



Hyperledger AnonCreds: Using ZKP Verifiable Credentials Everywhere

May 31, 2023 @ 8 AM Pacific / 17:00 CEST

WORKSHOPS

Stephen
Curran



Rodolfo
Miranda



Patrick
St-Louis





HYPERLEDGER ANONCREDS

Goals

- Understand verifiable credentials in general
- Understand AnonCreds unique capabilities
- Experience the steps in using AnonCreds
- Understand (a bit of) the cryptography in AnonCreds
- See how AnonCreds are used today in a variety of contexts
- Newly capabilities available with AnonCreds
- The future of AnonCreds

Takeaways

- How to use AnonCreds in your applications
- Build frameworks that embed AnonCreds
- Enable rooting AnonCreds in other ledgers/VDRs
- Contribute to the AnonCreds implementations
- Contribute to future AnonCreds specifications and implementations



HYPERLEDGER ANONCREDS

Part 1

- Online Identity With Verifiable Credentials
- Introduction to Hyperledger AnonCreds
- AnonCreds Data and Processes (Hands On)
 - Setup
 - Issuing
 - Presenting
 - Revocation

Part 2

- Zero Knowledge Proofs: The High School Math Edition (time permitting)
- Ledger-Agnostic AnonCreds
- AnonCreds in the W3C VC Data Model Format
- AnonCreds Exchange using CHAPI
- Making Credentials Beautiful — OCA
- Futures: AnonCreds v2.0

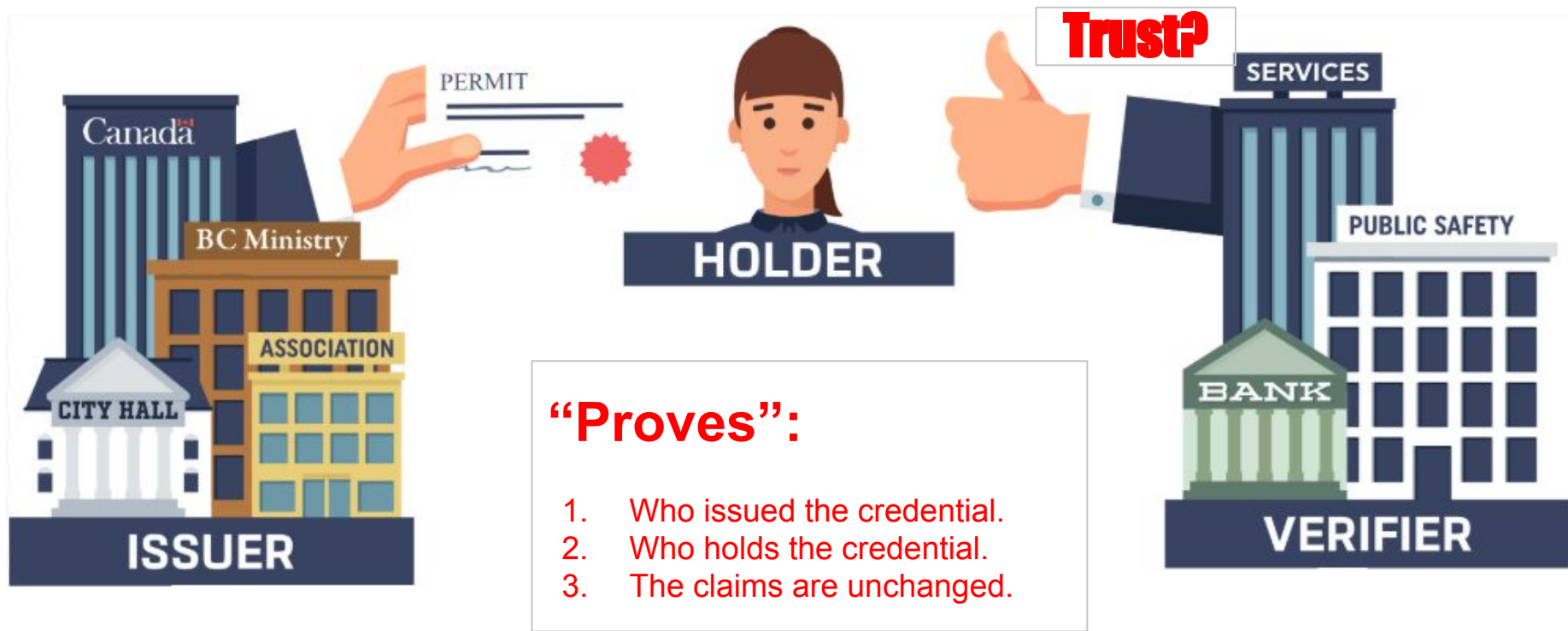
Online Identity with Verifiable Credentials



Licensed under the [Unsplash+ License](#)



The Paper Credential Model



Presenting Paper Credentials

Verifiers make trust decisions:

- Technology
 - Is it a legitimate representation?
 - Does it look forged?
- Governance
 - Is the source of the representation trustworthy?
 - Where does their authority come from?
 - Do they have trusted processes?



Credentials in the Digital World? Scan 'em!!



Selfie with ID

Max limit of 1 file.



Drag file to upload or click to browse files

I want to upload an additional selfie image (optional)

Provide a self-portrait photo ('selfie') while holding government-approved photo identification. Make sure that your selfie isn't blurry and text is legible. [More photo requirements](#)

To prevent fraudulent verification requests from people impersonating others, we need to verify the identity of the person on whose behalf the request is made (you or the relevant person, organisation or entity). Please obscure parts of the document (e.g. national ID number or other national identifier) as long as the remaining information identifies the relevant individual. Google will use this information solely to help us assess and document the authenticity of your request, and will delete the document within 30 days of verifying your application.



Examples of acceptable identification are:

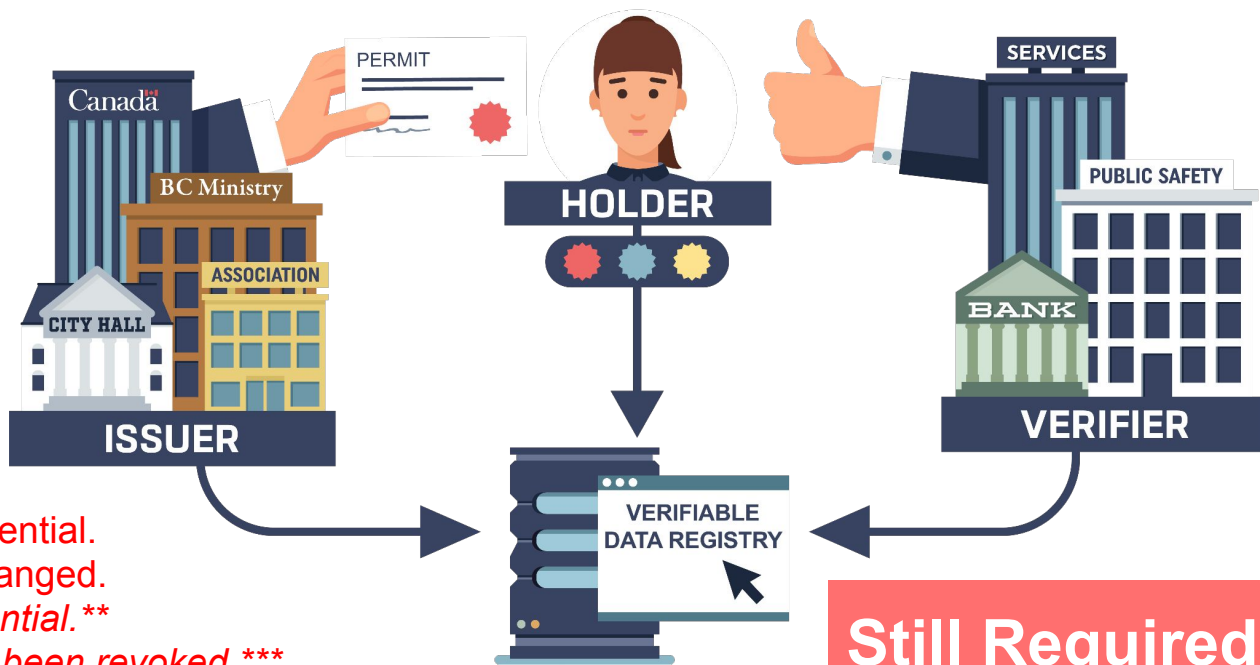
1. A valid government-issued photo ID, such as a driving licence or national ID card
2. In the event that you do not have a government-issued photo ID, please use your birth certificate and additionally a form of identification that has your photo and your name. Please note that we may require

What is the Verifiable Credentials model?

Credentials, Presentations and Protocols

Proves:

1. Who issued the credential.
2. The claims are unchanged.
3. Who holds the credential.**
4. The claims have not been revoked.***

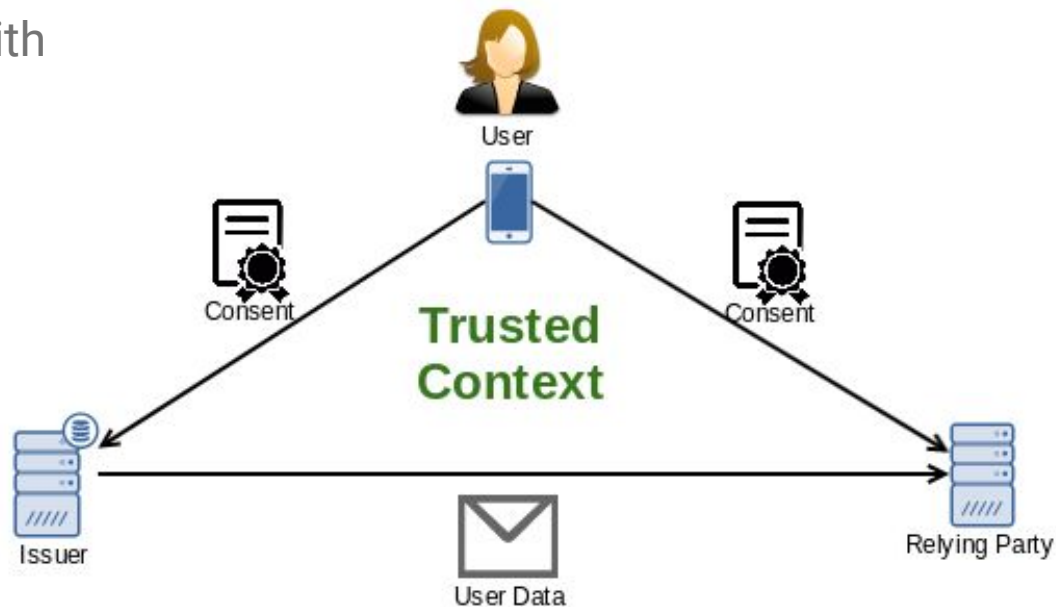


Still Required:

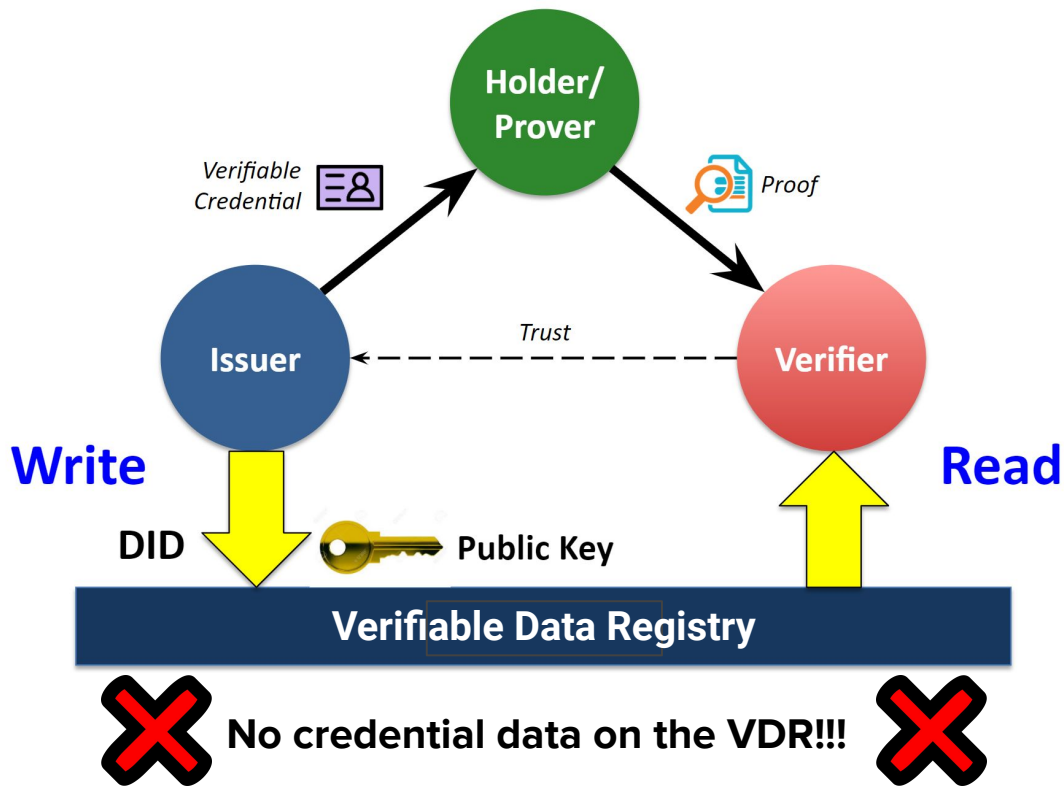
Do you trust the Issuer?

VCs Are Different from OpenIDConnect

- OpenIDConnect is the “Login with Facebook” model
- Same parties:
 - User == Holder
 - Relying Party == Verifier
- The flow is different – one transaction
- The Issuer is involved in every interaction
 - We don't want that!!



What is a Verifiable Data Registry?



VC Ecosystems – 2023

| Meeting Place | W3C Credentials Community Group (CCG) | Hyperledger Foundation | OpenID Foundation | ISO mDL Working Group |
|------------------------------------|--|---|--|--|
| Verifiable credential “type” | Verifiable Credentials using JSON-LD using Data Integrity Proofs | Hyperledger AnonCreds with ZKPs and VC JSON-LD | JWT, SD-JWT and potentially others | mDL and mdocs |
| Exchange Protocols | CHAPI and VC-API | DIDComm Messaging and Aries Data Exchange Protocols | OpenID4VCs OpenID4VCI OpenID4VCP and SIOPv2 | ISO standards being developed for presentation and issuance |


Hyperledger AnonCreds

Privacy: AnonCreds Zero Knowledge Proofs

- Selective disclosure
- Predicate proofs
- Derived presentations with *unlinkable* identifiers
- Multi-credential presentations



Hyperledger AnonCreds?

- Project at the Hyperledger Foundation
- [AnonCreds Specification v1.0](#)
 - Working Group to evolve the specification
- Complete [open source implementation](#) in Rust of the AnonCreds specification.
 - Derived from Hyperledger Indy and IDemix from IBM
 - Heavily used for the past 7+ years in the Hyperledger Self-Sovereign Identity (SSI) stack
- Verifiable Data Registry-agnostic 
 - AnonCreds objects can be published anywhere
- Working Group defining AnonCreds v2.0

AnonCreds Capabilities



IMPORTANT

- All the cryptographic features of any “verifiable credentials” format:
 - Per W3C: Tamper-evident issuer attestations (usually) about a subject
- Application of privacy-preserving ZKP technology
 - Selective disclosure
 - Predicates (“I’m older than 21”)
 - Defined holder-binding mechanism, across multiple credentials
 - No correlatable identifiers shared by holder — including when using revocation
 - Derived presentation vs. sharing VC itself
 - Blinded holder/subject identifier
 - **Helpful for Governments:** No new identifier for people

No other popular verifiable credential technology offers all of these capabilities.

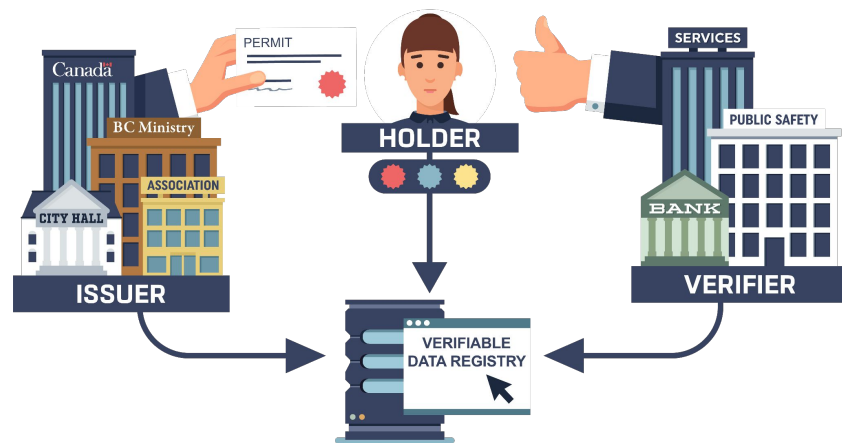


Familiar Cryptographic Processing

- Overall basis is [CL-Signatures](#), an RSA-based cryptographic suite
- [Canonicalization](#) — signatures are applied to 32-byte integers
 - Rules: Integers remain integers, all other elements are stringified and hashed into integers
- Predicates
 - Uses [Bullet Proofs](#), applicable only to integers, and only 4 expressions supported
- Selective disclosure, no correlatable identifiers
 - [Blinded signatures](#) make every presentation unique, with ZKPs to verify signatures
- Linked Secret
 - [Pederson Commitment](#) to the holder link secret made in the Offer/Request/Issue interaction
 - Aggregate proof that same linked secret used in all source verifiable credentials in presentation
- Revocation
 - [Accumulator-based](#) proof of non-revocation ZKP — credential identifier known to issuer, holder

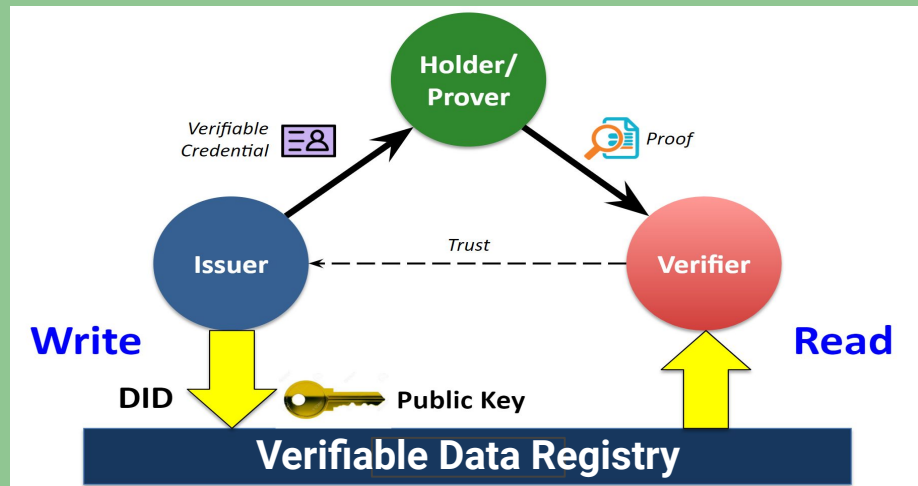
AnonCreds — What Happens?

- Setup
 - Issuer creates AnonCreds Objects — Schema, CredDef, [RevReg]
 - Issuer publishes AnonCreds Objects to ledger **Verifiable Data Registry (VDR)**
 - Holder creates link secret
- Issuance
 - Issuer - Holder interaction (offer, request)
 - Issuer creates/signs credential
- Presentation
 - Verifier requests presentation
 - Holder constructs presentation (w/revocation)
 - Verifier verifies presentation (reading from VDR)
- Revocation
 - Issuer publishes RevReg State to ledger VDR

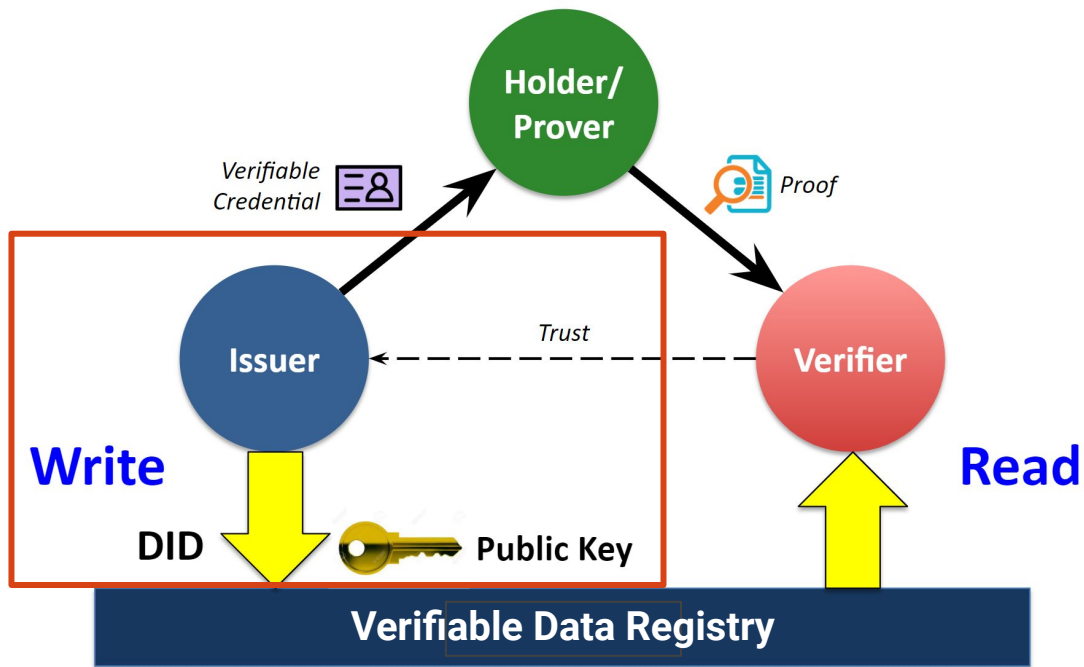


AnonCreds: Data and Processes

Setup (with a digression)



Publishing Objects



Setup — Publishing Schema

- Schema Object
 - List of attributes — [example](#)
 - Technically, not needed for the cryptography to work
 - More for the governance — share common schemas, minimize different schemas
 - Make the live of verifiers easier!
 - (Almost) No prescribed metadata, such as in the W3C VC Data Model (e.g., **issuanceDate**)
- A simple list
 - No complex JSON
 - But could be supported by flattening the data structure...
 - No dynamic arrays
 - **Cannot** be supported:
 - Requirement: A cryptographic element per data value — defined at setup time
 - Needed for use case?
 - Only option — make the array an item, with a **data:url** for the data in each credential

Setup - Publishing Credential Definition

- Credential Definition
 - List of attributes (again!) — this is why the schema is not really needed for the crypto — [example](#)
 - Extra attribute — “**master_secret**” — called “**link_secret**” (almost) everywhere.
 - Used for **holder binding** via a “blinded identifier”
 - Issuer keys
 - Including for revocation, if active — we’ll cover revocation later
- With a published credential definition — millions of credentials can be issued
 - NOTHING GOES ON THE LEDGER AT ISSUE TIME!

Setup — Holder Creates Link Secret

- Identifies the Holder and is (indirectly) put into all credentials they receive
- Implemented as a UUID (unique identifier)
- Handled as a blinded identifier
 - Blinded identifier given to the Issuer to put in a credential
 - Blinded identifier proven to be based on the link secret the Holder knows in a presentation
- Unlinkable, both for Issuers and Holders

Digression: The Verifiable Credential
Holder Binding Issue

The VC “Holder Binding” Issue

- When verifiers receive presentations, they **often** want to know the relationship between the holder (the prover) and the credential
 - Subject vs. Holder
 - “That the credential was issued to the holder”
- In the W3C VC Data Model — holder binding is currently out of scope
 - Issuer makes tamper-evident attestations about the subject — that’s it.
 - Anyone can present any VC — it’s up to the verifier to do holder binding (if it is needed...)

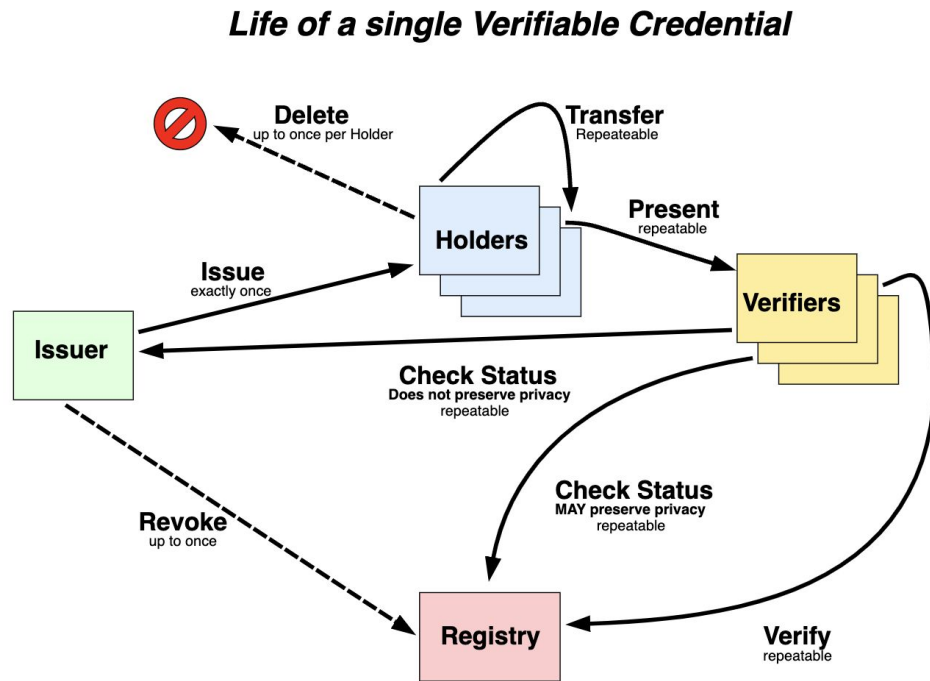


Figure 10 from the W3C Verifiable Credentials Data Model Standard, v1.1

Holder Binding — other techniques

- **Often**, a Holder's DID (public key) is used to bind a VC to a holder.
 - On issuance
 - Holder provides DID and proof (signature) of control over the DID
 - Issuer verifies signature, inserts DID into credential.
 - On presentation
 - Holder adds a proof that they control the DID in the credential.
 - The Holder DID is a correlatable identifier...
- Could be done in other ways:
 - Well-known mechanism by the issuer
 - Holder must present another piece of ID (perhaps paper) to accompany the VC
 - VC includes a picture of the subject, and that is presented and compared

AnonCreds Holder Binding: The `link_secret`

- AnonCreds *formally* defines holder binding
 - All AnonCreds are issued to the holder's `link_secret` so only the holder can present the VC
 - The holder **SHOULD** use the same `link_secret` for all credentials they receive
 - **MUST** for all credentials to be presented together
- The AnonCreds `link_secret` serves the same purpose as the “Holder DID”
 - But blinded, and so is presented as a *non-correlatable identifier*.
 - Holder to issuer during issuance
 - Request blinded identifier (with nonce) from Issuer
 - Blinded identifier plus proof from Holder to Issuer
 - Issuer inserts blinded identifier into the Credential
- All holders **MUST** present a ZKP to demonstrate holder binding
 - Presentation includes proof the same `link_secret` was used for all source VCs.

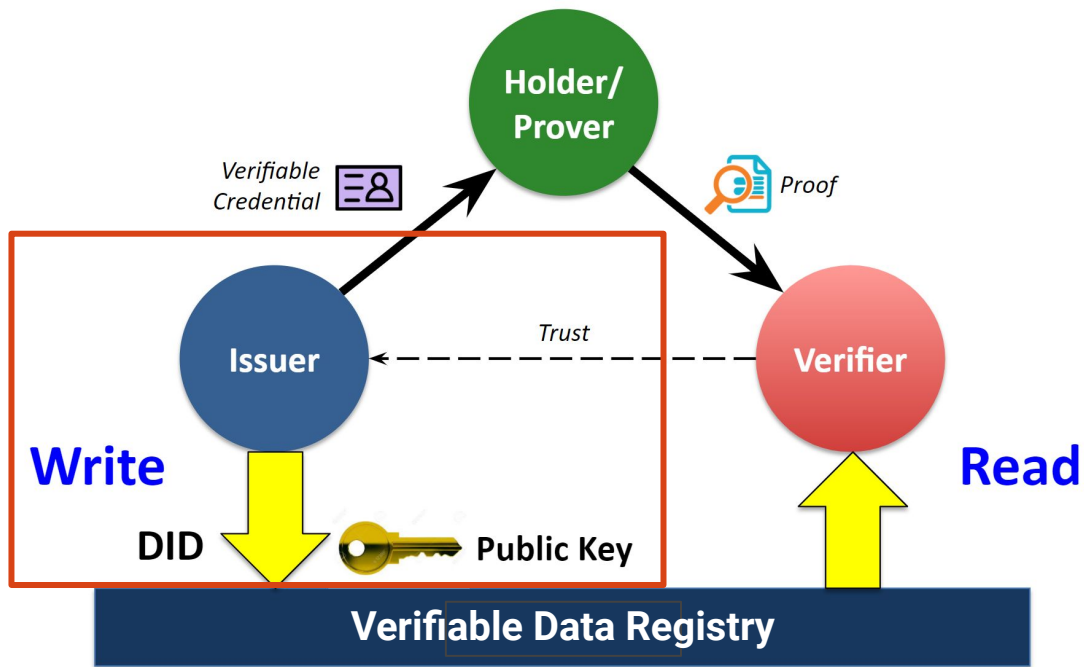
Lab 1: Preparing and Publishing a Credential

- Preparation Steps link: <https://bit.ly/AnonCredsPrep>
- Lab 1 link: <https://bit.ly/AnonCredsLab1>
- Steps:
 - Define a Schema (or two)
 - Publish the Schema
 - Review the published Schema
 - Publish Credential Definition for each schema
 - Review the published Credential Definition

Notes: Schema and Credential Definitions

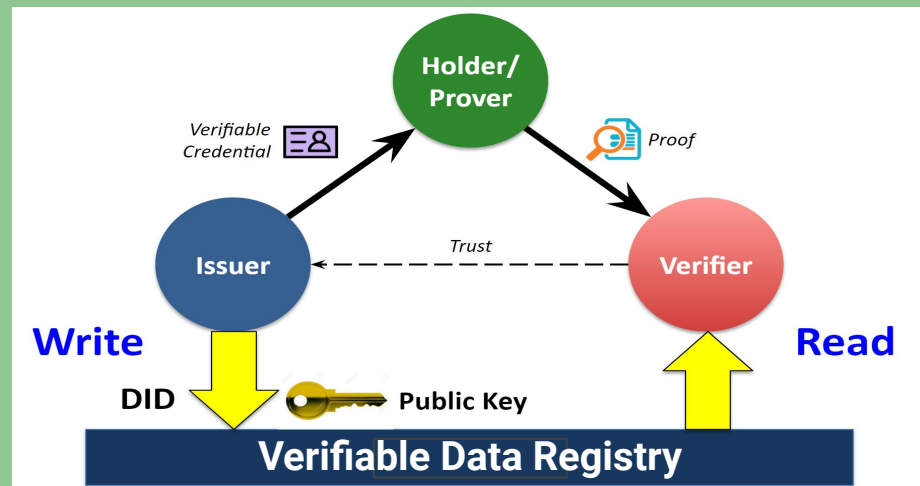
- Generally, CredDefs must be on the same Ledger/VDR as the Schema
 - Required for Hyperledger Indy
 - Why? Verification of the object by the ledger before writing
- Schema must have an associated “Publisher ID” (often a DID)
- Credential Definition must have an associated “Issuer ID” (often a DID)

Publishing Objects

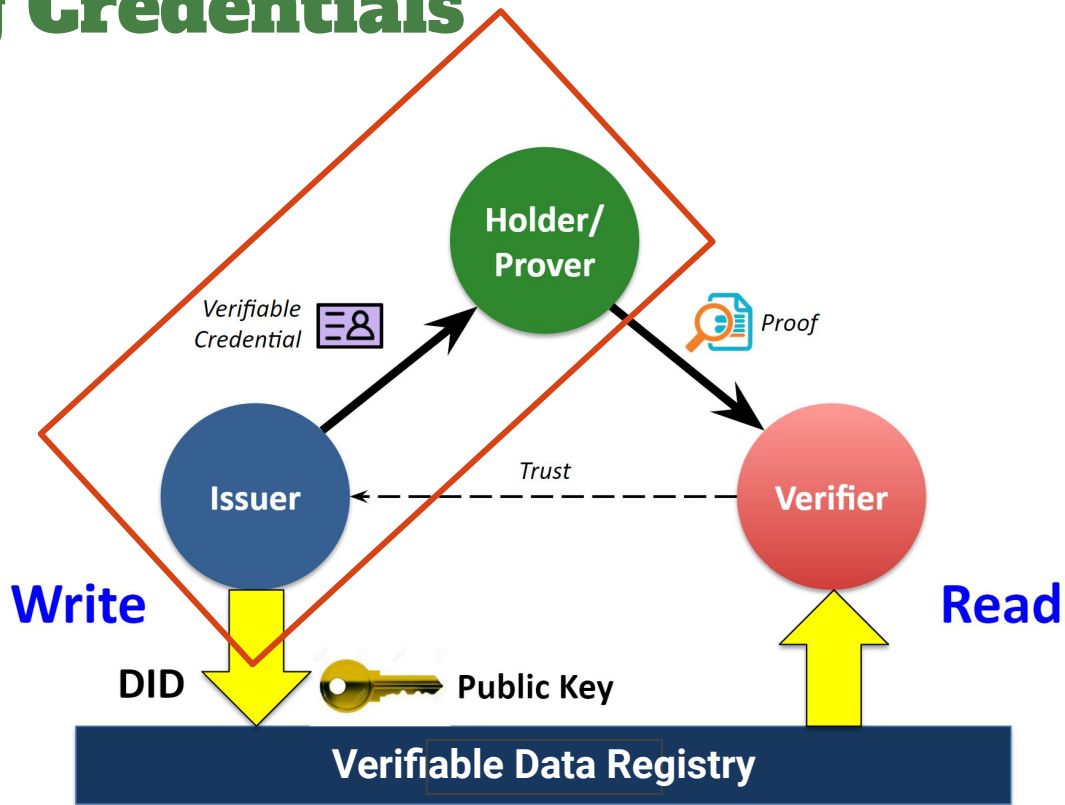


AnonCreds: Data and Processes

Issuing



Issuing Credentials



Issuing

- Three step process:
 - **Offer** — from Issuer to Holder
 - What type of credential is to be issued.
 - Key correctness proof.
 - A nonce for Issuer non-correlatibility.
 - **Request** — from Holder to Issuer
 - Blinded link secret.
 - Key correctness proof about link secret.
 - Entropy for credential.
 - **Issue** — from Issuer to Holder
 - The verifiable credential.

Aries Specific Processing

- [RFC 0453 Issue Credential Protocol](#)
 - General purpose issue credential protocol.
 - Not specific to AnonCreds, but useful.
 - A DIDComm Messaging Protocol, so how data moves is defined.
- Additions:
 - Proposal — from Holder to Issuer
 - Allows holder to initiate process.
 - Allows for negotiation.
 - Includes data for the credential.
 - Offer
 - Includes data for the credential.

AnonCreds Credential Data Model

```
{  
  "schema_id": "3av...s8W:2:fabername:0.1.0",  
  "cred_def_id": "3av...s8W:3:CL:13:default",  
  "rev_reg_id": null,  
  "values": {  
    "given_name": {  
      "raw": "Alice Jones",  
      "encoded": "728...2918"  
    }  
  },  
  "signature": {  
    "p_credential": {  
      "m_2": "5783...397",  
      "a": "203...785",  
      "e": "259...767",  
      "v": "626...819"  
    },  
    "r_credential": null  
  },  
  "signature_correctness_proof": {  
    "se": "163...839",  
    "c": "546...523"  
  },  
  "rev_reg": null,  
  "witness": null  
}
```

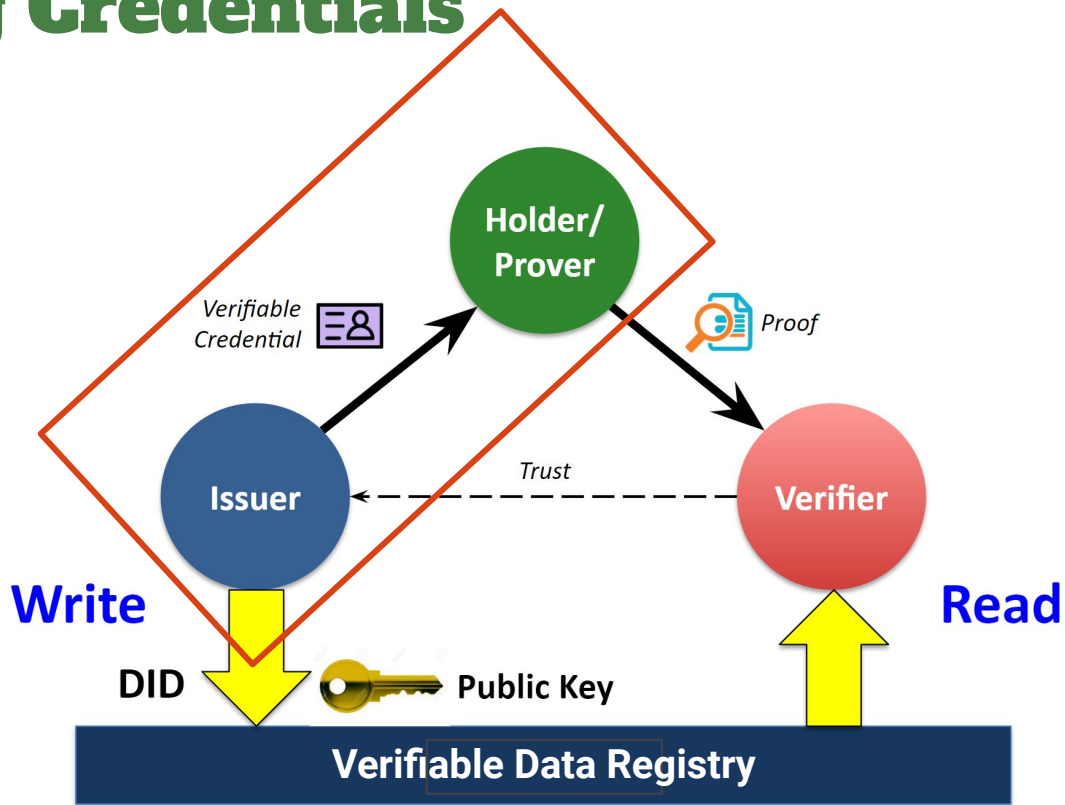
Raw and Encoded Values

- “raw” values are the attributes
- “encoded” values canonicalized attribute values as integers.
 - Only the encoded values are what is actually signed in an AnonCreds presentation.
 - Canonicalization rules:
 - Unsigned integers or integer strings are left as is.
 - Everything else is stringified and SHA256 hashed into an integer.
 - This is why only integers can be used in predicates — hashes are not sortable.
- Both “raw” and “encoded” values are included in credentials and presentations.
 - Issue: What if the issuer uses a different canonicalization scheme?
 - Issue: Verifier must make check that the encoded values are properly encoded (not forged).
- In future, encoding will be built into AnonCreds vs. handled by Issuer/Holder.

Lab 2: Issuing Credentials

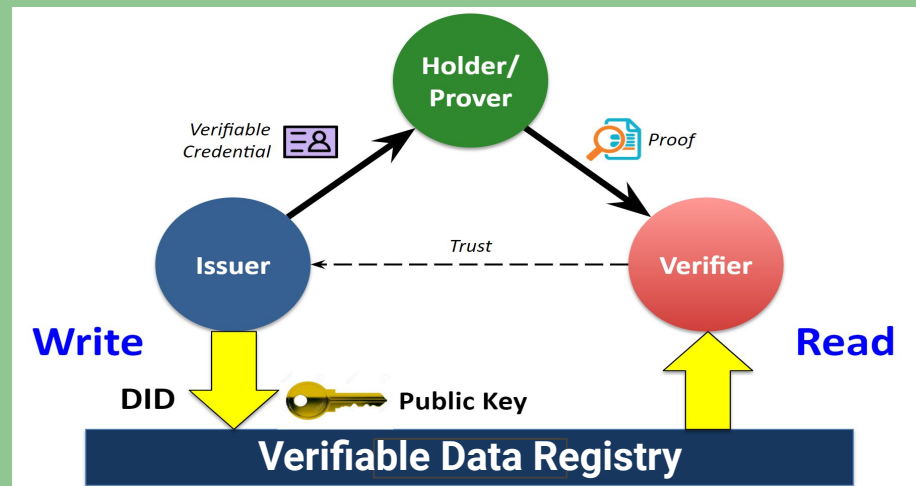
- Lab 2 link: <https://bit.ly/AnonCredsLab2>
- Steps:
 - Pick a Connection
 - Pick a Credential Definition
 - Fill in the Data (Click “*Enter Credential Value*” in Traction for UI)
 - Issue the Credential
 - In Wallet: Accept the Credential

Issuing Credentials

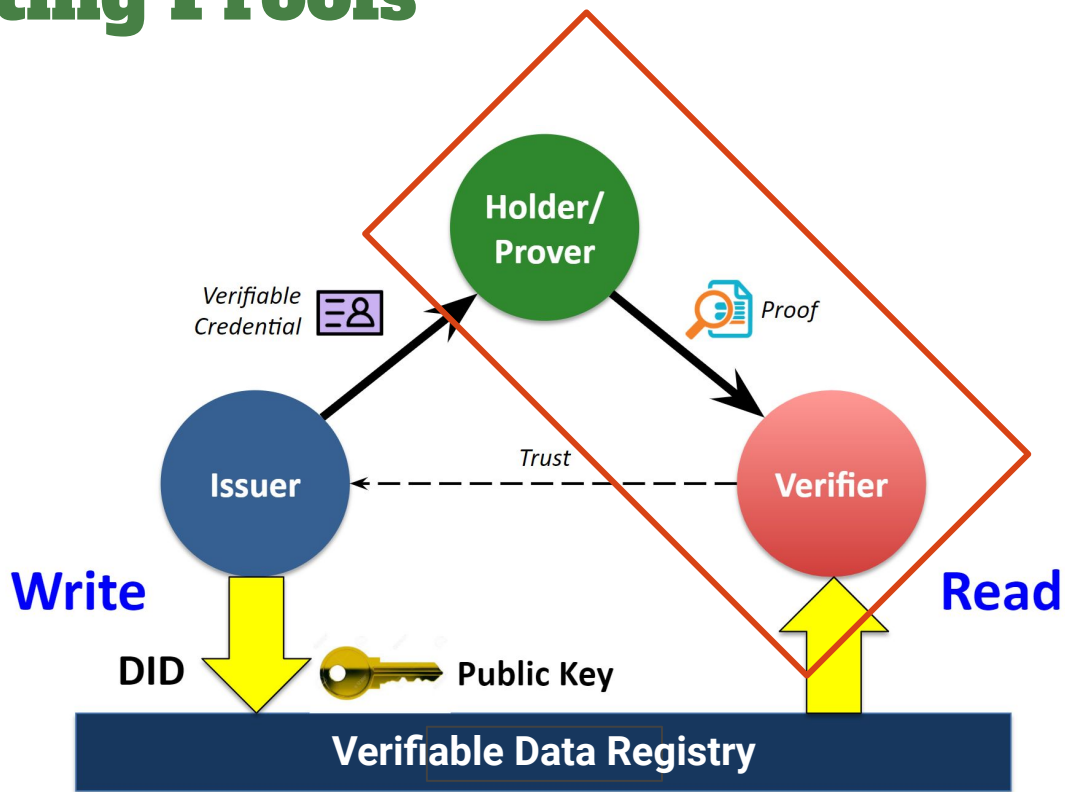


AnonCreds: Data and Processes

Presentation

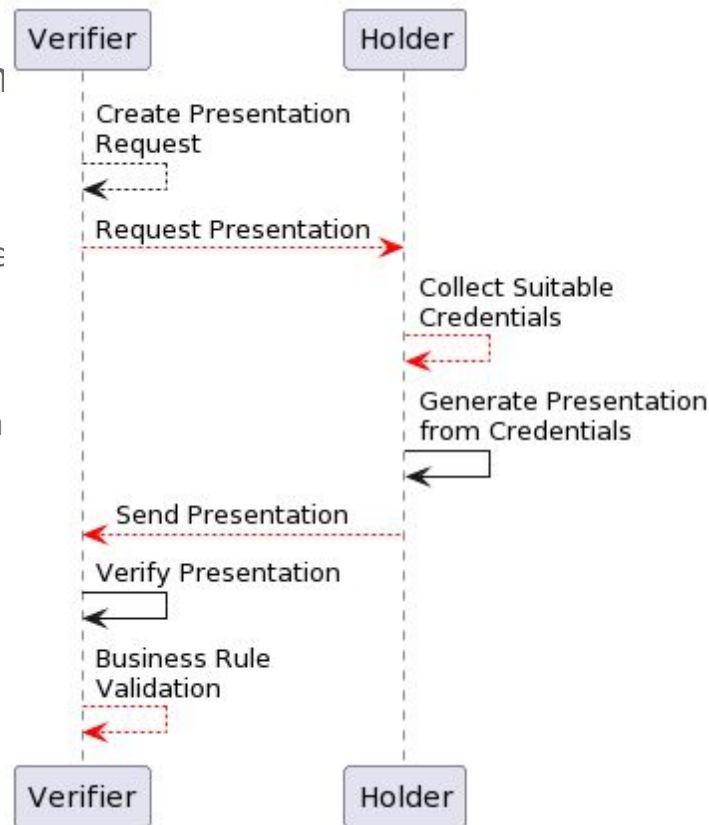


Presenting Proofs



Presentation Request / Presentation Flow

- Verifier to Holder — “This is what I need from
- Request/response model
 - Verifier asks for what they need
 - Holder constructs a presentation based on what cre
 - Cryptography is verified
- Note that verifiable credentials are not given
 - A presentation is derived from the source credentia



Request Presentation Components

- Requested Data:
 - Requested Attributes — groups of attributes from a single source credential
 - Requested Predicates — true/false numeric expressions (no strings!!)
 - Built from an attribute, a value and an operator — one of “<”, “<=”, “>”, “>=”
 - Works great for dates, but they MUST be integers — “dateint” or “Unix Time”
- Restrictions — logical AND/OR expression about acceptable source credentials
- Revocation intervals, combining both
 - Request for proof of non-revocation for some/all source credentials
 - Acceptable time period for proving non-revocation
 - “Was your insurance credential valid on June 26, 2021 when the accident occurred?”
 - “I’m offline, but I have revocation registry info from last week to yesterday”

Example Presentation Request

```
{
  "name": "proof-request",
  "nonce": "1234567890",
  "version": "1.0",
  "requested_attributes": {
    "studentInfo": {
      "names": [
        "given_names",
        "family_name"
      ],
      "restrictions": [
        {
          "schema_name": "student id"
        }
      ]
    }
  },
  "requested_predicates": {
    "not_expired": {
      "name": "expiry_dateint",
      "p_type": ">=",
      "p_value": 20230527,
      "restrictions": [
        {
          "schema_name": "student id"
        }
      ]
    }
  }
}
```

Presentation Request Restrictions

- Logical equality expressions, ANDed and ORed together ([Specification section](#))
- Basic elements:
 - Schema Publisher Identifier
 - Schema Identifier (DID)
 - Schema Name
 - Schema Version
 - Issuer Identifier (DID)
 - Credential Definition Identifier
 - Credential Attribute Name
 - Credential Attribute Value

- Example:

```
"restrictions": [  
  {  
    "schema_name": "student id",  
    "schema_id": "0.1.1",  
    "schema_did": "<DID>",  
    "issuer_did": "<DID>",  
    "issuer_did": "<DID>"  
  },  
  {  
    "schema_name": "student id",  
    "schema_id": "2.1.1",  
    "schema_did": "<DID>"  
  }  
]
```

Diagram illustrating the structure of the restrictions array:

- The first restriction object is annotated with a red box labeled "ANDs" pointing to its internal fields.
- The two restriction objects are connected by a red box labeled "OR" pointing to the comma between them.
- The second restriction object is annotated with a red box labeled "ANDs" pointing to its internal fields.

Date Predicates

- Must have date as an integer
 - dateint — Integer YYYYMMDD
 - 20230531 or 20,230,531
 - Unix Time — seconds since Jan 1, 1970
 - Start of Workshop: 1685545200

- Example:

```
{
  "requested_predicates": {
    "not_expired": {
      "name": "expiry_dateint",
      "p_type": "<=",
      "p_value": 20230527,
      "restrictions": [
        {
          "schema_name": "student id"
        }
      ]
    }
  }
}
```

- Example:

```
{
  "requested_predicates": {
    "older_than_19": {
      "name": "dob_dateint",
      "p_type": ">=",
      "p_value": 20040527,
      "restrictions": [
        {
          "schema_name": "student id"
        }
      ]
    }
  }
}
```

- Does **NOT** work with an ISO (string) Date!

Holder Presentation Generation

- Receive (somehow...) the Presentation Request
- Search in secure storage (aka wallet) for credentials to satisfy the request
 - Each attribute group must come from the same source credential.
 - Multiple request/predicate groups may come from the same credential
 - Business logic to decide if not found or if multiple credentials found for a group.
- As needed, get credential definitions and revocation registry data.
- Generate a presentation from the request and the source credentials
 - For each source credential:
 - Generate a proof across all encoded attributes in the credential
 - If necessary, generate a non-revocation proof
 - Reveal attributes (raw values)
 - Generate a proof per predicate
 - Generate the aggregate proof using the link secret for each credential — must all use same LS

Presentation Verification

- Four levels:
 - Cryptographic verification — do all of the proofs verify?
 - Credentials, predicates, non-revocation proofs and the aggregate proof
 - Adherence to the Presentation Request
 - Were all the groups satisfied?
 - Was the revocation interval satisfied?
 - Agent framework processing
 - More verification of the presentation against the presentation request
 - Unrevealed attributes
 - Raw values matching the encoded values (should be in the first level!)
 - Business logic
 - Is the presentation sufficient for the business purpose?
 - Do we trust the issuer?

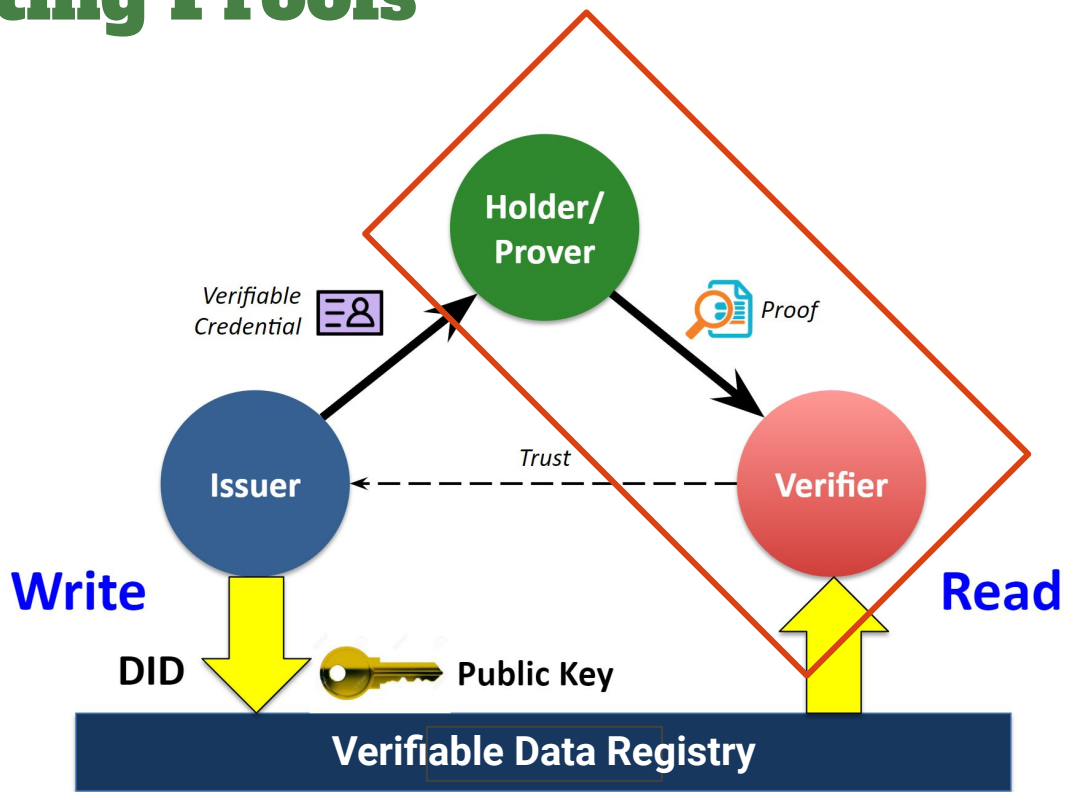
“Trust the Issuer” Handling

- Pre-verification in the Presentation Request
 - Restrictions in presentation request limit credentials from specific, trusted issuer(s)
 - Example: The only issuer of a specific type of credential
 - Example: Government IDs from specific jurisdictions
- Post-verification
 - Restrictions limit credentials based on attributes/schema, but allow credentials from **any** issuer
 - Problem: Anyone can be an issuer, use the schema, and issue **themselves** a credential.
 - Solution:
 - After in the business verification, check the issuer
 - Local list — same as built-in, but not expressed in presentation request (e.g., too many)
 - Trust registry — trusted, external list of trusted issuers
 - Dynamically — as new issuers are encountered, have humans verify them.

Lab 3: Issuing Credentials

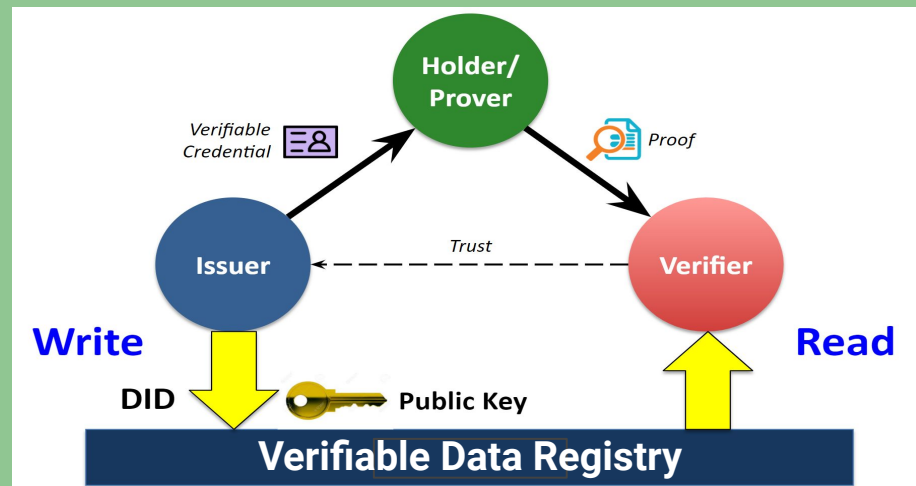
- Lab 2 link: <https://bit.ly/AnonCredsLab3>
- Steps:
 - Pick a Connection
 - Pick a Credential Definition
 - Fill in the Data (Click “*Enter Credential Value*” in Traction for UI)
 - Issue the Credential
 - In Wallet: Accept the Credential

Presenting Proofs

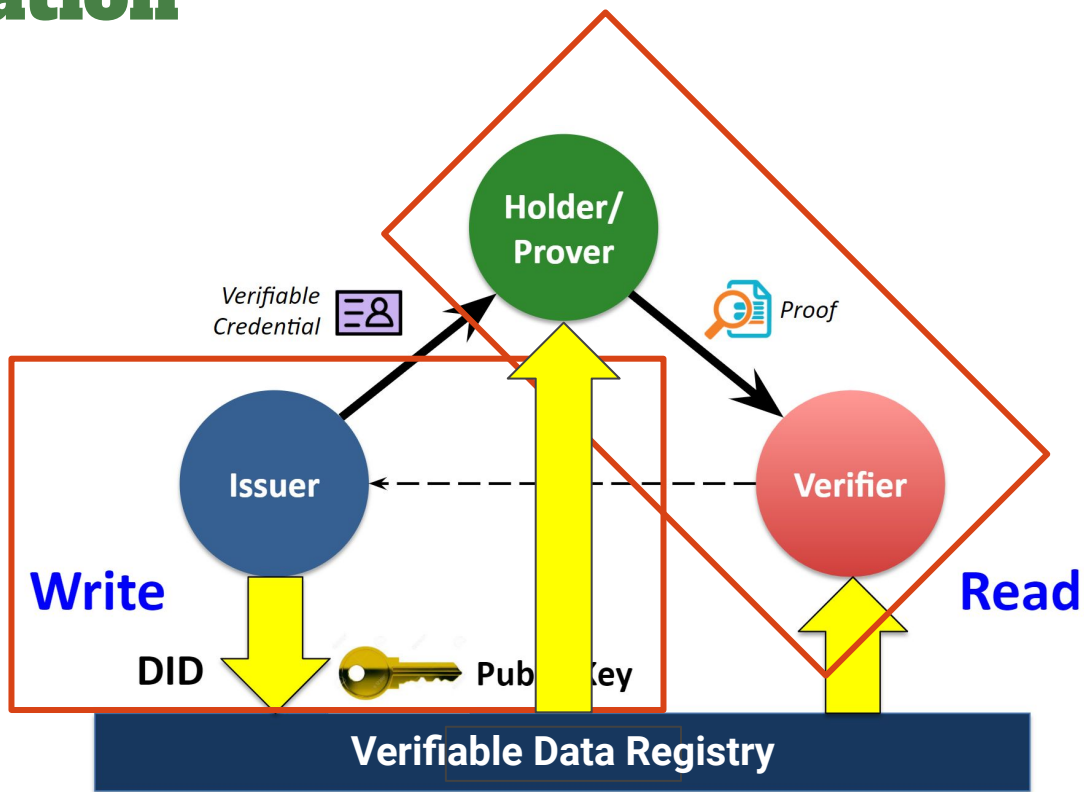


AnonCreds: Data and Processes

Revocation (briefly)



Revocation



Revocation

- An issuer issues a credential, and later decides to revoke it.
- Why?
 - The credential is no longer accurate.
 - Data has changed in the credential — e.g., change of address.
 - Holder's authorization to use the credential has changed — e.g., loss of driver privileges.
 - Someone made a mistake in issuing the credential.
 - Note: Expiration can be handled separately without revocation. Put in an “expires” attribute.
- Action is unilateral by the Issuer, perhaps with a notification to the Holder.
- Revocations are published somewhere accessible to parties needing them.
- Verifier can detect revocation in subsequent presentations.
 - Depending on the revocation scheme, a verifier may be able to monitor revocations.



AnonCreds Revocation

- At the AnonCreds level — the most complicated part
 - AnonCreds 1.0 Revocation works, but is not great — does not scale very well for huge use cases
 - Relatively easy for the Holder and Verifier, but lots of work for the Issuer!
- Mechanism: Accumulator-based Non-Revocation Proof
 - Holder proves their credential is not revoked, WITHOUT revealing the ID of their credential.
 - How?
 - A large prime number is associated with every credential in a Revocation Registry
 - **Accumulator** is the modulo product of the primes of all unrevoked credentials
 - Holder creates ZKP using their prime, their **witness**, and the (published) accumulator
 - Their witness is the modulo product of all unrevoked of the primes of all unrevoked credentials **except** their prime
 - Verifier can verify the proof — and hence, the credential is not revoked.

Issuer Revocation Activities

- For Issuers: Frameworks simplify the activities — e.g. Aries Cloud Agent Python
 - When creating a Credential Definition, flag “Use Revocation”, and the RevReg size.
 - On issuing credentials, track the “RevocationID” for later use.
 - When needed revoke a holder’s credential, use their “RevocationID”.
 - When needed, publish revocations — as they happen, or periodically (e.g., daily).
- For Holders:
 - Retrieve the “tails file” for their RevReg
 - When creating a presentation for a source credential, retrieve a RevRegEntry (state) from the VDR.
 - AnonCreds does the rest — creating a Non-Revocation Proof to include in the presentation.
- For Verifiers:
 - Retrieve the RevRegEntry the Holder used from the VDR.
 - AnonCreds does the rest — verifies the Non-Revocation Proof.
 - No correlatable identifier given.
 - No way to monitor the revocation state of the credential going forward.

Frameworks Hide Revocation Complexity

- Each RevReg is a limited size (number of credentials)
 - When credentials in a RevReg are used, must create a new RevRef before issuing more credentials.
 - ACA-Py keeps track of RevReg usage, creates one RevReg ahead, and the issuer is never runs out.
- A “tails file” per RevReg must be published for the holder to download
 - ACA-Py handles that, along with a specialized “Tails Service”
- When publishing revocation batches for a type of credential, an Issuer may have to write multiple RevRegEntries (updates)
 - ACA-Py tracks unpublished revocations across all revocation registries for a CredDef

Limitations of AnonCreds 1.0 Revocation

- The Tails File holds all of the primes for the RevReg.
- Must be retrieved by the Holder, often a mobile wallet app.
 - Limited bandwidth, limited storage
- Tails file increase in size linearly with the number of credentials in the RevReg.
 - 1000 Credentials / 250k file, 10000 / 2.5M file, etc.
- Practical limit is about 5,000-10,000 credentials per RevReg.
- Can have many RevRegs for a Credential type (so still unlimited credentials)

Additional AnonCreds v1.0 Revocation Details

- Handling of the presentation request “revocation_interval”
 - Semantics
 - Best practices
- Verification that a given presentation is within the “revocation_interval”
- Handling revocation interval when a source credential does not support revocation, and vice versa
 - Verifier may not know if holder’s credentials is revocable or not.
 - E.g. Request “degree” credential issued by many universities.
 - Some may support revocation, others may not
 - Verifier does not know what credential the holder might have - revocable or not.
 - Presentation should work regardless!

Need more information on AnonCreds Revocation? Let me know...

Hyperledger AnonCreds Workshop

End of Part 1

Hyperledger AnonCreds Workshop

Part 2

ZKPs — using High School Math

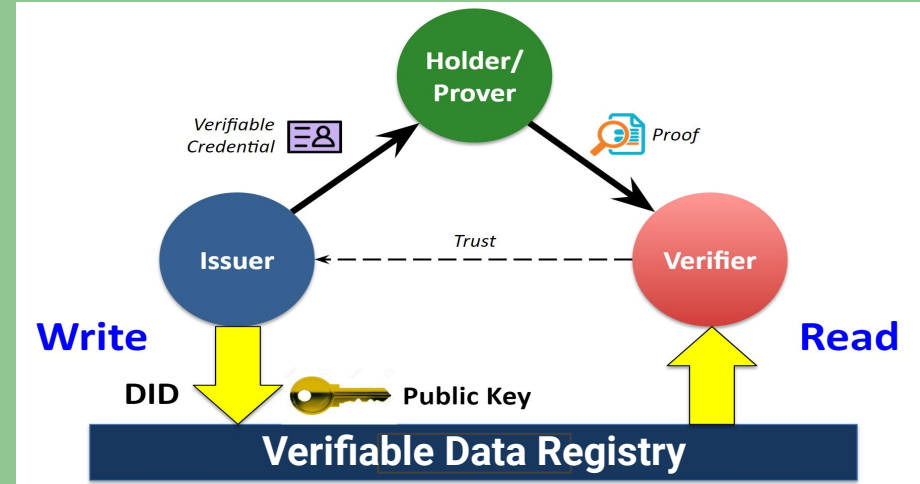
- Back to basics!
- [Slides](#)
- [Recording](#)

ZERO-KNOWLEDGE PROOFS

HIGH SCHOOL
MATH EDITION

CRYPTOGRAPHIC ✨MAGIC✨

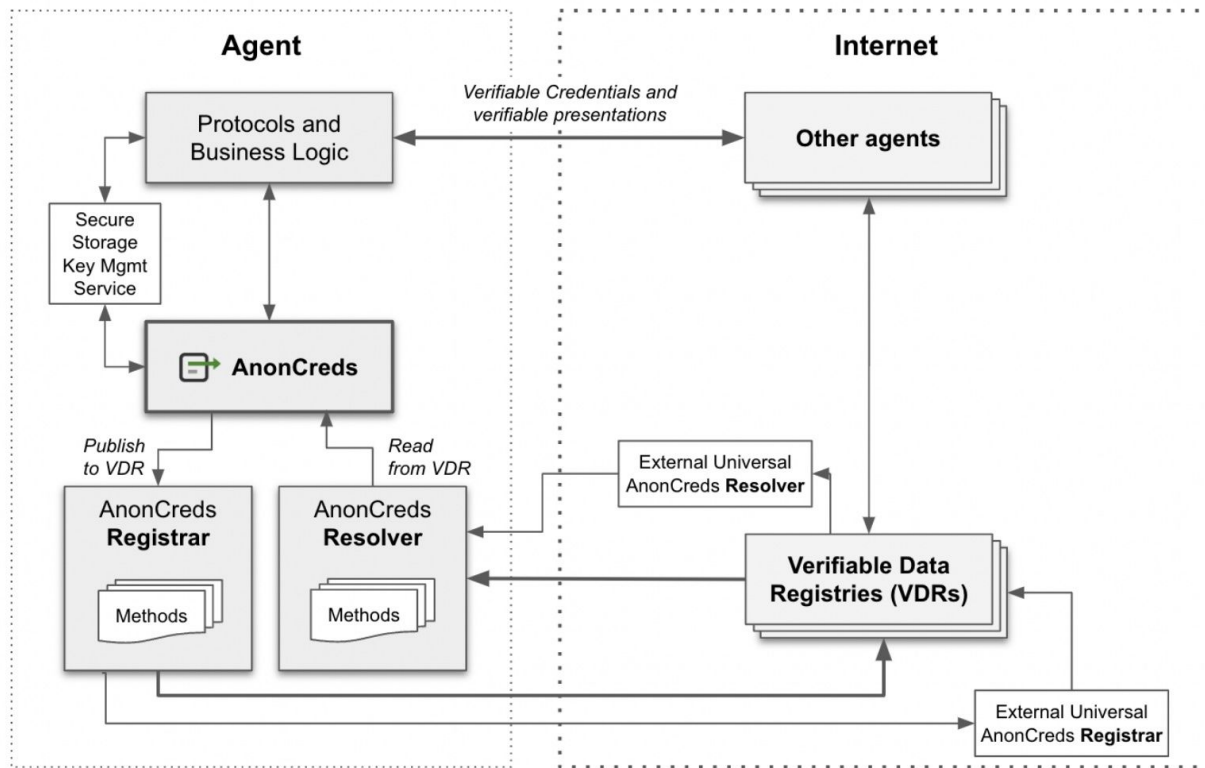
Ledger-Agnostic AnonCreds



Hyperledger Indy

- Historically, AnonCreds has been used primarily with Hyperledger Indy
 - Sovrin, IDunion, CANDy, FINdy, Swiss Test Instance, Indicio, BCovrin, etc.
- Indy was the “all-in-one” ledger+vc format+agents
- Aries agent portion extracted out in 2018
- AnonCreds extracted out in 2022
- Specification work revealed that there was little tying AnonCreds to Indy
 - Many were using AnonCreds independent of Indy
- In extracting the AnonCreds implementation from Indy, it was made “ledger-agnostic” — independent of Indy

VDR-Agnostic AnonCreds Architecture



Ledger-Agnostic AnonCreds

Rodolfo Miranda,
RootsID



<https://bit.ly/AnonCredsSlides>

AnonCreds in W3C VC Data Model Format

- Long a controversy in the verifiable credentials community.
 - AnonCreds does not use the W3C VC Data Model Format.
- Several initiatives have been started to align the models.
 - Most significant one (“Rich Schemas”) fizzled out because it added lots of complexity.
- What to do?



AnonCreds & the W3C VC Data Model



AnonCreds to W3C VC Format, and Back

- It turns out...putting AnonCreds in W3C VC Format is pretty easy...
- [Proof of Concept](#) (code and examples):
 - AnonCreds **Credential** into W3C **VC** Data Model Standard into AnonCreds **Credential**
 - AnonCreds **Presentation** into W3C **VP** Data Model Standard into AnonCreds **Presentation**
- Done by moving around JSON
 - Signature on VC and VP contain identical **data** to AnonCreds Credential/Presentation
 - AnonCreds canonicalization, signing, creating presentation, verifying presentation stay the same
 - All features of AnonCreds fully supported

AnonCreds ⇔ W3C

```
{
  "schema_id": "3av...s8W:2:fabername:0.1.0",
  "cred_def_id": "3av...s8W:3:CL:13:default",
  "rev_reg_id": null,
  "values": {
    "name": {
      "raw": "Alice Jones",
      "encoded": "728...2918"
    }
  },
  "signature": {
    "p_credential": {
      "m_2": "5783...397",
      "a": "203...785",
      "e": "259...767",
      "v": "626...819"
    },
    "r_credential": null
  },
  "signature_correctness_proof": {
    "se": "163...839",
    "c": "546...523"
  },
  "rev_reg": null,
  "witness": null
}
```

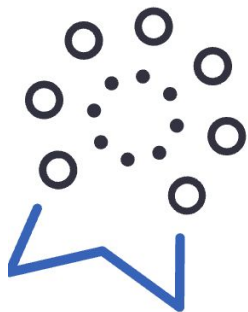
```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://bit.ly/anoncreds-context",
    { "@vocab": "urn:anoncreds:attributes#" }
  ],
  "type": [
    "VerifiableCredential",
    "AnonCredsCredential"
  ],
  "issuer": "did:sov:3avoBCqDMFHFaKUHug9s8W",
  "issuanceDate": "2022-12-15T01:17:32Z",
  "credentialSchema": {
    "type": "AnonCredsDefinition",
    "id": "did:sov:3avo...9s8W:3:CL:13:default",
    "schema":
      "did:sov:3avo...9s8W:2:fabername:0.1.0"
  },
  "credentialSubject": {
    "name": "Alice Jones"
  },
  "proof": {
    "type": "CLSignature2022",
    "encoding": "auto",
    "signature": "AAAgf9w5..."
  }
}
```

Crypto-Agility

- VC Crypto-Agility — adding multiple types of signatures to a credential
 - Flexibility and long term vulnerability protection (e.g. NIST-approved + Ed25519 + Quantum-safe)
- Works with AnonCreds (PoC carried out)
 - Convert AnonCreds Credential to W3C VC Data Model Standard format, with AnonCreds Signature
 - Pass to a LD-Signature library to generate, add signatures to VC
- Result: Present VC using AnonCreds or not
 - Generate an AnonCreds presentation, with all it's privacy-preserving goodness
 - Present the VC directly using one or more LD-Signatures (but no AnonCreds capabilities)
- Enabled by
 - An AnonCreds JSON-LD context and format for the AnonCreds credential that enables compliant JSON-LD processing to create a signature.

AnonCreds and CHAPI

Patrick St-Louis,
Digital Identity Lab of Canada
Laboratoire d'identité numérique
du Canada
idlab.org



Digital Identity
LABORATORY

<https://bit.ly/AnonCredsSlides>

AnonCreds and Overlays Capture Architecture

Making Credentials
Beautiful!



Verifiable Credential Metadata

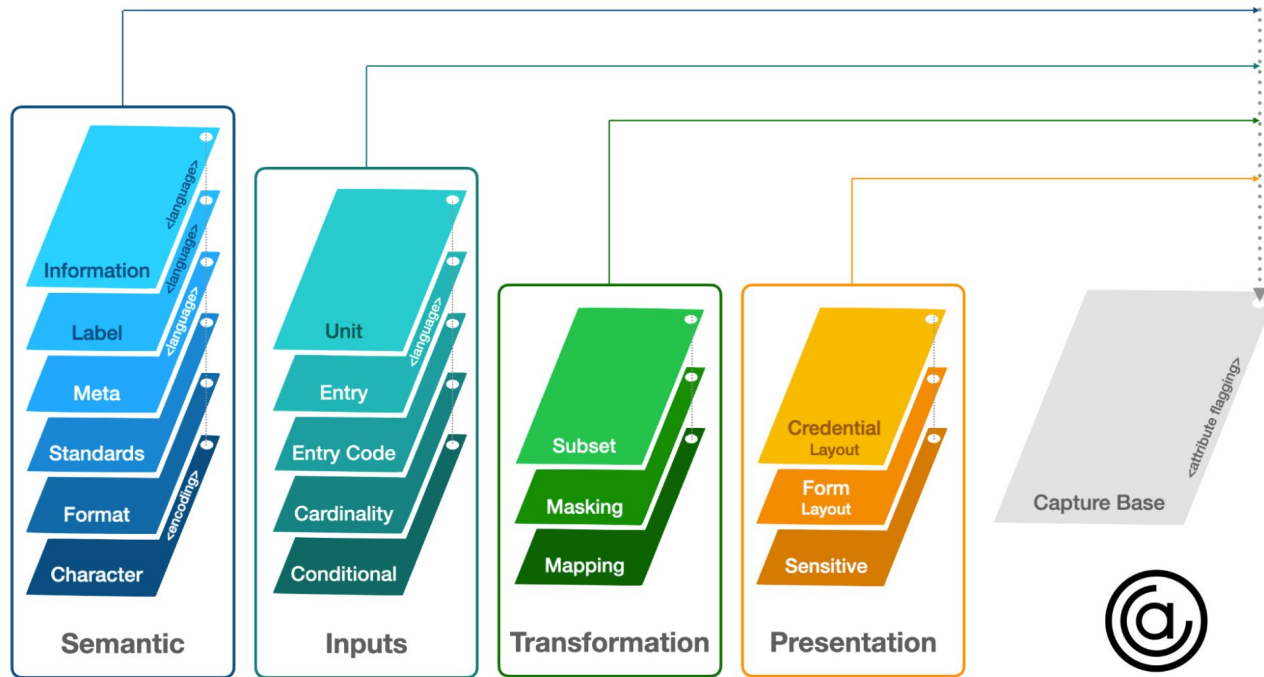


Focus for *now* is on AnonCreds verifiable credentials in Aries.

- Problem: AnonCreds provides little metadata about credentials
 - Issuer DID, but no standardized way to get more information about the issuer (name, description)
 - Self-asserted “Alias” on the DIDComm connection between the holder and the issuer
 - Name and version of [Schema object](#)
 - Schema is a simple list of attributes (e.g. “given_name”, “family_name”, “birth_dateint”, etc.)
 - Credential Definition — an [object](#) created by an Issuer that enables issuing credentials using a schema object
 - Only remotely human identifier is a tag and link to the schema object
- Issuer Branding? Nothing!

How can we present beautiful, multilingual credentials with so little data?

Overlay Capture Architecture (OCA) Specification



OCA Overlay Format

- *Dataset*-based, rather than *attribute*-based as is JSON-LD
- JSON
 - Array of overlays
 - Overlays can be in separate files
- Each overlay has a hash to link to “Capture Base”
 - A “SAID” Self-Addressing ID
- In some overlays, country-language code
- In most overlays, an attributes list from the capture base with a value per attribute

```
{  
  "capture_base": "EfeQ...t0cNsc",  
  "type": "spec/overlays/label/1.0",  
  "language": "fr",  
  "attr_labels": {  
    "given_name": "Nom",  
    "family_name": "Prénom",  
    "birth_dateint": "Date De Naissance",  
    "parent_1_name": "Nom Du Parent 1",  
    "parent_2_name": "Nom Du Parent 2",  
    "issued_dateint": "Date D'Émission",  
    "expiry_dateint": "Date D'Expiration",  
    "photo": "Photo"  
  }  
}
```

< Credential Details

5E6HVk25KqZniFJHy student_card:2.3
MuhJb

Issued: September 26, 2022

Hide all

student_first_name

Alice Hide

student_last_name

Smith Hide

expiry_date

20261026 Hide

Home Scan Credentials

< Credential Details

BestBC College Student Card

Issued: September 26, 2022

Hide all

First Name

Alice Hide

Last Name

Smith Hide

Expiry Date

October 26, 2026 Hide

Home Scan Credentials

< Détails des justificatifs

BestBC College Carte d'étudiant

Date de délivrance : 26 septembre 2022

Masquer tout

Prénom

Alice Masquer

Nom de famille

Smith Masquer

Date d'expiration

26 Octobre 2026 Masquer

Accueil Numériser Justificatifs

Challenge: Delivering OCA Bundles

- Options
 - Could be published by Schema Publisher — example Canadian “Person” credential, where issuers are Provinces.
 - Could be the Issuer.
 - Could be an independent party that the Wallet trusts — creates OCA Bundles on behalf of issuers.
 - Likely to be a single party — we don’t expect to collect overlays published by multiple parties.
- Several options for publishing OCA bundles considered.
 - Linked into the Credential (e.g. a hashlink)
 - Delivered with the Credential
 - Put on a ledger
- Short term decision...

OCA Bundle GitHub Repository

- Initial cut in the BCGov GitHub organization: [aries-oca-bundles](#) but intended for wider use (Canada, Aries)
 - With an understanding that this will only get us so far — a year or two would be great...
- Defines some [governance](#):
 - Repository “Editors”, have a similar role to Open Source Maintainers in processing pull requests (PRs)
 - PR is an OCA Bundle from an “authorized” community member (“human” verification)
 - The OCA Bundle matches the schema (which is a real AnonCreds schema)
 - Images are actually images, etc.
 - Enforce that some needed metadata about the OCA Bundle is provided — e.g., submitting org, authorized updaters
 - **Don’t** enforce anything about the credential format or OCA Bundle content — translations, colours, etc.
- Tools
 - Generate OCA Bundle from source files — Excel and Branding JSON file
 - Generate a list of OCA Bundles in the repository for download — Identifier + path to OCA Bundle
 - OCA Explorer

OCA Explorer

Click Image to Open

The screenshot shows the OCA Explorer interface. At the top left is the British Columbia logo. Below it is a dropdown menu for 'Bundle ID' with the selected option 'BC Best College Demo Student Card'. To the right of the dropdown is '- OR -'. Below that is a blue button with a document icon and the text 'Upload OCA Bundle'. Underneath is a 'Language' section with two radio buttons: 'en-CA' (selected) and 'fr-CA'. Below the language section is a 'STUDENT CARD' section with a help icon and the text 'An example Student Card that might be issued from a Canadian University or College' and 'Best BC College'. At the bottom, there are two student card examples. The first is a dark green card with the Best BC College logo and the text: 'Best BC College Student Card', 'Student Last Name Smith', and 'Student First Name Alice'. The second is a card with a library background and a dark green bottom section with the Best BC College logo and the text: 'Best BC College Student Card'. Below the second card, the text 'Student Last Name Smith' and 'Student First Name Alice' is displayed.



AnonCreds v2.0

What's Coming?

AnonCreds v2.0

- A working group started in early 2023 to define “AnonCreds Next”

The goal of AnonCreds v2.0 is to retain and extend the privacy-preserving features of AnonCreds v1.0, while improving capabilities, performance, extensibility, and security.

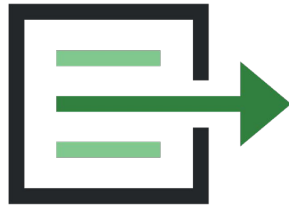
AnonCreds 2.0: Expectations

- Enable support for richer ZKP capabilities, beyond predicates.
 - Range proofs — “My salary is between \$50,000 and \$80,000”
 - Enumerations — “I live in one of a set of countries”
 - Encrypted Identifiers — “Here is my encrypted, unlinkable Credit Card number”
 - ZKP Attribute Equality — “Prove these attributes are the same, without revealing them”
- Enable the use of different underlying cryptographic signatures.
 - Most likely BBS+ or PS Signatures.
- Scalable revocation.
 - Urgently needed — may come before the rest. Help wanted!!
- W3C Verifiable Credentials Data Model Standard format.
- DIF Presentation Exchange for Presentation Request, Presentation formats.
- Relatively small changes in the Data Objects, and Data Models.
 - E.g., Same objects (perhaps with one more), same relationships.



Wrap Up

Where to go next?



HYPERLEDGER ANONCREDS

Goals

- Understand verifiable credentials in general
- Understand AnonCreds unique capabilities
- Experience the steps in using AnonCreds
- Understand (a bit of) the cryptography in AnonCreds
- See how AnonCreds are used today in a variety of contexts
- New capabilities available with AnonCreds
- The future of AnonCreds

Takeaways

- Use AnonCreds in your applications
- Build frameworks that embed AnonCreds
- Enable rooting AnonCreds in other ledgers/VDRs
- Contribute to the AnonCreds implementations
- Contribute to future AnonCreds specifications and implementations

Using AnonCreds

- Use one of the Hyperledger Frameworks, and you have full AnonCreds support
 - [Aries Cloud Agent Python](#)
 - [Aries Framework JavaScript](#)
 - [Aries VCX](#)
 - [Mobile Wallet: Aries Bifold](#)
- Look at the various Aries commercial offerings that support AnonCreds
- Join the [Hyperledger Discord](#) community, with channels on Aries and AnonCreds
- edX courses:
 - [LFS 172x - Introduction to Hyperledger Self-Sovereign Identity Blockchain Solutions](#)
 - [LFS 173x - Becoming an Aries Developer](#)

Build Frameworks that Embed AnonCreds

- Open Source Aries Frameworks (listed previously)
 - Join the Aries community working groups/calls — [Aries](#) and [AnonCreds](#)
- Your proprietary framework — add AnonCreds!
- Add support for AnonCreds beside other credential formats

Enable Rooting AnonCreds In Other VDRs

- Follow the lead of others in supporting AnonCreds on VDRs:
 - Hyperledger Indy
 - cheqd.io
 - Cardano
 - Juno Network — [NYMLAB](#) added an AnonCreds verifier into an onchain smart contract
 - did:web
 - and more...

Contribute to Current Hyperledger AnonCreds

- [Main page](#)
- [Meetings calendar](#)
- [AnonCreds v1.0 Specification](#)
- [AnonCreds v1.0 Implementation](#) — Rust, plus wrappers
- [CL Signatures Implementation](#) — Rust

Contribute to Next Hyperledger AnonCreds

- [AnonCreds v2.0 Specification](#) — early days
- Key Projects:
 - AnonCreds in W3C Format
 - Scalable, ZKP-based Revocation



Thanks for Joining Us!