今更聞けない電子認証入門
OAuth 2.0 / OIDC から FIDO まで
＜改定2版＞

LangEdge,Inc.
有限会社 ラング・エッジ

プログラマ/取締役
宮地直人（miyachi@langedge.jp）

2020年9月29日

LE miyachi

人と言うエンティティ（実体）は様々なアイデンティティとIDを持つ。

AITC クラウド部会 メンバー

マイナンバー：〇〇〇〇〇〇
氏名：宮地 直人、性別：男
住所：東京都江戸川区〇〇〇
生年月日：19〇〇年2月〇日

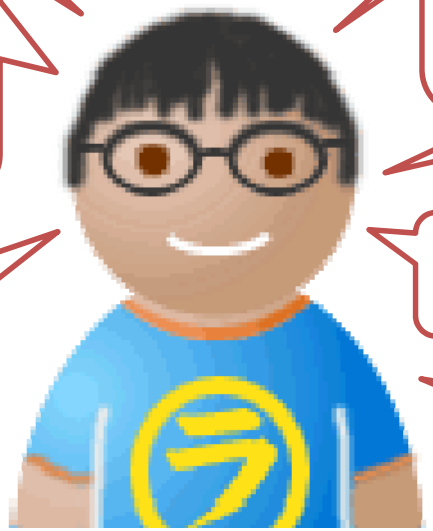
有限会社ラング・エッジ
プログラマ／取締役
社員番号：0002
miyachi@langedge.jp
PKI・PDF系製品開発等
https://www.langedge.jp/

JIIMA 会員

Twitter
@le_miyachi

JT2A / JNSA
電子署名WG所属
標準化 ISO TC 154
http://eswg.jnsa.org/

OpenIDファウンデーション・ジャパン
会員企業
https://www.openid.or.jp/



https://www.slideshare.net/lemiyachi
https://github.com/miyachi/

オープンソース署名＆認証ラボ
OsSAL SAL No.0001
勉強会・オープンソース開発
https://www.ossal.org/

Facebook
nao.miyachi

1. 認証入門：ID・アイデンティティ・認証
2. 認証基本：NIST SP 800-63-3
3. 認証利用：OAuth 2.0・OpenID Connect
4. 認証応用：PKCE・FIDO2

ID とアイデンティティ（Identity）

ID： １つの実体（エンティティ）に付す識別子となる属性情報
Identity：実体（エンティティ）に関する属性情報の集合

- IDとはIdentityの持つ属性情報の１つで識別に利用する。
- １つの実体は、複数のID/Identityを持つことができる。
- 人は実体をIdentity（属性の集合）を通じて認識する。

Authentication/AuthN（認証）：
端末の前にいる実体がサービス側が認識するどのIdentityと紐付いているかの確証を得ること。
Authenticator（認証器/認証コード）：Credential（信用）情報を扱う

Authorization/AuthZ（認可）：
端末の前にいる実体のIdentityがサービス側が提供するリソースにアクセスをする権限を持っているか確認すること。

Access Control＝AuthN＋AutnZ＋Audit [logging]

本人確認(Identity Proofing)と当人確認(認証)

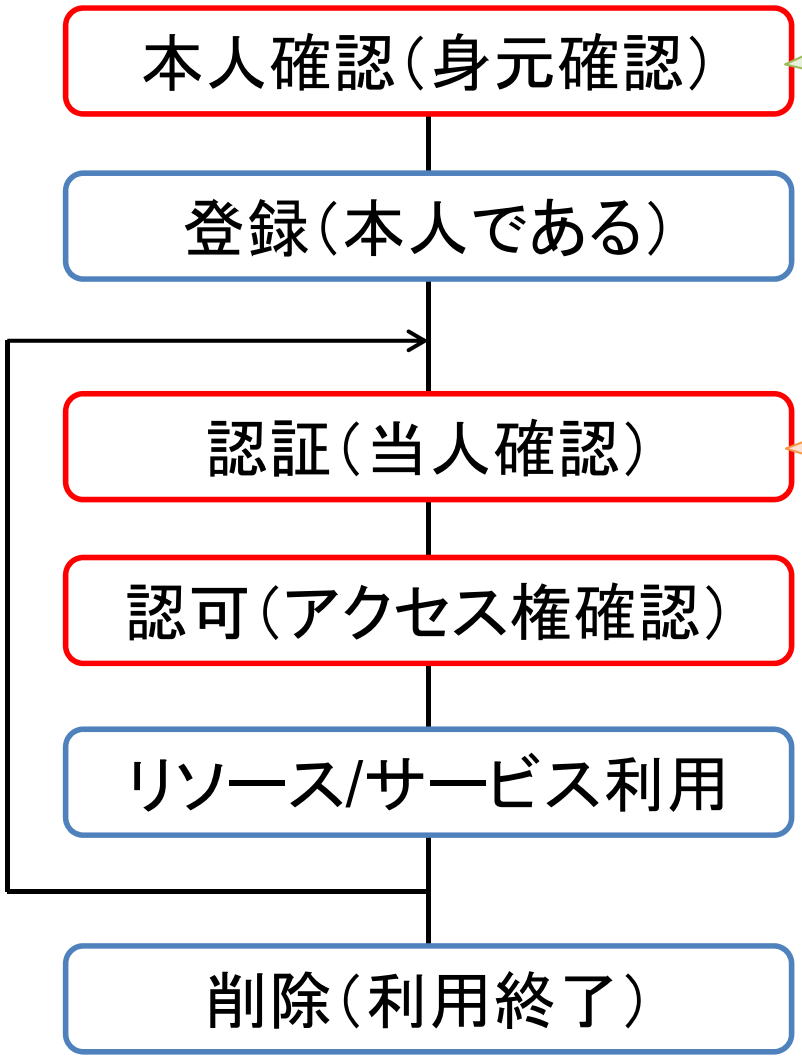
Onbording（登録時）のアカウントID登録時には、
Identity Proofing（本人確認）が必要

- **Identity Proofing：属性情報の確かさを確認する**
  例：メール到達・運転免許書・マイナンバーカード等を利用
- **KYC（Know Your Custorer）も本人確認**
  サービス事業者のための、本人確認手続き（KYC）に関する調査レポート
  https://www.openid.or.jp/news/oidfj_kycwg_report_20200123.pdf
- **本人確認には継続した確認が必要でありコストもかかる**

Ongoing（利用中）のアカウントID利用時には、
Authentication（当人確認:認証）が必要

- **Authentication：ID情報との紐付けを確認する**
  例：ID/パスワード・指紋等生体情報・ICカード等を利用

電子認証利用の流れ



どのような情報で本人確認をしたかで
IDの保証レベルが異なる（IAL）。

例）
高：対面で公的写真付きIDを確認
低：メール到達により実在を確認

どのような情報で当人確認をしたかで
認証の保証レベルが異なる（AAL）。

例）
高：2要素認証により確認
低：4桁固定のPINだけで確認

必要機能を全て１システムで提供することも可能だが、大変なので機能毎に分けたID連携をするようになってきた。

認証画面と認可画面の例

カレンダーサービス **Jorte Cloud** が、**Googleカレンダー** に対するアクセスの認可を求める場合の画面の例。

ID連携（ID Federation）とSSO（Single Sign On）

ID連携：異なるドメイン間にてIDを連携する仕組み。
SSO：異なるドメイン間にて認証セッションを継続する仕組み。

主なSSOの方式＞

1. **ケルベロス認証方式**（同一ドメインSSO内向け）
   - サーバ・クライアント間で認証処理を行う

   例：ActiveDirectory（Windowsドメイン内でのリソースアクセス等）
2. **リバースプロキシ方式**（同一ドメインSSO内向け）
   - 全てのサーバ接続をリバースプロキシ経由として認証処理を行う
3. **エージェント方式**（同一ドメイン内SSO向け）
   - 各サーバに常駐するエージェントソフトが認証処理を行う

本資料の対象

4. **フェデレーション方式**（**複数ドメイン間**を跨いだSSO向け）
   - IDプロバイダ（IdP）へリダイレクト（連携）して認証を行う
   - FacebookやTwitter等のクラウドサービスでの利用が多い
   - 仕様が標準化されており相互運用性が高い（クラウドで普及）

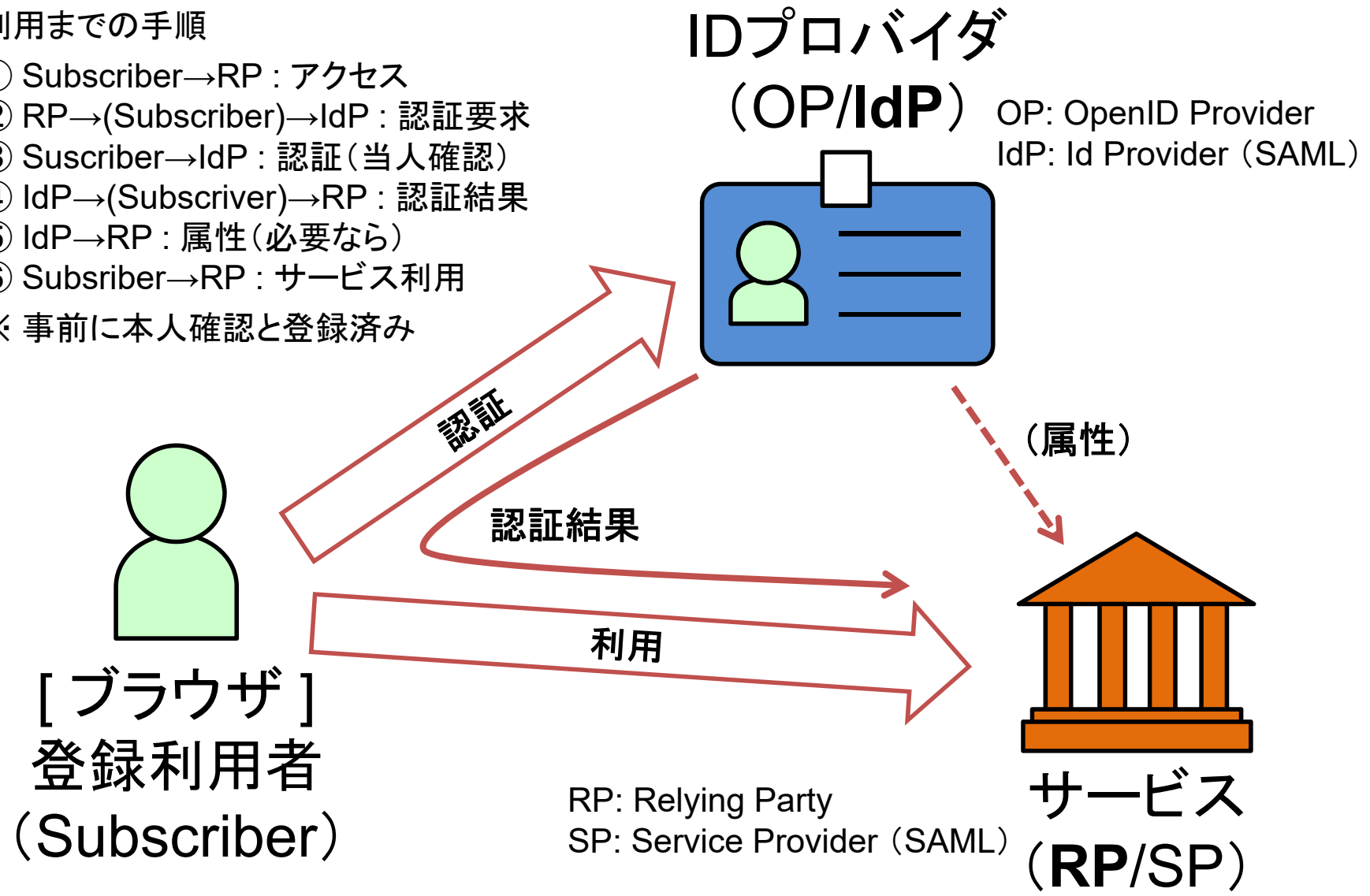
   例：SAML/**OAuth2.0+OpenID Connect**

フェデレーション方式の構成と利用手順

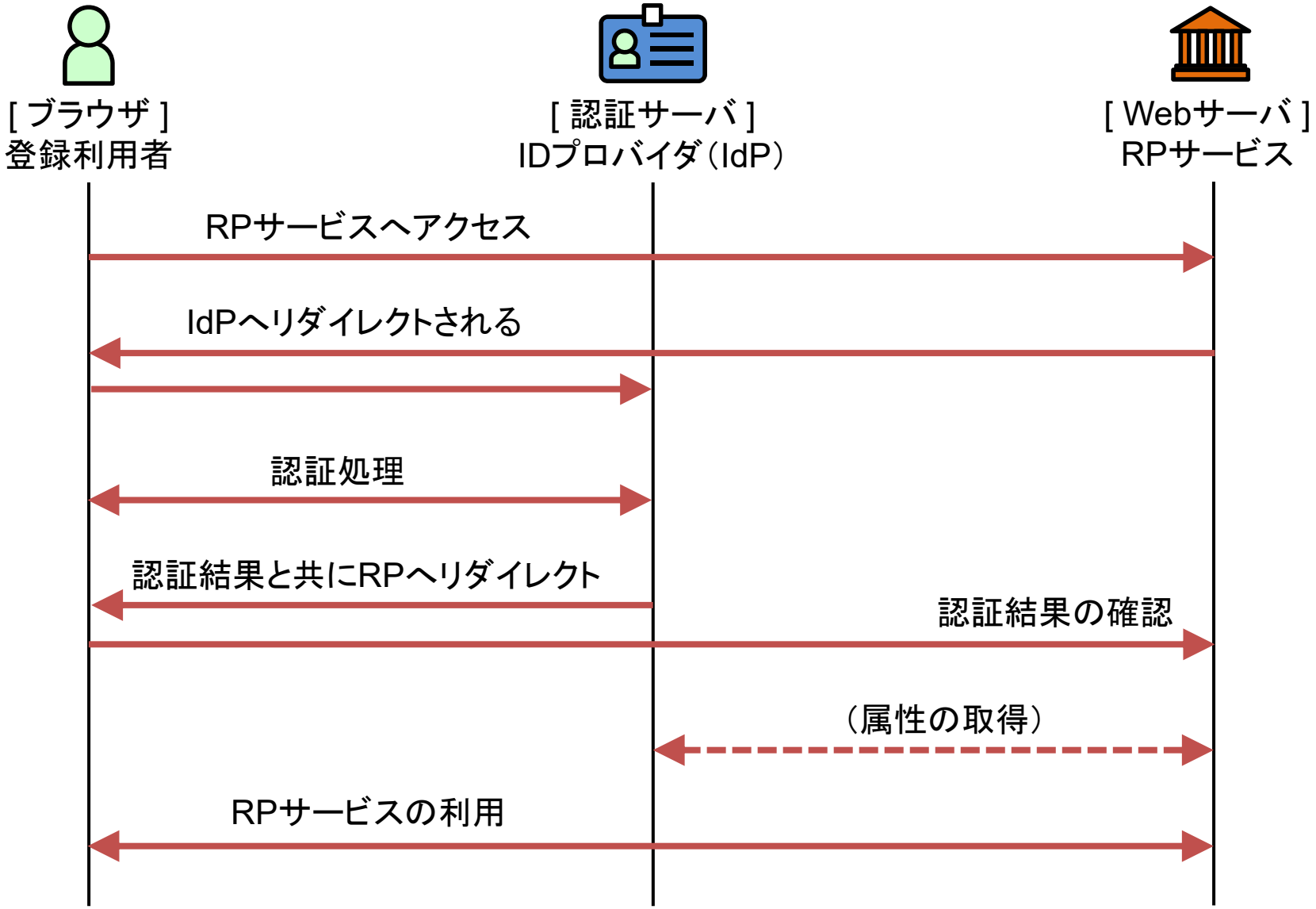
利用までの手順

① Subscriber→RP : アクセス
② RP→(Subscriber)→IdP : 認証要求
③ Suscriber→IdP : 認証（当人確認）
④ IdP→(Subscriver)→RP : 認証結果
⑤ IdP→RP : 属性（必要なら）
⑥ Subsriber→RP : サービス利用

※ 事前に本人確認と登録済み



フェデレーション方式の利用シーケンス



OECD（経済協力開発機構）8原則（参考）

1980年に定められた個人情報保護の基本原則

1．目的明確化の原則（Purpose Specification Principle）
- 個人データの収集目的を明確にし、データ利用は収集目的に合致するべき。
- 個人情報保護法：第15条（利用目的の特定）

2．利用制限の原則（Use Limitation Principle）
- データ主体（個人情報の持ち主）の同意がある場合や法律の規定による場合を除いては、収集したデータを目的以外に利用してはいけない。
- 個人情報保護法：第16条（利用目的による制限）、第23条（第三者提供の制限）

3．収集制限の原則（Collection Limitation Principle）
- 個人データは、適法・公正な手段により、かつ情報主体に通知または同意を得て収集されるべき。
- 個人情報保護法の対応：第17条（適正な取得）

4．データ内容の原則（Data Quality Principle）
- 収集する個人データは、利用目的に沿ったもので、かつ、最新・正確・完全であるべき。
- 個人情報保護法：第19条（データ内容の正確性の確保）

5．安全保護の原則（Security Safeguards Principle）
- 合理的安全保護措置により、紛失・破壊・使用・修正・開示等から保護すべき。
- 個人情報保護法：第20条（安全管理措置）、第21条（従業者の監督）、第22条（委託先の監督）

6．公開の原則（Openness Principle）
- 個人データ収集の実施方針等を公開し、データの存在、利用目的、管理者等を明示するべき。
- 個人情報保護法：第18条（取得に際しての利用目的の通知等）、第24条（保有個人データに関する事項の公表等）

7．個人参加の原則（Individual Participation Principle）
- 自己(データ主体)に関するデータの所在及び内容を確認させ、または異議申立を保証するべき。
- 個人情報保護法：第25条（開示）、第26条（訂正等）、第27条（利用停止等）

8．責任の原則（Accountability Principle）
- 個人データの管理者は諸原則実施の責任を有する。
- 個人情報保護法：第31条（苦情処理）

アイデンティティ 7原則（参考）

2005年に元MSのキム・キャメロン氏が作成した基本原則、以後マイクロソフトのアイデンティティ基本原則となる

原則1. ユーザーによる制御と同意

- アイデンティティ・システムは、ユーザーの同意がなければユーザーを識別する情報を開示すべきではない。

原則2. 限定された用途で最低限の公開

- 最も安定し、長期にわたって使用できるソリューションとは、開示するアイデンティティ情報を最小限にし、情報へのアクセスを適切に制限するソリューションである。

原則3. 正当な関係者のみへの情報開示

- アイデンティティ・システムは、特定の状況において識別情報を必要とし、かつ入手できる正当な権利を持つ関係者のみに対して情報を開示するように設計されなければならない。

原則4. 方向付けられたアイデンティティ

- アイデンティティ・システムは、公に使用する「全方位的」な識別子とプライベートで使用する「特定の方向性」を持った識別子の両方をサポートしなければならない。このことにより公共性を維持しながら不必要に関連付けの公開を防止できる。

原則5. 複数のアイデンティティ・プロバイダと技術の相互運用性

- アイデンティティ・システムは、複数のアイデンティティ・プロバイダによって実行される複数のアイデンティティ技術の相互運用性を保持しなければならない。

原則6. 人間の統合

- アイデンティティ・システムは、利用者たるユーザーを分散システムの1つのコンポーネントとして定義しなければならない。明確なマンマシン・インターフェイスを策定してユーザーを分散システムに統合し、アイデンティティを保護しなければならない。

原則7. シンプルで一貫性のあるユーザー・エクスペリエンス

- アイデンティティ・システムは、さまざまな状況下でのアイデンティティ・コンテキストの分離を可能にしつつも、一貫性のあるユーザーとテクノロジのインターフェイスを提供しなければならない。

1. 認証入門：ID・アイデンティティ・認証
2. **認証基本：NIST SP 800-63-3**
3. 認証利用：OAuth 2.0・OpenID Connect
4. 認証応用：PKCE・FIDO2

NIST SP 800-63-3 : Digital Identity Guidelines

2017年6月にNIST（米国標準技術研究所）からFinal版が公開。
米国政府機関向けのデジタルID実装ガイドライン。
https://pages.nist.gov/800-63-3/

SP 800-63-3	**Digital Identity Guidelines** 「デジタルIDガイドライン」 全体の概要
SP 800-63A	**Enrollment and Identity Proofing** 「登録と身元情報の検証」 本人確認のガイドライン IAL（Identity Assurance Level）の定義
SP 800-63B	**Authentication and Lifecycle Management** 「認証とライフサイクル管理」 当人確認のガイドライン AAL（Authenticator Assurance Level）の定義
SP 800-63C	**Federation and Assertions** 「連携とアサーション」 連携時の認証/認可/属性情報 FAL（Federation Assurance Level）の定義

日本でも「行政手続におけるオンラインによる本人確認の手法に関するガイドライン」等が参照。
https://cio.go.jp/sites/default/files/uploads/documents/hyoujun_guideline_honninkakunin_20190225.pdf

保証レベル（Assurance Level）

SP 800-63-2 では**１種類**の**LoA**で４レベルを規定。

- 各項目のレベルを合わせる必要があると言う問題があった。
  LoA：Level of Assurance

SP 800-63-3 では**３種類**の**AL**で各３レベルを規定。

- 用途に合わせて各項目のレベルの組み合わせが可能になった。

A) 本人確認保証レベル：**IAL**（Identity Assurance Level）
B) 当人認証保証レベル：**AAL**（Authenticator Assurance Level）
C) 連携情報保証レベル：**FAL**（Federation Assurance Level）

- ✓ レベル１は、最低限（minimum）の要件。
- ✓ レベルを合わせる必要は無く、匿名（IAL1）でAAL2でも良い。
- ✓ デジタルIDシステムを設計するなら特にIALとAALは重要。
- ✓ FALは現実的にはレベル１かせいぜいレベル２止まり。
- ✓ レベル３は米国の機密情報へのアクセスレベル（非現実的）。

IAL（Identity Assurance Level）

ユーザが申請者（Applicant）として、新規登録（SignUp）する際に、CSP（Credential Service Provider）が行う、本人確認（Identity Proofing）の厳密さや強度、を示す。

IAL.1	本人確認不要、自己申告での登録でよい。
IAL.2	サービス内容により識別に用いられる属性をリモートまたは対面で確認する必要あり。
IAL.3	識別に用いられる属性を対面で確認する必要があり、確認書類の検証担当者は有資格者の必要あり。

AAL（Authenticator Assurance Level）

登録済みユーザ（Claimant）が、ログインする際の認証プロセス（単要素認証or多要素認証、認証手段）の強度、を示す。
Claimant は認証されることで加入ユーザ（Subscriber）となる。

AAL.1	単要素認証でよい。
AAL.2	2要素認証が必要、2要素目の認証手段はソフトウェアベースのものでよい。
AAL.3	2要素認証が必要、かつ2要素目の認証手段はハードウェアを用いたもの（ハードウェアトークン等）が必要。

認証要素と段階

認証要素の種類（3種類のみ）：

知識	当人しか知り得ない情報（PIN番号・パスフレーズ 等）
所有	当人しか所有していない物に依存（ICカード・トークン 等）
生体	当人の生体としての情報（指紋・光彩・顔認識 等）

２要素認証：

3つの要素の種類のうちの2つの要素による認証。
例：知識＋所有 や 知識＋生体 の組み合わせが多い。

２段階認証：

認証プロセスを２段階に分けて行う。
例：最初にパスワード認証を行い、次に秘密の質問を行う。

※ NIST SP 800-63B では**2段階認証の価値を認めていない。**
※ NIST SP 800-63B ではパスフレーズの定期的変更は非推奨。
→ ただしパスフレーズが危殆化した場合には変更を強制する。

Authenticator（認証コード）の種類

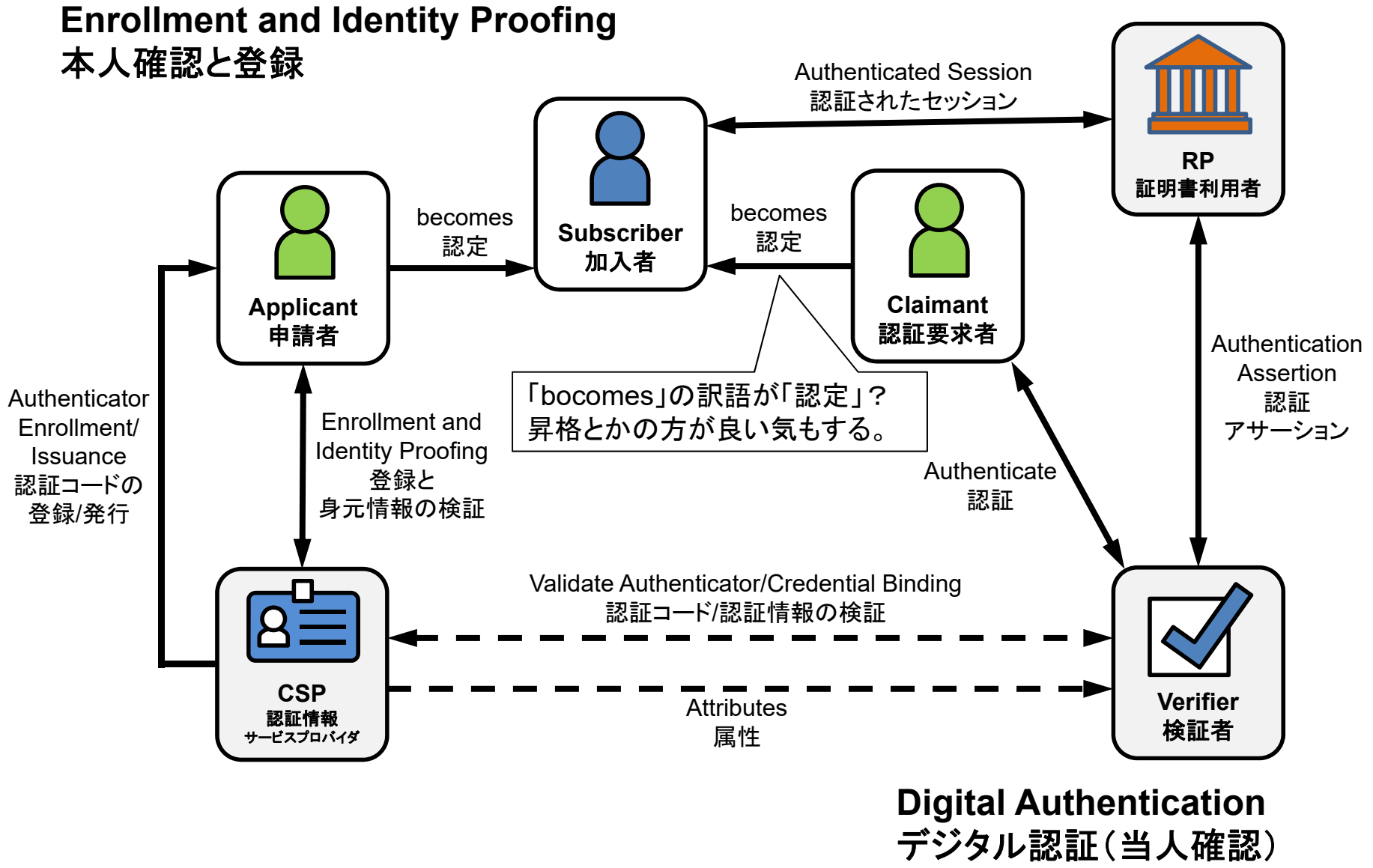
No	名称	要素種別	鍵所持証明	ハードウェア
1	**Memorized Secret** **記憶シークレット**（パスワード等）	知識	認められない	認められない
2	**Look-up Secret** **ルックアップシークレット**（乱数表等）	所有	認められない	認められない
3	**Out of Band Device** **経路外デバイス**（SMS等、メール/VoIPは不可）	所有	認められない	認められない
4	**SF（Single Factor）OTP Device** **単一要素OTPデバイス（アプリ）**	所有	認められない	△内容による
5	**MF（Multi Factor）OTP Device** **多要素OTPデバイス（アプリ）**	所有 +知識/生体	認められない	△内容による
6	**SF Cryptographic Software** **単一要素暗号ソフトウェア**（クライアント証明書等）	所有	○認められる	認められない
7	**SF Cryptographic Device** **単一要素暗号デバイス**（USBドングル等）	所有	○認められる	○認められる
8	**MF Cryptographic Software** **多要素暗号ソフトウェア**	所有 +知識/生体	○認められる	認められない
9	**MF Cryptographic Device** **多要素暗号デバイス**（FIDO/ICカード等）	所有 +知識/生体	○認められる	○認められる

FAL（Federation Assurance Level）

IDトークンやSAML Assertion等、Assertion（認証情報・属性・認可情報）のフォーマットやデータやり取りの仕方の強度、を示す。

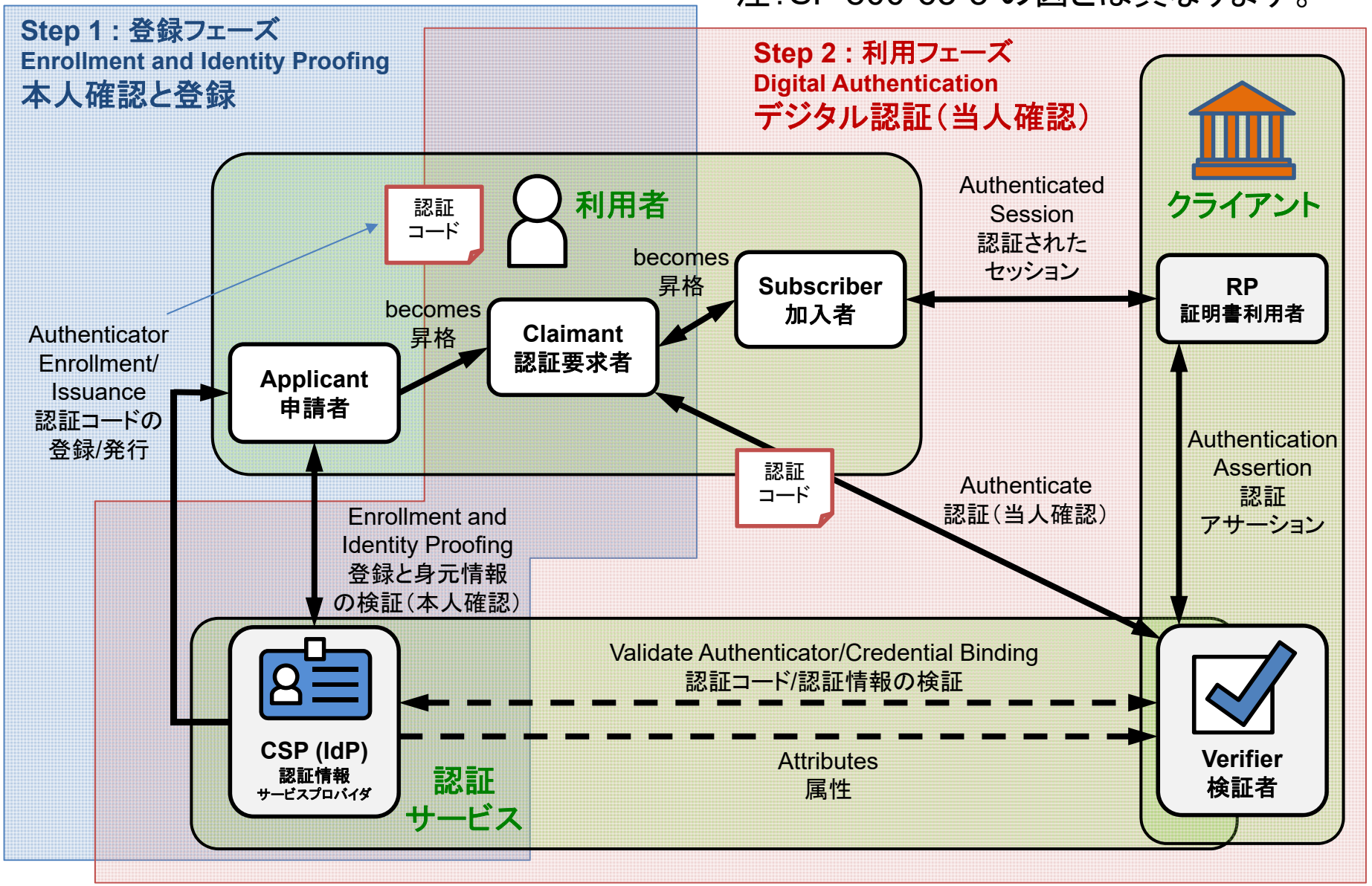
FAL.1	Assertion（RPに送るIdPでの認証結果データ）への署名が必要。
FAL.2	レベル1に加えて、対象RPのみが復号可能な暗号化が必要。つまり署名＋暗号化が必要。
FAL.3	レベル2に加えて、Holder-of-Key Assertionの利用（ユーザごとの鍵とIdPが発行したAssertionを紐づけてRPに送り、RPはユーザがそのAssertionに紐づいた鍵を持っているか（ユーザの正当性）を確認）が必要。

SP 800-63-3 デジタルIDモデル図

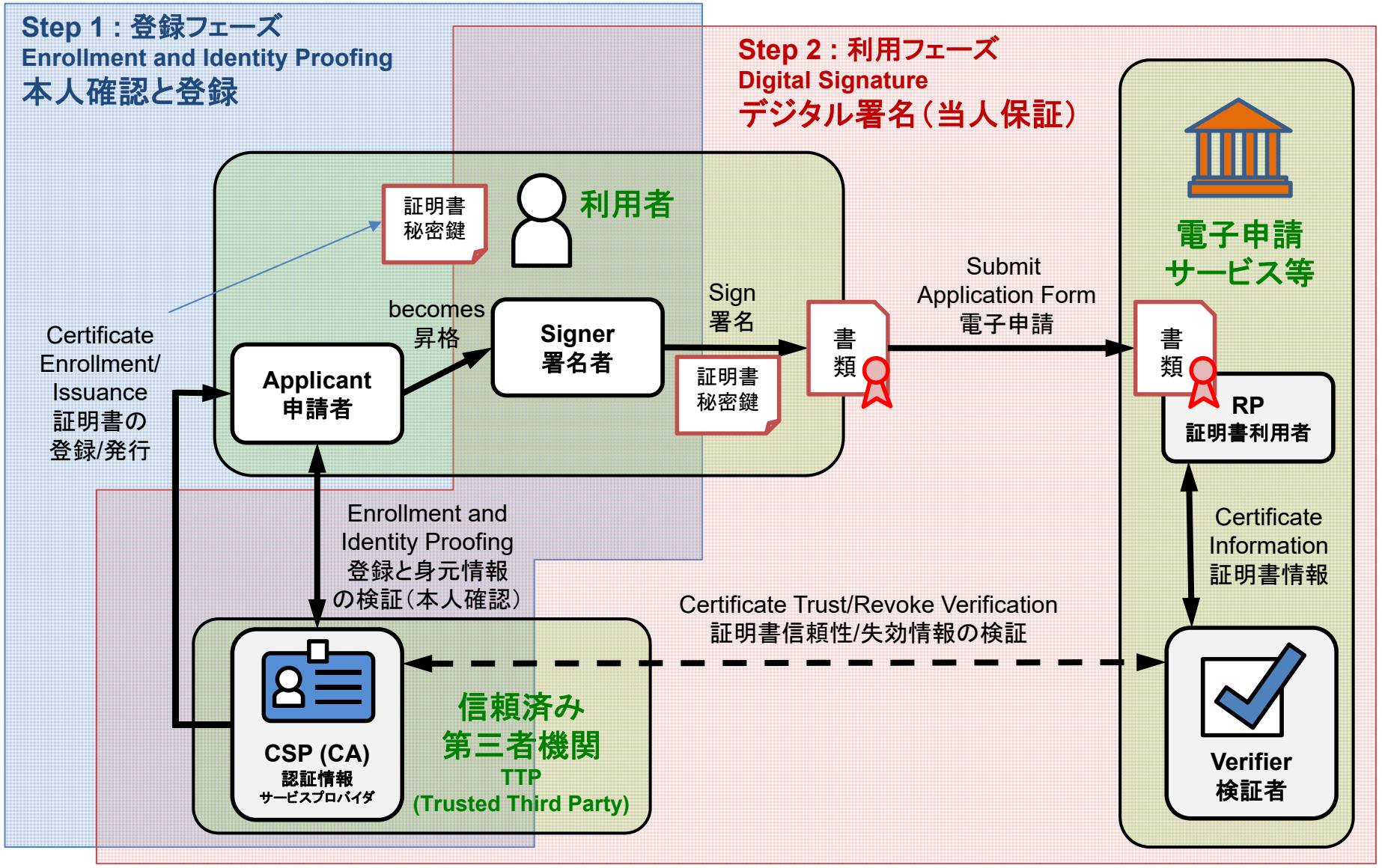


デジタルIDモデル図（改）

注：SP 800-63-3 の図とは異なります。



デジタル署名モデル図（参考）



※ Step1の登録フェーズはデジタルIDモデルとほぼ同じ内容。

1. 認証入門：ID・アイデンティティ・認証
2. 認証基本：NIST SP 800-63-3
3. **認証利用：OAuth 2.0・OpenID Connect**
4. 認証応用：PKCE・FIDO2

SAMLとOAuth2.0

SAML（Security Assertion Markup Language）

http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0.html
策定：OASIS　v2.0 を2005年発行
メッセージ送受信：**SOAP/HTTP**
通信言語：**XML**ベース
主な実装：Shibboleth（学術系で広く利用、日本では**学認**が利用）
※「SAML is dead!」と言われながらまだまだ使われている。

OAuth 2.0 Authorization Framework（RFC 6749）

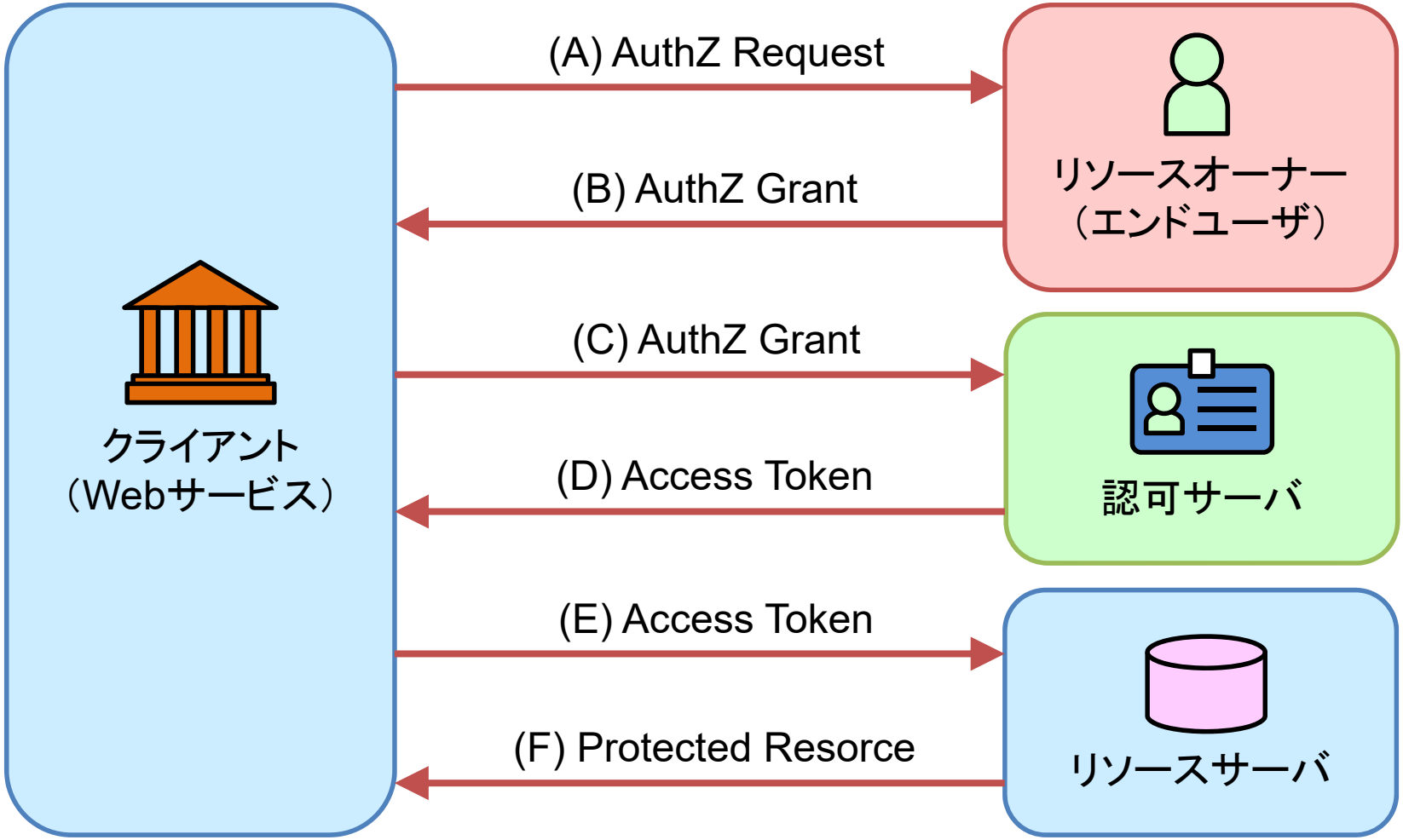
https://tools.ietf.org/html/rfc6749
https://openid-foundation-japan.github.io/rfc6749.ja.html [和訳]
策定：IETF（インターネットコミュニティ）2012年発行
メッセージ送受信：**HTTPS**（**RESTful**）
通信言語：**JSON**ベース
利用：実装のハードルが低くクラウドサービスで広く利用されている

OAuth 2.0 登場人物(ロール)

OAuth 2.0 のロールは以下4種類:

1. **Resource Owner**(またはエンドユーザ)
   - エンティティが人間の場合にはエンドユーザである。
2. **Resource Server**(リソースサーバ)
   - アクセストークンを受理し、保護されたリソースへのアクセスのレスポンスを返す。クライアントと同じ場合も多い。
3. **Client**(クライアント)
   - リソースオーナーの認可を受けて、保護されたリソースへのアクセスを行うアプリケーション(Webサービス等)。
4. **Authorization Server**(認可サーバ)
   - リソースオーナーの認可を確認してアクセストークンを発行するサーバ。

OAuth 2.0 登場人物(ロール)の関係



AuthZ Grant：リソースにアクセスするために使用するオーナーの認可を示すクレデンシャル。
Access Token：リソースにアクセスする為に認可サーバが発行するチケット(トークン)。

OAuth 2.0 フロー種別

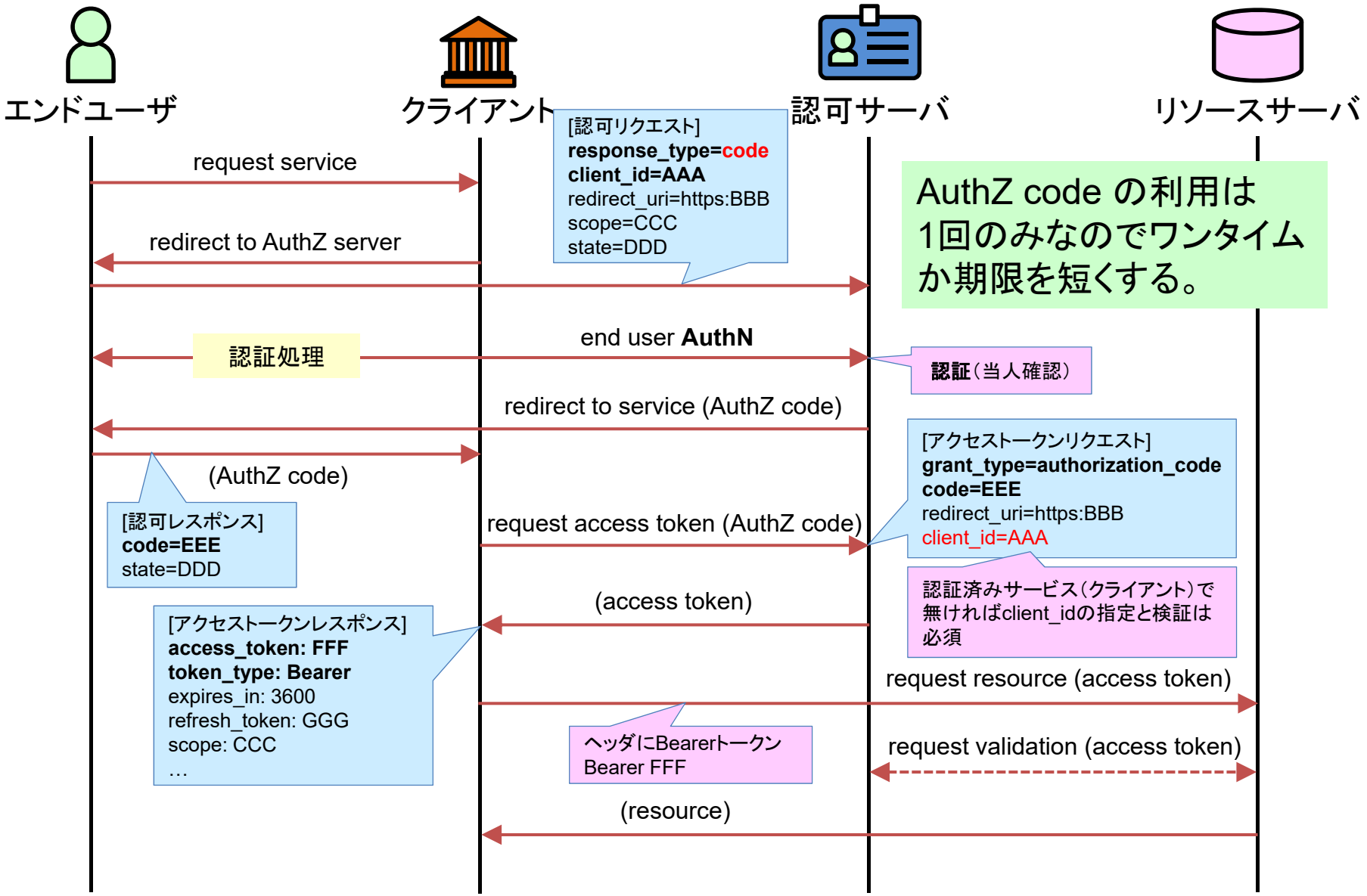
OAuth 2.0 の主なフローは以下4種類:

1. **Authorization Code Flow**（基本:Webサービスでの推奨フロー）
   - 認可リクエスト時:response_type=code
   - アクセストークンリクエスト時:grant_type=authorization_code
   - 認証トークンからWebサービスが直接アクセストークンを取得
2. **Implicit Flow**（簡略化されたフローで時々使われている）
   - 認可リクエスト時:response_type=token
   - 認可時に直接アクセストークンを取得するのでgrant_typeは定義されていない
   - クライアントアプリがアクセストークンを取得するリスクがある
3. **Resource Owner Password Credentials Flow**（説明省略）
   - アクセストークンリクエスト時:grant_type=password
   - アクセストークンリクエスト時に認証情報を付けて要求する
   - オーナー保有デバイス等からアクセストークンリクエストして直接トークンを取得
4. **Client Credentials Flow**（説明省略）
   - アクセストークンリクエスト時: grant_type=client_credentials
   - オーナーが信頼関係にありユーザ認証しなくても良い時のフロー

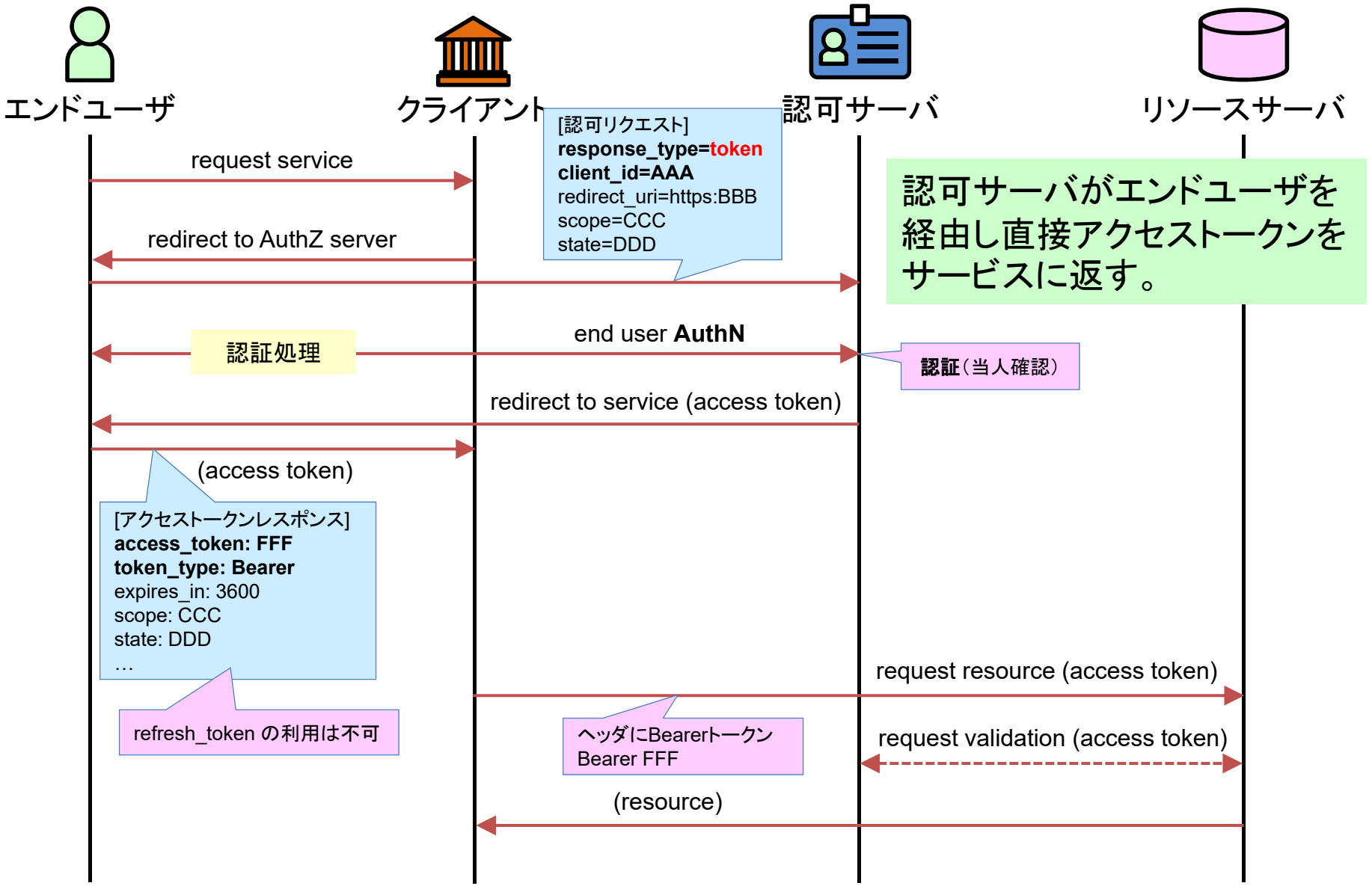
Webサービスでは Authorization Code Flow を使うべき。

参考 https://www.ipa.go.jp/security/awareness/vendor/programmingv2/contents/709.html

Authorization Code Flow のシーケンス



Implicit Flow のシーケンス



OAuth 2.0 のパラメーター

1. **scope**（アクセス権限の範囲指定）
   - アクセスする範囲を示すパラメーターで省略時はデフォルト値を使うかエラーとすべき。エンドユーザに範囲を提示して確認することが必要。独自に設計する必要がある。
     例：'Mail.Read'
2. **state**（なりすまし対策）
   - CSRF（クロスサイトリクエストフォージェリ）対策として利用する、サービス（クライアント）が指定する乱数パラメーター。
   - 必須パラメーターではないが指定すべき。
3. **refresh_token**（アクセストークンの更新）
   - アクセストークン有効期間内に同じ認可グラントに対して、新しいアクセストークンを取得する為の属性（Refresh Token Flow）。
   - 事実上アクセストークンの有効期間を延長することが可能。

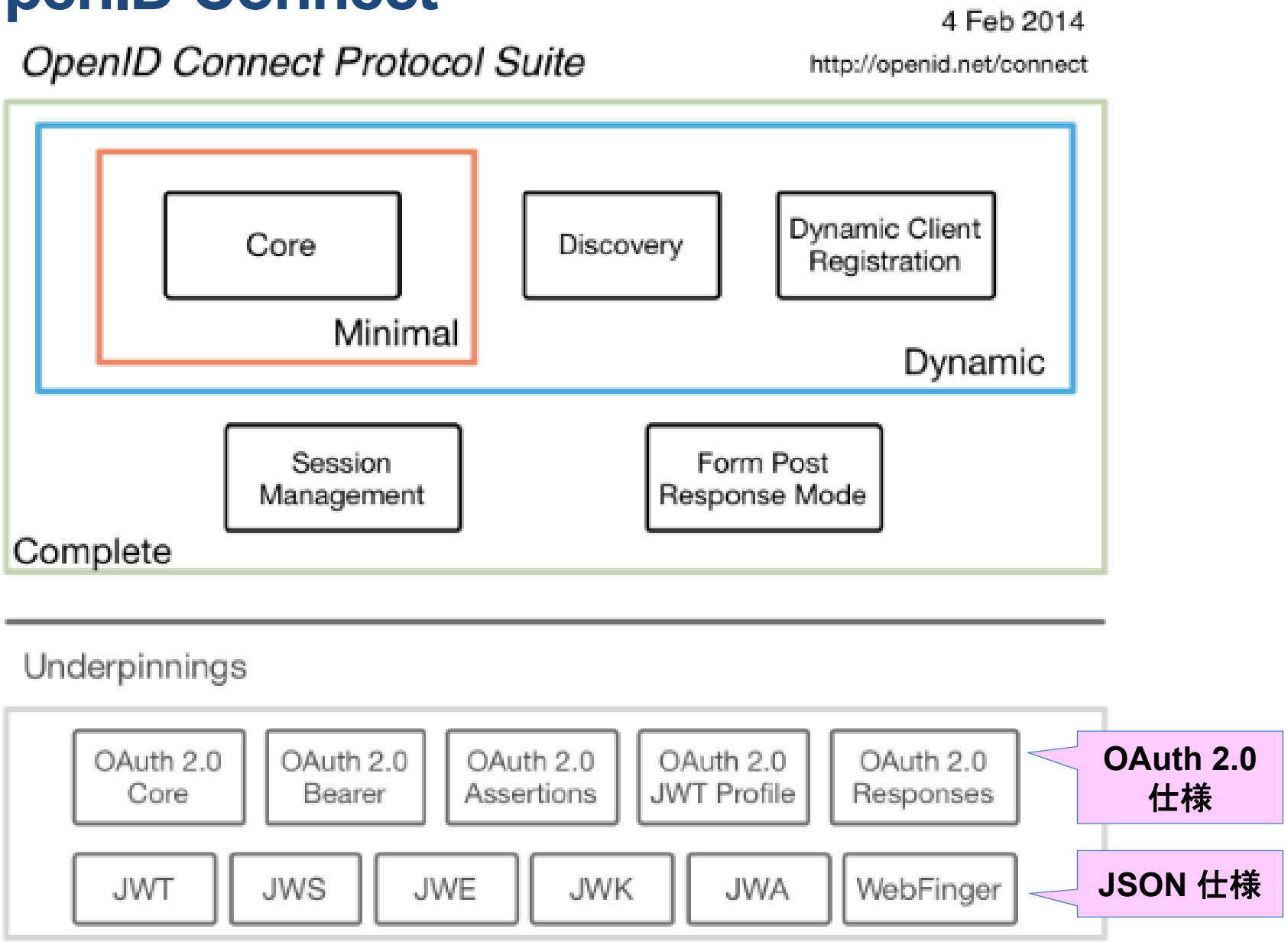
OAuth 2.0 ＋ OpenID Connect

OAuth は「Authorization Framework（認可フレームワーク）」であり認証フレームワークを含まない。この為に OAuth だけで認証も行った場合（OAuth認証と呼ばれる：後述）にセキュリティリスクを生じる可能性がある。なおSAMLは認証認可に対応している。

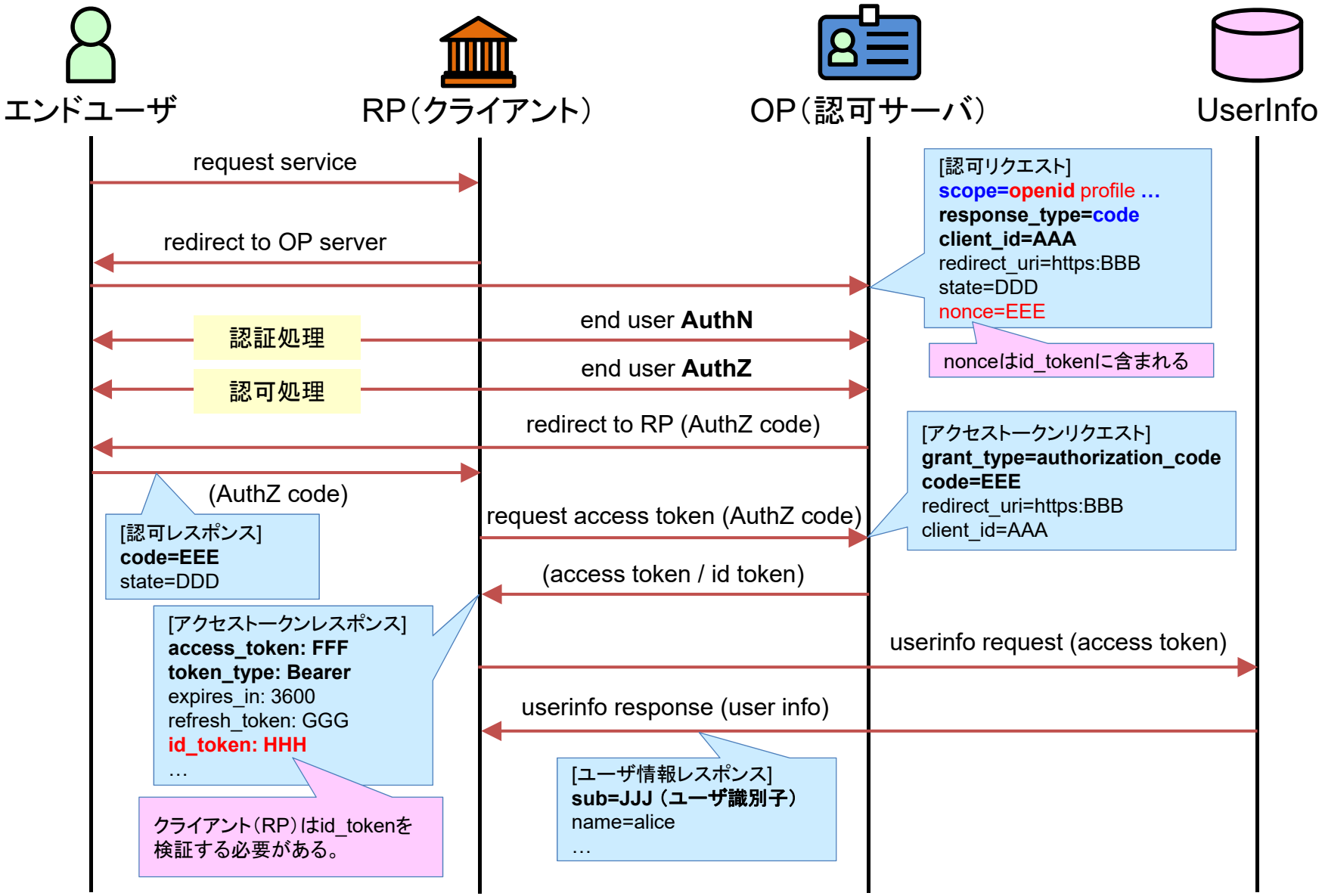
OpenID Connect は、OAuth をベースに「アイデンティティ層」を拡張した、2014年発行の OpenID 最新仕様。OAuth 2.0 をベースに新規策定された為 OpenID 1.0/2.0 との互換性は無い。
策定：オープンIDファウンデーションが2014年に1.0を発行
https://openid.net/connect/
http://openid-foundation-japan.github.io/openid-connect-core-1_0.ja.html [和訳]

OAuth 2.0	認可層
OpenID Connect 1.0	アイデンティティ層
FIDO2（後述）	認証層

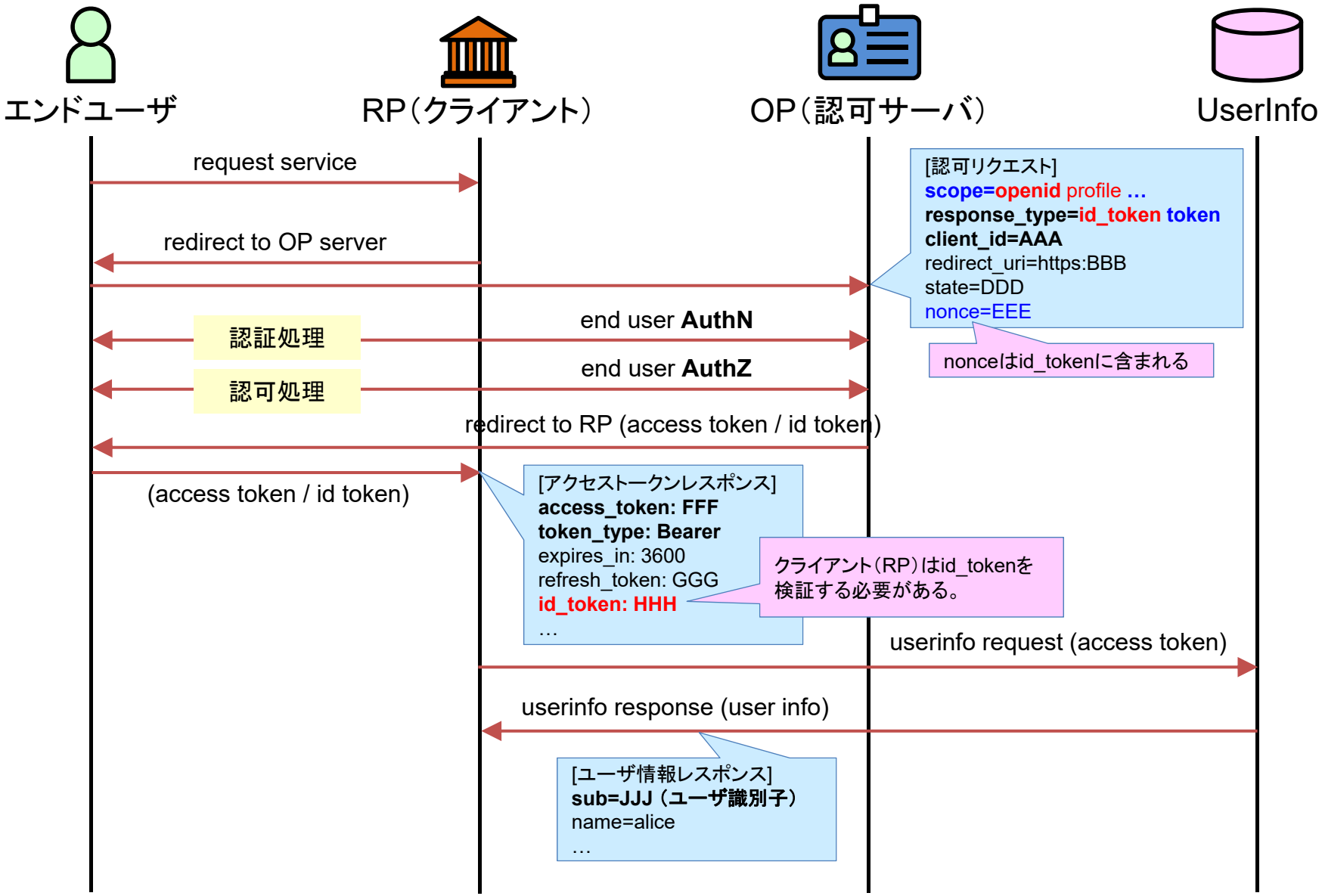
OpenID Connect



Authorization Code Flow（+ OpenID Connect）



Implicit Flow（+ OpenID Connect）



IDトークン付きアクセストークンレスポンス例

OPから取得するアクセストークンレスポンスはJSON形式で返される。

OAuth 2.0

OpenID Connect の、id_token部は**JWT**形式にパッケージされて返される。


```
{
  "access_token": "SlAV32hkKG",
  "token_type": "Bearer",
  "refresh_token": "8xLOxBtZp8",
  "expires_in": 3600,
  "id_token": "eyJhbGciOiJSUzI1NiIsImtpZCI6IjFlOWdkazcifQ.ewogImlzc
    yI6ICJodHRwOi8vc2VydmVyLmV4YW1wbGUuY29tIiwKICJzdWIiOiAiMjQ4Mjg5
    NzYxMDAxIiwKICJhdWQiOiAiczZCaGRSa3FOMyIsCiAibm9uY2UiOiAibiOwUzZ
    fV3pBMk1qIiwKICJleHAiOiAxMzExMjgxOTcwLAogImlhdCI6IDEzMTEyODA5Nz
    AKfQ.ggW8hZ1EuVLuxNuuIJKX_V8a_OMXzROEHR9R6jgdqrOOF4daGU96Sr_P6q
    Jp6IcmD3HP99Obi1PRs-cwh3LO-p146waJ8IhehcwL7F09JdijmBqkvPeB2T9CJ
    NqeGpe-gccMg4vfKjkM8FcGvnzZUN4_KSPOaAp1tOJ1zZwgjxqGByKHiOtX7Tpd
    QyHE5lcMiKPXfEIQILVq0pc_E2DzL7emopWoaoZTF_m0_N0YzFC6g6EJbOEoRoS
    K5hoDalrcvRYLSrQAZZKflyuVCyixEoV9GfNQC3_osjzw2PAithfubEEBLuVVk4
    XUVrWOLrLl0nx7RkKU8NXNHq-rvKMzqg"
}
```

JWT（Json Web Token）

JWT はURL引数のトークンに使えるJSONベース仕様で RFC 7519 定義。
※ 署名の **JWS**（Json Web Signature）、暗号化の **JWE**（Json Web Encryption）をベースとしている。

Header/Payload/Cryptoの 3パートを**Base64化**して **“.” で接続**する。

Part1. **Header Input** ： 署名アルゴリズム等の属性をJSON記述
Part2. **Payload Input** ： 署名対象のJSON記述
Part3. **Crypto Output** ： 署名値そのもの（バイナリ値）

eyJ0eXAiOiJKV1QiLA0KICJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJqb2UiLA0KICJleHAi
OjEzMDA4MTkzODAsDQogImh0dHA6Ly9leGFtcGxlLmNvbS9pc19yb290Ijp0cnV
lfQ.dBjftJeZ4CVP-mB92K27uhbUJU1p1r_wW1gFWFOEjXk

JWTのサンプル

ヘッダ（Header）部 Base64展開	署名対象（Payload）部 Base64展開
{ “typ”: “JWT”, “alg”: “HS256” }	{ “iss”: ”oe”, “exp”: 1300819380, “http://example.com/is_root”: true }

“alg”:”HS256” = HMAC-SHA256
“alg”:”RS256” = RSA-SHA256

ID Token（認証情報）

ID Token は発行元であるOPの署名が付与されているJWTデータ。
JWS署名をOP公開鍵で検証することでOP発行を保証している。
ID Token のJWTペイロード部のサンプルを以下に示す。

```
{
  "iss": "https://server.example.com",
  "sub": "24400320",
  "aud": "s6BhdRkqt3",
  "exp": 1311281970,
  "iat": 1311280970,
  "auth_time": 1311280969,
  "nonce": "n-0S6_WzA2Mj",
}
```

Issuer （発行元:OP）
Subject ID （ユーザ識別子）
Audience （RPのclient_id）
Expire （有効期限）
発行時刻
認証時刻
セッションとid_tokenを紐付

nonce はリプレイアタック対策
必須では無いが使いましょう

Issuer ：**OP**のこと（IdP：IDプロバイダ）
Audience ：**RP**のこと（クライアント：サービス等）
Subject ：**エンドユーザ**のこと（ユーザID：識別子）

ID Token：**OPがRPの為にエンドユーザ認証情報を提供**（非改ざんと発行元を保証）

属性情報とUserInfoエンドポイント

認可リクエスト時の **scope** で取得する**属性情報群（Claims）を指定**。
認証済みのクライアントに対して属性情報群（Claims）を返す。
OpenID Connect では属性取得の **UserInfo** 仕様も追加された。

属性情報を取得する方法は以下の2種類：

1. ID Token に含めて返す。
2. UserInfo エンドポイントのリクエストに返す。

UserInfo エンドポイント：

A) OP/リソースサーバが提供するRESTfulなAPI。
B) RPは取得したアクセストークンを用いて属性情報を取得。
C) 認可リクエスト時に指定されたscopeの属性情報を返す。
　指定例：**scope=openid profile email phone**
D) 取得した属性情報は通常JSON形式で返される。

標準定義の scope と Claims

scope	Claim（返される属性情報）	属性概要
---	sub	Subject - Issuer における End-User の識別子
profile	name	End-User の表示用フルネーム．肩書きや称号 (suffix) 等が含まれることもある.
	family_name	End-User の姓 (surname / last name).
	given_name	End-User の名 (given name / first name).
	middle_name	End-User の middle name.
	nickname	End-User のニックネーム．これは given_name と同じこともあれば異なることもある.
	preferred_username	janedoe や j.doe といった, End-User の簡略名.
	profile	End-User のプロフィールページの URL.
	picture	End-User のプロフィール画像の URL.
	website	End-User の Web ページやブログの URL.
	gender	End-User の性別．本仕様では female, male を定義する.
	birthdate	End-User の誕生日. ISO 8601:2004 [ISO8601-2004] YYYY-MM-DD 形式で表現される.
	zoneinfo	End-User のタイムゾーンを示す zoneinfo [zoneinfo] に登録された文字列.
	locale	End-User の locale. BCP47 [RFC5646] 言語タグ表現.
	updated_at	End-User に関する情報の最終更新日時.
email	email	End-User の Email アドレス. RFC 5322 [RFC5322] のシンタックスに従うこと.
	email_verified	End-User の Email アドレスが検証済であれば true, さもなければ false.
phone	phone_number	End-User の電話番号. この Claim のフォーマットとしては E.164 [E.164] を推奨する.
	phone_number_verified	End-User の電話番号が検証済であれば ture, さもなければ false.
address	address	End-User の郵送先住所. JSON Object で表記.

※ 独自の scope と Claim の組み合わせも独自に定義が可能。　※ OAuth 2.0 の scope パラメータの指定は全て独自定義だった。

PPID と sub Claim（ユーザ識別子）

PPID：Pairwise Pseudonymous Identifier （対の仮名識別子）

※ PPIDの対語は、Public Identifier（共通識別子）

- Entity（ユーザ）識別子として、あるRP（サービス）に対してのみ提供するID。
- 他のRPに対して、同じPPIDをそのEntityと関連付けて提供してはいけない。
- PPID採用：Facebook/LINE、共通ID採用：Google/Twitter/Fahoo

※ **名寄せ対策**（と言っても実際にはセキュリティ向上の為に導入されている）

sub：Subject ID（ユーザ識別子）

メールアドレスが乗っ取られた場合の影響が大きい

- 共通IDとして sub = email を利用することは、ID連携では推奨されない。
- パーマメントな（耐久性がある）ID（ユーザ識別子）の利用をする。
- sub Claim は口頭での名寄せの可能性があり、ユーザには提示しない方が安全。

OpenID Connect でのPPIDの生成仕様：

- ✓ Subject Identifier 値が, OpenID Provider 以外の Party にとって, 可逆であってはならない (MUST NOT)
- ✓ 異なる Sector Identifier 値は, 異なる Subject Identifier 値にならなければならない (MUST)
- ✓ 同じ入力に対して必ず同じ結果となる決定的アルゴリズムでなければならない (MUST)

クライアントのURLを利用することが多い

計算例：

sub = SHA-256 (sector_identifier || local_account_id || salt)

sub = AES-128 (sector_identifier || local_account_id || salt)

認証HTTPS通信のパラメータ指定

認証HTTPS通信での認証パラメータの渡し方は3種類ある：

1. HTTPSのPOSTメソッドでFORM形式渡し（リダイレクト不可）

```
POST /token HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded

tmp_token=aaa
```

POSTメソッドはリダイレクト時にBODY部が失われる。

サーバへのパラメーター指定時はJSON形式では無いことが多い。JSONの場合もある。

2. HTTPSのGETメソッドでURL引数渡し（リダイレクト可）

```
https://server.example.com/token?tmp_token=aaa
```

3. HTTPSのGETメソッドのURLフラグメント部渡し（リダイレクト時）

```
https://server.example.com/token#tmp_token=aaa
```

通常フラグメント部はid名への移動等に利用

GETメソッドのURLフラグメント部はリダイレクトにより**ブラウザ経由**するとサーバには渡らない。フラグメント部の処理はブラウザが行う為で、その点URL引数と異なる。
リダイレクト時のURLフラグメント部にパラメーター指定することで、一旦ブラウザ上の**JavaScriptで取得して確認やURL引数等へ変換する必要がある。**

※ OAuth 2.0 Multiple Response Type Encoding Practices（Implicit Flow Response 等）

1. 認証入門：ID・アイデンティティ・認証
2. 認証基本：NIST SP 800-63-3
3. 認証利用：OAuth 2.0・OpenID Connect
4. **認証応用：PKCE・FIDO2**

Access Token（Bearer Token）の問題

Access Token を Bearer（持参人）Token 仕様でサーバに提示。
利用時HTTP通信でAuhtorizationヘッダにBearerを追加する：

Authorization: Bearer <access_token>

リソースの利用認可としては**「これだけが条件」**となる。
誰かが不正に Access Token を入手して提示しても分からない。

Token置換攻撃（OAuth認証）： Implicit Flow

不正クライアントが OAuth 2.0 の Implict Flow にて、不正に入手した Access Token を使って他RPへ **なりすまし攻撃** することが出来てしまう。

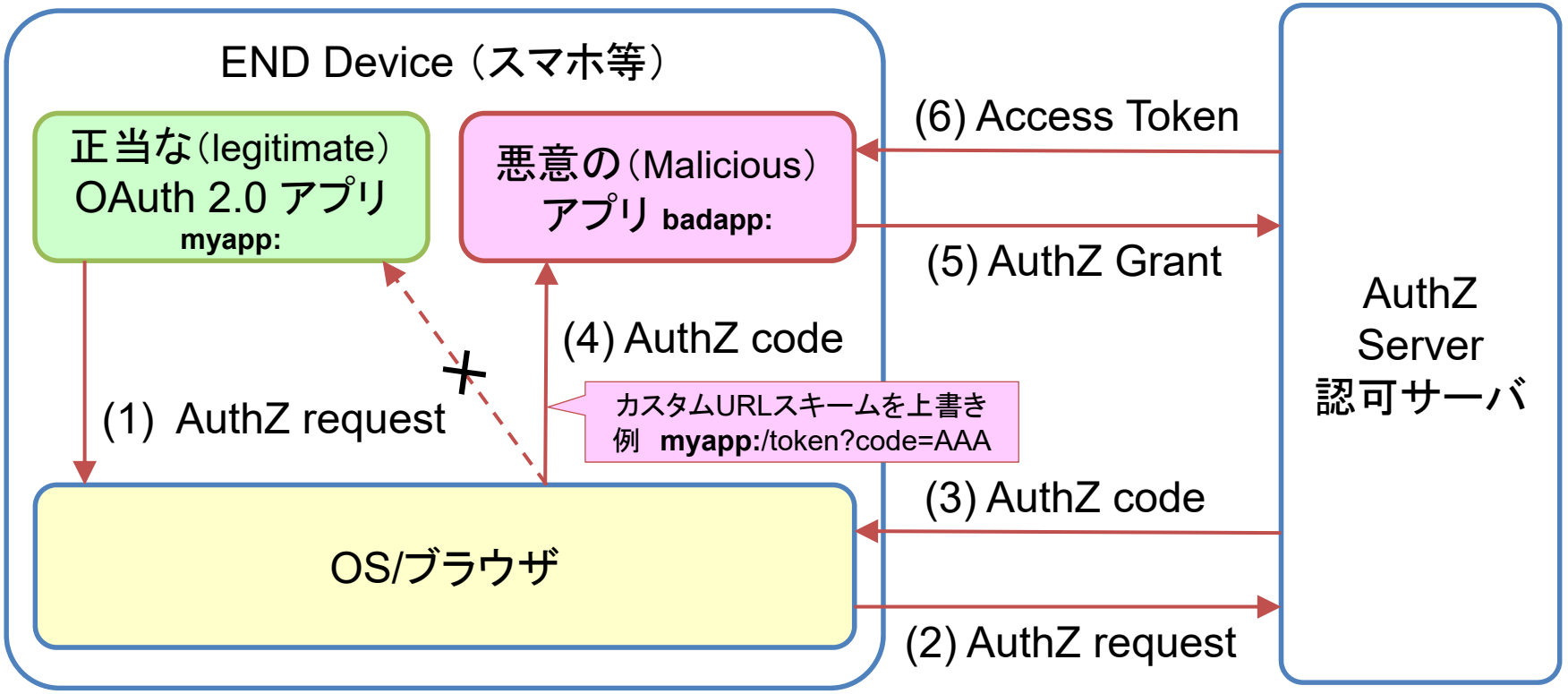
- OpenID Connect を使う（提供先RPを確認）ことも対策の1つ。

認可コード横取り攻撃： Authorization Code Flow

クライアントが**モバイルアプリ**（ネイティブアプリ）の時にリダイレクトを悪用して認可コードを取得して **Access Token の横取り** が出来てしまう。

- モバイルアプリの認証では、内蔵ブラウザで認証処理を行い、カスタムURLスキーム（アプリ用の起動スキーム）にリダイレクトすることで連携をしている。

認可コード横取り攻撃（Authorization Code Interception Attack）



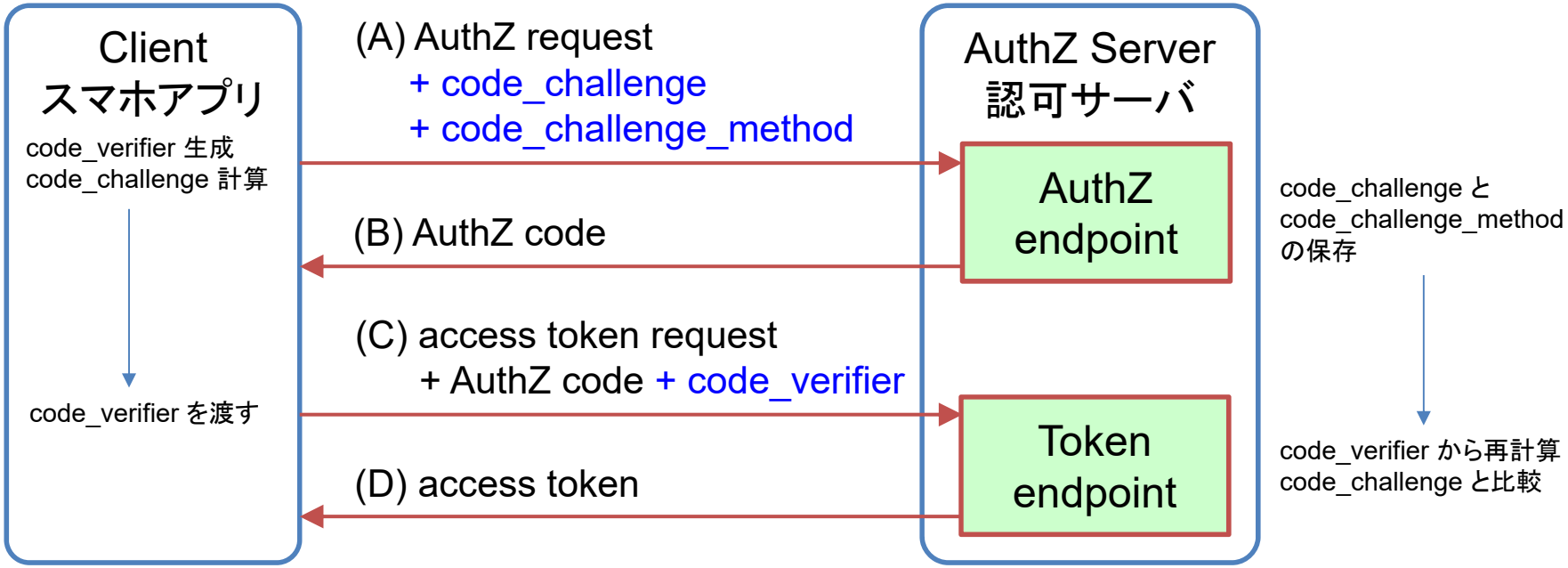
悪意のアプリはカスタムURLスキームを悪用して、
認可サーバから取得した AuthZ code を横取りし、
認可サーバに対し Access Token を要求する攻撃。

OAuth PKCE（ピクシー：認可コード横取り攻撃対策）

RFC 7636：Proof Key for Code Exchange by OAuth Public Clients
https://tools.ietf.org/html/rfc7636

code_verifier：クライアントで作成するランダム値
code_challenge_method：チャレンジ作成方法（例：S256）
code_challenge = code_challenge_method (code_verifier)

SHA256



※ PKCEは、もともとモバイルアプリ向け仕様だったが、最近では**全利用形態での利用が推奨**されて来ている。

Confidential Client と Public Client

クライアント認証に関するクライアントの分類：

Confidential Client ：

- シークレット情報の漏えいを心配する必要がないクライアント。
- クライアントIDとシークレット情報でクライアント認証できる。

例：Webアプリ（Webサービス）

Public Client ：

- シークレット情報の漏えいする可能性があるクライアント。
- シークレット情報によるクライアント認証にはリスクがある。
- **PKCE**の利用が推奨される。

例：ネイティブアプリ（モバイルアプリ）、JavaScriptアプリ

※ redirect_uri パラメータ ：

クライアントから認可サーバに送られる redirect_uri も、クライアント登録時に登録されたものと同一であることを確認すべき。

FIDO（Fast IDentity Online）

FIDO Alliance が策定する**認証標準**仕様

https://fidoalliance.org/　メンバー：Google, Microsoft, Alibaba 等、200社近くが加盟

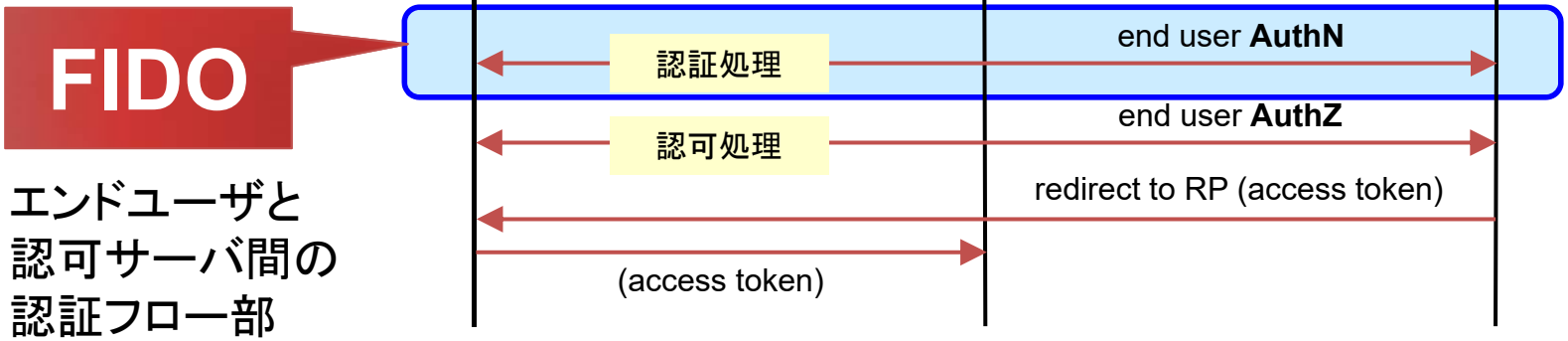
「**FIDO** is an '**authentication**' standard, not an '**identity**' standard.」

FIDOは**認証標準**であってアイデンティティ標準では無い！

FIDOは状況に応じて**認証強度**と**UX**を選択できる**認証処理の標準**。

※ OpenID Connect は認証処理の詳細に関しては**仕様対象外**。

USB Authenticator



エンドユーザと
認可サーバ間の
認証フロー部

FIDO と FIDO2

FIDO（FIDO1）

UAF：Universal Authentication Framework

➢ パスワードレス認証 仕様

生体情報等によりパスワードレス認証を実現するアプリ（スマホ組込み）
※ FIDO2のCTAP1仕様に移行

U2F：Universal Second Factor

➢ 2要素認証 仕様

パスワード認証に加えて別要素で認証する機器（ハード）の仕様
※ 追加認証仕様の為他仕様とは異なる

移行/統合　連携動作　上位互換

FIDO2

CTAP：Client To Authenticator Protocol

➢ 認証器 通信仕様（CTAP2）

認証器との通信プロトコル仕様
外付けの認証器をFIDO2対応にする為の仕様（多要素可）

WebAuthn：Web Authentication

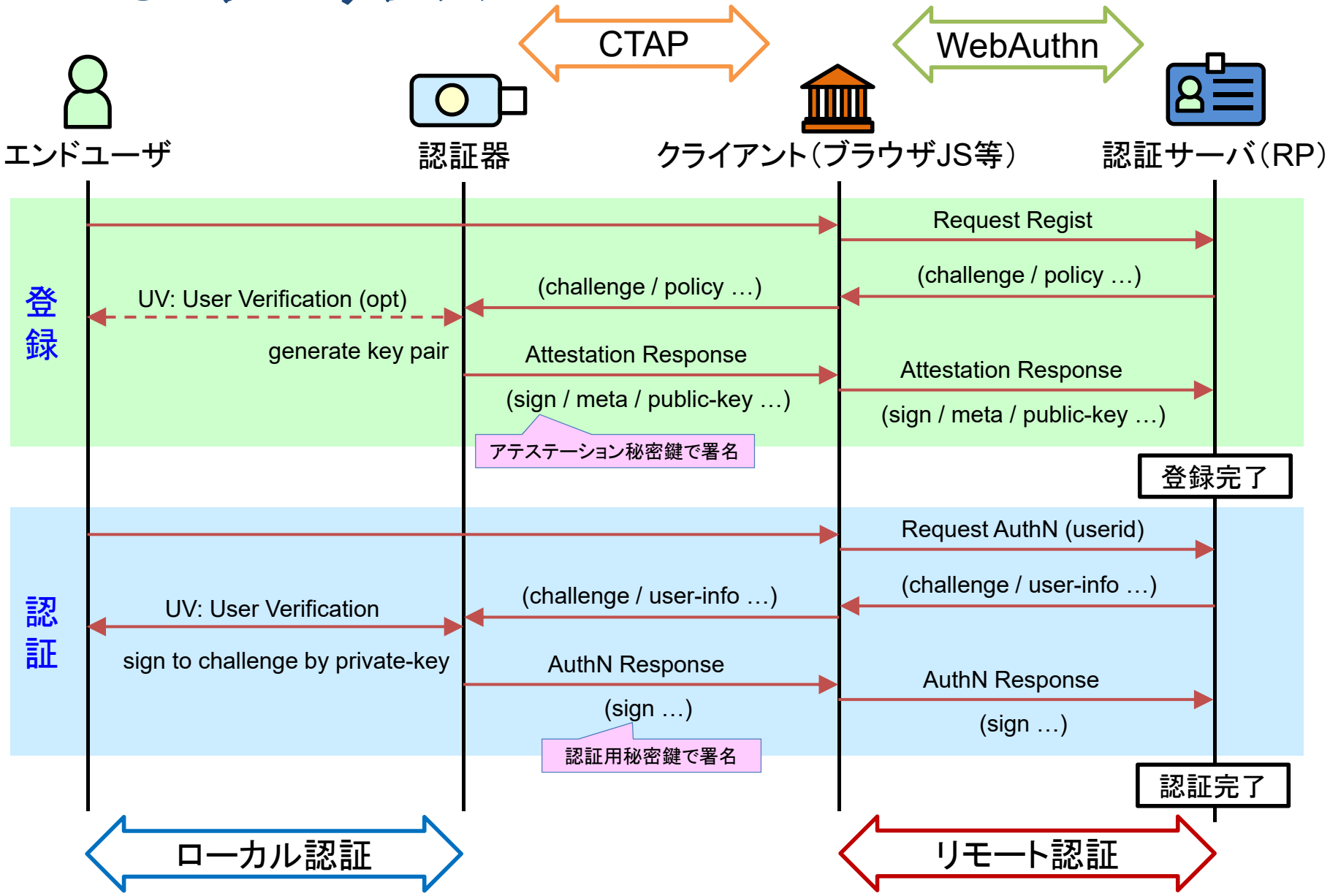
➢ W3C勧告 Web認証仕様

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

公開鍵暗号認証のAPI
An API for accessing Public Key Credentials
※ JavaScript/AndroidのAPIと通信プロトコル

※ 仕様　https://fidoalliance.org/specifications/download/

FIDO2 シーケンス


CTAP
WebAuthn
エンドユーザ
認証器
クライアント（ブラウザJS等）
認証サーバ（RP）
登録
Request Regist
(challenge / policy …)
UV: User Verification (opt)
(challenge / policy …)
generate key pair
Attestation Response
(sign / meta / public-key …)
Attestation Response
(sign / meta / public-key …)
アテステーション秘密鍵で署名
登録完了
認証
Request AuthN (userid)
(challenge / user-info …)
UV: User Verification
(challenge / user-info …)
sign to challenge by private-key
AuthN Response
(sign …)
AuthN Response
(sign …)
認証用秘密鍵で署名
認証完了
ローカル認証
リモート認証

WebAuthn: 公開鍵暗号方式の認証

登録：navigator.credentials.create()

1. 認証器（Authenticator）内で**公開鍵ペアを生成**する。
   - 生成する**公開鍵ペアは認証サーバ（RP）毎に異なる**
   - **秘密鍵は認証器内で保管**（認証器の外には出さない）
2. 認証サーバへアテステーションと公開鍵を送り**公開鍵を登録**する。
   - アテステーション（**Attestation**）：認証器の発行ベンダー署名付き情報

認証：navigator.credentials.get()

a. 認証サーバ（RP）は**チャレンジ情報をクライアントに送る**。
b. 認証器の機能で認証を行い、**利用者を認証**する。
   - 利用する認証器はポリシーで指定・選択が可能
c. 認証完了したら**認証器はチャンレンジ情報に署名**して返す。
d. 認証サーバは登録済み**公開鍵で検証**して認証完了を確認する。

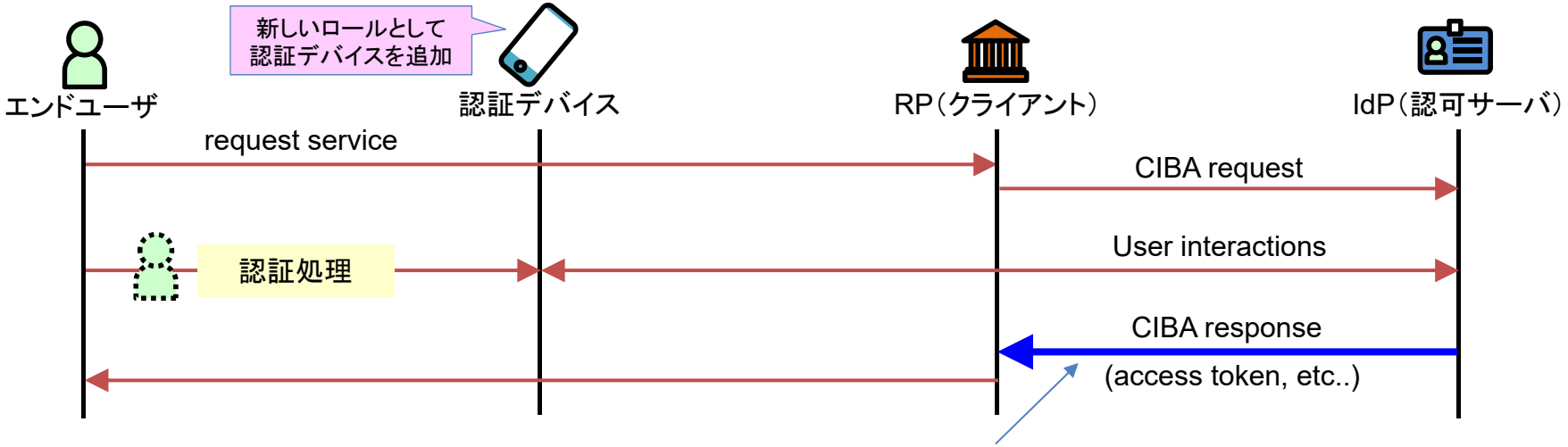
※ 生体等の認証情報はローカルで利用されるだけなので**サーバに漏えい**しない。
※ 認証サーバ・クライアント間の通信プロトコルは標準化されているので認証方法に関係なく**共通化**される（認証器を選べる）。
※ 利用者は公開鍵ペアに関して知る必要は無い（知らない）。

OpenID Connect CIBA（シーバ）Flow

CIBA：Client Initiated Backchannel Authentication

https://openid.net/specs/openid-client-initiated-backchannel-authentication-core-1_0.html

OIDCでドラフトを2018年に公開、クライアント開始バックチャンネル認証のフロー。



CIBA Response の３つのモード：非同期連携の為クライアントは認証終了を待つ

Poll（反復問合せ），Ping（通知後取得），Push（直接送信）

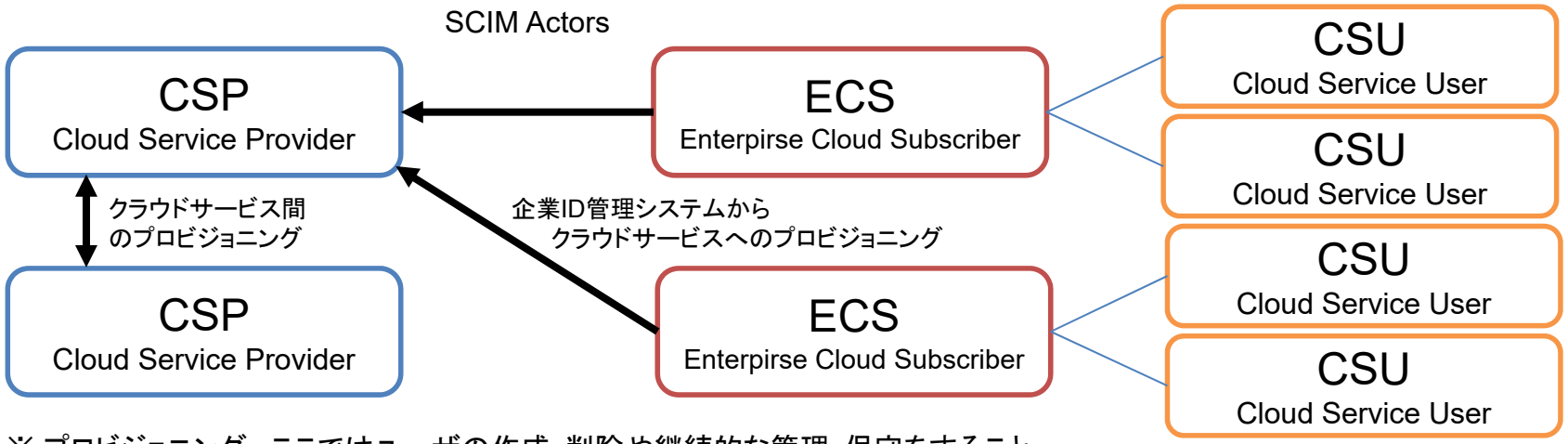
- **リダイレクトを使わない**のでエンドユーザに code や token が経由しない。
- エンドユーザと認証者は同じでなくても良い。例：エンドユーザが子供で認証者が親。
- クライアント認証が必須の為に **Confidential Client** のみ利用可能。
- 通常の認証処理と異なり目の前にいる顧客のアイデンティティ確認等への応用可。

SCIM（System for Cross-domains Identity Management）

ユーザの**アイデンティティ管理**を行う為のプロトコル標準：
RFC 7642（Overview）/ 7643（Core）/ 7644（Protocol）
最新仕様は SCIM2.0（2.0でXMLが非サポートに）、Slack や Azure AD 等が採用
http://www.simplecloud.info/

JSON ベースの **RESTful API** を使い標準化されている。

API： CRUD（生成/参照/更新/削除）、検索、同期、一括処理 等
スキーマ： ユーザとグループの最小限の定義と、拡張モデルを定義
認証認可： OAuth 2.0 を推奨（SCIM仕様には含まれない）



※ プロビジョニング：ここではユーザの作成・削除や継続的な管理・保守をすること。

まとめ…

電子認証の要素技術まとめ

OAuth 2.0：認可プロトコル標準

RESTful + JSON を使った認可を得る手順の標準仕様。
Access Token（認可チケット）を認可サーバから得る為の仕様。
Authorization Code Flow は code を得てから access_token に引換。
Implict Flow は直接 access_token を取得（セキュアにする為の工夫が必要）。

OpenID Connect：OAuth 2.0 へのアイデンティティ層の追加

OAuth 2.0 の**拡張仕様**（**JWT**/JWSの仕様も参照）。
ID Token（認証結果の情報）を認可サーバから得る為の仕様。
属性情報取得の UserInfo 仕様の追加（ID Tokenでも属性取得は可能）。
属性情報 Claim と scope の標準仕様と、nonce 仕様の追加。

OAuth PKCE：Authentication Code Flow の強化

code_verifier/code_challenge による**認可コード横取り攻撃**の対策。
認可サーバが正しい要求クライアントかを確認することを可能にする。

FIDO：認証標準

状況に応じて認証強度とUXを選択できる**認証標準**（多要素認証にも対応）。
OAuth 2.0 + OpenID Connect との組み合わせで利用できる。

最後に

お疲れ様でした！

本資料を読んで少しでも電子認証について分かった気になって頂けたならば幸いです。ただ電子認証を実現する為にはプロトコルの選定だけでは無くパラメータの考察や実装方法の検討も必要だと理解できたことと思います。難しいですね…、最後に幾つかの言葉を贈ります。

1. **実装時はめんどくさがらず必ず元仕様書を読み込みましょう。**
2. **可能な限り独自実装では無く実績がある実装を利用しましょう。**
3. **オレが考えた最強の仕様はだいたいセキュリティの穴があります。標準化された仕様に従いましょう。**
4. **分からないことは先達のブログや資料を検索しましょう。**

Enjoy and Thank You！