Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Grado en Ingeniería Informática

Una primera aproximación al Método DID para KERI

A first approach to the DID method for KERI

Pablo Molina Martínez

La Laguna, *2* de *Julio* 2021

D. José Luis Roda García, con N.I.F. 43.356.123-L profesor Titular de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

D. Antonio Estévez García, con N.I.F. 43.615.400-V Director del área de I+D de Open Canarias S.L., como cotutor

C E R T I F I C A (N)

Que la presente memoria titulada:

“Una primera aproximación al Método DID para KERI”

ha sido realizada bajo su dirección por D. **Pablo Molina Martinez**,

con N.I.F. 42.240.018-N.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 2 de Julio de 2021

Agradecimientos

Quiero agradecer a todas las personas que han estado conmigo durante esta etapa, ya sea dando apoyos con el proyecto o de manera personal.

Quiero agradecer a mis tutores de TFG Jose Luis Roda Garcia y Antonio Estévez García por su ayuda y apoyo en todo momento, y sobre todo por la manera en la que me han guiado para poder llevar a cabo este proyecto.

En especial quiero agradecer a mi familia cercana, padre, madre, hermana, por siempre apoyarme en todo momento y brindarme momentos de calma cuando lo necesitaba, gracias por apoyarme siempre en todo lo que hago y animarme a seguir adelante.

A todos ellos muchas gracias por haberme apoyado e inspirado.

Licencia

Resumen

En este trabajo de fin de grado se plantea el estudio, análisis y desarrollo de una tecnología emergente, la tecnología de la auto identidad soberana SSI (Self Sovereign Identity). Con el uso de esta tecnología se busca que un usuario tenga control total de sus datos y no necesite depender de terceros para poder realizar operaciones tan básicas como demostrar que uno mismo es quien dice ser. En un principio esto es una tarea muy complicada de conseguir, sobre todo en internet en donde, la suplantación de identidad, el *ip spoofing* y miles de métodos maliciosos más están a la orden del día.

Este trabajo poco a poco va a ir adentrándose en el mundo de las auto identidades soberanas, empezando por los fundamentos básicos de este nuevo grupo de tecnologías. Los identificadores descentralizados (DID) llevan ya un tiempo en uso y permiten la autenticación de un individuo sobre una fuente de conocimiento común, muchas veces conocida como blockchain. En este trabajo se va a ampliar el uso de los DID. Se pasará de necesitar una raíz de confianza común, a que cada individuo tenga su propia raíz de confianza. Se verá que no es necesaria la raíz de confianza común para conseguir la autenticación de un individuo. Gracias a no depender de dicha raíz de confianza común, el individuo genera y maneja claves públicas de forma segura. Dicha seguridad se basa en mecanismos de seguridad post cuánticos y sistemas de registro para poder llevar a cabo un control de las acciones que se realizan. Todos estas características se ven cumplidas con KERI, una infraestructura para el manejo de claves. A su vez, KERI es una tecnología que se encuentra en constante cambio y evolución, de reciente aparición que está buscando ser el estándar en el mundo de las SSI.

Para finalizar tengo que recordar la última frase del párrafo anterior y es que KERI es una tecnología de reciente aparición, la cual todavía no tiene un estándar de uso definido y en la cual todavía quedan muchas barreras, siendo sin duda la usabilidad una de las más difíciles. Como se verá en este trabajo, KERI es una tecnología que en su planteamiento parece que no tiene fallos, pero el verdadero problema llega a la hora de su implementación, es por eso que este trabajo busca arrojar un poco de luz a un abismo que tarde o temprano se va a tener que iluminar.

Palabras clave: Raíz de confianza, Identidad soberana, Identificadores descentralizados, KERI

Abstract

This end of degree project implies a study, analisis and development of an emergent technology, the Self Sovereign Identity (SSI) technology. The use of this technology aims for a user to have full control of their data, not depending on third parties for doing basic operations like acknowledging their own identity. This task is very hard to archive at first sight, moreover in the internet where the souplantation of identity, the ip spoofing and thousand oh malicious methods are everywhere.

This project is going to take on the world of the self sovereign identities slowly, starting from the basics of this new group of technologies.The Decentralized Identifiers (DID) have been present for a short period of time, allowing an individual to authenticate itself by using a common root of trust most of the time know as Blockchain. This project is expanding the use of the DID. Starting from needing a common root of trust as previously mentioned, to the individual being their own root of trust. Being the ultimate goal to demonstrate that a user can authenticate itself without needing to depend on others. What this implies is that the individual has full control over his keys, being the one who generates and performs the actions that are made with them. The security needed to archive this can be done with post cuantum mechanism and keeping control over the operation done with the identifier. All of these specifications can be meeted with KERI, an infrastructure that's designed for key management. KERI is an infrastructure that's in constant change and evolving, being a new technology in the world of the SSI, trying to be the standard.

To end this abstract i would like to remember the last phrase from the previous paragraph, KERI is a new technology, what that means is that there isn't a defined standard on how it should be implemented. It has many problems when implementing it, being the usability his biggest flaw right now. As it will be seen in this project, KERI is a technology that in its theoretical bases seems flawless, the implementation being its biggest problem, that's why this project tries to bring light over the abysm that is going to need to be lighted soon.

Keywords: Root of trust, Sovereign Identity, Decentralized Identifiers, KERI

Índice general

Índice de figuras

Índice de tablas

Capítulo 1 Introducción

Actualmente el estándar para la identificación en internet es usar la infraestructura conocida como PKI (Public Key Infraestructure), en la cual se basan la mayoría de sistemas de autenticación de hoy día [1]. Esta infraestructura empieza con una CA (Certified Authority) que se encarga de firmar los diferentes certificados. Una vez firmados son enviados a las VA (Validation Authorities), cuando un usuario quiere certificarse es necesario que tenga un certificado firmado por una CA y validado por una VA. Esto crea una estructura en pirámide ya que todo certificado que quiera ser válido necesita ser creado por una CA y que esa CA comunique a la VA la creación y validez de dicho certificado.

A continuación se puede observar lo previamente descrito en la figura 1.1, la cual representa la estructura de un PKI clásica en la cual una CA emite una clave pública a la VA y le da una clave privada al solicitante con la cual interactuar frente a la VA.

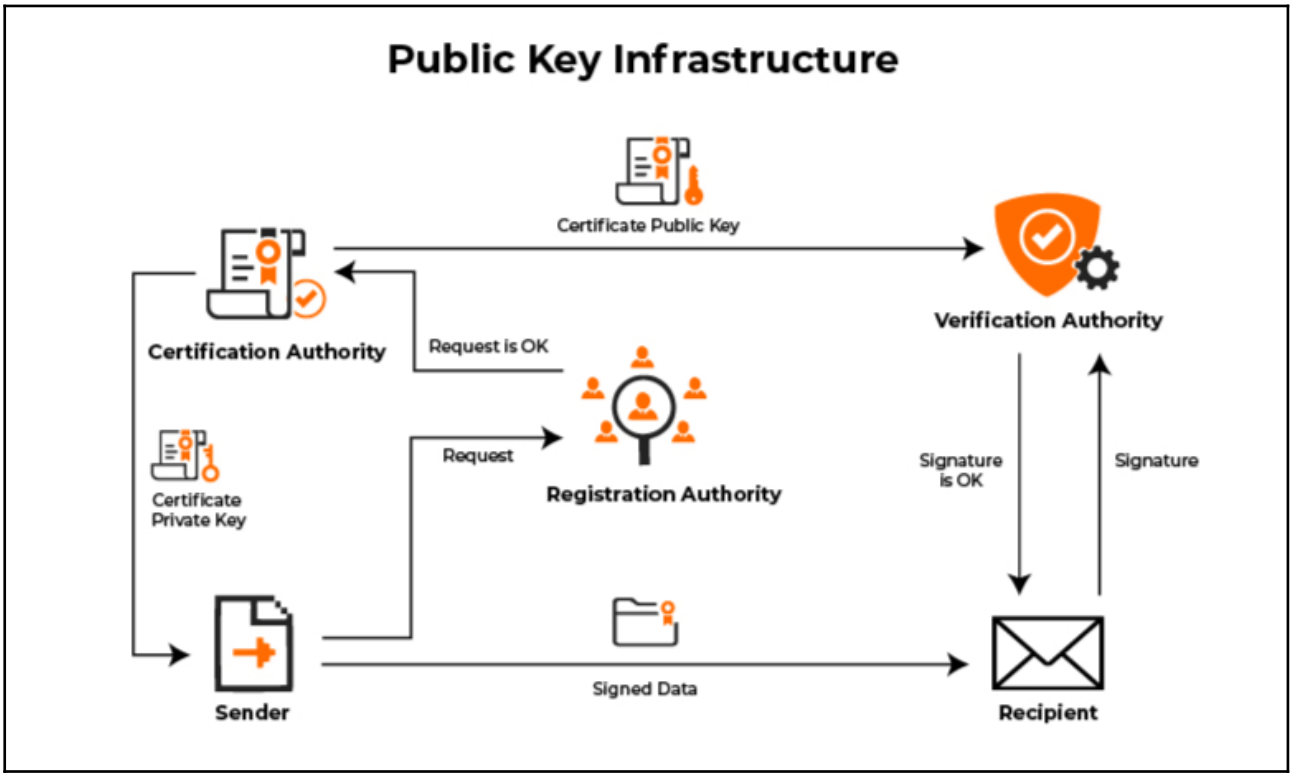

Figura 1.1: Estructura de las PKI
(Fuente:https://www.appviewx.com/education-center/pki/)

El mayor inconveniente es que se depende de una CA para poder tener credibilidad, es aquí cuando se plantean el uso de tecnologías descentralizadas para intentar tener una independencia digital.

En los últimos años, las tecnologías descentralizadas y más específicamente la Blockchain han crecido de manera exponencial [2]. Son tecnologías que no tienen una

estructura piramidal como la previamente mencionada en las PKI y que además ofrecen claridad y seguridad de los datos. La Blockchain es usada en todos los campos que podamos imaginarnos, desde transacciones de activos digitales hasta la capacidad de ser identificados gracias a ella, es en este último aspecto, en el de la identificación, donde la Blockchain puede ser una opción aparentemente viable, pero un sistema de identificación no necesita el 100% de las características proporcionadas por una Blockchain, no es necesario que los identificadores sigan un orden (como lo siguen los bloques de una cadena) se busca la capacidad de modificar nuestros datos en cualquier momento, que sigan siendo válidos y más importante, que los datos sean visibles solo para los miembros de la relación.

En tiempos más recientes han salido a la luz diferentes métodos para poder resolver de alguna manera los problemas anteriormente mencionados. La idea actual más innovadora es KERI [29] (Key Event Receipt Infraestructure), un sistema que pretende utilizar los conocimientos y utilidades de la Blockchain para llevar a cabo relaciones de confianza entre dos o más usuarios sin depender de una Blockchain común. Es una estructura que se encuentra todavía en desarrollo. Su modo de comunicación directo define una serie de características para realizar comunicaciones de confianza, siendo los propios usuarios certificadores de sí mismos y por tanto teniendo control total de los datos que envían y de quién los recibe, además de tener el control sobre la relación pudiendo mantenerlas u olvidarlas. En cuanto a su seguridad frente a otros mecanismos, KERI ofrece seguridad post cuántica gracias a su modelo de rotación de claves preparándose para un futuro donde los ordenadores cuánticos están a la orden del día.

En conjunto, KERI define un nuevo peldaño en el mundo de las identidades, dando más control, más seguridad y más flexibilidad de uso a los usuarios sobre sus identificadores.

1.1 Casos de uso

Para poder ilustrar mejor cual es la verdadera ventaja de utilizar sistemas de identidad auto soberana como KERI, voy a plantear una serie de casos de uso los cuales hacen verdaderamente brillar las capacidades de dicha tecnología. …

Caso de uso de identificación de un menor, en este caso veremos como KERI puede usarse para mantener el control sobre la identidad de un menor. Supongamos que un menor de edad necesita identificarse frente a terceros, por ejemplo un registro en un campamento de verano. Su tutor legal controla su identidad y puede usarla para verificar a su hijo frente al sistema informático del campamento siendo él el que controla los datos del menor. En un futuro el menor de edad pasará a ser considerado adulto y por tanto responsable de sus datos, el tutor legal en dicha situación puede delegar el control del identificador a un controlador sobre el cual tenga posesión el ahora adulto. Es decir, KERI nos proporciona maneras de tener control sobre identificadores que no nos representan y poder delegar su control en cualquier momento si se quisiera.

Otro caso de uso muy interesante que se puede explorar es el de el uso de un identificador sobre un producto para tener un seguimiento exacto de su vida útil, es decir un coche podría tener un identificador el cual está verificado por la casa automovilística del vehículo sobre el cual se pueden registrar los diversos eventos en la vida útil del mismo, empezando por un evento de compra asociado a un cliente y pudiendo agregar eventos a cada visita al mecánico, cambio de piezas e incluso cambió de dueño. Esto permitiría al nuevo dueño del coche tener un control de todos los eventos que han ocurrido en la vida del coche además de ayudar a un mecánico o a la propia casa a

buscar maneras para arreglar el vehículo viendo su historial de fallos.

Por último me gustaría explicar el caso de uso más común y es el de identificar al individuo frente a una organización, parecido con el del padre y el menor pero en este caso el propio individuo es el controlador del identificador. Este caso es muy interesante porque al tratarse de una organización, por ejemplo un red social, para identificarnos frente a la red social seríamos nosotros los que tendrían todos los datos de acceso, números de teléfono, correos de contacto etc., dando solo a la red social los datos que nosotros queramos de nuestro identificador. Esta es una de las grandes ventajas de las entidades descentralizadas, que los usuarios tengan un control total sobre los datos que comparten.

1.2 **Objetivos**

El objetivo de este proyecto de fin de grado es el de realizar una investigación sobre las tecnologías previamente mencionadas y establecer una base teórica para poder empezar con su desarrollo e implementación futuras. No se busca desarrollar al completo dichas tecnologías ya que es un trabajo que se sale del alcance al que se puede llegar en un TFG, se busca acercar y entender una serie de nuevos conceptos desde un enfoque técnico con ejemplos reales y posibles implementaciones futuras que ayuden a entender un campo que se encuentra en constante evolución.

Es por eso que esta investigación está basada en diferentes estudios hechos por profesionales del campo, siendo la misma una recopilación de dichos estudios puestos sobre un mismo contexto y relacionándolos entre ellos, sacando conclusiones que den a ver las posibilidades de dichas tecnologías y cómo podrían ser usadas en el mundo real.

Además de lo previamente mencionado, uno de los objetivos es crear un prototipo del modelo de funcionamiento esperado del método para en un futuro poder realizar implementaciones reales.

Capítulo 2 Identificador Descentralizado (DID)

Para poder empezar a entender este trabajo voy a ir explicando los conceptos que componen su título Método DID para KERI, empezando con DID continuando con KERI y finalizando con la unión de ambos.

Un DID, por su abreviación en inglés Decentralized Identifier [3] es un Identificador Descentralizado, es decir es un identificador que no necesita estar asociado a ninguna entidad centralizada, como puede ser un Gobierno o Empresa. Esto en sí dice poco, pero es un concepto que abarca mucho. Al fin y al cabo, un DID no es más que una cadena de caracteres conocida como URI (Uniform Resource Identifier) la cual asocia a un Sujeto DID con un Documento DID, mediante un identificador del Método DID con el que interactuar con dicho Documento DID y un identificador específico del Método DID el cual concreta la interacción con el Documento DID [4].

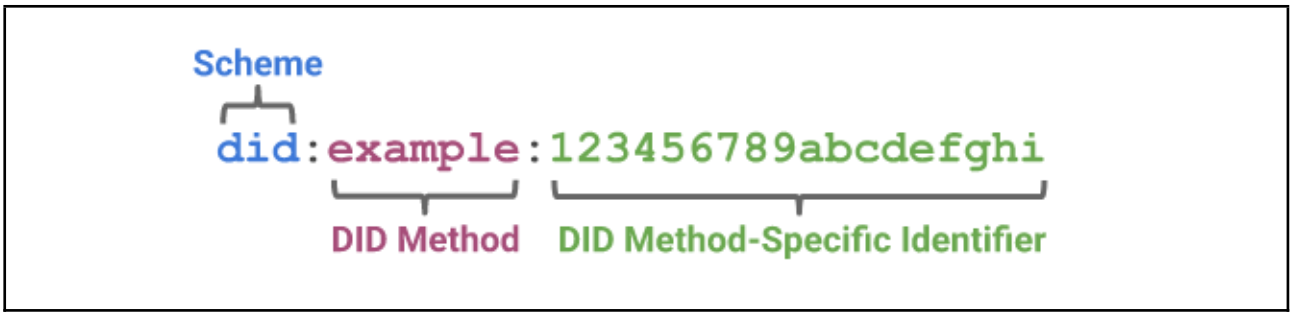

Figura 2.1: Esquema de un DID
(Fuente:https://www.w3.org/TR/did-core/#a-simple-example)

Para que un identificador sea considerado un DID ha de cumplir 4 propiedades [5], las cuales tienen son:

1. Identificador permanente, es decir que no necesite cambiar para poder cumplir su función.
2. Identificador resoluble, que se pueda utilizar para recoger datos asociados al sujeto.
3. Identificador criptográficamente verificable sobre el cual se pueda probar su posesión y control mediante métodos criptográficos.
4. Identificador descentralizado que no necesite de ninguna autoridad centralizada para su registro.

Las 2 primeras propiedades se cumplen en la mayoría de identificadores ya que son las que permiten el uso del mismo. Lo que hace único a los DID son las propiedades 3 y 4 ya que pueden ser resueltas dependiendo únicamente de la criptografía. Un Controlador que posea la llave privada de un DID puede generarlo sin necesidad de una autoridad

centralizada y verificar que es su dueño mediante la relación que mantienen las claves público privadas obviando otra vez el uso de una autoridad centralizada.

2.1 Sujeto y controlador

Un DID hace referencia a un Sujeto [6], es decir a una entidad, dicha entidad puede ser una persona, un objeto, una organización … en conclusión algo que se necesite representar e identificar. El DID está asociado siempre al mismo Sujeto que a su vez hace referencia a un Documento DID, para poder controlar las interacciones con el Documento y para las acciones que se quieren realizar con el mismo existe el papel del Controlador, una entidad encargada de realizar los cambios en el Documento a través del uso de un par de claves públicas. Dicho par se conoce como el par de claves de control, que como veremos en KERI son usadas en conjunto con un par de claves de rotación.

Cabe destacar que el Controlador de un DID [7] pueden ser varias entidades, además de ser a la vez Controlador y Sujeto. Un ejemplo que ilustra bien esta situación es el de un menor de edad que es representado por un DID siendo el menor el Sujeto, su tutor legal siendo el Controlador que toma el control sobre las acciones que puede realizar hasta que sea mayor de edad. El Sujeto podría realizar una serie de acciones permitidas por el Controlador. Esta delegación de acciones crea un Delegado, que es un controlador que no tiene control total sobre el DID, pero puede realizar una serie de acciones determinadas por el Controlador. Ilustrado con el ejemplo anterior, el tutor dejaría al menor de edad usar el DID para identificarse, pero no le permite modificar los valores del Documento.

2.2 El Documento DID

El Documento DID [8] es una serie de datos que describen al Sujeto DID, dichos datos suelen estar relacionados con la información que representa al Sujeto así como con el material criptográfico como claves pública que el Sujeto o Delegado pueden usar para autenticarse o realizar operaciones.

La estructura básica de un Documento DID, está compuesta por las propiedades del documento [9] y la opción de dos tipos de secciones, las secciones de métodos de verificación y las secciones de servicios:

- Las propiedades del documento están compuestas por una serie de parámetros de los cuales solo el ID del documento es obligatorio para que sea considerado un documento DID. Opcionalmente puede tener una serie de secciones que ayuden a darle utilidad al documento, estas secciones son las relacionadas con los métodos de verificación, los cuales pueden ser secciones de autenticación, acuerdo de claves, capacidad de delegación, etc., además de los servicios que proporciona el documento.
- La sección que representa los métodos de verificación, está compuesta por los campos obligatorios que son el id, el controlador y el tipo de verificación que se quiere realizar (verificación, acuerdo de claves ...), además se puede especificar opcionalmente la manera para recuperar la clave pública del método.
- La sección que representa los servicios disponibles para usar con el DID. Esta sección está compuesta de 3 parámetros obligatorios, el id, el tipo de servicio y el punto de encuentro con el servicio.

Viendo los diferentes elementos que componen un Documento DID falta ver cómo se

representan dichos elementos para poder usar el Documento. La W3C propone dos métodos de representación para los Documentos DID.

2.2.1 JSON

Los documentos JSON [10] (Javascript Object Notation) son un formato de archivos para almacenar información en un formato fácil de entender el cual usa una estructura de pares de atributo valor y arrays.

La ventaja de usar JSON es que se encuentra soportado por la mayoría de la infraestructura actual, siendo considerado el estándar cuando se trata de archivos en la web.

```
{
  "id": "did:example:123456789abcdefghi",
  "authentication": [{ //metodo de verificacion
    "id": "did:example:123456789abcdefghi#keys-1",
    "type": "Ed25519VerificationKey2018",
    "controller": "did:example:123456789abcdefghi",
    "publicKeyBase58": "H3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV"
  }],

  "keyAgreement": [{
    "id":"did:example:123#zC9ByQ8aJs8vrNXyDhPH",
    "type": "X25519KeyAgreementKey2019", // external (property value)
    "controller": "did:example:123",
    "publicKeyMultibase": "z9hFgmPVfmBZwRvFEyniQDBkz9LmV7gDEq"
  }],

  "service": [{
    "id":"did:example:123#linked-domain",
    "type": "LinkedDomains",
    "serviceEndpoint": "https://bar.example.com"
  }]
}
```

Figura 2.2 Ejemplo de Documento DID JSON
(Fuente:https://www.w3.org/TR/did-core/#json)

2.2.2 JSON-LD

Un documento JSON-LD [11] (Javascript Object Notation Linked Data) [14], tiene un formato de datos enlazado basado en el formato de los JSON. Su origen viene de la necesidad de utilizar RDF (Resource Description Framework), el cual permite modelar la información presente en un recurso web, con el actual estándar de archivos en la web JSON.

Para conseguir la unificación de ambos conceptos se le añade a un documento JSON un contexto el cual ayuda a mapear el modelo de datos JSON a un modelo RDF. Esto da mucha utilidad pudiendo definir ontologías web y por tanto aprovechando las características de las mismas como la separación de los documentos por áreas de información y por tanto ayudando a la separación y ordenación de los datos.

```
{
  "@context": [
    "https://www.w3.org/ns/did/v1",
    "https://did-method-extension.example/v1"
  ],
  "id": "did:example:123456789abcdefghi",
  ...
}
```

Figura 2.3 Ejemplo de Documento DID JSON-LD
(Fuente:https://www.w3.org/TR/did-core/#json-ld)

2.3 Los métodos DID

Los DID están presentes en diferentes sistemas de confianza, por ejemplo, se puede tener un DID asociado a una blockchain y otro DID asociado a otra blockchain, los dos tienen que poder cumplir las mismas operaciones independientemente del lugar en el que se encuentren y para poder conseguir esto, están los Métodos DID [12] que permiten a un DID cumplir con sus funciones sin verse limitados a la raíz de confianza a la que pertenecen. Recordemos que, en la estructura de un DID, la segunda parte especifica el método al cual hace referencia, *DID:metodox:123456...*, este *métodox* permite identificar qué sistema de confianza está asociado al DID y cómo interactuar con el Documento DID. Una vez el método está identificado entra en juego el identificador específico del método (en el ejemplo anterior *123456...*), el cual define la operación a realizar y los parámetros que se quieren aplicar, dichas operaciones se ven a continuación.

2.3.1 Operaciones CRUD

Un Método DID necesita poder ejecutar una serie de operaciones para poder manejar los Documentos DID. Estas operaciones están basadas en las actualmente usadas en una base de datos para poder interactuar con ella, son las operaciones CRUD [13] (Create Read Update Delete) pero con una ligera modificación para poder interactuar con un DID.

- **Create**: Define cómo crear un DID y su documento asociado.
- **Resolve**: Define cómo se recupera el documento asociado con un DID.
- **Update**: Define cómo se modifican los contenidos de un documento asociado a un DID.
- **Deactivate**: Define como se inhabilita un DID para que no se pueda usar más.

Además de las operaciones previamente mencionadas es necesario la implementación de una operación extra para poder autorizar a los diferentes Controladores a realizar diferentes operaciones ya sea de firma o de generación de nuevas claves criptográficas.

2.4 Funcionamiento

Una vez se tiene un concepto de los componentes principales que conforman un DID es hora de ver como funcionan entre ellos, primero pasaremos a ver una imagen la cual relaciona todos los componentes y conceptos:

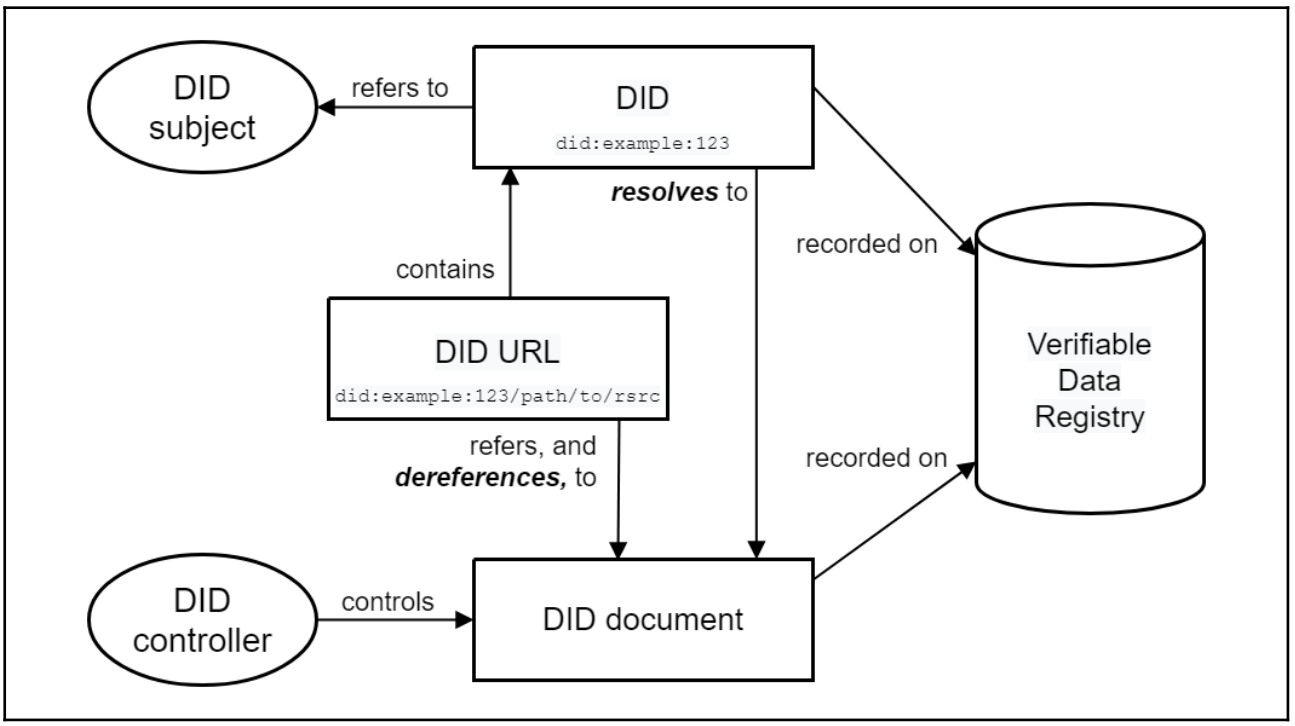

Figura 2.4:Funcionamiento de un DID.
(Fuente:https://www.w3.org/TR/did-core/#architecture-overview)

Como se puede ver en la Figura 2.2, el DID hace referencia a un sujeto, además contiene una url la cual se puede utilizar para resolverse a su documento, el control sobre el documento lo realiza el controlador, que recordemos puede o no ser también el sujeto sobre representado por el DID. Finalmente se tiene un registro de datos verificable en el cual se almacena el DID y el documento. Como veremos en KERI este registro de datos verificable se encuentra en todo momento al alcance del controlador permitiendo así tener un control directo sobre lo que se tiene almacenado en él y sobre lo que se resuelve del mismo.

Para poder entender mejor el funcionamiento utilizare uno de los casos de uso que se presentó al inicio del documento, en concreto el primer caso. Recordemos que en este caso tenemos a un menor el cual tiene su identidad manejada por su tutor legal y se quiere identificar al menor frente a un campamento para poder inscribirlo. Siguiendo el esquema anterior, el menor es el sujeto al cual representa el DID, el controlador es su tutor legal y en el documento se encontrarán las claves que le permiten verificar su autenticidad además de la posibilidad de tener un servicio que sirve para identificarlo frente al campamento. El registro de datos verificable sería la identificación del menor o en el caso que veremos con KERI un log eventos sobre el cual se pueda verificar la autenticidad del menor.

Capítulo 3 KERI

Una vez que ya se ha visto lo que son los DID, podemos entender lo que es un DID y un método DID. Ahora es cuando se resuelve la última parte para poder entender el título de este trabajo, KERI.

KERI, son las siglas de Key Event Receipt Infrastructure [27], la traducción literal nos dice que KERI es una infraestructura de recibos de eventos de clave, esto puede sonar muy complicado. Para poder entender mejor este concepto digamos que KERI es una infraestructura para el manejo de claves que permite el manejo de dichas claves de manera distribuida y descentralizada.

KERI parte de una serie de características propias de una DLT sobre las cuales descarta las que no son necesarias para cumplir su función simplificando y optimizando así su funcionamiento, mejorando aspectos frente a los sistemas de autenticación que usan DLT clásicas como blockchain.

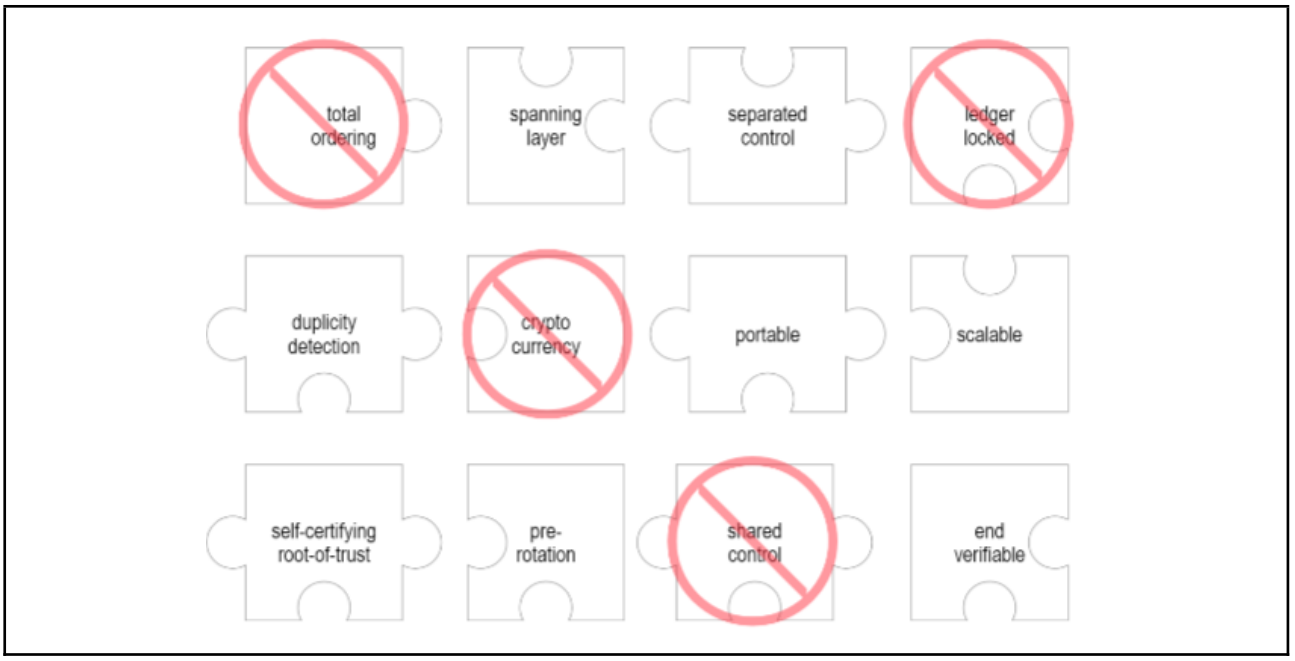

Figura 3.1: características de una DLT y cuales descartar KERI.
(Fuente:https://github.com/SmithSamuelM/Papers/blob/master/whitepapers/KERI_WP_2.x.web.pdf)

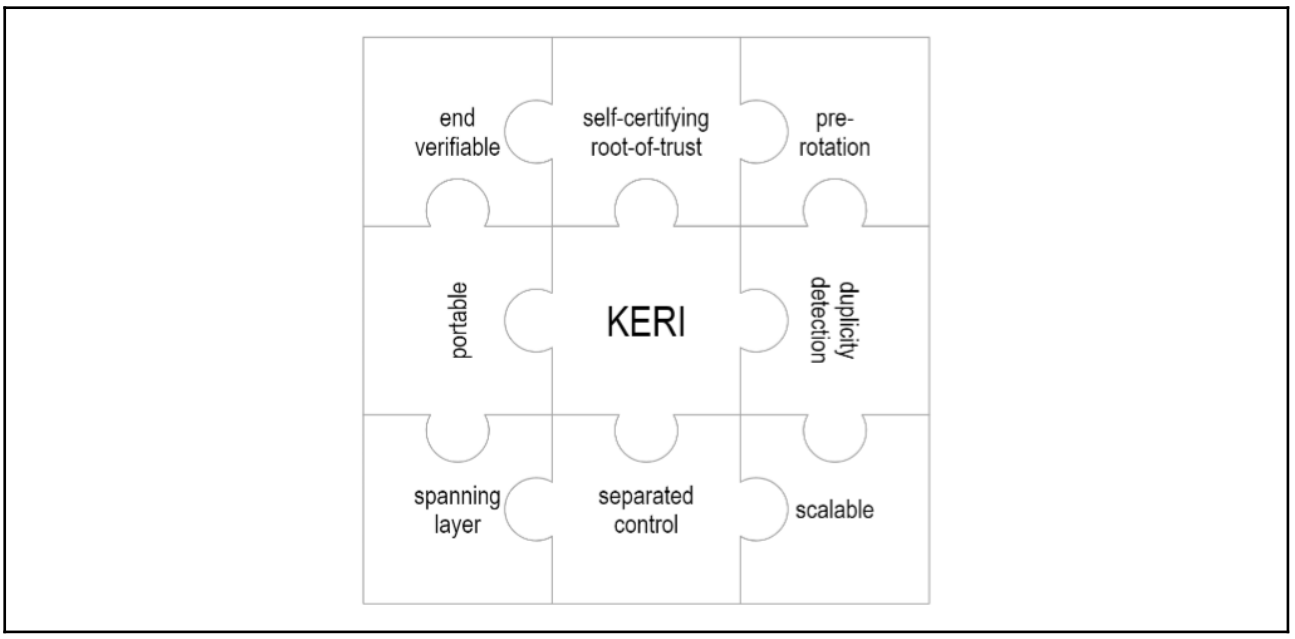

Figura 3.2: Características que usa KERI.
(Fuente:https://github.com/SmithSamuelM/Papers/blob/master/whitepapers/KERI_WP_2.x.web.pdf)

Las figuras anteriores muestran las características que KERI descarta. Características como el ordenamiento total presente por ejemplo en los bloques de una Blockchain. Esto se debe a que como se verá más adelante es el controlador el encargado de ordenar los eventos. Se prescinde también de la necesidad de usar una DLT y del uso de criptomonedas. El control compartido sobre el identificador no es necesario ya que el controlador es el único propietario del identificador. Todas estas características no son necesarias para un sistema que pretende identificar a un usuario frente a una entidad y es por eso que KERI prescinde de ellas [30].

Las características que sí utiliza KERI son su verificabilidad final, raíz de confianza auto certificada, pre rotación, portabilidad, detección de duplicidad, expansión de capa, separación del control y escalabilidad [31]. Todas estas características poco a poco se van a ir entendiendo y explicando durante el desarrollo de este capítulo.

KERI, al descartar dichas características, pierde capacidades para crear por ejemplo sistemas de criptomonedas, donde el orden de los eventos es indispensable para poder llevar a cabo un correcto control de la divisa, o por ejemplo las actualmente famosas NFT [34] (Non Fungible Tokens), los cuales es necesario tenerlos ordenados y asociados a una Blockchain para mantener su validez.

3.1 Diferentes tipos de raíces de confianza

Previamente se menciona que KERI sigue un tipo de raíz de confianza [32] conocido como raíz auto certificada, para poder entender mejor qué es una raíz auto certificada se van a exponer los diferentes tipos de raíces de confianza que se pueden encontrar en un identificador.

3.1.1 Raíz Administrativa

Es el tipo de confianza que se usa en las PKI clásicas, es necesario que exista una Autoridad Certificadora (CA) manejada por seres humanos los cuales firman dichos certificados siguiendo una serie de procedimientos y protocolos seguros. La raíz de confianza está manejada por una entidad, dicha raíz tiene control sobre la validez y uso

de todos los certificados que genera pudiendo revocarse en sus Listas de Revocación de Certificados (CSR). Este tipo de raíz representa una estructura jerárquica y centralizada donde la confianza de todos los certificados finales recae sobre la confianza que se tenga en la entidad raíz y se asume que han sido verificados de manera correcta.

3.1.2 Raíz Algorítmica

Este tipo de raíz de confianza es relativamente reciente, basa su funcionamiento en la resolución de un algoritmo el cual crea un sistema seguro descentralizado sobre el cual nadie tiene la propiedad. El ejemplo más usado y común hoy en día es el de las blockchain, las cuales basan su confianza sobre este principio. Algunos parámetros para medir el nivel de seguridad que existe sobre los sistemas que se basan en la raíz algorítmica pueden ser la cantidad de nodos que participan en la red, la longevidad de la red, el mecanismo de consenso usado entre los diferentes nodos … Estas redes sin embargo tienen un inconveniente y es que suelen gastar muchos recursos ya sea en sus sistemas de ordenación ya que están orientadas a múltiples usos más allá de la identificación por lo que necesitan que los eventos sigan un orden, como en las operaciones realizadas sobre la raíz de confianza ya que tienen que ser aprobadas de alguna manera por los nodos que componen la red.

3.1.3 Raíz auto certificada

Por último tenemos a la raíz auto certificada, este sistema de confianza está basado en la generación de un número aleatorio y el uso de operaciones criptográficas. Un DID puede ser generado en un principio sin la necesidad de comunicarse con ningún actor externo al propio controlador, es decir un DID se puede generar dentro de una wallet que actúe como controlador sin necesidad de que contacte con nadie. El mayor inconveniente de este tipo de confianza es que la seguridad de la raíz está asociada directamente al hardware donde es generada es por eso que la seguridad de estas raíces recae en el hardware usado para crearlas, las especificaciones y protocolos que siga las certificaciones que cumpla …

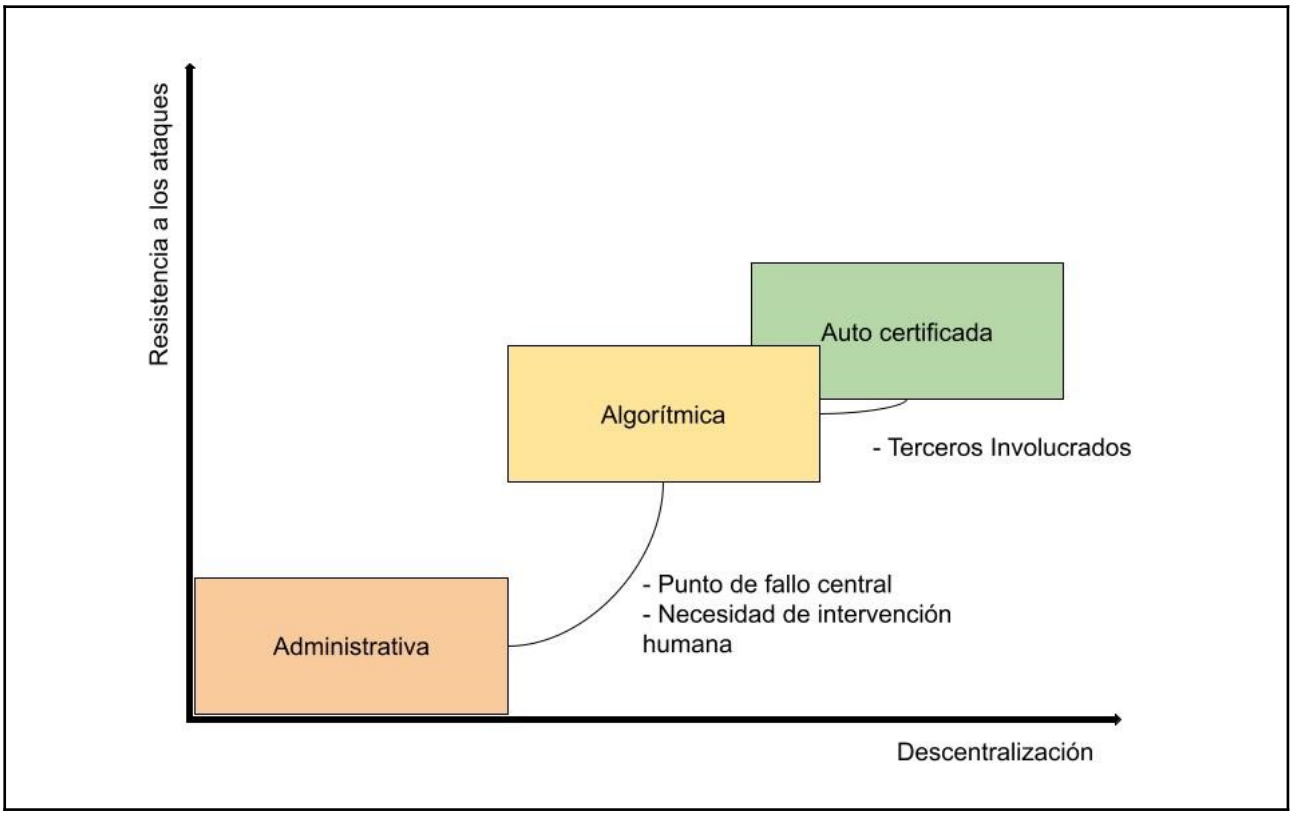

Figura 3.3: Comparativa de los diferentes tipos de raíz de confianza

3.2 KERL y KEL

En KERI se tienen dos logs de eventos, los cuales veremos en el próximo apartado. Dichos logs de eventos son el KERL (Key Event Receipt Log) y el KEL (Key Event Log) [28]. Estos logs son muy importantes ya que forman parte de la raíz de confianza de un identificador. Son usados para establecer la propiedad sobre un identificador, verificar las acciones que se realizan etc.

3.3 KEL

El log de evento (Key Event Log) es una cadena de eventos ordenada generada por el controlador sobre los eventos realizados con un identificador. Los eventos se encuentran relacionados unos con otros mediante resúmenes (explicación en el próximo apartado), los cuales contienen la información de su evento previo. Este log incluye todos los eventos, no importa del tipo que sean. La utilidad de este log de eventos se ve reflejada en la verificación del control sobre un identificador que necesita hacer un validador. Un KEL es parte de la raíz de confianza del identificador ya que en él se encuentra toda su historia desde su creación.

3.4 KERL

El log de recibos de evento (Key Event Receipt Log) es un KEL en el cual se encuentran los recibos enviados por un testigo (o verificador en el caso del modo directo como ya veremos). Los testigos mantienen un KERL diferente para cada uno de los identificadores sobre los que observan. Los KERL contienen una serie de eventos clave, los cuales están enlazados entre sí criptográficamente mediante el resumen de eventos. Un evento que indica la creación del identificador es el primer enlace, si un testigo recibe un evento sobre el cual no pueda verificar la cadena hasta el evento de creación lo descarta. Al igual que el KEL el KERL forma parte de la raíz de confianza de un identificador. Un validador que sea a la vez testigo puede verificar que el controlador de un identificador es el dueño real gracias a el KERL. En la próxima sección todos estos conceptos se aclaran en las explicaciones del resumen de eventos y el funcionamiento de la pre rotación.

3.5 Eventos de KERI

En KERI un evento es un mensaje asociado con el protocolo [28]. Comúnmente suelen estar relacionados con acciones asociadas a un identificador, creación, uso, actualización, etc. Los eventos en KERI tienen que ser verificables y para ello KERI dispone de una serie de mecanismos entre los cuales está la pre rotación de claves. Para resumir todos los eventos son procesados por un algoritmo generando un hash y firmando ese hash. De esta manera se pueden tener evidencias criptográficas de las acciones que se han realizado.

3.5.1 Resumen de eventos

El resumen de un evento en KERI es su transformación a un hash criptográficamente verificable. Dicho resumen contiene la función de hash que se ha utilizado para generarlo junto con la información de su proceso de derivación [33]. Este resumen da como resultado la creación de una cadena de caracteres única la cual contiene la información del evento.

3.5.2 Eventos de clave

En KERI un evento de clave es aquel en el que se ve asociado un identificador y sigue un orden establecido por el controlador. Los eventos de clave son usados para manejar las claves del identificador o para usarlas. Para verificar la validez de dichos eventos se incluyen firmas generadas con las claves del identificador, la representación de un evento de clave es la siguiente:

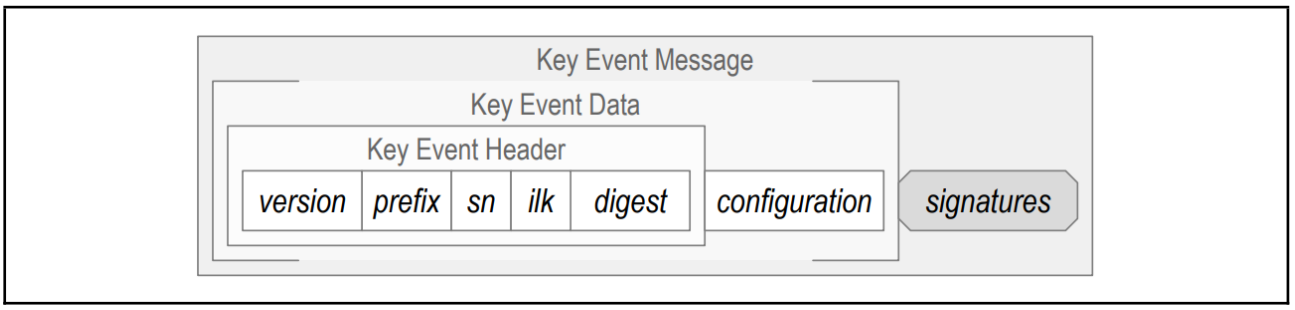

Figura 3.4: Cabecera de un evento de clave
(Fuente:https://github.com/SmithSamuelM/Papers/blob/master/whitepapers/KERI_WP_2.x.web.pdf)

3.5.3 Establecimiento

Un evento de establecimiento es un evento de clave usado para declarar el actual par de claves de control del identificador. La utilidad que proporciona este evento es la de declarar quién es el controlador del identificador. Se suele usar en la creación de las claves asociadas al identificador o en su rotación, esto junto con los datos presentes en la configuración permiten mantener el control sobre el identificador. Además como se puede observar se hace un resumen de la siguiente clave, esto es uno de los pasos claves para poder entender la pre rotación de claves en KERI.

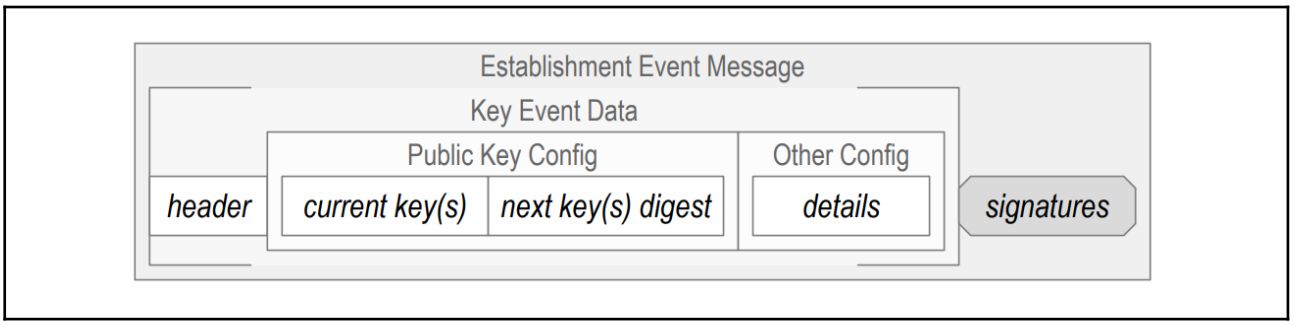

Figura 3.5: Cabecera de un evento de establecimiento
(Fuente:https://github.com/SmithSamuelM/Papers/blob/master/whitepapers/KERI_WP_2.x.web.pdf)

3.5.4 Origen (icp)

Un evento de origen es un evento de establecimiento por lo que a su vez es un evento de clave. Representa la creación de un identificador. En él se encuentran una derivación de su par de claves usado para su creación además del resumen de su próximo par de claves. Si no se añade un posible próximo par de claves se considera un identificador estático por lo cual no podrá ser transferido. Por último se encuentra una configuración la cual es útil para la estructura donde se vaya a utilizar. El evento de origen se ha de realizar una única vez durante la creación del identificador y es el precursor al primer evento de establecimiento.

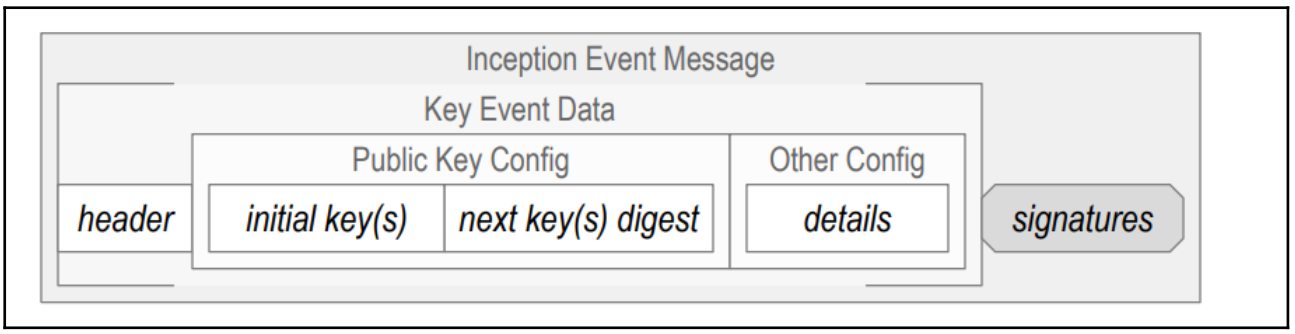

Figura 3.6: Cabecera de un evento de origen
(Fuente:https://github.com/SmithSamuelM/Papers/blob/master/whitepapers/KERI_WP_2.x.web.pdf)

3.5.5 Rotación (rot)

Un evento de rotación es considerado un evento de establecimiento. Representa la operación de rotar un par de claves, desechando las actuales claves de control para darle el control sobre el identificador a un par nuevo, el par de claves de rotación. Esta rotación es la base del sistema de pre rotación que usa KERI. Cada evento de rotación se encarga también de asegurar y generar el próximo par de claves de rotación. En el momento en el que el próximo par de claves de rotación sea nulo es cuando el identificador deja de ser transferible.

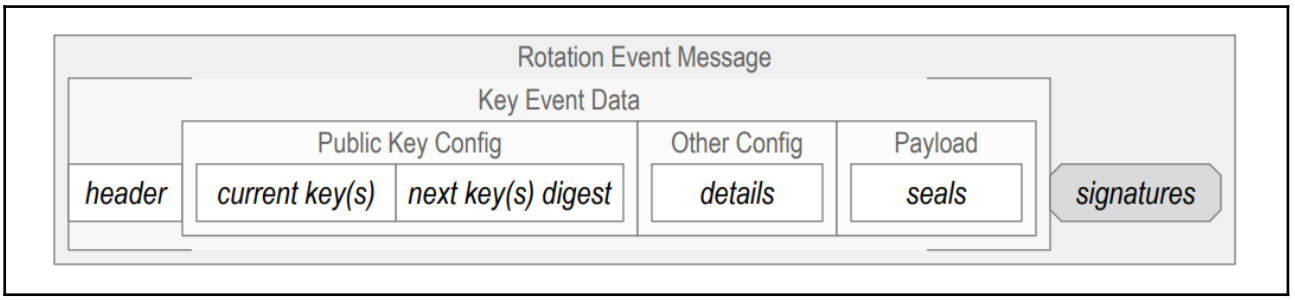

Figura 3.7: Cabecera de un evento de rotación
(Fuente:https://github.com/SmithSamuelM/Papers/blob/master/whitepapers/KERI_WP_2.x.web.pdf)

Pre Rotación

El sistema de pre rotación que utiliza KERI es lo que le proporciona un alto nivel de seguridad. Este sistema funciona usando dos pares de claves, un par de claves de control y un par de claves de rotación. El par de claves de control trabaja con el identificador para realizar operaciones. El par de claves de rotación, el cual se encuentra oculto y nunca sale del poder del controlador, se utiliza para establecer el próximo par de claves. Cuando se roten las claves públicas el par de rotación es uno de los contenidos que se encuentran en el resumen del siguiente par de claves.

Cuando se realiza la operación de rotación se desecha el par de claves de control actual y se sustituye por el par de claves de rotación que permanecían ocultas. De esta manera se genera un nuevo par de claves de rotación, añadiendo su clave pública a un resumen. Este sistema recuerda al de una cadena en la que los pares de claves se encuentran enlazados criptográficamente gracias a los resúmenes. Para verificar la validez de un par de claves de control se hace una verificación sobre el resumen, el cual contendrá la clave usada para generar su anterior versión, esto se ve mejor de manera gráfica:

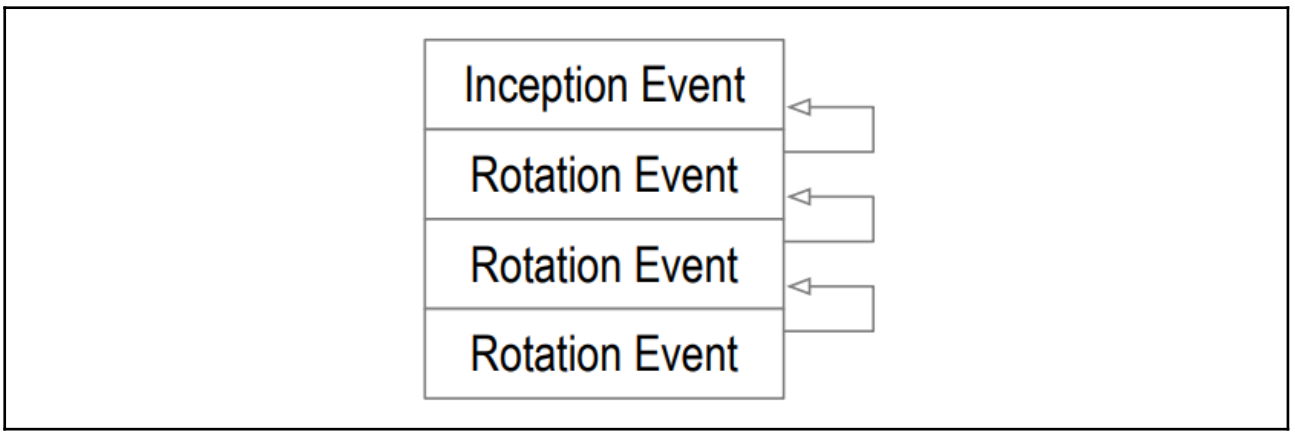

Figura 3.8: Estructura de rotación de eventos
(Fuente:https://github.com/SmithSamuelM/Papers/blob/master/whitepapers/KERI_WP_2.x.web.pdf)

Como se puede ver es posible derivar después del último evento de rotación el evento de origen gracias a los resúmenes. De esta manera se verifica la cadena si se llega al evento de origen y concuerda con el establecido.

3.5.6 Interacción (ixn)

Un evento de interacción es lo que se llama en KERI como eventos no establecidos. Son eventos que no declaran una pre rotación de claves y solo usan las claves de control actuales. En su carga lleva un sello el cual confirma quien es el controlador actual, además de un resumen del evento que lo precede. Gracias a este resumen puede resolverse hasta un evento de rotación sobre el cual confirmar la veracidad del par de claves de control actuales. Estos eventos pueden ocurrir múltiples veces entre eventos de rotación pero esto no quiere decir que entre 2 eventos de rotación tengan que ocurrir obligatoriamente eventos de interacción.

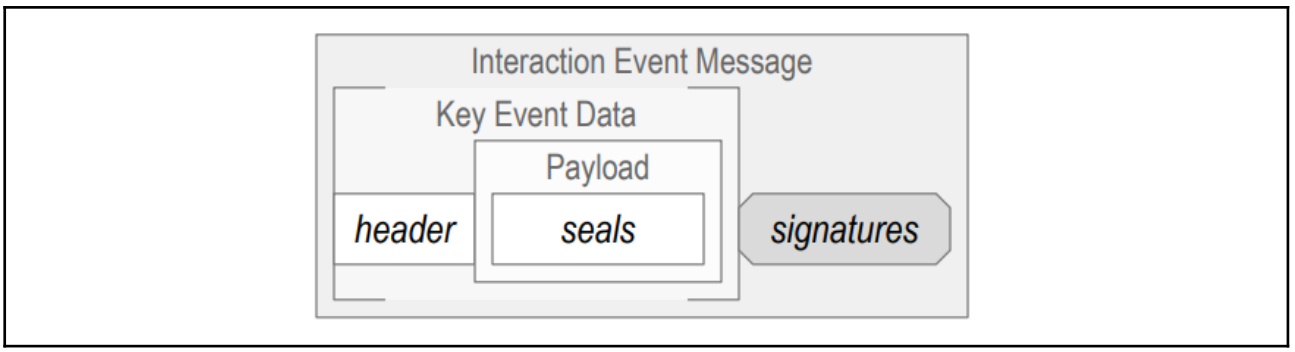

Figura 3.9: Cabecera de evento de interacción
(Fuente:https://github.com/SmithSamuelM/Papers/blob/master/whitepapers/KERI_WP_2.x.web.pdf)

3.5.7 Recibo (rct)

Un evento de recibo es un tipo de evento especial el cual contiene información sobre los demás eventos que se han realizado. Este evento va firmado por los testigos, los validadores o los controladores. En un evento de recibo se incluye la información del evento en sí mismo o de una referencia a los eventos previos. La etiqueta de testigo hace referencia al validador de dicho testigo (como veremos más adelante el validador y el testigo pueden ser la misma entidad) . La función que cumple un evento de recibo es la de verificar que un testigo ha observado un evento asociado a una clave. Es decir su uso es para tener constancia de los eventos del identificador que está siendo observado. Dichos

recibos son enviados al identificador donde se almacenarán en su KERL como confirmación de las acciones realizadas.

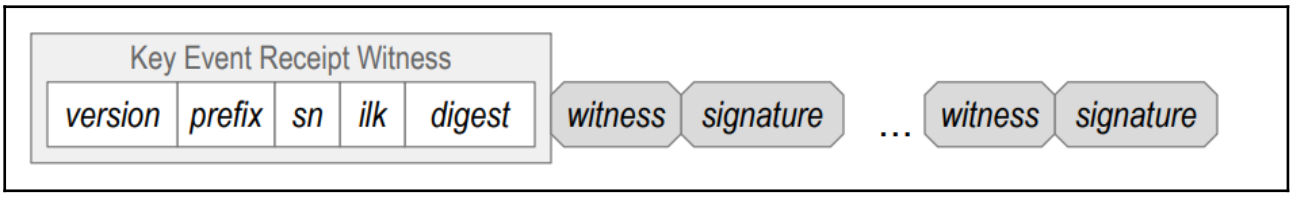

Figura 3.10: Cabecera de un evento de recibo
(Fuente:https://github.com/SmithSamuelM/Papers/blob/master/whitepapers/KERI_WP_2.x.web.pdf)

Estos son los eventos que considero más importantes entender para poder comprender el funcionamiento de KERI, ya que en realidad hay muchos más eventos que no están explicados aquí por la poca relevancia que aportan al objetivo de este trabajo. Algunos de esos eventos son eventos de delegación, rotación extendida, recibo múltiple … Para más información acerca de estos eventos revisar el trabajo oficial de KERI [28]

3.6 Método de respuesta directo e indirecto

Recordemos que en KERI la rotación de claves es un mecanismo que nos sirve para mantener el control sobre un identificador. Este control se mantiene durante un periodo de tiempo mediante la validación del identificador ante un validador. El validador puede llegar a detectar eventos de duplicidad sobre un identificador, es ahí cuando en KERI se usan los mecanismos de repetición de la secuencia de eventos previamente explicados. Para revisar estas secuencias de eventos existen dos modos de funcionamiento, el modo de respuestas directo y el modo de respuestas indirecto [28] los cuales se explican a continuación.

3.6.1 Método de respuesta directo

En el método de respuestas directo el controlador no puede realizar las verificaciones si no se encuentra conectado con el validador. Esto significa también que para que un validador pueda acceder al log de eventos de un identificador, es necesario que el controlador del identificador se encuentre conectado. Debido a esta característica este modo es ideal para comunicaciones entre dos entidades, una de ellas cumpliendo con el papel de controlador, mientras que la otra actúa como validador.

Para empezar la relación, lo primero que tiene que realizar el validador es una comprobación de que el controlador del identificador es quien dice ser. Dicha comprobación se consigue mediante el envío de un evento de origen por parte del controlador firmado con su correspondiente clave privada. Desde la perspectiva del validador el evento de origen establece al controlador como el propietario del identificador que se usará en la relación. Esta declaración de propiedad por parte del controlador se mantiene mediante el envío, desde el controlador al validador, de los eventos de rotación de claves que genere.

El controlador establece el orden de los eventos de claves ya que solo el controlador crea dichos eventos. La razón es que el controlador es su propia raíz de confianza y no depende de ninguna otra fuente de confianza.

El controlador establece entonces una secuencia de eventos incluyendo todos los eventos posteriores al de origen. Para verificar la validez de dichos eventos se hace un resumen de los mismos permitiendo verificar su validez cuando se hace la rotación de

claves. Cada evento tiene un número de secuencia incremental el cual ayuda a identificarlos. Junto con el resumen criptográfico y el número de secuencia se puede crear una cadena de eventos inmutable que se verifique mediante la validez de las operaciones creadas por el controlador. Eso se vio previamente en la explicación de los resúmenes de eventos y la rotación de claves.

Para poder verificar la validez del par de claves de control actual del controlador, el validador necesita acceso al log de eventos donde se guardan los eventos del controlador, es decir el validador necesita acceso al KEL del controlador. De esta manera el validador puede verificar que el controlador es quien dice ser, siempre y cuando mantenga una copia del último resumen que realizó el controlador.

En el evento de origen con el cual el controlador se comunica con el validador es necesario que el validador tenga una copia completa del KEL del controlador hasta el momento del evento de origen. Una vez recibida la copia es capaz de verificar otras copias del KEL del controlador siempre que el verificador tenga el último resumen del controlador.

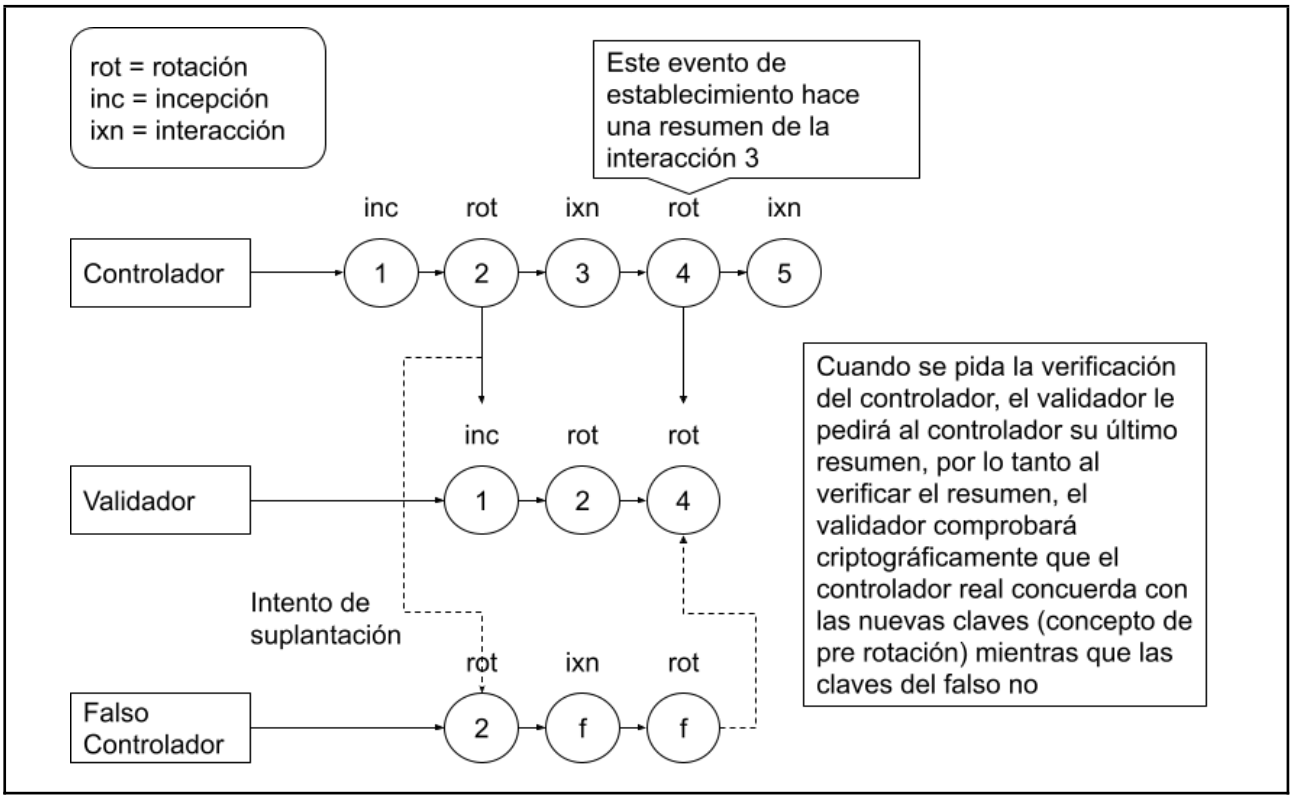

Figura 3.11: Estructura de respuesta método directo

Ya que cada evento de rotación se ha de comunicar entre el controlador y el validador, si en algún momento se realiza una interacción y alguno de los 2 no se encuentra disponible se paraliza dicha interacción hasta que se puedan comunicar.

Cuando el verificador recibe un evento envía un recibo de confirmación de dicho evento al controlador. Este recibo contiene una firma asociada al evento. El controlador tiene ahora un recibo que verifica el evento, dicho recibo se almacena en el KERL, al fin y al cabo, el validador es para el controlador un testigo de las acciones que realiza.

Esta relación uno a uno convierte al controlador y verificador en controlador y verificador simultáneamente, permitiendo que los dos realicen ambos papeles en su relación.

El controlador puede crear un identificador único para usar con el verificador. Generando un evento de origen que sería el que se transmite como evento al verificador. Cada relación uno a uno puede crear un nuevo par de identificadores y pares de claves asociadas a cada miembro de la relación.

3.6.2 Método de respuesta indirecto

En el modo de respuesta indirecto la transmisión de un evento a un validador desde un controlador puede ocurrir sin que el controlador se encuentre disponible, por tanto no pudiendo comunicarse de manera directa con el validador. Esto permite que se puedan crear relaciones de orden superior a las de pares ya que no existe la necesidad de que todos los miembros de la relación se encuentren conectados en el momento de la comunicación. Esta característica habilita modelos de comunicación uno a muchos, muchos a muchos e incluso uno a uno que necesiten disponer de la información del otro en todo momento. El problema que surge con este tipo de funcionamiento es el de tener que manejar una estructura de confianza compuesta por múltiples testigos. Al tener múltiples testigos pueden detectarse casos de duplicidad o casos de múltiple modificación. Para entender mejor cómo es el funcionamiento indirecto, la siguiente imagen lo ilustra de manera simple.

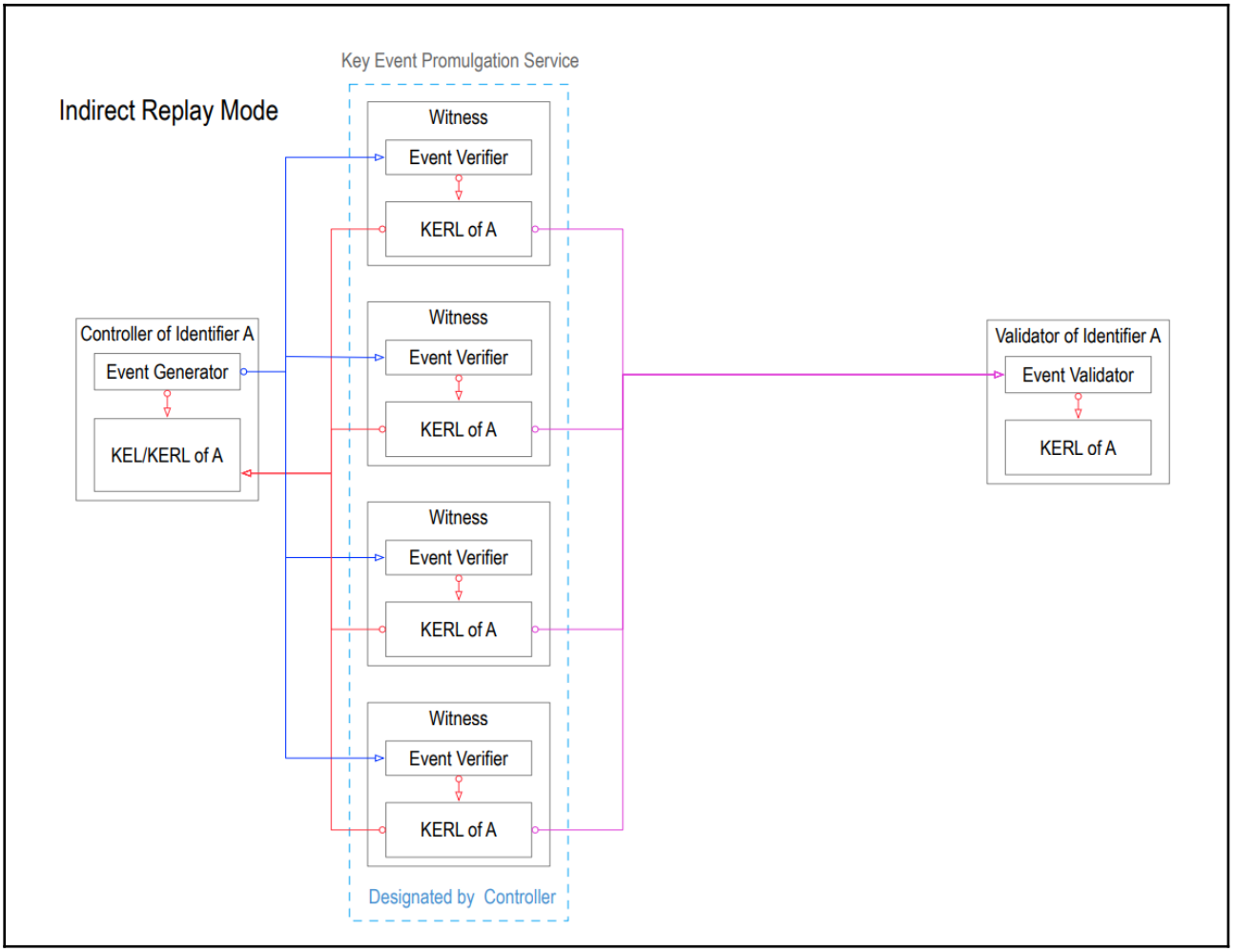

Figura 3.12: Estructura de respuesta método indirecto
(Fuente:https://github.com/SmithSamuelM/Papers/blob/master/whitepapers/KERI_WP_2.x.web.pdf)

Como se puede observar la mayor diferencia que se tiene con el modelo directo es que la comunicación no ocurre de manera directa, es mediante una red de testigos sobre la

cual se transmiten los eventos.

Este modo de respuesta se encuentra definido con mucha mayor profundidad en el paper de KERI [28], pero el objetivo de esta investigación es buscar un método DID para KERI, dicho método se complica si se buscara implementar el método de respuesta indirecto es por eso que se decide usar el modo de respuesta directo, un método el cual está definido y sigue unos estándares sencillos de definir.

Capítulo 4 Método DID para KERI

Una vez que ya se tiene el concepto de que es un DID, sus métodos y la infraestructura que representa KERI, se puede vislumbrar que es lo que se pretende al buscar un método DID para KERI. El objetivo de este trabajo es la investigación sobre el desarrollo de un método DID que use la infraestructura de manejo de claves y validación que propone KERI. Se busca comunicar dos entidades mediante el uso del modo de respuesta Directo que ofrece KERI. Recordemos que su modo de respuesta indirecto es otro modelo de funcionamiento orientado a modelos de comunicación más complejos. En dichos modelos las relaciones son de más de dos individuos, invalidando así el objetivo de la comunicación uno a uno.

4.1 Análisis de los métodos DID actuales

El primer paso a realizar para buscar este DID de KERI es el de investigar los métodos que existen actualmente [15] y ver si hay alguno que acerque su funcionamiento a lo que buscamos.

Para dicha búsqueda he revisado todos los métodos registrados por la W3C (87) a fecha de febrero de 2021, utilizando como parámetros para su validez:

- Última actualización del método
- Uso de una DLT (recordemos que en el modo directo son comunicaciones uno a uno)
- Estado de las operaciones CRUD del método
- Además de una serie de observaciones como popularidad de la comunidad, idioma de desarrollo, equipo o fundación detrás del desarrollo, propiedad del método ...

Nombre	**DLT o Networks**	**CRUD**	**Observaciones**
life	RChain	NO	Abandonado
sov	Sovrin	SI	No usa blockchain pero está enfocada al sector privado.
v1	Veres One	Parcial	Solo contiene Create y Update, no se actualiza 2019
com	Commercio.network	Parcial	Falta Delete, no se actualiza 2019
ont	Ontology	SI	Antiguo (2018), seguridad y privacidad pendientes
ion	Bitcoin	--	Revisar con mayor profundidad, no

			sigue esquema de la w3c, pero tiene bastante fama
jolo	Ethereum	SI	
bryk	bryk	Si	Documentación bastante completa y poco famoso
peer	peer	SI	Avalado por la SSI Fundation, muy completo, menciona el uso de KERI alejándose del uso de blockchain. https://identity.foundation/peer-did-method-spec/index.html
git	DID Specification	SI	Parece bastante completo pero antiguo (2019)
schema	Multiple storage networks, currently public IPFS and evan.network IPFS	SI	Usa IPFS
key	Ledger independent DID method based on public/private key pairs	Parcial	El key method está diseñado para acciones que no necesitan ser registradas, updateadas o desactivadas, como puede ser identificarte. Por esto no soporta los Update y Desactivate de CRUD
corda	Corda	SI	El concepto se puede quizás extrapolar fuera de la Corda Network que mencionan es necesaria para su funcionamiento
uns	Uns.network	SI	

Tabla 4.1: Fragmentos de la tabla utilizada para el análisis de los métodos DID

Una vez se realizó la investigación de todos los métodos se llegó a una conclusión clara. Primero es que la mayoría de métodos están orientados a resolverse mediante el uso de una DLT, lo cual para la implementación de KERI en modo directo no es útil. Debido a esto pocos métodos cumplían con las características dejando solo dos métodos DID que eran prometedores para el modelo de funcionamiento de KERI. Dichos métodos son:

- El DID Key [16] está diseñado para acciones de un solo uso que no necesitan ser registradas, actualizadas o desactivadas. En un principio pinta muy bien para su implementación con la estructura de KERI. Tratándose de un DID muy simple diseñado para comunicaciones efímeras no soporta los Update y Deactivate de CRUD. Esta característica bloquea el uso de rotación de claves que plantea KERI. El uso del DID Key está enfocado a relaciones cortas que necesiten una verificación momentánea y no a mantener relaciones de confianza duraderas.
- El DID Peer [17], como su nombre indica es un DID enfocado a comunicaciones Peer-To-Peer. Es un método que está diseñado para funcionar sin depender de una DLT además de soportar diferentes modos de funcionamiento. Un modo de

funcionamiento público, un modo de funcionamiento entre una pareja y un modo de funcionamiento entre N entidades. Además de los modos de funcionamiento que posee tiene bien definidas sus operaciones CRUD y es un método moderno que está en continuo desarrollo.

4.2 DID Peer

En esta sección voy a comentar en mayor profundidad todo lo relacionado con el DID Peer [18], desde los beneficios que aporta hasta su estructura, finalizando con un análisis centrado en la seguridad del método. Recordemos que la aplicación que se busca con la unión de los DID y KERI está orientada a su uso en conexiones que pueden llevar una carga de datos sensibles PII (Personally Identifiable Information).

4.2.1 Introducción al DID Peer

La premisa de este método es la de mantener las comunicaciones entre las dos entidades que las realizan. Si la comunicación es entre Alice y Bob que se resuelva entre Alice y Bob. Esto significa que puede mover las interacciones necesarias para llevar a cabo dicha comunicación fuera de una blockchain pero dejando la opción de conectarse a una si fuera necesario. De esta manera se permite a un usuario tener el control total de las relaciones digitales que lleve a cabo y cómo las lleva a cabo. Dichas relaciones se hacen entre diferentes entidades que llamaremos peers.

El uso del DID Peer está orientado a tres maneras de funcionamiento:

- Con un número desconocido de peers, orientado a uso público.
- Con un par de peers, orientado a relaciones Alice Bob.
- Con N número de peers, orientado a relaciones entre varios peers que comparten intereses, por ejemplo un consorcio de 3 empresas.

4.2.2 Documento DID

El peer DID opta por el uso de documentos en formato JSON-LD [19], cumpliendo con todas las características que recomienda la W3C para estos documentos. No añade ninguna funcionalidad extra haciéndolo fácil de comprender y usar por otros métodos si fuera necesario.

```
{
  "@context": "https://w3id.org/did/v1",
  "id": "did:peer:2.Ez6MkpTHR8VNsBxYAAW...",
  "authentication": [{
     "id": "did:peer:2.Ez6MkpTHR8VNsBxYAAW...=#Vz6MkpTHR8V...",
     "type": "JsonWebKey2020",
     "controller": "did:peer:2.Ez6MkpTHR8VNsBxYAAW...",
     "publicKeyJwk": {
          "kty": "EC",
          "crv": "P-256",
          "x": "38M1FDts7Oea7urmseiugGW7tWc3mLpJh6rKe7xINZ8",
          "y": "nDQW6XZ7b_u2Sy9slofYLlG03sOEoug3I0aAPQ0exs4"
    }
```

```
  }],
  "keyAgreement": [{
     "id": "did:peer:2.Ez6MkpTHR8VNsBxYAAW...=#EzXwpBnMdCm...",
     "type": "X25519KeyAgreementKey2019",
     "controller": "did:peer:2.Ez6MkpTHR8VNsBxYAAW…”,
     "publicKeyBase58": "JhNWeSVLMYccCk7iopQW4guaSJTojqpMEELgSLhKwRr"
  }],
  "service": [{
    "id": "did:peer:2.Ez6MkpTHR8VNsBxYAAW...#didcommmessaging",
    "type": "didcommmessaging",
    "serviceEndpoint": "https://example.com/endpoint",
    "routingKeys": ["did:example:somemediator#somekey"]
  }]
}
```

Figura 4.1: Documento DID de un Peer DID en formato JSON-LD (fuente:https://identity.foundation/peer-did-method-spec/#diddocs)

Modificaciones en el documento DID

Cada vez que un documento DID cambia [20] es necesario guardar dicho cambio para poder verificar que el cambio se hizo y autorizó en su momento en posteriores versiones. Los creadores del peer DID han solucionado este problema con el uso de Deltas en los cuales se van almacenando los cambios, quién los realizó y cuando los realizó aproximadamente. Gracias a los Deltas es posible hacer un seguimiento de los cambios de un documento.

```
{
    "change": <base64url encoding of a change fragment>,
    "by": [ {"key": <id of key>, "sig": <signature value>} ... ],
    "when": <ISO8601/RFC3339 UTC timestamp with at least second
precision>
}
```

Figura 4.2: Delta de un Peer DID (fuente:https://identity.foundation/peer-did-method-spec/#backing-storage)

Los deltas son únicos siendo diferenciados por el hash firmado en el parámetro “change”. Aunque dos deltas representen la misma modificación (por ejemplo se borra algo, se vuelve a añadir y después de vuelve a borrar), los hashes de estos deltas han de ser diferentes. Gracias a esta característica se evita que dos peers que hayan realizado la misma operación con los mismos parámetros en momentos diferentes no la consideren la misma operación. Si los dos tuvieran el mismo hash, ambos consideran que tienen el documento actualizado con respecto al otro cuando pueden haber pasado x operaciones intermedias.

En conclusión la resolución de un documento DID es la acumulación de todos los deltas hasta el punto de su resolución añadiendo el DID.

Manejo de múltiples modificaciones

Recordemos que el peer DID está enfocado para conexiones entre múltiples nodos. Para prevenir múltiples modificaciones al mismo tiempo sobre un documento por múltiples peers, los creadores almacenan las modificaciones sobre el documento mediante un

sistema de CRDT (Conflict-free Replicated Data Type) [21]. Este sistema de replicamiento de la información necesita que se cumplan una serie de características. Es necesario que todos las claves, reglas y servicios del documento DID estén representados por un único id, además estos datos han de ser inmutables. En el caso de querer cambiarlos tendrán que ser eliminados y regenerados, al borrar dichos datos sus ids son añadidos a una lista de parámetros borrados. Estas restricciones permiten gestionar los problemas de múltiple modificación. Si dos peers modifican sus claves sobre el mismo identificador el validador es capaz de gestionar las dos claves como válidas hasta que se vuelvan a sincronizar. De esta manera se permiten acciones de añadir claves, reglas y servicios de manera descoordinada evitando conflictos. El problema ocurre cuando se delega el control a otros peers y se envían acciones contradictorias. Para resolver este caso se opta por una mecanismo de el primero que llegue es el que se aplica. Aun así se verifican acciones futuras las cuales puedan influir sobre los cambios que llegarán después. Un ejemplo puede ser que el peer A1 niega al peer A2 y el peer A2 niega al peer A1. El verificador B esperará a acciones futuras, por ejemplo el peer A3 requiriendo algo del peer A2 deshabilitará las acciones aplicadas por el peer A1 y por tanto aplicará las del peer A2.

Este sistema es muy complejo y se escapa al alcance de este proyecto es por eso que aquí se da una breve explicación del mismo. Si se quisiera profundizar más en el mismo existe extensa documentación acerca de él ya que es usado por múltiples aplicaciones que requieren la interacción de varios usuarios sobre una misma fuente de datos. Un ejemplo de uso es el de varios usuarios modificando un documento en Google Docs.

4.2.3 Operaciones CRUD

Las operaciones CRUD de Peer [22] se establecen mediante el envío de mensajes, a continuación se verá el funcionamiento esperado para cada operación.

Create: Crear un DID no necesita de ningún tipo de conexiones externas, es en el registro del documento frente a un peer externo cuando se necesita el uso de un protocolo. Para solventar este problema se plantean dos maneras, una utilizando protocolos clásicos como HTTPS, la cual no es del todo segura y complica el esquema. La otra manera es utilizar la librería de comunicación para DID, la librería DIDComm [23], la cual funciona de la siguiente manera:

1. Uno de los peers manda una invitación de conexión sin cifrar, permitiendo al otro peer ver qué características necesita y las pueda adquirir. En el caso que se quisiera hacer esto de manera privada, (evitar ataques MITM) se tendrían que utilizar canales externos como sms, emails … además de cifrar la invitación.
2. El peer invitado entonces envía una petición de conexión encriptada. En dicha invitación se incluye información acerca del momento de la creación del documento, la razón de esto es poder probar la propiedad sobre el identificador.
3. Finalmente el peer que envió la invitación responde a la petición de conexión con una aprobación encriptada. La aprobación contiene la confirmación de que recibió la información sobre la creación del documento del invitado.

Esto sin embargo sólo proporciona una conexión segura. Para que exista confianza sobre dicha comunicación se deben de realizar intercambios de credenciales verificables al inicio de la conexión.

Resolve: Debido al modelo de funcionamiento que siguen los DID peer, la operación de resolver la puede realizar el peer en local o como un peer externo a través de una conexión previamente establecida. Para resolver el documento en local no hay problema,

simplemente es mirar los deltas que se poseen. Cuando resuelve un peer externo es cuando surgen problemas. Para ello los creadores del peer DID han puesto una serie de restricciones para que un peer externo pueda resolver el documento de otro peer:

1. Solo peers pueden resolver documentos de peers, es decir si una petición de resolución no viene de un peer se descarta.
2. Un peer solo puede resolver un documento si conoce los datos sobre el momento de creación del mismo. Por lo que un peer que no conozca los datos de creación de un documento no podrá pedir su resolución a otro.

Una vez comprobadas estas dos restricciones podemos aprovechar la ya mencionada anteriormente DIDComm para proceder con la comunicación. Empezando con el envío de un mensaje de resolución entre los peers y finalizando con la respuesta en la cual se incluye el documento y los datos que ayuden a identificar la operación.

Update: Los updates los puede realizar un peer sobre su DID utilizando como testigo a cualquier peer que conozca el DID sobre el que se quiere realizar la update. Los creadores del método ofrece una solución con métodos convencionales para realizar la acción, la cual necesita resolver una serie de problemas:

1. Problemas en el orden de las actualizaciones.
2. Problemas para probar que la update es correcta sobre el DID.
3. Problemas para conseguir consistencia en las actualizaciones.
4. Problemas para detectar imperfecciones o inconsistencias sobre los participantes conectados.
5. Problema al asumir que la comunicación de un update ha sido aceptada por todos los nodos dependientes de quien la emitió.

El mayor problema con esta implementación es este último punto ya que obliga a centralizar los documentos sobre un peer y que los nodos dependan directamente del mismo. Centralizando a su vez la gestión del documento sobre un nodo y perdiendo por tanto la característica de la descentralización.

Es por eso que la implementación de DIDComm vuelve a ser recomendada. Gracias a la implementación de un estado de sincronización, los diferentes peers pueden mandar mensajes entre ellos en lo conocido como chismorreo (gossip). Gracias a esto se consigue que los diferentes peers comparen el estado del documento por la información incluida en el chismorreo, evaluando si fuera necesario realizar una update del mismo. El funcionamiento simplificado del chismorreo consiste en, si un peer recibe un chisme y el tiene el mismo estado que el descrito en el chisme, envía una confirmación sobre la información. En caso de que sea algo nuevo para él aplica los cambios que se describen en el chisme y envía una confirmación. Por último en caso de que tenga un estado más adelantado al del chisme comunica acerca de la información nueva que tiene al peer que le envió el chisme mediante el envío de un estado de sincronización.

Deactivate: Para desactivar el documento se podría simplemente dejar de comunicar con los peers que lo necesiten y borrarlo. Es recomendable anunciarlo para que todos los peers que tengan ese documento puedan eliminar los datos referentes al mismo. Este anuncio se puede realizar mediante DIDComm. Mediante el uso de DIDComm se da la opción a los receptores de responder con la confirmación de la desactivación del DID.

La conclusión directa que se puede sacar de cómo funcionan las operaciones CRUD en el peer DID es mediante el uso de DIDComm. Es verdad que los creadores de Peer ofrecen la alternativa de usar mecanismos convencionales para poder realizar dichas

operaciones. Estas alternativas necesitan cubrir muchos problemas, no siendo proporcional las complicaciones que supone implementarlos a los beneficios que tiene usar DIDComm. Para analizar en mayor profundidad cómo utilizar DIDComm y muchas como implementar muchas de las características que necesita cumplir los DIDs la referencias de Hyper Ledger Aries [24] puede ser de mucha utilidad.

4.2.4 Seguridad

En cuanto a la seguridad [25] que ofrece dicho método el mayor problema se produce en las conexiones iniciales TOFU (Trust On First Use). Al ser la primera conexión se puede producir el conocido ataque MITM (Man In The Middle), dicho ataque se puede dar de dos maneras:

1. El atacante intercepta la petición del invitado y pretende ser el invitado. (simple)
2. Una vez hecho lo anterior el atacante puede proceder a establecer conexión con el que realizó la invitación real. Manejando así los mensajes que se envían de un extremo a otro. (complejo)

Los creadores del método afirman que el segundo caso, mucho más complejo y peligroso es casi imposible de realizar debido a la complejidad de manejar ambos extremos. El primer ataque puede pasar con normalidad y aunque el segundo sea muy difícil es mejor cubrirse las espaldas para el futuro. Para resolver este problema se proponen una serie de soluciones.

Canales de confianza de terceros

Esta suele ser la solución más común aplicada hoy día, los canales de terceros más usados para confirmar dicha conexión suelen ser:

- Llamadas telefónicas
- SMS
- Códigos por correo
- Envío de un randomart [26] (mejor opción)

El problema con estos métodos es que no están orientados a ser fáciles para el usuario y no suelen escalar bien a excepción del último de randomart, el cual en mi opinión es la mejor opción para usar en este caso. A continuación, se va a profundizar un poco más en dicho método.

Randomart

En este método se genera un randomart que se envía por correo seguro o se presentaría de manera física como un código QR. Dicha imagen puede ser comparada por el usuario (Alice), si no coincidiera con el generado por la supuesta entidad (Bob) se podría sospechar de un ataque MITM y por tanto se repetiría la conexión más tarde volviendo a comprobar. Este método además asegura conexiones futuras, ya que una vez que el randomart es confirmado se puede comprobar para conexiones futuras sin necesidad de ser enviado de nuevo.

Credenciales verificables

Este método consiste en una vez establecida la conexión, se busca una fuente de conocimiento común externa a la conexión (Charles). Esta fuente verifica que los dos comunicantes (Alice y Bob) son quien dicen ser. Presentado dicha validación solucionaría parcialmente el problema del MITM, pero el atacante podría manejar dichos datos. Para

ello la entidad que quiere verificarse (Alice) genera dicha verificación con los datos de la conexión que estableció con la fuente externa (Charles) que la verificó. De esta manera para poder verificar los datos de la conexión actual (Alice, Bob) deberán de coincidir con los datos conexión establecida con la identidad común que verificó a ambos (Alice, Charles) y (Bob, Charles).

4.3 Planteamiento del método DID para KERI

Para finalizar con el análisis del método Peer, se van a plantear las similitudes más importantes que comparte el método Peer con una posible implementación de KERI en su modo de funcionamiento directo.

Como se puede observar después de haber leído como está orientado el modelo de funcionamiento del método Peer para conexiones uno a uno y como está pensado KERI para funcionar en su modelo de respuesta directa, están claras las similitudes que comparten ambos. El método Peer ya define como llevar a cabo este modelo de comunicación para conexiones entre dos o más peers, además de como establecer las conexiones iniciales entre los peers. En el caso de implementar un modelo de funcionamiento que quiera asimilarse a Peer con las ventajas de KERI, no es necesario implementar los sistemas de replicación que nos ofrece Peer. Además recordemos que KERI ya de por sí define un sistema para la gestión de cambios mediante el uso de eventos, los cuales son almacenados en el KEL y KERL, por lo que también se prescindirá del uso de los deltas de Peer.

Las características principales que se quieren de Peer para implementar el método KERI son su gestión de la comunicación entre nodos, como se realizan las operaciones CRUD y como se establecen las conexiones iniciales.

De KERI se busca usar su sistema de logs, su sistema de pre rotación de claves y sus sistema para generar los identificadores, es decir se quiere un modelo que funcione internamente como KERI pero trabajé como lo haría Peer.

Capítulo 5 Prototipo básico del método DID

5.1 Rust como eleccion de desarrollo

Para el desarrollo del prototipo de un posible método KERI, se ha elegido como lenguaje de programación Rust [35], esto es debido a que Rust es un lenguaje de programación de alto nivel pero con la eficiencia de C + +. Esto lo consigue mediante una serie de normas que fuerzan al programador a llevar a cabo un buen uso de la memoria. Gracias a esta manera de programar es posible compilarlo a bajo nivel de manera efectiva. Esta capacidad de poder ser compilado a bajo nivel es una de las ventajas más grandes de Rust, ya que en un futuro se espera que esta tecnología se aplique a dispositivos de IOT (Internet Of Things), dichos dispositivos tienen hardwares diferentes por lo tanto si se puedo compilar a un bajo nivel el traslado del código a estos dispositivos resultará más sencillo.

Para poder empezar a trabajar con Rust [36] fue necesario aprender el funcionamiento del lenguaje, lo cual hice mediante el uso del manual oficial de Rust []. Además aunque no se usen en el prototipo realizado, por una serie de problemas que se discutirán más adelante, Rust sirvió para entender el funcionamiento de las librerías DIDComm [37], la cual se encarga de comunicar a los diferentes nodos usando los DIDs y la implementación del núcleo de Keri en Rust [38], esencial para el funcionamiento del método.

Dichas librerías definen los primeros paso a seguir para poder crear un método Keri funcional, además de las librerías previamente mencionadas era importante revisar en la DIDComm como se transmitirán los mensajes entre los diferentes nodos, para ello se tenía pensado usar JOSE [39] (Javascript Object Signing and Encryption), la cual tiene varias implementaciones en Rust.

5.2 Operación CRUD

Una vez visto el porqué del uso del lenguaje y cómo lo aprendí, empecé con el desarrollo de las operaciones CRUD que tendría que realizar un cliente sobre el agente KERI. Para ello se creó un prototipo de un Agente Keri sobre el cual se generan de manera local el DID y su Documento. A continuación se verán definidas las operaciones CRUD sobre las cuales tendría que actuar dicho agente.

5.2.1 Create

La operación de Create simula el evento de origen que produce KERI al generar un nuevo identificador. En esta simulación del evento se crea un agente al que se le pasa el

DID con el cual va a trabajar, el agente deberá de asociar a dicho nuevo DID su par de claves de control.

```
pub fn create() -> AgentKeri {
        let new_did: Did =
did!("did:keri:123123/www.identidad.com?nombre=pablo&apellidos=molina-m
artinez&correo=alu0101103181#id");

        AgentKeri {
            did : new_did,
            cont : None,
            clave : None
        }
}
```

Figura 5.1: Prototipo de Create

5.2.2 Resolve

La operación de Resolve pide al Agente los datos disponibles sobre el DID del que se quiere obtener el documento. La generación de dicho documento DID contiene toda la información de la que dispone el agente en el momento de la creación.

```
pub fn resolve (&self) -> doc::DidDocument {
        let mut new_doc = doc::DocumentBuilder::new(&self.did);
        if self.cont != None
        {
            new_doc =
new_doc.with_controller(self.cont.as_ref().unwrap());
            if self.clave != None{
                new_doc =
new_doc.add_verification_method(self.cont.as_ref().unwrap(),
                self.clave.as_ref().unwrap().0.clone(),
                self.clave.as_ref().unwrap().1.clone());
            }
        }
        new_doc.build()
}
```

Figura 5.2: Prototipo de Resolve

5.2.3 Update

La operación de Update actúa diferente dependiendo del tipo de evento que se quiera realizar, actualmente solo están implementadas los prototipos de 3 eventos, el evento de delegación, el de interacción y el más importante el de rotación de claves. Dichos eventos envían al agente los datos de la operación que se quiere realizar. En el caso del evento de rotación se envían al agente las claves actuales y se establecen unas nuevas.

```
pub fn update (&self, x: Type) -> AgentKeri {
        match x {
            Type::Delegacion(did) => {
                AgentKeri {
```

```
                    did : did!(&self.did.to_id_string()),
                    cont : Some(did!(&did)),
                    clave : None
                }
            }
            Type::Interaccion(did) => {
                AgentKeri {
                    did : did!(&self.did.to_id_string()),
                    cont : Some(did!(&did)),
                    clave : None
                }
            }
            Type::Rotacion(did, clave) => {
                AgentKeri {
                    did : did!(&self.did.to_id_string()),
                    cont : Some(did!(&did)),
                    clave : Some(clave)
                }
            }
        }
    }
}
```

Figura 5.3: Prototipo de un Update

5.2.4 Delete

La operación de Delete por como está definida en KERI es un evento de rotación sobre el cual no se establece una nueva clave para rotar,

```
pub fn delete (&self) -> AgentKeri {
        let vt = doc::VerificationType::Ed25519VerificationKey2018;
        let pk = doc::KeyFormat::Base58(String::from(" "));

self.update(Type::Rotacion(String::from("did:keri:none"),(vt,pk)))
}
```

Figura 5.4: Prototipo de un Delete

5.3 Problemas durante el desarrollo

Debido a la reciente creación del método, el principal problema que se encontró en el desarrollo del mismo fue el de un inesperado cambio sobre el código original en el que se basaba la empresa para la creación del agente con el cual se iba a crear el prototipo. Este cambio sobre el código original dejaba inservible el agente que habían creado en la empresa, es por ello que se tuvo que trabajar con una simulación de como funcionaria dicho agente y no se puedo llevar a cabo una implementación real del método, lo importante con el desarrollo del método era comprobar su funcionamiento lo cual se pudo hacer de forma parcial.

Capítulo 6 Conclusiones y líneas futuras

Como se ha podido ver este trabajo de fin de grado no es convencional ya que se centra en la investigación y sobre todo en la comprensión de un nuevo tipo de tecnología la cual cada vez promete más. Esto no son palabras vacías ya que la Unión Europea [41] ha mostrado interés en el método de respuesta directa que define KERI . Es un hecho que el próximo paso para la identidad es hacerla soberana, darle control total al usuario de sus datos y facilitar la gestión de los mismos, eso es algo que la Unión tiene claro. Recientemente han estado cerrando la brecha digital [40] que supone para un usuario el uso de todas estas nuevas tecnologías. Uno de los mayores obstáculos que tiene que sobrepasar KERI y las tecnologías de identidad soberana es el de la usabilidad. Es por eso que la empresa con la que he estado desarrollando este TFG ha pensado ya a futuro y previsto la necesidad de tener una buena usabilidad para que el usuario pueda usar estas tecnologías de la manera más natural posible. Claro que esto significa una serie de conceptos que no se han definido. En el caso de KERI por ejemplo conceptos tales como la rotación de clave suponen una barrera si se pretende que el usuario lo haga manualmente. Si se decide hacerlo automático es difícil determinar un momento para generar dicha rotación. Este es solo un ejemplo del nivel de abstracción que se necesita conseguir siendo de las cosas que quedan pendientes en el desarrollo de la tecnología.

Aun así, todavía queda tiempo para que tecnologías como KERI se encuentren a plena disposición del usuario. La mayor parte de KERI sigue desarrollándose, siendo implementado en múltiples lenguajes y todavía sin una implementación oficial que sea funcional. Este ha sido otro de los grandes retos con los que me he encontrado durante el desarrollo del trabajo. Partes del código de KERI en Rust, han sufrido cambios drásticos durante los últimos meses debido al rápido avance de su desarrollo.Dichos cambios han dejando inservibles implementaciones en desarrollo como la que se estaba realizando en Open. Es por eso que todavía las implementaciones completas de esta tecnología son realmente difíciles de desarrollar.

En cuanto a los próximos pasos a seguir con KERI y los DIDs, como ya mencioné anteriormente uno de las mayores barreras que existen es la de desarrollar una aplicación. Ya que el sistema de funcionamiento de la infraestructura ha de permitir al usuario promedio su uso sin mayores dificultades. Dejando de lado los problemas de usabilidad otra de las grandes tareas que queda en KERI es su desarrollo para el modo indirecto. Desarrollar un DID que pueda funcionar con un trasfondo de KERI y a la vez en múltiples DLTs es una tarea titánica por lo que todavía queda mucho que avanzar en ese apartado.

KERI es una tecnología que promete cambiar el mundo y cómo interactuamos con la red. Queda mucho por hacer y todavía hay cierta incertidumbre sobre este tipo de tecnologías. Siento que son trabajos como el que realizó los que poco a poco le dan visibilidad e intentan estandarizar lo que para mí y muchos es posiblemente el futuro de la

identidad en internet. Un futuro en el que se tiene control sobre los datos que representan al individuo y las relaciones que mantenemos con ellos.

Capítulo 7 Summary and Conclusions

As it's shown, this end of degree project isn't something convencional. It's a project focusing on the research and comprehension of a new technology that is getting more compelling with time. These affirmations aren't based on nothing, the European Union has seemed to take interest over the direct reply method that KERI defines. It's a fact that the next step for authentication is being sovereign, making the user have total control over his data and making it easy to manage them, that's something that the Union is aiming for. Recently they have been closing the digital breach that these new technologies mean to an user. One of the biggest obstacles that KERI and the self sovereign identity technologies need to overcome is the usability. That's why the company that I've been doing this project for has thinked ahead and predicted the necessity for this technology to be natural and easy going for a user. But this carries out a series of concepts that have not been defined. In the case of KERI, concepts like key rotation mean a barrier for users if they need to do it manually. If these concepts are made automatically it needs to be defined in a time and place for them to be done. This is just one example of the level of abstraction that's needed, being one of the topics that's pending to be worked on.

It's going to take some time until technologies like KERI can be at disposal for the user. The vast majority of KERI is still in development, being implemented in multiple languages, without having an oficial implementation that works. This has been another of the challenges that represent the making of this project. In the last months part of the KERI oficial code has suffered major changes, leading the code developed by Open in an unusable state. That's the main reason why full implementations of this technology are really hard to develop right now.

The nexts steps to follow for KERI and the DIDs, as I mentioned earlier, are the development of an application that's usable. The infrastructure needs to work in a way that a normal user can use without major difficulties. Leaving aside the usability problems, another of the major tasks that's pending, is the development of the indirect replay mode in KERI. Developing a DID that can work with the properties of KERI and with multiple DLTs it's a titanic task to do, right now it isn't nearly close to being done.

KERI is a technology aiming to change the world and how we interact with it. There's a lot of work to do and some insecurities around this technology. I feel like projects like mine give visibility to this new tech and try to standardize something that for me and many others is surely the future of the identiti in the net. A future where users have full control over their data and the relationships they carry on with them.

Capítulo 8 Presupuesto

Tipo	Descripción
Investigación	Costes sobre la investigación final y el planteamiento concreto sobre el desarrollo del método y su implementación.
Desarrollo del método	Costes del desarrollo del agente KERI y las funcionalidades necesarias para usarlo
Integración del método con la aplicación	Costes asociados a integrar el agente y métodos relacionados con KERI en una aplicación previamente diseñada por un equipo dedicado a la usabilidad.
Fase de pruebas	Costes asociados al mantenimiento y pago de beta testers para comprobar el funcionamiento de la aplicación a pequeña escala.
Formación	Costes dependientes de la formación necesaria para formar al equipo que vaya a trabajar en el proyecto.

Tabla 8.1: Tabla de tipos

Tipo	Personal	Tiempo	Valor	Coste
Investigación	Senior x 3	100 h	40€/h	4.000 x 3
	Junior x 2		20€/h	2.000 x 2
Total: 16.000				
Desarrollo del método	Senior x 3	150 h	40€/h	6.000 x 3
	Junior x 2		20€/h	3.000 x 2
Total: 24.000				
Integración del método con la app	Senior x 3	60 h	40€/h	2.400 x 3
	Junior x 2		20€/h	1.200 x 2

Total: 9.600				
Fase de pruebas	Testers x 100	30 días	Valor por uso 2€/d	Máx: 6.000 Min: Gratis
Total: 0 - 6000				
Formación	Senior x 3	50 h	40€/h	2.000 x 3
	Junior x 2		20€/h	1.000 x 2
Total: 8.000				

Tabla 8.2: Tabla de costes

Bibliografía.

Toda la documentación utilizada para el desarrollo del TFG se puede encontrar online en los siguientes enlaces.

[1] What is PKI? https://www.appviewx.com/education-center/pki/

[2] El blockchain generará un impacto de más de 20.000 millones en España en 2030 https://www.eleconomista.es/tecnologia/noticias/11151497/04/21/El-blockchain-generara-un-impacto-de-mas-de-20000-millones-en-Espana-en-2030.html

[3] Decentralized Identifiers (DIDs) v1.0 https://www.w3.org/TR/did-core/

[4] Decentralized Identifiers (DIDs) v1.0 DID-syntax https://www.w3.org/TR/did-core/#did-syntax

[5] Use Cases and Requirements for Decentralized Identifiers https://www.w3.org/TR/did-use-cases/

[6] Decentralized Identifiers (DIDs) v1.0 DID-Subject https://www.w3.org/TR/did-core/#did-subject

[7] Decentralized Identifiers (DIDs) v1.0 DID-Controller https://www.w3.org/TR/did-core/#did-controller

[8] Decentralized Identifiers (DIDs) v1.0 DID-Documents https://www.w3.org/TR/did-core/#dfn-did-documents

[9] Decentralized Identifiers (DIDs) v1.0 DID-CoreProperties https://www.w3.org/TR/did-core/#core-properties

[10] Decentralized Identifiers (DIDs) v1.0 JSON https://www.w3.org/TR/did-core/#json

[11] Decentralized Identifiers (DIDs) v1.0 JSON-LD https://www.w3.org/TR/did-core/#json-ld

[12] Decentralized Identifiers (DIDs) v1.0 Methods https://www.w3.org/TR/did-core/#methods

[13] Decentralized Identifiers (DIDs) v1.0 MethodsOperations https://www.w3.org/TR/did-core/#method-operations

[14]JSON-LD - JSON for Linking Data https://json-ld.org/

[15] W3c DID Specification Registry https://w3c.github.io/did-spec-registries/#did-methods

[16] The did:key Method v0.7 https://w3c-ccg.github.io/did-method-key/

[17] Peer DID Method Specification https://identity.foundation/peer-did-method-spec/

[18] Peer DID Method Specification Core https://identity.foundation/peer-did-method-spec/#core

[19] Peer DID Method Specification DIDDocument https://identity.foundation/peer-did-method-spec/#diddocs

[20] Peer DID Method Specification Backing Storage
https://identity.foundation/peer-did-method-spec/#backing-storage

[21] Peer DID Method Specification CRDT
https://identity.foundation/peer-did-method-spec/#crdts

[22] Peer DID Method Specification CRUD Operations
https://identity.foundation/peer-did-method-spec/#crud-operations

[23] DIDComm Messaging Specification
https://identity.foundation/didcomm-messaging/spec/

[24] Hyperledger Aries Features
https://github.com/hyperledger/aries-rfcs/tree/master/features

[25] Peer DID Method Specification Security Considerations
https://identity.foundation/peer-did-method-spec/#security-considerations

[26] What is a randomart ?
https://superuser.com/questions/22535/what-is-randomart-produced-by-ssh-keygen

[27] KERI Resources https://keri.one/keri-resources/

[28] “Key Event Receipt Infrastructure (KERI) Design” por Dr. Samuel M. Smith 2 de febrero 2021
https://github.com/SmithSamuelM/Papers/blob/master/whitepapers/KERI_WP_2.x.web.pdf

[29] KERI: A more Performant Ledger for Trusted Identities
https://medium.com/spherity/introducing-keri-8f50ed1d8ed7

[30] KERI: For every DID, a microledger
https://medium.com/decentralized-identity/keri-for-every-did-a-microledger-f9457fa80d2d

[31] Q & A about KERI https://identity.foundation/keri/docs/Q-and-A.html

[32] Capítulo 10 del “Self-Sovereign Identity MEAP” por el Dr. Samuel M. Smith
https://github.com/SmithSamuelM/Papers/blob/master/whitepapers/10-ssi-key-management.pdf

[33] Función de derivación de clave
https://es.wikipedia.org/wiki/Funci%C3%B3n_de_derivaci%C3%B3n_de_clave

[34] Token no fungible https://es.wikipedia.org/wiki/Token_no_fungible

[35] Rust Programming Language Website https://www.rust-lang.org/

[36] Rust Programming Language Guide https://doc.rust-lang.org/book/title-page.html

[37] DIDComm messaging specifications Rust
https://github.com/decentralized-identity/didcomm-rs

[38] Rust implementation of the KERI Core Library
https://github.com/decentralized-identity/keriox

[39] Javascript Object Signing and Encryption (JOSE)
https://jose.readthedocs.io/en/latest/

[40] La Union Europea nos prepara un “monedero digital”
https://www.xataka.com/aplicaciones/union-europea-nos-prepara-identidad-digital-dni-carnet-conducir-pagos-contrasenas-aplicacion

[41] ESSIF: The European self-sovereign identity framework
https://ssi-ambassador.medium.com/essif-the-european-self-sovereign-identity-framework-4572f6875e12