UNIVERSIDAD NACIONAL DE LA PLATA

Facultad de Informática

Crypto-Eventos

Hacia un modelo que garantice a perpetuidad la integridad de la información registrada con fines de auditoría, por parte de la electrónica de una red de datos

Tesis para obtener el grado de

Magister en Redes de Datos

Autor: Ing. Hugo Orlando Ortega
Director: Dra. Patricia Bazán
Co-Director: Mg. Nicolás del Río
Mayo 2020

…a mis queridos viejos, eternamente agradecido por el ejemplo y la familia que formaron

…a mi esposa Liliana y a mis hijos Máximo, Paula, Hugo y Victoria

Agradecimientos

Quiero agradecer al Mg. Nicolás del Río (Codirector de Tesis) y a la Dra. Patricia Bazán (Directora de Tesis) por su apoyo durante el proceso de desarrollo del presente trabajo.

A la Facultad de Ciencias Exactas y Tecnología de la Universidad Nacional de Tucumán, en especial al Laboratorio de Redes del Departamento Ciencias de la Computación, lugar donde paso muchas horas de mi vida en tareas académicas y de investigación.

Por último, agradecer, defender y valorar al sistema educativo de mi país, definido por ley como “un bien público, un derecho personal y social, del cual se hace cargo el estado”. En especial, en defensa de la universidad pública y gratuita, gracias a la cual pude educarme, como tantos miles de argentinos.

Resumen

La integridad de la información que contiene un registro de evento, originado en los componentes de una red, condiciona el éxito de una auditoría. En todos los casos, los datos aportados son claves en el momento de probar hechos delictivos.

El aumento del uso de las criptomonedas ha generado un gran interés en su tecnología subyacente, a saber, Blockchain. El componente central en una Blockchain es un libro mayor distribuido y compartido. Un libro mayor comprende una serie de bloques, que a su vez contienen una serie de transacciones. Una copia idéntica del libro mayor se almacena en todos los nodos en una red blockchain. Mantener la integridad y seguridad del libro mayor es uno de los aspectos cruciales de diseño de cualquier plataforma blockchain. Por lo tanto, generalmente hay mecanismos de validación incorporados que aprovechan la criptografía para garantizar la validez de los bloques entrantes antes de comprometerlos en el libro mayor.

El presente trabajo tiene como objetivo contribuir al mantenimiento de la integridad a lo largo del tiempo de la información contenida en los registros de eventos de los componentes de una red. Se propone una solución basada en Hyperledger Fabric, una plataforma de blockchain autorizada y de código abierto que está en pleno crecimiento y tiene un gran apoyo de la comunidad.

Palabras claves: Blockchain, Libro mayor, Registro de Eventos, Hyperledger Fabric

Abstract

The integrity of the information contained in an event log, originating from the components of a network, conditions the success of an audit. In all cases, the data provided are key at the time of proving criminal acts.

The increased use of Crypto-currency has generated a great deal of interest in its underlying technology, namely, Blockchain. The central component in a Blockchain is a distributed and shared ledger. A general ledger comprises a series of blocks, which in turn contain a series of transactions. An identical copy of the general ledger is stored on all nodes in a blockchain network. Maintaining the integrity and security of the general ledger is one of the crucial design aspects of any blockchain platform. Thus, there are typically built-in validation mechanisms leveraging cryptography to ensure the validity of incoming blocks before committing them into the ledger.

This paper aims to contribute to the maintenance of the integrity over time of the information contained in the event logs of the components of a network. A solution based on Hyperledger Fabric, an open source, licensed blockchain platform that is growing rapidly and has strong community support, is proposed.

Keywords: Blockchain, Ledger, Event Log, Hyperledger Fabric

Índice General

Índice de Figuras y Tablas

1. Capítulo 1: Introducción

Un gran porcentaje del hardware y el software que forman parte de un sistema de información puede registrar evidencias de todas y cada una de las transacciones o eventos que ocurrieron durante un tiempo dado. Esas evidencias son conocidas como logs, registros de eventos o registros de auditoría.

La importancia que tiene la información que proporciona un registro de eventos justifica el esfuerzo económico y tecnológico necesario para proteger la inmutabilidad de su contenido.

En informática, un log o registro, es la grabación en un archivo de texto o en una base de datos de un acontecimiento, acción o evento que afecta a un proceso en particular como, por ejemplo, una aplicación, una actividad de una red o un movimiento de un robot. Un conjunto de logs o historial de logs, constituyen una evidencia del comportamiento del sistema durante el período en que fueron registrados los mismos.

El almacenamiento de logs que fueron producidos tanto por el hardware de una red (switches, routers, firewall)[1], como por las aplicaciones de un sistema de información, permitirá prevenir, detectar y analizar delitos informáticos.

Se sabe que los logs o registros de auditoría pueden ser modificados o eliminados, ya sea por negligencia o imprudencia; en otros casos puede ser el resultado de acciones mal intencionadas, con el objetivo de cubrir un delito. Cualquiera fueran los motivos, el hecho de no poder saber: quién, cuándo, dónde y cómo se produjo un evento o transacción, convierte a todo sistema de informático en vulnerable y muy poco confiable.

El presente trabajo de tesis para obtener el grado de Magister en Redes de Datos pretende encontrar un mecanismo para asegurar la inmutabilidad de la información que contienen los registros de auditoría de un sistema digital. Una vez encontrado el mencionado mecanismo, cualquier sistema logrará ser auditado utilizando información confiable, que podrá ser usada como medio de prueba válida para el análisis forense. En este nuevo escenario, los sistemas de información digitales serán cada vez más confiables, robustos y menos vulnerables.

Elaborar una solución, en primer lugar, va a implicar estudiar cuáles son los avances obtenidos por la ciencia en aspectos que tienen que ver con la protección de la información a perpetuidad. En segundo lugar, será necesario comprender los beneficios y las metodologías de implementación de distintas herramientas. En tal sentido, es necesario:

- ✓ Entender cómo se gestiona un registro de auditoría y cuáles son las herramientas y protocolos que permiten centralizar el almacenamiento de los mismos, para luego procesarlos.
- ✓ Conocer las ventajas de poder distribuir el procesamiento y almacenamiento de los componentes de un sistema, esto es entender los sistemas distribuidos
- ✓ Comprender cómo favorecer a la confidencialidad e integridad de la información que viaja a través de las redes, y que luego es almacenada. En

[1] Equipos que interconectan y protegen las redes de voz, datos y video

ese sentido es clave el concepto de criptografía, como una ciencia que ayuda a garantizar tres aspectos claves de la información: autenticidad, integridad y confidencialidad.

La implementación de una solución que permita descentralizar el almacenamiento de los registros de auditoría, en un ambiente protegido, escalable y robusto, es un elemento clave en la solución que se busca.

Teniendo en cuenta las necesidades descriptas, una tecnología joven y disruptiva, como lo es blockchain, sin lugar a dudas cuenta en su diseño y operatividad con todas las respuestas que el presente trabajo precisa. El último desafío será elegir entre tantos desarrollos de cadenas de bloques, la blockchain que favorezca a una implementación baja en costo, alta en perfomance y lo suficientemente escalable.

1.1. Objetivo

El objetivo del presente trabajo de Tesis para obtener la Maestría en Redes de Datos es asegurar a perpetuidad la inmutabilidad del contenido de los registros de logs que fueron generados por los dispositivos electrónicos de interconexión de red (switches, routers, firewalls)[2] en una organización que tiene desplegada una infraestructura de networking[3].

Teniendo en cuenta la importancia del contenido de los registros de auditoría, se va a analizar, diseñar, proponer e implementar un sistema que almacene copias de los logs en distintos nodos de una red, utilizando un esquema de Infraestructura de Clave Pública (PKI) que autentique procesos, usuarios y proteja los datos.

Esta garantía de integridad a largo plazo puede implementarse con tecnologías de libro mayor distribuida (DLT), en particular aquella que confía en una cadena de bloques enlazada (blockchain). La puesta en funcionamiento de la solución se hará utilizando una plataforma abierta y en constante evolución: Hyperledger Fabric.

Se infiere que, utilizando una arquitectura distribuida y un sistema de almacenamiento en forma de cadena de bloques cifrada, se va a garantizar el mantenimiento intacto del contenido del log de auditoría, más allá de negligencias o alteraciones involuntarias que pudieran producirse durante los procesos de auditoría; fenómenos naturales o cualquier tipo de acción que pretenda esconder actividades delictivas.

[2] Dispositivos electrónicos con capacidad de configuración física y lógica que brindan conectividad y control de acceso a los usuarios de una organización.
[3] Infraestructura que permite a cualquier organización brindar a sus usuarios conectividad interna y externa, posibilitando el intercambio de datos, voz y video entre ellos y el mundo.

1.2. Motivación

Tucumán, junio de 2014. Ingenio azucarero. Sistema de control del rendimiento de la caña de azúcar implementado en una red LAN[4]. Se recibe caña de distintos productores y el laboratorio químico de la planta industrial calcula el rendimiento de la materia prima[5]. El código que identifica al productor y el rendimiento se almacenan en una base de datos a través de una transacción. Al momento de generar las liquidaciones de pago, un productor se queja, manifestando que el rendimiento de su caña es mucho menor al rendimiento que la misma tiene en otro ingenio. Se controla el valor de los tickets que posee el productor, con los valores actuales almacenados en el sistema. Se detectan inconsistencias. Alguien manipuló los registros almacenados en la base de datos. Se solicitan los backups de los registros de eventos de la base de datos. Por problemas en las cintas donde fueron grabados, no se puede recuperar los logs del mes completo antes requerido. Conclusión: se alteraron los valores de rendimiento de la caña de azúcar en la base de datos (favoreciendo a determinados productores, en detrimento de otros), los registros de auditoría no están disponibles; es imposible saber qué usuario, a qué hora y desde qué dispositivo se hicieron los cambios. Hay un delito, las huellas no están.

Lo descripto representa un caso real. En numerosas situaciones, cuando se pretende recuperar un backup realizado en cinta magnética, se producen errores de lectura que impiden contar con los datos. En ocasiones, políticas erróneas sobre la planificación del backup sobrescriben o "pisan" información que debería permanecer salvada. En otros escenarios se pueden presentar acciones negligentes que contribuyen a sobrescribir información (cinta, disco interno o externo, pen drive).

En caso de acciones delictivas, un atacante experimentado tiene dos objetivos: cometer el ilícito y no dejar rastros o "huellas" digitales. En ese sentido, va a utilizar todas las herramientas a su alcance para eliminar registros que pudieran incriminarlo.

Cualquiera fuera el sistema digitalizado, ¿qué es más importante: la transacción que cambia el valor del estado del activo[6], o la registración de quién, cómo, cuándo y dónde la hizo? La respuesta pareciera elemental; ambas son igual de importantes, y una de ellas no tiene valor, sin la existencia de la otra. Toda transacción es avalada por un registro que indica quién, cuándo, cómo y desde dónde la hizo. Todo registro de auditoría avala una transacción, y le otorga trazabilidad al estado de un activo.

El depósito de dinero en una cuenta bancaria, la reserva de un pasaje, el alta de una historia clínica, la modificación del dueño de un inmueble, ente otras., son todas transacciones que deben estar respaldadas por un registro de auditoría generado por el sistema correspondiente.

Es indudable que hay sobradas motivaciones operacionales, económicas, leyes civiles y penales, reglamentaciones gubernamentales, que justifican la necesidad de

[4] LAN: Local Area Network, red de área local.
[5] Representa la cantidad de caña de azúcar que es necesaria para producir un kilogramo de azúcar refinada para consumo humano.
[6] Activo, cualquier componente lógico o físico que tiene valor y debe ser auditado

disponer de recursos para preservar la prueba de una transacción. Esto implica disponer de una estrategia para colectar, salvar y mantener la información que contienen logs.

Existen distintas metodologías que permiten recoger, clasificar, analizar, integrar y almacenar en forma centralizada los archivos de logs generados por los componentes electrónicos de una red, como así también firewalls, servidores de correo electrónico, servidores de páginas web, etc. El hecho de contar con un repositorio centralizado tiene dos objetivos principales:

a) Disponer de información para detectar posibles ataques, vulnerabilidades, fallos de software, entre otros, que permitan ejecutar medidas para mitigar los mismos.
b) Contar con información precisa de cada uno de los eventos que se produjeron en los dispositivos y aplicaciones en la red, con el fin de detectar actividades irregulares o delictivas como, por ejemplo, detener servidores, cambiar configuraciones de switches y routers, robar y/o dañar información confidencial, cambiar privilegios de usuarios o producir información con fines delictuosos.

Para una organización, detectar y frenar ataques, como así también reparar vulnerabilidades es trascendente. No menos importante es conocer los eventos que sucedieron en la red y sus aplicaciones en un período de tiempo, como así también preservar las pruebas para descartar o confirmar acciones fraudulentas. Las transacciones delictivas debidamente probadas van a servir para aplicar a los delincuentes la ley 26.388 del Código Penal Argentino[7].

Es claro que el acceso no autorizado a las redes corporativas y ejecutar transacciones con privilegios de un usuario de la organización, representan hechos de profunda gravedad y constituyen un delito. Alterar o eliminar el contenido de un registro de auditoría, además de ser un delito en sí mismo, constituye el encubrimiento de un posible delito mayor, y las leyes penales en distintos países lo castigan cada vez con mayor severidad, en un mundo dónde el temor a los ciberdelitos está en franco aumento.

Por lo expuesto, no cabe dudas que el valor de la información que contiene un log, es fundamental como medio de prueba en un proceso de auditoría y si el resultado demostrara que hay delito, la misma podrá ser usada en el proceso penal correspondiente, de acuerdo a lo previsto en la ley antes mencionada.

Ahora bien, el esfuerzo y la motivación implican sólo almacenar los registros de auditoría y preservar su confidencialidad, o será necesario tener en cuentas otros aspectos. Para responder esta inquietud es preciso recurrir a un trabajo de Investigación de la Universidad de Princeton, en 2004 [1]. Allí se describen las características necesarias para que un log sea considerado un medio de prueba válido, las cuales son resumidas con las siguientes tres propiedades:

[7] Ley de Delitos Informáticos: http://servicios.infoleg.gob.ar/infolegInternet/anexos/140000-144999/141790/norma.htm

a) Resistente a la manipulación: debe garantizar que nadie más que el sistema generador del registro pueda crear entradas válidas, y que una vez que se hayan creado las entradas, no se puedan modificar.
b) Verificable: debe ser posible verificar que todas las entradas en el registro estén presentes y no han sido alteradas. Primero, cada entrada debe contener información suficiente para verificar su autenticidad cuando se considera por sí sola. En segundo lugar, las entradas individuales también deben vincularse de una manera que permita determinar si faltan entradas. El encadenamiento de hash, donde cada entrada contiene un resumen criptográfico de la entrada anterior, es una posible solución.
c) Accesibles con control de acceso: si bien los datos en un registro de auditoría pueden ser confidenciales, deben estar encriptados, se debe permitir el acceso de búsqueda legítimo a un subconjunto de todas las entradas del registro de auditoría

El planteo es claro, además de preservar los logs, es necesario asegurar que no fueron alterados y que responden a un ordenamiento en el tiempo.

Estamos en condiciones de asegurar que, si los registros se almacenan en un repositorio centralizado, el cual es configurado y mantenido por una sola entidad, existirá un alto riesgo de perder la inmutabilidad de los datos. En este escenario, existe una única superficie de ataque: la base de datos de registros. Hay un solo objetivo dónde atacar y todas las herramientas de los interesados van a intentar acceder y alterar el contenido.

Aun existiendo un sitio de contingencia en dónde se duplique la información, el escenario sigue siendo endeble. Es conocido que los sitios de contingencia son en el mejor de los casos una copia en línea del sitio original. Atacado el sitio original, se copia la manipulación del dato en el sitio de contingencia.

Lo descripto, nuevamente permite concluir que es necesario proponer una solución que se desprenda de la idea del sitio central, administrado por una única entidad.

En definitiva, no caben dudas que existen suficientes motivaciones para desarrollar un modelo de almacenamiento de logs que sean inmutables. Encontrar la solución, permitirá conocer toda la actividad sobre un sistema y entender el estado de un activo. Esto se contrapone a la experiencia por la que pasaron los investigadores del delito perpetrado en el ingenio azucarero, dónde ni siquiera pudieron conocer la cantidad de alteraciones que sufrió la base de datos.

La presente tesis ha sido dividida en dos secciones principales: Sección 1, Conceptos y Generalidades y Sección 2: Solución Propuesta. Los Anexos complementan los resultados y conclusiones obtenidas aportando información específica acerca de la temática resuelta. La descripción de cada una de las secciones y los anexos es la siguiente:

- ✓ Sección 1: – Conceptos y Generalidades. Está compuesta por los capítulos 2,3,4,5 y 6. En esta sección, en primer lugar, se desarrolla el estado del arte respecto al problema del mantenimiento de la integridad de la información en los sistemas de auditoría, luego se describe detalladamente acerca del *"activo"* que se desea resguardar: el registro de auditoría. Finalmente se describe el estado del arte de las

tecnologías y las herramientas básicas que serán utilizadas en la solución: sistemas distribuidos, criptografía y tecnologías de libro distribuido (DLT), en especial blockchain.

- ✓ Sección 2 – Solución Propuesta: Esta última sección está compuesta por los capítulos 7,8,9,10 y 11. El primero de estos capítulos contiene la propuesta formal de la solución, luego en el capítulo 8 se justifica la elección de Hyperledger Fabric como plataforma y se describe su arquitectura y funciones, siempre en el marco del problema a resolver. El capítulo 9 detalla los pasos necesarios para configurar e implementar la solución, y el capítulo 10 verifica que la red de cadenas de bloques y el *"contrato inteligente"* responden a los resultados esperados. Finalmente, en el capítulo 11 se exponen las conclusiones obtenidas y se proponen trabajos de investigación a futuro.
- ✓ Anexos: El anexo A describe las interfaces de consultas a la base de datos y a la blockchain (el *"contrato inteligente"*). El Anexo B contiene el código de las funciones "Invoke" y "Query", las cuales acceden a la blockchain para realizar las altas y las consultas respectivamente.

Sección 1: Conceptos y Generalidades

2. Capítulo 2: Estado del Arte

Si bien es cierto, 50 o 60 años atrás la mayoría de los sistemas de información no estaban digitalizados (al menos en Argentina), en la actualidad es difícil imaginar un sistema comercial, administrativo o industrial que no haya sido informatizado. El desarrollo de sistemas financieros, el comercio electrónico, la digitalización de entes gubernamentales, los sistemas de información en el campo de la salud, las soluciones automatizadas en la industria, entre otros, responden a un modelo digital, en dónde hay en juego miles de millones de dólares, e inclusive algo más importante, millones de vidas humanas.

Surge una pregunta: Las soluciones para el almacenamiento, la integridad y la disponibilidad de los registros de información ¿cuánto han evolucionado? y en caso que hayan evolucionado ¿cuán acorde es esa evolución en función de la importancia que tienen los mismos en los sistemas de información que los generan?

Siempre se supo que un registro de auditoría contiene información de gran valor, que existió y existe la necesidad de mantener inalterables el contenido de la misma asegurando su disponibilidad en las distintas situaciones en que sea requerida (auditorías, análisis forense, recuperación de transacciones, entre otras).

El objetivo primario fue encontrar procedimientos que conserven la integridad y la confidencialidad de la información. En la actualidad el desafío incluye la inmutabilidad (entiéndase como inmutable, la preservación en todo sentido, esto significa, evitar la modificación o eliminación de registros de auditoría sin dejar rastros)

Los desafíos descriptos han sido abordados por la comunidad científica a lo largo del tiempo. La problemática ha sido investigada y se desarrollaron distintas soluciones que fueron evolucionando junto al progreso de la tecnología del hardware y el software. Entre algunas de las propuestas, podemos mencionar:

- ✓ Mihir Bellare y Bennet Yee en 1997 desarrollaron el concepto "Integridad hacia adelante" usando esquemas MAC[8], número de secuencia y marcas de época para proteger el borrado de entradas en el historial [2]. Se propuso el uso de MAC para cada entrada de registro. El código MAC se genera con una clave que cambia cada cierto período de tiempo. Con esta técnica, si un atacante descubre la clave, sólo puede manipular los logs que se generaron con la última MAC (la clave encontrada sólo autenticó los logs a partir del último cambio de clave). Una muy interesante aproximación, pero aún se pueden borrar registros, como así también manipularlos, aunque sólo aquellos generados con la última clave, que es la clave que puede ser atacada.

[8] Message Authentication Code (MAC), valor que se calcula mediante una función hash y una clave secreta que conocen el emisor y el destinatario. Permite garantizar integridad y autenticidad del dato.

- ✓ Bruce Scheneier y John Kelsey entre 1998 y 1999 idearon un sistema que cifraba las entradas de log y un mecanismo para garantizar que un verificador V puede leer entradas de log individuales. [3]. Este método, se inicia con una clave compartida entre la máquina que genera los registros y la máquina que los va a almacenar. A partir de allí se cambia la clave de autenticación cada vez que se genera un registro. Cómo el mismo *paper*[9] lo indica, el sistema tiene una debilidad: un intruso puede tomar control de la máquina que almacena los registros y producir nuevas entradas corruptas (sin intentar eliminar o modificar un registro ya escrito), que son aceptadas como normales por el sistema.

- ✓ Brent R. Waters y otros entre 2003 y 2004 propusieron mejoras en el cifrado de las entradas del log. [1]. Desarrollaron un sistema robusto, en el que existe un agente, el cual a partir de una clave secreta que sólo él conoce, genera una clave pública que se la transfiere al servidor generador de logs, luego éste los encripta[10] con la clave pública. Esta última es usada por el agente, para generar la clave secreta, que se utilizara, para generar la próxima clave pública con la que el servidor va a encriptar el segundo registro, y así se repite el procedimiento. Se implementó este esquema de registros seguro en un sistema de consultas de datos MySQL, y el mismo introdujo una considerable sobrecarga.

- ✓ Wengsheng Xu y otros en 2005 construyeron un servicio web para ficheros de registros usando el esquema de Bruce Schneier y John Kelsey, junto a una Base de Computador Confiable para almacenar claves criptográficas. [4]. Tiene la flexibilidad de haber sido construido sobre una solución basada en aplicaciones web, los registros se pueden almacenar en una máquina no confiable, y para poder acceder a los mismos, se necesita recuperar las claves almacenadas en el computador confiable. El sistema tiene como objetivo limitar la capacidad de manipular los logs por parte de un atacante, y en caso de lograrlo, poder detectarla.

- ✓ Jason E. Holt desarrolla una solución llamada Logcrypt [5] basado en el trabajo desarrollado por [1]. El principal aporte es sustituir la MAC por firmas digitales. Este cambio desfavorece a la perfomance del sistema, pero permite corroborar identidades con claves públicas. Es una solución robusta, que fue aplicada para controlar la integridad de los archivos de logs de los servidores web. Si bien es una solución escalable, un atacante que toma control del sistema puede eliminar información.

- ✓ Vasilios Stathopoulos y otros entre 2006 y 2008 construyeron una solución para añadir seguridad a los logs generados en una red pública. Partieron de la solución propuesta por Bruce Schneier y John Kelsey a la que añadieron una autoridad confiable, responsable de almacenar las firmas de los registros y verificar la integridad de los mismos. Esto implicaba para los atacantes la necesidad de implementar un procedimiento adicional para manipular los

[9] Un paper representa un trabajo de investigación o comunicación científica publicado en alguna revista especializada.
[10] Acción de ocultar datos utilizando una clave y un algoritmo

registros de auditoría. [6]. Representa una solución muy confiable, pero de costosa implementación y compleja configuración.

- ✓ Giorgia Azzurra Marson y Bertram Poettering en 2013 usando MAC desarrollaron una cadena de hash sobre una función de un solo sentido, donde se construye el elemento n a partir del elemento i (n>i) sin tener que evaluar la función n-i veces. [7]. La solución propuesta se enfoca en la seguridad hacia adelante y en la posibilidad de validar las entradas en cualquier orden, pero con el menor costo computacional. Se basa en el desarrollo de un generador de claves secuenciales de rápida búsqueda. Si bien las claves pueden almacenarse en un servidor alternativo, el atacante se debe centrar en un único punto para poder manipular los registros: el servidor de claves.
- ✓ Una publicación de la IEEE en el año 2018 [8] desarrolla la idea de implementar el almacenamiento y análisis de log de auditoría sobre una estructura descentralizada y segura: blockchain. El artículo hace un análisis de la importancia que tienen los registros de auditoría. Basa su afirmación en el crecimiento de la necesidad de auditar transacciones, los cambios legales, las regulaciones, y la necesidad que tienen las corporaciones de tarjetas de crédito de cumplir con el nuevo estándar de alcance mundial PCI DSS (Payment Card Industry Data Security Standards). El artículo dice textualmente: *"Dependiendo del requisito de seguridad, la gestión de registros basada en una cadena de bloques podría ser una gran oportunidad. Soluciones parciales ya se encuentran disponibles en el mercado, como la protección de falsificaciones, pero las nuevas tecnologías pueden contener un potencial desconocido"*.

Lo descripto resalta los avances más relevantes en lo que respecta a la protección de logs de auditoría a lo largo del tiempo. Existen en desarrollo numerosos trabajos de investigación y nuevas propuestas para preservar los registros. Sin embargo, al momento de desarrollar la presente tesis, la vulnerabilidad de los sistemas de almacenamiento de logs sigue siendo reconocida en la comunidad digital y tratada entre otras organizaciones, por la prestigiosa comunidad OWASP[11].

2.1. Conclusiones del capítulo

A pesar de todos los esfuerzos realizados, los esquemas centralizados de almacenamiento de registros y claves siguen siendo vulnerables. Planteos realizados en el último artículo descripto (Fehér & Sándor, 2018), permite inferir que la comunidad ha empezado a considerar soluciones basadas en esquemas descentralizados (por ejemplo, blockchain).

A partir del contexto descripto y en función de la evolución de las soluciones propuestas, una tecnología disruptiva como los es blockchain, tiene todas las

[11] Open Web Application Security Project (OWASP) es una comunidad en línea que produce artículos, metodologías, documentación, herramientas y tecnologías disponibles libremente en el campo de la seguridad de las aplicaciones web.

posibilidades de brindar una respuesta eficiente al problema de mantenimiento de los registros de auditoría.

Es válido pensar en una migración desde la estrategia centralizada que emplea entidades intermedias (Figura 2.1.1, izquierda), hacia la implementación de ideas disruptivas, que permitan almacenar múltiples copias de los registros de auditoría, usando bloques encriptados, en un modelo descentralizado y tolerante a fallas (Figura 2.2.1, derecha). Esto es posible gracias a la extraordinaria evolución que han experimentado las redes de datos (velocidad, alcance geográfico, confiabilidad, escalabilidad, perfomance, tolerancia a fallas, etc.)

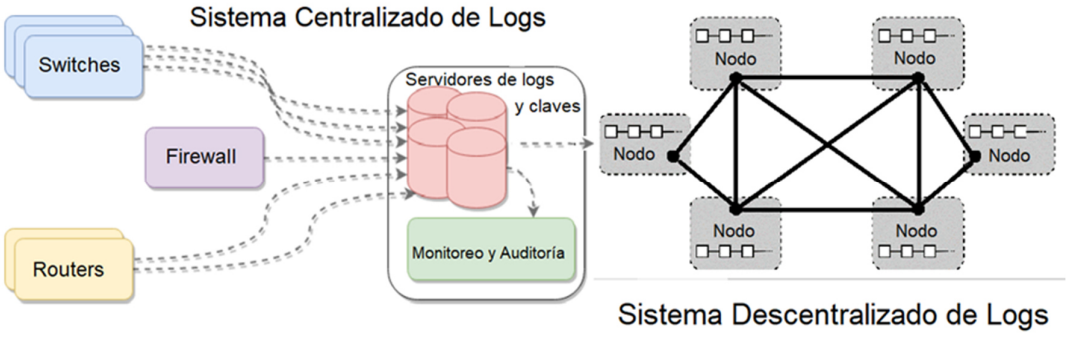

Figura 2.1.1: Sistema de almacenamiento de logs distribuido

Para avanzar y proponer una solución de almacenamiento de logs inmutable, que pueda ofrecer una alta perfomance, tanto para las altas, como así también para las consultas de registros, es apropiado abordar conceptos claves: cuales son los datos a salvar, qué herramientas existen y cuáles son los requisitos que debe cumplir el sistema. Es lo que se describe en la próxima sección.

3. Capítulo 3. Gestión de registros

Un registro es información que genera un ordenador, un dispositivo de red o una aplicación en respuesta a un estímulo. Esta información está contenida en una estructura de datos y puede ser enviada a una pantalla, a un archivo o a una base de datos en un servidor central.

Los estímulos pueden ser de variados orígenes, por ejemplo, una base de datos crea un log cada vez que se hace una "inserción" de registro en algunas de sus tablas, un AP (Access Point o Punto de Acceso) lo genera cuando algún equipo se conecta o desconecta de alguno de los puntos de acceso de la red, los servidores web generan un log cuando un cliente accede a alguna de las páginas web que alberga, un router lo hace cuando un usuario ejecuta cambios en la topología de la red, entre otros.

Podemos mencionar una gran cantidad de ejemplos como los anteriores y en cada uno, la información que contiene el log es diferente. Ésa información explica el motivo por el cual se ha generado el log.

Surge entonces una pregunta básica: ¿Qué significa gestionar un log?

Gestionar un log de auditoría implica llevar a cabo un conjunto de tareas que van a permitir salvar información relevante a cerca de los eventos que sucedieron en un sistema. Las funciones básicas de una gestión de logs son:

- ✓ Configurar el software y hardware para habilitar la registración de eventos.
- ✓ Determinar el tipo de evento a auditar.
- ✓ Seleccionar el método de colección de logs y el lugar de almacenamiento
- ✓ Elegir un formato y una sintaxis standard para facilitar el procesamiento de los logs
- ✓ Procesar y monitorear los registros
- ✓ Fijar políticas de backup definitivo (por ejemplo, cinta, disco externo, nube, DVD)
- ✓ Determinar procedimientos para el tratamiento de logs ante requerimientos legales o procedimientos de análisis forense.

Cualquiera fuera el escenario, en la figura 3.1 está descripto el ciclo de vida de un registro de auditoría. La primera acción necesaria es habilitar la registración de eventos en el hardware o software que se desea auditar (en muchos casos la auditoría está deshabilitada por cuestiones de perfomance). Luego se generan los registros que son enviados a un repositorio central, previa configuración del protocolo de recolección de logs (el standard es Syslog). Los registros son formateados de acuerdo a las necesidades del auditor, para luego ser monitoreado por aplicaciones disponibles en el mercado o desarrolladas en la misma organización. Las aplicaciones de monitoreo y análisis de logs pueden generar nuevos registros, por ello la retroalimentación existente entre el bloque de procesamiento y el de almacenamiento de los logs. En una última instancia, los registros pueden ser salvados definitivamente en cintas magnéticas, discos ópticos, sistemas de almacenamiento local o directamente en la nube.

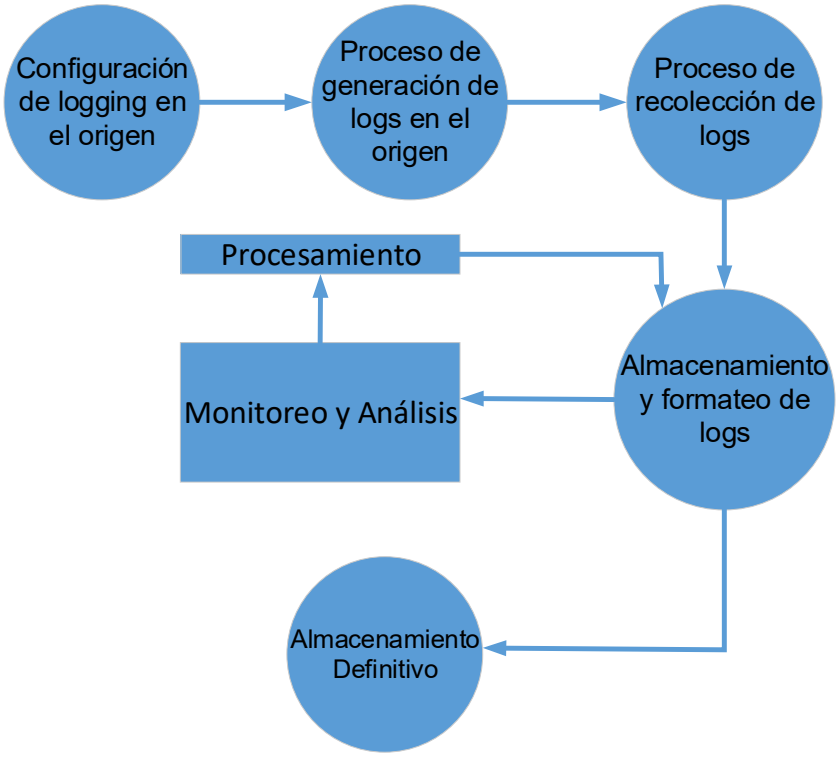

Figura 3.1: Ciclo de Vida de un Log

Ahora bien, ¿se van a generar cualquier tipo de logs, es necesario salvar todos los registros, la inmutabilidad del contenido es aplicable a toda la información? Estas preguntas se responden sólo si se implementa una apropiada gestión de logs. Para implementar una correcta gestión de registros, es necesario tener claro ciertos conceptos. Estos tienen que ver con comprender acerca de las distintas categorías, sintaxis, contenidos y originadores de registros, entre otras cosas. Es lo que se aborda en los próximos párrafos.

3.1. Categorías

Los logs que se originan tanto en dispositivos de hardware, como así también en distintas aplicaciones, sistemas operativos, bases de datos, detectores de intrusos, y demás dispositivos, se categorizan de acuerdo a los tipos de eventos que registran:

- ✓ *Debug:* son generados por las aplicaciones con la finalidad de ayudar a los desarrolladores a identificar problemas y depurar el código de dicha aplicación.
- ✓ *Informational:* están pensados para hacer saber a los usuarios y administradores que ha ocurrido algo sin gravedad. Por ejemplo, cuando debido a un evento programado, un servidor se reinicia, se generará el registro correspondiente. Esto en principio no tendría mayor importancia, a no ser que el reinicio se produjese en una situación en la que no debería, por ejemplo, cuando el servidor es reiniciado imprevistamente en un entorno de producción.
- ✓ *Notice:* generados por eventos que no son frecuentes, pero no son considerados errores, por lo que su importancia, al igual que el tipo anterior, dependerá del contexto.

- ✓ *Warning:* estos logs están pensados para situaciones donde falte algo que el sistema necesita, pero la ausencia de esto no tiene impacto sobre el correcto funcionamiento del mismo. También son usados para indicar que ocurrirá un error si no se realiza alguna acción. Por ejemplo, cuando es necesario actualizar el *firmware*[12] de un router, se va a generar un registro de *Warning*, más allá de que el dispositivo continuará funcionando correctamente
- ✓ *Error:* se usa para notificar los errores ocurridos en el sistema. Por ejemplo, un router generará un log de tipo error cuando no pueda iniciar alguna de sus interfaces.
- ✓ *Critical:* notifican algo que debe ser corregido, pero no es necesario que sea inmediatamente, por ejemplo, la pérdida de conexión con el ISP secundario mientras la conexión con el primario funciona correctamente.
- ✓ *Alert:* son usados para notificar algo que debe ser corregido inmediatamente, por ejemplo, la perdida de conectividad con un servidor. También están relacionadas con lo firewalls o IDS (Intrusion Detection System)[13], que generan un log de este tipo cuando detectan conexiones maliciosas.
- ✓ *Emergency:* notifican eventos por los que el sistema ha quedado inservible.

Hay tres acciones básicas que son recomendables se lleven a cabo antes de actuar en respuesta a un evento, ellas son: predecir, evaluar y detectar. La tabla 3.1.1 describe cuál o cuáles categorías de registro colaboran en mayor medida con cada una de las acciones.

Tabla 3.1.1: Acciones básicas a partir de las categorías de logs

Acción	Categoría de Log	Comentario
Predecir	Warning Error	Las advertencias y ciertos tipos de errores son pruebas que permiten predecir un acontecimiento
Evaluar	Informational Notice	Este tipo de logs deben ser evaluados por el auditor, si bien no representan un error, pueden indicar actividades que deberían suspenderse.
Detectar	Debug Critical Alert Emergency	Esta categoría de logs sirve para detectar situaciones anómalas, sobre las cuales es necesario actuar con urgencia.

Un enfoque distinto para clasificar los logs está directamente relacionado con la clase de evento que produce la generación de un registro nuevo. Si bien es cierto en la actualidad no hay oficialmente un estándar que defina tal clasificación, a partir del análisis de distintos trabajos de investigación se propone agrupar los mismos de acuerdo a la Figura 3.1.2:

[12] Software que tiene directa interacción con el hardware, siendo así el encargado de controlarlo para ejecutar correctamente las instrucciones externas

[13] IDS (Intrusion Detection System): es un programa de detección de accesos no autorizados a un computador o a una red.

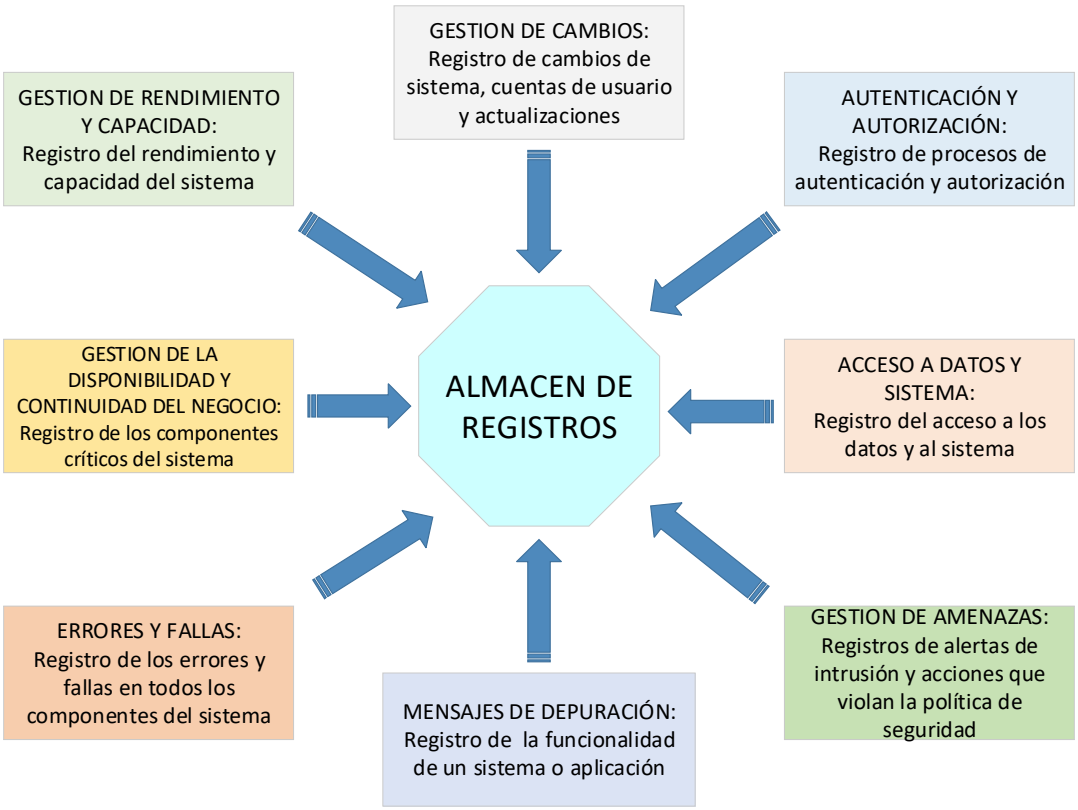

Figura 3.1.2: Eventos que generan registros de auditoría

3.2. Motivos para monitorear y analizar logs

Existen fundados motivos por los cuales es recomendable revisar los logs que genera todo sistema informático, entre los que incluyen el cumplimiento de leyes (por ejemplo, el Banco Central de la República Argentina establece normas en la operación de una Fintech[14]).

A partir del análisis del material bibliográfico, como así también la experiencia recogida en la administración de redes, se puede concluir que tres son los motivos principales por los cuales es imprescindible analizar los eventos que genera un sistema de información:

- ✓ Detectar Fallas y Solucionar Problemas: por ejemplo, ante la caída del servicio de resolución de nombres (DNS), un administrador puede revisar los logs generados por el sistema operativo del equipo y descubrir la causa de la baja del servicio, e iniciar acciones correctivas. La figura 3.2.1 muestra el contenido de un registro de auditoría informando un error en el servidor de nombres de dominio (DNS)

[14] Industria financiera que aplica nuevas tecnologías a actividades financieras y de inversión

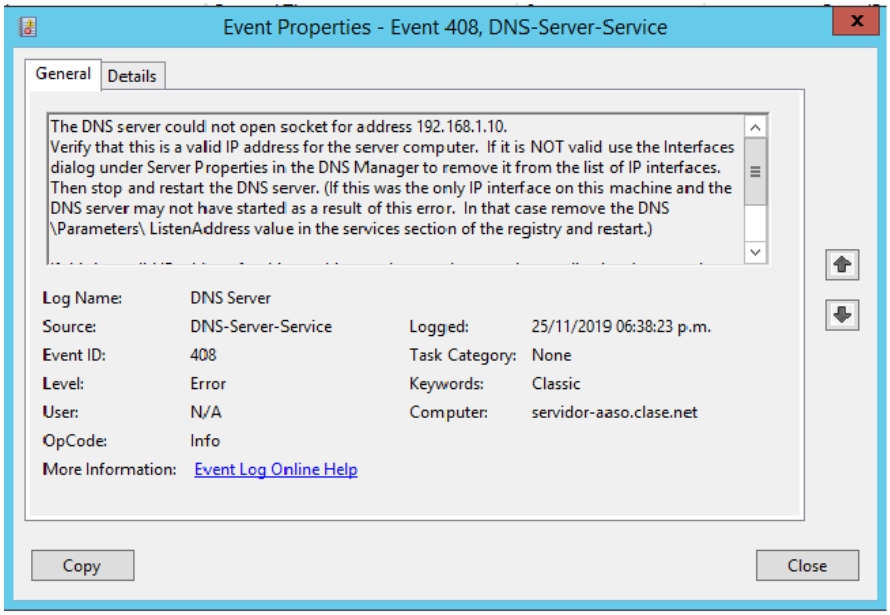

Figura 3.2.1: Log informando un error en el servicio DNS
Fuente: Servidor Windows del Laboratorio de Redes de la FACET – UNT

- ✓ Detección de Intrusos: Los dispositivos de seguridad de red como IDS o Firewalls generan logs que reflejan los eventos relacionados con la detección de intrusos. En el caso de la figura 3.2.2 se puede observar el registro de log de una herramienta llamada SNORT[15], la cual se configura para detectar patrones de ataques a la red, en este caso se puede ver la dirección ip y número de puerto de origen, el tipo de mensaje, la dirección ip y número de puerto destino, etc.

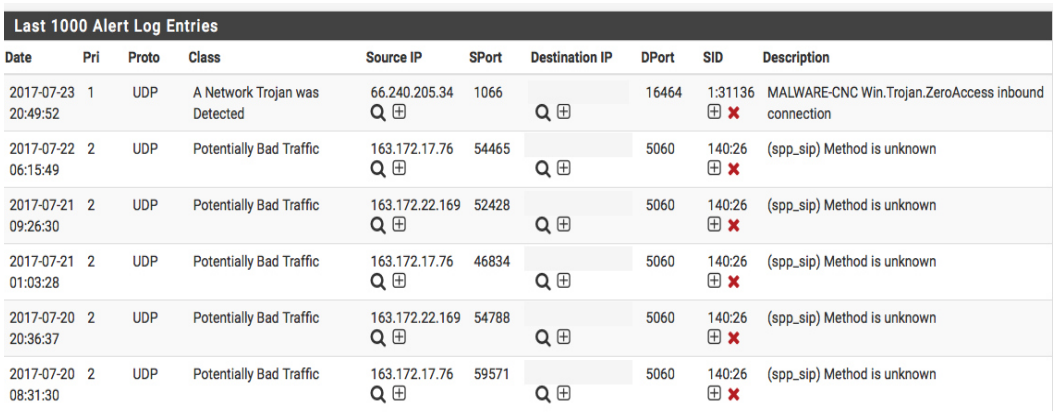

Figura 3.2.2: Archivo de logs generado por el detector de intrusos SNORT
Fuente: https://docs.netgate.com/pfsense/en/latest/ids-ips/snort-alerts.html

[15] Snort es un sistema de detección de intrusos en red, libre y gratuito. Ofrece la capacidad de almacenamiento de registros en archivos de texto y en bases de datos abiertas, como MySQL

- ✓ Análisis Forense: El análisis forense es el proceso consistente en construir una imagen de lo que ha ocurrido una vez que ha concluido un evento. Para construir un informe consistente, es crítica la credibilidad de la información usada. Los logs pueden ser una parte esencial en el análisis forense.

  Una vez guardados, los logs no son alterados durante el transcurso del uso normal del sistema, por lo que pueden constituir una fuente de información más fiable que otros datos del sistema que son más susceptibles a ser alterados o corrompidos.

  Además, normalmente los logs guardan una marca temporal que indica el momento en el que se generó. Gracias a esto, podemos crear una secuencia lógica de eventos ordenados en el tiempo.

  Por otra parte, los logs presentan una ventaja, y es que pueden ser enviados a un servidor central de logs (mediante un servicio de gestión de logs). Esto nos proporciona una fuente de evidencia externa, que se supone más fiable en el caso que se ponga en duda la integridad de la información de la fuente original (por ejemplo, si un intruso ha modificado o borrado los logs originales).

  En el caso de la Figura 3.2.3, el registro de auditoría está informando que el pasado 24 de noviembre a las 11:41 horas se intentó un inicio de sesión ilegal con privilegios de administrador sobre el módulo "ssh". En un ambiente real de auditoría, estos intentos fallidos son la antesala de una probable penetración al sistema, dado que está informando distintas "pruebas" para descubrir la clave del administrador.

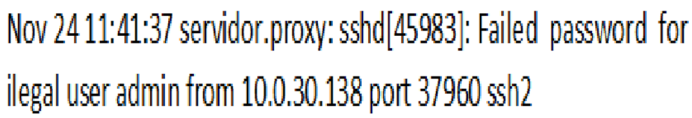

Figura 3.2.3: Intentos fallidos de inicio de sesión ssh
Fuente: Servidor Proxy del Laboratorio de Redes del DCC - FACET – UNT

3.3. Sintaxis de un log

Todos los logs, sea cual fuera su formato, tienen también una sintaxis. La sintaxis se refiere a como se estructura el contenido del registro. Cada log tiene una estructura propia, pero, basándonos en los formatos de logs más usados, podemos establecer un conjunto de campos que suelen aparecer en todos los registros:

- ✓ Fecha y hora (timestamp) en la que se generó el mensaje. Ayuda a ordenar los mensajes cronológicamente y analizar cómo se sucedieron los eventos.
- ✓ Equipo que generó el log. Esto es muy útil cuando tenemos varios equipos enviando logs a un servidor central, ya que permite identificar el origen cada log.
- ✓ Aplicación o componente que generó el log. Permite distinguir dentro de cada equipo, cual fue la entidad que generó el evento.

- ✓ Rigor (severity), prioridad, o nivel de importancia del log.
- ✓ Usuario: Si el log está relacionado con cualquier proceso iniciado por un usuario, indicará el nombre de usuario usado para el acceso.
- ✓ Contenido: Información que describe detalladamente al evento.

3.4. Contenido de un log

Es en este campo donde reside la verdadera información que se intenta transmitir con el log.

En algunos casos, como en el protocolo syslog, el contenido del mensaje es de estructura libre a diferencia del resto de campos. Esto dificulta el análisis automático ya que el contenido del log puede ser muy variado.

Con la idea de identificar el significado de los distintos tipos de contenido que forman parte de un registro de auditoría, se desarrolla una breve descripción de la información que proporcionan, y para ello utilizamos la clasificación expuesta en la Figura 3.1.2:

- ✓ Gestión de cambios: el campo describe cambios en el sistema, como la activación de un protocolo en un router; cambios de componentes, como el agregado de una placa de red a un servidor, actualizaciones, como la actualización de los patrones de ataque de un detector de intrusos, cambios en las cuentas, como la asignación a un usuario para formar parte de un nuevo grupo, y todo aquello que pueda ser objeto de un proceso de gestión de cambios.
- ✓ Gestión de autenticación y autorización: la información del campo contenido tiene que ver con el resultado de procesos de autenticación, como el inicio de una sesión con cuenta de usuario y contraseña; y autorización, por ejemplo, un intento de acceso correcto o fallido a un recurso del sistema o acceso al sistema privilegios especiales. Este tipo de log es generado por los subsistemas de seguridad de los sistemas operativos, routers, switches, firewall, entre otros. Identifican cada entidad que accede y/o utiliza privilegios.
- ✓ Gestión de Acceso a datos: informa acerca del acceso a componentes de los sistemas operativos, aplicaciones y datos, por ejemplo, informan detalladamente el tipo de acceso a un archivo o una tabla de una base de datos Indica si el acceso fue exitoso o fallido, como así también si fue para leer, modificar o borrar datos, por ejemplo, carpetas, registros o tablas. Este tipo de logs está estrechamente relacionado con el anterior, y busca mantener la privacidad de los usuarios informando quién y cómo accede a los datos en cada momento.
- ✓ Gestión de amenazas: informa detalladamente acerca de alertas de intrusos u otras actividades que violan las políticas de seguridad, como intento de acceso de clientes internos a sitios externos prohibidos, el intento de usuarios externos a servicios y aplicaciones instaladas en servidores en la red corporativa; como

así también la detección de patrones de ataques (denegación de servicio), identificando origen y consecuencias, entre otros. Estos logs son producidos por dispositivos de red con funcionalidades de seguridad (firewall, IDS, etc.).

- ✓ Gestión de rendimiento y capacidad: notifica sobre la gestión de rendimiento y capacidad del sistema, incluyendo memoria, procesamiento, espacio de almacenamiento, tráfico de red, etc., como por ejemplo los tiempos de acceso a disco, placa de red, cantidad de bytes enviados y recibidos por unidad de tiempo a través de la placa de red, porcentaje de tiempo de uso del procesador en un router, generación de alertas cuando el porcentaje de uso de disco ha superado un valor umbral, de la misma forma, alertas cuando el procesador o la memoria superan el valor máximo determinado para un aceptable tiempo de respuesta, etc.
- ✓ Gestión de continuidad y disponibilidad del negocio: informa sobre el resultado de operaciones relacionadas con las copias de seguridad, funcionamiento de cluster, sitios de contingencia y toda implementación que tenga que ver con las características de continuidad del negocio. El contenido de este tipo de logs informa acerca del estado de un cluster de servidores o firewalls, el estado de finalización de un backup en cinta, los eventos de sincronización de sitios de contingencia, el estado de las migraciones en caliente de máquinas virtuales, etc.
- ✓ Gestión de Errores y fallos: Registros de errores de sistemas que pueden o no demandar una acción por parte del administrador del equipo. El contenido de un registro de error informa por ejemplo la caída de un servicio como DNS, el error que genera el intento de conexión de una interface de un router con su vecino próximo, la caída de un enlace a internet, etc.
- ✓ Gestión de Depuración: Registros de mensajes de depuración generados por las aplicaciones. Generalmente el contenido de este tipo de logs es utilizado por los desarrolladores para encontrar el motivo de las fallas generadas por las aplicaciones que desarrollaron. En el momento de compilar un programa, se debe indicar al compilador que incluya información extra para la depuración del código. Con esa información extra en forma de logs, más la posibilidad que brinda el depurador de detener el programa en un punto de ruptura, los desarrolladores pueden corregir el código que genera el error.

3.5. Fuentes que generan logs

Todo dispositivo red, ordenador y/o aplicación que forma parte del sistema informático, tiene la capacidad de generar logs. Por otro lado, al existir una variedad tan importante de fuentes, se puede justificar por qué existen tantos formatos de logs distintos y la dificultad que esto genera a la hora de procesarlos.

Entre algunas de las múltiples fuentes que generan logs se encuentran:

- ✓ Sistemas Operativos (Linux, Unix, Windows, Mac OS y otros)

- ✓ Hardware de Servidores (HP, Lenovo, Dell, etc.)
- ✓ Swtiches y Gateways (HPE, Cisco, D-Link, Mikrotik, etc.)
- ✓ Routers (Cisco, Huawei, TPlink, etc.)
- ✓ Puntos de Acceso Inalámbrico (Cisco, Aruba, TPlink, Huawei, entre otros)
- ✓ Controladores de Puntos de Acceso
- ✓ Firewalls (Iptables, Ipcop, Untagle NG)
- ✓ Sistemas de Detección de Intrusos (Snort, Suricata, Bro, entre los más usados en la plataforma de código abierta)
- ✓ Bases de Datos (MySQL, SQLServer, DB2, etc.)
- ✓ Servidores HTTP (Apache, Internet Information Server, Nginx, Google GWS y otros)
- ✓ Servidores de Correo Electrónico (Sendmail, Postfix, Qmail, Zimbra, Exchange, entre otros)

3.6. Recolección y transmisión de logs

La transmisión y recolección de logs es una tarea sin complejidad. Un sistema de registración de logs puede configurarse sobre cualquier dispositivo de red o un computador, a través del cual puede generar un mensaje cada vez que se produzca un evento relacionado al sistema pre-configurado. Por otro lado, es necesario contar con un lugar donde enviar y recolectar los logs; puede ser desde una simple pantalla, un archivo común, archivos múltiples o hasta un servidor centralizado de recolección de logs.

Un servidor de logs es un computador, generalmente instalado con un sistema operativo Linux o Windows, donde se envían y recolectan todos los registros con el objetivo de centralizar la información. Utilizar un servidor de logs tiene bastantes ventajas, entre las que podemos destacar las siguientes:

- ✓ Permite almacenar logs provenientes de múltiples localizaciones en un lugar centralizado.
- ✓ Ofrece la posibilidad de almacenar una copia de seguridad de los registros que puede ser útil en caso de fallo del sistema que los generó.
- ✓ Facilita el análisis de toda la información que contienen todos los eventos de nuestra red.

La forma más común de enviar mensajes de log es usando el protocolo Syslog, el cual es un estándar de facto[16] para el intercambio de mensajes log. Se usa

[16] Un estándar *de facto* es aquel patrón o norma que se caracteriza por no haber sido consensuada ni legitimada por un organismo de estandarización. Se trata de una norma generalmente aceptada y ampliamente utilizada por un gran número de interesados

principalmente en plataformas Linux, pero también podemos encontrarlo en sistemas Windows y otras plataformas no basadas en Linux. Básicamente, lo que nos dice este protocolo es que hay un cliente que envía mensajes Syslog (con el formato especificado por el protocolo) a un servidor que está *"escuchando"* en un puerto, ya sea UDP (mayor performance) o TCP (entrega ordenada y confiable). La tarea principal del servidor es recibir los mensajes log en formato Syslog y almacenarlos en el disco local, donde podrán ser analizados. Syslog-ng es la versión actualizada de syslog, la cual ofrece nuevas prestaciones.

Syslog no es el único mecanismo para la transmisión y recolección de logs. Por ejemplo, Microsoft ha implementado su propio sistema de logging para Windows conocido como Windows Event Log. A pesar de esto, existen aplicaciones que se ejecutan sobre Windows, generando registros para Event Log, que luego convierten los mismos a un formato Syslog, para poder enviarse a un servidor Syslog, el cual es generalmente más usado para la gestión de los registros de auditoría.

Otro ejemplo de mecanismo para la transmisión de logs, es el protocolo SNMP[17]. Este es un protocolo para la gestión de dispositivos de red. El protocolo se basa en dos conceptos: push y pull. Un push o trap es una forma de mensaje de log que un dispositivo o equipo monitoreado emite cuando algo ha ocurrido. Este trap se envía a un equipo gestor o servidor de monitoreo que sería el análogo a un servidor de logs en Syslog. El concepto de pull, se refiere a la petición que puede realizar el equipo gestor o servidor de monitoreo a un dispositivo monitoreado, siempre consultando por el valor de variables predefinidas.

La figura 3.6.1 muestra el hardware y software de una red corporativa que contiene distintos servidores, sistemas de almacenamiento, bases de datos, aplicaciones, pc's clientes, switches, un firewall y un router conectado a internet. El hardware y software enumerado representan a los productores de registros de auditoría. En el presente ejemplo, todos han sido configurados para usar *"*Syslog*"* como protocolo de envío de mensajes de auditoría, junto con la dirección ip del cluster de servidores centrales de logs (192.168.1.250). En el gráfico, se puede observar hacia la izquierda, el cluster de servidores y su almacenamiento respectivo para capturar los registros provenientes de los originantes.

[17] SNMP (del inglés *Simple Network Management Protocol*) es un protocolo de la capa de aplicación que facilita el intercambio de información de administración entre dispositivos de red

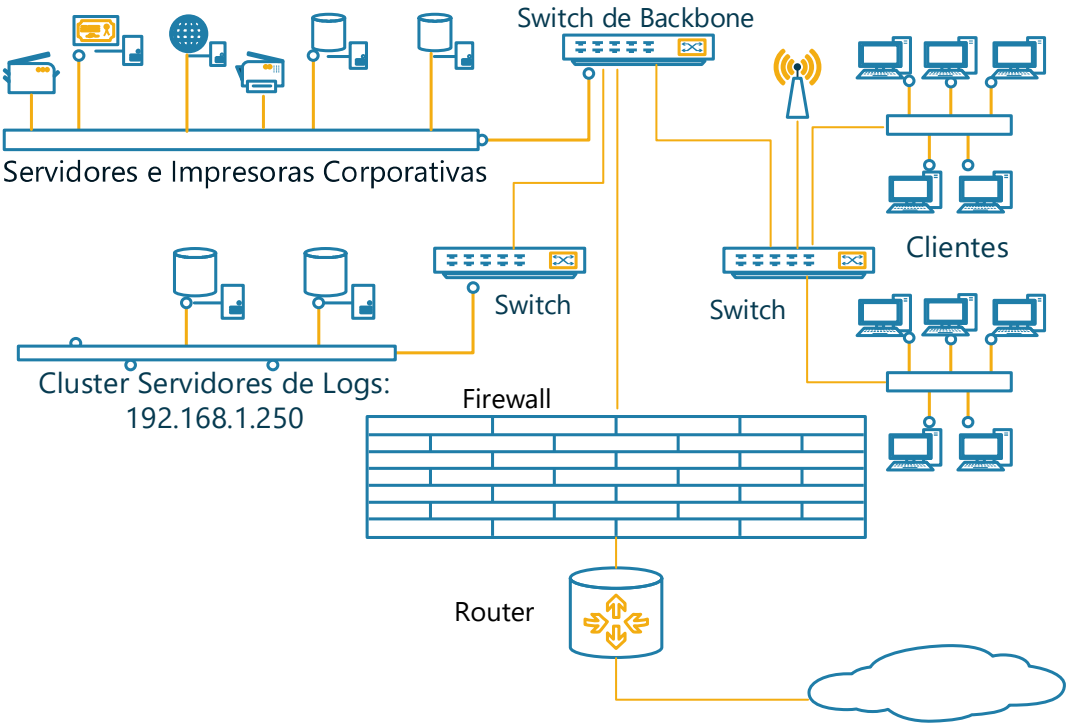

Figura 3.6.1: Red corporativa con un servidor de logs centralizado

3.7. Filtrado de logs

Luego de haber configurado nuestros dispositivos para generar y enviar mensajes log, el siguiente paso consiste en filtrar y normalizar estos mensajes.

Colectar y almacenar todos los tipos de logs que puede generar un dispositivo de red, un sistema operativo o una aplicación es una acción impracticable y sin sentido. Para ello existe el filtrado, el cual se encarga de incluir o excluir mensajes de log basándose en el contenido de estos mensajes. Algunas fuentes de logs tienen la capacidad de realizar este filtrado y en otros casos será necesario efectuarlo en el servidor colector de logs. No existen reglas o recomendaciones sobre qué logs filtrar y cuáles no, eso depende de las necesidades en cada momento. Por ejemplo, en un período de mantenimiento, mientras se está actualizando el firmware de los routers, switches y puntos de acceso de una red, sería recomendable filtrar los logs generados al reiniciar los equipos, ya que no nos aportan información.

La forma óptima de realizar el filtrado consiste en determinar lo que se quiere monitorear y luego filtrar en cada fuente los logs que no aportan información al respecto. De esta forma los registros que no se requieren, no se envían al servidor, por lo tanto, no consumen recursos de la red. En caso de que algunas de las fuentes no soporten filtrado de logs, sería recomendable realizarlo en el colector de logs más cercano a la fuente.

Normalización es el acto de tomar los mensajes de logs que por sus distintos tipos de fuentes poseen diferentes formatos y convertirlos a un formato común. Esto permite manipular y analizar los datos. Cabe destacar que la normalización es independiente de las fuentes y el protocolo usado para enviar los logs.

Un punto de partida es normalizar las distintas clases de prioridades y agruparlas utilizando una escala normalizada predefinida por el administrador, por ejemplo:

- ✓ Prioridad baja: Eventos meramente informacionales que no necesitan ninguna acción por parte del administrador.
- ✓ Prioridad media: Eventos que necesitan una acción por parte del administrador, pero no de forma inmediata. Por ejemplo, se bloqueó el tráfico proveniente de un dominio que ocasionalmente es consultado. Se debe averiguar las causas y tomar una decisión respecto al mismo
- ✓ Prioridad alta: Eventos que necesitan una acción inmediata. Por ejemplo, un servidor que se reinicia fuera del periodo de mantenimiento, un IPS alertando de un posible robo de información, un router informando de que se ha perdido la conexión con el IPS principal, etc.

La normalización se complementa con el concepto muy utilizado en el lenguaje informático: parser[18]. Consiste en analizar una entrada de texto a fin de determinar su estructura y convertir la entrada de texto en una estructura de datos que es apropiada para ser procesada. La figura 3.7.1 muestra un ejemplo de un log normalizado en formato JSON.[19]

```
{
    "Prioridad":"Media",
    "Timestamp":"2546981254",
    "Fuente":"server.web",
    "Aplicación":"adduser",
    "Id proceso":"323561",
    "Tipo":"new user",
    "Nombre de usuario":"usuario1",
    "uid":"124",
    "gid":"136",
    "Directorio home":"/home/usuario1",
    "Shell":"/bin/bash",
    "Mensaje original":"Nov 7 13:11:47
    server.web: adduser[323561]: new user:
    name=usuario1, uid=124, gid=136,
    home=/home/user1, Shell=/bin/bash"
}
```

Figura 3.7.1: Mensaje de Log formalizado en formato JSON

[18] Parser vocablo inglés que significa analizador sintáctico, acción de analizar sintácticamente. Se lo utiliza para indicar que se ha analizado la sintaxis de un programa o una estructura de datos.
[19] Es el acrónimo de JavaScript Object Notation, «notación de objeto de JavaScript», representa un formato de texto sencillo para el intercambio de datos.

3.8. Herramientas para la gestión de logs

Las herramientas para gestionar logs en una organización, se pueden obtener a través de 3 orígenes: desarrollar una solución personalizada, adquirir una solución lista para implementar, o contratar un servicio externo de gestión de logs [9].

En el caso de desarrollar una solución a medida, la gran ventaja consiste en que se puede dar respuestas específicas a la medida de los requerimientos de la organización, y se cuenta con libertad para seleccionar las herramientas. La desventaja radica en la necesidad de asignar recursos y tiempo al proyecto, además de no contar con soporte externo.

Si la idea es adquirir una solución ya desarrollada, la gran ventaja es poder contar con la aplicación lista para ser usada, con soporte y actualizaciones aseguradas. La desventaja, tiene que ver con la necesidad de parametrizar la solución y en algunos casos adaptar la empresa a los requisitos de la solución adquirida.

Por último, si se elige la opción de contratar un servicio externo, la gran ventaja es que no se requiere destinar recursos humanos, ni tiempo en la operación y el mantenimiento de la solución. La desventaja radica en la pérdida del control de los datos.

A continuación, la tabla 3.8.1 representa un resumen de las distintas herramientas y aplicaciones disponibles en la comunidad:

Tabla 3.8.1: Tabla de herramientas de gestión de logs

Herramienta	Función	Tipo Licencia	Descripción
grep	Análisis básico de logs	Código abierto	Permite la búsqueda de patrones en archivos de logs planos de texto. http://www.gnu.org/software/grep/grep.html
Microsoft log parser	Análisis básico de eventos windows	Sin costo	Permite filtrar la información contenida en fuentes de logs en ambientes Windows. https://www.microsoft.com/en-us/download/details.aspx?id=24659
splunk	Recolección y Análisis de Logs	Versión sin costo y versión full paga	Permite la centralización, almacenamiento y análisis de logs. http://www.splunk.com
OSSIM	SIEM	Código Abierto y versión full paga	Permite descubrimiento de activos, evaluación de vulnerabilidades, detección de amenazas, monitoreo de comportamiento e inteligencia de seguridad. https://www.alienvault.com/products/ossim

Tabla 3.8.1: Tabla de herramientas de gestión de logs (continuación)

Snare	Centralización de Eventos Windows	Sin costo	https://www.snaresolutions.com/solutions/log-monitoring-and-management/
syslog syslog-ng	Centralización de logs	Código Abierto	Permite recolectar los logs de múltiples fuentes y centralizarlas en un servidor.
rsyslog	Centralización de logs	Código Abierto	Permite recolectar los logs de múltiples fuentes y centralizarlas en un servidor.
awk	Extracción y Análisis Simple de Logs	Código Abierto	Permite extraer y procesar información de archivos de logs planos de texto. http://savannah.gnu.org/projects/gawk/
sed	Búsqueda, Extracción y Sustitución Simple de Logs	Código Abierto	Permite extraer y procesar información de archivos de logs planos de texto. http://www.gnu.org/software/sed/
NetIQ Sentinel	Recolección y Análisis de Logs	Sin costo y versión full paga	Permite, además de la gestión de logs, detección de anomalías y manejo de identidades. https://www.microfocus.com/en-us/products/netiq-sentinel/overview
IBM QRdar	Recolección y Análisis de Logs	Paga	Permite la centralización, almacenamiento y análisis de logs. https://www.ibm.com/us-en/marketplace/ibm-qradar-siem?mhsrc=ibmsearch_a&mhq=qradar
Loggly	Proveedor de servicios de gestión de logs en la nube	Paga	Permite la gestión de logs en la nube. https://loggly.com
Graylog	Centralización y análisis de logs	Código Abierto	Construido sobre código abierto, graylog recopila, mejora, almacena y analiza los datos de registro https://www.graylog.org/products/open-source

Tabla 3.8.1: Tabla de herramientas de gestión de logs (continuación)

Security Event Manager	Centralización y análisis de logs	Paga	Recolecta normaliza registros Detecta y responde en forma automática amenazas. Muy buen nivel de reportes https://www.solarwinds.com/security-event-manager?CMP=BIZ-RVW-CMPRTCH-10BestLogMngmtTools
Logstach	Centralización de logs	Código Abierto	Logstash es un pipeline de procesamiento de datos de open source del lado del servidor que ingesta datos de una multitud de fuentes simultáneamente, los transforma y luego los almacena. https://www.elastic.co/es/products/logstash
XpoLog	Colección, análisis y monitoreo	Paga	https://www.xplg.com/
Amazon CloudWatch Logs	Recolección y Monitoreo de registros	Paga	Centraliza los registros de todos los sistemas, aplicaciones y servicios de AWS en un único servicio de gran escalabilidad. Se puede verlos, buscar patrones, filtrarlos o archivarlos de forma segura para analizarlos en el futuro. https://docs.aws.amazon.com/es_es/AmazonCloudWatch/latest/logs/WhatIsCloudWatchLogs.html
Stackdriver Logging (Google)	Gestión y análisis de logs en tiempo real	Paga	Permite almacenar, buscar, analizar, supervisar y utilizar alertas sobre eventos y datos de registros desde Google Cloud Platform y Amazon Web Services (AWS) https://cloud.google.com/logging/?hl=es

3.9. Conclusiones del capítulo

Comprender lo que representa un registro de eventos, entender cómo se gestionan y conocer las distintas funcionalidades que ofrecen las herramientas de centralización y gestión de registros de auditoría, representan pasos indispensables para avanzar en una primera propuesta de solución al problema planteado.

Podemos concluir que todos los sistemas de gestión de logs necesitan de una u otra forma centralizar su almacenamiento para poder implementar un monitoreo efectivo de toda la red. Esta centralidad implica una gran debilidad. Más allá de las consideraciones de seguridad física e inclusive criptográfica, un atacante, con solo acceder al repositorio central, puede provocar daños irreversibles.

Es necesario pensar en algún esquema distribuido, que no sólo permita el almacenamiento replicado en distintas entidades (cuánto más entidades, mejor), sino que también plantee algún esquema de criptografía encadenada. Un escenario con estas características demandará un esfuerzo extraordinario para penetrar, agregar, modificar y/o eliminar registros de auditoria a lo largo del tiempo.

4. Capítulo 4. Conceptos de Sistemas Distribuidos

Los sistemas computacionales experimentan una extraordinaria evolución. Desde aquel no tan lejano 1945, cuando comenzó la era moderna de las computadoras, hasta mediados de la década del 80 en el siglo pasado, los computadores ocupaban grandes espacios físicos y su precio era inalcanzable para el grueso de la comunidad. Como resultado, las organizaciones contaban con no más de 1 o 2 computadoras que trabajaban en forma aislada.

Sin embargo, hacia la segunda mitad de la década de 80, hubo dos avances claves en el desarrollo del hardware, que produjo grandes cambios.

El primero de estos avances fue el desarrollo de poderosos microprocesadores, que evolucionaron de los 16 a los 32 y posteriormente 64 bit. De esta forma una computadora que cabía en un escritorio y costaba alrededor de 1.000 dólares pasó a tener el poder de un supercomputador que costaba 500.000 dólares. Tan impactante es el avance que, "si la industria del automóvil hubiera mejorado en valores similares a la industria informática, en la actualidad un Rolls Royce costaría 1 dólar y tendría un rendimiento de 1000 millones de kilómetros con casi 4 litros de combustible". [10]

El segundo gran avance fue la invención de las redes de área local (LAN, Local Area Network), de computadoras de alta velocidad. Estas redes, permiten la interconexión de cientos de computadoras ubicadas en un predio de 2 km., o en un edificio, y de esta forma se puede transferir pequeños volúmenes de datos a 10 Mbps, 100 Mbps, 1 Gbps o 10 Gbps[20]. Por otro lado, las redes de área amplia (WAN, Wide Area Network), que permite interconectar las redes LAN a lo largo del mundo, uniendo miles de millones de computadoras en una gran interred, logran vincular sitios remotos a velocidades simétricas de hasta 10 Gbps conectados con enlaces de fibra óptica.

El resultado de estos avances hace factible, y relativamente fácil poner a trabajar sistemas de cómputo compuestos por grandes cantidades de ordenadores interconectados mediante redes LAN y WAN. Estos sistemas se les conoce como redes de computadoras o sistemas distribuidos, al contrario de los sistemas centralizados que por lo general constan de una sola computadora, sus periféricos, y quizás algunas terminales remotas.

4.1. Definición de Sistema Distribuido

En la literatura especializada, existen distintas definiciones para los sistemas distribuidos, ninguna de ellas es completa, como así tampoco guarda concordancia con las demás. A los fines del desarrollo del presente trabajo, es suficiente la siguiente definición extraída del libro de Andrew S. Tanenbaum [10]: *"Un sistema distribuido es una colección de computadoras independientes que dan al usuario la impresión de constituir un único sistema coherente"*.

[20] 10 Mbps, velocidad de transferencia de 10 millones de bit por seg., 100 Mbps son 100 millones de bits por seg., 1 Gbps, son mil millones de bit por seg., 10 Gbps, son diez mil millones de bit por seg.

Un ejemplo categórico de un sistema distribuido es la implementación de Internet en el mundo. Una revolución que va más allá de la tecnología y que cambió para siempre la vida del hombre. Internet representa un gran sistema, constituido por miles de millones de ordenadores que ofrecen entre otros servicios, cientos de millones de páginas web a usuarios que se conectan desde distintos dispositivos sin la necesidad de tener que conocer absolutamente nada acerca del servicio que usa. Simplemente, en el caso de las páginas web, un usuario cualquiera, con un ordenador o dispositivo con conectividad (pc, notebook, tableta, teléfono móvil, etc.), en un lugar cualquiera del plantea, con sólo contar con acceso a la red, una simple aplicación llamada navegador, y una dirección muy amigable de recordar y fácil de escribir (por ejemplo, www.cronista.com), tiene la posibilidad de acceder al contenido completo, en este caso, de una plataforma con un servicio de noticias. La figura 4.1.1 representa la realidad omnipresente de Internet en cada rincón del planeta.

Figura 4.1.1: Internet, un verdadero sistema distribuido

La definición desarrollada en el párrafo anterior nos permite vislumbrar cuales son las características que definen a un sistema distribuido, por el cual un usuario interactúa con el mismo a través de una simple interface, la cual esconde la enorme complejidad desarrollada para brindar el servicio que requiere el interesado.

Para permitir que las computadoras y redes heterogéneas puedan brindar servicios y comunicarse entre ellas, mientras se ofrece a la vista del usuario un sistema único, los sistemas distribuidos comparten en forma transversal y lógica una capa de software intermedia entre dos capas bien definidas:

- ✓ Capa de alto nivel formada por usuarios, como aplicaciones
- ✓ Capa de bajo nivel constituida por sistemas operativos y recursos básicos de comunicación.

Esta capa de software intermedia es un sistema distribuido y se le conoce como middleware, y se lo puede observar en la figura 4.1.2.

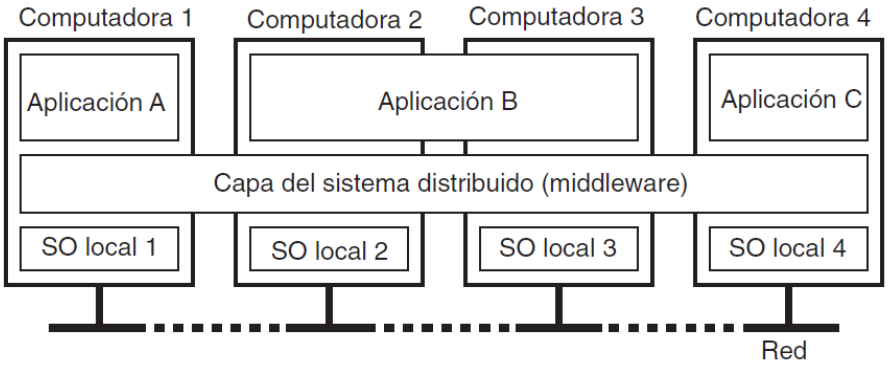

Figura 4.1.2: Midelware. Interfaz única en la heterogeneidad del hardware\software.
Fuente: [10]

4.2. Objetivos de un sistema distribuido

Para comprender en profundidad la gran revolución que se generó a partir del desarrollo de soluciones utilizando una arquitectura de sistemas distribuidos, es necesario conocer claramente cuáles son los objetivos que pretende alcanzar este nuevo tipo de diseño. Los objetivos que persigue todo sistema distribuido son:

- ✓ Disponibilidad de Recursos: el acceso de los usuarios y las aplicaciones a recursos remoto debe ser simple. Los recursos pueden ser lógicos (páginas web, datos, procedimientos, aplicaciones) o físicos (discos, impresoras, cintas, etc.).
- ✓ Transparencia: Es la capacidad de ocultar al usuario o a la aplicación que hace uso del sistema distribuido, la complejidad de la distribución de los procesos y los recursos a lo largo de varios sistemas. Este es un objetivo difícil de alcanzar, y distintos autores, entre los que se encuentra [10], clasifican los tipos de transparencia en:
  - ✓ Acceso: oculta diferencias en la representación de los datos (por utilizar diferentes SO, arquitecturas) y mecanismos de invocación. Ejemplo, un usuario accede a un recurso local de la misma forma que accede a un recurso remoto.
  - ✓ Ubicación: oculta la ubicación real de los objetos. Ejemplo: un usuario desea acceder al sitio de Debian, accede a una aplicación conocida como navegador y escribe: https://debian.org. No tiene que conocer ningún nombre de servidor, ni ubicación de la página en el sistema de archivos.
  - ✓ Concurrencia: ejecución simultánea de múltiples procesos que comparten recursos sin interferir entre sí (compartir un archivo en un file server). Es importante dejar el recurso en un estado consistente. Ejemplo, se puede compartir un archivo en Google Drive, y trabajar simultáneamente sobre él.
  - ✓ Replicación: múltiples instancias de recursos aumentan la confiabilidad y performance, ocultando a usuarios finales la existencia de las réplicas. Ejemplo, los 13 (trece) servidores raíz, que se utilizan en un proceso de

resolución de nombres (Root Hint), se encuentran replicados varias veces a lo largo del mundo.

- ✓ Fallas: ocultar los fallos de software o hardware, de forma que los usuarios del servicio distribuido no perciban la degradación. Ejemplo, cuando se pierde un vínculo entre dos routers, el protocolo de ruteo de la interred aplica mecanismos para encontrar caminos alternativos a la entrega de los paquetes de red. Luego reconfigura todos los routers.
- ✓ Performance: permite al sistema reconfigurarse dependiendo de la variación de la carga. Ejemplo, los protocolos de ruteo dinámicos e inteligentes (BGP), pueden recalcular el uso de rutas, en base al tráfico existente en un momento dado.
- ✓ Migración / Reubicación: permite la movilidad de recursos sin afectar la operación del servicio. Tiene que ver con la posibilidad de implementar distintos tipos de clúster[21] de servidores, de forma tal que puedan cambiar o reubicar algún miembro del clúster, sin que afecte la operación del sistema en producción.

- ✓ Sistemas Abiertos: Los sistemas deben poder interactuar con otros sistemas, sin importar el ambiente subyacente: deben respetar interfaces bien definidas, soportar la portabilidad de aplicaciones, ser extensibles de forma simple.
- ✓ Escalabilidad: Un sistema es escalable si se mantiene el nivel de confiabilidad y la performance cuando hay un incremento significativo en la cantidad de recursos y usuarios. Para cumplir con el objetivo de lograr un sistema escalable, hay algunas técnicas que pueden ayudar:
  - ✓ Ocultar los tiempos de latencia en las comunicaciones, por ejemplo, un ordenador inicia un requerimiento de un recurso remoto y mientras se demora la respuesta, inicia otro proceso, hasta recibir la misma (esto no funciona en aplicaciones interactivas).
  - ✓ Distribuir, consiste en tomar un objeto, dividirlo y distribuirlo a lo largo de la red. El ejemplo que mejor cabe es el sistema de nombres de dominio (DNS, Distributed Name System), el cual tiene la capacidad de resolver todos los nombres de los dominios del mundo, utilizando una base de datos que se encuentra distribuida en cientos de miles de servidores. El gráfico 4.2.1 representa una porción del espacio de nombres distribuido en forma de árbol invertido.
  - ✓ Replicar, consiste en duplicar o triplicar recursos con el objetivo de aumentar la disponibilidad y permitir el balanceo de carga, mejorando la performance.

[21] Sistema que se basa en la unión de varios servidores, los cuales trabajan como si fuera un solo servidor.

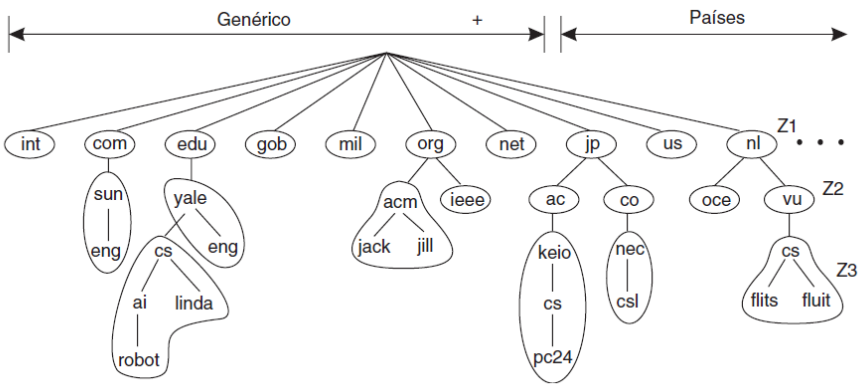

Figura 4.2.1: Distribución de una base de datos para resolver nombres de dominio
Fuente: [10]

- ✓ Heterogeneidad: este objetivo está referido a los sistemas operativos de los dispositivos y ordenadores, redes LAN y WAN, hardware, lenguajes de programación, aplicaciones, etc. En este punto cobra importancia el desarrollo del Middleware, que representa a la capa de software que provee abstracción de la programación, enmascarando la heterogeneidad de las redes, hardware, los sistemas operativos y los lenguajes subyacentes. La mayoría del middleware utiliza como protocolo de transporte y ruteo la suite TCP/IP. Algunos ejemplos:
  - ✓ JRMI (Java Remote Method Invocation): es un mecanismo ofrecido por Java para invocar un método de manera remota.
  - ✓ RPC (Remote Procedure Call): es un programa que utiliza una computadora para ejecutar código en otra máquina remota sin tener que preocuparse por las comunicaciones entre ambas
  - ✓ SQL (Structured Query Languaje): es un lenguaje de dominio específico utilizado en programación, diseñado para administrar, y recuperar información de sistemas de gestión de bases de datos relacionales.
  - ✓ CORBA (Common Object Request Broker Architecture): es un estándar definido por Object Management Group (OMG) que permite que diversos componentes de software escritos en múltiples lenguajes de programación y que corren en diferentes computadoras, puedan trabajar juntos
  - ✓ REST (Rrepresentational State Transfer): es un estilo de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web
  - ✓ SOAP (Simple Object Access Protocol): es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML

- ✓ Seguridad: Muchos de los recursos mantenidos por los sistemas distribuidos tienen un gran valor para sus usuarios. La seguridad, cuando se refiere a los datos digitales, tiene tres componentes:
  - ✓ Confidencialidad: protección de acceso a usuarios no autorizados
  - ✓ Integridad: protección de alteración o corrupción
  - ✓ Disponibilidad: protección de interferencias que no permitan acceder al servicio (DoS)[22]
- ✓ Concurrencia: Generalmente los procesos que manejan un recurso compartido deberían atender un cliente por vez, pero esta forma de trabajar limita el rendimiento y por tanto no es real, dado que múltiples requerimientos se atienden concurrentemente. Por lo tanto, se debe asegurar que las operaciones concurrentes sobre objetos puedan sincronizarse de forma tal de lograr consistencia en los datos.

4.3. Tipos de Sistema Distribuidos

Se puede implementar sistemas distribuidos para resolver distintos tipos de problemas de la vida real, es posible que se necesite distribuir carga en sistemas computacionales de alto rendimiento, u organizar procesos y datos en forma distribuida con el objeto de mejorar tiempos de respuesta en el proceso de la información, como así también, en la actualidad, se deben definir la condiciones para la operación de sistemas masivos, entre los que podemos citar Internet de la Cosas (IoT)[23]. Los sistemas distribuidos se pueden clasificar en:

- ✓ Sistemas Distribuidos Computacional: Varios sistemas distribuidos se configuran para obtener potencial de computación de alta performance, como puede observarse en la figura 4.3.1:
  - ✓ Clusters: Grupo de sistemas de similar dimensión conectados a través de LAN.
  - ✓ GRID (computación en malla): grupo de nodos heterogéneos, generalmente dispersos a nivel geográfico y de organizaciones que se conectan a través de la red WAN (Red de Area Extendida)

[22] Denial of Service: es un ataque a un sistema de computadoras o red que causa que un servicio o recurso sea inaccesible a los usuarios legítimos

[23] Internet de las cosas es una red de objetos físicos (autos, casas, máquinas, electrodomésticos, etc.), los cuales utilizan sensores y APIs, para conectarse e intercambiar datos por internet.

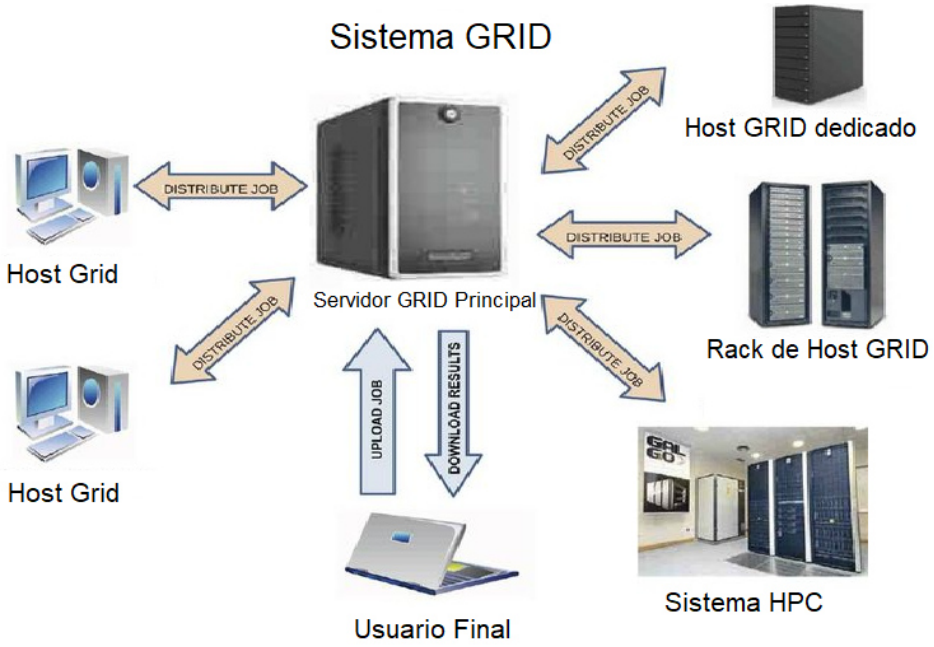

Figura 4.3.1: Arquitectura de Servidores GRID

- ✓ Sistemas Distribuidos de Información: sistemas basados en transacciones: el acceso a un recurso (objeto o base de datos) se realiza por medio de operaciones que se ejecutan atómicamente de forma que se ejecuten todas exitosamente o ante una la falla de una de ellas, el proceso dejará los recursos tal cual estaban antes de ser procesados. En la actualidad el mundo depende de los sistemas distribuidos de información, por ejemplo, pensar en un sistema de ventas de pasajes y hoteles como Despegar, o en un sistema de ventas como Mercado Libre, hasta un sistema de gestión de seguridad social como lo es el sistema de ANSES.

- ✓ Sistemas Distribuidos Masivos: un nuevo tipo de sistemas distribuidos basado en nodos pequeños y/o móviles, que generalmente recogen información para otros sistemas. Se puede pensar en sistemas de monitoreo de variables ambientales en distintos puntos de una provincia, en un sistema de logística en general, o simplemente operar a distancia un aire acondicionado o lavarropas de un hogar. Estos sistemas tienen características específicas:

    - ✓ Frecuentes cambios de contexto
    - ✓ Fácil configuración, puesto que tienen una llegada masiva, a usuarios que generalmente no son expertos
    - ✓ Los nodos de estos sistemas se encuentran en las extremidades de la red, pueden estar en constante movimiento, y en general están censando información del medio.

4.4. Arquitectura de Sistemas Distribuidos. Las redes P2P

La arquitectura de un sistema distribuido representa el principio básico que se sigue para organizar la interacción entre los componentes de software que lo conforman. Existen diferentes aproximaciones o metodologías de diseños, entre las que incluyen el modo de capas, la orientación a objetos, la orientación a eventos, y la orientación a espacios de información.

Un estilo altamente centralizado, es la arquitectura cliente-servidor, en dónde existen dispositivos con el rol de clientes y otros con el rol de servidor. El cliente envía una petición a un servidor, y luego que el mismo la procesa, le devuelve el resultado al solicitante. De esta forma, al repartir diferentes componentes de software en diferentes dispositivos, obtenemos una distribución física natural de las funciones a través de una colección de ordenadores.

En arquitecturas descentralizadas, a menudo vemos que los procesos que constituyen un sistema distribuido desempeñan un papel similar, pero en un modo dinámico, sin depender de roles estáticos, ni comunicaciones predefinidas, al contrario, cada nodo puede ser cliente y servidor en virtud del proceso que se ejecuta. En este estilo de diseño, los procesos pueden utilizar listas aleatorias para comunicarse unos con otros.

A partir del concepto de sistemas distribuidos montados sobre una arquitectura descentralizada, nacen las redes P2P (Peer to Peer o par a par): una red de ordenadores en la que todos o algunos aspectos funcionan sin clientes ni servidores fijos, al contrario, se comportan como iguales entre sí. Esto implica que actúan simultáneamente como clientes y servidores respecto a los demás nodos de la red. Las redes P2P permiten el intercambio directo de información, en cualquier formato, entre los ordenadores interconectados.

En mayo de 1999, con una población de cientos de millones de personas subidas a Internet, Shawn Fanning introdujo una aplicación para compartir música y archivos llamada Napster[24]. Este fue el comienzo de las redes peer-to-peer, como las conocemos hoy en día, donde "los usuarios que participan pueden establecer una red virtual, totalmente independiente de la red física, sin tener que obedecer a cualquier autoridad administrativa o restricciones".

Las redes P2P tienen ciertas características que la convierten en una arquitectura de sistemas distribuidos capaz de satisfacer distintas implementaciones, sobre todo, en sistemas y datos que no dependan de una autoridad central:

- ✓ Escalabilidad. Las redes P2P tienen un alcance mundial con cientos de millones de usuarios potenciales. Cuantos más nodos estén conectados a una red P2P, mejor será su funcionamiento. Así, cuando los nodos se suman a la red y comparten sus propios recursos, aumentan la disponibilidad total del sistema. Esto es diferente en una arquitectura del modo servidor-cliente con un sistema fijo de servidores, en los cuales la adición de clientes podría significar una transferencia de datos más lenta para todos los usuarios.

- ✓ Robustez. La naturaleza distribuida de las redes peer-to-peer también incrementa la robustez en caso de haber fallos, puesto que implementa un

[24] Napster es un servicio de distribución de archivos de música (en formato MP3).

esquema de réplica excesiva de los datos hacia múltiples destinos, permitiendo a los peers encontrar la información sin hacer peticiones a ningún servidor centralizado de indexado. En el último caso, no hay ningún punto singular de falla en el sistema.

- ✓ Descentralización. Estas redes por definición son descentralizadas y todos los nodos son iguales. No existen nodos con funciones especiales, y por tanto ningún nodo es imprescindible para el funcionamiento de la red.

- ✓ Distribución de costes entre los usuarios. Se comparten o donan recursos a cambio de recursos. Según la aplicación de la red, los recursos pueden ser archivos, ancho de banda, ciclos de proceso o almacenamiento de disco.

- ✓ Anonimato. Es deseable que en estas redes quede anónimo el autor de un contenido, el editor, el lector, el servidor que lo alberga y la petición para encontrarlo, siempre que así lo necesiten los usuarios.

- ✓ Seguridad. Es una de las características deseables de las redes P2P menos implementada. Los objetivos de un P2P seguro serían identificar y evitar los nodos maliciosos, el contenido infectado, el espionaje de las comunicaciones entre nodos, etc. Se avanzó con la implementación de varios mecanismos: cifrado asimétrico, gestión de derechos de autor (la industria define qué puede hacer el usuario; por ejemplo, la segunda vez que se oye la canción se apaga), reputación (permitir acceso sólo a los conocidos), comunicaciones seguras, comentarios sobre los ficheros, etc.

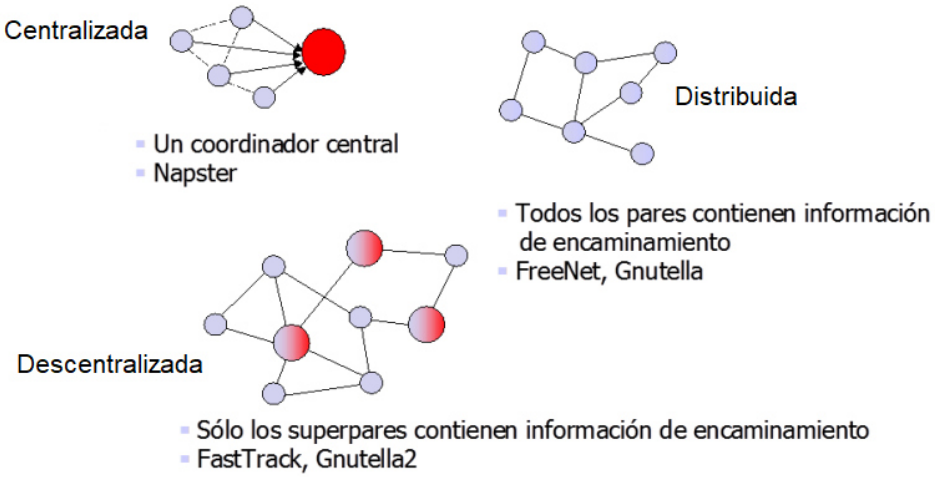

Figura 4.4.1: Arquitecturas de Redes P2P

Algunas implementaciones de redes P2P que se basan en algunas de las 3 arquitecturas descriptas en el gráfico 4.4.1, se detallan a continuación:

- ✓ Sistemas de ficheros distribuidos, como Gnutella[25] o Freenet[26], en ambos casos están montados sobre un modelo P2P puro.
- ✓ Un sistema que proporcionan anonimato, como i2p[27], responde a un tipo tecnología conocida como red oscura y representa al modelo P2P distribuido y anónimo.
- ✓ Sistemas como Napster y Audiogalaxi[28] responden a modelos P2P centralizado.

4.5. Conclusiones del capítulo

El desarrollo de los sistemas distribuidos nos permite crear soluciones sobre arquitecturas totalmente descentralizadas, en dónde no hay dependencias de una entidad única. Las redes P2P son el resultado de aplicar diseños totalmente distribuidos, sin un control centralizado, donde todos los participantes son un punto de acceso y un punto de control para el sistema.

La construcción de este tipo de redes son una invalorable aproximación para encontrar una solución al objetivo planteado en el presente trabajo, puesto que nos permite pensar en un sistema distribuido, descentralizado, formado quizás por cientos o miles de nodos que pueden de alguna forma almacenar los registros bajo algún modelo criptográfico. Corromper a una importante cantidad nodos que tienen almacenada información encriptada y replicada, es por lo menos una actividad que tiene un gran costo en términos de conocimientos técnicos, tiempo y capacidad computacional.

[25] Proyecto de software distribuido para crear un protocolo de transferencia de archivos entre pares
[26] Red de distribución de información sin censura que promueve la libertad entre los participantes
[27] I2P (sigla para Invisible Internet Project, que significa Proyecto de Internet invisible) es un software que permite la creación de herramientas y aplicaciones de red con un fuerte anonimato
[28] Audiogalaxy comenzó un motor de búsqueda mp3 para los participantes de la red. En 2012 lo compró dropbox

5. Capítulo 5. Conceptos de Criptografía

Cualquiera fuera el sistema distribuido que se va a implementar debe contemplar la ejecución de los procesos y la gestión de los datos en un ambiente seguro, manteniendo la integridad, confidencialidad, disponibilidad y el no repudio de la información que es generada y almacenada.

En la implementación de una solución de almacenamiento de registros de auditoría que pretenda ser seguro y confiable, debe inexorablemente contemplarse la implementación de un esquema de criptografía haciendo uso de una Infraestructura de Clave Pública, que sea la proveedora de los certificados digitales, para ser usados por los procesos y los usuarios.

Es recomendable conocer definiciones como certificado digital, claves, firma digital, cifrado de datos, etc., con el fin de usar en forma eficiente las herramientas de criptografía, para proteger a usuarios, procesos y los datos que ellos generan, entendiendo que existe una real necesidad de "encubrir" toda información sensible, desde su generación hasta su almacenamiento.

La criptografía es el nombre genérico que une dos disciplinas opuestas y a la vez complementarias: criptografía y criptoanálisis. La criptografía se ocupa del diseño de procedimientos para cifrar; es decir, para enmascarar una determinada información de carácter confidencial. El criptoanálisis se ocupa de romper esos procedimientos para así recuperar la información. Ambas disciplinas siempre se han desarrollado de forma paralela, pues cualquier método de cifrado lleva siempre emparejado su criptoanálisis correspondiente. La figura 5.1 muestra la dinámica del proceso criptográfico: a partir de un mensaje original que se desea proteger, al cual se le aplica un proceso de cifrado por el cual será ilegible en todo su recorrido, para aplicar en el destino un proceso de descifrado que lo convertirá en el mensaje original.

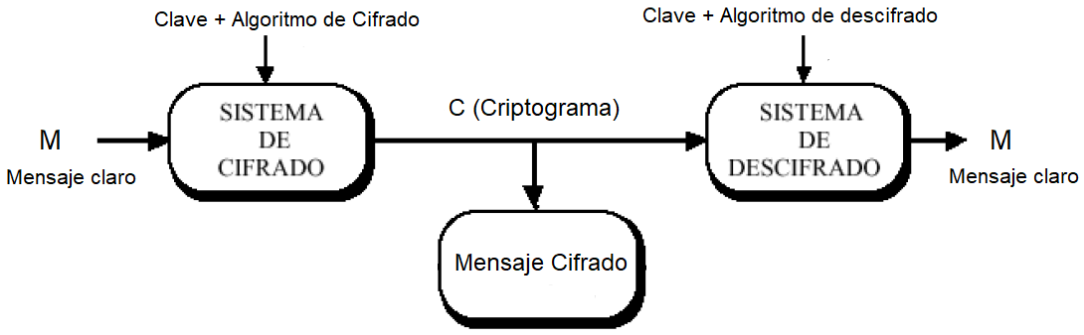

Figura 5.1: Proceso Criptográfico

A y B son el emisor y el receptor de un determinado mensaje. A transforma el mensaje original mediante un determinado procedimiento de cifrado controlado por una clave, en un mensaje cifrado (llamado también criptograma) que se envía por un canal

público. En recepción, B con conocimiento de la clave transforma ese criptograma en el mensaje original.

Cuando la información se traslada de A hasta B, el criptograma puede ser interceptado por un enemigo criptoanalista que lleva a cabo una labor de desencriptado; es decir intenta, a partir del criptograma y con diferentes técnicas "desencriptar" o sea recuperar el mensaje original. Un buen sistema criptográfico será, por tanto, aquel que ofrezca un descifrado sencillo (conociendo la clave), pero un desencriptado imposible o muy difícil si no se cuenta con ella.

La criptografía moderna se enfoca en el concepto de comunicaciones seguras, y persigue tres objetivos:

- ✓ Mantener la confidencialidad del mensaje; es decir, que solo sea visto por aquellos que tienen que ver la información
- ✓ Certificar la autenticidad del destinatario y remitente; es decir, asegurar la identidad del destinatario y remitente.
- ✓ Garantizar la integridad en el mensaje; es decir, que el mensaje enviado por el emisor sea el mismo que reciba el receptor

5.1. Criptografía: claves simétricas, asimétricas y función hash

Para lograr los objetivos descriptos (confidencialidad, autenticidad e integridad), la criptografía se vale de tres herramientas básicas, sistemas de clave simétrica, sistemas de clave asimétrica y funciones hash:

1. Sistemas de clave simétrica o única: son aquellos en los que los procesos de cifrado y descifrado son llevados a cabo por una única clave. Esta forma de cifrado utiliza una clave secreta, denominada *"secreto compartido"*, compartida por emisor y receptor. El receptor necesita la clave secreta para desbloquear los datos, esto lo hace por medio de un algoritmo de cifrado. Se denomina criptografía simétrica porque tanto para cifrar como para descifrar se necesita la misma clave. Tiene la desventaja de la distribución de la clave, puesto que al usar un medio público puede ser interceptada. La ventaja está en la rapidez de los algoritmos simétricos. Algunas implementaciones: AES, DES, 3DES, IDEA[29], etc. La figura 5.1.1 describe el proceso de cifrado y descifrado en un sistema simétrico.

[29] AES (Advanced Encryption Standard), DES (Data Encryption Standard), IDEA (International Data Encryption Algorithm), son todos algoritmos de clave simétrica.

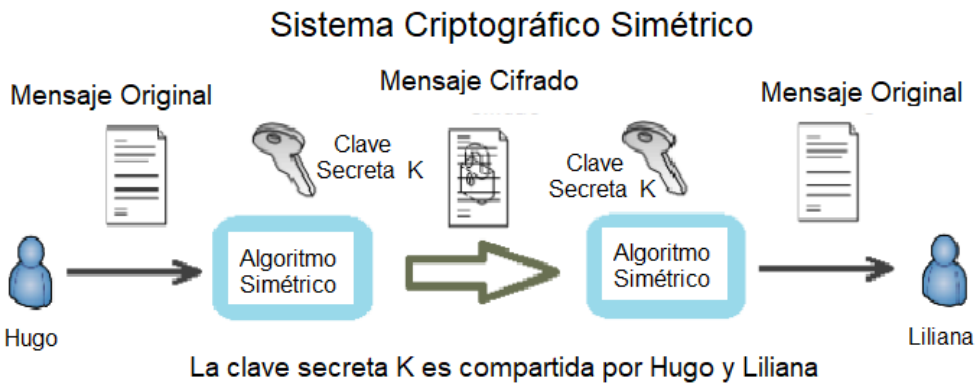

Figura 5.1.1: Sistema de clave simétrica

2. Sistemas de clave pública o asimétrica (Criptografía de clave pública): son aquellos en los que los procesos de cifrado y descifrado son llevados a cabo por dos claves distintas y complementarias. Esta forma de cifrado utiliza dos claves: una clave es secreta y una clave pública. El mensaje lo ciframos con la clave pública del destinatario. Este puede descifrar el mismo con su propia clave privada, que la mantiene en secreto. La diferencia de este sistema es que nadie necesita la clave privada de otro para poder enviar un mensaje en forma segura, cada vez que Hugo le desea enviar un mensaje a Liliana, utiliza la clave pública de Liliana, la cual no necesita ser mantenida en secreto. Luego, Liliana será la única que podrá descifrar el mensaje con su clave privada. Estos sistemas son muy seguros en cuanto a la gestión de claves, pero los algoritmos son más lentos que los simétricos. En este tipo de solución tanto el emisor como el receptor deben confiar en una entidad que se conoce como Autoridad de Certificación. Algunas implementaciones: Diffie-Hellman, RSA, DSA, ElGamal, Merkle-Hellman[30], etc. La figura 5.1.2 describe el proceso de cifrado y descifrado en un sistema asimétrico.

[30] Diffie-Hellman, debido a sus autores, Whitfield Diffie y Martin Hellman; RSA debido a sus autores Rivest, Shamir y Adleman; DSA (Digital Signature Algorithm); ElGamal, debido a su autor un reconocido criptógrafo egipcio; Merkle-Hellman, inventado por Ralph Merkle y Martin Hellman

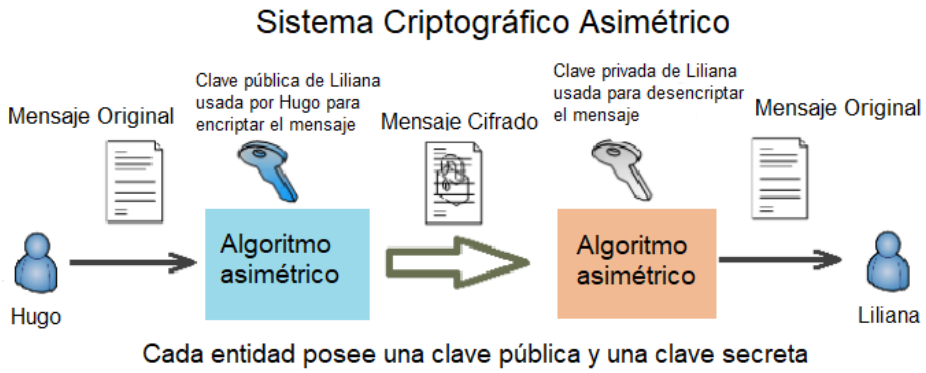

Figura 5.1.2: Sistema de clave Asimétrica

3. Función hash: La función hash seguro toma un flujo de datos y lo reduce a un conjunto de datos más pequeño, utilizando una función matemática unidireccional. El resultado se denomina resumen de mensaje y puede considerarse como la huella digital de los datos. El resumen del mensaje puede ser reproducido por cualquier entidad aplicando la misma función al mismo flujo de datos, pero es prácticamente imposible crear un flujo de datos diferente que produzca la misma huella digital (mismo resumen del mensaje).

Se puede usar un resumen de mensaje para proporcionar integridad. Si Hugo envía un mensaje y su resumen a Liliana, la receptora puede volver a calcular el resumen del mensaje, para corroborar que no hubo cambios accidentales los datos. La figura 5.1.3 muestra el proceso para obtener un valor hash partiendo de un mensaje al que se le aplica la función correspondiente.

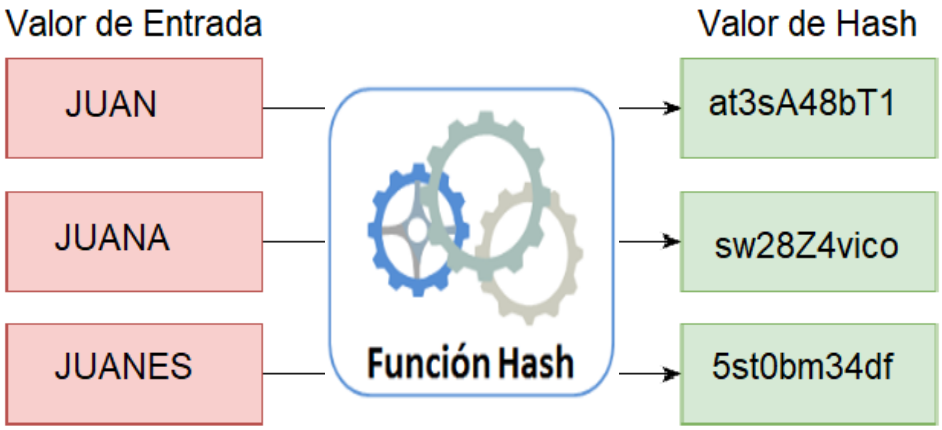

Figura 5.1.3: Función Hash

5.2. Aplicación de la Criptografía

Hemos visto que se puede usar criptografía de clave pública de forma tal que un usuario "Hugo" envía datos a "Liliana", cifrando los mismos con la clave pública de la receptora, y ésta puede ver la información que le envió Hugo, gracias a que cuenta con su clave privada o secreta, la cual utiliza para descifrar el mensaje enviado. Esta combinación permite implementar exitosamente el concepto de confidencialidad de la información, en este caso, los datos que Hugo le envía a Liliana, y que sólo ella puede leer por ser la única entidad que posee la clave secreta en toda la red.

Por otra parte, Hugo puede adosarle a los datos enviados a Liliana, el resultado de aplicarle a los mismos una función hash segura, para luego firmar este resultado con su clave privada. Cuando Liliana reciba los datos, más el resultado del hash firmado con la clave privada de Hugo, descifrará la firma recibida con la clave pública de Hugo, en ese momento obtendrá el valor de la función hash seguro a la que llamaremos H1. Luego aplicará la función hash seguro sobre los datos recibidos y obtendrá un H2. Si H2 es igual a H1, podrá concluir dos cosas: que el archivo fue enviado verdaderamente por Hugo (firma digital) y que el mismo no fue alterado (integridad). La figura 5.2.1 muestra el proceso de adosar y corroborar una firma digital.

El concepto de sistemas asimétricos, implementados bajo una infraestructura de clave pública, constituye el principal resultado de la aplicación de la criptografía para dotar de integridad, confidencialidad y no repudio a la información generada por los sistemas distribuidos modernos.

A partir de contar con una PKI o usar certificados digitales de autoridades de certificación confiables, es posible desplegar el uso de los mismos para asegurar servidores de aplicación, correo electrónico, páginas web, establecimiento de comunicaciones seguras punto a punto, etc. Por ejemplo, se puede instalar un certificado digital en un servidor de contenido web para cifrar la comunicación entre éste y sus clientes, usando el protocolo HTTPS[31] y encriptando el envío y recepción de datos a través de SSL (Secure Socket Layer)[32].

[31] Hypertex Transfer Protocol Secure, es un protocolo de aplicación basado en la transferencia encriptada de hipertexto (páginas web).
[32] Secure Socket Layer (Capa de puerto seguro), es un protocolo criptográfico que proporciona comunicaciones seguras en internet

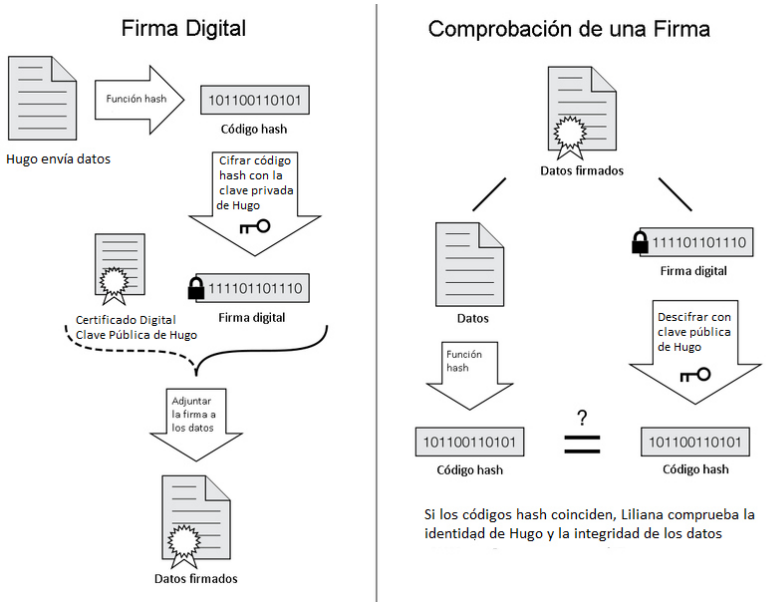

Figura 5.2.1: Firma Digital

5.3. Definición de Infraestructura de Clave Pública (PKI)

La infraestructura de clave pública (Public Key Infrastructure – PKI) es la combinación de software, tecnologías de cifrado y servicios que permite a cualquier sistema interconectado por una red de voz, datos y video, proteger la seguridad de sus comunicaciones y las transacciones a lo largo de toda la red. Public Key Infrastructure integra certificados digitales, criptografía de clave pública y autoridades de certificación en una arquitectura que convierte la red de una organización, en una autopista confiable para que circulen sus datos, en un estado de integridad y confidencialidad. Una infraestructura de seguridad típica aplicada a las operaciones de una la empresa incluye la emisión de certificados digitales a usuarios individuales y servidores; software de inscripción de usuarios finales; integración con directorios de certificados; herramientas para administrar, renovar y revocar certificados; y servicios y soporte relacionados [11]

El término infraestructura de clave pública se deriva de la criptografía de clave pública, la tecnología en la que PKI se basa. La correcta implementación de este tipo de infraestructura tiene características únicas que la hacen invaluable al momento de considerar funciones de seguridad en sistemas distribuidos.

5.4. Componentes de Infraestructura de Clave Pública (PKI)

El componente principal es el Certificado Digital, el cual es un documento público en el cual aparece información relativa a un usuario, servidor u otra entidad, que se denomina sujeto y que sirve para asociar la clave pública del mismo con su identidad.

Un certificado digital tiene un formato standard X.509[33], y contiene información que identifica al sujeto, a su clave pública y a la entidad que emitió el certificado (Autoridad de Certificación).

La Autoridad de Certificación (AC) es la encargada de la emisión dentro de una jerarquía de autoridades, y todos los componentes del sistema distribuido deben confiar en ella. La forma que tiene una AC de dar validez a un certificado digital, es incluir en él su firma. Todos los estándares que rigen una PKI se denominan PKCS (Public Key Cryptography Standars) seguido de un número, por ejemplo PKCS#1 (trata el standard criptográfico de RSA[34]).

La AC es una pieza fundamental en la implementación de PKI, dada que es la entidad que firma los certificados digitales emitidos, y los revoca cuando existen motivos para hacerlo. Dado que la firma digital de una AC es la clave del funcionamiento de todo el sistema, es necesario dotar con el máximo de seguridad lógica y física a la misma. Se puede optar por confiar en AC externas a la organización.

La Autoridad de Registro (AR) es la encargada de llevar a cabo los procedimientos de registración de los solicitantes de certificados digitales. Una vez que ha comprobado fehacientemente la identidad del solicitante (puede ser presencial o remota, cada AR define los requisitos), procede a generar una solicitud de certificado a la AC. Una vez que la AC aprueba y firma el certificado, el mismo se ubica en un directorio público para su uso.

La AR tiene también la responsabilidad de recibir las solicitudes de revocación de los certificados emitidos. Luego de implementar procesos para verificar las causas del pedido de revocación, emitirá una solicitud de revocación de certificado a la AC, la cual, luego de firmarlo, lo ubicará en un directorio público de certificados revocados, para que sea consultado cada vez que fuera necesario.

Un repositorio es una base de datos de certificados digitales activos para un sistema mantenido por una AC. La función principal del repositorio es proporcionar datos que permitan a los usuarios confirmar el estado de los certificados digitales para individuos y empresas que reciben mensajes firmados digitalmente. Las AC publican certificados y CRL (Certified Revocation List – Lista de Certificados Revocados) en repositorios.

Los usuarios de PKI son organizaciones o individuos que usan la PKI, pero no emiten certificados. Ellos confían en los otros componentes de la PKI para obtener certificados y verificar los certificados de otras entidades con las que intercambian datos y procesos. Las entidades finales confían en el certificado emitido por una AC, y así conocen con certeza la clave pública de otra entidad y al titular del certificado, el cual seguramente va a firmar documentos digitales.

La figura 5.4.1 muestra todos los componentes y los procesos asociados que forman parte de una PKI.

[33] Es un estándar UIT-T para infraestructuras de claves públicas, cuya última versión fue publicada en 2008.
[34] RSA (Rivest, Shamir y Adleman) es un sistema criptográfico de clave pública desarrollado en 1979

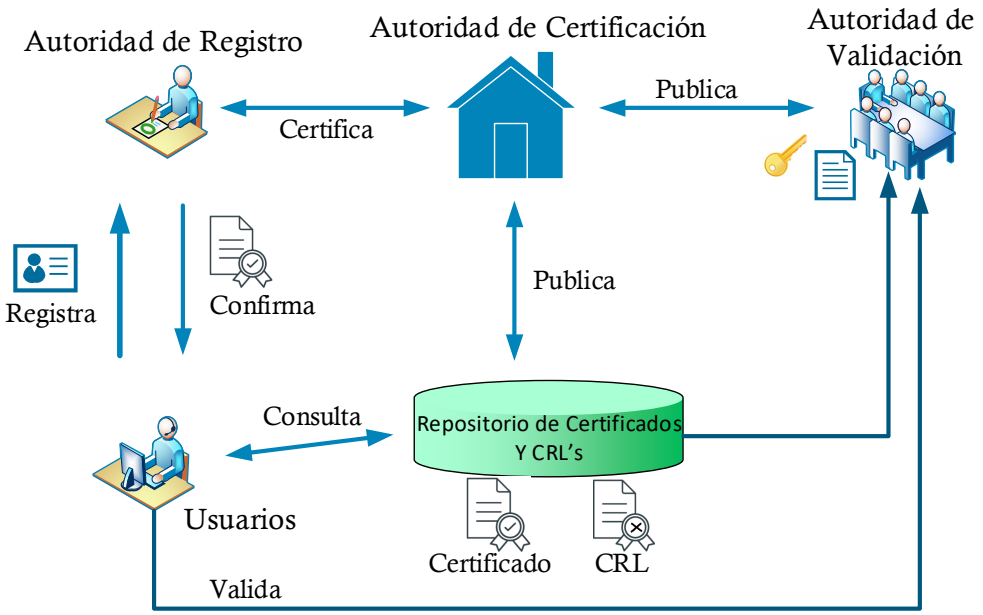

Figura 5.4.1: Infraestructura de Clave Pública (PKI)

5.5. Conclusiones del capítulo

Las infraestructuras de clave pública (PKI) pueden acelerar y simplificar la comunicación entre entidades y servicios al proporcionar enfoques electrónicos a los procesos de autenticación que históricamente han sido basados en papel.

Estas soluciones que distribuyen certificados digitales, implementadas sobre complejos sistemas distribuidos, ayudan a garantizar la integridad y autenticidad de los datos. Ambas se pueden lograr vinculando un certificado digital único a un individuo, un servidor o un proceso, asegurando que el mismo no se puede falsificar. El emisor puede firmar datos digitalmente y el destinatario puede verificar tanto al creador de los datos y como así también que los mismos no han sido modificado.

Incorporar a un sistema de gestión de logs, procesos y usuarios debidamente identificados por certificados digitales, colabora con la integridad y la autenticidad de los eventos almacenados.

Por lo expuesto, es claro que la criptografía, en particular la infraestructura de clave pública, es un elemento clave que deberá ser contemplado en cualquier solución que proponga el presente trabajo.

6. Capítulo 6. DLT (Tecnologías de libro de cuentas distribuido)

Las tecnologías basadas en la distribución de la información "contable" replicada en diversos nodos representan el nuevo paradigma de compartición de la información que toda empresa deberá dominar, según la visión de la prestigiosa empresa Accenture (brinda soluciones tecnológicas en 120 países), publicado a través de reporte en 2019 ([12] . Según éste, las empresas líderes deberán afrontar el reto de implementar y crecer en cuatro áreas tecnológicas bien definidas, conocida con la sigla DARQ (Distributed Ledger Technologies (DLT), Artificial Intelligence (AI), Extended Reality (XR) y Quantum Computing (Q)).

La definición de DLT hace referencia a un libro de cuentas distribuido (Ledger). La información se almacena en forma distribuida entre los nodos de la red. Los nodos son servidores independientes que tienen la capacidad de registrar, compartir, sincronizar y validad transacciones.

Las DLT pueden clasificarse de acuerdo a la forma en que los nodos y los usuarios pueden participar de ella. Basados en diversas publicaciones consultadas se puede establecer la siguiente taxonomía:

- ✓ Pública: no existen restricciones en la lectura, ni en el envío de transacciones para agregar al libro de cuentas.
- ✓ Privada: el acceso a la información y a las transacciones está limitada a entidades predefinidas
- ✓ Sin permiso (permissionless): no hay restricciones en los nodos que deben procesar las transacciones
- ✓ Con permiso (permissioned): el procesamiento de las transacciones es realizado por nodos debidamente identificados

La primera implementación de una DLT fue Bitcoin, la cual se desarrolló en forma de una cadena de bloques o blockchain, pero existen otras que abandonan la idea de la cadena de bloques como ser la red Tangle de IOTA[35] o HashGraft[36].

Con frecuencia se emplean los términos DLT o blockchain como sinónimos. Ambos son conceptos y no representan implementaciones como Bitcoin, Ethereum o Hyperledger. Es preciso tener claro entonces que blockchain es un tipo de DLT, y existen DLT's que no son blockchain.

En este capítulo se va a describir la tecnología de blockchain con el objeto de considerar la misma como una herramienta para lograr el objetivo propuesto.

[35] IOTA es un protocolo de contabilidad distribuida que en lugar de utilizar blockchain, implementa una arquitectura conocida como Tangle.

[36] Hashgraph es una tecnología de contabilidad distribuida desarrollada por Leemon Baird, cofundador y CTO de Swirlds, en 2016. No utiliza mineros para validar transacciones, y utiliza gráficos acíclicos dirigidos para transacciones de secuencia de tiempo sin agruparlos en bloques.

6.1. Blockchain

Las tecnologías Blockchain están arrasando en el mundo, en gran parte debido al cambio de paradigma que ha generado Bitcoin [13]. Una cadena de bloques, también llamada libro mayor distribuido, es simplemente una estructura de datos mantenida por un conjunto de nodos que no confían plenamente el uno en el otro, y que son administrados por entidades que no tienen ninguna relación entre ellas.

Los nodos en la cadena de bloques acuerdan un conjunto ordenado de bloques, cada uno con múltiples transacciones, también ordenadas cronológicamente, por lo que la cadena de bloques se puede ver como un registro de transacciones ordenadas, que se encuentra exactamente replicada en cada nodo perteneciente a la red de nodos.

Desde el punto de vista de una base de datos, se puede ver a Blockchain como una solución para la gestión de transacciones distribuidas: los nodos mantienen réplicas de los datos y acuerdan un orden de ejecución de transacciones. Sin embargo, las bases de datos tradicionales suponen un entorno confiable y emplean el concepto de concurrencia para ordenar transacciones. En Blockchain la propiedad clave es que supone que los nodos se comportan en de manera arbitraria (o bizantina). Ser capaz de tolerar una falla en un diseño bizantino, hace de Blockchain un sistema extraordinariamente seguro, mucho más que el de las bases de datos.

En el diseño original, la cadena de bloques de Bitcoin almacena monedas como dice el sistema. Los nodos de Bitcoin implementan un modelo de máquina de estado replicado simple que mueve monedas de una dirección a otra. Desde entonces, Blockchain ha crecido más allá de las criptomonedas y ser considerado como una herramienta para implementar soluciones en una gran diversidad de sistemas distribuidos, etc.

Un buen ejemplo del uso de Blockchain más allá de las criptomendas es Ethereum [14], una solución que permite ejecutar aplicaciones descentralizadas y replicadas, conocidas como “contratos inteligentes”. La industria ha comenzado a impulsar nuevas plataformas de blockchain diseñadas para brindar soluciones específicas donde los participantes se autentican. Estas últimas son conocidas como blockchain privadas, a diferencia de los primeros sistemas que operan en entornos públicos (cualquiera puede unirse, operar y retirarse).

En la actualidad, la tecnología de blockchain es utilizada en numerosos ámbitos y organizaciones (Figura 6.1.1), desde sistemas que mantienen contratos en el área de marketing, aplicaciones que implementan soluciones de votación electrónica, sistemas de mantenimiento de historias clínicas en el área de la salud y la seguridad social, implementación de sistemas de logística, aplicaciones en el área fiscal de los estados, soluciones de para sistemas de inversión de capitales, sistemas financieros, y hasta en el entorno de la educación según lo tiene planeado la Unión Europea.

Figura 6.1.1: Blockchain es una solución que tiene una vasta aplicación

Todos los sistemas antes mencionados fueron implementados con un modelo basado en bases de datos de clase empresarial, compatible con soluciones como MS SQL Server, Oracle, DB2, etc. Sin embargo, blockchain tiene la capacidad de motivar la migración a nuevas soluciones, porque está basada en un despliegue de recursos menos costoso a nivel hardware, aplicaciones y recursos humanos. De todas formas, hay algo que sobresale nítidamente a la hora evaluar una solución basada en cadenas de bloques: la inmutabilidad y la transparencia ayudan a reducir el riesgo humano errores y la necesidad de intervención manual debido a conflictos datos

6.2. Arquitectura de una blockchain

Como su nombre lo indica, las transacciones se almacenan en un bloque, hasta que éste llega a su tamaño máximo, a partir del cual se genera un próximo bloque en el cual se insertarán las nuevas transacciones. El segundo bloque se encadena con un

puntero al hash de la cabecera del primer bloque. Este proceso continúa, en un modelo donde el bloque i se encadena al hash de la cabecera del bloque i-1.

Todos los nodos que participan de la red de blockchain tienen una copia de cada bloque y cuando éste se llena, se dispara un algoritmo de consenso, de forma tal que un solo nodo "confirma" el contenido del bloque lleno y lo replica a los demás. Todas las transacciones se transmiten a la red utilizando software. Los mensajes se entregan utilizando un servicio orientado al "mejor esfuerzo"[37].

Los bloques se encadenan a través de una estrategia por la cual, cada bloque que contiene el hash de la cabecera del bloque anterior. Si se cambiara un bloque previamente publicado, provocaría que cambie su hash de cabecera. Esto a su vez haría que todos los bloques posteriores también tengan diferentes hashes ya que incluyen el hash del bloque anterior. Esto permite detectar fácilmente y rechazar cualquier cambio en los bloques publicados anteriormente.

Se llama bloque génesis al primer bloque de cada blockchain. Este tipo de bloques sirve para iniciar una cadena de bloques. Cualquier solución basada en bloques encadenados, tendrá un bloque de similares características que inicia su propia blockchain.

La figura 6.2.1 muestra la estrategia de crecimiento de la cadena de bloques, cuyo inicio es el bloque génesis.

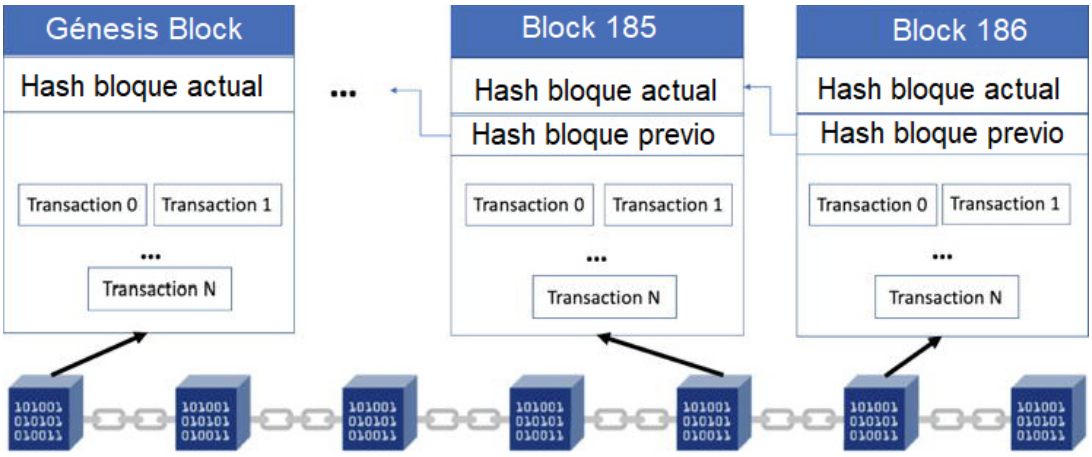

Figura 6.2.1: Cadena de bloques
Fuente: [15]

Con el objeto de comprender con mayor detalle el contenido de un bloque, se muestra la figura 6.2.2, la cual es una versión ampliada la arquitectura de la solución de blockchain:

[37] La entrega de mejor esfuerzo describe un servicio de red en el que la red no proporciona ninguna garantía de que se entreguen datos o que la entrega cumpla con alguna calidad de servicio determinada.

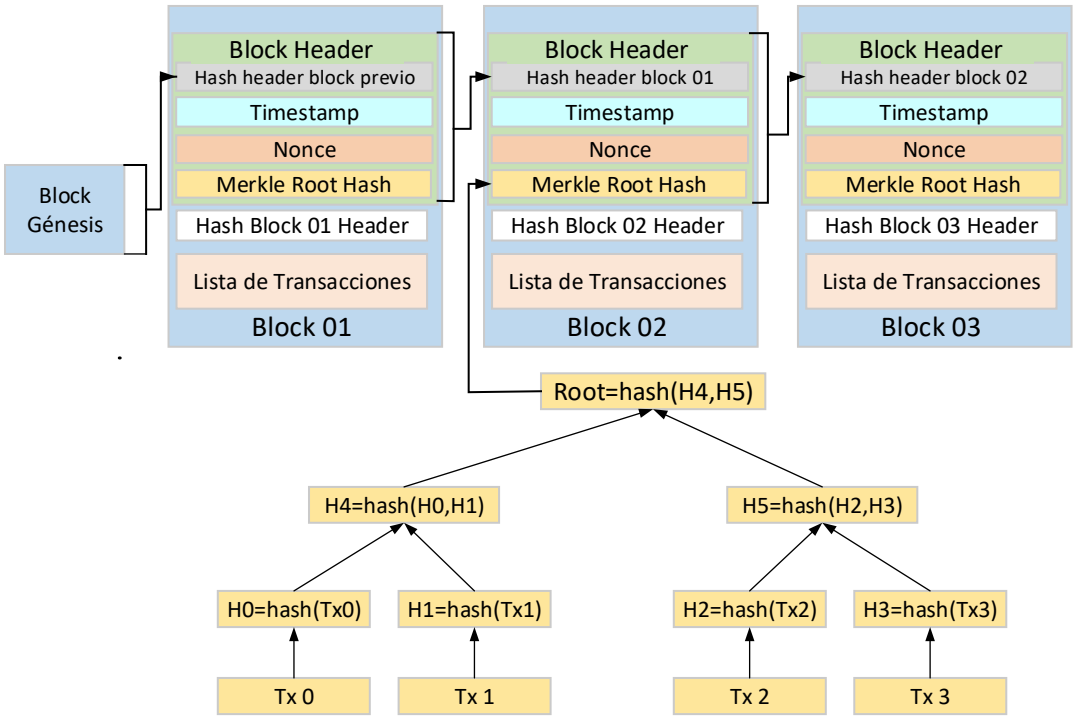

Figura 6.2.2: Arquitectura de Blockchain
Fuente**:** [16]

La descripción de cada uno de los campos que forman la estructura de un bloque es la siguiente:

- ✓ El número de bloque, también conocido como altura de bloque (forma parte de la cabecera)
- ✓ El valor hash del bloque anterior (forma parte de la cabecera)
- ✓ El hash de raíz del árbol de Merkle (forma parte de la cabecera)
- ✓ Una marca de tiempo, timestamp (forma parte de la cabecera)
- ✓ El valor hash del bloque actual
- ✓ El tamaño del bloque
- ✓ El valor "nonce", que es un número manipulado por el nodo de minería para resolver el hash rompecabezas que les da derecho a publicar el bloque
- ✓ Una lista de transacciones incluidas dentro del bloque

6.3. Características de una blockchain

Toda blockchain está compuesta por características bien definidas, que en general la convierten en una solución de extrema confiabilidad cuando es necesario

mantener la inmutabilidad de la información y la trazabilidad de los datos. Tal es la robustez de la arquitectura de una cadena de bloques, que se constituyó en la solución base para el intercambio de monedas virtuales como el bitcoin. A continuación, una descripción detallada de las características de una cadena de bloques encriptada:

- ✓ Descentralización: En un sistema convencional de transacciones centralizadas, cada transacción debe ser validada a través de un proceso central. En este modelo es inevitable pagar el costo por los cuellos de botellas del servidor. En contraste al modo centralizado, ya no se necesitan terceros en blockchain. Los algoritmos de consenso en blockchain se usan para mantener consistencia de datos en una red distribuida, con procesos distribuidos.
- ✓ Persistencia. Las transacciones se pueden validar rápidamente y las transacciones inválidas no serán admitidas por los nodos mineros Es casi imposible eliminar o revertir transacciones una vez que se incluyen en la cadena de bloques. Se pueden descubrir bloques que contienen transacciones no válidas inmediatamente.
- ✓ Anonimato. Cada usuario puede interactuar con blockchain con una dirección generada, que no revela la identidad del usuario.
- ✓ Auditable: por ejemplo en el caso de Bitcoin, cualquier transacción debe referirse a alguna transacción anterior no gastada. Una vez que la transacción actual es registrada en blockchain, el estado de esa transacción no gastada cambia del estado *"no gastada",* al estado de *"gastada"*. Así que las transacciones pueden ser fácilmente verificadas y rastreadas.
- ✓ Firma digital: Cada usuario posee un par de claves: privada y clave pública. Se utiliza la clave privada que se mantiene en secreto para firmar las transacciones. Las transacciones firmadas digitales son transmitidas en toda la red.

Por su taxonomía, los sistemas de cadenas de bloques se clasifican en tres tipos: blockchain públicas, blockchain privadas y blockchain consorcio [17].

En las blockchain públicas, todos los registros son visibles para el público y todos podrían participar en el consenso proceso. Sus características se detallan a continuación:

- ✓ Cualquier persona puede mantener la base de datos distribuida
- ✓ Cualquier persona puede acceder a la base de datos
- ✓ Cualquier persona puede realizar transacciones, las cuales se registrarán en la base de datos
- ✓ La base de datos en distribuida masivamente
- ✓ El incentivo para mantener la base de datos es económico-criptográfico. Se recompensa económicamente por una labor criptográfica.

En cuanto a las blockchain privadas, solo aquellos nodos pertenecientes a una organización específica y debidamente identificados se les permitirían unirse al proceso de consenso. Se puede destacar que:

- ✓ Una entidad o un grupo de entidades mantiene la base de datos
- ✓ Sólo las entidades preestablecidas pueden acceder a la base de datos, y se pueden limitar los derechos sólo a partes de estas
- ✓ Sólo las entidades con permisos podrán realizar transacciones que se registrarán en las bases de datos
- ✓ La base de datos se mantiene por un interés propio de las entidades participantes

Una blockchain consorcio es una suerte de combinación de las dos anteriores. De manera diferente, solo un grupo de nodos preseleccionados participaría en el proceso de consenso, pero las transacciones son públicas.

6.4. Algoritmos de consenso

En blockchain, llegar a un consenso para "minar" o "sellar" un bloque, a partir de nodos que no son confiables, es todo un desafío. En este tipo de solución no hay un nodo central que garantice que los libros de contabilidad en los nodos distribuidos sean todos exactamente iguales. Es clave la implementación de protocolos de consenso para asegurar que los libros de contabilidad en diferentes nodos sean consistentes. A continuación, presentamos algunos de los algoritmos de consenso que se implementan en blockchain:

- ✓ PoW – Proof of Work (Prueba de trabajo) es una estrategia de consenso utilizada en Bitcoin. En una red descentralizada, alguien tiene que ser seleccionado para registrar las transacciones. La forma más fácil es una selección aleatoria. Sin embargo, ésta es vulnerable a ataques. Entonces, si un nodo quiere publicar un bloque de transacciones, tiene que dedicar sus recursos a resolver un problema complejo, de esta forma se garantiza que el nodo no va a destinar recursos para atacar la red. Generalmente se trata de un "acertijo matemático". En PoW, cada nodo de la red está calculando un valor hash del encabezado del bloque. El encabezado del bloque contiene un "nonce" y los mineros cambiarían el "nonce" con frecuencia para obtener diferentes valores hash. El consenso requiere que el valor calculado debe ser igual o menor que cierto valor dado. Cuando un nodo alcanza el valor objetivo (resuelve el problema), debe avisar a los otros nodos, quienes confirmaran la corrección del valor hash. Si el bloque está validado, todos los mineros agregarán este nuevo bloque a sus propias cadenas de bloques. Los nodos que calculan los valores hash se llaman mineros (y reciben un pago por resolver el problema y agregar un nuevo bloque) y el procedimiento PoW se llama minería en Bitcoin

- ✓ PoS - Proof-of-Stake (Prueba de participación): Este algoritmo de consenso nace como una alternativa al PoW y tiene como dos objetivos: lograr un consenso distribuido y es una alternativa que ahorra mucha energía respecto a PoW. La estrategia requiere que un nodo apueste, mantenga o bloquee monedas y valide la propiedad de las mismas. En pocas palabras, es un mecanismo donde los bloques se validan de acuerdo a la "participación" de los involucrados. Si bien hay diferentes formas en que se seleccione el nuevo creador de bloques, para evitar la centralización por la cantidad de monedas que se pueden tener bloqueadas o en "apuesta" (nodos ricos), en general, la cadena de bloques está asegurada por un proceso de selección pseudoaleatorio que considera la riqueza del nodo y la antigüedad de las monedas junto con un factor de aleatorización.

- ✓ PBTF (Práctica de Tolerancia Bizantina): En un escenario dónde se supone pueden existir nodos maliciosos, un nuevo bloque se agrega luego de una serie de rondas de elecciones. En cada ronda, se selecciona un nodo de acuerdo con algunas reglas. Este es el responsable de ordenar la transacción. Todo el proceso puede dividirse en fases: pre-preparada, preparada y comprometida. En cada fase, un nodo entraría en la siguiente fase si ha recibido votos de más de 2/3 de todos los nodos. Entonces PBFT requiere que cada nodo sea conocido por la red. La Tolerancia a fallos bizantina práctica (PBFT), por ejemplo, garantizará la consistencia en un sistema con 3 * k + 1 nodos solo si el número de traidores no excede k. Por lo tanto, necesitarás al menos 2 * k + 1 votos para un bloque. La razón por la que necesita 2 * k + 1 es que por cada número menor que eso, no puede estar seguro de que los traidores no sean mayoría para este subconjunto

- ✓ DPoS - Delegated Proof of Stacke (Prueba de Participación Delegada): Al igual que en PoS, los mineros tienen su prioridad para generar nuevos bloques en base a su participación. La principal diferencia entre PoS y DPoS es que el primero es una democracia directa, mientras que DPoS es una democracia representativa. Las partes interesadas eligen a sus delegados para generar y validar un bloque. Con significativamente menos nodos compitiendo por validar el bloque, este podría confirmarse rápidamente, lo que permite ratificar las transacciones rápidamente. Mientras tanto, los parámetros de la red, como el tamaño de bloque y los intervalos de bloque, pueden ser recalculados con mayor facilidad. Además, los usuarios no necesitan preocuparse por los delegados deshonestos porque estos pueden ser echados fácilmente mediante votación

6.5. Contratos Inteligentes

Un contrato inteligente es una colección de código y datos (a veces denominados funciones y estado) que se implementa en una cadena de bloques. El contrato ejecuta el método apropiado con los datos proporcionados por el usuario para realizar un

servicio. El código, al estar almacenado en la blockchain, es inmutable y, por lo tanto, puede usarse (entre otros fines) como si existiese una tercera parte confiable. En una solución para el mercado financiero, un contrato inteligente puede realizar cálculos, almacenar información y enviar automáticamente fondos entre las cuentas que lo componen.

Un contrato inteligente es un programa que vive en un sistema no controlado por ninguna de las partes, o sus agentes, y que ejecuta un contrato automático el cual funciona como una sentencia if-then (si-entonces) de cualquier otro programa que resida en un computador. Tiene como objetivo brindar una seguridad superior a la ley de contrato tradicional y reducir costos de transacción asociados a la contratación (se elimina el intermediario, llámese banco, tarjeta de crédito, comisionista, etc.).

La transferencia de valor digital mediante un sistema que no requiere confianza (ej. bitcoins) abre la puerta a nuevas aplicaciones que pueden hacer uso de los contratos inteligentes. Además, también se puede aplicar a otros campos, servicios públicos, Internet de Cosas (IoT), sistemas electorales, en el campo de la medicina, el desarrollo de soluciones de logística, servicios de seguridad, etc.

6.6. Funcionamiento de una blockchain

El funcionamiento de una blockchain es como se describe en la figura 6.6.1: un usuario dispara una transacción invocando un contrato inteligente (1), el conjunto de nodos mineros o selladores escucha la transacción (2) y la almacena en su bloque junto a transacciones anteriores (3). Cuando un bloque se llena, comienza el proceso de minado, dónde todos los nodos compiten a través de un algoritmo de consenso. Un solo minero va a sellar el bloque, enlazando el mismo con la cabecera del bloque inmediato anterior (4). Los demás mineros recibirán una copia del bloque recién sellado (5). A partir de ese momento, la información que almacena el bloque es imposible de adulterar.

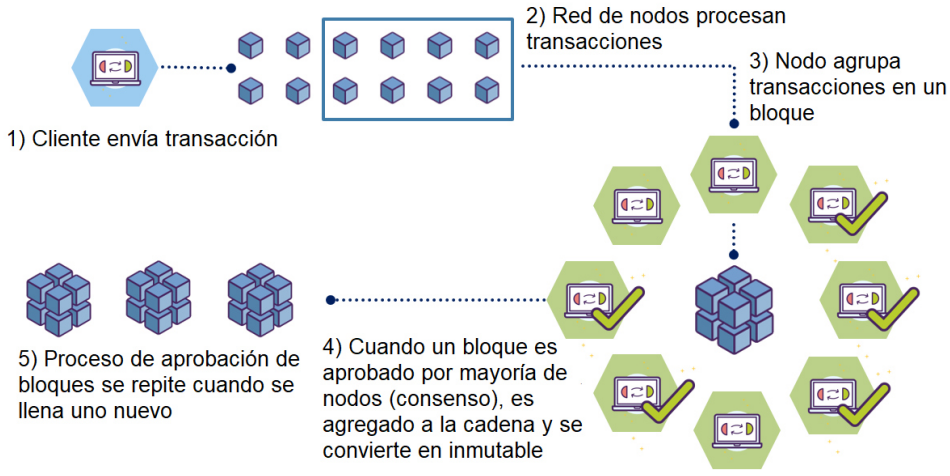

Figura 6.6.1: Funcionamiento de Blockchain
Fuente: https://bfa.ar/blockchain/

6.7. Implementaciones de blockchain

Una lista de blockchains implementadas, junto a sus propiedades más importantes se muestran continuación en la Tabla 6.7.1. Se incluyen los sistemas de cadenas de bloque más conocidos, teniendo en cuenta que la lista no es exhaustiva, especialmente dado el creciente interés comercial y académico que existe sobre la plataforma de bloques encadenados.

Tabla 6.7.1: Tabla comparativa de blockchains
Fuente: [18]

Blockchain	**Aplicación**	**Ejecución del Contrato Inteligente**	**Lenguaje del Contrato Inteligente**	**Modelo de datos**	**Consenso**
Hyperledger V.1.4.2	Aplicación general	Docker	Golang, Java, JavaS	Clave-Valor	Solo, Raft, Kafka
Bitcoin	Crypto-currency	Nativa	Golang, C++	Basado en Transacciones	PoW
Litecoin	Crypto-currency	Nativa	Golang, C++	Basado en Transacciones	PoW (memory)
ZCash	Crypto-currency	Nativa	C++	Basado en Transacciones	PoW (memory)
Ethereum	Aplicación General	EVM	Solidity, Serpent, LLL	Basado en Cuenta	PoW
Multichain	Activos Digitales	Nativa	C++	Basado en Transacciones	Validadores Confiados (Round Robin)
Quorum	Aplicación General	EVM	Golang	Basado en Cuenta	Raft
BigchainDB	Activos Digitales	Nativa	Phyton, cripto-condiciones	Basado en Transacciones	Validadores Confiados
Monax	Aplicaciones Generales	EVM	Solidity	Basado en Cuenta	Tendermint
Ripple	Activos Digitales			Basado en Cuenta	Consenso Ripple
Corda	Activos Digitales	JVM	Kotlin, JVM	Basado en Transacciones	Raft
Parity	Aplicaciones Generales	Nativa	Phyton	Clave-Valor	Validadores Confiados (Round Robin)

6.8. Conclusiones del capítulo

Las blockchains son sistemas de registración digitales inmutables implementados de manera distribuida (es decir, sin una estructura de datos centralizada) y generalmente sin una autoridad central (adiós al sistema intermediario), desplegados bajo la forma de una red P2P. En su nivel más básico, permiten que una comunidad de usuarios registre transacciones en un libro de "contabilidad", bajo un esquema de consenso basado en la mayoría de participantes; siendo las mismas protegidas por una arquitectura de criptografía encadenada, de modo tal que no se pueda cambiar ninguna transacción una vez que fue aceptada y publicada.

Es innegable que escuchar el concepto inmutable, bajo un esquema de seguridad implementado con infraestructura de clave pública, en un modelo de red distribuido, sin la necesidad de contar con un "servidor centralizado", le otorga a la tecnología blockchain un posicionamiento clave para ser elegida como la plataforma de la solución que permitirá alcanzar los objetivos del presente trabajo de tesis.

Por lo hasta aquí desarrollado: los sistemas distribuidos; la infraestructura de clave pública que cifran las comunicaciones y los datos almacenados; las redes P2P que permiten interactuar directamente entre los usuarios; los contratos inteligentes que ejecutan procesos y gestionan información, utilizando algoritmos de consenso, que permiten prescindir de una entidad central (servidor, sistema blade, cluster de servidores, etc.); los repositorios de datos replicados en distintos nodos, etc.; son todos elementos que van a aportar un gran valor en el momento de desarrollar una solución robusta y eficiente, que preserve información a lo largo del tiempo.

En un modelo de solución basado en blockchain, la capacidad de penetración y daño que puede desarrollar un atacante, con el objetivo de modificar, borrar, y/o inyectar información falsa en un sistema de logs de auditoría es lo suficientemente baja, tanto que requiere recursos humanos altamente capacitados además de un esfuerzo computacional y monetario importantes.

El atacante debería contar con una extraordinaria capacidad para descifrar un sistema de claves pertenecientes a todos los participantes de la cadena de bloques, como así también, cambiar el comportamiento de los nodos (convertirlos en nodos maliciosos al momento del consenso). Aun logrando los dos objetivos mencionados, al intentar realizar modificaciones en bloques que fueron sellados, la cadena se destruirá inmediatamente, porque como su nombre lo indica, los punteros que unen los bloques se encuentran enlazados a partir del valor hash del bloque anterior, valor que cambiará en el mismo momento que se modifique un solo bit dentro del mismo.

Sección 2: Solución Propuesta

7. Capítulo 7. Descripción de la solución

Siguiendo la metodología de investigación en ciencias del diseño de varios autores, entre ellos Peffers y otros [19], se van a describir los objetivos y requisitos para la solución.

El objetivo principal de un sistema de registro seguro se basa en tres pilares: mantener la disponibilidad, integridad y autenticidad de la información contenida en los registros almacenados.

En un sistema de registros de auditoría, cualquier log puede ser trascendental en la detección de un ataque, la penetración al sistema o la consumación de un delito. En este contexto, es clave preservar la evidencia. Entonces es necesario definir qué es la evidencia de un log.

La evidencia de un registro de auditoría se puede dividir en dos conceptos: evidencia del dato y prueba de integridad:

1. La evidencia del dato consiste en el evento real y los metadatos asociados. Específicamente consiste en la información de registro, un identificador con la firma del sistema generador (router, firewall, switch, etc.) y una marca de tiempo (instante del evento).

2. La prueba de integridad está representada por un hash de la evidencia más la marca de tiempo, los cuales confirman la existencia del evento registrado a una determinada hora. Si esta información se almacena de manera inmutable, más tarde puede ser usada para probar que los datos de evidencia no han cambiado desde la creación de la prueba.

La evidencia en la generación de logs (fase 1) se lleva a cabo en las fuentes (routers, switches, firewalls, etc.), en la medida que soporten la generación de logs firmados digitalmente, los cuales se firman con la clave privada de la fuente generadora.

La evidencia en la fase de transferencia (fase 2) consiste en la acción de transferir el evento desde su origen hacia el sistema de almacenamiento centralizado. En sistemas que requieren altos estándares de seguridad (como los sistemas financieros), es recomendable que la transmisión de logs de seguridad se lleve a cabo utilizando protocolos orientados a conexión a través de canales cifrados. Para ello es necesario dos cosas:

1. Implementar un mecanismo de transporte confiable, como por ejemplo el protocolo de transmisión de datos TCP (Transmission Control Protocol). Este es un estándar para la transmisión de datos orientado a conexión, lo cual implica que asegura la entrega de todos los segmentos con información que fueron enviados, como así también garantiza que sean ordenados en el destino.
2. Recurrir a un protocolo de tunelización para cifrar el canal entre el origen y el destino (se recomienda, sobre todo cuando se usan enlaces públicos). Algunos

protocolos de tunelización que se pueden implementar a través de redes privadas virtuales (VPN – Virtual Private Network) son: PPTP (Point to Point Tunneling Protocol), IPSec (Internet Protocol Security), L2TP (Layer 2 Tunneling Protocol), MPLS (Multiprotocol Label Switching), etc.

Para llevar a cabo ambas recomendaciones, es necesario que el hardware que produce los registros de auditoría que se pretenden enviar, soporte la configuración del protocolo TCP para el envío de registros de auditoría y de cualquiera de los protocolos de tunelización que ya fueron enunciados. Los switches, routers o firewalls que permiten configuraciones básicas (utilizados en la mayoría de las organizaciones de pequeña y mediana dimensión), en general no soportan estas prestaciones. Pese a ello, la mayoría se puede configurar usando un protocolo no orientado a conexión, como UDP (User Data Protocol) y de esta forma transferir los logs de eventos desde el origen al almacenamiento central.

La evidencia en la fase de almacenamiento (fase 3), representa el objetivo principal de la presente tesis. Cualquier log, junto con sus datos deben almacenarse de manera que conserven la disponibilidad e integridad para uso posterior. Esto incluye prevenir la modificación o eliminación perpetrada por negligencia o por un atacante que busca borrar huellas. La confidencialidad también es una preocupación debido a la potencial sensibilidad de la información contenida en los registros de anotaciones.

Luego de ser detalladas las tres instancias anteriores, se está en condiciones de describir la fase 4 (Resolución de conflictos), la cual ocurre cuando se ha producido una intrusión y han sido adulterados los logs almacenados en el sistema original. Tanto el intruso como la víctima pueden intentar negar la autenticidad de la evidencia. Un auditor deberá luego verificar que nadie haya modificado el registro desde la generación. Esta verificación de evidencia consiste en: corroborar que la firma de los logs es auténtica y verificar la prueba de integridad. Para la verificación de la firma, el auditor debe tener acceso a los correspondientes certificados de clave pública de las fuentes generadoras. La prueba de integridad está representada por el hash y la marca de tiempo inmodificable que está almacenada en la blockchain.

La figura 7.1, representa las fases para asegurar que se ha preservado la evidencia de un log.

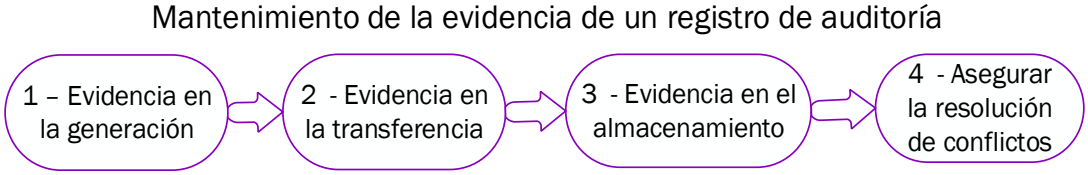

Figura 7.1: Fases en el mantenimiento de la evidencia de un log
Fuente: Gráfico adaptado de [20]

Se trata, entre otras cosas, de preservar la evidencia de cada uno de los registros de auditoría. Con el objetivo de cumplir con esa finalidad, se va a proponer almacenar los logs en una arquitectura basada en cadenas de bloques enlazadas y cifradas.

Para tener un control total de la solución, manipular la configuración de la cadena de bloques sin restricciones y bajar los costos de despliegue de la misma, se va a optar por una blockchain privada con permisos. Este tipo tecnología define entidades que son identificadas por certificados digitales y controlan los tipos de acceso en base a perfiles de usuarios registrados.

Dada la importancia de la escalabilidad y los tiempos de respuesta, se va elegir una solución de alta perfomance, baja latencia y adaptable a distintos volúmenes de datos.

7.1. ¿Por qué elegir la tecnología blockchain?

Blockchain cuenta en su diseño con tres pilares básicos que son claves para lograr el objetivo principal de esta tesis: *"asegurar la inmutabilidad de la información contenida en los registros de auditoría"*.

Se detallan a continuación cada uno de esos pilares:

- ✓ Descentralización

  La naturaleza descentralizada de la tecnología blockchain significa que no se basa en un punto central de control, de esta forma logra que el sistema sea más justo, seguro y tolerante a fallas. No hace falta un sistema intermediario. Los protocolos de consenso aseguran transacciones válidas y registran datos en una manera que es incorruptible.

  Protocolo de consenso
  El protocolo de consenso describe cómo es el proceso de comunicación entre los nodos para determinar cuáles son las transacciones ciertas, y cuál es su orden. El consenso se logra cuando hay suficientes dispositivos de acuerdo con la validez y el orden de un conjunto de transacciones.

- ✓ Transparencia

  La transparencia de una cadena de bloques proviene del hecho de que las tenencias y las transacciones de cada usuario son todas fehacientemente verificables.

- ✓ Inmutabilidad

  La inmutabilidad significa que una vez que se ha ingresado algo en la blockchain, no puede ser manipulado. Incluso el más mínimo cambio en la entrada tiene enormes consecuencias en el hash de salida.

Alterar cualquiera de los bloques ya vinculados en la cadena de bloques rompería el puntero hash al bloque anterior

7.2. Arquitectura de la solución

La arquitectura completa de la solución propuesta se muestra en la Figura 7.2.1. Los logs de auditoría se generan en cada una de las fuentes (base de datos, router, switches, firewall, IDS, sistemas operativos, aplicaciones de gestión administrativa, etc.) previa configuración de cada una de ellas. Cualquiera sea el originante del log, se debe aplicar al mismo políticas de seguridad estrictas: control de acceso físico o lógico, contraseñas de administración complejas, actualización de firmware o parches de seguridad. En caso de obviar estas recomendaciones, el originante puede ser atacado, deshabilitando las configuraciones de registración de eventos.

En caso que las fuentes soporten el agregado de firma digital, como lo describe el RFC 5848 (Signed Syslog Messages), los registros van a ser generados con un campo extra, el cual contiene el resultado de aplicarle a los datos de log, la función sign(), utilizando la clave privada y secreta del dispositivo generador del evento [21]. En el presente trabajo, no se usan fuentes generadoras de logs firmados digitalmente (en general lo hacen hardware de mayor complejidad y costo).

Los datos generados se transmiten a un sistema de almacenamiento central utilizando protocolos diseñados a tal fin. Un estándar para transmisión de logs muy usado por su confiabilidad y facilidad para ser configurado, es el protocolo Syslog estandarizado en la RFC 5424 (*"The Syslog Protocol"*), el cual utiliza como mecanismo de transporte a UDP (User Data Protocol). Éste favorece a mejorar la perfomance en la entrega de datos, aunque no es orientado a conexión (en caso de perderse en el camino uno o más segmentos con información, la detección de la pérdida debe implementarse como una función extra en el desarrollo del modelo). Una alternativa para lograr una transmisión confiable, es el RFC 3195 (*"Reliable Delivery for Syslog"*). En este caso el protocolo de transporte usado es TCP (Transmission Control Protocol), el cual como ya se explicó anteriormente, utiliza un mecanismo orientado a conexión (garantiza orden y entrega de todos los segmentos con datos).

Cualquiera fuera el protocolo usado para transportar los logs de eventos desde las fuentes a la base de datos central, es posible agregar mayor seguridad al proceso de transmisión. Para ello se debe encapsular el protocolo origen en otro protocolo seguro, por ejemplo, IPSec, PPTP, L2TP, etc., el cual cifra los datos que circulan por una autopista pública e insegura, permitiendo que se mantengan su integridad y privacidad. En el destino, corresponde ejecutar la operación inversa (quitar el protocolo seguro) con el objetivo de recuperar el paquete de datos original, y poder manipular la información del mismo.

Cuando los datos llegan al servidor de logs centralizado, los mismos son almacenados en una base de datos confiable y segura. Una aplicación monitorea la base de datos para detectar la llegada de nuevos logs. Por cada nueva entrada que se almacena en la base de datos central, se extrae la información relevante del registro que acaba de llegar:

- ✓ IP del sistema generador
- ✓ Marca de tiempo en que se generó el log
- ✓ Tipo de log
- ✓ Descripción del mismo.

Estos son los datos que se envían a la blockchain generando una transacción que es almacenada en un bloque junto con transacciones que llegan desde otros nodos del sistema distribuido. Cada transacción es aprobada por los nodos pares del sistema de la red de blockchain en un tiempo máximo, predefinido como tiempo de espera. En caso haber vencido el tiempo de espera o no haber sido aprobada la transacción, los datos no serán dados de alta en el bloque correspondiente y se rechazará la transacción. El tiempo de espera asegura que el retraso entre la generación de registros y su inclusión en la cadena de bloques sea lo más bajo posible.

En el momento que se llega a una cantidad máxima de transacciones permitidas para el bloque, el mismo es agregado como un nuevo eslabón en la cadena de blockchain, previo proceso de aprobación de los pares que forman parte de la red. Dicho proceso es denominado consenso. Para lograr el consenso, cada bloque en la cadena deberá ser aprobado por al menos un integrante de cada organización que forme parte de la solución.

Todas las organizaciones participantes también podrán agregar transacciones que van a contener datos de sus propias implementaciones de registración de logs de auditoría.

Una vez almacenada la transacción en la cadena de bloques, cualquier auditor puede verificar la evidencia en una fecha posterior a su ocurrencia. El valor real de la evidencia del dato permanece inmutable en el almacenamiento local del nodo de blockchain, el cual es una réplica de todos los nodos que forman parte de la red.

Con el objeto de verificar la integridad de los datos de evidencia, el auditor primero utilizará una aplicación con la cual podrá verificar la firma, con la clave pública del sistema generador del log. Este paso certifica el origen del log y asegura su no repudio. Luego calculará el hash de los datos almacenados en la base de datos centralizada y extraerá el hash de los datos almacenados en la blockchain. La aplicación va a comparar el valor hash de la evidencia propuesta con el valor hash almacenado en la blockchain para esa transacción. Si ambos valores son idénticos, se prueba que el log contiene información íntegra. En este punto habrá comprobado la autenticidad e integridad de la evidencia.

Es oportuno mencionar que, en el caso de la presente tesis, se guardan en la blockchain todos los datos referidos a un evento, entre ellos la dirección del originante, hora, criticidad y descripción del evento, etc. Este tipo de proceder tiene como objetivo recuperar toda la información de registros que pudieran haber sido eliminados en acciones fraudulentas.

Existe, como alternativa directa y menos compleja, la opción de sólo almacenar en la blockchain el identificador de la evidencia y el hash de los datos de la misma. En este caso, se mejorará la performance en los procesos de altas y consultas sobre la blockchain; pero la solución solo aplicará para el control de integridad de los registros que se encuentren en la aplicación de log central. De esta forma, se gana en tiempo de

respuesta, pero no se podrá recuperar información que correspondan a registros borrados del almacenamiento centralizado.

Una característica a destacar en la solución propuesta en el gráfico 7.2.1 es la posibilidad de que convivan varias organizaciones, de forma tal de disminuir los costos de mantenimiento de la blockchain. En este caso son 3 organizaciones que aportan logs de eventos a una blockchain que los almacena en 3 nodos distintos.

Se puede adicionar a la solución una característica que garantice la inmutabilidad de registros pertenecientes a los sistemas de información de varias empresas. Por ejemplo, un parque industrial, dónde convivan "n" número de industrias, con sus respectivos sistemas. Cada una de ellas almacena sus propios logs en la cadena de bloques. De hecho, también cada una puede aportar un nodo a la blockchain. Tendremos "n" organizaciones, salvando todos sus logs en una blockchain de "m" nodos. En estos casos cuando se necesite recuperar un log, será importante que pueda probarse fehacientemente que pertenece a la organización que muestra la información del registro.

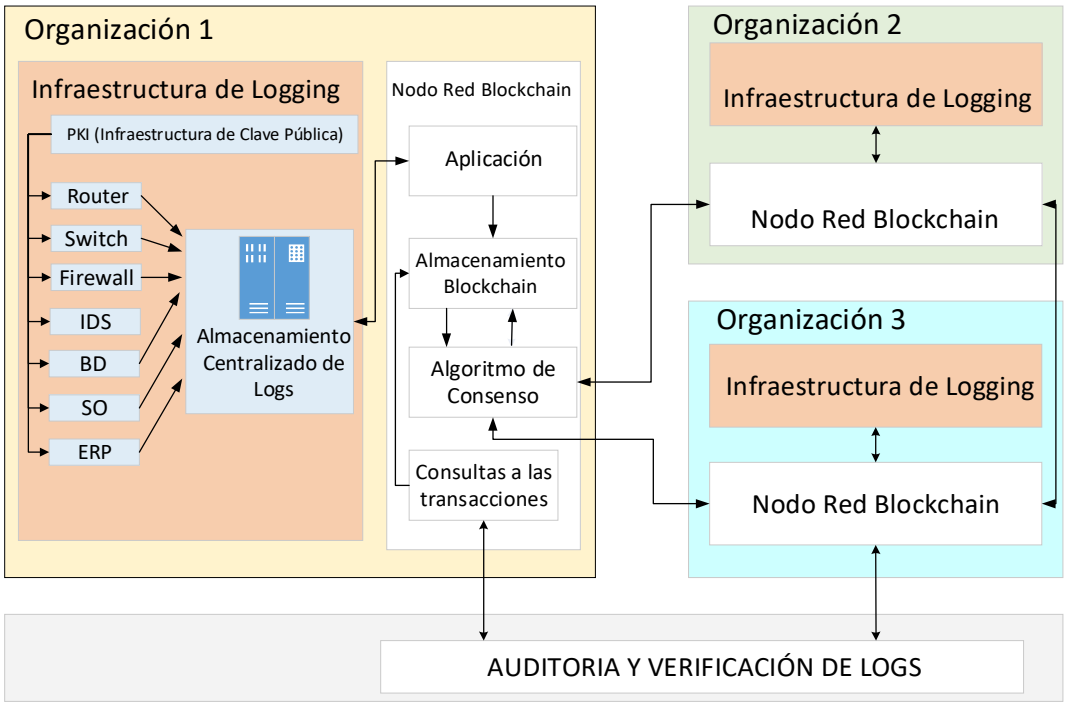

Figura 7.2.1: Arquitectura de la solución

7.3. Fases del proceso de registración y consultas de logs en la blockchain

Antes de detallar el proceso de registración de los registros de auditoría en el repositorio inmutable (blockchain), se van a definir distintas variables, con el objetivo de ayudar a la formalización de la propuesta. En ese contexto, cada una de las organizaciones participantes de la solución, tendrá un identificador, en este caso "o", con

"o" ϵ {1.n}; una clave privada para firma digital de cada una de ellas, $Z_{o,}$ y la clave pública correspondiente, K_o.

Se definirán dos procesos, P1 (Proceso Aplicación Cliente) y P2 (Proceso Aplicación Blockchain):

- ✓ Proceso Aplicación Cliente (P1): se establece un contador de log k, con k ϵ {1.l} y un contador de transacciones i, con i ϵ {1.t}. También se define a cada log que llega al servidor central como $L_{o,k}$ y al procesamiento del mismo para extraer la información importante que será almacenada en la blockchain, se lo denomina $P_{o,i}$. A los fines de la presente tesis, la "información importante" de un log $L_{o,k}$ está compuesta por :

  - ✓ Identificador de organización ("or")
  - ✓ Dirección ip del originante ("ip")
  - ✓ Timestamp[38] del registro ("ts")
  - ✓ Nivel de criticidad ("cr")
  - ✓ Prioridad ("pr")
  - ✓ Descripción del evento ("de")
  - ✓ Firma ("si"), aquí se almacena el resultado de aplicar la función sign() a los datos del log, utilizando la clave privada del sistema que lo origina [21][39]

  Al comienzo k=i, dado que cada log genera una transacción. La aplicación cliente se puede ejecutar en el sistema operativo como un *"proceso demonio"* o en un esquema de planificación cada 10 minutos (scheduling). En cualquier caso, cada vez que vez que detecta un registro nuevo en el almacén central, lo extrae como $L_{o,k}$, lo procesa convirtiéndolo en $P_{o,i}$ (información importante para almacenar) y lo incluye como parte de la nueva transacción en blockchain, $T_{o,i.}$ Haciendo uso de las propiedades de la implementación de la solución utilizando Hyperledger Fabric, la transmisión de $T_{o,i}$ al nodo de blockchain se hace utilizando un canal cifrado. La expresión formal completa de P1 es así:

  $T_{o,i} = (P_{o,i}, H_{o,i}, S_{o,i})$ dónde cada uno de los argumentos se define como:

  $P_{o,i} = (or, K_o, L_{o,k})$

  $H_{o,i} =$ hash $(L_{o,k})$, pero de $L_{o,k}$ solo [or, ip, ts, cr, pr, de]

  $S_{o,i} =$ sign $(Z_o, P_{o,i})$, usado para el no repudio de la organización

- ✓ Proceso Nodo Blockchain (P2): Después de que la transacción fue propagada a la red y propuesta como parte de un bloque por el líder del protocolo de

[38] Marca de tiempo, es la fecha y hora en la que se produjo el evento en el dispositivo o software originante del mismo.

[39] Si bien en las pruebas del modelo de la presente tesis no se usan fuentes que generan logs firmados digitalmente, el modelo conceptual se desarrolló para ser implementado con cualquier tipo registros.

consenso, el procesamiento real se lleva a cabo en cada nodo Se incluye una marca de tiempo basada en el consenso distribuido con cada transacción y la unicidad de la entrada del registro es verificado por cada nodo de la siguiente forma:

$$i > 1 : H_{o,i} \ni \{H_{o,1} \cdot H_{o,(i-1)}\}, \ \forall o \in 1..n$$

Si esta condición falla, no se producen cambios en el estado persistente como resultado de la transacción. Dos entradas de registro idénticas con el mismo valor hash no puede ser parte del sistema. Si la condición pasa y más de dos tercios de todos los nodos han procesado con éxito la transacción (proceso de consenso entre nodos compañeros), la misma se almacena irreversiblemente en el libro mayor.

El gráfico 7.4.1 muestra el diagrama de la solución propuesta, teniendo en cuenta a cada una de las entidades y la secuencia de los procesos, en este caso el P1 y el P2:

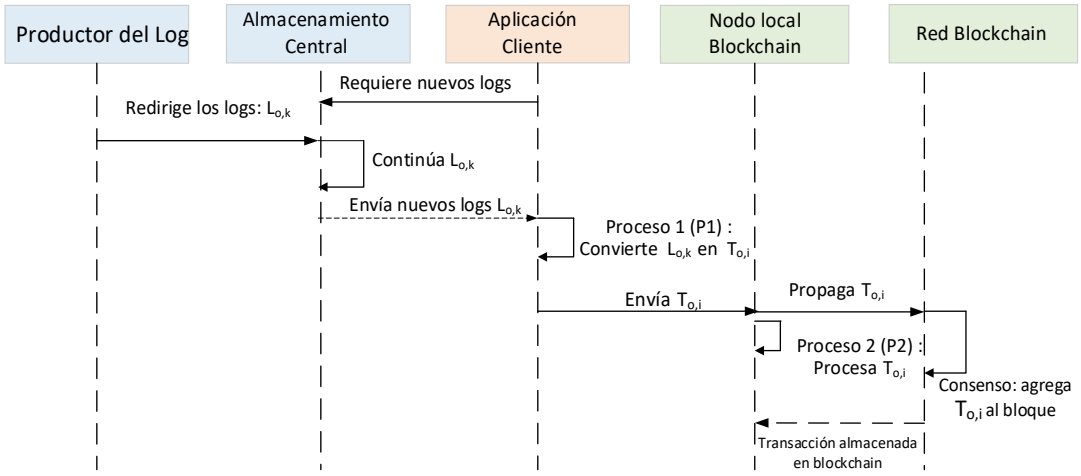

Figura 7.3.1: Diagrama de los procesos P1 y P2 involucrados en la solución

Queda por formalizar los procesos de verificación de la integridad de un log y por último el proceso de listado y recuperación de uno o más logs borrados del almacén central.

Para ello se definirán el Proceso Verificación (P3) y el Proceso Recuperación (P4):

- ✓ Proceso de Verificación (P3): para la verificación, desde la aplicación consultas se debe ingresar tres valores: el ID de organización (o), la dirección ip del originante del registro (ip) y el timestamp (ts) de la entrada de log que se quiere verificar. Ésta accederá al almacén central, buscará el log correspondiente ($L_{o,ip,ts}$) y calculará el hash de [or,ip,ts, cr, us, de]. A continuación, solicitará una prueba de inmutabilidad a la red blockchain, consultando a la misma con el valor hash calculado (h1). Si en la blockchain se encuentra una transacción cuyo

valor H = hash (or, ip, ts, cr, pr, de) es igual a valor hash calculado (h1), habrá comprobando la integridad del registro almacenado en el repositorio central. Respecto al no repudio de la organización, si la blockchain no implementa en forma nativa el control de no repudio; el proceso P3 deberá corroborar la firma de la misma, utilizando la función verify(). Ésta última función es la verificación correspondiente a sign(), por lo tanto, si la firma de la organización es correcta devuelve un 1, caso contrario un 0. La expresión formal de P3 es:

$$H_o = \text{hash}\ (L_{o,ip,ts}) \ \wedge \ \text{verify}\ (K_o, P_{o,ip,ts}, S_{o,i}\ ),\ K_{or} \in \{K_1 ....K_n\}$$

En caso que el valor hash calculado h1 no fuera encontrado en las transacciones almacenadas en la blockchain, se concluirá que, por algún motivo, la transacción no fue agregada a la cadena de bloques o los datos usados para calcular h1 fueron adulterados en el almacenamiento centralizado. Un perito informático o un auditor deberán determinar la causa.

- ✓ Proceso de Listado y Recuperación (P4): para el proceso de listado y recuperación de logs, un usuario deberá ingresar a la aplicación de consultas: la organización (o), una hora de inicio (hi) y una hora de final (hf). Estas variables serán utilizadas para hacer dos consultas: la primera, a la base de datos de repositorio central (entradas L_o entre [hi,hf]) generando un listado L_{rc} (listado transacciones registradas en el repositorio central); y la segunda, va a consultar al nodo de la blockchain (entradas T_o entre [hi,hf]) generando un listado L_b (listado de transacciones agregadas a la blockchain). Ambos listados podrán ser comparados, a partir de lo cual, se podrán presentar cuatro resultados:
  - ✓ Las ocurrencias se detectan en L_{rc} y en L_b y H_o (hash de los datos del registro almacenado en la blockchain) = hash(L_o) (hash de los datos almacenados en el repositorio central). Este es el resultado esperado. Los datos en el almacenamiento central no fueron alterados. Es el escenario ideal.
  - ✓ Las ocurrencias se detectan en L_{rc} , pero no forman parte de L_b. Esto implica que existen entradas de logs en el repositorio central que no fueron agregadas a la blockchain.
  - ✓ Las ocurrencias se detectan en L_b y pero no se listan en L_{rc}. Esto implica que hay registros de logs que permanecen inmutables en la blockchain, pero que fueron eliminados del repositorio central.
  - ✓ Las ocurrencias se detectan en L_{rc} y en L_b y H_o (hash de los datos del registro almacenado en la blockchain) ≠ hash(L_o) (hash de los datos almacenados en el repositorio central). En este caso corresponden a entradas de logs que se mantienen inmutables en la blockchain, pero fueron adulteradas en el repositorio central.

  En todos los casos (excepto el primero), corresponderá al auditor tomar una decisión sobre qué hacer con las diferencias. La expresión formal para P4 es:

Si ∀ hash (L_o) ∄ H_o para el período hi y hf ∴ log no fue procesado por la blockchain

Si ∀ H_o ∄ hash (L_o) para el período hi y hf ∴ log en almacén central fue eliminado para la marca de tiempo dada.

Si $H_o \neq$ hash (L_o) ∧ [or, ip y marca de tiempo] de L_o = [or, ip y marca de tiempo] de T_o, para el período hi y hf ∴ log en almacén central adulterado

El gráfico 7.3.2 muestra el diagrama de la solución propuesta, teniendo en cuenta a cada una de las entidades y la secuencia de los procesos, en este caso el P3 y el P4.

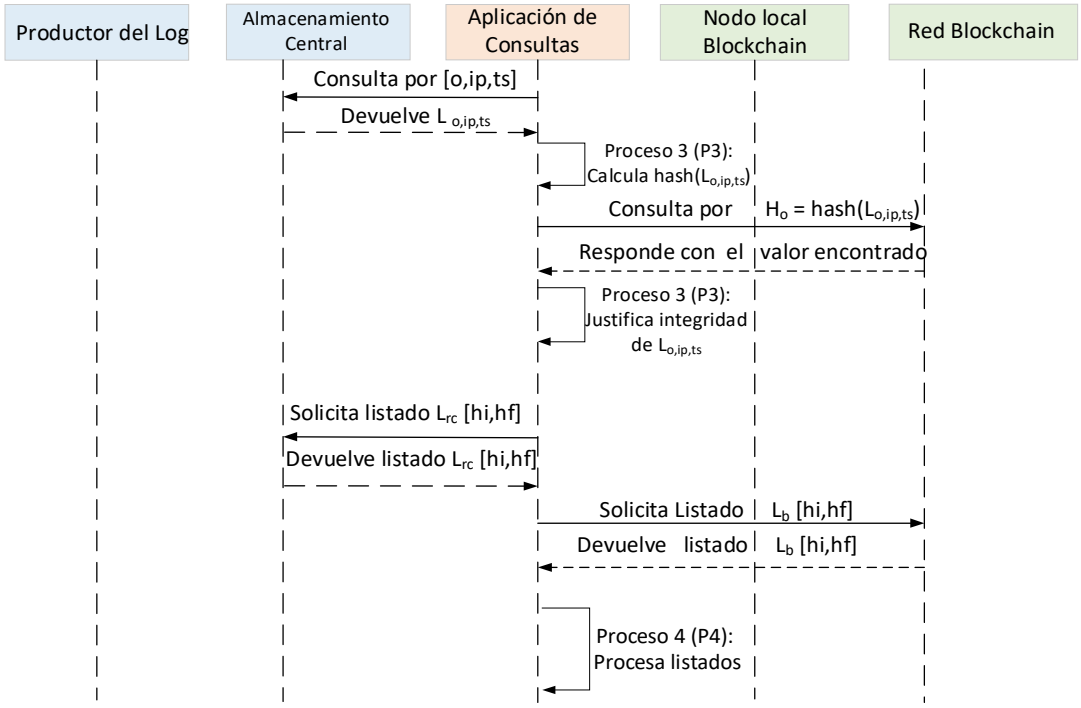

Figura 7.3.2: Diagrama de los procesos P3 y P4

La figura 7.3.3 describe la solución desde una perspectiva integral. En (1), un router configurado para generar y enviar logs, genera un evento y lo envía a un almacén centralizado de logs. En (2), el evento es recibido por el servidor central de logs, luego procesado y por último enviado a una base de datos configurada a tal fin. En (3), una aplicación que posee dos interfaces (una para la base de datos y otra para la blockchain), se conecta con la base de datos, extrae el log de un evento, le asigna un formato y por último invoca una función que ejecuta la interface con blockchain, para proponerle una inserción a la cadena de bloques. En (4), la cadena de bloques (recibe propuestas de inserción de distintas aplicaciones) ejecuta un algoritmo de consenso para garantizar la integridad de la estructura de datos. En (5), luego de aprobado el consenso, el log de evento

propuesto en la instancia anterior es agregado como parte de un bloque inmutable a la cadena.

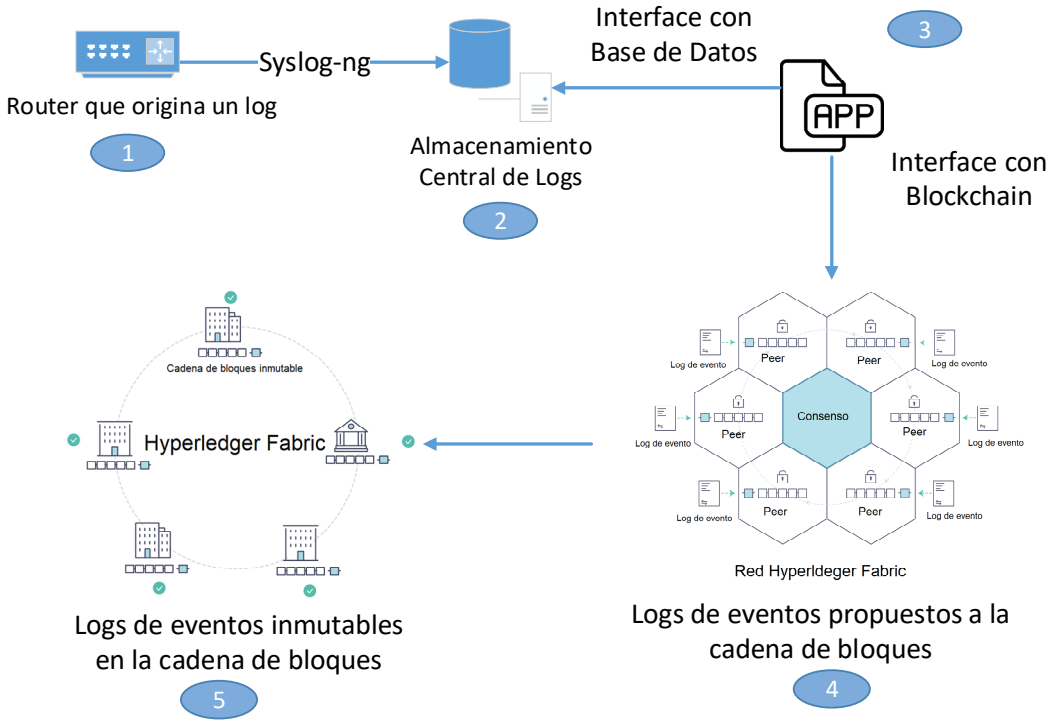

Figura 7.3.3: Instancias de la solución propuesta

7.4. Conclusiones del capítulo

En el presente capítulo se ha formalizado una propuesta de solución, se describió la arquitectura de la misma, y se especificaron los procesos correspondientes.

La formalización de la propuesta permite demostrar en el plano teórico que existe una solución, al menos en la dimensión mencionada. El próximo paso es hacer. viable y factible la implementación de la solución. Para lograrlo, se hará uso de herramientas y tecnologías que basan su funcionamiento en una arquitectura distribuida junto a un modelo de datos agrupados en bloques que son enlazados por “punteros” criptográficos.

Es tiempo de elegir entre las distintas versiones disponibles una blockchain que se ajuste a las necesidades del problema. El próximo capítulo trata acerca de la tecnología elegida: Hyperledger Fabric.

8. Capítulo 8. Hyperledger Fabric, una blockchain privada y eficiente

En el año 2015, la Fundación Linux fundó el proyecto Hyperledger con el objetivo de desarrollar las tecnologías blockchain en el campo de la industria.

Hyperledger Fabric es uno de los proyectos de blockchain dentro de Hyperledger y se define como una "tecnología de libro de cuentas distribuido", DLT (Distributed Ledger Technologies). Al igual que otras tecnologías de blockchain, tiene un libro mayor, utiliza contratos inteligentes y es un sistema mediante el cual los participantes gestionan sus transacciones.

Hyperledger Fabric es una cadena de blockchain privada y sus miembros deben ser conocidos, a diferencia de las implementaciones pública, como Bitcoin, dónde se permite que identidades desconocidas participen de la red, y mediante algoritmos de prueba de trabajo, puedan sellar bloques.

En el caso de Fabric, los miembros de la red deben inscribirse en un MSP (Membership Service Provider – Proveedor de Servicios de Membresía). También prevé la posibilidad de definir canales, de forma tal que un grupo de usuarios pueda crear un libro mayor por separado por cada canal.

En primer lugar, cada nodo participante (peer – compañero) tiene una copia del libro mayor de la red a la que pertenece. Ahora bien, el libro mayor, tiene dos componentes: el *"estado global"* y *"bloques encadenados"* (registro de todas las transacciones). El primero describe el estado del libro mayor en un momento dado y lo implementa en una base de datos (Fabric implementa LevelDB y CouchDB). El segundo componente representa el registro de todas las transacciones, las cuales verifican y otorgan trazabilidad a los valores almacenados en el *"estado global"*.

En segundo lugar, Hyperledger Fabric utiliza contratos inteligentes, que son agrupados en cadenas de código conocidas como *"chaincode"*. Estos son invocados desde aplicaciones externas a la cadena de bloque, para interactuar con el libro mayor.

Por último, las transacciones deben escribirse en el libro mayor en el orden en que ocurren. Para que esto suceda, se debe establecer el orden de las transacciones y el método para rechazar las que son erróneas. Para ello se utilizan distintos mecanismos de consenso y en diferentes niveles.

El modelo de Hyperledger Fabric que permite implementar una solución integral de una blockchain privada y con permisos (las transacciones son aprobadas por un grupo conocido de nodos validados por los dueños de la blockchain) está compuesto por:

- ✓ Activos: se utilizan para el intercambio de casi cualquier cosa con valor monetario a través de la red (desde alimentos hasta divisas)
- ✓ Chaincode: permite ejecutar la "inteligencia" del negocio a través de las transacciones

- ✓ Libro mayor: el libro mayor inmutable y compartido almacena el historial de transacciones para cada canal, e incluye capacidad de consulta tipo SQL (Structured Query Language, Lenguaje de Consulta Estructurado)
- ✓ Privacidad: los canales permiten transacciones privadas y confidenciales, entre empresas y organizaciones que intercambian activos en una red común.
- ✓ Servicios de seguridad y membresía: la membresía autorizada proporciona una red blockchain confiable, donde los participantes saben que todas las transacciones pueden ser detectadas y rastreadas por reguladores y auditores autorizados.
- ✓ Consenso: un enfoque de consenso dinámico permite la flexibilidad y escalabilidad necesarias para la organización.
- ✓ Servicios de Ordenamiento: se encarga de agrupar las transacciones aprobadas y asignarle un orden. Es un componente fundamental en el ordenamiento secuencial en el tiempo de las transacciones.

8.1. ¿Por qué Hyperledger Fabric?

El modelo propuesto por Fabric tiene algunas características que lo distinguen de las demás soluciones, más allá de ser una plataforma más que atractiva para organizaciones que no pueden justificar, ni asumir, el costo de implementar, mantener u operar en una blockchain pública. Los rasgos que caracterizan a la solución son:

- ✓ Código abierto: Hyperledger Fabric es una plataforma de código abierto, por lo que todos pueden usarla libremente.
- ✓ Administración de identidades: cuenta con un mecanismo que permite identificar a todos los participantes de la red
- ✓ Privacidad y confidencialidad: todos los datos, incluidas las transacciones, miembros e información del canal son invisibles e inaccesibles para quienes no tengan permisos
- ✓ Procesamiento Eficiente: maneja el concepto de roles dentro de la red. Para lograr concurrencia y paralelismo, las transacciones de ejecución son separadas de las transacciones de órdenes de ejecución y acuerdos
- ✓ Funcionalidad de los contratos inteligentes: los contratos inteligentes codifican la lógica que se invoca mediante tipos específicos de transacciones.
- ✓ Diseño modular: cuenta con una estructura modular. Esta permite por ejemplo elegir un algoritmo para el cifrado, otro para la identidad y otro para el consenso. Para ello lo único que se debe hacer es configurar interface de blockchain de Hyperledger Fabric.

Por otra parte, el resultado de diferentes pruebas de rendimiento y perfomance, ha posicionado a Fabric como un producto con informes muy alentadores. Entre los distintos trabajos afines, se puede citar a una publicación de la IEEE [18], la cual comparó las prestaciones de tres soluciones de blockchain utilizando dos plataformas de *"benchmarking"*[40]:

- ✓ Yahoo Cloud Serving Benchmark (YCSB) es un conjunto de programas y especificaciones de código abierto que a menudo se usa para comparar el rendimiento relativo de los sistemas de gestión de bases de datos NoSQL.
- ✓ Smallbank: Smallbank es un punto de referencia para la carga de trabajo OLTP (On-Line Transaction Processing) por ejemplo, un contrato inteligente que transfiere dinero de una cuenta a otra.

En la figura 8.1.1, podemos observar a la izquierda a Hyperledger con la mayor cantidad de transacciones por segundo: 1273 tps para YCSB y 1122 tps para Smallbank) y la segunda menor latencia: 38 ms[41] para YCSB y 51 ms para Smallbank.

En el mismo gráfico 8.1.1 a la derecha, se puede concluir que Hyperledger tiene la mejor cantidad de transacciones por segundo con una estructura de hasta 20 nodos y la segunda menor latencia, también con un crecimiento hasta 20 nodos.

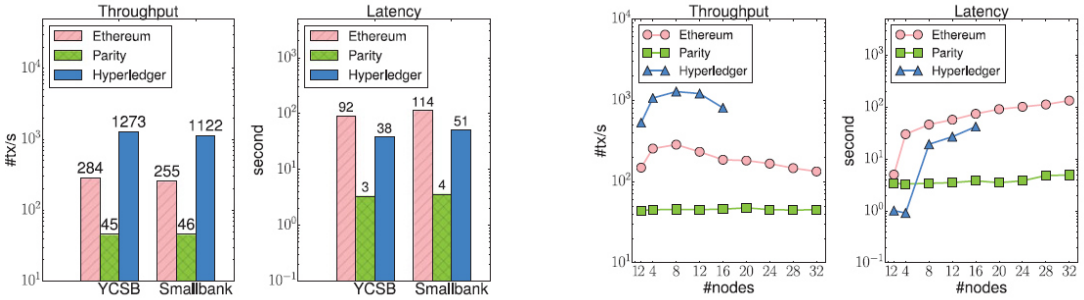

Figura 8.1.1: Comparación de perfomance y escalabilidad entre blockchains
Fuente: [18]

Otras mediciones realizadas sobre Hyperledger Fabric, en dónde se optimizaron el tamaño del bloque, se implementó paralelismo en las propuestas de transacciones y sólo se publicó el ID de la transacción, lograron resultados altamente satisfactorios.

En la figura 8.1.2 a la izquierda se puede observar la configuración por defecto de Fabric (Fabric 1.2) y tres configuraciones optimizadas (Opt P-I, Opt P-II y Opt P-III). La última de ellas (Opt P-III) logra la menor latencia, 12,36 ms. En la parte derecha de la misma figura, la misma configuración optimizada (Opt P-III) permite atender exitosamente más de 20.000 transacciones por segundo. Una perfomance envidiable para

[40] Término asociado a la medición de prestaciones con el objetivo de comparar distintas plataformas.
[41] ms (milisegundo), unidad de tiempo que representa una milésima de segundo

una plataforma blockchain privada, la cual convierte a Hyperledger Fabric como una opción merecidamente a ser considerada.

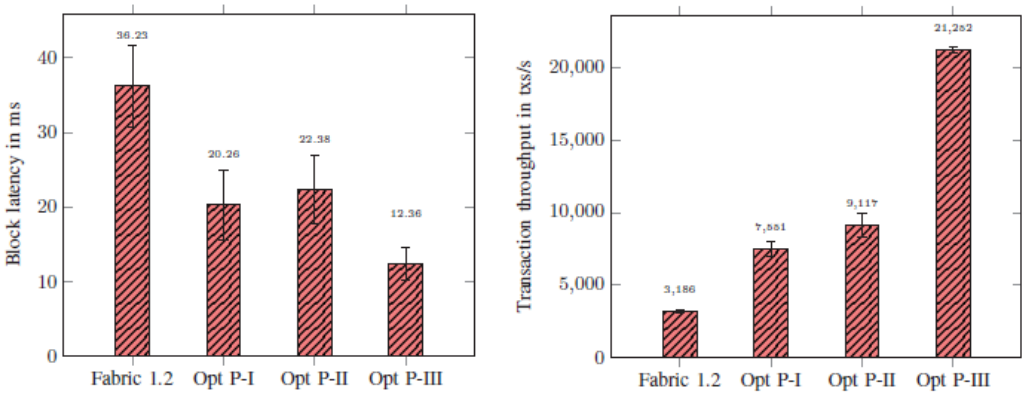

Figura 8.1.2: Configuraciones optimizadas de Hyperledger Fabric
Fuente: [22]

La suma de lo descripto a lo largo del presente análisis sirve como un importante sustento técnico para seleccionar a Hyperledger Fabric entre las decenas de opciones que ofrece la tecnología de blockchain.

8.2. Componentes una red Hyperledger Fabric

La figura 8.2.1 resume en un solo gráfico la totalidad de componentes que forman parte de una red Hyperledger Fabric. La descripción de cada uno de ellos se desarrolla a continuación:

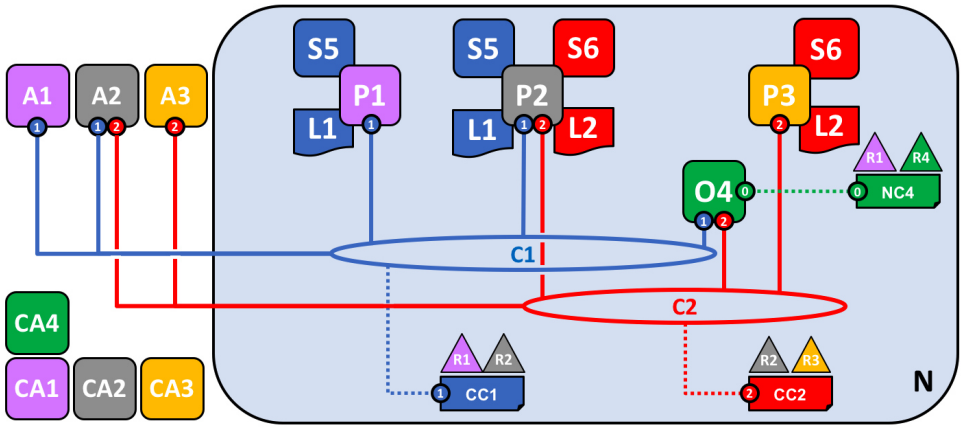

Figura 8.2.1: Una red Hyperledger Fabric (N) y sus componentes
Fuente**:** https://www.hyperledger.org/projects/fabric

8.2.1. Organization (Organización):

La organización representa a una empresa, universidad, organismo del estado, etc. Es el miembro principal de la cadena de bloques. Toda organización se une a una cadena Hyperledger Fabric agregando un proveedor de servicios de Membresía (MSP).

El MSP define la pertenencia de los miembros de la red, a una organización; y lo hace verificando que las firmas fueron generadas por una identidad válida, con un certificado válido, emitido por una autoridad de certificación aprobada por esa organización. En la figura 8.2.1 hay definidas 4 organizaciones: R1, R2, R3 y R4.

Para el modelo propuesto, vamos a identificar la Organización con el Id "1", aunque la solución prevé la convivencia de "n" organizaciones en la misma blockchain (Figura 7.2.1).

La implementación de la solución contempla la creación de la *Organización1* y la *Organización2*.

8.2.2. Certificate Authority (Autoridad de Certificados - CA):

Una autoridad de certificados emite y respalda certificados digitales. Fabric tiene incorporada una autoridad de certificados por defecto, aunque se pueden usar autoridades de certificados de terceras partes. En la figura 8.2.1 está representado por CA4.

CA4 juega un papel clave puesto que publica los certificados X.509 que identifican los componentes pertenecientes a la organización R4 (nodos de ordenamiento, nodos compañeros, organización, aplicaciones, canales, usuarios, etc.).

En el ejemplo de la figura 8.2.1, se pueden observar 4 organizaciones distintas (R1, R2, R3 y R4). En estos escenarios es recomendable utilizar distintas autoridades de certificados; en este caso están identificadas como CA1, CA2, CA3 y C4.

La autoridad de certificados es el componente principal del proveedor de servicios de membresía (Member Service Provider - MSP). Este es un componente abstracto del sistema que proporciona credenciales a los clientes y sus pares para que participen en una red Hyperledger Fabric. Los clientes usan estas credenciales para autenticar sus transacciones, y los pares usan estas credenciales para autenticar los resultados del procesamiento de transacciones (endosos).

En la solución propuesta se prevé la creación de dos autoridades de certificación, una autoridad para cada organización. En este caso se llamarán *ca-org1* y *ca-org2*.

8.2.3. Member Service Provider (Proveedor de Servicios de Membresía - MSP)

A pesar de su nombre, el proveedor de servicios de membresía en realidad no proporciona nada. Más bien, la implementación del requisito de MSP es un conjunto de carpetas que se agregan a la configuración de la red.

Se utiliza para definir una organización tanto internamente (las organizaciones deciden quiénes son sus administradores) como externamente (al permitir que otras organizaciones validen que esas entidades tienen la autoridad para hacer lo que intentan hacer).

Mientras que las autoridades de certificación generan los certificados que representan identidades, el MSP contiene una lista de identidades autorizadas.

El MSP identifica qué CA raíz y CA intermedio se aceptan para definir los miembros de un dominio de confianza al enumerar las identidades de sus miembros o al identificar qué CA están autorizadas a emitir identidades válidas para sus miembros.

Pero el poder de un MSP va más allá de simplemente enumerar quién es un participante de la red o miembro de un canal. Es el MSP que convierte una identidad en un rol mediante la identificación de privilegios específicos que un actor tiene en un nodo o canal. Se debe tener en cuenta que cuando un usuario está registrado en una Fabric CA, un rol de administrador, par, cliente, orden o miembro debe estar asociado con el usuario. Por ejemplo, las identidades registradas con el rol de *"peer"* deberían, naturalmente, ser entregadas a un *"peer"*. Del mismo modo, las identidades registradas con el rol de *"administrador"* se deben dar a usuario con destreza técnica que cumplirá el rol de administrador de la organización.

La figura 8.2.3.1 correspondiente a la descripción de un MSP permite observar claramente la distribución de las carpetas en cada nodo con el certificado digital de cada uno de los roles en la red Fabric.

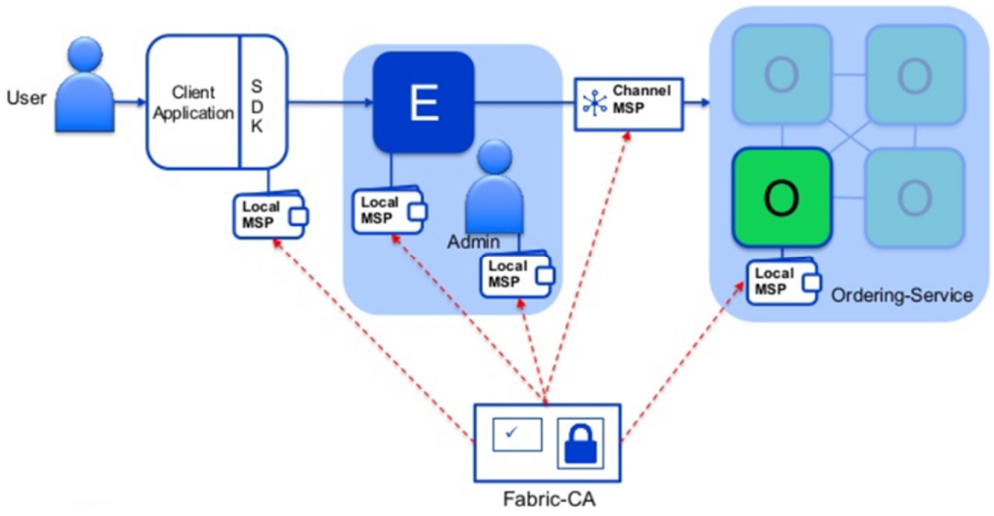

Figura 8.2.3.1: Estructura de un Proveedor de Servicios de Membresía

8.2.4. Asset (Activo):

Los activos constituyen objetos con información. Se representan a través de un par clave-valor. En la solución propuesta para la presente tesis, el activo es el registro de auditoría que se va a almacenar. A continuación, se muestra un ejemplo del mismo, generado por un router *Mikrotik* y capturado por el protocolo syslog-ng:

Datetime	**IP**	**Level**	**Facility**
2020-04-23 18:45:03	192.168.88.1	Notice	User

Prioridad	**Mensaje**	**Tag**	**Program**	**Firma**
Notice	Login failure for user Admin from 192.168.88.253	0d	System, Error, Critical	NULL

8.2.5. Identity (Identidad):

Como se observa en la figura 8.2.1, la red Fabric está compuesta por diferentes objetos: pares, ordenadores, aplicaciones cliente, administradores y más. Cada uno de estos objetos, elementos activos dentro o fuera de una red capaz de consumir servicios, tiene una identidad digital encapsulada en un certificado digital X.509. Estas identidades realmente importan porque determinan los permisos exactos sobre los recursos y el acceso a la información que los mismos tienen en una red blockchain. Además, el hecho de contar cada uno de ellos con un certificado digital, les permite encriptar y autenticar datos y transacciones.

8.2.6. Ordering Service (servicio de ordenamiento o de pedidos):

Se despliega sobre uno o más nodos conocidos como "nodo de pedido". El servicio de pedidos es el componente que reúne las transacciones aprobadas de las aplicaciones y las ordena en bloques de transacciones, que posteriormente se distribuyen a todos los nodos pares del canal. En la figura 8.2.1 el nodo de pedidos es O4.

Claramente, los nodos de pedidos tienen la función de ordenar las transacciones.

A diferencia de otras cadenas de bloques como Ethereum o Bitcoin en las cuales cualquier nodo puede participar del proceso de consenso (algoritmos de consenso probabilísticos), este servicio de ordenamiento administra un algoritmo de consenso determinista, por lo cual asegura que cualquier bloque validado por él, será final y correcto.

Esta estrategia permite evitar la bifurcación del libro mayor (fenómeno presente en blockchain distribuidas sin permiso), como así también garantiza mayor rendimiento y escalabilidad, evitando los cuellos de botellas que suelen presentarse cuando el pedido y la ejecución son realizadas en el mismo nodo.

El servicio de pedidos maneja las transacciones y las actualizaciones de acuerdo al proceso de consenso que se elija: Raft, Kafka o Solo. La última versión de Hyperledger Fabric (v2.0, liberada en el 29/01/2020), he eliminado a Kafka. Raft es un algoritmo de consenso tolerante a fallas (Crash Fault Tolerance – CFT)[42].

En la implementación de la solución se genera un servicio de pedidos que se llama *orderer.example.com*.

El proceso Raft utiliza un modelo de "líder y seguidor", en el que un líder se elige dinámicamente entre los nodos de pedido de un canal (esta colección de nodos se conoce como el "conjunto de consentimiento"), y ese líder replica los mensajes a los nodos seguidores. Debido a que el sistema puede soportar la pérdida de nodos, incluidos los nodos líderes, siempre que haya una mayoría de nodos de pedido (lo que se conoce como "quórum"), se aprueba la creación del bloque.

Se dice que Raft es "tolerante a fallas" (CFT). En otras palabras, si hay tres nodos en un canal, puede soportar la pérdida de un nodo (dejando dos restantes). Si tiene cinco nodos en un canal, puede perder dos nodos (dejando tres nodos restantes).

En la figura 8.2.6.1 se muestra el diagrama de estado del algoritmo Raft, a partir del modelo: seguidor, candidato o líder; el cual funciona de la siguiente forma: en el tiempo t0 todos los nodos están en estado de seguidores esperando la comunicación de un líder. Cada nodo tiene un tiempo de espera aleatorio, al cabo del cual, si no recibe comunicación, se postula como candidato. En el tiempo t1, el nodo “n” se postula como candidato. Necesita la mitad más uno de los votos. Todos los nodos seguidores están en

[42] CTF puede soportar caídas o bloqueos de nodos. Si la red tiene N nodos, necesita N/2 + 1 nodos vivos para funcionar. Una mejora a CTF es BTF (descripto en 6.5), dado que soporta la presencia de nodos maliciosos.

condiciones de votar. En el tiempo t2 el nodo "n" es elegido líder por un nuevo término (también conocido como período). El nuevo líder debe enviar cada cierto tiempo un mensaje, ya sea proponiendo una actualización o no, para que sus seguidores no agoten su tiempo de espera.

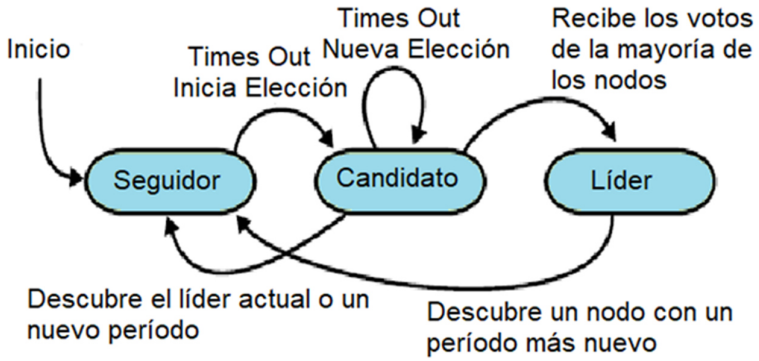

Figura 8.2.6.1: Diagrama de estados del proceso de consenso RAFT
Fuente: Gráfico traducido de [23]

8.2.7. Peer (Nodo Par):

Los nodos pares[43] o compañeros, son los componentes de la red donde se alojan las copias del libro mayor de blockchain, como así también, allí se almacenan los smart contract o contratos inteligentes.

A lo largo del presente trabajo, cuando se desarrolló el concepto de blockchain al nodo par se lo llamó sólo nodo. Observando la figura 8.2.1 se aprecia que el propósito del nodo "par 1" (P1) es simplemente alojar una copia del libro mayor (L1), para que otros accedan. De hecho, L1 está alojado físicamente en P1, y se puede actualizar o consultar usando el canal 1 (C1).

Los nodos pares tienen la función de validar y ejecutar las transacciones.

Hyperledger Fabric tiene definido 4 tipos de pares, de acuerdo a la función que cumplen (pueden tener más de una función):

- ✓ Par Aprobador (Endorsing peer): es un tipo especial de par validador que tiene un rol adicional para respaldar una transacción. Estos respaldan la solicitud de transacción que proviene del cliente. Cada par aprobador posee la copia del contrato inteligente instalado y una copia del libro mayor. La función principal del *"endorser"* es simular la transacción. Ejecuta el programa que está en el contrato inteligente sobre la copia personal del libro mayor, y genera los conjuntos de lectura

[43] El término *"Par"*, en el ambiente de blockchain, tiene el significado compañero o igual, y se refiere a que estos almacenan una copia de la cadena de datos encriptada.

/ escritura que se envían al cliente. Aunque durante la simulación, la transacción no se compromete con el libro mayor.

- ✓ Par Validador (Commited Peer): par que adiciona en su propia copia de la cadena de bloques, el bloque que recibe del servicio de pedidos. Este bloque contiene la lista de transacciones que cada uno de los pares que lo recibieron se comprometen a validar, marcando cada transacción como válida o no válida. Todas las transacciones (válidas o no) se guardan junto con el bloque.
- ✓ Par Ancla (Anchor Peer): como la red Fabric puede extenderse a través de múltiples organizaciones, necesitamos algunos pares para tener comunicación a hacia otras organizaciones. Son pares especiales que solo están autorizados para hacer este tipo intercambios. Los pares de anclaje se definen en la configuración del canal.
- ✓ Par líder (Leader Peer): Los compañeros líderes son aquellos que comunican o difunden mensajes del servicio de pedidos a otros compañeros de la misma organización. Tienen una comunicación directa con el servicio de pedidos. Estos pares usan el protocolo Gossip para asegurarse de que el bloque que recibió del servicio de pedidos les llegue a todos los demás pares. Si algún par principal no responde o está fuera de la red, entonces se selecciona un nuevo par principal del conjunto de pares disponible en función de un proceso de votación. Existe la posibilidad de elegir en forma estática (mediante comando) a un par líder.

Todo *"peer"* (par) tiene por defecto el rol de "par validador"

En la solución propuesta se definen cuatro *"peers"*, dos para cada una de las organizaciones participantes: *peer0.org1.example.com* y *peer1.org1.example.com* pertenecen a la O*rganización1*, así como *peer0.org2.example.com* y *peer1.org2.example.com* pertenecen a la *Organización2*. Tanto el *peer0.org1.example.com* como el *peer0.org2.example.com* son pares ancla y también pares aprobadores. Todos son pares validadores.

8.2.8. Channel (Canal):

Un canal es una entidad que conecta a distintos componentes de la blockchain privada permitiendo el aislamiento y la confidencialidad de los datos es decir comunicaciones privadas. Esto permite aprovechar la infraestructura de red y compartirla con más de una organización.

Un canal se define por miembros (organizaciones), nodos pares, el libro mayor compartido, las aplicaciones de chaincode y los nodos de servicio de pedidos.

Un libro mayor específico del canal se comparte entre los pares en el canal, y las partes que realizan transacciones deben autenticarse en un canal para interactuar con él. En el gráfico 8.2.1 se pueden distinguir 2 canales (C1 y C2), cada uno de ellos conectándose con distintos componentes.

8.2.9. Ledger (Libro Mayor)

En Hyperledger Fabric, un libro de mayor consta de dos partes distintas, aunque relacionadas, un estado mundial y una cadena de bloques. Cada uno de estos representa un conjunto de hechos sobre un conjunto de objetos

En primer lugar, hay un estado mundial: una base de datos que contiene los valores actuales de un conjunto de estados contables. El estado mundial facilita que un programa acceda directamente al valor actual de un estado en lugar de tener que calcularlo recorriendo todo el registro de transacciones. Los estados contables se expresan, por defecto, como pares clave-valor. El estado mundial puede cambiar con frecuencia, ya que los estados se pueden crear, actualizar y eliminar.

El estado mundial se implementa como una base de datos. Esto tiene mucho sentido porque una base de datos proporciona un amplio conjunto de operadores para un eficiente almacenamiento y recuperación de datos. Hyperledger Fabric implementa por defecto una base de datos llamada LevelDB y está integrada en el mismo proceso del sistema operativo. CouchDB es una opción de implementación de base de datos en Fabric. Es apropiada cuando el libro mayor recibe transacciones comerciales que manejan muchos datos. En esto casos la base de datos admite consultas enriquecidas y actualizaciones de datos más complejas. En la solución se implementan cuatro instancias de base de datos CouchDB, una para cada peer de la solución

En segundo lugar, hay una cadena de bloques: un registro de transacciones que almacena todos los cambios que han dado como resultado el estado actual de los activos. Las transacciones se recopilan dentro de los bloques que se agregan a la cadena de bloques, lo que le permite comprender el historial de cambios que han resultado en el estado mundial actual. La estructura de datos de blockchain es muy diferente al estado mundial porque una vez escrita, no se puede modificar, es inmutable.

La figura 8.2.9.1 describe un libro mayor L compuesto por la cadena de bloques B y el estado mundial W, donde la cadena de bloques B determina el estado mundial W. También podemos decir que el estado mundial W se deriva de la cadena de bloques B. Debe quedar claro que cada *"peer"* tiene una copia de la cadena de bloques y del estado mundial.

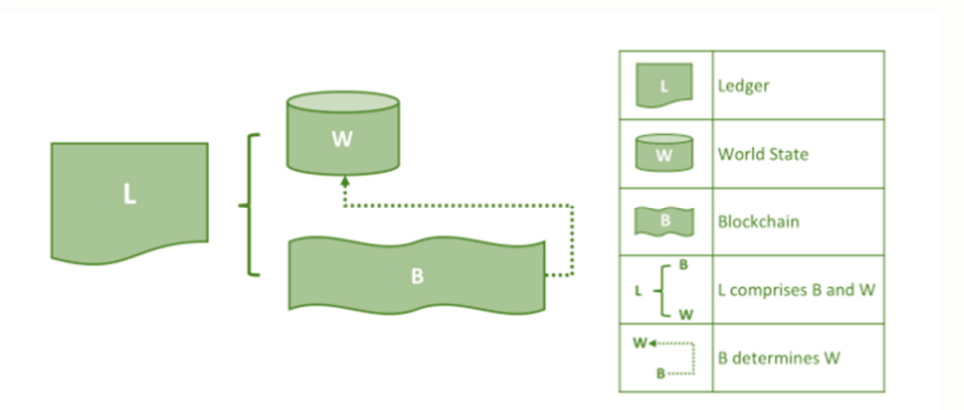

Figura 8.2.9.1: Libro mayor Hyperledger Fabric

8.2.10. Applications, Chaincode y Smart Contract (Aplicaciones, Cadenas de código y Contrato Inteligente):

Las aplicaciones son la interface entre el usuario y la red de blockchain (en la figura 8.2.1 podemos distinguir A1, A2 y A3). Como todo objeto en Hyperledger Fabric, una aplicación tiene una identidad asociada a una determinada organización.

El usuario en primer lugar accede a la aplicación que representa el Front End de la solución. Dentro de la aplicación se invoca a un Chaincode (Cadena de códigos) que puede estar codificada usando Go, Java o JavaScript[44]. Un chaincode puede contener varios smart contract (contratos inteligentes) quienes acceden verdaderamente al libro mayor invocando API's conocidas como Fabric-SDK[45].

En general, un contrato inteligente define la lógica de las transacciones soportadas por los activos dentro de blockchain (lógica del negocio). De esta forma, como se mencionó anteriormente, los contratos inteligentes se empaquetan en una cadena de códigos (chaincode). Todos los contratos inteligentes que forman parte del chaincode están disponibles para las aplicaciones que genera el usuario.

Debe quedar claro que la cadena de código, como los contratos inteligentes que ella contiene, son todas operaciones conocidas y acordadas por todos los integrantes de la blockchain.

En el caso del ejemplo graficado en 8.2.1, se muestra un chaincode S1 que accede al libro mayor L1 y un chaincode S2 que accede al libro mayor L2.

La figura 8.2.10.1 permite distinguir una aplicación (a la izquierda del gráfico) que pertenece a una organización "1" y a la derecha una cadena de código (chaincode), que en este caso está compuesta por dos contratos inteligentes (smart contract). El primer contrato inteligente, va a insertar un log en el bloque, incorporando los atributos de un log de auditoría. El segundo contrato inteligente, va a recuperar un log de auditoría a partir de una clave compuesta: "or+ip+ts" (organización, dirección ip y marca de tiempo).

[44] Go, Java y JavaScript son lenguajes soportados en Hyperledger Fabric que pueden invocar API's desarrolladas por la comunidad para acceder al libro mayor. Esas API's se conocen como Fabric-SDK.
[45] Hyperledger Fabric SDK permite que las aplicaciones interactúen con una red de blockchain de Fabric. Proporciona una API simple para enviar transacciones a un libro mayor o consultar el contenido de un libro mayor con un código mínimo.

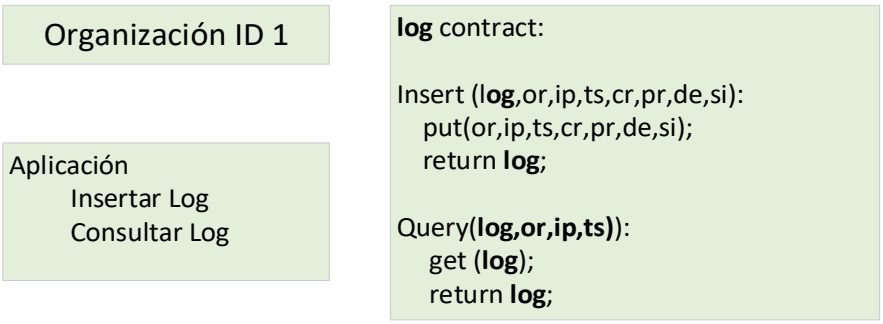

Figura 8.2.10.1: Aplicación, cadena de código y contrato inteligente

8.3. Flujo de una transacción en Fabric

Una vez descripto todos los componentes de la plataforma Fabric es momento de describir el proceso de una transacción que se propone desde una aplicación cliente.

El flujo a describir supone que existe un canal configurado y funcionando. El usuario de la aplicación que inserta los registros de auditoría, se ha registrado e inscrito en la Autoridad de certificación (CA) de la organización y ha recibido el material criptográfico necesario, que se utiliza para autenticarse en la red.

El chaincode (que contiene un conjunto de pares clave-valor que representan el estado inicial del bloque) está instalado en los pares e instanciado en el canal. El chaincode contiene lógica que define un conjunto de instrucciones de la transacción para insertar un nuevo registro. También se presume que está establecida una política de aprobación para este chaincode, que determina que tanto peerA como peerB deben aprobar cualquier transacción.

I. La aplicación cliente inicia la transacción (Transaction Proposal)

La aplicación A está enviando una solicitud para insertar un nuevo registro de log. Esta solicitud se dirige a peerA y peerB, que son respectivamente representantes de la aplicación cliente. La política de aprobación establece que ambos pares deben respaldar cualquier transacción, por lo tanto, la solicitud va a peerA y peerB. Cada aplicación puede tener una política de aprobación distinta. Cada una de esas políticas de aprobación constituyen el consenso a nivel nodos "peer".

A continuación, se construye la propuesta de transacción (Figura 8.3.1). Una aplicación que aprovecha un SDK compatible, en este caso Node utiliza una de las API disponibles para generar una propuesta de transacción. La propuesta es una solicitud para invocar una función chaincode con ciertos parámetros de entrada, con la intención de leer y / o actualizar el libro mayor.

El SDK sirve como un envoltorio para empaquetar la propuesta de transacción en el formato correctamente diseñado (búfer de protocolo sobre gRPC) y toma las credenciales criptográficas del usuario para producir una firma única para esta propuesta de transacción.

Figura 8.3.1: Aplicación cliente inicia una propuesta de transacción
Fuente: http://hyperledger.org

II. Los nodos pares aprobadores (Endorsing peers), ejecutan y verifican la transacción

Los pares que tienen la función de aprobar, verifican: que la propuesta de transacción está bien formada, que no se haya presentado en el pasado (protección de repetición de ataques), que la firma sea válida (usando el MSP) y que el remitente (cliente de la aplicación A) esté debidamente autorizado para realizar la operación propuesta en ese canal

Los pares aprobadores toman la propuesta de transacción como argumento para la función de chaincode invocada (Figura 8.3.2). El chaincode se ejecuta contra la base de datos del estado actual para producir los resultados de la transacción, que incluyen un valor de respuesta, un conjunto de lectura y un conjunto de escritura (es decir, pares clave / valor que representan un activo para crear o actualizar).

En esta instancia no se realizan actualizaciones en el libro mayor. El conjunto de estos valores, junto con la firma del par aprobador que lo respalda, se devuelve como una "respuesta de propuesta" al SDK que analiza la carga útil para que la aplicación la consuma.

El MSP es un componente instalado en los *"peers"* que permite a los pares verificar las solicitudes de transacciones que llegan de los clientes y firmar los resultados de las transacciones (endosos). La política de escritura se define en el momento de la creación del canal y determina qué usuarios tienen derecho a enviar una transacción a ese canal

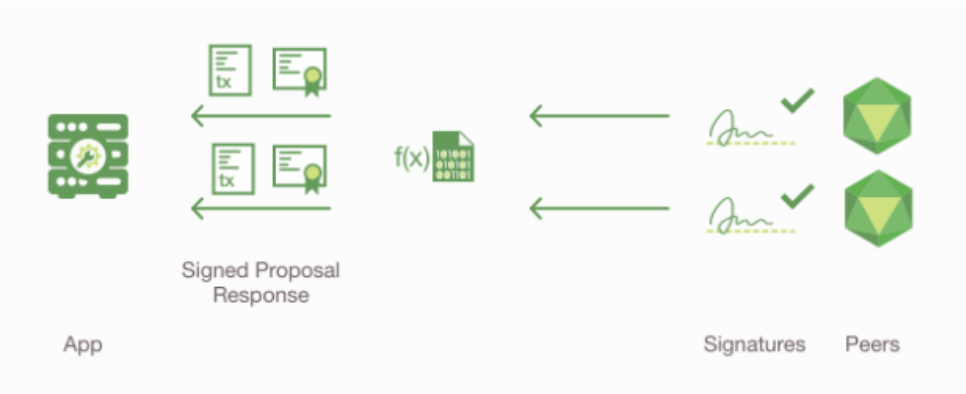

Figura 8.3.2: Los pares aprobadores ejecutan y verifican la transacción
Fuente: http://hyperledger.org

III. Las respuestas de la propuesta son inspeccionadas (Proposal Response)

La aplicación verifica las firmas aprobadoras y compara las respuestas de la propuesta para determinar si son las mismas (Figura 8.3.3).

Si el chaincode sólo consulta el libro mayor, la aplicación inspeccionará la respuesta de la consulta y, por lo general, no presentará la transacción al servicio de pedidos (Ordering Service).

Si la aplicación del cliente implica actualizar el libro mayor, ésta determina si la política de aprobación especificada en la respuesta se ha cumplido. En este ejemplo, si tanto peerA como peerB han aprobado la transacción.

La arquitectura es tal que incluso si una aplicación elige no inspeccionar las respuestas y reenvía una transacción no respaldada (aprobada), la política de aprobación aún será aplicada por los pares y se mantendrá en la fase de validación de confirmación.

Figura 8.3.3: Inspección de la respuesta a la propuesta
Fuente: http://hyperledger.org

IV. La aplicación cliente reúne todos los endosos dentro de una transacción y los envía al servicio de pedidos, quién los ordena.

La aplicación "transmite" la propuesta de transacción y la respuesta. Utiliza un "mensaje de transacción" que le envía al servicio de pedidos (Ordering Service).

La transacción contendrá los conjuntos de lectura / escritura, las firmas los pares aprobadores que avalan y la ID del canal. El servicio de ordenamiento (pedidos) no necesita inspeccionar todo el contenido de una transacción para realizar su operación, simplemente recibe transacciones de todos los canales de la red, las ordena cronológicamente por canal y crea bloques de transacciones por canal (Figura 8.3.4). Esos bloques son definitivos y formaran parte de la blockchain.

Para mantener la integridad del orden de las transacciones utiliza uno de los mecanismos de consenso a nivel nodos de servicio: solo, Raft o Kafka

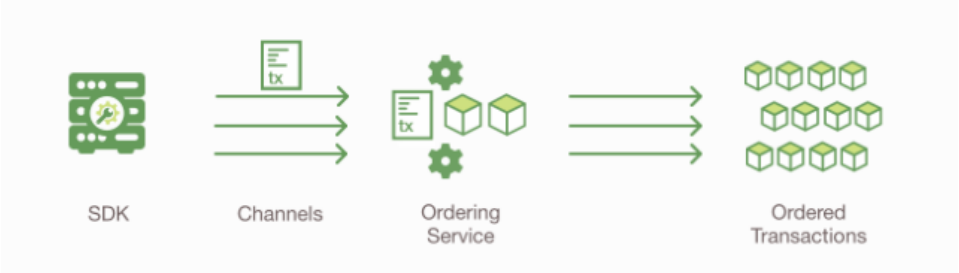

Figura 8.3.4: Cliente transmite el pedido al servicio de pedidos y éste los ordena
Fuente: http://hyperledger.org

V. El servicio de ordenamiento envía por un mensaje de broadcast el bloque de transacciones a todos los “peers” (aprobadores y validadores). Las transacciones dentro del bloque son aprobadas y validadas en cada par.

Los bloques de transacciones se "entregan" a todos los pares en el canal. Las transacciones dentro del bloque se validan para garantizar que se cumpla la política de aprobación y para garantizar que no haya habido cambios en el estado del libro mayor para las variables del conjunto de lectura desde que la ejecución de la transacción generó el conjunto de lectura. Las transacciones en el bloque se etiquetan como válidas o no válidas (Figura 8.3.5).

Figura 8.3.5: Transacciones de cada bloque son validadas
Fuente: http://hyperledger.org

VI. El libro mayor actualizado

Cada par agrega el bloque a la cadena del canal, y para cada transacción válida, los "conjuntos de escritura" se confirman en la base de datos del estado actual. Se emite un evento para notificar a la aplicación del cliente que la transacción (inserción del registro de logs) se ha agregado de manera inmutable a la cadena, así como la notificación de si la transacción se validó o no (Figura 8.5.6).

Las aplicaciones deben escuchar el evento de transacción después de enviar el mismo. Pueden utilizar la API *"submitTransaction"*, la cual escucha automáticamente los eventos de transacción. Si no se tiene en cuenta este aspecto, nunca se sabrá si la transacción realmente ha sido ordenada, validada y comprometida con el libro mayor.

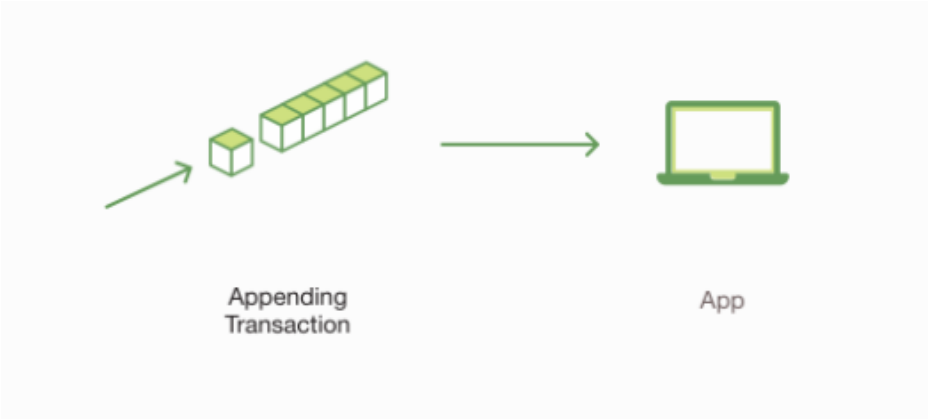

Figura 8.3.6: El libro mayor es actualizado
Fuente: http://hyperledger.org

8.4. Conclusiones del capítulo

A partir del nacimiento de blockchain como una infraestructura subyacente para la implementación de Bitcoin, esta tecnología no ha parado de evolucionar. Han surgido una gran cantidad de propuestas de cadena de bloques que van captando segmentos del mercado de acuerdo a las distintas prestaciones desarrolladas.

En el presente capítulo, en primer lugar, se justificó la elección de Hyperledger Fabric, como plataforma para la implementación de la solución. La fundamentación está basada en el análisis de distintos documentos científicos que avalan una muy buena combinación de perfomance, seguridad y costo de despliegue; propiedades que caracterizan al proyecto Fabric.

En segundo lugar, se describieron los componentes de una red Hyperledger Fabric, se detallaron las funciones de cada uno de ellos y se describieron los alcances y las prestaciones que forman parte de la plataforma.

Por último, se desarrolló una descripción detallada del proceso por el cual se inserta un nuevo bloque a la cadena de blockchain. De esta forma se ha documentado con precisión cuales son los procesos de consenso, los cuales garantizan el orden y el contenido de las transacciones dentro de cada bloque

9. Capítulo 9: Implementación de la solución

Se va a desplegar una red Hyperledger Fabric que va a constar de tres organizaciones: Org1, Org2 y Orderer. En cada organización Org1 y Org2 se van a definir dos peers: peer0 y peer1. En la organización Orderer, se va definir un orderer. Se definirán dos autoridades de certificados, uno para cada organización: ca_peerorg1 y ca_peerorg2. Se definirá un canal llamado mychannel. Cada uno de los peers va a mantener una instancia de una base de datos couchdb

Los registros provenientes del router Mikrotik se almacenarán en una base de datos Mysql instalada en el servidor central, será llamada *"syslog"* y contendrá una tabla con nombre *"auditoría"*.

Para la validación de la solución propuesta (a describir en el próximo capítulo) se deben desarrollar aplicaciones tanto en el Front End como en el Back End. Para generar la aplicación "cliente" se codificará usando "React"[46] y para la interacción con la blockchain se desarrollará con las herramientas JavaScript[47], Node.js[48] y las API's de Fabric-SDK-Node.

La siguiente figura 9.1 resume la arquitectura de la solución. Los registros de eventos son captados a través del protocolo syslog-ng por el servidor centralizado de registros quién los recibe en un buffer y los almacena en la base de datos Mysql. Luego un proceso "cliente" escrito en lenguaje JavaScript (P1, explicado en 7.3) los extrae de la base de datos y luego envía a un proceso "server" que representa la interface con blockchain. La mencionada interface está desarrollada utilizando lenguaje JavaScript y accede a los dos smart contract (altas y consultas) utilizando las API's de Fabric-SDK-Node. De esta forma, se pueden ejecutar los procesos de altas en la cadena de bloques (P2, explicado en 7.3), como así también consultas (P3, explicado en 7.3). Por último, un usuario auditor de los registros de eventos de auditoría, inicia la aplicación desarrollada para el Front End (escrita en React) y puede consultar tanto la base de datos Mysql, como la blockchain (invocando la API correspondiente de Fabric-SDK-Node), para producir listados de verificación de integridad (P4, explicado en 7.3).

[46] React es una biblioteca JavaScript de código abierto diseñada para crear interfaces de usuario con el objetivo de facilitar el desarrollo de aplicaciones en una sola página.
[47] JavaScript (abreviado comúnmente JS) es un lenguaje de programación interpretado y orientado a objeto
[48] Node.js es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor (pero no limitándose a ello) basado en el lenguaje de programación JavaScript

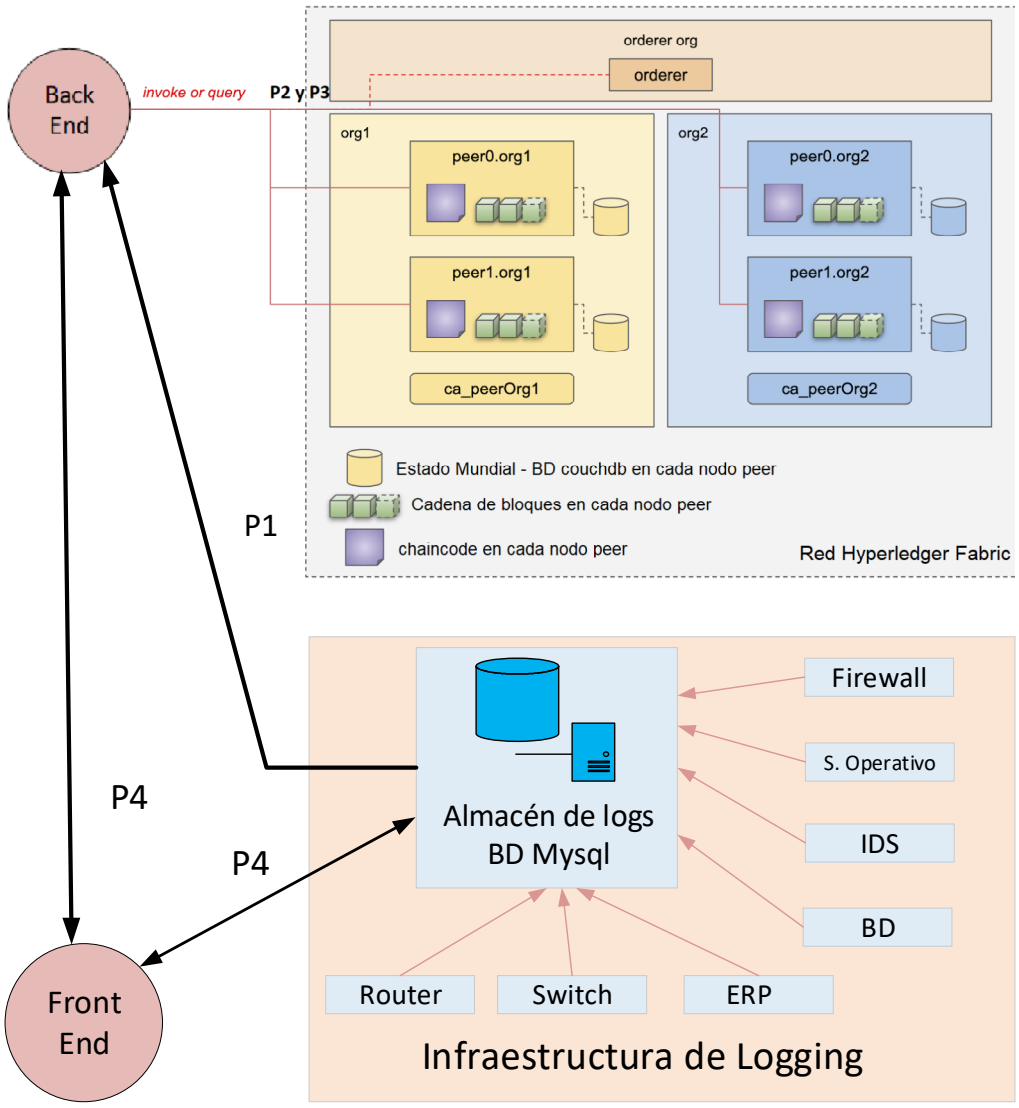

Figura 9.1: Arquitectura de la implementación de la solución

9.1. Configuración del Servidor Central de Logging con syslog-ng y Mysql

El primer paso para implementar la solución implica configurar un espacio de almacenamiento centralizado de todos los registros de eventos. Se implementa un servidor con un sistema operativo Ubuntu 18.04.2 LTS Bionic. Se instalan, el protocolo de transferencia de registros syslog-ng versión 3.13 y una base de datos Mysql versión 14.14. En el Anexo C se describe con mayor detalle el protocolo syslog-ng.

Pasos para configurar el servidor syslog-ng y la base de datos Mysql

1. Instalar syslog-ng, el servidor mysql, phpmyadmin y librerías asociadas con el siguiente comando:

```
$ apt-get -y install apache2 mc wget make gcc mysql-server mysql-
client curl phpmyadmin libdbd-pgsql aptitude libboost-system-dev
libboost-thread-dev libboost-regex-dev syslog-ng libmongo-
client0 libesmtp6 syslog-ng-mod-sql libdbd-mysql libdbd-mysql
```

2. Editar y configurar el archivo /etc/syslog-ng/syslog-ng.conf, para capturar mensajes de toda la red, luego filtra las direcciones de los dispositivos de origen que desea monitorear y lo envía a la base de datos syslog, figura 9.1.2:

```
#Configura el origen de los mensajes (toda la red utilizando el
puerto udp)
source red { udp (); };

#Filtra Mikrotik IP
filter f_mikrotik { host( "192.168.88.1" ); };

# Configura como destino mysql
destination d_mysql { pipe("/var/log/mysql.pipe" template("INSERT
INTO
logs (ip,pr,cr,ts,de) VALUES
('$HOST','$PRIORITY','$LEVEL','$YEAR-$MONTH-$DAY
$HOUR:$MIN:$SEC','$MSG');\n") template-escape(yes)); };

# Syslog-ng recibe los mensajes de la red, los filtra y luego los
envía a Mysql
log {
source(red);
filter(f_mikrotik);
destination(d_mysql);
};
```

Figura 9.1.2: Archivo de configuración de syslog-ng de la solución

3. Crear la base de datos *"syslog",* y dentro de ella la tabla *"auditoria"* usando los comandos de Mysql. Aquí se van a almacenar los registros

```
#Crea la base de datos para el protocolo syslog-ng
create database syslog;
use syslog;

#Crea la tabla en la db syslog
CREATE TABLE `auditoria` (
`id` int(11) NOT NULL AUTO_INCREMENT,
`secuencia` int(11) NOT NULL,
```

```
`organizacion` int(11) NOT NULL,
`ip` varchar(30) NOT NULL,
`datetime` datetime NOT NULL,
`level` varchar(45) NOT NULL,
`facility` varchar(45) NOT NULL,
`prioridad` varchar(45) NOT NULL,
`mensaje` varchar(45) NOT NULL,
`tag` varchar(45) NOT NULL,
`firma` varchar(45) NOT NULL
`program` varchar(45) NOT NULL
PRIMARY KEY (`id`)
KEY `organización` (`organización`),
KEY `ip` (`ip`),
KEY `datetime` (`datetime`),
KEY `priridad` (`pr`),
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
exit;
```

4. Crear un script que inicie un canal (pipe) que comunique los registros que llegan por syslog-ng con la base de datos. El canal debe tener permiso de ejecución

```
$ mkdir /temp
$ touch /temp/canal.sh
$ chmod +x /temp/canal.sh
```

Ejecutar un editor de texto y escribir lo siguiente en el archivo canal.sh:

```
$ nano /temp/canal.sh
#!/bin/bash
SQLID="root"
SQLPASS="1234"
export MYSQL_PWD=$SQLPASS
if [ ! -e /var/log/mysql.pipe ]
then
mkfifo /var/log/mysql.pipe
fi
while [ -e /var/log/mysql.pipe ]
do
mysql -u$SQLID syslog  </var/log/mysql.pipe > /dev/null
done
```

9.2. Configuración de las Políticas de Logging en el Origen

Se activa la registración de eventos en el dispositivo origen. En este caso, se configura un router Mikrotik AP series. Se inicia la interface web de configuración (la ip por defecto es 192.168.88.1, y el usuario es *"admin"*, sin clave).

Se accede en la sección izquierda del navegador a “System -> “Logging”, dónde se puede apreciar dos pestañas: Reglas y Acciones.

La figura 9.2.1 muestra la pantalla para configurar una regla que indica que los registros del tipo *“error”* sean enviados a un destino *“remote”*.

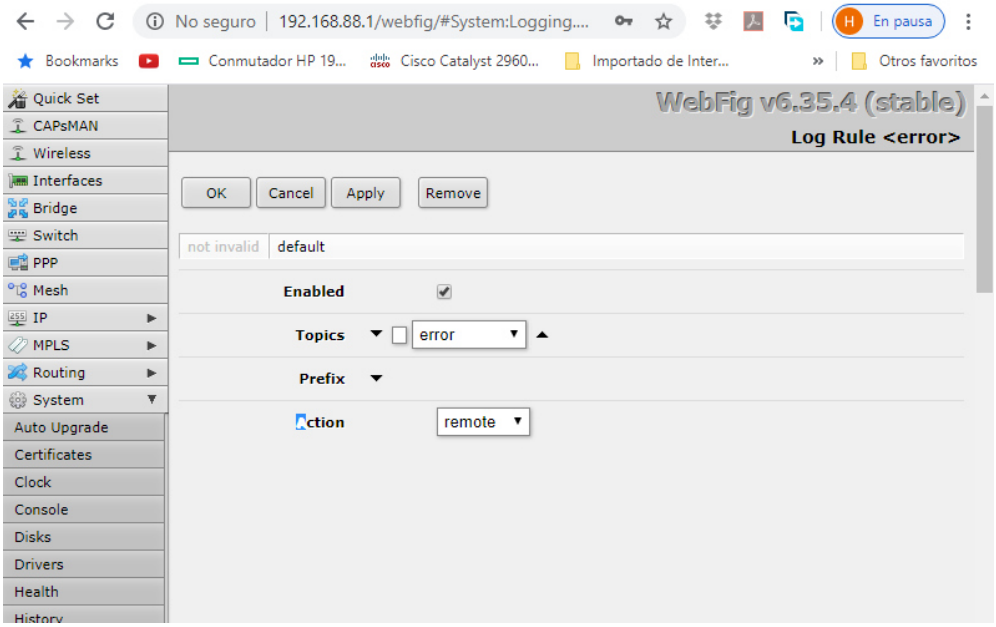

Figura 9.2.1 – Configuración de una regla en Mikrotik

La figura 9.2.2 muestra la pantalla para configurar una Acción, esto implica poder elegir dónde se mostrarán o almacenarán los registros configurados en las Reglas. En este caso, se configura la acción *“remote”* con la dirección ip y el puerto del servidor de almacenamiento de registros, para enviarlos usando el protocolo syslog-ng.

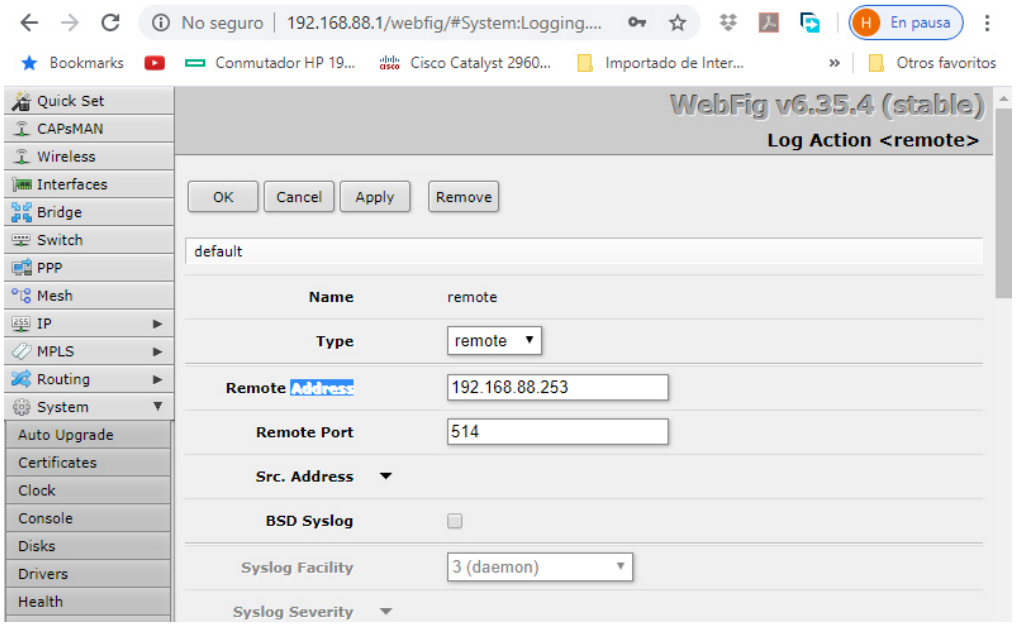

Figura 9.2.2 – Configuración de envío de los registros usando syslog-ng

9.3. Configuración de Túnel VPN (opcional)

En determinados escenarios, sobre todo cuando los registros de auditoría deben "viajar" a través de la internet pública, existe la posibilidad de brindar confidencialidad e integridad a los datos que viajan por la autopista insegura.

Ante la necesidad de generar un canal privado, existen numerosos protocolos de tunelización, como los descriptos en la sección 2 del capítulo 7 (IPSec, PPTP, L2TP, entre otros). En todos los casos la configuración va a depender de los sistemas operativos de los routers que conectan las puntas del túnel.

La figura 9.3.1 muestra un escenario en el cuál el servidor centralizado de logs colecta información de una red remota, la cual se encuentra conectada con la red local a través de Internet. Se configura un túnel para proteger la integridad y la confidencialidad de los datos.

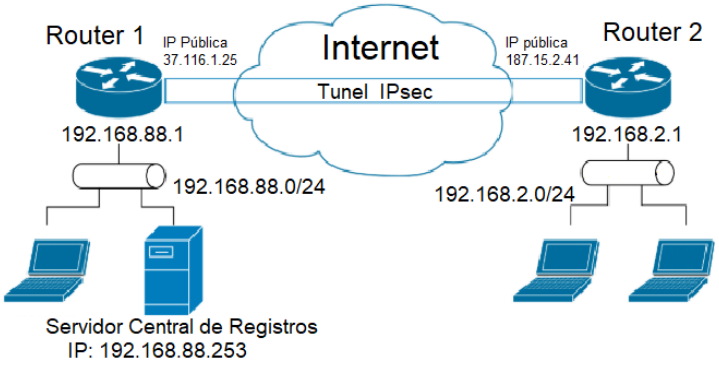

Figura 9.3.1: Tunel IPSec para proteger datos a través de internet

Se puede observar la generación de un túnel, en este caso usando IPsec para dotar a los datos de una capa de confidencialidad e integridad. La configuración de los routers, en caso de ser de la familia de equipos Mikrotik, es muy sencilla. A continuación, se muestra un script que permite configurar ambos routers (para el ejemplo si tuvieron en cuenta dos routers Mikrotik hAP):

- ✓ Iniciar sesión en cada uno de los router Mikrotik (http://192.168.88.1 y http://192.168.2.1): Login: *"admin"* y Password: por defecto está en blanco
- ✓ En el menú vertical, a la izquierda de la pantalla, elegir la opción "New Terminal"
- ✓ Copiar y pegar el código que está en la figura 9.3.2
- ✓ Cerrar la terminal, en el menú de la izquierda de la pantalla elegir "System" y luego "Reboot", para reiniciar cada uno de los routers.

```
#Configuración del Router "A"
/ip ipsec peer
add address=187.15.2.41 secret="ipsec-pass" port=500
auth-method=pre-shared-key

/ip ipsec proposal
add name="ipsec" auth-algorithms=sha1 enc-algorithms=3des
lifetime=30m pfs-group=modp1024

/ip ipsec policy
add proposal="ipsec" src-address=192.168.88.0/24
dst-address=192.168.2.0/24 sa-src-address=37.116.1.25
sa-dst-address=187.15.2.41 tunnel=yes

/ip firewall nat
add action=accept chain=srcnat src-address=192.168.88.0/24
dst-address=192.168.2.0/24 place-before=0


#Configuración del Router "B"
/ip ipsec peer
add address=37.116.1.25 secret="ipsec-pass" port=500
auth-method=pre-shared-key

/ip ipsec proposal
add name="ipsec" auth-algorithms=sha1 enc-algorithms=3des
lifetime=30m pfs-group=modp1024

/ip ipsec policy
add proposal="ipsec" src-address=192.168.2.0/24
dst-address=192.168.88.0/24 sa-src-address=187.15.2.41
sa-dst-address=37.116.1.25 tunnel=yes

/ip firewall nat
add action=accept chain=srcnat src-address=192.168.2.0/24
dst-address=192.168.88.0/24 place-before=0
```

Figura 9.3.2: Secuencia de comandos para configurar un túnel en Mikrotik

9.4. Instalación y Configuración de Hyperledger Fabric

La implementación de una red Fabric se hace a través de imágenes que se instancian en contenedores, utilizando una solución de virtualización de máxima perfomance como lo es Docker. Para hacer eficiente el despliegue e inicialización de todos los contenedores Docker que definen a Hyperledger Fabric, se utilizan archivos *"yaml",* los cuales son generados por los arquitectos de la red y luego ejecutados con una de las herramientas de Docker, llamada docker-compose. En resumen, las imágenes generadas por la herramienta docker se instancian en distintos contenedores a través de la ejecución

de los comandos docker-compose junto a los archivos *"yaml"*. En el Anexo C se describe Docker y una de sus herramientas claves usadas en la solución: docker-compose,

9.4.1. Prerrequisitos para instalar la red Hyperledger Fabric

Antes de poner en funcionamiento Hyperledger es necesario instalar las herramientas que permitirán el despliegue y la ejecución de los contratos. En este caso, como ya fue mencionado, se implementará Hyperledger Fabric sobre la plataforma operativa de Ubuntu 18.4. Los pasos a seguir son:

- ✓ Instalar y actualizar Ubuntu v18.04.2 LTS bionic
- ✓ Instalar cURL
- ✓ Instalar Docker v18.09.7
- ✓ Instalar Docker Compose v1.23.1
- ✓ Instalar node.js v8.10.0
- ✓ Instalar npm v5.6.0
- ✓ Instalar Python v2.7.15+
- ✓ Crear una carpeta y luego instalar los binarios, las imágenes Docker y los ejemplos:

```
$: cd /usr/local/
$: sudo mkdir hyperledger
$: cd Hyperledger
$: curl -sSL http://bit.ly/2ysbOFE | bash -s
```

El comando *"curl"* primero descarga implementaciones de redes y aplicaciones de ejemplos con sus respectivos archivos *"yaml"* para usar con docker-compose (a partir de una de las redes de ejemplos se desarrolló esta implementación), luego descarga las herramientas para generar la red (archivos binarios) y por último clona las imágenes de los objetos característicos de Hyperledger Fabric: ca, peer, orderer, couchdb, javaenv, tolos, entre otras.

9.4.2. Generar los certificados para las organizaciones y los peers

Se deben generar los certificados X509 y las claves privadas de todas las entidades que participan de la red. El material criptográfico generado es usado para la encriptación de datos y la autenticación de identidades. Se debe invocar la herramienta *"cryptogen"*, un archivo binario que fue descargado en 9.4.1, el cual consume un archivo llamado *"crypto-config.yaml"*(ver contenido en Anexo B). En este último se indica el nombre de las organizaciones, la cantidad de peers y usuarios que tendrá cada una, como así también la cantidad y los nombres de los orderers. Se debe ejecutar el siguiente comando:

```
$: cryptogen generate  --config  ./crypto-config.yaml
```

Luego de ejecutado el anterior comando se crea una carpeta llamada *"crypto-config"* que contiene todos los certificados y las claves de las entidades que se identificarán en la red.

9.4.3. Generar el servicio de pedidos, canal y dos pares de borde

Es el momento de generar cuatro elementos claves en la arquitectura de la solución: Orderer Génesis Block, el canal de transacciones y dos peers de borde (uno para cada organización).

Se debe invocar la herramienta *"configtxgen"*, un archivo binario que fue descargado en 9.4.1, el cual consume un archivo llamado *"configtx.yaml"(* ver contenido en Anexo B). El mencionado archivo es clave en la configuración de la red. En él se debe definir los atributos de la red en cuanto a: organizaciones, aplicaciones, orderer, políticas y tipos de perfiles.

En el caso de la presente solución, se ejecutan cuatro comandos para generar los respectivos cuatro "artefactos" esenciales de la red: el bloque génesis, el canal y los dos pares de anclaje.

Lo primero que se debe hacer es configurar la variable de entorno correspondiente para que *"configtxgen"* pueda acceder al archivo *"configtx.yaml"*, el comando es el siguiente*:*

```
$ export FABRIC_CFG_PATH=$PWD)
```

Luego deben ejecutarse los siguientes cuatro comandos:

```
$: ../bin/configtxgen -profile TwoOrgsOrdererGenesis -outputBlock
./channel-artifacts/genesis.block

$: export CHANNEL_NAME=mychannel  && ../bin/configtxgen -profile
TwoOrgsChannel -outputCreateChannelTx ./channel-
artifacts/channel.tx -channelID $CHANNEL_NAME

$: ../bin/configtxgen -profile TwoOrgsChannel -
outputAnchorPeersUpdate ./channel-artifacts/Org1MSPanchors.tx -
channelID $CHANNEL_NAME -asOrg Org1MSP

$: ../bin/configtxgen -profile TwoOrgsChannel -
outputAnchorPeersUpdate  ./channel-artifacts/Org2MSPanchors.tx -
channelID $CHANNEL_NAME -asOrg Org2MSP
```

9.4.4. Iniciar la red Hyperledger Fabric: generar los contenedores con docker-compose

Se debe iniciar la red compuesta por: autoridades de certificados, orderers, peers, canal, y el entorno de ejecución. Para hacer un uso eficiente de las herramientas docker, se personalizan los archivos de configuración "yaml" que fueron descargados en

el punto 9.4.1. Estos archivos son: "docker-compose-ca.yaml", "docker-compose-cli.yaml", "docker-compose-couchdb.yaml" (ver contenido en Anexo B). Los comandos a ejecutar en el terminal son:

```
$: docker-compose -f  docker-compose-ca.yaml up -d
$: docker-compose -f  docker-compose-cli.yaml up -d
$: docker-compose -f  docker-compose-couch.yaml up -d
```

9.4.5. Unir los peers y el orderer al canal creado. Actualizar peers de borde

Para unir los *"peers"* y el *"orderer"* se pueden ejecutar los comandos correspondientes desde un contenedor que ofrezca una terminal. Para ello se debe levantar un contenedor con el siguiente comando:

```
$: docker exec -it cli bash
```

Una vez dentro del terminal del contenedor se debe chequear las variables de entorno, con el comando *"env"*:

```
root@2s345: env
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fab
ric/peer/crypto/peerOrganizations/org1.example.com/users/Admin@org
1.example.com/msp

CORE_PEER_ADDRESS=peer0.org1.example.com:7051

CORE_PEER_LOCALMSPID="Org1MSP"

CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger
/fabric/peer/crypto/peerOrganizations/org1.example.com/peers/peer0
.org1.example.com/tls/ca.crt
```

Ejecutar el siguiente comando para unir el *"orderer"* (servicio de ordenamiento):

```
root@2s345: peer channel create -o orderer.example.com:7050 -c
$CHANNEL_NAME -f ./channel-artifacts/channel.tx --tls --cafile
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererO
rganizations/example.com/orderers/orderer.example.com/msp/tlscacer
ts/tlsca.example.com-cert.pem
```

Es momento de unir al canal los cuatro *"peers"* (peer0.org1, peer0.org2, peer1.org1 y peer1.org2). El proceso es similar al anterior, es necesario chequear las variables de entorno y luego ejecutar el correspondiente comando. A continuación, se muestra la secuencia para unir el peer0.org2:

```
root@2s345: env

CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fab
ric/peer/crypto/peerOrganizations/org2.example.com/users/Admin@org
2.example.com/msp

CORE_PEER_ADDRESS=peer0.org2.example.com:9051

CORE_PEER_LOCALMSPID="Org2MSP"

CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger
/fabric/peer/crypto/peerOrganizations/org2.example.com/peers/peer0
.org2.example.com/tls/ca.crt

root@2s345: peer channel join -b mychannel.block
```

Actualizar los *"peers"* de borde, en esencia es agregar información de configuración adicional en la parte superior del bloque de génesis del canal. Para definir a los peer0.org1 y peer0.org2 como pares de bordes, se deben controlar las variables de ambientes y correr los comandos respectivos. A continuación, la inscripción del peer0.org2 como par de borde:

```
root@2s345: env
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fab
ric/peer/crypto/peerOrganizations/org2.example.com/users/Admin@org
2.example.com/msp

CORE_PEER_ADDRESS=peer0.org2.example.com:9051

CORE_PEER_LOCALMSPID="Org2MSP"

CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger
/fabric/peer/crypto/peerOrganizations/org2.example.com/peers/peer0
.org2.example.com/tls/ca.crt

root@2s345: peer channel update -o orderer.example.com:7050 -c
$CHANNEL_NAME -f ./channel-artifacts/Org2MSPanchors.tx --tls --
cafile
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererO
rganizations/example.com/orderers/orderer.example.com/msp/tlscacer
ts/tlsca.example.com-cert.pem
```

9.4.6. Instalar e instanciar el chaincode

Las aplicaciones interactúan con el libro mayor de blockchain a través de chaincode. Como tal, es necesario instalar el chaincode en cada peer que lo ejecutará y luego respaldará las transacciones. Posteriormente se debe crear una instancia del chaincode en el canal.

En la presente solución, como se explicó al comienzo del capítulo, el chaincode está constituido por dos smart contract (P2 y P3) que están codificados con JavaScript, los cuales acceden al libro mayor usando las API's provistas por Fabric-SDK-Node. Los contratos inteligentes que implementan la solución al problema planteado están íntegramente desarrollados en el mencionado chaincode. En el Anexo A se adjuntan tres archivos que contienen el código escrito en lenguaje JavaScript y que representan la solución propuesta (Chaincode y dos smart contract).

Una vez definida la lógica del contrato (chaincode), el mismo se debe instalar en todos los "pares "que ejecutarán y respaldarán las transacciones. En el presente proyecto será instalado en un par de la organización 1 (peer0.org1) y en un par de la organización 2 (peer0.org2).

Para instalar el chaincode escrito en node.js en el par peer0.org1.example.com se debe ejecutar:

```
root@2s345: peer chaincode install -n fabcar -v 1.0 -l node -p
/opt/gopath/src/github.com/chaincode/fabcar/javascript/
```

Luego previo cambio en las variables de entorno, se debe ejecutar el mismo comando para instalar el chaincode escrito en node.js en el par peer0.org2.example.com

Es el momento de crear una instancia de chaincode en el canal. Esto inicializará el chaincode en el canal y establecerá la política de respaldo (endorsement) para el mismo. También, automáticamente, se iniciará un contenedor de chaincode para el peer que se está instanciando. El nombre de ese contenedor para el peer0.org1 es: dev-peer0.org1.example.com-fabcar-1.0.

El argumento –P es usado para especificar la política de aprobación (endorsement) requerida para que se valide una transacción contra este chaincode. Como se explicó, en esta solución se requerirán la aprobación de al menos un peer de la organización 1 y un peer de la organización 2 (dos endosos). Esto se logra agregando al comando de instanciación: *-P "AND ('Org1MSP.peer', 'Org2MSP.peer')"*

Para instanciar el chaincode en el par peer0.org1.example.com ejecutar el siguiente comando:

```
root@2s345: peer chaincode instantiate -o orderer.example.com:7050
--tls --cafile
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererO
rganizations/example.com/orderers/orderer.example.com/msp/tlscacer
ts/tlsca.example.com-  cert.pem -C $CHANNEL_NAME -n fabcar -l
node -v 1.0 -c '{"Args":[]}' -P "AND
('Org1MSP.peer','Org2MSP.peer')"
```

9.4.7. Ajustes finales de la configuración inicial

Cuando se genera la red por primera vez, es necesario que el archivo JSON que contiene la configuración de inicio de la misma (*connection-org1.json*) sea copiado desde la ubicación donde fue generado (*/fabric-samples/first-network*) a la carpeta */crypto-eventos/syslog-blockchain-master/server/network*. Una vez en el destino, se renombra el mencionado archivo con el siguiente valor: *config.json*

Otro aspecto que se debe considerar por única vez consiste en instalar todas las dependencias de Hyperledger Fabric para la aplicación escrita, en este caso JavaScript. Se debe instalar en dos carpetas: */crypto-eventos/syslog-blockchain-master/server* y */crypto-eventos/syslog-blockchain-master/cliente*. El comando es:

```
$: npm install
```

Posteriormente se iniciarán los servicios de la interface con blockchain, como así también de la aplicación en el Front End. La interface de la aplicación se ejecuta en el puerto 3000 del servidor y la interface de blockchain en el puerto 5000 Se debe ejecutar los siguientes comandos.

```
$: /crypto-eventos/syslog-blockchain-master/server  npm start

$: /crypto-eventos/syslog-blockchain-master/cliente  npm start
```

Se debe inscribir el usuario administrador (usuario: 'admin", contraseña: 'admin'). Este usuario administrador ha sido registrado en la CA de la Organización 1 en el momento de crear la red. Para inscribir el usuario admin, se debe ingresar el navegador: http:/localhost:5000/api/enroll-admin. La figura 9.4.7.1 muestra la operación realizada con éxito:

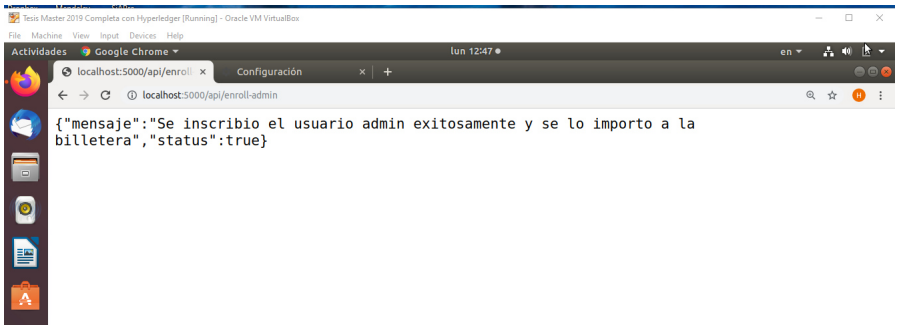

Figura 9.4.7.1: Inscribir el usuario administrador

Las actividades descriptas desde 9.4.2 hasta 9.4.6 inclusive están implementadas en un archivo script llamado *"starFabric.sh"* (ver contenido en Anexo B), el cual se ejecuta con el siguiente comando:

```
$: ./startFabric javascript
```

9.5. Conclusiones del capítulo

En el presente capítulo se detallaron todas las instancias que se deben transitar para poner a punto la red Fabric antes de iniciar el proceso de recolectar y salvar los registros de eventos. También se describió cuáles son las características y lenguajes utilizados en el desarrollo de la aplicación que se utiliza en la validación de la solución.

Todos los pasos enunciados fueron ejecutados en un servidor sobre el cual se instaló un software de virtualización, para luego configurar un servidor virtual montado sobre la plataforma VirtualBox[49]. Una vez instalado y configurado el servidor virtual, utilizando Linux como sistema operativo (distribución Ubuntu 18.04.2 LTS Bionic), se procedió a descargar las imágenes de Hyperledger Fabric, para luego personalizar la red

[49] Oracle VM VirtualBox (conocido generalmente como VirtualBox) es un software de virtualización para arquitecturas x86/amd64

blockchain instanciando los contenedores respectivos. Por último, se instaló la aplicación que fue desarrollada para cumplir con el objetivo de salvar los registros de eventos en la blockchain,

Con la entrega del soporte magnético, junto al documento de la presente tesis, se adjuntará la máquina virtual con las imágenes y contenedores que simulan la red fabric, como así también el código completo del proyecto de desarrollo de la solución, el cual está salvado en la carpeta *"/home/hugo/crypto-eventos"*

10. Capítulo 10. Validación de la solución

Para iniciar el proceso de validación de la solución es necesario generar eventos en el dispositivo elegido para las pruebas, en este caso, el router *Mikrotik hAP series.* Para ello, se va a forzar un inicio de sesión fallido con la cuenta "admin" (se ingresa una contraseña errónea a propósito), para luego ejecutar un inicio de sesión exitoso con la misma cuenta. La figura 10.1 muestra el inicio de sesión fallido en el router.

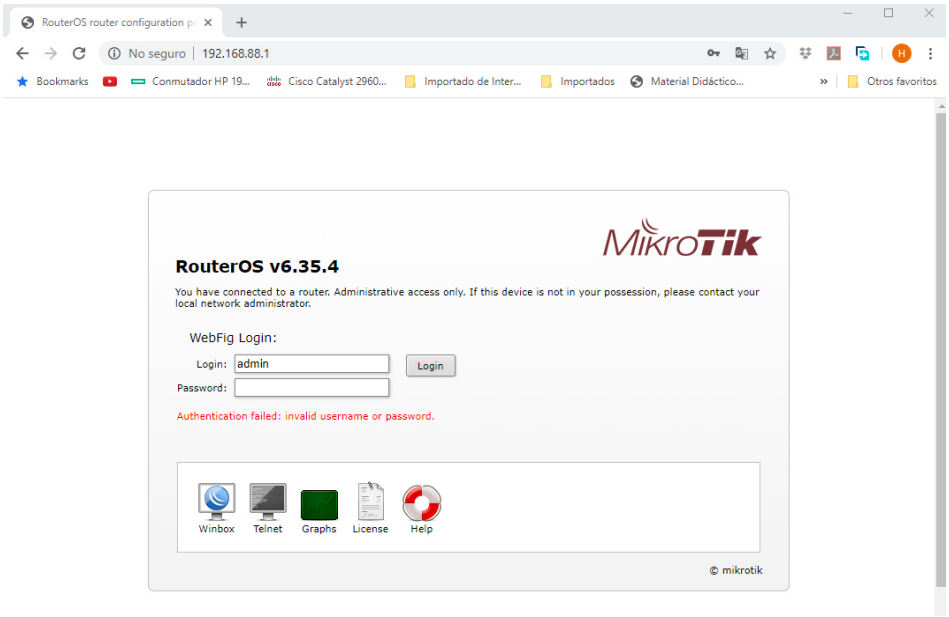

Figura 10.1: Inicia de sesión fallido en router de prueba

Una vez generado los eventos, se comprueba si los mismos fueron capturados por el protocolo syslog-ng y luego enviados a la base de datos Mysql para su almacenamiento central. La figura 10.2 muestra el resultado de la consulta realizada sobre la tabla "auditoria" en la base de datos *"syslog"*. Para mostrar la misma se hace uso de la herramienta *"phpmyadmin"*.

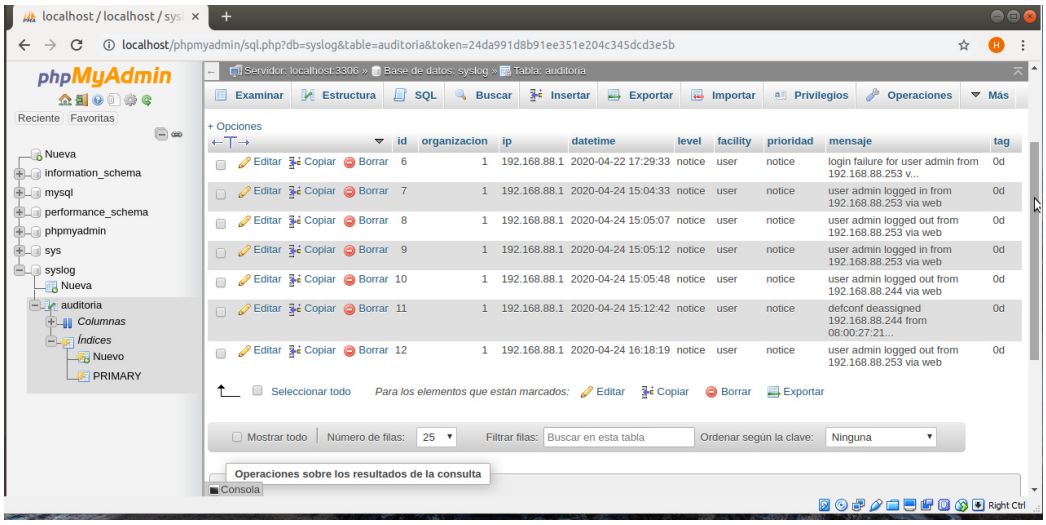

Figura 10.2: Registro de eventos capturados y almacenados en Msql

Luego de haber comprobado que los eventos están siendo capturados y enviados a la base de datos, es necesario comprobar que la red Hyperledger Fabric está levantada y todos sus componentes están activos a la espera de recibir solicitudes de actualizaciones y consultas. Las figuras 10.3 y 10.4 muestran los contenedores activos luego de ejecutar la instrucción *"docker ps"*. Ellos representan a los *"peers", "orderers", "Autoridades de Certificación" y "couchdb" (base de datos del estado mundial)*, junto a las configuraciones de canal, organizaciones, chaincode, entre otras.

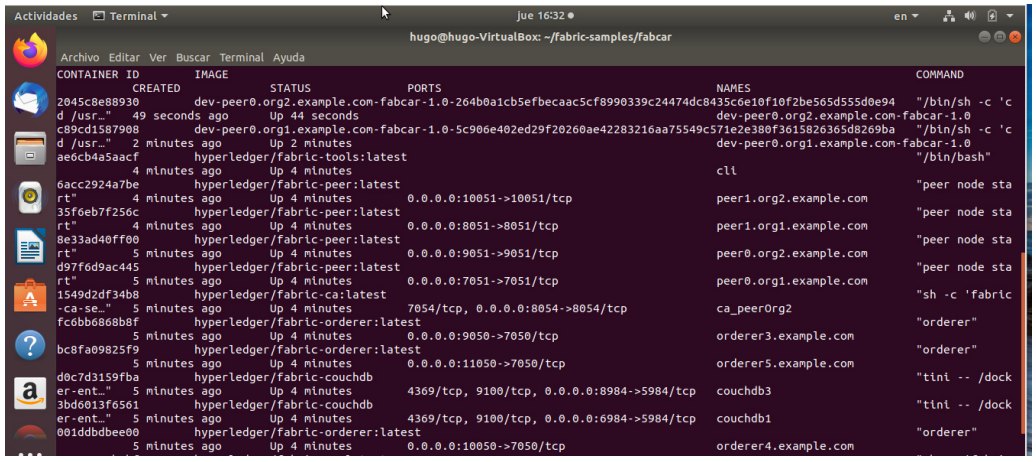

Figura 10.3: *Orderers, Peers y Base de Datos* instanciadas en contenedores (1)

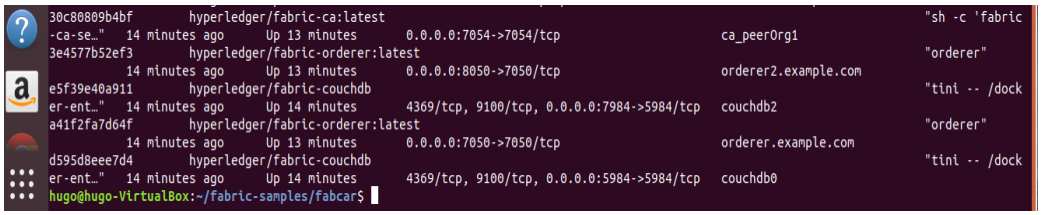

Figura 10.4: *Orderers, Peers y Base de Datos* instanciadas en contenedores (2)

Los nombres de los contenedores en ejecución son:

- ✓ cli
- ✓ peer1.org2.example.com
- ✓ peer1.org1.example.com
- ✓ peer0.org2.example.com
- ✓ peer0.org1.example.com
- ✓ ca.peerOrg2
- ✓ orderer3.example.com
- ✓ orderer5.example.com
- ✓ couchdb3
- ✓ couchdb1
- ✓ ca.peerOrg1
- ✓ orderer4.example.com
- ✓ couchdb2
- ✓ orderer.example.com
- ✓ couchdb0

La figura 10.5 muestra el resultado de ejecutar el comando *"netstat"*[50] en el servidor virtual, el cual confirma las direcciones ip y puertos que están activos en la red y que corresponden con las "máquinas virtuales" de la red Fabric instanciadas a través de los contenedores correspondientes.

[50] netstat (network statistics) es una herramienta de línea de comandos que muestra un listado de las conexiones activas de una computadora

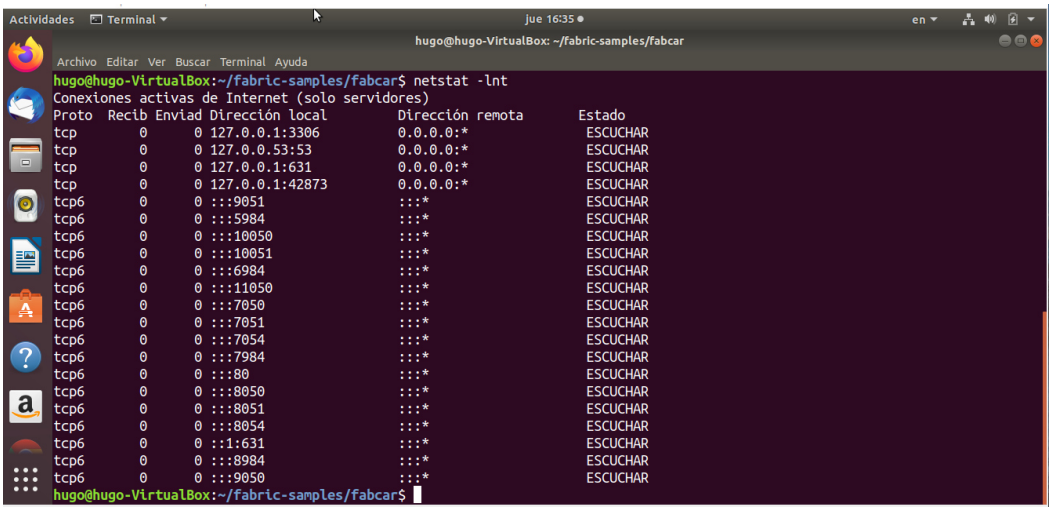

Figura 10.5: Puertos a la espera de conexiones

Luego de corroborar que la red Fabric está operativa y el protocolo syslog-ng está capturando registros de auditoría del router Mikrotik, se debe ingresar a la aplicación desarrollada, la cual se llama “Crypto-Eventos”. Para ello se debe escribir en el navegador la siguiente dirección: http:/localhost:3000. La figura 10.6 muestra la pantalla de acceso:

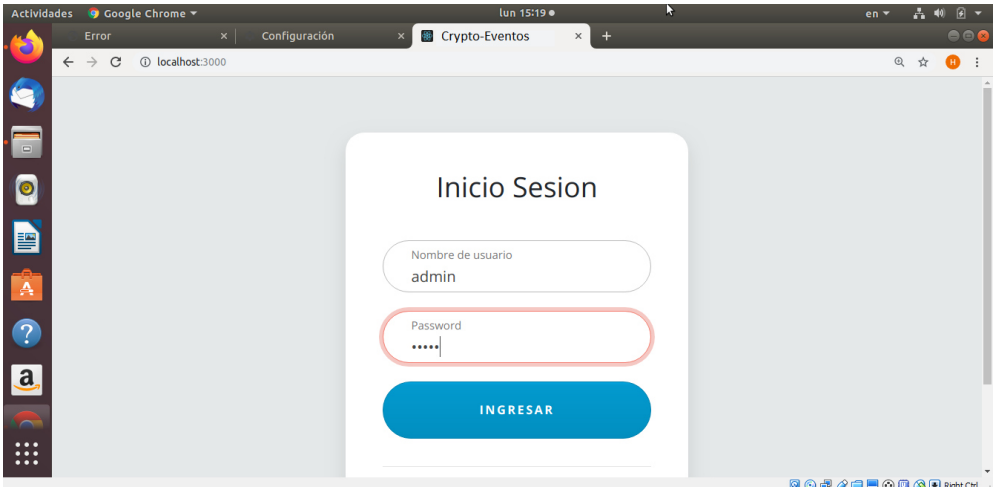

Figura 10.6: Inicio de sesión en el sistema

Una vez ingresado al sistema se puede apreciar los registros que están almacenados en la blockchain y los registros que se encuentran la base de datos a la espera de ser almacenados en la blockchain (fueron capturados por el protocolo syslog-ng). En la figura 10.7 se puede ver que no hay datos en la blockchain y que la base de datos tiene información disponible capturada a través de syslog-ng. En este caso, a los efectos de demostrar paso a paso el funcionamiento del sistema, los registros se agregan

manualmente. En un ambiente de producción, los registros se insertan a la blockchain en forma automática (como lo indica la solución). Para hacerlo, en este caso, en forma manual, hay que seleccionar el botón “Añadir”.

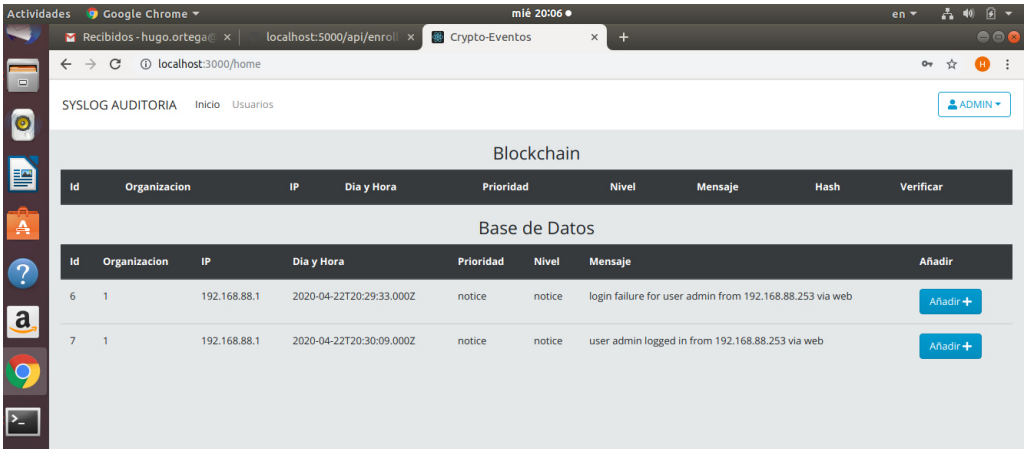

Figura 10.7: Vista de registros en la base de datos

Se añade el registro 6 en la blockchain seleccionando el botón *“Añadir”* (se controla que la clave sea única, para luego aplicar una función hash a los datos del registro y finalmente salvar en la blockchain dichos datos más el hash calculado).

La figura 10.8 muestra el registro añadido en la blockchain (registro 6 junto con el hash aplicado a todos sus datos). En este caso, la información que contiene el registro 6 y el hash calculado se guardan como una transacción dentro del bloque, y su contenido pasará al estado de inmutable.

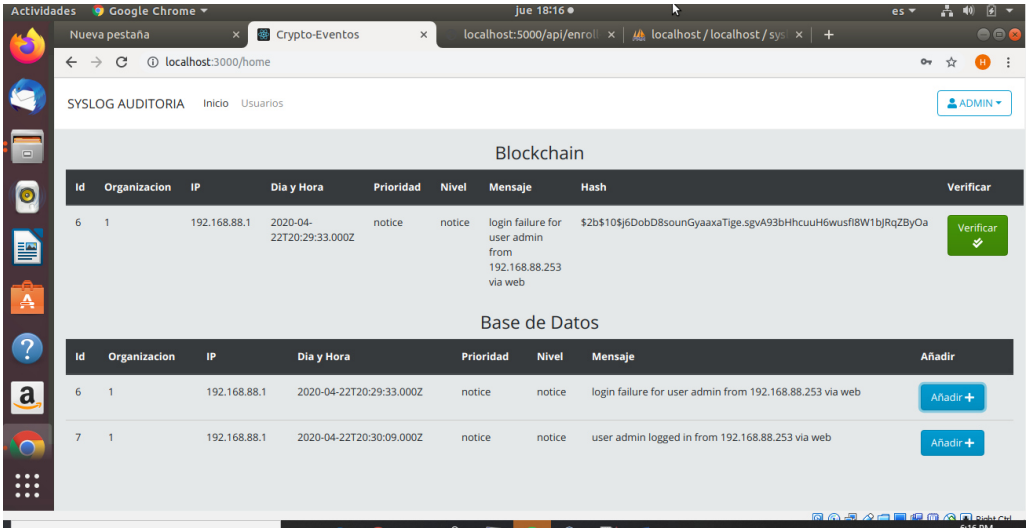

Figura 10.8: Registro de evento salvado en Hyperledger Fabric

Luego se procede a añadir el registro 7, operación que es exitosa, pudiéndose observar en la figura 10.9 que la blockchain puede listar dos registros: el registro 6 y 7. Luego se prueba la unicidad de la clave, intentando agregara nuevamente a la blockchain el registro 6, por lo cual el sistema nos devuelve un mensaje advirtiendo que el registro fue agregado. Esto también se puede observar en la figura 10.9

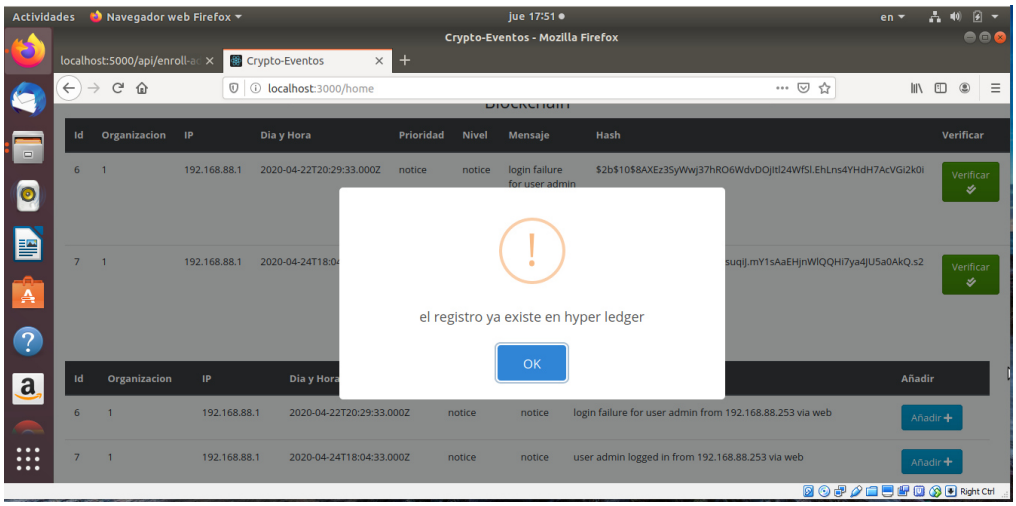

Figura 10.9: Prueba de detección de duplicidad de claves

El sistema contempla un módulo para crear usuarios. Estos pueden tener permisos para agregar y consultar registros de eventos, otros tendrán sólo acceso a las consultas. También está previsto la posibilidad de adicionar usuarios que tengan un perfil técnico avanzado y puedan cambiar las políticas de respaldo, agregar peers, modificar la cantidad de orderers y sus protocolos de consenso, entre otros perfiles que pueden ser definidos por el administrador. La figura 10.10 muestra una pantalla para la creación de usuarios, en la cual se configura por defecto una contraseña que tiene el mismo valor que el nombre del usuario (luego debe ser cambiada en el primer inicio de sesión).

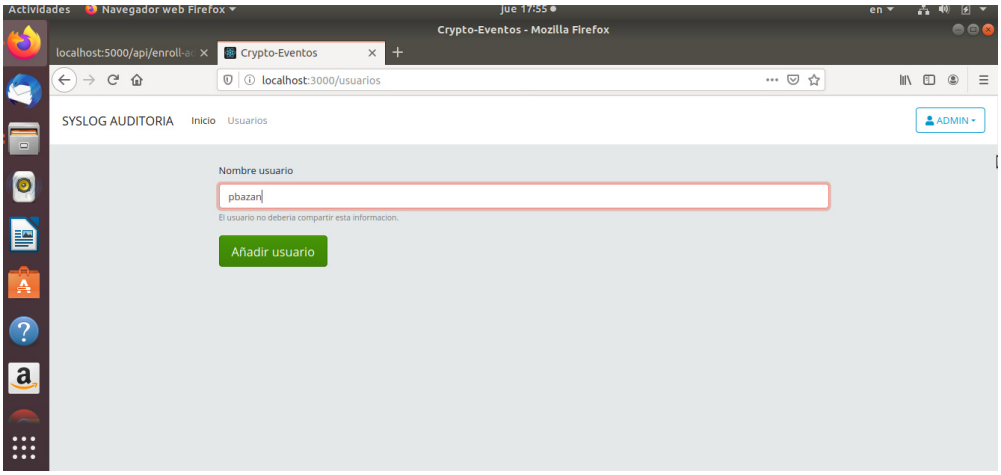

Figura 10.10: Creación de usuarios

10.1. Pruebas de inmutabilidad en la blockchain

En cuanto a la aplicación que genera las consultas se muestra la figura 10.1.1 en dónde se puede apreciar un proceso de consulta por registros de eventos almacenados en la blockchain. Éste muestra que el registro de auditoría con "ID" 6 permanece inalterado en la base de datos. Esto implica que la información en la base de datos permanece íntegra.

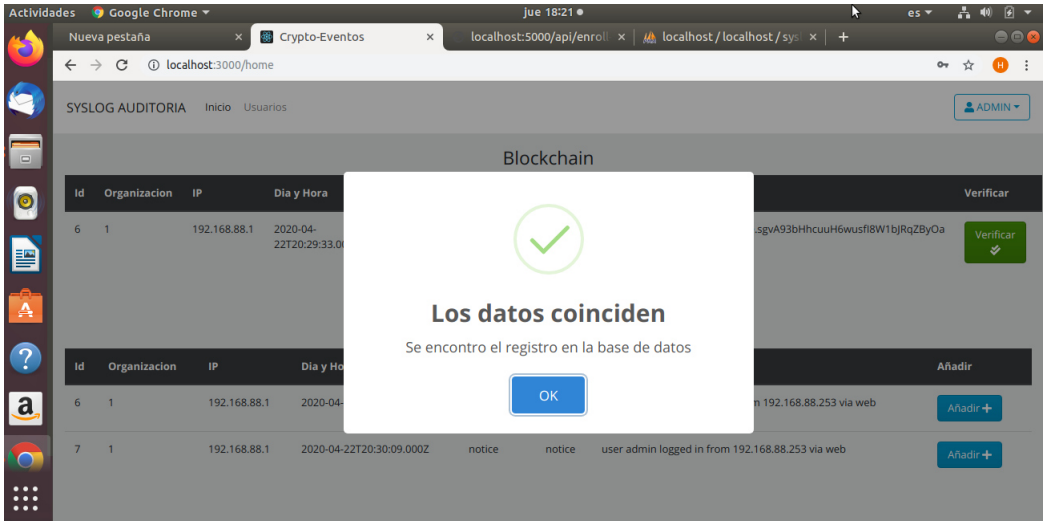

Figura 10.1.1: Auditoría de inmutabilidad de registro de evento exitosa

En la figura 10.1.2 la verificación de la integridad ha resultado negativa. En este caso se supone que un *"intruso"* ha tomado el control del perfil de un usuario que al menos tiene permiso de escritura en la tabla "auditoría" y ha ejecutado una operación por la cual adulteró información de la misma, en este caso, en el registro de auditoría con "ID" 7 (el campo ip tiene un valor 192.168.88.5 en la base de datos, mientras que en la blockchain mantiene el valor original: 192.168.88.1). Un perito informático o un especialista en auditorías deberá tomar una decisión al respecto.

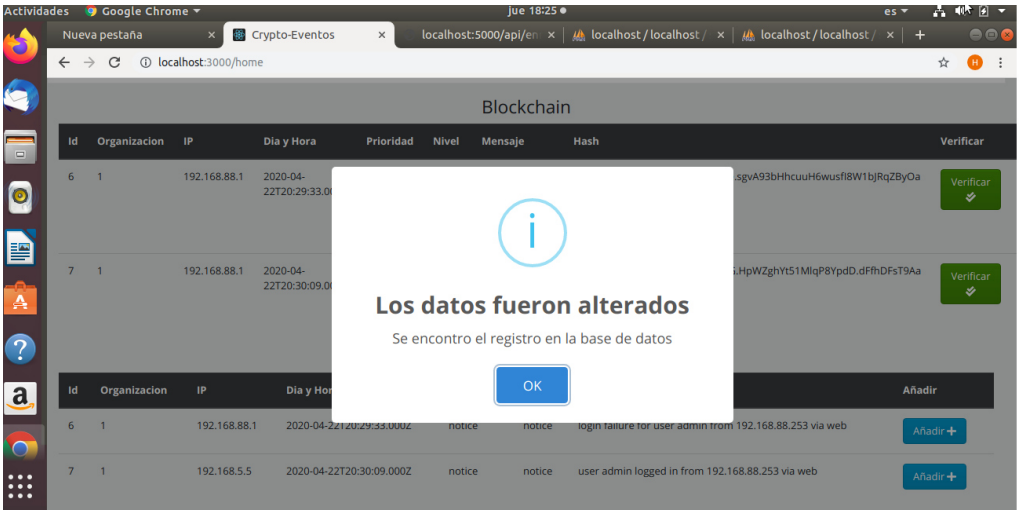

Figura 10.1.2: Auditoría de inmutabilidad de registro de evento fallida

En la próxima figura 10.1.3, se puede apreciar que luego de aplicar una auditoría a la integridad de la base de datos de registros de eventos, el registro con "ID" 7 fue eliminado de la base de datos. En este caso, nuevamente un *"intruso"* con perfil de un usuario que al menos tiene permiso de escritura sobre la tabla *"auditoría"* ha ido más lejos, eliminó completamente el registro, operación que sin lugar a dudas busca borrar pruebas de transacciones delictivas.

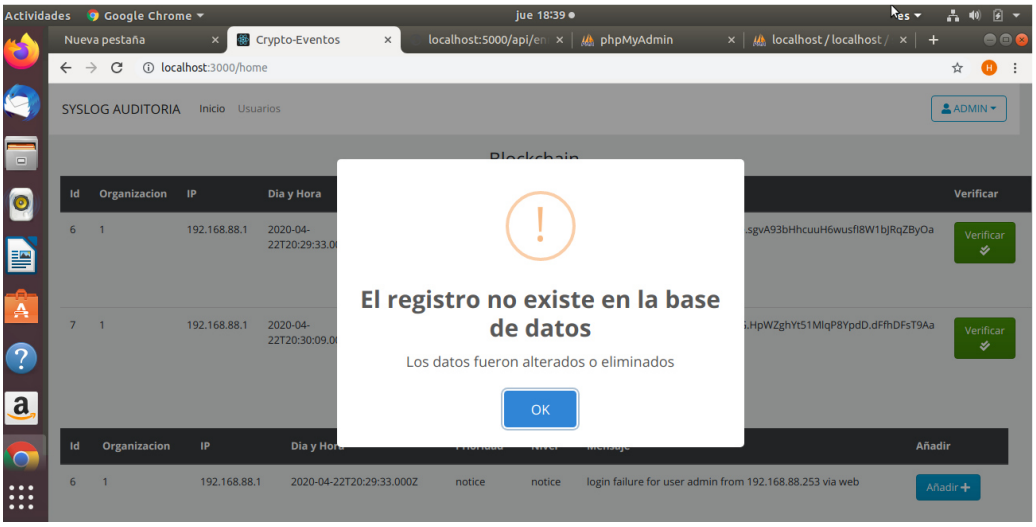

Figura 10.1.3: Detección de eliminación de registro de eventos

Respecto a la integridad de la información salvada en Hyperledger Fabric (los registros de auditoría), resulta extremadamente complejo alterar el contenido de una cadena de bloques. En los siguientes dos ejemplos se puede observar en primer lugar la complejidad de acceso y, en segundo lugar, la complejidad del formato de la información recuperada.

En la figura 10.1.3 se muestra el resultado de ejecutar un comando que proporciona información del último bloque generado en la blockchain. Para lograr esto, hay que cumplir ciertos requisitos: se debe contar con el rol de administrador de la organización a la que pertenece la cadena de bloques, tener permisos de lectura sobre el canal y acceso a los certificados digitales del *"peer"* y el *"orderer"* al que se le realiza la consulta. En este ejemplo podemos observar que se recupera el bloque 9 y se aprecia el valor del hash del bloque actual y el anterior. Con los datos recuperados, se puede verificar la integridad de la cadena de bloques, pero no existe margen en este caso para alterar la información de la blockchain. Por otra parte, teniendo en cuenta las restricciones de acceso, el supuesto "intruso", deberá tener conocimientos para penetrar sistemas y capacidad computacional para ejecutar aplicaciones que descifran claves de usuarios o servicios

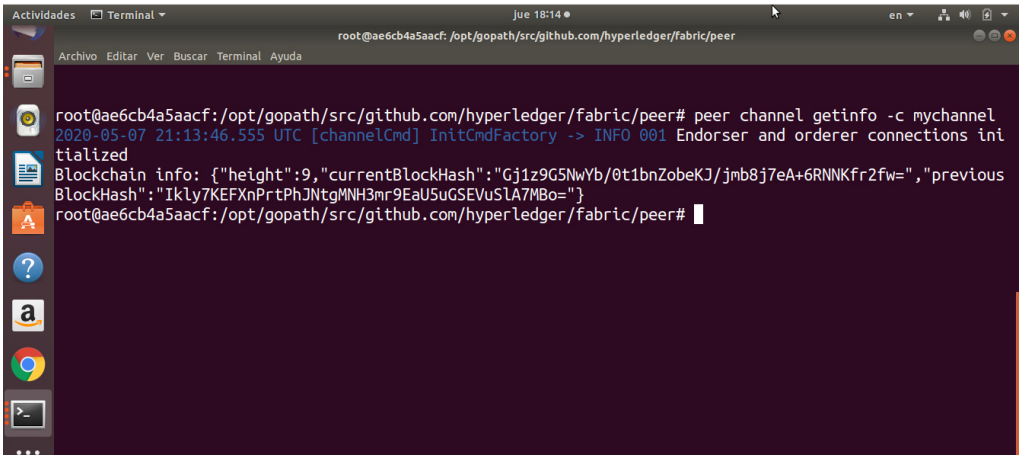

Figura 10.1.4: Recuperación de Información del último bloque

Otro ejemplo de las serias dificultades que va a tener un "intruso" en el caso de intentar adulterar la cadena de bloques se puede observar en las figuras 10.1.5 y 10.1.6. En este caso, se recuperó un bloque completo, en el formato original que usa Fabric: "*protobuf*"[51], se lo salvó en un archivo y luego se decodificó el archivo con formato *protobuf* a formato JSON. Para llevar a cabo estas operaciones, en primer lugar, se accedió a una la terminal del contenedor "cli" con el siguiente comando:

```
$:docker excec -it cli bash
```

Una vez en el contenedor, se ejecutó el comando para recuperar un bloque completo en un archivo llamado en este ejemplo "mychannel.block". El comando *"peer channel fetch"* está acompañado de un argumento *"newest",* lo cual implica que se va a recuperar el último bloque completo, y por último es necesario agregar los argumentos *"tls"* y *"cafile"* por los cuales se indica que se va a realizar una comunicación encriptada

[51] Protocol Buffer (protobuf) es la serialización de datos estructurados de Google con el mecanismo extensible de idioma neutral, plataforma neutral.

y se envía el certificado digital del *orderer* para que cuando se devuelva la respuesta, ésta pueda *"viajar"* encriptada con la clave pública del *orderer*.

```
$:peer channel fetch newest mychannel.block --channelID mychannel
--orderer orderer.example.com:7050 --tls --cafile
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererO
rganizations/example.com/orderers/orderer.example.com/msp/tlscacer
ts/tlsca.example.com-cert.pem
```

Una vez recuperado el archivo en formato *"protobuf"*, se debe ejecutar un comando que decodifique el archivo y lo convierta a un formato JSON. Esto se logra ejecutando la herramienta de Fabric "configtxlator". El comando completo es el siguiente (está indicado con una línea roja en la figura 10.1.5):

```
$:configtxlator proto_decode  - - input mychannel.block  - -type
common.Block
```

Para poder realizar todas las operaciones detalladas anteriormente el usuario debe contar con perfil de administrador de la Organización, contar con el rol de "admin" en el canal, como así también tener a la Autoridad de Certificación para obtener el certificado público para iniciar la transmisión encriptada TLS[52].

Parte del archivo decodificado que el usuario Administrador de la blockchain obtuvo se muestra en las siguientes dos figuras (10.1.5 y 10.1.6):

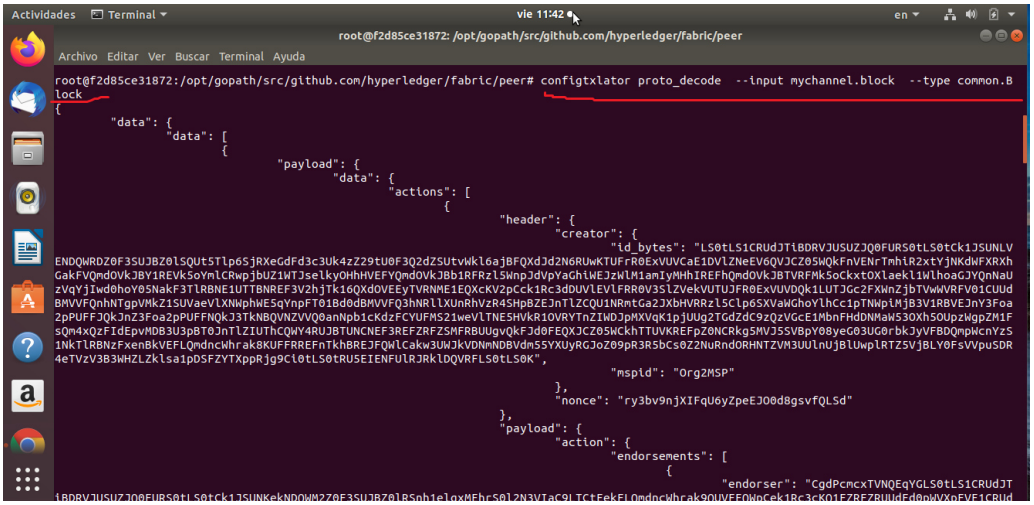

Figura 10.1.5: Decodificación y edición de un bloque completo (1)

[52] Transport Layer Security (TLS) (seguridad de la capa de transporte), es un protocolo de cifrado que se utiliza para la transmisión de datos en Internet

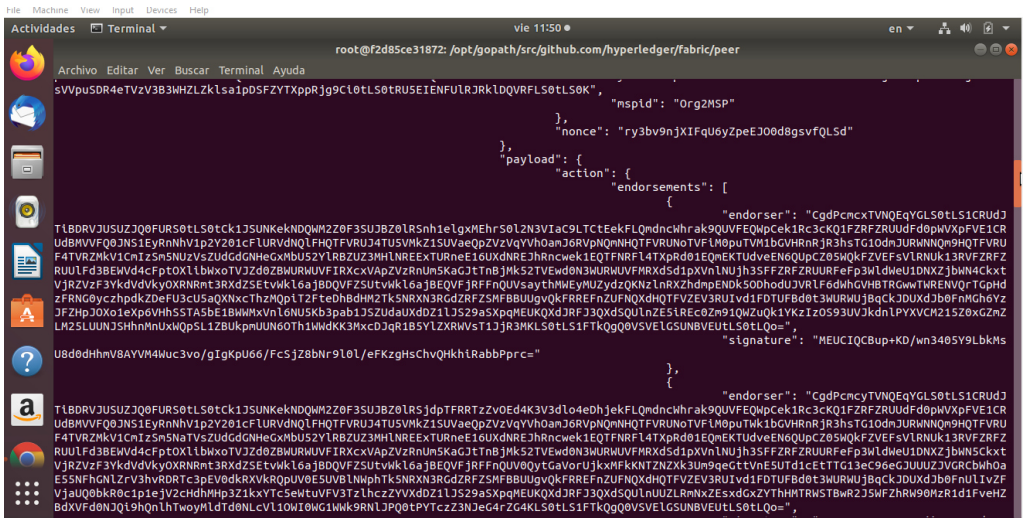

Figura 10.1.6: Decodificación y edición de un bloque completo (2)

Se comprueba que, más allá de los grandes esfuerzos para *"hackear"*[53] una blockchain (en este caso Hyperledger Fabric) no hay nada que se pueda alterar, principalmente por la esencia de su modelo de datos (punteros enlazados por funciones hash). Por otra parte, la cadena de bloques está replicada en "n" nodos conocidos como *"peer"*, por lo que, si el *"intruso"* ingresara a la red y pudiera elevar sus permisos al perfil de administrador, lo único que podría lograr es eliminar o romper la cadena en el *"peer"* al que se hubiera conectado. Para cometer el ilícito en forma integral, debería tener acceso a los "n" nodos *"peer"* que cuentan con "n" copias del libro mayor o cadena de datos y necesitará tiempo para eliminar cada una de éstas. Para cuando hubiera podido acceder al segundo o tercer nodo, la red no debería estar operativa si se configuraron correctamente las políticas de respaldo y consenso de las transacciones.

10.2. Conclusiones del capítulo

Se desarrolló una explicación detallada de la implementación del proyecto de colección de registros de eventos de una red, almacenándolos en una base de datos Mysql, para luego insertarlos en una arquitectura de cadena de bloques, y por último contar con las herramientas de consultas para verificar la integridad de la información contenida en el repositorio Mysql.

Se mostró cada una de las interfaces del sistema, las cuales permiten consultar tanto la base de datos como la blockchain. Luego se hicieron pruebas sobre la inmutabilidad de la información. Por último, se demostró la complejidad que implicaría intentar un acceso no autorizado a la red y a la cadena de bloques. Se probó también que, aun habiendo logrado el objetivo, es en extremo difícil adulterar información.

[53] Según la Real Academia Española, introducirse sin autorización a un sistema informático.

Con lo antes expuesto, se concluye que la información salvada en la red Fabric permanece inmutable (objetivo principal de la presente tesis). Se está en condiciones de detectar alteraciones en la información contenida en un registro de eventos o directamente la eliminación de los mismos cuando son salvados en repositorios centralizados. Definitivamente, es muy probable, que éstas operaciones que pretenden alterar los eventos que en realidad sucedieron, tengan como objetivo esconder actividades delictivas sobre las redes en general y los sistemas informáticos, en particular.

11. Capítulo 11: Conclusiones y Proyectos a Futuro

Blockchain es una tecnología que ha cambiado el paradigma de la protección de datos y la trazabilidad de las transacciones a través de sus características: inmutabilidad, transparencia, técnicas de cifrado criptográfico, etc.

Como se detalló a lo largo de este documento, una cadena de bloques está basada en tecnologías preexistentes de red, criptografía, distribución de procesos y mantenimiento de registros existentes. El gran aporte que hace blockchain a la trazabilidad de la información es que cambia la forma de implementar dichas tecnologías.

Se investigó sobre las características que ofrece una tecnología DLT (Distributed Ledger Technologies) en cuánto a privacidad, métodos de participación, mecanismos de consenso, comunicación con elementos externos, etc.

El presente trabajo diseñó, propuso e implementó una infraestructura para el almacenamiento inmutable de los registros de auditoría y las pruebas de integridad utilizando una cadena de bloques privada y autorizada.

Para establecer el modelo se tuvo en cuenta los requisitos legales de evidencia admisible.

La solución escogida representa una alternativa de menor costo, por lo cual en la etapa de análisis de la solución se descartaron opciones de terceros, cadenas de bloques públicas y pagas o hardware especializado de solo escritura. Es más, sin la necesidad de un proveedor de servicios externo, se asegura la inmutabilidad de la información de los registros de auditoría mediante la cooperación y el intercambio de datos entre nodos independientes.

La solución propuesta permitió el procesamiento de pruebas con fines de análisis de seguridad al tiempo que garantiza que el registro original del evento sea auditable.

El análisis de seguridad contempló configuraciones para que el sistema sea resistente a intentos de interceptar, interrumpir o modificar los datos de registro procesados.

Se implementó una prueba de concepto, por la cual se instaló y configuró Hyperledger Fabric, se desarrolló una aplicación cliente que permitió interactuar con la blockchain, principalmente, agregar y consultar por registros de eventos almacenados en el libro mayor.

En cuanto al rendimiento y perfomance del modelo propuesto, no hubo ensayos con un importante número de transacciones (no forma parte del objetivo de esta tesis). De todas formas, se investigó y comparó distintas alternativas de DLT's, lo que permitió concluir que la solución desarrollada ofrece una excelente relación costo/perfomance (puede soportar hasta 20000 transacciones por segundo).

Durante la fase de pruebas se pudo comprobar que Hyperledger Fabric es una propuesta técnica confiable y robusta.

Respecto a la evolución de la plataforma, se puede decir que es un proyecto continuo desarrollo, el cual en forma sostenida experimenta crecimiento y madurez. Entre otras características, la plataforma cuenta con una gran comunidad de usuarios *"tester"*, usuarios *"desarrolladores"* y usuarios de *"mantenimiento"*.

Respecto a la redacción de la solución, como así también la descripción detallada de la plataforma Hyperledger Fabric, se logró elaborar un documento escrito en su totalidad en español (existe muy poca bibliografía al respecto), el cual podrá ser usado como herramienta por estudiantes de grado en el sistema universitario

11.1. Trabajos a Futuro

A partir del presente trabajo se sugiere las siguientes líneas de investigación que ayudarán a la evolución de la solución propuesta:

- ✓ Desarrollar un prototipo mejorado que implemente la rotación de registros con el objetivo de bajar los costos de almacenamiento
- ✓ Investigar la implementación de sistemas Open Authorization (OAuth)[54] para la identificación de los nodos, esto favorecerá a la perfomance del sistema.
- ✓ Evaluar la posibilidad de interactuar con otra plataforma DLT utilizando el protocolo Interledger (ILP)[55]. Hyperledger lo está implementando a través de Hyperledger Quilt
- ✓ Estudiar el impacto en la perfomance de la solución cuando se incorporan logs de auditoría firmados en el originante.
- ✓ Incorporar al prototipo diseñado módulos extras de seguridad e integridad que contemplen situaciones específicas: administrar la registración de logs en la blockchain cuando las transacciones sean rechazadas, desarrollar aplicaciones cliente que en forma dinámica puedan disparar las propuestas de transacciones a distintos servicios ordenantes (Ordering Service), detectar en base a estadísticas de registración, cambios mal intencionados en la configuración de generación de logs en el originante

11.2. Conclusiones del capítulo

Las cadenas de bloques representan una estructura de datos distribuida que está evolucionado en forma permanente en respuesta a demandas de la comunidad tecnológica.

Desarrollar una solución descentralizada y colaborativa, en dónde procesos y personas puedan interactuar sin la necesidad de un sistema intermediario, es un logro

[54] OAuth es un estándar abierto que permite flujos simples de autorización para sitios web o aplicaciones informáticas

[55] ILP, es un conjunto de protocolos abiertos para enviar pagos a través de diferentes libros de contabilidad.

identificado directamente con la reducción de los costos en múltiples áreas: financiera, energética y medio ambiente, entre otros.

Generar un sistema en el cual la inteligencia del negocio sea conocida y aceptada por todos los integrantes de la solución, e implementada como un proceso distribuido al que todos invocan, es un paso hacia adelante en la desburocratización de la tecnología de los sistemas de información. Si además ofrece una estructura de almacenamiento inmutable, sin lugar a dudas, se convertirá en un pilar básico de la protección de la información generada por los sistemas digitalizados a lo largo del mundo.

Bibliografía

[1] B. D. D. G. S. D. Waters Brent R., «Building an Encrypted and Searchable Audit Log,» *Network and Distributed System Security Symposium (NDSS),* 2004.

[2] B. S. Y. Mihir Bellare, «Forward Integrity For Secure Audit Logs,» p. 16, 23 Noviembre 1997.

[3] K. j. Schneier Bruce, «Secure audit logs to support computer forensics,» *ACM Transactions on Information and System Security (TISSEC),* pp. 159-176, 2 Mayo 1999.

[4] C. D. O. S. Xu Wensheng, «A PKI-BASED SECURE AUDIT WEB SERVICE,» *IASTED Communications, Network and Information and CNIS. , Phoenix,,* 24 Noviembre 2008.

[5] J. E. Holt, «Logcrypt: Forward Security and Public Verification for Secure Audit Logs,» *Internet Security Research Lab,* 2006.

[6] K. P. M. E. Stathopoulos Vassilios, «Secure log management for privacy assurance in electronic communications,» *Computers & Security, Volume 27, Issues 7–8,* pp. 298-308, Diciembre 2008.

[7] G. A. P. B. Marsson, «Practical Secure Logging: Seekable Sequential Key Generators,» *European Symposium on Research in Computer Security,* pp. 111-128, 2013.

[8] D. J. Fehér y B. Sándor, «Log File Authentication and Storage on Blockchain Network,» *SISY 2018 • IEEE 16th International Symposium on Intelligent Systems and Informatics •,* 13-15 Seitembre 2018.

[9] S. K. J. Chuvakin Anton, The Authoritative Guide to Understanding the Concepts Surrounding Logging and Log Management, Syngress, 2012.

[10] V. S. M. Tanenbaum Andrew S., Sistemas Distribuidos. Principios y Paradigmas, México: PEARSON EDUCACIÓN, 2008.

[11] H. V. C. P. T. W. C. S.-J. Kuhn Richard D., «Introduction to Public Key Technology and the Federal PKI Infrastructure,» 2001.

[12] Accenture, «Accenture-TechVision-2019-Tech-Trends-Report.pdf,» 2019. [En línea]. Available: https://www.accenture.com/t20190304T094157Z__w__/us-en/_acnmedia/PDF-94/Accenture-TechVision-2019-Tech-Trends-Report.pdf#zoom=50.

[13] N. Satoshi, «Bitcoin: un sistema de dinero en efectivo electrónico peer-to-peer,» 2008. [En línea]. Available: https://bitcoin.org/files/bitcoin-paper/bitcoin_es.pdf .

[14] G. Good, «Ethereum: A secure decentralised generalised transaction ledger,» *Ethereum project yellow paper,* 12 4 2017.

[15] Q. N. P. D. D. B. A. M. Y. Zhang, «https://arXiv.org,» 03 Mayo 2018. [En línea]. Available: https://arXiv.org. [Último acceso: 04 Mayo 2020].

[16] D. Yaga, P. Mell, N. Roby y K. Scarfone, «Blockchain Technology Overview - Draft NISTIR 8202,» Inernal Report - NITS (National Institute of Standards and Technology), 2018.

[17] Z. Zheng, S. Xie, H. Dai, X. Chen y H. Wang, «An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends,» *IEEE 6th International Congress on Big Data,* 2017.

[18] T. T. A. Dinh, L. Rui, M. Zhang, G. Chen, B. C. Ooi y J. Wan, «Untangling Blockchain: A Data Processing View of Blockchain Systems,» *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING,* vol. 30, nº 7, Julio 2018.

[19] K. Peffers, T. Tuunamen y M. A. C. S. Rothenberger, «A Design Science Research Methodology for Information Systems Research,» *Journal of Management Information Systems,* vol. 24, pp. 45-78, December 2007.

[20] J. Onieva, J. López y J. Zhou, Secure Multi-Party Non-Repudiation Protocols and Applications, springer.com, 2009.

[21] J. C. J. C. A. Kelsey, «Signed Syslog Messages - RFC5848,» Mayo 2010. [En línea]. Available: https://tools.ietf.org/html/rfc5848.

[22] C. L. S. S. K. L. G. Gorenflo, «FastFabric: Scaling Hyperledger Fabric to 20,000 Transactions per Second,» *IEEE International Conference on Blockchain and Cryptocurrency,* 14-17 Mayo 2019.

[23] D. Ongaro y J. Ousterhout, «In Search of an Understandable Consensus Algorithm ((Extended Version)),» *Proceedings of USENIX ATC '14: 2014 USENIX Annual Technical Conference.,* 19 Junio 2014.

[24] A. A. D. López Guevara Arturo, «La criptografía y su aplicación a las tecnologías de la información,» *Coumunicaciones de Telefónica I+D ,* 2000.

[25] Z. D. H.-N. Zhend, «Blockchain Challenges and Opportunities: A survey,» *International Journal of Web and Grid Services,* Octubre 2018.

[26] G. Mohay, «Technical Challenges and Directions for Digital Forensics,» *Proceedings of the First International Workshop on Systematic Approaches to Digital Forensic Engineering,* 2005.

[27] X. Xu, I. Weber, M. Staple, J. Bos, L. Bass y C. R. P. Pautaso, «Taxonomy of Blockchain-Based Systems for Architecture Design,» *2017 IEEE International Conference on Software Architecture,* 2017.

[28] B. A. Carrion Ramiro, *Diseño e Implementación de una solución de gestion centralizada de logs de aplicaciones, sistemas y dispositivos basada en Logstash,* Barcelona, 2015.

[29] E. Conrad, S. Misenar y J. Feldman, Eleventh Hour CISSP - 3rd Edition, Syngress, 2017.

[30] Y. Kazuhiro, N. Yoshihide, Z. Ence, P. Bingfeng y S. Jun, «Potential Risks of Hyperledger Fabric Smart Contracts,» *IWBOSE 2019, Hangzhou, China, 978-1-7281-1807-9/19/$31.00 ,* 2019.

[31] J. Haris, H. Chengchen, B. Gordon y S. Xilinx, «Optimizing Validation Phase of Hyperledger Fabric,» *arXiv:1907.08367v1,* 19 Julio 2019.

[32] H. Sukhwani, *Performance Modeling & Analysis of Hyperledger Fabric (Permissioned Blockchain Network),* Pensylvania, 2018.

[33] N. Shah, Blockchain for Business with Hyperledger Fabric: A complete guide to enterprise blockchain implementation using Hyperledger Fabric, BPB Publications,, 2019, p. 326.

[34] C. Azaustre, «Aprendiendo Javascript,» Madrid, 2016.

Anexo A: Chaincode de la solución propuesta (inserción y consulta de registros de eventos)

En el presente Anexo se adjunta el chaincode que incluye dos smart contract, con los cuales se insertan y consultan los registros de eventos en la blockchain. El archivo chaincode se llama fabacar.js y el mismo invoca funciones desarrolladas en los smart contract llamados invoke.js y query.js. Las operaciones de inserción y consultas de registros en la blockchain están codificadas usando JavaScript.

Archivo "fabcar.js"

```
/*
 * SPDX-License-Identifier: Apache-2.0
 */

'use strict';

const { Contract } = require('fabric-contract-api');

class FabCar extends Contract {

    async initLedger(ctx) {

    }


    async queryAuditoria(ctx, AuditoriaId) {
        const AudiAsBytes = await ctx.stub.getState(AuditoriaId); //
get the auditoria from chaincode state
        if (!AudiAsBytes || AudiAsBytes.length === 0) {
            throw new Error(`${AuditoriaId} no existe`);
        }
        return AudiAsBytes.toString();
    }
  // id, secuencia, organizacion, ip, fecha y hora, level, facility,
prioridad, mensaje, tag, hash

    async createAuditoria(ctx, id, seqblock, orblock, ipblock,
tsblock, crblock, fablock, prblock, deblock, tablock, hashblock) {

        const auditoria = {

            id, seqblock, orblock, ipblock, tsblock, crblock,
fablock, prblock, deblock, tablock, hashblock

        };

        await ctx.stub.putState(id.toString() + orblock.toString() +
ipblock.toString() , Buffer.from(JSON.stringify(auditoria)));

    }
```

Archivo “fabcar.js” (continuación)

```
async queryAllAuditorias(ctx, startKey, endKey) {


        const iterator = await ctx.stub.getStateByRange(startKey,
endKey);


        const allResults = [];

        while (true) {

            const res = await iterator.next();


            if (res.value && res.value.value.toString()) {

                console.log(res.value.value.toString('utf8'));


                const Key = res.value.key;

                let Record;

                try {

                    Record =
JSON.parse(res.value.value.toString('utf8'));

                } catch (err) {

                    console.log(err);

                    Record = res.value.value.toString('utf8');

                }

                allResults.push({ Key, Record });

            }

            if (res.done) {

                console.log('end of data');

                await iterator.close();

                console.info(allResults);

                return JSON.stringify(allResults);

            }

        }

    }
```

Archivo “invoke.js” (continuación)

```
/*
 * SPDX-License-Identifier: Apache-2.0
 */

'use strict';

const { Gateway, FileSystemWallet } = require('fabric-network');
const fs = require('fs');
const path = require('path');
const md5 = require('md5');

const createAuditoria = async (auditoria, user, res) => {
    try {
        // load the network configuration
        const ccpPath = path.resolve(__dirname, 'config.json');
        let ccp = JSON.parse(fs.readFileSync(ccpPath, 'utf8'));

        // Create a new file system based wallet for managing
identities.
        const walletPath = path.join(process.cwd(),
'./network/wallet');
        const wallet = new FileSystemWallet(walletPath);
        console.log(`Wallet path: ${walletPath}`);

        // Check to see if we've already enrolled the user.
        const identity = await wallet.exists(user);
        if (!identity) {
            res.status(400).send({
                mensaje: `La identidad para el usuario '${user}' no
existe, por favor registrese`,
                status: false
            });
            return;
    }

    // Create a new gateway for connecting to our peer node.
        const gateway = new Gateway();
        await gateway.connect(ccp, { wallet, identity: user,
discovery: { enabled: true, asLocalhost: true } });

        // Get the network (channel) our contract is deployed to.
        const network = await gateway.getNetwork('mychannel');

        // Get the contract from the network.
        const contract = network.getContract('fabcar');

        // generamos hash con bcrypt
        const bcrypt = require('bcrypt');
        const saltRounds = 10;
```

Archivo "invoke.js" (continuación)

```
let fecha = new Date(auditoria.tsblock)

        let auditoriaHash = {
            id: auditoria.id,
            seqblock: auditoria.seqblock, orblock:
auditoria.orblock,
            ipblock: auditoria.ipblock, tsblock: fecha.toString(),
            crblock: auditoria.crblock, fablock: auditoria.fablock,
            prblock: auditoria.prblock, deblock: auditoria.deblock,
            tablock: auditoria.tablock
        }

        console.log(auditoriaHash)

        const salt = bcrypt.genSaltSync(saltRounds);
        const hash = bcrypt.hashSync(JSON.stringify(auditoriaHash),
salt);

        //id, seqblock , orblock, ipblock, tsblock, crblock,
fablock, prblock, deblock, tablock, hashblock
        await contract.submitTransaction('createAuditoria',
auditoria.id, auditoria.seqblock, auditoria.orblock,
auditoria.ipblock,
        auditoria.tsblock, auditoria.crblock, auditoria.fablock,
auditoria.prblock, auditoria.deblock,
        auditoria.tablock, hash );

        auditoria.hashblock = hash

        res.status(201).send({
            mensaje: `El registro se guardó correctamente`,
            registro: auditoria,
            status: true
        });
        // Disconnect from the gateway.
        await gateway.disconnect();

    } catch (error) {
        res.status(500).send({
            mensaje: `Error al enviar la transaccion: ${error}`,
            status: false
        });
        console.error(`Error al enviar la transaccion: ${error}`);
    }

}

module.exports = {

    createAuditoria

}
```

Archivo “query.js”

```
/*
 * SPDX-License-Identifier: Apache-2.0
 */

'use strict';

const { Gateway, FileSystemWallet } = require('fabric-network');
const path = require('path');
const fs = require('fs');

const isExistUser = async (user, res) => {

    try {
        // cargamos la configuracion de la red
        const ccpPath = path.resolve(__dirname, 'config.json');
        const ccp = JSON.parse(fs.readFileSync(ccpPath, 'utf8'));

        // creamos un nuevo archivo basado en la wallet para manejar
identidades
        const walletPath = path.join(process.cwd(),
'./network/wallet');
        const wallet = new FileSystemWallet(walletPath);
        console.log(`Wallet path: ${walletPath}`);

        // verificamos si el usuario ya esta inscripto en la wallet
        const identity = await wallet.exists(user);
        if (!identity) {
            res.status(400).send({
                mensaje: `La identidad para el usuario '${user}' no
existe, por favor registrese.`,
                status: false
            });
            return;
        }else{
            res.status(200).send({
                mensaje: `Inicio de sesion valido.`,
                status: true
            });
            return;
        }

    } catch (error) {
        res.status(400).send({
            mensaje: `Error al conectar con la red de Blockchain`,
            status: false
    });
    }

}
```

Archivo “query.js” (continuación)

```
const queryAuditoria = async ( auditoriaId, user, res ) => {
    try {
        // cargamos la configuracion de la red
        const ccpPath = path.resolve(__dirname, 'config.json');
        const ccp = JSON.parse(fs.readFileSync(ccpPath, 'utf8'));

        // creamos un nuevo archivo basado en la wallet para manejar
identidades
        const walletPath = path.join(process.cwd(),
'./network/wallet');
        const wallet = new FileSystemWallet(walletPath);
        console.log(`Wallet path: ${walletPath}`);

        // verificamos si el usuario ya esta inscripto en la wallet
        const identity = await wallet.exists(user);
        if (!identity) {
            res.status(400).send({
                mensaje: `La identidad para el usuario '${user}' no
existe, por favor registrese`,
                status: false
            });
            console.log('La identidad para el user1 no existe en la
wallet');
            return;
        }

        // creamos un nuevo gateway para conectar con nuestro peer
        const gateway = new Gateway();
        await gateway.connect(ccp, { wallet, identity: user,
discovery: { enabled: true, asLocalhost: true } });

        // Get the network (channel) our contract is deployed to.
        const network = await gateway.getNetwork('mychannel');

        // Get the contract from the network.
        const contract = network.getContract('fabcar');


    // Evaluate the specified transaction.
        const result = await
contract.evaluateTransaction('queryAuditoria', auditoriaId);

        res.status(200).send({
            mensaje: `consulta realizada con exito`,
            registro: JSON.parse(result.toString()),
            status: true
        });

  } catch (error) {
        res.status(500).send({
            mensaje: `Error al enviar la transaccion: ${error}`,
```

Archivo “query.js” (continuación)

```
status: false
        });
        console.error(`Error el la transaccion: ${error}`);
    }
}

const queryAllAuditorias = async ( user, res) => {
    try {
        // cargamos la configuracion de la red
        const ccpPath = path.resolve(__dirname, 'config.json');
        const ccp = JSON.parse(fs.readFileSync(ccpPath, 'utf8'));

        // creamos un nuevo archivo basado en la wallet para manejar
identidades
        const walletPath = path.join(process.cwd(),
'./network/wallet');
        const wallet = new FileSystemWallet(walletPath);
        console.log(`Wallet path: ${walletPath}`);

        const identity = await wallet.exists(user);
        if (!identity) {
            res.status(400).send({
                mensaje: `La identidad para el usuario '${user}' no
existe, por favor registrese`,
                status: false
            });
            console.log('La identidad para el user1 no existe en la
wallet');
            return;
        }

        const gateway = new Gateway();
        await gateway.connect(ccp, { wallet, identity: user,
discovery: { enabled: true, asLocalhost: true } });

        const network = await gateway.getNetwork('mychannel');

        const contract = network.getContract('fabcar');

        // Evaluate the specified transaction.
        const result = await
contract.evaluateTransaction('queryAllAuditorias', '0',
'9999999999999');

        res.status(200).send({
            mensaje: `consulta realizada con exito`,
            registros: JSON.parse(result.toString()),
            status: true
        });
```

Archivo “query.js” (continuación)

```
} catch (error) {
        res.status(500).send({
            mensaje: `Error al enviar la transaccion: ${error}`,
            status: false
        });
        console.error(`Error el la transaccion: ${error}`);
    }
}

module.exports = {
    queryAuditoria,
    queryAllAuditorias,
    isExistUser
}
```

Anexo B: Archivos de Configuración de la red Fabric

En el presente anexo se adjuntan todos los archivos de configuración de la red Hyperledger Fabric que fue utilizada para la implementación de la solución:

Archivo "docker-compose-ca.yaml"

```
# Archivo Configuración Autoridades de Certificado Proyecto Tesis
# crypto-eventos 2020
version: '2'
networks:
  byfn:
services:
 ca0:
    image: hyperledger/fabric-ca:$IMAGE_TAG
    environment:
      - FABRIC_CA_HOME=/etc/hyperledger/fabric-ca-server
      - FABRIC_CA_SERVER_CA_NAME=ca-org1
      - FABRIC_CA_SERVER_TLS_ENABLED=true
      - FABRIC_CA_SERVER_TLS_CERTFILE=/etc/hyperledger/fabric-ca-
            server-config/ca.org1.example.com-cert.pem
      - FABRIC_CA_SERVER_TLS_KEYFILE=/etc/hyperledger/fabric-ca-
            server
      - config/${BYFN_CA1_PRIVATE_KEY}
      - FABRIC_CA_SERVER_PORT=7054
    ports:
      - "7054:7054"
    command: sh -c 'fabric-ca-server start --ca.certfile
      /etc/hyperledger/fabric-ca-  server-
      config/ca.org1.example.com-cert.pem --ca.keyfile
      /etc/hyperledger/fabric-ca-server-
      config/${BYFN_CA1_PRIVATE_KEY} -b admin:adminpw -d'
    volumes:
      - ./crypto-
      config/peerOrganizations/org1.example.com/ca/:/etc/hyperled
      ger/fabric- ca-server-config
    container_name: ca_peerOrg1
    networks:
      - byfn
  ca1:
    image: hyperledger/fabric-ca:$IMAGE_TAG
    environment:
      - FABRIC_CA_HOME=/etc/hyperledger/fabric-ca-server
      - FABRIC_CA_SERVER_CA_NAME=ca-org2
      - FABRIC_CA_SERVER_TLS_ENABLED=true
      - FABRIC_CA_SERVER_TLS_CERTFILE=/etc/hyperledger/fabric-ca-
            server-config/ca.org2.example.com-cert.pem
      - FABRIC_CA_SERVER_TLS_KEYFILE=/etc/hyperledger/fabric-ca-
            server-config/${BYFN_CA2_PRIVATE_KEY}
      - FABRIC_CA_SERVER_PORT=8054
```

Archivo “docker-compose-ca.yaml” (continuación)

```
    ports:
      - "8054:8054"
    command: sh -c 'fabric-ca-server start --ca.certfile
      /etc/hyperledger/fabric-ca- server-config/ca.org2.example.com-
      cert.pem --ca.keyfile /etc/hyperledger/fabric-ca-server-
      config/${BYFN_CA2_PRIVATE_KEY} -b admin:adminpw -d'
    volumes:
      - ./crypto-
      config/peerOrganizations/org2.example.com/ca/:/etc/hyperledger/
      fabric-ca-server-config
    container_name: ca_peerOrg2
    networks:
      - byfn
```

Archivo “crypto-config.yaml”

```
# Archivo Configuración Certificados Digitales Proyecto Tesis crypto-
# eventos 2020
# Orderer

      - Name: Orderer
        Domain: example.com

      Specs:
        - Hostname: orderer
        - Hostname: orderer2
        - Hostname: orderer3
        - Hostname: orderer4
        - Hostname: orderer5

# "PeerOrgs" - Definition of organizations managing peer nodes

PeerOrgs:

  - Name: Org1
    Domain: org1.example.com
    EnableNodeOUs: true

    Template:
      Count: 2
    Users:
      Count: 1

  - Name: Org2
    Domain: org2.example.com
    EnableNodeOUs: true
    Template:
      Count: 2
```

Archivo “configtx.yaml”

```
# Archivo Generador del Bloque Génesis, canal y los peers del
# Proyecto Tesis crypto-eventos 2020
Section: Organizations
Organizations:
        - &OrdererOrg
            Name: OrdererOrg
            ID: OrdererMSP
            MSPDir: crypto-config/ordererOrganizations/example.com/msp
            Policies:
                  Readers:
                        Type: Signature
                        Rule: "OR('OrdererMSP.member')"
                  Writers:
                        Type: Signature
                  Rule: "OR('OrdererMSP.member')"
                  Admins:
                        Type: Signature
                        Rule: "OR('OrdererMSP.admin')"
            OrdererEndpoints:
            - orderer.example.com:7050
      - &Org1
           Name: Org1MSP
           ID: Org1MSP
           MSPDir: crypto-
config/peerOrganizations/org1.example.com/msp
            Policies:
            Readers:
                Type: Signature
                Rule: "OR('Org1MSP.admin', 'Org1MSP.peer',
                      'Org1MSP.client')"
            Writers:
                Type: Signature
                Rule: "OR('Org1MSP.admin', 'Org1MSP.client')"
            Admins:
                Type: Signature
                Rule: "OR('Org1MSP.admin')"
            Endorsement:
                Type: Signature
                Rule: "OR('Org1MSP.peer')"
        AnchorPeers:
              - Host: peer0.org1.example.com
                Port: 7051
```

Archivo “configtx.yaml” (continúa)

```
- &Org2
        Name: Org2MSP
        ID: Org2MSP
        MSPDir: crypto-config/peerOrganizations/org2.example.com/msp
        Policies:
            Readers:
                Type: Signature
                Rule: "OR('Org2MSP.admin', 'Org2MSP.peer',
                      'Org2MSP.client')"
            Writers:
                Type: Signature
                Rule: "OR('Org2MSP.admin', 'Org2MSP.client')"

            Admin:
                Type: Signature
                Rule: "OR('Org2MSP.peer')"
        AnchorPeers:
            - Host: peer0.org2.example.com
              Port: 9051
```

Archivo “docker-compose-cli.yaml”

```
# Archivo compose-cli Proyecto Tesis crypto-eventos 2020
version: '2'
volumes:
  orderer.example.com:
  peer0.org1.example.com:
  peer1.org1.example.com:
  peer0.org2.example.com:
  peer1.org2.example.com:
networks:
  byfn:

services:
  orderer.example.com:
    extends:
      file:   base/docker-compose-base.yaml
      service: orderer.example.com
    container_name: orderer.example.com
    networks:
      - byfn
peer0.org1.example.com:
    container_name: peer0.org1.example.com
    extends:
      file:  base/docker-compose-base.yaml
      service: peer0.org1.example.com
    networks:
      - byfn
```

Archivo “docker-compose-cli.yaml” (continuación)

```
peer1.org1.example.com:
    container_name: peer1.org1.example.com
    extends:
      file:  base/docker-compose-base.yaml
      service: peer1.org1.example.com
    networks:
      - byfn

  peer0.org2.example.com:
    container_name: peer0.org2.example.com
    extends:
      file:  base/docker-compose-base.yaml
      service: peer0.org2.example.com
    networks:
      - byfn

  peer1.org2.example.com:
    container_name: peer1.org2.example.com
    extends:
      file:  base/docker-compose-base.yaml
      service: peer1.org2.example.com
    networks:
      - byfn

  cli:
    container_name: cli
    image: hyperledger/fabric-tools:$IMAGE_TAG
    tty: true
    stdin_open: true
    environment:

      - SYS_CHANNEL=$SYS_CHANNEL
      - GOPATH=/opt/gopath
      - CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock
      #- FABRIC_LOGGING_SPEC=DEBUG
      - FABRIC_LOGGING_SPEC=INFO
      - CORE_PEER_ID=cli
      - CORE_PEER_ADDRESS=peer0.org1.example.com:7051
      - CORE_PEER_LOCALMSPID=Org1MSP
      - CORE_PEER_TLS_ENABLED=true
      -
CORE_PEER_TLS_CERT_FILE=/opt/gopath/src/github.com/hyperledger/fabri
c/peer/crypto/peerOrganizations/org1.example.com/peers/peer0.org1.ex
ample.com/tls/server.crt

CORE_PEER_TLS_KEY_FILE=/opt/gopath/src/github.com/hyperledger/fabric
/peer/crypto/peerOrganizations/org1.example.com/peers/peer0.org1.exa
mple.com/tls/server.key
      -
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/f
abric/peer/crypto/peerOrganizations/org1.example.com/peers/peer0.org
1.example.com/tls/ca.crt
```

Archivo "docker-compose-cli.yaml" (continuación)

```
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabr
ic/peer/crypto/peerOrganizations/org1.example.com/users/Admin@org1.
example.com/msp
    working_dir: /opt/gopath/src/github.com/hyperledger/fabric/peer
    command: /bin/bash
    volumes:
        - /var/run/:/host/var/run/
        - ./../chaincode/:/opt/gopath/src/github.com/chaincode
        - ./crypto-
config:/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/
        -
./scripts:/opt/gopath/src/github.com/hyperledger/fabric/peer/script
s/
        - ./channel-
artifacts:/opt/gopath/src/github.com/hyperledger/fabric/peer/channe
l-artifacts
    depends_on:
      - orderer.example.com
      - peer0.org1.example.com
      - peer1.org1.example.com
      - peer0.org2.example.com
      - peer1.org2.example.com
    networks:
      - byfn
```

Archivo "docker-compose-couchdb.yaml

```
version: '2'

networks:
  byfn:

services:
  couchdb0:
    container_name: couchdb0
    image: hyperledger/fabric-couchdb

    environment:
      - COUCHDB_USER=
      - COUCHDB_PASSWORD=

    ports:
      - "5984:5984"
    networks:
      - byfn
```

Archivo “docker-compose-couchdb.yaml (continuación)

```
peer0.org1.example.com:
    environment:
      - CORE_LEDGER_STATE_STATEDATABASE=CouchDB
      - CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb0:5984
      - CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME=
      - CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD=
    depends_on:
      - couchdb0

  couchdb1:
    container_name: couchdb1
    image: hyperledger/fabric-couchdb

    environment:
      - COUCHDB_USER=
      - COUCHDB_PASSWORD=

    ports:
      - "6984:5984"
    networks:
      - byfn

  peer1.org1.example.com:
    environment:
      - CORE_LEDGER_STATE_STATEDATABASE=CouchDB
      - CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb1:5984
      - CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME=
      - CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD=
    depends_on:
      - couchdb1

  couchdb2:
    container_name: couchdb2
    image: hyperledger/fabric-couchdb

    environment:
      - COUCHDB_USER=
      - COUCHDB_PASSWORD=

    ports:
      - "7984:5984"
    networks:
      - byfn

peer0.org2.example.com:
    environment:
      - CORE_LEDGER_STATE_STATEDATABASE=CouchDB
      - CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb2:5984
      - CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME=
      - CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD=
depends_on:
      - couchdb2
```

Archivo “docker-compose-couchdb.yaml (continuación)

```
couchdb3:
    container_name: couchdb3
    image: hyperledger/fabric-couchdb

    environment:
      - COUCHDB_USER=
      - COUCHDB_PASSWORD=

    ports:
      - "8984:5984"
    networks:
      - byfn

  peer1.org2.example.com:
    environment:
      - CORE_LEDGER_STATE_STATEDATABASE=CouchDB
      - CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb3:5984
      - CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME=
      - CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD=
    depends_on:
      - couchdb3
```

Archivo “starfabric.sh”

```
#!/bin/bash
# Copyright IBM Corp All Rights Reserved
# SPDX-License-Identifier: Apache-2.0
# Exit on first error
set -e

# don't rewrite paths for Windows Git Bash users
export MSYS_NO_PATHCONV=1
starttime=$(date +%s)
CC_SRC_LANGUAGE=${1:-"go"}
CC_SRC_LANGUAGE=`echo "$CC_SRC_LANGUAGE" | tr [:upper:] [:lower:]`
if [ "$CC_SRC_LANGUAGE" = "go" -o "$CC_SRC_LANGUAGE" = "golang"  ];
then
CC_RUNTIME_LANGUAGE=golang
CC_SRC_PATH=github.com/chaincode/fabcar/go
elif [ "$CC_SRC_LANGUAGE" = "java" ]; then
CC_RUNTIME_LANGUAGE=java
CC_SRC_PATH=/opt/gopath/src/github.com/chaincode/fabcar/java
elif [ "$CC_SRC_LANGUAGE" = "javascript" ]; then
CC_RUNTIME_LANGUAGE=node # chaincode runtime language is   node.js
CC_SRC_PATH=/opt/gopath/src/github.com/chaincode/fabcar/javasc
      ript
```

Archivo "starfabric.sh" (continuación)

```
elif [ "$CC_SRC_LANGUAGE" = "typescript" ]; then
      CC_RUNTIME_LANGUAGE=node # chaincode runtime language is
      node.js
      CC_SRC_PATH=/opt/gopath/src/github.com/chaincode/fabcar/typesc
      ript
      echo Compiling TypeScript code into JavaScript ...
      pushd ../chaincode/fabcar/typescript
      npm install
      npm run build
      popd
      echo Finished compiling TypeScript code into JavaScript
else
      echo The chaincode language ${CC_SRC_LANGUAGE} is not
      supported by this script
      echo Supported chaincode languages are: go, javascript, and
      typescript
      exit 1
fi

# clean the keystore
rm -rf ./hfc-key-store
# launch network; create channel and join peer to channel
cd ../first-network
echo y | ./byfn.sh down
echo y | ./byfn.sh up -a -n -s couchdb

CONFIG_ROOT=/opt/gopath/src/github.com/hyperledger/fabric/peer
ORG1_MSPCONFIGPATH=${CONFIG_ROOT}/crypto/peerOrganizations/org1.exa
mple.com/users/Admin@org1.example.com/msp
ORG1_TLS_ROOTCERT_FILE=${CONFIG_ROOT}/crypto/peerOrganizations/org1
.example.com/peers/peer0.org1.example.com/tls/ca.crt
ORG2_MSPCONFIGPATH=${CONFIG_ROOT}/crypto/peerOrganizations/org2.exa
mple.com/users/Admin@org2.example.com/msp
ORG2_TLS_ROOTCERT_FILE=${CONFIG_ROOT}/crypto/peerOrganizations/org2
.example.com/peers/peer0.org2.example.com/tls/ca.crt
ORDERER_TLS_ROOTCERT_FILE=${CONFIG_ROOT}/crypto/ordererOrganization
s/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.exa
mple.com-cert.pem
set -x
echo "Installing smart contract on peer0.org1.example.com"
docker exec \
  -e CORE_PEER_LOCALMSPID=Org1MSP \
  -e CORE_PEER_ADDRESS=peer0.org1.example.com:7051 \
  -e CORE_PEER_MSPCONFIGPATH=${ORG1_MSPCONFIGPATH} \
  -e CORE_PEER_TLS_ROOTCERT_FILE=${ORG1_TLS_ROOTCERT_FILE} \
  cli \
  peer chaincode install \
    -n fabcar \
    -v 1.0 \
    -p "$CC_SRC_PATH" \
    -l "$CC_RUNTIME_LANGUAGE"
```

Archivo “starfabric.sh” (continuación)

```
echo "Installing smart contract on peer0.org2.example.com"
docker exec \
  -e CORE_PEER_LOCALMSPID=Org2MSP \
  -e CORE_PEER_ADDRESS=peer0.org2.example.com:9051 \
  -e CORE_PEER_MSPCONFIGPATH=${ORG2_MSPCONFIGPATH} \
  -e CORE_PEER_TLS_ROOTCERT_FILE=${ORG2_TLS_ROOTCERT_FILE} \
  cli \
  peer chaincode install \
    -n fabcar \
    -v 1.0 \
    -p "$CC_SRC_PATH" \
    -l "$CC_RUNTIME_LANGUAGE"

echo "Instantiating smart contract on mychannel"
docker exec \
  -e CORE_PEER_LOCALMSPID=Org1MSP \
  -e CORE_PEER_MSPCONFIGPATH=${ORG1_MSPCONFIGPATH} \
  cli \
  peer chaincode instantiate \
    -o orderer.example.com:7050 \
    -C mychannel \
    -n fabcar \
    -l "$CC_RUNTIME_LANGUAGE" \
    -v 1.0 \
    -c '{"Args":[]}' \
    -P "AND('Org1MSP.member','Org2MSP.member')" \
    --tls \
    --cafile ${ORDERER_TLS_ROOTCERT_FILE} \
    --peerAddresses peer0.org1.example.com:7051 \
    --tlsRootCertFiles ${ORG1_TLS_ROOTCERT_FILE}

echo "Waiting for instantiation request to be committed ..."
sleep 10

echo "Submitting initLedger transaction to smart contract on
mychannel"
echo "The transaction is sent to the two peers with the chaincode
installed (peer0.org1.example.com and peer0.org2.example.com) so
that chaincode is built before receiving the following requests"
```

Archivo “starfabric.sh” (continuación)

```
docker exec \
  -e CORE_PEER_LOCALMSPID=Org1MSP \
  -e CORE_PEER_MSPCONFIGPATH=${ORG1_MSPCONFIGPATH} \
  cli \
  peer chaincode invoke \
    -o orderer.example.com:7050 \
    -C mychannel \
    -n fabcar \
    -c '{"function":"initLedger","Args":[]}' \
    --waitForEvent \
    --tls \
    --cafile ${ORDERER_TLS_ROOTCERT_FILE} \
    --peerAddresses peer0.org1.example.com:7051 \
    --peerAddresses peer0.org2.example.com:9051 \
    --tlsRootCertFiles ${ORG1_TLS_ROOTCERT_FILE} \
    --tlsRootCertFiles ${ORG2_TLS_ROOTCERT_FILE}
set +x
```

Anexo C: Herramientas y Protocolos utilizados en la solución propuesta

Protocolo syslog-ng

Por lo general, syslog-ng se usa para administrar mensajes de eventos e implementar el registro centralizado de los mismos. Los diferentes dispositivos ejecutan una versión cliente del protocolo syslog-ng y recopilan los registros de auditoría que genera esa fuente, en este caso, la electrónica de red. Luego toda la información, es encapsulada y enviada en un formato de mensaje específico a un repositorio central, el cual está ejecutando la versión servidor del protocolo syslog-ng.

¿Por qué syslog-ng?

Syslog-ng es la evolución de syslog. Ofrece funcionalidades que ayudan a mejorar la administración de los registros de eventos. Tiene la posibilidad de filtrar en función del contenido del mensaje utilizando expresiones regulares.

La aplicación syslog-ng puede comparar el contenido de los mensajes de registro con una base de datos de patrones de mensajes predefinidos, por lo que syslog-ng puede identificar el tipo exacto de los mensajes y clasificarlos en clases de mensajes.

Las versiones recientes de syslog-ng también hacen posible la correlación de eventos en tiempo real. Esto puede ser útil en muchas situaciones diferentes. Por ejemplo, los datos importantes para un solo evento a menudo se dispersan en múltiples mensajes de syslog. Además, los eventos de inicio y cierre de sesión a menudo se registran lejos el uno del otro, incluso en diferentes archivos de registro, lo que dificulta el análisis. Mediante la correlación, estos se pueden recopilar en un único mensaje nuevo.

Estructura del archivo de configuración de syslog-ng

El archivo de configuración del protocolo syslog es:

/etc/syslog-ng/syslog-ng.conf

La estructura de dicho archivo se muestra en la siguiente figura C.1:

```
source s_src {
            system();
            internal();
            udp();
};
filter f_routers { facility(local0); };
destination routers {
      file("/var/log/network/$YEAR/$MONTH/$DAY/$HOST-$YEAR-
            $MONTH-$DAY-$HOUR.log"
      owner(root) group(root) perm(0644) dir_perm(0755)
      create_dirs(yes)
      template("$YEAR $DATE $HOST $MSG\n"));
};

log {
            source(s_src);
            filter(f_routers);
            destination(routers);
};
```

Figura C.1: Estructura del archivo de configuración de syslog-ng

Se puede observar en la primera sección la configuración del origen de los mensajes con registros de eventos que serán capturados (puede ser el sistema operativo del servidor o cualquier dispositivo de una red). En una segunda sección podemos observar sentencias de filtrado para esos mensajes; en este caso filtra los registros que cumplen con valor "personalizado" por el administrador (*Facility(local 10*)). Por último, en la tercera sección se puede apreciar la configuración correspondiente al destino o lugar dónde se almacenarán los registros capturados (en este caso archivos de texto que se generan por hora, identificando en el nombre de archivo el dispositivo que los origina). Al final del archivo de configuración, se ejecuta la sentencia "log" con los parámetros origen, filtro y destino, por lo que se inicia la captura de registros de eventos utilizando los argumentos invocados.

Docker

Docker es un proyecto de código abierto que automatiza el despliegue de aplicaciones, utilizando imágenes que instancian contenedores de software, proporcionando una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos.

Las imágenes se crean con el comando *"docker build"*. Una vez creada una imagen, se pueden almacenar en el Hub Docker. Se pueden utilizar millones de imágenes

almacenadas. Cuando una imagen es invocada con el comando "docker run" se genera un contenedor.

Los contenedores le permiten ejecutar sus aplicaciones en procesos aislados de recursos. Se parecen a las máquinas virtuales, sin embargo, los contenedores son más portátiles, tienen más recursos y son más dependientes del sistema operativo host.

En la siguiente figura C.2 se puede apreciar dos cosas: en primer lugar, en la parte superior de la misma, la diferencia conceptual entre virtualización tradicional y virtualización usando contenedores, luego en segundo lugar, en la parte inferior, la relación entre la arquitectura Fabric (peer's, orderer's, CA's, y otras) y los contenedores instanciados en docker.

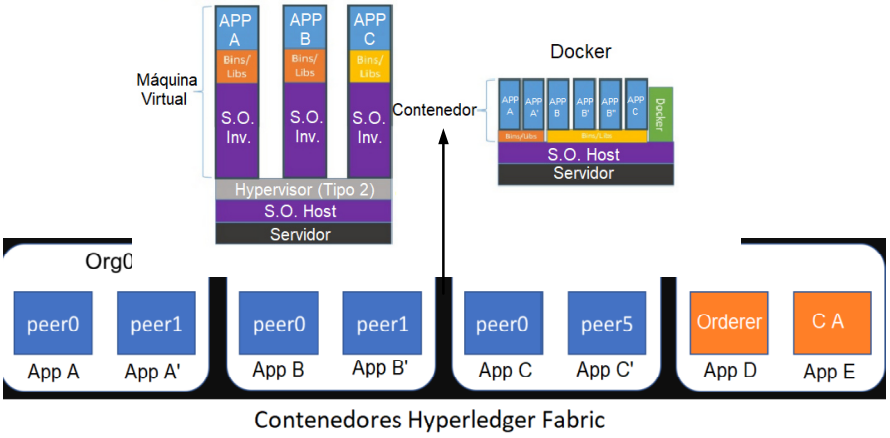

Figura C.2: Hyperledger Fabric y Docker

Docker Compose

Compose es una herramienta para definir y ejecutar aplicaciones Docker de múltiples contenedores. Este ejecutable se usa junto a un archivo Compose para configurar los servicios de su aplicación. Luego, utilizando un solo comando, creará e iniciará todos los servicios desde su configuración.

Para usar Compose es necesario tener en cuenta lo siguiente:

- ✓ Definir el entorno de la aplicación con un Dockerfile para que pueda reproducirse en cualquier lugar.
- ✓ Definir los servicios que conforman la aplicación en docker-compose.yml para que puedan ejecutarse juntos en un entorno aislado.
- ✓ Ejecutar *"docker-compose up"* y Compose se iniciará y ejecutará toda la aplicación.

En la siguiente figura C.3 se puede observar cómo los eventos de cada una de las organizaciones se dirigen al servidor central, el cual está ejecutando la aplicación *"cliente"* de Fabric. Esta última aplicación es la interface con la blockchain, por lo que tiene capacidad para conectarse a los múltiples contenedores que fueron generados con la herramienta docker compose: peer0.org1, peer0.org2, orderer1, orderer2 y FabricCA.

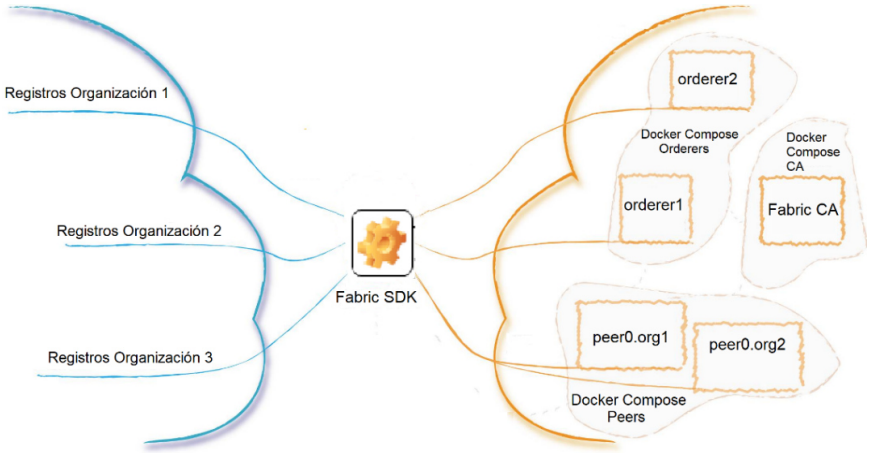

Figura C.3: Contenedores Hyperledger Fabric instanciados con docker-compose