TOPPAN

令和 4 年度補正

Trusted Web 開発等推進事業に係る調査研究

（規格・実装動向調査）

2024 年 3 月

TOPPAN 株式会社

本書の位置づけ

様々な社会活動のデジタル化が進む一方で、やりとりされるデータそのものの信頼への懸念、先鋭化していくプライバシーリスク、データの取扱いへの懸念からくる産業界におけるデータ活用の停滞、勝者総取り等によるエコシステムのサステナビリティへの懸念など、信頼できる自由なデータ流通（DFFT）を妨げる、様々な歪みが生じている。

これらの懸念は、データそのものが信頼できない、データのやり取りをする相手を信頼できない、相手方におけるデータの取扱いを信頼できないといった現状が主な原因と考えられる。

上記背景から、インターネット上で、DFFT を確保する枠組みを構築すべく、特定のサービスに依存せずに、個人・法人によるデータのコントロールを強化する仕組み、やり取りするデータや相手方を検証できる仕組みなどの新たな信頼の枠組みを付加することを目指す Trusted Web 構想を実現していくことが重要であり、実証や調査、コミュニティ形成を進めていくことが求められている。

本事業は、2022 年度事業である 13 件のユースケースの開発実証等や、内閣官房において活動を進めている「Trusted Web 推進協議会」における Trusted Web ホワイトペーパー策定等の活動、他検討結果を踏まえて、TOPPAN 株式会社（以下、TOPPAN）がデジタル庁の委託を受けて以下の業務 [1]を実施した。

1. Trusted Web ユースケース開発実証に係る調査研究
2. 官民コンソーシアム組成・運営
3. 調査・普及啓発

本報告書は、上記調査・普及啓発の中で、規格・実装にかかる動向調査をとりまとめたものである。

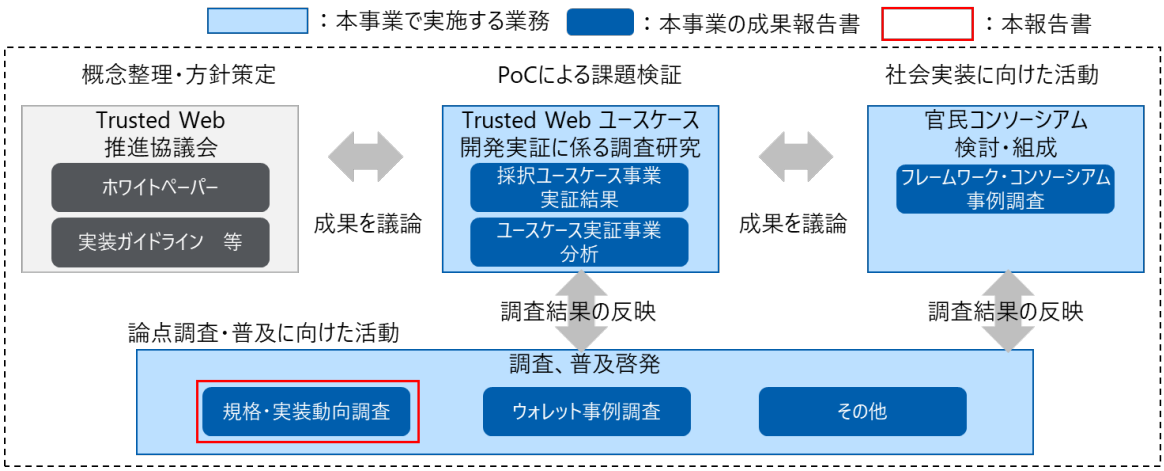


図 本事業のスコープ

[1] 本事業の業務は一部デロイトトーマツグループの委託を受けて実施している

目次

1. 背景・目的

【背景】

近年、アイデンティティにかかるデータ主権を個人に帰属させることや、単一障害点の排除等の観点で、分散型アイデンティティが注目されており、W3C(World Wide Web Consortium)で規格化された VC(Verifiable Credentials)や DIDs (Decentralized Identifiers)の技術を活用したサービスの実装検討が増えてきている。

これらの規格は、DIF(Decentralized Identity Foundation)や、Hyperledger、OpenID Foundation 等の国際団体で標準化に向けた取組が進められている。また、EU・カナダ等の国・地域では、アイデンティティサービス提供にかかるフレームワーク等を策定して VC/DIDs を活用したサービスの相互運用性を高める取組が進められている。

ただし、現時点ではこれらの実装方法は多様であり、どのように収束していくかを把握するのが課題である。

今後本邦においても相互運用性が確保されたサービスを提供していくためにも、現時点の規格動向や、規格を普及させるための取組がどうなっていて、各主要な団体がどのような実装を検討されているかを把握することが求められる。

【目的】

本調査では、VC/DIDs 等を活用したアイデンティティサービスにかかる技術スタックにおける規格の動向を調査し、実装パターンがどの程度あるかを把握する [2]

- 規格実態
- 技術スタック詳細
- 実装パターンの抽出

[2] 本調査は 2024 年 3 月までの既知の情報を元に記載している。また、実装パターンを紹介しているが本領域は発展途上の段階であり、各技術および組合せに関する安全性、信頼性を担保していない。また、本調査はデジタル庁の意見を表明しているものではない。

2．調査アプローチ

デジタルアイデンティティに関連する規格の実態調査を行った後、調査・比較できる技術スタックの詳細調査、サービスにおける実装パターンの抽出と現在市中で公開されているライブラリ・OSS 等で対応している技術範囲の整理等を実施した。

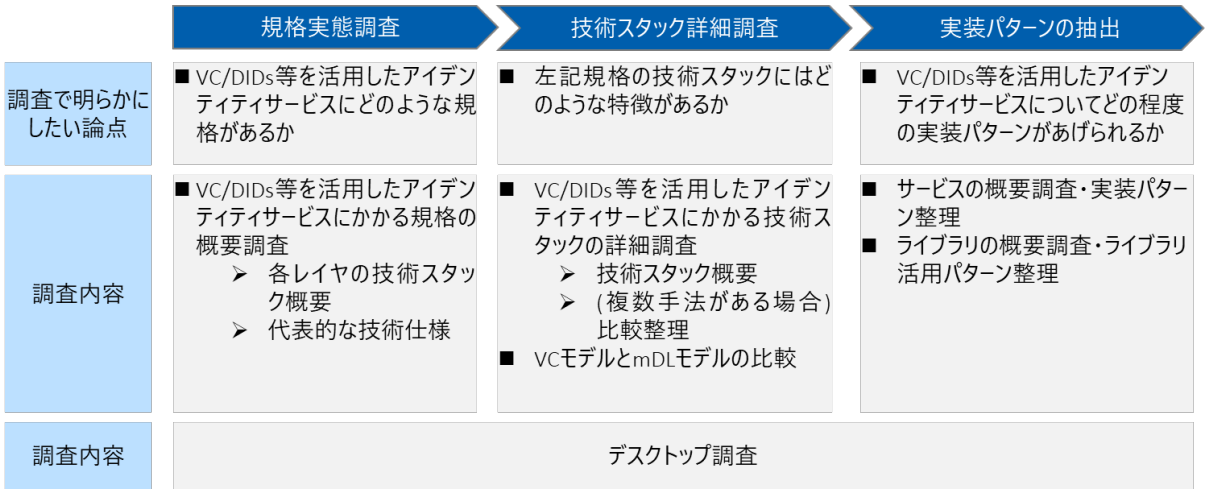


図 2-1　調査アプローチ

3．調査まとめ

本調査の結果のまとめを以下に記載する。

表 3-1　調査結果まとめ

規格実態調査	・ DIF で整理されている「Interoperability Mapping Excercise」をもとにレイヤ別(「Credential Layer」、「Agent Layer」、「Public Trust Layer」の「Vertical / Cross-cutting」)にデータフローのイメージ、規格概要、代表的な規格を取りまとめた ・ 業界ごとの VC データモデルの検討状況を整理した ・ ISO で検討されている mDL 関連規格(18013 シリーズ/23220 シリーズ)の各項目で検討されている規格概要・規格ステータスを整理した
技術スタック詳細調査	➢ Credential Layer では、証明書フォーマット・証明書の検証モデル・証明書に記載されている情報の選択的開示に関する規格の概要・比較整理を行った ➢ Agent Layer では、通信プロトコル・デバイス連携に関する規格の概要・比較整理を行った ➢ Public Trust Layer では、DID Document を活用した場合/しなかった場合の証明書の検証方法、DID Document を活用した場合のストレージ比較、名前解決方法、データ格納庫等について整理した ➢ mDL・VC の特徴(データモデル・アーキテクチャ・検証フロー)の整理を行った
実装パターンの抽出・事例調査	【サービス・フレームワーク】 ➢ 実際に普及しているサービス・ガイドラインのサービス概要・実装されている技術スタック当を調査し、その中から現時点で考えうる実装パターンを分析した 【ライブラリ】 ➢ 実際に普及しているライブラリの概要・提供形態・対応範囲等を調査し、事業者が実装する場合にどのライブラリを活用すべきかの分析を行った

4. 規格実態調査

4.1. 規格全体観

規格の全体観は DIF の「Interoperability Mapping Excercise」[3]をもとに整理を行った。相互運用性に大きくかかわる証明書のフォーマットや提示、交換、紐づけ方を規定する(1)Credential Layer、証明書のやり取りを行う通信等を規定する(2)Agent Layer、トラストの基盤となるプラットフォームや情報の格納形式、アクセス方式を規定する(3)Public Trust Layer、(1)～(3)にに共通的に関与する要素技術となる(4)Vertical / Cross-cutting に分類することができる。

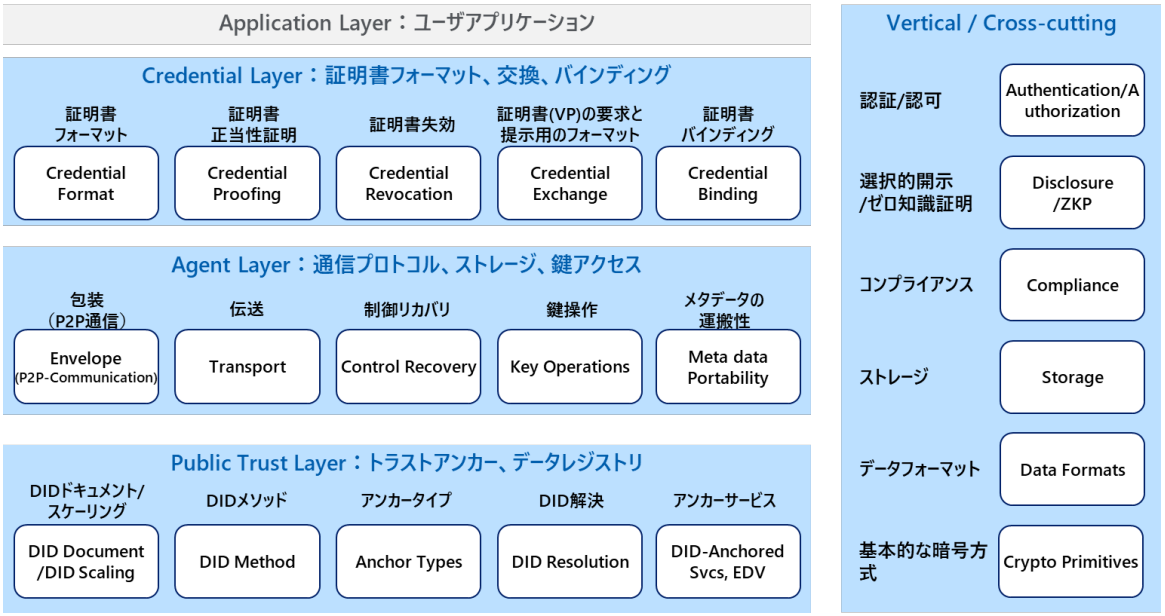


図 4-1-1　相互運用性に関わる技術スタックマッピング

[3] https://github.com/decentralized-identity/interoperability/blob/master/assets/interoperability-mapping-exercise-10-12-20.pdf

4.2. 各規格の概要・代表的な仕様

（1） Credential Layer

Credential Layer は、主に、証明書のフォーマット、失効、交換方式に関する規格群が定義されている。証明書連携フローについて Credential Layer で行われていること概要を図示すると以下のようになる。

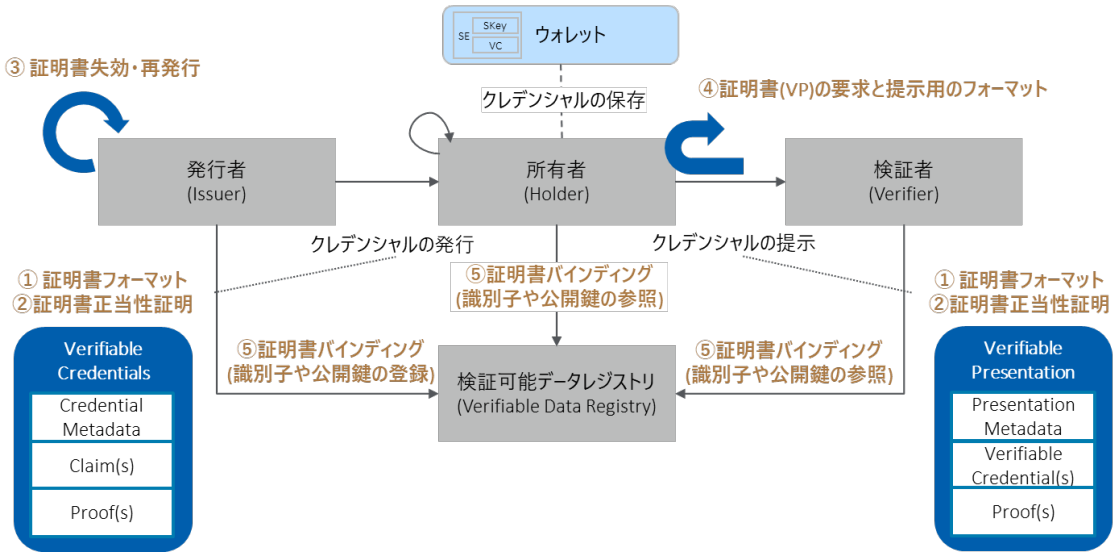


図 4-2-1　証明書連携フロー（Credential Layer）

以下、果たす役割と主要な規格について説明する

1. Credential Format（証明書フォーマット）

- 検証可能な資格情報を発行・運用するデータモデル/方式を定義している
- 代表的な規格として Verifiable Credentials (W3C)[4]、OpenID Foundation が策定した OID4VCI[5]、OID4VP[6]が挙げられる。

2. Credential Proofing（証明書正当性証明）

- 証明書のデータ形式を定義しており、選択的属性開示とも関連が深い。
- 代表的な規格として VC JWT[7]、VC JSON-LD Proofs[8] 、SD-JWT VC[9]、AnonCreds[10]等が挙げられる。

---

[4] W3C Verifiable Credentials には、2024 年 3 月現在、主に Ver1.1 と、Ver.2.0 が検討されている
Ver1.1：https://www.w3.org/TR/vc-data-model/ 、Ver2.0：https://www.w3.org/TR/vc-data-model-2.0/

[5] https://openid.net/specs/openid-4-verifiable-credential-issuance-1_0.html

[6] https://openid.net/specs/openid-4-verifiable-credential-issuance-1_0.html

[7] https://identity.foundation/jwt-vc-presentation-profile/

[8] https://w3c.github.io/vc-data-integrity/#data-model

[9] https://www.ietf.org/archive/id/draft-ietf-oauth-sd-jwt-vc-00.html

[10] https://hyperledger.github.io/anoncreds-spec/

3. Credential Revocation（証明書失効）

- 代証明書の失効管理方法について定義する。
- 代表的な規格として x.509[11]の Online Certificate Status Protocol や Certificate Revocation List、VC Status Revocation List[12]、AnonCreds Revocation Status List v1/v2[13]等が挙げられる。

4. Credential Exchange（証明書(VP)の要求と提示用のフォーマット）

- 証明書を所有者が検証者に渡す際の交換方式を定義しており、クレデンシャル(VP)の要求と提示に関するフォーマットを示す。
- 代表的な規格として DIF の Presentation Exchange[14]、W3C の Verifiable Presentation Request[15] や OpenID Foundation の OpenID Connect Credential Provider[16]、Hyperledger で検討している Present Proof Protocol[17]等が挙げられる。

5. Credential Binding：証明書を紐づけるもの（識別子等）

- 特定の証明書と特定の公開鍵が関連していることを表す識別子等の情報群を定義しており、例えば、証明書の真正性を保証するための署名を検証する公開鍵がどこにあるか示すものや、ユーザのアイデンティティを証明する情報群を提供している。
- 代表的な規格として W3C[18]で定義している Decentralized Indentifiers や OpenID Connect ID Token[19]等が挙げられる。

参考：業界別 VC データモデル整備状況

学歴証明、製品の品質保証、ワクチン証明等でのユースケースにおいて、VC データモデルにかかる標準化整備が進んでいる。

表 4-2-1　VC データモデルが整備されているユースケース [20]

大学卒業証明	・ 大学卒業資格を定義（学士・博士等の課程や学位）
認定ミル	・ 金属等を用いて作られた製品が特定の規格に準拠した品質で生産されていることを保証された証明書の定義
原油取引	・ 原油の品質や運搬・貿易にかかる記録等を定義
Covid-19	・ Covid-19 の接種証明書の定義

[11] https://datatracker.ietf.org/doc/html/rfc5280
[12] https://www.w3.org/community/reports/credentials/CG-FINAL-vc-status-list-2021-20230102/
[13] https://hyperledger.github.io/anoncreds-revocation/
[14] https://identity.foundation/presentation-exchange/
[15] https://w3c-ccg.github.io/vp-request-spec/
[16] https://mattrglobal.github.io/oidc-client-bound-assertions-spec/
[17] https://github.com/hyperledger/aries-rfcs/blob/main/features/0454-present-proof-v2/README.md
[18] https://github.com/hyperledger/aries-rfcs/blob/main/features/0454-present-proof-v2/README.md
[19] https://openid.net/developers/how-connect-works/
[20] https://w3c-ccg.github.io/vc-examples/

また、現時点ではデータモデルの整備が確認されていないが、下記図に記載の通り、教育・小売・金融・ヘルスケア・職業資格確認・法的身分確認・モノの情報管理等において VC を活用したユースケース創出が期待されている。

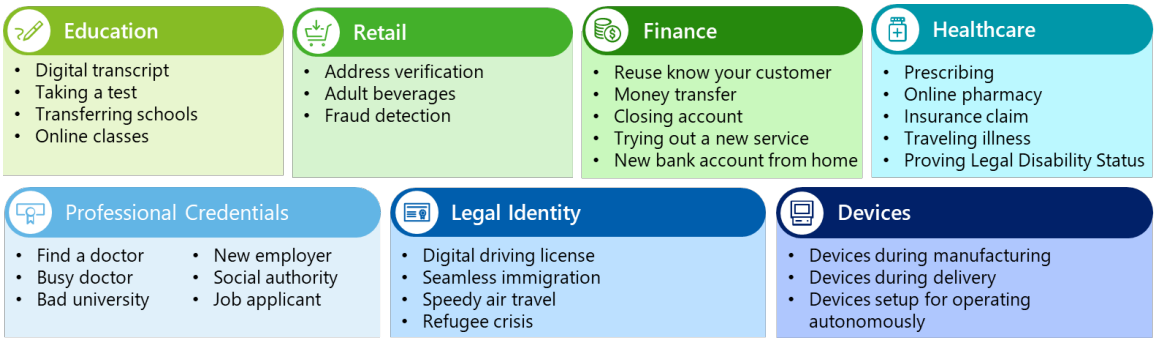


図 4-2-2　データモデル整備が期待されているユースケース [21]

[21] https://www.w3.org/TR/vc-use-cases/

（2） Agent Layer

Agent Layer は、主に、エージェント間のセキュアな Peer-to-Peer コミュニケーションを実現するための証明書の検証方法や連携方法、および鍵情報へのアクセス方式等の規格群が定義されている。

証明書連携フローについて Agent Layer で行われていること概要を図示すると以下のようになる。

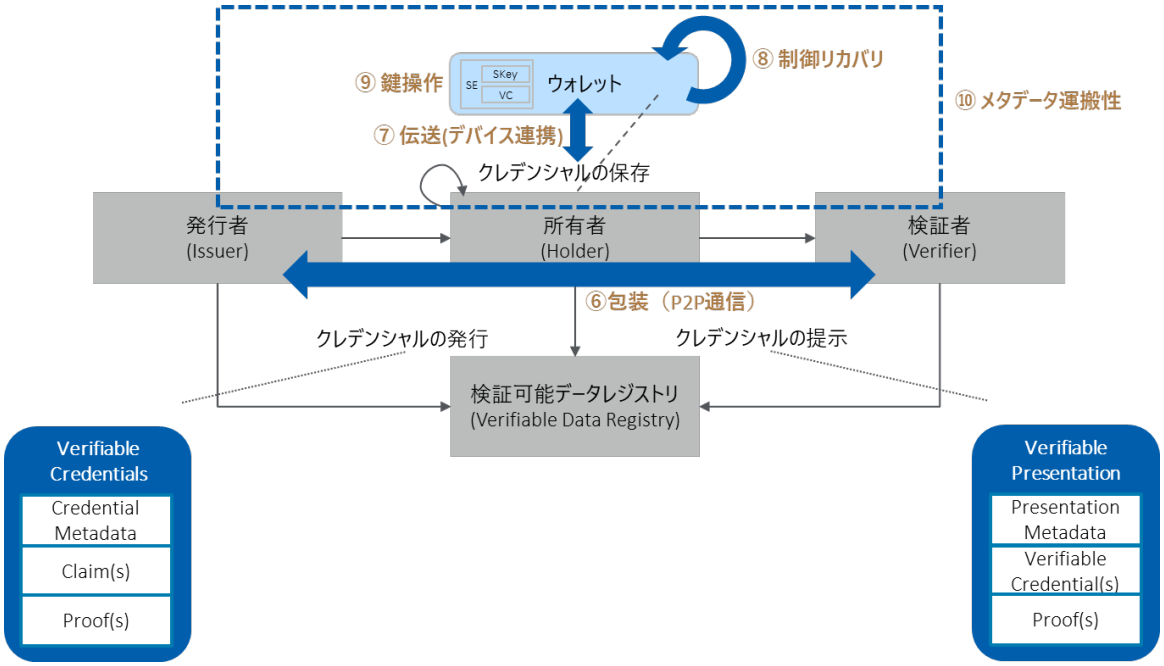


図 4-2-3　証明書連携フロー（Agent Layer）

以下、果たす役割と主要な規格について説明する。

6.　Envelope(P2P-Communication)（包装(P2P 通信)）

- エージェント間で通信に用いるプロトコル、エンコード方式等を定義する。
- 代表的な規格として IETF が定義している JSON Web Message、DIF で定義している DID Comm[22]や Self-Issued OpenID Connect Provider DID Profile v0.1[23]等が挙げられる。

7.　Transport（伝送）

- 異なるデバイス間で証明書やアイデンティティ情報の格納先などを連携する方法を定義する。
- 代表的な規格として近距離無線通信規格の NFC や Bluetooth、QR コード、HTTP 等が挙げられる。

[22] https://identity.foundation/didcomm-messaging/spec/v2.0/

[23] https://identity.foundation/did-siop/

8. **Control Recovery（制御リカバリ）**
   - 証明書やそれに関連する鍵情報などを品質した場合のリカバリ方法について定義する。
   - 代表的な規格としてブロックチェーンのウォレットのバックアップ用として用いられる BIP-39[24]や生体認証と組み合わせてバックアップする Horcrux Protocol[25]等が挙げられる。

9. **Key Operations（鍵操作）**
   - 鍵情報をセキュアに格納する方式について定義する。
   - 代表的な規格としてハードウェアに鍵を格納する Cloud/Local HSM[26]や TEE Cips[27]等が挙げられる。

10. **Meta data Portability（メタデータ運搬性）**
    - エージェント間でデータを運搬する方式について定義する。
    - 代表的な規格として W3C で定義している Universal Wallet 2020[28]が挙げられる。

[24] https://trezor.io/learn/a/what-are-bips-slips#BIP39_-_Mnemonic_code_for_generating_deterministic_keys
[25] https://github.com/johncallahan/activestorage-horcrux
[26] https://cloud.google.com/kms/docs/hsm?hl=ja
[27] https://en.wikipedia.org/wiki/Trusted_execution_environment
[28] https://w3c-ccg.github.io/universal-wallet-interop-spec/

（３） Public Trust Layer

Public Trust Layer は、主に分散型 ID のトラストの基盤となるプラットフォームや情報の格納形式、アクセス方式が定義されている。

証明書連携フローについて Public Trust Layer で行われていること概要を図示すると以下のようになる。

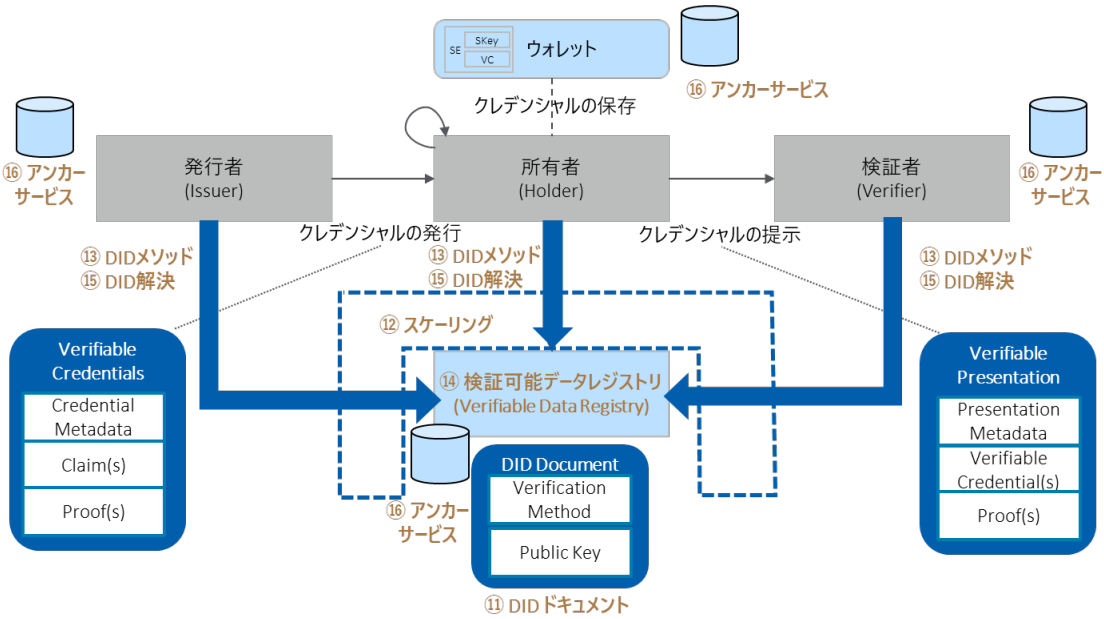


図 4-2-4　証明書連携フロー（Public Trust Layer）

以下、果たす役割と主要な規格について説明する。

11.　DID Document（DID ドキュメント）

- 証明書の発行元を証明する署名検証のための各種情報を格納したデータモデルを定義する。
- 代表的な規格として DID Document[29]が挙げられる。

12.　DID Scaling（DID スケーリング）

- 分散型 ID のトラスト基盤プラットフォームとしてブロックチェーンが用いられることが多いが、一般的にアイデンティティ管理システムとしてのスケーラビリティを確保するための方式を定義する。
- 代表的な規格として DIF の Sidetree Protocol[30]や KERI[31]（Key Event Receipt Infrastructure）が挙げられる。

13.　DID Method（DID メソッド）

- 分散型アイデンティティシステムの相互運用性を高めるための、各方式を識別する方法を定義する。

---

[29] https://www.w3.org/TR/did-core/

[30] https://identity.foundation/sidetree/spec/

[31] https://github.com/decentralized-identity/keri

- 代表的な規格として W3C が定義している DID Method[32]が挙げられる。
- DID Method にて各方式ごとに定義されており、Ethereum を用いる did:ethr、Hyperledger で検討されている Sovrin Network を用いる did:sov、データレジストリを挟まず P2P で直接やり取りする did:key 等がある。

14. Anchor Types（アンカータイプ）

- データレジストリの具体的な実装基盤を定義する。
- 代表的な規格として Hyperledger Indy[33]を用いるプライベートチェーンである Sovrin Network[34]や、Ethereum[35]や Bitcoin[36]などのパブリックチェーン等が挙げられる。

15. DID Resolution（DID 解決）

- DID Method を参照してデータレジストリを特定するための名前解決の仕組みを定義する。
- 代表的な規格として W3C が定義している Decentralized Identifier Resolution (DID Resolution) v0.3[37]や DIF で定義している Universal Resolver[38]が挙げられる。

16. DID-Anchored Svcs（アンカーサービス）

- 鍵情報や DID Document を格納するためのセキュアなストレージサービスを定義する。
- 代表的な規格として DIF が定義している Identity Hub[39]や Digital Bazaar が定義している Encrypted Data Vaults[40]（EDV）が挙げられる。

---

[32] https://www.w3.org/TR/did-core/
[33] https://hyperledger-indy.readthedocs.io/projects/indy/en/latest/
[34] https://sovrin.org/overview/
[35] https://sovrin.org/overview/
[36] https://identity.foundation/ion/
[37] https://w3c-ccg.github.io/did-resolution/
[38] https://github.com/decentralized-identity/universal-resolver
[39] https://didproject.azurewebsites.net/docs/hub-overview.html
[40] https://digitalbazaar.github.io/encrypted-data-vaults/

（4） Vertical / Cross-cutting

Vertical / Cross-cutting では、暗号方式や選択的開示方式等のレイヤーを横断する要素技術・取組等について提示している。

17. Authentication/Authorization（認証/認可）

- 分散型アイデンティティと連携する認証/認可システムを示している。
- 代表的な規格として W3C の WebAuthn[41]や OpenID Foundation の、OpenID Connect、SIOP[42]等が挙げられる。

18. Disclosure/ZKP（選択的開示/ゼロ知識証明）

- 所有者が検証者に提供する証明書を必要最低限の情報にすることでプライバシーを担保するため、証明書から開示する情報を選択できる方式を示している。
- 代表的な規格として Hyperledger で定義している AnonCreds ZKPs[43]や、W3C で規定している BBS+ Signature[44]等が挙げられる。

19. Compliance（コンプライアンス）

- 分散型アイデンティティの実現する上で前提となる国や組織のコンプライアンスを示している。
- 代表的な規格として欧州の GDPR[45]や eIDAS[46]等が挙げられる。

20. Storage（ストレージ）

- 各種データを格納するためのデータベース、分散型ストレージ等を示している。
- 代表的な規格としてデータベースの MySQL、CouchDB や分散型ストレージの IPFS[47]等が挙げられる。

21. Data Format（データフォーマット）

- 各種データを表現するデータフォーマットを示している。
- 代表的な規格として JSON[48]、XML、CBOR[49]等が挙げられる。

22. Cripto Primitives（基本的な暗号方式）

- 証明書の署名方式や暗号方式を示している。
- 代表的な規格として ECDSA、EdDSA や secp256k1 が挙げられる。

---

[41] https://digitalbazaar.github.io/encrypted-data-vaults/

[42] https://identity.foundation/did-siop/

[43] https://github.com/hyperledger/anoncreds-spec/blob/main/README.md

[44] https://w3c.github.io/vc-di-bbs/

[45] https://commission.europa.eu/law/law-topic/data-protection/data-protection-eu_en

[46] https://joinup.ec.europa.eu/sites/default/files/document/2020-04/SSI_eIDAS_legal_report_final_0.pdf

[47] https://github.com/ipfs/specs

[48] https://datatracker.ietf.org/doc/html/rfc8259

[49] https://datatracker.ietf.org/doc/html/rfc8949

参考：mDL 関連の技術規格動向

mDL は ISO を中心に規格策定を進められており、（ISO/IEC18013 シリーズ、23220 シリーズ)標準化が一定完了し文書として公開されているものと、策定途中のものが存在する。

表 4-2-1　mDL に関する仕様概要

ISO/IEC シリーズ	項目		定義概要	ステータス
ISO/IEC18013 (運転免許証にかかる標準化)	18013-1	Physical characteristics and basic data set[50]	免許証の物理特性と基本的なデータセット	Published
	18013-2	Machine-readable technologies[51]	免許証の機械読取部分のデータ構造や読取方法	Published
	18013-3	Access control, authentication and integrity validation[52]	機械読取部分のアクセス制御、整合性検証方法	Published
	18013-4	Test methods[53]	適合性テストに使用される方法	Published
	18013-5	Mobile driving licence (mDL) application[54]	mDL 実装のためのインターフェイス仕様	Published
	18013-6	mDL test methods[55]	mDL のテスト方法	Under development
	18013-7	Mobile driving licence (mDL) add-on functions[56]	mDL のアドオン機能	Under development
ISO/IEC23220 (デジタル ID にかかる標準化)	23220-1	Generic system architectures of mobile eID systems[57]	モバイル ID 管理のアーキテクチャとライフサイクル	Published
	23220-2	Data objects and encoding rules for generic eID systems[58]	汎用 ID のデータ構造とエンコード規則	Under development
	23220-3	Protocols and services for issuing phase[59]	発行フェーズのプロトコルとサービス	Under development
	23220-4	Protocols and services for operational phase[60]	運用フェーズのプロトコルとサービス	Under development
	23220-5	Trust models and confidence level assessment[61]	信頼モデルと信頼度評価	Under development
	23220-6	Mechanism for use of certification on trustworthiness of secure area[62]	セキュアエリアの信頼度に関する認証利用の仕組み	Under development

[50] https://www.iso.org/standard/63798.html
[51] https://www.iso.org/standard/70486.html
[52] https://www.iso.org/standard/72366.html
[53] https://www.iso.org/standard/74961.html
[54] https://www.iso.org/standard/69084.html
[55] https://www.iso.org/standard/79805.html
[56] https://www.iso.org/standard/82772.html
[57] https://www.iso.org/standard/74910.html
[58] https://www.iso.org/standard/86782.html
[59] https://www.iso.org/standard/86783.html
[60] https://www.iso.org/standard/86785.html
[61] https://www.iso.org/standard/86786.html
[62] https://www.iso.org/standard/86787.html

5. 規格詳細・実装方式調査

5.1. 調査対象

4 章で整理した技術スタックのうち、現時点で仕様がある程度固まっているかつ実装パターンに影響が出る範囲を調査対象とした [63]。各技術要素で、実装手法が複数選択肢がある場合は主要なものの特徴を比較・整理した。

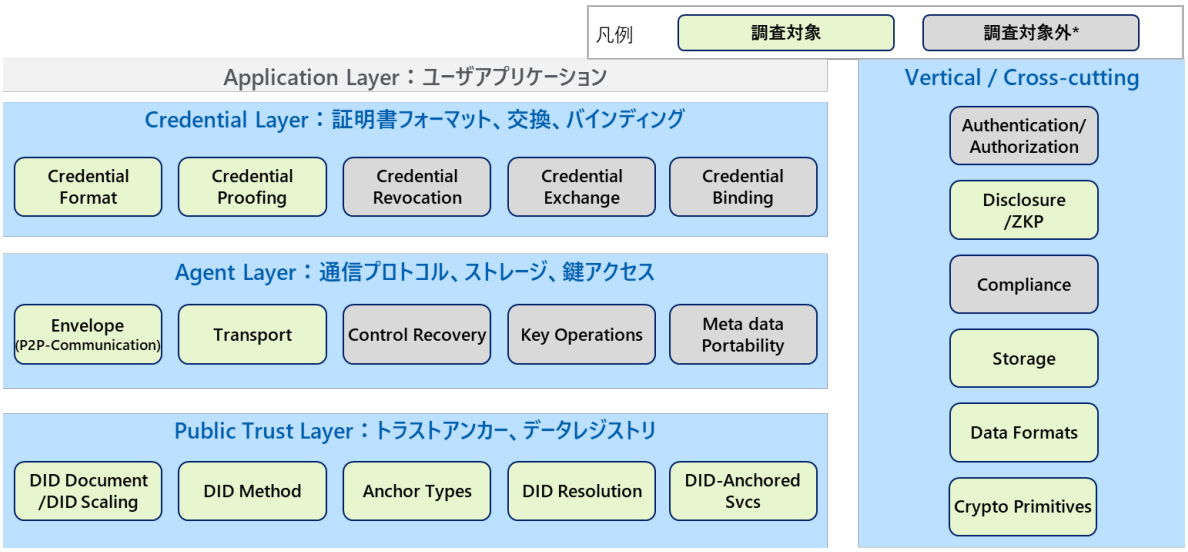


図 5-1-1　比較対象の規格

---

[63] 調査対象外とした理由

Credential Layer (Revocation / Exchange / Binding)：仕様が十分に定まっていないため調査対象外

Agent Layer (Control Recovery / Key Operations / Meta data Portability)：ハードウェア仕様がまだ定まっていないため確定要素が大きく調査対象外

Authentication / Authorization ：アプリレイヤーのため、相互運用性に影響ないため調査対象外

Compliance：技術スタックとは関係ないため調査対象外

5.2. 各実装方式の詳細

（1） 証明書を表現する際に利用されるデータフォーマットの規格

コンピュータ黎明期には低容量で高速なバイナリが一般的であったが、次第に可読性の高いテキストに置き換わっていった。証明書の利用としては JSON、JSON-LD が Web アプリで多用される JavaScript との親和性が高く利用されている。また、リソースの小さい IoT 機器等で利用する場合には CBOR の導入が進んでいる。

表 5-2-1　データフォーマット比較

	ASN.1[64]	XML[65]	JSON[66]	JSON-LD[67]	CBOR[68]
名称	Abstract Syntax Notation One	eXtensible Markup Language	JavaScript Object Notation	JavaScript Object Notation Linked Data	Concise Binary Object Representation
概要	1988 年に ITU と ISO 共同で策定された通信プロトコル用のデータフォーマット	SGML の拡張言語で、タグで挟むことで、データを表す。データ構造は XML Schema で定義	JavaScript のオブジェクト表記法のためフロントエンドでシームレスに利用できる。構造は JSON Schema 等で定義	JSON を利用したデータ形式で、Web 上のデータを構造化し、データ定義や関連性を明確に表現できる。	JSON と互換性があり、データフォーマット。認証器とクライアントを接続するプロトコルにも用いられる。構造は CDDL で定義
利用実態	(低データ量で処理が高速な)バイナリデータが特徴、黎明期において利用されていた	ASN.1 よりは可読性が高く、JSON 普及前に一般的に利用されていた	可読性が高く、フロントエンドとの接続と相性が良く、現在主流のフォーマット	Web ページのデータを構造化することで、SEO 対策に利用されている（Google 検索で推奨）	IoT 領域やスマートフォンのセキュア領域等低レイヤへの書き込みに利用されている
可読性	独自色が強く、バイナリ形式のため可読性は低い	テキスト形式であり、項目名のタグで囲むため、可読性が高いが容量が大きくなりやすい	テキスト形式であり、項目と値をコンパクトに定義できるため、可読性が高く容量が小さい	テキスト形式であり、特定のデータの意味付けや関係性を定義できるため、JSON と比較して表現度が高い	バイナリベースのため可読性は低い
可用性	通信プロトコル等に使用されることが多く対応言語を選ぶ	ほぼ全ての言語でサポート	ほぼ全ての言語でサポート	JSON-LD 固有のライブラリが必要、多くの言語で対応されている	サポートは増えつつあるが、現時点では中程度
パフォーマンス	高速で処理可能（バイナリエンコードの場合）	構造解析に時間がかかる	バイナリ形式に劣るが XML よりも高速に処理可能	JSON と比較して構造が複雑化しやすいがパフォーマンス的な差異は小さい	高速で処理可能
利用ケース	通信プロトコル(SMTP、LDAP)、データ交換	複雑な文書、SOAP、設定ファイル	Web API、設定ファイル、データ交換	検索エンジン、データ連携、セマンティック Web	IoT デバイス、WebAuthn、CTAP
証明書利用	×	×	○ (主に VC で活用)	○ (主に SEO 対策、VC で活用)	○ (主に mDL で活用)

[64] https://www.itu.int/rec/T-REC-X.680
[65] https://www.w3.org/XML/
[66] https://datatracker.ietf.org/doc/html/rfc8259
[67] https://www.w3.org/TR/json-ld/
[68] https://tex2e.github.io/rfc-translater/html/rfc8949.html

（２） アイデンティティサービスを構築する場合に基準とされる証明書モデルの規格

Verifiable Credential (VC) と Verifiable Presentation (VP) は、デジタルアイデンティティを確認し、信頼性を担保するための概念で、これらは W3C (World Wide Web Consortium) により仕様が標準化[69]されている。

VC (Verifiable Credential)

特定の主張（例えば、個人の名前や年齢、組織の所在地等）について、それが信頼できる発行者から発行されたものであることを証明するデジタル証明書の一種である。

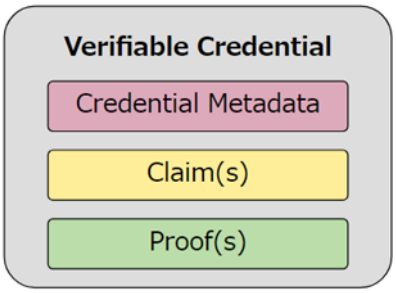


図 5-2-1　W3C で定義している VC の基本構成

VP (Verifiable Presentation)

一つまたは複数の VC を保持し、それを他のエンティティに提示するためのパッケージ。VP は、提示者(通常は Holder)によってデジタル署名され、その署名で提示者と VC の真正性が確認できる。

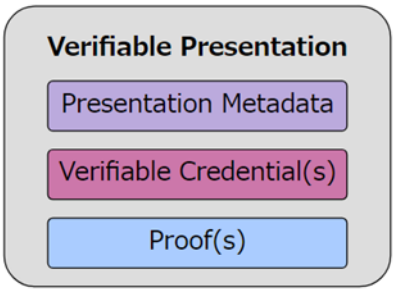


図 5-2-2　W3C で定義している VP の基本構成

また、OpenID Foundation で策定が進められている OpenID for Verifiable Credential (OID4VC) は以下の 3 つの仕様で構成されている[70]。W3C で定義されている仕様はエンティティ間のプロトコルまで定義されておらず、実装者にゆだねられているため、プロトコルの標準化を進めている。

表 5-2-2　OpenID Foundation で策定されている OID4VC の仕様

OpenID for Verifiable Credential Issuance(OID4VCI)	・ 検証可能な資格情報（VC）を発行するための API と対応する OAuth ベースの認定メカニズムを定義
OpenID for Verifiable Presentations(OID4VP)	・ OAuth2.0 上にプロトコルフローの一部として検証可能な資格情報の形式でクレームを定義できるようにするメカニズムを定義

[69] https://www.w3.org/TR/vc-data-model/

[70] https://openid.net/sg/openid4vc/

Self-Issued OpenID Provider v2 (SIOPv2)	・ 自己主権型アイデンティティとしてエンドユーザーが自分で管理する OpenID プロバイダーを使用できるようにする

現時点でアイデンティティサービスを構築する場合、基準とされる証明書モデルの規格は、W3C で標準化された VC/VP と、OpenID Connect の通信/認証プロトコルに対応している OID4VCI/OID4VP の 2 つに大別される。

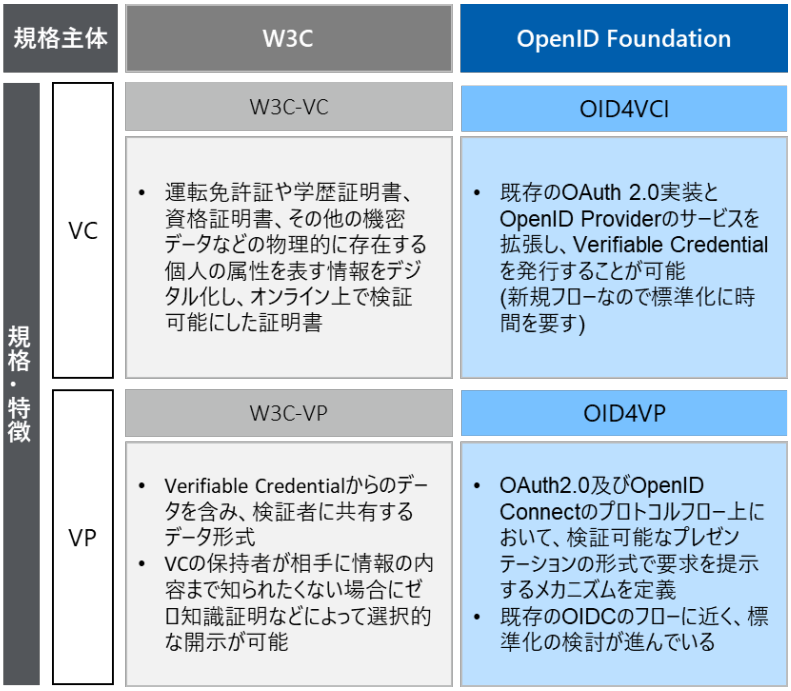


図 5-2-3　証明書モデルの比較

OID4VP は既存の OpenID Connect のフローと類似しており(ID トークン付与のプロセスに VP を追加するイメージ)、標準化が進んでいるが、OID4VCI は新規フローであるため、規格検討に比較的時間を要している。

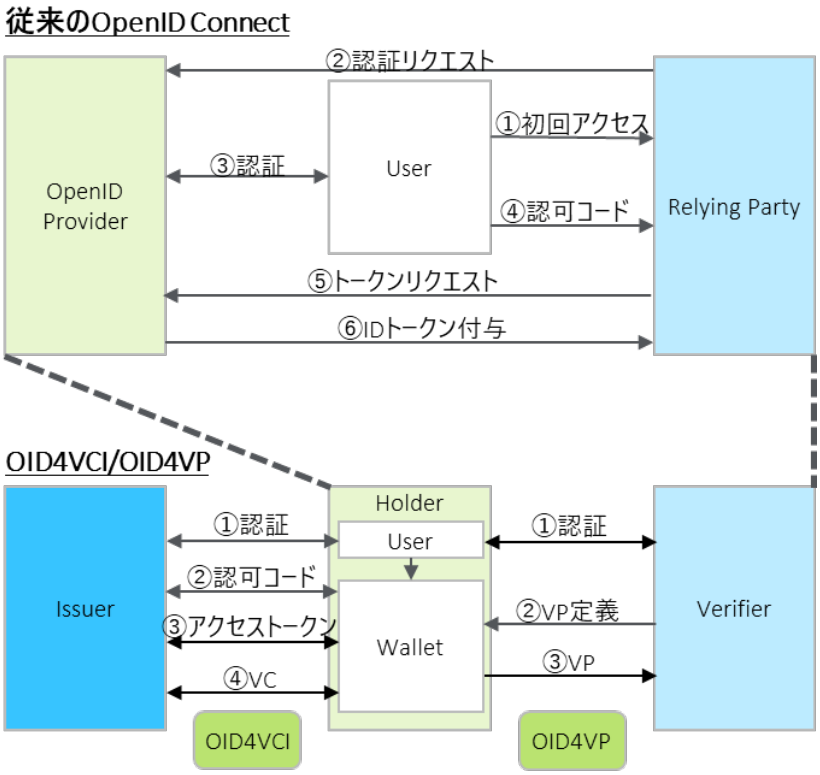


図 5-2-4　OID4VCI/OID4VP の拡張イメージ

（3） 選択的開示含む証明書検証のための規格

証明書のデータ形式としては、JSON が大部分を占めており（一部 JSON をバイナリ形式も存在）選択的開示を容易にする拡張仕様が複数規格が存在している[71]。

[71]証明書規格・署名アルゴリズムの組み合わせは下記サイトで取り上げられているものを記載
「A credential profile comparison matrix to facilitate technical and non-technical decision making」
https://github.com/vcstuff/credential-profile-comparison?tab=readme-ov-file
mDL にゼロ知識証明を活用した事例は現在確認例が少なく一般的でないが、CYBERNETICA 社の取組を確認している
https://cyber.ee/uploads/Zero_Knowledge_Proofs_report_89d6bc5438.pdf

証明書規格	データフォーマット	通常採用される署名アルゴリズム	特徴	選択的属性開示	
				データ最小化	ゼロ知識証明
JWT VC（JSON Web Token）	JSON JSON-LD	ECDSA EdDSA	• OAuth2.0およびOpenID Connet等のフレームワークで認証トークンとして広く使用されているJWTをベースにしているため、開発者が実装する上で必要なライブラリやツールが豊富に存在する • プレーンJSONを用いたJWTベースのVCは暗号化とセキュリティのために既存のJOSEフレームワークを使用しており、コンテキストをうまく表現できないため、選択的属性開示機能の実装が困難	×	×
SD-JWT VC	JSON JSON-LD	ECDSA	• 選択的開示を実現、実データにソルトを加えて生成したハッシュ群に対し署名を行うことで、実データ自身を送付せずに、署名検証できる	○	×
LDP-VC（Linked Data Proofs）	JSON-LD	BBS+	• プレーンJSONでは表現が困難なコンテキストを定義可能な形式。Linked Dataによりデータの記述に使用する用語（URI）を一意に特定できる • 選択的開示はBBS+署名・ゼロ知識証明で実現する	○	○
AnonCreds	AnonCreds (JSON)	CL	• Hyperledger AnonCredsプロジェクトで検討している形式 • 選択的開示はゼロ知識証明+CL(Camenisch-Lysyanskaya)署名を使用する	○	○
mDL	CBOR	ECDSA	• 運転免許証の文書形式(mdoc)を拡張して、他ユースケースでも活用できるようにしたもの	○	△

図 5-2-5　証明書フォーマット一覧

（4） 選択的開示方式の比較

選択的属性開示手法として適用のし易さを重視した SD-JWT、厳密性と拡張性を重視した JSON-LD(BBS+)が代表的である。以下に特徴を記載する。

SD – JWT

【概要】

- 実データにソルトを加えて生成したハッシュ群に対し署名した SD-JWT と、実データを格納した SVC を用意。
- SVC はそのまま Holder が保管し、Verifier に送付する際に SVC の Claim を選択し、Holder の署名をつけて開示する。
- SD-JWT-R のデータのハッシュ値と、SD-JWT に格納された値の同一性を確認することで真正性を検証。

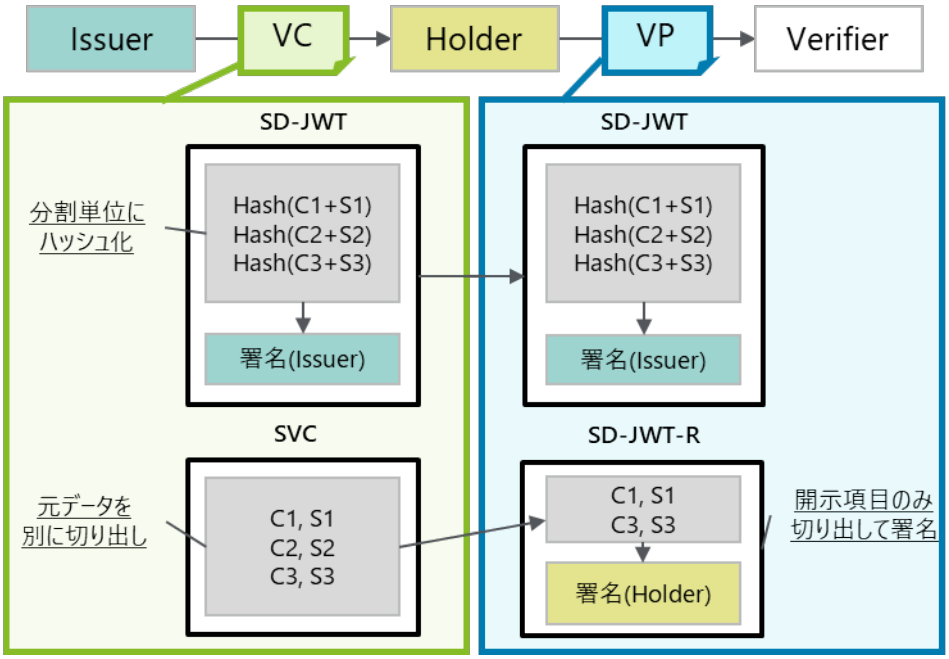


図 5-2-6　SD-JWT を活用した選択的属性開示の実現イメージ

【評価】

- ユーザが毎回同じ署名を利用するため、複数の検証処理を照合することで利用者特定が可能であり、Unlinkability を担保できない
- Web システムのトークン仕様として、広く普及しているフォーマットのため、システム実装者の理解度、既存システムへの適用性が高い

JSON – LD + BBS+署名

【概要】

- マルチメッセージのデジタル署名の一種。通常、秘密鍵を使用してメッセージ全体を署名するが、マルチメッセージシステムでは、秘密鍵で署名されたメッセージをより小さな属性に分割して共有および検証できる。
- そのため、Holder は署名されたデータを項目ごとに切り出して開示することが可能。ゼロ知識証明との組み合わせが可能。

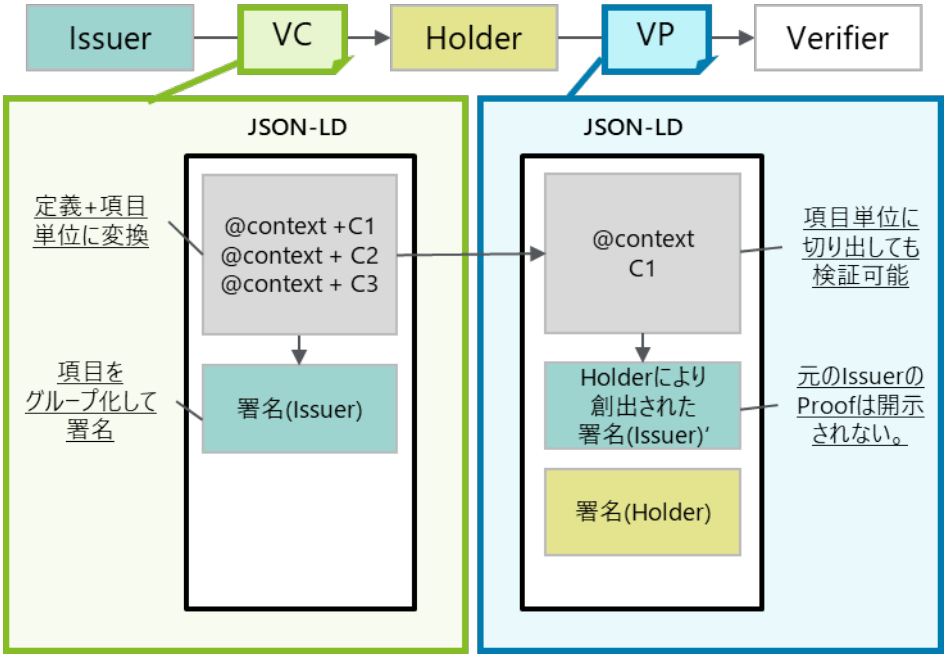


図 5-2-7　JSON-LD + BBS+署名を活用した選択的属性開示の実現イメージ

【評価】

- ゼロ知識証明と組み合わせることで、署名検証の都度新しい証明を作成できるため、Unlinkability を担保できる。
- 新しいフォーマット仕様であるため、普及には時間がかかる可能性が高い。

（5） エンティティ間で証明書データを連携する規格

DID Comm は一般的なメッセージングの仕様を定めており、SIOPv2 は認証プロセス全体を規定している。

表 5-2-3　エンティティ間で証明書データを連携する規格の比較

規格	規格団体	特徴
DIDComm Message v2	DIF / Hyperledger	➢ DID ベースのシステム上でセキュリティで保護されたプライベート通信方法を提供する。 ➢ DID を宛先とする非同期単方向のメッセージングプロトコル仕様で、DID の持ち主へのメッセージ配送にはサービスエンドポイントというエンドポイント(URI)を使用する。このエンドポイントは DID Document 内に任意に設定され、特定のフォーマットのメッセージを通じて DID の持ち主にメッセージをエンドツーエンドで届ける役割がある。 ➢ DIDComm の特徴は、具体的な配送方法を規定しておらず、あらゆる方法や中継者を通じての配送が許容される。ただし、メッセージの到達確認や返信保証はない。
SIOPv2	OpenID Foundation	➢ OpenID Connect を拡張し、エンドユーザーによる制御領域を拡大するもの。具体的には、 エンドユーザーが Self-Issued Open ID Provider（SIOP）を制御し、エンドユーザーの 制御下にあるキーで暗号化した ID トークンを自己発行して、自分自身を認証することができる 。 ➢ エンドユーザーはクライアント（RP）に発行された識別子（ID）と資格情報（Claim）を制御でき、いつ、どのように使用するかをユーザーが決定できる。 ➢ 仕様の中では、発行する識別子や資格情報について W3C の DIDs、VCs を参照している

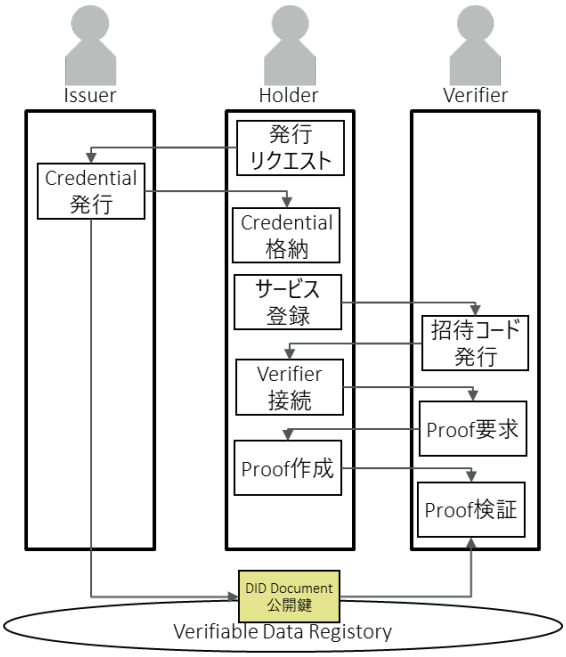


図 5-2-8　データ交換の流れ

（６） デバイス連携のための規格

デバイス連携の手法としては、QR コード/NFC/Bluetooth（BLE）等の複数デバイス間（Cross Device）の連携と、同一デバイス内（Same Device）に大別される。

QR コードは付属カメラ等で情報を取り込むため特別な拡張は不要。BLE や NFC は通信プロトコルレベルの拡張が必要になる。従って、具体的な実装には拡張仕様もしくは業者が独自に設計する必要がある（OID4VP においては、VP 交換の Bluetooth 拡張機能が存在）。同一デバイス内であればアプリ間での標準的な遷移で対応可能だが、どの方式においてもプライバシーを保護する適切な制御が必要となる。

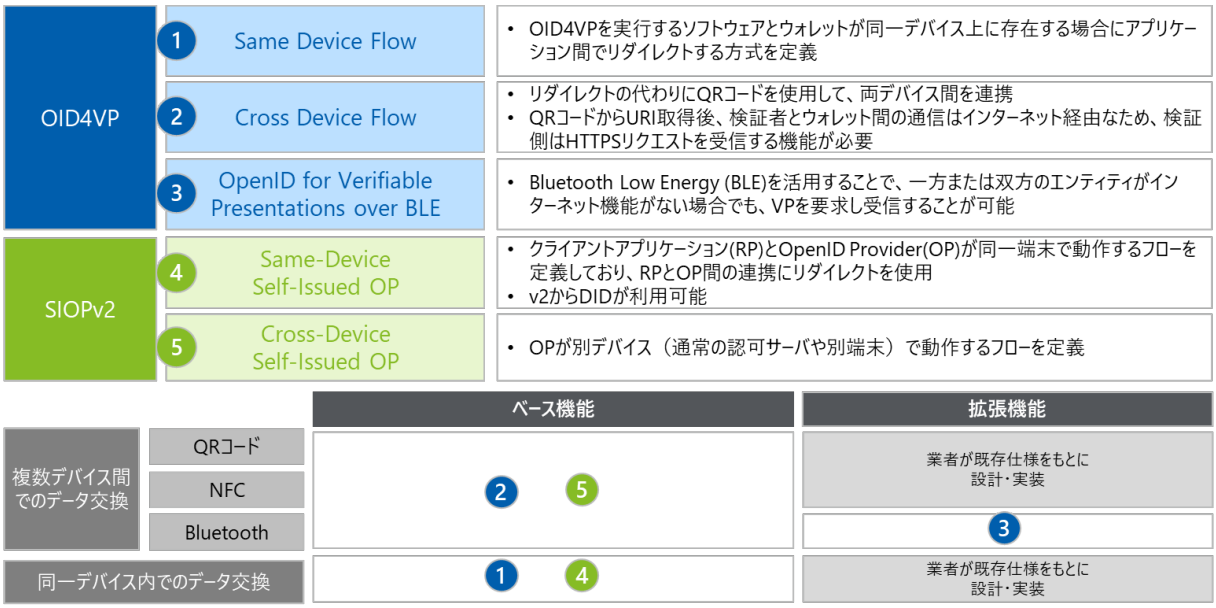


図 5-２-8　デバイス連携規格の比較

（7） DID Document を活用した証明書の検証方法

DID Document と VC の紐付けは DID からシステム内の DID Document を特定し、VC 内の Proof セクションに指定された公開鍵で検証を行う。

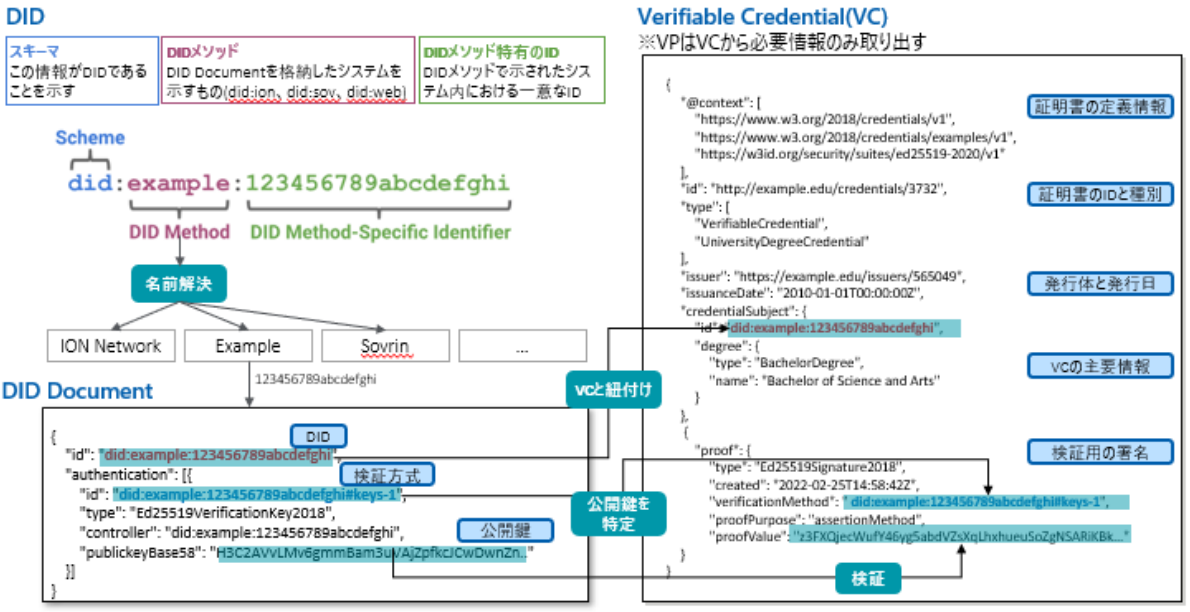


図 5-2-9　DID Document を活用した証明書の検証方法

（8） DID Doucument の格納場所

DID Document の格納し情報共有する方式は、大きく分類して①ブロックチェーン、②WEB サーバ、③P2P 通信の 3 通りある

①ブロックチェーン

- Issuerが特定のブロックチェーン上にDID Documentを保存し、Verifierはそこを参照してVCの真正性を確認
- ブロックチェーンが耐改ざん性、可用性を担保してくれるため、Data Registryとしては一般的な方式（安全性・機密性の観点からPermissionedなブロックチェーンを用いられることが多いが、スケーラビリティに課題がある）

②WEBサーバ

- Issuerが特定のWebServerに公開鍵を含むDID Documentを保存し、Verifierはそこを参照してVCの真正性を確認
- Webサーバに格納されたDID Documentの耐改ざん性や可能性の担保が課題

③P2P通信

- Verifierの要求に応じてIssuerがDID Documentを直接送信
- 直接送付するため秘匿性は高い（DID Documentに秘匿性の高い情報を載せる場合等には有効）
- 送信方法にも依存するが、方法は複数存在し確立していない（メール、FTP...）

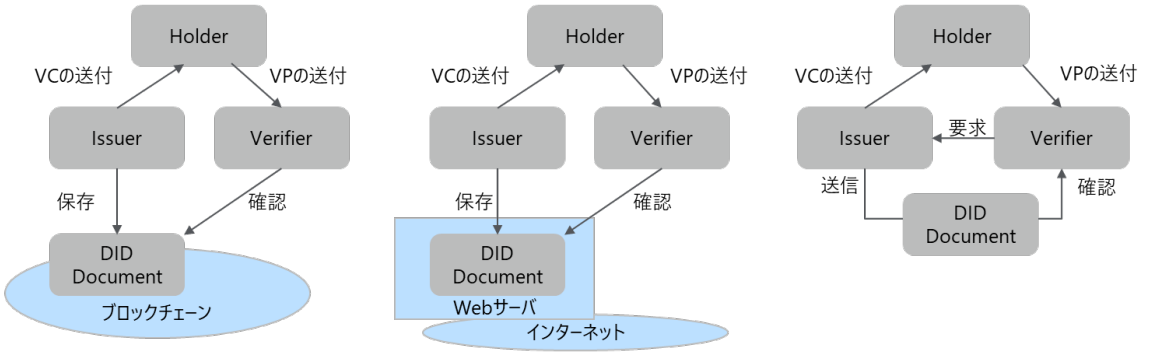


図 5-2-10　DID Document の格納場所

参考：DID Document を活用しない方式

DID は複数のプラットフォーム間で、一意に ID を識別する仕組みのため、複数プラットフォーム連携ニーズが無い場合、VC は DID および DID Document を適用しなくても成立する。

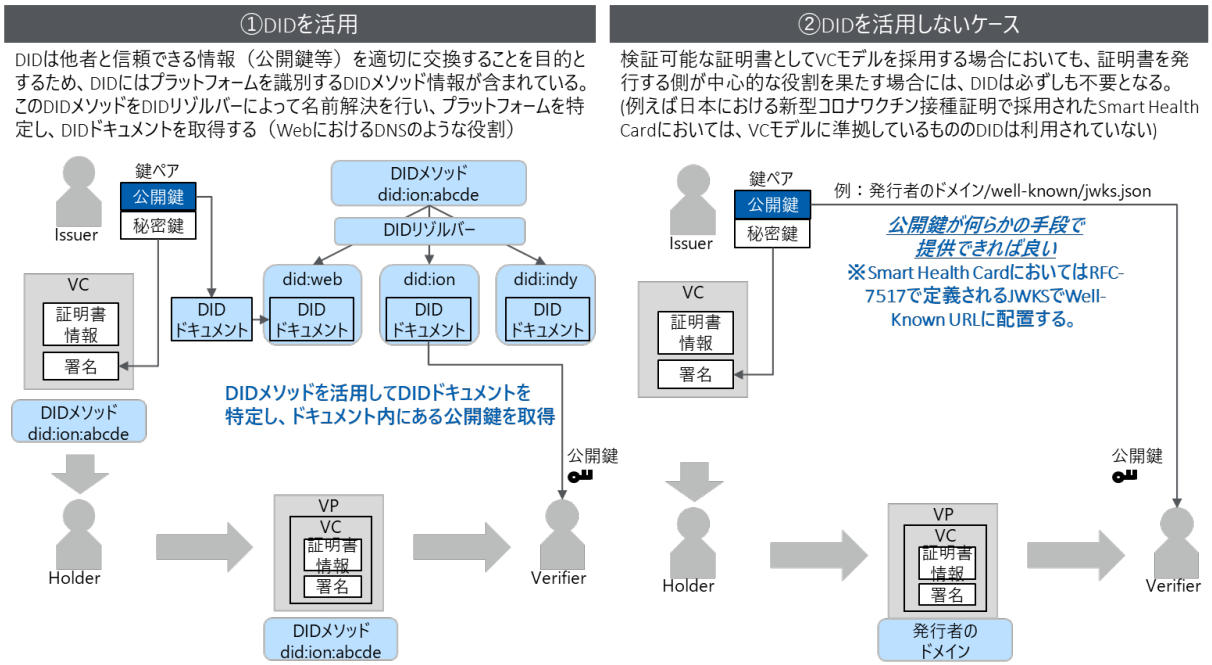


図 5-2-11　DID Document を活用しない方式

（9） ブロックチェーン比較

分散型アイデンティティに使用されるブロックチェーンは不特定ユーザとの検証が前提となるため、全公開が基本となる。ブロック作成、検証行為のみが許可制となる場合がある。

図 5-2-12　ブロックチェーン比較

表 5-2-4　アイデンティティ管理等で活用する主要ブロックチェーン比較

	Ethereum[72,73]	Hyperledger Indy[74]	Hyperledger Fabric[75]
VC 活用事例	Civic Pass	BC Digital Trust/NB Orbit	Interac Verification Service
総評	■ 参加制限がなくデータは全公開のため、秘匿情報を別に扱うオフチェーンの仕組みが必要 ■ アカウント失効等の実装が課題になり標準化が停滞している状況 ■ 近年では SBT に統合されて仕様検討が進んでいる ■ 自己管理の考え方が強く、NFT や DAO 等他の Web3 サービスとの連携に活用可能性がある	■ 分散型アイデンティティ管理に特化しており、クレデンシャル定義や失効レジストリ等の機能を有す ■ ネットワークへの参加は許可制であるが、情報は全公開であり、クライアント用のツールキットの Aries で、エンティティ間のメッセージ交換や格納を制御 ■ 信頼性と運用性のバランスが取り易いが、コンソーシアム運営が課題	■ エンタープレイズ利用を前提とした汎用プラットフォーム ■ チャネルやプライベートトランザクション等機能を搭載しており秘匿情報をブロックチェーン上で一元的に扱える利点あり ■ アクセス制御の組合せが多いとデータ容量・運用面の設計の複雑さや、Indy と同様コンソーシアム運営などが課題
参加者制限	×	○	○
ノード保有者のデータアクセス制御	×	×	○
スマートコントラクト有無	○	×	○
スループット(TPS)	10～15	100	400
コンセンサスアルゴリズム	Proof of Stake	Plenum	Raft & Endoring-Ordering-Validation

[72] https://ethereum.org/ja/
[73] https://github.com/ethereum
[74] https://www.hyperledger.org/projects/hyperledger-indy
[75] https://www.hyperledger.org/projects/fabric

参考：ブロックチェーン領域で活用される秘匿化技術

機密性やプライバシーレベルが高い情報を秘匿化する技術（PET：privacy-enhancing technologies/techniques）は多数存在し、大きくは 3 つに分類される。また、パブリックチェーン / プライベートチェーンで活用される技術は異なる。

	共有先制御型PET	非可読化型PET	関係性隠匿型PET
説明	各参加者がネットワーク上の全取引の一部にしかアクセスできないようにする手法	暗号化技術を用いることで、第三者が取引情報を解釈できないようにする手法	台帳に記録された送金者・受領者情報から、第三者が取引当事者を特定することを困難にする手法
パブリック/プライベートチェーン双方で活用される技術	**ペイメントチャネル** オフチェーンで取引することで、秘匿性を強化する仕組み。参加者は個々の取引をネットワーク全体にブロードキャストすることがない	**ゼロ知識証明（ZKP）** データを公開することなく、そのデータの真実性のみを証明する。VCでは、BBS＋署名と組み合わせて取引情報の秘匿化が行われる **準同型暗号** データを暗号化したまま計算可能な暗号方式	**秘密分散** データを断片化して一定以上の断片が揃わないと復元できない仕組み **リング署名** 複数人が記載された公開鍵リストがあり、その中に署名者が存在することを保証する(誰であるかは特定できない) **グループ署名（BBS+署名/CL署名）** あるグループに属する者の署名からはそのグループに属することしかわからず、グループ管理者だけが特定できる。VCでは、ゼロ知識証明と組み合わせて取引情報の秘匿化が行われる
プライベートチェーンで主に活用される技術	**Flow Framework（Corda）** 取引当事者間のみデータを共有するトランザクションを発行可能 **Channel（Hyperledger Fabric）** グループ単位にブロックチェーンを分割する方式。トランザクション送信時にチャネルを指定することで、任意のグループのみデータを共有可能	**Private Transaction（Quorum）** トランザクションに秘匿情報を含まないハッシュ等のみを格納（アンカリング）し、データの実体は指定ノードのみ共有する方式 **Private Data Collection（Hyperledger Fabric）** トランザクションに秘匿情報を含まないハッシュ等のみを格納し、データの実体は指定ノードのみ共有する方式。チャネルより細かい単位で共有範囲を指定	–

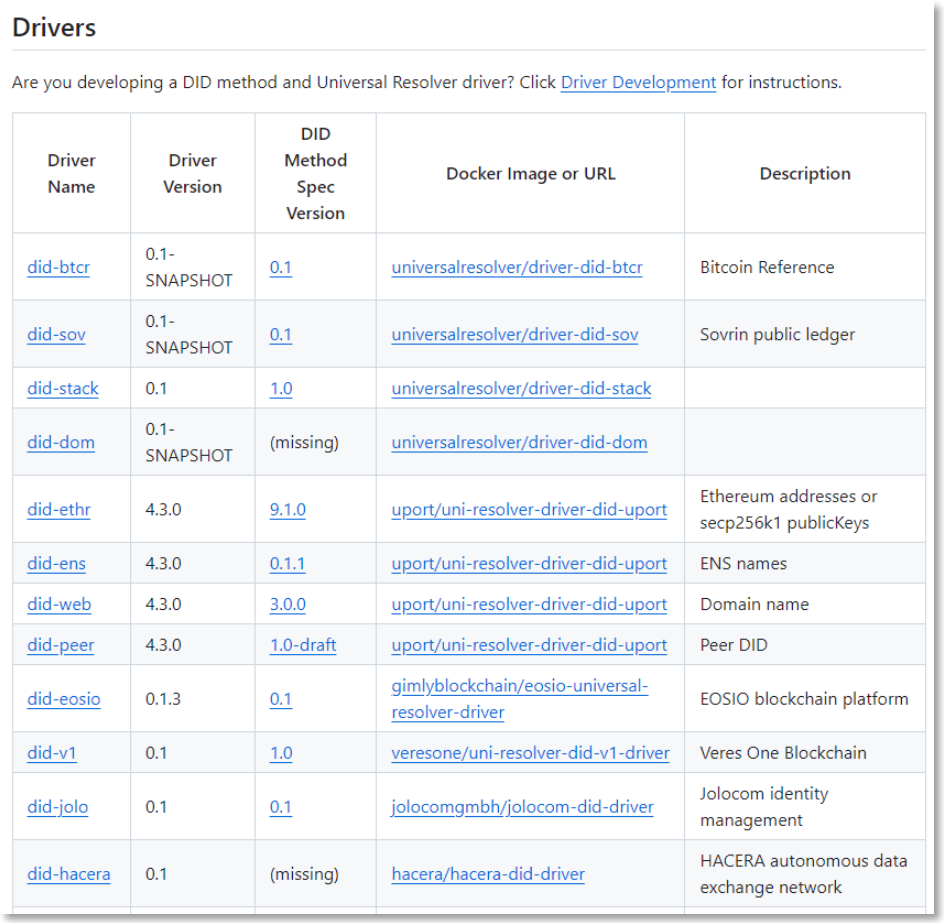
図 5-2-13　ブロックチェーン領域で活用される秘匿化技術 [76]

[76] https://www.boj.or.jp/paym/fintech/rel200212a.htm をもとに TOPPAM 作成

（１０） Universal Resolver

DID Method を識別し、名前解決を行う仕組みとして実質的な標準となっているのが Universal Resolver。DID Method をサポートするためにプラグインとなる Driver を追加する。

- W3C が制定する DID Core 1.0 および DID Resolution 仕様に基づいてさまざまな DID メソッドに対応した名前解決や DID の登録を行う。
- Docker イメージで提供され REST API を利用して動作する。ドライバー形式を採用しており、特定の DID メソッドに対応したドライバーを組み込むことでサポートするメソッドを拡張できる。
- 現時点で 60 個以上のドライバに対応しており、標準化はまだされていないものの、事実上のディファクトスタンダードとなっている。
- 現在は DIF Identifiers & Discovery Working Group の作業項目となっている



Drivers

Are you developing a DID method and Universal Resolver driver? Click Driver Development for instructions.

Driver Name	Driver Version	DID Method Spec Version	Docker Image or URL	Description
did-btcr	0.1-SNAPSHOT	0.1	universalresolver/driver-did-btcr	Bitcoin Reference
did-sov	0.1-SNAPSHOT	0.1	universalresolver/driver-did-sov	Sovrin public ledger
did-stack	0.1	1.0	universalresolver/driver-did-stack	
did-dom	0.1-SNAPSHOT	(missing)	universalresolver/driver-did-dom	
did-ethr	4.3.0	9.1.0	uport/uni-resolver-driver-did-uport	Ethereum addresses or secp256k1 publicKeys
did-ens	4.3.0	0.1.1	uport/uni-resolver-driver-did-uport	ENS names
did-web	4.3.0	3.0.0	uport/uni-resolver-driver-did-uport	Domain name
did-peer	4.3.0	1.0-draft	uport/uni-resolver-driver-did-uport	Peer DID
did-eosio	0.1.3	0.1	gimlyblockchain/eosio-universal-resolver-driver	EOSIO blockchain platform
did-v1	0.1	1.0	veresone/uni-resolver-did-v1-driver	Veres One Blockchain
did-jolo	0.1	0.1	jolocomgmbh/jolocom-did-driver	Jolocom identity management
did-hacera	0.1	(missing)	hacera/hacera-did-driver	HACERA autonomous data exchange network

図 5-2-14　対応ドライバリスト [77]

[77] https://identity.foundation/edv-spec/

（１１） 証明書を安全に格納するサービス規格

暗号化データ保管庫（EDV：Encrypted Data Vault）は個人や企業が所有(契約)するクラウドストレージやモバイルストレージ等のデータを安全に格納、インデックス、共有するためのメカニズムを定義する仕組みでありW3C で検討されている。

ユーザや企業（エンティティ）はストレージプロバイダーに内容を知られることなくデータを格納可能で、かつプロバイダーの管理者がアクセスできなくなる仕組みを提供する。クライアントは DID に紐付けられたキーを使って独自の暗号化・復号を行うため、クライアントが参照先を完全に管理できる。

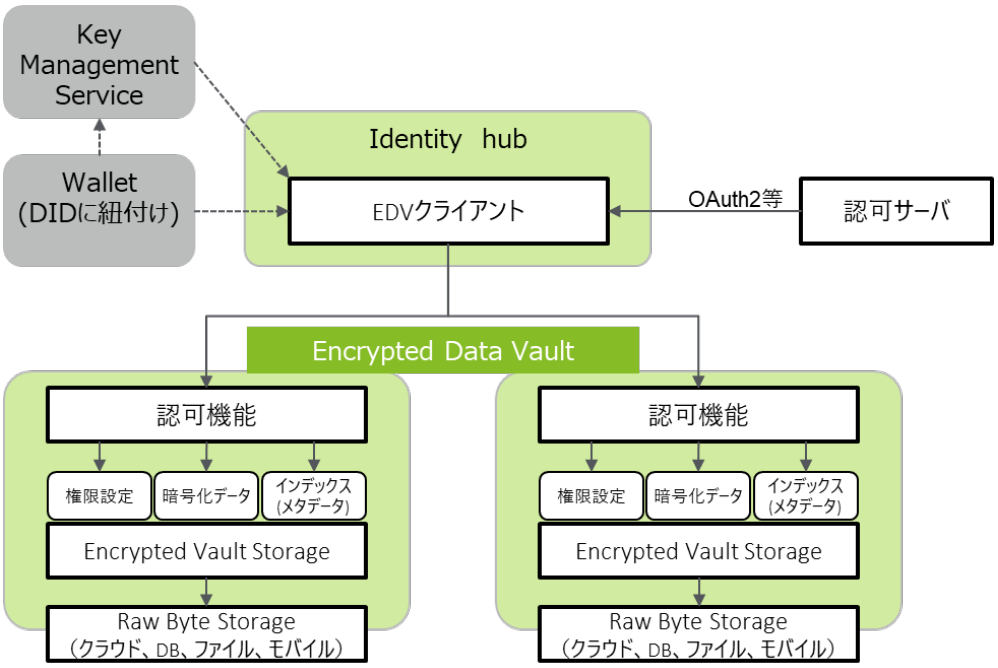


図 5-2-15　EDV 概要 [78]

[78] https://identity.foundation/edv-spec/

5.3. mDL と VC の比較

（1） 全体比較

mDL と VC は両規格とも所有者が自身の資格情報を制御できるが、mDL はデータフォーマットや通信プロトコルまで厳密に定義されており、資格情報管理も発行機関が一元的に管理できる選択肢を残している点がもっとも大きな相違点としてあげられる。

	mDL	VC
サマリ	各エンティティの役割、処理フロー、データモデル（基本項目）からハードウェア、通信規格まで厳密に決められており、最低限のサービスの互換性を担保	制御に必要な必要最低限のデータ項目のみ定義されており、ハードウェアやソフトウェアの詳細仕様が、明確に定義されていないため個別に解釈して独自実装するか、関連団体(OIDFやDIF等)に移譲
アーキテクチャ	• 発行機関のサーバから資格情報を取得することも可能であるが、その場合、発行者はいつ・誰によって使用されたか知ることができる（事前同意は行う） • オプションとして証明性の検証用にVICALによるルート証明書を配布できる(失効確認)	• VDRに検証のためのメタ情報を格納し、所有者のデバイスに資格情報を格納することが前提
データモデル	• データモデルとして規定された用途(運転免許)がある • フォーマットはCBOR(※)またはJSON • 基本名前空間と拡張用名前空間を持つ	• 汎用的なモデルとして定義されている • 基本的な資格項目は存在しない • JSONLDまたはJSONが標準であるが、他の形式でも利用可能
通信プロトコル	• mDLとリーダー、発行機関とリーダーとのI/Fを明確に定義(デバイス接続：NFCまたはQRを使用、通信I/F：WiFi、BLE、NFCを使用) • サーバアクセスはAPI/OIDCのエンドポイント使用	• 資格情報の交換に必要なインターフェイスは定義されていない
セキュリティ	• セッション暗号化（AES） • MSO(Mobile Security Object)に格納された情報に付与したデジタル署名を検証	• 特定形式のデジタル署名は未定義 • 全てのエンティティがVDRを信頼する必要がある。ただし、VDRに必要な条件は規定されていない
共通事項	• 発行した資格情報を所有者の管理下にあるデバイスまたはレジストリに格納する • 資格情報は完全に所有者に制御されて、検証者に提供する判断は所有者にある • 発行者は所有者の同意なしに、認証情報を検証者に直接公開できない • 所有者がいつ・どこでmDLを使用するか発行者・検証者は認識できない	

図 5-3-1　mDL と VC 比較 [79]

[79] mDL の規格(ISO シリーズ)、Verifiable Credentials の規格をもとに TOPPAN 作成

※mDL について詳細を確認したい場合、ISO サイト(ISO 18013-5：https://www.iso.org/standard/69084.html)から購入すること

参考：Unlinkability とは (ISO/IEC 27551[80]より)

個人のプライバシーを尊重する上では、Unlinkability が担保されていることに留意する必要がある。VC や mDL を活用する目的の一つに個人のデータを最小限にしてプライバシーを保護して自身の資格・属性証明を行うことが挙げられるが、Unlinkability が担保されていない場合はその目的を達成できない可能性があるため、サービスを実装する上では、Unlinkability 担保を踏まえて証明書連携フローを設計することが重要である。

【背景】

ISO/IEC 29100 で定めている個人識別情報(PII)の処理に関わるアクターに適用されるプライバシー原則の中の 1 つに「収集の制限」があるが、現在は、インターネットサイトがサービスへのアクセス時に必要以上の情報を収集することが一般的となっている。

例えば、Web サイトが PII の主体が特定の年齢以上であることのみを確認したいときに、ユーザーの永続的識別子などの不必要な情報まで取得していることが挙げられる。これによって同一の PII 主体による異なるサイトへの訪問や、同一サイトへの二回以上の訪問をリンクすることが可能となっており「収集の制限」の原則が達成されていない状況である。

収集の制限の原則に従うためには、上記のケースのサイトは PII 主体による二回以上の訪問をリンクさせないタイプのエンティティ識別子を使用すべきで、二つのトランザクションが行われた場合、それらのトランザクションが同一ユーザーによるものか、異なる二人のユーザーによるものかを区別できないこと(Unlinkability の確保)が求められる。

属性ベースのリンク不可能なエンティティ認証(ABUEA：attribute-based unlinkable entity authentication)は、PII 主体が、Unlinkability を確保したうえで、身元属性情報の真正性を確立する手段を提供することができこの関連技術標準をまとめたものが ISO/IEC 27551(Information security, cybersecurity and privacy protection — Requirements for attribute-based unlinkable entity authentication)で規定されている。

【VC/mDL を活用する際に留意すべき事例】

証明書の署名値を同一のものを使用した VC 証明書を Holder が Verifier に証明書の提示を行うと、Verifier の結託によってどこに情報提示したか類推される。

Server Retrieval で証明書の失効問い合わせを Issuer にすると、Issuer は、特定の Holder と Verifier がやり取りしたかが類推できる。

[80] Unlinkability について詳細を確認したい場合、ISO サイト(ISO/IEC27551：https://www.iso.org/standard/72018.html)から購入すること

6．実装パターンの抽出

6.1．サービス

サービス実装パターン詳細

（1）JWT-VC・SD-JWT・LDP-VC

総評　パターン数（112）＝ Credential Layer（14）× Agent Layer（8）× Public Trust Layer（NA）

Credential Layer の証明書フォーマットおよび署名アルゴリズムの暗号方式は下位レイヤに影響しないため、組合せのバリエーションが多い。検証者に提供する証明書の項目を選択的に開示するため既存フォーマットで対応するか、より高機能かつ柔軟性の高い組合せにするかで異なる。

Credential Layer　※14 パターンであると整理

W3C に準拠しつつ独自に実装するパターンと OIDF で規定している OID4VP に対応する 2 パターンに大別される。証明書のフォーマットは選択的開示対応を既存フォーマットで行うか、厳密に構造化された新しいフォーマットで行うかで別れる。証明書と署名方式の組み合わせは事例ベースで確認した[81]。

Agent Layer　※8 パターンであると整理

JWT-VC、SD-JWT、LDP-VC は別レイヤの実装に影響しないものの、DIF で規定する DIDComm か、OIDF の自己発行を規定した SIOP に大別される。デバイス連携・鍵管理については明確に定義されていないため、自前で実装が必要。

Public Trust Layer　※N/A であると整理

JWT-VC、SD-JWT、LDP-VC は別レイヤの実装に影響しないため、本レイヤにおける組合せはプラットフォームごとに異なり、複数存在するため、パターンを絞り込むことが出来ない。

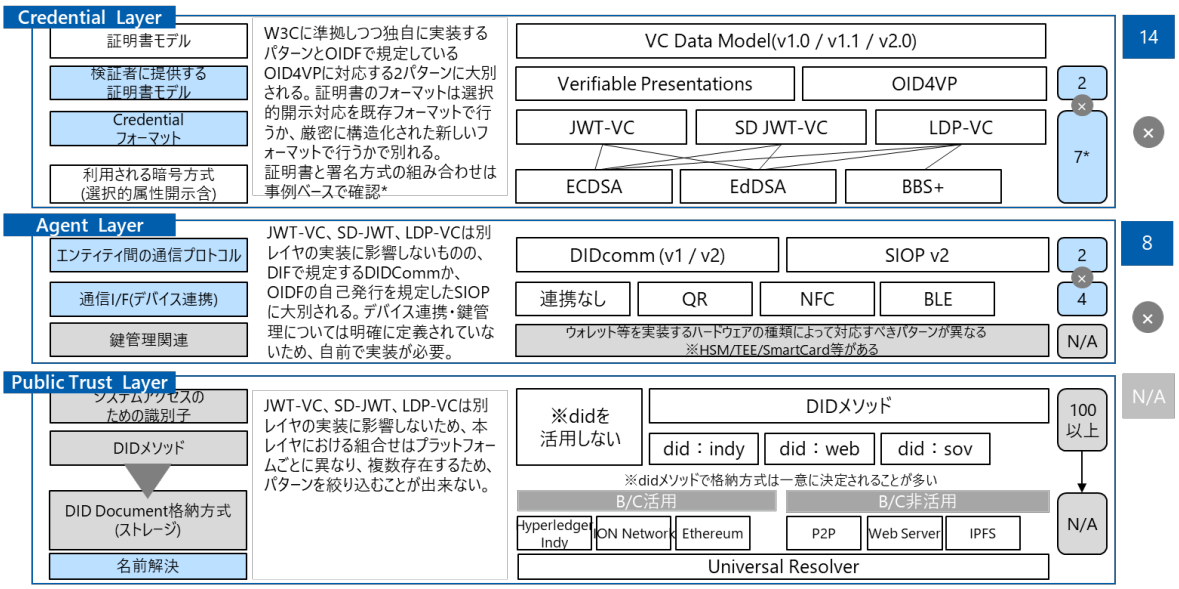


図 6-1-1　JWT-VC・SD-JWT・LDP-VC を活用した実装パターン整理

[81] https://docs.google.com/spreadsheets/d/1X93ptJcmfX1NZEo5E7ElnqJ-knDS4Dj6JOYSJ_2PsUw/edit#gid=1590639334

（２）AnonCreds

総評　パターン数（64）＝ Credential Layer（4）× Agent Layer（4）× Public Trust Layer（4）

AnonCreds は Hyperleder プロジェクトで検討されている。レガシー仕様では Hyperledger Indy を前提に検討されていたが、近年では様々なプラットフォームに対応することが検討されている。ただし、前提となるバリエーションはそれほど多くない。

Credential Layer　※4 パターンであると整理

当初は CL 署名のみ対応していたが、AnonCreds v2 において、PS 署名に対応し、BBS+署名もサポートしている。また、ポスト量子オプションも検証している [82]。

Agent Layer　※4 パターンであると整理

証明書を安全に伝達するためのプロトコルとして DIDComm が定義、デバイス連携は対象外、鍵管理は aries-askar で検討している [83]。

Public Trust Layer　※4 パターンであると整理

DID メソッドは AnonCreds Methods Registry[84]にて定義されている、レガシー仕様として Hyperledger Indy を利用する場合には did:indy が推奨されているが、新規としては 4 種類定義されている。HTTP は did:web とほぼ同一である。

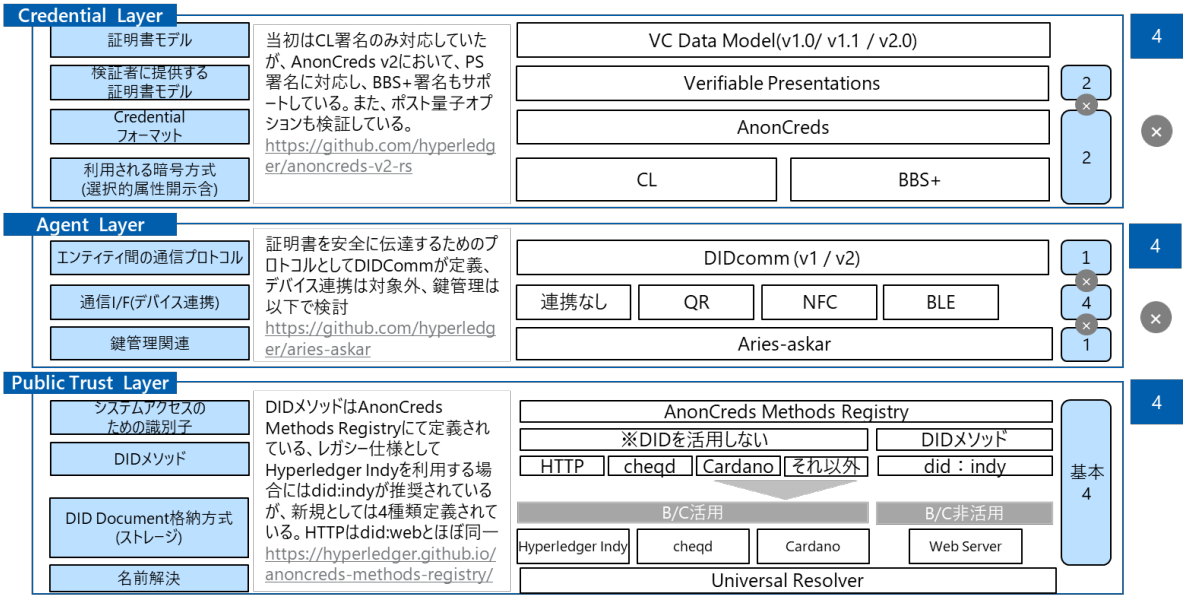


図 6-1-2　AnonCreds を活用した実装パターン整理

---

[82] https://github.com/hyperledger/anoncreds-v2-rs

[83] https://github.com/hyperledger/aries-askar

[84] https://hyperledger.github.io/anoncreds-methods-registry/

（3）mDL

総評　パターン数（144）＝ Credential Layer（12）× Agent Layer（4）× Public Trust Layer（3）

mDL は ISO にて厳密に処理方式からフォーマット、デバイス連携、エンティティ間のデータの受け渡し方法等を明確に定義しているため、ISO 内ではそれほどバリエーションがないが、仕様そのものは特定のフォーマットや実装に依存しないため、複数の仕様に適用させることが可能。

Credential Layer　※12 パターンであると整理

ISO/IEC 18013-5 ではデジタル運転免許証の提示と検証に関する一連のプロトコルとデータ交換フォーマットを定義されており、証明書モデルは実装や具体的なアプリケーションに依存する。署名アルゴリズムは ECDSA および EdDSA が規定されている。

Agent Layer　※4 パターンであると整理

ISO で定めた通信プロトコル[85]があるものの、他の方式での実装も可能である。デバイス連携も想定されているため、組合せは多い。

Public Trust Layer　※4 パターンであると整理

mDL においては、Issuer Athority が Holder に対し証明書を直接送信するか、またはサーバに格納するかの 2 つの選択肢が存在する。

ストレージはローカルかサーバか選択可能だが、サーバの派生で B/C に格納することも特定条件をクリアすることで可能である(通信暗号化、JWS 署名対応、アクセス制御)。

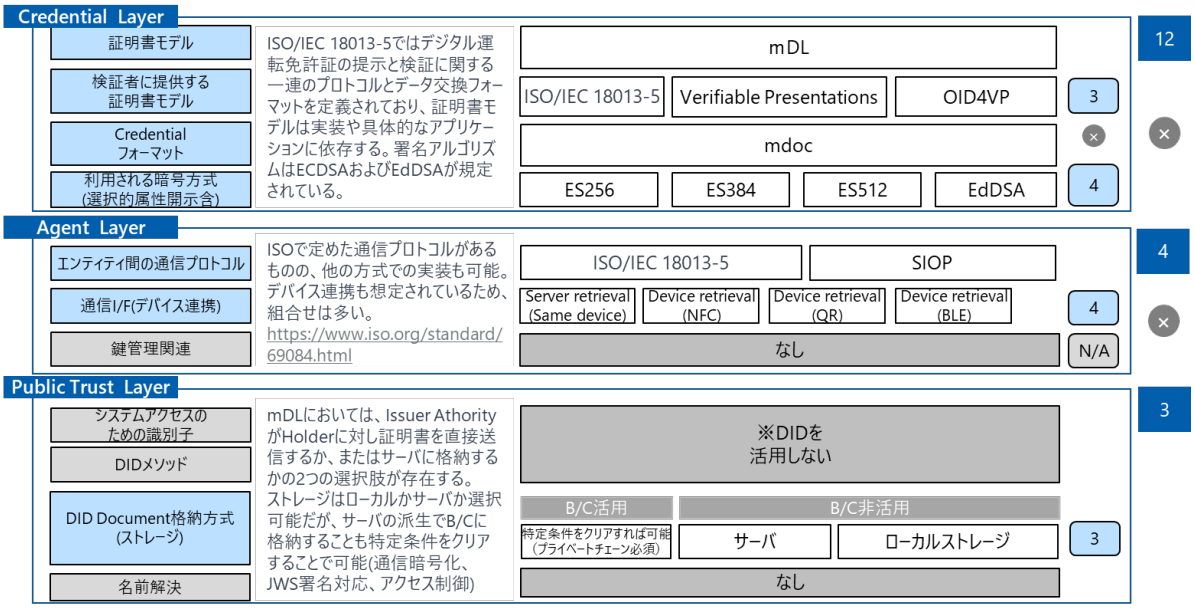


図 6-1-2　mDL を活用した実装パターン整理

[85] https://www.iso.org/standard/69084.html

サービス実装詳細

（1） EU DIW（EU ARF で策定されているもの）

EU DIW は、2026 年に EU 加盟国で運用が予定されているサービスで現時点では、サービス提供がされていない。ARF[86]は EU DIW の社会実装に向けて、エコシステムの各アクターの役割、ウォレットの機能要件および非機能要件等を定義している。

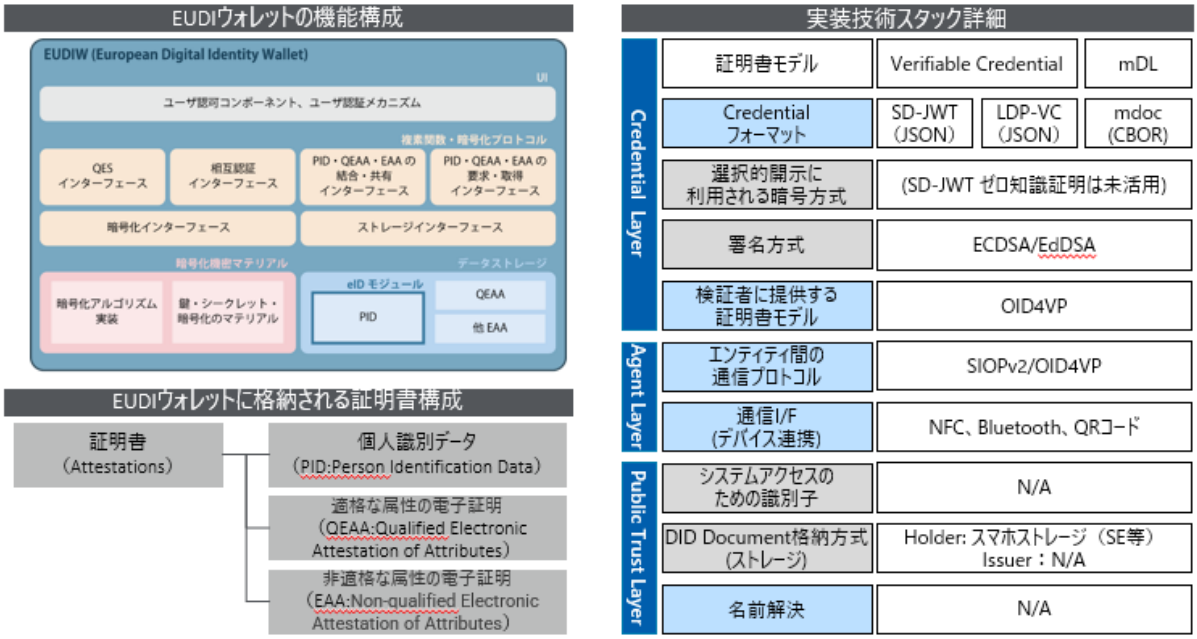


図 6-1-4　EU DIW（EU ARF で策定されているもの）概要

[86] https://digital-strategy.ec.europa.eu/en/library/european-digital-identity-wallet-architecture-and-reference-framework

（２） Lissi

Lissi は、EUDIW の ARF の技術標準([87]SD-JWT、OID4VC 系等)と、IDUnion ネットワークで構想している技術仕様 [88](Hyperledger AnonCreds / Indy、DIDComm 等)の双方に対応している。

表 6-1-1　Lissi 概要

<table>
<tr><td>開発元</td><td>Lissi GmbH</td></tr>
<tr><td>サービス概要</td><td>・ eIDAS2.0 に準拠しているウォレットを提供
・ IDunion(ドイツ)の IDunion ネットワークをサポート</td></tr>
<tr><td>利用事例</td><td>・ 顧客の会員証
・ 従業員の資格証
・ 従業員・顧客の ID アクセスマネジメント
・ ※30 ほどのユースケース事例があると記載されている</td></tr>
<tr><td>関与/参照している主要規格団体・フレームワーク</td><td>・ eIDAS2.0
・ IDUnion network
・ W3C (W3C VC, SD-JWT)
・ DIF (DIDComm)
・ OpenID Foundation (OID4VP, SIOPv2)
・ Hyperledger Foundation (AnonCreds / Indy / Aries)
・ ISO/IEC 18013 シリーズ　等</td></tr>
<tr><td>今後の動向</td><td>N/A</td></tr>
<tr><td>技術スタック詳細</td><td>
<table>
<tr><td rowspan="5">Credential Layer</td><td>証明書モデル</td><td colspan="2">ISO/IEC18013-5:2021, W3C VC</td></tr>
<tr><td>Credential フォーマット</td><td colspan="2">CBOR+MSO、SD-JWT VC、JSON-LD+LD-Proofs</td></tr>
<tr><td>選択的開示の実現方式</td><td colspan="2">SD-JWT</td></tr>
<tr><td>署名方式</td><td colspan="2">ECDSA</td></tr>
<tr><td>検証者に提供する証明書モデル</td><td colspan="2">ISO/IEC18013-5:2021, OID4VP</td></tr>
<tr><td rowspan="2">Agent Layer</td><td>エンティティ間の通信プロトコル</td><td colspan="2">ISO/IEC18013-5:2021, SIOP v2, DIDComm</td></tr>
<tr><td>通信I/F (デバイス連携)</td><td colspan="2">QRコード, e-mail</td></tr>
<tr><td rowspan="3">Public Trust Layer</td><td>システムアクセスのための識別子</td><td>N/A</td><td>didi:indy</td></tr>
<tr><td>DID Document格納方式 (ストレージ)</td><td>スマホストレージ (SE等)</td><td>Hyperledger Indy</td></tr>
<tr><td>名前解決</td><td colspan="2">N/A</td></tr>
</table>
</td></tr>
</table>

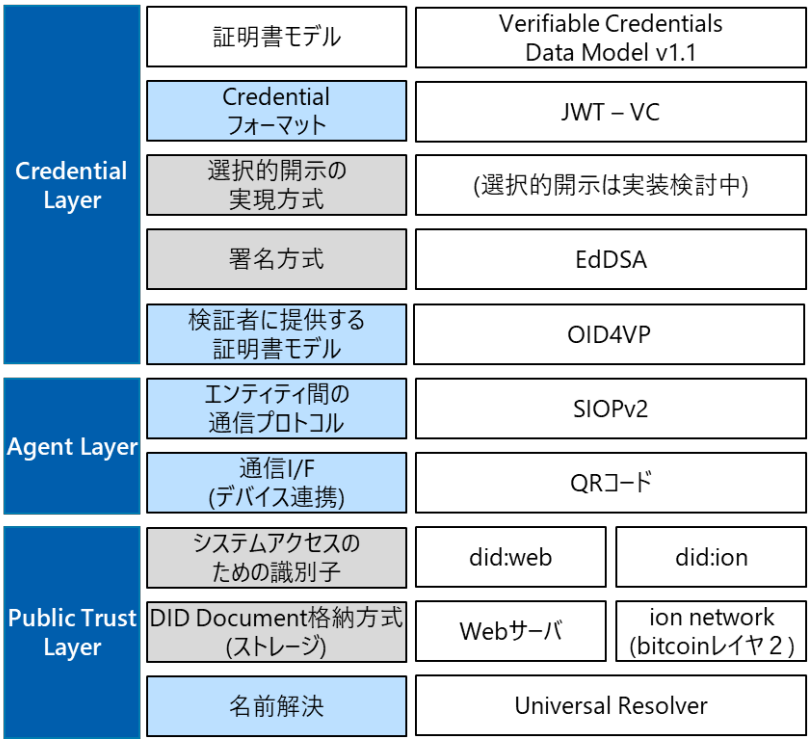
[87] https://www.lissi.id/eidas-2-0

[88] https://idunion.org/wp-content/uploads/2023/07/2023_06_05_TDI_Framework_for_Walletsecurity.pdf

（3） Microsoft Entra Verified ID

Microsoft Entra Verified ID は、Web サーバ、ブロックチェーン(ion network)のストレージに対応した証明書発行・検証サービスを提供している。

表 6-1-2　Microsoft Entra Verified ID[89]概要

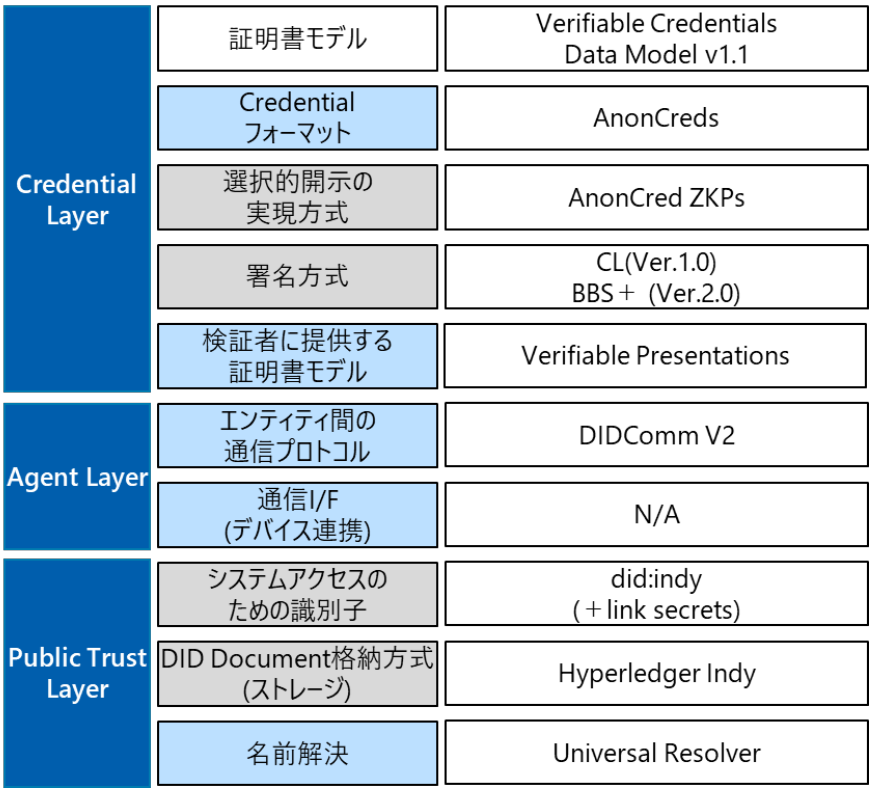
開発元	Microsoft
サービス概要	・ 従業員・顧客等の資格情報の発行・検証が可能なサービスを提供 ・ Microsoft Entra ID のサービスに含まれており、Identity Access Management 関連のサービスの拡張で、追加費用なしで利用可能
利用事例	・ NHS(National Health Service)（医療従事者の資格管理） ・ ロイヤルメルボルン工科大学（成績証明書としての活用） ・ イギリス教育省（成績証明書としての活用※実証）
関与/参照している主要規格団体・フレームワーク	・ W3C (VC データモデル、JWT-VC 等) ・ IETF (JWT-VC) ・ DIF (Presentation Exchange v1.0 / Well Known DID Configuration 等に準拠) ・ OIDF (SIOPv2 / OID4VC)
今後の動向	N/A
技術スタック詳細	

[89] https://www.microsoft.com/ja-jp/security/business/identity-access/microsoft-entra-verified-id

（4） AnonCreds

AnonCreds は Hyperledger Foundation がサポート、現在は OID4VC 系への対応がないが今後対応が期待される。

表 6-1-3　AnonCreds[90]概要

開発元	Hyperledger Foundation
サービス概要	・ VC の検証に重要なプライバシー保護(ゼロ知識証明)機能を基本機能として追加されていることが特徴
利用事例	・ 他 Hyperledger Indy 等を活用しているサービス
関与/参照している主要規格団体・フレームワーク	・ W3C ・ Hyperledger プロジェクト(Indy / Aries)
今後の動向	・ AnonCreds v2 に向けた取り組み (BBS+署名への対応) ・ OpenID Foundation 系の規格への対応
技術スタック詳細	

[90] https://www.hyperledger.org/projects/anoncreds

（５） BC Digital Trust

BC Digital Trust は Hyperledger Indy を活用して、カナダブリティッシュコロンビア州のサービス(弁護士資格確認・大学学生資格確認)を提供している。

表 6-1-4　BC Digital Trust[91,92]概要

<table>
<tr><td>開発元</td><td colspan="3">Province of British Columbia (Canada)</td></tr>
<tr><td>サービス概要</td><td colspan="3">・ VC を発行・検証するためのソフトウェア、オープンソースの BC ウォレットを提供、検証者が発行者に問い合わせることなく証明書の検証を行うので高いプライバシーレベルを確保
・ BC 州で登録された組織に関する信頼できる情報を提供するデジタル資格情報を使用する検索可能な公開ディレクトリを提供</td></tr>
<tr><td>利用事例</td><td colspan="3">・ 州弁護士の資格確認
・ 州大学の学生資格確認</td></tr>
<tr><td>関与/参照している主要規格団体・フレームワーク</td><td colspan="3">・ W3C (W3C VC, JSON-LD)
・ Hyperledger Foundation (Indy /Aries)
・ Trust Over IP Foundation 等</td></tr>
<tr><td>今後の動向</td><td colspan="3">N/A</td></tr>
<tr><td>技術スタック詳細</td><td colspan="3">
<table>
<tr><td rowspan="5">Credential Layer</td><td>証明書モデル</td><td>Verifiable Credentials Data Model v1.1</td></tr>
<tr><td>Credential フォーマット</td><td>JSON-LD + LD-Signatures</td></tr>
<tr><td>選択的開示の実現方式</td><td>LD-Signature+BBS+、ZKP</td></tr>
<tr><td>署名方式</td><td>EdDSA、BBS+</td></tr>
<tr><td>検証者に提供する証明書モデル</td><td>Verifiable Presentations</td></tr>
<tr><td rowspan="2">Agent Layer</td><td>エンティティ間の通信プロトコル</td><td>DIDcomm V2</td></tr>
<tr><td>通信I/F (デバイス連携)</td><td>QRコード</td></tr>
<tr><td rowspan="3">Public Trust Layer</td><td>システムアクセスのための識別子</td><td>did:indy</td></tr>
<tr><td>DID Document格納方式 (ストレージ)</td><td>OrgBook BC (Hyperledger Indy)</td></tr>
<tr><td>名前解決</td><td>Universal Resolver</td></tr>
</table>
</td></tr>
</table>

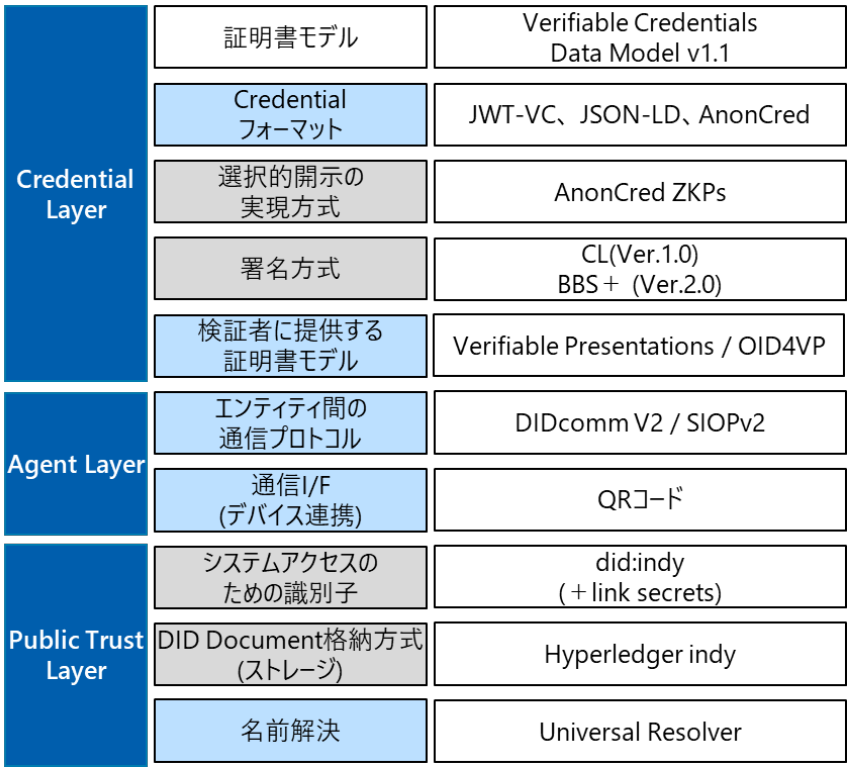
[91] https://digital.gov.bc.ca/digital-trust/digital-credentials/bc-wallet/

[92] https://www.hyperledger.org/blog/bc-digital-trust-leveraging-hyperledger-tools-for-digital-trust

（6） Nothern Block

Nothern Block は、法人向け・個人向けのウォレットサービスを提供、EU ARF を受けて、直近 OID4VC 系の対応とそれに合わせた Credential フォーマット(JWT-VC、JSON-LD)の追加がされた

表 6-1-5　Nothern Block[93,94]概要

開発元	Northern Block
サービス概要	・　企業向けのデジタル証明書管理・発行プラットフォーム・Web ベースのウォレットサービス(Orbit Enterprise)と個人向けウォレットサービス(Orbit Edge Wallet)を提供
利用事例	・　鉱業関連（鉱業データ・鉱山事業資格検証）
関与/参照している主要規格団体・フレームワーク	・　W3C (W3C VC) ・　Hyperledger Foundation　(AnonCreds / Indy / Aries) 等 ・　OIDF (OID4VC)
今後の動向	N/A
技術スタック詳細	

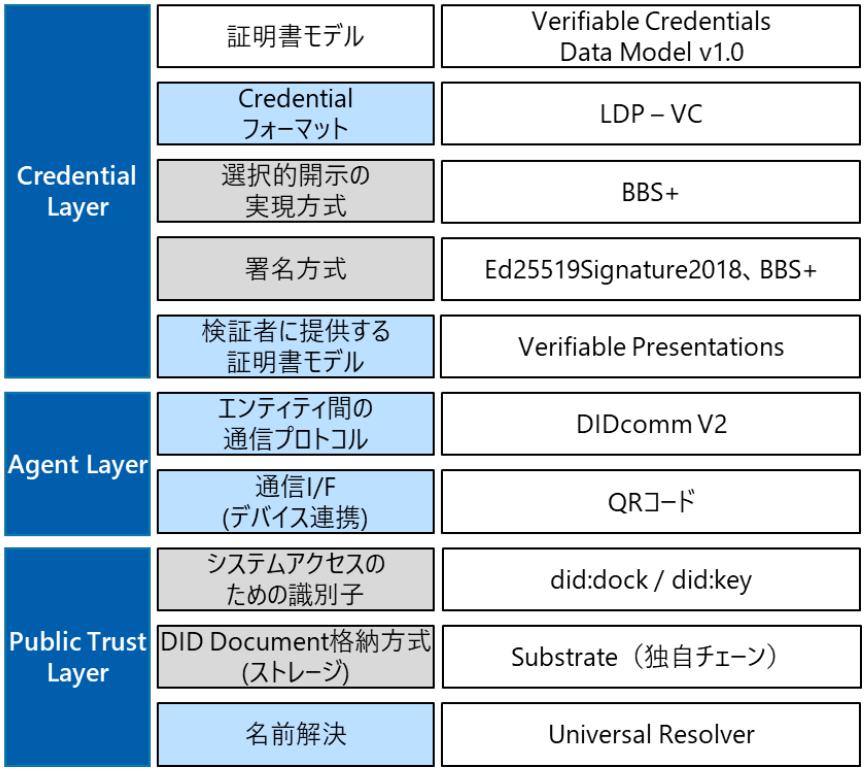
[93] https://northernblock.io/orbit-enterprise/

[94] https://northernblock.io/blog/interoperability-update-addition-of-openid4vc-to-northern-block-products/

（7） Dock Certs

Dock Certs は独自のブロックチェーンを活用して資格証明書発行・検証にかかるノーコードプラットフォームサービスを提供しており、学歴やスキル証明サービスを提供している事業者に対してサービス提供を行っている。

表 6-1-6　Dock Certs[95]概要

開発元	Dock Certs
サービス概要	・ 組織が検証可能な資格情報を効率的かつ安全に発行、検証、管理、および取り消すことを可能にする、ユーザーフレンドリーなノーコードプラットフォームを事業者向けに提供
利用事例	・ BurstIQ 従業員の身元情報・健康情報・業績等を証明書にして管理・データ分析するサービスを提供 ・ Gravity 高所作業にかかる資格証明・健康情報証明書検証サービスを提供 ・ SEVENmile デジタル卒業証明書発行サービスの提供 (オーストラリアニューサウスウェールズ州教育局と提携して 2024 年までに 1,500 の高校に導入予定)
関与/参照している主要規格団体・フレームワーク	・ W3C/DIF/IETF の仕様に対応
今後の動向	・ 2024 年第 2 四半期にウォレット型 SDK を提供予定
技術スタック詳細	

[95] https://www.dock.io/

（８） Blockcert

Blockcert は、ブロックチェーンを台帳として活用した資格証明書管理サービスを提供しており、主に学歴証明・専門職の資格証明に関連したサービスを提供している

表 6-1-7　Blockcert[96]概要

<table>
<tr><td>開発元</td><td>MIT Media Lab、Learning Machine（現在 Hyland Credential）が共同開発</td></tr>
<tr><td>サービス概要</td><td>・ ブロックチェーンベースの公式記録を発行および検証するアプリを構築するためのオープンスタンダード、主に学歴証明書で活用されている</td></tr>
<tr><td>利用事例</td><td>・ MIT 等の教育機関
学歴証明書として活用
・ マルタ
生涯にわたる学習履歴を証明書として 1 か所に保存、教育機関・企業に対して学習記録を提示するサービスを政府として提供
・ Federation of State Medical Boards(FSMB)
医師資格の資格証明として活用</td></tr>
<tr><td>関与/参照している主要規格団体・フレームワーク</td><td>・ W3C の仕様に準拠
(Verifiable Claims / Linked Data Signatures / Rebooting Web of Trust Decentralized Identifiers)</td></tr>
<tr><td>今後の動向</td><td>・ 対応ブロックチェーン基盤の追加
・ 失効モデルの柔軟性向上(発行者の失効権限の分散化等)</td></tr>
<tr><td>技術スタック詳細</td><td>
<table>
<tr><td rowspan="5">Credential Layer</td><td>証明書モデル</td><td colspan="2">Verifiable Credentials Data Model v1.1</td></tr>
<tr><td>Credential フォーマット</td><td colspan="2">LDP – VC</td></tr>
<tr><td>選択的開示の実現方式</td><td colspan="2">(選択的開示は実装検討中)</td></tr>
<tr><td>署名方式</td><td colspan="2">ECDSA</td></tr>
<tr><td>検証者に提供する証明書モデル</td><td colspan="2">Verifiable Presentations</td></tr>
<tr><td rowspan="2">Agent Layer</td><td>エンティティ間の通信プロトコル</td><td colspan="2">DIDcomm</td></tr>
<tr><td>通信I/F (デバイス連携)</td><td colspan="2">QRコード</td></tr>
<tr><td rowspan="3">Public Trust Layer</td><td>システムアクセスのための識別子</td><td colspan="2">did:web</td></tr>
<tr><td>DID Document格納方式 (ストレージ)</td><td>Ethereum</td><td>Bitcoin</td></tr>
<tr><td>名前解決</td><td colspan="2">Universal Resolver</td></tr>
</table>
</td></tr>
</table>

[96] https://www.blockcerts.org/

6.2. ライブラリ

ライブラリ全体観

DID/VC に関わるライブラリ、プラットフォームは VC モデル等の一連の処理全体をサポートするものから、証明書発行など各エンティティが提供する一部の機能をサポートするもの、選択的開示機能等特定領域をサポートするものに大別している。

ライブラリの大部分はプラットフォーム（基盤部分を担う Public Trust Layer ）への依存度が高く、特定のブロックチェーンを前提としたもの、複数のブロックチェーンに汎化したもの、ブロックチェーンを使用しないケースも考慮したものに分類できる。

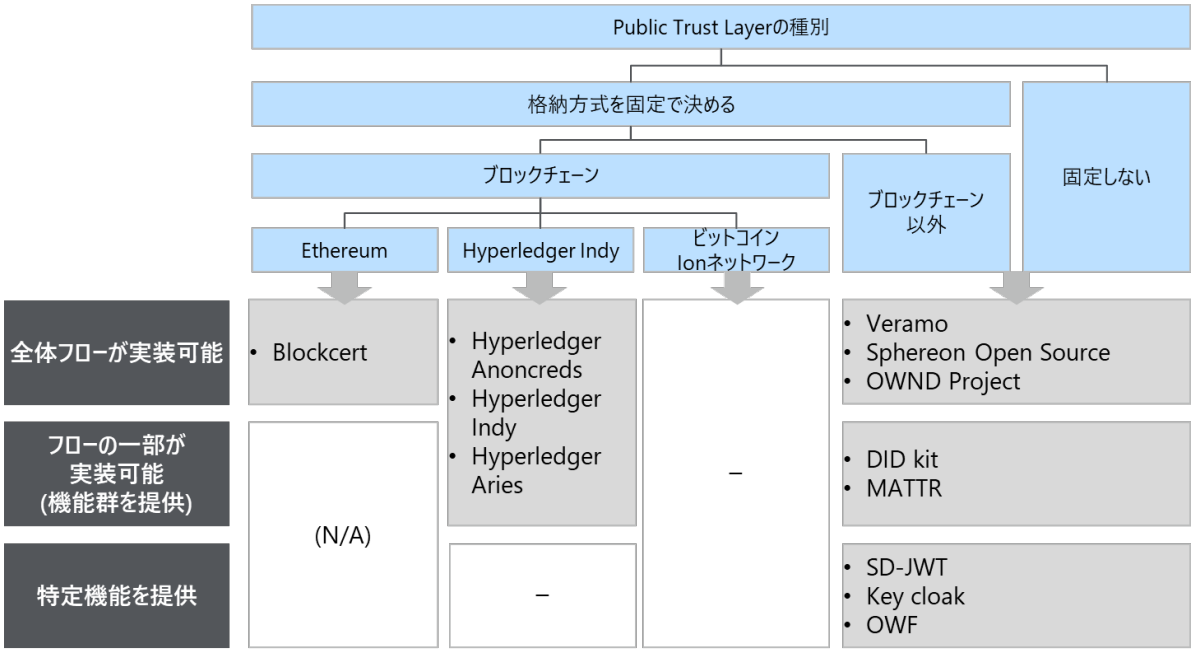


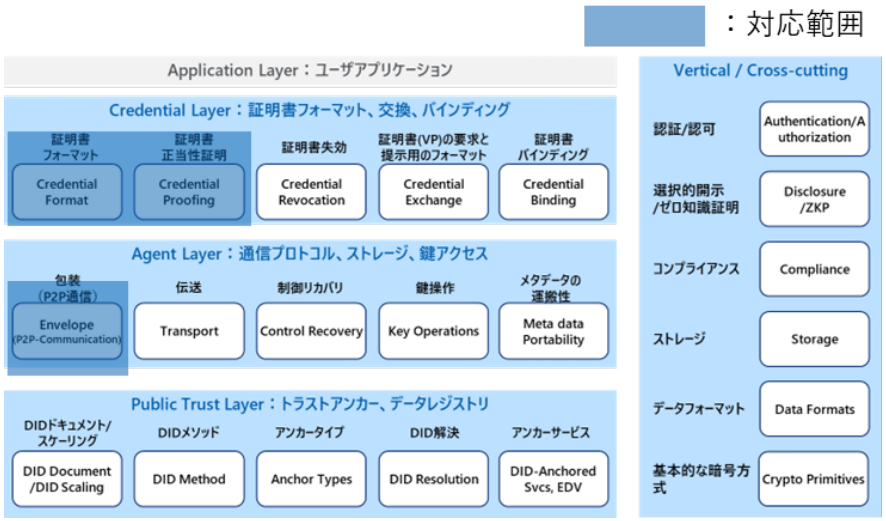
図 6-2-1　ライブラリマッピング

ライブラリ詳細

（1） Blockcert

Bloclcert は、ブロックチェーンベースの公式記録を発行および検証するアプリを構築するためのオープンスタンダードであり、オープンソースのライブラリ、ツール、モバイル アプリで構成されている。

表 6-2-1　Blockcert 概要

公開元	MIT Media Lab、Hyland-credentials
ライブラリ名	■ cert-schema[97](スキーマと仕様、スキーマと JSON-LD を検証するための python ライブラリ) ■ cert-issuer[98](Bitcoin または Ethereum 上で証明書を発行するための Python ライブラリ) ■ cert-verifier-js[99]（Node.js アプリやブラウザで使用する検証用 Javascript ライブラリ） ■ blockcerts-verifier[100](スタンドアローンの検証ツール) ■ wallet-iOS[101](iOS 用ウォレット実装) ■ wallet-android[102](Android 用ウォレット実装)
提供形態	ソースコード
ドキュメント	https://www.blockcerts.org/guide/
ライセンス	MIT License
包含技術スタック	

[97] https://github.com/blockchain-certificates/cert-schema

[98] https://github.com/blockchain-certificates/cert-issuer

[99] https://github.com/blockchain-certificates/blockcerts-verifier

[100] https://github.com/blockchain-certificates/cert-verifier-js

[101] https://github.com/blockchain-certificates/wallet-iOS

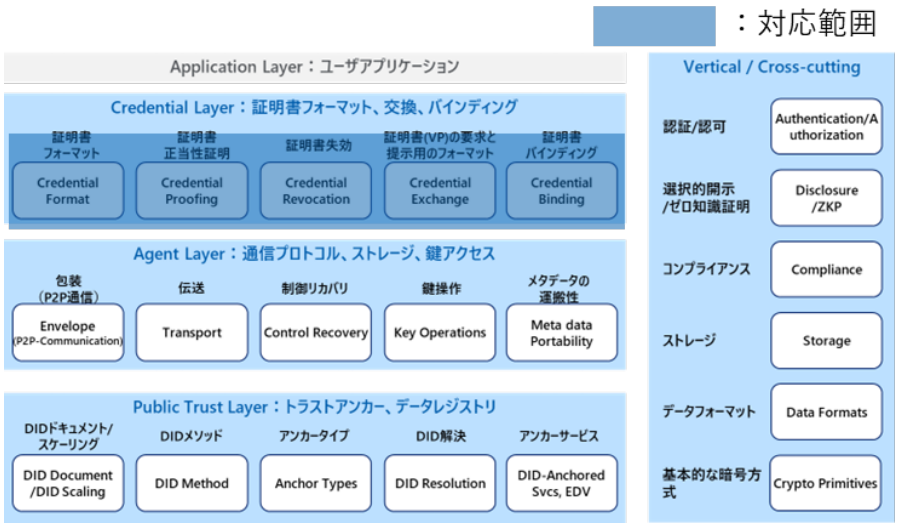
[102] https://github.com/blockchain-certificates/wallet-android

（2） Hyperledger AnonCreds

AnonCreds は 1985 年から検討が続いている仕様であるが、2016 年以降は Hyperldeger で検討が続けられている。

AnonCreds v1.0 は CL 署名に基づいており、AnonCreds v2.0 は BBS+署名に基づいた実装となっている

表 6-2-2　Hyperledger AnonCreds 概要

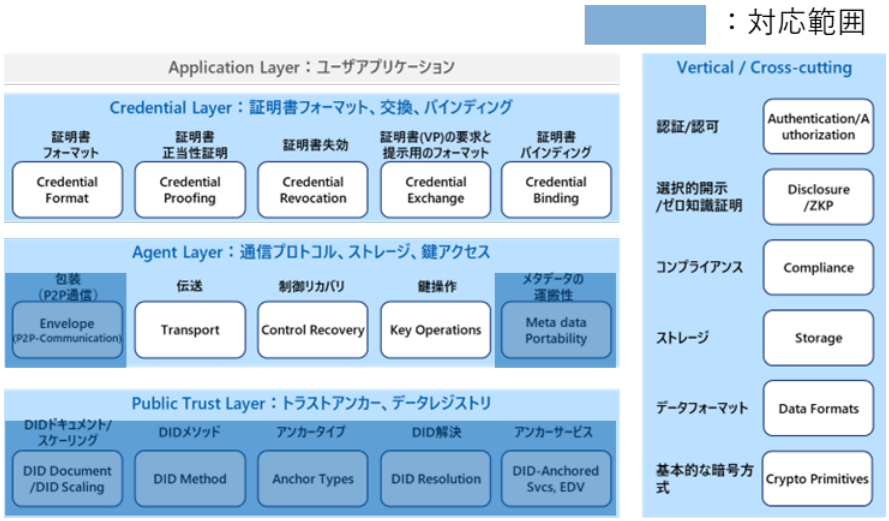
公開元	Linux Foundation (Hyperledger Project)
ライブラリ名	■ anoncreds-rs[103] 重要なプライバシー保護機能である ZKP (ゼロ知識証明) 機能をコア VC 保証に追加する VC の一種。台帳やクライアントに依存せず、Hyperledger Indy や Aries とは独立して動くことを想定して設計されているため、他の検証可能なデータレジストリ/台帳および検証可能な資格情報クライアントとともに使用できる
提供形態	ソースコード
ドキュメント	https://hyperledger.github.io/anoncreds-spec/
ライセンス	Apache License Version 2.0
包含技術スタック	

[103] https://github.com/hyperledger/anoncreds-rs

（3） Hyperledger Indy

分散型アイデンティティ専用のブロックチェーン基盤である Hyperledger Indy に接続するライブラリ群となっている。Hyperledger Indy の公式 SDK であった Indy SDK が非推奨となり、個別の実装に置き換わりつつある状況である。

表 6-2-3　Hyperledger Indy 概要

公開元	Linux Foundation (Hyperledger Project)
ライブラリ名	■ Indy-VDR (Verifiable Data Registry)[104] Indy Node との接続モジュール ■ Aries Askar[105] Wallet 機能の実装 ■ indy-cli-rs[106] Indy のコマンドラインインターフェイス
利用ケース	AnonCreds の証明書フォーマットを採用した証明書関連サービス
提供形態	ソースコード
ドキュメント	https://hyperledger.github.io/indy-did-method/
ライセンス	Apache License Version 2.0 MIT License
包含技術スタック	

[104] https://github.com/hyperledger/indy-vdr

[105] https://github.com/hyperledger/aries-askar

[106] https://github.com/hyperledger/indy-cli-rs

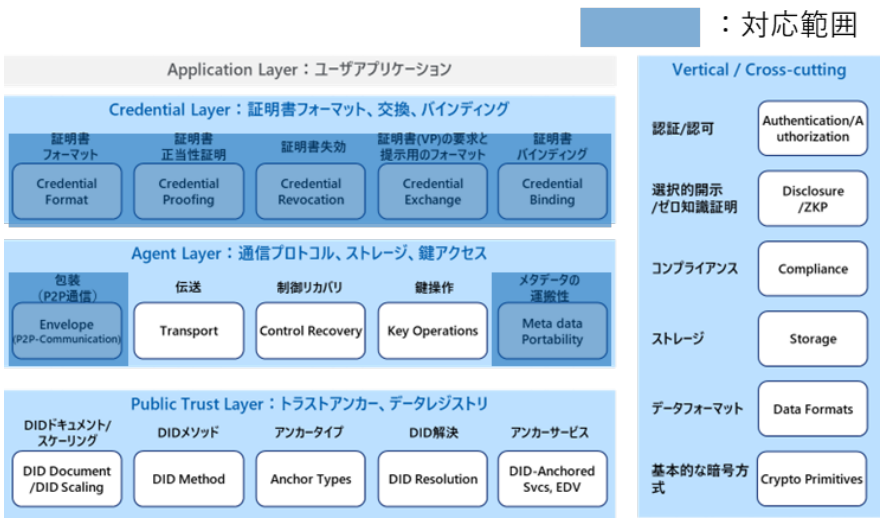
（4） Hyperledger Aries

エンティティ間で相互に信頼できる P2P 接続を行うコンポーネント群（ユーザエージェント、DID 通信、キー管理、プロトコル）で構成されている。

複数のプロジェクトが立ち上がっており、現在も活発に活動中。ただし、更新を停止し、アーカイブに格納されたプロジェクトも多数存在(モバイル用途に開発された.NET 版や Ruby 版、テストフレームワークなど)。

共有ライブラリとして、ストレージプラグインを含む鍵管理、データリポジトリと接続するインターフェイス、各種ユーティリティ等を提供する C ライブラリを公開予定である（現在はフレームワークごとに個別実装）。

表 6-2-4　Hyperledger Aries 概要

公開元	Linux Foundation　(Hyperledger Project)
ライブラリ名	■ Hyperledger Aries Cloud Agent Python(ACA-Py)[107] ■ Static Agent Library(SAL)[108] ■ Aries Framework Go(AFG)[109]
提供形態	ソースコード
ドキュメント	https://aries-cloud-agent-python.readthedocs.io/en/latest/
ライセンス	Apache License Version 2.0
包含技術スタック	

[107] https://github.com/hyperledger/aries-cloudagent-python

[108] https://github.com/hyperledger/aries-staticagent-python

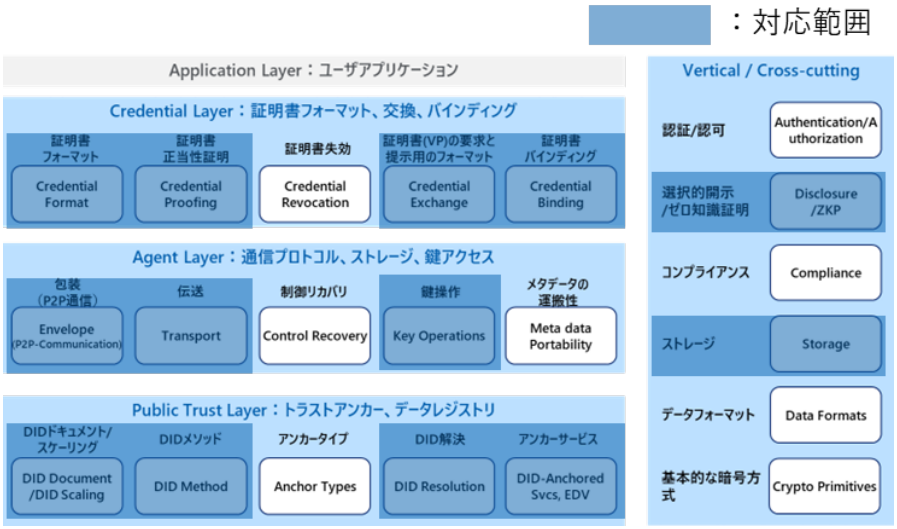
[109] https://aries-cloud-agent-python.readthedocs.io/en/latest/

（5） Veramo

Ethereum を利用した分散型デジタル ID サービス「uPort」から分割して設立されたプロジェクト。W3C と DIF と協力して開発されており、DIF の Github にソースコードが存在する。更新頻度は高い。

選択的開示は開発中で SD-JWT、JSON-LD BBS+が検討されている（SD-JWT はソースコードが存在）。

表 6-2-5　Veramo 概要

公開元	Veramo
ライブラリ名	■ Veramo[110] Veramo は、検証可能なデータと SSI のための JavaScript フレームワーク。柔軟なモジュール式に設計されており、プラグインによって拡張可能。以下の機能に対応されている - 署名と暗号化のためのキーの作成と管理 - DID の作成と管理 - VC と VP の発行 - 選択的開示による資格情報の提示 - DIDComm によるエージェント間の通信データの受信、フィルタリング、保存、提供
提供形態	ソースコード
ドキュメント	https://veramo.io/docs/basics/introduction
ライセンス	Apache License Version 2.0
包含技術スタック	

（6） Spheron Open Source

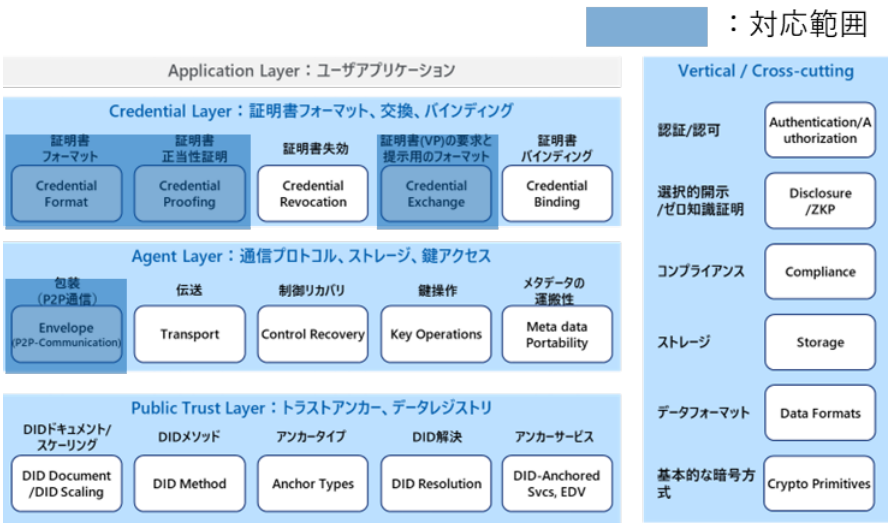
行政、医療、臨床試験、モビリティ、教育、その他の業界向けのデータ交換ソリューションであり、ウォレット機能、SIOPv2 に準拠したデータ交換、eIDAS に準拠した署名クライアント、SSI SDK 等を提供している。

---

[110] https://github.com/decentralized-identity/veramo

選択的開示機能はまだ実装されていないものの、BSS+署名ベースの検討はされている模様。

表 6-2-6　Spheron Open Source 概要

公開元	Sphereon Open Source
ライブラリ名	■ SSI-SDK[111] ■ OID4VCI[112] ■ SIOP-OID4VP[113] ■ ssi-mobile-wallet[114] ■ eidas-signature-client[115] ■ Presentation Exchange v1 and v2 TypeScript Library[116]
提供形態	ソースコード
ドキュメント	https://github.com/Sphereon-Opensource/SSI-SDK
ライセンス	Apache License Version 2.0
包含技術スタック	

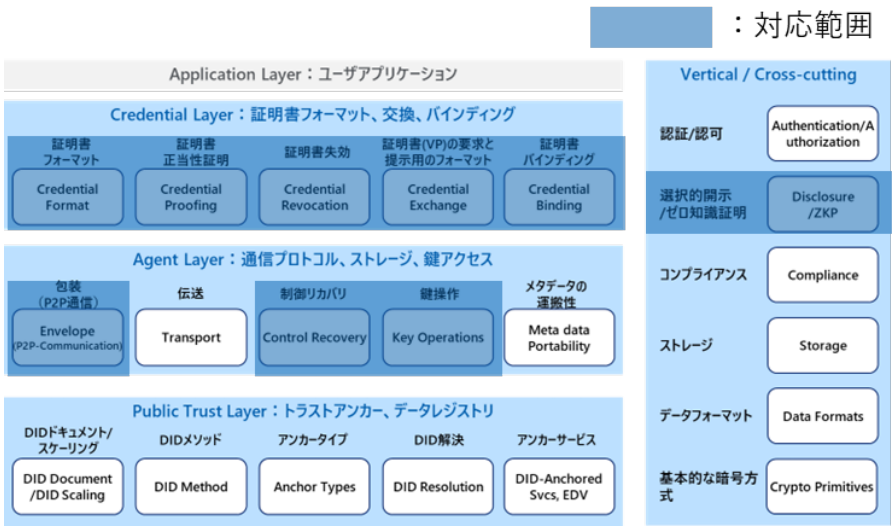
（7） OWND Project

国際標準技術に準拠したホワイトラベルのデジタルアイデンティティウォレットであり、OID4VCI / OID4VP / SIOP v2 に対応しており、iOS 用と Android 用が提供されている。選択的開示機能として SD-JWT に対応、JSON-LD ZKP with BBS+も対応予定である。OID4VCI に準拠したデジタルアイデンティティ発行サービス（OWND Project VCI）とマイナンバーカード情報、従業員 ID、イベント参加証明書の 3 つの Web アプリケー

[111] https://github.com/Sphereon-Opensource/SSI-SDK
[112] https://github.com/Sphereon-Opensource/OID4VCI
[113] https://github.com/Sphereon-Opensource/SIOP-OID4VP
[114] https://github.com/Sphereon-Opensource/ssi-mobile-wallet
[115] https://github.com/Sphereon-Opensource/eidas-signature-client
[116] https://github.com/Sphereon-Opensource/PEX

ションが含まれる。OWND wallet を用いてアイデンティティを管理できる E2E 暗号化に対応したメッセージングアプリケーションを提供している（サーバ機能、クライアント機能、React SDK）。

表 6-2-7　OWND Project 概要

公開元	OWND Project
ライブラリ名	■ OWND Wallet：ホワイトラベルのデジタルアイデンティティウォレット - OWND-Wallet-iOS[117]、OWND-Wallet-Andorid[118] ■ OWND-Project-VCI：デジタルアイデンティティ発行サービスとサンプルコード - OWND-Project-VCI[119] ■ OWND Messenger：アイデンティティ管理可能なメッセージングアプリ - OWND-Messenger-Server[120] - OWND-Messenger-Client[121] - OWND-Messenger-React-SDK[122]
提供形態	ソースコード
ドキュメント	https://github.com/OWND-Project/whitepaper
ライセンス	MIT License
包含技術スタック	

（8） DID Kit

W3C 検証可能クレデンシャルの署名・検証、多言語ライブラリサポート、HTTP/HTTPS サーバー提供、Linked Data Proofs と JOSE トークン間の変換、多様な W3C 分散型識別子の処理、およびオブジェクト機能モデルに基づく認証トークンの発行と使用が可能なツール。VC-API に対応したサーバ用コンテナも提供する。

---

[117] https://github.com/OWND-Project/OWND-Wallet-iOS

[118] https://github.com/OWND-Project/OWND-Wallet-Android

[119] https://github.com/OWND-Project/OWND-Project-VCI

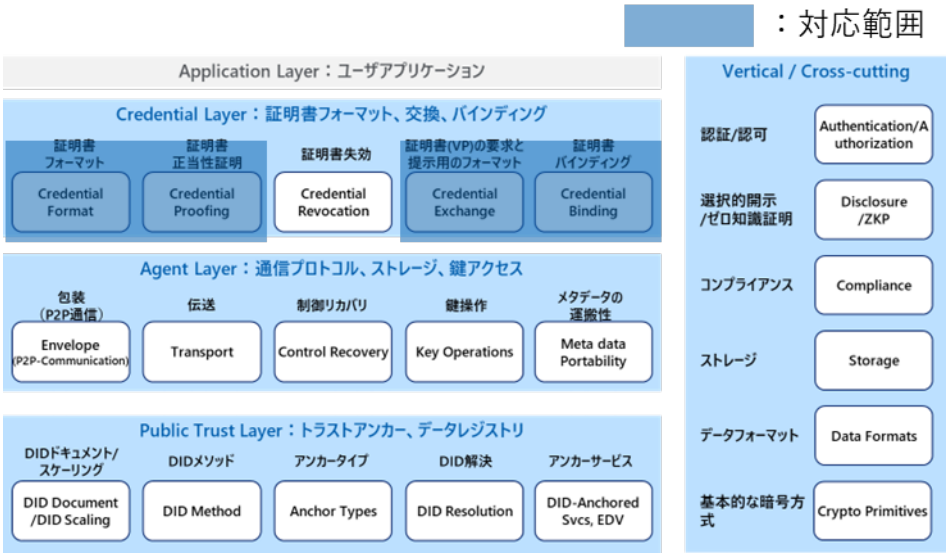
[120] https://github.com/OWND-Project/OWND-Messenger-Server

[121] https://github.com/OWND-Project/OWND-Messenger-Client

[122] https://github.com/OWND-Project/OWND-Messenger-React-SDK

更新頻度は比較的高く、コマンドライン、HTTP サーバおよびモバイル環境として C, Java, Android および Flutter での利用が想定されている。

表 6-2-1　DID Kit 概要

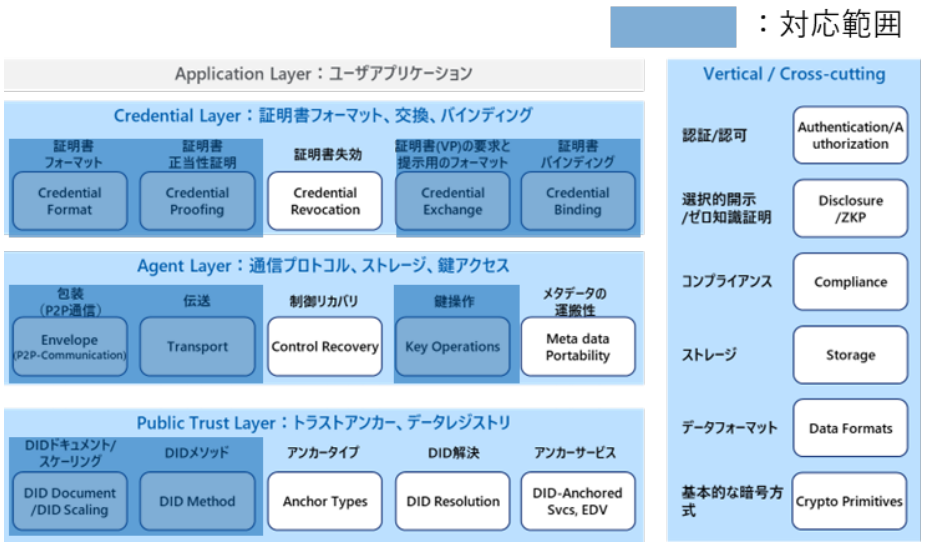
公開元	Spruce Systems Inc.
ライブラリ名	■ DIDKit[123]
提供形態	ソースコード
ドキュメント	https://www.spruceid.dev/didkit/didkit
ライセンス	Apache License Version 2.0
包含技術スタック	

[123] https://github.com/spruceid/didkit

（9） MATTR

検証可能なデータと検証可能な資格情報と連携し、情報を安全に共有、保持、検証するための製品群を提供している。製品はサービス（REST API）、SDK で提供されており、ソースコードでの提供ではないが、JSON-LD+BBS+署名などのサンプルコードは公開されている。

表 6-2-2　MATTR 概要

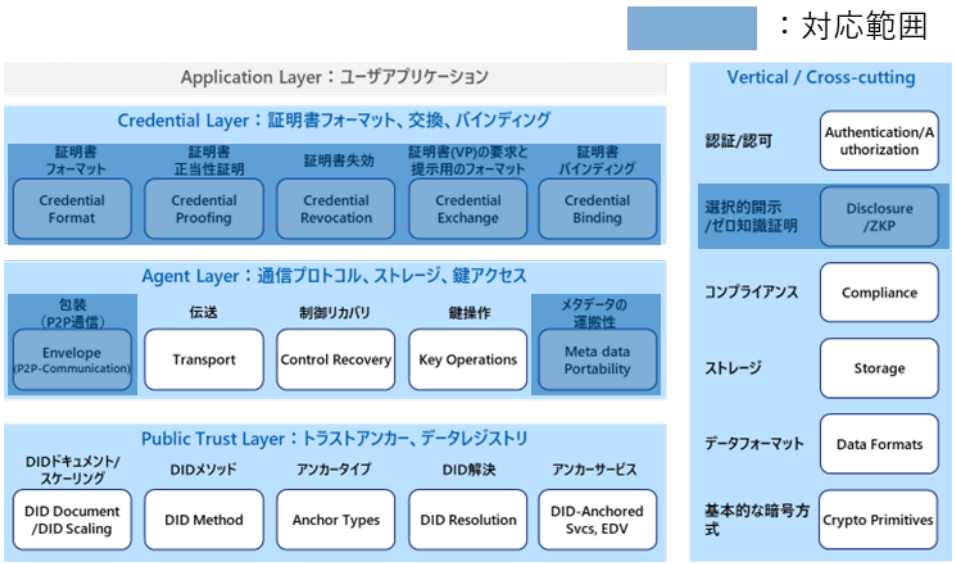
公開元	MATTR Platform
ライブラリ名	■ MATTR VII[124]：検証可能な認証情報を生成・管理するための API と各種機能を備えたクラウドプラットフォーム ■ MATTR Pi[125]：ウォレットと検証機能を開発するための SDK - キーと分散型識別子 (DID) の作成と管理 - OpenID 資格情報プロバイダー - 検証機能 - BBS+署名による選択的開示 - DID ベースのメッセージング - カスタマイズ可能なユーザーエクスペリエンス - iOS と Android で利用可能な共通コード ■ MATTR GO[126]：独自のブランディング、色、タイポグラフィなどを使用してユーザーエクスペリエンスをカスタマイズできるプラットフォーム
提供形態	SDK＋SaaS サービス
ドキュメント	https://learn.mattr.global/docs/
ライセンス	MATTR
包含技術スタック	

[124] https://learn.mattr.global/docs/vii-platform/overview
[125] https://learn.mattr.global/docs/pi-platform/overview
[126] https://learn.mattr.global/docs/go-platform/overview

（１０） OpenWallet Foundation

デジタル ID ウォレットを構築するためのオープン フレームワーク。OID4VC および SD-JWT をサポートする。また、OID4VC/ OID4VP / SIOP v2 に対応している。Hyperledger で取り組まれていた Aries Framework .NET および Aries Framework Java Script が OpenWallet Foundation に移管されて移行作業中である

表 6-2-3　Open Wallet Foundation の概要

公開元	OpenWallet Foundation
ライブラリ名	■ Wallet Framework for .NET[127] ■ Aries Framework - JavaScript(AF-JS)[128]
提供形態	ソースコード
ドキュメント	https://aries-cloud-agent-python.readthedocs.io/en/latest/
ライセンス	Apache License Version 2.0
包含技術スタック	

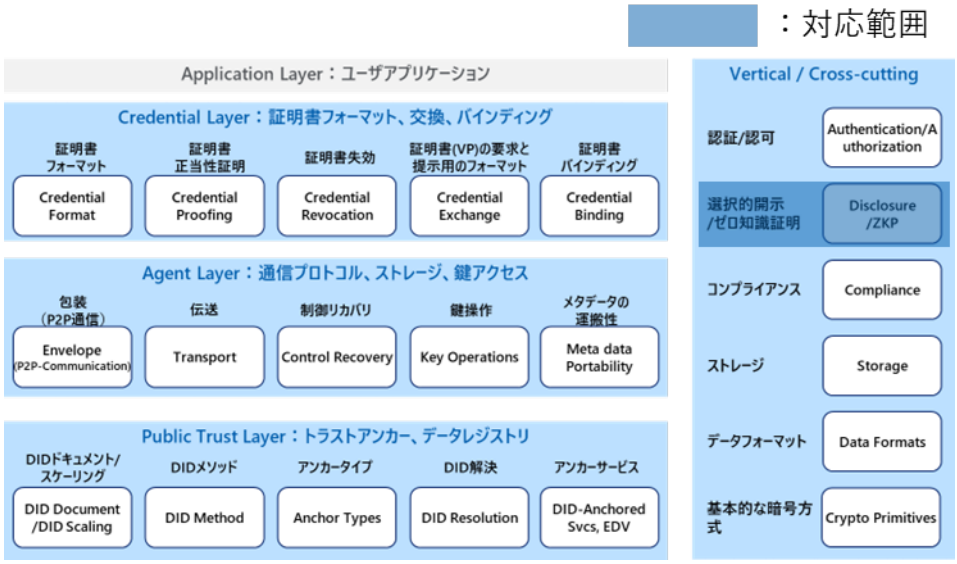
[127] https://github.com/openwallet-foundation-labs/wallet-framework-dotnet

[128] https://github.com/openwallet-foundation/agent-framework-javascript

（１１） SD-JWT（OpenWallet Foundation）

IETF SD-JWT 仕様の例を生成するために使用され、SD-JWT を実装するための他のプロジェクトでも使用可能、複数の言語に対応している。

表 6-2-4　SD-JWT 概要

公開元	OpenWallet Foundation
ライブラリ名	■ SD-JWT Rust Reference Implementation[129] ■ SD-JWT Implementation in JavaScript (TypeScript)[130] ■ SD-JWT Python Reference Implementation[131] ■ SD-JWT-DotNet[132] ■ SD-JWT Implementation in Kotlin[133]
提供形態	ソースコード
ドキュメント	https://datatracker.ietf.org/doc/draft-ietf-oauth-selective-disclosure-jwt/
ライセンス	Apache License Version 2.0
包含技術スタック	

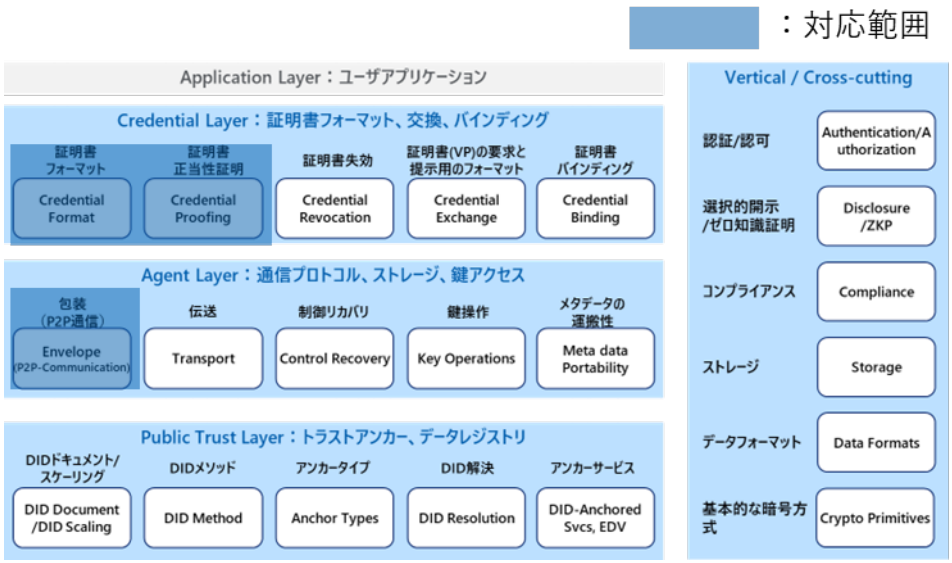
[129] https://github.com/openwallet-foundation-labs/sd-jwt-rust
[130] https://github.com/openwallet-foundation-labs/sd-jwt-js
[131] https://github.com/openwallet-foundation-labs/sd-jwt-python
[132] https://github.com/openwallet-foundation-labs/sd-jwt-dotnet
[133] https://github.com/openwallet-foundation-labs/sd-jwt-kotlin

（１２） Keycloak

Keycloak は OIDC に準拠したアイデンティティ管理ミドルウェアであり、SIOP-2 / OIDC4VP クライアントをサポートし、OIDC4VCI プロトコルを通じて 準拠ウォレットに Verifiable Credentials を発行するための Keycloak のプラグインである。

FIWARE は、欧州連合（EU）の ICT プロジェクトとして、2011 年からの 5 年間に実施された次世代インターネット官民連携プログラム（FI-PPP）において開発されている。V8.4.0 から認証ミドルウェアの Keycloak に VC 機能を持たせたモジュールが提供されている。

表 6-2-5　Keycloak 概要

公開元	Keycloak (Cloud Native Computing Foundation)
ライブラリ名	■ keycloak-vc-issuer[134]
提供形態	ソースコード
ドキュメント	https://www.keycloak.org/guides
ライセンス	Apache License Version 2.0
包含技術スタック	

[134] https://github.com/FIWARE/keycloak-vc-issuer