



UNIVERSIDAD DE GRANADA

TRABAJO FIN DE GRADO
GRADO EN INGENIERÍA DE TECNOLOGÍAS DE
TELECOMUNICACIÓN

Evaluación de rendimiento de entorno SSI basado en tecnología *blockchain*

Evaluación de rendimiento de entorno SSI basado en
tecnologías *blockchain* del proyecto Hyperledger

Autor
Rafael Adán López

Director
Gabriel Maciá Fernández



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE
TELECOMUNICACIÓN

—
Granada, Junio de 2022

Evaluación de rendimiento de entorno SSI basado en tecnología *blockchain*

Autor
Rafael Adán López

Director
Gabriel Maciá Fernández

Evaluación de rendimiento de entorno SSI basado en tecnología *blockchain*

Rafael Adán López

Palabras clave: cadena de bloques, identidad auto-soberana, rendimiento, hyperledger, aries, indy.

Resumen

La confianza en Internet está rota. Antiguamente el número de entidades en Internet era limitado, pero ahora son necesarios mecanismos para asegurar la identidad digital de las entidades. El esquema más utilizado y estandarizado es el de usuario y contraseña para cada servicio requerido. Este esquema presenta diferentes vulnerabilidades como la exposición a la ingeniería social y a la correlación de los datos.

Como alternativa se propone el uso de proveedores de identidad, como lo son Google o Facebook. Mediante este esquema se solucionan algunas de las vulnerabilidades del anterior esquema. Sin embargo, sigue expuesto a la correlación (aunque en este caso, con consentimiento).

Para resolver estos problemas, surge el concepto de identidad auto-soberana. Utilizando la tecnología *blockchain* combinada con la gestión de identidades, se provee al usuario de la capacidad de controlar sus datos, es decir, cuando y cómo se proporciona a otros usuarios, y que cuando se comparten, sea de manera segura, sin necesidad de ninguna autoridad centralizada.

El objetivo de este proyecto es desplegar un entorno SSI basado en *blockchain* utilizando el proyecto Hyperledger. Se diseñará un ecosistema de agentes y se probará las distintas funcionalidades de estos. Finalmente, se desarrollará un entorno de pruebas con el objetivo de evaluar el rendimiento de los agentes frente a distintos escenarios.

Performance evaluation of SSI environment based on blockchain technology

Rafael Adán López

Keywords: blockchain, self-sovereign identity, performance, hyperledger, aries, indy.

Abstract

Trust in the Internet is broken. In the past the number of entities on the Internet was limited, but now mechanisms are needed to ensure the digital identity of entities. The most used and standardized scheme is the one with username and password for each service required. This scheme presents different vulnerabilities such as exposure to social engineering and data correlation.

An alternative is the use of identity providers, such as Google or Facebook. This scheme solves some of the vulnerabilities of the previous scheme. However, it is still exposed to correlation (although in this case, with consent).

To solve these problems, the concept of self-sovereign identity arises. Using blockchain technology combined with identity management, the user is provided with the ability to control their data, that is, when and how it is provided to other users, and that when it is shared, it is in a secure manner, without the need for any centralized authority.

The goal of this project is to deploy a blockchain based SSI environment using the Hyperledger project. An ecosystem of agents will be designed and their different functionalities will be tested. Finally, a test environment will be developed in order to evaluate the performance of the agents in different scenarios.

Yo, **Rafael Adán López**, alumno de la titulación Grado en Ingeniería de Tecnologías de Telecomunicación de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 77157458W, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Rafael Adán López

Granada a 21 de junio 2022 .

D. **Gabriel Maciá Fernández**, Profesor del Área de Ingeniería Telemática del Departamento de Teoría de la Señal, Telemática y Comunicaciones de la Universidad de Granada.

Informa:

Que el presente trabajo, titulado ***Evaluación de rendimiento de entorno SSI basado en tecnología blockchain***, ha sido realizado bajo su supervisión por **Rafael Adán López**, y autoriza la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expide y firma el presente informe en Granada a 1 de julio de 2022 .

El director:

Gabriel Maciá Fernández

Agradecimientos

Quiero agradecer primeramente a mis padres, Rafa y Tere, y a mi hermano, Ángel, por haberme apoyado siempre a la hora de tomar la decisión de empezar esta carrera. Sin vuestro apoyo incondicional las cosas habrían sido mucho más difíciles.

A mi novia, Marta, por haber sido un pilar fundamental durante estos duros 4 años, que siempre ha estado ahí cuando lo he necesitado, celebrando mis triunfos y consolándome en mis fracasos, pero siempre ahí. Y a sus padres y hermanos, Paco, Paloma, Laura y Álvaro, a los que ya considero mi segunda familia.

A mi abuelo Manolo y a mi tía Gema, por haber confiado siempre en mí y haberme demostrado tanto amor y cariño a pesar de la distancia. A las personas que por desgracia ya no están, mis abuelas Rita y Toni, con las que ojalá pudiese compartir la alegría de haber acabado.

No puedo olvidarme de mis compañeros de carrera, tanto a las nuevas amistades que he hecho y que espero que duren muchos años: Iván, Pepe-lu, Alex y Fernando; como a mis grandes amigos Eric y David, con los que tantas horas he pasado estudiando en el PTS, que en época de exámenes parecía nuestra segunda casa. Vuestra compañía y amistad estos 4 años y desde que llegué a Granada es algo de lo que siempre estaré agradecido. Ha sido un placer estudiar y trabajar con todos vosotros y espero que os vaya genial en todos vuestros futuros proyectos.

Por último quiero agradecer a todos los profesores que han hecho posible mi formación y que, aunque duramente, disfrutar esta carrera y futura profesión. Gracias por supuesto a mi tutor Gabriel, por haberme ayudado y haber hecho posible la realización de este proyecto.

'With great power comes great responsibility'

Ben Parker - Spider-Man

Índice general

Dedication	13
1. Introducción	1
1.1. Motivación y contexto del proyecto	1
1.2. Objetivos del proyecto y logros conseguidos	4
1.3. Estructura de la memoria	5
1.4. Contenidos teóricos para la comprensión de la solución adoptada	6
1.4.1. Self-Sovereign Identity (SSI)	6
1.4.2. Credenciales Verificables	8
1.4.3. Decentralized Identifiers (DIDs)	11
1.4.4. Agentes	15
1.4.5. Tecnología <i>blockchain</i>	16
1.4.6. Zero-Knowledge Proof (ZKP)	18
1.4.7. Revocación	19
2. Planificación y costes	21
2.1. Planificación del proyecto	21
2.2. Costes asociados al proyecto	23
2.2.1. Recursos humanos	23
2.2.2. Recursos software	24
2.2.3. Recursos hardware	24
2.2.4. Coste total del proyecto	25
3. Análisis del problema	27
3.1. Posibles soluciones existentes de identidad auto-soberana	27
3.1.1. Alastria	27
3.1.2. Hyperledger	29
3.1.3. uPort	32
3.2. Discusión, comparación y solución escogida	33
3.3. Análisis de la solución escogida	34
3.3.1. <i>Framework</i> de los agentes	35
3.3.2. Cadena de bloques	35

4. Diseño	41
4.1. Introducción	41
4.2. Roles de los agentes	42
4.3. <i>Von-network</i>	43
4.4. Servidor de registros de revocación: <i>indy-tails-server</i>	44
4.4.1. Acumuladores criptográficos	45
4.4.2. Archivos Tails	45
4.4.3. Configuración	46
4.4.4. Prueba de no revocación	46
4.4.5. Preparación para la revocación en la emisión	47
4.4.6. Presentación de prueba de no revocación	48
4.4.7. Puntos clave	48
4.5. Arquitectura de los agentes	49
4.6. Protocolos de Aries	51
4.6.1. Introducción: interoperabilidad en Aries	51
4.6.2. Protocolos de mensajería	52
4.6.3. Protocolo de establecimiento de conexión	53
4.6.4. Protocolo de emisión de credencial	59
4.6.5. Protocolo de presentación de credencial	65
4.7. Entorno de evaluación de funcionalidades	69
4.8. Entorno de evaluación de rendimiento	70
4.8.1. Emisión y presentación de credenciales simples	71
4.8.2. Emisión y presentación de credenciales revocables sin llegar a revocarlas	71
4.8.3. Credenciales revocables revocadas	72
5. Implementación	75
5.1. Introducción	75
5.2. Tecnologías y herramientas usadas	75
5.3. Implementación de <i>von-network</i>	76
5.4. Implementación de <i>indy-tails-server</i>	78
5.5. Implementación de los agentes	80
5.5.1. <i>run_demo</i> y <i>ngrok-wait.sh</i>	80
5.5.2. Alice	82
5.5.3. Faber	82
5.5.4. Acme	83
5.6. Implementación del entorno de evaluación de rendimiento	83
5.6.1. <i>performance.py</i>	83
5.6.2. <i>proof_presentation_CS.py</i>	84
5.6.3. <i>proof_presentation_CR_no_revocation.py</i>	85
5.6.4. <i>proof_presentation_CR_one_by_one_revocation.py</i>	87
5.6.5. <i>proof_presentation_CR_all_at_once_revocation.py</i>	89
5.6.6. <i>plot_CS.py</i>	91
5.6.7. <i>plot_CR_no_revocation.py</i>	97

5.6.8. <i>plot_CR_one_by_one_revocation.py</i>	97
5.6.9. <i>plot_CR_all_at_once_revocation.py</i>	97
6. Evaluación y pruebas	99
6.1. Prueba de evaluación de funcionalidades	99
6.1.1. Despliegue de los agentes	99
6.1.2. Conexión de los agentes	101
6.1.3. Emisión de credencial	101
6.1.4. Presentación de credencial	103
6.1.5. Revocación de credencial	106
6.2. Prueba de evaluación de rendimiento	108
6.2.1. Emisión y presentación de credenciales simples	109
6.2.2. Emisión y presentación de credenciales revocables sin ser revocadas	110
6.2.3. Emisión y presentación de credenciales revocables revocadas y publicadas una a una	111
6.2.4. Emisión y presentación de credenciales revocables revocadas y publicadas juntas al final	113
6.2.5. Conclusiones	114
7. Conclusiones	117
7.1. Resultados y trabajo realizado	117
7.2. Trabajos futuros	118
7.3. Valoración personal	119
Bibliografía	123
Glosario de siglas	125
A. Manual de usuario	127
A.1. Virtual Box: máquina virtual Ubuntu 20.04 LTS	127
A.2. Módulos y requerimientos	131
A.2.1. Git	131
A.2.2. Visual Studio Code	132
A.2.3. Docker y Docker Compose	132
A.3. Implementación de los entornos	133

Índice de figuras

1.1.	Esquema de usuarios y contraseñas.	2
1.2.	Esquema de IdP.	3
1.3.	Brechas de datos clasificadas según la sensibilidad de los datos [1].	3
1.4.	El modelo de credenciales en papel [2].	9
1.5.	El modelo de credenciales verificables según W3C [2].	10
1.6.	Ejemplo de DID [2].	11
1.7.	DID Público [2].	13
1.8.	DID Privado [2].	14
1.9.	Usos de DIDs privados y públicos [2].	14
1.10.	Ejemplo de intento de modificar una cadena de bloques [3]. .	18
1.11.	Diferencia entre arquitecturas de red [3].	19
2.1.	Diagrama de Gantt.	23
3.1.	Principios SSI según Alastria [4].	28
3.2.	El modelo de ID_Alastria [5].	29
3.3.	Infraestructura de ID_Alastria [5].	29
3.4.	Frameworks y herramientas de Hyperledger [2].	30
3.5.	Arquitectura de uPort [6].	33
3.6.	Tipos de <i>blockchain</i> [2].	37
3.7.	Qué hay dentro de la <i>blockchain</i> [2].	38
4.1.	Esquema del sistema diseñado.	42
4.2.	Esquema de la red <i>von-network</i>	44
4.3.	Archivo Tails [7].	46
4.4.	Archivo Tails tras revocar credenciales [7].	47
4.5.	Arquitectura lógica de agente Aries [8].	50
4.6.	Arquitectura de agente Aries [8].	51
4.7.	Protocolos de mensajería [8].	53
4.8.	Tabla de estado del remitente [9].	54
4.9.	Tabla de estado del receptor [9].	54
4.10.	Tablas de estado del solicitante y respondedor [10].	58

4.11. Diagrama de coreografía del protocolo de emisión de credencial [11].	59
4.12. Diagrama de coreografía del protocolo de presentación de credencial [12].	65
4.13. Arquitectura del entorno de evaluación de funcionalidades.	69
4.14. Diseño escenario CS.	72
4.15. Diseño escenario CR sin revocar.	73
4.16. Diseño escenario CR revocando.	74
5.1. Inicio <i>von-network</i>	77
5.2. Interfaz web <i>von-network</i>	77
5.3. Creación nuevo DID.	78
5.4. Transacción asociada a la creación de nuevo DID.	79
5.5. Inicio <i>indy-tails-server</i>	80
5.6. Url del servidor de revocación.	80
6.1. Despliegue de Alice.	100
6.2. Publicación DID de Faber.	100
6.3. Publicación de DID y atributos de Faber en la red.	101
6.4. Publicación de esquema de credencial de Faber.	102
6.5. Esquema de credencial de Faber en la red.	103
6.6. Publicación de definición de credencial y registro de revocación de Faber.	103
6.7. Definición de credencial de Faber en la red.	103
6.8. Registro de revocación de Faber en la red.	104
6.9. Código QR para conexión Faber.	104
6.10. Respuesta de Alice al mensaje <i>invitation</i>	104
6.11. Mensaje de Alice a Faber.	105
6.12. Recepción de mensaje de Faber.	105
6.13. Emisión de credencial de Faber a Alice.	105
6.14. Recepción de credencial de Alice emitida por Faber.	105
6.15. Solicitud de Presentación de Credencial de Acme a Alice.	105
6.16. Prueba de presentación de credencial de Alice a Acme.	106
6.17. Emisión de credencial de Acme a Alice.	106
6.18. Recepción de credencial de Alice emitida por Acme.	106
6.19. Revocación de credencial de Faber.	107
6.20. Revocación de credencial de Faber en la red.	107
6.21. Presentación de credencial fallida de Alice a Faber.	107
6.22. Entradas del servidor de revocación.	108
6.23. Anatomía diagrama de cajas y bigotes [13].	108
6.24. Tiempos para escenario de credenciales simples.	109
6.25. Medidas de tiempo para escenario de credenciales simples.	110
6.26. Tiempos para escenario de credenciales revocables sin llegar a ser revocadas.	111

6.27. Medidas de tiempo para escenario de credenciales revocables sin ser revocadas.	112
6.28. Tiempos para escenario de credenciales revocables revocando una a una.	113
6.29. Medidas de tiempo para escenario de credenciales revocables revocadas y publicadas una a una.	114
6.30. Tiempos para escenario de credenciales revocables revocadas y publicadas juntas al final.	115
6.31. Medidas de tiempo para escenario de credenciales revocables revocadas y publicadas juntas al final.	116
A.1. Interfaz Virtual Box.	128
A.2. Página de descarga Ubuntu 20.04 LTS.	128
A.3. Creación máquina virtual.	129
A.4. Crear disco virtual.	129
A.5. Selección de disco de inicio.	130
A.6. Guía de instalación Ubuntu.	130
A.7. Software adicional.	131
A.8. Final de instalación Ubuntu.	132

Índice de tablas

1.1. Los <i>Diez Principios</i> de SSI.	7
2.1. Tiempo dedicado al proyecto.	23
2.2. Costes asociados a recursos humanos.	24
2.3. Costes asociados a recursos software.	25
2.4. Características del ordenador utilizado.	25
2.5. Coste total asociado.	25
3.1. Características de las soluciones y comparación	34
4.1. Tipos de respuesta <i>Out-Of-Band</i> [9].	57

Listados de código

4.1. Ejemplo de mensaje <i>invitation</i> [9].	55
4.2. Ejemplo de mensaje <i>propose-credential</i> [11].	60
4.3. Ejemplo de mensaje <i>offer-credential</i> [11].	62
4.4. Ejemplo de mensaje <i>request-credential</i> [11].	63
4.5. Ejemplo de mensaje <i>issue-credential</i> [11].	63
4.6. Ejemplo de vista previa de credencial [11].	64
4.7. Ejemplo de mensaje <i>propose-presentation</i> [12].	66
4.8. Ejemplo de mensaje <i>request-presentation</i> [12].	66
4.9. Ejemplo de mensaje <i>presentation</i> [12].	67
4.10. Ejemplo de vista previa de presentación [12].	68
5.1. Código de <i>Dockerfile.demo</i>	81
5.2. Código de <i>ngrok-wait.sh</i>	82
5.3. Código de <i>proof_presentation_CS.py</i>	84
5.4. Código de <i>proof_presentation_CR_no_revocation.py</i>	85
5.5. Código de <i>proof_presentation_CR_one_by_one_revocation.py</i> . . .	87
5.6. Código de <i>proof_presentation_CR_all_at_once_revocation.py</i> . .	89
5.7. Código de <i>plot_CS.py</i>	91

Capítulo 1

Introducción

1.1. Motivación y contexto del proyecto

Según dijo una vez Peter Steiner [14]: 'En Internet, nadie puede saber si eres un perro'. Esta curiosa afirmación resume de forma sencilla el gran reto existente para poder identificar a las personas o entidades en Internet.

Antiguamente todos los ordenadores eran de confianza al haber solo un número limitado de ellos. No era necesario entonces tener un sistema para controlar la identidad de las entidades. A medida que creció Internet, se fueron creando diversos esquemas para identificar la identidad de los participantes.

El esquema más básico de identidad en Internet se basa en el uso de Identificador (ID) y contraseña para cada servicio, almacenado en los servidores de dicho servicio (también conocido como modelo aislado) (Figura 1.1). Para utilizarlo, el usuario se registra, crea su ID y elige una contraseña. Sin embargo, este esquema tiene varias desventajas/debilidades:

- Es necesario tener tantos IDs y contraseñas como sitios webs/aplicaciones quiera utilizar el usuario. Esto conlleva que a menudo se olviden las contraseñas. Los mecanismos de recuperación de contraseñas normalmente son una vía de ataque fácil para los hackers.
- Elegir contraseñas fáciles de recordar. Esto supone que al igual que son fáciles de recordar, son fáciles de adivinar.
- Utilizar la misma contraseña en distintos sitios. Esto conlleva, ya no solo la correlación de datos, sino a que cuando hay una filtración de datos todas las contraseñas sean expuestas y puedan ser utilizadas.
- La presencia de correlación de datos, es decir, la recolección sin consentimiento de información. Esta correlación es posible debido a los iden-

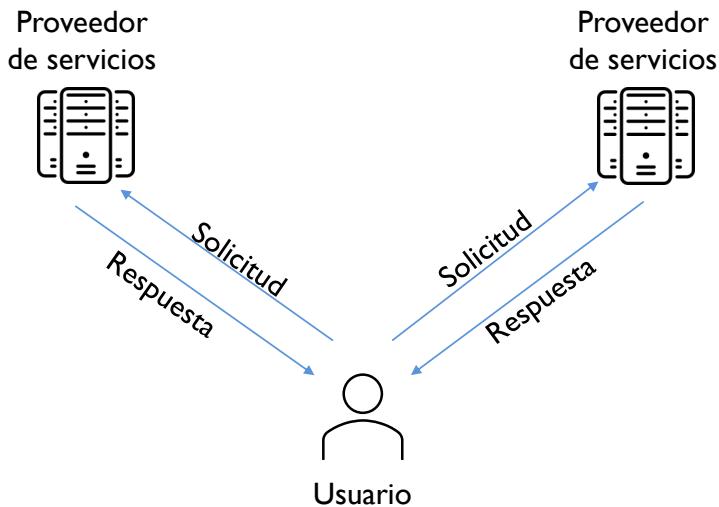


Figura 1.1: Esquema de usuarios y contraseñas.

tificadores utilizados: correos electrónicos, nombre de usuario, número de teléfono, DNI, etc. Mediante estos datos, las empresas intentan relacionar al usuario entre diferentes sitios.

Una alternativa que actualmente ha cogido bastante popularidad es la de utilizar un *Identity Provider* (*IdP*) (Figura 1.2). Ejemplos son Facebook o Google. Estos funcionan de tal manera que cada vez que el usuario quiera iniciar sesión en un sitio web/aplicación utiliza la cuenta del IdP.

Si bien es verdad que es una mejor solución que la anterior, sigue teniendo también algunos inconvenientes:

- Cada vez que se inicia sesión, el IdP aprende cosas sobre el usuario: sus hábitos, intereses, etc. La privacidad se pierde.
- Se es más susceptible al *phishing*. El usuario se identifica, pero la página a la que se conecta no. Esto hace que los usuarios sean vulnerables a entrar en páginas con dominios muy parecidos al original (como por ejemplo facebook.com) y que den su información.
- El esquema de IdP también es un punto de correlación, aunque en este caso, con consentimiento. Esto es, como ya se ha mencionado anteriormente, debido a que éste puede realizar un seguimiento de cada vez que el usuario inicia sesión, correlando las actividades en línea de este.

Ambos esquemas están expuestos a las famosas brechas de datos. La Figura 1.3, tomada de la referencia [1], muestra como los datos personales de

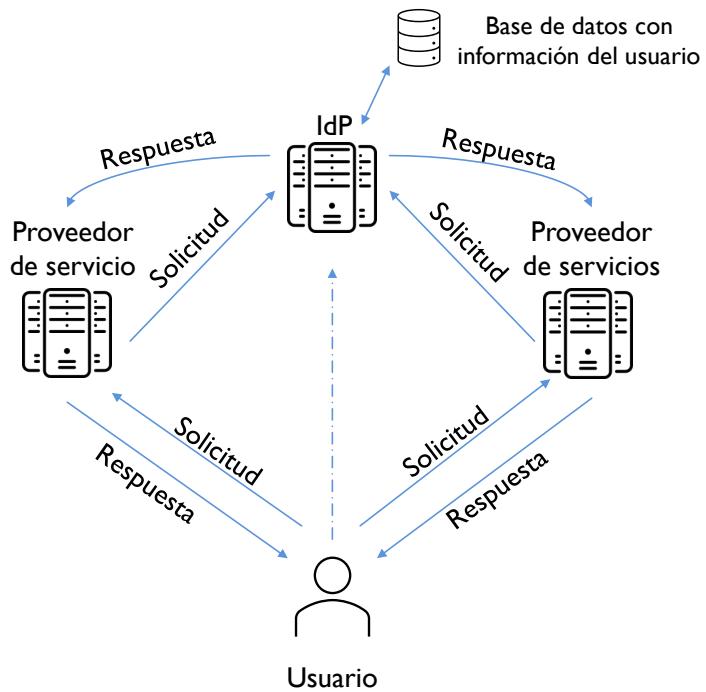


Figura 1.2: Esquema de IdP.

usuarios así como los correos electrónicos y números de teléfono son filtrados en brechas de datos de proveedores de servicios tan comúnmente usados como Facebook, LinkedIn o Twitter.

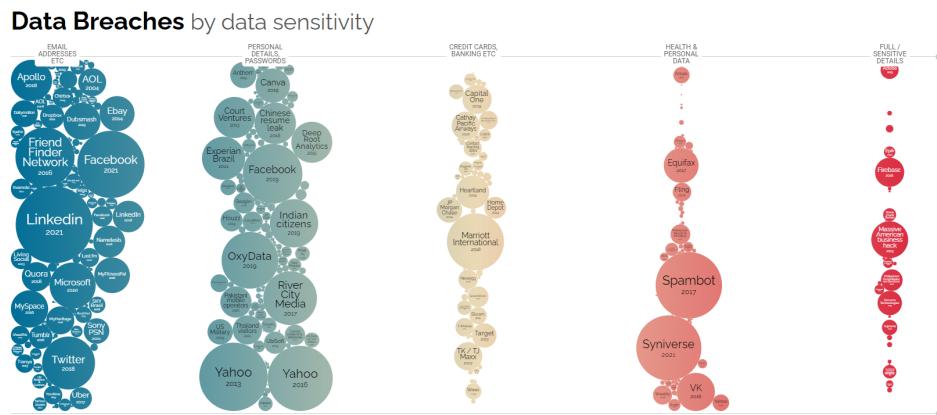


Figura 1.3: Brechas de datos clasificadas según la sensibilidad de los datos [1].

A raíz de todo esto surge la necesidad de que el usuario tenga un control

pleno de sus datos. Vistas las desventajas de los modelos centralizados, se empieza a poner el foco en modelos descentralizados. Aquí es donde entran las cadenas de bloques (tecnología *blockchain*).

La popularidad de la tecnología *blockchain* ha aumentado enormemente en los últimos años. Esta tecnología se ha aplicado a numerosos campos, entre ellos la gestión de identidades.

Así como las cadenas de bloques, la gestión de identidades también ha experimentado un aumento de interés debido al gran tiempo que pasan las personas en Internet, dando lugar al uso de identidades digitales. Se entiende como identidad digital a cualquier medio que pueda probar electrónicamente que una persona es quien dice ser [15].

Las cadenas de bloques se combinan adecuadamente con la gestión de identidades debido a la red que estas proporcionan. Se trata de una red *Peer-to-Peer (P2P)* donde solo se puede agregar información, la cual es inmutable debido a la criptografía subyacente (a través de *hashing* criptográfico).

Todo esto lleva a plantear cómo encontrar una solución a los problemas de identidad a Internet: entornos *Self-Sovereign Identity (SSI)* basados en *blockchain*.

1.2. Objetivos del proyecto y logros conseguidos

El objetivo del proyecto es implementar un entorno SSI basado en *blockchain* y evaluar su rendimiento frente a diversas pruebas.

Los logros propuestos para el proyecto son los siguientes:

- Realizar una investigación sobre la actualidad de la identidad digital y la confianza en Internet.
- Afianzar los conocimientos necesarios sobre la identidad auto-soberana y la tecnología *blockchain*.
- Realizar un estudio sobre las tecnologías e implementaciones disponibles, compararlas y elegir una de ellas justificadamente.
- Diseñar e implementar un entorno SSI en el que se prueben las funcionalidades de este:
 - Emisión de credenciales.
 - Presentación y verificación de credenciales.

- Revocación de credenciales.
- Desarrollar de un entorno de pruebas con el objetivo de evaluar el rendimiento de las funcionalidades anteriormente mencionadas.
- Sacar conclusiones respecto de las pruebas realizadas y localizar trabajos futuros en el proyecto.

1.3. Estructura de la memoria

A modo de introducción y resumen antes de leer el informe, se citan los capítulos de este:

1. **Introducción:** este capítulo sirve para introducir el contexto y la motivación del proyecto, con el objetivo de dar una vista general del propósito del trabajo realizado. Así mismo, se explicarán los conceptos teóricos necesarios para comprender la tecnología escogida para el proyecto.
2. **Planificación:** este capítulo muestra los tiempos estimados dedicados al proyecto, así como la organización y reparto de tareas, apoyado de un Diagrama de Gantt. Finalmente se hará una estimación de costes: coste del personal (tutor y estudiante), coste software y coste hardware.
3. **Análisis:** en este capítulo se estudian las distintas tecnologías para identidades descentralizadas basadas en *blockchain*. Se compararán entre ellas y se escogerá una tecnología justificadamente. Una vez escogida la tecnología, se irá analizando las soluciones que propone y se escogerán las más adecuadas.
4. **Diseño:** en este capítulo se explicarán los componentes de la solución escogida, analizando la arquitectura resultante. Se explicarán además los protocolos mediante los cuales se comunican los agentes. Finalmente se explicará el diseño y arquitectura de los dos escenarios creados para evaluar el entorno SSI.
5. **Implementación:** en este capítulo se explicará el código desarrollado para la implementación del entorno, así como desplegar los agentes y el esquema diseñado en la sección de diseño. Se guiará al lector para poder ejecutarlo e ir entendiendo el por qué de las decisiones.
6. **Pruebas:** en este capítulo, una vez montado el entorno, se realizarán pruebas para ver cómo se comporta la solución escogida. Por un lado, se ejecutará la prueba de evaluación de funcionalidades para ver cómo el sistema consigue funcionar y brindar las características y principios

de SSI. Tras esto, se ejecutará la prueba de evaluación de rendimiento, para comprobar el comportamiento y rendimiento del entorno frente a diferentes pruebas y escenario, con diferentes números de credenciales.

7. **Conclusiones:** en este apartado se llegarán a conclusiones basadas en el proyecto, así como de posibles implementaciones e investigaciones futuras. Finalmente se acabará con una valoración personal del trabajo realizado.

1.4. Contenidos teóricos para la comprensión de la solución adoptada

A continuación se presentan los conceptos teóricos clave para la comprensión de la tecnología escogida. El objetivo de esta sección es que el lector comprenda los aspectos básicos para poder entender como funciona Hyperledger con respecto a identidades digitales. Se han escogido como referencia los cursos “*Introduction to Hyperledger Sovereign Identity Blockchain Solutions: Indy, Aries & Ursa*” [2] y “*Introduction to Hyperledger Blockchain Technologies*” [3] de The Linux Foundation. Para más información, consultar las referencias.

1.4.1. Self-Sovereign Identity (SSI)

SSI es la idea de poder controlar nuestros datos, es decir, cuando y cómo se proporciona a otros usuarios, y que cuando se comparten, sea de manera segura. Gracias a la criptografía subyacente y a la tecnología *blockchain*, SSI permite que los reclamos de identidad se puedan verificar con certeza criptográfica.

Drummond Reed, gurú de la identidad y fideicomisario fundador de la Fundación Sovrin, describe la identidad auto-soberana como [2]: “Es una identidad portátil de por vida para cualquier persona, organización o cosa que no depende de **ninguna autoridad centralizada** y que no se puede quitar”.

Otra definición de SSI es la que da Christopher Allen [16], donde explica los *Diez Principios* de la identidad auto-soberana. En la Tabla 1.1 se pueden ver estos principios explicados.

Una última alternativa es el *framework* conocido como “las leyes de identidad”[17]. Estas leyes son:

- **Control y consentimiento del usuario:** la información del usuario solo debe revelarse cuando el usuario dé el consentimiento.

Principio	Explicación
Existencia	Los usuarios deben tener una existencia independiente.
Control	Los usuarios deben controlar sus identidades, es decir, deben ser la máxima autoridad. Son necesarios unos algoritmos seguros y fáciles de entender para hacer esto posible.
Acceso	Los usuarios deben tener acceso a sus propios datos, y conocer todos los atributos y credenciales dentro de su identidad.
Transparencia	Los sistemas y algoritmos deben ser transparentes, siendo de gran importancia que sean gratuitos y de código abierto.
Persistencia	Las identidades deben durar a lo largo del tiempo. Preferiblemente para siempre, o al menos hasta que el usuario lo desee.
Portabilidad	La información y los servicios de identidad deben ser portables. Las identidades portables garantizan que el usuario mantenga el control de su identidad pase lo que pase, independientemente de que las entidades puedan desaparecer en Internet.
Interoperabilidad	Las identidades deben ser ampliamente utilizables. Si solo funcionan en nichos limitados, estas pierden valor.
Consentimiento	Los usuarios deben ser los que acepten el uso de su identidad. El intercambio solo debe ocurrir con el consentimiento del usuario.
Minimización	La divulgación de atributos debe ser mínima, solo la necesaria.
Protección	Los derechos de los usuarios deben ser protegidos.

Tabla 1.1: Los *Diez Principios* de SSI.

- **Divulgación mínima para un uso restringido:** la solución que revela menor cantidad de información y limita mejor su uso es, a largo plazo, la solución más estable. Conservar información (por si acaso) es necesario no es una opción viable.
- **Partes justificables:** el sistema de identidad debe hacer consciente a su usuario de la parte o partes con las que está interactuando mientras comparten información.

1.4. Contenidos teóricos para la comprensión de la solución adoptada

- **Identidad dirigida:** un sistema de identidad universal debe admitir tanto identificadores ‘omnidireccionales’ para uso público como identificador ‘unidireccionales’ para uso privado. Esto evita identificadores de correlación.
- **Pluralismo de operadores y tecnologías:** la solución debe permitir la interoperabilidad entre distintas soluciones y diferentes esquemas y credenciales.
- **Integración humana:** la experiencia del usuario debe ser consistente con sus necesidades y debe saber en todo momento las implicaciones de sus interacciones con el sistema.
- **Experiencia consistente en todos los contextos:** los usuarios deben poder esperar una respuesta consistente experiencia en diferentes contextos de seguridad y plataformas tecnológicas

Destacar que, si se lee con atención los principios de SSI y las leyes de identidad, ambas son muy parecidas, lo que nos lleva a la conclusión de que SSI cumple con los requisitos que necesitan los usuarios para tener un entorno confiable en Internet.

1.4.2. Credenciales Verificables

Para poder entender correctamente la solución escogida, es preciso conocer primero como son las credenciales en papel.

Una credencial es una certificación que prueba una calificación, competencia o autoridad emitida a una entidad (ya sea un individuo o una organización) por un tercero que está autorizado a emitir credenciales. Ejemplos de credenciales son el carné de conducir, un pasaporte, un título académico, etc.

Por tanto, si se pone como ejemplo el carné de conducir, es el gobierno el que da la credencial (el emisor), la cual guarda el usuario (el titular) en su monedero o cartera. Posteriormente el titular podrá usar esa credencial para verificar que tiene permiso para conducir o que es mayor de edad para consumir en un pub (el verificador).

Por tanto, son tres los participantes:

- El emisor de la credencial.
- El titular de la credencial.
- El verificador de la credencial.

Este modelo de credenciales en papel (Figura 1.4) consigue demostrar:

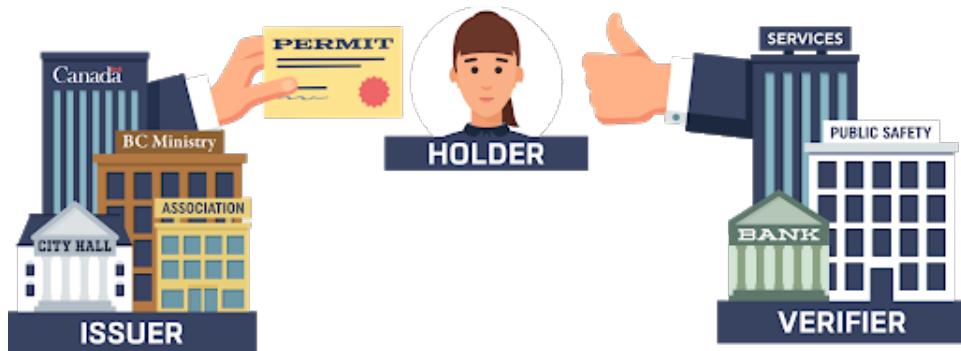


Figura 1.4: El modelo de credenciales en papel [2].

- Quién emitió la credencial.
- Quién tiene la credencial.
- Que la credencial no ha sido modificada.

Esta tercera afirmación no es 100 % cierta, ya que las credenciales se pueden falsificar, sea un ejemplo crear un DNI falso para poder entrar en discotecas donde solo se permiten mayores de edad.

Para que esto no ocurra, los verificadores necesitan tomar medidas para evitar o reducir la probabilidad de que la información proporcionada sea falsa. Estas medidas pueden ser pedir más datos de los necesarios, revisar los documentos para ver si parecen falsificados, etc. Estas medidas tienen consecuencias negativas tanto para el verificador como para el titular. Al titular le tomará más tiempo completar la transacción, teniendo que hacer varias visitas en persona y proveer más información de la que debería ser requerida. Por parte del verificador, el aumento de la complejidad del proceso puede traducirse en el posible abandono de algunos clientes, así como verse un aumento del costo del personal TI para proteger los datos extra requeridas de los clientes así como aprender a inspeccionar documentos.

Normalmente a la hora de utilizar credenciales el papel, el riesgo del verificador se centra en el titular, la persona que presenta la información. Sin embargo, existe un segundo riesgo: si el emisor es una autoridad válida de los datos que se presentan. Con los pasaportes es fácil pero, ¿qué ocurre con, por ejemplo, verificar un título universitario emitido por una universidad que el verificador no conoce? Los verificadores, normalmente, solo suelen aceptar credenciales de entidades/emisores que conocen.

Aquí es donde entran las credenciales verificables. Estas son datos protegidos criptográficamente que se pueden usar para demostrar tu identidad.

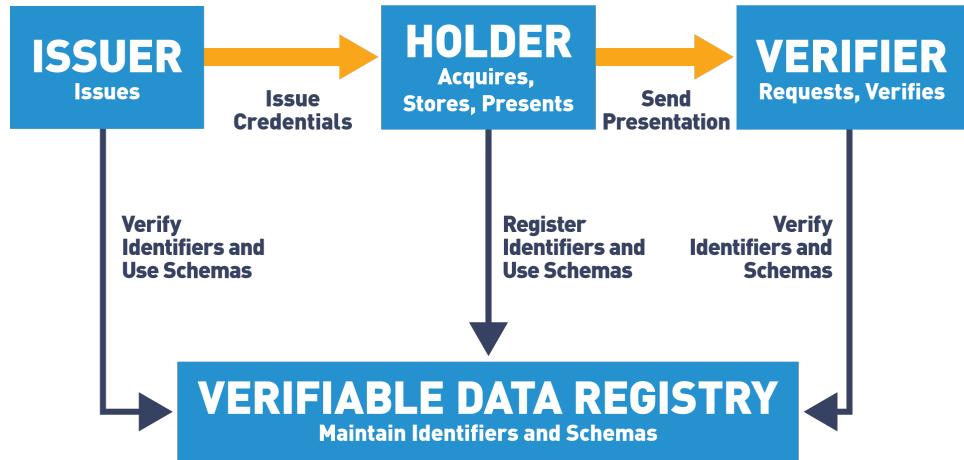


Figura 1.5: El modelo de credenciales verificables según W3C [2].

En la Figura 1.5 se puede observar el modelo de credenciales verificables según el *World Wide Web Consortium (W3C)*. En este se pueden apreciar tres roles fundamentales, como ya se vio en el modelo de credenciales en papel. El emisor (*issuer*) se encarga de verificar la identidad de los titulares (*holder*), y emitir credenciales para ellos. A su vez, sube los esquemas y definiciones de credenciales a la *blockchain*. El titular guarda en su *wallet* todas la credenciales recibidas, así como consultar el *ledger*. Por último el verificador (*verifier*) comprueba las credenciales e identificadores de los titulares, mediante presentaciones de pruebas.

A diferencia del anterior, en el modelo de credenciales verificables existe un registro de datos que contiene las claves criptográficas e identificadores, que permite verificar los datos y credenciales. Esta es la parte de *blockchain*, de la que se hablará en la Sección 1.4.5.

Este modelo es mucho más seguro que su versión en papel. El verificador, que antes tenía que comprobar si una credencial estaba falsificada, se olvida de este problema ya que si la verificación criptográfica tiene éxito, los datos pueden ser aceptados sin preocupación, debido a la fuerte seguridad de la criptografía.

Con este modelo se consigue uno de los principios fundamentales de SSI: se mejora la privacidad, ya que el titular tiene el control de cuándo, qué y con quién comparte su información. Además, los emisores y verificadores también tienen ventajas. Se reducen las conexiones entre emisores y verificadores, ya que el verificador puede comprobar la validez de las credenciales

del titular sin necesidad de contactar con el emisor.

1.4.3. Decentralized Identifiers (DIDs)

Un componente fundamental para poder hacer posible SSI son los *Decentralized Identifier (DID)*. Estos son un nuevo tipo de identificador que permite lo que SSI requiere: un ecosistema descentralizado.

Los DIDs son un tipo especial de identificador que crea su propio propietario. En la Figura 1.6 se puede ver un ejemplo de este tipo de identificador.

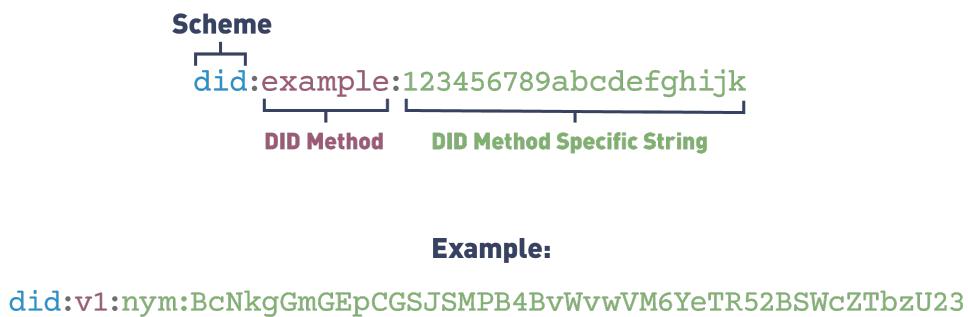


Figura 1.6: Ejemplo de DID [2].

Resumiendo, las características más importantes de los DIDs son:

- *Uniform Resource Identifier (URI)*. Es decir, al igual que las URLs, los DIDs se pueden resolver, mediante los llamados DID Resolvers. Estos funcionan como un navegador. Dado un DID, se devuelve el documento DID (DIDDoc). El DIDDoc es un documento JSON que contiene información sobre la entidad que lo ha generado.
- Son creados por cualquier persona en cualquier momento. Además, el usuario puede actualizar el DIDDoc cuando lo necesite.
- Globalmente único, siempre que se genere correctamente, es decir, que sigue las reglas para crear un tipo específico de DID.
- Altamente disponibles. Es decir, los DIDs se pueden resolver incluso si algunos servidores están caídos. Esto es debido a que los DIDs se resuelven mediante la lectura de la *blockchain*, la cual dispone de alta disponibilidad debido a sus múltiples copias.
- Verificables criptográficamente. Esto significa que se puede verificar el control de un DID si se le pide a la entidad que lo demuestre mediante el uso de su clave privada y clave pública.

Esta última característica lleva a volver a plantear qué información contiene el DIDDoc. Este contiene, como mínimo:

- Claves públicas con sus respectivas claves privadas asociadas (evidentemente, estas no están incluidas en el DIDDoc, las guarda secretamente el creador del DID).
- Puntos finales de servicio que permiten la comunicación con esa entidad.

Por tanto, el proceso en el que participan los DIDs es el siguiente:

1. Resolver un DID para obtener el DIDDoc.
2. Dentro del DIDDoc, se busca la clave pública y el punto final para contactar con la entidad.
3. El propio hecho de que la entidad (creador del DID) conteste, ya verifica que tiene el control del DID (porque para descifrar el mensaje cifrado con clave pública, se necesita la privada, que guarda secretamente el creador del DID).
4. Una vez establecida la comunicación, ambas entidades comparten un canal de comunicación seguro. Se podrán iniciar procesos desde un simple mensaje básico hasta emisión o presentación de credenciales.

Ya se ha visto un ejemplo de para qué sirven los DIDs, pero, ¿por qué son tan importantes en el ámbito de SSI?. Por sus potentes características que se describen a continuación:

- **Control del usuario:** como ya se ha dicho anteriormente, es el propio usuario el que crea el DID, y lo comparte bajo su control, principio fundamental de SSI.
- **Verificable:** gracias a la asociación de pares de claves pública/privada, se hace trivial probar el control del DID.
- **No correlacionables:** en el esquema tradicional de identificadores, el usuario proporciona, por ejemplo, su dirección de correo electrónico. El receptor del identificador puede correlatar su actividad según el identificador. Con los DIDs esto se evita, ya que se puede crear un DID para cada servicio que necesite el usuario (o conexión).
- **Comunicaciones seguras:** debido al esquema criptográfico de clave pública/privada, se tiene un mecanismo de mensajería cifrada segura de extremo a extremo.

Un aspecto muy importante de los DIDs son sus tipos. Dependiendo de la conexión que se requiera y del tipo de entidad que se sea, puede ser necesario utilizar un DID público o un privado.

DID Público

Los DIDs públicos están destinados a estar disponibles para cualquier persona que lo requiera. Eso sí, al estar público para todo el mundo es posible que se correlacionen. Sin embargo, esto no es un problema. Este tipo de DID es el más adecuado para empresas o gobiernos. Estos DIDs se colocarán en las cadenas de bloques. Se puede observar un esquemático en la Figura 1.7.

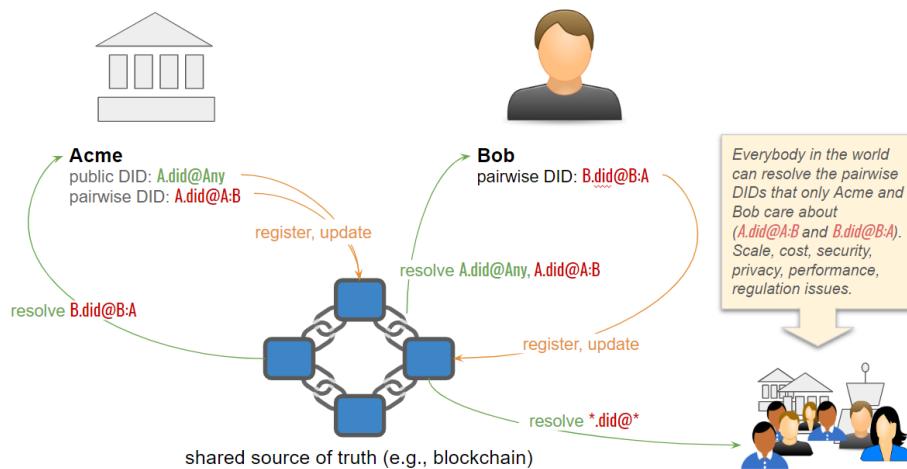


Figura 1.7: DID Público [2].

DID Privado

Este tipo de DID se utiliza para las conexiones entre dos partes. En lugar de publicar el DID en la *blockchain*, las entidades lo envían directamente a las otras partes. Se puede observar un esquemático en la Figura 1.8.

Al tener DIDs privados, se reduce enormemente el tamaño de la cadena de bloques, estando solo los DIDs públicos (y otros datos explicados en la sección dedicada a la *blockchain* (Sección 1.4.5).

Uso de los DIDs en modelo de credenciales verificables

Resumiendo todo lo visto en secciones previas, los DIDs se utilizan de la siguiente manera con credenciales verificables:

1. Por una parte, los DIDs se utilizan para establecer canales de mensajería punto a punto seguros entre agentes. En este uso de DID, no

1.4. Contenidos teóricos para la comprensión de la solución adoptada

14

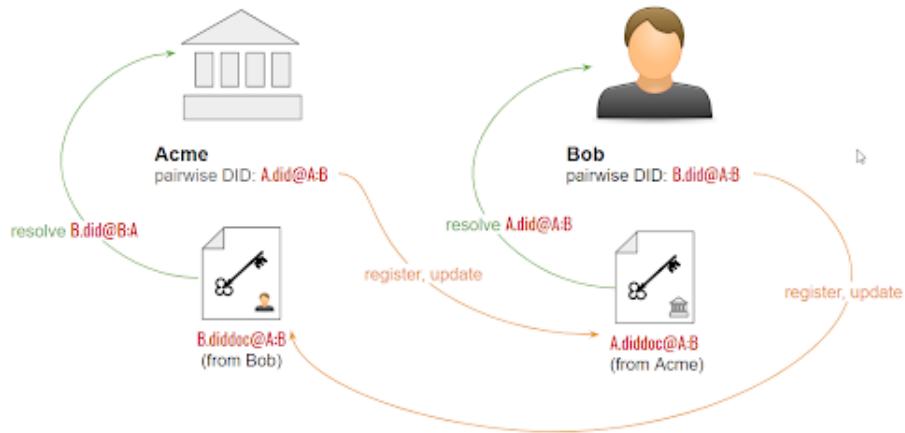


Figura 1.8: DID Privado [2].

hay credenciales verificables involucradas, solo mensajes usando la información en los DIDDocs (claves públicas y puntos finales).

2. Por otra parte, con una credencial verifiable, los DIDs se utilizan como identificador del emisor. El DID del emisor se utiliza para identificar de forma única al emisor y se resuelve para obtener una clave pública relacionada con el DID. Luego, esa clave pública se usa para verificar que los datos en la credencial verifiable realmente provengan del emisor, para verificar la firma criptográfica usando la clave pública del DID.

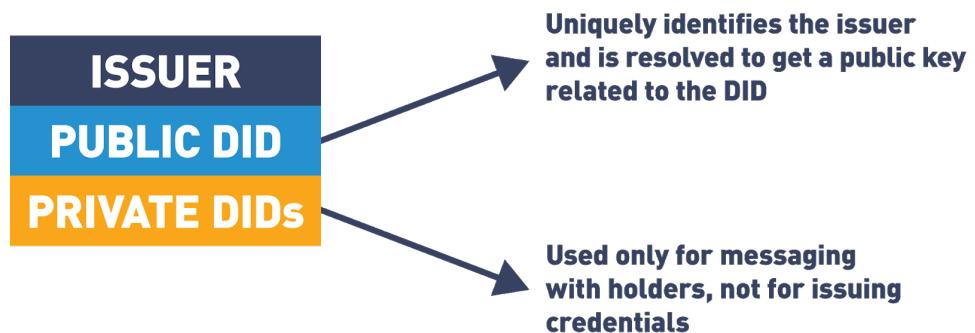


Figura 1.9: Usos de DIDs privados y públicos [2].

En la Figura 1.9 se pueden ver los usos de cada DIDs en el modelo de credenciales verificables, dependiendo de si son privados o públicos. Los públicos se utilizarán para emitir credenciales (ya que identifica públicamente al emisor), y los privados para cada conexión que una entidad pueda llegar a tener.

1.4.4. Agentes

Uno de los componentes más importantes del ecosistema es el propio software que se utiliza para procesar las credenciales verificables y los DIDs.

Un agente es un software que permite a un usuario (persona o entidad) que asume uno o más roles en el modelo de credenciales verificables (emisor, titular y/o verificador) y permite interactuar con otras entidades que también participan en el modelo de credenciales verificables.

Los agentes utilizan DIDs privados (por pares) para asegurar la comunicación *peer-to-peer* (gracias a la criptografía subyacente). Tendrán un DID diferente para cada relación.

Existen diferentes tipos de agentes, dependiendo del uso y necesidad de la entidad:

- **Agentes personales:** para personas individuales. Lo normal es que el agente sea una aplicación móvil debido a su movilidad y facilidad de uso, aunque también pueden ser ejecutados en PCs. El agente permitirá a la persona desde enviar simples mensajes como si fuese una red social (aprovechando los principios de SSI (privacidad, seguridad...)) hasta presentar credenciales o pedirle a personas que verifiquen sus credenciales.
- **Agentes para empresas:** las empresas utilizarán estos agentes tanto para pedirle pruebas de credenciales a los clientes como para emitir sus propias credenciales.
- **Agentes para dispositivos:** no es necesario explicar el creciente aumento de los dispositivos IoT. Estos dispositivos pueden utilizar agentes para emitir credenciales basadas en datos medidos por sus sensores, demostrando que los datos no se manipulan mientras se usa. Véase como ejemplo que los sensores de un coche generen credenciales que rastrean el kilometraje y el mantenimiento del coche, lo que hace que se pueda probar a la hora de pasar la ITV o venderlo a otra persona. Las posibilidades son infinitas.
- **Agentes de enrutamiento:** de forma resumida, estos agentes son intermediarios para otros agentes. Véase como ejemplo la situación de agentes móviles, donde no se puede dirigir un mensaje directamente a un móvil. También son conocidos como mediadores. Por tanto, los mediadores son como un servicio postal: el agente de enrutamiento no sabe nada del mensaje (está encriptado), solo a dónde debe ir este.

1.4.5. Tecnología *blockchain*

Antes de explicar la tecnología *blockchain*, es importante entender la tecnología *Distributed Ledger Technology (DLT)*. Esta es un tipo de estructura de datos que reside en múltiples dispositivos informáticos, generalmente distribuidos en ubicaciones o regiones diferentes.

La tecnología DLT incluye tecnologías como las cadenas de bloques o contratos inteligentes. De forma resumida, la tecnología de contabilidad distribuida generalmente consta de tres componentes básicos:

- Un modelo de datos que captura el estado actual del libro mayor.
- Un lenguaje de transacciones que cambia el estado del libro mayor.
- Un protocolo de consenso para que los participantes decidan qué transacciones serán aceptadas y en qué orden.

Una vez entendida la tecnología DLT, ya se puede explicar como funcionan las cadenas de bloques, que son un tipo de tecnología DLT.

Según Hyperledger [18]: “Una cadena de bloques es un libro mayor distribuido entre pares forjado por consenso, combinado con un sistema para contratos inteligentes y otras tecnologías adicionales. Juntos, estos pueden usarse para construir una nueva generación de aplicaciones transaccionales que establecen confianza, responsabilidad y transparencia en su núcleo, al tiempo que agiliza los procesos comerciales y las restricciones legales”.

Es importante entender algunos conceptos clave sobre la definición:

- Los contratos inteligentes son simplemente programas de computadora que ejecutan acciones predefinidas cuando se cumplen ciertas condiciones dentro del sistema.
- El consenso se refiere a un sistema para garantizar que las partes acuerden un cierto estado del sistema como el estado verdadero.
- Un bloque se refiere a un conjunto de transacciones que se agrupan y se agregan a la cadena al mismo tiempo.
- El sellado de tiempo es otra característica clave de la tecnología *blockchain*. Cada bloque tiene una marca de tiempo, y cada bloque nuevo hace referencia al bloque anterior. Combinada con *hashes* criptográficos, esta cadena de bloques proporciona un registro inmutable de todas las transacciones en la red.

- El registro de un evento, protegido criptográficamente con una firma digital que se verifica, ordena y agrupa en bloques, forma lo que se conoce como transacciones en la cadena de bloques.

La criptografía tiene un papel vital que desempeñar tanto en la seguridad como en la inmutabilidad de las transacciones registradas.

La inmutabilidad de los datos que se encuentran en la cadena de bloques es quizás la razón más poderosa y convincente para implementar soluciones basadas en cadenas de bloques. Una vez que se escribe una transacción en la cadena de bloques, nadie puede cambiarla, o al menos, sería extremadamente difícil.

Esto es debido a que cada bloque está vinculado al anterior al incluir el *hash* del bloque anterior. Este *hash* incluye el *hash* raíz de Merkle de todas las transacciones del bloque anterior. Si una sola transacción cambiase, no solo cambiaría el *hash* de la raíz de Merkle, sino también el *hash* contenido en el bloque modificado. Además, cada bloque subsiguiente tendría que actualizarse para reflejar este cambio. La cantidad de energía requerida para volver a cambiar el *nonce* para este bloque y cada bloque subsiguiente sería prohibitiva.

En la Figura 1.10 se puede ver un ejemplo de intento de modificar la cadena de bloques. En la parte de arriba se ven los bloques originales y las transacciones del Bloque 11. Específicamente se ve la raíz de Merkle para las transacciones en el Bloque 11 que es *Hash ABCD*. Este es el *hash* combinado de las cuatro transacciones en este bloque.

Si alguien intenta modificar, por ejemplo, la Transacción A a A', la acción modificará el *hash* de Merkle a *Hash A'BCD*. Además, el *hash* del Bloque anterior almacenado en el Bloque 12 también debe modificarse ya que habrá cambiado el hash del Bloque 11.

Volviendo al tipo de red, históricamente, las aplicaciones utilizan el esquema cliente-servidor, donde para que un cliente envíe un mensaje a otro usuario en la red, la solicitud debe pasar primero por un servidor central.

Las redes P2P consisten en sistemas informáticos conectados directamente entre sí a través de Internet, sin un servidor central. Los pares contribuyen al poder de cómputo y al almacenamiento que se requiere para el mantenimiento de la red. Este tipo de redes se consideran más seguras que las redes centralizadas, ya que no tienen un único punto de ataque.

En la Figura 1.11 se puede ver la diferencia entre las redes basadas en

BLOCKCHAIN IMMUTABILITY

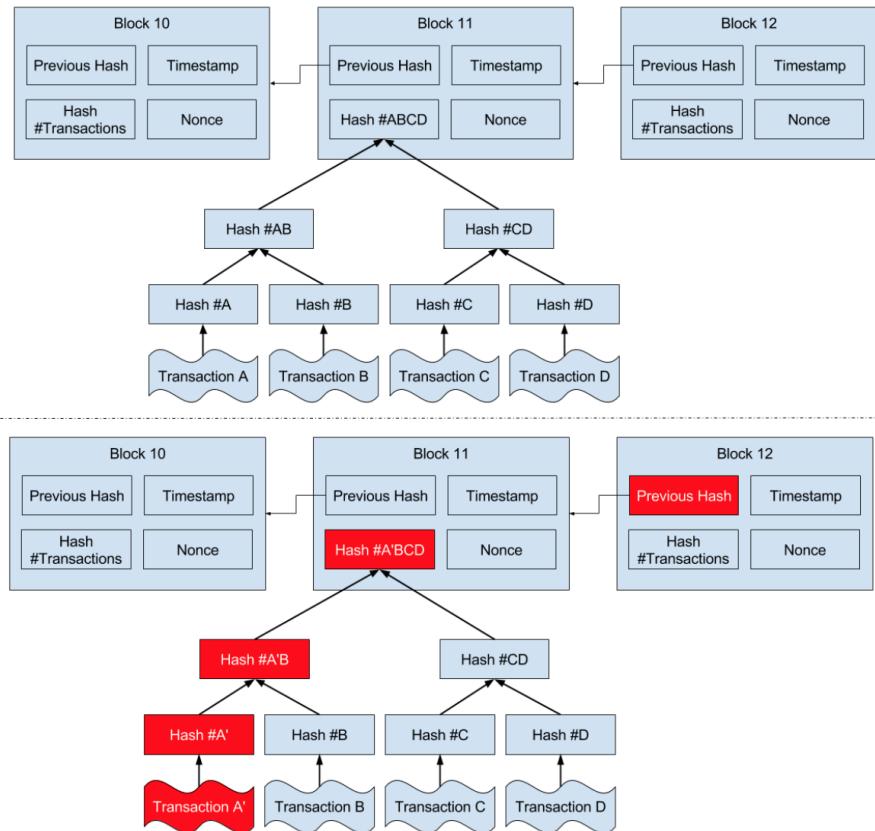


Figura 1.10: Ejemplo de intento de modificar una cadena de bloques [3].

servidor y la redes P2P. En las primeras es necesario pasar primero por un servidor central y en las P2P la conexión entre ‘parejas’ es directa.

1.4.6. Zero-Knowledge Proof (ZKP)

El objetivo de las *Zero-Knowledge Proof (ZKP)* es probar los atributos de una entidad sin exponer un identificador correlacionable. Formalmente, se trata de presentar la credencial sin exponer la clave de la parte que realiza la prueba al verificador. ZKP, por tanto, es un método criptográfico que expone los atributos que requiere el verificador sin proporcionar un identificador que pueda ser correlacionado.

NETWORK ARCHITECTURES

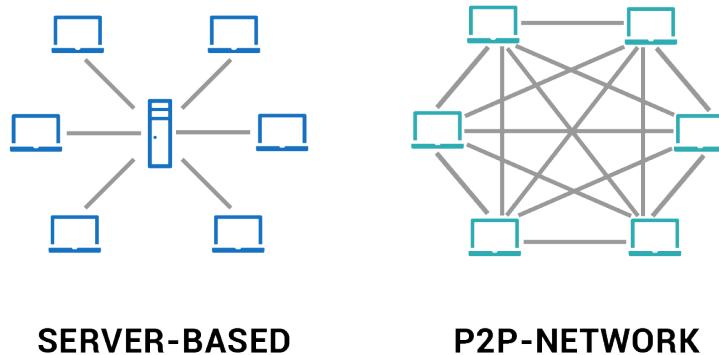


Figura 1.11: Diferencia entre arquitecturas de red [3].

En otros tipos de credenciales que no utilizan ZKP, el titular demuestra al emisor que tiene control sobre un DID, y el emisor lo incluye en la credencial emitida. Cuando el verificador reciba la prueba de la credencial, podrá extraer el DID del titular, lo que es un punto de correlación.

Divulgación Selectiva

El modelo de Indy *AnonCreds* permite algunas funcionalidades muy útiles. Entre ellas está la divulgación selectiva. Esto significa que se mostrarán sólo los datos necesarios y mínimos, sin necesidad de mostrar los datos subyacentes.

Un ejemplo de divulgación selectiva es una persona intentando demostrar que es mayor de edad. Enseñando su carné de conducir, y tapando toda la información excepto su foto y la información correspondiente al emisor (el gobierno), se demuestra que es mayor de edad, y no ha revelado ni su nombre, ni fecha de nacimiento, ni edad, sólo que es mayor de edad.

1.4.7. Revocación

La revocación de una credencial verificable es la capacidad de un emisor de publicar que una credencial verificable ya no está activa. Esta acción la hace el emisor (y, opcionalmente, avisa al emisor).

Que una credencial haya sido revocada no tiene por qué ser el resultado de algo ‘malo’ hecho por el titular. Simplemente puede ser porque se ha

1.4. Contenidos teóricos para la comprensión de la solución adoptada

publicado una nueva versión de la credencial (véase la renovación del DNI, donde el antiguo queda inválido).

La revocación ZKP normalmente usa una ID de credencial que sólo el titular y el emisor saben, y un registro de revocación. En este caso, el verificador no tiene un ID, por lo que no hay correlación. En su lugar, se genera una prueba de no revocación, que el verificador puede comprobar con los datos publicados en el registro de revocación.

Capítulo 2

Planificación y costes

2.1. Planificación del proyecto

A continuación se muestra la planificación seguida durante la realización del proyecto, desde el mes de julio de 2021 que es cuando se llegó a un acuerdo con el tutor hasta el mes de junio de 2022 que es cuando se ha terminado de elaborar la memoria:

1. **Fase de Investigación (julio 2021 - octubre 2021):** durante esta fase se ha realizado una investigación de las tecnologías que se iban a usar en el proyecto, con el objetivo de dominar el tema. Se propusieron las siguientes tareas:
 - Estudio del modelo de identidad SSI y de su actual uso.
 - Estudio de la tecnología *blockchain* y su relación con SSI.
 - Estudio de las herramientas a utilizar durante el proyecto: Docker, Virtual Box, Visual Studio Code y Linux.
2. **Fase de Formación (noviembre 2021 - diciembre 2021):** el objetivo de esta fase es, una vez entendido los conceptos, formarse en ellos. Para ello se realizaron diversos cursos:
 - Curso de Python disponible en Udemy [19].
 - Curso “*Introducción a las tecnologías Blockchain de Hyperledger*” [3].
 - Curso “*Introducción a las soluciones de Hyperledger para SSI basado en la tecnología Blockchain: Indy, Aries y Ursa*” [2].
 - Curso “*Convertirse en un Desarrollador Aries*” [8].
3. **Pausa por exámenes (enero 2022):** durante el mes de enero se realizó un descanso del TFG para centrarse en los últimos exámenes del curso.

4. **Fase de Análisis y Diseño (febrero 2022 - marzo 2022):** durante esta fase se empieza a diseñar el entorno a implementar. Para ello hay que elegir, de entre las soluciones que Hyperledger ofrece, las más adecuadas. Las tareas propuestas para esta fase son:

- Elección del *framework*. Finalmente se acaba escogiendo *Aries Cloud Agent - Python (ACA-Py)*.
- Elección de la cadena de bloques a utilizar. La elección tomada es *von-network*.
- Diseño del esquema. Se definen los roles de los agentes, así como su interconexión.

5. **Fase de Desarrollo e Implementación (abril 2022 - mayo 2022):** en esta fase, una vez con el diseño acabado, se desarrolla el entorno de pruebas con el objetivo de hacer una evaluación del rendimiento, así como el entorno de evaluación de funcionalidades. Finalmente se implementará y ejecutarán los entornos finales con el objetivo de analizar los resultados en la siguiente fase. Las tareas propuestas son:

- Modificación y desarrollo de la solución disponible. A partir del repositorio de GitHub de ACA-Py se elaboran las modificaciones necesarias para las necesidades del proyecto.
- Desarrollo de los entornos de pruebas. Se elaborarán *scripts* mediante el lenguaje Python para el despliegue de los diferentes escenarios.
- Comienzo de la elaboración del informe.
- Implementación y ejecución de los entornos de pruebas. Del entorno de pruebas de rendimiento se guardan todos los resultados en ficheros *logs* para analizarlos posteriormente.

6. **Fase de Conclusiones y Documentación (junio 2022):** durante esta última fase se intenta llegar a una conclusión final y se termina la elaboración del informe para presentarlo ante tribunal.

Para una comprensión visual se muestra el Diagrama de Gantt en la Figura 2.1.

Con respecto a las horas dedicadas al proyecto, se ha estimado una dedicación aproximada de 480 horas al proyecto (Tabla 2.1), dedicadas especialmente en los meses de mayo y junio.

La asignatura “Trabajo de Fin de Grado” es de 12 ECTS, por lo que su dedicación a esta ha de ser de 300h, siendo la de 1 ECTS, 25h. Por tanto se han cumplido las horas estimadas de dedicación al proyecto con creces.

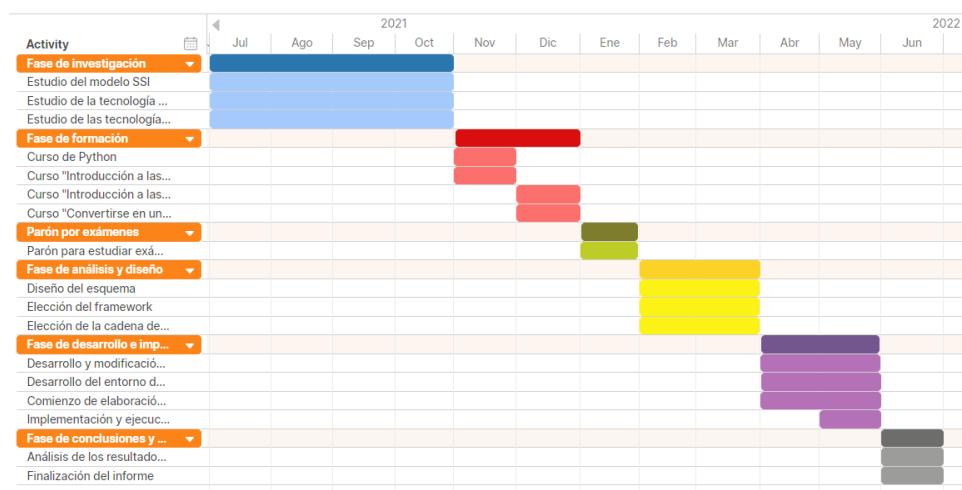


Figura 2.1: Diagrama de Gantt.

Fase	Tiempo dedicado (h)
Investigación	80
Formación	80
Análisis y Diseño	100
Desarrollo e Implementación	120
Conclusiones y Documentación	100
TIEMPO DEDICADO TOTAL	480

Tabla 2.1: Tiempo dedicado al proyecto.

2.2. Costes asociados al proyecto

A parte del tiempo dedicado al proyecto, este también ha tenido costes asociados. A continuación, se explican los costes que ha tenido el proyecto.

2.2.1. Recursos humanos

Para el cálculo de los costes asociados a los recursos humanos se ha estimado que el alumno Rafael Adán López se puede considerar Graduado en Ingeniería de Tecnologías de Telecomunicación, con un coste asociado de 20€/h. Por otra parte, al tutor Gabriel Maciá Fernández, Doctor y Profesor Titular en la Universidad de Granada, se le ha estimado un coste de 40€/h. Los gastos totales en recursos humanos, como se puede ver en la Tabla 2.2, son de 11.200€.

Persona	Título/Puesto	Coste asociado (€/h)	Tiempo dedicado (h)	Coste total (€)
Rafael Adán López	Ingeniero Técnico de Tecnologías de Telecommunicación	20	480	9.600
Gabriel Maciá Fernández	Doctor en Telecomunicaciones y Profesor Titular	40	40	1.600
COSTE TOTAL ASOCIADO A RECURSOS HUMANOS				11.200

Tabla 2.2: Costes asociados a recursos humanos.

2.2.2. Recursos software

Durante la realización del proyecto se ha intentado optar siempre por software open-source con el objetivo de minimizar los costes del proyecto. Es por ello que herramientas como Virtual Box y Docker, que son open-source, no han tenido coste alguno.

Sin embargo se ha hecho uso de algunos programas de pago. Por una parte, se ha utilizado el paquete Microsoft Office Familiar para la elaboración de figuras y la presentación PowerPoint, así como el análisis de datos con Excel, con un coste asociado de 100€. Por otra parte, se ha utilizado el sistema operativo Windows 10 Home, con un coste asociado de 145€ (ambos precios sacados de la web de Microsoft [20]).

Por último, se ha hecho uso de 3 cursos de la plataforma de recursos en línea edX [21]. Ambos se pueden acceder de forma gratuita, pero dan la opción de tener más recursos, acceso a actividades de evaluación y a un certificado verificado final, cuyo valor para el CV es de gran utilidad. El coste de cada curso tomando esta opción es de 190€ cada uno. Los cursos tomados son:

- *LFS171x: An Introduction to Hyperledger Blockchain Technologies* [3].
- *LFS172x: Identity in Hyperledger: Indy, Aries and Ursa* [2].
- *LFS173x: Becoming a Hyperledger Aries Developer* [8].

El coste total de los recursos software se puede ver en la Tabla 2.3, que es de 815€.

2.2.3. Recursos hardware

Durante la realización del proyecto se ha utilizado el portátil HP OMEN 15-DC0009NS, cuyo valor es de 1199€ (según PcComponentes [22]).

Las características del ordenador se recopilan en la Tabla 2.4.

Recurso	Coste asociado (€)
Sistema operativo Windows 10 Home	145
Paquete Microsoft Office Familiar	100
Cursos formativos edX	3 x 190
COSTE TOTAL ASOCIADO A RECURSOS SOFTWARE	815

Tabla 2.3: Costes asociados a recursos software.

Componente	Características
Procesador	Intel® Core™ i7-8750H (frecuencia de base de 2,2 GHz, hasta 4,1 GHz con tecnología Intel® Turbo Boost, 9 MB de caché, 6 núcleos)
Memoria RAM	SDRAM DDR4-2666 de 16 GB (1 x 16 GB)
Disco duro	1 TB 7200 rpm SATA + 256 GB PCIe® NVMe™ M.2 SSD
Controlador gráfico	NVIDIA® GeForce® GTX 1050 (GDDR5 de 4 GB dedicados)
Sistema operativo	Windows 10 Home 64 Bits
Dimensiones	36 x 26.3 x 2.5 cm

Tabla 2.4: Características del ordenador utilizado.

2.2.4. Coste total del proyecto

Tras haber analizado los recursos utilizados, es posible estimar el coste total asociado al proyecto. El coste total estimado es de 13.214€, como se observa en la Tabla 2.5. A este coste total se le añade un 10% de coste indirectos asociados: luz, gasolina para ir a la biblioteca a realizar el proyecto, suministros, etc. Sumado esto, el coste final del Trabajo Fin de Grado (TFG) es de 14.535€.

Recursos	Coste asociado (€)
Humanos	11.200
Software	815
Hardware	1.199
COSTE TOTAL ASOCIADO	13.214

Tabla 2.5: Coste total asociado.

Capítulo 3

Análisis del problema

En esta sección se pretende dar una visión global del problema. Primero, se especificarán los requisitos planteados para el proyecto. Seguidamente, se compararán las soluciones ya existentes basadas en *blockchain* que implementan el esquema de identidad auto-soberana. Finalmente, se escogerá una de las opciones y se justificará adecuadamente la elección.

3.1. Posibles soluciones existentes de identidad auto-soberana

En este apartado se estudiarán tres soluciones basadas en *blockchain* que pueden satisfacer los requisitos de las leyes de identidad y brindar un entorno SSI al usuario. Para más información sobre estas soluciones, consultar los artículos [5] y [23].

3.1.1. Alastria

Alastria es un sistema español basado en *blockchain* y que usa la plataforma Ethereum. Su objetivo es que las empresas digitalicen sus activos. Para ello, se establecen representaciones digitales de los activos, denominados ‘tokens’. Estos tokens se pueden utilizar para poner en marcha nuevos productos y servicio.

Alastria tiene en curso diversos proyectos relacionados con cadenas de bloques. El proyecto que interesa para el trabajo de investigación es ID_Alastria [4].

ID_Alastria es un modelo de identidad digital para su uso en servicios digitales, incluso más allá de la propia tecnología *blockchain* e inspirado en el

28 3.1. Posibles soluciones existentes de identidad auto-soberana

concepto SSI. Este proyecto es el foco principal de Alastria en la actualidad. Este permite a los usuarios tener el control total sobre sus datos personales. Se busca la privacidad y seguridad de los datos, así como el control de los datos por parte del usuario.

Alastria ha basado su modelo en SSI, como se puede observar en la Figura 3.1, en los pilares de seguridad, controlabilidad y portabilidad.



Figura 3.1: Principios SSI según Alastria [4].

Por otra parte, en la Figura 3.2 se observa el modelo de ID_Alastria. Este modelo es ya conocido en el campo de SSI y *blockchain*. Se observan tres entidades:

- **Emisores:** certifican o emiten las credenciales para los usuarios.
- **Proveedores de servicios:** proveen de servicios a los usuarios. Normalmente le piden datos a los usuarios para acceder a estos servicios.
- **Usuarios:** son los usuarios finales que poseen los datos. Estos deciden cuando, cómo, cuáles y con quién compartirlos.

Finalmente, la infraestructura de este modelo consta de los siguientes tres elementos:

1. **Aplicación:** aplicación a través de la cual el usuario interactúa con el ecosistema. En esta aplicación se almacenan las claves y los datos personales.
2. **Repositorio o wallet:** ubicado fuera de la *blockchain* (normalmente fusionado con la aplicación), se almacenan datos de información del modelo: claves, alegaciones, testimonios y documentos asociados.

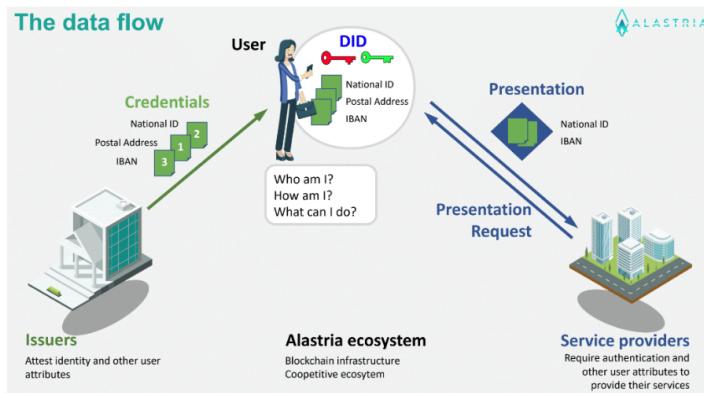


Figura 3.2: El modelo de ID_Alastria [5].

3. **Cadena de bloques:** en esta se almacenan la clave pública del usuarios y los *hashes* de los testimonios y alegaciones.

Esto se puede ver con más detalle en la Figura 3.3.

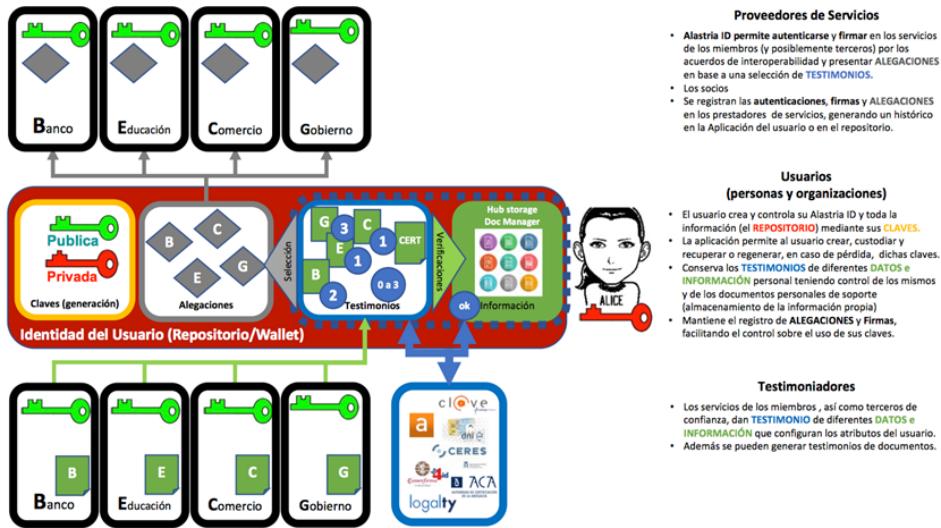


Figura 3.3: Infraestructura de ID_Alastria [5].

3.1.2. Hyperledger

Hyperledger es un proyecto open-source creado por Linux Foundation, activo desde 2015. Su objetivo es crear soluciones *blockchain* de código abierto para empresas. Hyperledger, gracias a la tecnología *blockchain*, proporciona características necesarias para los principios de SSI: registro único,

30 3.1. Posibles soluciones existentes de identidad auto-soberana

inmutabilidad y robustez.

Hyperledger tiene en marcha diversos proyectos, los cuales se pueden observar en la Figura 3.4. Los señalados en rojo son en los que se va a hacer mayor hincapié porque son los basados en identidad.

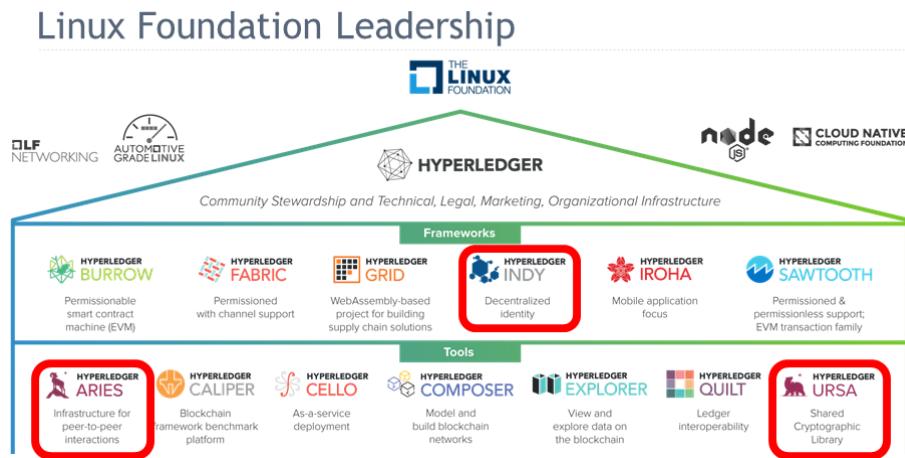


Figura 3.4: *Frameworks* y herramientas de Hyperledger [2].

Hyperledger Indy

Hyperledger Indy es un libro mayor distribuido que se diseñó específicamente para la identidad descentralizada. Éste proporciona herramientas, bibliotecas y componentes reutilizables para proporcionar identidades digitales arraigadas en cadenas de bloques u otros libros mayores distribuidos para que sean interoperables. Permite, entonces, que las personas puedan administrar y controlar sus identidades digitales.

Hyperledger Aries

Hyperledger Aries es una librería que proporciona un kit de herramientas compartido, reutilizables e interoperable diseñado para iniciativas y soluciones centradas en la creación, transmisión y almacenamiento de credenciales digitales verificables.

Es de vital importancia saber diferenciar entre Indy y Aries. Aries es una infraestructura para interacciones *peer-to-peer* basadas en *blockchain*; no es una *blockchain* ni una aplicación. Aries se centra en los agentes que utilizan

la *blockchain*, la cual es creada/implementada por Indy.

El objetivo final de Hyperledger Aries es proporcionar un conjunto dinámico de capacidades para almacenar e intercambiar datos relacionados con la identidad basada en *blockchain*. Estas capacidades varían desde el almacenamiento seguro y secreto de datos, como claves privadas, hasta la capacidad de datos accesibles globalmente que cualquier persona puede ver y acceder.

Hyperledger Ursula

El proyecto Ursula produce paquetes criptográficos que se pueden usar para crear aplicaciones de nivel superior. Entre sus funcionalidades, están:

- Generación de pares de claves pública/privada.
- Cifrado y descifrado de datos.
- Firma y verificación de datos.

Fundación Sovrin

Es posible que el lector haya oído hablar de Sovrin y la Fundación Sovrin [24]. Sovrin proporciona un ecosistema de identidad soberana y de confianza descentralizada de tipo open-source. Es una cadena de bloques autorizada dirigida por la Fundación Sovrin. Esta fundación es una organización sin ánimo de lucro que ha organizado la primera implementación de Hyperledger Indy. Por tanto, es una implementación **pública** de un sistema de identidad auto-soberana.

Características y funcionalidades de Hyperledger Indy

Los aspectos básicos de que tecnologías/esquema utiliza Hyperledger se pueden consultar en la Sección 1.4. A modo de resumen, las características principales de Hyperledger Indy son las siguientes [5]:

- *Ledger* distribuido, diseñado para SSI.
- Resistente a correlación.
- Uso de DIDs, lo que elimina la necesidad de una autoridad centralizada.
- Identificadores paritarios (DIDs privados) que establecen conexiones seguras 1:1.
- Credenciales verificables, actualmente estandarizado por W3C.

32 3.1. Posibles soluciones existentes de identidad auto-soberana

- ZKP, que se trata de un mecanismo para revelar lo mínimo y estrictamente para un verificador, cumpliendo principio de divulgación mínima.

3.1.3. uPort

uPort es un sistema de identidad descentralizada (y soberana) basado en *blockchain* de la plataforma Ethereum [25]. Además es de código abierto.

Las identidades de uPort son creadas y manejadas por los clientes de uPort (como su aplicación móvil). Las identidades son propiedad del creador y no depende de ninguna entidad central.

Las características principales de uPort son:

- Identidad basada en la tecnología Ethereum.
- Inicio de sesión sin necesidad de contraseña a partir de código QR.
- Gestión simplificada de claves.
- Vinculación de firmas digitales.
- Sistema de reputación de usuario.
- Verificación de la identidad.
- Soporte para cadenas de bloques privadas.
- La información del usuario nunca entra en la *blockchain*.

Arquitectura

La arquitectura de uPort consiste en tres elementos principales:

1. **Contratos inteligentes:** certifican la identidad del usuario y permiten recuperar su identidad en caso de pérdida o robo del dispositivo.
2. **Aplicación móvil:** contiene las claves del usuario y le permite comunicarse con el *smart contract*. Las claves permanecen siempre dentro del dispositivo y no hay manera de exportarla.
3. **Librerías para desarrolladores:** mecanismo mediante el cual los desarrolladores de aplicaciones de terceros integrarían uPort en sus aplicaciones.

El identificador de uPort es una cadena hexadecimal de 20 bytes. Este es global, único y permanente. Este identificador es también conocido como contrato *proxy*.

El contrato *proxy* sirve para emitir las transacciones y es el mecanismo para interactuar con otros contratos inteligentes en la *blockchain* de Ethereum. Cuando el usuario quiere identificarse e interactuar con otro contrato inteligente, se envía una transacción mediante un contrato controlador, que contiene la lógica de acceso. Después, el contrato *proxy* reenvía esta transacción al contrato inteligente de la aplicación (Figura 3.5)

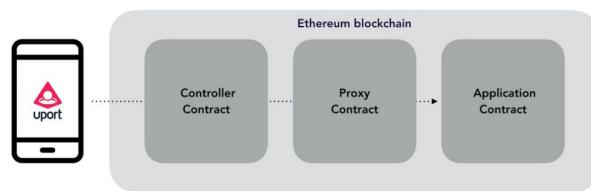


Figura 3.5: Arquitectura de uPort [6].

Esta arquitectura permite ver la dirección del contrato *proxy* como la entidad que interactúa. El propósito de contar con un contrato *proxy* como ID central es que permite al usuario cambiar su clave privada en cualquier momento (sea el caso, por ejemplo, de pérdida o robo del dispositivo). Si el identificador fuera la clave pública correspondiente a la clave privada, perdería el control sobre su identificador al perder el dispositivo.

3.2. Discusión, comparación y solución escogida

Una vez presentadas las posibles soluciones, se va a realizar una comparación entre ellas con el fin de elegir la mejor solución acorde al proyecto. En la Tabla 3.1 se pueden ver esta comparación. La explicación de las características es:

- **Tipo de blockchain:** el tipo de *blockchain* sobre el cuál está basado el sistema de identidad auto-soberana.
- **Minado:** si es necesario un algoritmo de consenso para validar los datos de la *blockchain*.

- **Gestión de claves:** cómo se gestionan las claves privadas de los usuarios.
- **Importación de datos de identidad:** si se le permite al usuario crear atributos.
- **Divulgación selectiva:** si se comparten los mínimos datos posibles.
- **Almacenamiento de datos:** cómo se almacenan los datos del usuario.
- **Confianza requerida:** si se requiere la presencia de algún organismo de confianza.
- **Smart Contracts:** si hace uso de contratos inteligentes.

Características\Solución	ID Alastria	Hyperledger/Sovrin	uPort
Tipo de Blockchain	Semi-pública - permisionada	Permisionada	Pública
Minado	No	No	Sí
Gestión de claves	Usuario y DPKI	DPKI	Usuario y DPKI
Importación de datos de identidad	?	Sí	Sí
Divulgación selectiva	Sí	Sí	Sí
Almacenamiento de los datos	Dentro y fuera de la <i>blockchain</i>	Dentro y fuera de la <i>blockchain</i>	fuerza de la <i>blockchain</i>
Confianza requerida	Sí	Sí	No
<i>Smart Contracts</i>	Sí	No	Sí

Tabla 3.1: Características de las soluciones y comparación

Analizando las comparaciones anteriores, la solución a escoger es Hyperledger, debido a:

- La cadena de bloques usada es permisionada, por lo que no es necesario un minado.
- La gestión de claves.
- Cumple el principio de divulgación selectiva.
- Cuenta con numerosos repositorios y documentación, así como una gran comunidad que aporta al proyecto.

Aunque Alastria también es una buena alternativa, es una *blockchain* de carácter nacional, y está bastante lejos del gran desarrollo, apoyo y avance con el que cuenta Hyperledger.

3.3. Análisis de la solución escogida

Una vez se ha decidido que se va a utilizar el proyecto Hyperledger, hay que elegir entre las posibles soluciones que este proyecto ofrece.

3.3.1. *Framework* de los agentes

Es necesario escoger un *framework* para los agentes, que determinará el comportamiento de estos y como interactúan con el *ledger* y otros agentes. Normalmente, en el mundo de código abierto, los *frameworks* se identifican principalmente por su lenguaje de implementación. Por el momento, hay cuatro *frameworks* principales que implementan Aries. De forma resumida, son:

- **aries-cloudagent-python (ACA-Py)**: es adecuado para todas las aplicaciones de agentes no móviles y tiene implementaciones de producción. En este, ACA-Py y un controlador se ejecutan juntos y se comunican a través de una interfaz HTTP. El controlador se puede escribir en cualquier lenguaje que pueda enviar y recibir solicitudes HTTP, que es prácticamente cualquier lenguaje.
- **aries-framework-dotnet (AF-.NET)**: se puede usar para crear agentes móviles y el lado del servidor y también tiene implementaciones de producción. Muchos de los agentes Aries actuales en Play Store se basan en AF-.NET. Trinsic Studio de Trinsic es un ejemplo de aplicación de credenciales verificables construida en AF-.NET.
- **aries-framework-go (AF-Go)**: es un marco Golang que implementa una arquitectura similar a la de ACA-Py.
- **aries-framework-javascript (AFJ)**: es un marco bastante nuevo que se está desarrollando con el objetivo de admitir la creación de aplicaciones móviles basadas en un marco móvil JavaScript como React Native. También se está creando para ejecutarse en el lado del servidor con Node-JS.

Todos son opciones interesantes y que darían mucho de sí para el proyecto. Sin embargo, todo excepto ACA-Go admiten AIP 1.0 y solo ACA-Py y AF-Go admiten AIP 2.0 (esto se explicará con más detalle en la Sección 4.6).

Debido a que ACA-Py admiten ambos AIP, que el lenguaje utilizado es Python y que es uno de los *frameworks* más antiguos y por lo tanto con más documentación, es este el que se escogerá para el proyecto. Se describirá su arquitectura y sus funcionalidades en la Sección 4.5.

3.3.2. Cadena de bloques

Hyperledger Indy implementa (basada en la criptografía de Ursu) una cadena de bloques especialmente diseñada para la identidad en Internet, lo

que permite tener certeza sobre quién está hablando con quién en una transacción digital.

Evidentemente, la cadena de bloques es muy importante, sin embargo, muchos desarrolladores pierden mucho tiempo intentando entender como funciona Indy, lo que, si no se va a desarrollar específicamente en él, no es útil.

El objetivo de la cadena de bloques es que los emisores de credenciales verificables publiquen sus claves criptográficas y metadatos de credenciales para que un titular de una credencial pueda probar criptográficamente que es poseedor de esa credencial.

Los atributos que hacen a la cadena de bloques ideal para Aries son:

- Los datos escritos en la *blockchain* son inmutables (no se pueden cambiar).
- Los datos de la cadena de bloques no se pueden cambiar. De esta manera el emisor no puede borrar los datos (claves públicas y metadatos de credenciales) que harían que la credencial del titular quedase inválida de verificarse.
- Múltiples partes (validadores) llegan a un consenso para escribir en la cadena, previniendo que se escriban ataques maliciosos.
- Los datos se replican a través de diversos nodos, por lo que está altamente disponible.

La *blockchain* de Indy

Como todas las cadenas de bloques, la *blockchain* de Indy es immutable: los datos se escriben una vez y nunca se modifican.

Tiene múltiples nodos operados de forma independientes que escriben transacciones en el libro mayor. Estos nodos se comunican para acordar (llegar a un consenso) sobre qué transacciones deben escribirse y en qué orden.

Existen varios tipos de cadenas de bloques, clasificadas según su acceso y validación, como se puede ver en la Figura 3.6.

Cadenas de bloques muy conocidas son Bitcoin y Ethereum: son redes públicas sin permiso. Esto significa que cualquier puede acceder a ellas (público) y cualquiera puede participar en el proceso de validación (sin permiso). De este proceso de validación surge la conocida minería de Bitcoin.

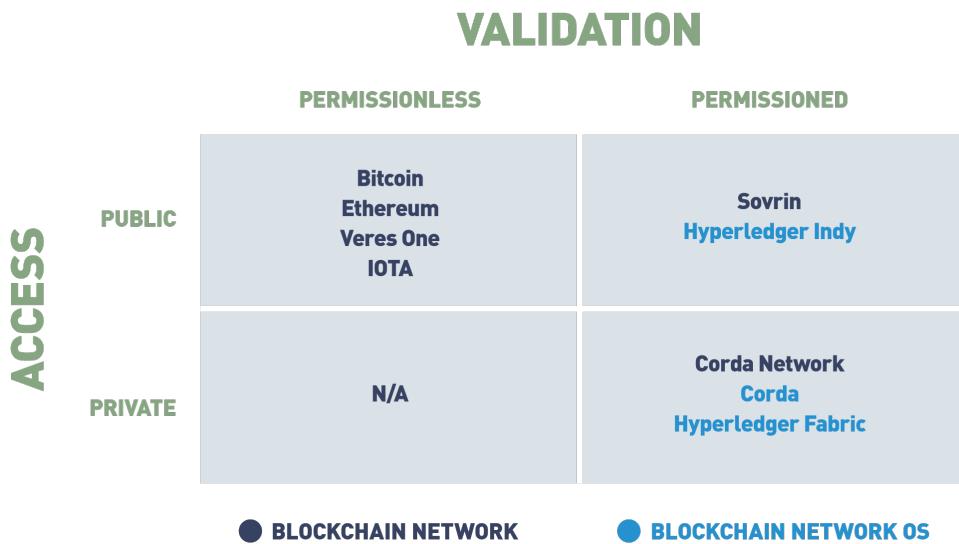


Figura 3.6: Tipos de *blockchain* [2].

También se pueden observar redes privadas y autorizadas como Fabric (otro proyecto de Hyperledger).

Indy se encuentra entre estos dos modelos. Está diseñado para funcionar de manera que todos puedan ver su contenido (público), pero solo los participantes aprobados previamente, conocidos como *stewards*, puedan participar en el proceso de validación (autorizados). En lugar de competir por la recompensa por escribir en el siguiente bloque como en *blockchain* públicas (Bitcoin y Ethereum), los nodos de Indy llegan a un consenso para evitar comportamientos defectuosos o maliciosos.

El hecho de que la *blockchain* de Indy sea pública impone una restricción sobre qué datos colocar en una cadena de bloques de Indy. Por tanto, y es muy importante, **no hay datos privados en la cadena de bloques de Indy**. Ni aunque estén encriptados. Esto se debe a que este proyecto tiene como objetivo que dure bastante en el futuro, y no se sabe si los algoritmos utilizados actualmente serán rotos en el futuro.

Antes se ha hablado de que los administradores llegan a un algoritmo de consenso. Específicamente, Indy usa Plenum, una implementación de un algoritmo de tolerancia a fallas bizantinas (BFT).

¿Qué hay dentro de la *blockchain*?

Como ya se ha mencionado, no hay datos privados en la *blockchain*. Dichos datos privados se almacenan en las credenciales verificables que van

directamente a las *wallets* de los agentes.

Entonces, ¿qué hay en la cadena de bloques y por qué es necesaria? Los datos que se almacenan en la *blockchain* se pueden visualizar en la Figura 3.7.

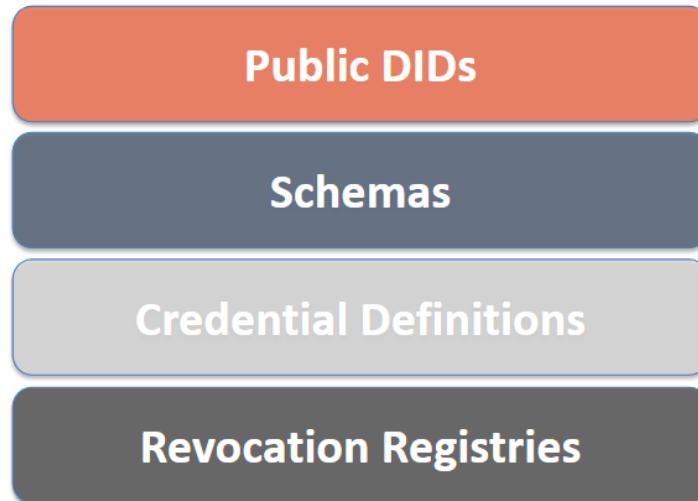


Figura 3.7: Qué hay dentro de la *blockchain* [2].

Las funciones de cada tipo de dato son:

- **DIDs públicos:** si un emisor quiere emitir una credencial, debe tener un DID en la cadena de bloques que permita a los verificadores averiguar quiénes son y determinar si puede confiar en este.
- **Esquemas:** además del DID, el emisor debe publicar un esquema en la cadena de bloques. Un esquema sirve para indicar que atributos tendrá una credencial a emitir. Una ventaja que tiene esto es que, si una credencial es muy común, el esquema puede reutilizarse por otros emisores, evitando así sobrecargar de esquemas la *blockchain*.
- **Definición de credencial:** antes de emitir la credencial, se debe publicar una definición de credencial en el *ledger*. La definición de credencial dice qué DID usa el emisor, qué esquema, y las claves públicas utilizadas (para cada atributo).
- **Registros de revocación:** este último tipo de dato es opcional. Como se explica en la Sección 1.4.7, existe la posibilidad de revocar credenciales. Por tanto, si se quiere dar posibilidad a una credencial de que se pueda revocar, hay que subir al *ledger* un registro de revocación vinculado a la definición de la credencial.

Elección de cadena de bloques

Una vez visto como funciona y que rol tiene la cadena de bloques en la solución, hay que elegir cómo implementar la *blockchain*.

Son dos las opciones posibles a la hora de utilizar la cadena de bloques:

- Ejecutar una red local Indy usando *von-network*, una red Indy ‘pre-empaquetada’ construida por el equipo del Gobierno de British Columbia (BC).
- Utilizar un *sandbox* de red pública de Indy ya existente. Las redes como BCovrin del gobierno de British Columbia están abiertas para que cualquiera las use y son bastante confiables.

Ambas opciones son válidas. A primera vista puede parecer que utilizar una red local puede significar trabajo adicional para el desarrollador. Sin embargo, la usabilidad de *von-network* y el mayor control que esta ofrece hace que sea la opción escogida para el proyecto.

Capítulo 4

Diseño

En este capítulo se definirá la arquitectura del sistema propuesto, así como los roles del sistema y los protocolos usados entre ellos para la comunicación. Se explicará los dos entornos diseñados: entorno de prueba de funcionalidades y entorno de evaluación de rendimiento.

4.1. Introducción

En la Figura 4.1 se puede observar un esquema conceptual del sistema diseñado. En primer lugar, se implementará con la herramienta Virtual Box una máquina virtual con el sistema operativo Ubuntu 20.04 LTS.

Una vez desplegada la máquina virtual, se utilizará la herramienta de contenerización Docker para ejecutar los contenedores que se ven en la Figura 4.1:

- **La cadena de bloques**, que se implementará mediante una red local. Se utilizará *von-network*, una instancia de una red Indy ‘preempaquetada’.
- **El servidor de archivos de revocación**, donde se publicarán los archivos de revocación para poder verificar si una credencial ha sido revocada. Se utilizará el proyecto *indy-tails-server*, desarrollado por el Gobierno de British Columbia.
- **Los agentes Alice, Faber y Acme**, que interactuarán entre ellos. Realizarán desde acciones sencillas como mandar un mensaje básico hasta emitir y presentar credenciales.
- **Los scripts para implementar el entorno de evaluación de rendimiento**. Estos automatizarán la ejecución de diversos escenarios con el objetivo de sacar datos y conclusiones.

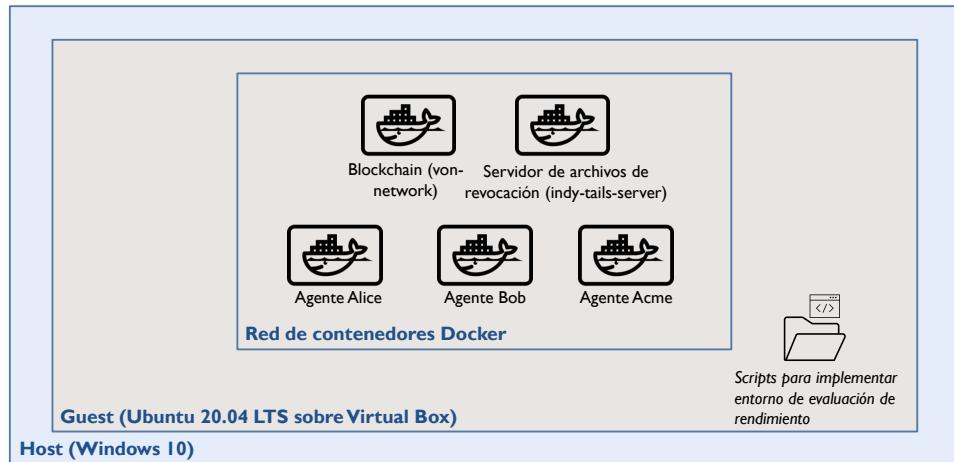


Figura 4.1: Esquema del sistema diseñado.

Este esquema muestra las entidades/elementos que componen el diseño. En las siguientes secciones se irá explicando cada elemento con detalle. Finalmente, se explicarán los diseños de los dos entornos creados en las secciones 4.7 y 4.8.

4.2. Roles de los agentes

A lo largo del proyecto, se utilizarán algunos personajes ya conocidos en la nomenclatura de SSI para referirse a los emisores, titulares y verificadores. Por tanto, es importante introducirlos antes de empezar a explicar el diseño del proyecto:

1. **Alice**: es una persona que tiene una aplicación donde recibe y almacena credenciales, las cuáles utiliza para probar su identidad online. Alice ha acabado una carrera universitaria e intentará aplicar a un puesto de trabajo (en la empresa Acme) utilizando su título universitario.
2. **Faber**: se trata de la universidad en la que ha estudiado Alice, por lo que le emitirá una credencial de título de universidad.
3. **Acme**: es una compañía para la que Alice quiere trabajar. Esta empresa, para poder contratarla, le pedirá a Alice una prueba de que ha acabado la carrera en la universidad Faber.

4.3. *Von-network*

Como se analizó en la Sección 3.3.2, se va a utilizar el proyecto *von-network* para desplegar una red Indy local.

Von-network es una red Indy preparada para su uso creada por el Gobierno de British Columbia [26]. Es la manera más fácil de implementar una instancia de Hyperledger Indy. Todo el contenido está en el repositorio del Gobierno de British Columbia [27].

Este proyecto incluye las siguientes características:

- Imágenes de contenedores de Indy actualizadas.
- Una interfaz web que permite navegar por las transacciones de la cadena de bloques.
- Una API para acceder al fichero génesis de la red (fichero que contiene la información necesaria para conectarse a la cadena de bloques).
- Un formulario web para registrar DIDs en la red.
- Despliegue de 4 nodos que participarán en el consenso de la red.

En la Figura 4.2 se puede ver el esquema conceptual de la red *von-network*. Se puede ver como los 4 nodos participan en un consenso para decidir que transacciones entran al *ledger* y en qué orden. Además, cada nodo tendrá una copia del *ledger*, incrementando la disponibilidad.

El funcionamiento es el siguiente:

- Coexisten 4 nodos (aunque se pueden añadir nuevos), los cuales cada vez que hay que añadir información al *ledger* se llega a un consenso.
- Cada nodo tiene una réplica de la cadena de bloques, lo que hace que la disponibilidad se incremente.
- Para acceder a los nodos, existen los Access Point (AP). Para que los agentes se conecten al AP (y en consecuencia a la red), existe el fichero de génesis, el cuál indica el punto de conexión de cada nodo de la red (dirección IP y puerto).

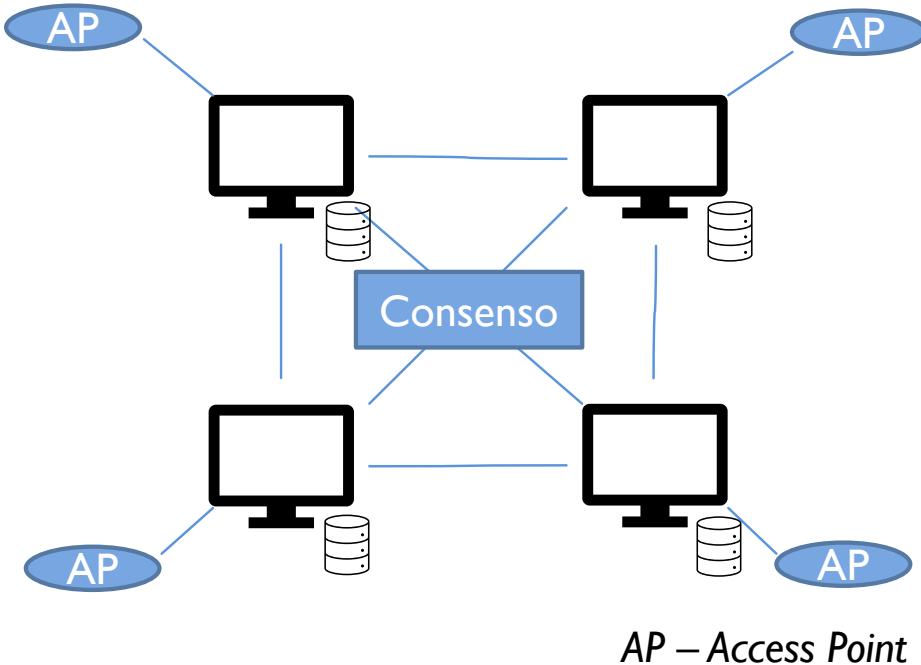


Figura 4.2: Esquema de la red *von-network*.

4.4. Servidor de registros de revocación: *indy-tails-server*

Para implementar el servidor de revocación se utilizará el proyecto *indy-tails-server* localizado en el repositorio citado [28], también desarrollado por el Gobierno de British Columbia.

Indy Tails Server es un servidor de archivos que está diseñado para recibir, almacenar y servir archivos de revocación de Hyperledger Indy. Estos archivos son generados por un emisor de credenciales cuando crea un registro de revocación, y deben publicarse en una ubicación accesible para todos los titulares que reciben una credencial a la que se hace referencia en ese registro de revocación.

Aunque ya se introdujo el concepto de revocación en la Sección 1.4.7, es necesario realizar una explicación más profunda para entender como funciona la revocación en Indy (según [7]).

4.4.1. Acumuladores criptográficos

Es necesario comprender los acumuladores criptográficos, aunque sea a un nivel muy alto. Se puede pensar en un acumulador como el producto de multiplicar muchos números. En la ecuación $a * b * c * d = e$, el acumulador sería e , que *acumula* un valor a medida que se multiplica cada nuevo factor. Si los números son, por ejemplo $a = 2$, $b = 3$, $c = 5$ y $d = 7$, entonces $e = 210$, y se puede decir que 3 (b) está en e porque es un factor. Si ahora se saca a b del acumulador, se divide 210 entre 3 y se obtiene 70 ($2*5*7$). Se ha eliminado entonces el factor b .

También es posible producir e multiplicando cualquier factor individual como a multiplicando por el producto de los demás factores ($b * c * d$). Esta es una característica muy útil, ya que permite decir a otra persona el valor de a y el producto de todas las demás entradas del acumulador, pero no las entradas en sí mismas.

4.4.2. Archivos Tails

En el ejemplo anterior solo hay 4 factores y se usan números pequeños. También se está usando aritmética estándar, donde puedes invertir la multiplicación dividiendo. En tal sistema, el contenido de un acumulador puede someterse a ingeniería inversa mediante factorización prima simple.

Para ser útiles para la revocación, los acumuladores de Indy no pueden ser reversibles; es decir, debe darse el caso de que la única forma de derivar el valor del acumulador sea conocer los factores. Esto se logra utilizando aritmética modular (donde la división no está definida), y usando números masivos para los factores y acumuladores.

Un archivo Tails está asociado con un acumulador y sus factores. Es un archivo binario que contiene una matriz de factores generadores aleatoriamente para un acumulador. En lugar de número pequeños como 2, 3 y 7, estos factores son número masivos.

Un archivo Tails no es secreto; se publica como texto sin formato para todo el mundo y cualquier persona puede descargarlo libremente. A cada credencial potencial o real emitida por un emisor en particular se le asigna un índice a un factor acumulador en un archivo Tails. Sin embargo, solo las credenciales que no han sido revocadas contribuyen al valor del acumulador (Figura 4.3).

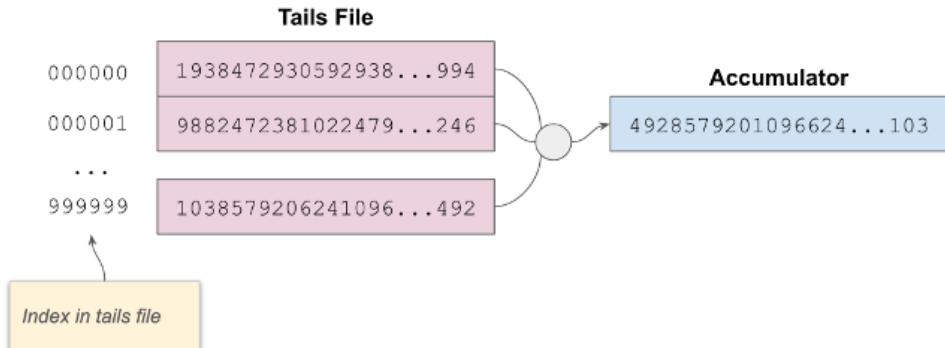


Figura 4.3: Archivo Tails [7].

4.4.3. Configuración

Antes de que se puedan emitir credenciales revocables, una serie de condiciones deben ser ciertas sobre el ecosistema:

- El emisor debe publicar en el libro mayor el esquema de credencial.
- El emisor debe publicar en el libro mayor una definición de credencial (esquema más claves).
- El emisor también debe publicar un registro de revocación. Estos metadatos hacen referencia a una definición de credencial y especifican cómo se manejará la revocación para ese tipo de credencial. El registro de revocación indica qué acumulador se puede usar para probar la revocación y proporciona el URI y el *hash* del archivo de colas asociado.
- El emisor debe publicar en el libro mayor un valor acumulado que describa el estado de revocación de todas las credenciales asociadas. Este acumulador debe actualizarse periódicamente o según sea necesario. Para ello, cuando se elimina una credencial el acumulador cambia ya que no se multiplican las credenciales revocadas en el acumulador (Figura 4.4).

4.4.4. Prueba de no revocación

Cuando un titular o probador le da una prueba a un verificador, se suele pensar en los atributos (fecha de nacimiento, dirección...). Esto es la **prueba principal**.

Pero hay otra dimensión de la prueba que también es necesaria cuando se trata de credenciales revocables: el probador debe demostrar que la(s)

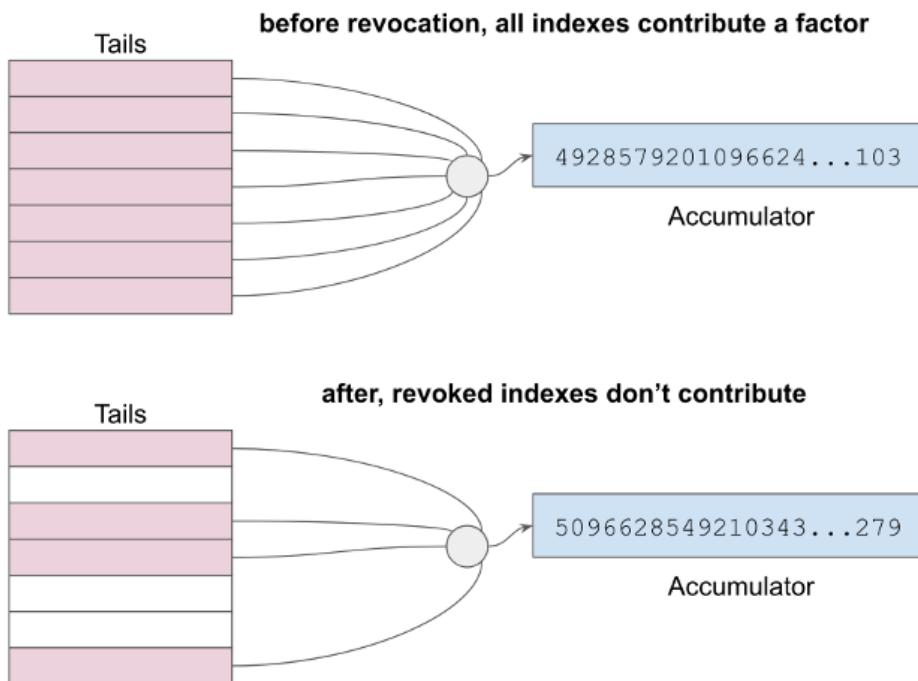


Figura 4.4: Archivo Tails tras revocar credenciales [7].

credencial(es) no ha(n) sido revocada(s). Esto se llama **prueba de no revocación**.

En Indy, la prueba de no revocación se logra haciendo que los probadores demuestren que pueden derivar el valor del acumulador para su credencial utilizando un factor para el acumulador que conocen, más el producto de todos los demás factores.

4.4.5. Preparación para la revocación en la emisión

Cuando se emite una credencial revocable, el emisor comunica dos datos extra vitales:

- El índice correspondiente a esta credencial en el archivo Tails. Esto le permite al titular buscar su factor privado.
- El producto de los otros factores que contribuyen al acumulador (todos los factores excepto el privado para esta credencial). Este valor se llama **testigo**.

4.4.6. Presentación de prueba de no revocación

Cuando el probador necesita demostrar que su credencial no está revocada, muestra que puede proporcionar matemáticas que derivan el valor acumulado en el libro mayor usando su factor privado multiplicado por el testigo. Esto se hace sin revelar el factor privado (importante para evitar correlación).

Pero existe una complicación: ¿y si el acumulador ha cambiado de valor desde que se emitió una credencial? En este caso, el factor privado multiplicado por el testigo no será igual al acumulador.

Esto se maneja requiriendo actualización para publicar un **delta testigo** como parte de la misma transacción. Este delta les dice a los probadores como ajustar su testigo (haciendo referencia a otros índices en el archivo Tails público). Si Alice es la probadora y su testigo era correcto para la actualización 21 del acumulador, pero el acumulador ha cambiado desde entonces a la versión 29, entonces necesita obtener los deltas 22-29 y aplicarlos a su testigo.

4.4.7. Puntos clave

Resumiendo todo lo visto sobre revocación y el servidor de archivos Tails, los puntos más importantes son:

- Los emisores revocan cambiando un número en el libro mayor. Pueden revocar tantas credenciales como quieran en una sola transacción, ya que solo están cambiando la respuesta a un problema matemático que incluye o no los factores que eligen.
- La revocación es reversible.
- Los probadores demuestran la prueba de no revocación de una manera que preserva la privacidad. No se pueden correlacionar con algo como una ID de credencial o un índice de colas.
- La verificación de la prueba de no revocación es extremadamente fácil y económica. Los verificadores no necesitan archivos de colas y el cálculo es trivial. Probar la credencial es algo más costoso para los probadores, pero no demasiado complejo.
- Los verificadores no necesitan ponerse en contacto con los emisores o consultar una lista de revocación para probar la revocación.

4.5. Arquitectura de los agentes

Para los agentes, se utilizará Hyperledger Aries. Este proporciona un kit de herramientas diseñado para soluciones centradas en la creación, transmisión y almacenamiento de credenciales digitales verificables. En esta librería es donde se centran los desarrolladores de aplicaciones. La parte de la *block-chain* la lleva Indy, pero el trabajo se centra en Aries.

Gracias a Indy, Aries y Ursa, es posible:

- Establecer un canal seguro y privado con otra entidad pero sin necesidad de hacer login ni ningún servidor central.
- Recibir y enviar mensajes con alta seguridad y privacidad.
- Demostrar cosas sobre sí mismo, mediante presentación de credenciales.
- El manejo del usuario de su propia identidad.

Todos los agentes de Aries están formados por 2 componentes lógicos: un *framework* y un controlador (Figura 4.5). El *framework* contiene las capacidades que permiten que un agente interactúe con el entorno: cadenas de bloques, almacenamiento, credenciales verificables, presentaciones y otros agentes. El *framework* es algo que, como desarrollador de aplicaciones Aries, no se tiene que tocar ni mantener, simplemente se integra en la solución. Este sabrá como iniciar conexiones, responder solicitudes, almacenar las credenciales de forma segura, etc.

Brevemente, los componentes que incluye el *framework* son los siguientes:

1. **Servicio de administración de claves (KMS):** es el almacenamiento seguro (por encriptación) donde se recolecta toda la información importante: DIDs, claves, conexiones, credenciales, etc. Se puede utilizar SQLite para pequeñas implementaciones e incluso PostgreSQL para empresas si el anterior se queda corto.
2. **Interfaz de mensajería:** permite al agente establecer y gestionar conexiones así como mandar mensajes. Los detalles de cómo mandan mensajes los agentes se discutirá en la Sección 4.6.
3. **Interfaz con la cadena de bloques:** permite realizar operaciones en el *ledger* (leer transacciones y escribir en él). Para esto, el agente cuenta con un DID Resolver (para conseguir el DIDDoc dependiendo del tipo de DID) y un mecanismo de escritura.

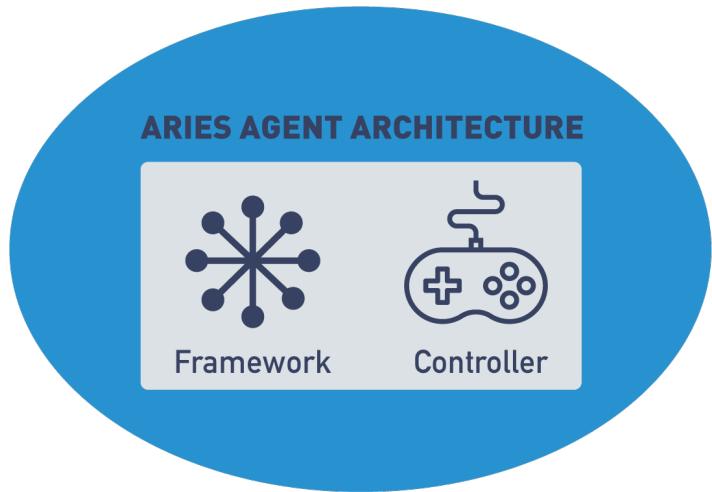


Figura 4.5: Arquitectura lógica de agente Aries [8].

Por otro lado, está el controlador, que es donde el desarrollador entra. El controlador maneja el comportamiento de una instancia de un marco Aries. El desarrollador crea y modifica el controlador dependiendo del uso que se le vaya a dar a la aplicación. Un ejemplo de controlador es la aplicación móvil que interactúa con el usuario.

Existen numerosos *frameworks*, pero el que se va a utilizar para este proyecto es ACA-Py (como ya se dijo en la Sección 3.3.1), debido a que está dedicado a aplicaciones no móviles y a que el lenguaje de programación utilizado es Python mayoritariamente. La arquitectura de este *framework* es la que se puede ver en la Figura 4.6.

El *framework* se encarga de las funcionalidades de Aries, que son interactuar con la *blockchain*, interactuar con otros agentes, enviar notificaciones al controlador y recibir instrucciones del controlador. El controlador por su parte sigue la lógica que el desarrollador haya decidido para su aplicación. Se pueden observar dos interfaces entre el *framework* y el controlador. Por una parte, el agente manda eventos a través de *webhooks* (notificaciones) y el controlador decide qué hacer. Para interactuar con el *framework*, el controlador manda solicitudes a la API.

A lo largo del proyecto, para referirse al *framework* se utilizará en algunos casos también el término agente, siendo los componentes de la arquitectura el agente y el controlador.

Para implementar los agentes (tanto controlador como framework, ínte-

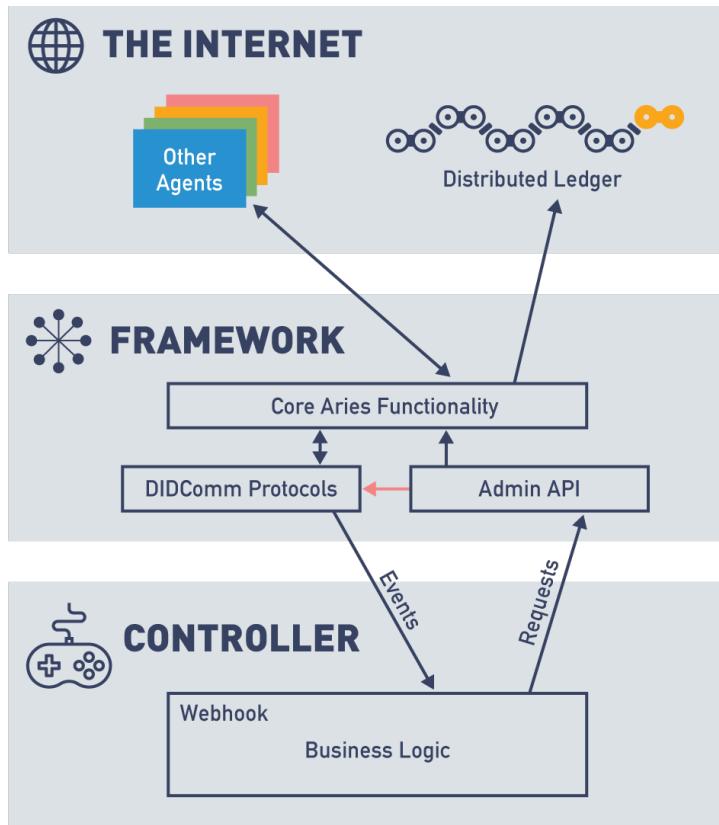


Figura 4.6: Arquitectura de agente Aries [8].

gramente) se utilizará el repositorio de Hyperledger de ACA-Py como referencia [29].

4.6. Protocolos de Aries

4.6.1. Introducción: interoperabilidad en Aries

Como ya se ha explicado, Aries permite mensajería P2P. Desde un simple mensaje en texto plano hasta la emisión de una credencial. Todo este proceso necesita, evidentemente, unas reglas. Para eso existen los protocolos.

Antes de profundizar más en los protocolos, es importante resaltar que uno de los principales objetivos de Aries es la interoperabilidad. Se han definido múltiples protocolos tanto para intercambiar mensajes como para emitir y presentar credenciales. Sin embargo, los protocolos están en constante cambio por surgir nuevos casos de uso y mejorar los ya existentes. Todos los protocolos se pueden consultar en el repositorio de ‘Aries RFC’

[30]. En este repositorio se revisan y envían cambios constantemente.

La solución escogida por la comunidad de Aries para solucionar el problema de la interoperabilidad fue la definición de perfiles de interoperabilidad de Aries (AIP). El concepto está definido formalmente en el Aries RFC 302 [31]. De forma resumida, los puntos clave son:

- Cada AIP tiene una misión y objetivos que las implementaciones de Aries podrán lograr si se cumplen con la versión AIP.
- Cada AIP define un conjunto de Aries RFCs (cada cual referenciado a un enlace de GitHub) que irán ligados a las implementaciones.
- Un conjunto de pruebas que demuestran la interoperabilidad entre las implementaciones de Aries.

En febrero de 2020 se definió el AIP 1.0 seleccionando 19 RFCs. Este consiguió permitir una interoperabilidad bastante confiable en muchas implementaciones de Aries.

A principios de 2021 se inicio un esfuerzo para definir AIP 2.0, y fue finalmente aprobado en mayo de ese mismo año. Su misión es la de agregar soporte para libros de contabilidad y formatos de credenciales verificables que no sean los de Hyperledger Indy.

En el proyecto se utilizarán las versiones más actualizadas, es decir, el AIP 2.0, debido a sus mejores funcionalidades. Aun así, existe la posibilidad de utilizar AIP 1.0.

4.6.2. Protocolos de mensajería

En Aries hay dos niveles de protocolos de mensajería (Figura 4.7):

- **DIDComm envelope protocol:** es el protocolo de bajo nivel. Se encarga del intercambio de mensajes, independientemente de su contenido. Se le llama *envelope* debido a que funciona como un servicio de correo postal.
- **Protocolo de contenido de Aries:** son la parte de alto nivel. Define las secuencias de mensajes para llegar a un objetivo compartido: desde enviar un simple mensaje a la emisión o revocación de una credencial.

La gran diferencia entre ellos reside en que el primero no se preocupa de lo que lleva el mensaje, sino que se centra en como se manda; mientras que los protocolos de alto nivel se centran en el que hacer con el contenido del mensaje, y no en cómo se manda.

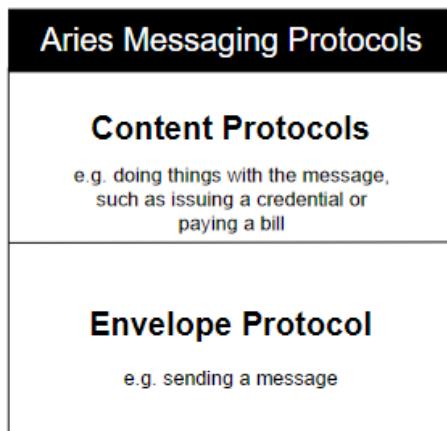


Figura 4.7: Protocolos de mensajería [8].

4.6.3. Protocolo de establecimiento de conexión

Uno de los protocolos más importantes que es necesario saber es el protocolo de establecimiento de conexión. El más reciente es el protocolo *DID-Exchange* (Aries RFC 0023 [10]), dentro del AIP 2.0. Antiguamente se utilizaba el protocolo de conexión (Aries RFC 0160 [32]). Sin embargo, como se ha usado el primero al utilizar AIP 2.0, es el que se va a explicar. Para nuevas conexiones, hay dos posibilidades:

- Usar el protocolo *Out-Of-Band* (Aries RFC 0434 [9]). El protocolo *Out-Of-Band* se utiliza para iniciar una conexión por primera vez con alguien, del que no se sabe el DID.
- A partir del DID público conectarse con la entidad.

Protocolo *Out-of-Band*

El protocolo *Out-of-Band* se utiliza para cuando se desea interactuar con un agente y no se tiene una conexión *DIDComm* para usar en la interacción. Dado que no hay una conexión *DIDComm* con la que cifrar los mensajes, estos van en texto plano y se envían fuera de banda, normalmente a través de un código QR, un mensaje de correo electrónico o cualquier otro canal disponible.

El protocolo tiene dos posibles roles. El remitente, que genera el mensaje fuera de banda y lo pone a disposición de la otra parte; y el receptor, que lo recibe y decide cómo responder. No hay ningún mensaje específico con el que responderá el receptor. De hecho, se responderá con un mensaje de otro protocolo que el remitente entienda (aquí es donde entra el protocolo

DID-Exchange).

Para el entendimiento del protocolo se utilizan las máquinas de estados. Esta puede ser algo difícil de comprender, debido a que el protocolo consiste en un solo mensaje que inicia un co-protocolo (en este caso *DID-Exchange*).

En las Figura 4.8 y 4.9 se pueden ver las tablas de estado del remitente y el receptor, respectivamente.

State Machine for 'sender' Role		
Events → ↓ States	send out-of-band message	receive DIDComm response message
initial	transition to await-response	error; send problem-report
await-response	impossible	transition to done optional: await-response
done	impossible	optional error; send problem-report

Figura 4.8: Tabla de estado del remitente [9].

State Machine for 'receiver' Role		
Events → ↓ States	receive out-of-band message	send DIDComm response message
initial	transition to prepare-response	impossible
prepare-response	impossible	transition to done
done	transition to prepare-response	impossible

Figura 4.9: Tabla de estado del receptor [9].

El protocolo *Out-of-Band* es un mensaje único que envía el remitente. En el Listado 4.1 se puede ver el esquema del mensaje *invitation*:

```

1
2 {
3     "@type": "https://didcomm.org/out-of-band/%VER/
4         invitation",
5     "@id": "<id used for context as pthid>",
6     "label": "Faber College",
7     "goal_code": "issue-vc",
8     "goal": "To issue a Faber College Graduate
9         credential",
10    "accept": [
11        "didcomm/aip2;env=rfc587",
12        "didcomm/aip2;env=rfc19"
13    ],
14    "handshake_protocols": [
15        "https://didcomm.org/didexchange/1.0",
16        "https://didcomm.org/connections/1.0"
17    ],
18    "requests$\\sim$attach": [
19        {
20            "@id": "request-0",
21            "mime-type": "application/json",
22            "data": {
23                "json": "<json of protocol message>"
24            }
25        },
26        "services": ["did:sov:LjgpST2rjsoxYegQDRm7EL"]
27    }
28 }
```

Listado de código 4.1: Ejemplo de mensaje *invitation* [9].

Los elementos del mensaje son:

- **@type**: el tipo de mensaje *DIDComm*.
- **@id**: identificación única del mensaje.
- **label [opcional]**: cadena auto-certificada para mostrar al usuario que envió el mensaje.
- **goal_code [opcional]**: un código auto-certificado que el receptor puede querer mostrar al usuario o utilizar para decidir automáticamente qué hacer con el mensaje.

- ***goal [opcional]***: una cadena auto-certificada que el receptor puede querer mostrar al usuario sobre el objetivo específico del mensaje fuera de banda.
- ***accept [opcional]***: una matriz de tipos de medios en el orden de preferencia del remitente que el receptor puede usar para responder al mensaje.
- ***handshake_protocols [opcional*]***: una matriz de protocolos en el orden de preferencia del remitente que el receptor puede usar para responder al mensaje con el fin de crear o reutilizar una conexión con el remitente.
- ***request~attach [opcional*]***: un decorador de archivos adjuntos que contiene una serie de mensajes de solicitud en orden de preferencia que le receptor puede usar para responder el mensaje.
- ***services***: una matriz de tipos de unión que puede utilizar el receptor al responder al mensaje. Cada elemento es un objeto *DIDComm service* o un DID.

*Uno o ambos elementos (*handshake_protocols* y *request~attach*) deben incluirse en el mensaje.

Una de las funcionalidades que ofrece este protocolo es la reutilización de conexiones. Si bien se espera que el receptor responda con un mensaje de inicio de un elemento *handshake_protocols* o *requests~attach* utilizando un servicio ofrecido, el receptor puede responder reutilizando una conexión existente.

Si el receptor quiere utilizar la conexión existente y hay un mensaje *request~attach*, el receptor debe responder al mensaje adjunto utilizando la conexión existente.

Si el receptor desea utilizar la conexión existente, y no hay ningún *request~attach*, el receptor debe intentar hacerlo con los mensajes *reuse* y *reuse-accepted*.

La Tabla 4.1 resume las diferentes formas de respuesta al mensaje *invitation* dependiendo de la presencia (o no) del *handshake_protocols*, el *request~attach* y si ya existe o no una conexión entre los agentes.

¿Está handshake_protocols?	¿Está request~attach?	¿Hay conexión existente?	Acción del receptor
No	No	No	Imposible.
Sí	No	No	Utiliza el primer protocolo admitido de handshake_protocols para realizar una nueva conexión utilizando la primera entrada del elemento services.
No	Sí	No	Envía una respuesta al primer mensaje de solicitud admitido utilizando la primera entrada del elemento services admitida.
No	No	Sí	Imposible.
Sí	Sí	No	Utiliza el primer protocolo admitido de handshake_protocols para realizar una nueva conexión mediante la primera entrada del elemento services admitida.
Sí	No	Sí	Envía un mensaje handshake-reuse.
No	Sí	Sí	Envíe un mensaje de respuesta al primer mensaje de solicitud admitido utilizando la conexión existente.
Sí	Sí	Sí	Envíe un mensaje de respuesta al primer mensaje de solicitud admitido utilizando la conexión existente.

Tabla 4.1: Tipos de respuesta *Out-Of-Band* [9].

Protocolo *DID-Exchange*

Este protocolo surge por la necesidad de los agentes Aries de establecer relaciones entre sí e intercambiar información de forma segura utilizando claves y puntos finales en *DIDDoc*s.

El protocolo *DID-Exchange* utiliza dos roles: solicitante y respondedor. El solicitante es la parte que inicia el protocolo después de recibir una mensaje *invitation* (usando *Out-of-Band* como se explica anteriormente) o usando una invitación implícita de un DID público. El respondedor debe tener un agente capaz de interactuar a través de *DIDComm*.

Como en el protocolo *Out-of-Band*, se utiliza una tabla de máquinas de estados para representar el comportamiento del protocolo (Figura 4.10).

Requester State Machine							
↓ States	Events →	receive explicit or find implicit invitation	send request	receive response	send complete	receive problem-report	send problem-report
start		transition to "invitation-received"	impossible	impossible	impossible	impossible	impossible
invitation-received		impossible		impossible	impossible	impossible	transition to "start" or "abandoned"
request-sent		impossible	impossible	transition to "response-received"	impossible	transition to "invitation-received" or "abandoned"	impossible
response-received		impossible	impossible	impossible	transition to "completed"	impossible	transition to "request-sent" or "abandoned"
abandoned		impossible	impossible	impossible	impossible	impossible	impossible
completed							

Responder State Machine							
↓ States	Events →	send explicit or publish implicit invitation	receive request	send response	receive complete	receive problem-report	send problem-report
start		transition to "invitation-sent"	impossible	impossible	impossible	impossible	impossible
invitation-sent		impossible	transition to "request-received"	impossible	impossible	transition to "start" or "abandoned"	impossible
request-received		impossible	impossible	transition to "response-sent"	impossible	impossible	transition to "invitation-sent" or "abandoned"
response-sent		impossible	impossible	impossible	transition to "completed"	transition to "request-received" or "abandoned"	impossible
abandoned		impossible	impossible	impossible	impossible	impossible	impossible
completed							

Figura 4.10: Tablas de estado del solicitante y respondedor [10].

El funcionamiento del protocolo es el siguiente (para más información sobre los mensajes y sus elementos consultar el Aries RFC 0023 [10]):

1. El respondedor proporciona información provisional al solicitante, ya sea mediante el mensaje explícito *invitation* del protocolo *Out-of-Band*, o de una invitación implícita en un DID que publica el respondedor (*DIDDoc* público).

2. El solicitante utiliza esta información provisional y envía un DID y *DIDDoc* al respondedor mediante el mensaje *request*.
3. El respondedor utiliza la información del *DIDDoc* recibido para enviar un DID y un *DIDDoc* al solicitante mediante el mensaje *response*.
4. El solicitante envía al respondedor el mensaje *complete* que confirma que recibió el mensaje *response* procedente del solicitante.
5. El intercambio entre el solicitante y el respondedor se ha completado. Esta relación no tiene confianza asociada. El siguiente paso debe ser aumentar la confianza a un nivel suficiente para el propósito de la relación, por ejemplo, a través de un intercambio de pruebas.

4.6.4. Protocolo de emisión de credencial

Es necesario un protocolo para formalizar los mensajes utilizados para emitir una credencial. Para esto surge el protocolo de emisión de credencial (Aries RFC 0036 [11]), el protocolo estándar para la emisión de credenciales. El protocolo utiliza dos roles: emisor y titular.

Para explicar el funcionamiento del protocolo se utiliza un diagrama de coreografía (Figura 4.11).

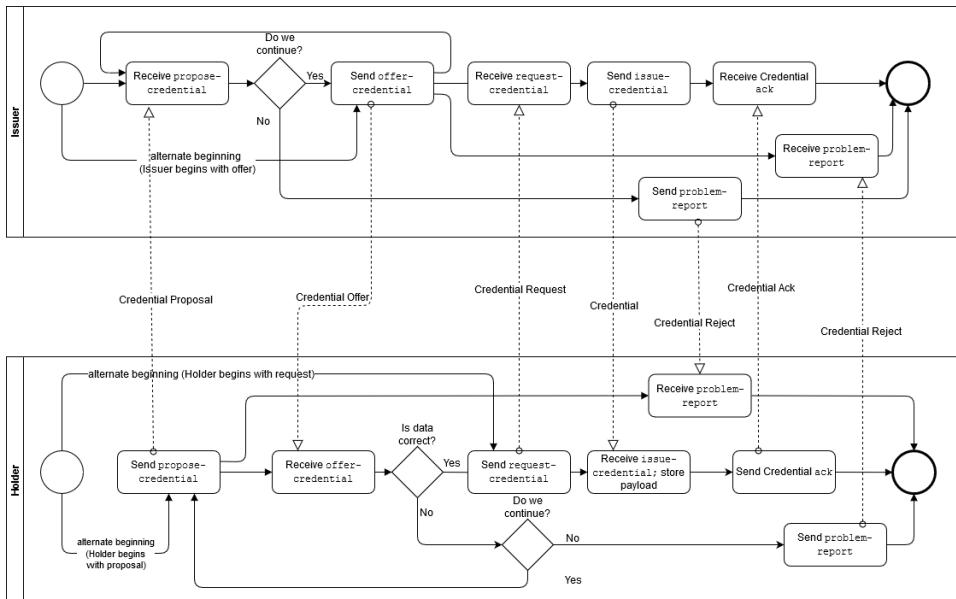


Figura 4.11: Diagrama de coreografía del protocolo de emisión de credencial [11].

Se puede observar como el protocolo tiene 3 inicios alternativos:

- El emisor comienza con una oferta.
- El titular comienza con una propuesta.
- El titular comienza con una solicitud.

Los mensajes de oferta y propuesta son parte de una fase de negociación opcional y pueden desencadenar contadores de ida y vuelta. Una solicitud no está sujeta a negociación; sólo puede ser aceptada o rechazada.

Los mensajes de los que consta el protocolo son los siguientes:

- ***propose-credential***: titular potencial a emisor (opcional). El titular potencial le dice al emisor lo que espera recibir.
- ***offer-credential***: emisor a titular potencial (opcional en algunas implementaciones, obligatorio en Hyperledger Indy). El emisor indica al titular potencial lo que pretende emitir, y si es el caso, el precio de la emisión.
- ***request-credential***: titular potencial a emisor. Si no se utiliza ninguno de los mensajes anteriores, es el mensaje que inicia el protocolo.
- ***issue-credential***: emisor a titular. Contiene la credencial final.
- ***ack y problem-report***: adoptados para la confirmación y el manejo de errores.

Mensaje *propose-credential*

Es un mensaje opcional enviado por el titular potencial al emisor para iniciar el protocolo o en respuesta a un mensaje *offer-credential* cuando el titular desea que se realicen algunos ajustes a los datos de credenciales ofrecidos por el emisor.

El esquema del mensaje es el mostrado en el Listado 4.2:

```

1 {
2   "@type": "https://didcomm.org/issue-credential/1.1
3   /propose-credential",
4   "@id": "<uuid-of-propose-message>",
5   "comment": "some comment",
6   "credential_proposal": <json-ld object>,
7   "schema_issuer_did": "DID of the proposed schema
8   issuer",

```

```
7   "schema_id": "Schema ID string",
8   "schema_name": "Schema name string",
9   "schema_version": "Schema version string",
10  "cred_def_id": "Credential Definition ID string"
11  "issuer_did": "DID of the proposed issuer"
12 }
```

Listado de código 4.2: Ejemplo de mensaje *propose-credential* [11].

La descripción de los campos es:

- **comment:** un campo opcional que proporciona información sobre la propuesta de credencial.
- **credential_proposal:** un objeto JSON-LD opcional que presenta los datos de credenciales que el titular potencial quiere recibir. Coincide con el esquema de vista previa de credencial (Sección 4.6.4).
- **schema_issuer_did:** filtro opcional para solicitar credenciales en función de un DID de emisor de esquema en particular.
- **schema_id:** filtro opcional para solicitar credenciales en función de un esquema en particular.
- **schema_version:** filtro opcional para solicitar credenciales en función de una versión del esquema.
- **cred_def_id:** filtro opcional para solicitar una credencial en función de una definición de credencial en particular.
- **issuer_did:** filtro opcional para solicitar una credencial emitida por el titular de un DID particular.

Mensaje *offer-credential*

Es un mensaje enviado por el emisor al titular potencial, describiendo la credencial que pretende ofrecer y posiblemente el precio a pagar. En Hyperledger Indy, este mensaje es obligatorio porque obliga al emisor a realizar un compromiso criptográfico con el conjunto de campos de la credencial final y, por lo tanto, evita que los emisores inserten datos falsos. En implementaciones de credenciales donde este mensaje es opcional, un emisor puede usar el mensaje para negociar la emisión posterior a la recepción de un mensaje *request-credential*.

El esquema del mensaje es el mostrado en el Listado 4.3:

```

1  {
2      "@type": "https://didcomm.org/issue-credential/1.0
3          /offer-credential",
4      "@id": "<uuid-of-offer-message>",
5      "comment": "some comment",
6      "credential_preview": <json-ld object>,
7      "offers$\\sim$attach": [
8          {
9              "@id": "libindy-cred-offer-0",
10             "mime-type": "application/json",
11             "data": {
12                 "base64": "<bytes for base64>"
13             }
14         }
15     ]
}

```

Listado de código 4.3: Ejemplo de mensaje *offer-credential* [11].

La descripción de los campos es:

- ***comment***: un campo opcional que proporciona información legible sobre la oferta de credencial.
- ***credential_preview***: un objeto JSON-LD que representa los datos de credenciales que el emisor está dispuesto a emitir. Coincide con el esquema de vista previa de credencial (Sección 4.6.4).
- ***offers~attach***: una variedad de archivos adjuntos que definen aún más la credencial que se ofrece.

El emisor puede añadir un *payment-request* a este mensaje para transmitir la necesidad de pago antes de la emisión de la credencial.

Mensaje *request-credential*

Este es un mensaje enviado por el potencial titular al emisor, para solicitar la emisión de una credencial. Cuando las circunstancias no requieran un mensaje de *offer-credential* anterior, este mensaje inicia el protocolo. En Hyperledger Indy, este mensaje solo se puede enviar en respuesta a un mensaje de *offer-credential*.

El esquema es el mostrado en el Listado 4.4:

```

1 {
2   "@type": "https://didcomm.org/issue-credential/1.0
3   /request-credential",
4   "@id": "<uuid-of-request-message>",
5   "comment": "some comment",
6   "requests$sim$attach": [
7     {
8       "@id": "attachment id",
9       "mime-type": "application/json",
10      "data": {
11        "base64": "<bytes for base64>"
12      }
13    },
14  ]
}

```

Listado de código 4.4: Ejemplo de mensaje *request-credential* [11].

La descripción de los campos es:

- ***comment***: un campo opcional que proporciona información legible sobre la solicitud de credencial.
- ***requests~attach***: una serie de archivos adjuntos que define los formatos solicitados para la credencial.

Este mensaje puede tener un ~payment-receipt para demostrarle al emisor que el titular potencial ha cumplido con el requisito de pago.

Mensaje *issue-credential*

Este mensaje contiene como carga útil adjunta las credenciales que se emiten y se envía en respuesta a un mensaje *request-credential* válido.

El esquema es el mostrado en el Listado 4.5:

```

1 {
2   "@type": "https://didcomm.org/issue-credential/1.0
3   /issue-credential",
4   "@id": "<uuid-of-issue-message>",
5   "comment": "some comment",
6   "credentials$sim$attach": [
7     {
8       "@id": "libindy-cred-0",
9       "mime-type": "application/json",

```

```

9         "data": {
10            "base64": "<bytes for base64>"
11        }
12    ]
13 }
14 }
```

Listado de código 4.5: Ejemplo de mensaje *issue-credential* [11].

La descripción de los campos es:

- ***comment***: un campo opcional que proporciona información legible sobre la credencial emitida.
- ***credentials~attach***: una serie de archivos adjuntos que contienen las credenciales emitidas.

Si el emisor desea que el titular confirme que ha aceptado la credencial, el mensaje debe incluir un *~please-ack*, de tal manera que el titular responderá con un mensaje *ack*.

Vista previa de credencial

Este no es un mensaje sino un objeto interno para otros mensajes en este protocolo. Se utiliza para construir una vista previa de los datos de la credencial que se va a emitir. Su esquema sigue la estructura vista en el Listado 4.6:

```

1 {
2   "@type": "https://didcomm.org/issue-credential/1.0
3   /credential-preview",
4   "attributes": [
5     {
6       "name": "<attribute name>",
7       "mime-type": "<type>",
8       "value": "<value>"
9     },
10    // more attributes
11  ]
```

Listado de código 4.6: Ejemplo de vista previa de credencial [11].

El elemento principal es *attributes*. Es una matriz de especificaciones de atributos. La semántica es la siguiente:

- ***name***: se asigna al nombre del atributo como una cadena.

- ***mime-type y value***: El mensaje opcional *mime-type* informa al emisor sobre cómo representar un atributo. El campo *value* contiene el valor del atributo. Si *mime-type* es nulo, *value* es una cadena. Si no es nulo, *value* es una cadena codificada en base64url.

4.6.5. Protocolo de presentación de credencial

Es necesario un protocolo para formalizar los mensajes utilizados para presentar una prueba de credencial. Para esto surge el protocolo de presentación de credencial (Aries RFC 0037 [12]), el protocolo estándar para la presentación de prueba de credenciales.

El protocolo utiliza dos roles: probador y verificador.

Para explicar el funcionamiento del protocolo se utiliza un diagrama de coreografía (Figura 4.12). Se puede observar como se hace uso de *acks* y *problem-report* para confirmación y manejo de errores.

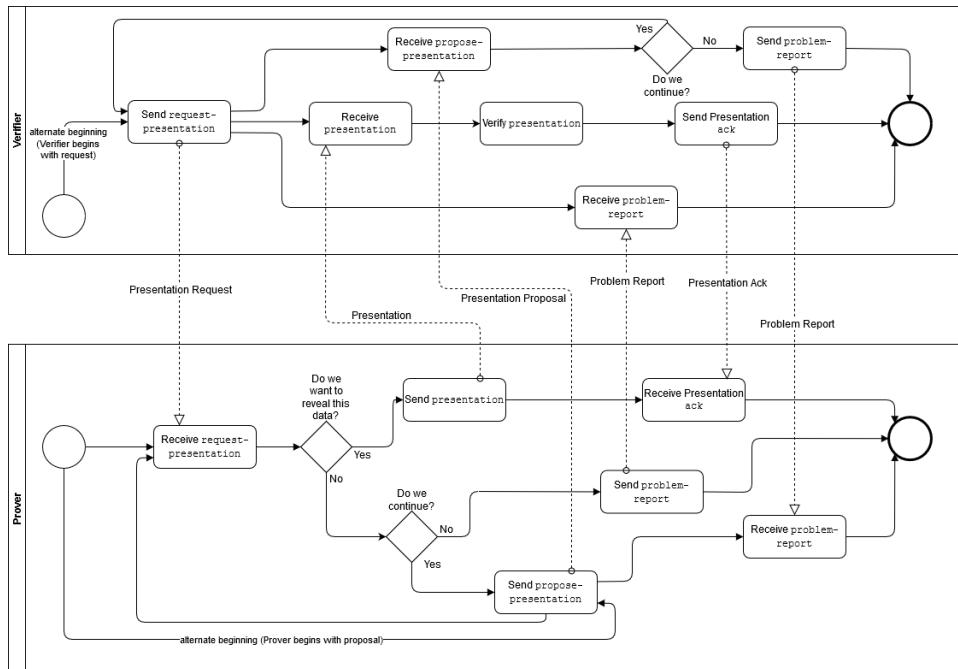


Figura 4.12: Diagrama de coreografía del protocolo de presentación de credencial [12].

El protocolo consta de tres etapas:

- Proposición de presentación (opcional): del probador al verificador.

- Solicitud de presentación: del verificador al probador.
- Presentación de prueba: del probador al verificador.

Los mensajes de los que consta el protocolo son los siguientes:

Mensaje *propose-presentation*

Un mensaje opcional enviado por el probador al verificador para iniciar un proceso de presentación de prueba, o en respuesta a un mensaje *request-presentantion* cuando el probador quiere proponer utilizar un formato de presentación diferente.

El esquema del mensaje es el mostrado en el Listado 4.7:

```

1  {
2      "@type": "https://didcomm.org/present-proof/1.0/
3          propose-presentation",
4      "@id": "<uuid-propose-presentation>",
5      "comment": "some comment",
6      "presentation_proposal": <json-ld object>
7 }
```

Listado de código 4.7: Ejemplo de mensaje *propose-presentation* [12].

La descripción de los campos es:

- ***comment***: un campo opcional que proporciona información legible sobre la presentación propuesta.
- ***presentation_proposal***: un objeto JSON-LD que representa el ejemplo de presentación que el probador quiere proporcionar. Debe seguir el esquema de Vista Previa de Presentación (Sección 4.6.5).

Mensaje *request-presentation*

Mensaje que manda el verificador al probador, y que describe los atributos que deben revelarse y los predicados que deben cumplirse.

El esquema del mensaje es el mostrado en el Listado 4.8:

```

1  {
2      "@type": "https://didcomm.org/present-proof/1.0/
3          request-presentation",
4      "@id": "<uuid-request>",
5      "comment": "some comment",
6      "request_presentations$\\sim$attach": [
```

```

6      {
7          "@id": "libindy-request-presentation-0",
8          "mime-type": "application/json",
9          "data": {
10              "base64": "<bytes for base64>"
11          }
12      }
13  ]
14 }
```

Listado de código 4.8: Ejemplo de mensaje *request-presentation* [12].

La descripción de los campos es:

- ***comment***: un campo opcional que proporciona información legible sobre la solicitud de presentación.
- ***request-presentations~attach***: una serie de archivos adjuntos que definen los formatos aceptables para la presentación.

Mensaje *presentation*

Este mensaje es la respuesta al mensaje *request-presentation* y contiene la presentación firmada.

El esquema del mensaje es el mostrado en el Listado 4.9:

```

1 {
2     "@type": "https://didcomm.org/present-proof/1.0/
3         presentation",
4     "@id": "<uuid-presentation>",
5     "comment": "some comment",
6     "presentations$\\sim$attach": [
7         {
8             "@id": "libindy-presentation-0",
9             "mime-type": "application/json",
10            "data": {
11                "base64": "<bytes for base64>"
12            }
13        }
14    ]
}
```

Listado de código 4.9: Ejemplo de mensaje *presentation* [12].

La descripción de los campos es:

- ***comment***: un campo opcional que proporciona información legible sobre la presentación.
- ***presentations~attach***: una serie de archivos adjuntos que contienen la presentación en los formatos solicitados.

Vista previa de presentación

Este no es un mensaje sino un objeto interno para otros mensajes en este protocolo. Se utiliza para construir una vista previa de los datos para la presentación. Su esquema sigue la estructura del Listado 4.10:

```

1
2 {
3     "@type": "https://didcomm.org/present-proof/1.0/
4         presentation-preview",
5     "attributes": [
6         {
7             "name": "<attribute_name>",
8             "cred_def_id": "<cred_def_id>",
9             "mime-type": "<type>",
10            "value": "<value>",
11            "referent": "<referent>"
12        },
13        // more attributes
14    ],
15    "predicates": [
16        {
17            "name": "<attribute_name>",
18            "cred_def_id": "<cred_def_id>",
19            "predicate": "<predicate>",
20            "threshold": <threshold>
21        },
22        // more predicates
23    ]
}

```

Listado de código 4.10: Ejemplo de vista previa de presentación [12].

La clave obligatoria ‘*attributes*’ se asigna a una lista (posiblemente vacía para proponer una presentación sin atributos) de especificaciones, una por atributo. La clave ‘*name*’ se asigna al nombre del atributo.

La clave obligatoria ‘*predicates*’ se asigna a una lista (posiblemente vacía para proponer una presentación sin predicados) de especificaciones de predicados, una por predicado. La clave ‘*name*’ se asigna al nombre del atributo.

La clave ‘*predicate*’ se asigna al operador del predicado: ‘<’, ‘ \leq ’, ‘ \geq ’, ‘>’.

4.7. Entorno de evaluación de funcionalidades

El primer entorno que se va a desarrollar tiene como objetivo evaluar las funcionalidades de los agentes. La arquitectura del entorno se puede ver en la Figura 4.13.

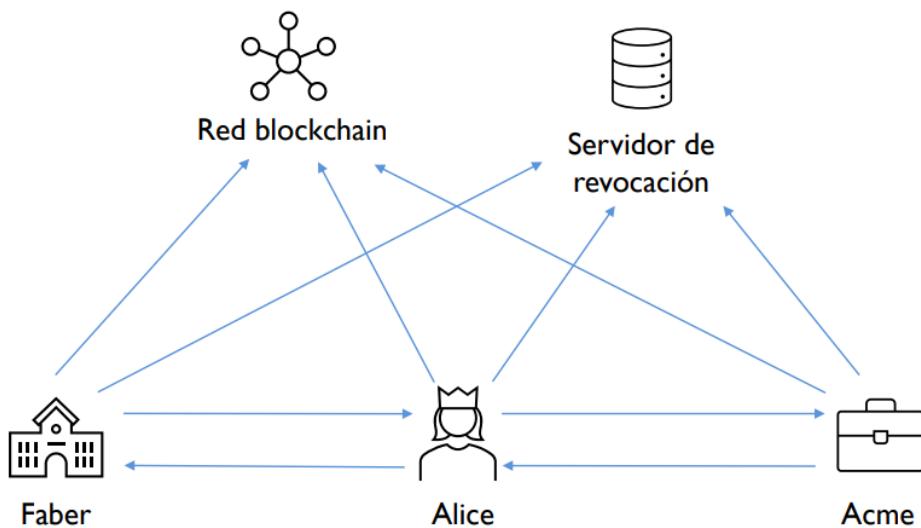


Figura 4.13: Arquitectura del entorno de evaluación de funcionalidades.

Las funcionalidades y conexiones de cada entidad son las siguientes:

- **Faber:** Faber puede escribir en el *ledger*, ya que es un emisor autorizado. Se conectará con Alice para emitir la credencial y con el *ledger* para publicar el esquema y definición de credencial, así como el registro de revocación. Para subir los archivos Tails se conectará al servidor de revocación.
- **Alice:** Alice no está autorizada para escribir en el *ledger* (ya que es permisionado) pero si a leerlo (ya que es público). Se conectará a Faber para que este le emita la credencial, y a Acme para presentarle la credencial emitida por Faber y que este le emita una credencial nueva. Se conectará también al servidor de revocación para poder enviar las pruebas de no revocación.
- **Acme:** Acme está, al igual que Faber, autorizado para escribir en el *ledger*. Se conectará a Alice para que esta le envíe una prueba de

credencial y, si es correcta, emitirle una nueva credencial. También estará conectado al servidor de revocación, en caso de que la credencial que emita sea revocable.

- **Red blockchain:** en la red estarán todos los datos necesarios para hacer funcionar el entorno SSI. Esto es, los esquemas y definiciones de credencial, el registro de revocación (que incluye el acumulador para la prueba de no revocación como se vio en la Sección 4.4). También se encuentran los DIDs públicos de las entidades que así lo requieran (es decir, Faber y Acme).
- **Servidor de revocación:** en el servidor de revocación residen los archivos Tails, necesarios para que la revocación sea posible para los emisores y probadores. Acme y Faber subirán a este los archivos de revocación para que Alice (o cualquiera, ya que es público) pueda consultarlos.

Una vez visto el papel de cada entidad/agente en el diseño, el proceso que se va a seguir para evaluar las funcionalidades es el siguiente:

1. Alice se conecta a Faber. Faber le emite la credencial del título universitario a Alice, y la guarda en su *wallet*.
2. Alice se conecta a Acme porque quiere conseguir un trabajo en esa empresa. Para ello Acme le pide una prueba del título universitario. Una vez verificada y comprobada la prueba que Alice le manda a Acme, Acme contrata a Alice y le emite una credencial que prueba que trabaja en esa empresa. Alice podrá utilizar esa credencial para otros servicios, como para por ejemplo pedir un préstamo en un banco, donde el requerimiento sea tener un trabajo.
3. Faber revocará la credencial del título universitario de Alice, y posteriormente le pedirá una prueba de la credencial, con el objetivo de visualizar como en este caso la prueba es falsa (evidentemente).

4.8. Entorno de evaluación de rendimiento

Por último, se pretende diseñar un entorno de evaluación de rendimiento. Para ello, se van a desplegar distintos escenarios y se van a medir los siguientes parámetros:

- **Tiempo de despliegue de agentes:** tiempo que tardan los agentes en desplegarse.
- **Tiempo de conexión.:** tiempo que tardan Alice y Faber en conectarse.

- **Tiempo de publicación:** tiempo que tarda Faber en publicar los datos necesarios en el *ledger*.
- **Tiempo medio por emisión de una credencial:** tiempo que tarda Faber en emitir una credencial a Faber y Alice la recibe.
- **Tiempo medio por presentación de prueba de credencial (si es el caso):** tiempo que tarda Alice en presentar una prueba de credencial y Faber consigue verificarla.
- **Tiempo medio por revocación de credencial (si es el caso):** tiempo que tarda Faber en revocar una credencial y publicar el nuevo acumulador en el *ledger*.
- **Tiempo total de ejecución:** tiempo total del escenario (para una sola prueba).

El objetivo por tanto es diferente al entorno de evaluación de funcionalidades. Este busca analizar y probar las funcionalidades de los agentes, mientras que el entorno de evaluación de rendimiento busca estudiar cómo reaccionan estos a distintos números de credenciales en términos de rendimiento.

Antes de explicar los escenarios que se van a poner a prueba, es importante diferenciar entre dos tipos de credenciales:

- **Credencial Simple (CS):** son aquellas credenciales que no tienen posibilidad de revocación.
- **Credencial Revocable (CR):** son aquellas credenciales que tienen la posibilidad de ser revocadas.

4.8.1. Emisión y presentación de credenciales simples

En este primer escenario se realizará la emisión y presentación de credenciales simples. El diseño del escenario se puede observar en la Figura 4.14. Faber emitirá N credenciales a Alice y, posteriormente, le pedirá a esta que presenta pruebas de las credenciales emitidas. Este escenario se repetirá M veces para poder tener suficientes datos y que los resultados sean precisos.

4.8.2. Emisión y presentación de credenciales revocables sin llegar a revocarlas

También se realizarán pruebas con credenciales revocables, pero sin llegar a revocarlas. El diseño se puede observar en la Figura 4.15. Al igual que en el escenario anterior, Faber emitirá N credenciales a Alice y, posteriormente,

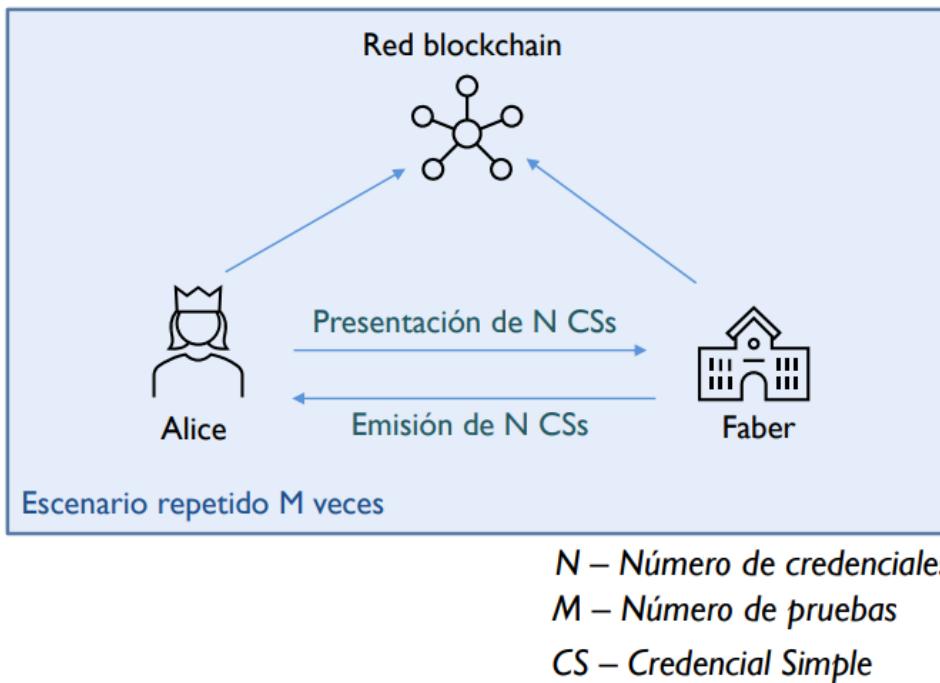


Figura 4.14: Diseño escenario CS.

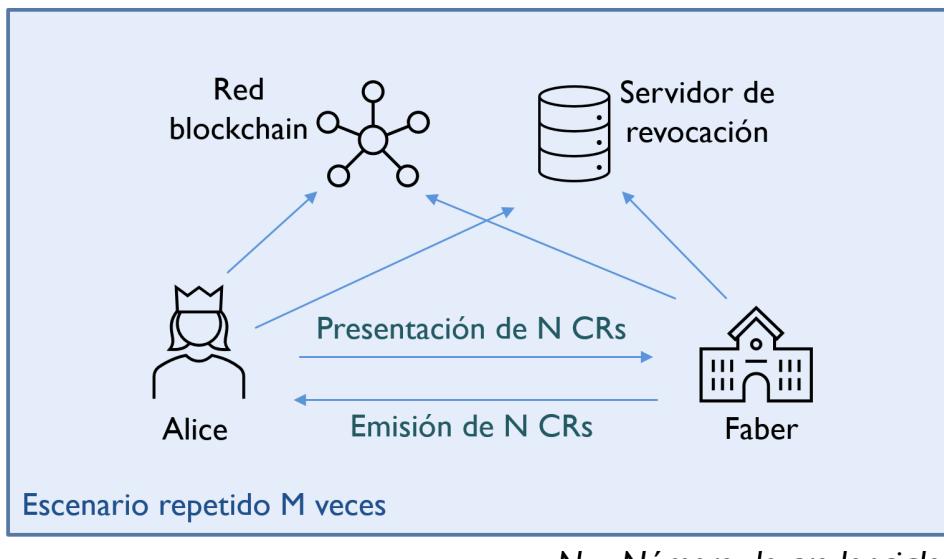
le pedirá a esta que presenta pruebas de las credenciales emitidas. Este escenario se repetirá M veces para poder tener suficientes datos y que los resultados sean precisos.

4.8.3. Credenciales revocables revocadas

En este último escenario se emitirán credenciales, y posteriormente se revocarán. Surgen dos subescenarios: revocar todas las credenciales a la vez o una a una (recordar que revocar una credencial es publicar un nuevo acumulador en el *ledger*). El diseño de este escenario se muestra en la Figura 4.16.

Credenciales revocables revocadas y publicadas una a una

En este escenario se verá la revocación de credenciales y su publicación una a una en el *ledger*. Faber emitirá a Alice N credenciales (N será un número menor que en los dos escenarios anteriores debido a la carga computacional de revocar tantas credenciales una a una) y posteriormente las revocará. Se ejecutará el escenario un número de M veces.



N – Número de credenciales

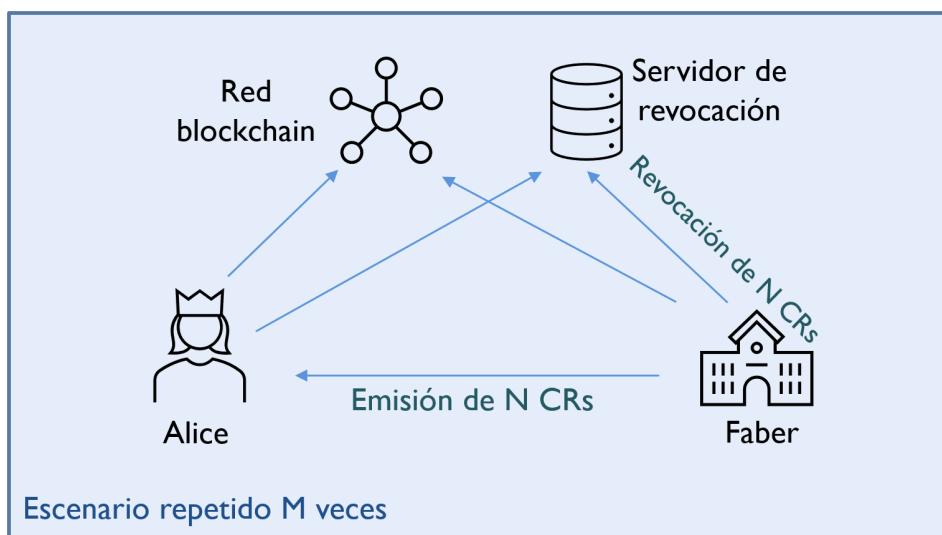
M – Número de pruebas

CR – Credencial Revocable

Figura 4.15: Diseño escenario CR sin revocar.

Credenciales revocables revocadas y publicadas a la vez

En este escenario se verá la revocación de credenciales y su publicación en el *ledger* al final de todas las revocaciones. Faber emitirá a Alice N credenciales y posteriormente las revocará. Se ejecutará el escenario un número de M veces. En este caso, la publicación se realizará al final de todas las revocaciones.



N – Número de credenciales

M – Número de pruebas

CR – Credencial Revocable

Figura 4.16: Diseño escenario CR revocando.

Capítulo 5

Implementación

5.1. Introducción

En este capítulo se va a explicar cómo se ha llevado a la práctica el diseño explicado en la Sección 4. Para ello, se identificarán las tecnologías y herramientas utilizadas, así como explicar los fragmentos de código necesarias para implementar la solución. Para poder seguir la implementación se hará una explicación utilizando el código del alumno encargado del proyecto, Rafael Adán López, por lo que se recomienda descargarlo para poder seguirlo. El código está en el repositorio <https://github.com/Rafaadan/aca-py-performance>.

5.2. Tecnologías y herramientas usadas

Las tecnologías y herramientas más importantes utilizadas a lo largo del proyecto son:

- Virtual Box.
- Docker y Docker-compose.
- Visual Code Studio.
- Git y Github.

Se espera que el lector tenga el equipamiento y las herramientas necesarias. Para poder instalar todo el entorno necesario para la implementación, consultar el Anexo A. En este anexo, además de explicar las instalaciones necesarias, explica las tecnologías y herramientas usadas.

5.3. Implementación de *von-network*

Para la implementación de la red local *von-network* se hace uso del proyecto ubicado en el repositorio del Gobierno de British Columbia [27].

Para desplegar la red *von-network* se utiliza el *script* bash *manage*. Este *script* tiene como función principal levantar la red de 4 nodos que se explicó en la Sección 4.3. Simplifica el proceso de ejecución de la red VON, proporcionando los puntos de entrada comunes que necesita usar. También proporciona una serie de variables de entorno que puede usar para personalizar la ejecución del *script*. Para ello, tiene diversas opciones para su despliegue, de las cuales las más importantes son:

- ***build***: compila las imágenes docker para el proyecto.
- ***start***: ejecuta todos los contenedores.
- ***start-web***: ejecuta un servidor web para monitorizar una red existente.
- ***logs***: para ver los logs de los contenedores en ejecución.
- ***down/rm***: echa abajo los contenedores y borra los datos asociados a ellos.
- ***stop***: para los contenedores pero no los borra como la opción *down*.

Para desplegar la red, es necesario ejecutar por tanto el *script* *manage*. En primer lugar, se ejecuta el siguiente comando:

```
./manage build
```

Esto compilará las imágenes según se especifica en el archivo *Dockerfile* y en el *docker-compose.yml* (ya que son 4 nodos).

Tras esto, se inicia la red con el siguiente comando:

```
./manage start
```

Una vez hecho esto, ya estará desplegada la red *von-network* y lista para usarse (Figura 5.1).

Von-network dispone de una interfaz web (Figura 5.2) que permite ver con más detalle el funcionamiento de la cadena de bloques. Para ello, hay que acceder a <http://localhost:9000>.

```
rafa@rafa-VirtualBox:~/aca-py-performance/von-network$ ./manage start
Using: docker-compose --log-level ERROR

Creating von_node3_1    ... done
Creating von_webserver_1 ... done
Creating von_node4_1    ... done
Creating von_node2_1    ... done
Creating von_node1_1    ... done
Want to see the scrolling container logs? Run "./manage logs"
rafa@rafa-VirtualBox:~/aca-py-performance/von-network$
```

Figura 5.1: Inicio *von-network*.

The screenshot shows the von-network web interface running on localhost. The main page has a dark header with the text "Contributed by the Province of British Columbia". Below the header, there are two main sections:

- Validator Node Status:** This section lists four nodes (Node1, Node2, Node3, Node4) with their status details. Each node entry includes its DID, Uptime, Txns, and Indy-node version.
- Ledger State:** This section allows users to view the state of the ledgers, with links for Domain, Pool, and Config.

On the right side of the page, there are three additional sections:

- Connect to the Network:** A form for downloading the genesis transaction file to connect to the network, with options for "Genesis Transaction" and "JSON".
- Authenticate a New DID:** A form for easily writing a new DID to the ledger for new identity owners, with fields for "Wallet seed" (32 characters or base64), "DID" (optional), "Alias" (optional), and "Role" (set to "Endorser").
- Register DID:** A button labeled "Register DID".

Figura 5.2: Interfaz web *von-network*.

Navegando por la interfaz se puede conseguir diversa información sobre el *ledger*. Por un lado, se puede ver el estado de los 4 nodos, con información como desde cuando está activo, su DID o la versión de Indy que tiene.

Por otro lado, se puede descargar el archivo de génesis. Este archivo es generado cada vez que se ejecuta una instancia de *von-network*. Este fichero contiene la información para conectarse al *ledger*, y es un parámetro que deberán conocer los agentes Aries para conectarse a este. Está escrito en formato JSON y tiene atributos como la dirección IP y puerto de los nodos y sus DIDs.

Otra funcionalidad es la de crear un nuevo DID. Para ello, la forma más fácil utilizar la opción de *Register from Seed*. Para eso hay que añadir una semilla y el Alias para ese DID. También se pide el rol que tendrá la entidad.

Se puede escoger *Endorser*, lo que permitirá escribir en el *ledger*. Utilizando como semilla ‘Rafa’ y como alias ‘Rafa Adan’, se registra el DID en el *ledger* (Figura 5.3).

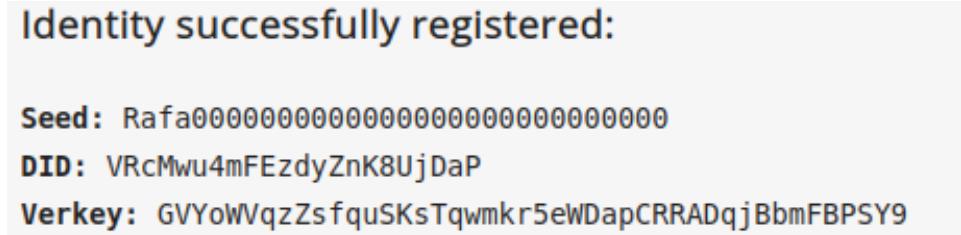


Figura 5.3: Creación nuevo DID.

Como última funcionalidad, y que sirve para demostrar la anterior, se pueden explorar las transacciones en el *ledger*. Hay tres categorías como se puede ver en la Figura 5.2:

- **Domain:** permite ver transacciones relacionadas con los DIDs, los esquemas, etc. Es decir, lo que se escribe en el *ledger*.
- **Pool:** permite ver transacciones relacionadas con los nodos del *ledger*.
- **Config:** permite ver transacciones relacionadas con los cambios en la configuración de la red.

La que más interesa es la parte ‘Domain’, donde están las transacciones relacionadas con los agentes que escriben en la cadena de bloques. Si accedemos a este, se puede ver la transacción asociada a la creación del nuevo DID (Figura 5.4). Cada transacción tendrá un ID asociado, así como una marca de tiempo y de quién lo publicó al *ledger*.

5.4. Implementación de *indy-tails-server*

Para la implementación del servidor de archivos de revocación se hace uso del proyecto *indy-tails-server*, también implementado por el Gobierno de British Columbia [28].

Para desplegar el servidor se utiliza el *script* bash *manage*. Este *script* simplifica el proceso del despliegue. Las opciones que brinda el *script* son:

- **build:** compila las imágenes docker para el proyecto.
- **start/up:** ejecuta todos los contenedores.

#6 Message Wrapper

Transaction ID: 4b3d1e427d6945354bcc1783a325be9409e5e2182c9de2fafb40fd7d95c12e69
 Transaction time: 30/5/2022, 12:35:28 (1653906928)
 Signed by: V4SGRU86Z58d6TV7PBUE6f

Metadata

From nyx: V4SGRU86Z58d6TV7PBUE6f
 Request ID: 1653906928749936000
 Digest: 64711685b7c5b967daaa1a1954e1037147506ac12ca52853f02255614ab6dc01

Transaction

Type: NYM
 Alias: Rafa Adan
 Nym: VRcMwu4mFEzdyZnK8UjDaP
 Role: ENDORSER
 Verkey: GYYoWVqzzSfqUSKS Tqwmkr5eWDapCRRADqjBbmFBPSY9

Raw Data ▾

Figura 5.4: Transacción asociada a la creación de nuevo DID.

- ***logs***: para ver los *logs* de los contenedores en ejecución.
- ***rm***: echa abajo los contenedores y borra los datos asociados a ellos.
- ***stop/down***: para los contenedores pero no los borra como la opción *rm*.

Para desplegar el servidor es necesario por tanto ejecutar el *script manage*. Este *script* se encuentra en la carpeta docker del repositorio. En primer lugar, es necesario compilar las imágenes:

```
./manage build
```

Tras esto, ya se puede desplegar el servidor:

```
./manage start
```

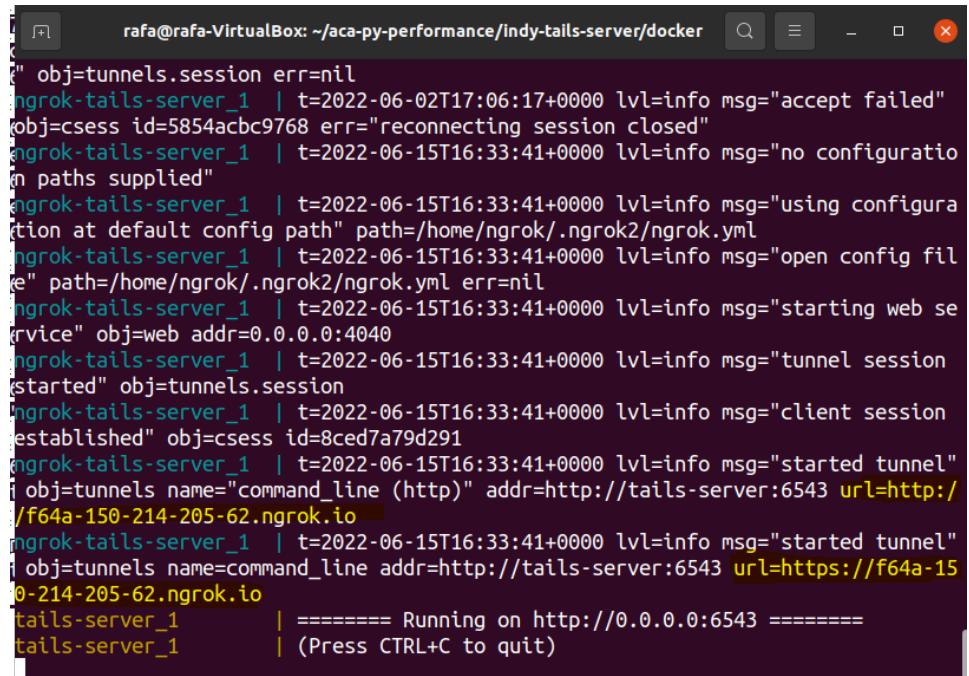
Hecho esto, el servidor ya está listo para usarse (Figura 5.5). Sin embargo, es necesario saber la dirección del servidor para poder proporcionársela a los agentes a la hora de desplegarlos. Para ello, se va a utilizar la opción de *logs*, con el siguiente comando:

```
./manage logs
```

La información que interesa es la url, como se puede observar en la Figura 5.6.

Mediante la ventana de *logs* también se pueden ver los registros añadidos y borrados del servidor, como se verá en la Sección 6.

```
rafa@rafa-VirtualBox:~/aca-py-performance/indy-tails-server/docker$ ./manage start
Starting docker_ngrok-tails-server_1 ... done
Recreating docker_tails-server_1 ... done
Run './manage logs' for logs
```

Figura 5.5: Inicio *indy-tails-server*.


```
" obj=tunnels.session err=nil
ngrok-tails-server_1 | t=2022-06-02T17:06:17+0000 lvl=info msg="accept failed"
obj=csess id=5854acbc9768 err="reconnecting session closed"
ngrok-tails-server_1 | t=2022-06-15T16:33:41+0000 lvl=info msg="no configuration paths supplied"
ngrok-tails-server_1 | t=2022-06-15T16:33:41+0000 lvl=info msg="using configuration at default config path" path=/home/ngrok/.ngrok2/ngrok.yml
ngrok-tails-server_1 | t=2022-06-15T16:33:41+0000 lvl=info msg="open config file" path=/home/ngrok/.ngrok2/ngrok.yml err=nil
ngrok-tails-server_1 | t=2022-06-15T16:33:41+0000 lvl=info msg="starting web service" obj=web addr=0.0.0.0:4040
ngrok-tails-server_1 | t=2022-06-15T16:33:41+0000 lvl=info msg="tunnel session started" obj=tunnels.session
ngrok-tails-server_1 | t=2022-06-15T16:33:41+0000 lvl=info msg="client session established" obj=csess id=8ced7a79d291
ngrok-tails-server_1 | t=2022-06-15T16:33:41+0000 lvl=info msg="started tunnel"
obj=tunnels name="command_line (http)" addr=http://tails-server:6543 url=http://f64a-150-214-205-62.ngrok.io
ngrok-tails-server_1 | t=2022-06-15T16:33:41+0000 lvl=info msg="started tunnel"
obj=tunnels name=command_line addr=http://tails-server:6543 url=https://f64a-150-214-205-62.ngrok.io
tails-server_1 | ====== Running on http://0.0.0.0:6543 ======
tails-server_1 | (Press CTRL+C to quit)
```

Figura 5.6: Url del servidor de revocación.

5.5. Implementación de los agentes

Por último, queda implementar los agentes Aries. Para ello, se ha utilizado como referencia el repositorio de Hyperledger dedicado a ACA-Py [29]. Se ha utilizado el *framework* para los agentes, y para los controladores, los creados en la carpeta demo del repositorio. Sin embargo, se han hecho algunas modificaciones para adaptar los controladores a las necesidades del proyecto. Estas modificaciones se irán definiendo a lo largo de la implementación.

5.5.1. *run_demo* y *ngrok-wait.sh*

En primer lugar, es necesario explicar como se despliegan los contenedores docker. Aquí es donde entra el *script* bash *run-demo*.

El único parámetro obligado que necesita el *script* es el agente que se quiere desplegar en el contenedor. A parte, se le pueden pasar más parámetros, que serán los que reciba el *script* de Python del agente correspondiente definido en el primer parámetro.

El *script run-demo*, independiente de los parámetros que se le pasen, compila y ejecuta un contenedor basado en la imagen contenida en el fichero *Dockerfile.demo*. Este fichero (que se encuentra en la carpeta docker del repositorio) le dice a Docker que imagen tiene que compilar, así como algunos requerimientos necesarios. También se añaden algunas carpetas del directorio puesto que el agente las va a necesitar (Listado 5.1).

```

1 FROM bcgovimages/von-image:py36-1.15-1
2
3 ENV ENABLE_PTVSD 0
4 ENV ENABLE_PYDEVD_PYCHARM 0
5 ENV PYDEVD_PYCHARM_HOST "host.docker.internal"
6
7 RUN mkdir bin && curl -L -o bin/jq \
8     https://github.com/stedolan/jq/releases/download/jq-1.6/jq-linux64 && \
9     chmod ug+x bin/jq
10
11 # Add and install Indy Agent code
12 ADD requirements*.txt ./
13
14 RUN pip3 install --no-cache-dir \
15     -r requirements.txt \
16     -r requirements.askar.txt \
17     -r requirements.bbs.txt \
18     -r requirements.dev.txt
19
20 ADD aries_cloudagent ./aries_cloudagent
21 ADD bin ./bin
22 ADD README.md ./
23 ADD scripts ./scripts
24 ADD setup.py ./
25
26 RUN pip3 install --no-cache-dir -e .
27 RUN mkdir demo logs && chown -R indy:indy demo logs && chmod -R ug+rw demo
28     logs
29
30 # Add and install demo code
31 ADD demo/requirements.txt ./demo/requirements.txt
32 RUN pip3 install --no-cache-dir -r demo/requirements.txt
33 ADD demo ./demo
34
35 ENTRYPOINT ["bash", "-c", "demo/ngrok-wait.sh \"$@\"", "--"]

```

Listado de código 5.1: Código de *Dockerfile.demo*.

En la última línea se puede ver como se ejecuta un *script* bash llamado *ngrok-wait.sh*. Este *script* tiene como función abrir el *localhost* en caso de que se quiera hacer público la red local y los agentes. Como no es el caso,

ya que se están realizando pruebas locales, el *script* no creará el túnel. La parte que interesa es la que se puede observar en el Listado 5.2. Lo que hace el *script* es ejecutar los *scripts* Python del agente elegido utilizando el compilador de Python, y con los argumentos pasados.

```

1
2 export AGENT_NAME=$1
3 shift
4 echo "Starting [$AGENT_NAME] agent with args @{$@}"
5 python -m demo.runners.$AGENT_NAME @{$@}

```

Listado de código 5.2: Código de *ngrok-wait.sh*.

5.5.2. Alice

Para la implementación del agente Alice se ha utilizado el script *alice.py* disponible en el repositorio de Hyperledger de ACA-Py [29], en la carpeta /demo/runners. Para ver el código consultar el repositorio. A continuación se explican los parámetros más importantes que se pueden utilizar con Alice:

- **--port <port>** (por defecto 8030): puerto del agente desplegado.
- **--revocation**: activa las credenciales revocables.
- **--aip <aip>** (por defecto 2.0): aip (1.0 o 2.0) que utilizará el agente, lo que determinará que protocolos usará.
- **--tails-server-base-url <tails-server-base-url>**: para indicar la url del servidor de revocación.

Para desplegar el agente Alice es necesario lo siguiente:

- Que esté desplegada la red *von-network* a la que se conectará el agente.
- En caso de ser credenciales revocables, que esté desplegado el servidor Tails.
- Utilizar el script *run-demo*. Para desplegar el agente usar el comando:

```
.\run_demo alice <parámetros>
```

5.5.3. Faber

Al igual que Alice, el código necesario para implementar a Faber está disponible en el repositorio de Hyperledger de ACA-Py [29], en la carpeta /demo/runners. Para ver el código consultar el repositorio. Los parámetros importantes así como su despliegue son similares al del agente Alice.

5.5.4. Acme

A diferencia de Alice y Faber, Acme ha sido implementado para el proyecto, con el objetivo de tener un escenario más grande y más representativo. El código está disponible en el repositorio del alumno Rafael Adán [33]. Los parámetros importantes así como su despliegue son los mismos que los de Alice y Faber.

5.6. Implementación del entorno de evaluación de rendimiento

La implementación del entorno de evaluación de rendimiento ha sido desarrollado casi íntegramente por el alumno. Se ha utilizado como referencia el script *performance.py* del repositorio de Hyperledger de ACA-Py [29], en la carpeta /demo/runners. Sin embargo, se ha modificado ampliamente para incluir las funcionalidades necesarias para el proyecto.

Por otra parte, se han creado diversos *scripts* para automatizar la ejecución del escenario con diferentes números de credenciales y diversas pruebas, para que los resultados sean los más precisos posibles.

5.6.1. *performance.py*

Es el *script* más importante del entorno. Este *script* es el que despliega el entorno en sí: despliega los dos agentes, Alice y Faber y, según los parámetros dados, crea un escenario u otro. El código se encuentra en el repositorio del alumno Rafael Adán [33] en la carpeta /demo/runners.

Los parámetros más importantes que se pueden usar en el *script performance.py* son los siguientes. Para más información sobre los parámetros no mencionados, utilizar la opción *--help*.

- **--count:** número de credenciales que se quieren emitir (y, si es el caso, presentar y/o revocar).
- **-port:** puerto inicial donde se empieza a escuchar.
- **--ping:** en vez de mandar credenciales se mandan pings.
- **--revocation:** activa credenciales revocables.
- **--revoke_credentials:** revoca las credenciales emitidas anteriormente.

84 5.6. Implementación del entorno de evaluación de rendimiento

- **--proof_presentation:** activa la presentación de pruebas de credencial por parte del titular.
- **--publish_revocations_at_once:** todas las revocaciones se publican juntas al final.
- **--tails_server_base_url <tails-server-base-url >:** para indicar la url del servidor de revocación.

5.6.2. proof_presentation_CS.py

Este *script* automatiza el escenario de emisión y presentación de credenciales simples. El código se puede ver en el Listado 5.3.

```
1 ...
2 ...
3 AUTOR: RAFAEL ADAN LOPEZ.
4 FECHA: 20 DE JUNIO DE 2022
5
6 UNIVERSIDAD DE GRANADA
7 TRABAJO DE FIN DE GRADO: EVALUACION DE RENDIMIENTO DE ENTORNO SSI BASADO
EN BLOCKCHAIN
8
9
10 SCRIPT PARA EJECUTAR UN ENTORNO DE PRUEBAS DE EMISION Y POSTERIOR
    PRESENTACION DE CREDENCIALES SIMPLES (CS) (SIN POSIBILIDAD DE
    REVOCACION) ENTRE DOS AGENTES.
11
12 CAMBIAR LOS DIRECTORIOS PARA USO DEL SCRIPT, PUESTO QUE ESTA PUESTO LA
    RUTA DEL DESARROLLADOR.
13
14 ...
15
16 #Importacion de librerias necesarias
17 import subprocess
18 import os
19
20 #Numero de credenciales que se quieran probar
21 credenciales = [10, 20, 50, 100, 150, 200, 250, 300, 400, 500]
22
23 #Numero de pruebas realizadas con cada credencial
24 pruebas = 25
25
26 pid = os.getpid()
27
28
29 for cred in credenciales:
30     for prueba in range(1, pruebas + 1):
31
            #En caso de no existir el directorio para guardar los logs, lo
            crea
            if(not os.path.exists(f"/home/rafa/aca-py-performance/demo/pruebas
/CS/{cred}_credenciales")):
                os.makedirs(f"/home/rafa/aca-py-performance/demo/pruebas/CS/{
cred}_credenciales")
            os.makedirs(f"/home/rafa/aca-py-performance/demo/pruebas/CS/{
cred}_credenciales/datosCPUyRAM")
36
```

```

37     #Ejecuta dos procesos: la prueba en s utilizando el script bash
38     run_demo y un proceso que mide la CPU y RAM utilizada y lo guarda en
39     logs
40     cpu_process = subprocess.Popen(["bash", "cpu_y_ram.sh", f"{pid}",
41                                     f"/home/rafa/aca-py-performance/demo/pruebas/CS/{cred}_credentials/
42                                     datosCPUyRAM/CPU_{cred}_credentials_prueba_{prueba}.txt"])
43     p = subprocess.Popen(["bash", "run_demo", "performance", "--count",
44                           , f'{cred}', "--proof_presentation"], stdout=subprocess.PIPE, text=
45                           True)
46
47     #Se guarda la salida del proceso en ficheros logs
48     file = open(f"/home/rafa/aca-py-performance/demo/pruebas/CS/{cred}_credentials/prueba{prueba}_con_{cred}_credentials.txt", "w")
49     file.write(p.communicate()[0])
50     file.close()
51
52     #Matar proceso de CPU y RAM
53     cpu_process.kill()
54
55     #Optimizacion de las imagenes docker para que no consuman espacio
56     #adicional
57     subprocess.Popen('yes | docker image prune', shell = 'False')
58
59
60 print("Ejecucion finalizada")

```

Listado de código 5.3: Código de *proof-presentation_CS.py*.

5.6.3. *proof_presentation_CR_no_revocation.py*

Este *script* automatiza el escenario de emisión y presentación de credenciales revocables, sin llegar a revocarlas. El código se puede ver en el Listado 5.4.

```

1
2
3 """
4 AUTOR: RAFAEL ADAN LOPEZ.
5 FECHA: 20 DE JUNIO DE 2022
6
7 UNIVERSIDAD DE GRANADA
8 TRABAJO DE FIN DE GRADO: EVALUACION DE RENDIMIENTO DE ENTORNO SSI BASADO
9 EN BLOCKCHAIN
10 SCRIPT PARA EJECUTAR UN ENTORNO DE PRUEBAS DE EMISION Y POSTERIOR
11     PRESENTACION DE CREDENCIALES REVOCABLES (CR)
12 (CON POSIBILIDAD DE REVOCACION) ENTRE DOS AGENTES, PERO SIN LLEGAR A
13     REVOCARLAS.
14
15 """
16
17 #Importacion de librerias necesarias
18 import os
19 import subprocess
20
21
22 def main(

```

86 5.6. Implementación del entorno de evaluación de rendimiento

```
23     tails_server_base_url: str = None, #url del servidor tails para las
24     revocaciones
25   ):
26     #Numero de credenciales que se quieran probar
27     credenciales = [10, 20, 50, 100, 150, 200, 250, 300, 400, 500]
28
29     #Numero de pruebas realizadas con cada credencial
30     pruebas = 25
31
32     pid = os.getpid()
33
34     for cred in credenciales:
35       for prueba in range(1, pruebas + 1):
36         #En caso de no existir el directorio para guardar los logs, lo
37         crea
38         if(not os.path.exists(f"/home/rafa/aca-py-performance/demo/
39         pruebas/CR/sin_revocar/{cred}_credenciales")):
40           os.makedirs(f"/home/rafa/aca-py-performance/demo/pruebas/
41           CR/sin_revocar/{cred}_credenciales")
42           os.makedirs(f"/home/rafa/aca-py-performance/demo/pruebas/
43           CR/sin_revocar/{cred}_credenciales/datosCPUyRAM")
44
45         #Ejecuta dos procesos: la prueba en s utilizando el script
46         bash_run_demo y un proceso que mide la CPU y RAM utilizada y lo guarda
47         en logs
48         cpu_process = subprocess.Popen(["bash", "cpu_y_ram.sh", f"{pid}
49         ", f"/home/rafa/aca-py-performance/demo/pruebas/CR/sin_revocar/{cred}
50         _credenciales/datosCPUyRAM/CPU_{cred}_credenciales_prueba_{prueba}.txt
51         "])
52
53         p = subprocess.Popen(["bash", "run_demo", "performance", "--
54         count", f"{cred}", "--proof_presentation", "--revocation", "--tails-
55         server-base-url", f"{tails_server_base_url}"], stdout=subprocess.PIPE,
56         text= True)
57
58         #Se guarda la salida del proceso en ficheros logs
59         file = open(f"/home/rafa/aca-py-performance/demo/pruebas/CR/
60         sin_revocar/{cred}_credenciales/prueba{prueba}_con_{cred}_credenciales
61         .txt", "w")
62         file.write(p.communicate()[0])
63         file.close()
64
65         #Matar proceso CPU y RAM
66         cpu_process.kill()
67
68         #Optimizacion de las imagenes docker para que no consuman
69         espacio adicional
70         subprocess.Popen('yes | docker image prune', shell = 'False')
71
72     print("Ejecucion finalizada")
73
74 if __name__ == "__main__":
75   import argparse
76
77   parser = argparse.ArgumentParser(
78     description="Script para ejecucion entorno CR sin llegar a
79     revocar"
80   )
81
82   parser.add_argument(
83     "--tails-server-base-url",
84     type=str,
85     metavar=<tails-server-base-url>,
```

```

68         help="Tails server base url",
69     )
70
71     args = parser.parse_args()
72
73
74     try:
75         main(
76             args.tails_server_base_url
77         )
78
79     except KeyboardInterrupt:
80         os._exit(1)

```

Listado de código 5.4: Código de *proof_presentation_CR_no_revocation.py*.

5.6.4. *proof_presentation_CR_one_by_one_revocation.py*

Este *script* automatiza el escenario de emisión y presentación de credenciales revocables, revocándolas y publicándolas de una en una. El código se puede ver en el Listado 5.5.

```

1 """
2
3 AUTOR: RAFAEL ADAN LOPEZ.
4 FECHA: 20 DE JUNIO DE 2022
5
6 UNIVERSIDAD DE GRANADA
7 TRABAJO DE FIN DE GRADO: EVALUACION DE RENDIMIENTO DE ENTORNO SSI BASADO
EN BLOCKCHAIN
8
9
10 SCRIPT PARA EJECUTAR UN ENTORNO DE PRUEBAS DE EMISION Y POSTERIOR
    PRESENTACION DE CREDENCIALES REVOCABLES (CR)
11 (CON POSIBILIDAD DE REVOCACION) ENTRE DOS AGENTES, Y REVOCANDO DE UNA EN
    UNA.
12
13 CAMBIAR LOS DIRECTORIOS PARA USO DEL SCRIPT, PUESTO QUE ESTA PUESTO LA
    RUTA DEL DESARROLLADOR.
14
15 """
16
17 #Importacion de las librerias necesarias
18 import os
19 import subprocess
20
21 def main(
22     tails_server_base_url: str = None, #url del servidor tails para las
        revocaciones
23 ):
24
25     #Número de credenciales que se quieren probar y el número de pruebas
        a realizar con cada una
26     credenciales = [10, 20, 50, 100, 150, 200, 250, 300, 400, 500]
27     pruebas = 25
28
29     pid = os.getpid()
30
31     for cred in credenciales:
32         for prueba in range(14, pruebas + 1):

```

88 5.6. Implementación del entorno de evaluación de rendimiento

```
33         #En caso de no existir el directorio para guardar los logs, lo
34         crea.
35             if(not os.path.exists(f"/home/rafa/aca-py-performance/demo/
36             pruebas/CR/revocando_uno_a_uno/{cred}_credenciales")):
37                 os.makedirs(f"/home/rafa/aca-py-performance/demo/pruebas/
38             CR/revocando_uno_a_uno/{cred}_credenciales")
39                 os.makedirs(f"/home/rafa/aca-py-performance/demo/pruebas/
40             CR/revocando_uno_a_uno/{cred}_credenciales/datosCPUyRAM")
41
42             #Ejecuta dos procesos: la prueba en s utilizando el script
43             bash run_demo y un proceso que mide la CPU y RAM utilizada y lo guarda
44             en logs
45                 cpu_process = subprocess.Popen(["bash", "cpu_y_ram.sh", f"{pid}
46             ", f"/home/rafa/aca-py-performance/demo/pruebas/CR/
47             revocando_uno_a_uno/{cred}_credenciales/datosCPUyRAM/CPU_{cred}
48             _credenciales_prueba_{prueba}.txt"])
49                 p = subprocess.Popen(["bash", "run_demo", "performance", "--
50             count", f"{cred}", "--revocation", "--revoke_credentials", "--tails-
51             server-base-url", f"{tails_server_base_url}"], stdout=subprocess.PIPE,
52             text= True)
53
54             #Se guarda la salida del proceso en ficheros logs
55             file = open(f"/home/rafa/aca-py-performance/demo/pruebas/CR/
56             revocando_uno_a_uno/{cred}_credenciales/prueba{prueba}_con_{cred}
57             _credenciales.txt", "w")
58                 file.write(p.communicate()[0])
59                 file.close()
60
61             #Matar proceso cpu y ram
62             cpu_process.kill()
63
64             #Optimizacin de las im genes docker para que no consuman
65             espacio adicional
66             subprocess.Popen('yes | docker image prune', shell = 'False')
67
68             print("Ejecuci n finalizada")
69
70
71 if __name__ == "__main__":
72     import argparse
73
74     parser = argparse.ArgumentParser(
75         description="Script para ejecuci n entorno CR sin llegar a
76         revocar"
77     )
78
79     parser.add_argument(
80         "--tails-server-base-url",
81         type=str,
82         metavar="",
83         help="Tails server base url",
84     )
85     args = parser.parse_args()
86
87
88 try:
89     main(
90         args.tails_server_base_url
91     )
92
93 except KeyboardInterrupt:
```

```
77     os._exit(1)
```

Listado de código 5.5: Código de *proof_presentation_CR_one_by_one_revocation.py*.

5.6.5. *proof_presentation_CR_all_at_once_revocation.py*

Este *script* automatiza el escenario de emisión y presentación de credenciales revocables, revocándolas y publicándolas de una en una. El código se puede ver en el Listado 5.6.

```
1 """
2 """
3 AUTOR: RAFAEL ADAN LOPEZ.
4 FECHA: 20 DE JUNIO DE 2022
5
6 UNIVERSIDAD DE GRANADA
7 TRABAJO DE FIN DE GRADO: EVALUACION DE RENDIMIENTO DE ENTORNO SSI BASADO
   EN BLOCKCHAIN
8
9 SCRIPT PARA EJECUTAR UN ENTORNO DE PRUEBAS DE EMISION Y POSTERIOR
   PRESENTACION DE CREDENCIALES REVOCABLES (CR)
10 (CON POSIBILIDAD DE REVOCACION) ENTRE DOS AGENTES, Y REVOCANDO TODAS AL
    FINAL.
11
12 CAMBIAR LOS DIRECTORIOS PARA USO DEL SCRIPT, PUESTO QUE ESTA PUESTO LA
   RUTA DEL DESARROLLADOR.
13 """
14 """
15
16
17 #Importacion de librerias necesarias
18 import os
19 import subprocess
20
21 def main(
22     tails_server_base_url: str = None, #url del servidor tails para las
       revocaciones
23 ):
24
25     #Numero de credenciales que se quieren probar y el n mero de pruebas
       a realizar con cada una
26     credenciales = [10, 20, 50, 100, 150, 200, 250, 300, 400, 500]
27     pruebas = 25
28
29     pid = os.getpid()
30
31     for cred in credenciales:
32         for prueba in range(1, pruebas + 1):
33
34             #En caso de no existir el directorio para guardar los logs, lo
               crea
35             if(not os.path.exists(f"/home/rafa/aca-py-performance/demo/
pruebas/CR/revocando/{cred}_credenciales")):
36                 os.makedirs(f"/home/rafa/aca-py-performance/demo/pruebas/
CR/revocando/{cred}_credenciales")
37                 os.makedirs(f"/home/rafa/aca-py-performance/demo/pruebas/
CR/revocando/{cred}_credenciales/datosCPUyRAM")
```

90 5.6. Implementación del entorno de evaluación de rendimiento

```
39         #Ejecuta dos procesos: la prueba en s utilizando el script
40         bash run_demo y un proceso que mide la CPU y RAM utilizada y lo guarda
41         en logs
42         cpu_process = subprocess.Popen(["bash", "cpu_y_ram.sh", f"{pid}",
43             f"/home/rafa/aca-py-performance/demo/pruebas/CR/revocando/{cred}_credentials/datosCPUyRAM/CPU_{cred}_credentials_prueba_{prueba}.txt"])
44         p = subprocess.Popen(["bash", "run_demo", "performance", "--count",
45             f"{cred}", "--revocation", "--revoke_credentials", "--publish_revocations_at_once", "--tails-server-base-url", f"{tails_server_base_url}" ],
46             stdout=subprocess.PIPE, text= True)
47
48         #Se guarda la salida del proceso en ficheros logs
49         file = open(f"/home/rafa/aca-py-performance/demo/pruebas/CR/revocando/{cred}_credentials/prueba{prueba}_con_{cred}_credentials.txt", "w")
50         file.write(p.communicate()[0])
51         file.close()
52
53         #Matar proceso CPU y RAM
54         cpu_process.kill()
55
56         #Optimizacion de las imagenes docker para que no consuman
57         #espacio adicional
58         subprocess.Popen('yes | docker image prune', shell = 'False')
59
60         print("Ejecucion finalizada")
61
62 if __name__ == "__main__":
63     import argparse
64
65     parser = argparse.ArgumentParser(
66         description="Script para ejecucion entorno CR y publicando la
67         revolucion de una vez"
68     )
69
70     parser.add_argument(
71         "--tails-server-base-url",
72         type=str,
73         metavar="",
74         help="Tails server base url",
75     )
76
77     args = parser.parse_args()
78
79     try:
80         main(
81             args.tails_server_base_url,
82         )
83     except KeyboardInterrupt:
84         os._exit(1)
```

Listado de código 5.6: Código de *proof_presentation_CR_all_at_once_revocation.py*.

5.6.6. *plot_CS.py*

Este *script* plotea los datos generados del *script* *proof_presentation_CS.py*. Para ello, extrae los tiempos de los *logs* generados y, utilizando la librería *Pandas*, se crean *dataFrames* para mostrar los tiempos y los datos estadísticos. Las tablas generadas son tanto ploteadas al momento como exportadas a tablas *Excel*.

En el Listado 5.7 se muestra el código del *script* (en los siguientes *scripts* no se hará debido a su similitud). Se puede observar como se van abriendo los *logs* generados anteriormente y buscando los patrones como ‘*Startup duration.*’ para poder extraer los tiempos. Así se extraen todos los tiempos. Se crea un *dataFrame* para cada tipo de tiempo, con el objetivo de crear diagramas de cajas y bigotes y visualizar los datos estadísticamente. Tras ello se hace un *dataFrame* para cada número de credencial, que se plotea y se extrae a una tabla *Excel*. También se crea una tabla *Excel* con los datos estadísticos de la tabla anterior. Finalmente, se hace una tabla final con las medias de los 10 números de credenciales y se plotea cada serie.

```
1
2
3
4 """
5 AUTOR: RAFAEL ADAN LOPEZ.
6 FECHA: 20 DE JUNIO DE 2022
7
8 UNIVERSIDAD DE GRANADA
9 TRABAJO DE FIN DE GRADO: EVALUACION DE RENDIMIENTO DE ENTORNO SSI BASADO
   EN BLOCKCHAIN
10
11 SCRIPT PARA PLOTEAR LAS PRUEBAS GENERADAS CON EL SCRIPT
   PROOF_PRESENTATION_CS.PY
12
13 CAMBIAR LOS DIRECTORIOS PARA USO DEL SCRIPT, PUESTO QUE ESTA PUESTO LA
   RUTA DEL DESARROLLADOR.
14
15 """
16
17 #Importacion de librerias usadas
18 import re
19 import subprocess
20 import os
21 import pandas as pd
22 import matplotlib.pyplot as plt
23
24
25
26 #Vector de vectores donde se guardaran los tiempos y la CPU y RAM usada (
   cred X pruebas)
27 tiempos_startup = []
28 tiempos_connect = []
29 tiempos_publish = []
30 tiempos_avg_credential = []
31 tiempos_avg_proof = []
32 tiempos_total = []
```

92 5.6. Implementación del entorno de evaluación de rendimiento

```
33 cpu_array = []
34 ram_array = []
35 pid = os.getpid()
36
37 #Número de credenciales y pruebas
38 credenciales = [10, 20, 50, 100, 150, 200, 250, 300, 400, 500]
39 pruebas = 25
40
41 for cred in credenciales:
42
43     #Vector de tiempos con las 25 pruebas para una credencial
44     tiempos_startup_p = []
45     tiempos_connect_p = []
46     tiempos_publish_p = []
47     tiempos_avg_credential_p = []
48     tiempos_avg_proof_p = []
49     tiempos_total_p = []
50     tiempos_revoke_p = []
51     ram_p = []
52     cpu_p = []
53
54
55     for prueba in range(1,pruebas+1):
56
57         #Se abre el archivo de log para la prueba 'prueba' y el numero de
58         #credencial 'cred' y se buscan los patrones para extraer los tiempos
59         with open(f"/home/rafa/aca-py-performance/demo/pruebas/CS/{cred}_"
60                   "_credenciales/prueba{prueba}_con_{cred}_credenciales.txt","r") as file
61             :
62                 for line in file:
63                     if re.search("Startup duration:",line):
64                         primero = line.index(":")
65                         segundo = line.index("s")
66                         tiempos_startup_p.append(float(line[primero+2:segundo]))
67
68                     elif re.search("Connect duration:",line):
69                         primero = line.index(":")
70                         segundo = line.index("s", 26)
71                         tiempos_connect_p.append(float(line[primero+2:segundo]))
72
73                     elif re.search("Publish duration:",line):
74                         primero = line.index(":")
75                         segundo = line.index("s", 26)
76                         tiempos_publish_p.append(float(line[primero+2:segundo]))
77
78                     elif re.search("Average time per credential:",line):
79                         primero = line.index(":")
80                         segundo = line.index("s")
81                         tiempos_avg_credential_p.append(float(line[primero+2:segundo]))
82
83                     elif re.search("Average time per proofs:",line):
84                         primero = line.index(":")
85                         segundo = line.index("s", 60)
86                         tiempos_avg_proof_p.append(float(line[primero+2:segundo]))
87
88                     elif re.search("Total runtime:",line):
89                         primero = line.index(":")
90                         segundo = line.index("s")
91                         tiempos_total_p.append(float(line[primero+2:segundo]))
92
93
94         #Se abre el archivo de log para la prueba 'prueba' y el numero de
95         #credencial 'cred' y se buscan los patrones para extraer la CPU y RAM
```

```
usada
86     with open(f"/home/rafa/aca-py-performance/demo/pruebas/CS/{cred}
87         _credenciales/datosCPUyRAM/CPU_{cred}_credenciales_prueba_{prueba}.txt
88         ","r") as file:
89             total_cpu = 0
90             total_ram = 0
91             contador = 0
92             for line in file:
93                 total_cpu = total_cpu + float(line[1:6])
94                 total_ram = total_ram + float(line[7:10])
95                 contador = contador + 1
96
97             cpu_p.append(total_cpu/contador)
98             ram_p.append(total_ram/contador)
99
100 #En caso de que alguna prueba haya tenido errores, es posible que no
101 #haya tiempos. En ese caso, se rellena los elementos necesarios para
102 #llegar a 25 pruebas
103 #con la media del vector
104 while(len(tiempo_startup_p) != pruebas):
105     tiempo_startup_p.append(sum(tiempo_startup_p)/len(
106     tiempo_startup_p))
107 while(len(tiempo_publish_p) != pruebas):
108     tiempo_publish_p.append(sum(tiempo_publish_p)/len(
109     tiempo_publish_p))
110 while(len(tiempo_connect_p) != pruebas):
111     tiempo_connect_p.append(sum(tiempo_connect_p)/len(
112     tiempo_connect_p))
113 while(len(tiempo_avg_credential_p) != pruebas):
114     tiempo_avg_credential_p.append(sum(tiempo_avg_credential_p)/len(
115     tiempo_avg_credential_p))
116 while(len(tiempo_avg_proof_p) != pruebas):
117     tiempo_avg_proof_p.append(sum(tiempo_avg_proof_p)/len(
118     tiempo_avg_proof_p))
119 while(len(tiempo_total_p) != pruebas):
120     tiempo_total_p.append(sum(tiempo_total_p)/len(tiempo_total_p))
121
122 #Se añade el vector de 25 pruebas para la credencial 'cred' al vector
123 #de vectores.
124 tiempo_startup.append(tiempo_startup_p)
125 tiempo_connect.append(tiempo_connect_p)
126 tiempo_publish.append(tiempo_publish_p)
127 tiempo_avg_credential.append(tiempo_avg_credential_p)
128 tiempo_avg_proof.append(tiempo_avg_proof_p)
129 tiempo_total.append(tiempo_total_p)
130 cpu_array.append(cpu_p)
131 ram_array.append(ram_p)
132
133 #Ploteo los tiempos tiempos en diagramas de cajas y bigotes para ver los
134 #datos estadísticos
135
136 datastartup = pd.DataFrame({
137     '10 credenciales' : tiempo_startup[0],
138     '20 credenciales' : tiempo_startup[1],
139     '50 credenciales' : tiempo_startup[2],
140     '100 credenciales' : tiempo_startup[3],
141     '150 credenciales' : tiempo_startup[4],
142     '200 credenciales' : tiempo_startup[5],
143     '250 credenciales' : tiempo_startup[6],
144     '300 credenciales' : tiempo_startup[7],
145     '400 credenciales' : tiempo_startup[8],
146     '500 credenciales' : tiempo_startup[9],
```

94 5.6. Implementación del entorno de evaluación de rendimiento

```
136 })
137
138 datastartup.to_excel(f"tablas_excel/CS/startup.xlsx")
139 datastartup.plot(kind='box', title=f"Startup CS", legend=True, xlabel="Numero de credenciales", ylabel = "Tiempo (s)")
140 plt.show()
141
142 dataconnect= pd.DataFrame({
143     '10 credenciales' : tiempos_connect[0],
144     '20 credenciales' : tiempos_connect[1],
145     '50 credenciales' : tiempos_connect[2],
146     '100 credenciales': tiempos_connect[3],
147     '150 credenciales': tiempos_connect[4],
148     '200 credenciales': tiempos_connect[5],
149     '250 credenciales': tiempos_connect[6],
150     '300 credenciales': tiempos_connect[7],
151     '400 credenciales': tiempos_connect[8],
152     '500 credenciales': tiempos_connect[9],
153 })
154
155 dataconnect.to_excel(f"tablas_excel/CS/connect.xlsx")
156 dataconnect.plot(kind='box', title=f"Connect CS", legend=True, xlabel="Numero de credenciales", ylabel = "Tiempo (s)")
157 plt.show()
158
159 datapublish = pd.DataFrame({
160     '10 credenciales' : tiempos_publish[0],
161     '20 credenciales' : tiempos_publish[1],
162     '50 credenciales' : tiempos_publish[2],
163     '100 credenciales': tiempos_publish[3],
164     '150 credenciales': tiempos_publish[4],
165     '200 credenciales': tiempos_publish[5],
166     '250 credenciales': tiempos_publish[6],
167     '300 credenciales': tiempos_publish[7],
168     '400 credenciales': tiempos_publish[8],
169     '500 credenciales': tiempos_publish[9],
170 })
171
172 datapublish.to_excel(f"tablas_excel/CS/publish.xlsx")
173 datapublish.plot(kind='box', title=f"Publish CS", legend=True, xlabel="Numero de credenciales", ylabel = "Tiempo (s)")
174 plt.show()
175
176 dataavgcred = pd.DataFrame({
177     '10 credenciales' : tiempos_avg_credential[0],
178     '20 credenciales' : tiempos_avg_credential[1],
179     '50 credenciales' : tiempos_avg_credential[2],
180     '100 credenciales': tiempos_avg_credential[3],
181     '150 credenciales': tiempos_avg_credential[4],
182     '200 credenciales': tiempos_avg_credential[5],
183     '250 credenciales': tiempos_avg_credential[6],
184     '300 credenciales': tiempos_avg_credential[7],
185     '400 credenciales': tiempos_avg_credential[8],
186     '500 credenciales': tiempos_avg_credential[9],
187 })
188
189 dataavgcred.to_excel(f"tablas_excel/CS/avgcred.xlsx")
190 dataavgcred.plot(kind='box', title=f"Avg per credential CS", legend=True,
191                   xlabel="Numero de credenciales", ylabel = "Tiempo (s)")
191 plt.show()
192
193 dataavgproof = pd.DataFrame({}
```

```
194     '10 credenciales' : tiempos_avg_proof[0],
195     '20 credenciales' : tiempos_avg_proof[1],
196     '50 credenciales' : tiempos_avg_proof[2],
197     '100 credenciales' : tiempos_avg_proof[3],
198     '150 credenciales' : tiempos_avg_proof[4],
199     '200 credenciales' : tiempos_avg_proof[5],
200     '250 credenciales' : tiempos_avg_proof[6],
201     '300 credenciales' : tiempos_avg_proof[7],
202     '400 credenciales' : tiempos_avg_proof[8],
203     '500 credenciales' : tiempos_avg_proof[9],
204   })
205
206 dataavgproof.to_excel(f"tablas_excel/CS/avgproof.xlsx")
207 dataavgproof.plot(kind='box', title=f"Avg per proof CS", legend=True,
208                   xlabel="Numero de credenciales", ylabel = "Tiempo (s)")
209 plt.show()
210
211 datatotal = pd.DataFrame({
212     '10 credenciales' : tiempos_total[0],
213     '20 credenciales' : tiempos_total[1],
214     '50 credenciales' : tiempos_total[2],
215     '100 credenciales' : tiempos_total[3],
216     '150 credenciales' : tiempos_total[4],
217     '200 credenciales' : tiempos_total[5],
218     '250 credenciales' : tiempos_total[6],
219     '300 credenciales' : tiempos_total[7],
220     '400 credenciales' : tiempos_total[8],
221     '500 credenciales' : tiempos_total[9],
222   })
223
224 datatotal.to_excel(f"tablas_excel/CS/total.xlsx")
225 datatotal.plot(kind='box', title=f"Total CS no revocation", legend=True,
226                  xlabel="Numero de credenciales", ylabel = "Tiempo (s)")
227 plt.show()
228
229 #Se pasa las credenciales a string para el dataframe
230 credenciales_string = []
231 for credentials in credenciales:
232     credenciales_string.append(str(credentials))
233
234 #Vectores con las medias de las pruebas para cada credencial
235 medias_startup = []
236 medias_connect = []
237 medias_publish = []
238 medias_avg_credential = []
239 medias_avg_proof = []
240 medias_total = []
241
242 #Creacion de directorios para tablas excel
243 if(not os.path.exists(f"/home/rafa/aca-py-performance/demo/tablas_excel/CS
244   ")):
245     os.makedirs(f"/home/rafa/aca-py-performance/demo/tablas_excel/CS")
246
247 for cred in range(1, len(credenciales)+1):
248
249     #Se crea un DataFrame para cada credencial
250     dataframe = None
251     startup = tiempos_startup[cred-1]
252     connect = tiempos_connect[cred-1]
```

96 5.6. Implementación del entorno de evaluación de rendimiento

```
253     total = tiempos_total[cred-1]
254     cpu = cpu_array[cred-1]
255     ram = ram_array[cred-1]
256     dataframe = pd.DataFrame({
257         'Startup': startup,
258         'Connect': connect,
259         'Publish': publish,
260         'Average cred': avg_credential,
261         'Average proof': avg_proof,
262         'Total': total,
263         'CPU': cpu,
264         'RAM': ram
265     })
266
267     #Se pasa a float y se muestran los datos estadísticos de las pruebas
268     #para cada credencial
269     dataframe = dataframe.astype(float)
270     print(f"Datos estadísticos para {credenciales[cred-1]} credenciales \
271         \n")
272     print(dataframe.describe())
273     print("\n")
274
275     dataframe.to_excel(f"tablas_excel/CS/{credenciales[cred-1]} \
276         _credenciales.xlsx")
277     dataframe.describe().to_excel(f"tablas_excel/CS/{credenciales[cred-1]} \
278         _credenciales_describe.xlsx")
279
280     #Se plotea un esquema de cajas y bigotes del DataFrame creado
281     dataframe.plot(kind='box', title=f"Pruebas para {credenciales[cred-1]} \
282         credenciales")
283
284     #Se añade al vector de medias la media de cada tiempo del DataFrame
285     medias_startup.append(dataframe["Startup"].mean())
286     medias_connect.append(dataframe["Connect"].mean())
287     medias_publish.append(dataframe["Publish"].mean())
288     medias_avg_credential.append(dataframe["Average cred"].mean())
289     medias_avg_proof.append(dataframe["Average proof"].mean())
290     medias_total.append(dataframe["Total"].mean())
291
292     #Para mostrar los ploteos del bucle for
293     plt.show()
294
295     #Se crea un DataFrame con las medias calculadas anteriormente
296     df_final = pd.DataFrame({
297         'Startup': medias_startup,
298         'Connect': medias_connect,
299         'Publish': medias_publish,
300         'Average cred': medias_avg_credential,
301         'Average proof': medias_avg_proof,
302         'Total': medias_total,
303     }, index = credenciales_string)
304
305     #Se muestran los datos de la tabla final con los tiempos para cada numero
306     #de credencial
307     print(f"TABLA FINAL \n")
308     print(df_final)
309     df_final.to_excel("tablas_excel/CS/tablafinal.xlsx")
310
311     #Se plotea una grafica para cada tipo de tiempo con las medias calculadas
312     #para cada numero de credencial
313     df_final["Startup"].plot(title="FINAL Startup")
314     plt.show()
```

```
308 df_final["Connect"].plot(title="FINAL Connect")
309 plt.show()
310
311 df_final["Publish"].plot(title="FINAL Publish")
312 plt.show()
313
314 df_final["Average cred"].plot(title="FINAL Average cred")
315 plt.show()
316
317 df_final["Average proof"].plot(title="FINAL Average proof")
318 plt.show()
319
320 df_final["Total"].plot(title="FINAL Total")
321 plt.show()
```

Listado de código 5.7: Código de *plot_CS.py*.

5.6.7. *plot_CR_no_revocation.py*

Este script tiene la misma funcionalidad que el código del Listado 5.7 pero utiliza los datos generados del script *plot_CR_no_revocation.py*.

5.6.8. *plot_CR_one_by_one_revocation.py*

Este script tiene la misma funcionalidad que el código del Listado 5.7 pero utiliza los datos generados del script *plot_CR_one_by_one_revocation.py*.

5.6.9. *plot_CR_all_at_once_revocation.py*

Este script tiene la misma funcionalidad que el código del Listado 5.7 pero utiliza los datos generados del script *plot_CR_all_at_once_revocation.py*.

Capítulo 6

Evaluación y pruebas

En este capítulo se pretende evaluar el entorno implementado. En primer lugar, se realizará una primera prueba del entorno de evaluación de funcionalidades. Se probarán las funcionalidades y se comparará con el diseño teórico. Tras esto, se analizarán los resultados conseguidos tras la implementación y despliegue del entorno de evaluación de rendimiento.

6.1. Prueba de evaluación de funcionalidades

Se van a realizar distintas pruebas para entender los tipos de funcionalidades que proporciona Hyperledger Aries. Para ello, se desplegará el diseño explicado en la Sección 4.7, e implementado según la Sección 5.

6.1.1. Despliegue de los agentes

En primer lugar, se desplegarán los agentes Alice, Faber y Acme. Para ello, evidentemente, es necesario que la red *von-network* este previamente desplegada, así como el servidor de revocación *indy-tails-server*, según se explica en la sección de Implementación (Sección 5).

Tras haber desplegado la red y el servidor de revocación, se desplegarán los agentes con los parámetros *--revocation* y *--tails-server-base-url* junto con la url del servidor de revocación, para que los agentes sepan la localización del servidor (siguiendo las instrucciones explicadas en la Sección 5.5). Una vez hecho se desplegarán los agentes, que estarán a la espera de una conexión. Se puede observar como al desplegar los agentes, se crea una *wallet* así como el agente (con agente se hace referencia al *framework*) (Figura 6.1).

A diferencia de Alice, Faber en su despliegue tiene que publicar en el ledger varios elementos:

```
#7 Provision an agent and wallet, get back configuration details
# Create webnode [testagent] on port 8053
Alice --url "http://node1/pynode/testagent" --port 8053
Alice --url "http://node1/pynode/testagent" --port 8053
#    --auto-ping-connections --auto-respond-messages --inbound-transport "http", 0.0.0.0, 8030; --outbound-transport "http", 0.0.0.0, 8030
#    --admin-insecure-mode --admin-port-type "tiny" --wallet-name, alice.agent.B85187 --wallet-key, "alice.agent.B85187" --no-reserve-exchange-records --auto-provision --public-invites, --emit-new-didcomm-prefix "-" --genesis-transactions, [{"reqSignature": ""}, {"reqSignature": ""}], --data [{"alias": "Node1", "blskey": "AN8AaNG19jG0qkRnhN6EFD6t1x0nYy9E9kr1nJkvglM36IMFF7m56Q41QhVn2233nvesFvSn9n1lMFNJBtyVnWnHMTv7n6fLu3LBURkyAVCHsf1hHeU9XcahexhZ6h1CylP2zD9druvSwpehB8RklKm3a", "blskey": "blskey"}, {"blskey": "RAhY1CvIopCtVTPm7cTsYr18AUJxJdhNdebb1hKjlBXNwHnJx0XyHfMyvhqRhoqT37QYemH51k0pL7RANT22LzE2HkgJ3B0RQnJy8y7vJn9Ch7AtBn9Q1kbaVLwEmPwB14EmlwPwRPAuXy", "blskey": "blskey"}, {"blskey": "f2e010e8944914e7a9d5296d46f539f39e456461cbe2d9508b", "ver": "1"}], --client-ip, "10.0.2.15", --client-port, "9702", --node_id, "10.0.2.15", --node_port, "9701", --services [{"VALIDATOR_ID": "1"}, {"desc": "GwpD0hBcQp0Tz7QzfpTgtJchqBzXpK3Xp05d9h", "metadata": [{"from": "Th7g1TaRzVnRyplabds51Y"}, {"type": "0"}]}, {"txmMetadata": {"seqNo": "1", "txId": "f2e010e8944914e7a9d5296d46f539f39e456461cbe2d9508b", "ver": "1"}], --reqSignature: "", {"tx": {"data": [{"alias": "Node2", "blskey": "37ApvOxKzKh7gk5le2XyLXuXpLGMDLBD7S5GLb251s3f7Qz5tKzWm1kuCH1QKpnzaFrlG1vN98r138m9f97ENz04lQdHjzQsUuo0nH7Prp7oN02JpNaEvbRjYkCbavJ3XKXheszhpZ40Ma5j0p9", "blskey": "blskey"}, {"blskey": "656872ZCYXQW0XGK0d50p9", "ver": "0.5"}, {"txmMetadata": {"seqNo": "2", "txId": "1ac8ace22131RMsBewBvNgW7n4AS5FpVlytuDr9N9JebDf2t14C2PA3HsAn44K0H9aRv9EgtaYsBzF0", "client_ip": "10.0.2.15", "client_port": "9704", "node_ip": "10.0.2.15", "node_port": "9703", "services": [{"VALIDATOR_ID": "1"}, {"desc": "8ECV5K79js1RKLWtOs1ppL6pEPHxtyaStPhP5GdA", "metadata": [{"from": "EBd4BaNeTHel385GSVUWY"}, {"type": "0"}]}, {"txmMetadata": {"seqNo": "2", "txId": "1ac8ace22131RMsBewBvNgW7n4AS5FpVlytuDr9N9JebDf2t14C2PA3HsAn44K0H9aRv9EgtaYsBzF0", "ver": "1"}], --reqSignature: "", {"tx": {"data": [{"alias": "Node3", "blskey": "3F6pdBgcT5CnLy7ZeVjubhqlALFB1C5drNUh1jks3Qf10j6Qm2pbl7QzUqXdkm7swA45Drk2tBzL2GzL9nQjU8dpzup0yu6T2KHTPQbMuYX4QF0yF2zU2j3kldpgeySw34zLsJduD0NMKsC5", "blskey": "blskey"}, {"blskey": "QwDeb2K3nX80RxDcBvQ3KjRyWn7d4t801TjXBYXPhLJagXayJt0r87", "ver": "0.5"}, {"txmMetadata": {"seqNo": "3", "txId": "793573df5d7864268d0e36972420746571c51e26aefb2f574967debed4", "client_ip": "10.0.2.15", "client_port": "9706", "node_ip": "10.0.2.15", "node_port": "9705", "services": [{"VALIDATOR_ID": "1"}, {"desc": "GKFGXzFTXUyTSN7HBeB3fd4fWn1y1L2zDmOpjxya", "metadata": [{"from": "4Cu1Wm82Fa2rxJXkzPC"}, {"type": "0"}]}, {"txmMetadata": {"seqNo": "4", "txId": "data"}, {"tx": {"data": [{"alias": "Node4", "blskey": "2zN3BHM1n4rLz54JH9wSwpcZyp1hJ5sHweoLAjXvCwMh10Q1KSpWPUzDzTuvh6uEVNKAxKeVmansSwVjKjRebEdxEabykzCjyGPedyle1qjTzC7Bh1VnM2pu0yUdShpQf5K7nRxf7nAeAbunJ4tFvDkLUW2bzhsLztr7eLbV7Rw3tGwCpTz", "blskey": "blskey"}, {"blskey": "RPLaxR8XdmFwzny7n4ZwHtbyJ8LRS2U51zT2g1PQyCQhUn2Hu4ocD2BzH2kJah4fJt7C7Bh1VnM2pu0yUdShpQf5K7nRxf7nAeAbunJ4tFvDkLUW2bzhsLztr7eLbV7Rw3tGwCpTz", "client_ip": "10.0.2.15", "client_port": "9708", "node_ip": "10.0.2.15", "node_port": "9707", "services": [{"VALIDATOR_ID": "1"}, {"desc": "4PS30D3wt1cLc672061e0752820939345443b6e4f02d3b03676656f008", "metadata": [{"from": "TmTCRQR22JHm79712pW"}, {"type": "0"}]}, {"txmMetadata": {"seqNo": "4", "txId": "a9e197d7cc2601e0752820939345443b6e4f02d3b03676656f008", "ver": "1"}, {"tx": {"data": [{"alias": "Node5", "blskey": "1...n - webhook-url", "blskey": "http://10.0.2.15:8032/webhooks", "ver": "0.5"}, {"txmMetadata": {"seqNo": "5", "txId": "a9e197d7cc2601e0752820939345443b6e4f02d3b03676656f008", "ver": "1"}], --noNotifyRevocation, --monitorRevocationNotification, --tailsServerBaseUrl, "https://2F4f-159-214-205-78.ngrok.io", --autoAcceptInteractions, --autoAcceptRequests, --autoAcceptSt...]
```

Figura 6.1: Despliegue de Alice.

- **DID público:** necesario para que los agentes contacten mediante el ledger con el emisor. En la Figura 6.2 se ve como Faber publica su DID en la red, y en la Figura 6.3 la transacción generada en la red.
 - **Esquema de credencial:** este servirá para indicar los atributos que tendrá la credencial a emitir. En la Figura 6.4 se ve el esquema de credencial de Faber donde aparecen los atributos que tendrá esa credencial (nombre, fecha, grado, fecha de nacimiento). En la Figura 6.5 se observa la transacción generada en el *ledger* como consecuencia de la publicación.
 - **Definición de credencial y registro de revocación:** incluye el DID que usa el emisor, el esquema y las claves públicas utilizadas. El registro de revocación indicará donde se encuentra el servidor de revocación y que tipo de revocación se utiliza. En la Figura 6.6 se ve como Faber publica los registros de revocación (dos registros diferentes, con tamaño de 100 credenciales cada uno), y las Figuras 6.7 y 6.8 se muestran las transacciones generadas por la publicación de la definición de credencial y el registro de revocación.

```
Faber | Registering faber.agent ...
Faber |nym_info: {'did': 'grhjk6z895XGCaecFJHn', 'seed': 'd_00000000000000000000000000000000000538795', 'verkey': 'NLxUJAqy091Xe348bYJ4mEdda6FVSVLo64l_j2AXng
se'}
Faber | Registered DID: grhjk6z895XGCaecFJHn
Created public DID
```

Figura 6.2: Publicación DID de Faber.

Acme tiene un despliegue idéntico al de Faber, por lo que no se mostrarán las figuras correspondientes.

The screenshot shows a blockchain interface displaying two transactions from agent Faber. Transaction #9 is a 'Message Wrapper' with a complex JSON payload. It includes fields like 'From nym', 'Request ID', 'Digest', and a 'Transaction' section with 'Type: kyc', 'Alias: faber_agent', 'Nym: grhjvk6z895XGcaecFJHn', 'Role: ENDORSER', and a 'Verifier' key. Transaction #10 is also a 'Message Wrapper' with similar structure, including a 'Transaction' section with 'Type: ATTRIB', 'Nym: grhjvk6z895XGcaecFJHn', and an 'Attribute data' field. Both transactions have a 'Raw Data' dropdown and a 'Metadata' section.

Figura 6.3: Publicación de DID y atributos de Faber en la red.

6.1.2. Conexión de los agentes

Faber y Acme, al ser agentes que suelen emitir credenciales, crean un código QR para que los agentes que requieran una credencial se conecten a ellos (Figura 6.9). A su vez, la invitación se muestra en formato JSON. Se puede comprobar como la invitación se trata del mensaje *invitation* del protocolo *Out-of-Band*, como se explico en la Sección 4.6

Por otra parte, Alice espera los detalles de la invitación, que será lo mostrado por Faber y Acme. Una vez introducida la invitación, Alice contesta al mensaje con un inicio del protocolo DID Exchange, como ya se vio en la Sección 4.6 (Figura 6.10), que termina en los agentes finalmente conectados.

Una vez conectados los agentes, ya se pueden realizar acciones desde mandar un mensaje básico (como se muestra en la Figuras 6.11 y 6.12) de forma segura gracias a la criptografía subyacente y la conexión segura; hasta emitir una credencial, como se explica en la siguiente sección.

6.1.3. Emisión de credencial

Tras la conexión exitosa ya se pueden emitir credenciales. Utilizando la opción de *Issue credential*, Faber emite una credencial a Alice (Figura 6.13). Alice recibe la credencial y la almacena, como se observa en la Figura 6.14, donde la credencial tiene elementos como sus propios atributos, los IDs de revocación, definición de credencial, etc.

El protocolo utilizado es el descrito en la Sección 4.6.4. Primeramente

```
#3/4 Create a new schema/cred def on the ledger
Schema:
{
  "sent": {
    "schema_id": "grhjvk6z895XGCaecFJHn:2:degree schema:3.5.95",
    "schema": {
      "ver": "1.0",
      "id": "grhjvk6z895XGCaecFJHn:2:degree schema:3.5.95",
      "name": "degree schema",
      "version": "3.5.95",
      "attrNames": [
        "timestamp",
        "date",
        "name",
        "degree",
        "birthdate_dateint"
      ],
      "seqNo": 11
    }
  },
  "schema_id": "grhjvk6z895XGCaecFJHn:2:degree schema:3.5.95",
  "schema": {
    "ver": "1.0",
    "id": "grhjvk6z895XGCaecFJHn:2:degree schema:3.5.95",
    "name": "degree schema",
    "version": "3.5.95",
    "attrNames": [
      "timestamp",
      "date",
      "name",
      "degree",
      "birthdate_dateint"
    ],
    "seqNo": 11
  }
}
Schema ID: grhjvk6z895XGCaecFJHn:2:degree schema:3.5.95
```

Figura 6.4: Publicación de esquema de credencial de Faber.

Faber envía un mensaje *offer-credential* a Alice ofreciéndole la emisión de una credencial. Alice contesta a Faber con un *request-credential*, a lo que le sigue la emisión de la credencial con el mensaje *issue-credential* (proceso visible en Figuras 6.14 y 6.13).



Figura 6.5: Esquema de credencial de Faber en la red.

```

Schema ID: grhjvk6z895XGaecfJHn:2:degree_schema:3.5.95
Faber   | Revocation registry: (undetermined) state: int
Faber   | Revocation registry: grhjvk6z895XGaecfJHn:3:CL:11:faber.agent.degree_schema:CL_ACCUM:b0ef5e9e-6853-41f8-b629-8d3c6
95bac14 state: generated
Faber   | Revocation registry: grhjvk6z895XGaecfJHn:4:grhjvk6z895XGaecfJHn:3:CL:11:faber.agent.degree_schema:CL_ACCUM:b0ef5e9e-6853-41f8-b629-8d3c6
95bac14 state: posted
Faber   | Revocation registry: grhjvk6z895XGaecfJHn:4:grhjvk6z895XGaecfJHn:3:CL:11:faber.agent.degree_schema:CL_ACCUM:b0ef5e9e-6853-41f8-b629-8d3c6
95bac14 state: active
Faber   | Revocation registry: (undetermined) state: int
Faber   | Revocation registry: grhjvk6z895XGaecfJHn:4:grhjvk6z895XGaecfJHn:3:CL:11:faber.agent.degree_schema:CL_ACCUM:5d2e4bda-8654-4b8b-b503-09d40
7f028cd state: generated
Faber   | Revocation registry: grhjvk6z895XGaecfJHn:4:grhjvk6z895XGaecfJHn:3:CL:11:faber.agent.degree_schema:CL_ACCUM:5d2e4bda-8654-4b8b-b503-09d40
7f028cd state: posted
Faber   | Revocation registry: grhjvk6z895XGaecfJHn:4:grhjvk6z895XGaecfJHn:3:CL:11:faber.agent.degree_schema:CL_ACCUM:5d2e4bda-8654-4b8b-b503-09d40
7f028cd state: active
Cred def ID: grhjvk6z895XGaecfJHn:3:CL:11:faber.agent.degree_schema
Publish schema/cred def duration: 27.21s
  
```

Figura 6.6: Publicación de definición de credencial y registro de revocación de Faber.

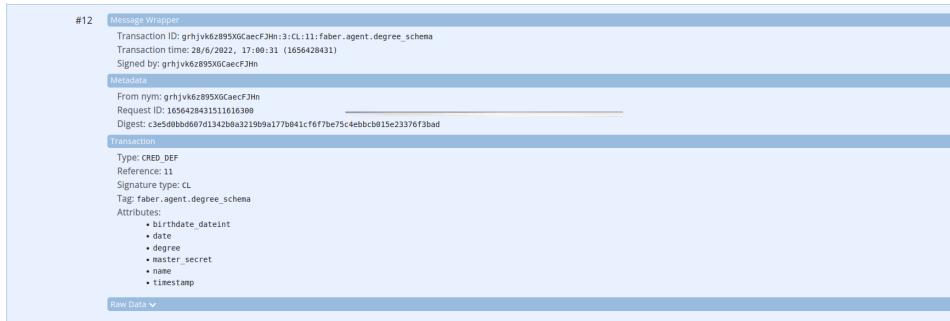


Figura 6.7: Definición de credencial de Faber en la red.

6.1.4. Presentación de credencial

Una vez guardada la credencial en su *wallet*, Alice puede utilizarla para otras comunicaciones. Para demostrar como presentar una prueba de credencial, se conecta el agente Alice al agente Acme, de igual manera que se hizo con Faber.

Acme pedirá una presentación de prueba de credencial con la opción *Send Proof Request* (Figura 6.15). Esto le mandará una solicitud de prueba de credencial a Alice, que contestará con la prueba en sí (Figura 6.16).

```
#10  Message: Revocation
Transaction ID: 4odsyj0KY1j|FxrTwD0K:4-4odsyj0KY1j|FxrTwD0K:3:CL:8:faber.agent.degree_schema:CL.ACUM:df303a3c-14ef-4c39-b2c2-9699e7e5f97c
Transaction time: 27/6/2022, 17:50:37 (1655345493)
Signed by: 4odsyj0KY1j|FxrTwD0K
Metadata
From mym: 4odsyj0KY1j|FxrTwD0K
Request ID: 16553454934671181098
Digest: a3171968161c9306c93630807bf723ce8948e18ee0e3a1a0554060d179f1
File
Type: REVOC_REG_DEF
Revocation registry type: CL_ACUM
Credential definition ID: 4odsyj0KY1j|FxrTwD0K:3:CL:8:faber.agent.degree_schema
Tag: df303a3c-14ef-4c39-b2c2-9699e7e5f97c
Issuance type: BY_CREDENTIAL
Maximum credential count: 100
Maximum credential age: 1000
Tails file hash: SwYkA49Nj0Q9W7cesAvvE8r4xR201xz3xyj7cx8
Tails file location: https://95fb-159-214-205-61.ngrok.io/4odsyj0KY1j|FxrTwD0K:4-4odsyj0KY1j|FxrTwD0K:3:CL:8:faber.agent.degree_schema:CL.ACUM:df303a3c-14ef-4c39-b2c2-9699e7e5f97c
Accumulator root key: 1
1. ECDCF79194E5A8774AE5D0B005C59052C901F9941DE1E3A5414C47310F 1 0327B0CA0B4B0507BAC747EDC284FA7C998AB038999CMA967D07740F 1 40031B8B30715F01E2EBF934AAC4252656404060F08322379B24F413325 1
082FFBF09F57239462837EMAC8E2113390886EAD094264333 1 08A0973E8363377120317377881235484475946F5008812998114687F58P9CF 1 08A0988MF944185791754F48999C998F1288842998119650111940ECE3A3805329 1
0524F9EC4CA7E149529548EAD0328404048A34250983CA392400909MC9780 1 129845252053E28466F458290C9F518E2329851591014024818187FC425 1 15CKA4MF988215798E84545360006040488637799581D4040835C5C789970F 1
23AA81188BA2ABD70685ACD2BE18CBB86604F35C29B2002248E37E5C4001 1 0A9129C084BD32C43732CEABE9A949FC262584658C2FA8625E4C94941E4237 1 05C8E3B899C086C7878BC1668523A14AC5988BCCC8A198950404C4CC3AAE81A7
```

Figura 6.8: Registro de revocación de Faber en la red.



Figura 6.9: Código QR para conexión Faber.

```
#9 Input faber.py invitation details
Invitation details: {"url": "https://didcomm.org/out-of-band/1.0/invitation", "id": "bcf4b6bc-b8a7-4580-aec2-6a412cd76207", "handshake_protocols": ["https://didcomm.org/didexchange/1.0"], "label": "faber.agent", "services": [{"id": "#inline", "type": "did-communication", "recipientKeys": ["did:key:z6MktmZDDXgPfwatrVqWZ7nYnNjzCeazGbVxEpNTd7fq"]}], "serviceEndpoint": "http://10.0.2.15:8020"}]
Invitation response:
{
  "invitation_mode": "once",
  "connection_protocol": "didexchange/1.0",
  "connection_id": "4b7300c2-2a47-425b-bcca-9c2dc1355190",
  "invitation_id": "f133c59LqnpU59GtYjQzjnVj1LqgnuudJc26VcCT4H",
  "my_did": "bf71b770-62cb-4a9a-8593-b9290c62cf14",
  "my_did_state": "request-sent",
  "status": "requested",
  "updated_at": "2022-06-28T15:06:41.149160Z",
  "their_role": "Inviter",
  "their_label": "faber.agent",
  "routing_state": "none",
  "accept": "auto",
  "invitation_msg_id": "bcf4b6bc-b8a7-4580-aec2-6a412cd76207",
  "created_at": "2022-06-28T15:06:41.059846Z",
  "my_did": "A21hoXh6El3mf0lqglukcb"
}

Connect duration: 0.12s
Waiting for connection...
Alice | Connected
Alice | Check for endorser role ...
Connect duration: 0.37s
(3) Send Message
(3) Send Message
(4) Input New Invitation
```

Figura 6.10: Respuesta de Alice al mensaje *invitation*.

El protocolo utilizado es el descrito en la Sección 4.6.5. En primer lugar, Acme manda un mensaje *request-presentation* para solicitar la prueba a Alice. Alice contesta con el mensaje *presentation*, que incluye la prueba,

```
(3) Send Message
(3) Send Message
(4) Input New Invitation
(X) Exit?

[3/4/X] 3

Enter message: Hola esto es una prueba

Alice | Received message: faber.agent received your message
```

Figura 6.11: Mensaje de Alice a Faber.

Faber	Check for endorser role ...
Faber	Received message: Hola esto es una prueba
	(4) Endorse

Figura 6.12: Recepción de mensaje de Faber.

```
#13 Issue credential offer to X
Faber | Credential: state = offer-sent, cred_ex_id = c32f9647-5250-4b4a-bb62-6889ee6ed651
Faber | Credential: state = request-received, cred_ex_id = c32f9647-5250-4b4a-bb62-6889ee6ed651

#17 Issue credential to X
Faber | Revocation registry ID: grhjvk6z895XGCaecFJHn:4:grhjvk6z895XGCaecFJHn:3:CL:11:faber.agent.degree_schema:CL_AC
CUM:b0ef5e9e-6853-41f8-b629-bd3c695bac14
Faber | Credential revocation ID: 1
Faber | Credential: state = credential-issued, cred_ex_id = c32f9647-5250-4b4a-bb62-6889ee6ed651
Faber | Credential: state = done, cred_ex_id = c32f9647-5250-4b4a-bb62-6889ee6ed651
```

Figura 6.13: Emisión de credencial de Faber a Alice.

```
Alice | Credential: state = offer-received, cred_ex_id = 1a1f3da7-13b7-4f25-8980-14334caa9abb

#15 After receiving credential offer, send credential request
Alice | Credential: state = request-sent, cred_ex_id = 1a1f3da7-13b7-4f25-8980-14334caa9abb
Alice | Credential: state = credential-received, cred_ex_id = 1a1f3da7-13b7-4f25-8980-14334caa9abb

#18.1 Stored credential 8059d20b-ce38-4bb8-ae46-37eae013fc23 in wallet
Credential details:
{
  "referent": "8059d20b-ce38-4bb8-ae46-37eae013fc23",
  "stx": [
    {
      "date": "2018-05-28",
      "name": "Alice Smith",
      "timestamp": "1656429974",
      "birthdate_datetime": "19980628",
      "degree": "Maths"
    }
  ],
  "schema_id": "grhjvk6z895XGCaecFJHn:2:degree_schema:3.5.95",
  "cred_def_id": "grhjvk6z895XGCaecFJHn:3:CL:11:faber.agent.degree_schema",
  "rev_req_id": "grhjvk6z895XGCaecFJHn:4:grhjvk6z895XGCaecFJHn:3:CL:11:faber.agent.degree_schema:CL_ACCUM:b0ef5e9e-6853-41f8-b629-bd3c695bac14",
  "cred_rev_id": "1"
}

Alice | credential_id 8059d20b-ce38-4bb8-ae46-37eae013fc23
Alice | cred_def_id grhjvk6z895XGCaecFJHn:3:CL:11:faber.agent.degree_schema
Alice | schema_id grhjvk6z895XGCaecFJHn:2:degree_schema:3.5.95
Alice | Credential: state = done, cred_ex_id = 1a1f3da7-13b7-4f25-8980-14334caa9abb
```

Figura 6.14: Recepción de credencial de Alice emitida por Faber.

```
(1) Issue Credential
(2) Send Proof Request
(3) Send Message
(X) Exit?
[1/2/3/X]

#20 Request proof of degree from alice
Acme | Presentation: state = request-sent, pres_ex_id = 85eef09b-a37b-4f5c-afa5-c71f60876c4a
Acme | Presentation: state = presentation-received, pres_ex_id = 85eef09b-a37b-4f5c-afa5-c71f60876c4a

#27 Process the proof provided by X

#28 Check if proof is valid
Acme | Presentation: state = done, pres_ex_id = 85eef09b-a37b-4f5c-afa5-c71f60876c4a
Acme | Proof = true
```

Figura 6.15: Solicitud de Presentación de Credencial de Acme a Alice.

que Acme podrá verificar comprobando si el acumulador coincide con el pu-

```
Alice      | Presentation: state = request-received, pres_ex_id = da18f0ce-6621-4d4f-9ff0-221b90521c6c
#24 Query for credentials in the wallet that satisfy the proof request
#25 Generate the proof
#26 Send the proof to X: {"indy": {"requested_predicates": {"_0_birthdate_dateint_GE_uuid": {"cred_id": "8059d20b-ce38-4bb8-ae46-37eae013fc23"}}, "requested_attributes": {"_0_name_uuid": {"cred_id": "8059d20b-ce38-4bb8-ae46-37eae013fc23", "revealed": true}, "0_degree_uuid": {"cred_id": "8059d20b-ce38-4bb8-ae46-37eae013fc23", "revealed": true}, "self_attested_attributes": {}}}
Alice      | Presentation: state = presentation-sent, pres_ex_id = da18f0ce-6621-4d4f-9ff0-221b90521c6c
Alice      | Presentation: state = done, pres_ex_id = da18f0ce-6621-4d4f-9ff0-221b90521c6c
```

Figura 6.16: Prueba de presentación de credencial de Alice a Acme.

blicado en el registro de revocación del *ledger*.

Tras haber verificado la credencial, el agente Acme puede estar seguro de que Alice tiene la credencial emitida por Faber, por lo que puede decidir emitir una nueva credencial a Alice como se hizo en la Sección 6.1.3 (Figuras 6.17 y 6.18).

```
(1) Issue Credential
(2) Send Proof Request
(3) Send Message
(X) Exit?
[1/2/3/X]1

#13 Issue credential offer to X
Acme      | Credential: state = offer-sent, cred_ex_id = 4f70ea6f-07eb-4443-85ce-e2330bedfef1
Acme      | Credential: state = request-received, cred_ex_id = 4f70ea6f-07eb-4443-85ce-e2330bedfef1

#17 Issue credential to X
Acme      | Revocation registry ID: PK6oT5bqMGobFdNhSrRqkly:4:PK6oT5bqMGobFdNhSrRqkly:3:CL:8:acme.agent.employee_id_schema:CL_ACCUM:49ea4ae4-ea63-45d2-9d2c-f56037e4a96e
Acme      | Credential revocation ID: 1
Acme      | Credential: state = credential-issued, cred_ex_id = 4f70ea6f-07eb-4443-85ce-e2330bedfef1
Acme      | Credential: state = done, cred_ex_id = 4f70ea6f-07eb-4443-85ce-e2330bedfef1
```

Figura 6.17: Emisión de credencial de Acme a Alice.

```
Alice      | Credential: state = offer-received, cred_ex_id = e1439012-ec69-46ba-819b-c9518714898d
#15 After receiving credential offer, send credential request
Alice      | Credential: state = request-sent, cred_ex_id = e1439012-ec69-46ba-819b-c9518714898d
Alice      | Credential: state = credential-received, cred_ex_id = e1439012-ec69-46ba-819b-c9518714898d

#18.1 Stored credential 3f7e09fe-b4fc-43b6-aff0-723b5138e1cb in wallet
Credential details:
{
  "referent": "3f7e09fe-b4fc-43b6-aff0-723b5138e1cb",
  "attrs": [
    {"name": "Alice Smith",
     "date": "2022-05-28",
     "position": "CEO",
     "employee_id": "ACME009"
    },
    {"schema_id": "PK6oT5bqMGobFdNhSrRqkly:2:employee_id_schema:77.63.52",
     "cred_def_id": "PK6oT5bqMGobFdNhSrRqkly:3:CL:8:acme.agent.employee_id_schema",
     "rev_reg_id": "PK6oT5bqMGobFdNhSrRqkly:4:PK6oT5bqMGobFdNhSrRqkly:3:CL:8:acme.agent.employee_id_schema:CL_ACCUM:49ea4ae4-ea63-45d2-9d2c-f56037e4a96e",
     "cred_rev_id": "1"
    }
]
Alice      | credential_id 3f7e09fe-b4fc-43b6-aff0-723b5138e1cb
Alice      | cred_def_id PK6oT5bqMGobFdNhSrRqkly:3:CL:8:acme.agent.employee_id_schema
Alice      | schema_id PK6oT5bqMGobFdNhSrRqkly:2:employee_id_schema:77.63.52
Alice      | Credential: state = done, cred_ex_id = e1439012-ec69-46ba-819b-c9518714898d
```

Figura 6.18: Recepción de credencial de Alice emitida por Acme.

6.1.5. Revocación de credencial

Aunque Alice tenga dos credenciales en su *wallet*, existe la posibilidad de que alguna de ellas sea revocada, ya sea por motivos de maluso o por

actualizaciones de la credencial.

Para revocar la credencial, Faber utiliza la opción *Revoke Credential*. Para poder revocar la credencial es necesario el ID del registro de revocación y el ID de la credencial. Una vez revocada la credencial, se da la opción de publicar ya la revocación o más tarde, pero se elige revocarla ahora (Figura 6.19). Consecuentemente, publicar la revocación significa actualizar el acumulador criptográfico y publicarlo en el *ledger*. La transacción asociada se muestra en la Figura 6.20.

```
(1) Issue Credential
(2) Send Proof Request
(2a) Send *Connectionless* Proof Request (requires a Mobile client)
(3) Send Message
(4) Create New Invitation
(5) Revoke Credential
(6) Publish Revocations
(7) Toggle tracing on credential/proof exchange
(X) Exit?
[1/z/3/4/5/6/T/X] 5
Enter revocation registry ID: grhvjk6z895XGCaecFJHn:4:grhvjk6z895XGCaecFJHn:3:CL:11:faber.agent.degree_schema:CL_ACCUM:b0e
f5e9e-6853-41f8-b629-8d3c695bac14

Enter credential revocation ID: 1
Publish now? [Y/N]: Y
Faber | Credential: state = credential-revoked, cred_ex_id = c32f9647-5250-4b4a-bb62-6889ee6ed651
```

Figura 6.19: Revocación de credencial de Faber.

```
#22 Message Wrapper
Transaction ID: 40d4sy0KY1|FxrTwIDNk:4:40d4sy0KY1|FxrTwIDNk:3:CL:8:faber.agent.degree_schema:CL_ACCUM:d3b3a3c-14ef-4c39-b2c2-9699e7e5f97c
Transaction time: 27/6/2022, 18:04:31 (1656345871)
Signed by: 40d4sy0KY1|FxrTwIDNk

Metadata
From nym: 40d4sy0KY1|FxrTwIDNk
Request ID: 16054587923559008
Digest: 401ba156212a31a439c6d0b0073c1999a1c42e0e2f104b8a257246591ad7

Type: REVOKE_REG_ENTRY
Revocation registry type: CL_ACCUM
Revocation registry ID:
40d4sy0KY1|FxrTwIDNk:4:40d4sy0KY1|FxrTwIDNk:3:CL:8:faber.agent.degree_schema:CL_ACCUM:d3b3a3c-14ef-4c39-b2c2-9699e7e5f97c
Accumulator digest: 21:1373196139925138E3B04D9788E7C72F57956EAAFA1AEE260BF3742B9F3 21:12976258C7393C681F21E2B2872E2C82C659749C9D59E3AC34C2B0B707F35C8B4TF 6:596ACB866977280CA297FC896262992283A848ED4701225F4E7226229565455 4:1E9585C330801C98E8454AE3886CDC7A6581819A1A2CE207E3BACCE4897237 6:5F7440669088E6809668025845FF3F3F24BFF764F090F502C0CA09E9F198FA4 4:6A5264EA1BC3A669914F735CB4B08411085178196EFA1C65C015873276013135
Raw Data: w
```

Figura 6.20: Revocación de credencial de Faber en la red.

Si ahora Faber le manda una solicitud de presentación de prueba de credencial como se hizo en la Sección 6.1.4, la prueba será verificada como falsa (Figura 6.21).

```
#20 Request proof of degree from alice
Faber | Presentation: state = request-sent, pres_ex_id = ced558fc-d00c-4fcc-870c-355b9ffebd44
Faber | Presentation: state = presentation-received, pres_ex_id = ced558fc-d00c-4fcc-870c-355b9ffebd44
#27 Process the proof provided by X
#28 Check if proof is valid
Faber | Presentation: state = done, pres_ex_id = ced558fc-d00c-4fcc-870c-355b9ffebd44
Faber | Proof = False
```

Figura 6.21: Presentación de credencial fallida de Alice a Faber.

Finalmente, se pueden ver todas la entradas del servidor de revocación en la Figura 6.22, tanto de Acme como de Faber.

```
tails-server_1 | ====== Running on http://0.0.0.0:6543 ======
tails-server_1 | (Press CTRL+C to quit)
ngrok-tails-server_1 | t=2022-06-28T15:00:38+0000 lvl=info msg="join connections" obj=join id=a355a90bee1b l=172.19.0.3:6543 r=150.214.205.78:3312
tails-server_1 | 2022-06-28 15:00:38,655 aiohttp.access INFO 172.19.0.2 [28/Jun/2022:15:00:38 +0000] "PUT /grhjvkz895XGCaecfJH:n:4;grhjvk62895XCC
aeCFJH:n:3;CL:11:faber.agent.degree.schema:CL_ACCUM:b0ef5e9e-6853-41f8-bc29-8d3c695ac14 HTTP/1.1" 200 195 "-" "Python/3.6 aiohttp/3.8.1"
ngrok-tails-server_1 | t=2022-06-28T15:00:41+0000 lvl=info msg="join connections" obj=join id=f1326a0c5711 l=172.19.0.3:6543 r=150.214.205.78:17517
tails-server_1 | 2022-06-28 15:00:41,432 aiohttp.access INFO 172.19.0.2 [28/Jun/2022:15:00:41 +0000] "PUT /PK6oT5bqMGobfDnNsRqQy:4;PK6oT5bqMGobf
dNhSrRqQy:3;CL:8:acme.agent.employee_id.schema:CL_ACCUM:49ea4ee4-ea63-452d-9d2c-f56037e4a96e HTTP/1.1" 200 195 "-" "Python/3.6 aiohttp/3.8.1"
tails-server_1 | 2022-06-28 15:00:44,326 aiohttp.access INFO 172.19.0.2 [28/Jun/2022:15:00:44 +0000] "PUT /grhjvkz895XGCaecfJH:n:4;grhjvk62895XCC
aeCFJH:n:3;CL:11:faber.agent.degree.schema:CL_ACCUM:5d2e4bda-8654-4b8b-b593-09d407f028c6 HTTP/1.1" 200 195 "-" "Python/3.6 aiohttp/3.8.1"
tails-server_1 | 2022-06-28 15:00:47,512 aiohttp.access INFO 172.19.0.2 [28/Jun/2022:15:00:47 +0000] "PUT /PK6oT5bqMGobfDnNsRqQy:4;PK6oT5bqMGobf
dNhSrRqQy:3;CL:8:acme.agent.employee_id.schema:CL_ACCUM:3fb008c5-710b-46f5-8d81-90dd380d1fa0 HTTP/1.1" 200 195 "-" "Python/3.6 aiohttp/3.8.1"
```

Figura 6.22: Entradas del servidor de revocación.

6.2. Prueba de evaluación de rendimiento

Por último, se van a ejecutar los escenarios explicados en la Sección 4.8. Para analizar los resultados se utilizará el programa Excel. Se mostrarán tablas con los tiempos así como diagramas de cajas y bigotes, debido a su potencial para comparar datos en una sola gráfica. Este tipo de diagrama se puede ver en la Figura 6.23.

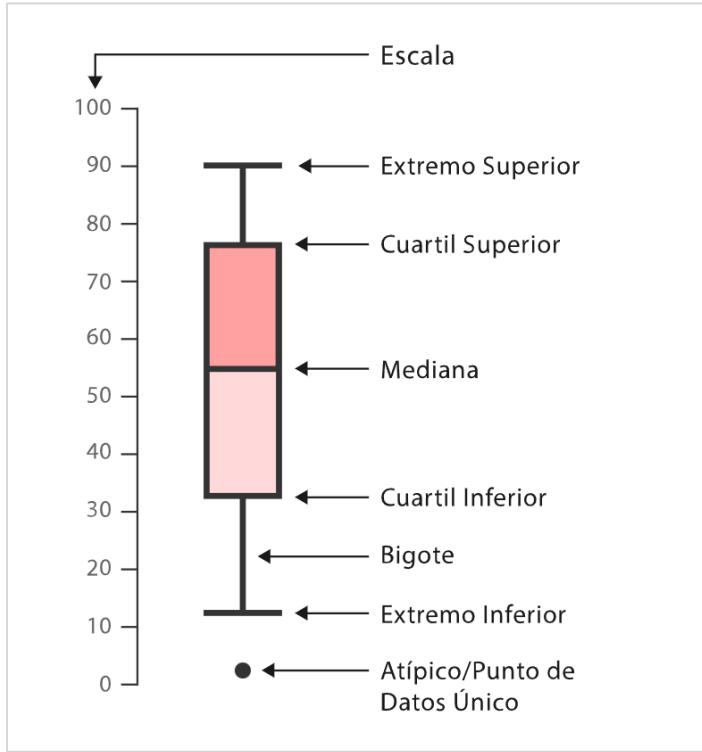


Figura 6.23: Anatomía diagrama de cajas y bigotes [13].

En este tipo de diagrama se muestran visualmente los grupos de datos

numéricos a través de sus cuartiles. Las líneas que se extienden paralelas a las cajas se conocen como ‘bigotes’, y se usan para indicar la variabilidad fuera de los cuartiles superior e inferior. Los valores atípicos se representan como puntos individuales.

6.2.1. Emisión y presentación de credenciales simples

En esta prueba se ejecutan 25 pruebas con 10 números diferentes de credenciales. Se utiliza el script *proof_presentation_CS.py* para automatizar las pruebas y el script *plot_CS.py* para analizar los resultados, filtrarlos, pasarlos a tablas Excel y mostrarlos.

En la Figura 6.24 se muestra una tabla con cada uno de los tiempos tras hacer la media de todas las pruebas. Por otro lado, en la Figura 6.25 se muestran las gráficas de cada uno de los tiempos según el número de credenciales emitidas y presentadas.

Credenciales	Startup	Connect	Publish	Average cred	Average proof	Total
10	8.1352	0.2768	10.0328	0.272	0.3632	26.42
20	8.0076	0.2724	13.2964	0.2424	0.318	34.404
50	8.0372	0.2796	15.01	0.2192	0.2948	50.58
100	8.2016	0.276	10.968	0.224	0.2896	72.3608
150	8.6296	0.2944	11.1728	0.2336	0.2964	101.1196
200	8.3596	0.2744	11.72167	0.221666667	0.302916667	127.3967
250	8.3116	0.2768	10.3108	0.2244	0.3168	155.8552
300	8.0328	0.2708	10.9128	0.2264	0.3308	188.4612
400	8.0396	0.2708	10.9668	0.2272	0.3564	254.7288
500	8.0108	0.2824	11.8088	0.2296	0.3864	329.7612

Figura 6.24: Tiempos para escenario de credenciales simples.

Se puede observar como los tiempos de despliegue, conexión y publicación no se hace mayor aunque aumente el número de credenciales. Esto es debido a que estos tiempos son independientes a las credenciales que se van a emitir. Antes de emitir las credenciales (seas cuales sean), el agente debe desplegarse, publicar (si es el caso) sus atributos y (si es emisor de credenciales) los esquemas y definiciones de credencial necesarios.

Por otro lado, el tiempo medio por emisión de credencial y presentación tampoco varía demasiado aunque aumenten las credenciales. La pequeña variación/aumento se debe a una posible falta de hebras, que podrás solucionarse con un equipamiento más potente.

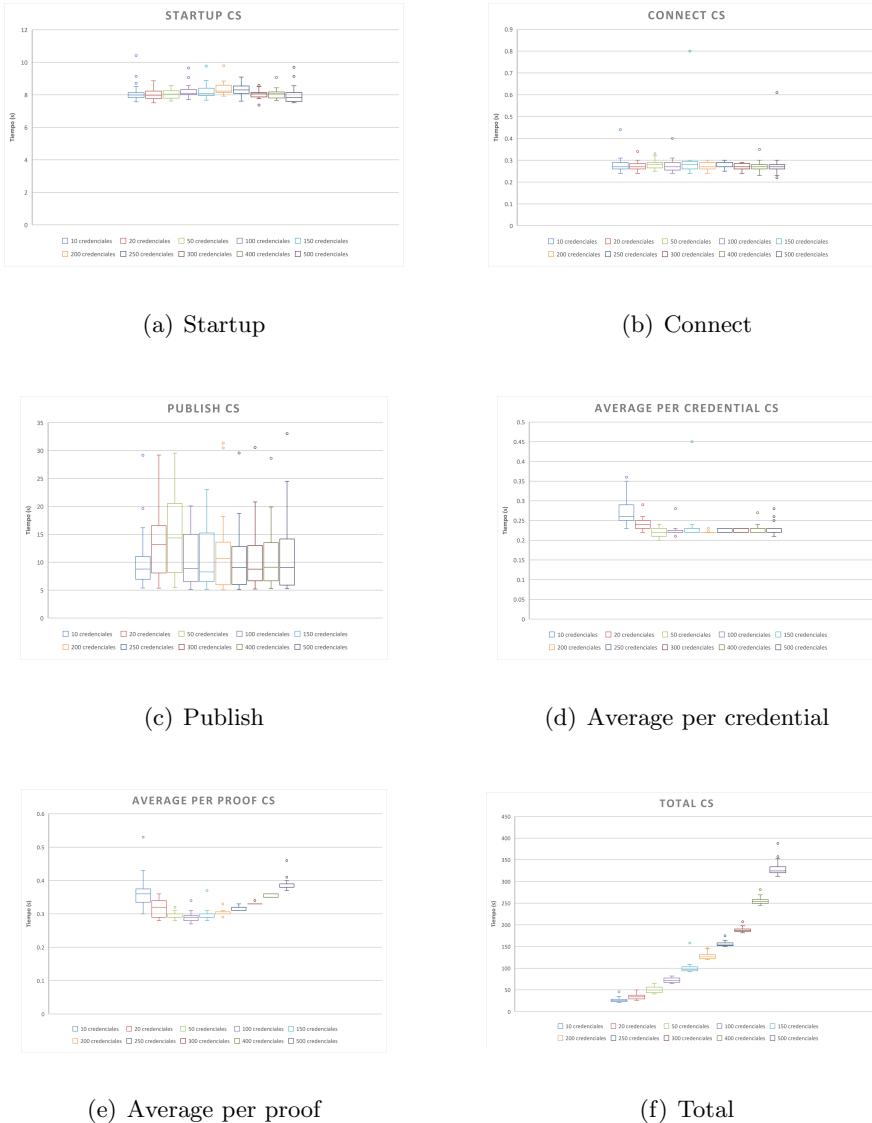


Figura 6.25: Medidas de tiempo para escenario de credenciales simples.

6.2.2. Emisión y presentación de credenciales revocables sin ser revocadas

En esta prueba se ejecutan 25 pruebas con 10 números diferentes de credenciales. Se utiliza el script *proof_presentation_CR_no_revocation.py* para automatizar las pruebas y el script *plot_CR_no_revocation.py* para analizar los resultados, filtrarlos, pasarlos a tablas Excel y mostrarlos.

En la Figura 6.26 se muestra una tabla con cada uno de los tiempos tras hacer la media de todas las pruebas. Por otro lado, en la Figura 6.27 se muestran las gráficas de cada uno de los tiempos según el número de credenciales emitidas y presentadas.

Credenciales	Startup	Connect	Publish	Average cred	Average proof	Total
10	7.8712	0.2788	24.06609	0.320869565	0.455652174	41.6387
20	8.0976	0.2776	25.12292	0.279583333	0.43875	49.54667
50	7.9984	0.2812	23.87833	0.266666667	0.415416667	68.14917
100	7.984	0.2708	24.6276	0.2708	0.3944	101.0616
150	7.9132	0.2756	23.545	0.244166667	0.408333333	131.42
200	7.9472	0.2796	24.22542	0.260833333	0.43625	173.6258
250	8.075833	0.2766667	25.57625	0.249583333	0.456666667	212.4121
300	8.242273	0.2786364	21.98045	0.266818182	0.486818182	258.1314
400	7.9304	0.2764	24.59042	0.262083333	0.517727273	346.1468
500	8.0932	0.2804	24.4376	0.2644	0.56	445.1869

Figura 6.26: Tiempos para escenario de credenciales revocables sin llegar a ser revocadas.

Los tiempos de despliegue y conexión son similares al escenario de credenciales simples. Sin embargo, el tiempo de publicación llega a casi duplicarse. Esto es debido a que, al ser credenciales revocables, Faber tiene que publicar los registros de revocación correspondientes para indicar como funciona la revocación, el acumulador, y la localización del servidor de revocación.

Por otro lado, el tiempo medio de emisión de credencial aumenta un poco debido a que Faber al emitir la credencial deberá incluir nuevos elementos (ID de la credencial en el servidor de revocación y el testigo para calcular el acumulador y poder probar la no revocación con éxito).

Finalmente, el tiempo medio de presentación de prueba de credencial aumenta de aproximadamente un 50 % respecto de las credenciales simples. Esto se debe, como se puede intuir, al hecho de que las credenciales sean revocables. El titular deberá presentar la prueba de no revocación, alargando el proceso de prueba de credencial.

6.2.3. Emisión y presentación de credenciales revocables revocadas y publicadas una a una

En esta prueba se ejecutan 25 pruebas con 4 números diferentes de credenciales. Se han utilizado menos credenciales debido al enorme gasto computacional que tiene la simulación, como se podrá ver en los tiempos siguientes. Se utiliza el script *proof_presentation_CR_one_by_one_revocation.py*



Figura 6.27: Medidas de tiempo para escenario de credenciales revocables sin ser revocadas.

para automatizar las pruebas y el script *plot_CR_one_by_one_revocation.py* para analizar los resultados, filtrarlos, pasarlos a tablas Excel y mostrarlos.

En la Figura 6.28 se muestra una tabla con cada uno de los tiempos tras hacer la media de todas las pruebas. Por otro lado, en la Figura 6.29 se muestran las gráficas de cada uno de los tiempos según el número de

credenciales emitidas y revocadas.

Credenciales	Startup	Connect	Publish	Average cred	Revocation	Total	Average revocation
10	7.9624	0.2828	22.91417	0.322173913	27.55695652	63.45	2.755695652
20	7.986	0.274	25.1976	0.279565217	57.77	98.71826	2.8885
50	7.9688	0.278	26.1304	0.26	152.2734783	201.1622	3.045469565
100	7.694	0.2748	23.53042	0.268823529	309.2288235	369.3118	3.092288235

Figura 6.28: Tiempos para escenario de credenciales revocables revocando una a una.

Los tiempos de despliegue, conexión y publicación son similares al escenario de credenciales revocables sin ser revocadas por los motivos ya explicados anteriormente. Por otra parte, se observa que el tiempo medio por revocación de una credencial es de en torno a 3s, un tiempo muy alto si se habla de un número alto de credenciales. Esto es debido a que se está publicando la revocación una a una, cada vez que se revoca una credencial. Esto implica, como se ha explicado anteriormente, que se publique en el *ledger* el nuevo valor del acumulador, lo que resulta en ese tiempo medio tan alto de revocación por credencial.

6.2.4. Emisión y presentación de credenciales revocables revocadas y publicadas juntas al final

En esta prueba se ejecutan 25 pruebas con 4 números diferentes de credenciales. Se utiliza el script *proof_presentation_CR_all_at_once_revocation.py* para automatizar las pruebas y el script *plot_CR_all_at_once_revocation.py* para analizar los resultados, filtrarlos, pasarlo a tablas Excel y mostrarlos.

En la Figura 6.30 se muestra una tabla con cada uno de los tiempos tras hacer la media de todas las pruebas. Por otro lado, en la Figura 6.31 se muestran las gráficas de cada uno de los tiempos según el número de credenciales emitidas y revocadas.

Los tiempos de despliegue, conexión y publicación son similares como ya se ha visto en los apartados anteriores. Sin embargo, el tiempo medio de revocación de una credencial ha disminuido enormemente (de 3s a 0.1s). Esto es debido, como se puede intuir, a que solo se publica una transacción en el *ledger*, al final de todas las revocaciones.



Figura 6.29: Medidas de tiempo para escenario de credenciales revocables revocadas y publicadas una a una.

6.2.5. Conclusiones

Tras las pruebas de los distintos escenarios se puede llegar a las siguientes conclusiones:

- El tiempo que tardan los agentes en desplegarse y conectarse entre ellos no depende del número de credenciales.

Credenciales	Startup	Connect	Publish	Average cred	Revocation	Total	Average revocation
10	7.8376	0.2724	23.234	0.3232	1.0712	37.2584	0.10712
20	7.8512	0.2816	22.1524	0.2824	1.75	39.3116	0.0875
50	7.8452	0.2768	27.09174	0.264347826	4.316956522	54.73348	0.08633913
100	8.3348	0.284	21.7336	0.276	9.0924	68.7188	0.090924

Figura 6.30: Tiempos para escenario de credenciales revocables revocadas y publicadas juntas al final.

- Observando los diagramas de cajas y bigotes de los tiempos de publicación, se observa que tiene una alta variabilidad. Esto es debido a que el tiempo de publicación depende del tráfico que haya ese momento en el *ledger*. Es por eso que a veces los tiempos de publicación son menos para un mayor número de credenciales, y también es debido a que se han hecho 25 pruebas, con más pruebas los tiempos serían más similares.
- La revocación es una herramienta muy útil en el esquema de credenciales. Sin embargo, su uso empeora el rendimiento del entorno. El tiempo de emisión de credenciales es prácticamente el mismo, pero el de presentación de prueba de credencial aumenta un 50 %. Aun así, el precio a pagar es justo por las múltiples ventajas que ofrece la revocación.
- Publicar las revocaciones una a una es inviable. Los tiempos son enormes y si se trata de un número grande de credenciales el rendimiento es pobre. Por otra parte, tardar mucho en publicar una credencial puede dar problemas al emisor, ya que el titular puede hacer un uso indebido de la credencial cuando debería estar revocada. Esto se puede solucionar, por ejemplo, con un número mínimo de credenciales revocadas antes de publicarlo en el *ledger*. Otra opción más sencilla sería publicar las revocaciones X veces al día a unas horas específicas.



Figura 6.31: Medidas de tiempo para escenario de credenciales revocables revocadas juntas al final.

Capítulo 7

Conclusiones

En este capítulo final se pretende llegar a conclusiones en base a los resultados obtenidos, así como valorar posibles trabajos futuros y la influencia del proyecto.

7.1. Resultados y trabajo realizado

Tras terminar el proyecto, se ha conseguido implementar el entorno SSI basado en *blockchain*, que era el principal objetivo del proyecto:

- Se estudiaron las tecnologías necesarias (*blockchain*, Hyperledger Aries, Hyperledger Indy) así como las herramientas utilizadas (Docker, Virtual Box, Linux, Visual Code Studio).
- Se ha realizado una formación e investigación exhaustiva sobre los conocimientos necesarios para el proyecto. Entre ellos, formación en el lenguaje Python o en cursos sobre Hyperledger, *blockchain* y SSI.
- Una vez preparado para poder empezar el proyecto, se explicaron, discutieron y compararon tres soluciones posibles para implementar el entorno SSI deseado. Finalmente se escoge Hyperledger por sus múltiples ventajas y su gran comunidad.
- Se diseñó un entorno SSI con Hyperledger Aries e Hyperledger Indy capaz de probar las funcionalidades que confirman los principios de SSI. Se diseñaron dos subentornos diferentes: uno de evaluación de funcionalidades y otro de evaluación de rendimiento.
- Se desarrollaron e implementaron los dos subentornos diseñados utilizando las tecnologías estudiadas, y se publicaron en el repositorio correspondiente.

- Se llevaron a cabo las pruebas diseñadas y se sacaron diferentes conclusiones sobre las funcionalidades y el rendimiento del proyecto.

7.2. Trabajos futuros

Es cierto que el proyecto ha intentado profundizar lo máximo posible en el concepto de identidad auto-soberana y en el proyecto Hyperleger. Sin embargo, las posibilidades son prácticamente infinitas. Hay algunas características sobre los agentes que o no se han mencionado o no se ha profundizado sobre ellas:

- **Multi-ledger:** existe la posibilidad de que un agente necesite acceder a más de un *ledger*, debido a que hay múltiples instancias en el mundo. Es por ello que los agentes necesitan un mecanismo para poder estar conectados a más de un *ledger*. Este proceso está actualmente en desarrollo, y se implementará en los agentes (*frameworks*), por lo que el desarrollador del controlador no deberá preocuparse.
- **Multi-tenancy:** esta característica permite que un agente tenga varias *wallets* en una sola instancia de ACA-Py. Esto permite que se pueda usar el agente para más casos de uso. Véase el ejemplo de una empresa que crea una *wallet* para cada departamento.
- **Mediación:** la utilización de agentes intermediarios para facilitar el flujo de mensajes entre otros agentes. Véase el caso de agentes móviles donde se necesita este tipo de agentes mediadores.
- **Pérdida de dispositivo y credenciales:** ¿qué ocurre si se estropea un dispositivo y no se vuelve a tener acceso a las credenciales?. La recuperación del almacenamiento de agentes es un tema que no está completamente definido y aún hay pocas implementaciones. Esto lo hace un campo interesante y necesitado en el que investigar.

Por otro lado, en el proyecto se ha utilizado el *framework* ACA-Py. Son varios los *frameworks* posibles para experimentar. Posibles trabajos futuros serían desde aplicar SSI a entornos móviles hasta crear interfaces y aplicaciones para ellos.

Finalmente, un campo que podría encajar perfectamente con SSI es el de los dispositivos Internet of Things (IoT). La capacidad de emitir credenciales verificables basadas en datos de los sensores del dispositivo es una característica que puede ser de gran utilidad debido a la gran emergencia de este tipo de dispositivos. Véase de ejemplo que los sensores del coche rastreen los kilómetros del coche, de lo que se estará seguro que no se manipulan mientras se usa.

7.3. Valoración personal

El trabajo realizado ha tenido una duración de prácticamente un año. Es verdad que, al principio sobre todo, fue un gran reto, debido a que las tecnologías propuestas eran completamente nuevas para mí. Aun así, conforme las cosas salían con esfuerzo y dedicación, la satisfacción es enorme. Es el primer trabajo/proyecto de carácter profesional al que me he enfrentado, y sé que me ayudará en un futuro próximo. He aprendido a que las cosas no son tan fáciles como en las clases de universidad, donde los profesores nos guían de la mano por la asignatura.

Estoy seguro de que el proyecto podrá servirme en un futuro, ya sea como carta de presentación o como el inicio a una tecnología que me ha parecido fundamental e importante en el mundo de las telecomunicaciones. Espero poder seguir trabajando/estudiando en este ámbito, ya que pienso, tras las conclusiones a las que he llegado, que se demandaran profesionales con conocimientos sobre *blockchain* y SSI.

Bibliografía

- [1] D. McCandless, T. Evans, and P. Barton, “World’s Biggest Data Breaches & Hacks — Information is Beautiful,” <https://informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks/>, Acceso: 05/07/2022.
- [2] The Linux Foundation, “Introduction to Hyperledger Sovereign Identity Blockchain Solutions: Indy, Aries and Ursu,” <https://learning.edx.org/course/course-v1:LinuxFoundationX+LFS172x+2T2021>, Acceso: 05/07/2022.
- [3] ——, “Introduction to Hyperledger Blockchain Technologies,” <https://learning.edx.org/course/course-v1:LinuxFoundationX+LFS171x+3T2020>, Acceso: 05/07/2022.
- [4] Alastria, “Página web de ID_Alastria,” <https://alastria.io/id-alastria/>, Acceso: 05/07/2022.
- [5] D. I. Stefanescu, “Estudio y evaluación de la identidad digital en Blockchain,” 2020.
- [6] M. Hamza, “All you need to know about uPort Identity management,” <https://medium.com/@hamzamaslah/all-you-need-to-know-about-uport-identity-management-3fc49db25332>, Acceso: 05/07/2022.
- [7] D. Hardman, “0011: Credential Revocation,” <https://github.com/hyperledger/indy-hipe/tree/main/text/0011-cred-revocation>, Acceso: 05/07/2022.
- [8] The Linux Foundation, “Becoming a Hyperledger Aries Developer,” <https://learning.edx.org/course/course-v1:LinuxFoundationX+LFS173x+3T2021>, Acceso: 05/07/2022.
- [9] R. West, D. Bluhm, M. Hailstone, S. Curran, S. Curren, and G. Aristy, “Aries RFC 0434: Out-of-Band Protocol 1.1,” <https://github.com/hyperledger/aries-rfcs/tree/main/features/0434-outofband>, Acceso: 05/07/2022.

- [10] R. West, D. Bluhm, M. Hailstone, S. Curran, S. Curren, S. Curran, and G. Aristy, “Aries RFC 0023: DID Exchange Protocol 1.0,” <https://github.com/hyperledger/aries-rfcs/tree/main/features/0023-did-exchange>, Acceso: 05/07/2022.
- [11] N. Khateev, “Aries RFC 0036: Issue Credential Protocol 1.0,” <https://github.com/hyperledger/aries-rfcs/tree/main/features/0036-issue-credential>, Acceso: 05/07/2022.
- [12] ———, “Aries RFC 0037: Present Proof Protocol 1.0,” <https://github.com/hyperledger/aries-rfcs/tree/main/features/0037-present-proof>, Acceso: 05/07/2022.
- [13] S. Ribeca, “Diagrama Cajas y Bigotes,” https://datavizcatalogue.com/ES/metodos/diagrama_cajas_y_bigotes.html, Acceso: 05/07/2022.
- [14] I. Hirskyj-Douglas and A. Lucero, “On the Internet, Nobody Knows You’re a Dog... Unless You’re Another Dog,” in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 2019, pp. 1–12.
- [15] A. Mühle, A. Grüner, T. Gayvoronskaya, and C. Meinel, “A survey on essential components of a self-sovereign identity,” *Computer Science Review*, vol. 30, pp. 80–86, 2018.
- [16] C. Allen, “The Path to Self-Sovereign Identity,” <http://www.lifewithalacrity.com/2016/04/the-path-to-self-sovereign-identity.html>, Apr 2016.
- [17] K. Cameron, “The laws of identity,” *Microsoft Corp*, vol. 12, pp. 8–11, 2005.
- [18] Hyperledger, “Página oficial de Hyperledger,” <https://www.hyperledger.org/>, Acceso: 05/07/2022.
- [19] Udemy, “Cursos sobre Python - Udemy,” <https://www.udemy.com/es/topic/python/>, Acceso: 05/07/2022.
- [20] Microsoft, “Página oficial de Microsoft,” <https://www.microsoft.com/es-es/>, Acceso: 05/07/2022.
- [21] edX, “Página oficial de edX,” <https://www.edx.org/>, Acceso: 05/07/2022.
- [22] PcComponentes, “HP OMEN 15-DC0009NS Intel Core i7-8750H/16GB/1TB+256GB SSD/GTX 1050/15.6-PcComponentes,” <https://www.pccomponentes.com/hp-omen-15-dc0009ns-intel-core-i7-8750h-16gb-1tb-256gb-ssd-gtx-1050-156>, Acceso: 05/07/2022.

- [23] P. Dunphy and F. Petitcolas, “A First Look at Identity Management Schemes on the Blockchain,” *IEEE Security Privacy*, vol. 16, 01 2018.
- [24] Sovrin, “Página oficial de Sovrin,” <https://sovrin.org/>, Acceso: 05/07/2022.
- [25] uPort, “Página oficial de uPort,” <https://www.uport.me/>, Acceso: 05/07/2022.
- [26] Province of British Columbia, “Página oficial del Gobierno de British Columbia,” <https://www2.gov.bc.ca/gov/content/home>, Acceso: 05/07/2022.
- [27] ——, “Repositorio en GitHub del proyecto *von-network*,” <https://github.com/bcgov/von-network>, Acceso: 05/07/2022.
- [28] ——, “Repositorio en Github de proyecto *indy-tails-server*,” <https://github.com/bcgov/indy-tails-server>, Acceso: 05/07/2022.
- [29] Hyperledger, “Repositorio en GitHub del proyecto *aries-cloudagent-python*,” <https://github.com/hyperledger/aries-cloudagent-python>, Acceso: 05/07/2022.
- [30] Hyperledger, “Aries RFCS - GitHub,” <https://github.com/hyperledger/aries-rfcs>, Acceso: 05/07/2022.
- [31] Stephen Curran and John Jordan Province of British Columbia, “0302: Aries Interop Profile,” <https://github.com/hyperledger/aries-rfcs/tree/main/concepts/0302-aries-interop-profile>, Acceso: 05/07/2022.
- [32] R. West, D. Bluhm, M. Hailstone, S. Curran, and S. Curren, “Aries RFC 0160: Connection Protocol,” <https://github.com/hyperledger/aries-rfcs/blob/main/features/0160-connection-protocol/README.md>, Acceso: 05/07/2022.
- [33] R. Adán López, “Repositorio en Github del proyecto *aca-py-performance*,” <https://github.com/Rafaadan/aca-py-performance>, Acceso: 05/07/2022.
- [34] Virtual Box, “Página oficial de Oracle Virtual Box,” <https://www.virtualbox.org/>, Acceso: 05/07/2022.
- [35] Ubuntu, “Ubuntu 20.04.4 LTS (Focal Fossa),” <https://releases.ubuntu.com/20.04/>, Acceso: 05/07/2022.
- [36] Visual Studio Code, “Página oficial de Visual Studio Code,” <https://code.visualstudio.com/>, Acceso: 05/07/2022.

Siglas

ACA-Py Aries Cloud Agent - Python.

AP Access Point.

CR Credencial Revocable.

CS Credencial Simple.

DID Decentralized Identifier.

DLT Distributed Ledger Technology.

ID Identificador.

IdP Identity Provider.

IoT Internet of Things.

P2P Peer-to-Peer.

SSI Self-Sovereign Identity.

TFG Trabajo Fin de Grado.

URI Uniform Resource Identifier.

W3C World Wide Web Consortium.

ZKP Zero-Knowledge Proof.

Apéndice A

Manual de usuario

En este apéndice se explicará como instalar las herramientas y tecnologías necesarias para poder desplegar los entornos diseñados e implementados en el proyecto: tanto el de evaluación de funcionalidades como el de evaluación de rendimiento.

A.1. Virtual Box: máquina virtual Ubuntu 20.04 LTS

Se ha utilizado el software de virtualización Virtual Box a lo largo del proyecto, más específicamente la versión 6.1.26, por lo que es la que se recomienda si se quiere desplegar el entorno creado. Este software ya bastante conocido permite desplegar máquinas virtuales con sistemas operativos diferentes al del *host*. El sistema operativo del *host* utilizado para ejecutarlo ha sido Windows 10.

Para instalar Virtual Box, acceder a la página web oficial de Virtual Box [34] y descargar la versión deseada. Una vez instalada, debe salir una interfaz como la de la Figura A.1.

Lo siguiente es instalar el sistema operativo de la máquina virtual que se va a utilizar. Durante el proyecto se ha utilizado Ubuntu 20.04 LTS. Para poder utilizarlo, hay que descargar la imagen en la página de Ubuntu [35] (Figura A.2).

Seguidamente hay que crear la máquina virtual. Para ello, ir a Máquina y Nueva. Se pedirá que se rellenen el nombre de la máquina y el sistema operativo como se indica en la Figura A.3.

Hay que elegir el tipo de archivo de disco duro que se quiere crear. Escoger VDI (VirtualBox Disk Image). Hay dos opciones: reservado dinámicamente y tamaño fijo. Se recomienda usar tamaño fijo debido a que al

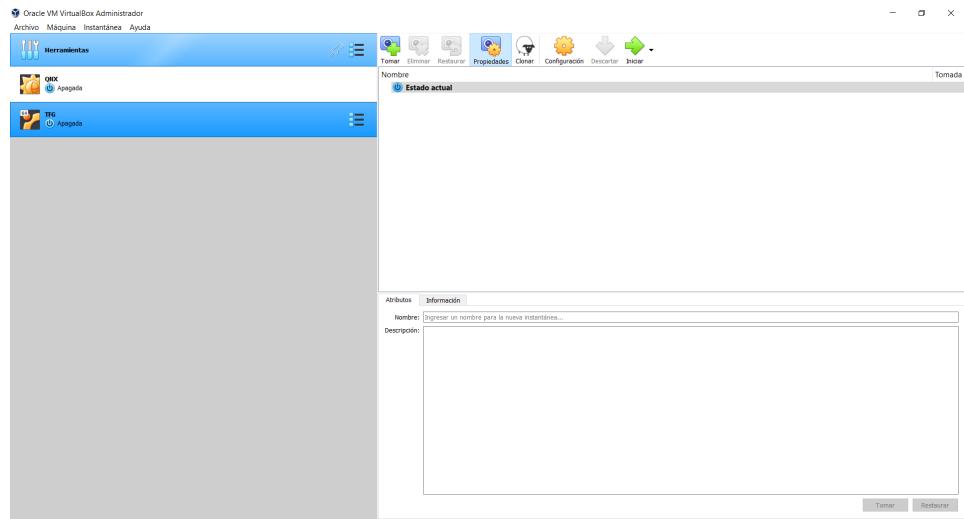


Figura A.1: Interfaz Virtual Box.



Figura A.2: Página de descarga Ubuntu 20.04 LTS.

reservarlo dinámicamente el rendimiento de la máquina virtual decae bastante (Figura A.4).

Una vez configurada y creada la máquina virtual hay que iniciarla. Se pedirá elegir el disco de inicio, donde se deberá elegir la imagen de Ubuntu 20.04 LTS descargada anteriormente (archivo .iso) (Figura A.5).

Una vez seleccionado, se iniciará una guía de instalación de Ubuntu (Figura A.6).

Se pedirán que seleccione si quiere instalar software adicional como reproductores multimedia o juegos. Al ser la máquina virtual de propósito educativo no se recomienda marcar esa opción para ahorrar espacio (Figura

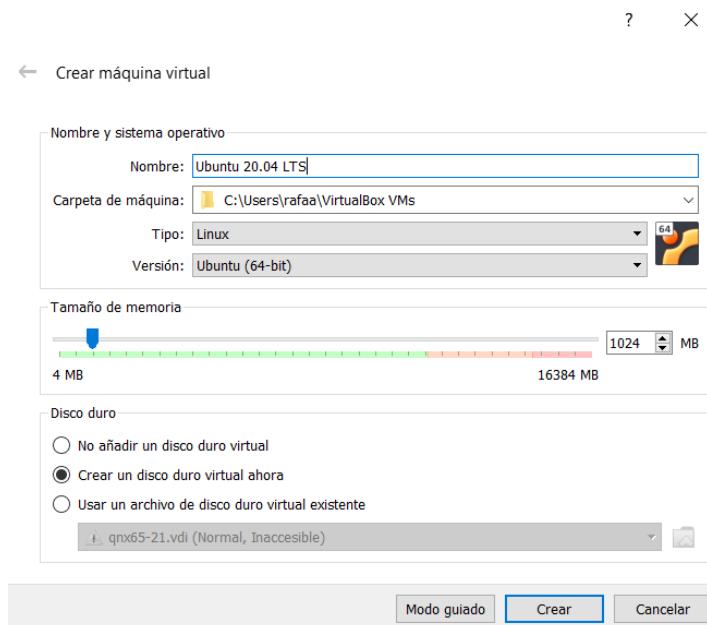


Figura A.3: Creación máquina virtual.

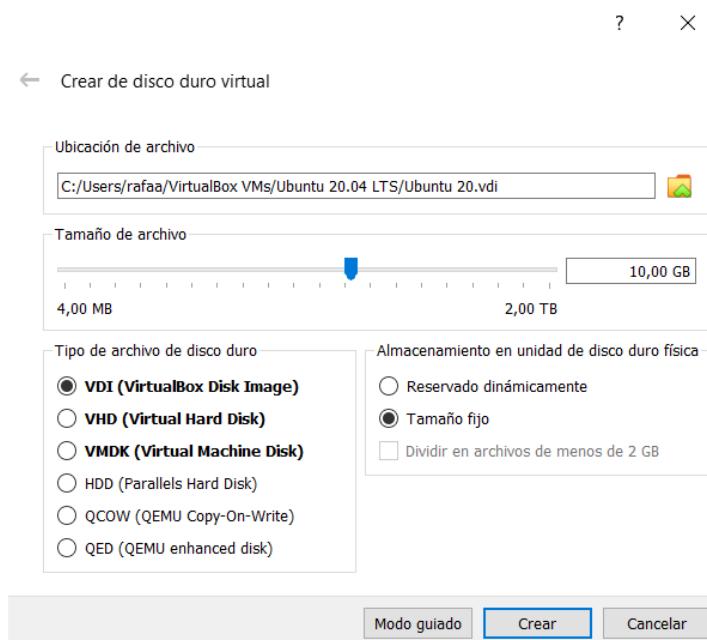


Figura A.4: Crear disco virtual.

A.7).

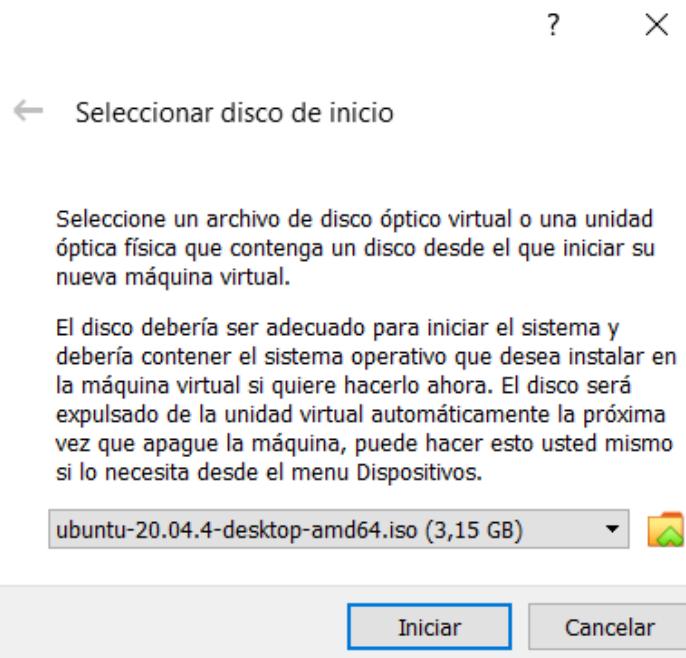


Figura A.5: Selección de disco de inicio.

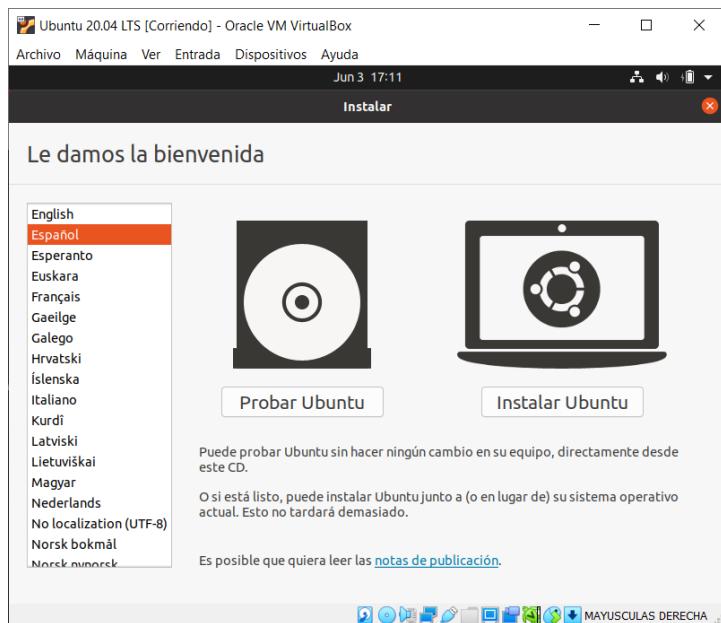


Figura A.6: Guía de instalación Ubuntu.

Finalmente, para acabar con la instalación, se marcará la opción de bo-

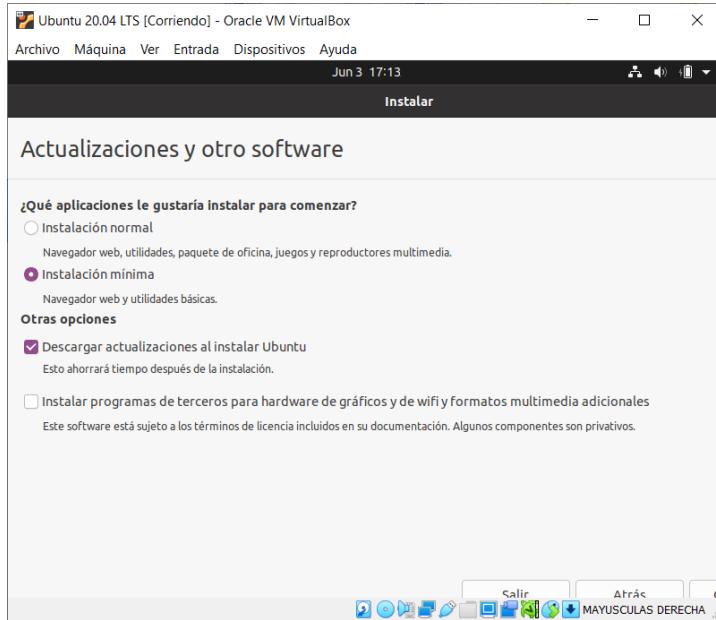


Figura A.7: Software adicional.

rrar el disco e instalar Ubuntu (Figura A.8). La instalación tomará unos minutos. Una vez acabada ya se tendrá una máquina virtual con sistema operativo Ubuntu 20.04 LTS sobre Virtual Box.

A.2. Módulos y requerimientos

Una vez dentro de la máquina virtual es necesario la instalación de algunos módulos y programas.

A.2.1. Git

Para el control de versiones y poder tener disponible el repositorio en la nube se utiliza el software Git (y la plataforma GitHub [33]). Para instalarlo es necesario ejecutar los siguientes comandos:

```
sudo apt update
sudo apt install git
```

Una vez instalado, ya se puede descargar el repositorio del proyecto. Simplemente hay que ejecutar el siguiente comando:

```
git clone https://github.com/Rafaadan/
aca-py-performance
```

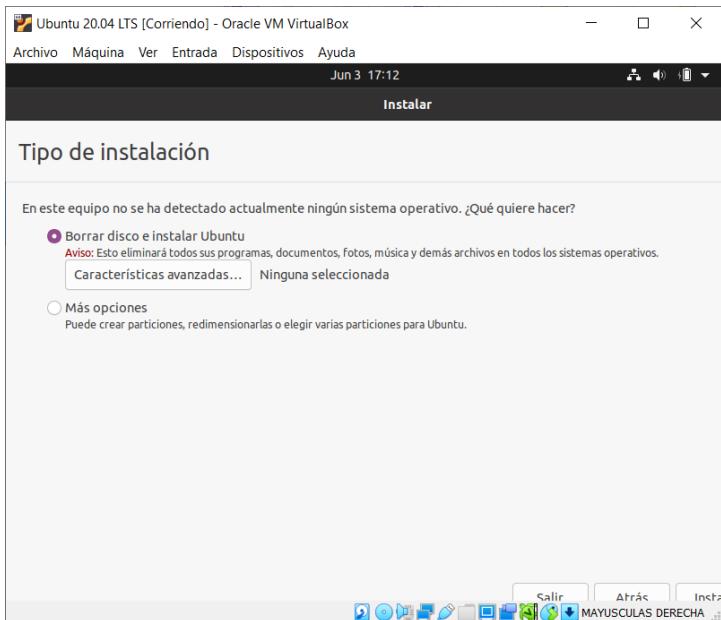


Figura A.8: Final de instalación Ubuntu.

Una vez hecho esto ya estará descargado el repositorio del proyecto. También hay que descargar a parte los repositorios externos utilizados (*von-network* e *indy-tails-server*) y colocarlos en las carpetas correspondientes del repositorio del proyecto:

```
git clone https://github.com/bcgov/von-network
git clone https://github.com/bcgov/indy-tails-server
```

A.2.2. Visual Studio Code

Visual Studio Code [36] es un editor de código fuente que se ha utilizado a lo largo del proyecto. Cuenta con extensiones para todos los tipos de lenguajes lo que lo hace una elección adecuada.

Para instalarlo en la máquina virtual, es necesario ejecutar en la cmd el siguiente comando:

```
sudo snap install --classic code
```

A.2.3. Docker y Docker Compose

Para el despliegue de contenedores se ha utilizado Docker y Docker Compose. Docker es una aplicación que simplifica el proceso de aplicación en contenedores. Se despliegan procesos que gastan menos recursos que las máquinas virtuales. Para instalarlo, ejecutar los siguientes comandos:

```
sudo apt update
sudo apt install apt-transport-https
ca-certificates curl software-properties-common

curl -fsSL https://download.docker.com/linux/
ubuntu/gpg | sudo apt-key add -

sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/
ubuntu focal stable"

sudo apt update

sudo apt install docker-ce
```

Para poder ejecutar Docker sin *sudo*:

```
sudo usermod -aG docker ${USER}

su - ${USER}
```

Por otro lado, es necesario instalar también Docker Compose:

```
sudo curl -L "https://github.com/docker/
compose/releases/download/1.26.0/docker-
compose-$(uname -s)-$(uname -m)" -o /usr/
local/bin/docker-compose

sudo chmod +x /usr/local/bin/docker-compose

docker-compose --version
```

A.3. Implementación de los entornos

Para poder implementar los entornos, consultar la Sección 5.