



Universidad Politécnica
de Madrid

**Escuela Técnica Superior de
Ingenieros Informáticos**



Master in Data Science

Master Thesis

Decentralized Identity Demonstrator

Author: Alisson Vaneza Gutierrez Garcia

Madrid, July, 2024

<https://github.com/alivane/TFM-decentralize-identity>

This Master Thesis has been deposited in ETSI Informáticos de la Universidad Politécnica de Madrid.

Master Thesis

Master in Data Science

Title: Decentralized Identity Demonstrator
July, 2024

Author: Alisson Vaneza Gutierrez Garcia
Supervisor: Víctor Rodríguez Doncel
Departamento de Inteligencia Artificial
ETSI Informáticos
Universidad Politécnica de Madrid

Summary

Varias aplicaciones web hoy en día requieren que los usuarios se identifiquen con un nombre de usuario y contraseña o usen otras formas de identificación, como Google Gmail o Facebook para registrarse. Estos tipos de autenticación siempre dependen de un tercero, como el proveedor o el propio sistema para la autenticación. Además, otras personas podrían potencialmente acceder a la aplicación web usando esas credenciales sin la autorización del usuario original, lo que genera problemas de confianza.

El objetivo principal de este trabajo de maestría es diseñar y desarrollar un prototipo que utilice identificadores descentralizados (DIDs) y credenciales verificables (VCs) para autenticar a los usuarios en un escenario de intercambio de divisas entre pares (P2P), al que llamamos Expid. La implementación de estos componentes mejora tanto la seguridad como el control del usuario sobre los datos personales. Este proyecto se alinea con el Nivel de Preparación Tecnológica 3 (TRL 3), centrándose en la prueba experimental de concepto y la validación de las funcionalidades principales.

La aplicación Expid no solo demuestra la aplicación práctica de las tecnologías de identidad autosoberana (SSI), DID y VC, sino que también aborda aspectos críticos de la soberanía de los datos, la gobernanza y la fiabilidad. En el contexto de los espacios de datos e iniciativas como Gaia-X, este trabajo destaca el potencial para el intercambio y la gestión de datos seguros y descentralizados, adhiriéndose a las normas y regulaciones europeas.

Los resultados demuestran la viabilidad de la solución propuesta, mostrando que los usuarios pueden intercambiar divisas de manera segura sin intermediarios, reduciendo así los costos de transacción y mejorando la privacidad al verificar las identidades de los usuarios. La aplicación Expid sirve como una herramienta didáctica, ilustrando la integración de componentes de SSI y destacando el potencial transformador de la identidad descentralizada en el intercambio de divisas.

Esta tesis contribuye al debate más amplio sobre la identidad descentralizada, ofreciendo perspectivas sobre los desafíos y beneficios de su implementación y estableciendo una base para futuros avances en un entorno seguro y confiable.

Abstract

Several web applications today require users to identify themselves with a username and password or use other forms of identification such as Google Gmail or Facebook for registration. These types of authentication always depend on a third party, such as the provider or the system itself for authentication. Additionally, other people could potentially access the web application using those credentials without the original user's authorization, leading to trust issues.

The primary objective of this master's work is to design and develop a prototype that utilizes decentralized identifiers (DIDs) and verifiable credentials (VCs) to authenticate users in a peer-to-peer (P2P) currency exchange scenario, which we call Expid. The implementation of these components enhances both security and user control over personal data. This project achieves the Technology Readiness Level 3 (TRL 3), focusing on experimental proof of concept and validation of the core functionalities.

The Expid application not only showcases the practical application of self-sovereign identity (SSI), DIDs, and VCs technologies but also addresses critical aspects of data sovereignty, governance, and reliability. In the context of data spaces and initiatives such as Gaia-X, this work underscores the potential for secure, decentralized data sharing and management, adhering to European standards and regulations.

The results demonstrate the viability of the proposed solution, showing that users can securely exchange currencies without intermediaries, thus reducing transaction costs and enhancing privacy by verifying users' identities. The Expid application serves as a didactic tool, illustrating the integration of SSI components and highlighting the transformative potential of decentralized identity in currency exchange.

This thesis contributes to the broader discourse on decentralized identity, offering insights into the implementation challenges and benefits, and setting a foundation for future advancements in secure and trustworthy environment.

Contents

| | | |
|----------|--|----------|
| 1 | Introduction | 1 |
| 1.1 | Motivation | 1 |
| 1.2 | Objectives | 1 |
| 1.3 | Thesis Organization | 2 |
| 1.4 | Legal framework | 2 |
| 2 | State of the Art | 5 |
| 2.1 | General Notions on Identity and Security | 5 |
| 2.1.1 | Authentication and Authorization | 5 |
| 2.1.2 | Web Authentication or WebAuthn | 6 |
| 2.2 | Self Sovereign Identity (SSI) | 8 |
| 2.2.1 | World Wide Web Consortium (W3C) | 9 |
| 2.2.2 | W3C Decentralized Identifiers (DID) | 9 |
| 2.2.3 | W3C Verifiable Credentials (VCs) | 10 |
| 2.2.4 | W3C WebID | 11 |
| 2.2.5 | W3C Solid | 12 |
| 2.3 | Cryptography Techniques | 13 |
| 2.3.1 | Symmetric Encryption | 13 |
| 2.3.2 | Asymmetric Encryption | 13 |
| 2.3.2.1 | Digital Signatures | 14 |
| 2.3.2.2 | Web Application Examples | 14 |
| 2.3.3 | One-Time Pad | 15 |
| 2.4 | Blockchain | 15 |
| 2.4.1 | SHA-256 Algorithm | 16 |
| 2.4.2 | Peer to Peer or P2P Network | 16 |
| 2.4.3 | Ethereum | 17 |
| 2.5 | European Blockchain Services Infrastructure (EBSI) | 17 |
| 2.6 | Data Spaces and Gaia-X | 18 |
| 2.7 | Applications in the Domain of Currency Exchanges | 19 |
| 2.7.1 | PayPal | 19 |
| 2.7.2 | Hodl Hodl | 19 |
| 2.8 | Technologies | 19 |
| 2.8.1 | Veramo | 19 |
| 2.8.2 | InterPlanetary File System (IPFS) | 20 |
| 2.8.2.1 | InterPlanetary Linked Data (IPLD) | 20 |
| 2.8.2.2 | InterPlanetary Name System (IPNS) | 22 |
| 2.8.3 | Method IPID for DID | 22 |
| 2.9 | Platforms for Managing Decentralized Applications | 22 |

| | |
|---|-----------|
| 2.107 Laws of Identity | 23 |
| 2.11 Terminology | 24 |
| 3 Design and Implementation | 27 |
| 3.1 Methodology | 27 |
| 3.2 Proof of Concept Design | 28 |
| 3.2.1 Functional and Non-Functional Requirements | 28 |
| 3.2.2 Mockup Design | 29 |
| 3.2.3 Data Storage | 30 |
| 3.2.3.1 Collections | 30 |
| 3.2.3.2 Data Structure and Data Types | 31 |
| 3.2.4 Workflow Overview | 32 |
| 3.2.5 Sign Up and Sign In Workflows | 33 |
| 3.3 Implementation | 37 |
| 3.3.1 Technological Choice | 37 |
| 3.3.2 Repositories and Application | 38 |
| 3.3.3 Problems encountered | 39 |
| 4 Evaluation and Conclusion | 41 |
| 4.1 Evaluation | 41 |
| 4.1.1 Description of the Demo Steps | 41 |
| 4.1.1.1 Home Overview | 42 |
| 4.1.1.2 Sign Up Workflow | 42 |
| 4.1.1.3 Login Workflow | 48 |
| 4.1.1.4 Currency Offering Process | 49 |
| 4.1.1.5 Steps to Get Currency. | 51 |
| 4.1.1.6 Display and Verify Currency Offerings | 53 |
| 4.1.1.7 QR Code Generation and Verification for Buyers. | 55 |
| 4.1.1.8 DID Document | 56 |
| 4.1.1.9 Verifiable Credentials | 58 |
| 4.1.2 Validation of Requirements | 62 |
| 4.1.2.1 Validation Against Cameron's Laws of Identity | 63 |
| 4.1.2.2 Compliance with Requirements | 63 |
| 4.2 Conclusion | 64 |
| 4.2.1 Main Contribution | 65 |
| 4.2.2 Future Work | 65 |
| References | 70 |
| Bibliography | 70 |
| Annex | 71 |
| .1 README files | 71 |
| .1.0.1 Front-End | 71 |
| .1.0.2 Back-End | 72 |
| .1.0.3 Veramo Agent | 73 |

Chapter 1

Introduction

1.1 Motivation

In today's world, people often travel to foreign countries to explore new places, cuisines, and cultures. While many travelers rely on credit cards for transactions, there remains a challenge for those who prefer to exchange their local currency for foreign currency. Traditional currency exchange offices typically impose high commissions, both in the traveler's home country and abroad.

What if there was a peer-to-peer (P2P) solution for currency exchange, eliminating the need for intermediaries and exorbitant fees? The aim of this thesis is to investigate the possibility of decentralized identity using standard specifications, demonstrated in the context of secure money exchange between individuals. By establishing a standardized exchange rate based on the current market value in the respective country, users can exchange currency without exceeding this set rate. Through decentralized identity verification, we can ensure the authenticity of users and mitigate the risk of interexchange money between people.

Currently, there are various methods of authentication and authorization, many of which operate in a centralized manner. For example, OAuth 2.0 is primarily designed for centralized authentication and authorization scenarios. However, for this thesis, we aim to work with a decentralized approach to demonstrate that it is possible to develop applications that do not rely on a central entity for identity management, giving users greater control over their data than the entity itself.

1.2 Objectives

This thesis aims to demonstrate the application of decentralized identity in peer to peer currency exchange, thereby eliminating intermediaries and exorbitant fees prevalent in traditional currency exchange methods. The primary objectives are as follows:

1. Demonstrate decentralized identity verification to verify that the person is who they claim to be.
2. Develop a prototype platform for currency exchange that utilizes decentralized identities and verifiable credentials to validate the authenticity of users when

they exchange currency peer-to-peer.

3. Develop a proof of concept that validates the proposed method, reaching a Technology Readiness Level (TRL) of 3.

1.3 Thesis Organization

This master thesis is organized into four chapters:

1. **Introduction:** The first chapter explains the main motivation behind this project, outlining the objectives, thesis organization, and the legal frameworks involved.
2. **State of the Art:** Chapter two delves into the primary concepts researched for developing the proof-of-concept (PoC). It discusses relevant use cases, particularly focusing on exchange applications, and explains the technologies and tools integrated into this PoC, such as Veramo for verifiable credentials (VCs). Additionally, this chapter covers Kim Cameron's seven laws of identity and general terminology used in the project.
3. **Design and Implementation:** Chapter three details the PoC's development process, including the methodology, overall design, data storage design, initial mockups, and the authentication validation workflow. It also explains the technological choices, provides URLs generated for the repositories and web application, and addresses the challenges encountered during the implementation.
4. **Evaluation and Conclusion:** The final chapter evaluates the demo process, outlining each step and its functionality. It validates the requirements, including compliance with Cameron's laws of identity and the functional and non-functional requirements. The chapter concludes with a discussion of the main contributions, conclusions, and future work.

1.4 Legal framework

This section helps in understanding the legal value and the limits of the technical contributions presented in this work.

Nowadays, there are no specific EU policies for SSI in force; however, there are initiatives to promote its implementation. An example of this is the European Self-Sovereign Identity Framework¹ (EU SSI Framework). Although not recent, it explores the technical and legal implementations and supports the implementation of SSI for Small and Medium-sized Enterprises (SMEs) and technology companies in Europe through funding. Additionally, there is an agreement between two countries to work with SSI. In this case, Germany and Spain have agreed to test a cross-border digital identity ecosystem².

The EU Digital Identity³ is a pilot implementation by the governments in the EU. The objective is to implement authentication using an e-identity system to share personal

¹<https://essif-lab.eu/>

²<https://mksmart.com.vn/world-news/germany-and-spain-to-test-cross-border-digital-identity-ecosystem.html>

³https://commission.europa.eu/strategy-and-policy/priorities-2019-2024/europe-fit-digital-age/european-digital-identity_en

Introduction

information with public providers systems across all member states without the need for a password. This implementation is for all EU citizens.

The eIDAS Regulation (regulation on electronic identification and trust services for electronic transactions in Europe) is a legal framework for a digital identification allowing to trust services across all EU member states, "eIDAS is a key enabler for secure cross-border transactions"⁴. The focus are in electronic identification, electronic signature, and website authentication.

The forthcoming regulation on identity is called the "eIDAS 2 Regulation". "eIDAS 2 would make the implementation of verification processes in any industry easier by simply relying on the technology that reads the e-ID of citizens through their wallets"⁵. The eIDAS2⁶ was published as a proposal on June 3, 2021, and is based on the original eIDAS Regulation. This proposal focuses on user control over data, introduces stricter authentication methods, and includes new trust services like personal data wallets. It is estimated that EU member states will provide EU digital identity wallets in the first quarter of 2024.

Several other norms and initiatives exist globally that relate to SSI such as Gaia-X⁷ that is a federated system on the cloud aligned with the EU values and supporting the principle of SSI, DGA (European Data Governance Act)⁸ that is a legislation focusing in sharing and reuse the data within the European Union countries, GDPR (General Data Protection Regulation)⁹ that is a regulation to protect individual information privacy.

Additionally, international standardization bodies play an important role defining the technical specifications such as W3C (World Wide Web Consortium)¹⁰ that is a consortium that establish different standards and guidelines to implement SSI, including Verifiable Credentials (VCs) and Decentralized Identifiers (DIDs).

⁴<https://digital-strategy.ec.europa.eu/en/policies/eidas-regulation>

⁵<https://www.mobbeel.com/en/blog/what-is-eidas2-and-what-changes-from-eidas/>

⁶<https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:52021PC0281from=ES>

⁷<https://gaia-x.eu/what-is-gaia-x/about-gaia-x/>

⁸<https://digital-strategy.ec.europa.eu/en/policies/data-governance-act>

⁹<https://gdpr.eu/what-is-gdpr/>

¹⁰<https://www.w3.org/>

Chapter 2

State of the Art

2.1 General Notions on Identity and Security

According to the definition of the Spanish Royal Academy (RAE), one of the meanings of identity is "the set of characteristics of a person or thing that allow them to be distinguished from others"¹.

It is important to note that identity is something valuable that can be taken without permission by another person. Moreover, in the digital world, individuals can impersonate your identity, or the providers that currently have your personal information can use it without your consent for research or even to sell your information.

2.1.1 Authentication and Authorization

The various methods of authentication and authorization can sometimes be confusing when navigating the internet and accessing various platforms. While these terms may seem similar, they represent distinct concepts.

Authentication primarily concerns the verification of a user's identity. It involves validating the credentials provided by the user, such as username and password, biometric data, or tokens. Authentication ensures that the user is who they claim to be, granting them access to their account or system.

Authorization, on the other hand, pertains to determining what actions or resources a user is allowed to access once they have been authenticated. It involves granting or denying permissions based on the user's identity and predefined rules or policies. Authorization controls access to platforms, services, and other resources, ensuring that users only have access to the resources they are authorized to use.

So we can say that authentication verifies the user's identity, while authorization determines what actions or resources the user can access once authenticated.

Authentication:

- **Email/Username and Password:** This authentication method is commonly used for individuals registered on a platform. Users provide their email address or username, along with a corresponding password, to verify their identity and

¹<https://dle.rae.es/identidad>

2.1. General Notions on Identity and Security

gain access to the platform. The platform compares the provided credentials against its stored records to authenticate the user.

- **OpenID Connect or OIDC:** It is an authentication protocol that allows individuals to authenticate with third-party identity providers, such as Google, LinkedIn, or Facebook, among others. With OIDC, users can use their existing accounts from these providers to log in to other websites or applications without the need to create new credentials. OIDC builds upon the OAuth 2.0 framework and adds an authentication layer, enabling secure and standardized authentication processes across different platforms and services.
- **JSON Web Tokens or JWT:** Are an open standard for securely transmitting information between parties as a compact with a self-contained token. JWTs consist of encoded JSON data that can include various claims, such as user identity, permissions, or other attributes. These tokens are commonly used for authentication and authorization purposes in web applications and APIs. JWTs can encapsulate information about the user's identity and access privileges, allowing applications to make access control decisions based on the token contents
- **Others:** Biometric authentication uses unique biological characteristics such as fingerprints, facial features, or iris patterns for user verification. Single sign-on (SSO) allows users to access multiple applications or services with a single set of credentials. Multi-factor authentication (MFA) requires users to provide multiple forms of verification, such as passwords, security tokens, or biometric scans, to access an account, enhancing security.

Authorization:

- **Access Control Lists or ACLs:** Are a mechanism used to control access to system resources by specifying permissions for individuals or systems. An ACL is associated with each resource and determines which users or groups are granted or denied access to that resource and what actions they are allowed to perform. Additionally it could consist of a list of entries, each specifying a user, group, or system entity and the permissions granted to them.
- **Open Authorization or OAuth:** It is an authorization framework that enables third-party applications to access resources on behalf of a resource owner without requiring the owner to share their credentials. Instead, OAuth facilitates secure access by providing a mechanism for the client application to obtain an access token, which grants access to specific resources on the owner's behalf. This access token is obtained through an authorization process, during which the resource owner grants consent for the client application to access their resources.

2.1.2 Web Authentication or WebAuthn

As we saw in the subsection before regarding authentication, we need a mechanism to verify that we are who we claim to be. Most authentication methods require a password for this verification. However, there is now another method of authentication that integrates cryptographic keys, eliminating the need for a password to access web applications.

Web Authentication is a standard protocol developed by the World Wide Web Consor-

State of the Art

tium (W3C) to enable passwordless authentication on the web. It provides a standardized approach for users to authenticate themselves using biometrics, PINs, or security keys, rather than traditional passwords. Additionally, WebAuthn includes APIs that developers can use to integrate this authentication mechanism into their web applications.

The benefits of integrating WebAuthn into applications include mitigating many common cyberattacks, such as phishing and remote replay attacks on credentials. Additionally, WebAuthn reduces the risk of password reuse across multiple platforms, as users no longer need to remember passwords. Implementing WebAuthn also promotes modern security practices.

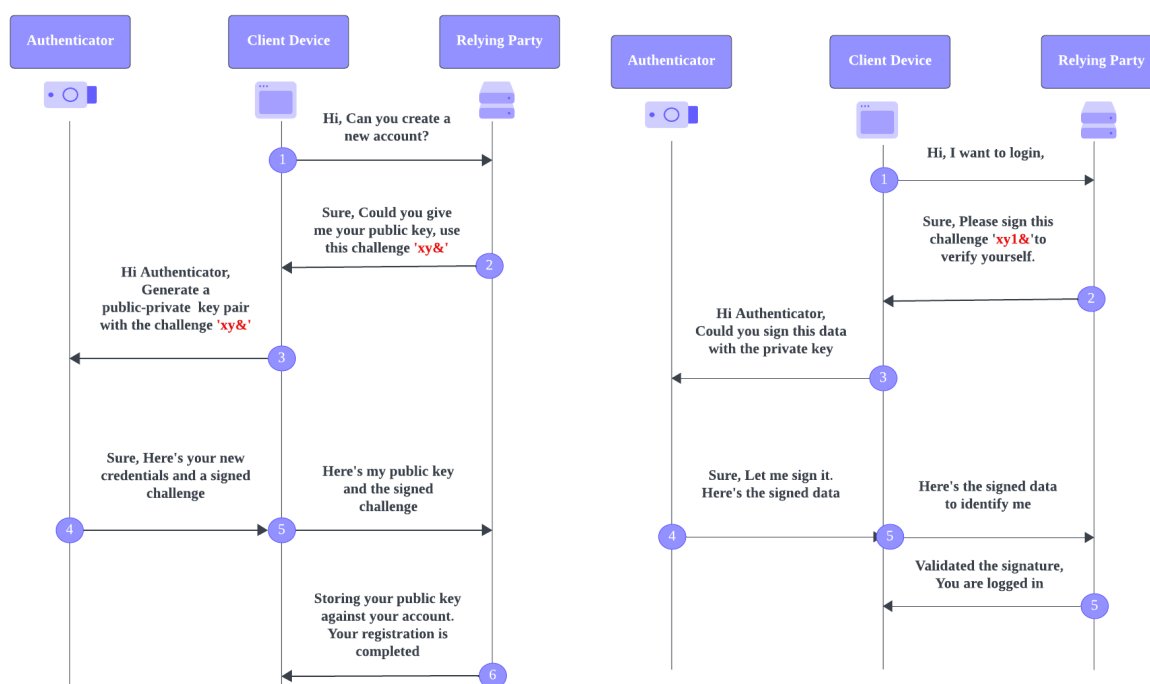


Figure 2.1: WebAuthn Flow: Registration on the Left, Authentication on the Right

In Figure 2.1, we can observe two flows illustrating how WebAuthn operates. These diagrams are sourced from the blog². Before explaining both diagrams, it is important to describe some key concepts needed to understand the flow of WebAuthn in registration and authentication:

- **Asymmetric cryptography:** During the registration process, a pair of cryptographic keys is generated. The public key is registered on the server, while the private key is stored securely on the user's device.
- **Relying Party:** It is the web service or application that relies on WebAuthn for user authentication. It interacts with the client device and verifies authentication requests. The relying party is responsible for initiating and managing the authentication process, validating the authenticity of user credentials, and granting access to resources based on the authentication outcome.

²<https://auth0.com/blog/webauthn-a-short-introduction/>

2.2. Self Sovereign Identity (SSI)

- WebAuthn client device: It refers to the user's device, such as a smartphone or computer, which is compatible with WebAuthn and where the user's private keys are stored to interact with it.
- Authenticator: There are two types of authenticators used in WebAuthn. The first type is the Platform authenticator, which refers to devices such as laptops or smartphones. The second type is the Roaming authenticator, which includes external hardware security keys that users can carry with them to authenticate across multiple devices or platforms.
- Challenge: It is a code generated by the relying party, typically consisting of alphanumeric characters or other types of codes. This challenge is presented to the user for signing, usually as part of the authentication process.

Following the flow of the Figure 2.1, the registration flow is the following:

1. The user wants to create an account in a web application.
2. The relying party send a challenge.
3. The client send a request to the authenticator to generate the asymmetric keys.
4. The authenticator sent the signed challenge and the public key to the client.
5. The client send the public key and the signed challenge to the relying party.
6. The relying party store the public key and the registration is compleated.

In contrast to registration, the authentication process (Figure 2.1, on the right side) in WebAuthn does not require the relying party to store the public key. Instead, it only requires the user to sign the challenge provided by the relying party. This signed challenge serves as the proof of authentication, allowing the relying party to verify the user's identity without storing sensitive information.

2.2 Self Sovereign Identity (SSI)

Presently, data holds a value comparable to that of gold in previous years. The various applications that we generally use, the websites to which we have access, and even the cookies applied on these websites store our information and are marketed with our data, yielding profits. But what happens if we only want to share the minimum information or refrain from sharing any information with these applications and websites and also retain our personal information by ourselves? This is where the term "*self-sovereign identity*" emerges.

Before understanding the term *Self Sovereign Identity*, it is essential to grasp the definition of "sovereign"³ as "having or exercising supreme power or authority", and "identity"⁴ as "the distinguishing character or personality of someone or something". These definitions help introduce the term "*Self Sovereign Identity*".

In the context of this work, *Self Sovereign Identity* or SSI refers to the ownership or full control of our personal digital information and our freedom to share only the minimum necessary information with third party platforms, websites or individuals and

³<https://www.merriam-webster.com/dictionary/sovereign>

⁴<https://www.merriam-webster.com/dictionary/identity>

even revoke the access to our information as when we want, everything without the intervention of third party providers and manage these information by ourselves that is, independent of any organization, with this system implementation we can keep the anonymity of our information when desired and protect our sensitive information.

2.2.1 World Wide Web Consortium (W3C)

The World Wide Web Consortium or W3C is an international organization that "develops a standards and guidelines to help everyone build a web based on the principles of accessibility, internationalization, privacy and security⁵.", some famous standards that we use every day on the web are the HTML⁶ (HyperText Markup Language), CSS⁷ (Cascading Style Sheets) and XML⁸ (Extensible Markup Language).

2.2.2 W3C Decentralized Identifiers (DID)

Before the advent of portable telephone numbers, using a phone number depended on the provider each user had. For example, if you already had a phone number provided by Lebara and you wanted to switch to another company, you could not keep the same phone number. You would have to change your phone number, which meant all your contacts needed to update your new number.

However, nowadays it is different. People can change providers without needing to change their phone numbers. Today, phone numbers are required to be unique within the country where people are located. For instance, if two people had the same number in the same country with the same country code (e.g., +34 for Spain), WhatsApp might not work for both, or it could work, but the privacy of messages for one person might be compromised for the other.

The W3C DID is a way to identify any subject (e.g., persons, companies, organizations, etc.). This is achieved through the URI (Uniform Resource Identifier) and other standardization defined by the W3C that we will see in this section. The DID was defined as: "Decentralized identifiers (DIDs) are a new type of identifier that enables verifiable, decentralized digital identity"⁹.

Unlike phone numbers, if a person stops using their DID, it is not reassigned to another person because DIDs are permanent identifiers, meaning they never change. With the implementation of DIDs, it is possible to retrieve information contained in the DID document (this is explained later) corresponding to the identifier we want to use. The way to prove that a DID is valid is through cryptographic techniques, adding a security layer to the DID. As they are decentralized, DIDs do not require intermediaries or providers to register the identifier. This means that the entity that wants a DID can register it on their own servers, local devices, or anywhere else they have control

The DID is composed of the scheme identifier, which is "did", the DID method (such as "ethr" corresponding to an Ethereum address), and the DID method-specific identifier. An example is shown below:

⁵<https://www.w3.org/>

⁶https://www.w3.org/wiki/HTML/Training/What_is_HTML

⁷<https://www.w3.org/Style/CSS/>

⁸<https://www.w3.org/XML/>

⁹<https://www.w3.org/TR/did-core/>

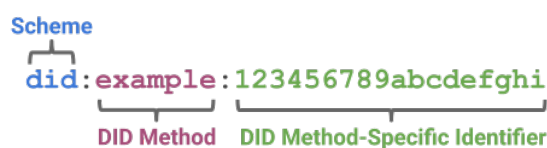


Figure 2.2: A simple example of a decentralized identifier (DID)

The DID document contains information in JSON-LD format. Typically, the contents of this document include public keys or other cryptographic proof material, service endpoints for trusted interactions, authentication mechanisms for proving control of the DID, and other metadata. An example is provided below:

```
{
  "@context": [
    "https://www.w3.org/ns/did/v1",
    "https://w3id.org/security/suites/ed25519-2020/v1"
  ],
  "id": "did:example:123456789abcdefghi",
  "service": [{
    "id": "did:example:123#linked-domain",
    "type": "LinkedDomains",
    "serviceEndpoint": "https://bar.example.com"
  }],
  "authentication": [{
    "id": "did:example:123456789abcdefghi#keys-1",
    "type": "Ed25519VerificationKey2020",
    "controller": "did:example:123456789abcdefghi",
    "publicKeyMultibase": "zH3C2AVvLMv6...PV"
  }]
}
```

The item ID corresponds to the DID subject. The "service" item, which is the first item in the "serviceEndpoint", corresponds to the endpoints that allow communication with this DID subject. The "authentication" item is the verification method that works only for authenticating the DID subject.

2.2.3 W3C Verifiable Credentials (VCs)

Nowadays, the validation of our credentials, such as driver's licenses and university degrees, as well as personal information such as age and birthday, needs to be verifiable. Even healthcare data requires verification via the internet. The question here is: How can we trust that the information provided by one person via the internet is authentic and has not been cloned or duplicated by another person?

Verifiable Credentials (VCs) represent physical credentials in digital form, establishing proof between the entity that issues the credential, the individual or entity it pertains to, and the entity requesting the information. Additionally, incorporating additional technologies such as digital signatures makes them more tamper-evident and trustworthy compared to their physical counterparts.¹⁰

¹⁰<https://www.w3.org/TR/vc-data-model-2.0/what-is-a-verifiable-credential>

The ecosystem of VCs are composed by:

- Issuer: Is the entity to emit the verifiable credentials.
- Holder: Is who is responsible to keep that credentials
- Verifier: Is who is responsible to verifier that credentials such as a person

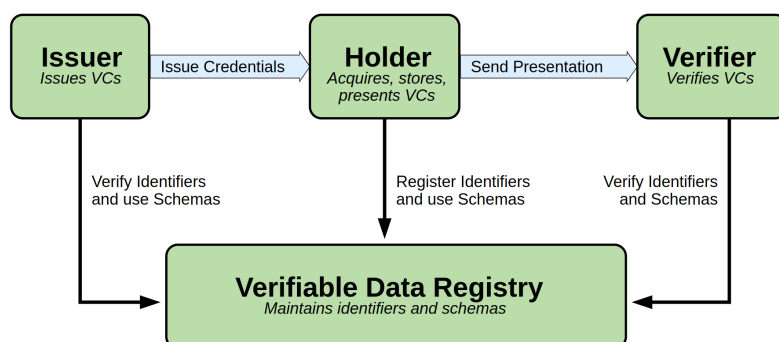


Figure 2.3: The roles and information flows forming the basis for this specification

2.2.4 W3C WebID

Maintaining profiles, adding contact details, uploading photos, and so on can be exhausting in the networked age. In this day and age, the primary method of authorization across different platforms or providers is through email or a unique username and password, which often necessitates registration on these services. The information provided during registration, such as name, last name, and birthday, serves as authentication. This process is not limited to a single platform but extends to others like Facebook, LinkedIn, and more. Each time we wish to access a specific platform or service on the internet, we are required to fill out some form of personal information, perpetuating this cycle.

The WebID was introduced by Dan Brickley and Tim Berners-Lee in 2000 and they said that: "A WebID is a way to uniquely identify a person, company, organisation, or other agent using a URI."¹¹.

As mentioned earlier, a WebID can identify natural people or entities on the internet using a URI. It eliminates the need to register repeatedly on new platforms or services and fill out information each time. The objective of WebID is to have the entity's or person's information already created in their own domain, bypassing traditional methods of authentication and authorization. This is achieved through an RDF file with FOAF¹² vocabulary, stored on the server within the domain containing the person's or entity's information.

The URL format should appear as follows:

¹¹<https://www.w3.org/wiki/WebID>

¹²<https://en.wikipedia.org/wiki/FOAF>

2.2. Self Sovereign Identity (SSI)

`http://your.isp.com/whatever/~yourusername/foaf.rdf#ABC`

In this structure, `http://your.isp.com` denotes the domain, `foaf.rdf` indicates the RDF file, and `#ABC` represents the initials, which can be customized as identifiers such as `#me` or `#this`.

The structure presented earlier pertains solely to authentication. Once authentication is completed, we need a method for authorization and it involves asymmetric certificates in the WebID. Traditionally, accessing different platforms requires a typical login process as mentioned before. However, with WebID authentication protocol, we must generate asymmetric certificates to authorize platform access to our data. Therefore, instead of providing our usual credentials, we simply inform the platforms where we wish to gain access that we intend to do so using our generated public key.

The WebID helps us avoid rewriting information and recreating our typical email and password. While other implementations such as OAuth for authorization, attempt to avoid silos with OpenID for authentication, they store the information in their own service providers like Facebook, Gmail, LinkedIn and so on. However, these are not portable because the information is not accessible to us at all. The WebID is not just a homepage or a webpage URL. To have a WebID, we need the points mentioned earlier, in the following graphical 2.4 we can see the step of the WebID authentication protocol makes:

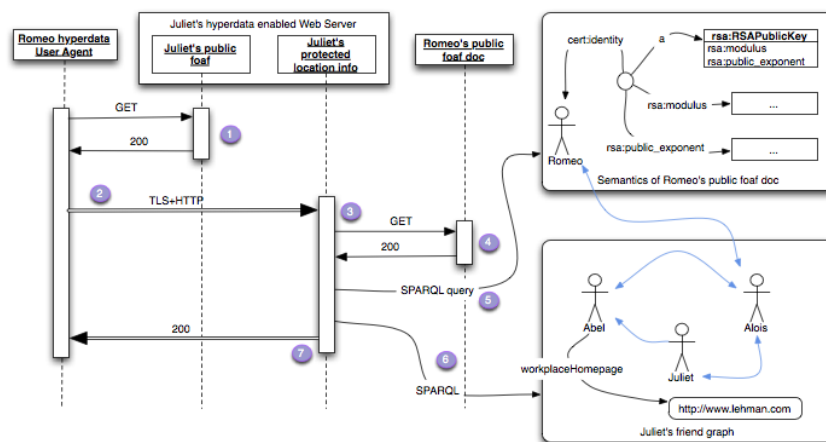


Figure 2.4: Foaf+ssl sequence diagram¹³

2.2.5 W3C Solid

As mentioned in this project, the current method of data handling involves relying on providers where we do not have full control. The approach to managing data has now changed with W3C Solid¹⁴, a decentralized web technology uses open standards where developers from W3C, developer communities, and interested organizations collaborate. The goal is to achieve full control of our data without the need to store it in a centralized location or work with specific providers. This is called a POD by Solid.

¹⁴<https://solidproject.org/about>

There are several Pod providers¹⁵ that users can use to start using Solid. Some important points to consider are whether the pod is third-party and its geographical location, as legislation varies between countries. It is important to understand the terms that providers offer and how they handle data. Additionally, with Solid, users are not tied to a specific provider and can change as needed. However, it is advisable to have our own server to host our data.

2.3 Cryptography Techniques

2.3.1 Symmetric Encryption

Symmetric Encryption is a type of encryption where both the sender and receiver share a secret key. This key is used to encrypt and decrypt the information, ensuring that only authorized parties can access the data, as is shown in the image below 2.5.

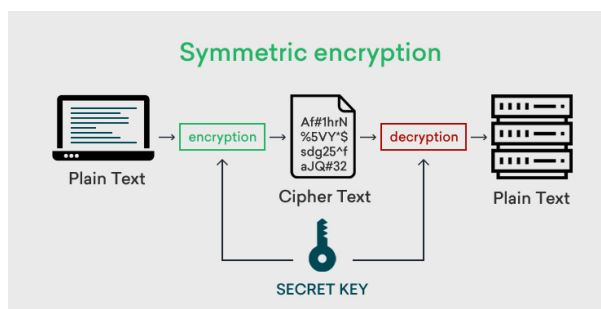


Figure 2.5: Symmetric encryption example

2.3.2 Asymmetric Encryption

Asymmetric encryption¹⁶, also known as public key cryptography, differs from symmetric cryptography in that it utilizes a public key that can be shared publicly. It consists of a private key, which is kept by the individual who generates it, and a public key, which is shared with the public.

An example of how it can be used is as follows: Suppose individual A wants to share a document with individual B. Both individuals have public keys shared to the public, while their private keys are kept confidential.

¹⁵<https://solidproject.org/users/get-a-pod>

¹⁶<https://www.bitpanda.com/academy/en/lessons/what-is-asymmetric-encryption/>

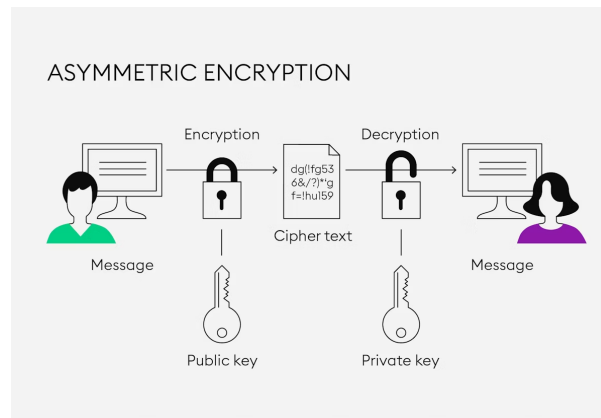


Figure 2.6: Asymmetric Encryption Example

The processes of the image example 2.6 is the following: When individual A shares document X, it is encrypted with the public key of individual B, ensuring that only individual B can decrypt it. Alternatively, if individual A wants to share a document and ensure that it has not been altered, individual A signs the document with their private key. Others can then verify that the document was signed by individual A using the corresponding public key belonging to the individual A, thus preventing any tampering or forgery.

2.3.2.1 Digital Signatures

Digital signatures is a cryptography methods where are commonly used to ensure the authenticity and non-repudiation of data. This cryptographic technique involves the sender signing the message with their private key, which could be a document or any other form of data. The receiver then verifies the authenticity of the message using the sender's public key.

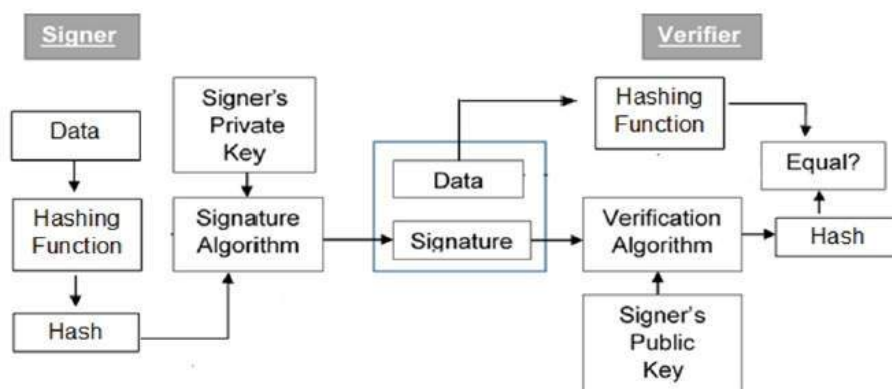


Figure 2.7: Model of digital signature.

2.3.2.2 Web Application Examples

Some examples of web applications that use it are the following:

- **VALIDe - REDSARA:** Online signature and certificate validation application and signature services demonstrator¹⁷
- **FNMT Electronic Citizen Certificate:** The FNMT Electronic Citizen Certificate is the electronic certification issued by the FNMT-RCM that links its subscriber to Signature Verification Data and confirms their identity. The electronic Citizen Certificate will allow to people to carry out procedures safely with the Public Administration and Private Entities through the Internet ¹⁸

2.3.3 One-Time Pad

The one-time pad, or OTP, is an encryption technique that generates a random key. Typically, this key is equal to or longer than the message being encrypted. Both the sender and receiver must know the random key, keeping it secret between them. Importantly, the key should only be used once, ensuring it is not reused in whole or in part.

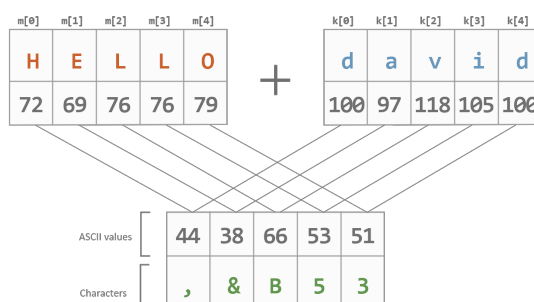


Figure 2.8: One-time pad example of complete encryption: The plaintext "Hello" is encrypted using the secret key "david".

2.4 Blockchain

In several countries, when people want to buy a new house, they need to go through different steps to obtain the title property of the new house. This is necessary to formally establish on paper that a house belongs to a particular individual. The justification for this ownership must be provided either to a notary who will authenticate it, or to some authority responsible for endorsing the property ownership in the country. This process is not just about purchasing the house but also obtaining the official title property that confirms ownership.

However, what happens if that title is corrupted by the government or any other entity? Or if the title property no longer exists due to an accident involving the document that endorsed the property ownership?

The example described early was for introduce how are the transactions today in the not digital world, but there is term that arose in the digital world regarding to the transactions and that term is the Blockchain.

¹⁷<https://valide.redsara.es/valide/faqs.html?lang=en>

¹⁸<https://www.sede.fnmt.gob.es/certificados/persona-fisica>

The blockchain is a technology of store transaction that introduces various other terms such as decentralization, distribution, peer-to-peer (P2P) networks, security, immutability, smart contracts, and more that we will see in this section.

Firstly, blockchain is a series of blocks that store transaction information or even pieces of code, along with the previous hash and other details. Every time a block is created, a hash is generated, which acts like a fingerprint. This concept is explained further in the subsection 2.4.1. Each block is linked to the one before it through the previous hash, hence the term Blockchain. This linkage makes it difficult to alter a single block, as changing one block's hash would require changing the hashes of subsequent blocks as well, this is called as an immutable ledger. Additionally, adding more nodes to the blockchain further enhances its security. If a block is altered, the other nodes in the network require consensus to update the altered blockchain.

2.4.1 SHA-256 Algorithm

Everyone has a different fingerprint that identifies us, similar to how each person's fingerprint is unique. While it is highly improbable for two people to have the same fingerprint, in the digital world, we use methods to ensure that the content we share every day remains uncorrupted. One such method is applying the SHA256 or Secure Hash Algorithm, which belongs to the SHA-2 family¹⁹. There is also the SHA512, which corresponds to 64-bit words. The SHA256 algorithm was designed by the United States National Security Agency (NSA).

Some key features of the SHA256 algorithm include the requirement that the message length should be less than 264 bits, and irreversibility, meaning that for each input, we have just one output and not the other way around. The output of the SHA256 algorithm is a string of 64 hexadecimal characters. An example is as follows:

```
3d20e34f5b76f74f265beade0cc7b2b46153abf76865b89c6a7cc119ac5bf782
```

2.4.2 Peer to Peer or P2P Network

As explained in the introductory example of this section, in terms of blockchain, if an individual attempts to change the hash of a block and the subsequent blocks in an effort to corrupt the blockchain, the question that arises is: how can we restore it?

This is where the concept of peer-to-peer (P2P) networks comes into play. P2P networks consist of different PCs connected to each other, referred to as nodes. These nodes can act as both servers and clients and do not pass through an intermediary server; it is decentralized.

When P2P network principles are applied to the blockchain, the ledgers created make copies on the nodes. Therefore, following the previous example, if the blockchain on one node is altered, the ledgers on the other nodes remain uncorrupted. When the majority consensus determines that the corrupted blockchain is the only one different, the altered blockchain is replaced with the original blockchain, making it an immutable ledger.

¹⁹https://en.wikipedia.org/wiki/Secure_Hash_Algorithms

2.4.3 Ethereum

In the realm of blockchain technology, Ethereum is a decentralized platform that not only allows us to build applications but also to deploy them in a decentralized manner (dApps) and execute smart contracts. As Ethereum is decentralized, it does not have a central authority to control the applications or their hosting. One way to host applications is by paying with Ether, which is a cryptocurrency used for buying, selling, and investing.

2.5 European Blockchain Services Infrastructure (EBSI)

EBSI is an initiative of the European Commission and European Blockchain²⁰. The focus of EBSI is to establish a cross-border infrastructure for digital public services, it is an infrastructure where not only public organizations but also private organizations can develop their applications connected to EBSI nodes. The EBSI platform was built based on current blockchain technology. The idea is to offer cross-border public services initially for European Union members, ensuring decentralized trust and compliance with EU regulations.

There are three fundamental pillars for understanding EBSI. Firstly, the business aspect, which EBSI refers to as use case families and domains. Here, the platform provides a framework to solve various real-world problems, such as the scenario of Verifiable Credentials. Secondly, the technology pillar consists of three fundamental components: APIs, Smart Contracts, and ledgers. APIs facilitate access from the internet to the platform, Smart Contracts act as intermediaries between the APIs and the ledger, which is a decentralized database where transactions are recorded. Finally, the infrastructure pillar involves the nodes of EBSI located throughout Europe, forming EBSI's pan-European network, this decentralized structure follows EBSI's governance rules in each node.

Although EBSI was built based on blockchain technology, EBSI itself is not a blockchain project. Additionally, EBSI does not require as much energy consumption as traditional blockchains because it uses a proof of authority mechanism²¹ with a limited number of actors.

As we saw earlier, EBSI benefits everyone who belongs to the European Union at the moment. It could be expanded around the world, but since EBSI is an initiative, when it is well-established, it can be expanded while also complying with EU regulations such as GDPR.

Finally, in the image below (see Figure 2.9), we can see the process of implementing EBSI with two universities for verifying the transcript records of both institutions:

²⁰<https://ec.europa.eu/digital-building-blocks/sites/display/EBSI>

²¹<https://www.coinhouse.com/what-is-proof-of-authority/>

- KU Leuven (1) issues her an Educational ID and Una Europa Alliance ID which she (2) stores in her personal digital wallet.
- Eva then applies for an Erasmus programme at the University of Bologna (IT) by (3) presenting her Education ID and Una Europa Alliance ID.
- The University of Bologna (4) verifies the authenticity of her credentials against the EBSI ledger and accepts her application.
- At the University of Bologna, she is now (5) granted access to educational environments via her Education ID and Alliance ID.
- Post-Erasmus, Eva is (6) issued a Transcript of Records from the University of Bologna.
- Returning to KU Leuven, she (7) presents her transcript, which is then (8) verified against the EBSI ledger and integrated into her academic curricula.

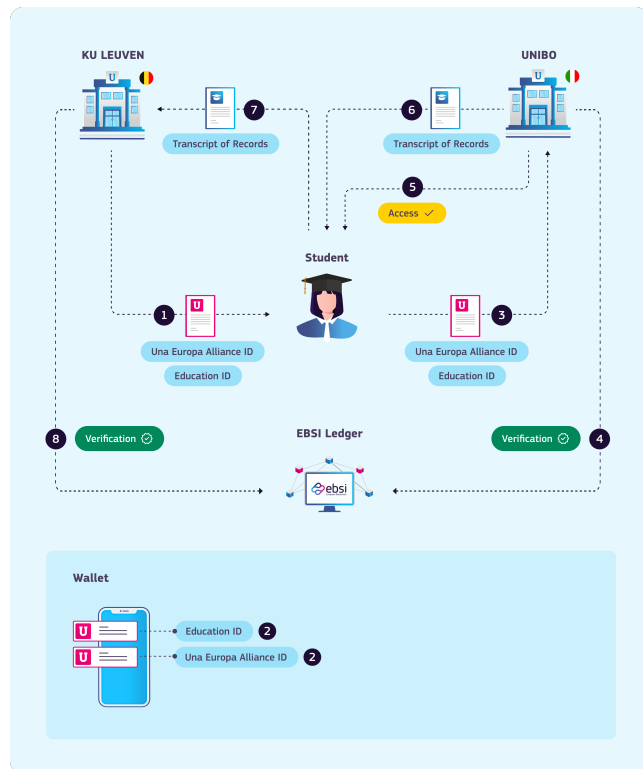


Figure 2.9: Access to Erasmus Programme and MOOC (Una Europa Alliance)²³

2.6 Data Spaces and Gaia-X

The governance of data, the reliability of data, and user sovereignty over their own data are not fully implemented yet. However, there are initiatives to address these issues, particularly in Europe. Strategies are being developed to ensure data sovereignty, facilitate reliable data exchange, and promote standardization.

In data spaces, users can share their information whenever they want, with whomever they choose, and under their own terms. Parties can connect to this data, but only with the user's explicit permission, ensuring controlled and secure data sharing. The infrastructure serves as the pillar of data spaces, focusing on the use of data, its governance, interoperability, and the protection of individuals. Understanding and emphasizing the rights and protection of individuals is crucial, along with fostering trust and the regularization in the data. [38]

One such initiative is Gaia-X, which aims to create a secure and transparent digital ecosystem where data and services can be made available, collated, and shared in a trustworthy environment.

Within Gaia-X, decentralized identifiers (DIDs) and verifiable credentials (VCs) are integral components of the SSI (Self-Sovereign Identity) ecosystem architecture. Gaia-X leverages decentralized identities, allowing for the use of various decentralized

²³<https://ec.europa.eu/digital-building-blocks/sites/display/EBSI/University+Alliances>

registries. It also incorporates trustworthy communication protocols, such as DID-Comm (Decentralized Identifier Communication) or OpenID Connect (OIDC), for secure agent-to-agent interactions. Additionally, Gaia-X utilizes the W3C's data model specifications to ensure that data is shared in a trustworthy manner, maintaining high reliability and regulatory compliance. [39]

2.7 Applications in the Domain of Currency Exchanges

Currently, there are many platforms that allow the exchange of bitcoin and various foreign currencies. These platforms offer exchanges with low commissions, and some of them do not require intermediaries. In this section, we will explore some of these platforms that facilitate currency exchange.

2.7.1 PayPal

PayPal is a platform where people can send or receive payments online. It is one of the most popular digital platforms, and users are not required to share their financial information with the recipient.

2.7.2 Hodl Hodl

This is a peer-to-peer (P2P) platform that allows users to trade cryptocurrencies with each other without intermediaries. Some characteristics of this platform are that users' funds do not need to stay in custody on the platform; instead, they are redirected to users' wallets. Additionally, users have privacy because they can choose to operate anonymously throughout the trading process. Furthermore, all communication between users is encrypted. The way to trust the operation is through smart contracts using multi-signature. ²⁴

The process that Hodl Hodl follows is as follows: firstly, the seller offers a bitcoin. These bitcoins do not need to be held in custody on the platform. Later, when buyer wants to purchase a bitcoin, the buyer signs a contract for the acquisition. Then, the seller is required to deposit the bitcoin into the unique escrow address that was created. Until the seller deposits the bitcoin, the buyer is required to deposit the money for the exchange. Once it is deposited, the seller needs to confirm that the deposit was made to release the bitcoin to the buyer, and finally, the transaction is completed.

2.8 Technologies

2.8.1 Veramo

When starting to create a decentralized application using W3C DIDs and W3C Verify credentials, it can often be difficult to follow these standards from scratch.

Veramo is a framework that utilizes W3C standardization to create DIDs and Verify credentials. It is also a multi-platform tool that works for NodeJs, ReactJs, and0 React Native. Additionally, it offers a command-line interface (CLI) for easy creation

²⁴<https://hodlhodl.com/>

of DIDs and Verify credentials. Veramo is extensible, meaning we can add different plugins or extensions according to our needs, and it operates as an open-source project with the Apache2 license

2.8.2 InterPlanetary File System (IPFS)

The data is increasing every day, not just by 1MB, but also by more than 1MB, this data is generally stored on the server managed by the provider, and we cannot access it directly. There are also other problems that can arise, for instance, the provider may no longer want to store the data, or, typically in Gmail, every free account has a limited space for saving data, when it exceeds the limit, we have to create another account or pay for a subscription to obtain more free space. Additionally, when the storage is managed by the provider of a platform or service, there may be times when we cannot access the platform because the server could be down.

The InterPlanetary File System²⁵ (IPFS) is a decentralized file system, utilizing a peer-to-peer network. It is similar to BitTorrent²⁶, where users can upload, download, or read content, it also employs hash functions to access content. As it is public and accessible to everyone, we can use encryption to ensure the content. Additionally, the IPFS is immutable, that means that the Content Identifier or CID generated are changed whenever the content change.

Unlike location-based addressing²⁷, which involves identifying and accessing data at a physical location in the network, throughout DNS²⁸ to access the IP where the server is located, content-based addressing²⁹ allows direct identification and access to data based on its content. This means we do not need to know the location, we can access content directly using the hash function generated by its content³⁰.

2.8.2.1 InterPlanetary Linked Data (IPLD)

Before introducing InterPlanetary Linked Data or IPLD within the IPFS ecosystem, it is essential to understand the concept of a Directed Acyclic Graph or DAG.

In a Directed Acyclic Graph (DAG), nodes are typically represented as circles. The term "directed" indicates that there is a defined direction for the edges, which are the links between nodes, this means that each edge has a starting point and an endpoint, indicating the direction of the relationship between nodes. Additionally, "acyclic" means that the graph does not contain any cycles or loops. In other words, it is impossible to traverse a sequence of edges and return to the starting point. Instead, the edges form a directed flow from one node to another, ensuring that the graph structure remains acyclic. Therefore, every edge in a DAG leads to a distinct endpoint, making it a useful data structure for representing relationships and dependencies without creating loops.

Within the DAG is the Merkle DAG, which combines the DAG structure with the

²⁵<https://docs.ipfs.tech/>

²⁶<https://www.bittorrent.com/>

²⁷<https://proto.school/content-addressing/02>

²⁸<https://www.cisco.com/c/en/us/support/docs/ip/domain-name-system-dns/12683-dns-descript.html>

²⁹<https://proto.school/content-addressing/03>

³⁰<https://proto.school/content-addressing/04>

State of the Art

Merkle tree. It consists of leaf nodes where the "blocks" or data, intermediate nodes that contain the hash or CID of the concatenation of the hashes CIDs of their child nodes, and a root node that represents the concatenation of all the nodes associated with it.

IPLD is a data format designed for decentralized networks in this case for IPFS, and it comprises the following components for go from data to data formats:

- **Links:** That represent relationships between different pieces of data. Each link consists of a Content Identifier or CID, which uniquely identifies a piece of data, and a name or label that indicates the type of relationship between the current data and the linked data. Links enable hierarchical and interconnected data structures, allowing for efficient traversal and access of related data across decentralized networks.
- **Data:** This refers to the content of the current document or data object. It can include any type of structured or unstructured data that needs to be stored, accessed, and shared within the IPLD ecosystem.
- **Data Model:** It defines a standardized way of representing data across different formats and protocols. It provides a unified schema for organizing and accessing data, allowing developers to work with data from various sources in a consistent manner. Additionally it is designed to be agnostic to specific data format, making easy the integration and interoperability between different data sources and applications.

The JSON snippet 2.1 illustrates IPLD's Data Model, featuring "Links" referencing CIDs of IPFS documents and a "Data" field containing the current document's content. This structure enables efficient organization and retrieval of decentralized data within the IPFS ecosystem.

```
{
  "Links": [
    {
      "Name": "File1",
      "Hash": "QmYN9f4cRGPreJDSi3YoFTt5eTVS2Jo9ePN3wH3TfgbB8u",
      "Size": 262158
    },
    {
      "Name": "File2",
      "Hash": "QmTJ1rwQQ7FC4HiwmxS1jFe2eJeb6kyxgRWKGyHjf7nYMN",
      "Size": 262158
    },
    {
      "Name": "File3",
      "Hash": "QmSEuztdUaJNLGhf3Hrpd9f8eHXftusY8QCbqUbzGv7LNx",
      "Size": 210174
    }
  ],
  "Data": "\u0008\u0002\u0018\u0010 \u0010\u000c"
}
```

Listing 2.1: Example JSON with IPLD Links

2.9. Platforms for Managing Decentralized Applications

2.8.2.2 InterPlanetary Name System (IPNS)

InterPlanetary Name System or IPNS is a component within the IPFS ecosystem that addresses the need for mutable data in decentralized networks. While Content Identifiers or CIDs in IPFS ensure the immutability of data, there are scenarios where data needs to be updated or changed over time. IPNS provides a solution to this challenge by allowing to create mutable pointers to data.

When content changes, IPNS creates a new CID for the updated version of the content and generates a cryptographic key pair. The public key is then associated with the new CID and published to the distributed network. This creates a pointer, known as an IPNS record, which maps the IPNS key to the latest CID of the content.

Through IPNS, users can access the most recent version of mutable data by resolving the IPNS key to its corresponding CID. This dynamic linking mechanism enables efficient management of mutable data within the IPFS ecosystem, ensuring that users can access the latest versions of content as it evolves over time.

2.8.3 Method IPID for DID

As discussed in the section on W3C Decentralized Identifiers (DIDs), establishing decentralized identities requires a standardized method. The W3C has introduced a method specifically designed for working with IPFS, known as the IPID method. This method provides a standardized approach for creating and managing DIDs within the IPFS ecosystem.

An example utilizing the did method is the following:

```
did:ipid:QmZqtqHudH8nbdTrLPgUXMzxBpZgEqB6Zr3LE8GHfT6NX9
```

where the `QmZqtqHudH8nbdTrLPgUXMzxBpZgEqB6Zr3LE8GHfT6NX9` represent the CID generated by the IPFS.

2.9 Platforms for Managing Decentralized Applications

When working with decentralized applications, there are many platforms to choose from. Although there are numerous options available, in this project, we will be working with two platforms that operate on IPFS as a base. However, these two platforms are similar; their approaches differ.

Pinata Cloud

Pinata Cloud³¹ is a cloud service focused on providing an intuitive user interface for storing files in the IPFS network. Additionally, this platform offers other services such as managing pins and access control. For manage the storage we will use this platform for have a better control.

³¹<https://www.pinata.cloud/>

2.10 7 Laws of Identity

The digital user is a person or entity that performs actions on the internet through devices. Over the years, people have learned to fill out forms or register for platforms to obtain valid credentials for browsing. However, digital identity lacks regulation and a standardized layer to address these issues. Users have become accustomed to providing information whenever requested by a webpage or platform, but this practice also presents various problems such as phishing. As digital identity becomes more pervasive, it is crucial for users to understand where they are connected.

Many companies offering internet platforms seek to manage user identities, but is the user who decides which platforms receive their information.

Both internet protocols and hardware systems have evolved over the years to support various applications without compromising usability. Similarly, digital identity must evolve to achieve acceptance on the internet. This is where the principles of the 7 laws of identity comes in as a framework.

1. **User Control and Consent**

This law refers to the user must have the control above their own digital identities, the user must be owners to manage their information and choose which information they can share, with whom and users should have the ability to provide or deny consent about their identity information accessed or used by others.

2. **Minimal Disclosure for Constrained Use**

To minimize risk, avoid storing unnecessary information for future use if it is not required for the current transaction.

3. **Justifiable Parties**

This law states that the digital systems should be limit to the parties with a justify needed for a user's personal information access. If parties want access to a user's information, they should be able to justify it.

4. **Directed Identity**

Identity systems should be bidirectional for system connectivity, allowing users to have control over their own information and to have different identities for different interactions, selecting them as needed.

5. **Pluralism of Operators and Technologies**

Users should have the option to choose from a variety of digital identity providers to access the platform. Identity management should be polycentric and polymorphic instead than being locked into a single proprietary system, as this promotes diversity among identity providers.

6. **Human Integration**

Humans are essential components in the system. The communication channel between the user and the system should be intuitive and easy to navigate. Therefore, the system should incorporate human involvement to create it user-friendly. By prioritizing human-centric design principles, digital identity systems, it improve usability and effectively meet the needs of their users.

7. **Consistent Experience Across Contexts**

To ensure a consistent experience across different contexts, digital identity sys-

tems should be provide a good user experience regardless of the platform, application, or service being used. Users should be able to utilize their digital identity across various platforms or select the most suitable identity based on context and specific requirements.

2.11 Terminology

In this section, we explain key terms and concepts relevant to decentralized identity and safe money exchange on the Expid platform. Knowing these words helps understand how the system works technically and what rules it follows.

- **Expid:** A Proof of Concept (PoC) that demonstrates decentralized identity, allowing user registration through asymmetric keys instead of centralized data. Users can generate keys, register with a public key, and create verifiable credentials for secure transactions.
- **Asymmetric Keys:** A cryptographic key pair used for encryption and decryption, where one key (public key) is used for encryption and the other key (private key) is used for decryption. In the context of Expid, users generate these keys to secure their identities.
- **Public Key:** The part of an asymmetric key pair that is shared publicly and used for encrypting information or verifying digital signatures. In Expid, users register with their public keys.
- **Private Key:** The part of an asymmetric key pair that is kept secret and used for decrypting information or creating digital signatures. Users must safeguard their private keys to maintain the security of their identities.
- **X.509 Certificate:** A standard format for public key certificates, which are used to verify the ownership of a public key. Expid allows users with X.509 certificates to register in the application.
- **Challenge File:** A file used in authentication processes that a user must sign with their private key to prove ownership of their public key. In Expid, challenge files are used during registration, login, transactions to verify user identity.
- **Digital Signature:** A cryptographic signature created using a private key, which can be verified with the corresponding public key. Digital signatures ensure the authenticity and integrity of a message or document.
- **Authentication:** The process of verifying the identity of a user or system. In Expid, authentication involves the use of DIDs, public keys, and digital signatures.
- **DID (Decentralized Identifier):** A type of identifier that enables verifiable, self-sovereign digital identities. DIDs are generated and managed in a decentralized manner, without a central registry, and are used to uniquely identify entities.
- **IPFS (InterPlanetary File System):** A protocol and peer-to-peer network for storing and sharing data in a distributed file system. It uses content-addressing to uniquely identify each file in a global namespace.

State of the Art

- **CID (Content Identifier):** A unique identifier used in IPFS (InterPlanetary File System) that points to a specific piece of content based on its hash. It ensures that content is retrieved exactly as it was stored, providing integrity and immutability.
- **DID Method Key:** A specific method used within a decentralized identity system to generate and manage keys. Method keys are integral to the creation and operation of DIDs. For instance: `did:key:ENCODING_CODE_PUBLIC_KEY`
- **Credential Issuer:** An entity that creates and issues verifiable credentials to individuals or entities. In Expid, the web application acts as the credential issuer for users.
- **Verifiable Credential:** A digital statement made by an issuer about a subject, which can be cryptographically verified. In Expid, verifiable credentials are used to validate users' identity claims and transaction details.
- **Verifiable Presentation:** A tamper-evident package of verifiable credentials shared by an individual to prove information about themselves. In Expid, users generate verifiable presentations to authenticate transactions and exchanges.
- **Peer-to-Peer (P2P):** A decentralized network model where each participant (peer) can act as both a client and a server. In the context of Expid, currency exchanges are conducted directly between users in a P2P manner.

Chapter 3

Design and Implementation

3.1 Methodology

The Expid application development followed a systematic methodology encompassing four key phases: investigation, design, implementation, and validation. The initial investigation phase involved extensive research into decentralized identity (DID) and verifiable credentials (VCs). This research included reviewing existing literature, industry reports, and relevant standards to understand current practices and emerging technologies in these fields.

Following the investigation phase, the design process centered on creating an intuitive user experience through user interface (UI) mockups. These mockups visualized the workflow of the Expid application, detailing steps such as registration, authentication, currency exchange, and contract processes. Furthermore, data models and schemas were developed to accurately represent user profiles, credentials, and currency exchange transactions, ensuring a comprehensive and coherent design for the application's functionality.

The implementation phase turned the design ideas into real parts of the application. Using ReactJS and Material UI, the frontend was built to match the UI mockups. At the same time, the backend was created with TypeScript, using PostgreSQL for user credentials and Firebase Realtime Database for managing other data. Tools like Veramo (for creating and checking credentials), Pinata (for pinning the CID created on IPFS), and Render (for hosting the server, agent and front-end application) were added to improve the app's functionality.

Furthermore, a proof-of-concept (PoC) approach was adopted to validate the feasibility and effectiveness of the proposed solution. The PoC involved developing a prototype platform for currency exchange, utilizing decentralized identities (DIDs) and verifiable credentials (VCs) to ensure the authenticity and security of user interactions. This prototype was rigorously tested and evaluated to assess its performance, usability, and compliance with functional and non-functional requirements, as well as the Kim Cameron's seven laws of identity.

3.2 Proof of Concept Design

The design of the proof-of-concept (PoC) for the Expid platform focuses on demonstrating the feasibility and functionality of decentralized identity management and secure currency exchange between individuals. The PoC will illustrate the core features and capabilities of the Expid system, validating that it meets the specified requirements.

3.2.1 Functional and Non-Functional Requirements

In order to design and implement a robust proof-of-concept (PoC), the functional and non-functional requirements are described in this sub section.

Functional Requirements:

- **R1.** Users should be able to generate their key pairs within the Expid web application.
- **R2.** Users with X.509 certificates should be able to register in the Expid web application.
- **R3.** Registration must only require the user's public key and signature, without requesting the private key.
- **R4.** The Expid web application must ensure that users are who they claim to be during interactions.
- **R5.** The Expid web application must create a verifiable credential for users when they register a new monetary exchange or wish to create a contract after verifying the user's identity claims.
- **R6.** Users should be able to verify the authenticity of the provided information by scanning the QR code generated by the other party.

Non-functional Requirements:

- **NFR1.** The system must ensure the secure generation of asymmetric key pairs on the client site. The Expid web application must use robust cryptographic algorithms to protect user data and transactions. Private keys must never keep in the Expid web application.
- **NFR2.** The application must adhere to privacy standards, ensuring that user data is not shared without consent. Only necessary information should be included in verifiable credentials.
- **NFR3.** The web application must have an intuitive and user-friendly interface. Key generation, registration, and credential verification processes should be easy for users.
- **NFR4.** Verifiable credentials must be compatible with w3c did standards.
- **NFR5.** The application must be reliable, ensuring that user data and credentials are consistently available and accurate. The system should handle failures, maintaining data integrity and user trust.
- **NFR6.** The system must comply with relevant regulations and standards for data protection and cryptographic security.

3.2.2 Mockup Design

Before delving into the development of the Expid web application, a mockup was created using Draw.io¹. This mockup served as an initial blueprint to define the structure, layout, and core functionality of the web application. The primary goal was to map out the user interface and ensure that all essential elements were appropriately placed for an intuitive and user-friendly experience.

The illustration below shows the Home page section of the Expid web application. It includes registration and login buttons. The arrows indicate the next steps in the registration process. Users will find forms to fill in their details, generate or upload key pairs, and complete the registration process.

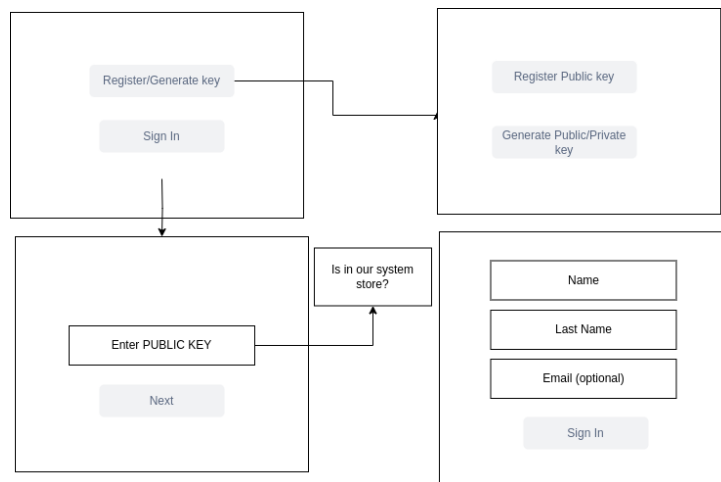


Figure 3.1: Home Page and Registration Page

The following mockup illustrates the sections of the Expid web application designed for currency exchange transactions. It includes areas for sellers to offer their currencies and for buyers to connect with these sellers. Additionally, it outlines the process for generating verifiable credentials via QR codes.

¹<https://www.drawio.com/>

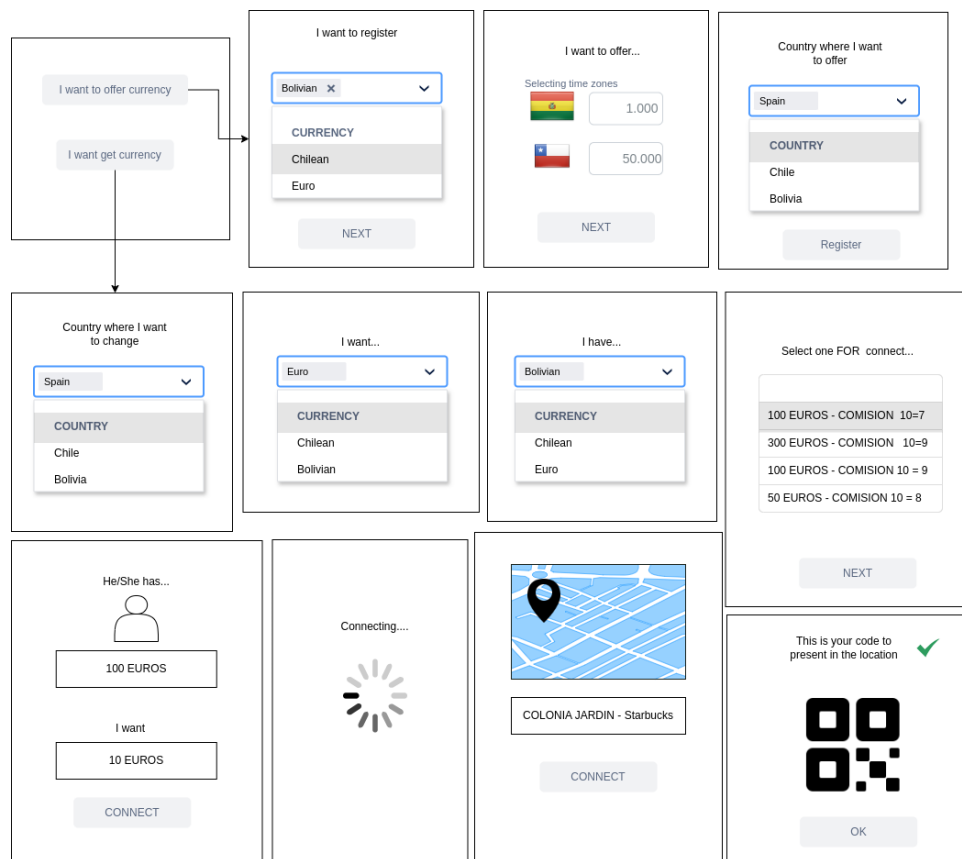


Figure 3.2: Currency Exchange and Verification Pages

3.2.3 Data Storage

This sub section details how the data used within the Expid application will be stored and managed. Given Expid's focus on decentralized identities and peer-to-peer transactions, a NoSQL database like Firebase presents a suitable choice due to its flexibility and scalability.

3.2.3.1 Collections

Firestore utilizes a document-oriented approach. Each document represents an entity such as user, currency and contract, each collection contains key-value pairs for its attributes. To organize this data effectively, we will establish three primary collections:

1. **Users:** This collection maintains user profiles, including decentralized identifiers (DIDs), credentials, and personal information. Each user is identified by their DID.
2. **Currency Exchanges:** The currency exchanges collection records details of currency exchange offers, including the currencies involved, their values, and the associated DID for the subject.
3. **Contract Currencies:** This collection stores details of currency exchange contracts between buyers and sellers. Each contract includes information about the

Design and Implementation

currencies being exchanged, the parties involved, and the details of the transaction.

3.2.3.2 Data Structure and Data Types

The significance of each field in every collection is as follows:

Users:

- **cid:** The content identifier for the DID document created on IPFS.
- **did:** Decentralized Identifier (DID).
- **document_id:** The document for the user such as Spain DNI.
- **email:** The user's email address.
- **last_name:** The user's last name.
- **name:** The user's first name.
- **public_key:** An array containing the user's public keys in base64 encoding.

Currency Exchanges:

- **country_of_exchange:** The country where the exchange took place.
- **currency_cost:** The currency offered (e.g., "CLP - Chilean Peso").
- **currency_cost_value:** The amount of the offered currency.
- **currency_sell:** The desired currency received (e.g., "USD - US Dollar").
- **currency_sell_value:** The amount of the desired currency received.
- **did_subject:** Decentralized Identifier (DID) involved in the transaction (the seller DID).
- **id_credential:** Identifier of the credential made for this transaction (ID of the Verifiable Credential).

Contract Currencies:

- **country_of_exchange:** The country where the exchange is taking place.
- **currency_cost:** The currency being offered (e.g., "BOB - Bolivian Boliviano").
- **currency_cost_value:** The amount of the offered currency.
- **currency_sell:** The desired currency to receive (e.g., "USD - US Dollar").
- **currency_sell_value:** The amount of the desired currency.
- **did_buyer:** Decentralized Identifier (DID) of the buyer.
- **did_seller:** Decentralized Identifier (DID) of the seller.
- **id_credential_buyer:** Identifier of the buyer's credential (likely a Verifiable Credential).
- **id_credential_seller:** Identifier of the seller's credential (likely a Verifiable Credential).

- **location:** An array containing latitude and longitude coordinates for the exchange location.
- **message_to_seller:** A message from the buyer to the seller.
- **time:** The time of the exchange.

Firestore offers various data types, including strings, numbers, booleans, arrays, and objects. We will map the data types for each collection.

| users | |
|-------|------------------------------|
| Key | UniqueID |
| | cid (String) |
| | did (String) |
| | document_id (String) |
| | email (String) |
| | last_name (String) |
| | name (String) |
| | public_key (Array of String) |

| contract_currencies | |
|---------------------|-------------------------------|
| Key | UniqueID |
| | country_of_exchange (String) |
| | currency_cost (String) |
| | currency_cost_value (String) |
| | currency_sell (String) |
| | currency_sell_value (String) |
| | did_buyer (String) |
| | did_seller (String) |
| | id_credential_buyer (String) |
| | id_credential_seller (String) |
| | location (Array of Numbers) |
| | message_to_seller (String) |
| | time (String) |

| currency_exchanges | |
|--------------------|------------------------------|
| Key | UniqueID |
| | country_of_exchange (String) |
| | currency_cost (String) |
| | currency_cost_value (String) |
| | currency_sell (String) |
| | currency_sell_value (String) |
| | did (String) |
| | id_credential (String) |

Figure 3.3: Data Types of the Collections

3.2.4 Workflow Overview

User Registration:

- Users can optionally generate their asymmetric key pairs within the Expid web application.
- Registration is completed using the public key, without exposing the private key.
- The system verifies the user's identity and creates a DID subject.

Credential Issuance:

- Users can request and receive verifiable credentials for specific actions, such as currency exchanges.
- The backend agent verifies the user's identity and issues the credential.

Currency Exchange:

- Users can offer and search for currency exchange opportunities.
- Transactions are validated through verifiable credentials and decentralized identity verification.

Verification Process:

- Users can generate and scan QR codes to verify the authenticity of credentials.

Design and Implementation

- The system ensures that all interactions are securely validated.

3.2.5 Sign Up and Sign In Workflows

As mentioned earlier in the Authentication section, WebAuthn is a relatively new protocol that eliminates the need for passwords, hence it is referred to as passwordless. However, one of the challenges it faces is the dependency on the device storing the private keys. For instance, if a user stores their private keys on a mobile device and loses it, they lose access to their authentication credentials. To address this issue, in the subsequent development of the use case, another alternative is considered. So the application will allow users to generate asymmetric keys or upload existing keys within the DID Document.

In the following sequence diagrams, Figures 3.4 and 3.5, we illustrate the process of signing up and logging in to a web application for authentication. These diagrams provide a visual representation of the interactions between the user, the Expid web client, the Expid server-side, the IPFS as a (decentralized file system storage) and Database during both the sign up and login processes.

Additionally, Expid application employs two layers of security for user data. First, all content generated by both the server and client is encrypted with the Expid application's private key. This encryption protects user data even when stored on IPFS, a (decentralized file system storage) solution where anyone can retrieve the content itself (using the CID). Second, the challenge code used in the authentication process includes a timestamp, ensuring the validity and timeliness of the process.

The process of the sign up is the following:

- **Initiate Sign up (Optional Key Generation):** The user starts the sign up process through the Expid web application. They can choose to generate a new asymmetric key pair (public and private key) or skip this step.
- **Key Generation (Optional):** If the user chooses to generate keys, the Expid web client creates the key pair. **Important:** The private key is never stored on the Expid server or client for security reasons. Only the public key is used further in the process.
- **Account Information and Public Key Upload:** The user provides their account information like name, last name (email is optional) and a required Document ID. They upload their public key, either the generated one or an existing X.509 formatted key.
- **Challenge for Verification:** The Expid server generates a unique challenge code similar to WebAuthn.
- **Signing the Challenge:** The user receives the challenge code and uses their private key (not uploaded) to sign it. This proves they own the corresponding public key.
- **Uploading Signed Challenge:** The user uploads the signed challenge file back to the Expid web client.
- **Signature Validation:** The Expid server receives the signed challenge and verifies the signature using the provided public key.

3.2. Proof of Concept Design

- **DID Subject Creation (Success):** If the signature is valid, the server creates a DID Subject (identifier) based on the public key, following W3C DID specifications (did:key method).

- **DID Document Creation:** A DID Document is created for later authentication. It stores only the DID Subject and the public key (the authentication key).

- **DID Document Storage:** The DID Document is stored in a (decentralized file system storage) solution called IPFS, identified by a Content Identifier (CID).

- **User Information and CID Storage:** The user's additional information (name, etc.) and the CID are stored in the Expid database.

- **Signup Completion:** Upon successful account creation, the user receives their DID Subject, which they can use for login.

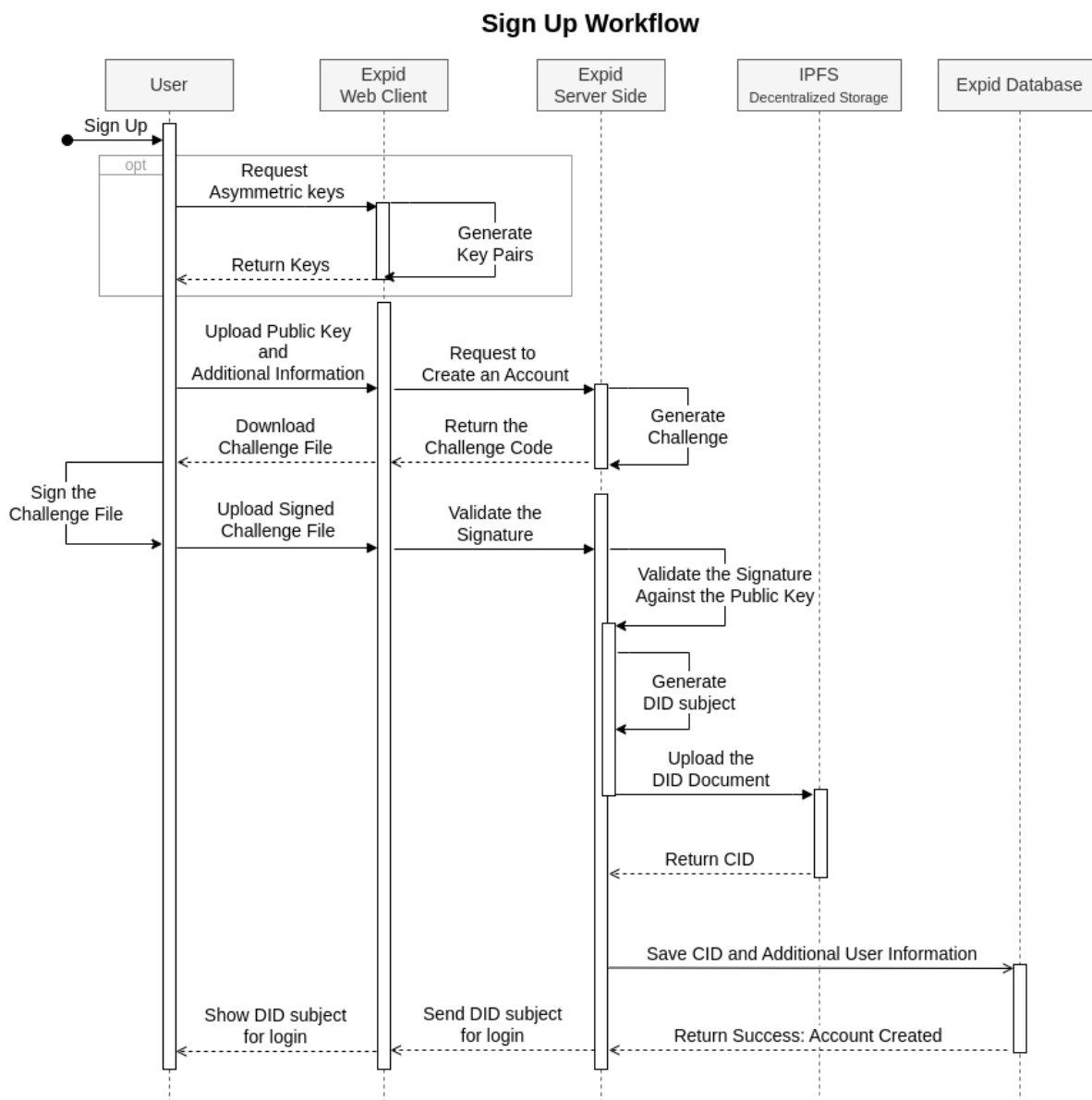


Figure 3.4: Sign Up Process Sequence Diagram

The process of the Sign In is the following:

- **Login Initiation:** The user starts the login process on the Expid web application. They provide either their public key or DID Subject obtained during sign up.
- **User Registration Check:** The Expid application verifies if the user's public key or DID Subject is registered in the system.
- **Challenge Generation (If Valid User):** If the user is registered, the web application generates a unique challenge code.
- **Signing the Challenge:** The challenge code is presented to the user. They use their private key to sign it, proving they own the corresponding public key.

3.2. Proof of Concept Design

- **Uploading Signed Challenge:** The user uploads the signed challenge code back to the Expid web application.
- **Challenge Verification:** The Expid web application forwards the signed challenge to the Expid server for verification.
- **DID Document Retrieval (Server-Side):** The Expid server retrieves the DID Document associated with the user's DID Subject. It likely retrieves the CID (Content Identifier) from the Expid database and uses it to access the DID Document stored in IPFS (decentralized file system storage).
- **Signature Validation:** The server uses the public key retrieved from the DID Document's "Authentication" section to validate the user's signature on the challenge code.
- **Login Success (If Valid Signature):** If the signature is valid, the server confirms successful login and allows the user to access the Expid web application.
- **Login Failure (If Invalid Signature):** If the signature validation fails, the login attempt is denied, and the user may receive an error message.

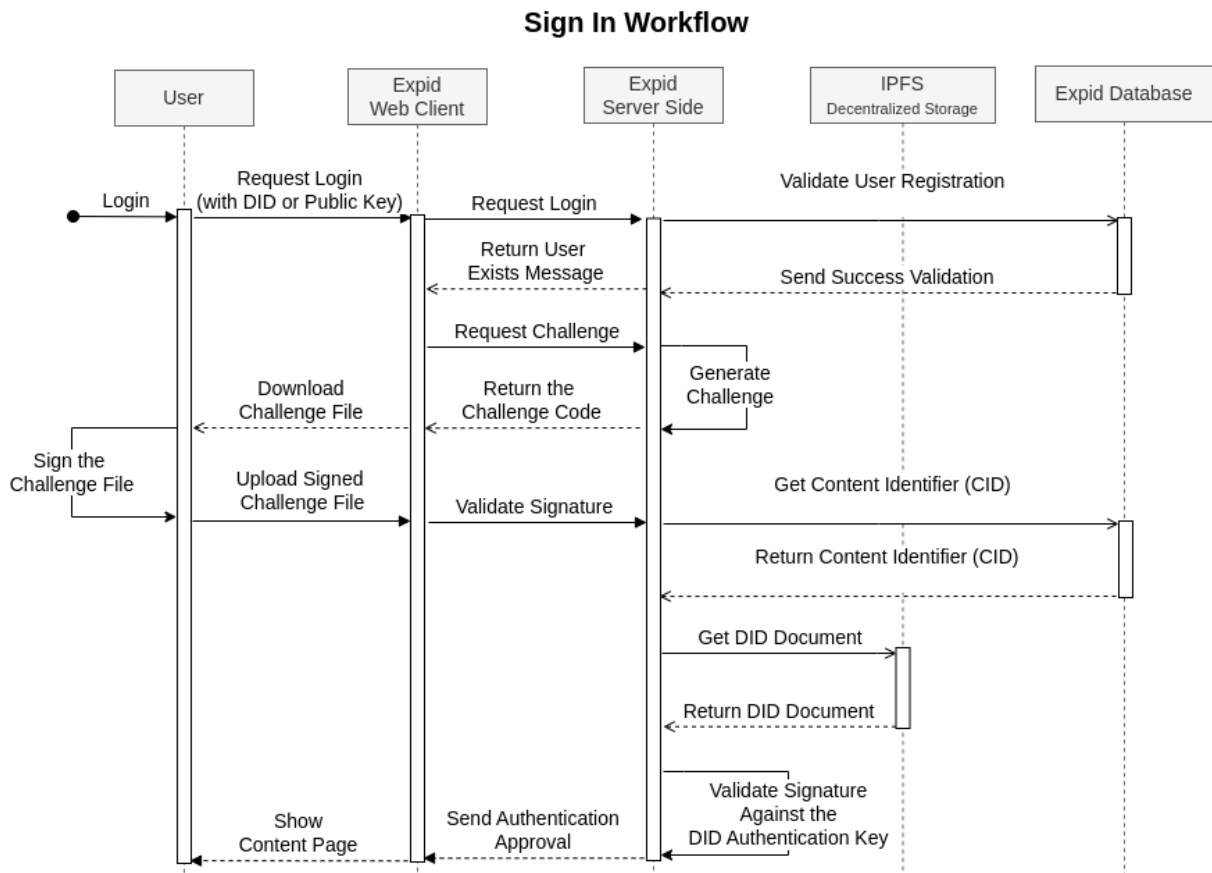


Figure 3.5: Login Process Sequence Diagram

3.3 Implementation

3.3.1 Technological Choice

In the development of the Expid platform, careful consideration was given to technological choices to ensure robustness, security, and scalability across all components of the system:

Front-End:

- **React.js:** A JavaScript library for building user interfaces, known for its component-based architecture and declarative syntax. ⁽²⁾
- **MaterialUI:** A popular React component library that provides pre-designed UI components following Google's Material Design principles, enhancing the visual aesthetics and usability of the application. ⁽³⁾

Back-End:

- **TypeScript:** A superset of JavaScript that adds static typing and other advanced features, improving code maintainability and scalability. ⁽⁴⁾

Databases:

- **PostgreSQL:** An open-source relational database management system (RDBMS) known for its robustness, scalability, and support for complex queries and transactions. ⁽⁵⁾
- **Firebase Realtime Database:** A cloud-hosted NoSQL database that allows for real-time data synchronization across clients, enabling efficient storage and retrieval of application data. ⁽⁶⁾

Other Tools and Technologies:

- **OnRender:** A cloud platform for hosting server-side applications, providing scalable infrastructure and deployment services. ⁽⁷⁾
- **Docker:** A containerization platform in this case used to package the back-end and front-end components of the application into portable and isolated containers. ⁽⁸⁾
- **IPFS (InterPlanetary File System):** A distributed file system for storing and sharing data in a peer-to-peer network. ⁽⁹⁾
- **Pinata:** This platform provides easy storage and retrieval of media files on IPFS through its PINES feature. ⁽¹⁰⁾

²<https://react.dev/>

³<https://mui.com/>

⁴<https://www.typescriptlang.org/>

⁵<https://www.postgresql.org/>

⁶<https://firebase.google.com/docs/database>

⁷<https://render.com/>

⁸<https://www.docker.com/>

⁹<https://ipfs.tech/>

¹⁰<https://www.pinata.cloud/>

- **Veramo:** A framework for building decentralized identity solutions, offering tools and APIs for managing DIDs, verifiable credentials. ⁽¹¹⁾
- **Forge:** A cryptographic library used for encrypting and decrypting data using the AES (Advanced Encryption Standard) algorithm, ensuring the confidentiality and integrity of sensitive information. ⁽¹²⁾
- **MapBox:** Is an API and SDK that is implemented to retrieve the locations such as google maps, it is utilizing in the Expid application to choice the location to make the exchange currency. ⁽¹³⁾

3.3.2 Repositories and Application

The following resources provide access to the various components of the Expid platform demo, including the code repositories and deployed services.

Demo Video: <https://www.youtube.com/watch?v=59SDMaiw56g>

GitHub Repository:

- **Expid Monorepo:** This repository contains both the backend and frontend code for the Expid platform, organized as a monorepo. Detailed instructions for setting up and running the project can also be found in the repository.

Url: <https://github.com/alivane/TFM-decentralize-identity/>

Agent for Creating Verifiable Credentials:

- **Expid Agent:** This is the deployed agent that creates verifiable credentials and verify its.

Url: <https://expid-veramo-agent.onrender.com>

- **GitHub Repository of the Agent:** The code for the agent is available in this repository. It was created using the Veramo Agent Deploy project deployed on Heroku. For the Expid application, the agent is deployed on Render.

Url: <https://github.com/alivane/expid-veramo-agent/>

Expid Application URLs:

- **FrontEnd Application:** This is the URL for the deployed frontend of the Expid application.

Url: <https://tfm-frontend.onrender.com/>

- **BackEnd Application:** This is the URL for the deployed backend of the Expid application.

Url: <https://tfm-decentralize-identity.onrender.com/>

¹¹<https://veramo.io/>

¹²<https://digitalbazaar.github.io/forge/>

¹³<https://www.mapbox.com/>

3.3.3 Problems encountered

In this section, we address the challenges and obstacles faced during the development and implementation of the Expid platform. By identifying and discussing these problems, we aim to provide insights into the complexities of the project and the strategies employed to overcome them.

The large size of the message to encrypt:

- **Problem:** The large size of the message to encrypt poses a challenge when the data exceeds the capacity of traditional asymmetric encryption methods, such as RSA, which are suitable for smaller data sizes. When attempting to encrypt large messages using RSA, the encryption process becomes inefficient and impractical due to the limitations of the public key size.
- **Solution:** To address this issue, a hybrid encryption approach inspired by a solution from Stack Overflow¹⁴ was employed:
 - A random symmetric key K is chosen (a raw sequence of, e.g., 128 to 256 random bits).
 - The big message is symmetrically encrypted with K, using a proper and efficient symmetric encryption scheme such as AES.
 - K is asymmetrically encrypted with RSA.

DID Method in Draft:

- **Problem:** Initially, the Expid application was designed to create DID documents using the IPID method, which is currently in draft standard development. Implementing the IPID method requires signing by the private key located on the user's own server. However, this method's reliance on each user having a server to manage their DID is impractical, as it demands a low-level server setup that most users cannot provide.
- **Solution:** To address this issue, we switched to using the DID KEY method. The DID method generates a unique DID subject using the public key encoded, eliminating the need for users to maintain their own servers. Additionally, the DID KEY method¹⁵ has more comprehensive documentation, making it easier to implement and ensuring better support and understanding the developing.

¹⁴<https://stackoverflow.com/questions/5866129/rsa-encryption-problem-size-of-payload-data>

¹⁵<https://w3c-ccg.github.io/did-method-key/rsa>

Chapter 4

Evaluation and Conclusion

This chapter presents the evaluation and conclusions from the development and implementation of the Expid application. It is organized into two main sections: the evaluation of the proof-of-concept (PoC) and the subsequent conclusions, including the main contributions and future work.

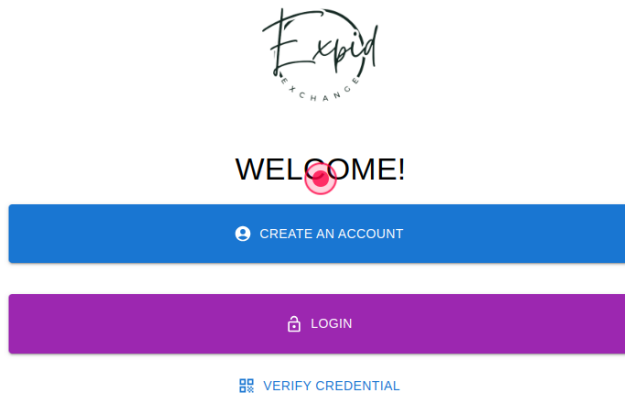
4.1 Evaluation

The evaluation will be conducted through a detailed demo to verify that all specified requirements are met. The demo will showcase key functionalities such as user registration, authentication, digital signature, verifiable credentials and currency exchange transactions. Additionally, we will assess whether the Expid platform adheres to Cameron's Laws of Identity, ensuring that the principles of secure and decentralized identity management are maintained.

4.1.1 Description of the Demo Steps

To validate the effectiveness and functionality of the Expid proof-of-concept (PoC), a structured demonstration was conducted. This section outlines the demo steps taken to ensure that Expid meets its objectives and aligns with the requirements. The demo involves key processes such as user registration, credential issuance, and verification. Each step is designed to showcase how Expid operates in real-world scenarios.

4.1.1.1 Home Overview



The home page of the Expid web application serves as the central hub for users to access key functionalities. The primary features available on the home page include buttons for creating an account, logging in, and scanning a QR code to verify credentials, as shown in the figure 4.1.

Figure 4.1: Home Page

When users click the "Create an Account" button, they are redirected to the registration page where they can generate their key pairs and register using their public key. Conversely, when users click the "Login" button, they are taken to the login page, where they can enter their credentials to access their account, using either their DID subject or public key.

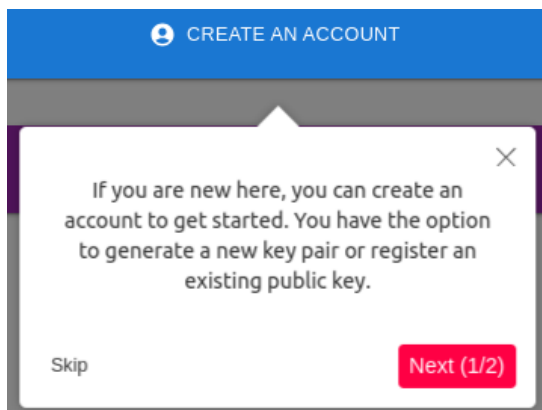


Figure 4.2: Create an Account Instruction

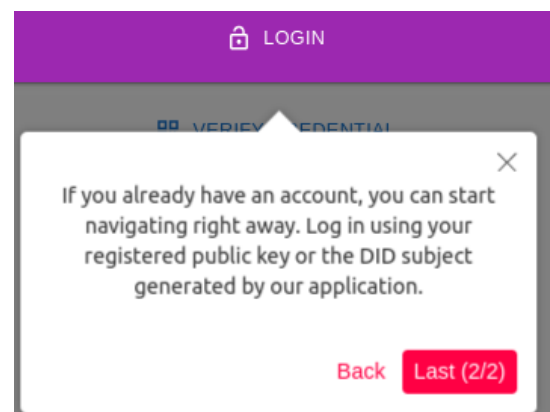


Figure 4.3: Login Instruction

4.1.1.2 Sign Up Workflow

The Sign Up workflow in the Expid web application consists of a series of steps designed to securely register a new user. Below is a detailed description of each step:

Step 1: Fill in the Form

In the first step of the Sign up workflow, users are required to provide some basic information by filling in a form. The form asks for the user's first name, last name, and an optional email address. Additionally, the user must provide a valid document ID. This information is used to create an initial user profile and to associate the decentralized identity (DID). The inclusion of an email address, though optional, can be useful for account recovery and communication purposes.

Expid
EXCHANGE

Sign Up

First Name* Demo Last Name* TFM

Email Address

Document ID* 123456 i.g., Passport, Driver's License, National ID

NEXT

Already have an account? [Sign In](#)

1 Form 2 Upload Public Key 3 Download File and Verify Sign 4 Assign credentials

Figure 4.4: Sign up, step 1. Fill in the Form

Expid
EXCHANGE

Sign Up

(OPTIONAL) GENERATE KEY PAIR

OR

(REQUIRED) UPLOAD PUBLIC KEY

Already have an account? [Sign In](#)

1 Form 2 Upload Public Key 3 Download File and Verify Sign 4 Assign credentials

Figure 4.5: Sign up, step 2. Generate Asymmetric Key or Upload Public Key

Step 2: Generate Asymmetric Key or Upload Public Key

Once the form is filled out, users proceed to the second step, which involves key management. Users have the option to generate a new pair of asymmetric keys (public and private keys) directly within the Expid application. This can be done by clicking a button that initiates the key generation process, after which the keys are downloaded to the user's device. When users already possess a public key X.509, they can upload it instead. Uploading an existing public key X.509 is mandatory to the users.

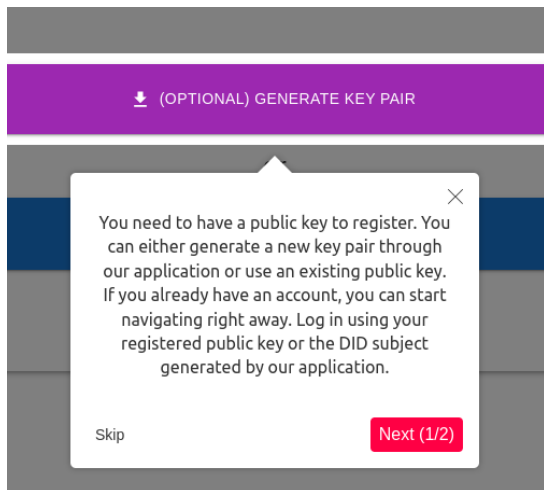


Figure 4.6: Download asymmetric keys generated in the Expid client site.

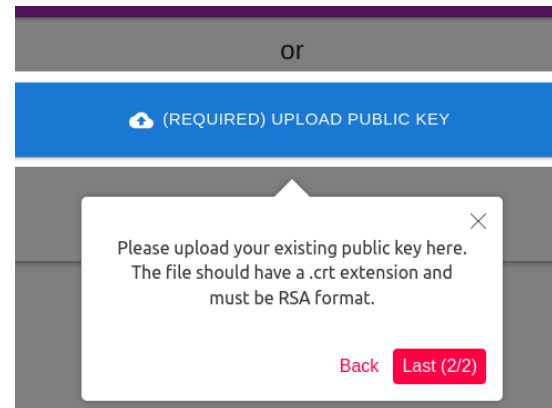


Figure 4.7: Upload an existent public key in X.509 format.

Step 2.1: Generate Asymmetric Key (Optional)

Due that Generate Asymmetric Key is optional, the user can click to the button to generate a pair of asymmetric keys (public and private keys) and the generated keys are downloaded to the user's device.

Asymmetric keys are saved with the public key having a (.crt) extension and the private key having a (.pem) extension.

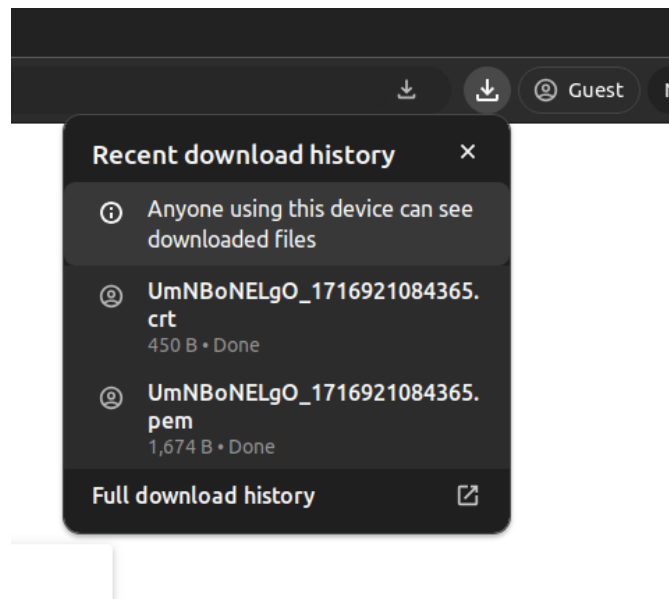


Figure 4.8: Asymmetric keys saved.

Step 2.2: Upload Public Key (Required)

To ensure the security and integrity of the registration process in the Expid web application, users are required to confirm their public key file before uploading it. This step involves selecting the appropriate public key file in X.509 format from their device and verifying that it is the correct file to be associated with their decentralized identity as shows in the figure 4.9.

Evaluation and Conclusion

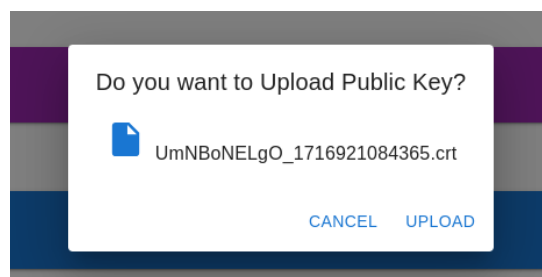
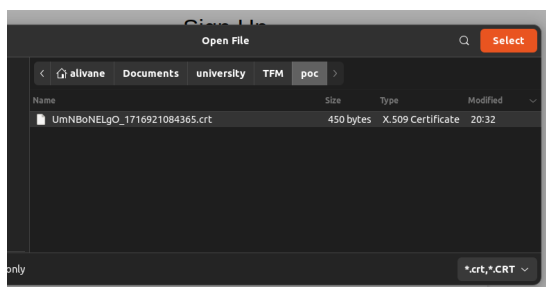


Figure 4.9: (A). Selecting the public key file from the user's device for uploading. (B). Confirming the public key file to upload in the Expid web application.

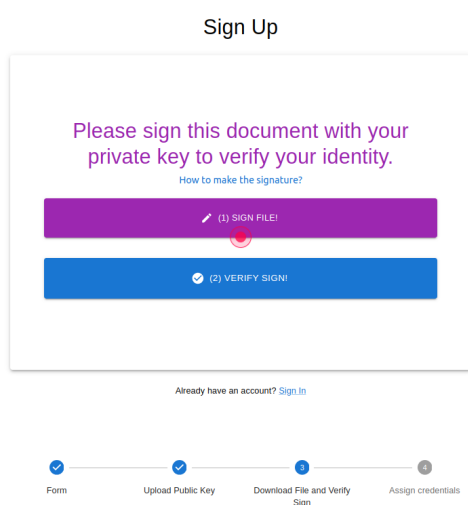


Figure 4.10: Download Challenge File and Sign It

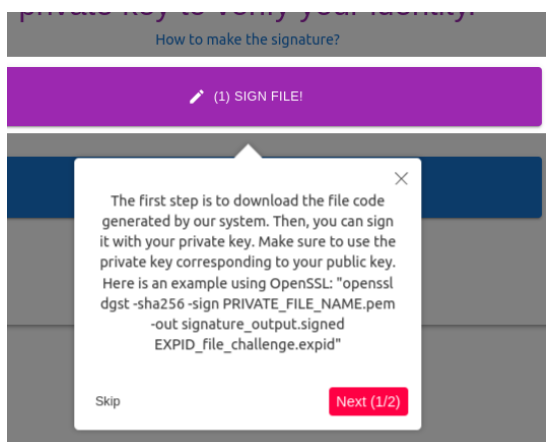


Figure 4.11: Download the challenge file from the Expid client site.

Step 3: Download Challenge File and Sign It

In the third step, the user must authenticate their control over the private key associated with their public key. This is done by downloading a challenge file from the Expid application. The user is required to sign this challenge file using their private key. After signing the challenge file, the user uploads the generated signature back to the Expid application. This step verifies that the user indeed possesses the private key corresponding to the submitted public key, ensuring the authenticity of the identity being registered.

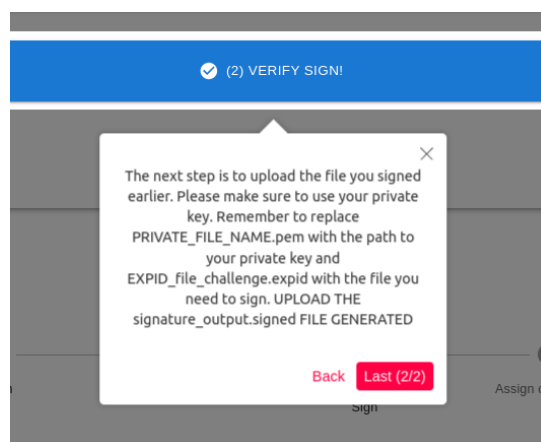


Figure 4.12: Upload the signed challenge.

Step 3.1: Download Challenge File (Required)

The user simply needs to click a designated button within the application interface, which triggers the download of a unique challenge file to their device. This file will later be used to create a digital signature, demonstrating that the user indeed possesses the corresponding private key.

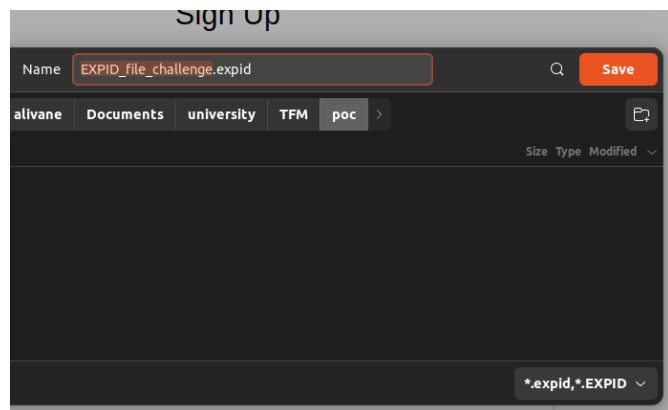


Figure 4.13: The challenge file is downloaded with the extension (.expid) to identify it.

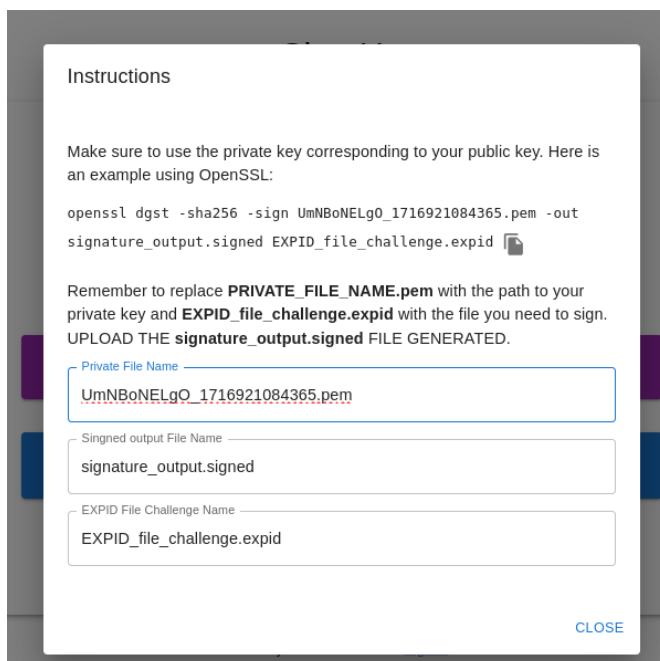


Figure 4.14: How to make the signature? (Optional)

Step 3.2: How to make the signature? (Optional)

When users click in "How to make the signature?" button, detailed instructions are provided, guiding users on how to use the OpenSSL command line interface (CLI) to sign the file. The instructions include steps to replace file names and execute the necessary commands.

Step 3.3: Sign the Challenge File (Required)

Users must execute a command generated in the "How to make the signature?" instructions provided by the Expid web application. Typically, users copy the command line from the instructions and paste it into their command line interface (CLI) or terminal to execute the signing process, as shown in the figure 4.15.

```
alivane@ali:~/Documents/university/TFM/poc$ openssl dgst -sha256 -sign UmNB0NELgO_1716921084365.pem -out signature_output.signed EXPID_file_challenge.expid
alivane@ali:~/Documents/university/TFM/poc$ ls
EXPID_file_challenge.expid  UmNB0NELgO_1716921084365.crt
signature_output.signed    UmNB0NELgO_1716921084365.pem
alivane@ali:~/Documents/university/TFM/poc$
```

Figure 4.15: Pasting the generated command line into the terminal.

Step 3.4: Upload the Signature (Required)

The user completes the process by uploading the signature file generated in the previous step. This signature file is produced after executing the command line using OpenSSL, as instructed in the preceding step.

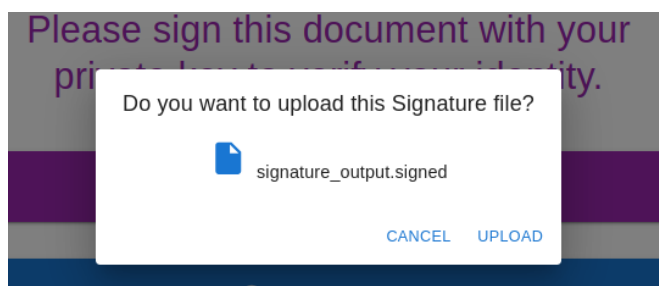


Figure 4.16: Uploading the signature file generated using the OpenSSL command line.

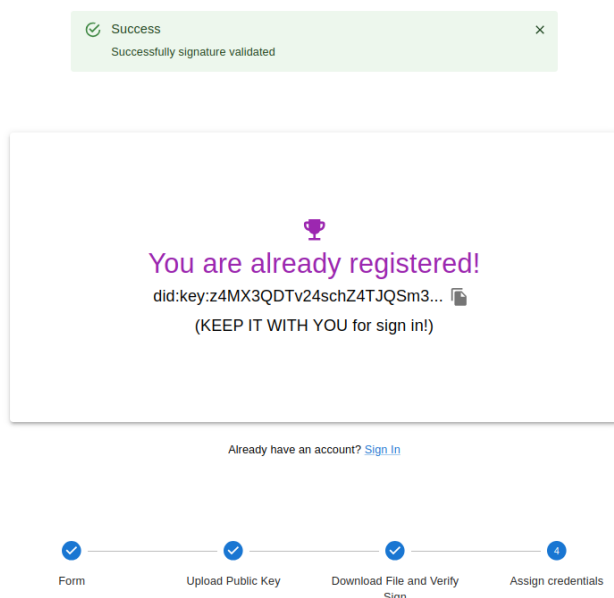


Figure 4.17: Step 4: Validation and DID Subject Creation

Step 4: Validation and DID Subject Creation

The final step involves the Expid application validating the uploaded signature. If the signature is successfully validated, the application proceeds to create a decentralized identifier (DID) subject for the user. This DID subject is a unique identifier that represents the user's identity within the decentralized system. Upon successful creation, the application returns a response to the user, confirming the successful registration and providing the DID subject.

Additionally, a "Copy" button is made available to allow users to easily copy their new DID subject for future use. Did generated by Expid web application is the following:

```
did:key:z4MX3QDTv24schZ4TJQSm3yLJR267BswCGSA57kP6DAki8PJZpse6AVRcbmt d
qZrRMxBBedwLdQdubViADzuUU3ws2TSRmgvJqCi7NTxrcKWUXURCBSutBxEc1gCaQbUg
S39BMrYZucN6VbGsUntTnNX1qMqbLrDy9DPkATEfRRYfN2kyC8u1pbPqg1oLfdT7xjnjf
```

8NUoeovqNE juL8DRXFCd5Z5x4SHupTv4fUX3ykRv7enxTuGb33WwPtLQ27U6k88mRZfui sdi7YfGrekG4dQ12FaTXUdkREKCCwFYsGdwcxt1TUfZyLQsJgZ6fRdLuK8G2RUeJxUdpu 6NXBhiMnj6d9Xaa je61WJpS33EnXqba2o7hhQbka72y4F9HN2iV7A81Z5UMHqJnUwk3WqE

4.1.1.3 Login Workflow

The login process in the Expid web application involves several steps to securely authenticate users.

Step 1: Authentication Options (Required)

Users have the option to log in using either their DID subject created during registration or their public key.

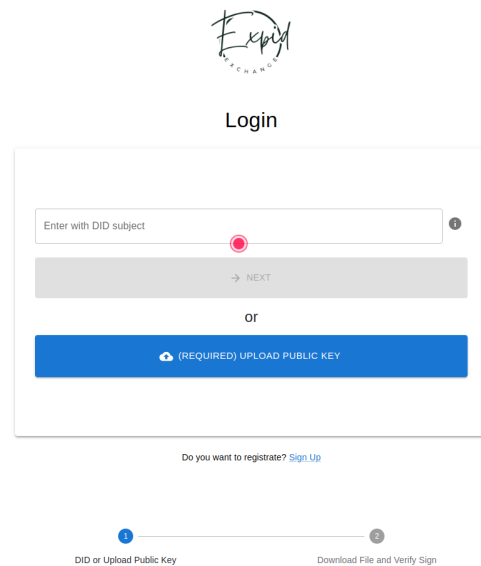
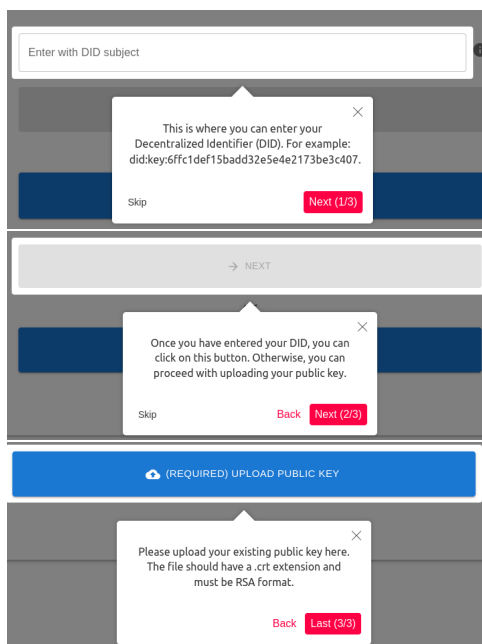


Figure 4.18: Authentication Options.



Step 1.1: Instructions for Authentication Options.

When a user enters their DID subject, the "Next" button becomes available, allowing them to proceed with the authentication process. However, if the DID subject field is left empty, indicating the user has not provided this information, the "Next" button remains disabled. In this case, users have the option to choose an alternative method for authentication by selecting the option to enter their public key, as illustrated in Figure.

Figure 4.19: Instructions for Authentication Options.

Step 2: Challenge File Signature.

Upon selecting the desired authentication method, users proceed to download a challenge file. This file serves as a means of verifying the user's identity.

Similar to the registration process, users are required to sign the challenge file using their private key. This step ensures that the user possesses the private key corresponding to the selected authentication method.

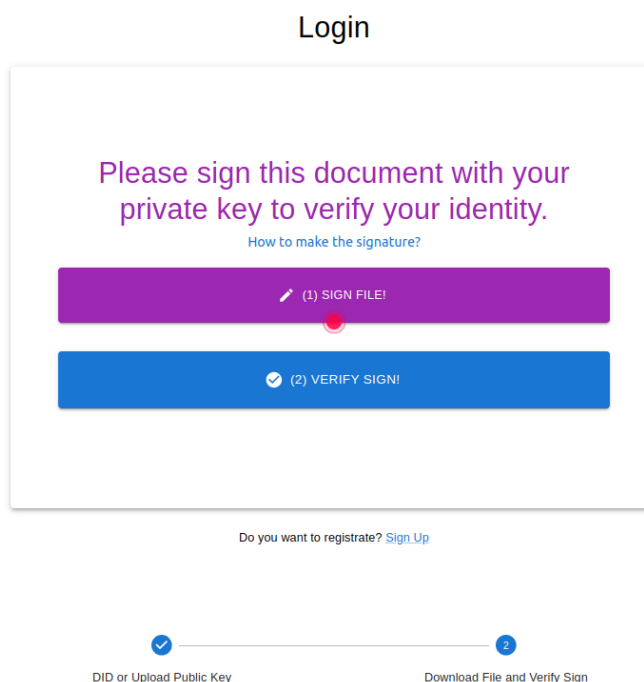


Figure 4.20: Challenge File Signature.

After successfully validating the signature, users gain access to the Expid web application.

4.1.1.4 Currency Offering Process

The currency offering process involves the seller entering currency details, signing a challenge to authenticate their offer, and receiving a verifiable credential upon successful validation. This process ensures that currency exchanges are securely registered and verified within the Expid application.

Step 1: Offer Currencies Adding Currency Details: The first step involves the seller entering the details of the currencies they want to offer or register. The seller specifies different currencies they wish to exchange, such as USD (US Dollar) and EUR (Euro). For each currency, the seller must input the amount they are offering. For example, the seller might offer 150 USD and indicate that it costs 136,462 CLP (Chilean Pesos). Similarly, they might offer 100 EUR, which costs 742 BOB (Bolivian Boliviano).

Selecting the Exchange Location: After entering the currency details and their respective exchange rates, the seller must select the country where the exchange will take place. For instance, if the seller is located in Spain, they choose Spain as the exchange location. This information helps potential buyers know where the exchange can occur.

Figure 4.21: Offer Currencies.

Step 2: Sign the Challenge.

Figure 4.22: Sign the Challenge.

Signature Process: This process is similar to the signature step during account creation but does not require the seller to upload their public key or DID. Instead, the system automatically detects the seller's DID to validate the signature.

Generating the Signature: The seller downloads the challenge file and signs it using their private key. This step ensures the authenticity and integrity of the offer.

Step 3: Validate and Create Verifiable Credential.

Validation and Credential Creation: After the seller uploads the signed challenge, the Expid application validates the signature. If the signature is successfully validated, the system creates a verifiable credential for the seller's offer. This credential registers the details of the offered currencies and ensures that the offer is securely recorded.

Successful Response: Once the credentials are created, the seller receives a successful response from the application, confirming that their currency offering has been registered. This response assures the seller that their offer is now part of the Expid system and available for potential buyers.

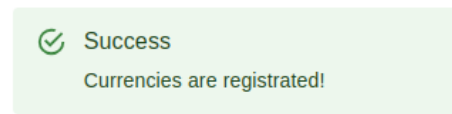


Figure 4.23: Successful Response.

4.1.1.5 Steps to Get Currency.

The process of getting currency involves navigating to the "I want to get currency" tab, filling in the necessary details, searching for and selecting a suitable offer, and then making a connection with the seller. The buyer completes the process by signing a challenge and obtaining a verifiable credential, ensuring a secure and authenticated transaction.

Step 1: Navigate to "I want to get currency".

When users navigate to the "I want to get currency" tab, they are presented with a set of input fields to fill in. The first field, labeled "Where?", requires the buyer to choose the country where they wish to obtain the currency. This helps in filtering the available offers based on the buyer's location. The next input, "I want currencies from...", allows the buyer to select multiple currencies they are interested in exchanging. This field provides the flexibility to choose from different exchange currencies.

WANT TO OFFER CURRENCY I WANT TO GET CURRENCY MY CURRENCIES MY CONNECTIONS

I WANT TO GET CURRENCY

Country where I want to change ...

Where?

I want currencies from ...

Currency...

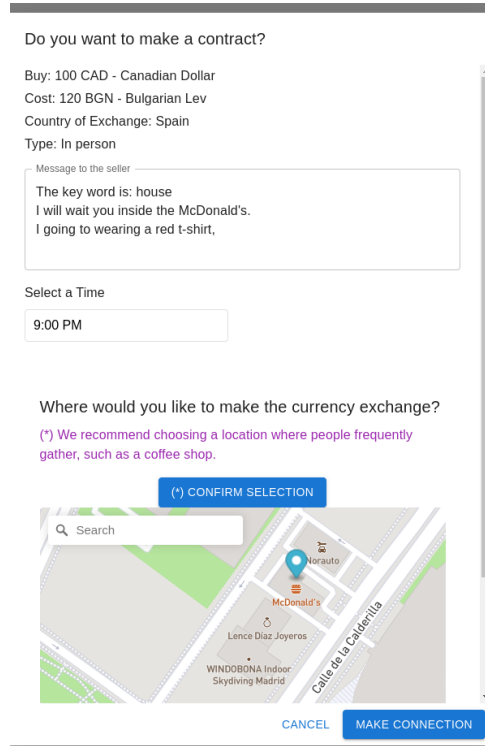
| Currency Value | Currency | Cost Value | Cost | Connect |
|----------------|-----------------------|------------|---------------------|--|
| 100 | CAD - Canadian Dollar | 120 | BGN - Bulgarian Lev | <input type="button" value="CONNECT"/> |

Figure 4.24: Get Currencies Offered.

Step 2: Searching for Currency Offers.

Once the buyer has filled in the necessary details, they click on the "Get" button to initiate the search. The application then loads a list of currency offers from various

sellers. The buyer reviews these offers, which might include exchanges like 100 CAD (Canadian Dollar) for 120 BGN (Bulgarian Lev). After reviewing the list, the buyer can click the "Connect" button next to the desired offer.



Step 3: Connecting with a Seller.

Clicking "Connect" opens a popup with detailed information about the selected offer. In this popup, the buyer can write a message to the seller to arrange the meeting for the currency exchange. The buyer specifies the time and location for the meeting, such as a local McDonald's. After filling in these details, the buyer clicks the "Make connection" button to proceed.

Figure 4.25: Sign the Challenge.

Step 4: Signing the Challenge.

Next, the buyer needs to complete a challenge signature process similar to the one used during registration. They download a challenge file and sign it using their private key, then upload the signed file back to the Expid application. Once the signed challenge is uploaded, the application validates the signature.

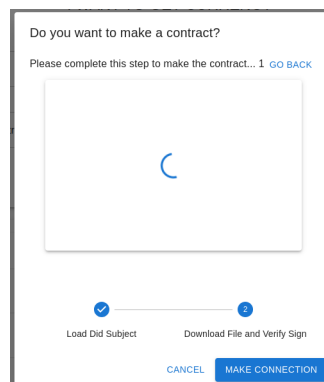


Figure 4.26: Loading the DID subject.

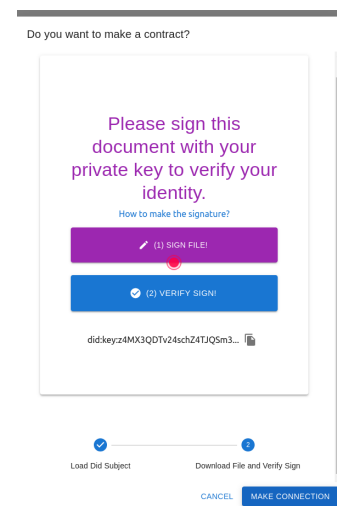


Figure 4.27: Download the Challenge File and Upload the Signature.

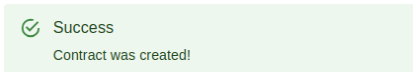


Figure 4.28: Successfully Connection Created.

Step 5: Validation and Credential Creation.

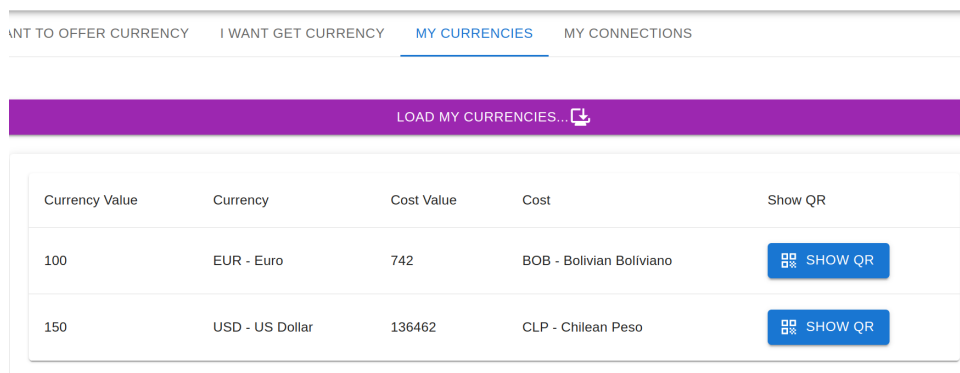
Upon successful validation, the Expid application creates a verifiable credential for the contract initiated by the buyer. This credential confirms the buyer's intent to obtain the specified currency and records the agreement between the buyer and the seller. This secure and authenticated transaction process ensures trust and transparency in the currency exchange.

4.1.1.6 Display and Verify Currency Offerings

After creating currency offerings, the seller can easily access and display them in the "My Currencies" tab. By generating and sharing a QR code, the seller can allow potential buyers to verify their credentials securely. This process enhances trust and transparency in currency exchanges within the Expid web application.

Step 1: Accessing "My Currencies"

In this tab, the seller clicks the "Load my Currencies..." button. This action retrieves and displays all the currencies that the seller has offered.

The screenshot shows a web interface with a navigation bar at the top containing four tabs: "WANT TO OFFER CURRENCY", "I WANT GET CURRENCY", "MY CURRENCIES" (which is active and underlined), and "MY CONNECTIONS". Below the navigation bar is a purple button labeled "LOAD MY CURRENCIES..." with a download icon. Underneath is a table with five columns: "Currency Value", "Currency", "Cost Value", "Cost", and "Show QR". The table contains two rows of data.



| Currency Value | Currency | Cost Value | Cost | Show QR |
|----------------|-----------------|------------|--------------------------|---|
| 100 | EUR - Euro | 742 | BOB - Bolivian Boliviano |  SHOW QR |
| 150 | USD - US Dollar | 136462 | CLP - Chilean Peso |  SHOW QR |

Figure 4.29: Currencies Created by the Seller.

Step 2: Generate QR Code.

The seller can click the "Show QR" button next to each currency listing. This option allows the seller to choose what information they want to display to a verifier, such as a potential buyer. Upon clicking "Show QR", the seller can select specific details to share. After selecting the desired information, the seller clicks "Apply". This action enables a new button labeled "Generate QR".

Clicking "Generate QR" generates a QR code. The seller can then show this QR code to the buyer. The QR code can be scanned by any device with a camera, such as a smartphone. The Expid application also provides an option on the home page to scan the QR code without needing to log in.

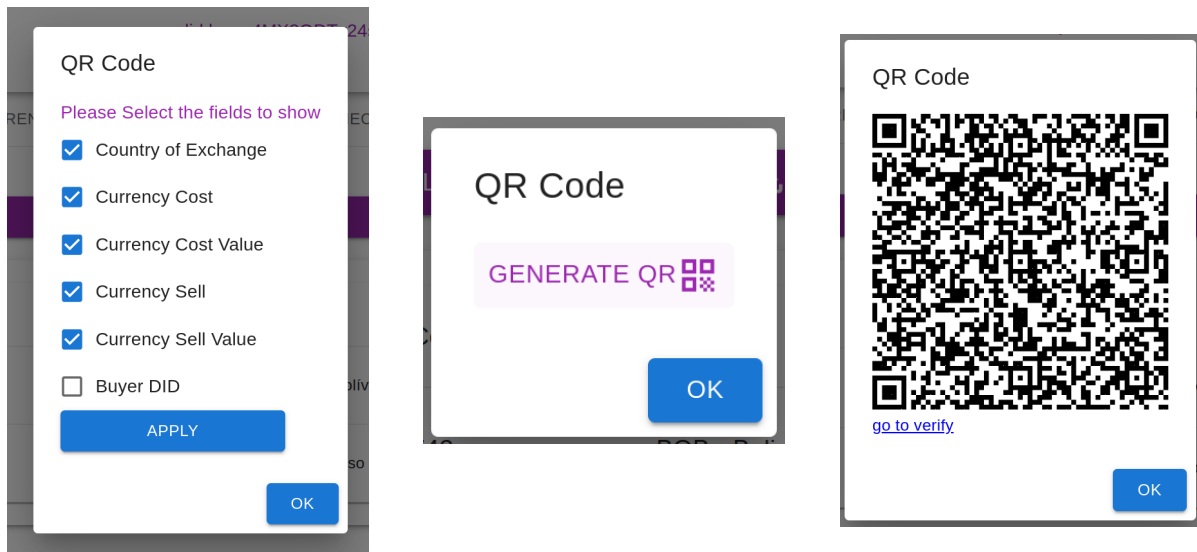


Figure 4.30: (A) Selecting fields to show. (B) Generate QR Code. (C) QR Code.

Step 3: Verifying Credentials.

When the buyer (verifier) scans the QR code, they are redirected to a verification page. This page displays the credentials and specific details that the seller chose to share. It confirms that the seller is who they claim to be and verifies the authenticity of the offered currencies.

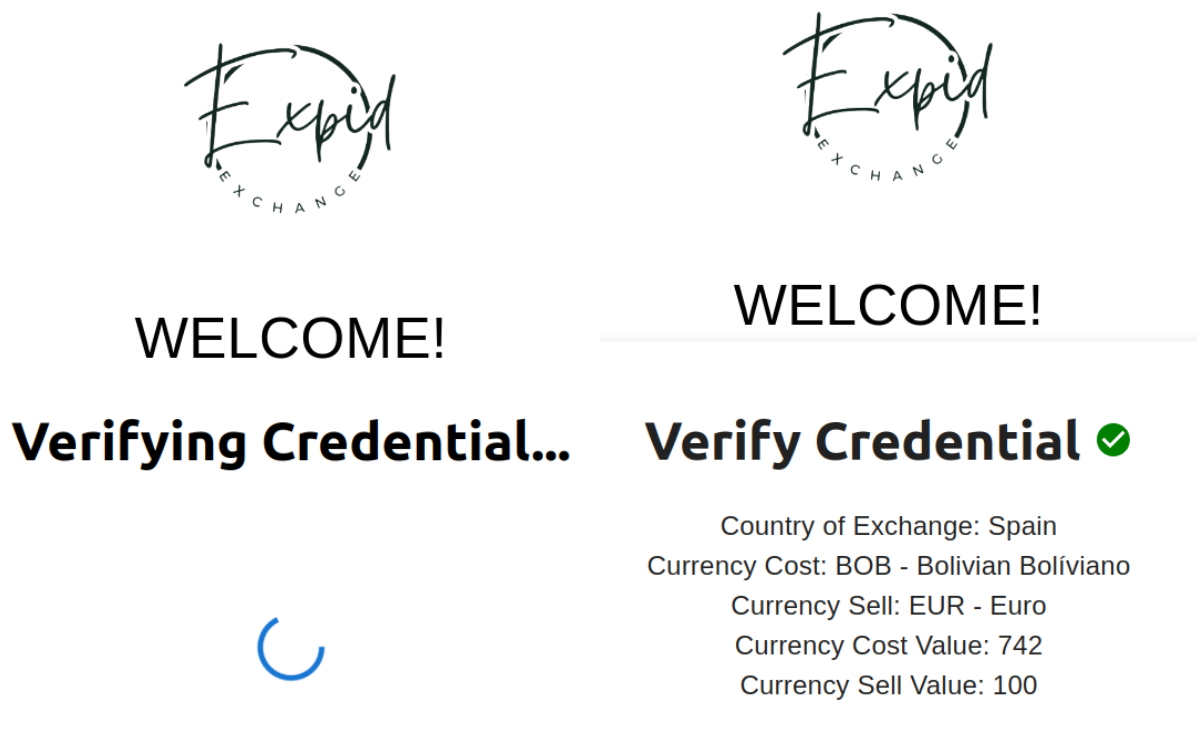


Figure 4.31: (A) Verification Page Loading. (B) Displaying the Verification Data.

Evaluation and Conclusion

4.1.1.7 QR Code Generation and Verification for Buyers.

The process for buyers to generate and verify QR codes is parallel to the process used by sellers. Buyers can manage their currency requests, generate QR codes to share with sellers, and ensure their requests are authenticated and verified.

Step 1: Accessing "My Connections"

Similar to how sellers manage their currency offers, buyers also have a way to manage and verify their requests. After making a currency request, buyers can go to the "My Connections" tab within the Expid web application. Here, they can view all their active currency requests.

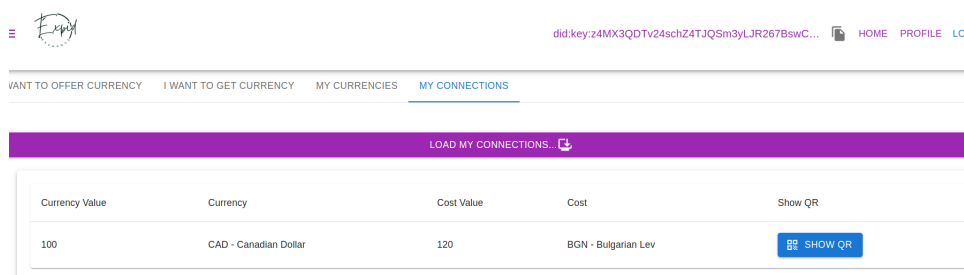


Figure 4.32: Connections Tab.

Step 2: Generating QR Code

Next to each request, there is a "Show QR" button. The buyer clicks this button to start the process of generating a QR code for verification purposes.

A popup appears where the buyer can select what specific information they want to share with the seller. This might include details about the requested currencies and any other relevant information.

After selecting the details to display, the buyer clicks "Apply", which enables a new button labeled "Generate QR". Clicking "Generate QR" generates a QR code that the buyer can present to the seller.

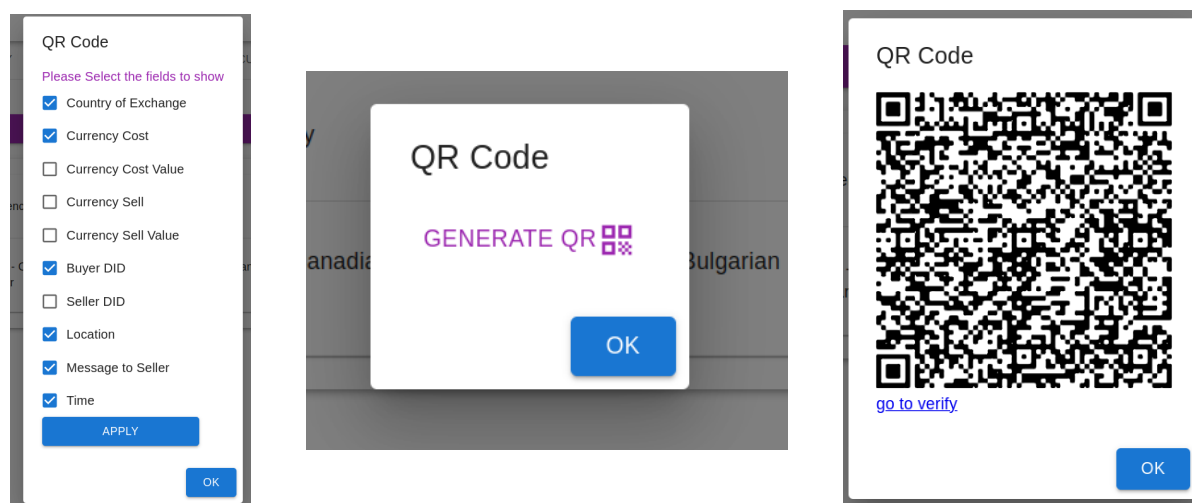


Figure 4.33: (A) Selecting fields to show. (B) Generate QR Code. (C) QR Code.

Step 3: Verification by Seller. The seller scans the QR code presented by the buyer. This can be done using a mobile device or within the Expid application itself.

Upon scanning, the seller is redirected to a verification page. This page displays the specific details the buyer chose to share, allowing the seller to verify that the buyer's request is legitimate and matches the information provided.

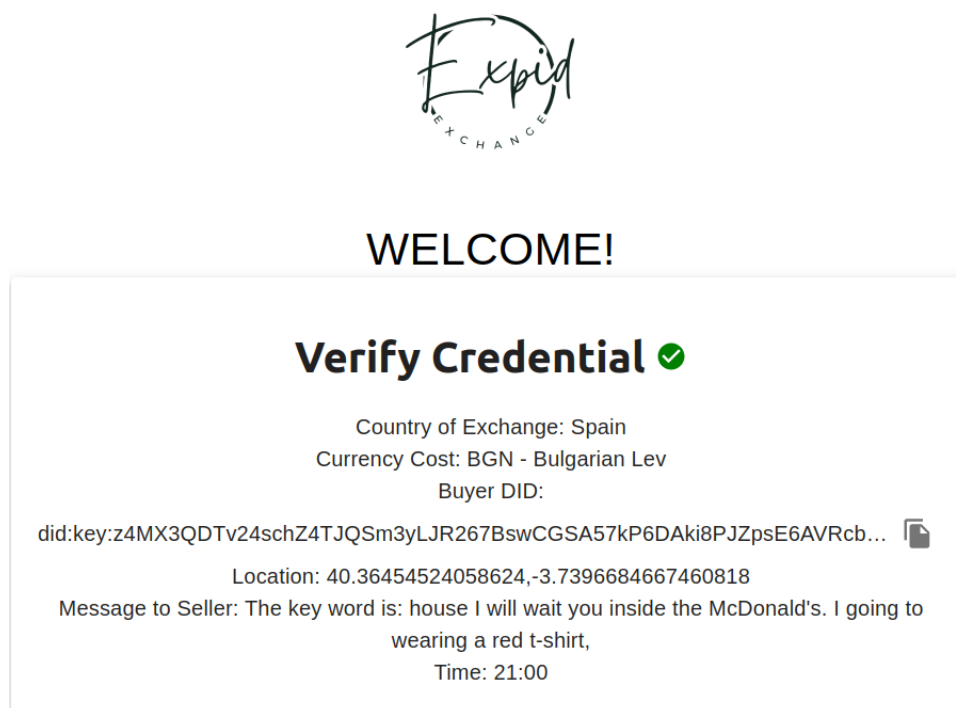


Figure 4.34: Verifying Credential

4.1.1.8 DID Document

When the Expid Application create a DID Subject it is based on the uploaded public key, Expid creates a DID Subject (identifier) following the W3C DID specification for the "*did:key*" method. This identifies the user.

Then, a DID Document is created to hold information about the user's DID Subject and the key used for authentication as following:

- **@context:** References the standards used for interpreting the document (e.g., W3C DID standard, security context).
- **id:** This is the DID Subject created for the user (in this case: *did:key:...*).
- **publicKey:** An array containing details about the public key used for authentication. This includes:
 - **id:** A unique identifier for the key within the DID Document.
 - **type:** The type of public key (in this case "*RsaVerificationKey2018*").
 - **publicKeyPem:** The actual public key data in PEM format.

Evaluation and Conclusion

- **authentication:** An array containing information about the authentication key, mirroring the publicKey section and referencing the same key used for signing.

```
{
  "@context": [
    "https://www.w3.org/ns/did/v1",
    "https://w3id.org/security/v1"
  ],
  "id": "did:key:
    z4MX3QDTv24schZ4TJQSm3yLJR267BswCGSA57kP6DAki8PJZpsE6AVRcbmt
    dqZrRMxBBedvLdQdubViADzuUU3ws2TSRmgvJqCi7NTxrcKWUXURCBSutBxwEc1
    gCaQbUgS39BMrYZucN6VbGsUntTnNX1qMqbLrDy9DPkATEfRRYfn2kyC8ulpbPqg1o
    LfDT7xjnjf8NUoiovqNEjuL8DRXFCd5Z5x4SHupTv4fUX3ykRv7enxTuGb33WwPtLQ
    27U6k88mRZfuisdi7YfGrekG4dQ12FaTXUdkREKCCwFYSgDwcxt1TUfZyLQSJgZ6fR
    dLuK8G2RUeJxUdpu6NXBhiMnj6d9Xaa je61WJpS33EnXqba2o7hhQbka72y4F9HN2
    iV7A81Z5UMHqJnUwk3WqE",
  "publicKey": [
    {
      "id": "did:key:
        z4MX3QDTv24schZ4TJQSm3yLJR267BswCGSA57kP6DAki8PJZpsE6AVRc
        bmtdqZrRMxBBedvLdQdubViADzuUU3ws2TSRmgvJqCi7NTxrcKWUXURCBSutBxwEc1
        gCaQbUgS39BMrYZucN6VbGsUntTnNX1qMqbLrDy9DPkATEfRRYfn2kyC8ulpbPqg1o
        LfDT7xjnjf8NUoiovqNEjuL8DRXFCd5Z5x4SHupTv4fUX3ykRv7enxTuGb33WwPtLQ
        27U6k88mRZfuisdi7YfGrekG4dQ12FaTXUdkREKCCwFYSgDwcxt1TUfZyLQSJgZ6fR
        dLuK8G2RUeJxUdpu6NXBhiMnj6d9Xaa je61WJpS33EnXqba2o7hhQbka72y4F9HN2
        iV7A81Z5UMHqJnUwk3WqE#keys-1",
      "type": "RsaVerificationKey2018",
      "controller": "did:key:
        z4MX3QDTv24schZ4TJQSm3yLJR267BswCGSA57kP6DAki8PJZpsE6AVRc
        bmtdqZrRMxBBedvLdQdubViADzuUU3ws2TSRmgvJqCi7NTxrcKWUXURCBSutBxwEc1
        gCaQbUgS39BMrYZucN6VbGsUntTnNX1qMqbLrDy9DPkATEfRRYfn2kyC8ulpbPqg1o
        LfDT7xjnjf8NUoiovqNEjuL8DRXFCd5Z5x4SHupTv4fUX3ykRv7enxTuGb33WwPtLQ
        27U6k88mRZfuisdi7YfGrekG4dQ12FaTXUdkREKCCwFYSgDwcxt1TUfZyLQSJgZ6fR
        dLuK8G2RUeJxUdpu6NXBhiMnj6d9Xaa je61WJpS33EnXqba2o7hhQbka72y4F9HN2
        iV7A81Z5UMHqJnUwk3WqE",
      "publicKeyPem": "-----BEGIN PUBLIC KEY-----\
        nMIIBITANBgkqhkiG9w0BAQEFAAOCAQ4AMIIBCQKCAQBAsLpBIa jeZc+bgEIyK+D0\
        nhcyfJt5BPdimBXxzUSYM4byu0uaUq0fFOz8Sna4mCs7u9sURo+mYYRgu7eVuLRYb\
        ng4neFxxgCwIC0z1AbWXuo4kQMMZZ2SMWVExfa idPX+7y6/fIXT1VIR5xTad/safw\
        nwgo/WuDIF7lh23bwSIie5TNx0VF70A7x0UD8YqhxE3Ozm7fyLELbGcUfvtogsA0P\
        n69bFLT/srKYybHYbD1ZEgyqONIkD3zuKsf+wWBgpKwfdO+uzviZkNdY4tyJRS/nt\
        nzNZJH5DDhO/vTfFJr/OPKqNxrzc4DiUSInrfkKw6pSEiBJ0v+nmGOC4rhnNdnRhr\
        nAgMBAAE=\n-----END PUBLIC KEY-----"
    }
  ],
  "authentication": [
    {
      "id": "did:key:
        z4MX3QDTv24schZ4TJQSm3yLJR267BswCGSA57kP6DAki8PJZpsE6AVRc
        bmtdqZrRMxBBedvLdQdubViADzuUU3ws2TSRmgvJqCi7NTxrcKWUXURCBSutBxwEc1
        gCaQbUgS39BMrYZucN6VbGsUntTnNX1qMqbLrDy9DPkATEfRRYfn2kyC8ulpbPqg1o
        LfDT7xjnjf8NUoiovqNEjuL8DRXFCd5Z5x4SHupTv4fUX3ykRv7enxTuGb33WwPtLQ
        27U6k88mRZfuisdi7YfGrekG4dQ12FaTXUdkREKCCwFYSgDwcxt1TUfZyLQSJgZ6fR
        dLuK8G2RUeJxUdpu6NXBhiMnj6d9Xaa je61WJpS33EnXqba2o7hhQbka72y4F9HN2i
        V7A81Z5UMHqJnUwk3WqE#keys-1",
```

```

"type": "RsaSignatureAuthentication2018",
"controller": "did:key:
  z4MX3QDTv24schZ4TJQSm3yLJR267BswCGSA57kP6DAki8PJZpsE6AVRc
  bmtdqZrRMxBBedwLdQdubViADzuUU3ws2TSRmgvJqCi7NTxrcKWUXURCBSutBxwEc1
  gCaQbUgS39BMrYZucN6VbGsUntTnNX1qMqblRdy9DPkATEfRRYfN2kyC8u1pbPgg1o
  LfDT7xjnjf8NUOeovqNEjuL8DRXFCd5Z5x4SHupTv4fUX3ykRv7enXTuGb33WwPtLQ
  27U6k88mRZfuisdi7YfGrekG4dQ12FaTXUdkREKCCwFYSgDwcxt1TufZyLQsJgZ6fR
  dLuK8G2RUeJxUdpu6NXBhiMnj6d9Xaa je61WJpS33EnXqba2o7hhQbka72y4F9HN2i
  V7A81Z5UMHqJnUwk3WqE",
"publicKeyPem": "-----BEGIN PUBLIC KEY-----\
  nMIIBITANBgkqhkiG9w0BAQEFAAOCAQ4AMIIBCQKCAQBAsLpBIa jEZc+bgEiYk+D0\
  nhcyfJt5BPdimBXxzUSYM4byu0uaUq0fFOz8Sna4mCs7u9sURo+mYYRgu7eVuLRYb\
  ng4neFxxgCwIC0z1AbWXuo4kQMMZ2SMWVexfaidPX+7y6/fIXT1VIR5xTad/safw\
  nwgo/WuDIF7lh23bwSIie5TNx0VF70A7x0UD8YqhxE30zm7fyLELbGcUfvtogsA0P\
  n69bFLT/srKYybHYbD1ZEgyqONIkD3zuKsf+wWBgpKwfd0+uzviZkNdY4tyJRS/nt\
  nzNZJH5DDH0/vTfFJr/OPKgNxrpc4DiUSInRfkW6pSEiBJ0v+nmGOC4rhnNdnRHR\
  nAgMBAAE=\n-----END PUBLIC KEY-----"
}
]
}

```

Listing 4.1: DID Document generated by the Server

The syntax validation of the DID Document was also performed using the DID Lint tool, which confirmed successful validation¹.

The screenshot shows the DID Lint Beta - The DID Validator interface. At the top, there is a logo and a navigation button labeled 'OwnYourData.eu'. Below the header, a blue banner states: 'This page checks the syntax and structure of a DID document to ensure that it conforms to the W3C Decentralized Identifiers Core Specification.' A text input field contains the DID key: 'did:key:z4MX3QDTv24schZ4TJQSm3yLJR267BswCGSA57kP6DAki8PJZpsE6AVRcbmtdqZrRMxBBedwLdQdubViADzuUU3ws2TSRmgvJqCi'. To the right of the input is a 'Resolve' button. Below the input, the 'DID Document' section displays the JSON document content. At the bottom of this section are 'Validate' and 'Reset' buttons. The 'Result' section shows a green banner with the text: 'Conforms to W3C DID Spec v1.0'.

Figure 4.35: DID Document Validated with DID Lint Tool.

4.1.1.9 Verifiable Credentials

This sub section analyzes a Verifiable Credential (VC) issued by the Expid application, demonstrating its potential role in facilitating secure and verifiable transactions.

As mentioned in previous sections, the Verifiable Credential (VC) is a tamper-proof digital document that can be used to prove ownership of certain characteristics, such

¹<https://didlint.ownyourdata.eu/validate>

Evaluation and Conclusion

as identity or qualifications. The Verifiable Credentials (VCs) illustrated below pertain to buyer and seller interactions.

First, it is important to explain the general structure, which contains the *hash* and *verifiableCredential* fields:

- **hash:** This is a unique identifier for the VC, likely generated using a cryptographic hash function.
- **verifiableCredential:** This is the core of the VC and contains details about the credential itself.
 - **credentialSubject:** This section holds information about the subject of the credential. It will be explain in for each verify credential issued by the Expid application for sellers and buyers.
 - **issuer:** This identifies the entity that issued the VC, which in this case is "did:web:expid-veramo-agent.onrender.com". This is the Expid's agent acting on its behalf.
 - **type:** An array specifying the type of credential. For this is: "VerifiableCredential".
 - **@context:** An array of URLs referencing the standards used to interpret the VC such as W3C VC standard and Veramo profile context.
 - **issuanceDate:** The date and time the VC was issued, for instance: May 29th, 2024.
 - **proof:** This section contains cryptographic proof that the VC is genuine and has not been tampered with. It uses the "JwtProof2020" format and includes a JSON Web Token (JWT) for verification.

The *credentialSubject* for the sellers includes the following fields:

- **currency_sell:** This field indicates "USD - US Dollar" is being offered by the seller.
- **currency_sell_value:** This field specifies the amount being offered in USD, which is "150".
- **currency_cost:** This field indicates "CLP - Chilean Peso" is expected as payment.
- **currency_cost_value:** This field specifies the amount expected in CLP, which is "136462".
- **country_of_exchange:** This field indicates "Spain" as the intended location for the exchange.
- **id:** It contains a the did subject of the seller.

```
{
  "hash": "QmWW6XPoTEkrUcgxGcE2M9cNun2iQciEL9GSaWKZ4Nm5Vn",
  "verifiableCredential": {
    "credentialSubject": {
      "currency_sell": "USD - US Dollar",
      "currency_sell_value": "150",
      "currency_cost_value": "136462",
```

```

"currency_cost": "CLP - Chilean Peso",
"country_of_exchange": "Spain",
"id": "did:key:
    z4MX3QDTv24schZ4TJQSm3yLJR267BswCGSA57kP6DAki8PJZpsE6AVRcb
    mtdqZrRMxBBedwLdQdubViADzuUU3ws2TSRmgvJqCi7NTxrcKWUXURCBSutBxwEc1gC
    aQbUgS39BMrYZucN6VbGsUntTnNX1qMqbLrDy9DPkATEfRRYfN2kyC8u1pbPqg1oLfD
    T7xjnjf8NUoiovqNEjuL8DRXFCd5Z5x4SHupTv4fUX3ykRv7enxTuGb33WwPtLQ27U6
    k88mRZfuisdi7YfGrekG4dQ12FaTXUdkREKCCwFYsGdwcx1TUfZyLQsJgZ6fRdLuK8
    G2RUeJxUdpu6NXBhiMnj6d9Xaa je6lWJpS33EnXqba2o7hhQbka72y4F9HN2iV7A81Z
    5UMHqJnUwk3WqE"
},
"issuer": {
  "id": "did:web:expid-veramo-agent.onrender.com"
},
"type": [
  "VerifiableCredential"
],
"@context": [
  "https://www.w3.org/2018/credentials/v1",
  "https://veramo.io/contexts/profile/v1"
],
"issuanceDate": "2024-05-29T18:06:53.000Z",
"proof": {
  "type": "JwtProof2020",
  "jwt": "eyJhbGciOiJIJzEiLCJ0eXciOiJ1dGV4dCI6WyJod
    HRwczovL3d3dy53My5vcmcvMjAxOC9jcmVkbW50aWFscy92MSIsImh0dHBzOi8vdmV
    YW1vLmlvL2NvbnRleHRzL3Byb2ZpbGUvdjEiXSwidHlwZSI6WyJWZXJpZmlhYm91Q3J
    lZGVudG1hbCJdLCJpc3RfYyIiw3VycmVudjEiLCJjZmVudG1hbCJdLCJpc3RfYyI
    mN5X2Nvc3RfdmFsdWUiOiIxMzY0Y0NjIiLCJ0eXciOiJ1dGV4dCI6WyJod
    bGVhbiBQZXNvIiw3VycmVudjEiLCJ0eXciOiJ1dGV4dCI6WyJod
    kaWQ6a2V5Ono0TVgzUURUdjEiLCJ0eXciOiJ1dGV4dCI6WyJod
    FraThQSlpwc0U2QVZSY2JtdGRxWnJSTXhCQmVkd0xkUWR1Y1ZpQUR6dVVVW3dzM1RTU
    mlndkpxQ2k3TlR4cmNLV1VYVVJDQ1N1dEJ4d0VjMwYVZlZk4ya3lDOHUXcGJQcW
    YkdzVW50VG50WDFxTXFiTHJEeTlEUGtBEVEMU1JZZk4ya3lDOHUXcGJQcWcx
    3eGpuamY4TlVvZW92cU5FanVMOERSWEZDZDVANXg0U0h1cFR2NGZVWDN5a1J2N2
    R1R2IzM1d3UHRMUTI3VTZrODhtUlpmdWl3ZGk3WWZHcmVrRzRkUTEyRmFUFVka1J
    ONDd0ZUZU2dEd2N4dDFUUVZaeUxRU0pnWjZmUmRmdUs4RzJSVWVKeFVkcHU2TlhCa
    bmo2ZDlYYWFqZTYxv0pwUzMzRW5YcWJhMm83aGhRYmthNzJ5NEY5SE4yaVY3QTg
    VTUhxSm5Vd2szV3FFIiwibm9kaWQ6d2ViOmV4cG
    lklXZlcmFtby1hZ2VudC5vbnRlbnRlc3RfYyIiw3VycmVudjEiLCJ0eXciOiJ1dGV
    xZlcmFtby1hZ2VudC5vbnRlbnRlc3RfYyIiw3VycmVudjEiLCJ0eXciOiJ1dGV4d
    jUjg4sF0dZK6c6C1EL9L9bU_aq1BJEs5ocQVzvj_01fmWFB5NNfFcy7Z9aBw"
  }
}
}

```

Listing 4.2: Verifiable Credential For Sellers

The Verifiable Credential for buyers when they are making a connection in the section of *credentialSubject* is as following:

- **did_buyer and did_seller:** These are Decentralized Identifiers (DIDs) representing the buyer and seller, respectively.
- **id_credential_seller:** This is an identifier assigned to the specific credential

Evaluation and Conclusion

issued by the seller.

- **message_to_seller:** This field contains a message from the buyer to the seller, including details about the meetup, such as location hints and clothing description, to facilitate recognition and coordination.
- **currency_sell and currency_sell_value:** These fields specify the currency and amount the seller is offering, for example, 100 CAD.
- **currency_cost and currency_cost_value:** These fields specify the currency and amount the buyer is expected to pay, for instance, 120 BGN.
- **country_of_exchange:** This indicates the country where the exchange is supposed to happen, such as Spain.
- **location:** This array contains the GPS coordinates of the planned meetup location, providing precise details for where the exchange will take place.
- **time:** This specifies the planned meeting time, in this example, 21:00, ensuring both parties are synchronized for the exchange.

```
{
  "hash": "QmerGKjn8c7itfywveNBBpvSr8n17zcLAgEEWmmZWJo5AG",
  "verifiableCredential": {
    "credentialSubject": {
      "did_buyer": "did:key:
        z4MX3QDTv24schZ4TJQSm3yLJR267BswCGSA57kP6DAki8PJZpsE6AVRcb
        mtdqZrRMxBBEdwLdQdubViADzuUU3ws2TSRmgvJqCi7NTxrcKWUXURCBSutBxwEc1gC
        aQbUgS39BMrYZucN6VbGsUntTnNX1qMqbLrDy9DPkATEfRRYfn2kyC8ulpbPqg1oLfd
        T7xjnjf8NUoiovqNEjuL8DRXFCd5Z5x4SHupTv4fUX3ykRv7enxTuGb33WwPtLQ27U6
        k88mRZfuisdi7YfGrekG4dQ12FaTXUdkREKCCwFYSgDwcxt1TUfZyLQsJgZ6fRdLuK8
        G2RUeJxUdpu6NXBhiMnj6d9XaaJe61WJpS33EnXqba2o7hhQbka72y4F9HN2iv7A81Z
        5UMHqJnUwk3WqE",
      "did_seller": "did:key:
        z4MX3QDTv24schZ4TJQSm3yLJR267BswCGSA57kP6DAki8PJZpsNTvWhCm
        ENFeiKs95SrnH6CwA7i1PLo9BTiWp1eKHh2g2ja2c2rjVxMR2boxdeONH4CAXfMDLNN
        gL3HvQnzLw3hX82LP9S3Qwe2S8J6EA8xT5UXG2cDunwb2EwomGHeQVzD6CVpjRUCZ5L
        fyTGx9BJcfU2kgtMnTdoeoQsbWRnYm2fJAnue2cFaMmPUSJxcyDinASpXcgE5cX4pm
        2NAvwrR3KfpyEYQKZRsJRSazEDYPEZTcPqvbxctGfDcwCoMn7wSFs89jAZhVRVDQJ6
        XyhH3QHSnxWkmQr4cnQuhpYvqbjQc87vfEigJmKX9g7JaFrrbZz8J8fE8WvjpXiN9xw
        oNLPaHntBx77da",
      "id_credential_seller": "credentialId",
      "message_to_seller": "The key word is: house \nI will wait you
        inside the McDonald's.\nI going to wearing a red t-shirt,",
      "currency_sell": "CAD - Canadian Dollar",
      "currency_sell_value": "100",
      "currency_cost_value": "120",
      "currency_cost": "BGN - Bulgarian Lev",
      "country_of_exchange": "Spain",
      "location": [
        40.36454524058624,
        -3.7396684667460818
      ],
      "time": "21:00"
    },
    "issuer": {
```


Evaluation and Conclusion

Laws of Identity. By conducting this evaluation, we aim to ensure that the platform not only meets its intended functionality but also complies with essential principles of identity management and data protection.

4.1.2.1 Validation Against Cameron's Laws of Identity

Regarding the Seven Laws of Identity, we can confirm that the Expid web application aligns with at least four of these established principles, as detailed below:

1. **User Control and Consent:** Expid ensures that users generate their own asymmetric key pairs and control their private keys, providing consent for identity verification and credential issuance. Users decide when and with whom to share their verifiable credentials, maintaining control over their personal data.
2. **Minimal Disclosure for a Constrained Use:** Expid adheres to this principle by only requiring the user's public key and a signature for registration. Verifiable credentials include only the necessary information for specific transactions, ensuring minimal disclosure.
3. **Justifiable Parties:** Expid ensures that identity information is shared only with verified entities involved in monetary exchanges or contractual agreements. Users can generate verifiable presentations that are shared with justifiable parties, validated through QR codes.
4. **Human Integration:** Expid features an intuitive web application interface that simplifies key generation, registration, and verification processes. Users interact with the system in a easy manner, enhancing usability and adoption.

4.1.2.2 Compliance with Requirements

Functional Requirements:

- **R1.** The Expid web application includes a key generation feature on the registration page. Users can easily generate their asymmetric key pairs using this feature, which ensures that the keys are created securely on the client side.
- **R2.** The registration process allows users to upload their X.509 certificates. The application extracts the public key from the certificate and uses it for registration, ensuring that users with existing certificates can seamlessly join the platform.
- **R3.** During registration, users provide their public key and a digital signature of a challenge file. The application verifies the signature against the public key to authenticate the user, ensuring the private key remains secure and is never uploaded to the system.
- **R4.** The application employs robust cryptographic techniques to verify user identities during interactions. This includes verifying digital signatures and utilizing their public key to validate the user authentication.
- **R5.** Upon successful verification of user identity, the application generates verifiable credentials for users. These credentials are issued when users register a new monetary exchange or create a contract, ensuring the authenticity of the user's claims.

- **R6.** The QR code generated by the application contains all necessary information for verification. Users can scan the QR code using any QR code reader or the Expid application, which verifies the data against the stored credentials.

Non-functional Requirements:

- **NFR1.** The key pairs are generated on the client side using secure algorithms, and the private keys are never transmitted or stored by the Expid application. This ensures that the private keys remain secure and under the control of the users.
- **NFR2.** The application follows strict privacy standards. User data is only included in verifiable credentials when necessary and with user consent. The design ensures that only minimal information required for verification is shared.
- **NFR3.** The Expid web application features a user-friendly interface built with React.js and Material UI. The key generation, registration, and verification processes are streamlined and easy to follow, ensuring a positive user experience.
- **NFR4.** The application uses Veramo for handling DIDs and verifiable credentials, ensuring compliance with W3C DID standards.
- **NFR5.** The backend infrastructure, hosted on Render and utilizing PostgreSQL and Firebase, ensures high availability and reliability. The system includes error handling mechanisms to maintain data integrity and user trust even in case of failures.
- **NFR6.** The system must comply with relevant regulations and standards for data protection and cryptographic security, including the General Data Protection Regulation (GDPR). This ensures that user data is handled with the utmost confidentiality and integrity, providing transparency and control over personal information in accordance with legal requirements. An exception to this compliance is the deletion of the DID Document initially generated, as it is stored on IPFS and is immutable.

4.2 Conclusion

In conclusion, this master's thesis has explored and demonstrated the feasibility and effectiveness of integrating decentralized identity (DID) and verifiable credentials (VCs) within a peer-to-peer (P2P) currency exchange application, which we called Expid. By developing a proof-of-concept (PoC) and validating its core functionalities, this research has shown that DID and VCs can significantly enhance security, privacy, and user control in digital world.

During the research and implementation of the Expid application, it became evident that while there are numerous implementations of SSI, VCs, and DIDs, and a growing development and documentation of standards in Europe, there remains a lack of detailed information on integrating DIDs using alternative methods such as DID IPID.

The main contributions and future work are detailed below. It is important to know that the Expid application diverged from traditional applications by not requiring a digital wallet; instead, an agent was employed to manage credentials, as mentioned earlier. Additionally, blockchain technology, commonly associated with DIDs in other

Evaluation and Conclusion

applications, was not necessary for this PoC. Instead, a document-based store was used to manage user transaction information, ensuring data ownership and user control. Identity verification was achieved through signature validation with users' private keys and registration of their public keys, effectively implementing SSI in this project.

Overall, this thesis shows how decentralized identity can change how we trust and control our digital information, making whichever transactions safer and more user-friendly on the digital world.

4.2.1 Main Contribution

The main contributions made in this master's thesis project were:

- **C1. Innovative Utilization of Decentralized Identity and Verifiable Credentials:** The PoC Expid introduce the implementation of decentralized identity (DID) and verifiable credentials (VCs) in a practical setting. Unlike traditional implementations requiring digital wallets for VC storage, Expid utilizes an agent-based approach to this use case, demonstrating innovative methods for credential management.
- **C2. Validation of Development Methodology:** The development of Expid serves as a robust proof-of-concept (PoC) showcasing the efficacy of its development methodology. This approach underscores the feasibility of integrating DID and VCs to create secure and intuitive applications, validating their potential in enhancing user authentication and data security.
- **C3. Peer-to-Peer Currency Exchange:** Expid introduces a peer-to-peer (P2P) currency exchange model empowered by DID and VCs. This innovative use case eliminates intermediaries, thereby reducing transaction costs and improving the user privacy and control over their transactions.
- **C4. Educational Demonstration of DIDs, VCs, and SSI:** Through comprehensive demos, Expid elucidates the collaborative workings of DIDs, VCs, and self-sovereign identity (SSI). By replacing password-based authentication with cryptographic signing, users securely assert their identity, emphasizing data minimization and user-centric data sharing practices.

4.2.2 Future Work

Regarding to the future work on Expid application developed could focus on several key areas to enhance its functionality and impact:

1. **Scalability of Users:** Enhance Expid's scalability to support a larger user base and decentralized technologies.
2. **User Experience and Accessibility:** Refine the user interface and experience to ensure accessibility and ease of use across different devices.
3. **Integration with Emerging Technologies:** Explore opportunities to integrate the Expid application with emerging technologies such as blockchain interoperability protocols to enhance functionality and expand use cases.

4. **Signature Creation:** Introduce a desktop application similar to Kleopatra that simplifies the process of creating digital signatures. This application should offer an intuitive interface where users can securely generate a signature without relying on command-line interfaces like openSSL.
5. **Infrastructure Deployment:** Improve the current infrastructure deployment process to reduce startup times for the Expid web application and other transaction processes. Evaluate and streamline deployment configurations and dependencies to ensure faster initialization and improved user experience.

References

- [1] Blockchain Facts: What Is It, How It Works, and How It Can Be Used. (n.d.). Retrieved March 22, 2024, from <https://www.investopedia.com/terms/b/blockchain.asp>
- [2] What Is Blockchain and How Does It Work? | Synopsys. (n.d.). Retrieved March 22, 2024, from <https://www.synopsys.com/glossary/what-is-blockchain.html>
- [3] What is blockchain storage? (n.d.). Storage. Retrieved March 22, 2024, from <https://www.techtarget.com/searchstorage/definition/blockchain-storage>
- [4] Peer-to-Peer (P2P) Service: Definition, Facts, and Examples. (n.d.). Investopedia. Retrieved March 22, 2024, from <https://www.investopedia.com/terms/p/peertopeer-p2p-service.asp>
- [5] Cope, J. (2002, April 8). What's a Peer-to-Peer (P2P) Network? Computerworld. <https://www.computerworld.com/article/2588287/networking-peer-to-peer-network.html>
- [6] S, J. (2022, August 27). Peer to peer: ¿qué es y cuáles son sus ventajas? Economía3. <https://economia3.com/peer-to-peer/>
- [7] Romero, M. S. (2019, September 14). ¿Qué es P2P y en qué consiste? Computer Hoy. <https://computerhoy.com/reportajes/tecnologia/p2p-que-es-489221>
- [8] What is a Peer-To-Peer Network (P2P)? - Definition from Techopedia. (n.d.). Retrieved March 22, 2024, from <https://www.techopedia.com/definition/25777/peer-to-peer-network-p2p-network>
- [9] What is PayPal and How Does it Work | PayPal US. (n.d.). Retrieved March 22, 2024, from <https://www.paypal.com/us/digital-wallet/how-paypal-works>
- [10] Introduction | Performant and modular APIs for Verifiable Data and SSI. (n.d.). Retrieved March 22, 2024, from <https://veramo.io/docs/basics/introduction>
- [11] Reynolds, N. (2023, January 26). Veramo is Switching to ESM (and pnpm). Veramo. <https://medium.com/veramo/veramo-is-switching-to-esm-and-pnpm-57cf8e841ffa>
- [12] Johnson-Ubah, A. (2023, October 24). How Verifiable Credentials Work. Medium. <https://medium.com/veramo/how-verifiable-credentials-work-2648733abd59>

-
- [13] Reynolds, N. (2023, July 17). Veramo DIDComm: Bringing Identity to AI. Medium. <https://medium.com/veramo/veramo-didcomm-bringing-identity-to-ai-f7efd1fbdf5d>
- [14] Zettler, J. (2017, October 10). How to Save a File on IPFS — A 7-Step Primer. Medium. <https://medium.com/@JohnZettler/how-to-save-a-file-on-ipfs-a-7-step-primer-3476469536c7>
- [15] Get a Pod · Solid. (n.d.). Retrieved March 22, 2024, from <https://solidproject.org/users/get-a-pod>
- [16] Mwiti, D. (2022, July 21). Image Segmentation: Architectures, Losses, Datasets, and Frameworks. Neptune.AI. <https://neptune.ai/blog/image-segmentation>
- [17] IEEE Xplore Full-Text PDF: (n.d.). Retrieved April 1, 2024, from <https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=10194951>
- [18] EU Digital Identity Wallet Pilot implementation | Shaping Europe's digital future. (2024, March 21). <https://digital-strategy.ec.europa.eu/en/policies/eudi-wallet-implementation>
- [19] Build a Scalable Database with Typescript and IPFS | by Radovan Stevanovic | Level Up Coding. (n.d.). Retrieved April 2, 2024, from <https://levelup.gitconnected.com/build-a-scalable-database-with-typescript-and-ipfs-11ecef97e7d>
- [20] Ethereum API | IPFS API Gateway | ETH Nodes as a Service | Infura. (n.d.). Retrieved April 2, 2024, from <https://www.infura.io/blog/post/a-developers-tale-building-a-database-with-ipfs>
- [21] libp2p | IPFS Docs. (n.d.). Retrieved April 2, 2024, from <https://docs.ipfs.tech/concepts/libp2p/>
- [22] How to Host Dynamic Content on IPFS | IPFS Blog News. (n.d.). Retrieved April 2, 2024, from <https://blog.ipfs.tech/2023-how-to-host-dynamic-content-on-ipfs/>
- [23] A Technical Guide to IPFS – the Decentralized Storage of Web3. (n.d.). Retrieved April 3, 2024, from <https://www.freecodecamp.org/news/technical-guide-to-ipfs-decentralized-storage-of-web3/>
- [24] How to Perform RSA Encryption in Javascript (React.js) and Golang - Bartłomiej Mika. (n.d.). Retrieved April 3, 2024, from <https://bartlomiejmika.com/post/2022/how-to-perform-rsa-encryption-in-javascript-and-golang/>
- [25] IOTA: MAM Eloquently Explained | by ABmushi | Coinmonks | Medium. (n.d.). Retrieved April 3, 2024, from <https://medium.com/coinmonks/iota-mam-eloquently-explained-d7505863b413>
- [26] How do I Know It's Really You? Verifying Authenticity of Public Keys | by Vinnie Moscaritolo | Storm4 | Medium. (n.d.). Retrieved April 4, 2024, from <https://medium.com/storm4/how-do-i-know-its-really-you-verifying-authenticity-of-public-keys-1cba1ca36db6>

REFERENCES

- [27] IPID DID Method. (n.d.). Retrieved April 6, 2024, from <https://did-ipid.github.io/ipid-did-method/>
- [28] (31) ResNetLab: Elective Course Module - InterPlanetary Linked Data (IPLD) - YouTube. (n.d.). Retrieved April 6, 2024, from https://www.youtube.com/watch?v=Sgf6j_mCdjIt=2s
- [29] FIDO2 Explained: What Is FIDO2 and How Does It Work? | Hideez. (n.d.). Retrieved April 7, 2024, from <https://hideez.com/blogs/news/fido2-explained>
- [30] A Short Introduction to WebAuthn Authentication. (n.d.). Retrieved April 7, 2024, from <https://auth0.com/blog/webauthn-a-short-introduction/>
- [31] Web Authentication: An API for accessing Public Key Credentials - Level 3. (n.d.). Retrieved April 7, 2024, from <https://www.w3.org/TR/webauthn-3/>
- [32] IPLD From Data to Data Structures. (n.d.). Retrieved April 8, 2024, from <https://ipld.io/docs/motivation/data-to-data-structures/>
- [33] 8.2 The One-Time Pad and Perfect Secrecy. (n.d.). Retrieved April 17, 2024, from <https://www.cs.toronto.edu/~david/course-notes/csc110-111/08-cryptography/02-one-time-pad.html>
- [34] Symmetric Key Encryption - why, where and how it's used in banking. (n.d.). Retrieved April 17, 2024, from <https://www.cryptomathic.com/news-events/blog/symmetric-key-encryption-why-where-and-how-its-used-in-banking>
- [35] What is Symmetric Encryption? Symmetric-Key Algorithms. (n.d.). Retrieved April 17, 2024, from <https://www.clickssl.net/blog/what-is-symmetric-encryption>
- [36] Signing and Encrypting with JSON Web Tokens I. (n.d.). Retrieved April 19, 2024, from <https://www.praetorian.com/blog/signing-and-encrypting-with-json-web-tokens/>
- [37] Understanding Decentralized IDs (DIDs) | by Adam Powers | Medium. (n.d.). Retrieved April 22, 2024, from https://medium.com/@adam_14796/understanding-decentralized-ids-dids-839798b91809
- [38] What are the main elements of a data space? | Datos gob es. (n.d.). Retrieved May 10, 2024, from <https://datos.gob.es/en/blog/what-are-main-elements-data-space>
- [39] Gaia-X secure and trustworthy ecosystems with Self Sovereign Identity | by Berthold Maier and Prof. Dr. Norbert Pohlmann | Gaia X. Retrieved May 10, 2024, from https://gaia-x.eu/wp-content/uploads/2022/06/SSI_White_Paper_Design_Final_EN.pdf
- [40] Longares Díez, R. M. (n.d.). Automating the response to GDPR's Data Subject's Rights. Retrieved March 1, 2024, from https://oa.upm.es/71724/1/TFM_RICARDO_MARIA_LONGARES_DIEZ.pdf
- [41] Sanz González, G. (n.d.). Diseño e Implementación de un Sistema de Identidad Digital Descentralizada para Ciudadanos de la Unión

Europea en el Ámbito Sanitario. Retrieved June 13, 2024, from https://oa.upm.es/72965/1/TESIS_MASTER_GUILLERMO_SANZ_GONZALEZ.pdf

Annex

.1 README files

The README.md files for both the front-end and back-end projects are detailed below and can be viewed on the following page: <https://github.com/alivane/TFM-decentralize-identity/blob/main/README.md>.

.1.0.1 Front-End

This application is built to provide a user-friendly interface for currency exchange transactions in the Expid Application.

- **Deployment:** The main branch of this repository is automatically deployed on the server using `o.onrender`.
- **Environment Variables:** The frontend application utilizes environment variables for configuration. An example `.env` file is provided in the project, where you can define variables such as API endpoints or other settings.

```
# You will need to get a assymetric key pair
REACT_APP_PRIVATE_KEY = [PRIVATE KEY]
REACT_APP_ENDPOINT=[URL OF THE API || http://localhost:3001]
# You will need to get from https://www.mapbox.com/
# (This is for the MAP)
REACT_APP_MAPBOX_API_KEY=[MAPBOX API KEY]
```

- **Technologies Used:** ReactJS, Material UI, NodeJS, npm or yarn.
- **Available Scripts:** In the project directory (`cd frontend`), you can run:

```
npm install
```

To install the dependencies of the project.

```
npm start
```

Runs the app in the development mode.

Open `http://localhost:3000` to view it in your browser.

The page will reload when you make changes.

You may also see any lint errors in the console.

- **Additional Requirements:**

```
# Build the Docker image for the frontend application
docker build -t expid-frontend .
```

```
# Run the Docker container for the frontend application
docker run -p 80:80 expid-frontend
```

.1.0.2 Back-End

This application is responsible for handling server-side logic and database interactions for the Expid Application.

- **Deployment:** The main branch of this repository is automatically deployed on the server using `o.onrender`.
- **Environment Variables:** The backend application utilizes environment variables for configuration. An example `.env` file is provided in the project, where you can define variables such as database connection strings or API keys.

```
# You will need to get a project ID
# from infura https://www.pinata.cloud/
REACT_APP_PINATA_JWT=[PINATA SECRET JWT]
PINATA_SECRET=[PINATA SECRET ID]
PINATA_KEY=[PINATA KEY ID]

# You will need to get a assymmetric key pair
APP_PUBLIC_KEY = [PUBLIC KEY]
APP_PRIVATE_KEY = [PRIVATE KEY]

# API PORT
PORT=[PORT || 3001]

# You will need to get from GPC https://console.cloud.google.com
# for FIREBASE DATABASE REAL TIME service
GCP_SERVICE_ACCOUNT=[SERVICE ACCOUNT OF THE GCP IN JSON FORMAT]

# You will need to get a POSTGRESQL database, the same that the AGENT
DATABASE=[URL DATABASE OF AGENT]

# You will need to get from an AGENT launched.
VERAMO_API_KEY=[VERAMO API KEY]
VERAMO_AGENT_OPEN_API = https://[URL AGENT EXPID]
VERAMO_DID_SUBJECT = "did:web:[URL AGENT EXPID]"
```

- **Technologies Used:** NodeJS, TypeScript, PostgreSQL, Firebase Realtime Database, Mapbox API and npm or yarn.
- **Available Scripts:** To run the backend application locally (`cd backend`), you need to have NodeJS and TypeScript installed on your system. After cloning the repository, navigate to the project directory in your terminal and run the following command to install dependencies:

```
npm install
or
yarn install
```


REFERENCES

To install the dependencies of the project.

```
yarn ts-node --esm ./src/server.ts
```

Runs the api server app in the development mode. Open <http://localhost:3001> to view the API.

- **Additional Requirements:**

```
# Build the Docker image for the backend application
docker build -t expid-backend .
```

```
# Run the Docker container for the backend application
docker run -p 3001:3001 expid-backend
```

.1.0.3 Veramo Agent

This was generated from <https://github.com/decentralized-identity/veramo-agent-deploy>. The readme is in the following github page: <https://github.com/alivane/expid-veramo-agent/blob/next/readme.md>

The official Veramo Docker agent is built from this source.

- **Environment variables:** These are the default environment variables used in the production config.

BASE_URL: Set this to your base app URL. Your default web:did will be based on this when it gets created on first run. Be sure to replace **APP_NAME** with the actual name of your app.

API_KEY: Used for authorization.

SECRET_KEY: Used for encrypting the database.

AGENT_PATH: The path where the agent will be accessible from, default to agent.

MESSAGING_PATH: The path where the messaging will be accessible from, default to messaging.

PORT: The port to run the server on.

DATABASE_URL: The database connection string for Postgres.

- **Deploy to RENDER server:** Go to the <https://o.onrender.com/> project for this agent. Be sure to set the **BASE_URL** correctly, with your App name replacing **APP_NAME** in the URL.

You will be able to access your agent at

<https://your-app-name.o.onrender.com/agent>

and you can find your OpenAPI Schema at

<https://your-app-name.o.onrender.com/open-api.json>.

You can find the required API Key through the Render project settings (Reveal Config Vars).

The deployment will create a link between your forked repo and the Render instance. When you update your repo (e.g. updating the Veramo package version, changing your agent configuration), Render will automatically deploy.

- **Deploy to Render using Docker:** You will need to fork this repo and add the following key/value pair to the root of `app.json`. You can then use the same deploy button as above or use the Render CLI.

```
{
  "stack": "container"
}
```