Ruby /WebAuthnでのパスワードレス（パスキー）実装動向

コーポレートデザイン部
セキュリティグループ　板倉景子

Copyright© Medley, Inc. ALL RIGHTS RESERVED.

島根県出雲市出身

2005年 **Unisys** 総合技術研究所
システムエンジニア

2008年 **Microsoft** Consulting Services
技術コンサルタント

2015年 **IBM** セキュリティ事業部
Manager

2019年 **楽天グループ** エコシステムサービス部
Principal Information Security Specialist／
FIDO アライアンス
Japan Leadership Group/副座長

2023年 **メドレー** コーポレートデザイン部
全社情報セキュリティ推進責任者

Copyright© Medley, Inc. ALL RIGHTS RESERVED.

MEDLEY

パスワードレス認証（パスキー）に関わる最新動向について、Rubyでの実装の観点をふまえご紹介させていただきます。

1 認証手段の歴史と課題

2 パスキー（synced passkeys)の登場と関連技術

3 Rubyでのパスキー実装

Copyright© Medley, Inc. ALL RIGHTS RESERVED.

MEDLEY

1 **認証手段の歴史と課題**

2 パスキー（synced passkeys)の登場と関連技術

3 Rubyでのパスキー実装

Copyright© Medley, Inc. ALL RIGHTS RESERVED.

パスワード認証における課題

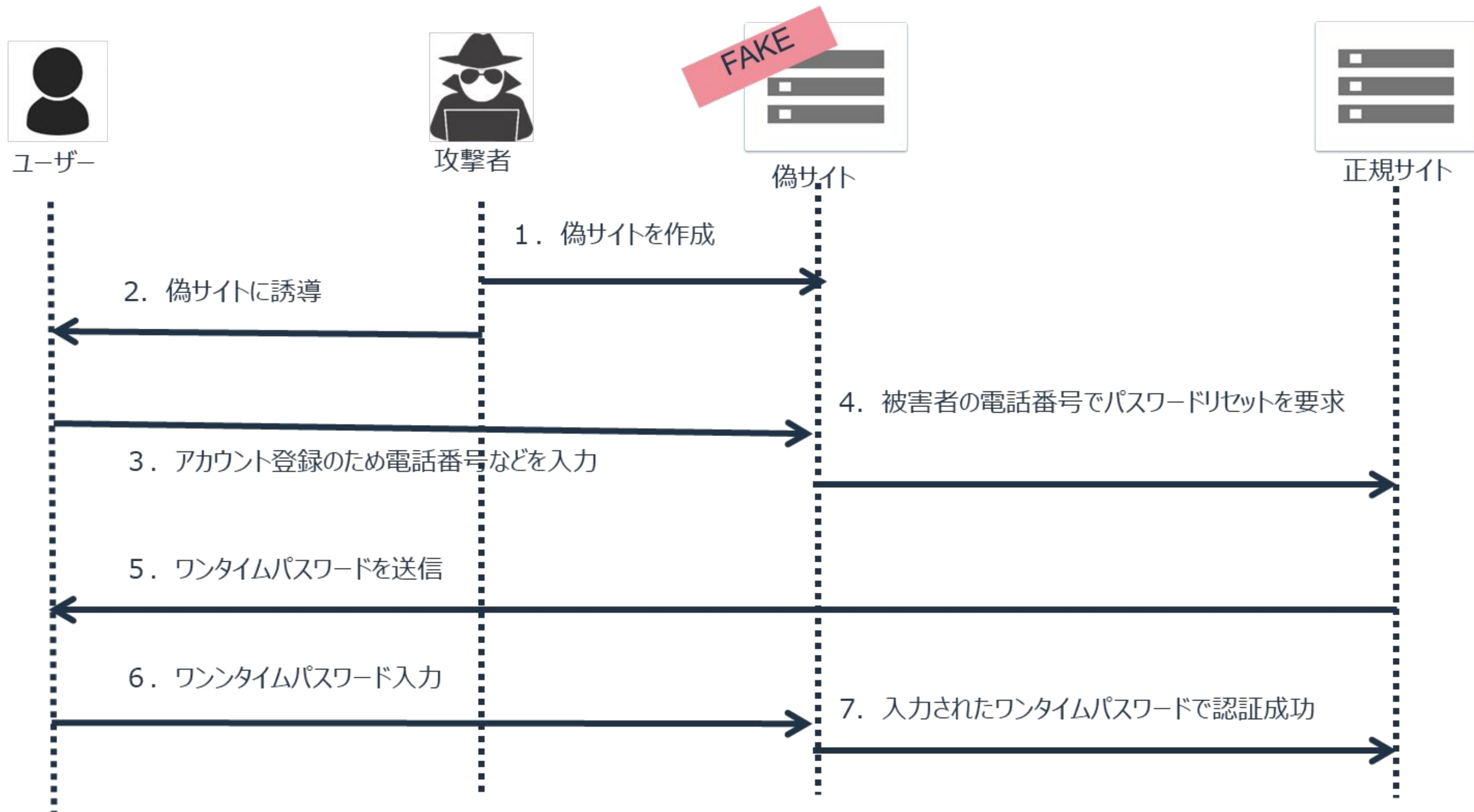
On the Internet, nobody knows you're a dog
「インターネット上ではあなたが犬だと誰も知らない」

- パスワードは覚えられない
  - 覚えやすい簡単なパスワードの設定
  - パスワードの使いまわし
- 多くのセキュリティインシデントがアカウント乗っ取りを起因として発生
- パスワード忘れによる問い合わせ対応、パスワードリセット運用コストの発生

Copyright© Medley, Inc. ALL RIGHTS RESERVED.

多要素認証における課題

パスワードリセットを悪用してアカウントを乗っ取る等、多要素認証でも防げない攻撃も報告されている。

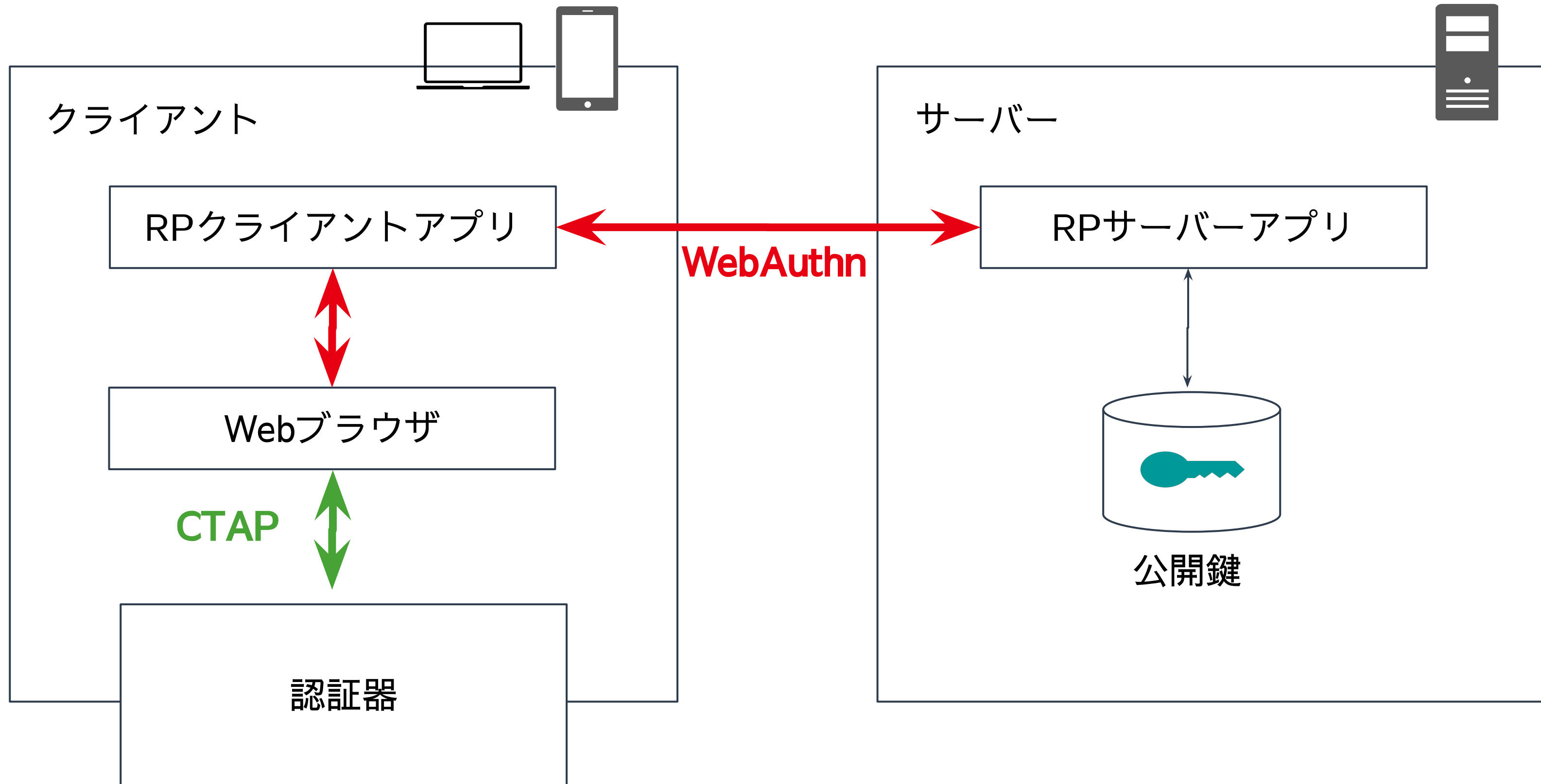


Copyright© Medley, Inc. ALL RIGHTS RESERVED.

従来のパスキー（device-bound passkeys）

MEDLEY

公開鍵暗号方式を活用し、認証器による所持をベースとした認証を行う。



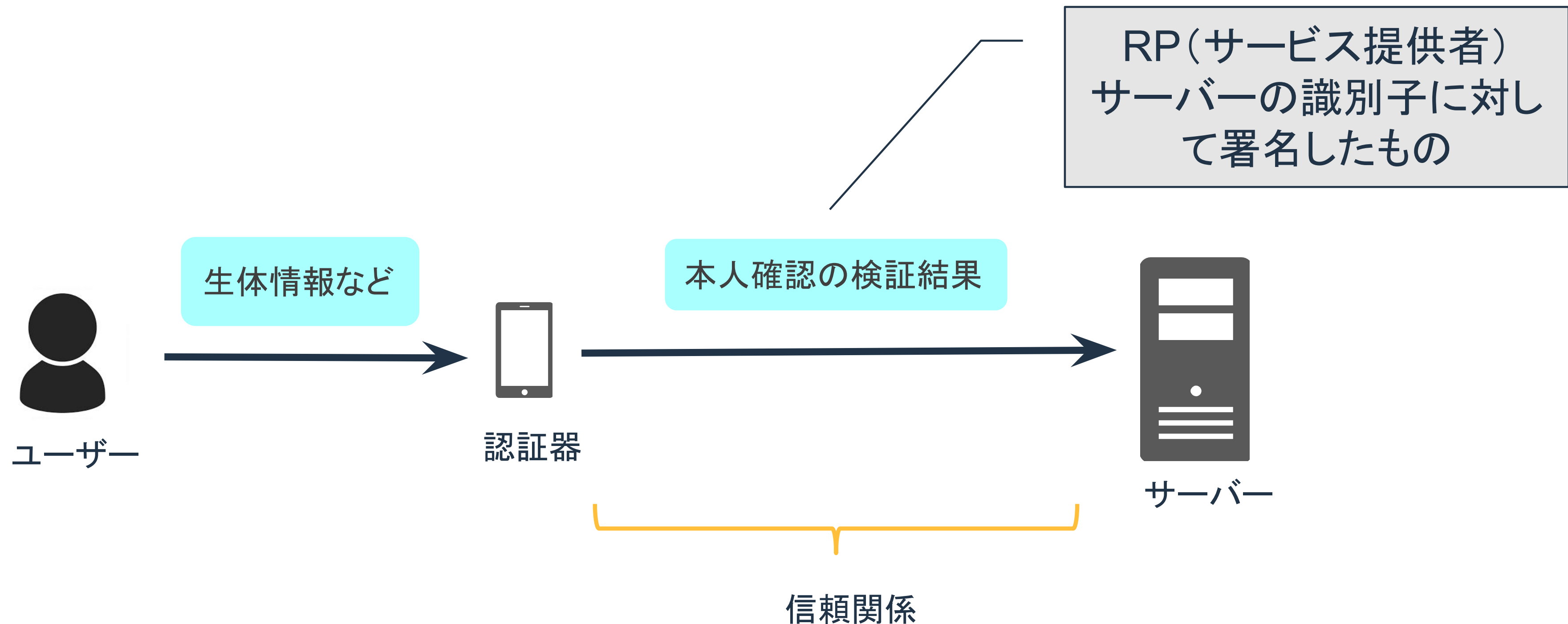
WHAT IS A PASSKEY?

Any passwordless FIDO credential is a passkey.

https://fidoalliance.org/passkeys/

Copyright© Medley, Inc. ALL RIGHTS RESERVED.

フィッシング耐性の一例として、認証器はRP（サービス提供者）サーバーの識別子に対して署名したものを送付し、RP（サービス提供者）サーバーはその署名を検証する。
＝中間者が入っていれば署名検証で検知できる＆RP（サービス提供者）サーバーとドメインが異なるサーバーでは認証できない。



Copyright© Medley, Inc. ALL RIGHTS RESERVED.

MEDLEY

1 認証手段の歴史と課題

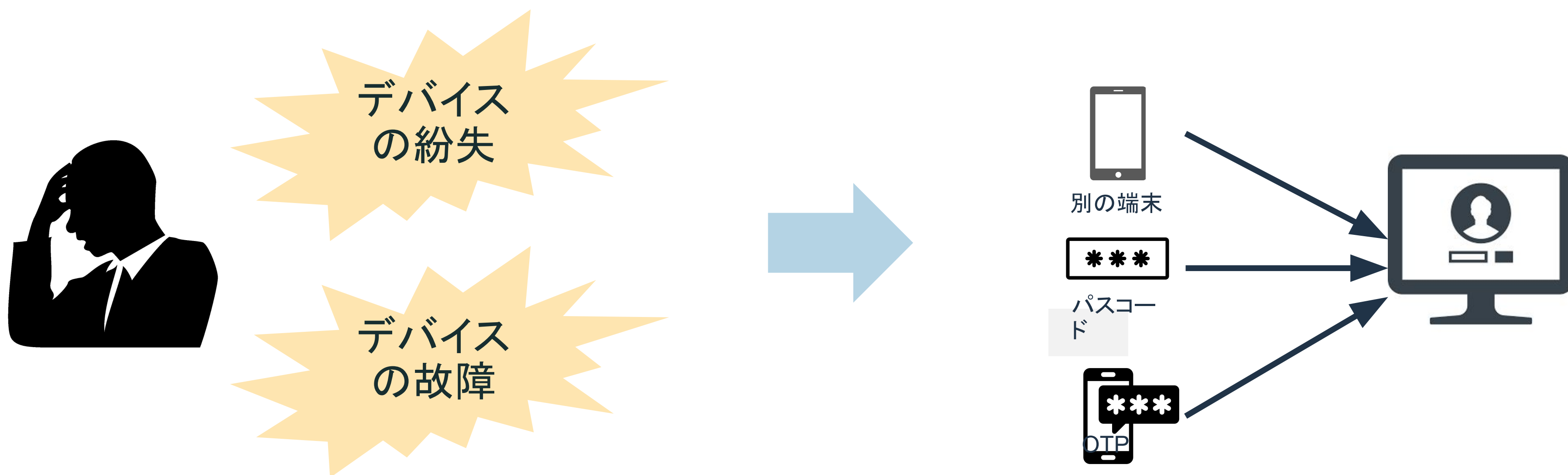
2 Synced Passkeysの登場と関連技術

3 Rubyでのパスキー実装

Copyright© Medley, Inc. ALL RIGHTS RESERVED.

従来のパスキーの課題

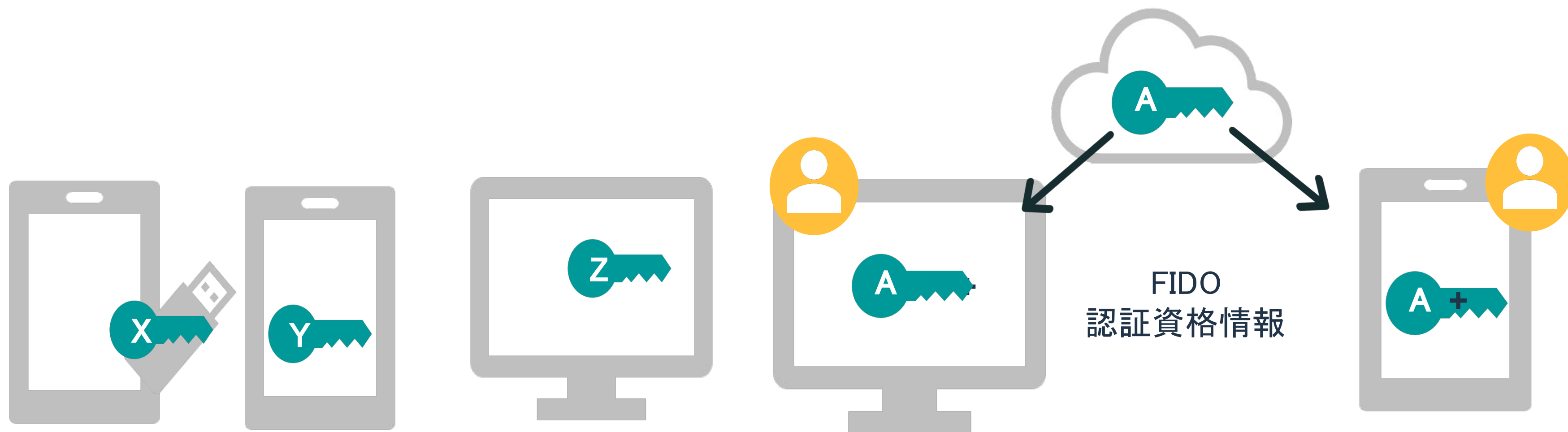
認証器の保持を前提とした認証方式のため、認証器自体を紛失した場合や破損した場合、どうやってアカウントを復旧させるかが課題であった。



Copyright© Medley, Inc. ALL RIGHTS RESERVED.

Synced passkeysとは

パスキーをパスキープロバイダー（クラウド事業者やパスワードマネージャ事業者）に保存し、FIDO認証に関する設定も移行するようにするもの。
※Device boundな資格情報の必要性についても引き続き議論されている。



Device-bound passkeys
（それぞれの端末に別々の鍵が保存される）

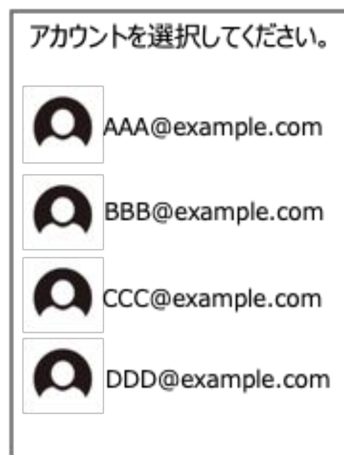
Synced passkeys
（同じ鍵が利用デバイスに保存される）

Copyright© Medley, Inc. ALL RIGHTS RESERVED.

Autofill UI – 自動入力 UI

現在のユーザーが 資格情報を持っているかどうかを確認し、リストとして表示する。
（パスキーもこのリストに含まれる）

ユーザーは、パスワードのみならずIDを入力する必要もなくなる。



Copyright© Medley, Inc. ALL RIGHTS RESERVED.

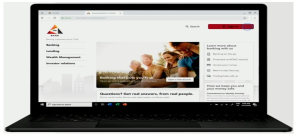
MEDLEY

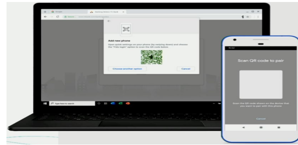
スマートフォンを認証器として活用可能。プラットフォームをまたいだPasskeysの活用にも利用できるBluetoothにもとづいた技術。

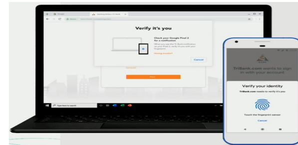
デスクトップPCで□サインイン

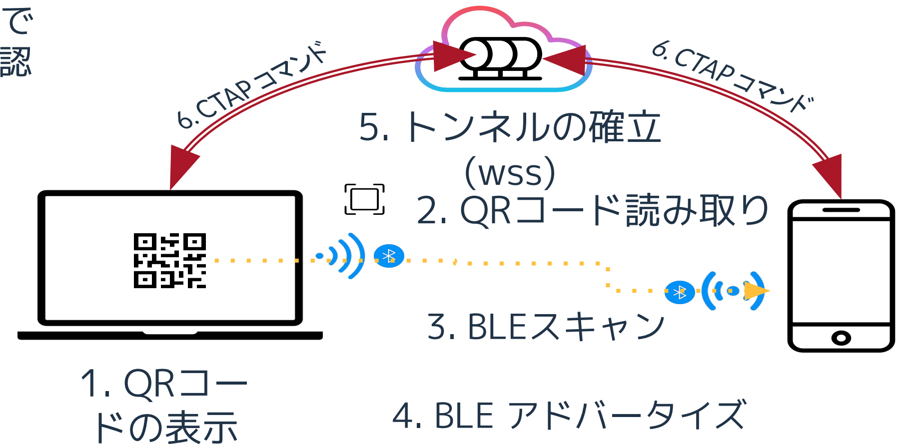
認証用QRコードをスマートフォンで読み取り

□ スマートフォンで本人確認を行い、認証成功



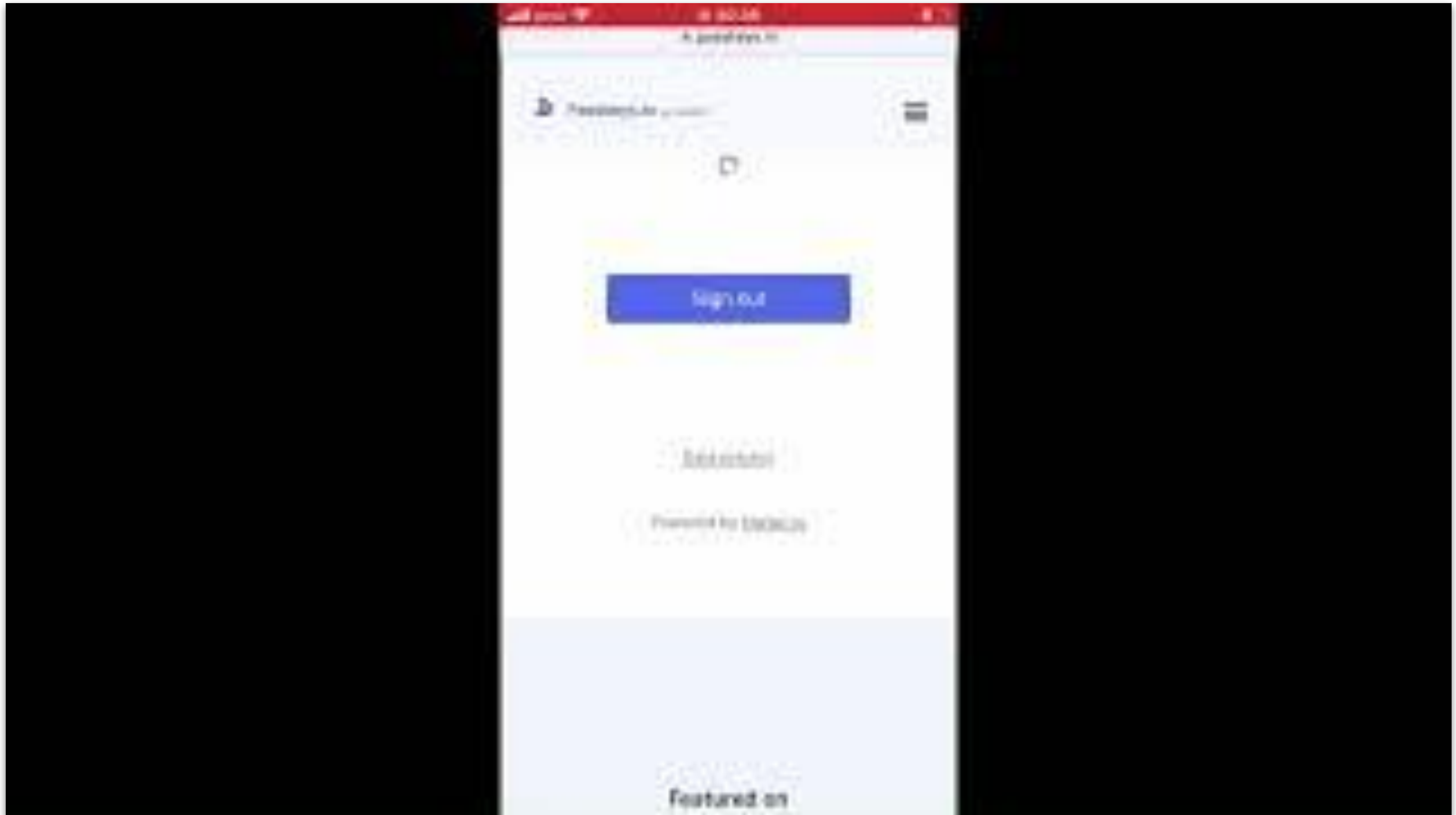






Copyright© Medley, Inc. ALL RIGHTS RESERVED.

デモンストレーションを実施します



Copyright© Medley, Inc. ALL RIGHTS RESERVED.

デモ

MEDLEY

https://www.youtube.com/watch?v=_QwOXyoetD8

Copyright© Medley, Inc. ALL RIGHTS RESERVED.

従来の認証方法との比較－1

従来の認証手段とパスキーでは、リスクシナリオ（守る場所/対象、責任の主体、影響など）が異なる

	パスワード／ワンタイムパスワード	パスキー
資格情報の機密性	パスワードの再利用	チャレンジを生成
	フィッシング被害	RPサーバーの識別子に対する署名検証
	サーバー側での資格情報の侵害	サーバーに秘密情報を送らない
	パスワード保存領域の侵害	パスキープロバイダーの侵害

authenticate2023
Dean H.Saxe [The New Passkey Threat Model]をもとに加工

Copyright© Medley, Inc. ALL RIGHTS RESERVED.

従来の認証方法との比較－２

可用性の観点でもパスキープロバイダーのリスク評価がポイント
※パスキープロバイダーの情報はAAGUIDから取得可能
https://github.com/passkeydeveloper/passkey-authenticator-aaguids/blob/main/aaguid.json

	パスワード／ワンタイムパスワード	パスキー
資格情報の可用性	パスワード忘れ	N/A
	パスワード保存領域からの削除	認証器やパスキープロバイダーからの削除
	EメールやSMS接続不可	N/A
	パスワード保存領域接続不可	Device bound passkeys デバイスの紛失や故障
		Synced passkeys パスキープロダイバー接続不可 アカウントリカバリ失敗 パスキープロバイダーアカウントの削除 パスキープロバイダーのサービス停止

authenticate2023
Dean H.Saxe [The New Passkey Threat Model]をもとに加工

Copyright© Medley, Inc. ALL RIGHTS RESERVED.

従来の認証方法との比較－３

AirDropによるパスキーの共有など、パスキーの利便性の正しい活用とリスク評価もポイント

	パスワード／ワンタイムパスワード	パスキー
資格情報の復旧	RPでのアカウントリカバリ	Device bound passkeys 事前登録した別端末からのリカバリ RPによるアカウントリカバリ
		Synced passkeys パスキープロダイバーからのリカバリ RPでのアカウントリカバリ
資格情報の共有	口頭での共有	N/A
	値の共有	Apple Airdrop

authenticate2023
Dean H.Saxe [The New Passkey Threat Model]をもとに加工

Copyright© Medley, Inc. ALL RIGHTS RESERVED.

プラットフォームごとの対応状況

MEDLEY

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 Planned: Edge ❌ Firefox	Planned	✅ Safari Chrome Edge Firefox	✅ Safari Chrome [2] Planned: Edge ❌ Firefox	❌ Not Supported	✅ Chrome [3] Planned: Edge Firefox
Cross-Device Authentication ***Authenticator***	✅ v9+	❌ Not Supported	✅ v16+	❌ Not Supported	❌ Not Supported	❌ Not Supported
Cross-Device Authentication ***Client***	Planned	✅ v108+	✅ v16+	✅ v13+	✅ Chrome Edge	✅ v23H2+
Third-Party Passkey Providers	✅ v14+	❌ Not Supported	✅ v17+	✅ v14+	❌ Not Supported	Planned

Advanced Capabilities

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	✅
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/（2023/10/20時点）

Copyright© Medley, Inc. ALL RIGHTS RESERVED.

1 認証手段の歴史と課題

2 Synced Passkeysの登場と関連技術

3 Rubyでのパスキー実装

Copyright© Medley, Inc. ALL RIGHTS RESERVED.

MEDLEY

パスキーは、WebAuthn API（Web Authentication API）を利用して実装できる。

WebAuthnの仕様はLevel3（ワーキングドラフト）として公開されている。

https://www.w3.org/TR/webauthn-3/

Rubyのライブラリが複数公開されている

※FIDOアライアンスの試験に合格し認定されているものの利用を推奨

- webauthn-ruby (Cedarcode)
- devise-passkeys (Ruby Passkeys, wrapper around webauthn-ruby)
- warden-webauthn (Ruby Passkeys, wrapper around webauthn-ruby)

当該ライブラリを利用したサンプルコード
webauthn-rails-demo-app

※Passkeysには複数のライブラリがある
awesome-webauthn

Copyright© Medley, Inc. ALL RIGHTS RESERVED.

パスキーの実装箇所

Web Authentication API は登録とログインの2つの基本的な機能を持つ。

	メソッド	利用場面	機能
登録	navigator.credentials.create()	● サービスへのユーザー（アカウント）登録時のパスキー登録 ● サービスの既存ユーザー（アカウント）によるパスキー登録	署名と公開鍵情報を送信
認証	navigator.credentials.get()	● 既存サービスのログイン時のパスキー認証 ● ログイン中のユーザー（アカウント）に求めるパスキー認証	署名を送信

Copyright© Medley, Inc. ALL RIGHTS RESERVED.

MEDLEY

- 登録ステップでは、2つのAPIを用意する必要がある

  （API1) ユーザーIDに紐づいたチャレンジレスポンスを発行するAPI
  （API2）認証器が発行した公開鍵を検証しデータベースに保存するAPI

- 登録ステップ概要

  0. ユーザーによるパスキー登録要求
  1. RP（サーバー）によるPublicKeyCredentialCreationOptionsオブジェクト作成（API1）
  2. RP（JS APP）によるnavigator.credentials.create()実行
  3. 認証器による鍵ペア作成とAttestation署名
  4. 認証器によるPublicKeyCredential返却
  5. RP（JS APP）によるAuthenticatorAttestationResponse返却
  6. RP（サーバー）による認証情報検証、登録完了(API2)

Copyright© Medley, Inc. ALL RIGHTS RESERVED.

アプリのGemfileに以下を追記

```
gem 'webauthn'
```

$ bundle　を実行

```
$ bundle
```

もしくは以下を実行

```
$ gem install webauthn
```

Copyright© Medley, Inc. ALL RIGHTS RESERVED.

MEDLEY

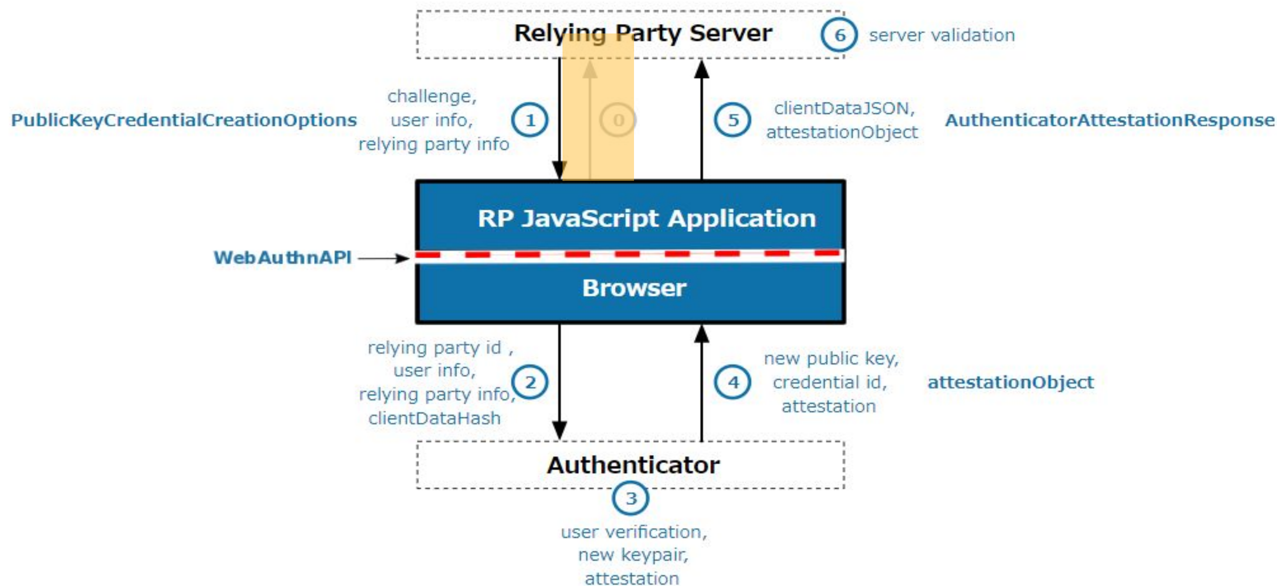
構成情報としてドメイン名やRPの名前を設定する。

```ruby
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."
end
```

Copyright© Medley, Inc. ALL RIGHTS RESERVED.

0. ユーザーによるパスキー登録要求

ユーザーが「パスキーを登録する」などのボタンを表示し、ボタン押下などのイベントを通じてアカウント登録時に必要な情報をサーバーに送信。登録時に必要な情報はサービスごとに異なる。（名前や住所など）

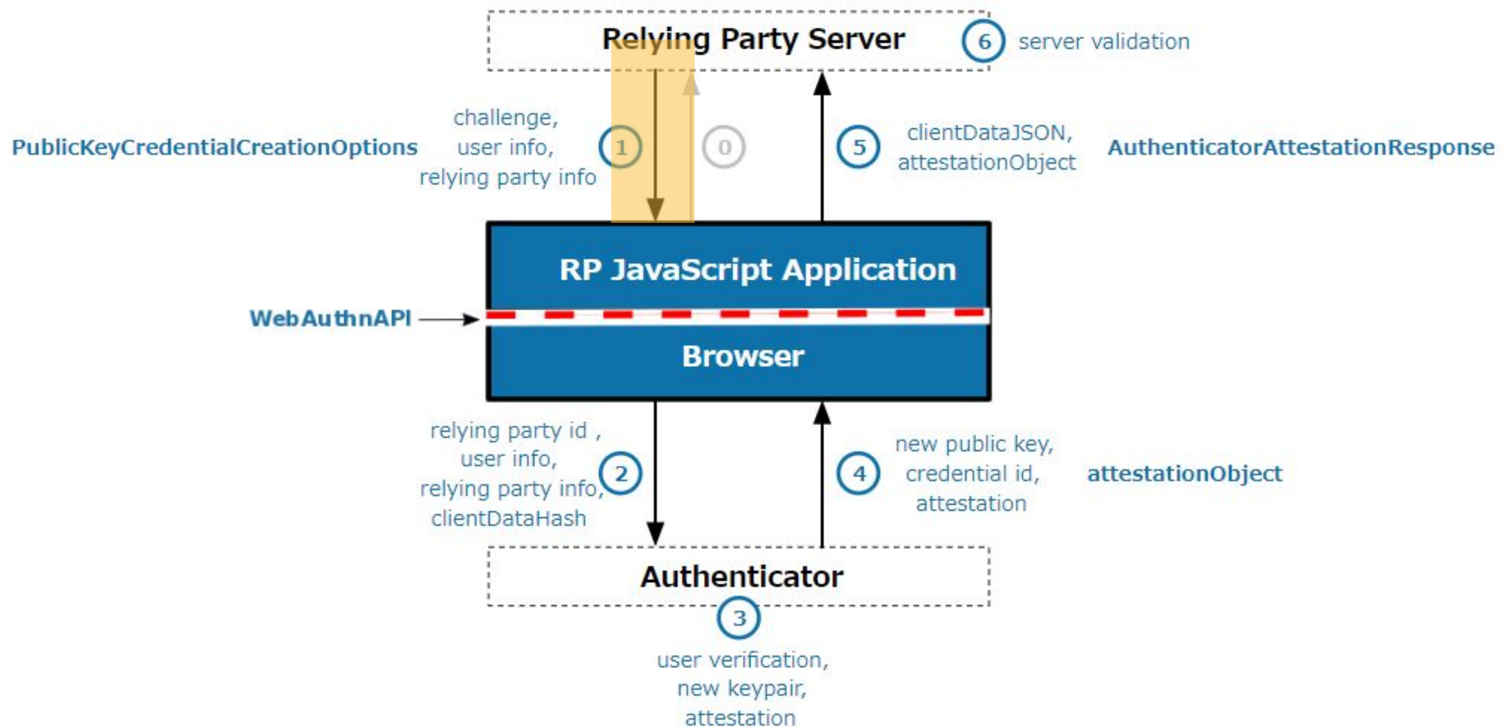



```
<%= form_with url: registration_path, .... do |form| %>
    <%= form.text_field :username %>
    <%= form.submit 'Registration' %>
<% end %>
```

Copyright© Medley, Inc. ALL RIGHTS RESERVED.

MEDLEY

ユーザーからのリクエストを受け取ると、サーバー側ではチャレンジ（乱数）を生成し、RelyingPartyの情報など次のステップに必要な情報と合わせPublicKeyCredentialCreationOptionsオブジェクトを作成する



Copyright© Medley, Inc. ALL RIGHTS RESERVED.

PublicKeyCredentialCreationOptionsオブジェクトの中身

パラメータ	必須	説明
rp	✓	Relying Partyの情報 id:識別子（ドメイン名） name:名前
user	✓	ユーザーの情報 id: RP内での識別子 name: ユーザー名 displayName:表示名
challenge	✓	サーバーで生成した乱数
pubKeyCredParams	✓	鍵の情報 type: Credentialのタイプ alg: Attestationに入る公開鍵の暗号化アルゴリズム
AuthenticatorSelection		認証器が利用可能な機能の制御 AuthenticatorAttachement :プラットフォームへの同期の制御 requireResidentkey : Discoverable Credentialの利用有無 userVerification : ローカル認証の要求度合い
timeout		API呼び出しの待機時間
excludeCredentials		既存のパスキー情報（設定することで、同じパスキーが複数登録されることを防ぐ）

Copyright© Medley, Inc. ALL RIGHTS RESERVED.

1. RP（サーバー）による PublicKeyCredentialCreationOptionsオブジェクト作成

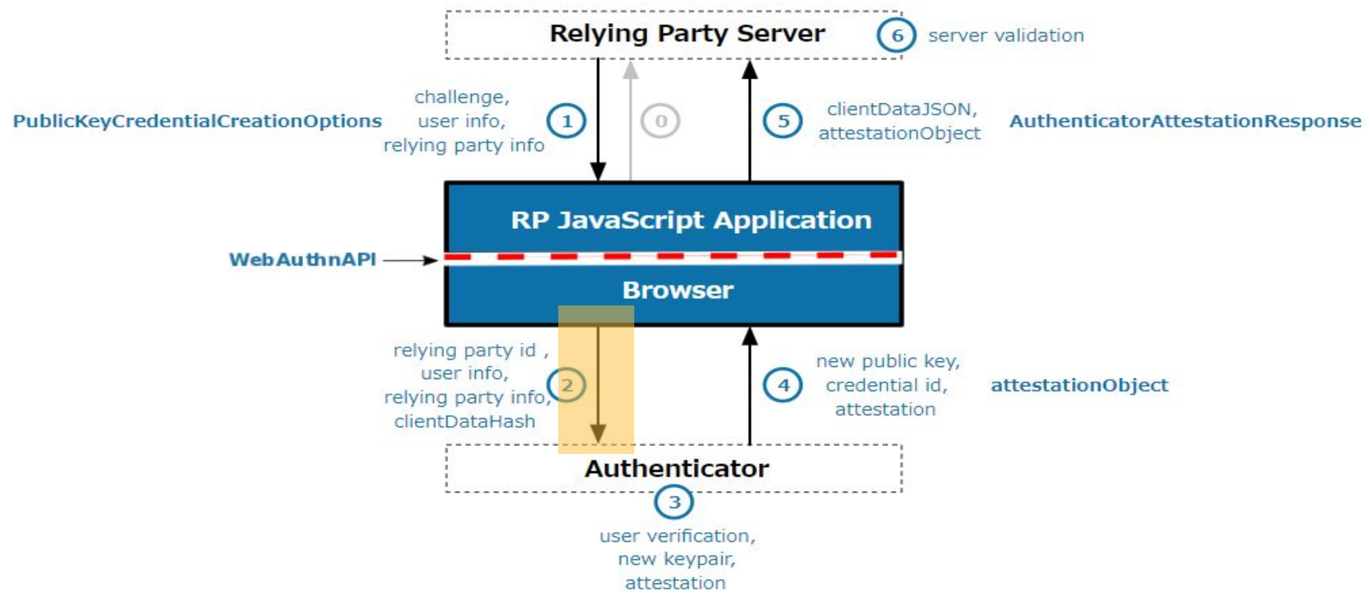
MEDLEY

```ruby
def create
  user = User.new(username: params[:registration][:username])

  create_options = WebAuthn::Credential.options_for_registration(
    user: {
      name: params[:registration][:username],
      id: user.webauthn_id
    },
    authenticator_selection: {
      user_verification: "required"
      require_resident_key:true
      authenticatorAttachment: "platform" }
  )

  if user.valid?
    session[:current_registration] = { challenge: create_options.challenge, user_attributes: user.attributes }

    respond_to do |format|
      format.json { render json: create_options }
    end
  else
    respond_to do |format|
      format.json { render json: { errors: user.errors.full_messages }, status: :unprocessable_entity }
    end
  end
end
```

Copyright© Medley, Inc. ALL RIGHTS RESERVED.

2. RP（JS APP）によるnavigator.credentials.create()実行

PublicKeyCredentialCreationOptionsオブジェクトとPublicKeyを引数として
navigator.credentials.create()メソッドを呼ぶ




```
const cred = await navigator.credentials.create({
  publicKey: options,
});
```

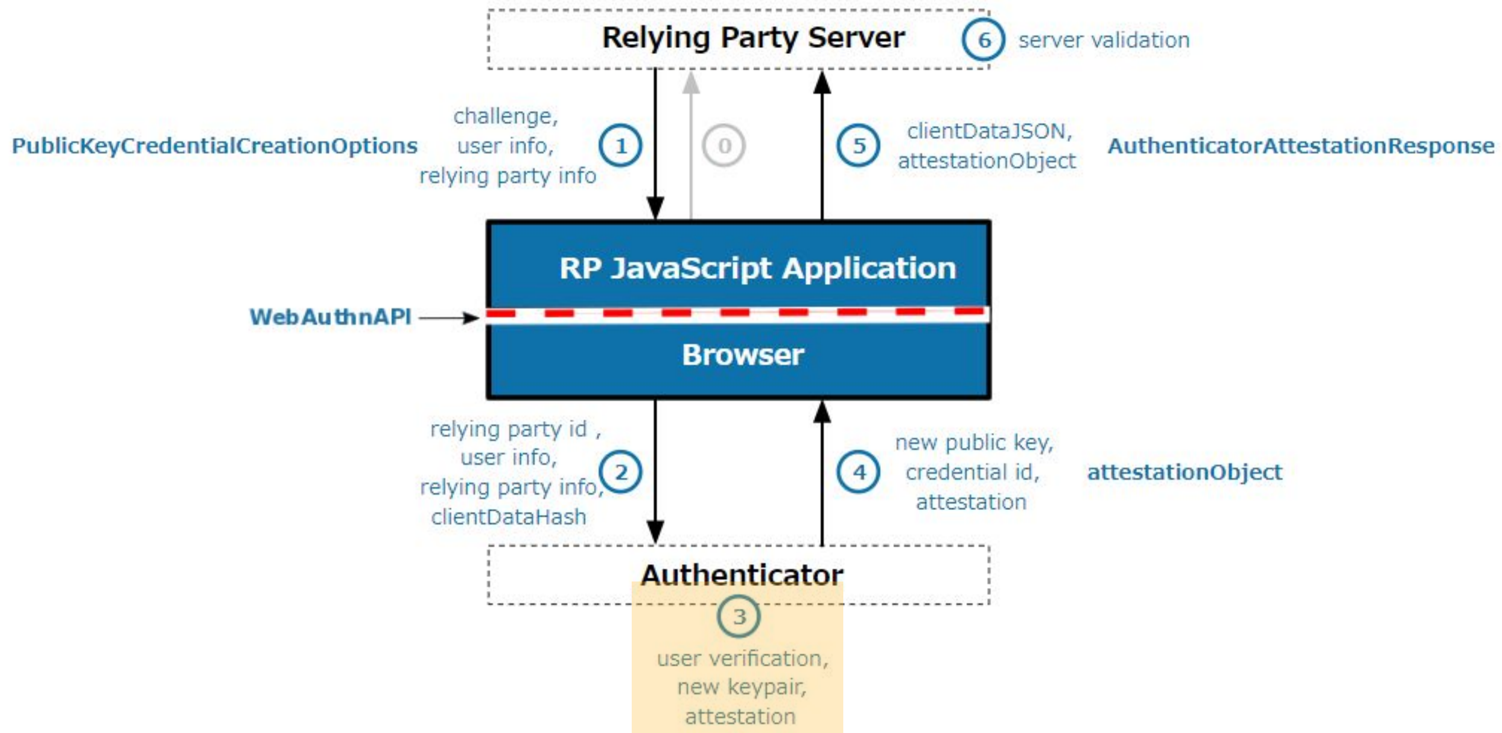
JavaScriptのライブラリも複数存在

例）[webauthn-json]

Copyright© Medley, Inc. ALL RIGHTS RESERVED.

3. 認証器による鍵ペア作成とAttestation署名

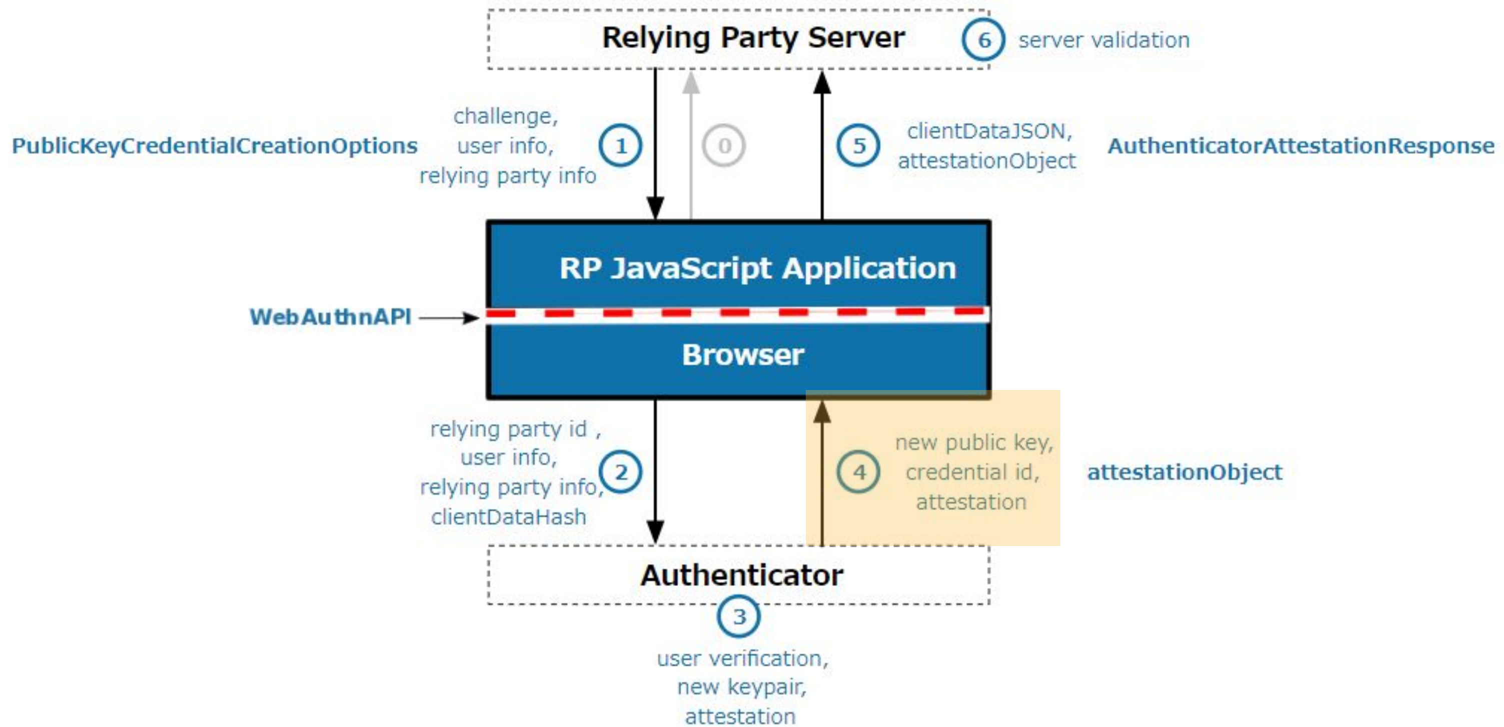
認証器でのローカル認証（本人確認）が成功したら、鍵ペアを作成し、アテステーション用の秘密鍵で署名



Copyright© Medley, Inc. ALL RIGHTS RESERVED.

4. 認証器によるPublicKeyCredential返却

認証器は作成された公開鍵とAttestation等の認証情報をattestationObjectとしてブラウザに返却



Copyright© Medley, Inc. ALL RIGHTS RESERVED.

4. 認証器によるPublicKeyCredential返却

PublicKeyCredentialの中身抜粋

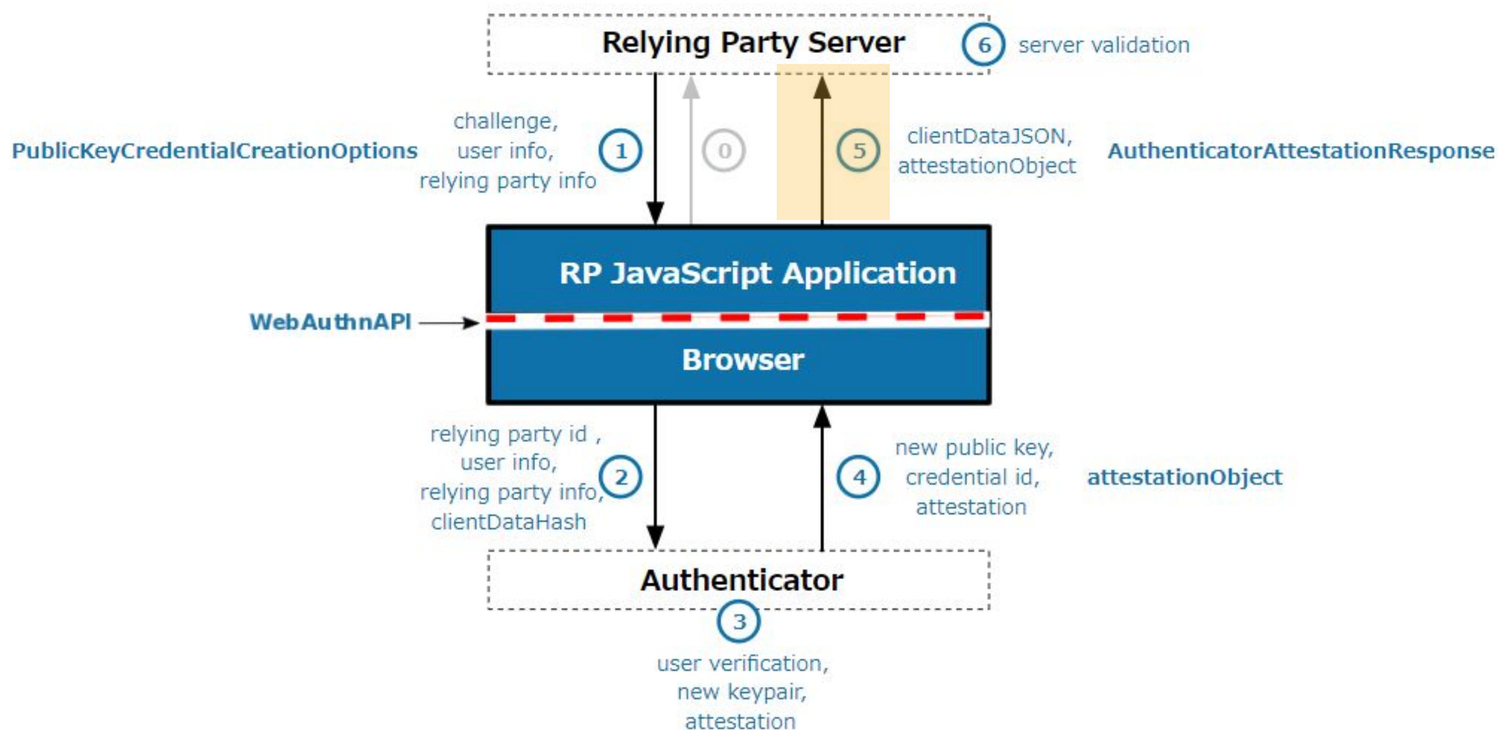
パラメータ	説明
rawId	生成した公開鍵の識別子のバイナリデータ（ArrayBuffer）
id	生成した公開鍵の識別子で、実際には、上記rawIdのBase64urlエンコードしたデータ
response	クレデンシャル生成要求に対する認証器の応答(AuthenticatorAttestationResponse) attestationObject 認証器によるユーザー検証の結果として、COSE形式の認証器のデータとアテステーション情報 clientDataJSON クレデンシャル生成要求の際にWebブラウザーから認証器に伝えた情報（ClientData）のJSON形式の情報
type	PublicKeyCredentialのタイプ。通常public-key

Copyright© Medley, Inc. ALL RIGHTS RESERVED.

5.RP（JS APP）による AuthenticatorAttestationResponse返却

MEDLEY

RPサーバーのAttesstationResponseエンドポイントに認証情報を送信する。

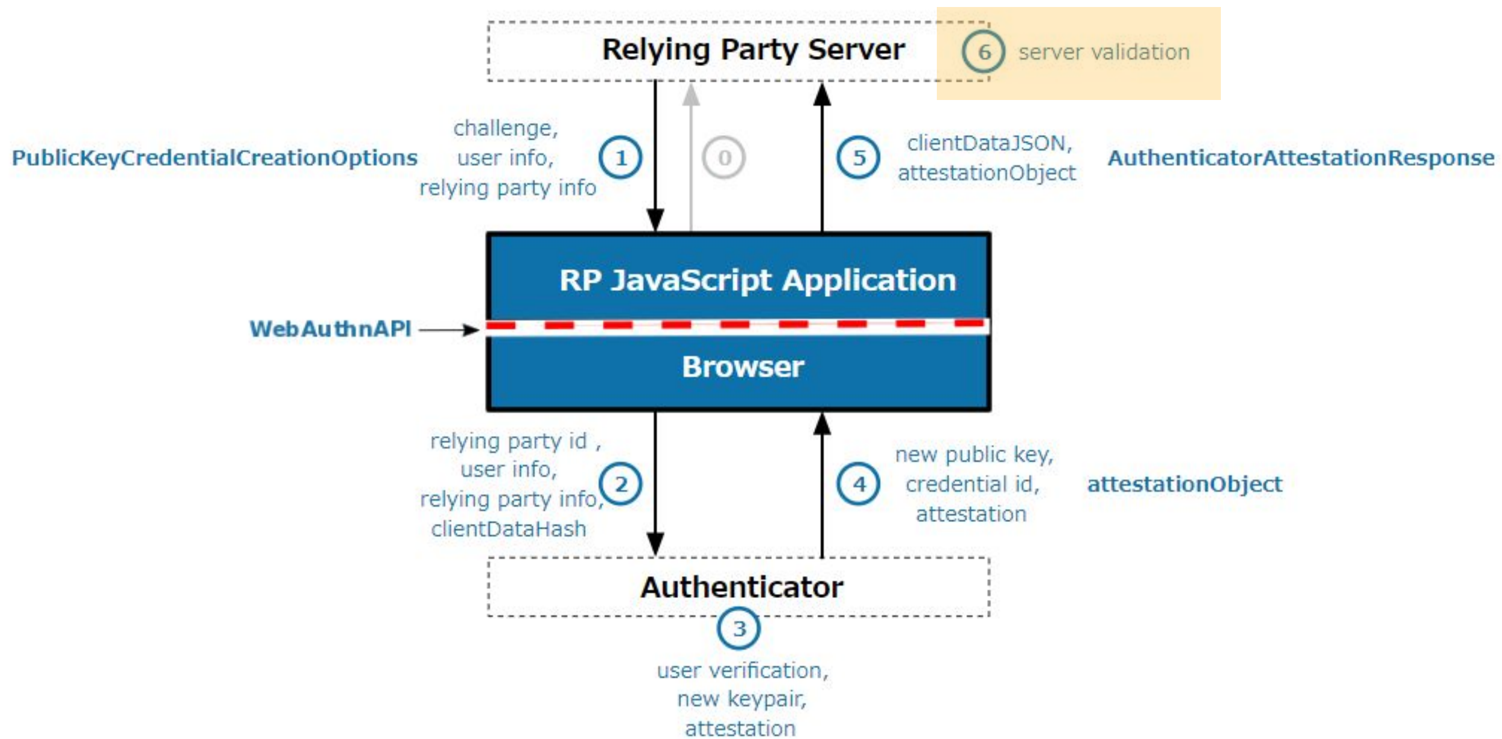


Copyright© Medley, Inc. ALL RIGHTS RESERVED.

6.RP（サーバー）による認証情報検証、登録完了

MEDLEY

RPサーバーは受信した認証情報を検証する。検証成功後、適切な成功レスポンスを返し、認証時に利用するために、公開鍵をDBに保存。ユーザ/認証器の登録が完了。



Copyright© Medley, Inc. ALL RIGHTS RESERVED.

6.RP（サーバー）による認証情報検証、登録完了

MEDLEY

```
def callback
  webauthn_credential = WebAuthn::Credential.from_create(params)

  begin
    webauthn_credential.verify(
      session["current_registration"]["challenge"],
      user_verification: true
    )

    credential = user.credentials.build(
      external_id: Base64.strict_encode64(webauthn_credential.raw_id),
      nickname: params[:credential_nickname],
      public_key: webauthn_credential.public_key,
      sign_count: webauthn_credential.sign_count
    )

    if credential.save
      sign_in(user)

      render json: { status: "ok" }, status: :ok
    else
      render json: "Couldn't register your Security Key", status: :unprocessable_entity
    end
  rescue WebAuthn::Error => e
    render json: "Verification failed: #{e.message}", status: :unprocessable_entity
  ensure
    session.delete("current_registration")
  end
end
```

Copyright© Medley, Inc. ALL RIGHTS RESERVED.

実装内容概要（認証）

- 認証ステップでは、2つのAPIを用意する必要がある

（API1) ユーザーIDに紐づいたチャレンジレスポンスを発行するAPI
（API2）ブラウザの認証器が発行した署名をサーバに保存した公開鍵で検証してセッションを発行する API

- 認証ステップ概要

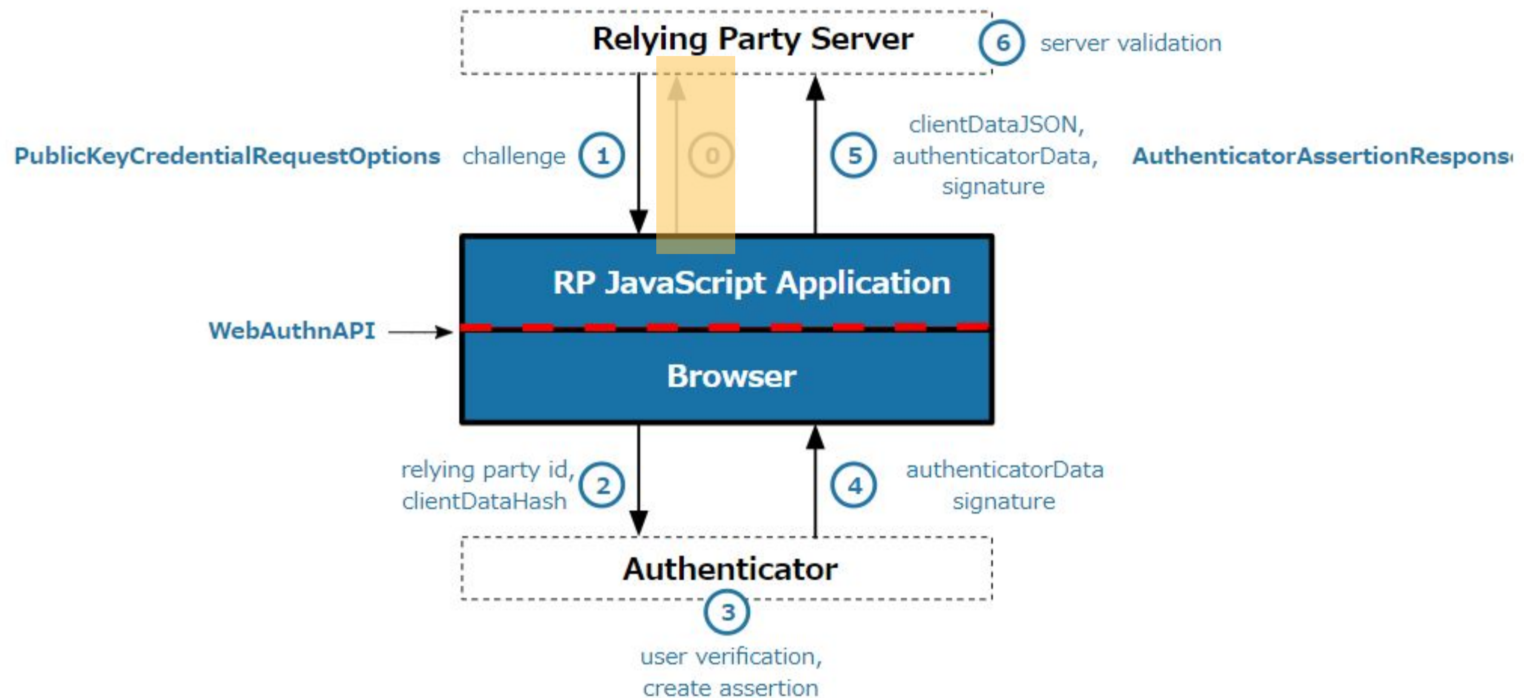
0. ユーザーによるパスキー認証要求

1. RP（サーバー）によるPublicKeyCredentialRequestOptionsオブジェクト作成（API1）
2. RP（JS APP）によるnavigator.credentials.get()実行
3. 認証器によるユーザ確認とAssertion作成
4. 認証器による認証情報返却
5. RP（JS APP）によるAuthenticationAssertionResponse返却
6. RP（サーバー）による署名検証、認証完了(API2)

Copyright© Medley, Inc. ALL RIGHTS RESERVED.

0 ユーザーによるパスキー認証要求

登録と同様に認証エンドポイントへのリクエスト仕様もWebAuthnの仕様の範囲外。通常ユーザIDなどユーザを特定できるユニークな識別子を利用することが多い




```
<%= form_with url: session_path, .... do |form| %>
    <%= form.text_field :username %>
    <%= form.submit 'Login' %>
<% end %>
```

Copyright© Medley, Inc. ALL RIGHTS RESERVED.

Autofillを使いたいときにはログインページのinputフィールドにautocomplete属性を指定する

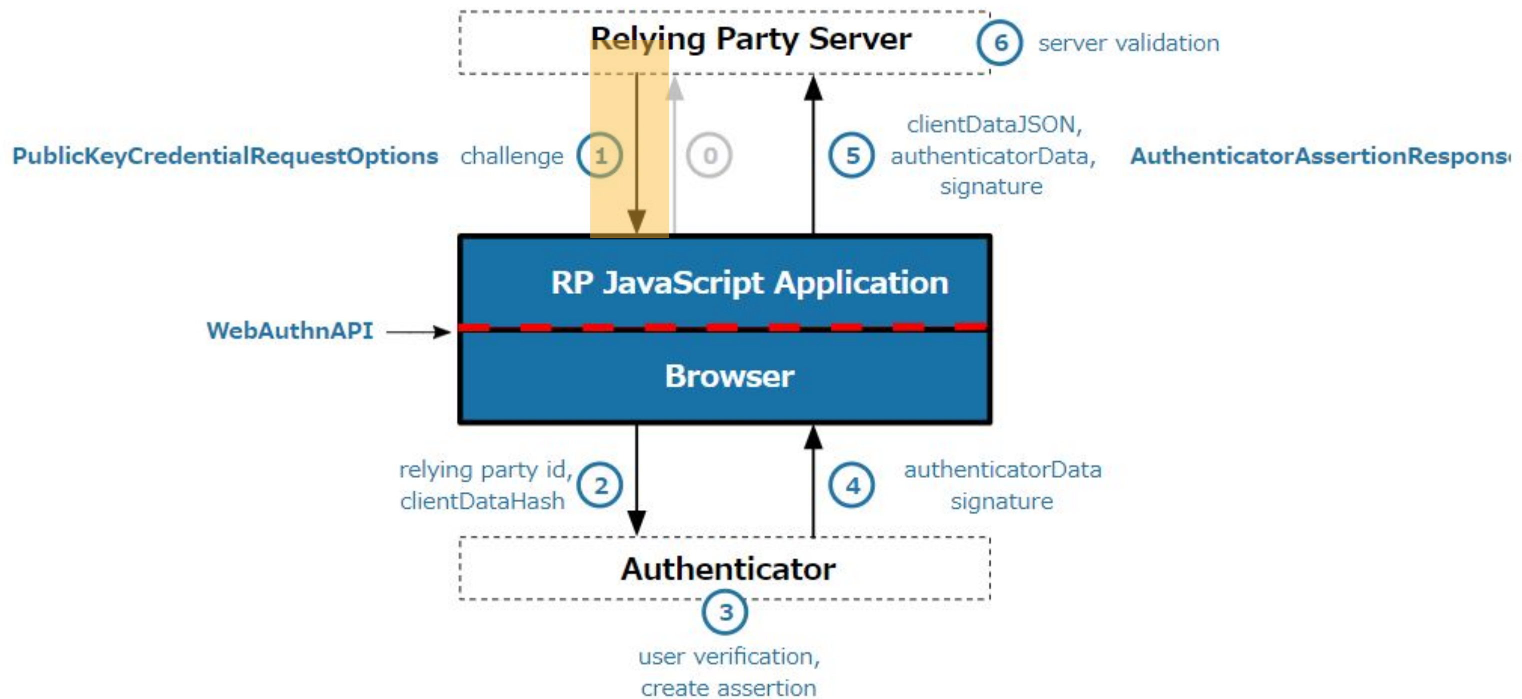
```
<input
    required
    type="email"
    name="email"
    autoComplete="username webauthn"
/>
```

Copyright© Medley, Inc. ALL RIGHTS RESERVED.

1. RP（サーバー）によるPublicKeyCredentialRequestOptionsオブジェクト作成

MEDLEY

ユーザーからのリクエストを受け取ると、サーバー側ではチャレンジ（乱数）を生成する。また、DBに保存されている認証器が作成した公開鍵の情報をallowCredentialsに格納しPublicKeyCredentialRequestOptionsオブジェクトを作成する



Copyright© Medley, Inc. ALL RIGHTS RESERVED.

1．RP（サーバー）による PublicKeyCredentialRequestOptionsオブジェクト作成


```ruby
def create
  user = User.find_by(username: session_params[:username])

  if user
    get_options = WebAuthn::Credential.options_for_authentication(
      allow: user.credentials.pluck(:external_id),
      user_verification: "required"
    )

    session[:current_authentication] = { challenge: get_options.challenge, username: session_params[:username] }

    respond_to do |format|
      format.json { render json: get_options }
    end
  else
    respond_to do |format|
      format.json { render json: { errors: ["Username doesn't exist"] }, status: :unprocessable_entity }
    end
  end
end
```

Copyright© Medley, Inc. ALL RIGHTS RESERVED.

MEDLEY

PublicKeyCredentialRequestOptionsオブジェクトの中身の一例

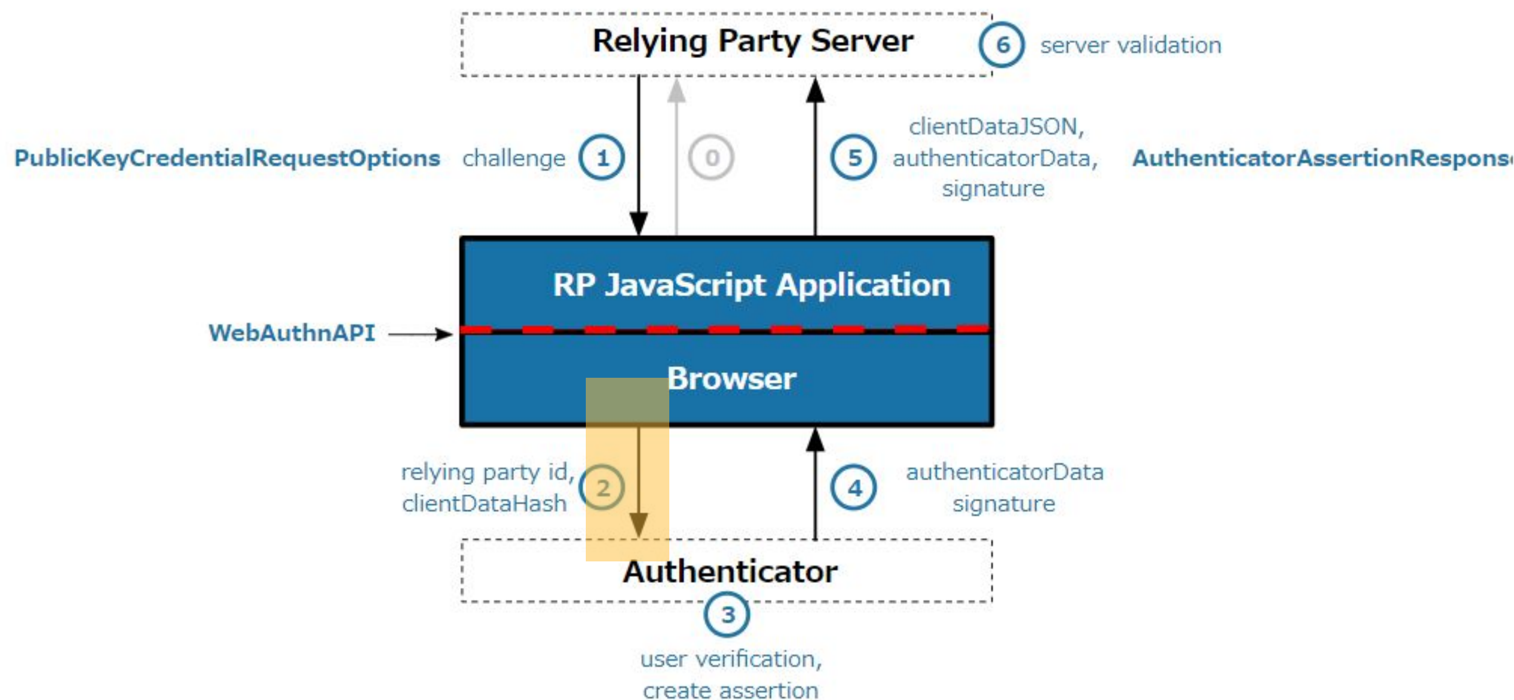
パラメータ	必須	説明
challenge	✓	サーバーで生成した乱数
allowCredentials		type: PublicKeyCredentialのタイプ id: create()のときに取得したCredentialID transports: 認証器との通信方法

Copyright© Medley, Inc. ALL RIGHTS RESERVED.

2. RP（JS APP）によるnavigator.credentials.get() 実行

MEDLEY

RPのレスポンスを元に、navigator.credentials.get()を実行し認証処理を実行。



Copyright© Medley, Inc. ALL RIGHTS RESERVED.

2. RP（JS APP）によるnavigator.credentials.get() 実行

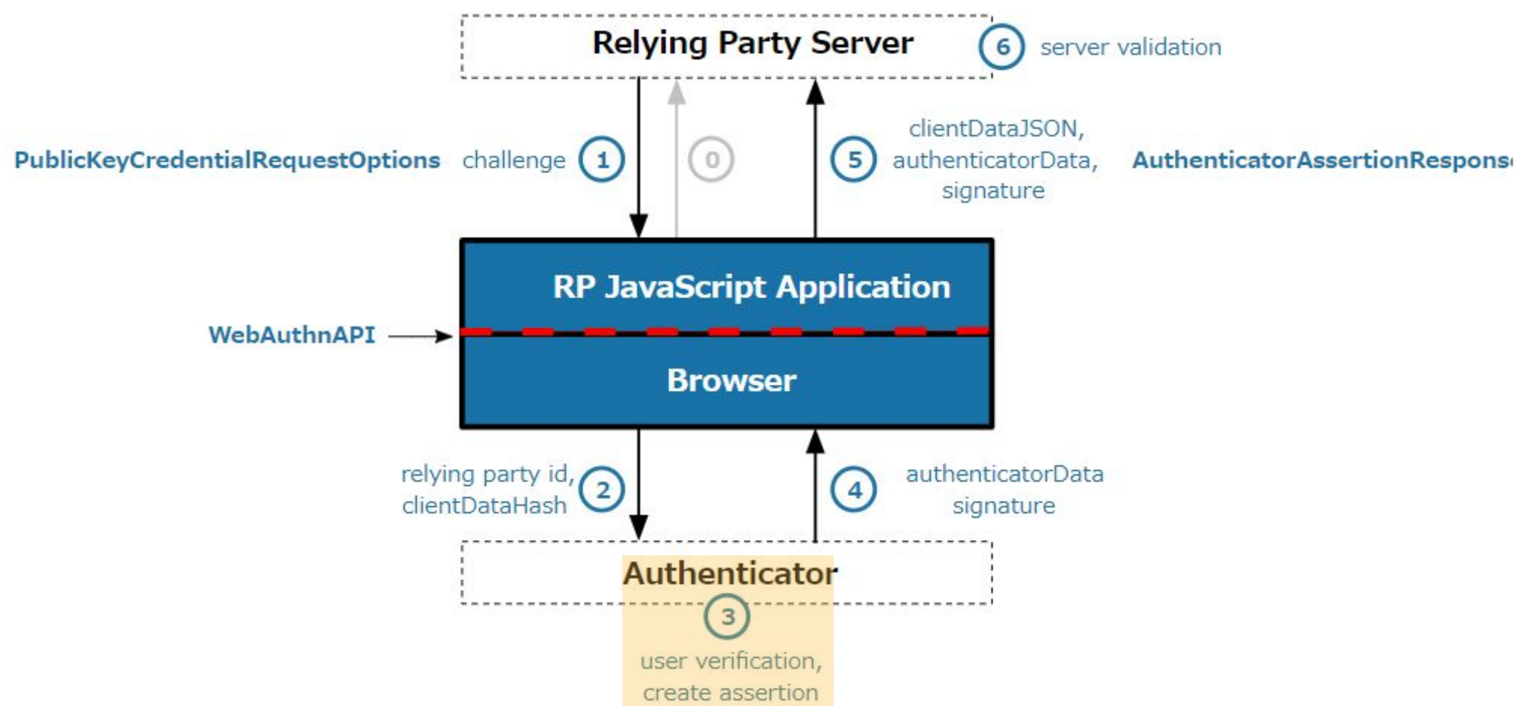
publickKeyをキーとしてPublicKeyCredentialRequestOptionsオブジェクトを渡す。

```
const cred = await navigator.credentials.get({
  publicKey: options,
  // Request a conditional UI
  mediation: conditional ? 'conditional' : 'optional'
});
```

Copyright© Medley, Inc. ALL RIGHTS RESERVED.

3. 認証器によるユーザ確認とAssertion作成

認証器はChallengeやRelyingPartyの情報を元にユーザを確認しAssertionを作成する。

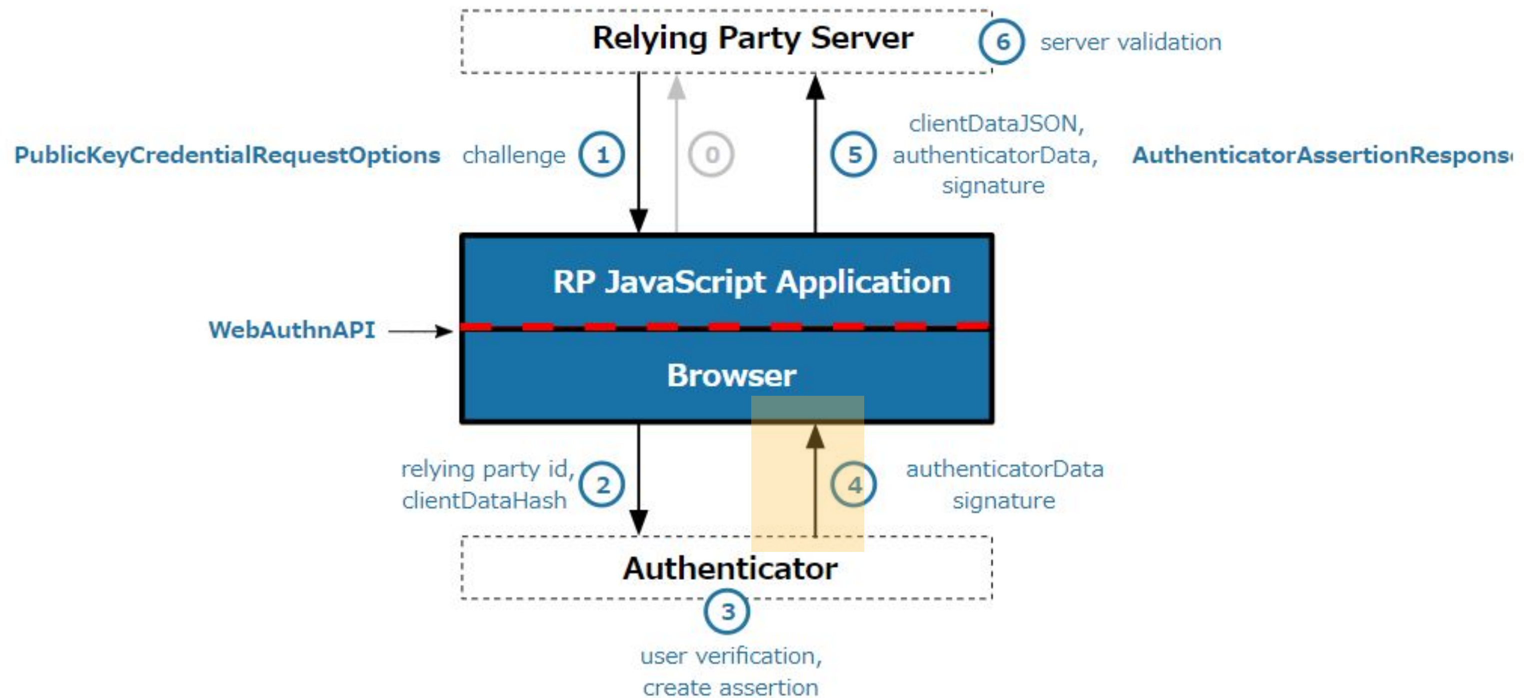


Copyright© Medley, Inc. ALL RIGHTS RESERVED.

4. 認証器によるPublicKeyCredential返却

MEDLEY

認証器が生成したPublicKeyCredential（assertion signature, clientDataHash, AuthenticatorDataなど）をブラウザに返す。



Copyright© Medley, Inc. ALL RIGHTS RESERVED.

4. 認証器によるPublicKeyCredential返却

MEDLEY

PublicKeyCredentialの中身抜粋

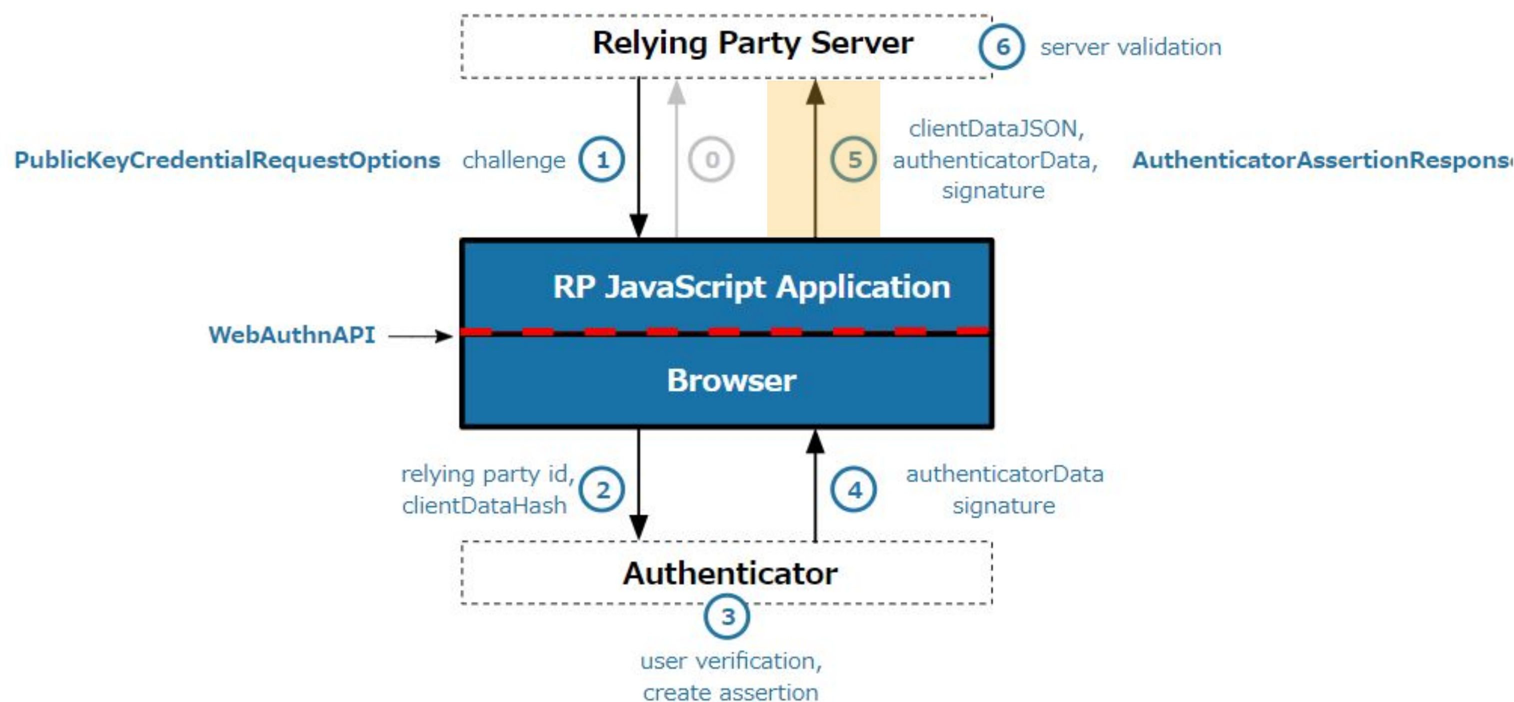
パラメータ	説明
rawId	公開鍵の識別子のバイナリデータ（ArrayBuffer）
id	公開鍵の識別子で、実際には、上記 rawIdのBase64urlエンコードしたデータ
response	クレデンシャル認証要求に対する認証器の応答 (AuthenticatorAssertionResponse) AuthenticatorData 認証器に関するメタデータ clientDataJSON クレデンシャル認証要求の際にWebブラウザーから認証器に伝えた情報（ClientData）のJSON形式の情報 signature 認証器が生成したPublicKeyCredentialに対する署名データ。上記authenticatorDataとClientDataのハッシュ値（ClienthDataHash）を連結したデータに対して、登録されたユーザーの秘密鍵を用いて生成される。
type	PublicKeyCredentialのタイプ。通常public-key

Copyright© Medley, Inc. ALL RIGHTS RESERVED.

5.RP（JS APP）によるAuthenticationAssertionResponse返却

MEDLEY

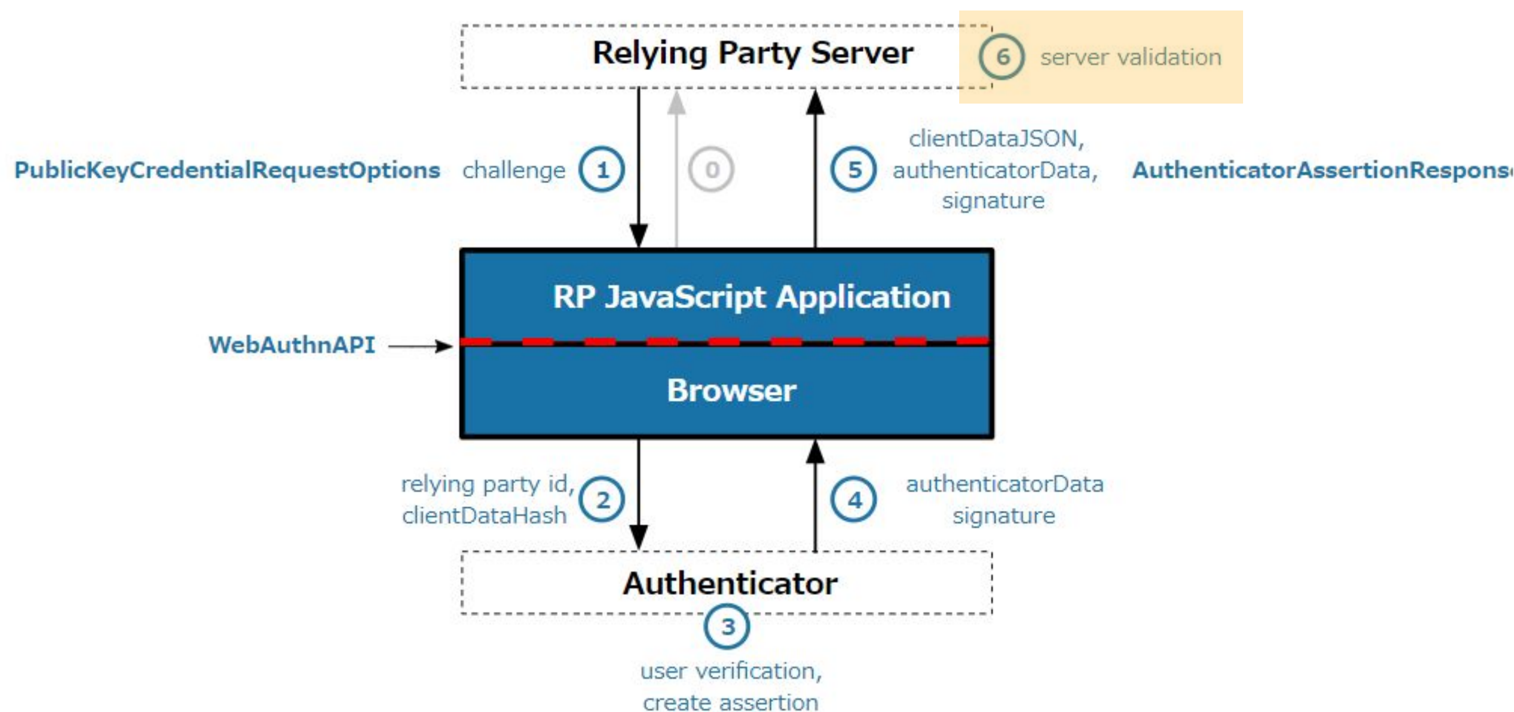
登録時と同様に、認証器から受信したパラメータに加え、rawId、id、type, clienetDataJSON などのパラメータを含め、AssertionResponse エンドポイントに送信する。



Copyright© Medley, Inc. ALL RIGHTS RESERVED.

6.RP（サーバー）による署名検証、認証完了

RPはAssertionを検証し、検証が成功したら認証成功とする。セッションを発行しセッションIDをCookieにセットする等、一般的な認証サーバと同じように処理を継続する。



Copyright© Medley, Inc. ALL RIGHTS RESERVED.

6.RP（サーバー）による署名検証、認証完了

MEDLEY

```ruby
def callback
  user = User.find_by(username: session["current_authentication"]["username"])
  raise "user #{session["current_authentication"]["username"]} never initiated sign up" unless user

  begin
    verified_webauthn_credential, stored_credential = relying_party.verify_authentication(
      params,
      session["current_authentication"]["challenge"],
      user_verification: true,
    ) do |webauthn_credential|
      user.credentials.find_by(external_id: Base64.strict_encode64(webauthn_credential.raw_id))
    end

    stored_credential.update!(sign_count: verified_webauthn_credential.sign_count)
    sign_in(user)

    render json: { status: "ok" }, status: :ok
  rescue WebAuthn::Error => e
    render json: "Verification failed: #{e.message}", status: :unprocessable_entity
  ensure
    session.delete("current_authentication")
  end
end
```

Copyright© Medley, Inc. ALL RIGHTS RESERVED.

- セキュアでユーザビリティも向上できるパスキーを活用できる環境が以前よりさらに整ってきている（Rubyのライブラリやサンプルコードもあり）
- Synced Passkeyならではの注意点や継続議論されている課題にも注目して実装する
  - パスキープロバイダーの安全性
  - Device boundであることの確認
- 既存の認証UXからのスムーズな移行にも留意する
  - UXガイドラインなども公開されている

Copyright© Medley, Inc. ALL RIGHTS RESERVED.