Author: Fabrice Rochette, Verana Foundation

Audience: Newcomers with strong technical backgrounds (developers, architects, security engineers, data scientists) who want a practical, conceptual, and implementation-oriented introduction to the technologies Verana builds on.

What you will learn:

- What a Decentralized Identifier (DID) is, how DID Documents work, and why DIDs are stronger identifiers than URLs for security-sensitive systems.
- How W3C Verifiable Credentials (VCs) work, the issuer–holder–verifier trust triangle, and why VCs enable privacy-preserving data sharing.
- Why trust registries and decentralized trust networks are necessary, how they interoperate, and how they enable sustainable, privacy-preserving business models.

0) Problem Statement: Trust at Internet Scale

We want **portable, verifiable trust** between parties that don't share a database or a single identity provider. The Web's default identifier (the URL) ties trust to DNS, certificate authorities, and domain control—great for content addressing, fragile for **entity** addressing (people, orgs, services, agents). We need identifiers that are **cryptographically bound to controllers**, interoperable across ecosystems, and work offline. We need a way to attach **verifiable claims** to those identifiers, prove them selectively, and check who's authorized to issue/verify them. That's what **DIDs + VCs + Trust Registries** provide.

1) Decentralized Identifiers (DIDs)

A **DID** is a globally unique identifier (URI) whose **ownership is proven cryptographically**, not by a central registry. Example:

```
did:web:example.com
```

1.1 DID Document (the metadata for a DID)

Resolving a DID returns a **DID Document**—a small JSON object describing public keys, verification relationships, and endpoints controlled by the DID's controller.

```
{
  "id": "did:example:123",
  "verificationMethod": [
    {
      "id": "#keys-1",
      "type": "JsonWebKey2020",
      "controller": "did:example:123",
      "publicKeyJwk": { "kty": "EC", "crv": "P-256", "x": "...", "y":
"..." }
    }
  ],
  "authentication": ["#keys-1"],
  "assertionMethod": ["#keys-1"],
  "service": [
    { "id": "#agent", "type": "DIDCommMessaging", "serviceEndpoint":
"https://agent.example.com" }
  ]
}
```

Why DIDs > URLs for identifying entities:

- **Self-certifying control:** Proof of control comes from possession of the private key corresponding to keys in the DID Document, not just control of a domain or account.
- **Key agility:** Rotate keys and add multiple keys/relations without changing the DID.
- **Transport agnostic:** Works online/offline and across transports (HTTP, DIDComm, Bluetooth, QR, NFC).
- **Method choice:** Different **DID methods** (below) offer different root-of-trust models (web, logs, ledgers, etc.).

> **Note:** Anti-correlation best practice is to use **pairwise DIDs** (a distinct DID per relationship) where appropriate.

2) Examples of DID Methods You'll Meet in Verana Ecosystems

A DID's first path segment is its **method** (e.g., `did:web:…`). Methods define how DIDs are created, resolved, and updated.

2.1 `did:web` — leverage DNS + HTTPS

- **How it works:** Host a DID Document under a domain you control (e.g., `https://example.com/.well-known/did.json`).
- **Pros:** Simple, deployable today, integrates well with existing web ops; good for organizations and services.
- **Cons:** Trust ultimately rooted in DNS and Web PKI; no built-in verifiable key history.
- **When to use:** Public-facing services, developer portals, early integrations, discovery via the web.

2.2 `did:webvh` — did:web + verifiable history

- **Idea:** Keep the operational convenience of `did:web` but add a **verifiable key/event history** (e.g., using KERI-style key event logs) so resolvers can audit control changes over time.
- **Pros:** Mitigates `did:web`'s weakest point: unverifiable history and key compromises; maintains compatibility with web tooling.
- **Cons:** More moving parts than plain `did:web`.
- **When to use:** Enterprises and ecosystems needing web-based identifiers **with** historical accountability and stronger assurance.

2.3 `did:webs` — web-anchored, cryptographically secured by event logs

- **Idea:** A web-discoverable DID whose **trust is rooted in cryptographic key events** (not DNS/CA). Typically built on **KERI** key event receipts.
- **Pros:** Stronger independence from DNS/CA operators; verifiable key rotation history.
- **Cons:** Newer method; requires event-log infra.
- **When to use:** High-assurance use cases that still want web discovery.

2.4 `did:ebsi` — DIDs in the European EBSI ecosystem

- **Idea:** EBSI provides DID methods and registries for EU trust ecosystems (e.g., legal entities). DID Documents are registered and governed per EU rules.
- **Pros:** Alignment with EU trust lists, accreditation, and conformance tooling; clear governance.
- **Cons:** Tied to EU frameworks and onboarding processes.
- **When to use:** Interop with EU public-sector and EBSI-compliant ecosystems.

> **Takeaway:** `did:web` is the easiest on-ramp; `did:webvh`/`did:webs` add **provable key history** and stronger assurances; `did:ebsi` aligns you with EU trust frameworks.

3) W3C Verifiable Credentials (VCs)

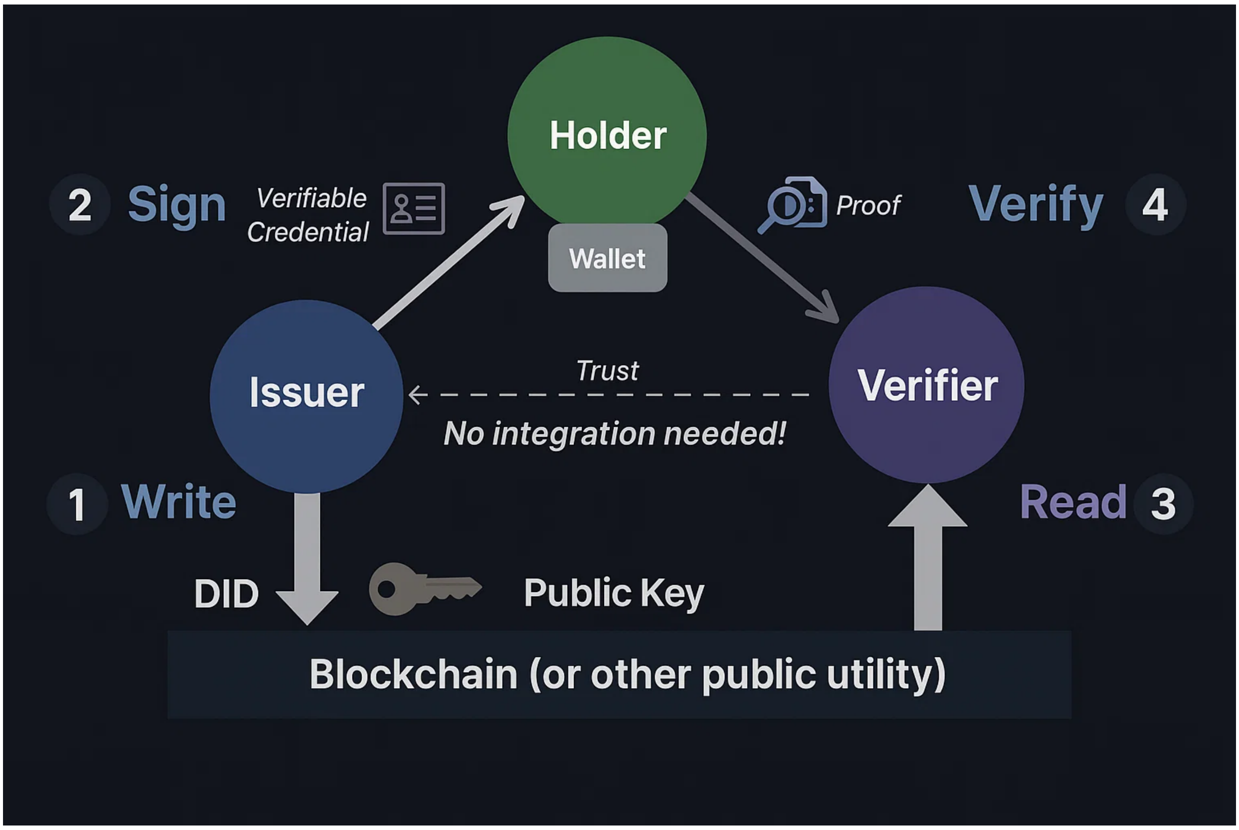
A **Verifiable Credential** is a tamper-evident package of claims that an **issuer** makes about a **subject** (identified by a DID or other identifier). A **Verifiable Presentation (VP)** is a holder-curated bundle of one or more credentials/derived proofs presented to a verifier.

3.1 Anatomy (minimal JSON example)

```
{
  "@context": ["https://www.w3.org/ns/credentials/v2"],
  "type": ["VerifiableCredential", "EmployeeCredential"],
  "issuer": "did:web:acme.example",
  "credentialSubject": {
    "id": "did:web:alice.example",
    "employeeId": "E-12345",
    "role": "Engineer"
  },
  "validFrom": "2025-01-01T00:00:00Z",
  "credentialStatus": {
    "type": "StatusList2021Entry",
    "statusPurpose": "revocation",
    "statusListIndex": "4242",
    "statusListCredential": "https://acme.example/status/employee-
2025.json"
  },
  "proof": { /* Data Integrity or JOSE/COSE proof */ }
}
```

3.2 The Trust Triangle

- **Issuer** (e.g., a company, university) signs a VC.
- **Holder** (person, org, service, or AI agent) stores it in a wallet/agent and creates a VP when needed.
- **Verifier** checks the cryptographic proof, the credential's schema, its status, and whether the issuer is **authorized** for that schema in the relevant **trust registry**.



3.3 Why VCs enable privacy-preserving data sharing

- **Selective disclosure:** Share **only** the claims necessary (e.g., "age over 18") using cryptosuites like **BBS+** or formats like **SD-JWT VC**.
- **Unlinkability:** Derived proofs prevent verifiers from correlating different presentations.
- **Pairwise identifiers:** Use different DIDs per verifier to avoid cross-domain correlation.
- **Revocation at scale: Status List** credentials enable privacy-preserving revocation checks without calling home on each presentation.

3.4 Crypto options you'll see

- **Data Integrity** proofs (JSON-LD; pluggable cryptosuites like **BBS+** for selective disclosure and predicates).
- **JOSE/COSE** proofs (JWT/JWS, SD-JWT, COSE) for familiar JOSE/CBOR stacks.

4) Trust Registries and Decentralized Trust Networks

4.1 What is a Trust Registry?

A **Trust Registry** (aka "trust list") publishes **authoritative mappings** like:

- **Schemas**: machine-readable definitions of credential types.
- **Issuers**: who is authorized to issue which schemas (under what accreditation).
- **Verifiers**: who is authorized to request/verify which schemas (and for what purpose).
- **Policies & governance**: links to the **Ecosystem Governance Framework (EGF)** that defines rules, evidence, liability, and audit.

4.2 Why registries are necessary

- **Automatable trust decisions**: Verifiers need to know, *programmatically*, whether an issuer is legitimate for a credential type.
- **Interoperability**: Common schemas + canonical issuer lists enable multi-vendor, multi-jurisdiction networks.
- **Privacy-preserving business models**: Authorization checks occur against **ecosystem policy**—not by centralizing user data. Users disclose **minimal** claims; verifiers check issuer authorization against the registry.

4.3 Examples and patterns

- **EU/EBSI** maintains **Trusted Issuers** and **Trusted Schemas** registries; credentials reference status lists and schema URIs.
- **ToIP** defines a **Trust Registry Query Protocol (TRQP)**—a simple, DNS-like query API to ask: "Does entity X hold authorization Y under governance Z?"

4.4 Decentralized Trust Networks

A **decentralized trust network** is a federation of agents, wallets, services, and registries operating under one or more EGFs. Key properties:

- **Composability**: Multiple registries (schemas/issuers/verifiers) can interoperate; cross-network bridges answer queries across EGFs.
- **Layered trust**: Global discovery → ecosystem policy → cryptographic verification → privacy-preserving presentation.
- **No data honeypots**: Registries list **who** is trusted for **what**; they do **not** store end-user PII or presentations.

5) End-to-End Flow (Step-by-Step)

1. **Identify entities with DIDs**
   - Choose a method per assurance needs: `did:web` (simple), `did:webvh`/`did:webs` (verifiable key history), `did:ebsi` (EU alignment). Publish DID Docs.
2. **Define schemas**
   - Model credential types using JSON Schema or JSON-LD contexts. Register schema URIs in the network's **Trusted Schemas Registry**.
3. **Governance**
   - The ecosystem's EGF defines roles, accreditation, audit, and dispute processes. Issuers/verifiers are onboarded against the EGF.
4. **Registry onboarding**
   - Authorize issuers/verifiers for specific schemas and publish entries in **Trust Registries**.
5. **Issue credentials**
   - Issuer signs VCs to subjects' DIDs. Include `credentialStatus` (e.g., Status List 2021) and schema references.
6. **Hold & manage**
   - Wallets/agents store VCs, manage key material, rotate DIDs, and support pairwise DIDs.
7. **Present**
   - Holder derives a **Verifiable Presentation** with selective disclosure/predicates; binds presentation to the verifier (nonce/audience) to prevent replay.
8. **Verify**
   - Verifier checks: (a) proof and binding, (b) schema conformance, (c) credential status, and (d) **authorization** of the issuer (and, if applicable, the verifier) via the **Trust Registry**.
9. **Log & comply**
   - Record only what policy allows (e.g., attest outcome, not raw PII). Maintain auditability without building correlation graphs.

6) Design Choices & Trade-offs

- **did:web ↔ did:webvh/did:webs**: operational simplicity vs. verifiable key history and CA/DNS independence.

- **Data Integrity vs. JOSE/COSE**: JSON-LD expressiveness and ZK-friendly suites (e.g., BBS+) vs. JOSE/COSE familiarity and SD-JWT VC adoption.
- **Single vs. multiple registries**: Simplicity vs. separation of concerns (schemas, issuers, verifiers) and cross-ecosystem scaling.
- **Privacy posture**: Prefer pairwise DIDs, status lists, and selective disclosure by default.

7) Why this stack is the future of the Internet

- **Portable trust**: Credentials work across org and jurisdiction boundaries.
- **Composability**: Mix methods, crypto suites, and governance without rewiring the Web.
- **Security**: Self-certifying identifiers, verifiable history, and hardware-anchored keys.
- **Privacy by design**: Minimal disclosure, unlinkability, pairwise identifiers, and revocation at scale.
- **Market fit**: Clear role separation (issuers/holders/verifiers) enables **new business models** around authorization, not data brokerage.

8) Quick Reference

- **DID:** Cryptographically controlled identifier with a resolvable metadata document.
- **DID Document:** Lists keys, verification relationships, and service endpoints for the DID controller.
- **VC:** Signed set of claims about a subject; **VP** is a holder-curated presentation.
- **Trust Registry:** Authoritative list of schemas and who's authorized to issue/verify them under an EGF.
- **Decentralized Trust Network:** An interoperable federation of wallets, agents, verifiers, issuers, and registries under one or more EGFs.

9) Further Reading (high-level)

- W3C DID Core: https://www.w3.org/TR/did-core/
- did:web method: https://w3c-ccg.github.io/did-method-web/
- did:webvh background: https://identity.foundation/didwebvh/next/
- did:webs spec: https://keri.one/did-webs
- EBSI DID methods: https://ec.europa.eu/digital-strategy/our-policies/european-blockchain-services-infrastructure
- W3C Verifiable Credentials Data Model 2.0: https://www.w3.org/TR/vc-data-model-2.0/
- W3C Data Integrity: https://www.w3.org/TR/vc-data-integrity/
- VC JOSE/COSE: https://www.w3.org/TR/vc-jose-cose/
- BBS+ cryptosuite: https://w3c-ccg.github.io/ldp-bbs2020/
- SD-JWT VC draft: https://datatracker.ietf.org/doc/draft-ietf-oauth-sd-jwt-vc/
- Status List 2021: https://www.w3.org/TR/vc-status-list/
- ToIP TRQP spec: https://trustoverip.github.io/trust-registry-protocol/
- EBSI Trusted Issuers Registry: https://ec.europa.eu/digital-strategy/our-policies/ebsi
- EBSI Trusted Schemas Registry: https://ec.europa.eu/digital-strategy/our-policies/ebsi

10) Appendix: Minimal Snippets

10.1 `did:web` location rule (example)

```
DID: did:web:example.com
DID Document URL: https://example.com/.well-known/did.json
```

10.2 Status List 2021 credential pointer (within a VC)

```
"credentialStatus": {
  "type": "StatusList2021Entry",
  "statusPurpose": "revocation",
  "statusListIndex": "4242",
  "statusListCredential": "https://issuer.example/status/employee-
2025.json"
}
```

10.3 Holder-bound VP (sketch)

```
{
  "type": ["VerifiablePresentation"],
  "holder": "did:web:alice.example",
  "verifiableCredential": [ /* derived proofs with selective
disclosure */ ],
  "proof": {
    "challenge": "d3f7b9...",    // verifier’s nonce
    "domain": "login.service.example" // audience binding
  }
}
```

End of guide.