



Instituto Tecnológico y de Estudios Superiores de
Monterrey

Escuela de Ingeniería y Ciencias

Ingeniería en Ciencia de Datos y Matemáticas

**Implementación de criptografía de clave pública para
protección de comunicaciones**

Mauricio Soria Gutiérrez A01720983 Diego Alejandro Flores Meza A00831191
Andrés Saldaña Rodríguez A01721193 José Eugenio Morales Ortiz A01734612
Gilberto Yárritu A01721594 Lester Santiago Garcia Zavala A01721128
Rodrigo López Gómez A01275067

Profesores: Alberto Francisco Martínez Herrera y Daniel Otero Fadul LiCore

Monterrey, Nuevo León. 17 de junio de 2023

Resumen

Dentro de este trabajo lo que se realiza es una solución de bajo costo para la envío y almacenamiento de información de lectores de energía, de manera que se puedan enviar desde los auditores de manera encriptada y firmada digitalmente lecturas de datos de consumo de energía hacia un centro de control para ser almacenados de manera segura y de igual manera que el centro de control pueda comunicarse con los auditores para recibir nuevas instrucciones. Mediante el uso de conceptos de criptografía como lo son el firmado digital ECDSA, encriptación RSA y también protocolos de envío de datos de manera segura se pudo utilizar Raspberry Pi para enviar información a nuestro centro de control en AWS EC2 para finalmente ser almacenados de manera segura mediante el uso de una base datos relacional en AWS RDS.

1. Introducción

En el contexto actual, la distribución de la energía para el uso público ya no es eficiente dado la centralización que esta tiene no da la oportunidad para poder implementar el uso de energías renovables.

Para esto se busca la implementación de dispositivos SMART que se encuentren encargados de monitorear el consumo de energía de un lugar en particular que cuente con generadores de energía sustentable en el cual dependiendo de la demanda de energía y la disponibilidad de la misma, esta se pueda regular de tal manera que se tenga un uso óptimo de los recursos, un esquema general de su funcionamiento es el siguiente.

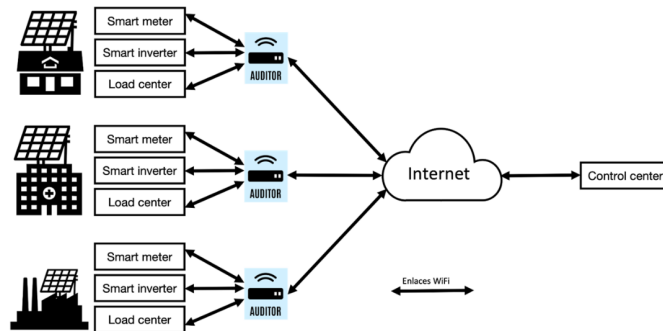


Figura 1: Esquema base de intercambio de datos, Canvas

Para dicha situación en particular se estará trabajando con LiCore, una empresa que busca revolucionar la forma en la que manejamos la energía en nuestro hogar. LiCore tiene fe en un futuro verde, en el cual la energía renovable es abundante. Parte de este futuro es que los hogares cuenten con sus propias fuentes de energía renovable. La energía renovable más accesible para los hogares son los paneles solares ya que estos son fáciles de instalar y no tiene un costo elevado, aunque tampoco son tan económicos que cualquiera pudiese comprarlos. En un escenario ideal los costos de los paneles bajen en un futuro y puedan ser instalados en la mayoría de las casas.

La energía que generan estos paneles depende completamente de factores externos, como la cantidad de nubes, la temporada del año, temperatura exterior, etc. Por esto la generación de energía tiende a variar mucho, y puede ser difícil aprovecharla al 100%. Es común que la energía se desperdicie. Sin embargo, LiCore está trabajando en una especie de medidor que estará al pendiente de la cantidad de energía que se está generando en la vivienda, y pudiera decidir en almacenar la energía, usar la energía, o distribuirla de otra manera. Estos medidores utilizarían los datos de consumo de una manera inteligente, sin embargo antes de poder llevar a cabo todas estas funciones de distribución es necesario que lleguen los datos a un centro de recopilación de LiCore donde se analizarán, a continuación se presenta una figura que describe de manera general la solución a implementar.

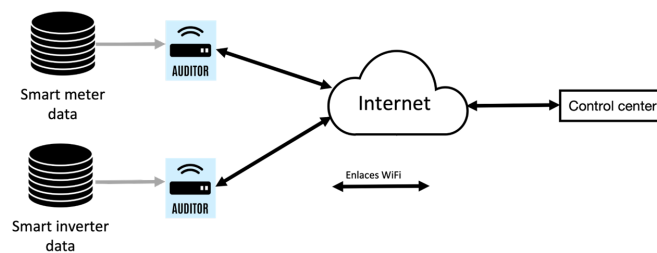


Figura 2: Esquema sugerido para solución, Canvas

Existe un factor de riesgo en la integridad de la información, ya que alguien pudiera interceptar estos datos sensibles y robarlos. Como LiCore quiere implementar esta nueva funcionalidad de medidores inteligentes, se requiere que estos cumplan con al menos las siguientes tres características:

1. **Confidencialidad:** Se busca que solo aquellos que se encuentren involucrados puedan acceder a la información que proporcionan los sensores, esto también debe de ser válido cuando la información se almacene en la base de datos.
2. **Integridad:** Se desea que la información provista por los sensores no se vea alterada y también que se pueda verificar que sea así, de igual manera que con el punto anterior esto también se busca cumplir para el almacenamiento.
3. **Autenticidad:** Se busca que cada sensor pueda ser monitoreado e identificado, para entonces ser capaz de determinar qué datos se generaron.

El reto que se describió anteriormente va relacionado con algunos de los Objetivos de Desarrollo Sostenible. Con los objetivos que más se relaciona dicho reto es con el número 7, 9 y el 11. Con el número 7 se relaciona debido a que LiCore busca que las personas hagan lo que se conoce como transición energética en donde se pueda generar energía limpia en zonas urbanas o comerciales. Esto para que LiCore pueda monitorear la energía generada y demandada para poder distribuirla de manera adecuada. El número 9 se relaciona

debido a la innovación que quiere implementar la compañía al igual que fomentar la instalación de las diferentes tecnologías como lo es la solar fotovoltaica para poder cumplir con el objetivo de optimizar el uso de energía. Por último el objetivo 11 se relaciona debido a que en un futuro cuando LiCore cumpla con su objetivo las ciudades generarán electricidad a partir de energía limpia. Esto le ayudará a las ciudades a reducir el impacto ambiental y de igual manera contribuir al crecimiento económico por la cantidad de electricidad que generan y usan de manera adecuada.

2. Estado del arte

Previo a iniciar con el desarrollo de nuestro trabajo, es conveniente el investigar diversas fuentes sobre soluciones similares ya existentes que nos ayuden a tener un mejor entendimiento de la estructura, metodología y equipo utilizado para así poder decidir el mejor curso de acción a tomar.

2.1. Blockchain en privacidad y protección de datos en smart grids

Un primer ejemplo que se puede observar es el uso de Blockchain para temas de protección de privacidad, autenticación de identidad, agregación de datos y costos de electricidad para la recolección de datos en smart grids.

En este primer trabajo de investigación se encuentra recopilado una cantidad de soluciones donde se implementa el uso de blockchain para el tema previamente mencionado, entre ellos se encuentran: asuntos de seguridad en IoT, transmisión de energía mediante P2P, problemas de seguridad en smart grids, protección de smart grids, sistemas en microgrid, entre otros (Cao et al., [2023](#)). De aquí podemos mencionar algunos cuantos para darnos una idea.

Como se mencionó previamente y también como parte de lo que se busca de la solución del reto, el sistema debe de ser capaz de poder autenticar la identidad del smart meter para corroborar que efectivamente la información que nos está siendo enviada es de parte de alguien autenticado, ahora, ellos hablan de diferentes tipos de autenticación dado que se ven varios aspectos como el de vehículos eléctricos, sistemas V2G (Vehicle to grid), entonces se busca algo que sea más cercano a nuestro tópico.

Hay mención de una solución de la autenticación para intercambio de energía basado en Blockchain con computación frontera, el sistema utiliza firmas digitales marcas de tiempo para establecer autenticación de identidad, además, como está basado en computación frontera (Edge computing), los participantes del blockchain son tratados como nodos el mecanismo de consenso “Proof of Stake” (PoS) son utilizados para seleccionar los nodos contados (Cao et al., [2023](#)).

En esta primera referencia también se menciona como se puede enfrentar los problemas de la comunicación de entidades en smart grids basados en Edge computing, se diseña una autenticación de identidad de dos vías y un acuerdo de protocolo de llave, en este sistema solo los servidores frontera y las autoridades registradas (AR) actúan como nodos del blockchain, lo cual hace imposible a los servidores frontera el

asociarse con identidades anónimas al usuario. El protocolo establece la seguridad de la autenticación basado en las asunción de curvas elípticas discretas logarítmicas (ECDL) y las curvas elípticas computacionales diffie hellman (ECCDH), se establece la seguridad del proceso de negociación de la llave basado en ECCDH y adopta contratos inteligentes para habilitar al AR para monitorear la identidad de malos usuarios basándose en la clave pública que posee para obtener el anonimato condicional. En adición, el sistema permite actualizar o eliminar las llaves privadas de usuarios (Cao et al., 2023).

2.2. Uso de AES para encriptación de micro-grids

Pasando a otro trabajo encontrado, se hallaron soluciones para la encriptación de sistemas remotos para microgrids utilizando el cifrado AES en un arduino ESP32 con módulo LoRa.

Un concepto general para los procesos de smartgrids y redes micro-smartgrids bajo los sistemas SCADA (Supervisory Control and Data Acquisition) es la falta de sistemas de computación que tengan el mínimo consumo, sean seguros y a su vez cuenten con autenticidad, entonces se optó por el sistema de cifrado AES sin el involucramiento de terceras partes. En particular nos estaremos concentrando en el arduino ESP32 LoRa dado que el otro objeto de prueba no fue capaz de soportar una sola ronda del método de encriptación (Iqbal y Iqbal, 2018).

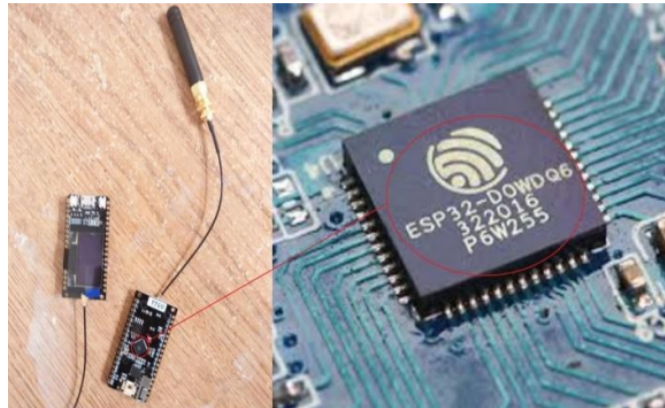


Figura 3: Arduino ESP32 con LoRa para encriptación AES, (Iqbal y Iqbal, 2018)

Para la implementación, MAC fue añadida para autenticar la comunicación. Antes de enviar cualquier mensaje, primero se encripta mediante AES y luego se crea un MAC único de 64 bits a partir del plaintext y luego el ciphertext y MAC, ambas cadenas se concatenan y se envían, de manera similar, la entidad que recibe primero separa la MAC y el ciphertext, la MAC es verificada y después el texto es descifrado para continuar. (Iqbal y Iqbal, 2018).

El resultado de este trabajo fue la incorporación de un arduino de un costo aproximado de 40 dólares con una potencia de alrededor 5mW capaz de soportar la implementación del AES acompañado también de la MAC, teniendo así una implementación a sistemas SCADA para poder enviar información de manera

remota. Si bien el resultado no es de mucha utilidad para nuestro caso dado que no utilizan el sistema que nosotros buscamos implementar (PKI), aún así de aquí podemos observar cómo es posible hacer encriptación y envío de sistemas en grids con materiales de relativo bajo costo como se muestra en la correspondiente a esta referencia.

2.3. Comunicación de medidores inteligentes con red AMI

Para este trabajo se tiene una implementación de seguridad en smart grids con el uso de PKI y que también a su vez incrementan la robustez de la seguridad mediante el uso de un VPN.

Además de los lineamientos de los protocolos que ellos siguieron para su solución, a su vez pudieron identificar los lineamientos de ciberseguridad en redes inteligentes que deberían tener (Sáenz Leguizamón, [s.f.](#)).

- Identificación y autenticación de usuario
- Aislamiento de las funciones de seguridad
- Protección ante negación del servicio (Denial of Service (DoS))
- Protección perimetral
- Integridad de las comunicaciones
- Confidencialidad de las comunicaciones
- Confidencialidad de la información en reposo

En cuanto a la arquitectura de su red se basa en acceder a una red AMI (Advanced Metering Infrastructure) por enlaces inalámbricos mediante el uso de Wi-Fi, lo cual implica el uso de un protocolo de seguridad Wi-Fi. En este caso se utilizara el protocolo de WPA2 (Wi-Fi Protected Access) por medio de una llave previamente compartida (PSK), en cuanto al cifrado se recurrió al método de AES, de manera general se tiene la siguiente figura de como sería su estructura.

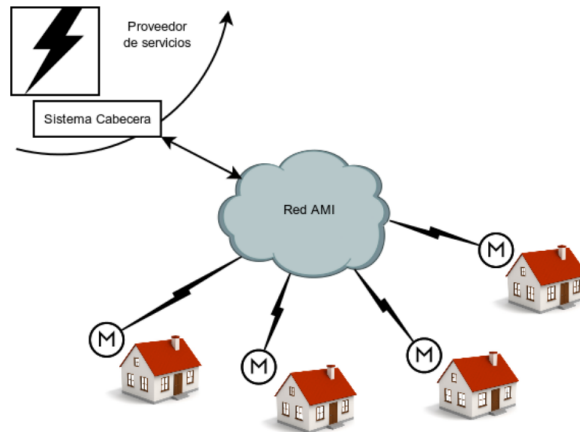


Figura 4: Arquitectura propuesta de la red AMI, (Sáenz Leguizamón, [s.f.](#))

Ahora, para poder garantizar la seguridad perimetral, autenticación y correcta identificación de usuarios decidieron el uso de VPN (Virtual Private Network). En su caso particular decidieron utilizar herramientas de uso libre tales como OpenVPN.

Durante el resto del documento se describe la implementación del uso y pruebas del VPN para la mejora de la privacidad de los datos, si bien esto sale un poco de nuestra área de trabajo, este ejemplo sirve como ideas que se puede considerar como trabajos futuros.

2.4. Criptosistemas PKI con curvas elípticas

Un aspecto importante de la resolución de este trabajo es la garantía de la integridad de los datos, por lo cual nosotros debemos indicar al auditor algún método para poder hacer una firma digital, en este caso se observa el uso de curvas elípticas en criptosistemas PKI.

El objetivo general de este trabajo fue el desarrollo de un sistema de cifrado PKI mediante el uso de AES, ECDSA y Firmas digitales SHA-1 para comunicaciones en una red tipo jerárquica y no privada. Para esto parten asumiendo que se tiene una infraestructura de hardware como la siguiente (Mellado Ramírez, [s.f.](#)).

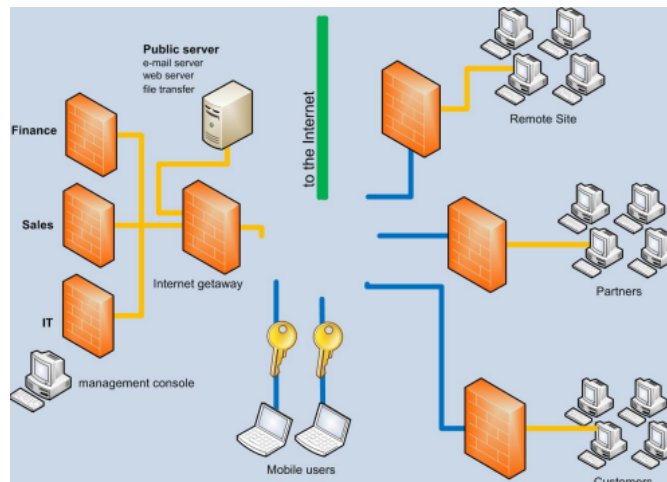


Figura 5: Infraestructura física de la PKI, (Mellado Ramírez, [s.f.](#))

En el desarrollo de su PKI se codifica el sistema en AES para cifrar los mensajes, la firma digital se codifica mediante SHA-1 y ECDSA (Eliptic Curve Digital Signature Algorithm) para el cifrado de la llave y la firma digital.

En la sección de resultados se discute que una de las limitaciones de esta propuesta es el tiempo de computación y la capacidad de cómputo para manejar las operaciones de las curvas elípticas, lo que hay que tomar en cuenta es que este trabajo tiene aproximadamente diez años y el procesador fue un Core i3 con 4 Gb de RAM, por lo cual podemos entonces no tener que preocuparnos en este ámbito ya que a día actual el poder computacional es exponencialmente mejor al que se tenía en ese tiempo, lo cual nos permite no descartar el uso de ECDSA.

2.5. Infraestructura PKI para smart grids para el uso de blockchain

En el siguiente trabajo volvemos a observar un trabajo que utiliza el sistema de Blockchain en smart grids, pero en esta ocasión utilizado para cuestiones de PKI.

El esquema que este trabajo plantea simplifica los certificados digitales (creación, verificación, revocación) y también reduce la dependencia de terceras personas que las PKI suelen tener, en la figura siguiente podemos observar la arquitectura propuesta por ellos.

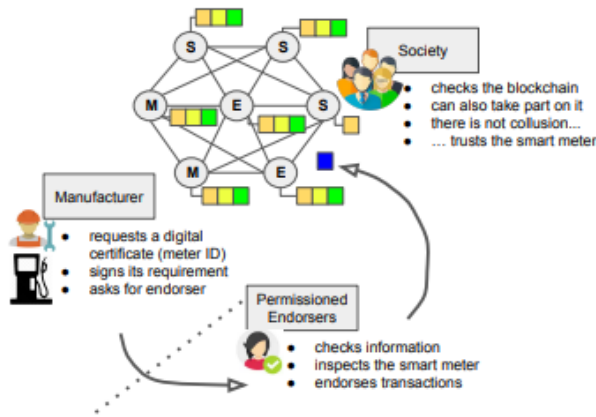


Figura 6: Estructura PKI basada en blockchain para smart meters, (Melo et al., 2020)

Para la implementación se utilizó Hyperledger Fabric, junto con un programa realizado en Python para la simulación de los smart meters, donde cada uno poseía una clave asimétrica. El medidor también sabe como exportar su clave pública.

Cada vez que creada una instancia, se genera la petición de la creación de su clave pública y se sube la transacción al blockchain, todos los peers en la cadena de blockchain replican y validan la información, si la información es consistente, el blockchain nuevo se agrega un activo digital al medidor con su clave pública respectiva, una vez que se encuentra ahí, cualquier persona con acceso la puede invocar y verificar la firma digital.

En cuanto a la firma digital, cuya función en las aplicaciones de criptografía de clave pública es asegurar la integridad, autenticidad y no repudio de alguna pieza de información, su implementación depende del par de llaves asimétricas, la entidad que envía calcula la información del hash y encripta usando su llave privada, de manera complementaria cualquier entidad puede verificar la firma digital usando la clave pública para descryptar el hash y revisar la correspondencia de la pieza de información original (Melo et al., 2020).

En este caso de estudio se escogió como llave asimétrica ECDSA el cual crea un activo digital del blockchain, dicha curva puede ser implementada mediante la librería de python. En cuanto al hardware utilizado para este experimento, se contó con un servidor que corría Hyperledger Fabric 1.4 LTS, un Intel Server S2600Cw con 256GB RAM y dos Xeon E5-2650 v4 CPUs con una velocidad de 2.2GHz.

2.6. Plataformas IoT

En este siguiente trabajo también se trabajó con una solución basada en blockchain para smart grids, aunque en esta ocasión se presta atención específica a las soluciones para IoT, estas se basan en varios dispositivos comúnmente denominados nodos los cuales cuentan con componentes básicos de un ordenador tales como son el CPU, RAM, memoria Flash, sensores, actuadores, entre otros (Hernández Camero,

2019). De aquí podemos generar un pequeño listado de los dispositivos propuestos, los cuales seremos capaces de tomar como referencia, también en caso de querer observar las especificaciones de cada uno de estos, se encuentran de manera detallada en la referencia.

- Adafruit Feather 32u4 Bluefruit LE
- Arduino/Genuino MKR1000
- BeagleBone Black
- NodeMCU ESP-32S
- NodeMCU v3
- Raspberry Pi 3 Modelo B+
- Terasic DE10-Nano kit
- WiPy 3.0

2.7. Comparación y relación de búsqueda

Con toda la información recopilada, podemos entonces resumir la información recabada hasta el momento, junto con el o los conceptos clave de nuestro interés para nuestra solución.

Referencia	Sección estado del arte	Aporte
(Cao et al., 2023)	2.1	IoT, EC
(Iqbal y Iqbal, 2018)	2.2	Arduino ESP32, AES
(Sáenz Leguizamón,s.f.)	2.3	PKI,Wi-Fi
(Mellado Ramírez,s.f.)	2.4	EC, PKI, SHA
(Melo et al., 2020)	2.5	PKI, EC
(Hernández Camero,2019)	2.6	IoT, Raspbery Pi, Arduino

Cuadro 1: Resumen del estado del arte.

Con esta tabla nos podemos dar una idea de cómo la información recopilada nos puede aportar como sustento a la propuesta a realizar, ya que se tienen ejemplificaciones de casos previos en el que se usan los conceptos, hardware, software, métodos de encriptación y firmas en casos similares a lo que nosotros buscamos.

3. Propuesta

Una vez definido el estado del arte y habiendo contemplado las distintas soluciones similares en otros ámbitos por diferentes entidades, entonces se puede realizar una propuesta para nuestro caso particular, utilizando las ideas de otros autores para generar así la solución que se está buscando.

Para esto se tendrá que definir qué tipo de infraestructura se va a utilizar, como realizaremos la firma de los datos para validar la integridad de los mismos realizado por el auditor, el cifrado de los datos para poder enviarlos por internet, proponer que piezas de hardware utilizaremos para la solución y mostrar cómo configurarlo, o en su defecto proponer algún tipo de emulador que pueda obtener el mismo resultado en dado caso que se tengan recursos limitados, y finalmente mostrar cómo es que se puede encriptar los datos ya verificados para poder ser almacenados de manera segura en la base de datos.

Aquí tenemos un bosquejo inicial de como es que se buscaría que fuera la infraestructura con la que se piensa trabajar.

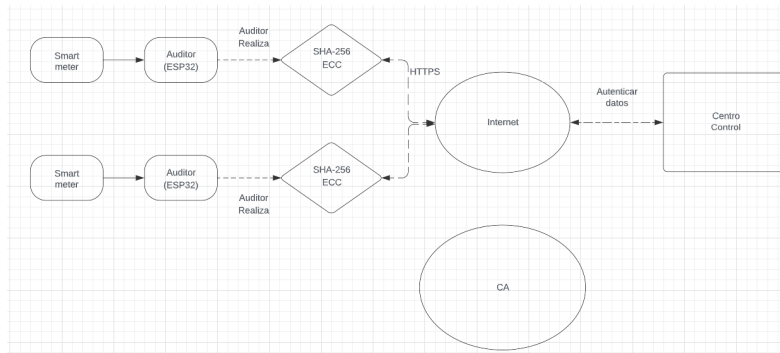


Figura 7: Bosquejo de arquitectura inicial

Una explicación un tanto simple y resumida de como es que funciona la arquitectura propuesta es que los auditores reciben la información de los smart meters, estos aplican un cifrado y firmado mediante ECDSA, para luego ser enviados vía internet hacia el centro de control, donde el mismo valida la información recibida y procede a guardarla dentro de su base de datos de manera segura. Esto es de manera genera a lo que se espera a llegar eventualmente, pero debido a ciertas limitaciones presentes en el desarrollo, algunos puntos tendrán variaciones, pero la esencia del proceso se mantendrá igual.

En la siguiente tabla podemos ver un pequeño resumen de los componentes de nuestra propuesta.

	Funciones	
Parte de la arquitectura	Arranque	Producción
Auditores (Arduino ESP32)	<ul style="list-style-type: none"> -Petición de registro a centro de control -Confirmación de alta por centro 	<ul style="list-style-type: none"> -Recibir datos -Formatear datos -Firma de datos -Encriptación datos -Envío de datos -Recibir actualizaciones -Verificar actualizaciones -Instalar actualizaciones
Centro de control (Flask)	<ul style="list-style-type: none"> -Recibir petición de registros de auditores -Registrar auditores con usuario y "llave" -Levanta servidor mediante HTTPS 	<ul style="list-style-type: none"> -Recibir y desencriptar datos -Verificar firmas -Verificar usuarios registrados -Enviar actualizaciones -Resguardar datos encriptados
Smart meters (datos csv)	<ul style="list-style-type: none"> -Registro de datos 	<ul style="list-style-type: none"> -Envío de datos al auditor

Cuadro 2: Resumen de la propuesta.

De manera general se observan los componentes principales de la arquitectura, junto con la parte que representan y la función a realizar dependiendo si es en el arranque o durante su producción, en las siguientes secciones se habla un poco más a detalle del como funcionan y detalles relacionados de las componentes mencionadas.

3.1. Implementación de los esquemas de clave pública

Debido a la naturaleza del reto, se utilizará una infraestructura de clave pública (PKI), entonces es pertinente el primero entender que es una PKI, su funcionamiento, protocolos, y todo lo necesario para poder realizar su implementación.

Una infraestructura de claves públicas (PKI) es un sistema el cual se dedica al cifrado de claves públicas para autenticar las diferentes fases del proceso de intercambio de información en la web. Esta tecnología funciona mediante certificados, usando estos certificados se implementa la CA o Autoridad de Certificación. Esta tecnología valida la información y la firma de una manera digital. Este proceso asegura que los certificados no puedan ser alterados. Los dispositivos involucrados en este intercambio de información reciben los certificados encriptados y usan la clave pública para desencriptarlos. (Public, 2021)

Las claves públicas utilizan criptografía asimétrica, estas son responsables de todo el intercambio seguro de información a través del internet. Comenzando desde el principio, la encriptación de datos, se refiere a

tomar un texto y hacerlo indescifrable sin el uso de algún tipo de llave secreta. Esto se consigue utilizando certificados digitales, los cuales son emitidos por organizaciones específicas y determinadas. En general las llaves públicas sirven para 3 cosas: Autenticar a personas que no conoces, proteger la integridad de tu información y encriptar información que quieras enviar a otro lado.

Hoy en día hay 3 principios matemáticos que se usan para generar llaves públicas y privadas, RSA, ECC, y Diffie-Hellman. Aunque estos algoritmos se basan en el mismo principio de generar 2 números primos los cuales tengan 1024 bits de longitud, luego se multiplican y el resultante es la llave pública. Mientras que los números primos son la llave privada. Haciéndolo de esta manera sería casi imposible adivinar la llave privada desde la pública. (Public, 2021)

Las llaves públicas funcionan de la siguiente manera. Supongamos que hay 2 personas que quieren intercambiar información de una manera privada. Cada persona tiene una llave pública y una llave privada, las cuales están matemáticamente relacionadas. La persona 1 utiliza la llave pública de la segunda persona (a la cual todos tienen acceso) para encriptar el mensaje deseado y se lo manda de regreso a la persona 2. Esta persona utiliza su llave privada para poder descifrar el mensaje. De esta manera aunque alguien intercepte el mensaje no puede descifrarlo sin esta llave privada a la que solo tiene acceso el propietario. Este proceso que involucra 2 llaves se le conoce como encriptación asimétrica. Es la evolución de la encriptación simétrica en la cual solo había 1 llave para cifrar y descifrar un mensaje. Este método era muy riesgoso porque había que hacerle llegar la llave al recipiente del mensaje sin que se perdiera su integridad.

Desafortunadamente esta no es toda la solución, ya que alguien pudiera imitar a la persona 2 y obtener la misma información de la persona 1. De aquí salen los certificados digitales, los cuales se encargan de verificar la identidad de la persona propietaria de la llave pública. Estos certificados solo se pueden emitir por organizaciones designadas por país, para que no cualquier persona pueda obtener uno. Replantando el escenario original de 2 personas intercambiando información. Ahora la persona 1 puede pedir el certificado digital de la persona 2, el cual está ligado a su llave de clave pública, con esto puede asegurar de que al mandar el mensaje lo está mandando a la persona deseada. Realmente todo este proceso es para poder asignar una identidad a cada una de las llaves públicas particulares de las personas.

Otra manera de llamar a estos certificados de autenticación es X.509. Estos certificados esencialmente son pasaportes, contienen la información de un individuo o identidad. Aparte son resistentes a modificaciones y contienen información que puede probar su autenticidad, también pueden ser rastreados al usuario e incluso cuentan con fecha de expiración. Los certificados se usan comúnmente con SSL/TLS para asegurarse que el browser sea seguro. Durante una conexión de este tipo el servidor autentifica de acuerdo con el handshake establecido. En este momento el servidor presenta un documento X.509 firmado por el cliente. (X.509, 2021)

Cada sistema operativo tiene una lista de autoridades de certificación válidas. Por ejemplo, Firefox tiene 150 autoridades que acepta como válidas para la emisión de estos certificados. Los certificado X.509 tienen

la siguiente estructura:

- versión: el número de versión del certificado, por ejemplo, la versión 3 del certificado tiene más parámetros
- numero de serie: número único que es creado para ese certificado por la organización que lo emitió
- firma: el algoritmo que se usa para generar la firma
- remitente: el nombre distinguido de la organización que lo emite
- validez: 2 fechas, la del tiempo de emisión y la de expiración
- sujeto: para quien se está emitiendo el certificado es decir el receptor
- llave: la llave pública y el algoritmo que se usó Ej, Diffie-Hellman

El proceso para poder crear un certificado es el siguiente. Primero se crea la llave privada y pública de una persona en particular. Luego la organización emisora le pide los datos personales a la persona que quiere el certificado. Luego sigue programar esta información a el certificado, es decir llenan los datos que se mencionaron anteriormente. Este "documento" se firma por el propietario para comprobar que realmente es él y lo acepta. Por último la organización valida todo el proceso y firma con su propia llave privada. Con todas estas medidas de seguridad y protocolos, hacen que sea casi imposible falsificar un certificado de este tipo. (X.509, [2021](#))

Una opción que tenemos para implementar PKI en Python es la librería CryptoSys. Es de fácil instalación y el único requisito para poder usarla es tener la versión 3 de Python instalada. Esta librería sirve para poder encriptar información desde "strings" de Python hasta archivos de texto completo. Cuenta con diversas funcionalidades. También puedes elegir si encriptar la información utilizando SHA-1 hasta SHA-256. Un ejemplo que se muestra en la documentación de la librería es como utilizan AES-128 para encriptar un archivo de texto, el resultado de esta operación es base64-encoding de IV+CT. Esta librería también ofrece la funcionalidad de descryptar dichos archivos. Tomando todo esto en cuenta, parece ser una buena opción con mucho potencial a considerar para todo el proyecto.

3.2. Método de firma digital

Para la cuestión del método de firma que garantice la autenticación de nuestros datos previos a ser encriptados y enviados, se ha decidido realizar mediante el método de curvas elípticas.

Estas son referenciadas en el estado del arte en las secciones 2.1, 2.4 y 2.5, que al ver los beneficios que trae el reducir el tamaño de la clave mientras que a su vez mantiene la robustez de la seguridad deseada y que además cuenta con librerías en python para su implementación, nos parece adecuado el utilizarlo como el método de firma para nuestra propuesta, no solamente basta la firma mediante curvas elípticas para esta parte, también se requiere utilizar algún tipo de hasheo para poder hacer una implementación completa, por lo cual se optara por utilizar SHA-256 que si bien puede ser un tanto más lento en comparación con

SHA-1 pero con esto podemos garantizar una mejor seguridad debido a que se regresa una cadena de 256 bits, y debido a que se cuenta con buenos recursos computacionales, el tiempo extra de procesamiento no será un mayor inconveniente.

En la siguiente fase se dará una explicación más detallada sobre el funcionamiento de las partes de esta sección.

3.3. Cifrado de los datos

Para poder registrar a los auditores con el centro de control, se debe de hacer una solicitud de registro, dicha solicitud tiene que viajar por la red para hacerlo y entonces debemos de asegurar una manera de poder mantenerlas seguras durante el proceso, lo que se propone entonces es hacer el encriptado y envío en línea del auditor al centro de control vía internet, para eso se propone el protocolo HTTPS, el cual podemos proceder a explicar.

En el esquema sugerido para la solución planteamos la idea de manejar un intercambio de información de los sensores al centro de control por medio del uso de internet (WiFi). Teniendo en mente esto es importante hablar sobre cómo funciona el cifrado de los datos a través de redes WiFi.

El protocolo utilizado generalmente para cifrar el intercambio de información a través de redes WiFi es el HTTPS. Que este en sí tiene dos tipos: HTTP sobre TLS y HTTP sobre SSL.

Cuando el internet comenzó Tim Berners-Lee se planteó la idea de diseñar un protocolo que se pudiera utilizar fácilmente para 5 objetivos: transferir archivos, que tuviera la habilidad de buscar un índice en un banco de hipertexto, que hubiera una negociación de formato y la que tuviera la habilidad de referir a otro servidor. Después de desarrollar esta idea se creó el HTTP (Hyper Text Transfer Protocol). (Ilya, 2017)

Una manera fácil de entender HTTP es así. HTTP es como una lista de instrucciones que necesita un servidor y un cliente para poder comunicarse de manera confiable. Un cliente le pregunta a un servidor siguiendo las instrucciones y el servidor entonces contesta siguiendo estas mismas. El beneficio de esto es que al saber que los dos se adhieren a este protocolo o instrucciones se sabe que la información va a mantener su integridad y el que intercambio va a ser confiable.

Ahora vamos a tocar el tema de HTTPS que es una versión segura de HTTP. Cuando tocamos el tema de seguridad ahora es importante mencionar que es diferente al tema de que sea confiable que se tocó en el párrafo pasado. Que sea confiable significa que el protocolo tenga la capacidad de mover la información sin que el usuario tema que se deteriore la información en el traslado pero ahora vamos a hablar de proteger la información tanto de ataques para deformar la información o de personas que quieren interceptarla. Como mencionamos al principio existen dos tipos HTTP sobre TLS y HTTP sobre SSL, TLS es la versión nueva de SSL y son muy similares, pero qué significa esto. TLS o SSL son capas de protección que cifran los datos al ser enviados por el internet. El servidor hace un intercambio de información inicial

donde se verifican su autenticidad y seguridad que se llama "handshake" donde al confirmarse se inicia el intercambio de información, esto se tocó brevemente en la sección 3.1.

Ahora vamos a describir paso por cómo sería una conexión HTTPS entre cliente y servidor para hacer más claros los conceptos (Gourley et al., 2002):

1. El cliente se conecta al servidor.
2. Se hace el "handshake" de TLS/SSL.
3. El cliente manda la información cifrada y viaja con el protocolo HTTP
4. El servidor recibe la información con el protocolo HTTP y descifra la información.
5. Se cierra la conexión cifrada TLS/SSL entre el servidor y cliente.
6. Se cierra la conexión entre el servidor y el cliente.

3.4. Hardware propuesto y forma de implementación

Para hacer la elección del hardware que será utilizado, se tuvo que hacer un análisis extenso. Las dos principales opciones que teníamos en consideración eran Arduino y Raspberry PI. Para poder utilizar alguno de estos hardwares teníamos que verificar que se pudiera programar en Python al utilizarlos. Se hizo una investigación, donde se encontró que en Arduino se necesita descargar una librería para así poder hacer toda la programación en Python. Esta implementación se llama MicroPython y se puede utilizar mediante la instalación en el hardware. Mientras tanto, en Raspberry PI, se utiliza Python como el lenguaje de programación base. Por lo tanto, no se tiene que agregar de manera manual y ya está adaptado para su utilización.

Después de verificar la utilización de Python en estos hardwares, se tiene que hacer una comparación entre ellos para ver cual sería más útil para llevar a cabo nuestro proyecto. Por ende, se tuvo que investigar acerca de las ventajas y desventajas de utilizar cada uno de estos hardwares. Con esto, pudimos encontrar que Raspberry PI es más eficiente dado a que tiene ya su propio sistema operativo instalado. De igual manera, Raspberry PI está basado alrededor de microprocesadores que ya están conectados a internet a diferencia de Arduino que tiene micro-controles que no están previamente conectados a internet. Sin embargo, Arduino es más accesible dado a que es más barato y si se llegara a querer aprender más acerca de hardwares se recomienda utilizarlo.

Para la elección en torno a nuestra propuesta, es requerido primero consultar el material disponible y contrastarlo con lo encontrado en el estado del arte, antes podemos mostrar una tabla que nos resuma las piezas que consideramos como candidatas a utilizar junto con algunas características.

Con esta tabla de las opciones de hardware candidatos a utilizar podemos tomar una decisión para nuestra implementación. Se llega entonces a la decisión de utilizar ESP32, los motivos son que si bien Raspberry tiene mucha potencia y buenas especificaciones, el precio es bastante elevado para la índole

Referencia	Dispositivo	SRAM	Costo	Software compatible
(Teja, 2021)	ESP32	520 KB	200-500 MXN Varía por marca	Arduino IDE MicroPython JavaScript
(RaspberryPi, s.f.)	Raspberry Pi B	1 GB	3,500 MXN Aprox.	Raspeberry OS Linux Python
(make-it, s.f.)	NodeMCU v3	64 KB	60- 100 MXN Varía por marca	Arduino IDE Mycropython

Cuadro 3: Tabla resumen de hardwares candidatos.

de respuesta que estamos proponiendo, y en cuanto a NodeMCU v3, este tiene un precio asequible, pero su poder de cómputo es de menor magnitud, y debido a las operaciones y transacciones que estaremos realizando, ESP32 cuenta con un precio considerablemente proporcional junto a las capacidades que este puede realizar, lo bueno de nuestra elección es que se pudo confirmar que se encuentra presente en el laboratorio.

3.5. Centro de control

Flask es un framework web de Python que se utiliza para construir aplicaciones web. Es una herramienta flexible que facilita la creación de aplicaciones web y API. Flask está basado en el patrón de diseño Modelo Vista Controlador, lo que significa que separa la lógica de la aplicación en tres partes principales: datos, presentación y la lógica de la aplicación. (Vyshnavi y Malik, 2019)

Flask funciona mediante la creación de una aplicación de Flask, que se define en un archivo Python. La aplicación de Flask define las rutas de URL para la aplicación web y define cómo se deben procesar las solicitudes HTTP que se reciben en cada ruta.

Cuando se recibe una solicitud HTTP en una ruta URL determinada, Flask llama a la función asignada a esa ruta y procesa la respuesta para enviarla al cliente. En resumen, Flask es un framework de Python que hace más fácil el desarrollo de aplicaciones web y API a través de la definición de rutas URL y la separación de la lógica de la aplicación en modelos, vistas y controladores. (Vyshnavi y Malik, 2019)

Debido a que se tiene experiencia utilizando esta librería. Por lo que la implantación puede ser un poco más sencilla. Para poder montar un centro de control en Flask va ser necesaria una base de datos. Para esta función podemos utilizar SQLAlchemy para poder crear una base de datos relacional Sqlite. Esta base funcionaria como nuestro sistema central al que los auditores estarían reportando. Es decir cada auditor estaría recopilando información del medidor, la cual luego encarpetaría con una clave pública y enviaríamos la base de datos localizado en el sistema centralizado para guardar esta información. No

habría necesidad de que los datos viajen mediante internet ya que se pueden guardar directamente en la base de datos centralizada desde cada auditor individualmente. Es decir que todo estaría ocurriendo en la misma red local del hogar de la persona.

Tendríamos una interfaz personalizada para que las personas a cargo de la operación puedan acceder a los datos. Incluso se podrían crear gráficos y más interacciones que nos pidan. Tendría la misma funcionalidad de administrador típica de otras aplicaciones. Cabe mencionar que no habría necesidad de generar una interfaz para el usuario, ya que él no necesita ver las métricas de consumo de energía, sin embargo, en caso de ser necesario también se podría habilitar esta opción de generar un interfaz para el usuario. Simplemente se conectarán a la página y verán sus grafos personalizados, y no tendrán acceso a todos los comandos y funciones de administrador.

Lo bueno de utilizar una implementación con Flask, es que nos da mucha libertad de personalización y expansión de la aplicación. Realmente tendríamos muy pocos límites para el desarrollo del proyecto, solamente tendríamos que definir ciertas metas que sean alcanzables en el tiempo que tenemos para desarrollarlo.

Para que un centro de control Flask funcione de manera adecuada es necesario cumplir con ciertos requisitos. (Flask, 2022)

1. Tener instalada la versión de python 3.7 o la mas reciente ya que tienen mayor compatibilidad.
2. Instalar los controladores de base de datos correspondientes para la base de datos que se va a utilizar, en este caso se va a utilizar una base de datos MySQL, por lo que se instalará el controlador MySQL para Python .
3. Es recomendable crear un entorno virtual para el proyecto. Ya que permite aislar las dependencias del proyecto de las instaladas en el sistema.
4. Se necesita un editor de código como Visual Studio Code, Atom, PyCharm, entre otros. Para escribir y modificar el código del proyecto Flask.
5. Tener conocimientos de seguridad web para asegurar que el centro de control Flask sea seguro y esté protegido contra ataques malintencionados.
6. Tener conocimientos básicos de python y conocimiento de HTML, CSS y Javascript.

4. Implementación

En esta sección se explicará de manera más detallada lo mencionado en la propuesta, para posteriormente implementarlo en conjunto, antes de eso se presenta la nueva arquitectura básica de la solución, donde en las siguientes partes del documento se explica más el funcionamiento.

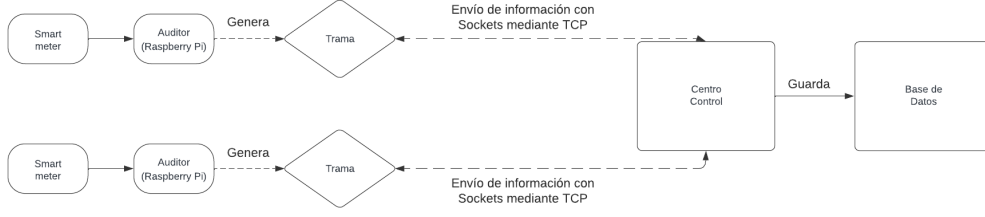


Figura 8: Arquitectura actualizada de la propuesta

4.1. ECDSA

Para el desarrollo del método de firma digital con curvas elípticas primero se tienen que desarrollar algunos conceptos como las curvas elípticas y las operaciones necesarias para la generación de la curva y sus puntos, posteriormente ya podremos ir revisando los estándares y parámetros propuestos por el National Institute of Standards and Technology (NIST) para la cuestión de la implementación de la solución.

4.1.1. Curvas elípticas

Definamos de manera general a una curva elíptica en un grupo Z_p de la siguiente manera.

Sea $p > 3$ un número primo, una curva elíptica E en el grupo Z_p se define por la ecuación (Johnson et al., 2001).

$$y^2 = x^3 + ax + b \quad (1)$$

Donde $a, b \in Z_p$ y $4a^3 + 27b^2 \not\equiv 0 \pmod{p}$. Entonces el conjunto de números de $E(Z_p)$ consiste de los puntos (x, y) , $x \in Z_p$, $y \in Z_p$ de tal manera que satisfaga la ecuación 1, junto con un punto especial \mathcal{O} llamado punto al infinito.

Para nuestro caso en particular nos interesa el saber el funcionamiento de la suma de los puntos de la curva (Johnson et al., 2001).

Se le conoce a la adición de punto cuando $P = (x_1, y_1) \in E(Z_p)$ y $Q = (x_2, y_2) \in E(Z_p)$, donde $P \neq \pm Q$. Entonces $P + M = (x_3, y_3)$, donde.

$$x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1} \right)^2 - x_1 - x_2, \text{mod}(p) \quad (2)$$

$$y_3 = \left(\frac{y_2 - y_1}{x_2 - x_1} \right) (x_1 - x_3) - y_3, \text{mod}(p) \quad (3)$$

Luego se tiene el punto doble. Sea $P = (x_1, y_1) \in E(Z_p)$ donde $P \neq -P$, entonces $2P = (x_3, y_3)$ se da como (Johnson et al., 2001).

$$x_3 = \left(\frac{3x_1^2 + a}{2y_1} \right)^2 - 2x_1, \text{mod}(p) \quad (4)$$

$$y_3 = \left(\frac{3x_1^2 + a}{2y_1} \right) (x_1 - x_3) - y_1, \text{mod}(p) \quad (5)$$

4.1.2. Prerrequisitos

Antes de poder pasar al proceso general del ECDSA, primero hay que establecer un par de cosas para su funcionamiento. En particular se habla del calculo de la pendiente de la suma de los puntos bajo una curva, normalmente no habría mucho problema con esto pero dado que estamos realizando operaciones bajo un grupo de números de un modulo en particular, entonces las reglas cambian un poco. En este escenario en particular el poder encontrar la pendiente de bajo un modulo p es multiplicando el numerador por el inverso multiplicativo del denominador modulo p de la siguiente manera, empezando por el caso de la suma $P + Q$.

$$\left(\frac{y_2 - y_1}{x_2 - x_1} \right)^2$$

$$((y_2 - y_1)(x_2 - x_1)^{-1})^2$$

Para el caso de $2P$ es bastante similar.

$$\left(\frac{3x_1^2 + a}{2y_1} \right)^2$$

$$((3x_1^2 + a)(2y_1)^{-1})^2$$

De manera que requerimos tener una función que nos ayude a encontrar el inverso multiplicativo de un numero bajo cierto grupo.

Para esto recurrimos a algo conocido como el algoritmo Euclidiano extendido, el nos sirve para encontrar el máximo común divisor de un numero "d" de dos números enteros a,b, y también para encontrar dos números enteros x,y de tal manera que $ax + by = d$. Los pasos del algoritmo son como se presentan.

Algorithm 1 Algoritmo Euclidiano Extendido

Require: $a, b \in \mathbb{Z}, a > b$ **if** $b = 0$ **then** $d = a, x = 1, y = 0$ **return** d, x, y **end if** $x_1 = 0, x_2 = 1, y_1 = 1, y_2 = 0$ **while** $b \neq 0$ **do** $q = \lfloor \frac{a}{b} \rfloor, r = a - qb, x = x_2 - qx_1, y = y_2 - qy_1$ $a = b, b = r, x_2 = x_1, x_1 = x, y_2 = y_1, y_1 = y$ **end while** $d = a, x = x_2, y = y_2$ **return** d, x, y

Esto nos será de ayuda para poder calcular el inverso de algunos números bajo un módulo establecido. También se tienen otras dos funciones que nos ayudan a realizar operaciones de la suma de puntos, una de ellas es el calculo de inverso aditivo bajo un módulo, y la otra es una que funciona como exponencial binaria, si bien esta hace la misma función que la previamente mostrada en Euclidiano, la diferencia radica que en algunas ocasiones esta puede ser más rápida que la forma anterior, se tiene que se encuentra restringida a únicamente ser funcional cuando tu mod n es un número primo. Los detalles de las variables y la documentación se puede encontrar en el repositorio.

4.1.3. Generación de la firma

Para la generación de la firma, se requiere contar con una curva elíptica, pero esta no puede ser una curva aleatoria, tenemos que utilizar alguna curva que se encuentre verificada como segura para el uso del ECDSA.

Siendo concretos, nosotros decidimos utilizar la curva P - 256 para ECDSA de 256 bits en un campo primo, los parámetros de dicha curva en valores hexadecimales se presentan en la siguiente figura, además del siguiente enlace para su consulta (CROCS, 2020).

256-bit prime field Weierstrass curve.
Also known as: **secp256r1** **prime256v1**

Parameters

Name	Value
a	0xfffffffff000000010000000000000000000000000fffffffffffffffbfffff
p	0xfffffffff000000010000000000000000000000000fffffffffffffffbfffffc
b	0x5ac63d8aa39fe7b3ebbd557698bb6c15de0bbc53bf0e63be3c3e27d2604b (0xb617d1fe21c2a247f8bce563da4df277037d812deb33af0a13945d898c296, G 0xf4e342e2fe1a7f9b80ee7eba47cf09ef162bc63576b315cecbb6406837bf5f5)
G	0xfffffffff00000000fffffffffffffffbc6faada7179E84f3B9ca2fc632551
n	0x1

Donde los valores significan lo siguiente (Pornin, 2013).

- Otra cosa que se tiene que mencionar previamente a comenzar con el proceso de firmado, es el como generar la llave pública a utilizar, para esto se siguen los siguientes pasos (Johnson et al., 2001).

- Explicando un tanto mejor el paso dos del procedimiento, lo que se nos esta pidiendo en sí es sumar una cantidad de d -veces el valor de G , el cual como ya vimos son coordenadas generadoras, visto de otra manera el paso dos nos pide:

Bajo circunstancias normales esto no debería de ser mucho problema, pero dada la magnitud de los números que se manejan el hacer esta suma bajo métodos convencionales no es sugerido, por lo cual se recurre a otras opciones, siendo específicos se utiliza un método llamado double and add, el cual utiliza la expresión binaria para poder hacer operaciones de manera eficiente, el funcionamiento del método se encuentra en la siguiente referencia (Corbellini, 2015).

Con esto en consideración, es que podemos ya proceder de manera formal al proceso de generación de la firma, los pasos para poder realizarlo se describen a continuación (Johnson et al., 2001).

1. Generar un número aleatorio o pseudoaleatorio entero k que cumpla que $1 \leq k \leq n - 1$.
2. Calcular $kG = (x_1, y_1)$ y convertir x_1 en un número entero.
3. Calcular $r = x_1 \bmod n$. Si $r = 0$ se regresa al primer paso.
4. Calcular $k^{-1} \bmod n$.
5. Calcular $\text{SHA}(m)$ y convertir el string de bits a un entero e .
6. Calcular $s = k^{-1}(e + dr) \bmod n$. Si $s = 0$ Entonces regresar al paso 1.
7. La firma del mensaje esta dada por (r, s) .

Como se puede observar, el proceso es un tanto directo y conciso con las instrucciones, se menciona que en el repositorio se puede encontrar el código de este proceso junto con documentación misma dentro del código y también ejemplos de los vectores de prueba, dichos ejemplos también serán puestos en este reporte en un par de secciones.

4.1.4. Verificación de la firma

Una vez con la firma generada, es también importante el poder verificar la firma para que este proceso tenga sentido, de manera similar al inciso anterior del reporte, las instrucciones del funcionamiento de la verificación se muestran a continuación (Johnson et al., 2001).

1. Verificar que los números enteros s y r se encuentren dentro del intervalo $1 \leq k \leq n - 1$.
2. Calcular $\text{SHA}(m)$ y convertir el string de bits a un entero e
3. Calcular $w = s^{-1} \bmod n$
4. Calcular $u_1 = ew \bmod n$, y también $u_2 = rw \bmod n$.
5. Calcular $X = u_1G + u_2Q$
6. Si $X = \mathcal{O}$, entonces rechazar la firma. Caso contrario, convertir la coordenada x x_1 de X a un entero, después calcular $v = x_1 \bmod n$.
7. Aceptar la firma solamente si $v = r$.

De manera similar a la sección anterior, detalles de la documentación junto con ejemplos y comentarios puede ser encontrada en el repositorio.

4.1.5. Casos de vectores de prueba

Una vez con el proceso de firmado realizado, ahora tenemos que realizar una prueba para revisar que efectivamente esta funcionando de manera adecuada. Para esto utilizamos vectores de prueba previamente

establecidos de la curva verificada, en nuestro casos son los siguientes (Pornin, 2013).

```
Key pair:
curve: NIST P-256
q = FFFFFFFF00000000FFFFFFFFFFFFFFFFFBC6E6FAADA7179E84F3B9CAC2FC632551
(qlen = 256 bits)
private key:
x = C9FA9D845BA75166B5C21576781D6934E50C3D836E89B127B8A622B120F6721
public key: U = xG
Ux = 60FED4BA255A9D31C961E874C6356D68C049B8923B61FA6CE669622E60F29F86
Uy = 7903FE1008B8BC99A41AE9E95628BC64F2F1B20C2D7E9F5177A3C294D4462299
```

Figura 10: Parámetros de vectores de prueba, (Pornin, 2013)

En nuestro caso se hará una prueba con el texto de "sample" mediante un SHA-256, por lo cual los resultados esperados son los siguientes.

```
With SHA-256, message = "sample":
k = A6E3C57D001ABE90086538398355DDC4B17AA873382B0F24D6129493D8AA0D6
r = EFD48B2AACB6A8FD1140DD9CD45E1D69D2C877B56AAF991C34D0EA84EAF3716
s = F7C81C942D657C41D436C7A18E629F65F3E900DB89AFF4064DC4AB2F843ACDA8
```

Figura 11: Resultado esperado a obtener de vectores de prueba, (Pornin, 2013)

Esto, junto con lo previamente establecido, podemos ahora pasar a realizar una prueba formal dentro de Python para verificar que sea correcto.

[illegible]

Figura 12: Resultados de evaluar el vector de prueba

Entonces, como se puede apreciar, en la parte final se tiene que el resultado es "True", lo cual indica que el proceso de firma y verificación se hicieron de manera correcta, se sabe que se están omitiendo detalles, pero nuevamente mencionamos que estos se pueden encontrar en el repositorio.

Una vez completado esto se concluye como tal el armado del proceso ECDSA, entonces en los siguientes pasos se intentará ahora enviar tramas de medidas hacia el centro de control.

4.2. Tramas

Para poder tratar con los datos que nos fueron proporcionados primero fue requerido el hacer modificaciones a los archivos ya que estos no se encontraban en el mejor formato para poder hacer el proceso de firma y verificación de las lecturas que hacen sus medidores.

Los archivos representan las lecturas de sus medidores en el transcurso de un día en intervalos de 15 minutos, originalmente los archivos csv contaban con más de 96 columnas ya que dichas columnas significan los intervalos de 15 minutos que hay en el lapso de un día, por lo cual primero fue necesario hacer cambios a los datos.

Primeramente, del csv original se crearon dos csv derivados, uno para la cuestión de producción, y el otro para la cuestión de consumo, a la hora de crear los dos nuevos archivos se les agrego el identificador *P* y *C* para producción y consumo de manera respectiva. Posteriormente lo que se hizo fue cambiar el orden de las columnas para que esos intervalos pasen a ser renglones de nuestra tabla de datos y de esta manera poder llevar un mejor registro de los datos, entonces ahora por cada día registrado en las mediciones hechas, ya se cuenta con las 96 observaciones realizadas mostradas de una mejor manera. A continuación se muestra como es que quedaron las tramas.

A	B	C	D	E	
	Identificador	Fecha	Intervalo	Medida	
0	C	02/11/2013	1	58	
1	C	02/11/2013	2	75	
2	C	02/11/2013	3	65	
3	C	02/11/2013	4	0.08	
4	C	02/11/2013	5	67	
5	C	02/11/2013	6	69	
6	C	02/11/2013	7	0.07	
7	C	02/11/2013	8	73	
8	C	02/11/2013	9	68	
9	C	02/11/2013	10	0.06	
10	C	02/11/2013	11	77	
11	C	02/11/2013	12	69	
12	C	02/11/2013	13	91	
13	C	02/11/2013	14	64	
14	C	02/11/2013	15	74	
15	C	02/11/2013	16	74	
16	C	02/11/2013	17	74	
17	C	02/11/2013	18	67	
18	C	02/11/2013	19	61	
19	C	02/11/2013	20	89	
20	C	02/11/2013	21	58	
21	C	02/11/2013	22	75	
22	C	02/11/2013	23	57	

TramasConsumo

Figura 13: Formato actualizado para las tramas de los datos

4.2.1. Prueba de lectura de tramas

Antes de pasar a enviar información mediante nuestro auditor, se decidió hacer una pequeña prueba de que efectivamente nuestro programa de firma y verificación funcione afuera de los vectores de prueba.

Para esto simplemente lo que se hizo fue tomar los archivos csv existentes e ir firmando y verificando cada trama, dicho resultado fue el siguiente.

```
308 file = open("TramasConsumo.csv", 'r')
309
310 text = file.readline()
311 start_time = time.time()
312 while text != "":
313     #start_time = time.time()
314     text = file.readline()
315     ' '.join(text)
316     print(text)
317     time.sleep(15)
318     r,s = firmar_mensaje(text,x)
319     verificar_firma(r,s,text,n,(gx,gy),(Ux,Uy),a)
320     #print("--- %s seconds ---" % (time.time() - start_time))
321     print('\n')
322     print("--- %s seconds ---" % (time.time() - start_time))
323
324
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

9,C,02/11/2013,10,0.06

Verificación de la firma: True

10,C,02/11/2013,11,77.0

Figura 14: Prueba de firmado y verificación de tramas de los datos

De lo observado se tiene que efectivamente es funcional para firmar y verificar las tramas, los siguientes pasos serían dar instrucciones al auditor de solo firmar y enviar el mensaje para que el centro de control pueda recibir tanto la trama como la llave pública del método para poder verificar que la información sea valida, de igual manera se podrá aplicar el caso contrario.

4.3. Creación de certificados

Para esta sección, seguimos los pasos para crear un certificado CA utilizando el servidor Apache, se utilizaron curvas elípticas para generar este certificado con *OpenSSL*. Para empezar este proceso, primero se observaron los estándares definidos de las curvas elípticas, para mantener consistencia con el programa generado para el firmado se utiliza una clave de 256-bits prime256v1. Cabe mencionar que para hacer todo este proceso se utilizarán dos máquinas virtuales, una actuará como el cliente, y otra como el servidor.

Siguiendo con nuestro certificado, se crean estructuras para poder guardar dicho certificado con toda su base de datos, donde dentro de este se hicieron dos estructuras más. Una de estas tendrá la clave privada, y la otra el certificado. De igual manera, se tuvo que obtener un documento de configuración de *openssl*. Luego, creamos la clave privada ECC y generamos el certificado CA con esta clave. Para verificar nuestro certificado utilizamos la clave privada y *openssl*. Con esto, pudimos comparar la clave pública con la privada, para revisar que efectivamente fueran iguales.

Para generar el certificado del servidor, se utilizó una nueva clave privada que se tuvo que guardar en otro directorio. Para hacer esta clave, seguimos los mismos pasos que en la primera vez. Luego, se tuvo que crear un CSR (Certificate Signing Request) para firmar el certificado del servidor. Para esto, tuvimos que asignar nuestra información, la cuál debe de ser la misma que en el certificado CA. Para seguir con nuestro certificado, tuvimos que utilizar la clave privada de CA, el certificado anterior, y el servidor que creamos con la información que asignamos. De igual modo que en el proceso anterior, se tuvo que validar para verificar que todo salió bien. Este proceso se tuvo que repetir pero ahora para el certificado del cliente, tuvimos que generar una clave privada de curvas elípticas. Para esto, verificamos el nombre de la curva que se usa en la clave. Después generamos el CSR (Certificate Signing Request) de nuevo, y con esto crear el certificado del cliente.

A continuación, tuvimos que configurar Apache para validar los certificados de las curvas elípticas. Para lograr esto, tuvimos que instalar los paquetes de Apache. Luego, tuvimos que configurar el servidor, donde se crearon unos directorios para poder guardar los certificados del servidor. Finalmente, tuvimos que realizar dos últimos pasos para concluir nuestro proceso. El primero tuvo que ver con la configuración del nodo cliente para la autenticación. Esto lo hicimos con las claves privadas, el certificado CA, y la autenticación TLS. El otro paso tiene que ver con esta autenticación TLS, dado a que se tuvo que validar con los certificados de curvas elípticas.

4.4. Auditores

Para instalar el Raspberry Pi Desktop se utilizó VirtualBox. Una vez instalado, se descarga la imagen de Raspberry Pi Desktop desde el sitio web oficial. Después de descargar la imagen se abre la VirtualBox y se crea una nueva máquina virtual.

Se asigna la cantidad de RAM que necesitada, luego, creamos un disco duro virtual para la máquina virtual. Se opta por un disco dinámico que crece según sea necesario.

Una vez creada la máquina virtual, accedemos a la configuración en la sección "Almacenamiento" y añadimos el archivo de la imagen como una unidad de CD/DVD en la máquina virtual.

Se inicia la máquina virtual y seguimos las instrucciones de instalación, una vez finalizada la instalación, podremos utilizar la maquina virtual como Auditor.

También a continuación se tiene un diagrama general de como el auditor envía la información.

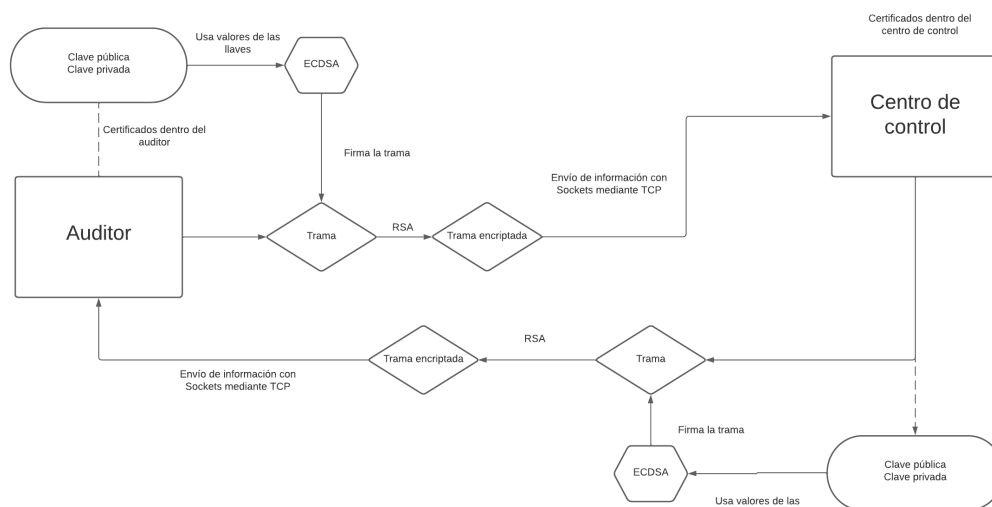


Figura 15: Diagrama del Auditor

4.5. Centro de control

Una parte del centro de control se creo localmente para poder probar antes de montar en la instancia de EC2. El centro de control es ambos el lugar donde se decifran y verifican los datos del auditor como también es una interfaz accesible al cliente para poder visualizar la información. La versión del centro de control que estamos desarrollando tiene varias partes que están como referencia y que en producción se podrían hacer funcionales como la autenticación.

Una vez que accedemos al centro de control localmente se presenta una página de inicio de sesión donde se pide un correo y una contraseña. La contraseña ya esta predeterminada así que es cuestión de poner la contraseña correcta para poder ingresar a la aplicación. Solamente hay una contraseña correcta aunque se pueden añadir más. El correo que se ingresa realmente no es importante ya que este no se checa, simplemente esta ahí el campo por si en un futuro se quisiera añadir una base de datos con una lista de usuarios aprobados.

Una vez que se ingresa la contraseña correcta se genera un numero aleatorio el cual es parte de la url de la página siguiente, es decir de la página que se le muestra al usuario una vez que acceso correctamente. Este número aleatorio se comparó con otro, este mismo dentro de la página de la aplicación, esto se hace para que un usuario que no cuenta con la contraseña no pueda escribir la url de acceso y saltarse tener que poner la contraseña correcta. Es como un tipo de verificación, aunque sea 100 por ciento segura, ya que hay mejores maneras de conseguir lo mismo, es la más sencilla y cumple con su propósito, esta es otra área de mejora para un proyecto más robusto.

En la página de adentro que es la interfaz del cliente, se despliegan una serie de botones que llevan a 3 tipos de gráficas de línea para ilustrar los datos. La primera gráfica es de los datos históricos, aquí se

despliega como una línea del tiempo de los datos recabados de un auditor. Esta pudiera ser en meses o años. La segunda gráfica que se presenta es de los datos históricos pero diarios, así que es como una gráfica pero únicamente de las últimas 24 horas. Al final tenemos un apartado para poder comparar 2 gráficas entrelazándolas. Funciona más como una herramienta visual más precisa que tenerlas simplemente una al lado de otra. Estas son las 3 áreas que consideramos más importantes para *LiCore*, sin embargo en caso de que prefieran otro tipo de visualizaciones también se pudieran hacer.

También se añadió una sección para cambiar el delay con el que se mandan los datos, este vendría siendo un cuarto botón que se encuentra en la columna con los botones de control de interfaz. Cabe mencionar que este dato de delay se encuentra en otra tabla separada. Ya que se encuentran dos tablas declaradas en el código, la tabla de "Crypto" la cual tiene las columnas de identificación, fecha, intervalo, medida, y código identificador. En esta tabla se almacenan todos los datos de LiCore y del auditor. También tenemos la tabla de Tiempo, que tiene la columna identificadora y la columna de dato, que almacena el delay. Es un dato único, así que solamente tiene una fila, pero encontramos que esta sería la manera más fácil de cambiar y consultar el dato.

4.6. Centro de control en la nube

Ahora sigue el proceso de montar nuestro centro de control en la nube, y que se pueda acceder desde cualquier lado. La manera en la que hicimos esto es mediante una instancia de Ubuntu en AWS EC2, utilizando dicha instancia montamos aquí nuestra aplicación que habíamos creado localmente utilizando Flask, Gunicorn y Nginx. (Huiyeon, [s.f.](#)).

Ahora lo que sigue es acceder desde el exterior a nuestra instancia. La manera en la que hicimos esto inicialmente fue directamente con la IP publica pero este método no es muy seguro por lo que intentamos otra cosa. Utilizamos el servicio de Route 53 de AWS junto con un dominio de godaddy para redirigir nuestra página a un nombre propio. Esto es bueno para reconocer el website mas fácilmente pero la verdadera utilidad de esto fue que pudimos certificar nuestro dominio con certbot de la EFF para poder crear una conexión segura por medio de HTTPS y de esta manera garantizar seguridad a nuestro cliente (Certbot, [s.f.](#)).

Ahora es posible acceder a nuestra aplicación desde cualquier computador de manera segura con navegador y usando el dominio licoreproyectotec.com.

En cuanto al intercambio de información entre los auditores y el centro de control en línea, utilizamos sockets en python para intercambiar información con TCP. Para hacer seguro el intercambio de información utilizamos el proceso de curvas elípticas para firmar y verificar certificados y usamos RSA para cifrar la información con llave pública y poder descifrar la información con su llave privada.

4.7. Base de datos

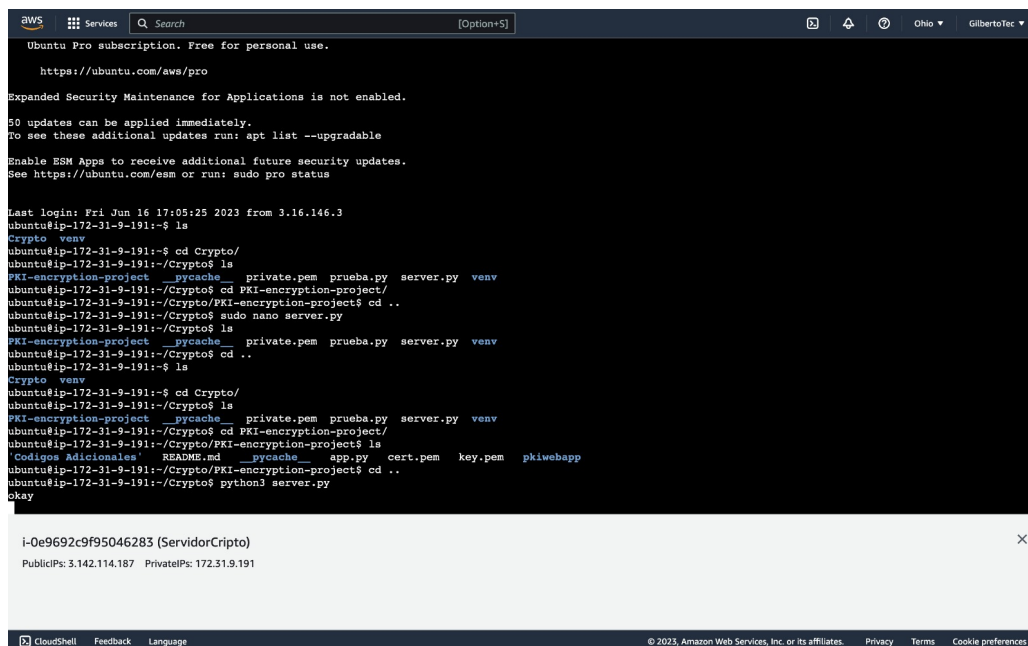
Una vez recibida la información lo que se hace es depositar la información en una base de datos relacional que esta montada también en AWS, se utiliza el servicio de RDS para tenerla en la nube. La ventaja de hacer esto de esto es que podemos acceder a la base de datos de manera separada a la instancia y por lo tanto podemos hacer *queries* y mantenimiento de la base de datos sin tener que conectarnos a la instancia. Otra ventaja importante de usar RDS es que se conecta fácilmente a servicios como EC2 y además tiene opción de cifrado utilizando AES-256 (Amazon, [s.f.](#)).

La base de datos en si contiene 2 tablas. La primera tabla se llama cryptos donde se guardan los registros que mandan los auditores. Esta tabla tiene las columnas id, fecha, intervalo, medida e identificador. La segunda tabla es donde se registra el intervalo de tiempo que va a esperar un auditor para mandar mas información por lo que solo tiene un valor la tabla que es el tiempo.

5. Resultados

A continuación se muestra el proceso detallado del ensamblaje y las pruebas de comunicación hechas, así como los resultados de dichas pruebas, para todos los detalles referentes al código se puede consultar en el repositorio github (Equipo2, [2023](#)).

Comenzamos las pruebas haciendo un intercambio de información entre el Raspberry Pi (auditor) y el centro de control. Esto comienza mandando a llamar el archivo de Python *server.py* en la terminal de la instancia de Ubuntu.



```
aws
Services
Search
[Option+S]
Ohio
GilbertoTec

Ubuntu Pro subscription. Free for personal use.
https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled.
50 updates can be applied immediately.
To see these additional updates run: apt list --upgradable
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

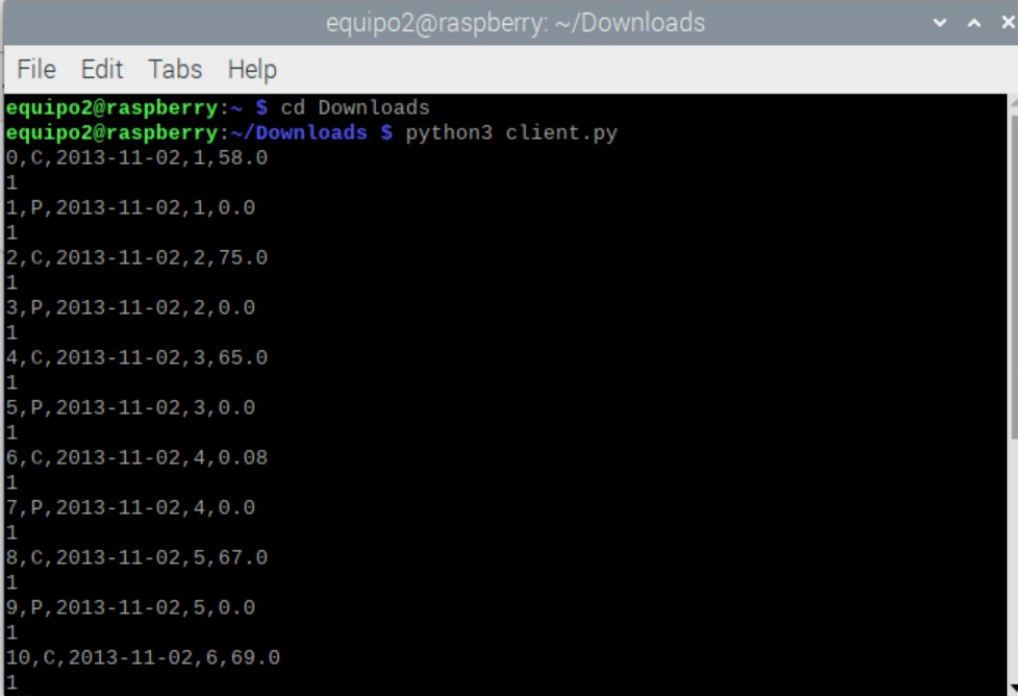
Last login: Fri Jun 16 17:05:25 2023 from 3.16.146.3
ubuntu@ip-172-31-9-191:~$ ls
Crypto  venv
ubuntu@ip-172-31-9-191:~$ cd Crypto/
ubuntu@ip-172-31-9-191:~/Crypto$ ls
PKI-encryption-project  __pycache__  private.pem  prueba.py  server.py  venv
ubuntu@ip-172-31-9-191:~/Crypto$ cd PKI-encryption-project/
ubuntu@ip-172-31-9-191:~/Crypto/PKI-encryption-project$ cd ..
ubuntu@ip-172-31-9-191:~/Crypto$ sudo nano server.py
ubuntu@ip-172-31-9-191:~/Crypto$ ls
PKI-encryption-project  __pycache__  private.pem  prueba.py  server.py  venv
ubuntu@ip-172-31-9-191:~/Crypto$ cd ..
ubuntu@ip-172-31-9-191:~$ ls
Crypto  venv
ubuntu@ip-172-31-9-191:~$ cd Crypto/
ubuntu@ip-172-31-9-191:~/Crypto$ ls
PKI-encryption-project  __pycache__  private.pem  prueba.py  server.py  venv
ubuntu@ip-172-31-9-191:~/Crypto$ cd PKI-encryption-project/
ubuntu@ip-172-31-9-191:~/Crypto/PKI-encryption-project$ ls
'Codigos Adicionales'  README.md  __pycache__  app.py  cert.pem  key.pem  pkiwebapp
ubuntu@ip-172-31-9-191:~/Crypto/PKI-encryption-project$ cd ..
ubuntu@ip-172-31-9-191:~/Crypto$ python3 server.py
okay

i-0e9692c9f95046283 (ServidorCripto)
PublicIPs: 3.142.114.187 PrivateIPs: 172.31.9.191

CloudShell Feedback Language
© 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences
```

Figura 16: Inicio de comunicación por lado del servidor

Ahora después de hacer esto comenzamos la comunicación entre el auditor mandando a llamar en el Raspberry Pi el archivo *client.py* y el servidor ya esta escuchando por conexiones. Cuando se conectan el auditor comienza a mandar los mensajes cifrados y con su certificado. Luego el servidor descifra y verifica la autenticidad de los datos y cuando son auténticos los envía a la base de datos en RDS. Todo esto ocurre corriendo estos dos archivos.



```
equipo2@raspberrypi: ~/Downloads
File Edit Tabs Help
equipo2@raspberrypi:~ $ cd Downloads
equipo2@raspberrypi:~/Downloads $ python3 client.py
0, C, 2013-11-02, 1, 58.0
1
1, P, 2013-11-02, 1, 0.0
1
2, C, 2013-11-02, 2, 75.0
1
3, P, 2013-11-02, 2, 0.0
1
4, C, 2013-11-02, 3, 65.0
1
5, P, 2013-11-02, 3, 0.0
1
6, C, 2013-11-02, 4, 0.08
1
7, P, 2013-11-02, 4, 0.0
1
8, C, 2013-11-02, 5, 67.0
1
9, P, 2013-11-02, 5, 0.0
1
10, C, 2013-11-02, 6, 69.0
1
```

Figura 17: Comunicación por lado del auditor

Ahora nuestra siguiente parte de las pruebas es utilizar la interfaz del centro de control en linea. Dado que ya tenemos montado el servidor y accesible en la web, nuestro primer paso es llamarlo utilizando nuestro dominio licoreproyectotec.com. Como podemos notar nuestra conexión es segura por medio de https y ahora utilizamos nuestras credenciales para ingresar y poder visualizar la información.

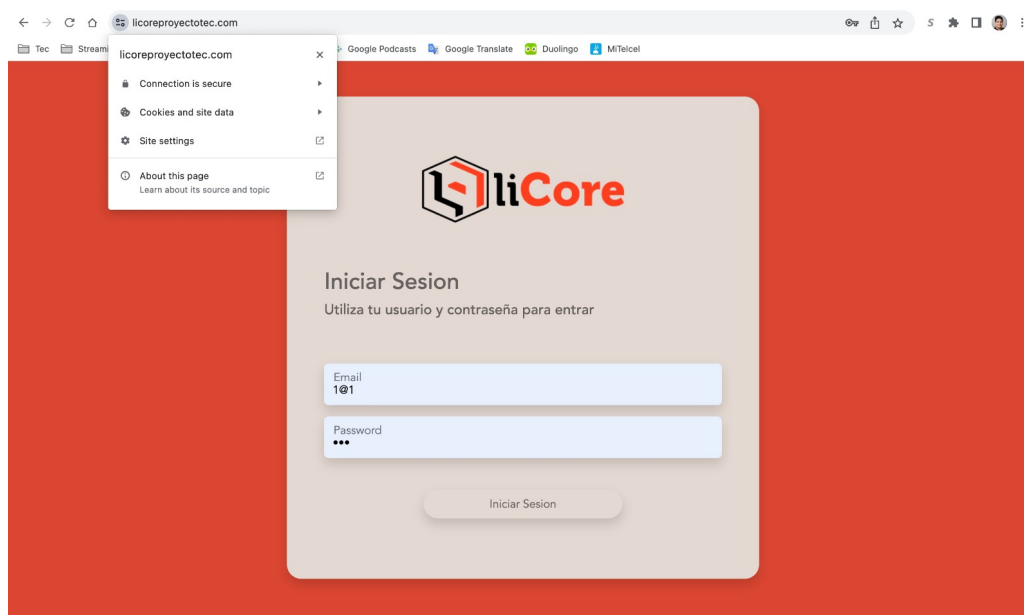


Figura 18: Pagina de inicio del dominio

Ya dentro vamos a probar una de las funciones que utilizan *queries* de la base de datos para visualizar información. Nos dirigimos a la sección de comparación y escogemos dos días diferentes del mismo auditor para comparar el consumo en dos días.



Figura 19: Gráfico de comparación de dos días diferentes

En esta prueba comprobamos que podemos enviar y recibir datos del centro de control y auditor, acceder a la página de manera segura con HTTPS y visualizar la información almacenada en gráficas en el centro

de control.

Así como en la figura 19, podemos visualizar los gráficos históricos y los de gráficos diarios. No incluimos estas imágenes por redundancia, pero en caso de que si quisieran ver en acción, se pueden meter a la pagina (licoreproyectotec.com) y explorarla por si mismos. Aquí también podrán encontrar el último apartado de "configuración", que nos permite cambiar el delay con el que se mandan los datos, aunque sera difícil observar el cambio. Es importante mencionar que los demás gráficos operan de manera muy similar a la imagen incluido, en cuanto al tipo de *query* y como se genera el gráfico para desplegarse, sin embargo, consideramos que el gráfico de comparación es el más completo en cuanto a funcionalidad y diseño.



Figura 20: Gráfico histórico de los datos recibidos

6. Conclusiones

De los resultados obtenidos tenemos como conclusión que se pudo realizar de manera efectiva la implementación de la arquitectura propuesta, tanto como en componentes como en la cuestión de realizar una solución de bajo presupuesto acorde a la magnitud de la tarea solicitada.

Durante la realización de este proyecto se logró aprender e implementar varios conceptos sobre la seguridad informática, desde el envío seguro de la información mediante RSA y TCP via sockets de python, como el concepto de firma digital para verificar la autenticidad de los datos mediante el uso de curvas elípticas.

Para los pasos finales a este trabajo se pueden tener varios, algunos de los que se pueden considerar de momento son los siguientes.

1. Implementación de usuarios, debido a que esto fue la fase de prueba no esta incluido, pero cuando se intente pasar a un desarrollo formal se incorporaría la cuestión de creación de usuarios de empleados de LiCore para que únicamente las personas autorizadas puedan acceder a esta información.
2. Actualmente solo se trabajaron con 2 auditores, pero cuando se decida dar los siguientes pasos sería ajustarlo para que puedas soportar la escala que la OSF tiene en mente.
3. Implementación de alguna aplicación para que las personas que cuenten con los smartmeters puedan ingresar y saber cual es su consumo actual y también su histórico.
4. Cambios constantes a la interfaz de usuario y código de ejecución, esto es natural ya que los trabajos tienden a cambiar conforme se van desarrollando.
5. Otra opción es cambiar el TSP y RCA por TLS/SSL para poder mandar mas información tambien de manera cifrada
6. Podríamos mejorar la visualización de los datos con diferentes grafos no solo con los que mostramos e incluso implementar inteligencia artificial para poder hacer predicciones para días siguientes.

Por último, algunas de las limitantes existentes a mencionar durante la realización de la solución es principalmente el alcance que tiene, el principal motivo de dicha limitante son por los recursos, debido a que se nos pidió realizar algo de bajo costo, existen opciones limitadas de hasta donde podemos realizar avances.

Si bien la realización de este trabajo fue principalmente un concepto de prueba, se sugiere fuertemente a quienes quieran considerar realizar el proyecto a su escala completa no escatimar en gastos, además de construir un equipo multidisciplinario para que se puedan realizar las tareas de mejor manera.

Referencias

- Amazon. (s.f.). Amazon RDS Security. <https://aws.amazon.com/es/rds/features/security/>
- Cao, Y.-N., Wang, Y., Ding, Y., Guo, Z., Wu, Q., & Liang, H. (2023). Blockchain-empowered security and privacy protection technologies for smart grid. *Computer Standards Interfaces*, 85, 103708. <https://doi.org/https://doi.org/10.1016/j.csi.2022.103708>
- Certbot. (s.f.). get your site on https. <https://certbot.eff.org/>
- Corbellini, A. (2015). Elliptic Curve Cryptography: a gentle introduction. <https://andrea.corbellini.name/2015/05/17/elliptic-curve-cryptography-a-gentle-introduction/>
- CROCS. (2020). Standard curve database. <https://neuromancer.sk/std/nist/P-256>
- Equipo2. (2023). Repositorio del código fuente. <https://github.com/AndresSaldanaRdz/PKI-encryption-project>
- Flask. (2022). Welcome to Flask. <https://flask.palletsprojects.com>

- Gourley, D., Totty, B., Sayer, M., Aggarwal, A., & Reddy, S. (2002). *HTTP: the definitive guide*. O'Reilly Media, Inc."
- Hernández Camero, M. (2019). Solución basada en Blockchain para dispositivos IoT de bajas capacidades.
- Huiyeon, K. (s.f.). Step-by-step visual guide on deploying a Flask application on AWS EC2. <https://medium.com/techfront/step-by-step-visual-guide-on-deploying-a-flask-application-on-aws-ec2-8e3e8b82c4f7>
- Ilya, G. (2017). *HTTP Protocols*. O'Reilly Media, Inc."
- Iqbal, A., & Iqbal, T. (2018). Low-cost and secure communication system for remote micro-grids using AES cryptography on ESP32 with LoRa module. *2018 IEEE Electrical Power and Energy Conference (EPEC)*, 1-5.
- Johnson, D., Menezes, A., & Vanstone, S. (2001). The elliptic curve digital signature algorithm (ECDSA). *International journal of information security*, 1, 36-63.
- Mellado Ramírez, A. Ó. (s.f.). Criptosistema PKI usando la curva elíptica.
- Melo, W., Machado, R. C., Peters, D., & Moni, M. (2020). Public-key infrastructure for smart meters using blockchains. *2020 IEEE International Workshop on Metrology for Industry 4.0 & IoT*, 429-434.
- Pornin, T. (2013). Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA). <https://www.rfc-editor.org/rfc/rfc6979#appendix-A.2.5>
- Public, A. (2021). What is PKI? A Public Key Infrastructure Definitive Guide. <https://www.keyfactor.com/resources/what-is-pki/>
- Sáenz Leguizamón, J. A. (s.f.). Implementación de un sistema de seguridad para las comunicaciones en medidores inteligentes de baja tensión en Smart Grids. *Departamento de Ingeniería de Sistemas e Industrial*.
- Vyshnavi, V. R., & Malik, A. (2019). Efficient Way of Web Development Using Python and Flask. *Int. J. Recent Res. Asp*, 6(2), 16-19.
- X.509, a. (2021). What is an X.509 Certificate? Understanding this Vital Safeguard. <https://www.keyfactor.com/blog/what-is-x509-certificate/>