IDENTIFYING AND TRACKING PHYSICAL OBJECTS WITH HYPERLEDGER DECENTRALIZED APPLICATIONS

David Chicano[1], Matevž Pustišek[2]

[1] *Universitat Politécnica de Catalunya, ETSETB, c. Jordi Girona 1-3.08034 Barcelona. Spain*
[2] *University of Ljubljana,Faculty of Electrical Engineering, Tržaška 25, 1000 Ljubljana, Slovenia*
Email: david.chicano.valenzuela@gmail.com

IDENTIFYING AND TRACKING PHYSICAL OBJECTS WITH HYPERLEDGER DECENTRALIZED APPLICATIONS

Abstract. *The passing of time has made clear the trend of decentralization. Distributed ledger technologies like blockchain have brought a full range of new possibilities to improve, in this case, trust and identity management on the internet as Self Sovereign Identity (SSI) does. This thesis has analyzed the blockchain impact on IoT with the Linux Foundation Project called Hyperledger, concretely, with its Identity Stack solution. Even though it is a really unmatured project, it has some useful tools to start understanding and testing the capabilities of this technology. The project consists of a practical scenario simulation of communication and verified credentials sending between a Raspberry Pi with an RFID sensor, which will be tracking an object's state of delivery, and a graphical Java Application. Everything through the Hyperledger SSI Stack, formed by a public Indy network instance and Aries agents.*

1 Introduction

The purpose of this project is to analyze decentralized applications for IoT based on Hyperledger technologies. In particular, it will explore and demonstrate options for identifying and tracking physical objects with Hyperledger decentralized applications.
The project is based on the Self Sovereign Identity concept which is built in a blockchain-based ledger. There are many different implementations, but this project will be using a solution from the Hyperledger Foundation called the Identity Stack.

1.1 Blockchain

We are not interested in a deep explanation of blockchain, since we will be using an already created/configured network and its interactions will be automated with the Hyperledger software.
A blockchain is a growing list of data blocks linked together, a peer-to-peer distributed ledger forged by consensus, combined with a system for "smart contracts" and other assistive technologies. Together these can be used to build a new generation of transactional applications that establishes trust, accountability, and transparency at their core, while streamlining business processes and legal constraints. All without middlemen (so-called trusted third parties).
Each block is timestamped, with each new block referring to the previous block. Combined with cryptographic hashes, this timestamped chain of blocks provides an immutable record of all transactions in the network, from the very first (or genesis) block.
An interesting point to know about for this thesis is that we can differentiate between two types; permissioned and permissionless.

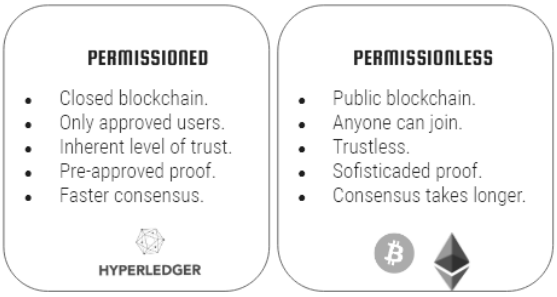

Figure 1 Blockchain: Permissioned vs. Permissionless

1.2 Self Sovereign Identity (SSI)

Self Sovereign Identity (SSI) is the idea that you control your own data, you control when and how it is provided to others, and when it is shared, it is done so in a trusted way. With SSI, there is no central authority holding your data that passes it on to others upon request. And because of the underlying cryptography and blockchain technology, SSI means that you can present claims about your identity and others can verify it with cryptographic certainty.
In SSI, entities are identified by decentralized identifiers (DID). The trust and management of DIDs are assured by blockchain technology, and DIDs can be decoupled from centralized registries, identity providers, and certificate authorities. Like the URLs that we are familiar with, DIDs can be resolved (often resolved by reading from a blockchain). When we pass a valid DID to a piece of software called a DID Resolver, it works like a browser given a URL, resolving the DID and returning a document. However, instead of returning a web page, a DID Resolver returns a DID Document

(DIDDoc), a JSON document whose format is defined in the DID specification. A DIDDoc contains (usually) public keys whose private keys are held by the entity that controls the DID, and (usually) service endpoints that enable communication with that entity.

Figure 2 DID structure example

With this comes the use of verifiable credentials. A credential is (formally) an attestation of qualification, competence, or authority issued to an entity by a third party with a relevant or de facto authority or assumed competence to do so. For identity, verifiable credentials are digital, cryptographically-protected data from authorities that you can use to prove you are you.
In all uses of a verifiable credential, what is issued is a credential. However, in some implementations, when a credential is presented, the credential is proven, while in others (including in Hyperledger Indy), the claims within the credential are proven individually. Your verifiable credentials are issued to you, stored in your digital wallet, and you decide when and where you want to use them. Verifiable presentation data is proven without needing to call back to the issuer.
Another pair of terms that might seem to be used interchangeably are proof and presentation. Proof is evidence of the claim.
The same framework can be adapted for non-person subjects as IoT.

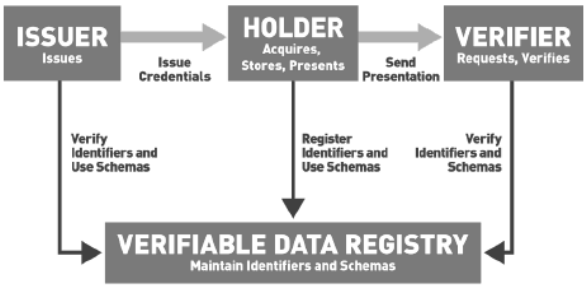

Figure 3 SSI flow diagram

1.3 Hyperledger Identity Stack

"The Hyperledger Foundation is a global ecosystem for enterprise blockchain technologies. As part of the Linux Foundation, it is a neutral home for developers to collaborate, contribute, and maintain open-source software." [1]
The main strength of Hyperledger is having a specific solution for each necessity. In this case, the Identity Stack creates an optimized ecosystem for SSI. It is composed of three projects; Indy, Ursa, and Aries.

Figure 4 Hyperledger Identity Stack

Hyperledger Indy was Hyperledger's first "identity-focused" blockchain framework. Includes verifiable credentials based on zero-knowledge proof (ZKP) technology, decentralized identifiers, a software development kit (SDK) for building agents and an implementation of a public, permissioned distributed ledger. [2]
When the issuer wants to issue a credential, they must have:

- A DID on the blockchain that allows verifiers to find out who they are.
- A schema on the blockchain with the list of attribute names that the credential will contain.
- A credential definition on the blockchain which specifies which DID, schema and public keys are going to be used.

Finally, the issuer may want to be able to revoke credentials and if they do, they must also write a revocation registry to the ledger before issuing credentials.

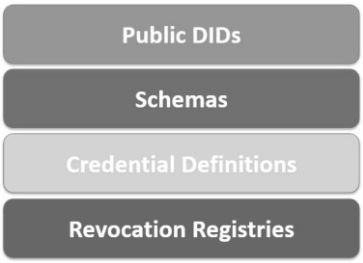

Figure 5 What goes on Indy blockchain

As Indy evolved within Hyperledger, there was a realization that the cryptography in Indy could be used in several Hyperledger projects, and even outside of Hyperledger. The decision was made to migrate the indy-crypto code repository out of Indy and into its own project: Hyperledger Ursa. [2]
Since Ursa work is made "behind the scenes" it is not important to know how it works for this research.
Indy is not the only implementation. In the long term, there will likely be many implementations that are used by different people, organizations and communities. This is when Aries comes, capable of using agents to interact between multiple ecosystems.

Aries is a toolkit designed for creating, transmitting, storing and using verifiable digital credentials. Its core are protocols enabling connectivity between agents using secure messaging to exchange information. Aries is all about peer-to-peer interactions between agents controlled by different entities—people, organizations and things.
Verifiable credentials can be exchanged based on DIDs rooted in different ledgers (based on Indy or other technology), and it is "verifiable credential-agnostic"—support different verifiable credential implementations within a single agent.
An Aries agent is a piece of software that enables an entity (a person, organization or thing) to assume one or more of the roles within the verifiable credential model—an issuer, holder or verifier—and allows that entity to interact with others that also have verifiable credential roles. [2]

2 Prototype solution

We want to create interaction between a holder (Application) and an issuer (IoT device) using Aries Cloud Agent (ACA) [3] through a public Indy ledger. This will be under the simulation of a delivery (imagine you have purchased a product and you want to see the current delivery status).
This makes taking into account all the necessary parts, including the creation of a graphical application, the configuration of the Raspberry Pi with the RFID sensor, the use of ACA project [3] and the configuration of the Indy ledger.

2.1 Scenario

The delivery status will be tracked by the IoT device. At the very beginning of the delivery, the product will be attached to an IoT device with an RFID sensor. We will assume that the deliverer has an RFID tag and the final user has another. When the deliverer gets the product, passes his RFID tag near the sensor and immediately the IoT device updates the status to "in delivery". As soon as the product reaches the final user he would do the same and the status would be updated to "delivered".
In order to watch in real-time the status, we will have a graphical Application. This App is not only meant for data tracking, it will have another feature, the creation invitation for the connection between the App and the sensor, which we will discuss later.
The following diagram shows the experiment, all the components involved, and their connections:

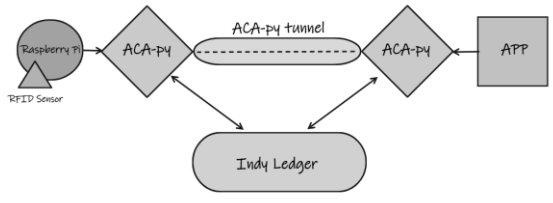

Figure 6 Components diagram

As you could see in the diagram, the IoT part is simulated with a Raspberry Pi and an RFID sensor. In this case, we will be using a Raspberry Pi 4 Model B and a PN532 NFC RFID MODULE V3. Its function will be to read the UID from the two different tags and make the proper calls to the Agent API to send de new status.
Immediately we have the ACA-py [3] instance for the sensor. It will be configurated with a public API address, the IoT is not running on the same machine and we need to make the calls through the internet.
Its function will be to communicate with the ledger and the App Agent to send the new status. Quite similar to the other part, we have the ACA-py [3] instance for the App. It is not necessary to configure a public API address because it will be running on the same machine as the App, so we can use the localhost address. Its function will be to communicate with the ledger and the sensor Agent to store the new status.
Then we have the App, which will be a simple graphical interface. It will have a log-in menu at the start, where you should introduce the App wallet password. If it is correct you will be sent to the menu, where you can create an invitation connection for the sensor and also see the credentials stored in the wallet, those which have the status written. Its function will be to create the connection invitation and fetch the credentials in the wallet.
Finally, we have the Indy ledger, in this case, we will be using one of the free public instances currently available on the internet, the BCovrin Dev ledger [4]. It has a really simple interface to see the nodes running, register DIDs, and see the ledger state. Its function will be to store the sensor DID, its schemas, and credential definitions.

2.2 Achieve Sending Credentials

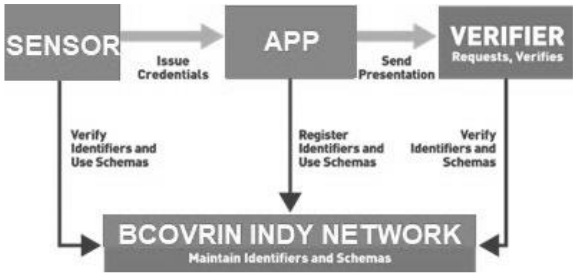

Figure 7 Project SSI flow diagram

The sending and receiving of the delivery status updates will be done by verifiable credentials. This credential contains an attribute field where we will write the new status value. Then it will be sent from the sensor to the App. In order to make this work, it is necessary to understand every step before and during the sending.
Just to clarify, this procedure is adapted to the Aries protocols, it may change in other SSI projects/solutions. Also, the verifier part is not implemented.
We will have a look at the whole process inside the ACA logic [5]. We will differentiate three parts.

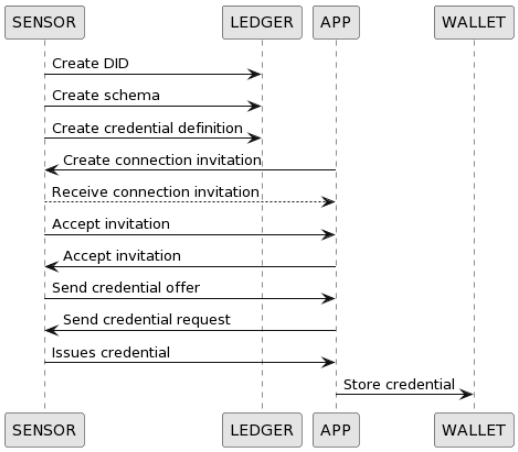

Figure 8 Concept UML diagram

The first one would be everything the ledger needs to have before sending a credential. Which is to have registered a public DID from the sender/issuer, a schema, and a credential definition related to that schema.
Once you have a DID registered you can create the connection, these steps are:
1. App creates a connection invitation.
2. Sensor gets the invitation and does the receive connection invitation.
3. Sensor accepts the invitation.
4. App accepts the invitation.
Now the connection should be created and we can move on to the credential dance. For this, it is necessary to have the schema and credential definition already created, in the ledger and the wallet.
1. Sensor sends a credential offer.
2. App sends a credential request.
3. Sensor creates and issues the credential.
4. App stores it in the wallet.
With this, we could fetch the credential from the App wallet in order to read it.

3 Conclusions

In conclusion, this paper has covered the concept that refers to sending IoT data using SSI blockchain technology, from a more theoretical part, which explained the main concepts of blockchain and SSI, from a more practical part providing a guide to sending verifiable credentials from an IoT device, a Raspberry Pi with an RFID sensor, using Hyperledger Aries Cloud Agent with an Hyperledger Indy public network.
We have been able to demonstrate in a real case how this technology could be used as a solution to the identification of IoT devices. Not only that, but we managed also to send data in the same SSI ecosystem using verifiable credentials.
There are many different options for implementing an SSI system, for example using a blockchain as Ethereum, but it should be written from scratch. This means creating the proper smart contracts to manage the DIDs, the credentials, the verifier, etc. This is truly a complex thing and not a really good option to deal with scalability, with ecosystem adaptation, and you are tied to the blockchain project properties changes.
This is why Hyperledger is a perfect solution to this. It brings a blockchain adaptation for each kind of system. In this case, the Identity Stack is a clear example of it.
With Indy we have the optimized solution for SSI network, and with Aries we have the solution to interact with the network.
Extrapolating to the IoT context, being certain that your device message is really your device who has sent it, is a big step in improving security. Not only that, a great number of device identification is also well managed. Furthermore, since IoT devices could have a big amount of interactions in a small period of time, the solution of Aries to bring a secure channel in the connection between agents could have a great impact on the system efficiency.
Certainly, it will take more time to bring this to a real production state because this is a very new technology that is still in progress, but the potential is amazing.
In the future development, exploring the part of the verifier would be an important point to see. The verifier is responsible for requesting proof from the holder and verifying that what is said in the credential is true. Here, we are relying on the Aries connection and that we are the ones that have done everything. But in a real implementation you may not have this confidence with every part of the equation, here is when the verifier comes in.

Literature

[1] «HL_Paper_HyperledgerOverview_102721.pdf», https://www.hyperledger.org/wp-content/uploads/2021/11/HL_Paper_HyperledgerOverview_102721.pdf.
[2] Introduction to Hyperledger Sovereign Identity Blockchain Solutions: Indy, Aries & Ursa.” LearnThings.Online (blog), March 5, 2020. https://learnthings.online/other/2020/03/05/introduction-to-hyperledger-sovereign-identity-blockchain-solutions-indy-aries-ursa.
[3] Hyperledger Aries Cloud Agent - Python. Python. 2019. Reprint, Hyperledger, 2022. https://github.com/hyperledger/aries-cloudagent-python/blob/00d97b3e0e6f713dfab383eb2e5e14e58472a47d/DevReadMe.md.
[4] “BCovrin Dev Indy Network.” Accessed June 6, 2022. http://dev.bcovrin.vonx.io/.
[5] Jong, Laurence de. “Becoming a Hyperledger Aries Developer - Getting Started.” Laurence de Jong, March 11, 2021. https://ldej.nl/post/becoming-a-hyperledger-aries-developer-getting-started/.