CS61065: Theory and Applications of Blockchain

IDENTITY MANAGEMENT

Department of Computer Science and Engineering

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

Sandip Chakraborty
sandipc@cse.iitkgp.ac.in

What is Identity?

- People are known by their identities - drives every business and social interactions

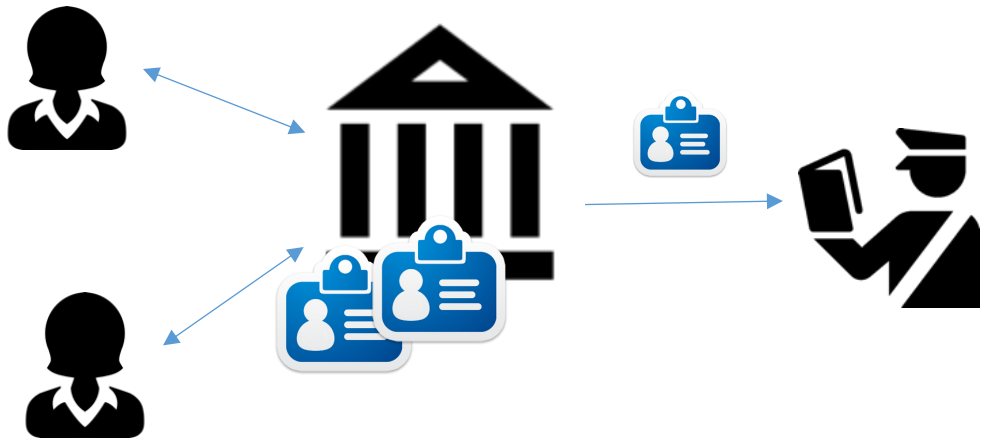
- Physical Identity is a collection of attributes
  - Name
  - Age
  - Financial history
  - Work history
  - Address history
  - Social history

Source: https://securityintelligence.com/reimagining-the-future-of-identity-management-with-blockchain/

Centralized Digital Identity

- Individuals do not have any control over the information that comprises their identities

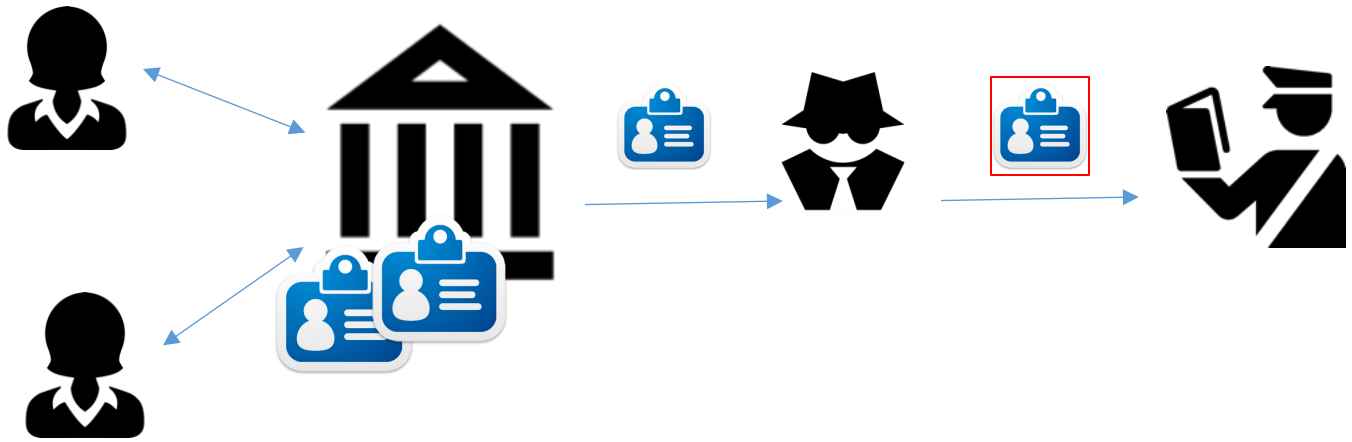
- **Identity fraud** - no visibility over the identity attributes
  - Authentication
  - Authorization
  - Verification



Centralized Digital Identity

- Individuals do not have any control over the information that comprises their identities

- **Identity fraud** - no visibility over the identity attributes
  - Authentication
  - Authorization
  - Verification



Digital Identity - Single Sign On (SSO)

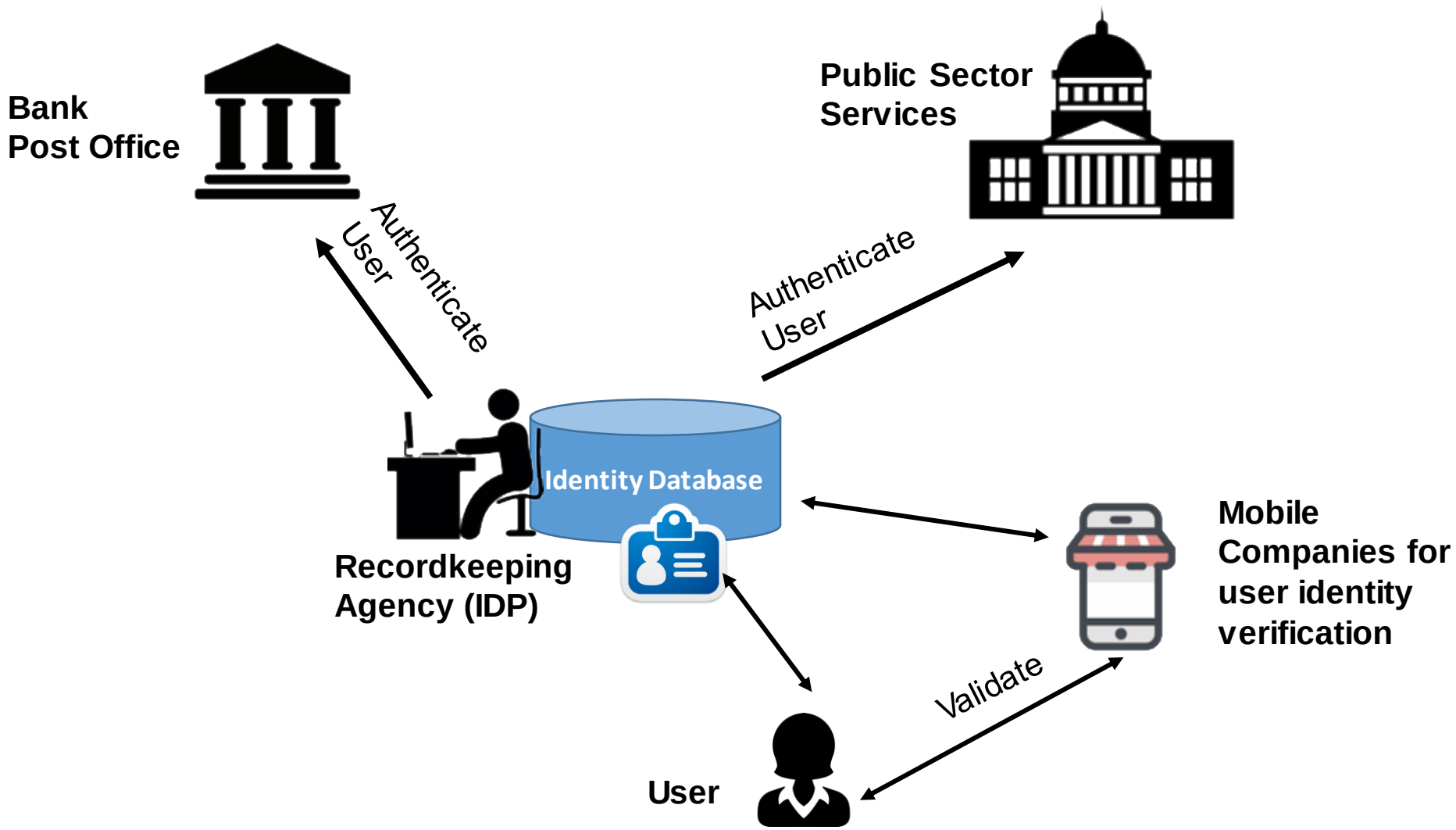
- **Single identity for various purposes**
  - No need to maintain multiple identity documents
- Widely conceptualized in software industry
  - One password to access multiple services

- Single identity provider (IDP) maintains the identity

- Identity consumers (services) use the IDP to authenticate the identity holder
  - **During authentication, the identity is not exposed to the services**

Image Source: https://www.e-spincorp.com/global-theme-and-feature-topics/single-sign-on-sso/

SSO and Decentralization



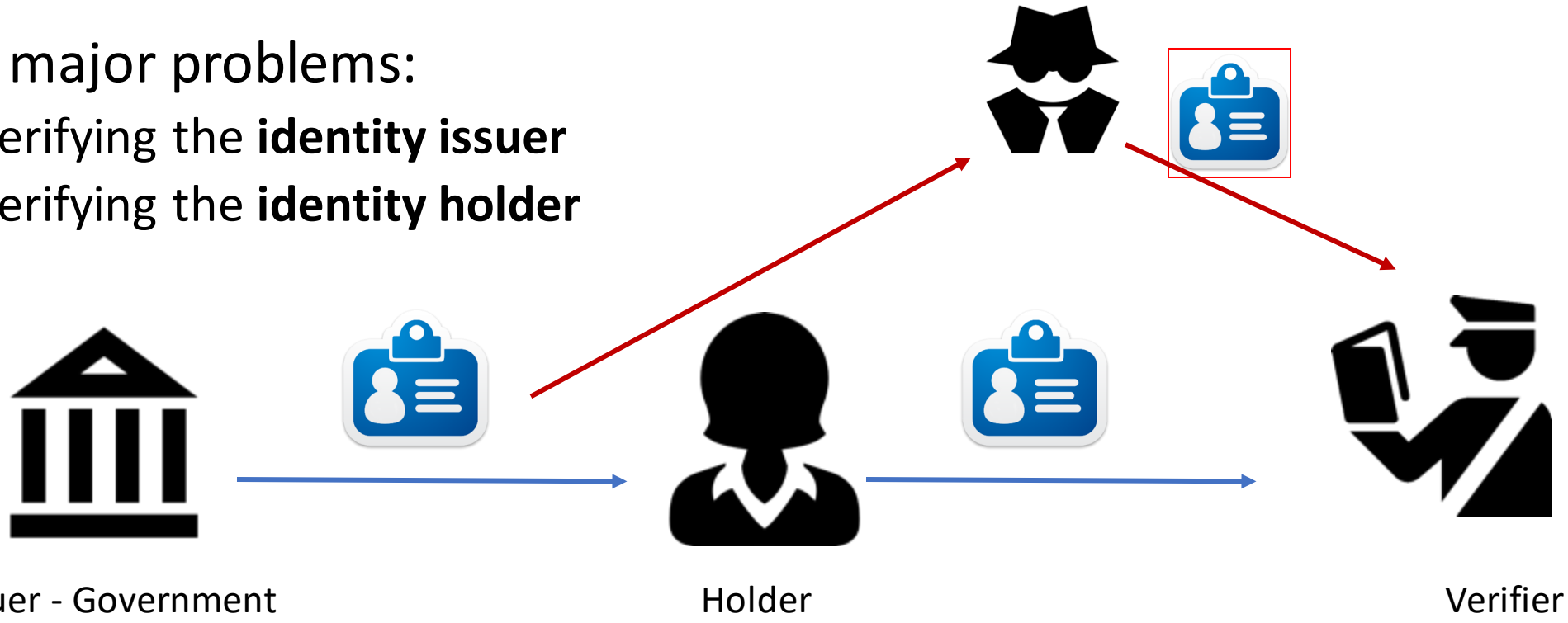
Decentralizing Digital Identity

- No Centralized Trusted Identity Provider / Registry
- Digital representation of physical identity.
- Two major problems:
  - Verifying the **identity issuer**
  - Verifying the **identity holder**

Issuer - Government Holder Verifier

Decentralizing Digital Identity

- No Centralized Trusted Identity Provider / Registry
- Digital representation of physical identity.
- Two major problems:
  - Verifying the **identity issuer**
  - Verifying the **identity holder**



Fundamental Principles of Digital Identity Management

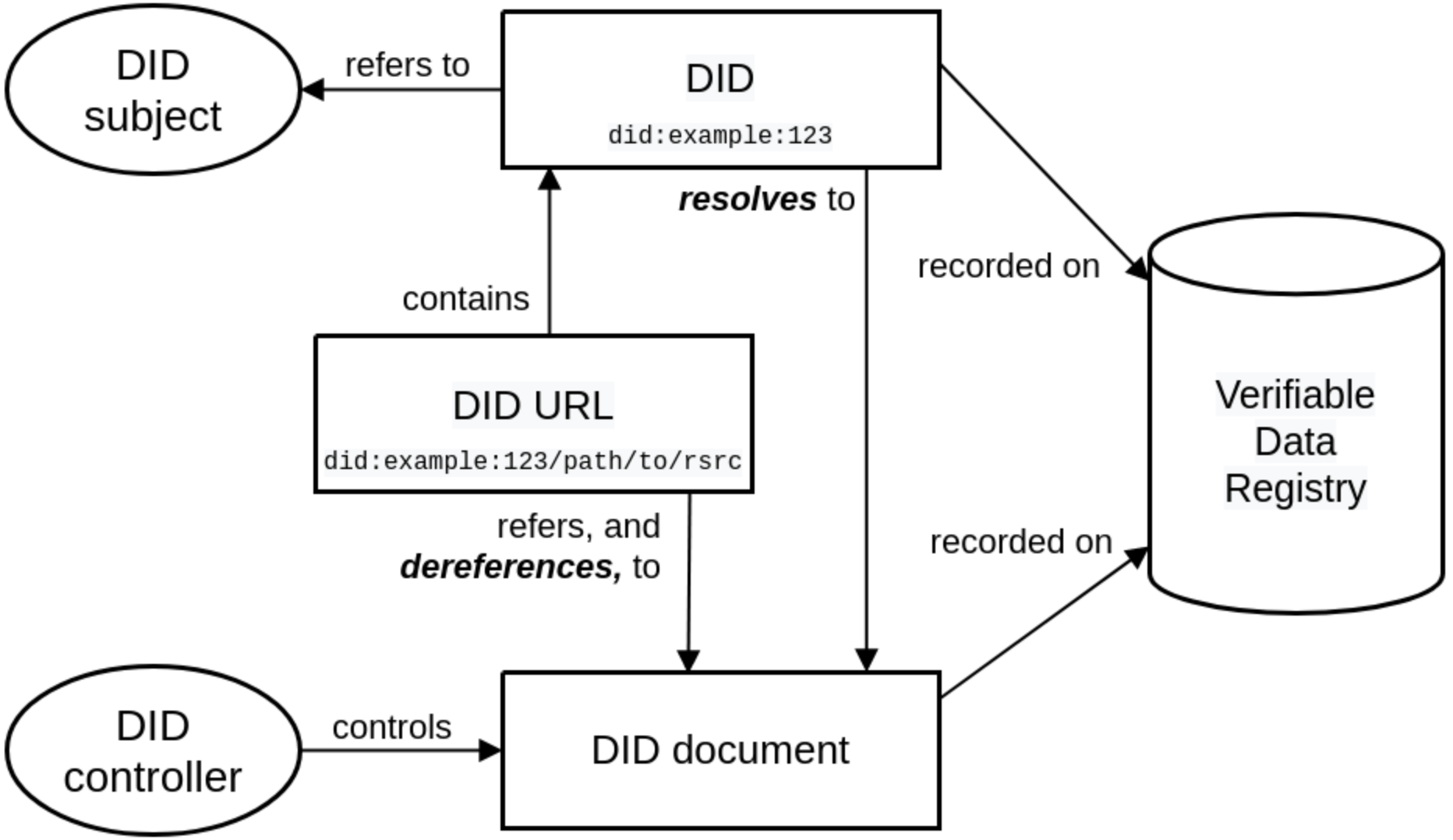
- **Self-Sovereign Identity (Privacy Control)**
  - Individual should have **full control and ownership of their identity** information
  - Individuals can **control the usage of their own identity** profile for business and social interactions (Consent - agreement for information usage)
  - **Identical to how we use our physical identity**
    - **Holder possesses the ID**
    - **Holder chooses whom to present the ID**
  - **Burden at individual user?**

Decentralized Identifiers (DIDs)

- Provides Verifiable, Decentralized Digital identity
- Designed to be decoupled from:
  - centralized registries
  - identity providers
  - certificate authorities
- Holder of DID can prove its ownership on the DID without the help of any other party.
- W3C Proposed Recommendation

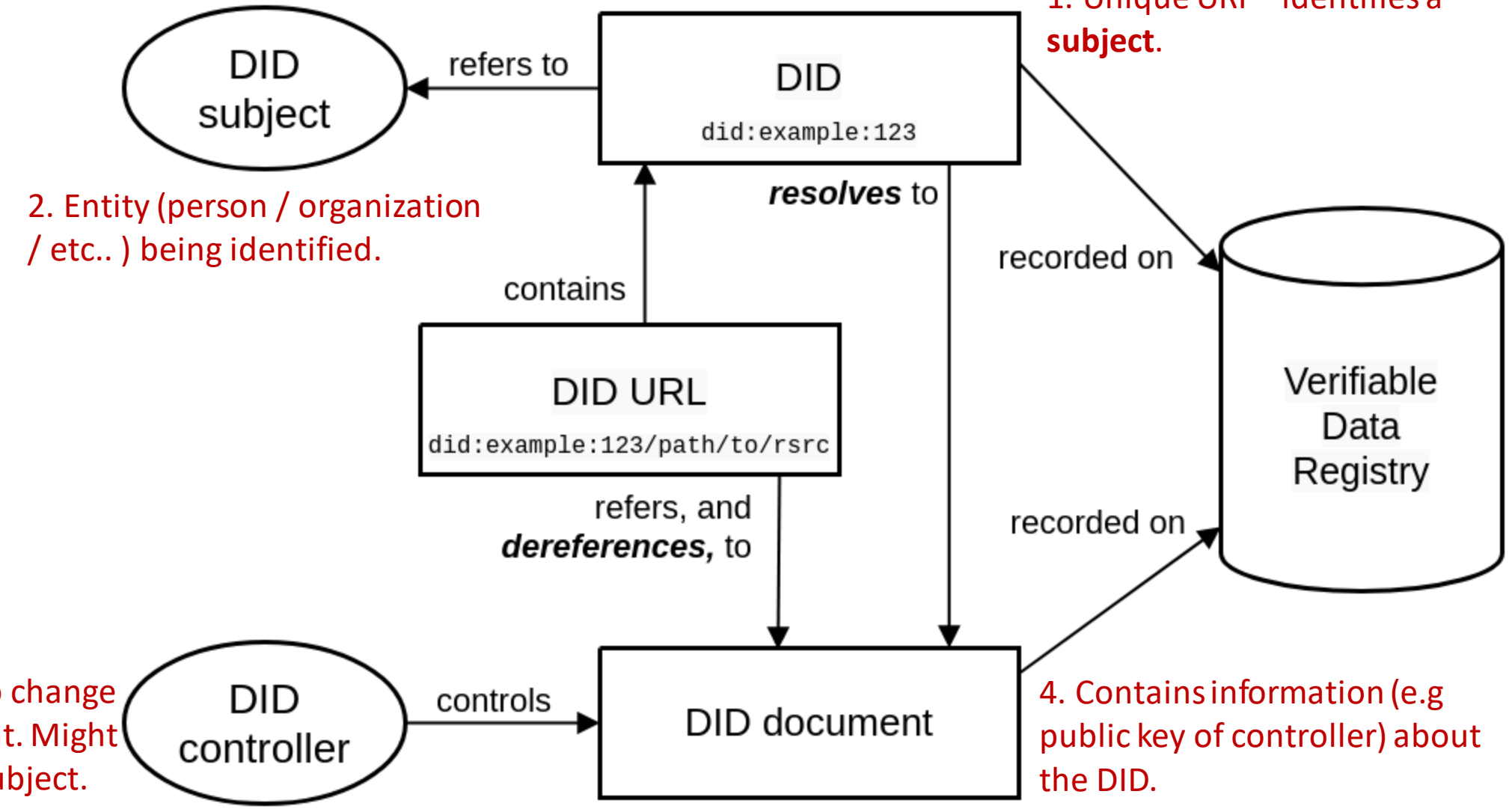
https://www.w3.org/TR/did-core/

DID Architecture



https://www.w3.org/TR/did-core/

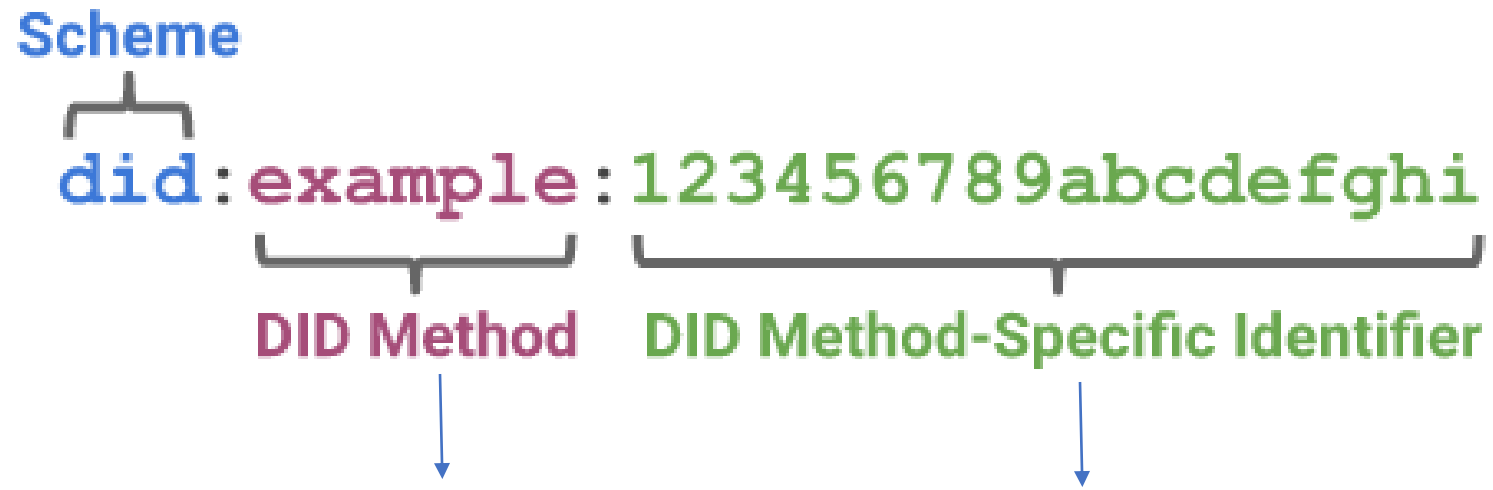
DID Architecture



https://www.w3.org/TR/did-core/

DID URI

- Controller controls a **DID Document**.
- A **DID** is a unique address (URI) to the location of that document.



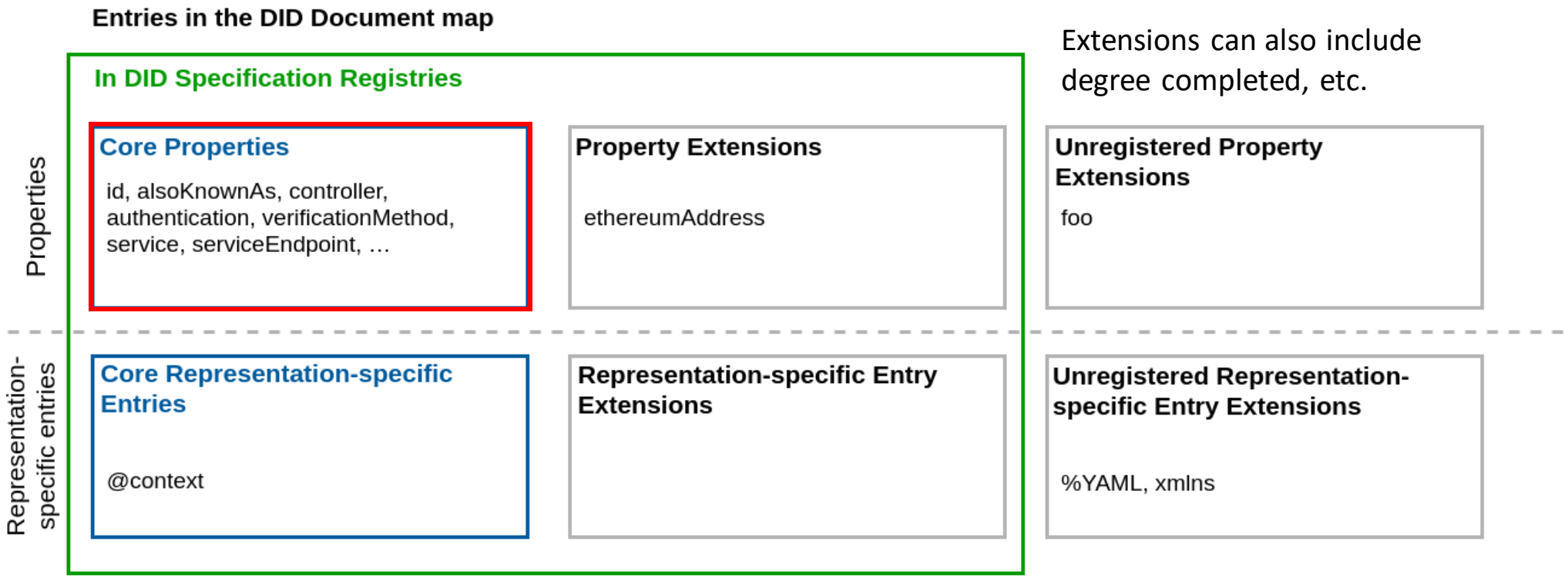
http://www.faqs.org/rfcs/rfc2396.html

https://www.w3.org/TR/did-core/

DID Document

A **DID Document** is a set of data describing the DID subject, including mechanisms, such as cryptographic public keys, that the DID subject or a DID delegate can use to authenticate itself and prove its association with the DID.

A DID document consists of a map of entries, where each entry consists of a key/value pair.



Extensions can also include degree completed, etc.

Representation-specific entries include JSON, XML, etc

https://www.w3.org/TR/did-core/

DID Document Example (JSON)

```
{
    "id": "did:example:123456789abcdefghi",

    "authentication": [{
        "id": "did:example:123456789abcdefghi#keys-1",
        "type": "Ed25519VerificationKey2020",
        "controller": "did:example:123456789abcdefghi",
        "publicKeyMultibase": "zH3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV"
    }],

    "service": [{
      "id":"did:example:123456789abcdefghi#linked-domain",
      "type": "LinkedDomains", // external (property value)
      "serviceEndpoint": https://bar.example.com
     }]
}
```

DID for a particular DID subject

Verification Method specifying how the DID subject can authenticate itself.

Service Endpoint denoting ways of communicating with the DID subject

It tells how to reach the subject. Otherwise, there is no meaningful use of authentication

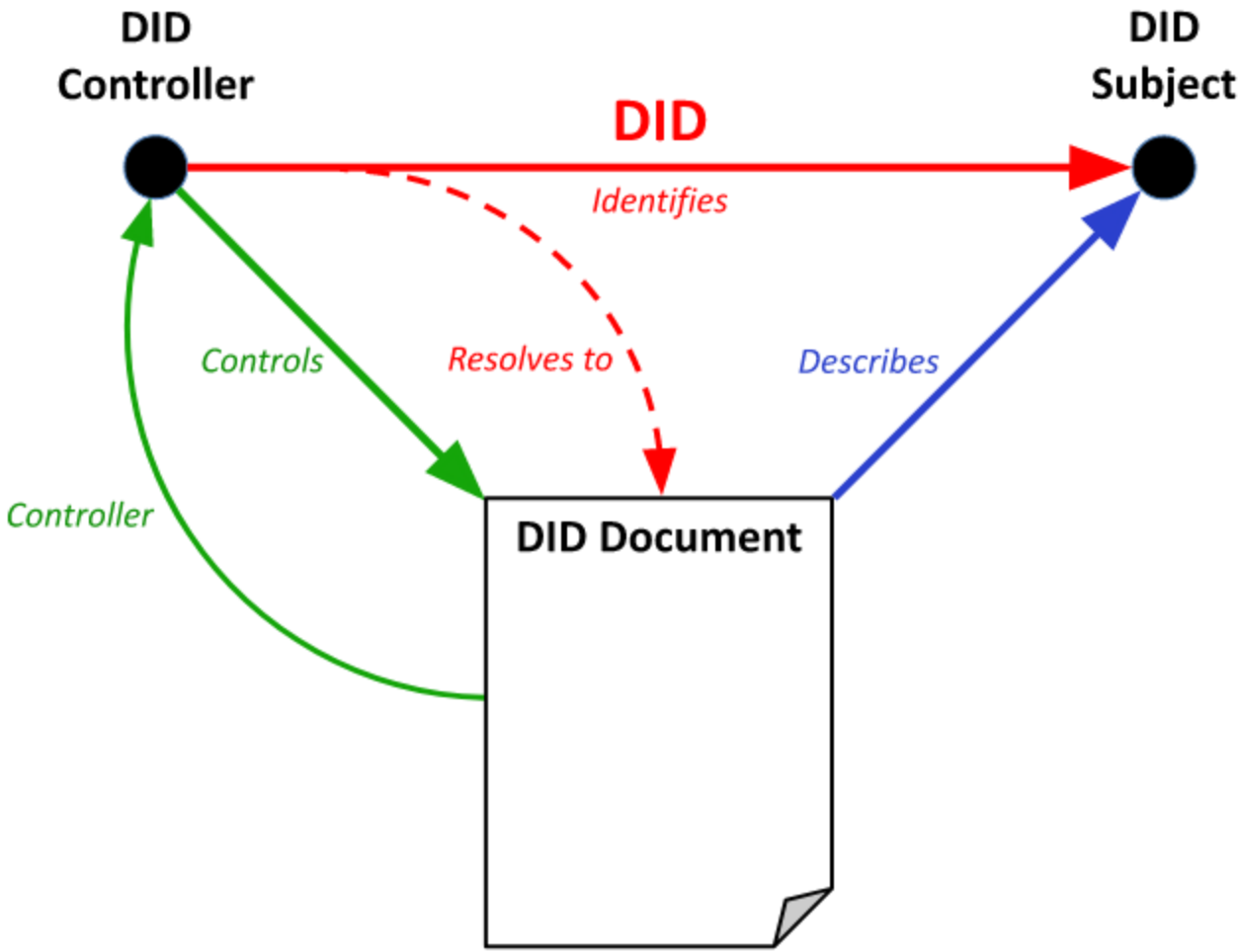
Note: There is no sensitive information in DID document

https://www.w3.org/TR/did-core/

Relationship Between DID, DID Document and DID Controller

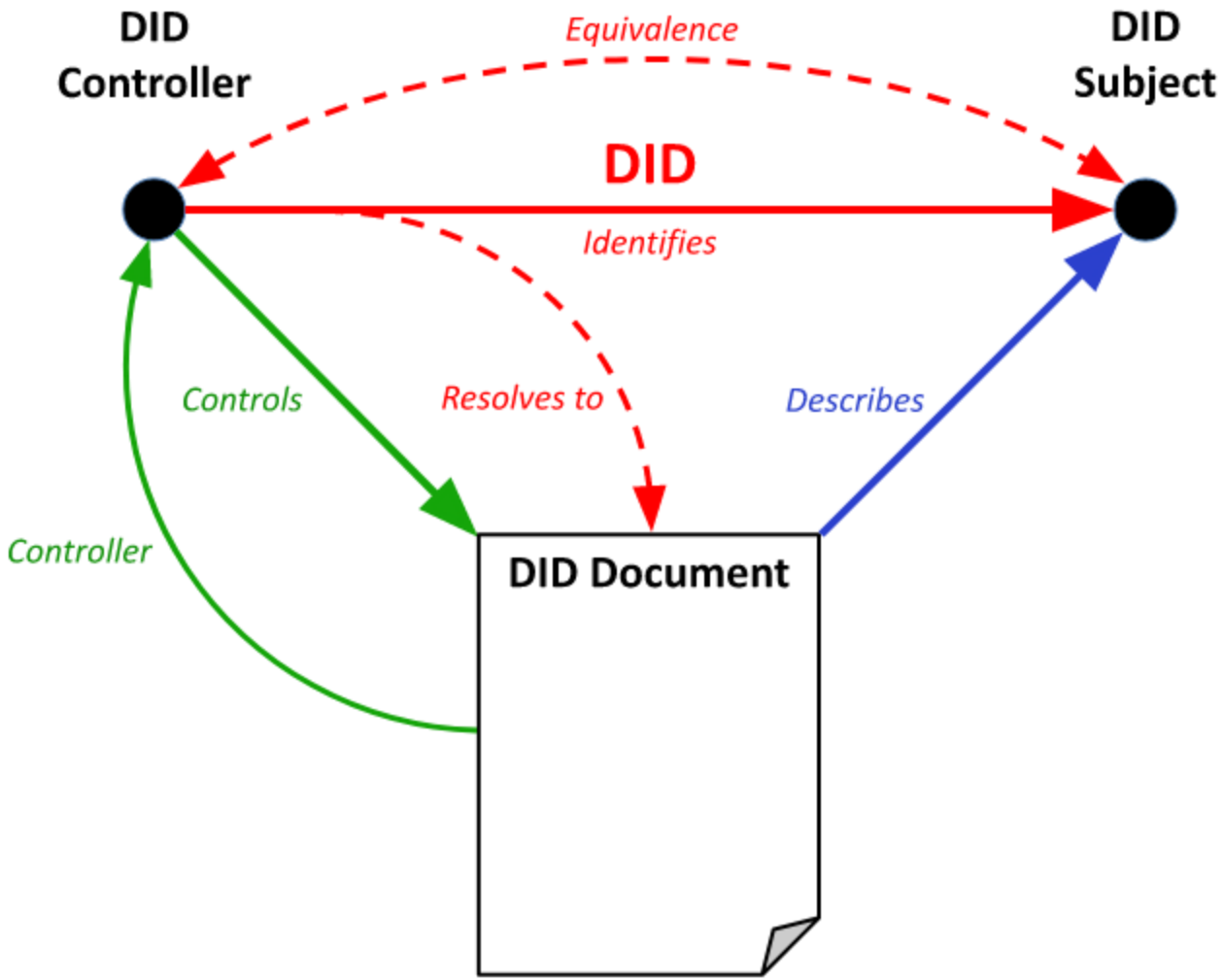
-A DID is an identifier assigned by a DID controller to refer to a DID subject and resolve to a DID document that describes the DID subject.

- The DID document is an artifact of DID resolution and not a separate resource distinct from the DID subject.
- DID document resides inside verifiable data registry



Relationship Between DID, DID Document and DID Controller

Often the DID Subject and the DID Controller are the same entity.



DID Flow – DID Registration

Alice

0. Generate Keys

Public key: **Pk**,

Secret key: **Sk**

1. Create DID Document

id: did:registry1:alice,

authentication: **Pk**

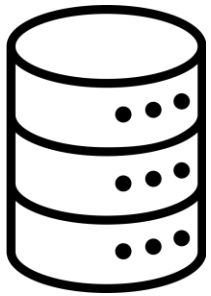
controller: did:registry1:alice

2. Register DID

DID Doc signedby(**Sk**)

3. Authenticates DID controller (here based on signature)

4. Updates DID Document



Verifiable Data Registry - (registry1)

DID Method Implementation

DID Flow – Proving Control over DID

Alice

0. Generate Keys

Public key: **Pk**,

Secret key: **Sk**

1. Create DID Document

id: did:registry1:alice,

authentication: **Pk**

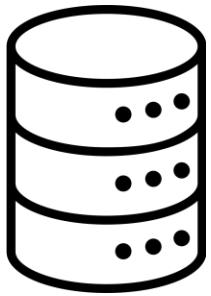
controller: did:registry1:alice

2. Register DID

DID Doc signedby(**Sk**)

3. Authenticates DID controller (here based on signature)

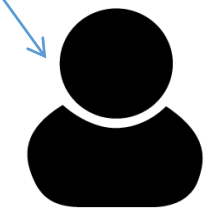
4. Updates DID Document



Verifiable Data Registry - (registry1)

5. Fetch DID Document for did:registry1:alice

6. Authenticate Alice based on authentication method **Pk**



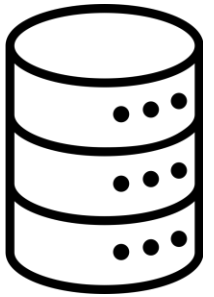
Bob

(wants to authenticate the subject of did:registry1:alice)

DID Method Implementation

DID Method Security

- DID Registry ideally enforces DID Method protocols.
- Centralized DID Registry brings in risks
  - Manipulating DID Documents
    - Changing authentication methods
  - Censoring DID Documents
    - Refusing to resolve certain DID Documents
- Lack of Transparency.



Verifiable Data Registry - (registry1)

DID Method Implementation

Centralized

Decentralized DID Registry

- Blockchain Based Implementation of Verifiable Data Registry
- DID Methods are implemented as smart contracts.
  - Smart contracts enforce how authorization is performed to execute all operations, including any necessary cryptographic processes.
- Transparent Immutable Ledger allows verifiability of DID Documents
  - Any party can validate if a DID Document's creation / updation transactions were authenticated or not.

Verifiable Data Registry

Blockchain based DID Registry

Public permissioned ledger based registry.

- Any party can read the ledger.
- Only selected (registered) parties and write to the ledger.

https://hyperledger-indy.readthedocs.io/en/latest/

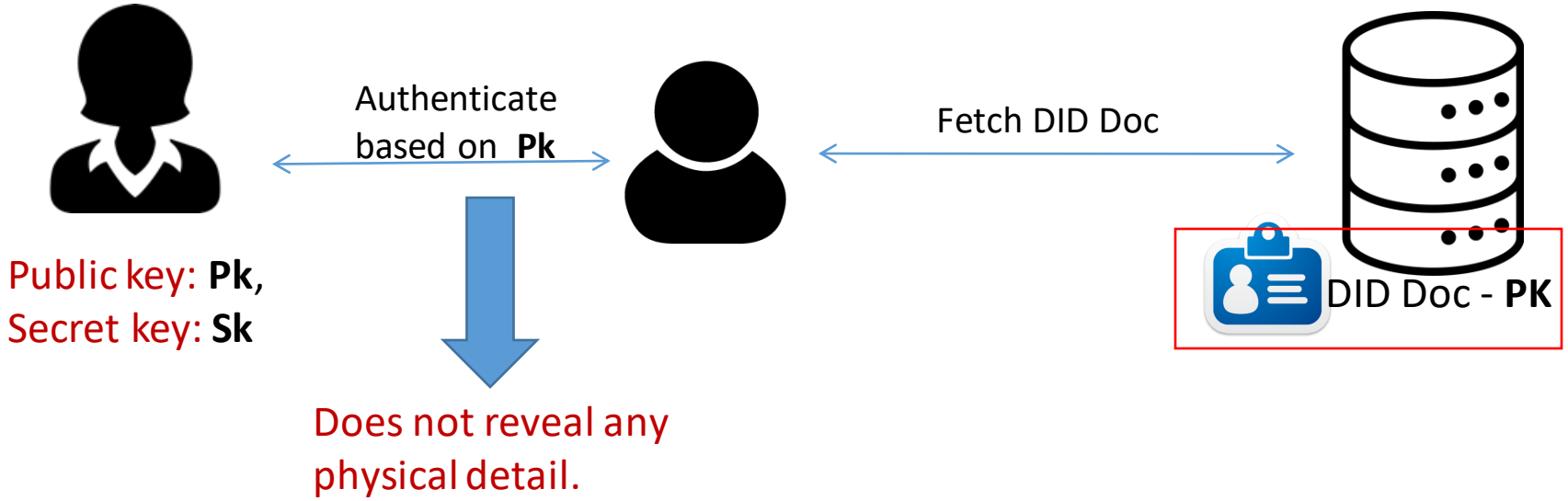
Sidetree

Protocol for creating scalable DIDnetworks that can run atop any existing permissionless blockchain. (e.g. Bitcoin, Ethereum, etc.)

https://identity.foundation/sidetree/spec/

Binding DID to Physical Identity

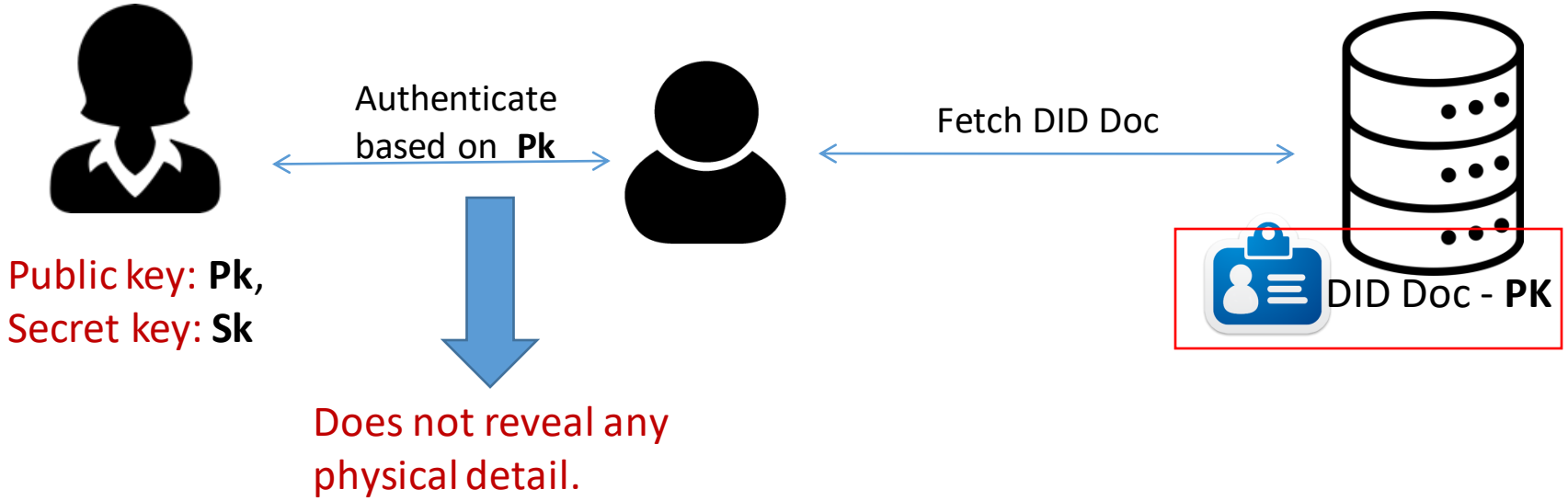
- DIDs only allow a DID controller to prove its control over its DID Document.
  - This is useful to authenticate an entity with respect to its DID



If some physical detail is presented, then that is only self attested by the DID controller, and not any verified information.

Binding DID to Physical Identity

- DIDs only allow a DID controller to prove its control over its DID Document.
  - This is useful to authenticate an entity with respect to its DID



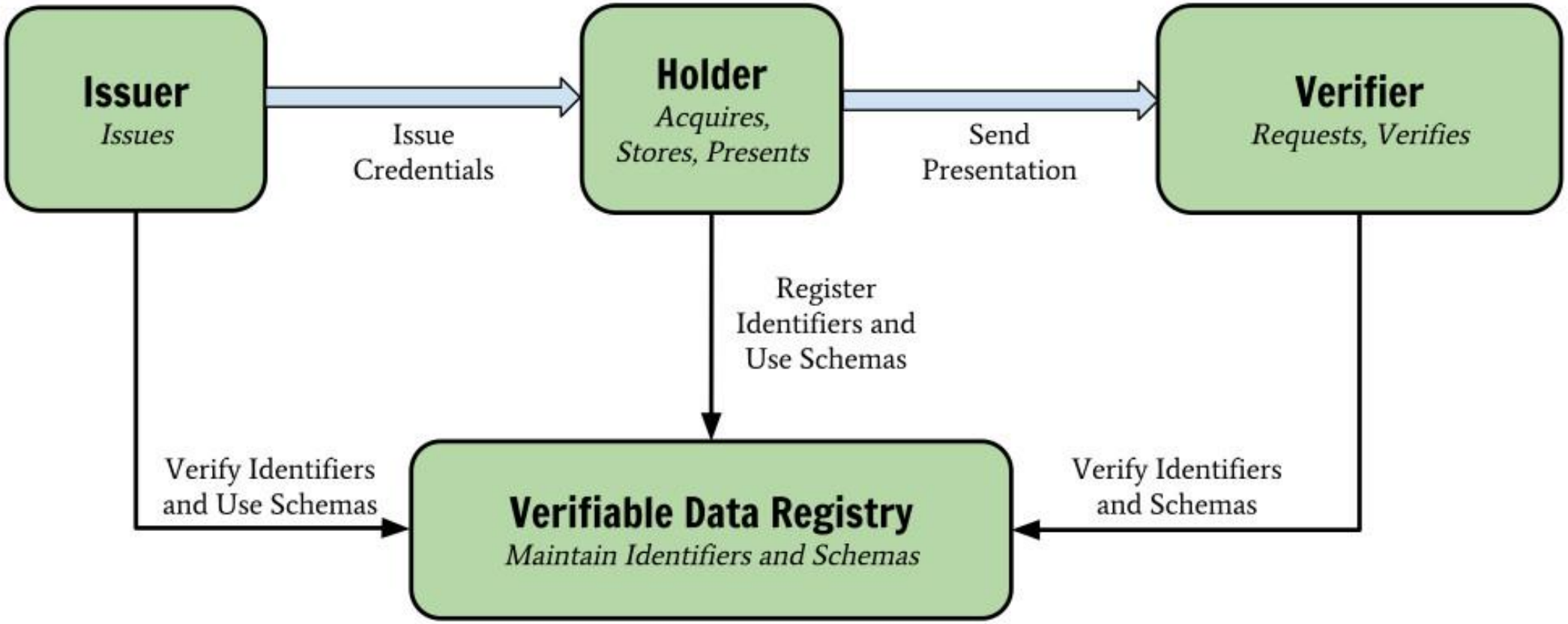
DID are not inherently tied to any physical identity (real world identity).

Verifiable Credentials

- **Verifiable Credentials Data Model** – W3C Recommendation
- Digital Representation of Credentials
  - Driver's licenses - assert that capability of operating a motor vehicle
  - University degrees - assert our level of education
  - Government-issued passports - permit to travel between countries
  - Identity – Birth Certificate, Citizenship Certificate, etc.
- **Decouples Issuer, Holder and Verifier**
- **Cryptographically secure**
- **Privacy respecting**
- **Machine-verifiable**

https://www.w3.org/TR/vc-data-model/

VC Data Model Components

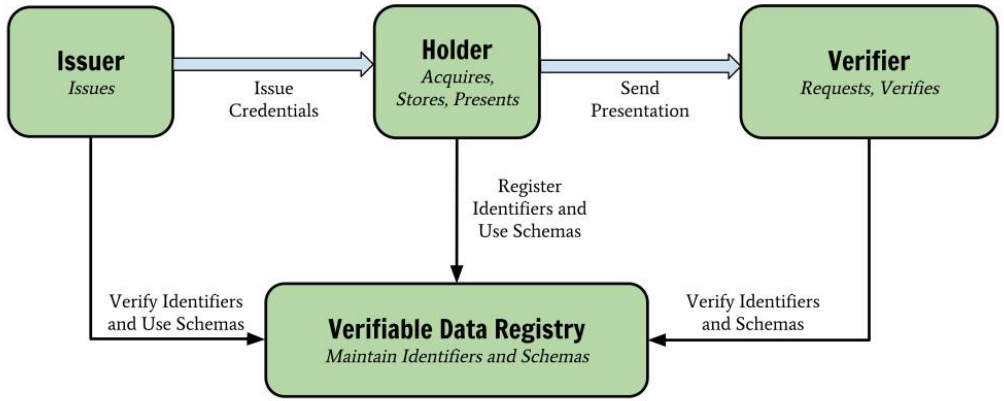


https://www.w3.org/TR/vc-data-model/

VC Data Model Components

Holder - possesses one or more VC and generating **verifiable presentations** from them. Example holders include students, employees, and customers.

Issuer –Asserts claims (in physical world) about one or more subjects, creating a VC from these claims, and transmitting the VC to a holder. Example issuers include universities, governments, etc.



Subject - Entity about which claims are made. Example subjects include human beings, animals, and things. Holder of a VC might not be the subject - 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). Note: some credentials might even be self-certified by the subject

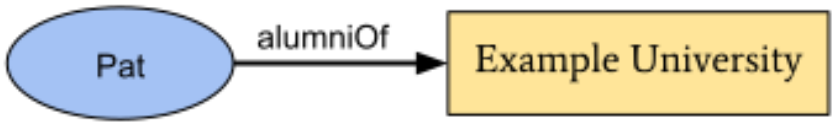
Verifier – Receives verifiable presentation to assert claims about subject. Example verifiers include employers, security personnel, and websites.

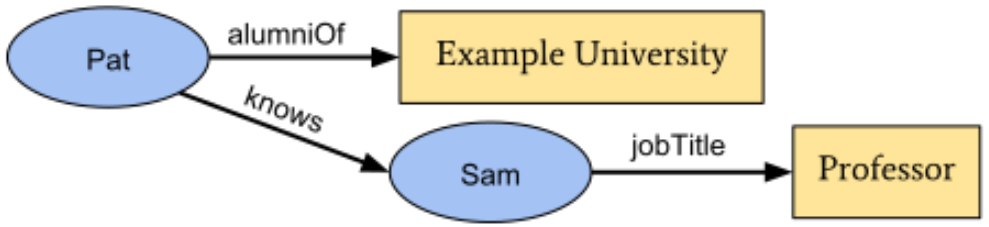
Verifiable data registry - System for creation and verification of DID, keys, and other relevant data, such as VC schemas, revocation registries, issuer public keys, and so on.

https://www.w3.org/TR/vc-data-model/

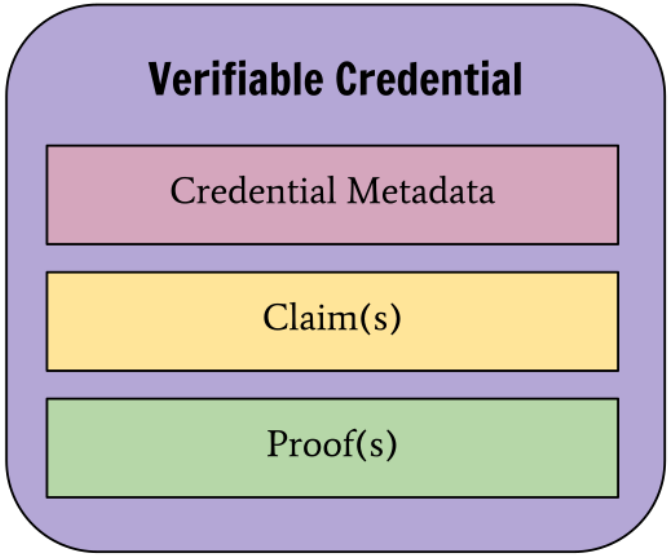
VC Data

Claims





A claim is a statement about a subject. Here Pat and Sam are subjects.

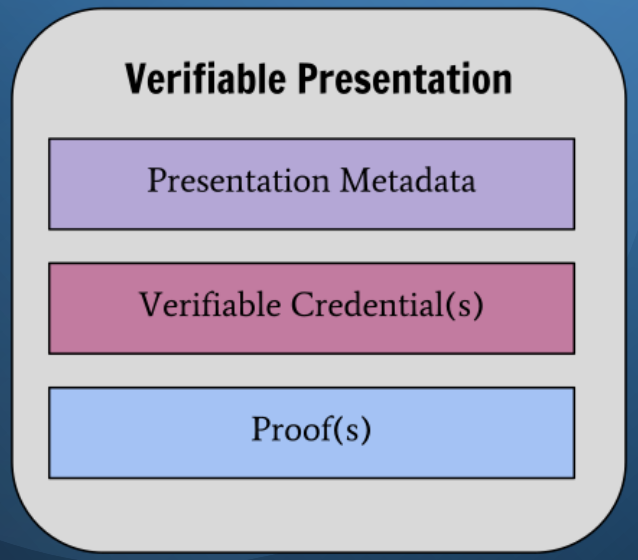


A credential is a set of one or more claims made by the same entity.

Proof is usually signature by the issuer

https://www.w3.org/TR/vc-data-model/

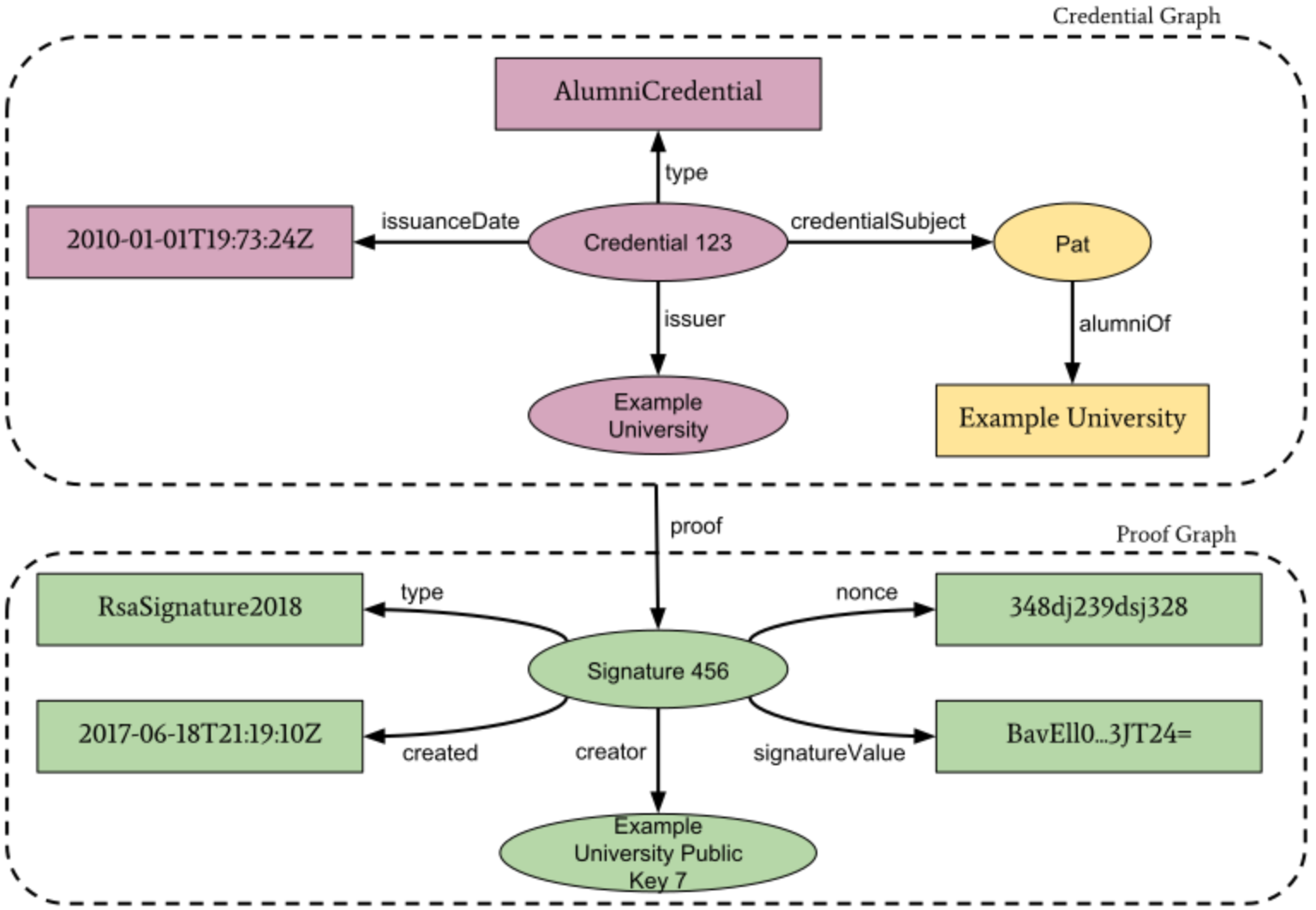
Information graph of Verifiable Presentation



A verifiable presentation expresses data from one or more VCs, and is packaged in such a way that the authorship of the data is verifiable. Holder has to convince that indeed the VC was issued to him

https://www.w3.org/TR/vc-data-model/

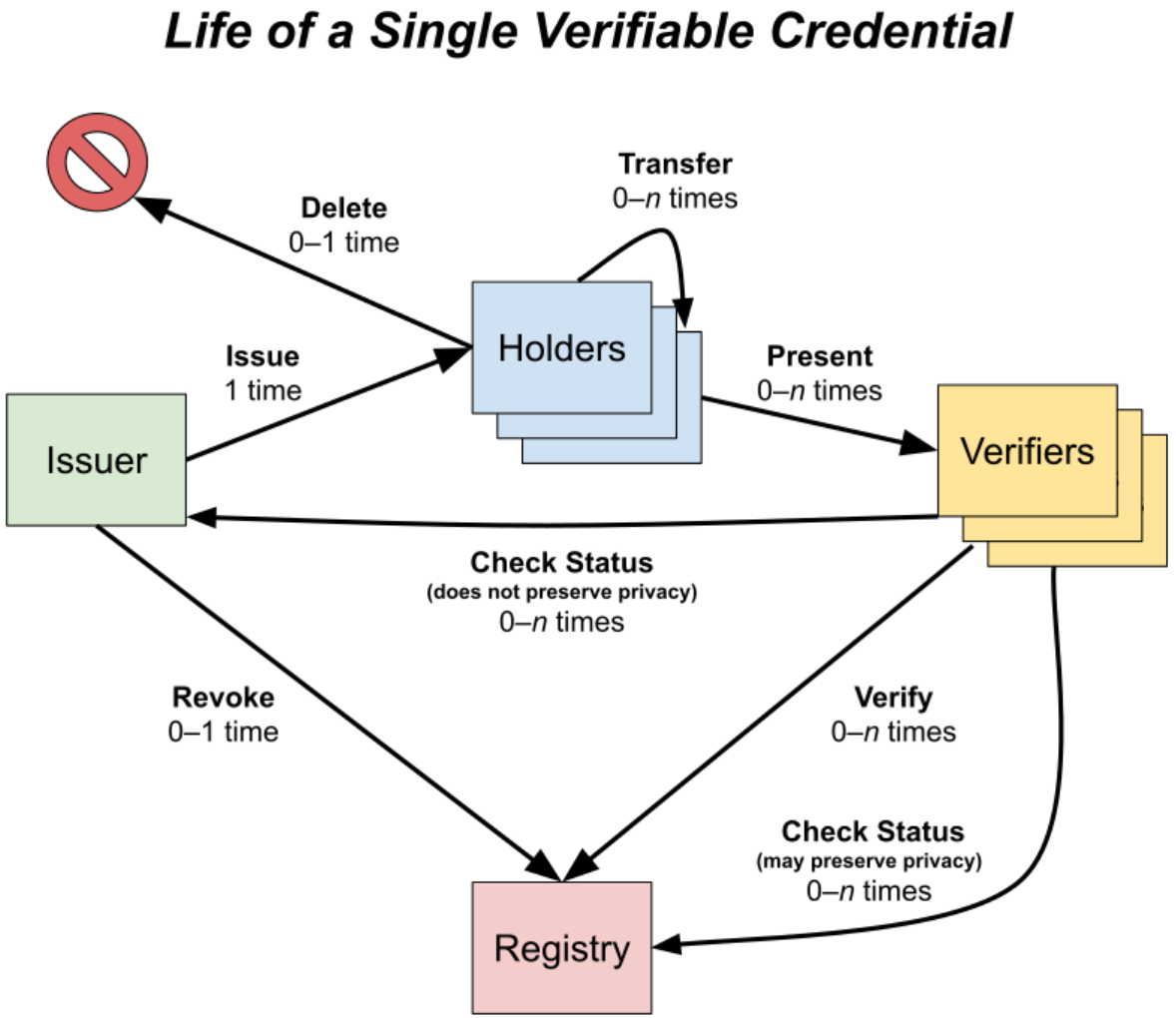
Information graph of a basic Verifiable Credential



These two together are effectively forming the verifiable credential for Pat

https://www.w3.org/TR/vc-data-model/

Verifiable Credentials Flow



https://www.w3.org/TR/vc-data-model/

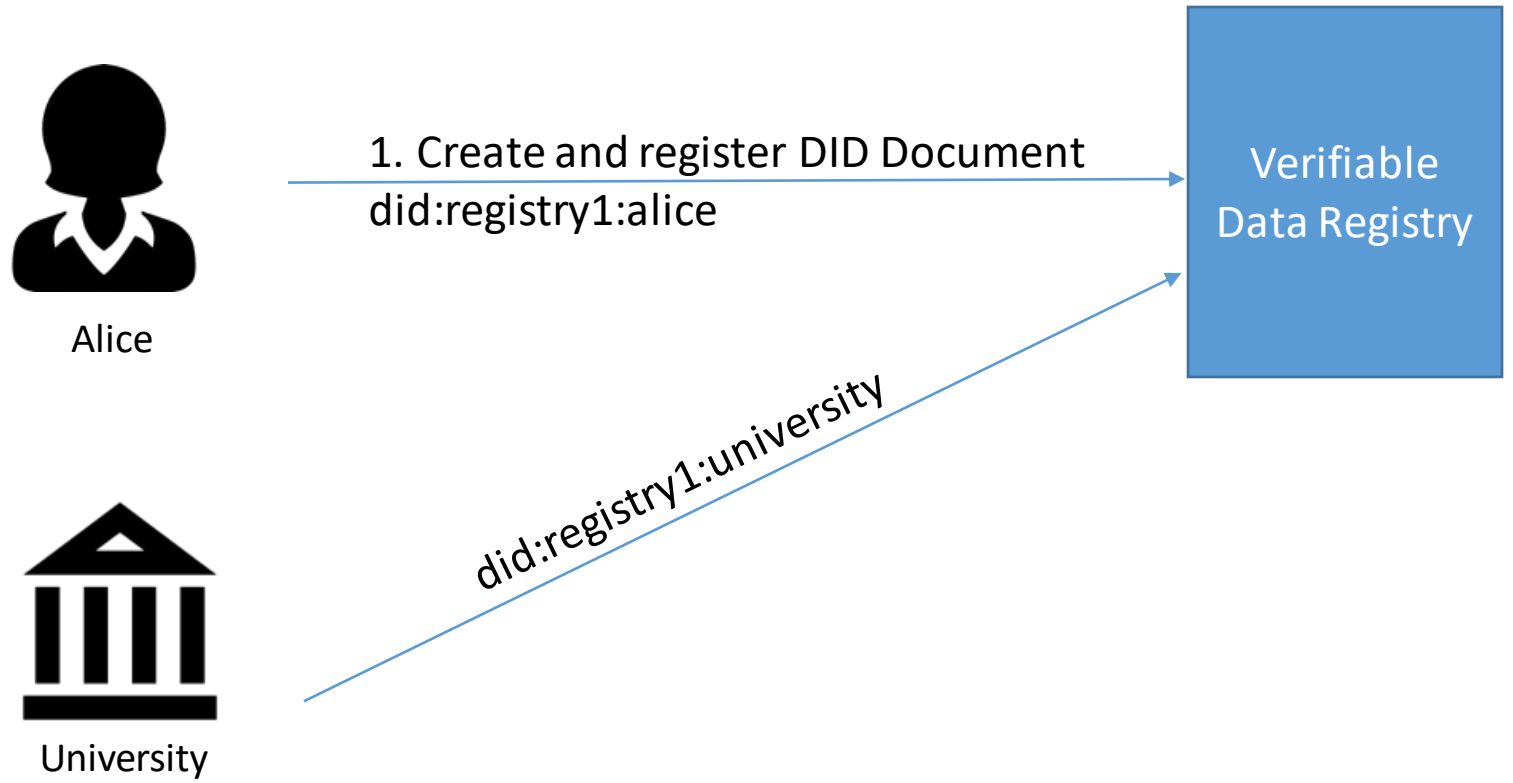
VC Trust Model

- Acting as issuer, holder, or verifier requires neither registration nor approval by any authority, as the trust involved is bilateral between parties.
- Verifier trusts the issuer to issue the VC that it received. To establish this trust, a VC is expected to either:
  - Include a proof establishing that the issuer generated the credential (signature), or
  - VC has been transmitted in a way clearly establishing that the issuer generated VC is not tampered in transit or storage.
- All entities trust the verifiable data registry to be tamper-evident and to be correct. Blockchain can help??
- The holder and verifier trust the issuer to issue true (that is, not false) credentials about the subject, and to revoke them quickly when appropriate.

https://www.w3.org/TR/vc-data-model/

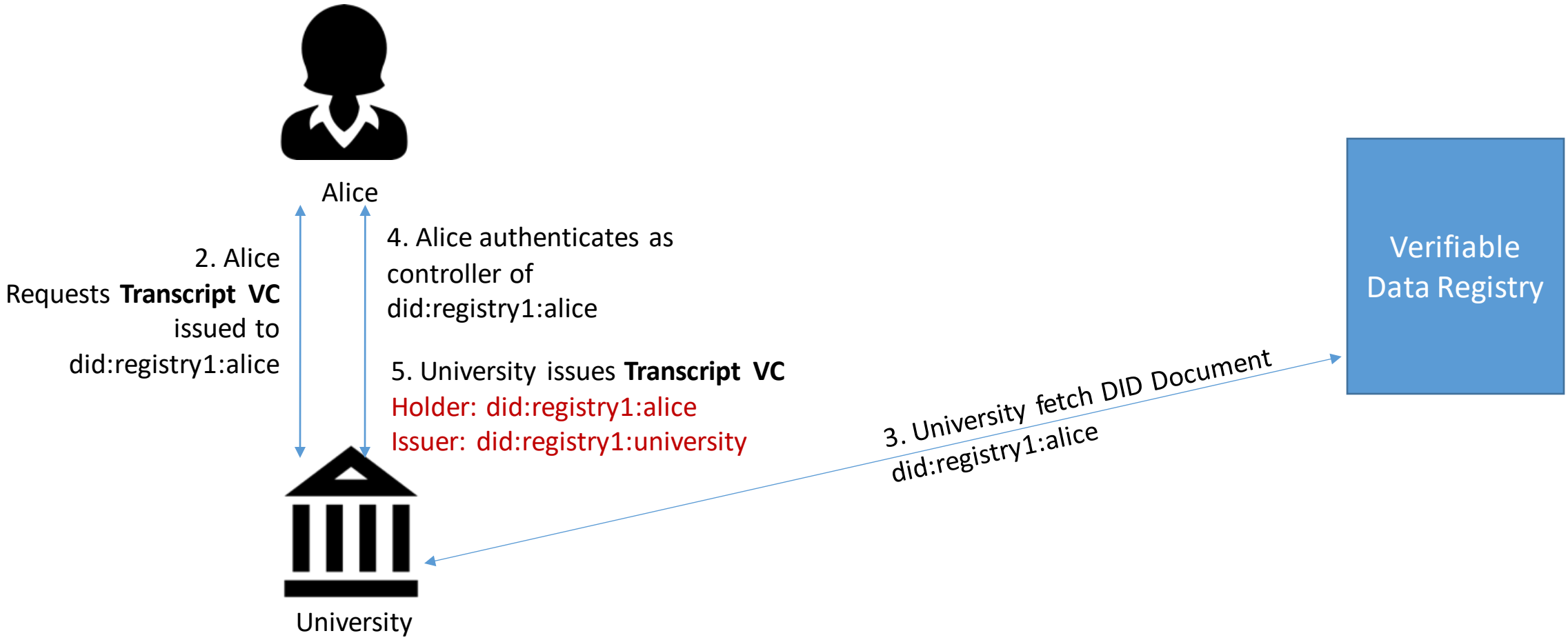
Combining DIDs and VCs

Step 1. Create and register DID



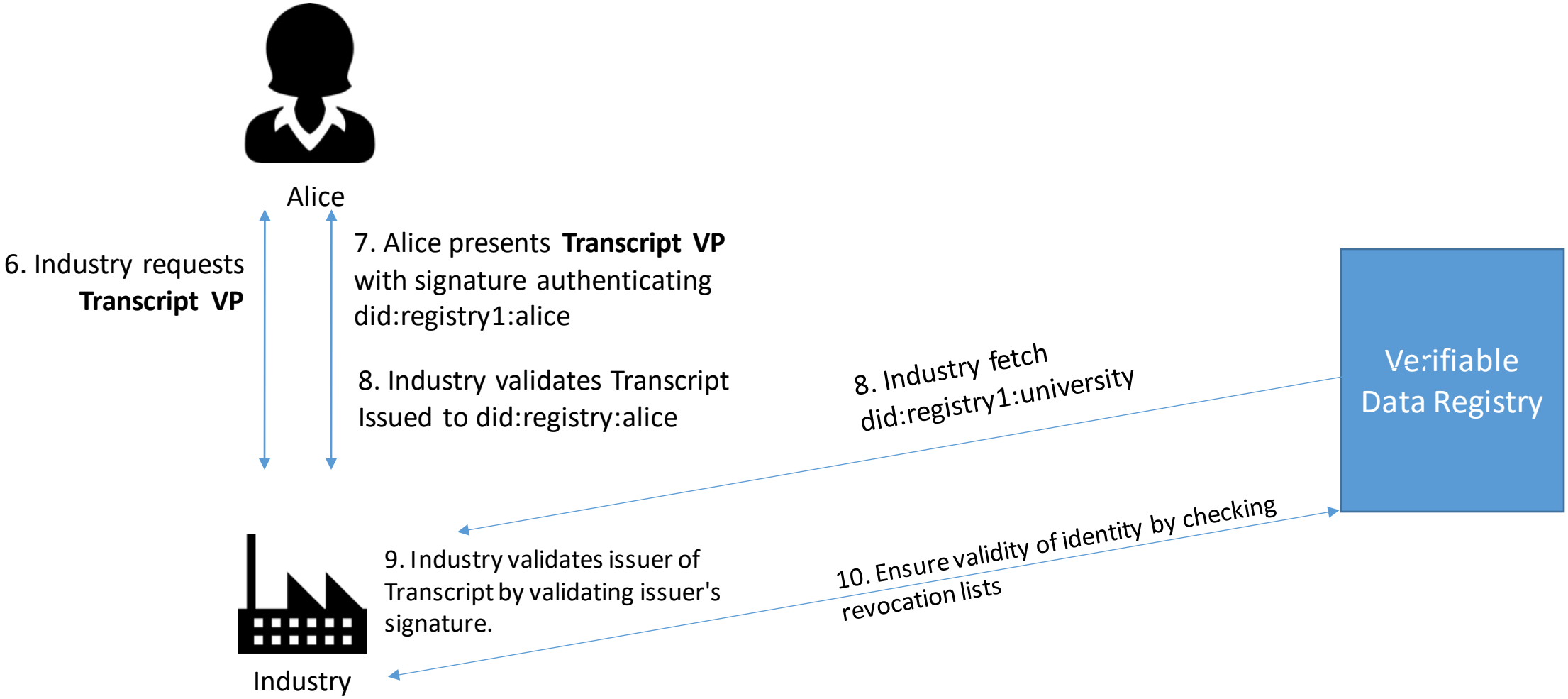
Combining DIDs and VCs

Step 2. Issue Verifiable Credential



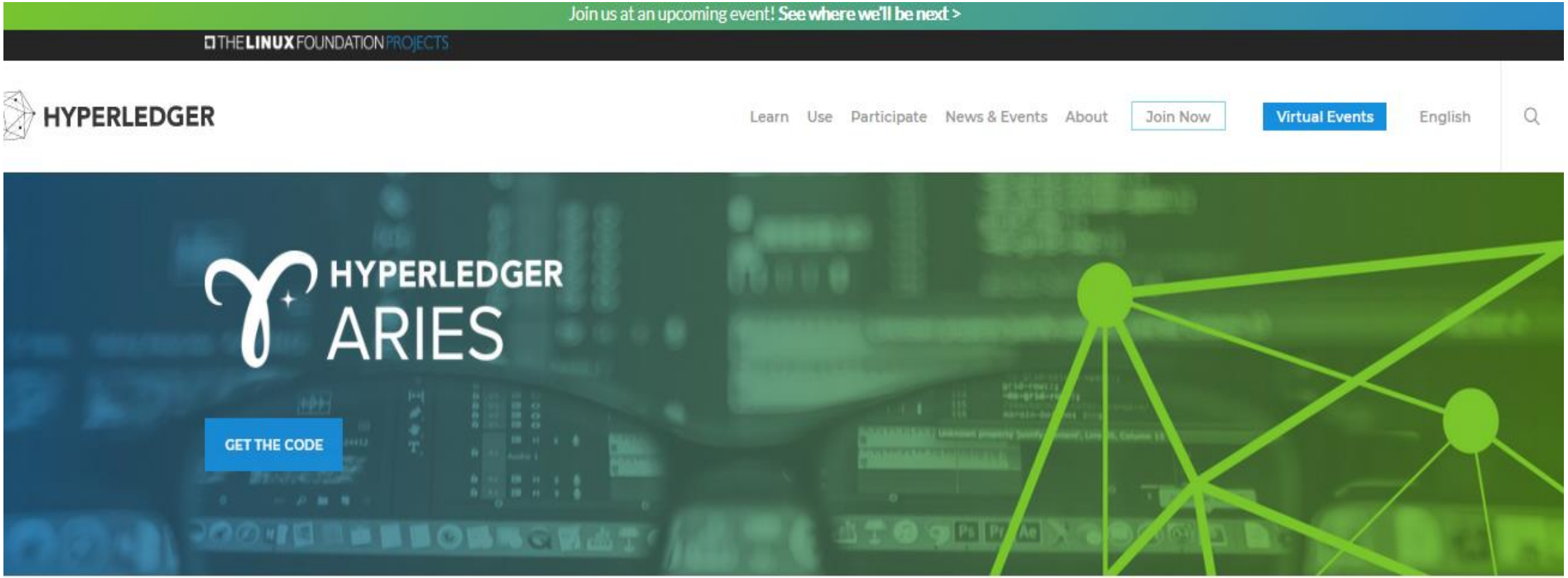
Combining DIDs and VCs

Step 3. Verifiable Presentation and Verification



Use of Blockchain for VCs

- Hyperledger Aries is meant for creating, transmitting and storing verifiable digital credentials




thank you!