第 3 章

FIDO と OpenID Connecct の関係性について

こんにちは@super_reader です。普段は認証や ID 連携のことをやっており、その中でも主に FIDO(特に WebAuthn) を扱っている Web エンジニアです。

今回 FIDO に関してなにかネタはないかと考えたときに、自分が取り組んでいる認証と ID 連携の仕組みについて関連付けたものを書いたら、読んでくださる方の役にたつのでは思いました。

そこで、今回は「FIDO と OpenID Connect の関係性」について書かせていただきました。私自身、ID 連携に関して知識をかじった後に FIDO などの認証の世界を学んでいった経緯もあり、それぞれの仕様について触れてきた立場から、簡単にはなりますがそれぞれの仕様がどのように関連しているのかを解説したいと思います。

今回解説したい題材はこちらになります。

1. OpenID Connect にとっての FIDO とは？ 2. FIDO と OpenID Connect での Relying Party について 3. FIDO と Self-issued について

それではさっそく FIDO と OpenID Connect の関係を見てみましょう。

3.1 OpenID Connect にとっての FIDO とは？

まずは OpenID Connect について簡単に説明したいと思います。
OpenID Connect は ID 連携の仕様の一つです。ID 連携とは連携先のサービス (Relying Party) に対して、ID やパスワードを渡すことなく、ユーザーの認証情報を提供して、OpenId Provider に保存されているユーザーデータ（属性情報）にアクセスすることが可能になる仕組みのことを指します。ID 連携の仕様の一つである OpenID Connect は現在

さまざまなサービスで使われている仕様です。また、OpenId Provider は認証と認可の機能、そして、サービスがほしい属性情報を持っているサービスになります。実際の企業でいいますと Google や Yahoo! JAPAN、LINE などの ID Provider を指します。

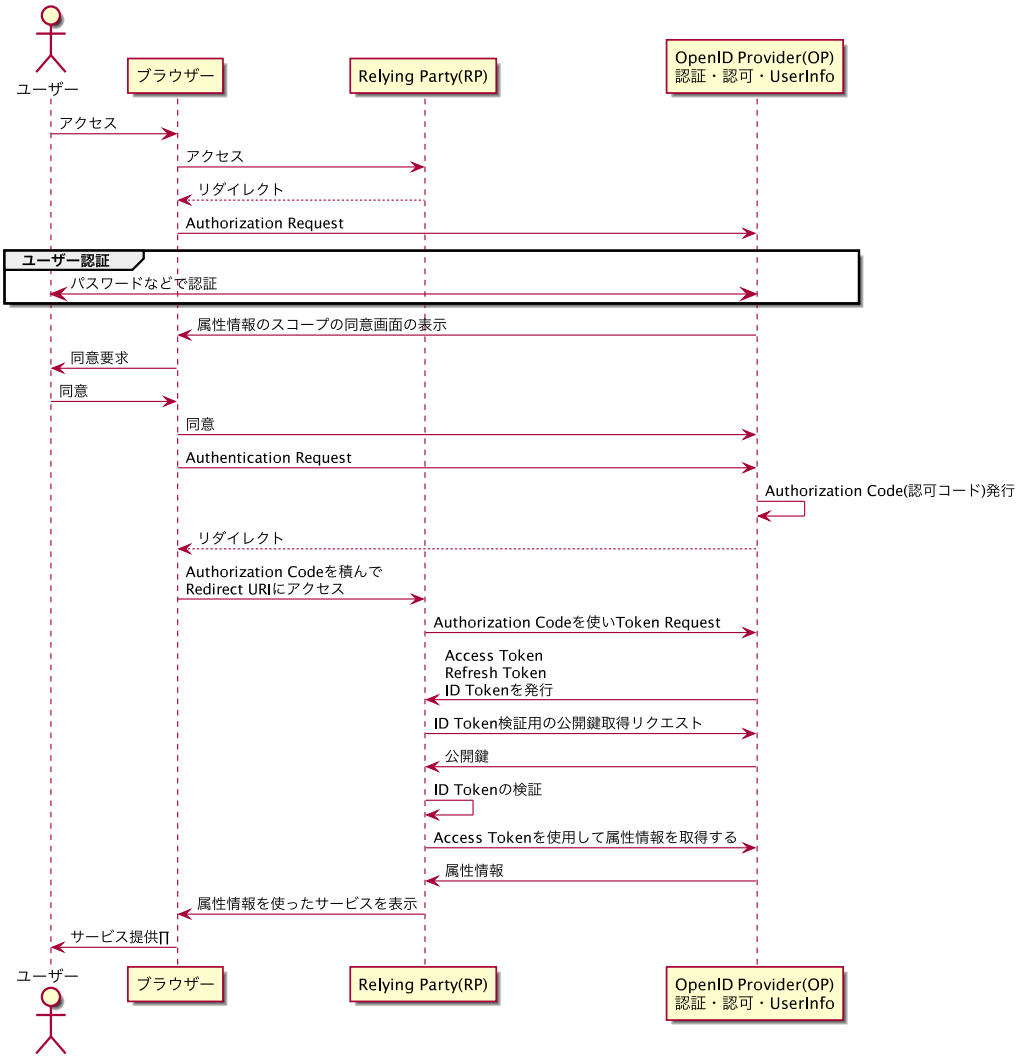
簡単に OpenID Connect に関して説明をさせていただきましたが、細かな仕様のここでの説明は割愛させていただきます。

その代わりに OpenID Connect の理解に役立ちそうなサイトを記載させていただきます。

- 一番分かりやすい OpenID Connect の説明
- OAuth2.0 & OpenID Connect 基礎
- OpenID Connnect 入門

3.1.1 OpenID Connect と FIDO のフローを確認してみよう

さて、本題に入りたいと思います。OpenID Connect と FIDO の関係を考えていく上で、まずは OpenID Connect がどのような流れで処理をされているのかを見てみましょう。図 4.1 は OpenID Connect のフローの一つである Authorization Code フローを簡単に書いたものになります。



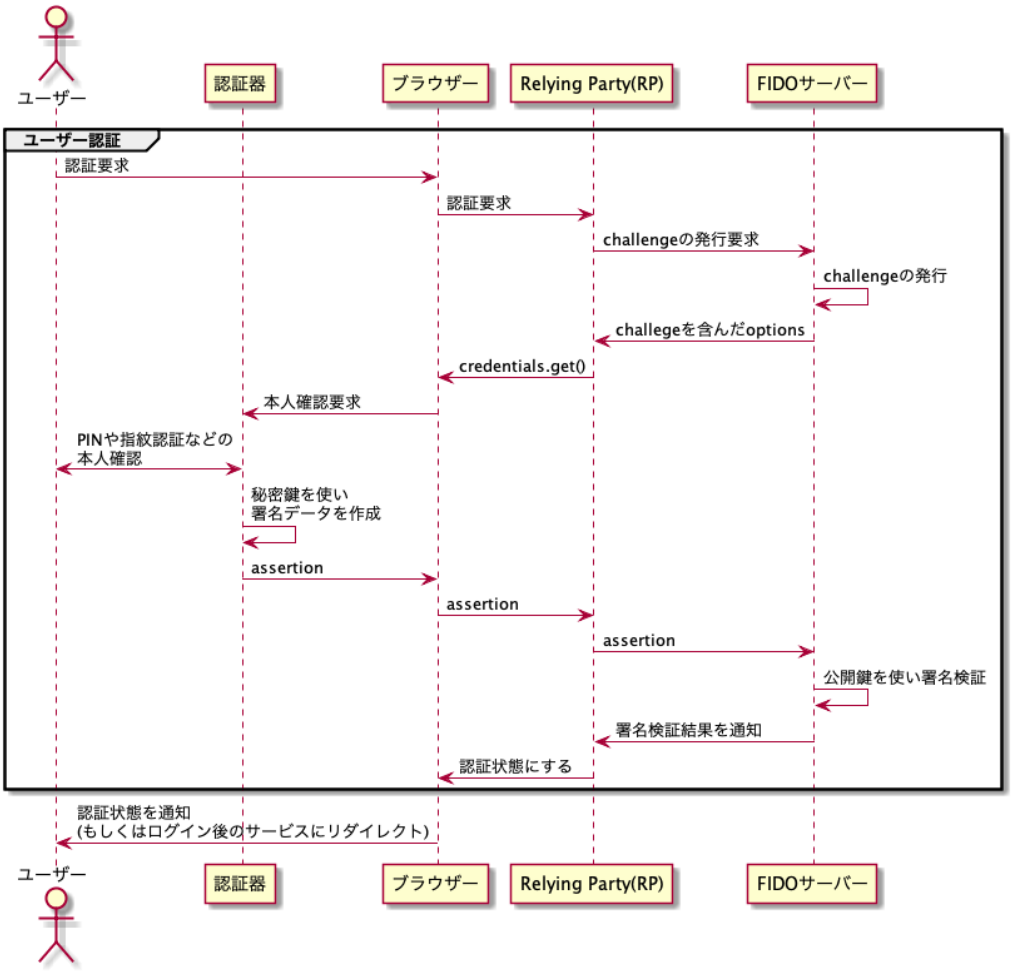
▲図 3.1　Authorization Code フロー

図 4.1 は ID 連携を行いたいサービス (Relying Party) が、OpenId Provider の発行する認可用の Access Token を使い、属性情報を取得するためのフローになります。
このフローの中でも書かれていますが Relying Party に認可コードを渡すための「ユーザー認証」を行っています。OpenID Connect の使用説明の中に OPTION ですが認証方法 (Authentication Methods References) について以下のような記述があります。

認証時に用いられた認証方式を示す識別子文字列の JSON 配列. 例として, パスワードと OTP 認証が両方行われたことを示すといったケースが考えられる.**amr**

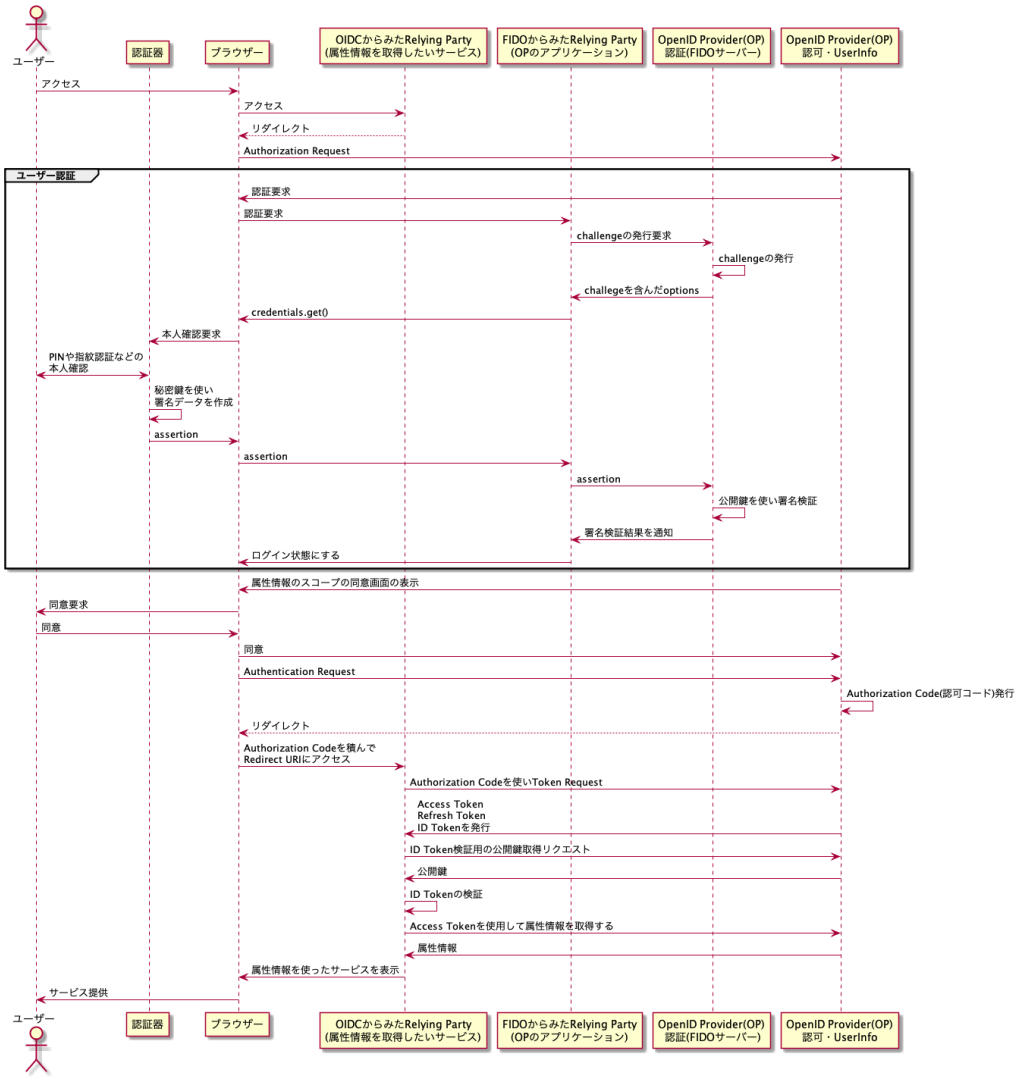
Claim にどのような値を用いるかは本仕様の定めるところではない. この値の意味するところはコンテキストによって異なる可能性があるため, この Claim を利用する場合は, 関係者間で値の意味するところについて合意しておくこと.amr は大文字小文字を区別する文字列である.

このように、OpenID Connect の仕様の中でも認証方法については仕様で細かく言及されていません。
これは言い換えると、認証方法に関しては OpenId Provider の実装方法に委ねられているということが言えます。
さて、ここで FIDO(WebAuthn) の認証フローを見てみましょう。



▲図 3.2　WebAuthn の認証フロー

わざとらしく書いてますが、図 4.2 を見てみると FIDO の仕様は認証だけで閉じています。
ここで注目してほしいことは、FIDO は認証の仕様ということです。
OpenID Connect の Authorization Code フローでも書かれていた「ユーザー認証」の部分に FIDO をそっくりそのまま入れ込むことができます。これは、OP が FIDO 対応の認証手段を持っていた場合に OpenID Connect のフローの中に FIDO を組み込むことが可能ということを示しています。OpenID Connect の Authorization Code フローに FIDO(WebAuthn) のフローを追加したものが図 4.3 です。



▲図 3.3　Authorization Code フローに FIDO のフローを追加した図

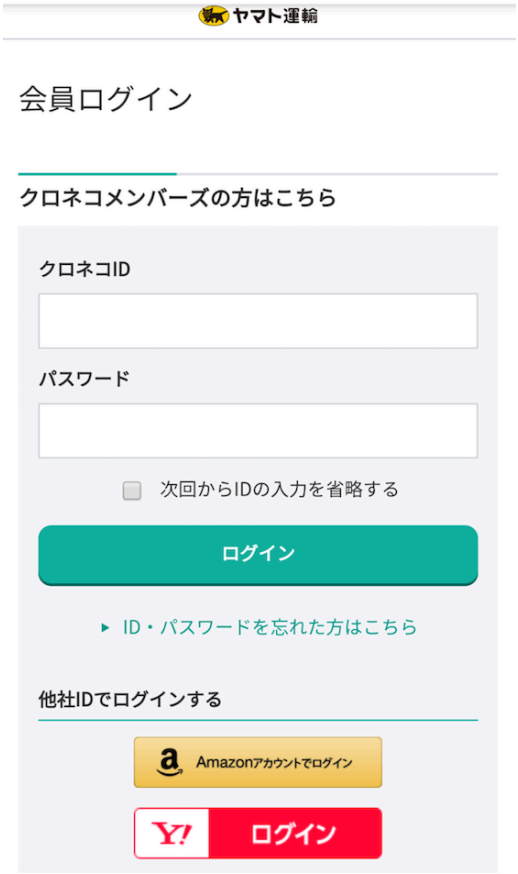
このように、ID 連携の仕様である OpenID Connect と認証の仕組みである FIDO は同居することが可能であり、組み合わせることで ID 連携をしたいサービスでも FIDO を使った認証体験を実現することが可能となります。

3.1.2　実際の具体例を見てみよう

この仕組みが実際に行われているのが Yahoo!ID 連携です。Yahoo!ID 連携の画面フローをもとにどのような処理が行われているのかをたどっていきましょう。

今回は具体例としてヤマト運輸のクロネコメンバーズの会員登録を進めるまでの過程を見てみましょう。

図 4.4 はクロネコメンバーズのログイン画面です。Yahoo! JAPAN のログインボタンを押すことで ID 連携がスタートします。



▲図 3.4　RP の ID 連携ボタンがある画面

まずは OpenId Provider 側の認証を行う必要があります。図 4.5 は OpenId Provider である Yahoo! JAPAN のログイン画面です。未ログイン状態からログインするために ID

を入力して、「次へ」ボタンを押します。ここで使用するアカウントはすでに WebAuthn での登録フローが完了しているアカウントになります。

▲図 3.5　OpenId Provider のログイン画面

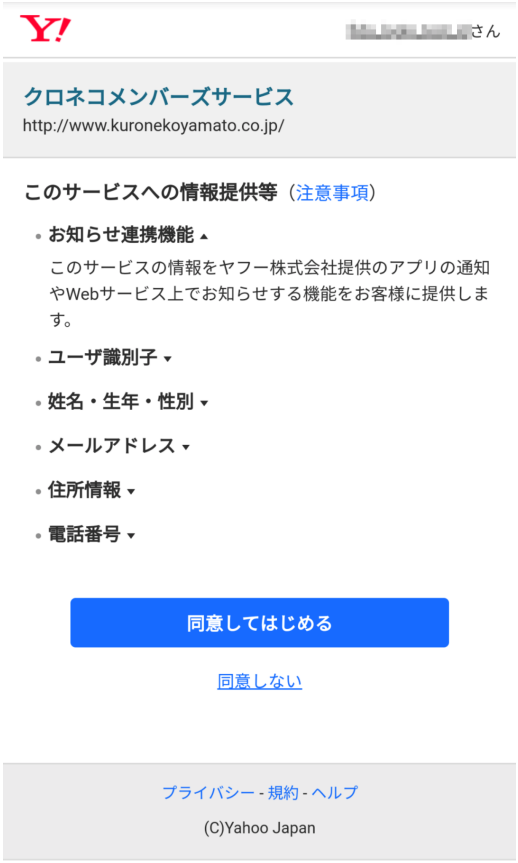
「次へ」ボタンが押された瞬間に裏側では challenge の要求が走り、サーバーからは challenge を含んだ必要な options が返却されます。そして、credentials.get() が叩かれ、ブラウザに対して認証命令を送ります。この認証命令がブラウザに送られ、認証器（今回の場合は Android 内の認証器）が呼び出され、図 4.6 のローカル認証として指紋が要求されています。

▲図 3.6　FIDO の認証で指紋が要求される画面

指紋認証に成功したら、サーバー側に対して assertion が送られ、署名されたデータを OpenId Provider の FIDO サーバーに保存してある公開鍵で検証を行います。

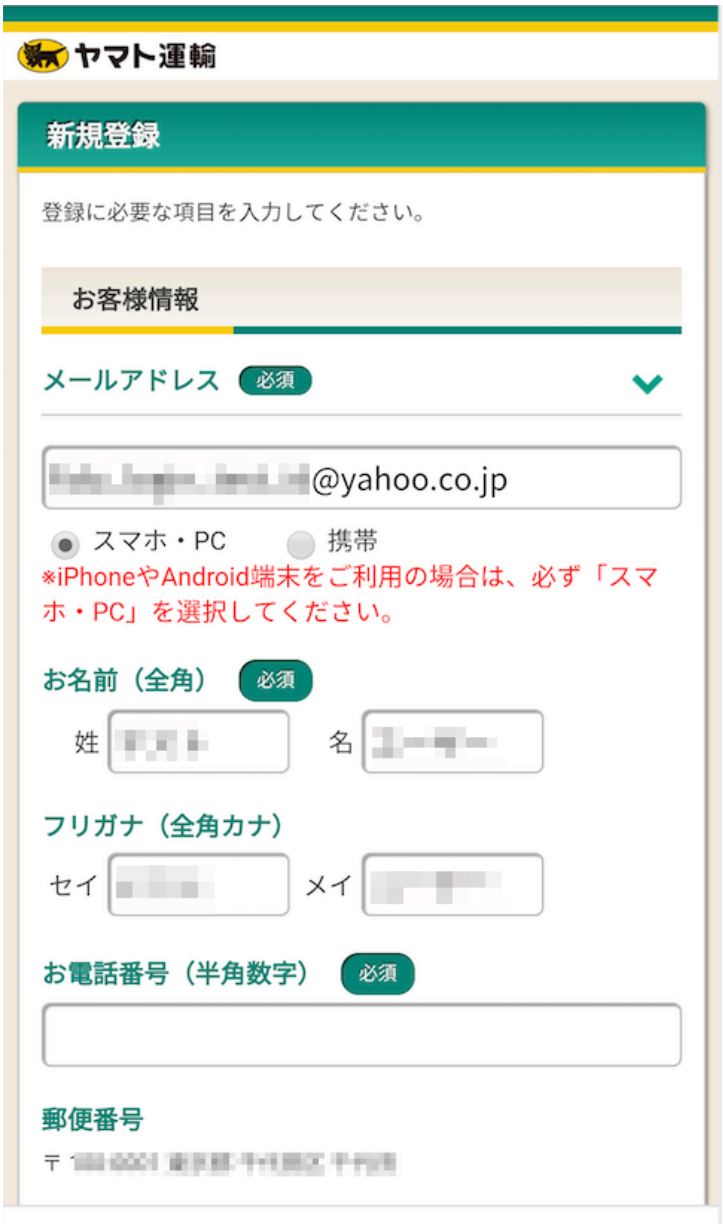
ここで認証のフローは終わり、Token 発行のフローに入ります。

その後、認可コードを発行するために OpenId Provider にアクセスします。このときに RP(ヤマト運輸) に対して、どのような属性情報を渡すのか同意を取る画面が現れ、ユーザーはどこまでの情報を RP に渡していいのかを確認します。



▲図 3.7　RP に対してどのような属性情報を渡すのかの同意画面

属性情報の同意が取れましたら、裏では図 4.3 のように各種 Token が払い出され、Access Token を使用して属性情報を取得します。
取得された属性情報はクロネコメンバーズの登録画面にプリセットされます。モザイクばかりになってしまいましたが、図 4.8 のように登録画面に OpenId Provider(Yahoo! JAPAN) に保存されていた属性情報が埋め込まれた登録画面が現れます。



▲図 3.8　RP の登録時に属性情報がプリセットされている画面

このように ID 連携をすることで事前に必要な情報を取得することができ、ユーザーが登録をしやすいような土台を RP(ヤマト運輸) はユーザーに提供することができます。

以上の具体例からも OpenID Connect と FIDO は共存することが可能であることがわ

かります。

3.2 FIDO と OpenID Connect での Relying Party について

私が FIDO を一番最初に学んだときに一番最初に引っかかった部分がここでした。

突然ですが、先程見てみた Authorization フローに FIDO のフローを追加した図 (図 4.3) をもう一度見てみましょう。

図 4.3 の中には Relying Party(RP) という単語が 2 回出てきます。OpenID Connect と FIDO の両方で登場するこの単語なのですが、２つの仕様で指し示すサービスが違います。

OpenID Connect の中では ID 連携を行いたいサービスを RP と呼び、FIDO では主に ID Provider(OpenID Connect では OpenId Provider となっている部分) のことを RP と呼びます。同じ ID 関連の仕様なのに同じ単語で違う意味合いになっているのはなぜでしょうか？

この RP が二箇所で出てきてしまっている問題を考えるにあたって、RP の元々の意味を考える必要が出てきます。ID 関連の仕様で使われている RP の元々の意味を「認証 (本人検証) を委任しているサービス」と捉えることにより、この疑問は解消されるはずです。（ここではわかりやすく考えるため OpenID Provider = ID Provider と考えてください）

OpenID Connect の場合

> ID 連携を行いたいサービス（RP）が OpenId Provider に対して認証 (本人検証) を委任している

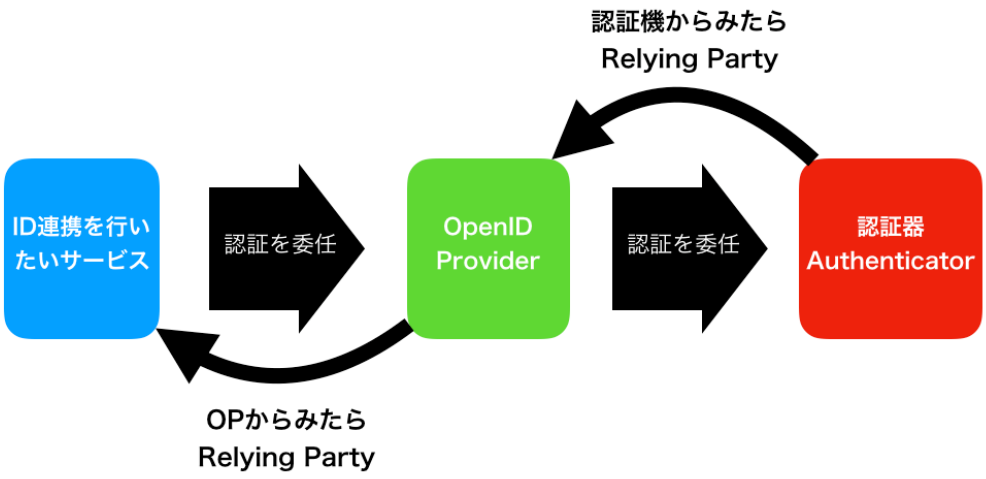
FIDO の場合

> ID Provider(RP) が認証器 (Authenticator) に対して認証 (本人検証) を委任している

という風に解釈できます。

よくあるパスワードでのログインをする場合などにおいて ID Provider が認証 (本人確認) を委任することは基本的にはないのですが、FIDO の文脈では ID Provider が認証（本人確認）を認証器 (Authenticator) に委任しています。

そのため OpenID Connect の枠組みに FIDO を導入しようと思った場合、認証 (本人確認) は図 4.9 のように委任を繰り返します。



▲図 3.9　それぞれの RP の関係

FIDO を学び始めたとき、最初は同じ意味を指している単語だと思い、自分の解釈が間違っていると思い RP という単語で混乱していました。
しかし、RP という言葉の意味をよく考えてみると矛盾なく、同じ単語でサービスが違っているのかも理解することができます。

3.3　FIDO と Self-issued について

OpenID Connect の仕様の中にも認証をどうするかについて書かれた **Self-issued** という仕様があります。
参考:`https://openid.net/specs/openid-connect-core-1_0.html#SelfIssued`
この仕様を簡単に説明すると

1. OpenID Provider の認証機能をローカルデバイスの中に持っていき、認証を行う
2. ユニークな鍵ペアを作成して安全に管理する
3. ローカルデバイスに保存された秘密鍵で署名された Id Token の検証を行うことで認証とする
4. Issuer は https://self-issued.me が使用される

などが挙げられます。
実際に OpenID Connect の枠組みを使い、Self-issued を取り入れている企業としては

リクルートが有名です。

さて、先程簡単に説明させていただいた特徴をどこかで見たことがないでしょうか。そうです。FIDO 認証で使われている仕組みによく似ています。

実際にサービスとして運用しているリクルートの仕組みでも端末側で鍵ペアを作成したり、公開鍵をサーバー上で管理したりなど、大雑把な技術的な構成だけを考えたら FIDO とあまり変わらないことを行っています。

もちろんパラメーターの渡し方などに関しては大きく違います。あくまで、公開鍵暗号方式を用いてログインをするという点に関してはとても良く似ているという意味です。

では、FIDO と Self-issued の違いは何でしょうか？ 私自身もここに関してはわからない部分が多かったので、今回有識者にヒアリングなどをしつつ自分なりに考えてみました。

まず、FIDO と Self-issued を考えるときは、技術的な視点ではなく FIDO や OpenID Connect の枠組みを考える必要があります。先程から何回か出ていますが、OpenID Connect は ID 連携の仕様、FIDO は認証の仕様という感じにそれぞれの立場 (レイヤー) が違います。

Self-issued はあくまで ID 連携から派生した仕様であり、基本的には認可トークンをどのように発行するかが重要で、認証はそのための手段です。なので、Self-issued を使いたい場合も OpenID Connect の枠組みなので本質としては属性情報をやり取りすることが目的になります。そのため、公開鍵暗号を使って認証を行っていますが、その情報自体は属性情報のやり取りの中に内包されて仕様ができています。

一方 FIDO は認証に特化しています。認証におけるエコシステムやフレームワークを FIDO Alliance が標準仕様を考えて導入を推し進めています。そのため、鍵ペアを作成し秘密鍵で署名を行う認証器についてや、サーバー側での Attestation の検証方法に関しても厳密に仕様が決まっています。

なので、FIDO と Self-issued に関して簡単に言ってしまうと

> 手段は一緒だけど、そもそもの目的が違う

だと思われます。

私見になってしまいますが、実際にサービス導入を考えていく上では一長一短あると思います。厳密な認証注目したフレームワークやエコシステムの導入を考えている場合は FIDO を組み込むことを考えますし、OpenID Connect の仕様内の属性情報のやり取りの中で公開鍵暗号方式の認証を行いたい場合は Self-issued を導入することも考えると思います。

3.4 まとめ

FIDO と OpenID Connect の関係性を中心にお話させていただきました。OpenID Connect などを使って ID 連携をしているサービスはたくさんあります。それらのサービスでも現在運用している ID 連携の枠組みを維持しながら FIDO のような新しい認証体験をユーザーに提供することが可能だということが少しでも伝わりましたでしょうか。

FIDO や OpenID Connect のような認証認可の仕様は、より強固でより利便性の高く、さまざまな分野で使ってもらえるような仕様へと日々アップデートしています。特に FIDO の WebAuthn などはサービス導入が始まったばかりなので、いろいろなサービスが使い始めたら今後もどんどん発展していくでしょう。そのときにはきっと FIDO と OpenID Connect の関連性を考えるときがくると思います。その時の手助けに今回の記事が参考になったら幸いです。

3.4.1 参考資料

- Yahoo! JAPAN での生体認証の取り組み（FIDO2 サーバーの仕組みについて）
- Yahoo! ID 連携　Authorization Code フロー
- OpenID Connect Self-Issued OP(IdP) の仕様と応用
- OpenID Connect Core 1.0 incorporating errata set 1
- OpenID Connect　Self-Issued IdP を応用した Single Sign On の実装