UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y COMUNICACIONES

Tema:

SISTEMA DE SMART HOME APLICANDO IOT Y BLOCKCHAIN

Trabajo de Titulación Modalidad: Proyecto de Investigación, presentado previo a la obtención del título de Ingeniero en Electrónica y Comunicaciones.

ÁREA: Electrónica

LINEA DE INVESTIGACIÓN: Tecnología de la Información y Sistemas de Control

AUTOR: Oswaldo David Acurio Conteron

TUTOR: Ing. Carlos Diego Gordón Gallegos, Mg. PhD

Ambato – Ecuador

Julio - 2021

APROBACIÓN DEL TUTOR

En calidad de tutor del Trabajo de Titulación con el tema: SISTEMA DE SMART HOME APLICANDO IOT Y BLOCKCHAIN, desarrollado bajo la modalidad de proyecto de investigación por el señor Oswaldo David Acurio Conteron estudiante de la Carrera de Ingeniería en Electrónica y Comunicaciones, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, de la Universidad Técnica de Ambato, me permito indicar que el estudiante ha sido tutorado durante todo el desarrollo del trabajo hasta su conclusión, de acuerdo a lo dispuesto en el Artículo 15 del Reglamento para obtener el Título de Tercer Nivel, de Grado de la Universidad Técnica de Ambato, y el numeral 7.4 del respectivo instructivo.

Ambato, julio 2021

Firmado electrónicamente por:
CARLOS DIEGO GORDON GALLEGOS

Ing. Carlos Gordón, Mg. PhD

AUTORÍA

El presente proyecto de investigación titulado: SISTEMA DE SMART HOME APLICANDO IOT Y BLOCKCHAIN es absolutamente original, auténtico y personal, En tal virtud, el contenido, efectos legales y académicos que se desprenden del mismo son exclusiva responsabilidad del autor.

Ambato, julio 2021

Oswaldo David Acurio Conteron

CC:1804624433

AUTOR

APROBACIÓN DEL TRIBUNAL DE GRADO

En calidad de par calificador del Informe Final del Trabajo de Titulación presentado por el señor Oswaldo David Acurio Conteron, estudiante de la Carrera de Ingeniería en Electrónica y Comunicaciones, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, bajo la Modalidad Proyecto de Investigación, titulado SISTEMA DE SMART HOME APLICANDO IOT Y BLOCKCHAIN , nos permitimos informar que el trabajo ha sido revisado y calificado de acuerdo al Artículo 17 del Reglamento para obtener el Título de Tercer Nivel, de Grado de la Universidad Técnica de Ambato, y al numeral 7.6 del respectivo instructivo. Para cuya constancia suscribimos, conjuntamente con la señora Presidenta del Tribunal.

Ambato, julio 2021

Firmado electrónicamente por:
ELSA PILAR URRUTIA

Ing. Pilar Urrutia, Mg.

PRESIDENTA DEL TRIBUNAL

Firmado electrónicamente por:
VICTOR SANTIAGO MANZANO VILLAFUERTE

Ing. Santiago Manzano, Mg.

PROFESOR CALIFICADOR

Firmado electrónicamente por:
CLARA AUGUSTA SANCHEZ BENITEZ

Ing. Clara Sánchez, Mg.

PROFESORA CALIFICADORA

DERECHOS DE AUTOR

Autorizo a la Universidad Técnica de Ambato, para que haga uso de este Trabajo de Titulación como un documento disponible para la lectura, consulta y procesos de investigación.

Cedo los derechos de mi Trabajo de Titulación en favor de la Universidad Técnica de Ambato, con fines de difusión pública. Además, autorizo su reproducción total o parcial dentro de las regulaciones de la institución

Ambato, julio 2021

Oswaldo David Acurio Conteron

CC:1804624433

AUTOR

AGRADECIMIENTO

A mi familia por su apoyo

en la realización de mi proyecto de titulación, a mis compañeros y amigos por su colaboración en todo momento, y demás familiares que colaboraron de una y otra forma para la culminación del este proyecto.

Oswaldo David Acurio Conteron

Índice General

Índice de Tablas

Índice de Figuras

RESUMEN EJECUTIVO

El presente proyecto diseña e implementa un Sistema de Smart Home aplicando IoT y Blockchain, con el objetivo de mostrar la influencia que tiene la tecnología Blockchain en aplicaciones IoT como Smart Home.

La tecnología Blockchain abarca características como la confiabilidad y la descentralización, ya que garantiza la integridad de los datos en dispositivos con tecnología IoT. Para la interacción del dispositivo IoT y la cadena de bloquea se utilizó la biblioteca Web3.py y el lenguaje de programación Vyper para la creación de contrato inteligente que brinda un mayor nivel de seguridad a diferencia de Solidity, el cual requiere de framework truffle para escribir, compilar y migrar para lo cual se trabajó y Node.js que interacciona con la cadena de bloques Ethereum en Ganache

En el diseño del Sistema de Smart Home aplicando IoT y Blockchain se realizó en Ganache como una cadena de bloques Ethereum para guardar los datos de acceso de los usuarios al momento de ingresar al Hogar después de ser autentificados por el contrato inteligente que compara los datos enviados por el Dispositivo IoT y los datos registrados en el contrato inteligente para desbloquear la puerta.

El sistema Smart Home con IoT y Blockchain se implementó en una Raspberry pi 4 incorporada una cámara de 5MP, un relé conectado a una cerradura de solenoide a 12 V, el conjunto en sí permite reconocimiento facial basado en los algoritmos clasificador en cascada Haar Y Histograma de gradientes orientados los cuales se ejecutan en un script programado en Python y que por medio de la librería Web3.py interactuaran con la cadena de bloques Ethereum instalada en un computador donde se almacenará los datos de acceso al hogar e interactúa con un contrato inteligente que administrar las políticas de acceso. Este Sistema examina aspectos de funcionalidad, tiempos de respuestas, rendimiento adecuados y seguridad.

Descriptores: Blockchain, Contratos inteligentes, Ethereum, Smart home, Python

ABSTRACT

This project designs and implements a Smart Home System applying IoT and Blockchain, with the aim of showing the influence that Blockchain technology has on IoT applications such as Smart Home.

Blockchain technology encompasses features such as reliability and decentralization, as it ensures data integrity on devices with IoT technology. For the interaction of the IoT device and the block chain, the Web3.py library and the Vyper programming language were used for the creation of a smart contract that provides a higher level of security unlike Solidity, which requires a truffle framework to write, compile and migrate for which it was worked and Node.js that interacts with the Ethereum Blockchain in Ganache

In the design of the Smart Home System applying IoT and Blockchain, it was carried out in Ganache as an Ethereum Blockchain to save the access data of the users at the time of entering the Home after being authenticated by the smart contract that compares the data sent by the IoT Device and the data recorded in the smart contract to unlock the door.

The Smart Home system with IoT and Blockchain was implemented in a Raspberry pi 4 built-in a 5MP camera, a relay connected to a solenoid lock at 12V, the set itself allows facial recognition based on the Haar And Histogram cascade classifier algorithms of oriented gradients which are executed in a script programmed in Python and that through the Web3.py library will interact with the Ethereum Blockchain installed on a computer where the home access data will be stored and interact with a smart contract that manage access policies. This System examines aspects of functionality, response times, adequate performance, and security.

Keywords: Blockchain, Smart contracts, Ethereum, Smart home, Python

CAPITULO I

MARCO TEÓRICO

1.1 Antecedentes Investigativos

En el año 2017 Kazim Rifat y Arda Yurdakul publicaron el artículo científico: "Work-in-Progress: Integrating Low-Power IoT devices to a Blockchain-Based Infrastructure". El propósito de este documento es crear una prueba de concepto para permitir que los dispositivos finales de IoT de baja potencia y con recursos limitados accedan a una infraestructura basada en Blockchain. Para lograr este objetivo, se configura una puerta de enlace IoT como un nodo Blockchain y se propone un mecanismo de mensajería basado en eventos para dispositivos finales IoT de baja potencia. Una demostración de dicho sistema se realiza utilizando nodos LoRa y puerta de enlace en una red privada de Ethereum. [1]

Como resultado se puede inferir que los sistemas basados en Blockchain se usan ampliamente, el desarrollo de aplicaciones y el procesamiento de datos se pueden llevar a cabo de manera masiva mediante el uso de contratos inteligentes como se demuestra con nuestra prueba de concepto 'Blocky'. Actualmente se trabaja para demostrar que, independientemente de sus capacidades informáticas y de almacenamiento, es posible integrar un cliente Blockchain a cualquier dispositivo IoT [1]

En el año 2018, Md. Abdur Rahman, Khalid Abualsaud, Stuart Barnes, Mamunur rashid y Syed Maruf Abdullah publican el articulo cientifico denominado: "A Natural User Interface and Blockchain-Based In-Home Smart Health Monitoring System" fue publicado en: Fondren Library Rice University. El documento tiene como objetivo presentar un sistema de interacción con dispositivos inteligentes de salud de IoT para el hogar con personas mayores o con necesidades especiales. [2]

El sistema consta de dos dispositivos IoT una cerradura inteligente y una bombilla inteligente que podría controlarse utilizando dos dispositivos IoT basados en gestos, Leap Motion y el brazalete Myo. El intermediario MQTT, el servidor API y los nodos Hyperledger Fabric Blockchain se ejecutan en una máquina virtual AWS, ejecutando en Ubuntu. Se utilizó Infura, una infraestructura de Blockchain para un acceso seguro, confiable y escalable a las API transaccionales de Ethereum y las puertas de enlace IPFS. [2]

Como resultado se presenta un prototipo funcional de interacción segura basada en gestos con dispositivos IoT médicos domésticos inteligentes. La comunicación, la identificación de los usuarios y los dispositivos de IoT, y los datos de interacción y de IoT sin procesar se almacenan de forma segura fuera de la cadena, mientras que las transacciones más importantes se almacenan en la cadena de bloques. [2]

En el año 2018, Dinan Fakhr y Kusprasapta Mutijarsa publican el artículo "Secure IoT Communication using Blockchain Technology". El documento presenta como objetivo crear un sistema IoT sin Blockchain y otro usando Blockchain que luego se

comparará entre los dos. El protocolo de comunicación utilizado en el primer sistema IoT es MQTT. Mientras tanto en el segundo sistema, la plataforma Blockchain utilizada es Ethereum junto con un contrato inteligente. Ambos sistemas de IoT se analizarán para determinar su nivel de seguridad simulando ataques y observando sus aspectos de seguridad. [3]

En el primer sistema IoT está conformado por un refrigerador inteligente enviará datos a través del agente MQTT a la televisión inteligente. Los datos se cifran con el estándar de cifrado avanzado. Además, se utiliza una función hash para proteger la clave de cifrado. [3]

El segundo sistema IoT se conforma de una red Blockchain y un contrato inteligente. Este contrato inteligente funciona como intermediario para 2 dispositivos IoT que están conectados y utilizados para almacenar y recuperar datos contenidos en la red Blockchain. El refrigerador inteligente almacenará datos en la red Blockchain mediante un contrato inteligente, mientras que la televisión inteligente utilizará el contrato inteligente para obtener datos que se hayan almacenado en la red Blockchain. [3]

Basado en varias pruebas que se han llevado a cabo, obtuvieron que el sistema IoT que usa la tecnología Blockchain puede resolver los problemas de seguridad que surgen en la comunicación entre dispositivos IoT porque tiene un mayor nivel de seguridad que el sistema IoT sin tecnología Blockchain para que los datos La integridad puede ser garantizada. Esto se observó en las pruebas de simulaciones de ataque y observaciones de efectos de avalancha llevadas a cabo donde el sistema IoT que usa la tecnología Blockchain tiene una mejor seguridad. [3]

En el año 2019, Ulfah Nadiya, Muhammad Ilham Rizqyawan y Oka Mahnedra publican el articulo “Blockchain-based Secure Data Storage for Door Lock System” fue publicado: “4th International Conference on information Technology, Information Systems and Electrical Engeneering”. El sistema propuesto fue simulado utilizando dos PC, una PC como nodo de cámara web y otra como nodo de propietario que controlará los datos. También hay un contrato inteligente que funciona como intermediario para los dispositivos IoT que están conectados y da políticas a los dispositivos que pueden almacenar o recuperar los datos en la red Blockchain. La cámara web almacenará datos en la red Blockchain mediante un contrato inteligente, mientras que el propietario usará el contrato inteligente para obtener datos almacenados en la red Blockchain. [4]

Este estudio utiliza la plataforma de cadena de bloques Ethereum y la máquina virtual Ethereum utilizada es Go Ethereum con el lenguaje de programación Golang. El marco de contrato inteligente es truffle con Solidity como lenguaje de programación. La criptografía en Ethereum usa el algoritmo de cifrado del algoritmo de firma digital de curva elíptica (ECDSA) y el proceso de transacción en Ethereum usa Keccak-256 como una función hash. [4]

El resultado de esta investigación fue que la función hash y el algoritmo de cifrado de la cadena de bloques Ethereum pueden fortalecer el sistema IoT, que ha demostrado por el nivel de seguridad que calcula utilizando el efecto de avalancha. El índice de efecto de avalancha de la función hash y el algoritmo de cifrado de la cadena de bloques Ethereum es de aproximadamente 96% en promedio, indica que el sistema es seguro. [4]

En el año 2019, Kerem Ataşen y Hakan Üstünel publican el articulo: "Designing a Secure IoT Network by Using Blockchain", en el "3rd International Symposium on Multidisciplinary Studies and Innovative Technologies ". En el documento se describe la creación de una cadena de bloques privada de Ethereum combinando dispositivos Raspberry Pi 3 y 2 nodos completos Ethereum diferentes que se ubican en la misma computadora. En esta cadena de bloques establecida de Ethereum, se puede ejecutar una aplicación descentralizada. Esta DApp puede asociarse con contratos inteligentes según el objetivo. Las tecnologías de almacenamiento distribuido, como IPFS o Storj, se pueden usar como soluciones al problema de escalabilidad de Blockchain. [5]

Como resultado esta red privada de Ethereum, puede ser implementada especialmente en áreas que requieren bajo costo, bajo consumo de energía y donde los datos no son muy densos, como la protección de datos digitales, seguridad de infraestructura, propiedad de derechos de autor, seguridad de ciudad inteligente, etc. [5]

En el año 2019, Ahsan Manzoor, Madhsanka Liyanage, An Braeken, Salil S. Kanhere y Mika Ylianttila publican el artículo: "Blockchain based Proxy Re-Encryption Scheme for Secure IoT Data Sharing" en el "2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)". Este documento presenta un esquema de re encriptación de proxy basado en Blockchain. El sistema almacena los datos de IoT en una nube distribuida después del cifrado. Para compartir los datos recopilados de IoT, el sistema establece contratos inteligentes dinámicos de tiempo de ejecución entre el sensor y el usuario de datos sin la participación de un tercero de confianza. [6]

El sistema tiene tres sensores IoT, tres computadoras mineras, cinco nodos completos Ethereum, dos usuarios regulares y un servidor de almacenamiento en la nube. Configuramos y conectamos todos los dispositivos a internet. Utilizamos el protocolo de autodescubrimiento de Geth para conectar los mineros y los nodos completos, y configuramos la nube de Google rebase para el almacenamiento. [6]

En este documento se propuso una plataforma de negociación basada en Blockchain con la combinación de un esquema de encriptación de proxy libre de emparejamiento para garantizar la transferencia segura de los datos del sensor al usuario. También validamos el modelo de prueba de concepto en un banco de pruebas privado de

Ethereum y demostramos la practicidad del diseño del sistema usando computadoras portátiles y Raspberry Pi. [6]

1.2 Contextualización del problema

La tecnología IoT ha crecido en los últimos años. Un sistema Smart Home utiliza esta tecnología para brindar comodidad a sus propietarios. Sin embargo , IoT tiene desafíos relacionados con la seguridad de los datos , además de la propiedades de una red distribuida y a gran escala. [4].

Según los datos de Juniper Research, la cantidad de dispositivos de IoT que se conectarán en el mundo aumentarán en un 140% de 21 millones en 2018 a 50 mil millones en 2023. Se puede decir que IoT es un avance importante en el campo de la tecnología de la información. [3]

El rápido desarrollo de IoT condujo a la aparición de varios problemas, uno de los cuales fue la vulnerabilidad a los ataques cibernéticos. Hacer que los dispositivos IoT permanezcan seguros es algo difícil, porque se violan muchas de las políticas de seguridad para que los dispositivos IoT sean económicos, pequeños y fáciles de usar. [3] El ciberdelito explota los dispositivos y redes de IoT que no muestra signos de mitigación. Por lo tanto, la seguridad, la privacidad y la verificación de identidad siguen siendo preocupaciones fundamentales en las implementaciones de sistemas IoT. [7]

La tecnología IoT puede admitir diversas aplicaciones y servicios en varios dominios. Uno de los servicios que puede soportar un sistema IoT es el sistema Smart Home. El uso de IoT en el sistema de Smart Home puede ayudar a los propietarios a monitorear y controlar su hogar las 24 horas. Los dispositivos IoT generan, procesan e intercambian grandes cantidades de datos de seguridad, así como información confidencial y privada, por consecuencias es un objetivo atractivo de varios ataques cibernéticos. Si se piratea una gran cantidad de información, puede tener un gran impacto en toda la comunidad, por lo tanto , la seguridad de los datos en los sistemas domésticos inteligentes debe tomarse en cuenta en su implementación. [4]

En una Smart Home el dato de acceso a la cerradura de la puerta de ingreso al hogar debe mantenerse de forma segura. Un sistema de bloqueo de puerta es una parte importante del sistema Smart Home porque está directamente relacionado con la seguridad e identificación de los propietarios. Estos datos no deben ser vulnerables a la falsificación y la piratería para garantizar la integridad de los datos, es creando un sistema de almacenamiento basado en Blockchain, además, también puede fortalecer la seguridad de IoT. [4]

Blockchain tiene propiedades inmutables y de no repudio, los datos almacenados en la cadena de bloques son difíciles de cambiar. Un sistema basado en Blockchain puede garantizar que los datos de la tienda se obtengan en un dispositivo autorizado que ya está registrado en el sistema al verificar la clave pública del dispositivo, ya que puede garantizar la integridad de los datos. [4]

El proyecto surge ante el crecimiento acelerado de la tecnología IoT en los hogares los cuales se benefician de sus altas prestaciones, pero al manejar datos importantes la seguridad es un factor que se ha dejado a un lado, lo que ha permitido que terceras personas en este caso cibercriminales hagan uso de esta información.

Este estudio desarrolla una solución que se puede utilizar para superar los desafíos relacionados con la seguridad de los datos en sistemas Smart Home. Una puerta inteligente con desbloqueo facial aplicando técnicas de machine learning que permite que permite detectar y reconocer a la persona que intenta ingrese al hogar y se autentifica por medio de un contrato Inteligente que además almacena los datos obtenidos de la persona por medio de la creación cadena de bloques en la Red Blockchain garantizando la seguridad del dispositivo IoT en una Smart Home.

El sistema propuesto utiliza la tecnología de cadena de bloques Ethereum para almacenar datos de acceso a la cerradura de la puerta y un contrato inteligente escrito en lenguaje de programación Vyper para administrar las políticas. El dispositivo de reconocimiento facial está registrado en la red Blockchain y puede enviar los datos para ser almacenado en la cadena de bloques y recibir los datos de autentificación de los usuarios registrados en el contrato inteligente para el ingreso del hogar, por lo tanto, se garantiza la integridad de los datos. [4]

1.3 Fundamentación teórica

1.3.1 Blockchain

Definición

Definición de Layman: Blockchain es un sistema de mantenimiento de registros compartido, seguro y en constante crecimiento en el que cada usuario de los datos tiene una copia de los registros, que solo se puede actualizar si todas las partes involucradas en una transacción acuerdan actualizar. [8]

Definición técnica: Blockchain es un libro mayor distribuido de igual a igual que es criptográficamente seguro, solo para anexos, inmutable (extremadamente difícil de cambiar) y que se puede actualizar solo por consenso o acuerdo entre pares. [8]

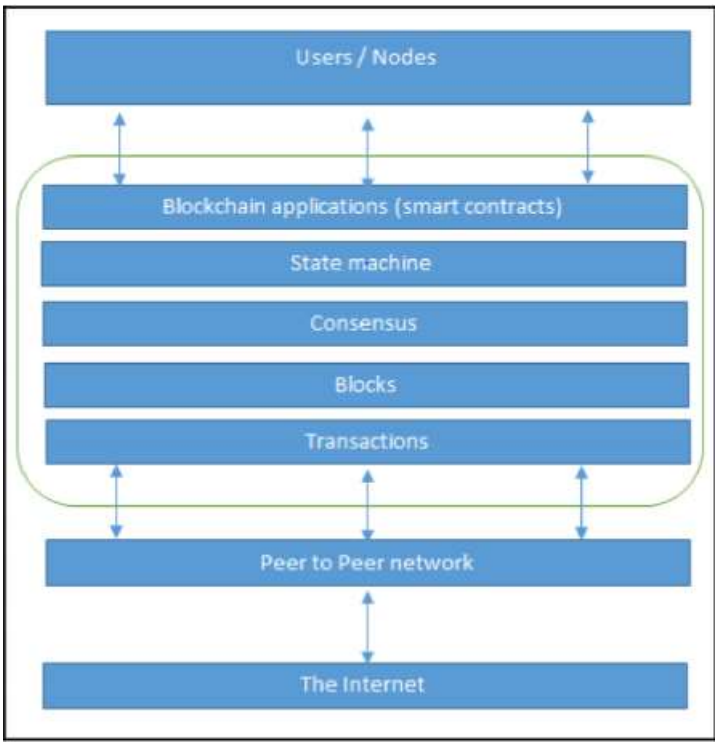


Figura 1. Estructura genérica de un bloque [8]

Elementos genéricos de una cadena de bloques

La estructura de una cadena de bloques genérica se puede visualizar con la ayuda del siguiente diagrama:

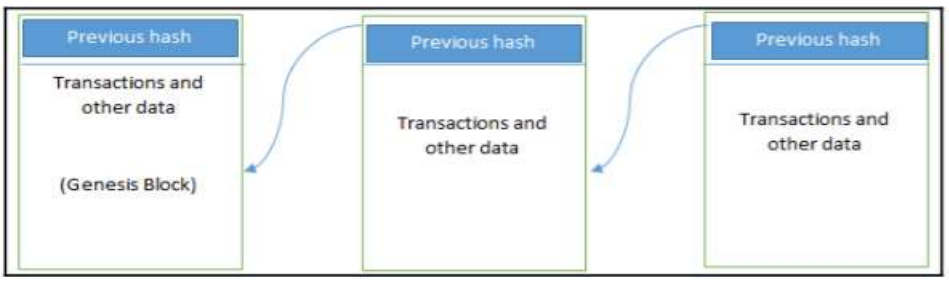


Figura 2. Estructura generico de Blockchain [8]

Dirección: Las direcciones son identificadores únicos que se utilizan en una transacción de cadena de bloques para indicar remitentes y destinatarios. Una dirección suele ser una clave pública o se deriva de una clave pública y las direcciones en sí mismas son únicas. [8]

Transacción: una transacción es la unidad fundamental de una cadena de bloques. Una transacción representa una transferencia de valor de una dirección a otra. [8]

Bloque: un bloque se compone de varias transacciones y otros elementos, como el hash del bloque anterior (puntero hash), la marca de tiempo y el nonce. [8]

Red peer-to-peer: como su nombre lo indica, una red peer-to-peer es una topología de red en la que todos los pares pueden comunicarse entre sí y enviar y recibir mensajes. [8]

Lenguaje de programación: Los scripts o programas realizan varias operaciones en una transacción para facilitar diversas funciones. [8]

Máquina virtual: Esta es una extensión del script de transacción presentado anteriormente. Una máquina virtual permite que el código completo de Turing se ejecute en una cadena de bloques; mientras que un script de transacción tiene un funcionamiento limitado. [8]

Máquina de estado: una cadena de bloques puede verse como un mecanismo de transición de estado mediante el cual un estado se modifica de su forma inicial a la siguiente y, finalmente, a una forma final por los nodos en la red de cadena de bloques como resultado de la ejecución, validación y finalización de una transacción proceso. [8]

Nodo: Un nodo en una red Blockchain realiza varias funciones según el rol que asume. Un nodo puede proponer y validar transacciones y realizar minería para facilitar el consenso y asegurar la cadena de bloques. Este objetivo se logra siguiendo un protocolo de consenso. Los nodos también pueden realizar otras funciones, como verificación de pago simple, validación y muchas otras funciones, según el tipo de cadena de bloques utilizada y el rol asignado al nodo. Los nodos también realizan una función de firma de transacciones. Las transacciones son creadas primero por nodos y luego también firmadas digitalmente por nodos que utilizan claves privadas como prueba de que son el propietario legítimo del activo que desean transferir a otra persona en la red Blockchain. [8]

Contrato inteligente: estos programas se ejecutan sobre la cadena de bloques y encapsulan la lógica empresarial que se ejecutará cuando se cumplan determinadas condiciones. Estos programas se pueden hacer cumplir y se pueden ejecutar automáticamente. Los contratos inteligentes tienen muchos casos de uso, que incluyen, entre otros, la gestión de identidades, los mercados de capitales, la financiación comercial, la gestión de registros, los seguros y el gobierno electrónico. [8]

Funcionamiento de Blockchain

Los nodos son mineros que crean nuevos bloques y acuñan Criptomonedas o bloquean firmantes que validan y firman digitalmente las transacciones. Una decisión crítica que debe tomar toda red Blockchain es averiguar qué nodo agrega el siguiente bloque a la cadena de bloques. Esta decisión se toma mediante un mecanismo de consenso. [8]

Acumulación de bloques

Este esquema se presenta aquí para darle una idea general de cómo se generan los bloques y cuál es la relación entre transacciones y bloques:

1. Un nodo inicia una transacción creando primero y luego firmándola digitalmente con su clave privada. Una transacción puede representar varias acciones en una cadena de bloques. Por lo general, esta es una estructura de datos que representa la transferencia de valor entre usuarios en la red Blockchain. La estructura de datos de la transacción generalmente consiste en alguna lógica de transferencia de valor, reglas relevantes, direcciones de origen y destino y otra información de validación. [8]
2. Una transacción se propaga mediante un protocolo de inundación, llamado protocolo Gossip, a los pares que validan la transacción según criterios preestablecidos. Por lo general, se requiere más de un nodo para verificar la transacción. [8]
3. Una vez validada la transacción, se incluye en un bloque, que luego se propaga a la red. En este punto, la transacción se considera confirmada.
4. El bloque recién creado ahora pasa a formar parte del libro mayor, y el siguiente bloque se vincula criptográficamente con este bloque. Este enlace es un puntero hash. En esta etapa, la transacción obtiene su segunda confirmación y el bloque obtiene su primera confirmación. [8]
5. Las transacciones se vuelven a confirmar cada vez que se crea un nuevo bloque. Por lo general, se requieren seis confirmaciones en la red Bitcoin para considerar la transacción como definitiva. [8]

Las características de una cadena de bloques:

Consenso de distribución: el consenso distribuido es el sustento principal de una cadena de bloques. Este mecanismo permite que una cadena de bloques presente una versión única de la verdad, que es acordada por todas las partes sin el requisito de una autoridad central. [8]

Verificación de transacciones: todas las transacciones publicadas desde los nodos de la cadena de bloques se verifican en función de un conjunto predeterminado de reglas. Solo se seleccionan transacciones válidas para su inclusión en un bloque.

Plataforma para contratos inteligentes: una cadena de bloques es una plataforma en la que se pueden ejecutar programas para ejecutar la lógica empresarial en nombre de los usuarios. [8]

Transferencia de valor entre pares: Blockchain permite la transferencia de valor entre sus usuarios a través de token. Los tokens pueden considerarse portadores de valor. [8]

Generación de Criptomonedas: esta característica es opcional según el tipo de Blockchain en uso. Una cadena de bloques puede crear Criptomonedas como incentivo para sus mineros que validan las transacciones y gastan recursos para asegurar la cadena de bloques. [8]

Propiedad inteligente: ahora es posible vincular un activo digital o físico a la cadena de bloques de una manera tan segura y precisa que nadie más puede reclamarlo. Usted tiene el control total de su activo y no puede gastarse ni poseerse dos veces. Compare esto con un archivo de música digital, por ejemplo, que se puede copiar muchas veces sin ningún control. [8]

Proveedor de seguridad: la cadena de bloques se basa en tecnología criptográfica probada que garantiza la integridad y disponibilidad de los datos. Generalmente, no se brinda confidencialidad debido a los requisitos de transparencia. Esta limitación es la principal barrera para su adopción por parte de instituciones financieras y otras industrias que requieren privacidad y confidencialidad de las transacciones. [8]

Inmutabilidad: una vez que los registros se agregan a Blockchain, son inmutables. Existe la posibilidad remota de revertir los cambios, pero esto debe evitarse a toda costa ya que hacerlo consumiría una cantidad exorbitante de recursos informáticos. [8]

Singularidad: esta característica de Blockchain asegura que cada transacción sea única y no se haya gastado ya.Esta característica es especialmente relevante con las Criptomonedas, donde la detección y evitar el doble gasto es un requisito vital. [8]

Consenso

Consenso es la columna vertebral de una cadena de bloques y, como resultado, proporciona descentralización del control a través de un proceso opcional conocido como minería. La elección del algoritmo de consenso también se rige por el tipo de Blockchain en uso; es decir, no todos los mecanismos de consenso son adecuados para todos los tipos de cadenas de bloques. [8]

El consenso es un proceso de acuerdo entre los nodos que desconfían del estado final de los datos. Para lograr el consenso, se utilizan diferentes algoritmos. Es fácil llegar a un acuerdo entre dos nodos, pero cuando varios nodos están participando en un sistema distribuido y necesitan acordar un valor único, lograr el consenso se convierte en todo un desafío. Este proceso de lograr un estado o valor común de acuerdo entre múltiples nodos a pesar de la falla de algunos nodos se conoce como consenso distribuido. [8]

Mecanismo de Consenso

Un consenso es un conjunto de pasos que toman la mayoría o todos los nodos de una cadena de bloques para acordar un estado o valor propuesto. Durante más de tres

décadas, este concepto ha sido investigado por informáticos en la industria y el mundo académico. Los mecanismos de consenso se han convertido recientemente en el centro de atención y han ganado una popularidad considerable con la llegada de Blockchain y Bitcoin. [8]

Hay varios requisitos que deben cumplirse para proporcionar los resultados deseados en un mecanismo de consenso. A continuación, se describen estos requisitos:

- **Acuerdo:** Todos los nodos honestos deciden sobre el mismo valor
- **Terminación:** Todos los nodos honestos terminan la ejecución del proceso de consenso y finalmente llegan a una decisión
- **Validez:** El valor acordado por todos los nodos honestos debe ser el mismo que el valor inicial propuesto por al menos un nodo honesto
- **Tolerante a fallas**: el algoritmo de consenso debe poder ejecutarse en presencia de nodos defectuosos o maliciosos
- **Integridad:** Este es un requisito de que ningún nodo puede tomar la decisión más de una vez en un solo consenso ciclo

Tipos de mecanismos de consenso

Todos los consensos se desarrollan para hacer frente a las fallas en un sistema distribuido y para permitir que los sistemas distribuidos alcancen un estado final de acuerdo. Hay dos categorías generales de mecanismos de consenso. Estas categorías tratan con todo tipo de fallas. Estos tipos comunes de mecanismos de consenso son los siguientes: [8]

- **Basado en la tolerancia a fallas bizantina tradicional (BFT):** sin operaciones de cálculo intensivo, como la inversión parcial de hash, este método se basa en un esquema simple de nodos que son mensajes firmados por el editor. Finalmente, cuando se recibe una cierta cantidad de mensajes, se llega a un acuerdo. [8]
- **Mecanismos de consenso basados en elecciones de líderes:** este arreglo requiere que los nodos compitan en una lotería de elecciones de líderes, y el nodo que gana propone un valor final. [8]

Consenso en Blockchain

Consenso es un concepto de computación distribuida que se ha utilizado en Blockchain para proporcionar un medio para aceptar una única versión de la verdad por parte de todos los pares en la red Blockchain.

- **Proof-based:** basado en lotería de elección de líder o el consenso de Nakamoto mediante el cual un líder es elegido al azar y propone un valor final. Esta categoría también se conoce como el totalmente descentralizado o sin permiso tipo de mecanismo de consenso. [8]
- **BFT-based:** es un enfoque más tradicional basado en rondas de votaciones. Esta clase de consenso también se conoce como consorcio o autorizado tipo de mecanismo de consenso. [8]
- **Prueba de trabajo (PoW**): este tipo de mecanismo de consenso se basa en la prueba de que se han gastado recursos computacionales adecuados antes de proponer un valor para la aceptación de la red. [8]

- **Prueba de participación (PoS):** este algoritmo funciona con la idea de que un nodo o usuario tiene una participación adecuada en el sistema; es decir, el usuario ha invertido lo suficiente en el sistema para que cualquier intento malintencionado de ese usuario supere los beneficios de realizar dicho ataque en la red. [8]
- **Prueba de participación delegada (DPoS):** esta es una innovación sobre la PoS estándar, mediante la cual cada nodo que tiene una participación en el sistema puede delegar la validación de una transacción a otros nodos mediante votación. Se utiliza en la cadena de bloques BitShares. [8]
- **Prueba de tiempo transcurrido (PoET):**utiliza un entorno de ejecución confiable para proporcionar aleatoriedad y seguridad en el proceso de elección de líderes a través de un tiempo de espera garantizado. [8]
- **Prueba de depósito (PoD):** en este caso, los nodos que deseen participar en la red deben realizar un depósito de seguridad antes de poder extraer y proponer bloques. [8]
- **Prueba de importancia (PoI):** PoI no solo se basa en la cantidad de participación que tiene un usuario en el sistema, sino que también monitorea el uso y movimiento de tokens por parte del usuario para establecer un nivel de confianza e importancia. [8]
- **Consenso federado o consenso bizantino federado:** este mecanismo se utiliza en el protocolo de consenso estelar. Los nodos de este protocolo retienen un grupo de pares de confianza pública y propagan solo aquellas transacciones que han sido validadas por la mayoría de los nodos de confianza. [8]
- **Mecanismos basados en la reputación:** como sugiere el nombre, un líder es elegido por la reputación que ha construido a lo largo del tiempo en la red. Se basa en los votos de otros miembros. [8]
- **PBFT:** este mecanismo logra la replicación de la máquina de estado, lo que proporciona tolerancia contra los nodos bizantinos. [8]
- **Prueba de actividad (PoA):** este esquema es una combinación de PoS y PoW, que asegura que un interesado sea seleccionado de manera pseudoaleatoria, pero uniforme. [8]
- **Prueba de capacidad (PoC):** este esquema utiliza el espacio del disco duro como recurso para extraer los bloques, donde se utilizan los recursos de la CPU.
- **Prueba de almacenamiento (PoS):** este esquema permite la subcontratación de la capacidad de almacenamiento. Este esquema se basa en el concepto de que un determinado dato probablemente sea almacenado por un nodo *que* sirve como medio para participar en el mecanismo de consenso. [8]

1.3.2 Criptografía simétrica

La criptografía es la ciencia que garantiza la seguridad de la información en presencia de adversarios. Lo hace bajo el supuesto de que los adversarios disponen de recursos ilimitados. Los cifrados son algoritmos que se utilizan para cifrar o descifrar datos, de modo que, si los intercepta un adversario, los datos no tienen sentido para ellos sin el descifrado, que requiere una clave secreta. [8]

La criptografía se utiliza principalmente para brindar un servicio de confidencialidad. Por sí solo, no puede considerarse una solución completa, sino que

sirve como un componente fundamental dentro de un sistema de seguridad más amplio para abordar un problema de seguridad. [8]

Además de un servicio de confidencialidad, la criptografía también proporciona otros servicios de seguridad como integridad, autenticación y no repudio. Además, también se proporciona responsabilidad, que es un requisito en muchos sistemas de seguridad. [8]

Criptografía

En el siguiente diagrama se muestra un modelo de criptografía genérico:

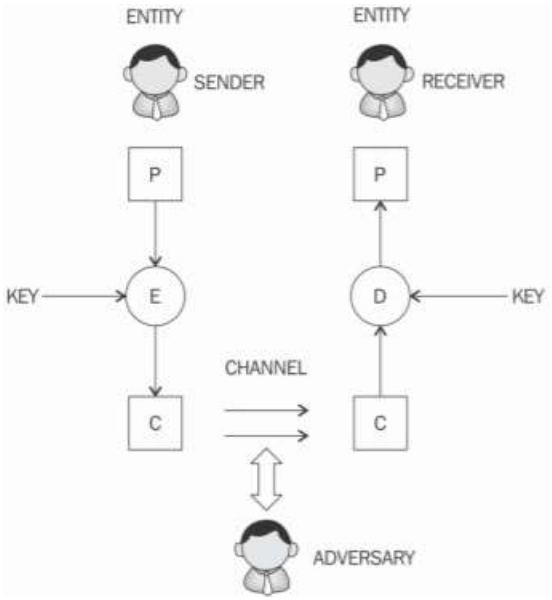


Figura 3. Un modelo del modelo genérico de cifrado y descifrado[8]

En la figura 3, P, E, C y D representan texto plano, cifrado, texto cifrado y descifrado, respectivamente. También con base en este modelo, las explicaciones de conceptos como entidad, remitente, receptor, adversario, clave y canal siguen: [8]

- **Entidad:** ya sea una persona o sistema que envía, recibe o realiza operaciones con datos
- **Remitente:** es una entidad que transmite el dato
- **Receptor:** esta es una entidad que recibe los datos
- **Adversario:** esta es una entidad que intenta eludir el servicio de seguridad
- **Clave:** una clave son datos que se utilizan para cifrar o descifrar otros datos
- **Canal:** el canal proporciona un medio de comunicación entre entidades

Los servicios de criptografía mencionados:

Confidencialidad:

La confidencialidad es la garantía de que la información solo está disponible para entidades autorizadas.

Integridad:

Los datos no pueden ser falsificados o modificados por un adversario intencionalmente o por errores accidentales o no intencionales. Aunque la integridad de los datos no puede evitar la alteración de los datos, puede proporcionar un medio para detectar si se modificaron. [9]

Autenticación:

La autenticación proporciona seguridad sobre la identidad de una entidad o la validez de un mensaje.

Hay dos tipos de mecanismos de autenticación:

Autenticación de la entidad

La autenticación de la entidad es la garantía de que una entidad está actualmente involucrada y activa en una sesión de comunicación. [8]

factores de autenticación:

- El primer factor es algo que tenga, como un token de hardware o una tarjeta inteligente. En este caso, un usuario puede usar un token de hardware además de las credenciales de inicio de sesión para obtener acceso a un sistema. Este mecanismo protege al usuario al requerir dos factores de autenticación. Un usuario que tenga acceso al token de hardware y conozca las credenciales de inicio de sesión podrá acceder al sistema. Ambos factores deben estar disponibles para acceder al sistema, lo que convierte a este método en un mecanismo de autenticación de dos factores. En caso de que se pierda el token de hardware, por sí solo no será de ninguna utilidad a menos que, algo que sepa, la contraseña de inicio de sesión también se utilice junto con el token de hardware. [8]
- El segundo factor es algo que eres, que utiliza características biométricas para identificar al usuario. Con este método, se utiliza la huella digital, la retina, el iris o la geometría de la mano de un usuario para proporcionar un factor adicional de autenticación. De esta manera, se puede garantizar que el usuario estuvo presente durante el proceso de autenticación, ya que las características biométricas son únicas para cada individuo. Sin embargo, requiere una implementación para garantizar un alto nivel de seguridad, ya que algunas investigaciones han sugerido que los sistemas biométricos pueden eludirse en condiciones específicas. [8]

No repudio

El no repudio es la garantía de que una entidad no puede negar un compromiso o acción anterior proporcionando evidencia incontrovertible. Es un servicio de seguridad que ofrece prueba definitiva de que se ha producido una determinada actividad. Esta propiedad es fundamental en situaciones discutibles en las que una entidad ha negado las acciones realizadas. Este servicio produce evidencia criptográfica en transacciones electrónicas para que en caso de disputas se pueda utilizar como confirmación de una acción. [8]

Responsabilidad

Responsabilidad de cuentas es la garantía que establece que las acciones que afectan la seguridad se pueden rastrear hasta la parte responsable. Esto generalmente se proporciona mediante mecanismos de registro y auditoría en sistemas donde se requiere una auditoría detallada debido a la naturaleza del negocio. [8]

1.3.2.1 Primitivas criptográficas

Las primitivas criptográficas son los bloques de construcción básicos de un protocolo o sistema de seguridad. Un protocolo de seguridad es un conjunto de pasos que se toman para lograr los objetivos de seguridad requeridos mediante la utilización de mecanismos de seguridad adecuados. Se utilizan varios tipos de protocolos de seguridad, como protocolos de autenticación, protocolos de no repudio y protocolos de gestión de claves. [8]

La taxonomía de las primitivas criptográficas se puede visualizar como se muestra aquí:

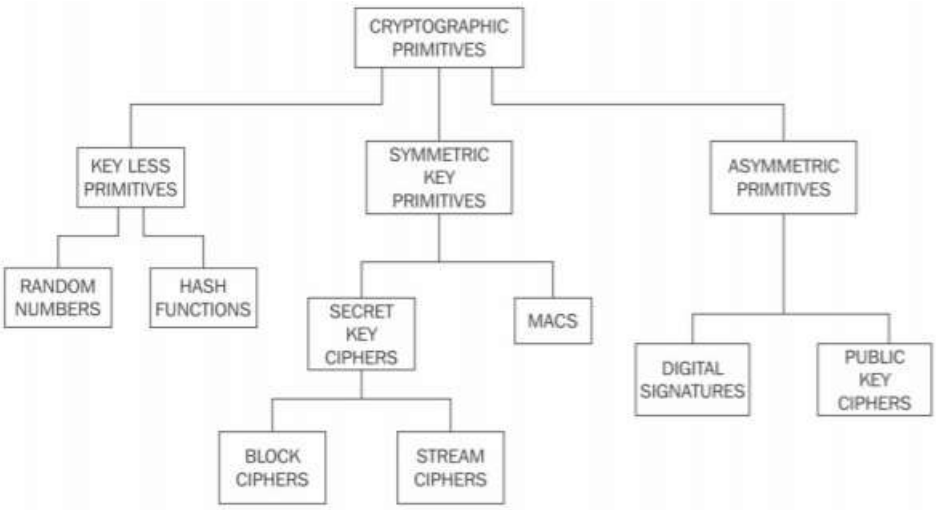


Figura 4. Primitivas criptográficas [8]

Como se muestra en el figura 4 la taxonomía de primitivas criptográficas, la criptografía se divide principalmente en dos categorías: criptografía simétrica y criptografía asimétrica. [8]

Criptografía simétrica

La criptografía simétrica se refiere a un tipo de criptografía donde la clave que se utiliza para cifrar los datos es la misma que se utiliza para descifrar los datos. Por lo tanto, también se conoce como criptografía de clave compartida. La clave debe establecerse o acordarse antes de que se produzca el intercambio de datos entre las partes comunicantes. Esta es la razón por la que también se denomina criptografía de clave secreta. [8]

Hay dos tipos de cifrados simétricos:

- cifrados de flujo
- cifrados de bloque.

El estándar de cifrado de datos (DES) y el estándar de cifrado avanzado (AES) son ejemplos típicos de cifrado de bloques, mientras que RC4 y A5 son cifrados de flujo de uso común. [8]

Cifrados de flujo

Los cifrados de flujo son algoritmos de cifrado que aplican algoritmos de cifrado bit a bit al texto sin formato mediante un flujo de claves. Hay dos tipos de cifrados de flujo: cifrados de flujo síncronos y cifrados de flujo asíncronos: [8]

- Los cifrados de flujo síncronos son aquellos en los que el flujo de claves depende solo de la clave [8]
- Los cifrados de flujo asíncronos tienen un flujo de claves que también depende de los datos cifrados [8]

En cifrados de flujo, cifrado el descifrado son la misma función porque son simples adiciones módulo 2 u operaciones XOR. El requisito fundamental en los cifrados de flujo es la seguridad y la aleatoriedad de los flujos de claves. Se han desarrollado varias técnicas que van desde generadores de números pseudoaleatorios hasta verdaderos generadores de números aleatorios implementados en hardware para generar números aleatorios, y es vital que todos los generadores de claves sean criptográficamente seguros: [8]

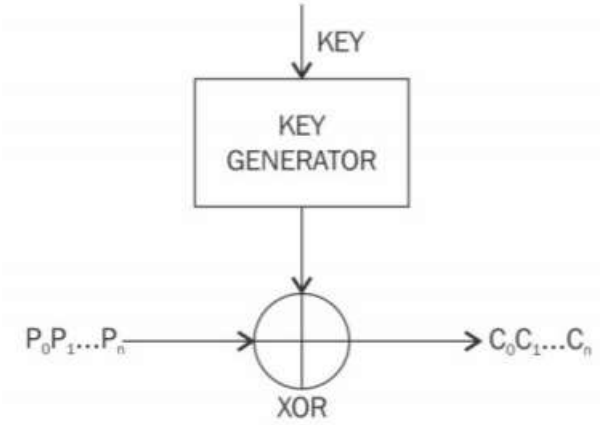


Figura 5. Operation of a stream cipher

Cifrados en bloque

Los cifrados en bloque son algoritmos de cifrado que dividen el texto a cifrar en bloques de una longitud fija y aplican el cifrado bloque por bloque. [8]

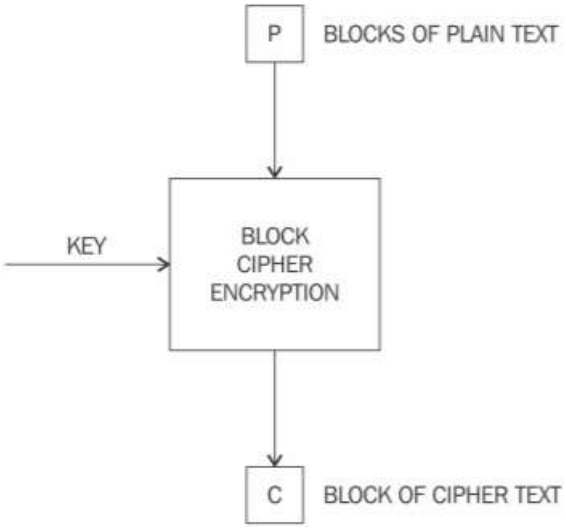


Figura 6. Operación simplificada de un cifrado en bloque [8]

Modo de cifrado de bloques

En modo de cifrado de bloques, el texto sin formato se divide en bloques de longitud fija según el tipo de cifrado utilizado. Luego, la función de cifrado se aplica a cada bloque.

Libro de códigos electrónico

Libro de códigos electrónicos es un modo básico de funcionamiento en el que los datos cifrados se producen como resultado de la aplicación del algoritmo de cifrado uno por uno a cada bloque de texto sin formato. Este es el modo más sencillo, pero no debe usarse en la práctica ya que es inseguro y puede revelar información: [8]

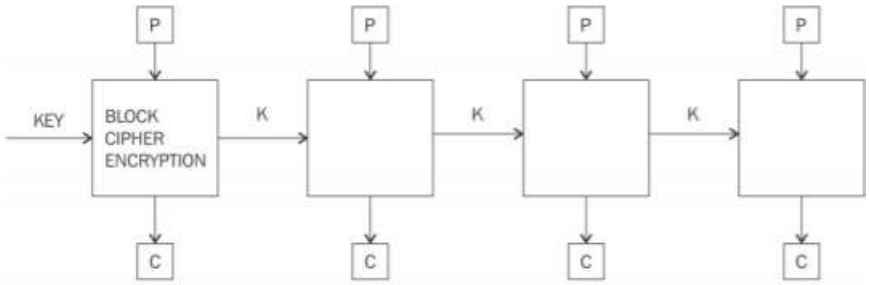


Figura 7. Modo Libro de Códigos Electrónicos para cifrados en bloque[8]

Encadenamiento de bloques de cifrado

En el modo Encadenamiento de bloques de cifrado, cada bloque de texto plano se XOR con el bloque previamente cifrado. El modo CBC utiliza el vector de inicialización para cifrar el primer bloque. Se recomienda que el IV se elija al azar:

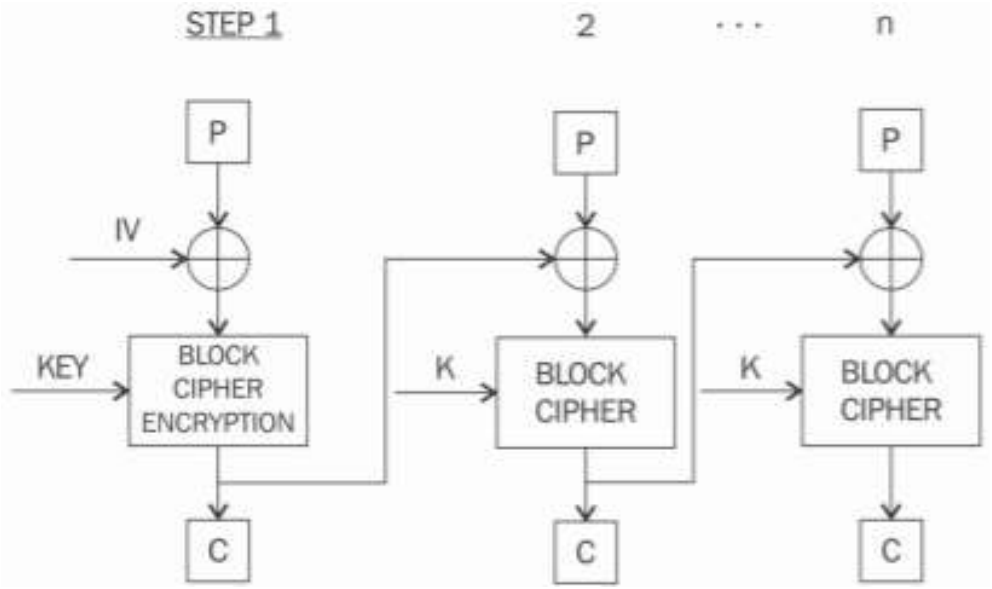


Figura 8. Modo de encadenamiento de bloques de cifrado[8]

Contador modo

El contador utiliza eficazmente un cifrado de bloques como cifrado de flujo. En este caso, se proporciona un nonce único que se concatena con el valor del contador para producir un flujo de claves: [8]

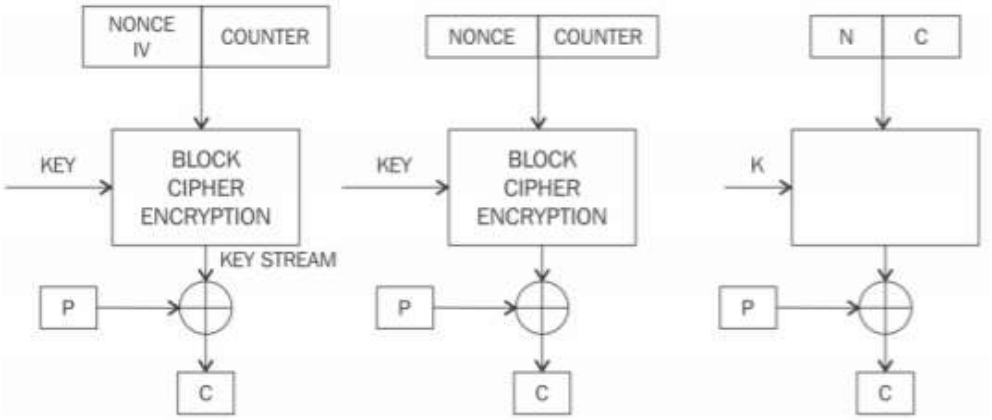


Figura 9. Modo contador[8]

Modo de generación de flujo de claves

En el flujo de claves, la función de cifrado genera un flujo de claves que luego se XOR con el flujo de texto plano para lograr el cifrado.

Modo de autenticación de mensajes

En el modo de autentificación de mensajes, un código de autenticación de mensajes resulta de una función de cifrado. El MAC es una suma de comprobación criptográfica que proporciona un servicio de integridad. El método más común para generar un MAC usando cifrados en bloque es CBC-MAC, donde una parte del último bloque de la cadena se usa como MAC. [8]

Modo hash criptográfico

Las funciones hash se utilizan principalmente para comprimir un mensaje en un resumen de longitud fija. En el modo hash criptográfico, los cifrados en bloque se utilizan como función de compresión para producir un hash de texto sin formato. [8]

Estándar de cifrado de datos

El estándar de cifrado de datos (DES) fue introducido por el Instituto Nacional de Estándares y Tecnología de EE. UU. (NIST) como un algoritmo estándar para el cifrado, y fue de uso generalizado durante los años ochenta y noventa. [8]

DES usa una clave de solo 56 bits, lo que generó algunas preocupaciones. Este problema se abordó con la introducción de Triple DES (3DES), que proponía el uso de una clave de 168 bits mediante tres claves de 56 bits y el mismo número de ejecuciones del algoritmo DES, haciendo casi imposibles los ataques de fuerza bruta.

Sin embargo, otras limitaciones, como un rendimiento lento y un tamaño de bloque de 64 bits, no eran deseables. [8]

Estándar de cifrado avanzado

Funcionamiento de AES

El procesamiento del algoritmo AES, una matriz de bytes de 4 x 4 conocida como estado, se modifica utilizando múltiples rondas. El cifrado completo requiere de 10 a 14 rondas, según el tamaño de la clave [8]

Una vez que se inicializa el estado con la entrada al cifrado, se realizan cuatro operaciones realizadas en cuatro etapas para cifrar la entrada. [8]

1. En el paso AddRoundKey, la matriz de estado se XOR con una subclave, que se deriva de la clave maestra
2. SubBytes es el paso de sustitución donde una tabla de búsqueda (S-box) se usa para reemplazar todos los bytes de la matriz de estado
3. El paso ShiftRows se usa para desplazar cada fila a la izquierda, excepto la primera, en la matriz de estado a la izquierda de manera cíclica e incremental
4. Finalmente, todos los bytes se mezclan en el paso MixColumns de forma lineal, en columnas.

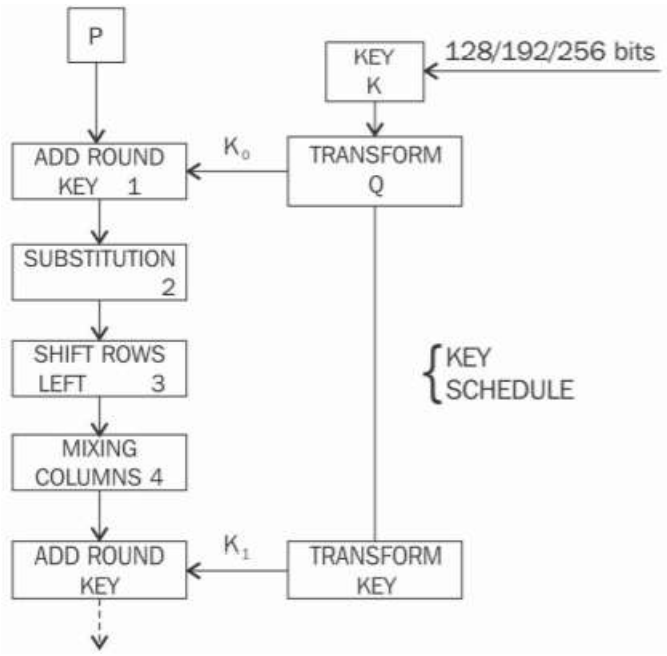


Figura 10. Diagrama de bloques AES, que muestra la primera ronda de cifrado AES[8]

1.3.3 Criptográfica de clave publica

Criptografía asimétrica

La criptografía asimétrica se refiere a un tipo de criptografía donde la clave que se utiliza para cifrar los datos es diferente de la clave que se utiliza para descifrar los datos. Esto también se conoce como criptografía de clave pública. Utiliza claves públicas y privadas para cifrar y descifrar datos, respectivamente. Se utilizan varios esquemas de criptografía asimétrica, incluidos RSA, DSA y El Gammal. [8]

En el siguiente diagrama se muestra una descripción general de la criptografía de clave pública:

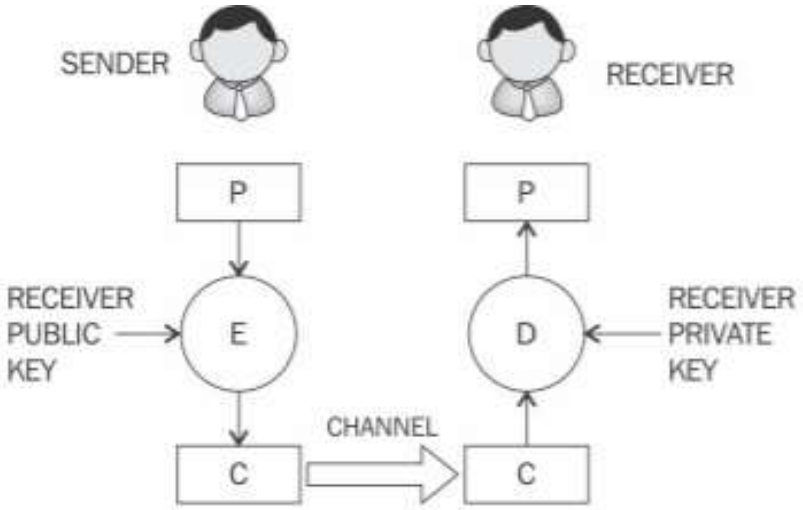


Figura 11. Cifrado y descifrado usando claves públicas y privadas[8]

Los mecanismos de seguridad que ofrecen los criptosistemas de clave pública incluyen el establecimiento de claves, firmas digitales, identificación, cifrado y descifrado.

Los mecanismos de establecimiento de claves tienen que ver con el diseño de protocolos que permitan configurar claves en un canal inseguro. Los servicios de no repudio, una propiedad muy deseable en muchos escenarios, se pueden proporcionar mediante firmas digitales. A veces, es importante no solo autenticar a un usuario, sino también identificar la entidad involucrada en una transacción. Esto también se puede lograr mediante una combinación de firmas digitales y protocolos de respuesta al desafío. Por último, el mecanismo de cifrado para proporcionar confidencialidad también se puede obtener utilizando criptosistemas de clave pública, como RSA, ECC y El Gammal.

Los algoritmos de clave pública son más lentos en términos de cálculo que los algoritmos de clave simétrica. Por lo tanto, no se usan comúnmente en el cifrado de archivos grandes o los datos reales que requieren cifrado. Suelen utilizarse para intercambiar claves por algoritmos simétricos. Una vez que las claves se establecen de forma segura, se pueden utilizar algoritmos de claves simétricas para cifrar los datos.

Los algoritmos de criptografía de clave pública se basan en varias funciones matemáticas subyacentes.

Las tres categorías principales de algoritmos asimétricos son:

Factorización de enteros

Los esquemas de factorización de enteros se basan en el hecho de que los números enteros grandes son muy difíciles de factorizar. RSA es el mejor ejemplo de este tipo de algoritmo. [8]

Logaritmo discreto

Un esquema de logaritmo discreto se basa en un problema de aritmética modular. Es fácil calcular el resultado de la función módulo, pero computacionalmente no es práctico encontrar el exponente del generador. En otras palabras, es extremadamente difícil encontrar la entrada del resultado. Esta es una función unidireccional. [8]

Curvas elípticas

Una curva elíptica es una curva cúbica algebraica sobre un campo, que se puede definir mediante la siguiente ecuación. La curva no es singular, lo que significa que no tiene cúspides ni auto intersecciones. Dispone de dos variables a y b,así como un punto de infinito.

$$y^2 = x^3 + ax + b$$

Claves públicas y privadas

Una clave privada, como su nombre indica, es un número generado aleatoriamente que sus usuarios mantienen en secreto y en privado. Las claves privadas deben estar protegidas y no se debe otorgar acceso no autorizado a esa clave; de lo contrario, todo el esquema de criptografía de clave pública se ve comprometido, ya que esta es la clave que se utiliza para descifrar los mensajes. [8]

Una clave pública está disponible gratuitamente y es publicada por el propietario de la clave privada. Cualquiera que desee enviar un mensaje cifrado al editor de la clave pública puede hacerlo cifrando el mensaje utilizando la clave pública publicada y enviándolo al titular de la clave privada. Nadie más puede descifrar el mensaje porque el destinatario previsto guarda la clave privada correspondiente de forma segura. Una vez que se recibe el mensaje cifrado con clave pública, el destinatario puede descifrar el mensaje utilizando la clave privada. Sin embargo, existen algunas preocupaciones con respecto a las claves públicas. Estos incluyen autenticidad e identificación del editor de las claves públicas.

RSA

Este tipo de criptografía de clave pública se basa en el problema de factorización de enteros, donde la multiplicación de dos números primos grandes es fácil, pero es difícil factorizar de nuevo a los dos números originales. El meollo del trabajo involucrado con el algoritmo RSA es durante el proceso de generación de claves. [8]

Se genera un par de claves RSA realizando los siguientes pasos:

1. **Generación Módulo:**
    - Seleccionar p y q, que son números primos muy grandes

- Multiplicar p y q, n = pq para generar módulo n

2. Generar coprimos:

- suponga un número llamado *e*.
- E debe satisfacer una determinada condición; es decir, debe ser mayor que 1 y menor que (p-1) (q-1). En otras palabras, e debe ser un número tal que no hay ningún número distinto de 1 puede dividir e y (p-1)(q-1).Esto se llama coprimo, es decir, e es el coprimo de (p-1) (q-1).

3. Generar la clave pública:

el módulo generado en el paso 1 y el coprimo e generado en el paso 2 es un par que es una clave pública. Esta parte es la parte pública que se puede compartir con cualquiera; sin embargo, p y q necesidad de mantenerse en secreto.

4. Generar la clave privada:

La clave privada, llamada d aquí, se calcula a partir de p, q, y e. La clave privada es básicamente la inversa de e modulo (p-1) (q-1). En la forma de la ecuación, es como el siguiente:

$$ed \ = \ 1 \ mod \ (p-1) \ (q-1)$$

Cifrado y descifrado mediante RSA

RSA utiliza la siguiente ecuación para producir texto cifrado:

$$C \ = \ Pe \ mod \ n$$

Esto significa que el texto plano *P* se eleva a *e* número de veces y luego se reduce a módulo *n*. El descifrado en RSA se proporciona en la siguiente ecuación:

$$P \ = \ Cd \ mod \ n$$

Esto significa que el receptor que tiene un par de claves públicas (*n*, *e*) puede descifrar los datos elevando *C* al valor de la clave privada *d y* reduciendo a módulo *n*.

Criptografía de curva elíptica

La criptografía de curva elíptica se basa en el problema del logaritmo discreto basado en curvas elípticas sobre campos finitos. El principal beneficio de ECC sobre otros tipos de algoritmos de clave pública es que requiere un tamaño de clave más pequeño mientras proporciona el mismo nivel de seguridad.

Matemáticas de ECC

Una curva elíptica es básicamente un tipo de ecuación polinomial conocida como ecuación de Weierstrass, que genera una curva sobre un campo finito. El campo más utilizado es donde todas las operaciones aritméticas se realizan módulo *a* primo *p*. Los grupos de curvas elípticas consisten en puntos en la curva sobre un campo finito.

Una curva elíptica se define en la siguiente ecuación:

$$y^2 = x^3 + Ax + B \ mod \ P$$

Suma de puntos

La suma de puntos se muestra en el siguiente diagrama. Ésta es una representación geométrica de la suma de puntos en curvas elípticas. En este método, se traza una línea diagonal a través de la curva que interseca la curva en dos puntos P y Q, como se muestra en el diagrama, lo que produce un tercer punto entre la curva y la línea. Este punto se refleja como P + Q, que representa el resultado de la suma como R.

Esto se muestra como P + Q en el siguiente diagrama:

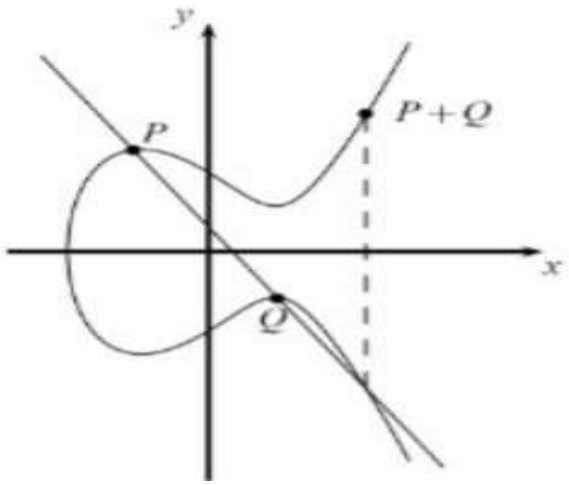


Figura 12. Suma de puntos sobre R[8]

La operación de grupo denotada por el + signo para la suma produce la siguiente ecuación:

$$P + Q = R$$

En este caso, se suman dos puntos para calcular las coordenadas del tercer punto de la curva:

$$P + Q = R$$

Más precisamente, esto significa que se suman las coordenadas, como se muestra en la siguiente ecuación:

$$(x1, y1) + (x2, y2) = (x3, y3)$$

La ecuación de la suma de puntos es la siguiente:

$$X3 = s2 - x1 - x2 \, mod \, p$$

$$y3 = s\,(x1 - x3) - y1\ mod\ p$$

Aquí, vemos que el resultado de la ecuación anterior:

$$s = \frac{y_2 - y_1}{(x2 - x1)} mod\ p$$

S en la ecuación anterior representa la línea que va a través de *P* y Q.

Duplicación de puntos

La otra operación de grupo en curvas elípticas se llama duplicación de puntos. Este es un proceso en el que P se agrega a sí mismo. En este método, se traza una línea tangente a través de la curva, como se muestra en el siguiente gráfico. Se obtiene el segundo punto, que se encuentra en la intersección de la tangente trazada y la curva.

Este punto luego se refleja para producir el resultado, que se muestra como *2P = P + P*:

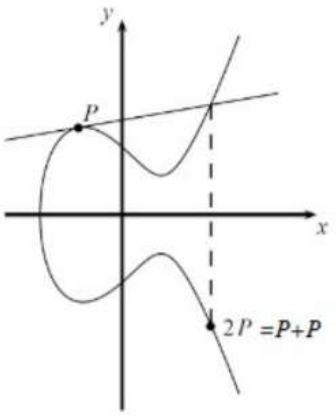


Figura 13. Gráfico que representa la duplicación de puntos sobre números reales[8]

En el caso de duplicación de puntos, la ecuación se convierte en:

$$x_3 = s^2 - x_1 - x_2\ mod p$$
$$y_3 = s(x_1 - x_2) - y_1\ mod p$$
$$s = \frac{3x_1^2 + a}{2y_1}$$

Funciones hash

Las funciones hash se utilizan para crear resúmenes de longitud fija de cadenas de entrada arbitrariamente largas. Las funciones hash no tienen llave y proporcionan el servicio de integridad de datos. Por lo general, se construyen utilizando técnicas de construcción de funciones hash iteradas y dedicadas. [8]

Las funciones hash también se utilizan normalmente para proporcionar servicios de integridad de datos. Estos se pueden utilizar como funciones unidireccionales y para construir otras primitivas criptográficas, como MAC y firmas digitales. Algunas aplicaciones utilizan funciones hash como un medio para generar Generador de números pseudoaleatorios (PRNG). Hay dos propiedades prácticas y tres de seguridad de las funciones hash que deben cumplirse según el nivel de integridad requerido. [8]

Tienen tres propiedades de seguridad, a saber, resistencia a preimagen, segunda resistencia a preimagen y resistencia a colisiones:

Compresión de mensajes arbitrarios en una longitud fija

Esta propiedad se relaciona con el hecho de que una función hash debe poder tomar un texto de entrada de cualquier longitud y generar un mensaje comprimido de longitud fija. Las funciones hash producen una salida comprimida en varios tamaños de bits, generalmente entre 128 y 512 bits. [8]

fáciles de calcular

Las funciones hash son funciones unidireccionales rápidas y eficientes. Se requiere que las funciones hash sean muy rápidas de calcular independientemente del tamaño del mensaje. La eficiencia puede disminuir si el mensaje es demasiado grande, pero la función debe ser lo suficientemente rápida para un uso práctico. [8]

Resistencia a la preimagen

Esta propiedad se puede explicar usando la ecuación simple que se muestra a continuación:

$$h\,(x)\;=\;y$$

Aquí, h es la función hash, x es la entrada e y es el hash. La primera propiedad de seguridad requiere que y no se pueda calcular a la inversa a x. x se considera una preimagen de y, de ahí el nombre de resistencia a preimagen. Esto también se denomina propiedad unidireccional. [8]

Segunda resistencia a la preimagen

La segunda propiedad de resistencia de la preimagen requiere que, dados x y h (x), es casi imposible encontrar cualquier otro mensaje m, ¡donde m! = X y hash de m = hash de x o h (m) = h (x). Esta propiedad también se conoce como resistencia a colisiones débil. [8]

Resistencia a la colisión

La propiedad de resistencia a la colisión requiere que dos mensajes de entrada diferentes no tengan la misma salida. En otras palabras, ¡h (x)! = H (z). Esta propiedad también se conoce como fuerte resistencia a colisiones. [8]

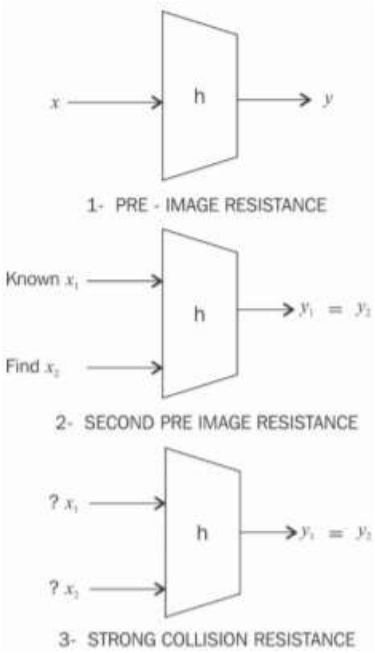


Figura 14. Tres propiedades de seguridad de las funciones hash[8]

Algoritmos de hash seguro

SHA-256

SHA-256 tiene un tamaño de mensaje de entrada menor a 264bits. El tamaño del bloque es de 512 bits y tiene un tamaño de palabra de 32 bits. La salida es un resumen de 256 bits. [8]

La función de compresión procesa un bloque de mensajes de 512 bits y un valor hash intermedio de 256 bits. Hay dos componentes principales de esta función: la función de compresión y un programa de mensajes. [8]

El algoritmo funciona de la siguiente manera, en ocho pasos:

Pre procesamiento:

1. El relleno del mensaje se utiliza para ajustar la longitud de un bloque a 512 bits si es más pequeño que el tamaño de bloque requerido de 512 bits.
2. Analizar el mensaje en bloques de mensajes, lo que garantiza que el mensaje y su relleno se dividen en bloques iguales de 512 bits.

3. Configuración del valor hash inicial, que consta de las ocho palabras de 32 bits obtenidas al tomar los primeros 32 bits de las partes fraccionarias de las raíces cuadradas de los primeros ocho números primos. Estos valores iniciales se eligen aleatoriamente para inicializar el proceso y proporcionan un nivel de confianza de que no existe una puerta trasera en el algoritmo. [8]

Cálculo hash:

4. Luego, cada bloque de mensajes se procesa en una secuencia y se requieren 64 rondas para calcular la salida de hash completa. Cada ronda utiliza constantes ligeramente diferentes para garantizar que no haya dos rondas iguales.
5. Se prepara la programación de mensajes.
6. Se inicializan ocho variables de trabajo.
7. Se calcula el valor hash intermedio.
8. Finalmente, se procesa el mensaje y se produce el hash de salida. Una ronda de una función de compresión SHA-256

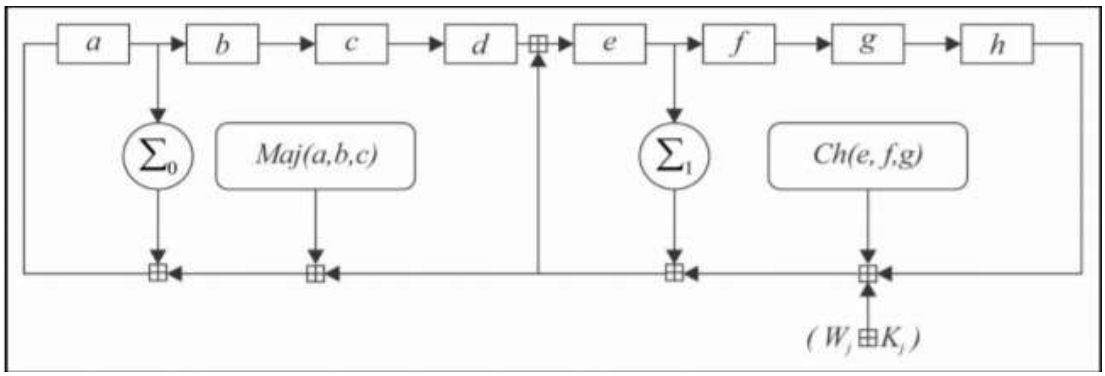


Figura 15. Una ronda de una función de compresión SHA-256 [8]

SHA-3

La estructura de SHA-3 es muy diferente a la de SHA-1 y SHA-2. La idea clave detrás de SHA-3 se basa en permutaciones sin clave, a diferencia de otras construcciones típicas de funciones hash que usaban permutaciones con clave. [8]

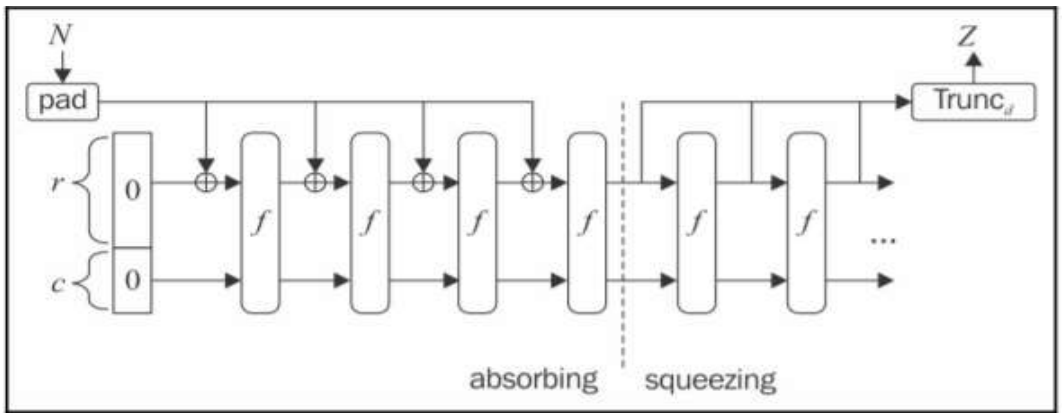


Figura 16. Función absorbente y exprimidora SHA-3 [8]

Firmas digitales

Las firmas digitales proporcionan un medio para asociar un mensaje con una entidad desde la cual se originó el mensaje. Las firmas digitales se utilizan para proporcionar autenticación y no repudio del origen de los datos. [8]

Las firmas digitales se utilizan en Blockchain donde las transacciones son firmadas digitalmente por los remitentes que usan su clave privada antes de transmitir la transacción a la red. [8]

Algoritmo de firma digital RSA

El siguiente es el algoritmo de firma digital RSA:

1. Calcule el valor hash del paquete de datos: esto proporcionará la garantía de integridad de los datos, ya que el hash se puede calcular nuevamente en el extremo del receptor y compararlo con el hash original para verificar si los datos se han modificado en tránsito. Técnicamente, la firma de mensajes puede funcionar sin aplicar hash a los datos primero, pero no se considera segura. [8]
2. Firma el valor hash con la clave privada del firmante: Como solo el firmante tiene la clave privada, se garantiza la autenticidad de la firma y los datos firmados. [8]

Las firmas digitales tienen algunas propiedades importantes, como autenticidad, imposibilidad de falsificación y no reutilización. Autenticidad significa que las firmas digitales son verificables por una parte receptora. La propiedad de imposibilidad de falsificar garantiza que solo el remitente del mensaje pueda utilizar la función de firma con la clave privada. En otras palabras, nadie más puede producir el mensaje firmado producido por un remitente legítimo. La no reutilización significa que la firma digital no puede separarse de un mensaje y volver a utilizarse para otro mensaje. [8]

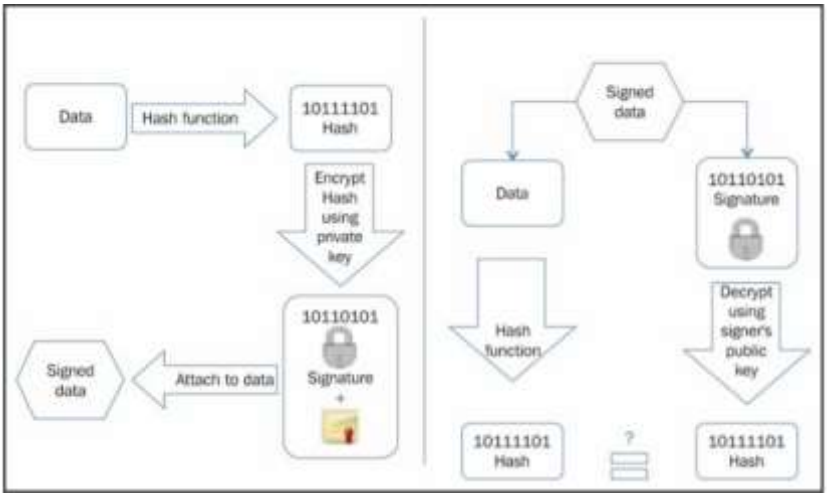


Figura 17. Firma digital y proceso de verificación [8]

Algoritmo de firma digital de curva elíptica

Para firmar y verificar utilizando el esquema ECDSA, se debe generar el primer par de claves:

1. En primer lugar, definir una curva elíptica E:
   - Con módulo P
   - coeficientes a y b
   - Generador punto A que forma un grupo cíclico de orden primo q
2. Según número entero d elige aleatoriamente para que 0 <d <q.
3. Calcule la clave pública B para que B = d A.

La clave pública es el séxtuple en la forma que se muestra aquí:

$$Kpb = (p, a, b, q, A, B)$$

La clave privada, d se elige aleatoriamente en el paso 2:

$$Kpr = d$$

Ahora la firma se puede generar usando la clave pública y privada.

4. Primero,una clave efímera K_e se elige, donde 0 <K_e <q. Debe asegurarse que K_e sea verdaderamente aleatorio y que no haya dos firmas con la misma clave; de lo contrario, se puede calcular la clave privada.
5. Otro valor R se calcula usando R = K_e A; es decir, multiplicando A (el punto generador) y la clave efímera aleatoria.

6. Inicialice una variable r con el x valor de la coordenada del punto R para que r = xR.
7. La firma se puede calcular de la siguiente manera:

aquí, m es el mensaje para el que se calcula la firma y h (m) es el hash del mensaje m.

La verificación de la firma se realiza siguiendo este proceso:

1. El valor auxiliar w se calcula como w = s-1 mod q.
2. Valor auxiliar u1 = w. h (m) mod q.
3. Valor auxiliar u2 = w. r mod q.
4. Calcule el punto P, P = u1A + u2B.
5. La verificación se lleva a cabo como sigue:

r, s se acepta como una firma válida si la x coordenada del punto P calculada en el paso 4 tiene el mismo valor que el parámetro de firma r mod q; es decir:

$$Xp = r \bmod q \; significa \; firma \; válida$$

$$Xp! = r \bmod q \; significa \; firma \; inválida$$

1.3.4 Contratos inteligentes

Definición

No hay consenso sobre una definición estándar de contratos inteligentes. Es esencial definir qué es un contrato inteligente, y lo siguiente es mi intento de proporcionar una definición generalizada de un contrato inteligente:

Un contrato inteligente es un programa informático seguro e imparable que representa un acuerdo que se puede ejecutar y hacer cumplir automáticamente.

La disección de esta definición revela además que un contrato inteligente es, de hecho, un programa de computadora que está escrito en un lenguaje que una computadora o máquina de destino puede entender. Además, engloba acuerdos entre partes en forma de lógica empresarial. Otra idea fundamental es que los contratos inteligentes se ejecutan automáticamente cuando se cumplen determinadas condiciones. Son ejecutables, lo que significa que todos los términos contractuales se ejecutan según lo definido y esperado, incluso en presencia de adversarios

Contratos ricardianos

La idea fundamental es redactar un documento que sea comprensible y aceptable tanto por un tribunal como por un software informático. Los contratos ricardianos abordan el desafío de la emisión de valor a través de Internet. Identifica al emisor y captura todos los términos y cláusulas del contrato en un documento para que sea aceptable como contrato legalmente vinculante.

En la práctica, los contratos se implementan mediante la producción de un solo documento que contiene los términos del contrato en prosa legal y las etiquetas legibles por máquina requeridas. Este documento está firmado digitalmente por el emisor utilizando su clave privada. Luego, este documento se codifica mediante una función de resumen de mensajes para producir un hash mediante el cual se puede identificar el documento. Este hash es luego utilizado y firmado por las partes durante

la ejecución del contrato para vincular cada transacción, y el hash del identificador sirve como prueba de la intención. Esto se muestra en el siguiente diagrama, generalmente llamado modelo de pajarita.

El diagrama muestra el número de elementos:

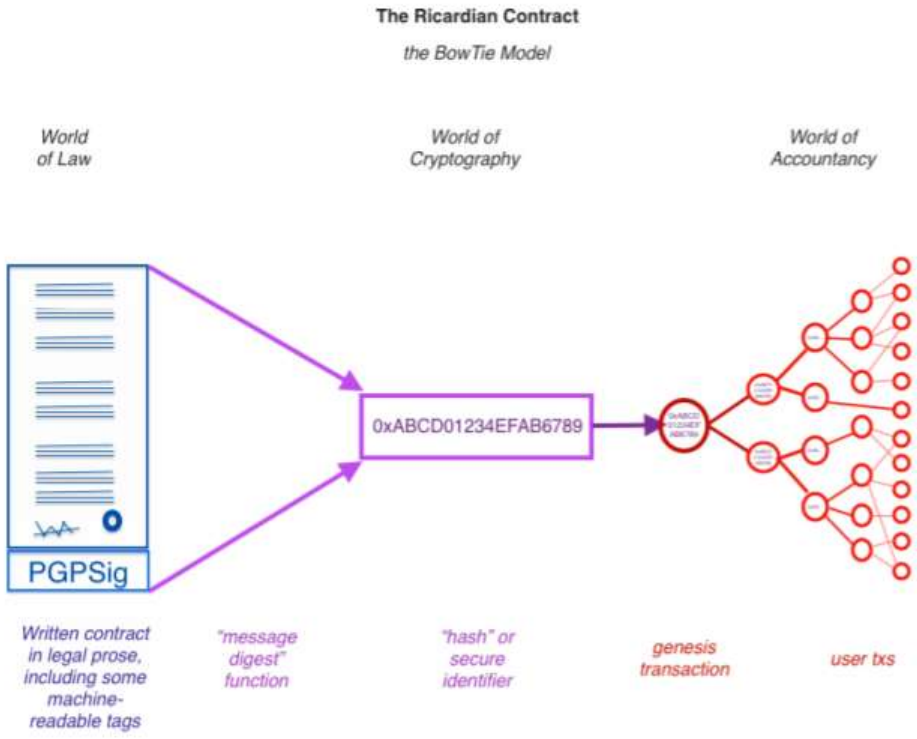


Figura 18. contratos ricardianos, diagrama de pajarita[8]

Plantillas de contratos inteligentes

Los contratos inteligentes se pueden implementar para cualquier industria donde sea necesario, pero la mayoría de los casos de uso actuales están relacionados con la industria financiera. Esto se debe al hecho de que Blockchain encontró primero muchos casos de uso en la industria financiera y despertó un gran interés de investigación en la industria financiera mucho antes que otras industrias. Un trabajo reciente en el espacio de contratos inteligentes específico para la industria financiera ha propuesto la idea de plantillas de contratos inteligentes. La idea es crear plantillas estándar que proporcionen un marco para respaldar los acuerdos legales para los instrumentos financieros. [8]

Los contratos en la industria financiera no son un concepto nuevo, y varios DSL de lenguaje específicos de dominio ya están en uso en la industria financiera para proporcionar un lenguaje específico para un dominio específico. Por ejemplo, existen DSL disponibles que respaldan el desarrollo de productos de seguros, representan derivados de energía o se utilizan para crear estrategias comerciales. [8]

También es importante comprender el concepto de lenguajes específicos de dominio, ya que este tipo de lenguajes se pueden desarrollar para programar contratos inteligentes. Estos lenguajes se desarrollan con expresividad limitada para una aplicación o área de interés en particular. Los lenguajes específicos de dominio son diferentes de los lenguajes de programación de propósito general. Las DSL tienen un pequeño conjunto de características que son suficientes y están optimizadas para el dominio en el que están diseñadas y, a diferencia de las GPL, generalmente no se usan para crear grandes programas de aplicación de propósito general. [8]

Sobre la base de la filosofía de diseño de los DSL, se puede prever que dichos lenguajes se desarrollarán específicamente para escribir contratos inteligentes. Ya se ha realizado algo de trabajo, y Solidity es uno de esos lenguajes que se ha introducido con Ethereum Blockchain para escribir contratos inteligentes. Vyper es otro lenguaje que se ha introducido recientemente para el desarrollo de contactos inteligentes de Ethereum. [8]

Oráculos

Los oráculos son un componente importante del ecosistema de contratos inteligentes. La limitación de los contratos inteligentes es que no pueden acceder a datos externos, que podrían ser necesarios para controlar la ejecución de la lógica empresarial; por ejemplo, el precio de las acciones de un producto de seguridad que requiere el contrato para liberar los pagos de dividendos. Los oráculos se pueden utilizar para proporcionar datos externos a contratos inteligentes. Un Oracle es una interfaz que entrega datos de una fuente externa a contratos inteligentes. [8]

Dependiendo de la industria y los requisitos, Oracles puede entregar diferentes tipos de datos que van desde informes meteorológicos, noticias del mundo real y

acciones corporativas hasta datos provenientes de Internet de las cosas. Los oráculos son entidades de confianza que utilizan un canal seguro para transferir datos a un contrato inteligente. [8]

Los oráculos también son capaces de firmar digitalmente los datos demostrando que la fuente de los datos es auténtica. Los contratos inteligentes pueden suscribirse a los Oráculos, y los contratos inteligentes pueden extraer los datos o los Oráculos pueden enviar los datos a los contratos inteligentes. También es necesario que Oracles no pueda manipular los datos que proporciona y debe poder proporcionar datos auténticos. Aunque se confía en los oráculos, es posible que en algunos casos los datos sean incorrectos debido a la manipulación. Por lo tanto, es necesario que los oráculos no puedan cambiar los datos. Esta validación se puede proporcionar mediante varios esquemas notariales, que se describen más adelante en este capítulo. [8]

El siguiente diagrama muestra un modelo genérico de un ecosistema de contratos inteligentes y de Oracle:

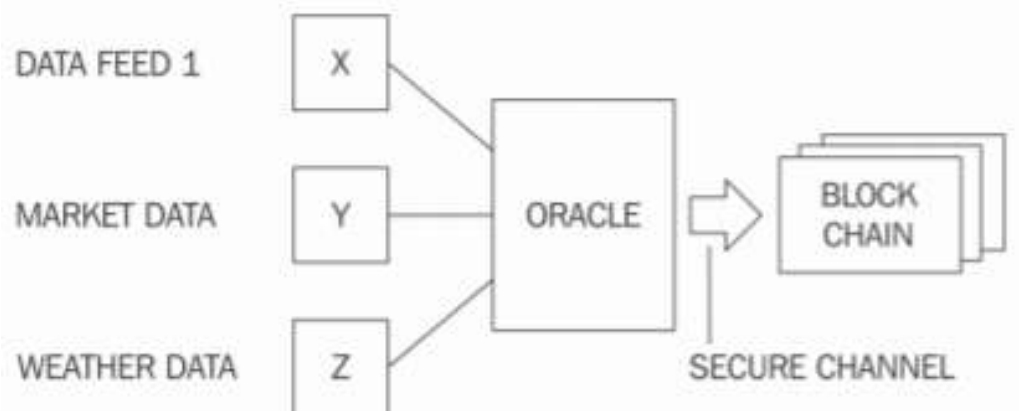


Figura 19.Un modelo genérico de un ecosistema de contratos inteligentes y de Oracle[8]

Oráculos inteligentes

Una idea de Oracle inteligente también ha sido propuesta por Ripple labs. Los Smart Oracles son entidades como Oracles, pero con la capacidad adicional de ejecución de código de contrato. Los Smart Oracles propuestos por Codius se ejecutan utilizando Google Native Client, que es un entorno de espacio aislado para ejecutar código nativo x86 que no es de confianza.

Implementar contratos inteligentes en una cadena de bloques

Los contratos inteligentes pueden implementarse o no en una cadena de bloques, pero tiene sentido implementarlos en una cadena de bloques debido al mecanismo de consenso distribuido y descentralizado proporcionado por la cadena de bloques.

Ethereum es un ejemplo de una plataforma Blockchain que admite de forma nativa el desarrollo y la implementación de contratos inteligentes. Los contratos inteligentes en la cadena de bloques Ethereum suelen ser parte de una aplicación más amplia, como la organización autónoma descentralizada. [8]

1.3.5 Ethereum

La idea fundamental era propuesta el desarrollo de un lenguaje Turing completo que permite el desarrollo de programas arbitrarios para Blockchain y aplicaciones descentralizadas. Este concepto contrasta con Bitcoin, donde el lenguaje de programación es de naturaleza limitada y solo permite las operaciones necesarias. [8]

Ethereum Blockchain

Ethereum, al igual que cualquier otro Blockchain, se puede visualizar como una máquina de estado basada en transacciones. [8]

La idea central es que en Ethereum Blockchain, un estado de génesis se transforma en un estado final mediante la ejecución de transacciones de forma incremental. La transformación final se acepta entonces como la versión absoluta indiscutible del estado. En el siguiente diagrama, se muestra la función de transición de estado de Ethereum, donde la ejecución de una transacción ha resultado en una transición de estado: [8]

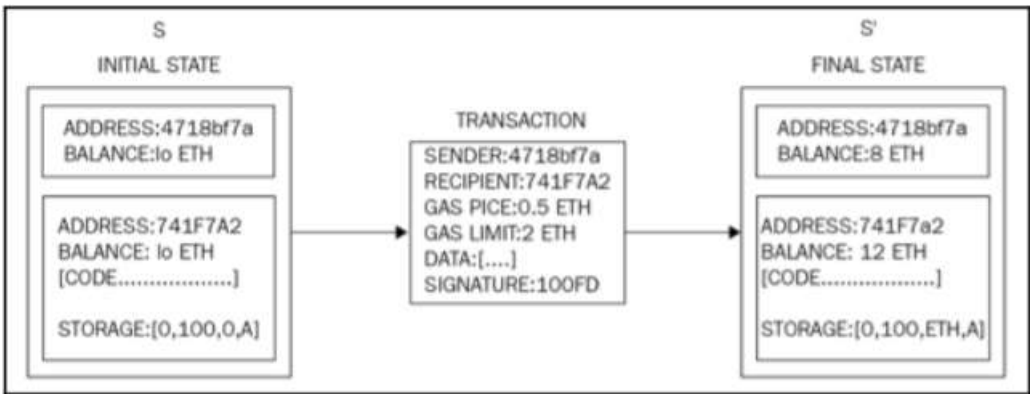


Figura 20. Función de transición de Estado Ethereum[8]

La red Ethereum

La red Ethereum es una red peer-to-peer donde los nodos participan para mantener la cadena de bloques y contribuir al mecanismo de consenso. Las redes se pueden dividir en tres tipos, según los requisitos y el uso. [8]

Mainnet

Es la red activa actual de Ethereum. La versión actual de mainnet es Byzantium y su ID de cadena es 1. El ID de cadena se utiliza para identificar la red. [8]

Testnet

Testnet también se llama Ropsten y es la red de prueba más utilizada para la cadena de bloques Ethereum. Esta cadena de bloques de prueba se utiliza para probar contratos inteligentes y DApps antes de implementarse en la cadena de bloques en vivo de producción. Además, al ser una red de prueba, permite la experimentación y la investigación. La red de prueba principal se llama Ropsten y contiene todas las

características de otras redes de prueba más pequeñas y de propósito especial que se crearon para lanzamientos específicos. [8]

Red privada

Esta es la red privada que se puede crear generando un nuevo bloque génesis. Este suele ser el caso en las redes de libros contables distribuidos de cadena de bloques privados, donde un grupo privado de entidades inicia su cadena de bloques y la usa como una cadena de bloques autorizada. [8]

Componentes del ecosistema Ethereum

La pila de Blockchain de Ethereum consta de varios componentes. En el núcleo, está la cadena de bloques Ethereum que se ejecuta en la red Ethereum peer-to-peer. En segundo lugar, hay un cliente Ethereum que se ejecuta en los nodos y se conecta a la red Ethereum peer-to-peer desde donde se descarga y almacena la cadena de bloques localmente. Proporciona varias funciones, como minería y administración de cuentas. La copia local de la cadena de bloques se sincroniza regularmente con la red. Otro componente es la biblioteca web3.js que permite la interacción con el cliente Geth a través de la llamada de la interfaz procedimiento remoto. [8]

Esta arquitectura se puede visualizar en el siguiente diagrama:

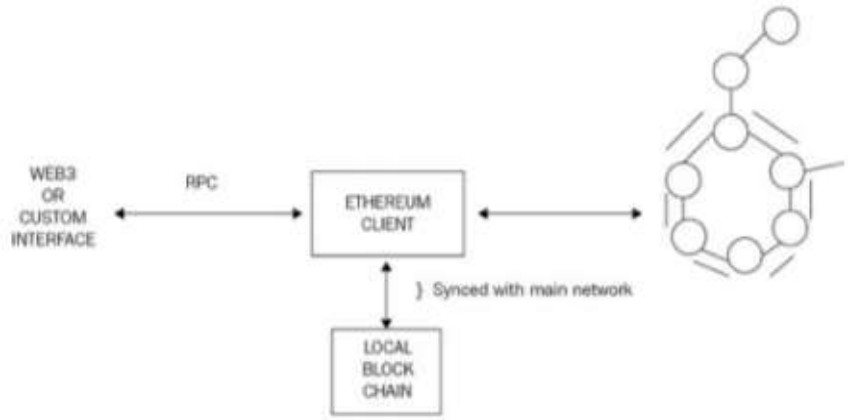


Figura 21. La pila de Ethereum que muestra varios componentes[8]

Los elementos de alto nivel presentes en la cadena de bloques Ethereum:

- Claves y direcciones
- Cuentas
- Transacciones y mensajes
- Criptomonedas / tokens Ether
- EVM
- contratos inteligentes

Claves y direcciones

Las claves y direcciones se utilizan en Ethereum Blockchain principalmente para representar la propiedad y transferencia de Ether. Las claves se utilizan en pares de tipo público y privado. La clave privada se genera aleatoriamente y se mantiene en

secreto, mientras que una clave pública se deriva de la clave privada. Las direcciones se derivan de las claves públicas que son un código de 20 bytes que se utiliza para identificar cuentas. [8]

1. Primero, se elige aleatoriamente una clave privada según las reglas definidas por la especificación secp256k1 de la curva elíptica.
2. Luego, la clave pública se deriva de esta clave privada mediante la función de recuperación ECDSA. Discutiremos esto más adelante en la siguiente sección, *Cuentas* en el contexto de firmas digitales.
3. Una dirección se deriva de la clave pública, que es la más correcta de 160 bits del hash Keccak de la clave pública.

Ejemplo de cómo se ven las claves y direcciones en Ethereum:

Clave privada:
b51928c22782e97cca95c490eb958b06fab7a70b9512c38c36974f47b954ffc 4

Clave pública:

3aa5b8eefd12bdc2d26f1ae348e5f383480877bda6f9e1a47f6a4afb35cf998
ab847f1e3948b1173622dafc6b4ac198c97b18fe1d79f90c9093ab2ff9ad992 60

Dirección: 0x77b4b5699827c5c49f73bd16fd5ce3d828c36f32

Cuentas

Las cuentas son uno de los principales componentes básicos de la cadena de bloques Ethereum. Ethereum, al ser una máquina de estado impulsada por transacciones, el estado se crea o actualiza como resultado de la interacción entre las cuentas y la ejecución de la transacción. Las operaciones realizadas entre y sobre las cuentas representan transiciones de estado. La transición de estado se logra utilizando lo que se llama la función de transición de estado de Ethereum, que funciona de la siguiente manera: [8]

Tipos de cuentas

Existen dos tipos de cuentas en Ethereum:

- Cuentas de propiedad externa (EOAs)
- Cuentas de contrato (CAs)

El primer tipo es EOA y el otro es CA. Las EOA son similares a las cuentas controladas por una clave privada en Bitcoin. Las CA son las cuentas que tienen un código asociado junto con la clave privada. [8]

EOs:

- EOA tiene saldo de ether
- Son capaces de enviar transacciones
- No tienen código asociado
- Están controladas por claves privadas Las
- cuentas contienen un almacén de clave-valor
- Están asociadas con un usuario humano

CAs:

- Las CA tienen equilibrio de Ether.
- Tienen un código asociado que se guarda en la memoria / almacenamiento en la cadena de bloques.
- Pueden activarse y ejecutar código en respuesta a una transacción o un mensaje de otros contratos.
- las CA pueden mantener su estado permanente y pueden convocar otros contratos. Se prevé que, en el comunicado de serenidad, se pueda eliminar la distinción entre cuentas de propiedad externa y cuentas de contrato.
- No están intrínsecamente asociados con ningún usuario o actor en la cadena de bloques.
- Las CA contienen un almacén de valores-clave.

Transacciones y mensajes

Una transacción en Ethereum es un paquete de datos firmado digitalmente que utiliza una clave privada que contiene las instrucciones que, cuando se completan, dan como resultado una llamada de mensaje o la creación de un contrato. Las transacciones se pueden dividir en dos tipos según el resultado que producen:

- **Transacciones de llamada de mensaje**: esta transacción simplemente produce una llamada de mensaje que se utiliza para pasar mensajes de una cuenta de contrato a otra.
- **Transacciones de creación de contrato**: como su nombre indica, estas transacciones dan como resultado la creación de una nueva cuenta de contrato. Esto significa que cuando esta transacción se ejecuta con éxito, crea una cuenta con el código asociado.

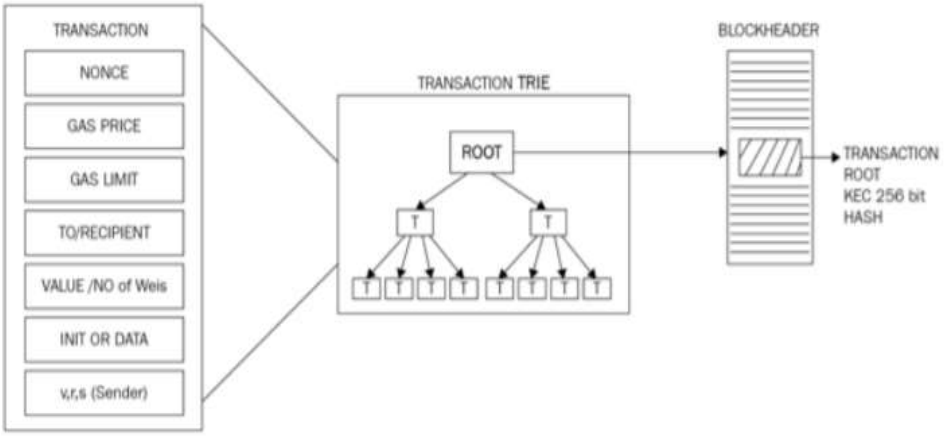


Figura 22.La relación entre transacción, ensayo de transacción y encabezado de bloque[10]

Mensajes

Los mensajes son los datos y el valor que se pasan entre dos cuentas. Un **mensaje** es un paquete de datos que se pasa entre dos cuentas. Este paquete de datos contiene datos y valor. Puede enviarse a través de un contrato inteligente o desde un actor externo en forma de una transacción que ha sido firmada digitalmente por el remitente.

Los contratos pueden enviar mensajes a otros contratos. Los mensajes solo existen en el entorno de ejecución y nunca se almacenan. Los mensajes son similares a las transacciones; sin embargo, la principal diferencia es que son producidos por los contratos, mientras que las transacciones son producidas por entidades externas al entorno Ethereum.

Un mensaje consta de los componentes mencionados aquí:

- El remitente del mensaje
- Destinatario del mensaje
- Cantidad de Wei a transferir y mensaje a la dirección del contrato
- Campo de datos opcional
- La cantidad máxima de gas que se puede consumir

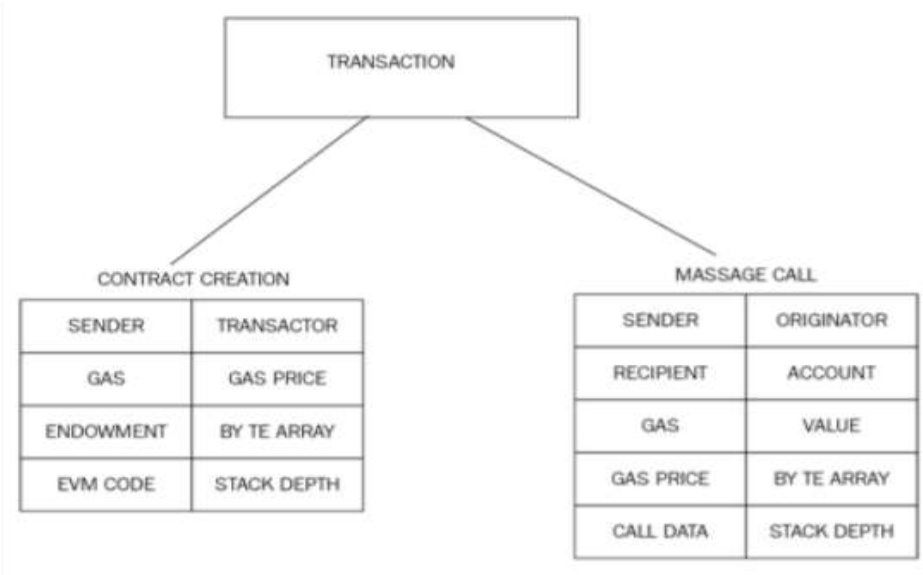


Figura 23.Tipos de transacciones, parámetros necesarios para su ejecución[10]

Llamadas

Una llamada no transmite nada a la cadena de bloques; en cambio, es una llamada local a una función de contrato y se ejecuta localmente en el nodo. Es casi como una llamada de función local. No consume gas ya que es una operación de solo lectura. Es similar a una carrera en seco o una carrera simulada. Las llamadas se ejecutan localmente en una máquina virtual de nodo y no dan como resultado ningún cambio de estado porque nunca se extraen.

Validación y ejecución de transacciones

Las transacciones se ejecutan después de verificar la validez de las transacciones. Las pruebas iniciales se enumeran de la siguiente manera:

- Una transacción debe estar bien formada y codificada con RLP sin bytes finales adicionales
- La firma digital utilizada para firmar la transacción es válida El valor de la
- transacción debe ser igual al límite de gas del nonce actual de la cuenta del remitente no debe ser menor que el gas utilizado por la transacción

- La cuenta del remitente contiene saldo suficiente para cubrir el costo de ejecución.

Almacenamiento de estado en la cadena de bloques Ethereum

En un nivel fundamental, la cadena de bloques Ethereum es una máquina de estado impulsada por transacciones y consenso. El estado debe almacenarse permanentemente en la cadena de bloques. Para este propósito, el estado mundial, las transacciones y los recibos de transacciones se almacenan en Blockchain en bloques. A continuación, discutimos estos componentes. [8]

El estado mundial

Es un mapeo entre las direcciones de Ethereum y los estados de las cuentas. Las direcciones tienen una longitud de 20 bytes. Esta asignación es una estructura de datos que se serializa mediante el prefijo de longitud recursiva. [8]

RLP es un esquema de codificación especialmente desarrollado que se utiliza en Ethereum para serializar datos binarios para su almacenamiento o transmisión a través de la red y también para guardar el estado en un árbol Patricia en los medios de almacenamiento. La función RLP toma un elemento como entrada, que puede ser una cadena o una lista de elementos y produce bytes sin procesar que son adecuados para el almacenamiento y la transmisión a través de la red. RLP no codifica datos; en cambio, su propósito principal es codificar estructuras. [8]

La máquina virtual de Ethereum (EVM)

EVM es una máquina de ejecución simple basada en pilas que ejecuta instrucciones de código de bytes para transformar el estado del sistema de un estado a otro. El tamaño de palabra de la máquina virtual se establece en 256 bits. El tamaño de la pila está limitado a 1024 elementos y se basa en la último en entrar, primero en salir.

EVM es una máquina completa de Turing, pero está limitada por la cantidad de gas que se requiere para ejecutar cualquier instrucción. Esto significa que los bucles infinitos que pueden resultar en ataques de denegación de servicio no son posibles debido a los requisitos de gas. EVM también admite el manejo de excepciones, en caso de que ocurran, como no tener suficiente gas o instrucciones no válidas, en cuyo caso la máquina se detendría inmediatamente y devolvería el error al agente ejecutor. [8]

EVM es un entorno de ejecución completamente aislado y en espacio aislado. El código que se ejecuta en el EVM no tiene acceso a ningún recurso externo, como una red o un sistema de archivos. Esto da como resultado una mayor seguridad, una ejecución determinista y permite que se ejecute código que no es de confianza en Ethereum Blockchain. [8]

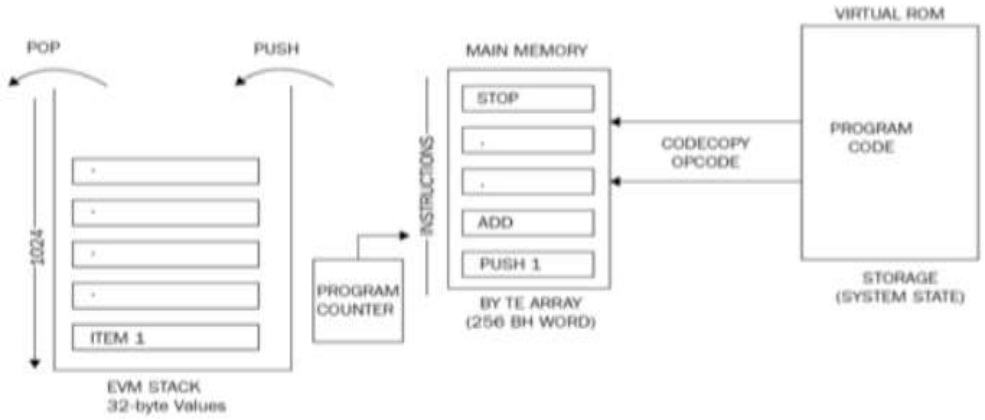


Figura 24.Trie de cuentas, tupla de cuenta, trie de estado mundial y hash de raíz de estado y su relación[8]

Contratos

Basta decir aquí que Ethereum apoya el desarrollo de contratos inteligentes que se ejecutan en el EVM. También hay varios contratos que están disponibles en el formato pre compilado en Ethereum Blockchain para admitir diferentes funciones. Estos contratos, conocidos como contratos pre compilados o contratos nativos, se describen en las siguientes subsecciones. [8]

Estos no son contratos estrictamente inteligentes en el sentido de contratos inteligentes de solidez programados por el usuario, sino que, de hecho, son funciones que están disponibles de forma nativa en la cadena de bloques para admitir varias tareas computacionalmente intensivas. Se ejecutan en el nodo local y están codificados dentro del cliente Ethereum, por ejemplo, paridad o Geth. [8]

1.3.6 Internet de las cosas

IoT se puede definir como una red de objetos físicos computacionalmente inteligentes que son capaces de conectarse a Internet, detectar eventos o entornos del mundo real, reaccionar a esos eventos, recopilar datos relevantes y comunicarnos a través de Internet. [8]

Esta simple definición tiene enormes implicaciones y ha dado lugar a conceptos interesantes, como dispositivos portátiles, hogares inteligentes, redes inteligentes, automóviles conectados inteligentes y ciudades inteligentes, todos ellos basados en este concepto básico de un dispositivo de IoT. Después de analizar la definición de IoT, se descubren cuatro funciones realizadas por un dispositivo de IoT. Estos incluyen sentir, reaccionar, recopilar y comunicarse. Todas estas funciones se realizan mediante el uso de varios componentes en el dispositivo IoT. [10]

La detección se realiza mediante sensores. La reacción o el control se realiza mediante actuadores, la recopilación es una función de varios sensores y la comunicación se realiza mediante chips que proporcionan conectividad de red. Una cosa a tener en cuenta es que todos estos componentes son accesibles y controlables a través de Internet en el IoT. Un dispositivo de IoT por sí solo es quizás útil hasta cierto punto, pero si es parte de un ecosistema de IoT más amplio, es más valioso. [10]

Un IoT típico puede consistir en muchos objetos físicos que se conectan entre sí y a un servidor en la nube centralizado. Esto se muestra en el siguiente diagrama:

Figura 25. Sistemas IoT[10]

Los elementos del IoT se distribuyen en varias capas y existen varias arquitecturas de referencia que se pueden utilizar para desarrollar sistemas de IoT. Se puede utilizar un modelo de cinco capas para describir IoT, que contiene una capa de objeto físico, una capa de dispositivo, una capa de red, una capa de servicios y una capa de aplicación. Cada capa o nivel es responsable de varias funciones e incluye múltiples componentes. [10]

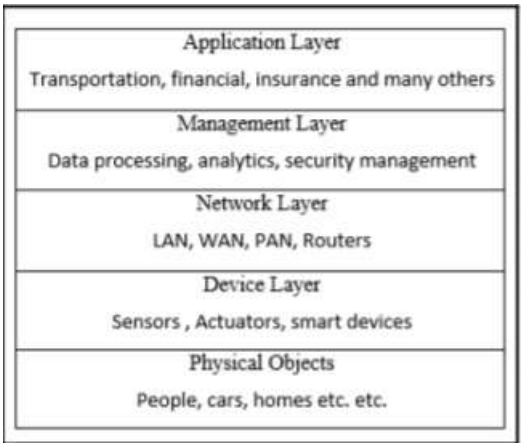


Figura 26. Capas de sistema IoT[10]

Capa de red

Esta capa se compone de varios dispositivos de red que se utilizan para proporcionar conectividad a Internet entre dispositivos y la nube o servidores que forman parte del ecosistema de IoT. Estos dispositivos pueden incluir puertas de enlace, enrutadores, concentradores y conmutadores. Esta capa puede incluir dos tipos de comunicación. [10]

Primero está el medio de comunicación horizontal, que incluye radio, Bluetooth, Wi-Fi, Ethernet, LAN, Zigbee y PAN y se puede utilizar para proporcionar comunicación entre dispositivos de IoT. En segundo lugar, tenemos comunicación con la siguiente capa, que generalmente es a través de Internet y proporciona comunicación entre máquinas y personas u otras capas superiores. La primera capa se puede incluir opcionalmente en la capa del dispositivo, ya que reside físicamente en la capa del dispositivo donde los dispositivos pueden comunicarse entre sí en la misma capa. [10]

Capa de gestión

Esta capa proporciona la capa de gestión para el ecosistema de IoT. Esto incluye plataformas que permiten el procesamiento de datos recopilados de los dispositivos de IoT y lo convierten en información significativa. Además, la gestión de dispositivos, la gestión de la seguridad y la gestión del flujo de datos se incluyen en esta capa. También gestiona la comunicación entre el dispositivo y las capas de la aplicación. [10]

Capa de aplicación

Esta capa incluye aplicaciones que se ejecutan en la parte superior de la red de IoT. Esta capa puede constar de muchas aplicaciones en función de los requisitos, como transporte, asistencia sanitaria, finanzas, seguros o gestión de la cadena de suministro. Esta lista, por supuesto, no es una lista exhaustiva por ningún tramo de la imaginación; Hay una gran cantidad de aplicaciones de IoT que pueden caer en esta capa. [10]

1.3.7 Integración de Blockchain y IoT

los sistemas de IoT se enfrentan a muchos desafíos, como la heterogeneidad de los sistemas de IoT, la interoperabilidad deficiente, las limitaciones de recursos de los dispositivos de IoT, las vulnerabilidades de privacidad y seguridad. Las tecnologías Blockchain pueden complementar los sistemas de IoT con la interoperabilidad mejorada y la privacidad y seguridad mejoradas. Además, Blockchain también puede mejorar la confiabilidad y escalabilidad de los sistemas IoT [10]

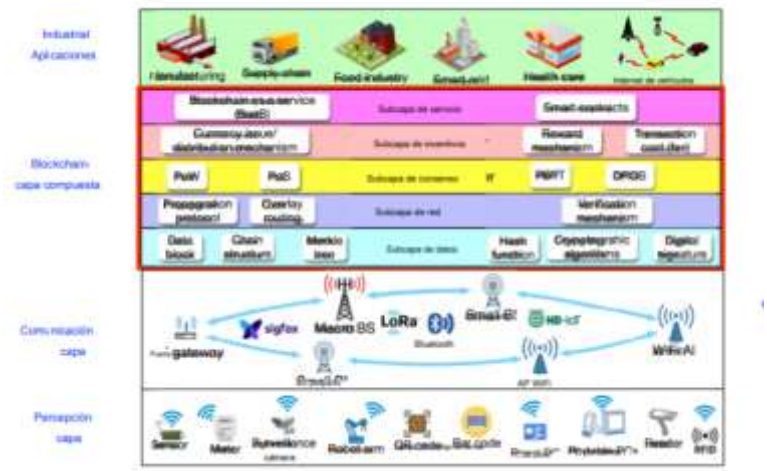


Figura 27. Capas con Integración IoT[10]

Interoperabilidad mejorada de los sistemas de IoT:

Blockchain esencialmente puede mejorar la interoperabilidad de los sistemas de IoT mediante la transformación y el almacenamiento de datos de IoT en Blockchain. Durante este procedimiento, los tipos heterogéneos de datos de IoT se convierten, procesan, extraen, comprimen y finalmente almacenan en Blockchain [10]

Seguridad mejorada de los sistemas de IoT:

Por un lado, los datos de IoT se pueden proteger mediante Blockchain, ya que se almacenan como transacciones de Blockchain que están encriptadas y firmadas digitalmente por claves criptográficas. Además, la integración de sistemas IoT con tecnologías Blockchain puede ayudar a mejorar la seguridad de los sistemas IoT

actualizando automáticamente los firmwares de los dispositivos IoT para remediar brechas vulnerables mejorando así la seguridad del sistema. [10]

Trazabilidad y Fiabilidad de datos de IoT

Los datos de Blockchain se pueden identificar y verificar en cualquier lugar y en cualquier momento. Mientras tanto, todas las transacciones históricas almacenadas en las cadenas de bloques son rastreable. Por ejemplo, el trabajo de ha desarrollado un sistema de trazabilidad de productos basado en Blockchain, que proporciona a los proveedores y minoristas servicios rastreables. [10]

Autonómico interacciones de los sistemas de IoT

Las tecnologías de cadena de bloques pueden permitir que los dispositivos o subsistemas de IoT interactúen entre sí automáticamente. [10]

Arquitectura de Blockchain e IoT

En esta arquitectura, la capa compuesta de Blockchain juega como un middleware entre IoT y aplicaciones industriales.

Este diseño tiene dos méritos:

1) ofrecer una abstracción de las capas inferiores en IoT

2) proporcionar a los usuarios servicios basados en Blockchain.

En particular, la capa compuesta de Blockchain oculta la heterogeneidad de las capas inferiores. Por otro lado, la capa compuesta de Blockchain ofrece una serie de servicios basados en Blockchain, que son esencialmente interfaces de programación de aplicaciones para admitir diversas aplicaciones industriales. Como resultado, la dificultad de desarrollar aplicaciones industriales también se puede reducir debido a la abstracción lograda por la capa compuesta de Blockchain. [10]

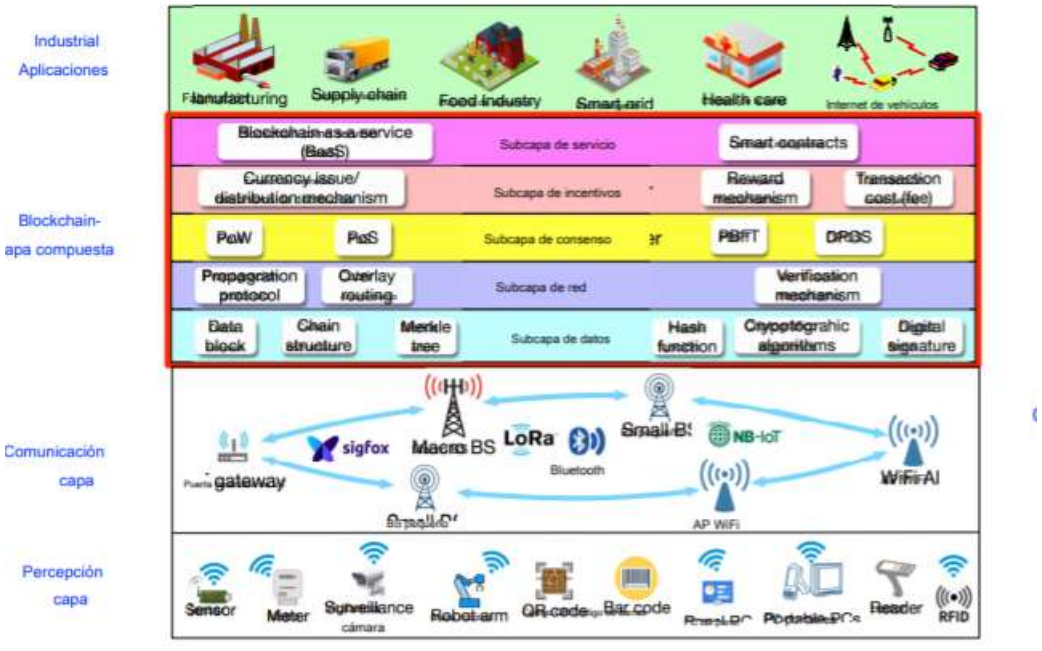


Figura 28. Arquitectura IoT con Blockchain[10]

1. Subcapa de datos recopila los datos de IoT de la capa inferior y envuelve los cifrados datos con firma digital mediante algoritmos criptográficos asimétricos y funciones hash. Estos bloques de datos conectados consecutivamente forman la cadena de bloques después de la validación distribuida. Las diferentes plataformas de Blockchain pueden elegir diferentes algoritmos criptográficos y funciones hash. [10]
2. Subcapa de red es esencialmente una red P2P superpuesta trabajo que se ejecuta en la parte superior de la capa de comunicación. La red superpuesta consta de enlaces virtuales o físicos que conectan nodos en las redes de comunicación subyacentes. Un nodo simplemente transmite el bloque de transacciones a sus pares conectados. Una vez recibido el bloque de transacciones, otros pares lo verificarán localmente. Si es válido, el bloque se propagará más a otros nodos a través de la red superpuesta. [10]
3. Subcapa de consenso está principalmente involucrado con el consenso homenajeado por la confianza de un bloque. El consenso se puede lograr mediante varios algoritmos de consenso como PoW, PoS, PBFT y DPOS. Vale la pena mencionar que los mecanismos de propagación de bloques son el requisito previo para los protocolos de consenso distribuidos. [10]
4. Subcapa de incentivos es responsable de lo siguiente tareas:
    1) emisión de moneda digital
    2) distribución de moneda digital
    3) diseño de mecanismos de recompensa
    4) manejo de costos de transacción.
5. Subcapa de servicio proporciona a los usuarios con Blockchain Los servicios para diversos sectores industriales incluyen fabricación, logística, cadenas de suministro, industria alimentaria y servicios públicos. La cadena de bloques como servicio (BaaS) se puede lograr mediante contratos inteligentes, que se pueden activar automáticamente cuando ocurre un evento especial. [10]

1.3.8 Reconocimiento facial

El reconocimiento facial se puede dividir en unas pocas etapas simples y esas etapas se pueden dividir en etapas más sofisticadas. Al principio, las imágenes se capturan con la ayuda de una cámara y luego las imágenes se toman como entradas. Las caras se diferencian de las imágenes y solo las características importantes de una cara se mantienen en la base de datos, lo que reduce la complejidad del espacio y, a su vez, la complejidad computacional general. Después de eso, la máquina está capacitada con estas características para una evaluación adicional. En el diagrama de flujo de la Figura 29 se muestra un procedimiento de reconocimiento facial simple. [11]

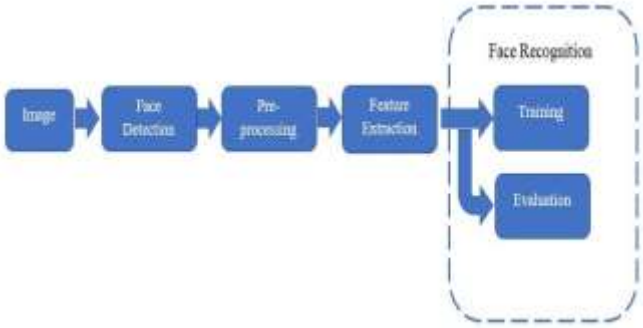


Figura 29.Diagrama de flujo de reconocimiento facial[11]

Detector de caras: Algoritmo de Viola Jones

Clasificadores en cascada

El clasificador en cascada, es decir, la cascada de clasificadores potenciados que trabajan con características similares a las de Haar, es un caso especial de aprendizaje por conjuntos, llamado refuerzo. Por lo general, se basa en clasificadores Adaboost

Los clasificadores en cascada se entrenan en unos cientos de imágenes de muestra de la imagen que contienen el objeto que queremos detectar y otras imágenes que no contienen esas imágenes. [11]

El algoritmo, llamado marco de detección de objetos Viola-Jones, que incluye todos los pasos necesarios para la detección de rostros:

- Selección de características de Haar, características derivadas de wavelets de Haar
- Crear imagen integral
- Entrenamiento Adaboost
- Clasificadores en cascada

1. Selección de características de Haar

Hay algunas características comunes que encontramos en los rostros humanos más comunes: [11]

- Una región de ojos oscura en comparación con la parte superior de las mejillas
- Una región brillante del puente de la nariz en comparación con los ojos

- Alguna ubicación específica de ojos, boca, nariz

Las características se denominan características Haar. El proceso de extracción de características se verá así:

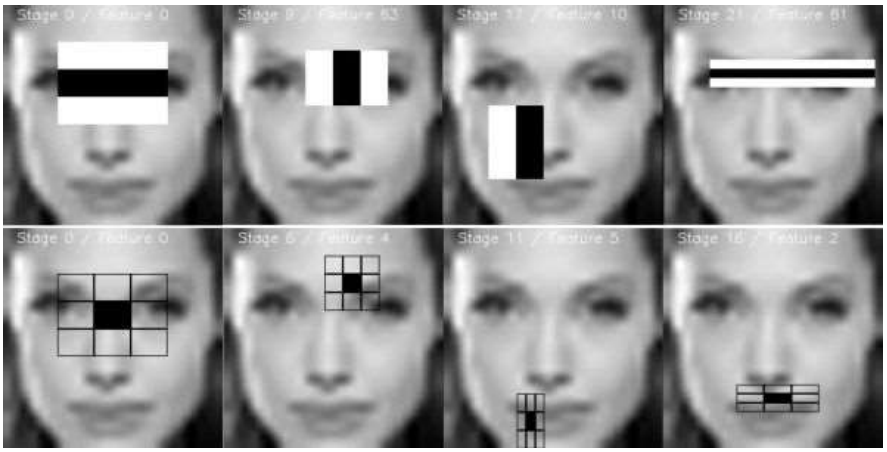


Figura 30. Proceso de extracción de características [11]

El valor de la característica se calcula simplemente sumando los píxeles en el área negra y restando los píxeles en el área blanca.

$$RectangleFeature = \sum(pixelsblackarea) - \sum(pixelswhitearea)$$

Aplicamos este rectángulo como un núcleo convolucional, sobre toda nuestra imagen. Para ser exhaustivos, debemos aplicar todas las dimensiones y posiciones posibles de cada kernel. Una simple imagen de 24 * 24 normalmente daría como resultado más de 160.000 características, cada una hecha de una suma / resta de valores de píxeles.

- Una vez que la región buena ha sido identificada por un rectángulo, es inútil ejecutar la ventana sobre una región completamente diferente de la imagen. Esto se puede lograr con Adaboost.
- Calcule las características del rectángulo utilizando el principio de imagen integral, que es mucho más rápido.

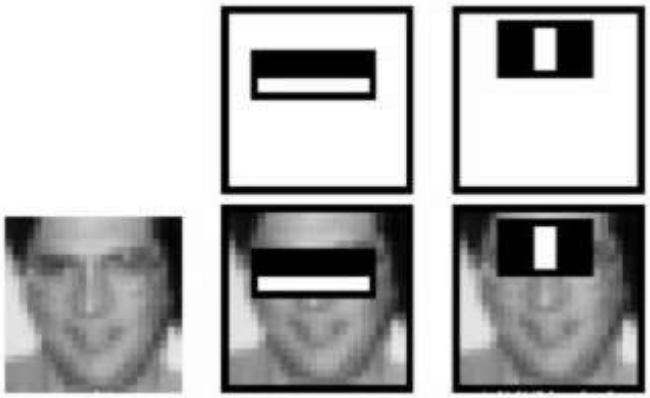


Figura 31. Extracción de características [11]

Hay varios tipos de rectángulos que se pueden aplicar para la extracción de características Haar.

- La característica de dos rectángulos es la diferencia entre la suma de los píxeles dentro de dos regiones rectangulares, que se utiliza principalmente para detectar bordes (a, b)
- la función de tres rectángulos calcula la suma dentro de dos rectángulos exteriores restada de la suma en un rectángulo central, que se utiliza principalmente para detectar líneas (c, d)
- la característica de cuatro rectángulos calcula la diferencia entre pares diagonales del rectángulo (e)

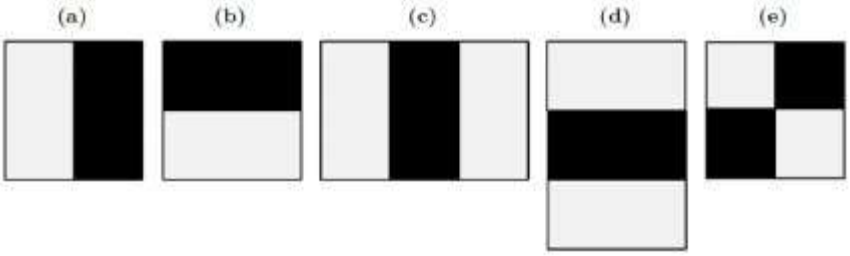


Figura 32. Característica calculada dentro de rectángulos [11]

Ahora que se han seleccionado las características, se aplica en el conjunto de imágenes de entrenamiento usando la clasificación de Adaboost, que combina un conjunto de clasificadores débiles para crear un modelo de conjunto preciso. [11]

2. La imagen integral

Calcular las características del rectángulo en un estilo de kernel Convolucional puede ser largo, muy largo. Por ello, los autores, Viola y Jones, propusieron una representación intermedia de la imagen: la imagen integral. La función de la imagen integral es permitir que cualquier suma rectangular se calcule de forma sencilla, utilizando sólo cuatro valores. [11]

Supongamos que queremos determinar las características del rectángulo en un píxel dado con coordenadas (X, y). Luego, la imagen integral del píxel en la suma de los píxeles arriba y a la izquierda del píxel dado. [11]

$$ii(x,y) = \sum_{X\prime \le X, y\prime \le y} \text{yo}(X', y')$$

dónde $ii(x,y)$ es la imagen integral y (X,y) es la imagen original.

Cuando calcula la imagen integral completa, hay una forma de recurrencia que requiere solo una pasada sobre la imagen original. De hecho, podemos definir el siguiente par de recurrencias: [11]

$$s(X,y) = s(X,y-1) + yo(X,y)$$

$$i(x,y) = ii(x-1,y) + s(x,y)$$

dónde (X,y) es la suma acumulada de filas y $s(X-1) = 0, ii(-1,y) = 0 s(X-1) = 0, ii(-1,y) = 0$. [11]

Considere una región D para la que nos gustaría estimar la suma de los píxeles. Hemos definido otras 3 regiones: A, B y C.

- El valor de la imagen integral en el punto 1 es la suma de los píxeles en el rectángulo A-El valor en el punto 2 es A + B
- El valor en el punto 3 es A + C
- El valor en el punto 4 es A + B + C + D.

Por lo tanto, la suma de píxeles en la región D se puede calcular simplemente como: 4+1-(2+3)4+1-(2+3). [11]

Y en una sola pasada, hemos calculado el valor dentro de un rectángulo usando solo 4 referencias de matriz. [11]

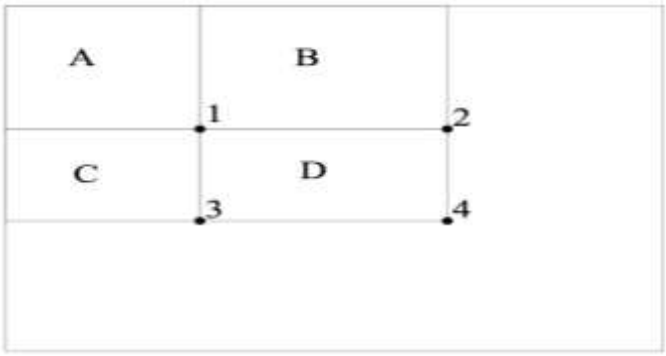


Figura 33. Referencia de Matriz [12]

Uno simplemente debe tener en cuenta que los rectángulos son características bastante simples en la práctica, pero suficientes para la detección de rostros. Los filtros orientables tienden a ser más flexibles cuando se trata de problemas complejos. [11]

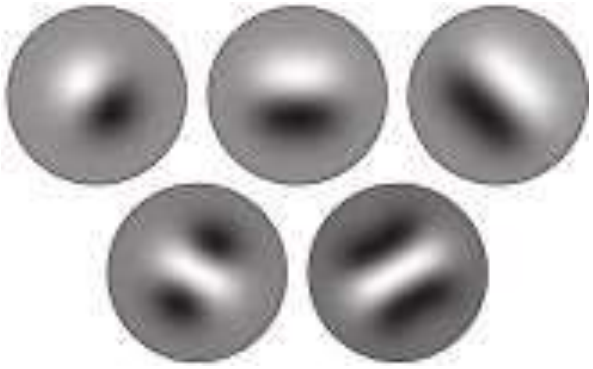


Figura 34. Filtros dirigibles [12]

3. Aprendiendo la función de clasificación con Adaboost

Dado un conjunto de imágenes de entrenamiento etiquetadas (positivas o negativas), Adaboost se usa para:

- seleccione un pequeño conjunto de características
- y entrenar al clasificador

Dado que se supone que la mayoría de las características entre los 160.000 son bastante irrelevantes, el algoritmo de aprendizaje débil alrededor del cual construimos un modelo de impulso está diseñado para seleccionar la característica de un solo rectángulo que divide los mejores ejemplos negativos y positivos. [11]

4. Clasificador en cascada

Existe un problema importante. En una imagen, la mayor parte de la imagen es una región sin rostro. Dar la misma importancia a cada región de la imagen no tiene sentido, ya que deberíamos enfocarnos principalmente en las regiones que tienen más probabilidades de contener una imagen. Viola y Jones lograron una mayor tasa de detección al tiempo que reducían el tiempo de cálculo utilizando clasificadores en cascada. [11]

La idea clave es rechazar las subventanas que no contienen caras mientras se identifican las regiones que sí las contienen. Dado que la tarea es identificar correctamente el rostro, queremos minimizar la tasa de falsos negativos, es decir, las subventanas que contienen un rostro y no han sido identificadas como tales. [11]

Se aplica una serie de clasificadores a cada subventanas. Estos clasificadores son árboles de decisión simples:

- si el primer clasificador es positivo, pasamos al segundo
- si el segundo clasificador es positivo, pasamos al tercero

Cualquier resultado negativo en algún momento conduce a un rechazo de la subventanas por contener potencialmente una cara. El clasificador inicial elimina la mayoría de los ejemplos negativos a un bajo costo computacional, y los siguientes clasificadores eliminan ejemplos negativos adicionales, pero requieren más esfuerzo computacional. [11]

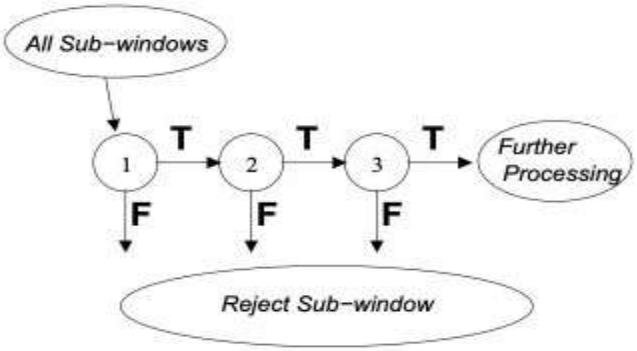


Figura 35. Clasificadores en Cascada [12]

Los clasificadores se entrenan usando Adaboost y ajustando el umbral para minimizar la tasa de falsos. Al entrenar dicho modelo, las variables son las siguientes:

- El número de etapas del clasificador
- La cantidad de funciones en cada etapa
- El umbral de cada etapa

Reconocimiento facial: Histograma de gradientes orientados (HOG) en Dlib

La idea detrás de HOG es extraer características en un vector y alimentarlo en un algoritmo de clasificación como una máquina de vectores de soporte, por ejemplo, que evaluará si una cara está presente en una región o no. [12]

Las características extraídas son la distribución (histogramas) de direcciones de gradientes (gradientes orientados) de la imagen. Los degradados suelen ser grandes alrededor de los bordes y esquinas y nos permiten detectar esas regiones. [12]

En el artículo original, el proceso se implementó para la detección del cuerpo humano, y la cadena de detección fue la siguiente:

1. **Pre procesamiento**

En primer lugar, las imágenes de entrada deben tener el mismo tamaño. Los parches que aplicaremos requieren una relación de aspecto de 1: 2, por lo que las dimensiones de las imágenes de entrada pueden ser 64x128o, 100x200. [12]

2. **Calcular las imágenes de degradado**

El primer paso es calcular los gradientes horizontales y verticales de la imagen, aplicando los siguientes núcleos:

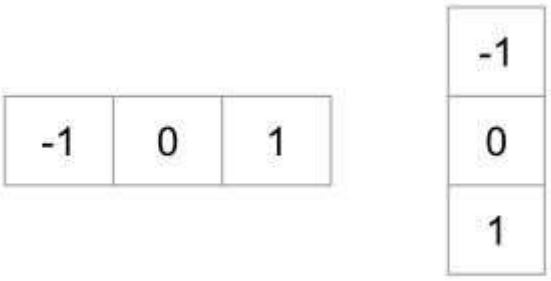


Figura 36. Gradientes horizontales y verticales [12]

El degradado de una imagen normalmente elimina información no esencial.

3. Calcule el HOG

La imagen se divide en celdas de 8x8 para ofrecer una representación compacta y hacer que HOG sea más resistente al ruido. Luego, calculamos un HOG para cada una de esas celdas. [12]

Para estimar la dirección de un gradiente dentro de una región, simplemente construimos un histograma entre los 64 valores de las direcciones del gradiente (8x8) y su magnitud (otros 64 valores) dentro de cada región. Las categorías del histograma corresponden a los ángulos del gradiente, de 0 a 180 °. Hay 9 categorías en total: 0 °, 20 °, 40 °… 160 °. [12]

Dado como información:

- Dirección del gradiente
- Magnitud del gradiente

Cuando construimos el HOG, hay 3 subcasos:

- El ángulo es menor de 160 ° y no está a medio camino entre 2 clases. En tal caso, el ángulo se agregará en la categoría correcta del HOG
- El ángulo es menor de 160 ° y exactamente entre 2 clases. En tal caso, consideramos una contribución igual a las 2 clases más cercanas y dividimos la magnitud en 2

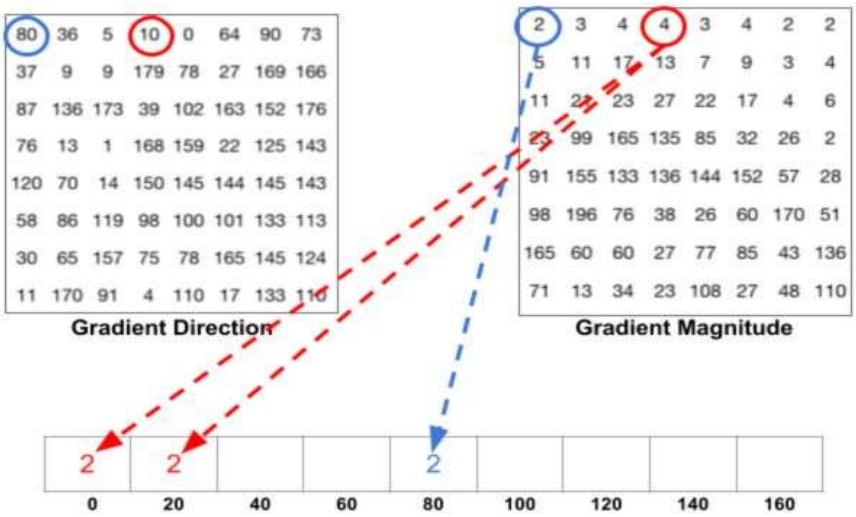


Figura 37. Histograma de Gradientes

- el ángulo es superior a 160 °. En tal caso, consideramos que el píxel contribuyó proporcionalmente a 160 ° y a 0 °.

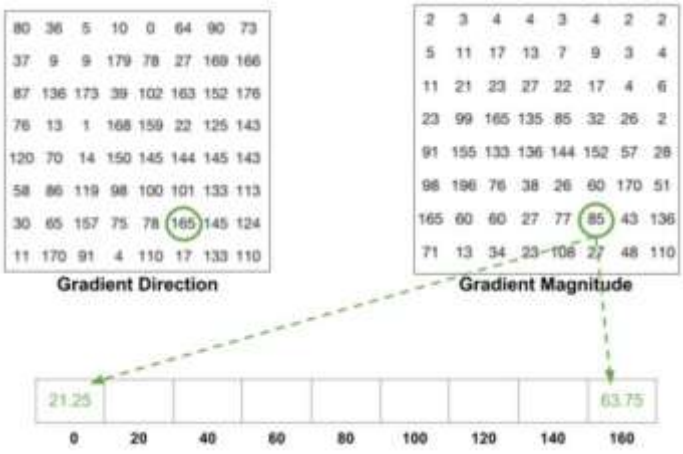


Figura 38.Histograma de Gradiente

El HOG se ve así para cada celda de 8x8:

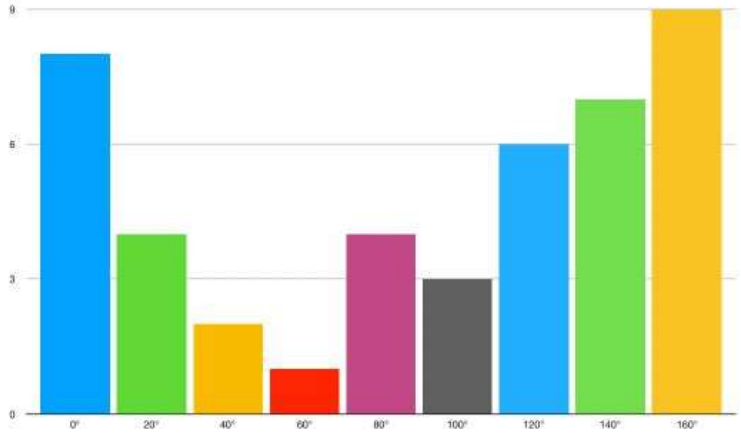


Figura 39. Histograma de gradiente

4. Normalización de bloque

Finalmente, se puede aplicar un bloque de 16x16 para normalizar la imagen y hacerla invariable a la iluminación, por ejemplo. Esto se logra simplemente dividiendo cada valor del HOG de tamaño 8x8 por la norma L2 del HOG del bloque 16x16 que lo contiene, que de hecho es un simple vector de longitud 9*4 = 36. [12]

5. Normalización de bloque

Finalmente, todos los vectores de 36x1 se concatenan en un vector grande. Tenemos nuestro vector de características, en el que podemos entrenar un clasificador SVM suave (C = 0.01). [12]

1.4 Objetivos

1.4.1 Objetivo general

Implementar un Sistema de Smart Home aplicando IoT y Blockchain

1.4.2 Objetivos específicos

- Investigar sobre la tecnología Blockchain para la gestión de la seguridad de los datos y la descentralización de la información
- Diseñar un Sistema de Smart Home aplicando IoT y Blockchain
- Evaluar la integridad y confidencialidad de la información de acceso en la integración de Blockchain con IoT.

CAPÍTULO II.

METODOLOGÍA.

2.1 Materiales

2.1.1 Selección de los componentes de hardware y software

2.1.1.1 Sección del Sistema de bloqueo de puerta en Smart Home

El sistema de bloqueo de puerta consta de dos partes. En la primera parte para el reconocimiento facial utiliza un módulo de cámara de Omnivision 5647 de 5Mpx incorporada en una Raspberry Pi óptimo para el algoritmo propuesto en el sistema bloqueo de puerta. En la segunda parte del sistema consta de una cerradura de solenoide de 12v conectada a un módulo relé y un regulador de voltaje LM2596 para el bloqueo o desbloque de la puerta

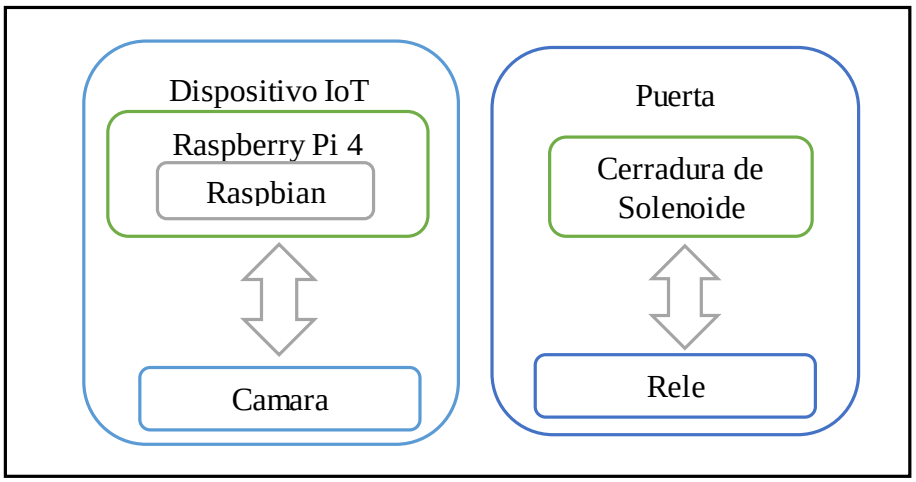


Figura 40. Diagrama de Sistema de bloqueo de puerta en la Smart Home.
Elaborado por el Investigador

2.1.1.2 Sección Red Blockchain

En la red Blockchain se realizó en una PC con requerimientos como: 16 GB de RAM, con procesador Core i7-7700HQ 7th Gen y Tarjeta de video dedicada Nvidia Ge Forcé GTX 1050 4GB GDDR5. La cadena de bloques privada en este sistema propuesto almacenara todas las transacciones de acceso con cerradura de puerta.

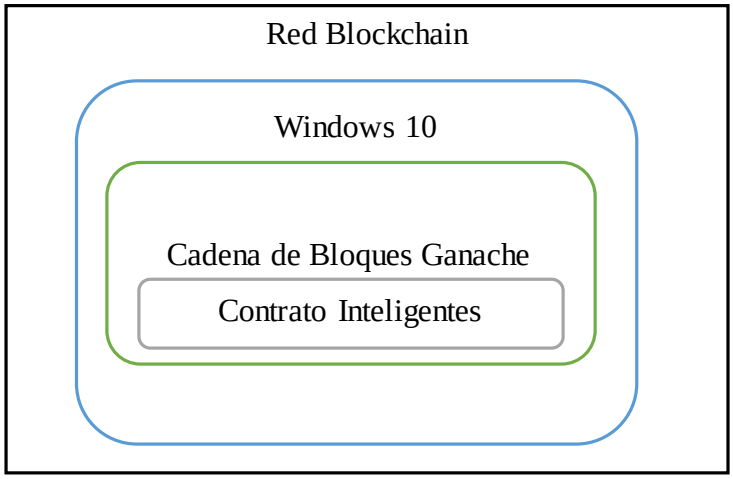


Figura 41. Red Blockchain

Elaborado por el Investigador

2.2 Métodos

2.2.1 Modalidad de la Investigación

El presente proyecto fue de investigación y desarrollo (I+D), puesto que se aplicarán los conocimientos teóricos indagados para el desarrollo de este. Haciendo uso de la investigación bibliográfica se indagará en libros, papers, revistas científicas y bases de datos para justificar los argumentos establecidos y brindar soporte al desarrollo de técnicas específicas para la elaboración del proyecto.

Los antecedentes investigativos permitirán buscar indicios y relaciones respecto al tema, los mismos, permitirán extraer información trascendental para apoyar el desarrollo del proyecto (I. Experimental).

2.2.2 Recolección de Información

A través del internet y buscadores conocidos se recolectará información de papers, revistas científicas, libros, tesis y bases de datos, también se contará con la información brindada por el tutor.

2.2.3 Procesamiento y Análisis de Datos

Para el procesamiento y análisis de datos se llevará a cabo una serie de pasos descritos a continuación:

- Establecer prioridades en la búsqueda de información.
  - Decidir qué información buscar
  - Decidir cómo buscar la información
- Recopilar los datos de bases de datos existentes y de las fuentes disponibles

- Determinar un sistema de almacenamiento y nombramiento de archivos.
- Analizar la información recopilada a través de una lectura comprensiva.
- Interpretar los resultados

2.2.4 Desarrollo del proyecto

1. Identificación de los componentes que conforma una tecnología Blockchain
2. Análisis de la integración de dispositivos IoT en soluciones gestionadas por redes Blockchain.
3. Selección del hardware y software para el desarrollo del sistema.
4. Diseño del Sistema de Smart Home aplicando IoT y Blockchain
5. Construcción de los nodos con tecnología Blockchain para la gestión de información del sistema.
6. Implementación del Sistema de Smart Home aplicando IoT y Blockchain.
7. Evaluación de la integridad y confidencialidad de la información de acceso en la integración de Blockchain con IoT.
8. Análisis del resultado obtenido en la integración de Blockchain con IoT
9. Elaboración del trabajo escrito

CAPÍTULO III.

3. RESULTADOS Y DISCUSIÓN

3.1 Análisis y discusión de los resultados.

3.1.1 Desarrollo de la propuesta

3.1.1.1 Descripción del Sistema de Smart Home aplicando IoT y Blockchain

Para el presente proyecto se propone un Sistema de Smart Home aplicando IoT Blockchain. La arquitectura del prototipo de sistema se muestra en la Figura 42, basado en la arquitectura IoT-Blockchain propuesta en la sección 1.3.7

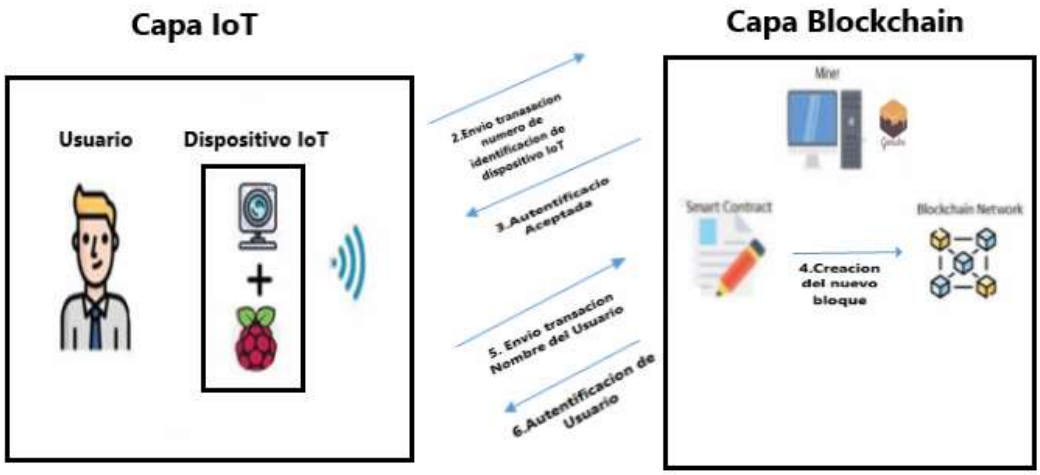


Figura 42. Arquitectura IoT Blockchain del prototipo a implementar

Elaborado por el Investigador

Capa IoT

Esta capa está compuesta de 3 procesos realizados en un dispositivo IoT que interactuar con la cadena de bloques Ethereum. En el primer proceso realiza la autentificación del dispositivo IoT en el contrato inteligente implementado en la cadena de bloques Ethereum, El segundo proceso utiliza la cámara para detección y reconocimiento de la persona que ingresan al hogar, ya sea que este registrado o no y El tercer proceso activa la cerradura después de la autentificación del usuario por medio del contrato Inteligente implementado en la cadena de bloques Ethereum. En

unos segundos, después de que se abra la puerta, la puerta se bloqueará automáticamente de nuevo.

Capa Blockchain

Esta capa consta de una cadena de bloques Ethereum en Ganache en el cual se implementó un contrato inteligente para interactuar con los datos enviados por el dispositivo IoT. En la cadena de bloques propuesto almacenara todas las transacciones de acceso con la cerradura inteligente y realizara la autentificación del dispositivo IoT y la persona que ingresa al hogar.

3.1.1.2 Selección de los elementos para la implementación del sistema Smart Home con IoT y Blockchain

Para implementar el modelo de puerta inteligente, necesitamos una lista de materiales que se menciona brevemente a continuación:

Tabla 1. Hardware para el dispositio IoT

Categorías	Raspberry pi 4	Nvidia Jetson Nano	Esp32-Cam
Tipo	microordenadores	microordenadores	Microcontrolador
UPC	ARM de cuatro núcleos Cortex-A72 de 64 bits	ARM 64 bits	32 bits
Velocidad de reloj	1.5 Ghz	1,42 Ghz	160 Mhz
GPU	Broadcom Video Core VI	NVIDIA Maxwell	NO
Memoria	LPDDR4 de 4 GB	LPDDR4 de 4 GB	SRAM
Redes	Gigabit Ethernet / Wifi	Gigabit Ethernet / M.2 Key E	Wifi
Cámara	Puerto MIPI CSI	Puerto MIPI CSI	NO
Almacenamiento	Micro-SD	Micro-SD	NO
Precio	64	99,99	10
Disponibilidad en el país	Si	No	No

Para la selección del hardware del dispositivo IoT que más se apegue a los requerimientos técnicos y eléctricos para su instalación en un Hogar, se realizó una comparación entre tres diferentes equipos donde se compara tipo de cpu, gpu y etc. La Raspberry pi 4 las especificaciones, el precio, y la disponibilidad en el mercado determinaron que esta placa se adapta a las necesidades del proyecto

Tabla 2. Módulo Cámaras para Rabpberry Pi

Módulo Cámaras	Modelo	Sensor	Pixel	Resolución	Cuadros por Segundos	Precio
SONY	CMT-8MP-IMX219-M366	CMOS SONY IMX219 DE 1/4	8 MP	3296 (H) x 2480 (V)	90 fps	47,99
Omnivision	CMT-5MP-RP-OV5647-X011	Omnivision OV5647 de 1/4	5 Mp	2592 x 1944	30 fps	25,00
Omnivison	CMT-5MP-OV5647-H020	Omnivision OV5647 de 1/4	5 Mp	2592x1944	15 fps	34,99
Omnivision	CMT-5MP-RP-OV5647-X010	Omnivision OV4647 de 1/4	5 MP	2592 x 1944	30 fps	15,99

Para la cámara se optó por el modelo CMT-5MP-RP-OV5647-X010 de la marca Omnivision de 5 megapíxeles con un precio justo y fácil de configurar. La calidad de imagen con poca luz es aceptable y el consumo de energía es menor.

Tabla 3. Disipadores de calor para rasberry pi

Modelo	Método de Refrigeración	Potencia Nominal	Disponibilidad en el mercado	Precio
GeeekPi	refrigerador colorido de la torre de hielo 52Pi se inspira en los conjuntos de disipador térmico y ventilador de estilo torre (HSF) de una sola pila	0,4 W 5 V, 0,08 A	No	19,99
Fan Shim	El ventilador se puede controlar en el software para que pueda encenderlo a cierta temperatura de la CPU	1,3 v	No	21,20
Normal	El ventilador se conecta a 5v y funciona sin interrupción.	5v	Si	3,90

Para la refrigeración se optó por un ventilador pequeño de 5v con gran capacidad de disipar el valor con un bajo consumo de energía y disponible en mercado aun precio justo

Tabla 4. Lenguaje de programación para el reonocmiento facial

JavaScript	Python
no tiene el concepto de mutable e inmutable	tipos de datos mutables e inmutables.
Codificación UTF-16	Codificación ASCII
números de punto flotante.	tipos numéricos diferentes como int, decimal de coma fija, etc.
Usa llaves	Uso sangría
tiene menos módulos como date, math, regexp, JSON.	incluye una amplia gama de módulos.
Los objetos tienen propiedades que pueden estar compuestas por atributos subyacentes que le permiten definir una propiedad.	las funciones getter y setter se utilizan para definir un atributo.
utiliza un modelo de herencia basado en prototipos.	usa el modelo de herencia basado en clases.
una buena opción para el desarrollo móvil.	no es una buena opción para desarrollar aplicaciones móviles.
JavaScript le ayuda a crear un sitio web o una aplicación nativa.	tareas relacionadas con el análisis de datos, el aprendizaje automático y las operaciones intensivas en matemáticas.

Para la elección del software para el sistema de reconocimiento facial se optó por Python ya que soporta tareas de machine learning y se trabaja con cadenas de bloques Ethereum por medio de la biblioteca web3.py Blockchain

Tabla 5. Hardware de la red Blockchain

Marca	**Dell**
Tipo de portátil	Laptop para juegos estándar no convertible
Tamaño de pantalla	15,6 pulgadas
Resolución de la pantalla	1920x1080 píxeles Full HD
Procesador (CPU)	Core i7-7700HQ 7th Gen
Gráficos (GPU)	Tarjeta de video dedicada Nvidia GeForce GTX 1050 4GB GDDR5
RAM	16GB
Almacenamiento de datos	HDD de 1000GB (1TB) + SSD de 128GB
Entrada de teclado	Teclado retroiluminado con teclado numérico dedicado
Altavoces y audio	Micrófono combinado y conector de audio para auriculares
Cámara	Cámara web frontal
Wi-Fi inalámbrico	802.11AC
Puerto de red Ethernet	si
Bluetooth	si
Puertos USB	3 x USB 3.0
Puertos de video	1 x HDMI
Lector de tarjetas multimedia	Lector de tarjetas SD
Batería	6 celdas 74 WHr

Para la instalación de la cadena de bloques Ethereum se optó por un ordenador portátil de marca Dell con procesador i7 de séptima generación y una gpu de Nvidia GeForce GTX 1050 4GB óptimo para ejecución de Blockchain

Tabla 6. Software para la Red Blockchain

Plataforma	Ethereum	Hyperledger Fabric	R3 Corda
Caso de uso	aplicaciones generalizadas y se utiliza principalmente par operacines P2P YB2C	Operaciones B2B,utilizada principalmente en empresas	plataforma de registros distribuidos para las necesidades de la industria finaciaera
Gobernanza	desarrolladores	Fundacion linux a cargo	Empresa R3
Modo de operacion	Blockchain publica	Blockchain privado	Blockchain privado
Consensos	Proof-of-take para la toma de decisiones	No todos los nodos de una red deben participar en el proceso de conseso	Solo las partes involucradas en la transacción participan en la toma de decisión
Contratos Inteligentes	Solidity	Golang	Kotlin
Moneda	Ether	No tiene	No tiene

Para el software de la tecnología Blockchain se optó por Ethereum el cual permite configurar cadena de bloques privadas o públicas y puede trabajar con una gran variedad de lenguajes de programación como JavaScript, Python, Go, etc.

Tabla 7. Lenguajes de programación para Contratos Inteligentes

Idioma	**Paradigma**	**Turing-complete**	**Metas**
Solidez	Orientado a contratos	Si	Realizar contratos inteligentes en la máquina virtual Ethereum
Bamboo	Procedimiento de	Si	Proporcionar contratos polimórficos que se pueden ejecutar en un orden predefinido con auditabilidad mejorada
Obssidian	Orientada a objetos	-	facilita a los programadores escribir programas correctos mientras el aprovechamiento de un tipo
			Sistema para proporcionar fuertes garantías de seguridad
Vyper	Procedimientos	No	proporcionar más segura, simple y contratos inteligentes auditables en EVM
Flint	Procedimiento	No	Escribir robustos contratos inteligentes en Etereum
Babbage	Mecánica	-	Exponer cómo diferentes partes interactúan y en forma junto
Pyramid	funcional	y	Esquema programación objetivos de idioma en EVM
DAML	funcional	No	las aplicaciones componiblesCrearen un resumen de DA Ledger Modelo
Yul	Procedimiento	y	Diseñado para ser un denominador común utilizable de varias diffErent backends en
			Etereum
IELE	basado en rigister	Y	Diseñado para proporcionar seguridad en contratos inteligentes, verificación formal, legibilidad humana, y portabilidad

Para el lenguaje de programación de contrato inteligentes se optó por Vyper debido a que proporciona contratos seguros, simple y fáciles de auditar por otros programadores y con mayor restricción a diferencia de Solidity

3.1.1.3 Direccionamiento y topología del prototipo

Direccionamiento

Para el direccionamiento se considera una sola red de área local inalámbrica para el dispositivo IoT y la red Blockchain.

Tabla 8. Direccionamiento del dispositivo IoT y Red Blockchain

Dispositivo	Direccionamiento	Mascara
Cerradura inteligente	192.168.0.106	/24
Red Blockchain	192.168.104	/24

Topología

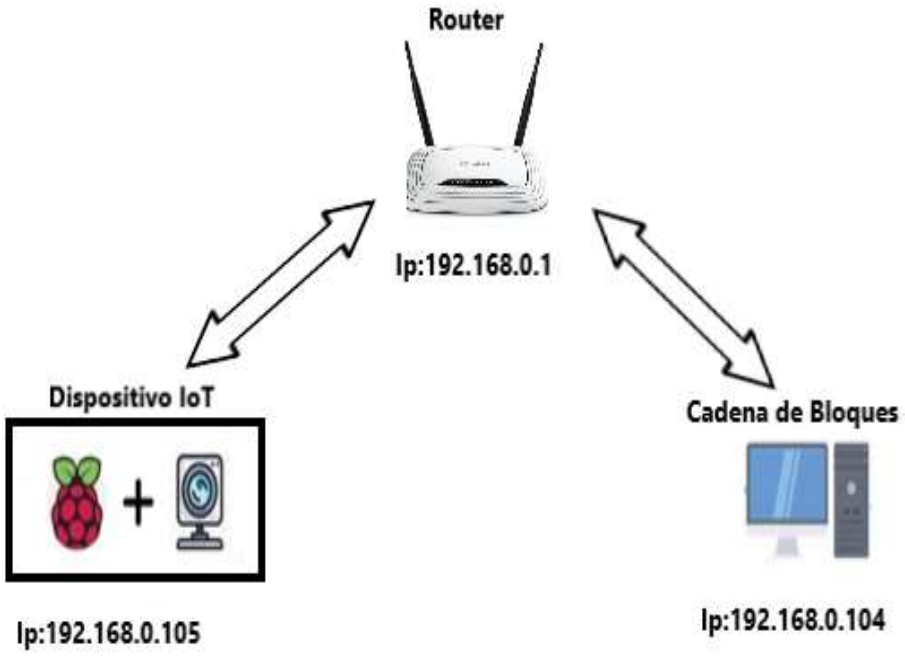


Figura 43. Topología del Sistema de Smart Home aplicando IoT y Blockchain

Elaborado por el Investigador

3.1.1.4 Diseño e Implementación sistema de bloque de puerta con reconocimiento facial en un Smart Home

Diseño del software para el sistema de reconocimiento facial con Raspberry Pi

La figura 44 muestra el diagrama de flujo del sistema de reconocimiento facial para el bloqueo de puerta en una Smart Home. Primero, la cámara captura la imagen del rostro. La función de detección de rostros localiza y segmentaria sólo la región del rostro. Segundo, la imagen de la cámara se introduce en la función de reconocimiento fácil y devuelve el nombre del usuario. Tercero, ingresa a la función de envió de transacción a la cadena de bloques Ethereum el cual enviará un mensaje de respuesta de la transacción si es un usuario autentificado, el sistema desbloquea la puerta activando la cerradura magnética y apagado led indicador, después de 10 segundos, el sistema volverá a bloquear, revisar Anexo E.

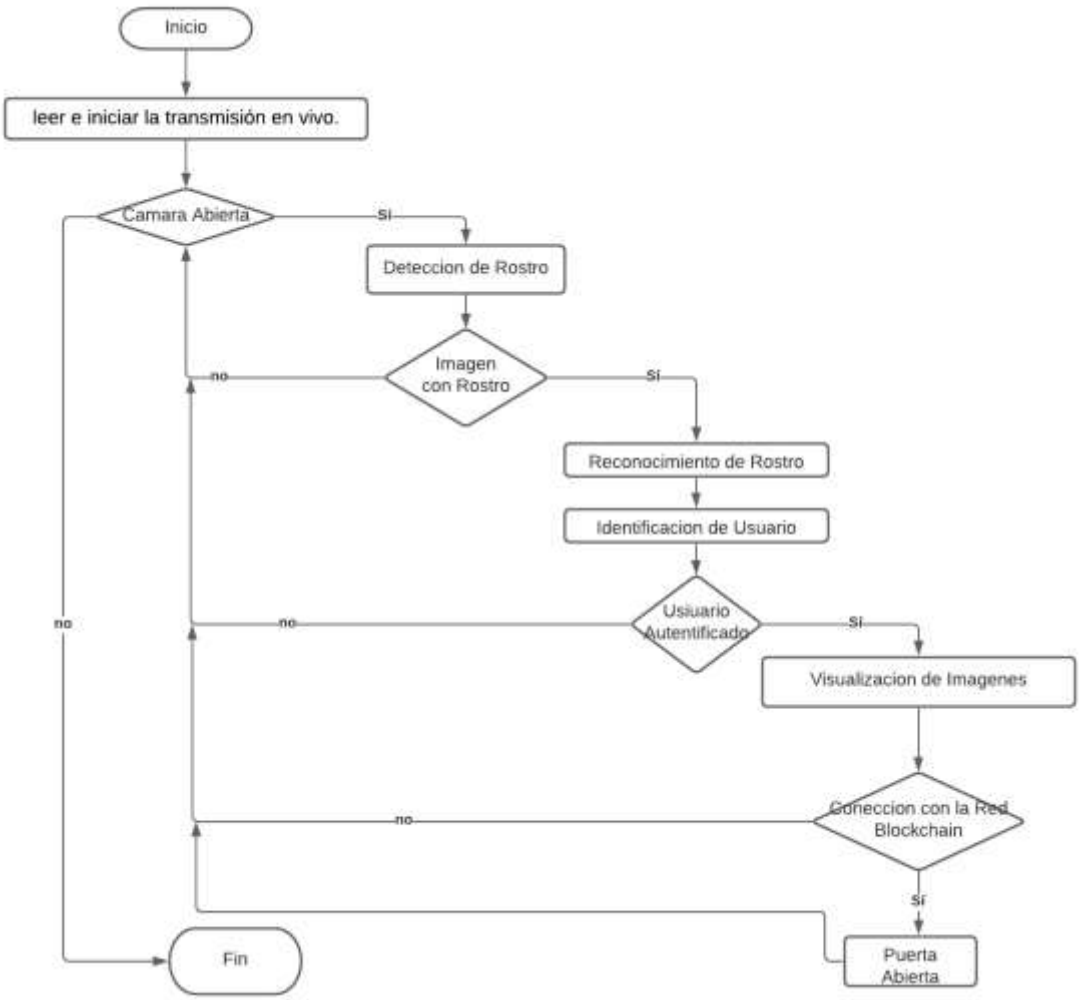


Figura 44.Diagrama de flujo del Sistema de Smart Home aplicando IoT y Blockchain

Elaborado por el Investigador

Algoritmo utilizado para el sistema de reconocimiento facial

La detección de rostros es un proceso para encontrar el área del rostro en el fotograma. En el proyecto se utilizó Algoritmo clasificador en cascada Haar para detectar rostros. Los clasificadores en cascada de Haar tienen la ventaja de una rapida detección en comparación con otros clasificadores lo que permite tener resultados en tiempo real óptimo para dispositivos IoT.

Visualización de características de HAAR

Figura 45. Características del algoritmo HARR CASCADE

Elaborado del Investigador

Algoritmo para el reconocimiento facial

El sistema de reconocimiento facial está basado en el algoritmo HOG por que los histogramas de gradientes orientados han demostrado ser un descriptor eficaz para el reconocimiento de objetos en general y el reconocimiento facial en particular. Los algoritmos de reconocimiento facial de dlib permiten al usuario implementar fácilmente la detección facial, el reconocimiento facial y el seguimiento facial en tiempo real.

Para lo cual el clasificador Haar Cascade detecta una sola cara, la cara se recortará fuera de la escena. Luego el algoritmo de reconocimiento facial que se ha entrenado reconocerá la cara recortada y devolverá una lista de valores verdaderos y falsos que

indican qué codificaciones de caras conocidas coinciden con la codificación de caras detectadas para verificar al usuario que intenta ingresar al hogar.

Visualización de características de HOG

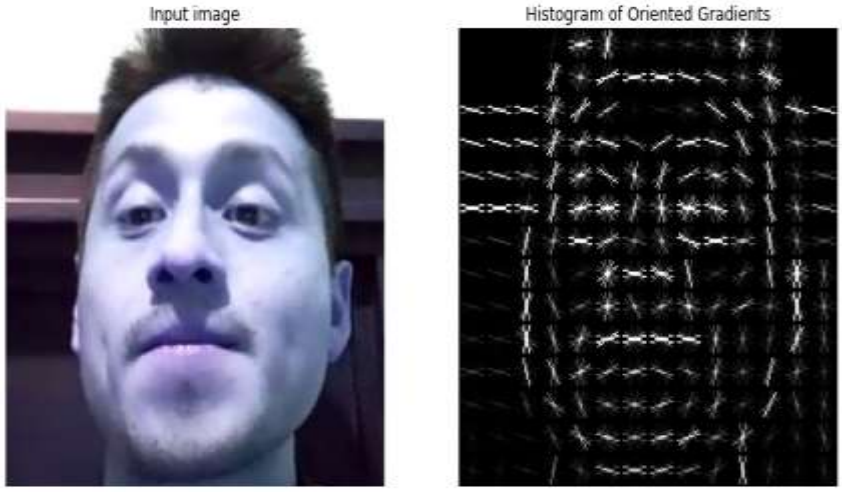


Figura 46. Características del HOG del rostro

Elaborado por el Investigador

Implementación del sistema de reconocimiento facial con Raspberry Pi

1. Fase de instalación de dependencias

Instalación de las dependencias para el sistema de reconocimiento facial

Para la implementación del sistema de reconocimiento facial con Raspberry Pi es necesario instalar Raspbian como sistema operativo, para lo cual revisar Anexo A

Instalación de dependencias para el reconocimiento facial

Para el reconocimiento facial se instalará Opencv, face_recognition, imutils y demás dependencias para preparar el dispositivo IoT para el aprendizaje automático y el reconocimiento facial, revisar Anexo B

- **Opencv** es una biblioteca de software de código abierto para procesar imágenes y videos en tiempo real con capacidades de aprendizaje automático.
- **face_recognition** es un paquete de Python para calcular el cuadro delimitador alrededor de cada cara, calcular la incrustación facial y comparar caras en el conjunto de datos de codificación.
- **Imutils** es una serie de funciones de conveniencia para acelerar la computación Opencv en Raspberry Pi.

2. Fase de creación de la base de datos de los usuarios

Para la creación de una base de datos para el modelo de reconocimiento facial se realizó mediante la toma muestra cada usuario por medio de un clasificador en cascada HAAR con el script tk_captura.py revisar Anexo C.

Figura 47. Interfaz gráfica para captura de rostros

Elaborado por el Investigador

En la figura 47 se muestra una interfaz gráfica que captura una colección de 30 rostros y los almacena en una base de datos, todas las imágenes tienen un tamaño 224x224 y cada una se guarda en una carpeta especifica con el nombre del usuario correspondiente.

Fase de entrenamiento del modelo para el reconocimiento facial en el dispositivo IoT

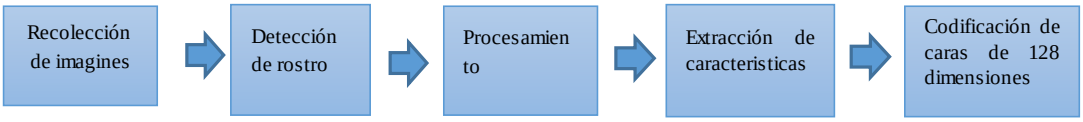


Figura 48. Procesamiento de entrenamiento

Elaborado por el Investigador

En esta fase se entrena el modelo de reconocimiento facial para lo cual se ejecuta el script tk_capture.py, revisar el Anexo C. El proceso consiste en extraer de cada imagen almacenada una codificación de 128 dimensione y estos datos en conjunto con el nombre de cada usuario será serializado y guardado en el archivo encodings.pickle

Figura 49. Script para entrenamiento del reconocimiento facial

Elaborado por el Investigador

En la figura 49 se muestra el procesamiento de cada imagen que al final se creara un archivo encodings.pickle que contiene los criterios para identificar los rostros de todos los usuarios

3. Fase reconocimiento facial

Esta fase se ocupa de la detección, reconocimiento de los usuarios que ingresar al Smart Home y la envió de la información a la red Blockchain para autentificar y guarda los datos., revisar Anexo D

Descripción de las funciones

Tabla 9. Funciones utilizados para el reconocimiento facial

Función	Descripción
detec(img)	Esta rutina toma como parámetro la imagen capturada por la cámara, en donde el algoritmo Haar Cascades detecta el rostro en la imagen ingresada y devuelve una tupla con la coordenadas x,y y el ancho y alto del rostro , luego convertimos en una lista de tuplas que contiene los punto superior derecha, inferior, izquierda y retorna la variable boxes.
reconocimiento(rgb,boxes)	En esta función toma dos para la imagen en el formato RGB, las coordenadas que retornaron de la función de detección, donde se codifica la nueva imagen y comparan con la todos los usuario codificados en el archivo encodings.pickle y devuelve una lista con los nombres de los rostros detectados en la imagen ingresada
transaccion(nombre)	Esta función toma como parámetro el nombre del usuario detectado se crea una transacción con el contrato inteligente Smart_home.vy compilado en la red Blockchain, revisar el anexo F. en la red Blockchain procesa la transacción y regresa la autorización del usuario, la función espera que transacción se reciba en la red Blockchain y devuelve un string que puede tener dos posibles valores Autorizado o No Autorizado.

3.1.1.4.3Diseño del hardware para el sistema de reconocimiento facial

El diagrama de bloques propuesto del sistema de reconocimiento facial se muestra en la Figura 44. La Raspberry Pi está conectada al módulo de la cámara de 5MP y a un circuito de relé que proporciona una conexión para la cerradura de solenoide. Por último, la potencia requerida es 3A, por lo que se requiere una fuente de alimentación compacta.

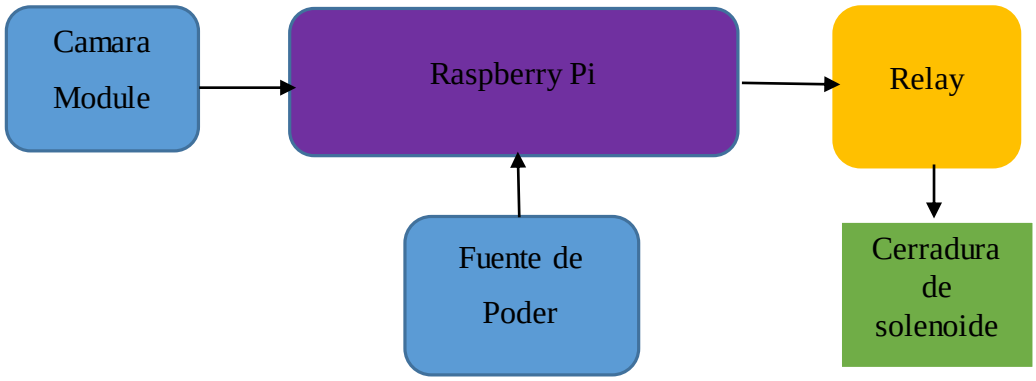


Figura 50. Diagrama de bloques del hardware propuesto

Elaborado por el Investigador

Diseño 3D de la estructura del dispositivo IoT

El diseño de la estructura del dispositivo IoT se llevó a cabo en Tinkercad. El diseño proporciona un ajuste para los tres elementos principales del dispositivo IoT.

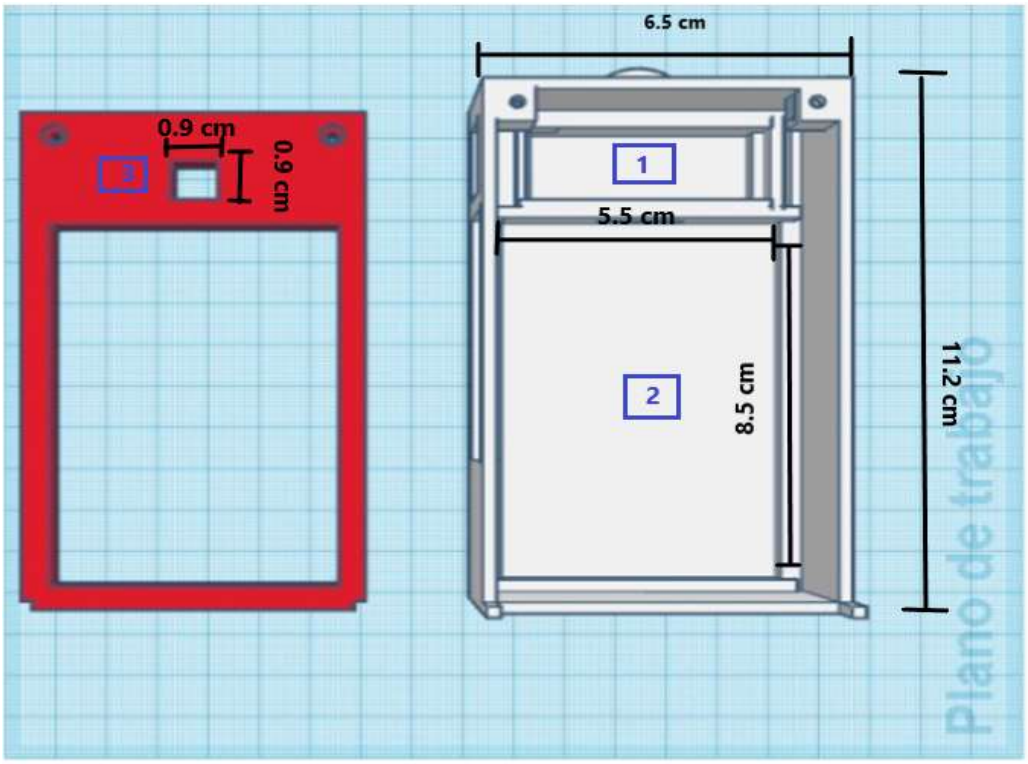


Figura 51. Diseño 3D de la estructura del dispositivo IoT

Elaborado por el Investigador

1. Abertura para el Relay
2. Abertura para la Raspberry Pi 4
3. Abertura para la cámara 5MP

En la figura 52 se observa las vistas de la carcasa que contendrá a la Raspberry pi 4, al módulo relé y la cámara de 5MP

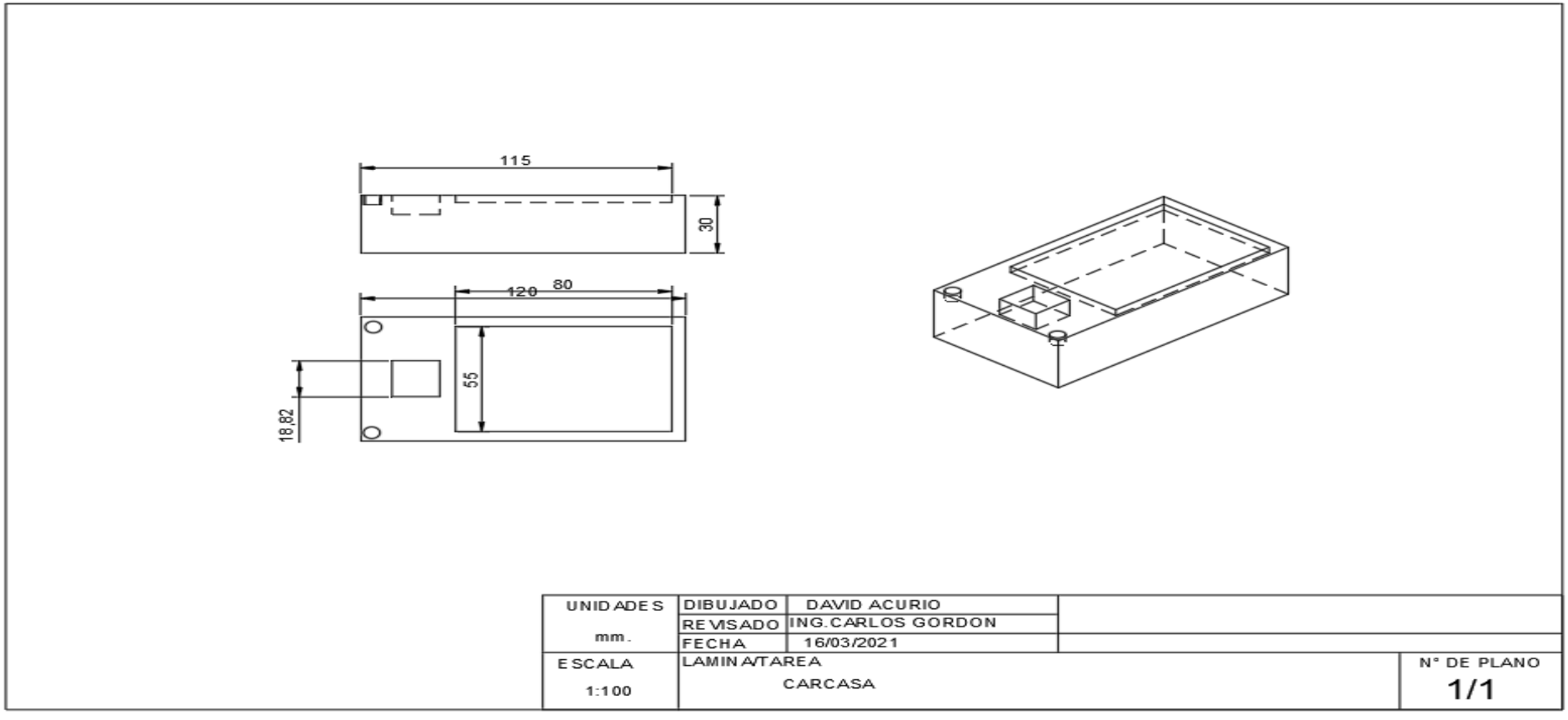


Figura 52. Vistas frontal y lateral de la carcasa del dispositivo IoT

Elaborado por el Investigador

3.1.1.4.4 Implementación del hardware del sistema de reconocimiento facial con Raspberry Pi

El prototipo completo del sistema de bloque de puerta con reconocimiento en una Smart Home se muestra en la figura 53. La caja contiene, el relé, led indicador y un ventilador. El prototipo se colocará en la puerta y se conectará a una cerradura magnética como se muestra en la Figura 54 que se desbloquea cuando el usuario autorizado accede al sistema. Si un personal no autorizado intenta acceder, la puerta permanecerá bloqueada y la caja se colocará a unos 1,55 m del suelo según el análisis de documentos de instalación de video porteros y la cerradura se colocará en la parte superior de la puerta. La fuente de alimentación se tomará de un enchufe de pared y estas son las únicas entradas y salidas de la caja

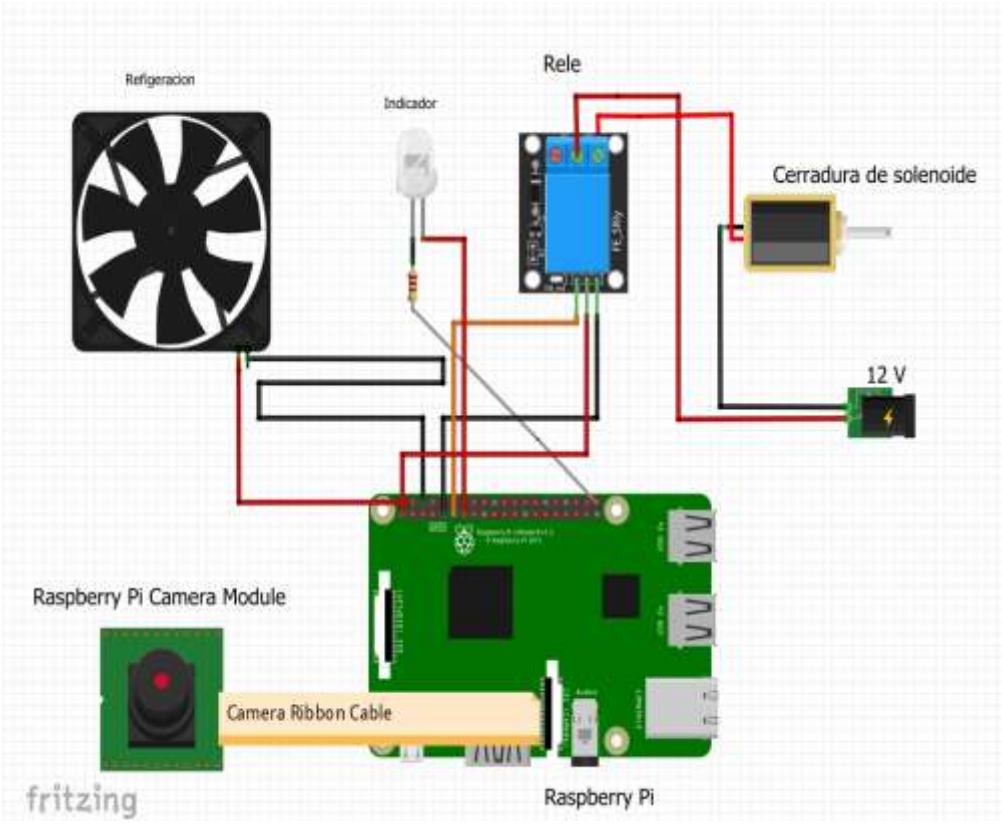


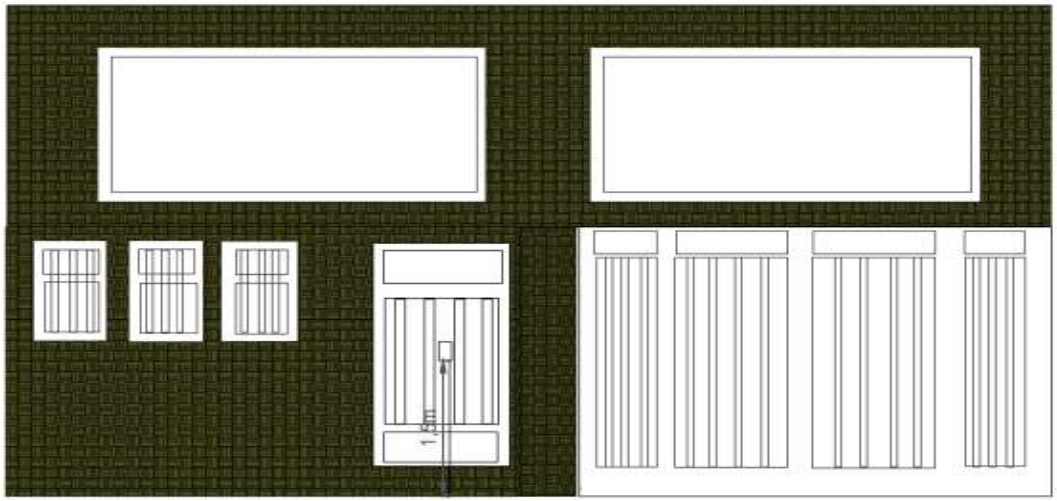
Figura 53. Diagrama de circuito

Elaborado por el Investigador

Figura 54. Implementación del Sistema de Smart Home y Blockchain

Elaborado por el Investigador

Para la ubicación del dispositivo IoT se analizó dos factores la altura del usuario y el rango de los usuarios es de 1.50-1.85 por lo que se optó por una altura 1,5 m tomado desde el nivel de la acera como se observa en la Figura 54



UNIDADES	DIBUJADO	David Acurio	
mm.	REVISADO	Ing.Carlos Gordon	
	FECHA	27/03/2021	
ESCALA	LAMINA/TAREA		N° DE PLANO
1:1	Instalacion del Dispositivo IoT		1/1

Figura 55. Ubicación del dispositivo IoT en el Hogar

Elaborado por el Investigador

3.1.1.5 Implementación de la Cadena de Bloques Ethereum

3.1.1.5.1 Instalación de un entorno de desarrollo Ethereum

Para la implementación de la red Blockchain se requiere instalar los paquetes necesarios para su ejecución

1. Instalación de Node.js

Node.js es un marco popular para desarrollar aplicaciones web, aplicaciones móviles y aplicaciones descentralizadas. Para instalar en la plataforma Windows, revisar anexo G.

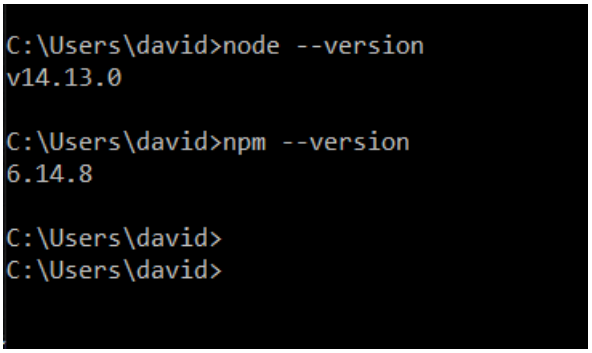


Figura 56. Instalación del framework Node.js

Elaborado por el Investigador

2. Instalación de Truffle

Para la compilación e implementación de contratos inteligentes en la red Blockchain utilizamos el marco truffle que nos permite el desarrollo de contratos inteligente en Solidity y Vyper.

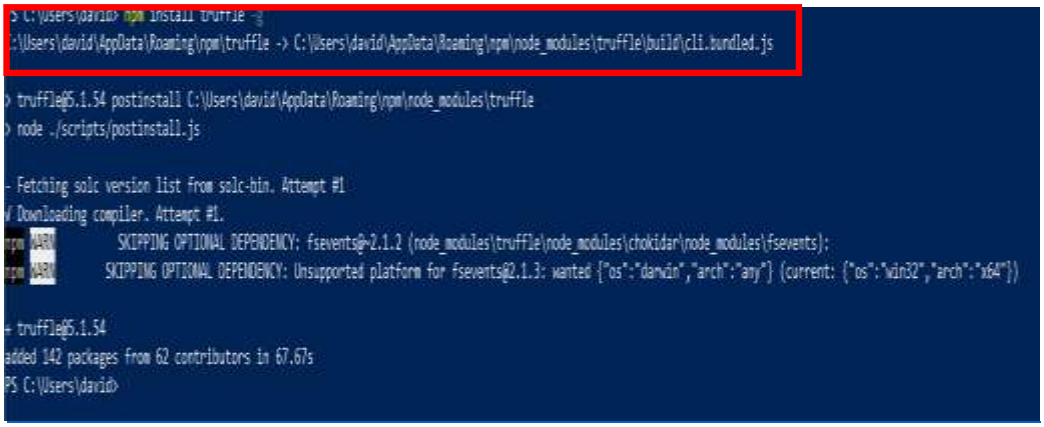


Figura 57. Instalación del framework Truffle

Elaborado por el Investigador

3. Instalación de Ganache

Ganache: es una cadena de bloques personal para el desarrollo rápido de aplicaciones distribuidas de Ethereum y Cordana.

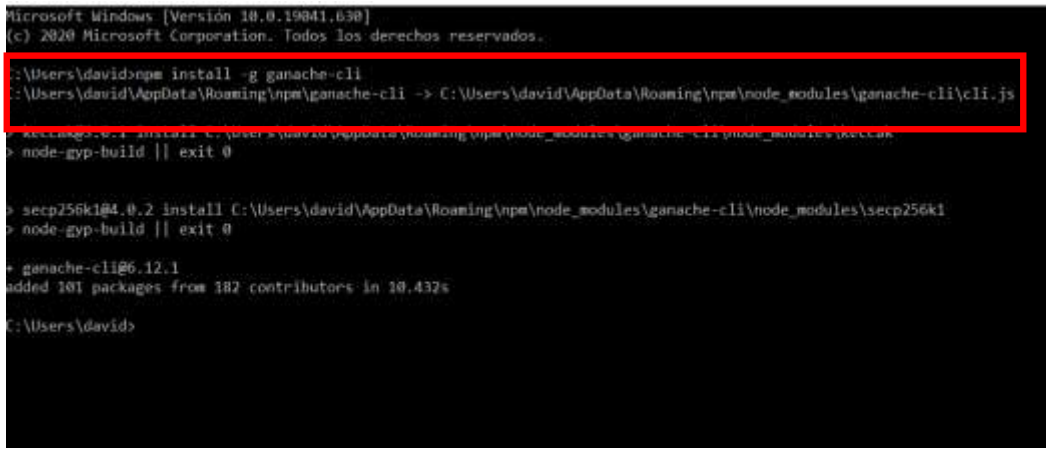


Figura 58. Instalación de la cadena de bloque Ganache

Elaborado por el Investigador

- **Creación del entorno de trabajo en ganache**

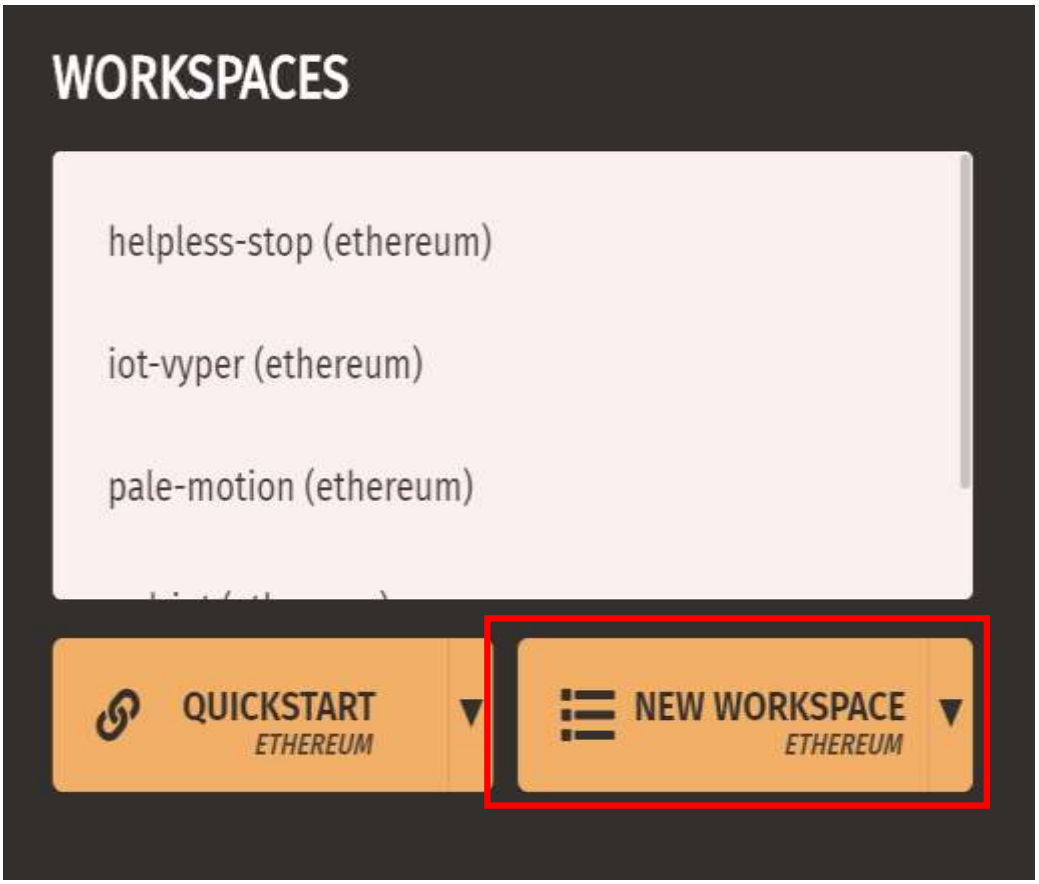


Figura 59. Entorno de Trabajo Ethereum

Elaborado por el Investigador

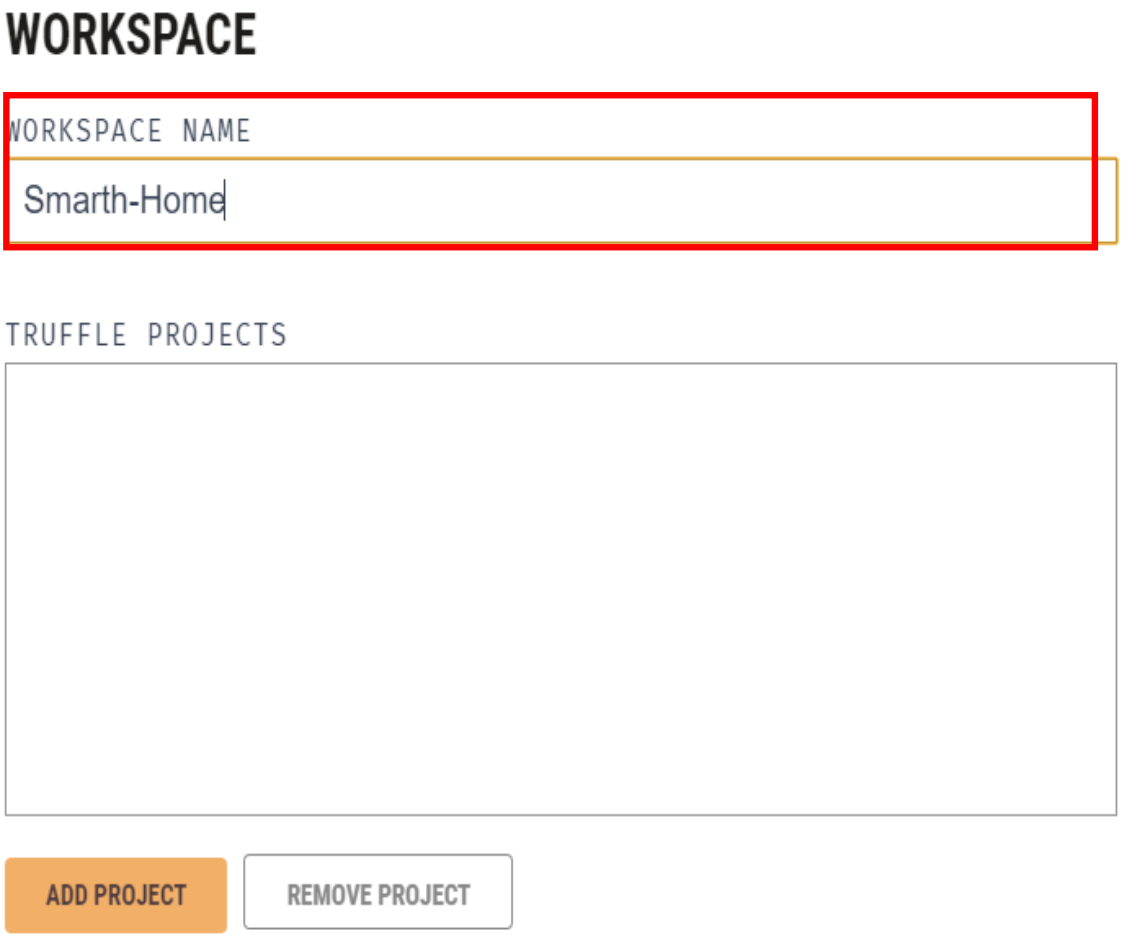


Figura 60. Espacio de Trabajo Smart-Home

Elaborado por el Investigador

4. Diseño del contrato Inteligente

Para la interacción entre el dispositivo IoT y la red Blockchain se necesita de un contrato inteligente el cual realizara en el lenguaje de programación Vyper, revisar Anexo F

Descripción de funciones

Tabla 10. Funciones del contrato Inteligente

Función	Descripción
def addDispositivo(id:int128)	Esta función añade al dispositivo IoT en un arreglo de 5 elementos
defautentificacion(_user: string[100])	Esta función devuelve la autentificación del Usuario por el dispositivo IoT a las Red Blockchain
def get_data()	Esta función retorna el Usuario enviado por el Dispositivo IoT
def get_auten()	Esta función retorna con la autentificación del usuario para ingresar al Hogar
def get_numdis()	Esta función devuelve el número de identificación de todos los dispositivos IoT conectados
def get_dispositivo(id:int128)	Esta función devuelve el número de identificación de un dispositivo IoT en especifico
def get_numdisp()	Esta función devuelve el número total de dispositivos Conectados

5. Implementación del contrato Inteligente en Ganache

1. Crear un nuevo directorio y luego lo inicializaremos con la herramienta de desarrollo Truffle

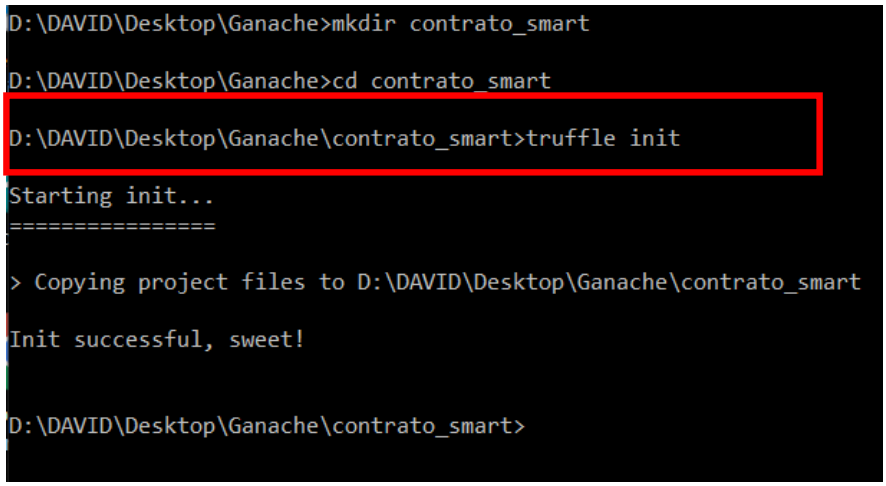


Figura 61. Creación del directorio

Elaborado por el Investigador

2. Guardar en la carpeta “contracts” el contrato Smart_home.vy

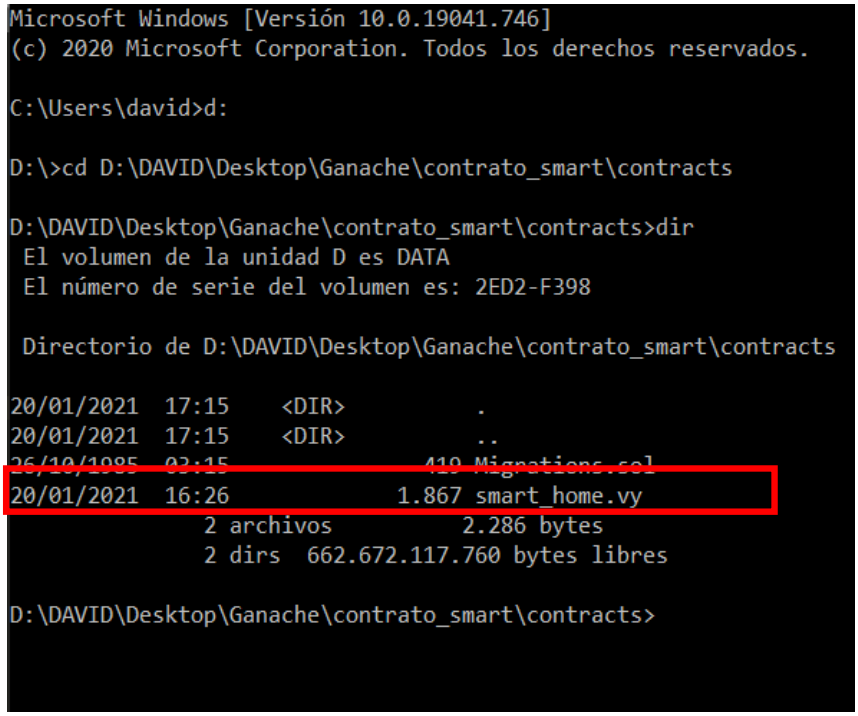


Figura 62. Carpeta contrats para guardar el archivo Smart_home.vy

Elaborado por el Investigador

3. Abrir la carpeta "migrations" y cree un nuevo archivo llamado "2_deploy_contracts.js". Las migraciones son simplemente scripts que nos ayudarán a implementar nuestros contratos en una cadena de bloques.

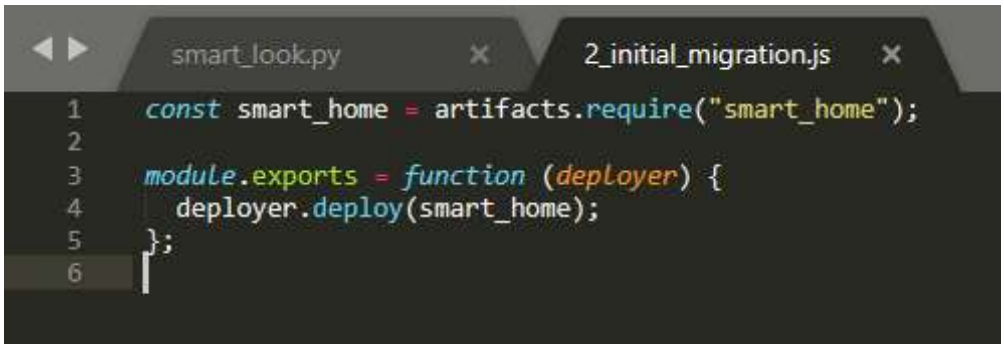


Figura 63. Archivo JavaScript 2_initial_migration.js

Elaborado por el Investigador

4. Edite el archivo de configuración de Truffle para lo cual abra el archivo truffle-config.js, limpie y agregue las siguientes líneas de códigos.

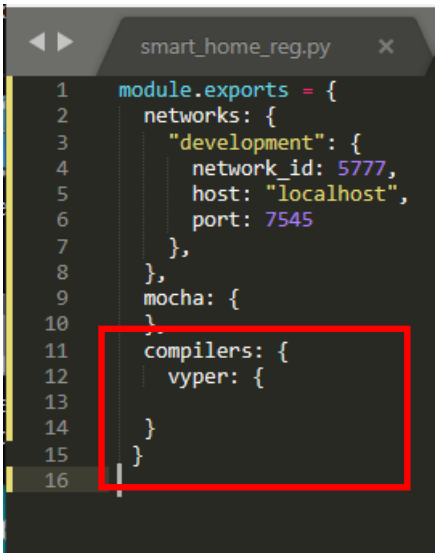


Figura 64. Archivo JavaScript para compilar el contrato inteligente

Elaborado por el Investigador

El host y el port se toman del servidor RPC en la pantalla Ganache, y network_id se toma del ID de red en la pantalla Ganache.

Figura 65. Dirección del RCP SERVER

Elaborado por el Investigador

5. Ejecute scripts de migración para implementar el contrato inteligente

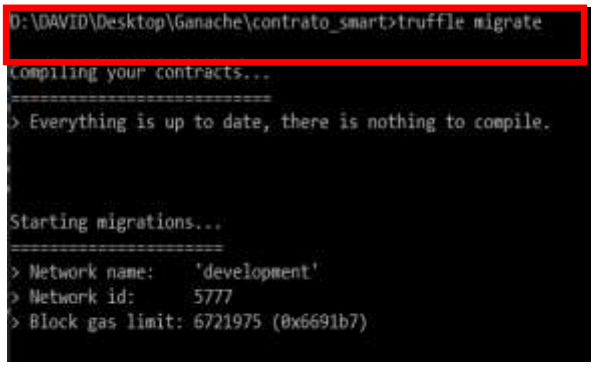


Figura 66. Implementación del contrato Inteligente

Elaborado por el Investigador

El marco Truffle tomará su código de bytes definido en el Donation.json archivo y lo enviará a Ganache. Esto le proporcionará el siguiente resultado:

Figura 67.Dirección del contrato Inteligente

Elaborado por el Investigador

Su contrato inteligente escrito con Vyper se implementó en Ganache. La dirección de su contrato inteligente es la siguiente:

- **contract address**: 0x98524d3Ce1D06FED808DA7B4019E9421947Cb1db
- **account:** 0x9942b70Bef262B4788B6e952ac34E0e1A0ec2E66

En la cadena de bloques de Ganache tenemos la siguiente salida:

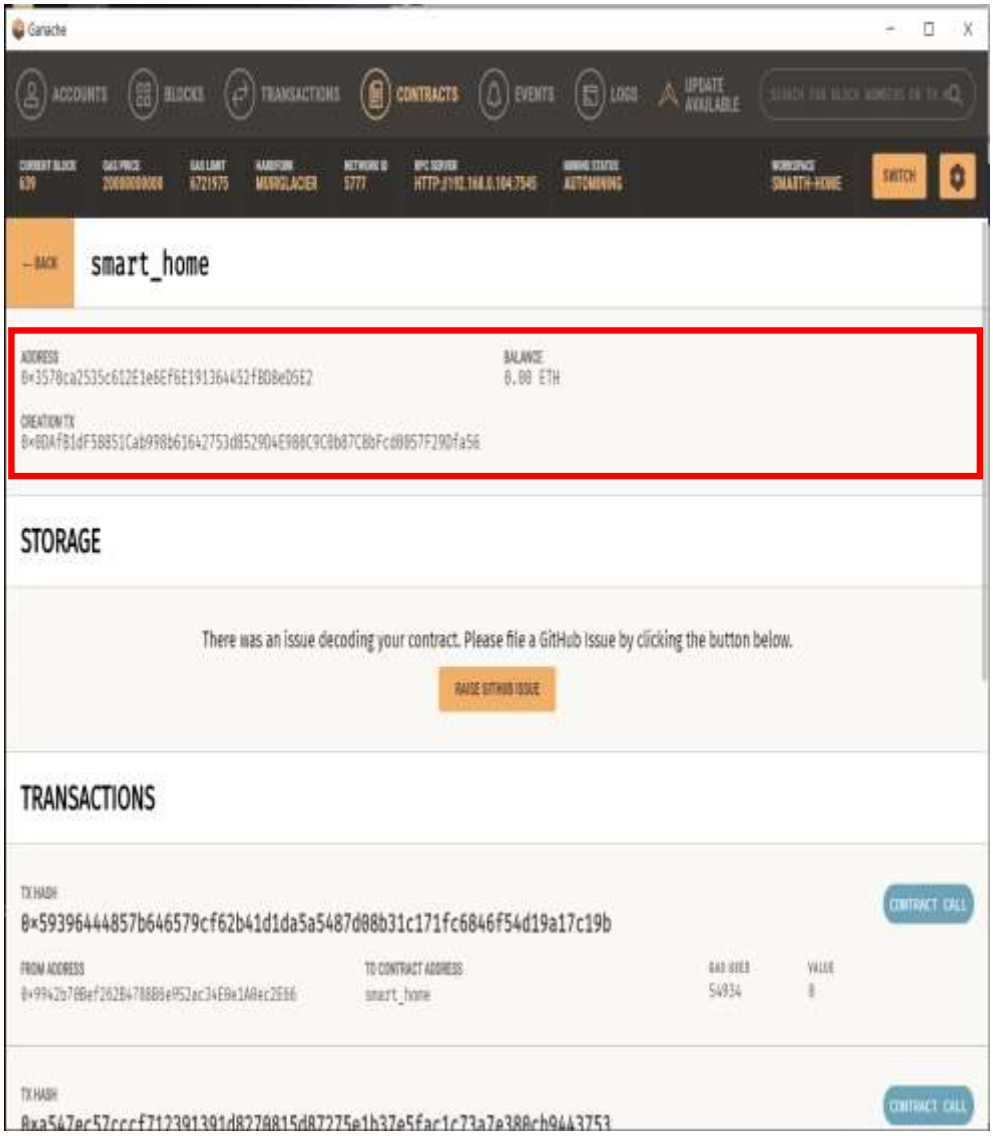


Figura 68. Contrato Inteligente implementado en la Cadena de Bloques en Ethereum en la plataforma Ganache

Elaborado por el Investigador

Implementación del sistema Smart home aplicando IoT y Blockchain

Para la implementación se tomó 10 personas de la base de datos, para ser registrados en el contrato inteligente como usuarios Autentificados, ver Anexo D. La figura 70 se muestra el diagrama físico del sistema completo con sus dos componentes principales el dispositivo IoT que se encarga del análisis y envió de los datos a la cadena de bloques Ethereum que a su vez se encarga de la seguridad y almacenamiento de los mismo.

Tabla 11. Usuarios Autentificados en el Contrato Inteligente

Usuario	Altura(m)	Tiempo Ingreso(s)	Autentificación
David	1.81	4	User1
Martha	1.63	7	User2
Oswaldo	1.71	6	User3
Edisson	1.76	4	User4
Alberto	1.80	5	User5
Santiago	1.55	5	User6
Roberto	1.78	4	User7
Christian	1,75	5	User8
Micaela	1.65	4	User9
Ricardo	1.85	5	User10

En la figura 69 la el diagrama de bloques del sistema muestra cada componente que lo conforma y su interpretación de cómo se interconectan entre cada dispositivo por medio de una conexión wifi

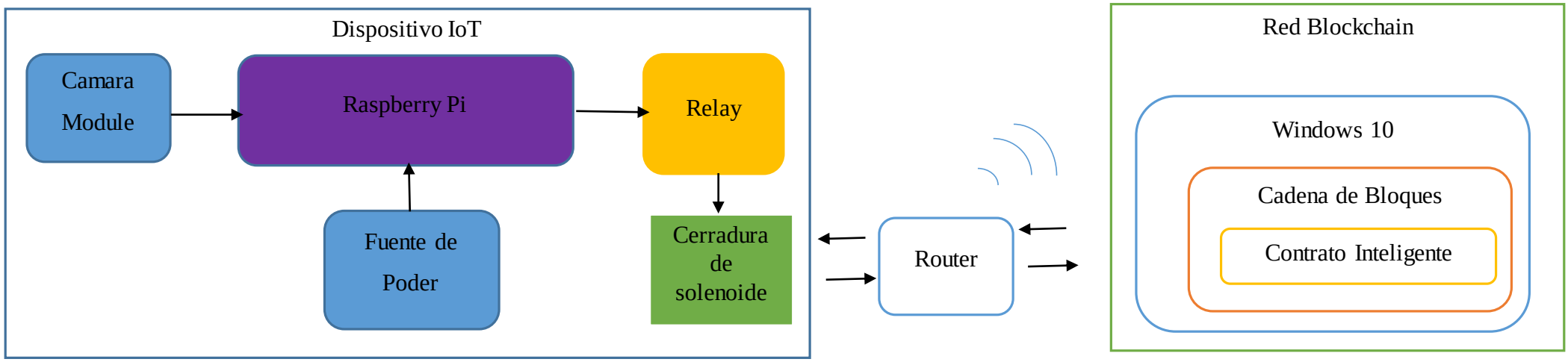


Figura 69. Diagrama de Bloques de Sistema Smart Home aplicando IoT y Blockchain

Elaborado por el Investigador

Figura 70. Diagrama fisico del Sistema de Smart Home aplicando IoT y Blockchain

Elaborado por el Investigador

Para el tiempo de espera del sistema después de la creación del nuevo bloque de la red Blockchain y autentificado por el contrato inteligente se estimó un tiempo de 10 segundos en la que cerradura de solenoide permanece activa lo que permite a los usuarios ingresar con tranquilidad al hogar.

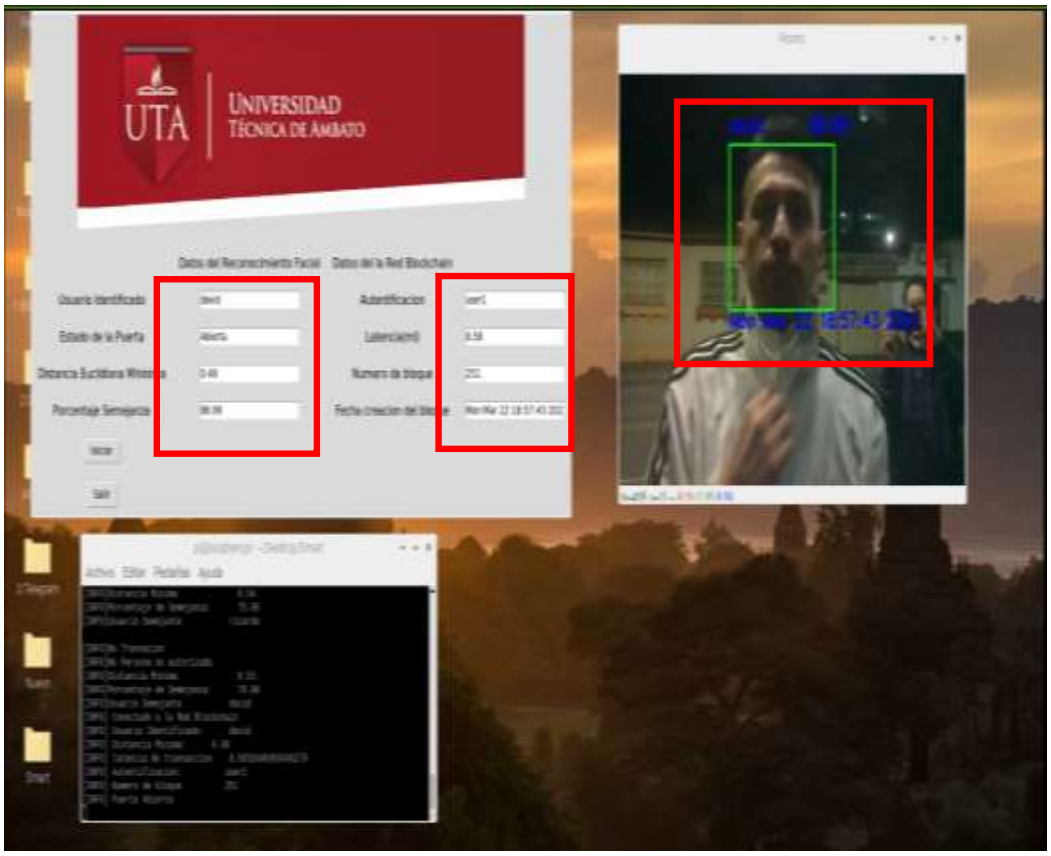


Figura 71. Implementación del Sistema Smart Home Aplicando IoT y Blockchain

Elaborado por el Investigador

En la figura 71 se observa en interfaz gráfica dos bloques en el lado derecho están los datos concernientes al reconocimiento facial nombre del usuario , distancia euclidiana y el porcentaje de similitud del rostro detectado y el bloque de la izquierda esta una descripción de los datos de la cadena de bloques Ethereum como el nombre de autentificación correspondiente al ingresado en el contrato inteligente, la latencia que existe entre él envió de los datos por el dispositivo IoT y la creación de un nuevo bloque en el Blockchain y el mensaje de autentificación emitido por el contrato inteligente. En la ventana Rostros se observa un marco de color verde en el rostro detectado y en la parte superior del rectángulo el nombre del usuario y el porcentaje de similitud y en la parte inferior la descripción de la hora en la que se creó el nuevo bloque.

3.1.1.5 Pruebas realizadas

Para el análisis del sistema se realizó el análisis del sistema completo por medio del tiempo de respuesta entre el dispositivo IoT y cadena de bloques Ethereum, después se realizaron varias pruebas al sistema de reconocimiento facial para garantizar su optimo desempeño y finalmente se realizó las pruebas de seguridad a la cadena de bloques Ethereum para garantizar la integridad de los datos tanto de envío y recepción entre el dispositivo IoT y la cadena de bloques Ethereum que se describen a continuación

3.1.1.5.1 Pruebas del Sistema de Smart Home aplicando IoT y Blockchain

Métricas de evaluación y análisis desempeño del Sistema

El análisis de rendimiento explora diferentes tipos de configuraciones para lograr un rendimiento óptimo para un número determinado de transacciones. Para este propósito se utilizó la métrica de latencia.

Latencia espera a que la transacción especificada se incluya en un bloque y luego devuelve el recibo de la transacción.

Para la prueba de latencia entre el dispositivo IoT y el contrato inteligente smart_home.vy ver anexo F implementado en la cadena de bloques Ganache se tomó un lote 50 muestras como se muestrea en la tabla 12 ver anexo H

El promedio de latencia de cada transacción entre el dispositivo IoT y la cadena de bloques en Ganache es de 0,4023 milisegundos lo cual permite que la información no se pierda y la confirmación del usuario sea al instante como se esperaría de un sistema en tiempo real.

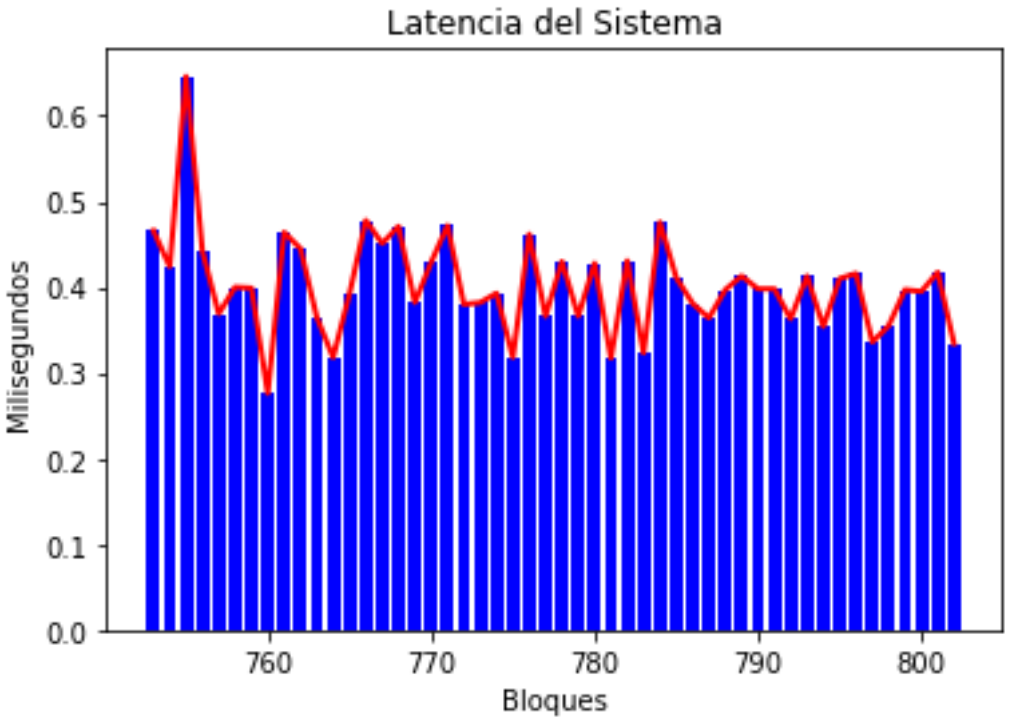


Figura 72. Muestras de latencia del Sistema de Smart Home aplicando IoT y Blockchain

Elaborado por el Investigador

3.1.1.5.2 Pruebas en el sistema de reconocimiento facial

3.1.1.5.2.1 Pruebas de precisión y Sensibilidad

Métricas y parámetros

Esta sección se describe la métrica y parámetros que se utilizó, además de definir una configuración adecuada para los algoritmos de detección y reconocimiento.

Métricas de rendimiento

1. **N:** el número total de muestras de la imagen
2. **Matriz de confusión:** La matriz de confusión es una matriz que se utiliza para determinar el rendimiento de los modelos de clasificación para un conjunto dado de datos de prueba. Explicación de los términos asociados con la matriz de confusión es la siguiente:
    - **Verdadero negativo-TN: el** modelo ha dado la predicción No, y el valor real o real también fue No.
    - **Verdadero positivo-TP:** el modelo predijo que sí, y el valor real también fue cierto.
    - **Falso negativo-FN:** el modelo predijo que no, pero el valor real fue Sí, también se denomina **error de tipo II**.
    - **Falso positivo-FP:** el modelo ha predicho Sí, pero el valor real fue No. También se denomina **error de tipo I.**
3. **Precisión:** Define la frecuencia con la que el modelo predice la salida correcta.

$$P = \frac{TP + TN}{N}$$

4. **Sensibilidad:** la recuperación puede definirse como la cantidad de positivos devueltos por el modelo

$$S = \frac{TP}{TP + FN}$$

Parámetros de los algoritmos

6. **Clasificadores en cascada (Haar Cascades)**
   - **ScaleFactor:** Este parámetro especifica que tanto va a ser reducida la imagen
   - **MinNeighbors:** Este parámetro especifica cuántos vecinos debe tener cada rectángulo candidato para retenerlo.
   - **MinSize:** Este parámetro indica el tamaño mínimo posible del objeto.
   - **MaxSize:** Este parámetro indica el tamaño máximo posible del objeto
7. **Histograma de gradientes orientados (HOG) en Dlib**
   - **Tolerance**: Cuánta distancia entre caras para considerarlo una coincidencia. Cuanto más bajo es más estricto. 0,6 es el mejor rendimiento típico.

Prueba 1

El primer set de pruebas está compuesto de 30 usuarios, con un numero de muestras de 100 imágenes por usuario, para lo cual se realizó un script test_dace_reg.py, revisar Anexo H

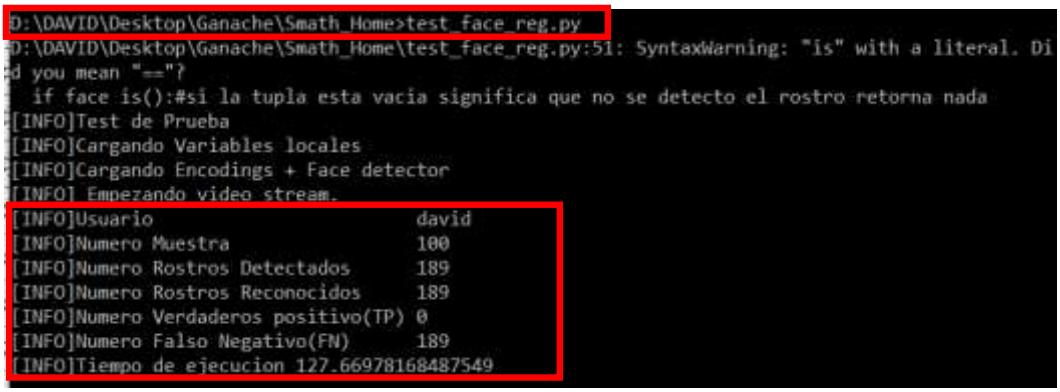


Figura 73. Primera prueba del Sistema de Reconocimiento Facial

Elaborado por el Investigador

Para la primera prueba se configuraron loa parámetros valores bajos para el detector y valor altos para el algoritmo de reconocimiento.

Tabla 12. Parámetros de la Primera Prueba

Algoritmo	Parámetros
Detector	**Scale Factor:** 1.01 **Min Neighbors:** 1 **Min Size:** None **Max Size:** None
Reconocimiento	**Tolerance**=0.8

En la tabla se muestra la cantidad de rostros detectados y reconocidos en una muestra de 100 imágenes por usuario y los valores correctos según el usuario ingresado

Tabla 13. Resumen de la Primera Prueba

N.	Usuario	Muestras	Rostros detectados	Rostros reconocidos	Verdadero Positivo(TP)	Falso Negativo (FP)
1	**Alejandro**	100	404	404	180	224
2	**Alberto**	100	310	310	101	209
3	**Alex**	100	220	220	19	201
4	**Anahi**	100	370	370	28	348
5	**Andres**	100	200	200	21	179
6	**Andy**	100	201	201	0	201
7	**Anthony**	100	690	690	13	677
8	**Camila**	100	204	204	1	203
9	**Caroline**	100	588	588	113	475
10	**Christian**	100	491	491	1	490
11	**Cristian**	100	344	344	1	343
12	**David**	100	189	189	0	189
13	**Edisson**	100	123	122	0	123
14	**Gabriel**	100	217	216	0	217
15	**Jeff**	100	853	853	198	655
16	**jose**	100	126	126	0	126
17	**joshep**	100	110	110	9	101
18	**Katerine**	100	104	104	2	102
19	**Maria**	100	209	209	3	206
20	**Martha**	100	143	143	0	143
21	**Mayra**	100	419	419	0	419
22	**Micaela**	100	378	378	0	378

23	**Olga**	100	759	754	4	755
24	**Oscar**	100	146	146	0	146
25	**Oswaldo**	100	141	141	0	141
26	**Paola**	100	151	151	0	151
27	**patricia**	100	201	201	0	201
28	**Ricardo**	100	103	103	0	103
29	**Roberto**	100	181	181	0	181
30	**Santiago**	100	330	330	0	300
	Total	3000	8905	8898	694	8187

En la tabla se observa el resultado obtenido en un conjunto de 3000 imagenes de test. Con los datos verdaderos positivos y falso negativo, se procede a calcular las estadísticas mencionadas en la sección anterior.

Precisión

$$\boldsymbol{P = \frac{TP + TN}{N} = \frac{694}{3000} = 0.2313 * 100 = 23,13\%}$$

Sensibilidad:

$$\boldsymbol{S = \frac{TP}{TP + FN} = \frac{694}{694 + 8187} = 0.0781 * 100 = 7,81\%}$$

Los resultados obtenidos durante la prueba 1 fueron bajos de cierta manera por lo valores de los parámetros configurados en el detector y reconocimiento facial. Se puede observar que la probabilidad de reconocer con éxito un usuario es 23,13 %y la de tener falsa alarmas es del 8 %.

Ahora se observa los resultados de la primera prueba del sistema de reconocimiento facial, como se muestra en la Figura 74.

Figura 74. Resultado del sistema de reconocimiento Facial

Elaborado por el Investigador

Prueba 2

En la segunda prueba se aumentó la cantidad de muestras a 150 por usuario y una configuración de los parámetros valores medios para el detector y valor medios para el algoritmo de reconocimiento.

Tabla 14. Parámetros de la Segunda Prueba

Algoritmo	Parámetros
Detector	**ScaleFactor:** 1.1 **MinNeighbors:** 2 **MinSize:** (20,20) **MaxSize:** (400,400)
Reconocimiento: **Histograma de gradientes orientados (HOG) en Dlib**	**Tolerance**=0.6

En la tabla 11 se muestra la cantidad de rostros detectados y reconocidos en una muestra de 150 imágenes por usuario y los resultados según la variación de parámetros de cada uno de los algoritmos utilizados

Tabla 15. Resumen de la Segunda Prueba

N.	Usuario	Muestras	Rostros detectados	Rostros reconocidos	Verdadero Positivo(TP)	Falso Negativo (FP)
1	**Alejandro**	150	156	139	100	56
2	**Alberto**	150	150	150	150	0
3	**Alex**	150	46	46	15	31
4	**Anahi**	150	150	150	150	0
5	**Andres**	150	150	150	92	58
6	**Andy**	150	150	150	150	0
7	**Anthony**	150	150	150	23	127
8	**Camila**	150	150	150	150	0
9	**Caroline**	150	149	149	149	0
10	**Christian**	150	150	150	146	4
11	**Cristian**	150	153	151	30	123
12	**David**	150	148	148	48	100
13	**Edisson**	150	150	150	150	0
14	**Gabriel**	150	150	150	75	75
15	**jeff**	150	31	30	30	1

16	**jose**	150	150	150	32	118
17	**joshep**	150	150	150	150	0
18	**Katerine**	150	150	148	148	2
19	**Maria**	150	170	150	150	20
20	**martha**	150	150	150	126	24
21	**mayra**	150	150	150	150	0
22	**micaela**	150	153	151	108	45
23	**olga**	150	161	155	6	155
24	**oscar**	150	123	108	29	94
25	**oswaldo**	150	149	149	139	10
26	**paola**	150	118	118	116	2
27	**patricia**	150	185	175	40	145
28	**ricardo**	150	150	150	0	150
29	**roberto**	150	164	154	0	164
30	**santiago**	150	150	150	12	138
		4500	4306	4221	2664	1642

En la tabla 12 se observa una congruencia entre las imágenes de muestra y los rostros detectado los que permitió valores de verdadero positivo aumentaran

Precisión

$$\boldsymbol{P = \frac{TP + TN}{N} = \frac{2664}{4500} = 0.594 * 100 = 59,2\%}$$

Sensibilidad:

$$\boldsymbol{S = \frac{TP}{TP + FN} = \frac{2664}{2664 + 1642} = 0.618 * 100 = 61.86\%}$$

Los resultados obtenidos durante la prueba 2 mejoraron en base a la nueva configuración de los parámetros de cada algoritmo. Se observa que la probabilidad de reconocer con éxito un usuario es 59.2 %y la de tener falsa alarmas es del 61,86%.

Ahora se observa los resultados de la segunda prueba del sistema de reconocimiento facial, como se muestra en la Figura 75.

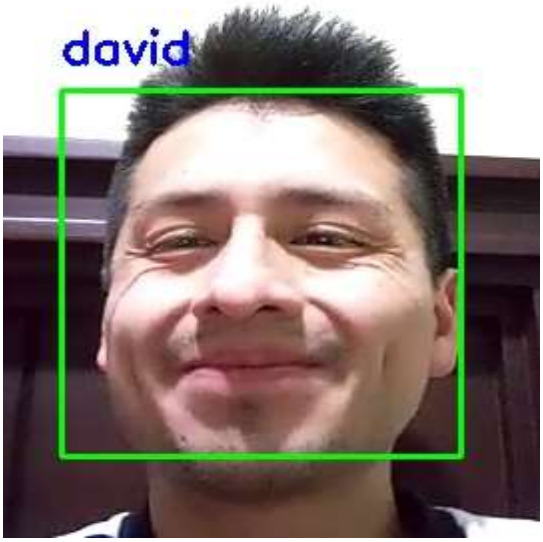


Figura 75. Resultado de la Segunda Prueba

Prueba 3

Para la prueba número 3 aumentó la cantidad de muestras a 200 por usuario y una nueva configuración de los parámetros.

Tabla 16. Parámetros de la tercera Prueba

Algoritmo	Parámetros
Detector	**Scale Factor:** 1.1 **Min Neighbors:** 3 **Min Size:** (1000,100)
Reconocimiento	**Tolerance**=0.5

Tabla 17. Resumen de la tercera Prueba

N.	Usuario	Muestras	Rostros detectados	Rostros reconocidos	Verdadero Positivo(TP)	Falso Negativo (FP)
1	**Alejandro**	**200**	**203**	**191**	**191**	**12**
2	**Alberto**	**200**	**200**	**200**	**200**	**0**
3	**Alex**	**200**	**198**	**198**	**198**	**0**
4	**Anahi**	**200**	**200**	**192**	**192**	**8**
5	Andres	200	200	200	200	0
6	Andy	200	200	200	200	0
7	Anthony	200	200	200	191	9
8	Camila	200	200	200	200	0
9	Caroline	200	159	93	93	66
10	Christian	200	180	156	154	26
11	Cristian	200	170	170	170	0
12	David	200	195	194	194	1
13	Edisson	200	200	200	200	0
14	Gabriel	200	200	174	174	26
15	jeff	200	17	17	17	0
16	jose	200	198	194	194	4
17	joshep	200	200	200	200	0
18	Katerine	200	198	116	116	82
19	Maria	200	202	200	200	2
20	martha	200	200	200	200	0
21	mayra	200	200	199	199	1
22	micaela	200	197	197	197	0
23	olga	200	184	114	52	132
24	oscar	200	200	187	187	13
25	Oswaldo	200	173	173	173	0
26	Paola	200	104	104	104	0
27	patricia	200	199	199	199	0
28	Ricardo	200	200	175	29	171

29	Roberto	200	195	179	179	16
30	Santiago	200	200	200	200	0
Total		6000	5572	5222	5003	569

En la tabla 14 se observa un gran aumento en los valores verdaderos positivos con un mayor número de muestras por Usuario.

Precisión

$$\boldsymbol{P = \frac{TP + TN}{N} = \frac{5003}{6000} = 0.833 * 100 = 83.3\%}$$

Sensibilidad:

$$\boldsymbol{S = \frac{TP}{TP + FN} = \frac{5003}{5003 + 569} = 0.897 * 100 = 89.78\%}$$

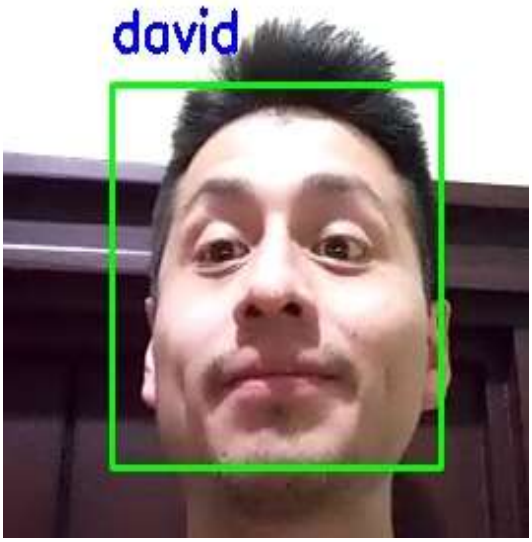


Figura 76. Resultado de la Tercera Prueba

3.1.1.5.2.2 Pruebas de eficiencia del sistema de reconocimiento facial

Para la prueba 4 se realizó cálculo de la eficiencia de la Detecion y reconocimiento para 5 personas que no se encuentran en la base de datos, donde primero se realizó el análisis de distancia euclidiana entre mismo rostro de cada usuario en la base de datos para luego hace un análisis de la distancia entre rostros de personas que no se encuentran en la base de datos y ver la similitud que existe.

Tabla 18. Usuarios no registrados en la Base Datos

Usuarios no Registrados en la base de datos	
1	Pablo
2	Nancy
3	Marlon
4	Daniel
5	Alison

Para lo cual hacemos uso de la face_distance () función compara una lista de codificaciones de caras con una codificación de caras conocida y obtiene una distancia euclidiana para cada rostro en la base de datos. La distancia euclidiana permite conocer qué tan similares son las caras. Ver Anexo D

Tabla 19. Distancia Euclidiana de los usuarios de la base de datos

Usuarios	Distancia Euclidiana
Alberto	0.4527
Alejandro	0.4516
Alex	0.3107
Anahi	0.3245
Andres	0.3930
Andy	0.4349
Anthony	0.4349
Camila	0.4837
Caroline	0.4562
Christian	0.4942
Cristian	0.3877
David	0.3757
Edisson	0.3364
Gabriel	0.4910
Jeff	0.2966
jose	0.4191
joshep	0.4179

Katerine	0.4974
Maria	0.4192
Martha	0.3728
Mayra	0.4803
Micaela	0.3341
Olga	0.4687
Oscar	0.4691
Oswaldo	0.1508
Paola	0.4580
patricia	0.2888
Ricardo	0.4079
Roberto	0.3613
Santiago	0.3678
Promedio	0.4012

En la tabla 14 se muestra la distancia mínima euclidiana de los 30 usuarios ingresados en la base de datos en donde se muestra que la distancia mínima está en un rango de 0.3 a 0.4 para lo cual se tomó el promedio de 0.401 para el análisis con usuario no registrados en la base de datos.

Tabla 20.Distancia euclidiana de usuario no Registrado número 1

Usuario no Registrado: Pablo			
Usuario	Referencia	Distancia Euclidiana Referenci	Eficiencia(%)
Alberto	0.40	0.68555552	52.51
Alejandro	0.40	0.78487125	35.93
Alex	0.40	0.70285494	49.62
Anahi	0.40	0.68854229	52.01
Andres	0.40	0.59965375	66.86
Andy	0.40	0.71253524	48.01
Anthony	0.40	0.63180758	61.49
Camila	0.40	0.73113866	44.90
Caroline	0.40	0.63594438	60.80
Christian	0.40	0.63668672	60.67
Cristian	0.40	0.68668287	52.32
David	0.40	0.57929749	70.26
Edisson	0.40	0.84898064	25.22
Gabriel	0.40	0.75694887	40.59
Jeff	0.40	0.74092476	43.27
jose	0.40	0.64531208	59.23
joshep	0.40	0.63828476	60.41

Katerine	0.40	0.62351977	62.87
Maria	0.40	0.55635545	74.09
Martha	0.40	0.62948967	61.88
Mayra	0.40	0.56641644	72.41
Micaela	0.40	0.5831391	69.62
Olga	0.40	0.68601222	52.44
Oscar	0.40	0.68832161	52.05
Oswaldo	0.40	0.69391903	51.12
Paola	0.40	0.65832524	57.06
patricia	0.40	0.61493119	64.31
Ricardo	0.40	0.68624427	52.40
Roberto	0.40	0.61713481	63.94
Santiago	0.40	0.77462818	37.64

En el análisis al primer usuario no registrado se obtuvo como mínima distancia euclidiana de 0.566 con un porcentaje de eficiencia de 72,41 con respecto al usuario en la base de datos lo cual está fuera del rango establecido lo que resulta sin coincidencias con ningún usuario en la base de datos

Tabla 21. Distancia euclidiana de usuario no Registrado número 2

Usuario no registrado Nancy			
Usuario	Referencia	Distancia Euclidiana Calculada	Eficiencia (%)
Alberto	0.40	0.62253952	63.04
Alejandro	0.40	0.56019513	73.45
Alex	0.40	0.72884409	45.28
Anahi	0.40	0.67367232	54.50
Andres	0.40	0.73653695	44.00
Andy	0.40	0.78848747	35.32
Anthony	0.40	0.73208514	44.74
Camila	0.40	0.82816654	28.70
Caroline	0.40	0.79798782	33.74
Christian	0.40	0.72968378	45.14
Cristian	0.40	0.55932093	73.59
David	0.40	0.72017949	46.73
Edisson	0.40	0.73019861	45.06
Gabriel	0.40	0.7760243	37.40
Jeff	0.40	0.75293154	41.26
jose	0.40	0.65422493	57.74
joshep	0.40	0.7258089	45.79
Katerine	0.40	0.76039771	40.01
Maria	0.40	0.7201539	46.73
Martha	0.40	0.7696932	38.46
Mayra	0.40	0.72668622	45.64
Micaela	0.40	0.74190631	43.10
Olga	0.40	0.73578124	44.12

Oscar	0.40	0.71915432	46.90
Oswaldo	0.40	0.71577363	47.47
Paola	0.40	0.72583533	45.79
patricia	0.40	0.79514203	34.21
Ricardo	0.40	0.56846124	72.07
Roberto	0.40	0.61793102	63.81
Santiago	0.40	0.75810924	40.40

El valor de semejanza más alto es de 73.59 con una distancia euclidiana de 0.55 un valor por encima del valor de referencia 0.4 lo que arroja como resultado que es una persona desconocida

Tabla 22. Distancia Euclidiana de usuario no Registrada_número3

Usuario no Registrado: Marlon			
Usuario	Referencia	Distancia Euclidiana Calculada	Eficiencia
Alberto	0.40	0.70211982	49.75
Alejandro	0.40	0.78146054	36.50
Alex	0.40	0.66748661	55.53
Anahi	0.40	0.62164477	63.19
Andres	0.40	0.6452999	59.24
Andy	0.40	0.70249164	49.68
Anthony	0.40	0.66768514	55.50
Camila	0.40	0.71021603	48.39
Caroline	0.40	0.65346802	57.87
Christian	0.40	0.68841426	52.04
Cristian	0.40	0.70577866	49.14
David	0.40	0.87805371	20.37
Edisson	0.40	0.78016086	36.71
Gabriel	0.40	0.73146229	44.85
Jeff	0.40	0.78853336	35.32
jose	0.40	0.64818618	58.75
joshep	0.40	0.69751987	50.51
Katerine	0.40	0.65707001	57.27
Maria	0.40	0.66371578	56.16
Martha	0.40	0.63787119	60.48
Mayra	0.40	0.75570164	40.80
Micaela	0.40	0.66527437	55.90
Olga	0.40	0.65656756	57.35
Oscar	0.40	0.66302417	56.28
Oswaldo	0.40	0.66546083	55.87
Paola	0.40	0.73638033	44.02
patricia	0.40	0.72081095	46.62
Ricardo	0.40	0.65919261	56.92
Roberto	0.40	0.63413706	61.10
Santiago	0.40	0.67954047	53.52

En la tabla 17 se observa que la eficiencia en este usuario no registrado es de 59% con una distancia euclidiana mínima 0.64 lo que esta fuera del rango y reconoce el rostro como desconocido

Tabla 23. Distancia Euclidiana de usuario no Regitrado numero 4

Usuario no reconocido Daniel			
Usuario	Referencia	Distancia Euclidiana Calculada	Eficiencia
Alberto	0.40	0.68132389	53.22
Alejandro	0.40	0.7745505	37.65
Alex	0.40	0.73161998	44.82
Anahi	0.40	0.584057	69.46
Andres	0.40	0.69805708	50.42
Andy	0.40	0.62945157	61.88
Anthony	0.40	0.66876782	55.32
Camila	0.40	0.80869224	31.95
Caroline	0.40	0.62007076	63.45
Christian	0.40	0.69426137	51.06
cristian	0.40	0.63975317	60.16
david	0.40	0.65567612	57.50
david	0.40	0.77638557	37.34
edisson	0.40	0.83023991	28.35
gabriel	0.40	0.70027324	50.05
jeff	0.40	0.62021841	63.42
jose	0.40	0.60567936	65.85
joshep	0.40	0.64178607	59.82
katerine	0.40	0.65487502	57.64
maria	0.40	0.66831849	55.39
martha	0.40	0.67656086	54.01
martha	0.40	0.7208631	46.62
mayra	0.40	0.63280217	61.32
micaela	0.40	0.62893169	61.97
olga	0.40	0.69702705	50.60
oscar	0.40	0.68720539	52.24
oswaldo	0.40	0.71542682	47.52
oswaldo	0.40	0.68478558	52.64
paola	0.40	0.77358972	37.81
patricia	0.40	0.7513466	41.53
ricardo	0.40	0.6582821	57.07
roberto	0.40	0.61344062	64.56
santiago	0.40	0.67949491	53.52

En la tabla numero 18 los resultados son eficiencia máxima es de 69.46 con una distancia euclidiana mínima de 0.58 que esta fuera del rango lo que da como resultado como usuario desconocido

Tabla 24. Distancia Euclidiana del usuario No registrado numero 5

Usuario no registrado Alison			
Usuario	Referencia	Distancia Euclidiana Calculada	Eficiencia(%)
alberto	0.40	0.69123	51.56
alejandro	0.40	0.77553	37.49
alex	0.40	0.733434	44.52
anahi	0.40	0.584057	69.46
andres	0.40	0.698344	50.38
andy	0.40	0.623435	62.89
anthony	0.40	0.634356	61.06
camila	0.40	0.812543	31.31
caroline	0.40	0.620904	63.31
christian	0.40	0.565759	72.52
cristian	0.40	0.834326	27.67
david	0.40	0.655676	57.50
edisson	0.40	0.833453	27.81
gabriel	0.40	0.700273	50.05
jeff	0.40	0.620218	63.42
jose	0.40	0.605679	65.85
joshep	0.40	0.641786	59.82
katerine	0.40	0.654875	57.64
maria	0.40	0.668318	55.39
martha	0.40	0.676561	54.01
mayra	0.40	0.783423	36.17
micaela	0.40	0.628932	61.97
olga	0.40	0.697027	50.60
oscar	0.40	0.687205	52.24
oswaldo	0.40	0.713454	47.85
Paola	0.40	0.589723	68.52
patricia	0.40	0.723415	46.19
Ricardo	0.40	0.658282	57.07
Roberto	0.40	0.547576	75.56
Santiago	0.40	0.634364	61.06

En los resultados de la tabla numero 19 la eficiencia máxima es de 75.56 con una distancia mínima euclidiana de 0.54 lo que arroja como resultado, que es un usuario

En el análisis de la eficiencia la métrica de la distancia euclidiana que existe entre los rostros de la base de datos y los rostros que no lo están se evidencia que los porcentajes de semejanza son muy bajos entre un rango mínimo de 30% y un

rango máximo de un 75% lo que indica que el sistema de reconocimiento facial no se confundirá con rostros que no se encuentren en la base de datos.

Análisis del ingreso de una persona discapacitada

Para este análisis se simuló ingreso de una persona en silla de ruedas para observar la respuesta de sistema y el tiempo de ingreso al hogar es el adecuado para personas con discapacidad

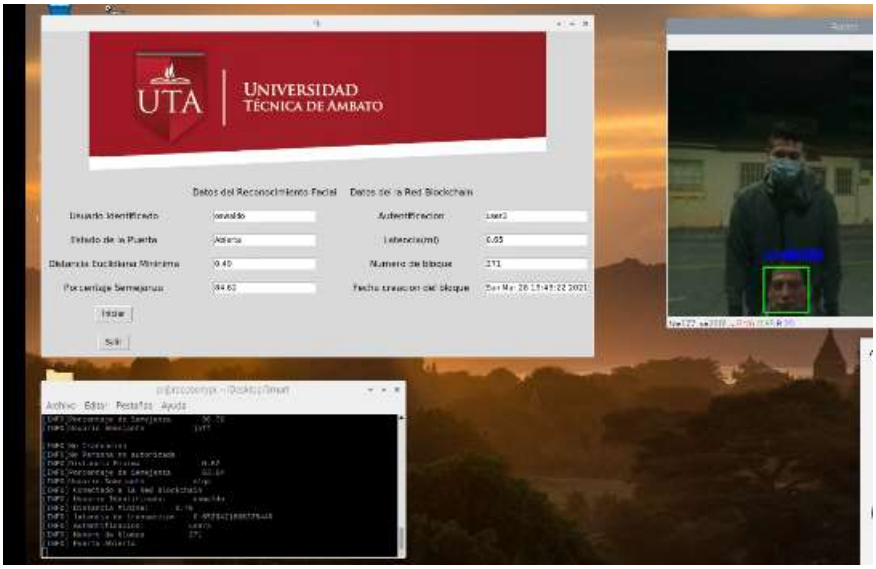


Figura 77. Detección de la persona discapacitada

Como se observa en la Figura 77 detecta y reconoce a la persona discapacitada y se registra en la cadena Blockchain y lo autentifica User3

Figura 78. Ingreso de la persona discapacitada

Elaborado por el Investigador

En la Figura 78 se ve el ingreso de la persona discapacitada corroborando que el tiempo establecido en el script de 10 segundos es suficiente para que pueda ingresar con tranquilidad al hogar

3.2 Pruebas de Seguridad de la Red Blockchain

El sistema de Smart Home aplicando IoT y Blockchain deben considerar tres requisitos de seguridad principales: confidencialidad, integridad y disponibilidad. La confidencialidad garantiza que solo los usuarios autorizados pueden participar en el sistema. Integridad es responsable de los mensajes enviados al destino sin cambios, y la disponibilidad significa que los usuarios siempre pueden manejar los datos cuando los necesiten. Se evaluó el margen de seguridad de nuestro método propuesto bajo varias amenazas.

Para la prueba de seguridad del sistema se instaló el software Kali Linux, revisar Anexo I

3.2.1.5.1.1 Ataque Man-in-the-middle

El ataque de hombre en el medio es un tipo de ciberataque, que se realiza en una red de área local. En este ataque, el hacker se interpone entre las dos partes de la comunicación e intercepta los datos.

Para el ataque man in middle en el Sistema de Smart Home aplicando IoT y Blockchain se insertarán las comunicaciones entre el dispositivo IoT y la cadena de bloques Ethereum.

Verificamos la puerta de enlace de la red

Figura 79. Puerta de enlace del Sistema de Smart Home aplicando IoT y Blockchain

Elaborado por el Investigador

Verificamos la dirección del dispositivo IoT

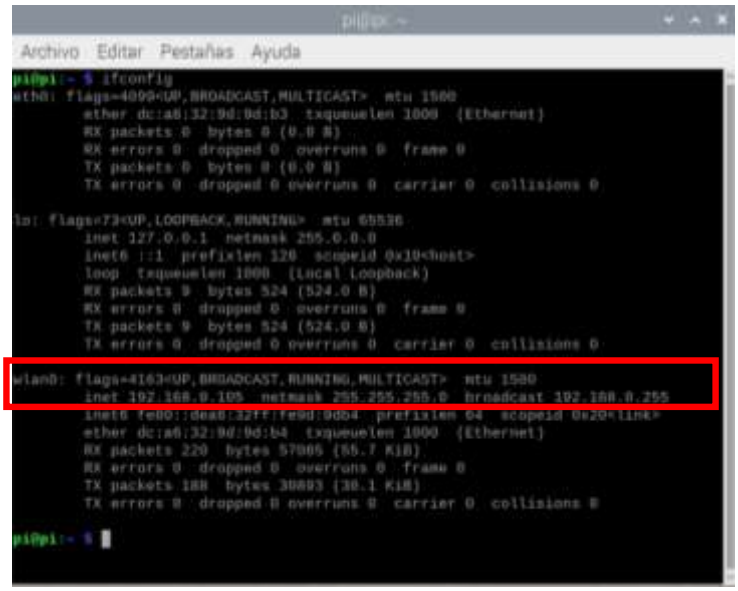


Figura 80. Dirección de Ip del dispositivo IoT

Elaborado por el Investigador

Tabla 25. Direccion de dispositivos del Sistema de Smart Home aplicando IoT y Blockchain

Información	**Dirección Ip**
IP del enrutador	**192.168.0.1**
IP del Dispositivo IoT	**192.168.0.105**

Implementación del ataque usando la máquina virtual Kali Linux

1. Inicié la máquina virtual Kali Linux conectada a la misma red que el dispositivo IoT y escriba el siguiente comando para abrir Ettercap:
   -sudo Ettercap –G

Figura 81. Ettercap interceptor/sniffer/registrador para LANs con switch

Elaborado por el Investigador

2. Seleccione la dirección IP del enrutador 192.168.0.1 y agregue eso al destino 1, también seleccione la dirección IP del dispositivo IoT de destino 192.168.0.105 y agregue eso al destino 2.

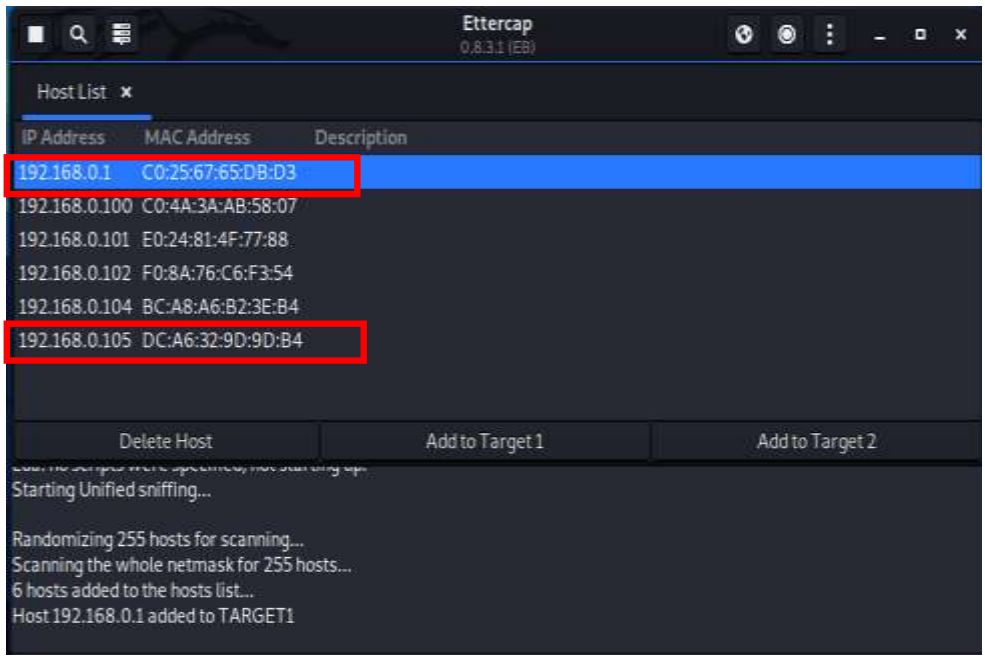


Figura 82. Ingreso de la dirección de los dispositivos Sistema de Smart Home aplicando IoT y Blockchain

Elaborado por el Investigado

3. Abra el menú de arriba y haga clic en la pestaña MITM y el menú desplegable tendrá una selección llamada "Envenenamiento por ARP"

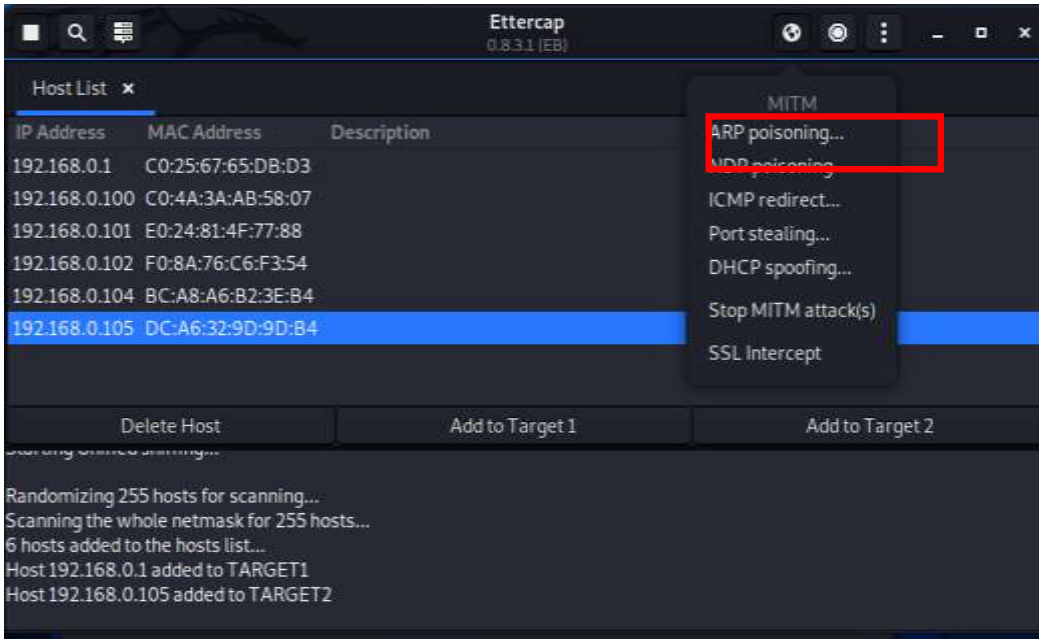


Figura 83. Envenenamiento por ARP

Elaborado por el Investigador

4. Selecciónelo y se abrirá una ventana emergente como la siguiente. Seleccione "Detectar conexiones remotas".

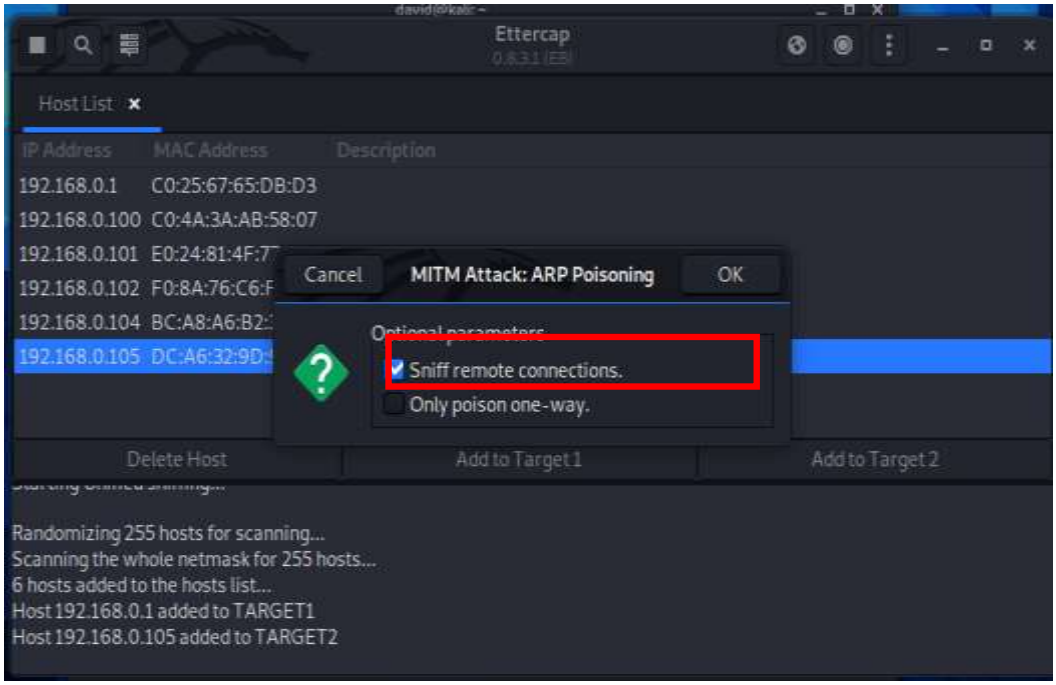


Figura 84. Detección de conexiones remotas

Elaborado por el Investigador

5. Ettercap comenzará el envenenamiento de ARP y verá que Ettercap responde en sus ventanas principales con el siguiente mensaje.

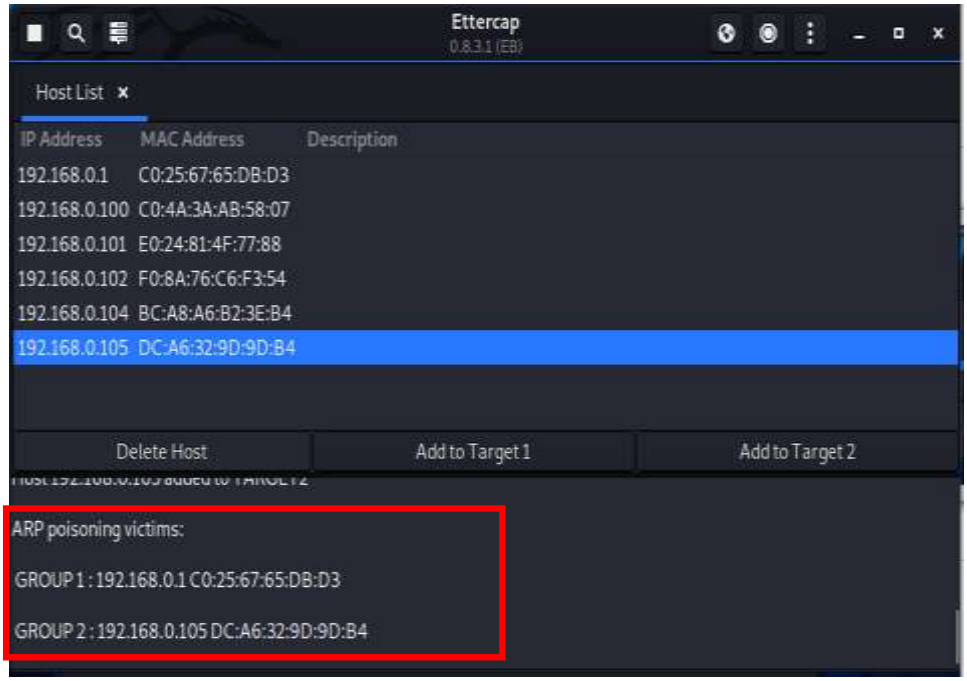


Figura 85. Obtención de la información con el Ataque Man-in-the-middle

Elaborado por el Investigador

En el programa Ettercap no se obtuvo información del Sistema de Smart Home aplicando IoT y Blockchain para lo cual se optó por la aplicación Wireshark para monitoria los datos que se envía desde el dispositivo IoT hacia la red Blockchain lo cual arrojó solo el hash de la transacción realizada

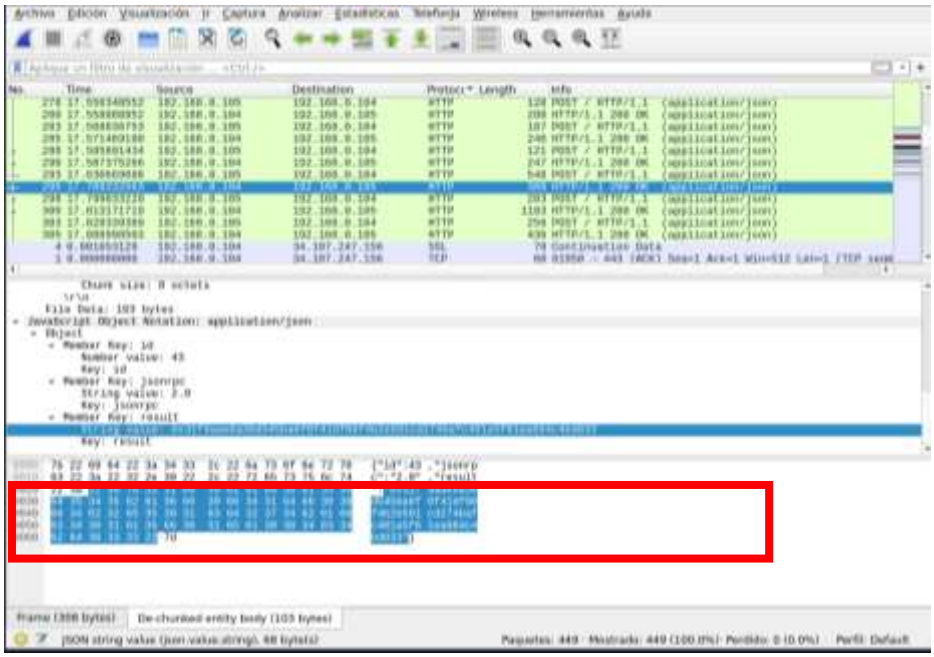


Figura 86. Monitoreo con Wireshark

Elaborado por el Investigador

En la red Blockchain se observar que la información del Tx hash en lo único visible por medio de Wireshark, pero los datos de la dirección del contrato, dirección de la cuenta y la clave privada de la cuenta no son obtenidos en el tráfico de la red por la aplicación.

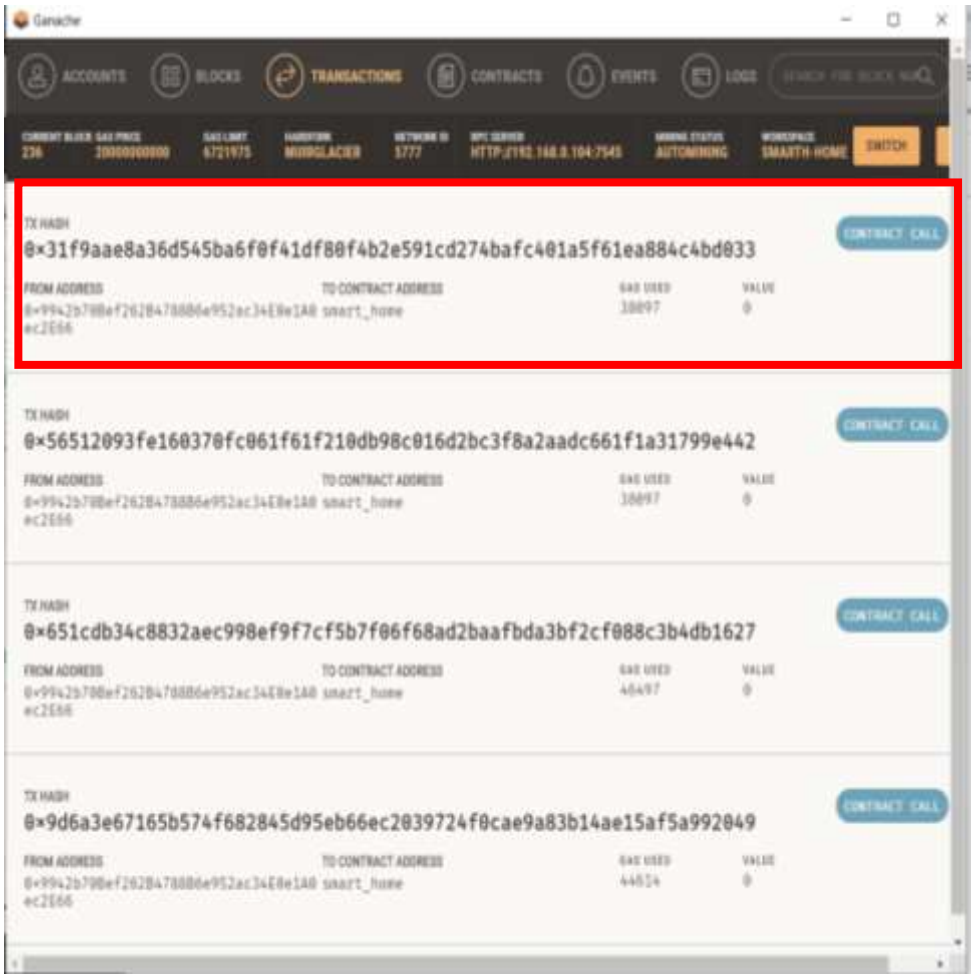


Figura 87. Hash de la transacción en la cadena de bloques Ganache

Elaborado por el Investigador

En ataque Man in middle se obtuvo información del hash de la transacción, pero no los datos que envía el dispositivo IoT y de datos de autentificación que el contrato inteligente envía desde la cadena de bloques Ethereum. El uso de un valor nonce y marcas de tiempo en el mensaje de autenticación previene la reproducción y los ataques MIM

3.2.1.5.1.2 Ataque de denegación de servicio

Los atacantes intentan impedir que los usuarios auténticos accedan a los servicios de la red. Aquí, los atacantes aumentan el tráfico en la red al iniciar una enorme cantidad de información inútil para causar congestión en la red. La arquitectura descentralizada de la cadena de bloques puede ser resistente a los ataques DoS / DDoS. Incluso si los atacantes bloquean algunos nodos con éxito, no pueden bloquear todos los nodos de la red. Además, las transacciones en Blockchain requieren pagos.

Para el ataque de denegación de servicio se ha utilizado Metasploit Auxiliary para lazar el ataque a la cadena de bloques Ethereum en Ganache con la inundación SYN es un tipo de ataque DoS que utiliza para enviar una gran cantidad de sincronización para consumir todos los recursos del sistema objetivo.

Los pasos son los siguientes:

1. Escribir en la consola msfconsole

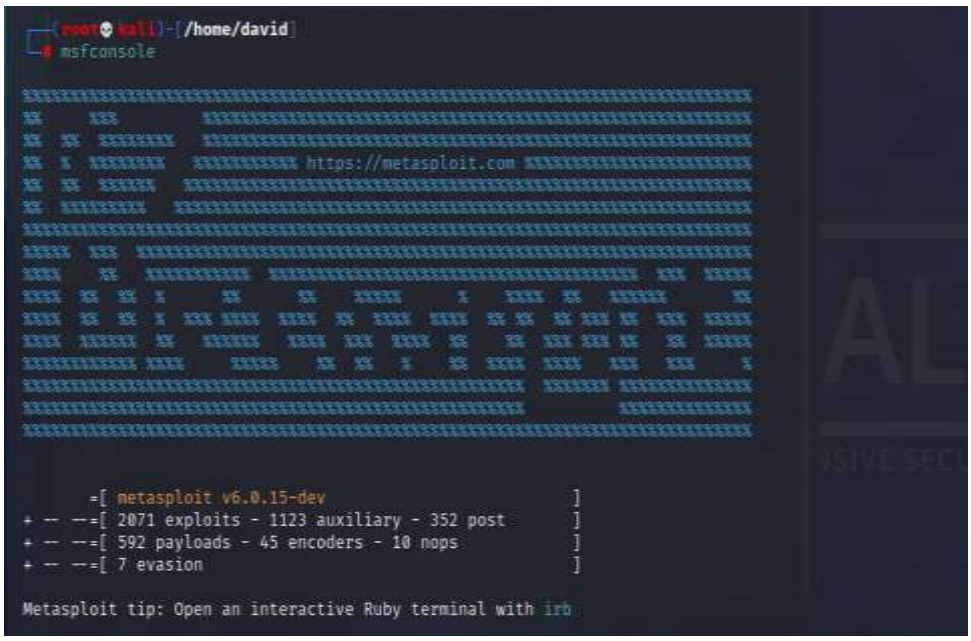


Figura 88.Framework Metasploit

Elaborado por el Investigador

2. Utilice la opción de seleccionar el auxiliar "auxiliar / dos / tcp / synflood"

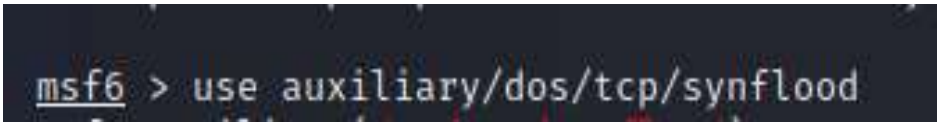


Figura 89. Ejecución msf6

Elaborado por el Investigador

3. Configurar rhost **y** rport**,** que es la dirección de destino y los números de puerto respectivamente.

```
msf6 auxiliary(dos/tcp/synflood) > set rport 7545
rport ⇒ 7545
msf6 auxiliary(dos/tcp/synflood) > set rhost 192.168.0.104
rhost ⇒ 192.168.0.104
```

Figura 90. Parámetros de la cadena de Bloques Ganache

Elaborado por el Investigador

4. para lanzar el ataque, simplemente escriba exploit, para que comience la inundación de sincronización

```
msf6 auxiliary(dos/tcp/synflood) > exploit
[*] Running module against 192.168.0.104

[*] SYN flooding 192.168.0.104:7545 ...
```

Figura 91. Ejecución del exploit

Elaborado por el Investigador

5. colocamos Wireshark en la máquina de destino para mostrar cuántos paquetes llegan a la máquina.

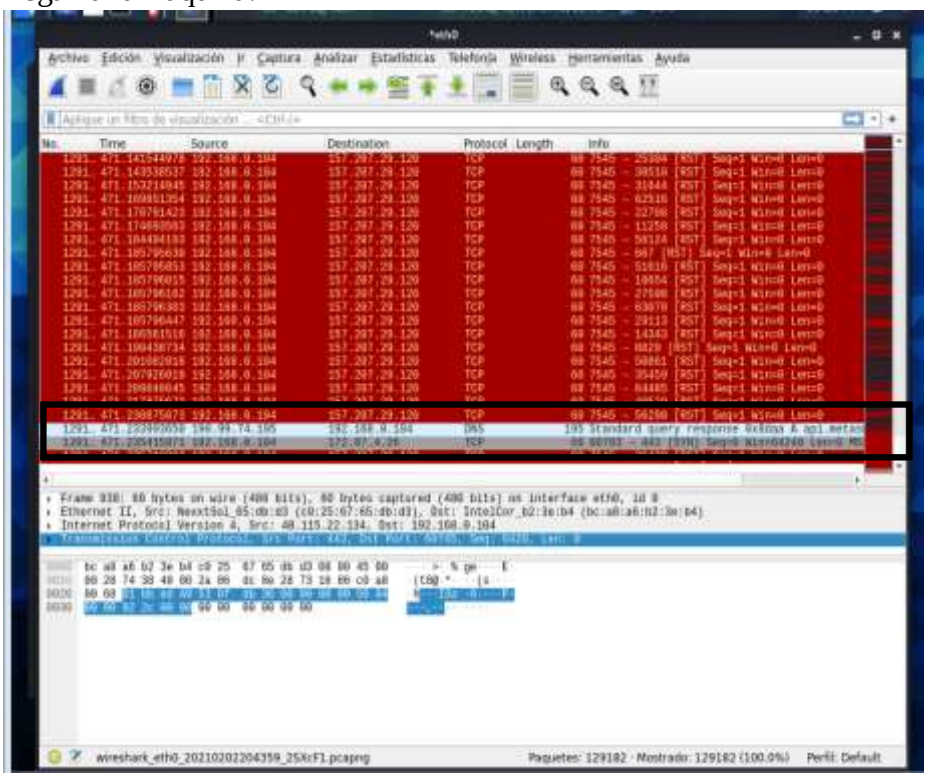


Figura 92. Inundación de paquetes por el exploit

Elaborado por el Investigador

En la aplicación Wireshark se observa un valor de 129182 paquetes capturados minutos después del lanzamiento del ataque, pero la cadena de bloques continúa realizando transacción entre el dispositivo IoT y la cadena de bloques Ethereum como se muestra en la figura 93

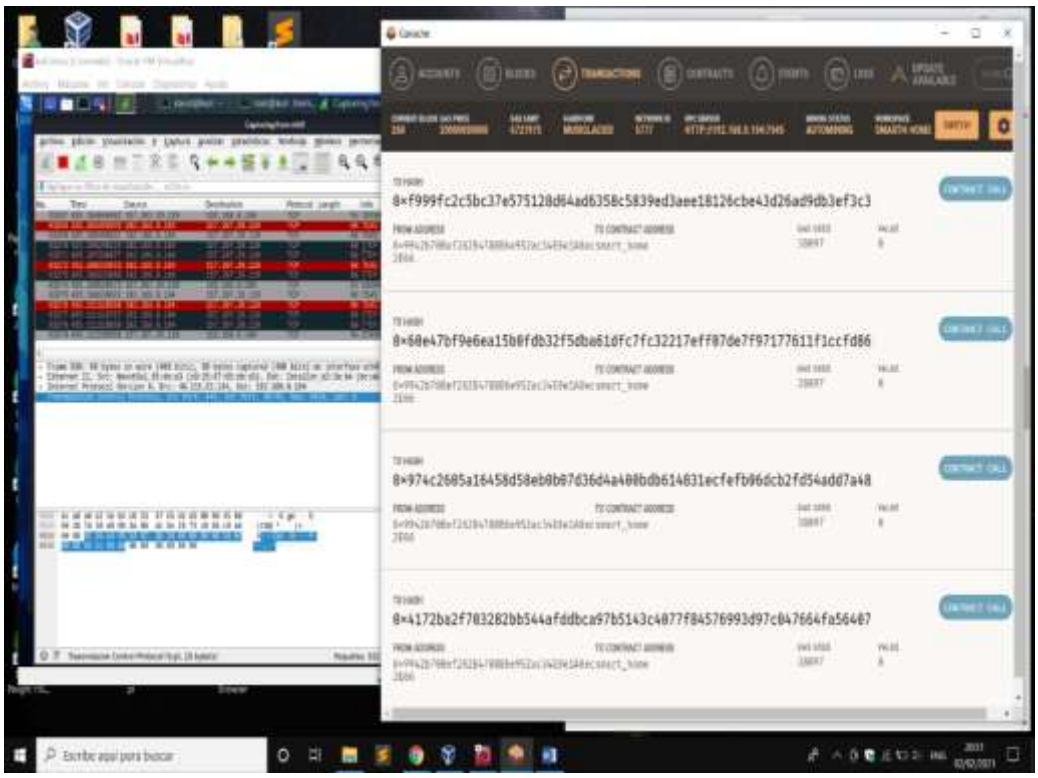


Figura 93. Transacciones de la cadena de bloques en el momento de la ejecución del exploit

Elaborado por el Investigador

3.2.1.5.1.3 Ataque Spoofing

Un ataque de suplantación de identidad, en ciberseguridad, se refiere a un objeto que pretende ser la identidad de otro objeto en un intento de ganarse la confianza de uno, ingresar al sistema y robar datos o fondos o difundir malware.

Para este ataque se simuló un dispositivo intruso en la Pc que contiene el mismo script que el dispositivo IoT para que envié datos a la cadena de bloques Ethereum y comprobar si puede ingresar al hogar como se observa en la Figura 94, a pesar de realizar transacciones y crear nuevos bloques el dispositivo IoT no es engañado y no se puede activar o desactivar la cerradura de la puerta inteligente por lo cual no se ve comprometido el sistema.

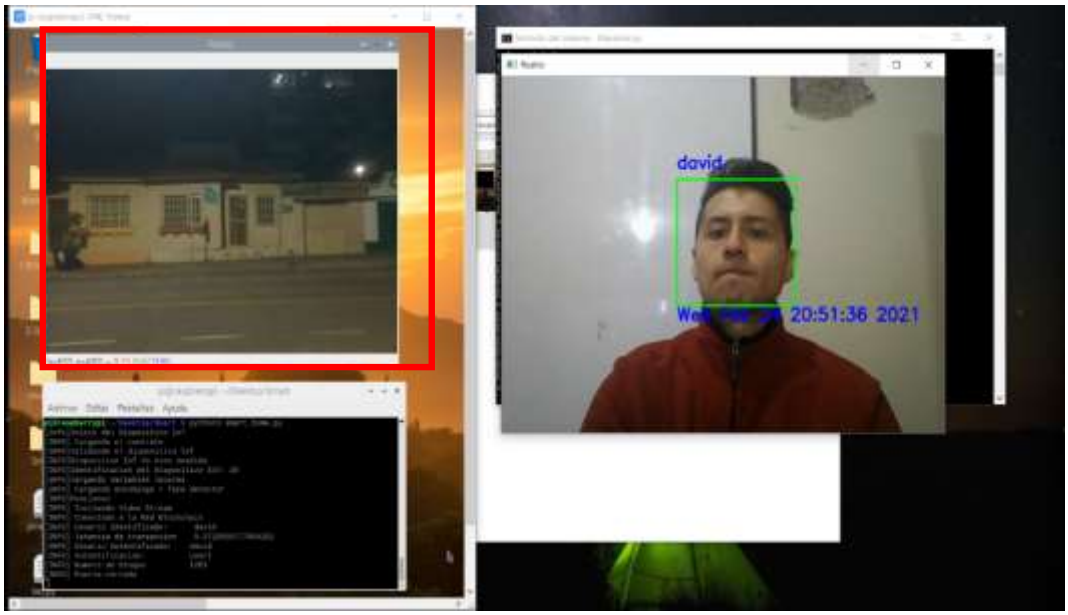


Figura 94. Ataque Spoofing al Dispositivo IoT

Elaborado por el Investigador

En la figura 95 Se observa que se realizan varias transacciones del Usuario 1 por parte del dispositivo atacante pero el dispositivo IoT no recibe ningún mensaje de confirmación por lo cual el sistema no se es violentado y no se puede ingresar al Hogar.

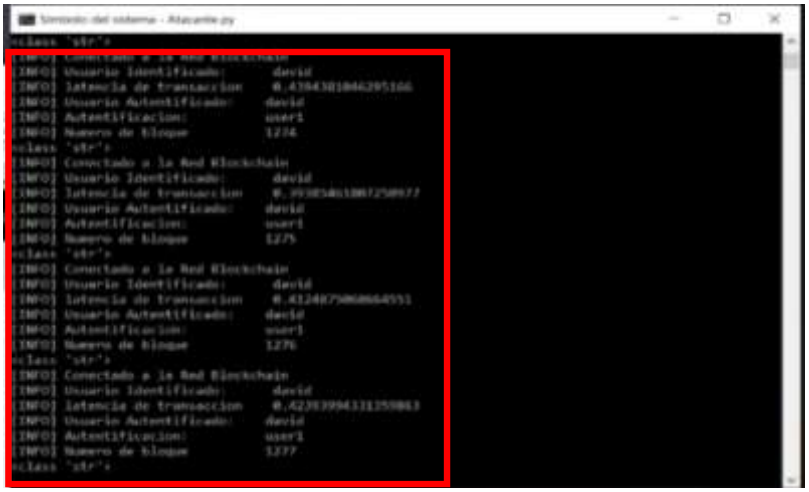


Figura 95. Envió de datos del dispositivo atacante a la red Blockchain

Elaborado por el Investigador

3.2.1.6 Costos del prototipo

Tabla 26. Costos del Sistema de Smart Home aplicando IoT y Blockchain

Equipo	Cantidad	Precio Unitario	Total
Raspberry pi 4	1	80	80
Cámara Raspberry pi	1	20	20
Relay	1	2.50	2.50
Carcasa impresa en 3D	1	10	10
Cerradura de Solenoide	1	15	15
Cable de Timbre	1	1.50	1.50
Fuente de poder 12v	1	3	3
Pc Dell i7	1	800	800
Total			932

CAPITULO IV

4. CONCLUSIONES Y RECOMENDACIONES

4.1. Conclusiones

- Se simuló con éxito la manipulación de un sistema IoT mediante el uso de un contrato inteligente utilizando el lenguaje Vyper en la cadena de Bloques Ethereum en Ganache. La función hash y el algoritmo de cifrado de la cadena de bloques Ethereum puede fortalecer el sistema IoT que ha demostrado el nivel de seguridad en las pruebas realizadas.
- El contrato inteligente implementado en la cadena de bloques Ethereum permite manejar las políticas de control de acceso, el almacenamiento de datos y la gestión del flujo de datos. Con el contrato inteligente se puede establecer las reglas para controlar de acceso a la Smart Home.
- Las pruebas realizadas han demostrado que el uso de la tecnología Blockchain en el contexto del IoT es factible con niveles de latencia promedio de 0,4 milisegundos en la en la creación de la función hash y el algoritmo de cifrado de Ethereum lo que indica que el sistema es adecuado para ser en tiempo real
- Mediante la prueba de seguridad se proporciona un análisis de la integridad de los dispositivos IoT y se demostró que la solución propuesta logra los objetivos de seguridad y es resistente a ataques como Man in middle, DoS y Spoofing. Además, se optó como lenguaje Vyper que permite contratos más seguros, fáciles de implementar y legibles porque se restringe liberadamente acciones que están permitidos en otros lenguajes.

4.2. Recomendaciones

- Para la implementación de contratos inteligentes en la cadena de bloques Ethereum en Ganache se debe instalar Truper herramienta para compilar contratos de Vyper compatible con truffle
- Para realizar cualquier transacción se debe calcular el valor del gas para que la transacción se realice con éxito
- Para el reconocimiento facial la base de datos de las imágenes del usuario debe contener fotos acordes al ambiente en donde se va a trabajar

5. Bibliografía

[1] K. R. Ö. a. A. Yurdakul, "Work-in-Progress: Integrating Low-Power IoT devices to a Blockchain-Based Infrastructure," IEEE, Seoul, Republic of Korea, 2017.

[2] K. A. B. R. a. S. M. A. Abdur Rahman, "System, A Natural User Interface and Blockchain-Based In-Home Smart Health Monitoring," IEEE, Qatar, 2018.

[3] K. M. Dinan Fakhri, "Secure IoT Communication using Blockchain Technology," IEEE, Bandung, Indonesia, 2018.

[4] M. I. R. M. Ulfah Nadiya, "Blockchain-based Secure Data Storage for Door Lock System," in *4th Intenational Conference on Information Technology, Information Systems and Electrical Engineering*, Indonesia, 2019.

[5] H. Ü. Kerem Ataşen, *Designing a Secure IoT Network by Using Blockchain*, IEEE, Ed., Kirklareli: IEEE, 2019.

[6] M. L. A. B. S. S. K. M. Y. Ahsan Manzoor, *Blockchain based Proxy Re-Encryption Scheme for Secure IoT Data Sharing*, Finland: IEEE, 2019.

[7] N. Petracek, "Forbes," Forbes, 2018 Julio 18. [Online]. Available: https://www.forbes.com/sites/forbestechcouncil/2018/07/18/is-Blockchain-the-way-to-save-iot/?sh=104d26275a74. [Accessed 07 Noviembre 2020].

[8] I. Bashir, Mastering Blockchain, Segunda ed., B. Rai, Ed., Birmingham B3 2PB, UK: Packt Publishing Ltd, 2018 , pp. 8-41.

[9] G. D. a. P. S. P. Bikramaditya Singhal, A Beginner's Guide to Building, New York, 233: Apress, 2018.

[10] H.-N. Dai, "Blockchain for Internet of Things: A Survey," Mayo 2019.

[11] M. Fabien, "towardsdatascience.com," towards data science, 4 Abril 2019. [Online]. Available: https://towardsdatascience.com/a-guide-to-face-detection-in-python-3eab0f6b9fc1. [Accessed 2021 01 08].

[12] D. C. V. Daniel Paternain Dallo, *Detección automática de personas mediante Histograma de Gradientes Orientados*, Navarra: Universidad Publica de Navarra, 2017.

[13] X. X. •. I. W. •. M. Staples, Architecture for Blockchain Applications, Eveleigh, NSW: 3, 2019.

[14] A. Y. Kazım Rıfat Özyılmaz, *Work-in-Progress: Integrating Low-Power IoT devices to a Blockchain-Based Infrastructure*, Seoul, Republic of Korea: IEEE, 2017.

[15] M. I. R. M. Ulfah Nadiya, *Blockchain-based Secure Data Storage for Door Lock System*, Bandung: IEEE, 2019.

[16] T. S. Gunawan*, "Development of Face Recognition on Raspberry Pi for Security Enhancement of Smart Home System," *Indonesian Journal of Electrical Engineering and Informatics (IJEEI)*, p. 10, 2017.

[17] R. Sourav, M. Z. H. Md Nasir Uddin and M. J. Kabir, "Design and Implementation of the Smart Door Lock System with Face Recognition Method using the Linux Platform Raspberry Pi," *International Journal of Computer Science and Network,*, vol. 7, p. 7, 2018.

[18] F. r. b. d. u. s. u. R. Pi, "Thulluri Krishna Vamsi,Kanchana Charan Sai,Vijayalakshmi M," *International Journal of Advance Research, Ideas and Innovations in Technology*, vol. Volume 5, p. 5, 2019.

[19] M. V. P. C. K. S. D. Ishita Gupta, "FACE DETECTION AND RECOGNITION USING RASPBERRY PI," *t: https://www.researchgate.net/publication/335219198*, p. 6, 2016.

[20] P. L. N. T. Niclas Kullig, "Prototype Implementation and Evaluation of a Blockchain Component on IoT Devices," *The 15th International Conference on Future Networks and Communications (FNC)*, 2020.

[21] M. K. M. A. M. A. K. S. Randa Almadhoun, "A User Authentication Scheme of IoT Devices using Blockchain-enabled Fog Nodes," 09 November 2018.

[22] P. L. N. T. Niclas Kullig, *Prototype Implementation and Evaluation of a Blockchain Component on IoT Devices*, Germany: Elsevier B.V, 2020.

[23] K. Salah, *A User Authentication Scheme of IoT Devices using Blockchain-enabled Fog Nodes*, Abu Dhabi, UAE: Khalifa University of Science and Technology, 2018.

[24] J. Z. a. M. Wu, *Blockchain Use in IoT for Privacy-Preserving Anti-Pandemic Home Quarantine*, Nanjing : School of Computer Science & Technology, Nanjing University of Posts and Telecommunication, 2020.

[25] Z. H. Z. L. M. X. Quanqing Xu, "Building an Ethereum-based Decentralized Smart Home System," in *24th International Conference on Parallel and Distributed Systems (ICPADS)*, Beijing, 2018.

[26] Y. H. M. L. P. E. K. T. Ahsan Manzoor, *Demo: A Delay-Tolerant Payment Scheme on the Ethereum Blockchain,* Sydney: IEEE, 2018.

[27] J. M. L. D.-S. K. Kevin Putra Dirgantoro, *Generative Adversarial Networks Based on Edge Computing With Blockchain Architecture for Security System,* Gumi,Corea del Sur: Kumoh National Institute of Technology, 2020.

[28] M. S. Erick Fernando, *Blockchain Technology Implementation In Raspberry Pi For Private Network*, Bina Nusantara University: IEEE, 2019.

[29] T. T. Yu Nandar Aung, *Review of Ethereum: Smart Home Case Study*, Bangkok, Thailand : Faculty of ICT, Mahidol University, 2017.

[30] D. S. Connelly, *Smart Contr t Contract Vulnerabilities on the Ether abilities on the Ethereum Blockchain: a Current Perspective,* Portland: Portland State University , 2020 .

[31] D. W. W. X. Y. W. Fengyin Li, *Wireless Communications and Mobile Computing BlockchainBased Trust Management in Distributed Internet of Things,* Rizhao: Hindawi, 2020.

[32] G. S. R. G. K. R. M. Rupa Ch, *Security and privacy of UAV data using Blockchain technology*, Journal of Information Security and Applications, 2020.

[33] S. &. J. L. A. C. M. B. P. H. L. ARED R. BUTCHER, *Cybersecurity Tech Basics: Blockchain Technology Cyber Risks and Issues:Overview,* Thomson Reuters. , 2019.

[34] E. KARATAŞ, *Developing Ethereum Blockchain-Based Document Verification Smart Contract for Moodle Learning Management System,* Ankara, Türkiye: International Journal of Informatics Technologies, 2018.

[35] H. J. D. K.-K. R. C. Zeli WANG, *Ethereum smart contract security research: survey and future research opportunities,* Higher Education Press, 2020.

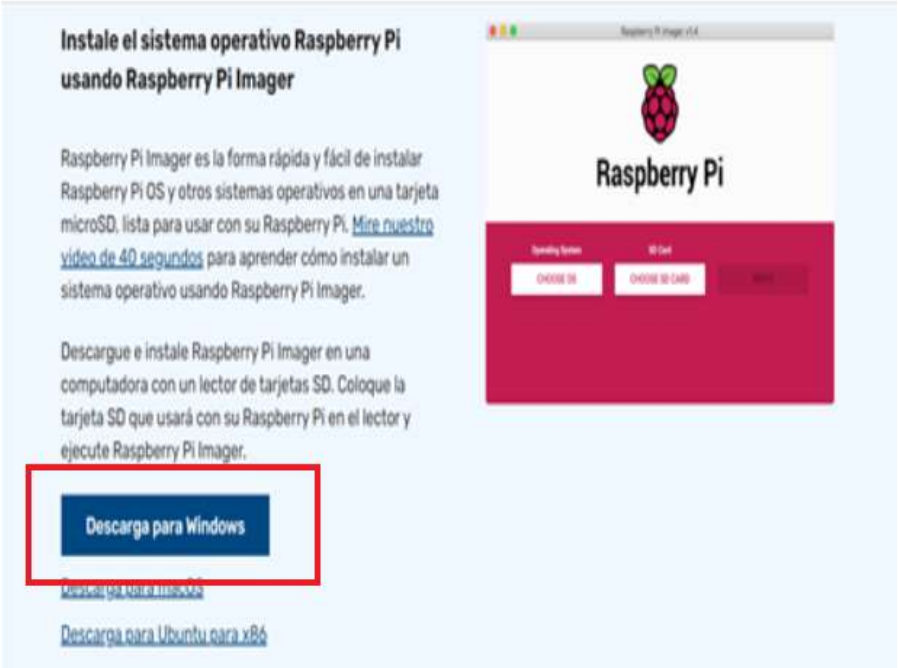
[36] A. E. C. G. Fátima Lea, *Performance Evaluation of Private Ethereum Networks,* Dublin: Springer Nature Singapore Pte Ltd, 2020.

[37] A. E. C. L. Horacio González–Vélez, *Performance Evaluation of Private Ethereum Networks,* Dublin: Springer Nature Singapore Pte Ltd, 2020.

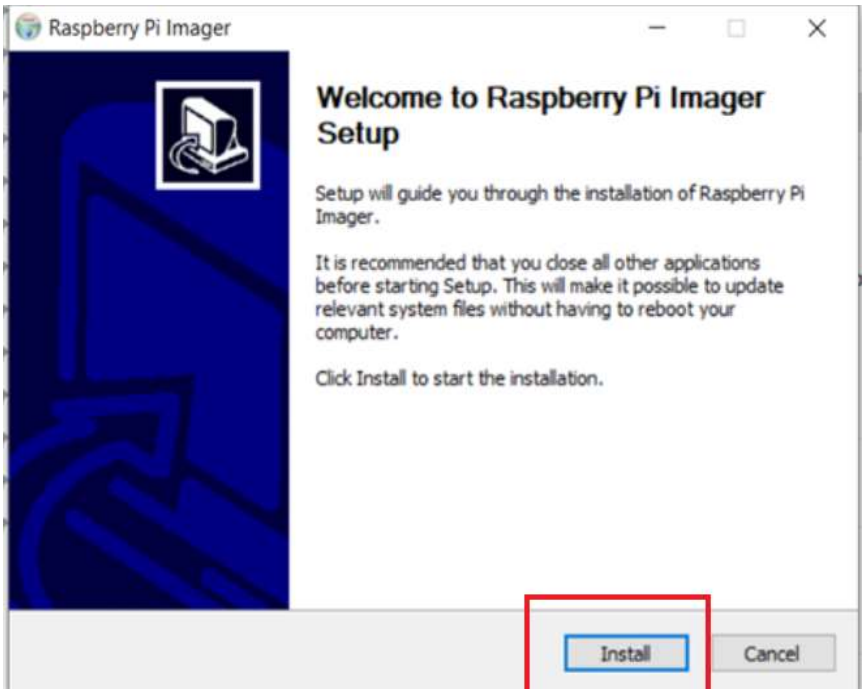
Anexos

Anexo A: Instalación del Sistema Operativo Rasbian en la Raspberry pi 4

Descargue Raspberry Pi Imager en una computadora con un lector de tarjetas SD



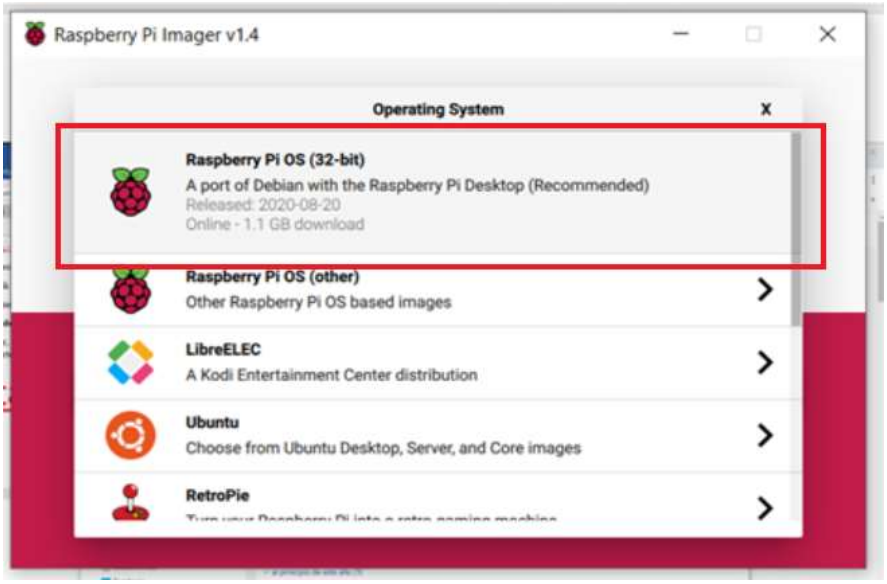
Instalar Raspberry Pi Imager



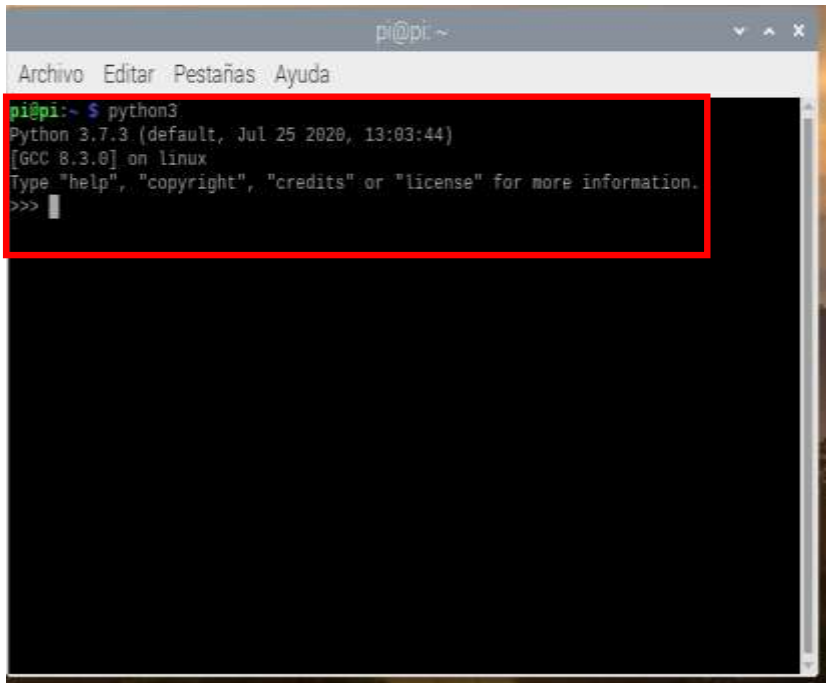
Finalizamos la instalación haciendo click en Finish

Una vez que se ha realizado la instalación vamos a conectar nuestra micro SD al computador por el puerto de tarjeta SD y abra Raspberry Pi Image

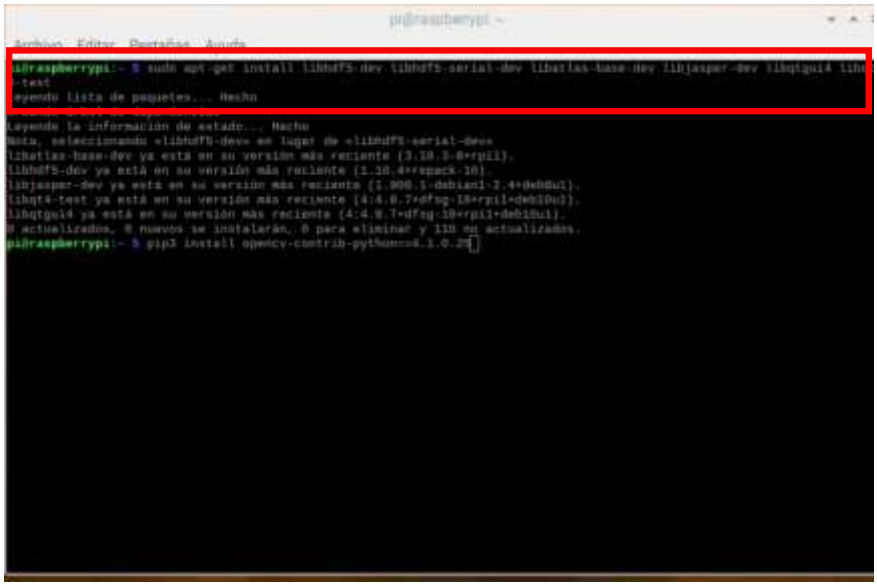
En el botón CHOOSE OS elegimos el sistema operativo acorde a las necesidades del proyecto en este caso seleccione la primera opción



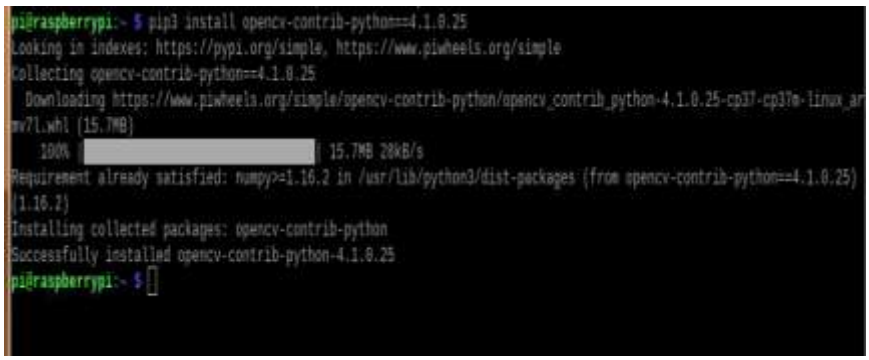
Anexo B instalación de dependencia para el reconocimiento facial



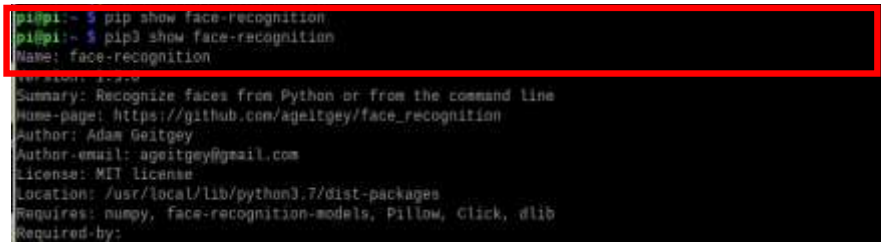
Previo a la instalación de Opencv en Python 3, necesitamos instalar algunos paquetes



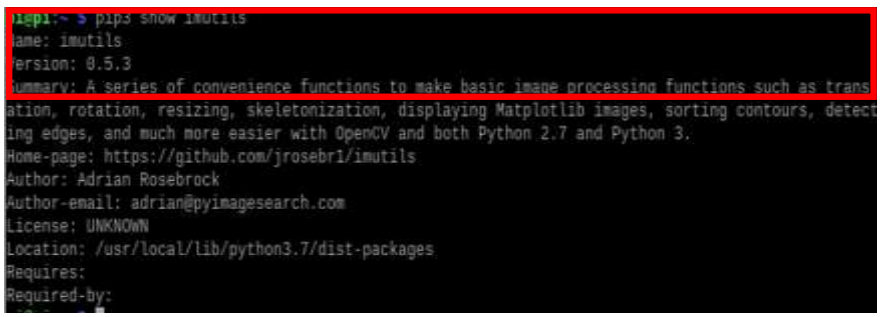
Instalación de la versión de Opencv 4.1.0.25



Instalación de face-recognition



Instalación de imutils



Anexo C Script para crear la base de Datos y Entrenamiento

```
# -*- coding: utf-8 -*-
```

```
"""
Created on Tue Mar  9 00:09:10 2021

@author: david
"""
from tkinter import *
from tkinter import filedialog
from PIL import Image
from PIL import ImageTk
import cv2
import imutils
import os
from tkinter import messagebox
import face_recognition
"""Intanciacion"""
face_cascade=cv2.CascadeClassifier('haarcascade_frontalface_default.xml')#
"""Variables"""
#cap=None
"""Funciones"""
def captura_rostro():
    ruta_data='./Data'#ruta de la base de datos de las imagenes de cada usuario
    nombre=user.get()#variable que especifica el nombre para cada usuario
    num_muestra=muestra.get()
    if nombre !="" and num_muestra >="":
        num_muestra=int(num_muestra)
        ruta_nombre=os.path.join(ruta_data,nombre)#join permite unir un directorio con un string para generar una ruta havcia los archivos
        if not os.path.exists(ruta_nombre):#en el caso de que la carpeta de nuevo usuario no este creada permite generarlo
            print("Creado carpeta",ruta_nombre)#imprimimos la ruta de la nueva carpeta genrada
            os.makedirs(ruta_nombre)#creacion de la nueva carpeta
        cap=cv2.VideoCapture(0,cv2.CAP_DSHOW)
        cont=0#inizializacion del contador de imagenes capturadas
```

while (cap.isOpened):#blucle infinito si se cumple con la condicion de que se inicializo la captura

ret,frame=cap.read()#lea cada capruea de fotogram en fotograma y delvuelve un True/False dependiendo si lee el marco corectamente

if ret==True:#si la varable ret es verdadera ejecuta la captura de rostros

#frame=cv2.flip(frame,1)

#frame=cv2.rotate(frame,cv2.ROTATE_90_COUNTERCLOCKWISE)#En el caso de video se hace una rotacion de 90 grado

dim=(300,400)#dimeciones de altura y ancho para cambia el tamano del fotograma

img=cv2.resize(frame,dim,interpolation=cv2.INTER_AREA)#cambiamos la dimenciones del fotograma segun la dimensiones previamente establecidad

gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)#convertimos el fotro gram de formato RGB a un

auxFrame=img.copy()#realizamos una copia del fotograma

#frame=cv2.resize(frame,(350,400),interpolation=cv2.INTER_AREA)

#gray=cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)

#auxFrame=frame.copy()

faces=face_cascade.detectMultiScale(gray,1.3,5)#e almacenarán los puntos x, y, ancho y alto del rostro que ha sido detectado (esto lo obtendremos de la variable ‘face’ que aquí se ha utilizado)

#gray es la imagen donde va actuar el detector

#ScaleFactor 1.3 es el parametro especifica que tanto va a ser reducida la imagen para crear ina piramide imagenes

#MineNeighbos especifica el numero minimo de cuadros delimitadores o vecinos que debe tener un rostros para que sea detectado como tal

for (x,y,w,h) in faces:#extraemos los rectagulos que contiene los rostros detectados

print(x,y,w,h)#vizulzamoslos puntos

x-=20#aumentamos punto x

y-=20#aumentams el punto y

#cv2.rectangle(frame,(x,y),(x+w,y+h),(0,0,255),2)#dibuja un rectagulo al rededor del rostro detectado

cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),2)

```
                rostro=auxFrame[y:y+h+40,x:x+w+40]#optiene todos los puntos del rostro capturado

rostro=cv2.resize(rostro,(224,224),interpolation=cv2.INTER_CUBIC)#cambia la dimensiones a (224,224) para los rotros capturados

                cont=cont+1#contador aumenta en 1 en cada iteracion

                cv2.imwrite(ruta_nombre+'/{}{}.jpg'.format(nombre,cont),rostro) #guarda la imagen en la nueva capeta segun el usuario indicado

            cv2.imshow('video',img)

            #cv2.imshow('original',frame)#miestra la imagen que se capturado

            if cv2.waitKey(100)&0xFF == ord('q') or cont>=num_muestra:#detemina si el numero de muestras se realizo sale de bucle

                #la funcion waithey acepta el tiempo en milisegundos como argumento y en caso que se presione la tecl q saldra del ciclo

                break #sale del ciclo while

        else:

            break#sale del ciclo while

    cap.release()#cierra la captura de fotogramas

    cv2.destroyAllWindows()#destruye la ventana emergente

else:

    print("No ingreso el Usuario")

    info_user()


def salir():

    valor=messagebox.askokcancel("Salir","Desea Salir del Programa?")

    if valor == True:

        root.destroy()

def info_user():

    messagebox.showinfo("Falta de Datos","No a Ingesado ningun usuario")


def entrenar():

print("Empezando el entrenamiento")

path="./Data"#ruta donde se encuentra la base de datos de imagenes de cada uno de los usuarios

img_paths=list(paths.list_images(path))#cre una lista de la ruta donde se guardan las imagenes de cada usuario
```

Encoding=[]#inicializamos la varaoble encoding para al macenar la codificacion de cada usuario

Names=[]#se va agragando el nombre del usuario en esta lista vacia

for(i,img_path) in enumerate(img_paths):#enumera la ruta de cada imagen segun el usuario corespondiente

    name=img_path.split(os.path.sep)[-2]#devuelve el nombre de cada usuario

    print("[INFO] Usuario:{}".format(name),"imagen prosezando{}/{}".format(i+1,len(img_paths)))#imprimir el numero de usuarui segun cada iteracion

  #split divide la ruta segun el separador "/"

  #[2] toma el valor del usuario

  #os.path.sep: El carácter utilizado por el sistema operativo a SEP componentes de nombre de ruta arate. Esto es '/'para P OS IX y '\\'para Windows.

  #print(name)

  #cargar la entrada y conrvetir en un formato RGB to dlib ordenado RGB

image=cv2.imread(img_path)#le la imagen del usuario

rgb=cv2.cvtColor(image,cv2.COLOR_BGR2RGB)#cambia el formato de la imagen BGR a RGB

boxes=face_recognition.face_locations(rgb,model='hog')#Encuentra todas las caras en la imagen usando el modelo predeterminado basado en HOG.

encodings=face_recognition.face_encodings(rgb)#Dada una imagen, devuelve la codificación de caras de 128 dimensiones para cada cara de la imagen.

for encoding in encodings:

    Encoding.append(encoding)#anade a la lista en Encoding

    Names.append(name)#anade el nombre del usuaria a la lista Names

print("[INFO] serializing encodings…")

data = {"encodings": Encoding, "names": Names}#serealiza cada usuario con la codificaion optenida

f = open("encodings.pickle", "wb")#función abre un archivo y lo devuelve como un objeto de archivo.

f.write(pickle.dumps(data))#Escriba la representación encurtida del objeto encondings en el archivo de objeto de archivo abierto

f.close()#cierra el nuevo archivo

"""GUI ----------"""

root=Tk()

'''Varaibles'''

```
user=StringVar()
muestra=StringVar()
"""logo"""
imagen=PhotoImage(file="uta.png")
img=Label(root,image=imagen)
img.grid(row=0,column=0,columnspan=20)
"""Label para control"""
label_info=Label(root, text="Video de Captura", font="bold")
label_info.grid(row=1,column=0,columnspan=2)
"""Usuario"""
nom=Label(root,text="Usuario",font="bold")
nom.grid(row=2,column=0,padx=10,pady=10)
#nom.config(fg="red")
nom_entry=Entry(root,textvariable=user)
nom_entry.grid(row=2,column=1,padx=10,pady=10)
"""Muestras"""
muestra_label=Label(root,text="N. Muestras",font="bold")
muestra_label.grid(row=3,column=0,padx=10,pady=10)
#nom.config(fg="red")
muestra_entry=Entry(root,textvariable=muestra)
muestra_entry.grid(row=3,column=1,padx=10,pady=10)
"""Boton Inicio"""
boton_on=Button(root,text="Iniciar Captura",command=captura_rostro)
boton_on.grid(row=4,column=0,padx=10,pady=10)
"""Boton de entrenamient"""
boton_on=Button(root,text="Iniciar Entrenameiento",command=entrenar)
boton_on.grid(row=5,column=0,padx=10,pady=10)
"""Boton de salida"""
boton_exit=Button(root,text="Salir",command=salir)
boton_exit.grid(row=6,column=0,padx=10,pady=10)
root.mainloop()
```

Anexo D El script para entrenamiento del reconocimiento facial

```
print('[Info]Inicio del Dispositivo IoT')
```

```
from tkinter import *
from tkinter import filedialog
from tkinter import messagebox
import cv2
import face_recognition
import pickle
import time
from web3 import Web3, HTTPProvider
from vyper import compile_codes
import numpy as np
import RPi.GPIO as GPIO
"""Pines Rasberry Pi"""
GPIO.setwarnings(False)#
GPIO.setmode(GPIO.BCM)
GPIO.setup(17, GPIO.OUT)#declaracion del pin 17 activacion del Rele
GPIO.setup(27, GPIO.OUT)#declaracion del pin 27 Led Indicador
GPIO.output(27,True)
GPIO.output(17,True)
print("[INFO] Cargando el contrato")
contract_source_code = '''
#estructuras
data: string[100]#variable que identifica el usuario que ingreso a la Smart Home
ids: int128[5]#Matriz de cinco dispositivos IoT que se puden conectar
idIndexs: int128#identificador de la matriz para los dispositivos IoT
auntentificado:string[100]#varable que autenfica al usuario que ingreso

@public
def addDispositivo(id:int128):
self.ids[self.idIndexs]=id #identificador del dispositivo
self.idIndexs +=1 #numero dispositivos conectados

@public
def autentificacion(_user: string[100]):
```

```
self.data= _user
if self.ids[0]==28:#si el dispositivo conincide con el numero asinado permitira autenficar al usuarui
    if self.data=="david":#si el usuario es david permitira que se abra la puerta
        self.auntentificado="user1"#asignara  a la variable user 1
    elif self.data=="martha":#si  el usuario es martha  permitira autentificar a este usuario
        self.auntentificado="user2"#asinara  la variable user2
    elif self.data=="oswaldo":#si el usuario es oswaldo autenficara a este usuario
        self.auntentificado="user3"#asignara  la varaible user3
    elif self.data=="edisson":
        self.auntentificado="user4"
    elif self.data=="alberto":
        self.auntentificado="user5"
    elif self.data=="santiago":
        self.auntentificado="user6"
    elif self.data=="roberto":
        self.auntentificado="user7"
    elif self.data=="christian":
        self.auntentificado="user8"
    elif self.data=="micaela":
        self.auntentificado="user9"
    elif self.data=="ricardo":
        self.auntentificado="user10"
    else:
        self.auntentificado="No  User"
else:
    self.auntentificado="Dispositivo  Invalido"#en caso de conplir la condicion la variable tomara el valor No User
@public
@constant
def get_data() -> string[100]:#funcion  para verificar el usuario enviado
return self.data#retona la variable user
@public
```

@constant

def get_auten() -> string[100]:#funcion para saber el usuario atentificado

return self.auntentificado

@public

@constant

def get_numdis() -> int128[5]:#funcion para saber las variables de los dispositivos conectados

return self.ids

@public

@constant

def get_dispositivo(id:int128) -> int128:#funcion para saber las variables de los dispositivos conectados

return self.ids[id]

@public

@constant

def get_numdisp() -> int128:#funcion para ver el numero de dispositivos conectados

return self.idIndexs

'''

smart_contract = {}

smart_contract['smart_home'] = contract_source_code

format = ['abi','bytecode']

compiled_code = compile_codes(smart_contract, format, 'dict')

abi = compiled_code['smart_home']['abi']

"""Autentificacion del dispositivo IoT"""

w3 = Web3(HTTPProvider('HTTP://192.168.0.106:7545'))#El proveedor es cómo web3 se comunica con la cadena de bloques. Los proveedores toman solicitudes JSON-RPC y devuelven la respuesta. Esto se hace normalmente enviando la solicitud a un servidor basado en socket HTTP o IPC.

#Este proveedor maneja interacciones con un servidor JSON-RPC basado en HTTP o HTTPS.

address ='0x405bd1FCf1053586b5E6909F6bb49099CA42A993'#direccion del contrato

private_key ='183d50a0de2948e3d5b88ec9790de09984db22f9c5e87e8f2b50ce7e5cc1b978'# clave privada de la cuenta

w3.eth.defaultAccount = '0x6002ac604B9c943a303B05765E4C9AF707b52EDb'

```
Smart = w3.eth.contract(address=address, abi=abi)
#proporciona una interfaz predeterminada para implementar e interactuar con contratos inteligentes de Ethereum
#Si addressse proporciona, este método devolverá una instancia del contrato definido por abi.
print("[INFO]Validando el dispositivo IoT")
aux_Iot=Smart.functions.get_dispositivo(0).call()
ident_Iot=28
if aux_Iot!=ident_Iot:
    nonce = w3.eth.getTransactionCount(w3.eth.defaultAccount)
    txn = Smart.functions.addDispositivo(ident_Iot).buildTransaction({
            'gas': 70000,
            'gasPrice': w3.toWei('1', 'gwei'),
            'nonce': nonce
    })
    signed_txn = w3.eth.account.signTransaction(txn, private_key=private_key)
#Devuelve una transacción que ha sido firmada por la clave privada del nodo, pero que aún no se ha enviado.
    signed_txn_hash = w3.eth.sendRawTransaction(signed_txn.rawTransaction)
#Envía una transacción firmada y serializada. Devuelve el hash de la transacción como un objeto HexBytes.
    w3.eth.waitForTransactionReceipt(signed_txn_hash)
#Espera a que la transacción especificada por transaction_hashse incluya en un bloque y luego devuelve el recibo de la transacción.
else:
    print("[INFO]Dispositivo IoT Ya esta Anadido")
"""Variables"""
nombre_actual='desconocido'
prevTime=0
doorUnlock=False
#count_dect=0#contador de fotogrmas detectados
count_reco=0#contador de rostros reconocidos
tolerancia=0.5
#muestras=0#numero de muestras
puerta=0
```

```
#Num_larencia=[]
#Num_bloque=[]
print("[INFO] Algoritmos y Codificacion")
"""Algoritmos"""
encodings="encodings.pickle"
cascade="haarcascade_frontalface_default.xml"
data=pickle.loads(open(encodings,"rb").read())
detector=cv2.CascadeClassifier(cascade)
"""funciones"""
def detec(img):
face=detector.detectMultiScale(img,1.1,4,minSize=(150,150))
boxes=[(y,x+w,y+h,x)  for (x,y,w,h) in face]#convertimos en Una lista de tuplas de ubicaciones de caras encontradas en orden css (superior, derecha, inferior, izquierda)
return boxes##top, right, bottom, left
def reconocimiento(rgb,boxes):
global nombre_actual,count_reco#para  cambiar el valor de una variable local se utiliza la palabra clave global
encodings=face_recognition.face_encodings(rgb,boxes)#Una lista de codificaciones de caras de 128 dimensiones (una para cada cara de la imagen)
names=[]
distancia=[]
dis_minima=[]
for encoding in encodings:

    matches=face_recognition.compare_faces(data["encodings"],encoding,tolerancia)#Dada una imagen, devuelve la codificación de caras de 128 dimensiones para cada cara de la imagen.

    face_distance=face_recognition.face_distance(data["encodings"],encoding)#genera un lista de la distacia de cada uno de los usuarios de la base e datos codificados con respecto al rostro reconocido
    dis_min=face_distance[np.argmin(face_distance)]#obtenemos la distancia minima
    name="desconocido"
    if True in matches:#ingresa la sentencia if si existe un True en la lista de matches
```

```
                matchedIdxs=[i  for (i,b) in enumerate(matches)  if b]#
                counts={}# inicilizamos el diccionario que almacena el nombre del usuario reconocido como clave
                for i in matchedIdxs:#
                        name=data["names"][i]
                        counts[name]=counts.get(name,0)+1
                        name=max(counts,key=counts.get)
                if nombre_actual  !=name:
                        nombre_actual=names
        names.append(name)
        distancia.append(face_distance)
        dis_minima.append(dis_min)
return names,distancia,dis_minima
def transacion(nombre):
    nom=nombre[0]
    nonce = w3.eth.getTransactionCount(w3.eth.defaultAccount)
    txn = Smart.functions.autentificacion(nom).buildTransaction({
                'gas': 120280,
                'gasPrice': w3.toWei('1', 'gwei'),
                'nonce': nonce
    })
    #Crea un diccionario de transacciones basado en la llamada a la función de contrato especificada.
    signed_txn = w3.eth.account.signTransaction(txn,  private_key=private_key)
    #Devuelve una transacción que ha sido firmada por la clave privada del nodo, pero que aún no se ha enviado.
    signed_txn_hash  = w3.eth.sendRawTransaction(signed_txn.rawTransaction)
    #Envía una transacción firmada y serializada. Devuelve el hash de la transacción como un objeto HexBytes.
    w3.eth.waitForTransactionReceipt(signed_txn_hash)
    #Espera a que la transacción especificada por transaction_hashse incluya en un bloque y luego  devuelve el recibo de la transacción.
    auto=Smart.functions.get_auten().call()
    #Llame a una función de contrato, ejecutando la transacción localmente usando la eth_callAPI. Esto no creará una nueva transacción pública.
```

```
    return auto
def VideoStream():
#Num_larencia=[]
#Num_bloque=[]
global doorUnlock,prevTime,puerta
print("[INFO] Iniciando Video Stream")
cap=cv2.VideoCapture(0)#instanciamos el objeto VideoCapture
while(cap.isOpened()):
        ret,frame=cap.read()
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)#cambia el fotogram a un formato en escala de frises
        rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)#cambuamos el fotograma de un formato BGR A RGB
        if detec(gray):#si detecta un rostro ingresa a la sentencia
                boxes=detec(gray)#almacenamos la lista de coordenas retornadas por la funcion detec

        nombres,dis_list,dis_min=reconocimiento(rgb,boxes)#almacenamos el nombre que retorna la funcion reconocimiento
                porcentaje=(((1-dis_min[0])*100)/(1-0.4))
                if nombres[0] !="desconocido":#ingresa a la sentencia si el nombre reconocido es diferente a 'desconocido'
                        doorUnlock=True
                        prevTime=time.time()
                for ((top,right,bottom,left),name,dis)in zip(boxes,nombres,dis_min):
                        cv2.rectangle(frame,(left,top),(right,bottom),(0,255,0),2)
                        y=top-15 if top-15> 15 else top+15
                        por=((1-dis)*100)/(1-0.4)
                        por=(round(por,2))

        cv2.putText(frame,name,(left,y),cv2.FONT_HERSHEY_SIMPLEX,.8,(255,0,0),2)
                        cv2.putText(frame,str(por),(right-45,y),cv2.FONT_HERSHEY_SIMPLEX,.8,(255,0,0),2)
                cv2.imshow("Rostro",frame)
```

```
                if w3.isConnected()==True and doorUnlock ==True and nombres[0]!='desconocido':
                        if puerta==0:
                                print("[INFO] Conectado a la Red Blockchain")
                                print("[INFO] Usuario Identificado: ",nombres[0])
                                usuario.set(nombres[0])
                                print("[INFO] Distancia Minima: ",round(dis_min[0],2))
                                dis_min_var.set(str(round(dis_min[0],2)))
                                por_var.set(str(round(porcentaje,2)))
                                inicio_time=time.time()#iniciamos el tiempo para medir la latencia
                                autorizacion=transacion(nombres)#guardamos los datos Obtenidos del Contrato inteligente en la Red Blockchain
                                final_time=time.time()#
                                latencia=(final_time-inicio_time)#calculamos el tiempo de ejcucion del la transaccion relizada
                                print("[INFO] latencia de transaccion ",latencia)
                                latencia_var.set(round(latencia,2))#enviamos la latencia de lacerancion del nuevo blque y la respuesta del contrato inteligente

        usurio_autentificado=Smart.functions.get_data().call()

        local_time=time.asctime(time.localtime(final_time))#guardamos la fecha en la que se relizo la transaccion
                                for (top,right,buttom,left) in boxes:#graficamos la fecha en la que se genero el nuevo bloque
                                        x=left
                                        y=bottom+20

        cv2.putText(frame,local_time,(x,y),cv2.FONT_HERSHEY_SIMPLEX,.8, (255, 0, 0), 2)
                                cv2.imshow("Rostro", frame)
                                time.sleep(0.75)
                                if autorizacion in ('user1','user2','user3','user4','user5','user6','user7','user8','user9','user10'):
```

```
                                        print("[INFO] Autentificacion: ",autorizacion)#usurio atentificado
                                        auten_var.set(autorizacion)
                                        #muestras+=1

        bloque=w3.eth.getTransactionCount(w3.eth.defaultAccount)#bloque generado
                                        print("[INFO] Numero de bloque ",bloque)
                                        bloque_var.set(bloque)
                                        fecha.set(local_time)
                                        #print("[INFO] Numero de muestra ",muestras)
                #Num_latencia.append(latencia)
                #Num_bloque.append(bloque)
                                        print("[INFO] Puerta Abierta")
                                        puerta_var.set("Abierta")
                                        #GPIO.output(17,False)
                #GPIO.output(27,False)
                                        puerta=1
                                else:
                                        if puerta==0:
                                                print("[INFO]Usuario No Autentificado ",usurio_autentificado)
                                                print("[INFO]No Autorizado ",autorizacion)
                #else:
                        if puerta==0:
                                print("/n")
                                print("[INFO]No Transacion")
                                print("[INFO]No Persona no autorizada")
                                print("[INFO]Distancia Minima ",round(dis_min[0],2))
                                try:
                                        user_cercano= data["names"][np.argmin(dis_list)]
```

```
                                usuario.set(user_cercano)
                                print("[INFO]Usuario Semejante ",user_cercano)
                        except :
                                pass
                        print("[INFO]Porcentaje de Semejanza ",round(porcentaje,2))
                        dis_min_var.set(round(dis_min[0],2))
        if doorUnlock==True and time.time()-prevTime >5:
                print("[INFO] Espera 10 segundos")
                time.sleep(10)
                doorUnlock=False
                #GPIO.output(17,True)
        #GPIO.output(27,True)
                print("[INFO] Puerta cerrada")
                puerta_var.set("Cerrada")
                usuario.set("")
                latencia_var.set("")
                bloque_var.set("")
                auten_var.set("")
                fecha.set("")
                puerta=0
        cv2.imshow("Rostro",frame)
        if puerta==0:
                #usuario.set("")
                latencia_var.set("")
                bloque_var.set("")
                auten_var.set("")
                fecha.set("")
                puerta_var.set("")
                dis_min_var.set("")
                por_var.set("")

        try:
```

```
                root.update()
        except :
                pass
        key=cv2.waitKey(1)&0xFF
        if key == ord("q"):
                break
cap.release()
cv2.destroyAllWindows()
   #GPIO.cleanup()
def salir():
   valor=messagebox.askokcancel("Salir","Desea Salir del Programa?")
   if valor == True:
        try:
                cap.release()
                cv2.destroyAllWindows()
        except :
                print("No ha inicado Video Stream")
        root.destroy()
#Gui----------------------------
root=Tk()
"""varables"""
usuario=StringVar()
latencia_var=StringVar()
bloque_var=StringVar()
auten_var=StringVar()
fecha=StringVar()
puerta_var=StringVar()
dis_min_var=StringVar()
por_var=StringVar()
"""Wigets"""
"""logo"""
imagen=PhotoImage(file="uta.png")
img=Label(root,image=imagen)
```

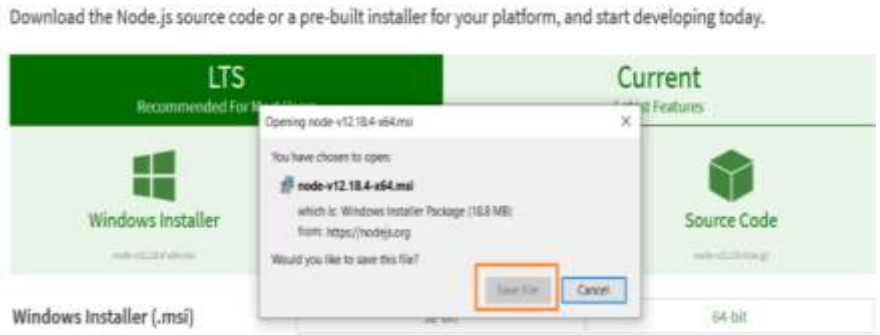
```
img.grid(row=0,column=0,columnspan=20)
"""datos de usuario"""
reconocimiento_label=Label(root,text="Datos del Reconocimiento Facial",font="bold")
reconocimiento_label.grid(row=1,column=1,padx=10,pady=10)
usuario_label=Label(root,text="Usuario Identificado",font="bold")
usuario_label.grid(row=2,column=0,padx=10,pady=10)
usuario_entry=Entry(root,textvariable=usuario)
usuario_entry.grid(row=2,column=1,padx=10,pady=10)
puerta_label=Label(root,text="Estado de la Puerta",font="bold")
puerta_label.grid(row=3,column=0,padx=10,pady=10)
puerta_entry=Entry(root,textvariable=puerta_var)
puerta_entry.grid(row=3,column=1,padx=10,pady=10)
puerta_label=Label(root,text="Distancia Euclidiana Mininima",font="bold")
puerta_label.grid(row=4,column=0,padx=10,pady=10)
puerta_entry=Entry(root,textvariable=dis_min_var)
puerta_entry.grid(row=4,column=1,padx=10,pady=10)
puerta_label=Label(root,text="Porcentaje Semejanza",font="bold")
puerta_label.grid(row=5,column=0,padx=10,pady=10)
puerta_entry=Entry(root,textvariable=por_var)
puerta_entry.grid(row=5,column=1,padx=10,pady=10)
"""Boton Inicio"""
boton_on=Button(root,text="Iniciar Captura",command=VideoStream)
boton_on.grid(row=6,column=0,padx=10,pady=10)
"""Boton de salida"""
boton_exit=Button(root,text="Salir",command=salir)
boton_exit.grid(row=7,column=0,padx=10,pady=10)
"""Datos blockchain----------------------------------------"""
blockchain_label=Label(root,text="Datos del la Red Blockchain",font="bold")
blockchain_label.grid(row=1,column=2,padx=10,pady=10)
user_aunte_label=Label(root,text="Autentificacion",font="bold")
user_aunte_label.grid(row=2,column=2,padx=10,pady=10)
user_aunte_entry=Entry(root,textvariable=auten_var)
user_aunte_entry.grid(row=2,column=3,padx=10,pady=10)
```

```
latencia_label=Label(root,text="Latencia(ml)",font="bold")
latencia_label.grid(row=3,column=2,padx=10,pady=10)
latencia_entry=Entry(root,textvariable=latencia_var)
latencia_entry.grid(row=3,column=3,padx=10,pady=10)
bloque_label=Label(root,text="Numero  de bloque",font="bold")
bloque_label.grid(row=4,column=2,padx=10,pady=10)
bloque_entry=Entry(root,textvariable=bloque_var)
bloque_entry.grid(row=4,column=3,padx=10,pady=10)
fecha_label=Label(root,text="Fecha  creacion del bloque",font="bold")
fecha_label.grid(row=5,column=2,padx=10,pady=10)
fecha_entry=Entry(root,textvariable=fecha)
fecha_entry.grid(row=5,column=3,padx=10,pady=10)
root.mainloop()
#print("Numero  Total Lantencia",Num_latencia)
#print("[INFO] LISTA DE DISTACIA EUCLIDIANA",distacias)
#print("Numero  Total de blqoues",Num_bloque)
```

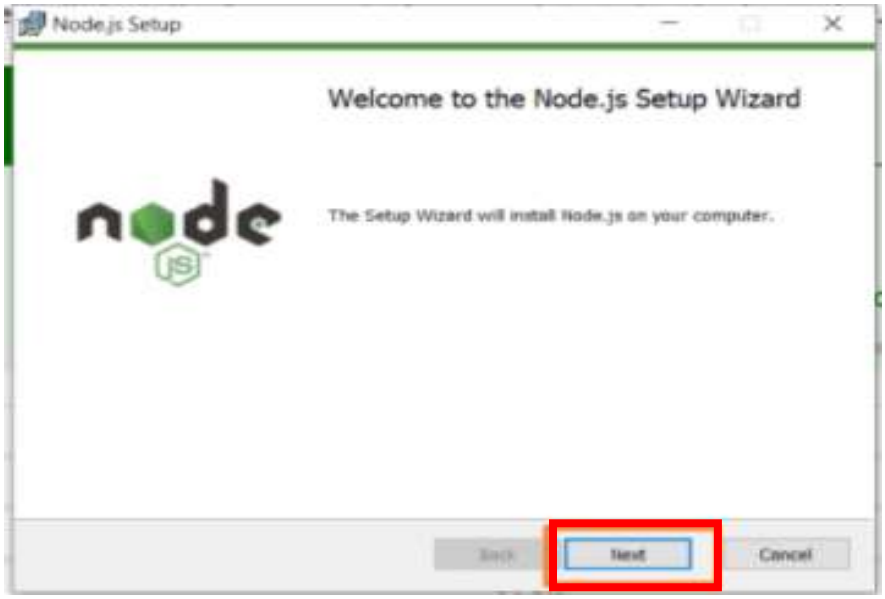
Anexo E Instalación de Node.js

Descargue el último paquete node.js del sitio oficial de node.js y haga clic en el instalador de Windows

guarde la configuración de msi de node.js descargada, haga clic en Guardar archivo**.**

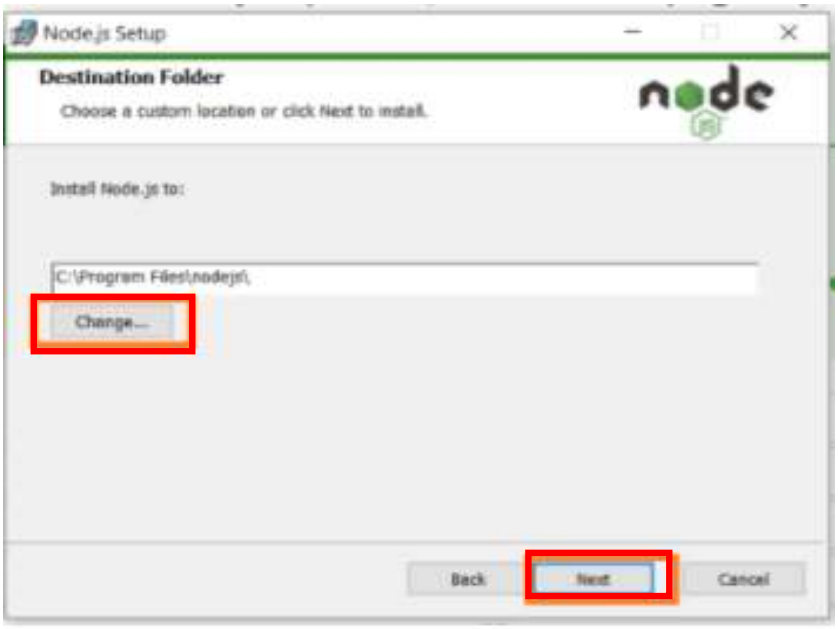


haga doble clic en node.js archivo .msi instalado en Windows y verá el Asistente de configuración de Node.js, haga clic en Siguiente como se muestra a continuación.



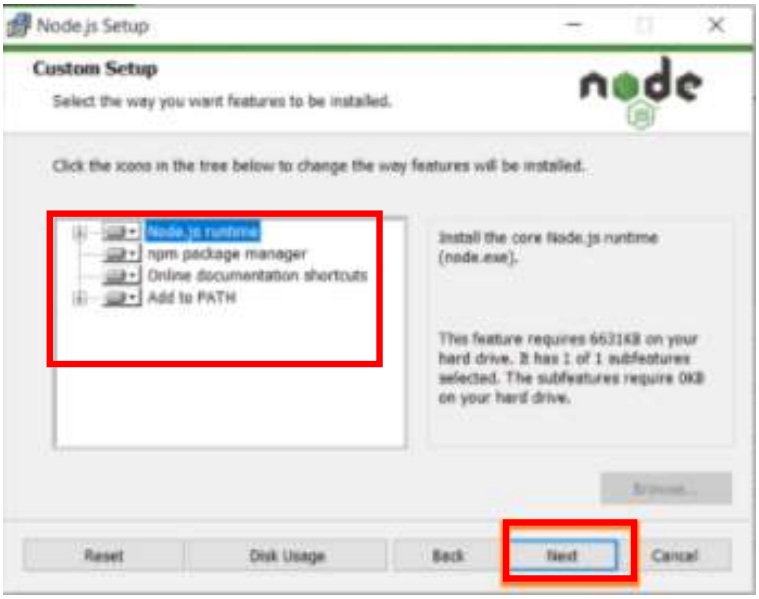
haga clic en **Acuerdo de licencia de Node.js** y haga clic en **Siguiente**.

la carpeta de destino donde desea instalar Node.js y haga clic en siguiente.



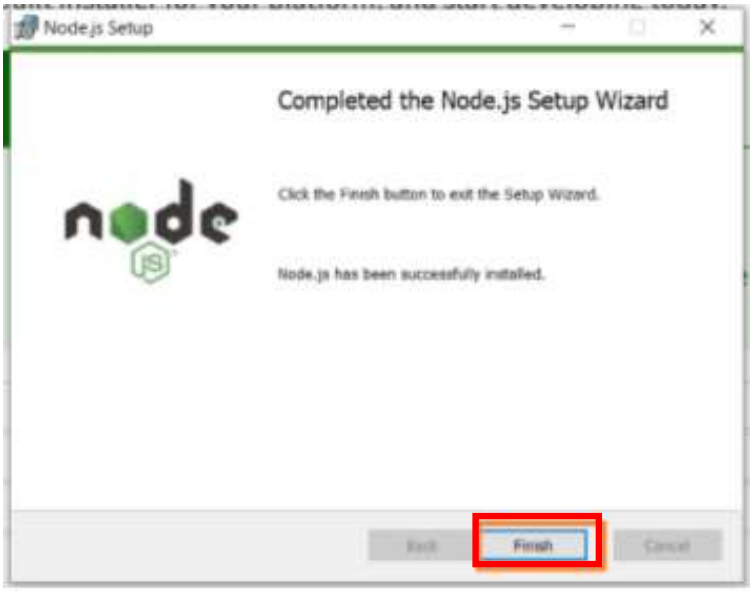
Seleccione para instalar módulos npm como python y Visual Studio Build Tools si no están instalados y haga clic en Siguiente

configuración personalizada para Node.js y haga clic en siguiente



Ahora instale Node.js en Windows 1

terminar para instalar Node.js en Windows.



Anexo F Contrato inteligente

#estructuras

data: string[100]#variable que identifica el usuario que ingreso a la Smart Home

ids: int128[5]#Matriz de cinco dispositivos IoT que se puden conectar

idIndexs: int128#identificador de la matriz para los dispositivos IoT

auntentificado:string[100]#varable que autenfica al usuario que ingreso

@public

def addDispositivo(id:int128):

self.ids[self.idIndexs]=id #identificador del dispositivo

self.idIndexs +=1 #numero dispositivos conectados

@public

def autentificacion(_user: string[100]):

self.data= _user

if self.ids[0]==28:#si el dispositivo conincide con el numero asinado permitira autenficar al usuarui

if self.data=="david":#si el usuario es david permitira que se abra la puerta

self.auntentificado="user1"#asignara a la variable user 1

elif self.data=="martha":#si el usuario es martha permitira autentificar a este usuario

self.auntentificado="user2"#asinara la variable user2

elif self.data=="oswaldo":#si el usuario es oswaldo autenficara a este usuario

```
            self.auntentificado="user3"#asignara  la varaible user3
        elif self.data=="edisson":
            self.auntentificado="user4"
        elif self.data=="alberto":
            self.auntentificado="user5"
        elif self.data=="santiago":
            self.auntentificado="user6"
        elif self.data=="roberto":
            self.auntentificado="user7"
        elif self.data=="christian":
            self.auntentificado="user8"
        elif self.data=="micaela":
            self.auntentificado="user9"
        elif self.data=="ricardo":
            self.auntentificado="user10"
        else:
            self.auntentificado="No  User"
else:
    self.auntentificado="Dispositivo  Invalido"#en caso de conplir la condicion la variable tomara el valor No User
@public
@constant
def get_data() -> string[100]:#funcion  para verificar el usuario enviado
return self.data#retona la variable user
@public
@constant
def get_auten()  -> string[100]:#funcion  para saber el usuario atentificado
return self.auntentificado
@public
@constant
def get_numdis()  -> int128[5]:#funcion para saber las variables de los dispositivos conectados
return self.ids
```

@public

@constant

def get_dispositivo(id:int128) -> int128:#funcion para saber las variables de los dispositivos conectados

return self.ids[id]

@public

@constant

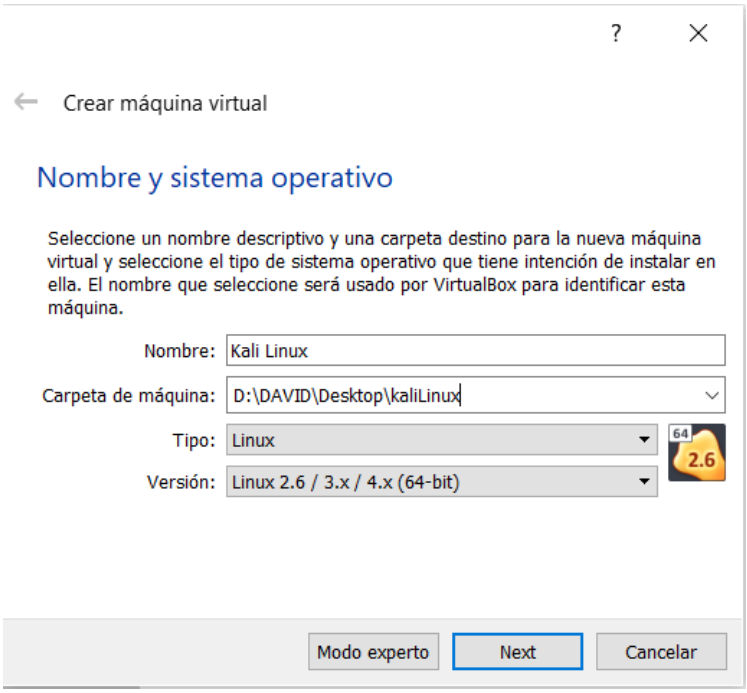
def get_numdisp() -> int128:#funcion para ver el numero de dispositivos conectados

return self.idIndexs

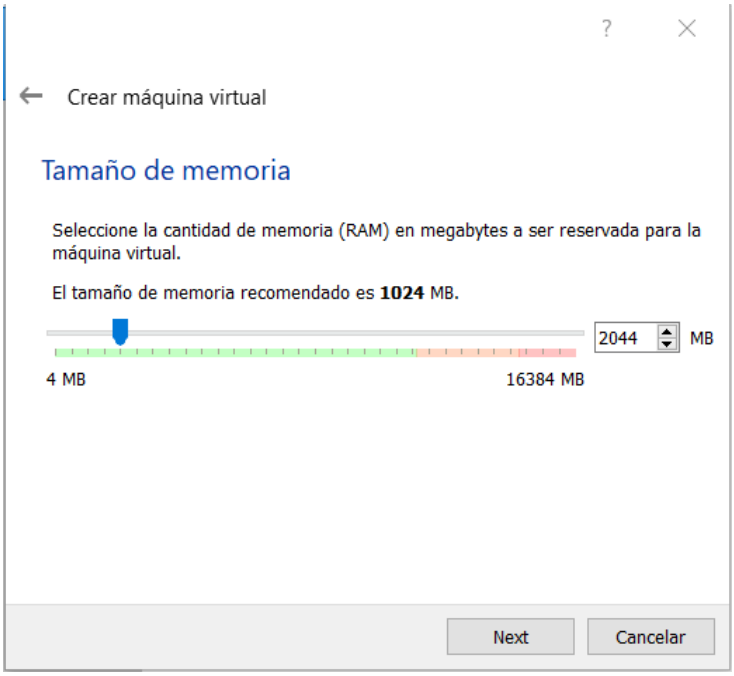
Anexo G instalación de Kali Linux

Descargue la imagen ISO de Kali Linux

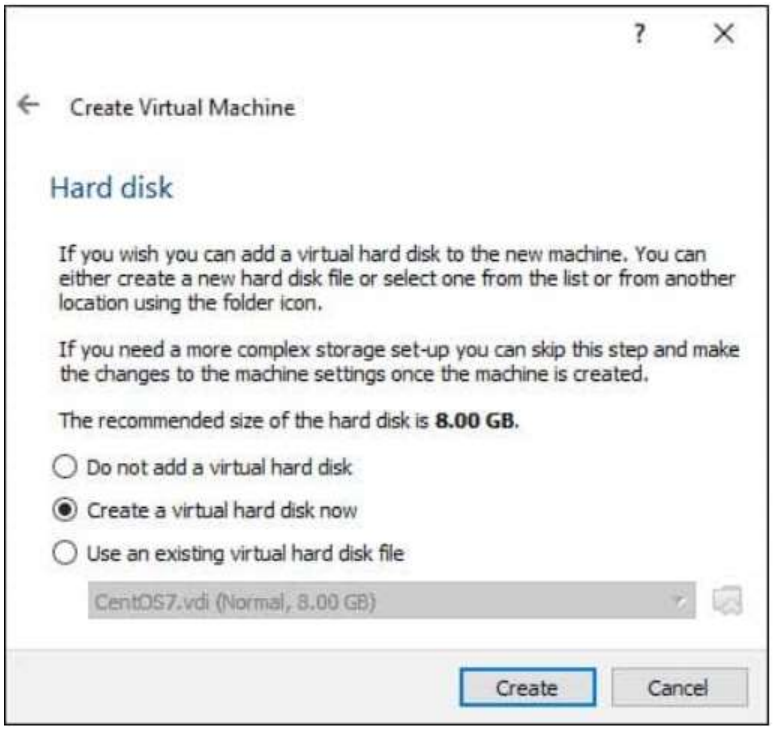
Cree el contenedor Kali Linux VirtualBox



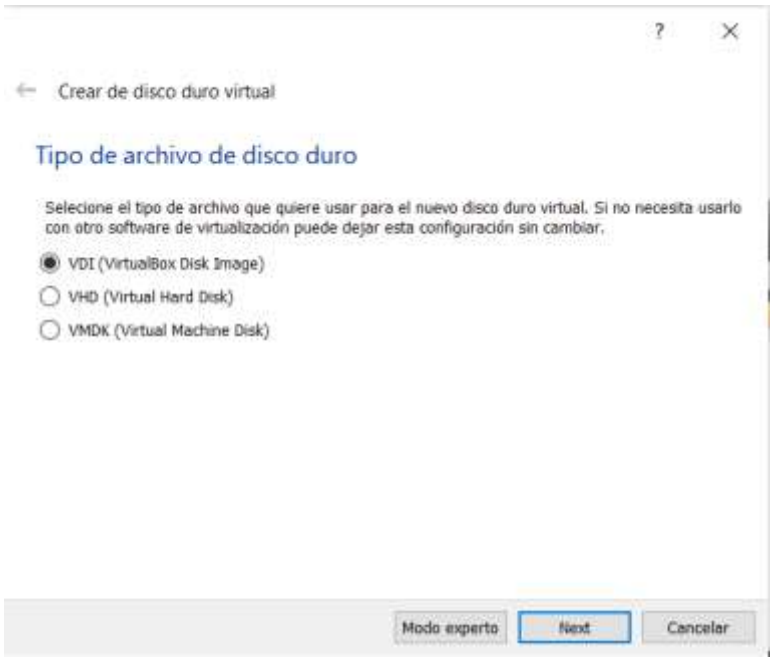
Elija cuánta memoria asignar a la máquina virtual y haga clic en siguiente



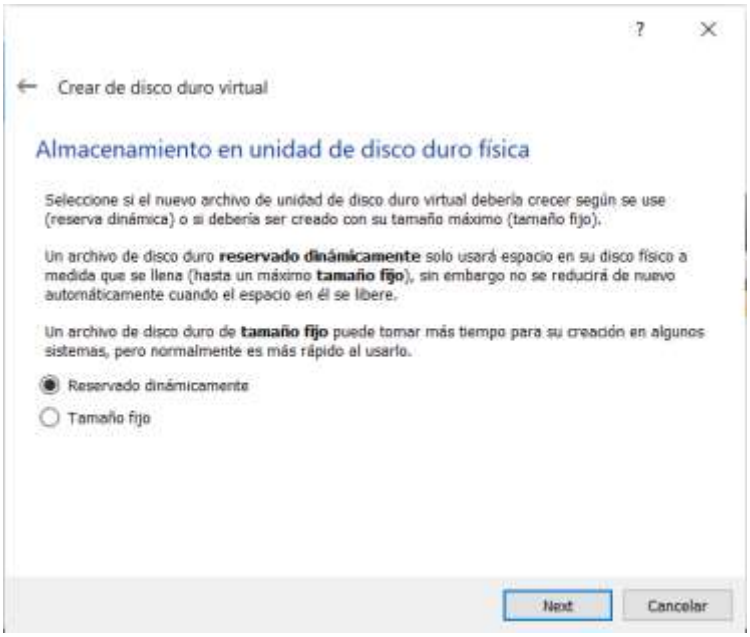
La opción predeterminada es crear un disco duro virtual para la nueva máquina virtual.



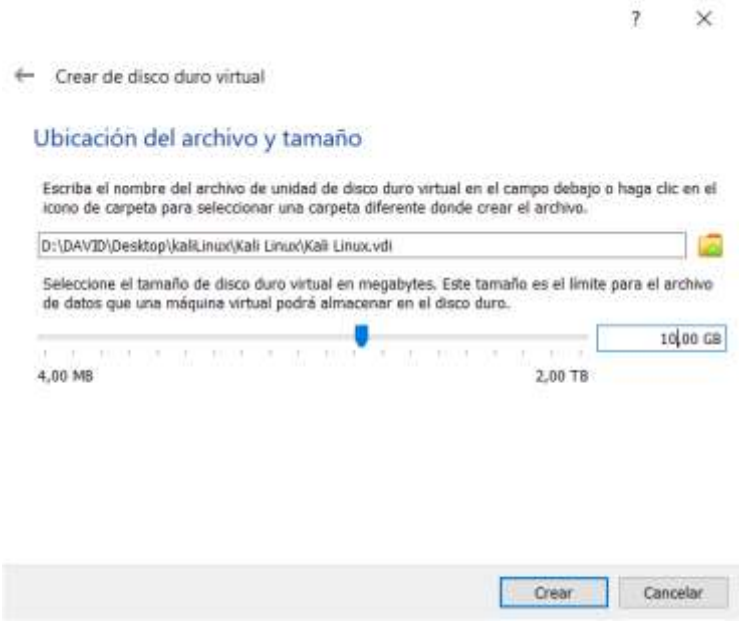
Siga el tipo de archivo predeterminado para el nuevo disco duro virtual, VDI **(** VirtualBox Disk Image **)**



Decida entre tamaño fijo y asignado dinámicamente**.**

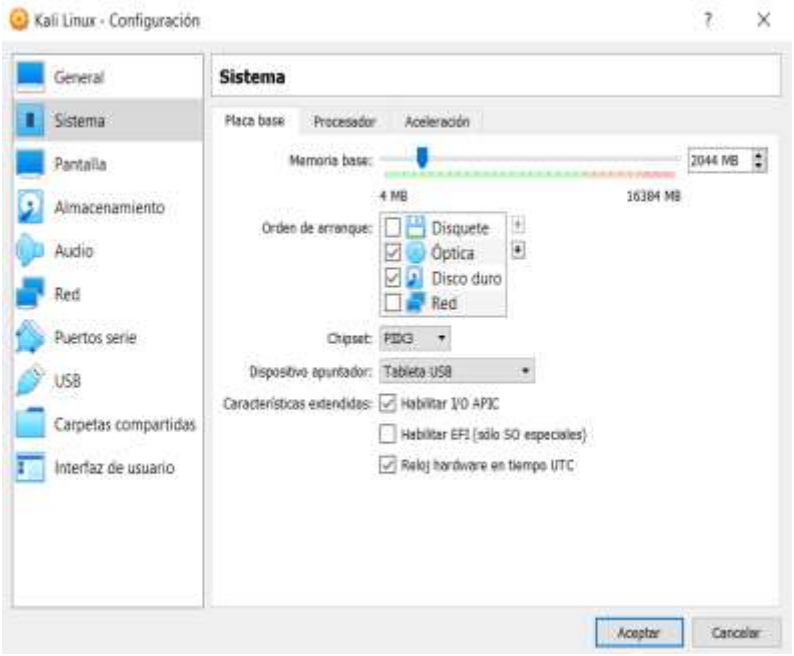


Ubicación y tamaño del archivo. Especifique el nombre y dónde desea almacenar el disco duro virtual.

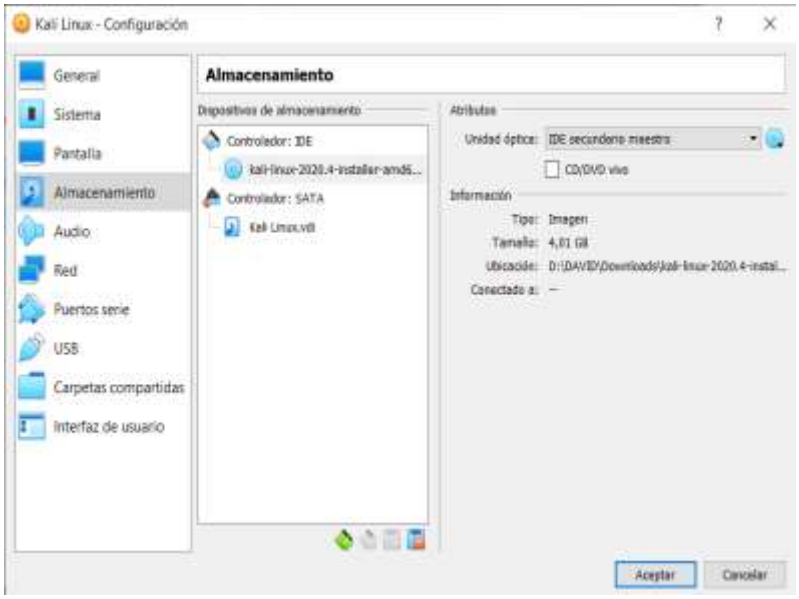


Configurar la máquina virtual

Establezca el orden de inicio para que comience

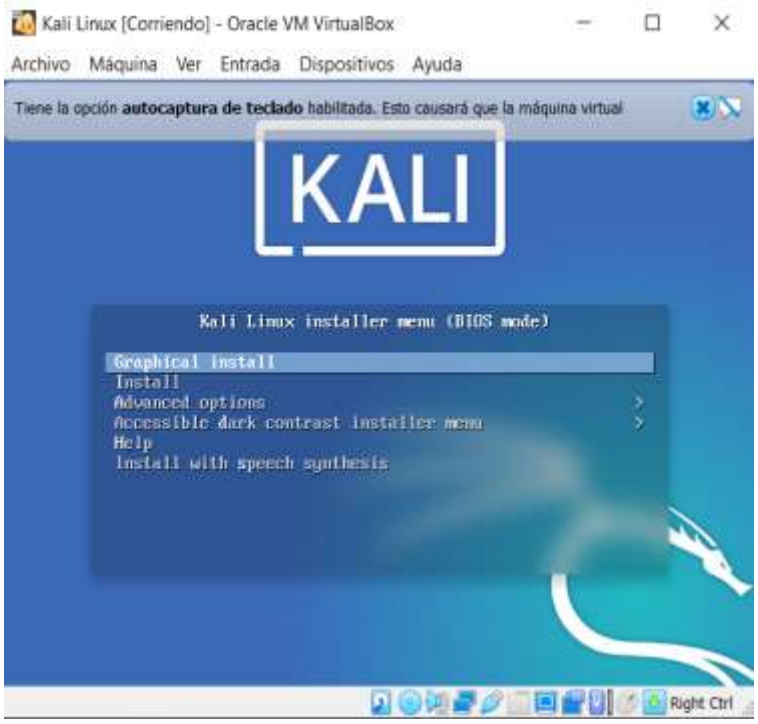


Agregue la imagen Kali descargada a un dispositivo de almacenamiento en Controlador:



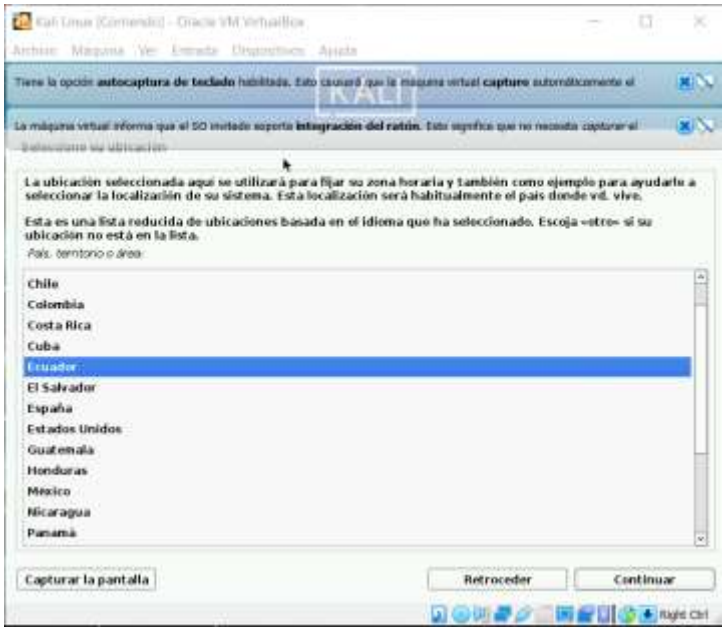
Haga clic en el icono Inicio para comenzar a instalar Kali.

Instalar y configurar Kali Linux



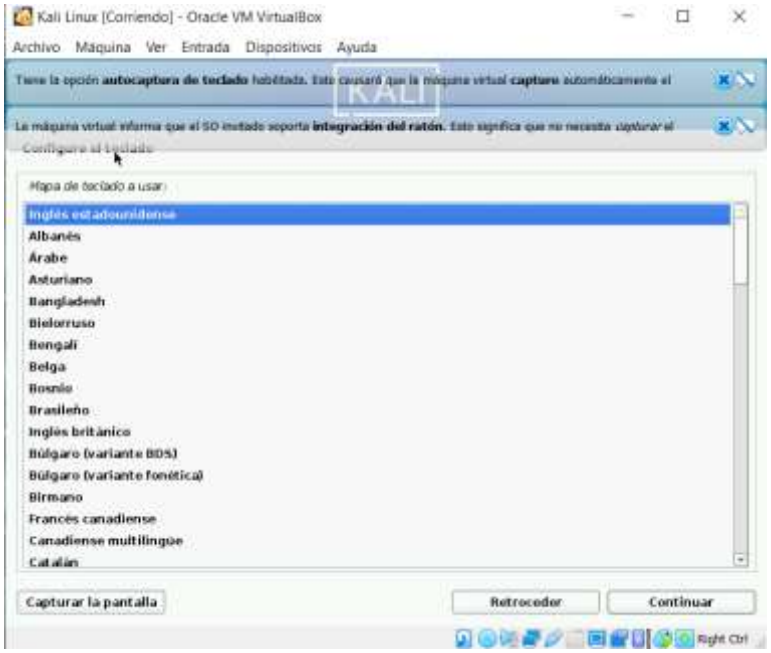
Seleccione un idioma

Seleccione su ubicación

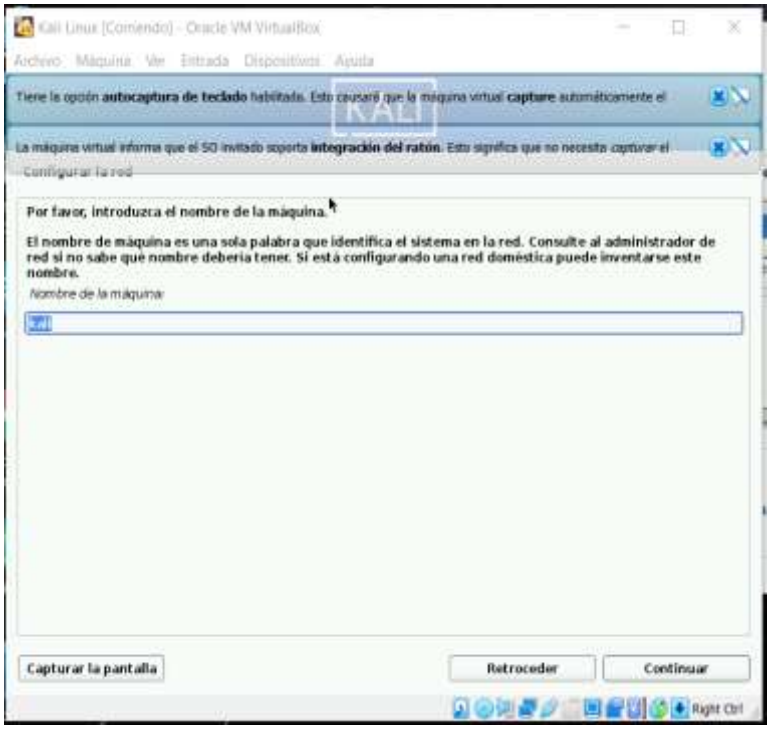


.

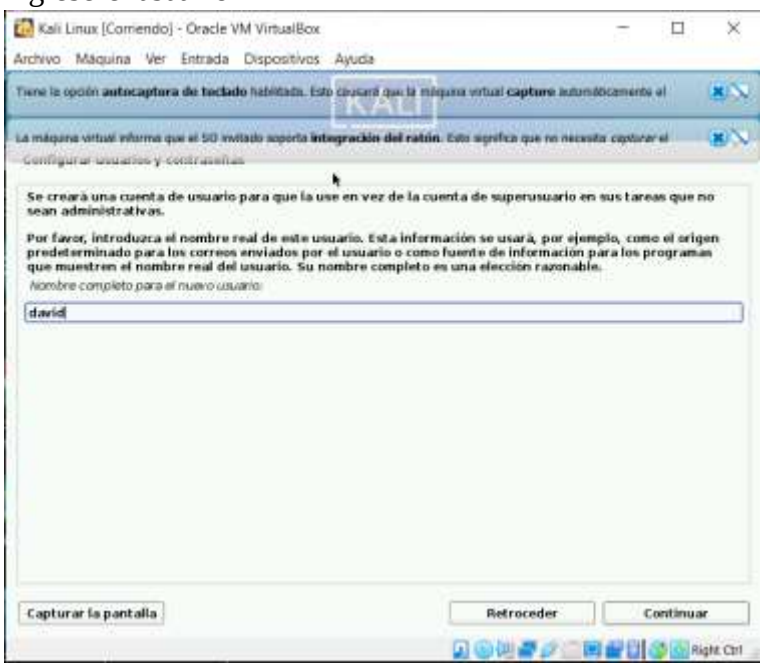
Configure el teclado.



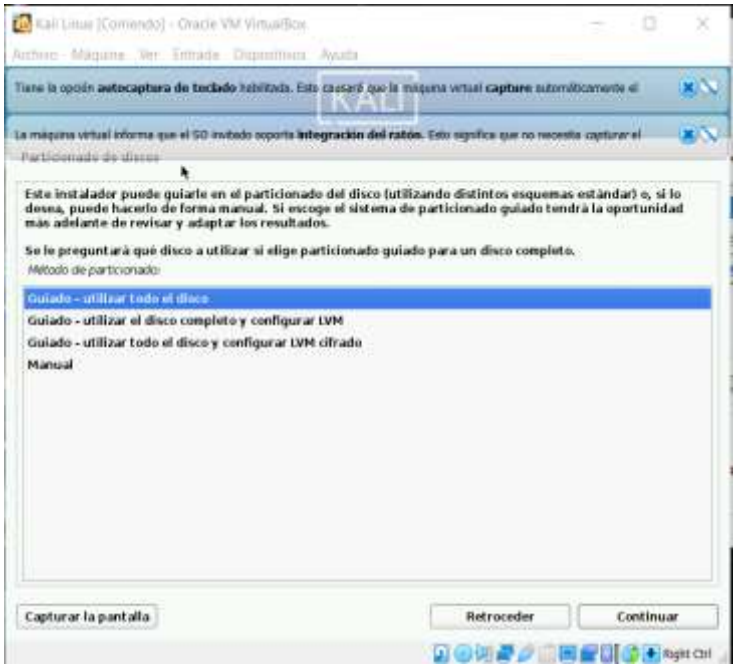
Configure la red**.**



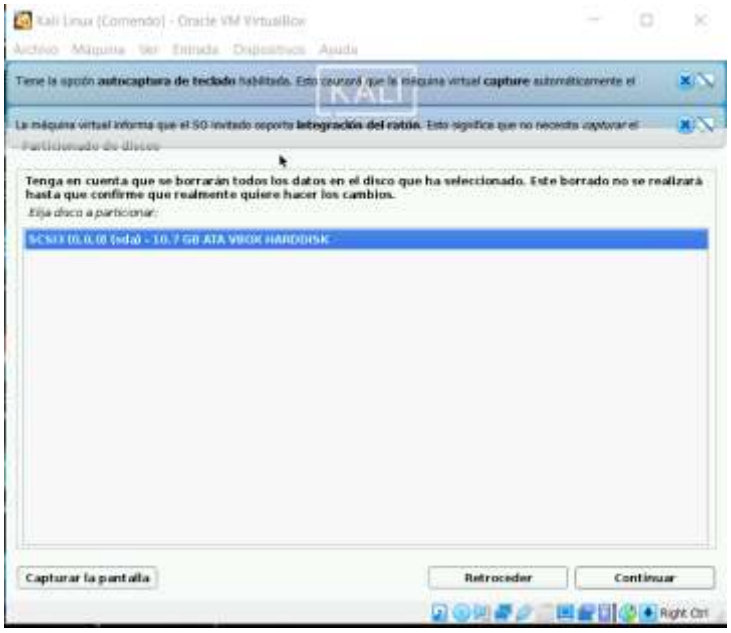
Ingrese el usuario



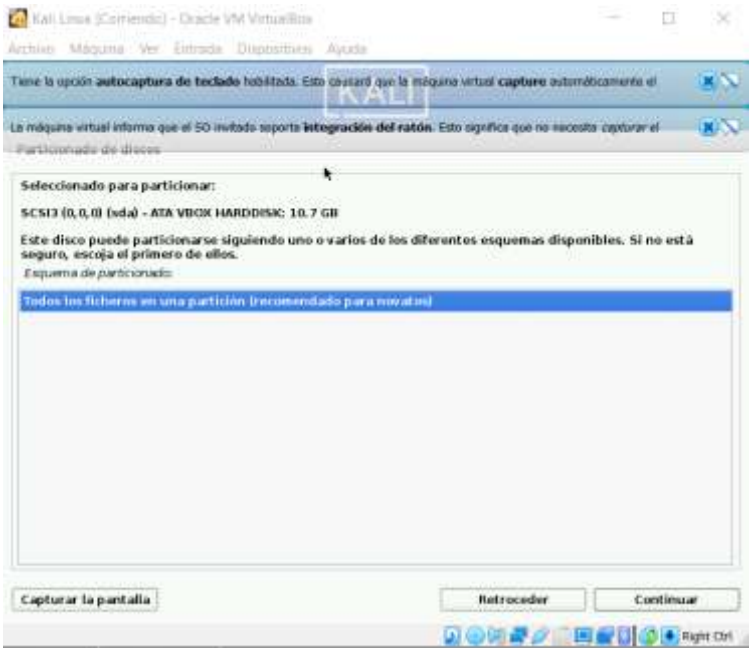
Discos de partición.



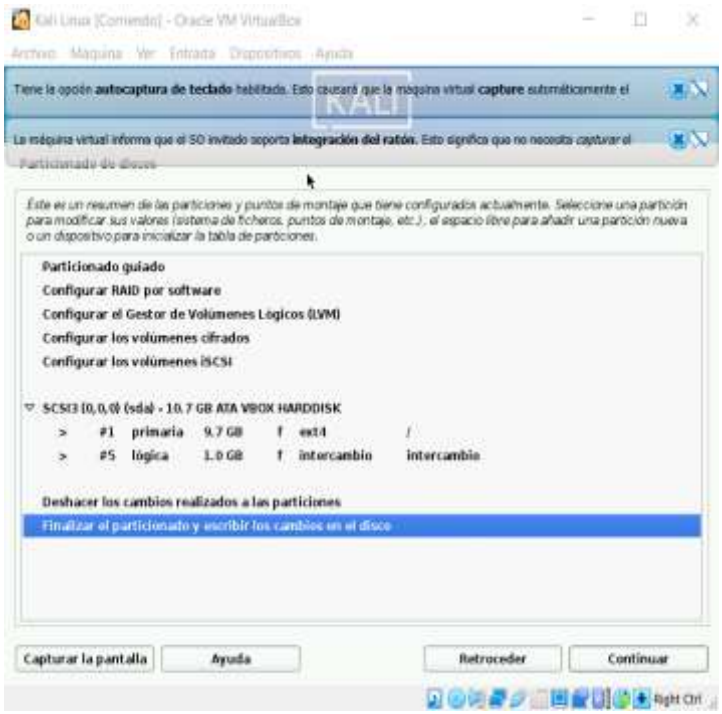
Luego, seleccione qué disco desea usar para particionar.



A continuación, seleccione el esquema de partición



El asistente le ofrece una descripción general de las particiones configuradas.



Anexo H Muestras de latencia del sistema

Usuario	Bloque	Respuesta de la transacción	Latencia(milisegundos)
David	753	user1	0,466
David	754	user1	0,424
David	755	user1	0,646
David	756	user1	0,442
David	757	user1	0,369
David	758	user1	0,4
David	759	user1	0,399
David	760	user1	0,276
David	761	user1	0,464
David	762	user1	0,445
David	763	user1	0,365
David	764	user1	0,318
David	765	user1	0,392
David	766	user1	0,478
David	767	user1	0,451
David	768	user1	0,471
David	769	user1	0,382
David	770	user1	0,429
David	771	user1	0,472
David	772	user1	0,38
David	773	user1	0,382
David	774	user1	0,393
David	775	user1	0,318
David	776	user1	0,462
David	777	user1	0,366
David	778	user1	0,43
David	779	user1	0,366
David	780	user1	0,428
David	781	user1	0,317
David	782	user1	0,431
David	783	user1	0,323
David	784	user1	0,476
David	785	user1	0,411
David	786	user1	0,381
David	787	user1	0,364
David	788	user1	0,397
David	789	user1	0,413
David	790	user1	0,398
David	791	user1	0,398
David	792	user1	0,363
David	793	user1	0,413
David	794	user1	0,354
David	795	user1	0,41

David	796	user1	0,416
David	797	user1	0,336
David	798	user1	0,355
David	799	user1	0,397
David	800	user1	0,395
David	801	user1	0,4174
David	802	user1	0,334
Promedio			0,4023

Anexo I Manual de Instalación

MANUAL DE USUARIO

Sistema Smart Home Aplicando IoT y Blockchain

23 DE MARZO DE 2021

UNIVERSIDAD TECNICA AMBATO

Ambato-Ecuador

Introducción

Este sistema se ha diseñado para como aplicación de Puerta Inteligentes con reconocimiento facial de los miembros de hogar con autentificación y almacenamiento de los datos de usuario con un contrato inteligente escrito en lenguaje de programación Vyper e implementado en cadena de Bloques Ethereum llamada Ganache

Requisito del Sistema

Los requisitos previos de software instalado para poder ejecutar programa son

-Node js

-Truffle

-Python

Los requisitos mínimos de hardware son lo siguientes

-Procesador Cortex-A72, Intel Core i7,

-4GB de RAM o superior

-Cámara 5MP o superior

Implementación del Sistema de reconocimiento facial

Componentes de hardware

- Rasberry pi 4
- Modulo cámara
- Cerradura de solenoide

1. **Creación de la base de datos**

   Pasos para creación de la base de datos de los usuarios

   1. Instale OpenCv con el siguiente comando :
      -pip3 install opencv-contrib-python==4.1.0.25
   2. Descargue los algoritmos pre entrenados del siguiente link: https://github.com/opencv/opencv/tree/master/data/haarcascades
   3. Instale la librería face_reconition.py
   4. Cree una capeta donde se almacenara la base de datos

   5. Ejecute el script tk_captura_rostro.py

Entrenamiento del sistema

Pasos para entrenamiento del sistema de reconocimiento facial ejecute el mismo script y solo presiono la tecla inicia Entrenamiento

Instalación de la cadena de bloques y Implantación de contrato inteligente

Para la implementación de la red Blockchain se requiere instalar los paquetes necesarios para su ejecución

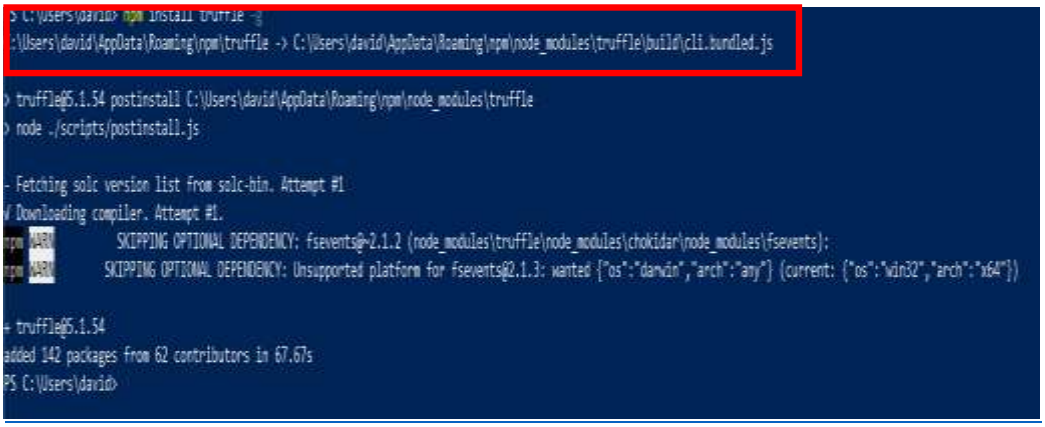
Instalación de Node.js

Node.js es un marco popular para desarrollar aplicaciones web, aplicaciones móviles y aplicaciones descentralizadas. Para instalar en la plataforma Windows, revisar anexo G.

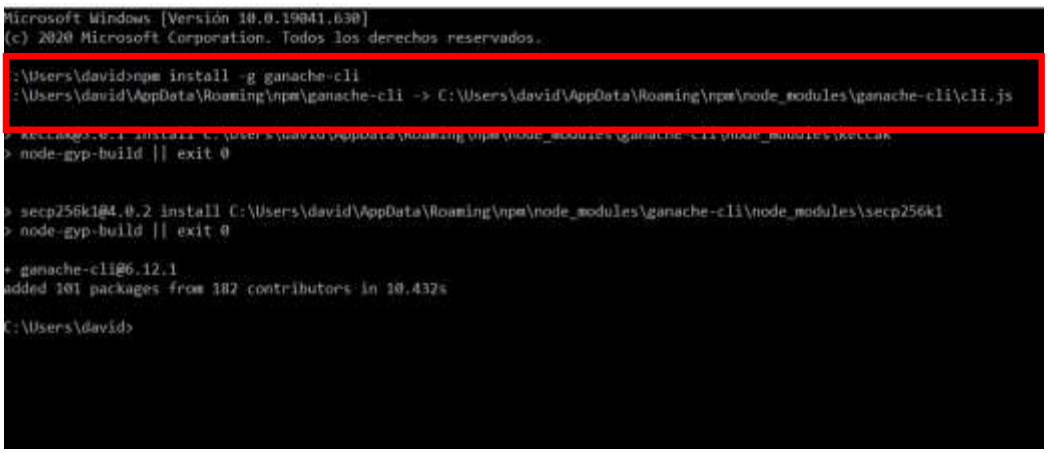
```
C:\Users\david>node --version
v14.13.0

C:\Users\david>npm --version
6.14.8

C:\Users\david>
C:\Users\david>
```

Instalación de Truffle

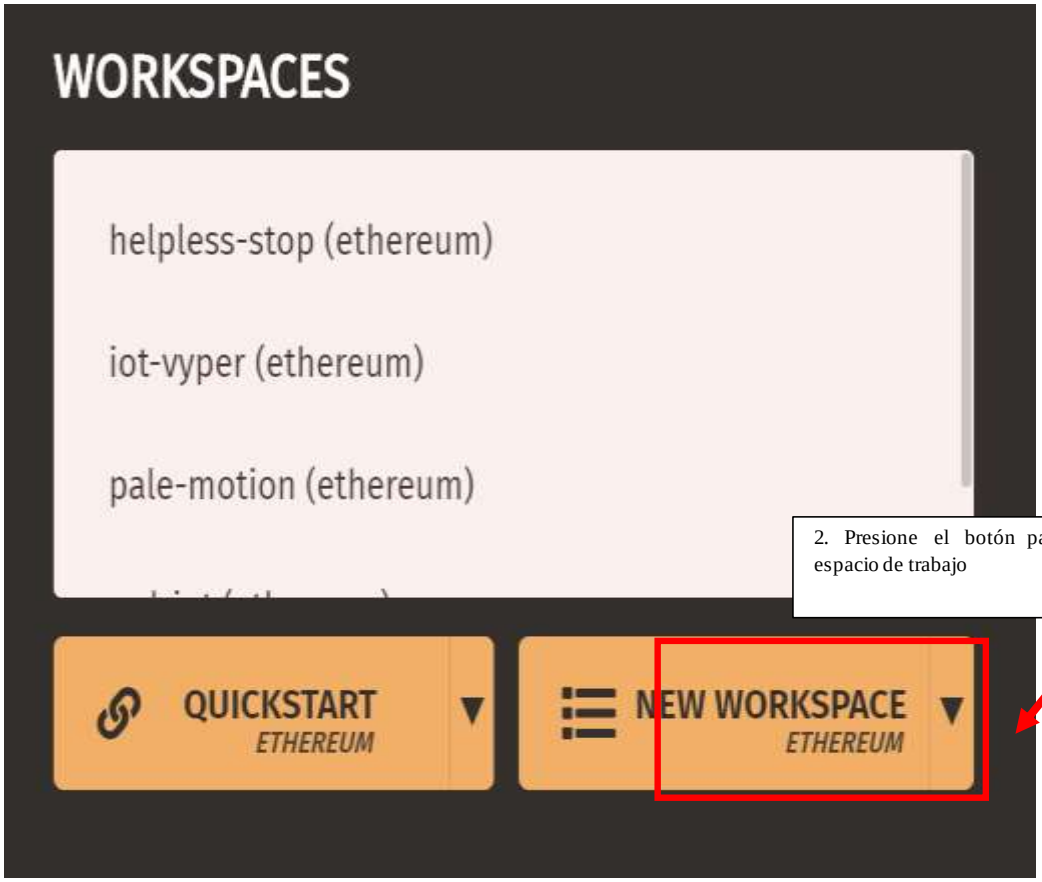
Para la compilación e implementación de contratos inteligentes en la red Blockchain utilizamos el marco truffle que nos permite el desarrollo de contratos inteligente en Solidity y Vyper.

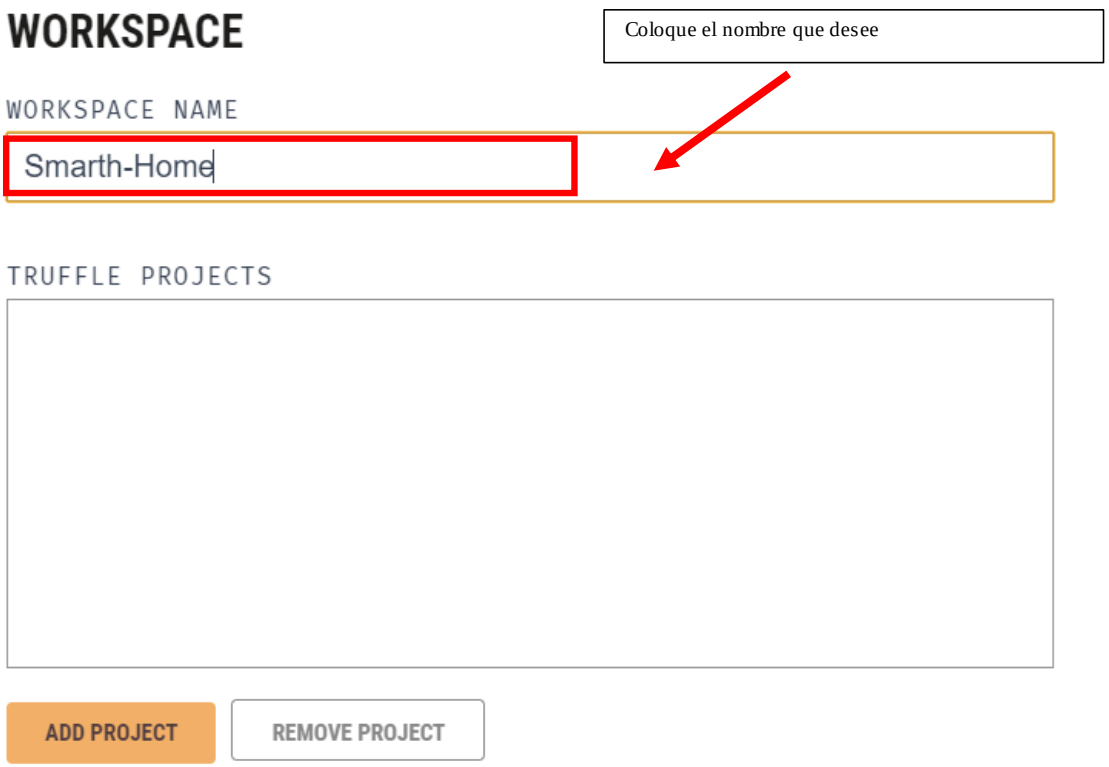


Instalación de Ganache



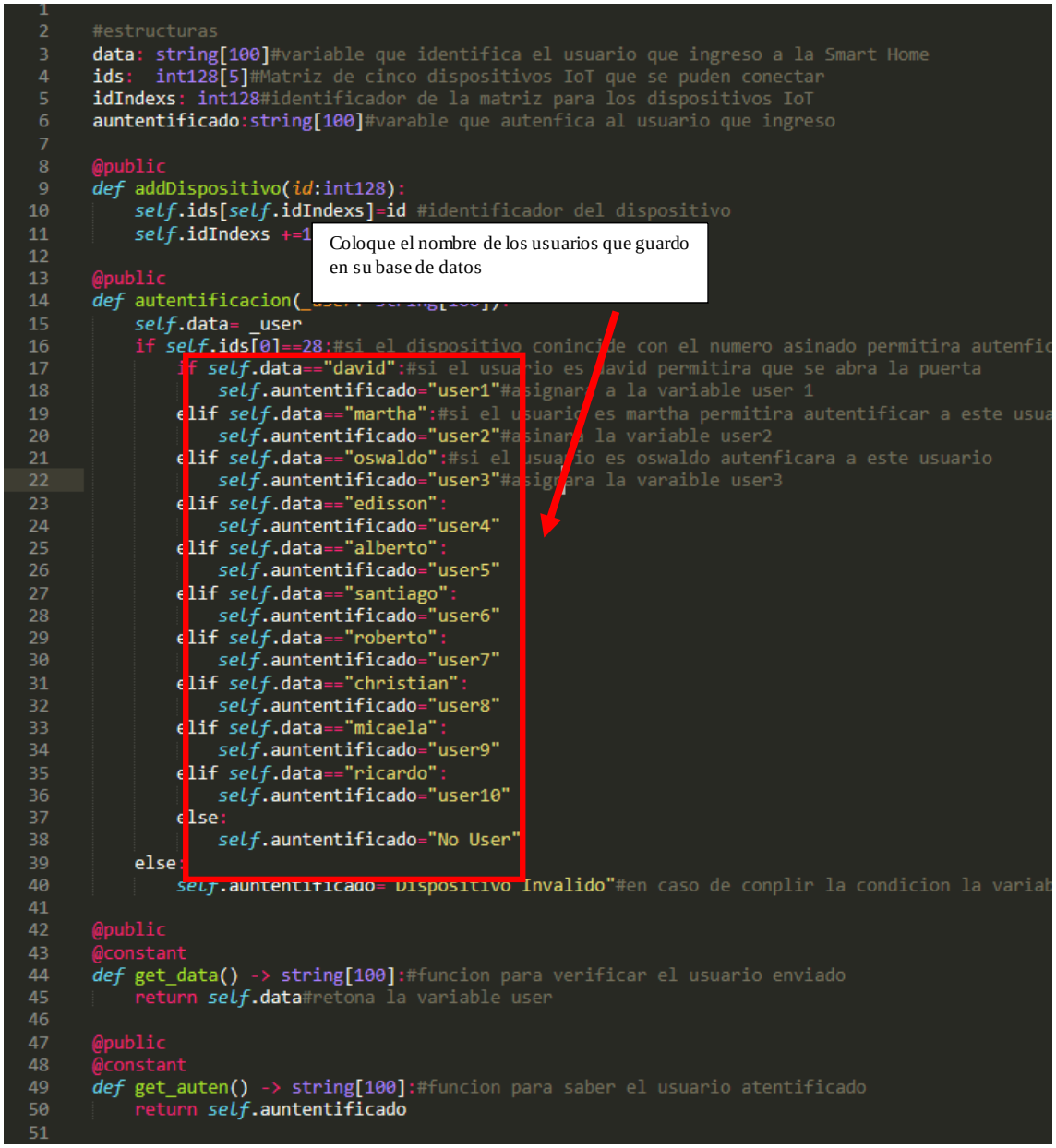
Creación del entorno de trabajo en ganache





Diseño del contrato Inteligente

Para la interacción entre el dispositivo IoT y la red Blockchain se necesita de un contrato inteligente el cual realizara en el lenguaje de programación Vyper, cambiar los nombres acordes a su base de datos




```
#estructuras
data: string[100]#variable que identifica el usuario que ingreso a la Smart Home
ids:  int128[5]#Matriz de cinco dispositivos IoT que se puden conectar
idIndexs: int128#identificador de la matriz para los dispositivos IoT
auntentificado:string[100]#varable que autenfica al usuario que ingreso

@public
def addDispositivo(id:int128):
    self.ids[self.idIndexs]=id #identificador del dispositivo
    self.idIndexs +=1

@public
def autentificacion(_user:string[100]):
    self.data= _user
    if self.ids[0]==28:#si el dispositivo coninc de con el numero asinado permitira autenfic
        if self.data=="david":#si el usuario es avid permitira que se abra la puerta
            self.auntentificado="user1"#asignar a la variable user 1
        elif self.data=="martha":#si el usuario es martha permitira autentificar a este usua
            self.auntentificado="user2"#asinar la variable user2
        elif self.data=="oswaldo":#si el usuario es oswaldo autenficara a este usuario
            self.auntentificado="user3"#asig ara la varaible user3
        elif self.data=="edisson":
            self.auntentificado="user4"
        elif self.data=="alberto":
            self.auntentificado="user5"
        elif self.data=="santiago":
            self.auntentificado="user6"
        elif self.data=="roberto":
            self.auntentificado="user7"
        elif self.data=="christian":
            self.auntentificado="user8"
        elif self.data=="micaela":
            self.auntentificado="user9"
        elif self.data=="ricardo":
            self.auntentificado="user10"
        else:
            self.auntentificado="No User"
    else:
        self.auntentificado= "Dispositivo Invalido"#en caso de conplir la condicion la variab

@public
@constant
def get_data() -> string[100]:#funcion para verificar el usuario enviado
    return self.data#retona la variable user

@public
@constant
def get_auten() -> string[100]:#funcion para saber el usuario atentificado
    return self.auntentificado
```

Implementación del contrato Inteligente en Ganache

Una vez que se haya completado todo el software necesario instalado, podemos empezar a escribir un contrato inteligente.

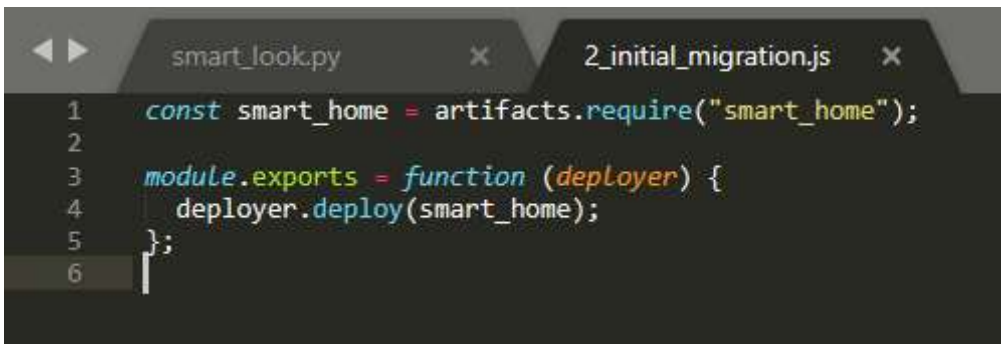
Crearemos un nuevo directorio y luego lo inicializaremos con la herramienta de desarrollo Truffle

```
D:\DAVID\Desktop\Ganache>mkdir contrato_smart

D:\DAVID\Desktop\Ganache>cd contrato_smart

D:\DAVID\Desktop\Ganache\contrato_smart>truffle init

Starting init...
================

> Copying project files to D:\DAVID\Desktop\Ganache\contrato_smart

Init successful, sweet!


D:\DAVID\Desktop\Ganache\contrato_smart>
```

Guardamanos en la carpeta “contracts” el contrato Smart_home.vy

```
Microsoft Windows [Versión 10.0.19041.746]
(c) 2020 Microsoft Corporation. Todos los derechos reservados.

C:\Users\david>d:

D:\>cd D:\DAVID\Desktop\Ganache\contrato_smart\contracts

D:\DAVID\Desktop\Ganache\contrato_smart\contracts>dir
 El volumen de la unidad D es DATA
 El número de serie del volumen es: 2ED2-F398

 Directorio de D:\DAVID\Desktop\Ganache\contrato_smart\contracts

20/01/2021  17:15    <DIR>          .
20/01/2021  17:15    <DIR>          ..
26/10/1985  03:15               419 Migrations.sol
20/01/2021  16:26             1.867 smart_home.vy
               2 archivos          2.286 bytes
               2 dirs  662.672.117.760 bytes libres

D:\DAVID\Desktop\Ganache\contrato_smart\contracts>
```

abra la carpeta "migrations" y cree un nuevo archivo llamado "2_deploy_contracts.js". Las migraciones son simplemente scripts que nos ayudarán a implementar nuestros contratos en una cadena de bloques.

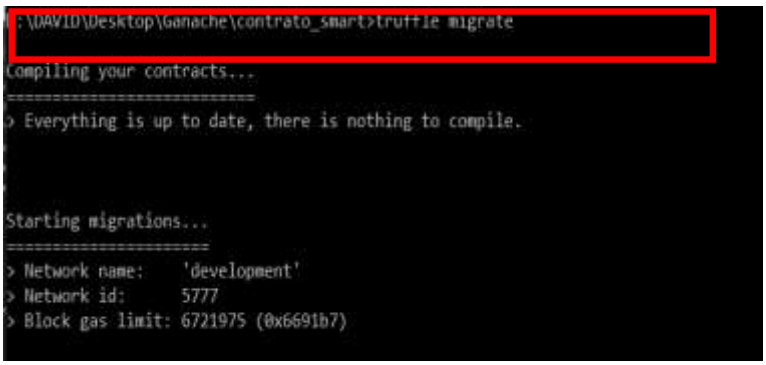


Edite el archive de configuración de Truffle: abra el archivo truffle-config.js, limpie y agregue las siguientes líneas de códigos.

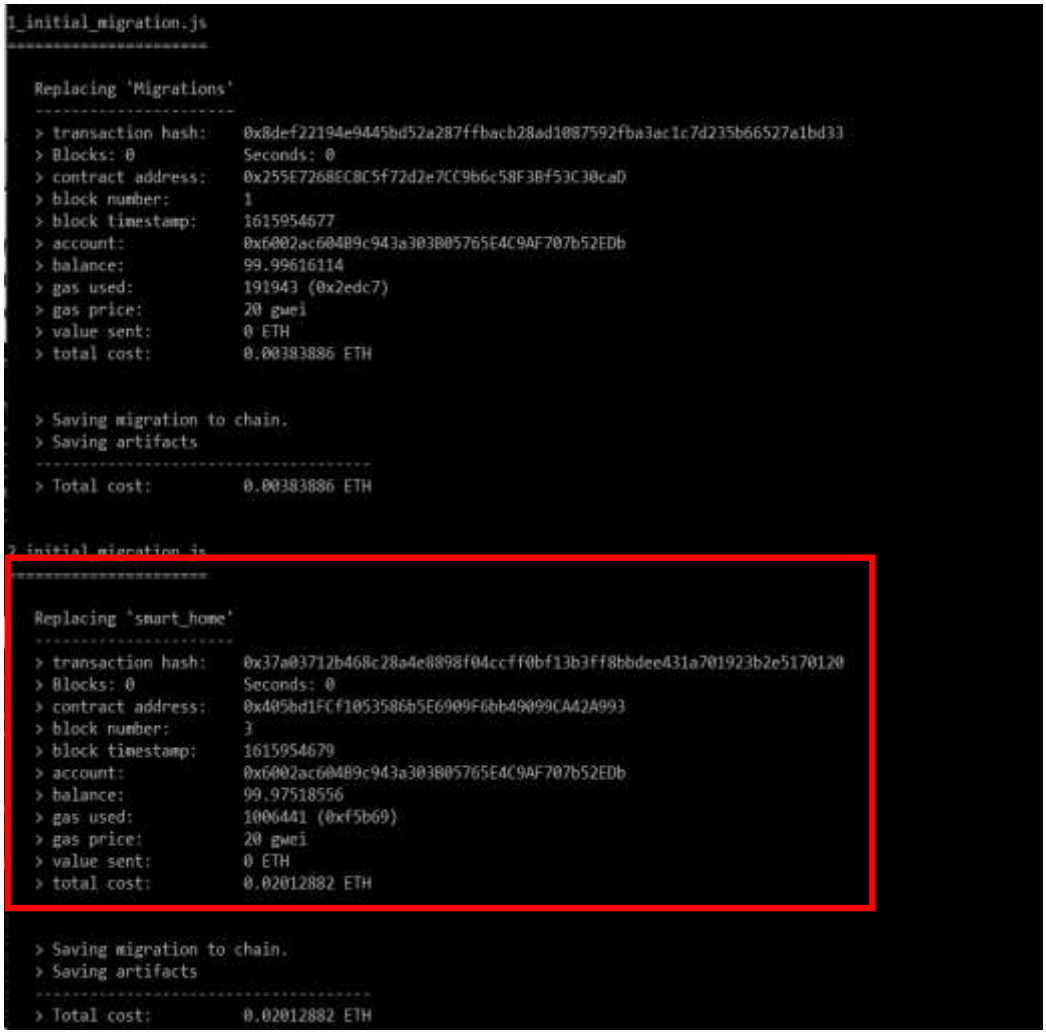
El host y el port se toman del servidor RPC en la pantalla Ganache, y network_id se toma del ID de red en la pantalla Ganache.

Ejecute scripts de migración: para implementar el contrato inteligente

Implementará el código en la cadena de bloques, en nuestro caso, la cadena de bloques se puede encontrar en el " development" de la red que configuramos anteriormente en el archivo "truffle-config.js".



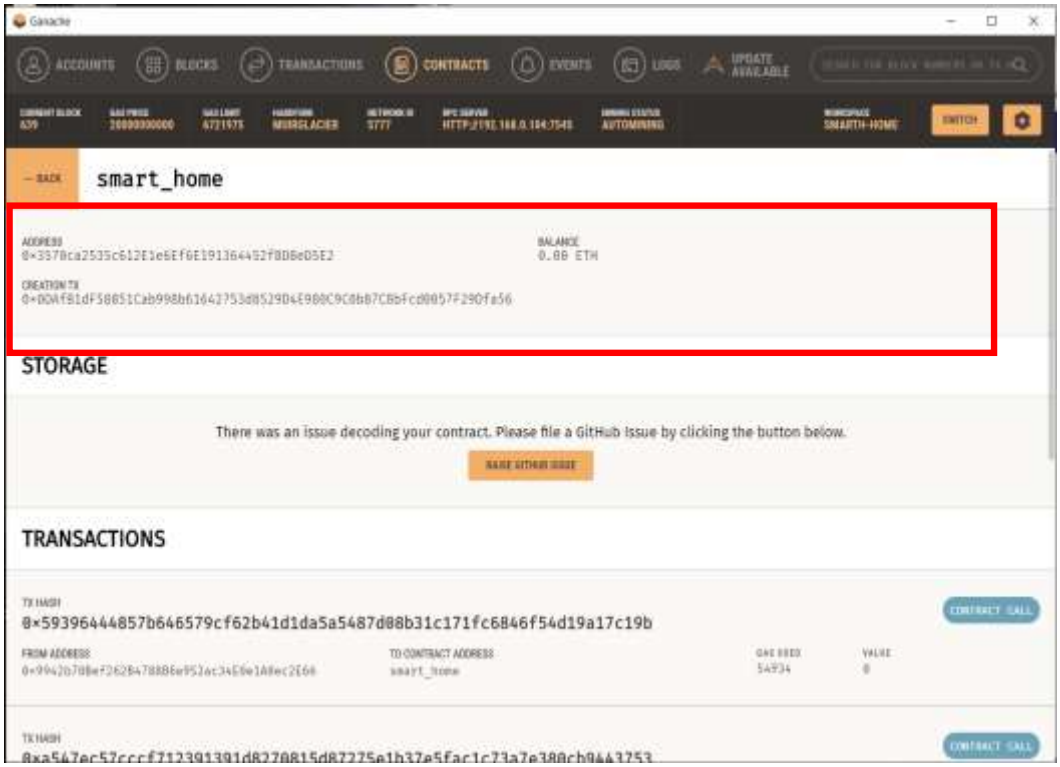
El marco Truffle tomará su código de bytes definido en el Donation.json archivo y lo enviará a Ganache. Esto le proporcionará el siguiente resultado:



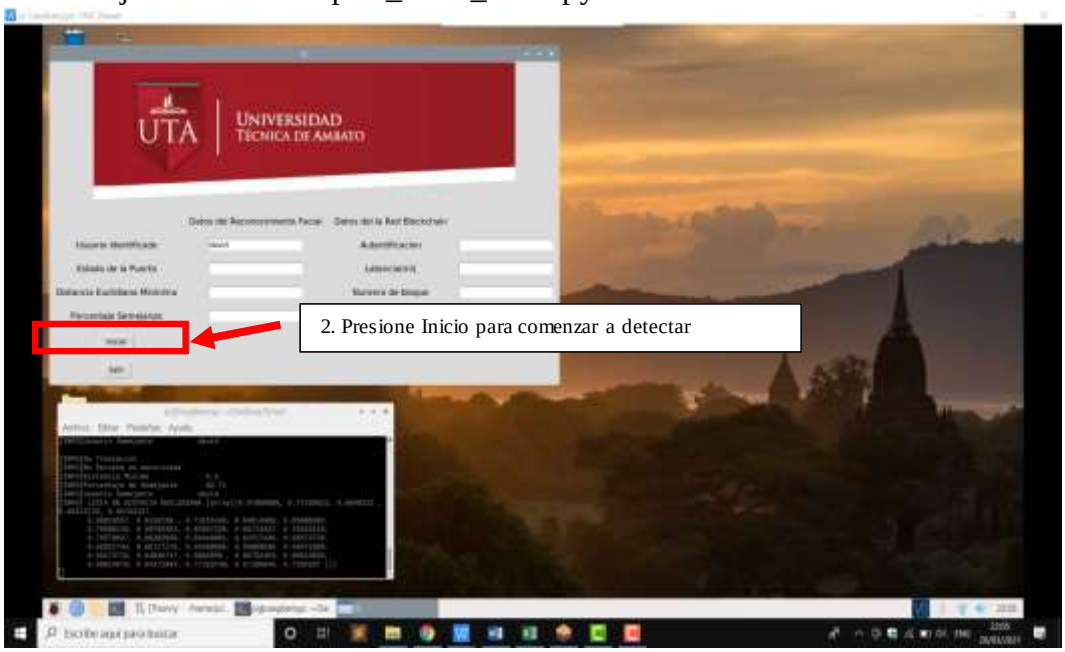
Su contrato inteligente escrito con Vyper se implementó en Ganache. La dirección de su contrato inteligente es la siguiente:

- **contract address**: 0x98524d3Ce1D06FED808DA7B4019E9421947Cb1db
- **account:** 0x9942b70Bef262B4788B6e952ac34E0e1A0ec2E66

En la cadena de bloques de Ganache tenemos la siguiente salida:

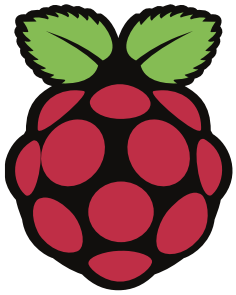


2. Ejecución del script tk_smart_home.py



Anexo I Manuales de los equipos

DATASHEET



Raspberry Pi 4 Model B

Release 1

June 2019

Table 1: Release History

Release	Date	Description
1	21/06/2019	First release

The latest release of this document can be found at https://www.raspberrypi.org

Introduction

The Raspberry Pi 4 Model B (Pi4B) is the first of a new generation of Raspberry Pi computers supporting more RAM and with siginficantly enhanced CPU, GPU and I/O performance; all within a similar form factor, power envelope and cost as the previous generation Raspberry Pi 3B+.

The Pi4B is avaiable with either 1, 2 and 4 Gigabytes of LPDDR4 SDRAM.

Features

Hardware

- Quad core 64-bit ARM-Cortex A72 running at 1.5GHz
- 1, 2 and 4 Gigabyte LPDDR4 RAM options
- H.265 (HEVC) hardware decode (up to 4Kp60)
- H.264 hardware decode (up to 1080p60)
- VideoCore VI 3D Graphics
- Supports dual HDMI display output up to 4Kp60

Interfaces

- 802.11 b/g/n/ac Wireless LAN
- Bluetooth 5.0 with BLE
- 1x SD Card
- 2x micro-HDMI ports supporting dual displays up to 4Kp60 resolution
- 2x USB2 ports
- 2x USB3 ports
- 1x Gigabit Ethernet port (supports PoE with add-on PoE HAT)
- 1x Raspberry Pi camera port (2-lane MIPI CSI)
- 1x Raspberry Pi display port (2-lane MIPI DSI)
- 28x user GPIO supporting various interface options:
    - ✓ Up to 6x UART
    - ✓ Up to 6x I2C – Up to 5x SPI
    - ✓ 1x SDIO interface
    - ✓ 1x DPI (Parallel RGB Display)
    - ✓ 1x PCM
    - ✓ Up to 2x PWM channels
    - ✓ Up to 3x GPCLK outputs

Software

- ARMv8 Instruction Set
- Mature Linux software stack
- Actively developed and maintained – Recent Linux kernel support
- Many drivers upstreamed
- Stable and well supported userland
- Availability of GPU functions using standard APIs
- Mechanical Specification

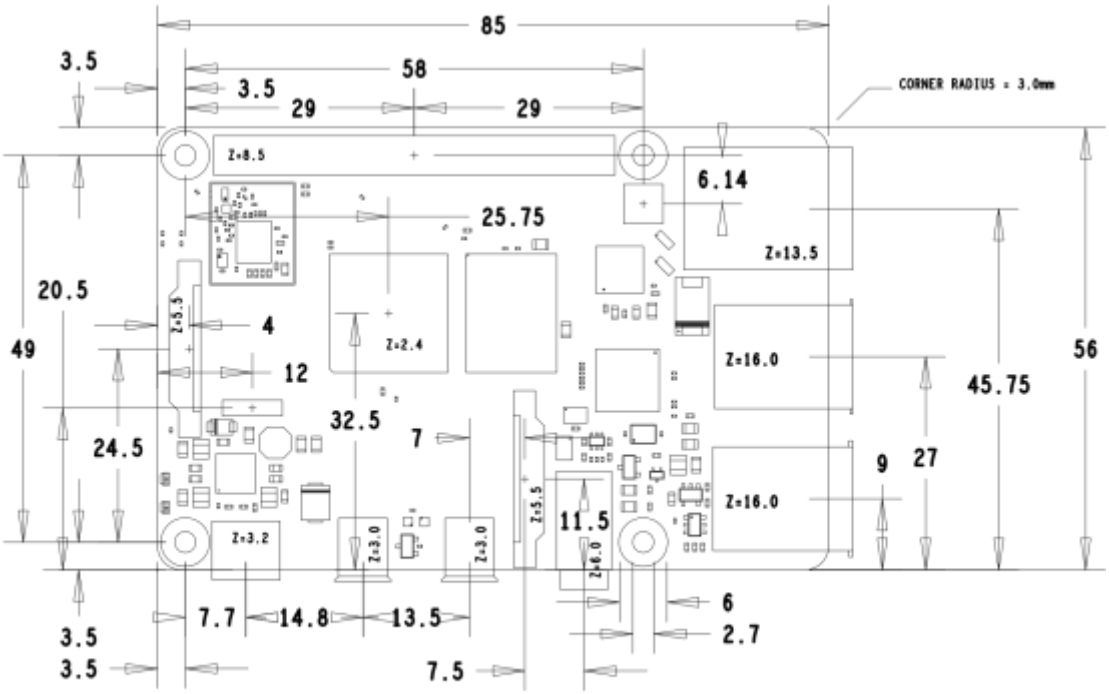


Figure 1: Mechanical Dimensions

Electrical Specification

Caution! Stresses above those listed in Table 2 may cause permanent damage to the device. This is a stress rating only; functional operation of the device under these or any other conditions above those listed in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Symbol	Parameter	Minimum	Maximum	Unit
VIN	5V Input Voltage	-0.5	6.0	V

Table 2: Absolute Maximum Ratings

Please note that VDD IO is the GPIO bank voltage which is tied to the on-board 3.3V supply rail.

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Unit
V_{IL}	Input low voltage[a]	VDD_IO = 3.3V	-	-	TBD	V
V_{IH}	Input high voltage[a]	VDD_IO = 3.3V	TBD	-	-	V
I_{IL}	Input leakage current	TA = +85°C	-	-	TBD	µA
C_{IN}	Input capacitance	-	-	TBD	-	pF
V_{OL}	Output low voltage[b]	VDD_IO = 3.3V, IOL = -2mA	-	-	TBD	V
V_{OH}	Output high voltage[b]	VDD_IO = 3.3V, IOH = 2mA	TBD	-	-	V
I_{OL}	Output low current[c]	VDD_IO = 3.3V, VO = 0.4V	TBD	-	-	mA
I_{OH}	Output high current[c]	VDD_IO = 3.3V, VO = 2.3V	TBD	-	-	mA
R_{PU}	Pullup resistor	-	TBD	-	TBD	kΩ
R_{PD}	Pulldown resistor	-	TBD	-	TBD	kΩ

[a] Hysteresis enabled
[b] Default drive strength (8mA)
[c] Maximum drive strength (16mA)

Table 3: DC Characteristics

Pin Name	Symbol	Parameter	Minimum	Typical	Maximum	Unit
Digital outputs	t_{rise}	10-90% rise time[a]	-	TBD	-	ns
Digital outputs	t_{fall}	90-10% fall time[a]	-	TBD	-	ns

[a] Default drive strength, CL = 5pF, VDD_IO = 3.3V

Table 4: Digital I/O Pin AC Characteristics

Figure 2: Digital IO Characteristics

Power Requirements

The Pi4B requires a good quality USB-C power supply capable of delivering 5V at 3A. If attached downstream USB devices consume less than 500mA, a 5V, 2.5A supply may be used.

Peripherals

GPIO Interface

The Pi4B makes 28 BCM2711 GPIOs available via a standard Raspberry Pi 40-pin header. This header is backwards compatible with all previous Raspberry Pi boards with a 40-way header.

GPIO Pin Assignments

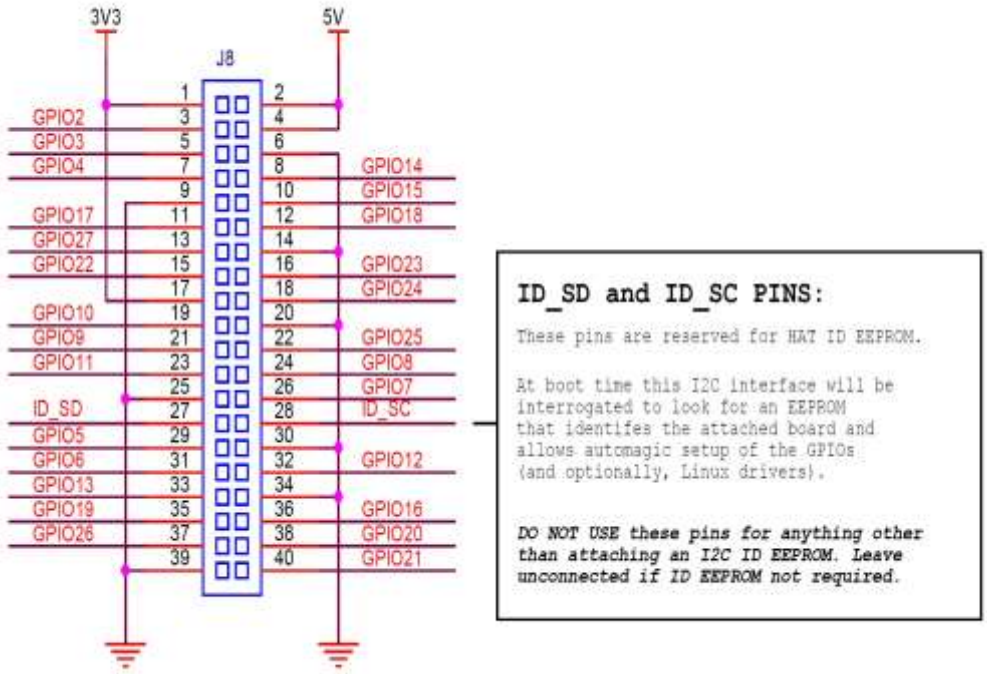


Figure 3: GPIO Connector Pinout

As well as being able to be used as straightforward software controlled input and output (with programmable pulls), GPIO pins can be switched (multiplexed) into various other modes backed by dedicated peripheral blocks such as I2C, UART and SPI.

In addition to the standard peripheral options found on legacy Pis, extra I2C, UART and SPI peripherals have been added to the BCM2711 chip and are available as further mux options on the Pi4. This gives users much more flexibility when attaching add-on hardware as compared to older models.

GPIO Alternate Functions

GPIO	Default Pull	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5
0	High	SDA0	SA5	PCLK	SPI3_CE0_N	TXD2	SDA6
1	High	SCL0	SA4	DE	SPI3_MISO	RXD2	SCL6
2	High	SDA1	SA3	LCD_VSYNC	SPI3_MOSI	CTS2	SDA3
3	High	SCL1	SA2	LCD_HSYNC	SPI3_SCLK	RTS2	SCL3
4	High	GPCLK0	SA1	DPI_D0	SPI4_CE0_N	TXD3	SDA3
5	High	GPCLK1	SA0	DPI_D1	SPI4_MISO	RXD3	SCL3
6	High	GPCLK2	SOE_N	DPI_D2	SPI4_MOSI	CTS3	SDA4
7	High	SPI0_CE1_N	SWE_N	DPI_D3	SPI4_SCLK	RTS3	SCL4
8	High	SPI0_CE0_N	SD0	DPI_D4	-	TXD4	SDA4
9	Low	SPI0_MISO	SD1	DPI_D5	-	RXD4	SCL4
10	Low	SPI0_MOSI	SD2	DPI_D6	-	CTS4	SDA5
11	Low	SPI0_SCLK	SD3	DPI_D7	-	RTS4	SCL5
12	Low	PWM0	SD4	DPI_D8	SPI5_CE0_N	TXD5	SDA5
13	Low	PWM1	SD5	DPI_D9	SPI5_MISO	RXD5	SCL5
14	Low	TXD0	SD6	DPI_D10	SPI5_MOSI	CTS5	TXD1
15	Low	RXD0	SD7	DPI_D11	SPI5_SCLK	RTS5	RXD1
16	Low	FL0	SD8	DPI_D12	CTS0	SPI1_CE2_N	CTS1
17	Low	FL1	SD9	DPI_D13	RTS0	SPI1_CE1_N	RTS1
18	Low	PCM_CLK	SD10	DPI_D14	SPI6_CE0_N	SPI1_CE0_N	PWM0
19	Low	PCM_FS	SD11	DPI_D15	SPI6_MISO	SPI1_MISO	PWM1
20	Low	PCM_DIN	SD12	DPI_D16	SPI6_MOSI	SPI1_MOSI	GPCLK0
21	Low	PCM_DOUT	SD13	DPI_D17	SPI6_SCLK	SPI1_SCLK	GPCLK1
22	Low	SD0_CLK	SD14	DPI_D18	SD1_CLK	ARM_TRST	SDA6
23	Low	SD0_CMD	SD15	DPI_D19	SD1_CMD	ARM_RTCK	SCL6
24	Low	SD0_DAT0	SD16	DPI_D20	SD1_DAT0	ARM_TDO	SPI3_CE1_N
25	Low	SD0_DAT1	SD17	DPI_D21	SD1_DAT1	ARM_TCK	SPI4_CE1_N
26	Low	SD0_DAT2	TE0	DPI_D22	SD1_DAT2	ARM_TDI	SPI5_CE1_N
27	Low	SD0_DAT3	TE1	DPI_D23	SD1_DAT3	ARM_TMS	SPI6_CE1_N

Table 5: Raspberry Pi 4 GPIO Alternate Functions

Table 5 details the default pin pull state and available alternate GPIO functions. Most of these alternate peripheral functions are described in detail in the BCM2711 Peripherals Specification document which can be downloaded from the hardware documentation section of the website.

Display Parallel Interface (DPI)

A standard parallel RGB (DPI) interface is available the GPIOs. This up-to-24-bit parallel interface can support a secondary display.

SD/SDIO Interface

The Pi4B has a dedicated SD card socket which suports 1.8V, DDR50 mode (at a peak bandwidth of 50 Megabytes / sec). In addition, a legacy SDIO interface is available on the GPIO pins.

Camera and Display Interfaces

The Pi4B has 1x Raspberry Pi 2-lane MIPI CSI Camera and 1x Raspberry Pi 2-lane MIPI DSI Display connector. These connectors are backwards compatible with legacy Raspberry Pi boards, and support all of the available Raspberry Pi camera and display peripherals.

USB

The Pi4B has 2x USB2 and 2x USB3 type-A sockets. Downstream USB current is limited to approximately 1.1A in aggregate over the four sockets.

HDMI

The Pi4B has 2x micro-HDMI ports, both of which support CEC and HDMI 2.0 with resolutions up to 4Kp60.

Audio and Composite (TV Out)

The Pi4B supports near-CD-quality analogue audio output and composite TV-output via a 4-ring TRS 'A/V' jack.

The analog audio output can drive 32 Ohm headphones directly.

Temperature Range and Thermals

The recommended ambient operating temperature range is 0 to 50 degrees Celcius.

To reduce thermal output when idling or under light load, the Pi4B reduces the CPU clock speed and voltage. During heavier load the speed and voltage (and hence thermal output) are increased. The internal governor will throttle back both the CPU speed and voltage to make sure the CPU temperature never exceeds 85 degrees C.

The Pi4B will operate perfectly well without any extra cooling and is designed for sprint performance expecting a light use case on average and ramping up the CPU speed when needed (e.g. when loading a webpage). If a user wishes to load the system continually or operate it at a high termperature at full performance, further cooling may be needed.

Availability

Raspberry Pi guarantee availability Pi4B until at least January 2026.

Support

For support please see the hardware documentation section of the Raspberry Pi website and post questions to the Raspberry Pi forum.

Datasheet Cerradura Selenoide

Esta cerradura eléctrica tipo solenoide tiene un diseño delgado que da la seguridad y estabilidad en su uso, es de bajo consumo y es muy útil para cerrar donde requiera; es fácil de instalar, por ejemplo, para un sistema de cierre automático de alguna puerta.

Este solenoide tiene un lingote con un corte inclinado; se trata básicamente de una cerradura electrónica que normalmente está con bloqueo activo para que no se puede abrir la puerta (por lo que el lingote está expandido) y no utiliza ningún voltaje en este estado. Cuando se aplica voltaje el lingote se contrae para que no sobresalga más y la puerta se pueda abrir fácilmente.

Importante: tenga en cuenta que los solenoides vienen con el lingote inclinado como se muestra en las fotos, pero se puede abrir con un destornillador y así darle la vuelta (rotándolo 90°, 180° o 270°) para que coincida con el sistema de la puerta electrónica que desee usar. No dejar energizada por mucho tiempo pues se recalienta.

Especificaciones técnicas

- Voltaje de operación: 6V ~ 12V DC (a menor voltaje más débil y lento su funcionamiento)
- Corriente: 0.6A a 12V | 0.5A a 9V
- Carrera de tracción: 0 ~ 4 mm
- Succión: 4mm≥80g (0.35A) kg
- Retención: 0mm≥120g
- Modo de operación: encendido en 90%, intervalo
- Tiempo de desbloqueo: 1 segundo
- Esperanza de vida: 10000000 veces
- Humedad relativa: 45% ~ 80%
- Temperatura de operación: -25°C ~ 80°C

Peso	*35 g*
Dimensiones	*3.2 × 2.7 × 1.7 cm*
Color	*Plateado*
Largo del cable	*22 cm*
Material	*Acero*