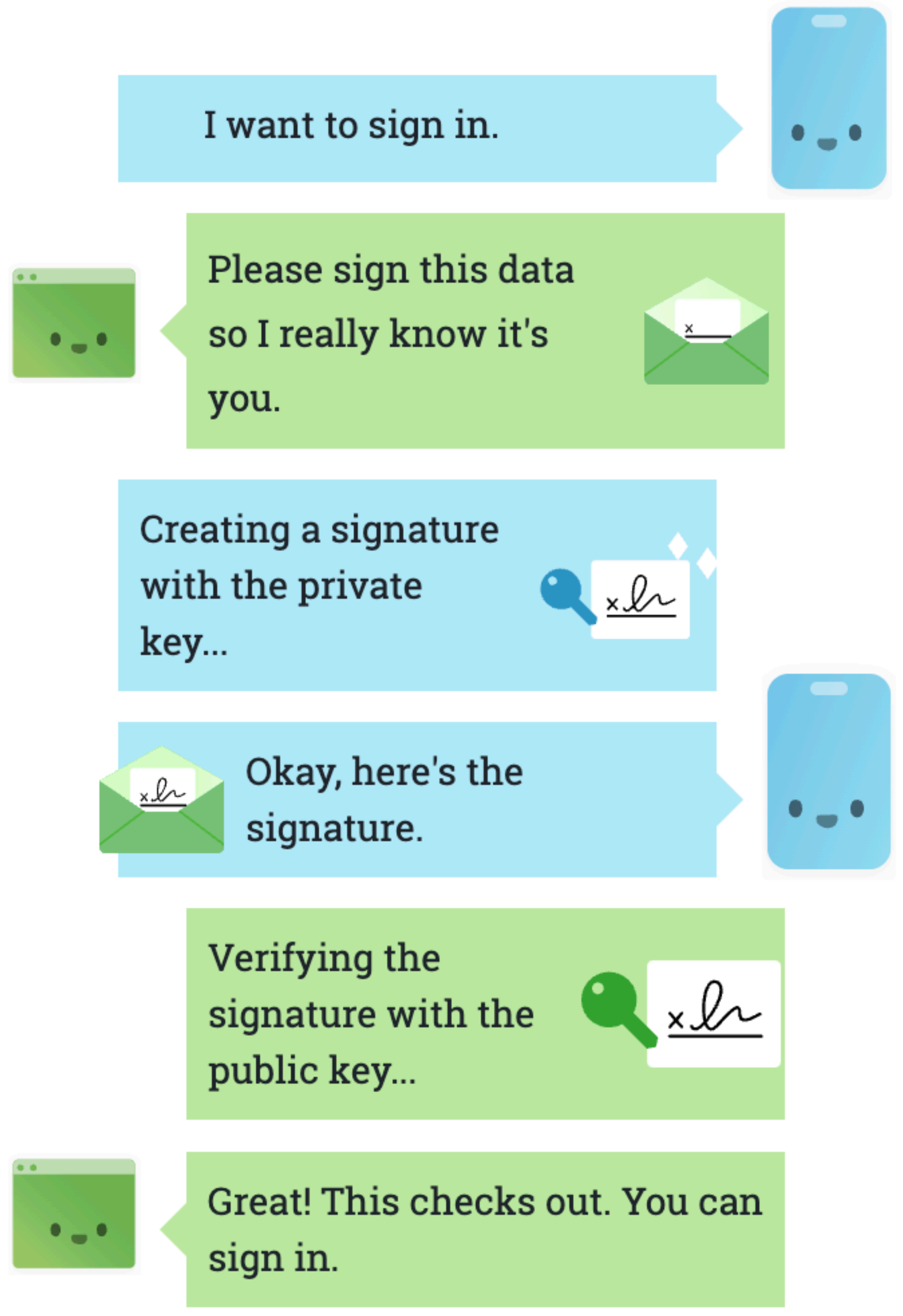
Registration


I want to create a new account.
Sure! Send me a public key.
All right! Creating a new keypair.
Okay, here's the public key!
Thanks! Registration complete.

https://webauthn.guide/

WebAuthn

Authentication



I want to sign in.

Please sign this data so I really know it's you.

Creating a signature with the private key...

Okay, here's the signature.

Verifying the signature with the public key...

Great! This checks out. You can sign in.

https://webauthn.guide/

WebAuthn

Browser API

- navigator.credentials.create() - registration
- navigator.credentials.get() - authentication
- With publicKey option
- Interactive guide with example payloads: https://webauthn.guide/
- https://www.w3.org/TR/webauthn-2/#sctn-sample-scenarios

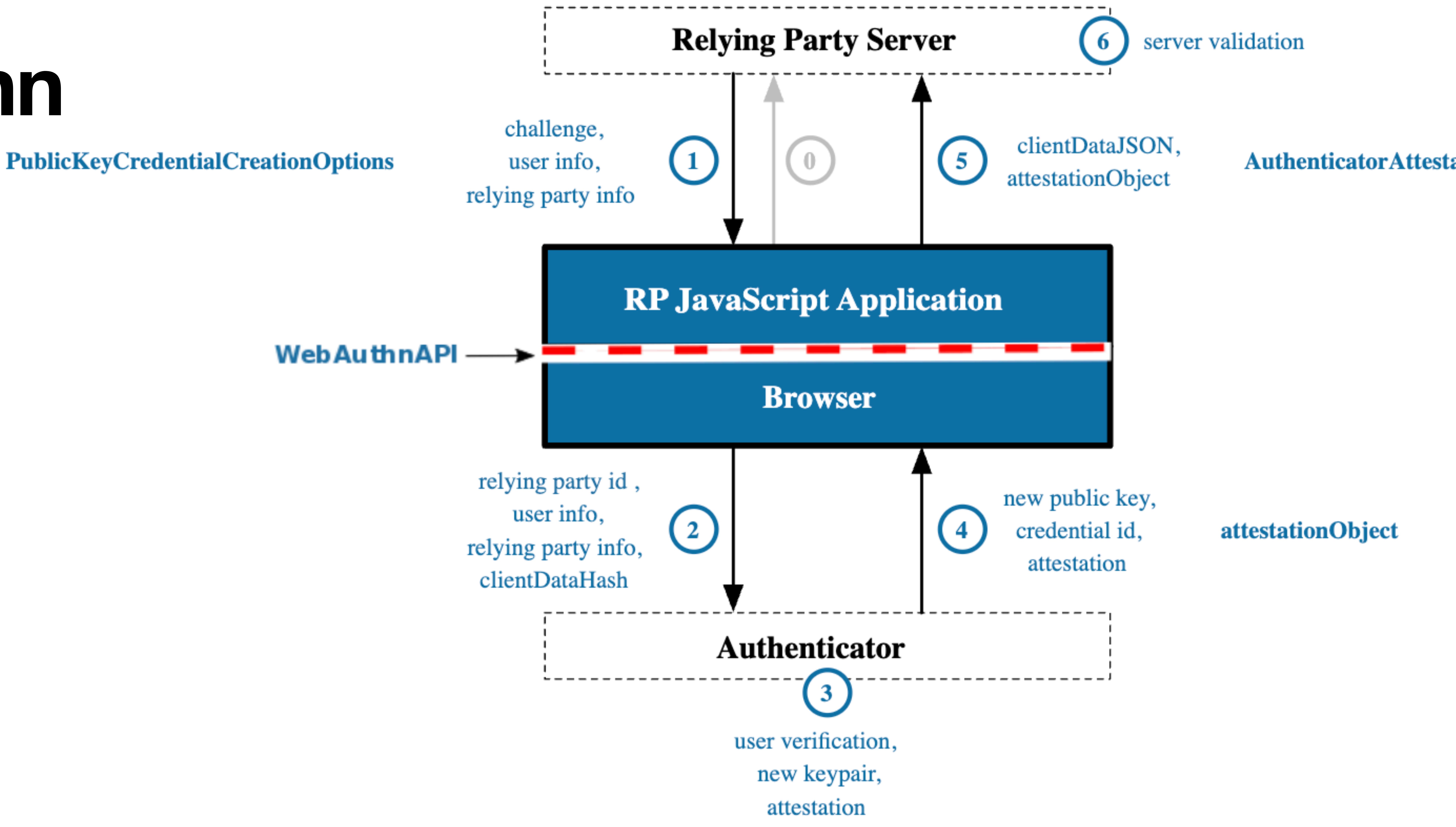
WebAuthn

Go support

- github.com/go-webauthn/webauthn
  - (hard fork of archived github.com/duo-labs/webauthn)
- Still before v1
- Breaking changes happens

WebAuthn

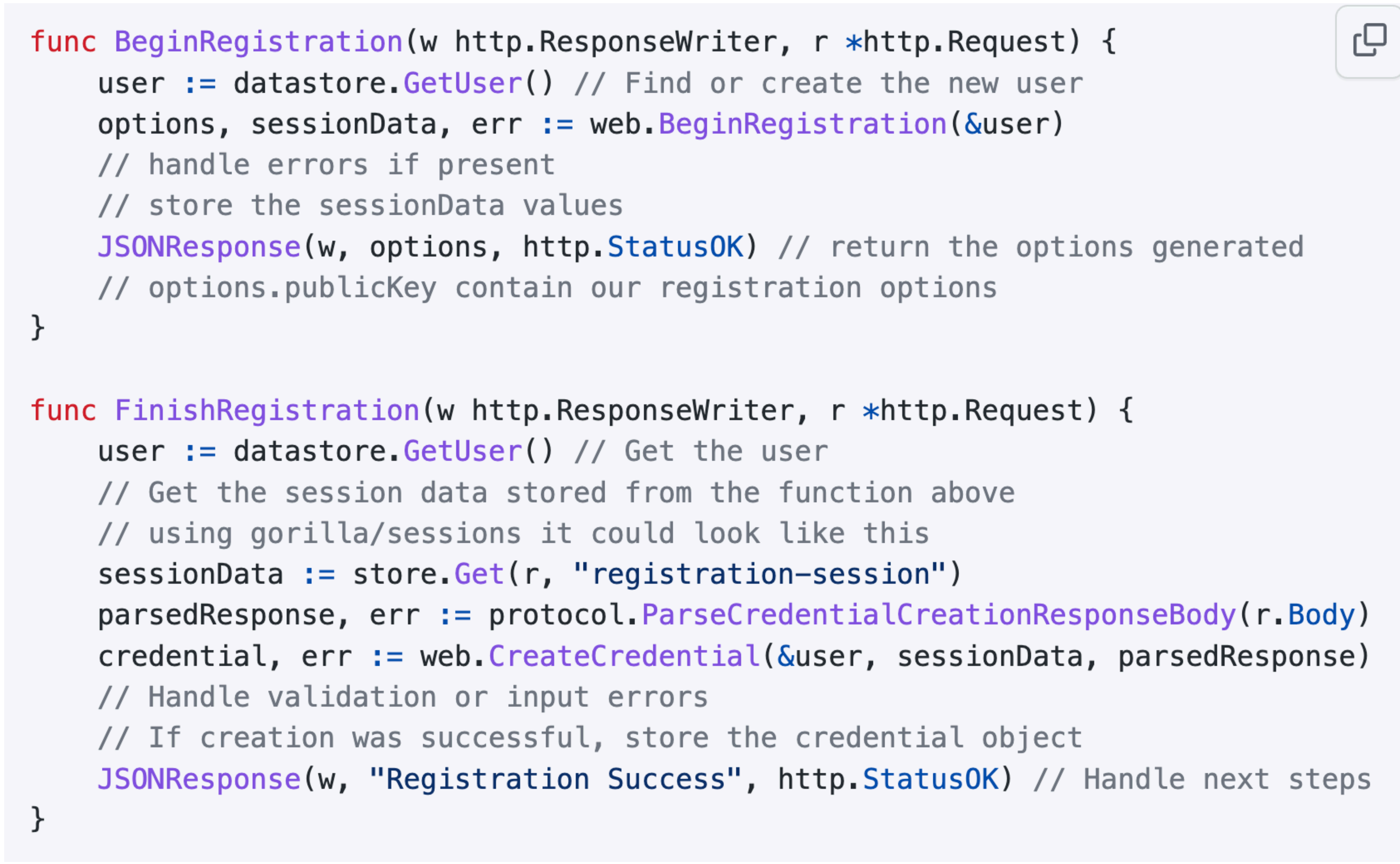
Registration details



https://www.w3.org/TR/webauthn-2/#sctn-api

WebAuthn

Registration details



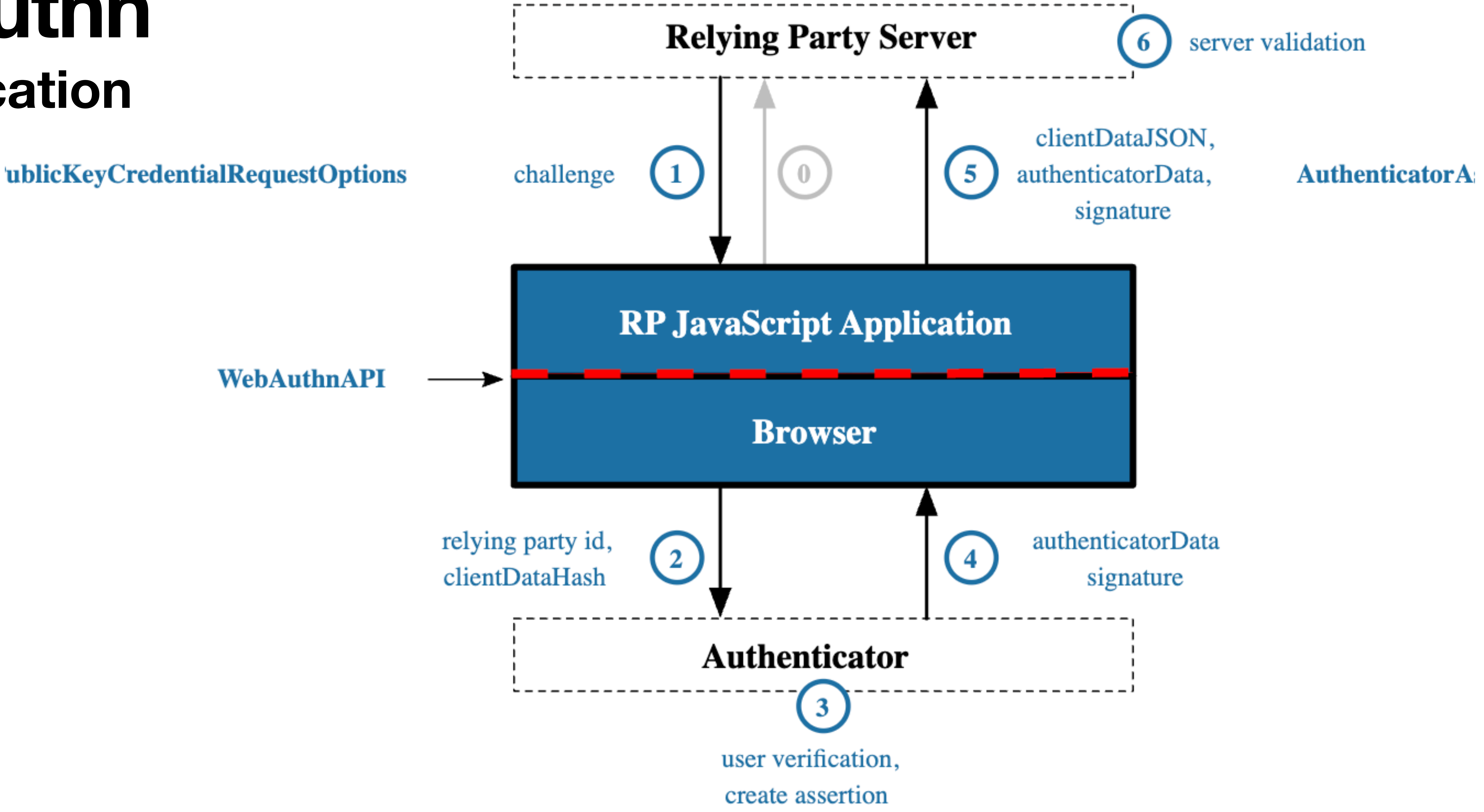
```go
func BeginRegistration(w http.ResponseWriter, r *http.Request) {
    user := datastore.GetUser() // Find or create the new user
    options, sessionData, err := web.BeginRegistration(&user)
    // handle errors if present
    // store the sessionData values
    JSONResponse(w, options, http.StatusOK) // return the options generated
    // options.publicKey contain our registration options
}

func FinishRegistration(w http.ResponseWriter, r *http.Request) {
    user := datastore.GetUser() // Get the user
    // Get the session data stored from the function above
    // using gorilla/sessions it could look like this
    sessionData := store.Get(r, "registration-session")
    parsedResponse, err := protocol.ParseCredentialCreationResponseBody(r.Body)
    credential, err := web.CreateCredential(&user, sessionData, parsedResponse)
    // Handle validation or input errors
    // If creation was successful, store the credential object
    JSONResponse(w, "Registration Success", http.StatusOK) // Handle next steps
}
```

https://github.com/go-webauthn/webauthn

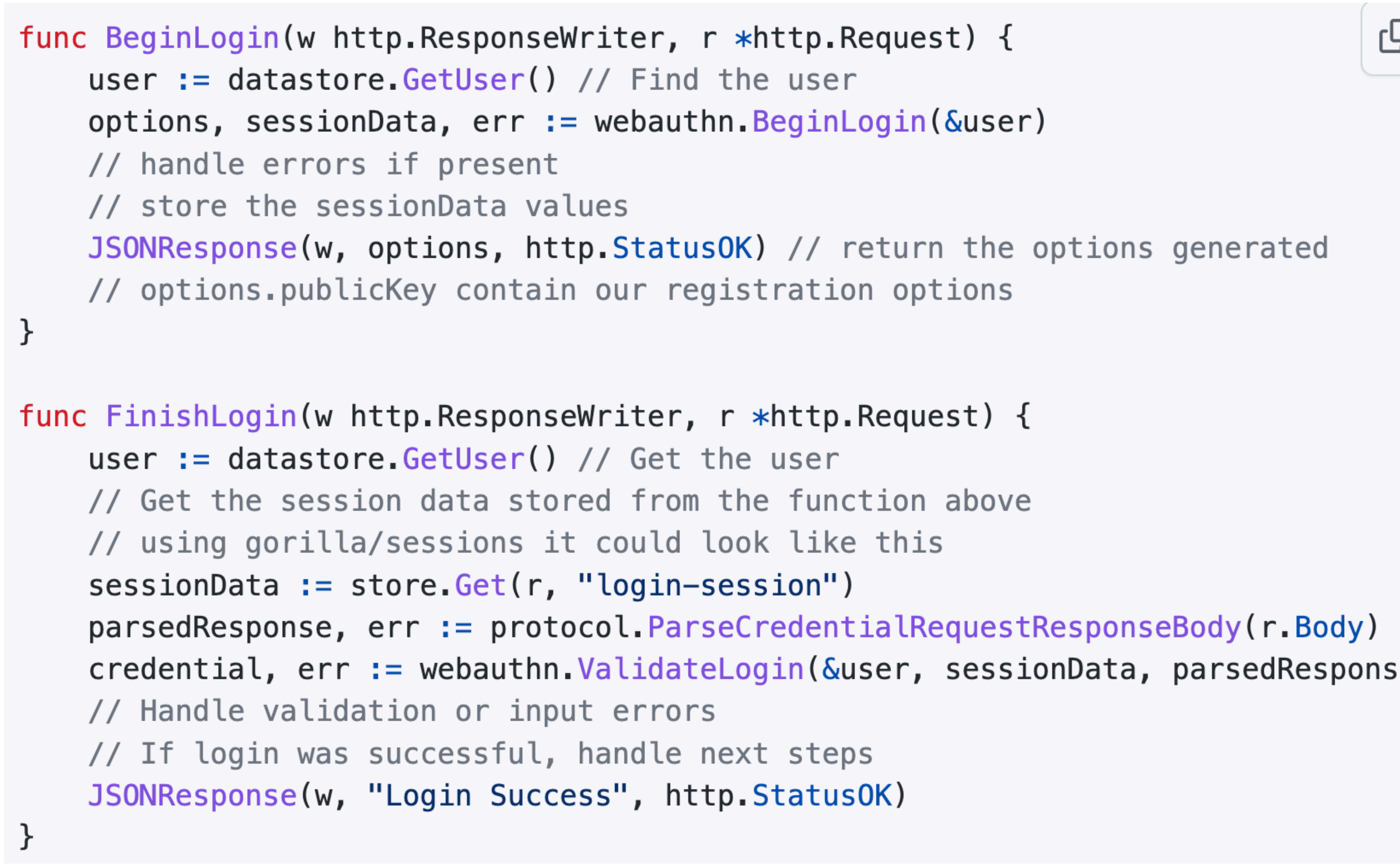
WebAuthn

Authentication details


Relying Party Server
6
server validation
ublicKeyCredentialRequestOptions
challenge
1
0
5
clientDataJSON, authenticatorData, signature
AuthenticatorA
RP JavaScript Application
WebAuthnAPI
Browser
relying party id, clientDataHash
2
4
authenticatorData signature
Authenticator
3
user verification, create assertion

WebAuthn

Authentication details



```
func BeginLogin(w http.ResponseWriter, r *http.Request) {
    user := datastore.GetUser() // Find the user
    options, sessionData, err := webauthn.BeginLogin(&user)
    // handle errors if present
    // store the sessionData values
    JSONResponse(w, options, http.StatusOK) // return the options generated
    // options.publicKey contain our registration options
}

func FinishLogin(w http.ResponseWriter, r *http.Request) {
    user := datastore.GetUser() // Get the user
    // Get the session data stored from the function above
    // using gorilla/sessions it could look like this
    sessionData := store.Get(r, "login-session")
    parsedResponse, err := protocol.ParseCredentialRequestResponseBody(r.Body)
    credential, err := webauthn.ValidateLogin(&user, sessionData, parsedRespons
    // Handle validation or input errors
    // If login was successful, handle next steps
    JSONResponse(w, "Login Success", http.StatusOK)
}
```

https://github.com/go-webauthn/webauthn

WebAuthn

Server side registration & authentication validation

- Multi step process
- https://www.w3.org/TR/webauthn-2/#sctn-rp-operations
- Provided by go-webauthn library

Demo

Passwordless

Aka Discoverable credentials

- U2F authentication ceremony
  - a proof of identity (aka "something you know", the password)
  - a proof of presence (aka "something you have", the tap in the authenticator)
- WebAuthn with discoverable credentials
  - user verification (authenticator promise that identity was verified, either via fingerprint sensor or PIN - stored on authenticator side)
- Not every authenticator support passwordless/usernameless login
- The one which support passwordless, store username, relaying party ID etc at authenticator

Reading materials

- Webauthn spec https://www.w3.org/TR/webauthn-2/
- https://webauthn.guide/ & https://webauthn.io/ by Duo
- https://fidoalliance.org/how-fido-works/
- https://goteleport.com/blog/webauthn-explained/
- https://goteleport.com/blog/how-passwordless-works/
- https://github.com/herrjemand/awesome-webauthn

Support for webauthn in Go CLI

- https://github.com/Yubico/libfido2 - c library for FIDO devices over USB/NFC
- Go wrapper around libfido2 - https://github.com/keys-pub/go-libfido2
- FIDO2 usb support teleport/lib/auth/webauthncli
- Touch ID support teleport/lib/auth/touchid
- Windows Hello support teleport/lib/authn/webauthnwin

Other topics

- Attestation
- Passkeys
  - You need at least 2 authenticators for recovery
  - Private key are stored in cloud, for example on apple iCloud for apple ecosystem

Thank you