Facultad de Informática

Tesis de Maestría

Título

Certificados Digitales: de una arquitectura jerárquica y centralizada a una distribuida y descentralizada

Carrera

Maestría en Redes de Datos

Tesista

Ing. Ignacio Martín Gallardo Urbini

Directora

Dra. Patricia Bazán

Asesora Científica

Lic. Paula Venosa

La Plata, Argentina. Año 2018

Índice

Presentación de Tesis

Resumen

El auge de la teleinformática en sus formas más conocidas: la informática y las telecomunicaciones, impusieron el tráfico y circulación veloz de documentos notariales, civiles, comerciales y de cualquier otra naturaleza, sumado a ello, el cambio de hábitos laborales que imponen el trabajo domiciliario en varios días de la semana.

La evolución tecnológica de los últimos años en el campo electrónico y digital, ha transformado la industria, el comercio, el sector servicios y doméstico, dando existencia cada vez más a una demanda mayor de las transacciones ante la necesidad de interactuar por intermedio de redes de computadoras.

El término "firma digital" nace de una oferta tecnológica para acercar la firma manuscrita (ológrafa) a lo que se llama el trabajo en redes o ciberespacio que garantiza los trámites hechos en Internet. Este concepto de firma digital fue introducido por Diffie y Hellman en 1976 básicamente como una aplicación tecnológica de la criptografía asimétrica planteada como un conjunto de datos asociados a un mensaje que permite asegurar la identidad del firmante y la integridad del mensaje.

La criptografía de clave pública ha sido ampliamente reconocida como una tecnología fundamental sobre la cual pueden construirse varios servicios básicos de seguridad, dado así, en 1988 comenzando conjuntamente con el estándar x.500 se adopta por parte las industrias el estándar x.509 que consiste en la aplicación de los conceptos anteriores implementados con el nombre de infraestructura de clave pública (PKI, Public Key Infrastructure), donde la objeción principal encuentra asidero respaldatorio en entidades/empresas certificadoras privadas y/o organismos públicos licenciados y respaldados por la ley 25.506 y normas complementarias [73] que impiden el delito de falsificación de los certificados digitales. Así la misma, queda equiparada a la firma de puño y letra.

Los principales esfuerzos de los últimos años se han concentrado en el problema de asignar de forma segura nombres a claves, de hecho, la comunidad científica ha ido progresivamente adoptando el uso de sistemas basados en la arquitectura de certificados digitales con el fin de proporcionar servicios de seguridad a los sistemas distribuidos, los cuales dependen de la existencia de un método centralizado y jerárquico para converger en la seguridad y fiabilidad en cuanto a gestión de claves públicas.

Toda esta misma problemática se traspola a la gestión del comercio electrónico confiable y seguro, donde en el modelo inicial se requería de una tercera entidad (un banco) la cual debía emitir divisa electrónica a los diferentes usuarios.

A principios del año 2009, Satoshi Nakamoto crea Bitcoin: una tecnología peer-to-peer para operar sin una autoridad central o bancos; la gestión de las transacciones y la emisión de moneda electrónica es llevada a cabo de forma colectiva por la red. Este modelo innovador combina varios conceptos desarrollados por las propuestas iniciales de la arquitectura de funcionamiento del dinero electrónico, para lograr un sistema completamente descentralizado.

A diferencia de la mayoría de las monedas, el funcionamiento de Bitcoin no depende de una institución central, sino de una base de datos distribuida y sincronizada llamada *blockchain* donde las transacciones se identifican y ordenan secuencialmente e impidiendo su modificación. El software ideado por Nakamoto emplea la criptografía asimétrica para proveer funciones de seguridad básicas, tales como la garantía de que los bitcoins solo puedan ser gastados por su dueño, y nunca más de una vez.

El objetivo de esta tesis es llevar a cabo y aplicar a la arquitectura de certificados digitales o infraestructura de clave pública lo que allá en el 2009 logró exitosamente Satoshi Nakamoto, un rediseño, reingeniería y cambio de paradigma en el comercio electrónico, de una arquitectura centralizada y jerárquica a una completamente descentralizada por medio de su innovación tecnológica llamada *blockchain* o cadena de bloques.

Introducción

Una forma de entender qué es una ***blockchain*** es antes comprender qué es una **criptomoneda** con sus diferencias respecto del dinero electrónico, y para comprender fácilmente estos conceptos, primero hay que definir qué es una moneda fiduciaria.
Moneda fiduciaria es toda moneda de curso legal designada y emitida por una **autoridad central** que las personas están dispuestas a aceptar a cambio de productos o servicios porque está respaldada por la regulación vigente y por la **confianza** en dicha autoridad central.
El dinero fiduciario es similar al dinero respaldado por los productos básicos en apariencia y uso pero no puede ser canjeado por uno de esos productos, como por ejemplo la plata y/o el oro.

Por el contrario, una **moneda virtual** es un tipo de **dinero digital** no regulado que emiten y controlan sus creadores, y que se utilizan y aceptan entre miembros de la comunidad virtual[41].

La criptodivisa **Bitcoin**[42] surgió en el año 2009 como una alternativa a la moneda fiduciaria, con la diferencia que éstos no se emiten como este último, sino que se "extraen" mediante un procedimiento denominado "minería de bitcoins" utilizando la capacidad de una inmensa red de cómputo conectada distribuidamente a través de todo el mundo. Esta tecnología surge con el objetivo de **descentralizar** los pagos entre usuarios, eliminando la necesidad de la **presencia de instituciones** financieras en las transacciones. Para llevar a cabo estos requerimientos, bitcoin se despliega en una **red P2P**(Peer to Peer) por la cual se mantienen, distribuyen y coexisten todas las transacciones asegurando la **no alteración** de las mismas sin tener que realizar operaciones demasiadas exigentes desde el punto de vista computacional para converger todo el sistema.

Esencialmente, no solo Bitcoin sino las demás criptomonedas como Ethereum[43], Litecoin[44], Ripple[45], etc. son nada más ni menos que un archivo digital donde se enumeran todas las transacciones de la red al mejor estilo "libro de contabilidad", con la peculiaridad que este mismo, se encuentra presente en todos los participantes del juego y no puede ser alterado.

Aparte del formato digital, son mínimas las similitudes entre el dinero electrónico y las criptomonedas. El dinero electrónico, como muchos otros formatos digitales de la moneda fiduciaria —como las tarjetas de crédito y débito, PayPal y las transferencias electrónicas—, es simplemente un mecanismo mediante el cual se interactúa con esa moneda fiduciaria. Para mitigar riesgos sistémicos y de protección del consumidor, el efectivo que respalda el dinero electrónico emitido habitualmente se deposita en instituciones financieras que siguen todas las regulaciones prudenciales. A diferencia de la moneda criptográfica, el dinero electrónico no es una moneda individual y está supervisado por la misma autoridad central que controla la moneda nacional que lo respalda.

Este sistema inicialmente aplicado a las criptomonedas es el primer ejemplo de una creciente revolución teleinformática, potencialmente aplicable a infinitas áreas de conocimiento, en la que mediante software de código abierto se resuelven **sincronizadamente** cálculos matemáticos[62] para **validar todas las operaciones** realizadas por cada individuo perteneciente a la red sin necesidad de ser comandados y regulados por un **ente central**, y no obstante, **manteniendo la integridad**, **máxima disponibilidad** del historial de las transacciones, y **desconcentrando** no solo la confianza en esta única autoridad central, sino también la información y los procesos, lo cual proporciona una ventaja a la hora de la existencia de una violación a la seguridad[63].

De todo este procedimiento descripto surge el término **cadena de bloques** o ***blockchain*** — mecanismo utilizado por Bitcoin— ya que todas las operaciones de la red se acumulan en bloques de transacciones y estos mismos se van adjuntando entre sí formando una cadena y aplicando los conceptos de un **árbol de Merkle**[1], no obstante, existe una réplica irrevocable

[1] Estructura de datos en árbol, binario o no, en el que cada nodo que no es una hoja está etiquetado con el hash de las de la concatenación de las etiquetas o valores (para nodos hoja) de sus nodos hijo. Son una generalización de las listas hash y las cadenas hash.

de este árbol —archivo digital— en todos los integrantes del sistema mantenido a la orden del día.
Resulta llamativo pensar que una tecnología implementada hace menos de nueve años y desarrollada por una persona desconocida podría pasar a ser más eficiente, confiable y popular que ya varias décadas de la vigencia del dinero electrónico, pero la realidad es que toda su arquitectura y diseño contempla hasta el último detalle a la hora de pensar en confiabilidad, robustez y seguridad.
El objetivo general de esta tesis consiste en analizar y seleccionar los conceptos de descentralización, distribución, validación masiva y comunitaria, sincronismo, integridad, transparencia, escalabilidad, redundancia y confiabilidad presentes en ***blockchain*** o **cadena de bloques,** para poder extraerlos y aplicarlos en una propuesta de rediseño y mejora a otro ámbito que ya lleva bastante tiempo de estudio desde el punto de vista no solo de la seguridad sino también de la ingeniería de software llamado: Arquitectura de **Certificados Digitales** o **Infraestructura de clave pública**[64]. Estas ventajas serán aprovechadas volcando de forma teórica en la modificación del diseño y arquitectura existente en los **certificados digitales**.
Esta tesis estará conformada por siete secciones, en donde se comenzará en la primer sección describiendo conceptos esenciales de la Seguridad en Teleinformática los cuales servirán de base para comprender las siguientes temáticas. La segunda sección será enriquecida con conceptos criptográficos esenciales para entender los principios de funcionamiento de ***blockchain*** —columna vertebral de esta tesis—, de **criptodivisas** en general y de la arquitectura de **certificados digitales**. En la tercer parte se ampliará sobre la arquitectura de sistemas distribuidos. En la cuarta sección se abordará una introducción a los conceptos de la arquitectura de certificados digitales con los estándares que ésta acarrea. Una detallada descripción de ***blockchain*** —mecanismo empleado por las criptomonedas para asegurar sus operaciones— se adueña del quinto capítulo para luego, junto a todos estos conocimientos presentar la **propuesta** y detallarla en un sexto. Finalmente, en la séptima sección se presentarán las conclusiones y trabajo a futuro.

Capítulo 1 - Seguridad en Teleinformática

Introducción Histórica

Con el fin de combatir los delitos y movimientos laborales tan comunes, ya a principios del siglo XX la seguridad comienza a identificarse como una de las funciones principales de las organizaciones, promovida en 1919 por el teórico y pionero de la administración Henry Fayol, luego de la técnica comercial, financiera, contable y directiva.
Al definir el objetivo de la seguridad, Fayol se refirió a: *"salvaguardar activos contra el robo, fuego, inundación, contrarrestar huelgas y traiciones por parte del personal, y de forma amplia todos los disturbios sociales que puedan poner en peligro el progreso e incluso la vida del negocio."*
Las medidas de seguridad a las que se refería Fayol, nó sólo se restringía a los exclusivamente materiales físicos de la instalación, sino también al personal perteneciente a la infraestructura de las organizaciones.
Los requerimientos de seguridad en las organizaciones han sufrido dos cambios importantes en las últimas décadas. El primero surge con la introducción de las computadoras, ya que la

necesidad de herramientas automatizadas para la protección de archivos y otra información almacenada se fue haciendo evidente. El segundo cambio surge con la aparición de los sistemas distribuidos, así como el uso de redes e instalaciones de comunicaciones para enviar información entre un servidor y una computadora o entre dos computadoras. Aquí la seguridad deja de tratarse desde el punto de vista computacional para abarcar también los aspectos de las telecomunicaciones.
En épocas contemporáneas del desarrollo tecnológico puede expresar que la informática y las comunicaciones se encuentran en un grado tan alto de integración que es muy difícil determinar con exactitud cuál es la frontera entre estas disciplinas.
Las tecnologías usadas para satisfacer los problemas de comunicaciones y los de informática son exactamente las mismas, donde éstas cada vez tienen mayor capacidad no solo de cómputo sino también de cambio. No únicamente la cantidad heterogénea de dispositivos que integran las infraestructuras de acceso teleinformático se ha multiplicado astronómicamente, sino también el tiempo en el que estos mismos se encuentran conectados y transfiriendo grandes volúmenes de información entre sí. No obstante, para que estas interacciones pudieran mantener los requerimientos legitimidad ante la presencia de severas amenazas, la ingeniería en seguridad también tuvo que dar un gran giro.

Definición de Seguridad en Teleinformática

La seguridad en Teleinformática se define como el área de conocimiento relacionada con la informática y la telemática que se enfoca en la protección de la infraestructura computacional y todo lo relacionado con esta y, especialmente, la información contenida en un sistema informático o circulante a través de las redes de computadoras[46].
El objetivo primordial de la seguridad en teleinformática prevalece en el establecimiento de normas que disipen potenciales riesgos y eviten posibles amenazas a la información o infraestructura informática.
Puesto que el propósito de los sistemas teleinformáticos no es otro que almacenar, procesar y transmitir información, consideramos entonces que un sistema teleinformático es seguro si maneja de forma correcta la información. Cabría entonces preguntarse cuáles son las condiciones que debe cumplir la información para que podamos determinar su calidad y para, en un paso posterior, determinar qué se podría hacer para garantizarla.
Los sistemas teleinformáticos incorporan medidas para garantizar su seguridad prácticamente a todos los niveles del sistema operativo, y abarcan todos los conceptos importantes que se verán en esta sección.

Problemas de los Sistemas Teleinformáticos

Cualquier situación en la vida está sujeta a la ocurrencia de situaciones no deseadas. En particular todos los sistemas informáticos están sujetos a la posibilidad de experimentar un funcionamiento anómalo, ya sea de manera accidental o provocada.

Para identificar estas situaciones, se definirán ciertos términos a modo de proporcionar facilidad en el entendimiento de dichos fallos en el funcionamiento deseado:

- Daño: perjuicio que se produce cuando un sistema informático falla.
- Ataque: acto deliberado de intentar provocar un daño.
- Riesgo: producto entre la magnitud de un daño y la probabilidad de ocurrencia del mismo.
- Amenaza: situación de daño cuyo riesgo de producirse es significativo.
- Vulnerabilidad: deficiencia de un sistema totalmente susceptible de producir un fallo en el mismo.
- Exploit: técnica que permite el aprovechamiento de una vulnerabilidad.

Amenazas a la Seguridad

No sólo las amenazas que surgen de la programación y el funcionamiento de un dispositivo de almacenamiento, transmisión o procesamiento deben ser consideradas, también hay otras circunstancias no informáticas que deben ser tomadas en cuenta. Muchas son a menudo imprevisibles o inevitables y estas pueden ser causadas por:

- Usuarios: causa del mayor problema ligado a la seguridad de un sistema informático. En algunos casos sus acciones causan problemas de seguridad, si bien en la mayoría de los casos es porque tienen permisos sobredimensionados, no se les han restringido acciones innecesarias, etc.
- Programas maliciosos: programas destinados a perjudicar o a hacer un uso ilícito de los recursos del sistema. Es instalado en el ordenador, abriendo una puerta a intrusos o bien modificando los datos. Estos programas pueden ser un virus informático, un gusano informático, un troyano, una bomba lógica, un programa espía o spyware, en general conocidos como malware.
- Errores de programación: la mayoría de los errores de programación que se pueden considerar como una amenaza informática es por su condición de poder ser usados como exploits por los crackers, aunque se dan casos donde el mal desarrollo es, en sí mismo, una amenaza. La actualización de parches de los sistemas operativos y aplicaciones permite evitar este tipo de amenazas.
- Intrusos: personas que consiguen acceder a los datos o programas a los cuales no están autorizados (crackers, defacers, hackers, script kiddie o script boy, viruxers, etc.).
- Siniestros (robo, incendio, inundación): una mala manipulación o mala intención derivan en la pérdida del material o de los archivos.
- Personal técnico interno: técnicos de sistemas, administradores de bases de datos, técnicos de desarrollo, etc. Los motivos que se encuentran entre los habituales son: disputas internas, problemas laborales, despidos, fines lucrativos, espionaje, etc.

- Fallos electrónicos o lógicos de los sistemas informáticos en general.
- Catástrofes naturales: rayos, terremotos, inundaciones, rayos cósmicos, etc.

El hecho de conectar una red a un entorno externo da la posibilidad de que algún atacante pueda entrar en ella y hurtar información o alterar el funcionamiento de la red. Sin embargo el hecho de que la red no esté conectada a un entorno externo, como Internet, no garantiza la seguridad de la misma. De acuerdo con el Computer Security Institute (CSI) de San Francisco, aproximadamente entre el 60 y 80 por ciento de los incidentes de red son causados desde dentro de la misma. Basado en el origen del ataque podemos decir que existen dos tipos de amenazas:

- Amenazas internas: generalmente estas amenazas pueden ser más serias que las externas, por varias razones como:
    - Si es por usuarios o personal técnico, conocen la red y saben cómo es su funcionamiento, ubicación de la información, datos de interés, etc. Además tienen algún nivel de acceso a la red por las mismas necesidades de su trabajo, lo que les permite mínimos movimientos.
    - Los sistemas de prevención de intrusos o IPS, y firewalls son mecanismos no efectivos en amenazas internas por no estar, habitualmente, orientados al tráfico interno. Que el ataque sea interno no tiene que ser exclusivamente por personas ajenas a la red, podría ser por vulnerabilidades que permiten acceder a la red directamente: rosetas accesibles, redes inalámbricas desprotegidas, equipos sin vigilancia, etc.
- Amenazas externas: Son aquellas amenazas que se originan fuera de la red. Al no tener información certera de la red, un atacante tiene que realizar ciertos pasos para poder conocer qué es lo que hay en ella y buscar la manera de atacarla. La ventaja que se tiene en este caso es que el administrador de la red puede prevenir una buena parte de los ataques externos.

El tipo de amenazas según el efecto que causan a quien recibe los ataques podría clasificarse en:

- Robo de información.
- Destrucción de información.
- Anulación del funcionamiento de los sistemas o efectos que tiendan a ello.
- Suplantación de la identidad, publicidad de datos personales o confidenciales, cambio de información, venta de datos personales, etc.
- Robo de dinero, estafas.

Se pueden clasificar por el modus operandi del atacante, si bien el efecto puede ser distinto para un mismo tipo de ataque:

- Virus informático: malware que tiene por objeto alterar el normal funcionamiento de la computadora, sin el permiso o el conocimiento del usuario. Los virus, habitualmente, reemplazan archivos ejecutables por otros infectados con el código de este. Los virus pueden destruir, de manera intencionada, los datos almacenados en una computadora, aunque también existen otros más inofensivos, que solo se caracterizan por ser molestos.
- Phishing: suplantación de identidad.
- Ingeniería social.

- Denegación de servicio.
- Spoofing: de ARP, DNS, IP, DHCP, etc.

Bases de la Seguridad

Hablar de seguridad Teleinformática en términos absolutos es imposible y por ese motivo se habla más bien de fiabilidad del sistema, que, en realidad es una relajación del primer término.
Definimos la Fiabilidad como la probabilidad de que un sistema se comporte tal y como se espera de él.
En general, un sistema será seguro o fiable si podemos garantizar cinco aspectos:

- Confidencialidad: acceso a la información solo mediante autorización y de forma controlada.
- Integridad: modificación de la información solo mediante autorización.
- Disponibilidad: la información del sistema debe permanecer accesible mediante autorización.
- Autenticación o Autentificación: propiedad que permite identificar el generador y origen de la información.
- No repudio o irrefutabilidad: la información se garantiza tanto que sale de origen como que llega a destino.

Existe otra propiedad de los sistemas que es la Confiabilidad, entendida como nivel de calidad del servicio que se ofrece. Pero esta propiedad, que hace referencia a la disponibilidad, estaría al mismo nivel que la seguridad. En este caso mantenemos la Disponibilidad como un aspecto de la seguridad[47].

Procedimientos de Seguridad en Sistemas Teleinformáticos

Los procedimientos de seguridad son técnicas que se utilizan para implementar un servicio de seguridad, es decir, aquel mecanismo que está diseñado para detectar, prevenir o recuperarse de un ataque de seguridad[48]; dicho de otra manera, para cumplir con los cinco aspectos que garantizan que un sistema sea seguro o fiable, y los mismos son:

- Encriptación: proporciona confidencialidad de la información y se lleva a cabo por medio de la criptografía simétrica y/o asimétrica.
- Firma digital: se lleva a cabo por medio de la criptografía asimétrica y asegura la integridad, no repudio y autenticidad del mensaje.
- Control de acceso: garantiza la autenticación y autorización. Utiliza la identidad autenticada para determinar y aplicar los derechos de acceso correspondientes a la

misma, de forma que si la entidad intenta acceder a un recurso no autorizado, este mecanismo rechazará dicha acción.

- Redundancia: provee disponibilidad tanto de la información como de los servicios en caso de comprometimiento de los mismos.

Consideraciones Generales

Según el contexto en el que se aplique, la seguridad teleinformática toma una importancia y unas medidas completamente distintas.
Un usuario particular deberá, en la medida de lo posible, preservar intactos los datos que considere confidenciales, privados y personales. Ya sea cuando intercambie información con su círculo más allegado, cuando consulte sus datos bancarios en línea o cuando realice sus compras por internet.
Para una organización, el sistema de información representa su valor, es lo esencial que hay que proteger. Comprometer este sistema es comprometer la empresa. Por consiguiente, conviene asegurar la seguridad del sistema, es decir, garantizar que los recursos se utilicen únicamente en el marco previsto, por las personas acreditadas y , sobre todo, que no se utilicen en cualquier situación.
Sin embargo, la seguridad no tiene que ser un obstáculo en la vida cotidiana y debe permitir utilizar el sistema con total confianza.
A nivel de un país, la inseguridad del sistema de información no es asumible. Los procedimientos puestos en práctica tienen que estar a la altura de la información que se protege, ya que está en juego la seguridad de toda la nación.
Una brecha en el sistema podría atentar contra los intereses fundamentales de cada individuo, minar la confianza pública del estado o, peor aún, convertirlo en víctima de un acto de terrorismo,

Capítulo 2 - Aspectos Criptográficos

Introducción Histórica

Eran mediados del siglo XIV, la dinastía Yuan reinaba el territorio chino, y los gobernantes, a fin de asegurar su dominio, imponían la orden de que cada diez familias usaran un solo cuchillo en su vida cotidiana (entre otras atrocidades y represiones) con el objetivo de que la gente quedase sin armas de metal por si querían levantarse en rebelión. Los habitantes hartos de las injusticias, decidieron sublevarse; para ello, los organizadores concibieron la idea de promover a los vecinos a regalarse mutuamente unas tortas llamadas "moon" en vísperas de las fiestas de otoño. Dentro de las tortas se puso una pequeña octavilla con un mensaje táctico para actuar en conjunto. La rebelión fue un éxito total y derivó en el establecimiento de la dinastía Ming[49].

El individuo, desde el momento en que experimentó la motivación de desenvolverse socialmente, sintió la necesidad de comunicarse, pero muchas veces, dicha comunicación debiera ser secreta para cierta porción de la sociedad; es decir, de alguna u otra forma debió comunicarse de manera enigmática o encubierta con algunas personas, sin que otras se enteren de dicha interacción.

Eficientemente, los pobladores chinos pudieron comunicarse de forma secreta para llevar a cabo la organización de la revuelta, logrando que dicha interacción pase desapercibida para los integrantes de la dinastía Yuan.

Estas técnicas de ocultamiento de información o comunicación secreta llevan el nombre de *esteganografía*. Esta misma, entonces definida como las ciencias de la comunicación encubierta fue el puntapié de la creación de la criptografía.

La criptografía moderna surge en conjunto con la computadora. Durante la Segunda Guerra Mundial, en un lugar llamado Bletchley Park, un staff de matemáticos y criptoanalistas — entre los que se encontraba Alan Turing — trabajaban en el proyecto ULTRA con el objetivo de quebrar el cifrado de los mensajes enviados por el ejército alemán. El trabajo conjunto fue un éxito y a partir de ese momento, las comunicaciones alemanas se pudieron interceptar fácilmente, acontecimiento primordial para el primer paso a su derrota.

Desde entonces hasta el día de hoy, la tecnología criptográfica creció de manera abismal convirtiéndose en una ciencia al alcance de todos — tanto civiles como militares —, tal así que se formalizó como piedra angular en asuntos tan importantes como el comercio electrónico, telefonía móvil y las nuevas plataformas de distribución de contenidos multimedia.

Definición de Criptografía

Se definirá criptografía entonces como la ciencia de la escritura enigmática, en la cual se aplican técnicas, métodos, algoritmos y conocimientos matemáticos — como álgebra y aritmética — para procesar la información y lograr este modo de escritura. Este conjunto de procedimientos tiene como primordial objetivo la transformación de la información en un código que resulte ilegible para cierta porción de los participantes que acceden a esa información pero desconocen ciertos parámetros involucrados en esta transformación.
La criptografía provee de **confidencialidad** para los datos, y es la base de varios protocolos, algunos de ellos también aseguran la **integridad** y **autenticidad** en los recursos.
Un sistema criptográfico o criptosistema posee elementos que son utilizados para el **cifrado** y **descifrado** de la información, una **clave** que es parametrizada en dichos métodos, y el contexto en el cual se encuentran definidos factores como el **mensaje** a cifrar — dato o información en **texto plano** — y su representación ya cifrada — criptograma —.
Es un hecho contrastado que la seguridad de un sistema criptográfico no pueda recaer en el desconocimiento por parte de la comunidad del funcionamiento interno de las funciones de cifrado o descifrado. Un exponente de la robustez del sistema es que dichos algoritmos y procedimientos sean públicos, de forma que su fortaleza pueda ser sometida a escrutinio público y que la seguridad del criptosistema recaiga en la no revelación uno de sus parámetros: la **clave**.
En función a si la **clave** utilizada para parametrizar dichas funciones coincide o no tanto en el cifrado como en el descifrado, los sistemas criptográficos pueden clasificarse en **simétricos** o **asimétricos**.

Criptografía Simétrica

Este sistema criptográfico emplea la misma clave tanto para **cifrar** como para **descifrar** un **mensaje**. Este parámetro se supone secreto para que la información cifrada sea protegida ante el posible acceso de terceras partes, y se la define como un **secreto compartido**, ya que el correcto funcionamiento del criptosistema recae en la distribución confidencial de la **clave** a las entidades pertinentes. No obstante, esta necesidad de compartir la clave en "secreto" limita en muchas ocasiones el uso de este tipo de criptografía a la hora de poner en contacto entidades sin ningún tipo de relación previa.
La utilización aislada de estos criptosistemas queda reducida a aquellos ámbitos en los cuales las partes pertenecientes a la comunicación cifrada disponen de algún tipo de medio de comunicación tercera mediante el cual puedan realizar el compartimento de este "secreto", sin embargo, este "compartimiento" también debería ser confidencial, lo que recaería recursivamente en lo mismo. Otra alternativa sería emplear un **centro de distribución de claves** — KDC, Key Distribution Center —, los cuales actúan como terceras partes confiables a la hora de suministrar las **claves** que protegerían las futuras comunicaciones de cifrado simétrico entre los participantes, pese a que estos KDC presenten serios problemas en

lo que respecta a la privacidad, suplantación de identidad o disponibilidad por las características de confianzas en terceros y convergencia centralizada.
Otra limitación que aborda este criptosistema es respecto a la necesidad de determinar con seguridad quien cifró o descifró un determinado mensaje ya que se utiliza la misma clave en estas dos funciones — cifrado y descifrado —, lo cual impide que pueda ser utilizada como mecanismo de no repudio o de autenticación de la identidad de la entidad cifrante.

Criptografía Asimétrica

La criptografía asimétrica o **criptografía de clave pública** difiere principalmente de la simétrica en que las claves empleadas para las funciones de cifrado/descifrado no son únicas, sino que forman pares compuestos por una **clave pública** y una **clave privada**.
Cada entidad pertinente posee un conjunto de claves de las cuales una de ellas prevalece protegida por su posesor y la otra se dispone de forma pública y visible por cualquier otra entidad tercera — de allí su clasificación en **pública** y **privada** — . La relación que prevalece entre el par de claves se entrelazan por medio de conceptos matemáticos/computacionales que aseguran y demuestran que resulta real y prácticamente imposible lograr descubrir una clave a partir del conocimiento de la otra, lo cual anula la necesidad de establecer **secretos compartidos** entre entidades ya que basta con tener acceso a una de las claves — en este caso la **pública** —.
El funcionamiento innovador que ofrece este criptosistema con respecto de los simétricos es que se pueden utilizar cualquiera de las dos claves en cualquiera de las dos funciones, es decir, si se utiliza la **clave privada** para la función de **cifrado** luego con la **clave pública** se realiza el **descifrado** y recíproco, por lo tanto, si la clave pública es de acceso comunitario, se podría determinar quién fue el que cifró — con la clave privada — el mensaje ya que cada par de claves se encuentra matemáticamente relacionado y sólo la entidad poseedora de esa **clave pública** tiene acceso a la correspondiente **privada**, convergiendo así a dos características esenciales: **confidencialidad** y **autenticidad**.
Las características ofrecidas por este criptosistema, junto con la combinación de funciones de **hashing** ofrecen otras características primordiales que hacen a un sistema totalmente fiable: **integridad** y **no repudio**. No obstante, la agrupación de estos últimos mecanismos con sus respectivas características como producto resulta ni más ni menos que a una **firma digital**.
Dicho esto, el problema surge cuando una entidad intenta acceder a la clave pública, ya que necesita realizarlo por un canal de comunicación adicional y asegurarse de que realmente esa clave accedida es auténtica, es decir, que esa **clave pública** a la que cualquiera tiene acceso pertenece realmente a la entidad con la cual se desea establecer contacto comunicacional y no fué alterada. Si efectivamente esta **clave pública** fue interceptada y modificada o reemplazada podría conllevar a la transmisión de información sensible a terceras partes equivocadas.
Otra limitación que abordan estos sistemas criptográficos es que requiere mucha capacidad de cómputo, de procesamiento y en algunos casos el mensaje a cifrar debe ser de menor o igual tamaño que la clave a utilizar, lo que lleva a su restringida utilización por cuestiones obvias.

Funciones Hashing o Resumen Digital

Al igual que los sistemas criptográficos, las **funciones hash** son algoritmos matemáticos que transforman cualquier bloque arbitrario de datos en una nueva serie de caracteres con una longitud fija. Independientemente de la longitud de los datos de entrada, el valor hash de salida tendrá siempre la misma longitud y siempre dará el mismo resultado si el algoritmo se aplica a la misma serie de caracteres, lo que proporciona una primer aproximación a la **integridad**, ya que produce un **resumen digital** como salida a una entrada determinada. El resultado de la aplicación de un algoritmo hash a una secuencia de datos es totalmente ilegible e irreversible, lo que asegura que a partir de un **resumen digital** no se pueda volver a crear la secuencia de datos iniciales, motivo principal por el que no utilizan claves y se diferencia de los algoritmos de cifrado.

Firma Digital

Una **firma digital** es un mecanismo criptográfico que permite al receptor de un mensaje firmado digitalmente identificar a la entidad originadora de dicho mensaje —**autenticación** de origen y **no repudio** — , y confirmar que el mensaje no ha sido alterado desde que fue firmado por el originador — **integridad** — .
La firma digital se aplica en aquellas áreas donde es importante poder verificar la autenticidad y la integridad de ciertos datos, por ejemplo documentos electrónicos o software, ya que proporciona una herramienta para detectar la falsificación y la manipulación del contenido.

Para garantizar la seguridad de las firmas digitales es necesario a su vez que estas sean:

- Únicas: Las firmas deben poder ser generadas solamente por el firmante y por lo tanto infalsificable. Por tanto la firma debe depender del firmante.
- Infalsificables: Para falsificar una firma digital el atacante tiene que resolver problemas matemáticos de una complejidad muy elevada, es decir, las firmas han de ser computacionalmente seguras. Por tanto la firma debe depender del mensaje en sí.
- Verificables: Las firmas deben ser fácilmente verificables por los receptores de las mismas y, si ello es necesario, también por los jueces o autoridades competentes.
- Innegables: El firmante no debe ser capaz de negar su propia firma.
- Viables: Las firmas han de ser fáciles de generar por parte del firmante.

El proceso de firma y verificación de la misma se puede resumir en lo siguiente: Se supone a dos participantes de una comunicación — Emisor A y Receptor B —, el primero confecciona un mensaje M con el objetivo de que cuando el mismo es recepcionado por B se pueda verificar la integridad y autenticidad del mismo. El emisor aplica al mensaje M una función

hashing — H(M) — la cual posteriormente es sometida a un cifrado con su clave privada — C(H(M)) —, lo adjunta al mensaje a enviar — P (M | C(H(M))) — y luego P es enviado a B. El receptor al recibir P lo desglosa obteniendo el mensaje M por un lado y una concatenación C. Por un lado aplica la función de hashing al mensaje M obteniendo un resultado X, por otro lado descifra C con la clave pública de A — conocida anteriormente por otros medios — obteniendo así H. Por último el Receptor B compara los resultados de X y H, no obstante si X = H entonces se confirma que el mensaje no sufrió modificaciones y que fue enviado por el Emisor A ya que es el único que posee la otra parte del par de claves utilizado en el criptosistema.

Intercambio o establecimiento de claves

Un **protocolo de establecimiento de claves** (key establishment protocols), también llamados **protocolos de intercambio de claves** (*key exchange protocols*) es un protocolo criptográfico en el que se establece una secuencia de pasos entre dos o más participantes a través de la cual los participantes se ponen de acuerdo en el valor de una información secreta compartida. A la información secreta compartida se le suele llamar clave debido a que esa información se suele usar como clave de algún algoritmo criptográfico simétrico y que, generalmente este proceso de compartimento es realizado por medio de sistemas criptográficos asimétricos.

Protocolo criptográfico

Un protocolo describe la forma en que un algoritmo debe usarse. Un protocolo lo suficientemente detallado incluye detalles acerca de las estructuras de datos y representaciones, punto en el cual puede usarse para implementar versiones interoperables múltiples de un programa.
Los protocolos criptográficos se usan ampliamente para transporte de datos seguros a nivel de aplicación. Un protocolo criptográfico comúnmente incorpora por lo menos uno de los siguientes aspectos:

- Establecimiento de claves
- Autenticación de entidades
- Cifrado simétrico y autenticación de mensajes
- Transporte de datos en forma segura a nivel de aplicación
- Métodos de no repudio

Por ejemplo, el protocolo criptográfico CHACHA20|Poly1305[50] es una implementación o una forma en que un algoritmo de cifrado simétrico debe utilizarse junto con un método de autenticación, por otro lado RSA[51]|SHA512 es una implementación concreta de criptografía asimétrica para establecimiento de claves simétricas, combinando con SHA512 como implementación de un algoritmo de resumen digital para así proporcionar una

funcionalidad de firma digital.

Criptografía de Curvas Elípticas

En las últimas décadas, la criptografía con curvas elípticas (ECC) ha adquirido una creciente importancia, llegando a formar parte de los estándares industriales. Su principal logro se ha conseguido en los criptosistemas basados en el famoso problema del logaritmo discreto[52], como los de tipo ElGamal[53]. Estos criptosistemas planteados en el grupo de puntos de una curva elíptica garantizan la misma seguridad que los construidos sobre el grupo multiplicativo de un cuerpo finito, pero con longitudes de clave mucho menores.
La criptografía con curvas elípticas aparece como una alternativa a los criptosistemas de clave pública clásicos como el RSA y el ElGamal, tanto por la disminución del tamaño de las claves que se requieren como por el abanico de grupos que ofrecen en el mismo cuerpo base. Su implantación en algunos sistemas de comunicaciones es un hecho constatable y su uso aumenta día a día debido a sus ventajas. Por ejemplo, se usa en tarjetas inteligentes, sistemas de identificación por radiofrecuencia, sistemas de voto electrónico, etc.
Como antes ya se mencionó, los sistemas de criptografía asimétrica o de clave pública utiliza dos claves distintas: una de ellas puede ser pública, la otra es privada. La posesión de la clave pública no proporciona suficiente información para determinar cuál es la clave privada. Este tipo de sistemas se basa en la dificultad de encontrar la solución a ciertos problemas matemáticos, donde uno de estos problemas es el llamado **logaritmo discreto**. Encontrar el valor de ***b*** dada la ecuación ***a^(b)=c***, cuando ***a*** y ***b*** son valores conocidos, puede ser un problema de complejidad exponencial para ciertos grupos finitos de gran tamaño; mientras el problema inverso, la exponenciación discreta puede ser evaluado eficientemente usando por ejemplo exponenciación binaria.
Una curva elíptica es una curva plana definida por una ecuación de la forma ***y^(2)=x^(3)+ax+b***. Con el conjunto de puntos ***G*** que forman la curva (todas las soluciones de la ecuación más un punto ***O***, llamado punto en el infinito) más una operación aditiva +, se forma un grupo abeliano[54]. Si las coordenadas ***x*** e ***y*** se escogen desde un cuerpo finito, entonces se está en presencia de un grupo abeliano finito. El problema del logaritmo discreto sobre este conjunto de puntos se cree que es más difícil de resolver que el correspondiente a los cuerpos finitos. De esta manera, las longitudes de claves en criptografía de curva elíptica pueden ser más cortas con un nivel de seguridad a la altura de las circunstancias contemporáneas.
Un algoritmo de curva elíptica funcionará en un subgrupo cíclico de una curva elíptica restringida a un campo finito. Luego, un algoritmo ECC necesita los siguientes parámetros:

- Un número primo ***p***, que determinará el tamaño del campo finito.
- Los coeficientes ***(a,b)*** de la ecuación de la curva elíptica.
- El punto base ***G*** que generará nuestro subgrupo.
- El orden ***n*** del subgrupo.
- El cofactor ***h*** del subgrupo.

Es decir, los parámetros para definir nuestros algoritmos, son el sexteto ***(p,a,b,G,n,h)***.
Por otra parte, la **clave privada** es un número entero ***d*** al azar elegido entre $\{1,\cdots,n-1\}$ (donde ***n*** es el orden del subgrupo).
La **clave pública** es el punto ***H=dG*** (dónde es el punto base del subgrupo).
Si se conoce ***d*** y ***G*** (junto con los demás parámetros), encontrar ***H*** es sencillo. Pero si se conoce ***H*** y ***G***, encontrar la **clave privada *d*** es considerablemente complejo, ya que requiere que se resuelva el problema del logaritmo discreto.

ECDH (Elliptic curve Diffie-Hellman)

ECDH es una variante del algoritmo Diffie-Hellman para curvas elípticas. De hecho, es un protocolo de establecimiento de claves, más que un algoritmo de cifrado. Esto básicamente significa que ECDH establece, hasta cierto punto, como se deberían generar e intercambiar las claves entre las partes implicadas.
El problema que resuelve este algoritmo es el siguiente: dos partes (***A*** y ***B***) quieren intercambiar información de forma segura, para que una tercera parte ("the Man In the Middle") no pueda decodificar la información si intercepta el mensaje.
Este sería el proceso:

1. Para comenzar, ***A*** y ***B*** generan su propio par de claves. Se tiene la clave privada ***dA*** y la clave pública ***HA=dAG*** de ***A***, y las claves ***dB*** y ***HB=dBG*** de ***B***.
2. Tanto ***A*** como ***B*** utilizan los mismos parámetros: el mismo punto base ***G*** en la misma curva elíptica restringida sobre el mismo campo finito.
3. ***A*** y ***B*** intercambian sus claves públicas ***HA*** y ***HB*** a través de un canal inseguro. El intermediario podría interceptar una o ambas claves públicas, pero en ningún caso tendrá acceso a las claves privadas (***dA*** y ***dB***) sin resolver el problema del logaritmo discreto.
4. A calcula ***S=dAHB*** (usando su propia clave privada y la clave privada de ***B***), y ***B*** calcula ***S=dBHA***(usado su propia clave privada y la clave pública de ***A***). Ver que el resultado ***S*** es el mismo para ambos, de hecho:

$$S=dAHB=dA(dBG)=dB(dAG)=dBHA$$

El intermediario, sin embargo, sólo conoce las claves públicas ***HA*** y ***HB*** (junto con el resto de los parámetros) y nunca podrá conocer la clave secreta compartida ***S***. Esto es lo que se conoce como el problema de Diffie-Hellman, que se puede definir como: Dados tres puntos ***P***, ***aP*** y ***bP***, ¿cuál es el resultado de ***abP***?
El protocolo de establecimiento de claves Diffie-Hellman: ***A*** y ***B*** pueden calcular "sencillamente" su clave secreta compartida, pero el "intermediario" tendría que resolver un problema "complejo".
Ya en esta instancia, tanto ***A*** como ***B*** han conseguido su clave secreta compartida y pueden intercambiar información de manera segura con cualquier algoritmo de cifrado simétrico.

ECDSA (Elliptic Curve Digital Signature Algorithm)

ECDSA es una variante del Digital Signature Algorithm (DSA) que utiliza la criptografía de curva elíptica como variante de la criptografía asimétrica o de clave pública.
Para ejemplificar su funcionamiento se realizará el siguiente escenario: ***A*** quiere firmar un mensaje con su clave privada (***dA***), y ***B*** quiere validar la firma de ***A*** usando su clave pública (***HA***). Nadie, a excepción de ***A***, debería poder generar firmas válidas. Todo el mundo debería poder comprobar una firma.
Una vez más, ***A*** y ***B*** usan los mismos parámetros.
ECDSA funciona con el hash del mensaje, en vez de con el mensaje en sí mismo. La elección de la función para generar el hash depende del que la va a utilizar, pero obviamente debería ser una función criptográficamente segura. El hash del mensaje tiene que truncarse, para que la longitud en bits del hash sea la misma longitud en bits de ***n*** (el orden del subgrupo). El hash truncado es un entero y se denotará con la letra ***z***.
El algoritmo ejecutado por ***A*** para firmar el mensaje es el siguiente:

1. Escoge un número entero aleatorio ***k*** elegido entre{1,⋯,n−1} (dónde nn todavía es el orden del subgrupo).
2. Calcula el punto ***P=kG*** (donde *G* es el punto base del subgrupo).
3. Calcula el número *r=xP* ***mod*** *n* (dónde ***xP*** es la coordenada ***x*** de ***P***).
4. Si ***r=0*** entonces elige otro valor para ***k*** y vuelve a intentarlo.
5. Calcula ***s=k−1(z+rdA) mod*** *n* (donde ***dA*** es la clave privada de ***A*** y ***k^(-1)*** es la inverso multiplicativo de *k* ***mod*** *n*).
6. Si ***s=0***, entonces elige otro ***k*** y prueba otra vez.

El par ***(r,s)(r,s)*** es la firma.
Para poder verificar la firma de ***A*** se necesitará su clave pública HA, el hash (truncado) ***z*** y, la firma ***(r,s)***.

1. Calcula el número entero ***u1=s^(−1) z mod*** *n*
2. Calcula el entero ***u2=r^(−1) z mod*** *n*
3. Por último, calcula el punto ***P=u1 G + u2 HA***

Finalmente la firma solo es válida si *r=xP* ***mod*** *n.*

Capítulo 3 - Arquitectura de Sistemas Distribuidos

Introducción Histórica

A fines de los años ochenta comienzan a converger diferentes aspectos en el universo informático, implantándose como detonante principal de un cambio en el proceso de ingeniería de los sistemas de información. Antes que nada comienza la explosión de las computadoras personales irrumpiendo fuertemente dentro de los centros de cálculos de las organizaciones. A pesar de que la mayor parte de la lógica de negocio aún residía en mainframes o en grandes estaciones de trabajo, la masiva presencia de equipos de bajo coste permitía a los ingenieros a desarrollar grandes aplicaciones modulares que podrían procesar información de manera integral y ubicada en diferentes ordenadores, dando así un innovador enfoque en el desarrollo de sistemas informáticos.
Inicialmente estos módulos de software funcionaban como bloques de cómputo independientes dentro del sistema, pero pronto, los desarrolladores de software vieron la necesidad de disponer de nuevas técnicas aplicadas a las comunicaciones y transferencia de información entre dichos elementos de cómputo. No obstante, y ajeno a estas necesidades comienzan a consolidarse nuevas líneas de investigación en cuestiones de programación

concurrente[55] y procesamiento en paralelo[56] motivados por la presencia y uso de nuevos sistemas operativos multiprocesadores.
Estos dos últimos acontecimientos dieron lugar a los primeros indicios en la elaboración de nuevas tecnologías para la programación de software aplicada a los sistemas informáticos distribuidos[57]. Más específicamente, uno de los resultados iniciales fue el desarrollo de la técnica RPC (Remote Procedure Call), origen de la gran parte de la tecnología middleware actual. Esta técnica permite que los programadores de software puedan diseñar sus aplicaciones mediante módulos comunicados entre sí, comportándose como un conjunto de procesos cooperativos independientes.
Este nuevo paradigma de construir aplicaciones divididas en partes comunicantes y residentes en distintos ambientes de cómputo fue un gran paso en el campo programación distribuida y obligó a los ingenieros de software a integrarlas con las aplicaciones antiguas, dando así el primer paso a los sistemas *legacy*, que hacen referencia a la integración de partes del sistema actual con sistemas anteriores que actualmente se encuentran en funcionamiento.
No obstante, la presencia de diferentes modelos de sistemas vigentes y con necesidad de interoperar entre sí, influenciados por los intereses de las grandes corporaciones y la constante evolución de las nuevas tecnologías (Json, Servlets, XML, SOAP, Rest, etc.), está haciendo que los ingenieros de sistemas tengan que hacer grandes procesos de ingeniería de requerimientos para seleccionar aquellas tecnologías adecuadas para el desarrollo de sus sistemas. Incluso, en la mayoría de las situaciones, los diseñadores de software se ven obligados a utilizar e incorporar múltiples métodos y técnicas para dar soporte a distintos clientes —de software y usuarios— del sistema de información.

Definición de Sistemas Distribuidos

Antes de comenzar a definir este término, primero se describe qué es **computación distribuida**. La computación distribuida se refiere a cualquier circunstancia en la cual se desenvuelve un sistema en una red de computadoras y trata de describir las tendencias hacia la funcionalidad distribuida: sistemas procesamiento distribuido, distribuidos, bases de datos distribuidas y cualquier otro término computacional que sea aplicado a componentes de un sistema modularmente separados que interactúan entre sí. Se puede mencionar entonces, que la Computación Distribuida hace referencia a los servicios que provee un Sistema de Computación Distribuido.
Sin más preámbulos, George Coulouris[58] define a un sistema distribuido como aquel que está compuesto por varias computadoras autónomas conectadas mediante una red de comunicaciones y equipadas con programas que les permitan coordinar sus actividades y compartir recursos. Por otro lado, Bal ofrece una definición muy similar: Un sistema de computación distribuida está compuesto por varios procesadores autónomos que no comparten memoria principal, pero cooperan mediante el paso de mensajes sobre una red de comunicaciones. Según Schroeder, todo sistema distribuido tiene tres características básicas: Existencia de varias computadoras, interconexión y estado compartido. Para Tanenbaum, un sistema distribuido era una colección de computadores independientes que aparecen ante los

usuarios como un único computador. Y Liu, se refería a sistemas distribuidos como un conjunto de computadoras independientes, interconectados a través de una red y que son capaces de colaborar entre sí para realizar una tarea.
Finalmente, el término de Computación Distribuida se define de varias maneras y lo mismo se aplica al término de Sistema Distribuido.

Características de los Sistemas Distribuidos

Una de las principales e importantes características que poseen los sistemas distribuidos es que debe ser considerablemente sencillo de escalar, lo cual se logra al tener computadoras independientes, pero al mismo tiempo "ocultar" las funciones de dichas computadoras en el sistema, aplicando esto mismo a la organización interna del sistema. Generalmente un sistema distribuido debe estar siempre disponible a pesar de que ciertas partes que lo conforman puedan no estar funcionando. Los usuarios y las aplicaciones clientes del sistema distribuido no deben notar en ningún momento que estas partes están siendo reemplazadas, reparadas, o que se están agregando nuevas funcionalidades al sistema para poder dar un servicio más eficiente y completo.
Todo desarrollador de software debe contar con los conocimientos necesarios para enfrentarse a todos los desafíos que pueden surgir al momento de considerar los requerimientos para el desarrollo de un sistema distribuido. No obstante, las diferentes características a tener en cuenta a la hora de desarrollar o interactuar con un sistema distribuido deben ser:

- Heterogeneidad: con respecto a las variedades y diferencias que se dan en los contextos de ejecución de los sistemas distribuidos, como ser en diferentes redes, sistemas operativos, variedad en hardware, implementaciones, cantidad diversa de desarrolladores de software y lenguajes de programación. La mejor forma de afrontar esta característica es mediante la utilización de estándares en cuanto a protocolos de comunicación y de presentación de información
- Middleware: se refiere a una capa de software que proporciona una abstracción de programación y de la heterogeneidad subyacente en los diferentes contextos, de manera de lograr modelos computacionales uniformes para facilitar la programación de los mismos.
- Escalabilidad: es una característica que permite determinar si un sistema puede ser ampliado en cuestiones de escalamiento funcional y computacional y/o re-implementado en distintos aspectos.
- Seguridad: concierne a todo lo que tenga que ver con asegurar que no ocurrirán violaciones en la legitimidad de los diferentes datos que envían los clientes para solicitar información a un servidor, y por supuesto, con la información que estos reciben como respuesta a sus peticiones. No obstante, no es suficiente con asegurar

que estos datos serán transmitidos de forma “oculta”, sino que también cumplir con los atributos principales de la seguridad teleinformática —confidencialidad, integridad, disponibilidad y autenticidad—. La seguridad es relativa a la amenaza que cada sistema afronta, afecta a todos los puntos del sistema y debe de ser sencilla de confrontar.

- Apertura: ocurre cuando un sistema ofrece servicios desarrollados de acuerdo a reglas estandarizadas que describen la sintaxis y la semántica de dichos servicios para brindar facilidad y posibilidad de que otros externos puedan interactuar fácilmente.
- Tolerancia a fallos: en cuanto a la transparente detección, enmascaramiento y rápida recuperación ante fallos.
- Concurrencia: posibilidad de que un mismo recurso sea accedido/procesado por varios componentes del sistema al mismo tiempo.
- Transparencia: es el ocultamiento al usuario y al programador de la separación de los componentes del sistema distribuido, de forma que se lo perciba como una unidad, más que como una colección de componentes individuales.

Programación por Capas

El objetivo principal de este estilo de programación radica en la separación de la lógica de negocios de la lógica de diseño; por ejemplo, la separación de la capa de datos de la de presentación al usuario, de forma tal de llevar a cabo el desarrollo del software en varios niveles, cada uno de ellos abstraídos totalmente y a su vez comunicándose entre sí a través de una interfaz de comunicación que cada uno publique para ser consumida.
En estas técnicas de programación se le confía a cada nivel una misión simple, lo que permite el diseño de arquitecturas escalables, cohesivas, coherentes e interoperables.
El diseño más utilizado actualmente es el diseño en tres niveles (o en tres capas):

- Capa de Presentación: es la que interactúa directamente con el usuario, se encarga de la presentación de la información (*outputs* del sistema), de la captura de información del usuario (*inputs* del sistema) y realiza un filtrado previo para comprobar que no hay errores de formato. Esta capa se comunica únicamente con la capa de negocio y también es conocida como interfaz gráfica, la cual debe tener la característica de ser "amigable" (entendible y fácil de usar) para el usuario.
- Capa de Negocio: es el núcleo de ejecución y procesamiento del sistema, se reciben las peticiones del usuario y se envían las respuestas. Esta capa se comunica con la capa de presentación, (para recibir estas solicitudes y presentar los resultados) y con la capa de datos (para solicitar al gestor de base de datos para almacenar o recuperar datos).
- Capa de Datos: es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

Estas tres capas podrían estar ubicadas en un único dispositivo de hardware o en varios, sin embargo, usualmente lo más utilizado es que las mismas residan en varios ordenadores de forma que, si el tamaño o complejidad de la base de datos aumenta, se pueda separar en varios ordenadores los cuales reciben las peticiones del ordenador en que resida la capa de negocio. Si, por el contrario, fuese la complejidad en la capa de negocio lo que obligase a la separación, esta capa de negocio podría residir en uno o más ordenadores que realizan solicitudes a una única base de datos.

Modos de arquitecturas de los Sistemas Distribuidos

Las arquitecturas de los sistemas reflejan la estrategia básica utilizada para estructurar dicho sistema y pertenece al diseño de más alto nivel de la estructura del mismo. Esta misma consiste en un conjunto de patrones y abstracciones coherentes que proporcionan un marco definido y claro para interactuar con el desarrollo de la totalidad del sistema.
Existen varios tipos de arquitecturas que pueden ser aplicadas de acuerdo a la estrategia tecnológica a implementar en diferentes situaciones determinadas tanto por los participantes como por las diferentes necesidades y requerimientos.

Cliente/Servidor

La arquitectura cliente-servidor es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados **servidores**, y los demandantes, llamados **clientes**. Un cliente realiza peticiones a otra unidad de cómputo, el servidor, el cual le proporciona una respuesta. Esta idea también se puede aplicar a programas que se ejecutan sobre una sola computadora, aunque es más ventajosa en un sistema operativo multiusuario distribuido a través de redes de computadoras.
En esta arquitectura la capacidad de proceso está repartida entre los **clientes** y los **servidores**, aunque son más importantes las ventajas desde el punto de vista de la gestión y organización debido a la centralización de la administración de la información y la separación de las responsabilidades, lo que facilita y clarifica el diseño del sistema.
La división entre **cliente** y **servidor** es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre un solo hardware ni es necesariamente una unidad de cómputo de software. Los tipos específicos de servidores incluyen los servidores web, los servidores de archivo, los de correo electrónico, etc. Mientras que sus propósitos varían de un servicio a otros, la arquitectura básica seguirá siendo la misma.
En esta arquitectura el remitente de una solicitud es conocido como el **cliente** y presenta las siguientes características:

- Es quien inicia solicitudes o peticiones, tienen por lo tanto un papel activo en la comunicación.
- Espera y recibe las respuestas del servidor.

- Por lo general, puede conectarse a varios servidores a la vez.
- Generalmente interactúa directamente con los usuarios finales mediante una interfaz gráfica de usuario.

En la otra parte, al receptor de la solicitud enviada por el cliente se lo conoce como servidor y sus características son:

- Al iniciarse espera a que lleguen las solicitudes de los clientes, desempeñan entonces un papel pasivo en la comunicación.
- Tras la recepción de una solicitud, la procesan y luego envían la respuesta al cliente.
- Por lo general, aceptan conexiones desde un gran número de clientes (en ciertos casos el número máximo de peticiones puede estar limitado).
- No es frecuente que interactúen directamente con los usuarios finales.

Peer to Peer

La arquitectura P2P o Peer to Peer elimina la necesidad de un **servidor** central y la de una relación cliente/servidor, pasando a funcionar como una arquitectura de red entre partes iguales sin jerarquías y distribuidas en las cuales las aplicaciones pueden comunicarse entre sí intercambiando información sin la intervención de un ente controlador central.
Una característica típica de este tipo de arquitectura es la descentralización, es decir que un "peer", nodo o unidad de cómputo perteneciente a la misma tiene dos papeles (de servidor y al mismo tiempo de cliente) frente a otras unidades de cómputos que pertenecen al mismo contexto. Sin embargo, esta característica no es obligatoria ya que puede darse el caso de algunas soluciones que utilizan una combinación entre esta y la de cliente /servidor.
Otras características que presentan estas arquitecturas son:

- Escalabilidad: Las redes P2P tienen un alcance mundial con cientos de millones de usuarios potenciales. En general, lo deseable es que cuantos más nodos estén conectados a una red P2P mejor será su funcionamiento. Así, cuando los nodos llegan y comparten sus propios recursos, los recursos totales del sistema aumentan. Esto es diferente en una arquitectura del modo servidor/cliente con un sistema fijo de servidores, en los cuales la adición de más clientes podría significar una transferencia de datos más lenta para todos los usuarios. Algunos autores advierten de que si abundan en grandes cantidades este tipo de redes, las cliente/servidor, podrían llegar a su fin, ya que a cada una de estas redes se conectarán muy pocos usuarios.
- Robustez: La naturaleza distribuida de las redes peer-to-peer también incrementa la robustez en caso de haber fallos en la réplica excesiva de los datos hacia múltiples destinos, y —-en sistemas P2P puros—- 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.
- Los costos están repartidos entre los usuarios. Se comparten recursos a cambio de otros recursos, como 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, evitar el contenido infectado, evitar el espionaje de las comunicaciones entre nodos, creación de grupos seguros de nodos dentro de la red, protección de los recursos de la red.

Consideraciones Generales

La arquitectura tradicional para el desarrollo de aplicaciones distribuidas está basada en el modelo cliente/servidor. Este modelo, empleado en internet para la práctica totalidad de los servicios más comunes (web, ftp, telnet, etc), consta de una serie de clientes que acceden simultáneamente a un conjunto de servidores que ofrecen ciertos recursos o aplicaciones. Cuando se pretenden descargar grandes volúmenes de información a muchos clientes, la arquitectura cliente/servidor es lenta, costosa y no escalable. No aprovecha, por ejemplo, que un mismo servidor remoto pueda estar siendo objeto de acceso por dos clientes muy cercanos entre sí.
En cambio, en la arquitectura para la distribución de contenidos empleada por los programas P2P es una arquitectura cliente/servidor, pero en la que también colaboran los clientes o pares. En este caso, los ordenadores clientes se ayudan entre sí, convirtiéndose en servidores de otros clientes. Estas redes pueden crecer indefinidamente sin incrementar el tiempo de búsqueda y sin necesidad de costosos recursos centralizados.
Según el grado de centralización entonces, se pueden implementar diferentes soluciones arquitectónicas, como ser puras o híbridas. En una solución P2P pura, los nodos cumplen el papel de cliente y servidor, y no hay otro servidor central que lo gestione. En una arquitectura híbrida, un servidor central es usado para mantener información sobre los nodos, y cualquier información de otros nodos es gestionado por este servidor central. Los nodos o peers mantienen la información.

Capítulo 4 - Arquitectura de Certificados Digitales

Introducción Histórica

La historia de la **certificación digital** inicia a mediados de la década de los 70, cuando Whitfield Diffie y Martin Hellman[59] publican el artículo titulado "New Directions in Cryptography" que introduce el concepto de criptografía asimétrica y proponen la utilización de un **archivo público** que sería consultado por las entidades pertinentes para averiguar las claves públicas del resto pertenecientes al sistema. Para evitar que un atacante pudiera interceptar de modo de alterar o modificar dicho archivo, todas las comunicaciones deberían ser llevadas a cabo implementando **firmas digitales**. No obstante, esta propuesta inicial presentaba numerosos inconvenientes tanto desde el punto de vista del rendimiento como de la seguridad.

Esta necesidad de asociar de forma confiable las claves públicas de los usuarios a su identidad lleva a Loren Kohnfelder[60] en 1978 a proponer el concepto de **certificado digital de clave pública** como un documento firmado digitalmente que contiene tanto la clave pública como la identidad del poseedor de la clave privada correspondiente, desligando así la dependencia a la base de datos pública centralizada propuesta por los anteriores. A su vez esta estructura de datos formada por la clave pública y la identidad del poseedor de la correspondiente privada debía ser emitida y firmada por alguna otra entidad que se ganase la confianza de los usuarios pertenecientes a la comunicación. De esta forma, este documento firmado digitalmente, junto a un mecanismo de comprobación de autenticidad e integridad conlleva a la posibilidad de ser almacenado en cualquier contexto no confiable de forma replicada asegurando su disponibilidad y no alteración.

Finalmente, esta arquitectura recae en la implantación del estándar **X.509**[61] como una propuesta de directorio global de entidades, junto con la implementación de la **infraestructura de clave pública** como un sistema integral con la función de la gestión del ciclo de vida de los certificados digitales.

La solución final consiste básicamente en que cada usuario del sistema disponga de un par de claves únicas y, a diferencia de lo que sucedía con la criptografía simétrica, la clave privada no es un secreto compartido sino que debe ser protegida por cada usuario (criptografía asimétrica). No obstante, la clave pública debe ser de libre acceso y disponible para cualquier otro usuario pueda obtenerla con el fin de proteger las comunicaciones realizadas con el portador de la misma. La seguridad de este sistema criptográfico recae en el hecho de que resulta computacionalmente inviable intentar descubrir una clave a partir del conocimiento de la otra, lo cual soluciona el problema principal de la criptografía simétrica, es decir, no se

necesita el compartimento de la misma clave entre las entidades, ya que basta únicamente con tener acceso a dichas claves públicas.
El hecho de que una clave sea de libre acceso para cualquier entidad perteneciente a la red, y de que no se necesiten de canales adicionales o encubiertos para obtener la misma, no la hace **auténtica**. Resulta completamente indispensable obtener un máximo grado de certeza de que la clave pública obtenida pertenece realmente a la entidad de quien dice ser, es decir, que esa clave pública es del usuario con el que se desea establecer la comunicación. Si surgiese algún problema en la asociación de la clave pública y el usuario portador de la misma, podría comprometerse de forma absoluta la confidencialidad, integridad, autenticidad y no repudio de la comunicación.
Descripto esto, junto con la necesidad de asociar confiablemente una clave pública a una entidad converge en los principios de la certificación digital, la cual resuelve eficazmente problemas relacionados a los nombrados en el párrafo anterior.
La principal diferencia entre término "certificación digital" con el "certificación de identidad", es que el primero se puede definir como un documento que posee información acerca de la entidad que la porta y, dicha información no se restringe a aspectos de identificación. Por lo tanto, este certificado asocia competencias o capacidades a las claves públicas y son denominados como **certificados de credenciales**, los cuales suelen estar ligados a los certificados de identidad con la particularidad de que son independientes uno del otro.
A lo largo de los años surgieron muchas especificaciones en materia de certificados de credenciales en la comunidad científica[68][69], donde cada una de estas proponen mecanismos de gestión de pertenencia a grupos y especificación de privilegios como herramienta fundamental de gestión de autorización. Sin embargo, el diseño e implementación de mecanismos de gestión del ciclo de vida de este tipo de certificados continúa siendo aún un campo abierto de investigación y desarrollo.

Esquemas de confianza

En la actualidad entre los esquemas de certificación de identidades mundialmente aceptados se encuentran el estándar X.509 y PGP[65]. Estas soluciones proveen diferentes modelos de confianza y eficaces a la hora de hablar de comunicaciones seguras; hoy en día se encuentran con un alto grado de implantación en la comunidad de internet.
El estándar X.509 considera que los certificados deben ser emitidos por entidades especiales, a las cuales se las denomina **autoridades de certificación**. Estas entidades centrales están permitidas a emitir certificados, o crear nuevas identidades digitales, y estos son considerados válidos por parte de una gran comunidad de usuarios, donde se genera una confianza impuesta tanto por los poseedores de los certificados como por las entidades encargadas de verificarlos. Cada uno de estos certificados está firmado por una y solo una autoridad de certificación. El estándar no restringe el esquema a la existencia de una única autoridad certificante, sino que contempla la posibilidad de que muchas autoridades certificantes independientes puedan interactuar de forma simultánea. Estas relaciones entre entidades

certificantes independientes hace converger a diferentes configuraciones posibles de modelos de confianza entre las entidades emisoras de certificados: Modelo Jerárquico, Certificación Cruzada y Modelo Basado en Autoridad de Certificación Puente[66].
En el mundo de PGP no existe el concepto de autoridades de certificación, es decir, cualquier entidad de la red puede certificar la clave pública de cualquier otra utilizando este protocolo. No obstante, este certificado tendrá validez sólo para aquellas mismas que certificaron dicho certificado, y esta configuración de modelo de confianza se denomina: Modelo de Confianza en Red.

Modelo Jerárquico

Posee una entidad de certificación raíz, cuya clave pública está contenida en un certificado autofirmado, es decir, certificado por sí misma, el cual debe ser distribuido de forma confiable a todas las entidades del sistema ya que no proporciona de por sí autenticación, sino sólo integridad sobre la información contenida. Las demás entidades certificadoras se encuentran por debajo de la entidad raíz, y poseen relaciones de dependencia a su entidad de nivel superior.
Este modelo presenta ciertas desventajas a tener en cuenta. Por un lado, al ser un esquema centralizado y jerárquico, el vector de ataque se encuentra en la cima de la pirámide. Por otro lado, si la clave privada de la entidad raíz es comprometida por un u otro factor, esto impactaría inminentemente en la totalidad los certificados del sistema; dicho esto, todos estos certificados deberán posteriormente ser revocados y firmados nuevamente para lograr su convergencia.
Por último, esta arquitectura jerárquica produce una relación de dependencia y dominio entre las organizaciones a las cuales están implicadas las entidades certificantes, lo cual no siempre se cumple.

Modelo Cruzado

Las entidades certificantes operan de igual a igual, es decir, no existe una entidad raíz de la cual depender. En este esquema, cuando una entidad de certificación desea establecer una relación de confianza con otra, intercambian sus claves públicas y se certifican una con otra, de esta forma, cada una de ellas toma el papel de entidad raíz.
El principal inconveniente que presenta este modelo es el número de certificados que se requieren crear para lograr conectar varias entidades certificantes, en consiguiente, si se quiere tener disponibles ***n*** entidades certificadoras, entonces, se deberán crear ***n(n-1)*** certificados cruzados.

Modelo Puente

En este esquema las entidades no se certifican entre sí, sino respecto a una entidad llamada autoridad de certificación puente, para la cual se emplean un modelo cruzado.
Comparado con el modelo anterior, esta requiere menos cantidad de certificados.

Modelo en Red

En esta solución, es un proceso de gestión de certificados totalmente manual, en donde las entidades que se desean comunicar construyen sus propios caminos de certificación a través de toda la red. Cada entidad o usuario dispone de una colección de claves públicas a la cual se le asocian dos atributos (de clave válida y nivel de confianza). En este esquema todas las entidades actúan como una de "certificación".
La particularidad de esta solución es que resulta impráctico utilizarla en comunidades grandes como en las que operan los modelos anteriores, pero funciona eficiente y eficazmente en una red de pequeña cantidad de participantes.

Internet X.509 Infraestructura de Clave Pública

El objetivo principal de la definición de este estándar es el diseño e implementación de una PKI o Infraestructura de Clave Pública, es decir, de una arquitectura de certificados digitales. Este hecho es fundamental, ya que es lo que distingue indubitablemente una clave criptográfica de un certificado digital. Este estándar establece diferentes parámetros para identificar al propietario del certificado, a su emisor (la entidad certificante), fechas de emisión, caducidad, etc.
Adicionalmente, este estándar también marca las directivas para la implementación de los mecanismos de validación que permiten a las demás entidades de los certificados comprobar su validez en cualquier momento. Además, considera los conceptos de validación, rutas de certificación y revocación.
Una PKI entonces, puede ser entendida como un conjunto de recursos humanos, de hardware y de software que por medio de la criptografía asimétrica posibilitan el uso de funciones de seguridad. Son cinco los actores primordiales que se encuentran en el core de funcionamiento de esta arquitectura:

- Una Autoridad de Certificación, que emita y revoque certificados digitales, como más adelante se describirá.
- Una Autoridad de Registro, encargada de validar la identidad del usuario que requiere una certificación de su clave pública.
- Una Autoridad de Validación, que puedan responder a las solicitudes de validación de los certificados digitales, verificando la fecha de caducidad y el estado del certificado digital.

- Usuarios o entidades certificadas, que puedan firmar, verificar firmas por medio de las claves privadas correspondientes y cifrar datos mediante las claves contenidas en los certificados digitales.
- Repositorios que almacenan y publican certificados válidos y revocados.

Estructura de datos

En el estándar X.509 se especifican los campos que deben estar presentes en un certificado digital, básicamente el formato que se define en el mismo satisface diferentes requerimientos que fueron surgiendo e impactando en varias modificaciones a lo largo del tiempo.
La tercera versión del formato es el más vigente en la actualidad el cual introdujo un mecanismo genérico de extensión de los certificados. Sus campos son:

- Versión (*version*): versión del formato del certificado empleado (en este caso, versión 3).
- Número de Serie (*serialNumber*): identificador único del certificado, asignado por la autoridad emisora del mismo.
- Algoritmo de firma (*signatureAlgorithm*): identificador del algoritmo empleado por la entidad emisora para firmar el certificado.
- Emisor (*issuer*): nombre de la autoridad/entidad que emitió el certificado.
- Vigencia (*validity*): período durante el cual el certificado se considera válido, salvo revocación.
- Información de clave pública del sujeto (*subjectPublicKeyInfo*): campo utilizado para llevar la clave pública e identificar el algoritmo de la misma.
- Sujeto (*subject*): identidad que se certifica.
- Identificador único del emisor (*issuerUniqueID*): campo opcional que permite reutilizar nombres de emisor.
- Identificador único del sujeto (*subjectUniqueID*): campo opcional que permite reutilizar nombres de sujeto.
- Extensiones (*extensions*): conjunto de campos opcionales que se pueden añadir al certificado, aunque no son imprescindibles.
- Firma (*digitalSignature*): firma del certificado, este campo es el que asegura mediante la firma de la autoridad certificante que es realmente legítima toda la información contenida en los campos: emisor, sujeto, vigencia, y otros campos de interés.

Tanto el campo identificador único del emisor como del sujeto se añadieron en una versión intermedia del formato de este certificado, con el fin de poder reutilizar los nombres de sujeto y/o emisor de certificados durante más tiempo, pero rara vez se utilizan y ya casi están quedando obsoletos. Por otro lado, y dicho anteriormente, en la última versión se le añadió el campo de extensiones para lograr mayor escalabilidad del protocolo.[67]

Ciclo de vida de los certificados

El ciclo de vida de un sistema define todos los procedimientos por el que es sometido el mismo desde el comienzo de su existencia, hasta su fin. Aplicado a este estándar, el mismo abarca diferentes operaciones como: gestión de claves, emisión de certificados, distribución de certificados, renovación de certificados, revocación de certificados y validación.
En la *Ilustración 1* se puede visualizar el ciclo de vida PKI completo.

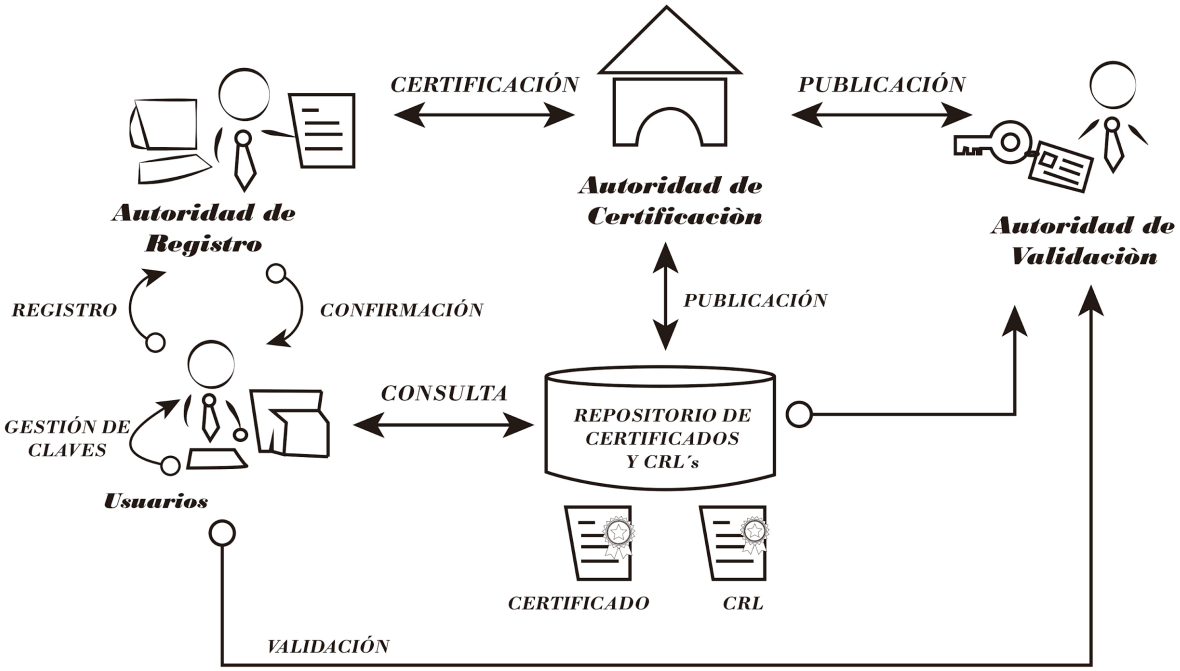

Ilustración 1: Esquema de ciclo de vida PKI

Proceso de gestión de claves

Es el paso inicial o de "etapa cero" antes de comenzar cualquier tipo de actividad en el ciclo de vida de los certificados digitales. Este procedimiento requiere la creación de claves criptográficamente asimétricas utilizando los protocolos estándares más vigentes, los cuales abarcan diferentes características a tener en cuenta a la hora de generar claves de cifrado, como ser: alto grado de aleatoriedad, longitud y entropía.
Dependiendo del uso que se les vaya a dar, la generación del par de claves puede ser llevado a cabo por parte del potencial propietario o por una entidad central, no obstante, en el caso de la utilización de las mismas en contexto de certificados digitales, tanto una como la otra es válida.
Una vez culminada la etapa de generación de claves, se debe abordar una serie de precauciones para proteger la parte privada del par, donde se utilizan diferentes tipos de almacenamiento como ser: módulo criptográfico simétrico de hardware, archivos cifrados simétricamente, o servidores de credenciales accedidos mediante autenticación clave-usuario. Estos mismos a su vez agregan otros factores de autenticación a demás de una clave simétrica y/o un usuario, como por ejemplo: datos biométricos y/o tokens temporales.

Proceso de emisión de certificados

Aquí la entidad certificante o **Autoridad de Certificación** toma un papel primordial. Las **autoridades de certificación** interactúan indirectamente con las **entidades a certificar** (usuarios), estas lo hacen a través de otro actor denominado **Autoridad de Registro**. Estas últimas se encargan de verificar la identidad de la entidad a certificar, mediante un proceso intensivo de presentación y comprobación de diferentes documentos de manera personal. Una vez terminada la etapa de corroboración de identidad, la autoridad acepta o rechaza la **solicitud de registro**. En el caso de ser aceptada, le llega a la **autoridad de certificación** y si la misma cumple con diferentes políticas de certificación y se encuentra en formato adecuado, es **firmada** por medio de un software que contiene la **clave privada** de la **Autoridad Certificante**.
A partir de este momento, el certificado ya se encuentra emitido y es enviado mediante una copia al usuario o entidad certificada y/o a un repositorio público de certificados. La autoridad certificante también podría almacenar una copia a motivo de backup.

Proceso de distribución de certificados

El método utilizado con mucha frecuencia para la distribución de los certificados es el basado en servidores de directorios. Tanto para operaciones de verificación de firmas o de cifrado, el acceso a los certificados se lleva a cabo mediante consultas a servidores de directorios donde los certificados se encuentran disponibles gracias a la autoridad certificante. Estos servidores además proporcionan información extra que eventualmente podría resultar útil, como ser, información de correo electrónico, información laboral, etc.

Proceso de renovación de certificados

Cada certificado emitido lleva consigo el tiempo de validez, que representa su fecha de expiración y se encuentra en el campo “validity” nombrado anteriormente. Tras el paso de este período, el mismo deberá o no ser renovado, y generalmente lo más apropiado es que esta acción esté acompañada de una renovación de las claves generadas al principio del ciclo de vida. Todo este proceso puede ser llevado a cabo de manera transparente al usuario, o también podría requerir del mismo, no obstante, se lo define de manera opcional.

Proceso de revocación de certificados

Así como existe el proceso de emisión de los certificados digitales, debe existir también su operación contraria, para hacer frente a todos los requerimientos que surgen al utilizar este tipo de soluciones.
La decisión de realizar este tipo de operaciones pertenece a la Autoridad de Certificación, y sólo puede ser invocada por la entidad certificada; para concretar el procedimiento, la autoridad certificante debe diseminar esta noticia a la red, con el objetivo de que este quede absolutamente invalidado. Estas invalidaciones de certificados quedan registradas en repositorios denominado CRL´s o **Lista de Certificados Revocados**. En dicha lista, además de los certificados ya no válidos se agrega información acerca del motivo de revocación y fecha a partir del cuando ese certificado deja de ser válido.

Proceso de validación de certificados

A la hora de establecer una comunicación entre dos entidades, se produce el intercambio de certificados digitales con el fin de generar un medio de comunicación seguro; para esto es necesario verificar las firmas de forma tal de poder confiar en el mismo. Por lo tanto, es necesario tener accesible el certificado digital de la autoridad firmante del certificado en cuestión. Este certificado digital puede estar disponible en el cliente utilizado para iniciar la comunicación, o bien en el sistema operativo donde opera el mismo. Para resumir el procedimiento, la entidad **A** se comunica con la entidad **B**, esta última envía su certificado digital a la primera. La entidad **A** obtiene la firma de dicho certificado y busca localmente la existencia el certificado digital de la Autoridad de Certificación que firmo el certificado de **B**. Si en su almacenamiento local encuentra varios certificados que coincidan con el firmante del certificado de **B** (lo que hoy en día es inusual), entonces verifica que coincida con el campo "*issuerUniqueID*" nombrado anteriormente en la estructura de datos del certificado digital. Una vez encontrado el certificado de la autoridad certificante, se verifica la firma del certificado de **B**. Si la firma no coincide, se detiene el proceso y se produce un rechazo el certificado; en su defecto, la firma verifica, se confía en el certificado de la autoridad y es tratado como otro certificado a validar, donde el paso uno es ejecutado recursivamente hasta que se construya un camino a un certificado de confianza o exista un error. Para lograr esto, será necesario tener acceso a todos los certificados digitales requeridos.
Concurrentemente a este procedimiento de validación, existen otros chequeos, como ser que las fechas introducidas en los certificados sean correctas, las extensiones y consultas a las CRL´s para verificar que no haya sido revocado.

Capítulo 5 - Base de Datos Blockchain

Introducción Histórica

Como primer indicio, la historia real de la ***Blockchain*** comienza en la década del 70, cuando las base de datos fueron creadas. En la era del Big Iron, las grandes organizaciones pagaban mucho dinero a IBM, Informix y otras organizaciones por grandes bases de datos donde colocaban todos sus activos de datos más preciados. El lenguaje SQL que alimenta la gran mayoría de los sistemas de gestión de contenidos que circulan por la web, fue originalmente un lenguaje de comandos para unidades de cinta.
Pasaron los años mientras la gente luchaba para incluir el mundo real en las bases de datos mediante la gestión del conocimiento, la web semántica, y muchas otras abstracciones. No todo llegó a funcionar íntegramente, pero de todos modos la realidad se materializaba en la sociedad utilizando estas herramientas. Las cosas que no funcionaban bien en las bases de datos fueron desechadas, y la evolución continuó. De vez en cuando una contracorriente técnica trataba de afianzarse y de replegar la tiranía de las bases de datos, pero la tendencia general se mantuvo firme: si no entra en la base de datos, no existe.
Es imposible pensar que se desconoce de este mundo de las bases de datos, ya que se vive en él. Cada vez que se llena un formulario de papel con los cuadrados que indica una letra por casilla, se está interactuando con una base de datos. Cada vez que se ingresa a un sitio web, hay una base de datos que interoperan por debajo de la superficie. Amazon, Facebook, son todos conjuntos inmensos de bases de datos.
El segundo indicio se llevó a cabo con la llegada de Tim Berners-Lee y la aparición de la web. A finales de los 80 se comenzó con la tendencia de redes de computadoras. Protocolos como Telnet, Gopher, Usenet y Email proporcionaban una interfaz de usuario para las primeras conexiones a Internet, pero no es hasta principios de los 90 que se alcanza la adopción masiva de las computadoras conectadas en red, lo que lleva gradualmente a que actualmente se redacte una tesis en Google Docs, y a que los profesores puedan corregirla e

interactuar por medio de un navegador web desde otra parte del mundo. Este proceso de unir los puntos fue rápido. A principios de los 90, ya existía un gran número de computadoras, pero fueron en gran medida dispositivos independientes, o conectados a unos pocos cientos de computadoras en una universidad. El software y hardware para la creación de redes en todas partes llevó a construir la red de redes, el Internet, y en consiguiente, el avance se extendió astronómicamente. Todavía se está surfeando la ola tecnológica mientras la red se vuelve más y más inteligente, más pequeña, más barata y comienzan a aparecer en elementos como "Internet de las Cosas".
Sin embargo, las bases de datos y las redes nunca llegaron a comprenderse totalmente**.** El Big Iron en las salas de máquinas y las diminutas e innumerables computadoras dispersas en Internet, no encontraron un estándar común que les permitiera interoperar sin problemas. Interactuar con una sola base de datos es bastante sencillo: a través de formularios y aplicaciones web como usualmente se utilizan en todo el mundo. Pero la dificultad es conseguir que las bases de datos trabajen juntas, de manera sincrónica, de forma transparente, y conseguir que las bases de datos interactúen sin problemas con diferentes procesos ejecutándose en múltiples computadoras diseminadas por el mundo.
Esos problemas técnicos son enmascarados por la burocracia, pero se siente su total impacto cada día en la cotidianeidad de los usuarios. Es un trabajo muy complejo conseguir que dos grandes organizaciones trabajen conjuntamente para un bien común, y en el fondo, es un problema de software.
En 1991 surge una primer solución de base de datos basada en una **cadena de bloques** o ***blockchain*** segura, utilizando criptografía que fue evolucionando hasta que en 1998, Wei Dai describe una solución descentralizada, sincronizada y distribuida para pagos electrónicos basada en criptografía de clave pública. Este primer trabajo es evolucionado por otros autores hasta que en 2008 se publica, con el pseudónimo de Satoshi Nakamoto, el artículo que define el mecanismo para implementar una moneda digital: **Bitcoin**. Este se basa en el uso de las **cadenas de bloques** para registrar las transacciones en una red ***peer-to-peer***.
El 3 de enero de 2009 entonces, entra en funcionamiento el **protocolo Bitcoin** con el primer programa de código abierto, y se crean las primeras criptodivisas. A partir de acá el desarrollo de la red de nodos y el uso de **Bitcoin** para realizar pagos sin intermediar con ninguna entidad ni regulador crece hasta el nivel de intensidad actual, con el **Bitcoin** valorado en altas sumas en dólares. En paralelo, fueron apareciendo otras criptomonedas montadas sobre esta solución de base de datos denominada ***blockchain***.

Comprendiendo Blockchain

Existen tres escalones básicos que los ingenieros atraviesan cuando intentan conseguir que las redes de datos y las bases de datos interactúen y trabajen sincronizadamente y con fluidez.
El primer paso consiste en conectar en red directamente dos o más computadoras entre sí, y resolver los problemas a medida que van surgiendo. Es decir, la computadora **A**, conectada con la **B**, la primera emite transacciones a través del medio de conexión y de forma teórica, la

computadora **B** recibe dichas transacciones, las persiste en su base de datos, y el trabajo es finalizado con éxito. En la práctica, hay algunos inconvenientes más.

El problema epistemológico no resulta trivial. Las bases de datos, como comúnmente desplegadas en la totalidad de las organizaciones, almacenan hechos. Si la base de datos dice que el saldo del cliente **C** es de **30 pesos**, esa es la verdad para el conjunto de la organización, excepto tal vez para el contador que tiene contacto físico con el dinero y es capaz de contar la cantidad, encontrando que el saldo real es **29 pesos**, introduciendo ese dato en la base de datos como corrección. La base de datos es la realidad institucional.

Pero cuando los datos dejan una base de datos y desembocan en otra, se cruza una frontera organizacional. Para la Organización **O**, el contenido de la base de datos **D** son la realidad operativa, verdadera hasta que se demuestre lo contrario. Pero para Organización **H**, el comunicado es una declaración de opinión. Si se considera un pago del usuario **A** hacia el **B**, el inicio del pago se materializa en un "pedido de pago", este pedido se efectiviza por medio de una petición, pero no se convierte en un hecho confirmado hasta que el pago se acredite pasado el punto de una devolución del cargo. Hasta ese momento, una señal que dice "error al realizar la transacción" puede reiniciar el proceso entero. El pedido existe en forma de hipótesis hasta que el pago se termine de realizar dejando la transacción irreversiblemente asentada como un registro en un expediente de hechos, es decir, este pedido de pago existió, el pago fue realizado, fue aceptado, y finalizado efectivamente. Pero hasta ese momento, ese pago es sólo una especulación.

El ciclo de vida tornadizo de una simple solicitud de pago fluyendo de una organización a otra, una declaración de intención a una declaración de hecho, no es algo que normalmente se tiene en cuenta.

El otro inconveniente que radica en la comunicación entre estos sistemas de base de datos que funcionan de forma distribuida es la gran inestabilidad. Una diminuta alteración en el software en cualquiera de los extremos ya comienza a desestabilizar el sistema. Pequeños **bugs** que se encuentran imperceptibles hasta que los datos de la transacción perteneciente al pago se hayan almacenado en los registros internos de la base de datos de la organización **H**. Un claro ejemplo: un pedido de pago siempre fue registrado en montos de **1000 pesos**, y se procesa como una caja. Pero por alguna razón, un día se realiza un pago de **1001 pesos**, y en algún momento dentro de la organización **H**, se corrompe la hoja de cálculo que maneja la recepción de pagos. No hay forma de realizar pagos mayores a **1000 pesos**, y el sistema informático se detiene.

A esta inestabilidad se le adiciona otro inconveniente: la necesidad de traducir los lenguajes internos privados de una organización a otra organización.

A través del cable, entre dos organizaciones comerciales, no es posible simplemente observar el código fuente de un tercero para determinar el error. Cada vez que dos organizaciones se encuentran y quieren automatizar sus conexiones de back-end, todos estos desarrollos tienen que ser hechos manualmente por los ingenieros de software. Es difícil, caro, y tan propenso al error que en la práctica se tiende a utilizar métodos "analógicos" como el papel.

Existen diversos intentos de solucionar estos inconvenientes, por ejemplo, para introducir estándares y reutilización del código para ayudar a simplificar estas operaciones y lograr interoperabilidad es posible optar entre EDI, XMI-EDI, JSON, SOAP, XML-RPC, JSON-RPC, WSDL, Rest Full, etc. No obstante, esta diversidad de normativas y estándares genera

una impresión de que ninguna de ellas funciona lo suficientemente bien como para que solo exista una opción.
El último problema que se presenta es el de escalabilidad. Es decir, si el usuario **A** envía un pago de **30 pesos** a **B** y la transacción termina de forma eficiente y eficaz, no siempre es posible lograr el mismo resultado con millones de usuarios interactuando concurrentemente. El costo de la escalabilidad sigue subiendo para cada nuevo usuario que se une a la red.
El segundo escalón presenta la forma de encarar estos problemas anteriores .
Ejemplificando, básicamente se comienza por tomar una organización —**Mastercard**—y todos los usuarios coincidieron en que van a conectarse con **Mastercard** utilizando su interfaz estándar. Cada organización tiene que llegar a un solo conector correcto, y **Mastercard** obtiene el **3%** de comisión, y se asegura de que todas las transacciones finalizan correctamente.
Existen pequeños inconvenientes con este enfoque, que se materializan en centralización y monopolio. El negocio de ser un ente central o una plataforma para otros es, literalmente, una licencia para imprimir dinero para cualquiera que logre dicho status. Poder político a la hora de establecer los términos de servicio y la negociación con los reguladores puede ser ejercida, pero sobre todo un acuerdo que podría haber comenzado con un esfuerzo para crear una columna vertebral neutral rápidamente se convierte en clientes de un ente central sin la cual un usuario simplemente no puede tomar diferentes decisiones.
Este patrón se repite una y otra vez en diferentes industrias, en diferentes niveles de complejidad y escala, desde los transportes públicos, los servicios de telefonía, la intervención quirúrgica a un paciente hasta la gestión financiera de los bancos.
En el mundo de las bases de datos, existe una misma materialización de este problema: la plataforma de la economía. Si este modelo centralizado es que todo el mundo opte por una solución Oracle o Windows Server o algún otro sistema de este tipo, y luego dependan de estas cajas para conectarse entre sí sin problemas, es porque después de todo son iguales, se tiene la misma proposición económica básica de antes: para ser un miembro de la red, confías en un intermediario que cobra lo que quiera por el privilegio de la afiliación de un usuario, con un impuesto disfrazado como coste técnico.
Mastercard obtiene un cierto porcentaje por una fracción muy importante de las transacciones en el mundo con este juego.
La existencia de un protocolo es el tercer paso. Internet se ejecuta con un conjunto de protocolos: estándares HTTP y HTML de Sir Tim Berners Lee han funcionado eficazmente, aunque, por supuesto, él simplemente comenzó la idea que luego llevó a la invención de SMTP, POP e IMAP, o más bien BGP que organiza a los grandes routers de la red de redes.
Estos y más estándares aspiran a apoyar la formación de protocolos, o incluso ser los protocolos, pero la definición semántica de cada campo de actividad tienden a crear una complejidad inherente que conduce de nuevo hacia el sistema centralizado.
Lo que existe por debajo de la **cadena de bloques** es una alternativa para obtener las bases de datos sincronizadas dependiendo de la impresión de hojas de papel. Si se tuviese un billete en formato de papel impreso: Llevar un conjunto de papeles desde un banco a otro, y el valor se mueve de la cuenta de un banco a otra. Computadoras como simulador de papel, una vez más. **Bitcoin** simplemente toma un proceso basado en papel, la representación fundamental

de dinero en efectivo, y lo reemplaza con un sistema digital: dinero digital. En este sentido, se podría ver a **Bitcoin** como otro simulador de papel, pero no lo es.
Resulta complejo asignar valor real a un objeto en un entorno digital, a diferencia de la economía tradicional, donde el valor de un objeto se determina en gran medida por su escasez (ley de oferta y demanda). En consiguiente, un objeto digital es representado por una secuencia de números, no obstante, también puede ser replicado infinidades de veces a un costo cero. Esta característica, desde el punto de vista tradicional, anula el valor. Las copias de programas, películas, imágenes, etc; son un claro ejemplo de esta problemática. Desde el punto de vista de las monedas esto resulta paradigmático, ya que son por definición elementos escasos. Las monedas fueron creadas para realizar intercambios, es decir, se entrega una cantidad de monedas a cambio de una contraprestación (servicio, objeto u otra moneda). En fin, el principal uso es la realización de **transacciones**.
Una moneda en el mundo digital debe toparse con la problemática del **doble gasto**. Cada unidad monetaria puede tener un identificador único, e incluso una **firma digital** de una entidad emisora, pero aún así, seguirá siendo muy sencillo realizar copias y gastar la misma más de una vez.
La solución del sistema tradicional a este problema (como se redactó antes) es utilizar una entidad central que gestione todas las operaciones. Básicamente, es lo que sucede hoy con las entidades bancarias, que llevan registro de las unidades monetarias que poseen sus usuarios así como de las operaciones que realizan, por lo tanto, si se intentase realizar un doble gasto por parte de algún usuario, se detectaría de manera instantánea y anularía la transacción.
Esta propuesta tradicional acarrea consigo algunos efectos secundarios poco deseados, como ser, comprometimiento de la seguridad (arquitectura centralizada), privacidad (datos custodiados por ente central tercero) y dependencia (tanto el usuario como el sistema recaen en la dependencia de la autoridad central coordinadora).
Como solución a estos efectos, basándose en arquitecturas distribuidas, podría construirse una base de datos descentralizada, que permita realizar copias de cada operación en múltiples nodos, de forma que, la caída de un nodo no tenga impactos en el sistema, la confianza no se deposita en una entidad centralizada, se deposita en el protocolo, que debe garantizar el buen funcionamiento del sistema distribuido, y se elimina la dependencia centralizada, ya que un nodo no puede decidir unilateralmente cambios en el sistema.
Sin embargo, una base de datos con arquitectura descentralizada acarrea consigo ciertos aspectos no triviales a tener en cuenta, por ejemplo, la sincronización y latencia (que todos los nodos de la red tengan la misma información a la vez), el doble gasto (validación de la totalidad de las operaciones), y el consenso (al actualizar información de diferentes nodos, pueden darse casos de operaciones contradictorias).
Estos inconvenientes y más son los que motivaron a Satoshi Nakamoto a acuñar esta solución ***blockchain*** o **cadena de bloques**, permitiendo el uso del **bitcoin** como una moneda criptográfica. Sus orígenes se sitúan en el artículo titulado cómo "Bitcoin: A Peer-to-Peer Electronic Cash System"[70].
La novedad que presenta Bitcoin es que se trataba de un protocolo público que implementa una moneda digital mediante criptografía asimétrica basada en una arquitectura ***peer to peer***. La definición de esta misma como una moneda distribuida le viene del hecho de que no existe

un organismo central que regule el valor o la cantidad total de monedas existentes, sino que su configuración recae en la capacidad computacional de la red de usuarios que la opera.
Para evitar la manipulación arbitraria por parte de terceros, **Bitcoin** a través de ***Blockchain*** hace uso de la característica denominada **prueba de trabajo**, que obliga a cada nodo a llevar a cabo una cierta tarea antes de adquirir el privilegio de formular ciertas operaciones. Es esta capacidad de cómputo la que determinará quién puede gestionar el historial de transacciones, añadiendo nuevas a esta **cadena de bloques**, y/o corroborando la validez de todas aquellas otras que se vayan a ir produciendo en el futuro.
Aunque la **cadena de bloques** está íntimamente relacionada con el protocolo **Bitcoin**, es relevante destacar que este sistema es totalmente válido para otro tipo de protocolo orientado a transacciones. De hecho, eso es lo que está realizando desde sus inicios la plataforma Ethereum[71], que tiene su propia cadena de bloques, su propia lógica de negocios, y su propia moneda, denominada **Ether**. A diferencia de **Bitcoin**, las transacciones de Ethereum son los contratos inteligentes, que pueden ser de mayor o menor complejidad y que permiten definir todo tipo de transacciones.
Al igual que ocurre con **bitcoin**, lo bueno de esas transacciones es que se mantendrán en la **cadena de bloques**, inalterables y accesibles durante toda la vida de esa ***blockchain***.

	Esquema Tradicional	Esquema Blockchain
Inconvenientes en la sincronización y persistencia de datos	SI	NO
Inconvenientes para el mantenimiento de la integridad de los datos	SI	NO
Inconvenientes para lograr una interoperabilidad eficiente	SI	NO
Ahorro en costos posteriores a las transacciones, lo que hace más eficiente los procesos de reconciliación de la información con contrapartes, auditores y reguladores.	NO	SI
Los registros distribuidos permiten verificar las transacciones, y la colaboración en distintos nodos asegura su autenticidad.	NO	SI
Las identidades de los usuarios son protegidas criptográficamente, además el sistema es completamente transparente.	NO	SI
Plataforma pública, y cualquier usuario habilitado puede obtener una copia del registro.	NO	SI
Es posible el intercambio entre dos partes sin la intermediación o supervisión de terceros, reduciendo riesgos considerablemente.	NO	SI
Mejor respuesta ante ataques maliciosos, ya que carece de punto central débil, al utilizarse redes descentralizadas.	NO	SI
Los datos están ampliamente disponibles, son exactos, privados, completos y llegan siempre a tiempo, mejorando la integridad de los datos y a un bajo costo.	NO	SI

Los usuarios pueden controlar completamente todas sus transacciones e información.	NO	SI
Los usuarios pueden tener la tranquilidad de que sus transacciones serán ejecutadas exactamente como marque el protocolo, sin necesidad de que supervisen terceros.	NO	SI
Cualquier modificación puede ser vista públicamente por cada parte, asegurando transparencia. Cada transacción es inmutable; no puede ser eliminada o modificada.	NO	SI
Reduce gastos generales y costes intermediarios innecesarios, al requerir menos seguimiento y control.	NO	SI

Tabla 1: Comparación entre la solución tradicional y la utilizando blockchain

Blockchain

La cadena de bloques es una lista creada de forma colectiva con todas las transacciones que han sido confirmadas y validadas por la propia red mediante la inclusión de transacciones en bloques y de estos últimos en la cadena.

Esta cadena se encuentra diseminada en la red alojada en la totalidad de los usuarios/no dos que la integran. Cuando un nodo de la red consigue crear un nuevo bloque, lo transmite al resto de los nodos, estos lo verifican y en caso de ser válido lo agregan a su cadena local y lo difunden, en su defecto lo rechazan. Mediante la difusión del nuevo bloque, éste se acabará añadiendo de la forma en que se ilustra en la *Ilustración 2* siempre y cuando no se haya creado otra rama en la cadena de bloques en la que haya participado una cantidad de usuarios con más capacidad de cómputo (*Ilustración 3*).

Por definición, de la blockchain se puede extraer el historial de posesión de todas las monedas, siguiendo la lista de transacciones. Así, un usuario no puede reutilizar monedas que ya utilizó, ya que la propia red rechazará la transacción.

No obstante, puede darse el caso de que haya monedas reutilizadas de manera no malintencionada, por ejemplo, por fallos de comunicación masivos, como caídas de la red, o cuando se crean varias grandes ramas en la cadena de bloques, conteniendo cada una aproximadamente la mitad de la potencia de cálculo del sistema, pero siempre el sistema termina convergiendo. Por esto es buena práctica esperar un tiempo determinado para confirmar una transacción, es decir, esperar que la transacción se añada en un bloque y luego de ese, se agreguen más bloques a la cadena.

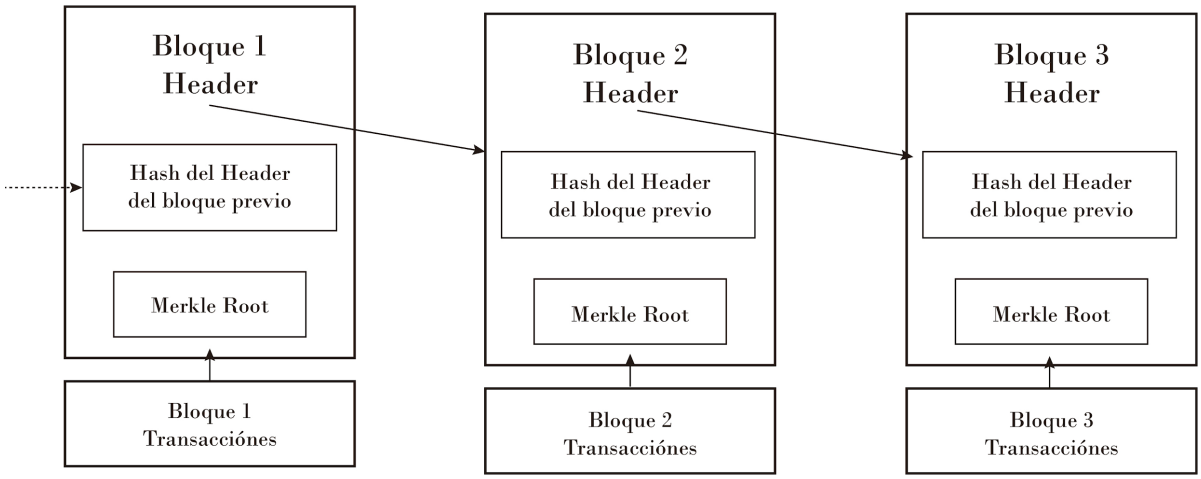

Ilustración 2: Representación Resumida de la Blockchain

La cadena de bloque es mantenida colaborativamente por pares anónimos en la red, por lo que se requiere que cada bloque demuestre que se invirtió una cantidad significativa de trabajo en su creación para asegurar que los pares no confiables que quieren modificar los bloques grandes tengan que trabajar más duro que los compañeros honestos que solo desea agregar nuevos bloques a la cadena de bloques.

Encadenar bloques juntos hace que sea imposible modificar las transacciones incluidas en cualquier bloque sin modificar todos los bloques siguientes . Como resultado, el costo de modificar un bloque en particular aumenta con cada nuevo bloque agregado a la cadena de bloques, aumentando el efecto de la **prueba de trabajo**.

La **prueba del trabajo** utilizado aprovecha la naturaleza aparentemente aleatoria de los hashes criptográficos. Para demostrar que se realizó un trabajo adicional para crear un bloque, se debe crear un hash del encabezado del bloque que no exceda un cierto valor (complejidad). Por ejemplo, si el máximo valor hash posible es [(2^256) - 1], puede probar que intentó hasta dos combinaciones produciendo un valor de hash inferior a 2^255 .

En el ejemplo anterior, producirá un hash exitoso en promedio cada dos intentos. Incluso se puede estimar la probabilidad de que un intento de hash determinado genere un número por debajo del umbral objetivo. **Blockchain** asume una probabilidad lineal de que cuanto más bajo sea el umbral objetivo, más intentos de hash (en promedio) necesitarán probarse.

Los nuevos bloques solo se agregaran a la cadena de bloque si su hash es al menos tan desafiante como el valor de dificultad esperado por el protocolo de consenso. Cada 2.016 bloques, la red utiliza marcas de tiempo almacenadas en cada encabezado de bloque para calcular el número de segundos transcurridos entre la generación del primero y el último de esos últimos 2.016 bloques. El valor ideal es 1.209.600 segundos (dos semanas).

- Si se necesitaron menos de dos semanas para generar los 2.016 bloques, el valor de dificultad esperado se incrementa proporcionalmente (hasta en un 300%), por lo que los siguientes 2.016 bloques deberían tomar exactamente dos semanas para generar si los hashes se revisan a la misma velocidad.
- Si llevó más de dos semanas generar los bloques, el valor de dificultad esperado se reduce proporcionalmente (hasta en un 75%) por la misma razón.

Debido a que cada encabezado de bloque debe hashear a un valor inferior al umbral objetivo, y que cada bloque está vinculado al bloque que lo precedió, se requiere (en promedio) tanta potencia de hashing para propagar un bloque modificado como toda la red de Blockchain gastada entre el vez que se creó el bloque original y la hora actual. Solo si adquirió la mayor parte del poder de hashing de la red podría ejecutar de manera confiable un ataque del 51 por ciento contra el historial de transacciones (aunque debe tenerse en cuenta que incluso menos del 50% de la potencia de hash aún tiene buenas posibilidades de realizar tales ataques).
El encabezado del bloque proporciona varios campos fáciles de modificar, como un campo **nonce** dedicado, por lo que la obtención de nuevos valores hash no requiere la espera de nuevas transacciones. Además, solo el encabezado del bloque de 80 bytes es hasheado para la prueba de trabajo, por lo que incluir un gran volumen de datos de transacción en un bloque no ralentiza el hashing con E / S adicionales, y agregar datos de transacciones adicionales solo requiere el recálculo de los hashes ancestrales en el **árbol de merkle**.
Cualquier nodo de la red que acierta con éxito un encabezado de bloque a un valor por debajo del umbral objetivo puede agregar todo el bloque a la cadena de bloques (suponiendo que el bloque sea válido). Estos bloques son comúnmente atendidos por su altura de bloque -el número de bloques entre ellos y el primer bloque de la cadena (bloque 0, más comúnmente conocido como el bloque de génesis). Por ejemplo, el bloque 2016 es donde la dificultad podría haberse ajustado primero.

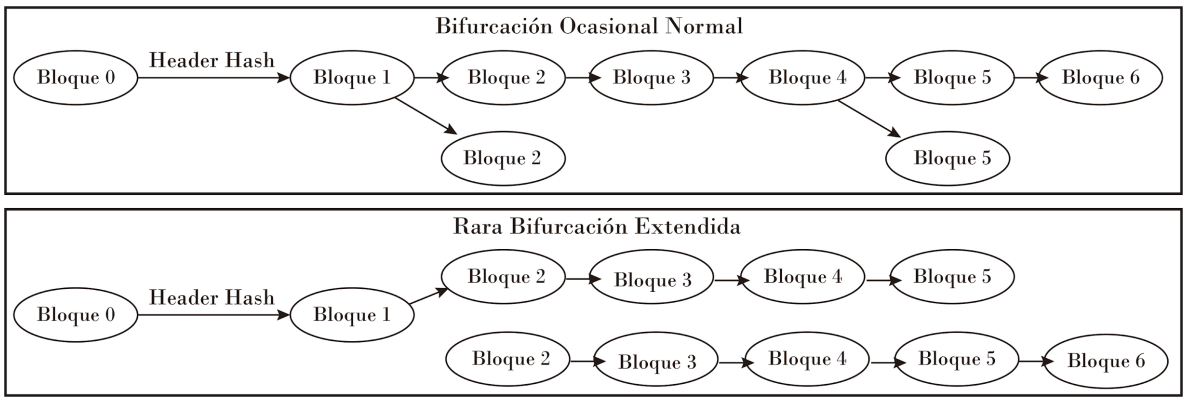

Ilustración 3: Bifurcación de la Cadena de Bloques

Múltiples bloques pueden tener la misma altura de bloque, como es común cuando dos o más mineros producen cada bloque aproximadamente al mismo tiempo. Esto crea una horquilla aparente en la cadena de bloques, como se muestra en la ilustración anterior.
Cuando los nodos que producen bloques, crean bloques simultáneos al final de la cadena, cada nodo elige individualmente qué bloque aceptar. A falta de otras consideraciones, que se analizan a continuación, los nodos generalmente usan el primer bloque que ven.
Eventualmente, un nodo produce otro bloque que se conecta a solo uno de los bloques competitivos de minados simultáneos. Esto hace que ese lado de la horquilla sea más fuerte que el otro lado. Suponiendo que una horquilla solo contenga bloques válidos, los pares normales siempre siguen la cadena más difícil para recrear y tirar los bloques obsoletos que pertenecen a las horquillas más cortas. (Los bloques obsoletos también se denominan a veces

huérfanos, pero esos términos también se usan para bloques huérfanos verdaderos sin un bloque primario conocido).
Las bifurcaciones a largo plazo son posibles si diferentes mineros trabajan con objetivos cruzados, como algunos mineros que trabajan para extender la cadena de bloques al mismo tiempo que otros mineros intentan un ataque del 51 por ciento para revisar el historial de transacciones.
Dado que múltiples bloques pueden tener la misma altura durante una horquilla de la cadena de bloques, la altura del bloque no se debe usar como un identificador global único. No obstante, los bloques se referencian generalmente por el hash de su encabezado (a menudo con el orden de bytes invertido y en hexadecimal).

Bitcoin

Como antes dicho, la novedad y éxito de esta tecnología radica en su forma de funcionamiento distribuido y sin una autoridad central que regule la emisión de moneda, o acepte o deniegue transacciones. Básicamente son los nodos pertenecientes a la red los que implícitamente toman estas decisiones de forma "democrática".
Por medio de conceptos que serán expuestos a continuación, es posible lograr entender el paradigma a través de los siguientes ejemplos:

- Los usuarios reciben bitcoins a modo recompensa por haber colaborado con la red (más adelante se verá cómo es que esto se lleva a cabo). Hasta acá, puede parecer que los usuarios podrían engañar al sistema para aumentar su recompensa pero, por construcción del sistema, la mayoría de los usuarios tendrán que validar posteriormente esa recompensa. Así, si el usuario la aumentase de forma oculta, esa acción sería rechazada por el resto.
- Un usuario **A** realiza una transferencia de **bitcoins** a **B**. Para evitar que posteriormente **A** vuelva a utilizar esos mismos **bitcoins** para pagar a un tercer usuario **C**, en **Bitcoin**, las transacciones se hacen públicas, no obstante, cuando el resto de la red detecte esta segunda transacción inválida, la rechazará, imposibilitando dicha reutilización de **bitcoins** por parte del usuario **A**.

Como se aprecia en estos ejemplos, son los mismos usuarios los que toman las decisiones que normalmente corresponden a una única autoridad central. Esto hace que **Bitcoin** sea una moneda "democrática". Como en cualquier democracia, su evolución se adapta a lo que la mayoría de la población quiere. Por consiguiente, en este caso no hay una equivalencia de "un usuario = un voto", ya que el peso de cada usuario depende de la potencia de cómputo que éste dedica a la red. Así, la ecuación anterior en Bitcoin, sería más bien "x% de cómputo = x% de votos". Por lo tanto, siempre y cuando más de un 50% de la potencia de cómputo de la red sea controlada por usuarios honestos, la red seguirá la evolución que estos decidan. La idea puede contemplarse como una "democracia ponderada" en función de la implicación en el sistema.

El papel de la criptografía

Las soluciones de alta complejidad como ser la implementación de estas criptomonedas, generalmente se respaldan por un conjunto de primitivas avanzadas.
Las funciones criptográficas de las que **Bitcoin** hace uso, son responsables principales de que se consigan las propiedades de seguridad ésta tecnología persigue. El aspecto más importante de la criptografía que le compete a esta criptomoneda es la **asimétrica** o de clave pública y sus capacidades de **firma digital**.
Bitcoin implementa el algoritmo ECDSA[23] para firmar digitalmente las transacciones, utilizando los parámetros recomendados por el Standards for Efficient Cryptography Group (SECG), secp256k1[22]. Las firmas utilizan la codificación DER[24] para empaquetar sus componentes en un único flujo de bytes.
ECDSA ofrece ventajas frente a otros esquemas de firma que lo hacen ideal para su utilización en un protocolo distribuido en Internet, como son: longitudes de clave, de firmas muy cortas, y generación y verificación de firmas muy rápidas.
En los cálculos de hashes realizados en **Bitcoin** se utilizan los estándares SHA-256[25] y, cuando se requiere que el hash sea más corto, RIPEMD-160[26]. Generalmente el cálculo de hashes se realiza en dos fases: la primera con SHA-256 y la segunda, dependiendo de las necesidades de longitud del resultado, con SHA-256 o RIPEMD-160.
La generación de números aleatorios es esencial para la criptografía y más en la aplicada a **Bitcoin**. Los ***nonces*** o números aleatorios que sólo se utilizan una vez son utilizados de forma directa para la generación de bloques en la ***blockchain*** de Bitcoin que se verá más adelante.
Las **pruebas de trabajo,** el principal componente de ***Blockchain*** que garantiza que la red tenga un comportamiento legítimo. Básicamente, esta idea hace que validar/calcular nuevos bloques de transacciones conlleve un coste computacional muy elevado, de forma que, para hacerse con el control de la red (y por tanto de qué se valida y qué no), un atacante necesitaría una potencia de cómputo extremadamente difícil de conseguir.
En síntesis, por medio de ***Blockchain***, en **Bitcoin** este control de complejidad en los cálculos para los nuevos bloques se realiza obligando a que el hash de cada nuevo bloque deba comenzar con un número determinado de ceros. Como se verá más adelante, para el cálculo de este hash se combinan datos de bloques anteriores y un nonce. Dado que las funciones hash criptográficas no son invertibles, para encontrar un bloque válido la única alternativa será ir obteniendo diferentes nonce hasta encontrar uno que cumpla el requisito preestablecido.

Contexto Operativo

Con el objetivo de contextualizar el medio operativo del sistema, es necesario describir los actores que interactúan en el mismo y la forma de cómo se efectúan las transacciones.
En cuanto a los actores que actúan en el sistema, se clasifican en tres tipos:

- **Nodos Normales**: realizan compras y pagos de bienes y servicios utilizando como moneda a los bitcoins, produciendo transacciones en el sistema.
- **Nodos Mineros**: son nodos normales que aparte dedican potencia de cómputo para validar nuevas transacciones, creando lo que se conoce como bloques de transacciones. Este rol, como antes se nombró, recibe recompensas en bitcoins por haber colaborado y proporcionado poder computacional a la red.

Un usuario, equivalente a un nodo, se identifica en el sistema por medio de una o más **direcciones Bitcoin**. Esta misma, se representa como una dirección virtual similar a una cuenta bancaria y se materializa como una clave pública de criptografía asimétrica. Estas direcciones pertenecientes a un usuario se almacenan y gestionan en un **monedero** virtual, que equivale a un monedero físico.

Una **transacción** es una transferencia de dinero de un **usuario** hacia otro, incluso a sí mismo, es decir, representa a la asignación de bitcoins de una **dirección Bitcoin** a otra. La constitución de una **transacción**, consiste en que si un usuario **A** transfiere dinero a uno **B**, el **usuario** de la **dirección Bitcoin** asignante (**usuario A**) firme una transcripción de la **dirección Bitcoin** del **usuario B** con la clave privada asociada a la dirección del **usuario A**, de esta forma se determinará que el nuevo propietario de esos bitcoins transferidos es la **dirección Bitcoin** del **usuario B**.

Estas transacciones, una vez estén pendientes a confirmar, se agrupan a su vez en un conjunto de transacciones pertenecientes a un **bloque,** el cual será sometido posteriormente al proceso de minería y luego adherido a la **cadena de bloques** o **blockchain**.

La **blockchain,** representa al core del funcionamiento de esta criptomoneda. Es básicamente un registro público de las transacciones validadas en orden cronológico, es decir, cuando un bloque ya fue confirmado por medio del proceso de minería, éste pasa a formar parte de este registro público. La **cadena de bloques,** presenta una característica peculiar que es la de público acceso, distribuida, descentralizada y de sólo modo lectura; esto lo logra gracias a su constitución de **Árbol de Merkle**[22].

Arquitectura de comunicaciones del sistema

Los nodos que integran la arquitectura de **Bitcoin** constituyen un sistema de comunicaciones **P2P** o ***peer-to-peer***. Como ya se ha mencionado, la filosofía aquí es evitar la existencia de autoridades centrales que controlan la red.

Como toda arquitectura **P2P**, **Bitcoin** dispone de una serie de mecanismos para descubrir nuevos nodos en la red, y mantener una lista actualizada de los mismos. Además, distintos clientes de **Bitcoin** pueden también ofrecer mecanismos adicionales, como por ejemplo mensajes de tipo **addr** y **getaddr**, mediante los cuales un cliente envía (o solicita) a otro un listado de clientes actualmente conectados a la red. También, en el código de los clientes se suele incluir un listado de nodos semilla, que se utilizarán para iniciar el proceso de conexión a la red en caso de que el resto de mecanismos fallen.

Además de los mecanismos para descubrir otros nodos en la red, hay otros tipos de mensajes de uso frecuente en **Bitcoin**. Por ejemplo, los mensajes **tx** y **block**, utilizados para enviar

datos de transacciones y bloques, respectivamente, de manera que los nodos de la red puedan mantener la sincronía requerida por el protocolo. O los mensajes de tipo **inv**, que se utilizan para anunciar (y retransmitir) nuevas transacciones[29].

Estructuras de datos del sistema

Como se describió con anterioridad, unos de los elementos que hace posible el funcionamiento de **Bitcoin** es la **criptografía asimétrica**. En ella, los distintos algoritmos funcionan a partir de una clave compuesta por dos elementos relacionados de modo que son fácilmente computables en una dirección (cifrado, descifrado y verificación de una firma digital) pero difícilmente computables en la contraria si se desconoce de la información secreta.

Direcciones y monederos

Una dirección **Bitcoin** convencional (P2PKH) es simplemente una cadena de texto codificada en **Base58Check** que tiene hasta 20 bytes de longitud y que consiste en el **hash** de la **clave pública** asociada con la dirección (*Ilustración 4*). Este formato es similar al **Base64**, con la diferencia que no solo pretende mantener la información codificada lo más legible y telecable posible para el usuario, sino que también permite verificar de forma más eficiente si una cadena arbitraria que satisfaga dicha expresión se corresponde con una dirección real o no, aplicando un mecanismo de validación redundante que ya se emplea en los números de tarjeta de crédito o documentos de identidad.

```
Versión = 1 byte de ceros
HashDeClave = Version + RIPEMD-160(SHA-256(ClavePública))
Checksum = SHA-256(SHA-256(HashDeClave))
DirecciónBitcoin = Base58Encode(HashDeClave + Checksum)
```

Ilustración 4: Formato de Dirección Bitcoin

Las direcciones cumplen con las siguientes características:

- Generación en tiempo computacionalmente reducido (milisegundos).
- La clave privada asociada a dicha dirección debe ser un problema computacionalmente complejo, con el fin de ofrecer garantías de que un tercero no logre generar una clave privada asociada a dicha dirección.
- Generación offline, con el fin de proporcionar una capa de seguridad a dicha creación.

Un usuario podría crear una o varias direcciones, no obstante, un conjunto de dichas direcciones constituye un **monedero** Bitcoin, mediante los cuales se realizan las transacciones que se verán a continuación.

Transacciones

Con el fin de representar el flujo de las criptomonedas en la red existe el concepto de transacciones.
Las transacciones en Bitcoin son estructuras de datos firmadas digitalmente que cambian el propietario de unidades de bitcoins asignándolas a otra dirección o propietario.
La estructura de datos de una transacción (*Ilustración 5*) se encuentra formada por **entradas**, **salidas**, **hash de transacción**, **firma digital del emisor**, **clave pública del emisor**, **total entradas**, **total salidas**, **bloqueo**, **versión:**

- Entradas: registros que referencian los fondos de transacciones previas. Las mismas se encuentran firmadas digitalmente por el "pagador" (proceso necesario y suficiente para desbloquear los fondos transferidos). Campo de tamaño variable.
- Salidas: registros que determinan el nuevo o los nuevos propietarios de las bitcoins transferidas. Estas salidas se utilizan como entradas de transacciones próximas. Campo de tamaño variable.
- Hash de transacción: resumen de toda la estructura de datos.
- Firma digital del emisor: encripción del **hash de la transacción** con la clave privada del emisor.
- Clave pública del emisor: se añade dicha para que se pueda verificar la firma digital cuando la transacción llegue a un nodo de la red que la deba procesar.
- Total entrada y salidas: número que indica cantidad de entrada y salidas adheridas a la transacción. Cada uno de estos campos puede contener entre 1 y 9 bytes.
- Versión: posee 4 bytes e indica el número de versión Bitcoin utilizado para esa transacción.
- Bloqueo: indica la fecha mínima en la cual dicha transacción puede ser agregada a la **cadena de bloques**. Si el valor indicado en este campo está entre cero y quinientos (incluidos ambos) indica la cantidad de bloques que deben agregarse a la cadena de bloques antes de agregar esta transacción; y si indica un valor mayor a quinientos, entonces se interpreta como una fecha límite en formato UNIX.En una transacción de transferencia de bitcoins, se deben utilizar todas las que se encuentran asignadas a la dirección origen.

Por ejemplo, si A posee 10 bitcoins y desea enviar sólo 5 bitcoins a B, pues entonces las salidas de la transacción van a ser 5 bitcoins para la dirección de B y 5 bitcoins para la dirección de A; donde esta última toma el rol de una "dirección de devolución". Por consiguiente, en una transacción siempre se "gastan" todos los bitcoins asignados.

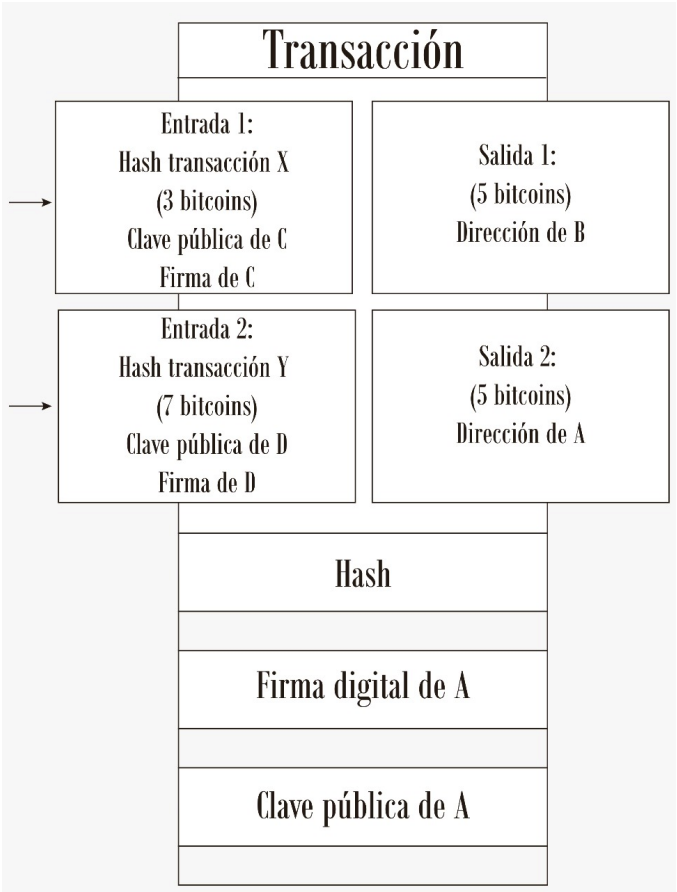

Ilustración 5: Esquema de transacción Bitcoin

Como se visualiza en la *Ilustración 6* el emisor A posee 10 bitcoins que antes los obtuvo por medio de dos transacciones diferentes (la transacción X y la transacción Y), donde tanto la "entrada 1" como la "entrada 2" antes fueron salidas de otras transacciones como la ilustrada (en este caso una transacción Y generada por D hacia A y otra transacción Y generada por D hacia A). Finalmente, A se queda con 5 bitcoins y B con los otros 5, los cuales podrán utilizar estas salidas para generar nuevas entradas en próximas transacciones.

La suma de la totalidad de las entradas debe ser igual o mayor que la suma de la totalidad de las salidas. En el caso de que la cantidad de bitcoins de la entrada sea mayor que la de la salida, la diferencia se considera una "comisión", y quien incluya esa transacción en la **cadena de bloques** o ***blockchain*** (base de datos distribuida y descentralizada) puede disponer de esa cantidad. Esta recompensa es una manera de motivar a los nodos **mineros**, que obtienen beneficios por su trabajo en forma de bitcoins. Las transacciones que poseen "comisiones" tienen prioridad por los nodos mineros al momento de elegir cual de ellas procesar primero, y en consecuencia, las transacciones que posean mayor monto en comisiones serán procesadas de forma más veloz en la red.

Cada **salida** y **entrada**, al igual que las transacciones, tienen su estructura interna. Como ya se vio, las **entradas** son referencias o "punteros" a **salidas** anteriores, es decir, cada **entrada** hace referencia a un identificador perteneciente a una **salida** (UTXO, Salida de Transacción Sin gastar) que se encuentra almacenada en la base de datos distribuida. Para gastar una

UTXO, la **entrada** de la transacción también incluye una condición de desbloqueo que satisface la condición especificada por la **UTXO**. Este código de desbloqueo normalmente consta de una firma la cual prueba la posesión de la dirección que se encuentra especificada en el código de bloqueo de la **UTXO**.

Las **entradas** están compuestas por los siguientes campos:

- Hash de transacción: puntero a la transacción que posee la **salida** perteneciente a esta **entrada**. Posee 32 bytes.
- Índice de la **salida**: índice de la **salida** perteneciente a esta **entrada**, es decir, el índice de la **UTXO** que se quiere gastar. Tiene 4 bytes.
- Tamaño del código de desbloqueo: especifica el tamaño en bytes que tiene el código de desbloqueo. De 1 a 9 bytes.
- Código de desbloqueo: el cual cumple las condiciones del código de bloqueo de la **UTXO**. Tamaño variable.

Las **salidas** están compuestas por:

- Monto: cantidad de bitcoins que se desean transferir. Posee 8 bytes.
- Tamaño del código de bloqueo: tamaño en bytes. De 1 a 9 bytes.
- Código de bloqueo: el cual define las condiciones que se deben de cumplir para poder gastar el monto. Generalmente, el código perteneciente a este campo realiza una transferencia de bitcoin a una dirección parametrizable. Tamaño variable.

Cada transacción crea salidas, las cuales son almacenadas en la base de datos distribuidas. Todas las salidas (excepto una) crean **UTXOs** las cuales son reconocidas por toda la red y están disponibles para que el poseedor haga uso de las mismas.

En síntesis, la transferencia de un monto en bitcoins es básica y sencillamente crear una **UTXO** asignada a la dirección bitcoin de destino.


```
"tx": [
    {
      "hash": "4a5e1e4baab89f3a32518a88c31bc87f618f76673e2cc77ab2127b7afdeda33b",
      "ver": 1,
      "vin_sz": 1,
      "vout_sz": 1,
      "lock_time": 0,
      "size": 204,
      "in": [
        {
          "prev_out": {
            "hash": "0000000000000000000000000000000000000000000000000000000000000000",
            "n": 4294967295
          },
          "coinbase": "04ffff001d0104455468652054696d65732030332f4a616e2f32303039204368616e63656c
            6c6f72206f6e206272696e6b206f66207365636f6e64206261696c6f757420666f722062616e6b73"
        }
      ],
      "out": [
        {
          "value": "50.00000000",
          "scriptPubKey":
            "04678afdb0fe5548271967f1a67130b7105cd6a828e03909a67962e0ea1f61deb649f6bc3f4cef38c4f3
            5504e51ec112de5c384df7ba0b8d578a4c702b6bf11d5f OP_CHECKSIG"
        }
      ]
    }
  ]
```

Ilustración 6: Esquema de Transacción en formato JSON

Transacción de Coinbase

Todas las transacciones en la red bitcoin no se crean por igual. Una transacción de coinbase es un tipo único de transacción de bitcoin que solo puede ser creada por un minero. Este tipo de transacción no tiene entradas, y hay una creada con cada bloque nuevo que se extrae en la red. En otras palabras, esta es la transacción que recompensa a un nodo minero con la recompensa en bitcoins por su trabajo. Cualquier tarifa de transacción cobrada por el minero también se envía en esta transacción.
La transacción coinbase es, en la mayoría de los casos, la primera transacción en un nuevo bloque. El destinatario de la transacción de coinbase puede elegir tener la recompensa del bloque, y las tasas de transacción enviadas a una dirección de bitcoin, o las bitcoins pueden enviarse a una multitud de direcciones diferentes. En este sentido, es como cualquier otra transacción en la red.

Códigos de Bloqueo y Desbloqueo

Script es el lenguaje usado para describir los **códigos** de Bitcoin, utilizados para bloquear/desbloquear de las transacciones. Cualquier combinación de un **script** o código de desbloqueo y uno de bloqueo que finalice con un valor de **VERDADERO** indica que la condición de la **UTXO** se cumple y por tanto es válido, en cualquier otro caso es inválido.
Cuando se valida una transacción, los scripts de entrada se concatenan con los scripts de salida y se evalúan. Una buena analogía de cómo funciona esto es que los scripts de salida son rompecabezas que especifican en qué condiciones se pueden gastar esos **bitcoins**. Los scripts de entrada proporcionan los datos correctos para que los scripts de salida se evalúen como verdaderos.
Por ejemplo si se tiene el código de bloqueo **3 OP_ADD 4 OP_EQUAL** éste se puede satisfacer con el script de desbloqueo **1** [33].
El lenguaje tiene las siguientes características:

- Simple
- Limitado. El lenguaje no es Turing completo debido a que no tienen ciclos ni controles de flujo complejos lo cual asegura que siempre termine. Por esta razón no es posible tener bombas lógicas que ocasionen un ataque de denegación de servicio en la red Bitcoin.
- Requiere un procesamiento mínimo.
- Puede ser implementado en una amplia gama de dispositivos.
- No hay un estado anterior o posterior a la ejecución del script. Toda la información necesaria para ejecutar el código debe estar contenida en él.
- Es un lenguaje que accede a la memoria basándose en una pila. Por tanto no hay variables. Para hacer más claro, el uso de la pila han decidido usar notación polaca

inversa. Los valores se van metiendo en la pila y los operadores meten o sacan uno o más parámetros de la pila, modifican los parámetros y finalmente pueden meter un resultado en la pila. Por ejemplo la instrucción **OP_ADD** saca dos elementos de la pila, los suma y mete el resultado en la pila. Al final del código, la cima de la pila es el valor de retorno. **Script** puede usar dos pilas: La principal y la alternativa. La alternativa se usa para almacenar datos de cálculos de pasos intermedios de forma similar a la tecla memoria de las calculadoras.

- Su funcionamiento es similar al del lenguaje ensamblador ejecutado sobre una CPU little-endian con un solo registro de memoria de 16 bits. Bitcoin implementa su procesador virtual para interpretar el código máquina **Script**.

En bitcoin existen 5 tipos de combinación códigos de bloqueo/desbloqueo (pay-to-public-key-hash, pay-to-pubkey, pay-to-multisig, pay-to-script-hash y data output) que son las únicas aceptadas por el cliente de referencia y la mayoría de los nodos mineros. Estas combinaciones pertenecen a tipos de transacciones aceptadas en el **protocolo Bitcoin**. Aunque es posible crear códigos de bloqueo/desbloqueo y transacciones que no son estándar, se tiene que encontrar un nodo minero que no siga estas limitaciones para que mine la transacción en un bloque con scripts o códigos no estándares.

Pay-To-Pubkey-Hash (p2pkh)

Pago a hash de clave pública es el script de salida de transacción más comúnmente utilizado. Se usa para pagar a una dirección de bitcoin (como visto antes, una dirección de bitcoin es un hash de clave pública codificado en base58check).

Script de Bloqueo:

scriptPubKey: OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG

Script de Desbloqueo:

scriptSig: <sig> <pubKey>

*Nota: **scriptSig** está en la entrada de la transacción de gasto y **scriptPubKey** está en la salida de la transacción.*

Así es como se procesa cada palabra:

Pila	Código	Descripción
Vacía.	<sig> <pubKey> OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG	scriptSig y scriptPubKey se combinan.

<pubKey> <sig>	OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG	Las constantes se agregan a la pila.
<pubKey> <pubKey> <sig>	OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG	El elemento superior de pila se duplica.
<pubHashA> <pubKey> <sig>	<pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG	El elemento superior de la pila se hashea.
<pubKeyHash> <pubHashA> <pubKey> <sig>	OP_EQUALVERIFY OP_CHECKSIG	Se agrega la otra constante.
<pubKey> <sig>	OP_CHECKSIG	La igualdad se verifica entre los dos elementos superiores de la pila.
VERDADERO	Vacío.	La firma se verifica para los dos elementos superiores de la pila.

Explicación detallada:

El remitente **A** solo tiene la dirección de Bitcoin del destinatario **B**, entonces, ¿cómo obtiene el **pubKeyHash** de su dirección de Bitcoin?.
La clave es que el remitente **A** no necesita obtener el **pubKeyHash** de "su" dirección de Bitcoin.
El script **scriptPubKey** está sucediendo primero. **A** crea este script con la dirección de bitcoin de **B** en lugar de su <pubKeyHash>. Ese es el trabajo de **A** hecho, y lo que **A** ha dicho es "1 BTC ahora pertenece a **B** pero, si y sólo si **B** puede demostrar que es el verdadero propietario de la dirección de bitcoin agregada en <pubKeyHash>".
B, en su billetera, ve que aparece esta cantidad de 1 BTC. Entonces técnicamente **B** "lo posee". Pero para que **B** lo gaste, es decir, enviárselo a otra persona, **B** necesita probar que la dirección de bitcoin a la que le dio el BTC a **A** era de hecho suya. Acá es donde aparece **scriptSig**. Entonces, la <sig> <pubKey> es responsabilidad de **B** quien conoce todos los parámetros para completar la misma. No obstante, **A** antes de haber enviado el BTC a **B** tuvo

que haber realizado el mismo procedimiento.

Pay-To-Pubkey (p2pk)

Los scripts de pago a clave pública son una forma simplificada de p2pkh, pero ya no se utilizan comúnmente en nuevas transacciones, porque los scripts p2pkh son más seguros (la clave pública no se revela hasta que se gasta el resultado).

Script de Bloqueo:
scriptPubKey: <pubKey> OP_CHECKSIG
Script de Desbloqueo:
scriptSig: <sig>

Procesamiento:

Pila	Código	Descripción
Vacío.	<sig> <pubKey> OP_CHECKSIG	scriptSig y scriptPubKey se combinan.
<pubKey> <sig>	OP_CHECKSIG	Las constantes se agregan a la pila.
cierto	Vacío.	La firma se verifica para los dos elementos superiores de la pila.

Pay-To-Multisig (p2ms)

Las salidas multigrado permiten compartir el control de bitcoins entre varios destinatarios. Al crear el script, uno especifica las claves públicas que controlan los fondos, y cuántas de esas claves se requieren para firmar transacciones de gasto para que sean válidas. Una salida con N claves públicas de las cuales se requieren M se denomina salida m-of-n (por ejemplo, 2 de 3, 3 de 5, 4 de 4, etc.)

Script de Bloqueo:
scriptPubKey: <m> <A pubkey> [B pubkey] [C pubkey...] <n> OP_CHECKMULTISIG
Script de Desbloqueo:
scriptSig: OP_0 <A sig> [B sig] [C sig...]

Pay-To-Script-Hash (p2sh)

Las salidas de hash de pago a script son scripts que contienen el hash de otro script llamado ***redeemScript***. Para gastar bitcoins enviados en un resultado p2sh, la transacción de gastos debe proporcionar un script que coincida con el hash del script y los datos que hacen que el script se evalúe como verdadero. Esto permite postergar la revelación de las condiciones de gasto al momento del gasto. También hace posible que el receptor establezca las condiciones para gastar esos bitcoins.

Script de Bloqueo:
scriptPubKey: OP_HASH160 <Hash160(redeemScript)> OP_EQUAL
Script de Desbloqueo:
scriptSig: <sig> [sig] [sig...] <redeemScript>

Data Output

Las salidas de datos se utilizan para insertar datos en la cadena de bloques. Pueden enviarse hasta 40 bytes de manera estándar, pero se pueden usar más datos si un minero decide aceptar la transacción.

scriptPubKey: OP_RETURN <0 a 40 bytes de datos>
(No pueden ser gastados, ya que no posee **scriptSig**)

El Satoshi

Como norma general, el **bitcoin** se divide en 8 partes u 8 dígitos decimales, es decir que si alguien posee 1 bitcoin (BTC) en realidad tiene 1.00000000 BTC, cuando se posee de medio bitcoin o cuarto citcoin, es cuando se tienen fracciones de bitcoins, siendo para **medio bitcoin** = 0.50000000 BTC y para **cuarto bitcoin** = 0.25000000 BTC.
Existen fracciones aún más pequeñas que se pueden manejar en el mundo Bitcoin, las mayormente usadas son estas tres: mBTC, uBTC y **Satoshis**, pero también existe el cBTC.
El cBTC simboliza el poco usado **centibitcoin**, que es lo mismo que: 0.01BTC = 0.01000000 BTC, es poco usada por los usuarios comunes, pero si muy usada por la comunidad que compran y venden la divisa, ya que no se enfocan mucho en las micro fracciones de esta moneda.
Los mBTC, o también llamada **millibitcoin**, usa la letra "m" como indicativo de 00.000, y que se debe agregar 00000 (5 ceros) al número que esté delante de la letra "m", por ejemplo:

- 1mBTC = 0.00100000 BTC.
- 15mBTC = 0.01500000 BTC.
- 100mBTC = 0.10000000 BTC.

El uBTC, o denominado también **microbitcoin**, y cualquier número que haya delante de la letra "u" tendrá 00 (2 ceros) seguidos, por ejemplo:

- 1uBTC = 0.00000100 BTC.
- 15uBTC = 0.00001500 BTC.
- 100uBTC = 0.00010000 BTC.

Por último, el **Satoshi**, la fracción más utilizada por los usuarios bitcoin, su nombre proviene de su creador y es la unidad de BTC más pequeña que se puede manipular (enviar o recibir), para entender fracciones Satoshis, simplemente se debe contar siete ceros luego de la coma, es decir, 1 Satoshi es igual a: 0.00000001 BTC.

Bloques

Los bloques son estructuras de datos que contienen un conjunto de transacciones ya confirmadas. Cada cierto tiempo (aproximadamente cada diez minutos), un nuevo bloque que incluye y nuevas transacciones se anexan a la cadena de bloques por medio del proceso de minería.

hash	Hash del bloque
ver	Versión del bloque
prev_block	Hash del bloque anterior
mrkl_root	Hash de la raíz del árbol de Merkle
time	Marca el tiempo de creación del bloque
bits	Especificación de la complejidad del bloque
nonce	Nonce que resuelve la prueba de trabajo
size	Número de bytes que siguen, hasta el final de bloque
n_tx	Número de transacciones en la siguiente lista
tx	Lista de transacciones contenidas en el bloque

Ilustración 7: Esquema de Bloque perteneciente a la Blockchain

De los campos expuestos en la *Ilustración 7*, el campo **versión**, **hash prev block**, **hash merkle root**, **time**, **bits** y **nonce**, pertenecen al header del bloque.
Los campos que comienzan con la palabra "**hash**" del esquema anterior tienen como fin establecer la cadena de bloques.
En el campo **Bits**, se define cual es la complejidad requerida en el momento de generación del bloque para que dicho bloque fue válido. Esta complejidad es variable en función de la

capacidad de cómputo total de la red, de forma que cada bloque que se genere cada diez minutos.

El campo **Nonce**, es el número que se resuelve en la prueba de trabajo. Básicamente es un campo que se va variando, y en cada variación calcula el hash del bloque hasta conseguir un hash compatible con la complejidad indicada en el campo **Bits**. Ejemplificando, si el campo **Bits** esta en 4, significa que el hash del bloque debería estar compuesto por 4 ceros al principio, no obstante, para lograr esto se deben calcular sucesivos hashes de este bloque modificando el campo **Nonce,** con el fin de obtener diferentes resultados, hasta llegar a un hash que contenga 4 ceros al comienzo.

Con el fin de optimizar el espacio en disco necesario para almacenar todos los bloques de la cadena de bloques, las transacciones que se incluyen en cada bloque se organizan en forma de árbol de Merkle.

En **Bitcoin**, el primer bloque perteneciente a la gran cadena de bloques se lo denomina **bloque génesis** (*Ilustración 8*) y cuya recompensa por resolverlo fue de 50 bitcoins y el valor encontrado dentro del campo **hash prev block** es “0”.


```
{
  "hash":"000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f",
  "ver":1,
  "prev_block":"0000000000000000000000000000000000000000000000000000000000000000",
  "mrkl_root":"4a5e1e4baab89f3a32518a88c31bc87f618f76673e2cc77ab2127b7afdeda33b",
  "time":1231006505,
  "bits":486604799,
  "nonce":2083236893,
  "n_tx":1,
  "size":285,
  "tx":[
    {
      "hash":"4a5e1e4baab89f3a32518a88c31bc87f618f76673e2cc77ab2127b7afdeda33b",
      "ver":1,
      "vin_sz":1,
      "vout_sz":1,
      "lock_time":0,
      "size":204,
      "in":[
        {
          "prev_out":{
            "hash":"0000000000000000000000000000000000000000000000000000000000000000",
            "n":4294967295
          },
          "coinbase":"04ffff001d0104455468652054696d65732030332f4a616e2f32303039204368616e63656c6
            c6f72206f6e206272696e6b206f66207365636f6e64206261696c6f757420666f722062616e6b73"
        }
      ],
      "out":[
        {
          "value":"50.00000000",
          "scriptPubKey":"04678afdb0fe5548271967f1a67130b7105cd6a828e03909a67962e0ea1f61deb649f6b
            c3f4cef38c4f35504e51ec112de5c384df7ba0b8d578a4c702b6bf11d5f OP_CHECKSIG"
        }
      ]
    }
  ],
  "mrkl_tree":[
    "4a5e1e4baab89f3a32518a88c31bc87f618f76673e2cc77ab2127b7afdeda33b"
  ]
}
```


Ilustración 8: Bloque Génesis en formato JSON

Minado de Bloques

El proceso de minado consta de la creación de nuevos bloques para la cadena, y es una tarea muy costosa desde el punto de vista computacional.
Los campos del bloque que llevan un papel importante en este proceso son: **ver**, **prev_block**, **mrkl_root**, **time**, **bits** y **nonce**.
Donde en el campo **mrkl_root,** se encuentra detallado el contenido del árbol de Merkle, que permite verificar que el bloque está correctamente incluido en la cadena.
El **hash** del bloque se calcula utilizando los campos de la cabecera, es decir, los nombrados anteriormente. Para determinar qué hashes serán considerados válidos, se utilizará el campo **bits**, en el cual se encuentra una versión codificada del máximo valor que puede tomar el hash del bloque para ser considerado válido. Para obtener dicho límite, hay que convertir el valor del contenido en hexadecimal y aplicar la siguiente función:
Valor máximo = HEX(bits)*2*(8*(0x19 - 3))
No obstante, cualquier hash con un valor inferior a dicho número hexadecimal, será válido.
Todos estos campos involucrados en la creación del hash son elementos fijos, excepto el **nonce**. Así, este es otro campo que hace que un hash sea válido o no, y es el que los mineros tienen que ir variando hasta encontrar un hash válido. En contexto, cuanto menor sea el límite, más difícil es encontrar un nonce válido (porque se reducen los números que satisfacen la fórmula). Este procedimiento de búsqueda del hash indicado se denomina **prueba de trabajo**.

Recompensas

Dado el proceso anterior una tarea computacionalmente tediosa, el nodo minero que encuentra un nuevo bloque recibe una recompensa.
Dicha recompensa puede percibirse por dos medios. Por un lado, Bitcoin tiene establecido un límite máximo de 21 millones de bitcoins y hasta que se llegue a ese límite, la generación de cada nuevo bloque es recompensada con una cantidad predefinida de bitcoins nuevas. Por ejemplo hasta el 2012, se recompensaba con 50 bitcoins a cada nuevo bloque. Desde entonces, la recompensa es de 25, hasta el 2016 que se redujo a la mitad, es decir 12,5 bitcoins, y así sucesivamente.
Por otro lado, para mantener una motivación similar para los mineros pese al decrecimiento de las recompensas (que potencialmente llegará a cero), existen las tasas de transacción, mediante las cuales, los usuarios pagan una comisión, como ya se describió anteriormente. Este dinero en bitcoins obtenidos de comisión sólo pueden ser gastados si y sólo si existieron 100 bloques por detrás del creado.

Aunque una transacción nueva haya sido incluida en un bloque y dicho bloque en la cadena, inicialmente puede ser posible que esa modificación sea revertida. Esto podría pasar cuando se crean dos ramas inicialmente válidas, lo cual puede ocurrir por diversos motivos. Al generarse dos ramas distintas, cada una de ellas será respaldada inicialmente por una cantidad determinada de mineros, que irán extendiéndola. Cuanto más similares sean las capacidades de cómputo de las ramas, más se tardará en resolver la ambigüedad, aunque eventualmente una de las ramas recibirá un nuevo bloque antes que la otra y prevalecerá sobre ella y la descartará. No obstante, esto es un caso posible y dar por válida una transacción no respaldada por nuevos bloques no es correcto. Por esto, es aconsejable esperar un número determinado de bloques hasta considerar una transacción como confirmada. El número de bloques puede variar dependiendo de la cantidad involucrada en la transacción, y obviamente, en función de las consideraciones personales. Normalmente, se considera que tras 10 bloques nuevos, la transacción será difícilmente revertida y, por tanto, se puede considerar confirmada.
Nótese también que la probabilidad de revertir una transacción decrece exponencialmente por cada nuevo bloque que la respalda.

Consideraciones Generales

En los modelos tradicionales, la confianza se deposita completamente en una autoridad o entidad que controla toda la información. En Bitcoin por medio de Blockchain, por el contrario, no existe dicha autoridad, si no que la información es gestionada por todos los usuarios. De esta forma, siempre y cuando la "mitad más uno" los usuarios del sistema sean honestos, las "políticas" establecidas por el sistema no podrán saltarse por ninguno de los usuarios deshonestos.
El uso de un sistema criptográfico asimétrico fuerte, como es ECDSA, y de algoritmos de hashing robustos, como SHA-256, garantiza la integridad actual del sistema. Pero teniendo en cuenta que la capacidad de cómputo aumenta considerablemente año tras año, además de producirse nuevos avances en la teoría criptográfica/criptoanalítica, el sistema está diseñado de forma que se pueda modificar el sistema criptográfico a utilizar, utilizando el mismo protocolo de comunicación entre pares y de gestión de transacciones.

Debilidades de la Arquitectura Blockchain

Las debilidades que el modelo Blockchain/Bitcoin presenta son:

Más del cincuenta por ciento

Si la mitad más uno de los nodos se hace del total del poder de procesamiento de la red, tendrán un mero control de las nuevas transacciones y bloques a minar [87].

Privacidad

En la blockchain estándar se encuentra la información de todas las transacciones en texto claro, no obstante, se podría realizar implementaciones acudiendo a la confidencialidad por medio de algún tipo de criptografía. En esta tesis se utilizó la blockchain estándar, es decir, la utilizada por Bitcoin[42].

Almacenamiento

En casi 10 años de uso masivo con aproximadamente 10 mil nodos en la red, la *blockchain* de Bitcoin ya llega a los 100 GB. En el caso de la solución propuesta, la estructura de datos de las transacciones es más simple, pero a medida que pase el tiempo va ir requiriendo cada vez más espacio, pero también la tecnología va evolucionando. Dicho esto, podrían también implementarse nodos livianos que almacenen únicamente cierta información y no toda la cadena de bloques (al igual que Bitcoin) [42].

Capítulo 6 - Solución Propuesta

Introducción

El concepto de firma digital, como antes visto, fue introducido por Diffie y Hellman en 1976. Básicamente una firma digital se plantea como un conjunto de datos asociados a un mensaje que permite asegurar la identidad del firmante y la integridad del mensaje.
En 1978 R. Rivest, A. Shamir, y L. Adleman, del MIT proponen el algoritmo hasta hoy más usado para lograr certificaciones digitales, denominado RSA.
Finalmente, en 1988 comenzando conjuntamente con el estándar x.500 se adopta por parte las industrias el estándar x.509 que consiste en la aplicación e integración de los conceptos anteriores implementados con el nombre de Infraestructura de Clave Pública, donde el funcionamiento principal es gestionado de forma centralizada por entidades privadas o públicas que custodian el ciclo de vida de los certificados digitales de la totalidad de los usuarios.
En 2009, Satoshi Nakamoto por medio de su publicación Bitcoin, logra romper con una arquitectura centralizada ya impuesta en todo el mundo (escenario similar a la arquitectura de certificados digitales vigente), proponiendo una solución para operar sin autoridades centrales (bancos o entidades financieras); la gestión de las transacciones y la emisión de moneda electrónica con esta solución es llevada a cabo de forma colectiva por la red. Este modelo innovador combina varios conceptos desarrollados por las propuestas iniciales de la arquitectura de funcionamiento del dinero electrónico, para lograr un sistema completamente descentralizado.
Las redes P2P presentan características que las convierten en un activo para el que resulta difícil encontrar comparación en el mundo real. En primer lugar, porque las conforman sistemas distribuidos y vivos que no presentan un único punto de fallo y que toleran la desconexión de algunos de ellos de forma flexible y sin dejar que el funcionamiento de la red en su conjunto se vea comprometida. Se trata de organismos cuya robustez radica

precisamente en el número de nodos que las componen y en cómo estos están conectados unos a otros.
La materialización de las criptomonedas como fenómeno contemporáneo tiene un trasfondo ideológico muy ligado a conceptos que forman parte de la cultura *hacker* tradicional: evitar la hasta ahora necesaria presencia de un organismo central de control financiero que es considerado como poco democrático y potencialmente corrompible. Aunque las implicaciones de una economía desregulada y el reparto de divisas son materias controvertidas, la consecución de estos objetivos requiere, desde un punto de vista técnico, la colaboración de distintas entidades que asuman aunadas ese rol de organismo central multifacético. Si a su naturaleza descentralizada se le suma que los nodos que forman parte de la red no tienen por qué ser conocidos entre sí y que, aun así, tienen que ser capaces de seguir funcionando de forma consensuada incluso en un escenario en el que hay que dar por supuesta la presencia de agentes no confiables, permite toparse con un escenario real del conocido *problema de los generales bizantinos de tolerancia a fallos* (Byzantine Fault Tolerance o BFT).
El libro mayor público de sólo lectura y distribuido ***blockchain*** (basado en una arquitectura de comunicaciones P2P), fue creado, para registrar, organizar y sostener la más importante criptomoneda, el Bitcoin, pero eso no quiere decir que esta tecnología pueda utilizarse sólo para ello. De hecho, ahora es cuando la ***blockchain*** tiene casi ilimitadas puertas por abrir: si ella es capaz de registrar digitalmente, de forma segura y pública a la vez todo el ciclo de vida de cada bitcoin; en teoría, también puede registrar todo el ciclo de vida de cualquier otra cosa (desde historias clínicas de un hospital, hasta oro, plata, etc).
En contraste a esto, y polarizando los mismos conceptos, la idea de descartar la estructura jerárquica en la que se basa hoy en día la infraestructura PKI, logra no sólo una total transparencia en la emisión de certificados sino también se disipa el riesgo y aumenta la confiabilidad, ya que no existe una entidad central de gestión.
La propuesta de esta tesis plantea una reingeniería y rediseño de la Arquitectura de Certificados Digitales hacia un modelo descentralizado y distribuido utilizando ***blockchain*** o la **cadena de bloques** como mecanismo de validación, sincronización y administración del ciclo de vida de esta arquitectura. Para lograr tal efecto, se aborda la construcción de un prototipo de software en donde se implementen y apliquen los conceptos diseñados y documentados en la descripción de la reingeniería planteada.

Otras aplicaciones utilizando blockchain

- *Spell of Genesis* es uno de los videojuegos que se basa en la ***blockchain***, permitiendo a sus jugadores intercambiar a través de la base de datos, de forma segura y rentable, sus propias cartas y sus propias ganancias en la criptomoneda del juego, el *BitCrystal*.***[34]***
- *El Centro de Estudios Sociales y Tecnológicos*, el primer instituto superior en registrar sus diplomas en ***blockchain***. De este modo, se asegura que los títulos sean auténticos y no modificables, sin necesidad de recurrir a intermediarios.***[35]***

- *UjoMusic*, cuya misión es lograr altos niveles de transparencia y rentabilidad en la industria de la música. Para ello, actualmente tienen como prototipo la nueva canción de Imogen Heap, Tiny Human, de la cual los usuarios pueden explorar todo su recorrido: desde sus políticas asociadas a cómo los pagos son automáticamente distribuidos a los diferentes colaboradores.***[36]***

Otras aplicaciones similares a la Propuesta en cuestión

- *Garman*, utiliza ***blockchain*** para implementar lo que ellos llaman "credenciales anónimas". Este proyecto expone la idea de emplear un registro público para la emisión y publicación de credenciales utilizando un protocolo muy similar a Bitcoin llamado Namecoin como alternativa al actual sistema de nombres de dominio (DNS).***[37]***
- *Fromknecht*, se da la propuesta de utilización de Namecoin para realizar funciones de **infraestructura de clave pública** para distribución de credenciales orientado a correo electrónico.***[38]***
- *Let's Encrypt*, es una iniciativa Open Source con apoyo de grandes corporaciones como Google, Mozilla, Akamai y Facebook. Este proyecto mantiene la misma arquitectura de certificados digitales que la existente en PKI, es decir jerárquica y centralizada, pero con la diferencia que su objetivo es proporcionar una Autoridad Certificante 100% gratuita, fácil de usar (instalación sencilla, renovaciones de certificados automáticos, implementación en la mayoría de los navegadores) y transparente (todos los certificados emitidos y revocados se encuentran registrados públicamente).***[39]***
- *Blockcert*, aplicación basada en ***blockchain*** orientada a la firma de documentos para la creación, emisión y control los certificados de credenciales académicas, certificaciones profesionales, desarrollo de mano de obra, y registros civiles.***[40]***
- *Convergence SSL,* es una solución que permite a un usuario optar por confiar en varios usuarios, la mayoría de los cuales avalan los mismos sitios. Si los usuarios no están de acuerdo sobre si la identidad de un sitio es correcta, el usuario puede optar por la mayoría de los votos o pecar de cauteloso y exigir que todos los usuarios estén de acuerdo o conformarse con un solo usuario (el método de votación está controlado con una configuración en el complemento del navegador). Este sistema posee a grandes rasgos una arquitectura similar a la de PGP.***[89]***

Diferencias entre la solución propuesta y las existentes

Ninguna de las aplicaciones existentes reúne todas las características que presenta la propuesta de esta tesis, es decir, una arquitectura descentralizada, distribuida, transparente,

que elimina las autoridades de certificación, que disminuye los tiempos de administración, aplicado a conexiones Web HTTPS, totalmente gratuita y montada sobre ***blockchain.***

Debilidades de la Arquitectura de Certificados Digitales actual

- Autoridades Centrales o Autoridades de certificación cobran sumas muy caras en la emisión de certificados.
- Los certificados raíz que se encuentran almacenados en los navegadores o sistemas operativos dificultan la creación de nuevos servicios de arquitectura de certificados digitales, los ingenieros de software son obligados a utilizar HTTPS debido a que de esta forma los navegadores o sistemas operativos confían en sus aplicaciones por medio de la preinstalación de los certificados digitales.
- El proceso de emisión de certificados digitales lleva mucho tiempo y consta de bastante burocracia.
- Suficientes ataques efectivos documentados.
- Los procesos de revocación fuera de línea no son viables ante los niveles de conectividad de la actualidad.
- El almacenamiento de los certificados digitales raíces en los navegadores y dispositivos forman parte de un potencial incidente de seguridad, ya que los usuarios confían totalmente en las llaves públicas raíces almacenadas, y no obstante, confían también en todos los certificados que se firmen con las mismas, abstrayéndose del nombre con el que se emitan.
- Arquitectura centralizada además de proveer poca transparencia, se presenta como potencial vector de ataque.

Reingeniería de Arquitectura Propuesta

La solución propuesta fue desarrollar un componente de software a modo de prototipo, que tiene como producto montar una lógica de negocios propia de gestión de certificados que descansa sobre ***blockchain***.
Esta unión de conocimientos e integración de funcionalidades toma las mejores características de cada solución tecnológica, incluyendo de la arquitectura de **certificados digitales** vigente, para no solo poder ofrecer una solución innovadora en términos de descentralización, distribución, seguridad y escalabilidad, sino también para abordar los requerimientos, problemas y necesidades del siglo XXI con conocimientos y herramientas del siglo XXI, y experiencias del siglo XX.
Desde un punto de vista abstracto, el sistema está compuesto por seis módulos de software (*Ilustración 9*). El módulo número uno (denominado **PkChain**) se encarga de la columna vertebral de esta propuesta, es decir, realiza una simulación del funcionamiento de ***blockchain*** que permite utilizar su "API" para luego incorporar las siguientes funcionalidades.

El segundo módulo es una implementación de la arquitectura de comunicaciones peer-to-peer (denominado **Gestor de Servicios**) para así poder tener este tipo de conectividad entre los diferentes nodos ***blockchain*** como propone el protocolo de Bitcoin.
El tercer componente de software agrega a la **cadena de bloques** la inteligencia propuesta en esta tesis. Esta inteligencia tendría el rol de un plugin/librería para la **PkChain**, y se denomina **LibCerts**.
El cuarto módulo es un **Cliente PkChain**, que básicamente consume la red ***blockchain*** como un nodo de la red.
El último módulo, con el fin de lograr el ciclo de vida completo de certificados, implementa un **Cliente Web** que realiza consultas a un **Servidor Web**.
Este prototipo de software propuesto contempla todas las tareas que implementa el ciclo de vida de la certificación digital, es decir, petición, emisión, revocación, renovación y proceso de validación de los certificados digitales. Así mismo, compatibilizando con el estándar x.509 de certificados digitales y con el protocolo HTTPS.
La arquitectura final de esta solución presenta una total descentralización y responsabilidad distribuida entre todos los nodos de la red, eliminando cualquier entidad o autoridad que concentre la gestión de los certificados digitales. Todos los nodos de la red funcionan como clientes y servidores (P2P) los cuales poseen de forma local una copia de la información de todo el sistema (la *blockchain*). La transparencia que brinda esta solución integral proporciona una total facilidad a la hora de auditar o controlar cualquier flujo transaccional.
A continuación se describe el funcionamiento de cada operación, secuencias y procedimientos involucrados en esta propuesta, no obstante, los detalles de implementación de la misma estarán disponibles en el **ANEXO I**.

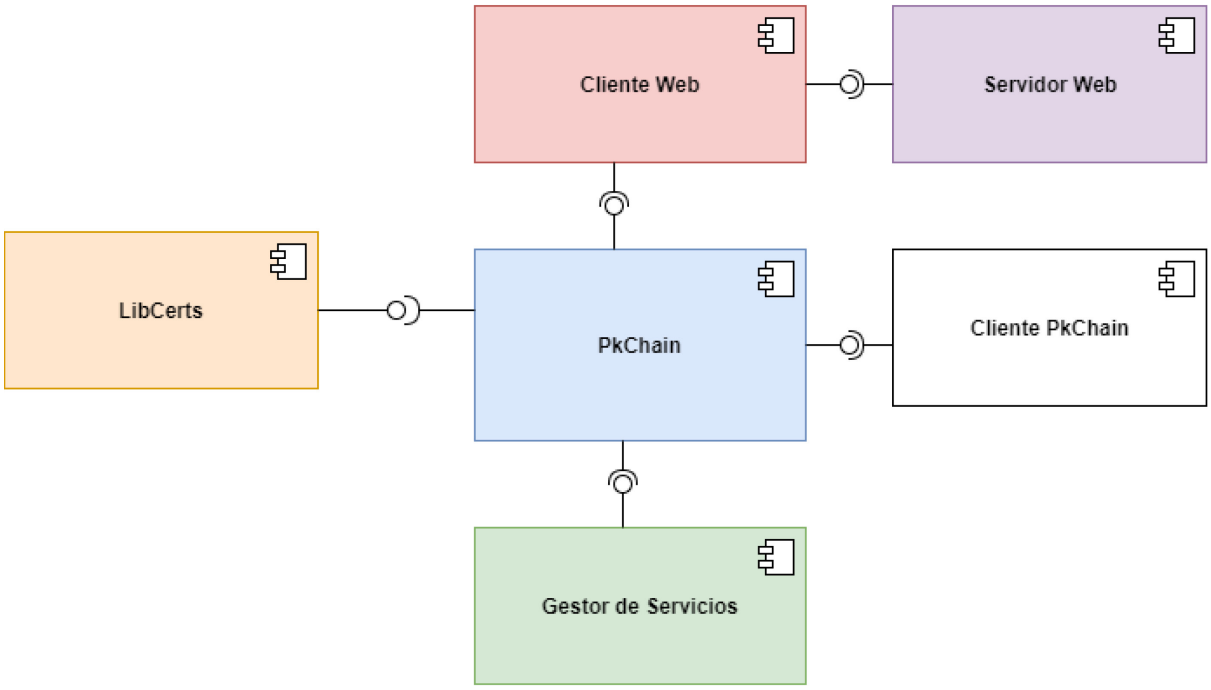

Ilustración 9:Diagrama de Componentes de la Propuesta

Procesos y Operaciones de la Arquitectura Propuesta

Las operaciones y procesos que la arquitectura contempla son: gestión de claves, emisión, distribución, revocación, renovación y validación.

Proceso de gestión de claves

Este proceso queda tal cual se describió en el Capítulo 4 (Arquitectura de Certificados Digitales), ya que bajo cuestiones de gestión/generación de claves la solución se encuentra compatible e interoperable a la actual, y desde el punto de vista técnico no se detectaron contramedidas en esta etapa, no obstante, se decidió no modificar el mismo.

Proceso de emisión de certificados

Al igual que la arquitectura PKI vigente, este proceso se encarga de aceptar la petición de emisión de certificados digitales de los diferentes usuarios.
En consiguiente, una vez generado el par de claves, el portador de las mismas expone la clave pública en formato Base58 Check en el servidor web, en la dirección "URL-RAÍZ/emit/pk", por ejemplo: "www.prueba-clave-publica.com/emit/pk". Es decir, si un usuario realiza una petición web GET de ese recurso, obtendría la clave pública expuesta anteriormente en formato Base58 Check.
Ya generado esto, se crea una **transacción** de emisión de certificados que lleva dentro:

- Como **dirección:** el hash de la misma.
- La **clave pública** del servidor (clave pública a dar de alta).
- El **dominio** portador de la clave pública (URL-RAÍZ).
- La ruta donde buscarla (/emit/pk).

Luego se disemina esta transacción por la red *peer to peer* y el portador de la clave pública aguarda su confirmación.
Todos los nodos que reciban dicha transacción, verificarán su integridad y realizarán una petición web GET hacia el recurso expuesto en la misma. Ya recibida la respuesta de la petición, se compara el resultado con la clave pública del servidor contenida dentro de la transacción y en caso afirmativo se retransmite a los demás nodos y se encola para realizar el proceso de minado, en su defecto la misma se descarta.
Una vez minada y confirmada la transacción (al igual que en Bitcoin, para la confirmación se aconsejan 10 bloques por delante del bloque en donde se encuentra la transacción), el creador de la transacción debe generar un certificado x.509 autofirmado o firmado por una Autoridad Certificante propia, desconocida por los demás o incluso conocida. Este certificado contendrá los diferentes campos correspondientes al estándar y a la configuración que se desee, salvo la particularidad de que en el campo opcional ***issuerUniqueID*** (Identificador único del emisor, visto anteriormente en el Capítulo 4) se dejará asentado el identificador de la **transacción** confirmada. Ya a esta altura el usuario se encuentra en condiciones de agregarlo a la

configuración HTTPS del servidor con su dominio en cuestión y/o firmar otros certificados digitales.
En la *Ilustración 10* se puede apreciar el diagrama de actividades de este procedimiento, y más adelante se describe en detalle la transacción generada por este proceso.

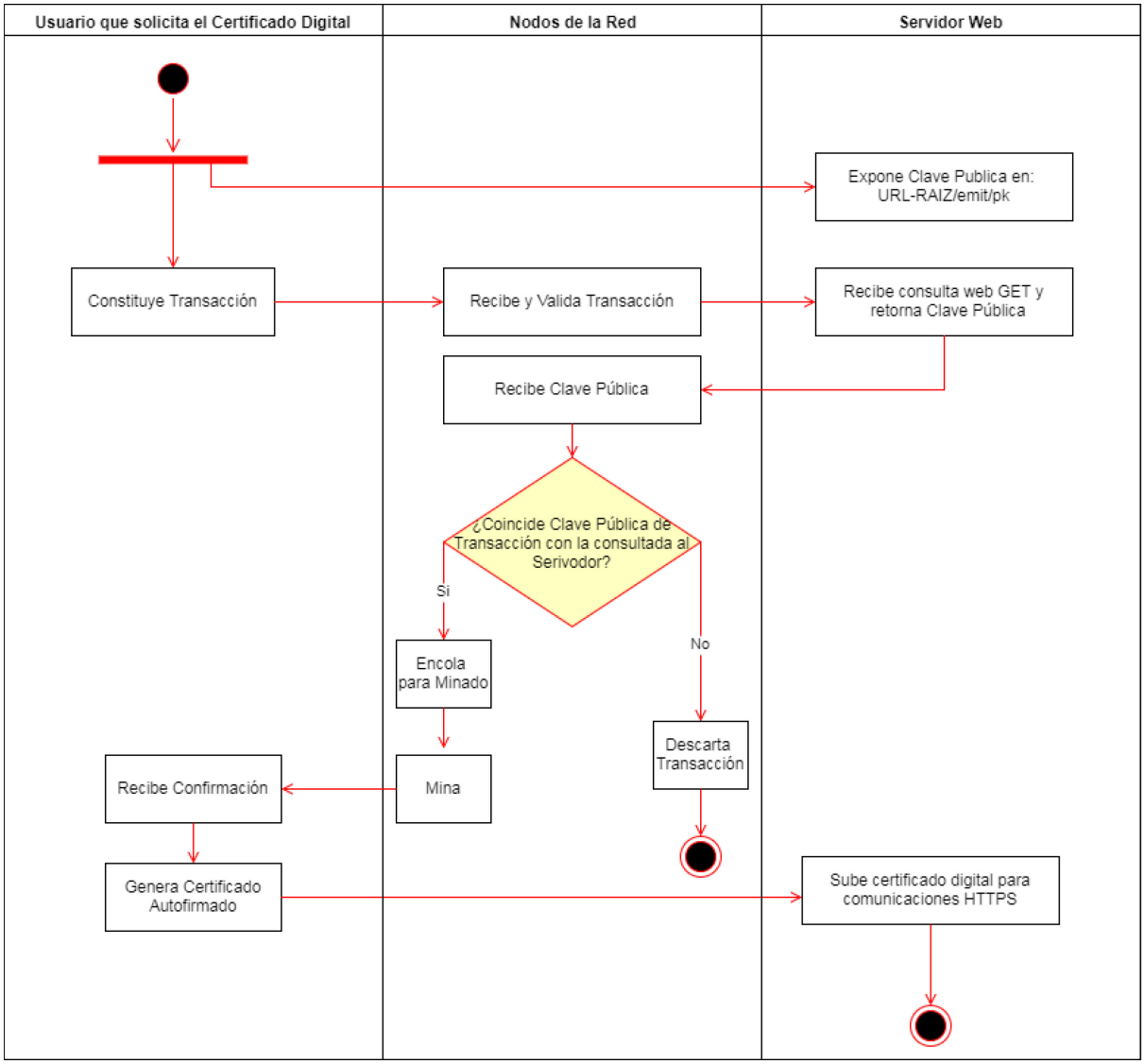

Ilustración 10:Diagrama de Actividades del Proceso de Emisión

Proceso de distribución de certificados

El método utilizado para la distribución de los certificados es el basado ***blockchain***, es decir, distribuido en la totalidad de los nodos en la red. Tanto para operaciones de verificación de firmas o de cifrado, el acceso a los certificados se lleva a cabo mediante consultas a la **cadena de bloques** donde las claves públicas emitidas, renovadas o revocadas se encuentran disponibles gracias a la convergencia y consenso de la red. Esta base de datos pública además proporciona información extra que resulta útil a la hora de realizar el proceso de validación,

como ser, información del dominio en donde validar las mismas. Esto es debido a que (como visto en el proceso anterior) cada usuario que desea emitir, renovar o revocar un certificado digital debe agregar su clave pública a modo de recurso web, donde cada nodo de la red accede a la misma para poder validarla y converger en la emisión, renovación o revocación del mismo.

Proceso de revocación de certificados

Este proceso se lleva a cabo con el fin de dar de baja o declararlo como no confiable a un certificado digital, por consiguiente y al igual que el proceso de emisión, al momento de generar una revocación, el usuario deberá exponer la clave a revocar en el servidor web para que sea accesible por los nodos que la validan, luego generar una transacción y diseminarse por la red.
Dicho ésto, el usuario portador de la clave pública del servidor expone la misma en "URL-RAÍZ/revoke/pk" y genera una transacción con las siguientes características:

- Como **dirección:** el hash de la misma.
- La **clave pública** del servidor (clave pública a revocar).
- El **dominio** portador de la clave pública.
- La ruta donde buscarla (/revoke/pk)
- La **dirección** de la transacción de emisión o de renovación (transacción anterior).

Todos los nodos que reciban dicha transacción, verificarán su integridad y realizarán una petición web GET hacia el recurso expuesto (por medio de la ruta proporcionada en la transacción de revocación o en la indicada en la transacción de emisión o renovación anterior). Ya recibida la respuesta de la petición, se compara el resultado con la clave pública del servidor contenida dentro de la **transacción de emisión o renovación** y en caso afirmativo se retransmite a los demás nodos y se encola para realizar el proceso de minado, en su defecto la misma se descarta.
Una vez minada y confirmada la transacción (al igual que en Bitcoin, para la confirmación se aconsejan 10 bloques por delante del bloque en donde se encuentra la transacción), el creador de la transacción ya da la clave por revocada.
En la *Ilustración 11* se puede apreciar el diagrama de actividades de este proceso, y más adelante se describirá en detalle la transacción generada por este proceso.

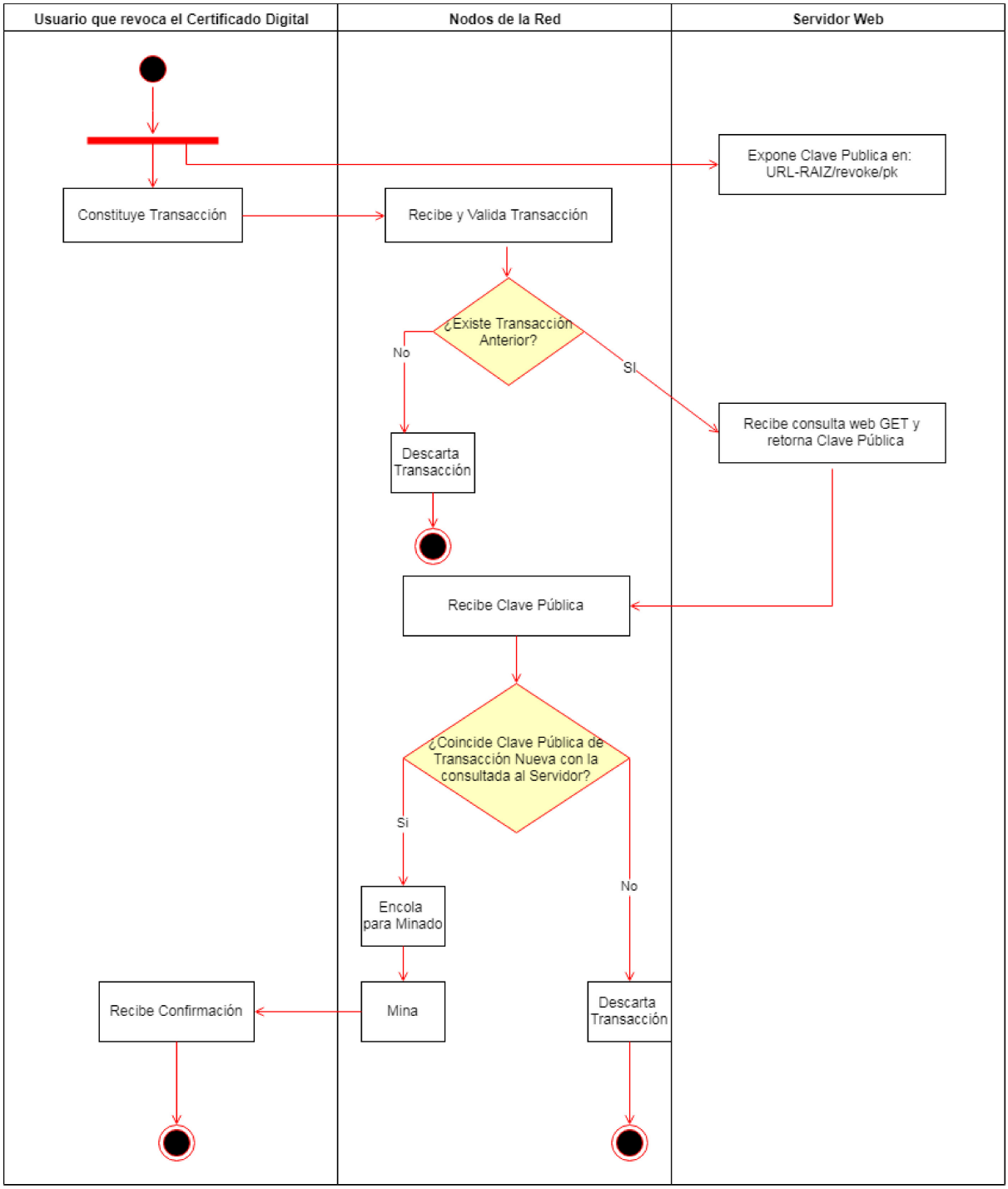

Ilustración 11:Diagrama de Actividades del Proceso de Revocación

En esta etapa usuario portador de la clave pública del servidor debe exponer la **nueva** clave pública en "URL-RAÍZ/renovate/pk" y genera una transacción con las siguientes características:

- Como **dirección:** el hash de la misma.
- Nueva **clave pública** del servidor.
- El **dominio** portador de la clave pública.
- La ruta donde buscarla (/renovate/pk).
- La **dirección** de la transacción de emisión o renovación (transacción anterior).

Todos los nodos que reciban dicha transacción, verificarán su integridad y realizarán una petición web GET hacia el recurso expuesto (por medio de la ruta proporcionada en la transacción de revocación o en la indicada en la transacción de emisión o renovación anterior). Ya recibida la respuesta de la petición, se compara el resultado con la clave pública del servidor contenida dentro de la **transacción** realizada recientemente y en caso afirmativo se retransmite a los demás nodos y se encola para realizar el proceso de minado, en su defecto la misma se descarta.
Una vez minada y confirmada la transacción (al igual que en Bitcoin, para la confirmación se aconsejan 10 bloques por delante del bloque en donde se encuentra la transacción), el creador de la transacción debe generar un certificado x.509. Este certificado al igual que en el proceso de emisión deberá rellenar en el campo opcional ***issuerUniqueID*** (Identificador único del emisor) con el identificador de la **transacción** confirmada. Ya a esta altura el usuario se encuentra en condiciones de agregarlo a la configuración HTTPS del servidor con su dominio en cuestión y/o firmar otros certificados digitales.
En la *Ilustración 12* se puede apreciar en detalle el diagrama de actividades de este proceso, y más adelante se describe en detalle la transacción generada por este proceso.

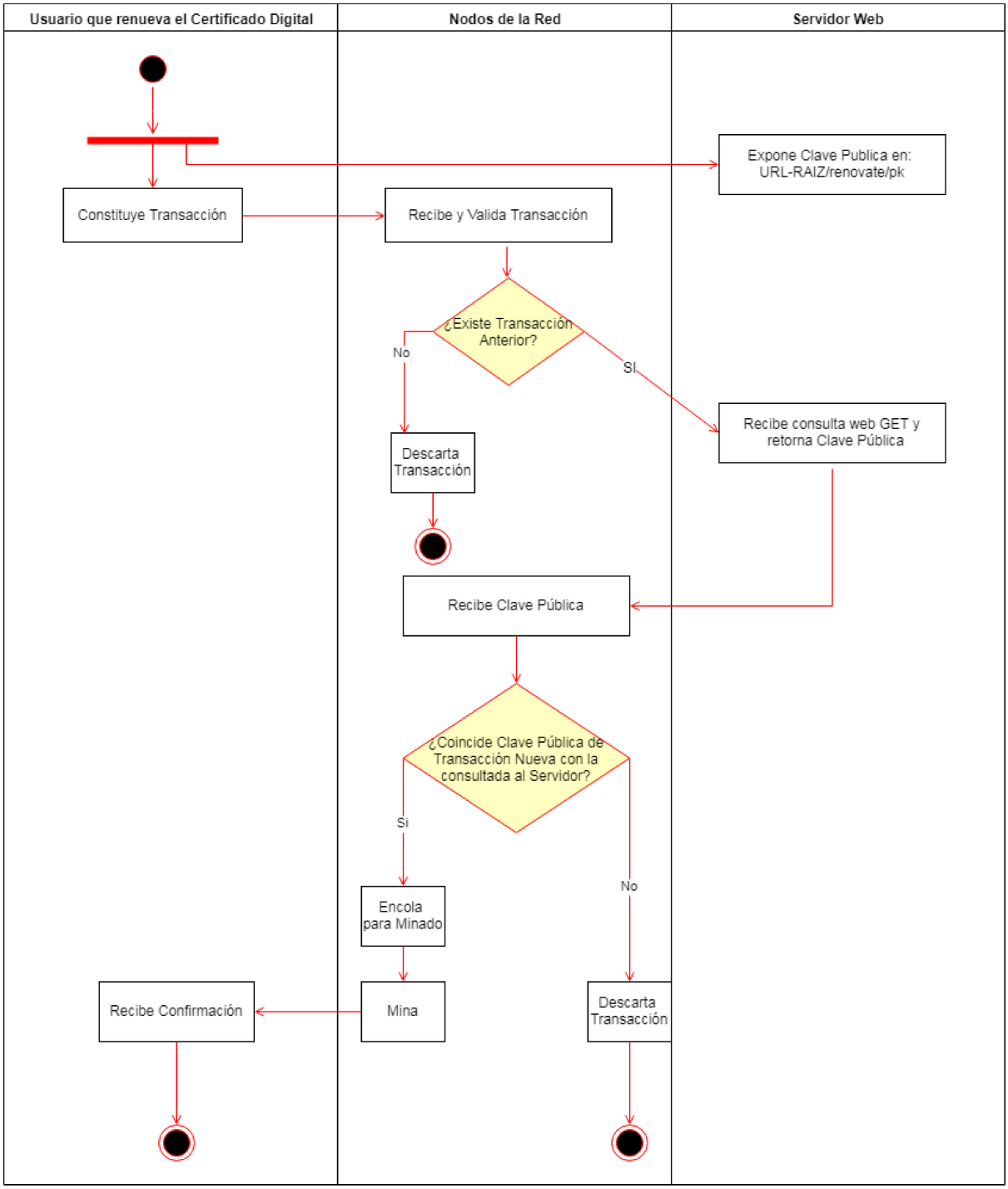

Ilustración 12:Diagrama de Actividades del Proceso de Renovación

Transacciones de la Arquitectura Propuesta

Las transacciones al igual que en Bitcoin son estructuras de datos firmadas digitalmente, pero que en vez de cambiar el propietario de un bitcoin, realizan una acción de emisión, revocación o renovación de certificados digitales.
La estructura de datos es la siguiente (*Ilustración 13, Ilustración 14 e Ilustración 15*):

- **Transacción anterior**: registro que referencia a una transacción previa. Para transacciones de **emisión** este campo se lo fija en 0, ya que no posee transacciones anteriores, y para transacciones de **renovación** o **revocación** se referencia a la transacción anterior.
- **Dominio del servidor:** portador del certificado a emitir, renovar o revocar.
- **Clave pública del servidor:** que se quiere dar de alta, relacionada a ese dominio.
- **Dirección (path/url):** en donde encontrar la **clave pública del servidor** para cotejar con la contenida en la transacción. Para transacciones de emisión se completa el campo "emit", para renovación el "renovate" y para revocación el"revoke".
- **Hash de transacción (identificador)**: resumen de toda la estructura de datos.
- **Firma digital del emisor**: encripción del **hash de la transacción** con la clave privada del nodo emisor.

Por ejemplo, si **A** desea **emitir** un certificado digital para el dominio "www.prueba-pkchain.com", entonces se genera el par de claves para su servidor, expone la clave pública en "www.prueba-pkchain.com/emit/pk", y genera la transacción con los siguientes campos:

- **server domain:** "www.prueba-pkchain.com"
- **emit:** "/emit/pk"
- **revoke:** "" (Vacío, ya que es una transacción de emisión)
- **renovate:** "" (Vacío, ya que es una transacción de emisión)
- **server public key**: "clave pública perteneciente al par de clave generado"
- **previous transaction:** "" (Vacío, ya una transacción de emisión no posee transacción anterior).

Si **A** desea **renovar** esta emisión, deberá generar el nuevo par de claves, publicar la nueva clave en la dirección de "renovate" declarada en esta nueva transacción. La transacción contendrá los siguientes campos:

- **server domain:** "www.prueba-pkchain.com"
- **emit:** "" (Vacío, ya que es una transacción de renovación)
- **revoke:** ""(Vacío, ya que es una transacción de renovación)
- **renovate:** "/renovate/pk"
- **server public key:** "clave pública perteneciente al nuevo par de clave generado".
- **previous transaction:** Identificador de transacción anterior.

```
{
"id": "f1b7fed773accafad5088eb4c3957cb050912535a4ce2ed81b0781ea69ba4669",
"libCert": {
        "serverDomain": "http://www.prueba-pkchain.com",
        "emit": "/emit/pk",
        "revoke": "",
        "renovate": "",
        "serverPublicKey": "04d50ce5ff067e2dbb28f197ff05ab3a4d363227f169da5dd821d821...",
        "previousTx": ""
      }
}
```


Ilustración 13: Transacción de Emisión en formato JSON


```
{
"id": "abc1345fdf2e75064b1559e5520a51f896c7f28c8576d68205e326a8d3dee873",
"libCert": {
        "serverDomain": "http://www.prueba-pkchain.com",
        "emit": "",
        "revoke": "",
        "renovate": "/renovate/pk",
        "serverPublicKey": "04fe71e62fc22b20e3473a90b3d1005d49f9d0c0aab69cfb7dd99122...",
        "previousTx": "f1b7fed773accafad5088eb4c3957cb050912535a4ce2ed81b0781ea69ba4669"
      }
}
```


Ilustración 14: Transacción de Renovación en formato JSON, que renueva la clave emitida en la ilustración 13

Por último, si **A** desea **revocar** un certificado, deberá publicar la clave a revocar en la dirección de "revoke" declarada en la transacción nueva. La estructura de datos contendrá los siguientes campos:

- **server domain:** "www.prueba-pkchain.com"
- **emit:** "" (Vacío, ya que es una transacción de revocación)
- **revoke:** "/revoke/pk"
- **renovate:** "" (Vacío, ya que es una transacción de revocación)
- **server public key:** clave pública que se quiere revocar.
- **previous transaction:** Identificador de transacción anterior.

Cada transacción excepto las de "emisión", poseen referencias o "punteros" a anteriores que se encuentran reconocidas por toda la red y almacenada en la base de datos distribuida, no obstante, se encuentran disponibles para que todos los nodos puedan consultarlas.


```
{
"id": "zf11345fdf2e75064b1559e5520a51f896c7f28c8576d68205e326a8d3deeds3",
"libCert": {
        "serverDomain": "http://www.prueba-pkchain.com",
        "emit": "",
        "revoke": "/revoke/pk",
        "renovate": "",
        "serverPublicKey": "04fe71e62fc22b20e3473a90b3d1005d49f9d0c0aab69cfb7dd99122...",
        "previousTx": "abc1345fdf2e75064b1559e5520a51f896c7f28c8576d68205e326a8d3dee873"
      }
}
```

Ilustración 15: Transacción de Revocación en formato JSON, que revoca la clave renovada en la ilustración 14 y por lo tanto la emitida en la ilustración 13

Construcción del Camino de la Certificación

Si un **cliente pkchain** se conecta a un servidor web, este último retorna el certificado digital. El cliente, accede al campo "***issuerUniqueID*** " y con el identificador de transacción contenido dentro busca en toda la blockchain hacia atrás. Si este no se encuentra como entrada de una transacción de revocación, la transacción existe contenida dentro de un bloque de la *blockchain*, y la información contenida dentro de la transacción coincide con la del certificado digital enviado por el servidor (dominio y clave pública), entonces se da luz verde y el mismo se acepta, de lo contrario se lo rechaza.

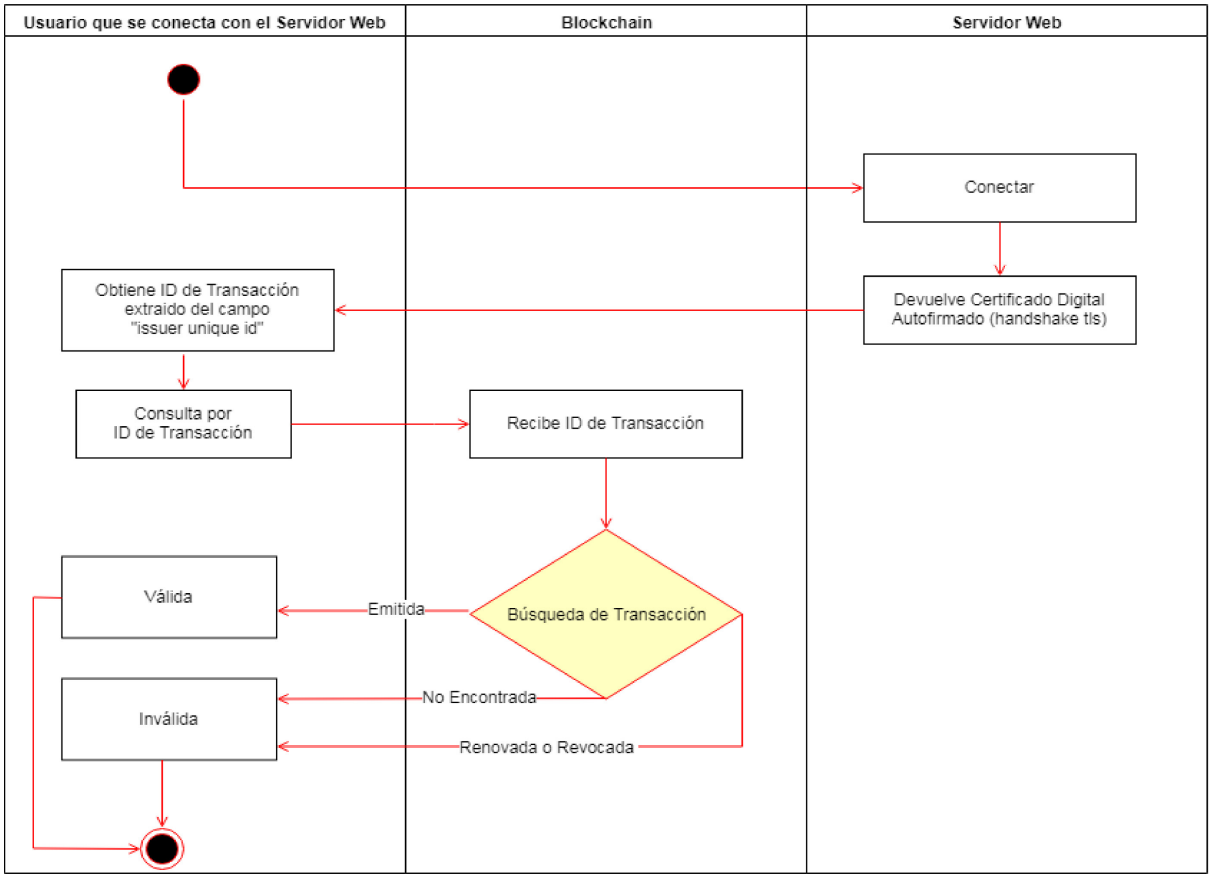

Ilustración 16: Proceso de Verificación del Certificado Digital

Consideraciones Generales de la Arquitectura Propuesta

Desde el punto de vista de la convergencia de la red a nivel blockchain, se intentó emular el proceso idéntico al implementado en Bitcoin, es decir, en cuestiones de minería, pruebas de trabajo, estructura de datos de bloques, tiempos entre bloques, criptografía utilizada y ajustes de dificultad se realizó una copia de la lógica vigente de la *blockchain* de Bitcoin. Estos detalles se encuentran descriptos en el **Anexo 1**.

Capítulo 7 - Proyecciones, Trabajos a Futuro y Conclusiones

Blockchain pone sobre la mesa la posibilidad de realizar transacciones entre entidades (personas u organizaciones) de manera distribuida, segura y sin necesidad de intermediarios, donde las reglas del juego están definidas por medio de algoritmos computacionales.
Llegar a tener una sociedad más distribuida y autónoma en cuanto a su estructura, sus redes y transacciones es algo que podría venir de la mano de esta tecnología y que podría cambiar la forma en que muchas cosas están pensadas hoy.
De la misma manera que **internet** cambió para siempre los modelos de negocio de miles de industrias y empresas, la ***blockchain*** o **cadena de bloques** está dando lugar a una nueva revolución, proponiendo nuevas formas de optimizar las relaciones entre usuarios, ahorrar

costes administrativos, favorecer cooperaciones y comprender todas las posibilidades imaginables que ofrecía el internet de la información en una segunda ola tecnológica de cambio.
Blockchain es la verdadera innovación que hizo posible la existencia de **Bitcoin** (su primer uso de esa tecnología) y eso le permitió ser, hasta el día de hoy, la más popular de todas, al punto de ser más popular en el consciente colectivo que ***Blockchain*** en sí, no obstante, lo más importante que se extrae de esta gran solución es la posibilidad de utilizar la **cadena de bloques** en otros ámbitos, como por ejemplo, su incorporación en la propuesta abordada en este trabajo de tesis.

Proyecciones y Trabajos a Futuro

En este trabajo de tesis se abordó una propuesta de reingeniería y migración de la arquitectura de certificados digitales vigente a una montada sobre ***blockchain***, sería interesante analizar una propuesta de implementación que se monte sobre su alternativa: ***hashgraph***.
En esta propuesta se ha desarrollado un prototipo a modo de validar el modelo, no obstante, la implementación real junto con la apertura de un proyecto ***open source*** para que la comunidad pueda realizar sus aportes invocando a la inteligencia colectiva, es el principal trabajo que queda pendiente por realizar.
Esta arquitectura de certificados digitales distribuida y descentralizada tiene un enfoque orientado a arquitecturas web, por lo tanto, realizar un análisis de viabilidad de implementación en diferentes ámbitos como ser firma digital de expedientes o en voto electrónico sería de gran interés.
Sería muy interesante poder también lograr la interoperabilidad entre la propuesta de esta tesis y la lógica de negocios de **Bitcoin**, de modo de integrar las dos soluciones para que puedan convivir al mismo tiempo en la misma **cadena de bloques**. Para lograr esto, en vez de quitar la lógica de negocios de **Bitcoin** (como se planteó en esta tesis), se deberá realizar una modificación en la misma para agregar la lógica de negocios de esta propuesta.
Por último, visto y considerando que la solución planteada en esta tesis descansa sobre ***blockchain***, también acarrea sus mismas debilidades, no obstante, es motivo de trabajo a futuro contemplar mejoras en este sentido.

Conclusiones

Dada la gran frecuencia en la que surgen proyectos montados sobre ***blockchain***, y no solo eso, sino la cantidad de criptomonedas vigentes y por aparecer que están revolucionando la economía mundial, es necesario conocer los fundamentos que se hacen del núcleo de este área de conocimiento de forma tal de dominar la tecnología, conocer sus posibles vulnerabilidades, y mejoras.
Ha habido gran cantidad propuestas para lograr una formalización de la ingeniería en seguridad de ***Blockchain***, pero la mayoría tienen un alcance limitado. Resulta totalmente

relevante incursionar en el tema ya que hasta el momento no se ha podido probar de manera exhaustiva si la arquitectura de ***Blockchain*** sea totalmente segura o no, o más aún, si es tan útil como se dice o no [81][82][83][84][85][86].

Dentro de la filosofía ***Blockchain*** existen muy pocas obras que realicen una utilización en serio de sus funcionalidades, y de las pocas que existen, la mayor parte la aborda con un enfoque indiscreto e irreal sobre cómo descentralizar en dos pasos al mundo.

Cabe destacar que en la actualidad casi no existe bibliografía seria y en español que aborde y exponga una detallada descripción de los procesos internos, arquitecturas y operaciones tanto de **Bitcoin** como de ***Blockchain***. La escasa bibliografía existente brinda diferentes análisis y proyecciones genéricas más inclinadas a puntos de vistas de las ciencias económicas que técnicas de la ingeniería.

En esta tesis se puede apreciar una detallada descripción en español de todos los conocimientos que se encuentran detrás de este telón, comenzando por los fundamentos básicos, hasta entrando en las entrañas matemáticas que dirigen la orquesta. No obstante, se hace mención a una necesidad contemporánea muy importante, básica y colectiva, que obliga a enfocarse hacia un cambio de paradigma en la arquitectura de certificados digitales vigente. En consiguiente, se presenta una propuesta que integra conocimientos y funcionalidades provenientes de tecnologías de punta como es la implementación de diferentes sistemas sobre ***Blockchain***.

Finalmente, como conclusión no solo personal, sino de la comunidad informática en general, es que el aporte más grande que ha dado Satoshi Nakamoto con su publicación en 2009 no fue **Bitcoin**, sino su columna vertebral, es decir, ***Blockchain***. Una tecnología que tiene características ilimitadas tanto desde el punto de vista técnico como económico, que permite abordar investigaciones, realizar propuestas innovadoras y simples implementaciones como la presentada en esta tesis.

Tabla de Definiciones

Bitcoin	Proyecto open source de moneda criptográfica, descentralizada, virtual e inteligible.
Ethereum	Alternativa de Bitcoin y plataforma open source, descentralizada que permite la creación de acuerdos de contratos inteligentes entre pares, basada en el modelo blockchain.
Litecoin	Criptomoneda alternativa de Bitcoin, sustentada por la red P2P, y un proyecto de software de código abierto publicado bajo la licencia MIT.
Arbol de Merkle	Estructura de datos en árbol, binario o no, en el que cada nodo que no es una hoja está etiquetado con el hash de las de la concatenación de las etiquetas o valores (para nodos hoja) de sus nodos hijo. Son una generalización de las listas hash y las cadenas hash.
Blockchain	Es una base de datos distribuida, descentralizada, sincronizada, formada por cadenas de bloques diseñadas para evitar su modificación una vez que un dato ha sido publicado usando un sellado de tiempo confiable y enlazando a un bloque anterior.
Hashgraph	Sistema de tecnología avanzada de contabilidad distribuida que elimina la necesidad de cálculos masivos y consumo de energía insostenible como los de la Blockchain de Bitcoin y Ethereum. Cada nodo en Hashgraph puede difundir información sellada (llamados eventos) sobre transacciones creadas y transacciones recibidas de otros, a otros nodos elegidos al azar. Estos nodos agregarán los eventos recibidos con la información recibida de otros nodos en un nuevo evento, y luego lo enviarán a otros nodos elegidos al azar. Este proceso continúa hasta que todos los nodos conocen la información creada o recibida al principio. La nueva información puede llegar a cada nodo de la red de una manera muy rápida.

Transacción	Conjunto de datos incluye una referencia a una transacción anterior e indica una cantidad de bitcoins que pasan a estar disponibles para una dirección Bitcoin de destino.
Transacción de Coinbase	Primer transacción de un bloque. Aquí se encuentran las recompensas del nodo minero.
Bloque	Estructura de datos que se encadena en la Blockchain y contiene una o más transacciones almacenadas.

Tabla de Ilustraciones

Anexo I

Introducción

Este anexo posee (a modo de documentación) una descripción completa de la implementación de esta solución propuesta.
El objetivo principal de este apartado consiste en explicar de forma simple la implementación del prototipo desarrollado para lograr su completo entendimiento. Esta solución experimental fue desarrollada con el fin de ejemplificar y demostrar los principios básicos de su funcionamiento y aplicabilidad.
El prototipo implementado de Arquitectura de Certificados Digitales Distribuido y Descentralizado está constituido por varios componentes de software, donde:

- El core de la gestión de certificados digitales fue construido bajo el framework "Node JS" [74] y librerías codificadas en "Javascript"[75] para dar soporte en de todo lo que tenga que ver con aspectos criptográficos, de comunicaciones P2P y HTTP. El core incluye los módulos "PkChain", "LibCert" y "Gestor de Servicios" (ver Ilustración 9).
- Los certificados digitales emitidos serán expuestos en un servidor HTTP simple construido en "Spring Framework"[76]. Aquí también se expondrán las claves públicas al momento de realizar las transacciones de emisión/revocación/renovación, de forma tal de que los nodos que validen dichas transacciones puedan cotejar y comparar contra éste. Esta implementación materializa al módulo "Servidor Web" (ver Ilustración 9).
- El "cliente HTTP" (ver Ilustración 9) que se conecta al servidor HTTP anterior y obtiene el certificado digital, que posteriormente será validado dentro de la PkChain, fué desarrollado en lenguaje de programación Java[79].
- El "cliente PkChain" (ver Ilustración 9) que controla el nodo PkChain toma dos formas. La primera interactúa dentro del módulo "cliente HTTP" anterior y está programada en lenguaje de Java[79], la segunda es independiente y utiliza el cliente REST Insomnia[80].

- Para emitir los certificados digitales y dentro de éstos colocar el ID de la Transacción pertinente (de emisión o renovación) se utilizará el “OpenSSL”[77].

La Blockchain

Con el fin de implementar las funcionalidades básicas de la cadena de bloques, se dispuso crear un componente llamado “PkChain” *(ver Ilustración 9)* que emule su funcionamiento, dentro del cual se incluye:

- Definición de la estructura de bloque y blockchain.
- Funciones para agregar nuevos bloques a la cadena de bloques con datos una o más transacciones de emisión, renovación o revocación.
- Nodos blockchain que se comunican y sincronizan la cadena de bloques entre sí.
- Una API HTTP para controlar el nodo blockchain e invocar todas las funciones.

Estructura del bloque

Campos contenidos dentro de la misma:

- **índice:** la altura del bloque en la cadena de bloques.
- **transacciones:** conjunto de transacciones.
- **marca de tiempo:** timestamp.
- **resumen del bloque:** hash.
- **resumen del bloque anterior:** hash del bloque anterior (lógica de Merkle).
- **dificultad:** dificultad en la que la red converge.
- **nonce:** campo que es variado por un nodo minero para lograr el hash del bloque en función a la dificultad convergente de la red.

Estructura del bloque en código fuente:

```
class Bloque{

   public indice: number;
   public hash: string;
   public hashPrevio: string;
   public timestamp: number;
   public transacciones: Transaccion[];
   public dificultad: number;
   public nonce: number;

   constructor(indice: number, hash: string, hashPrevio: string, timestamp:
number, transacciones: Transaccion[], dificultad: number, nonce: number) {
```

```
        this.indice = indice;
        this.hashPrevio = hashPrevio;
        this.timestamp = timestamp;
        this.transacciones = transacciones;
        this.hash = hash;
        this.dificultad = dificultad;
        this.nonce = nonce;
    }
}
```

El hash de bloque es una de sus más importantes propiedades. El hash se calcula sobre todos los datos del bloque. Esto significa que si algo en el bloque cambia, el hash original ya no es válido. Este hash también se puede considerar como el identificador único del mismo. Por ejemplo, pueden aparecer bloques con el mismo índice, pero todos tienen hashes diferentes.

Cálculo de hash del bloque en código fuente:

```
CryptoJS.SHA256(indice+hashPrevio+timestamp+transacciones+dificultad+nonce);
```

Generación de Bloques

Bloque Génesis

El primer bloque de la blockchain es el único que no posee hash de bloque anterior ni transacciones.
Creación del bloque génesis en código fuente:

```
const bloqueGenesis: Bloque = new Bloque(0,
'fb7dd991229153b9f732ba5334aafcd8e7266e47076996b55a14bf9913ee3145', '',
1579154514, [], 0, 0);
```

Bloques

Para generar un bloque, debemos conocer el hash del bloque anterior y crear el resto del contenido requerido (índice, hash, transacciones y timestamp). Los datos de las transacciones son proporcionados por el usuario final, pero el resto de los parámetros se generarán con el siguiente código:

```
const generarProximoBloque = (transacciones: Transaccion[]) => {
    const bloquePrevio: Bloque = getUltimoBloque();
    const proximoIndice: number = bloquePrevio.indice + 1;
    const proximoTimestamp: number = new Date().getTime() / 1000;
  const proximoHash: string = CryptoJS.SHA256(proximoIndice,bloquePrevio.hash,
proximoTimestamp,                                          transacciones);
```

```
 const nuevoBloque: Bloque = realizarPruebaDeTranajo(proximoIndice,
bloquePrevio.hash, proximoTimestamp,transacciones, getDificultad(getBlockchain())

);
    return                                                          nuevoBloque;
};
```

Y en este código está materializado el algoritmo de la prueba de trabajo a realizar por el nodo minero que creará el bloque. La prueba de trabajo, como antes vista, es el cálculo del hash del bloque, junto a la variación del campo "nonce", hasta llegar a un hash con la cantidad de "ceros" adelante igual a la dificultad obtenida de la red.

```
const realizarPruebaDeTranajo= (indice: number, hashPrevio: string, timestamp:
number, transacciones: Transaccion[], dificultad: number): Bloque => {
    let                    nonce                    =                    0;
    while                              (true)                              {
        const hash: string = CryptoJS.SHA256(indice, hashPrevio, timestamp, data,
dificultad,                                                    nonce).toString();
        if            (hashCorrecto(hash,            dificultad))            {
            return new Bloque(indice, hash, hashPrevio, timestamp, transacciones,
transacciones,                                                          nonce);
        }
        nonce++;
    }
};
```

La definición de la "dificultad" se lleva a cabo en base a dos parámetros:

- La frecuencia de creación de bloques.
- La frecuencia con la que se debe ajustar la dificultad (aumento o disminución de dificultad).

Por lo tanto, en el código fuente esto se materializa en dos atributos.

```
// en segundos
const INTERVALO_GENERACION_BLOQUE: number = 1000;

// en bloques
const INTERVALO_AJUSTE_DIFICULTAD: number = 15;


const getDificultad = (blockchain: Bloque[]): number => {
    const ultimoBloque: Bloque= blockchain[blockchain.length - 1];
    if (ultimoBloque.indice % INTERVALO_AJUSTE_DIFICULTAD === 0 &&
ultimoBloque.indice !== 0) {
        return getAjusteDificultad (ultimoBloque, blockchain);
    } else {
        return ultimoBloque.dificultad;
```

```
    }
};

const getAjusteDificultad = (ultimoBloque: Bloque, blockchain: Bloque[]) => {
    const bloqueAjustePrevio: Bloque= blockchain[getBlockchain().length -
INTERVALO_AJUSTE_DIFICULTAD];
    const tiempoEsperado: number = INTERVALO_GENERACION_BLOQUE*
INTERVALO_AJUSTE_DIFICULTAD;
    const tiempoTomado: number = ultimoBloque.timestamp -
bloqueAjustePrevio.timestamp;
    if (tiempoTomado < tiempoEsperado/ 2) {
        return bloqueAjustePrevio.dificultad + 1;
    } else if (tiempoTomado > tiempoEsperado * 2) {
        return bloqueAjustePrevio.dificultad - 1;
    } else {
        return bloqueAjustePrevio.dificultad;
    }
};
```

Almacenamiento de Bloques

Aquí entra en juego la cadena de bloques o blockchain, que por el momento se materializa en un Array de Javascript en memoria.
Definición de la blockchain en código fuente:

```
const blockchain: Bloque[] = [bloqueGenesis];

const getBlockchain = (): Bloque[] => blockchain;
```

Administración de la Blockchain

Validación Integridad

Para que un nodo sea íntegramente válido, debe:

- Contener el número del campo “Índice” mayor que el índice del bloque anterior.
- Campo “Hash de bloque previo” coincidir con el hash del bloque anterior.
- El campo “Hash” del bloque debe ser válido.

Validando en código fuente:

```
const nuevoBloqueValido= (nuevoBloque: Bloque, bloquePrevio: Bloque) => {
    if (bloquePrevio.indice + 1 !== nuevoBLoque.indice) {
        console.log('Índice inválido');
```

```
        return false;
    } else if (bloquePrevio.hash !== nuevoBloque.hashPrevio) {
        console.log('Hash previo inválido');
        return false;
    } else if (calcularHash(nuevoBloque) !== nuevoBloque.hash) {
        console.log(typeof (nuevoBloque.hash) + ' ' + typeof
calcularHash(nuevoBloque));
        console.log('Hash inválido: ' + calcularHash(nuevoBloque) + ' ' +
nuevoBloque.hash);
        return false;
    }
    return true;
};
```

Validación Timestamp

Para mitigar el ataque donde se introduce una marca de tiempo falsa para manipular la dificultad, se introducen las siguientes reglas:

- Un bloque es válido, si la marca de tiempo es como máximo 1 minuto en el futuro desde el momento en que se generó.
- Un bloque en la cadena es válido, si la marca de tiempo es como máximo 1 minuto en el pasado del bloque anterior.

```
const timestampValido= (nuevoBloque: Bloque, bloquePrevio: Bloque): boolean => {
    return ( bloquePrevio.timestamp - 60 < nuevoBloque.timestamp )
        && nuevoBloque.timestamp - 60 < getTimestampActual();
};
```

Carrera por la cadena más larga

Este mecanismo se implementa con el fin de contar siempre con una sola cadena de bloques, es decir, si existen más de una rama en la blockchain siempre se optará por una, y este criterio de elección consiste en la que posea mayor dificultad acumulativa, en otras palabras, la cadena que requirió la mayor cantidad de recursos computacionales para generar los bloques. Para obtener la dificultad acumulativa de una cadena se debe calcular 2^(dificultad) para cada bloque y tomar una suma de todos esos números. Se deberá usar la 2^(dificultad) opción cómo se elige la dificultad para representar el número de ceros que debe prefijar el hash en formato binario. Por ejemplo, si se comparan las dificultades de 4 y 10, se requieren 2 ^ (10-4) = 2 ^ 6 veces más trabajo para encontrar un bloque con esta última dificultad.

Transacciones

Las transacciones forman parte de la lógica de negocios propia de esta propuesta y materializan el módulo de software llamado "LibCert" *(ver Ilustración 9)* que es consumido como una librería por el módulo "PkChain" *(ver Ilustración 9)*.
La estructura de datos de las transacciones en código fuente es la siguiente

```
class Transaccion {

   public id: string;
   public libCert: LibCert;

}
```

Donde en "libCert" se encuentra la estructura de datos donde funciona la lógica del prototipo implementado:

```
class LibCert {
   public dominioServidor: string;
   public emision: string;
   public revocacion: string;
   public renovacion: string;
   public txPrevia: string;
   public clavePublicaServidor: string;
   public firma: string;

   constructor(dominioServidor: string, emision: string, revocacion: string,
renovacion: string, clavePublicaServidor: string, txPrevia: string) {
       this.dominioServidor = dominioServidor;
       this.emision = emision;
       this.revocacion = revocacion;
       this.renovacion = renovacion;
       this.clavePublicaServidor = clavePublicaServidor;
       this.txPrevia = txPrevia;
   }
}
```

La identificación de la transacción se calcula tomando un hash del contenido de la transacción.

```
const getIdDeTransaccion = (transaccion: Transaccion): string => {
return CryptoJS.SHA256(transaccion.libCert.dominioServidor+
transaccion.libCert.emision+transaccion.libCert.revocacion +
transaccion.libCert.renovacion+transaccion.libCert.txPrevia +
transaccion.libCert.clavePublicaServidor).toString();
};
```

Es importante que el contenido de la transacción no pueda ser alterado, después de que haya sido firmado. Como las transacciones son públicas, cualquiera puede acceder a las transacciones, incluso antes de que estén incluidas en la cadena de bloques. Solo se firmará el Id de la transacción. Si se modifica alguno de los contenidos en las transacciones, el Id de transacción cambiará, haciendo que la transacción y la firma no sean válidas.

```
const firmarTransacción= (transaccion: Transaccion, privateKey: string): string =>
{
    const          llave=          ec.keyFromPrivate(privateKey,          'hex');
    const   firma:   string   =   toHexString(llave.sign(transaccion.id).toDER());
    return                                                                  firma;
};
```

Se utiliza una biblioteca llamada “elíptica”[78] para la criptografía de clave pública, esta librería utiliza una implementación de curvas elípticas.

Actualización de lista de certificados digitales revocados y renovados

Cada vez que se agrega un nuevo bloque a la cadena, se debe actualizar la lista de certificados revocados y renovados, es decir, una lista en donde se encuentran las claves públicas no válidas por motivos de revocación o renovación. Esto se debe a que las nuevas transacciones deberán cotejar con la clave pública que persiste en estas listas.
La porción de código fuente que se encarga de administrar la actualización de estas listas es la siguiente:

```
for (let i = 0; i < nuevoBLoque.transacciones.length; i++) {

   const transaccionActual: Transaccion = nuevoBLoque.data[i];

       if(transaccionActual.libCert !== null){

           const revocacion: string = transaccionActual.libCert.revocacion;

           const renovacion: string = transaccionActual.libCert.renovacion;

           if(revocacion !== null){


agregarClavesRevocadas(transaccionActual.libCert.clavePublicaServidor);

           }else if(renovacion !== null){

              const tx = _(getBlockchain())

                    .map((blocks) => bloques.transacciones)
```

```
                    .flatten()
                    .find({'id': transaccionActual.libCert.txPrevia});
                agregarClavesRenovadas(tx.libCert.clavePublicaServidor);
            }
        }
}
```

Validación de transacciones

Al momento de crear una transacción, esta deberá ser validada bajo las siguientes restricciones:

- La transacción debe realizar sólo una acción, es decir, o emitir, o renovar o revocar, la clave pública no debe estar revocada, y si la transacción hace referencia a una transacción anterior, esta última debe estar minada:

```
let count = 0;

if(transaccion.libCert.emision !== ""){

   count ++;

}

if(transaccion.libCert.revocacion !== ""){

   count ++;

}

if(transaccion.libCert.renovacion !== ""){

   count ++;

}
```

```
if (count > 1) {

   throw Error('transacción inválida');

}

for (let i = 0; i < getPoolDeTransaccionesPendientes().length; i++) {

   if(transaccion.libCert.clavePublicaServidor                                    ===
getPoolDeTransaccionesPendientes()[i].libCert.clavePublicaServidor) {

       throw Error('la clave pública está siendo procesada en una transacción');

   }

}

for (let i = 0; i < getClavesRevocadas().length; i++) {

   if(transaccion.libCert.clavePublicaServidor === getClavesRevocadas()[i]){

       throw Error('la clave pública fué revocada');

   }

}
```

- Si la transacción es de emisión debe validarse lo siguiente:

```
   if(transaccion.libCert.txPrevia !== ""){

       throw Error('transacción inválida');

   }

   for (let i = 0; i < getClavesRenovadas().length; i++) {

       if(transaccion.libCert.clavePublicaServidor === getClavesRenovadas()[i]){
```

```
            throw Error('la clave publica fue revocada');
        }
    }
    for (let i = 0; i < getClavesEmitidas().length; i++) {
        if(transaccion.libCert.clavePublicaServidor === getClavesEmitidas()[i]){
            throw Error('la clave pública fue emitida');
        }
    }
```

- Si la transacción es de renovación debe validarse lo siguiente:

```
if(transaccion.libCert.txPrevia === ""){
    throw Error('transacción inválida');
}
const tx = _(getBlockchain())
    .map((bloques) => bloques.transacciones)
    .flatten()
    .find({'id': txPrevia});
if(tx === null || tx.libCert.revocacion !== ""){
    throw Error('transacción previa inválida');
}
if(tx.libCert.clavePublicaServidor === transaccion.libCert.clavePublicaServidor){
```

```
   throw Error('la clave a renovar es la misma que la anterior');

}
```

- Si la transacción es de revocación debe validarse lo siguiente:

```
if(transaccion.libCert.txPrevia === ""){

   throw Error('transacción inválida');

}

const tx = _(getBlockchain())

   .map((bloques) => bloques.transaccion)

   .flatten()

   .find({'id': transaccion.libCert.txPrevia});

if(tx === null  || tx.libCert.revocacion !== ""){

   throw Error('transaccion previa inválida');

}
```

- El nodo PkChain coteja la clave pública contra el dominio y clave pública expuesta en la transacción:

```
try {
   let res = request('GET', transaccion.libCert.dominioServidor + path);
   if (res.statusCode === 200 && res.getBody().toString() !==
transaccion.libCert.clavePublicaServidor) {
        throw Error('comparación inválida');
   }
}catch(Exception){
   throw Error('dominio inválido');
}
```

Controlador del Nodo PkChain

Cada instancia del módulo de software “PkChain” corresponde a un nodo diferente. Este nodo posee, al igual que el nodo de **Bitcoin**, una copia de toda la blockchain, una copia de las transacciones pendientes a minar, tiene la propiedad de minar bloques, de conectarse con otros nodos y de crear, validar y diseminar transacciones y bloques a todos sus nodos vecinos, para que estos mismos a su vez lo hagan con sus vecinos, y así termine convergiendo la red.
Cada nodo posee una interfaz abstracta (API HTTP) que será utilizada por el usuario final. Este último interactuará con esta interfaz por medio de un cliente http que representa al módulo de software “Cliente PkChain” *(ver Ilustración 9)*.
Un “Cliente PkChain” podrá indicarle al Nodo PkChain que realice:

- Creación de nuevas transacciones:

```
curl -H "Content-type: application/json" --data ' {
"dominioServidor" : "http://prueba.com",
"emision" : "/emit/pk",
"revocacion" : "",
"renovacion" : "",
"txPrevia" : "",
"clavePublicaServidor":
"304502201d0f05d87e589ef222019c2480cb73ca0e7d69311a6c4e43b7cd1c106dac4335022100a07
139bb92d76c2ec30a17fe71e62fc22b20e3473a90b3d1005d49f9d0c0aab6"
 }' http://miInterfazNodoPkChain:3001/enviarTransaccion
```

- Minar nuevos bloques:

```
curl -X POST http://miInterfazNodoPkChain:3001/minarBloque
```

- Mostrar cadena de bloques.

```
curl http://miInterfazNodoPkChain:3001/blockchain
```

- Mostrar transacciones.

```
curl http://miInterfazNodoPkChain:3001/transaccionesPendientes
curl http://miInterfazNodoPkChain:3001/transacciones
```

- Conectarse con nodos.

```
curl -H "Content-type:application/json" --data '{"peer" :
"ws://miNodoPkChain:6001"}' http://miInterfazNodoPkChain:3001/agregarNodo
```

- Mostrar Nodos Conectados.

```
curl http://miInterfazNodoPkChain:3001/nodos
```

- Consultar si una clave pública está emitida, revocada y/o renovada.

```
curl http://miInterfazNodoPkChain:3001/transaccion/emitida?id={idTransaccion}.
curl http://miInterfazNodoPkChain:3001/transaccion/revocada?id={idTransaccion}
curl http://miInterfazNodoPkChain:3001/transaccion/renovada?id={idTransaccion}
curl http://miInterfazNodoPkChain:3001/transaccion/valida?id={idTransaccion}
```

Comunicación entre Nodos PkChain

Un papel esencial de un nodo es compartir y sincronizar la blockchain con otros nodos. Esta actividad es llevada a cabo por el módulo "Gestor de Servicios" (ver Ilustración 9), el cual es consumido por la lógica del componente "PkChain".
Las siguientes reglas se utilizan para mantener la red sincronizada.

- Cuando un nodo genera un nuevo bloque, lo transmite a la red.
- Cuando un nodo se conecta a otro nodo consulta por el último bloque.
- Cuando un nodo encuentra un bloque que tiene un índice más grande que el bloque conocido actual, agrega dicho bloque a su cadena actual o consulta la cadena de bloques completa.

Se utilizará websockets para la comunicación entre los nodos. Los enlaces activos para cada nodo se almacenan en la variable:

```
const sockets: WebSocket[].
```

Para restringir el alcance y controlar el ambiente de pruebas no se utilizará descubrimiento de nodos automático. Las conexiones entre los mismos se deben agregar manualmente mediante el "cliente PkChain".

El Papel del OpenSSL

Una vez generada la transacción de emisión o renovación se deberá insertar el ID de la Transacción en el certificado digital que se expondrá en el "Servidor WEB" (ver Ilustración 9).
Por medio de los siguientes comandos se realiza esta tarea:

```
openssl genrsa -out myCA.key 2048
```

```
openssl req -x509 -new -nodes -key myCA.key -sha256 -days 1825 -subj
"/x500UniqueIdentifier={ID de Transacción}" -out myCA.pem
```

```
openssl asn1parse -in myCA.pem -inform PEM

openssl genrsa -out myServer.key 2048

openssl req -new -key myServer.key -out myServer.csr

openssl x509 -req -in myServer.csr -CA myCA.pem -CAkey myCA.key -CAcreateserial -
out myServer.crt -days 1825 -sha256



openssl pkcs12 -export -in myServer.crt -inkey myServer.key -out myServer.p12 -
name tomcat -CAfile myCA.pem -caname root -chain
```

El Servidor Web

El módulo “Servidor Web” (ver Ilustración 9) será en donde se expondrán las claves públicas para que los nodos de la PkChain puedan validar las transacciones (“/emit/pk”, “/revoke/pk”, “/renovate/pk”). Este componente, también es en el que se configurará el entorno HTTPS una vez emitido el certificado digital para otorgar seguridad en las comunicaciones.
Configuración HTTPS en Spring Framework (archivo *application.properties*):

```
server.port: 8443
server.ssl.key-store: classpath:myServer.p12
server.ssl.key-store-password: password
server.ssl.keyStoreType: PKCS12
server.ssl.keyAlias: tomcat
server.ssl.protocol=TLS
security.require-ssl=false
```

El Cliente Web

El módulo “Cliente Web” (ver Ilustración 9) se encarga de conectarse a “Servidor WEB”, obtiene el certificado digital, extrae el “Issuer Unique ID” (donde se encuentra el ID de la Transacción), y coteja contra la Blockchain por medio del “Cliente PkChain”. Si la transacción existe y es una transacción de “emisión” el certificado digital es válido y se encuentra vigente, en su defecto, no es válido.

Código Fuente:

```
HttpResponseInterceptor certificateInterceptor = (httpResponse, context) -> {
```

```
   ManagedHttpClientConnection routedConnection =
(ManagedHttpClientConnection)context.getAttribute(HttpCoreContext.HTTP_CONNECTION)
;
   SSLSession sslSession = routedConnection.getSSLSession();
   if (sslSession != null) {

       Certificate[] certificates = sslSession.getPeerCertificates();

       context.setAttribute(PEER_CERTIFICATES, certificates);
   }
};

SSLContextBuilder builder = SSLContexts.custom();
builder.loadTrustMaterial(null, new TrustStrategy() {
   @Override
   public boolean isTrusted(X509Certificate[] chain, String authType)
           throws CertificateException {
       return true;
   }
});
SSLContext sslContext = builder.build();
SSLConnectionSocketFactory sslsf = new SSLConnectionSocketFactory(
       sslContext, new X509HostnameVerifier() {
   @Override
   public void verify(String host, SSLSocket ssl)
           throws IOException {
   }

   @Override
   public void verify(String host, X509Certificate cert)
           throws SSLException {
   }

   @Override
   public void verify(String host, String[] cns,
                      String[] subjectAlts) throws SSLException {
   }

   @Override
   public boolean verify(String s, SSLSession sslSession) {
       return true;
   }
});

Registry<ConnectionSocketFactory> socketFactoryRegistry = RegistryBuilder
       .<ConnectionSocketFactory> create().register("https", sslsf)
       .build();

PoolingHttpClientConnectionManager cm = new PoolingHttpClientConnectionManager(
       socketFactoryRegistry);
```

```
CloseableHttpClient httpClient = HttpClients
        .custom().setConnectionManager(cm)
        .addInterceptorLast(certificateInterceptor)
        .build();

try {

   HttpGet httpget = new HttpGet("https://IpServidor:PuertoServidor");

   HttpContext context = new BasicHttpContext();
   httpClient.execute(httpget, context);

   Certificate[]           peerCertificates           =           (Certificate[])
context.getAttribute(PEER_CERTIFICATES);

   X509Certificate real = (X509Certificate) peerCertificates[0];
   System.out.println("Issuer Unique ID: " + real.getIssuerX500Principal());
   String urlOverHttps
           =  "http://{IpNodoBlockchain:puertoNodoBlockchain}/validTransaction/"  +
real.getIssuerX500Principal().toString().split("=")[1];

   URL obj = new URL(urlOverHttps);
   HttpURLConnection con = (HttpURLConnection) obj.openConnection();

   con.setRequestMethod("GET");
   con.setRequestProperty("User-Agent", "Mozilla/5.0");

   int responseCode = con.getResponseCode();

   if (responseCode == 200) {
       System.out.println("La transacción dice que el certificado fue emitido y no
fue revocado ni renovado");
   } else {
       System.out.println("La transacción dice que el certificado fue revocado o
renovado");
   }

   System.out.println("Response Code : " + responseCode);

   BufferedReader in = new BufferedReader(
           new InputStreamReader(con.getInputStream()));
   String inputLine;
   StringBuffer response = new StringBuffer();

   while ((inputLine = in.readLine()) != null) {
       response.append(inputLine);
   }
   in.close();

   System.out.println();
```

```
   PublicKey publicKey = real.getPublicKey();

   System.out.println("La clave publica que viene en el certificado digital del
servidor es: ");
   System.out.println("--------------------INICIO Pub Key-------------------------
------------");
   System.out.println(new String(Base64.encode(publicKey.getEncoded())));
   System.out.println("--------------------FIN Pub Key----------------------------
---------");

   System.out.println("La clave publica que viene en de la blockchain es: ");
   System.out.println("--------------------INICIO Pub Key-------------------------
------------");
   System.out.println(response.toString());
   System.out.println("--------------------FIN Pub Key----------------------------
---------");

}catch(Exception e){
   e.printStackTrace();
} finally {
   httpClient.close();
}
```

Referencias Bibliográficas

1. “Bitcoin Project”.https://bitcoin.org/es/ . Consultado en Septiembre 2017
2. “Blockchain**:** Blueprint for a New Economy”. Melanie Swan. ISBN 9781491920473
3. https://www.nytimes.com/es/2016/08/03/snowden-y-wikileaks-debaten-por-la-forma-de-divulgar-documentos-secretos/ . Consultado en Septiembre 2017
4. “Bitcoin For Dummies”. Pryptor . ISBN 9781119076131
5. “Deep Web”. Pablo Allegritti. ISBN 9789876277587
6. “Wayniloans Project”. https://www.wayniloans.com/ . Consultado en Septiembre 2017.
7. “Cubits Project”. https://cubits.com/ . Consultado en Septiembre 2017.
8. “Kraken Project”. https://www.kraken.com/ . Consultado en Septiembre 2017.
9. “Xapo Project”. https://xapo.com/ . Consultado en Septiembre 2017.
10. “Litecoin Project”. https://litecoin.org/es/. Consultado en Septiembre 2017.
11. “Ripple Project”. https://ripple.com/ . Consultado en Septiembre 2017.
12. http://www.diariobitcoin.com/index.php/2016/12/08/oficina-del-fiscal-de-estados-unidos-pago-1-400-en-bitcoins-a-extorsionistas-de-software-malicioso/ . Consultado en Septiembre 2017 .
13. https://hipertextual.com/2017/07/nuevo-modo-extorsion-empresas-ataques-ddos . Consultado en Septiembre 2017 .
14. http://masterhacks.net/noticias/detienen-a-hacker-uruguayo-por-extorsion-a-mutualista-con-bitcoins/ . Consultado en Septiembre 2017.
15. http://es.gizmodo.com/el-director-del-mayor-banco-de-ee-uu-cree-que-bitcoin-1806528213 . Consultado en Septiembre 2017.
16. https://criptonoticias.com/colecciones/especial-un-dia-ransomware-bitcoin-volvieron-socios/ . Consultado en Septiembre 2017.
17. https://criptonoticias.com/sucesos/surcorea-alerta-amenaza-hackers-siete-bancos-pais/ . Consultado en Septiembre 2017.
18. http://eldiariodemadryn.com/2017/05/deep-web-la-via-paralela-y-libre/ . Consultado en Septiembre 2017.

19. Privacidad como acceso restringido a la información. La privacidad en el ciberespacio. Una aproximación filosófica en el entorno digital a partir de un estudio de caso sobre el buscador Google. Ana María Arconada Beasoain de Paulorena. P 90.
20. Bitcoin: A Peer-to-Peer Electronic Cash System . Satoshi Nakamoto
21. http://p2pfoundation.ning.com/forum/topics/bitcoin-open-source. Consultado en Octubre 2017
22. Certicom Research, "SEC 2: Recommended Elliptic Curve Domain Parameters."
23. http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf . Consultado en Octubre 2017.
24. http://www.itu.int/ITU-T/studygroups/com17/languages/X.690-0207.pdf. Consultado en Octubre 2017.
25. https://es.wikipedia.org/wiki/Advanced_Encryption_Standard . Consultado en Octubre 2017.
26. https://es.wikipedia.org/wiki/RIPEMD-160 . Consultado en Octubre 2017
27. https://www.genbetadev.com/seguridad-informatica/tipos-de-criptografia-simetrica-asimetrica-e-hibrida. Consultado en Octubre 2017
28. https://es.wikipedia.org/wiki/%C3%81rbol_de_Merkle . Consultado en Octubre 2017
29. https://en.bitcoin.it/wiki/Protocol_documentation#Message_types . Consultado en Noviembre 2017
30. La revolución blockchain (Blockchain Revolution), Don Tapscott & Alex Tapscott. Deusto.
31. An Integrated Reward and Reputation Mechanism for MCS Preserving Users' Privacy. Cristian Tanas, Sergi Delgado-Segura, Jordi Herrera-Joancomartí. Febrero de 2016. Data Privacy Management, and Security Assurance. pp 83-99
32. La Revolución de la tecnología de Cadenas de Bloques en la economía: Impacto en los distintos Sectores Económicos, Santiago Moreno Ismael. Marzo de 2017. EAE.
33. https://es.wikipedia.org/wiki/Anexo:Script_(Bitcoin)#cite_note-basurto-1 . Consultado en Noviembre 2017.
34. Satoshi Client Node Discovery, https://en.bitcoin.it/wiki/Satoshi_Client_Node_Discovery. Consultada Julio en 2017
35. Bitcoin Protocol Specification: Message Types, https://en.bitcoin.it/wiki/Protocol_Specification#Message_types. Consultada Julio en 2017
36. Construyendo la identidad digital : Situación actual de la firma electrónica y de las entidades de certificación. ISBN: 9788461460724.
37. Introducción a Certificados Digitales: http://www.uv.es/sto/articulos/BEI-2003-11/certificados_digitales.html. Consultada Julio en 2017
38. Fundamentos de los Certificados Digitales: https://www.securityartwork.es/2014/04/07/fundamentos-sobre-certificados-digitales-el-estandar-x-509-y-estructura-de-certificados/. Consultada Julio en 2017
39. Understanding PKI: Concepts, Standards, and Deployment Considerations. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, United States of America, 2nd edition.

40. Blockchain Cards. https://spellsofgenesis.com/index.php?action=cards_blockchain. Consultada Julio en 2017
41. Banco Central Europeo (2012), Virtual Currency Schemes. http://www.ecb.europa.eu/pub/ pdf/other/virtualcurrencyschemes201210en.pdf . Consultado en Septiembre 2017
42. Nakamoto, Satoshi (s. f.), "Bitcoin: A Peer-to-Peer Electronic Cash System". Bitcoin.org. http://bitcoin.org/ bitcoin.pdf . Consultado en Septiembre 2017.
43. https://ethereum.org/ .Consultado en Septiembre 2017.
44. https://litecoin.org/es/ .Consultado en Septiembre 2017.
45. https://ripple.com/ .Consultado en Septiembre 2017.
46. "What is Computer security?", Matt Bishop, IEEE Security and Privacy Magazine 1(1):67 - 69, 2003. DOI: 10.1109/MSECP.2003.1176998
47. http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.199.pdf. Agosto 2017
48. http://profesores.fi-b.unam.mx/cintia/Mecanismos.pdf. Agosto 2017
49. Vega, Carlos-Luis de la (1971). Aspectos de la vida diaria en China durante la dinastía Ming. Boletín de la Asociación Española de Orientalistas (VII): 39-58. ISSN 0571-3692
50. https://tools.ietf.org/html/rfc7539. Consultado en Septiembre 2017
51. http://web.archive.org/web/http://www.matematicas.net/paraiso/cripto.php?id=rsa1. Consultado en Septiembre 2017
52. https://repositorio.unican.es/xmlui/bitstream/handle/10902/3101/Jennifer%20Santamaria%20Fernandez.pdf?sequence=1. Consultado en Agosto 2017
53. http://caislab.kaist.ac.kr/lecture/2010/spring/cs548/basic/B02.pdf. Consultado en Septiembre 2017
54. http://abstract.ups.edu/download/aata-20120811.pdf. Consultado en Agosto 2017
55. R. Milner. Communication and Concurrency. Prentice Hall, 1989.
56. K. Chandy and J. Misra. Parallel Program Design: A Foundation. Addison Wesley, Reading, MA, 1988.
57. J. R. Corbin. The Art of Distributed Applications. SV, 1991.
58. G. Coulouris, J. Dollimore, T. Kindberg, Editorial: Addison Wesley,2005, 4th edition. ISBN: 0321263545
59. W. Diffie and M. E. Hellman. New directions in cryptography. IEEE Transactions on Information Theory.
60. L. Kohnfelder. Toward a Practical Public-Key Cryptosystem. M.I.T., 1978. Bachelor's Thesis.
61. R. Housley, T. Polk, W. Ford, and D. Solo. Internet Public Key Infrastructure, Part I: X509 Certificate and CRL Profile.
62. Hoskinson, Charles (2013), "The Mathematician's Defense of Bitcoin: It's Just Another Option". En PBS Newhour, 9 de octubre. http://www.pbs.org/ newshour/businessdesk/2013/10/the-mathematiciansdefense-of.html.
63. "How Do Bitcoin Transactions Work?". 26 de noviembre. http://www.coindesk.com/ information/how-do-bitcoin-transactions-work/
64. RFC 5280 - Public Key Infrastructure.
65. J. Callas, L. Donnerhacke, H. Finney, and R. Thayer. OpenPGP Message Format, 1998
66. W. Ford and M. S. Baum. Secure Electronic Commerce. Prentice Hall.
67. https://www.ietf.org/rfc/rfc5280.txt. Consultado en Noviembre 2017.
68. C. Ellison, B Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. SPKi certificate theory.

69. M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized Trust Management. In Proceedings of the Symposium on Security and Privacy.
70. https://bitcoin.org/bitcoin.pdf. Consultado en Noviembre 2017
71. Etherscan.io . Consultado en Noviembre 2017
72. Elliptic Curve Digital Signature Algorithm - Algoritmo de Firma Digital de Curva Elíptica
73. http://servicios.infoleg.gob.ar/infolegInternet/anexos/70000-74999/70749/norma.htm. Consultado en Noviembre 2017
74. https://nodejs.org/es/ .Consultado en Marzo 2018
75. https://www.javascript.com/ .Consultado en Marzo 2018
76. https://spring.io/ .Consultado en Marzo 2018
77. https://www.openssl.org/ .Consultado en Marzo 2018
78. https://github.com/indutny/elliptic .Consultado en Marzo 2018
79. https://www.java.com/es/download/ .Consultado en Marzo 2018
80. https://insomnia.rest/download/ .Consultado en Marzo 2018
81. http://acis.org.co/revista145/content/blockchain-%E2%80%9Ccadena-de-bloques%E2%80%9D-reflexiones-sobre-seguridad-y-control .Consultado en Marzo 2018
82. https://www.criptonoticias.com/educacion/cientificos-comprueban-seguridad-algoritmos-bitcoin-ataques-tecnologia-cuantica/ .Consultado en Marzo 2018
83. http://www.eleconomista.es/economia/noticias/8267899/04/17/Integracion-seguridad-y-transparencia-las-aportaciones-del-Blockchain-a-las-smart-cities.html .Consultado en Marzo 2018
84. https://qanewsblog.com/2018/01/17/blockchain-vulnerabilidades-y-sus-consecuencias/ .Consultado en Marzo 2018
85. https://igniteoutsourcing.com/publications/blockchain-security-vulnerabilities-risks/ .Consultado en Marzo 2018
86. https://www2.deloitte.com/content/dam/Deloitte/ie/Documents/Technology/IE_C_BlockchainandCyberPOV_0417.pdf .Consultado en Marzo 2018
87. https://www.investopedia.com/terms/1/51-attack.asp . Consultado en Marzo 2018
88. http://wikis.fdi.ucm.es/ELP/Ataque_de_denegaci%C3%B3n_de_servicio. Consultado en Marzo 2018
89. https://en.wikipedia.org/wiki/Convergence_(SSL) . Consultado en Abril 2018