Authentic Chained Data Container
ACDC

Verifiable Data Structures (Graphs)
that support
Provenanced Authenticatable
Attestations & Entitlements

Samuel M. Smith Ph.D.
sam@prosapien.com
2022/09/15

Resources

Specification Documentation:

ACDC Internet Draft (ToIP - IETF)
https://github.com/trustoverip/tswg-acdc-specification

ACDC for Muggles
https://docs.google.com/presentation/d/1mO1EZa9BcjAjWEzw7DWi124uMfyNyDeM3HuajsGNoTo/edit#slide=id.ga411be7e84_0_0

Resources on KERI and ACDC
https://keri.one/keri-resources/

GLEIF vLEI Credentials (Global Legal Entity Identifier Foundation) ISO LEI specification
https://www.gleif.org/en/vlei/introducing-the-verifiable-lei-vlei

Community: (meetings, open source code, IETF internet drafts)

ACDC Working Group
https://wiki.trustoverip.org/display/HOME/ACDC+%28Authentic+Chained+Data+Container%29+Task+Force

Related Internet Drafts
https://github.com/WebOfTrust/keri

Important ACDC Features

ACDC is based on KERI so one gets all the features of KERI for free.

Leverage SAIDs (Self-Addressing Identifiers) and AIDs (Autonomic Identifiers)
    Based on white magic (dumb) crypto: digests, and digital signatures.
Leverages CESR (Composable Event Streaming Representation) to resolve the text vs binary tension.
JSON Schema (Type-is-schema, schema are immutable)
Chaining (labeled property graph fragment model)
Graduated Disclosure (compact, partial, private, selective)
Contractually Protected Disclosure (chain-link confidentiality, contingent disclosure)
    Protection against data exploitation from both statistical correlation and cryptographic correlation
Decentralized extensibility model
Zero-trust End-Verifiable

Basic ACDCs

Private Compact Variant JSON. (even more compact CBOR, MGPK, CESR also supported)


```
{
  "v":  "ACDC10JSON00011c_",
  "d":  "EAdXt3gIXOf2BBWNHdSXCJnFJL5OuQPyM5K0neuniccM",
  "u":  "0ABghkDaG7OY1wjaDAE0qHcg",
  "i":  "did:keri:EBkPreYpZfFk66jpf3uFv7vklXKhzBrAqjsKAn2EDIPM",
  "ri": "did:keri:ECmRy7xMwsxUelUauaXtMxTfPAMPAI6FkekwlOjkggt",
  "s":  "ED6jrVPTzlSkUPqGGeIZ8a8FWS7a6s4reAXRZOkogZ2A",
  "a":  "EEveY4-9XgOcLxUderzwLIr9Bf7V_NHwY1lkFrn9y2PY",
  "e":  "EFH3dCdoFOLe71iheqcywJcnjtJtQIYPvAu6DZIl3MOA",
  "r":  "EG71iheqcywJcnjtJtQIYPvAu6DZIl3MORH3dCdoFOLB"
}
```

Basic ACDC JSON Schema

Private Compact Variant

```
{
  "v":  "ACDC10JSON00011c_",
  "d":  "EAdXt3gIXOf2BBWNHdSXCJnFJL5OuQPyM5K0neuniccM",
  "u":  "0ABghkDaG7OY1wjaDAE0qHcg",
  "i":  "did:keri:EBkPreYpZfFk66jpf3uFv7vklXKhzBrAqjsKAn2EDIPM",
  "ri": "did:keri:ECmRy7xMwsxUelUauaXtMxTfPAMPAI6FkekwlOjkggt",
  "s":  "ED6jrVPTzlSkUPqGGeIZ8a8FWS7a6s4reAXRZOkogZ2A",
  "a":  "EEveY4-9XgOcLxUderzwLIr9Bf7V_NHwY1lkFrn9y2PY",
  "e":  "EFH3dCdoFOLe71iheqcywJcnjtJtQIYPvAu6DZIl3MOA",
  "r":  "EG71iheqcywJcnjtJtQIYPvAu6DZIl3MORH3dCdoFOLB"
}
```

Non-Composed JSON Schema

```
{
  "$id": "EBdXt3gIXOf2BBWNHdSXCJnFJL5OuQPyM5K0neuniccM",
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "title": "Compact Private ACDC",
  "description": "Example JSON Schema for a Compact Private ACDC.",
  "credentialType": "CompactPrivateACDCExample",
  "type": "object",
  "required":
  [
    "v",
    "d",
    "u",
    "i",
    "ri",
    "s",
    "a",
    "e",
    "r"
  ],
```

Non-Composed JSON Schema Continued

```
 "properties":
 {
   "v":
   {
     "description": "ACDC version string",
     "type": "string"
   },
   "d":
   {
    "description": "ACDC SAID",
     "type": "string"
   },
   "u":
   {
    "description": "ACDC UUID",
     "type": "string"
   },
   "i":
   {
     "description": "Issuer AID",
     "type": "string"
   },
   "ri":
   {
     "description": "credential status registry ID",
     "type": "string"
   },
   "s": {
     "description": "schema SAID",
     "type": "string"
   },
   "a": {
     "description": "attribute SAID",
     "type": "string"
   },
   "e": {
     "description": "edge SAID",
     "type": "string"
   },
   "r": {
     "description": "rule SAID",
     "type": "string"
   }
  },
  "additionalProperties": false
}
```

Uncompacted Private Attribute Section

Private Compact Variant


```
{
  "v":  "ACDC10JSON00011c_",
  "d":  "EAdXt3gIXOf2BBWNHdSXCJnFJL5OuQPyM5K0neuniccM",
  "u":  "0ABghkDaG7OY1wjaDAE0qHcg",
  "i":  "did:keri:EBkPreYpZfFk66jpf3uFv7vklXKhzBrAqjsKAn2EDIPM",
  "ri": "did:keri:ECmRy7xMwsxUelUauaXtMxTfPAMPAI6FkekwlOjkggt",
  "s":  "ED6jrVPTzlSkUPqGGeIZ8a8FWS7a6s4reAXRZOkogZ2A",
  "a":  "EEveY4-9XgOcLxUderzwLIr9Bf7V_NHwY1lkFrn9y2PY",
  "e":  "EFH3dCdoFOLe71iheqcywJcnjtJtQIYPvAu6DZIl3MOA",
  "r":  "EG71iheqcywJcnjtJtQIYPvAu6DZIl3MORH3dCdoFOLB"
}
```


Uncompacted Attribute Section from Above


```
{
  "a":
  {
    "d": "EEveY4-9XgOcLxUderzwLIr9Bf7V_NHwY1lkFrn9y2PY",
    "u": "0AwjaDAE0qHcgNghkDaG7OY1",
    "i": "did:keri:EpZfFk66jpf3uFv7vklXKhzBrAqjsKAn2EDIPmkPreYA",
    "score": 96,
    "name": "Jane Doe"
  }
}
```


Attribute Section as Composed JSON Schema


```
{
  "a":
  {
    "description": "attribute section",
    "oneOf":
    [
      {
        "description": "attribute SAID",
        "type": "string"
      },
      {
        "description": "uncompacted attribute section",
        "type": "object",
        "required":
        [
          "d",
          "u",
          "i",
          "score",
          "name"
        ],
        "properties":
        {
          "d":
          {
            "description": "attribute SAID",
            "type": "string"
          },
          "u":
          {
            "description": "attribute UUID",
            "type": "string"
          },
          "i":
          {
            "description": "Issuee AID",
            "type": "string"
          },
          "score":
          {
            "description": "test score",
            "type": "integer"
          },
          "name":
          {
            "description": "test taker full name",
            "type": "string"
          }
        },
        "additionalProperties": false,
      }
    ]
  }
}
```

Edge Section

```
{
  "e":
  {
    "d": "EBrzwLIr9Bf7V_NHwY1lkFrn9y2PgveY4-9XgOcLxUdY",
    "boss":
    {
      "d": "EFy2PgveY4-9XgOcLxUdYerzwLIr9Bf7V_NHwY1lkFrn",
      "n": “EIl3MORH3dCdoFOLe71iheqcywJcnjtJtQIYPvAu6DZA”,
      "s": “EFOLe71iheqcywJcnjtJtQIYPvAu6DZAIl3MORH3dCdo”,
    }
  }
}
```

Nested Edge Section with Operators


```
{
  "e":
  {
    "d": "EBrzwLIr9Bf7V_NHwY1lkFrn9y2PgveY4-9XgOcLx,UdY",
    "o": "AND",
    "boss":
    {
      "n": "EGl3MORH3dCdoFOLe71iheqcywJcnjtJtQIYPvAu6DZA",
      "o": ["NI2I", "NOT"]
    },
    "baby":
    {
      "n": "EMRH3dCdoFOLe71iheqcywJcnjtJtQIYPvAu6DZAIl3A",
      "o": "I2I"
    },
    "food":
    {
      "o": "OR",
      "plum":
      {
        "n": "EHIYPvAu6DZAIl3AORH3dCdoFOLe71iheqcywJcnjtJt",
        "o": "NI2I"
      },
      "pear":
      {
        "n": "ECtQIYPvAu6DZAIl3AORH3dCdoFOLe71iheqcywJcnjt",
        "o": "NI2I"
      }
    }
  }
}
```

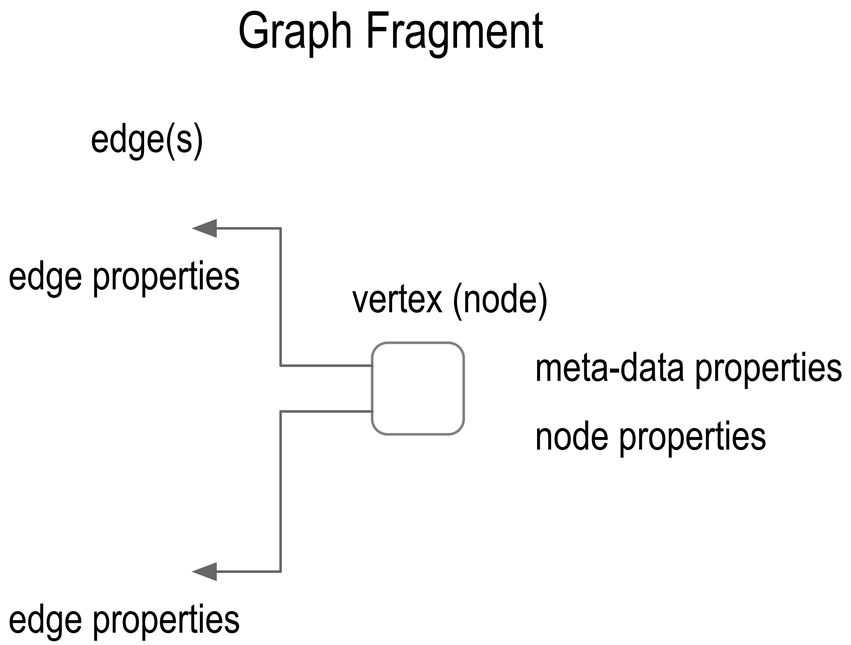
ACDC Normative Field Labels

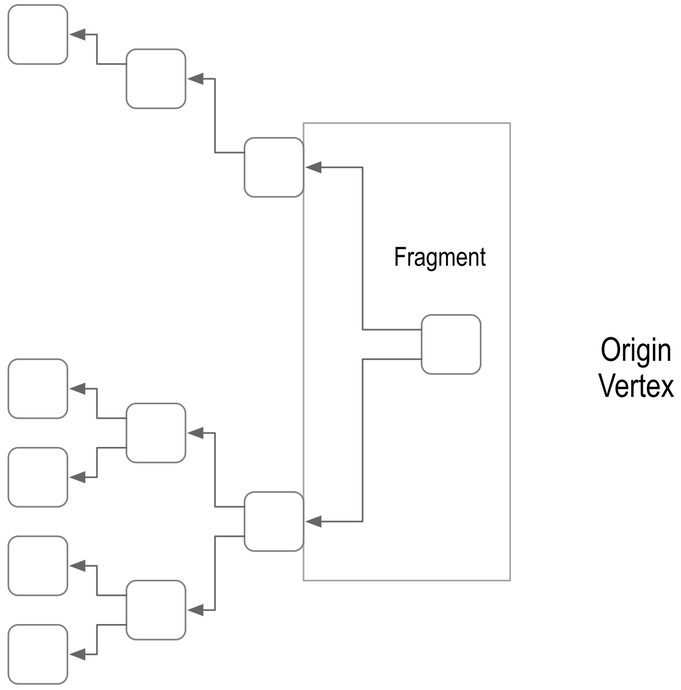
ACDC Field Labels

Label	Title	Description
		Top Level Fields
`v`	Version String	Regex-able format: `"ACDCvvSSSShhhhhh_"` that provides protocol type, protocol version, serialization type, serialized size, and terminator.
`d`	Digest (SAID)	Self-Addressing IDentifier. Self-referential fully qualified (agile) cryptographic digest of enclosing map in CESR format: `"EBdXt3gIXOf2BBWNHdSXCJnFJL5OuQPyM5K0neuniccM"`
`i`	Issuer Identifier (AID)	Autonomic IDentifier whose control authority is established via KERI verifiable key state in CESR format: `"EAkPreYpZfFk66jpf3uFv7vklXKhzBrAqjsKAn2EDIPM"`
`u`	UUID	Random Universally Unique IDentifier as fully qualified high entropy pseudo-random string, a salty nonce. Protection from rainbow table attack on private variants in CESR format: `"0ABghkDaG7OY1wjaDAE0qHcg"`
`ri`	Registry Identifier	Issuance and/or revocation, transfer, or retraction registry identifier, cryptographically derived from Issuer Identifier (AID-ish) in CESR format: `"ECmRy7xMwsxUelUauaXtMxTfPAMPAI6FkekwlOjkggt"`
`s`	Schema	Either the SAID in CESR format block in CESR format or the block itself: `"ED6jrVPTzlSkUPqGGeIZ8a8FWS7a6s4reAXRZOkogZ2A"`
`a`	Attribute	Either the SAID of a block of attributes in CESR format or the block itself in CESR format: `"EEveY4-9XgOcLxUderzwLIr9Bf7V_NHwY1lkFrn9y2PY"`
`A`	Attribute Aggregate	Either the Aggregate of a selectively disclosable block of attributes in CESR format or the block itself in CESR format: `"EHveY4-9XgOcLxUderzwLIr9Bf7V_NHwY1lkFrn9y2PY"`
`e`	Edge	Either the SAID of a block in CESR format of edges or the block itself in CESR format:
`r`	Rule	Either the SAID a block of rules in CESR format or the block itself.
		Other Fields
`d`	Digest (SAID)	Self-Addressing IDentifier. Self-referential fully qualified (agile) cryptographic digest of enclosing map in CESR format: `"EBdXt3gIXOf2BBWNHdSXCJnFJL5OuQPyM5K0neuniccM"`
`i`	Identifier (AID)	Autonomic IDentifier context dependent whose control authority is established via KERI verifiable key state in CESR format, such as, Issuee Identifier,: `"EAkPreYpZfFk66jpf3uFv7vklXKhzBrAqjsKAn2EDIPM"`
`u`	UUID	Random Universally Unique IDentifier as fully qualified high entropy pseudo-random string, a salty nonce. Protection from rainbow table attack on private variants in CESR format: `"0ABghkDaG7OY1wjaDAE0qHcg"`
`n`	Node	SAID of another ACDC in CESR format as the terminating point (vertex) of a directed edge that connects the encapsulating ACDC node to the specified ACDC as a distributed property graph (PG) fragment:
`o`	Operator	Either unary operator on edge or m-ary operator on edge-group in edge section. Enables expressing of edge logic on edge subgraph.
`w`	Weight	Edge weight property that enables default property for directed weighted edges and operators on directed weighted edges.
`l`	Legal Language	Text of Ricardian contract clause.

Big Picture: What is an ACDC?

- Decentralizable distributed verifiable data structure that is structurally constrained by immutable but composable JSON Schema.
- Each ACDC is universally uniquely referenced by its SAID.
- Authenticatable decentralizable distributed graph fragment that may be communicated securely.
- Graph fragments use SAIDs to hash-chain themselves together without any expansion needed.
- The composition of graph fragments is an authenticatable verifiable graph data structure, i.e. a chained set of ACDCs (zero-trust end-verifiable security model)
- Verifiable data structures all the way down





ACDC is a Hash Tree

The different variants of an ACDC form a hash tree (using SAIDs) that is analogous to a Merkle Tree in that the root hash provides an integrity proof of the tree.

Signing the top-level SAID of the compact version of the ACDC is equivalent to signing the root hash.

Different variants of an ACDC (SADs with SAIDs) correspond to different branches of the hash tree.

The process of verifying that a SAD via its SAID of a section is included in a schema authorized variant down from the top-level SAID is a type of proof of inclusion.

This allows a single signature to provide proof of Issuance of the presentation of any schema authorized variants of the ACDC.

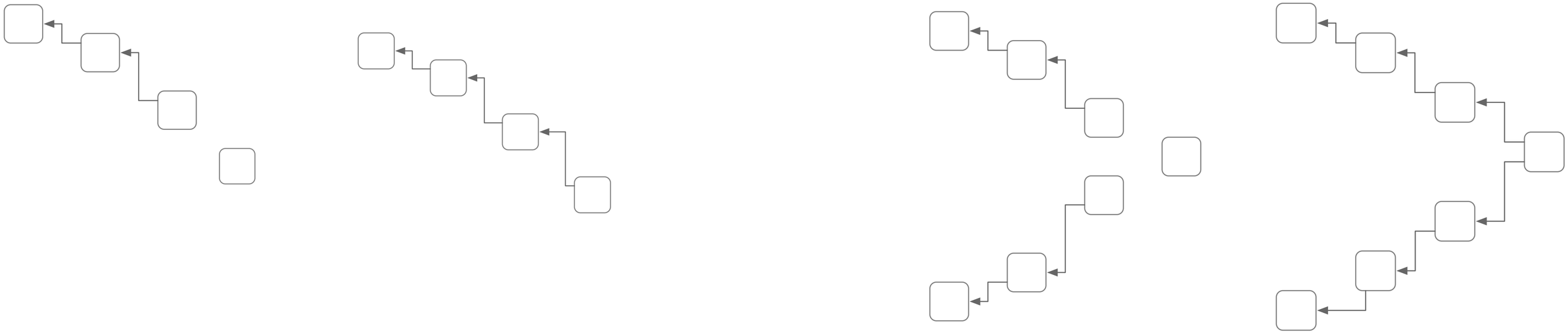
Chained ACDCs (graphs) Enable

Provenanced chains-of-custody of decentralizable authenticatable data attestations

Traceable data for data supply chains

Provenanced chains-of-authority for decentralizable authenticatable credentials

Verifiable delegated entitlements or authorizations

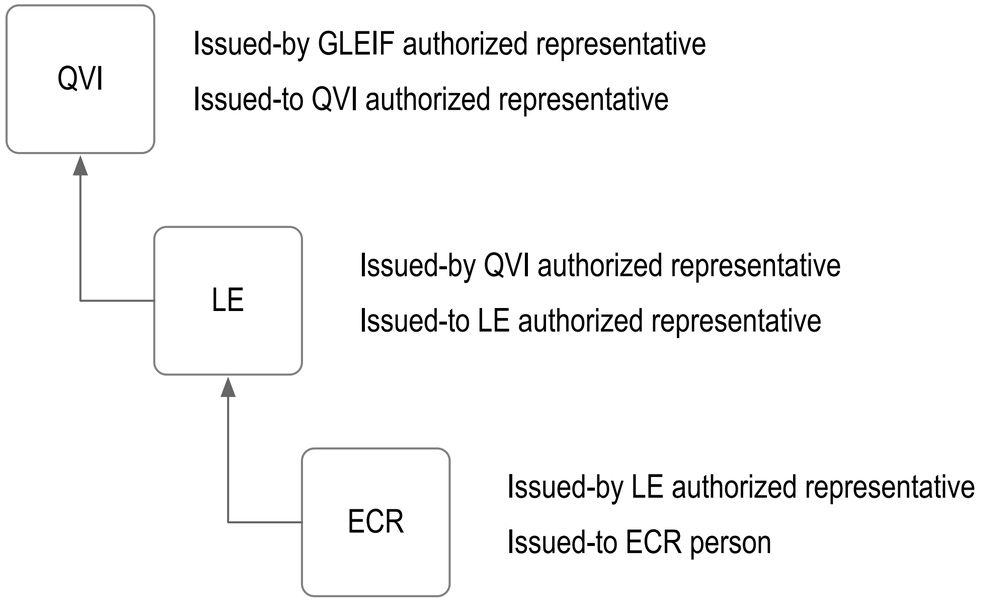


GLEIF vLEI Credential Example

Qualified vLEI Issuer (QVI) Credential

Legal Entity (LE) Credential

Engagement Context Role (ECR) Credential

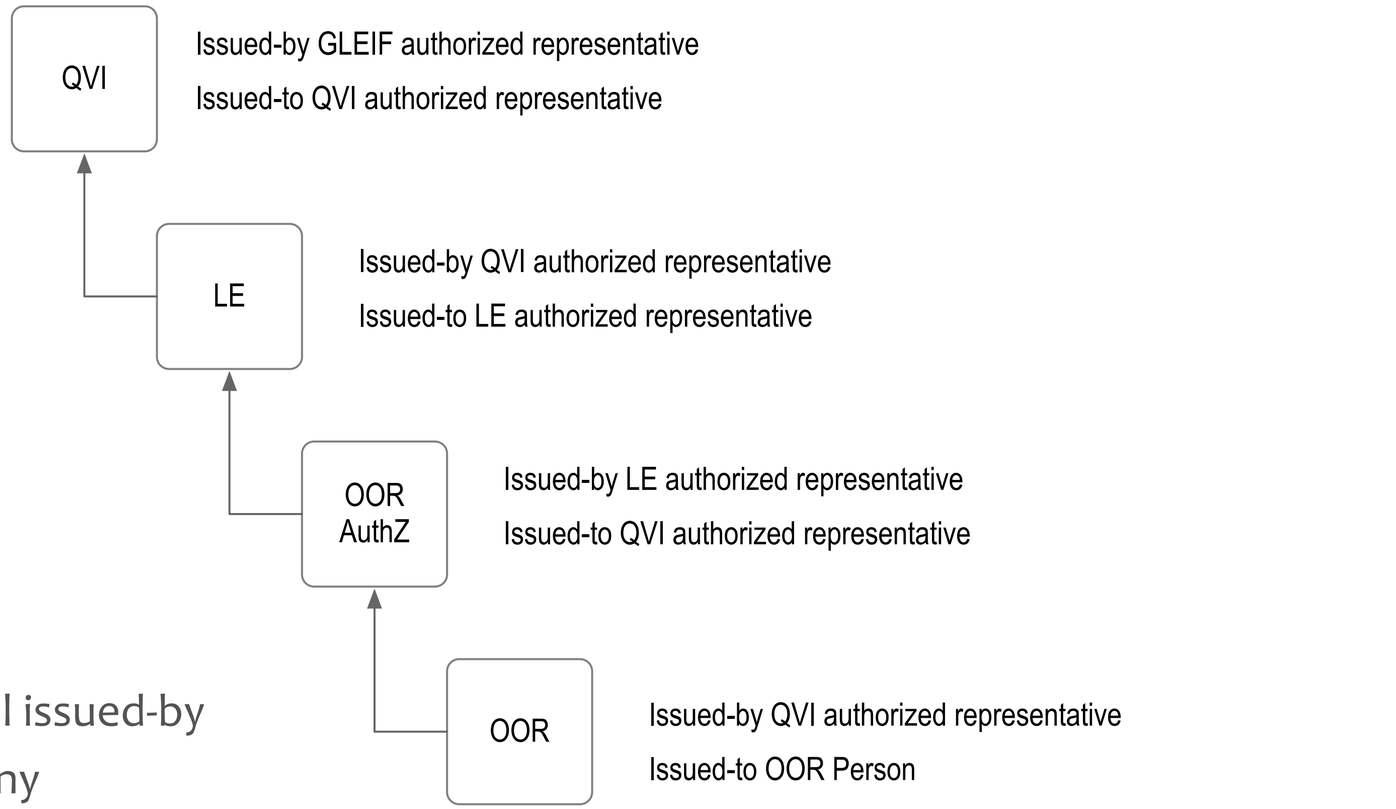


Qualified vLEI Issuer (QVI) Credential

Legal Entity (LE) Credential

Official Organizational Role Authorization (OOR-AuthZ) Credential

Official Organizational Role (OOR) Credential



Anyone in the chain-of-authority can revoke the credential issued-by them. This breaks the chain and thereby may invalidate any authorizations or attestations that are chained from their credential.

GLEIF vLEI Authorized Attestation Example

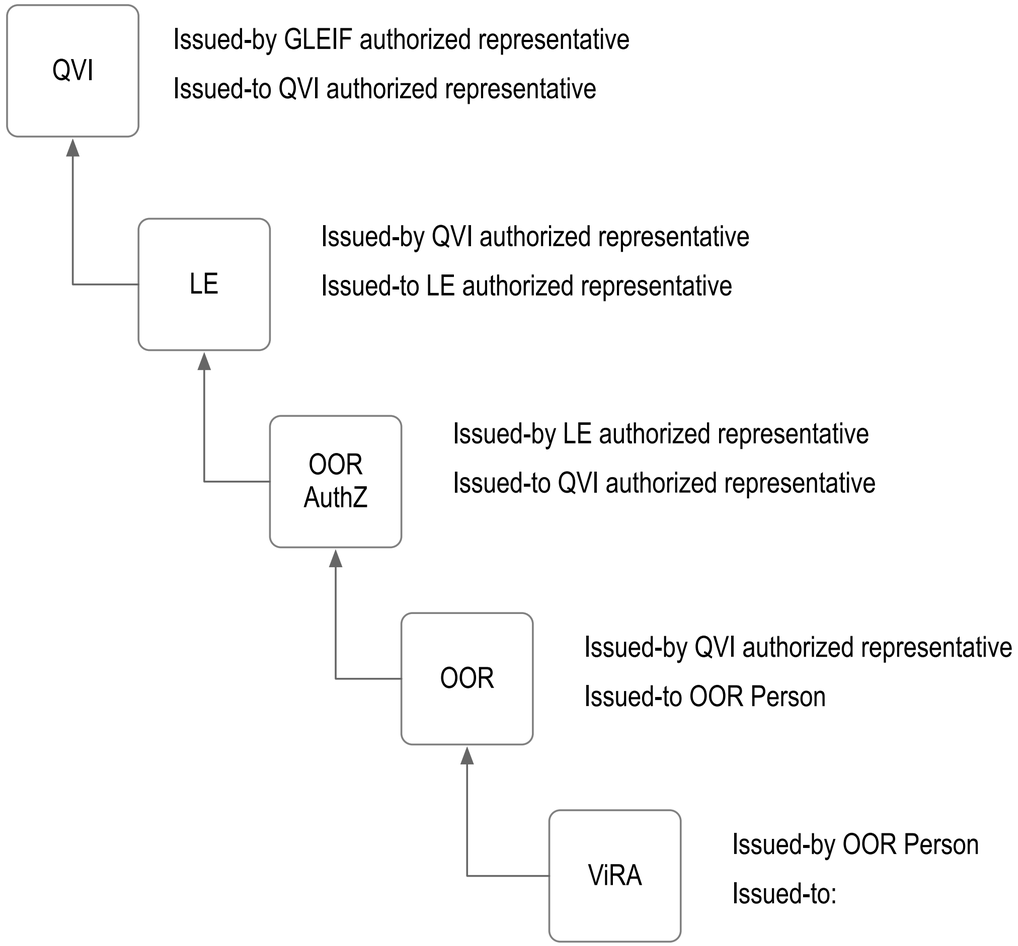
Qualified vLEI Issuer (QVI) Credential

Legal Entity (LE) Credential

Official Organizational Role Authorization (OOR-AuthZ) Credential

Official Organizational Role (OOR) Credential

Verifiable IXBRL Report Attestation (ViRA)



Anyone in the chain-of-authority can revoke the credential issued-by them. This breaks the chain and thereby may invalidate any authorizations or attestations that are chained from their credential.

GLEIF vLEI Credential Example: Schema Edge OOR-Auth

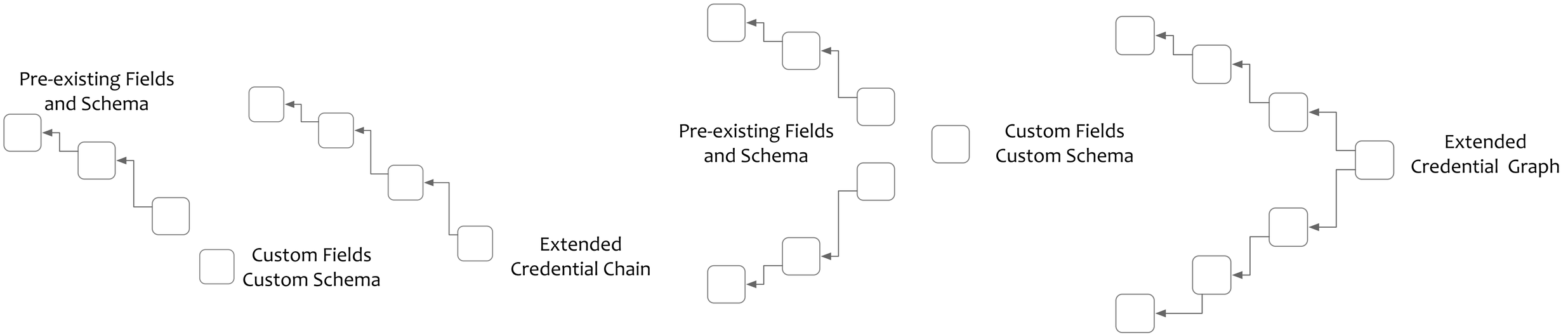
```
"e": {
      "oneOf": [
        {
          "description": "edge block SAID",
          "type": "string"
        },
        {
          "description": "edges block",
          "properties": {
            "d": {
              "description": "said of edges block",
              "type": "string"
            },
          "auth": {
                "description": "chain to Auth vLEI credential from legal entity",
                "properties": {
                  "n": {
                    "type": "string"
                  },
                  "s": {
                    "type": "string",
                    "description": "SAID of required schema of the credential pointed to by this
node",
                    "const": "EDpuiVPt4_sa1pShx6vOCnseru1edVPeNvRaQm6HrmMI"
                  },
                "o": {
                      "type": "string",
                      "description": "operator indicating this node is not the issuer",
                      "const": "I2I"
                    }
                  },
                  "additionalProperties": false,
                  "required": [
                    "n",
                    "s",
                    "o"
                  ],
                  "type": "object"
                }
              },
              "additionalProperties": false,
              "required": [
                "d",
                "auth"
              ],
              "type": "object"
            }
          ]
        },
```

Append-to-Extend

Append-only verifiable data structures have strong security properties that simplify end-verifiability & foster decentralization.

Append-only provides permission-less extensibility by downstream issuers, presenters, and/or verifiers

- Each ACDC has a universally-unique content-based identifier with a universally-unique content-based schema identifier.
- Fully decentralized name-spacing.
- Custom fields are appended via chaining via one or more custom ACDCs defined by custom schema (type-is-schema).
- No need for centralized permissioned name-space registries to resolve name-space collisions.
- The purposes of a registry now become merely schema discovery or schema blessing for a given context or ecosystem.
- The reach of the registry is tuned to the reach of desired interoperability by the ecosystem participants.
- Human meaningful labels on SAIDs are local context only.
- Versioning is simplified because edges still verify if new schema are backwards compatible. (persistent data structure model)



ACDC Summary

Push the functionality envelope of “verifiable credentials” using minimally sufficient means tooling.

Future looking features:

- SAIDS (agile self-referential content identifiers)
- Graduated Disclosure
- Contractually Protected Disclosure
- Chaining and Delegation
- Permissionless Extensibility
- Fully Decentralizable Zero-Trust via End-Verifiable Data Structures
- Multiple Serializations
- Scalability (text and binary streaming support) via CESR self-framing composable primitives and groups of primitives

Minimally Sufficient Means

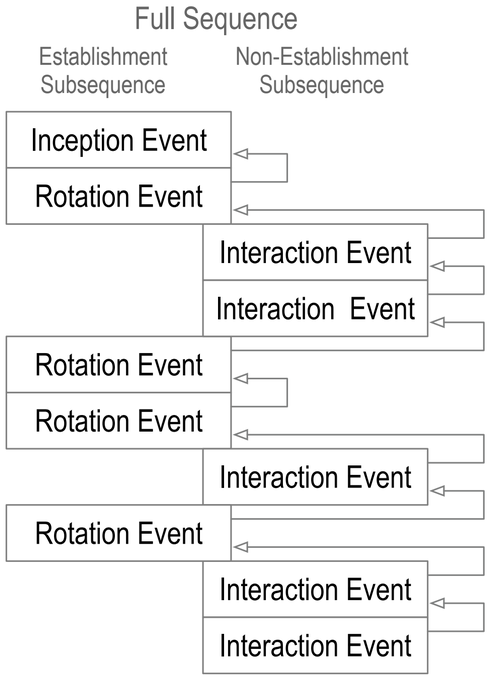
- Dumb Crypto (Digests and Digital Signatures)
- JSON and JSON Schema (composability features)
- Extensible Layered Model

KERI VDRs vs. Shared Ledger VDRs

Most DID methods use a shared ledger (commonly referred to as a *blockchain*) for their VDR. Typically, in order to interoperate all participants must use the same shared ledger or support multiple different DID methods. There are currently over 70 DID methods. Instead GLEIF has chosen to use KERI based DID methods. KERI stands for Key Event Receipt Infrastructure. KERI based VDRs are ledger independent, i.e. not locked to a given ledger. This provides a path for greater interoperability without forcing participants in the vLEI ecosystem to use the same shared ledger.

A KERI VDR is called a key event log (KEL). It is a cryptographically verifiable signed hash chained data structure, a special class of verifiable data structure. Each KERI based identifier has its own dedicated KEL. The purpose of the KEL is to provide proof of the establishment of control authority over an identifier. This provides cryptographically verifiable proof of the current set of authoritative keys for the identifier. KERI identifiers are long cryptographic pseudo random strings of characters. They are self-certifying and self-managing.

A KERI identifier is abstractly called an Autonomic Identifier (AID) because it is self-certifying and self-managing. A KERI DID is one concrete implementation of a KERI AID. The same KERI prefix may control multiple different DIDs as long as they share the same prefix.



`did:keri:`*`prefix`*`[:`*`options`*`][/`*`path`*`][?`*`query`*`][#`*`fragment`*`]`

`did:keri:ENqFtH6_cfDg8riLZ-GDvDaCKVn6clOJa7ZXXVXSWpRY`

KERI Identifier KEL VDR *Controls* Verifiable Credential Registry TEL VDR

A KERI KEL for a given identifier provides proof of authoritative key state at each event. The events are ordered. This ordering may be used to order transactions on some other VDR such as a Verifiable Credential Registry by attaching anchoring seals to KEL events.

Seals include cryptographic digest of external transaction data.

A seal binds the key-state of the anchoring event to the transaction event data anchored by the seal.

The set of transaction events that determine the external registry state form a log called a Transaction Event Log (TEL).

Transactions are signed with the authoritative keys determined by the key state in the KEL with the transaction seal.

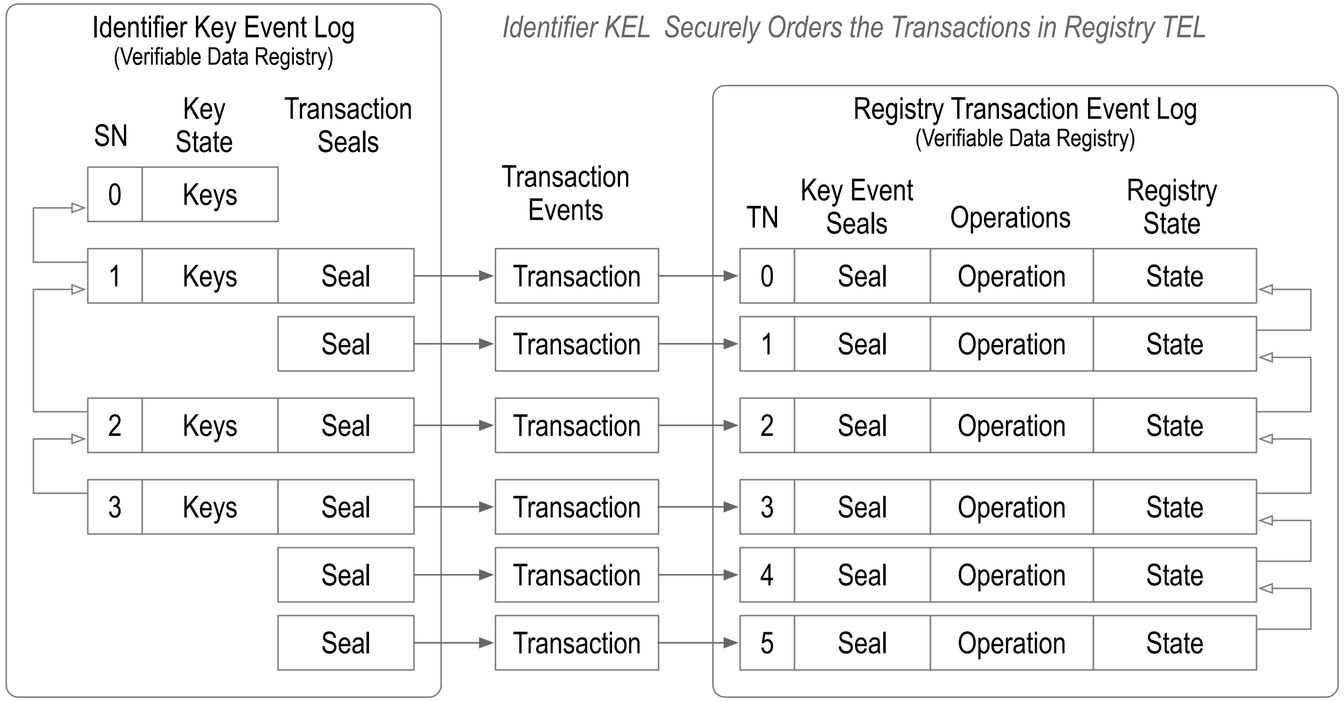
The transactions likewise contain a reference seal back to the key event authorizing the transaction.

This setup enables a KEL to control a TEL for any purpose. This includes what are commonly called “smart contracts”.

The TEL provides a cryptographic proof of registry state by reference to the corresponding controlling KEL.

Any validator may therefore cryptographically verify the authoritative state of the registry.

In the case of the vLEI the associated TEL controls a vLEI issuance and revocation registry.



seal = proof of authenticity

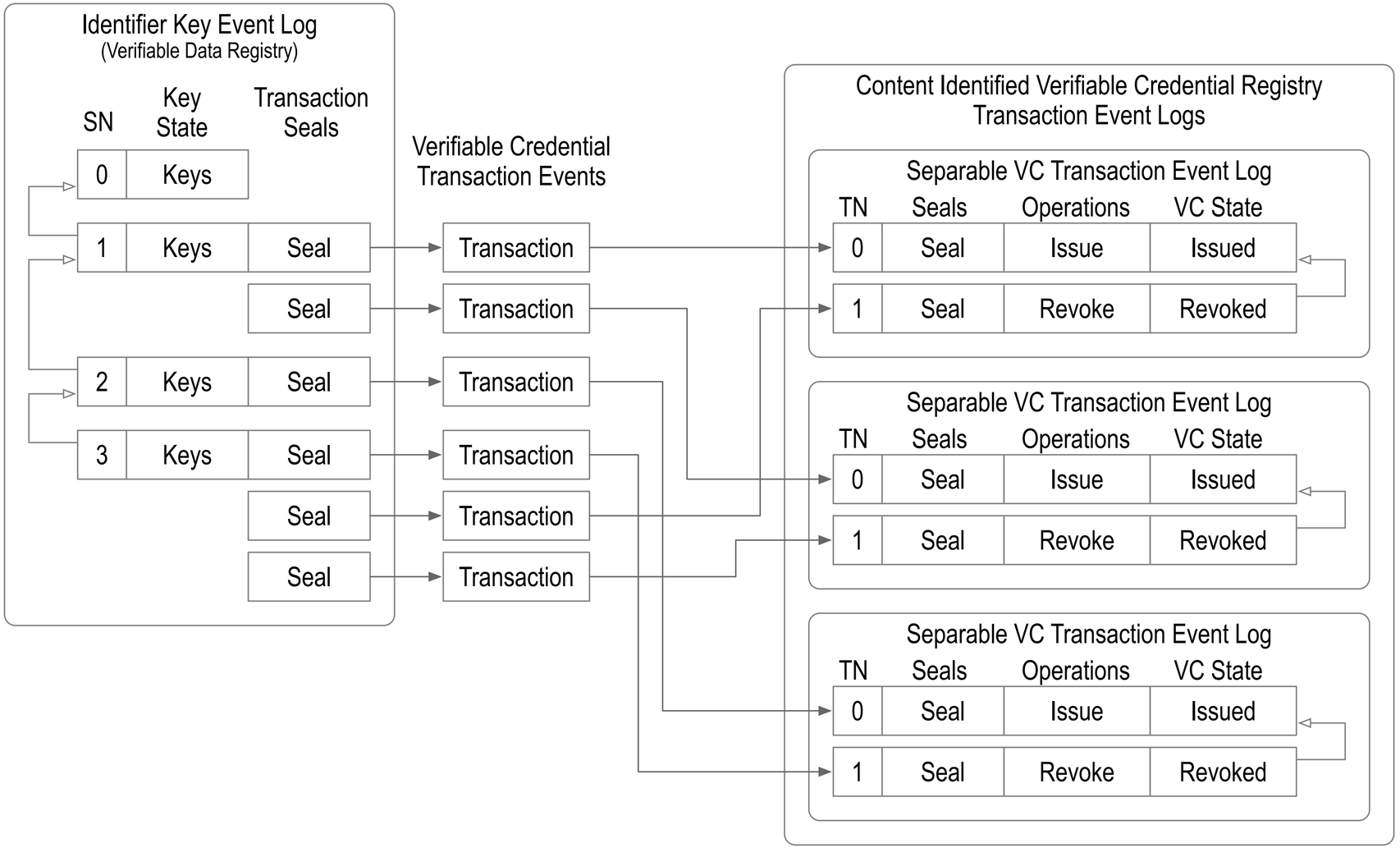
Registry with Separable VC Issuance-Revocation TELs

Each VC may be uniquely identified with a SAID.

Each VC also has a uniquely identified issuer using a KERI AID.

This combination enables a separable registry of VC issuance-revocation state.

The state may employ a cryptographic aggregation (such as an accumulator) for enhanced privacy



Least Disclosure

Principle of Least Disclosure

ACDCs are designed to satisfy the principle of least disclosure.

The system should disclose only the minimum amount of information about a given party needed to facilitate a transaction and no more.

Partial Disclosure

- Compactness
- Chain-link Confidentiality

Selective Disclosure

- Unbundling
- Bulk-issuance

Mechanisms:

- Compact via SAID over content
- Blinded via SAID over content with embedded UUID (salty-nonce)
- Unbundled via Aggregate of bundle of blinded content
- Uncorrelatable via bulk issued blinded content

Three Party Exploitation Model

First-Party = Discloser of data.
Second-Party = Disclosee of data received from First Party (Discloser).
Third-Party = Observer of data disclosed by First Party (Discloser) to Second Party (Disclosee).

Second-Party (Disclosee) Exploitation
implicit permissioned correlation.
    no contractual restrictions on the use of disclosed data.
explicit permissioned correlation.
    use as permitted by contract
explicit unpermissioned correlation with other second parties or third parties.
    malicious use in violation of contract

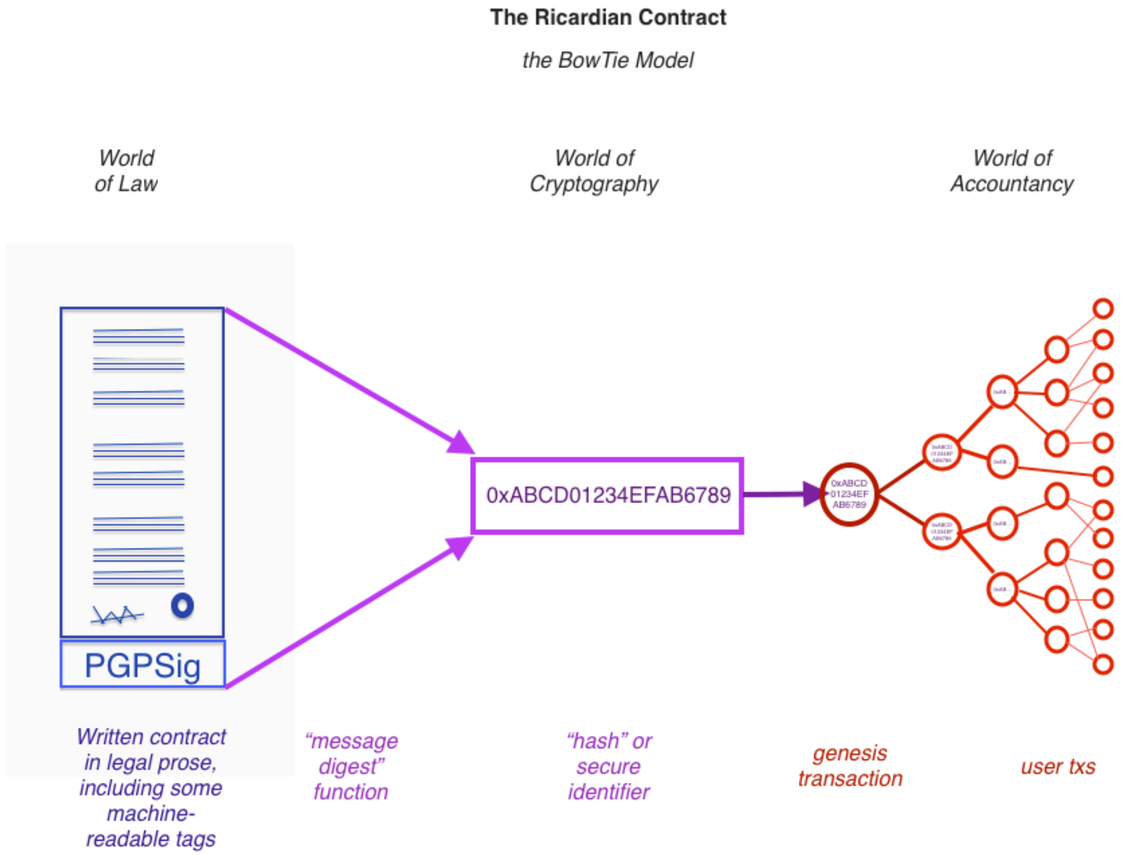
Third-Party (Observer) Exploitation
implicit permissioned correlation.
    no contractual restrictions on use of observed data.
explicit unpermissioned correlation via collusion with second parties.
    malicious use in violation of second party contract

Contractually Protected Disclosure

Ricardian Contracts:
https://en.wikipedia.org/wiki/Ricardian_contract

Chain-link Confidentiality
https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2045818

Consent, Waiver, Terms-of-use, Remuneration, etc.



Chain-Link Confidentiality

https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2045818

A chain-link confidentiality regime contractually links the disclosure of information to obligations to protect that information as the information moves downstream.

The system focuses on the relationships not only between the discloser of information and the initial recipient but also between the initial recipient and subsequent recipients.

Through the use of contracts, this approach links recipients of information as in a chain, with each subsequent recipient bound by the same obligation to protect the information.

These chain contracts contain at least three kinds of terms:

1) obligations and restrictions on the use of the disclosed information;

2) requirements to bind future recipients to the same obligations and restrictions; and

3) requirements to perpetuate the contractual chain.

This approach creates a system for the permissible dissemination of information.

It protects Disclosers by ensuring that the recipient's obligation to safeguard information is extended to third parties.

Contractual Exchange

Discloser provides a non-repudiable Offer with verifiable metadata (sufficient partial disclosure) which includes any terms or restrictions on use.

Disclosee verifies Offer against composed schema and metadata adherence to desired data.

Disclosee provides non-repudiable Accept of terms that are contingent on compliant disclosure.

Discloser provides non-repudiable Disclosure with sufficient compliant detail.

Disclosee verifies Disclosure using decomposed schema and adherence of disclosed data to Offer.

Disclosee may now engage in permissioned use and carries liability as a deterrent against unpermissioned use.

Interoperability Through Layering

The ACDC/KERI stack is opinionated about security with very precisely defined properties.

Fully decentralizable, distributable, zero-trust, end-verifiable over-the-wire mechanisms for authenticatable extensible verifiable data structures (append-only hash chained signed)

Mashups of security mechanisms that sort of provide those security properties are antithetical to the ACDC/KERI stack ethos.

Interoperable security first, then interoperable semantics for upper layers of the application stack.

ACDCs could be used as a secure conveyance for other representations that appear as an opaque payload in the ACDC.

ACDCs could be a blessed trust spanning layer for W3C VCs for those who want its security properties.

ACDC Community Interoperability Ask

Enable one-to-one mappings between ACDCs and other data models and representations without forcing ACDCs to use syntax from other data models and representations.

ACDCs use JSON-Schema with JSON, CBOR, MGPK, and CESR serializations, compact labels & CESR Primitives.

CESR primitives can be mapped one-to-one to JWT primitives.

ACDC normative field labels can be mapped one-to-one to their equivalents in other representations.

Allow JSON Schema (ACDCs need composition operators from JSON Schema).

Not asking to replace JSON-LD but be allowed to co-exist with JSON-LD.

JSON is well just JSON. Please no MUST have non-JSON artifacts (@context).

Questions?