

Instituto Tecnológico y de Estudios Superiores de Monterrey

MA2006B.601

Uso de álgebras modernas para seguridad y criptografía

Escuela de Ingeniería y Ciencias

Ingeniería en Ciencias de Datos y Matemáticas

Implementación de criptografía de clave pública para comunicaciones y almacenamiento de datos con IoT en entornos de monitoreo y consumo de energía.

Profesores

Daniel Otero Fadul & Alberto Francisco Martínez Herrera

Socio Formador: COCOA y LiCore

Alumnas y alumnos

Diana Paola Cadena Nito	A01197399
Enrique García Varela	A01705747
Javier Hernández Arellano	A01730548
Alexis Hernández Spinola	A01275180
Paola Sofía Reyes Mancheno	A00831314
María Fernanda Torres Alcubilla	A01285041

Monterrey, Nuevo León. 20 de junio de 2023

Índice

1. Introducción	3
2. Estado del arte	5
2.1. Trabajos relacionados	5
2.2. Dispositivos IoT en la industria energética	8
2.3. Arquitecturas de intercambio de información.	8
2.4. Esquemas de clave pública para la protección de comunicaciones.	9
2.5. Bibliotecas disponibles para acelerar la implementación de los esquemas de clave pública.	10
2.6. Algoritmos criptográficos	11
2.6.1. Comparación entre algoritmos de clave pública para IoT.	13
2.7. Recursos disponibles	13
3. Propuesta	14
4. Metodología	15
4.1. Conexión TLS	16
4.2. ECDSA	16
4.3. Base de datos entrada (auditor)	17
4.4. Bases de datos servidor	17
5. Desarrollo	18
5.1. Pasos para llegar a Resultado finales	18
5.2. Experimentación adicional	22
6. Resultados finales	23
7. Conclusiones y trabajo a futuro	24

Resumen

Prototipo de intercambio de información basado en arquitectura Edge-Cloud, conectando sensores de Raspberry PI, a través de conexión TLS y almacenamiento en base de datos relacional. Con algoritmos de verificación ligeros.

1. Introducción

Frente al desarrollo exponencial de la tecnología, así como al aumento de la población que tiene acceso a esta, la necesidad de encontrar nuevas formas de producción de energías que sean renovables es cada vez más urgente. Tomando en cuenta que la generación principal de energía a nivel mundial es mediante combustibles fósiles, aparte de ser un recurso finito, genera gran cantidad de contaminación; desde el año 2015 la Organización de las Naciones Unidas, ha lanzado los 17 Objetivos de Desarrollo Sostenible para el 2030, donde uno de estos objetivos corresponde a una industrialización inclusiva y sostenible. Varios países que forman parte de la ONU, han decidido unirse a esta iniciativa, uno de ellos es México. (ONU, 2023).

Este proceso de transición, trae consigo varios beneficios para diferentes sectores de la humanidad. En primer lugar, se encuentra uno de los beneficios más mencionados, que es el cuidado del medio ambiente, lo que conlleva a una ralentización del calentamiento global, mejorando así la calidad del aire, la capa de ozono, entre otros. Así también, se tiene previsto que la salud de los habitantes mejorará en gran magnitud, tomando en cuenta que, actualmente, las muertes por polución del aire debido a la quema de combustibles fósiles se encuentra alrededor de 4 millones de personas anualmente. (J. et al., 2019).

Además, la transición a energías limpias también ayudará a reducir la brecha económica, así como la disminución de la volatilidad de los precios energéticos. Esto al reducir la necesidad de la importación de energía a ciertos países, lo que encarece su precio. Incluso, al disminuir drásticamente los precios de la energía, y al tener procesos más eficientes para conseguir la misma, es posible dar acceso a una *cocina limpia* al sector vulnerable de la población, la cual se refiere a poder cocinar con fuentes limpias que no contaminen con químicos a los alimentos. Este último punto, ayudará a prevenir muertes prematuras de aproximadamente 1.8 millones de personas para el año 2030. (J. et al., 2019).

No obstante, hacer el cambio de fuentes fósiles a limpias, es más complejo de lo que suena. Hay que tomar en cuenta la capacidad económica de cada país, así como el acceso al desarrollo tecnológico necesario para implementar estas nuevas formas de energía. Realizar un cambio sistemático también implica revisar contratos internacionales o con empresas privadas relacionados con la producción de energía; factores que complican a ciertos países más que a otros. Es por esto, que se vuelve imprescindible los esfuerzos conjuntos entre lo público y privado, y entre diferentes industrias para poder realizar un cambio sustancial en la producción de energía. (A. and I.).

El proceso de generación de energía habitualmente tiene una forma jerárquica, la cual consta de 3 pasos específicos, los cuales son Generación, Transmisión y Utilización. Este proceso permite controlar muy fácilmente el equilibrio de oferta-demanda; no obstante, al pasar por el proceso de transmisión, la energía se vuelve más costosa. Por otro lado, los planes que se tienen para hacer la transición, piensan un sistema más eficiente tanto en la generación de la energía, pero también en su llegada al usuario final. Esto significa que las nuevas plantas de producción puedan encontrarse cualquiera de los 3 pasos, lo cual lleva como nombre Generación Distribuida. La única dificultad en este nuevo modelo se centra en cómo controlar el equilibrio de oferta y demanda. (J. et al., 2019).

Una disciplina que muchas veces no es conocida dentro de este proceso (aunque juega un papel principal),

es el área tecnológica, específicamente la ciberseguridad. En el contexto mexicano, dos de las empresas que están aportando a la transición energética del país son LiCore y COCOA, las cuales actualmente están colaborando para desarrollar herramientas de monitoreo de instalaciones de producción y lugares de consumo. El objetivo de este trabajo se centra aportar en la construcción de este proyecto, mediante la creación de un sistema criptográfico que permita la comunicación segura entre sistemas IoT con el uso de una clave pública. A gran escala, se está buscando que el modelo funcione como se muestra en la siguiente figura.

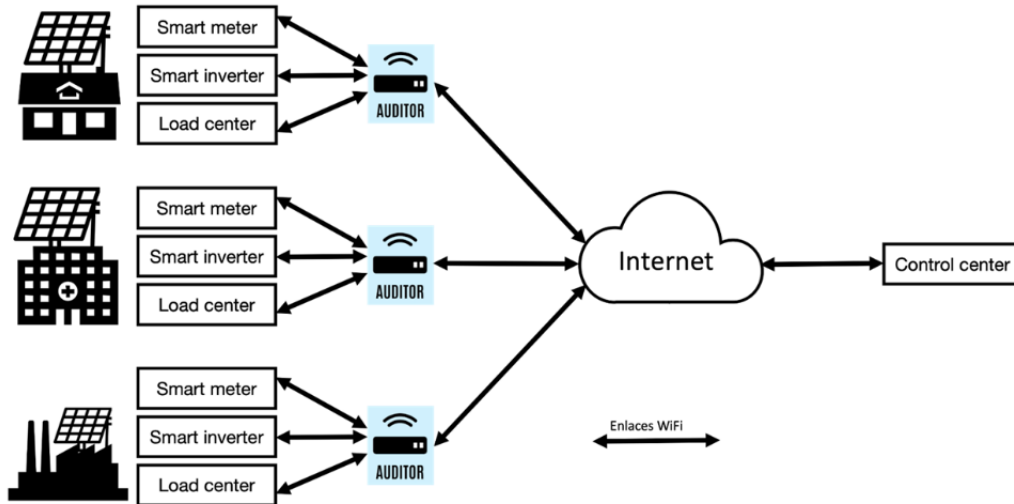


Figura 1: Esquema general del caso, obtenido de Canvas.

No obstante, se estará trabajando con un esquema de intercambio de información reducido. En la figura 2 se puede observar el modelo que se va a utilizar a lo largo de este trabajo, el mismo es una forma simplificada de lo que pasa en la vida real.

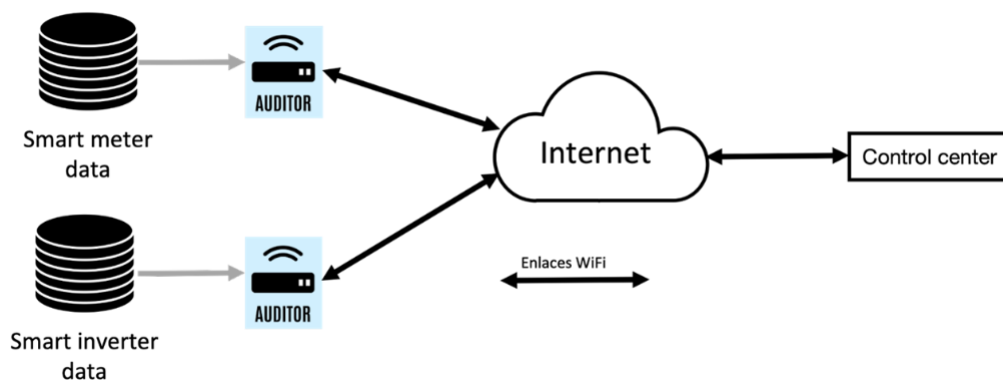


Figura 2: Esquema simplificado a utilizar en el reto. Tomado de Canvas

Los dispositivos IoT son un componente indispensable en el proceso de transición a energías limpias, pues al tener nuevas estaciones generadoras de energía de diferentes capacidades, se necesita respetar la demanda de energía y así evitar potenciales riesgos al tener producción de sobra. Estos dispositivos se identifican por poder conectarse a internet y por ende comunicarse con otros dispositivos del mismo tipo a través del transceptor que se encarga de enviar las mediciones a través de Wi-Fi, bluetooth o Zigbee; además, cuentan con un sensor que ayuda a tomar mediciones de lo que se desee. (M., 2021). No obstante, un área de oportunidad dentro de esta comunicación, es que es muy fácil de interceptar, lo cual en esta

industria puede significar un real problema de seguridad para las plantas productoras, así como de los usuarios de esta energía. Por esta razón, es necesario aplicar un esquema de clave pública que permita proteger la comunicación planta-usuario y se pueda permitir el avance de construcción de plantas para energía limpia.

2. Estado del arte

2.1. Trabajos relacionados

Con la finalidad de establecer un marco de referencia, en este apartado se muestran y detallan distintas arquitecturas empleadas, el uso de plataformas de servicios en la nube y distintas maneras en la que los datos dentro en dispositivos IoT pueden ser almacenados. Asimismo, se puede identificar el documento de donde fue recopilada la información.

Referencia	Arquitectura	Protocolo	Hardware	PKI	Lenguaje
(C. et al., 2015)	Client - server	-	Raspberry Pi B+	-	Python
(J., 2018b)	Peer to peer	CoAP, WebSocket	-	DHT	JavaScript
(L. et al., 2020)	Edge - cloud	MQTT	Raspberry Pi	-	-
(D., 2019)	Blockchain - Serverless y P2P	-	-	-	Go, AWS (IAM, S3 y Lambda), JavaScript
(M. et al., 2020)	Peer to peer	-	Arduino	-	-
(M. et al., 2018)	Fog-Mist	NTP	ESP32, Orange Pi PC	ECC	ESP32-IDF, Python
(J., 2018a)	Cloud-based single channel	MQTT	Raspberry Pi	-	Python

Cuadro 1: Trabajos relacionados (E., 2022)

Wai Zhao et al. condujeron un trabajo donde demostraron el uso de una Raspberry Pi en una arquitectura tipo *client server* aplicando diferentes escenarios para comunicar distintos dispositivos de manera inalámbrica (C. et al., 2015). El primer escenario que proponen es mediante una conexión WiFi, donde se configura un servidor en la Raspberry Pi con el software *Samba* que otorga permisos de lectura y/o escritura a los distintos usuarios. Para el segundo escenario se plantea el uso de *ZigBee*, una tecnología de comunicación de corto alcance, útil para un flujo de datos moderado y en distancias cortas, formando una malla de red.

En el documento (J., 2018b) se propone Peer to peer, con una nueva arquitectura de IoT software, siendo esta el framework de Devify, además de una programación basada en flujo para el intercambio de información. La arquitectura está basada en JavaScript y cuenta con un diseño de capas, en la que primero se tiene la aplicación lógica, un servidor intermedio basado en el estilo de operaciones REST y la capa de red de las cosas. Por otra parte, el framework propuesto, está instalado en una red local LAN, con el protocolo de CoAP, el cual funciona para cumplir el protocolo RESTful. Asimismo, se presenta un

segundo caso en el que se instaló en la nube a través del protocolo de WebSocket, el cual se basa en el método de handshake.

De igual manera, el documento (L. et al., 2020) comienza a combinar las arquitecturas de dispositivos IoT con la nube. En este caso, se abarca la incorporación de estas dos herramientas a través de un enfoque de seguridad de la red. Se mencionan las amenazas presentes, los distintos métodos para la protección de datos en estos ambientes así como alternativas y una comparación entre las mismas. En este documento, para la parte en la nube se hizo uso de Amazon Web Service y de una Raspberry Pi 4 junto con el soporte correspondiente para el uso de un ambiente Greengrass Edge. Asimismo, se implementaron certificados de seguridad entre las distintas componentes del modelo, generando lo que se considera como un modelo robusto en cuanto a seguridad.

Dentro del documento (D., 2019) se hace un enfoque hacia el área financiera. En este documento, se propone el desarrollo de un banco digital donde cada usuario sea capaz de comprobar su estado de cuenta y realizar transferencias a otras cuentas. La actualización de estos datos se da de manera local, en el código de cada cliente, y la conexión entre clientes se da a través de la nube. Además, se ofrece un almacenamiento en la nube sobre los estados de cuenta y el historial así como archivos de log de los movimientos realizados. Básicamente, se desarrolló una arquitectura P2P por medio de blockchain con cierta parte de su procesamiento en una plataforma en la nube.

Por otro lado, en la investigación (M. et al., 2020) se está realizando un proyecto con IoT y blockchain basado en peer to peer, para plataformas de intercambio energético y que se pueda mantener un balance entre la oferta y demanda del mercado. Esto se ha logrado a través de la estructura de Ethereum blockchain para generar los pagos de consumo de energía. Además, de estar ocupando un Arduino para detección de la cantidad de KWh consumidos, además la comunicación está basada en Node-Rd.

Más adelante, en el documento de (M. et al., 2018) se ocupa una arquitectura basada en cloud-edge pero aplicada con un sistema computacional fog y el método mist. Este tipo de sistemas son intermediarios entre cada una de las esquinas del sistema, para que la información se pueda cargar más rápido y de forma más segura. Esto es posible debido al desarrollo de eficiencia de energía que ha habido en los sistemas en Chip. En la experimentación se utilizó el protocolo de TLS y se comparó entre curvas elípticas y RSA, de modo que el primero le fue mejor tanto en consumo de energía y procesamiento de datos.

Por último, en el trabajo (J., 2018a) se detalla la incorporación de los dispositivos IoT en la nube junto con *gateways* y otros componentes. En particular, el documento habla acerca de la adaptación de tecnologías LoRa LPWAN por medio de una Raspberry Pi, donde LoRa está basado en el modelo OSI y el espectro ensanchado de chirp (CSS). La implementación de LoRa en dispositivos IoT se debe a su bajo consumo de energía para mandar mensajes a través de miles de kilómetros. No obstante, cabe mencionar que por lo mismo de su bajo consumo energético, el tamaño de los mensajes se encuentra limitado. Además, la comunicación entre los dispositivos por medio de LoRa se encuentra bajo el protocolo HTTP y MQTT. El objetivo del documento es establecer una red aplicable a la industria de bajo costo para diversas actividades que se busquen realizar.

Como se ha mencionado, los dispositivos IoT pueden estar conectados a través de la nube o no. Ambas opciones muestran sus respectivas ventajas y desventajas en cuanto a seguridad, costo y recursos en general. En la siguiente tabla, se muestran distintas plataformas que han sido aplicadas en IoT para mantener el centro de control y bases de datos en la nube, además de las mencionadas anteriormente.

Referencia	Plataforma	Protocolo	Costo	Comentarios	Base de datos
(M., 2022)	Home Assistant	TCP, MQTT	-	Home Assistant Cloud, configuración a Google Home manual	InfluxDB 2.0
(O., 2017)	Azure IoT Hub	MQTT, HTTP, AMQP	Hasta €4216.5 mensuales	Stream Analytics, distintos planes de pago	SQL Azure, hasta € 784 mensuales
(V., 2020)	AWS IoT Core	MQTT, HTTP	-	-	AWS DynamoDB

Cuadro 2: Trabajos relacionados: IoT en la nube

Donde en los documentos (M., 2022) y (V., 2020), se detalla la incorporación de los hardwares Raspberry Pi y Arduino Yún SDK respectivamente.

También, se presenta el proyecto de (M., 2022), en el que se esté realizando la estructura de un sistema IoT para sensores de calidad del aire, a través del microcontrolador ESP32 y un HomeAssistant por medio de Raspberry Pi, programado en Python. Los cuales están caminándose a través del protocolo en MQTT y su centro de control está basado en InfluxDB y es administrado a través de Kubernetes, además de guardarlo en un servidor local. Entre los resultados, se muestra que sería bueno ocupar una Raspberry tipo Pi Pico como microprocesador para mejorar la eficiencia.

Entre las referencias, se presenta (O., 2017). En la que se hace la integración de un micriporicesador en Raspberry Pi 3, conectada a sensores que permitan la detección de niveles de humedad. Otros recursos que se utilizaron fue SDK de Microsoft Azure, de modo que funcione como centro de control, entre sus limitaciones se encontraban la cantidad de dispositivos y los mensajes que se podrían recibir, a su vez la base de datos donde se está guardando la información es en SQL Server.

En el tercer documento, (V., 2020), se utiliza la plataforma Amazon Web Service, junto con OpenHAB y los protocolos HTTP y MQTT, dentro de una casa inteligente para el almacenamiento y generación de gráficas automática en la nube. A lo largo del documento, se detalla la conexión del dispositivo a AWS sin tomar los datos en tiempo real; en este caso, se establece cierto periodo de tiempo en la actualización de estos. OpenHAB es un software de código abierto utilizada para integrar distintos sistemas automáticos en un ambiente doméstico. Para esta propuesta del sistema, se hizo uso de los servicios gratuitos de AWS así como el código abierto por lo que el costo fue de cero. Cabe mencionar que, al ser un sistema IoT de uso 100 % doméstico, tiene limitantes en cuanto a su escalabilidad al ámbito industrial.

Los datos registrados dentro de sistemas y dispositivos IoT debe ser almacenado en una base de datos, las cuales pueden ser relacionales o no relacionales. Es decir, SQL o NoSQL. Por lo general, suele optarse por bases de datos NoSQL debido al gran tamaño de datos registrados y por registrar. Además, es relevante mencionar que estas bases deben ser encriptadas para que los datos estén protegidos también en esta parte del sistema. A continuación, se muestra una tabla con las distintas características de cada tipo de base de datos.

Referencia	Característica	SQL	No SQL
(J., 2020)	Almacenamiento	Tablas	JSON, tablas dinámicas
	Esquema	Rígido	Flexible
	Escalabilidad	Vertical	Horizontal
	Consultas	Joins requeridos, consultas más lentas.	Joins requeridos, consultas más rápidas.

Cuadro 3: Comparación entre bases de datos SQL y no SQL.

Como se puede observar en la tabla 2, algunas de las plataformas también ofrecen la encriptación de las bases de datos. No obstante, cabe mencionar que esto tendría un costo extra en caso de que se opte por utilizar estos servicios.

2.2. Dispositivos IoT en la industria energética

La comunicación Machine To Machine, o M2M, es imprescindible para la producción eficiente de energía en un contexto actual, por lo que existen varios sensores IoT que actualmente se están utilizando para establecer la comunicación entre la producción y consumo de la energía. En primer lugar, se encuentran los Smart Meters los cuales son medidores desarrollados específicamente para reemplazar los medidores tradicionales de consumo energético. Estos Smart Meters, permiten regresar los datos recopilados a la empresa, lo que no solamente ayuda a que los pagos sean más precisos, sino a que se puedan armar perfiles y patrones de consumo y la producción de energía se base específicamente en los patrones de sus usuarios. (C., 2018).

Así también, existen los *Smart Inverters* o Inversores Inteligentes. Un inversor común permite convertir la corriente que producen paneles solares u otras herramientas de energía limpia, a corriente alterna para ser utilizada como energía eléctrica en hogares. Por su parte, los Smart Inverters permiten regular la frecuencia y el voltaje así como la conexión con redes inteligentes con otros dispositivos IoT. (IREC).

2.3. Arquitecturas de intercambio de información.

Las arquitecturas de red son la representación del diseño y funcionamiento de las redes informáticas, en donde se conectan los protocolos y distintos softwares de la forma más efectiva en cuanto a costos. Las redes son de ayuda en cualquier ámbito debido a su reducción de redundancias, tiempo de dedicación y esfuerzo y facilitación del intercambio de información. (V. et al., 2018). Algunas características de las arquitecturas de red respecto al intercambio de información es la conectividad amplia entre cualquier nodo, con los niveles de seguridad adecuados y la administración de datos al interconectar distintos sistemas. (V. et al., 2018).

La arquitectura de client-server está basada en un servidor central que permite la administración de las computadoras que requieren servicios de cliente. Por lo mismo, la forma que tiene la red es estructurada para que sea una jerarquía, en el que cada nivel tendrá distintos privilegios, siendo así una administración centralizada.

De forma contraria, para las redes de Peer to peer estas tienen el objetivo de crear una arquitectura descentralizada. En la que los nodos son las computadoras conectadas entre sí, con la misma cantidad de privilegios dentro de la red. Entonces, para este caso cada nodo tiene función de ser cliente y servidor, de modo que entre estos se comparten archivos, almacenamiento y memoria. Sus ventajas permiten tener

resistencia a las fallas, ya que solo afecta a cada nodo de forma individual, sin embargo, por lo mismo incrementa la complejidad de la red. (IBM). Además, este permite una alta privacidad al funcionar con la tabla de hash distribuida, la cual estaría asignando una llave única respectiva a cada nodo.

Por último, la tecnología con más expectativas en los últimos años, el blockchain, se basa en una arquitectura descentralizada y distribuida, donde no existen los nodos centrales, esto permite que no se tenga un solo punto de ataque y sea más segura. Para el intercambio de información se crean bloques y huellas digitales donde con el uso de la criptografía asimétrica de clave pública se obtiene una descripción del bloque, la cual al compartir el bloque se cambia automáticamente y es detectado por la red. Este tipo de arquitectura se usa especialmente cuando se necesita tener un registro permanente de los datos y transacciones, disponer de la misma información por igual en un grupo de personas o mantener la integridad y privacidad de información sensible. (del Calzado de La Rioja).

2.4. Esquemas de clave pública para la protección de comunicaciones.

La infraestructura de clave pública, también denominada ICP o PKI por sus siglas en inglés (Public Key Infrastructure), es un sistema criptográfico que hace uso de un par de claves, pública y privada. (S. and Adrián, 2019). La clave pública, como su nombre lo dice, puede ser conocida por terceros y ser utilizada para cifrar datos, pero la clave privada únicamente debe ser conocida por el dispositivo receptor para descifrar correctamente los datos. (S. and Adrián, 2019). Para su correcto funcionamiento, se requiere de tanto software como hardware, así como procedimientos y políticas de seguridad correspondientes. Dentro de esta infraestructura, se hace uso de un cifrado y firma digital con criptografía para la autenticación de ambas partes en el intercambio de información. (F., 2018). Por lo general, esto se realiza bajo el estándar X.509, gestionado por un tercero de confianza y que a su vez vincula identidades a una clave criptográfica. (F., 2018). Una infraestructura de clave pública se puede ver de la siguiente manera.



Figura 3: ICP (F., 2018)

Las infraestructuras deben seguir distintos procesos y contar con ciertos componentes; entre estos están la firma digital, autenticación, integridad, no repudio, gestión de claves, autoridad de certificación, registro y validación, titular del certificado y un repositorio. (F., 2018). La firma digital es la clave privada del certificado, la cual puede estar incluida en algún documento o software. La autenticación es la verificación de la identidad del usuario legítimo por medio de tokens de seguridad y la integridad garantiza que la información no sea modificada sin autorización a través de la firma digital. De igual manera, existe un registro de las transacciones realizadas, por lo que el mismo intercambio de información puede ser trazado, siendo esto el no repudio de la infraestructura. La gestión de claves y el repositorio hacen referencia al almacenamiento de claves privadas y certificados, respectivamente. (F., 2018). En cuanto a las distintas

autoridades implicadas, la autoridad de registro verifica la información requerida para que la autoridad de certificación emita un certificado asociado a un usuario. La autoridad de certificación, además de emitir los certificados, también se encarga de gestionarlos una vez verificada la información y tiene la capacidad de revocarlos junto con la autoridad de validación. La autoridad de validación simplemente proporciona información acerca del estado de un certificado. (F., 2018).

En pocas palabras, las ICPs vinculan claves públicas a dispositivos a través de una autoridad de certificación con la finalidad de que el dispositivo cifre los datos para su protección durante la comunicación, sobre todo si estos serán intercambiados por medio del Internet. (S. and Adrián, 2019). El uso de estas en un ambiente IoT aumenta en seguridad conforme el ambiente aumenta en tamaño pero si la clave privada es filtrada, la seguridad del sistema se vuelve inexistente. Para evitar que esto suceda, se pueden tomar distintas acciones preventivas como mantener un control sobre el acceso a través de un archivo de log y/o la protección por medio de contraseñas y permisos de usuario. (S. and Adrián, 2019). De igual manera, se puede establecer un sistema de doble autenticación junto con la ICP, aumentando la protección de los datos.

Por otra parte, existe otro modelo de ICP que elimina la necesidad de terceros (certificados digitales) que a su vez representan una posible vulnerabilidad en el sistema. Esta es denominada como infraestructura de clave pública distribuida, o D-PKI. (F., 2018). La infraestructura es más segura, puesto que cada identidad tiene control sobre su identificador actual y el enlace de la clave pública, además de que el sistema se actualiza en su totalidad cuando se hace cualquier modificación en alguna identidad. Esto es posible a través del uso de almacenes de datos clave-valor descentralizados, como lo es el Blockchain. (F., 2018).

Debido a que la implementación de infraestructuras de este tipo no es tan compleja, las ICPs pueden ser utilizadas en distintos ámbitos e industrias; entre estas se encuentra la industria de electricidad. Las redes eléctricas inteligentes, o *smart grids*, suelen intercambiar información entre distintos elementos de la red. Puede ser el caso que esta comunicación sea llevada a cabo mediante redes externas, como lo es el Internet, por lo que mantener la seguridad de los datos se vuelve un aspecto muy importante en estos sistemas. Para automatizar y alcanzar cierto nivel de inteligencia en las *smart grids*, es necesario que los medidores proporcionen información detallada casi al instante, siendo este otro factor relevante en cuanto a la protección de los datos. Se tienen tres objetivos principales en la protección de datos en las *smart grids*: la disponibilidad, integridad y confidencialidad. (Y. et al., 2016). La disponibilidad hace referencia a que los datos deben contar con un acceso oportuno y confiable para que la demanda de la energía sea cumplida correctamente. La integridad en las *smart grids* busca que la información no sea alterada o destruida, evitando una mala administración de la energía. Por último, la confidencialidad de la información restringe el acceso de la misma y evita su divulgación a terceros, manteniendo la privacidad del usuario. (Y. et al., 2016).

2.5. Bibliotecas disponibles para acelerar la implementación de los esquemas de clave pública.

Uno de los esquemas de clave pública más populares como ya fue mencionado anteriormente es el esquema PKI. Este esquema puede ser implementado de forma segura a través de la interfaz CryptoSys PKI Pro. Dicha interfaz permite habilitar distintas funciones para cifrar mensajes utilizando algoritmos de clave pública, descifrarlos, verificar señales digitales, crear claves públicas y privadas con curvas elípticas, entre otras cosas. Dicha interfaz puede ser implementada en distintos sistemas operativos Windows, partiendo desde Windows XP hasta Windows 11. Además de poder ser usada con distintos lenguajes de programación como Visual Basic, C, C++ y Python. (Limited).

Otra librería comúnmente usada para la implementación de clave pública en el lenguaje de programación Javascript es TweetNaCl, originalmente creada para el lenguaje C. Dicha librería permite implementar algoritmos de clave pública, hashing y señales digitales. Dentro de ellas tiene módulos como secretbox, con el cual se hace todo el proceso de incryptación y desincryptación de mensajes con algoritmo de criptografía asimétrica manera eficiente. (Unknown, b).

El algoritmo estándar que usa es el Diffie-Hellmann y una de las ventajas es que prácticamente el desarrollo de la librería hace que el usuario que desee implementarla en su código no necesite que tomar decisiones acerca de que algoritmo elegir. El código fuente de TweetNaCl es constantemente auditado por la firma de ciberseguridad Cure53 y muestra no tener fallas en vulnerabilidad. (Unknown, a).

Por otro lado, se encuentra disponible para el público la librería de software OpenSSL. Siendo esta una librería con 25 años en despliegue, es una librería escrita en C, que puede ser usada para poner en marcha algoritmos criptográficos de distintos tipos, siendo unos de estos de esquema público. Una de sus características es que permite implementar PKI desde la línea de comandos de una manera intuitiva y sencilla, ya que solo requiere pocos pasos para su instalación y su uso. (S.). A manera de resumen, se muestra la siguiente tabla con las distintas bibliotecas disponibles.

Nombre	Lenguaje de implementación	Ventajas	Desventajas
CryptoSys PKI Pro	Python, C, C++	Instalación y uso rápido	Requiere sistema Windows
TweetNaCl	Javascript	Más rápida que librerías tradicionales como OpenSSL	Sólo se puede implementar en Javascript
OpenSSL	Shell scripting	Se puede implementar desde la línea de comandos	Lenta a comparación de librerías modernas

Cuadro 4: Librerías usadas para implementación de esquemas de clave pública

2.6. Algoritmos criptográficos

La criptografía busca preservar distintos aspectos de la seguridad de la información por medio de algoritmos criptográficos. Principalmente, se tienen algoritmos simétricos, asimétricos, híbridos e inclusive existen algoritmos propios generados por cada entidad. (A., 2018). Los algoritmos criptográficos simétricos se basan en el uso de una sola clave para el cifrado y descifrado de la información, donde el receptor y emisor conocen dicha clave. Este es un punto débil de estos algoritmos, puesto que es más sencillo poder interceptar una sola clave. (A., 2018). Uno de los algoritmos simétricos más utilizados es el AES, *Advanced Encryption Standard*. Este es un esquema de cifrado por bloques y también es conocido como Rijndael. Para poder contrarrestar la debilidad de los algoritmos simétricos, se desarrolló la criptografía asimétrica. Este tipo de criptografía requiere dos claves para el cifrado y descifrado del intercambio de información; también es conocida como de clave pública. En los algoritmos asimétricos, existe una clave pública y otra privada; ambas son generadas al mismo tiempo y la relación entre ellas debe ser compleja para que no se pueda obtener una al conocer la otra. (A., 2018). La criptografía de clave pública se explica a detalle en apartados anteriores.

Tipo de criptografía	Algoritmos
Simétrica	AES, DES, 3DES, Blowfish, Serpent
Asimétrica	RSA, ECC, ElGamal, DSA

Cuadro 5: Algoritmos criptográficos de clave pública y privada. (E., 2022)

En cuanto a la criptografía híbrida, esta es una mezcla entre los algoritmos simétricos y asimétricos. A continuación, se muestra una imagen de cómo funciona un sistema bajo esta unión.

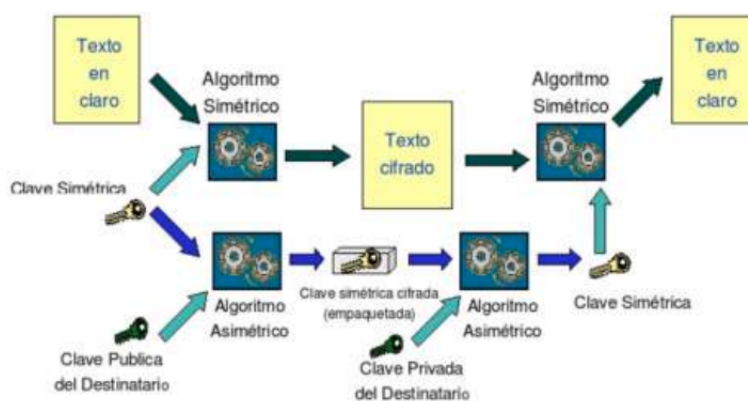


Figura 4: Algoritmo híbrido (A., 2018)

Por último, los algoritmos propios son aquellos creados por cada institución, para asegurar la protección de su información debido a la desconfianza en algoritmos de cifrado existentes y reconocidos a nivel mundial. (A., 2018). No obstante, cabe mencionar que si estos algoritmos de cifrado propios no son realizados de manera robusta, son susceptibles a una mayor cantidad de ataques cibernéticos.

Debido al funcionamiento propio del IoT, la protección y privacidad de los datos son fundamentales en estos sistemas por lo que el uso de algoritmos criptográficos es indispensable. Estos pueden ser utilizados ya sea para la protección de los canales de comunicación así como para el cifrado y descifrado de la información. (E., 2022). Existe una relación directa entre el algoritmo implementado, tomando en consideración los recursos que se tengan disponibles, y la seguridad general del sistema. En ocasiones, se opta por utilizar métodos criptográficos ligeros ya que equilibran el rendimiento y el consumo de energía junto con el espacio físico aunque no sean tan seguros como estándares criptográficos principales. (E., 2022). La criptografía ligera se utiliza en dispositivos que cuentan con recursos limitados, como lo son en IoT; suelen usarse estos algoritmos en etiquetas RFID y en redes de sensores inalámbricas (WSN) además de que pueden ser aplicados tanto a dispositivos de hardware como en software. (E., 2022).

Dentro de la criptografía ligera, también pueden dividirse los algoritmos entre simétricos y asimétricos. (E., 2022). A su vez, los algoritmos ligeros simétricos se pueden categorizar en cifrados de bloque, de flujo y funciones hash. Los cifrados de bloque pueden considerarse como un reemplazo del algoritmo AES, obteniendo un nivel de seguridad similar y con menos limitaciones. Entre estos, se encuentran PRESENT, CLEFIA, y SIMON. Los cifrados de flujo presentan altos niveles de seguridad y rendimiento así como un bajo coste; entre estos, están GRAIN, TRIVIUM y MICKEY. Por último, las funciones hash son usadas para crear resúmenes criptográficos de la información; algunos ejemplos son PHOTON y SPONGENT. (E., 2022).

Por otro lado, los algoritmos ligeros asimétricos son altamente recomendables para los dispositivos con limitaciones. Entre los algoritmos más usados se encuentra BlueJay y la curva elíptica ligera debido a la relación que hay entre nivel de seguridad, rapidez y tamaño. BlueJay utiliza alrededor de 2000-3000 GE y es empleado principalmente cuando se busca recibir información de sensores, autenticación en RFID y aplicaciones de únicamente clave pública. (E., 2022). Por otro lado, las curvas elípticas ligeras son consideradas como el algoritmo de clave pública más eficiente puesto que consume menos energía, un área más pequeña y menos ciclos de reloj. Este algoritmo está basado en las matemáticas de las curvas elípticas y usa claves generadas por un algoritmo discreto pero la mayoría de los diseños requieren más de 10,000 equivalencias de puerta (GE). (E., 2022).

Criptografía ligera	Algoritmos
Simétrica	CLEFIA, SIMON, TRIVIUM, PHOTON
Asimétrica	BlueJay, ECC (curvas elípticas ligeras)

Cuadro 6: Algoritmos de criptografía ligera. (E., 2022).

2.6.1. Comparación entre algoritmos de clave pública para IoT.

De acuerdo a la investigación realizada, en los dispositivos IoT se utilizan esquemas de clave pública para la protección de los datos y la red en sí. De igual manera, se menciona una afinidad hacia la criptografía ligera debido a su relación entre tamaño y seguridad. A continuación, se muestran tres algoritmos de clave pública, siendo estos de criptografía ligera, con la finalidad de realizar una comparación y definir el esquema a utilizar en la propuesta .

Referencia	Algoritmo	Ventajas	Desventajas	Tamaño de claves
(O. and Saarinen, 2012)	BlueJay	La encriptación es significativamente más rápida que el RSA. La operación de clave pública es más rápida que el RSA y 2-3 veces más rápida que el ECC. Es planeado para adquirir datos de sensores, autenticación RFID y un logging seguro. El tamaño del hardware es menor que 3000 GE, además de la memoria usada para almacenar la carga útil e información de la clave pública.	Su private key operation cuenta con una demanda similar al RSA. No se cuenta con mucha documentación al respecto.	1024-bit clave pública módulo n y 32-bit del tamaño de registro.
(S., 2017)	RSA	Clave de tamaño variable; la encriptación es más rápida que la desencriptación y la verificación es más rápida que la firma. Para romper el encriptado, se debe tener un algoritmo capaz de factorizar grandes números.	Lento al cifrar grandes volúmenes de datos, emplea operaciones matemáticas de alto costo y trabaja con claves de gran tamaño.	Entre 1024 y 4096 bits.
(Seabrooke) (P., 2021)	Curvas elípticas	Clave de cifrado corta. Un cifrado con CE de 160 bits es proporcional a la seguridad de un RSA de 1024 bits, por lo que se vuelve mas eficiente.	El algoritmo tiene una dificultad de implementación más alta	Entre 160 y 521 bits.

Cuadro 7: Algoritmos de clave pública

2.7. Recursos disponibles

Generalmente los componentes de hardware y software utilizados para aplicaciones en IoT son diseñados bajo un protocolo estándar. Por ejemplo, algunas plataformas como Raspberry y Arduino son especialmente útiles en etapas de prototipado y agilizar el tiempo necesario para implementar una configuración IoT.

Particularmente, la Raspberry Pi es un *computer board* de bajo costo y portable que se caracteriza por su

capacidad de expandirse por medio de placas de expansión apilables. Éste pequeño ordenador opera con Raspberry Pi OS basado en Linux y permite la implementación de distintos lenguajes de programación, aunque Python es el lenguaje número 1.

Dado que la Raspberry funciona como un microordenador, se requiere de una red eléctrica de una pantalla, monitor o televisión, para hacerla funcionar y, de preferencia, también un teclado y mouse.

Entre otros componentes importantes a tomar en cuenta, se encuentran los sensores, que funcionan como actuadores que obtienen la información, en este caso proveniente de una herramienta inteligente como un *smart meter*, y la transmiten de manera segura y encriptada a través de un proceso interno hasta el centro de control.

Asimismo, existen en el mercado una amplia variedad de servicios que facilitan el diseño y ejecución de un modelo IoT dependiendo de su arquitectura. En una arquitectura en la nube dentro de los más utilizados se encuentran Azure, AWS y Google Cloud. La implementación de alguno de estos servicios permite agilizar en gran medida procesos como la generación de certificados y claves públicas.

3. Propuesta

Tomando en cuenta que al trabajar con dispositivos IoT se necesitan esquemas de clave pública livianos que consuman la menor cantidad de poder computacional y memoria, se han comparado los tres algoritmos mencionados anteriormente con el fin de escoger cuál de ellos es la mejor opción para el objetivo de este trabajo. En primer lugar, el algoritmo RSA es el que tiene el mayor tamaño de sus claves, lo que lo hace menos eficiente. En cambio, entre BlueJay y Curvas Elípticas, investigaciones muestran que el primero de estos se está posicionando como uno de los algoritmos más eficientes para clave pública. No obstante, la falta de documentación al respecto representa una gran desventajas frente a Curvas elípticas, pues su desarrollo e implementación dentro de este trabajo se vuelve muy complejo.

Por esto, se propone desarrollar un algoritmo de curvas elípticas al tener claves de cifrado bastante cortas a comparación de los otros dos esquemas, y además sustentarse en varios trabajos de investigación previos. Además, considerando los resultados obtenidos en la investigación de (L. et al., 2020), y su comparación con el resto de trabajos, se ha decidido utilizar una arquitectura Edge cloud.

Así también, la función del auditor se dará a través de una Raspberry Pi 4 instalada en una máquina virtual; esto con la finalidad de facilitar el proceso pero de igual manera se podría trabajar con una tarjeta física. El auditor recolectará los datos del *smart meter* y *smart inverter*, enviándolos al centro de control, y regresando la información hacia el *smart inverter* para controlar la producción de energía eléctrica. El Raspberry se encuentra en la parte *edge* de la arquitectura.

Para la base de datos y considerando la investigación (AWS, s.f.b), buscamos utilizar servicios de AWS. En nuestro caso utilizamos Amazon RDS, el cual es una base de datos que trabaja con SQL. Es gratuita siempre y cuando se trabaje con menos de 20 GigaBytes de información. Se optó por esta solución debido a que Amazon RDS automatiza diversos procesos administrativos así como la escalabilidad que ofrece, además se va a encriptar a través del estándar AES-256. Para fines académicos, dicha herramienta es más que suficiente en términos de espacio necesario. Debido a que el modelo operacional todavía sigue en desarrollo y las cargas de trabajo pueden variar dependiendo de la cantidad de datos intercambiados, AWS proporciona un modo de asignación de recursos bajo demanda. Esta versión permite al administrador pagar solo por la cantidad de recursos que usa.

Además de esto, AWS proporciona un cifrado de datos en reposo, con lo cual se garantiza que una vez que la base de datos sea creada y conforme se vaya almacenando información esta sea accesible solo para quienes poseen la clave de acceso. Además, el código para la encriptación será hecho en Python y se enlazará con el servidor de la base de datos. (AWS, s.f.a). La base de datos a utilizar es la proporcionada por la organización socio formadora, donde se tienen registros cada 15 minutos. Para la encriptación y traslado de la información, se estará mandando únicamente una sola línea a la vez.

Finalmente, con relación a los protocolos que se estarán ocupando, se considera principalmente TLS y HTTPS. Se opta por estos protocolos para poder tener un algoritmo que asegure que el mensaje que se está mandando no sea *plain text* además de asegurar un ambiente encriptado. Además, TLS es capaz de utilizar tanto RSA como curvas elípticas, donde viene siendo de utilidad el programa previamente mencionado. Esta parte se realiza en conjunto con la librería de *SSL* en Python y *OpenSSL* para Linux. (M., 2022).

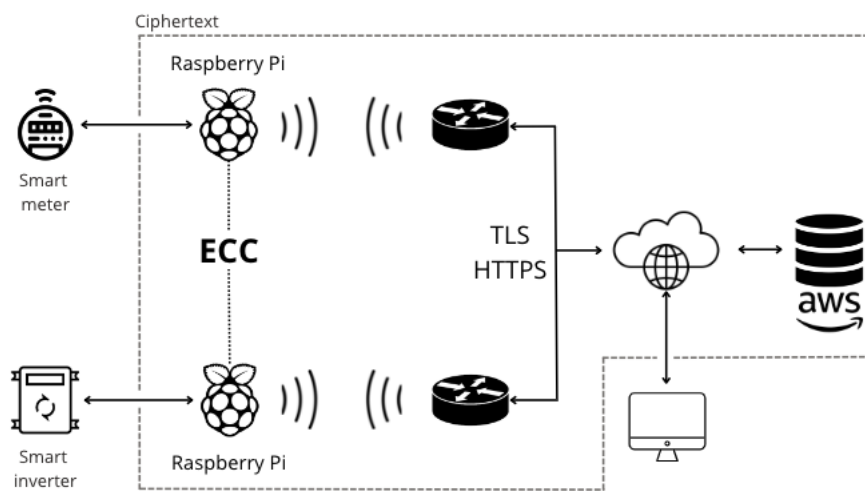


Figura 5: Esquema de modelo IoT propuesto

4. Metodología

Respecto a componentes de la arquitectura, el auditor es una máquina virtual implementada en Virtual Box que funciona con el sistema operativo Linux Debian. Esta máquina virtual emula una computadora RaspberryPi. Sus principales tareas son recolección de los datos, almacenamiento temporal de los mismos, buscar servidor para enlazar conexión TLS, envío de datos y solicitud de datos. Cabe destacar que para el envío su primer paso es hacer el firmado de ECDSA par obtener la trama a enviar, así como para la solicitud de los datos al momento de ser recibidos será necesario primero hacer la verificación ECDSA antes de mostrarlos.

Por otro lado, el servidor fue implementado en una instancia EC2 de AWS de sistema operativo Linux. Sus tareas son establecer la conexión TLS, recopilar la información obtenida por el auditor para el cual primero se tiene que hacer la verificación, ejecutar la acción de insertar datos en la instancia rds y enviar los datos solicitados por el cliente. Para dicha solicitud consiste en los pasos de hacer la verificación del query recibido, ejecutarlo en la instancia rds para seleccionar los renglones, firmar los datos obtenidos y enviarlos.

4.1. Conexión TLS

Para la configuración del protocolo TLS, se utilizó la librería de Python *SSL* que proporciona acceso a los módulos de encriptación y autenticación en una red basada en *sockets* para cliente y servidor. Además, es importante tomar en cuenta que los certificados de autenticación son generados por en instancia con con *OpenSSL* a través de curvas elípticas. Los certificados son generados en la instancia siendo los archivos de la central de autenticado, tanto su certificado .pem y .key, donde el primero viene siendo el archivo de certificado firmado y el segundo archivo es la clave privada.

Para lograr la conexión cliente-servidor como se muestra en la figura 6 se creó un socket en cada parte con la finalidad de poder intercambiar datos bidireccional. Para esto se utiliza el puerto TCP, pero es necesario asegurarse que todos los puertos en la máquina virtual de EC2 estén correctamente configurados. Lo anterior se logra a través de agregar nuevas reglas de seguridad a security groups donde se establezcan permisos para conexiones por HTTP y HTTPS.

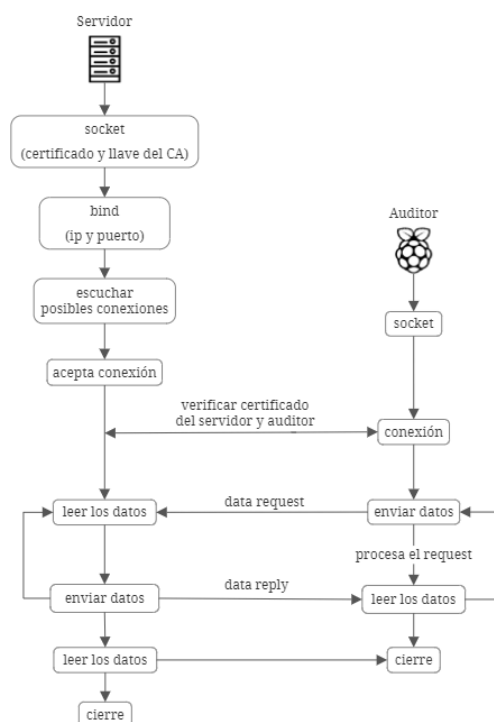


Figura 6: Conexión por sockets con TLS

4.2. ECDSA

Para el firmado y verificación se utilizó el algoritmo de curvas elípticas para firma digital (ECDSA), se creó una curva elíptica de 256-bit a través de la librería *tinyec* de Python, la cual tiene una llave privada generada de manera aleatoria de 256-bit y la llave pública es de 257-bit, debido a la compresión. Además, se el mensaje para este caso es un string de pandas transformada a bits.

Con los parámetros de la curva y el mensaje se generó la firma digital, a través del generador, elemento, llave privada y el hash del mensaje, en esta función se retornan los números r y s , además de la llave pública y el hash. Como se observa en el diagrama 7.

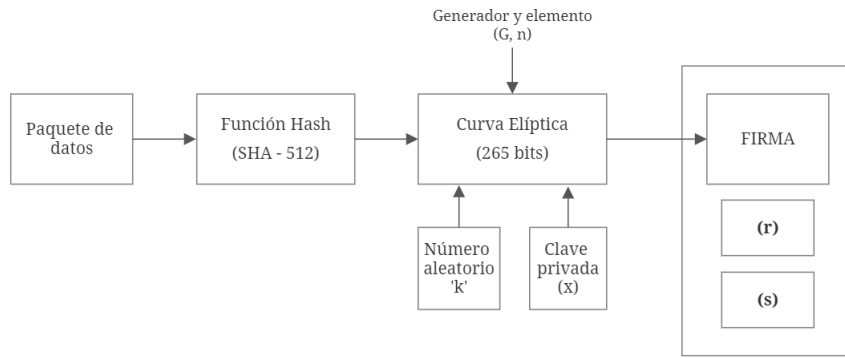


Figura 7: Generación de firma por ECDSA

Para la verificación de firma digital, el algoritmo primero recibe como input el mensaje y calcula su *hash value* utilizando la misma función HASH utilizada para el firmado, junto con la misma firma digital $\{r, s\}$ generada previamente. Adicionalmente recibe la clave pública (Q) que corresponde a la llave privada de quien firma. Internamente el algoritmo decodifica el valor 's' de la firma de regreso a su punto de origen en la curva elíptica (R) haciendo uso la clave pública junto con el *hash value*, de este modo compara la coordenada-x del punto recuperado (R) y lo compara con el valor 'r' de la firma digital. Finalmente el output que se obtiene es un valor booleano que indica si la firma es aceptada o rechazada. (Fig. 8)

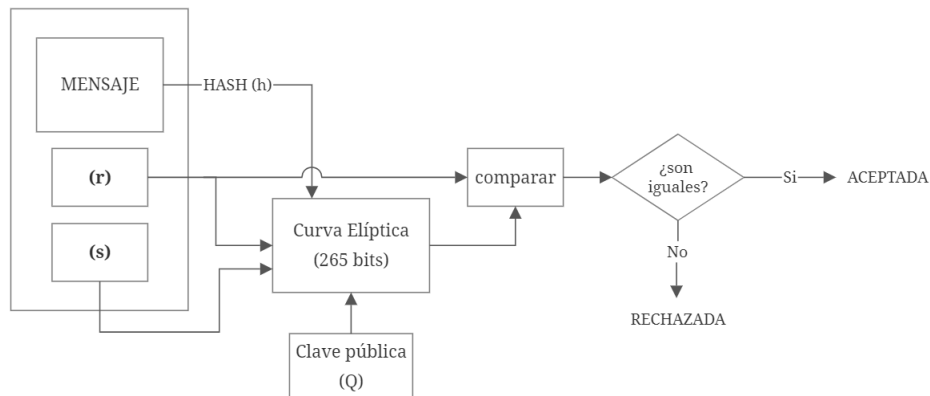


Figura 8: Verificación de firma por ECDSA

4.3. Base de datos entrada (auditor)

Actualmente, esta consiste en un archivo .csv que ha sido proporcionado por LiCore, se lee dentro de python como un DataFrame y se enviaría por pedazos respecto a los que se le indicará al auditor a estar enviando. Cabe destacar que cada uno de estos pedazos a enviar. Los cuales a su vez van a generar a las tramas, las cuales contienen la información sobre los parámetros de verificación de r y s, la llave pública y el mensaje.

4.4. Bases de datos servidor

Para la creación de una instancia en el servicio de administración de bases de datos de AWS en la nube, RDS, se recomienda que el usuario que tenga acceso a los servicios sea el administrador de la cuenta o que se cree un perfil de usuario IAM y se otorguen los permisos necesarios para consultar y configurar las instancias dentro del servicio. Para nuestro caso, todo el procedimiento fue creado con el usuario

administrador de la cuenta. El primer paso consistió en la creación de una instancia en RDS que tuviera las características del free tier, las cuales Amazon establece en 750 horas de uso al mes, usando memoria de uso general gp2, un tipo de instancia t2 o t3 micro y un máximo de 20 gb de almacenamiento. También se utilizó MYSQL debido a que es uno de los gestores de bases de datos relacionales permitidos.

Además de las características mencionadas, es conveniente que la base de datos desde un inicio sea enlazada a una instancia en EC2 para asegurarse de que la conexión sea exitosa. Sin embargo, nosotros optamos por enlazar varias instancias de EC2 después de la creación de la instancia en RDS.

5. Desarrollo

5.1. Pasos para llegar a Resultado finales

a. Creación de instancia de EC2, siguiendo la documentación AWS (2023), lo primero establecer el sistema operativo de linux 1, dar el tipo de instancia t2 para tener una capa gratuita, generar un key pair, que se utilizará para las conexiones de ssh. Habilitar el security group y abrir los puertos de https, ssh y http, luego mantener el volumen de almacenamiento predeterminado.

b. Crear certificados de Central de Autenticado siguiendo el blog de GoLinuxCloud (2023). Crea los archivos de registro de los certificados de la CA. Con el archivo de configuración de *openssl.cnf* crear llave privada de ECC prime256v1. Y el certificado a través del estándar x509. Se observa en 9 tener la clave correcta el certificado, así como sus parámetros.

```
ec2-user@ip-172-31-90-101 tls]$ openssl x509 -noout -text -in certs/ec-cacert.pem | grep -i algorithm
        Signature Algorithm: ecdsa-with-SHA256
        Public Key Algorithm: id-ecPublicKey
        Signature Algorithm: ecdsa-with-SHA256
ec2-user@ip-172-31-90-101 tls]$ openssl x509 -noout -pubkey -in certs/ec-cacert.pem
-----BEGIN PUBLIC KEY-----
MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAERHq1bu9o97Re5R4TWNGsen+aoJ9f
Ulh/EwcIgvAgiw5Kx4TdeSxwBsJLb9+Mvnjkxcz15q7C9VS/jgheja/eog==
-----END PUBLIC KEY-----
ec2-user@ip-172-31-90-101 tls]$ openssl pkey -pubout -in private/ec-cakey.pem
-----BEGIN PUBLIC KEY-----
MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAERHq1bu9o97Re5R4TWNGsen+aoJ9f
Ulh/EwcIgvAgiw5Kx4TdeSxwBsJLb9+Mvnjkxcz15q7C9VS/jgheja/eog==
-----END PUBLIC KEY-----
ec2-user@ip-172-31-90-101 tls]$
```

Figura 9: Generación Certificados

c. Para lograr pasar proporcionar el certificado de la CA a la máquina virtual se estaría utilizando ssh, la cual sería una conexión establecida autenticada por el key pair obtenido de la ec2. El cual a su vez llego al auditor a través de una carpeta compartida entre la computadora local desde donde se creo la instancia y la MV. La decisión de establecer conexión se muestra en 10.

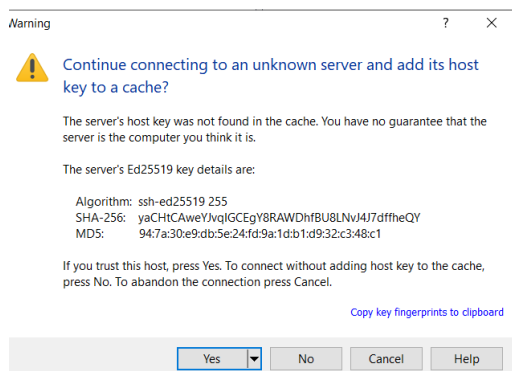
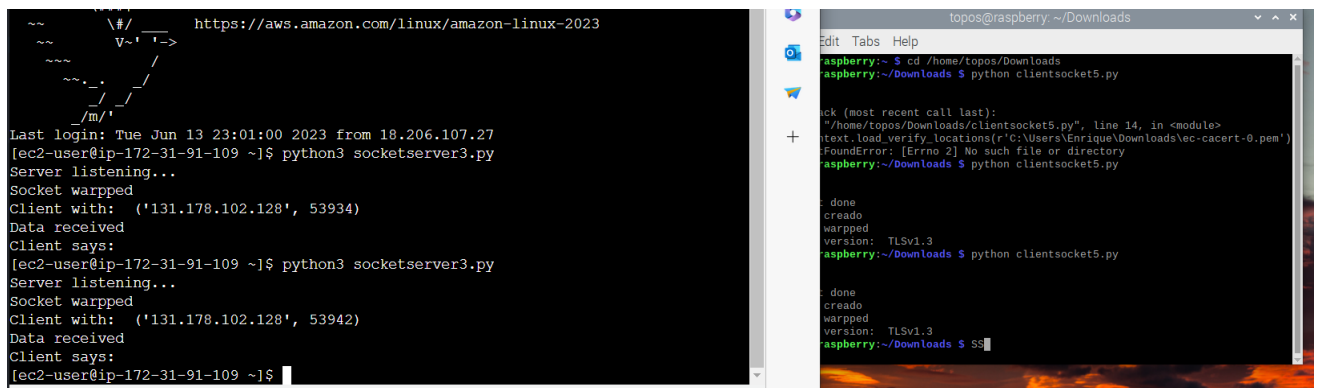


Figura 10: ssh conexión

d. Instalación de la Raspberry Pi en una máquina virtual. Donde se utilizó el siguiente artículo TheRoboticsBack-End (2023) a través del que se definió un sistema operativo de Debian, así como ir seleccionando un espacio de memoria recomendado para este caso se estaría haciendo la recomendación de 1024 MB. Además, de seleccionar una locación específica para tu disco duro al igual que un tamaño de memoria. Respecto al procesador, este estaría siendo sobre la cantidad de CPUs a utilizar los cuales en general tendrían que ser de 1.

e. Para la generación de sockets se siguió la documentación de Python Python (2023), los cuales estarían siendo utilizados para generar la conexión de TLS/SSL, ya que ellos permitirían aplicar las funciones que requiere hacer tanto el cliente como el servidor para generar un proceso de Handshake. En el que se estaría permitiendo ingresar la versión del protocolo a utilizar, así como los certificados para el la autenticación de usuario.

Cabe destacar que en este paso los errores pueden ser por una gran diversidad de causas como los puertos abiertos, la conexión a internet, tipo de certificado, versiones de software, entre otros por lo que en este paso fue de mucha utilidad resolverlo por partes. De modo que, primero se estableció la conexión sin TLS local, luego entre VM y EC2 y con después la misma con TLS 11.



The image contains two side-by-side terminal windows. The left window shows an SSH session from an EC2 instance to a Raspberry Pi. It displays the login banner, the last login time, and the execution of a Python script that acts as a socket server, receiving connections from the Raspberry Pi. The right window shows a terminal on the Raspberry Pi where a Python script named 'clientsocket5.py' is being executed. The script attempts to connect to the EC2 instance, but it fails with a 'FileNotFoundError' because it cannot find a local certificate file 'cacert-0.pem'.

Figura 11: conexión básica tls vm ec2

f. Para la creación del algoritmo validación de ECDSA, agregar un poco sobre la experimentación que se realizó el código a través de la creación de funciones del cálculo del inverso multiplicativo, así como las operaciones para obtener los parámetros de r y s. Entre lo que cabe mencionar la forma en que se obtiene el valor de k, por medio del algoritmo determinista planteado en el informe RFC6979 (August 2013). Donde además, se realiza la comprobación en los vectores de prueba sobre el parámetro correspondiente a la curva 256 y función de firmado de SHA-256. Lo cual se puede observar en la figura 12.

```
q = int('0xffffffff00000000fffffffffffffffffbce6faada7179e84f389cac2fc632551',0)
x = int('0xc9afa9d845ba75166b5c21576781d6934e50c3d836e89b12788a6228120f6721',0) # priv key
ux = int('0x60fed48a255a9d31c961eb74c6356d68c04988923861fa6ce669622e60f29f86',0)
uy = int('0x7903fe100888c99a41ae9e95628bc64f2f1b20c2d7e9f5177a3c294d4462299',0)
m = "sample".encode()
curve = registry.get_curve('brainpoolP256r1')
#hash = Lambda x: hashlib.sha1(x.encode())
k = '0xA6E3C57DD01ABE9088653839835DD4C3B17AA87338280F24D6129493D8AAD60'
r = '0xEF4882AACB6A8FD1140DD9CD45E81D69D2C877B56AAF991C34D0EA84EAF3716'
s = '0xF7CB1C942D657C41D436C7A1B6E29F65F3E900DB89AFF4064DC4AB2F843ACDA8'

k_calc = deterministic_k(q, x, m, hashlib.sha256)
print('k = ', k_calc)
if int(k, 0) == k_calc:
    print('mismos valores')

k = 75486370184466523516702714224272210659255809472406410223340475427961162083680
mismos valores
```

Figura 12: RFC prueba de msimsa k

Así mismo se inicio realizando un código para la generación de parámetros de r y s , para la verificación, el cual estaría cambiando para el documento final al este estar requiriendo modificaciones en las operaciones del grupo de la curva elíptica. De tal modo que se pudiera estar realizando la comprobación de tener los mismos resultados que las pruebas de testeo y ejemplo mencionadas en la sección A.2.5 de RFC6979 (August 2013). Por lo tanto se hizo una adaptación al código original de Pégourié-Gonnard (2023). Con el que se estaría ingresando el valor de determinado anteriormente de k y con ello tendríamos el resultado de obtener los mismos parámetros r y s 14.

```

        r=0xF1AB8023518351CD71D881567B1EA663ED3EFCF6C5132B354F28D3B087D38367,
        s=0x019F4113742A2B14BD25926849C649155F267E60D3814B4C0CC84250E46F0083,
    )
)

else:
    print('Error: diferente valor de k')

Ok:same k

: if __name__ == '__main__':
    print("P-256 ECDSA test vectors from RFC 6979 A.2.5...",
          end=' ', flush=True)

    # key generation
    tv = tv_ecdsa_rfc6979_key
    signer = EcdsaSigner(p256, tv['x'])
    pub = signer.public_key()
    assert(tv['ux'] == int(pub.x()))
    assert(tv['uy'] == int(pub.y()))
    verif = EcdsaVerifier(p256, pub)

    # signature generation and verification
    for tv in tv_ecdsa_rfc6979:
        h, k, r, s = tv['h'], tv['k'], tv['r'], tv['s']
        sig = signer.sign(h, k)
        assert(sig == (r, s))
        assert(verif.is_valid((r, s), h) is True)
        assert(verif.is_valid((r+1, s), h) is False)
        assert(verif.is_valid((r, s+1), h) is False)
        assert(verif.is_valid((r, s), h[::-1]) is False)
    print('Ok:same r and s')

P-256 ECDSA test vectors from RFC 6979 A.2.5... Ok:same r and s

```

Figura 13: RFC vectores de prueba 1

De modo que estas serían las nuevas funciones para firmado y verificación. Y con lo que al mismo tiempo se estarían realizando los vectores de prueba con toda la base de datos.

```

#primero con toda la base de datos
m_df = pickle.dumps(dataset, protocol=4)
k_det_df = deterministic_k(q, x, m_df, hashlib.sha256)
h=hashlib.sha256(m_df).digest()

r, s = signer.sign(h, k_det_df)
assert(verif.is_valid((r, s), h) is True)
assert(verif.is_valid((r+1, s), h) is False)
assert(verif.is_valid((r, s+1), h) is False)
assert(verif.is_valid((r, s), h[::-1]) is False)
print('Firma verificada')

Firma verificada

```

Figura 14: RFC vectores de prueba 2

g. Los siguientes pasos consisten en definir la forma de envío de información, dando así la recolección de tramas a enviar y definiendo el tamaño de buffer correcto. En este punto se generaron muchos bugs que afectaban para la validación del mensaje enviado. Como se muestra en 15

```

Socket conectado
Actions: socket
# Send (F)
# Receive (V)
# End
Select your action:
F
14966
Traceback (most recent call last):
  File "C:\Users\Enrique\OneDrive - Instituto Tecnológico y de Estudios Superiores de Monterrey\Documents\Escuela\F23\Algebra moderna Cripto\reto\Reto-Git\LiCore-Crypto-IoT\Client_Server\ecdsa_client.py", line 181, in <module>
    client_program(dataframe, host_public, port)
  File "C:\Users\Enrique\OneDrive - Instituto Tecnológico y de Estudios Superiores de Monterrey\Documents\Escuela\F23\Algebra moderna Cripto\reto\Reto-Git\LiCore-Crypto-IoT\Client_Server\ecdsa_client.py", line 108, in info_exchange
    info_exchange(client_socket, m)
  File "C:\Users\Enrique\OneDrive - Instituto Tecnológico y de Estudios Superiores de Monterrey\Documents\Escuela\F23\Algebra moderna Cripto\reto\Reto-Git\LiCore-Crypto-IoT\Client_Server\ecdsa_client.py", line 144, in client_program
    ver = client_socket.recv(65507).decode() # recibir verificación
ConnectionResetError: [WinError 10054] An existing connection was forcibly closed by the remote host

[ec2-user@ip-172-31-91-109 ~]$ python3 ecdsa_server.py
ip-172-31-91-109.ec2.internal
bind socket
Socket is listening...
Connection from: ('189.218.7.17', 51498)

Cliente: F

Longitud: 7180
Traceback (most recent call last):
  File "/home/ec2-user/ecdsa_server.py", line 243, in <module>
    server_program(host_private, port)
  File "/home/ec2-user/ecdsa_server.py", line 202, in server_program
    info_exchange(conn)
  File "/home/ec2-user/ecdsa_server.py", line 162, in info_exchange
    par = pickle.loads(mess1, encoding='bytes')
pickle.UnpicklingError: pickle data was truncated
[ec2-user@ip-172-31-91-109 ~]$ python3 ecdsa_server.py

```

Figura 15: error mientras verificación

h. Se crea la base de datos, así como las tablas donde se guarda la información. En este caso se utilizó un almacenamiento de 20GiB 16. Además, para el código de insertar información en el SQL nos basamos en la información proporcionada por Godalle (2023).

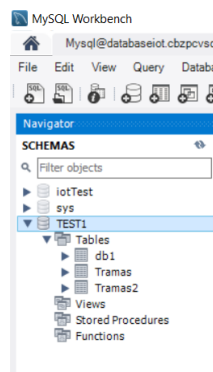


Figura 16: MySQL DB

i. Finalmente, se junta todo el código, de forma que se establece conexión bidireccional. Como es que se muestra en la 17 Además, en este ya también se sube en rds desde el servidor, después de haber sido verificado y se descarga de esta instancia, cuando se pide la información.

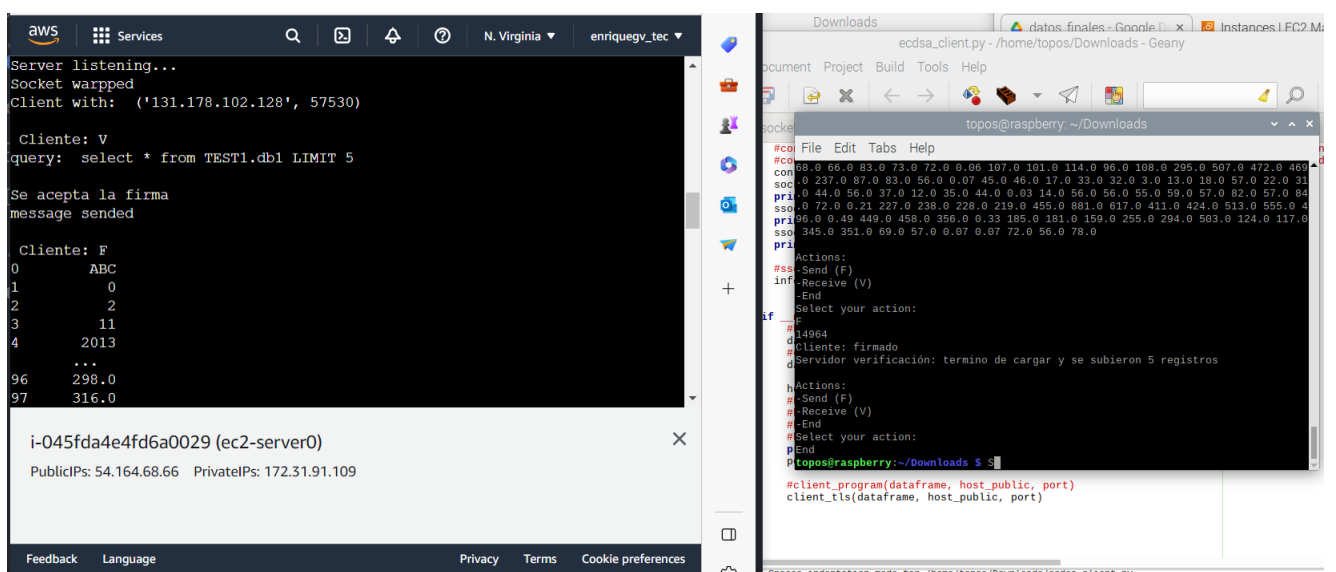


Figura 17: Firmado bidireccional ec2-vm

5.2. Experimentación adicional

A continuación se muestra una serie de pasos que no son necesarios para obtener la solución final, pero representaron parte de la experimentación.

Para la instancia de EC2, hubo una problemática de costos iniciales, al haberse pasado del límite de volumen, ya que se solo se permite hasta 30GiB. Además, se realizó configuración de servidor apache, sin embargo no se dio continuidad.

Instalación del servidor web con *Apache Server* (Fig. 18) dentro del entorno cloud que asignará la instancia de AWS como host. Así mismo, se generó la instancia RDS, sin embargo aun no hay la conexión con la EC2 (Fig. 19).

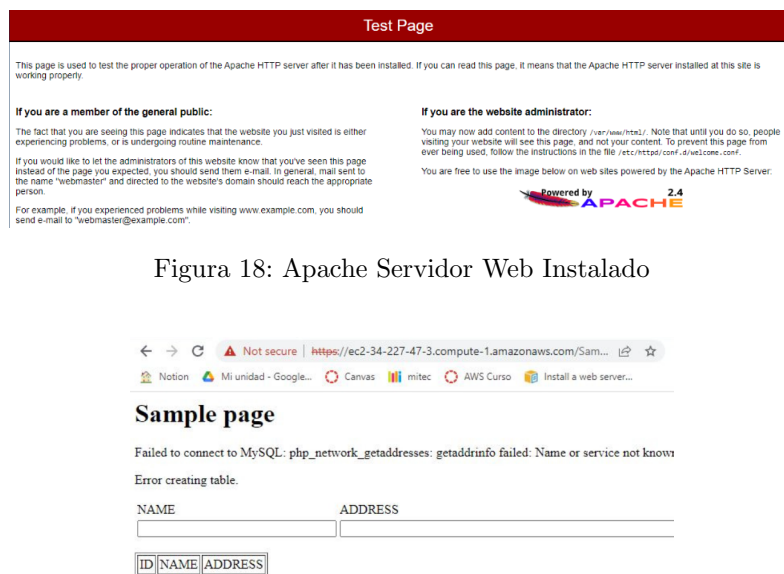


Figura 18: Apache Servidor Web Instalado



Figura 19: Instancia RDS conexión faltante

Se creo un protocolo Diffie–Hellman simétrico con EC y AES para el key exchange, sin embargo, solo se ha utilizado TLS para la encriptación de mensajes. Esto debido a que se requerían de más pruebas para verificar su confidencialidad, ya que se estaban utilizando APIs, además de la experimentación sobre el impacto que tendría en el tiempo computacional.

Primera conexión exitosa, envío de archivos local a EC2 con TLS y se hace comprobación en Wireshark 20.

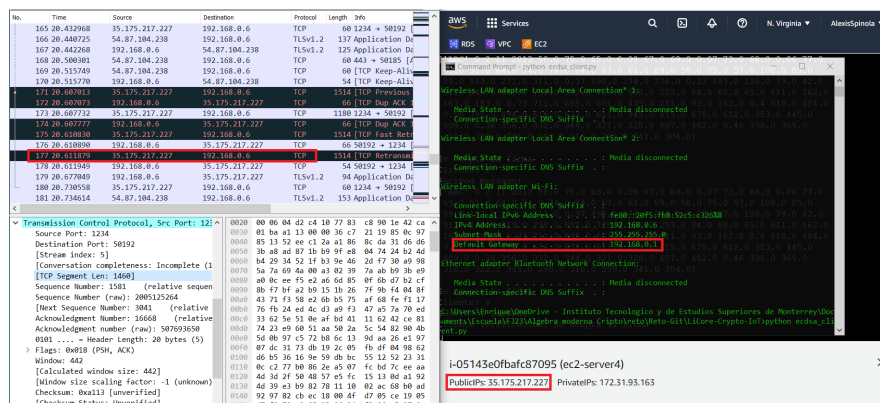


Figura 20: WS local EC2

Esta conexión ya se habían generado correctamente los sockets. Sin embargo, se apago la instancia y con ello la direcciones ip cambian, de modo que se tuvo que volver a realizar los certificados, una nueva instancia y correcciones en el código de los sockets.

6. Resultados finales

Finalmente, obtenemos el prototipo de una conexión bidireccional con arquitectura Cloud/Edge, que garantiza el criterio de autenticación por medio del protocolo TLS, así como la verificación de mensajes por medio de ECDSA. Así como el almacenamiento seguro de los dato en SQL encriptado con AES. Se puede observar su esquema final en 21

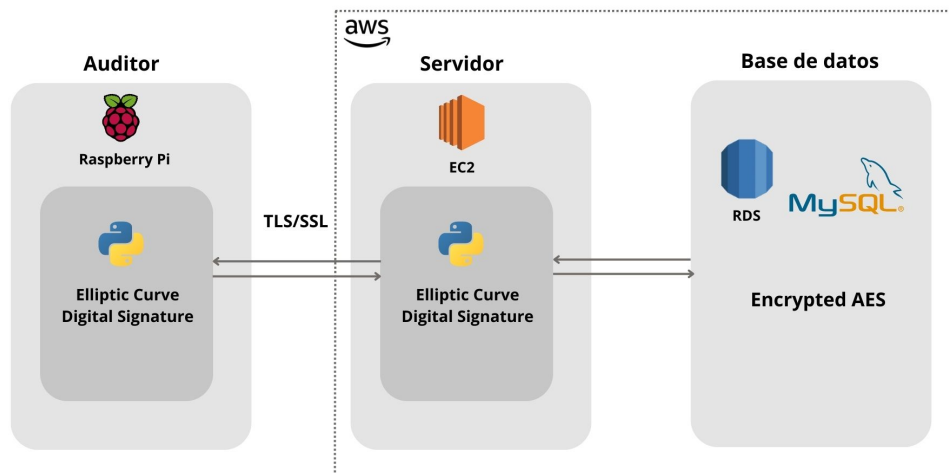


Figura 21: Esquema de red

Así como el flujo de intercambio de información que se estaría realizando a través del esquema de toda la red, se estaría mostrando en el diagrama 22.

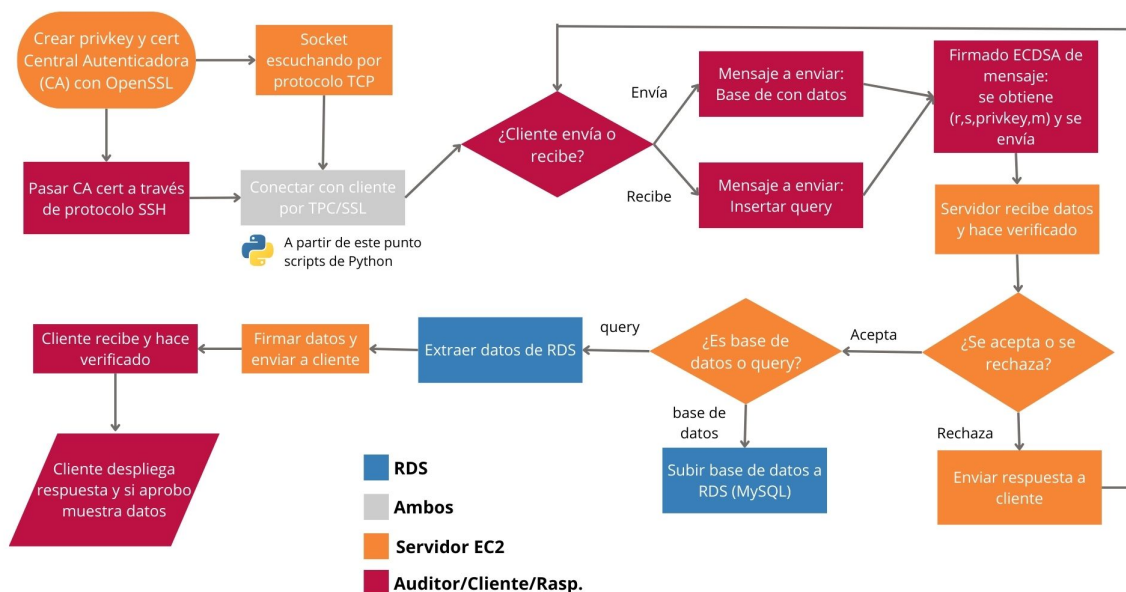


Figura 22: Flujo de mensaje

Los resultados pueden observarse en los siguientes enlaces:

- **Github:** <https://github.com/enriquegv001/LiCore-Crypto-IoT.git>

- Video puesto en práctica la comunicación: <https://youtu.be/dcRZlVgVFWk>

7. Conclusiones y trabajo a futuro

En conclusión se ha podido resolver el objetivo de intercambio de datos de forma segura para una red de energía distribuida, de forma que se ha generado un impacto de volar el desarrollo ecológico y tecnológico dirigido hacia los ODS 7, 13, 9 y 11.

Puntualizando, ha habido beneficios al contraponer retos la ética en ciber-seguridad, como fue la autenticación, privacidad de la red y usuarios, almacenamiento y respaldo de datos y pruebas de testeo para el diseño de los productos IoT.

A partir de esto, se recomienda como trabajo a futuro destinar un budget específico dependiendo de la cantidad de auditores que se van a tener, lo cual afectaría a los precios y características de la EC2 y RDS a contratar. Se recomendaría comenzar probando con un servidor tipo t3.small al necesitar este estar prendido todo el tiempo, pero además que logre soportar pocas conexiones, el cual cuesta \$0.014. Así como por parte de la base de datos que sea de tipo g3 teniendo un costo de 0.08 usd el MB/s-mes lo que es útil al estar siendo una página web promedio de 25k. De modo que el aproximado anual sea \$125.05.

El siguiente paso sería cambiar de la raspberry pi como máquina virtual a una tarjeta física. Además, de comprar un dominio actual, asociarlo al servidor y obtener certificados a través de CA gratuita como Let's Encrypt. Evitando así, que estos sean auto-firmados. Después, Habilitar más que solo a los usuarios a conectarse, el cliente no solo será el auditor en si. Y generar una integración con app para la visualización de datos y análisis. Igualmente, tomando en cuenta los comentarios del socio formador, queda como área de oportunidad la incorporación de un proceso de automatización para la ejecución de procesos y tareas correspondientes al proyecto desde la bash de del cliente. Esto con el objetivo que de acuerdo a un cierto tiempo las raspberry suban de forma automática a la base de datos los registros actualizados y validen de forma criptográfica que los datos están seguros.

De esta forma, quedaría cubierto los riesgos que se tienen actualmente, como sería el certificado auto firmado que mientras no se trabaje fuera de una red local este puede ser de baja seguridad. Así como posibles bugs que vayan surgiendo en el código. Y con el planteamiento dado, desarrollo y documentación se podrían mitigar costos tanto por la parte de servicios de Clouding como en el impacto de los algoritmos sobre el tiempo computacional.

Referencias

- Martínez A. and Razo I. Socio formador primer sesión ma2006bgx. URL https://drive.google.com/file/d/1R3D62Edw6VKyjQpZUjCSb8_GGWDLw7C3/view?usp=share_link.
- Samaniego Zanabria A. Evaluación de algoritmos criptográficos para mejorar la seguridad en la comunicación y almacenamiento de la información. 2018.
- AWS. Tutorial: Get started with amazon ec2 linux instances. 2023. URL https://docs.aws.amazon.com/es_es/AWSEC2/latest/UserGuide/EC2_GetStarted.html.
- AWS. Características de amazon dynamodb. s.f.a. URL <https://aws.amazon.com/es/dynamodb/features/#Serverless>.
- AWS. What is aws iot greengrass? s.f.b. URL https://docs.aws.amazon.com/greengrass/v2/developerguide/what-is-iot-greengrass.html?nc2=type_a.
- Cheminet C. *Energía Informacional-Smart Metering*. PhD thesis, Universidad de Buenos Aires, 2018.
- Zhao C., Jegatheesan J., and Loon S. Exploring iot application using raspberry pi. *International Journal of Computer Networks and Applications*, 2(1):27–34, 2015.
- Ortiz Sánchez D. Implantación de la tecnología blockchain en infraestructura serverless. *Universidad Complutense de Madrid*, 2019.
- Centro Tecnológico del Calzado de La Rioja. Blockchain. URL <https://www.ctcr.es/es/proyectos/2693-blockchain>.
- Gordillo Vega et al E. Técnicas criptográficas ligeras para dispositivos iot. 2022.
- Balmaseda-Aranda F. Aseguramiento de dispositivos iot con blockchain e infraestructura de clave pública. Master's thesis, 2018.
- Ed Godalle. How to connect to mysql using python and import the csv file into mysql and create a table? 2023. URL <https://www.projectpro.io/recipes/connect-mysql-python-and-import-csv-file-into-mysql-and-create-table#:~:text=Table%20of%20Contents%201%20Step%201%3A%20Prepare%20the,5%20Step%205%20%3A%20Query%20the%20Table%20>.
- GoLinuxCloud. Openssl: Generate ecc certificate verify on apache server. 2023. URL https://www.golinuxcloud.com/openssl-generate-ecc-certificate/#5_Create_CA_certificate_with_ECC_Key.
- IBM. What is networking? URL <https://www.ibm.com/topics/networking>.
- IREC. Smart inverters. URL <https://www.irecusa.org/our-work/smart-inverters/>.
- Chen J. Design and implementation of cloud-based single-channel lora iiot gateway using raspberry pi. *ACM SIGBED Review*, 15(2):31–36, 2018a.
- Chen J. Devify: decentralized internet of things software framework for a peer-to-peer and interoperable iot device. *Association for Computing Machinery*, 15(2):31–36, 2018b.
- Corfee-Morlot J., Westphal M., and Spiegel R. 4 ways to shift from fossil fuels to clean energy, 2019. URL <https://www.wri.org/insights/4-ways-shift-fossil-fuels-clean-energy>.
- Lacarte Carazo J. Iot asistencial. 2020.

- Tawalbeh L., Muheidat F., Tawalbeh M., and Quwaide M. Iot privacy and security: Challenges and solutions. *MDPI*, 10(12), 2020.
- D.I. Management Services Pty Limited. Cryptosys pki pro.
- Abad M. Estudio de la seguridad en los dispositivos iot (internet of things, 2021.
- Baig M., Iqbal M., Jamil M., and Khan J. Iot and blockchain based peer to peer energy trading pilot platform. In *2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 0402–0406, 2020. doi: 10.1109/IEMCON51383.2020.9284869.
- Roldán Martínez M. Diseño de sistema monitorización de calidad del aire mediante tecnologías y sensores de iot y domÓtica. *Universidad Politécnica de Cartagena*, 2022. URL <https://repositorio.upct.es/bitstream/handle/10317/11636/tfg-rol-dis.pdf?sequence=1&isAllowed=y>.
- Suárez-Albela M., Fraga-Lamas P., and Fernández-Caramés T. A practical evaluation on rsa and ecc-based cipher suites for iot high-security energy-efficient fog and mist computing devices. *MDPI*, 18(11), 2018.
- Markku-Juhani O. and Saarinen. The bluejay ultra-lightweight hybrid cryptosystem. *Cryptology ePrint Archive*, Paper 2012/195, 2012. URL <https://eprint.iacr.org/2012/195>. <https://eprint.iacr.org/2012/195>.
- Moreno Sánchez O. Diseño y despliegue de una arquitectura iot para el análisis de datos en tiempo real. *Universidad Politécnica de Madrid*, 2017. URL https://oa.upm.es/48108/9/TFM_OSCAR_MORENO_SANCHEZ.pdf.
- ONU. Objetivo 9: Construir infraestructuras resilientes, promover la industrialización sostenible y fomentar la innovación, 2023. URL <https://www.un.org/sustainabledevelopment/es/infrastructure/>.
- Vicioso P. Curvas elípticas y aplicación en la criptografía. 2021.
- Python. Tls/ssl wrapper for socket objects. 2023. URL <https://docs.python.org/3/library/ssl.html>.
- Manuel Pégourié-Gonnard. p256-m. 2023. URL <https://github.com/mpg/p256-m/blob/master/p256.py#L113>.
- ISSN RFC6979. Deterministic usage of the digital signature algorithm (dsa) and elliptic curve digital signature algorithm (ecdsa). August 2013. URL <https://www.rfc-editor.org/rfc/rfc6979#appendix-A.2.5>.
- Gavilán S. and Adrián. Seguridad en la internet de las cosas: propuesta de implantación segura de un sistema de seguridad con dispositivos iot en una pyme. 2019.
- Holek S. Simple pki. URL <https://pki-tutorial.readthedocs.io/en/latest/simple/index.html>.
- Layana Molina S. Estudio comparativo de la eficiencia de los algoritmos criptográficos aes y rsa: Caso de estudio de una institución en la ciudad de guayaquil. Master’s thesis, 2017.
- Seabrooke. Cuáles son las ventajas y desventajas de criptografía de curva elíptica para la seguridad inalámbrica? URL <https://www.seabrookewindows.com/eQLJOn3W8/>.
- TheRoboticsBack-End. Install raspberry pi os desktop on a virtual machine (virtualbox). 2023. URL https://roboticsbackend.com/install-raspbian-desktop-on-a-virtual-machine-virtualbox/#Install_VirtualBox.
- Unknown. Why nacl?, a. URL <https://confidential.pandastrike.com/guides/why-nacl/>.

Unknown. Tweetnacl, b. URL <https://tweetnacl.js.org/#/>.

Robado Gallardo V. Explotación de los recursos de una casa inteligente en plataformas en la nube mediante tecnología iot. *Escuela Técnica Superior de Ingeniería y Sistemas de telecomunicación*, 2020. URL https://oa.upm.es/67280/1/TFG_VICENTE_ROBADO_GALLARDO.pdf.

Tintín-Perdomo V., Caiza-Caizabuan J., and Caicedo-Altamirano F. Arquitectura de redes de información. principios y conceptos. *Revista Científica, dominio de las ciencias*, 4(2):103–122, 2018.

Montoya Mendoza Y., E. Ramírez, Pérez Di Santis T., Molina L., and García N. Estado del arte de smart grid: Parte i (state of art of smart grid: Part i). *energética*, 1:9, 2016.