

Verifiable Credentials WG

TPAC 2022

Day 1: September 15, 2022

Chairs: Brent Zundel, Kristina Yasuda

Location: Vancouver (and cyberspace)

Welcome!

- Logistics
- W3C WG IPR Policy and Health Policy
- Agenda
- IRC and Scribes
- Status
- Timeline Reminder

Logistics

- **Zoom call:**
 - See <https://www.w3.org/events/meetings/a91c3c5b-298a-46a8-8575-61cf15926a97> for dial in information (member only link)
- **Meeting times:**
 - Thursday Sep 15: [8:00-17:00 PST](#)
 - Friday Sep 16: [8:00-17:00 PST](#)
- **VC WG Agenda:** <https://tinyurl.com/3kyhzejj>
- **Live slides:** <https://tinyurl.com/458sp44s> (Google Slides)

W3C WG IPR Policy

- This group abides by the W3C patent policy
<https://www.w3.org/Consortium/Patent-Policy-20200915/>
- Only people and companies listed at <https://www.w3.org/groups/wg/vc/participants> are allowed to make substantive contributions to the specs
- Code of Conduct <https://www.w3.org/Consortium/cepc/>

Today's agenda

8:00 Chairs Introduction and logistics		
8:30	Chairs Introduction and logistics	Brent Zundel
9:00	Real World Use Cases	Chairs
9:30	Updating the Use Cases Note	Joe Andrieu
10:00		
10:15	Transformations of the core data model	Chairs
10:45	Transformations of the core data model	Chairs
11:15	Transformations of the core data model	Orie Steele
12:15		
13:15	APA WG Joint Meeting (use-cases, etc.)	
14:15	Streamlining Data Integrity Cryptosuites	Manu Sporny
14:45	Streamlining Data Integrity Cryptosuites	Manu Sporny
15:15		
15:30	VC Extensions Registries	Mike Jones/Orie Steele
16:00	VC Extensions Registries	Mike Jones/Orie Steele
16:30	Intro to ACDC	Sam Smith (remote)

IRC and Scribes

- Meeting discussions will be documented

- Text Chat:

<http://irc.w3.org/?channels=vcwg>

- IRC://[irc.w3.org:6665/#vcwg](irc://irc.w3.org:6665/#vcwg)

- Telecon info

- <https://www.w3.org/events/meetings/a91c3c5b-298a-46a8-8575-61cf15926a97>

	Morning 1	M2	Afternoon 1	A2
Thurs	Manu	Mkhraisha	Phila	CEL
Fri	Joe	Kevin/Phila	DLongley	Orie

```
<JoeAndrieu> q+ to comment on biometrics  
<brent> ack JoeAndrieu  
<Zakim> JoeAndrieu, you wanted to comment on biometrics
```

VC WG Mission and Goals

- “The mission of the Verifiable Credentials Working Group is to make expressing, exchanging, and verifying credentials easier and more secure on the web.”

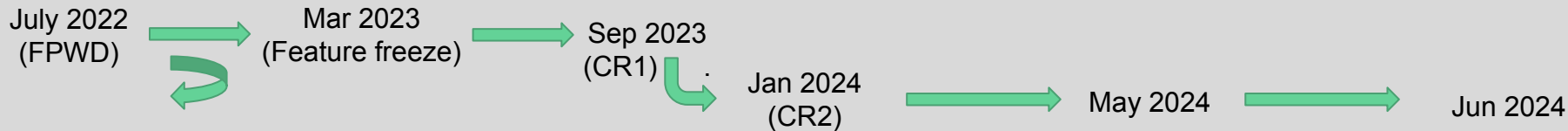
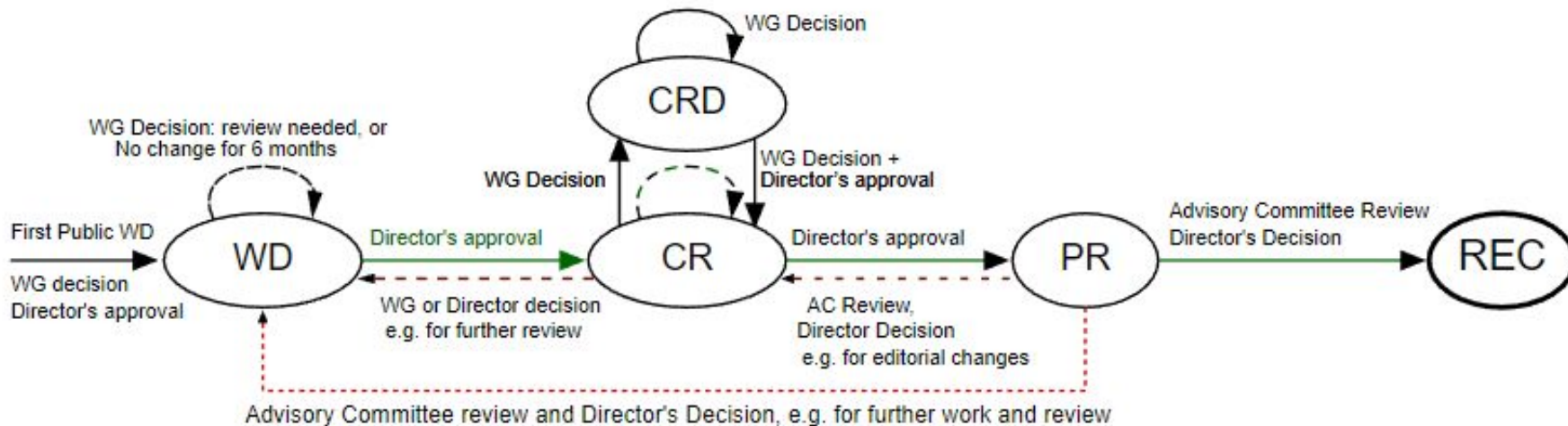
Charter Deliverables and Status

- **Verifiable Credentials Data Model (VCDM) 2.0**
- **Verifiable Credential Data Integrity 1.0**
- **Verifiable Credential JSON Web Token**
- **Verifiable Credential JSON Web Signature 2020**

W3C Technical Report Process

- Working Draft (WD) - does not imply consensus
- Candidate Recommendation (CR)
 - Entry - to publish as CR, the document is expected to be feature complete, have had wide review, and must specify the implementation requirements needed to exit
 - Exit - to exit CR (and move to PR), the document must satisfy the stated implementation requirements; it must also not have made any substantive change not warned about upon entry
- Proposed Recommendation (PR)
 - Basically a one-month sanity check during which the AC is encouraged to have any final review and discussion, but if anything major happens it's a fail (requiring a move back to CR or earlier)
- Recommendation - Done
 - But errata are possible

Timing of our primary spec



<https://www.w3.org/2021/Process-20211102/>

Goals for this meeting

- Make progress on the current set of boulders
 - Core Data Model
 - Border between core DM and Securing VCs
 - Transformations, etc.
 - Disposition of non-normative content
 - Data Integrity streamlining
 - VC Registries
 - Test Suites

Introductions

Real World Use Cases & what did we learn from implementing v1.1

(Brent, 15 min)

Where are VCs used Today?

- Travel documentation
- Staff onboarding
- Customer relationships
- Age verification
- Supply chain tracking
- Education credentials
- Content Authenticity
- Govt ID
- eKYC
- What else?

Discussion

- What did the implementers learn?
- Benefits of VC? What do we like about the vc-data-model v1.1 spec?
- Problems, asks, questions for vc-data-model-v2?

Updating Use Cases

(Joe Andrieu & Kevin Dean, 45 min)

Topics

- Current Use Case Document
- Lessons Learned
- Proposal for New Content
- Next Steps
- Open Discussion

Current Document

W3C Working Group Note

TABLE OF CONTENTS

- 1. Introduction**
 - 1.1 Importance of this work
 - 1.2 Use case model
- 2. User Roles**
- 3. User Needs**
 - 3.1 Education
 - 3.2 Retail
 - 3.3 Finance
 - 3.4 Healthcare
 - 3.5 Professional Credentials
 - 3.6 Legal Identity
 - 3.7 Devices
- 4. User Tasks**
 - 4.1 Issue Claim
 - 4.2 Assert Claim
 - 4.3 Verify Claim
 - 4.4 Store / Move Claim
 - 4.5 Retrieve Claim
 - 4.6 Revoke Claim
- 5. Focal Use Cases**
 - 5.1 Citizenship by Parentage
 - 5.2 Expert Dive Instructor

Verifiable Credentials Use Cases

W3C Working Group Note 24 September 2019



This version:

<https://www.w3.org/TR/2019/NOTE-vc-use-cases-20190924/>

Latest published version:

<https://www.w3.org/TR/vc-use-cases/>

Latest editor's draft:

<https://w3c.github.io/vc-use-cases/>

Previous version:

<https://www.w3.org/TR/2017/NOTE-verifiable-claims-use-cases-20170608/>

Editors:

[Shane McCarron](#) (Spec-Ops)

[Joe Andrieu](#) (Legendary Requirements)

[Matt Stone](#) (The Brightlink)

[Tzviya Siegman](#) (Wiley)

[Gregg Kellogg](#) (Spec-Ops)

[Ted Thibodeau, Jr.](#) (OpenLink Software, Inc.)

Authors:

[Nate Otto](#) (Badge Alliance)

[Sunny Lee](#) (Badge Alliance)

[Brian Sletten](#) (Bosatsu Consulting, Inc.)

[Daniel Burnett](#) (Standards Play)

[Manu Sporny](#) (Digital Bazaar, Inc.)

[Ken Ebert](#) (Sovrin Foundation)

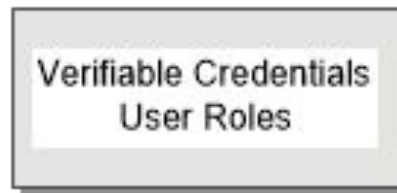
Participate:

[GitHub w3c/vc-use-cases](#)

<https://w3.org/TR/vc-use-cases>

Roles

Four Primary Roles in the system



Issuer



Verifier

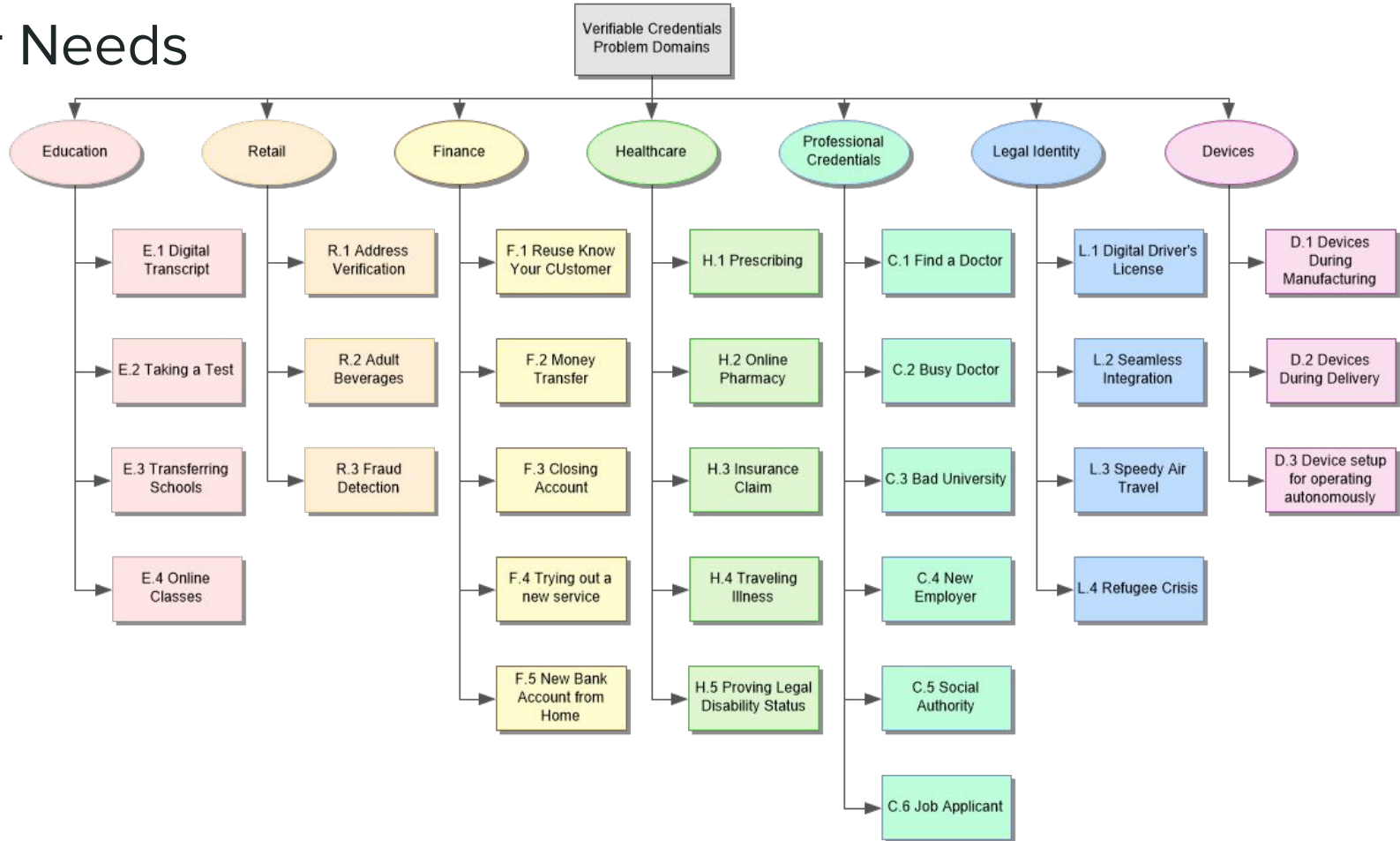


Subject



Holder

User Needs



User Needs

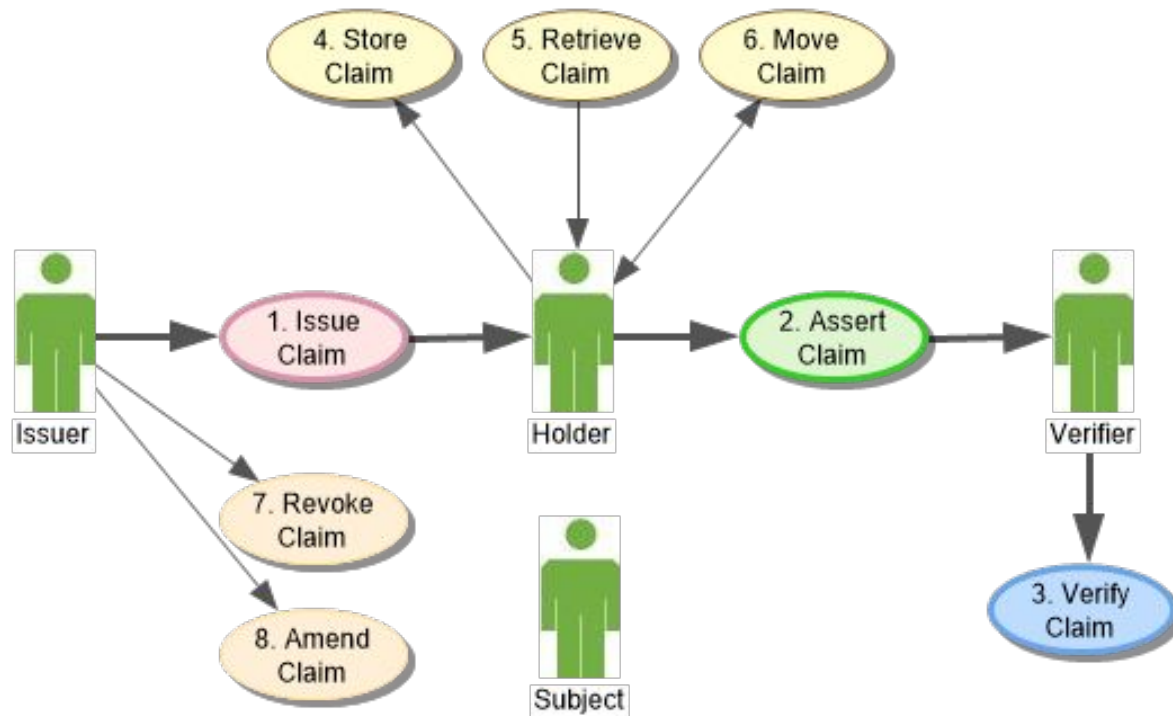
- 30 Use Cases
- In 7 Domains
- Problems/Opportunities
- Each with a paragraph

E.1 Digital transcript

Joleen is the registrar of Mega University and, by virtue of her office, is responsible for the integrity, accuracy, and security of academic records. Joleen has been a pioneering registrar in advocating an "extended transcript" that includes not only the standard set of course grades but also adds supplementary information on learner competencies. These might include work experiences and non-educational but marketable skills. Upon the request of her students, Joleen issues a digital credential that includes an extended transcript.

Verifiable Credentials
User Tasks

User Tasks



User Tasks

- Captures the tasks needed to support Normal use of VCs, for each Role
- 8 Tasks
- **Issue, Assert, Verify**, Store, Retrieve, Move, **Revoke, Amend**

4.1 Issue Claim

Requirement

It **MUST** be possible for any entity to issue a verifiable credential.

Motivation

Individuals and organizations need a way to issue claims about themselves or others that can be verified and trusted.

Needs

F.1 Reuse know your customer, E.1 Digital transcript, L.1 Digital driving license, H.1 Prescribing

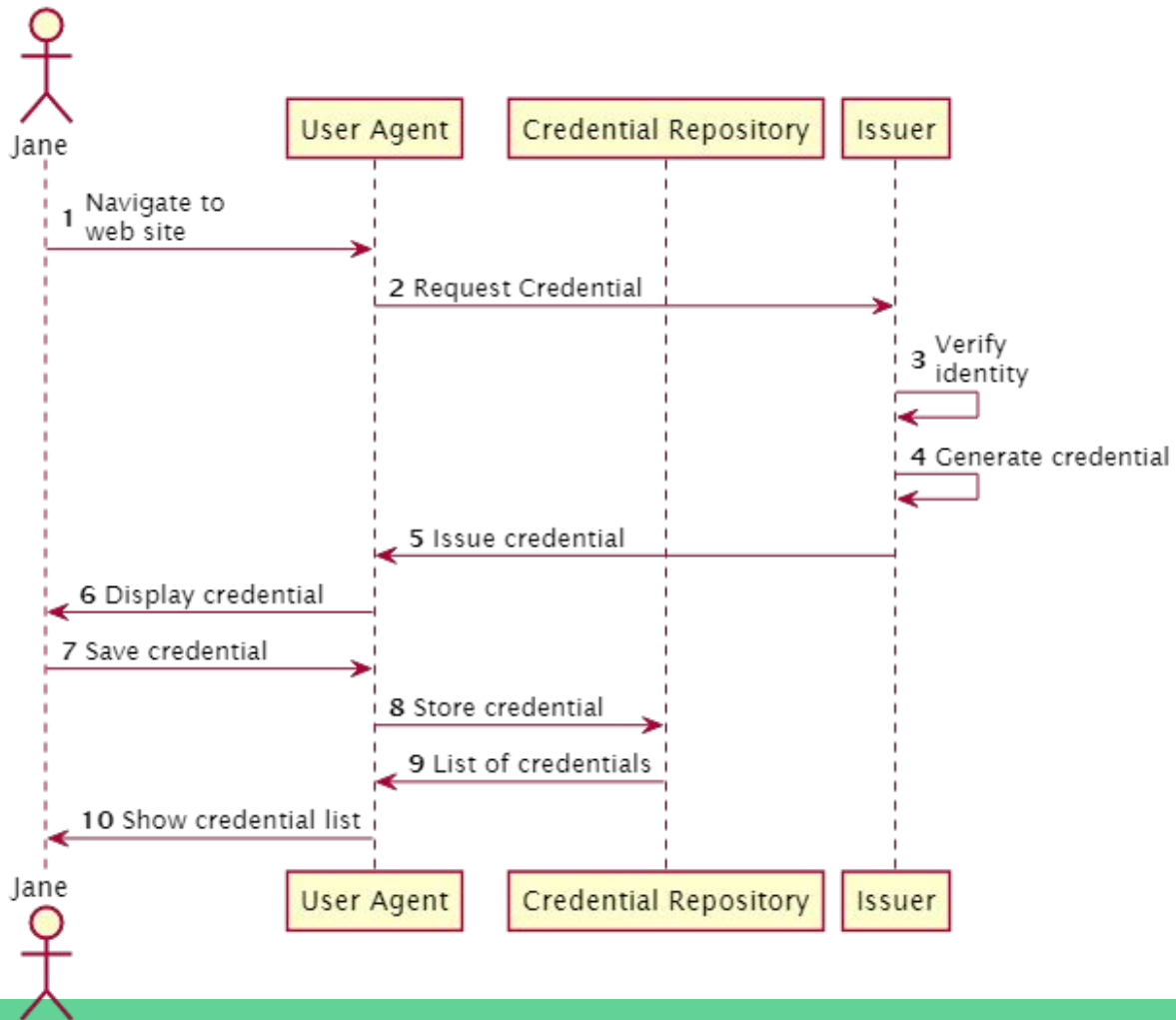
Focal Use Cases

- Three In-Depth Use Cases
 - Citizenship by Parentage
 - Expert Dive Instructor
 - International Travel with Minor and Upgrade
- Extra Content
 - Background
 - Distinction
 - VCs and VPs involved
 - Trust Hierarchy
 - Threat Model
 - Threat / Responses

User Sequences

Creating a VC

Example credential creation flow



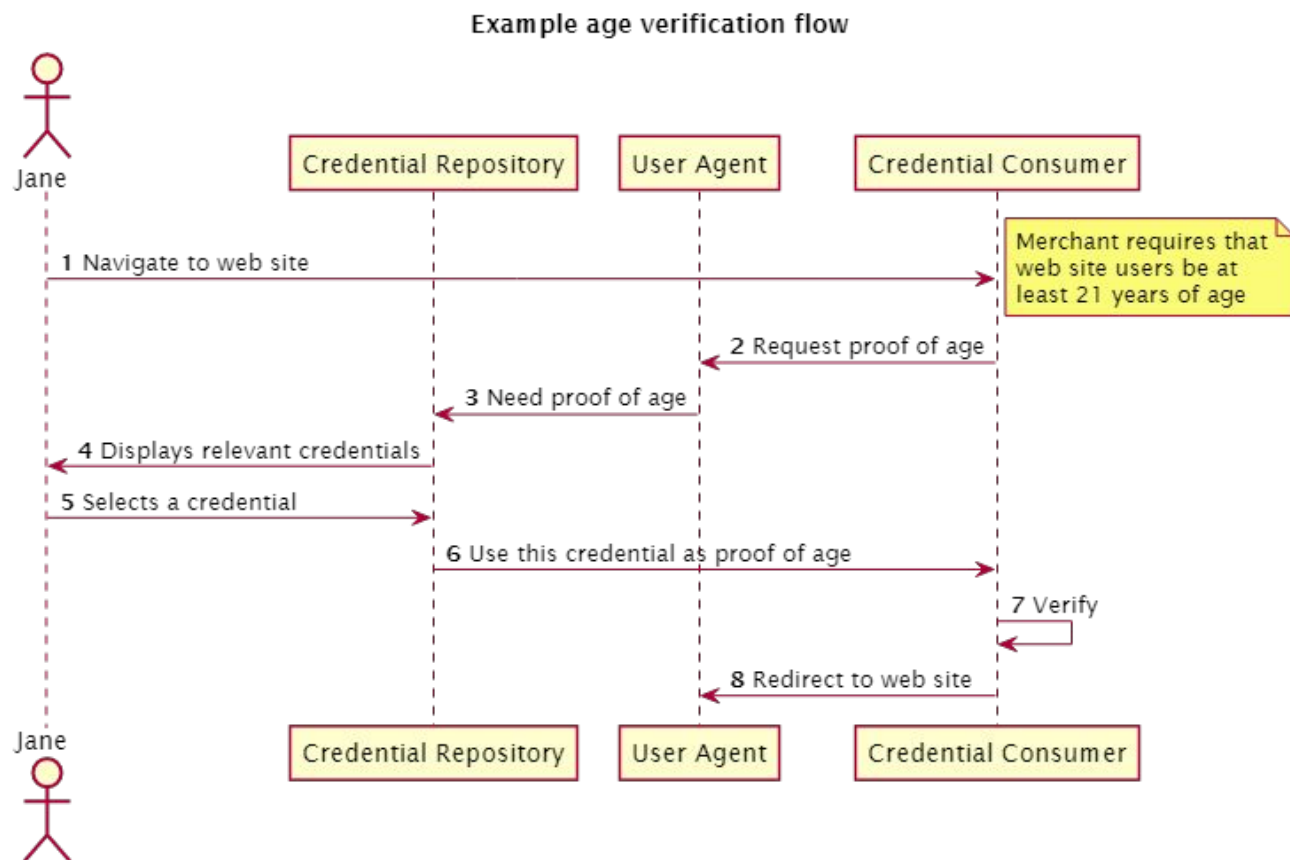
Example VCs (5)

EXAMPLE 1: Malathi's passport (simple model)

```
{
  "@context": [
    "https://w3id.org/credentials/v1",
    "https://example.com/travel-vocab/v1"
  ],
  "id": "urn:uuid:9f6878c8-73c7-11e8-ab37-23a1a3504fd0",
  "type": ["VerifiableCredential", "PassportCredential"],
  /* gov't DID */
  "issuer": "did:example:CCnF3zFaXkPN4zB94XaomRdvw2zX3XHPVX3aExcgo6PV",
  "expires": "2028-01-01T00:00:00Z",
  "claim": {
    "id": "did:example:BcRisGnqV4QPb6bRmDCqEjyuubBarS1Y1nhDwxBMTXY4",
    "givenName": "Malathi",
    "familyName": "Hamal",
    "citizenship": "US",
    /* any other claims made by gov't */
  },
  "proof": { /* signature by gov't */ }
}
```

User Sequences

Using a VC



Lessons Learned

- Amend
- Evidence
- How do you refer to a VC?
- Hard to get contributions
 - Hard to keep iterating
- VCs are statements, but ...
 - its subtle and we struggle to get that across.
 - @Context matters
- Wallet Descriptors/Display
- Visual Presentation Layer

New Content Proposal

1. Update VC Use Cases
2. New Editors: Joe Andrieu, Kevin Dean
3. Update Actions
 - a. Replace “Amend” with “Reference”
 - b. Store, Retrieve, Move?
4. New Use Cases for better coverage
 - a. Traceability
 - b. Evidence
 - i. Reference / inclusion of authority to issue (as VC)
 - ii. Reference / inclusion of proof of control
 - c. Mobile / Web Integration
 - d. Accessibility (especially screen-reader style complications)
 - e. Display of VCs?

Traceability - The Problem

- Full traceability records everything that happens as the product moves through the supply chain
 - Not just the movements themselves, but the conditions (temperature, humidity, etc.)
- Discovering data is difficult, usually just one-up and one-down
- A discovery service could respond to broadcast requests but who gets to ask?
 - Consumer with product in hand (authenticated by use of valid serial number)
 - Trading partner
 - Regulator

Traceability - Responding to Queries

- How does the discovery service respond to the broadcast?
 - No response (timeout)
 - “I have no data”
 - “I may have data”
 - “I have data”
- How does the discovery service respond to a direct query?
 - No response (timeout)
 - No data
 - Filtered data
 - All data

Traceability - Use of Verifiable Credentials

- Proof of role within the supply chain
 - Distributor - known to the manufacturer
 - Retailer - known to the distributor, who wants to hide identifying information from the manufacturer to protect customer base

Next Steps

1. Formally adopt the work item
2. Assign Editors
3. Additional Collaborators?
4. Call for input
5. Cadence for Review in main calls
 - a. every other month

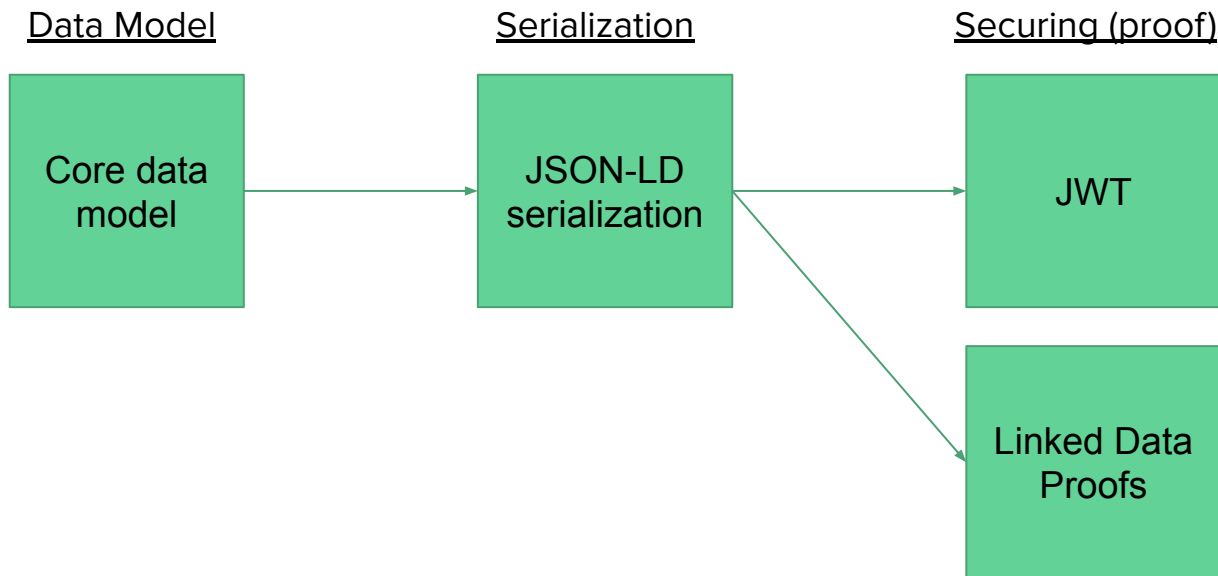
Discussion

Break
(15 mins)

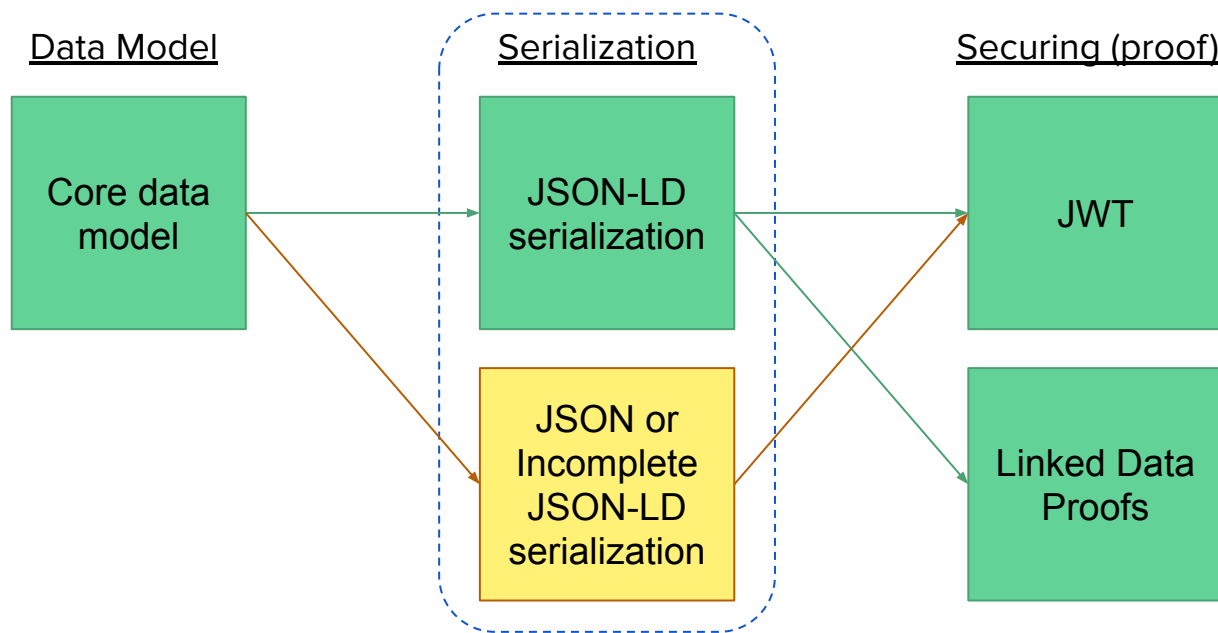
Serializations and the Core Data Model

(Kristina, 60 mins)

In VC-DATA-MODEL v1.1...Creating a Verifiable Credential



In Practice...Creating a Verifiable Credential

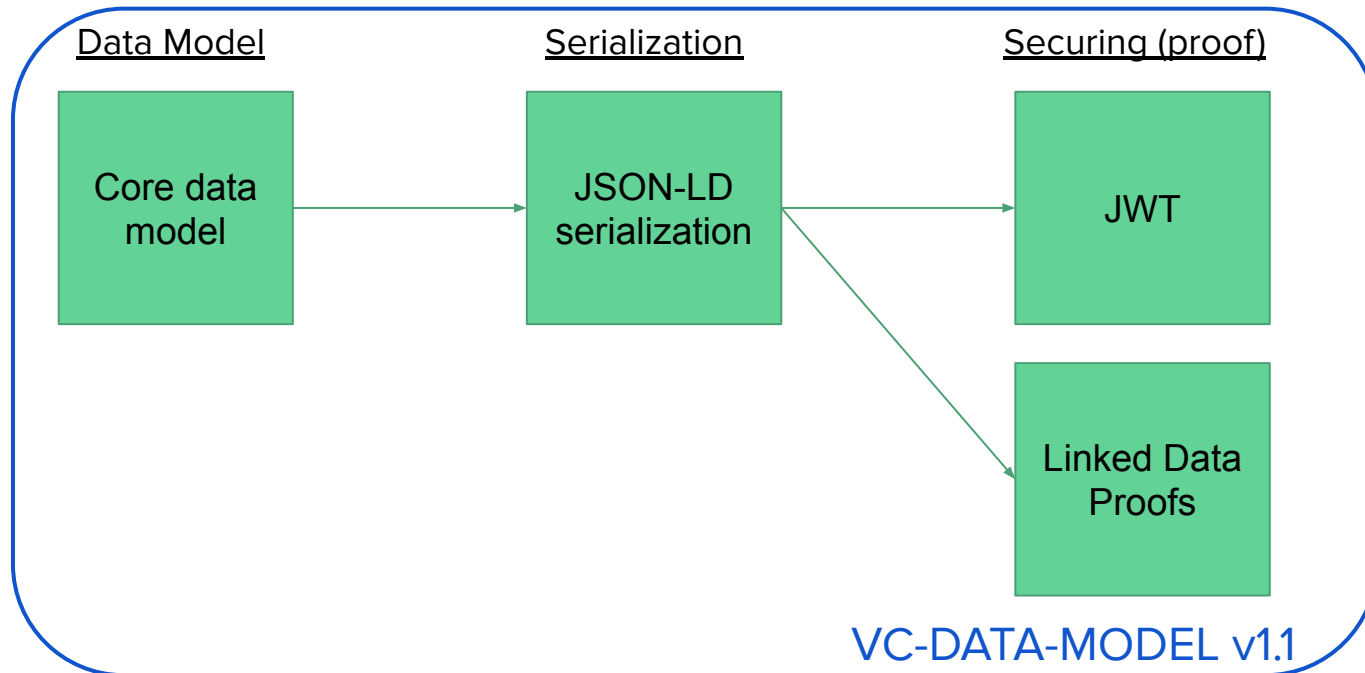


diff: valid @context present or not

-> **what to do with @context in JSON-serialized VCs?**

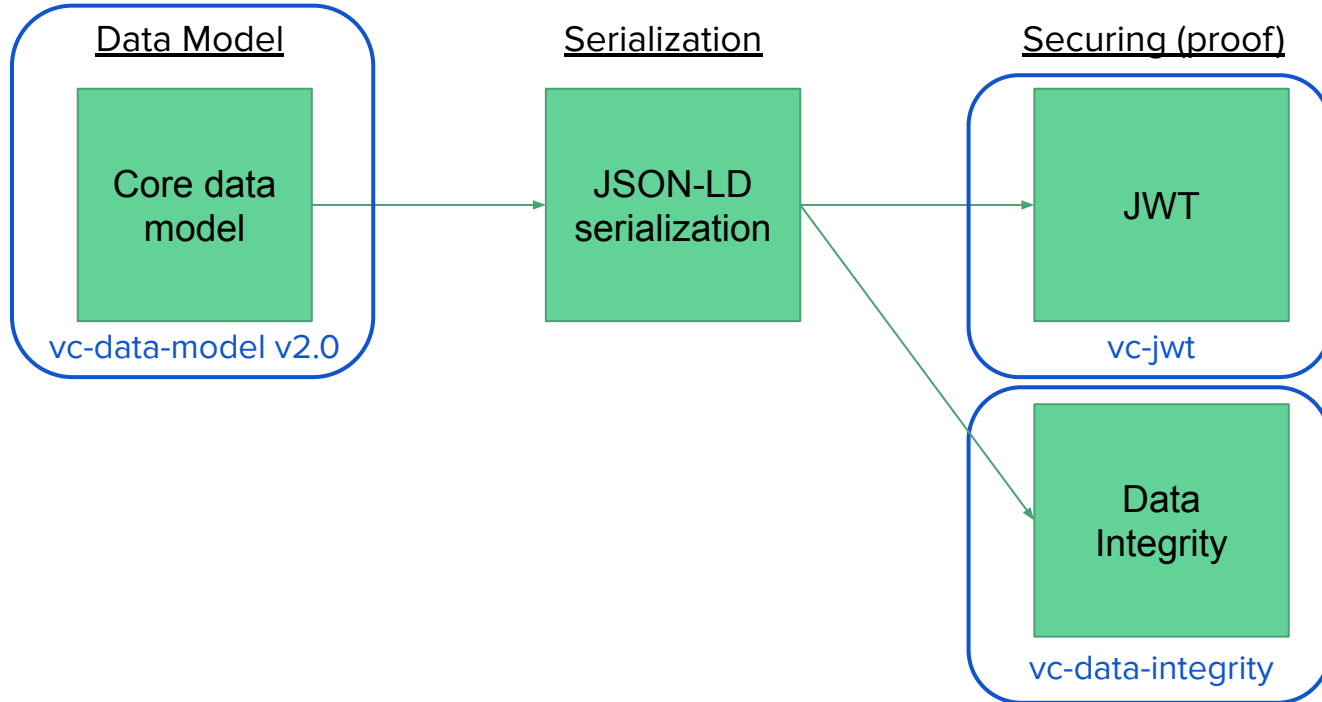
In VC-DATA-MODEL v1.1...

All components are in one specification:

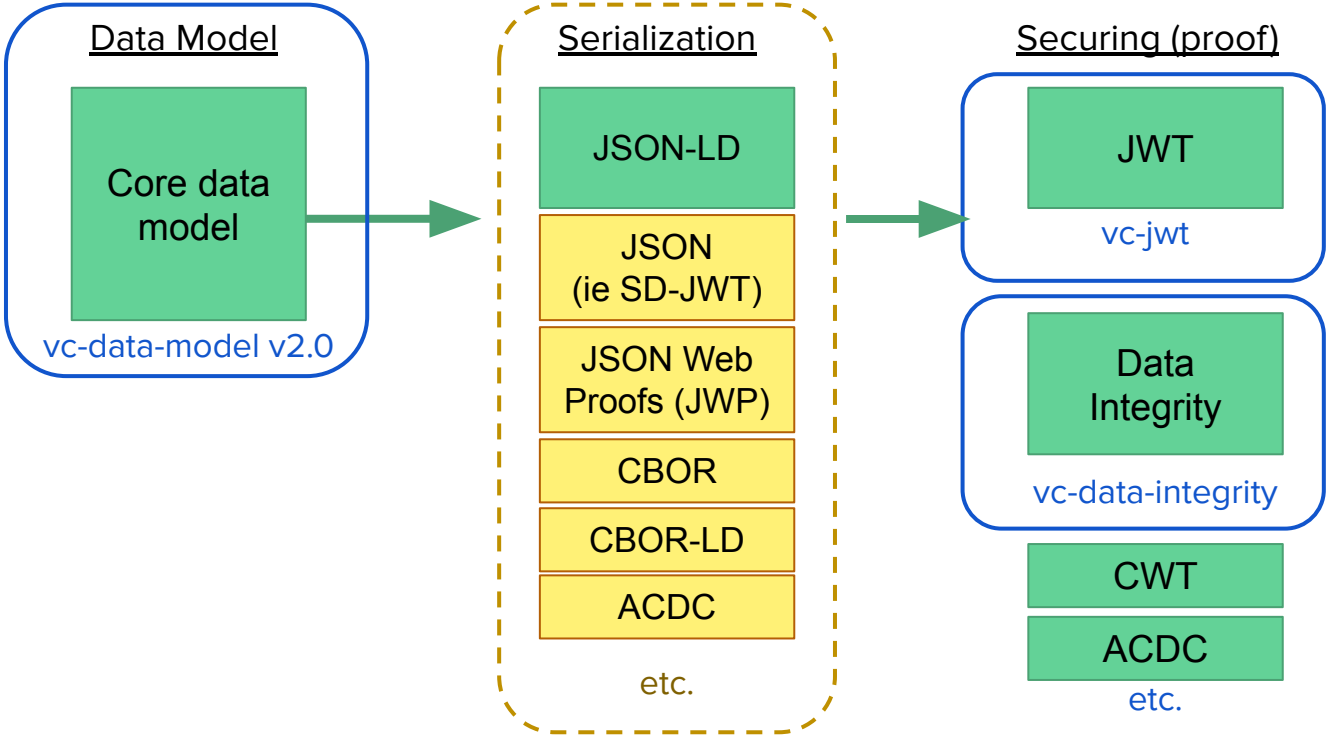


In VC-DATA-MODEL v2.0...

Different documents for each component...



Discussion topic: Where do Serialization rules sit?



Defining *vs* Securing

...the Core Data Model

(Ori Steele, 60 mins)

The branding problem.

Everyone wants to say their software implements W3C Standards...

Nobody wants to change their software.

How much of “Verifiable Credentials” is about “Graph Theory” or “Open World Data Models”... or the “Semantic Web”... or “Browser APIs”?



How do “Verifiable Credentials” reflect “W3C Values”?

What properties should **W3C** Verifiable Credentials have when compared to standards at IETF, ISO, or DIF?

[W3C TAG Ethical Web Principles](#) ?

Is the Semantic Web Dead?



The Core Data Model

Should we support decentralization?

How can Issuers, Holders and Verifiers agree to a shared information model without a trusted centralized registry?



The Core Data Model

Should be specific and useful enough to justify a new standard

We already have JSON and JOSE...

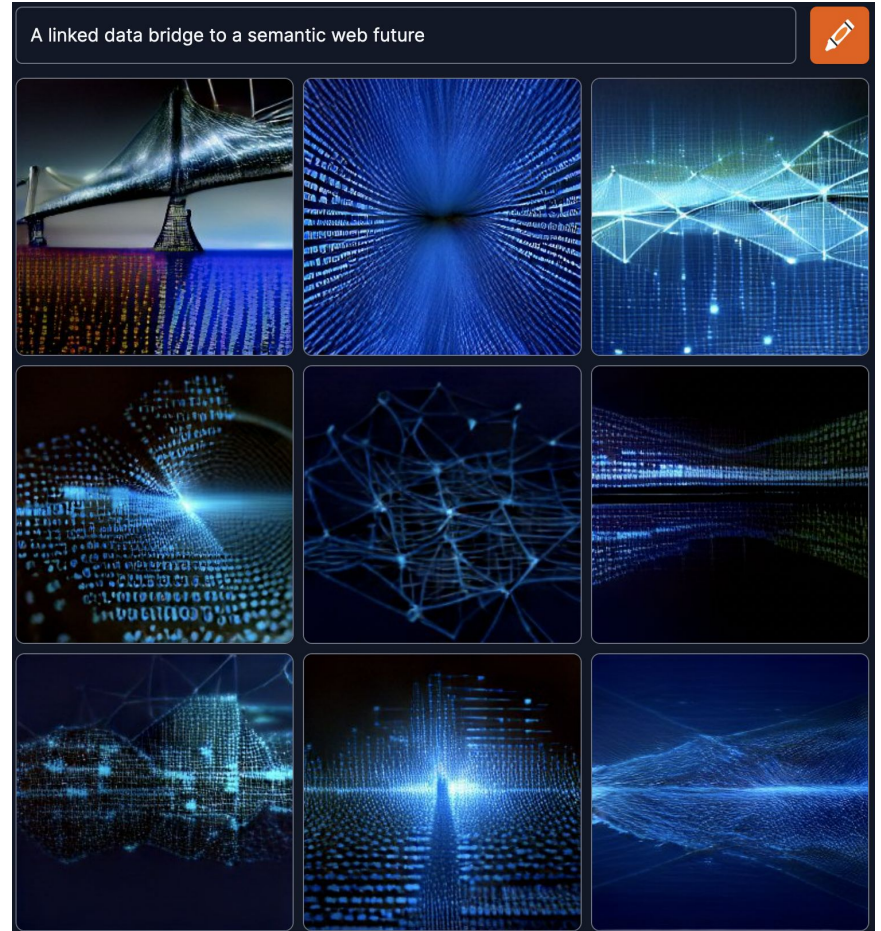
We already have CBOR and COSE...

Why are we here?



The Core Data Model

Provide a bridge linking the semantic web of the past, to the semantic web of the future?

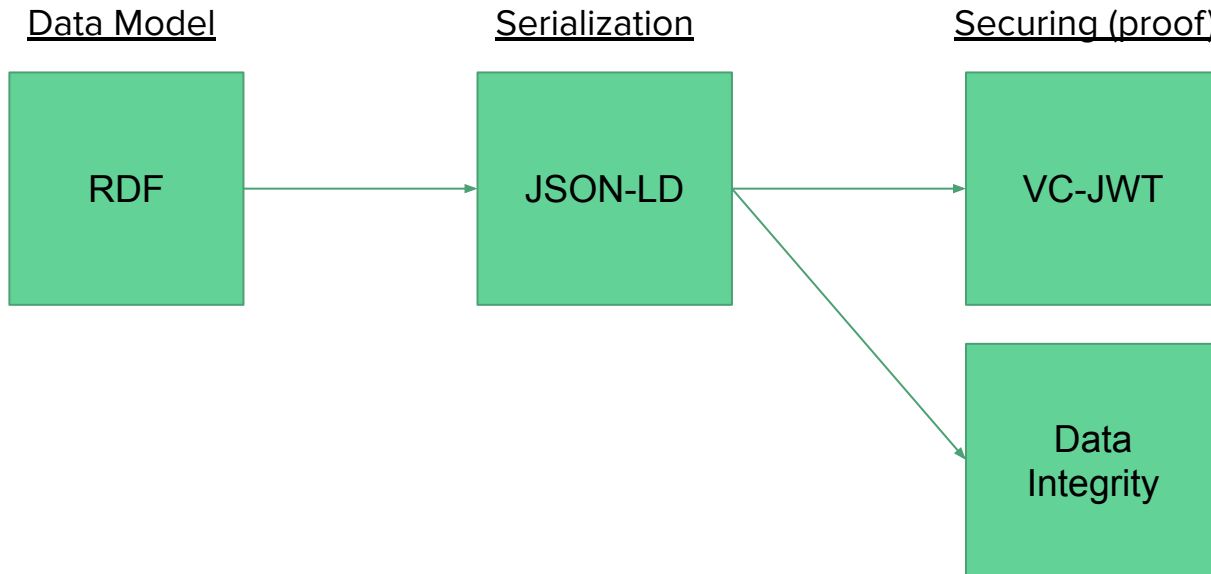


What is the Core Data Model?

- <https://www.w3.org/TR/2022/REC-vc-data-model-20220303/>
 - Normative statements only.
- What about all the informative sections?
 - Should they be moved to the implementation guide?
 - Should they remain in the core spec informatively?
 - Should they remain in the core spec normatively?

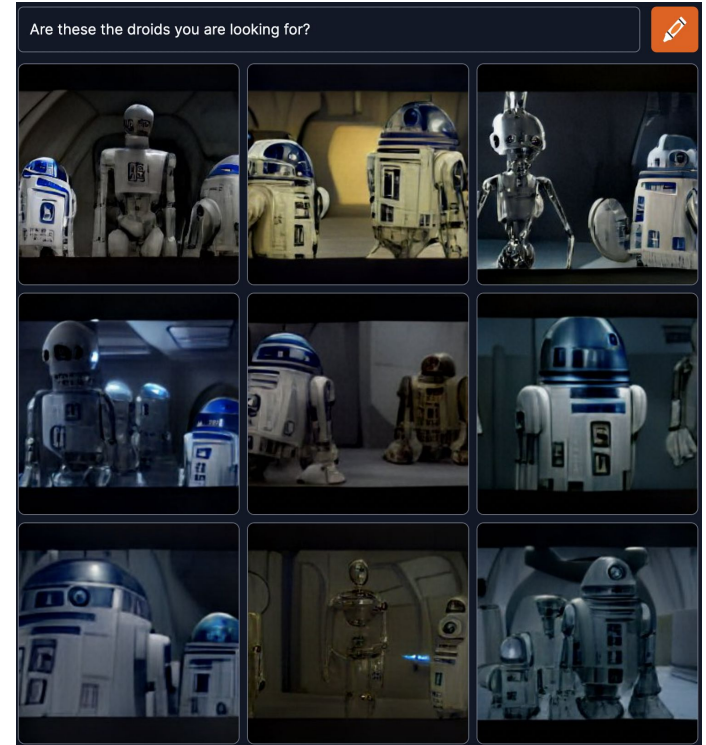
Creating a Verifiable Credential (in VC-DATA-MODEL v1.1)

- Vocabularies are used for extensibility
- Claims are defined in relation to RDF triples: subject, predicate, object



Where are the lines blurred today?

- “signature” or “proof” metadata
 - iat, nbf, iss, sub
 - proof.created, issuanceDate, issuer, credentialSubject
- Content Types, Semantics & Schema Languages (JSON Schema / JSON-LD)
- Which representations are “verifiable credentials”?
 - application/jose
 - application/jose+vc
 - application/jose+json+vc
 - application/jose+json+ld+vc
 - application/cose+vc
 - application/cbor+ld+vc



Which “core data model” terms do we also see in “securing formats” ?

- * [@context](#) → The context assigned to the credential → ... ? [private claim names](#) ?
- * [type](#) → The type (in the context), assigned to the credential → ...
- * [issuer](#) → The claim identifies the principal that issued the credential → [iss](#) + [public claim names](#) + [private claim names](#)
- * [issuanceDate](#) → When should a verify start accepting the claims from the issuer? → [nbf](#)
- * [credentialSubject](#) → The subject and claims about the subject of the credential → [sub](#) + [public claim names](#) + [private claim names](#)

[id](#) → The unique identifier assigned to the credential → [iti](#)

[holder](#) → The claim identifies the principal that holds a credential → [iss](#), [sub](#) ... ?

[expirationDate](#) → When should a verify stop accepting the claims from the issuer? → [exp](#)

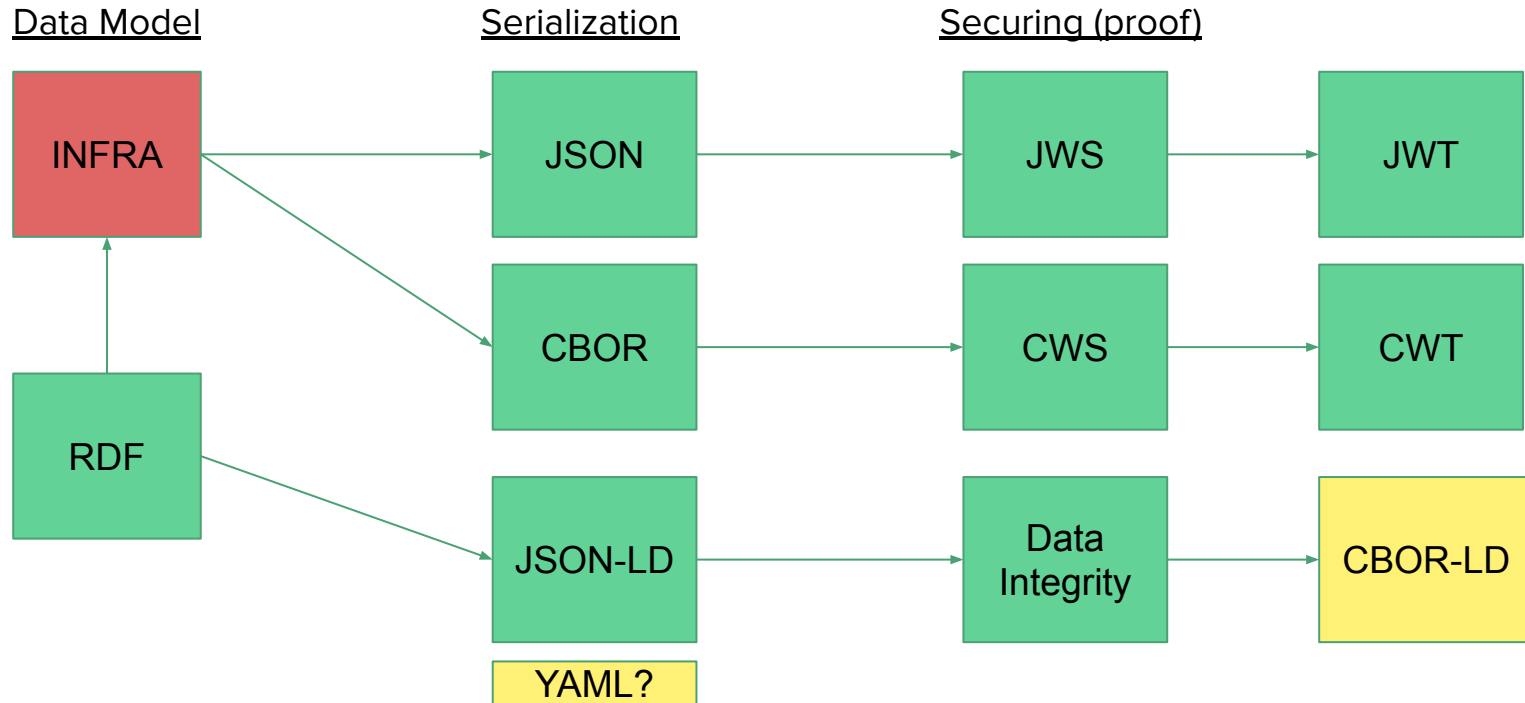
[credentialStatus](#) → Has the issuer revoked this credential? → ... ?

[credentialSchema](#) → What schema validates this credential? (what type is this json) → ... ?

Do we really want to specify all these for every new “proof format” ?

Who wants an unbounded number of representations for arbitrary content?

Repeating the mistakes(?) of DID Core?



Relaxing the verifiable credential requirements...

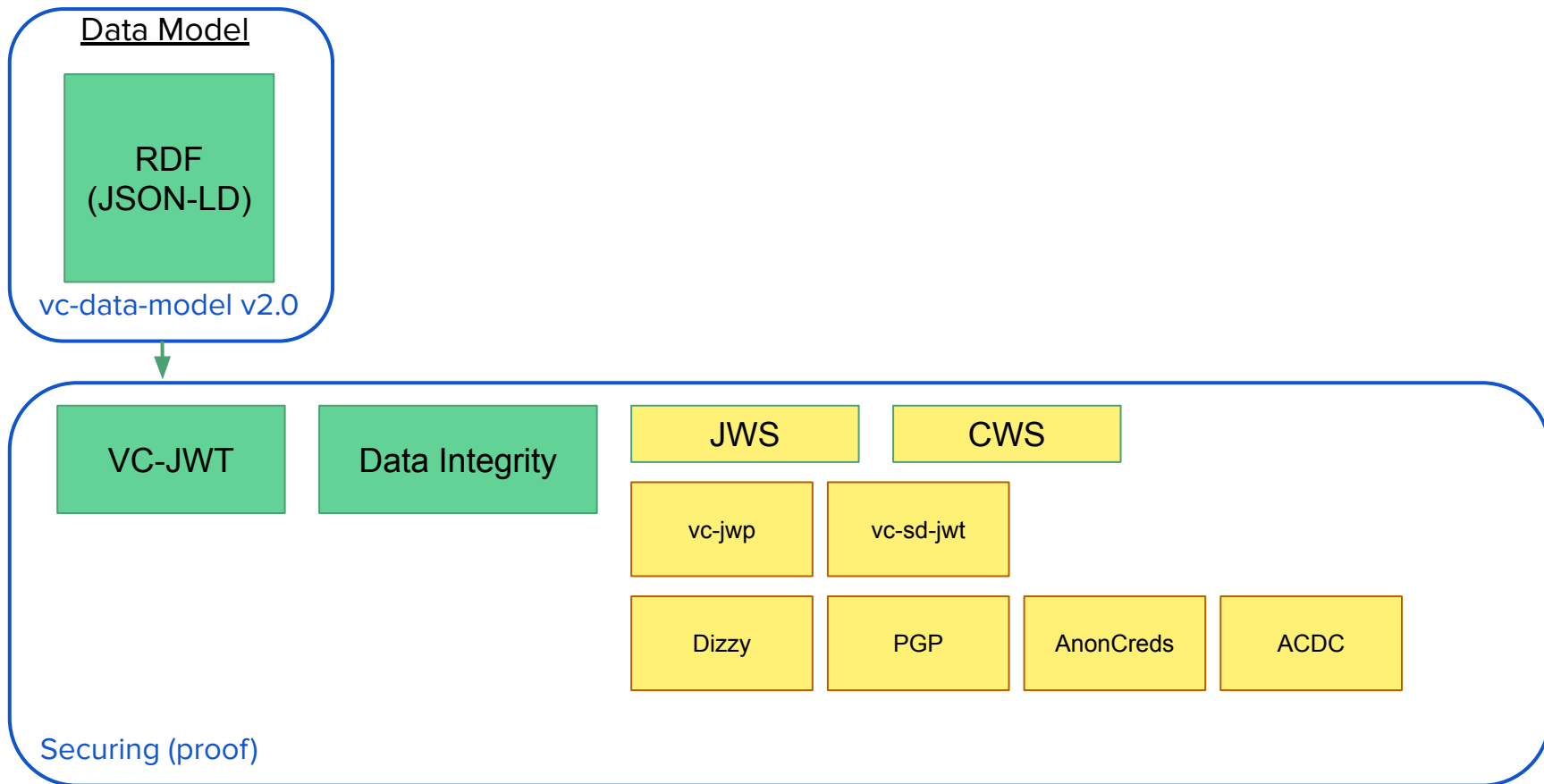
- v1: JWT + Linked Data Proofs... over JSON-LD.
- v2: JWT + Data Integrity Proofs + (JWS, JWE) ?.. over JSON-LD.
 - <https://or13.github.io/draft-osteele-vc-jose/>
- v3: ... ?

What about mDoc?

What about CBOR?...

What about TOML and YAML?

Producing and Securing **Semantic** Verifiable Credentials?



Open PRs

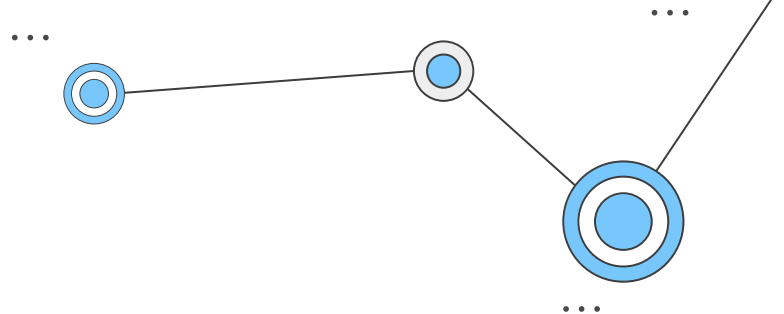
- Defining JWT production rules
 - <https://github.com/w3c/vc-jwt/pull/11>
- Relaxing the JWT requirements
 - <https://github.com/w3c/vc-jwt/pull/10>
- Leveraging IANA and existing registries
 - <https://github.com/w3c/vc-jws-2020/pull/24>

Images credits to the AI at: <https://www.craiyon.com>

Lunch
(60 mins)

Joint Session - APA WG (APA WG, 60 mins)

Streamlining Data Integrity (Manu Sporny, 60 mins)



VCWG Data Integrity

August 2022 - A proposal for
streamlining crypto suites

Agenda

01
...

The Problems

Agility and proliferation

02
...

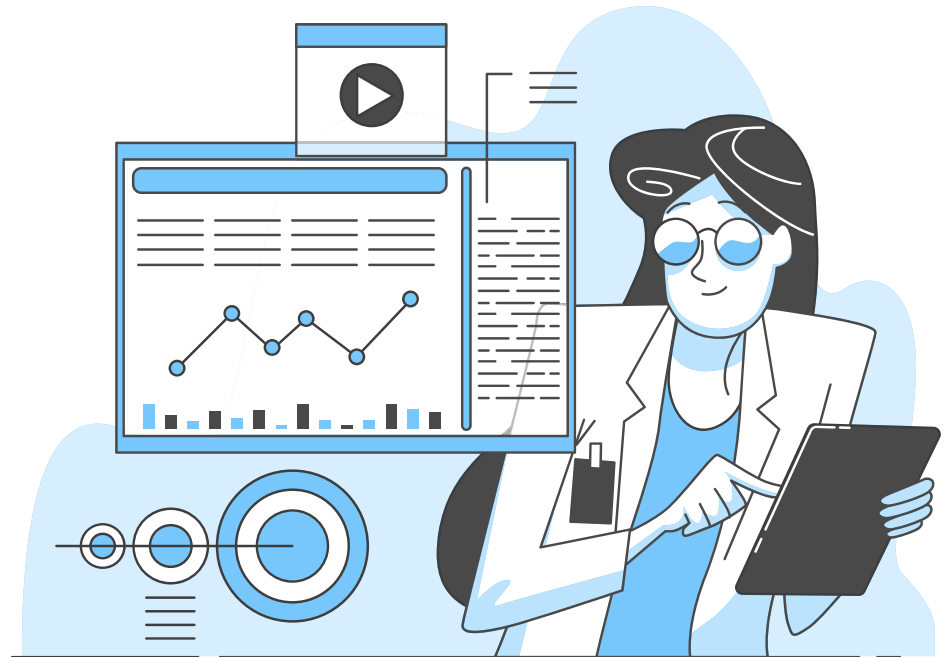
A Solution

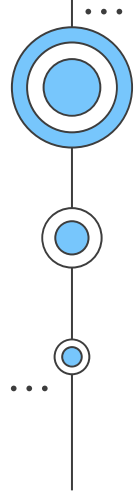
Simplification plan

03
...

Roadmap

Execution timeline





01

The Problems



The Problem (2020): Crypto Suite Proliferation

"Ok, let's be less coupled to the VC data model context and more agile.
Let's move crypto suite definitions into their own JSON-LD Contexts!"

```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://www.w3.org/2018/credentials/examples/v1",
    "https://w3id.org/security/suites/ed25519-2020/v1"
  ],
  "id": "http://example.edu/credentials/58473",
  "type": ["VerifiableCredential", "AlumniCredential"],
  ...
  "proof": {
    "type": "Ed25519Signature2020",
    "created": "2022-02-25T14:58:42Z",
    "verificationMethod": "https://example.edu/issuers/a#key-1",
    "proofPurpose": "assertionMethod",
    "proofValue": "z3FXQjecWufY46...UAUL5n2Brbx"
  }
}
```

New concern: "There are going to be soooo many crypto suites,
and they all have more or less the same properties!"

The Problem (2020): Crypto Suite Proliferation

How many crypto suites could there be? Well, there are at least this many today:

- <https://w3id.org/security/suites/ed25519-2020/v1>
- <https://w3id.org/security/suites/x25519-2019/v1>
- <https://w3id.org/security/suites/merkle-disclosure-2021/v1>
- <https://w3id.org/security/suites/secp256k1recovery-2020/v1>
- <https://w3id.org/security/suites/pgp-2021/v1>
- <https://w3id.org/security/suites/blockchain-2021/v1>
- <https://w3id.org/security/suites/jws-2020/v1>
- <https://w3id.org/security/suites/bls12381-2020/v1>
- <https://w3id.org/security/suites/eip712sig-2021/v1>
- <https://w3id.org/security/suites/secp256k1-2020/v1>
- <https://w3id.org/security/suites/secp256k1-2019/v1>
- <https://w3id.org/security/suites/merkle-2019/v1>
- <https://w3id.org/security/suites/chained-2021/v1>

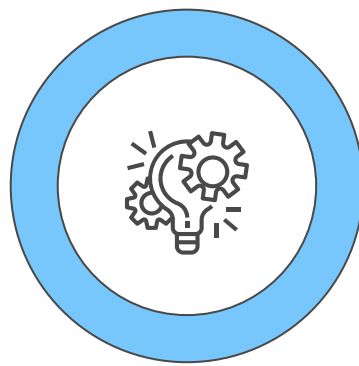
It's not terrible, and some of those are necessary, but most of them only differ by the crypto suite type that they define, such as `Ed25519Signature2020` or `JsonWebSignature2020`.



02

A Solution





A Solution

What if we define a base Data Integrity Proof type in the Verifiable Credentials v2 context that works for 80% of the crypto suites that we already have?

Since we only seem to be changing the crypto suite type in most crypto suites, if we shift that value to be a string, we can greatly reduce crypto suite proliferation.

This solution is backwards-compatible and does not preclude other more advanced crypto suites.

The Solution: A Backwards-Compatible Example

```
{
  "@context": [
    "https://www.w3.org/2022/credentials/v2",
    "https://www.w3.org/2022/credentials/examples/v2"
  ],
  "id": "http://example.edu/credentials/58473",
  "type": ["VerifiableCredential", "AlumniCredential"],
  ...
  "proof": {
    "type": "DataIntegrityProof",
    "cryptosuite": "eddsa-2022", <-- this is now a string value
    "created": "2022-02-25T14:58:42Z",
    "verificationMethod": "https://example.edu/issuers/a#key-1",
    "proofPurpose": "assertionMethod",
    "proofValue": "z3FXQjecWufY46...UAUL5n2Brbx"
  }
}
```

Other potential crypto suites: nist-ecdsa-2022, koblitz-ecdsa-2022, rsa-2022, pgp-2022, bbs-2022, eascdsa-2022, ibsa-2022, jws-2022, recommended-2022, selective-disclosure-2022, postquantum-2022, etc.



Downsides?

Semantic compression with CBOR-LD can't easily compress short, unique strings, so we become ~10-15 bytes less efficient per encoded signature.

...any other downsides?

...



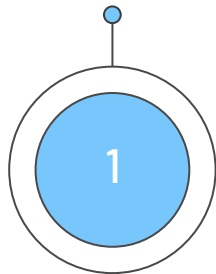
03

The Roadmap

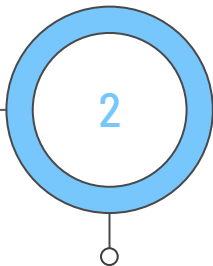


The Roadmap

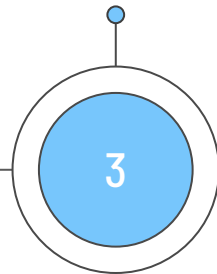
Define v2
DataIntegrityProof
(1 month)



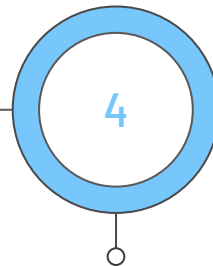
Demonstrate Implementation and
Test Suite
(3 months)

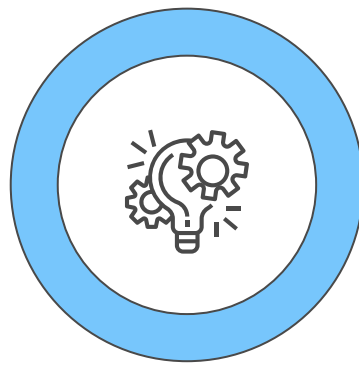


Call for Multiple
Implementations
(6 months)



Candidate Rec by June
2023





Future Data Integrity Work (for later discussion)

- The Multikey format
- Recommended, agile crypto suites

...



Discussion?

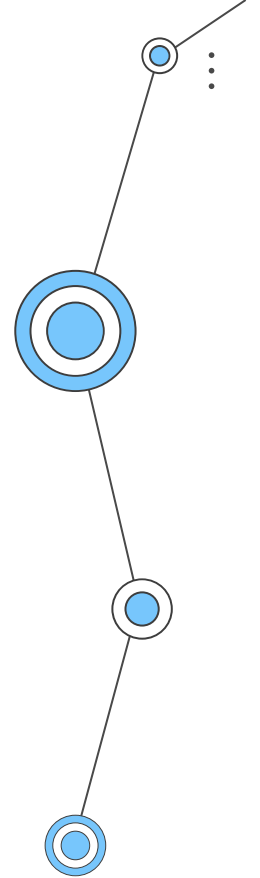
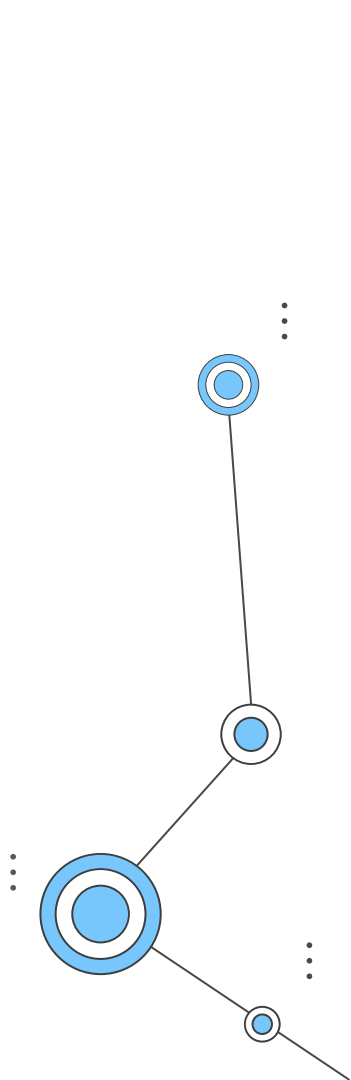
Credits

Do you have any questions?

msporny@digitalbazaar.com

<https://www.w3.org/2017/vc/WG/>

CREDITS: This presentation template was created by [Slidesgo](#), including icons by [Flaticon](#), infographics & images by [Freepik](#) and illustrations by [Stories](#)



Break
(15 mins)

VC Registries

(Ori Steele, Mike Jones, 60 mins)

What are Registries?

Many specifications define extension points

Registries are authoritative listings of items utilizing those extension points

A registry entry typically includes:

- Identifier for the item registered
- Short description of the item registered
- Reference to specification defining the registered item
- Who is authorized to update the registry entry

They may also contain other fields pertinent to the item registered

- Such as a status field

Example: IANA JSON Web Signature and Encryption Algorithms Registry

<https://www.iana.org/assignments/jose/jose.xhtml#web-signature-encryption-algorithms>

Two registry entries:

Algorithm Name	Algorithm Description	Algorithm Usage Location(s)	JOSE Implementation Requirements	Change Controller	Reference
ES256	ECDSA using P-256 and SHA-256	alg	Recommended+	[IESG]	[RFC7518, Section 3.4]
RSA-OAEP-384	RSA-OAEP using SHA-384 and MGF1 with SHA-384	alg	Optional	[W3C Web Cryptography Working Group]	[https://www.w3.org/TR/WebCryptoAPI/]

Why use Registries?

Registries enable *interoperable implementations of extensions*

- Registries reference authoritative definitions for each extension
- Avoids conflicting definitions for the same identifier
- Developers implement registered extensions using referenced definitions

For instance, enables distinct implementations of “RSA-OAEP-384” to interoperate

The IETF/IANA Registry Process

Internet Assigned Numbers Authority (IANA) administers registries for IETF

- Independent of working groups, IESG, etc.

Process:

- An RFC is created establishing a registry (e.g., JWA - RFC 7518)
 - The RFC defines registration criteria
 - The RFC populates the initial registry entries
- The IESG (the IETF area directors) selects Designated Experts
- Another spec requests registry of entries (via IANA Considerations §)
 - Typically can be any stable specification - not just an RFC
- Designated experts review request and approve/reject
- IANA adds new entries to the registry for approved requests

See [Guidelines for Writing an IANA Considerations Section in RFCs](#) [RFC 8126] for details

The W3C Registry Process

W3C enables designation of a specification of being a registry

W3C registry administered by a working group or community group

- No independent organization to objectively administer registries
- No designated experts
- No provision for registry administration beyond the lifetime of the WG or CG

Information above courtesy of Ivan Herman and Wendy Seltzer

For example, DID WG administers the DID Specification Registries

- <https://www.w3.org/TR/did-spec-registries/>

W3C Working Group using IANA Registries

WebAuthn WG chose to use IANA registries

- Created [Registries for Web Authentication \(WebAuthn\)](#) [RFC 8809] to establish the IANA registries that WebAuthn needs
- Registries at <https://www.iana.org/assignments/webauthn/webauthn.xhtml>
- Both WebAuthn specs and FIDO2 specs utilize these registries

We could choose to do the same

Registry Process GitHub Issue

Define policies for VC Extension Registry

- <https://github.com/w3c/vc-data-model/issues/909>

How do we “extend” verifiable credentials?

vc-data-model - <https://www.w3.org/2018/credentials>

vc-jwt - <https://www.iana.org/assignments/jose>

vc-data-integrity -

- <https://w3id.org/security/suites/jws-2020>

- <https://w3id.org/security/suites/ed25519-2020>

See also:

did - <https://www.w3.org/ns/did>

activity pub - <https://www.w3.org/ns/activitystreams>

web of things - <http://www.w3.org/ns/td>

Where do we “register” verifiable credential types?

- <https://w3id.org/traceability>
- <https://w3id.org/citizenship>
- <https://w3id.org/vaccination>
- <https://spec.rebase.xyz>
- <https://github.com/decentralized-identity/schema-directory>

What lessons did we learn from did spec registries?

- consider splitting registries up by “frequency of updates”.
- consider requiring / not requiring value judgment / impact statements for each registered item.
- consider the cost of registration for “multiple representations”.
- define the registration criteria in such a way that maintainers can operate autonomously

Intro to ACDCs (Sam Smith, 30 mins)

Authentic Chained Data Container ACDC

Verifiable Data Structures (Graphs)
that support
Authenticatable Attestations & Credentials

Samuel M. Smith Ph.D.
sam@prosapien.com
W3C TPAC 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 (property graph 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

```
{  
  "v": "ACDC10JSON00011c_",  
  "d": "EAdXt3gIXOf2BBWNHdSXCJnFJL5OuQPyM5K0neuniccM",  
  "u": "0ABghkDaG7OY1wjaDAE0qHcg",  
  "i": "did:keri:EBkPreYpZfFk66jpf3uFv7vklXKhzBrAqjsKAn2EDIPM",  
  "ri": "did:keri:ECmRy7xMwsxUelUauaXtMxTfPAMPAl6FkekwlOjkggt",  
  "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:EBkPreYpZfFk66jpf3uFv7vk1XKhzBrAqjsKAN2EDIPM",
  "ri":
"did:keri:ECmRy7xMwsxUelUauaXtMxTfPAMPAl6Fkekw10jkggt",
  "s": "ED6jrVPTz1SkUPqGGeIZ8a8FWS7a6s4reAXRZ0kogZ2A",
  "a": "EEveY4-9XgOcLxUderzwLlr9Bf7V_NHwY11kFrn9y2PY",
  "e": "EFH3dCdoFOLe71iheqcywJcnjtJtQIYPvAu6DZi13MOA",
  "r": "EG71iheqcywJcnjtJtQIYPvAu6DZi13MORH3dCdoFOLB"
```

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"
],
```

```
"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

```
{
  "a":
  {
    "d": "EgveY4-9XgOcLxUderzwLIr9Bf7V_NHwY1lkFrn9y2PY",
    "u": "0AwjaDAE0qHcgNghkDaG7OY1",
    "i": "did:keri:EpZfFk66jpf3uFv7vklXKhZBrAqjsKAn2EDIPmkPreYA",
    "score": 96,
    "name": "Jane Doe"
  }
}
```

Private Compact Variant

```
{
  "v": "ACDC10JSON00011c_",
  "d": "EAdXt3gIXOf2BBWNHdSXCJnFJL5OuQPmY5K0neuniccM",
  "u": "0ABghkDaG7OY1wjaDAE0qHcg",
  "i":
  "did:keri:EBkPreYpZfFk66jpf3uFv7vklXKhZBrAqjsKAn2EDIPM",
  "ri":
  "did:keri:ECmRy7xMwsxUelUauaXtMxTfPAMPAl6FkekwlOjkggt",
  "s": "ED6jrVPTz1SkUPqGGeIZ8a8FWS7a6s4reAXRZOkogZ2A",
  "a": "EEveY4-9XgOcLxUderzwLIr9Bf7V_NHwY1lkFrn9y2PY",
  "e": "EFH3dCdoFOLe71iheqcywJcnjttJtQIYPvAu6DZl13MOA",
  "r": "EG71iheqcywJcnjttJtQIYPvAu6DZl13MORH3dCdoFOLB"
}
```

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_NHwY11kFrn9y2PgveY4-9XgOcLxUdY",
    "boss":
    {
      "d": "EFy2PgveY4-9XgOcLxUdYerzwLIr9Bf7V_NHwY11kFrn",
      "n": "EIl3MORH3dCdoFOLe71iheqcywJcnjtJtQIYPvAu6DZA",
      "s": "EFOLe71iheqcywJcnjtJtQIYPvAu6DZAIl3MORH3dCdo",
    }
  }
}
```

Nested Edge Section with Operators

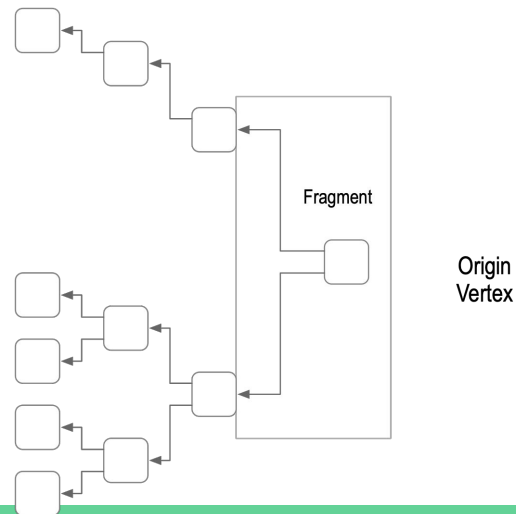
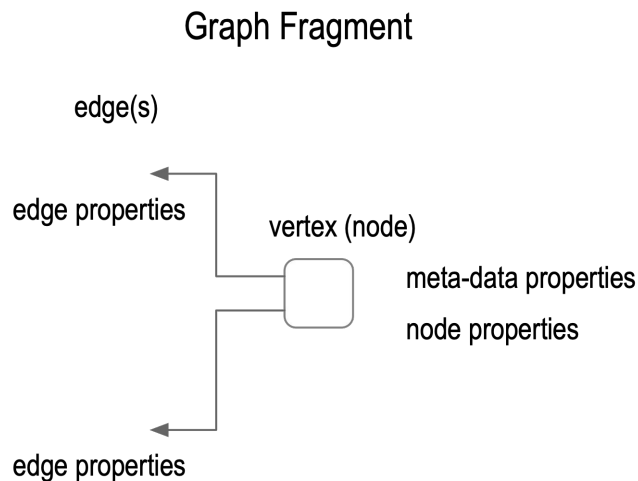
```
{
  "e":
  {
    "d": "EBrzwLIr9Bf7V_NHwY11kFrn9y2PgveY4-9XgOcLx,UdY",
    "o": "AND",
    "boss":
    {
      "n": "EGl3MORH3dCdoFOLe71iheqcywJcnjtJtQIYPvAu6DZA",
      "o": ["NI2I", "NOT"]
    },
    "baby":
    {
      "n": "EMRH3dCdoFOLe71iheqcywJcnjtJtQIYPvAu6DZAI13A",
      "o": "I2I"
    },
    "food":
    {
      "o": "OR",
      "plum":
      {
        "n": "EHIYPvAu6DZAI13AORH3dCdoFOLe71iheqcywJcnjtJt",
        "o": "NI2I"
      },
      "pear":
      {
        "n": "ECTQIYPvAu6DZAI13AORH3dCdoFOLe71iheqcywJcnjt",
        "o": "NI2I"
      }
    }
  }
}
```

ACDC Normative 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: "ECmRy7xMwsxUelUaaaXtMxTfPAMPAl6FkekwlOjkggt"
s	Schema	Either the SAID in CESR format block in CESR format or the block itself: "ED6jrVPTz1SkUPqGGeIZ8a8FWS7a6s4reAXRZOkogZ2A"
a	Attribute	Either the SAID of a block of attributes in CESR format or the block itself in CESR format: "EEveY4-9XgOcLxUderzwLIr9Bf7V_NHwY11kFrn9y2PY"
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_NHwY11kFrn9y2PY"
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 Bioridian 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) 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**



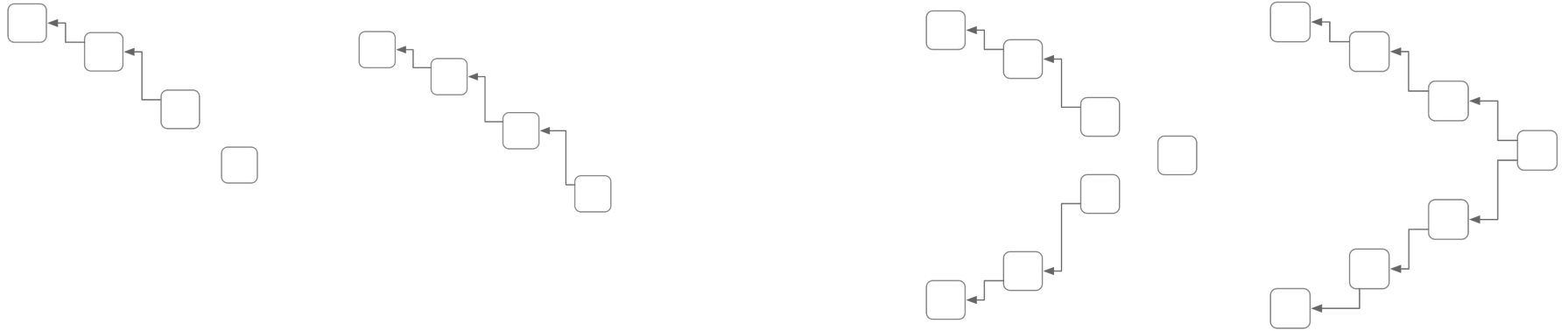
Chained ACDCs (graphs) Enable

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

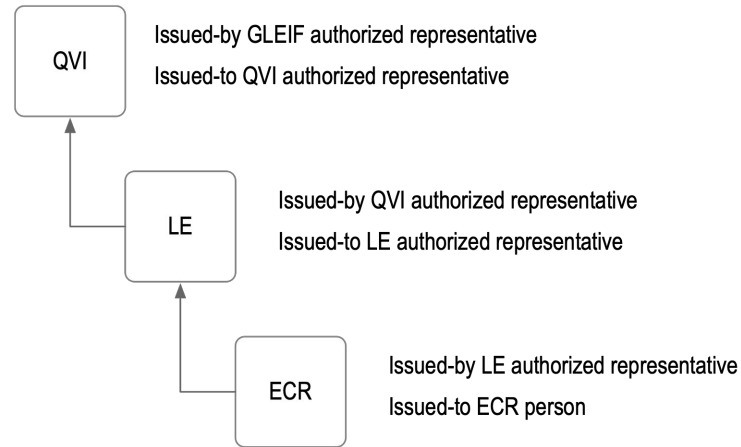


GLEIF vLEI Credential Example

Qualified vLEI Issuer (QVI) Credential

Legal Entity (LE) Credential

Engagement Context Role (ECR) Credential

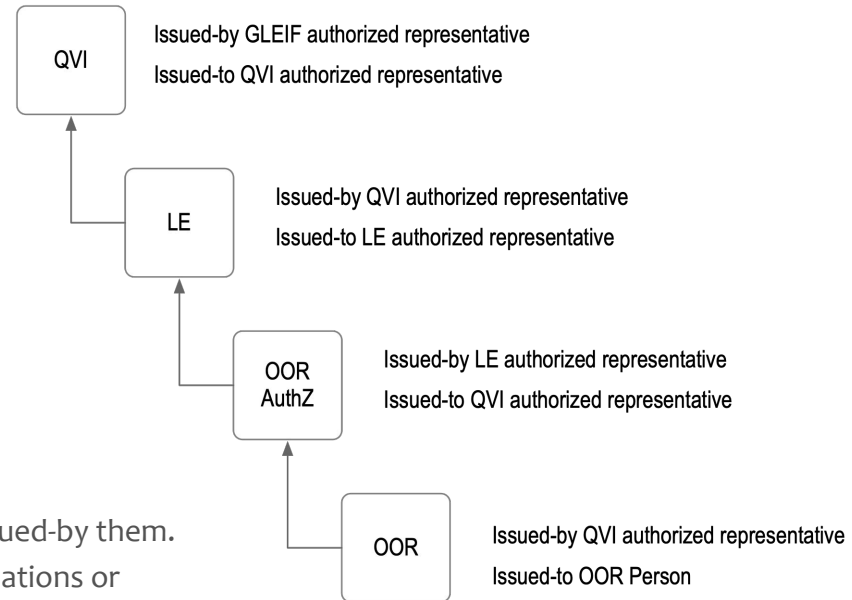


Qualified vLEI Issuer (QVI) Credential

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

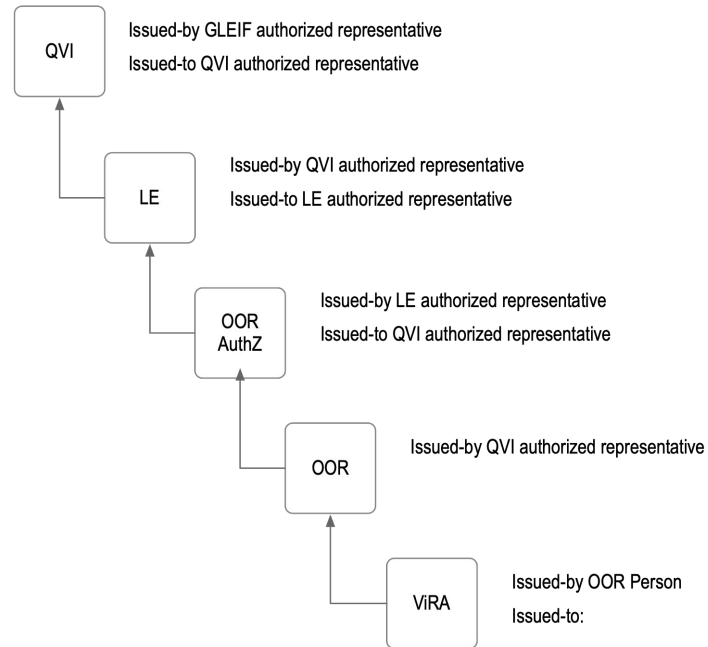
Qualified vLEI Issuer (QVI) Credential

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_salpShx6vOCnseruledVPeNvRaQm6HrmMI"
          },
        },
      },
    },
  ],
}
```

```
"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).

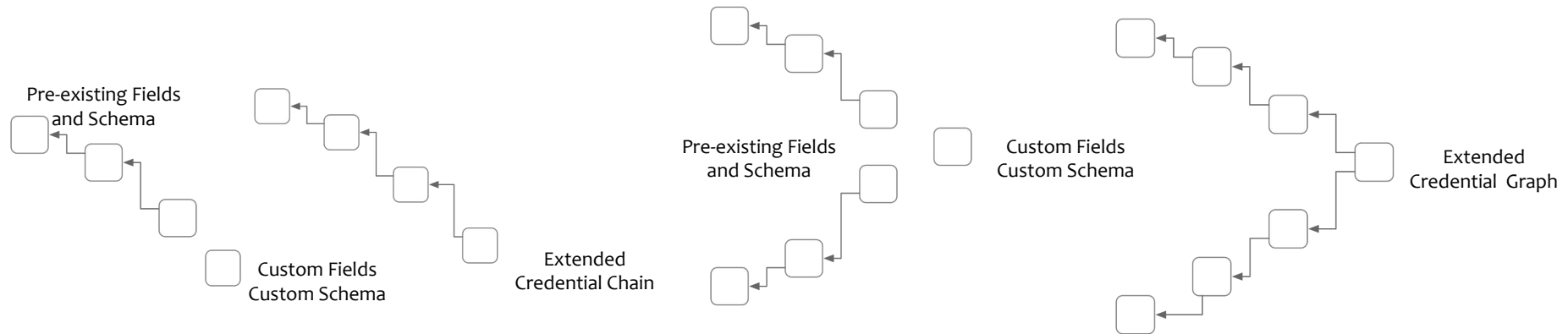
No need for centralized permissioned name-space registries to resolve name-space collisions.

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

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?

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

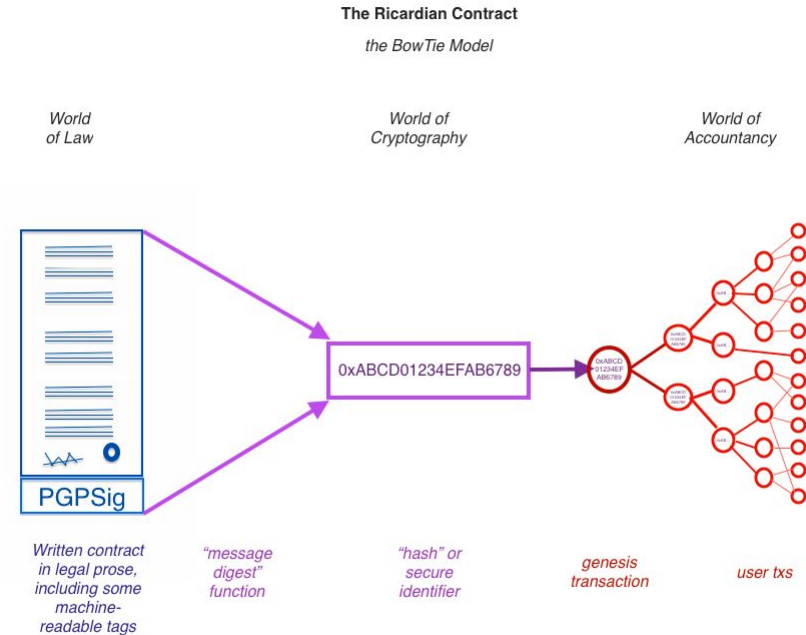
Ricardian Contracts:

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.

IPEX: Issuance & Presentation Exchange protocol

Presentation Exchange:

An exchange that provides disclosure of one or more ACDCs between a *Discloser* and a *Disclosee*.

A presentation exchange is the process by which authenticatable information may be exchanged between two parties, namely, the *Discloser* and *Disclosee*.

ACDC:

Type of data as issuance concretely defined by the ACDC specification.

Discloser:

An ACDC in a disclosure *is disclosed by* the *Discloser*.

Disclosee:

An ACDC in a disclosure *is disclosed to* the *Disclosee*.

Issuer:

An ACDC *is issued by* the *Issuer*. The *Issuer* identifier (AID) appears in the top level of the ACDC.

Issuee:

An ACDC is optionally *issued to* the *Issuee*. When present, the *Issuee* identifier (AID) appears at the top level of the attribute section or in the attribute list at the top level of the attribute aggregate section of the ACDC.

Each ACDC MUST have an *Issuer* and MAY have an *Issuee*

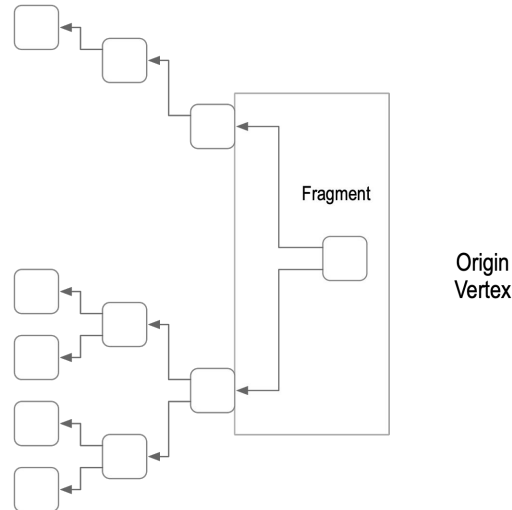
IPEX: Issuance & Presentation Exchange protocol

The set of ACDCs so disclosed in a presentation exchange MUST be chained.

This set of chained ACDCs define a directed acyclic graph (DAG) that MUST have at least one vertex and MAY have zero or more edges pointing to other vertices.

Each ACDC itself defines a graph fragment consisting of one vertex and zero or more directed edges.

Each directed edge contained in an ACDC points to a vertex contained in another ACDC. The ACDC that contains the origin vertex of the DAG is called the *origin* or *primary* ACDC of the presentation exchange.



IPEX: Issuance & Presentation Exchange protocol

The disclosure performed by a presentation exchange MAY be graduated and/or MAY be contractually protected.

Issuance Exchange:

A special case of a presentation exchange where the Discloser is the *Issuer* of the *origin (Primary)* ACDC of the DAG formed by the set of chained ACDCs so disclosed.

In an issuance exchange, when the *origin* ACDC has an Issuee, the *Disclosee* MAY also be the *origin (Primary)* ACDC's *Issuee*.

The Issuer MUST provide a signature on the SAID of the most compact variant defined by the schema of the ACDC. When more than one variant is defined by the schema via the oneOf composition operator for any top-level field, the most compact variant MUST appear as the first entry in the oneOf list. When only one variant of each top-level field is defined by the schema, that variant is therefore by definition the most compact variant.

ACDC analogy to Merkle Tree

The different variants of an ACDC form a hash tree (using SAIDs) that is analogous to a Merkle Tree.

Signing the top-level SAID of the compact version of the ACDC is equivalent to signing the Merkle Root of a Merkle Tree.

Different variants of an ACDC (SADs with SAIDs) correspond to different paths through a Merkle 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 equivalent to a Merkle Tree proof of inclusion along a path in the Merkle Tree down from its Root.

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

Proof-of-Issuance Proof-of-Disclosure

An Issuer MAY provide signatures of the SAIDS of other variants, as well as signatures of the SADs of other variants.

Proof of issuance is provided by disclosing the SAID of the most compact variant and a reference to the SEAL anchoring that SAID in either the KEL or TEL of the issuer.

Proof of disclosure is provided by disclosing the SAD of the most compact variant and then recursively disclosing the nested SADs of each of the top level sections of the most compact variant as needed for the promised disclosure.

Thus for any and all disclosed variants of an ACDC, the Disclosee need only verify the same proof of issuance as defined above and may need to verify the specific proof of disclosure for the given disclosed variant as defined above.

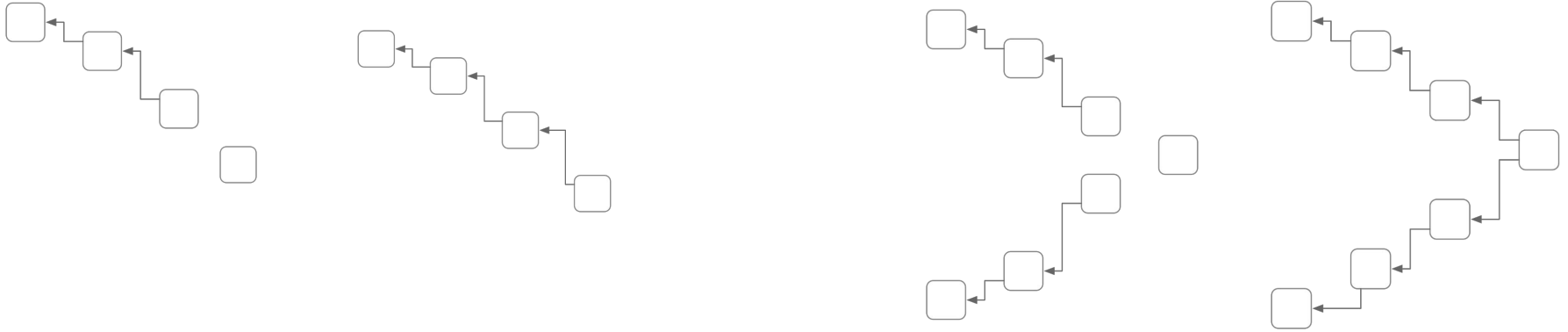
Append-to-Extend

Append-only verifiable data structures have strong security properties

Append-only simplifies end-verification

Append-only provides permission-less extensibility

Interoperable permissioned name-space registries are no longer needed with append-to-extend



Chain-Link Confidentiality

Disclosures via Presentations Exchanges may be contractually protected by Chain-Link Confidentiality (i.e a Chain-Link Confidential disclosure).

The chaining in this case is different from the chaining described above between Issuances in a DAG of chained Issuances. Chain-link confidentiality, in contrast, chains together a sequence of Disclosees.

Each Disclosee in the sequence in turn is the Discloser to the next Disclosee.

The terms-of-use of the original disclosure as applied to the original Disclosee **MUST** be applied by each subsequent Discloser to each subsequent Disclosee via each of the subsequent disclosures (presentation exchanges).

These terms-of-use typically constrain disclosure to only approved parties, i.e. imbue the chain of disclosures with some degree of confidentiality. These terms-of-use are meant to contractually protect the data rights of the original Issuer or Issuee of the data being disclosed.

Endorsers vs. Issuers

The ACDC Issuer is the securely attributable source of the ACDC. (Secure Attribution) that is designated in the Issuer field of the ACDC. The Issuer AID may be controlled by by set of controllers given multiple key-pairs with a threshold. Proof of Issuance requires the threshold be met.

An ACDC Endorser merely lends credibility (reputation) to an ACDC by signing the ACDC. The Endorser has its own AID with its own set set of controllers. The Endorser AID is not included in the ACDC.

Essentially a disclosure by the Discloser of an ACDC where the discloser is not the Issuer makes the Discloser by virtue of signing the disclosure, a type of Endorser. However in such a disclosure the purpose of the signing by the Discloser may be more than an endorsement such as to make a commitment with respect to a contractually protected disclosure.

When the Discloser is the Issuee then the signing of the Disclosure is not merely an endorsement but provides a temporal proof of control over the Issuee AID and could satisfy a “live” presentation requirement by the Disclosee.

Uncompacted Public Attribute Section

```
{
  "a":
  {
    "d": "EgveY4-9XgOcLxUderzwLlr9Bf7V_NHwY1lkFrn9y2PY",
    "i": "did:keri:EpZfFk66jpf3uFv7vklXKhzBrAqjsKAn2EDIPmkPreYA",
    "score": 96,
    "name": "Jane Doe"
  }
}
```

Public Compact Variant

```
{
  "v": "ACDC10JSON00011c_",
  "d": "EAdXt3gIXOf2BBWNHdSXCJnFJL5OuQPyM5K0neuniccM",
  "i":
"did:keri:EBkPreYpZfFk66jpf3uFv7vklXKhzBrAqjsKAn2EDIPM",
  "ri":
"did:keri:ECmRy7xMwsxUelUauaXtMxTfPAMPAl6Fkekwl0jkggt",
  "s": "ED6jrVPTz1SkUPqGGeIZ8a8FWS7a6s4reAXRZOkogZ2A",
  "a": "EEveY4-9XgOcLxUderzwLlr9Bf7V_NHwY1lkFrn9y2PY",
  "e": "EFH3dCdoFOLe71iheqcywJcnjtJtQIYPvAu6DZIl3MOA",
  "r": "EG71iheqcywJcnjtJtQIYPvAu6DZIl3MORH3dCdoFOLB",
}
```

Composed JSON Schema

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

Attribute Section

```
{  
  "a":  
    {  
      "d": "EgveY4-9XgOcLxUderzwLIr9Bf7V_NHwY1lkFrn9y2PY",  
      "u": "0AwjaDAE0qHcgNghkDaG7OY1",  
      "i": "did:keri:EpZfFk66jpf3uFv7vklXKhzBrAqjsKAn2EDIPmkPreYA",  
      "score": 96,  
      "name": "Jane Doe"  
    }  
}
```

Edge Section

```
{
  "e":
  {
    "d": "EerzwLlr9Bf7V_NHwY1lkFrn9y2PgveY4-9XgOcLxUdY",
    "boss":
    {
      "d": "E9y2PgveY4-9XgOcLxUdYerzwLlr9Bf7V_NHwY1lkFrn",
      "n": "EIl3MORH3dCdoFOLe71iheqcywJcnjtJtQIYPvAu6DZA",
      "w": "high"
    }
  }
}
```

Latent Accountability

Escrow

KYC

Contingent Enforcement and Recourse

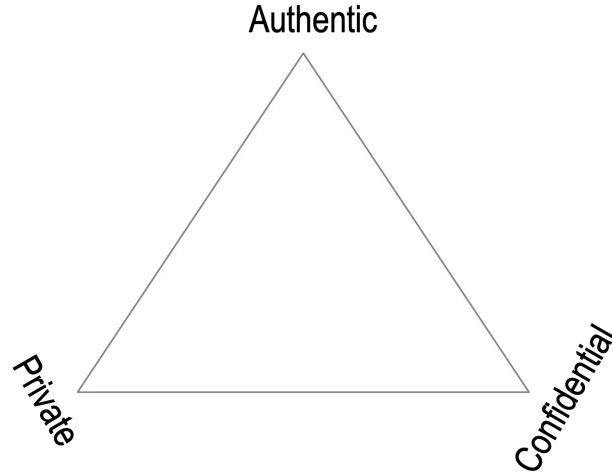
Bonding

Bounties

Example Use Cases

PAC Theorem

A conversation may be two of the three: *private*, *authentic*, and *confidential* to the same degree, but not all three at the same degree.



Trade-offs
required!

Definitions

Private:

The parties to a conversation are only known by the parties to that conversation.

Authentic:

The origin and content of any statement by a party to a conversation is provable to any other party.

Confidential:

All statements in a conversation are only known by the parties to that conversation.

Privacy:

about control over the disclosure of who participated in the conversation (non-content meta-data)

Authenticity:

about proving who said what in the conversation (secure attribution)

Confidentiality:

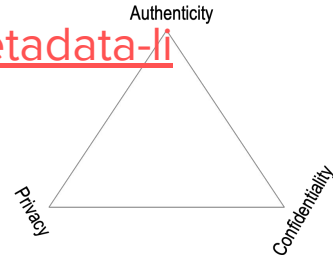
about control over the disclosure of what was said in the conversation (content data)

Relatively weak legal protection for non-content (supoena)

Relatively strong legal protection for content (search warrant)

<https://www.lawfareblog.com/relative-vs-absolute-approaches-contentmetadata-line>

<https://www.pogo.org/analysis/2019/06/the-history-and-future-of-mass-metadata-surveillance/>



Proving Authenticity

Non-repudiable Proof:

a statement's author cannot successfully dispute its authorship

Asymmetric key-pair digital signature

Repudiable Proof:

a statement's author can successfully dispute its authorship

DH shared symmetric key-pair encryption (auth crypt)

Shared secret makes every verifier a potential forger

Trade-offs

Private:

The parties to a conversation are only known by the parties to that conversation.

Authentic:

The origin and content of any statement by a party to a conversation is provable to any other party.

Confidential:

All statements in a conversation are only known by the parties to that conversation.

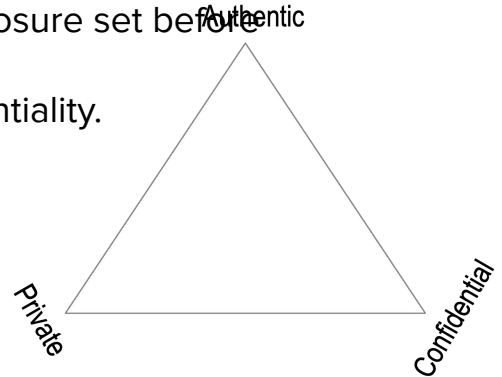
Non-repudiation means any party to conversation can prove to any other party exactly what was said by whom.

This means that technologically there is no way to prevent disclosure by any party to some third party. We can incentivize confidentiality by imposing a liability on the parties to the disclosure set before disclosure occurs.

Enforcement of that liability will usually necessarily violate privacy but not confidentiality.

Real world value often requires transitivity.

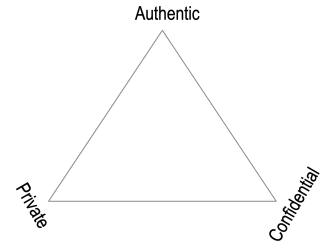
Transitive value transfer will violate complete privacy.



Layering

A communication system can layer the different properties in different orders thereby imposing a priority on each property.

Authenticity
Confidentiality
Privacy



Why Solve the Secure Attribution Problem

Secure attribution of any communication to its source

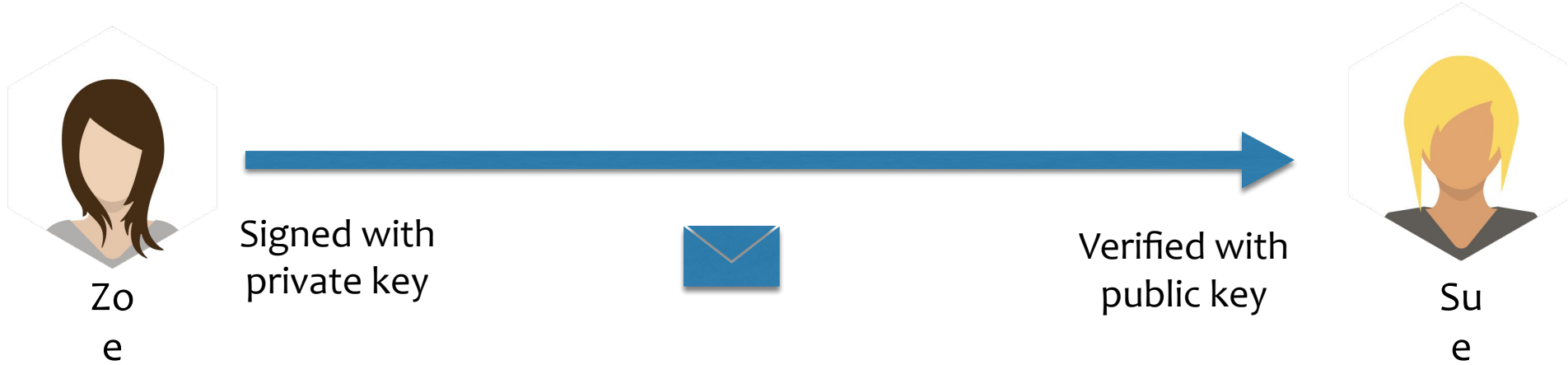
Authentic communication

Authentic interactions based on secure attribution of all statements by participants:

Data Provenance (Verifiable authenticity of data)

Data Supply Chains (Authentic data economy)

Non-Repudiable Authenticity



Non-repudiable authenticity is *zero-trust*

Repudiable Authenticity



Zo
e

Encrypted with
shared private key



Decrypted with
shared private key



Su
e

Repudiable authenticity requires trust (is not zero-trust)

Non-Repudiable Authenticity Is Legally Binding.
Repudiable Authenticity Is Not Legally Binding.



Zo
e

Encrypted with
shared private
key



Decrypted with
shared private
key



Su
e

Non-Repudiable authenticity has recourse.
Best fits current business and regulatory eco-systems.

Zero Knowledge Proof?



Zo

Authentic ZKP: Is the information proven in a repudiable or non-repudiable manner?



ZK



Su

e

one party (the prover) can prove to another party (the verifier) that a given statement is true, without conveying any information apart from the fact that the statement is indeed true.

ZKPoK (selective disclosure)

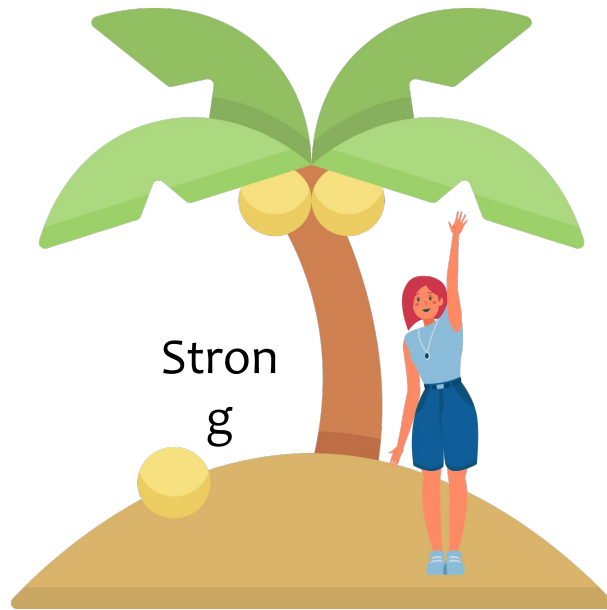
Zoe proves to Sue that a given statement is true without enabling Sue to prove to a third party that the statement is true and protecting Alice from Sue as a forger.

(plausible deniability, repudiability with forgery protection)

ZKP (selective disclosure)

Zoe proves an element of a bundle of information to Sue without disclosing any other element of the bundle (non-repudiable or repudiable but if repudiable may not be protected from forgery)

Other “non-ZKP” like methods can perform a non-repudiable selective disclosure



Stron
g

Privacy?



Wea
k



Wea
k

Strong Privacy

Definition: un-correlated interactions over unbounded time and space.

Super aggregators and state actors have effectively unlimited storage and compute capacity. Eventually all disclosed data will be at least statistically correlatable.

Weak Privacy

Definition: un-correlated interactions over bounded time and space.

when the cost of correlation exceeds the value of correlation the data will be un-correlated.

Operating Regimes

Political

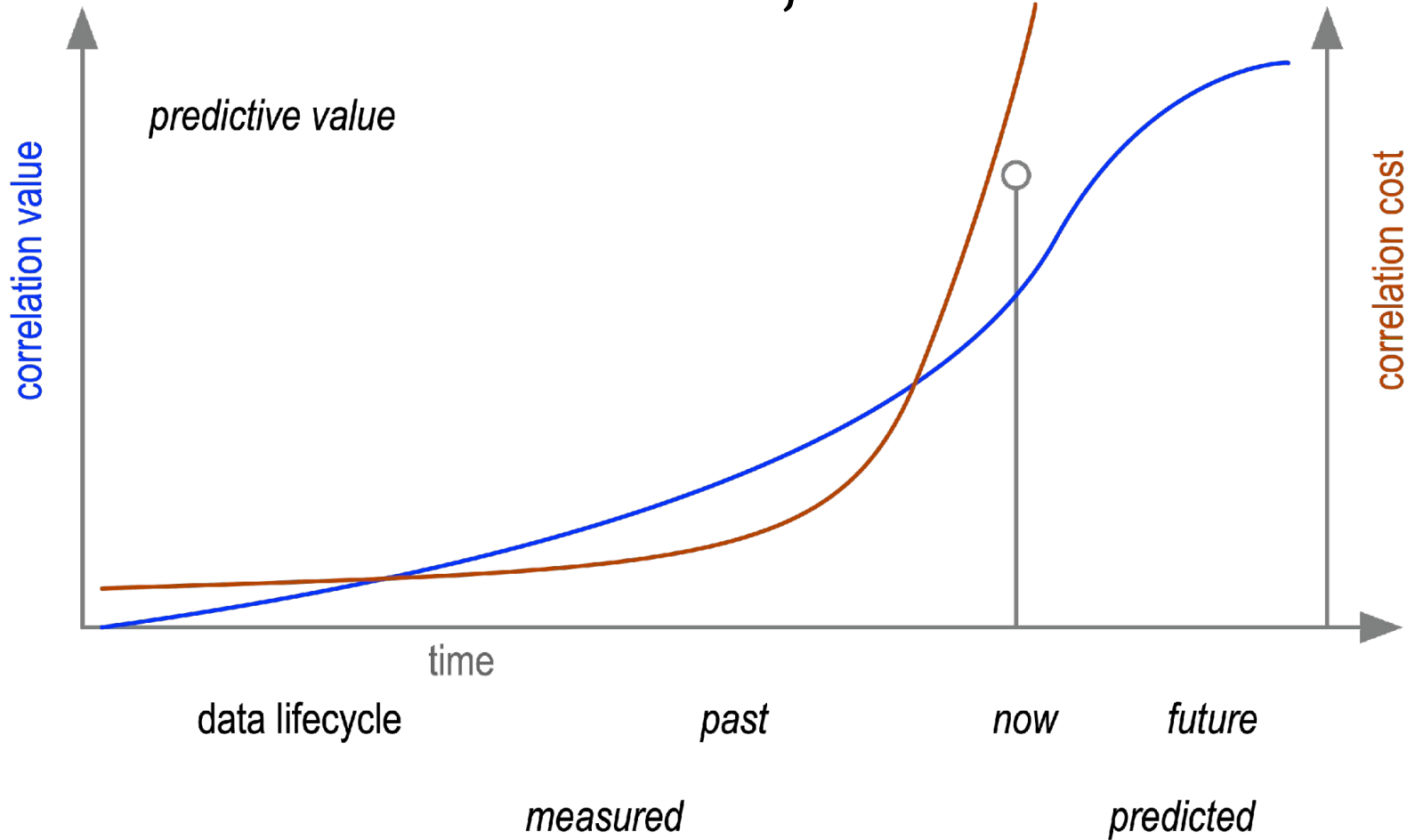
Legitimate	Hide or Bribe	Regulation and/or Legally Enforced Contracts
Illegitimate	Hide and Bribe	Criminally Enforced Contracts ?

Illegitimate

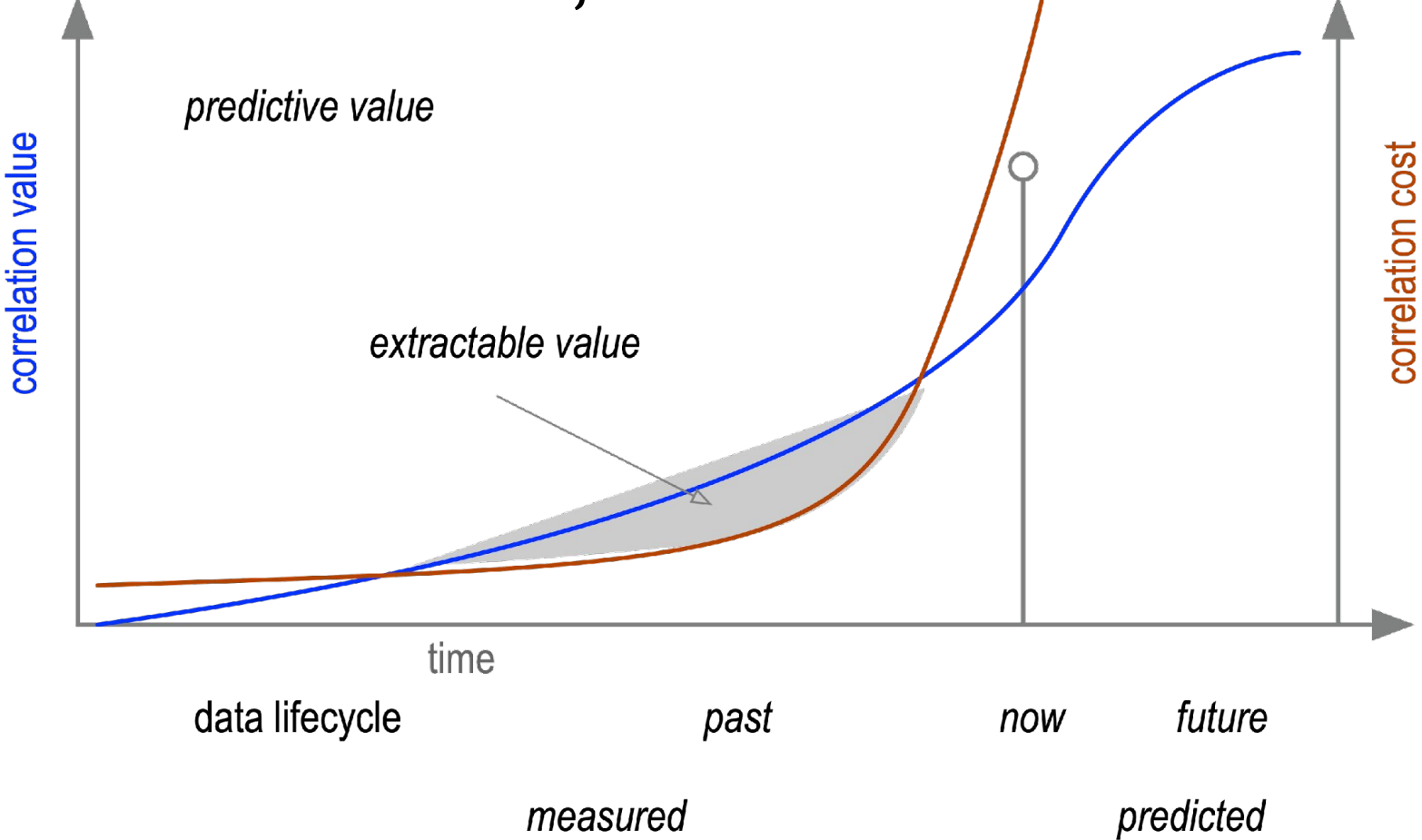
Legitimate

Economic

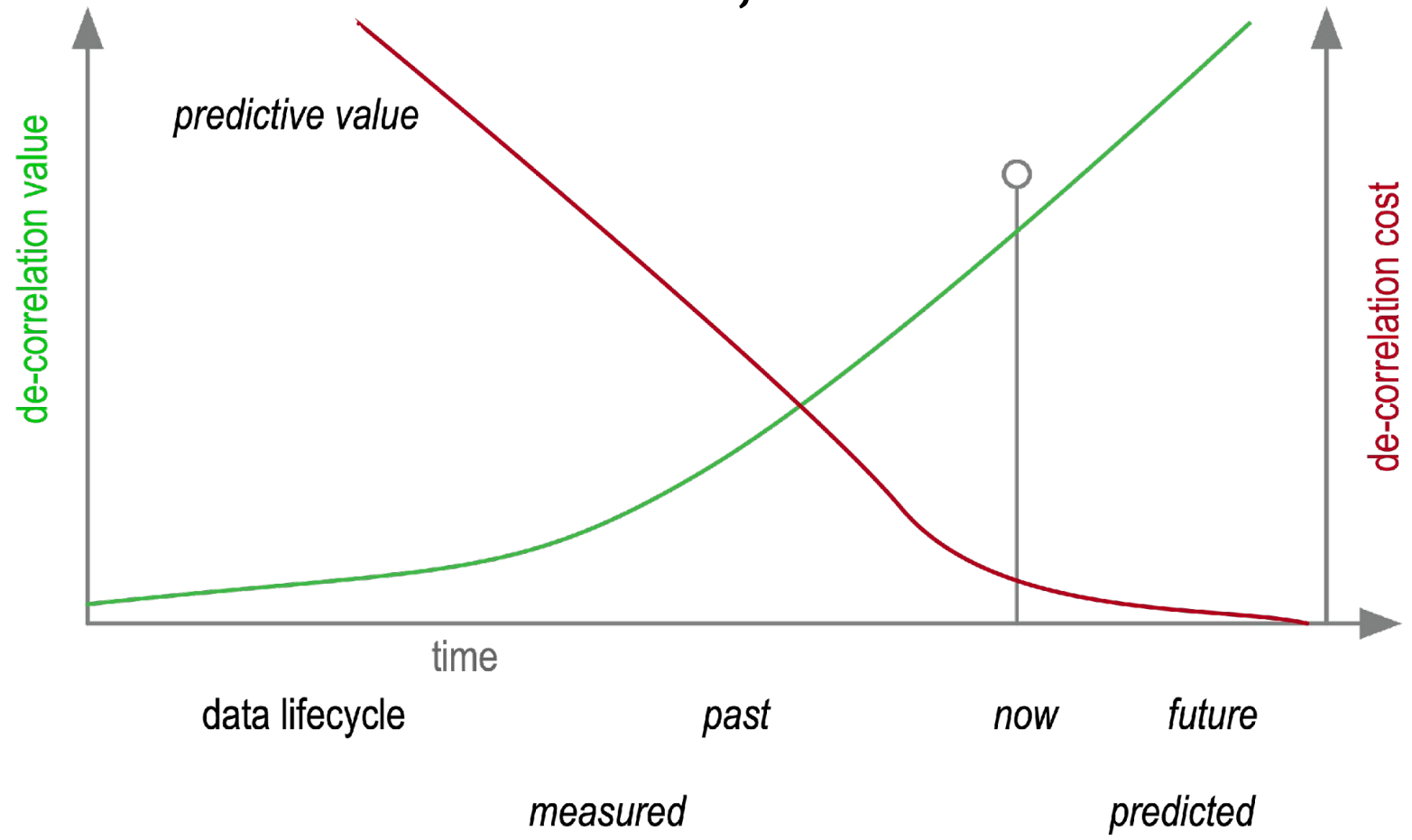
Economics of Correlator



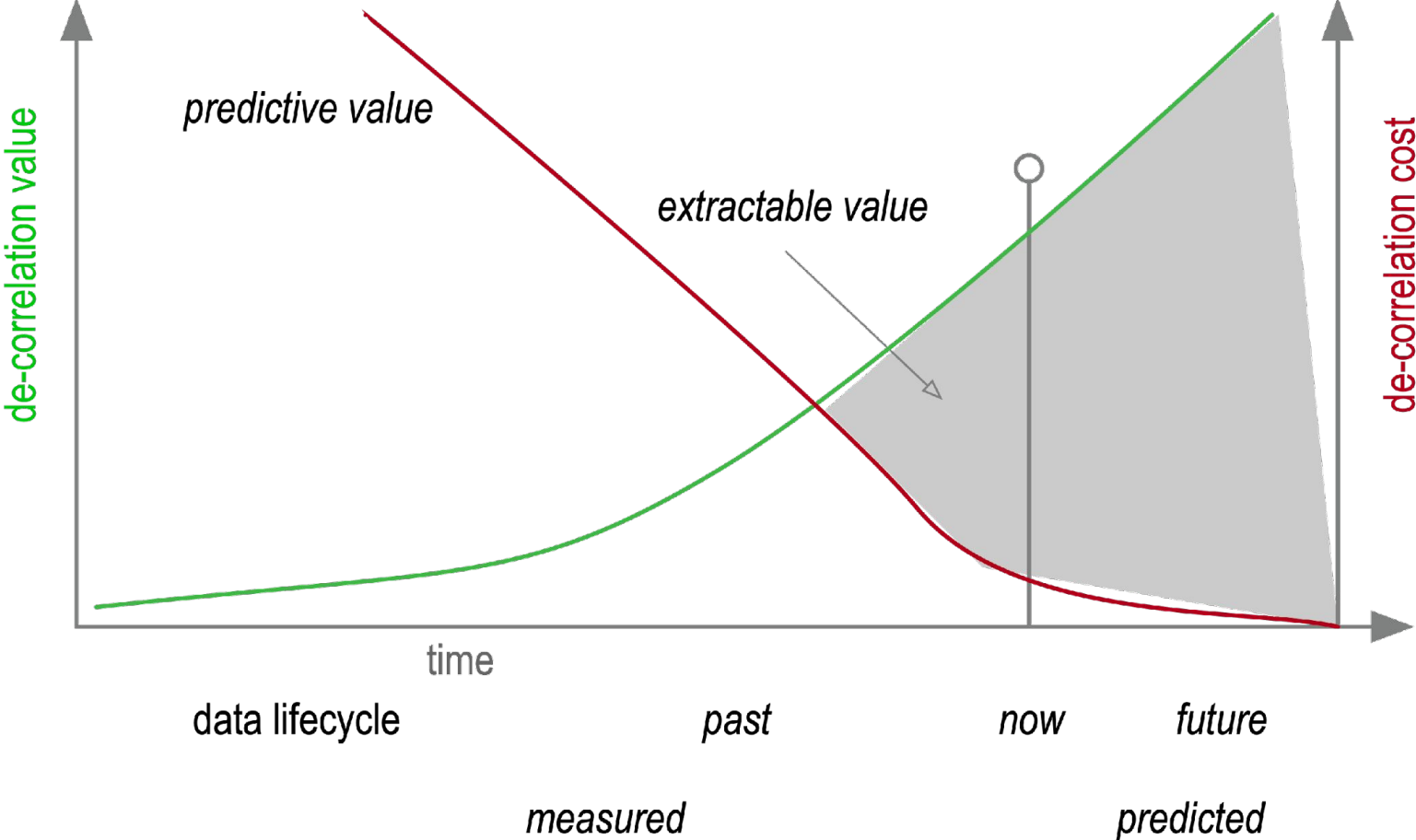
Economics of Correlator: Value Extraction



Economics of De-correlator



Economics of De-correlator: Value Extraction



Freedom

balanced

Freedom from ...

exploitation (commercial)

intimidation (political)

censorship (political)

Freedom to ...

extract value (commercial)

build relationships (social)

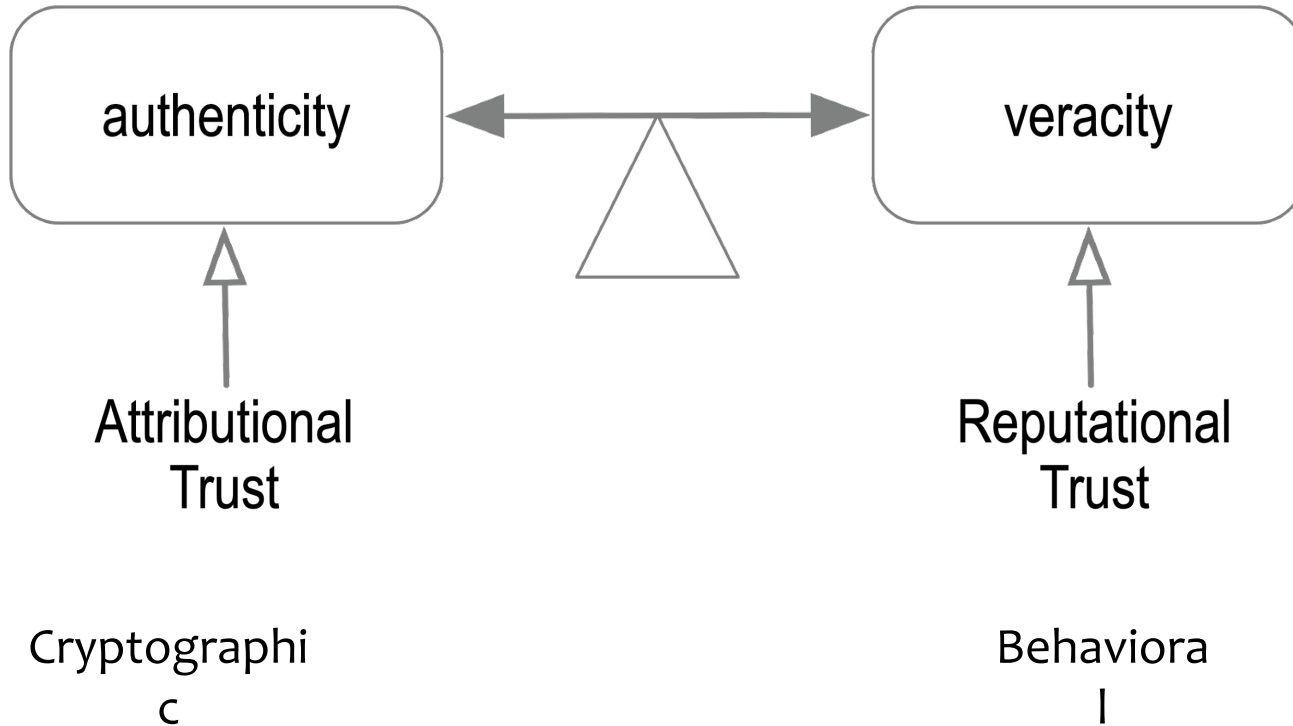
build community (political)

possibility of erasure = possibility of censorship

anonymity = loss-of-value from attribution

fairness = loss of privacy from attribution

Trust Balance



Unified Identifier Model

AID: Autonomic Identifier (primary)

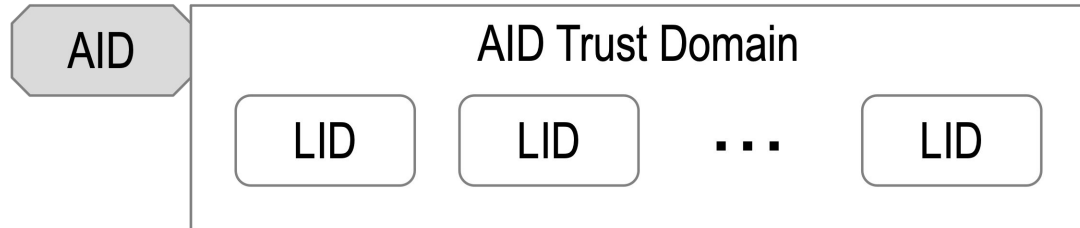
self-managing self-certifying identifier with cryptographic root of trust
secure, decentralized, portable, universally unique

LID: Legitimized Human Meaningful Identifier (secondary)

legitimized within trust domain of given AID by a verifiable authorization from AID controller

authorization is verifiable to the root-of-trust of AID

Forms $AID|LID$ couplet within trust domain of AID



AID|LID Couplet

625.127C125r

EXq5YqaL6L48pf0fu7IUhL0JRaU2_RxFP0AL43wYn148 | 625.127C125r

Background

BADA (Best Available Data Acceptance) Policy

Authentic Data:

Two primary attacks:

Replay attack:

Mitigation: Monotonicity

Deletion attack:

Mitigation: Redundancy

Replay Monotonicity:

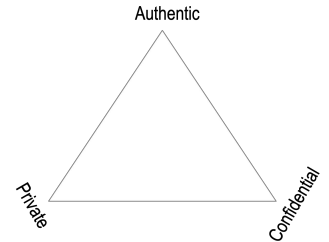
Interactive:

Nonce

Non-interactive:

Memory (sequence number, date-time stamp, nullification)

More scalable



RUN off the CRUD

Client-Server API or Peer-to-Peer.

Create, Read, Update, Delete (CRUD)

Read, Update, Nullify (RUN)

Decentralized control means server never creates only client. Client (Peer) updates server (other Peer) always for data sourced by Client (Peer). So no Create.

Non-interactive monotonicity means we can't ever delete.

So no Delete. We must Nullify instead. Nullify is a special type of Update.

Ways to Nullify:

- null value

- flag indicating nullified

Rules for Update : (anchored to key state in KEL)

- Accept if no prior record.

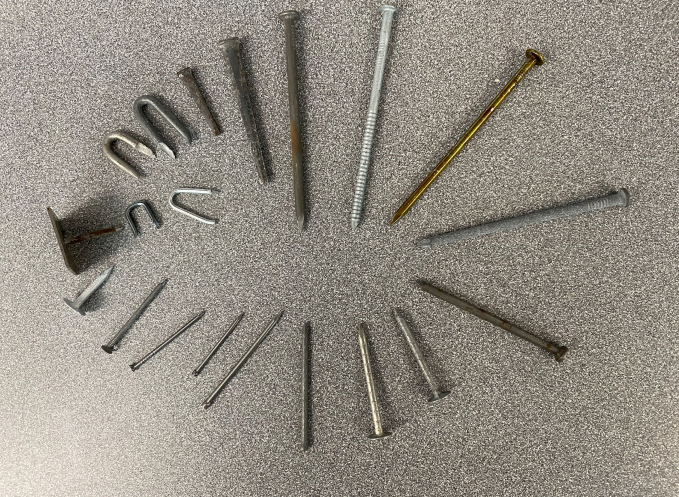
- Accept if anchor is later than prior record.

Rules for Update: (signed by keys given by key state in KEL, ephemeral identifiers have constant key state)

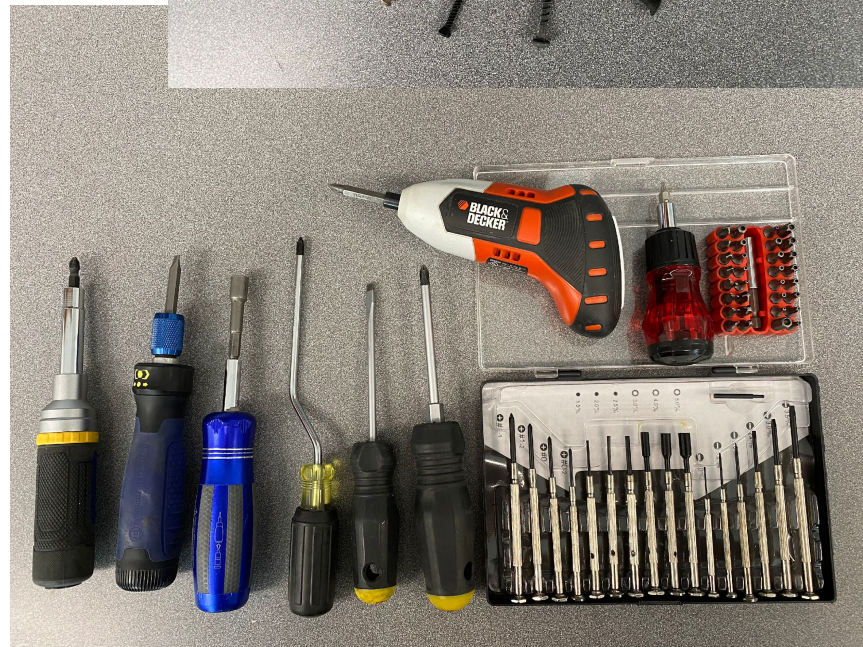
- Accept if no prior record.

- Accept if key state is later than prior record.

- Accept if key state is the same and date-time stamp is later than prior record.



Toolkits



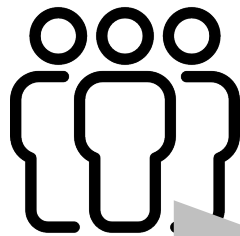
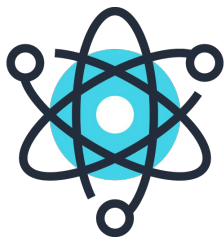
Only have one set of tools for
truly secure data control!

Entropy Derived Tools

Cryptographic one-way functions ...

hashes, ECC scalar multiplication...
digital signatures, ZKPs ...

its

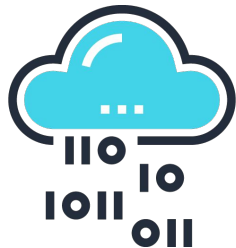


control



correlation

bits



cryptographic
pseudonymous identifiers



Tripartite Authentic Data (VC) Model

Issuer: Source of the VC. Creates (issues) and signs VC

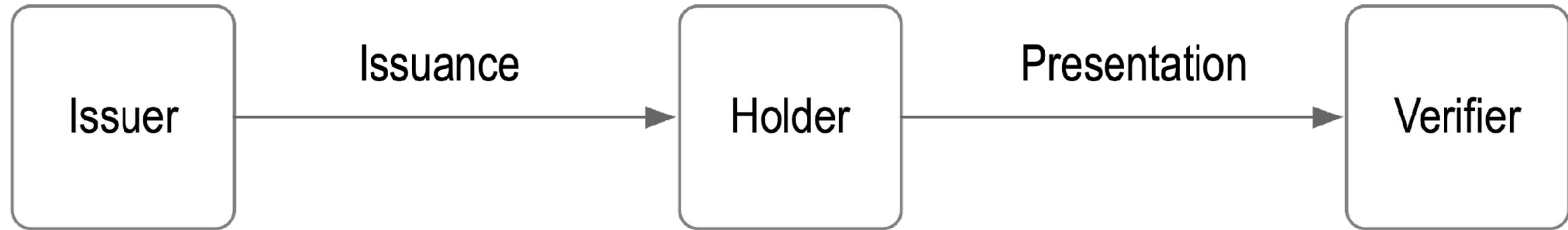
Holder: Usually the target of the VC. The holder is the “*issuée*” that receives the VC and holds it for its own use.

Verifier: Verifies the signatures on the VC and authenticates the holder at the time of presentation

The issuer and target each have a DID (decentralized identifier).

The DIDs are used to look-up the public key(s) needed to verify signatures.

Issuer-Holder-Verifier Model



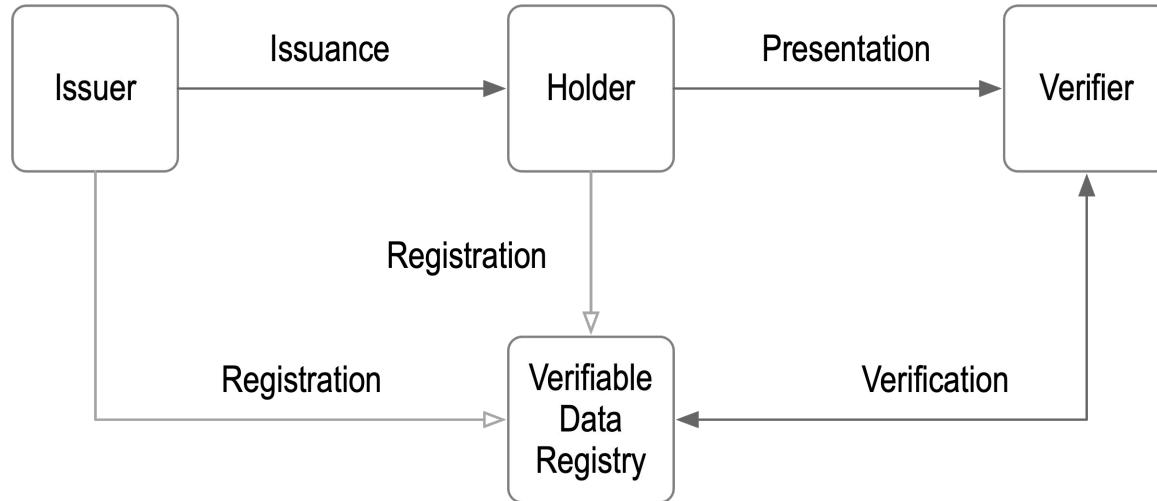
Tripartite Authentic Data (VC) Model with VDR

Verifiable Data Registry (VDR) enables decentralized but interoperable discovery and verification of authoritative key pairs for DIDs in order to verify the signatures on VCs. A VDR may also provide other information such as data schema or revocation state of a VC.

Each controller of a DID registers that DID on a VDR so that a verifier can determine the authoritative key pairs for any signatures.

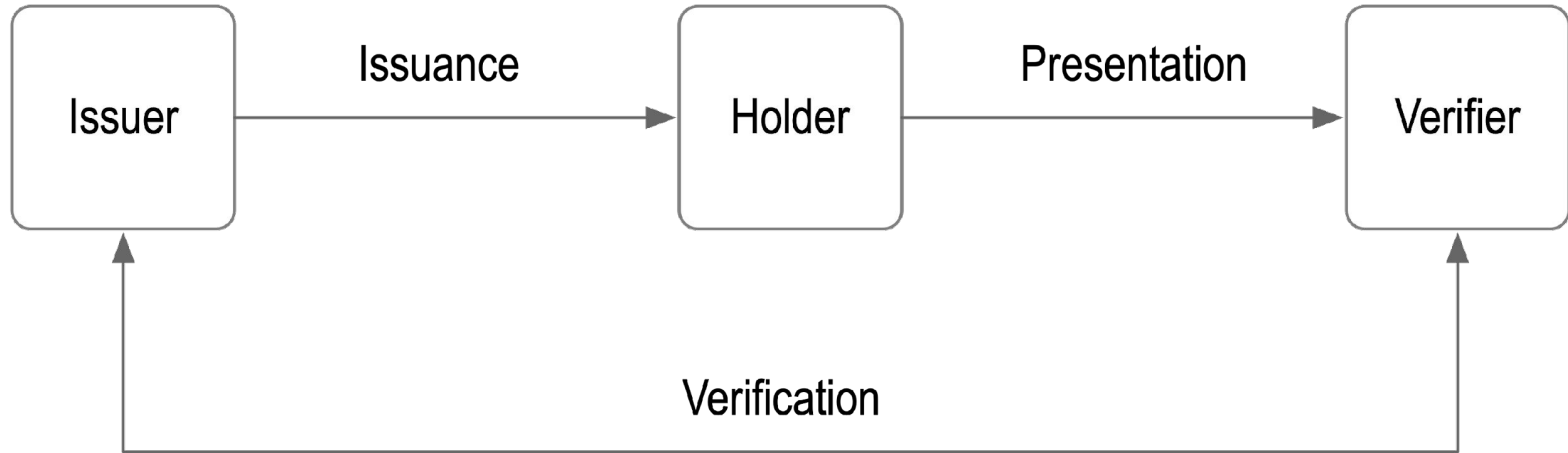
We call this determination, *establishment of control authority* over a DID.

Issuer-Holder-Verifier Model with Verification at Verifiable Data Registry



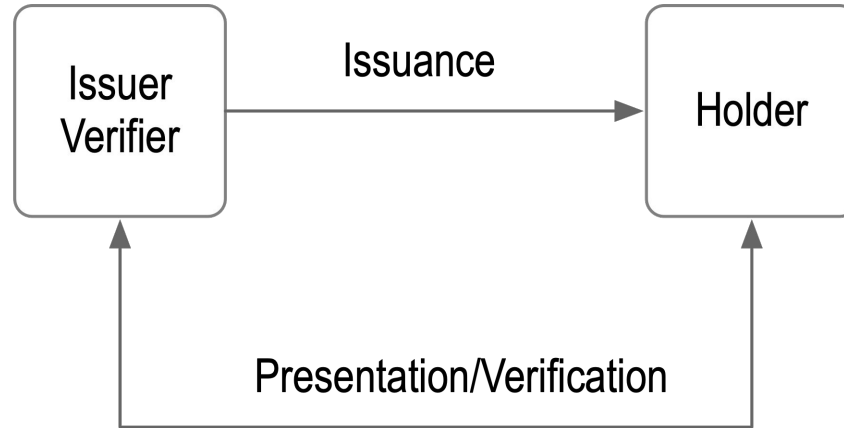
Tripartite without VDR

Issuer-Holder-Verifier Model with Verification at Issuer

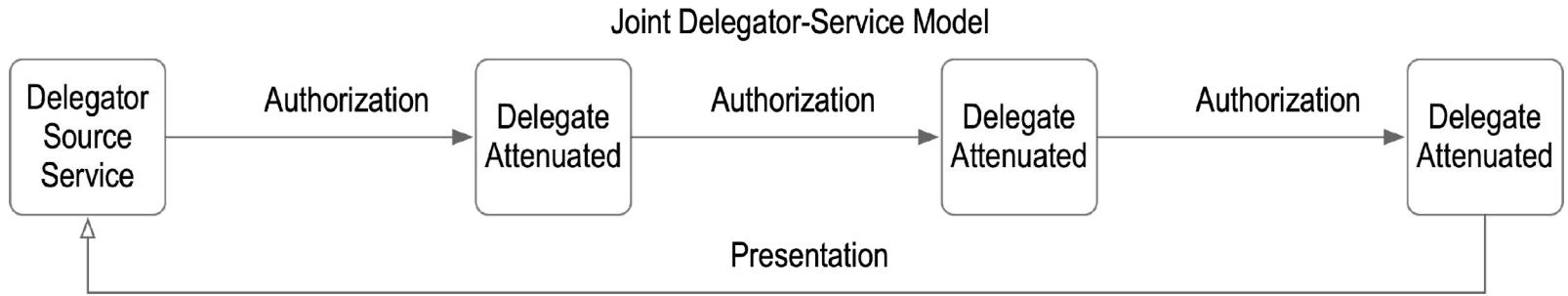


Bipartite Model

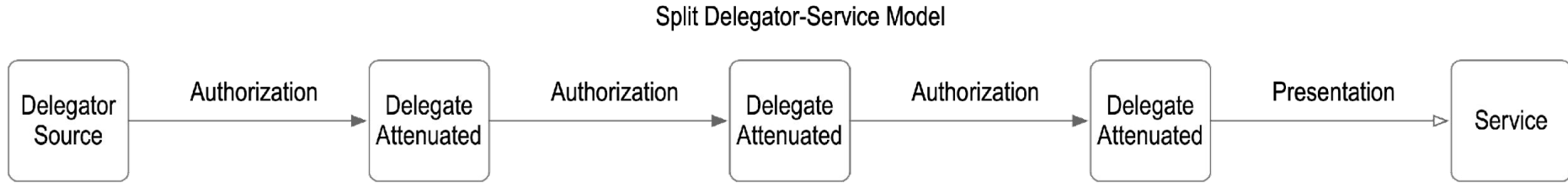
Issuer-Holder Model with Verification at Issuer



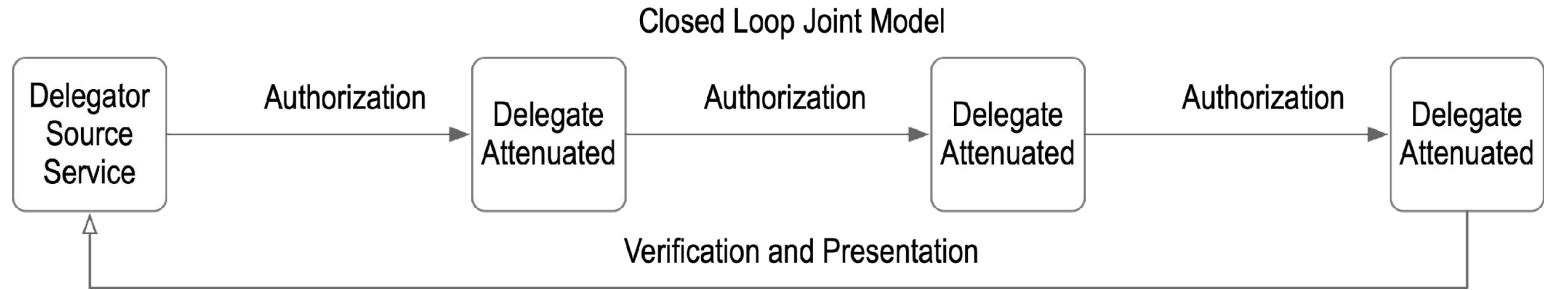
Joint Delegator-Service Model



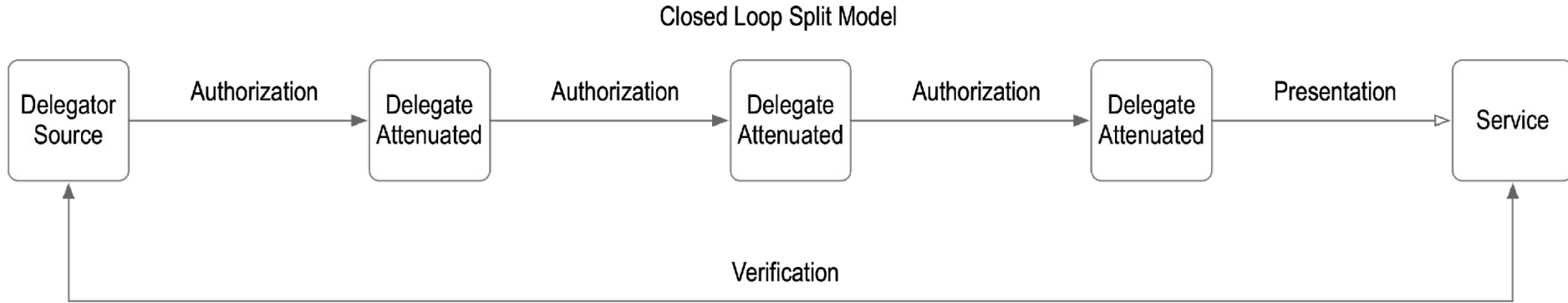
Split Delegator-Service Model



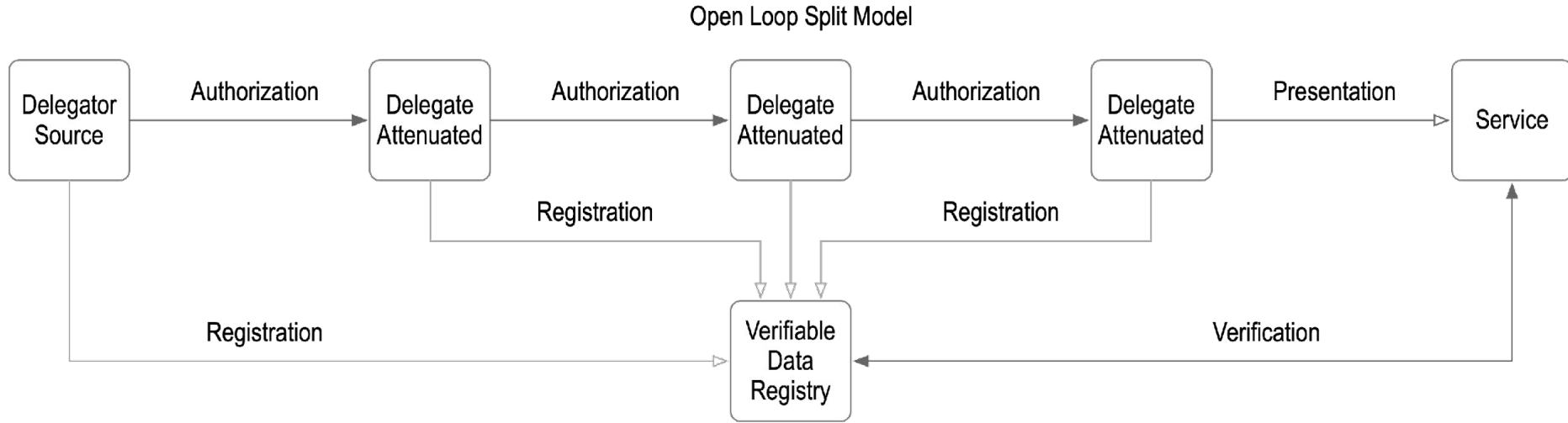
Closed Loop Joint Model



Closed Loop Split Model



Open Loop Split Model

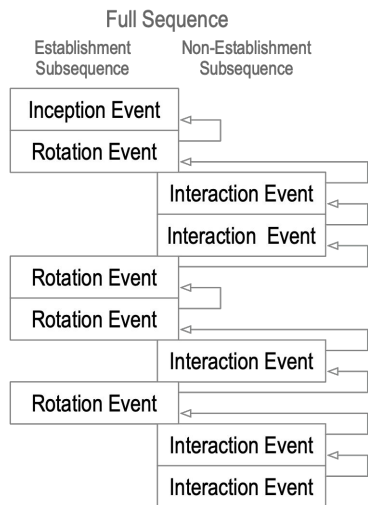


KERI VDRs vs. Shared Ledger VDRs

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-GDvDaCKVn6clOJa7ZXXVXSW  
pRY
```

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.

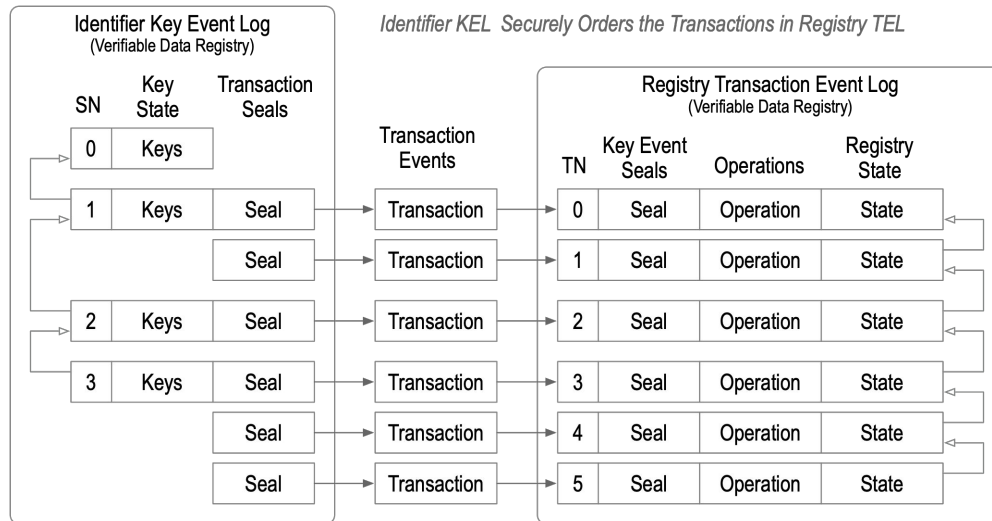
The transactions likewise contain a reference seal back to the key event authorizing the transaction.

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

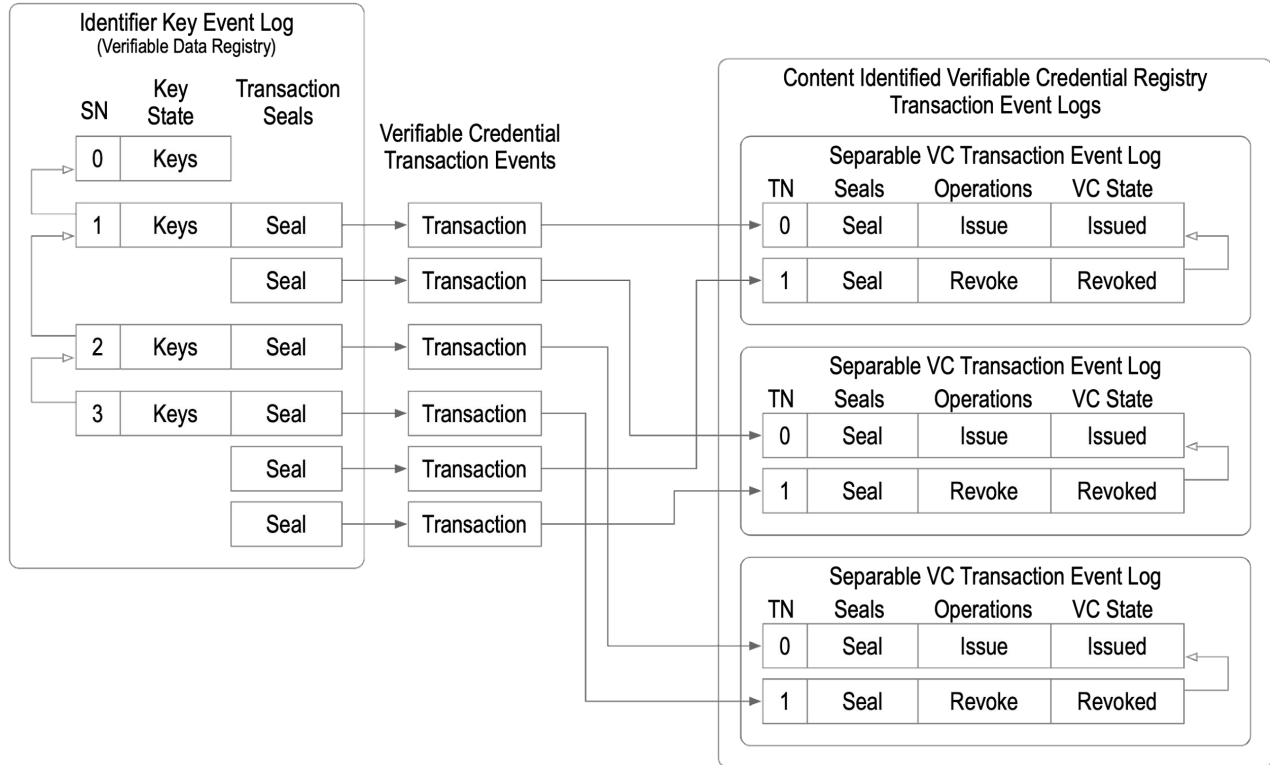
Registry with Separable VC Issuance-Revocation TELs

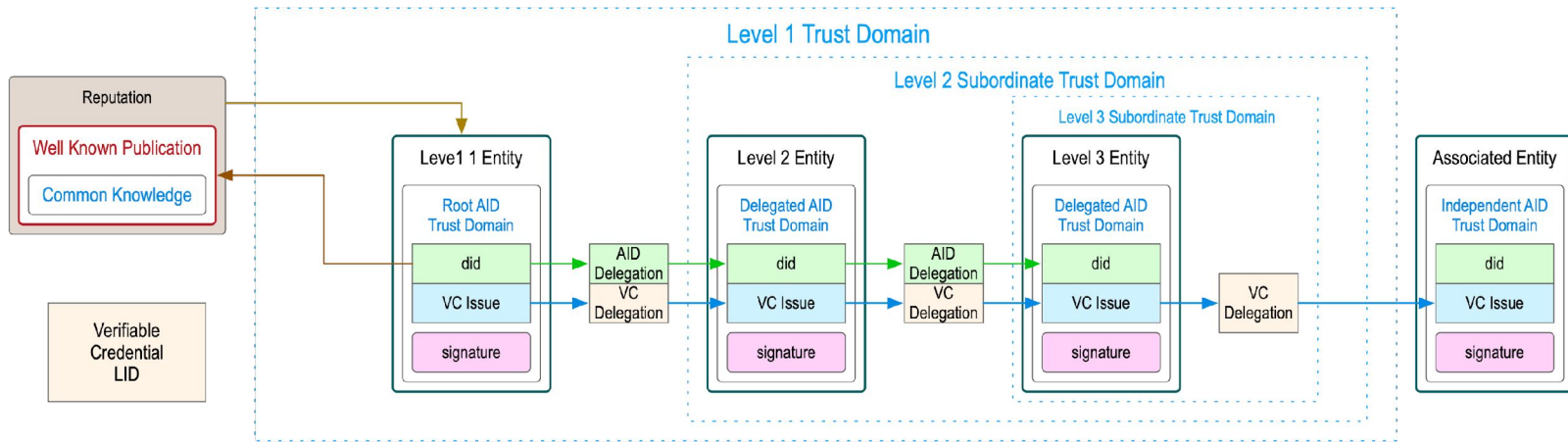
Each VC may be uniquely identified with a SAID.

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

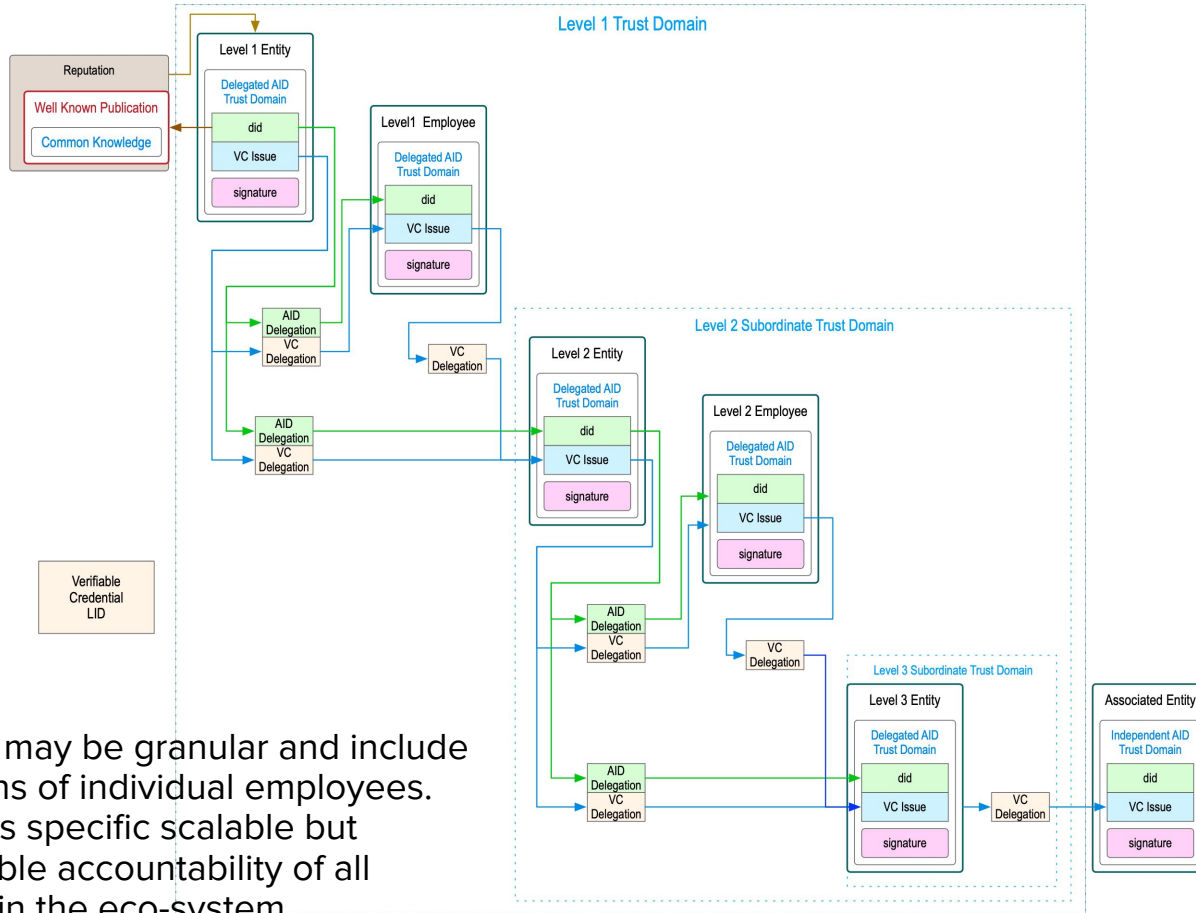




Each level of delegation forms a nested trust domain that is protected by the level above. This increases ultimate security while enabling higher performance event issuance in lower layers.

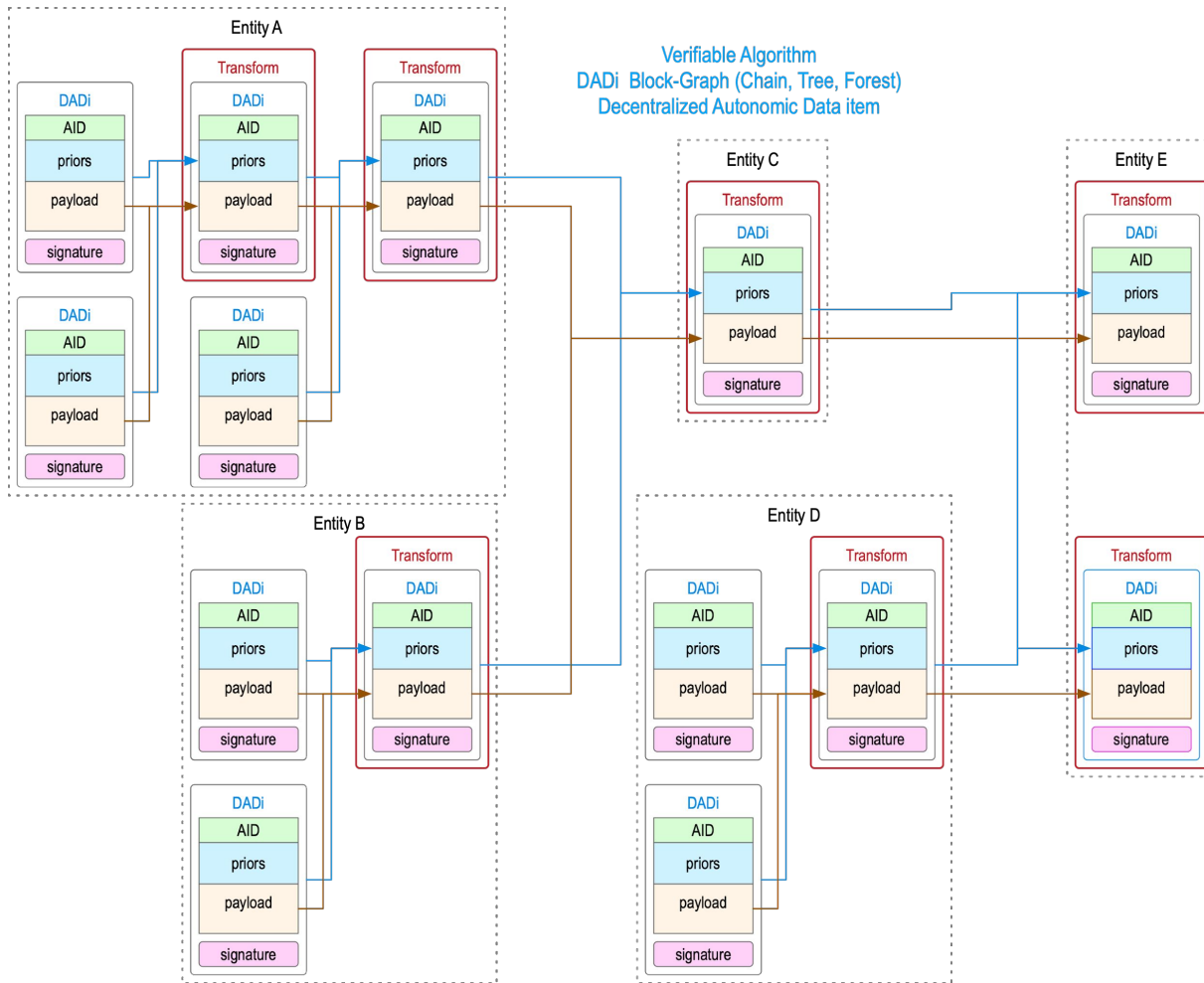
The Level 1 entity AID provides the root-of-trust for the whole ecosystem. This enables secure decentralized interoperability.

Each trust domain may make delegations of both identifiers and verifiable credentials to a subordinate trust domain. These delegations provide revocable authorizations.



Delegations may be granular and include authorizations of individual employees. This provides specific scalable but non-repudiable accountability of all participants in the eco-system.

Verifiable Algorithm
DADi Block-Graph (Chain, Tree, Forest)
Decentralized Autonomic Data item



Identifier System Security

Authentic transmission of data may be verified using an identity system security overlay.

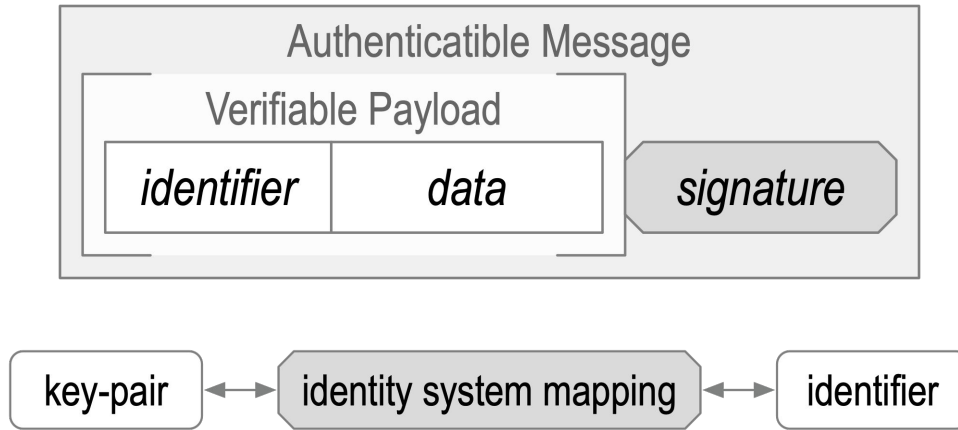
This overlay maps cryptographic key-pairs to identifiers.

When those identifiers are self-certifying they are derived via cryptographic one-way functions from the key pairs.

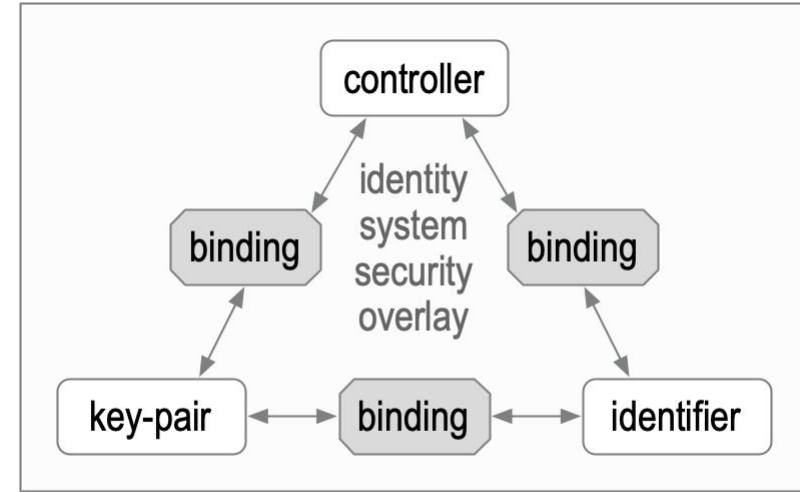
This provides a self-certifying identifier with a cryptographic root-of-trust.

A key event log (KEL) provide support for secure key rotation without changing the identifier.

Message authenticity is provided by verifying signatures to the authoritative keys pairs for the identifier included in the message.



The overlay's security is contingent on the mapping's security.



Identifier Issuance

Background References

self-Certifying Identifiers:

Girault, M., "Self-certified public keys," EUROCRYPT 1991: Advances in Cryptology, pp. 490-497, 1991

https://link.springer.com/content/pdf/10.1007%2F3-540-46416-6_42.pdf

Mazieres, D. and Kaashoek, M. F., "Escaping the Evils of Centralized Control with self-certifying pathnames," MIT Laboratory for Computer Science,

<http://www.sigops.org/ew-history/1998/papers/mazieres.ps>

Kaminsky, M. and Banks, E., "SFS-HTTP: Securing the Web with Self-Certifying URLs," MIT, 1999

<https://pdos.csail.mit.edu/~kaminsky/sfs-http.ps>

Mazieres, D., "Self-certifying File System," MIT Ph.D. Dissertation, 2000/06/01

<https://pdos.csail.mit.edu/~ericp/doc/sfs-thesis.ps>

TCG, "Implicit Identity Based Device Attestation," Trusted Computing Group, vol. Version 1.0, 2018/03/05

<https://trustedcomputinggroup.org/wp-content/uploads/TCG-DICE-Arch-Implicit-Identity-Based-Device-Attestation-v1-rev93.pdf>

Autonomic Identifiers:

Smith, S. M., "Open Reputation Framework," vol. Version 1.2, 2015/05/13

<https://github.com/SmithSamuelM/Papers/blob/master/whitepapers/open-reputation-low-level-whitepaper.pdf>

Smith, S. M. and Khovratovich, D., "Identity System Essentials," 2016/03/29

<https://github.com/SmithSamuelM/Papers/blob/master/whitepapers/Identity-System-Essentials.pdf>

Smith, S. M., "Decentralized Autonomic Data (DAD) and the three R's of Key Management," Rebooting the Web of Trust RWOT 6, Spring 2018

<https://github.com/SmithSamuelM/Papers/blob/master/whitepapers/DecentralizedAutonomicData.pdf>

Smith, S. M., "Key Event Receipt Infrastructure (KERI) Design and Build", arXiv, 2019/07/03 revised 2021

<https://arxiv.org/abs/1907.02143>

Smith, S. M., "Key Event Receipt Infrastructure (KERI) Design", 2020/04/22

https://github.com/SmithSamuelM/Papers/blob/master/whitepapers/KERI_WP_2.x.web.pdf

Stocker, C., Smith, S. and Caballero, J., "Quantum Secure DIDs," RWOT10, 2020/07/09

<https://github.com/WebOfTrustInfo/rwot10-buenosaires/blob/master/final-documents/quantum-secure-dids.pdf>

Smith, S. M., "Universal Identifier Theory", 2020/10/23

https://github.com/SmithSamuelM/Papers/blob/master/whitepapers/IdentifierTheory_web.pdf

Certificate Transparency:

Laurie, B., "Certificate Transparency: Public, verifiable, append-only logs," ACMQueue, vol. Vol 12, Issue 9, 2014/09/08

<https://queue.acm.org/detail.cfm?id=2668154>

Google, "Certificate Transparency,"

<http://www.certificate-transparency.org/home>

Laurie, B. and Kasper, E., "Revocation Transparency,"

<https://www.links.org/files/RevocationTransparency.pdf>

Internet Safety with KERI

Invasion Percolation Discovery OOBIs (Out-Of-Band-Introductions) Spanning Trust Layer

Samuel M. Smith Ph.D.
IIW 20201 B
sam@keri.one
<https://keri.one>



User Permissioned (web-of-trust) Percolated Discovery

Invasion-Percolation Graph Theory for attack resistance

https://en.wikipedia.org/wiki/Percolation_theory

https://en.wikipedia.org/wiki/First_passage_percolation

<http://www.physics.purdue.edu/flow/MMproject/Wilkinson1983.pdf>

<https://journals.aps.org/prl/abstract/10.1103/PhysRevLett.103.018701>

The Square and the Tower: Networks and Power. Niall Ferguson 2018

Percolation Theory uses graph theory to model the rate and extent of information flow by pair-wise or n-wise sharing of information. No global lookup. Weak and Strong Links etc.

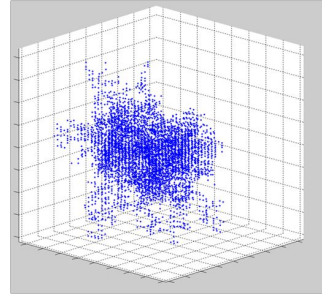
If network enables percolation and is spanned then all information is eventually available everywhere

Primary Result (Invasion-Percolation):

Eventually information fills (invades) all honest nodes in the graph whenever “capillary force“ (authenticity) is greater for good information over bad information.

User permissioning means honest nodes self-isolate dishonest-nodes.

Each honest user forms identity graph of other honest nodes it interacts with that forms web-of-trust anchoring percolation discovery network.



User Permission Percolated Discovery

Insight: Need-to-know just-in-time discovery (NTK-JIT)

Issuer may provide upon demand at issuance all information an Issuee (*Holder*) needs to verify the issuance. Now Holder has discovered by percolation what it *needs-to-know* (NTK) *just-in-time* (JTK) to verify.

Holder now may provide upon demand at presentation all information any verifier needs to to verify the presentation. Now verifier has discovered by percolation what it *needs-to-know* (NTK) *just-in-time* (JTK) to verify. This includes all the percolated discovery from Issuer to Holder.

Likewise the Verifier may imbue on a *NTK-JIT* basis any subsequent use of that information with all the percolated discovery information it already received from the Holder plus any other information the Verifier needs to contribute.

KERI End-Verifiability means zero-trust in the percolation path.

Discovery becomes an availability not a security problem.

User Permissioned Percolated Discovery

SPED (Speedy Percolated Endpoint Discovery)

Privacy preserving or public discovery as needed

User permissioned & totally decentralized

Replaces or Augments User Permissioned DHT

Watcher Network may provide super Nodes for aggregated discovery if desirable

End-to-end verifiability means any discovery source is as good as any other.

End verifiable “truth” is still true from whatever source it may have come.

This enables secure bootstrap of discovery from any source on a NTK JIT basis.

No need for a globally trusted discovery bootstrap resolver

Zero Trust Percolated Discovery

Primary Discovery Data are Endpoints of KERI Components:

Controllers, Agents, Backers (Witness, Registrar), Watchers, Jurors, Judges, Forwarders

Endpoint is URL: IP Scheme, Host, Port, Path etc

Data Model for Securely Managing EndPoint Data

Controller (Principal AID)

Authorizes a Component to act as Player in Role

Player is AID of Component Controller

Role is purpose or function such as Watcher

Zero Trust Data as Authorization in context of KERI KeyState

ACDC Issue Revoke Reissue model

RUN model (Read, Update, Nullify)

Anchored or Signed with replay and deletion attack protection

Safe Internet Use

Minimally Sufficient Means

Leverage existing internet but safely, with end-verifiability

Internet DNS/CA is out-of-band w.r.t. KERI security

Use DSN/CA for out-of-band introductions w.r.t. KERI only, not authentication

Use IP addresses (128.187.16.184) for communication

OOBI (Out-Of-Band-Introduction)

How to use DNS safely! Vacuum discovery of service endpoints.

Basic

```
https://hackmd.io/MxTAIBQTRkWU4-w140tNuA
```

OOBI = Url and AID Simple enough for QR Code

```
http://8.8.5.6:8080/oobi/EaU6JR2nmwyZ-i0d8JZAoTNZH3ULvYAfSVPzhzS6b5CM
```

Variant: Use query string to label endpoint to be discovered.

```
http://8.8.5.6:8080/oobi/EaU6JR2nmwyZ-i0d8JZAoTNZH3ULvYAfSVPzhzS6b5CM?role=watcher&name=eve
```

```
https://example.com/oobi/EaU6JR2nmwyZ-i0d8JZAoTNZH3ULvYAfSVPzhzS6b5CM?role=witness
```

Well-Known Variant:

```
/.well-known/keri/oobi/EaU6JR2nmwyZ-i0d8JZAoTNZH3ULvYAfSVPzhzS6b5CM
```

Result of well-known request is target URL or redirection

```
https://example.com/witness/witmer (redirection)
```

```
http://8.8.5.5:8080/witness/witmer (public IP)
```

```
http://10.0.5.15:8088/witness/witmer (private IP)
```

Any OOBI may forward to another OOBI.

This is safe because the eventual endpoint is end-verifiable (authenticated).

OOBI (Out-Of-Band-Introduction)

Verbose OOBI Multi-OOBI

```
{
  "v" : "KERI10JSON00011c_",
  "t" : "rpy",
  "d" : "EZ-i0d8JZAoTNZH3ULaU6JR2nmwyvYAfSVPzhzS6b5CM",
  "dt" : "2020-08-22T17:50:12.988921+00:00",
  "r" : "/oobi/witness",
  "a" :
  {
    "urls": ["http://example.com/watcher/watson", "http://example.com/witness/wilma"]
    "aid": "EaU6JR2nmwyZ-i0d8JZAoTNZH3ULvYAfSVPzhzS6b5CM"
  }
}
```

Special Route Path

```
{
  "v" : "KERI10JSON00011c_",
  "t" : "rpy",
  "d" : "EZ-i0d8JZAoTNZH3ULaU6JR2nmwyvYAfSVPzhzS6b5CM",
  "dt" : "2020-08-22T17:50:12.988921+00:00",
  "r" : "/oobi/EaU6JR2nmwyZ-i0d8JZAoTNZH3ULvYAfSVPzhzS6b5CM/watcher",
  "a" :
  {
    "eid": "BrHLayDN-mXKv62DAjFLX1_Y5yEUe0vA9YPe_ihiKYHE",
    "scheme": "http",
    "url": "http://example.com/watcher/wilma",
  }
}
```

Bare URL as Self or Blind OOB

A bare URL but no AID may be used as a bare OOB for blind or self introductions.

Querying that bare URL (OOB) may return or result in a default target OOB or default target endpoint reply.

This provides a mechanism for self-introduction, self OOB (SOOB) or blind-introduction, blind OOB (BOOB) .

`http://8.8.5.7:8080/oob`

`http://localhost:8080/oob``

`http://8.8.5.7:8080/oob?role=controller&name=eve`

`http://localhost:8080/oob?role=controller&name=eve`

By default the result of get request to this OOB URL could be another OOB with an AID that is the `self` AID of the node providing the bare OOB endpoint or the actual authenticatable `self` endpoint with its AID or a default set of authenticatable endpoints.

Useful to bootstrap components in an infrastructure where the target URLs do not use a public DNS address but use instead something more secure like an explicit public IP address or a private IP or private DNS address.

A self introduction provides a bootstrap mechanism similar to a hostname configuration file with the exception that in the OOB case the AID is not in the configuration file just the bare OOB URL and the given node queries that bare OOB to get the target endpoint AID. This allows bootstrap using bare IP addresses in systems where the IP infrastructure is more securely managed than public DNS or where some other Out-Of-Band-Authentication (OOBA) mechanism is used in concert.

Blind OOBI

Because the OOBI does not expose an AID, the resultant response when querying the OOBI may depend on other factors such as the source IP of the querier (requester) and/or another out-of-band-authentication (OOBA) mechanism. This supports private bootstrap of infrastructure.

Of course one could argue that this is just kicking the can down the road but IP addresses are correlatable and a blind OOBI can leverage IP infrastructure for discovery when used in combination with some other OOBA mechanism without unnecessary correlation.

Onion Routing with Blind OOBI

did-comm with Blind OOBI

Attack Protection

Replay Attack: Replay *of Authenticated (signed) Data*

TEL (ACDC) VDR Issue Revoke (kel anchored tel events) Heavyweight

Non TEL based: Best Available Data Model (BADA)

KEL anchored ordered data

KeyState-DateTime of signature ordered data.

Deletion Attack

Total erasure a security problem (GDPR flaw)

Once erased any stale authenticated data acting as authorization may be replayed without detection.

Mitigation for Deletion attack are redundant signed copies (eventually consistent DB)

BADA (Best Available Data Acceptance) Policy

Authentic Data:

Two primary attacks:

Replay attack:

Mitigation: Monotonicity

Deletion attack:

Mitigation: Redundancy

Replay Monotonicity:

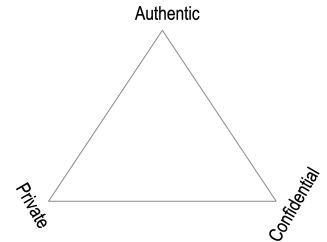
Interactive:

Nonce

Non-interactive:

Memory (sequence number, date-time stamp, nullification)

More scalable



BADA Rules

Update is included in or anchored to AID's key-state in KEL:

Rules for Acceptance of update :

Accept if no prior record.

Accept if update's anchor is later than prior record's anchor.

Update is signed by AID, but the update itself is not included in or anchored to AID's KEL:

1) Ephemeral AID whose key-state is fixed (no KEL needed)

2) Persistent AID whose key-state is provided by KEL

Rules for Acceptance of update :

If no prior record.

Accept if signature verifies against any key-state.

If prior record.

Compare key-state of the update's verified signature against key-state of prior record's verified signature.

Accept If update's key-state is later (in KEL) than prior record's key-state.

Accept if update's and prior record's key-states are the same

& update's date-time is later than prior record's date-time.

RUN off the CRUD

Client-Server API or Peer-to-Peer.

Create, Read, Update, Delete (CRUD)

Read, Update, Nullify (RUN)

Decentralized control means server never creates only client. Client (Peer) updates server (other Peer) always for data sourced by Client (Peer). So no Create.

Non-interactive monotonicity means we can't ever delete.

So no Delete. We must Nullify instead. Nullify is a special type of Update.

Ways to Nullify:

- null value

- flag indicating nullified

EndPoint Disclosure

Datetime stamped BADA authorization by CID of EID in Role (Update)

Datetime stamped BADA deauthorization by CID of EID in Role (Nullify)

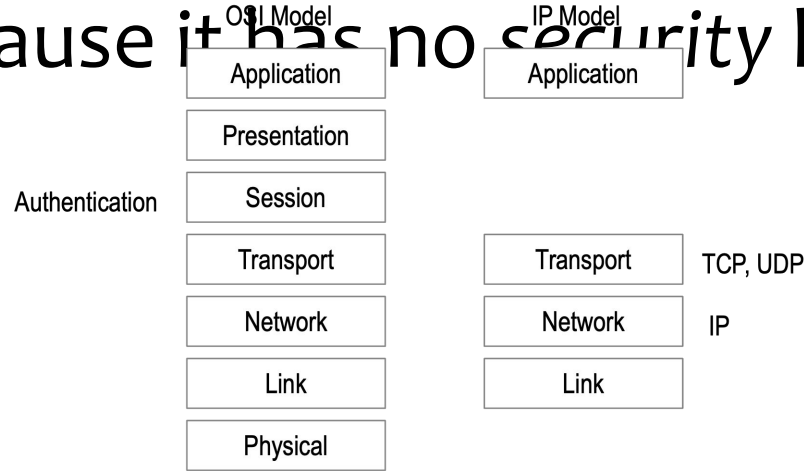
Datetime stamped BADA authorization by EID of URL for scheme (Update).

Datetime stamped BADA deauthorization by EID of URL for scheme (Nullify)

The Internet Protocol (IP) is

bro-ken

because it has no security layer.

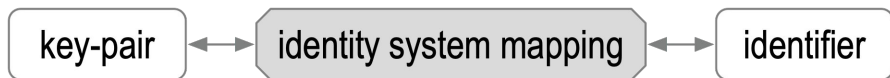


Instead

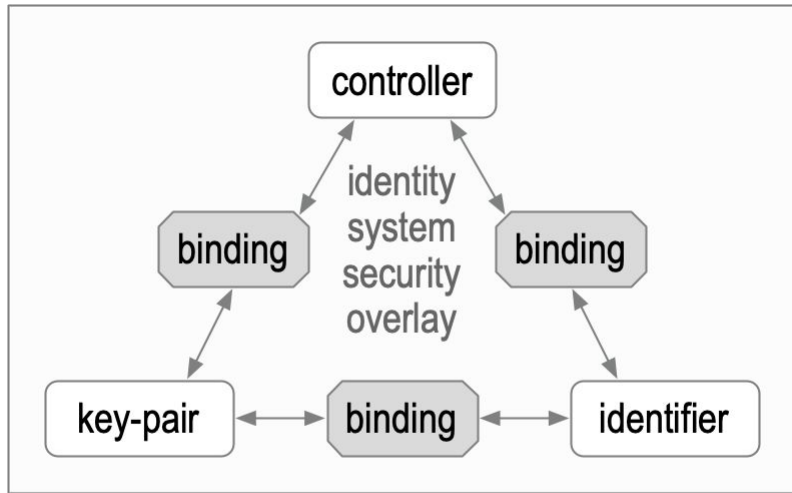
We use *bolt-on* identity-system security overlays.
(DNS-CA ...)

Identity System Security Overlay

Establish authenticity of IP packet's message payload.

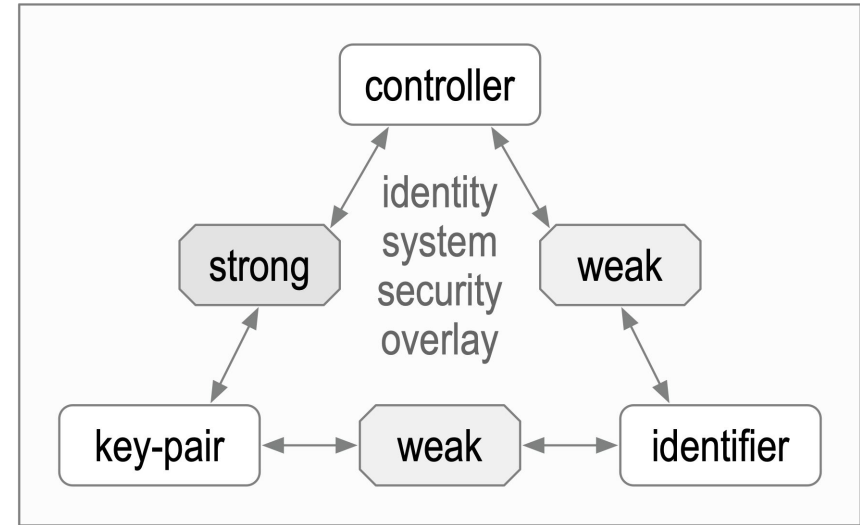
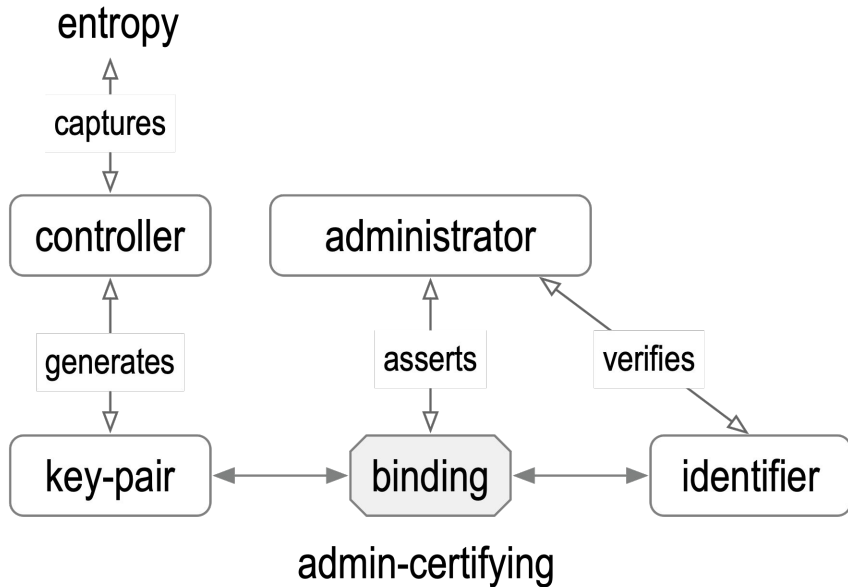


The overlay's security is contingent on the mapping's security.



Identifier Issuance

Administrative Identifier Issuance and Binding

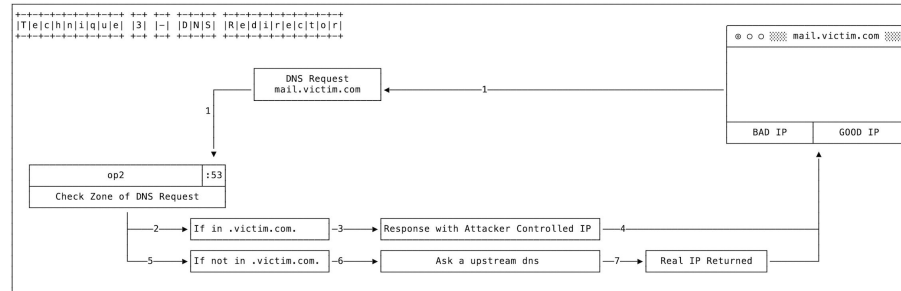
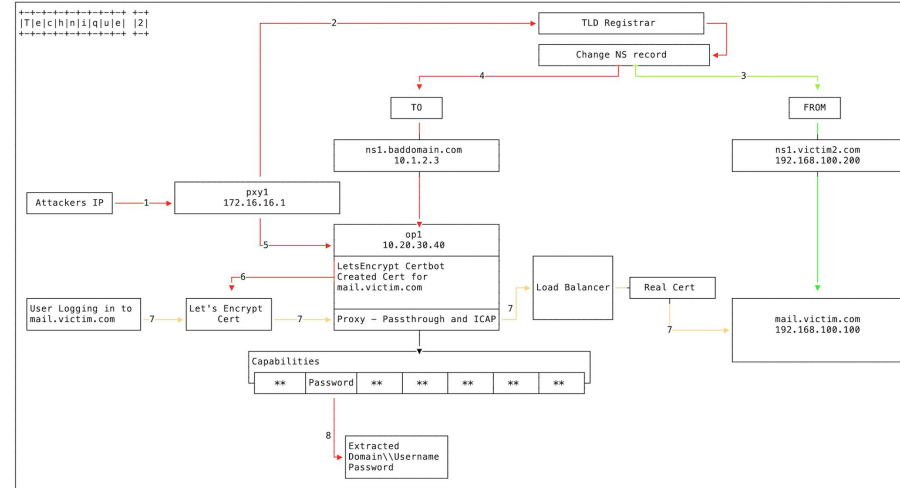
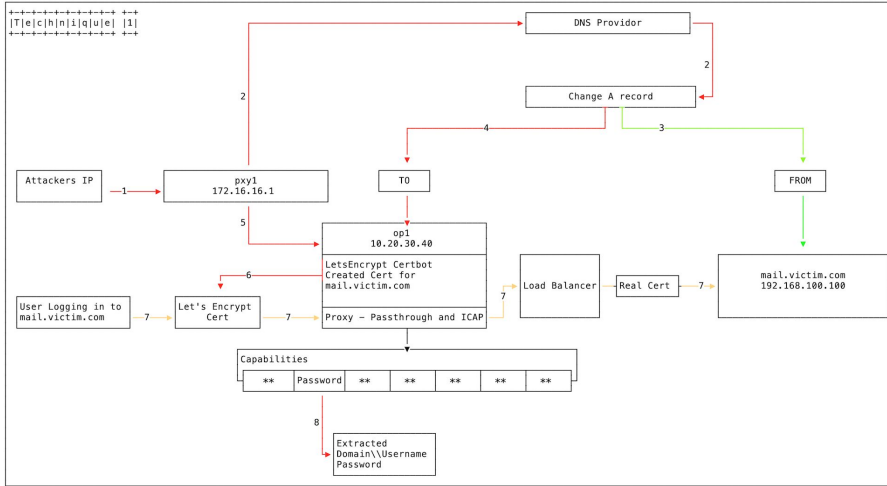


Admin-Certifying Identifier Issuance

DNS Hijacking

A DNS hijacking is occurring at an unprecedented scale. Clever tricks allows attackers to obtain valid TLS certificate for hijacked domains.

<https://arstechnica.com/information-technology/2019/01/a-dns-hijacking-wave-is-targeting-companies-at-an-almost-unprecedented-scale/>



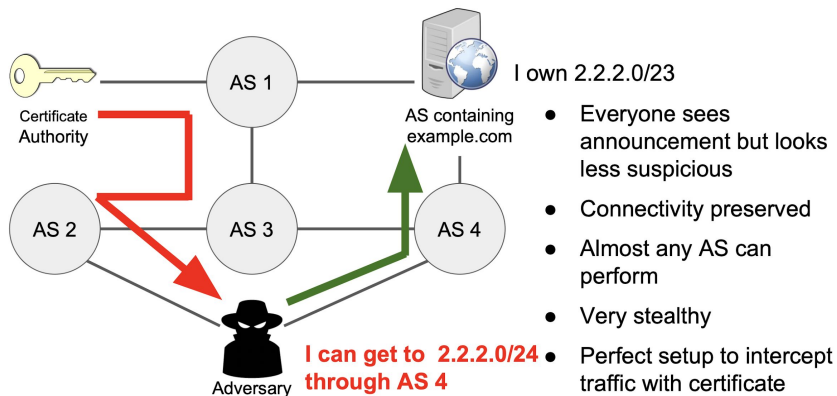
BGP Hijacking: AS Path Poisoning

Spoof domain verification process from CA. Allows attackers to obtain valid TLS certificate for hijacked domains.

Birge-Lee, H., Sun, Y., Edmundson, A., Rexford, J. and Mittal, P., "Bamboozling certificate authorities with {BGP}," vol. 27th {USENIX} Security Symposium, no. {USENIX} Security 18, pp. 833-849, 2018 <https://www.usenix.org/conference/usenixsecurity18/presentation/birge-lee>

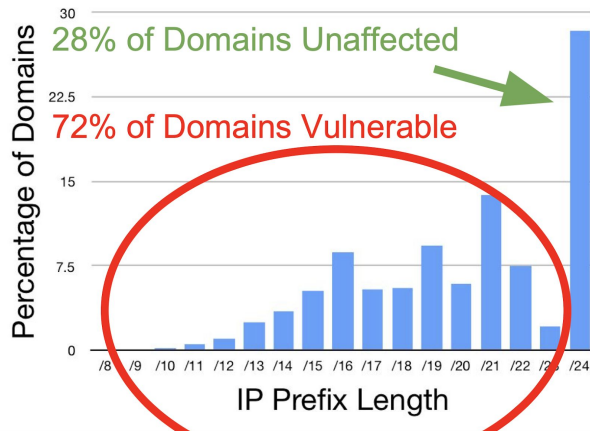
Gavrichenkov, A., "Breaking HTTPS with BGP Hijacking," BlackHat, 2015 <https://www.blackhat.com/docs/us-15/materials/us-15-Gavrichenkov-Breaking-HTTPS-With-BGP-Hijacking-wp.pdf>

AS path poisoning

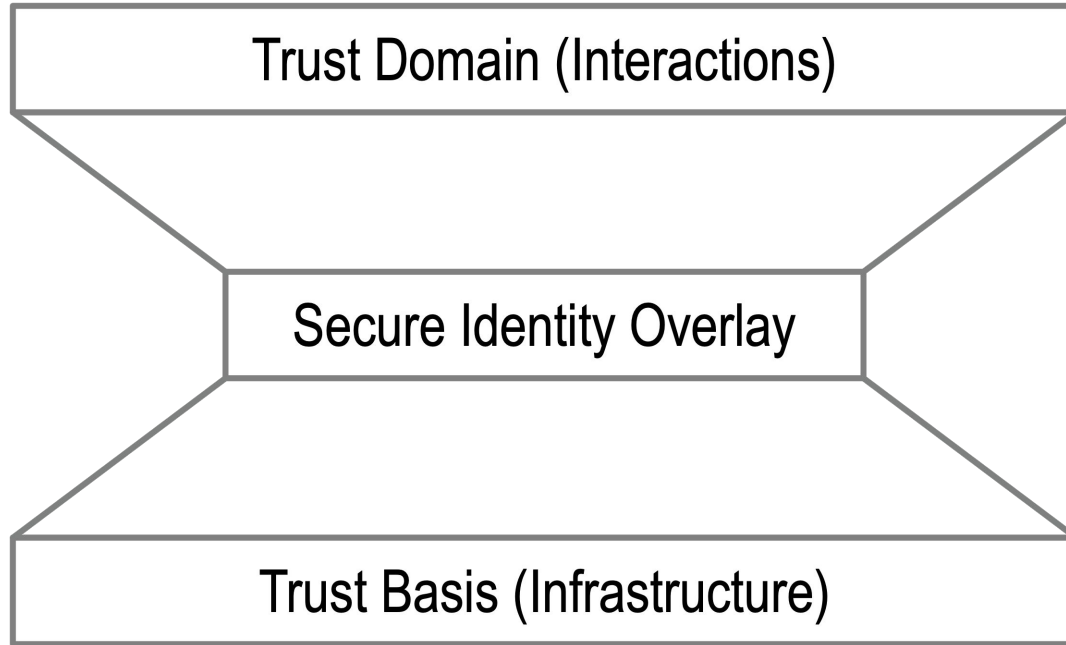


Vulnerability of domains: sub-prefix attacks

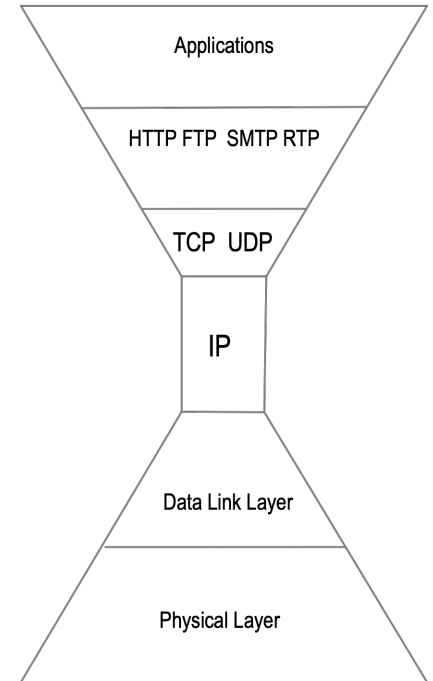
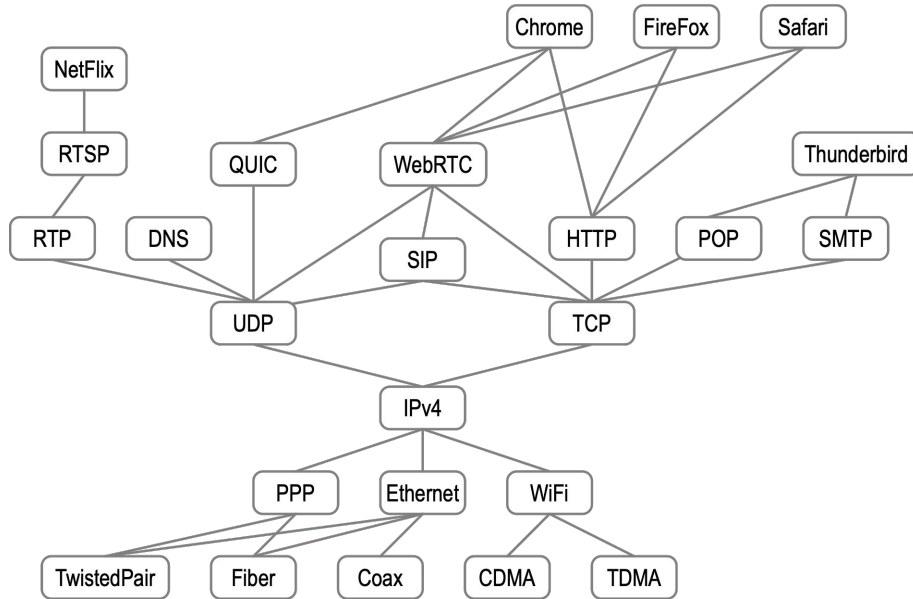
- Any AS can launch
- Only prefix lengths less than /24 vulnerable (filtering)



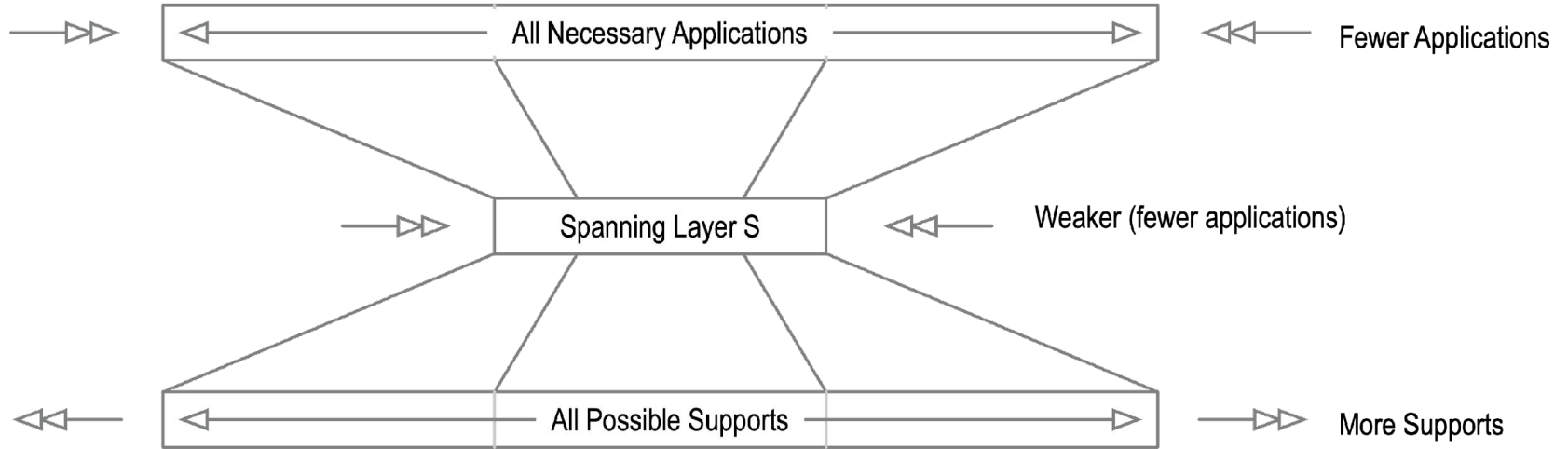
Identity System Security Overlay



Spanning Layer

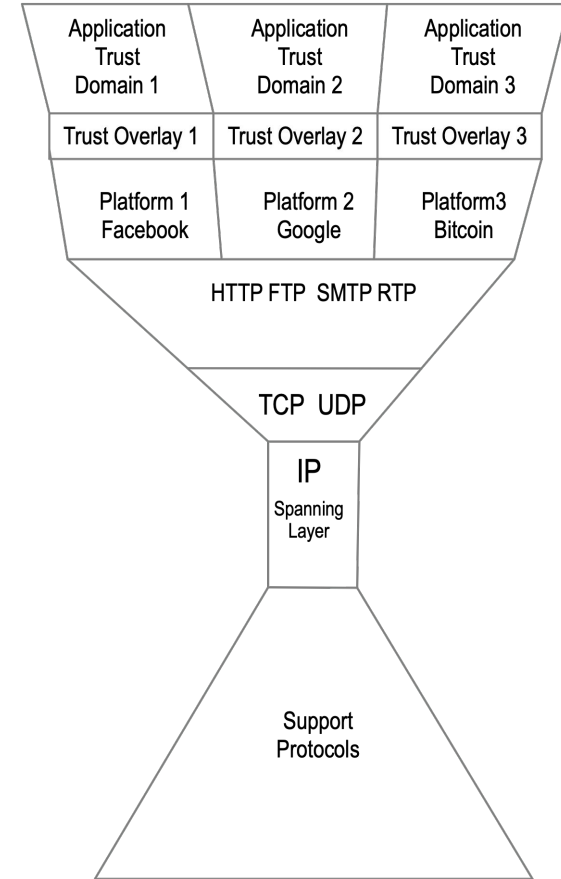
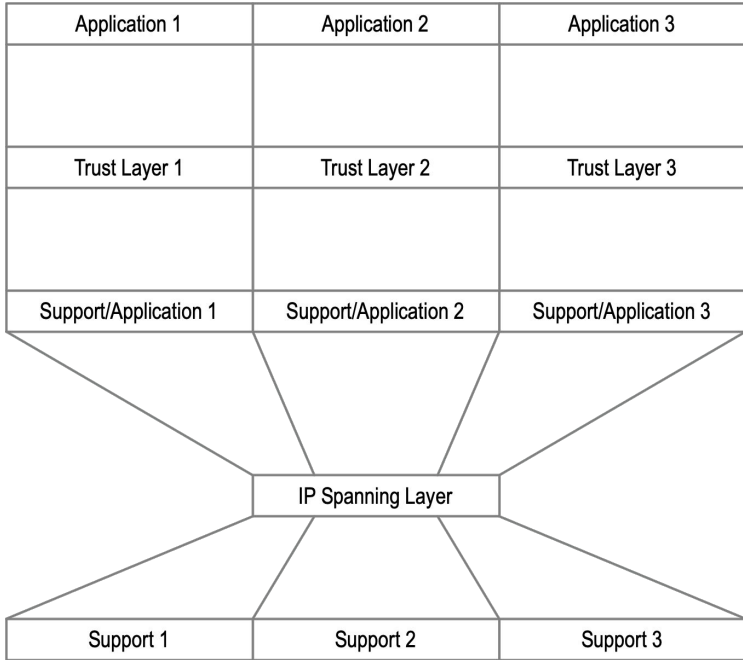


Hourglass



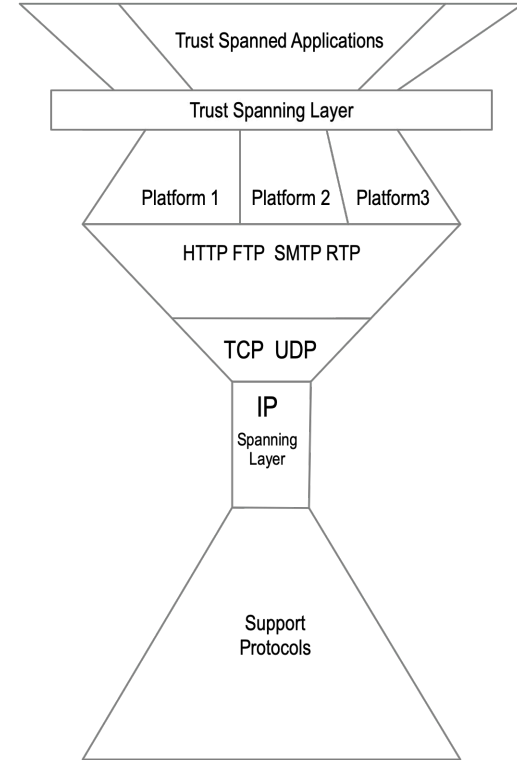
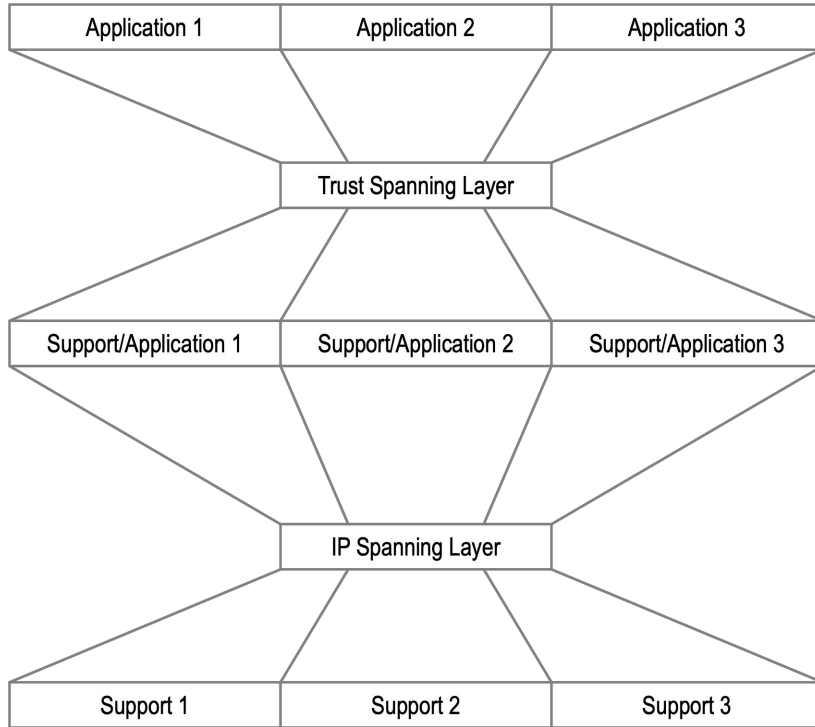
Platform Locked Trust

Trust Domain Based Segmentation



Each trust layer only spans platform specific applications
Bifurcated internet trust map

Solution: Waist and Neck



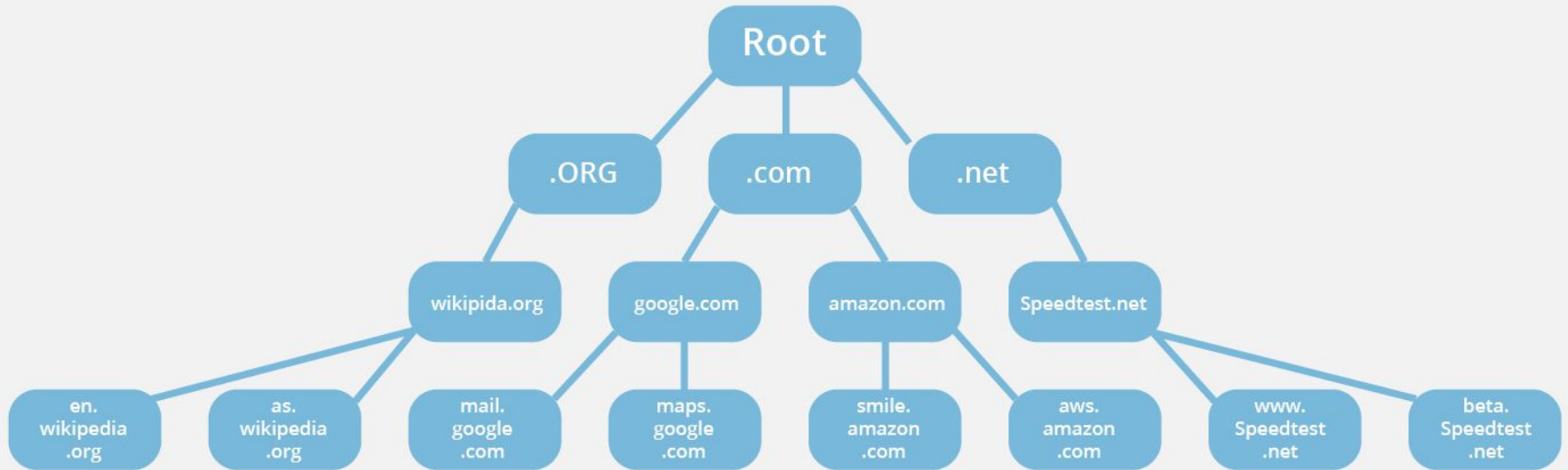
Background

Discovery

Ledger
Based

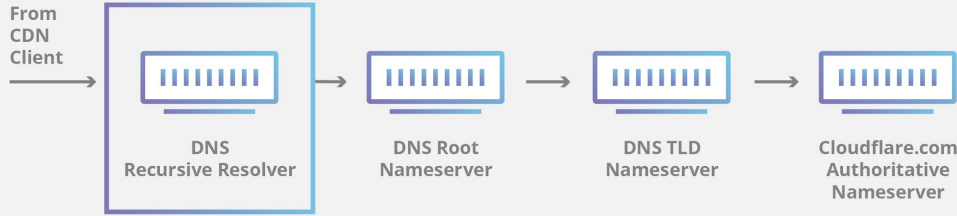
Non-Ledger
Based

DNS “Hierarchical” Discovery

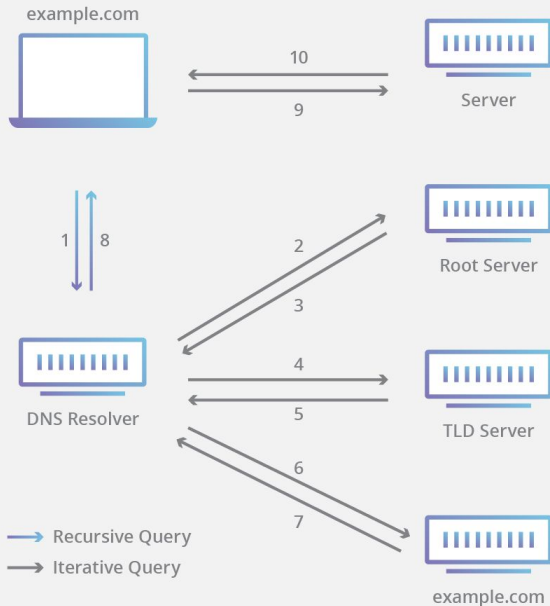


DNS "Hierarchical" Discovery

DNS Record Request Sequence



Complete DNS Lookup and Webpage Query



\$ORIGIN example.com.

ns1.p30.oraclecloud.net. (

responsible party

@ 3600 SOA

zone-admin.dyndns.com. ; address of

2016072701 ; serial number

3600 ; refresh period

600 ; retry period

604800 ; expire time

1800); minimum ttl

86400 NS ns1.p68.dns.oraclecloud.net.

86400 NS ns2.p68.dns.oraclecloud.net.

86400 NS ns3.p68.dns.oraclecloud.net.

86400 NS ns4.p68.dns.oraclecloud.net.

3600 MX 10 mail.example.com.

3600 MX 20 vpn.example.com.

3600 MX 30 mail.example.com.

60 A 204.13.248.106

3600 TXT "v=spf1 includespf.oraclecloud.net

~all"

mail 14400 A 204.13.248.106

vpn 60 A 216.146.45.240

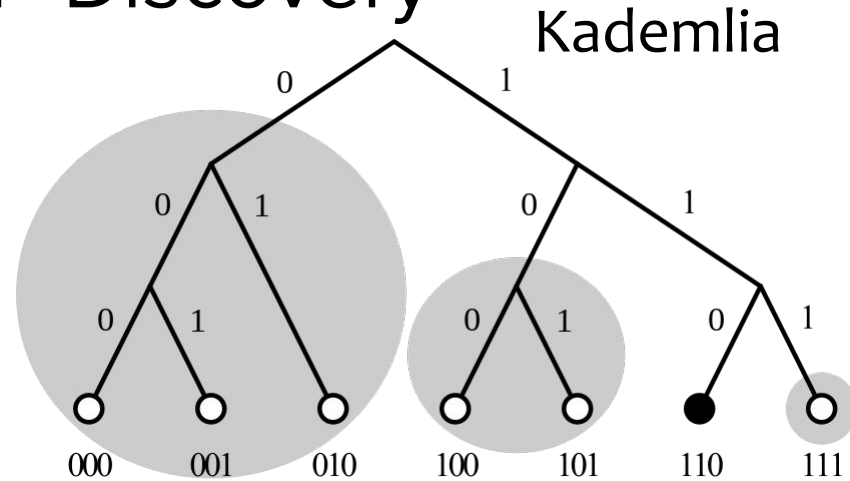
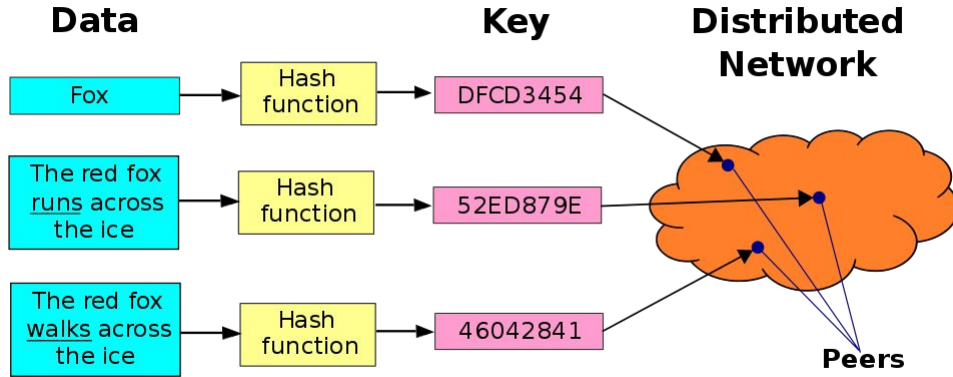
webapp 60 A 216.146.46.10

webapp 60 A 216.146.46.11

www

43200 CNAME example.com.

DHT “Distributed” Discovery



Certificate Transparency Problem

“The solution the computer world has relied on for many years is to introduce into the system trusted third parties (CAs) that vouch for the binding between the domain name and the private key. The problem is that we've managed to bless several hundred of these supposedly trusted parties, any of which can vouch for any domain name. Every now and then, one of them gets it wrong, sometimes spectacularly.”

Pinning inadequate

Notaries inadequate

DNSSEC inadequate

All require trust in 3rd party compute infrastructure that is inherently vulnerable

Certificate Transparency: (related EFF SSL Observatory)

Public end-verifiable append-only event log with consistency and inclusion proofs

End-verifiable duplicity detection = Ambient verifiability of duplicity

Event log is third party infrastructure but zero trust because it is verifiable.

Sparse Merkle Trees for revocation of certificates

Certificate Transparency Solution

Public end-verifiable append-only event log with consistency and inclusion proofs
End-verifiable duplicity detection = ambient verifiability of duplicity
Event log is third party infrastructure but it is not trusted because logs are verifiable.

Sparse Merkle trees for revocation of certificates
(related EFF SSL Observatory)

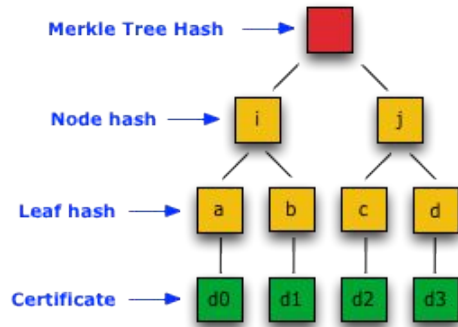


Figure 1

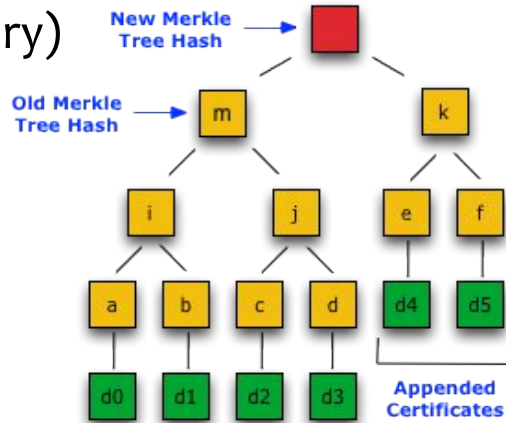


Figure 2

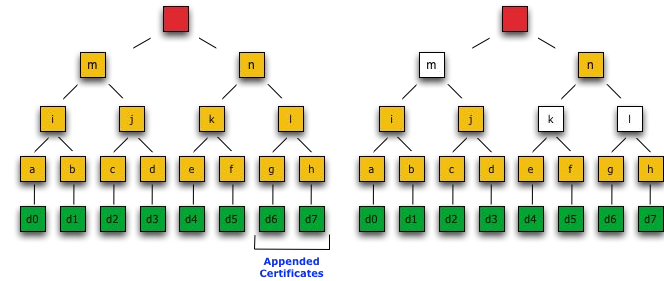


Figure 3

Figure 4

End of Day 1

Verifiable Credentials WG TPAC 2022

Day 2: September 16, 2022

Chairs: Kristina Yasuda, Brent Zundel

Location: Vancouver (and the World Wide Web)

Today's agenda

9:00	Setting expectations for day 2		15
9:15	Holder Binding	Oliver Terbu (remote)	45
10:00			15
10:15	RCH WG time	Markus and Phil	105
12:00			60
13:00	Delegated & Multi-party Credentials	Gabe Cohen, Ori Steele	40
13:40	SD-JWT	Kristina Yasuda	40
14:20	Test Suites	Manu Sporny	40
15:00			15
15:15	Internationalization/Multilingual (Shigeya)	Shigeya Suzuki	30
15:45	JSON Schemas in the VC Data Model (Gabe Cohen)	Gabe Cohen	30
16:15	Any Other Business	Chairs	45

IRC and Scribes

- Meeting discussions will be documented

- Text Chat:

<http://irc.w3.org/?channels=vcwg>

- IRC://[irc.w3.org:6665/#vcwg](irc://irc.w3.org:6665/#vcwg)

- Telecon info

- <https://www.w3.org/events/meetings/a91c3c5b-298a-46a8-8575-61cf15926a97>

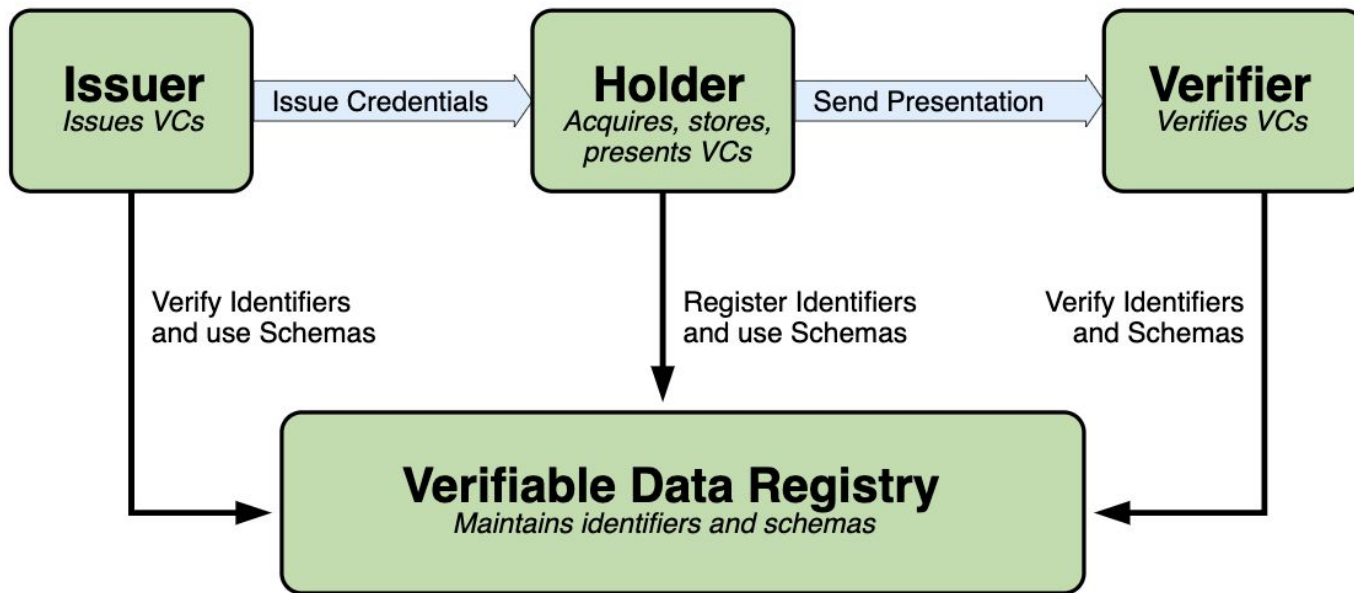
	Morning 1	M2	Afternoon 1	A2
Thurs	Manu	Mkhraisha	Phila	CEL
Fri	Joe	Kevin/Phila	Dave Longley	Orie

<JoeAndrieu> q+ to comment on biometrics
<brent> ack JoeAndrieu
<Zakim> JoeAndrieu, you wanted to comment on biometrics

Holder Binding

(Oliver Terbu, 45 mins)

Recap: Ecosystem Roles



<https://www.w3.org/TR/vc-data-model/#ecosystem-overview>

Recap: Ecosystem Roles

holder

“A role an entity might perform by possessing one or more verifiable credentials and generating verifiable presentations from them. Example holders include students, employees, and customers.” - <https://www.w3.org/TR/vc-data-model/#dfn-holders>

subject

“An entity about which claims are made. Example subjects include human beings, animals, and things. In many cases the holder of a verifiable credential is the subject, but in certain cases it is not. For example, a parent (the holder) might hold the verifiable credentials of a child (the subject), or a pet owner (the holder) might hold the verifiable credentials of their pet (the subject). For more information about these special cases, see Appendix C. Subject-Holder Relationships.” - <https://www.w3.org/TR/vc-data-model/#dfn-subjects>

Recap: Credential Subject

A [verifiable credential](#) contains [claims](#) about one or more [subjects](#). This specification defines a [credentialSubject](#) [property](#) for the expression of [claims](#) about one or more [subjects](#).

A [verifiable credential](#) *MUST* have a [credentialSubject](#) [property](#).

credentialSubject

The value of the [credentialSubject](#) [property](#) is defined as a set of objects that contain one or more properties that are each related to a [subject](#) of the [verifiable credential](#). Each object *MAY* contain an [id](#), as described in Section 4.2 [Identifiers](#).

<https://www.w3.org/TR/vc-data-model/#credential-subject>

Recap: Verifiable Presentation

Presentations *MAY* be used to combine and present credentials. They can be packaged in such a way that the authorship of the data is verifiable. The data in a presentation is often all about the same subject, but there is no limit to the number of subjects or issuers in the data. The aggregation of information from multiple verifiable credentials is a typical use of verifiable presentations.

A verifiable presentation is typically composed of the following properties:

id

The id property is optional and *MAY* be used to provide a unique identifier for the presentation. For details related to the use of this property, see Section [4.2 Identifiers](#).

type

The type property is required and expresses the type of presentation, such as `VerifiablePresentation`. For details related to the use of this property, see Section [4.3 Types](#).

verifiableCredential

If present, the value of the verifiableCredential property *MUST* be constructed from one or more verifiable credentials, or of data derived from verifiable credentials in a cryptographically verifiable format.

holder

If present, the value of the holder property is expected to be a URI for the entity that is generating the presentation.

proof

If present, the value of the proof property ensures that the presentation is verifiable. For details related to the use of this property, see Section [4.7 Proofs \(Signatures\)](#).

<https://www.w3.org/TR/vc-data-model/#presentations-0>

Holder Binding

- A VP can be created by anyone which can be different from the Subject of the VCs in the presentation.
- VC Data Model defines a proof in the VP but it does not define further semantics other than the proof of the VP can be used to verify the VP was not tampered with and to verify the authorship.
- Authorship means that the VP was generated by the Holder of the VP. It does not ensure that the Holder is the intended Holder of the presented VCs.
- Verifier would typically need to perform extra steps to ensure that the Holder is the intended Holder of the presented VCs.
 - Trivial if VC is bound to a Subject and the Holder of the VP is the same as the Subject, and Subject and Holder are identified by DIDs.

Holder Binding: what a lot of people do today ...

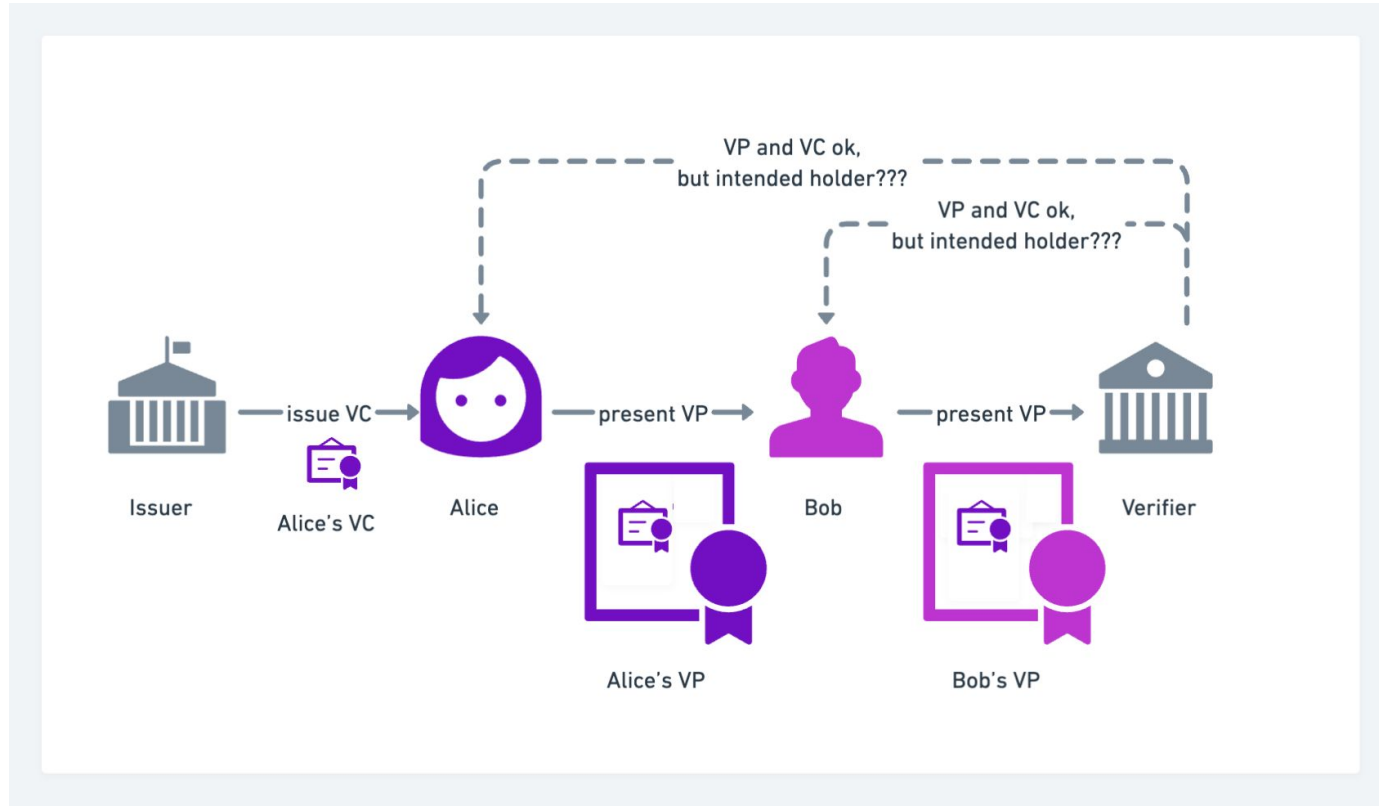
```
{
  "@context": [ ... ],
  "type": [
    "VerifiablePresentation",
    "CredentialManagerPresentation"
  ],
  "holder": "did:example:subject",
  "verifiableCredential": [ {
    "@context": [ ... ],
    "type": [ "VerifiableCredential",
      "UniversityDegreeCredential" ],
    "issuer": "https://example.edu/issuers/565049",
    "issuanceDate": "2010-01-01T00:00:00Z",
    "credentialSubject": {
      "id": "did:example:subject",
      "degree": {
        "type": "BachelorDegree",
        "name": "Bachelor of Science and Arts"
      }
    }
  } ],
  "proof": { ... }
},
"proof": {
  "type": "JsonWebSignature2020",
  "created": "2019-12-11T03:50:55Z",
  "jws": "eyJ...",
  "proofPurpose": "authentication",
  "verificationMethod": "did:example:subject#key-1"
}
}
```

“holder” is non-normative and optional, unclear who is “holder” when omitted

“credentialSubject.id” is optional

IF (holder.id == credentialSubject.id AND hasAuthnMethod(resolve(holder.id), vp.proof.verificationMethod) AND isValid(vp.proof)) THEN Print “Holder Binding validated”

Holder Binding: holder + VP don't solve the issue ...



Holder Binding

- Currently, there is no normative definition of Holder Binding and the `holder` property.
- Related Issues
 - Explicit reference should be added about binding the VC to the holder ([#789](#))
 - "bound"/"binding" terminology is a significantly stronger relationship than is actually present between VCs and their Subjects *or* Holders ([#821](#))
 - Add extension mechanism to allow different methods for VP-to-VC / holder binding to next version of standard to allow verification of rightfulness of presentation ([#882](#))
 - Making explicit the binding of the holder to a VC ([#923](#))

→ **What the community seems to want is to enable the Verifier to validate that the VP was presented by the intended Holder.**

Holder Binding

- More complicated if a different binding method was used
 - VC-based Holder Binding where the VP contains relationship-VCs to bind the Subject identifiers to the Holder.
 - DID-based Holder Binding which is similar to VC-based Holder Binding but the relationship established through DID Document, e.g., alsoKnownAs.
 - Delegation-based Holder Binding where the Subject of the VC delegates the capability to present the VCs to another Holder.
 - Signature IDs (or linked secret)-based Holder Binding for ZKP-based VCs.
 - BBS+, AnonCreds etc.
 - Evidence-based Holder Binding where the Holder Binding is established through an out-of-band agreement.
- **Are there any other binding methods not listed above?**

Holder Binding

- **Let's try to define Holder Binding?**

A method to validate that the intended Holder presented a set of VCs wrapped in a VP.

It binds the following together:

- Subject of the VC (even if vc.credentialSubject.id is undefined)
- Claims made about the Subject by the Issuer
- Holder of the VC (even if vp.holder.id is undefined)
- Proof in the VP
- Proof in the VC

- **Is Holder Binding the right term? Other proposals?**

Holder Binding: multiple proposals

- New VC-level property
`credentialHolder` property
 - Limited to VC only but binding might happen at presentment time
- Reuse NIST 800-63-3
`assurance level` to solve issue
 - Does not provide an answer to the intended Holder question
 - Evidence, VC proof → issuance assurance level
 - VP proof → authentication assurance level (does not solve intended holder)
- Reuse `termsOfUse` property
 - Might be too generic
 - Semantic might be a bit misleading
- Reuse `evidence` property
 - Not available in VPs
 - Relates only to issuance assurance level (identification)
 - Does not relate to authentication context
- New VC/VP-level
`holderBinding` property with registry
 - Acknowledges diversity of holder binding approaches
 - Not a new proof type, just defines how to bind components to validate intended Holder.
 - Allows max. flexibility

Holder Binding

- Requirements
 - Allow the Verifier to validate that the VP was created by intended Holder.
 - Allow the Issuer to attenuate how the holder binding can be proven by the Holder.
 - Allow the Holder to choose how to prove the Holder binding.
 - Holder identifiers should be not required.
 - Credential subjects might have no identifier.
 - Crypto or binding agility.
 - No additional new mandatory properties for VCs and VPs.
 - Holder binding is an optional feature.
 - Support for multiple VC Subjects.

- **Other requirements we should consider?**

Holder Binding (PROPOSAL)

Holder Binding

A method to validate that the intended Holder presented the VCs.

The `holderBinding` MAY be included in VCs and/or in VPs.

holderBinding

If present in the VP or VC, the value of the `holderBinding` property MUST include the following:

- `type` property, which expresses the Holder Binding method type. It is expected that the value will provide enough information to determine the Holder Binding method between the VP and included VCs.

The precise contents of the Holder Binding information is determined by the specific `holderBinding` type definition, and varies depending on the Holder Binding method. The Holder Binding information MAY also include information about for which VC in the VP the Holder Binding applies. For example, this can be done by including a reference of the VC such as the `id` of the VC.

Each Holder Binding method MUST define how Holder Binding for an input VP and one or more input VCs contained in the VP can be deterministically validated. For example, a simple Holder Binding method might define that for a given input VP Holder Binding could be verified based on checking that the `holder` property matches the `credentialSubject.id` property in every `verifiableCredential` object in the VP.

Holder Binding (PROPOSAL)

```
{
  "holder": "did:key:1234:...",
  "holderBinding": [
    {
      "type": "HolderSubjectMatchingBinding2022",
      "someOtherHolderBindingProperties": "..."
    },
    {
      "type": "DelegationHolderBinding2022",
      "delegation": "https://my.holder-binding.abc/12345"
    },
    {
      "type": "SomeAgreementBasedHolderBinding2022",
      "moreHolderBindingProperties": "..."
    },
    {
      "type": "SomeIdCardBasedHolderBinding2022",
      "evenMoreHolderBindingProperties": "..."
    }
  ],
  ...
  "proof": {... }
}
```


Break
(15 mins)

Joint Session - RCH WG

(RCH WG, 105 mins)

Agenda

Introductions from the chairs (Markus & Phil)

Relationship with VCWG (Markus)

Comparative review of the input docs (Phil).

Live presentations from each of

[Dave Longley](#)

[Aidan Hogan](#)

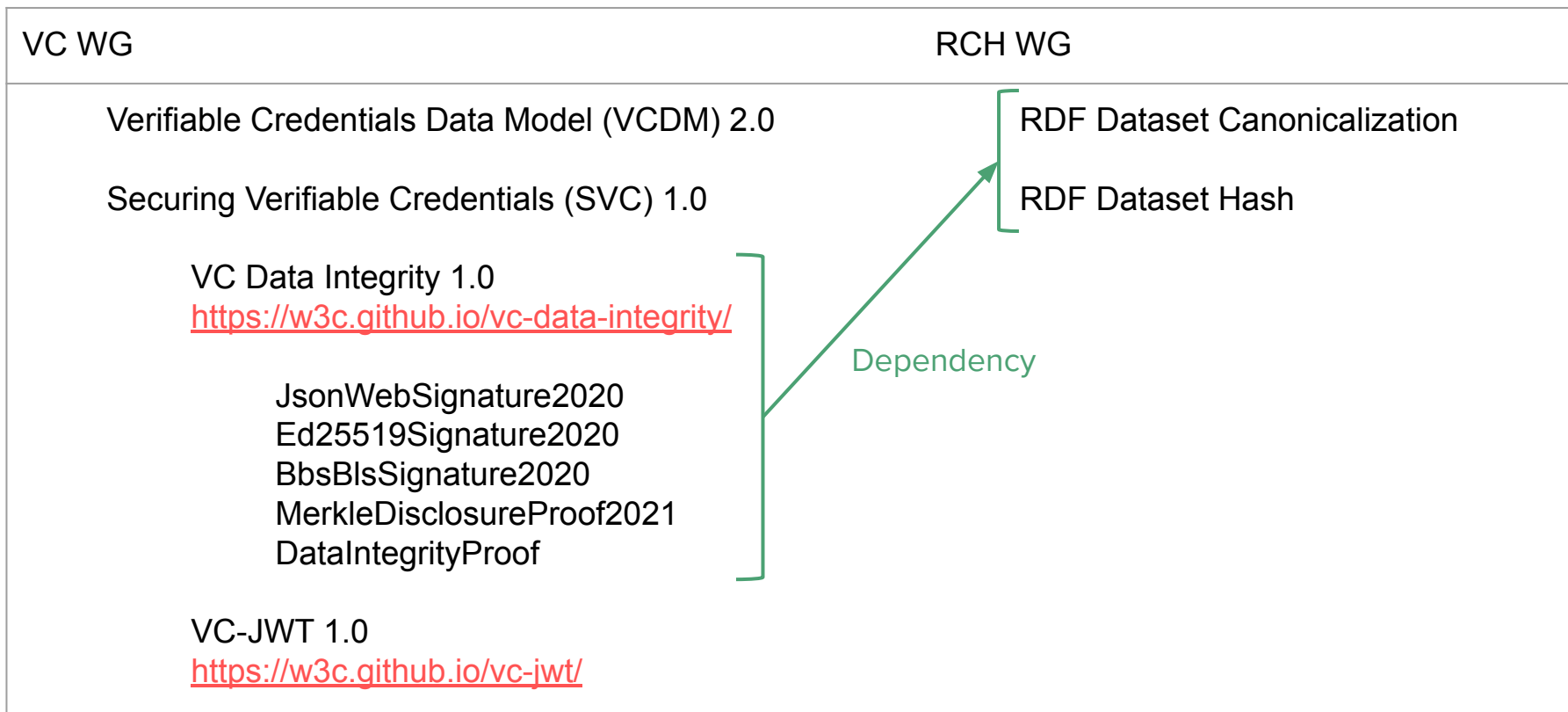
A simple JSON-LD VC example

```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://www.w3.org/2018/credentials/examples/v1"
  ],
  "id": "http://example.gov/credentials/3732",
  "type": ["VerifiableCredential", "UniversityDegreeCredential"],
  "issuer": "https://example.edu",
  "issuanceDate": "2010-01-01T19:23:24Z",
  "credentialSubject": {
    "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
    "degree": {
      "type": "BachelorDegree",
      "name": "Bachelor of Science and Arts"
    }
  },
  "proof": {
    "type": "Ed25519Signature2020",
    "created": "2021-11-13T18:19:39Z",
    "verificationMethod": "https://example.edu/issuers/14#key-1",
    "proofPurpose": "assertionMethod",
    "proofValue": "z58DAdFfa9SkqZMVPxAQpic7ndSayn1PzZs6ZjWp1CktyGesjuTSwrdoWhAfGFCF5bppETSTojQCrfFPP2oumHKtz"
  }
}
```

Something gets canonicalized,
hashed, and signed here!



VCWG and RCHWG deliverables



RCHWG deliverables

RDF Dataset Hash

RDF Dataset Canonicalization



```
<http://www.w3.org/People/EM/contact#me>  
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>  
<http://www.w3.org/2000/10/swap/pim/contact#Person> .  
<http://www.w3.org/People/EM/contact#me>  
<http://www.w3.org/2000/10/swap/pim/contact#fullName> "Eric Miller" .  
<http://www.w3.org/People/EM/contact#me>  
<http://www.w3.org/2000/10/swap/pim/contact#mailbox> <mailto:em@w3.org> .  
<http://www.w3.org/People/EM/contact#me>  
<http://www.w3.org/2000/10/swap/pim/contact#personalTitle> "Dr." .
```

```
z58DAdFfa9SskqZMVPxAQpic7ndSayn1Pz  
Zs6ZjWp1CktyGesjuTSwRdoWhAfgFCF5b  
ppETStojQCrfFPP2oumHKtz
```

Let's try to fix this slide together!

Verifiable Credentials Data Model (VCDM) 2.0

- How to construct most of the VC JSON-LD document, and how to add the "proof" property to a VC/VP.

VC Data Integrity 1.0, and Cryptosuites

- What's inside the "proof" property.
- What exactly needs to get canonicalized and hashed (RDF dataset, options, ...).

RDF Dataset Hash

- How to apply a hash function to an RDF dataset (or individual RDF statements?)

RDF Dataset Canonicalization

- Given an RDF dataset (and other inputs?), create a canonicalized string (or list of strings?)

```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://www.w3.org/2018/credentials/examples/v1"
  ],
  "id": "http://example.gov/credentials/3732",
  "type": ["VerifiableCredential", "UniversityDegreeCredential"],
  "issuer": "https://example.edu",
  "issuanceDate": "2010-01-01T19:23:24Z",
  "credentialSubject": {
    "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
    "degree": {
      "type": "BachelorDegree",
      "name": "Bachelor of Science and Arts"
    }
  },
  "proof": {
    "type": "Ed25519Signature2020",
    "created": "2021-11-13T18:19:39Z",
    "verificationMethod": "https://example.edu/issuers/14#key-1",
    "proofPurpose": "assertionMethod",
    "proofValue": "z58DAdFfa9S..."
  }
}
```

RDF Dataset Canonicalization and Hash, VCs, and URDNA2015 (Dave Longley, 10-15 mins)

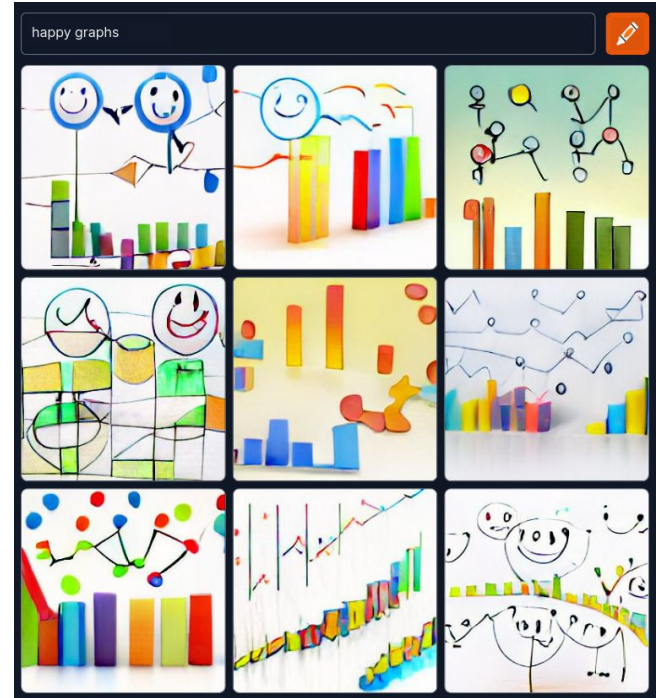
RDF and Verifiable Credentials

RDF (Resource Description Framework):

- RDF is a way to model graphs of information
- A graph is made up of nodes called "subjects" with links called "predicates" (aka "properties") to other values or subjects
- A dataset is a collection of one or more graphs

Verifiable Credentials Data Model:

- Claims are subject-property-value relationships
- Matches RDF... a framework that gives people a consistent, extensible way to:
 - Say anything about anything
 - Be globally unambiguous about it



-thanks, Orie!

What does "verifiable" mean and why?

What can we verify?

- The information in a credential hasn't changed since the issuer issued it
- The issuer is the one who issued it

Why do we need it?

- VC design intentionally creates separation between issuers and verifiers
- Increases holder privacy and expands the utility of credentials
- Verifiers don't "phone home" to the issuer to verify
- We need to verify that a VC is authentic without talking to the issuer



Cryptography FTW?

What do you "digitally sign"?

- Protection is over a specific set of bytes
- Different implementations may produce different bytes for the same VC (or the same dataset)

Why not just keep a copy of the exact bytes that were signed around all the time?

- Makes it easier on crypto implementers...
- Makes it harder on application developers!
- Failure to properly layer, now application developers have to care about the details and can't just get on with business as usual



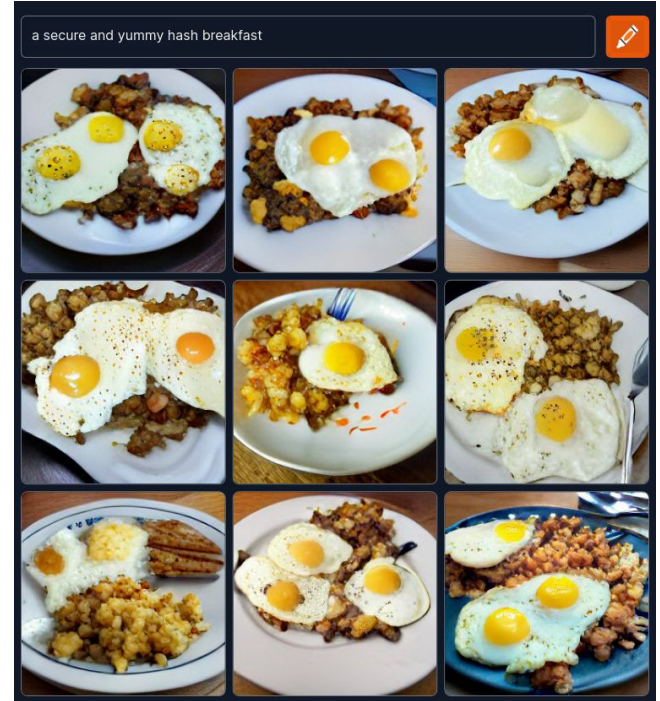
Canonicalization and Hash FTW

Canonicalization

- Most digital signatures sign a hash of the data: the hash is the bytes that are protected
- We can throw away the hash, because we trust the same data will always hash to the same output
- Canonicalization can do same job one layer up, ensuring implementations express data with the same bytes

Hash

- Canonize, hash, and then sign...
- Add the signature to the original data and don't keep duplicate intermediate forms around!



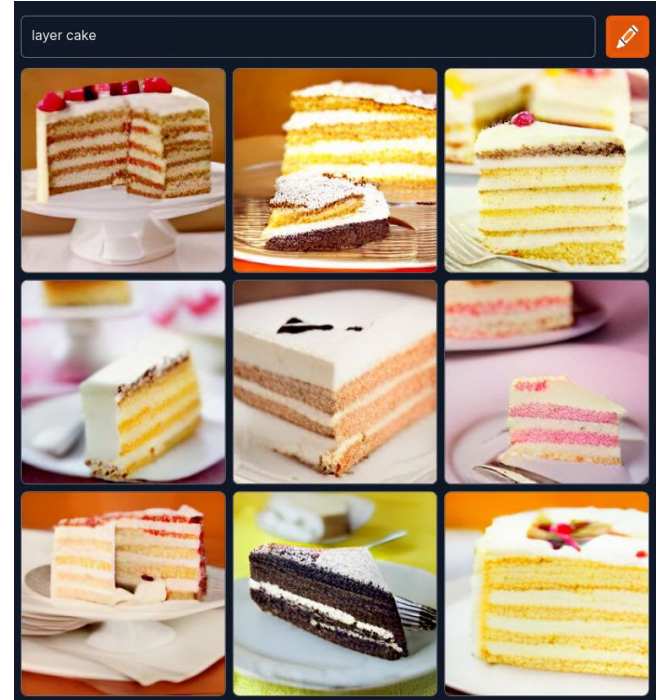
Trade offs

Cons

- Crypto implementers have to do a bit more work
- There's more security surface to get right

Pros (it's worth it!)

- Let application developers be application developers; use existing tools and paradigms without fear or specialization
- Decentralized innovation: experiment on your own without having to ask issuers to upgrade their systems and issue new formats with new signatures
- Transform the data (e.g., use CBOR-LD semantic-compression to fit a VC in a QR code!)



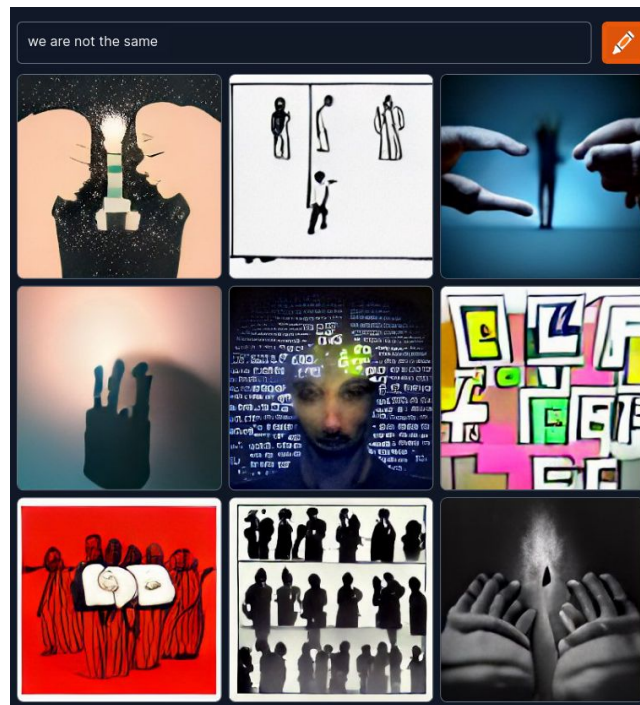
What needs canonizing and hashing?

Canonize anything that can be expressed differently:

- Subject identifiers; may be intentionally ambiguous for selective disclosure
- Expression of the data as bytes (order, syntax)

Decisions to be made:

- A canonicalization algorithm for deciding identifiers in the graph data
- A serialization format for canonicalized data that can be hashed using standard cryptographic hashes



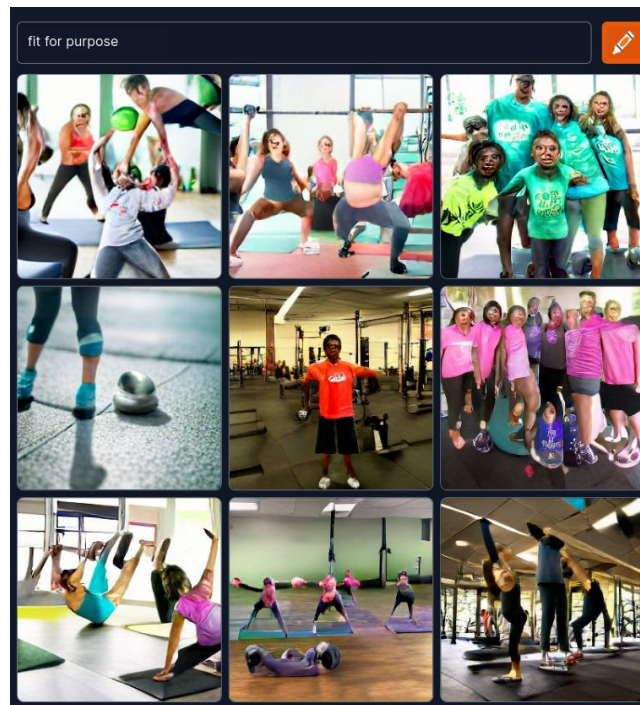
URDNA2015 - Universal RDF Dataset Normalization Algorithm

Based on URGNA2012, which was a single RDF graph canonicalization algorithm:

- Support for datasets and has a [W3C CCG spec](#)
- URDNA2015 has a correctness proof by Virginia Tech Professor of Mathematics Rachel Arnold, verified by Mirabolic Consulting (Princeton PhDs)
- Implementations in: JavaScript, Java, Rust, C++, Go, Python, Ruby, and C#

Design:

- Fit-for-purpose for RDF datasets; uses RDF relationships instead of graph abstractions
- Reuses existing primitives, N-Quads (an RDF serialization) and SHA-256 (a widely implemented cryptographic hash algorithm)



Questions?

RDF Dataset Canonicalization and Hash, VCs, and URDNA2015 (Aidan Hogan, 10-15 mins)

Slides in PPT and PDF

- PDF: <http://aidanhogan.com/rdfcanon/2022-09-16-rdf-canonicalisation.pdf>
- PPTX: <http://aidanhogan.com/rdfcanon/2022-09-16-rdf-canonicalisation.pptx>
- Slides are also copied below into Google Slides, but formatting may not be ideal

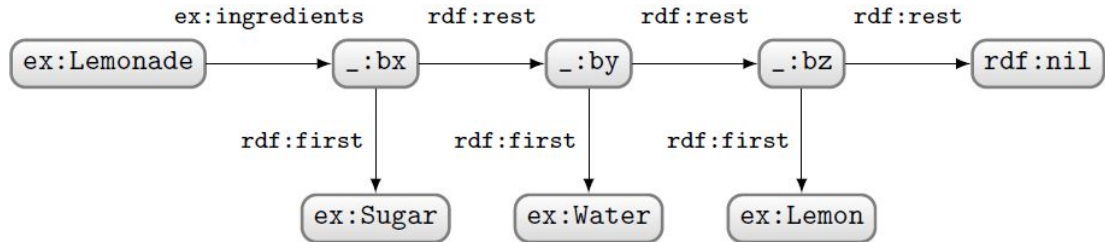
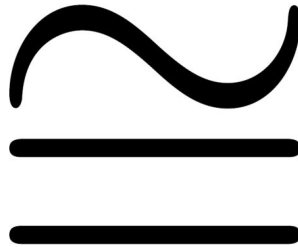
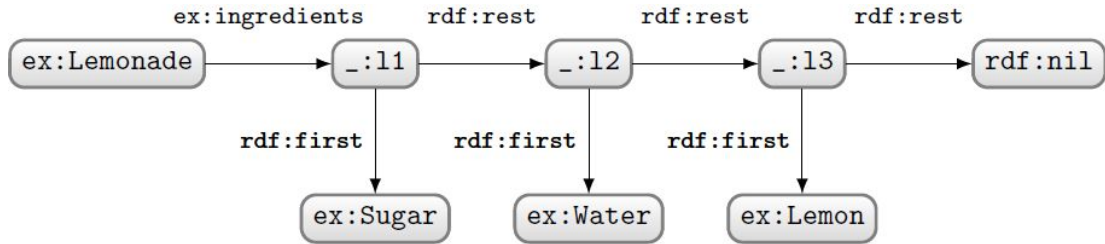
Blank nodes are common in real-world data ...

Nº	DOMAIN	BNODES	%BNODES	LOD?
1	data.gov.uk	54,898,287	27.39	✓
2	freebase.com	14,918,969	31.95	✓
3	livejournal.com	11,757,431	56.97	X
4	legislation.gov.uk	3,310,772	46.45	✓
5	ontologycentral.com	1,907,525	79.47	✓
6	vu.nl	658,538	37.27	✓
7	neuinfo.org	279,935	42.44	X
8	opera.com	233,578	6.89	X
9	geovocab.org	210,263	67.45	X
10	loc.gov	147,997	10.95	✓

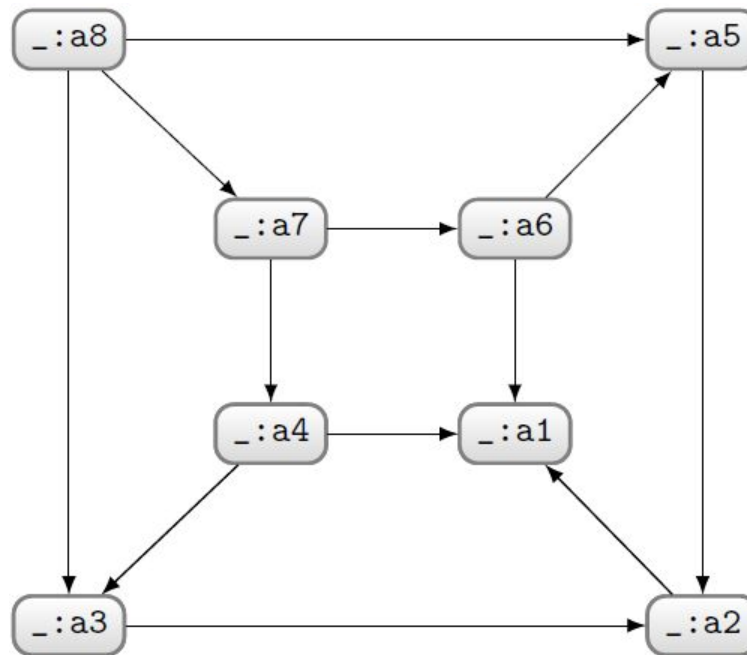
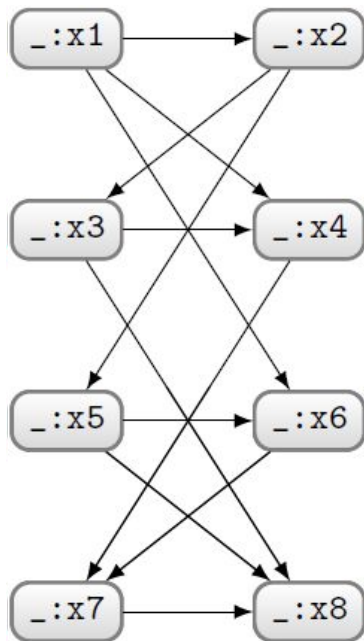
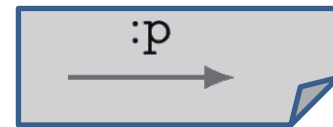
Aidan Hogan, Marcelo Arenas, Alejandro Mallea and Axel Polleres
"Everything You Always Wanted to Know About Blank Nodes".
Journal of Web Semantics 27: pp. 42–69, 2014

**BLANK NODES:
WHAT'S THE PROBLEM?**

Are two RDF graphs isomorphic?



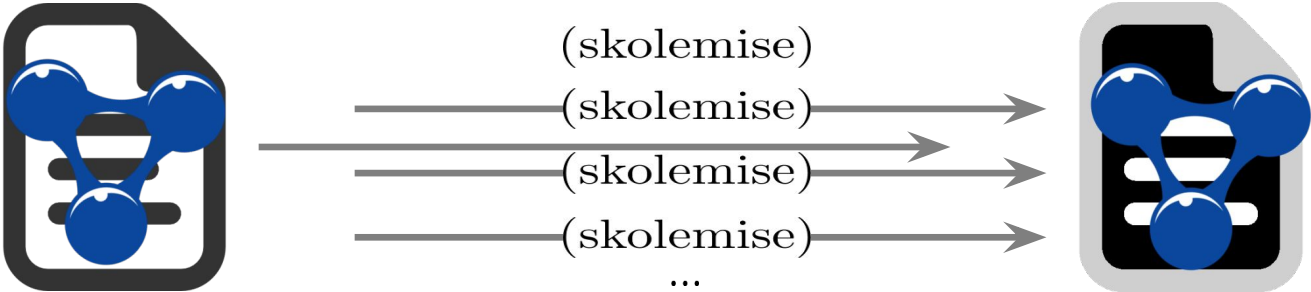
Are two RDF graphs isomorphic?



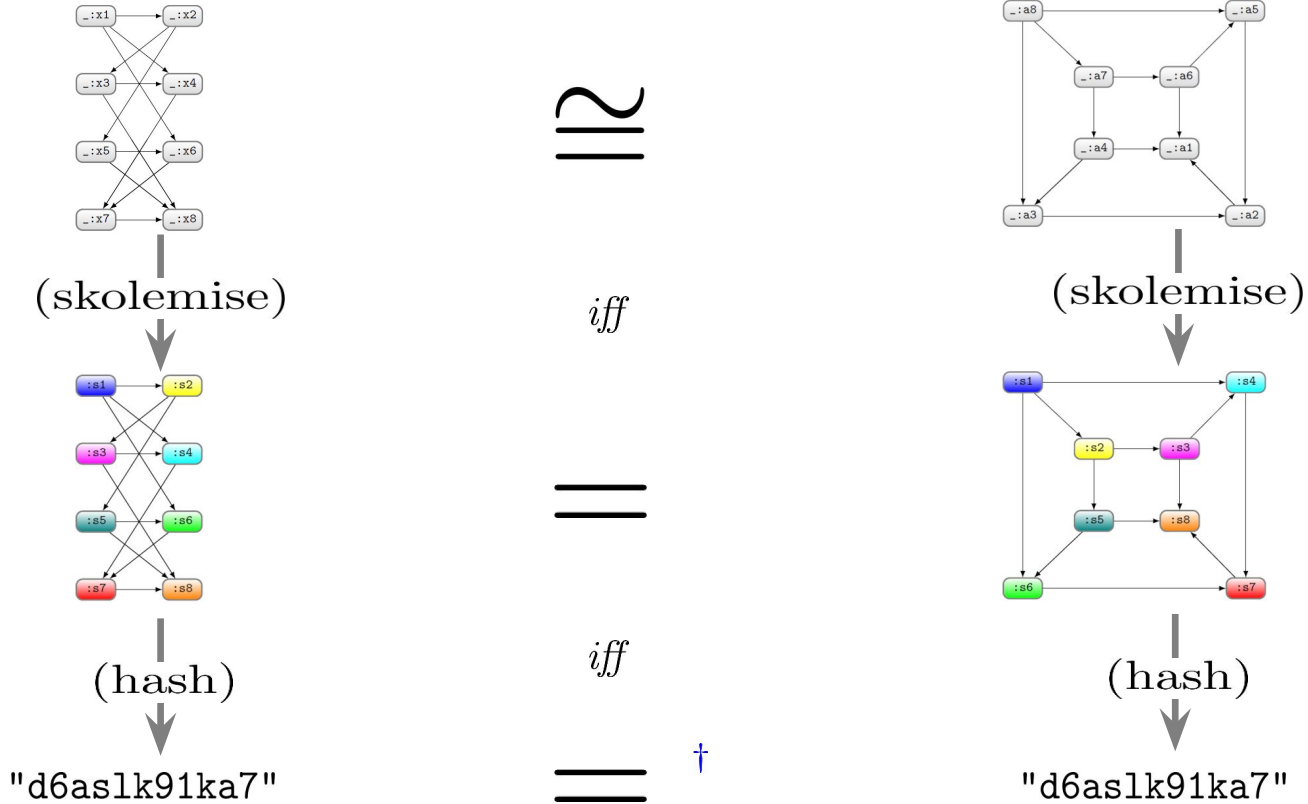
GI-COMPLETE

USE-CASES FOR CANONICAL LABELLING

Consistent Skolemisation

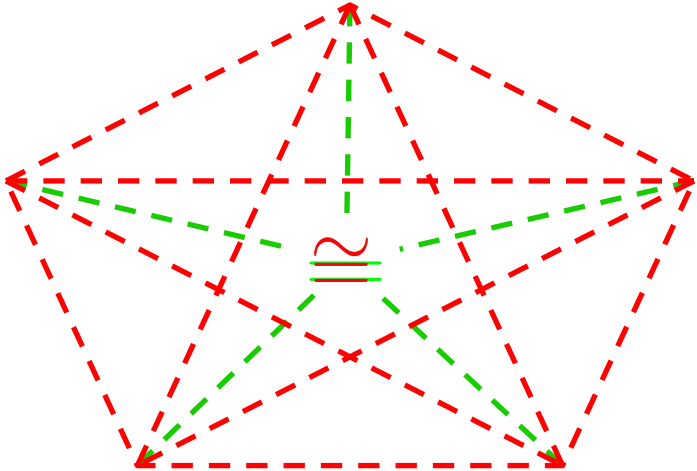
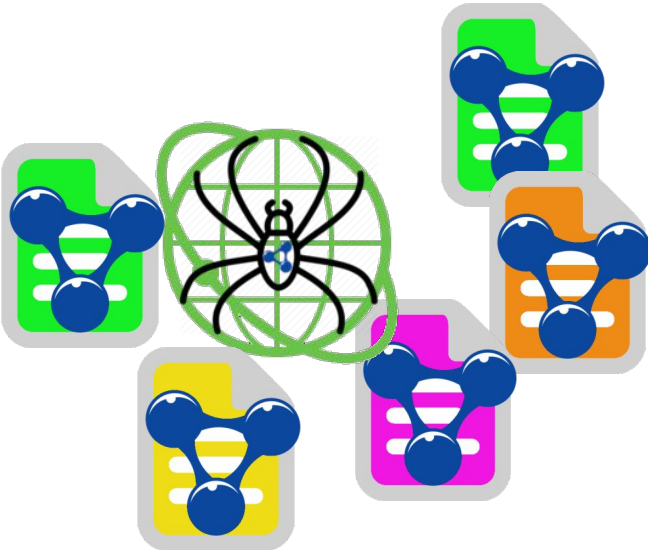


Hashing/signing graphs



† Assuming no hash collisions

Finding duplicate documents

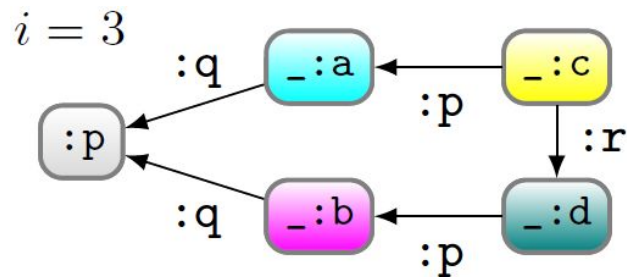
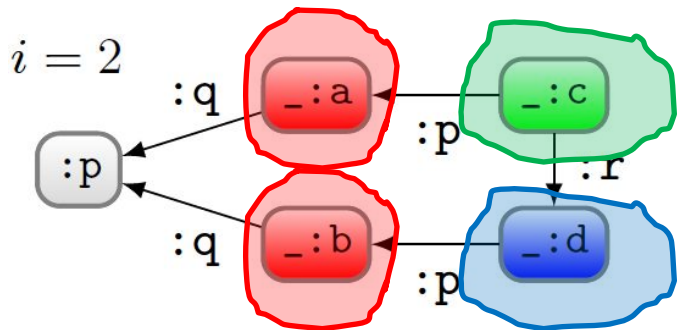
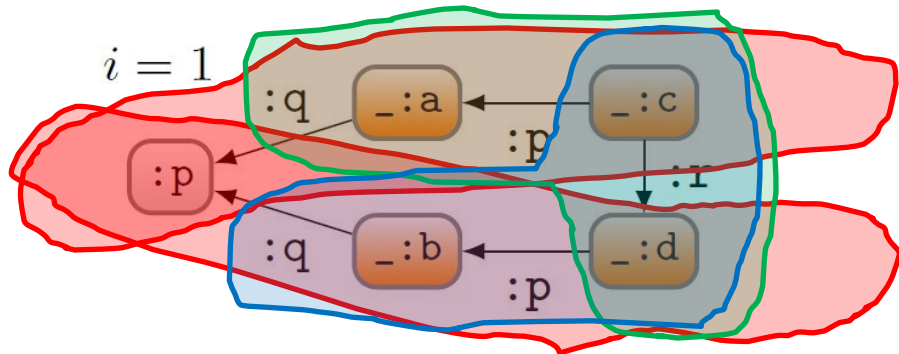
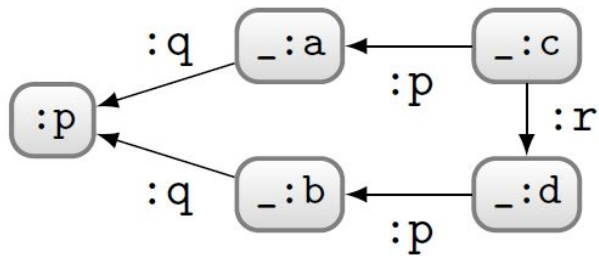


$O(n^2)$ pair-wise isomorphism checks

$O(n)$ hash checks

NAÏVE CANONICAL LABELLING SCHEME

(Naïve) Canonical labels for blank nodes



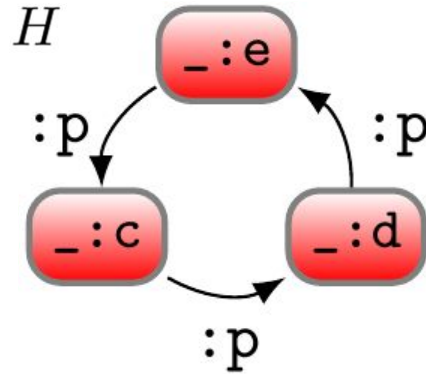
$$\text{hash}(:q, :p, +) \oplus \text{hash}(:p, \bullet, -) \oplus \bullet \rightarrow \bullet$$

$$\text{hash}(:q, :p, +) \oplus \text{hash}(:p, \bullet, -) \oplus \bullet \rightarrow \bullet$$

$$\text{hash}(:p, \bullet, +) \oplus \text{hash}(:r, \bullet, +) \oplus \bullet \rightarrow \bullet$$

$$\text{hash}(:p, \bullet, +) \oplus \text{hash}(:r, \bullet, -) \oplus \bullet \rightarrow \bullet$$

But wait ... what happens if ... ?



$$\begin{array}{c} \text{Aut}(H) \\ \hline \mu(\cdot) \quad _ :c \quad _ :d \quad _ :e \\ \hline _ :c \quad _ :d \quad _ :e \\ = \quad _ :d \quad _ :e \quad _ :c \\ _ :e \quad _ :c \quad _ :d \\ \hline \end{array}$$

$$\text{hash}(:p, \bullet, +) \oplus \text{hash}(:p, \bullet, -) \oplus \bullet \rightarrow \bullet$$

$$\text{hash}(:p, \bullet, +) \oplus \text{hash}(:p, \bullet, -) \oplus \bullet \rightarrow \bullet$$

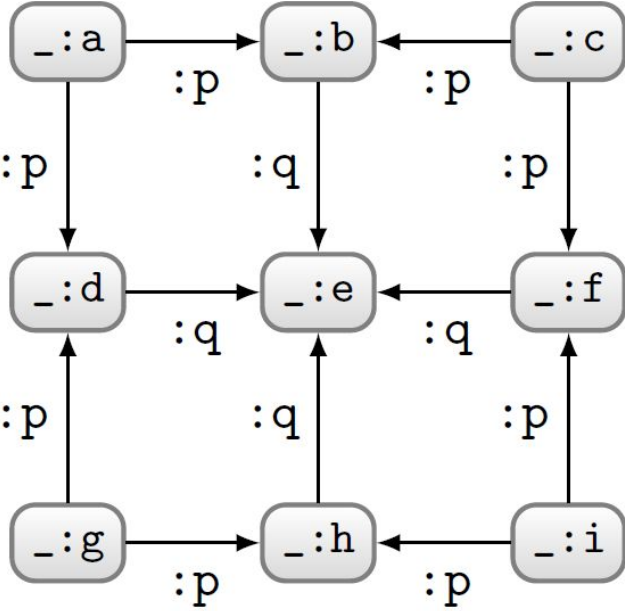
$$\text{hash}(:p, \bullet, +) \oplus \text{hash}(:p, \bullet, -) \oplus \bullet \rightarrow \bullet$$

$$\text{hash}(:p, \bullet, +) \oplus \text{hash}(:p, \bullet, -) \oplus \bullet \rightarrow \bullet$$

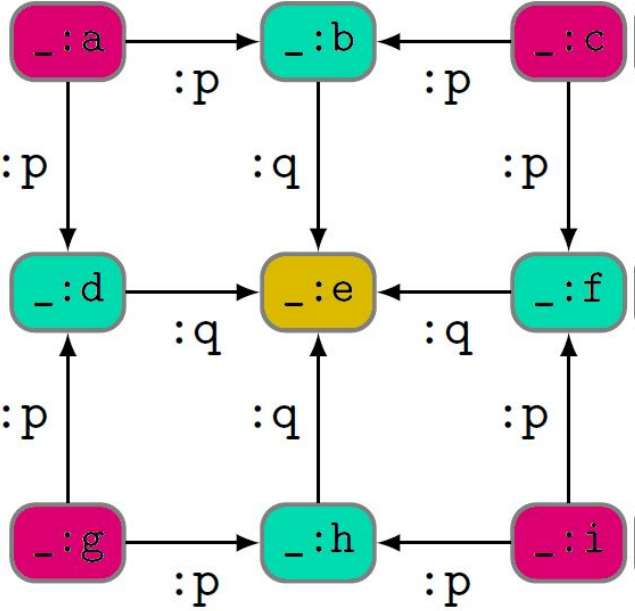
$$\text{hash}(:p, \bullet, +) \oplus \text{hash}(:p, \bullet, -) \oplus \bullet \rightarrow \bullet$$

$$\text{hash}(:p, \bullet, +) \oplus \text{hash}(:p, \bullet, -) \oplus \bullet \rightarrow \bullet$$

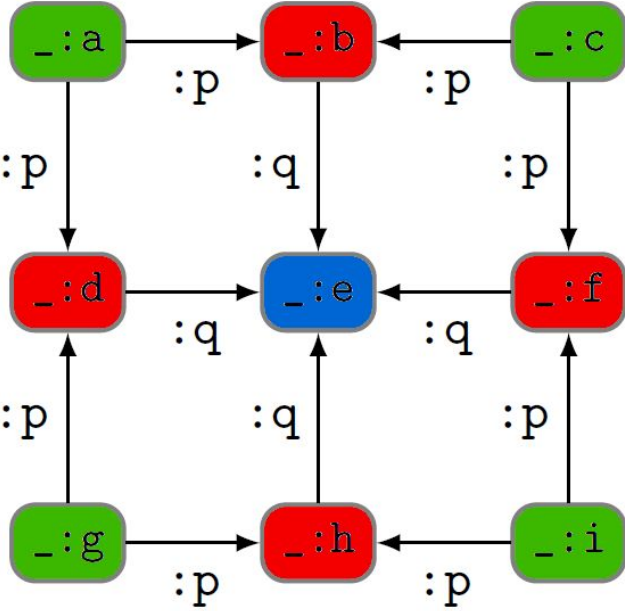
Or another case ...



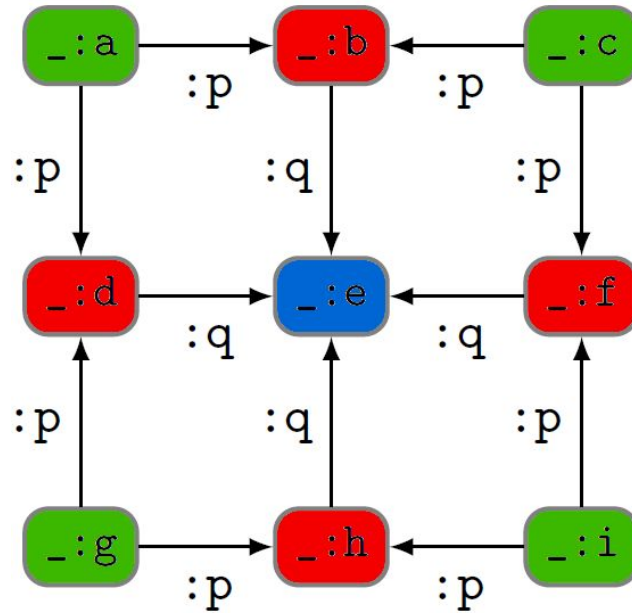
Or another case ...



Or another case ...



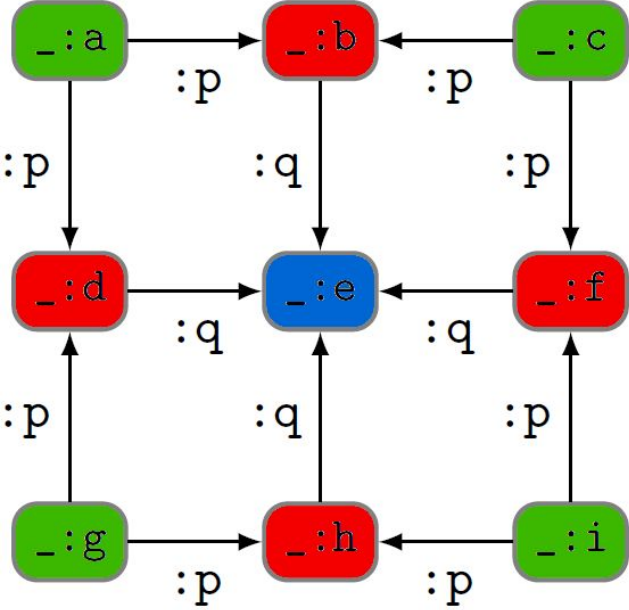
Fixpoint does not distinguish all blank nodes!



e	a,c,g,i	b,d,f,h
---	---------	---------

CANONICAL LABELLING SCHEME: ALWAYS DISTINGUISH ALL BLANK NODES

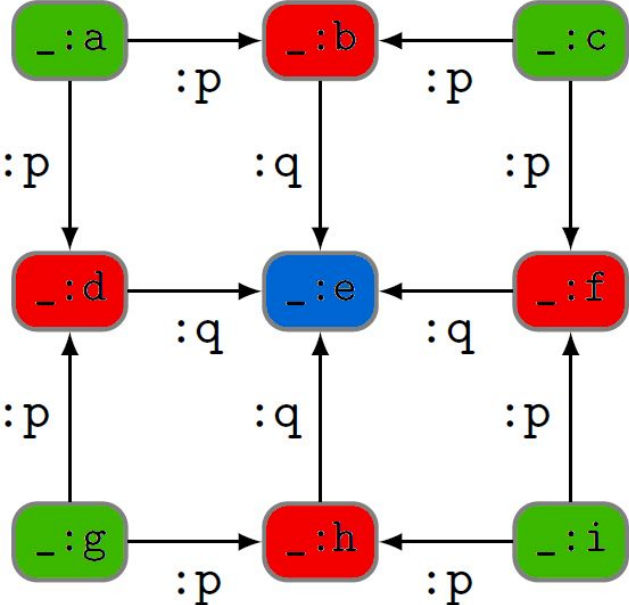
Start with a (non-distinguished) colouring ...



e	a,c,g,i	b,d,f,h
---	---------	---------

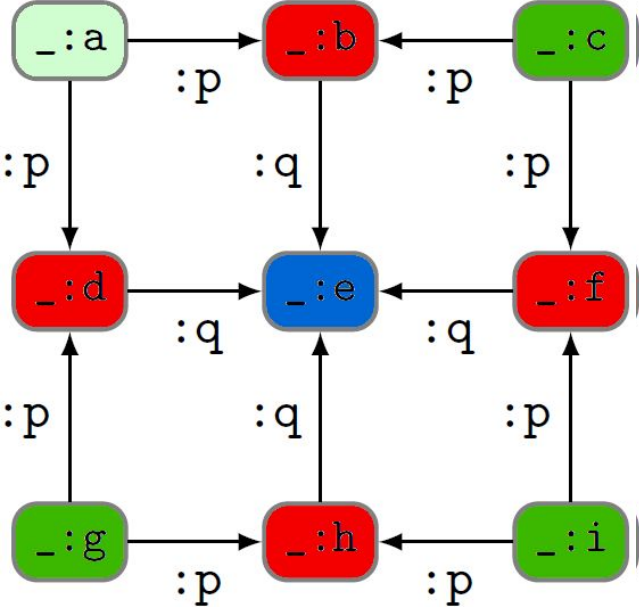
Let's distinguish a node ...

Let's select `_:a` for now ↓



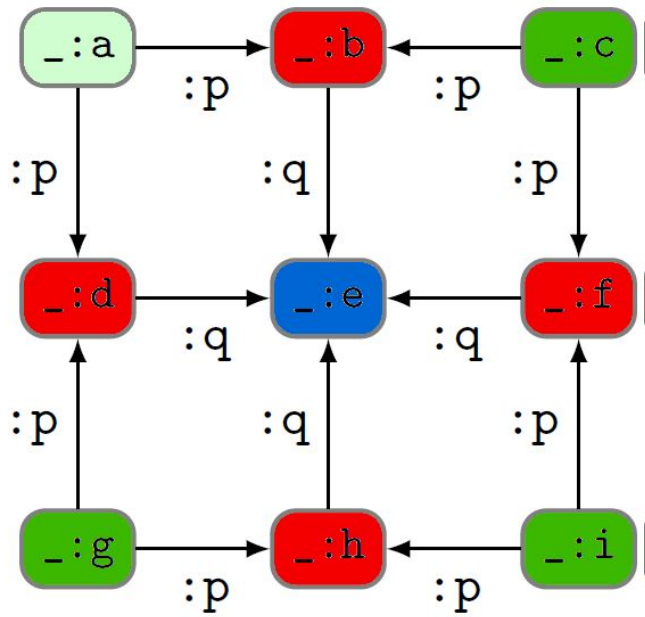
Let's distinguish a node ...

Let's select `_:a` for now ↓



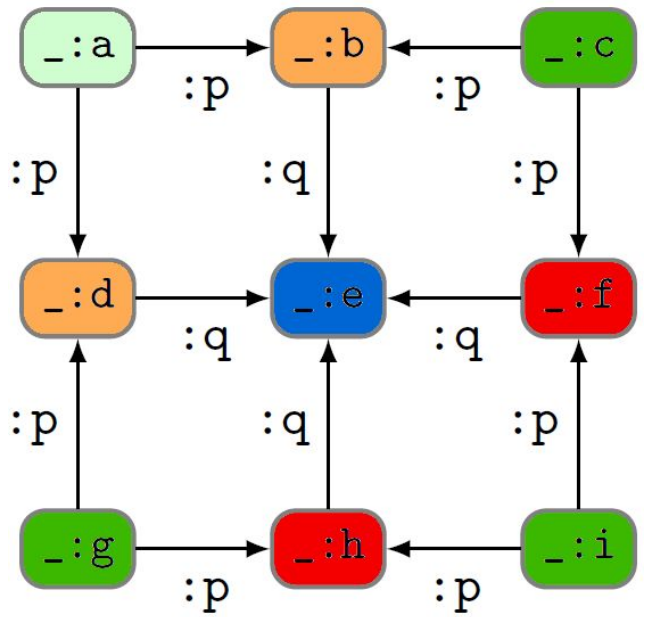
Colouring is no longer a fixpoint!

Let's select `_:a` for now ↓



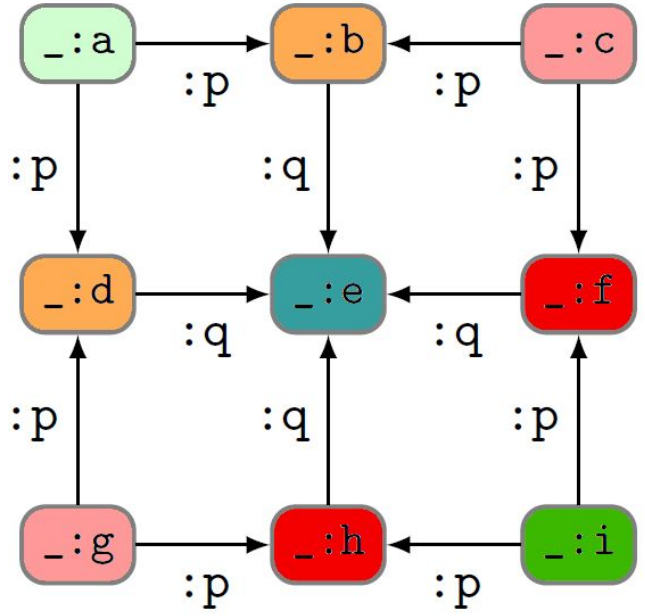
Rerun colouring to fixpoint

Let's select `_:a` for now ↓



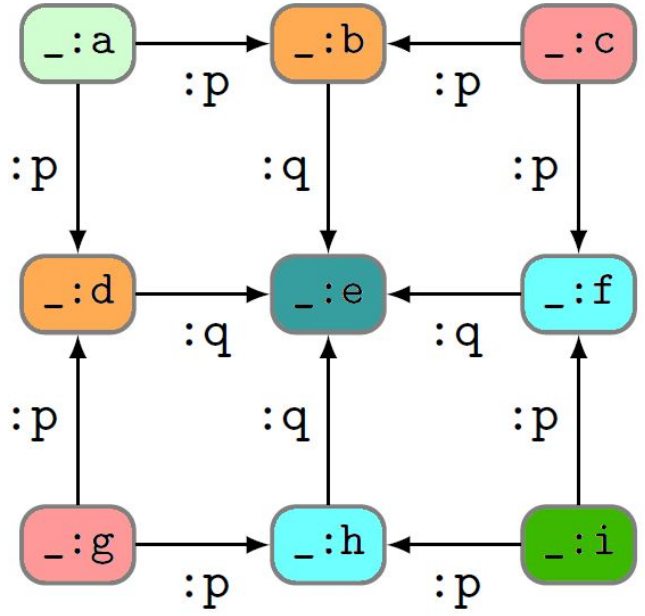
Rerun colouring to fixpoint

Let's select `_:a` for now ↓



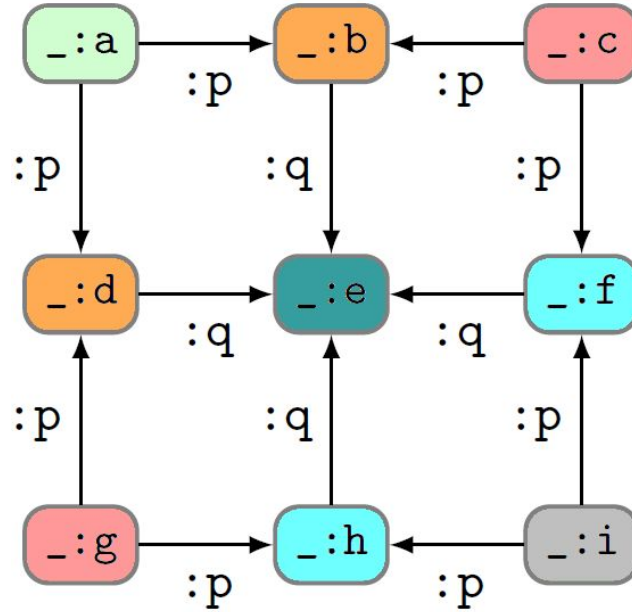
Rerun colouring to fixpoint

Let's select `_:a` for now ↓



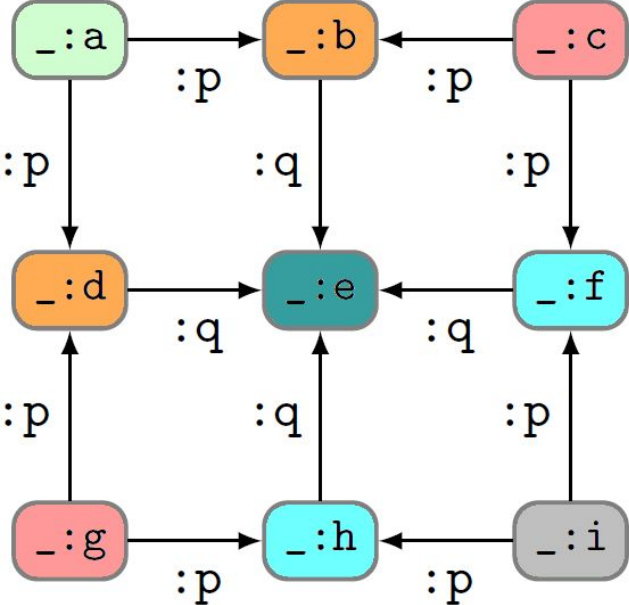
Rerun colouring to fixpoint

Let's select `_:a` for now ↓



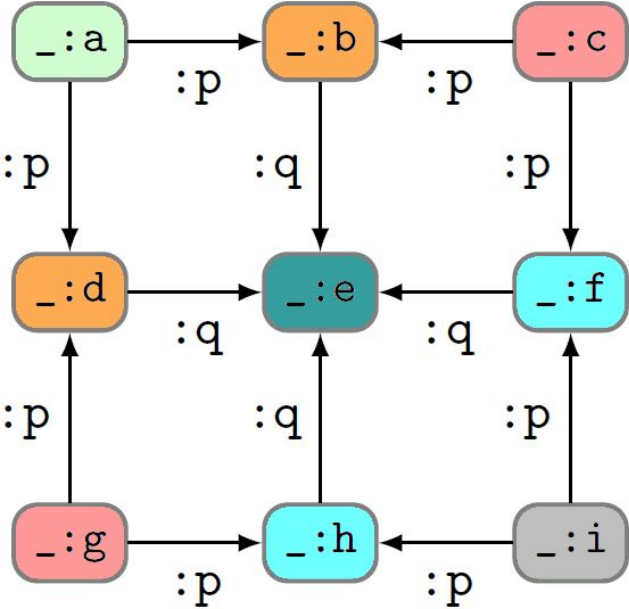
e	a	i	c,g	b,d	f,h
---	---	---	-----	-----	-----

Fixpoint reached: still not finished!



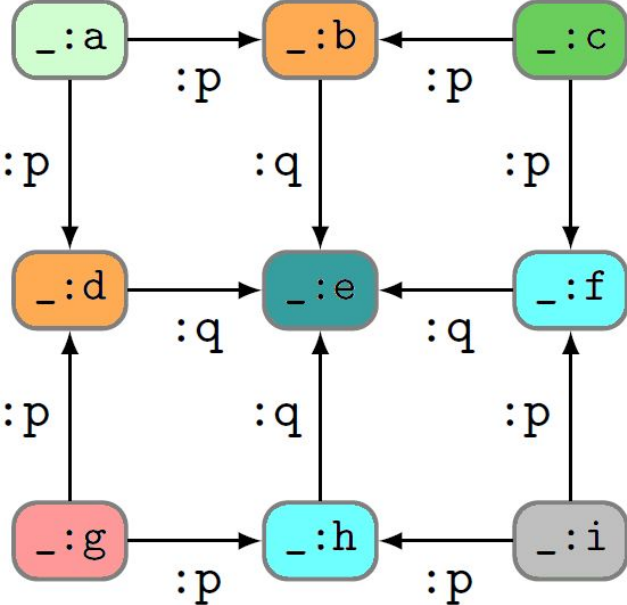
So again let's distinguish another ...

Next let's select `_:c` for now ↓



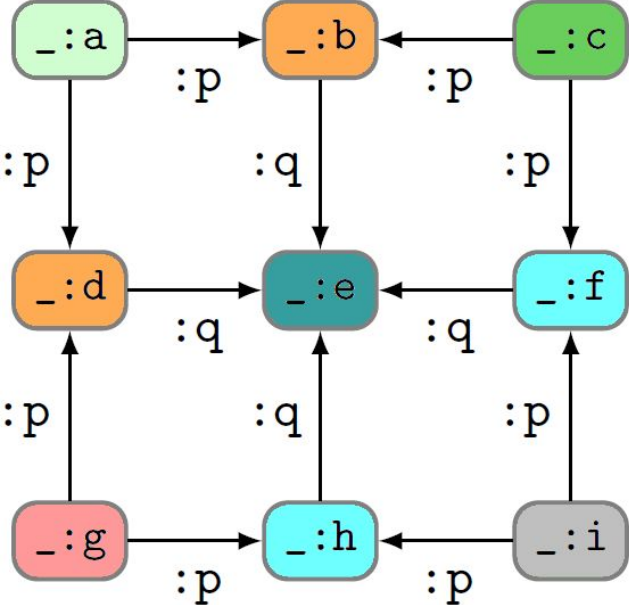
... and rerun colouring to fixpoint

Next let's select `_:c` for now ↓



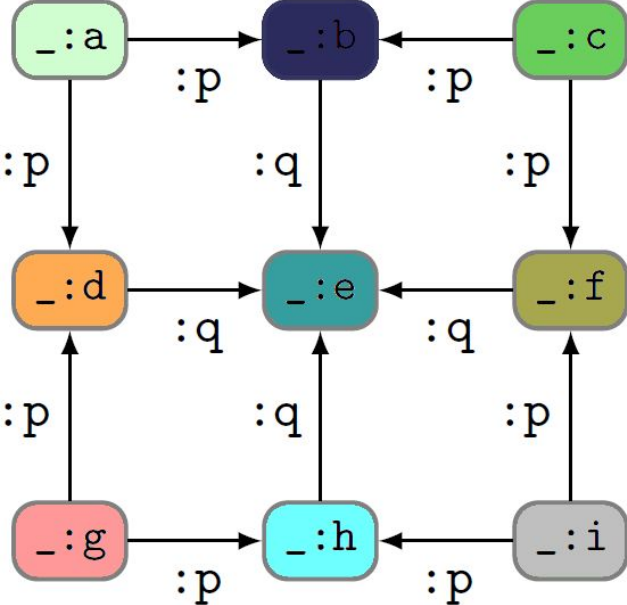
... and rerun colouring to fixpoint

Next let's select `_:c` for now ↓



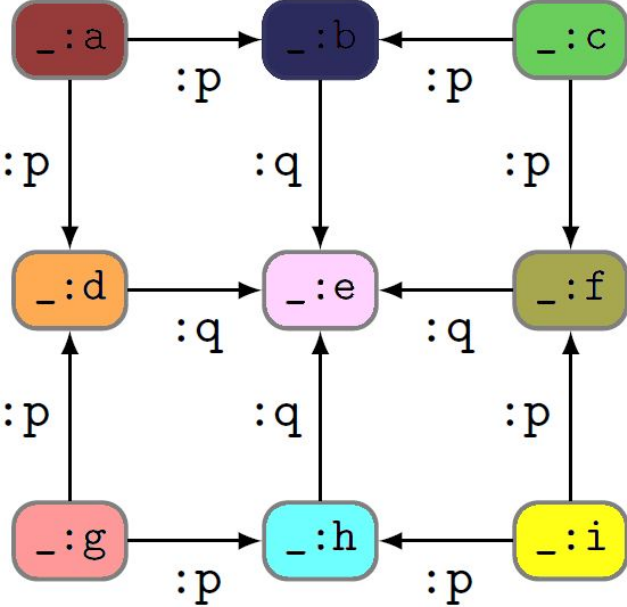
... and rerun colouring to fixpoint

Next let's select `_:c` for now ↓



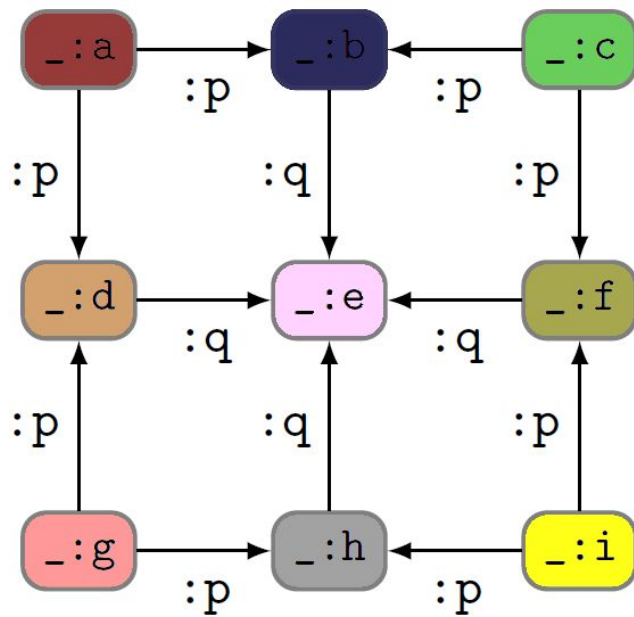
... and rerun colouring to fixpoint

Next let's select `_:c` for now ↓



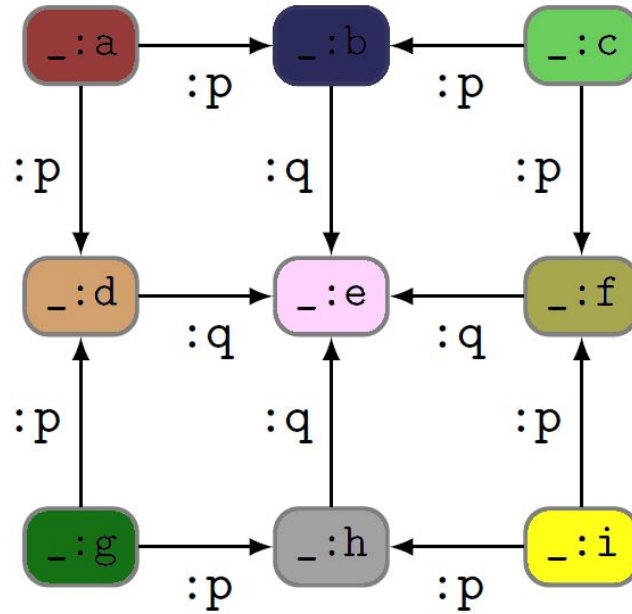
... and rerun colouring to fixpoint

Next let's select `_:c` for now ↓



... and rerun colouring to fixpoint

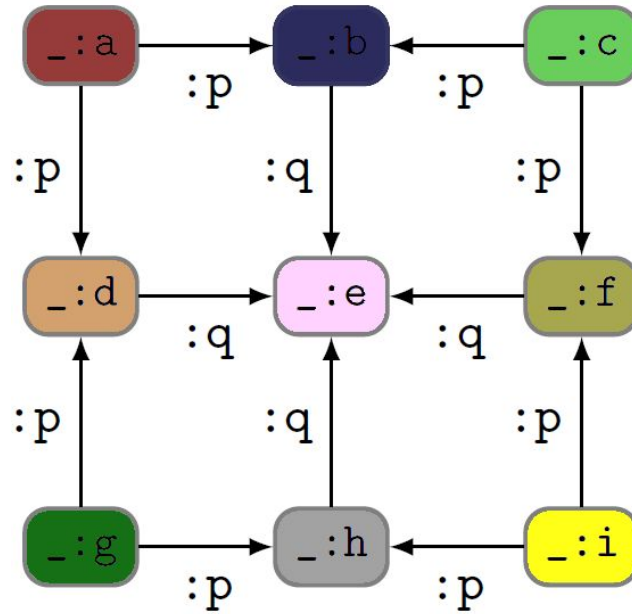
Next let's select `_:c` for now ↓



e	a	i	c	g	b	d	f	h
---	---	---	---	---	---	---	---	---

Now all blank nodes are distinguished!

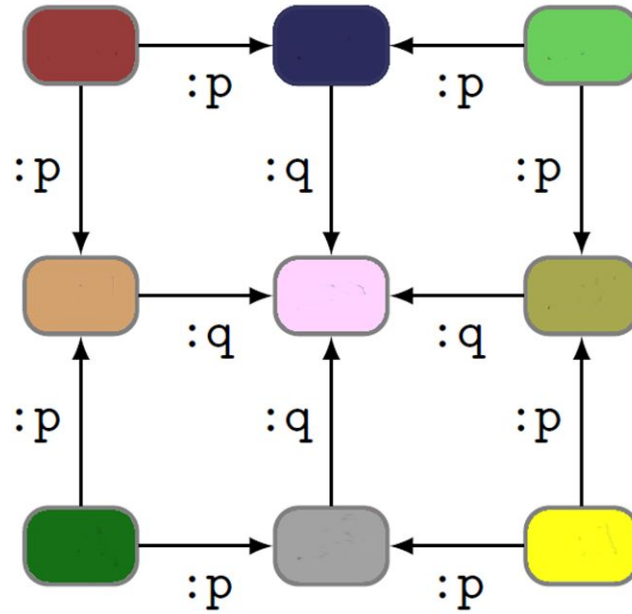
Next let's select `_:c` for now ↓



e	a	i	c	g	b	d	f	h
---	---	---	---	---	---	---	---	---

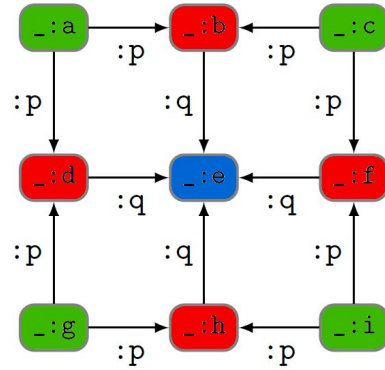
Blank node labels computed from colour

Next let's select `_:c` for now ↓



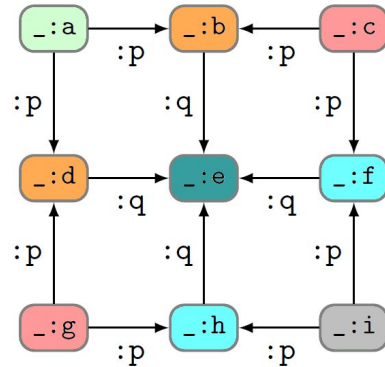
Let's go back: first, why pick `_:a` and `_:c`?

Let's select `_:a` for now ↓



Why not `_:b`, `_:d` ...?

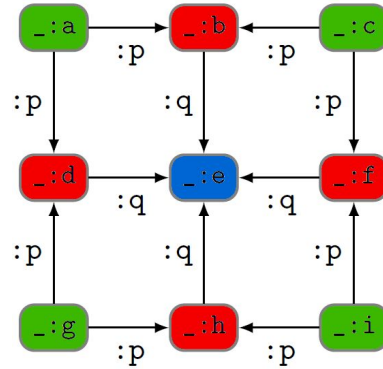
Next let's select `_:c` for now ↓



Okay so: why `_:a` ...

Let's select `_:a` for now ↓

But why not `_:b`, `_:c`, ...



Main idea (*sketch*):

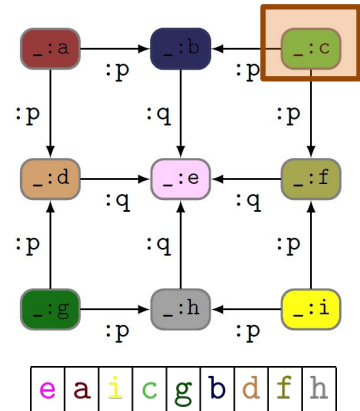
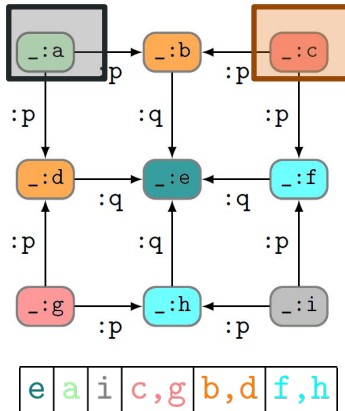
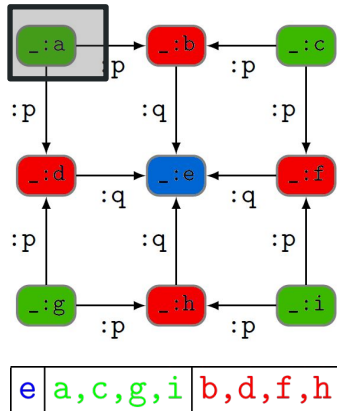
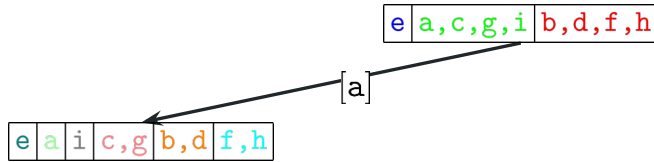
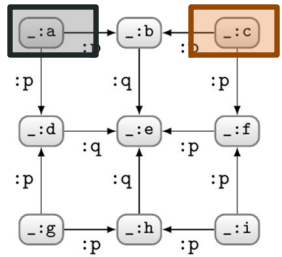
- define (deterministic) total ordering of colours; e.g., $\bullet < \color{green}\bullet < \color{red}\bullet$
- choose to distinguish node from lowest non-trivial colour

e	a, c, g, i	b, d, f, h
---	------------	------------

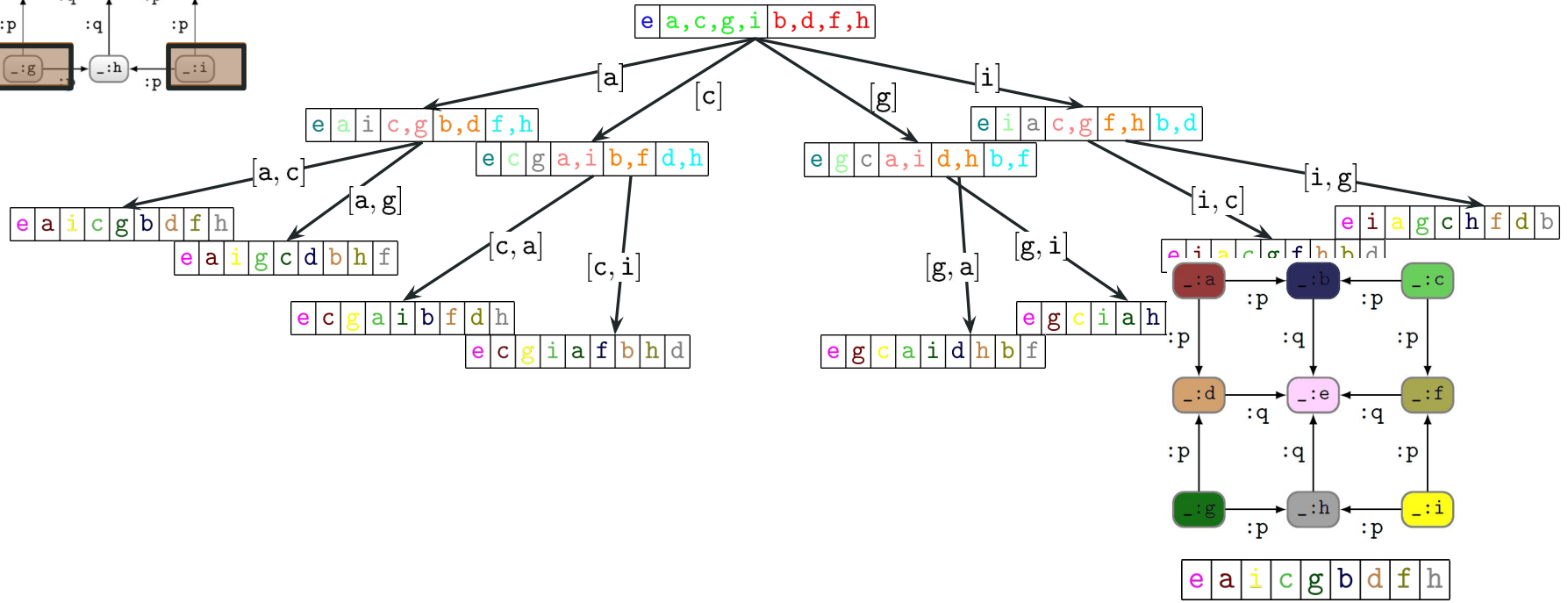
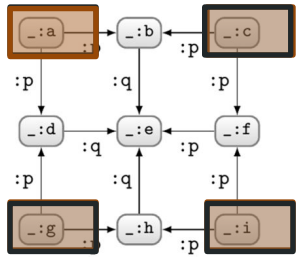
- but how can we choose between `_:a`, `_:c`, `_:g`, `_:i`?

We cannot. We need to try them all.

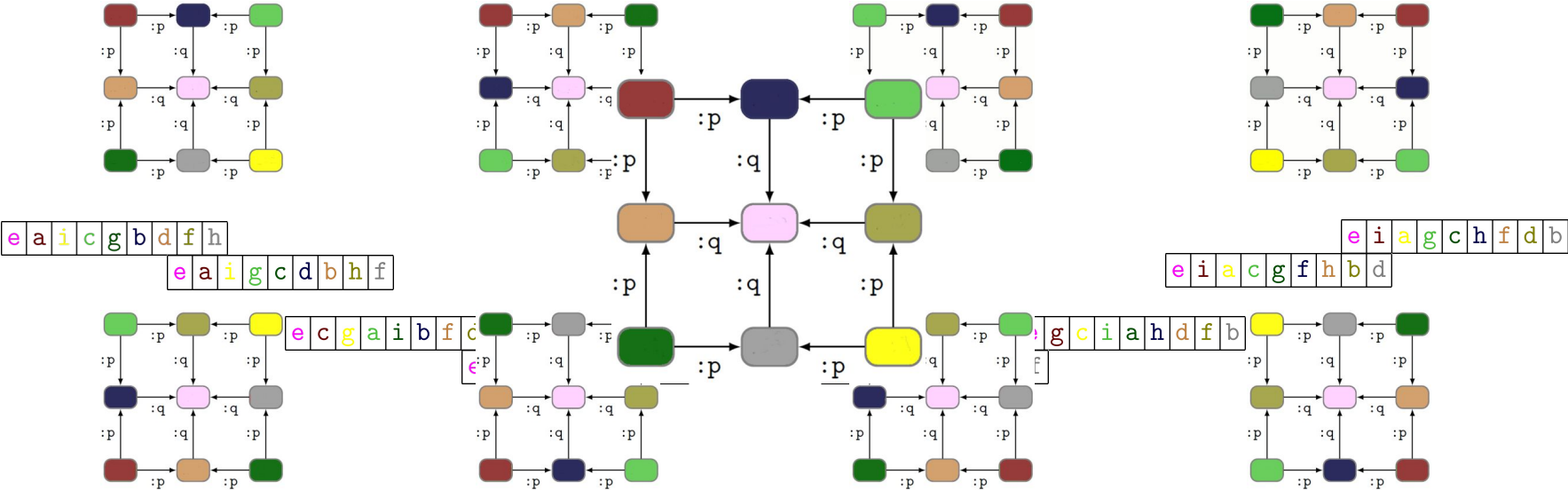
Adapt ideas from the Nauty algorithm (for standard graph isomorphism)



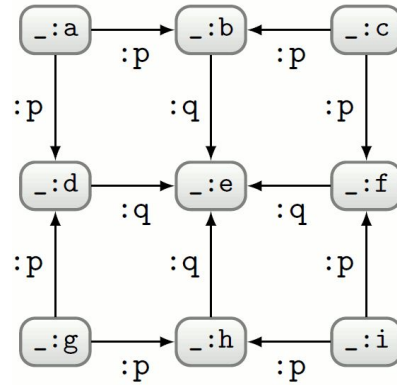
Adapt ideas from the Nauty algorithm (for standard graph isomorphism)



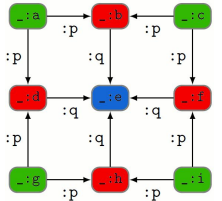
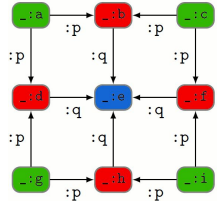
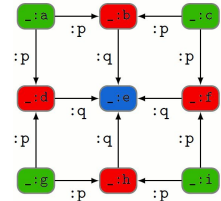
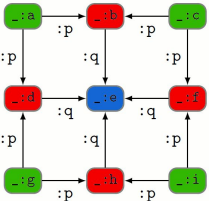
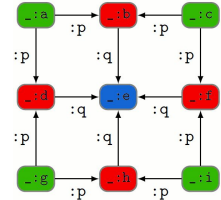
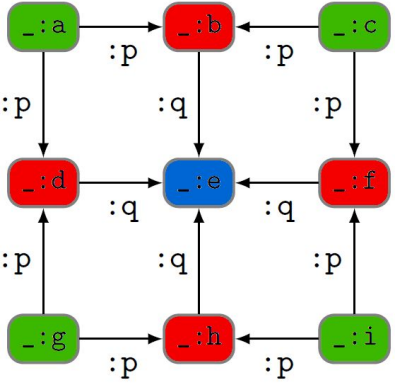
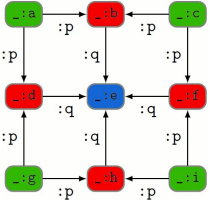
Check all leafs for minimum graph



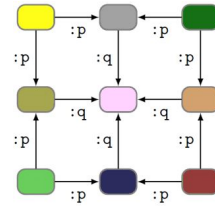
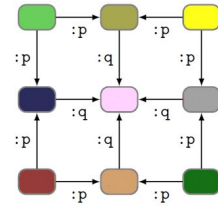
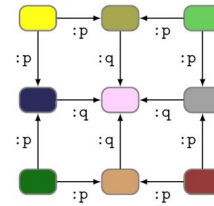
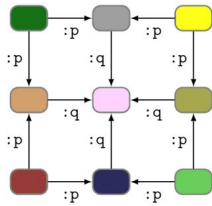
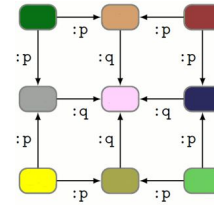
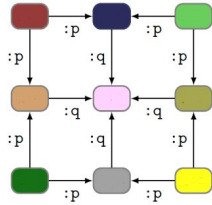
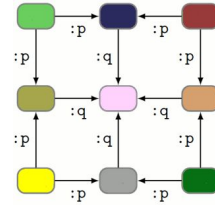
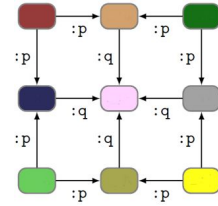
What happened?



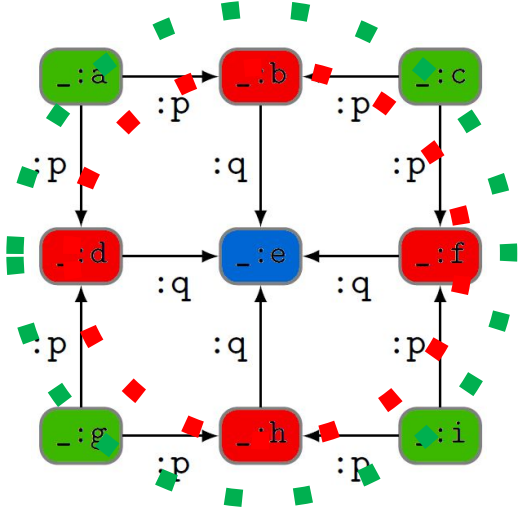
What happened?



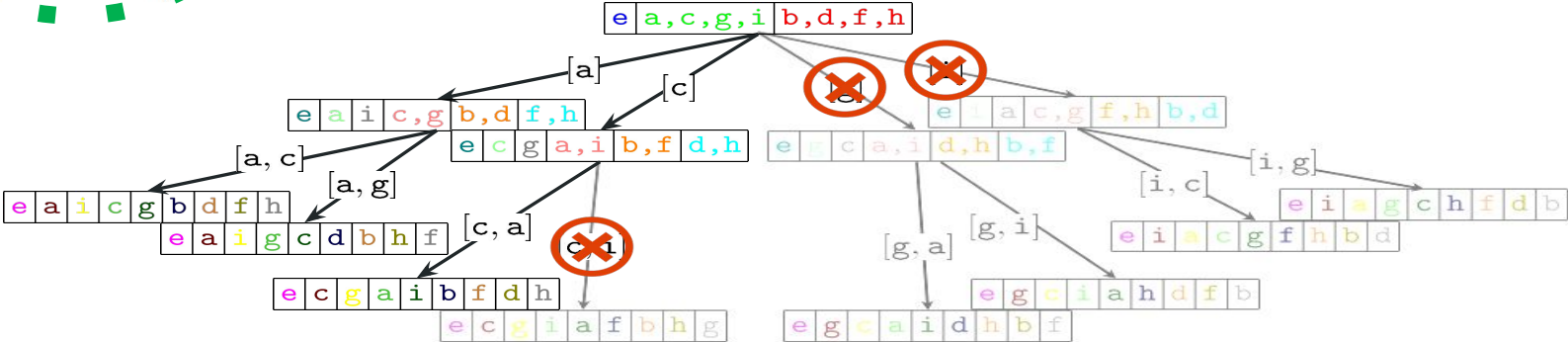
What happened?



Automorphisms cause repetitions



Can optimise for this!



EVALUATION

Evaluation: Real-world Graphs

- BTC-14 dataset:
 - 43.6 million RDF graphs (each from a Web doc.)
 - 9.8 million RDF graphs with blank nodes
 - 47,560 pay-level domains
 - 4 billion statements
- Single thread, 30GB Heap, 2.2GHz

Hash	Bits	Runtime (<i>h</i>)	Adjusted (<i>h</i>)
MD5	128	16.4	12.4
MURMUR3_128	128	13.6	9.6
SHA1	160	16.5	12.5

- ~4ms per graph with blank nodes
- Slowest graph: 35 s. (7.3 million triples; 254 thousand b-nodes)

Evaluation: Nasty Synthetic Graphs

- Harder cases converted to RDF from standard isomorphism benchmark
- Run on this laptop (1GB heap, 2.4GHz, single-threaded)
- 10 minute timeout

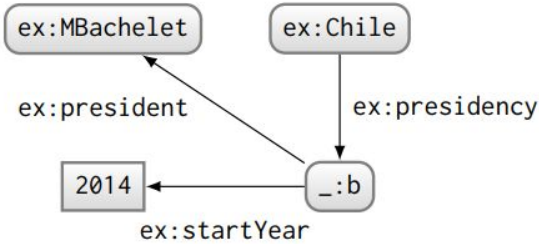
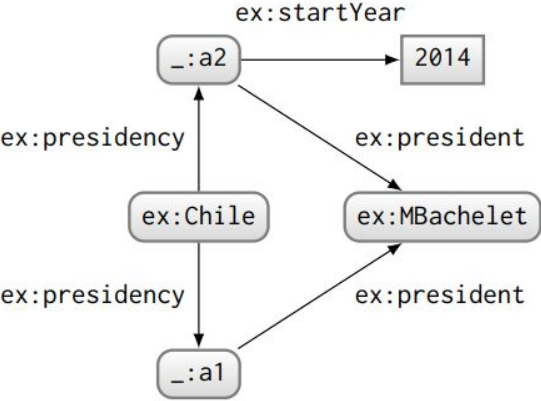
Class	k	Triples	BNodes	Time (<i>ms</i>)
GRID 2D	15	840	225	955
	100	39,600	10,000	58,482
GRID 3D	7	1,764	343	938
	19	38,988	6,589	37,402
TRIANGLE	9	504	36	656
	29	21,924	406	507,036
	30	24,360	435	—
LATTICE	6	360	36	532
	18	11,016	324	371,073
	19	12,996	361	—
CLIQUE	16	240	16	719
	55	2,970	55	578,830
	56	3,080	56	—
MIYAZAKI	2	120	40	140
	8	480	160	195,770
	10	600	200	—

Largest graph < 1 s.

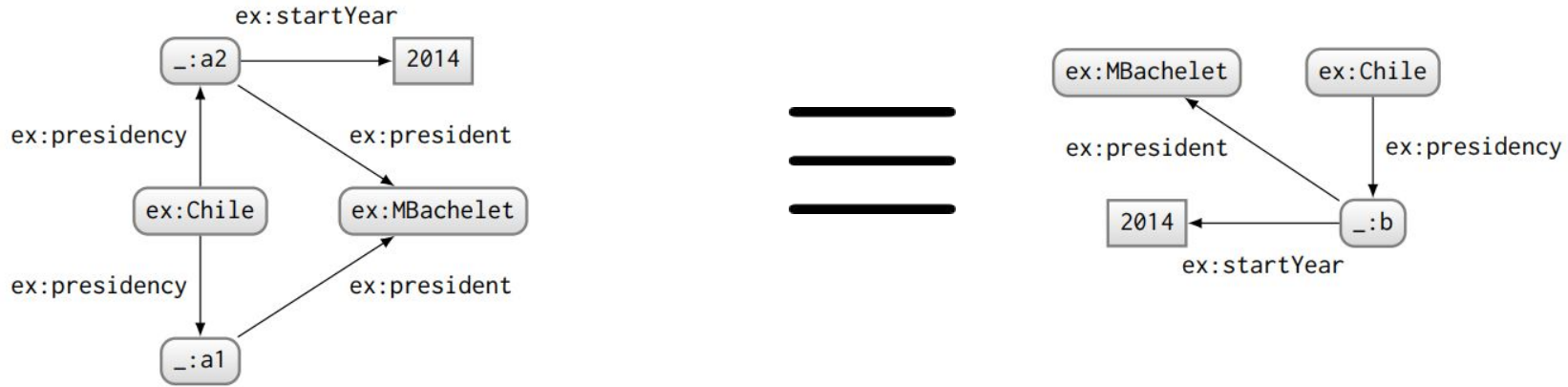
All graphs < 600 s.

EQUI-CANONICALISATION

Equivalence of RDF graphs



Equivalence of RDF graphs



- Canonicalisation algorithm for simple equivalence
 1. Load both RDF graphs
 2. Label them (as before)

CONCLUSIONS

Potential Inputs

- Publications:
 - Aidan Hogan. "SKOLEMISING BLANK NODES WHILE PRESERVING ISOMORPHISM". In the Proceedings of the 24th International World Wide Web Conference (WWW), Florence, Italy, May 18–22, 2015.
 - Aidan Hogan. "CANONICAL FORMS FOR ISOMORPHIC AND EQUIVALENT RDF GRAPHS: ALGORITHMS FOR LEANING AND LABELLING BLANK NODES". In ACM Transactions on the Web 11(4): 22:1-22:62, 2017.
- Reference Implementations:
 - Blabel (Java), Aidan Hogan: <https://blabel.github.io/>
 - Canonical RDF (`node.js`*), Ivan Herman: <https://zenodo.org/record/3154323>
 - Gonum/RDF (Go), Dan Kortschak: <https://pkg.go.dev/gonum.org/v1/gonum@v0.11.0/graph/formats/rdf>
- Verification using Coq Framework (ongoing)

Conclusions

- Algorithm is proven to be sound and complete
 - Ongoing: Automated verification in Coq (type theory)
 - Based heavily on graph isomorphism literature
- Algorithm is exponential in the worst-case
 - Efficient for “real world graphs” of moderate size (in memory)
 - Relatively efficient for regular graphs (e.g., 16-cliques in <1 sec)
 - Inefficient for exotic/constructed worst cases (e.g., Miyazaki)
- Algorithm has been (independently) implemented
 - Two independent implementations (with some help from me)
 - Automorphism optimisation tricky, but not necessary
- Some details would need to be fixed
 - Hashing algorithm used
 - Function to combine hashes (ordered and unordered)
 - Total ordering of elements (IRIs, literals, triples, graphs)
 - Behaviour in case of hash collisions
 - Extension to RDF datasets (should not be difficult)
 - Dataset canonicalisation (should not be difficult)

Lunch
(60 mins)

Delegated and Multi-Party Credentials (Gabe Cohen, Ori Steele 40 mins)

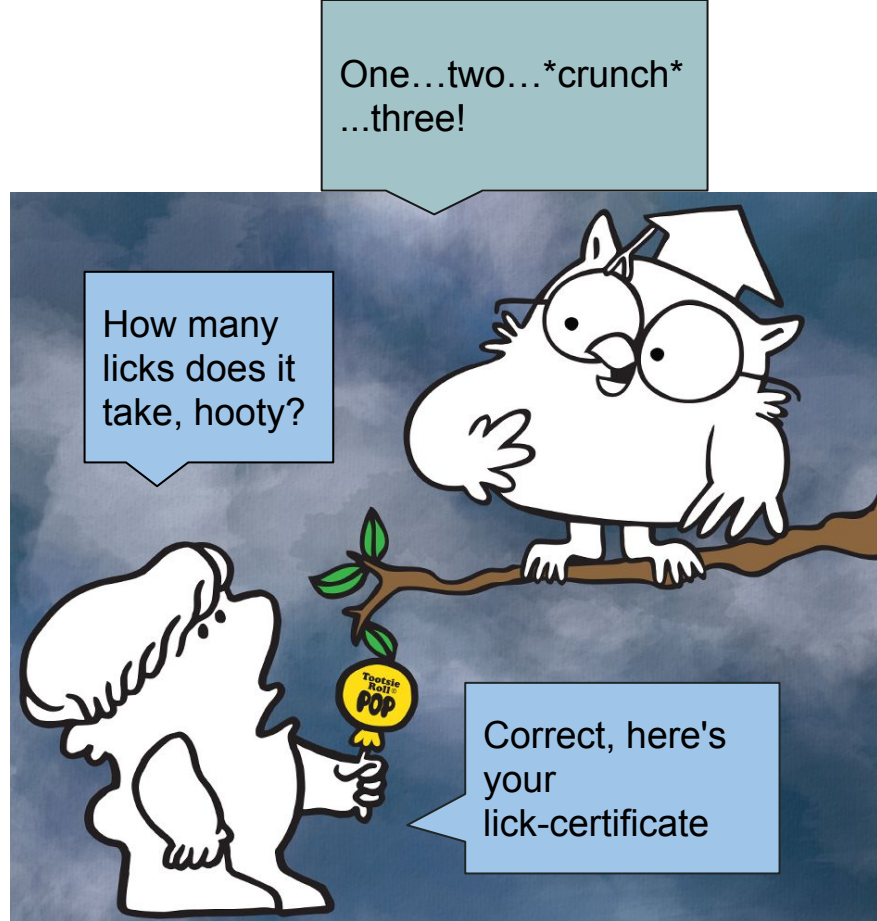
Credentials Today

Normally...

- 1 issuer
- 1 subject

Exhibit A:

- Tootsie man = issuer
- Mr. Owl = subject



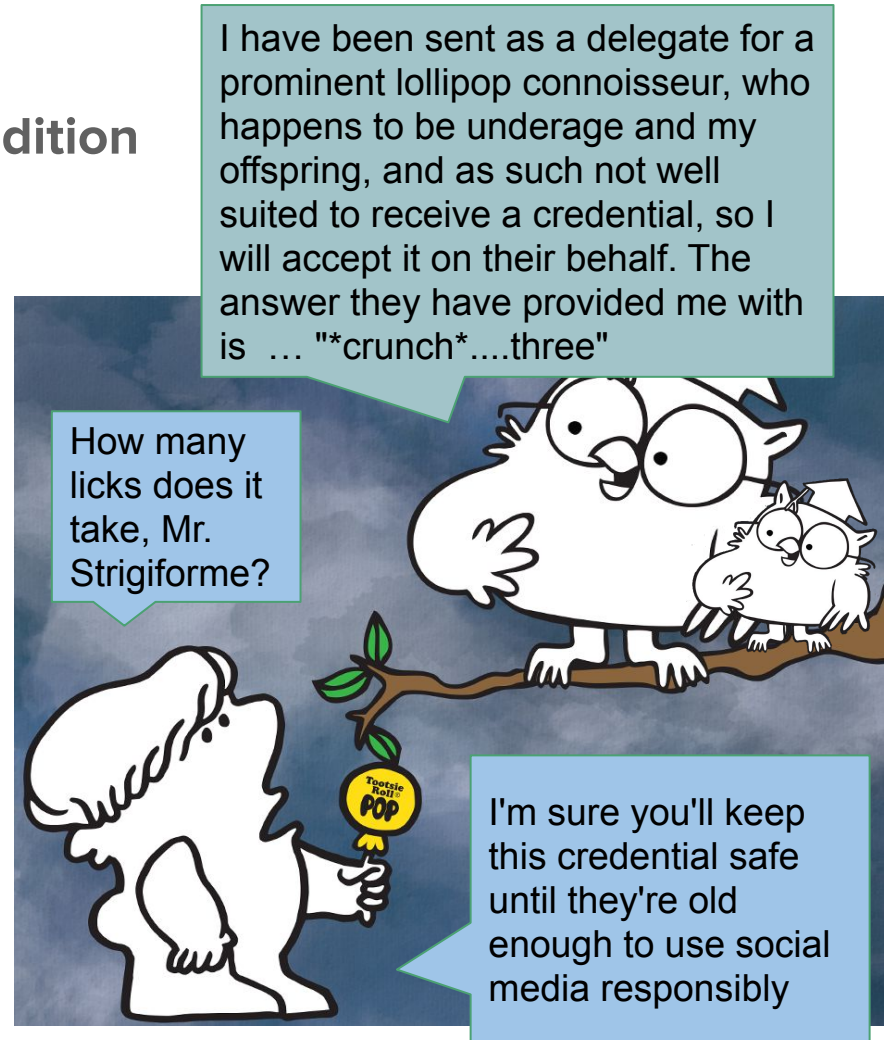
Credentials Tomorrow: Delegated Edition

Normally....

- 1 issuer
- 1 subject

Exhibit A:

- Tootsie man = issuer
- Mr. Owl = delegate subject
- Mr. Owl's offspring = primary subject



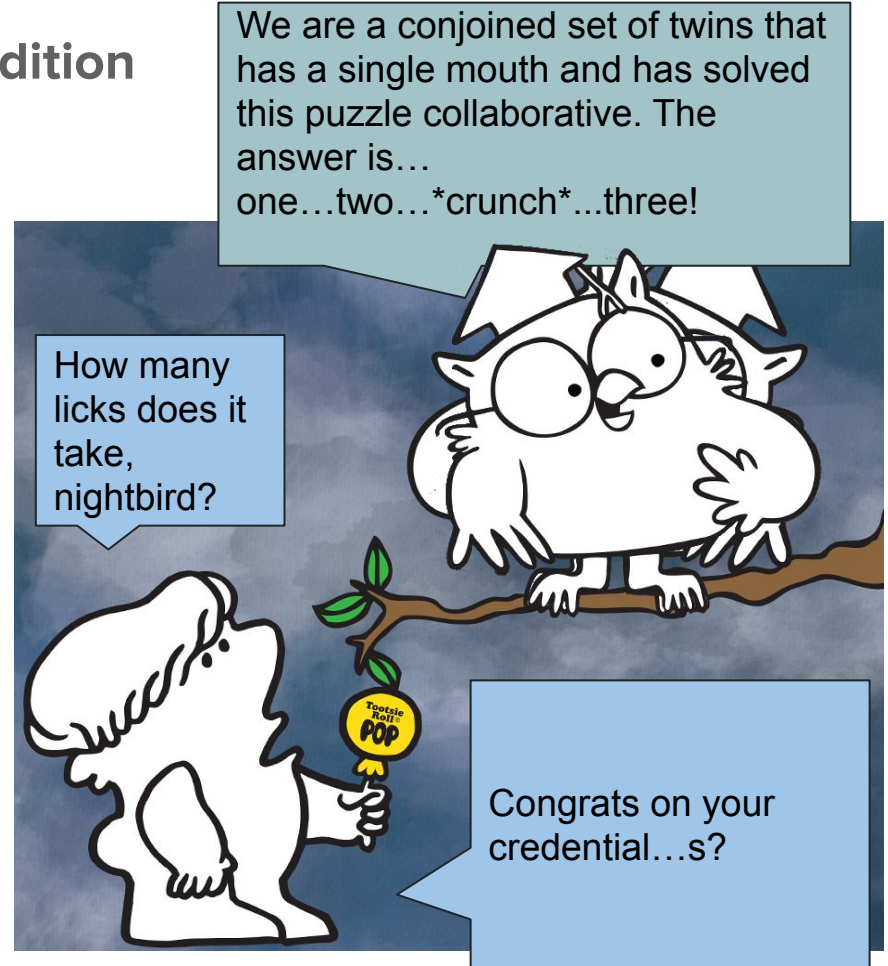
Credentials Tomorrow: Multiparty Edition

Normally...

- 1 issuer
- 1 subject

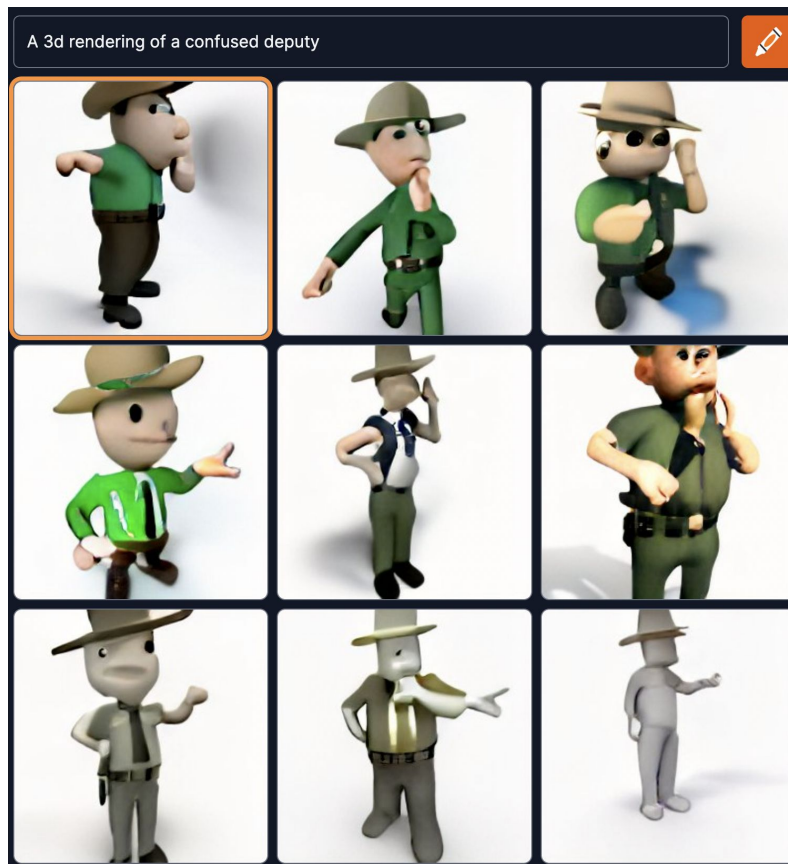
Exhibit A:

- Tootsie man = issuer
- Mr. Owl (left) = subject 1
- Mr. Owl (right) = subject 2



Prior Art: Delegated Credentials

- [zcap](#) with Data Integrity Proofs
- [ucan](#) with Json Web Tokens
- [dpop](#) with Json Web Tokens
- [transferable non fungible tokens](#) with ...
- [non transferable non fungible tokens](#) with ...



Delegated Credentials

```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://www.w3.org/2018/credentials/examples/v1"
  ],
  "id": "http://example.edu/credentials/3732",
  "type": ["VerifiableCredential", "ChildrensPassport"],
  "issuer": "https://example.com/issuer/123",
  "issuanceDate": "2010-01-01T00:00:00Z",
  "credentialSubject": {
    "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
    "name": "John Doe Jr.",
    "delegate": "Mr. John Doe Sr.",
    "delegateId": "did:example:abcdefghijklmn0p"
  }
}
```

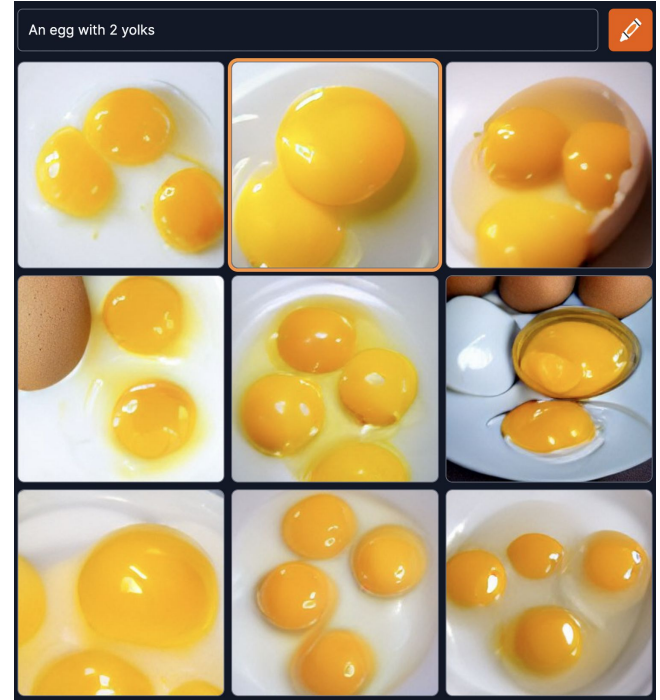
Multiparty Part 1: Multiple Subjects

Examples:

- Marriage certificates
- Vaccinated Traveler with Service Dog
(vaccination evidence for both dog and human?)
- (Delegated?) Parent / child relationships
- Relationships with legal or medical necessity *(e.g. spouse in a coma, parent with dementia)*

Questions:

- Isn't subject vs holder already confusion enough?
- Should credentials with multiple subjects even be allowed?
- What alternatives are there?



Multiparty Part 1: Multiple Subjects

It is possible to express information related to multiple [subjects](#) in a [verifiable credential](#). The example below specifies two [subjects](#) who are spouses. Note the use of array notation to associate multiple [subjects](#) with the `credentialSubject` property.

EXAMPLE 7: Specifying multiple subjects in a verifiable credential

```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://www.w3.org/2018/credentials/examples/v1"
  ],
  "id": "http://example.edu/credentials/3732",
  "type": ["VerifiableCredential", "RelationshipCredential"],
  "issuer": "https://example.com/issuer/123",
  "issuanceDate": "2010-01-01T00:00:00Z",
  "credentialSubject": [{
    "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
    "name": "Jayden Doe",
    "spouse": "did:example:c276e12ec21ebfeb1f712ebc6f1"
  }, {
    "id": "did:example:c276e12ec21ebfeb1f712ebc6f1",
    "name": "Morgan Doe",
    "spouse": "did:example:ebfeb1f712ebc6f1c276e12ec21"
  }]
}
```

Multiparty Part 2: Multiple Issuers

Examples:

- NFTs from a smart contract
(*contract address vs minter address?*)
- Co-tweets
- Parent's claims about their children
- Treaties, nuclear launch codes
- Co-signing for a loan

Questions:

- Should credentials with multiple issuers even be allowed?
- What alternatives are there?



Multiparty Part 2: Multiple Issuers

EXAMPLE 7: Specifying multiple subjects in a verifiable credential

```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://www.w3.org/2018/credentials/examples/v1"
  ],
  "id": "http://example.edu/credentials/3732",
  "type": ["VerifiableCredential", "TreatyCredential"],
  "issuer": [{
    "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
    "name": "United States of America"
  }, {
    "id": "did:example:c276e12ec21ebfeb1f712ebc6f1",
    "name": "Canada"
  }]
}
```

What's Next?

- **Welcome proposals on all three:**
 - [Delegated Credentials](#) (vc-data-model #930)
 - [Multi-Subject Credentials](#) (vc-data-model #931)
 - [Multi-Issuer Credentials](#) (vc-data-model #932)
- **Open Questions**
 - How are multiple issuers, multiple subjects represented in JWTs? Data Integrity Proofs?
 - How does this complicate selective disclosure / ZKP schemes?

Intro to SD-JWT (Kristina Yasuda, 40 mins)

Selective Disclosure for JWTs

- The purpose of this presentation
 - Introduce the concept
 - Make a preliminary decision on how SD-JWT VC and VP would look like
- An IETF draft adopted by OAuth WG
- Solves the problem that VC-JWTs were not selectively disclosable
- Use-Cases emerging

Why?

	Existing SD schemes	SD-JWT
Complexity	Complex (for most of mortals)	As simple as possible
Algorithms	Advanced cryptography	Standard cryptography with salted hashes
Format	Binary formats (ISO mDL)	JSON & JWT
Security	Hard to audit	Easy to understand & verify

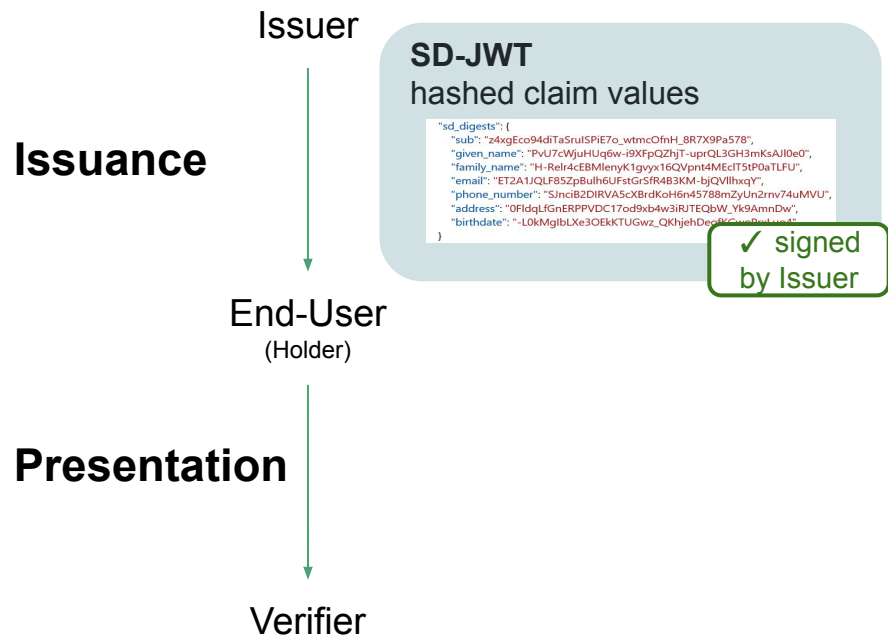
Salted Hash Approach

Idea:

- Issuer hashes each claim value together with a random salt
 - "John" → hash(salt, "John") → "PvU7cWjuHUq6w-i9XFpQZhjT-uprQL3GH3mKsAJ10e0"
- Issuer-signed credential only contains digests
 - "given_name": "PvU7cWjuHUq6w-i9XFpQZhjT-uprQL3GH3mKsAJ10e0"
- Holder selectively releases salt + plain-text value
 - "given_name" = hash("eluV50g3gSNII8EYnsxA_A", "John")
- Verifier checks by calculating hashes & issuer's signature

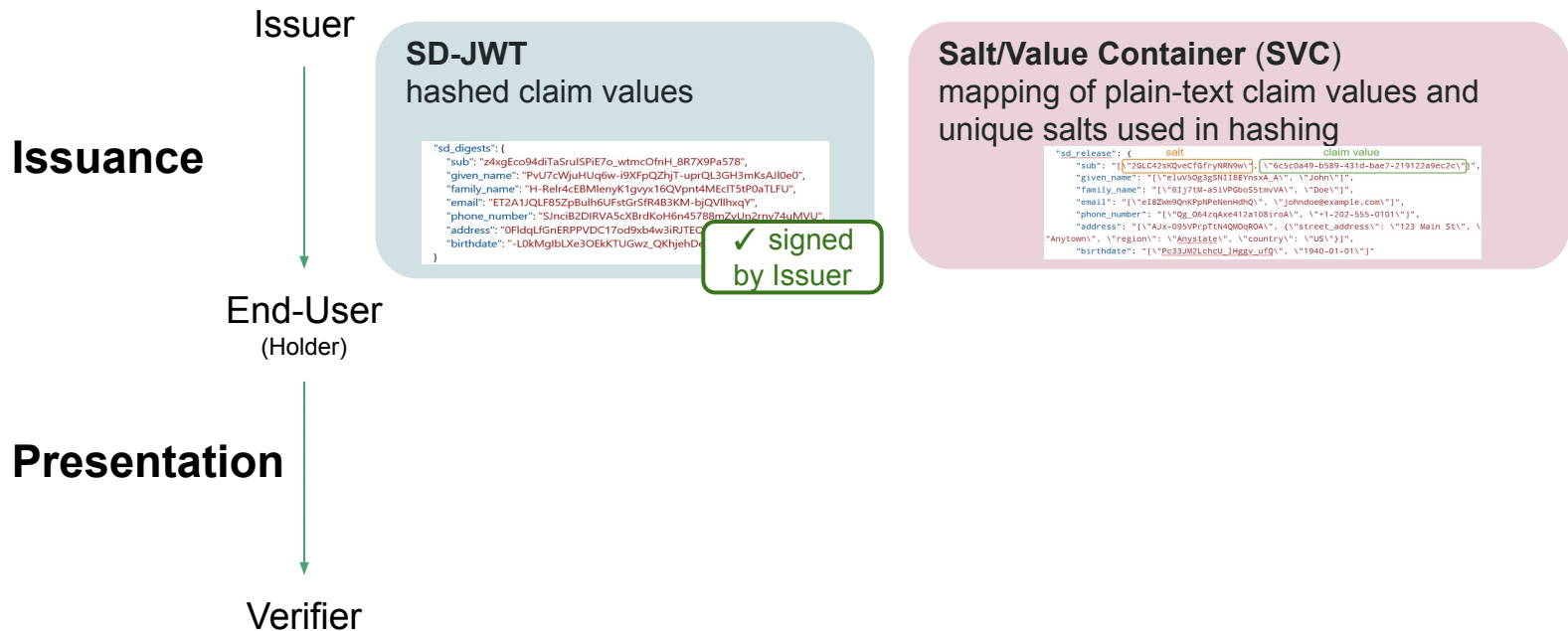
Overview: Salted Hash Approach

Issuance (1/2)



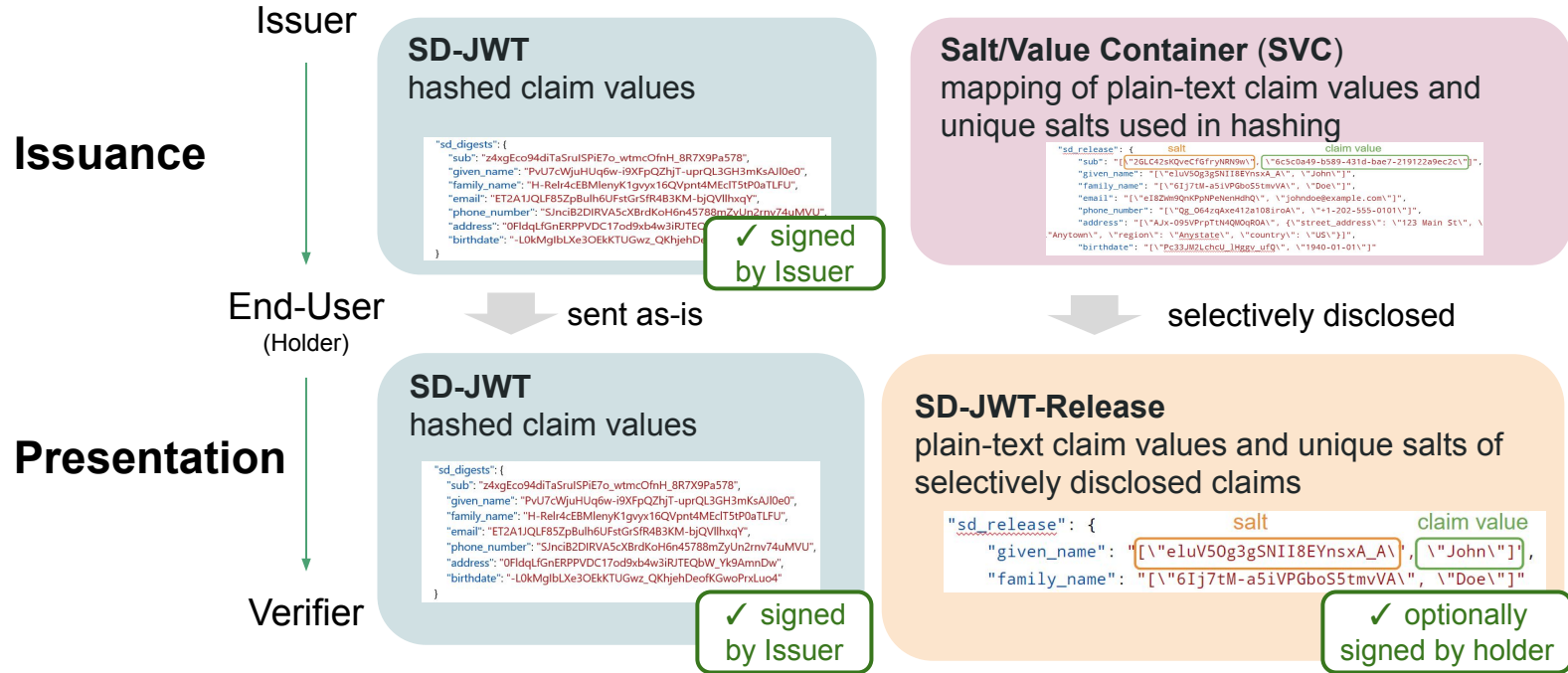
- Issuer hashes each claim value together with a random salt
-> "John" → hash(salt, "John") → "PvU7cWjuHUq6w-i9XFpQZht-uprQL3GH3mKsAJ10e0"
- Issuer-signed credential only contains digests
-> "given_name": "PvU7cWjuHUq6w-i9XFpQZht-uprQL3GH3mKsAJ10e0"

Issuance (2/2)



- Issuer sends a mapping of plain-text claim values and unique salts
-> given_name": "[{"e1uV50g3gSNII8EYnsxA_A"}]", [{"John"}]

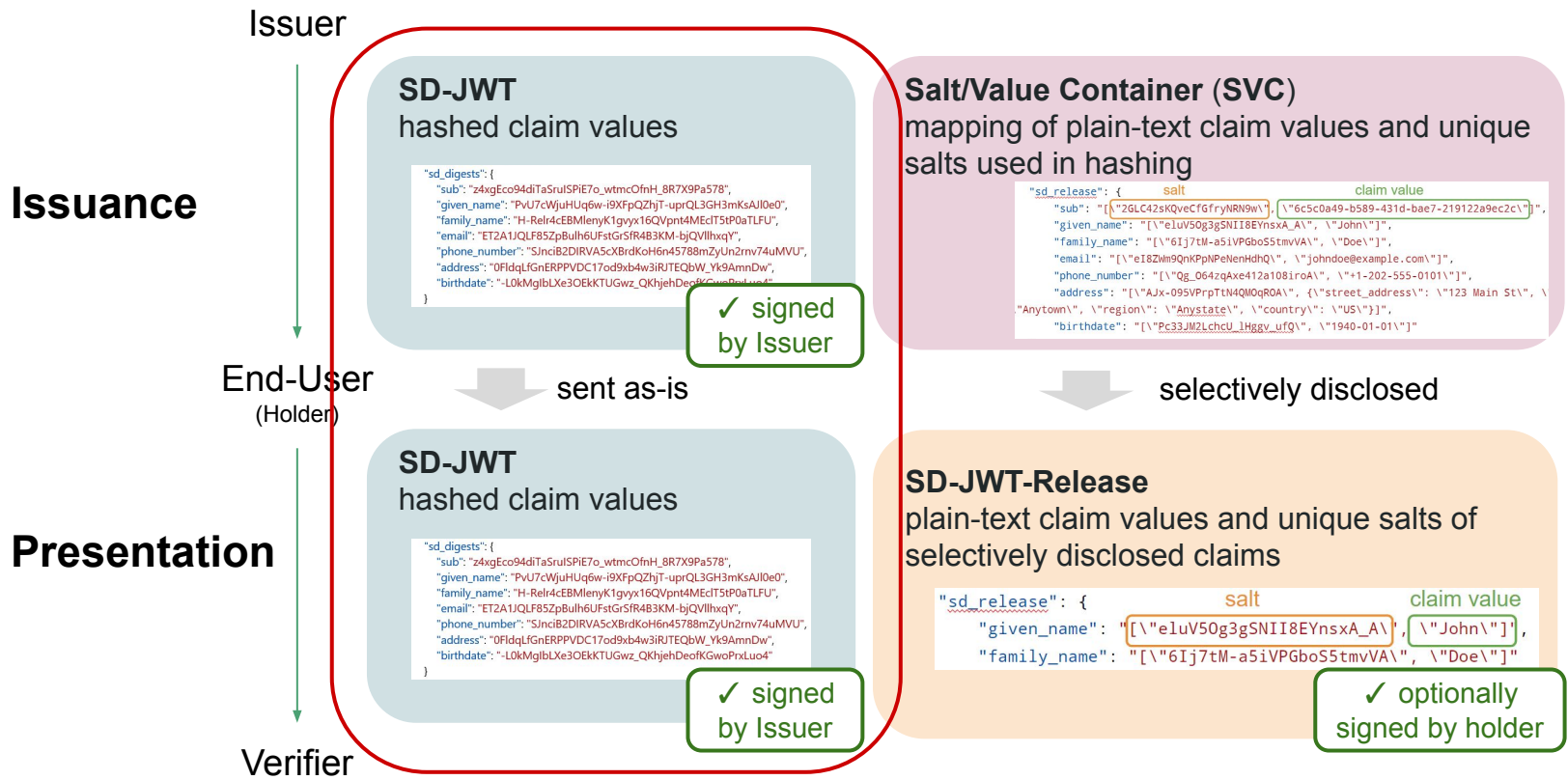
Presentation



- Holder selectively releases salt + plain-text value
-> "given_name" = hash("e1uV50g3gSNII8EYnsxA_A", "John")
- Verifier checks by calculating hashes & issuer's signature

Deep-dive of the Basics

Deep-dive on SD-JWT



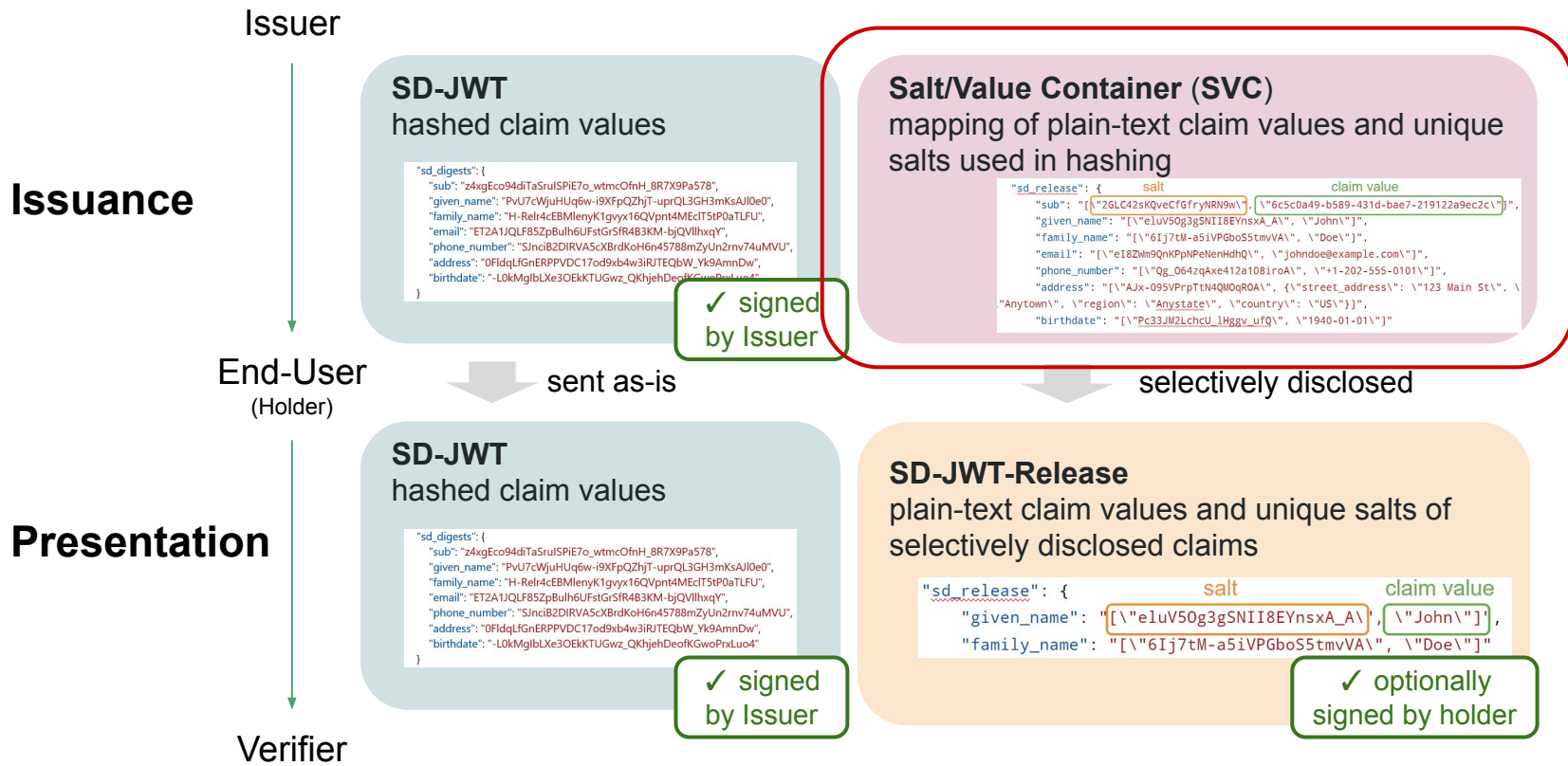
Example: SD-JWT

Issuer creates & sends to holder:

```
{
  "iss": "https://example.com/issuer",
  "sub_jwk": { # optional: public key of holder
    "kty": "RSA",
    "n": "pm4b0HBg-oYhAyP(...)7ihcw",
    "e": "AQAB"
  },
  "iat": 1516239022,
  "exp": 1516247022,
  "sd hash alg": "sha-256",
  "sd_digests": {
    "sub": "z4xgEco94diTaSruISPiE7o_wtmc0fnH_8R7X9Pa578",
    "given_name": "PvU7cWjuHUq6w-i9XFpQZhjT-uprQL3GH3mKsAJl0e0",
    "family_name": "H-Relr4cEBMlenyK1gvyx16QVpnt4MEclT5tP0aTLFU",
    "email": "ET2A1JQLF85ZpBulh6UFstGrSfR4B3KM-bjQVl1hxqY",
    "phone_number": "SJnciB2DIRVA5cXBrdKoH6n45788mZyUn2rnnv74uMVU",
    "address": "0F1dqLfgNERPPVDC17od9xb4w3iRJTEQbw_Yk9AmnDw",
    "birthdate": "-L0kMgIbLXe30EkKTUGwz_QKhjehDeofKGwoPrxLuo4"
  }
}
```

Digests of all the claims issued to the holder by the Issuer (currently defined to be a top level JWT claim)

Deep-dive on SVC



JSON strings?!

Example: Salt/Value Container

Issuer creates & sends to holder together with SD-JWT:

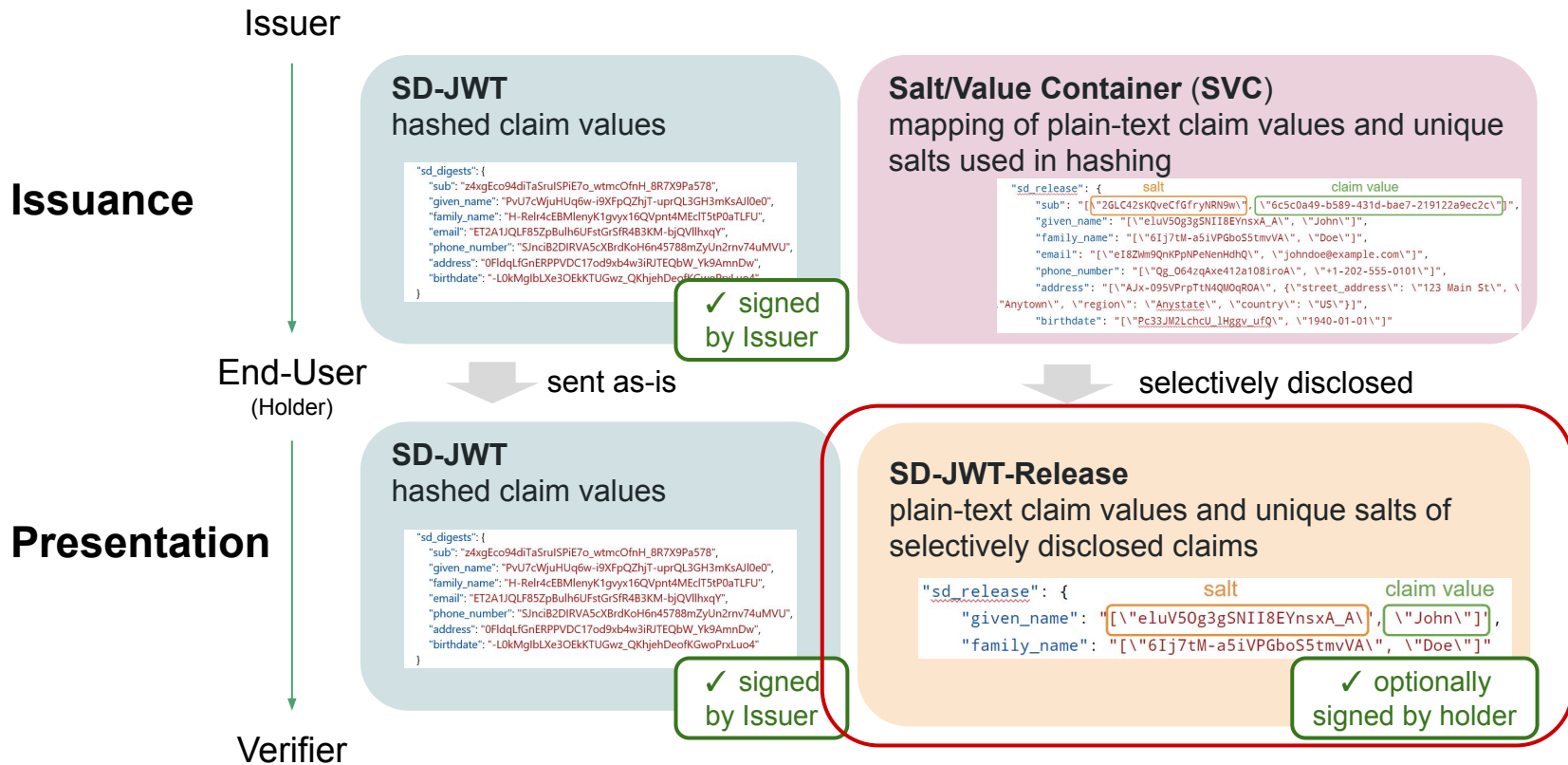
```
{
  "sd_release": {
    "sub": "[\"2GLC42sKQveCfGfryNRN9w\", \"6c5c0a49-b589-431d-bae7-219122a9ec2c\"]",
    "given_name": "[\"eLuV50g3gSNII8EYnsxA_A\", \"John\"]",
    "family_name": "[\"6Ij7tM-a5iVPGboS5tmvVA\", \"Doe\"]",
    "email": "[\"eI8ZWm9QnKPpNPeNenHdhQ\", \"johndoe@example.com\"]",
    "phone_number": "[\"Qg_064zqAxe412a108iroA\", \"+1-202-555-0101\"]",
    "address": "[\"AJx-095VPrpTtN4QM0qROA\", {\"street_address\": \"123 Main St\", \"locality\": \"Anytown\", \"region\": \"Anystate\", \"country\": \"US\"}]",
    "birthdate": "[\"Pc33JM2LchcU_lHggv_ufQ\", \"1940-01-01\"]"
  }
}
```

Mapping of salts and claim values of all the claims issued by the Issuer (Currently defined as an array passed as a string)

JSON string literals...

- ...ensure all parties hash the same values
- ...obsolete canonicalization even for complex claim values
- ...with extremely simple implementation

Deep-dive on SD-JWT-Release



Example: SD-JWT-Release

Holder creates from SVC & sends to verifier together with SD-JWT:

```
{  
  "nonce": "XZ0Uco1u_gEPknxS78sWWg",  
  "aud": "https://example.com/verifier",  
  "sd_release": {  
    "given_name": "[\"e1uV50g3gSNII8EYnsxA_A\", \"John\"],",  
    "family_name": "[\"6Ij7tM-a5iVPGboS5tmvVA\", \"Doe\"]"  
  }  
}
```

Salts and claim values of the claims that the Holder is releasing to the Verifier

Verifier checks:

- SD-JWT is valid (signature, etc.)
- SD-JWT-Release is valid (if signed: signature, nonce, etc.)
- digests in SD-JWT match hashes of released claims

Then: Extract plain-text values from released claims

Foot-gun: Verifier MUST calculate and check digests before using claims!

Deep-dive of the Advanced Concepts

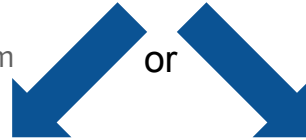
Granularity of hashing in SD-JWT

```
{
  "address": {
    "street_address": "123 Main St",
    "locality": "Anytown",
    "region": "Anystate",
    "country": "US"
  }
}
```

one digest per claim

or

digest per component of the claim



```
{
  "sd_digests": {
    "address": "0F1dqLfGnERPPVDC17od9xb4w...",
  }
}
```

```
{
  "sd_digests": {
    "address": {
      "street_address": "07_Isd6CmZqcSobPvpMgmJwB41hPUHHG8jg5LJ8YzfY",
      "locality": "w-zTF61jkQLTvVyp_JNyD3t5Waj-B2vb0AXH1q80sjI",
      "region": "nTvoKpGA6YQwEZipVBIM4WVH9KWEnwiqsRjEhrxhQz4",
      "country": "u-01yDQqDTTq0gUBSjwiIlgkMLZg_QOTELMfZrRT5e6k"
    }
  }
}
```

Works with Complex Objects!

Example of OpenID for Identity
Assurance

For full example, see the spec

```
{
  "sd_digests": {
    "verified_claims": {
      "verification": {
        "trust_framework": "w1mP4oPc_J9thBex0TaQi1vgxFmruQJxZYLfnkNFMAI",
        "time": "Pu3i0CWrpVLJW-LT30yF1bFBPP15B6-uKk3PnGDflv8",
        "verification_process": "8HqIXRmczsdYOZzGcLqI5-19xN5QbK2XDtXmdfH7z-4",
        "evidence": [{
          "type": "TnLuqGGQm6jfe0oa5uX1diKANUPuh-zHrpBFdX9MR-g",
          "method": "SagmakoSu-X-XUPIC3EgdrEEwIwXrWXX4-i68X9TyEo",
          "time": "ld2c5oYDRtQcfU6PzogPkx_95WYqhqIJNVRMnfcsicY",
          "document": {
            "type": "ufWjDaAa54MnHeji2ZUUHDdnpZ9zx6CUG6uR28VMtsQ",
            "issuer": {
              "name": "a4GMucU7Zb060r0Svd7huY6Qho1bIf3v1U5BvPR8q6Y",
              "country": "135k9M0m2SCnYRuOfHuYScYVS2q3eeY7IItyRsaBT8"
            },
            "number": "cUv0xLUp8RV7TTVliEiu-TQIel-LsE8E-XfUgfqk5gk",
            "date_of_issuance": "NIs8o1JnJ0v4J1qIEBKuTs2sEFs4fgGJhNqM6xdQt7E",
            "date_of_expiry": "HTR37vLtANT6Mwk-9dBqekFpCvaTG7zNf1ze56rnV64"
          }
        }]
      },
      "claims": {
        "given_name": "NB9XH_yJKqK0hXDmXkZKpMCKrB0m0Td8bqJFYDJYQnQ",
        "family_name": "hAUbJ66ZYL9VJLbjsDpmSs2e9Ff_Ohim_WR4bwZyvoQ",
        "birthdate": "6XOR4k56BgWk5tnNisbmbEHvoGX7RRfy6Z8HEN196cU",
        "place_of_birth": {
          "country": "CLTlhuy13Wwc3_ISon1kEypFwvCmfhLSpGUMCyAUG68",
          "locality": "AQoX8ixGpz-ipweEGLC-2umqwyQdhjIeiUB_TKwcE2E"
        },
        "nationalities": "nfoc__QK1MUHodmxwly-Kp-6ewgX3CdK7Ia0RJHIXVo",
        "address": "ngn04uQeOktM7YdFD8x82doS7WJnlZnq-rQE_RfuBSI"
      }
    },
    "birth_middle_name": "FeFSwd9drypEPtWvgIZ42N9j_yostt1Ds5PBpxT3Rng",
    "salutation": "57CMhvASQMNuzuQ0a_B1_VX5XdH73TcuPxyWGiorj5g",
    "msisdn": "leKbB0ro6q3jrVraCqt443uaGZVZisD3iGrKuKE2mqM"
  }
}
```

Running Code:

4 independent implementations!

- **Python:** <https://github.com/oauthstuff/draft-selective-disclosure-jwt>
- **Kotlin:** <https://github.com/IDunion/SD-JWT-Kotlin>
- **Rust:** https://github.com/kushaldas/sd_jwt
- **TypeScript:** <https://github.com/christianpaquin/sd-jwt>

```
### Produce SD-JWT and SVC
```

```
sdjwt = SDJWT(  
    user_claims,  
    issuer,  
    ISSUER_KEY,  
    HOLDER_KEY,  
    claims_structure,  
    blinded_claim_names,  
    iat,  
    exp,  
)
```

- Reference implementation in python: ~500 LoC.
- Evolves with the spec.
- Examples in the spec generated from the code.

Use-Case: W3C VC-Data-Model

JWT-VC (= SD-JWT) Option 3 in [Issue #908](#)

```
{
  "sub": "urn:ietf:params:oauth:jwk-thumbprint:sha-256:NzbLsXh8uDCCd-6MNwXF4W_7no
WXFZAfHkxZsRGC9Xs",
  "jti": "http://example.edu/credentials/3732",
  "iss": "https://example.com/keys/foo.jwk",
  "nbf": 1541493724,
  "iat": 1541493724,
  "exp": 1573029723,
  "cnf": {
    "jwk": { << Holder Binding Key>> }
  },
  "vc": {
    "@context": [
      "https://www.w3.org/2018/credentials/v1",
      "https://www.w3.org/2018/credentials/examples/v1"
    ],
    "type": [
      "VerifiableCredential",
      "UniversityDegreeCredential"
    ],
    "credentialSubject": {
      "first_name": "Jane",
      "last_name": "Doe"
    }
  },
  "sd_digests": {
    "vc": {
      "credentialSubject": {
        "email": "ET2A1JQLF85ZpBulh6UFstGrSfR4B3KM-bjQV1lhxyQ",
        "phone_number": "SJnciB2DIRVA5cXBrdKoH6n45788mZyUn2rnr74uMVU",
        "address": "0F1dqLfGnERPPVDC17od9xb4w3iRJTEQbW_Yk9AmnDw",
        "birthdate": "-L0kMgIbLXe30EkKTUGwz_QKhjehDeofKGwoPrxLuo4"
      }
    }
  }
}
```

JWT-VP (= SD-JWT-Release) - should be straightforward

```
{
  "iss": "did:example:ebfeb1f712ebc6f1c276e12ec21",
  "aud": "s6BhdRkqt3",
  "nbf": 1560415047,
  "iat": 1560415047,
  "exp": 1573029723,
  "nonce": "660!6345FSer",
  "vp": {
    "@context": [
      "https://www.w3.org/2018/credentials/v1"
    ],
    "type": [
      "VerifiablePresentation"
    ],
    "verifiableCredential": ["eyJhb...npyXw"]
  },
  "sd_release": {
    "given_name": "[\"6Ij7tM-a5iVPgboS5tmvVA\", \"John\"]",
    "family_name": "[\"eI8Zwm9QnKPPnPeNenHdhQ\", \"Doe\"]"
  }
}
```

A **verifiable credential** **MUST** have a `credentialSubject` **property**.

-> Can `credentialSubject` be empty?

Other options for SD-JWT-VC

JWT-VC (= SD-JWT) Option 2 in [Issue #908](#)

```
{
  "sub": "urn:ietf:params:oauth:jwk-thumbprint:sha-256:NzbLsXh8uDCcd-6MNwXF4W_7no
WXFZAfHkxZsRGC9Xs",
  "jti": "http://example.edu/credentials/3732",
  "iss": "https://example.com/keys/foo.jwk",
  "nbf": 1541493724,
  "iat": 1541493724,
  "exp": 1573029723,
  "cnf": {
    "jwk": { << Holder Binding Key>> }
  },
  "vc": {
    "@context": [
      "https://www.w3.org/2018/credentials/v1",
      "https://www.w3.org/2018/credentials/examples/v1"
    ],
    "type": [
      "VerifiableCredential",
      "UniversityDegreeCredential"
    ],
    "credentialSubject": {
      "first_name": "Jane",
      "last_name": "Doe",
      "sd_digests": {
        "email": "ET2A1JQLF85ZpBulh6UFstGrSfR4B3KM-bjQV1lhxqY",
        "phone_number": "SJnciB2DIRVA5cXBrdKoH6n45788mZyUn2rnnv74uMVU",
        "address": "0F1dqLfGnERPPVDC17od9xb4w3iRJTEQbW_Yk9AmnDw",
        "birthdate": "-L0kMgIbLXe30EkKTUGwz_QKhjehDeofKGwoPrxLuo4"
      }
    }
  }
}
```

JWT-VC (= SD-JWT) Option 1 in [Issue #908](#)

```
{
  "sub": "urn:ietf:params:oauth:jwk-thumbprint:sha-256:NzbLsXh8uDCcd-6MNwXF4W_7no
WXFZAfHkxZsRGC9Xs",
  "jti": "http://example.edu/credentials/3732",
  "iss": "https://example.com/keys/foo.jwk",
  "nbf": 1541493724,
  "iat": 1541493724,
  "exp": 1573029723,
  "cnf": {
    "jwk": { << Holder Binding Key>> }
  },
  "vc": {
    "@context": [
      "https://www.w3.org/2018/credentials/v1",
      "https://www.w3.org/2018/credentials/examples/v1"
    ],
    "type": [
      "VerifiableCredential",
      "UniversityDegreeCredential"
    ],
    "credentialSubject": {
      "first_name": "Jane",
      "last_name": "Doe",
      "sd_digests": {
        "email": "ET2A1JQLF85ZpBulh6UFstGrSfR4B3KM-bjQV1lhxqY",
        "phone_number": "SJnciB2DIRVA5cXBrdKoH6n45788mZyUn2rnnv74uMVU",
        "address": "0F1dqLfGnERPPVDC17od9xb4w3iRJTEQbW_Yk9AmnDw",
        "birthdate": "-L0kMgIbLXe30EkKTUGwz_QKhjehDeofKGwoPrxLuo4"
      }
    }
  }
}
```

Questions to the group

- A **verifiable credential** *MUST* have a `credentialSubject` **property**.
 - -> Can `credentialSubject` be empty?
- In which section should SD-JWT be described?
- Implications for @context..?

VC Test Suites

(Manu Sporny, 40 mins)

VC TEST SUITE 2.0

Radical

Verifiable Credential Testing Options



LETS GOOOOOO!*

... review all the mistakes we have made over the years.

* Historical accuracy note: This phrase did not mean the same thing in the 1980s.

1.0

TEST SUITE

Lessons learned

1.1

TEST SUITE

Lessons learned

2.0

REQUIREMENTS

Improvement goals

2.0

OPTIONS

Ecosystem survey



1.0

TAINTED LOVE

That time Christine used lisp for testing...



PRE-1.0 TEST SUITE

- Command-line driven Python script ported to Racket (lisp)
- Implementers had to install Racket (lisp) and run the test suite
- It was as bad as it sounds
- No one knew how to maintain the test suite; because: lisp

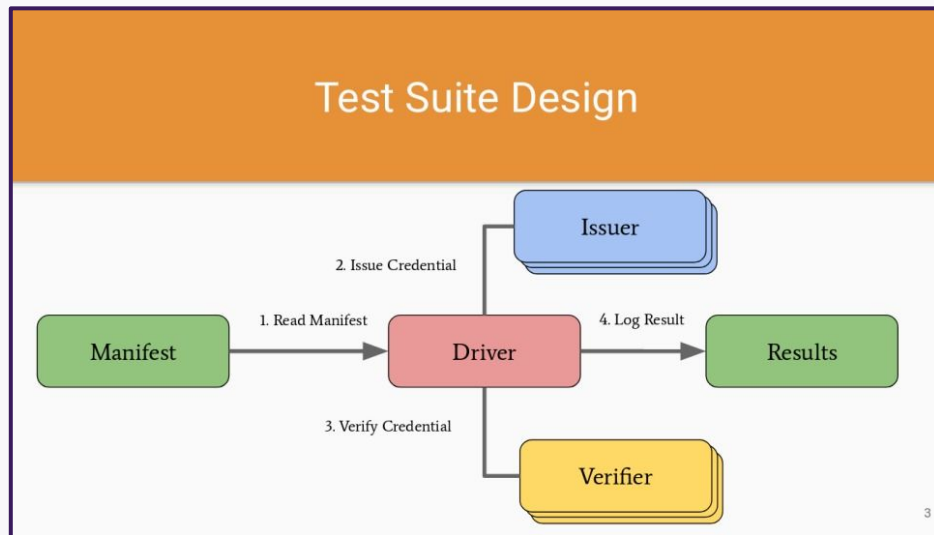
1.0 REC TEST SUITE

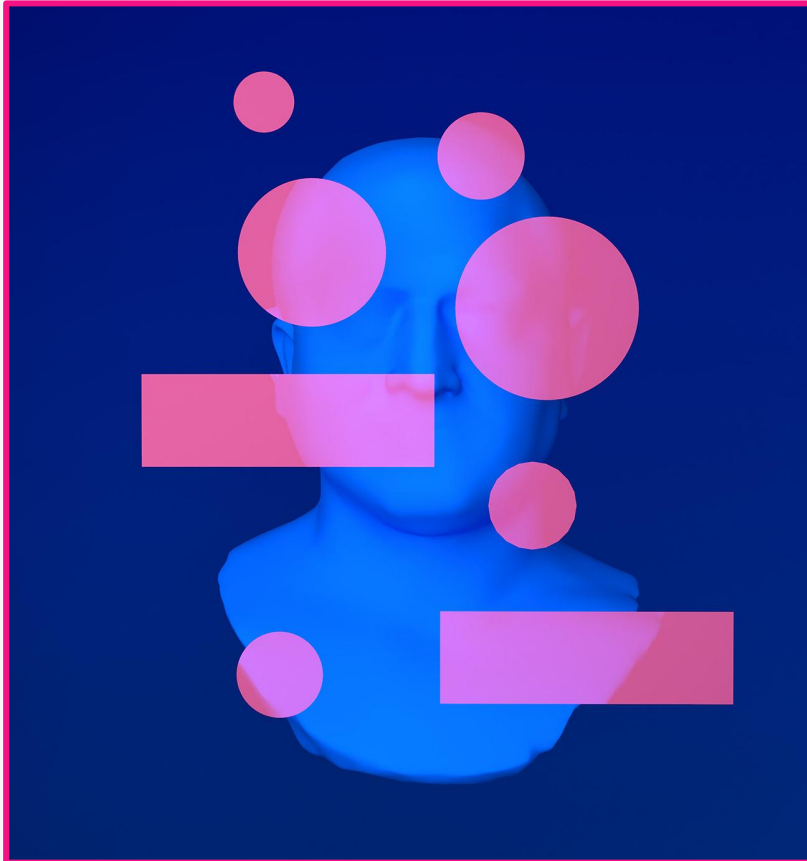
For Candidate Recommendation, we rewrote the test suite in Javascript:

- All tests listed in an input manifest.
- Driver read each test, issued a VC, then verified it.
- Results logged to output file.
- Report generated from output file.

The results:

- 11 implementations; not bad
- We survived REC and 1.0





1.0 TEST SUITE LESSONS LEARNED

- Don't pick a test suite environment that reduces the number of people that can contribute.
- We explored Docker, but then decided to not use it for the same reason as above - not enough expertise. Command-line interface seemed good enough.
- Once implementers got a "good" result, most of them never came back.
- The implementation report is out of date for most implementations by years.
- This is not sustainable for the ecosystem.



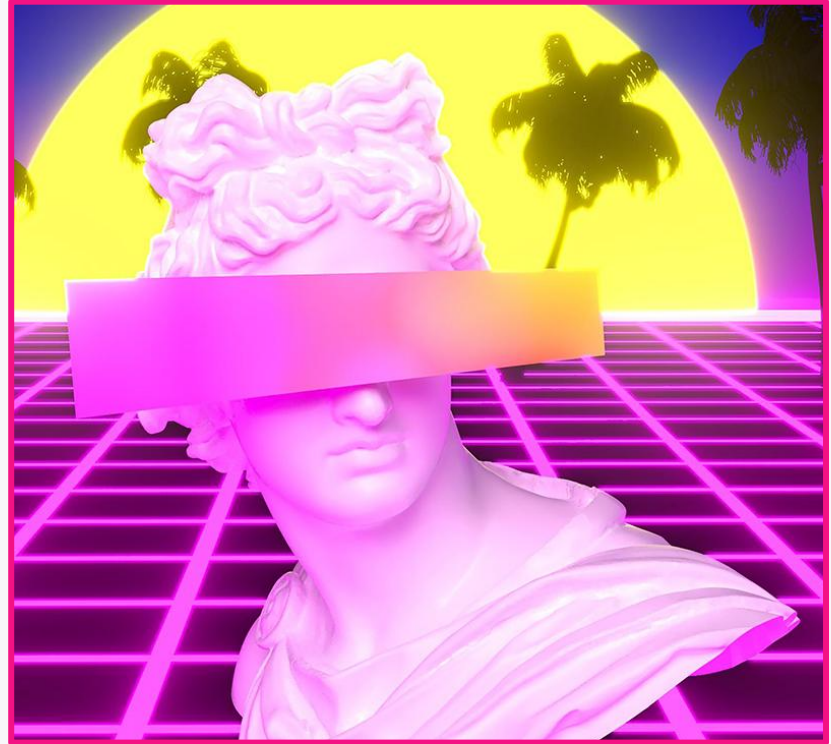
1.1

DON'T STOP BELIEVIN'

Hold on to that feelin'...

1.1 TEST SUITE LESSONS LEARNED

- We were not allowed to change large normative items, so the test suite didn't change much.
- We picked up two more implementations, but many of the implementation reports remain out of date.
- We survived REC and 1.1





2.0

BUST A MOVE

If you want it, you got it

2.0 REQUIREMENTS

- Test all "MUST" normative statements in specifications.
- Run +/- tests for each normative statement.
- Test data model
- Test Data Integrity, JWT-VC, SD-JWT specifications.
- Loose coupling between implementations and test suite.
- Run tests against implementations on weekly/nightly basis.

2.0 OPTIONS

- Keep test suite as-is (static), run against command line "issuer" and "verifier" implementations?
- Dockerize command line implementations?
- Use an HTTP interface for implementations?
- Some other type of interface?
- Examples: [vc-test-suite](#), [JWS-Test-Suite](#), [Traceability](#), [Verifier](#)



DISCUSS



mspotify@digitalbazaar.com
<https://www.linkedin.com/in/manusporny>

CREDITS: This presentation template was created by **Slidesgo**, and includes icons by **Flaticon**, and infographics & images by **Freepik**

Please keep this slide for attribution

Break
(15 mins)

Internationalization / Multilingual (Shigeya Suzuki, 30 mins)

Topics

{{ I want to talk/discuss about requirements, not how, now }}

- Why we need multilingual support?
- EXAMPLE1 in VCDM 1.1
- Construct of multiple language strings
- Default language and Language priority as a credential
- Externalization of mapping

Multilingual Support Needed

- In credentials used across borders, bilingual notation may be useful or even required.
 - Education related Certificates
 - Certificates required when traveling
- Improper use of local language embedding cause a problem.
I.e., United Airlines' app does not accept my SHC based vaccination record, because my name encoded in HL7/FHIR record is in Japanese
 - Part of the reason is HL7/FHIR's lack of multilingual support
 - There is similar but different encoding problem for the certificates for expats in Japan
 - Expat's name is encoded in Katana and placed into patient's first/last name, their English name is placed in a special section (no standards exists for this...)

Firstly... Issues in VC 1.1

“Example 1” shown
multilingual example, but
there are no discussions on
the notation either in
normative or non-normative
way.

```
// claims about the subjects of the credential
"credentialSubject": {
  // identifier for the only subject of the credential
  "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
  // assertion about the only subject of the credential
  "alumniOf": {
    "id": "did:example:c276e12ec21ebfeb1f712ebc6f1",
    "name": [{
      "value": "Example University",
      "lang": "en"
    }, {
      "value": "Exemple d'Université",
      "lang": "fr"
    }]
  }
}
```

Representation in the Data Model

- The least thing we need is have a normative rules for multilingual strings, similar to the VCDM 1.1 EXAMPLE 1
 - But we may also need “direction” -> use of BCP47 as JSON-LD does?
 - And the EXAMPLE1 style structure might not the best way to do
 - Do we want to allow only one style of construct ?
- In addition to above we may want to have
 - Externalization of such mappings from credential object to reduce size of credential objects, while supporting a lot of languages.
 - Translation of property name may also useful, and possibly externalized

Mapping String Construct Styles and Performance / Programmability

```
"credentialSubject": {
  "id": "did:example:xxx",
  "alumniOf": {
    "id": "did:example:xxx",
    "name": [{
      "value": "Example University",
      "lang": "en"
    }, {
      "value": "Exemple d'Université",
      "lang": "fr"
    }]
  }
},
```

```
"credentialSubject": {
  "id": "did:example:xxx",
  "alumniOf": {
    "id": "did:example:xxx",
    "name": [
      "en": "Example University",
      "fr": "Exemple d'Université"
    ]
  }
},
```

```
"credentialSubject": {
  "id": "did:example:xxx",
  "alumniOf": {
    "id": "did:example:xxx",
    "multilingual": {
      "en": {
        "name": "Example University"
      },
      "fr": {
        "name": "Exemple d'Université"
      },
    }
  }
},
```

Language tags, Default language and List of languages as a property

- Use of BCP47 for language tags
- List of language keys as a property may be useful depends on the construct used
- Specifying a language or order of languages is can be used as a hint

Externalization

```
"Key_language": "en",  
"External_language_string":  
"https://example_univ.edu/cert/basse.json#  
XXYYZZ",  
"credentialSubject": {  
  "id": "did:example:xxx",  
  "alumniOf": {  
    "id": "did:example:xxx",  
    "name": "Example University",  
  },  
}
```



```
{  
  "key_language": "en",  
  "Languages": [ "fr" ],  
  "Example University": {  
    "fr": "Exemple d'Université"  
  }  
}
```


Questions

- “JSON-LD compatible JSON”?
- JSON-LD language map
 - Requires ‘@language’ etc....

Next steps

- Consult with Internationalization/multilingual experts
- Drafting PR + discuss on:
 - Mapping
 - Externalization

JSON Schema and the VC Data Model

(Gabe Cohen, 30 mins)

Why JSON Schemas in VCs?

As an issuer, I want...

- A consistent vocabulary for my credentials
- Strict validations for the presence of required fields, and field-level validations (e.g. is that a valid email?)
- To have a consistently identifiable document that can be easily replicated, promoting widespread use and adoption
- Fast static validation; highly cacheable!

As a holder, I want...

- To know that my credentials will be accepted! (they are of the right shape, have the necessary data elements)

As a verifier, I want...

- To define a limited set of authoritative data-shapes that I accept
- Clear authorship for the schema that backs a credential I receive (e.g. is this the DMV's drivers license scheme, or Frank's from down the block?)
- Fast static validation; highly cacheable!

Background

§ 5.4 Data Schemas

Data schemas are useful when enforcing a specific structure on a given collection of data. There are at least two types of data schemas that this specification considers:

- Data verification schemas, which are used to [verify](#) that the structure and contents of a [credential](#) or [verifiable credential](#) conform to a published schema.
- Data encoding schemas, which are used to map the contents of a [verifiable credential](#) to an alternative representation format, such as a binary format used in a zero-knowledge proof.

It is important to understand that data schemas serve a different purpose from the `@context` property, which neither enforces data structure or data syntax, nor enables the definition of arbitrary encodings to alternate representation formats.

This specification defines the following [property](#) for the expression of a data schema, which can be included by an [issuer](#) in the [verifiable credentials](#) that it issues:

credentialSchema

The value of the `credentialSchema` property *MUST* be one or more data schemas that provide [verifiers](#) with enough information to determine if the provided data conforms to the provided schema. Each `credentialSchema` *MUST* specify its `type` (for example, `JsonSchemaValidator2018`), and an `id` property that *MUST* be a [URI](#) identifying the schema file. The precise contents of each data schema is determined by the specific type definition.

NOTE

The `credentialSchema` property provides an opportunity to annotate type definitions or lock them to specific versions of the vocabulary. Authors of [verifiable credentials](#) can include a static version of their vocabulary using `credentialSchema` that is locked to some content integrity protection mechanism. The `credentialSchema` property also makes it possible to perform syntactic checking on the [credential](#) and to use [verification](#) mechanisms such as JSON Schema [JSON-SCHEMA-2018] validation.

EXAMPLE 18: Usage of the `credentialSchema` property to perform JSON schema validation

```
Credential | Verifiable Credential (with proof) | Verifiable Credential (as JWT)

{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://www.w3.org/2018/credentials/examples/v1"
  ],
  "id": "http://example.edu/credentials/3732",
  "type": ["VerifiableCredential", "UniversityDegreeCredential"],
  "issuer": "https://example.edu/issuers/14",
  "issuanceDate": "2010-01-01T19:23:24Z",
  "credentialSubject": {
    "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
    "degree": {
      "type": "BachelorDegree",
      "name": "Bachelor of Science and Arts"
    }
  },
  "credentialSchema": {
    "id": "https://example.org/examples/degree.json",
    "type": "JsonSchemaValidator2018"
  }
}
```

Who is using this, and how?

Background

- In 2019, Workday came to the CCG and presented a scheme for using JSON Schemas with Verifiable Credentials
 - With the help of Orié and other community members, we got a Draft Community Group Report out: <https://w3c-ccg.github.io/vc-json-schemas/v1/index.html>
 - It defines...
 - Metadata to include in schemas (identifiers, versioning, authorship information)
 - A method to version schemas based on semantic versioning / schemaver
 - Some recommendations for identification using DID-relative URIs
(e.g. `did:work:MDP8AsFhHzhwUvGNuYkX7T;id=06e126d1-fa44-4882-a243-1e326fbe21db;version=1.1`)
- Gained moderate usage via a number of community members!
- In late 2021, I wanted to improve the spec, and see how it could gain more utility – <https://w3c-ccg.github.io/vc-json-schemas/v2/index.html>

Background

- Where are we?
 - A draft that needs some work, but has most of the right parts there.
 - Proven implementations in > 2 libraries (that I know of..)

Metadata

Top-level information on a [Credential Schema](#). Pieces of data wrapping the [JSON Schema](#) to context about the schema.

EXAMPLE 2

```
{
  "type": "https://w3c-ccg.github.io/vc-json-schemas/schema/2.0/schema.json",
  "version": "1.0",
  "id": "did:example:MDP8AsFhHzhwUvGNuYkX7T;id=06e126d1-fa44-4882-a243-1e326f21db",
  "name": "Email",
  "author": "did:example:MDP8AsFhHzhwUvGNuYkX7T",
  "authored": "2021-01-01T00:00:00+00:00"
}
```

Credential Schema

The data template for a credential. Refers to the entirety of a [Credential Schema](#), including both the [Metadata](#) and [JSON Schema](#). The term may refer to a document either with, or without a [proof](#).

EXAMPLE 1

```
{
  "type": "https://w3c-ccg.github.io/vc-json-schemas/schema/2.0/schema.json",
  "version": "1.0",
  "id": "did:example:MDP8AsFhHzhwUvGNuYkX7T;id=06e126d1-fa44-4882-a243-1e326f21db;version=1.0",
  "name": "Email",
  "author": "did:example:MDP8AsFhHzhwUvGNuYkX7T",
  "authored": "2021-01-01T00:00:00+00:00",
  "schema": {
    "$id": "email-schema-1.0",
    "$schema": "https://json-schema.org/draft/2019-09/schema",
    "description": "Email",
    "type": "object",
    "properties": {
      "emailAddress": {
        "type": "string",
        "format": "email"
      }
    },
    "required": ["emailAddress"],
    "additionalProperties": false
  }
}
```

Uptake & Usage

Precedent

- Multiple SSI companies (Workday, Block, IBM, Transmute, [Disco.xyz](#), [Kilt](#), [Traceability Vocab](#), Spruce, and more!) already using the CCG draft / JSON Schemas to back creds
- Many DIF specs make use of JSON Schemas with great success (Presentation Exchange, Credential Manifest, more...)

Widespread Support

- [JSON-Schema](#) has a number of IETF drafts
- Large number of [x-language libraries](#) that are well-maintained, web tools, code generation, and more

Plays nicely with JSON-LD

- *Best of both worlds*: LD for broader semantic interoperability, JSON-Schema for strict validation
- How I think of it: *JSON-LD defines the “world” – JSON Schema creates thoughtful subsets*
- No need to use both. Some will want to use one, the other, or both.

What's Next?

- Questions, considerations concerns?
- How does this get into VCDMV2?



justify why w3c verifiable credentials should be backed by a json schema

There are a few reasons why JSON Schema is a good choice for backing up W3C verifiable credentials:

1. JSON Schema is a standard format for specifying the structure of JSON data. This means that there is already a well-defined way to specify the structure of a verifiable credential, which makes it easier for developers to work with.
2. JSON Schema is a widely used format, and there are already many tools and libraries available for working with it. This means that it will be easier to find tools and libraries that support working with verifiable credentials backed by JSON Schema.
3. JSON Schema is a flexible format, and it is possible to add additional fields to a schema to support new features that may be added to verifiable credentials in the future. This means that JSON Schema can be used to support future evolution of the verifiable credentials standard.

Editorial Work Mode (Brent Zundel, 10 mins)

Editorial Work Mode

- Issues
 - Triage and label issues
 - Make concrete suggestions to help move toward consensus
- PRs
 - Substantive changes should reflect *WG* consensus and have time for review before they are merged
 - Editorial changes should have the approval of at least one more editor
 - But may be reverted if concerns are raised

End of Day 2
