

# 技術検証報告書

2020年度新技術ワーキンググループ<sup>o</sup> Dチーム

# アジェンダ

## 1. FIDOについて

### 1.1 FIDOとは

### 1.2 FIDO利用の流れ

### 1.3 FIDOの仕様

### 1.4 FIDOの登録・認証フロー

## 2. 検証について

### 2.1 検証の目的と概要

### 2.2 検証の内容

#### 2.2.1 検証①

#### 2.2.2 検証②

#### 2.2.3 検証③

## 3. 結果と考察

### 3.1 検証結果報告

#### 3.1.1 検証①

#### 3.1.2 検証②

#### 3.1.3 検証③

### 3.2 所感・まとめ

# 1. FIDOについて

## 1.1 FIDOとは

---

- Fast IDentity Onlineの略称
- FIDOアライアンスが推進している  
パスワードレス認証を実現するための仕組み
- 認証情報がネットワーク上に流れない為、  
情報が漏洩するリスクがほとんどない



※FIDOアライアンスとは  
パスワードレス認証を用いた認証技術  
(FIDO) の標準化を目的として  
2012年に米国で発足した団体。

→現在普及しているパスワード認証に代わる次世代の認証技術

# 1. FIDOについて

## 1.2 FIDO利用の流れ

端末：スマートフォン,PCなどのデバイス

認証器：秘密鍵・公開鍵のペアを生成し、FIDOサーバへ送信する署名を生成  
※FIDO規格に対応している必要がある

FIDOサーバ：ユーザーのIDを登録、認証し管理するサーバ

利用者は**端末**と**認証器**を用意

FIDO認証の利用を開始



# 1. FIDOについて

## 1.3 FIDOの仕様①

### UAF: (Universal Authentication Framework)



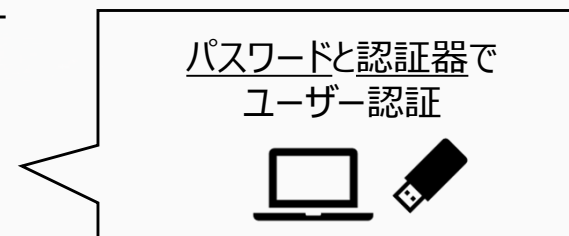
パスワードレスの認証方式  
主にスマートフォンアプリで利用



### U2F: (Universal Second Factor)



「記憶認証」と「所有認証」を用いた2要素認証  
FIDO未対応端末でも、認証器を  
購入することで利用可能



# 1. FIDOについて

## 1.3 FIDOの仕様②

### FIDO2:WebAuthn (Web認証) & CTAP



WebAuthn:パスワードレス認証をWebブラウザで使用可能にするAPI

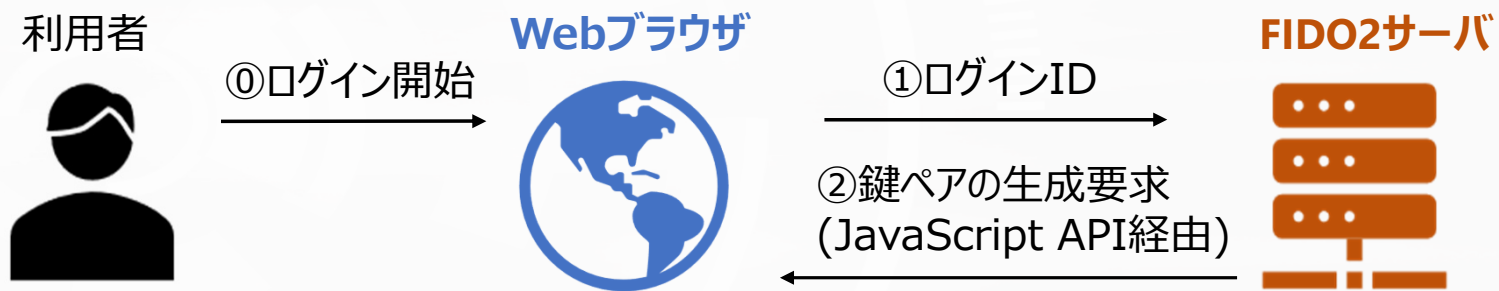
CTAP:デバイスと認証器の通信プロトコル



パスワードレスでWebブラウザ上の認証が可能

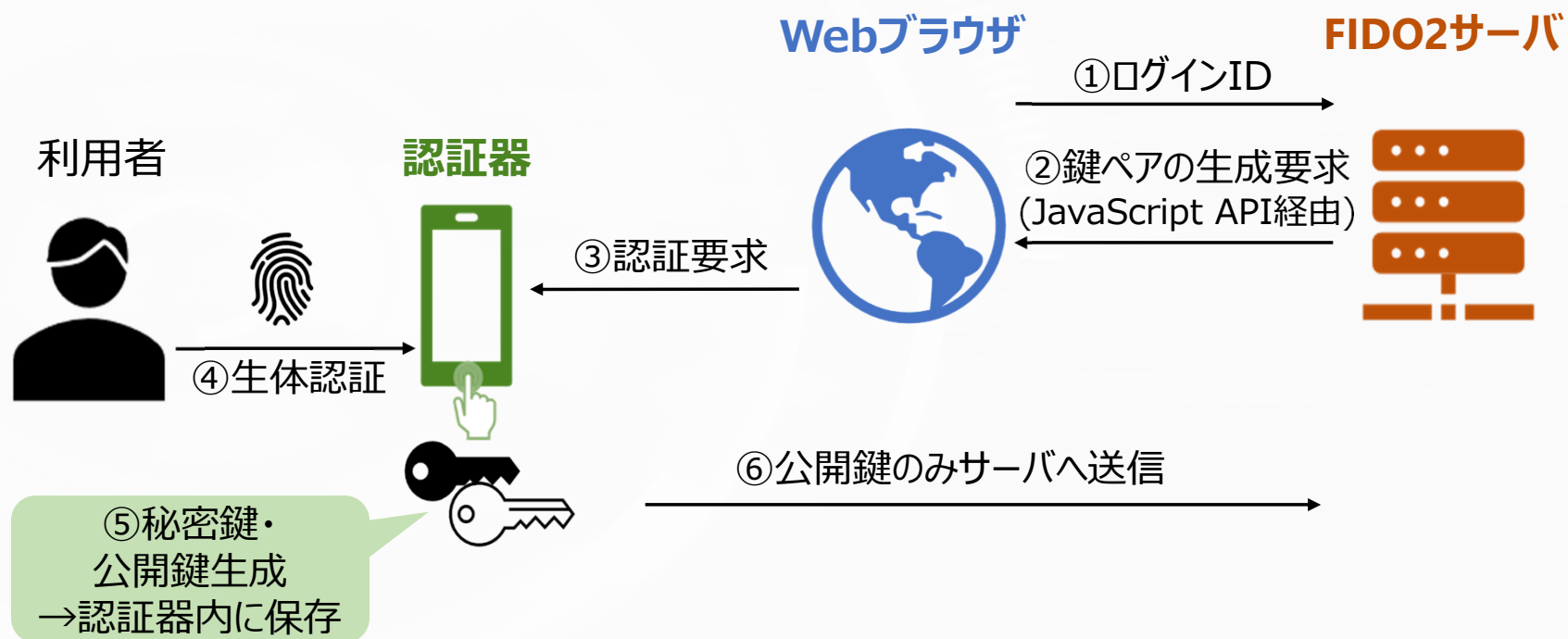
# 1. FIDOについて

## 1.4 FIDO2仕組み – 登録フロー –



# 1. FIDOについて

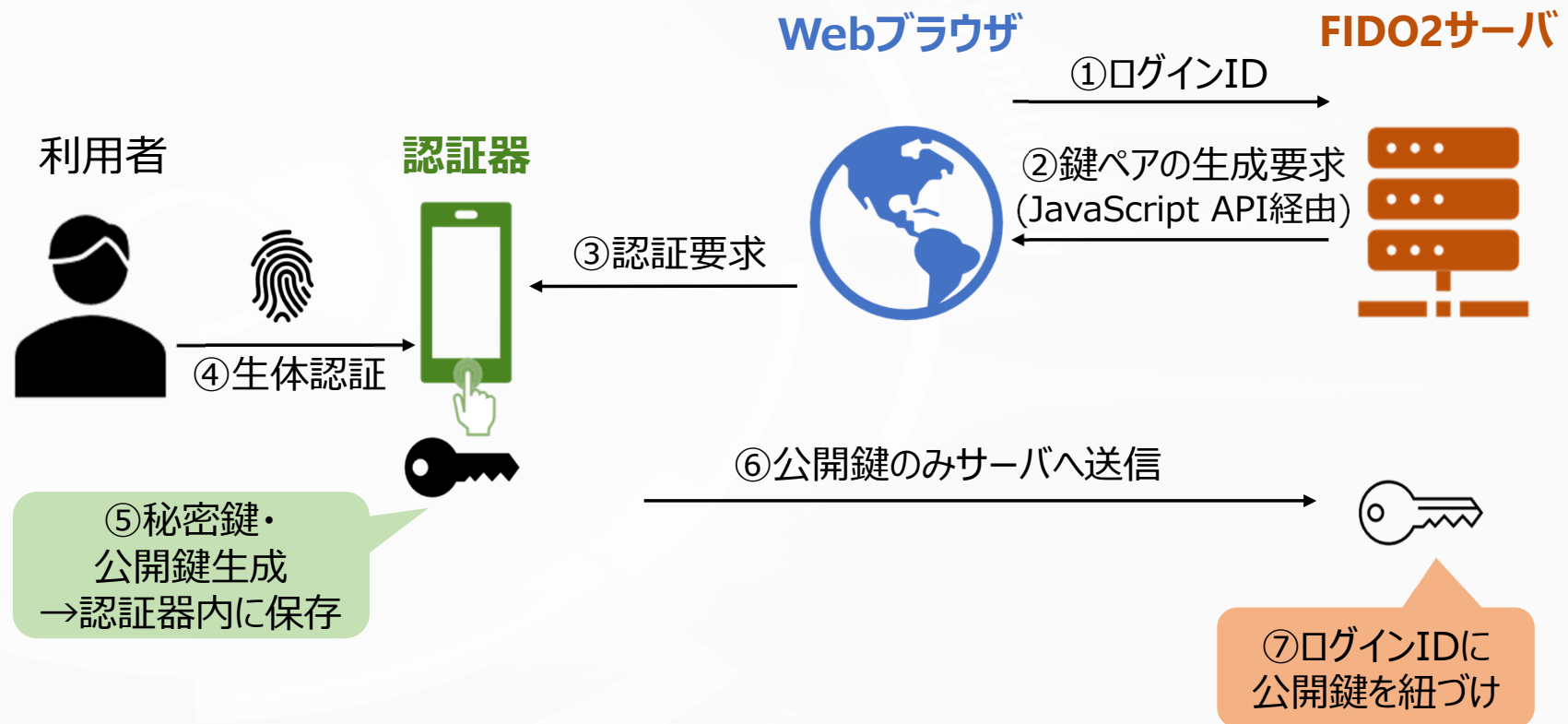
## 1.4 FIDO2仕組み – 登録フロー –





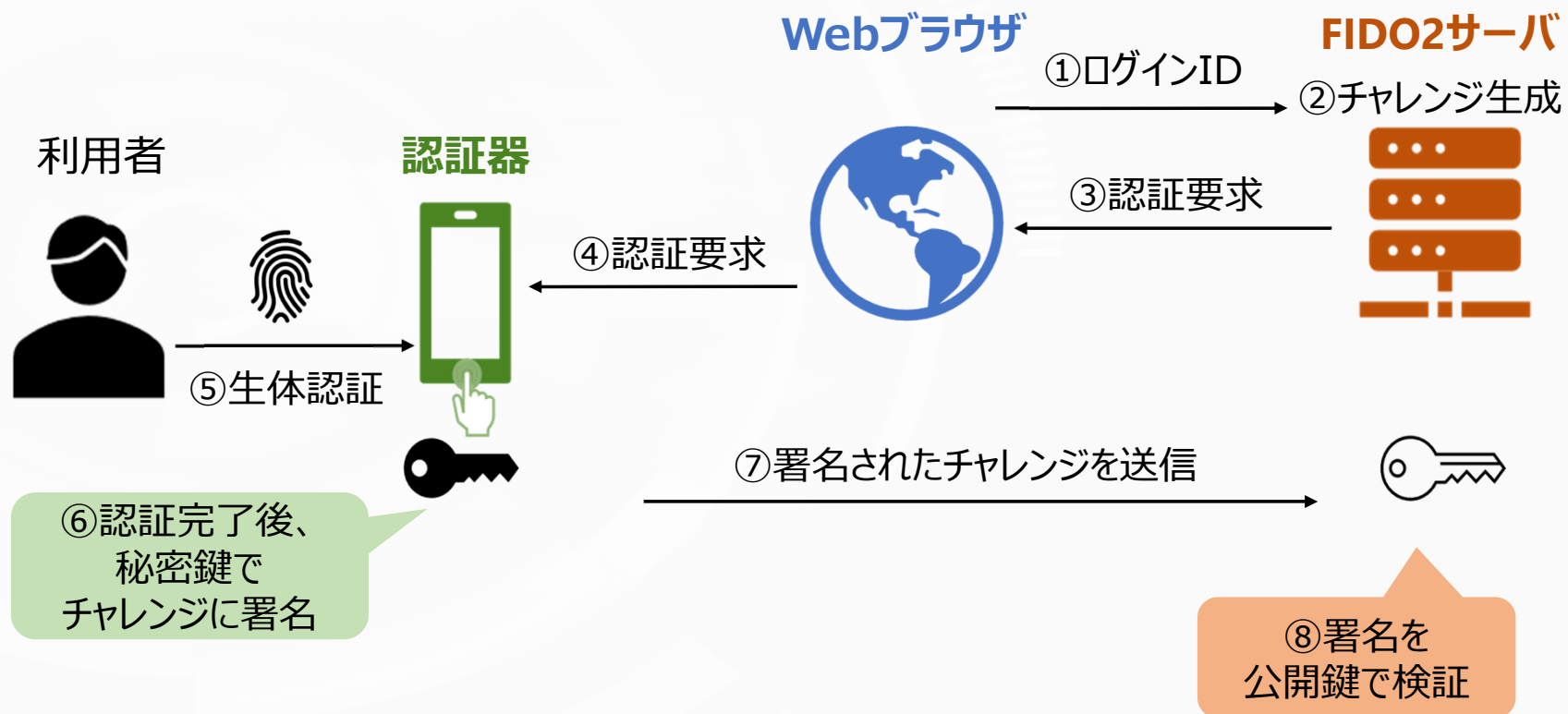
# 1. FIDOについて

## 1.4 FIDO2仕組み – 登録フロー –



# 1. FIDOについて


## 1.4 FIDO2仕組み – 認証フロー –



# 1. FIDOについて

## 1.4 FIDO2仕組み – 認証フロー –





# 検証について

## 2.1 検証の目的と概要

### 検証目的

- FIDO2の動作に対する知見の獲得
- FIDO2を利用した認証システムの構築に対する知見の獲得

### 検証概要

検証①： FIDOサーバと認証するアプリを構築し、端末とサーバ間の通信を確認

検証②： オンプレにてFIDO2に対応したSSO用認証サーバを構築し、SSOを実装

検証③： FIDO2に対応したSSOのクラウドサービスでSSOを実装

### 検証期間

2020年10月7日 - 2021年3月中旬

## 2. 検証内容

### 2.2.1 検証①

検証内容：

FIDOサーバとFIDO2に対応したアプリケーションを構築し、認証を実施

検証目標：

- ・ FIDOサーバとアプリケーションを構築すること
- ・ FIDO2における認証フローを確認すること

検証構成：



## 2.2 検証内容

### 2.2.2 検証②

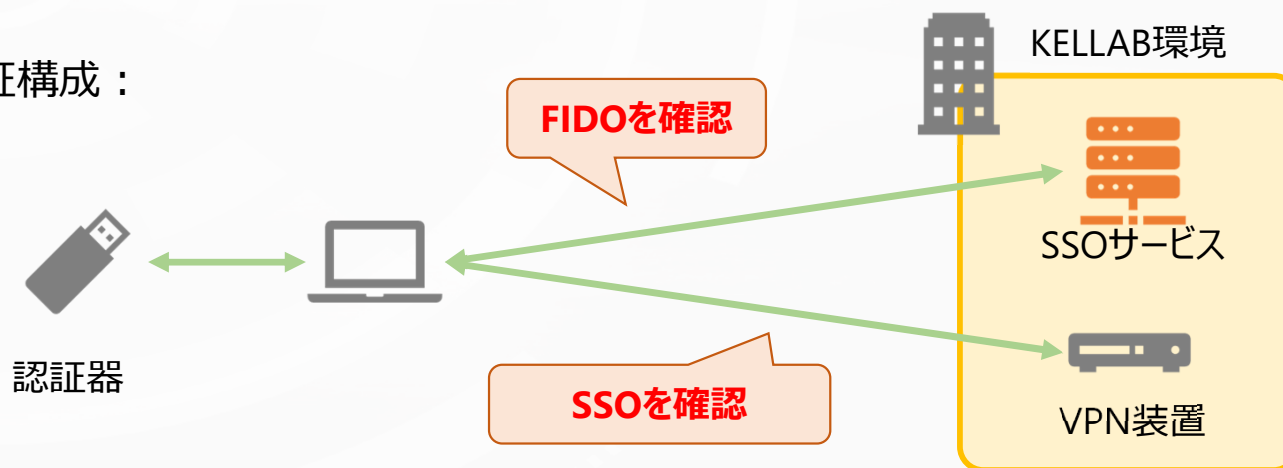
検証内容：

FIDO2に対応したオンプレミス型のSSOサービスを構築し、VPN装置にSSOでログイン

検証目標：

- ・ SSOサービスにFIDO2でログイン可能であること
- ・ SSOを利用してVPN接続が可能であること

検証構成：



## 2.2 検証内容

### 2.2.3 検証③

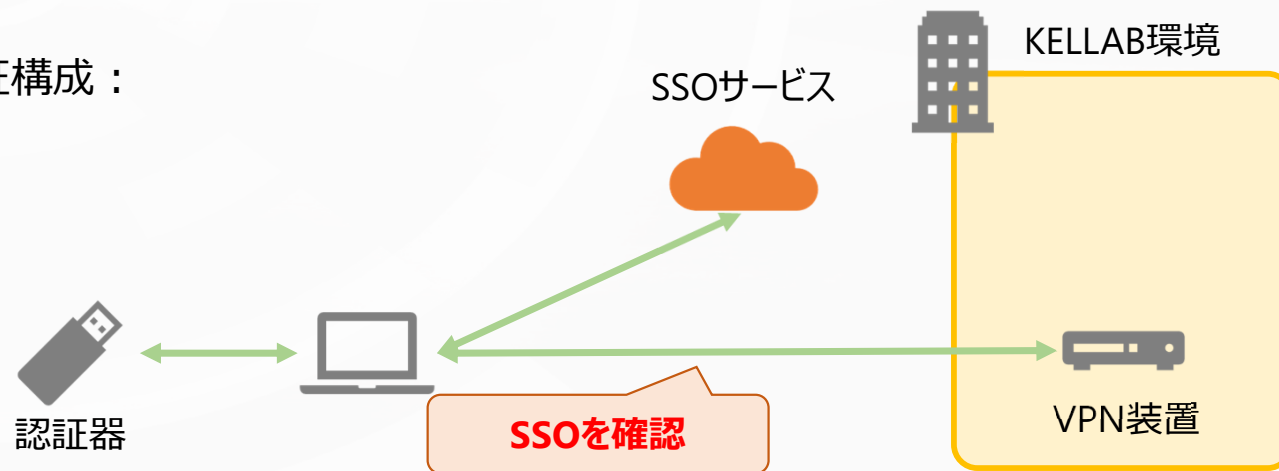
検証内容：

FIDO2に対応したクラウド型SSOサービスを利用し、VPN装置にSSOでログイン


検証目標：

- ・ SSOサービスにFIDO2でログイン可能であること
- ・ SSOを利用してVPN接続が可能であること

検証構成：







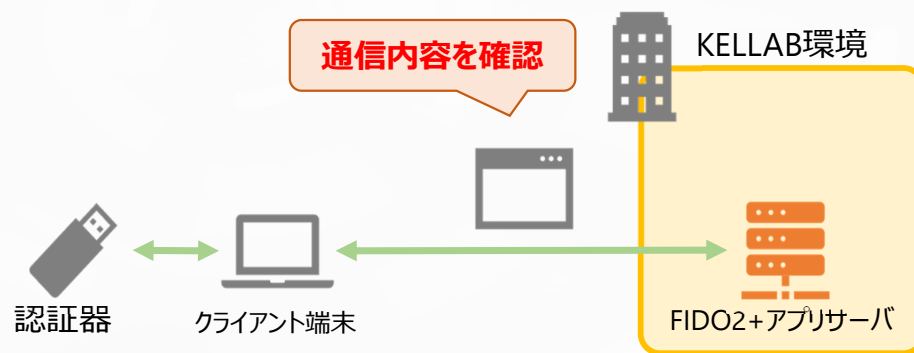
# 結果と考察

## 3. 結果と考察

### 3.1.1 検証結果報告①

検証目標①：FIDO2サーバとサンプルアプリケーションを構築すること →OK

検証目標②：FIDO2における登録/認証フローを確認すること →OK



#### 検証作業

- Linuxマシン上にFIDO2サーバを構築する
- Linuxマシン上にサンプルアプリケーションをインストールする
- クライアント端末と認証器を使用してユーザを登録する
- 登録したユーザアカウントを使ってログインする

## 3. 結果と考察

### 3.1.1 検証結果報告①

---

#### ● FIDO2サーバ・サンプルアプリケーションのインストールソース

GitHubにてStrongkeyが公開しているFIDO2サーバ・サンプルアプリケーションのサンプルコード及びインストール手順

<https://github.com/StrongKey/fido2>

#### ● 検証環境(FIDO2+アプリサーバ)の条件

##### ・サポートOS

RedHat 7, CentOS 7, Oracle 7, Ubuntu 18.04, Debian 9, Amazon Linux 2

##### ・メモリ4GB以上、ディスク空き容量10GB以上のVM

##### ・完全修飾ドメイン名(FQDN)を持ち、名前解決できる状態

## 3. 結果と考察

### 3.1.1 検証結果報告①

本検証の検証環境

#### ●FIDO2+アプリサーバ

- ・Linux RedHat 7
- ・メモリ32GB、ディスク容量100GB
- ・FQDN:fidoap.kensho1.local

#### ●DNSサーバ(FIDO2+APサーバの名前解決用)

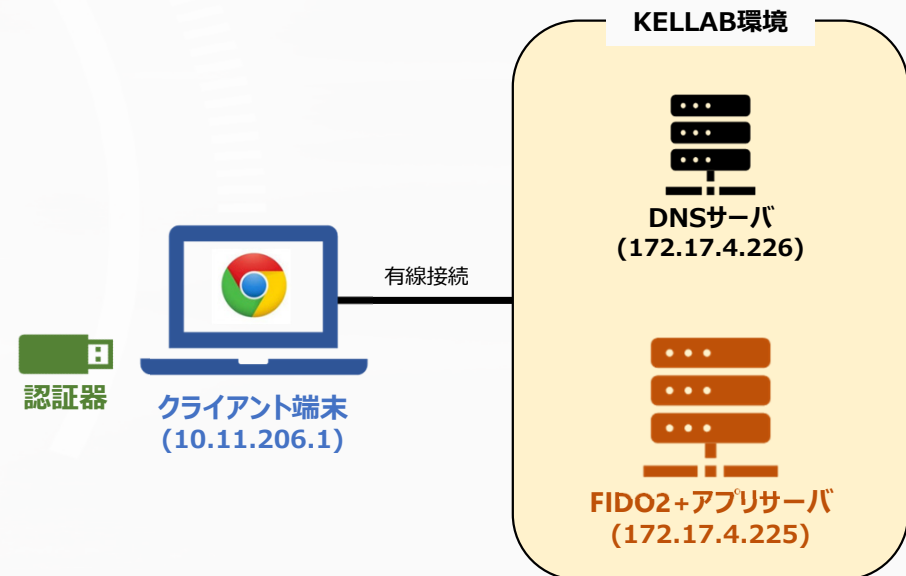
- ・Windows Server 2016

#### ●クライアント端末

- ・Windows 10
- ・サンプルアプリ利用ブラウザ: Google Chrome

#### ●認証器

- ・BioPassFIDO2 K27(Feitian Technologies社)



## 3. 結果と考察

### 3.1.1 検証結果報告①

#### FIDO2サーバ構築手順

- ① FIDO2サーバのFQDN名を設定

```
$ sudo hostnamectl set-hostname fidoap.kensho1.local
```

- ② rootユーザでwgetをインストール

```
$ sudo yum install wget
```

- ③ バイナリ配布ファイルをダウンロード

```
$ wget https://github.com/StrongKey/fido2/raw/master/fido2server-v4.3.0-dist.tgz
```

- ④ スクリプトを実行し、FIDO2サーバとアプリサーバPayaraをインストール、StrongKeyユーザアカウントを自動作成

```
$ sudo ./install-skfs.sh
```

## 3. 結果と考察

### 3.1.1 検証結果報告①

#### FIDO2サーバ構築手順

- ⑤ FIDO2サーバの動作確認。以下のようにWADLファイルが表示されれば動作に問題なし。

```
$ curl -k https://localhost:8181/skfs/rest/application.wadl
```

```
[fidoadmin@localhost ~]$ curl -k https://localhost:8181/skfs/rest/application.wadl
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<application xmlns="http://wadl.dev.java.net/2009/02">
  <doc xmlns:jersey="http://jersey.java.net/" jersey:generatedBy="Jersey: 2.25.1 2017-01-19 16:23:50"/>
  <doc xmlns:jersey="http://jersey.java.net/" jersey:hint="This is simplified WADL with user and core resources only
. To get full WADL with extended resources use the query parameter detail. Link: https://localhost:8181/skfs/rest/appl
ication.wadl?detail=true"/>
  <grammars/>
  <resources base="https://localhost:8181/skfs/rest/">
    <resource path="">
      <resource path="/ping">
        <method id="ping" name="POST">
          <request>
            <representation mediaType="application/json"/>
          </request>
          <response>
            <representation mediaType="*/*/>
          </response>
        </method>
      </resource>
      <resource path="/preregister">
        <method id="preregister" name="POST">
          <request>
            <representation mediaType="application/json"/>
          </request>
          <response>
            <representation mediaType="application/json"/>
          </response>
        </method>
      </resource>
      <resource path="/preauthenticate">
        <method id="preauthenticate" name="POST">
          <request>
            <representation mediaType="application/json"/>
          </request>
          <response>
            <representation mediaType="application/json"/>
          </response>
        </method>
      </resource>
    </resources>
  </application>
</doc>
```

## 3. 結果と考察

### 3.1.1 検証結果報告①

#### サンプルアプリケーション(Java)構築手順

- ① StrongKeyユーザでログイン後、アプリケーションの構成ファイルを作成し、FIDO2サーバの場所を記載

```
$ echo " webauthntutorial.cfg.property.apiuri = https://fidoap.kensho1.local:8181 " >  
/usr/local/strongkey/webauthntutorial/etc/webauthntutorial-configuration.properties
```

- ② Webアプリケーションファイル「basicserver.war」をダウンロード

```
$ wget https://github.com/StrongKey/fido2/raw/master/sampleapps/java/basic/basicserver.war
```

- ③ ②をpayaraに追加

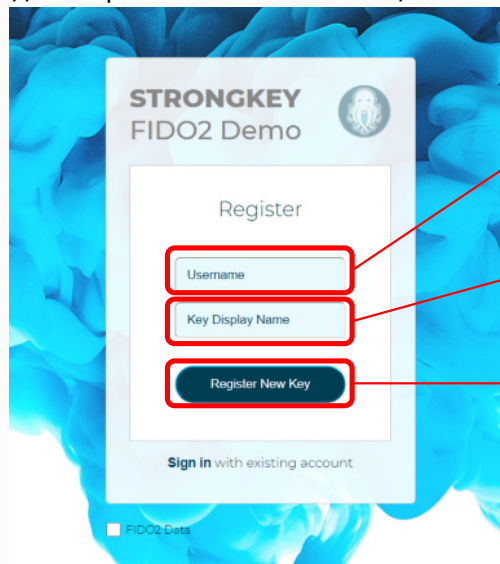
```
$ payara41 / glassfish / bin / asadmin deploy basicserver.war
```

## 3. 結果と考察

### 3.1.1 検証結果報告①

#### 登録フローの確認

https://fidoap.kensho1.local:8181/basicserver/



①任意のユーザ名を入力

②任意の鍵名を入力

③Register New Keyを押下

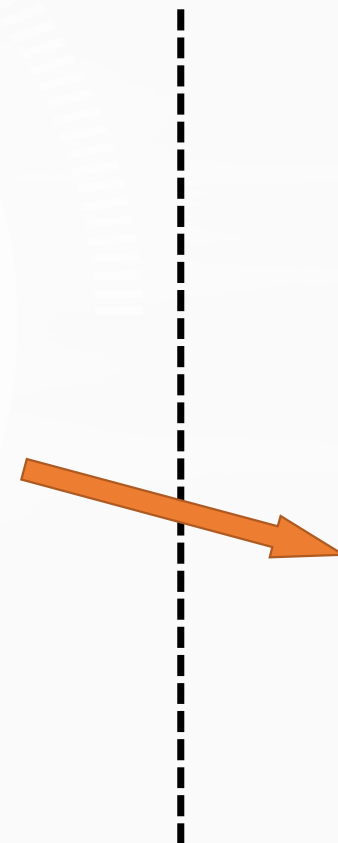


クライアント端末  
(10.11.206.1)

④新しいユーザ・鍵登録要求を受信



FIDO2サーバ  
(172.17.4.225)





## 3. 結果と考察

### 3.1.1 検証結果報告①

#### 登録フローの確認

https://fidoap.kensho1.local:8181/basicserver/



⑦ 認証を実施

⑥ 要求受信後、  
認証器を使用して鍵を登録する  
ポップアップウィンドウが表示される

⑧ 秘密鍵と公開鍵を生成し  
公開鍵のみFIDO2サーバへ送信

⑤ 秘密鍵と公開鍵の生成（鍵の認証）  
要求をクライアント端末へ向けて送信

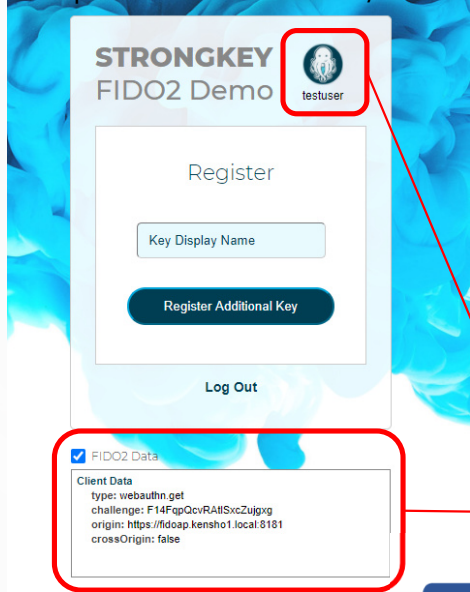


# 3. 結果と考察

## 3.1.1 検証結果報告①

### 登録フローの確認

https://fidoap.kensho1.local:8181/basicserver/



⑧秘密鍵と公開鍵を生成し  
公開鍵のみFIDO2サーバへ送信

⑨公開鍵を受信し登録完了、  
同時にログインも完了となる

ログイン完了画面を配信

ユーザ名が表示される

登録したユーザ・認証器の  
情報を見ることが出来る



認証器



クライアント端末  
(10.11.206.1)



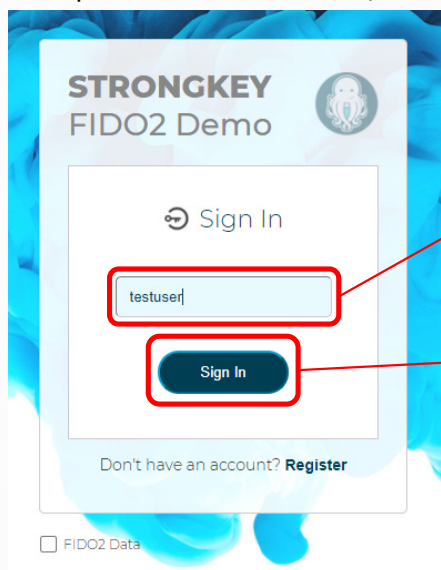
FIDO2サーバ  
(172.17.4.225)

# 3. 結果と考察

## 3.1.1 検証結果報告①

### 認証フローの確認

https://fidoap.kensho1.local:8181/basicserver/



①ログインするユーザ名を入力

②Sign Inを押下



クライアント端末  
(10.11.206.1)

③ログイン要求を受信



FIDO2サーバ  
(172.17.4.225)

## 3. 結果と考察

### 3.1.1 検証結果報告①

#### 認証フローの確認

https://fidoap.kensho1.local:8181/basicserver/



⑥ 認証を実施



認証器



クライアント端末  
(10.11.206.1)

⑤ 受信後、認証器を使用して  
ユーザを認証するポップアップウインドウ  
が表示される

⑦ 秘密鍵から生成した署名を記載した  
チャレンジをFIDO2サーバへ送信

④ ユーザの認証のための  
チャレンジを送信



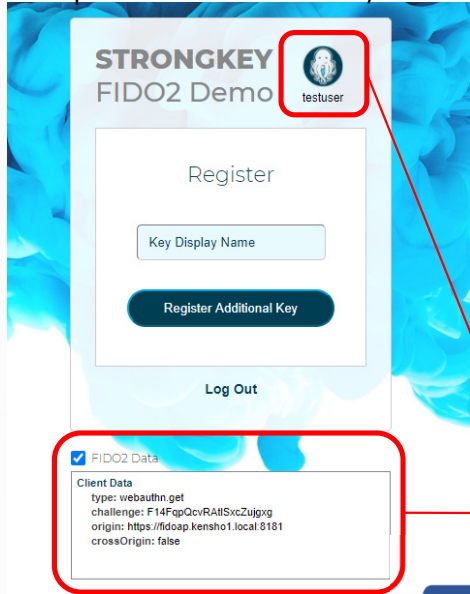
FIDO2サーバ  
(172.17.4.225)

# 3. 結果と考察

## 3.1.1 検証結果報告①

### 認証フローの確認

https://fidoap.kensho1.local:8181/basicserver/



⑦ 秘密鍵から生成した署名を記載した  
チャレンジをFIDO2サーバへ送信

ユーザ名が表示される

登録したユーザ・認証器の  
情報を見ることが出来る

ログイン完了画面を配信

⑧ 既に登録してある公開鍵でチャレンジを  
検証し、合致すればログイン完了となる



認証器



クライアント端末  
(10.11.206.1)



FIDO2サーバ  
(172.17.4.225)

# 3. 結果と考察

## 3.1.1 検証結果報告①

### FIDO2サーバでの登録ログ

```
[2021-03-10T10:10:22.072+0900] [Payara 4.1] [INFO] [] [] [tid: _ThreadID=29 _ThreadName=http-thread-pool::http-listener-2(6)] [timeMillis: 1615338622072] [levelValue: 800] [[
  POST https://fidoap.kensho1.local:8181/skfs/rest/preregister HTTP/1.1]]

[2021-03-10T10:10:22.126+0900] [Payara 4.1] [INFO] [] [] [tid: _ThreadID=33 _ThreadName=http-thread-pool::http-listener-2(10)] [timeMillis: 1615338622126] [levelValue: 800] [[
  json = {"username":"testuser","displayname":"testuser","options":{"attestation":"direct"},"extensions":{"}}]]

[2021-03-10T10:10:22.141+0900] [Payara 4.1] [INFO] [FIDO-MSG-0001] [SKFS] [tid: _ThreadID=33 _ThreadName=http-thread-pool::http-listener-2(10)] [timeMillis: 1615338622141] [levelValue: 800] [[
  FIDO-MSG-0001: Received preregister request; Input: [TXID=33-1615338622141]
  did=1
  protocol=FIDO2_0
  username=testuser
  displayname=testuser
  options={"attestation":"direct"}
  extensions={}]]

[2021-03-10T10:10:22.253+0900] [Payara 4.1] [INFO] [FIDO-MSG-0002] [SKFS] [tid: _ThreadID=33 _ThreadName=http-thread-pool::http-listener-2(10)] [timeMillis: 1615338622253] [levelValue: 800] [[
  FIDO-MSG-0002: Done with preregister request; Output: [TXID=33-1615338622141, START=1615338622141, FINISH=1615338622253, TTC=112]
  FIDO2Registration Challenge parameters = {"Response":{"rp":{"name":"demo.strongauth.com:8181"},"user":
  {"name":"testuser","id":"mst_2Lcq641B99cRMGKv4aX0bJ7uBH_Cqy8NWE4eIbY","displayName":"testuser"},"challenge":"tsLD0tBn9d4Zvwrbl2m6fg","pubKeyCredParams":[{"type":"public-key","alg":-7}, {"type":"public-
  key","alg":-35}, {"type":"public-key","alg":-36}, {"type":"public-key","alg":-8}, {"type":"public-key","alg":-43}, {"type":"public-key","alg":-65535}, {"type":"public-key","alg":-257}, {"type":"public-key","alg":-258},
  {"type":"public-key","alg":-259}, {"type":"public-key","alg":-37}, {"type":"public-key","alg":-38}, {"type":"public-key","alg":-39}], "excludeCredentials":[{"type":"public-key","id":"VUL0siUCvGcaa-
  r20Hm6_KLGeTdqg3yPm9J5g2ppnIwQoXWlW1t2rHhf-oifWwN9DrXw5LjodnSdN8JXhxg","alg":-7}], "attestation":"direct"}}]]

[2021-03-10T10:10:22.256+0900] [Payara 4.1] [INFO] [] [] [tid: _ThreadID=29 _ThreadName=http-thread-pool::http-listener-2(6)] [timeMillis: 1615338622256] [levelValue: 800] [[
  {"Response":{"rp":{"name":"demo.strongauth.com:8181"},"user":
  {"name":"testuser","id":"mst_2Lcq641B99cRMGKv4aX0bJ7uBH_Cqy8NWE4eIbY","displayName":"testuser"},"challenge":"tsLD0tBn9d4Zvwrbl2m6fg","pubKeyCredParams":[{"type":"public-key","alg":-7}, {"type":"public-
  key","alg":-35}, {"type":"public-key","alg":-36}, {"type":"public-key","alg":-8}, {"type":"public-key","alg":-43}, {"type":"public-key","alg":-65535}, {"type":"public-key","alg":-257}, {"type":"public-key","alg":-258},
  {"type":"public-key","alg":-259}, {"type":"public-key","alg":-37}, {"type":"public-key","alg":-38}, {"type":"public-key","alg":-39}], "excludeCredentials":[{"type":"public-key","id":"VUL0siUCvGcaa-
  r20Hm6_KLGeTdqg3yPm9J5g2ppnIwQoXWlW1t2rHhf-oifWwN9DrXw5LjodnSdN8JXhxg","alg":-7}], "attestation":"direct"}}]]
```

ユーザ情報

登録された公開鍵(チャレンジ)の情報

ユーザのパスワードや生体情報はFIDO2サーバに存在しない

## 3. 結果と考察

### 3.1.1 検証結果報告①

#### 躓きポイント・注意事項

##### ■ 秘密鍵と公開鍵の再生成

keymanagerのjarファイルにバグがあり新しいユーザーの登録が正しく行えなかったため、鍵を再生成した。

##### ■ 地域の変更

Webアプリケーションとサーバー間で通信リクエストの構成が異なっており、新しいユーザーの登録が正しく行えなかった。構成を揃えるため、地域設定を日本から米国に変更した。

##### ■ FQDNの設定

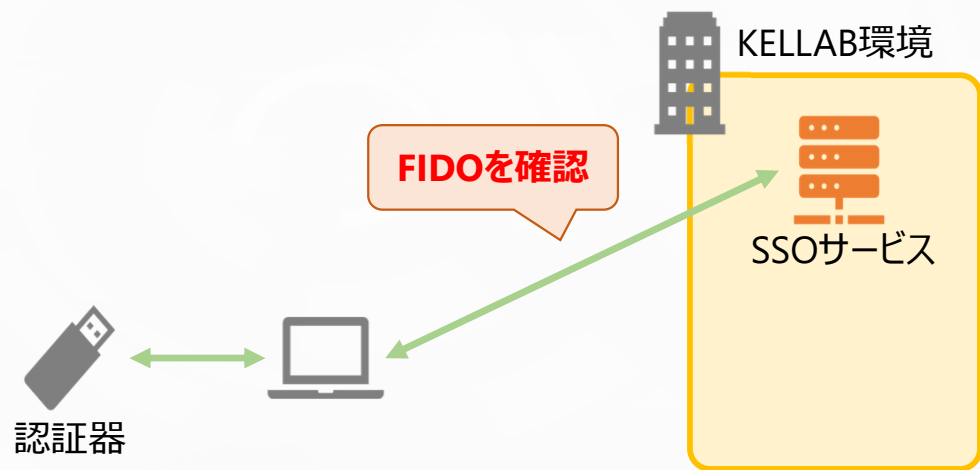
ドメインの名前解決ができずWebアプリケーションにアクセスできなかったため、DNSサーバを追加で構築し名前解決を実現した。

## 3. 結果と考察

### 3.1.2 検証結果報告②

検証目標①：SSOサービスにFIDO2でログイン可能であること

→OK



#### 検証作業

- ・ FIDO2に対応したオンプレミス型SSOサービスである Open AMを構築する

#### 検証完了条件

- ・ 認証器を利用してOpen AMにログインができること



## 3. 結果と考察

### 3.1.2 検証結果報告②

OpenAMインストール手順：

- ① 必要ファイルをインストールする

(JDK8以降、tomcat7以降、SSL証明書とApache-HTTP)

```
$ yum -y install tomcat mod_ssl
```

- ② Open AMコンソーシアムのgithubに公開されてる、OpenAM14のwarファイルをダウンロードする

※FIDO2はOpen AM 14.0.0以上のバージョンにしか対応していないので注意する

- ③ FQDN、ホスト名を登録する (fidotest.kensho4.local に設定)

```
$ vi /etc/hosts
```

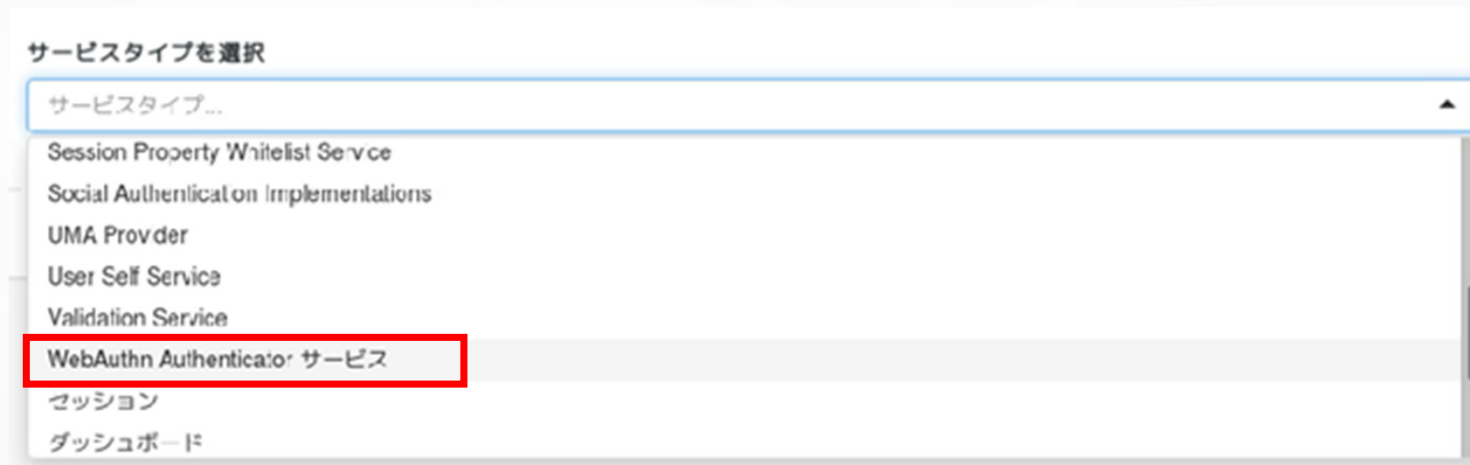
```
$ vi /etc/hostname
```

## 3. 結果と考察

### 3.1.2 検証結果報告②

WebAuthn認証設定方法：

- ① Open AMの管理ページに「amadmin」でログインする。 (<http://fidotest.kensho4.local:8080/openam/>)
- ② WebAuthn AuthenticatorサービスをOpen AMに追加する。



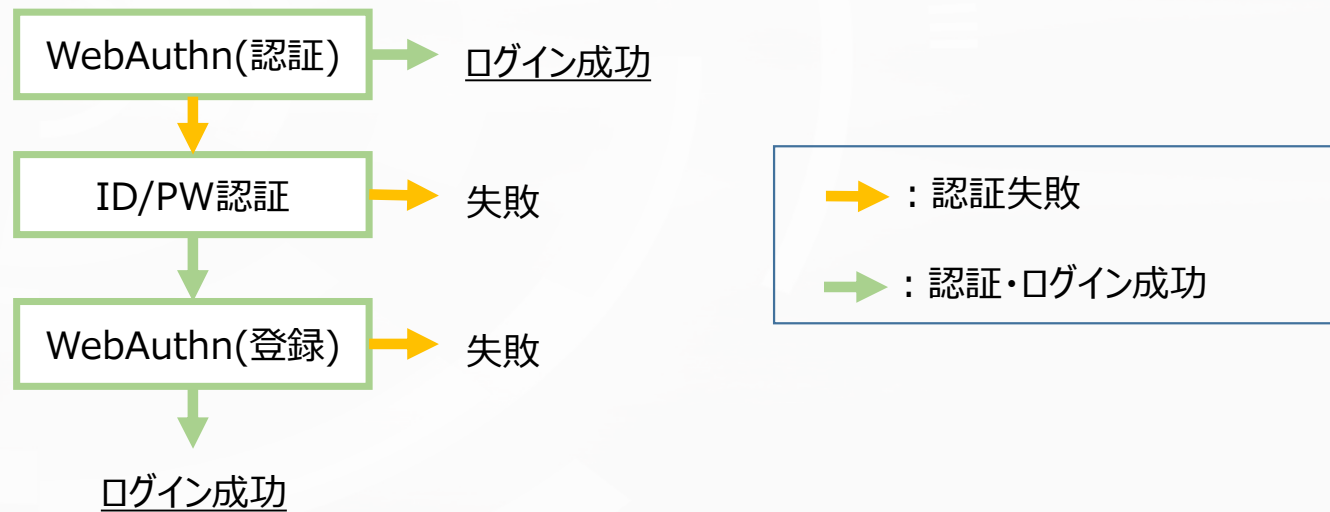
## 3. 結果と考察

### 3.1.2 検証結果報告②

パスワードレス認証連鎖の設定：

認証連鎖：認証モジュールを組み合わせて、それぞれの認証結果をもとに、次の認証モジュールの要不要を判定する機能

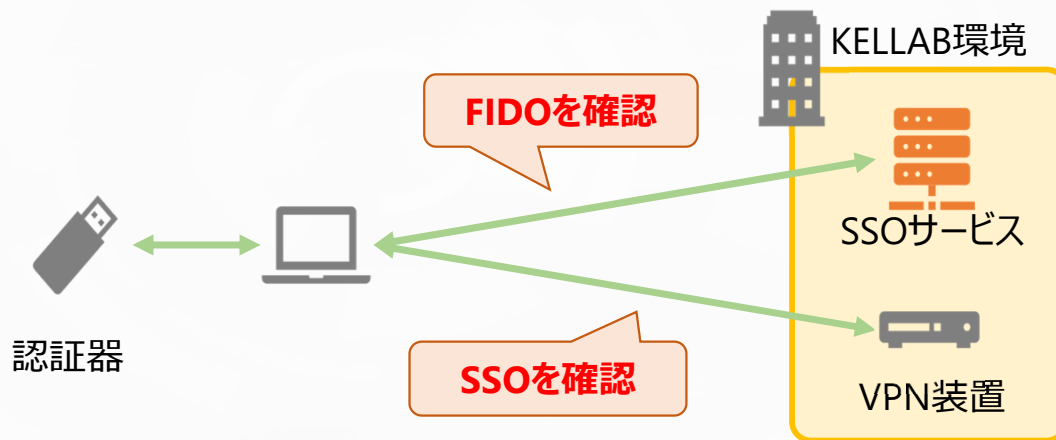
➡ 複数の認証モジュールを組み合わせることで、よりセキュアな認証を可能にする「多要素認証」機能を実装可能



## 3. 結果と考察

### 3.1.2 検証結果報告②

検証目標②：SSOを利用してVPN接続が可能であること ➡NG



#### 検証作業

- ・ OpenAM側とVPN装置側でSSOの設定を行う

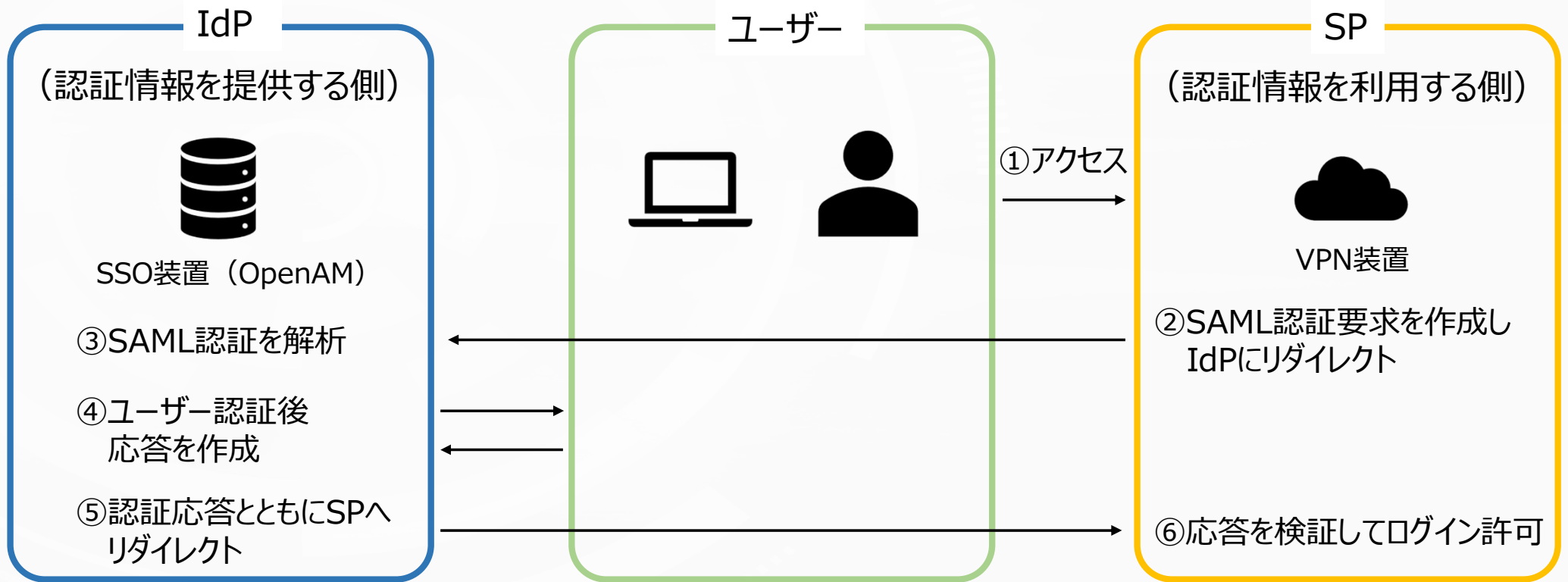
#### 検証完了条件

- ・ SSOを利用してVPN接続ができること

# 3. 結果と考察

## 3.1.2 検証結果報告②

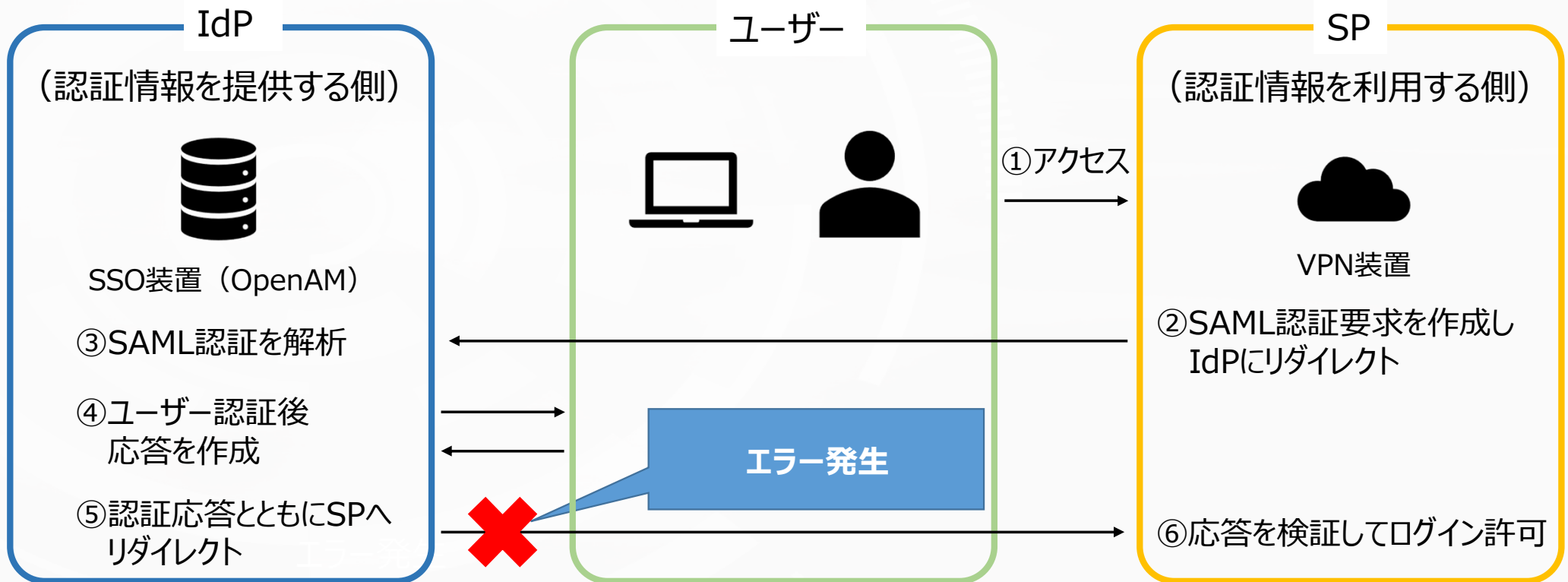
SAML : 異なるドメイン間でユーザー認証を行うための認証情報の規格



# 3. 結果と考察

## 3.1.2 検証結果報告②

検証②のユーザーからVPN装置への認証フロー



## 3. 結果と考察

### 3.1.2 検証結果報告②

OpenAMにおけるログ

access.csv : すべてのアクセスに対して、誰が、いつ、何をしたのかを出力するファイル



```
4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.150 Safari/537.36"  
"シングルサインオンも連携も実行できません。(The private key was null.)""}",
```

## 3. 結果と考察

### 3.1.2 検証結果報告②

エラー対処

```
4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.150 Safari/537.36"  
"シングルサインオンも連携も実行できません。 (The private key was null.)" }", "
```

➡ 「秘密鍵がない」というエラーが発生しているため、証明書が原因であると推測

OpenAMの設定画面

<https://114.156.12.23:443/SAML20/SP>

証明書エイリアス

署名:

暗号化:

キーサイズ:

アルゴリズム:

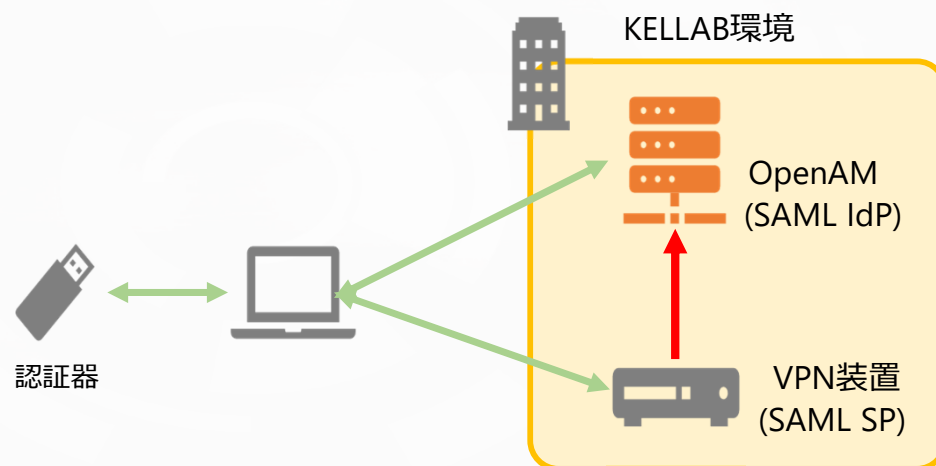
VPN装置 (SP) の証明書エイリアスが  
空白となっており何も登録されていない



## 3. 結果と考察

### 3.1.2 検証結果報告②

考えられる対策

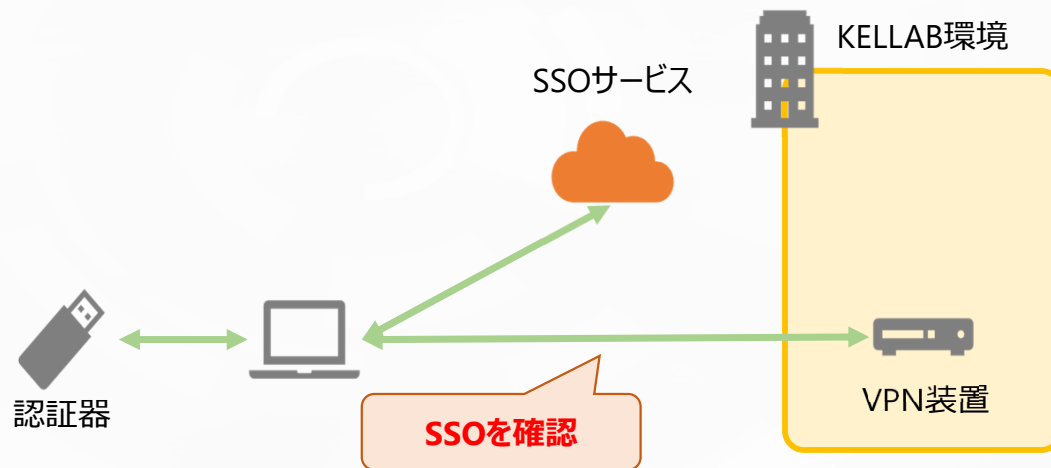


OpenAMにVPN装置の  
証明書を登録出来れば、解決する可能性がある

## 3. 結果と考察

### 3.1.3 検証結果報告③

検証目標：SSOを利用してVPN接続が可能であること **➡NG**



#### 検証作業

- ・クラウド型SSOサービスであるSoliton社OneGate側とVPN装置側でSSOの設定を行う

#### 検証完了条件

- ・SSOを利用してVPN接続が利用可能であること

## 3. 結果と考察

### 3.1.3 検証結果報告③

---

- SAML連携を利用した**オンプレミスのサービスへのSSOは未対応**  
※ロードマップとしては将来的に実装予定  
  
→ VPN装置との連携は現時点では不可能
- OneGateにおけるSSO(SAML)対応サービス一覧
  - O365 • G Suite • box • cybouzu.com • Salesforce
  - AgileWorks • Akamai • ASANA • AWS • SAP • Slack
  - Cisco Webex • X-point etc.

## 3. 結果と考察

### 3.2 所感・まとめ

---

- アプリケーションやSSOサービスへのFIDO2導入が実現すれば、ユーザの利便性向上とセキュリティ強化が同時に見込める
- FIDO2はセキュリティが厳重且つ認証が簡単であるが、最先端の技術であるため情報が少ない  
→今後も継続的な検証を行う価値あり
- エラーの未解決など検証に不十分な点があったため、引き続きFIDO2に関する調査・検証を行い、KELの事業領域拡大の一助となる可能性を模索する
- FIDO2に対応しているクラウド型SSOサービスにおいて、SSO連携が可能なサービスには制限があるため、選定には注意が必要である

ご清聴誠にありがとうございました