



Instituto Tecnológico y de Estudios Superiores de Monterrey

Escuela de Ingeniería y Ciencias

Ingeniería en Ciencias de Datos y Matemáticas

Uso de álgebras modernas para seguridad y criptografía

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

Nombre	Matrícula
Juan Pablo Echeagaray González	A00830646
Ricardo Camacho Castillo	A01654132
Michelle Yareni Morales Ramón	A01552627
Emily Rebeca Méndez Cruz	A00830768
Daniela García Coindreau	A00830236
Carolina Longoria Lozano	A01721279

Dr. Alberto F. Martínez

Dr. Daniel Otero Fadul

Socio Formador: COCOA, LICORE

Monterrey, Nuevo León

17 de marzo del 2023

Índice

1. Introducción	3
2. Objeto de estudio	3
3. Planteamiento del problema	4
3.1. Descripción general de la problemática	4
3.2. Delimitación del problema	4
4. Justificación	5
4.1. Enfoque en Sostenibilidad	5
4.2. Dimensión legal y económica	5
5. Estado del arte	7
5.1. Dispositivos IoT Endpoint	7
5.2. Encriptado asimétrico	8
5.3. Encriptado simétrico	8
5.4. Funciones Hash	8
5.5. Protocolos de comunicación	9
6. Recursos disponibles	9
7. Objetivos	9
8. Arquitectura propuesta	10
8.1. Preliminares	10
8.1.1. Curvas Elípticas	10
8.1.2. BLAKE2	11
8.1.3. ECDSA con BLAKE2	11
8.1.4. MQTT	11
8.2. Sumario de componentes	12
8.3. Arquitectura de red	13
8.4. Funcionamiento	13
9. Implementación y resultados	14
10. Medio de contacto	16

Índice de figuras

1. Intercambio de Datos a ser Protegido [Extraído de CANVAS]	4
--	---

2.	Esquema sugerido para la prueba de concepto (PdC).[Imagen referenciada de CANVAS] . . .	5
3.	Esquema de arquitectura propuesta	13
4.	Cliente de base de datos en Centro de Control	15
5.	Auditor publicando mensajes	15
6.	Base de datos con mensajes procesados	16

Índice de cuadros

1.	Listado general de los componentes de la red	4
2.	Listado de microcontroladores factibles	7
3.	Resumen de componentes empleados	12

1. Introducción

Mientras avanzamos en el camino de la industrialización, los datos se vuelven más y más vitales para el funcionamiento de cualquier empresa. Más allá de ser de suma importancia para el análisis de su rendimiento, muchas veces funcionan como materia prima de sus procesos. Las fugas de información no son necesariamente nuevas, pero la era digital ha permitido que ocurran de una manera masiva, y ha llegado a afectar a empresas renombradas como Yahoo, Microsoft y Facebook [1]. Muchas veces estas empresas manejan datos sensibles, y una filtración de datos es un problema grave para ellos y sus usuarios.

Este es el caso para nuestros socios formadores, *Cocoa* y *LiCore*. *Cocoa* se enfoca en facilitar la transición energías limpias utilizando herramientas digitales [2], y *LiCore* se enfoca en desarrollar tecnología electrónica en áreas relacionadas a la energía sostenible. Una parte clave en su proceso consiste en documentar información de la cantidad y calidad de la energía, que al obtenerse pasa a un auditor, y del auditor a un centro de control. Estos datos al ser de carácter sensible, necesitan ser transmitidos por un medio seguro, por lo que es imperativo la implementación de un medio de transporte seguro que se adapte a los dispositivos que utiliza el Socio Formador.

A continuación estableceremos el objeto de nuestro estudio, plantearemos el problema a ser solucionado y elaboraremos una justificación. Después, haremos una investigación extensa sobre el estado del arte. Analizaremos los recursos disponibles para la resolución del problema, y nuestros objetivos para solucionarlo. Determinaremos la arquitectura de red a ser utilizada y propondremos nuestra metodología. Finalmente, discutiremos los resultados obtenidos.

2. Objeto de estudio

El caso de estudio general que compete a un proyecto como el presentado es la implementación de un protocolo de clave pública para un *SmartGrid*, que es equivalente a una red heterogénea de dispositivos IoT [3].

La implementación de dicha arquitectura viene de la mano con un conjunto de desafíos asociados al poder computacional restringido de los dispositivos IoT, la necesidad de que la comunicación entre los miembros de la red suceda casi en tiempo real, en la disponibilidad de recursos económicos para la construcción y selección de los dispositivos físicos y/o servicios a contratar, y en la técnica a implementar para resguardar los datos de los auditores en una base de datos segura.

Los datos a enviar serán de carácter energético, conteniendo información del consumo y generación de energía eléctrica asociados a un domicilio y a una etiqueta de tiempo. Los datos generados son de carácter sensible, pues algún agente malicioso podría utilizar algún método de *spyware* para recolectar dichos datos, realizar algún proceso de caracterización para lanzar un ataque acotado y personalizado a los entes asociados al conjunto de datos vulnerados.

3. Planteamiento del problema

3.1. Descripción general de la problemática

En la Figura 1 se muestra un esquema simplificado más general de la red *SmartGrid* que se busca asegurar. En primera instancia, se busca proteger el intercambio de información entre cada auditor y el control center. Se busca asegurar la confidencialidad, integridad y autenticidad en todo el proceso de comunicación para que solamente los dispositivos autorizados por el centro de control puedan publicar y acceder a los datos medidos, que la información medida no sufra de alteración alguna y que existan los medios suficientes para aplicar un criterio de no repudio a los auditores participantes de la red.

En conjunto con la Figura 1, se proporciona la Tabla 3.1 en la que se enumera y describen las labores de los componentes básicos de la red solicitada por el Socio Formador.

Componente	Función	Dispositivo
SmartMeter	Monitoreo de parámetros de consumo y generación de energía eléctrica, envío de datos medidos a su auditor respectivo por medio de WiFi	No aplica
Auditor	Recolección de datos de SmartMeters, envío de datos al centro de control	Microcontroladores, tarjetas tipo Raspberry Pi
Centro de control	Procesamiento de lecturas enviadas por auditores, respaldo encriptado de mediciones en una base de datos	Servidor en la nube

Tabla 1: Listado general de los componentes de la red

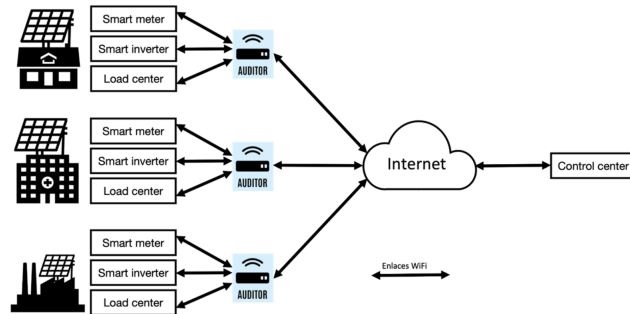


Figura 1: Intercambio de Datos a ser Protegido [Extraído de CANVAS]

3.2. Delimitación del problema

Para fines académicos a organización socio-formadora propone el escenario representado en la Figura 2. Se dispondrá de dos auditores, que reportan el consumo y generación de energía eléctrica de 2 residencias

distintas. Los datos en cuestión no serán obtenidos en tiempo real, sino que han sido proporcionados por el socio-formador en el formato de un archivo csv con los registros históricos de las mediciones.

Los datos recolectados por los auditores deben de ser enviados por internet protegidos hasta que lleguen al Centro de Control. En el Centro de Control, los datos serán almacenados en una base de datos en donde serán respaldados con un esquema de encriptado seguro.

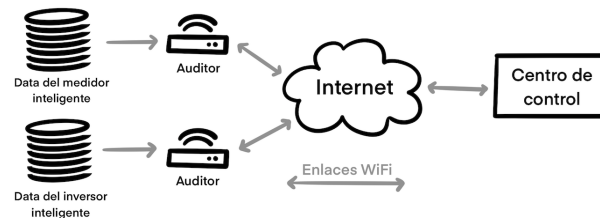


Figura 2: Esquema sugerido para la prueba de concepto (PdC).[Imagen referenciada de CANVAS]

4. Justificación

4.1. Enfoque en Sostenibilidad

Nuestro propósito se alinea con múltiples de los Objetivos de Desarrollo Sustentable de las ONU.

- **Objetivo 9:** «Construir infraestructuras resilientes, promover la industrialización sostenible y fomentar la innovación.» [4] Lo vemos reflejado en nuestro deseo de innovar e industrializar sin sacrificar lo sostenible.
- **Objetivo 11:** «Lograr que las ciudades sean más inclusivas, seguras, resilientes y sostenibles.» [5] Este objetivo esta integrado en las bases del proyecto, con la intención del socio formador de acercar la energía al consumidor, y con su elección de utilizar energías renovables.
- **Objetivo 12:** «Garantizar modalidades de consumo y producción sostenibles.»[6] Es uno de nuestros principales enfoques, al utilizar algoritmos ligeros que serán lo mas sostenibles posibles, y al ayudar a que un proceso de producción de energía sostenible pueda efectuar de manera segura.

4.2. Dimensión legal y económica

De acuerdo a IBM, el costo global promedio en 2022 de una filtración de datos fue de 4 millones de dólares [7]. Esta cifra sigue un patrón de aumento a través del tiempo, lo que nos indica que cada vez se vuelve más caro tener datos desprotegidos. Lo económico no es lo único que se ve afectado por un ataque de esta naturaleza. El estudio del Instituto Ponmeon, "The Aftermath of a Mega Data Breach: Consumer Sentiment", clasifica a las filtraciones de datos como el tercer factor más influyente para la reputación de una empresa, junto con desastres naturales y mal servicio al cliente [8].

Cada segundo se conectan a internet alrededor de 127 dispositivos nuevos. Estos dispositivos atraen a ciberdelicuentes debido a que muchas de las personas que compran dichos aparatos desconocen los puntos importantes de ciberseguridad al momento de adquirirlo, el cual puede poner en riesgo la privacidad del usuario. Los dispositivos IoT usualmente son objetivo de ataques forzados. Los ataques forzados usando motores de búsqueda especializados en donde se seleccionan equipos de protocolos disponibles para conexión, así forzando de manera automática los nombres de usuario y contraseñas comunes. Se estima que a nivel global, la cifra de ataques a dispositivos IoT es mayor a los mil 500 millones. En México, el primer semestre de 2021 se detectaron 661 mil 715 ataques a dispositivos IoT [9]. La cantidad de ataques tuvo un incremento de 197 por ciento a comparación con el año 2020 en donde se registran 222 mil 599.

Como se ha mencionado anteriormente los ataques a dispositivos IoT son comunes y costosos, a continuación se describirán algunos ataques o vulnerabilidades encontradas en dispositivos por el mal uso de primitivas criptográficas:

- En noviembre 2016, cibercriminales apagaron la calefacción de dos edificios en la ciudad finlandesa de Lappeenranta. Después de este ataque, se lanzó un segundo ataque DDoS, en donde obligaron a los controladores de calefacción a reiniciar el sistema repetidamente así evitando que la calefacción se encendiera. Este fue un ataque severo ya que en Finlandia se experimentan temperaturas muy bajas en esa época del año [10].
- En el 2016, investigadores encontraron que alrededor de 100 millones de vehículos Volkswagen vendidos desde 1995 pueden ser atacados por cibercriminales con una facilidad relativa para abrir puertas de forma inalámbrica sin tener la llave. Los vehículos eran vulnerable debido a que se usaron una mínima cantidad de claves privadas compartidas, en donde una vez que se viole una sola clave privada entonces millones de vehículos se verán comprometidos ya que los 100 millones de vehículos y sus llaveros fueron programados con pocas claves secretas comunes utilizadas en todo el mundo. Esto comprometía la seguridad de los pasajeros ya que ellos suponen que el vehículo esta bien cerrado comprometiéndolos a un robo en la carretera, a que criminales coloquen discretamente un objeto o una persona con intenciones maliciosas dentro del automóvil, pone en riesgo la computadora de a bordo del vehículo en donde el criminal puede desactivar los frenos mientras se enciende el sistema de limpiaparabrisas en una curva, etc [11].
- En el 2017, se descubrió que decenas de millones de dispositivos como cámaras de vigilancia de aeropuertos, sensores, equipos de red y dispositivos de IoT eran vulnerables debido a una falla que permitía a los cibercriminales obtener control remoto sobre los dispositivos o bloquearlos. La vulnerabilidad encontrada la nombraron "Devil's Ivy". Esta fue descubierto debido a que investigadores de Senrio quienes seleccionaron cámaras de seguridad de alta gama fabricadas por Axis Communications en donde 249 de los 251 cámaras eran vulnerables a atacantes remotos no autenticados que podían interpretar una transmisión de vídeo, reiniciar cámaras o pausar una transmisión de vídeo mientras cometen un delito [12]. Encontraron que no solamente Axis Communications usaba este software defectuoso si no que otras 34 compañías como Microsoft, IBM, Xerox y Adobe.

5. Estado del arte

La interconexión de dispositivos IoT al internet es uno de los principales desafíos a los que se enfrentan los ingenieros de esta era digital. La naturaleza de recursos restringidos de los dispositivos disponibles vuelve complicada la aplicación directa de los algoritmos y protocolos criptográficos usuales que pueden correr sin más inconveniente en un ordenador de uso personal o un teléfono inteligente.

Esta dificultad ha motivado el diseño de esquemas criptográficos ligeros que ofrecen un balance entre la robustez y seguridad de las interconexiones para con el costo y consumo energético en el que incurren los dispositivos.

5.1. Dispositivos IoT Endpoint

El análisis inicial parte de la selección óptima del dispositivo que fungirá como auditor en la problemática propuesta, la literatura moderna se enfoca en microcontroladores o tarjetas que no suelen sobrepasar el costo de 50 USD. En [13] se realiza un sumario general de las tarjetas usadas comúnmente para después concretar un enfoque del desempeño de distintos algoritmos en una tarjeta *Raspberry Pi 3B*.

Una alternativa atractiva a una tarjeta de gama superior como la anterior, es el microcontrolador *ESP32*. En [14] se emplea dicho microcontrolador en un ambiente similar a la problemática planteada por las bondades de la tarjeta que incluye una suite acelerada para la generación de claves públicas y privadas con RSA, la firma con SHA-256 y la facilidad de generar números suficientemente aleatorios (en el contexto criptográfico).

De forma similar se propone considerar la tarjeta *Raspberry Pi Pico*; no se ha encontrado un artículo científico en el que se detalle su uso en un entorno como el propuesto, pero se recalca que las especificaciones técnicas de la tarjeta descritas en [15] son equivalentes a las del *ESP32*.

Nombre	Chip	RAM	Disco Duro	Conectividad	OS	Lenguaje	Referencia
Raspberry Pi Model 3 B	ARM Cortex A53	1 GB	SD card	WiFi, Ethernet	Raspbian OS	Python, C++, etc...	[13]
ESP32	Tensilica Xtensa LX6	320 kB	448 KB Rom	WiFi, Bluetooth	-	Micropython, Circuit Python	[14]
Raspberry Pi Pico	Arm Cortex-M0+	264 kB	2 MB	WiFi	-	Micropython, C, C++	[15]

Tabla 2: Listado de microcontroladores factibles

5.2. Encriptado asimétrico

La parte más complicada de la comunicación segura entre dispositivos IoT ocurre en la generación, firmado y verificación de mensajes utilizando clave pública y privada. En la actualidad existen 3 principales métodos de encriptado de clave pública que se utilizan entornos IoT como el nuestro, de forma particular se suele analizar el desempeño de RSA, DSA y ECDSA.

En [13] se realiza una comparativa de estos 3 algoritmos en una tarjeta Raspberry Pi 3B. El estudio encontró que en promedio a RSA le toma 20 veces más tiempo firmar un mensaje que ECDSA y utiliza un tamaño de llave 16 veces superior para el mismo nivel de seguridad; el único rubro en el que RSA supera a ECDSA es en el verificado de mensajes, donde es 3 veces más rápido. Se puede ver que en promedio ECDSA tiene el mejor desempeño.

5.3. Encriptado simétrico

Una vez que se ha establecido un secreto en común entre 2 entes se puede aplicar alguna de las técnicas de encriptado simétrico que conocemos; esta suele ser una de las partes más sencillas de cualquier protocolo criptográfico ya que suele ser computacionalmente eficiente y rápido de calcular para evitar problemas de latencia en los procesos de comunicación.

En [13] se realizó una comparativa de los algoritmos de cifrado DES, 3DES, AES128, AES192 y AES256. Una conclusión esperada era que los algoritmos de la clase DES tuvieran una huella reducida a comparación de la suite de AES; a pesar de que su estudio confirma el supuesto, el NIST tiene registros de vulnerabilidades de este algoritmo desde el 2016 [16].

5.4. Funciones Hash

Una parte fundamental de los algoritmos de encriptado descritos es la función *hash*. La principal aportación de este componente es que brinda la habilidad de que un destinatario se asegure de que el mensaje que haya recibido no ha sido modificado en el transcurso de su envío.

Algunas de las cualidades que deben tener estas funciones hash es la sencillez con la que puede ser computada y el que sea segura a colisiones; se entiende por colisión el evento en el que a dos objetos distintos se les asigna un mismo hash.

En [13] se realiza un análisis general del desempeño de las funciones hash MD5, SHA-1, SHA-256 y SHA-512 en una tarjeta Raspberry Pi 3B. La investigación encuentra que las funciones hash MD5 y SHA-1 son las más veloces del conjunto original, sin embargo se ha declarado desde el 2008 que MD5 no es una función hash segura [17]; para el caso de SHA-1 ya se ha emitido un comunicado del *NIST* que recomienda dejar de utilizar esta función [18].

En [19] se realiza un estudio similar al anterior, pero enfocado a la función hash necesaria para el firmado con curvas elípticas. Su conjunto de pruebas consistió en parte de las mismas funciones anteriores en añadidura de HAVAL, KECCAK, y BLAKE. El estudio encontró que la familia SHA es de las mejores disponibles, pero es sobrepasada por la función hash BLAKE en cuanto a uso en curvas elípticas concierne.

5.5. Protocolos de comunicación

Al final del día las funciones y componentes mencionadas representan partes estrictamente aisladas del protocolo criptográfico a implementar para asegurar las conexiones en la red propuesta. Es por eso que después se proponen mezclas y técnicas que implementan los algoritmos criptográficos base que después son acotados y personalizados a diferentes casos de estudio.

En [20] se propone un enfoque similar a arquitecturas cliente-servidor, en el que se envían datos por medio de HTTPS haciendo uso de TLS para asegurar las vías de comunicación. El estudio propuso una arquitectura que necesita de menos mensajes para asegurar un canal de comunicación, pero que no logra concretar una política de acceso detallada basada en los recursos a los cuales se desea acceder y/o modificar.

En [21] se propone un esquema ligero que no utiliza el concepto de certificados para el proceso de autenticación y autorización, para ser remplazados por un esquema de comunicación sobre MQTT incluyendo *hasheo* con firmado de claves públicas generadas de forma local, sin que se incurra en un alto consumo energético o que se añada una latencia descomunal al proceso de comunicación.

6. Recursos disponibles

La organización socio formadora ha estipulado que el auditor propuesto no puede superar un costo total de 100 USD; sin embargo ha manifestado su preferencia por un auditor que tenga un costo alrededor de los 50 USD. Para emular dichos dispositivos se propone el uso de dos tarjetas de microprocesador Raspberry Pi 3B (modelo a determinar en base a disponibilidad del laboratorio de robótica).

Para la emulación del centro de control se propone utilizar una instancia de EC2 de Amazon Web Services, en la que se emulará un servidor con una base de datos que utilice MySQL.

7. Objetivos

El objetivo principal de este proyecto es el desarrollo de una arquitectura de red que permita que un conjunto de dispositivos auditores envíen lecturas de consumo y generación de energía eléctrica a un centro de control por medio de una conexión WiFi. Se parte de que los auditores tienen poder computacional bajo, y los datos generados deben de ser enviados y almacenados de forma segura.

En términos cuantitativos se plantean las siguientes metas:

1. Costo total del dispositivo auditor menor a **100 USD** por pieza.
2. El tiempo de envío seguro de datos debe de ser menor a **15 minutos**, puesto que las lecturas son generadas en ventanas de tiempo de 15 minutos.

8. Arquitectura propuesta

En base a la investigación realizada así como a las comparativas del desempeño de los diferentes algoritmos criptográficos y los requisitos monetarios impuestos por el Socio Formador; se propone un esquema criptográfico adaptado de [21]. Nuestra propuesta aprovecha el tamaño de llave reducido de curvas elípticas así como la rapidez con la que se pueden aplicar la función BLAKE-2 en dispositivos Raspberry Pi 3B.

En esta sección se presenta la arquitectura propuesta, así como los elementos que la conforman y el modelado del sistema. Se partirá con una explicación detallada de los fundamentos matemáticos de ECDSA así como del protocolo de comunicación MQTT; posteriormente se presentará un listado de los componentes de la red propuesta, así como una descripción de las tareas que realizarán en cada una de las etapas de manufactura, arranque y producción.

8.1. Preliminares

8.1.1. Curvas Elípticas

Las curvas elípticas son utilizadas para la reducción de la complejidad de un algoritmo en los métodos más empleados de la criptografía. La criptografía de curva elíptica es utilizada cada vez más en la práctica para protocolos de criptografía de clave pública, su utilidad es principalmente debido a sus características matemáticas de curvas elípticas planas [22].

La criptografía de curvas elípticas es una variante de la criptografía asimétrica, esta cuenta con una llave pública y otra privada con la que se descifra el mensaje. Una de sus ventajas es que cuenta con una mayor velocidad a comparación de los otros algoritmos de cifrado asimétrico, además de usar claves más cortas con un tamaño mínimo recomendado por la NIST de 160 bits para CCE en comparación con 1024 bits para los algoritmos RSA y DSA [23]. Esta reducción al tamaño de las claves de bits intercambiadas entre entidades se debe a la aplicación de la sucesión matemática que se muestra a continuación:

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

Donde a y b son números enteros que satisfacen

$$4a^3 + 27b^2 \neq 0 \quad (2)$$

Sobre campos finitos primos $F(q)$, seleccionando dos puntos enteros de la curva P y Q que cumplan que $P + Q = R$, siendo R el punto inverso de intersección entre los dos puntos anteriores. Esto produce una suma repetida de puntos P , de tal forma que se considerará $Q = P + P + \dots + P$; al emplear esta sucesión matemática se reduce considerablemente el tamaño de las claves de bits intercambiadas. Una clave de ECC de 224 bits brinda una seguridad semejante a una clave RSA de 2048 bits, esto permite una mayor eficiencia en la transmisión de información [22].

8.1.2. BLAKE2

BLAKE2b, simplemente conocido como BLAKE2 fue diseñado por un equipo de expertos en criptoanálisis, implementación e ingeniería criptográfica; miembros de ese equipo son, Jean-Philippe Aumasson, Samuel Neves, Zooko Wilcox-O'Hearn y Christian Winnerlein. BLAKE2 es una función hash criptográfica basada en el cifrado de flujo ChaCha. Sin embargo, en BLAKE2 se agrega una copia permutada del bloque de entrada, XORed con constantes de ronda, antes de cada ronda de ChaCha. BLAKE2 está optimizado para plataformas de 64 bits y produce resúmenes de cualquier tamaño entre 1 y 64 bytes. Una característica clave del algoritmo BLAKE2 es que este no requiere una construcción especial "HMAC", es decir, la construcción de un código de autenticación de mensaje hash para la autenticación de mensajes con clave, ya que este tiene un mecanismo de clave incorporado. Además, BLAKE2 puede ser utilizado en algoritmos de firma digital y autenticación de mensajes y mecanismos de protección de integridad en aplicaciones tales como infraestructura de clave pública (PKI), almacenamiento en la nube, detección de intrusos, suites forenses, sistemas de control de versiones y protocolos de comunicación segura. [24].

8.1.3. ECDSA con BLAKE2

Normalmente, las funciones hash SHA se utilizan en diversas aplicaciones y criptografía de clave pública, sin embargo, SHA tiene 80 iteraciones diferentes las cuales consumen energía del dispositivo inteligente. Por esta razón se opta por usar una función hash más ligera basada en BLAKE2. BLAKE2 usa 8 rondas de permutaciones y combinaciones para producir el resumen del mensaje de 256/512 bits. Se ha visto en un estudio que esta reducción en el cómputo ha atraído a los investigadores a utilizar las funciones de hash de BLAKE2 para diversas aplicaciones. A través de varios experimentos realizados en diferentes sistemas operativos, se ha demostrado que BLAKE2 tiene una mejor eficiencia computacional que la familia SHA de funciones hash para el esquema basado en ECDSA en diferentes sistemas operativos. [25].

8.1.4. MQTT

El protocolo MQTT (Message Queue Telemetry Transport) es un protocolo de publicación-suscripción, este facilita la conexión a una gran cantidad de dispositivos y además tiene cierta facilidad para comprobar que el mensaje llegue al objetivo proporcionando una seguridad mediante certificados electrónicos [?]. Este protocolo de mensajería asíncrona es utilizado en el ámbito de las comunicaciones machine-to-machine (M2M) en el campo de IoT [26].

El modelo publicación y suscripción funciona mediante la conexión a un broker por parte de los clientes. Para el envío de mensajes y para su posterior recepción, ambos se suscriben a un topic donde se disponen los mensajes, convirtiendo este modelo en un modelo no-bloqueante. Un cliente MQTT es tanto los suscriptores como los publicadores, un cliente MQTT puede ser un suscriptor, publicador o ambos a la vez [26].

El broker MQTT es el servidor encargado de la distribución de los mensajes a los receptores; este recibe los mensajes, realiza un chequeo del topic al que están suscritos y los encamina hacia los clientes suscritos a este mismo topic. El broker brinda la posibilidad que se guarden los mensajes hasta que el cliente al que

va dirigido se conecte. Otra función que tiene el broker es la de autenticar la identidad de los clientes como medida de seguridad [26].

El tópic, el tema del mensaje, sirve como identificador al broker para poder dirigir cada uno de los mensajes recibidos hacia los clientes adecuados que estén suscritos al topic [26].

8.2. Sumario de componentes

En la tabla 3 se presenta un resumen general de cada uno de los componentes necesarios para el desarrollo del proyecto, acompañados de una descripción concisa de las tareas que realizarán en las etapas de manufactura, arranque y producción.

Nombre	Manufactura	Arranque	Producción
Audidores (Raspberry Pi 3B)	<ul style="list-style-type: none"> ■ Instalación de software base ■ Instalación de certificado de fábrica 	<ul style="list-style-type: none"> ■ Registro de Smart Meters (esquema similar) ■ Carga del certificado propio, del Centro de Control y la CA ■ Conexión al Broker mediante TLS 	<ul style="list-style-type: none"> ■ Recepción de datos de Smart Meters (lectura de datos de prueba) ■ Firmado de tramas con ECDSA ■ Recepción y procesamiento de ajustes de parámetros del Centro de Control
Mosquitto Broker (EC2)	<ul style="list-style-type: none"> ■ Instalación de Mosquitto Broker ■ Sobre-escritura de archivo de configuración de Mosquitto ■ Instalación de certificado propio 	Establecimiento de conexión TLS entre cada cliente	<ul style="list-style-type: none"> ■ Recepción de tramas en tópicos permitidos ■ Reenvío de tramas de auditores hacia el Centro de Control y viceversa
Centro de Control (Laptop)	<ul style="list-style-type: none"> ■ Creación de certificados para auditores, Broker y sí mismo, estableciendo al Broker como CA ■ Creación de base de datos de sqlite3 	<ul style="list-style-type: none"> ■ Instalación de software base ■ Instalación de certificado propio, de la CA, y cada auditor ■ Conexión al Broker mediante TLS 	<ul style="list-style-type: none"> ■ Recepción de tramas firmadas de auditores ■ Verificación y firmas para escritura en la base de datos si la verificación es exitosa ■ Envío de actualizaciones de parámetros a los auditores

Tabla 3: Resumen de componentes empleados

Se aclara que que el Centro de Control deberá ser remplazado por una máquina distinta, el equipo recomienda el uso de otra instancia de EC2 con un mayor poder computacional que pueda procesar todas las tramas concurrentes de los auditores.

8.3. Arquitectura de red

En la figura 3 se presenta el esquema general de cómo se vería la red de auditores.

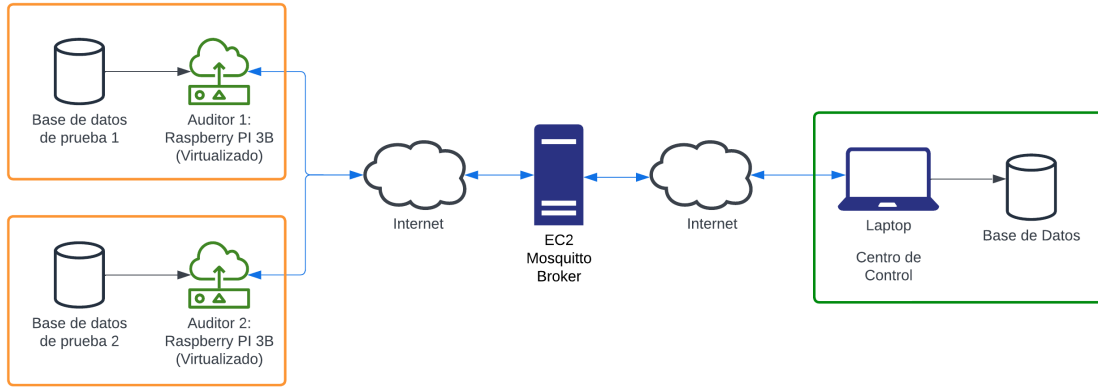


Figura 3: Esquema de arquitectura propuesta

En la figura se maneja un código de colores para representar el tipo de comunicación existente entre cada dispositivo:

- Flecha Azul: Canal cifrado con TLS
- Flecha Gris: Canal sin protección con envío de datos en texto plano
- Recuadro Naranja: Entorno ajeno al Socio Formador
- Recuadro Verde: Entorno seguro manejado por el Socio Formador

Se ha realizado una distinción entre la red que comunica a cada auditor y la red del Socio Formador, puesto que se considera necesario implementar medidas de seguridad extra para asegurarse de que las mediciones de los Smart Meters lleguen sin alteraciones a sus respectivos auditores.

El equipo no ha desarrollado software especializado para manejar esta situación puesto que no se dispuso de información alguna del tipo de dispositivos que generan estas mediciones.

8.4. Funcionamiento

1. **Fase de registro:** El Centro de Control CC crea certificados para todos los auditores A_i , el Broker B y para sí mismo. Se establece como CA a B .

2. **Envío de certificados:** CC distribuye todos los certificados a cada uno de los clientes. Para el caso de B se debe de seguir la guía de configuración propuesta en nuestra guía de certificados.

3. **Inicio de procesos:** se debe programar la corrida de los scripts `publisher.py` y `database_client.py` provistos. Para nuestra demostración se dejarán corriendo desde que inicien a operar.

Cada auditor solamente podrá publicar mensajes en el tópico `audits/measurements/ID-i`, y se suscribirá al tópico `control_center/updates/ID-i/#`.

4. **Carga de certificados propios:** Cada auditor A_i cargará su certificado, llave privada y llave pública en memoria. El CC realiza esta misma operación.

5. **Conexión con broker:** Cada auditor A_i y el CC se conectarán a B enviando sus certificados. El broker determinará la validez de los mismos y determinará si se establece una conexión.

6. **Envío de tramas:** Cada auditor A_i dispone de su identificador ID_i , la fecha de medición d , el tipo de medición t y el valor numérico observado x , se envía entonces un paquete como: $P = ID_i \backslash d \backslash t \backslash x$. Dicha trama es firmada utilizando la llave privada K_i , siendo el paquete final $P_f = \text{Sign}(P, K_i) || P$. Dicho paquete será publicado en el tópico del auditor antes siendo encriptado por la conexión TLS establecida.

7. **Autenticación de tramas:** CC verifica el paquete recibido haciendo uso de la clave pública Pub_i de A_i , si $\text{Verify}(\text{Sign}(P, K_i), Pub_i)$ produce un resultado válido, la trama P será guardada en la base de datos.

8. **Actualización de parámetros:** CC puede realizar un proceso similar para alterar el funcionamiento de A_i , deberá repetir los pasos 3 a 7 para enviar una actualización válida a través del tópico `control_center/updates/ID_i/#`

9. Implementación y resultados

Para probar el código que se ha desarrollado se ha de ejecutar los siguientes comandos en una terminal de Linux:

```
1 $ git clone https://github.com/JuanEcheagaray75/licore-pki
2 # Instalar certificados y llaves privadas provistas por
   separado
3 $ cd licore-pki/db
4 $ ./create_db.sh
5 $ cd ../src
6 $ python3 -m venv .venv
7 $ source .venv/bin/activate
8 $ pip install -r requirements.txt
9 $ cd clients
10 # Correr en laptop
11 $ python3 database_client.py
12 # Correr en Raspberry Pi
13 $ python3 publisher.py
14 # Probar actualización de base de datos
15 $ sqlite3 ../db/measures.sqlite3; select * from measures;
16
```

Para demostrar el funcionamiento del código se ha propuesto limitar el número de mensajes a enviar a 20. En la figura 4 se muestra el procesamiento de dichos 20 mensajes; para que estos sean procesados fue necesario que la firma del auditor fuera válida y que el mensaje no hubiera sido procesado previamente.

En la figura 5 se muestra el auditor emulado como un Raspberry PI 3B en VirtualBox enviando mensajes al broker. Se puede observar que el auditor se suscribe al tópico `control_center/updates/ID_i/#` y que publica mensajes en el tópico `audits/measures/ID_i`.

El auditor envía por defecto una trama cada 0.2 segundos, notamos también que al final se reportó un periodo de procesamiento de 9.57 segundos para los 20 mensajes enviados, siendo un tiempo de procesamiento por mensaje promedio de 0.478 segundos.

Finalmente se comprueba que los mensajes fueron creados y procesados exitosamente al consultar la base de datos, como se muestra en la figura 6.

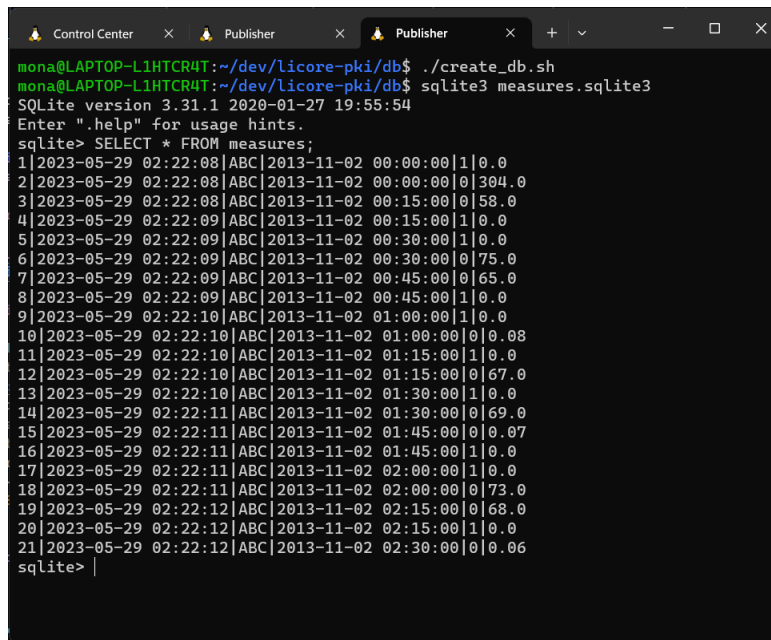
```
Control Center x Publisher x Publisher x + v - □ x
(.venv) mona@LAPTOP-L1HTCR4T:~/dev/licore-pki/src$ cd clients
(.venv) mona@LAPTOP-L1HTCR4T:~/dev/licore-pki/src/clients$ python3 database_client.py

Connected through MQTT 0
Packet ABC/2013-11-02 00:00:00/1/0.0 written to database
Packet ABC/2013-11-02 00:00:00/0/304.0 written to database
Packet ABC/2013-11-02 00:15:00/0/58.0 written to database
Packet ABC/2013-11-02 00:15:00/1/0.0 written to database
Packet ABC/2013-11-02 00:30:00/1/0.0 written to database
Packet ABC/2013-11-02 00:30:00/0/75.0 written to database
Packet ABC/2013-11-02 00:45:00/0/65.0 written to database
Packet ABC/2013-11-02 00:45:00/1/0.0 written to database
Packet ABC/2013-11-02 01:00:00/1/0.0 written to database
Packet ABC/2013-11-02 01:00:00/0/0.08 written to database
Packet ABC/2013-11-02 01:15:00/1/0.0 written to database
Packet ABC/2013-11-02 01:15:00/0/67.0 written to database
Packet ABC/2013-11-02 01:30:00/1/0.0 written to database
Packet ABC/2013-11-02 01:30:00/0/69.0 written to database
Packet ABC/2013-11-02 01:45:00/0/0.07 written to database
Packet ABC/2013-11-02 01:45:00/1/0.0 written to database
Packet ABC/2013-11-02 02:00:00/1/0.0 written to database
Packet ABC/2013-11-02 02:00:00/0/73.0 written to database
Packet ABC/2013-11-02 02:15:00/0/68.0 written to database
Packet ABC/2013-11-02 02:15:00/1/0.0 written to database
Packet ABC/2013-11-02 02:30:00/0/0.06 written to database
```

Figura 4: Cliente de base de datos en Centro de Control

```
Publisher-Raspbian [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
publisher@raspberrypi: ~/dev/licore-pki/src/clients
File Edit Tabs Help
publisher@raspberrypi:~/dev/licore-pki/src/clients$ python3 publisher.py
Publishing packets...
Packet ABC/2013-11-02 00:00:00/1/0.0 published
Connected through MQTT 0
Received parameter update as: hello auditor
Packet ABC/2013-11-02 00:00:00/0/304.0 published
Packet ABC/2013-11-02 00:15:00/0/58.0 published
Packet ABC/2013-11-02 00:15:00/1/0.0 published
Packet ABC/2013-11-02 00:30:00/1/0.0 published
Packet ABC/2013-11-02 00:30:00/0/75.0 published
Packet ABC/2013-11-02 00:45:00/0/65.0 published
Packet ABC/2013-11-02 00:45:00/1/0.0 published
Packet ABC/2013-11-02 01:00:00/1/0.0 published
Packet ABC/2013-11-02 01:00:00/0/0.08 published
Packet ABC/2013-11-02 01:15:00/1/0.0 published
Packet ABC/2013-11-02 01:15:00/0/67.0 published
Packet ABC/2013-11-02 01:30:00/1/0.0 published
Packet ABC/2013-11-02 01:30:00/0/69.0 published
Packet ABC/2013-11-02 01:45:00/0/0.07 published
Packet ABC/2013-11-02 01:45:00/1/0.0 published
Packet ABC/2013-11-02 02:00:00/1/0.0 published
Packet ABC/2013-11-02 02:00:00/0/73.0 published
Packet ABC/2013-11-02 02:15:00/0/68.0 published
Packet ABC/2013-11-02 02:15:00/1/0.0 published
Done!
9.576140308999902
0.4788070154499901
publisher@raspberrypi:~/dev/licore-pki/src/clients$
```

Figura 5: Auditor publicando mensajes



```
Control Center x Publisher x Publisher x + - □ x
mona@LAPTOP-L1HTCR4T:~/dev/licore-pki/db$ ./create_db.sh
mona@LAPTOP-L1HTCR4T:~/dev/licore-pki/db$ sqlite3 measures.sqlite3
SQLite version 3.31.1 2020-01-27 19:55:54
Enter ".help" for usage hints.
sqlite> SELECT * FROM measures;
1|2023-05-29 02:22:08|ABC|2013-11-02 00:00:00|1|0.0
2|2023-05-29 02:22:08|ABC|2013-11-02 00:00:00|0|304.0
3|2023-05-29 02:22:08|ABC|2013-11-02 00:15:00|0|58.0
4|2023-05-29 02:22:09|ABC|2013-11-02 00:15:00|1|0.0
5|2023-05-29 02:22:09|ABC|2013-11-02 00:30:00|1|0.0
6|2023-05-29 02:22:09|ABC|2013-11-02 00:30:00|0|75.0
7|2023-05-29 02:22:09|ABC|2013-11-02 00:45:00|0|65.0
8|2023-05-29 02:22:09|ABC|2013-11-02 00:45:00|1|0.0
9|2023-05-29 02:22:10|ABC|2013-11-02 01:00:00|1|0.0
10|2023-05-29 02:22:10|ABC|2013-11-02 01:00:00|0|0.08
11|2023-05-29 02:22:10|ABC|2013-11-02 01:15:00|1|0.0
12|2023-05-29 02:22:10|ABC|2013-11-02 01:15:00|0|67.0
13|2023-05-29 02:22:10|ABC|2013-11-02 01:30:00|1|0.0
14|2023-05-29 02:22:11|ABC|2013-11-02 01:30:00|0|69.0
15|2023-05-29 02:22:11|ABC|2013-11-02 01:45:00|0|0.07
16|2023-05-29 02:22:11|ABC|2013-11-02 01:45:00|1|0.0
17|2023-05-29 02:22:11|ABC|2013-11-02 02:00:00|1|0.0
18|2023-05-29 02:22:11|ABC|2013-11-02 02:00:00|0|73.0
19|2023-05-29 02:22:12|ABC|2013-11-02 02:15:00|0|68.0
20|2023-05-29 02:22:12|ABC|2013-11-02 02:15:00|1|0.0
21|2023-05-29 02:22:12|ABC|2013-11-02 02:30:00|0|0.06
sqlite> |
```

Figura 6: Base de datos con mensajes procesados

10. Medio de contacto

El desarrollo del proyecto así como la redacción del presente documento es un trabajo conjunto de:

- Juan Pablo Echeagaray González
- Ricardo Camacho Castillo
- Michelle Yareni Morales Ramóz
- Emily Rebeca Méndez Cruz
- Daniela García Coindreau
- Carolina Longoria Lozano

Así mismo se destacan los siguientes profesores, como asesores y supervisores de los avances en el desarrollo del proyecto:

- Dr. Alberto F. Martínez
- Dr. Daniel Otero Fadul

El benefactor principal del proyecto es la organización *LiCore*, la comunicación con la organización se vio llevada principalmente por el Dr. Iván S. Razo-Zapata.

En caso de encontrar fallas en el código fuente, o que se necesite de una aclaración de la implementación propuesta; se pide que se abra un *issue* en el repositorio en GitHub que puede ser accedido desde la siguiente liga.

Para dudas generales sobre el proyecto, se puede contactar a Juan Pablo Echeagaray González, al correo a00830646@tec.mx

Referencias

- [1] 2023. [Online]. Available: <https://www.upguard.com/blog/biggest-data-breaches-us>
- [2] “Cocoa.” [Online]. Available: <https://www.cocoa-ci.org/>
- [3] “Smart Grids - Analysis - IEA.” [Online]. Available: <https://www.iea.org/reports/smart-grids>
- [4] M. Moran, “Infraestructura - desarrollo sostenible,” Jun 2020. [Online]. Available: <https://www.un.org/sustainabledevelopment/es/infrastructure/>
- [5] —, “Ciudades - desarrollo sostenible,” Jun 2020. [Online]. Available: <https://www.un.org/sustainabledevelopment/es/cities/>
- [6] —, “Consumo y producción sostenibles - desarrollo sostenible,” Jun 2020. [Online]. Available: <https://www.un.org/sustainabledevelopment/es/sustainable-consumption-production/>
- [7] “Cost of a data breach 2022: A million-dollar race to detect and respond,” 2022. [Online]. Available: <https://www.ibm.com/reports/data-breach>
- [8] *The Aftermath of a Data Breach: Consumer Sentiment*, 2014. [Online]. Available: <https://www.ponemon.org/local/upload/file/Consumer%20Study%20on%20Aftermath%20of%20a%20Breach%20FINAL%202.pdf>
- [9] A. Ríos, “México i arriesgan seguridad con mal uso de iot,” Jan 2022. [Online]. Available: <https://dplnews.com/mexico-i-arriesgan-seguridad-con-mal-uso-de-iot/>
- [10] A. Husar, “Iot security: 5 cyber-attacks caused by iot security vulnerabilities,” Oct 2022. [Online]. Available: <https://www.cm-alliance.com/cybersecurity-blog/iot-security-5-cyber-attacks-caused-by-iot-security-vulnerabilities>
- [11] F. D. Garcia, D. Oswald, T. Kasper, and P. Pavlides, “Lock it and still lose it - on the (in)security of automotive remote keyless entry systems.”
- [12] T. Spring, “Bad code library triggers devil’s ivy vulnerability in millions of iot devices,” Jul 2019. [Online]. Available: <https://threatpost.com/bad-code-library-triggers-devils-ivy-vulnerability-in-millions-of-iot-devices/126913/>
- [13] M. El-Haii, M. Chamoun, A. Fadlallah, and A. Serhrouchni, “Analysis of cryptographic algorithms on iot hardware platforms,” in *2018 2nd Cyber Security in Networking Conference (CSNet)*. IEEE, 2018, pp. 1–5.
- [14] A. Anand, A. Galletta, A. Celesti, M. Fazio, and M. Villari, “A secure inter-domain communication for iot devices,” in *2019 IEEE International Conference on Cloud Engineering (IC2E)*. IEEE, 2019, pp. 235–240.

- [15] R. P. Ltd, “Buy a Raspberry Pi Pico - Raspberry Pi.” [Online]. Available: <https://www.raspberrypi.com/products/raspberry-pi-pico/>
- [16] “NVD - CVE-2016-2183.” [Online]. Available: <https://nvd.nist.gov/vuln/detail/CVE-2016-2183>
- [17] “What is MD5? Understanding Message-Digest Algorithms — Okta.” [Online]. Available: <https://www.okta.com/identity-101/md5/>
- [18] “NIST Retires SHA-1 Cryptographic Algorithm — NIST,” 12 2022. [Online]. Available: <https://www.nist.gov/news-events/news/2022/12/nist-retires-sha-1-cryptographic-algorithm>
- [19] V. Rao and K. Prema, “Comparative study of lightweight hashing functions for resource constrained devices of iot,” in *2019 4th International Conference on Computational Systems and Information Technology for Sustainable Solution (CSITSS)*, vol. 4. IEEE, 2019, pp. 1–5.
- [20] V. Abreu, A. O. Santin, E. K. Viegas, and V. V. Cogo, “Identity and access management for iot in smart grid,” in *Advanced Information Networking and Applications: Proceedings of the 34th International Conference on Advanced Information Networking and Applications (AINA-2020)*. Springer, 2020, pp. 1215–1226.
- [21] A. Lohachab *et al.*, “Ecc based inter-device authentication and authorization scheme using mqtt for iot networks,” *Journal of Information Security and Applications*, vol. 46, pp. 1–12, 2019.
- [22] J. Á. Sánchez Martín, “Solución de cifrado para dispositivos de bajas capacidades,” 2021.
- [23] A. Ramírez García *et al.*, “Sistemas multi agentes para la defensa de redes iot,” 2018.
- [24] Feb 2017. [Online]. Available: <https://www.blake2.net/>
- [25] V. Rao and K. V. Prema, “Comparative study of lightweight hashing functions for resource constrained devices of iot,” in *2019 4th International Conference on Computational Systems and Information Technology for Sustainable Solution (CSITSS)*, 2019, pp. 1–5.
- [26] F. Mahedero Biot, “Desarrollo de una aplicación iot para el envío de imágenes mediante el protocolo mqtt,” Ph.D. dissertation, Universitat Politècnica de València, 2020.