Instituto Tecnológico y de Estudios Superiores de Monterrey

Escuela de Ingeniería y Ciencias

Uso de álgebras modernas para seguridad y criptografía

Ingeniería en Ciencias de Datos y Matemáticas

Reporte del proyecto

Profesores
Dr. Alberto Francisco Martínez Herrera, Dr. Salvador Mancilla Hernández

Socio Formador
LiCore

Equipo 1

José Andrés Meyer Crabtree	A01366785
Diego Paasche Portillo	A01028103
Luis Leopoldo Jiménez Pérez	A01275004
Federico Medina García Corral	A01721441
Eugenio Andrade Lozano	A01721296
César Guillermo Vázquez Álvarez	A01197857

Monterrey, Nuevo León. 7 de febrero de 2023

Índice general

Resumen

El propósito del proyecto es hacer un programa que pueda enviar paquetes, firmarlos y validarlos con ayuda de los auditores y el Centro de Control, trabajando bajo un criterio de confidencialidad, autenticidad e integridad. Para esto utilizamos el algoritmo de Curvas Elípticas para firma digital, conocido como ECDSA. Este fue implementado en Python a su vez de estar en Django REST Framework. Finalmente se logró transferir mediante una red local datos de las Trazas proporcionadas vía paquetes, verificarlas y hacer su respectiva validación. Al finalizar el documento, se muestra una lista donde se proporcionan puntos específicos para la mejora del proyecto para futuros trabajos.

1. Introducción

La problemática se aborda a partir desde las necesidades en las que se encuentran referido y comentado por el Socio Formador, el cual es un proyecto criptográfico de un sistema de medición de energías renovables el cual busca la confidencialidad, autenticidad e integridad de los datos adquiridos por un dispositivo llamado auditor (realizado por el Socio Formador).

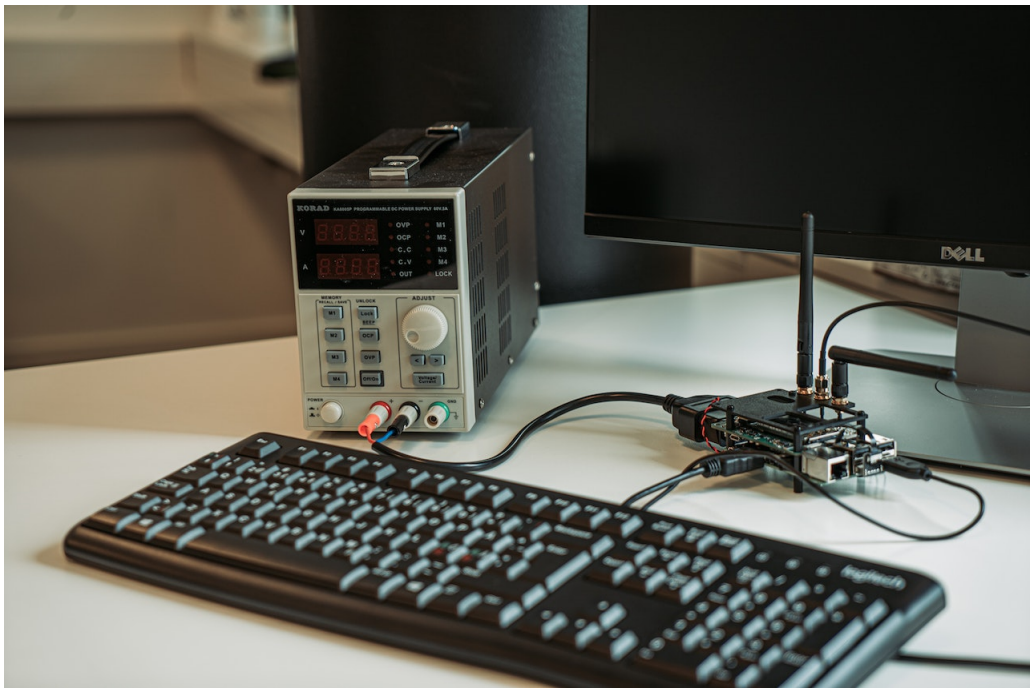

Figura 1: Tarjeta Raspberry Pi en entorno de desarrollo [Hajtas, 2021]

También, uno de los propósitos de trabajar con la organización socio formadora es crear soluciones que ayuden a cumplir con las metas de desarrollo sostenible (ODS). Las metas de Desarrollo Sostenible son un conjunto de 17 objetivos establecidos por la ONU para alcanzar un desarrollo sostenible en áreas como la economía, la sociedad y el medio ambiente [ONU, 2020]. Este proyecto de criptografía para un sistema de medición de energías renovables se relaciona con varias de las ODS, específicamente:

- **ODS 7**: Acceso a una energía asequible y no contaminante. El sistema de medición de energías renovables ayuda a medir y monitorear la producción de energía renovable, lo que es esencial para garantizar que se estén utilizando fuentes de energía limpias y sostenibles.
- **ODS 9**: Industria, innovación e infraestructura. El proyecto promueve la innovación en la medición y el monitoreo de energía renovable y contribuye a la construcción de infraestructura sostenible.
- **ODS 12**: Consumo y producción responsable. El sistema de medición de energías renovables puede ayudar a las empresas y las personas a monitorear y reducir su consumo de energía y aumentar la eficiencia energética.
- **ODS 16**: Paz, justicia e instituciones sólidas. La privacidad y seguridad de los datos es un tema importante en la construcción de instituciones sólidas y justas, por lo que el proyecto contribuye a esta meta al garantizar la privacidad y seguridad de los datos del sistema de medición de energías renovables.

De la misma forma la solución ayuda a mejorar la infraestructura energética de México y de países con condiciones parecidas; mediante el monitoreo de la producción de energías renovables, la toma de decisiones informadas, el fomento de la inversión en energías renovables, el aumento de la transparencia y la mejora de la seguridad energética.

1. **Monitoreo de la producción y recolección de energía renovable**: El sistema de medición puede ayudar a medir y monitorear la producción de energías renovables como la energía solar y eólica, lo que es esencial para garantizar que se estén utilizando fuentes de energía limpias y sostenibles.
2. **Toma de decisiones informadas**: Con los datos precisos y seguros de producción de energía renovable, el gobierno y las empresas pueden tomar decisiones informadas sobre cómo invertir en infraestructura energética y cómo mejorar la eficiencia energética. Fomentar la inversión en energía renovable: Un sistema de medición fiable y seguro puede atraer a más inversores a invertir en energías renovables en México, lo que ayudaría a aumentar la capacidad de producción de energía renovable y mejorar la infraestructura energética.
3. **Aumento de la transparencia**: El sistema ayuda a aumentar la transparencia en la medición de la producción y consumo de energía, lo que ayuda a combatir la corrupción y promover un uso más eficiente de la energía.
4. **Mejora de la seguridad energética**: Con un sistema seguro y confiable de medición de energía renovable, se pueden detectar y prevenir cualquier tipo de intrusión en la infraestructura energética.

Nuestro Socio Formador requiere la implementación de criptografía de clave publica para la protección de comunicaciones con IoT, como bien sabemos, este intercambio de información sensible se hará a través de internet, lo cual no es un método seguro.

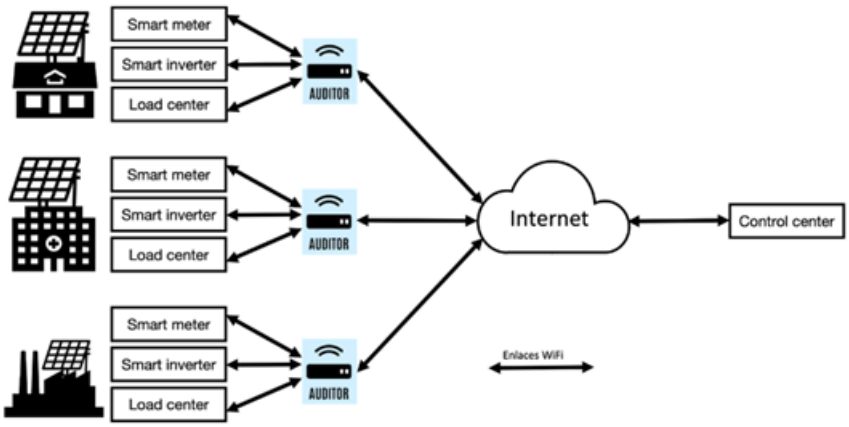

Figura 2: Esquema de intercambio de datos a ser protegido [Razo Zapata, 2023]

De acuerdo a la Figura 2, se observa el Centro de Control (control center), los auditores y el Internet, (el cual es el medio donde se hará el intercambio de información gracias a los enlaces Wi-Fi). Como se puede observar en la imagen en el primer auditor se tiene una casa, en el segundo auditor un hospital y en el tercer auditor una fábrica. En este caso, la información que es intercambiada en esta imagen, es siempre en el mismo formato. Por otro lado, al momento de hacer el intercambio de información se necesitan 3 características que son fundamentales, la confidencialidad (que únicamente los involucrados puedan acceder a la información proporcionada), la integridad (que la información no sea alterada ni modificada por ningún motivo) y por ultimo la autenticidad (que se pueda ser monitoreado e identificado el autor de dicho mensaje). También se puede observar diferentes componentes los cuales son bastante importantes. En estos están los dispositivos SMART, los cuales están encargados de monitorear los parámetros de energía cada cierto tiempo. De igual forma, se pueden encontrar los auditores digitales, el cual tienen como función recolectar datos de los dispositivos SMART y, por ultimo, el Centro de Control [Razo Zapata, 2023].

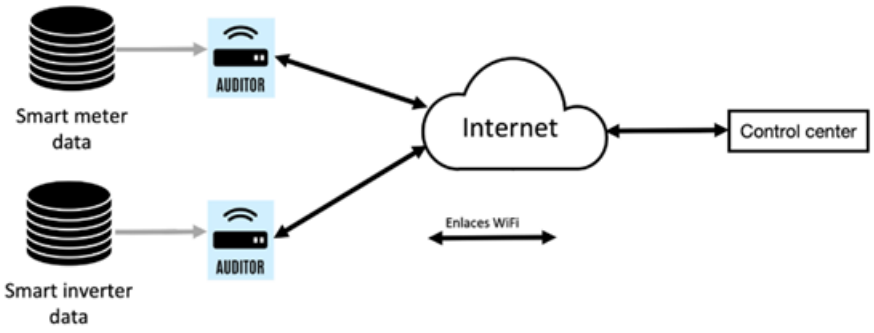

Figura 3: Esquema sugerido para la prueba de concepto (PdC) [Razo Zapata, 2023]

La Figura 3 es un esquema de lo que se busca realizar en el transcurso del reto, el cual este será realizado a manera de Prueba de Concepto. Como se puede observar en la imagen y para la realización del reto, se estarán utilizando 2 auditores, 1 Centro de Control y las bases de datos que el esquema tiene. Estas bases de datos tienen los siguientes nombres: Smart Meter Data y Smart Inverter Data. Este esquema de la Figura 3 es un esquema simplificado para el proyecto realizado. Se estarán tomando en cuenta a la hora de realizar el reto la confidencialidad, integridad y la autenticidad [Razo Zapata, 2023]. Los datos que se estarán usando ya están dentro de las bases de datos y serán transferidas al auditor y de esta forma se hará ya el intercambio de información, haciendo emulación de la transmisión.

En conjunto, este proyecto ayuda a alcanzar un desarrollo sostenible al medir y monitorear la producción de energías renovables, al promover la innovación y la eficiencia energética, y al garantizar la confidencialidad, autenticidad e integridad de los datos.

2. Estado del arte

El estado del arte del trabajo de criptografía para un sistema de medición de energías renovables se ha desarrollado en los últimos años debido a la creciente importancia de la medición y el monitoreo de la producción de energías renovables.

Existen diversos sistemas de medición de energía renovable en el mercado, como los medidores inteligentes y los sistemas de monitoreo en tiempo real. Sin embargo, estos sistemas a menudo carecen de medidas de seguridad y privacidad adecuadas, lo que puede exponer los datos a posibles interceptaciones y alteraciones no autorizadas.

Para abordar estos desafíos, se ha realizado un estado del arte de trabajos anteriores de la misma rama a este trabajo. Sin embargo, todavía existen desafíos en términos de eficiencia computacional y escalabilidad en la implementación.

2.1. Algoritmos criptográficos existentes

2.1.1. RSA

“RSA” Es un algoritmo de cifrado el cual es de los más utilizados hoy en día debido a su simplicidad y eficacia. Fue creado en 1997 por Ron Rivest, Adi Shamir y Leonard Adleman, de ahí el nombre RSA, que son las iniciales de los apellidos de estos.

El algoritmo funciona de forma asimétrica, es decir, trabaja con dos claves, una pública y una privada. El concepto es bastante sencillo, todo lo que se cifra con la llave pública lo podemos descifrar con la clave privada y viceversa.

Utilizar el método RSA es bastante sencillo, como primer paso tenemos que escoger 2 números primos a los cuales se denominarán “p” y “q”. Entre mayor sea el valor de p y q, mayor la efectividad del cifrado.

El método RSA resulta ser una opción muy viable porque funciona de forma correcta y no tiene una complejidad muy grande. El problema del algoritmo cae en que funciona debido a que en la aplicación los números p y q que se elijan son muy grandes y ocupa bastante potencia computacional. [Córdoba, 2022]

2.1.2. Curvas Elípticas

ECC, “Elliptic Curve Cryptography” usa propiedades matemáticas que se obtienen de las curvas elípticas el cual estas sirven para producir sistemas criptográficos de clave pública. ECC se basa en las funciones matemáticas que son fáciles de calcular hacia una dirección pero se vuelven sumamente difíciles al momento de revertirlo. Para ECC, esta dificultad es gracias al calcular el logaritmo discreto de un elemento de curva elíptica aleatoria con respecto a un punto base conocido públicamente, o el problema de logaritmo discreto de curva elíptica ECDLP, este, es un algoritmo de firma que se usa de manera frecuente dentro de la criptografía de clave pública que usa ECC. [SSL.com, 2021]

2.1.3. DSA

El Algoritmo de Firma Digital (“DSA” por sus siglas en inglés) incorpora las propiedades algebraicas de los logaritmos y de exponenciación modular para generar firmas electrónicas utilizables en distintas aplicaciones. El algoritmo fue nombrado un FIPS en 1994 por el Instituto Nacional de Estándares y Tecnología (NIST, por sus siglas en ingles) [Niazi, 2022].

Los problemas de logaritmo discretos como los de exponenciación modular son muy difíciles de calcular utilizando métodos de fuerza bruta.

El funcionamiento del DSA se basa en la utilización de 2 funciones, una para firmas y otra para verificación. En este algoritmo se utilizan las llaves tanto públicas como privadas para encriptar/desencriptar respectivamente. [Jena, 2022]

2.1.4. Comparación de Algoritmos

En la Tabla 1, se muestran de manera sintetizada, algunos aspectos positivos, negativos, así como el tamaño de clave de los algoritmos de clave pública RSA, DSA, y Curvas Elípticas. En esta Tabla, se encuentran en manera de resumen cada una de las ventajas y desventajas de dichos algoritmos de clave pública. Observando el contenido de la Tabla 1, se puede observar que RSA y DSA son fáciles de implementar, pero gastan mayor ancho de banda dado que el tamaño de clave es mayor que el de las curvas elípticas, pero, esta última es más complicada de usar. La información de la Tabla 1 se basó en la información obtenida de las siguientes fuentes: [Córdoba, 2022], [SSL.com, 2021], [Niazi, 2022], [Jena, 2022].

	RSA	Curvas Elipticas	DSA
Aspectos Positivos	Sistema rápido computacionalmente por lo que la distribución de las claves es fácil y segura. El algoritmo usa factorización de números grandes, por lo que es bastante seguro.	Es de mayor utilidad en dispositivos móviles, donde la potencia de cálculo , la memoria y la duración de la batería es limitado. El logaritmo discreto eliptico nos ayuda a que las curvas elipticas tengan una seguridad alta, debido a que son dificiles de solucionar.	La seguridad es robusta, haciendo que la velocidad de generación de llaves y desencriptación sea alta, requiriendo menos memoria para completar ciclos. Gracias al problema del logaritmo discreto, DSA también resulta ser un método super seguro.
Aspectos Negativos	Utiliza un mayor espacio que el mensaje original.	El proceso para implementarlo es muy complicado, lo cual pudiera crear errores en la programación.	Este no incluye la capacidad de cambio de llaves. Igualmente no soporta certificados. La llave no puede ser actualizada o alterada una vez creada. (en realidad todos los puntos son cubiertos al momento de usar certificados)
Tamaño de claves	Generalmente el tamaño de las claves suele ser entre 1024 bits y 4096 bits, usualmente mayor a 1024 bits.	Una clave de cifrado ECC 160 bits proporciona la misma seguridad que una clave de cifrado RSA de 1024 bits siendo hasta 15 veces más rápido, según el equipo de SSL.	Para cada aplicación, el tamaño de la llave varía. Además, DSA cuenta con un límite en la longitud de dichas llaves. Se debe especificar el tamaño de este. Generalmente es mayor a 1024 bits.

Tabla 1: Tabla comparativa de Algoritmos de Clave Pública

2.1.5. Tamaño de clave en Bits

Simétrico	ECDSA	RSA	DSA
80	160	1024	1024
112	224	2048	2048
128	256	3072	3072
192	384	7680	7680
256	512	15360	15360

Tabla 2: Tamaño de clave en Bits [SSL.com, 2021]

En la Tabla 2 se muestran las diferentes equivalencias de clave privada respecto a clave publica relacionado con los algoritmos RSA, DSA, y ECDSA, donde este último es la versión curvas elípticas de DSA. Se puede ver que, por ejemplo, la clave de 80 bits en un algoritmo de cifrado simétrico es equivalente a tener tanto en RSA como en DSA un tamaño de clave de 1024 bits, mientras que para curvas elípticas tenemos una equivalencia de 160 bits. En términos generales, la primer columna muestra los tamaños de clave de algoritmos simétricos, la segunda muestra los tamaños de clave de ECDSA, la tercera para RSA y la cuarta para DSA.

Los tamaños de clave más pequeños de ECDSA significan que se puede lograr un cifrado más fuerte con

menos potencia informática y ancho de banda de red que RSA; esto es especialmente ventajoso para dispositivos móviles de baja potencia y de Internet de las cosas, que se están volviendo cada vez más ubicuos. [SSL.com, 2021]

2.2. Certificate-Based Authentication

Una parte importante para la comunicación entre los auditores y el centro de control es que se pueda conocer y validar que la información recibida proviene de un cierto auditor. Se va a proponer alguna manera de poder hacer el arranque de los auditores con su certificado. Esto porque en algún momento podría darse el caso de que existan más auditores o se tengan que reiniciar y se tendría que dar su certificación de identidad. Una propuesta sería que desde el centro de control se registren los auditores con su certificación para que la información sea más segura. [Martínez Herrera, 2023]

2.3. Bibliotecas Disponibles

Biblioteca	Lenguaje	Licencia	Condiciones de uso	Clave Pública
pyOpenSSL	Python 3	Apache versión 2.0	Copyrighted	mín 160 bits
PyNaCl	Python 3	Apache versión 2.0	Copyrighted	mín 1024 bits
iTextSharp	C++	Licencia AGPL	Copyleft	mín 1024 bits
OpenSSL	C++	Apache versión 2.0	Copyrighted	mín 1024 bits
iText	Java	Licencia AGPL	Copyleft	mín 1024 bits
PDFNetPython3	Python 3	Licencia PDFTron	Copyrighted	40-256 bits
Crypto++	C++	Software boost 1.0	Copyrighted	mín 64 bits
wolfCrypt	C++	wolfMQTT	Open Source	mín 1024 bits
PyCryptodome	Python 3	BSD 2-Clause	Open Source	mín 160 bits

Tabla 3: Tabla descriptiva de bibliotecas disponibles

La información de la Tabla 3 fue obtenida de las páginas web oficiales de las empresas creadoras de las bibliotecas, son las siguientes fuentes:

- **OpenSSL**: [Kalin, 2019]
- **Crypto++, wolfCrypt**: [Dai, 2014]
- **pyOpenSSL, PyNaCl**: [ASF, 2004]
- **iTextSharp**: [Cantero, 2020]
- **PDFNetPython, PyCryptodome**:[KG, 2023]

La primera columna de la Tabla 3 muestra el nombre de las bibliotecas existentes que trabajan con claves públicas. La siguiente columna muestra el lenguaje de programación que se utiliza, mayormente se encuentran Python y C++, solo una biblioteca usa Java como lenguaje de programación. La tercera columna nombra las licencias que cada librería tiene, las dos más utilizadas son la de Apache versión 2.0 (creada por Apache Software Foundation) y la Licencia AGPL (Affero General Public License). La siguiente columna menciona sus condiciones de uso, ya sea Copyright (Derechos de Autor), que significa que se tiene que mencionar siempre al autor para poder utilizarla, "opensource", que significa que cualquiera puede utilizar y/o modificar la librería y "Copyleft", que a diferencia del "Copyright", se promueve la distribución de la información, entonces cualquiera puede modificar la información pero siguiendo las condiciones de la primera versión de la licencia. Se puede observar que la licencia de Apache versión 2.0 es copyrighted, mientras que la licencia AGPL es "copyleft". Finalmente, la última columna muestra el tamaño (en bits) que cada librería puede crear. Para el caso de pyOpenSSL y PyCryptodome, las librerías pueden crear llaves públicas mediante Curvas Elípticas, RSA o DSA. Para PyNaCl se puede crear claves con RSA o DSA al

igual que para iTextSharp, iText, OpenSSL. En el caso de Crypto ++ puede crear llaves con RSA, DSA, GDSA, ESIGN, and Rabin-Williams. Para PDFNetPython3, se utiliza desde RC4 con tamaño máximo de 40 bits hasta AES con tamaño máximo de 256 bits. Finalmente WolfCrypt solo trabaja con DSA.

2.4. Recursos Computacionales Disponibles

La computadora utilizada para la elaboración del proyecto es una Windows Surface Laptop 4, la cual contiene un procesador Intel i7 de 11va generación en donde el tipo de sistema es de 64 bits con su respectivo procesador x64 al igual que este cuenta con 16GB de RAM. Junto con todo lo anterior, se trabajó con el sistema de Windows 11. En la Tabla 4 se muestran las especificaciones del software del computador utilizado para la elaboración del proyecto.

Edition	Windows 11 Home
Version	21H2
Installed on	01/21/2021
OS Build	22000.1335
Experience	Windows Feature Experience Pack 1000.22000.1335.0

Tabla 4: Recurso Computacional disponible para la elaboración del proyecto

2.5. Recursos de Hardware Disponibles

Dispositivos	Procesamiento	Memoria interna	RAM	Lenguaje
Arduino Uno	ATmega328P 16 MHz	32KB	2KB	Java, MicroPython, C/C++
Esp8266	Tensilica Xtensa LX106 80 MHz (160 MHz overck.)	EEPROM de 4MB	64KB	Arduino, Lua, MicroPython, C/C++, Scratch
Raspberry Pi 3	Quad Core 1.2GHz Broadcom BCM2837 64bit CPU	No tiene memoria interna pero tiene puerto sd para integrar.	1GB	Python, C/C++
Raspberry Pi 4	System on Chip (SoC) Broadcom BCM2711 compuesto por un procesador ARMv8 de cuatro núcleos de 64bit de 1.5GHz.	No tiene memoria interna pero tiene puerto sd para integrar.	1GB, 2GB, 4GB or 8GB LPDDR4-3200 SDRAM (dependiendo del modelo)	Python, C/C++
Nvidia jetson nano (Developer Kit)	Quad-core ARM® A57	No tiene memoria interna pero tiene puerto sd para integrar.	4GB 64-bit LPDDR4	Python

Tabla 5: Tabla Comparativa de Hardware

La información de la Tabla 5 fue obtenida de las siguientes fuentes.

- Arduino Uno:[Guerrero, 2014]
- Esp8266: [Hernandez, 2022]
- Raspberry Pi 3, Raspberry Pi 4: [Huertos, 2019], [Hajtas, 2021]

- Nvidia jetson nano: [Nvidia, s.f.]

Como podemos observar en la tabla, podemos ver los dispositivos el cual tenemos Arduino uno, ESP8266, Raspberry Pi3 y Pi4 y Nvidia. Podemos ver las características tales como el procesamiento, memoria interna, RAM y el lenguaje, podemos ver características muy favorables para cada dispositivo, asi mismo los lenguajes, como podemos observar son muy amigables, en su mayoría, Python es el lenguaje que todos pueden utilizar y C++ esta por detras.

Dispositivos	**Pros**	**Cons**
Arduino Uno	Es un hardware barato y fácil de usar. Igualmente cuenta con soporte multiplataforma para Windows, MacOS, y Linux	No tiene soporte de comunicaciones incorporado, lo cual hace que sean limitadas en términos de compatibilidad tanto con Bluetooth y WiFi, así como limitaciones con memoria interna ya que solo cuenta con poca memoria EEPROM
Esp8266	Para muchos proyectos IoT y Wifi, EL ESP9266 puede hacer el trabajo por un precio mas bajo; Ambas placas se pueden programar usando Arduino IDE u otros IDE compatibles. Ambas placas adminten firmware MicroPython	No bluetooth, no hardware security la forma de programar no es sencilla
Raspberry Pi 3	Trae el módulo integrado de Wi-Fi. Tiene muchos puertos como USB, HDMI, SSD. Está soportado por Linux.	No tiene memoria interna, el costo es un poco elevado. No corre en Windows solo en Windows ARM.
Raspberry Pi 4	Se puede elegir entre 1, 2, 4 y 8GB de memoria RAM. Trae el módulo integrado de Wi-Fi.Tiene un procesador más avanzado.	Consume más energía. Los sistemas de Raspberry Pi no son compatibles con Pi 4. Han cambiado algunos puertos. No corre en Windows solo en Windows ARM.
Nvidia jetson nano (Developer Kit)	El sistema operativo es linux o también soporta windows arm. El procesador es muy avanzado incluso este tipo de dispositivos son utilizados para la implementación de ML.	No tiene memoria integrada, el costo energético es alto, de igual forma se llega a calentar por lo mismo del procesamiento.

Tabla 6: Tabla de Pros y Cons de Hardware

Para la elaboración de este proyecto a partir de la investigación hecha, evaluando los pros y contras, y dependiendo de que se nos provee del material necesario para laborar, se decidió trabajar con tarjetas Raspberry Pi 3. Esto se debió a que las Raspberry Pi 3 ofrecen una gran flexibilidad en cuanto a su configuración y posibilidades de programación, además de tener un bajo costo en comparación con otras opciones similares. Además, estas tarjetas cuentan con una amplia comunidad de desarrolladores y una gran cantidad de recursos disponibles en línea, lo que facilita la resolución de problemas y la implementación de nuevas funcionalidades. Sin embargo, también se tuvo en cuenta que las limitaciones que tiene la Raspberry Pi 3 para entornos fuera de pruebas, por lo que se debió realizar un análisis detallado de las necesidades del proyecto y comparar con otras opciones antes de tomar una decisión final, la cual si utilizar dos Raspberry Pi 3.

2.6. Ejemplos de Aplicaciones de IoT con modelos criptográficos

Un ejemplo de aplicación de IoT con modelo criptográfico es el de Arduino; este trabaja con un hardware y software fácil de utilizar y bastante flexible, por lo que se pueden hacer múltiples cosas con este. Estas características hacen que sea posible para una persona pueda crear sus propias placas con diferentes

funciones dependiendo de lo que se busca obtener. Investigando un poco sobre sus usos, se encontró que es popularmente utilizado para crear diferentes aparatos electrónicos domésticos hechos en casa, es decir, de manera casera. Por ejemplo, hay mucha gente que utiliza dichas placas para crear relojes con alarma, también hay casos de gente usando placas para crear básculas, máquinas simples como máquinas de chicles, máquinas de acceso a casas usando huellas digitales, sistema de voz, máquinas expendedoras, etc. En fin, Arduino permite que se hagan muchos proyectos distintos de manera fácil y flexible, permitiendo a la gente que haga prácticamente lo que quieran con esto, teniendo como limitante su propia creatividad.

Kittur et al. habla sobre el uso de RSA y de la dificultad y el costo computacional que puede haber cuando se agrega un sistema de verificación y autenticación de firmas individuales. Los autores proponen un sistema de verificación rápida mediante Batch Verification de las firmas digitales. Este modelo utiliza RSA junto con verificación muestral de las firmas, lo que permite reducir el tiempo de verificación significativamente. Los autores también mencionan que esta técnica sigue en investigación pero ha mostrado resultados muy prometedores [Kittur et al., 2017].

Otro ejemplo encontrado es el que Mughal et al. diseñaron la aplicación de un sistema ligero basado en DSA para productos basados en IoT centrados en el humano. Los productos IoT se basan en el intercambio de información, lo que hace indispensable el hecho de que esta información esté segura, especialmente tratándose de productos centrados en el humano. La publicación menciona el ejemplo en el que dispositivos inteligentes interactúan con el hombre para medir los parámetros de salud indispensables y enviar la información al repositorio central. Esta detección e intercambio de información podría ser víctima de ataques y podría ser secuestrada y/o alterada con propósitos distintos a los establecidos. Es por esto que se requiere de un sistema que proteja a los dispositivos de este tipo de ataques [Mughal et al., 2018].

Otro artículo encontrado es el elaborado por Z. Liu et al. donde estos comparten su investigación en la cual se define una familia emergente de curvas elípticas livianas para que se puedan cumplir los requisitos de algunos dispositivos con recursos limitados, presentando una ECC escalable, regular y altamente optimizada para los nodos MICAz y Tmote Sky. Su implementación parametrizada de la aritmética de grupos de curvas elípticas admite campos primos pseudo-Mersenne en diferentes niveles de seguridad con dos diseños optimizados, la versión de alta velocidad y la versión de memoria eficiente [Liu et al., 2016].

Por otro lado, dentro del artículo publicado por Veeramanickam y Mohanapriya, se hace mención sobre cómo las nuevas tecnologías han ido cambiando durante los últimos años y cómo se han ido implementando dichas nuevas tecnologías para lo que es el E-Learning. Dentro de este, se habla sobre cómo es la adaptación del E-Learning para el sector académico y así poder, en algún futuro cercano, llevar a los estudiantes a un ambiente "i-Campus". En el artículo se hace mención de cómo se estarán tomando medidas de seguridad para que cada vez esto sea más eficiente y con mejores resultados utilizando el IoT [Veeramanickam y Mohanapriya, 2016].

Otro claro ejemplo del uso de IoT en diferentes sectores, es en el que S. Naik habla en su artículo sobre cómo se utiliza el Raspberry Pi para poder monitorear a los pacientes, ayudando a tener un mejor y más rápido diagnóstico del paciente, todo esto dentro del área médica. Lo que hace el Raspberry Pi es recibir información de los sensores que utiliza el paciente y lo guarda dentro del servidor, donde el doctor puede ver la información actual para poder revisar que todo esté bien y, en caso contrario, poder ayudar en el momento más preciso. Igualmente, el artículo hace mención sobre el uso específicamente de Raspberry Pi gracias a que este tiene un costo muy bajo de energía. Finalmente, este hace la observación de que se utilizó Python como lenguaje de programación al igual que fue implementado en una red 4G, dando a entender que su implementación no requiere de muchas herramientas [S Naik, 2019].

Otro ejemplo del uso de Raspberry Pi en el sector médico se muestra en el artículo publicado por Vasanth K et al. habla sobre cómo se puede utilizar un Raspberry Pi para poder ayudar a la gente ciega o sorda. Lo que el artículo menciona es que por medio de peticiones a Google Speech API, el Raspberry Pi puede traducir lo que se habla a texto para la gente sorda o también para amplificar el audio para la gente que tiene problemas para escuchar o es ciega. Para la comunicación con el API de Google se utilizó el protocolo de TCP/IP para la comunicación de los datos y HTTP para comunicarse con el servidor. Finalmente, para el audio se utilizó la encriptación con el formato NDR. Con este claro ejemplo, se puede observar un gran uso de manera segura del Raspberry Pi para poder ayudar a gente con problemas de vista o de escucha y mostrando el gran impacto que tiene este en el sector médico [Vasanth K, 2019].

Por otro lado, gracias al soporte de IoT, el sector médico se ha visto altamente beneficiado por su uso. En otro artículo se hace mención que gracias a dicho soporte, la monitorización de corto plazo y notificaciones de señales de emergencia dentro del área de salud se han vuelto más accesibles. Sin embargo, IoT no previene diferentes obstáculos que puedan impedir el uso correcto de diferentes prácticas médicas. Como los datos vitales y de diferentes aspectos médicos son confidenciales, es necesario tener algún tipo de seguridad criptográfica para cuidar dichas cuestiones. Algunas cosas como las limitaciones de memoria, los procesos computacionales y el tamaño de los dispositivos contradicen lo robusto que es el proceso de cifrado que requiere la ayuda de la criptografía de bajo peso, para mantenerla manejable de manera práctica. Dentro del artículo, se habla de algunos algoritmos de encriptación como AES, SPECK y SIMON para ver cuál de estos es el mejor para aplicaciones médicas. Esto se midió en base al tiempo de ejecución, el consumo de energía, la capacidad de la memoria y la velocidad en la que trabajan. Todo esto se trabajó usando un simulador Cooja corriendo en el sistema operativo Contiki. [Alassaf y Gutub, 2019]

Uniendo los puntos mostrados anteriormente, es importante hacer mención que el uso de IoT ha tenido un alto impacto en el sector de salud y gracias a su constante evolución, se ha creado el nuevo término "Internet of Medical Things" (IoMT). Haciendo mención al artículo publicado por N. Alassaf y A. Gutub [Alassaf y Gutub, 2019], un equipo de investigadores en 2018 investigó sobre cómo es que el IoT ha ido sobrepasando las capacidades humanas de brindar servicio dentro de distintas áreas, respaldando lo que es un causante del nacimiento del IoMT. En este artículo, se explica sobre cómo es que se puede encriptar imágenes médicas confidenciales usando métodos distintos de encriptación para así lograr crear estrategias óptimas para conservar su privacidad. En resumen, se menciona sobre la seguridad que un hospital utiliza para conservar los reportes médicos de un paciente y sobre la importancia de este. Finalmente, después de distintas pruebas, se encuentra que la manera óptima del proceso de encriptación y desencriptación es usando algo llamado "Hybrid Swarm Optimization", esto en curvas elípticas [Elhoseny et al., 2018].

Haciendo énfasis en la protección de registros médicos, Alzubi publicó un artículo en el que propone el uso de un sistema de encriptación asistido por block-chain. Este articulo se basa en sistemas médicos con IoT. El autor propone usar Lamport Merkle Digital Signature (LMDS) para autenticar los dispositivos y proteger los datos sensibles de los pacientes [Alzubi, 2021].

Utilizando los ejemplos anteriores, se puede encontrar que el uso de IoT con modelos criptográficos es de suma importancia para diferentes áreas y disciplinas. Volviendo a los ejemplos anteriores, se puede observar que proteger datos de pacientes dentro del sector médico tiene una alta prioridad dentro de los hospitales y las instituciones de salud ya que se está hablando de información que puede significar la vida o muerte de algún individuo. Así como este, hay múltiples disciplinas que necesitan el uso de IoT con modelos criptográficos para poder proteger su información y así asegurar el uso correcto de la información que se tiene.

3. Propuesta de solución

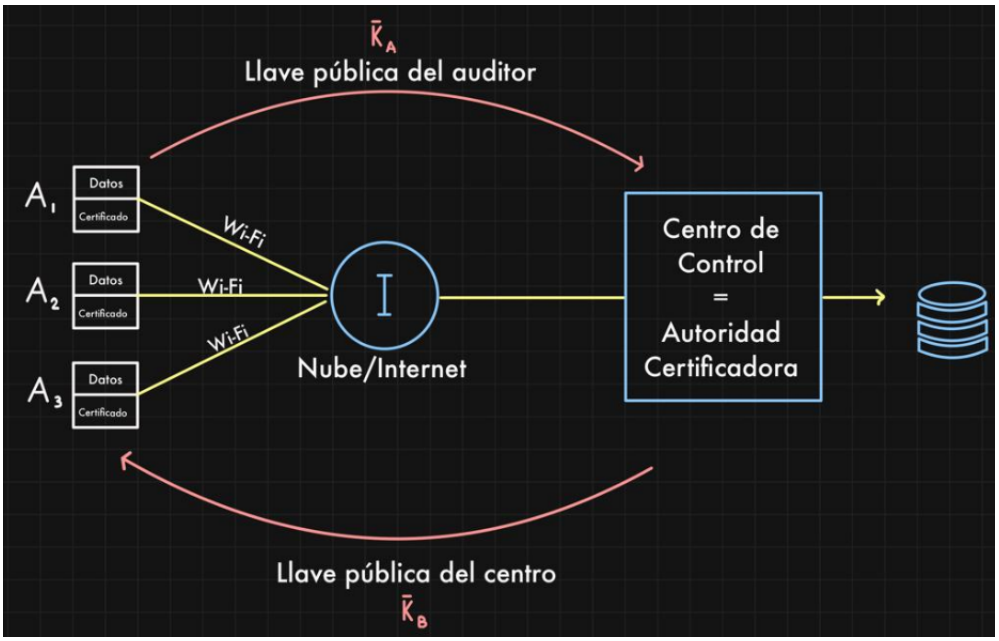

Figura 4: Propuesta Inicial de la solución

La propuesta es utilizar un hardware (ej. Raspberry Pi 3) que conecte los auditores a la nube/internet, programarlo mediante el lenguaje de programación Python y utilizando el framework de Django REST, para que esta información llegue al centro de control, teniendo en mente que haya integridad, autenticidad y confidencialidad, haciendo énfasis en esta, ya que es un pilar importante de la criptografía y que facilite la comunicación entre el centro de control y los auditores.

Django REST Framework facilita la comunicación entre la base de datos y el cliente (en este caso el auditor) mediante una API. Esto se va a utilizar para también darle una estructura a la información que se va a recibir y que se quiere verificar. La arquitectura incluirá modelos de Entradas, Auditores, Centro de Control, Llaves Públicas y Usuarios.

Una parte importante que se necesitaría es que los auditores estén certificados, entonces con la arquitectura que se va a crear se tendría un Centro de Control que se convertiría en la Autoridad Certificadora, ya que sería encargado de crear esos certificados y los auditores que están siendo certificados. También, se utilizarán Curvas Elípticas para poder cumplir con todos los puntos.

4. Metodología y desarrollo de la solución propuesta

4.1. Control de versiones/repositorio.

Para trabajar en diferentes versiones del código y poder documentar todos los códigos, para después compartirlo y laborar de manera ordenada, se utilizó el sistema de control de versiones GIT, y el flujo de trabajo que se utilizó es un sistema de control de versiones centralizado “GitHub”. GitHub es un servicio en línea que proporciona alojamiento para repositorios Git y facilita la colaboración en proyectos de software mediante funciones como el seguimiento de problemas, la revisión de código y el control de acceso. Utilizar Git y GitHub juntos permite un flujo de trabajo de desarrollo colaborativo y ordenado, y permite rastrear y documentar los cambios en el código a lo largo del tiempo. Igualmente, para poder expandir la comprensión de este, se incluyó una documentación y un README.md, los cuales explican cada una de las librerías utilizadas y las especificaciones del material, tanto hardware como software del proyecto para poder ser

imitadas por cualquiera que lo ocupe. A continuación se muestra el repositorio:

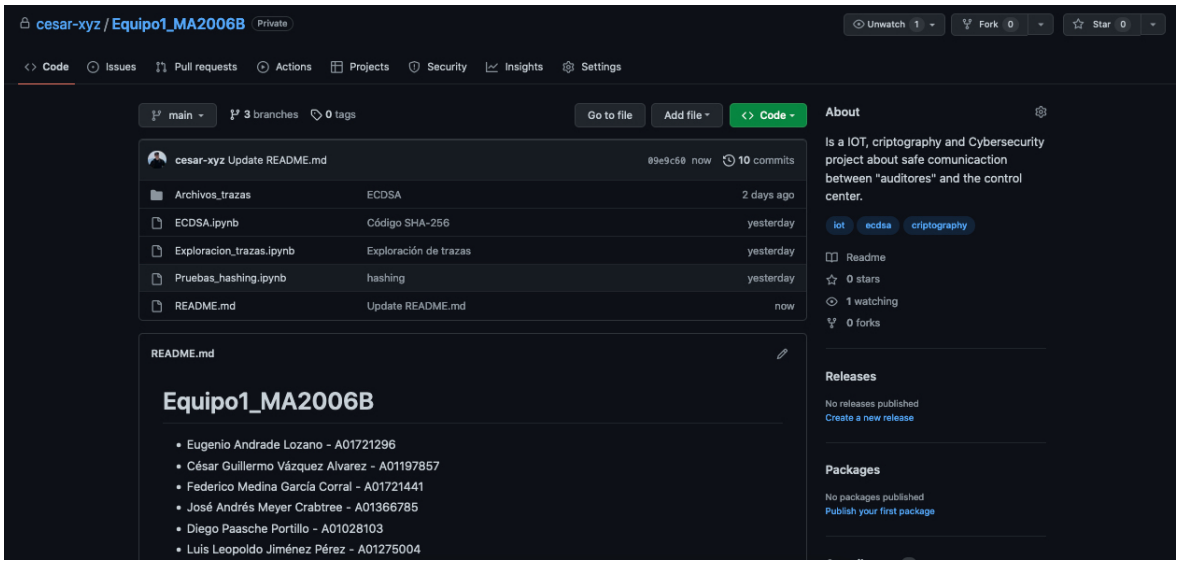

Figura 5: Repositorio del Equipo

Cabe recalcar que el repositorio con el cual se ha estado trabajando es privado y solo las personas autorizadas para realizar modificaciones pueden verlo y modificarlo, hasta el momento solo los miembros del equipo y por ende los únicos que pueden editar el repositorio. Es por eso que es importante que la comunicación y la colaboración dentro del equipo sean claras y que el proceso de revisión del código sea sólido para mantener la calidad y la coherencia del código. Debido a esto se tiene una serie de pasos para poder trabajar de manera estandarizada dentro del repositorio.

- Cada miembro del equipo debe de clonar el repositorio en su máquina local.
- Los miembros del equipo crean sus propias ramas para realizar cambios en el código base o, si se acordó, pueden trabajar en la rama principal.
- Cuando un miembro del equipo termina de hacer cambios, empuja su rama al repositorio central y crea un Pull Request.
- Otros miembros del equipo revisan el Pull Request y aportan comentarios o aprueban los cambios.
- Una vez aprobado el Pull Request, se hace merch con la rama principal.

4.2. Paquetes

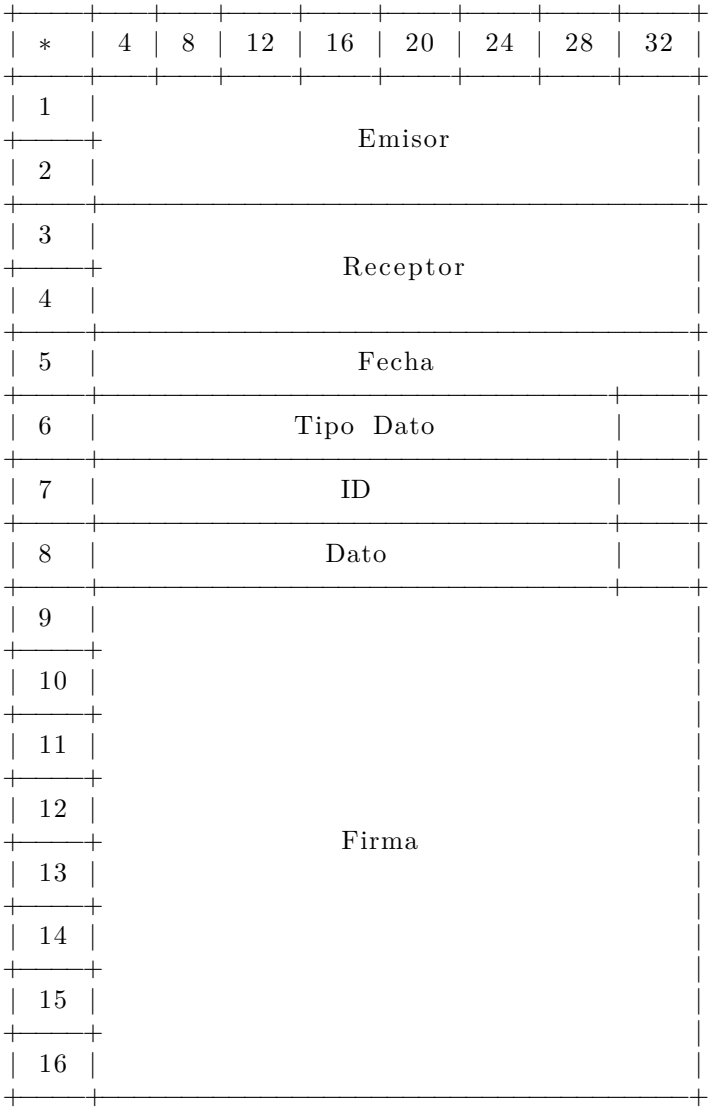

Tabla 7: Memoria que se utiliza para los paquetes

En la Tabla 7, se muestra cómo se distribuye la información dentro de los paquetes. Cada columna de este representa 4 bits, dando en total 32 bits por renglón. Los renglones representan las cantidad de veces que se repiten las columnas, por ejemplo, si una información requiere de dos renglones y todas las columnas, significa que requiere 32 bits 2 veces, dando en total 64 bits.

Un ejemplo claro de esto es el del Emisor, quien requiere 2 renglones y todas las columnas, mostrando que utiliza 64 bits, al igual que el Receptor. Por otro lado, la Fecha requiere únicamente un renglón y todas las columnas, lo cual equivale a 32 bits, al igual que Tipo Dato, ID y el Dato. Finalmente, la Firma ocupa 8 renglones, lo cual equivale a 256 bits. El total que requiere el paquete es de 500 bits.

ID	Fecha	Tipo Dato	Dato	Emisor	Receptor	Firma
001	1970-12-13T20:45:53+00:00	0	305.69	192.168.178.31	192.168.178.31	CCDB006926EA9565CBADC840829D8C384E05DE1F1E381B85

Tabla 8: Ejemplo de los datos que estarán dentro del paquete

Dentro de la Tabla 8, se muestra cómo es que se vería la información dentro del paquete, para así cumplir con los bits que se mostraron en la Tabla 7.

4.3. Certificados SSL

Para la elaboración del proyecto, se tuvo que implementar un auto-certificado SSL. Para este se utilizó una herramienta de configuración cero llamada mkcert, la cual crea certificados confiables de desarrollo de manera local.

Primeramente se tuvo que instalar mkcert para después asegurarnos que el Sistema Operativo del computador confíe en los certificados que se harán. Para lograr esto, se debe de instalar un Local Certificate Authority (CA) en el almacén de confianza del sistema. Una vez verificado este paso, se genera un certificado para el dominio localhost. Con esto, ya se generó un certificado SSL local que funcionará con cualquier servidor de desarrollo localhost que se ejecute en cualquiera de los puertos.

Nosotros escogimos certificado SSL debido a que consideramos que es el mejor o el más óptimo, nuestro firmado es de tipo RSA y esto no implica ningún peligro para el auditor y/o el centro de control, quien hace el certificado es la computadora y no la raspberry. Normalmente este certificado es emitido por una autoridad certificadora, nosotros lo hicimos de esta manera para poder certificarlo en nuestra red local.

4.4. Descripción de red creada

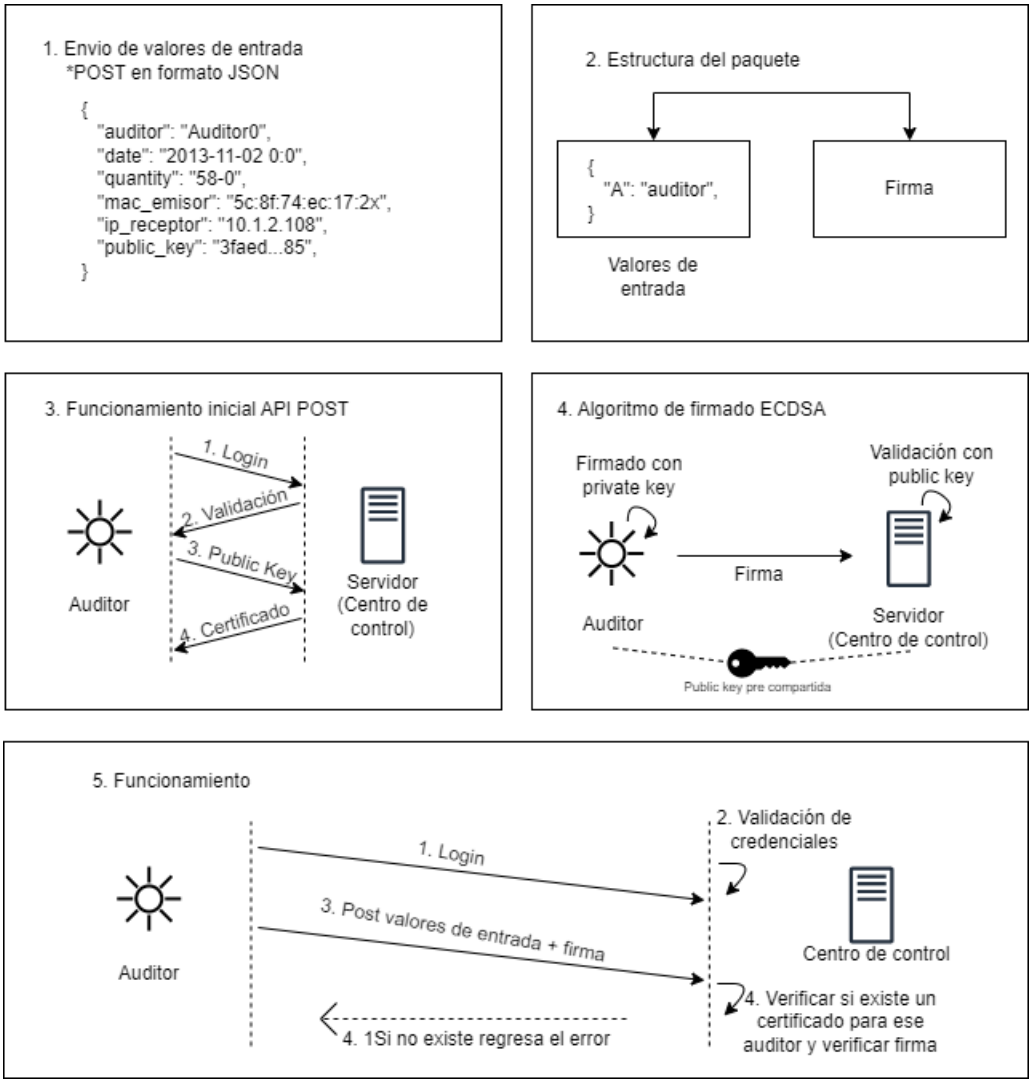

Figura 6: Diagrama de la red creada.

Este diagrama intenta explicar de forma ilustrativa el proceso que se lleva a cabo en la red que se creó. Primero se hace el envío de valores de entrada, este es un archivo de tipo JSON con el nombre del auditor, la fecha, la cantidad, la mac-address del emisor, la IP del receptor y finalmente la llave pública, cabe remarcar que todo esto es parte del paquete que hablamos en la sección 4.2. Siguiente a eso se observa la estructura del paquete, donde se tiene los valores de entrada y la firma. Después se tiene el funcionamiento inicial API POST, donde se ve que el auditor hace login para que después el servidor lo valide. Una vez validado, el auditor manda la llave pública para que el servidor haga el certificado. Después se muestra el algoritmo de firmado ECDSA, este es de llave pública y privada, donde el auditor a través de una llave simétrica manda la firma con la llave privada, y el servidor lo valida con la llave pública. Es importante remarcar que la llave pública está pre-compartida. Finalmente se tiene el funcionamiento, el auditor hace login y el centro de control hace la validación de credenciales para que después el auditor haga post de los valores de entrada y la firma. Después de esto el centro de control verifica si existe un certificado para ese auditor y verifica la firma, si no existe regresará un error. Para más información de este diagrama consultarlo en la documentación.

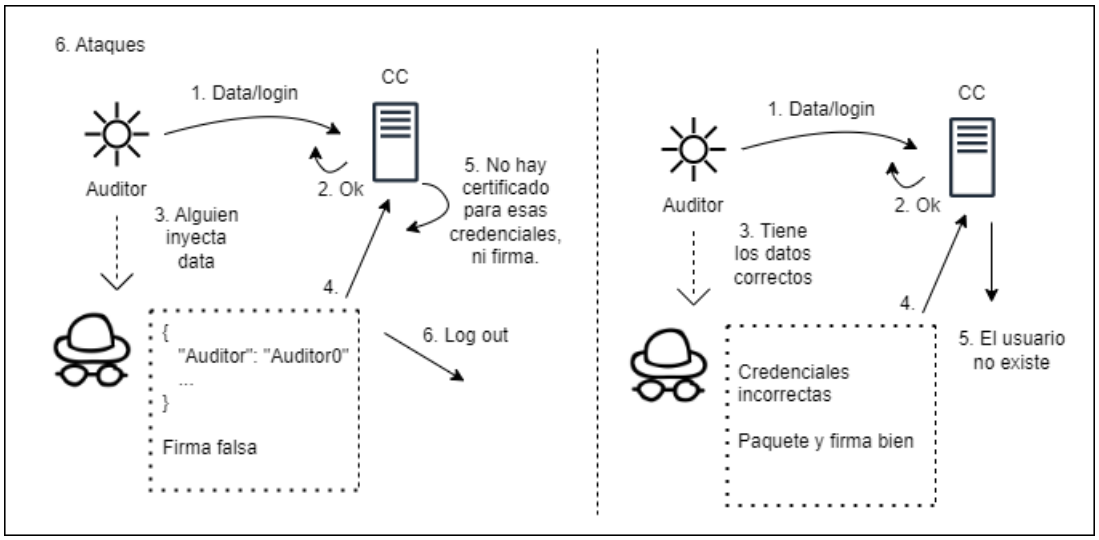

Figura 7: Diagrama de como funcionan los ataques.

En este diagrama podemos observar cómo funcionan los ataques que podrían ser recibidos en nuestro programa. Existen dos escenarios, en el primero, el auditor hace login o manda la data, y el servidor lo recibe, un hacker inyecta data falsa y la manda al servidor, pero como no hay ni certificado ni firma hace logout. El otro escenario es que el hacker en vez de inyectar una firma falsa, inyecte un paquete y firma correctas, sin embargo esto tampoco funcionara ya que las credenciales las tendrá incorrectas y el servidor le negará el acceso pensando que el usuario no existe.

4.5. Hashing

La librería con la que se trabajó el hash es HASHLIB en su versión 3.11.1. Esta librería es una de las más útiles para poder hacer hashing, aparte de que permite trabajar con los algoritmos SHA1, SHA224, SHA256, SHA384 y SHA512, aunque para este trabajo solo se trabajó con SHA256.

Con el algoritmo SHA256 en esta librería se tienen incluidas algunas funciones muy útiles como encode(). Esta función permite convertir un string en bits que, por default, utiliza el valor UTF-8. También se tiene la función hexdigest(), la cual ayuda a pasar de data codificada a formato hexadecimal, entre otras varias funciones.

```
import hashlib
string="prueba de string"
encoded=string.encode()
result = hashlib.sha256(encoded)
```

```
Output:
<sha256 _hashlib.HASH object @ 0x7fdcf63bb090>
```

Figura 8: Código de Hashlib.

En la Figura 8, se muestran las lineas de código ejemplo para utilizar la librería Hashlib. En la primera linea, se muestra cómo se importa la librería al código de Python 3 para poderla utilizar. En la segunda linea se crea un string ejemplo y se guarda en la variable "string". La tercer linea muestra cómo se codifica el string (se convierte de string a bits) y se guarda en la variable "encoded". La cuarta linea crea una variable "result" que guarda el string codificado ya convertido en un objeto tipo SHA-256.

4.6. Firmado por ECDSA.

El ECDSA es un algoritmo de firmado digital que permite asegurar la autenticidad y la integridad de los datos o mensajes mediante la utilización de un par de clave privada y pública, basado en Curvas Elípticas RFC6979 [Pornin, 1970].

La firma se genera utilizando un par de claves, una privada y una pública. La clave privada se utiliza para generar la firma y la clave pública se utiliza para verificarla. Es importante que la clave privada sea guardada de manera segura, ya que cualquier persona que tenga acceso a ella podría utilizarla para generar firmas no autorizadas.

El desarrollo del algoritmo es como se muestra a continuación:

1. Usar una curva elíptica
   - con módulo p
   - con coeficientes a, b
   - un punto A genera un subgrupo cíclico de orden q (primo)
2. Elegir un entero $0 < d < q$
3. Calcular $B = d \cdot A$
4. $k_{pr} = d$ y $k_{pub} = (p, a, b, q, A, B)$

Por otro lado, la generación de firmas es la siguiente:

1. Escoger entero (aleatoriamente) $0 < k_E < q$
2. Calcular $R = k_E \cdot A = (Xp, Yr)$
3. $r = k_R$
4. Calcular $s = (h(x) + d \cdot r) \cdot k_E^{-1} \, mod \, q$ que es la signature [h() es la función hash]

Por último, la verificación de la firma sigue los pasos a continuación:

1. Calcular el valor auxiliar $w = s^{-1}\ mod\ q$
2. Calcular el valor auxiliar $u_1 = w \cdot h(x)\ mod\ q$
3. Calcular el valor auxiliar $u_2 = w \cdot r\ mod\ q$
4. Calcular $P = u_1 \cdot A + u_2 \cdot B = (x_p, y_p)$
    - Verificar
        - Si, $x \equiv r\ mod\ q$ la firma es válida
        - Si tenemos que $x \not\equiv r\ mod\ \ q\ \ $ la firma no es válida

Lo mostrado anteriormente se hace referencia en la documentación del proyecto pero enfocado en programación. Hasta este punto, lo que se hizo fue sacar la k inversa con SHA-1 y SHA-256 en 192 Bits, y las variables que se utilizaron fueron q y k con los valores de vectores de prueba que se encuentran en la sección 7.1. La lógica utilizada es para obtener la K', se utilizó el algoritmo del inverso de Euler $k^{q-2}\ mod\ q$, para la validación se realizó $(k * k')\ mod\ n \equiv 1e$. Si la k' calculada es correcta entonces la congruencia anterior se cumplirá. A continuación fragmentos del código que en el futuro se podrán consultar a más detalle en el repositorio. Es importante aclarar que esto es con un hashing SHA-1, pero también se probó con SHA-256, que se encuentra en el repositorio, donde el resultado fue el mismo.

```
integer_q = int('FFFFFFFFFFFFFFFFFFFFFFFF99DEF836146BC9B1B4D22831', 16)
integer_k = int('D7CA00D2C7B0E5E412AC03BD44BA837FDD5B28CD3B0021', 16)
print("Q en Decimal: ", integer_q)
print("K en Decimal: ", integer_k)

Output:
Q en Decimal:  6277101735386680763835789423176059013767194773182842284081
K en Decimal:  2066848720639635734355727917297198142267815187464971881 7
```

Figura 9: Conversión del parámetro k (pre-creado pseudoaleatoriamente) y q de hexadecimal a decimal obtenidos del RFC6979.

Para la obtención de la k' se creó una función propia de inversa de Euler:

```
def inverse_euler(a, m):
    # Calcular el inverso utilizando el método de Euler
    # función pow() que sería equivalente a decir: el valor a elevado a la (m - 2) modulo m
    return pow(a, m - 2, m)

kinv = inverse_euler(integer_k, integer_q)
print("K inverso es igual a :", kinv)

Output:
K inverso es igual a : 226121287745807644814395663375658018058434950782031155264 2
```

Figura 10: Función de inversa por Euler.

Y finalmente para la evaluación de la k' se comprobó si es $k * k'$ módulo q es igual a 1.

```
(kinv * integer_k) % integer_q

Output:
1
```

Figura 11: Comprobación de la k'.

Finalmente, cabe recalcar que para la elaboración del programa se tuvo que utilizar la librería ECDSA ya que, como se explica con más detalle en la documentación y se justifica en la Sección 5.3, el uso del código elaborado por nosotros era muy ineficiente para poder llegar al objetivo de hacer verificar los paquetes enviados cada 0.15 segundos. Igualmente, es importante mencionar que esta librería únicamente se utilizó para la parte de verificación, el resto del código fue elaborado propiamente.

4.7. Formateo de la Base de Datos

Tras analizar el comportamiento de las trazas, es importante formatear los datos para facilitar el intercambio de información entre los dos Auditores y el Centro de Control. El primer paso del formateo fue separar la base de datos en 2, para crear 2 nuevas bases donde una sea de producción y la otra de consumo. Esta separación se realiza con el objetivo de enviar desde cada auditor los datos correspondientes a la producción y consumo respectivamente.

Tras tener las bases de datos separadas, se puede comenzar el formateo. Las Figuras que se mostrarán a continuación representan el proceso que se realizó para formatear la base de datos de Consumo, igualmente, dicho proceso es exactamente el mismo para la base de datos de Producción. El primer paso de formateo es acumular todos los registros de datos en una sola columna, tal y como se observa en la Figura 12. En la misma figura se crea el índice de la base de datos correspondiendo a la hora de cada registro (0:0, 0:15, sucesivamente).

```
# se crea una sola columna con los valores de cada hora
new_df0 = pd.DataFrame(df0_sub.stack())

# agregado de una columna nueva con los minutos a los que corresponde cada valor
new_df0.index = new_columns * 364
```

Figura 12: Creación de columna con registros de datos

El siguiente paso es filtrar los datos de ID, Fecha y Tipo de consumo para tener los registros. Al contar con esa información, cada valor se repite 96 veces para coincidir con la cantidad de datos por día. Con esto, cada registro en la nueva base de datos cuenta con la información adecuada de ID, Fecha y Tipo de consumo. Después, se guarda la columna de valores realizada en la Figura 13, en esta base de datos. El siguiente paso es juntar la columna de Fecha, con la columna de hora para que el registro de fecha cuente con el día y hora de cada valor. Siguiente a esto, se debe de filtrar nuevamente la base de datos gracias a que los procesos anteriores crean columnas no deseadas, y el filtrado las elimina. Y el paso final es resetear el índice de la base de datos.

```
# filtrado de base de datos con columnas requeridas
df0 = df0[["ID", "Consumo (0) / Producción (1)", "Dia"]]
df0 = df0.iloc[np.repeat(np.arange(len(df0)), 96)]

# creado de columna de valores por instante con el new_df0 ya creado
df0['Value'] = np.array(new_df0[0])

# union de la columna de fecha con la columna de minutos
df0['Fecha'] = df0['Dia'].astype(str) + " " + new_df0.index

# se filtran nuevamente solo las columnas requeridas
df0 = df0.filter(['ID', 'Fecha', 'Consumo (0) / Producción (1)', 'Value'])

# reseteo del indice de la base de datos
df0.reset_index(drop=True, inplace=True)
```

Figura 13: Formateo de la base de datos de Consumo

El formateo que se realizó en las Figuras 12 y 13 da el siguiente resultado:

	ID	Fecha	Consumo (0) / Producción (1)	Valor
1	ABC	2013-11-02 0:0	0	58.00
2	ABC	2013-11-02 0:15	0	75.00
3	ABC	2013-11-02 0:30	0	65.00
4	ABC	2013-11-02 0:45	0	0.08
5	ABC	2013-11-02 1:0	0	67.00

Tabla 9: Primeros 5 registros de la base de datos de Consumo

En la Tabla 9 se muestran los primeros 5 registros de la base de datos de Consumo. En esta se puede identificar el formato final de los datos. Dicho formato es el mismo para la base de datos de Producción.

4.8. Formateo y envío de datos por Auditor

Tras recibir los datos de las trazas ya formateados, es necesario prepararlos para que cada uno de los dos auditores puedan enviar la información. Esta sección de formateo busca agregar las direcciones MAC tanto del emisor como el receptor en las bases de datos de Consumo y Producción. Además de agregar la dirección MAC, cada uno de los registros de cada base de datos sera formateado como objeto JSON, que facilita el envío y reduce el peso de la información. El proceso mostrado es para el primer auditor que enviará la información de consumo, sin embargo, dicho proceso es el mismo para el segundo auditor que envía la información de producción. El primer paso del proceso es captar la dirección MAC de cada auditor para poder agregar la información a la base de datos. La Figura 14 muestra la linea de código utilizadas para captar automáticamente dicha dirección.

```
mac = gma()
```

Figura 14: Captura de dirección MAC de los auditores

La función gma() fue importada de la librería de Python 3.0, getmac. El siguiente paso es guardar los datos de la interacción con el centro de control. Las lineas de código utilizadas se observan en la Figura 15.

```
url = "https://0.0.0.0:8000/api/v1/entries/"
headers = {
    'Content-Type': 'application/json'
}
auth = ('admin@cocoa.com', '1234')
```

Figura 15: Guardado de datos para interacción con el CC

Los datos de URL cambiarán de la dirección IP de prueba a la dirección IP del centro de control. Los datos de header se quedarán igual y los datos de auth cambiarán a un usuario y contraseña más seguros, por lo tanto son valores prueba para lograr la comunicación.

```
def hash_dict(d):
    hash_string = ""
    for key, value in sorted(d.items()):
        hash_string += str(key) + str(value)
    return hashlib.sha256(hash_string.encode()).hexdigest()
```

Figura 16: Función para SHA-256

Para hashear la información, se creó una función que recibe los datos y los formatea con SHA-256. Esta función se observa en la Figura 16.

```
# creando las llaves de firmado
sk = SigningKey.generate(curve=NIST256p)
# private key
signing_key_hex_string = sk.to_string().hex()
# public key
verifying_key_hex_string = sk.verifying_key.to_string().hex()
```

Figura 17: Creación de llaves para firmado

La Figura 17 muestra las lineas de código donde se crean las llaves públicas y privadas. Para crear las llaves se utiliza la curva tipo NIST256p que tiene 256 bits de tamaño.

```
df0 = pd.read_csv("../Archivos_trazas/auditor0.csv")
for i in range(10):
    df0.loc[i, 'mac_emisor'] = mac
    payload = df0.iloc[i].to_json()
    jpayload = json.loads(payload)
    hashed = hash_dict(jpayload)

    # creando la firma
    sk = SigningKey.from_string(bytearray.fromhex(signing_key_hex_string), curve=NIST256p)
    signature = sk.sign(hashed.encode('utf-8'))
    signature_hex_string = signature.hex()

    # payload final (incluye la firma)
    df0.loc[i, 'signature'] = signature_hex_string
    new_payload = df0.iloc[i].to_json()
    new_jpayload = json.loads(new_payload)

    response = requests.post(url, json=new_jpayload, headers=headers, auth=auth, verify=False)
    print(response.status_code)
    print(response.text)
    time.sleep(.15)
```

Figura 18: Envío de datos del Auditor 0 al CC

La Figura 18 muestra las lineas de código que preparan y envían la información desde cada auditor al centro de control. La información es cargada como df0 ya que el ejemplo mostrado es para el primer auditor. En estas mismas lineas se agrega la dirección MAC a la base de datos y se guarda cada registro en formato JSON. Tras tener los datos en el formato requerido, se procede a realizar la interacción con el Centro de Control. En esta figura se muestra que el for loop solo corre 10 veces, esto cambiará a que se corra una vez por registro de la base de datos ya que se terminen las pruebas de interacciones. Dentro de cada interacción se crea una separación de tiempo de 0.15 segundos para no saturar el servidor de información.

4.9. Descripción del Centro de Control

El centro de control está dividido en 2 partes, el sitio del administrador y el modelo del centro de control. Para el segundo, se utiliza un modelo con el nombre y el dominio del centro de control que se va a utilizar para los certificados y la verificación de los auditores. Pero el manejo del Centro de Control es directamente desde el sitio del administrador.

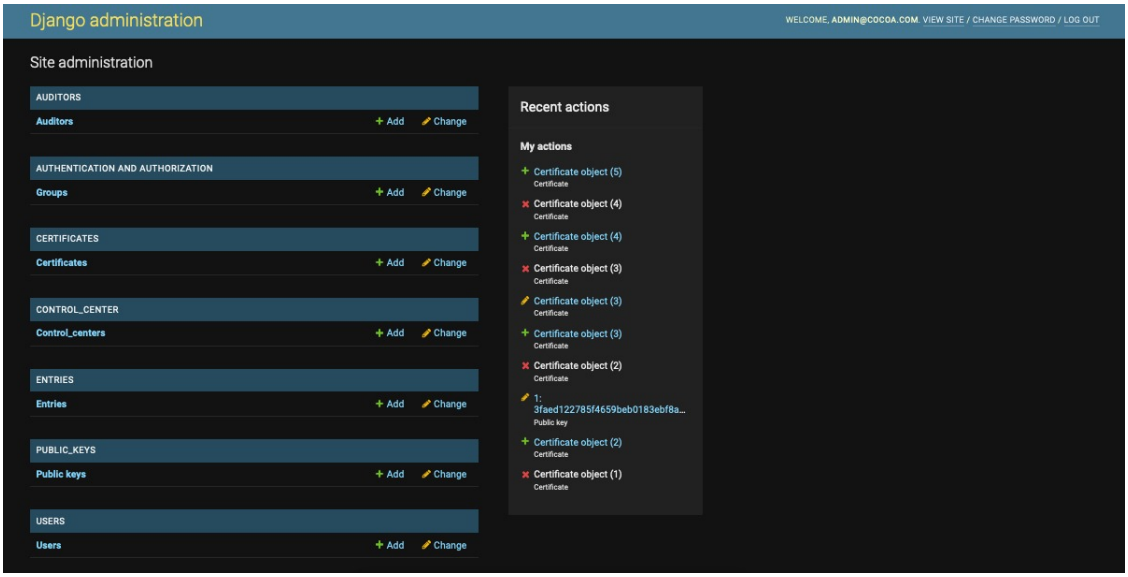

Figura 19: Administrador Django

La Figura 19 muestra la visualización del administrador. Dentro de aquí se pueden ver todas las características de la arquitectura de la propuesta. El administrador tiene el poder de ver, agregar y cambiar la información de los auditores, usuarios, entradas, etc. Cabe mencionar que cada modelo tiene sus propios permisos, pero a excepción de un par de casos, el administrador tiene todos los permisos. También, el sitio del administrador puede ver las acciones que se han realizado recientemente.

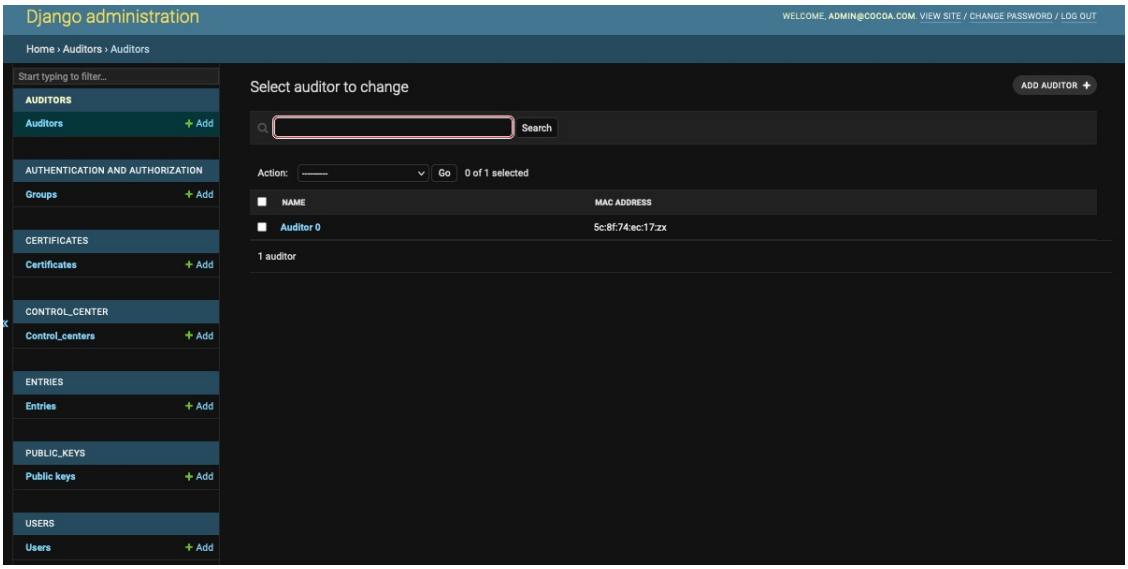

Figura 20: Selección del Auditor

En la Figura 20 se muestra el modelo/información de los auditores, con acciones como agregar y seleccionar. La información que se muestra es: nombre y dirección MAC.

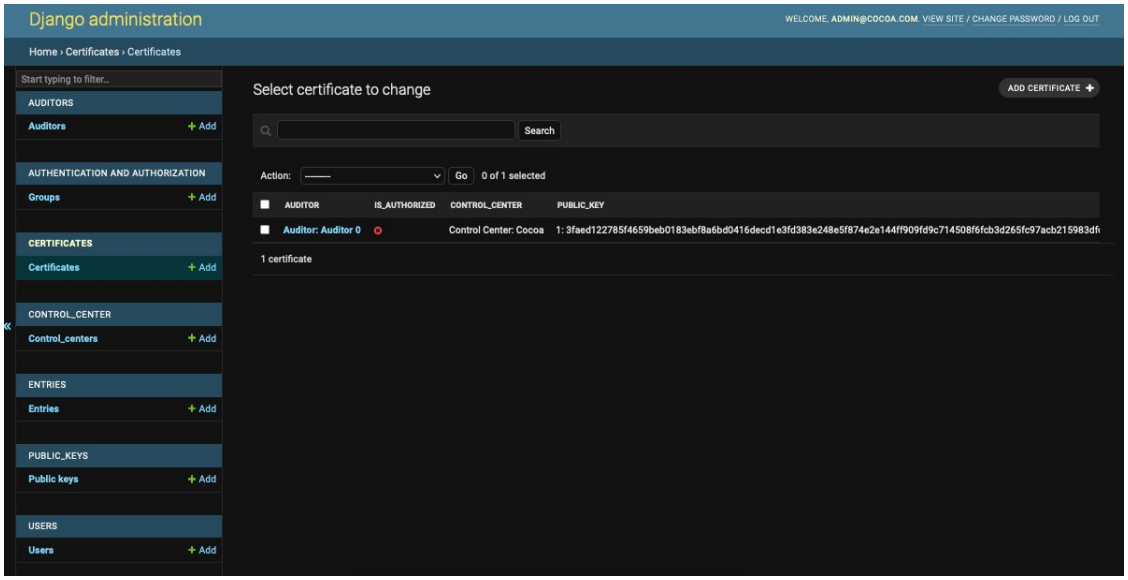

Figura 21: Selección del Certificado

La Figura 21 muestra el modelo de los certificados. A diferencia de los otros modelos que describen la información que tiene cada modelo o tabla. El modelo de certificado junta la relación entre los auditores, centro de control y llave pública, ya que es la información que se necesita para verificar y crear un certificado. También existe un campo donde verifica si está autorizado el certificado o no.

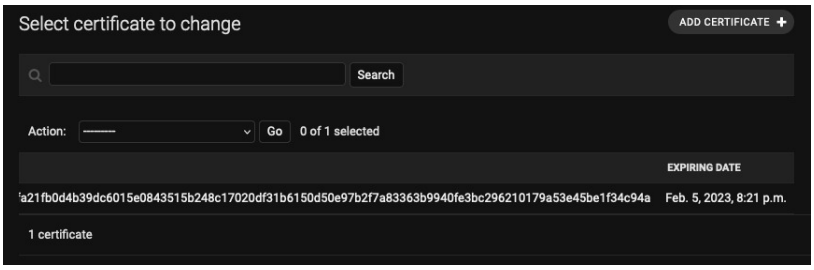

Figura 22: Expiración del Certificado

El último campo del Certificado, como se muestra en la Figura 22, es la expiración del mismo. Esto ayuda en la verificación del certificado y tomar las acciones correspondientes dependiendo de dicha verificación.

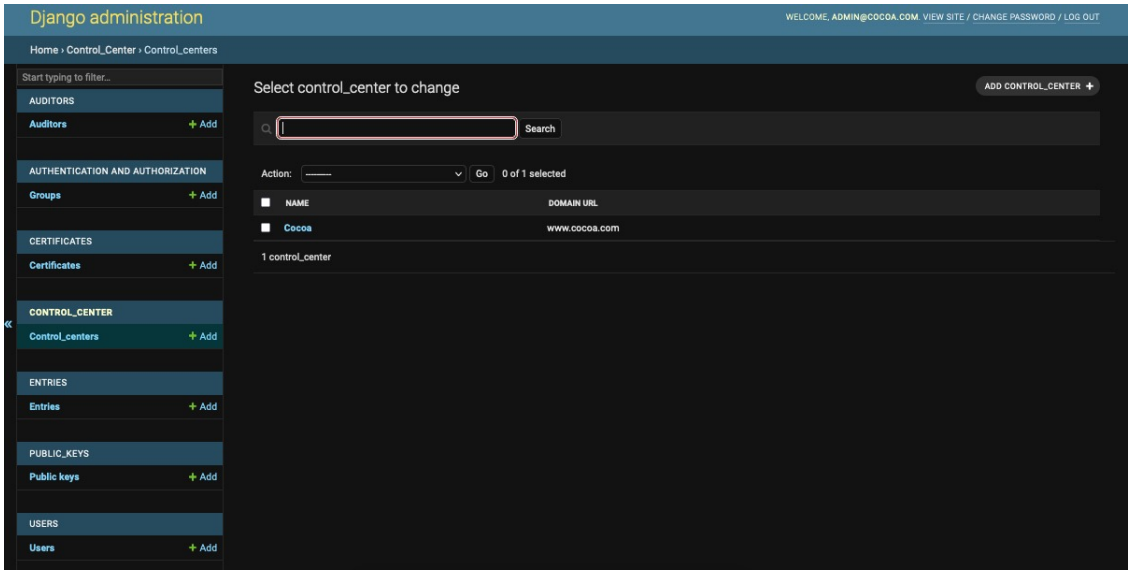

Figura 23: Selección del Centro de Control

La Figura 23 muestra el Centro de Control. Los campos que tiene es el nombre y el URL del dominio. Se realizó de esta manera para poder darle escabilidad al código, permitiendo que existan varios centros de control, cada uno con su nombre y dominio.

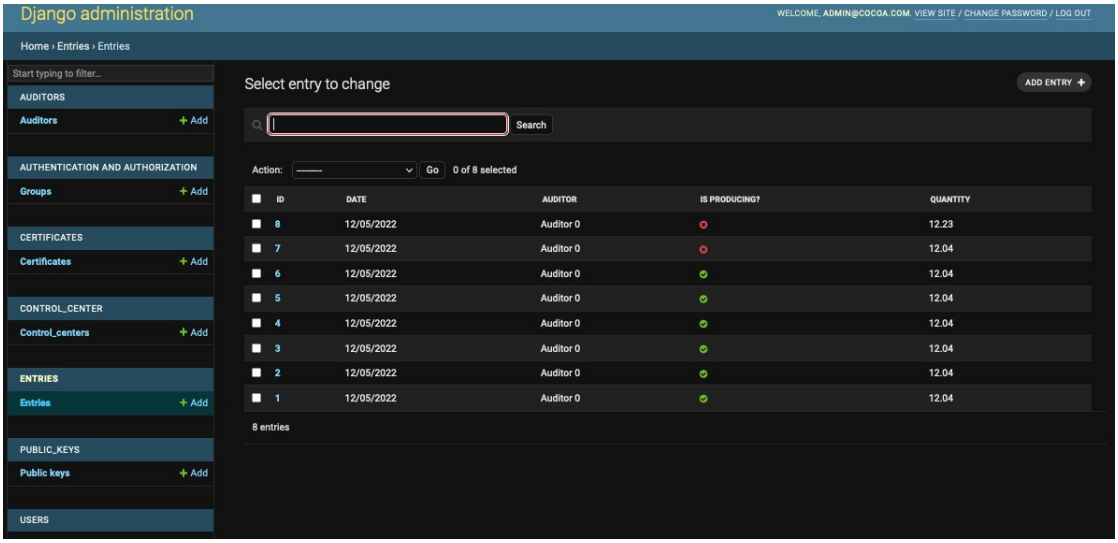

Figura 24: Selección de las entradas

La Figura 24 muestra el modelo de las Entradas que se van a recibir, incluye el auditor que lo mandó, la fecha y tiempo, si está produciendo (si está en verde) o consumiendo (si está en rojo) y la cantidad.

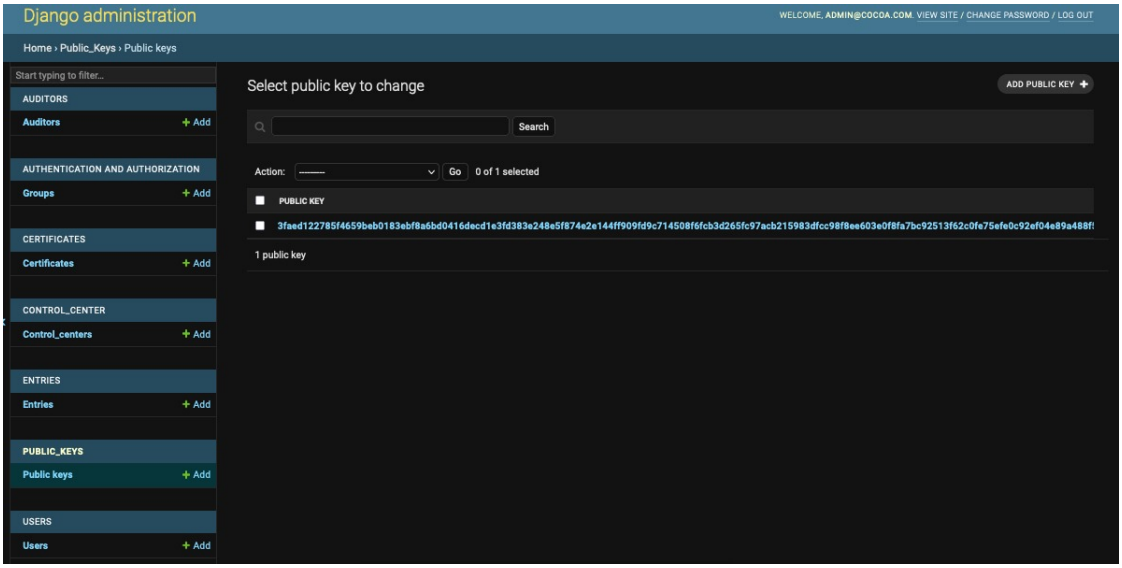

Figura 25: Selección de Llave Pública

La Figura 25 muestra como se guarda la llave pública, es un campo Hexadecimal que necesita una longitud mínima de 256 para que sea válida.

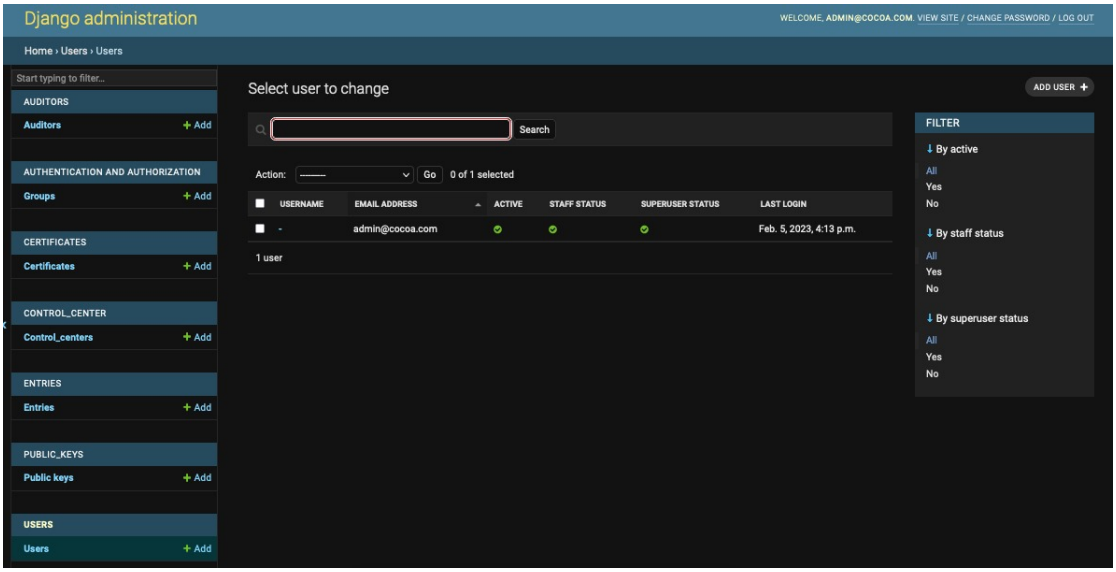

Figura 26: Selección de usuarios

Finalmente, el último modelo es el de los usuarios. Donde se muestra los usuarios existentes, si están activos, si son administradores y cuándo fue la última vez que estuvieron conectados.

Conociendo el sitio del administrador y lo que se muestra, también es importante entender a dónde se mandan las solicitudes para ver, agregar, cambiar o borrar información de cada modelo.

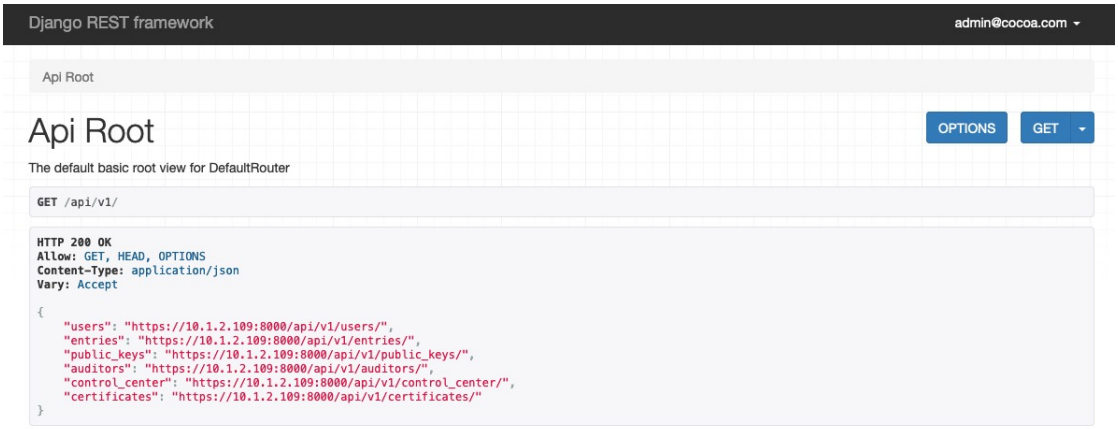

Figura 27: API Root

En la Figura 27 se puede ver los diferentes URLs (usuarios, entradas, llaves públicas, auditores, centro de control y certificados) que se deben de utilizar para mandar alguna solicitud. Por ejemplo, si se quiere hacer una solicitud de agregar una entrada, se mandaría al URL con terminación $v1/entries$.

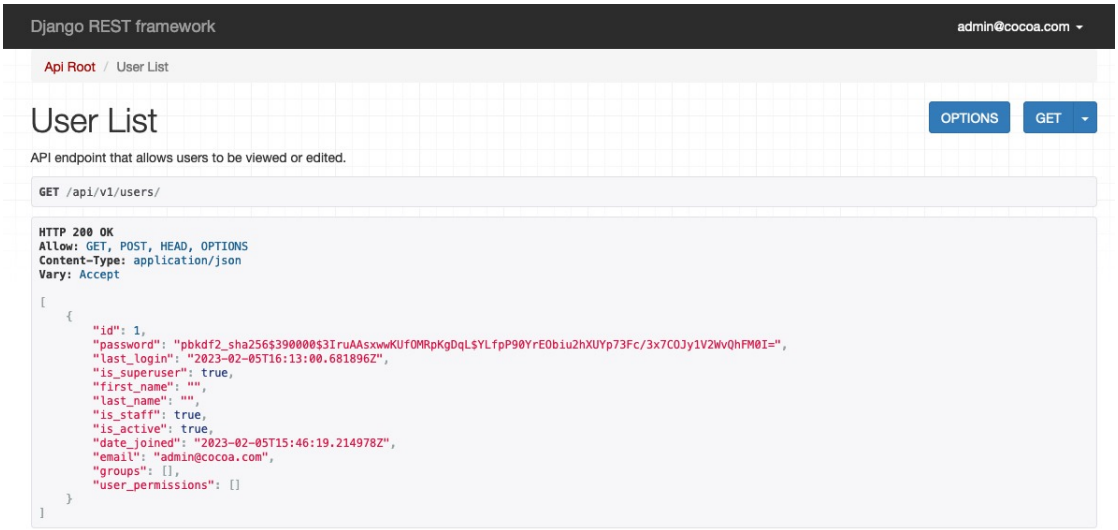

Figura 28: Lista de Usuarios

Dentro de cada URL se muestra la misma información que el sitio del administrador pero de su forma JSON, tal y como se observa en las Figuras 28, 29, 30, 31, 32 y 33.

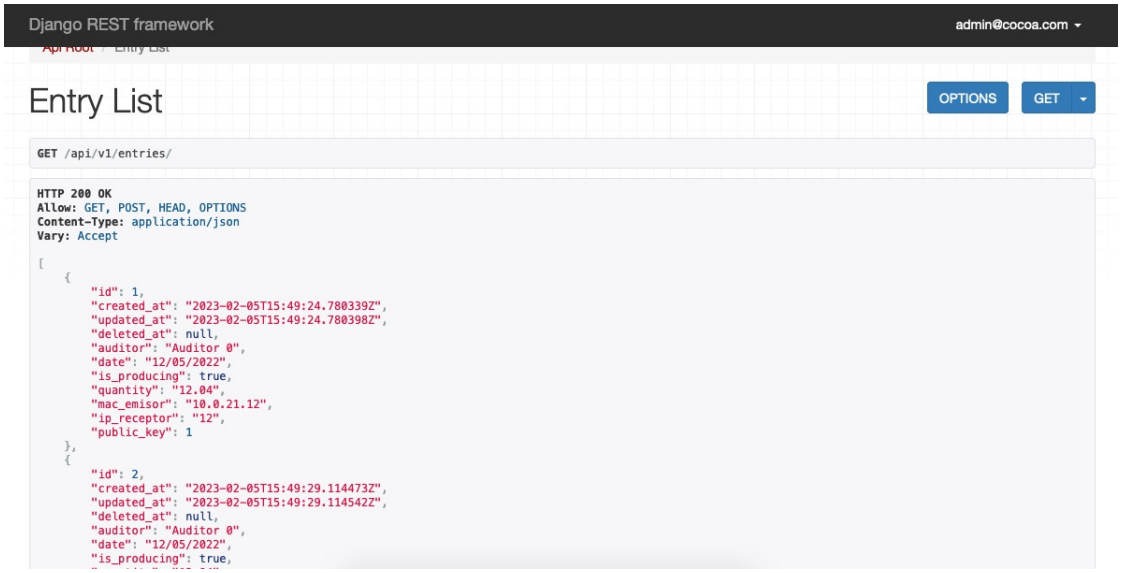

Figura 29: Lista de entradas

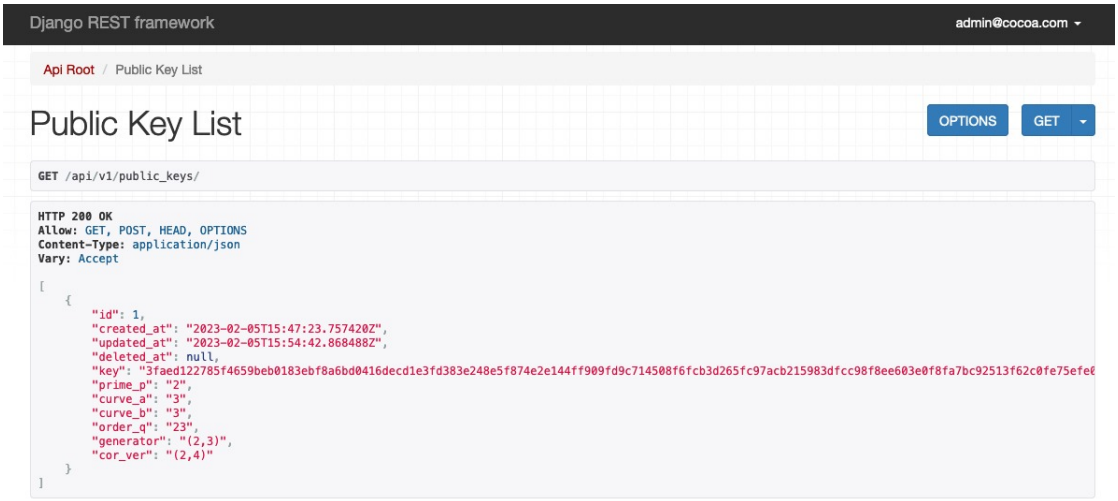

Figura 30: Lista de Llaves Públicas

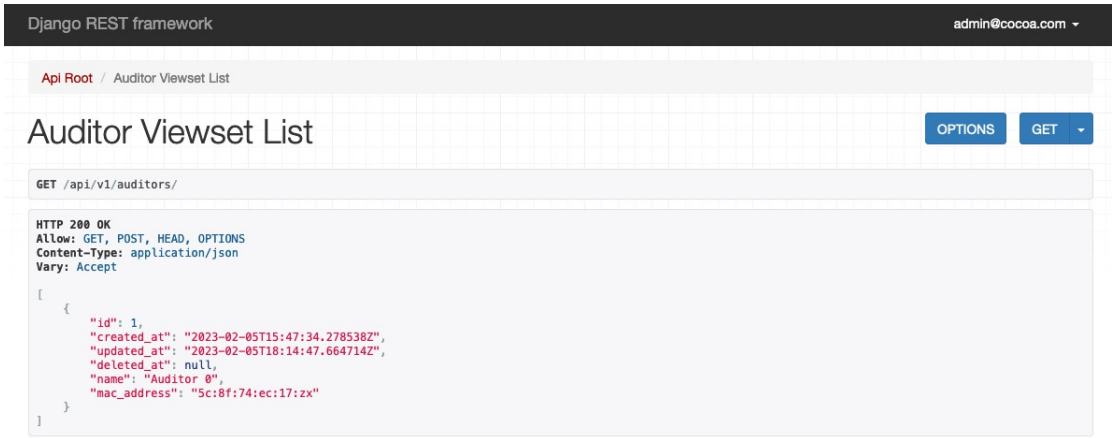

Figura 31: Lista de Auditores

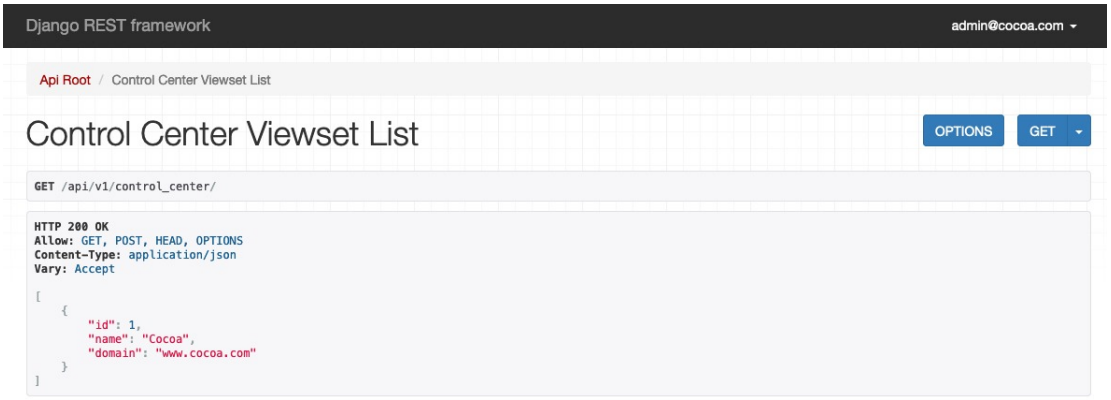

Figura 32: Lista de Centro de Control

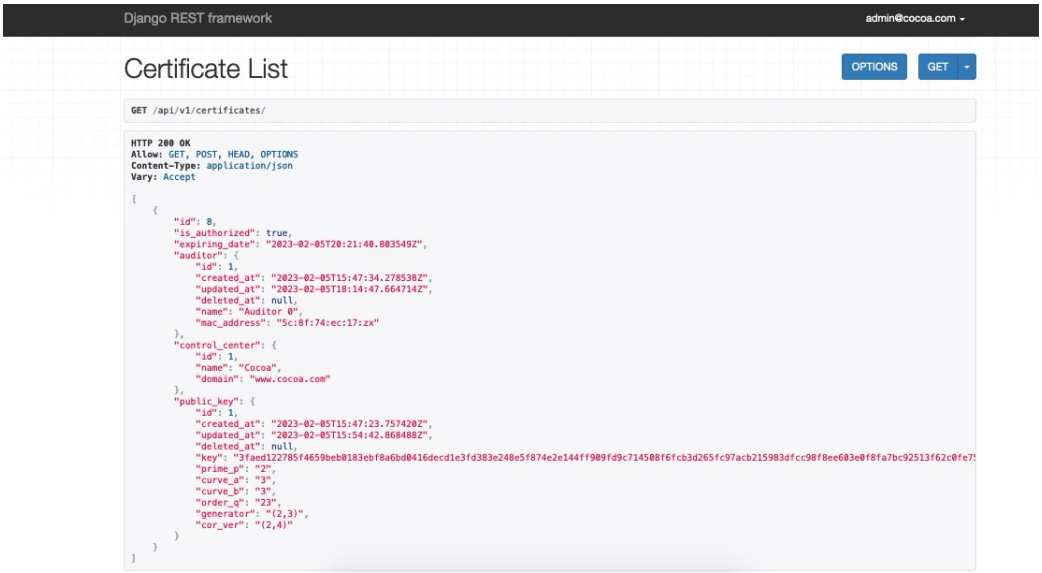

Figura 33: Lista de Certificados

4.10. Descripción del Certificado

El certificado utilizado fue creado por el equipo. Esto ya que el equipo es capaz de especificar con mayor detalle los factores que se requieren de un certificado estándar, para que la comunicación entre el CC y los Auditores sea segura. La Figura 34 muestra la construcción del certificado utilizando Python 3.

```python
class Certificate(models.Model):
    auditor = models.OneToOneField(
        "auditors.Auditor",
        verbose_name=_("auditor"),
        related_name="certificates",
        on_delete=models.SET_NULL,
        null=True,
        blank=True,
        unique=True,
    )
    control_center = models.ForeignKey(
        "control_center.ControlCenter",
        verbose_name=_("control_center"),
        related_name=_("certificates"),
        on_delete=models.SET_NULL,
        null=True,
        blank=True,
    )
    public_key = models.OneToOneField(
        "public_keys.PublicKey",
        verbose_name=_("public_key"),
        related_name="certificates",
        on_delete=models.SET_NULL,
        null=True,
        blank=True,
    )
    is_authorized = models.BooleanField(_("is_authorized"), default=True)
    expiring_date = models.DateTimeField(default=datetime.now(pytz.UTC) + timedelta(minutes=120),
    ↪  editable=False)


    def check_expiry(self):
        now = datetime.now(pytz.UTC)
        if self.expiring_date is not None and now >= self.expiring_date:
            self.is_authorized = False
            self.save()
            return False
        return True

    class Meta:
        verbose_name = _("certificate")
        verbose_name_plural = _("certificates")
```

Figura 34: Certificado

Como se puede observar, el certificado que se creó está limitando las acciones disponibles para la comunicación al definir con clases. En el certificado se crean 4 objetos: el auditor, el Centro de Control, la llave pública y la autorización. Dentro de los objetos se especifica las acciones que se pueden realizar y se les otorga un nombre. Además, el certificado debe de tener una vigencia, la cual está establecida con un timedelta. En el código mostrado en la Figura 38, se observa que la vigencia del certificado es para 120 minutos; este tiempo no es el final y es únicamente para realizar las pruebas adecuadas de autorización. El tiempo del certificado será a 1 año. Dicha fecha de expiración aparece en la API del CC como fue mostrado en el apartado anterior.

5. Resultados

5.1. Trazas del OSF.

Por parte de LiCore se recibió un archivo en formato CSV, se realizó un análisis sencillo de los datos contenidos en el archivo para entender mejor el contexto y determinar qué información estaba disponible. Una vez teniendo dicho contexto sobre las razones de los datos que se enviaron, se comenzó a crear visualizaciones para poder comprender a mayores rasgos el significado de estos.

Dentro de la Figura 35, titulada como "Consumo y Producción Anual del cliente ABC" se puede observar cuánto se consumió y se produjo energía dentro del año por el cliente ABC.

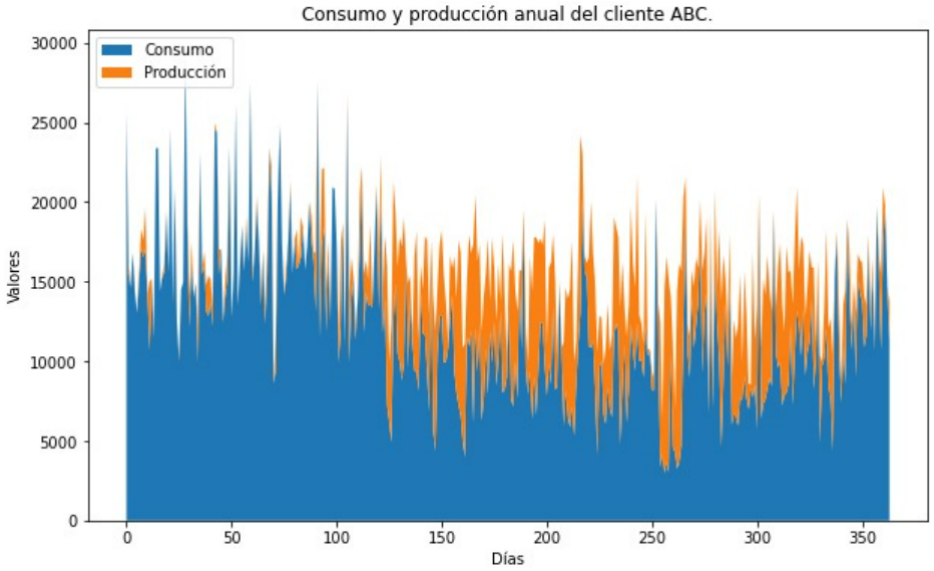

Figura 35: Consumo y Producción Anual del cliente ABC

En la Figura 36, se muestra la diferencia de consumo y producción por cada día, dando a observar de otra manera lo que se mostró en la Figura anterior. Dentro de esta se puede observar que el consumo de energía fue indudablemente mayor que la producción de esta. Por otro lado, se encontró que alrededor del mes de abril, se produjo más energía de la que se consumió.

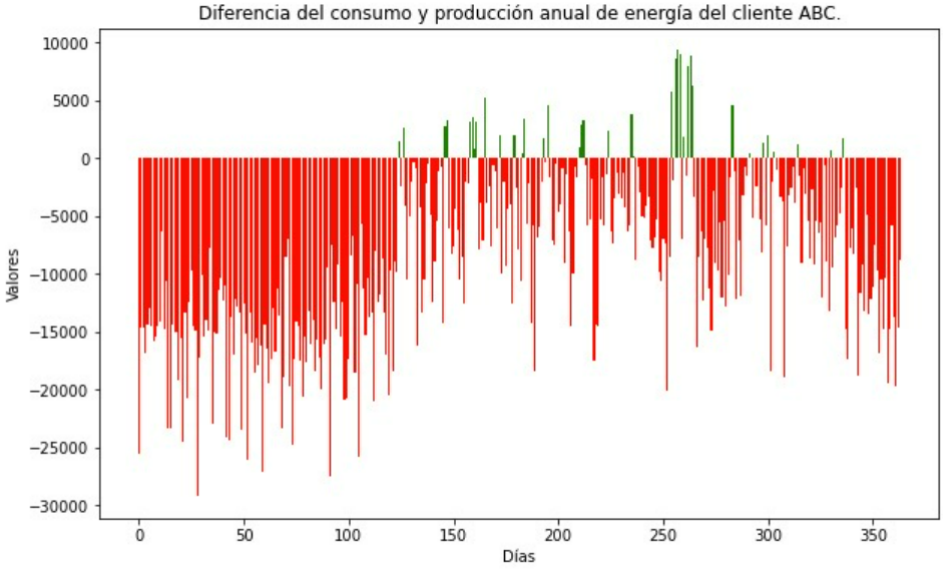

Figura 36: Diferencia del Consumo y Producción Anual de Energía del cliente ABC

Después, observando la Figura 37, se puede observar que esta representa el consumo de energía anual por hora. Claramente se puede observar que la mayoría del consumo se dio entre las 3pm y las 9pm. Igualmente es importante recalcar que el consumo entre 6am y 8am es significativo.

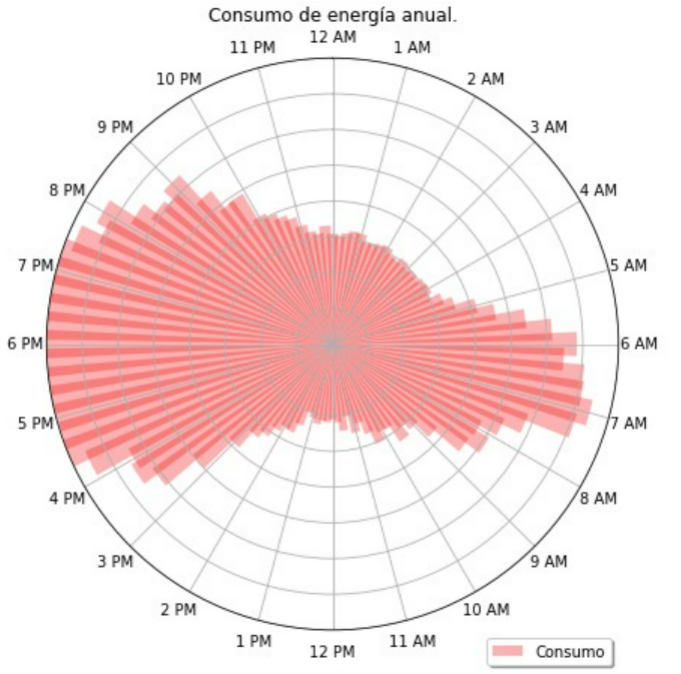

Figura 37: Consumo de Energía Anual

Por último, dentro de la Figura 38 se observa las horas en donde más se produjo energía durante el año, siendo entre las 9am y las 3pm las horas que más energía se pudo producir.

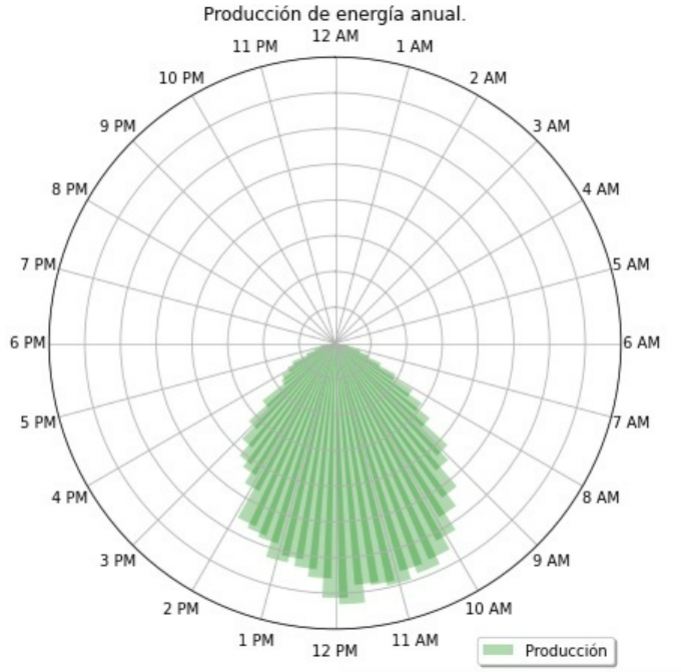

Figura 38: Producción de Energía Anual

5.2. Envío de datos desde los Auditores

Como fue mencionado en las Secciones 4.7 y 4.8, los auditores ya cuentan con la capacidad de enviar datos formateados para el Centro de Control, explícitamente explicado en la Sección 4.8 en la Figura 17 y en la Figura 18. El formato que reciben los datos se puede observar en la Figura 39.

```
{"auditor":"ABC",
    "date":"2013-11-02 0:0",
    "is_producing" : false,
    "quantity": 58.0,
    "mac_emisor" : "02:c2:c6:b0:b2:16",
    "signature":
    ↪  "22f229a6ef5a11e3fcd12928b95c5bc5e40feb6b9b25480b067ae677e348ff5bfa8971d13fa5812827faa3a7f71c1cfce
    ↪  7ab63e94aa19244ea577432994047e7"
}
```

Figura 39: Formato de datos

Como se puede observar, el formato aun no es un objeto tipo JSON. La Figura 39 muestra el ordenamiento de los datos, y la información que se puede leer es el primer registro de la base de datos de Consumo que enviara el Auditor 0. La Figura 40 muestra los datos con el formato JSON, el cual será enviado al Centro de Control.

```
{'auditor': 'ABC',
    'date': '2013-11-02 0:0',
    'is_producing': False,
    'quantity': 58.0,
    'mac_emisor': '02:c2:c6:b0:b2:16',
    'signature': '22f229a6ef5a11e3fcd12928b95c5bc5e40feb6b9b25480b067ae677e348ff5bfa8971d1
    ↪  3fa5812827faa3a7f71c1cfce7ab63e94aa19244ea577432994047e7'
}
```

Figura 40: Formato tipo JSON de datos

La firma que se envía en los datos fue obtenida con la llave privada que se puede observar en la Figura 41.

```
'8739a06de15ee6a56b442ffe7f2bb1d5821ed40a1f1e3a2bfbe48499c00b86665606e67ea80290c6d65ba964e6b6f18b54dc7f8064
↪  4c9f08f71b6d05822c1fb6'
```

Figura 41: Llave privada ejemplo

La llave privada mostrada y la firma son ejemplos para las pruebas. Por razones de seguridad y privacidad, las versiones utilizadas no se mostrarán en este reporte. La Figura 42 muestra los primeros 5 registros de la base de datos de Consumo. Estos mismos datos fueron enviados al Centro de Control.

```
{'auditor': 'ABC',
    'date': '2013-11-02 0:0',
    'is_producing': False,
    'quantity': 58.0,
    'mac_emisor': '02:c2:c6:b0:b2:16',
    'signature':
    ↪  '57f6967aed5dfd65b4c7b08f2377cc021c039cbc861a657da18d9484d75993d045c9d63f496bbb4f4f0972c54c40a1e
    ↪  7048bb7ad748237ad888b003ef8922732'
}

----------------------------------------------------------------------

{'auditor': 'ABC',
    'date': '2013-11-02 0:15',
    'is_producing': False,
    'quantity': 75.0,
    'mac_emisor': '02:c2:c6:b0:b2:16',
    'signature':
    ↪  '889fe023ef1201737b64a7dc95015ae0c28fa20e0acbd83f28ea12cd27b00aeadf853ef39bf3660142231bb10f1e3
    ↪  07e9f8f43f174723ac92b4d58d34eab6e18'
}

----------------------------------------------------------------------

{'auditor': 'ABC',
    'date': '2013-11-02 0:30',
    'is_producing': False,
    'quantity': 65.0,
    'mac_emisor': '02:c2:c6:b0:b2:16',
    'signature':
    ↪  '92d22170ac48464835332701cc11160593e136d37d443c36e58dbb62f1f6c4cd81b719145a006d4c6651cdebab0
    ↪  99815acaafc6a0669942854b232407c795e2e'
}

----------------------------------------------------------------------

{'auditor': 'ABC',
    'date': '2013-11-02 0:45',
    'is_producing': False,
    'quantity': 0.08,
    'mac_emisor': '02:c2:c6:b0:b2:16',
    'signature':
    ↪  'c88b80c384b3d5b706a4c5eead7e3e023de3b3cd38cdf77002fba5977f95ac735815cd0e131c34f7c26011437b
    ↪  2c42afd41f18a29ab02d6f248afa9b964c4caf'
}

----------------------------------------------------------------------

{'auditor': 'ABC',
    'date': '2013-11-02 1:0',
    'is_producing': False,
    'quantity': 67.0,
    'mac_emisor': '02:c2:c6:b0:b2:16',
    'signature':
    ↪  'c74ff180d11d880faa3637d93acfa8a6bfab10e2f7b4f621a97e739e926e4317841d816b1d5b5e18973e552c68
    ↪  5420b4bb7bcfd1a03dc5dfc6873b3faf571cfa'
}
```

Figura 42: Llave privada ejemplo

5.3. Verificación de los datos

Para verificar las firmas, se usa la librería de ECDSA en Python 3. Esta cuenta con las funciones requeridas para poder verificar firmas. Como la firma se realizo con los datos hasheados, para verificarla se debe den de hashear los datos recibidos. Al contar con eso se utiliza la clave publica y la misma curva que se utilizo para crear la firma, que es la NIST256p. La Figura 43 muestra las lineas de código utilizadas para verificar

las firmas.

```
verifyingkey = VerifyingKey.from_string(bytearray.fromhex(verifying_key_hex_string), curve=NIST256p)
signature = bytearray.fromhex(signature_hex_string)
print('Verify transmited data', verifyingkey.verify(signature, hashed.encode('utf-8')))
```

Figura 43: Verificación de llaves

La razón por la que se utilizó dicha librería para la verificación es debido a que el código que se hizo era muy ineficiente. Lo que se buscaba con este paso es que se hiciera la verificación cada 0.15 segundos, sin embargo, el código que se hizo con ayuda del Dr. Mancilla tardaba aproximadamente 75 minutos por cada paquete, por lo que se optó utilizar una librería externa, usando únicamente la parte de verificación de esta la cual es la Figura 43.

5.4. Recepción de los datos en el Centro de Control

Una vez verificado los datos con la firma y el certificado, el Centro de Control recibe y guarda la información en la base de datos. La representación desde el sitio del administrador se mostró en la sección 4.8 en la Figura 24, mientras que la representación desde el API (en formato JSON) está mostrada en la Figura 29 de la misma sección.

5.5. Bidireccionalidad entre Auditor y Centro de Control

Para la comunicación bidireccional entre el auditor y el centro de control se empleo el mismo principio que la recepción desde el auditor al CC. Pero en este caso el auditor le hace una petición de datos al centro de control mediante un modelo Output), compartiéndole su id y haciendo un GET de los datos de Outputs, después el centro de control verifica si el auditor tiene certificado y además si es que tiene un mensaje que compartirle le retorna por medio del GET los datos, el auditor si es que recibe los datos verifica el mensaje y ejecuta la acción que le pide el Centro de Control, en este caso si el mensaje es Upload se enciende un LED, si es Shutdown se apaga el LED, si no es ninguno de esos solo manda el mensaje pero no hace nada, cabe recalcar que el modelo que se creó solo permite mandarle un mensaje a cada Auditor y esto puede ser de manera simultanea o de forma independiente. Además que los certificados se pueden revocar aun cuando no se haya vencido el certificado.

6. Conclusiones

Durante la elaboración del proyecto se estuvo trabajando con múltiples programas, librerías, códigos y artículos de divulgación científica para poder obtener los resultados mencionados anteriormente. Con base en todo el trabajo realizado, se considera que se cumplieron los requisitos establecidos por el Socio Formador (Confidencialidad, Autenticidad e Integridad), completando cada uno de estos con el objetivo de ayudarle, al igual que a desarrollar nuestro conocimiento en el tema.

Para poder crear el proyecto, se tuvo que aprender la teoría detrás de múltiples algoritmos para, al final, seleccionar el que tiene todas las características que se ocupaban para la implementación del reto. Este algoritmo elegido fue el ECDSA, el cual son las Curvas Elípticas para el Algoritmo de Firma Digital.

Una vez entendiendo dicho algoritmo, se comenzó a desarrollar un programa en Django REST Framework para poder hacer el envío de paquetes a un Centro de Control, verificarlos y finalmente validarlos, todo de manera local. Después de haber completado este paso, se encontró que evidentemente el sistema creado por nosotros funcionaba para las trazas que fueron enviadas por el Socio Formador y que se podría usar de manera global.

Cabe recalcar que hay varias limitantes en el sistema creado, por lo que se creó una To-Do List, mostrada a continuación, en donde cualquiera que haya seguido los pasos que se mencionaron en el documento pueda continuar con su desarrollo y optimización, al igual que para poder ser implementado para cualquier proyecto que se busque aplicar.

6.1. Trabajo a futuro

Aunque este proyecto es bastante completo y eficiente, por problemas con el tiempo se tuvieron que dejar varios puntos a recomendación para futuro. Se considera que con estas adiciones al trabajo se puede llegar a tener un programa completo y que, sobre todo, pueda mantenerse actualizado con los algoritmos y las tecnologías más nuevas que se vayan desarrollando. Lo que se recomienda es lo siguiente:

1. Implementación en línea.
2. Posibilidad de certificarse por una entidad certificadora.
3. Hacer que el centro de control certifique a los auditores por SSL.
4. Optimizar el algoritmo de ECDSA.

El primer punto es el que se considera el más importante. El programa que se hizo funciona solamente de forma local con una máquina virtual. Este es el problema más grande y la corrección más importante para el trabajo. Es por esto que es de gran importancia poder hacer funcionar el programa desde internet, con ayuda de un dominio para facilitar su uso y hacerlo realmente práctico y viable.

El Punto (2) es también bastante importante ya que actualmente los desarrolladores son los que proporcionan el certificado, por lo que se considera que, para tener mayor seguridad y para que el protocolo sea más profesional, se recomienda buscar a una empresa tercera para que proporcione los certificados lo cual sería de gran ayuda para aumentar la seguridad del proceso.

El trabajo a futuro (3) va de la mano con el anterior ya que se está buscando que la empresa tercera que haga los certificados los haga por medio de SSL (Secure Socket Layer). Este verifica la identidad de la organización y garantiza la seguridad de la información transmitida a través del sitio web, por eso se recomienda que este sea el protocolo indicado. Actualmente se hace con un modelo propio que no tiene los beneficios del SSL.

Finalmente se considera que es bastante viable buscar optimizar el algoritmo ECDSA que se implementó. Esto debido a que, porque la forma en que se hizo, si bien funciona, consume bastantes recursos computacionales y a su vez tiempo. Con ayuda de librerías se podría reducir de forma considerable el tiempo que toma de correr, y sobre todo los recursos que se necesitan para hacerla.

7. Anexos.

7.1. Vectores de prueba para firmado ECDSA.

Se realizaron diferentes evacuaciones con vectores de prueba para poder realizar firmado con ECDSA, solamente comprobando si al calcular la k' es correcta. Los vectores de prueba para 192 bits fueron los siguientes:

- Para todos los SHA:
  - q = FFFFFFFFFFFFFFFFFFFFFFFF99DEF836146BC9B1B4D22831
  - q = n

- qlen = 192 bits
- Private key: x = 6FAB034934E4C0FC9AE67F5B5659A9D7D1FEFD187EE09FD4
- Public key: U = xG
- Ux = AC2C77F529F91689FEA0EA5EFEC7F210D8EEA0B9E047ED56
- Uy = 3BC723E57670BD4887EBC732C523063D0A7C957BC97C1C43

- En SHA-1, message = ”sample”:
  - k = 37D7CA00D2C7B0E5E412AC03BD44BA837FDD5B28CD3B0021
  - r = 98C6BD12B23EAF5E2A2045132086BE3EB8EBD62ABF6698FF
  - s = 57A22B07DEA9530F8DE9471B1DC6624472E8E2844BC25B64
- En SHA-256, message = ”sample”:
  - k = 32B1B6D7D42A05CB449065727A84804FB1A3E34D8F261496
  - r = 4B0B8CE98A92866A2820E20AA6B75B56382E0F9BFD5ECB55
  - s = CCDB006926EA9565CBADC840829D8C384E06DE1F1E381B85

Bibliografía

Alassaf, N., & Gutub, A. (2019). Simulating light-weight-cryptography implementation for IOT Healthcare Data Security Applications. *International Journal of E-Health and Medical Communications*, *10*(4), 1-15. https://doi.org/10.4018/ijehmc.2019100101

Alzubi, J. A. (2021). Blockchain-based Lamport Merkle digital signature: authentication tool in IoT healthcare. *Computer Communications*, *170*, 200-208.

ASF. (2004). APACHE. https://www.apache.org/licenses/LICENSE-2.0

Cantero, G. (2020). itextsharp. https://www.programandoamedianoche.com/2010/06/como-firmar-un-documento-pdf-desde-c-sharp-con-itextsharp/

Córdoba, D. (2022). RSA: ¿Cómo funciona este algoritmo de cifrado? https://juncotic.com/rsa-como-funciona-este-algoritmo/

Dai, W. (2014). Crypto++® Library 8.7. https://cryptopp.com/

Elhoseny, M., Shzankar, K., Lakshmanaprabu, S. K., Maseleno, A., & Arunkumar, N. (2018). Retracted article: Hybrid Optimization with cryptography encryption for medical image security in internet of things. *Neural Computing and Applications*, *32*(15), 10979-10993. https://doi.org/10.1007/s00521-018-3801-x

Guerrero, J. (2014). Arduino Uno: Especificaciones y características. https://pluselectric.wordpress.com/2014/09/21/arduino-uno-especificaciones-y-caracteristicas/

Hajtas, M. (2021). Escritorio ordenador portatil oficina industria. https://www.pexels.com/es-es/foto/escritorio-ordenador-portatil-oficina-industria-11700195/

Hernandez, L. (2022). Esp8266 todo lo que necesitas saber del Módulo WIFI Para Arduino. https://programarfacil.com/podcast/esp8266-wifi-coste-arduino/

Huertos, A. A. (2019). Raspberry pi 4 vs Raspberry Pi 3, ¿Qué mejora en el Nuevo Modelo? https://computerhoy.com/noticias/tecnologia/raspberry-pi-4-vs-raspberry-pi-3-mejora-nuevo-modelo-443801

Jena, B. K. (2022). Digital Signature Algorithm (DSA) in Cryptography: A complete guide: Simplilearn. https://www.simplilearn.com/tutorials/cryptography-tutorial/digital-signature-algorithm

Kalin, M. (2019). Getting started with openssl: Cryptography basics. https://opensource.com/article/19/6/cryptography-basics-openssl-part-1

KG, S. G. C. (2023). SETAPDF-Core Manual. https://manuals.setasign.com/setapdf-core-manual/standard-encryption/

Kittur, A. S., Jain, A., & Pais, A. R. (2017). Fast verification of digital signatures in IoT. *International Symposium on Security in Computing and Communication*, 16-27.

Liu, Z., Huang, X., Hu, Z., Khurram Khan, M., seo hwajeong, h., & Zhou, L. (2016). On emerging family of elliptic curves to secure internet of things: ECC comes of age. *IEEE Transactions on Dependable and Secure Computing*, 1-1. https://doi.org/10.1109/tdsc.2016.2577022

Martínez Herrera, A. F. (2023). Comentarios de la clase del 10 de Enero.

Mughal, M. A., Luo, X., Ullah, A., Ullah, S., & Mahmood, Z. (2018). A lightweight digital signature based security scheme for human-centered internet of things. *IEEE Access*, *6*, 31630-31643. https://doi.org/10.1109/access.2018.2844406

Niazi, R. (2022). Introduction to digital signature algorithm (DSA). https://www.makeuseof.com/introduction-to-digital-signature-algorithm/

Nvidia. (s.f.). Comprar El Kit Para Desarrolladores Nvidia jetson nano. https://www.nvidia.com/es-la/autonomous-machines/embedded-systems/jetson-nano-developer-kit/

ONU. (2020). Objetivos y metas de desarrollo sostenible. https://www.un.org/sustainabledevelopment/es/sustainable-development-goals/

Pornin, T. (1970). Deterministic usage of the digital signature algorithm (DSA) and elliptic curve digital signature algorithm (ECDSA). https://www.rfc-editor.org/rfc/rfc6979

Razo Zapata, I. S. (2023). Reto. https://experiencia21.tec.mx/courses/344009/pages/reto

S Naik, E. S. (2019). Smart healthcare monitoring system using raspberry Pi on IoT platform. *ARPN Journal of Engineering and Applied Sciences*, *14*(4), 872-876.

SSL.com, E. d. s. d. (2021). ¿Qué es la criptografía de curva elíptica (ECC)? https://www.ssl.com/es/preguntas-frecuentes/C2BFQuC3A9-es-la-criptografC3ADa-de-curva-elC3ADptica3F/

Vasanth K, V. R., Mounika Macharla. (2019). A Self Assistive Device for Deaf Blind People Using IOT. https://doi.org/10.1007/s10916-019-1201-0

Veeramanickam, M., & Mohanapriya, M. (2016). IOT enabled Futurus Smart campus with effective e-learning : I-campus. http : / / www . etc . edu . cn / public / 2019 / 2 / b2750bfb - f75f - 42c6 - aa3f - 526a7e177823.pdf