The World of Passkeys
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

- Ruby Conf Thailand: https://rubyconfth.com
- Ruby Conf Taiwan: https://2023.rubyconf.tw
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
The Passkeys Iceberg

2016: W3C, A Web API for accessing scoped credentials
1970s Public key cryptography
2FA or not 2FA

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

1. I want to sign up
2. Send me your public key
3. Create a Passkeys for this Ruby App
4. Face ID & create a Passkeys
5. Sync private key
6. Here is my public key and username
7. Your sign up is completed
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

1. I want to sign up
2. Send me your public key
3. Create a Passkeys for this Ruby App
4. Face ID & create a Passkeys
5. Here is your public key
6. Here is my public key and username
7. Your sign up is completed

RP

User
(& Browser & OS)

Cloud Acc
Registration - under the hood

1. I want to sign up

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

HTTP POST http://localhost:3000/registration

```json
{
  "username": "helio-77",
  "nickname": "helio-77"
}
```

2. Here is what I have for you

```json
{
  "challenge": "WuR5WRXrmmY0u2aT_AtJIX8hsK1NvGuyI7v7KxhB_5A",
  "timeout": 120000,
  "extensions": {},
  "rp": {
    "name": "",
    "id": ""
  },
  "user": {
    "name": "",
    "id": ""
  },
  "pubKeyCredParams": [],
  "authenticatorSelection": {
    "userVerification": "required"
  }
}
```
Registration - under the hood

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

```
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-   "user": {
25-     "name": "hello-77",
26-     "id": "3Ea7RGk0TLkuK9808VohV5-5LJ24N5yinvYaYCC0sm-V1zwgcSXlgN01bMLVf7kE2vEw8skw8nsIArRLBokuQ",
27-     "displayName": "hello-77"
28-   },
29-   "challenge": "WuR5WRXRrmY0u2aT_ATJI3XH5k1NvGuyfT7KxhB_SA"
30- }
```
Registration - under the hood

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",
  # Default: [:"ES256", "PS256", "RS256"]
  #
  # config.algorithms = "ES384"
end

Source: https://github.com/cedarcode/webauthn-ruby#configuration
Registration - under the hood

```ruby
# frozen_string_literal: true
WebAuthn.config do |config|
  # This value needs to match 'window.location.origin' evaluated by
  # the User Agent during registration and authentication ceremonies.
  config.origin = Settings.gitlab['base_url']

  # 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 = :base64

  # Possible values: "ES256", "ES384", "ES512", "PS256", "PS384", "PS512", "RS256", "RS384", "RS512", "RS1"
  # Default: ["ES256", "PS256", "RS256"]

  # config.algorithms << "ES384"
end
```

Source: https://gitlab.com/gitlab-org/gitlab/-/blob/master/config/initializers/webauthn.rb
Registration - under the hood

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

```ruby
# frozen_string_literal: true
require "webauthn/configuration"
require "webauthn/credential"
require "webauthn/credential_creation_options"
require "webauthn/credential_request_options"
require "webauthn/version"

module WebAuthn
  TYPE_PUBLIC_KEY = "public-key"

  def self.generate_user_id
    configuration.encoder.encode(SecureRandom.random_bytes(64))
  end
end
```

Source: https://github.com/cedarcode/webauthn-ruby/blob/master/lib/webauthn.rb
Registration - under the hood


```
module WebAuthn
  class PublicKeyCredential
    class Options
      def challenge
        encoder.encode(raw_challenge)
      end
    end
  end
end
```

```
module WebAuthn
  class PublicKeyCredential
    class Options
      def raw_challenge
        @raw_challenge ||= SecureRandom.random_bytes(CHALLENGE_LENGTH)
      end
    end
  end
end
```

challenge, generated by webauthn-ruby gem
Registration - under the hood

1. I want to sign up
2. Send me your public key
3. Create a Passkeys for this Ruby App
4. Face ID & create a Passkeys
5. Sync private key
6. Here is my public key and username
7. Your sign up is completed
Registration - under the hood

3. Call Browser API
navigator.credentials.create()

3. Verify user and create a Passkeys for this Ruby App

5. Here is your credential

6. PK!

4. Here is what I have for you

FOSDEM 2024 - Ruby devroom
Registration - under the hood

Browser response to create credential API call: `navigator.credentials.create()`
Registration - under the hood

1. I want to sign up
2. Send me your public key
3. Create a Passkeys for this Ruby App
4. Face ID & create a Passkeys
5. Sync private key
6. Here is my public key and username
7. Your sign up is completed
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

6. Here is all I’ve got

```json
1 - {
  2  "type": "public-key",
  3  "id": "dykn2KHLK4B_A2dm-lEiBQrTkgil",
  4  "rawId": "dykn2KHLK4B_A2dm-lEiBQrTkgi",
  5  "authenticatorAttachment": "platform",
  6  "response": {
  7  "clientExtensionResults": {},
  8  "credentialNickname": "hello-77",
  9  "registration": {
  10  }
  11  }
  12 }
```

7. Your sign up is completed!
Registration - under the hood

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

```ruby
# registrations_controller.rb

def callback
  begin
    webauthn_credential = relying_party.verify_registration(
      params,
      session[:current_registration][:challenge],
      user_verification: true,
    )
  end
end
```

Source: https://github.com/cedarcode/webauthn-rails-demo-app/blob/master/app/controllers/registrations_controller.rb
`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
1. WebAuthn::RelyingParty.verify_registration

```ruby
module WebAuthn
  class RelyingParty
    def verify_registration(raw_credential, challenge, user_verification: nil)
      webauthn_credential = WebAuthn::Credential.from_create(raw_credential, relying_party: self)
      if webauthn_credential.verify(challenge, user_verification: user_verification)
        webauthn_credential
      end
    end
  end
end
```

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

2. WebAuthn::PublicKeyCredential.verify

```ruby
module WebAuthn
  def verify(challenge, *args)
    valid_type? || raise("invalid type")
    valid_id? || raise("invalid id")
    true
  end
end
```

Registration - under the hood

3. WebAuthn::PublicKeyCredentialWithAttestation.verify

```ruby
module WebAuthn
  class PublicKeyCredentialWithAttestation < PublicKeyCredential
    def verify(challenge, user_verification: nil)
      super
      response.verify(encoder.decode(challenge), user_verification: user_verification)
      true
    end
  end
end
```

4. WebAuthn::
AuthenticatorResponse
.verify
Registration - under the hood

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

```ruby
module WebAuthn
  class AuthenticatorResponse
    def valid_challenge?(expected_challenge)
      OpenSSL.secure_compare(client_data.challenge, expected_challenge)
    end
  end
end
```

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

5. WebAuthn::AuthenticatorAttestationResponse.verify

```ruby
module WebAuthn
  class AuthenticatorAttestationResponse < AuthenticatorResponse
    def initialize(atestation_object:, **options)
      super
      verify_item(:attested_credential)
      if relying_party.verify_attestation_statement
        verify_item(:attestation_statement)
      end
    end
    true
  end
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)

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

```ruby
31 def callback
32     credential = user.credential.build
33         external_id: Base64.strict_encode64(webauthn_credential.raw_id),
34         nickname: params[:credential_nickname],
35         public_key: webauthn_credential.public_key,
36         sign_count: webauthn_credential.sign_count
37     )
```

**Remember:** var webauthn_credential
type is: WebAuthn::PublicKeyCredential

Source: https://github.com/cedarcode/webauthn-rails-demo-app/blob/master/app/controllers/registrations_controller.rb
3. Create the passkeys (in your Ruby app)

```ruby
[10] pry(main)> Credential.last
Credential Load (0.7ms) SELECT "credentials".* FROM "credentials" ORDER BY "credentials"."id" DESC LIMIT $1 [["LIMIT", 1]]
=> #<Credential::0x0000000010a575f20
  id: 9,
  external_id: "dykn2KHLK4B/A2dm+1EiBQmTk6I=",
  public_key: "[FILTERED]",
  user_id: 9,
  created_at: Thu, 14 Dec 2023 09:00:42.984827000 UTC +00:00,
  updated_at: Thu, 14 Dec 2023 09:00:42.984827000 UTC +00:00,
  nickname: "hello-77",
  sign_count: 0
Credential Load (0.7ms) SELECT "credentials".* FROM "credentials" ORDER BY "credentials"."id" DESC LIMIT $1 [["LIMIT", 1]]
=> "pQECayYgASFYIEqqb6yu7ABxIDYxiIbV8cbIf_MEIfP8MPsSRAGzXSyC1gg18uv8JMEfygrd70xEQELLIPoHQQ01iuKboaTWHnYac"
```
Registration - under the hood

1. I want to sign up
2. Send me your public key
3. Create a Passkeys for this Ruby App
4. Face ID & create a Passkeys
5. Sync private key
6. Here is my public key and username
7. Your sign up is completed
Authentication

1. I want to sign in
2. Please sign this data
3. Sign this data with this Ruby App Passkeys
4. Face ID & create signature with private key
5. Here is my signature
6. Here is my digital signature
7. Signature is valid! You are authenticated
Shall we look inside?!
Authentication - under the hood

- Reference app: `cedarcode/webauthn-rails-demo-app`
- Link: [https://github.com/cedarcode/webauthn-rails-demo-app](https://github.com/cedarcode/webauthn-rails-demo-app)

Consider yourself invited!
Live Demo?! 

https://localhost:3000 ?

https://<the_actual_product_name>.com ?
Hello Ruby!

Passkeys in the Ruby Community
Hello Ruby!

The trailblazers:

• Gonzalo and Braulio from CedarCode: https://www.cedarcode.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.cedarcode.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 HoneyBagder.io 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!

[link](https://ruby.social/@hacrods)