Member-only story

Decentralized Identity — Owning It!

Mabel Oza · Follow
Published in Coinmonks · 14 min read · Apr 5, 2022

160 3

Decentralized identities are going to flip the script. We are now at the mercy of bureaucratic organizations like governments and large companies to manage our personal data, with decentralized identity solutions we're the boss of what happens to our personal data.

This post is going to cover the following topics:

Securing Identity 🔒

Pieces of our identity have been compromised numerous times with all the hacks and back door deals. An unfortunate and common scenario is when 3rd party identity managers sell our data or carelessly lose our data from hacks. Taking back ownership of our data would change everything, which is possible using blockchain identity solutions.

Sharing Identity 🌎

Decentralized identity projects enable people to secure identity and easily share it worldwide. With the immutable nature of blockchain, corrupt governments can't delete identities or prevent people from accessing them. An excellent example of this scenario is when Jordan refugee camps used blockchain to create identities for Syrian refugees in 2018. As people migrated to different countries throughout history, they suffered from getting access to government programs and jobs because they lacked a proper form of identification.

Inside the Jordan refugee camp that runs on blockchain

A few times a month, Bassam pushes a shopping cart through the aisles of a grocery store stocked with bags of rice, a...

www.technologyreview.com

How are decentralized identities defined?

Our identity evolves from different events in our life like education, career, residence, marital status, etc. In blockchain identity solutions, we break up those individual events and allow various parties to issue a verification that those events happened. For instance, your school can verify that you got your degree but can't verify your current residence.

Verifiable Credentials (VC)

Different identity pieces (i.e., degree, place of residence, etc.) are all referred to as verifiable credentials.

> *Verifiable credentials represent statements made by an issuer in a tamper-evident and privacy-respecting manner.*

Roles in Blockchain Identity Solutions

Today, we will play the role of a hiring manager (Bob), and we need to verify if Alice got a Ph.D. at the University of Chicago or if she's embellishing her resume.

Holder: Credential manager and owner. This is the only person/organization authorized to share their credentials and select what piece of their identity they're willing to share. *In this scenario, Alice is the holder.*

- Subject: is the person, animal, or thing that the credentials are about. Sometimes the holder wouldn't be the 'subject' if the credential is for a child or animal. *In this scenario, Alice is also the subject.*

Issuer: Credential creator. This is the person/organization that would create the credentials for the holders on behalf of the subject. ***Even though they make the credentials, they cannot share them with anyone besides the holder (cryptographically, it's impossible).*** *In this scenario, the Univesity of Chicago is the issuer.*

Verifier: Credential viewer. This is the person/organization that requests the credentials for verification, ***they will only be presented the credentials if the holder authorizes it.*** *In this scenario, Bob, the hiring manager, is the verifier.*

Verifiable Data Registry: Credential validator. This is either a government database or in our scenario a blockchain network. Every verification credential is registered into the blockchain with a DID (Decentralized Identifier).

How do Verifications Work?

- Bob asks Alice for proof that she attended her University
- Alice creates a verifiable presentation. The verifiable presentation is like a verifiable credential but only has the attributes relevant to Bob's needs (she doesn't need to share her GPA, just the fact she attended the University).
- She shares it with Bob, and Bob can verify it on the blockchain network using the DID (Decentralized Identifier)

Setup Involved

Below are the steps required before you start verifying credentials:

1. University (issuer) sets a schema and credential definitions for their verifiable credential by picking out what attributes will be used and defining them. This is all done beforehand when the Univesity creates its issuer system.
2. University (issuer) offers the credential to Alice (holder)
3. Alice requests the credential

4. University (issuer) issues Alice (holder & subject) a verifiable credential, registers it on the blockchain network (verifiable data registry) by signing it with a public DID (Decentralized Identifier)

DID (Decentralized Identifiers)

DIDs are fundamental building blocks of SSI (Self Sovereign Identity) systems; they are unique identity identifiers (URIs). DIDs allow people/organizations to prove pieces of their identity using cryptographic proofs like digital signatures. All DIDs are held in a crypto wallet like that box at home containing the birth certificate, marriage certificates, scuba diving certificate, etc.

Below is the syntax of a DID; it comprises a scheme, method, and method-specific identifier.

DID Method

The method defines the mechanisms for creating, resolving, updating, and deactivating DIDs and DID documents using a specific type of verifiable data registry (blockchain network). In a blockchain, this usually refers to the blockchain network name, like the ones below:

A DID on the Indy Blockchain Network Sovrin MainNet Ledger:

```
did:indy:sovrin:7Tqg6BwSSWapxgUDm9KKgg
```

A DID on the Solana Blockchain Devnet Ledger:

```
did:sol:devnet:6Na3uiqyRGZZQdd19RLCb6kJHR51reFdhXzAuc6Y8Yef
```

Check out the docs on all the DID methods:

(DID) the Decentralized Identifier

Decentralized Identifiers (DIDs) v1.0 Decentralized identifiers (DIDs) are a new type of identifier that enables...

decentralized-id.com

DID Architecture

DIDs have a particular format so they can connect to different pieces in an identity solution. Below is DID architecture laid out by W3:

We are introduced to two new components, the DID controller and DID document.

DID Document

If you think of DIDs as a key-value database, DIDs would be the key and DID Documents would be the value. To get from the DID to the DID document, we use DID resolution.

Image from https://academy.affinidi.com/faqs-on-verifiable-credentials-and-self-sovereign-identity-3c3a505878bc

The DID document describes the public keys, authentication protocols, and service endpoints needed to verify an identity cryptographically.

Below is a DID Document with one verification method type:

```
{
    "@context": [
      "https://www.w3.org/ns/did/v1",
      "https://w3id.org/security/suites/ed25519-2020/v1"
    ],
    "id": "did:example:123",
    "authentication": [
      {
        "id":
"did:example:123#z6MkecaLyHuYWkayBDLw5ihndj3T1m6zKTGqau3A51G7RBf3",
        "type": "Ed25519VerificationKey2020", // external (property
value)
        "controller": "did:example:123",
        "publicKeyMultibase":
"zAKJP3f7BD6W4iWEQ9jwndVTCBq8ua2Utt8EEjJ6Vxsf"
      }
    ],
    "capabilityInvocation": [
      {
        "id":
"did:example:123#z6MkhdmzFu659ZJ4XKj31vtEDmjvsi5yDZG5L7Caz63oP39k",
        "type": "Ed25519VerificationKey2020", // external (property
value)
        "controller": "did:example:123",
        "publicKeyMultibase":
"z4BWwfeqdp1obQptLLMvPNgBw48p7og1ie6Hf9p5nTpNN"
      }
    ],
    "capabilityDelegation": [
      {
        "id":
"did:example:123#z6Mkw94ByR26zMSkNdCUi6FNRsWnc2DFEeDXyBGJ5KTzSWyi",
        "type": "Ed25519VerificationKey2020", // external (property
value)
        "controller": "did:example:123",
        "publicKeyMultibase":
"zHgo9PAmfeoxHG8Mn2XHXamxnnSwPpkyBHAMNF3VyXJCL"
      }
    ],
    "assertionMethod": [
      {
        "id":
"did:example:123#z6MkiukuAuQAE8ozxvmahnQGzApvtW7KT5XXKfojjwbdEomY",
        "type": "Ed25519VerificationKey2020", // external (property
value)
        "controller": "did:example:123",
        "publicKeyMultibase":
"z5TVraf9itbKXrRvt2DSS95Gw4vqU3CHAdetoufdcKazA"
      }
    ]
}
```

DID Controller

This is the entity (person, org, or software) that can change the DID document, and sometimes, this can be the holder and/or subject.

Data Formats of Verifiable Credentials and Presentations

Verifiable Credentials are usually in JSON like (JSON-LD) format, like the one below.

Alice's VC (Verifiable Credential) for her University

```
{
  // set the context, which establishes the special terms we will be
using
  // such as 'issuer' and 'alumniOf'.
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://www.w3.org/2018/credentials/examples/v1"
  ],
  // specify the identifier for the credential
  "id": "http://https://www.uchicago.edu/credentials/1872",
  // the credential types, which declare what data to expect in the
credential
  "type": ["VerifiableCredential", "AlumniCredential"],
  // the entity that issued the credential
  "issuer": "http://https://www.uchicago.edu/issuers/565049",
  // when the credential was issued
  "issuanceDate": "2010-01-01T19:23:24Z",
  // 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": "University of Chicago PhD",
        "lang": "en"
      }]
    }
  },
  // digital proof that makes the credential tamper-evident
  // see the NOTE at end of this section for more detail
  "proof": {
    // the cryptographic signature suite that was used to generate
the signature
    "type": "RsaSignature2018",
    // the date the signature was created
    "created": "2017-06-18T21:19:10Z",
    // purpose of this proof
    "proofPurpose": "assertionMethod",
    // the identifier of the public key that can verify the signature
    "verificationMethod":
"https://example.edu/issuers/565049/keys/1",
    // the digital signature value
    "jws":
"eyJhbGciOiJSUzI1NiIsImI2NCI6ZmFsc2UsImNyaXQiOlsiYjY0Il19..TCYt5X
      sITJX1CxPCT8yAV-TVkIEq_PbChOMqsLfRoPsnsgw5WEuts01mq-
pQy7UJiN5mgRxD-WUc
      X16dUEMGlv50aqzpqh4Qktb3rk-BuQy72IFLOqV0G_zS245-
kronKb78cPN25DGlcTwLtj
      PAYuNzVBAh4vGHSrQyHUdBBPM"
  }
}
```

Alice's VP (Verifiable Presentation) for her Hiring Manager

```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://www.w3.org/2018/credentials/examples/v1"
  ],
  "type": "VerifiablePresentation",
  // the verifiable credential issued in the previous example
  "verifiableCredential": [{
    "@context": [
      "https://www.w3.org/2018/credentials/v1",
      "https://www.w3.org/2018/credentials/examples/v1"
    ],
    "id": "http://https://www.uchicago.edu/credentials/1872",
    "type": ["VerifiableCredential", "AlumniCredential"],
    "issuer": "http://https://www.uchicago.edu/issuers/565049",
    "issuanceDate": "2010-01-01T19:23:24Z",
    "credentialSubject": {
      "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
      "alumniOf": {
        "id": "did:example:c276e12ec21ebfeb1f712ebc6f1",
        "name": [{
        "value": "University of Chicago PhD",
        "lang": "en"
        }]
      }
    },
    "proof": {
      "type": "RsaSignature2018",
      "created": "2017-06-18T21:19:10Z",
      "proofPurpose": "assertionMethod",
      "verificationMethod":
"https://www.uchicago.edu/issuers/565049/keys/1",
      "jws":
"eyJhbGciOiJSUzI1NiIsImI2NCI6ZmFsc2UsImNyaXQiOlsiYjY0Il19..TCYt5X
        sITJX1CxPCT8yAV-TVkIEq_PbChOMqsLfRoPsnsgw5WEuts01mq-
pQy7UJiN5mgRxD-WUc
```

```
        X16dUEMGlv50aqzpqh4Qktb3rk-BuQy72IFLOqV0G_zS245-
kronKb78cPN25DGlcTwLtj
        PAYuNzVBAh4vGHSrQyHUdBBPM"
    }
  }],
  // digital signature by Pat on the presentation
  // protects against replay attacks
  "proof": {
    "type": "RsaSignature2018",
    "created": "2018-09-14T21:19:10Z",
    "proofPurpose": "authentication",
    "verificationMethod":
"did:example:ebfeb1f712ebc6f1c276e12ec21#keys-1",
    // 'challenge' and 'domain' protect against replay attacks
    "challenge": "1f44d55f-f161-4938-a659-f8026467f126",
    "domain": "4jt78h47fh47",
    "jws":
"eyJhbGciOiJSUzI1NiIsImI2NCI6ZmFsc2UsImNyaXQiOlsiYjY0Il19..kTCYt5
       XsITJX1CxPCT8yAV-TVIw5WEuts01mq-
pQy7UJiN5mgREEMGlv50aqzpqh4Qq_PbChOMqs
       LfRoPsnsgxD-WUcX16dUOqV0G_zS245-kronKb78cPktb3rk-
BuQy72IFLN25DYuNzVBAh
       4vGHSrQyHUGlcTwLtjPAnKb78"
  }
}
```

If you would like to review the official W3C Recommendation on the Verifiable Credential Data Model, check out the link below:

Verifiable Credentials Data Model v1.1

Credentials are a part of our daily lives; driver's licenses are used to assert that we are capable of operating a…

www.w3.org

There are several data formats for credentials like JSON-JWT, JSON-LD with LD Signatures, ZKP-CL, and JSON-LD ZKP with BBS+.

Revocation

How do we revoke a credential if it's false or no longer valid? This happens when organizations make mistakes, or specific credentials have expirations (i.e., visas). With the immutability of a blockchain network, correcting these mistakes can be pretty tricky.

When a credential changes or we need to revoke it, we don't remove it; instead, we invalidate it. Since there are invalid DID's out there; we need to cross-check with a **revocation registry** in every proof.

Revocation Registry

The revocation registry was built to fulfill the following requirements:

- Credentials need to be revocable by the issuer

- Proving a credential is still valid should not involve the issuer

What is the Revocation Registry? It's a combination of accumulators and tail files

Accumulator

It's math that's baked into the process of issuing credentials. It uses a cryptographic accumulator; basically, it's a product from multiplying multiple numbers. So in a scenario where we have an equation a * b * c * d = e, e is the accumulator and a,b,c, and d are prime factors randomly generated that are usually large, like in the 100,000's or 10 million range. The accumulator can't be reversed by prime factorization because it uses modular arithmetic, where division is undefined.

Tail Files

All the factors are found in the tail file, they're listed out in an array, and the file is binary. Each credential issued is assigned an index of an accumulator factor is in a tail file. If a credential were revoked, it wouldn't contribute to the accumulator.

So if Alice got divorced and her marriage credential is revoked, we wouldn't include index 0 in the accumulator product.

Below is how the tail file contributes to the accumulator.

Below is what happens when we revoke a credential, the accumulator changes.

Now that Alice's (holder) accumulator isn't going to match the accumulator found in Bob's (verifier) proof, we know the credential is invalid and revoked.

Revocations at a High Level

We are just checking to see if the issuer and holder can get the same answer to this math problem represented in the proof. If it's not equal, then the credential is invalid. No one is running these calculations on the fly. The issuer runs this calculation when registering a revocation to the registry, and the holder has the answer to this calculation when they were issued the credential.

Check out Indy's detailed explanation of the revocation process below:

indy-hipe/text/0011-cred-revocation at master · hyperledger/indy-hipe

Author: Daniel Hardman Start Date: 2018–02–01 (approximate, backdated) Status: ADOPTED Status Date: (date of first...

github.com

What about Privacy? Zero-Knowledge Proofs (ZKPs)

What if Bob (Alice's hiring manager) wanted to verify her residence, but he couldn't know the address. This is where zero-knowledge proofs (ZKPs) come into play, it's a probabilistic way to prove that something is true without revealing additional information. **It solves the scenario where we don't trust the verifier. The idea is that you are convincing someone you know a fact without ever revealing what that fact is.**

Below is the best explainer of Zero-Knowledge Proofs by Amit Sahai from UCLA.

Computer Scientist Explains One Concept in 5 Levels of Difficulty |...
Share
Watch on YouTube

ZKPs have been around for a while, they were invented by Shafi Goldwasser, Silvio Micali, and Charles Rackoff in 1985. Recently there's been more applications with ZKPs, especially in the blockchain world.

With ZKPs you need to decide what you need to prove. To approach the problem you use NP-complete problems like the three colored-graph problem.

Zero-Knowlege Proof Example — Land of the 3 Monsters

Let's say we have a land with three types of monsters, and if the same kind of monster is close to each other, they will grow FIERCE, and the land will no longer be safe.

We want to answer is **this land safe?** Without ever revealing the locations of all the monsters.

Below is the location of all of the monsters with their locations exposed.

To protect them from hunters we cover up all the monsters so no one can see where they are.

To prove that the land is safe, we challenge the visitors to pick up any two hats adjacent to each other connected by a line.

- If two of the revealed **monsters are the same color**, I know this land is unsafe, and we know 100% certainty that the **land isn't safe.**

- If the **monsters are different colors**, we **continue**. We're **still not sure** the land is safe.

As we go through more successful rounds, the probability of the land being unsafe decreases. The probability of uncertainty can be determined by ((*E*-1)/*E* * (*E*-1)/*E,* where E is the number of edges in the graph. So after the first round of not finding adjacent monsters, (12–1)/12 * (12–1)/12, we're 84% uncertain, and if we haven't found adjacent monsters after checking 11 edges, we're 0.69% uncertain.

After every round, the monsters **shuffle** their locations while still staying away from their type. The visitor continues the previous step, where they reveal two monster locations adjacent to each other; this continues until the probability is low enough (negligible). The shuffling of locations after each round makes it difficult for the visitor to figure out the locations of the monsters between rounds.

There's much more to Zero-Knowledge Proofs and I recommend you check out the resources below to dive into it.

Zero Knowledge Proofs: An illustrated primer
One of the best things about modern cryptography is the beautiful terminology. You could start any number of punk bands...
blog.cryptographyengineering.com

Awesome Zero-Knowledge Proof Github

GitHub - matter-labs/awesome-zero-knowledge-proofs: A curated list of awesome things related to...
A curated list of awesome things related to learning zero-knowledge proofs Zero-Knowledge Proofs Starter Pack...
github.com

Popular ZKP protocols are SNARK (Succinct Non-Interactive Arguments of Knowledge), STARK (Scalable Transparent Argument of Knowledge), and Bulletproofs. A rough comparison between the three can be summarized in the table below:

Image from https://github.com/matter-labs/awesome-zero-knowledge-proofs

Time for some Action! 🥷

You can use several frameworks to get started with identity projects like Indy & Aries, Serto, Veramo, SpruceID, and Polygon ID (I haven't found the dev docs yet). To get started, I will use Veramo; from all the docs I came across, Veramo seems the simplest for beginners who want to experiment quickly.

Download the Veramo CLI — Only Supported by Linux and macOS

Install the Veramo CLI

```
npm i @veramo/cli -g
```

You can find all the veramo methods with the command

```
veramo --help
```

Before you begin, you need to create an agent. You can either connect to a hosted veramo instance or create a veramo instance locally.

Creating Agent

The agent is the entry point to managing identity. It's a standard interface to operate and orchestrate core and custom identity plugins. The agent takes care of low-level details like handling ether-did (Ethereum address), web-did (DNS domain), and did-key (simple public/private key pair).

Image from https://medium.com/uport/introducing-veramo-5a960bf2a5fe

Connecting Agent Local Instance

Run the command below to create a local veramo instance. By default, the command will produce a SQL lite database in the same folder.

```
veramo config create
```

Connecting Agent to Hosted Instance

Run the command below to connect to a hosted veramo instance.

```
veramo config create --template client
```

Below are the different YAML files created for **local instances (left)** and **hosted instances (right).** You'll notice that the local instance has to handle the configuration of SQLite while the hosted instance doesn't.

Left is the local instance YAML and the right is the hosted instance YAML

Create a DID

Now let's create our first DID by running the command below, selecting the identifier provider, key management system, and entering an alias.

```
veramo did create
```

Create a Verifiable Credential for One of the DIDs You Created

Create a credential by running the command below and filling out the details required for the credential.

```
veramo credential
```

You can also view the credential through the explorer by running the command below and selecting the verifiable credential (VC) you just created.

```
veramo explore
```

Create Verifiable Presentation (VP)

Now that you created your Verifiable Credentials (VC), you must share them. To share your VC(s), you need to create a Verifiable Presentation (VP). You need to run the command below and select which VCs you want to share to make a VP.

```
veramo presentation
```

Notice the two employments for Heisenberg, teacher and chemist, in his Verifiable Presentation (VP).

We just hit the tip of the iceberg; there's much more you can do with Veramo. I urge you to check out the official documentation.

Introduction | Performant and modular APIs for Verifiable Data and SSI

Veramo is in a public beta. There will be some breaking changes in the coming months, and we endeavor to communicate...

veramo.io

If you like the post, then you can buy me coffee, thank you in advance!

Have fun building!

> Join Coinmonks Telegram Channel and Youtube Channel learn about crypto trading and investing

Also, Read

- Uphold Card Review | Trust Wallet vs MetaMask
- Exness Review | MoonXBT Vs Bitget Vs Bingbon
- How to Start Earning Passive Income With Crypto Lending
- Cryptocurrency Savings Accounts | Crypto Trading Bots
- BigONE Exchange Review | CEX.IO Review | Swapzone Review

Ssi Did Decentralized Identity Self Sovereign Identity Blockchain Technology

160 3

Written by Mabel Oza

Follow

147 Followers · Writer for Coinmonks

Blockchain Engineer @Bakkt . My opinions do not reflect my company. Making the financial world more secure, accessible, and transparent.

More from Mabel Oza and Coinmonks

Mabel Oza in Coinmonks

Securing Keys with HSMs (Hardware Secure Module)

HSMs (Hardware Secure Modules) are specialized hardware devices that are tamp...

12 min read · Aug 29, 2022

102 3

Crypto with Roberto in Coinmonks

3 AI Crypto Tokens That Could Do the 100x in 2023

Artificial Intelligence (AI) is revolutionizing the world, and it's no secret that the...

· 3 min read · Feb 6

2.7K 20

Gaurav Agrawal in Coinmonks

10 Best FREE Crypto Trading Bots in 2023

Best crypto trading bots for Binance, Coinbase, Kucoin, and other crypto...

23 min read · May 2

2.2K 57

Mabel Oza in Coinmonks

B.O.L.T. Standards

B.O.L.T. stands for Basis of Lightning Technology (Lightning Network...

9 min read · May 24

12

See all from Mabel Oza

See all from Coinmonks

Recommended from Medium

Gianmarco Guazzo in Coinmonks

Ethereum Block Limit & Gas Usage

Did you know that Bitcoin and Ethereum have the block limit specification calculated…

· 2 min read · Jan 4

22 1

Jay Prakash in Silence Laboratories

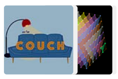
A Compute Perspective of MPC-TSS: Paillier in ECDSA Revisited

Yashvanth, Jay, and members at Silence Laboratories

8 min read · Feb 7

45

Lists



Medium Publications Accepting Story Submissions

145 stories · 158 saves

The PyCoach in Artificial Corner

You're Using ChatGPT Wrong! Here's How to Be Ahead of 99% of…

Master ChatGPT by learning prompt engineering.

· 7 min read · Mar 17

25K 437

Dana J. Wright in UX Collective

How to make decentralized apps (dApps) more user-friendly

I feel for Celsius, Three Arrows, Babel Finance and whoever the next floater happens to be.

· 7 min read · Jul 8, 2022

444 2

Prof Bill Buchana... in ASecuritySite: When Bob M...

Schnorr Zero Knowledge Proofs With Elliptic Cr

In Feb 1989, Claus Peter Schnorr submitted a patent which was assigned to no one. It had ...

· 5 min read · Jan 20

76

Ramsès Fernàndez-València in Innovation Stories

Homomorphic Signatures

Approaching the topic and the main proposals

11 min read · Sep 21, 2022

122

See more recommendations