SOVRA

The Digital Identity Stack

for humans — and agents.

Trust once. Use everywhere.

> **Decentralized identity infrastructure for government-issued verifiable credentials —** *and any institution that needs to issue, hold, or verify trusted documents.*

Version 1.0 — March 2026

Authors: Diego Fernández · Chuy Cepeda · Alecs Garza · Lucas Jolias · Matías Onorato · Manuel Iñaki Bilbao

Website: sovra.io · **GitHub:** github.com/sovrahq

CONTENTS

Table of Contents

PART 01

The Problem: Digital Trust Is Broken

Imagine if every time you walked into a building, you had to re-prove that you exist. Show your birth certificate. Wait for someone to call your mother. Do it again at the next building. That is how digital identity works today.

Every bank, hospital, government office, and employer runs its own verification process. You submit the same documents over and over. Each institution stores its own copy of your data—most of them unable to verify it is real. The result is a system that is slow, expensive, fragile, and increasingly dangerous.

Three forces are breaking it faster

$50B+ Wasted Annually

Institutions globally spend over $50 billion per year on redundant identity verification. The same person proves the same facts to dozens of organizations, each paying for the same checks.

80% Data Redundancy

Every service keeps its own copy of your data. Most cannot verify whether it is authentic. This creates massive duplication, inconsistency, and an ever-growing attack surface for breaches.

96% Deepfake Accuracy

AI-generated synthetic media—fake IDs, forged documents, cloned voices—now reaches 96% accuracy. Without cryptographic proof, seeing is no longer believing.

Zero Portability

A credential issued in one system cannot be verified in another. Your government ID does not talk to your bank. Your diploma does not talk to your employer. Every boundary means starting over.

The Inevitability Arc

This is not a problem that will fix itself. It is accelerating. And the path forward is now clear:

1 — **AI Breaks Trust**
Synthetic media makes visual verification obsolete

→

2 — **Trust Must Become Verifiable**
Cryptographic proof replaces human judgment

→

3 — **Identity Becomes Infrastructure**
Shared, reusable, privacy-preserving

Identity is not an application. It is the infrastructure layer that every application needs.

Four Converging Forces

Four independent trends are converging to make verifiable digital identity not just possible, but inevitable.

FORCE 01

AI Breaks Trust

Deepfakes, synthetic identities, and AI-generated documents make it impossible to trust what you see. A fake government ID can be generated in seconds. The only antidote is cryptographic proof—identity that is mathematically verifiable, not visually convincing.

FORCE 02

Regulation Is Converging

Europe's eIDAS 2.0 mandates digital identity wallets for all EU citizens by 2026. Latin America, Africa, and Asia are following with their own frameworks. Compliance demands are rising globally—and they all point toward verifiable credentials as the standard.

FORCE 03

Cryptography Has Matured

Verifiable Credentials, Zero-Knowledge Proofs, and Decentralized Identifiers are no longer academic research. They are deployable, standardized by the W3C, and battle-tested at national scale. The technology is ready.

FORCE 04

Web3 Primitives Are Ready

Portable, user-controlled identity is no longer theoretical. Ethereum Layer 2 solutions make blockchain anchoring affordable at scale. Passkey-based wallets eliminate seed phrases. The infrastructure exists. The question is no longer *if* but *who deploys it first*.

What "Reusable Identity" Means

Today, proving your identity works like showing your passport to every person you meet, every day. Reusable identity works differently: you prove who you are once, and from that point forward, anyone can verify it instantly—without seeing the original document, without calling the issuer, without storing your data.

Today (Broken)

Verify at Bank A. Verify again at Bank B. Verify again at Insurance C. Repeat at every new service. Each one stores your data. Each one can be breached.

With Reusable Identity

Verify once. Receive a cryptographic credential on your phone. Present it at Bank A, Bank B, Insurance C—anywhere. Verification takes under one second. Your data stays on your device.

Think of it like a digital passport that lives on your phone. But unlike a passport, it cannot be forged, you control exactly which pages anyone sees, and verifying it is instant and free.

The Cost of Inaction

The current system is not just inefficient. It is actively harmful. Every year it persists, the costs compound:

$50B+	80%	4.5B	850M+
Wasted annually on redundant verification	Of identity data is redundantly stored	Records breached in 2023 alone	People worldwide lack any legal ID

For governments, the cost is bureaucratic drag and eroded public trust. For businesses, it is compliance overhead and fraud exposure. For citizens, it is wasted time, lost privacy, and exclusion from services that require proof of identity they cannot easily produce.

> *"The internet was built without an identity layer. Every security problem, every privacy scandal, every cumbersome login flow traces back to that missing piece. It is time to build it."*

The technology exists. The standards are ratified. The regulatory tailwinds are blowing. What was missing was a team that could build it at institutional scale, deploy it with governments, and make it work for real people. That is what Sovra does.

PART 02

The Solution: The Sovra Stack

Sovra builds the identity infrastructure the internet never had. It is a complete stack—from citizen-facing wallets to blockchain-anchored proofs—designed to make digital identity verifiable, private, portable, and reusable at national scale.

The Trust Triangle

Every Sovra credential involves three parties. Understanding this triangle makes everything else click.

1 — Issuer
Government, university, hospital creates and signs a credential

→

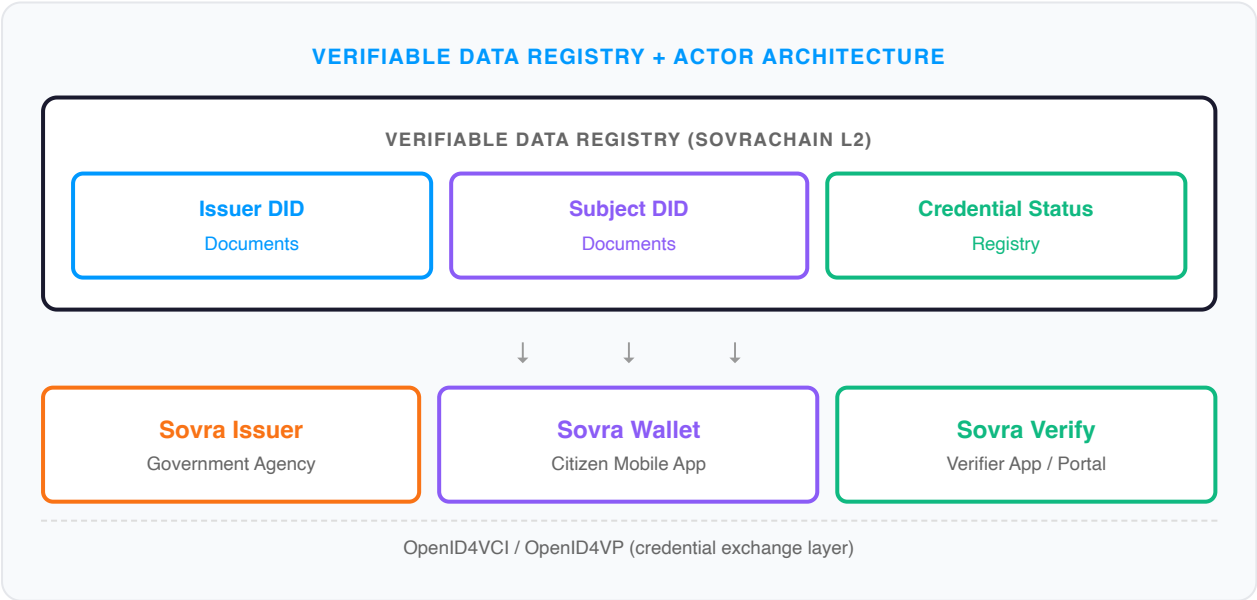
2 — Holder
Citizen stores it in their wallet on their phone

→

3 — Verifier
Bank, employer, pharmacy checks it cryptographically—no callback needed

The key insight: the verifier never contacts the issuer. A bank checking your government-issued ID does not call the government. The cryptographic signature in the credential is proof enough. This means the issuer never learns where or when you use your credential—your privacy is protected by mathematics, not policy.

System Architecture



The Product Suite

Sovra is built as four products that work together like layers of a single system. Each serves a different audience, but they share one infrastructure.

FOR INSTITUTIONS GOING DIGITAL

SovraGov

A no-code platform for governments and organizations that need to move from paper to digital. Configure services visually, issue verifiable documents, and onboard citizens—without writing code. Already used for driver's licenses, construction permits, commercial licenses, and civil registry across Latin America.

Think of it as: Shopify for government services—but with cryptographic trust built in.

FOR BUILDERS & INSTITUTIONS

SovraID

The identity engine. APIs and protocols that let any institution issue, verify, and manage digital credentials—plugging into existing systems without migration. REST APIs, TypeScript SDK, React hooks, and storage adapters. Certified as a Digital Public Good.

Think of it as: Stripe for identity—clean primitives, simple integration, standards-compliant.

FOR CITIZENS

SovraWallet

A free mobile app where citizens store all their digital documents—IDs, diplomas, prescriptions, licenses. You control who sees what. Share only what is needed. Verify in under a second. Open-source, available on iOS and Android. Already used by 1.2M+ people.

Think of it as: Apple Wallet for your identity—but you own the data, not Apple.

THE TRUST LAYER

SovraChain

An Ethereum Layer 2 blockchain that anchors every credential with cryptographic proof. Personal data never touches the chain—only proofs of authenticity. Built on Ethrex (LambdaClass) with ZK proofs by SP1 (Succinct Labs). 99.99% uptime, sub-second finality.

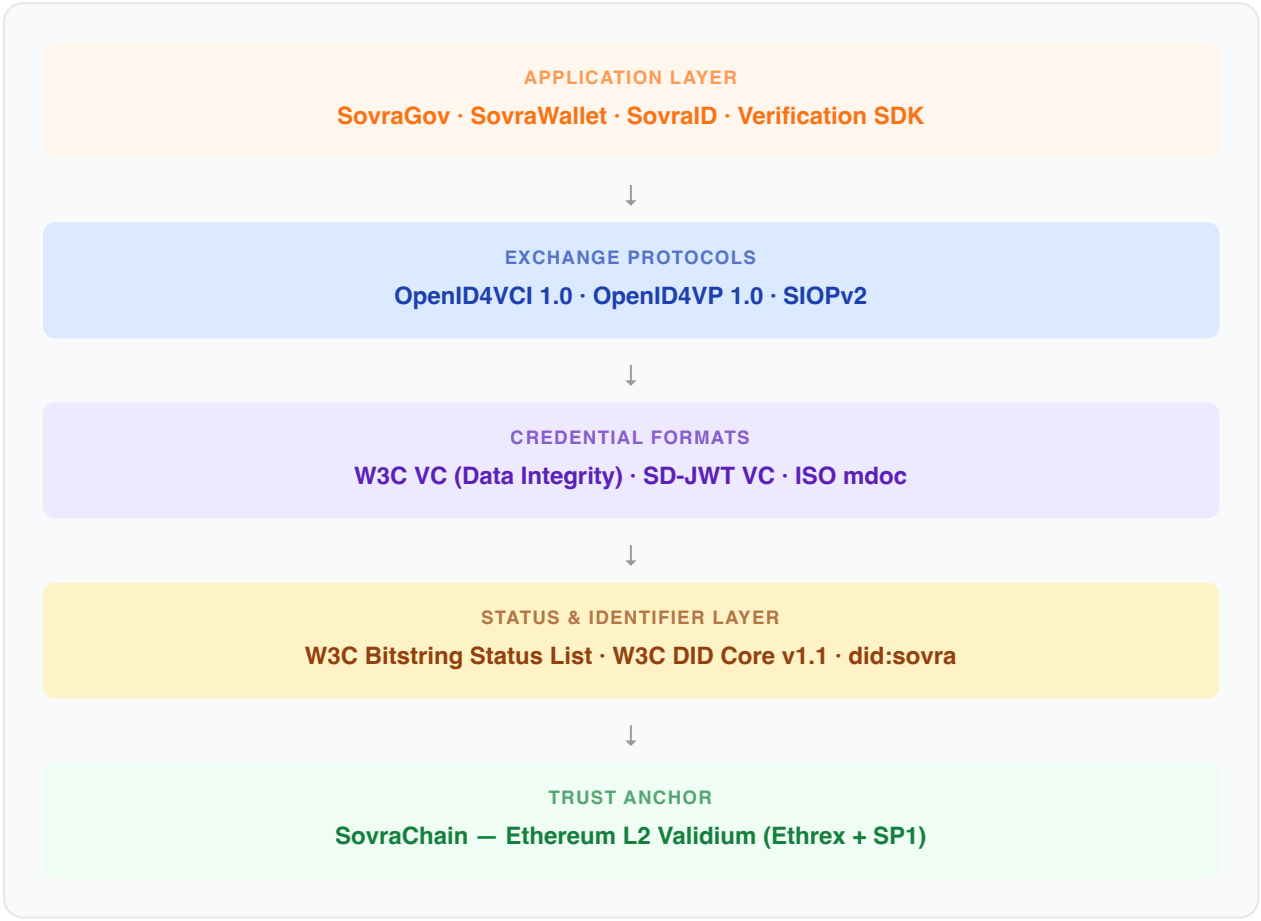
Think of it as: The notary stamp that makes everything unforgeable—but decentralized and instant.

Which product is right for you?

AUDIENCE	SITUATION	START WITH
Government with no digital services	Need to go digital from scratch	**SovraGov** (includes SovraID + SovraWallet)
Government with existing portals	Already digital, need verifiable credentials	**SovraID** + SovraWallet
Hospital, university, or bank	Already have digital systems	**SovraID** + SovraWallet
Developer building an app	Need identity features	**SovraID** (APIs & SDKs)
Citizen or end user	Want to hold and share credentials	**SovraWallet** (free)

System Architecture

The Sovra stack is organized in layers. Each layer builds on the one below it, and every component maps to a ratified open standard.



Standards Stack

No proprietary protocol is required. Every layer maps to ratified specifications.

Application Layer	SovraGov · SovraWallet · SovraID
Exchange Protocols	OpenID4VCI 1.0 · OpenID4VP 1.0 · SIOPv2
Credential Formats	W3C VC Data Integrity · SD-JWT (RFC 9901) · ISO mdoc
Status Mechanism	W3C Bitstring Status List v1.0
Identifier Layer	W3C DID Core v1.1 · did:sovra method
Trust Anchor	SovraChain — Ethereum L2 Based Rollup (Validium)

On-Chain vs. Off-Chain: What Goes Where

The ledger stores **trust anchors and status**. It never stores personal data. This is a fundamental design principle—not a tradeoff.

ON-CHAIN (PUBLIC, IMMUTABLE)	PURPOSE
DID Documents	Map DIDs to public keys and endpoints. Public, append-only, updatable by controller only. Under 1 KB each.
Credential Status Lists	Compressed bitstring per issuer. 131,072 entries in a few hundred bytes. No personal data—just bits.
Credential Schemas	Structural definitions. Public, immutable once published.
Trusted Issuer Registry	Which DIDs are authorized to issue which credential types. Governance-controlled.
DID Operation Log	Timestamped create/update/deactivate events. Tamper-evident, append-only audit trail.

OFF-CHAIN (PRIVATE, USER-CONTROLLED)	LOCATION
Verifiable Credentials	Citizen's device, encrypted at rest (AES-256-GCM)
Credential claim values	Inside the VC, in the wallet—never transmitted except by choice
Private keys	Device secure enclave (never leaves hardware in plaintext)
Presentation records	Ephemeral—wallet ↔ verifier only, not stored after session

Why this matters: No personal data on an immutable ledger means **zero GDPR tension**. The right to erasure applies to off-chain data under user control. The ledger holds only cryptographic anchors—public keys, status bits, and structural schemas. Credential content never touches the chain.

Key Flows

1. Citizen Onboarding

A citizen downloads SovraWallet and creates their digital identity in under two minutes.

1 — Download
Install SovraWallet from App Store or Google Play

→

2 — Create Passkey
Generate P256 passkey in device secure enclave via WebAuthn

→

3 — Register DID
Anchor identity on SovraChain (gas-free via EIP-7702 sponsorship)

→

4 — Ready
Receive credentials from any institution

2. Credential Issuance (OpenID4VCI)

A government office issues a driver's license to a citizen who verified their identity in person.

1 — Offer
Issuer generates QR code with credential_issuer, pre-authorized_code

→

2 — Authenticate
Wallet fetches metadata, exchanges code for access_token + c_nonce

→

3 — Issue
Issuer signs credential with assertionMethod key, binds to holder's DID

→

4 — Store
Credential saved encrypted in wallet, holder is sole custodian

3. Credential Verification (OpenID4VP)

A bank needs to verify a citizen's government-issued ID to open an account.

1 — Request
Verifier shows QR with client_id + request_uri (DCQL query)

→

2 — Consent
Citizen reviews and approves which fields to share

→

3 — Present
Wallet sends vp_token with selected claims + BBS+/KB-JWT proof

→

4 — Verify
Verifier checks signature, status, DID — all in under 1 second

---

> *"The verifier never contacts the issuer. The government never knows when or where you used your ID. Your privacy is protected by mathematics, not by policy."*

What "No Callback" Means in Practice

In traditional systems, a bank calls a university to verify a degree. With Sovra, the cryptographic signature embedded in the credential is proof enough. The bank confirms authenticity by checking the signature against the issuer's public key (published on SovraChain). No network request to the issuer. No tracking. No correlation.

PART 03

The Technology

Self-Sovereign Identity: The Paradigm Shift

Traditional identity works like this: some organization stores your data, and you ask them for permission to use it. They control it. You do not. Self-Sovereign Identity (SSI) flips this model. You own and control your own identity. Your data lives on your device, and you decide who sees what.

Christopher Allen formalized this in ten principles. Sovra implements all of them:

1 EXISTENCE	2 CONTROL	3 ACCESS	4 TRANSPARENCY	5 PERSISTENCE
6 PORTABILITY	7 INTEROP	8 CONSENT	9 MINIMIZATION	10 PROTECTION

In practice, this means: **Data minimization**—only collect what is strictly necessary. **Selective disclosure**—show only what is relevant. **No phone home**—issuers never learn when or where credentials are used. **Open standards**—no vendor lock-in. **Portability**—switch providers without losing credentials.

Decentralized Identifiers (DIDs)

A DID is like a username, but one that you own—no company or government controls it. In Sovra, every person and institution gets a DID anchored on SovraChain.

```
did:sovra:0x7f3a...4b2c
```

You Create It

No registration authority needed. You generate it on your device with a single tap.

You Control It

No one can take it away, modify it, or revoke it without your authorization.

It Is Universal

Works across any system that supports the W3C DID standard—globally interoperable.

It Contains No Personal Data

A DID is just a pointer to your public key—like a mailbox address, not the mail itself.

Each DID resolves to a DID Document containing separate key material for five verification relationships: **authentication** (prove DID control), **assertionMethod** (sign credentials), **keyAgreement** (encrypted channels), **capabilityInvocation** (update DID), and **capabilityDelegation** (delegate authority).

TECHNICAL DEEP-DIVE: DID:SOVRA METHOD

Under the hood, `did:sovra` identifiers follow a strict ABNF syntax and resolve through a dual-path algorithm depending on actor type.

PROPERTY	DETAIL
Syntax	`did:sovra:0x` + 40 lowercase hex digits (20-byte Ethereum address, no EIP-55 checksum)
Holder Resolution	Check if address has EIP-7702 delegation to `SovraDispatcher` → read `passkeyX/Y`, `recoveryKeyX/Y`, DID metadata directly from wallet contract storage via `eth_call`
Issuer/Verifier Resolution	Call `SovraDIDRegistry.resolve(address)` → returns verification methods (with relationship bitmasks), service endpoints, timestamps, version
Historical Versions	`?versionId=N` query parameter → resolver walks `DIDCreated` / `DIDUpdated` events to reconstruct the DID Document at version N
Key Binding	SD-JWT `cnf.kid` references `did:sovra:<addr>#passkey` — key rotation does not require credential re-issuance
Deactivation	Permanent. DID resolves with empty `verificationMethod` and `deactivated: true`. All bound credentials become unverifiable.

Conforms to W3C DID Core v1.1. Holder DID Documents contain two P256 keys (`#passkey` for all relationships, `#recovery-key` in `verificationMethod` only). Issuer documents add `service` entries for OID4VCI and status endpoints.

Verifiable Credentials & Digital Signatures

A Verifiable Credential (VC) is the digital version of any physical document—a passport, a diploma, a medical prescription, a driver's license. But unlike a photocopy or a PDF, a VC has a cryptographic signature that makes it impossible to forge, instant to verify, and privacy-preserving by design.

How Digital Signatures Work

Think of a wax seal on a medieval letter. If the seal is intact, you know the letter has not been opened or changed. Digital signatures are the unbreakable version of that seal.

1
Sign
Issuer uses their private key to create a mathematical stamp on the credential

→

2
Distribute
Holder receives the signed credential in their wallet

→

3
Verify
Anyone with the public key can confirm the stamp is real—if anything changed, the stamp breaks

Credential Formats

Sovra supports three credential formats to ensure interoperability across different ecosystems:

FORMAT	STANDARD	SIGNATURES	BEST FOR
W3C VC (Data Integrity)	W3C VC Data Model v2.0	BBS+, EdDSA	Maximum privacy (ZKP selective disclosure)
SD-JWT VC	IETF RFC 9901	ES256, EdDSA	OpenID4VC integration, eIDAS compliance
ISO mdoc	ISO/IEC 18013-5	ECDSA (P-256)	Mobile driver's licenses, proximity, offline

Zero-Knowledge Proofs: Proving Without Revealing

This is the most powerful concept in Sovra's technology. A Zero-Knowledge Proof (ZKP) lets you prove something is true without revealing any of the underlying information. This is not a metaphor. It is real mathematics.

Prove you are over 21

Without revealing your actual birthday, your name, or your address. The verifier learns one fact: "yes, this person is over 21."

Prove your income exceeds a threshold

Without revealing the exact amount. A landlord confirms you can afford the rent without seeing your bank balance.

Prove you hold a valid credential

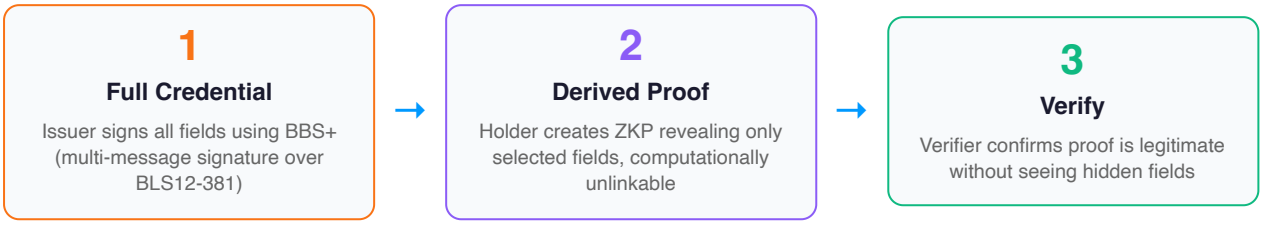
Without revealing its contents. An employer confirms you have a medical license without seeing any other details.

Prove group membership

Without revealing which member you are. Confirm you belong to an organization anonymously.

BBS+ Signatures & Selective Disclosure

Imagine your driver's license contains: name, photo, birthdate, address, ID number, organ donor status. Today, if someone needs to verify your age, you show the whole thing. With BBS+ selective disclosure, you create a proof that says "this person is over 21"—and that is all the verifier sees.



Why BBS+ matters: Unlike SD-JWT (which reveals hashes of hidden fields), BBS+ provides true unlinkability—two presentations of the same credential cannot be correlated. A constant 80-byte signature regardless of message count. Proof suite: `bbs-2023`.

PROPERTY	BBS+	SD-JWT	ISO MDOC
Mechanism	ZKP of signature knowledge	Hash-and-reveal	Hash-and-reveal (MSO)
Hidden claim count visible?	**No**	Yes	Yes
Presentation unlinkability	**Yes**	No	No
Holder binding	Optional	Optional (cnf + KB-JWT)	Mandatory (DeviceAuth)

ISO mdoc: Proximity & Offline Credentials

While SD-JWT powers Sovra's online flows, **ISO mdoc** (ISO/IEC 18013-5) is the international standard for mobile credentials in proximity scenarios—a police officer checking your license at a traffic stop, a border agent verifying your passport, or a pharmacy checking your prescription face-to-face.

Mobile Security Object (MSO)

The issuer signs a `COSE_Sign1` structure containing a `valueDigests` map: each claim is hashed with a random salt. The MSO is the cryptographic anchor—like a tamper-proof envelope listing the fingerprints of every document inside.

Selective Disclosure via Hash-and-Reveal

Each claim is an `IssuerSignedItem` with its own salt. The holder transmits the full MSO (so the verifier can check the issuer's signature) but only includes the `IssuerSignedItems` for fields they choose to share. Unrevealed claims remain hashes.

DeviceAuth: Mandatory Holder Binding

Unlike SD-JWT (where holder binding is optional), mdoc **requires** DeviceAuth on every presentation. The holder's device signs a session transcript proving physical possession—a stolen credential file is useless without the device.

CBOR Binary Encoding

mdoc uses CBOR (Concise Binary Object Representation), not JSON. This makes it compact and efficient for constrained environments: NFC tap, BLE exchange, or offline verification without internet connectivity.

How mdoc fits the Sovra stack: SovraWallet stores and presents all three formats (W3C VC, SD-JWT, ISO mdoc). OpenID4VP carries all three through the same flow via ISO/IEC 18013-7:2024. Format choice is a deployment decision—SD-JWT for online flows, mdoc for proximity, W3C VC (BBS+) when maximum privacy is required.

SovraChain: The Trust Layer

SovraChain is a purpose-built Ethereum Layer 2 blockchain optimized for identity. It exists for one reason: to make every credential in the Sovra ecosystem tamper-proof and independently verifiable.

SOVRACHAIN

Identity Infrastructure for Millions

EXECUTION	TYPE	PRIVACY
Ethrex	**Based Rollup**	**Validium**
LambdaClass · Rust	L1 sequencing	Off-chain data, on-chain proofs

ZK PROOFS	FOUNDATION
SP1 (Succinct Labs)	**Ethereum Layer 2**

Key innovation: EIP-7702 gas sponsorship. Citizens never need tokens or cryptocurrency. SovraChain sponsors all gas fees for identity operations. Passkey-based authentication (via EIP-7212 P256 precompile) eliminates seed phrases entirely.

TECHNICAL DEEP-DIVE: SOVRACHAIN — 9 SYSTEM COMPONENTS

The full SovraID system spans 4 on-chain contracts, 3 off-chain services, and 2 client applications, all coordinated through the Ethrex L2.

#	COMPONENT	TYPE	ROLE
1	Ethrex L2 Appchain	Chain	Execution layer. Native gas sponsorship for 7702 accounts. EIP-7212 P256 precompile.
2	EIP-7702 Wallet Contract	On-chain	Holder identity. Passkey validation, key rotation, DID metadata, 7-day recovery timelock.
3	DID Registry	On-chain	Issuer/verifier DID Documents. Verification methods with relationship bitmasks + service endpoints.
4	Issuer Registry	On-chain	Trust framework. Credential-type-scoped issuer authorization. Governance-controlled.
5	Revocation Registry	On-chain	Per-issuer bitmaps (W3C Bitstring Status List). Batch revocation + suspension support.
6	Recovery Server	Off-chain	HSM-backed P256 recovery key per holder. OAuth authentication. Recovery-only role.
7	Wallet App	Client	React Native. Passkey generation, encrypted credential storage, OID4VCI/VP flows.
8	Issuer Service	Off-chain	OID4VCI endpoints, schema management, HSM-backed credential signing. No claim retention.
9	Verification SDK + Verifier App	Client + Lib	12-step SD-JWT validation pipeline. DID resolution, trust checks, relay-based cross-device flow.

Build order: DID Method Spec → L2 Chain → Wallet Contract → DID Registry → Recovery Server → Wallet App → Issuer Registry → Revocation Registry → Issuer Service → Verification SDK. Tech stack: Solidity/Foundry (contracts), Elixir/Phoenix (backend), React Native/Expo (mobile), React/Vite (issuer portal), `@sovra/wallet-core` (shared TypeScript).

Security Model & Privacy Guarantees

Sovra's security is not based on trusting a single party. It is based on cryptographic guarantees that hold even if individual components are compromised.

Key Management

COMPONENT	HOW IT WORKS
Key Generation	All private keys generated inside hardware-backed secure enclaves (Apple Secure Enclave, Android StrongBox). Key material never leaves the enclave in plaintext. Issuers may use HSMs (FIPS 140-2/3).
Key Rotation	Lost or compromised device: authenticate via recovery, generate new keys on new device, update DID Document on-chain, deactivate old keys. Credentials remain valid (bound to DID, not specific key).
Recovery	2-of-2 multisig model (P256 passkey + P256 server key). Guardian co-signing server with HSM-backed key storage, OAuth/SSO authentication, rate limiting, and fraud detection before recovery signing.

Threat Mitigations

THREAT	MITIGATION
Malicious verifier tries to extract data	Selective disclosure + nonce/audience binding + pairwise DIDs
Stolen credential presented by attacker	Holder binding (TEE key) + biometric binding + device attestation
Compromised issuer creates fake credentials	Key rotation + ledger anchoring + short-lived credentials + multi-sig
Man-in-the-middle network attack	TLS 1.3 + ledger-anchored DIDs + nonce/audience binding
Issuer + verifier collude to track citizen	BBS+ unlinkability + pairwise DIDs + credential ID not disclosed

Privacy by Design

No Persistent Correlation

Pairwise DIDs: A holder MAY create a separate `did:sovra` for each verifier relationship. Each pairwise DID is a separate EOA with its own EIP-7702 delegation and its own key material—preventing cross-verifier correlation at the DID level. For maximum privacy, ephemeral session-scoped DIDs can be used for one-time interactions.

No Issuer Tracking

Status lists use 131,072+ entry herd privacy. Verifier caching + CDN distribution + holder stapling (wallet attaches cached status list for offline verification) + on-chain anchoring. Issuers never know when or where a credential is presented.

No Session Linkage

BBS+ proof unlinkability ensures two presentations of the same credential cannot be correlated. Combined with ephemeral DIDs and attribute coarsening policies, even repeated interactions with the same verifier remain unlinkable.

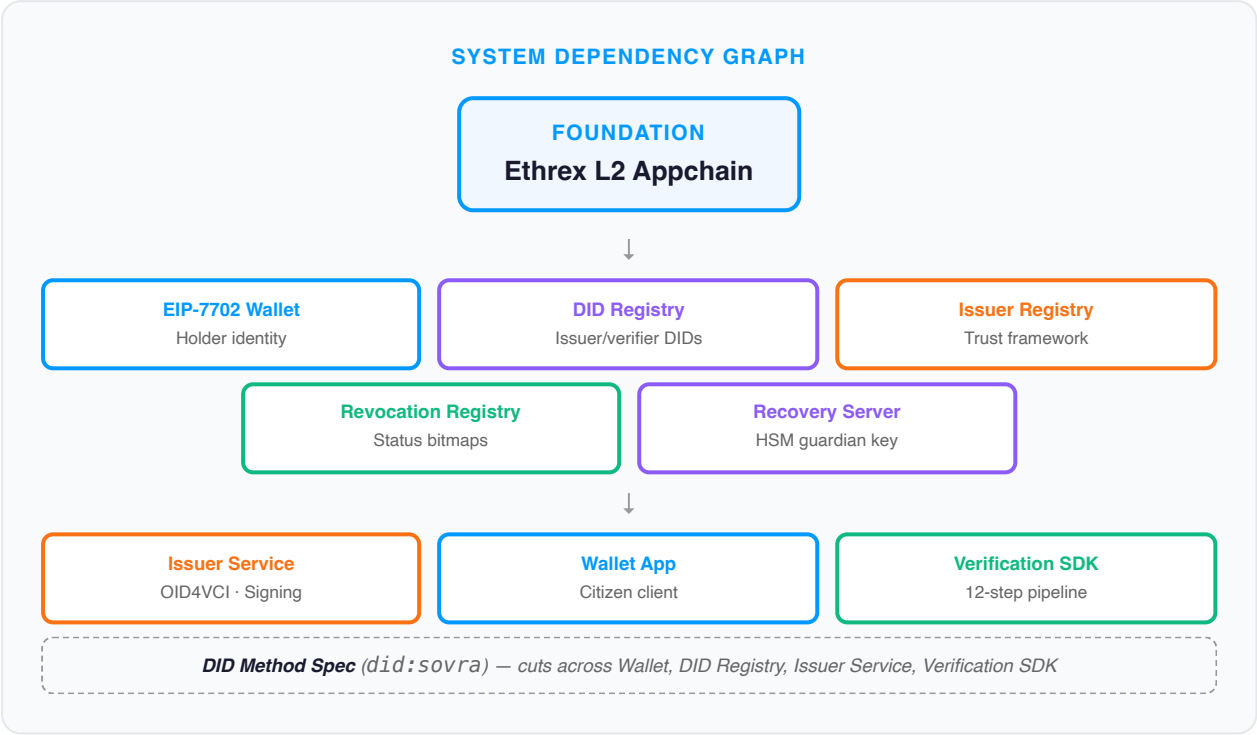
No Metadata Leakage

Random index assignment (no issuance order leak), binary-only status (no circumstance leak), encrypted TLS 1.3 transport. Gas sponsorship prevents financial identity leakage—on-chain patterns reveal nothing about the citizen.

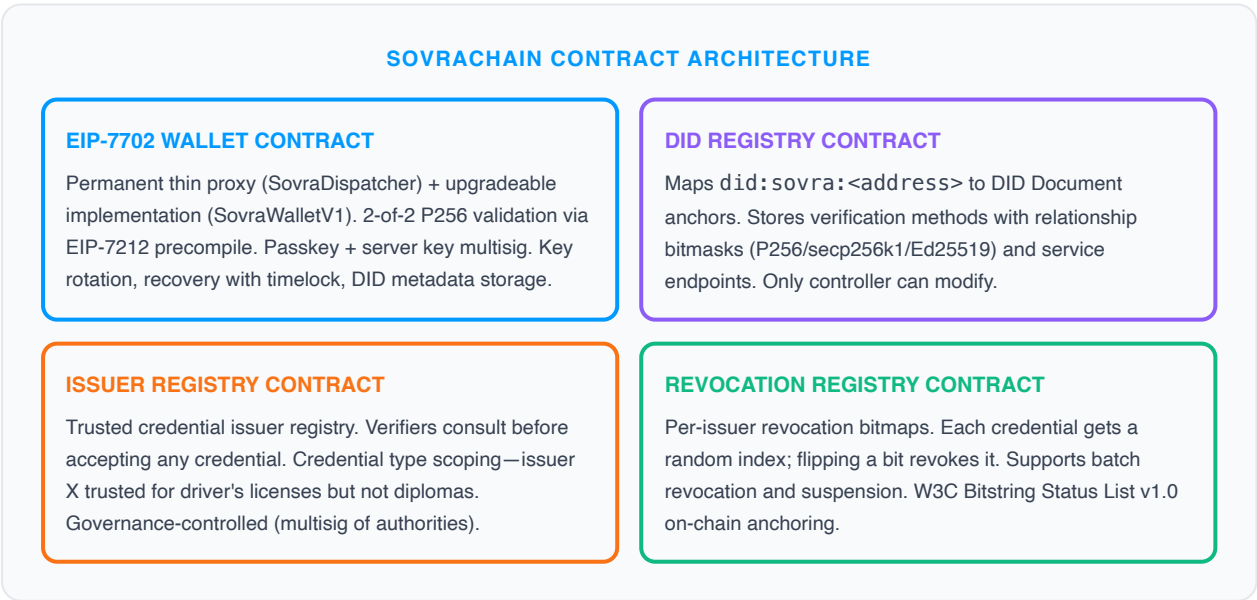
On-chain observability note: DID operations are public transactions. Creation timestamps and update frequency are visible. Gas sponsorship prevents financial correlation, and pairwise DIDs prevent identity correlation, but transaction pattern analysis remains a theoretical risk. Recovery keys SHOULD be generated randomly with no derivation relationship between holders.

Component Deep Dive

The Sovra system consists of nine tightly integrated components spanning on-chain smart contracts, off-chain services, and client applications. The dependency graph below shows how they relate:



On-Chain Smart Contracts



Off-Chain Services

COMPONENT	ROLE	KEY CAPABILITIES
Issuer Service	Credential issuance platform for government agencies	DID registration, VC signing (assertionMethod key), OpenID4VCI delivery, status list management, schema publishing. Does not store copies of issued credentials.
Co-signing Server	Guardian key holder for wallet recovery	HSM-backed P256 key storage, OAuth/SSO authentication before recovery signing, rate limiting, fraud detection, timelock enforcement. Recovery only—cannot initiate regular operations.
Verification SDK	Client library for verifiers	DID resolution, signature verification, status list checking, OpenID4VP presentation requests (same-device redirect + cross-device QR). Does not contact issuing agency.

TECHNICAL DEEP-DIVE: OID4VCI ISSUANCE FLOW

The Issuer Service implements the OpenID4VCI Pre-Authorized Code flow. Sovra manages all cryptographic operations — issuers never handle key material directly.

ENDPOINT	PURPOSE
`GET /.well-known/openid-credential-issuer`	Issuer metadata: supported credential types, endpoints, cryptographic suites
`POST /token`	Exchange `pre-authorized_code` + optional `tx_code` (PIN) for `access_token` + `c_nonce`
`POST /credential`	Wallet submits key possession proof JWT (signed with passkey, containing `c_nonce`). Service signs SD-JWT with issuer's P256 key (HSM-backed) and returns it.
`POST /deferred_credential`	Poll for credentials not ready immediately (returns `transaction_id`)

Each issuer gets a dedicated EOA (secp256k1, for L2 txs) and P256 signing key (for credentials), both in HSM. Credential claim values are discarded after delivery — the service retains only metadata (holder DID, status, timestamps). Schemas are versioned and stored off-chain. The `credentialType` string is hashed to `bytes32` for Issuer Registry lookups.

Credential Schema Example — issuers define schemas through the dashboard, specifying which claims are selectively disclosable:

CLAIM	TYPE	REQUIRED	DISCLOSABLE
`given_name`	string	✓	✓
`family_name`	string	✓	✓
`birth_date`	date	✓	✓
`document_number`	string	✓	✓
`issuing_country`	string	✓	—
`expiry_date`	date	✓	✓
`vehicle_category_code`	string	✓	✓

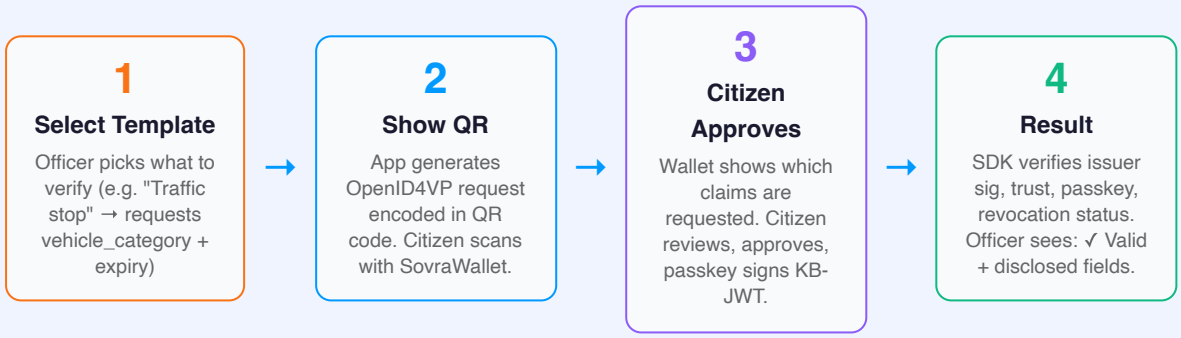
Schema ID: `driving-licence-v1`. Claims marked *disclosable* are wrapped in salted hashes (SD-JWT) so the holder chooses what to reveal. Non-disclosable claims (like `issuing_country`) are always visible—they provide context without leaking personal data. Validity period (e.g., 5 years) and credential type (`DrivingLicenceCredential`) are configured per schema.

Client Applications

APP	PLATFORM	KEY CAPABILITIES
SovraWallet	React Native / Expo (iOS + Android)	P256 passkey generation via WebAuthn/FIDO2, encrypted credential storage, selective disclosure, OpenID4VCI/VP flows, QR scanning, biometric access
Verifier App	React Native / Expo (iOS + Android)	QR-based presentation requests, credential verification via backend API, verification history
Issuer Portal	React / Vite (Web)	Credential type management, issuance dashboard, OID4VCI offer generation, WebAuthn for issuer authentication

TECHNICAL DEEP-DIVE: VERIFY APP — IN-PERSON VERIFICATION FLOW

The Sovra Verify App is an official mobile app used by verifier organizations (police, employers, border agents) for in-person credential checks via QR code.



Verification templates eliminate manual configuration per check:

TEMPLATE	CREDENTIAL TYPE	CLAIMS REQUESTED
Traffic stop	DrivingLicenceCredential	vehicle_category_code, expiry_date
Age check	NationalIDCredential	birth_date
Identity check	NationalIDCredential	given_name, family_name, birth_date, document_number
Employment check	DiplomaCredential	degree, institution, graduation_date

Each verifier organization authenticates to their own DID (`did:sovra:0xbbbb...`). The officer picks a template, the app generates the QR—zero configuration per interaction. Verification history is stored locally on device for audit.

Wallet Architecture

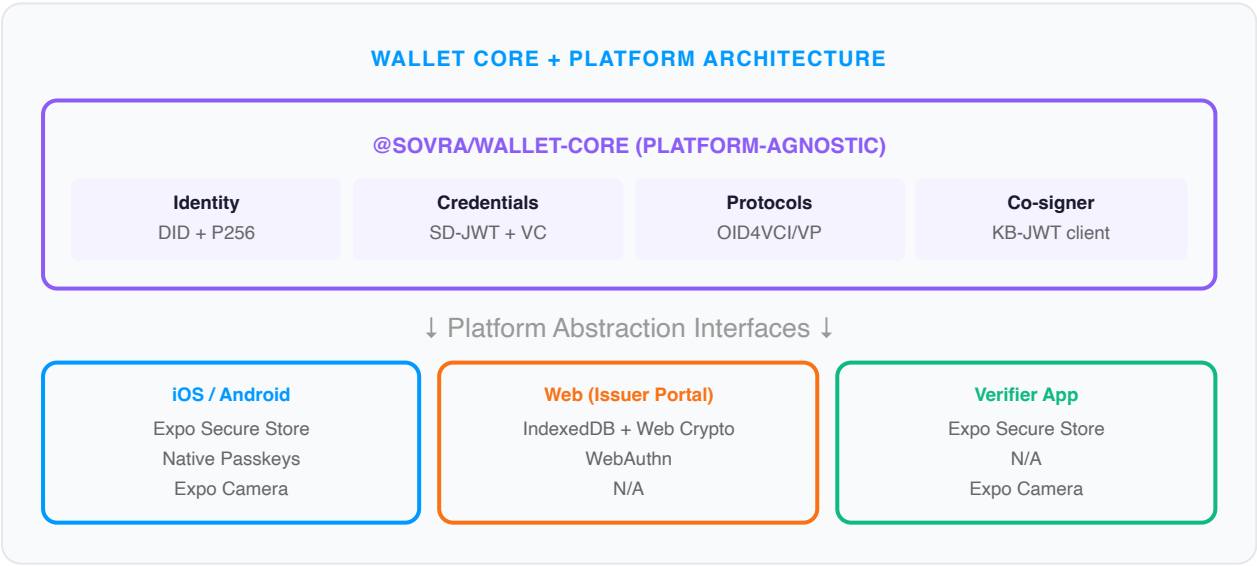
The SovraWallet uses a shared-core architecture: platform-agnostic logic lives in a TypeScript package (`@sovra/wallet-core`), while platform-specific capabilities (secure storage, passkeys, camera) are abstracted through interfaces implemented per platform.

TECHNICAL DEEP-DIVE: WALLET APP PLATFORM ABSTRACTION

The wallet app abstracts all platform-specific operations behind interfaces so the same credential logic runs on iOS, Android, and web.

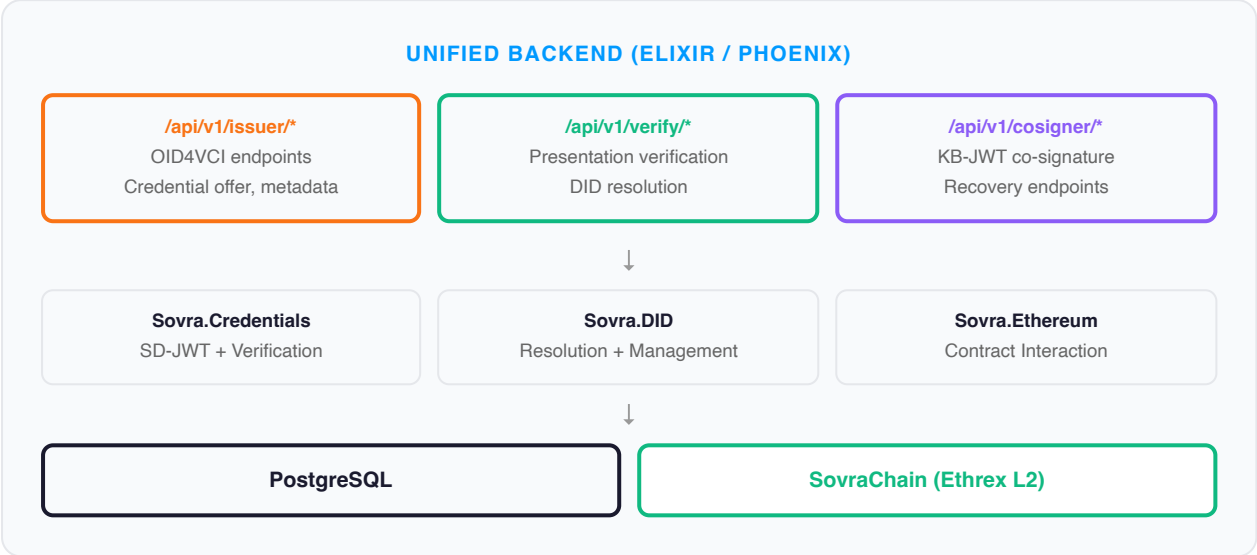
INTERFACE	IOS	ANDROID	WEB
`SecureStorage`	Secure Enclave via `expo-secure-store`	StrongBox Keystore	IndexedDB + Web Crypto API
`PasskeyProvider`	Native ASAuthorization API	FIDO2 / Credential Manager	`navigator.credentials` (WebAuthn)
`CameraScanner`	`expo-camera` (QR scanning)		N/A (deep links)

Credentials are encrypted at rest: `AES-256-GCM(sdJwt, SecureEnclave.deriveKey("sovra-credential-storage"))`. The encryption key never leaves hardware. Credentials are device-only — no cloud backup. Lost device requires re-issuance after recovery. Presentations are fully off-chain (wallet → verifier via HTTP); no L2 interaction during verification. BLE/NFC for in-person presentation is under consideration for future versions.



Backend Architecture

The unified backend is built on Elixir/Phoenix, providing a single API surface for all three actors.



PART 04

Real-World Impact

Sovra is not a whiteboard architecture. It is deployed, live, and serving millions of citizens across three countries.

Deployment Metrics

3 Countries Mexico · Argentina · Colombia	**8M+** Citizens with access to verifiable credentials	**1.5M+** Digital identities issued	**20+** Government deployments
50+ Implementations underway	**10M+** Credentials issued	**50M+** Verifications processed	**100M+** API calls handled

Case Studies

NUEVO LEÓN, MEXICO

"Burocracia Cero"

A digital government initiative that made public services up to 80% faster and cut bureaucracy by 30%. Citizens complete government procedures digitally instead of waiting in lines with paper documents. Driver's licenses, construction permits, and commercial licenses—all verifiable from a phone.

80% faster

SALTA, ARGENTINA

"Identidad Digital"

Over 1.4 million citizens gained access to verifiable digital credentials, enabling secure access to provincial government services from their phones. The largest subnational digital identity deployment in Argentina.

1.4M credentials

LUJÁN, ARGENTINA

"Mi Luján Digital"

A municipality of 120,000+ citizens went fully digital: 22% less bureaucracy, 66% fewer requirements to complete services, and 45% faster case resolution. A model for mid-sized city digital transformation.

45% faster resolution

BOGOTÁ, COLOMBIA

"Ciudad Digital"

Capital city digital transformation bringing verifiable credentials to city services for millions of residents. One of the largest urban digital identity deployments in Latin America, covering government services across the metropolitan area.

Incoming

Industry Applications

Verifiable identity applies anywhere people need to prove something about themselves. Here is how it works across eight industries.

INDUSTRY	PROBLEM TODAY	WITH SOVRA	KEY USE CASES
Government	Paper processes, long lines, manual verification	Digital services, instant verification, 80% faster	Citizen IDs, licenses, permits, benefits, civil registry
Healthcare	Repeated health info, paper prescriptions, locked records	Digital prescriptions, portable records, ZKP privacy	Vaccination proofs, Rx, provider licenses, insurance claims
Education	Diploma fraud, slow verification, paper transcripts	Tamper-proof diplomas, instant verification	Degrees, certifications, skills badges, transcripts
Finance	Redundant KYC, slow onboarding, $billions/year wasted	Reusable KYC, instant account opening, 90% cost reduction	KYC, remote account opening, AML compliance, credit
Employment	Weeks-long background checks, resume fraud	Instant credential verification, verified work history	Work history, professional licenses, references
Retail	Full ID exposure for age checks, fragmented loyalty	Privacy-preserving age verification, portable loyalty	Age verification, shopping identity, data control
Supply Chain	Counterfeits, unverifiable claims, slow customs	Cryptographic provenance, instant compliance	Product authenticity, origin tracking, certifications
Enterprise	Password-based security, fragmented access	Credential-based zero-trust authentication	Access control, device identity, cross-platform SSO

Partner Ecosystem

Ethereum Foundation
Core blockchain infrastructure and research partnership

LambdaClass
Ethrex execution engine, core chain development

Google
Cloud infrastructure and distribution partnership

> *"Mass adoption of digital identity will not come from tech enthusiasts. It comes from governments, banks, and public services making it part of everyday life. Sovra is built for that world."*

PART 05

The Stack & Standards

Standards Compliance

Sovra is built entirely on open, ratified standards. No proprietary protocols. No vendor lock-in. Every component maps to a specification maintained by an international standards body.

STANDARD	BODY	STATUS	SOVRA IMPLEMENTATION
Verifiable Credentials v2.0	W3C	Recommendation	Core credential format
DID Core v1.1	W3C	Recommendation	did:sovra method
Controlled Identifiers v1.0	W3C	Recommendation	DID Document management
Data Integrity BBS	W3C	Recommendation / CRD	Privacy-preserving presentations
Bitstring Status List v1.0	W3C	Recommendation	On-chain status registry
OpenID4VCI 1.0	OpenID Foundation	Final Specification	Issuance flow
OpenID4VP 1.0	OpenID Foundation	Final Specification	Verification flow
SIOPv2	OpenID Foundation	Implementer's Draft	Passwordless auth
SD-JWT (RFC 9901)	IETF	Proposed Standard	eIDAS-compatible format
ISO mdoc 18013-5/7	ISO	Published Standard	Government ID format
eIDAS 2.0	European Union	Regulation	Architecture alignment

W3C VC v2.0 | W3C DID Core | W3C BBS+ | OpenID4VCI | OpenID4VP | SIOPv2 | SD-JWT RFC 9901 | ISO mdoc | eIDAS 2.0 | GDPR

Credential Revocation: Bitstring Status Lists

Sovra implements the W3C Bitstring Status List v1.0 for credential revocation and suspension. Each credential is assigned a randomly chosen index in a compressed bitstring. Bit 0 means valid; bit 1 means revoked. The list is published as a signed credential and anchored on SovraChain.

Herd Privacy

Minimum 131,072 entries per list. Random index assignment prevents inference of issuance count or rate. No individual tracking. Separate revocation (permanent) and suspension (reversible) lists.

Efficient & Offline-Ready

GZIP-compressed, base64url-encoded. Verifier-side caching. Holder stapling for offline verification. 16 KB covers 131K credentials. Multi-bit support for richer status codes.

Open-Source Commitment

Sovra believes identity infrastructure must be transparent, auditable, and free from vendor lock-in. Two certifications formalize this commitment:

Digital Public Good (DPG)

SovraID and SovraWallet are certified as Digital Public Goods by the DPGA (Digital Public Goods Alliance). This means: open-source, privacy-respecting, free from vendor lock-in, and available for any country to adopt and customize without licensing fees.

Digital Public Infrastructure (DPI)

SovraGov is recognized as Digital Public Infrastructure—foundational technology that any country can build upon, like roads or postal services for the digital economy. It provides the shared identity layer that payments, data exchange, and government services need.

What Open Source Means in Practice

- **Sovereignty:** Countries go from being users of foreign technology to creators of their own solutions. No dependency on a single vendor.
- **Auditability:** Any researcher, regulator, or developer can inspect the code that protects citizen identity. Security through transparency, not obscurity.
- **Innovation:** Build on proven solutions instead of starting from scratch. Contribute improvements back to the ecosystem.
- **Cost:** No licensing fees. Governments can deploy and customize without recurring software costs.

Credential Delegation

Forward-compatible delegation—holders can authorize others to act on their behalf. Citizen to attorney, parent to guardian, ministry to sub-agency.

DID Document Readiness

Separate capabilityInvocation and capabilityDelegation key material is registered in all DID Documents from creation. Delegation can be activated without re-issuing credentials.

Attenuation

A delegate can never exceed the delegator's authority. Each step may add restrictions (credential type, time window, scope). Enforced cryptographically, verifiable by any party.

PART 06

Vision & Roadmap

The Adoption Flywheel

Sovra's adoption strategy is built on a simple insight: mass adoption of digital identity will not come from consumers downloading an app. It will come from governments and institutions making it part of everyday life.

More verifiers attract more issuers. More credentials attract more verifiers. The network effect compounds.

Implementation Roadmap

The SovraID project follows a phased implementation plan, progressing from core infrastructure to full production deployment.

PHASE 1 — FOUNDATION (Q1 2026)

Core Infrastructure

Monorepo scaffold, smart contracts (EIP-7702 wallet, DID registry, issuer registry, revocation bitmaps), unified Elixir/Phoenix backend with DID and credential domain modules, wallet-core TypeScript package with platform abstraction.

PHASE 2 — INTEGRATION (Q2 2026)

Protocol Flows

End-to-end OpenID4VCI issuance flow, OpenID4VP verification flow, SIOPv2 passwordless authentication, co-signing server with HSM-backed recovery, mobile wallet app with passkey generation and credential storage.

PHASE 3 — PRODUCTION (Q3 2026)

Deployment & Scaling

SovraChain mainnet launch on Ethrex L2, gas sponsorship activation for all identity operations, issuer portal for government agencies, verifier app with QR-based presentations, BBS+ selective disclosure in production.

PHASE 4 — EXPANSION (Q4 2026+)

Global Scale

Cross-border credential interoperability, eIDAS 2.0 alignment validation, ISO mdoc format support for mobile driver's licenses, credential delegation activation, 10+ country deployments, private sector integration pipeline.

The End State: 1 Billion Sovereign Identities

Sovra's long-term vision is a world where every person—and every autonomous agent—has a digital identity they own and control. An identity that works everywhere, that protects privacy, and that no institution can revoke or abuse.

2026

Foundation
3+ countries, 20+ governments, eIDAS alignment, SovraChain mainnet launch

2028

Scale
10+ countries, private sector adoption, cross-border credentials, agent identity schemas, protocol revenue

2030

Infrastructure
1B+ sovereign identities, global interoperability, identity as a public utility

Tech Stack Summary

COMPONENT	TECHNOLOGY	PURPOSE
Smart Contracts	Solidity / Foundry	Wallet (EIP-7702), DID Registry, Issuer Registry, Revocation
Backend API	Elixir / Phoenix	Unified API for issuance, verification, co-signing
Mobile Wallet	React Native / Expo / TypeScript	Citizen credential storage, selective disclosure
Verifier App	React Native / Expo / TypeScript	QR-based credential verification
Issuer Portal	React / Vite / TypeScript	Government agency credential management
Shared Logic	@sovra/wallet-core (TypeScript)	Platform-agnostic identity, credentials, protocols
Blockchain	Ethrex L2 (Rust) + SP1 (ZK)	Trust anchor, DID anchoring, status lists

> *"The internet was built without an identity layer. Every security problem, every privacy scandal, every cumbersome login traces back to that missing piece. We are building it now. Not as a product, but as infrastructure—open, private, portable, and owned by the people it serves."*

Trust once. Use everywhere.

sovra.io · github.com/sovrahq · @sovraio

APPENDIX A

Technical Specifications

DID Verification Relationships

Every participant is identified by a DID resolving to a DID Document. Sovra requires separate key material per verification relationship from DID creation:

RELATIONSHIP	PURPOSE	KEY TYPE
`authentication`	Prove control of the DID (SIOPv2 login)	Ed25519 / P256
`assertionMethod`	Sign VCs (issuer) or derived proofs (holder)	Ed25519 / BBS+
`keyAgreement`	Establish encrypted channels (ECDH)	X25519
`capabilityInvocation`	Exercise capabilities (update DID Document)	Ed25519 / P256
`capabilityDelegation`	Delegate capabilities to another party (reserved)	Ed25519 / P256

Required Fields per Actor

Not every participant needs the same DID Document. The table below shows what is required, recommended, or optional for each actor type:

FIELD	HOLDER	ISSUER	VERIFIER
`verificationMethod`	**MUST** (passkey + recovery-key)	**MUST** (1+ signing keys)	**MUST** (1+ keys)
`authentication`	**MUST** → #passkey	**MUST**	**MUST**
`assertionMethod`	**MUST** → #passkey	**MUST**	SHOULD
`capabilityInvocation`	**MUST** → #passkey	SHOULD	OPTIONAL
`capabilityDelegation`	SHOULD → #passkey	OPTIONAL	OPTIONAL
`keyAgreement`	OPTIONAL	OPTIONAL	OPTIONAL
`service`	OPTIONAL	SHOULD (OID4VCI + status endpoints)	OPTIONAL

Design note: The holder's `#recovery-key` is listed in `verificationMethod` but NOT referenced by any verification relationship. It exists solely for recovery—if the passkey is lost, the recovery key can initiate a time-locked key replacement. Both keys are P-256: passkey because WebAuthn mandates it, recovery key for uniformity and EIP-7212 compatibility.

Credential Data Model

All credentials conform to W3C VC Data Model v2.0:

- `validFrom` / `validUntil` temporal bounds (replaced legacy issuanceDate/expirationDate)
- `credentialStatus` array with entries for both revocation and suspension lists
- `proofPurpose: "assertionMethod"` linking to issuer DID Document
- Base context `https://www.w3.org/ns/credentials/v2` includes BitstringStatusList vocabulary

BBS+ Signature Internals

BBS signatures over BLS12-381 elliptic curve. Single signature commits to multiple messages via generator points:

```
B = P1 + Q1·domain + H1·msg_1 + ... + HL·msg_L
```

Constant 80-byte signature regardless of message count. Derived proofs use `ProofGen` with random scalar blinding—computationally unlinkable across presentations. Proof suite: `bbs-2023`.

SD-JWT Internals

Each disclosable claim replaced with SHA-256 digest. Disclosure format: `["<salt>", "<claim-name>", "<claim-value>"]`. Serialization: `<JWT>~<Disclosure_1>~...~[<KB-JWT>]`. Holder appends only chosen disclosures. Key Binding via `cnf` claim + KB-JWT signed by holder covering presentation hash, nonce, and audience.

EIP-7702 Wallet Contract

The smart contract that EOAs delegate to via EIP-7702. Architecture: permanent thin proxy (SovraDispatcher) with ERC-1967 implementation slot + upgradeable logic (SovraWalletV1).

FUNCTION	DESCRIPTION
`initialize`	Set initial passkey + server key, create DID metadata
`execute`	Execute arbitrary call with 2-of-2 P256 signature validation
`rotatePasskey`	Replace passkey (requires current passkey + server key)
`initiateRecovery`	Begin recovery with timelock (recovery key only)
`finalizeRecovery`	Complete recovery after timelock expires
`deactivate`	Permanently deactivate the wallet and DID

TECHNICAL DEEP-DIVE: EIP-7702 WALLET CONTRACT

The wallet contract uses a proxy architecture where the EOA delegates to a permanent `SovraDispatcher` (thin proxy with ERC-1967 slot), which forwards all calls to `SovraWalletV1`.

MECHANISM	IMPLEMENTATION
Passkey Verification	P256 via EIP-7212 precompile at `0x0100`: `staticcall(hash, r, s, pubKeyX, pubKeyY)` → returns 1 if valid
Message Hash	EIP-712 structured: `keccak256(DOMAIN_SEPARATOR, operationHash, nonce)`. Domain includes chain ID, wallet address, contract version.
Nonce Management	Sequential `uint256 nonce` incremented on every state-changing operation for replay protection
Recovery Flow	`initiateRecovery()` → 7-day timelock → either key can `cancelRecovery()` → `finalizeRecovery()` replaces the target key
Gas Model	`ethrex_sendTransaction(authorization_list, to, data)` — L2 sponsors gas for 7702-whitelisted accounts. No EOA signature or ETH balance required.
Upgrade Path	UUPS: `upgrade(newImpl, sig)` updates ERC-1967 slot. Passkey-authorized. Future versions must use ERC-7201 namespaced storage.

Onboarding is atomic: generate throwaway EOA → sign 7702 authorization → `initialize(passkeyXY, recoveryKeyXY, sigs)` → discard EOA key. The EOA address becomes `did:sovra:<address>`.

DID Registry Contract

On-chain registry for issuer and verifier DIDs (holders use the wallet contract). Maps `did:sovra:<address>` to DID Document anchors.

- Stores verification methods with relationship bitmasks (P256, secp256k1, Ed25519 keys)
- Service endpoints for OpenID4VCI issuance and status list resolution
- Only the controller (passkey-authorized wallet) can modify its own DID
- Timestamped create/update/deactivate events (tamper-evident, append-only)

TECHNICAL DEEP-DIVE: DID REGISTRY SMART CONTRACT

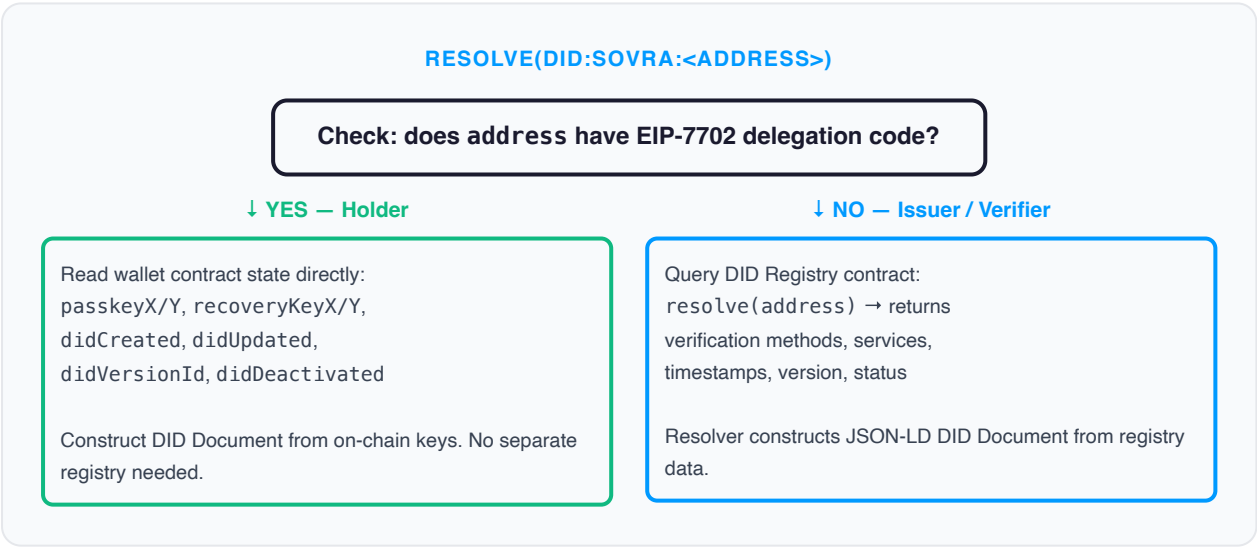
The `SovraDIDRegistry` stores issuer and verifier DID data fully on-chain. Holders do not use it — their DIDs resolve from wallet contract state.

FUNCTION	ACCESS	BEHAVIOR
`create(methods[], services[])`	`msg.sender`	Register initial keys + endpoints. Requires ≥1 verification method. Sets `versionId = 1`.
`update(methods[], services[])`	`msg.sender`	Full replacement of methods and services. Clears old data, stores new. Increments `versionId`.
`deactivate()`	`msg.sender`	Permanent. Sets `deactivated = true`. Keys remain in event history for historical verification.
`resolve(address)`	View (free)	Returns methods, services, timestamps, version, deactivation status. Resolver constructs JSON-LD DID Document.

Each `VerificationMethod` stores: `bytes32 id`, `uint8 keyType` (P256/secp256k1/Ed25519), `bytes32 pubKeyX/Y`, and a `uint8 relationships` bitmask (bit 0=authentication, 1=assertionMethod, 2=capabilityInvocation, 3=capabilityDelegation, 4=keyAgreement). No upgradeability — migration is a new contract deployment. Gas: ~150-200k for create/update, ~30k for deactivate.

Unified DID Resolver Routing

A single resolver handles all `did:sovra` identifiers. The routing decision is automatic:



This design means holders never interact with the DID Registry contract—their identity lives in their wallet contract. Issuers and verifiers use the registry because they don't have 7702-delegated wallets. The resolver auto-detects which path to take by checking if the address has delegation code, making resolution transparent to callers.

Issuer Registry Contract

On-chain registry of trusted credential issuers. Verifiers consult this registry before accepting any credential.

- Register and deregister trusted issuers with DID references and metadata hashes
- Credential type scoping: issuer X trusted for driver's licenses but not diplomas
- Governance-controlled (multisig of authorities or DAO)
- OID4VCI issuer metadata hash stored for discovery

TECHNICAL DEEP-DIVE: ISSUER REGISTRY GOVERNANCE

The `SovraIssuerRegistry` answers one question on-chain: "Is this DID trusted to issue this credential type?"

FUNCTION	ACCESS	PURPOSE
`registerIssuer(addr, typeIds[])`	Owner only	Add issuer as trusted for specific credential types
`addCredentialTypes(addr, typeIds[])`	Owner only	Expand an existing issuer's trust scope
`removeCredentialTypes(addr, typeIds[])`	Owner only	Revoke trust for specific types
`removeIssuer(addr)`	Owner only	Fully de-register issuer (immediate, retroactive)
`isTrusted(addr, typeId)`	View (free)	Core query: `registered && trustedFor[typeId]`

Credential types are `bytes32` hashes: `keccak256("DrivingLicenceCredential")`. No on-chain schema — types are conventions agreed by issuers and verifiers. The `owner` address starts as a single Sovra admin and can be transferred to a multisig or DAO via `transferOwnership()` without code changes. Removing an issuer is immediate and retroactive — verifiers checking the registry will reject existing credentials.

Revocation Registry Contract

Per-issuer revocation bitmaps implementing W3C Bitstring Status List v1.0 on-chain.

- Each credential assigned a randomly chosen index in a GZIP-compressed bitstring
- Bit 0 = valid, bit 1 = revoked/suspended. Separate lists for permanent revocation and reversible suspension
- Batch revocation support for key epoch rotation
- Multi-bit support (`statusSize` 2/4/8) for richer status codes
- Published as DID-Linked Resource, versioned and tamper-evident

Co-signing Server Specification

Off-chain service holding the second key in the 2-of-2 multisig wallet model.

ASPECT	DETAIL
Key Storage	HSM-backed P256 key material (FIPS 140-2/3 in production)
Authentication	OAuth/SSO before any recovery signing operation
Operations	KB-JWT co-signature for presentations, recovery transaction signing (key rotation only)
Safeguards	Rate limiting, fraud detection, timelock on recovery, cannot initiate regular wallet operations

TECHNICAL DEEP-DIVE: CO-SIGNING / RECOVERY SERVER PROTOCOL

The recovery server holds one P256 key per holder in HSM and participates only in onboarding initialization and key recovery — never in normal credential presentations or transactions.

ENDPOINT	WHEN USED	WHAT IT DOES
`POST /onboard`	Account creation	Verifies Google OAuth token, generates P256 recovery keypair in HSM, returns public key
`POST /cosign`	Onboarding + recovery	Validates 15-min session JWT, signs the provided hash with holder's recovery key, returns `{r, s}`
`POST /initiate-recovery`	Lost passkey	May require fresh OAuth. Signs `initiateRecovery` hash. 7-day cancel window then applies on-chain.
`POST /rotate-recovery-key`	Key hygiene	Generates new P256 key, signs rotation hash with old key, returns new public key + signature

Identity binding: Google `sub` claim is stored at onboarding and verified on every subsequent request. Key isolation: each holder's key is independent in HSM — no master key.

What the server CANNOT do — these flows operate without any server involvement:

CREDENTIAL PRESENTATION

Wallet → Verifier (direct HTTP). Passkey signs KB-JWT alone. Server never sees what credentials are presented, to whom, or when.

ON-CHAIN TRANSACTIONS

DID updates, key rotations, deactivations — all require passkey signature only. Server has no signing role in normal on-chain operations.

DID RESOLUTION

Verifiers resolve DIDs from on-chain state (wallet contract or DID Registry). Server is not in the resolution path — it cannot intercept or modify DID lookups.

If the server is compromised, attackers can only initiate time-locked recoveries (7-day window for holders to cancel with passkey). This minimal attack surface is by design — the server is a **recovery-only guardian**, not a gatekeeper.

APPENDIX B

DID Method & Exchange Protocols

The `did:sovra` Method

The `did:sovra` method resolves identifiers anchored on SovraChain. Resolution rules:

- **Format:** `did:sovra:<ethereum-address>`
- **Resolution:** Query DID Registry contract on SovraChain, retrieve current DID Document hash, fetch full document from linked storage
- **Holder DIDs:** Resolved from wallet contract state (passkey public key, DID metadata)
- **Issuer DIDs:** Resolved from DID Registry contract (separate key material per relationship, service endpoints)
- **Deactivation:** Controller sets deactivated flag; DID Document still resolvable but marked inactive

OpenID4VCI: Issuance Protocol

Pre-Authorized Code Flow (primary issuance pattern):

STEP	ACTION	DETAIL
1	Credential Offer	Issuer sends QR/deep link with `credential_issuer`, `credential_configuration_ids`, `pre-authorized_code`, optional `tx_code`
2	Metadata Discovery	Wallet fetches `/.well-known/openid-credential-issuer`
3	Token Request	Wallet POSTs `pre-authorized_code` + `tx_code` to `/token`
4	Credential Request	Wallet POSTs `credential_configuration_id` + JWT proof (with `c_nonce`, holder public key) to `/credential`
5	Credential Response	Signed VC/SD-JWT/mdoc. For deferred: `transaction_id` + poll interval

OpenID4VP: Verification Protocol

Cross-Device Flow (primary verification pattern):

STEP	ACTION	DETAIL
1	QR Code	Verifier displays `client_id` + `request_uri`
2	Request Object	Wallet fetches signed request with `dcql_query` (credential type + required claims), `nonce`, `response_uri`
3	Citizen Approval	Citizen reviews and approves which fields to share via selective disclosure
4	VP Response	Wallet POSTs `vp_token` (selected claims + KB-JWT/BBS+ proof/DeviceAuth) to `response_uri`
5	Validation	Verify VP signature, verify credential signature (from DID Document), check nonce, check DCQL satisfaction, fetch status list

TECHNICAL DEEP-DIVE: VERIFICATION SDK INTERNALS

The Verification SDK is a pure TypeScript library that performs a 12-step validation pipeline on every SD-JWT presentation string.

STEP	CHECK	ON FAILURE
1-2	Parse SD-JWT → extract issuer JWT, disclosures, KB-JWT	`PARSE_ERROR`
3-4	Resolve issuer DID → verify ES256 signature with `assertionMethod` key	`ISSUER_SIG_INVALID`
5	Query `IssuerRegistry.isTrusted(issuer, keccak256(vct))`	`ISSUER_NOT_TRUSTED`
6-8	Verify disclosure hashes, check `exp`, check Bitstring Status List	`CREDENTIAL_EXPIRED` / `CREDENTIAL_REVOKED`
9-11	Resolve holder DID → verify 64-byte ES256 KB-JWT passkey signature → validate `aud`, `nonce`, `sd_hash`	`KB_PASSKEY_SIG_INVALID`

DID resolution is cached in-memory (holder keys by address, issuer trust with 5-min TTL). The SDK auto-detects holder vs issuer by checking for 7702 delegation code. A relay server bridges the cross-device flow: Verify App uploads signed request, wallet fetches it, posts `vp_token` back. Sessions auto-expire after 5 minutes.

SIOPv2: Passwordless Authentication

Beyond credential presentation, Sovra enables **passwordless login** to any service using Self-Issued OpenID Provider v2 (SIOPv2). The citizen becomes their own identity provider—no centralized auth server, no shared secrets, no passwords to breach.

What makes this different: No username/password database to breach. No OAuth provider to go down. No session tokens stored server-side. The citizen proves they control their DID—nothing more. The relying party never learns anything beyond "this DID authenticated." SIOPv2 can be combined with OpenID4VP to simultaneously authenticate *and* present specific credentials in one flow.

Trust Assumptions & Threat Model

Trust Assumptions

- **Ledger consensus:** Liveness + safety. No adversary rewrites finalized history
- **Device secure enclave:** Keys do not leave TEE boundary in plaintext
- **Issuer identity proofing:** Crypto verifies authenticity, not the underlying enrollment rigor
- **Holder device integrity:** Unrooted, unmodified wallet software
- **DID resolution:** Returns authoritative DID Document, verified against ledger state

Adversary Model

ADVERSARY	MITIGATION
Malicious Verifier (extract undisclosed claims, link sessions)	Selective disclosure; nonce+audience binding; pairwise holder DIDs
Malicious Holder (present stolen credential, forge claims)	Holder binding (TEE key); issuer signature verification; device attestation
Compromised Issuer (fraudulent credentials)	Key rotation + ledger anchoring; revoke key epoch; short-lived credentials; multi-sig issuance
Compromised Device (extract keys)	TEE isolation; biometric binding; remote revocation
Network Attacker (MITM, replay)	TLS 1.3; ledger-anchored DIDs; nonce+audience binding
Colluding Parties (issuer+verifier tracking)	BBS+ unlinkability; pairwise DIDs; credential ID not disclosed
Ledger Attacker (rewrite DID history)	BFT consensus; monotonic sequence numbers; cross-chain anchoring

Residual Risks

- Verifier data retention post-presentation: mitigated by governance (GDPR minimization)
- Compromise-to-rotation detection lag: mitigated by operational SLAs
- Offline verifiers accept revoked credentials within cache TTL: mitigated by policy limits
- Highly specific disclosed attributes may re-identify subject: mitigated by attribute coarsening and predicate proofs

Credential Data Model

All credentials conform to W3C VC Data Model v2.0 — the global standard for digital credential structure.

TECHNICAL DEEP-DIVE: W3C VC DATA MODEL V2.0

Key properties: `validFrom` / `validUntil` (replaced `issuanceDate` / `expirationDate`), `credentialStatus` array with entries for both revocation and suspension, `proofPurpose: "assertionMethod"` linking to issuer DID Document. Base context `https://www.w3.org/ns/credentials/v2` includes BitstringStatusList vocabulary.

FORMAT	ENCODING	SIGNATURE	SELECTIVE DISCLOSURE	PRIMARY USE
W3C VC (Data Integrity)	JSON-LD	Ed25519 / BBS+	BBS+ derived proofs (ZKP)	General-purpose, privacy-maximizing
SD-JWT VC	JWT (compact)	ES256, Ed25519	Salted hash digests	OAuth/OpenID-native ecosystems
ISO mdoc	CBOR (binary)	ECDSA (P-256) COSE_Sign1	Salted hash commitments in MSO	Proximity, offline, mDL

Format is a deployment decision. SovraWallet stores and presents all three formats. OpenID4VP carries all through the same flow.

Forward Compatibility: Credential Delegation

Holders sometimes need to delegate: a citizen to an attorney, a parent to a guardian, a ministry to a sub-agency. Sovra is built for this from day one.

TECHNICAL DEEP-DIVE: DELEGATION ARCHITECTURE

Sovra registers separate `capabilityInvocation` and `capabilityDelegation` key material in all DID Documents from creation. Delegation can be **activated via DID Document update** without re-issuing credentials or re-registering DIDs.

Attenuation: A delegate can never exceed the delegator's authority. Each delegation step may add restrictions (credential type, time window, action scope). Enforced cryptographically, verifiable by any party.

Separation of concerns: `capabilityInvocation` keys invoke on-chain operations. `capabilityDelegation` keys delegate authority to third parties. Verification keys prove identity. No key type can be used for another purpose.

Privacy Analysis

Identity infrastructure must protect the people it serves. Sovra addresses four critical privacy threats:

TECHNICAL DEEP-DIVE: PRIVACY THREAT MITIGATIONS

THREAT	MITIGATION
Persistent DID correlation	Pairwise DIDs per verifier and ephemeral session-scoped DIDs. `did:key` requires no ledger resolution.
Issuer tracking via status checks	131,072+ entry herd privacy, verifier caching, CDN distribution, holder stapling, on-chain anchoring.
Verifier session linkage	BBS+ proof unlinkability, ephemeral DIDs, attribute coarsening policies.
Metadata leakage	Random index assignment (no issuance order leak), binary status only (no circumstance leak), TLS 1.3 (encrypted paths).

Non-Human Subjects: Agent Identity

The Sovra architecture makes no assumption that the subject of a credential is a natural person. A DID can identify any entity—a citizen, an organization, a device, or an autonomous AI agent. The credential data model (W3C VC Data Model v2.0) supports arbitrary subject types through its `credentialSubject` field.

As autonomous AI agents proliferate across government and enterprise workflows, a new trust requirement emerges: **verifiable agent identity.**

TECHNICAL DEEP-DIVE: AGENT IDENTITY PRIMITIVES

REQUIREMENT	MECHANISM
Deployer attestation	A Verifiable Credential issued by the deploying organization, bound to the agent's DID, proves provenance.
Permission scope	A credential encoding permission boundaries (credential types, action scopes, time windows)—structurally identical to the delegation model above.
Model integrity	An attestation signed by the deployer (or auditor) over the agent's configuration hash provides cryptographic proof of integrity.
Selective disclosure	A verifier can confirm an agent has a specific permission without learning its full configuration, using the same BBS+ and SD-JWT mechanisms described above.

No new cryptographic primitive is required. The DID layer, credential layer, selective disclosure layer, and exchange protocols apply directly. The credential schemas change—the infrastructure does not.

Current status: Architecturally ready, not yet deployed. Agent identity credential schemas will be defined when regulatory requirements (EU AI Act agent attestation mandates) and market demand mature. Sovra's standards-aligned, technology-abstract design ensures forward compatibility without re-architecture.

GLOSSARY

Key Terms

BBS+ Signatures	A special type of digital signature that lets you reveal only parts of a document while keeping the rest hidden. This is what makes selective disclosure possible.
Based Rollup	A Layer 2 blockchain that uses Ethereum's own validators to order transactions, maximizing decentralization. SovraChain uses this approach.
DID	Decentralized Identifier. A digital identity that you own and control—like a username, but no company or government can take it away.
DPI	Digital Public Infrastructure. Foundational digital systems—identity, payments, data exchange—that serve as public utilities.
DPG	Digital Public Good. Open-source software designed for public benefit, free to use and free from vendor lock-in.
eIDAS 2.0	EU regulation mandating digital identity wallets for all EU citizens by 2026.
EIP-7702	Ethereum standard enabling account delegation, allowing SovraChain to sponsor gas fees so citizens never need cryptocurrency tokens.
EIP-7212	Ethereum precompile for P256 (secp256r1) signature verification, enabling cheap passkey validation on-chain.
Ethrex	A modern, high-performance Ethereum execution engine built in Rust by LambdaClass. Powers SovraChain.
KYC	Know Your Customer. The identity verification process required by financial institutions. Sovra makes it reusable.
L2 (Layer 2)	A system built on top of a blockchain (Layer 1) to make it faster and cheaper while inheriting its security.
OpenID4VC	Protocols for issuing and verifying credentials over the internet, built on the same framework as "Login with Google."
Selective Disclosure	The ability to share only specific parts of a credential. Example: prove you are over 21 without revealing your actual birthdate.
SSI	Self-Sovereign Identity. The principle that individuals should own and control their own digital identity.
Trust Triangle	The three-party model: an Issuer creates a credential, a Holder keeps it, a Verifier checks it. No central authority needed.
Validium	A Layer 2 approach where data stays off-chain (private) but proofs go on-chain (verifiable). Ideal for identity.
VC	Verifiable Credential. A digital document with a cryptographic signature that makes it impossible to forge and instant to verify.
ZKP	Zero-Knowledge Proof. A way to prove something is true without revealing any of the underlying information.

REFERENCES

Specifications & Further Reading

Core Specifications

SPECIFICATION	URL
W3C DID Core v1.1	`w3.org/TR/did-core`
W3C Controlled Identifiers v1.0	`w3.org/TR/cid-1.0`
W3C VC Data Model v2.0	`w3.org/TR/vc-data-model`
W3C Bitstring Status List v1.0	`w3.org/TR/vc-bitstring-status-list`
W3C Data Integrity BBS Cryptosuites v1.0	`w3.org/TR/vc-di-bbs`
W3C Data Integrity EdDSA Cryptosuites	`w3.org/TR/vc-di-eddsa`
SD-JWT (RFC 9901)	`datatracker.ietf.org/doc/rfc9901`
OpenID4VCI 1.0	`openid.net/specs/openid-4-verifiable-credential-issuance-1_0`
OpenID4VP 1.0	`openid.net/specs/openid-4-verifiable-presentations-1_0`
SIOPv2	`openid.net/specs/openid-connect-self-issued-v2-1_0`
ISO/IEC 18013-5:2021 (mDL)	`iso.org/standard/69084.html`
ISO/IEC 18013-7:2024	`iso.org/standard/82772.html`
ZCAP-LD v0.3 (Delegation)	`w3c-ccg.github.io/zcap-spec`

Further Reading

- **Self-Sovereign Identity** by Alex Preukschat & Drummond Reed (Manning Publications)—the definitive guide to decentralized digital identity.
- **eIDAS 2.0 Regulation**—the EU framework mandating digital identity wallets for 450M+ citizens by 2026.
- **Digital Public Goods Alliance** (digitalpublicgoods.net)—the certification body for open-source public benefit software.
- **Sovra Knowledge Base** (sovra.io/knowledge)—educational resources on digital identity, verifiable credentials, and privacy-preserving technology.

About Sovra

Sovra was founded by leaders spanning government, cryptography, business, and finance. Together, they built Latin America's most adopted reusable identity platform and led the region's largest production-level digital identity deployments. The founding principle: technology should make trust easier, not harder. Identity is dignity—and the infrastructure that protects it should be built by people who understand both the technology and the institutions it serves.

SOVRA

The identity layer the internet never had. Verifiable. Private. Portable.

sovra.io · github.com/sovrahq · @sovraio · sovra.substack.com