Universidade Vigo

ARQUITECTURA BLOCKCHAIN PARA LA SECURIZACIÓN DE DISPOSITIVOS IOT MEDIANTE SMART CONTRACTS

Iago Tudela Díaz

Trabajo de Fin de Grado
Escuela de Ingeniería de Telecomunicación
Grado en Ingeniería de Tecnologías de Telecomunicación

Tutor
Manuel Fernández Veiga

2019

Universidade Vigo

Escuela de Ingeniería de Telecomunicación | Grado en Ingeniería de Tecnologías de Telecomunicación | Mención: Telemática

Arquitectura blockchain para la securización de dispositivos IoT mediante smart contracts

Autor: Iago Tudela Díaz
Tutor: Manuel Fernández Veiga
Curso: 2018-2019

1. Introducción

Desde la aparición de Internet of Things [1, 2], el mundo tecnológico ha ampliado su espectro de posibilidades de investigación y desarrollo de manera considerable. Sin embargo, antes de disfrutar de los beneficios de esta oportunidad, es necesario resolver primero los desafíos que presenta.

Concretamente, la revolución IoT está causando un crecimiento masivo en el número de dispositivos conectados a Internet. Se estima que en 2020 se alcanzarán cifras próximas a los 50 mil millones [3]. Esto plantea serios problemas a las arquitecturas en la nube, que siguen un paradigma centralizado, ya que serían incapaces de procesar cantidades de datos tan grandes. Por ello, técnicas descentralizadas como el Edge Computing [4], mejor preparadas para la distribución de carga, van a ocupar un rol fundamental en un futuro próximo [5, 6]. Otro problema que se presenta, causado por claras limitaciones de diseño, es la incapacidad para utilizar mecanismos de seguridad tradicionales por parte de los dispositivos IoT [7, 8], convirtiéndolos en posibles objetivos vulnerables [9, 10]. Estos, por su baja potencia y restricciones de consumo, necesitan soluciones más ligeras o el uso de tecnologías que sigan un planteamiento distinto.

En esta memoria se expone la implementación de una solución tecnológica a los problemas de escalabilidad y seguridad que se presentan por el auge del paradigma IoT. Para ello, como se profundiza más adelante, se utiliza una red blockchain privada que, mediante la repartición de recursos y el registro de transacciones, permita regular y proteger a estos dispositivos.

2. Definiciones

Antes de empezar a explicar los objetivos y soluciones del proyecto de manera detallada, es importante mencionar brevemente algunas definiciones. De esta manera, las ideas posteriores resultarán más claras.

2.1. IoT

Internet de las cosas (IoT) es la extensión de la conectividad de Internet a dispositivos físicos y objetos cotidianos. Esto les ofrece la capacidad de comunicarse e interactuar con otros a través de Internet y de ser gestionados o controlados de forma remota. Sus aplicaciones son extremadamente variadas y están causando un enorme impacto en la calidad de vida de los consumidores.

2.2. Función Hash Criptográfica

Una función Hash criptográfica [11] es una función matemática, adecuada para su uso en criptografía, que permite convertir un conjunto de datos, habitualmente cadenas de longitud arbitraria, en una cadena de bits de longitud fija. Es una función excepcionalmente difícil de invertir, pudiéndose considerar de sentido único.

2.3. Blockchain

Blockchain [12, 13] es una tecnología que permite la formación distribuida de un registro de transacciones. Estas están incluidas en bloques, formando una cadena mediante la aplicación de funciones matemáticas. Cada bloque contiene un hash criptográfico del bloque anterior, una marca de tiempo y datos de transacciones. Gracias a ello, se forma un árbol Merkle [14] que aporta beneficios como la resistencia a la modificación de datos o verificación de información a gran velocidad. El resultado es un registro público, distribuido y capaz de almacenar transacciones entre dos partes de manera eficiente, verificable y permanente.

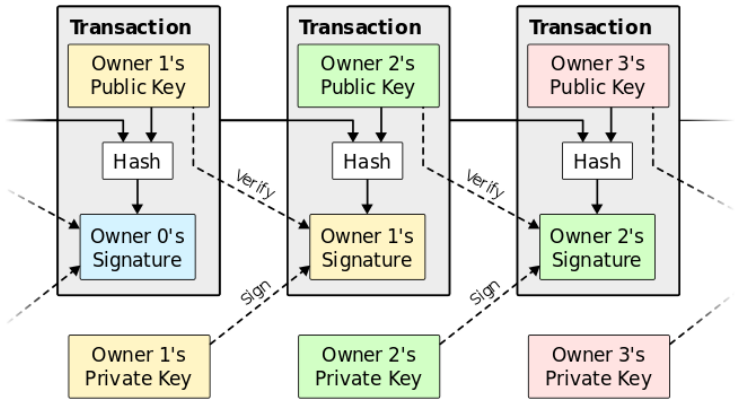

Figura 1: Representación de la cadena de bloques [15].

Esta tecnología fue inicialmente destinada al uso de criptomonedas, como Bitcoin [15, 16]. Sin embargo, sus propiedades ofrecen multitud de beneficios en paradigmas muy diversos. Concretamente, su naturaleza distribuida y las mejoras de seguridad son especialmente útiles para las arquitecturas IoT [17].

2.4. Smart Contract

Un contrato inteligente es un programa software destinado a facilitar, verificar o hacer cumplir digitalmente la negociación o el cumplimiento de un contrato. Los contratos inteligentes permiten la realización de transacciones verificables sin la necesidad de intervención de terceros. Estas transacciones son rastreables e irreversibles.

En una red blockchain con soporte para contratos inteligentes los nodos tienen el código de cada programa y, cuando uno de ellos deba ejecutarse, se lanzará simultáneamente en toda la red. Estos contratos son públicos e inalterables, de tal manera que su funcionamiento está garantizado.

2.5. Ethereum

Ethereum [18] es una plataforma de computación distribuida basada en blockchain, pública y de código abierto. Su gran innovación es la EVM (Ethereum Virtual Machine), una máquina virtual, considerada Turing completa, que permite la ejecución de cualquier programa si cuenta con el tiempo y los recursos necesarios. Esta propiedad permite ejecutar contratos inteligentes [19, 20] y ofrece muchísima flexibilidad a la hora de realizar proyectos descentralizados. Además, Ethereum utiliza Ethash [21] como función PoW (Proof of Work), resultando en una confirmación de transacciones mucho más rápida que las soluciones anteriores.

Se utiliza Ether como token económico intercambiable. Sin embargo, esta tecnología introduce un elemento adicional, llamado Gas, que protege al sistema de las fluctuaciones del mercado. En el Anexo B se realiza una explicación moderadamente detallada sobre Ethereum, el papel de la EVM en el sistema y el funcionamiento del Gas.

3. Objetivo

El objetivo de este proyecto es el de buscar, estudiar e implementar una solución adecuada a los problemas de escalabilidad y seguridad a los que se enfrenta el paradigma IoT.

Para ello se plantea una arquitectura blockchain distribuida, situada entre la nube y los usuarios, que regule, gestione y monitorice la actividad de dispositivos IoT mediante el uso de contratos inteligentes y el reparto de recursos. También se proponen mecanismos que permiten la descarga de actualizaciones de manera segura y el almacenamiento de la información de uso de la red por parte de dichos dispositivos.

Al seguir una aproximación distribuida, solventamos los problemas de escalabilidad a los que se enfrenta la nube centralizada, ya que toda la parte computacional se realizaría en un punto más próximo a los usuarios. Por otra parte las ventajas inherentes a la tecnología blockchain ofrecen una cantidad importante de soluciones a problemas de seguridad [22], como un registro de actividades o la impermutabilidad de los datos.

4. Solución

4.1. Elementos del Sistema

4.1.1. Red Blockchain

La red blockchain necesita ser capaz de soportar la ejecución de contratos inteligentes para poder gestionar la lógica de repartición de recursos e implementar los mecanismos de seguridad. Además, sería recomendable que fuese eficiente desde el punto de vista computacional y debe poder ser gestionada por la entidad que quiera implementarla.

Por ello, se ha decidido utilizar una red Ethereum. Esta tecnología permite ser configurada de manera privada, decidiendo qué direcciones forman parte de la red y cuáles no. También es posible crear nuevas cuentas que poder asignarles a los dispositivos IoT del sistema y ajustar las características de la red sin ningún tipo de dificultad. Como se menciona al principio, es posible utilizar contratos inteligentes con esta tecnología. Por tanto, Ethereum cubre todas las necesidades identificadas para el desarrollo del proyecto.

4.1.2. Smart Contract

En el contrato se implementan todos los mecanismos de seguridad, registro de información y gestión de versiones. Por tanto, en él se almacena la mayor parte de la lógica del sistema. Su funcionamiento se detalla en profundidad más adelante en una sección propia.

Para la programación del contrato se utiliza Solidity [23]. Este lenguaje de programación de alto nivel está diseñado para desarrollar contratos inteligentes que se ejecuten en la Ethereum Virtual Machine. Además, soporta herencia, estructuras de datos complejas, mapeado y tiene un gran apoyo por parte de la comunidad, facilitando la búsqueda de información. El compilador de Solidity se llama solc. Cuenta con varias herramientas de desarrollo, como Remix [24]; una plataforma que permite el diseño, compilado, depuración de errores, emulación y prueba de contratos en tiempo real.

4.1.3. Servidores

Los servidores son los nodos de la red blockchain que realizan las operaciones de minado. Es decir, son los encargados de realizar las operaciones que los dispositivos IoT no tienen capacidad para ejecutar. Tienen instalado Go Ethereum [25] (Geth) que les permite la ejecución de la red blockchain. Son elementos con unas características muy superiores a la de los clientes. Para el desarrollo del proyecto se utiliza un ordenador con 8GB RAM, 1TB de memoria y un procesador Intel Core I5 de séptima generación.

4.1.4. Clientes

Los clientes de esta arquitectura son los dispositivos IoT. Estos, al no tener la capacidad computacional para realizar operaciones complejas, necesitan apoyarse en los servidores de la red Ethereum.

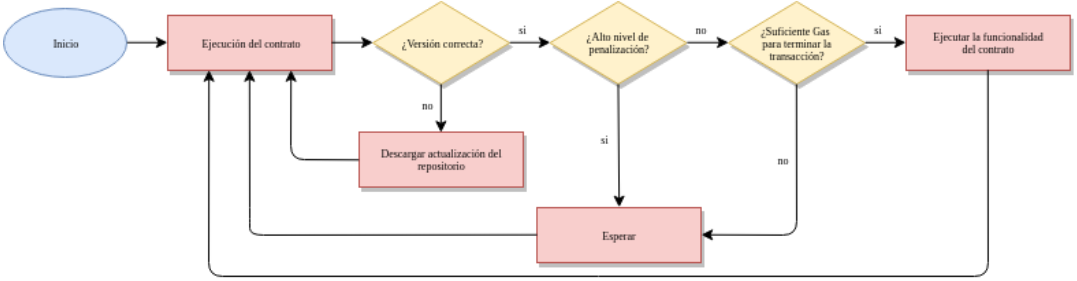

Figura 2: Funcionamiento de los clientes.

Los clientes utilizan un programa escrito en Java, que simula el comportamiento de un dispositivo IoT comercial, pidiendo recursos a la red blockchain de manera periódica. Tienen instalado Go Ethereum en modo ligero; es decir, no hacen operaciones de minado.

No todos los dispositivos IoT tienen la capacidad de utilizar un software de este tipo. En estos casos se podría implementar un proxy, situado en el mismo nodo, que actuaría como intermediario entre esos dispositivos y la red. El desarrollo del proyecto se limita a los dispositivos que sí que tienen esta capacidad, utilizando una Raspberry Pi 3B+.

4.1.5. Interfaz de los Smart Contracts

Para poder utilizar las funcionalidades que ofrecen los smart contracts de la red Ethereum es necesario que los clientes se comuniquen de alguna manera con dicha red. Esto lo hacen a través de una interfaz llamada web3j [26]. Web3j es una librería de Java altamente modular y reactiva que permite la integración con nodos de la red Ethereum y la migración y ejecución de contratos inteligentes.

4.1.6. Migración de Contratos

Antes de utilizar un contrato en la red blockchain, es necesario migrarlo para que los nodos que la forman tengan el código que debe ser ejecutado. Esto se hace mediante Truffle [27], un entorno de desarrollo de código libre idóneo para tareas de migración, compilado y prueba de contratos inteligentes en redes blockchain que utilicen la EVM.

4.2. Arquitectura

Como se menciona anteriormente, los dispositivos IoT tienen instalado Go Ethereum en modo ligero. Esto les permite comunicarse dinámicamente con otros servidores de la red blockchain en caso de que falle el más próximo. A cada uno de los elementos se le asigna una cuenta con su dirección correspondiente, una cantidad de Gas suficiente para utilizar el sistema y las claves pública y privada.

Los dispositivos se comunican con la red blockchain utilizando sus credenciales a través de la interfaz web3j integrada en el cliente programado en Java. Las credenciales se almacenan en formato JSON [28] dentro del propio dispositivo, siendo necesario control del mismo para poder utilizar su cuenta.

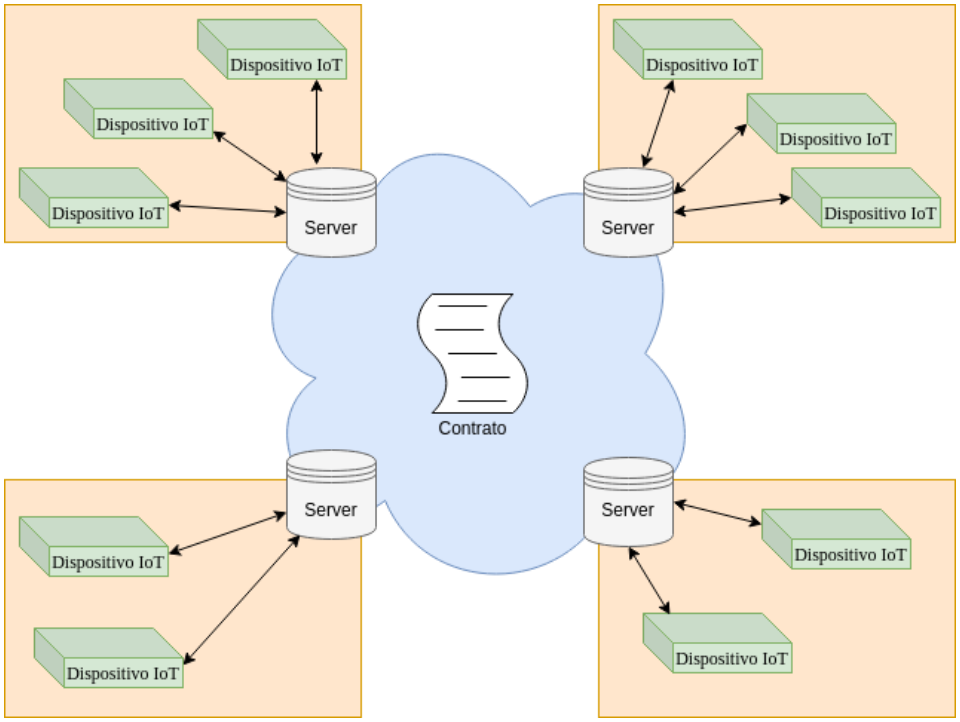

Figura 3: Ejemplo de topología de red Ethereum.

Los servidores ejecutan los contratos inteligentes, almacenando las transacciones de los dispositivos mediante la generación de los bloques de la cadena. Toda la actividad de los dispositivos IoT de la arquitectura quedaría registrada, permitiendo su revisión posterior. Al ser una red privada, no es necesario la utilización de algoritmos PoW laboriosos, dando lugar a un uso más eficiente de los recursos.

En la figura 3 podemos apreciar un ejemplo sencillo de topología. Esta arquitectura puede limitarse a un único servidor que gestione el uso de los dispositivos IoT o extenderse a un entorno mayor en el que múltiples servidores realicen las operaciones computacionales conjuntamente, como una universidad con varias facultades y escuelas. Para el desarrollo del proyecto, la arquitectura se limita a dos dispositivos IoT y un servidor.

4.3. Lógica del contrato

La lógica del contrato es la parte más compleja del proyecto. En ella se soluciona la repartición de recursos a los dispositivos IoT, el control de versiones, el soporte para publicar actualizaciones seguras, la regulación de la actividad de los dispositivos IoT y el registro de la información de estado de los elementos de la red. El diagrama de flujo del contrato está representado en la figura 4.

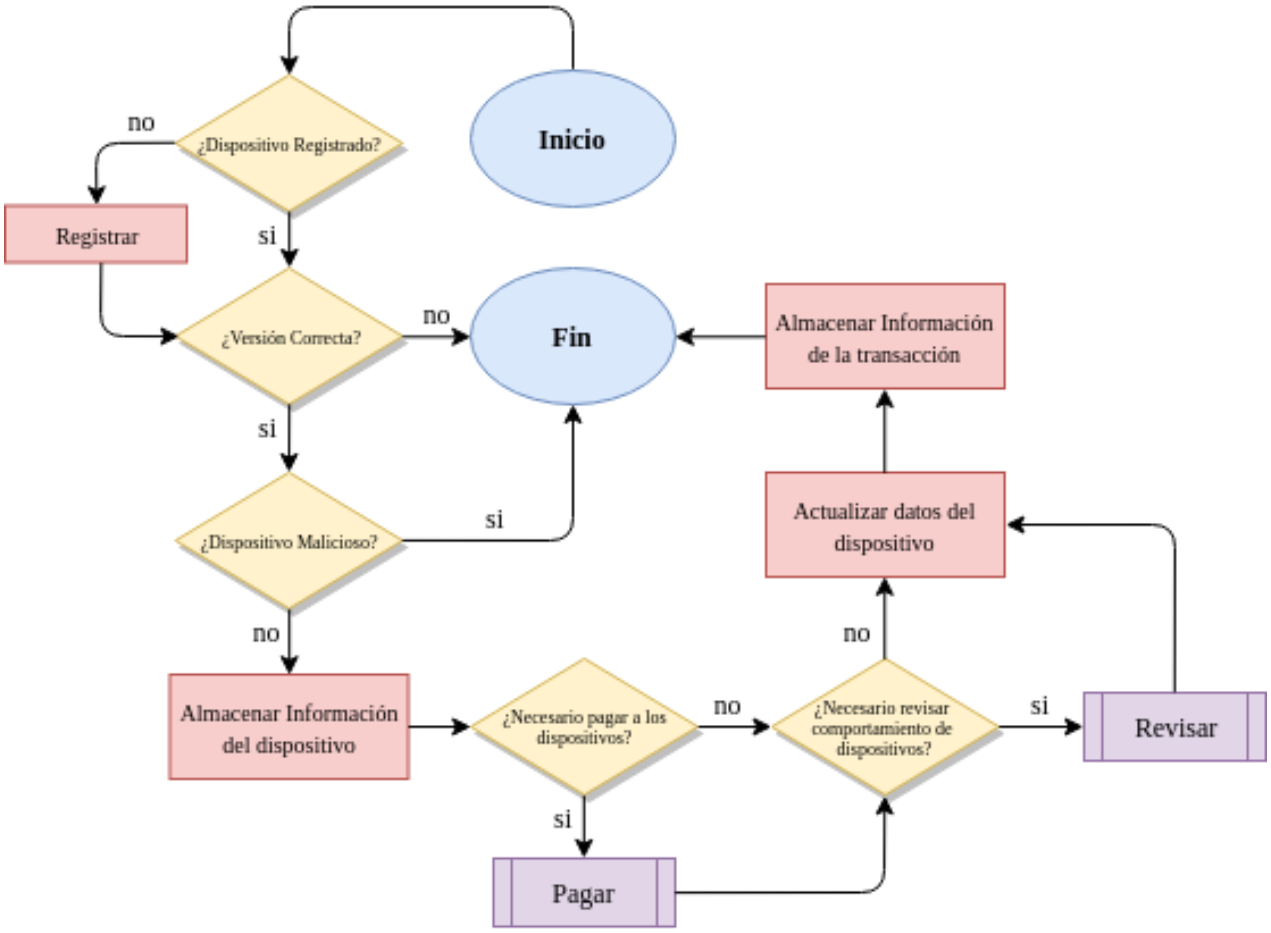

Figura 4: Diagrama de flujo del contrato.

Cuando se llama al contrato, lo primero que se comprueba es el conocimiento de la existencia del dispositivo. El contrato registra la información de cada cliente asociándola con su dirección Ethereum en un mapa. Este mecanismo permite identificar a cada dispositivo de manera inequívoca, ya que conocer la clave privada es estrictamente necesario para poder utilizar una cuenta. En caso de que sea la primera vez que el dispositivo participa en la red, se crearía la estructura de datos en el mapa en la que se guardaría la información de su actividad. Esta operación de registro se realiza una única vez por dispositivo.

Posteriormente, se comprueba que la versión que tiene el cliente sea la adecuada. Si no lo fuese, se terminaría la transacción y se le devolvería la URL del repositorio de Firebase [29], un servicio que permite el almacenamiento de datos en tiempo real, en el que se encuentra toda la información necesaria para la actualización de su software. Después se comprueba el comportamiento del dispositivo. En caso de que, debido a actividades irregulares o maliciosas haya sido penalizado de manera notable, terminaría la transacción.

A continuación, se almacena la información indispensable para el funcionamiento del sistema y, entonces, se realizan las operaciones que el dispositivo IoT ha pedido. Finamente, se guardan los datos necesarios para controlar el comportamiento del cliente a lo largo del tiempo y la cantidad de gas que han consumido por la utilización del sistema.

La revisión de seguridad y el reparto de recursos no se realizan siempre que se llama a un contrato, pues se consumirían una cantidad de recursos computacionales demasiado elevados. Estas partes sólo son ejecutadas cuando se cumplen ciertas condiciones. Blockchain es una tecnología distribuida y, por tanto, realizar operaciones basándonos en un periodo temporal no es posible sin la ayuda de elementos centralizados externos. Por ese motivo se utiliza el número de bloque en vez de una referencia temporal. Cuando el número de bloques entre la última revisión y el número de bloque de la transacción que se esté llevando a cabo sea mayor que cierta variable, se realiza la comprobación de seguridad o el reparto de recursos correspondiente.

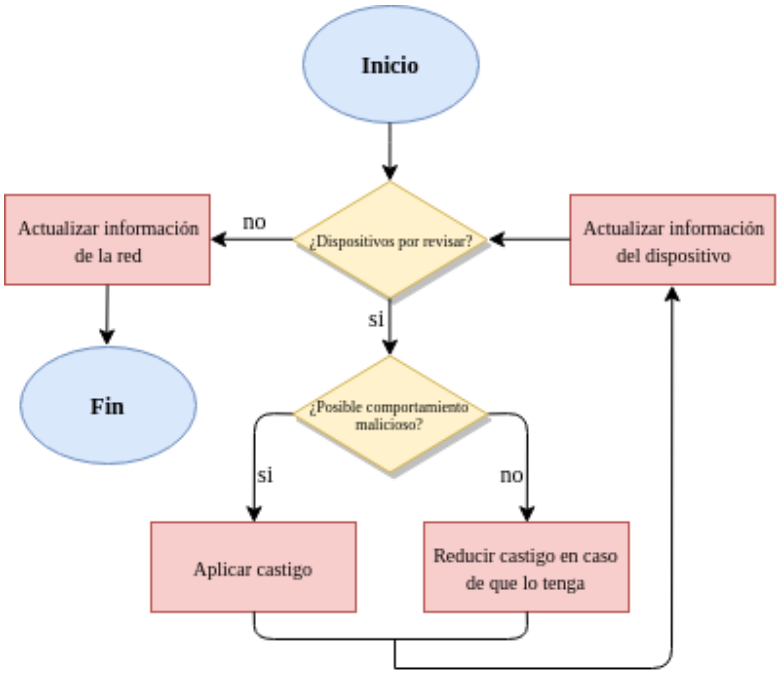

Figura 5: Diagrama de flujo de la función de revisión de comportamiento.

Se puede observar un diagrama de flujo de las comprobaciones de seguridad en la figura 5. Consiste en la iteración del mapa de dispositivos, revisando el comportamiento de los elementos en el último intervalo de bloques no observado, decidiendo si su actividad es irregular y actualizando el registro de actividad de cada dispositivo. En caso de que fuese maliciosa, se le aplicaría una penalización multiplicada por la severidad, variable almacenada en el contrato, del sistema. En caso contrario, si ese elemento tuviese una penalización previa, se reduciría.

En el desarrollo de este proyecto se considera que todos los dispositivos IoT son iguales y siguen la misma lógica de mi código Java. Esto no siempre sería así en un escenario real, pudiendo darse el caso de que haya múltiples dispositivos, con diferentes características y funciones distintas. En ese caso, las características de la red e intervalos de seguridad deberían ser ajustados de manera personalizada. Por simplicidad, tiempo y recursos se considera que superar la proporción de bloques del intervalo que le corresponde es considerado como comportamiento malicioso. Es decir, si hubiese dos dispositivos registrados en el contrato y teniendo en cuenta que, a priori, su comportamiento es el mismo, si uno realizase el 80 % de las transacciones, se le aplicaría una penalización proporcional a la tolerancia de la red.

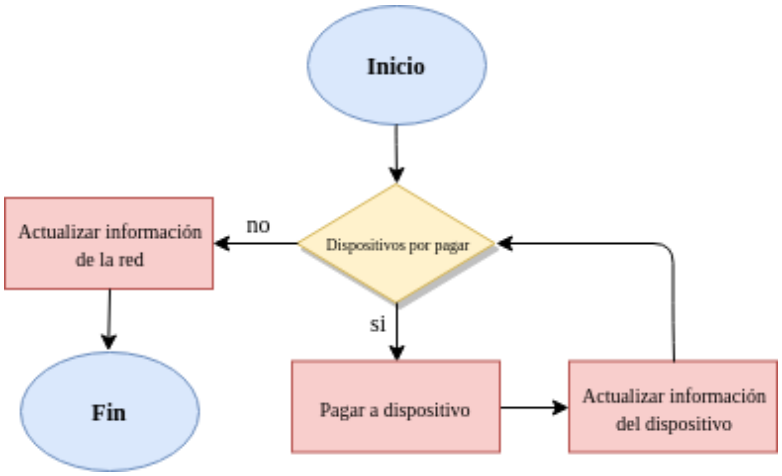

Figura 6: Diagrama de flujo de la función de repartición de recursos.

La lógica de la función de repartición de recursos se puede observar en la figura 6. Esta operación está formada por un bucle en el que el contrato le reembolsa a cada dispositivo la cantidad de Gas que haya consumido en el último intervalo, actualizando a su vez la información de los clientes almacenada en el contrato.

Hay que tener en cuenta que el nivel de penalización de un dispositivo no sólo impide el uso del sistema sino que, en niveles más bajos, divide la cantidad de Gas que se reembolsa durante la repartición de recursos. Es decir, el contrato no se limita a bloquear los dispositivos maliciosos, también regula el crédito de su cuenta, causando que en el futuro se queden sin el Gas suficiente para acceder al sistema.

El contrato también incluye funcionalidades adicionales útiles para el estudio del sistema y tareas de auditoría. Entre ellas encontramos métodos que permiten retirar información del balance del cliente y del contrato, recuperar la información de la actividad de un dispositivo desde su entrada en el sistema y mecanismos que permiten abonar Gas al contrato mediante transacciones. Esto último es muy útil durante el desarrollo pero en el diseño final el origen del balance se soluciona configurando el destino de las recompensas de los mineros.

5. Pruebas y resultados

Las pruebas realizadas tienen como objetivo comprobar el estado de las siguientes funcionalidades: control de versiones y descarga de su información de manera automática, registro de la actividad de los dispositivos del sistema, detección de actividades maliciosas y reparto de recursos.

En la primera prueba se verifica que un dispositivo es capaz de recoger datos de un repositorio indicado por la red blockchain cuando la versión de su software no es la adecuada. Primero debemos ejecutar la función del contrato, solamente accesible si es llamada desde la cuenta de administrador, que permite modificar la última versión. Entonces, todo dispositivo IoT que acceda al contrato con la información de su versión es rechazado con un mensaje en el que se indica la URL del repositorio de Firebase y descargaría los contenidos de la actualización. Si intentásemos llamar a esa función sin tener las credenciales adecuadas, seríamos rechazados del contrato por no tener los privilegios requeridos.

Blockchain es una tecnología que, intrínsecamente, registra todas las transacciones realizadas en la red. Además, se aprovecha la posibilidad de utilizar estructuras de datos complejas en Solidity para almacenar información de la actividad de los clientes en cada intervalo de bloques inspeccionado durante la revisión de seguridad. También se guardan datos de los dispositivos en las transacciones regulares. Para comprobar el funcionamiento del registro del comportamiento de los dispositivos se ejecutan los métodos de recuperación de información implementados en el contrato y se verificaa el estado de los datos almacenados.

La figura 7 justifica la repartición de recursos. En ella, podemos observar la cantidad de Gas que tiene un dispositivo a lo largo de la cadena de bloques. Los recursos del cliente se consumen de manera gradual a medida que se realizan transacciones. Como en este caso es el único dispositivo del sistema, el contrato no le pone ninguna limitación en el número de transacciones. Una vez se mina el número de bloques entre reembolsos que ha sido configurado se activa la función de repartición de recursos, permitiendo al dispositivo la recuperación del crédito perdido.

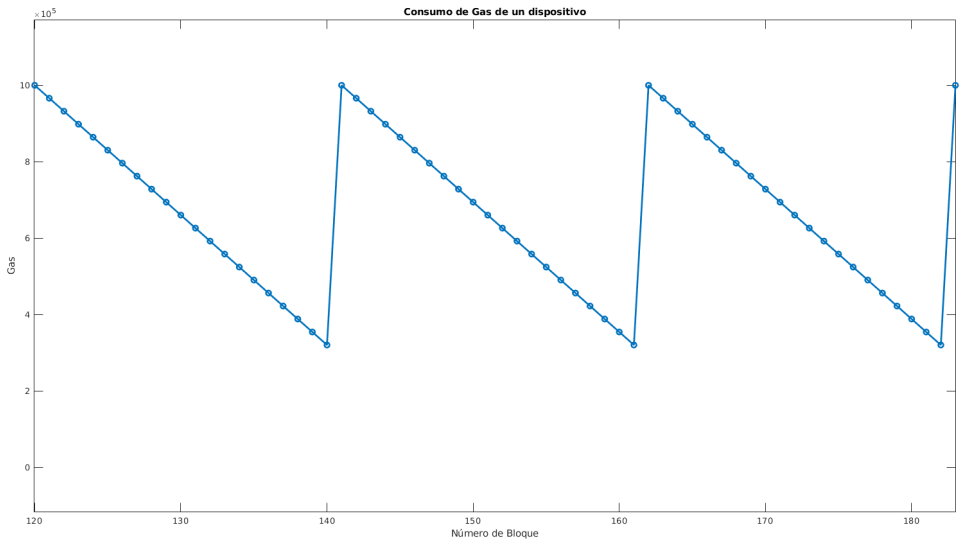

Figura 7: Consumo de Gas de un único dispositivo con intervalos de 20 bloques entre pagos.

En la figura 8 se justifica el funcionamiento del sistema de penalizaciones y de reembolso de Gas. En cada mitad de la gráfica se puede observar un intervalo de operaciones entre pagos. El cliente azul consume un porcentaje de transacciones mayor del que le corresponde y cuando se realiza la comprobación del comportamiento de los dispositivos durante una transacción del cliente naranja al final del segundo intervalo (donde se aprecia la transacción con mayor consumo de gas) se detectan las irregularidades del cliente azul y se le aplica una penalización. Esto resulta en la división del reembolso de Gas recibido durante la repartición de recursos.

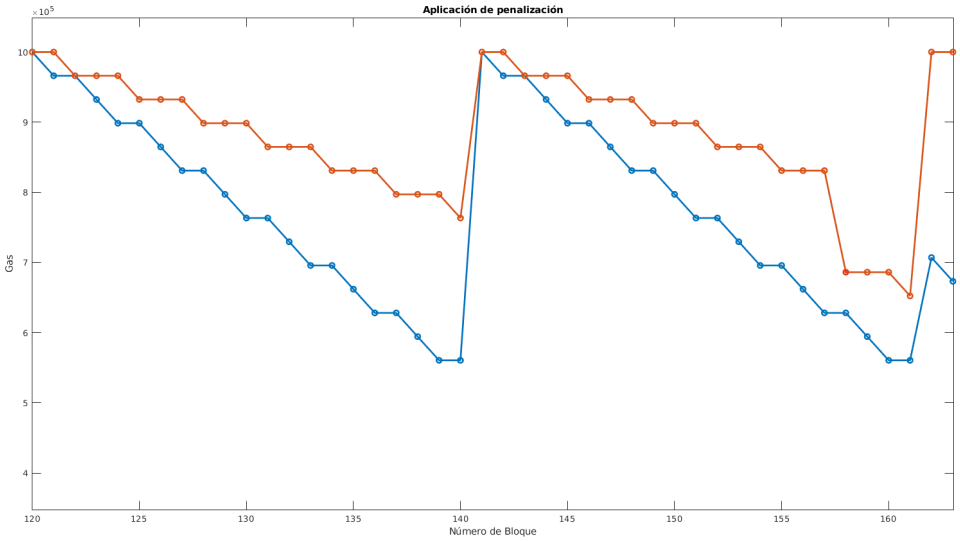

Figura 8: Impacto de una penalización en el reembolso de Gas.

El sistema actúa de manera diferente en las operaciones iniciales. Cuando el contrato se ejecuta por primera vez, inicializa las variables que regulan los intervalos de bloque, aumentando el coste de Gas de esa transacción. Además, durante el primer acceso de un cliente al contrato, se realiza el registro del mismo añadiendo una estructura de datos personalizada en el mapa de dispositivos. Las figuras 7 y 8 muestran escenarios en los que el sistema está estabilizado, buscando representar la prueba de una manera más ilustrativa.

6. Líneas Futuras

Se han alcanzado todos los objetivos propuestos durante el planteamiento del proyecto; sin embargo, mientras se realizaba la implementación, fue posible valorar nuevas funcionalidades, cambios u opciones alternativas que podrían resultar interesantes. A continuación se exponen brevemente posibles líneas de actuación en trabajos de la misma naturaleza.

Una posible mejora sería la diferenciación de dispositivos. Actualmente, se consideran todos los dispositivos IoT de la red como iguales, necesitando la misma cantidad de recursos. Sin embargo, sería viable recoger información durante el registro y calcular nuevos umbrales de actividad teniendo en cuenta esos datos. Esto permitiría una mayor precisión a la hora de comprobar comportamientos maliciosos en escenarios más variados. Es necesario tener datos concretos del sistema y especificaciones de los dispositivos que se van a utilizar para poder plantear una arquitectura que incluya esta funcionalidad.

También sería interesante desarrollar un proxy que actúe como intermediario, gestionando las comunicaciones entre la red blockchain y los dispositivos IoT que no fuesen capaces de soportar Go Ethereum. El proxy podría ser uno de los servidores o un elemento a parte con Geth en modo ligero. Esta propuesta generaría una arquitectura semi-centralizada en la que los intermediarios se convertirían en puntos críticos del funcionamiento del sistema. Un ataque exitoso a uno de ellos perjudicaría directamente a los dispositivos IoT que se comuniquen con él. En este escenario es vital blindar el proxy de manera apropiada.

Una alternativa sería sustituir los servidores previos por servidores Edge Computing. Estos lanzarían una máquina virtual con un servidor Ethereum, siguiendo la misma arquitectura planteada. De esta manera, tendríamos facilidades para lanzar máquinas virtuales con aplicaciones determinadas y, en vez de realizar operaciones directamente en el Smart Contract, podríamos permitirle a los dispositivos el acceso a las mismas. Ejemplos de tecnologías Edge Computing que permiten hacer esto son HomeCloud [30] o Cloudlet [31].

Una última idea sería integrar software que posibilite realizar operaciones centralizadas. Un ejemplo sería Oraclize [32], que permitite llamadas a servicios de terceros, cálculos temporales precisos y generación de números aleatorios. No ha sido necesario incluir un elemento con estas características para el desarrollo del proyecto pero sería interesante estudiar en mayor profundidad una arquitectura que pueda llamar a elementos externos a través del mecanismo de eventos que incluye Solidity.

7. Conclusiones

En esta memoria se ha expuesto una arquitectura segura para dispositivos IoT utilizando una red Ethereum privada. El comportamiento de los dispositivos se regula mediante la ejecución de un contrato inteligente, permitiendo la detección de actividades irregulares y el control de su actividad por medio del reembolso de recursos. El prototipo implementado también incluye mecanismos de publicación y descarga de actualizaciones de manera segura y funcionalidades que permiten mantener un registro inmutable de las operaciones de los elementos del sistema, facilitando futuras tareas de auditoría.

Actualmente, los contratos inteligentes en Ethereum todavía tienen mucho camino por recorrer. Solidity es un lenguaje joven y con mucho potencial, pero todavía es susceptible a modificaciones y cambios repentinos que pueden resultar en complicaciones durante la implementación de un sistema.

En conclusión, se ha podido observar que es posible utilizar blockchain para generar arquitecturas para servicios IoT manteniendo un coste computacional razonable. Las carácterísticas inherentes de una red blockchain sumadas a una perspectiva cercana a los usuarios resultan en una aproximación con mucho potencial para resolver problemas de escalabilidad y seguridad.

Anexos

A. Despliegue de la arquitectura

El objetivo de este anexo es explicar cual es el proceso de implementación de la arquitectura del proyecto. Primero se explica la configuración e inicialización de los servidores, luego se comenta el despliegue del contrato y por último se expone la preparación e inicialización de los dispositivos IoT.

El primer paso es instalar Go Ethereum en los nodos que van a actuar como servidores. Una vez hecho esto, es necesario crear y configurar el primer bloque de la red blockchain, llamado bloque génesis. Este archivo, escrito en formato JSON, contiene información de configuración como el identificador de la red, la dificultad de minado o el límite de Gas. El límite de Gas es un parámetro que debe ser considerado con gran precaución; un valor demasiado alto podría causar grandes perdidas de ether a los dispositivos IoT en caso de fallo en una transacción y un valor demasiado bajo limitaría el número de operaciones del contrato, pudiendo causar problemas durante los momentos que consumen más recursos como la revisión del comportamiento de los dispositivos. La dificultad tiene un efecto directo en la velocidad de minado de la red, por tanto, se ha decidido utilizar un valor intermedio que evite que el minado de bloques sea excesivamente lento y afecte al sistema o que sea muy alto y los nuevos bloques aumenten la memoria utilizada de manera excesiva.

A continuación se inicializan los servidores blockchain con un código JavaScript [33] de auto-mining, impidiendo que la red cree bloques si no hay transacciones pendientes, evitando así el consumo de memoria y procesado malgastado en el minado de bloques vacíos. También se lanza un servidor RPC [34] en una IP diferente a la localhost, para permitir comunicaciones con otros dispositivos. Una vez hecho esto se sincronizan los servidores, se les ordena empezar a minar y se crean las cuentas que van a ser utilizadas por los clientes IoT.

Después, se escribe la dirección de la cuenta del administrador de la red en el contrato para regular que sólo ese dispositivo pueda utilizar funciones restringidas a los usuarios convencionales, como la actualización de la versión del software. Entonces, se compila con solc el contrato, generando los archivos .bin y .abi y, a partir de ellos y mediante web3j, se crea una clase Java wrapper que permite interactuar al cliente con el contrato. Finalmente, se incluye el código generado en el proyecto donde se encuentra el cliente Java y se migra el contrato a la red blockchain utilizando comandos de Truffle.

Lo primero que hay que hacer para poner en marcha a los dispositivos IoT es instalar todo el software necesario para su correcto funcionamiento. Es decir, los clientes necesitan Java para ejecutar el programa que simula a un dispositivo doméstico, Maven [35] para facilitar la descarga de dependencias y Go Ethereum para comunicarse con la red blockchain. En último lugar, se descargan las dependencias del proyecto, se inicializa Go Ethereum en modo ligero, se sincronizan los dispositivos con la red y se ejecutan los clientes. El sistema funcionaría automáticamente a partir de ese momento.

B. Ethereum, EVM y Gas

Antes de la aparición de Ethereum, las aplicaciones blockchain se diseñaban para cumplir objetivos específicos mediante operaciones muy concretas. Un ejemplo de esto son las criptomonedas, como Bitcoin, desarrolladas exclusivamente para actuar como dinero P2P (peer-to-peer) digital. Esa limitación de las operaciones dificultaba el trabajo a los desarrolladores a la hora de crear nuevas tecnologías partiendo de aproximaciones previas. Con el objetivo de resolver este problema, nació Ethereum.

Su principal innovación es que permite ejecutar cualquier programa, independientemente del lenguaje en el que esté desarrollado, si se le asigna suficiente cantidad de tiempo y memoria. Esto es posible gracias a la EVM (Ethereum Virtual Machine), un software Turing completo que se ejecuta en la red Ethereum. Dicha novedad facilita enormemente el proceso de creación de aplicaciones blockchain, permitiendo el desarrollo de infinidad de proyectos diferentes en una única plataforma.

Además, al poder ejecutar cualquier tipo de código manteniendo todos los beneficios de seguridad de las redes blockchain, se puede utilizar esta tecnología para construir aplicaciones enteramente descentralizadas (dApps). Todo servicio centralizado podría ser descentralizado, dando lugar a multitud de posibles mejoras en sectores como el industrial, administrativo o financiero.

Las operaciones en la red Ethereum suponen un esfuerzo. En consecuencia, toda transacción o ejecución de un contrato inteligente requiere una cantidad económica proporcional al trabajo que los nodos necesiten para minar dicha operación. Esa cantidad será utilizada como recompensa monetaria por el esfuerzo computacional que los mineros hallan realizado.

Operation	Gas	Description
ADD/SUB	3	Arithmetic operation
MUL/DIV	5	
ADDMOD/MULMOD	8	
AND/OR/XOR	3	Bitwise logic operation
LT/GT/SLT/SGT/EQ	3	Comparison operation
POP	2	Stack operation
PUSH/DUP/SWAP	3	
MLOAD/MSTORE	3	Memory operation
JUMP	8	Unconditional jump
JUMPI	10	Conditional jump
SLOAD	200	Storage operation
SSTORE	5,000/ 20,000	
BALANCE	400	Get balance of an account
CREATE	32,000	Create a new account using CREATE
CALL	25,000	Create a new account using CALL

Figura 9: Tabla con el coste de Gas de cada operación. [36]

Ethereum tiene una moneda llamada Ether, sin embargo, al igual que el Bitcoin, su valor está sujeto a las fluctuaciones del mercado. El gas es un elemento que se introduce para que el coste de una transacción o la ejecución de un programa sea invariante en el tiempo. Toda operación, como se aprecia en la figura 9, tiene una cantidad de gas asociada, siendo las más exigentes a nivel computacional aquellas con un mayor coste y las más triviales, como una suma o una resta, las más asequibles.

Esta dualidad permite que Ethereum pueda ajustar dinámicamente la exigencia económica de la red en función de sus necesidades, alterando la cantidad de Ether por unidad de Gas necesaria para realizar una operación. Es decir, al igual que cualquier producto del mercado, cuanto más alta es la demanda, más elevado será el precio del Gas.

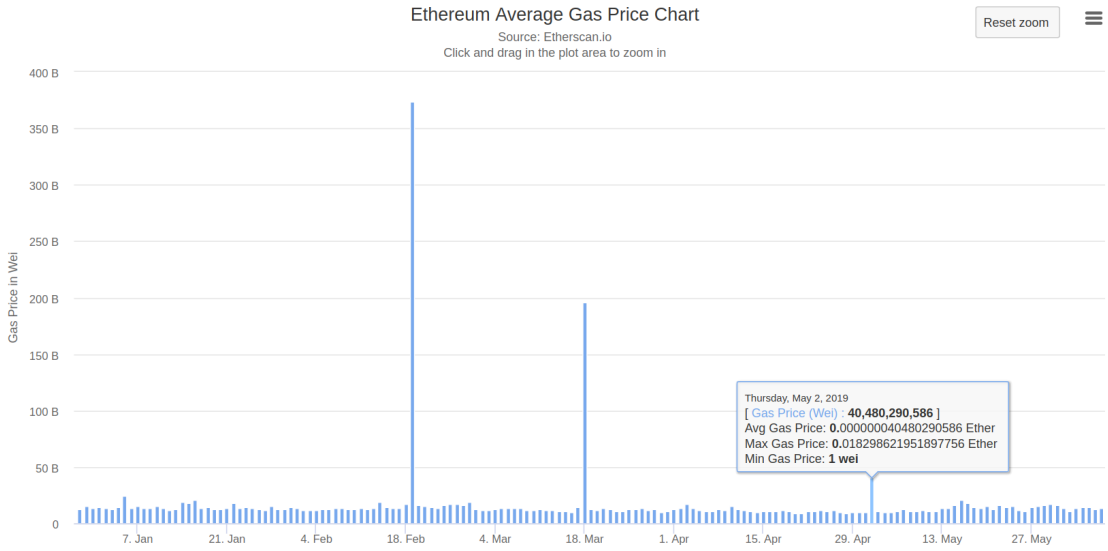

Figura 10: Gráfica con el precio medio diario del Gas en la red Ethereum [37].

Es importante mencionar que, aunque el precio del Gas está regulado por la red Ethereum, los usuarios pueden decidir tanto el precio como el límite de Gas que quieren pagar a la hora de realizar una transacción. Por tanto, un usuario puede acelerar la ejecución de una operación incrementando el precio al que paga el Gas, pues será más probable que los mineros prioricen esa transacción ya que su recompensa será mayor. Por el contrario, si se selecciona un valor inferior al umbral marcado por Ethereum los mineros considerarían la transacción como no rentable, la marcarían como inválida y no sería completada; sin embargo, el minero si que cobraría la cantidad de gas equivalente al trabajo realizado.

En conclusión, Ethereum es una red blockchain de código abierto que cuenta con los beneficios de seguridad, inalterabilidad de los datos, descentralización y eliminación de los intermediarios que proporcionaban aproximaciones anteriores. Además, esta tecnología, gracias a la EVM y a la presencia del Gas, permite la ejecución de cualquier programa o contrato en la red e incluye mecanismos que permiten un correcto funcionamiento del sistema independientemente de la volatilidad del mercado.

C. Estado del arte

El número de investigaciones y proyectos relacionados con blockchain y con IoT es cada vez mayor. Dichas tecnologías están en un punto álgido de crecimiento y popularidad, ideal para sacar a relucir todo su potencial. Por tanto, no es de extrañar que la aparición de propuestas que combinen ambas tecnologías esté al alza. Los documentos y proyectos con estas características son muy abundantes, así que en mi análisis del estado del arte me limitaré a aquellas publicaciones más similares a mi implementación.

- **EdgeChain**: An Edge-IoT Framework and Prototype Based on Blockchain and Smart Contracts [38]: esta publicación propone una arquitectura blockchain situada en servidores Edge Computing. Se gestiona la repartición de recursos a dispositivos IoT mediante contratos inteligentes y asignación de máquinas virtuales capaces de simular aplicaciones complejas.
- **Hybrid-IoT**: Hybrid Blockchain Architecture for Internet of Things - PoW Sub-blockchains [39]: Diseño de una arquitectura en la que grupos de dispositivos IoT forman redes sub-blockchain PoW (Proof of Work) que se conectan con un elemento de interconexión BFT (Byzantine fault tolerance [40]), como Polkadot [41] o Cosmos [42].
- **IoTChain**: A Blockchain Security Architecture for the Internet of Things [43]: Planteamiento de un sistema que combina la arquitectura OSCAR [44] y una red blockchain, que reemplaza a un servidor de autorización ACE [45], proporcionando acceso a recursos con autorización y seguridad.
- **LSB**: A Lightweight Scalable BlockChain for IoT Security and Privacy [46]: LSB implementa un algoritmo de consenso de IoT que elimina la necesidad de minar bloques en la red blockchain mediante PoW. Este proyecto propone un método distribuido de fidelidad en el que el tiempo de procesamiento para validar nuevos bloques disminuye gradualmente a medida que los nodos acumulan confianza entre sí.
- **DistBlockNet**: A Distributed Blockchains-Based Secure SDN Architecture for IoT Networks [47]: Arquitectura IoT distribuida que consiste en una red definida por software (SDN) que utiliza blockchain con el objetivo de proporcionar seguridad.

Entre las publicaciones expuestas, EdgeChain es la que más similitudes tiene con mi proyecto. Su documentación ha aportado múltiples ideas y reflexiones, mayormente comentadas en el apartado de líneas futuras de esta memoria. También merecen una mención especial la gran cantidad de publicaciones de divulgación y proyectos académicos relacionados con este campo que forman una excelente fuente información, estimulando el desarrollo de futuras ideas y lineas de investigación.

Referencias

[1] Z. H. Ali, H. A. Ali and M. M. Badawy, "Internet of Things (IoT): Definitions, Challenges and Recent Research Directions," 2015. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.735.8354&rep=rep1&type=pdf

[2] S. Madakam, R. Ramaswamy and S. Tripathi, "Internet of Things (IoT): A Literature Review," 2015. [Online]. Available: https://file.scirp.org/pdf/JCC_2015052516013923.pdf

[3] Ericsson Inc, "CEO to Shareholders: 50 Billion Connections 2020," 2010. [Online]. Available: http://www.ericsson.com/thecompany/press/releases/2010/04/1403231

[4] W. Shi, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal 3.5*, 2016.

[5] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog Computing and its role in the Internet of Things," *Proc. 1st Ed. ACM MCC Workshop Mobile Cloud Computing*, 2012.

[6] J. Pan and J. McElhannon, "Future Edge Cloud and Edge Computing for Internet of Things Applications," *IEEE Internet of Things Journal, Special Issue on Fog Computing in IoT*, 2018.

[7] R. Roman, "On the features and challenges of security and privacy in distributed internet of things," *Computer Networks*, vol. 27, 2013.

[8] A. Mosenia and N. K. Jha, "A comprehensive study of security of internet-of-things," *IEEE Transactions on Emerging Topics in Computing*, 2017.

[9] Motherboard, "How 1.5 Million Connected Cameras Were Hijacked to Make an Unprecedented Botnet," 2016. [Online]. Available: https://www.vice.com/en_us/article/8q8dab/15-million-connected-cameras-ddos-botnet-brian-krebs

[10] N. Woolf, "DDoS attack that disrupted internet was largest of its kind in history, experts say," 2016. [Online]. Available: https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet

[11] R. Sobti and G. Geetha, "Cryptographic Hash Functions: A Review," 2012. [Online]. Available: https://www.researchgate.net/publication/267422045_Cryptographic_Hash_Functions_A_Review

[12] M. Swan, *Blockchain: blueprint for a new economy*, 1st ed. O'Reilly Media, Jan 2015.

[13] D. Puthal, N. Malik, S. P. Mohanty, E. Kougianos, and G. Das, "Everything you wanted to know about the blockchain: Its promise, components, processes, and problems," *IEEE Consumer Electronics Mag.*, vol. 7, no. 4, pp. 6–4, July 2018.

[14] G. Becker, "Merkle Signature Schemes, Merkle Trees and Their Cryptanalysis," 2008. [Online]. Available: https://www.emsec.ruhr-uni-bochum.de/media/crypto/attachments/files/2011/04/becker_1.pdf

[15] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008. [Online]. Available: https://nakamotoinstitute.org/bitcoin/

[16] A. Antonopoulos, *Mastering Bitcoin*, 2nd ed. O'Reilly Media, 2017.

[17] M. A. Ferrag, M. Derdour, M. Mukherjee, A. Derhab, L. Maglaras, H. Janicke, "Blockchain Technologies for the Internet of Things: Research issues and Challenges," 2018. [Online]. Available: https://arxiv.org/pdf/1806.09099.pdf

[18] Ethereum, "A Next-Generation Smart Contract and Decentralized Application Platform," 2019. [Online]. Available: https://github.com/ethereum/wiki/wiki/White-Paper

[19] N. Szabo, "Smart Contracts: Formalizing and securing relationships on public networks," *First Monday*, 1997.

[20] M. Bartoletti and L. Pompianu, "An empirical analysis of smart contract: platforms, applications, and design patterns," 2017. [Online]. Available: https://arxiv.org/pdf/1703.06322.pdf

[21] Ethereum, "Ethash." [Online]. Available: https://github.com/ethereum/wiki/wiki/Ethash

[22] E. Zaghloul, T. Li, Matt Mutka, J. Ren, "Bitcoin and Blockchain: Security and Privacy," 2019. [Online]. Available: https://arxiv.org/pdf/1904.11435.pdf

[23] Ethereum, "Solidity," 2019. [Online]. Available: https://solidity.readthedocs.io/en/v0.5.9/

[24] ——, "Remix," 2019. [Online]. Available: https://remix.ethereum.org/

[25] ——, "Go Ethereum," 2019. [Online]. Available: https://github.com/ethereum/go-ethereum

[26] C. Svensson, "Transactions - web3j 4.1.0 documentation," 2019. [Online]. Available: https://web3j.readthedocs.io/en/latest/

[27] Truffle Blockchain Group, "Truffle Suite," 2019. [Online]. Available: https://github.com/trufflesuite/truffle

[28] Douglas Crockford, "The JavaScript Object Notation (JSON) Data Interchange Format," 2017. [Online]. Available: https://tools.ietf.org/html/rfc8259

[29] Google, "Firebase Realtime Database," 2019. [Online]. Available: https://firebase.google.com/docs/database

[30] J.Pan, L. Ma, R. Ravindran, and P. TalebiFard, "Homecloud: An edge cloud framework and testbed for new application delivery," *23rd International Conference in Telecommunications (ICT). IEEE*, pp. 1–6, 2016.

[31] M. Satyanarayanan, P. Bahl, R. Caceres and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEE Pervasive Comput.*, vol. 8, no. 4, pp. 14–23, 2009.

[32] Oraclize Dev Community, "Oraclize Documentation," 2019. [Online]. Available: https://docs.oraclize.it/

[33] Netscape Communications Corp, Mozilla Fundation, "JavaScript," 1995. [Online]. Available: https://es.wikipedia.org/wiki/JavaScript

[34] RPC, "Remote procedure call." [Online]. Available: https://en.wikipedia.org/wiki/Remote_procedure_call

[35] The Apache Software Foundation, "Welcome to Apache Maven," 2019. [Online]. Available: https://maven.apache.org/

[36] V. Bhushan, "Optimizing smart contracts for cost." [Online]. Available: https://labs.imaginea.com/post/contract-code-optimization

[37] Etherscan, "Ethereum Blockchain Explorer," 2019. [Online]. Available: https://etherscan.io

[38] J. Pan, J. Wang, A. Hester, I. Alqerm, Y. Liu, Y. Zhao, "EdgeChain: An Edge-IoT Framework and Prototype Based on Blockchain and Smart Contracts," 2018. [Online]. Available: https://arxiv.org/pdf/1806.06185.pdf

[39] G. Sagirlar, B. Carminati, E. Ferrari, J. D. Sheehan†, E. Ragnoli, "Hybrid-IoT: Hybrid Blockchain Architecture for Internet of Things - PoW Sub-blockchains," 2018. [Online]. Available: https://arxiv.org/pdf/1804.03903v3.pdf

[40] Wikipedia, "Byzantine fault," 2019. [Online]. Available: https://en.wikipedia.org/wiki/Byzantine_fault

[41] G. Wood, P. Czaban, R. Habermeier, "Polkadot: Decentralized Web 3.0 Blockchain Interoperability Platform." [Online]. Available: https://polkadot.network/

[42] Tendermint Inc., "Internet of Blockchains - Cosmos Network." [Online]. Available: https://cosmos.network/

[43] O. Alphand, M. Amoretti, T. Claeys, S. Dall 'Asta, A. Duda, G. Ferrari, F. Rousseau, B. Tourancheau, L. Veltri, F. Zanichelli, "IoTChain: A Blockchain Security Architecture for the Internet of Things," 2018. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01705455/document

[44] M. Vucinić, B. Tourancheau, F. Rousseau, A. Duda, L. Damon, and R. Guizzetti, *OSCAR: Object Security Architecture for the Internet of Things.* Ad Hoc Networks, 2019, vol. 32.

[45] L. Seitz, G. Selander, E. Wahlstroem, S. Erdtman, and H. Tschofenig, "Authentication and Authorization for Constrained Environments (ACE)," 2017. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-ace-oauth-authz-07

[46] A. Dorri, S. S. Kanhere, R. Jurdak, and P. Gauravaram, "LSB: A Lightweight Scalable BlockChain for IoT Security and Privacy," 2017. [Online]. Available: https://arxiv.org/pdf/1712.02969.pdf

[47] P. K. Sharma, S. Singh, Y-S. Jeong, and J. H. Park, "DistBlockNet: A Distributed Blockchains-Based Secure SDN Architecture for IoT Networks," *IEEE Commun. Mag.*, vol. 55, no. 9, pp. 78–85, 2017.