OBv3 / VC 署名デモアプリ — ユーザーガイド

1. このアプリとは

このデモアプリは、**Open Badges 3.0（OBv3）** で定義される **Verifiable Credential（検証可能な資格証明）**
が、どのように電子署名されるかを、すべての中間データを見ながら体験できる教育ツールです。

実際の画面で目にするもの：

- Ed25519 鍵ペアの生成と各種エンコーディング（hex / Multibase / JWK）
- サンプルの OBv3 AchievementCredential
- DataIntegrityProof の構築
- RDF Dataset Canonicalization（URDNA2015）による N-Quads への変換
- SHA-256 ハッシュ計算
- Ed25519 による署名
- 署名済み VC のダウンロードと検証

すべての暗号処理はブラウザ内で完結しており、秘密鍵がサーバーに送信されることはありません。

2. 主な用途

対象	用途
教育者・研修担当	OBv3 / VC の署名プロセスを可視化して説明する
エンジニア	`eddsa-rdfc-2022` 暗号スイートの実装参考として使う
発行機関（Issuer）	テスト用の署名済み VC を生成し、外部検証器で試す

3. 画面の操作方法

3.1 鍵ペアの生成（Step 01）

ページを開くと、自動的に Ed25519 の鍵ペアが生成されます。

- **Regenerate keypair** ボタンを押すと、新しい鍵ペアを作り直せます。
- 各鍵は以下の形式で表示されます：
- **Raw（hex）**: バイナリデータを16進数で表現
- **Multibase**: `z` プレフィックス＋base58btc エンコード（`did:key` と互換）
- **JWK**: JSON Web Key 形式の公開鍵

> ⚠ **Private key** は機密情報です。本番運用では決して画面に露出させないでください。

3.2 サンプル資格証明書の確認と編集（Step 02）

`Unsigned OBv3 AchievementCredential` セクションに、署名前の OBv3 JSON が表示されます。

主なフィールド：

- `@context`: スキーマ定義（W3C VC v2 + OBv3 コンテキスト）
- `issuer`: 発行者情報（`did:key` 形式の識別子）
- `credentialSubject`: 受領者と取得した実績（Achievement）
- `achievement.image`: バッジ画像（`id` に画像 URL、`type` は `"Image"`）

JSON の編集

表示されている JSON は **テキストエリア内で直接編集可能** です。受領者 ID、Achievement の名称や説明、発行日などを書き換えると、後続の Step 03〜09 の署名処理にそのまま反映されます。

- **Reset** ボタンで初期サンプルに戻せます。
- JSON 構文エラーがある場合は赤色のエラーメッセージで通知されます。

copy / view ボタン

JSON や中間データを表示する各カードの右上には、共通の操作ボタンが配置されています。

- **copy** — 現在の内容をクリップボードにコピーします。
- **view** — 現在の内容を **ブラウザの新しいタブ** で開きます。JSON は `application/json`、それ以外は `text/plain` として表示されるため、外部ツールへの貼り付け、ファイル保存、ブラウザ標準のフォーマット表示による確認に便利です。

バッジ画像の選択

JSON エディタの上にある **5 種類のバッジ画像**（Circle / Square / Hexagon / Shield / Rosette）から 1 つをクリックすると、`achievement.image.id` フィールドが自動的にそのバッジ画像の URL に書き換わります。

各 PNG は 400×400px、背景透過で、「デジタルクレデンシャル基礎 ハンズオン」と「1EdTech 2026-07」の文言が描かれたサンプル画像です。

バッジ概要カード

JSON エディタの下に、現在の Credential 内容を **ユーザーフレンドリーに可視化したカード** が表示されます。選択中のバッジ画像、Achievement 名、発行者、説明、発行日、受領者 ID をひと目で確認できます。JSON を編集するとリアルタイムに更新されます。

`credentialSubject.identifier` の追加（オプション）

JSON エディタの下にあるパネルから、credentialSubject に `identifier` を追加できます。

- **add credentialSubject.identifier** をチェックして有効化
- `identityType` を選択（例: `emailAddress`）
- raw な identifier 値を入力
- **hashed** を有効化すると `sha256(value + salt)` を計算し、`sha256$<hex>` 形式で `identityHash` に格納されます。`salt` も指定可能です。
- **Apply to credential** で `IdentityObject` として JSON に書き込まれます。

3.3 署名パイプラインの実行

Sign sample credential ボタンを押すと、Step 03 〜 Step 09 が順に表示されます。

Step 03 — Proof options

DataIntegrityProof のメタデータです。署名対象文書とは別に、署名の意味や発行時刻などを含みます。

```
{
  "type": "DataIntegrityProof",
  "cryptosuite": "eddsa-rdfc-2022",
  "created": "2026-06-20T00:00:00Z",
  "verificationMethod": "did:key:z6Mk...#z6Mk...",
  "proofPurpose": "assertionMethod"
}
```

Step 04 — RDF Dataset Canonicalization (URDNA2015)

JSON-LD 文書を RDF データセットに展開し、**URDNA2015**
アルゴリズムで正規化（Canonicalization）します。結果は **N-Quads** 形式のテキストです。

この処理の目的：

- JSON のオブジェクトキーの順序が異なっていても、同じ RDF グラフを持つ文書は同じ正規化結果になる
- 署名対象を一意に決定する

> N-Quads の各行は「主語 述語 目的語 グラフ」という4つ組（quad）で表現されます。

Step 05 — SHA-256 ハッシュ

正規化された文字列を UTF-8 エンコードし、それぞれ SHA-256 でハッシュします。

- `sha256(proofOptionsCanonical)` — 32バイト（64文字の hex）
- `sha256(documentCanonical)` — 32バイト（64文字の hex）

Step 06 — ハッシュの連結

`eddsa-rdfc-2022` スイートでは、以下の 64 バイトを署名対象（message）とします：

```
hashData = proofOptionsHash || documentHash
```

Step 07 — Ed25519 署名

発行者の**秘密鍵**で `hashData` に署名し、64 バイトの署名値を生成します。

- **signature（hex）**: 生の 64 バイト署名値
- **proofValue（Multibase）**: `z` プレフィックス＋base58btc でエンコードされた値。VC 文書内に埋め込まれます。

Ed25519 の特徴：

- 決定論的署名（同じ message と鍵から常に同じ署名が出る）
- 高速かつ短い署名値（64 バイト）

Step 08 — Signed Verifiable Credential

`proof` ブロックを元の Credential に埋め込み、**署名済み VC** が完成します。

画面下部には 2 つのダウンロードボタンが表示されます：

- **Download signed credential JSON** — 署名済み VC を `.json` ファイルとして保存します。
- **☒ Download baked badge PNG** — 選択したバッジ画像（PNG）に、署名済み VC の JSON を `iTXt` チャンク（キーワード `openbadgecredential`）として埋め込んだ **Baked Badge 画像** を生成・ダウンロードします。これは Open Badges 3.0 の "Baking" 仕様に準拠しており、画像 1 枚で資格情報を持ち運べます。

> Baked PNG は通常の画像として表示でき、かつ対応する検証器（1EdTech OB30Inspector など）にアップロードすると、埋め込まれた署名済み VC を読み取って検証できます。

Step 09 — Verification（検証）

保存された署名済み VC から `proof` を取り除き、再度 Step 04〜06 と同じ手順で `hashData` を導出します。そして公開鍵で署名値を検証します。

- ☒ **Signature is valid** ☒ 文書が改ざんされていないことを確認
- 導出した `hashData` が Step 06 と一致することも確認されます

3.4 DidMon 連携機能

このデモは [DidMon](https://didmon.net) と連携し、実際の DID / VC エコシステムに接続できます。

3.4.1 Holder / Issuer それぞれのログイン

画面ヘッダーに **Login with Holder** と **Login with Issuer** の 2
つのボタンが横並びで配置されています。どちらも PKCE フローで DidMon
にサインインし、それぞれ独立したセッションとして管理されます。

- **Login with Holder** — サインイン後、Holder の DID が `credentialSubject.id` に自動で反映されます。
- **Login with Issuer** — Issuer source で「DidMon Issuer」を選んだときに、この DID が `issuer.id`
  として使われます。
- コールバックは `/auth/callback` で処理されます。
- 各ユーザーの DID、name、`proof_id` はヘッダー下のカードに表示されます。

3.4.2 Issuer source の選択

Step 01 の下に **Issuer source** トグルがあり、3 つの発行モードを切り替えられます。

Mode	issuer.id	署名主体
Local Ed25519	ブラウザで生成した `did:key:...`	ブラウザ内 Ed25519
DidMon Issuer	ログイン中の Issuer ユーザーの `did:web:...`	DidMon Issuer API（ES256）
DidMon Lovable Demo Issuer (fixed)	`did:web:did.didmon.net:user:user_lovable_demo_issuer`	DidMon Backend Issuer（ES256、固定）

`DidMon Issuer` または `Lovable Demo Issuer` を選ぶと、Step 08 のダウンロードや PNG baking は **DidMon が発行した ES256 の署名済み VC**（JWT / JSON-LD）を利用します。ローカル `did:key` の proof
は使われません（controller と issuer が一致しないため）。

3.4.3 DidMon Issue セクション

Step 02 の下に **DidMon Issue** セクションがあります。

- **Local Ed25519** モードでは通常のブラウザ内署名パイプラインだけが動きます。
- **DidMon Issuer** モードでは `POST https://api.didmon.net/issuer/issue` を、Issuer
  のアクセストークン付きで呼び出します。
- **Lovable Demo Issuer** モードでは `POST https://api.didmon.net/issuer/lovable-demo/issue`
  をバックエンド管理の固定 Issuer で呼び出します。Holder のアクセストークンがあれば Bearer
  で付与されます。
- 発行に成功すると `vcId` / ES256 の `jwt` / `sdJwt` / JSON-LD `credential`
  が保存され、以後のダウンロード・検証・Verifier 起動に使われます。

3.4.4 Step 10 — Verify with DidMon

署名またはバックエンド発行が完了すると、Step 10 の **Verify with DidMon** が有効になります。

- **Verify** ボタンは `POST https://api.didmon.net/verifier/verify` を呼び出し、Holder Binding
  を検証します（現在は ES256 の JWT VC のみ送信します。EdDSA の場合はローカル検証結果のみを表示）。
- **Open in DidMon Verifier** ボタンは https://verifier.didmon.net
  を新しいタブで開きます。長い `vc_jwt` は URL の **hash fragment (`#`)** に載せることで CloudFront の
  403 を回避しています。`login_hint` のみ query string に付与されます。

• Registry 発行済みの ES256 JWT がない場合、Verifier は開けません（Lovable ローカル EdDSA JWT は転送しません）。

4. 外部検証器での確認

ダウンロードした `signed-credential.json` を、以下の検証サービスで試すことができます：

1EdTech OB30Inspector

https://vc.1ed.tech/upload?validatorId=OB30Inspector

Choose File でダウンロードした JSON を選択
Validate を実行
検証結果を確認

本アプリは `eddsa-rdfc-2022` 暗号スイートを採用しており、1EdTech の検証器と互換性があります。DidMon Issuer / Lovable Demo Issuer モードで発行した VC は ES256 (`did:web`) で署名されており、こちらも 1EdTech 検証器と互換です。

DidMon Verifier

https://verifier.didmon.net

Step 10 の **Open in DidMon Verifier** ボタンから、ES256 JWT VC をワンクリックで開けます。

5. 技術仕様まとめ

項目	仕様
鍵アルゴリズム	Ed25519
暗号スイート	`eddsa-rdfc-2022`
正規化	RDF Dataset Canonicalization（URDNA2015）
ハッシュ関数	SHA-256
キーエンコーディング	Multibase base58btc（`z` プレフィックス）
DID メソッド	`did:key`
標準	W3C Verifiable Credentials Data Model 2.0, Open Badges 3.0

使用ライブラリ

- `@noble/ed25519` — 純粋 JavaScript の Ed25519 実装
- `@noble/hashes` — SHA-256 / SHA-512
- `@scure/base` — Base58 / Base64URL エンコーディング
- `jsonld` — JSON-LD 処理と RDF Dataset Canonicalization

6. よくある質問

Q1. 本番で使う鍵ペアをこのアプリで生成してもよいですか？

推奨しません。 このアプリはデモ・教育目的です。本番環境では、HSM（Hardware Security Module）や専用のキー管理サービスを使用してください。

Q2. 署名時刻（created）は固定されていますか？

はい、デモ用に `2026-06-20T00:00:00Z` に固定しています。実際の発行システムでは現在時刻を使用します。

Q3. JCS（JSON Canonicalization Scheme）ではなく RDFC を使うのはなぜですか？

Open Badges 3.0 / 1EdTech の検証エコシステムでは、`eddsa-rdfc-2022`（RDF Dataset Canonicalization）が広くサポートされています。本アプリはその互換性を優先しています。

Q4. ブラウザのどの機能を使っていますか？

- Web Crypto API（`crypto.getRandomValues` による乱数生成）
- それ以外のすべての暗号処理は上記の純粋 JavaScript ライブラリで行っています

Q5. サーバー通信は行われますか？

いいえ。鍵の生成、署名、検証のすべてはクライアントサイド（ブラウザ内）で完結します。ネットワーク通信は発生しません。

7. 関連リンク

- [W3C Verifiable Credentials Data Model 2.0](https://www.w3.org/TR/vc-data-model-2.0/)
- [W3C Verifiable Credential Data Integrity 1.0](https://www.w3.org/TR/vc-data-integrity/)
- [1EdTech Open Badges Specification](https://www.imsglobal.org/spec/ob/v3p0/)
- [eddsa-rdfc-2022 Cryptosuite](https://www.w3.org/TR/vc-di-eddsa/#the-eddsa-rdfc-2022-suite)

このガイドはデモアプリの現行バージョンに対応しています。