

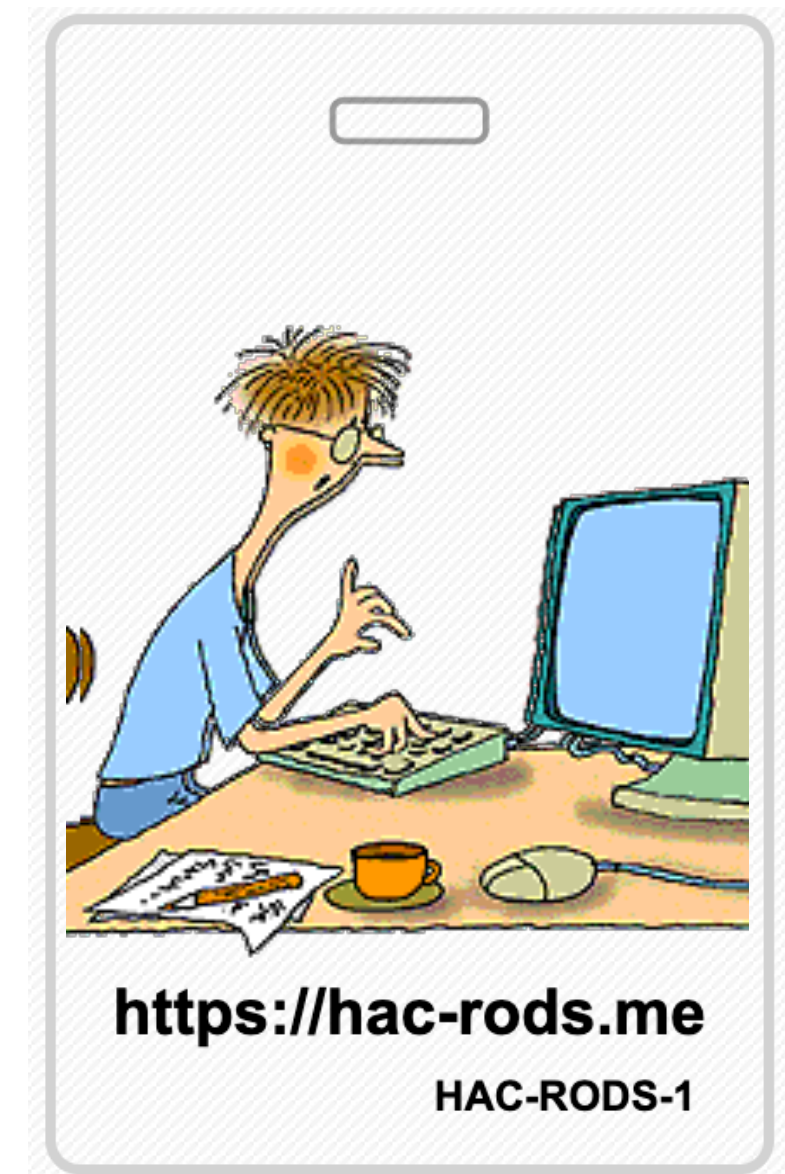
The World of Passkeys



Hi!

Hi, I am Helio Cola!

- ~23 years developing SW
- ~13 years since I started working with RoR
- ==> <https://hac-rods.me/>
- ==> <https://ruby.social/@hacrods>



Agenda

- My other talks
- What is passkey
- 2fa or not 2fa
- How it works && under the hood
- Does anybody want to see a live demo?
- Passkey in the Ruby Community



Before I start

Raise your hand...



- If you set Passkeys on your GitHub account
- Have you setup passkeys as 2fa method in GitLab?



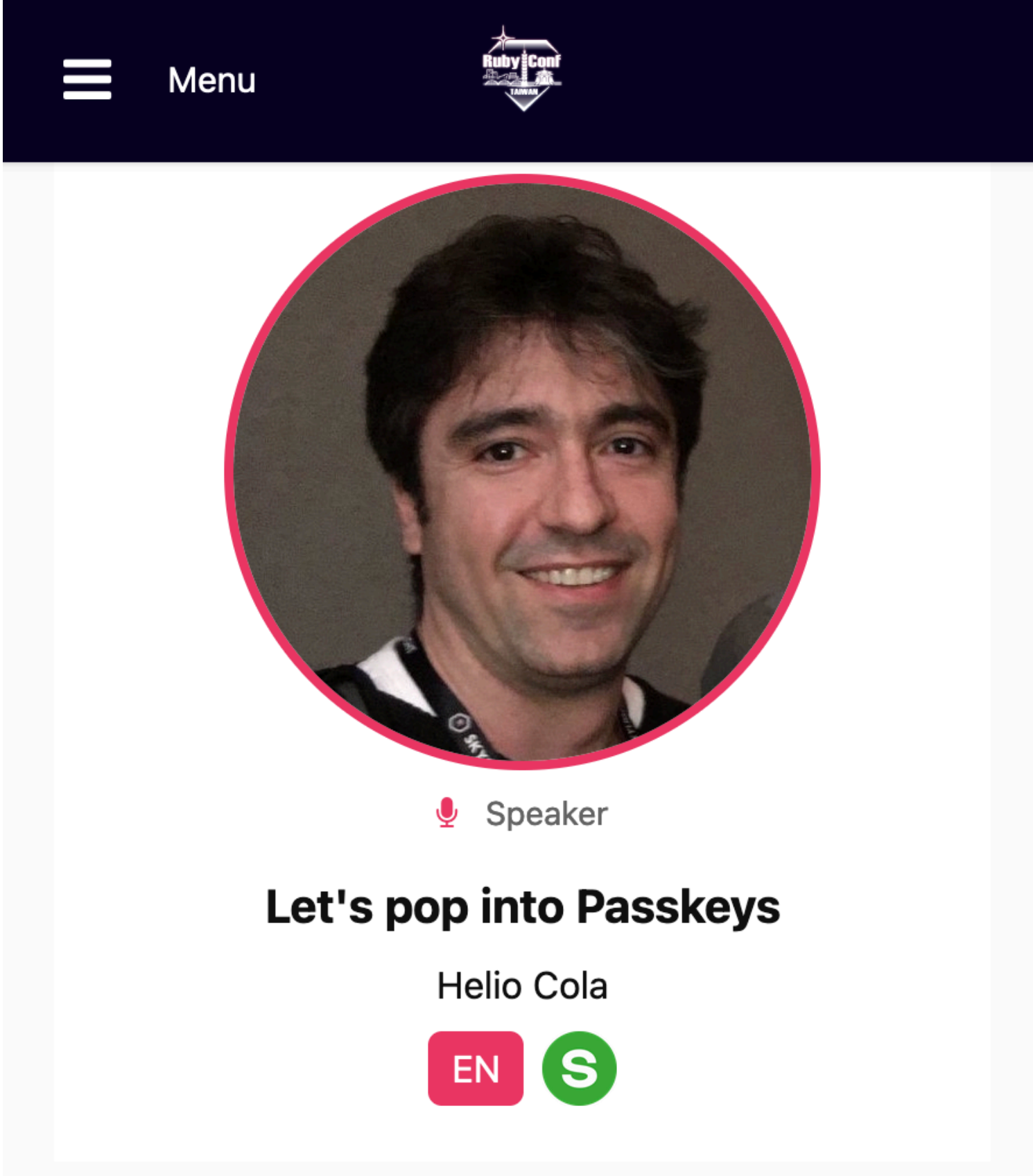
My other talks

RUBYCONFTH {



14:35	The world of Passkeys
-	 Ruby Slides Video
15:05	 Helio Cola

- Ruby Conf Thailand: <https://rubyconfth.com>
- Ruby Conf Taiwan: <https://2023.rubyconf.tw>



The screenshot shows a presentation slide with a dark blue header. On the left, there is a 'Menu' button with a hamburger icon. On the right, there is the 'Ruby Conf TAIWAN' logo. The main content area features a large circular profile picture of Helio Cola with a red border. Below the picture, it says 'Speaker' with a microphone icon. The title of the talk is 'Let's pop into Passkeys', followed by the speaker's name 'Helio Cola'. At the bottom, there are two buttons: a red one with 'EN' and a green one with 'S'.



What is Passkey

- Is a replacement for passwords
- It is part of a web authentication standard
- It is a public/private key pair used for challenge based authentication
- It is uses public key cryptography (invented in the 1970s)
- Sometimes it is protected by your device biometrics
- Sometimes it is discoverable
- Sometimes is bound to your device



What are Passkeys

A password is something that can be remembered and typed, and a passkey is a secret stored on one's devices, unlocked with biometrics.

Source: <https://passkeys.dev/docs/intro/what-are-passkeys/>



***Passkey is a public and private key pair,
protected by your device biometrics,
used for a challenge based authentication***



What is Passkey

“Passkey is a public and private key pair”

- A private and public key, used to encrypt and decrypt data
- A core concept of public key encryption

“protected by your device biometrics”

- To use it, your device will first execute a biometrics verification

“used for a challenge based authentication”

- User is asked to sign with private key
- Web app/site checks with users' public key



Who

- First version of Web Authentication API was published in May 2016
- Created by folks from: Nok Nok Labs, Microsoft, PayPal, and Google

Web Authentication: A Web API for accessing scoped credentials

W3C First Public Working Draft, 31 May 2016



This version:

<http://www.w3.org/TR/2016/WD-webauthn-20160531/>

Latest published version:

<http://www.w3.org/TR/webauthn/>

Editor's Draft:

<http://w3c.github.io/webauthn/>

Editors:

[Vijay Bharadwaj](#) (Microsoft)

[Hubert Le Van Gong](#) (PayPal)

[Dirk Balfanz](#) (Google)

[Alexei Czeskis](#) (Google)

[Arnar Birgisson](#) (Google)

[Jeff Hodges](#) (PayPal)

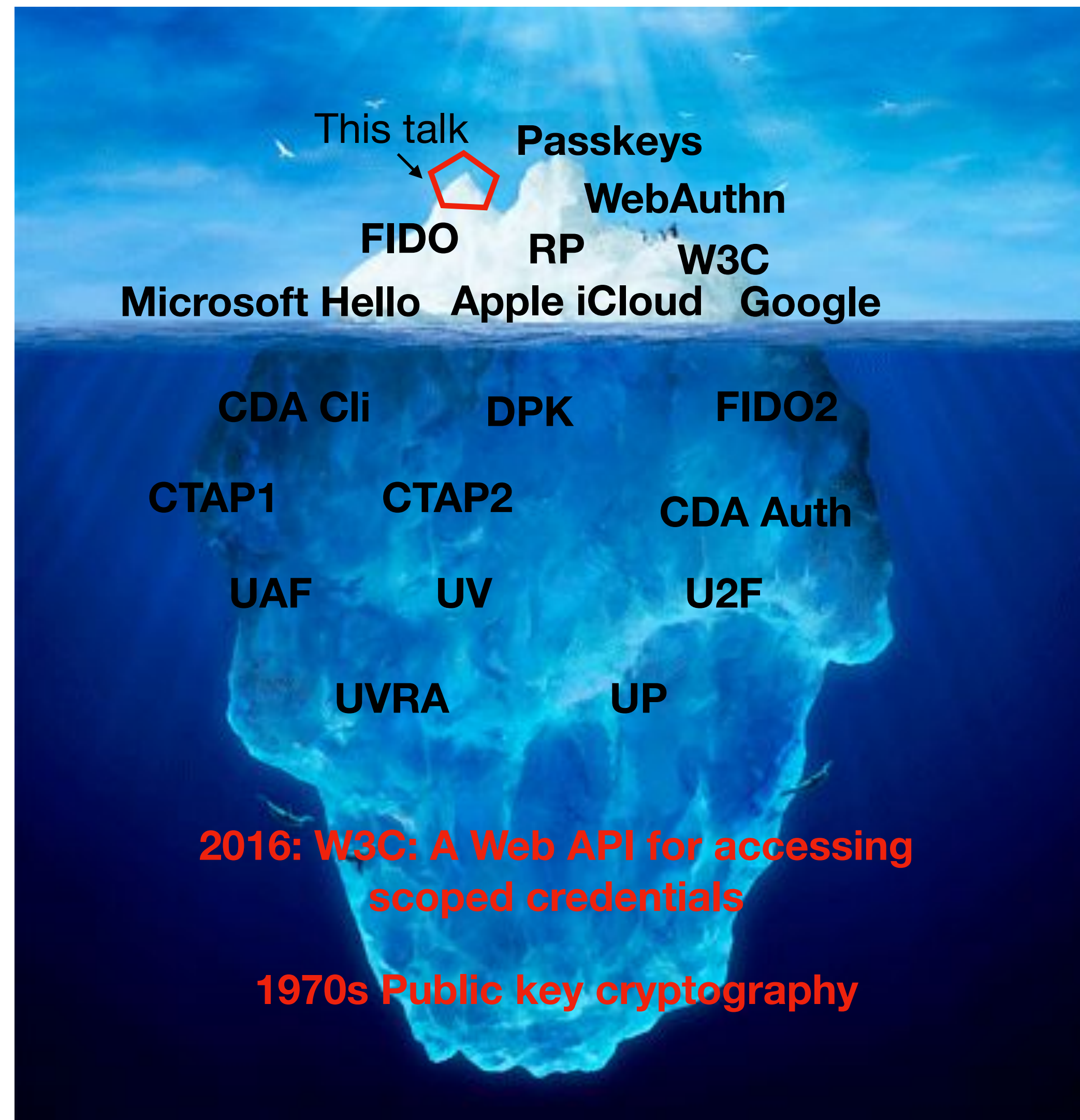
[Michael B. Jones](#) (Microsoft)

[Rolf Lindemann](#) (Nok Nok Labs)

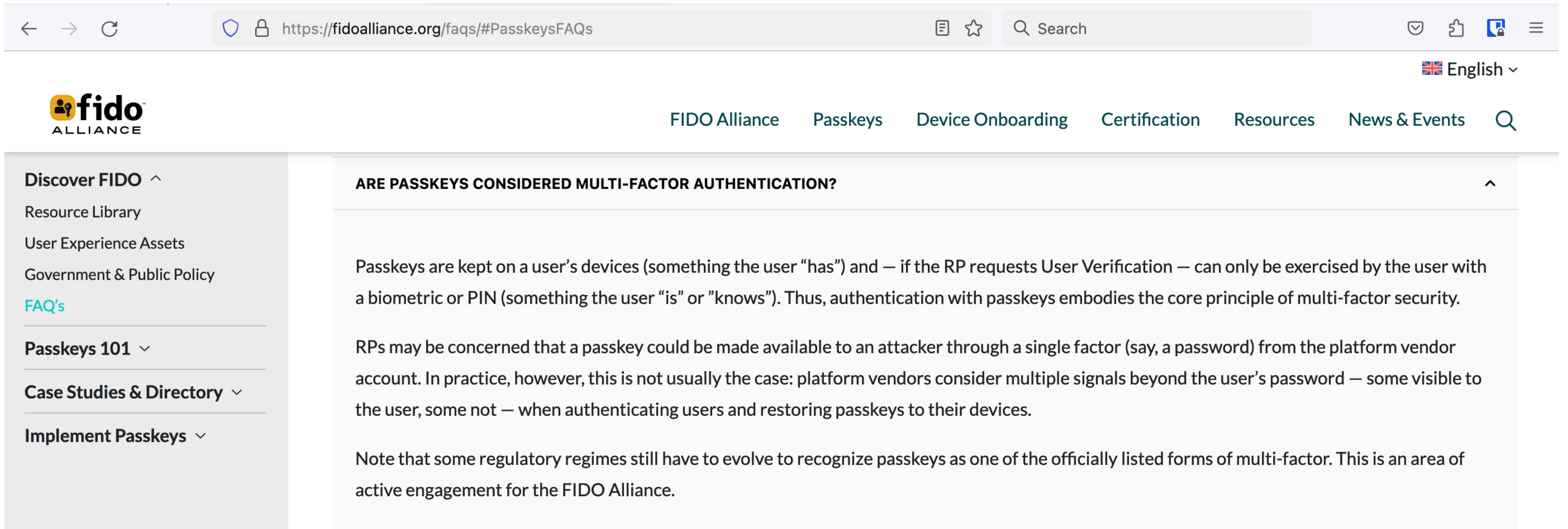
Source: <https://www.w3.org/TR/2016/WD-webauthn-20160531/>



The Passkeys Iceberg



2FA or not 2FA



The screenshot shows a web browser displaying the FIDO Alliance website. The address bar shows the URL <https://fidoalliance.org/faqs/#PasskeysFAQs>. The page header includes the FIDO Alliance logo and navigation links for FIDO Alliance, Passkeys, Device Onboarding, Certification, Resources, and News & Events. A language selector is set to English. The main content area features a sidebar with navigation options: Discover FIDO (with sub-links for Resource Library, User Experience Assets, and Government & Public Policy), Passkeys 101, Case Studies & Directory, and Implement Passkeys. The main article is titled "ARE PASSEYS CONSIDERED MULTI-FACTOR AUTHENTICATION?" and contains the following text:

Passkeys are kept on a user's devices (something the user "has") and — if the RP requests User Verification — can only be exercised by the user with a biometric or PIN (something the user "is" or "knows"). Thus, authentication with passkeys embodies the core principle of multi-factor security.

RPs may be concerned that a passkey could be made available to an attacker through a single factor (say, a password) from the platform vendor account. In practice, however, this is not usually the case: platform vendors consider multiple signals beyond the user's password — some visible to the user, some not — when authenticating users and restoring passkeys to their devices.

Note that some regulatory regimes still have to evolve to recognize passkeys as one of the officially listed forms of multi-factor. This is an area of active engagement for the FIDO Alliance.

Source: <https://fidoalliance.org/faqs/#PasskeysFAQs>



2FA meaning

2 out of the 3 below:

- **Something the user has:** any physical object in the possession of the user, such as a security token (USB stick), a bank card, a key, etc.
- **Something the user knows:** certain knowledge only known to the user, such as a password, PIN, PUK, etc.
- **Something the user is:** some physical characteristic of the user (biometrics), such as a fingerprint, eye iris, voice, typing speed, pattern in key press intervals, etc.



So... 2FA? Or not 2FA?

- Passkey is kept on the user device (phone, usb stick), sometimes replicated to your cloud/device account (Apple, Google, Microsoft)
something the user has
- Passkey can only be used after biometric (or pin) verification
something the user is (or knows)



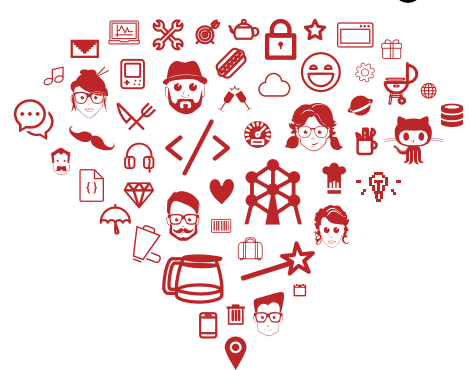
So... 2FA? 1FA? notFA? yesFA?

- FIDO: *“Note that some regulatory regimes still have to evolve to recognize passkeys as one of the officially listed forms of multi-factor. This is an area of active engagement for the FIDO Alliance.”*
- *Something the user has, is, or knows:*
 - *I **have** a phone/usb stick, and I need it*
 - *I **am** my me, my face, my finger, and I need it*
 - *I **know** my usb stick PIN or my usb stick validate my digital fingerprint*



What about Password Managers

- Should it become Passkeys Managers?
- Can Password Managers access your device biometrics?
 - *Should they?*
- Are Password Managers necessary in this new world where Passkeys exist?
 - *BTW: a few weeks ago, I stopped being able to use a Passkey in one of my webapps, on Safari, while I am logged in on my Password Manager's vault...*
 - *Yesterday: back working again... but buggy... but looks good!*



How it works && under the hood



How it works

- Registration

User sign up for a new service: email, username etc...

- Authentication

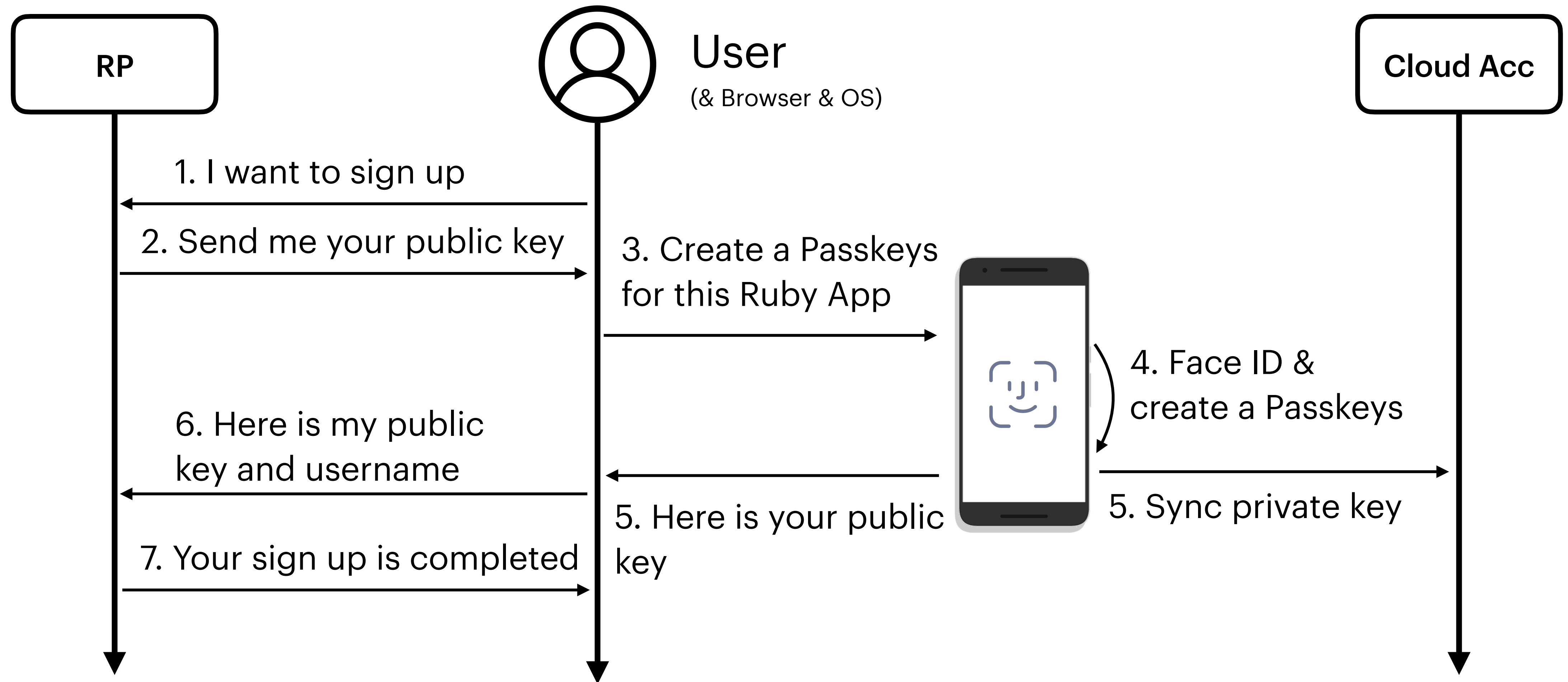
With my email/username and my passkeys

- Re-authentication

In case of sensitive transactions



Registration



Let's look inside

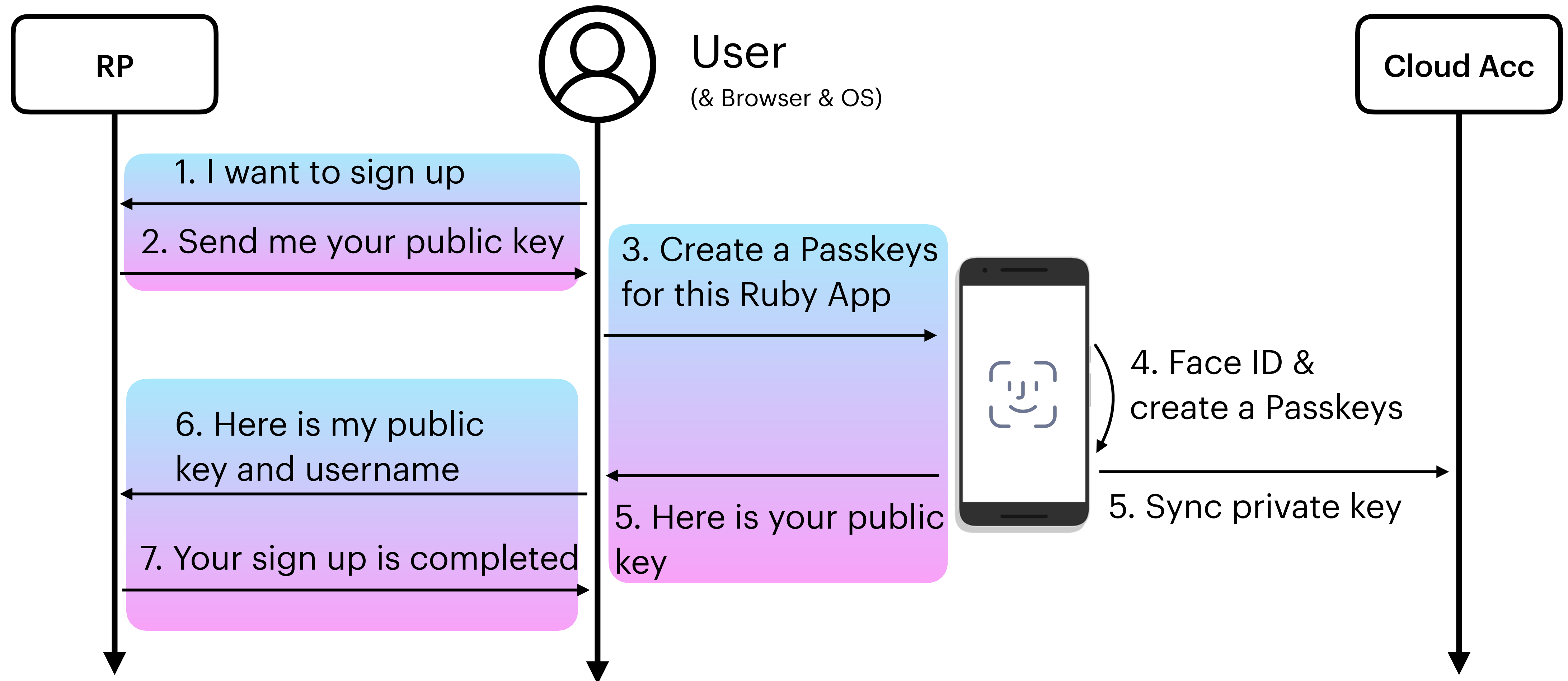


Under the hood

- Reference app: `cedarcode/webauthn-rails-demo-app`
 - Link: <https://github.com/cedarcode/webauthn-rails-demo-app>
- Registration flow steps:
 - Initiation phase
 - What happens in the browser
 - Verification phase

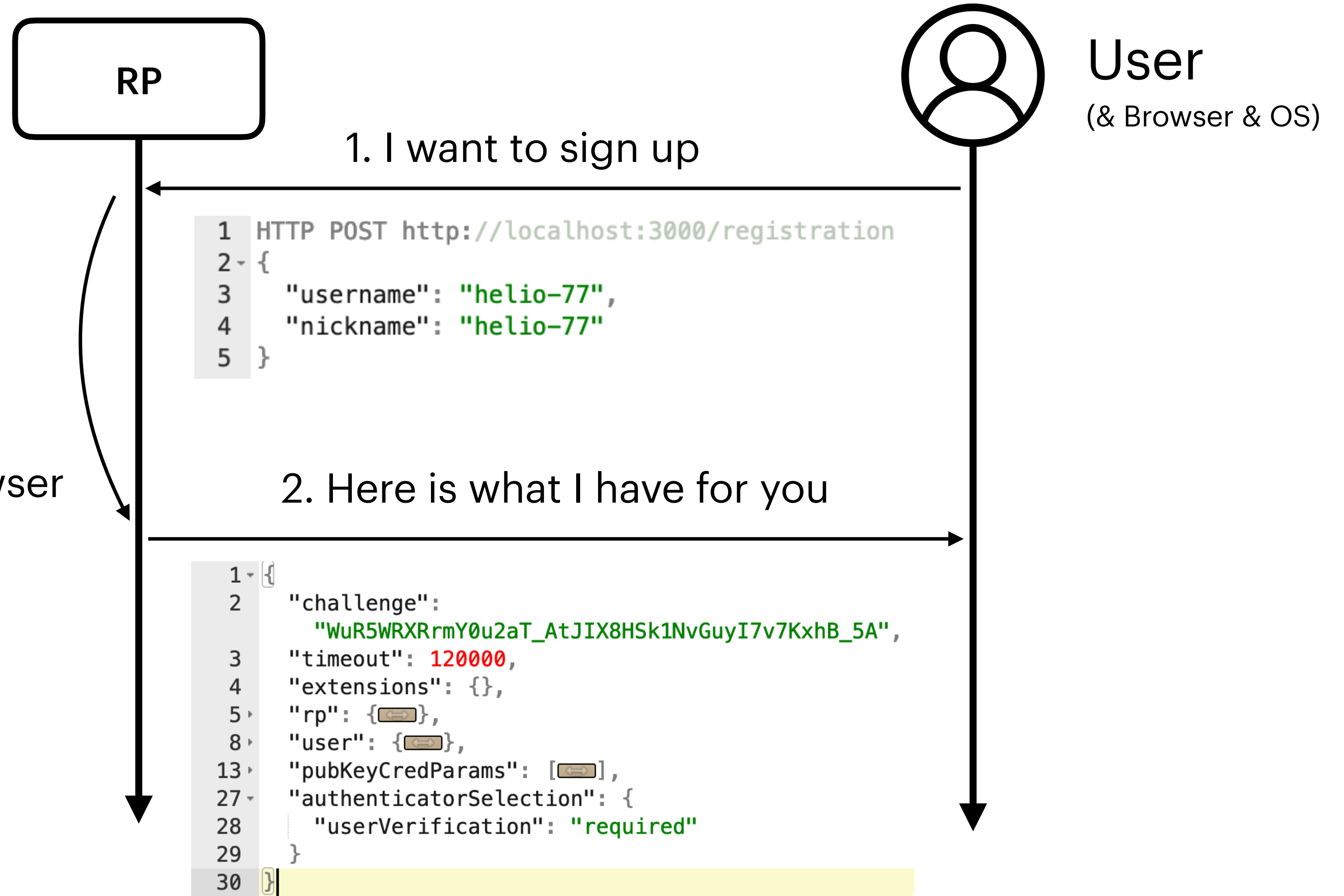


Registration - under the hood



Registration - under the hood

1. Generate Webauthn User Id
2. Load your app WebAuthn settings
3. Create a challenge
4. Return a JSON back to the user/browser



Registration - under the hood

```
1- {
2-   "rp": {
3-     "name": "WebAuthn Rails Demo App"
4-   },
5-   "timeout": 120000,
6-   "extensions": {},
7-   "pubKeyCredParams": [
8-     {
9-       "type": "public-key",
10-      "alg": -7
11-     },
12-     {
13-       "type": "public-key",
14-       "alg": -37
15-     },
16-     {
17-       "type": "public-key",
18-       "alg": -257
19-     }
20-   ],
21-   "authenticatorSelection": {
22-     "userVerification": "required"
23-   },
24-
25-   "user": {
26-     "name": "helio-77",
27-     "id": "3Ea7RGkd1TLkuK9808VohV5
28-       -5LJ24NsYinviaYCC0sm_V1zwcSxlGn01bMlVf7kE2vEw8skw8nsIARRLBokuQ",
29-     "displayName": "helio-77"
30-   },
31-   "challenge": "WuR5WRXRrmY0u2aT_AtJIX8Hsk1NvGuyI7v7KxhB_5A"
32- }
```

Application settings:

- Timeout is in milliseconds
- pubKeyCredParams are the algorithms your app decides to support. Those values represent: "ES256", "PS256", "RS256".
- userVerification required

Created for this user's session:

- id is based on Webauthn User handle specification
- challenge is used during the verification



Registration - under the hood

README MIT license Security

```
WebAuthn.configure do |config|
  # This value needs to match `window.location.origin` evaluated by
  # the User Agent during registration and authentication ceremonies.
  config.origin = "https://auth.example.com"

  # Relying Party name for display purposes
  config.rp_name = "Example Inc."

  # Optionally configure a client timeout hint, in milliseconds.
  # This hint specifies how long the browser should wait for any
  # interaction with the user.
  # This hint may be overridden by the browser.
  # https://www.w3.org/TR/webauthn/#dom-publickeycredentialcreationoptions-timeout
  # config.credential_options_timeout = 120_000

  # You can optionally specify a different Relying Party ID
  # (https://www.w3.org/TR/webauthn/#relying-party-identifier)
  # if it differs from the default one.
  #
  # In this case the default would be "auth.example.com", but you can set it to
  # the suffix "example.com"
  #
  # config.rp_id = "example.com"

  # Configure preferred binary-to-text encoding scheme. This should match the encoding scheme
  # used in your client-side (user agent) code before sending the credential to the server.
  # Supported values: `:base64url` (default), `:base64` or `false` to disable all encoding.
  #
  # config.encoding = :base64url

  # Possible values: "ES256", "ES384", "ES512", "PS256", "PS384", "PS512", "RS256", "RS384", "RS512"
  # Default: ["ES256", "PS256", "RS256"]
  #
  # config.algorithms << "ES384"
end
```

Source: <https://github.com/cedarcode/webauthn-ruby#configuration>



Registration - under the hood

GitLab.org / GitLab



GitLab

```
1 # frozen_string_literal: true
2
3 WebAuthn.configure do |config|
4   # This value needs to match `window.location.origin` evaluated by
5   # the User Agent during registration and authentication ceremonies.
6   config.origin = Settings.gitlab['base_url']
7
8   # Relying Party name for display purposes
9   # config.rp_name = "Example Inc."
10
11  # Optionally configure a client timeout hint, in milliseconds.
12  # This hint specifies how long the browser should wait for any
13  # interaction with the user.
14  # This hint may be overridden by the browser.
15  # https://www.w3.org/TR/webauthn/#dom-publickeycredentialcreationoptions-timeout
16  # config.credential_options_timeout = 120_000
17
18  # You can optionally specify a different Relying Party ID
19  # (https://www.w3.org/TR/webauthn/#relying-party-identifier)
20  # if it differs from the default one.
21  #
22  # In this case the default would be "auth.example.com", but you can set it to
23  # the suffix "example.com"
24  #
25  # config.rp_id = "example.com"
26
27  # Configure preferred binary-to-text encoding scheme. This should match the encoding scheme
28  # used in your client-side (user agent) code before sending the credential to the server.
29  # Supported values: `:base64url` (default), `:base64` or `false` to disable all encoding.
30  #
31  config.encoding = :base64
32
33  # Possible values: "ES256", "ES384", "ES512", "PS256", "PS384", "PS512", "RS256", "RS384", "RS512", "RS1"
34  # Default: ["ES256", "PS256", "RS256"]
35  #
36  # config.algorithms << "ES384"
37 end
38
```

Source: <https://gitlab.com/gitlab-org/gitlab/-/blob/master/config/initializers/webauthn.rb>



FOSDEM 2024 - Ruby devroom

Registration - under the hood

webauthn-ruby / lib / webauthn.rb 

 grzuy feat: expose API methods via WebAuthn::Credential 

Code

Blame

15 lines (12 loc) · 367 Bytes

Raw

```
1 # frozen_string_literal: true
2
3 require "webauthn/configuration"
4 require "webauthn/credential"
5 require "webauthn/credential_creation_options"
6 require "webauthn/credential_request_options"
7 require "webauthn/version"
8
9 module WebAuthn
10   TYPE_PUBLIC_KEY = "public-key"
11
12   def self.generate_user_id
13     configuration.encoder.encode(SecureRandom.random_bytes(64))
14   end
15 end
```

user id, generated by webauthn-ruby gem, based on Webauthn User handle specification

Source: <https://github.com/cedarcode/webauthn-ruby/blob/master/lib/webauthn.rb>



Registration - under the hood

```
webauthn-ruby / lib / webauthn / public_key_credential / options.rb

Code Blame 73 lines (55 loc) · 1.53 KB Raw

6   module WebAuthn
7     class PublicKeyCredential
8       class Options
19      def challenge
20        encoder.encode(raw_challenge)
21      end
```

challenge, generated by webauthn-ruby gem

```
webauthn-ruby / lib / webauthn / public_key_credential / options.rb

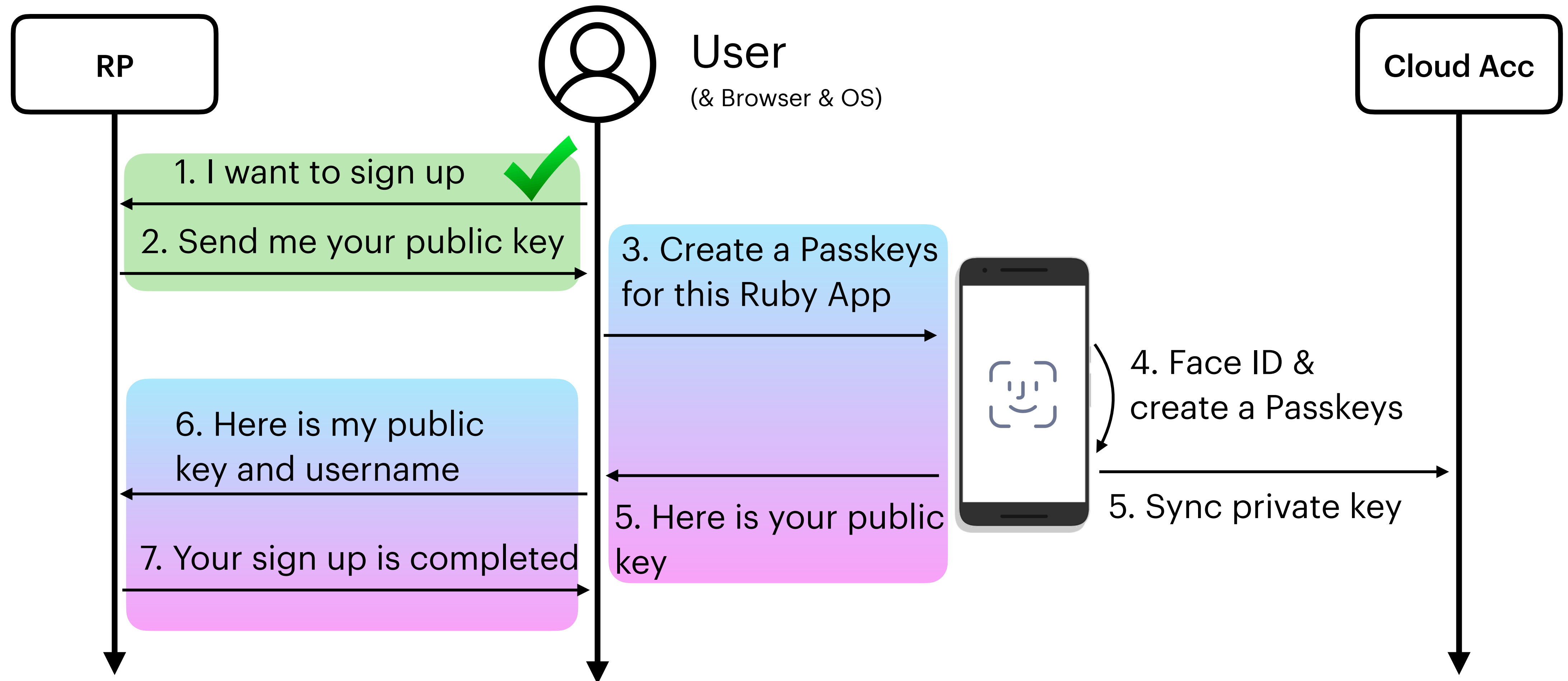
Code Blame 73 lines (55 loc) · 1.53 KB Raw

6   module WebAuthn
7     class PublicKeyCredential
8       class Options
56      def raw_challenge
57        @raw_challenge ||= SecureRandom.random_bytes(CHALLENGE_LENGTH)
58      end
```

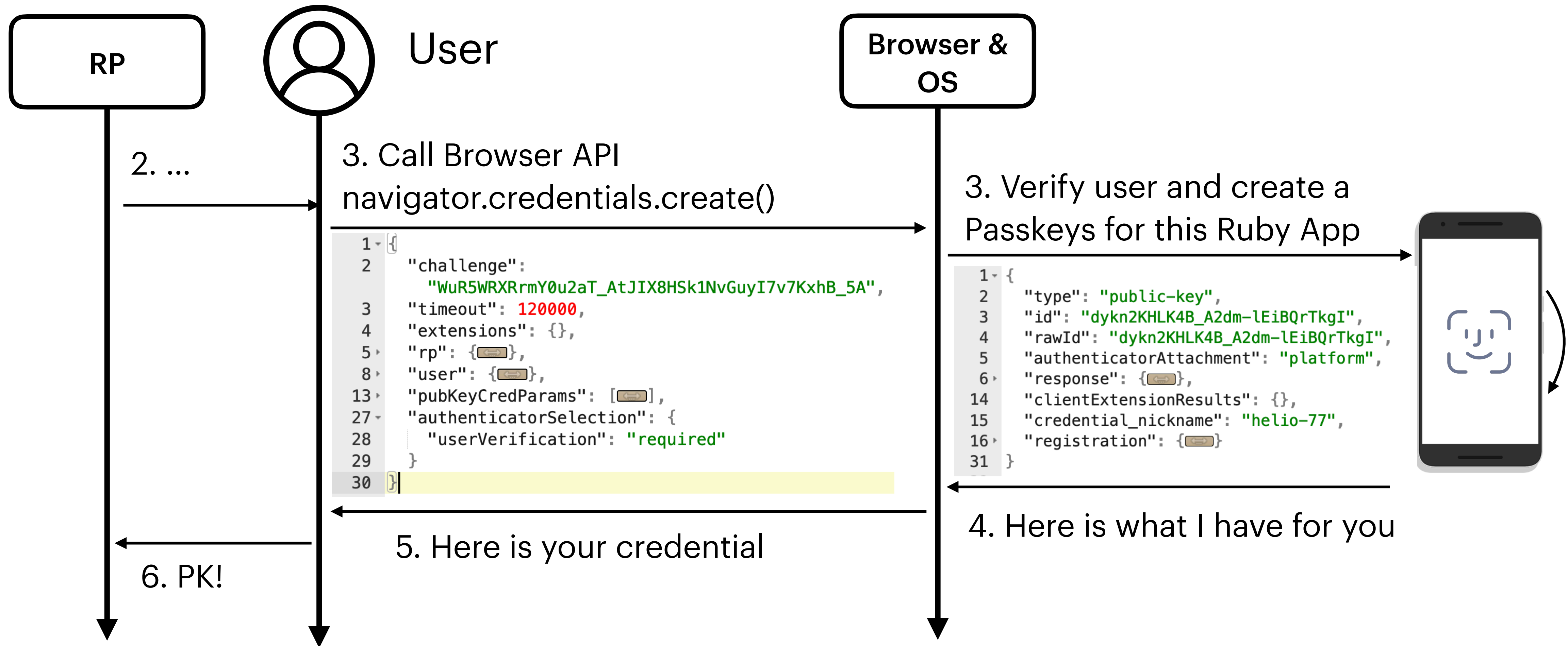
Source: https://github.com/cedarcode/webauthn-ruby/blob/master/lib/webauthn/public_key_credential/options.rb



Registration - under the hood



Registration - under the hood



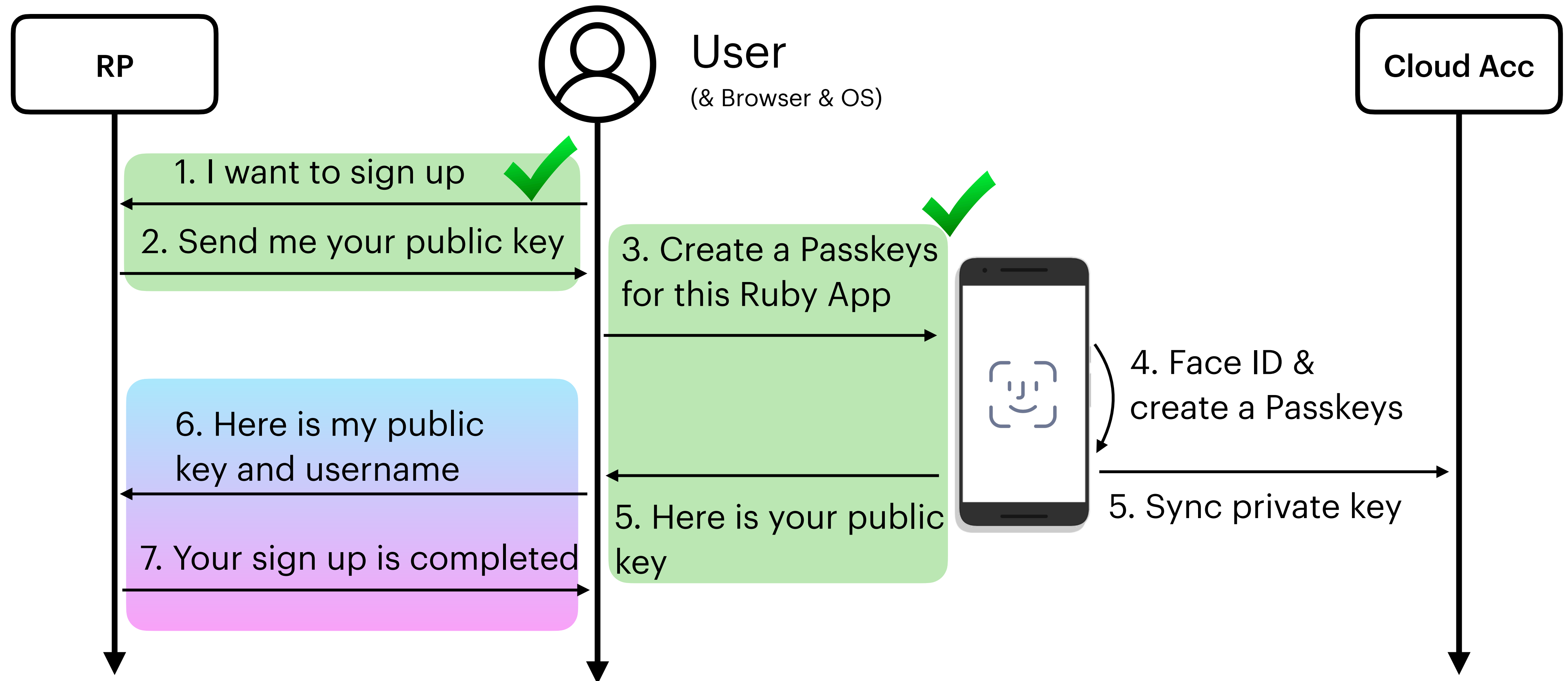
Registration - under the hood

Browser response to create
credential API call:
navigator.credentials.create()

```
1- {
2  "type": "public-key",
3  "id": "dykn2KHLK4B_A2dm-lEiBQrTkgI",
4  "rawId": "dykn2KHLK4B_A2dm-lEiBQrTkgI",
5  "authenticatorAttachment": "platform",
6- "response": {
7  "clientDataJSON": "eyJ0eXBliJoid2ViYXV0aG4uY3JlYXRlIiwia2hhbGxlbmdlIjoieYzZScEx5WEFDMlFPUUp
      FPUUpNkp3SG9tTjltOTZJUdDM2QUFE0XEwcDM5aWQ5ZyIsIm9yaWdpbiI6Imh0dHA6Ly9sb2NhbGhvc3Q6MzAwM
      zAwMCJ9",
8  "attestationObject":
      "o2NmbXRkbm9uZWdhdHRTdG10oGhhdXRoRGF0YViYSZYN5Yg0jGh0NBcPZHZgW4_krrmihjLHmVzzuoMdl2
      NdAAAAAAAAAAAAAAAAAAAAAAAAAFHcpJ9ihyyuAfwNnZvpRIgUK05ICpQECAyYgASFYIEqb6yu7ABxID
      YxiIbV8cbIf_MEIfP8MPsSRAGGzXSyCIlgg18uv8JMEfygrd70xEQELLIPoHQQ001iuKboaTWHnYac",
9- "transports": [
10  "internal",
11  "hybrid"
12 ]
13 },
14 "clientExtensionResults": {},
15 "credential_nickname": "helio-77",
16- "registration": {
17  "type": "public-key",
18  "id": "dykn2KHLK4B_A2dm-lEiBQrTkgI",
19  "rawId": "dykn2KHLK4B_A2dm-lEiBQrTkgI",
20  "authenticatorAttachment": "platform",
21- "response": {
22  "clientDataJSON":
      "eyJ0eXBliJoid2ViYXV0aG4uY3JlYXRlIiwia2hhbGxlbmdlIjoieYzZScEx5WEFDMlFPUUpNkp3SG9t
      TjltOTZJUdDM2QUFE0XEwcDM5aWQ5ZyIsIm9yaWdpbiI6Imh0dHA6Ly9sb2NhbGhvc3Q6MzAwMzAwMzAwM
      zAwMCJ9",
23  "attestationObject":
      "o2NmbXRkbm9uZWdhdHRTdG10oGhhdXRoRGF0YViYSZYN5Yg0jGh0NBcPZHZgW4_krrmihjLHmVzzuoM
      l2NdAAAAAAAAAAAAAAAAAAAAAAAAAFHcpJ9ihyyuAfwNnZvpRIgUK05ICpQECAyYgASFYIEqb6yu7A
      BxIDYxiIbV8cbIf_MEIfP8MPsSRAGGzXSyCIlgg18uv8JMEfygrd70xEQELLIPoHQQ001iuKboaTWHnYa
      c",
24- "transports": [
25  "internal",
26  "hybrid"
27 ]
28 },
29 "clientExtensionResults": {}
30 }
31 }
```

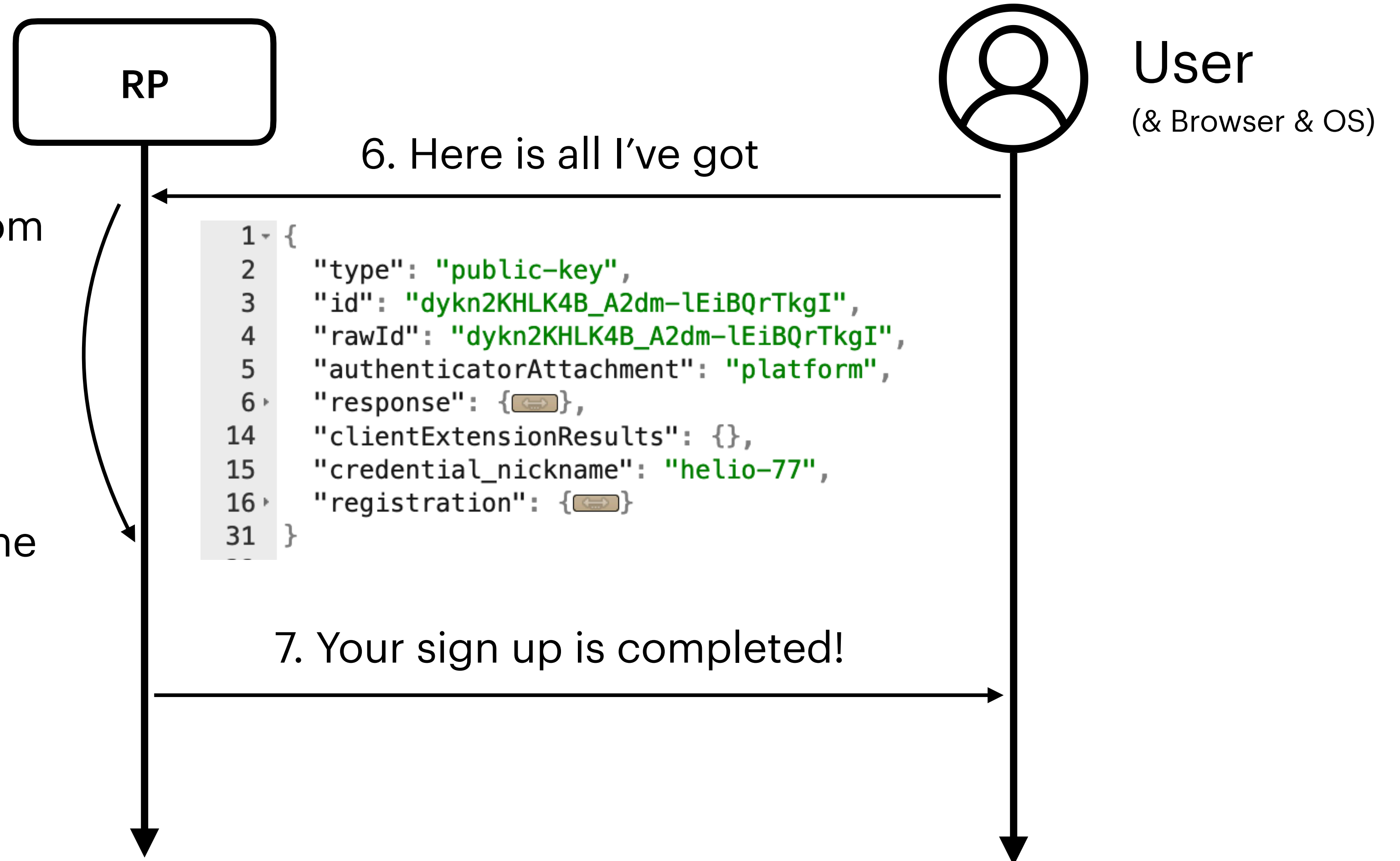


Registration - under the hood



Registration - under the hood

1. Verify the data with the challenge from the first step
2. Creating (or finalize) the user record
3. Create the passkeys
4. Return a success response back to the user/browser



Registration - under the hood

1. Verify the data with the challenge from the first step

```
webauthn-rails-demo-app / app / controllers / registrations_controller.rb ↑ Top  
  
Code Blame 61 lines (51 loc) · 1.71 KB Raw Copy Download Edit Dropdown  
  
31     def callback  
32  
33  
34     begin  
35         webauthn_credential = relying_party.verify_registration(  
36             params,  
37             session[:current_registration][:challenge],  
38             user_verification: true,  
39         )
```

Source: https://github.com/cedarcode/webauthn-rails-demo-app/blob/master/app/controllers/registrations_controller.rb



Registration - under the hood

`verify_registration` stack trace inside webauthn-ruby gem

1. WebAuthn::RelyingParty.verify_registration
2. WebAuthn::PublicKeyCredential.verify
3. WebAuthn::PublicKeyCredentialWithAttestation.verify
4. WebAuthn::AuthenticatorResponse.verify
5. WebAuthn::AuthenticatorAttestationResponse.verify



Registration - under the hood

1. WebAuthn::RelyingParty.verify_registration

webauthn-ruby / lib / webauthn / relying_party.rb

Code

Blame

120 lines (102 loc) · 3.63 KB

```
8     module WebAuthn
11         class RelyingParty
12
13         end
14     end
15
16     def verify_registration(raw_credential, challenge, user_verification: nil)
17         webauthn_credential = WebAuthn::Credential.from_create(raw_credential, relying_party: self)
18
19         if webauthn_credential.verify(challenge, user_verification: user_verification)
20             webauthn_credential
21         end
22     end
23 end
```

Source: https://github.com/cedarcode/webauthn-ruby/blob/master/lib/webauthn/relying_party.rb



Registration - under the hood

2. WebAuthn::PublicKeyCredential.verify

```
webauthn-ruby / lib / webauthn / public_key_credential.rb  
  
Code Blame 94 lines (74 loc) · 2.27 KB  
  
5     module WebAuthn  
41     def verify(challenge, *_args)  
47  
48         valid_type? || raise("invalid type")  
49         valid_id? || raise("invalid id")  
50  
51         true  
52     end
```

Source: https://github.com/cedarcode/webauthn-ruby/blob/master/lib/webauthn/public_key_credential.rb



Registration - under the hood

3. WebAuthn::PublicKeyCredentialWithAttestation.verify

```
webauthn-ruby / lib / webauthn / public_key_credential_with_attestation.rb

Code Blame 30 lines (23 loc) · 666 Bytes

6   module WebAuthn
7     class PublicKeyCredentialWithAttestation < PublicKeyCredential
12  def verify(challenge, user_verification: nil)
13     super
14
15     response.verify(encoder.decode(challenge), user_verification: user_verification)
16
17     true
18  end
```

Source: https://github.com/cedarcode/webauthn-ruby/blob/master/lib/webauthn/public_key_credential_with_attestation.rb



Registration - under the hood

webauthn-ruby / lib / webauthn / authenticator_response.rb

Code Blame 115 lines (90 loc) · 3.14 KB

```
7   module WebAuthn
21     class AuthenticatorResponse
27   def verify(expected_challenge, expected_origin = nil, user_verification: nil, rp_id: nil)
28     expected_origin ||= relying_party.origin || raise("Unspecified expected origin")
29     rp_id ||= relying_party.id
30
31     verify_item(:type)
32     verify_item(:token_binding)
33     verify_item(:challenge, expected_challenge)
34     verify_item(:origin, expected_origin)
35     verify_item(:authenticator_data)
36     verify_item(:rp_id, rp_id || rp_id_from_origin(expected_origin))
37
38     if !relying_party.silent_authentication
39       verify_item(:user_presence)
40     end
41
42     if user_verification
43       verify_item(:user_verified)
44     end
45
46     true
end
```

4. WebAuthn::
AuthenticatorResponse
.verify

Source: https://github.com/cedarcode/webauthn-ruby/blob/master/lib/webauthn/authenticator_response.rb



Registration - under the hood

4. WebAuthn::AuthenticatorResponse.verify_challenge (side note)

```
webauthn-ruby / lib / webauthn / authenticator_response.rb  
  
Code Blame 115 lines (90 loc) · 3.14 KB  
  
7     module WebAuthn  
21     class AuthenticatorResponse  
...  
81         def valid_challenge?(expected_challenge)  
82             OpenSSL.secure_compare(client_data.challenge, expected_challenge)  
83         end
```

Source: https://github.com/cedarcode/webauthn-ruby/blob/master/lib/webauthn/authenticator_response.rb



Registration - under the hood

5. WebAuthn::AuthenticatorAttestationResponse.verify

```
webauthn-ruby / lib / webauthn / authenticator_attestation_response.rb

Code Blame 83 lines (63 loc) · 2.24 KB

13  module WebAuthn
18  class AuthenticatorAttestationResponse < AuthenticatorResponse
33  def initialize(attestation_object:, **options)
40  def verify(expected_challenge, expected_origin = nil, user_verification: nil, rp_id: nil)
41    super
42
43    verify_item(:attested_credential)
44    if relying_party.verify_attestation_statement
45      verify_item(:attestation_statement)
46    end
47
48    true
49  end
```

Source: https://github.com/cedarcode/webauthn-ruby/blob/master/lib/webauthn/authenticator_attestation_response.rb



Registration - under the hood

Steps the server runs with the user data:

1. Verify the data with the challenge from the first step ✓
2. Create (or finalize) the user record
3. Create the passkeys
4. Return a success response back to the user/browser



Registration - under the hood

3. Create the passkeys (in your Ruby app)

```
webauthn-rails-demo-app / app / controllers / registrations_controller.rb

Code Blame 61 lines (51 loc) · 1.71 KB Raw

31     def callback
41         credential = user.credentials.build(
42             external_id: Base64.strict_encode64(webauthn_credential.raw_id),
43             nickname: params[:credential_nickname],
44             public_key: webauthn_credential.public_key,
45             sign_count: webauthn_credential.sign_count
46         )
```

Remember: var webauthn_credential
type is: WebAuthn::PublicKeyCredential

Source: https://github.com/cedarcode/webauthn-rails-demo-app/blob/master/app/controllers/registrations_controller.rb



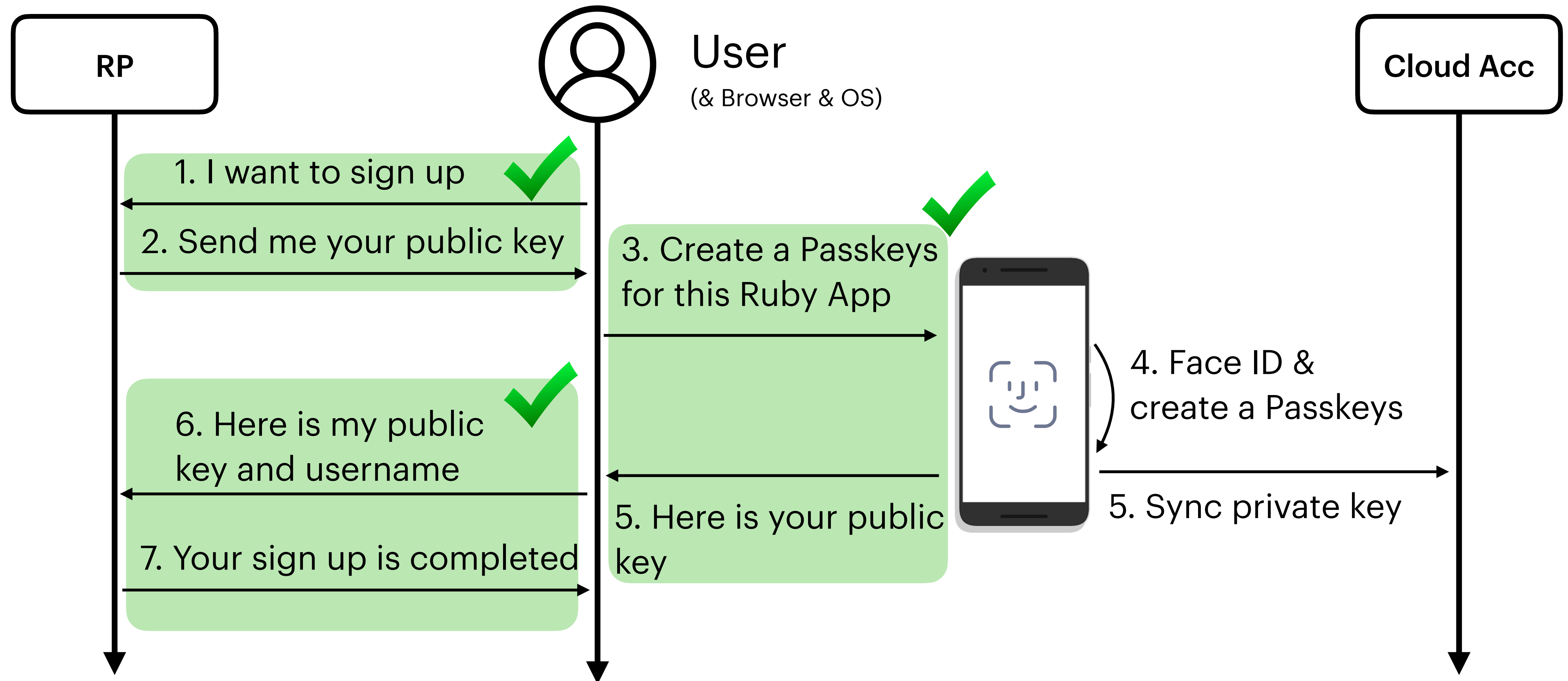
Registration - under the hood

3. Create the passkeys (in your Ruby app)

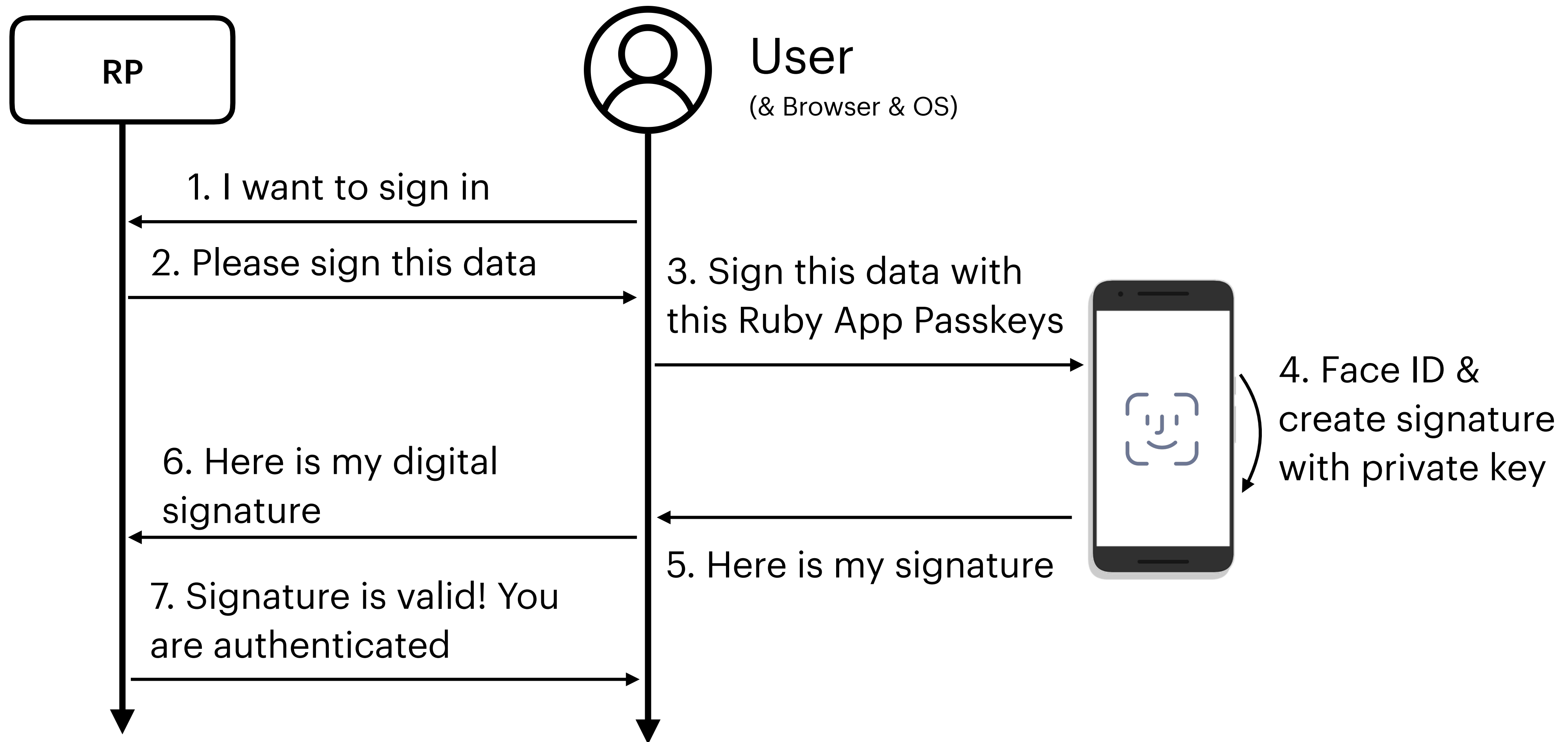
```
[10] pry(main)> Credential.last
  Credential Load (0.7ms) SELECT "credentials".* FROM "credentials" ORDER BY "credentials"."id" DESC LIMIT $1 [{"LIMIT", 1}]
=> #<Credential:0x000000010a575f20
  id: 9,
  external_id: "dykn2KHLK4B/A2dm+lEiBQrTkgI=",
  public_key: "[FILTERED]",
  user_id: 9,
  created_at: Thu, 14 Dec 2023 09:08:42.984827000 UTC +00:00,
  updated_at: Thu, 14 Dec 2023 09:08:42.984827000 UTC +00:00,
  nickname: "helio-77",
  sign_count: 0>
[11] pry(main)> Credential.last.public_key
  Credential Load (0.7ms) SELECT "credentials".* FROM "credentials" ORDER BY "credentials"."id" DESC LIMIT $1 [{"LIMIT", 1}]
=> "pQECAyYgASFYIEqb6yu7ABxIDYxiIbV8cbIf_MEifP8MPsSRAGGzXSyCIlgg18uv8JMEfygrd70xEQELLIPoHQQ001iuKboaTWHnYac"
```



Registration - under the hood

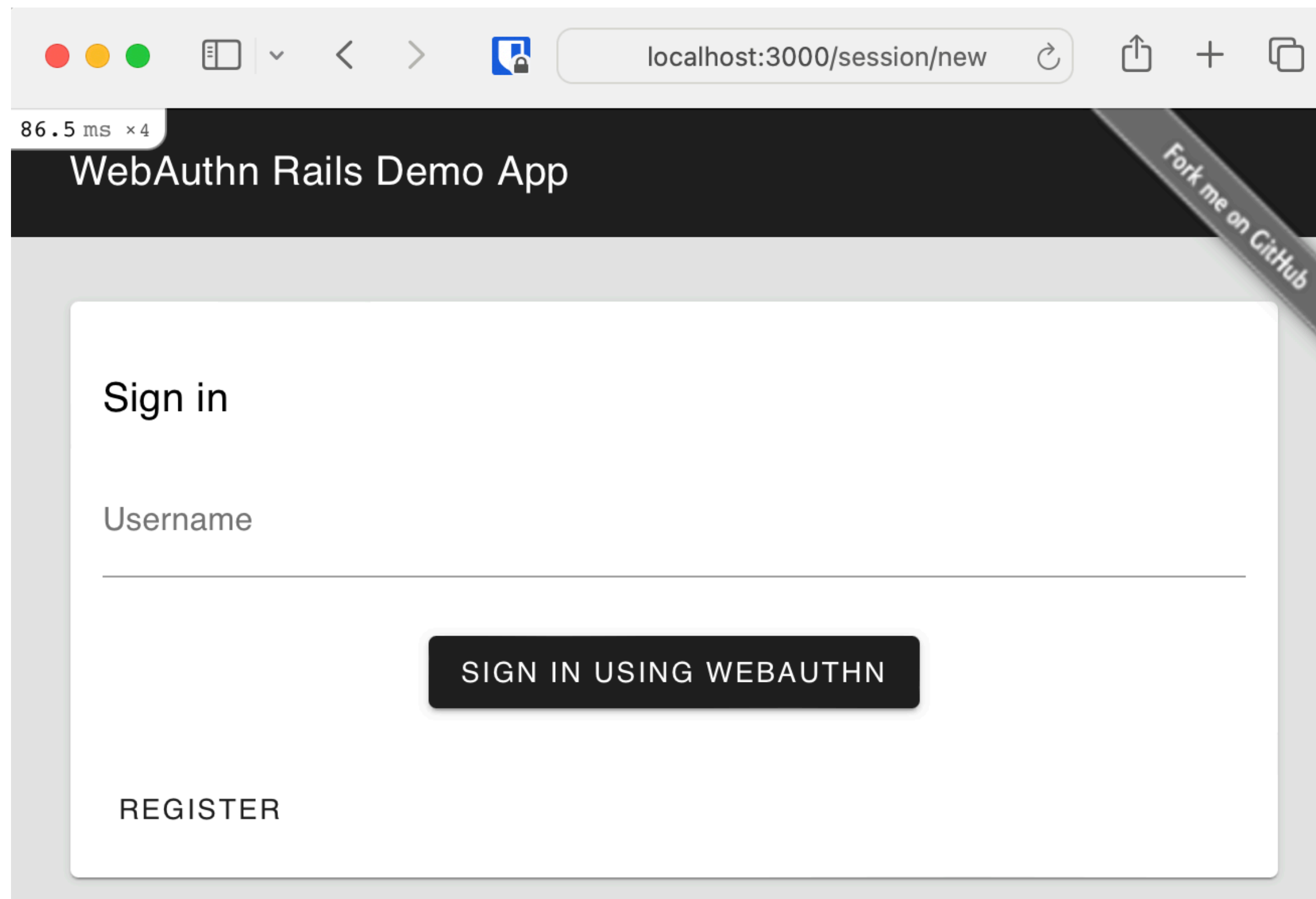


Authentication



Authentication - under the hood

- Reference app: `cedarcode/webauthn-rails-demo-app`
- Link: <https://github.com/cedarcode/webauthn-rails-demo-app>



**Consider yourself
invited!**



Hello Ruby!

The trailblazers:

- Gonzalo and Braulio from CedarCode: <https://www.cedarcodes.com>
- Petr Hlavicka: <https://petr.codes>
- Thomas Cannon: <https://thomascannon.me>



CedarCode

- Web Agency based in Uruguay
- Authors of webauthn-ruby gem



Gonzalo Rodriguez



Braulio Martinez



Source: <https://github.com/cedarcode/webauthn-ruby/blob/master/webauthn.gemspec>



webauthn-ruby gem

- Gonzalo released V0.0.0 on May, 9th 2018
- And so was webauthn-rails-demo-app

It is live: <https://webauthn.cedarcodes.com/>

- Latest release is v3.1.0, in December, 2023



Petr Hlavicka



- Petr is a Ruby on Rails developer

Can be found at: <https://petr.codes/>

- In 2021, he wrote an article:

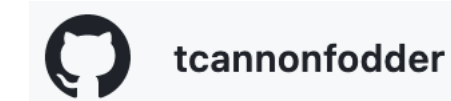
“Multi-Factor Authentication for Rails With WebAuthn and Devise”

Originally published at [HoneyBadger.io](https://www.honeybadger.io/blog/multi-factor-2fa-authentication-rails-webauthn-devise/) blog: <https://www.honeybadger.io/blog/multi-factor-2fa-authentication-rails-webauthn-devise/>

Companion Rails app: <https://github.com/CiTroNaK/webauthn-with-devise>



Thomas Cannon



- Creator of Ruby-Passkeys GitHub Org

<https://github.com/ruby-passkeys>

Can be found at: <https://thomascannon.me/>

- And the creator of gems:

warden-webauthn (v0.3.0)

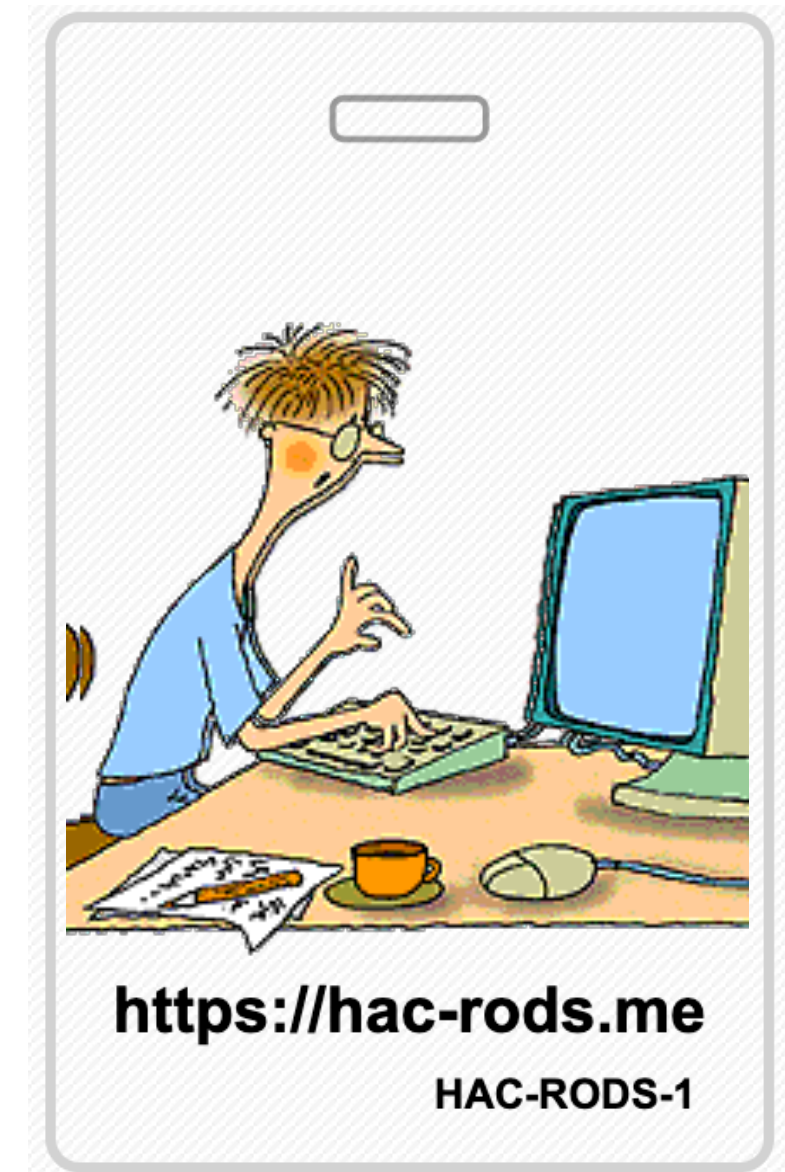
devise-passkeys (v0.3.0)

- And Rails template app “devise-passkeys-template”



This is it folks!

Questions?!



Thank you!

- Thank y'all for your time and your attention
- Thank you to all organizers of the Ruby Devroom!

<https://ruby.social/@hacrods>

