

Modern Authentication

FIDO2 Web Authentication (WebAuthn) を学ぶ

栗原 淳

兵庫県立大学 大学院応用情報科学研究科
株式会社ゼタント

May 5, 2020

はじめに

はじめに

この講義では、

- パスワード認証に代わるモダンな認証方式「FIDO」の概要
- FIDO の認証を Web ブラウザ経由で利用する「FIDO2 WebAuthn」の利用

のさわりを学ぶ。

この講義の対象と事前準備

対象:

- 暗号・セキュリティ技術に興味がある初学者
- Web に新しい認証技術を導入したい Web 系のエンジニア

必須ではないが触って楽しむのには必要な事前準備:

- Bash/Zsh, Git が使えるようになっていること
- Node.js, npm, yarn が使えるようになっていること
- Google Chrome 系ブラウザ and/or Firefox が利用可能のこと

パスワード認証から FIDO へ

認証とは

認証

「何らかの手段」で対象の正当性を確認すること。

- メッセージの正当性を確認 ⇒ メッセージ認証
- サービス利用ユーザの正当性を確認 ⇒ ユーザ認証
- etc.

※このスライドで単純に「認証」と呼んだときは、認証対象を「正規ユーザ本人」としたユーザ認証・本人認証を指すこととする。

本人認証の3つの要素

本人認証において、正当性確認のため検証されるものは大きく3要素に分類。

■ 知識

⇒ 本人しか知らない知識を持っていればOK (ex. パスワード)

■ 所有物

⇒ 本人しか持っていない物を提示できればOK (ex. HWキー)

■ 生体

⇒ 本人の体の一部を提示できればOK (ex. 指紋)



本人しか知らない



本人しか持っていない
(複製できない)



本人の体の一部

オンラインサービスでのパスワード認証

- サービスの利用者の識別子 (ID) と対応するパスワードをサービス事業者に登録、サービス利用時に利用者が自分の ID とパスワードを入力する。
- パスワードは個人の記憶にのみ存在するため、**パスワードを知っている人はそのサービスに登録してある本人と同一人物と考えることができる。**

おそらく、誰にとっても最も馴染み深い認証方式！

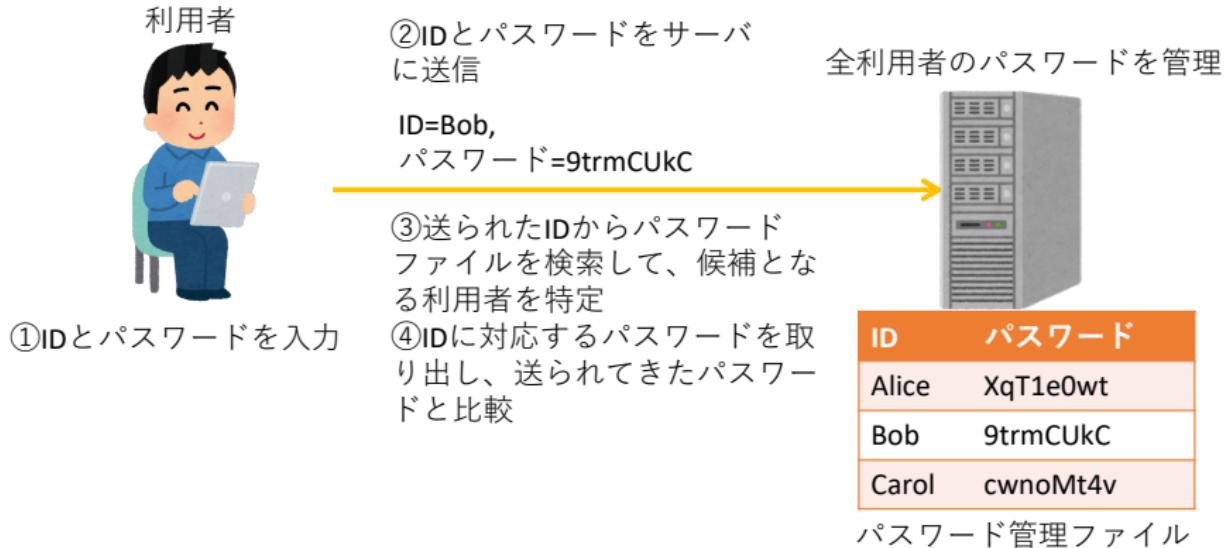


Figure: オンラインでの単純なパスワード認証

オンラインでのパスワード認証の問題

英数字・記号を組み合わせたパスワード:

- 攻撃者にとって比較的予測しやすい¹
- 「強い」 パスワードを使わせるにはユーザ教育が必要²
- 覚えられない
- etc...



予測できず、誰が使っても強力で、確実に認証できる方法が必要
⇒ ハードウェアセキュリティキーを使った認証が人気
⇒ FIDO はそのような手法の標準化された方式

¹しかもオンラインだと予測→認証トライを繰り返せる

²教育なしだと覚え易く「弱い」ものを利用しがち

FIDO (Fast IDentity Online)

業界団体 FIDO Alliance³ の策定する、ハードウェアセキュリティキー+生体認証⁴ と公開鍵暗号方式をベースとしたオンラインでの本人認証技術。

現在は FIDO2 (v2.0) が最新の規格。以降、FIDO2 の内容について触れていく。

厳密には、FIDO2 はパスワードレス認証をサポートしつつも、パスワード+デバイス・生体認証の多要素での認証もサポートする。

³<https://fidoalliance.org>

⁴すなわち、「所有物」と「生体」の二要素を同時に使った認証が可能。

FIDO 認証概略

FIDO 認証の特徴:

- 公開鍵暗号を利用した、オンラインでの認証方式の提供
 - 認証器によるローカルでの本人認証
 - 認証器内部に閉じた署名生成
- ⇒ 秘密鍵・パスワード等の秘密情報は外部に出ない

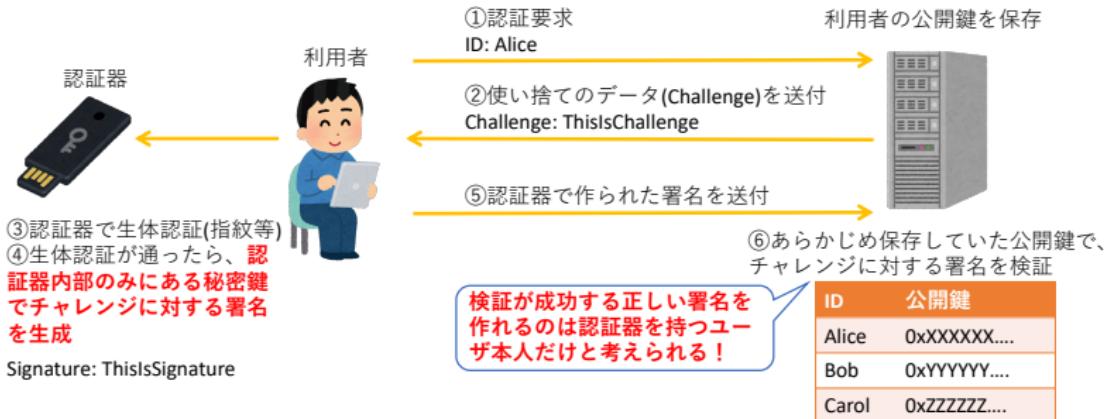
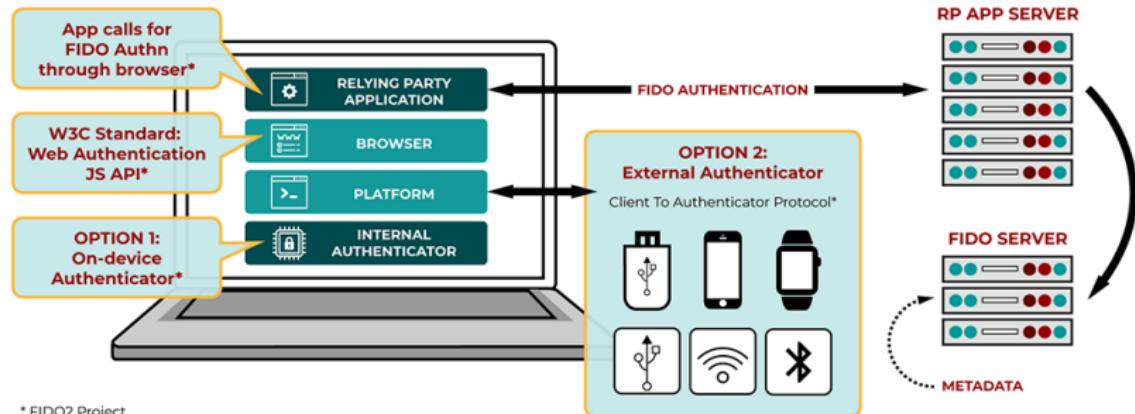


Figure: FIDO 認証概略

FIDO2 の要素

FIDO2 は、WebAuthn (Web Authentication)⁵と、CTAP
(Client-to-Authenticator Protocol)⁶の 2つの要素で構成される。



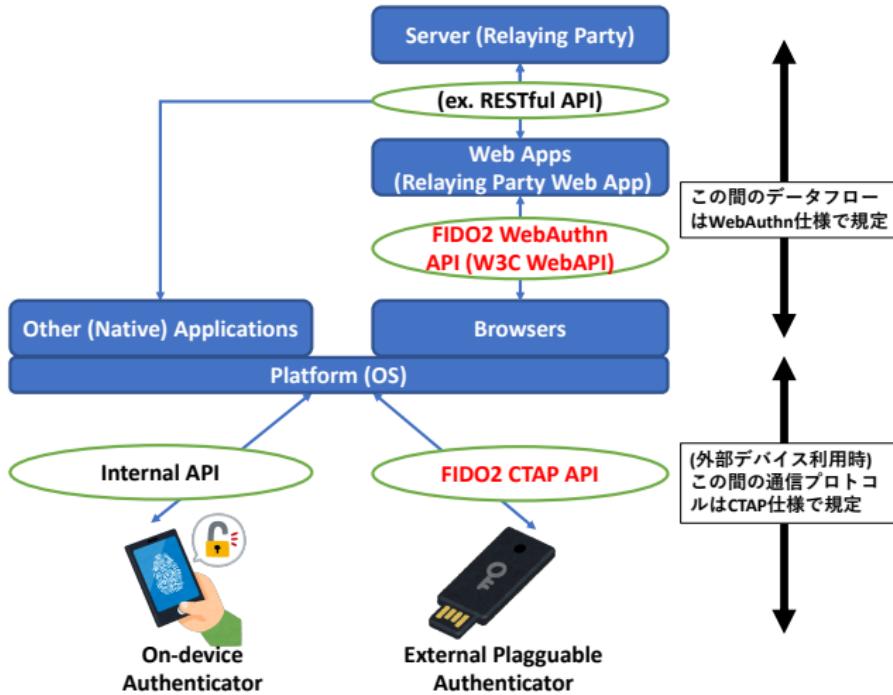
* FIDO2 Project

Figure: ©FIDO Alliance, from <https://fidoalliance.org/specifications/>

⁵Spec: <https://www.w3.org/TR/webauthn-1/>

⁶Spec: <https://fidoalliance.org/specs/fido2/fido-client-to-authenticator-protocol-v2.1-rd-20191217.html>

- **WebAuthn**: 内部/外部認証器を Call する WebAPI と、WebApp・サーバ間のデータフローを規定。
- **CTAP**: 外部認証器を Call する API と、クライアント/プラットフォームと認証器の通信プロトコルを規定。



FIDO2 CTAP⁷

USB/BLE/NFC 等で接続された HW セキュリティキーなどの外部認証器と、クライアントアプリ（ブラウザ等）およびプラットフォーム（OS）との間の通信プロトコルを規定。以下の要素で構成。

- USB/BLE/NFC など物理層の種別に応じた、通信確立のためのプロトコル
- 認証器での処理を Call する API
 - 外部認証器の情報取得
 - PIN によるローカルでのユーザ認証⁸
 - 認証器組込の秘密鍵での、ユーザ秘密鍵・証明書生成
 - ユーザ秘密鍵による署名の生成、など

ブラウザ・OS(のドライバ)は上記を実装した上で、より上位の WebAuthn のプロトコルをサポート。

⁷ <https://fidoalliance.org/specs/fido-v2.0-ps-20190130/fido-client-to-authenticator-protocol-v2.0-ps-20190130.html>

⁸ 指紋認証やジェスチャーなどは認証器のみで完結するので API は用意されない。署名生成などのときに認証器内での認証を要求するフラグを立てる。

FIDO2 WebAuthn⁹

Web ブラウザをプラットフォームとし、内部/外部認証器によって生成される署名を用いた、オンラインサーバでのユーザ認証のプロトコルを規定。より具体的には、以下を規定する。

- 認証器でのユーザの公開鍵証明書の生成、サーバへの登録プロトコル
- 認証器での署名生成、サーバでの認証プロトコル
- 認証器とやりとりするためブラウザが具備すべき Web API¹⁰。
大雑把に以下の 2 種類。
 - ユーザの公開鍵証明書の生成 (Credential Creation)
 - ユーザ秘密鍵による署名の生成 (Assertion Generation)

認証器とのやり取りはブラウザ/プラットフォームがサポート。
⇒ 基本的に Web App の観点からは、WebAuthn のみを意識する。

⁹ <https://www.w3.org/TR/webauthn-1/>

¹⁰ JavaScript で Call される API。ブラウザの内部でさらに認証器の API (CTAP や内部 API) を Call する。

補足: FIDO1

FIDO1 (v1.x) は、以下の 2 つの要素で構成されている。

- UAF (Universal Authentication Framework): 生体認証機能を持つ FIDO 対応端末 (スマートフォン等) でパスワードレス認証を行う機構。USB 接続などの外部 HW セキュリティキーは利用できない。
- U2F (Universal 2nd Factor)¹¹: ID・パスワード認証に加えた 2 要素認証を行うのに、外部 HW セキュリティキーを利用可能とする機構。

FIDO2 は、UAF と U2F を統合し、さらに外部 HW キーを用いてもパスワードレス認証可能な、より利便性の高い規格と見做せる。

¹¹ U2F は FIDO2 規格では CTAP1 と改称。FIDO2 で追加された仕様は CTAP2 と呼ばれる。

FIDO2 対応の認証器

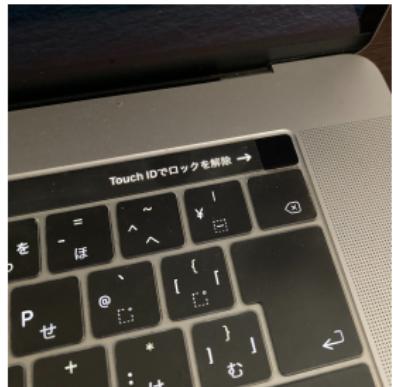
USB/NFC/BLE 等対応の外部認証器 (External Authenticator)、端末付属の認証器 (On-device/Internal Authenticator) 共々、様々な対応デバイスがリリースされつつある。



Security Key by Yubico
FIDO2 専用¹²



YubiKey 5Ci
FIDO2+OpenPGP+etc...



MacbookPro TouchID
FIDO2 認証可能¹³

¹² FIDO2 CTAP1=FIDO1 U2F には対応。

¹³ ブラウザ等が TouchID API を Call できれば FIDO2 の On-device Authenticator として動作。Chrome 等は対応済。

FIDO2 標準化状況

FIDO は業界団体の策定した規格ではあるが、

- **FIDO2 CTAP**: ITU-T で勧告として国際標準化¹⁴
- **FIDO2 WebAuthn**: W3C で勧告として国際標準化¹⁵

と、認証器とプラットフォーム/ブラウザ間の通信プロトコル、
サーバ・ブラウザ間の認証プロトコルの両者共に国際標準として
策定済。

¹⁴<https://fidoalliance.org/>

fido-alliance-specifications-now-adopted-as-itu-international-standards/

¹⁵<https://www.w3.org/2019/03/pressrelease-webauthn-rec.html.ja>

この後、FIDO2 WebAuthn の内容に実際に触れ、最新の認証技術について理解を深めてみよう。¹⁶

¹⁶今日は Web 技術から学ぶセキュリティに注力するため、ローレイヤの FIDO2 CTAP については別の機会で。

サンプルコードの準備

準備

説明を聞きつつ手を動かすため、まず環境準備。

今回は以下の 2 つを WebAuthn の API を Call しながら試す。

- 認証器を使ってユーザの公開鍵証明書を生成
⇒ 認証器からのメッセージを解してみて実際に証明書および生の公開鍵を取り出してみる。
 - 認証器を使ってユーザ認証用の証明を生成
⇒ 署名を解してみて、登録時に取り出した公開鍵で署名が通ることを確認してみる。
- ⇒ この 2 つが FIDO2 WebAuthn のパスワードレス認証の基礎。

環境

以下の環境が前提:

- Node.js LTS (≥ 12) がインストール済で yarn が使える¹⁷
- ブラウザとして、Google Chrome (系ブラウザ)、もしくは Firefox がインストール済み
- Visual Studio Code や WebStorm などの統合開発環境がセットアップ済みだとなお良い

¹⁷インストールコマンド: `npm i -g yarn`

JavaScript プロジェクトの準備

1 プロジェクトの GitHub リポジトリを Clone

```
$ git clone https://github.com/junkurihara/xxxxxxxxxxxxxxxxxxxx  
$ cd sample
```

2 依存パッケージのインストール

```
$ yarn install
```

3 ライブラリのビルド

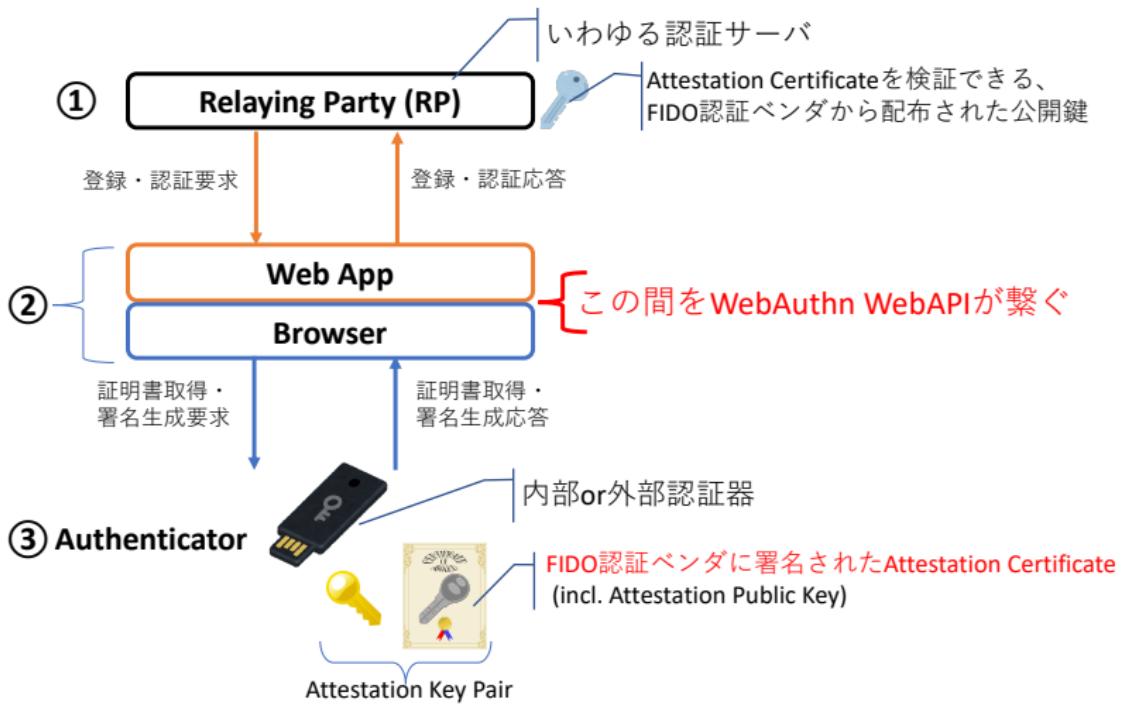
```
$ yarn build
```

FIDO2 WebAuthn のフロー

FIDO2 WebAuthn のフローの参加エンティティ

WebAuthn の動作フローでは、以下の 3 つの (抽象化された) 参加エンティティが存在。

- ① **Relaying Party (RP)**: サービス提供者の認証サーバ
- ② ブラウザ+RP の **WebApp**: ユーザおよび認証器とやり取り
- ③ **Authenticator**: 内部/外部認証器。以下をセキュアに保持。
 - **Attestation Key Pair**: 公開鍵・秘密鍵ペア
 - **Attestation Certificate**: 上記の公開鍵に対し、FIDO2 で承認された製造元が署名した証明書



WebAPI として用意される WebAuthn API は、ブラウザ経由で WebApp が認証器とやりとりする役割を担う。

FIDO2 における Attestation

FIDO2 では、「Attestation」という重要な概念が存在する。

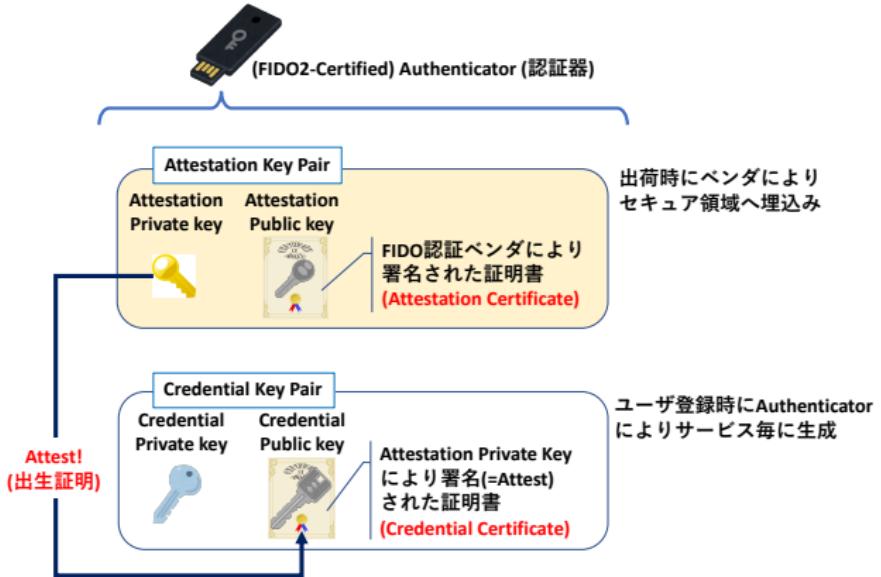
FIDO2 における Attestation

主として、「新しく生成・登録するユーザの公開鍵が、正しく FIDO2 認定を受けている認証器で生成された公開鍵であることを保証する機構」という意味。すなわち、出生証明。

Relying Party は、出生証明を確認してユーザを登録。
⇒ 偽造認証器を撃まされたユーザの登録を弾ける。
⇒ 認証器に基づく FIDO2 認証の安全性は維持。



Attestation の流れ。Attestation は 2 段階の証明で成立:



ユーザ登録時にユーザ公開鍵 (Credential Public Key) を出生証明して登録
⇒ Credential Certificate を Attestation Certificate で検証
⇒ Attestation Certificate を FIDO 認証ベンダの公開鍵¹⁸で検証

¹⁸ルート証明書

FIDO2 WebAuthn ユーザ登録フロー

WebAPI の観点では、認証器で新しいユーザ公開鍵証明書を生成。

FIDO2 WebAuthn ユーザ認証フロー

WebAPI の観点では、認証器においてユーザ秘密鍵で署名生成。

まとめ

