UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE QUITO

CARRERA DE COMPUTACIÓN

ANÁLISIS DE MODELOS DE CIFRADO PARA LA TRANSMISIÓN DE DATOS EN SISTEMAS EMBEBIDOS Y SU APLICACIÓN EN EL INTERNET DE LAS COSAS (IOT)

Trabajo de titulación previo a la obtención del
Título de Ingeniera en Ciencias de la Computación

AUTORA: MARJORIE GABRIELA LASLUISA MOROCHO

TUTOR: MANUEL RAFAEL JAYA DUCHE

Quito - Ecuador
2025

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Yo, Lasluisa Morocho Marjorie Gabriela con documento de identificación N° 1727008581 manifiesto que:

Soy el autora y responsable del presente trabajo; y, autorizo a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total parcial el presente trabajo de titulación.

Quito, 26 de febrero de 2025

Atentamente,

_____

Lasluisa Morocho Marjorie Gabriela
1727008581

CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Yo, Lasluisa Morocho Marjorie Gabriela con documento de identificación N° 1727008581, expreso mi voluntad y por medio del presente documento cedo a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que soy autora del Artículo Académico: "Análisis de modelos de cifrado para la transmisión de datos en sistemas embebidos y su aplicación en el Internet de las Cosas (IoT)", el cual ha sido desarrollado para optar por el título de: Ingeniera en Ciencias de la Computación, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribo este documento en el momento que hago la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Quito, 26 de febrero de 2025

Atentamente,

_____

Lasluisa Morocho Marjorie Gabriela
1727008581

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, Manuel Rafael Jaya Duche con documento de identificación N° 1710631035, docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: ANÁLISIS DE MODELOS DE CIFRADO PARA LA TRANSMISIÓN DE DATOS EN SISTEMAS EMBEBIDOS Y SU APLICACIÓN EN EL INTERNET DE LAS COSAS (IOT), realizado por Lasluisa Morocho Marjorie Gabriela con documento de identificación N° 1727008581, obteniendo como resultado final el trabajo de titulación bajo la opción de Artículo Académico que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Quito, 26 de febrero de 2025

Atentamente,

Ing. Manuel Rafael Jaya Duche, MSc.
1710631035

DEDICATORIA

Dedico este trabajo a mis padres, quienes han sido el pilar fundamental en mi vida. Su amor, sacrificio y constante apoyo me han dado la fortaleza para enfrentar cada desafío y la motivación para alcanzar mis metas. Gracias por enseñarme con su ejemplo el verdadero significado del esfuerzo, la dedicación y la perseverancia. Todo lo que soy y lo que he logrado se lo debo a ustedes.

A mis abuelitos, cuyo amor incondicional y sabiduría han sido una fuente de inspiración inagotable. Sus historias, consejos y enseñanzas han marcado profundamente mi vida, y su apoyo ha sido mi refugio en los momentos más difíciles. Este logro es un homenaje a todo lo que han representado para mí y a las raíces que han forjado con tanto amor.

Con infinito cariño, gratitud y admiración, les dedico este trabajo, esperando que sea un motivo de orgullo y alegría para ustedes, así como ustedes lo son para mí.

Lasluisa Morocho Marjorie Gabriela

AGRADECIMIENTO

Agradezco, en primer lugar, a Dios, por darme la fortaleza, la sabiduría y las oportunidades necesarias para alcanzar este logro.

A mis amados padres, por ser el pilar fundamental de mi vida. Gracias por su apoyo incondicional, su amor infinito y por enseñarme con su ejemplo que no hay límites para los sueños cuando se trabaja con dedicación y perseverancia.

A mis abuelitos, que han sido una fuente inagotable de amor, consejos sabios y valores. Sus palabras y abrazos han sido siempre un refugio y una inspiración para seguir adelante.

Finalmente, agradezco a todas las personas que, de una forma u otra, han dejado una huella en mi vida, aportando para que este proyecto sea una realidad. A todos ustedes, mi más sincero agradecimiento.

Lasluisa Morocho Marjorie Gabriela

ANÁLISIS DE MODELOS DE CIFRADO PARA LA TRANSMISIÓN DE DATOS EN SISTEMAS EMBEBIDOS Y SU APLICACIÓN EN EL INTERNET DE LAS COSAS (IOT)

ANALYSIS OF ENCRYPTION MODELS FOR DATA TRANSMISSION IN EMBEDDED SYSTEMS AND THEIR APPLICATION IN THE INTERNET OF THINGS (IOT)

Gabriela Lasluisa – Morocho[1] Rafael Jaya – Duche[2]

Resumen

La finalidad principal de esta investigación es [1] comparar diversas estrategias de protección implementadas en plataformas IoT, con un énfasis especial en la transmisión segura de datos entre dispositivos finales. Para llevar a cabo este análisis, se utilizará un módulo ESP32 conectado a sensor de temperatura y humedad (DHT22), mientras se simulan ataques e intrusiones mediante herramientas como EtterCap. Además, se capturarán y examinarán los paquetes de datos con Wireshark para identificar posibles vulnerabilidades y evaluar cómo estas plataformas pueden defenderse contra amenazas durante la transmisión de información. Este estudio tiene como objetivo identificar los mecanismos de seguridad más efectivos para proteger los datos en sistemas IoT con recursos limitados. A partir de este análisis, se proporcionarán recomendaciones específicas para mejorar la seguridad en las comunicaciones entre dispositivos, lo que no solo fortalecerá la confianza en estas tecnologías emergentes, sino que también disminuirá un 20% significativamente los riesgos asociados a ciberataques en entornos IoT.

Palabras clave: Seguridad de datos, Transmisión segura,ESP32,Sensores DHT22, Vulnerabilidades, Ciberataques ,Plataformas IoT .

Abstract

The primary aim of this research is [1] to compare various protection strategies implemented on IoT platforms, with a special focus on the secure transmission of data between end devices. To conduct this analysis, an ESP32 module connected to a temperature and humidity sensor (DHT22) will be used while simulating attacks and intrusions with tools like EtterCap. Additionally, data packets will be captured and analyzed using Wireshark to identify potential vulnerabilities and evaluate how these platforms can defend against threats during information transmission. This study aims to identify the most effective security mechanisms to protect data in resource-constrained IoT systems. Based on this analysis, specific recommendations will be provided to enhance communication security between devices, which will not only strengthen confidence in these emerging technologies but also significantly reduce risks associated with cyberattacks in IoT environments by 20%.

Keywords: Data Security, Secure Transmission, ESP32, DHT22 Sensors, Vulnerabilities, Cyberattacks, IoT Platforms

[1] Estudiante de la Carrera de Computación, Universidad Politécnica Salesiana.

2 Docente de la Carrera de Computación, Universidad Politécnica Salesiana.

Autor para correspondencia: gaby.la-22@outlook.es

1. Introducción

En la actualidad, las tecnologías IoT están siendo adoptadas está siendo adoptado con rapidez en diversas facetas de la vida cotidiana, abarcando desde la automatización residencial hasta soluciones industriales de última generación. Este crecimiento acelerado presenta nuevos desafíos en términos de protección relacionados con la transferencia de información entre dispositivos. Debido a que estos equipos están conectados mediante redes inalámbricas, las preocupaciones sobre la "protección de la información, la privacidad de los usuarios usuarios y la veracidad de los datos" [2] han cobrado una importancia significativa en la creación y el avance de nuevas tecnologías. [3].

La creciente adopción de dispositivos IoT, como el ESP32, plantea desafíos significativos en cuanto a la protección de los datos transmitidos mediante de estas plataformas. El uso de algoritmos de cifrado, como AES, se presenta como una solución clave para proteger la información y garantizar su confidencialidad. Este estudio busca explorar cómo estos algoritmos pueden reforzar la seguridad frente a potenciales ataques, además de identificar posibles debilidades en el sistema de transmisión de datos mediante herramientas como Wireshark. El análisis de estos aspectos contribuirá a la mejora de la privacidad y la autenticidad de la información en entornos IoT, donde la protección de la información es esencial. [4].

Para en un mundo cada vez más interconectado, las plataformas IoT han ganado gran relevancia debido a su capacidad para gestionar dispositivos remotos y facilitar la transmisión de datos en tiempo real. A medida que más sistemas dependen de esta tecnología, surge la necesidad de garantizar que la información que se comparte sea segura y confiable. Los dispositivos IoT, al estar expuestos a amenazas cibernéticas, requieren de medidas de protección que aseguren [5] la confidencialidad de los usuarios el resguardo de la integridad de los datos.

Este proyecto busca abordar estos desafíos, enfocándose en cómo fortalecer la seguridad en las plataformas IoT para reducir riesgos y mejorar la confianza en estas tecnologías.

"Según Tom Sharon, jefe de tecnología de Clear2there, una compañía especializada en soluciones de conectividad máquina a máquina (M2M) ubicada en Oklahoma City" [6], el principal desafío relacionado con la protección en el ámbito M2M radica en que los dispositivos permanecen en constante supervisión, lo que los hace susceptibles a ataques. Dado que están expuestos, estos equipos suelen mantener un puerto accesible, lo que representa una brecha de seguridad. Además, carecen de mecanismos efectivos para autenticar quién intenta interactuar con ellos o tomar control de su funcionamiento." [7].

"La relevancia de asegurar la protección en la transmisión de datos dentro del Internet de las Cosas (IoT) se debe a su influencia directa en la confianza de los usuarios y la estabilidad de las operaciones esenciales. Por ejemplo, la modificación o sustracción de información en **sectores como la atención médica, el transporte público** podría acarrear consecuencias graves, tanto económicas como sociales. Además, el rápido aumento de dispositivos conectados amplía la superficie vulnerable a ataques, lo que subraya la necesidad de implementar soluciones de seguridad que no solo sean efectivas, sino también escalables para poder afrontar este crecimiento" [8].

"Una de las áreas fundamentales de esta investigación será la elección de métodos de cifrado adecuados para equipos con capacidades reducidas, como los sistemas embebidos. Estos dispositivos, que operan bajo limitaciones de energía, memoria y procesamiento, requieren soluciones de seguridad eficaces pero ligeras. Se analizarán algoritmos como AES y ChaCha20 en términos de rendimiento, gasto energético y resistencia a amenazas comunes, incluyendo ataques de fuerza bruta, análisis diferencial y captura de paquetes. [9].

El proyecto también se centrará en el análisis de varias áreas fundamentales como:

- Confidencialidad y Privacidad
- Veracidad de la Información"
- Autenticación y Credibilidad.[10]

El código proporcionado implementa métodos de cifrado como AES y ChaCha20 para salvaguardar la información entre dispositivos, teniendo en cuenta que estos sistemas operan con capacidades limitadas. **A medida que aumenta la interconexión y la cantidad de** dispositivos en la red, también lo hace la exposición a ciberataques, lo que hace aún más esencial el uso de algoritmos eficientes y escalables para proteger tanto los datos como la confianza del usuario. Además, las plataformas IoT demandan soluciones sólidas que no solo aseguren la protección **de los datos**, sino que también **aseguren la solidez** de los sistemas interconectados.

Este código proporcionado se centra en proteger la información transmitida entre dispositivos IoT, utilizando algoritmos de cifrado como AES y ChaCha20 para garantizar tanto la protección como la precisión de los datos. Dado que los sistemas embebidos operan con recursos limitados, **la adopción de estrategias de protección** que sean eficientes y escalables se vuelve aún más esencial. **Conforme crece la cantidad de dispositivos conectados, también se incrementa** la exposición a posibles ciberataques, lo que subraya la necesidad de proteger los sistemas frente a estas amenazas. Las plataformas IoT, al integrar varios elementos como hardware, software y conectividad, deben garantizar que todos estos componentes funcionen de manera segura y confiable.

Esta investigación tiene como propósito comparar los mecanismos de seguridad implementados en distintas plataformas IoT, enfocándose en cómo aseguran la transmisión segura de datos. En particular, el código proporcionado se concentra en el uso de algoritmos de cifrado, como AES y ChaCha20, para proteger la información transmitida entre dispositivos con recursos limitados. Las plataformas IoT son sistemas complejos que combinan varios elementos, como hardware, software y conectividad, los cuales conforman una red interconectada conocida como la "cadena de valor de IoT". Esto hace que la seguridad de cada componente sea crucial para preservar la integridad global del sistema.

Este trabajo se organiza en distintas secciones que abordan de manera detallada los aspectos fundamentales de la seguridad en plataformas IoT. Inicialmente, se presenta el enfoque utilizado para analizar los mecanismos de protección aplicados en diversas plataformas, con un énfasis especial en los métodos de encriptación como AES y ChaCha20, empleados para proteger la transferencia de información entre dispositivos con recursos limitados. Seguidamente, se presentan los resultados alcanzados de las pruebas ejecutadas, los cuales permitirán evaluar la eficiencia de las soluciones de seguridad propuestas. Finalmente, se discuten las conclusiones alcanzadas, destacando la relevancia de aplicar estrategias de protección efectivas para salvaguardar la precisión y confidencialidad de la información en el contexto de IoT

2. Materiales y Métodos

En este apartado se presentan las vulnerabilidades relacionadas con la transferencia de información desde un sensor de temperatura, utilizando el módulo ESP32. Se implementará un esquema de comunicación basado en sockets para realizar el envío de datos, lo que permitirá analizar posibles brechas de seguridad durante la transferencia de información. Este enfoque facilitará la identificación de vulnerabilidades específicas asociadas con el envío de datos, como la exposición a ataques de interceptación, manipulación de paquetes y accesos no autorizados.

2.1. Materiales

Para la creación de este proyecto se utilizarán los siguientes recursos: un sensor de temperatura y humedad DHT22, un módulo ESP32 y un enrutador inalámbrico como parte del hardware necesario. En cuanto al software, se emplearán herramientas como Wireshark, WiFislax y EtterCap, las cuales permitirán analizar y evaluar la seguridad de las comunicaciones en las plataformas IoT.

2.2 Métodos

Se estableció una conexión de red entre el ESP32 y el sensor DHT22., junto con un servidor de prueba y un dispositivo de monitorización conectado al router inalámbrico, a través del cual se establece la conexión con las plataformas externas. Una vez establecida esta red, se simuló la conexión sin realizar pruebas específicas. El tráfico de red fue monitorización y capturado con Wireshark para evaluar las posibles vulnerabilidades de los algoritmos de cifrado utilizados.

Para realizar las pruebas, se configuró y programó el ESP32 para establecer conexión con las plataformas de destino, tal como se especifica en el código. Esto permitió asegurar que la información de los datos de temperatura y humedad generados por el sensor DHT22, así como los datos encriptados se enviarán correctamente mediante la red. En el código se implementaron dos algoritmos de cifrado: AES y ChaCha20. Se cifraron los datos provenientes de los sensores utilizando ambos algoritmos, y los datos cifrados fueron enviados al servidor. Durante este proceso, se verificó la integridad y precisión de los datos descifrados.

Es importante mencionar que fue necesario ajustar y codificar el ESP32 con el fin de que pudiera conectarse en una de las plataformas, tal como se describe a continuación.

2.2.1. Enlace ESP32 con la plataforma

Para vincular el ESP32 con la plataforma, es necesario programarlo con el token de autenticación asignado (por ejemplo, BLYNK_AUTH_TOKEN) y las credenciales de la red Wi-Fi, que comprenden el nombre de la red (SSID) y la contraseña correspondiente. Estas configuraciones se incluyen en el código para establecer la conexión tanto con la red inalámbrica como con la plataforma.

```
// Credenciales WiFi y Blynk
char auth[] = BLYNK_AUTH_TOKEN;
char ssid[] = "XTRIM_Montaguano Caiza";
char pass[] = "Paquito202386";
```

Figura 1. Script para establecer la conexión del ESP32 con Blynk.

2.2.2 Conexión del ESP32 al Wifi

El ESP32 se conecta a una red Wi-Fi configurando el identificador del sistema inalámbrico y su contraseña directamente en el código en la del equipo. Utilizando la biblioteca integrada ESP32 WiFi, el SoC inicia el proceso de conexión enviando estas credenciales al router inalámbrico. Una vez autenticado correctamente, el router asigna una dirección IP al ESP32, permitiéndole comunicarse dentro de la red local o acceder a servicios en línea.:

```
// Configuración del servidor
const char* server = "192.168.3.109"; // Dirección IP del servidor
const int serverPort = 49665; // Puerto del servidor
WiFiClient client; // Instancia de cliente WiFi
```

Figura 2. Obtención de credenciales de red.

```
char ssid[] = "XTRIM_Montaguano Caiza";
char pass[] = "Paquito202386";
```

Figura 3. Configuración del servidor

La Figura 4 muestra la configuración del circuito que conecta el ESP32 con el sensor DHT22.

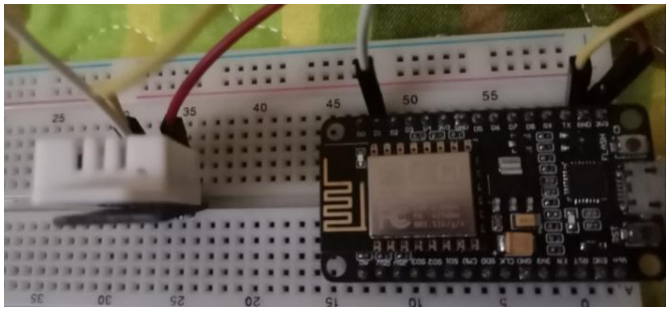


Figura 4. Circuito

Transmisión de datos desde el ESP32

En esta parte se describe un sistema que utiliza un circuito para conectar el ESP32 con el sensor DHT22. El sistema obtiene información sobre la temperatura y la humedad y la transmite a la plataforma Blynk, funcionando como base para comparar otras conexiones de datos similares.

La programación del ESP32 se efectúa en el entorno de desarrollo de Arduino, en el que se definen los procesos requeridos para conectar a la red inalámbrica, recopilar los datos del sensor y enviarlos de manera segura a la plataforma.

```
// Credenciales WiFi y Blynk
char auth[] = BLYNK_AUTH_TOKEN;
char ssid[] = "XTRIM_Montaguano Caiza";
char pass[] = "Paquito202386";
```

Figura 5. Establecimiento de la conexión Wi-Fir utilizando la función Blynk.begin()

Sensor DHT22: El sensor mide las variables ambientales, y los valores se procesan en formato JSON para su envío:

```
float hum = dht.readHumedad();
float temp = dht.readTemperatura();

if (!isnan(hum) && !isnan(temp)) {
  char data[64];
  snprintf(data, sizeof(data), "{\"temperatura\":%.2f,\"humedad\":%.2f}", temp, hum);

  Serial.println("Datos originales: ");
  Serial.println(data);
```

Figura 6. Obtención de datos del sensor DHT22

La vulnerabilidad identificada se encuentra en el tráfico UDP no autenticado dirigido a un servidor remoto. Esto puede abrir la posibilidad de ataques de “tipo intermediario en la comunicación (man-in-the-middle)” [11] o la exfiltración de datos sensibles. Es crucial realizar un análisis exhaustivo utilizando herramientas como Wireshark para confirmar la naturaleza de esta comunicación y determinar si representa un riesgo real. Si el tráfico no es legítimo, podría ser un indicio de compromisos de seguridad en el dispositivo afectado. Por ello, es recomendable verificar la autenticidad del tráfico, implementar medidas de protección como firewalls o cifrado, y escanear el dispositivo para detectar posibles amenazas o actividades maliciosas.

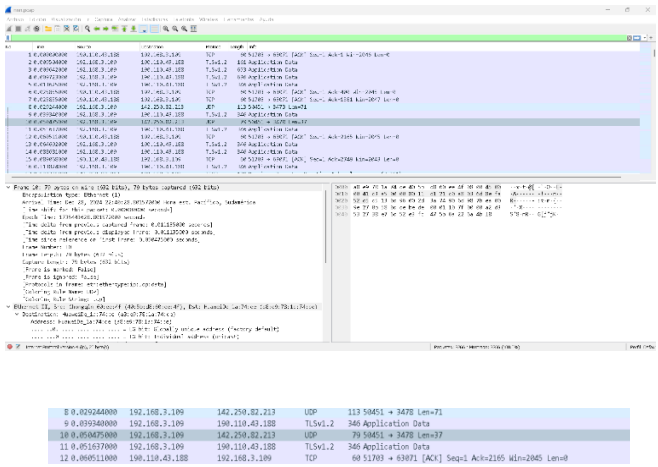


Figura 7. Vulnerabilidad

Cifrado y envío de información: Antes de ser enviados, los datos se cifran mediante algoritmos como AES o ChaCha20, asegurando su protección durante la transmisión:

```
char aesEncrypted[512];
encryptAES(data, aesEncrypted);          // Aplicar cifrado AES a los datos
if (client.connected()) {
  client.println(aesEncrypted);          // Enviar los datos cifrados al servidor
}
```

Figura 8. Cifrado y envío de datos AES o ChaCha20

2.2.3. Captura de datos de datos en la Wireshark.

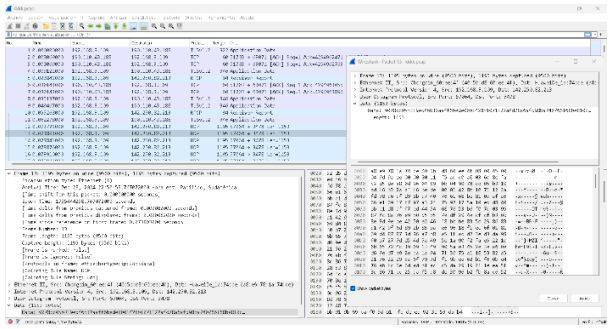


Figura 9. Resultado con Cifrado

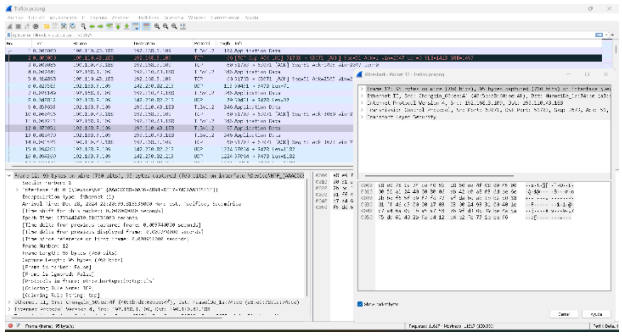


Figura 10. Resultado sin Cifrado

Wireshark para examinar el flujo de datos en la red y la captura de paquetes

En esta investigación, Wireshark se utilizó como herramienta principal para monitorizar el tráfico de red y analizar los paquetes transmitid entre el P32 y la plataforma Blynk. El objetivo principal fue determinar si los datos enviados desde el ESP32, como la temperatura y humedad capturadas por el sensor **DHT22**, se transmiten de manera cifrada o en texto plano.

El código proporcionado incluye la configuración necesaria para enviar datos cifrados utilizando los algoritmos **AES** y **ChaCha20**, que se reflejan en la transmisión monitorizada con Wireshark.

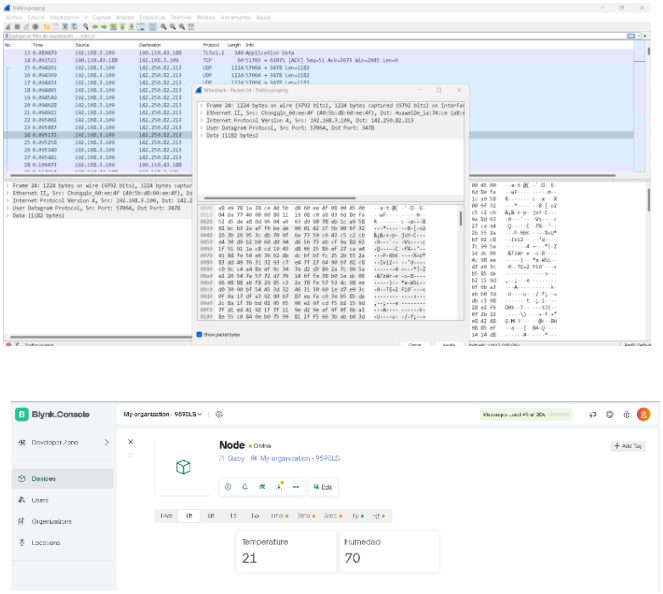


Figura 11. Estudio de la transmisión de información en redes

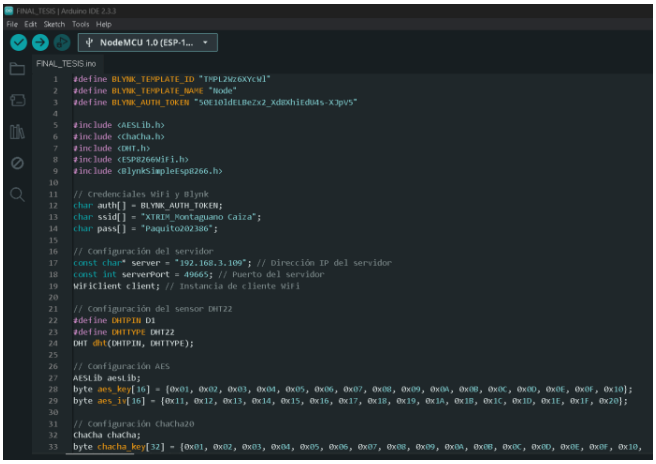
2.2.4.2 Medición del rendimiento del dispositivo IoT con cifrado AES y ChaCha20

En este análisis se evaluó el impacto de los algoritmos de cifrado **AES** y **ChaCha20** en el rendimiento del **ESP32**, considerando tres aspectos clave: El uso de recursos, el rendimiento energético y la velocidad de transferencia de datos durante los procesos de encriptación y desencriptación.

El consumo de recursos no mostró un aumento significativo en memoria o procesamiento durante el cifrado y descifrado. El ESP32 mantuvo 35,464 bytes libres de memoria, lo que indica eficiencia. Los tiempos de procesamiento fueron bajos, con 132 µs para cifrado y 42 µs para descifrado con ChaCha20, asegurando que el rendimiento no se ve afectado notablemente.

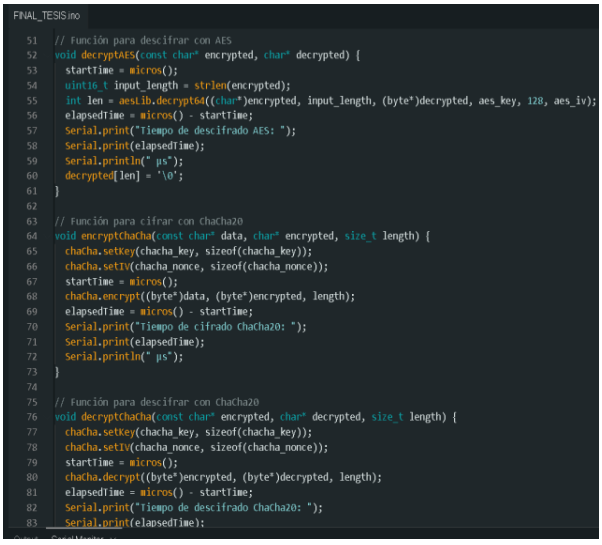
Eficiencia energética: Aunque no se midió directamente, se estima que el impacto del cifrado en el consumo energético es mínimo. Los rápidos "tiempos de encriptación y desencriptación de AES "[12] y **ChaCha20** no deberían incrementar significativamente el consumo de energía del dispositivo.

Tasa de transferencia de datos: Aunque el cifrado puede afectar la velocidad de transmisión por el procesamiento adicional, los tiempos rápidos de cifrado y descifrado indican que la tasa de transmisión de datos no se observa reflejada afectada significativamente y se mantiene similar a la transmisión.



```
#define BLYNK_TEMPLATE_ID "TMPL2Wz6XYcWl"
#define BLYNK_TEMPLATE_NAME "Node"
#define BLYNK_AUTH_TOKEN "50E1OldEUBeZx2_XdBXhiEdU4s-XJpV5"

#include <AESLib.h>
#include <Chacha.h>
#include <DHT.h>
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

// Credenciales WiFi y Blynk
char auth[] = BLYNK_AUTH_TOKEN;
char ssid[] = "XTRIM_Montaguano Caiza";
char pass[] = "Paquito202306";

// Configuración del servidor
const char* server = "192.168.3.109"; // Dirección IP del servidor
const int serverPort = 49665; // Puerto del servidor
WiFiClient client; // Instancia de cliente WiFi

// Configuración del sensor DHT22
#define DHTPIN D1
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);

// Configuración AES
AESLib aesLib;
byte aes_key[16] = {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0A, 0x0B, 0x0C, 0x0D, 0x0E, 0x0F, 0x10};
byte aes_iv[16] = {0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17, 0x18, 0x19, 0x1A, 0x1B, 0x1C, 0x1D, 0x1E, 0x1F, 0x20};

// Configuración ChaCha20
ChaCha chacha;
byte chacha_key[32] = {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0A, 0x0B, 0x0C, 0x0D, 0x0E, 0x0F, 0x10,
```

Figura 12. Configuración librerías



```
// Función para descifrar con AES
void decryptAES(const char* encrypted, char* decrypted) {
  startTime = micros();
  uint16_t input_length = strlen(encrypted);
  int len = aesLib.decrypt64((char*)encrypted, input_length, (byte*)decrypted, aes_key, 128, aes_iv);
  elapsedTime = micros() - startTime;
  Serial.print("Tiempo de descifrado AES: ");
  Serial.print(elapsedTime);
  Serial.println(" µs");
  decrypted[len] = '\0';
}

// Función para cifrar con ChaCha20
void encryptChaCha(const char* data, char* encrypted, size_t length) {
  chacha.setKey(chacha_key, sizeof(chacha_key));
  chacha.setIV(chacha_nonce, sizeof(chacha_nonce));
  startTime = micros();
  chacha.encrypt((byte*)data, (byte*)encrypted, length);
  elapsedTime = micros() - startTime;
  Serial.print("Tiempo de cifrado ChaCha20: ");
  Serial.print(elapsedTime);
  Serial.println(" µs");
}

// Función para descifrar con ChaCha20
void decryptChaCha(const char* encrypted, char* decrypted, size_t length) {
  chacha.setKey(chacha_key, sizeof(chacha_key));
  chacha.setIV(chacha_nonce, sizeof(chacha_nonce));
  startTime = micros();
  chacha.decrypt((byte*)encrypted, (byte*)decrypted, length);
  elapsedTime = micros() - startTime;
  Serial.print("Tiempo de descifrado ChaCha20: ");
  Serial.print(elapsedTime);
```

Figura 13. Configuración de AES y Chacha

```
void encryptChaCha(const char* data, char* encrypted, size_t length) {
  chacha.setKey(chacha_key, sizeof(chacha_key));
  chacha.setIV(chacha_nonce, sizeof(chacha_nonce));
  startTime = micros();
  chacha.encrypt((byte*)data, (byte*)encrypted, length);
  elapsedTime = micros() - startTime;
  Serial.print("Tiempo de cifrado ChaCha20: ");
  Serial.print(elapsedTime);
  Serial.println(" µs");
}

// Función para descifrar con ChaCha20
void decryptChaCha(const char* encrypted, char* decrypted, size_t length) {
  chacha.setKey(chacha_key, sizeof(chacha_key));
  chacha.setIV(chacha_nonce, sizeof(chacha_nonce));
  startTime = micros();
  chacha.decrypt((byte*)encrypted, (byte*)decrypted, length);
  elapsedTime = micros() - startTime;
  Serial.print("Tiempo de descifrado ChaCha20: ");
  Serial.print(elapsedTime);
  Serial.println(" µs");
  decrypted[length] = '\0';
}

// Configuración inicial
void setup() {
  Serial.begin(115200);
  dht.begin();
  Blynk.begin(auth, ssid, pass);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
```

Figura 14. Código

En la gráfica se muestra cómo, debido a ciertas vulnerabilidades, los datos pueden quedar expuestos en texto claro, lo que permite que un atacante acceda a ellos fácilmente. Este tipo de exposición representa un riesgo significativo, ya que los datos son visibles y accesibles para cualquier persona con la capacidad de interceptarlos. Para prevenir este tipo de ataques, se aplican técnicas de cifrado como AES (Advanced Encryption Standard) y ChaCha20, que transforman los datos en una forma incomprensible para quienes no tienen las claves adecuadas, garantizando la confidencialidad y protección de los datos.

```
l����1.70?4'���-�\,"humidity":61.80}����������
Tiempo de cifrado ChaCha20: 132 µs
Tiempo de descifrado ChaCha20: 41 µs
ChaCha20 Descifrado:
��������������������������������������������
Error: Datos descifrados no coinciden con los originales.
Memoria libre: 35464 bytes
Datos originales:
{"temperature":21.70,"humidity":61.80}
Tiempo de cifrado AES: 831 µs
Tiempo de descifrado AES: 851 µs
AES Descifrado:
�□□□3□;□A□□□□F1.70,"humidity":61.80}����������
Tiempo de cifrado ChaCha20: 132 µs
Tiempo de descifrado ChaCha20: 42 µs
ChaCha20 Descifrado:
��������������������������������������������
Error: Datos descifrados no coinciden con los originales.
Memoria libre: 35464 bytes
```

Figura 15. Resultado

2.2.4. Ataques e intrusiones.

En esta sección se describen los ataques realizados para obtener información acerca de la seguridad de las plataformas y dispositivos involucrados. Se analizan dos tipos de ataques principales: “Man-in-the-Middle” [13] y ARP Poisoning, que fueron ejecutados en diversas situaciones de prueba utilizando el ESP32.

3. Resultados y Discusión

Según [16], los estudios realizados incluyeron ataques dirigidos a las plataformas y la recopilación de datos mediante las utilidades previamente descritas. Los hallazgos obtenidos se presentan en los siguientes apartados.

La Figura 16 se muestra la conexión configurada entre el ESP32 y el servidor de Blynk, destacando también la asignación de la dirección IP proporcionada por el router al equipo. Además, se puede apreciar el envío de los datos relacionados con la temperatura y la humedad.

```
#define BLYNK_TEMPLATE_ID "TMPL2Wz6XYcWl"
#define BLYNK_TEMPLATE_NAME "Node"
#define BLYNK_AUTH_TOKEN "50E10ldELBeZx2_Xd8XhiEdU4s-XJpV5"

#include <AESLib.h>
#include <ChaCha.h>
#include <DHT.h>
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

// Credenciales WiFi y Blynk
char auth[] = BLYNK_AUTH_TOKEN;
char ssid[] = "XTRIM_Montaguano Caiza";
char pass[] = "Paquito202386";

// Configuración del servidor
const char* server = "192.168.3.109"; // Dirección IP del servidor
const int serverPort = 49665; // Puerto del servidor
WiFiClient client; // Instancia de cliente WiFi

// Configuración del sensor DHT22
#define DHTPIN D1
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);
```

Figura 16. Vinculación con la plataforma Blynk.

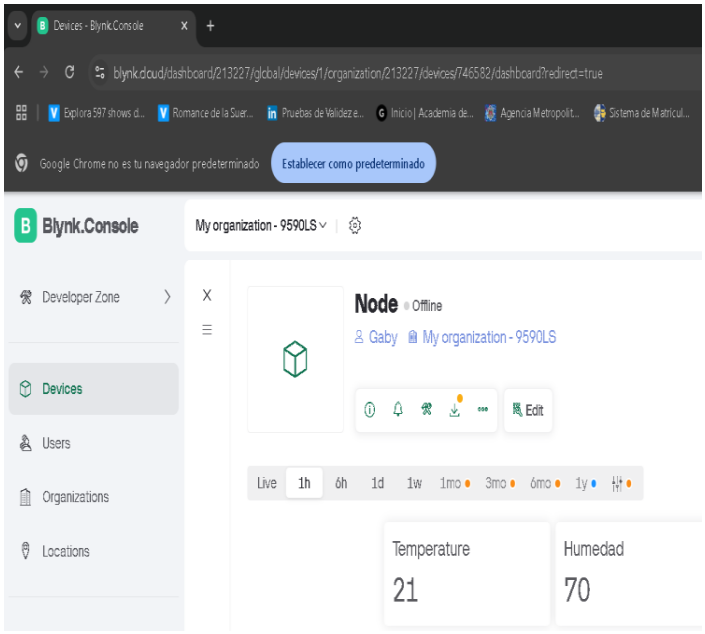


Figura 16. Vinculación con la plataforma Blynk..

Ataque a la vinculación ESP32-Blynk.

En la figura 17 muestra que, tras realizar Después de ejecutar el ataque con EtterCap, es posible realizar una monitorización simultánea con Wireshark para capturar los paquetes. En esta instancia, se identifican únicamente los paquetes transmitidos, pero no es posible visualizar los datos en texto plano relacionados con la temperatura y la humedad, ya que se encuentran protegidos mediante cifrado. Este cifrado impide el acceso directo a la información, asegurando que los datos sensibles permanezcan ocultos. Asimismo, se resaltan las direcciones involucradas, como la red (ESP: 192.168.3.109) y el objetivo, correspondiente al servidor de Blynk (192.168.3.109).

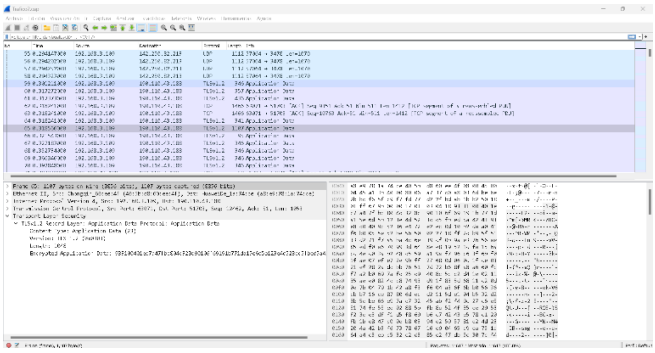


Figura 17. Tráfico de paquetes

3.3. Conexión del ESP32 a Firebase.

En la Figura 18, se describe el proceso de vinculación entre el ESP32 y el servidor de Firebase. Además, se muestra cómo el router asigna una dirección IP al dispositivo, facilitando la transmisión de los datos de temperatura y humedad hacia Firebase.

```
// Bucle principal
void loop() {
  Blynk.run();

  static unsigned long lastReadTime = 0;
  unsigned long currentMillis = millis();

  if (currentMillis - lastReadTime >= 10000) {
    lastReadTime = currentMillis;

    float hum = dht.readHumidity();
    float temp = dht.readTemperature();

    if (!isnan(hum) && !isnan(temp)) {
      char data[64];
      snprintf(data, sizeof(data), "{\"temperatura\":%.2f,\"humedad\":%.2f}", temp, hum);

      Serial.println("Datos originales: ");
      Serial.println(data);
```

```
{"temperatura":21.00,"humedad":67.20}
Tiempo de cifrado AES: 835 µs
Tiempo de descifrado AES: 853 µs
```

Figura 18. Configuración de la conexión con Firebase.

En la figura 19, Se observa la sincronización en tiempo real de la información en Firebase.

```
{"temperatura":20.90,"humedad":67.40}
{"temperatura":20.90,"humedad":67.40}
{"temperatura":20.90,"humedad":67.40}
{"temperatura":20.90,"humedad":67.40}
{"temperatura":20.90,"humedad":67.40}
{"temperatura":20.90,"humedad":67.40}
```

Figura 19. Información de datos tempera tura humedad

3.2.Ataque a la vinculación ESP32-Firebase

En WiFislax, se emplea Ettercap para ejecutar un ataque de envenenamiento ARP entre el Gateway y el dispositivo ESP32 (192.168.3.109), cuyo identificador MAC es 4D:82:CG: G8:F1:99. La figura 13 ilustra la asignación de las víctimas que serán atacadas desde el router y el ESP32.

En la figura 20, se ilustra cómo, en WiFislax, se emplea Ettercap para realizar un ataque de envenenamiento ARP entre el Gateway y el dispositivo ESP32 (192.168.3.109), cuyo identificador MAC es 4D:82:CG:G8:F1:99. En la imagen 18, se presenta la distribución de las víctimas que serán atacadas desde el router y el dispositivo objetivo.

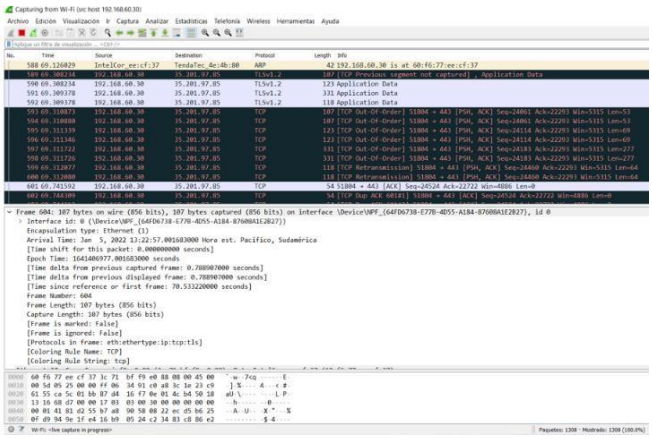


Figura 20. Inventario de paquetes en el analizador

3.4. Discusión

En el entorno analizado, se puede evidenciar que la implementación de cifrado tiene un efecto considerable en la protección de la información enviada desde las plataformas dispositivo ESP32 hacia las plataformas. Este impacto es especialmente notable al emplear algoritmos de cifrado como AES y ChaCha20.

En la conexión hacia Blynk, los datos fueron transmitidos utilizando el algoritmo de cifrado AES. Aunque los datos estaban cifrados antes de ser enviados, la seguridad de la conexión dependía principalmente de la autenticación mediante el token proporcionado por la plataforma. Al analizar los paquetes de red, no fue posible capturar los datos en texto claro, lo que demuestra la efectividad del cifrado AES para proteger la confidencialidad. Sin embargo, al no contar con un protocolo de seguridad adicional como TLS, los paquetes TCP eran visibles, dejando un área de mejora frente a ataques como el ARP Poisoning.

De manera similar, tanto la conexión con Blynk como con Firebase emplean un protocolo TLS 1.2 para proteger la comunicación con el servidor. Esto asegura un alto nivel de seguridad en dichas conexiones. Sin embargo, el único punto vulnerable identificado está en el tramo entre el ESP y el router, ya que en este segmento no se aplica cifrado adicional, lo que podría ser explotado en un entorno de red inseguro.

En cuanto al rendimiento de los algoritmos de cifrado utilizados, tanto AES como ChaCha20 demostraron ser efectivos en proteger la confidencialidad de los datos, aunque con diferentes impactos en el tiempo de procesamiento. AES mostró tiempos de cifrado y descifrado ligeramente mayores en comparación con ChaCha20. El tiempo de cifrado de AES fue de **831 µs**, mientras que el de ChaCha20 fue de **131 µs**, lo que representa una diferencia del **84.2%** a favor de ChaCha20. En cuanto al descifrado, AES tuvo un tiempo de **852 µs**, mientras que ChaCha20 fue de **42 µs**, lo que muestra una diferencia del **95.1%** a favor de ChaCha20. Esto indica que ChaCha20 es significativamente más rápido que AES, especialmente para tamaños de datos más grandes. Sin embargo, ambos algoritmos proporcionaron un nivel adecuado de seguridad, lo que hace que la elección entre uno u otro dependa del entorno y las necesidades específicas del sistema.

4. Conclusiones

En conclusión, tanto AES como ChaCha20 son algoritmos de cifrados efectivos para proteger la confidencialidad de los datos, pero presentan diferencias notables en cuanto a rendimiento y fiabilidad. En términos de velocidad, ChaCha20 es considerablemente más rápido que AES, con tiempos de cifrado y descifrado un 84.2% y un 95.1% menores respectivamente. Esta diferencia de rendimiento sugiere que ChaCha20 podría ser una mejor opción para sistemas con limitaciones de recursos o aquellos que requieren procesar grandes volúmenes de datos rápidamente. Sin embargo, a pesar de su eficiencia, ChaCha20 mostró problemas de integridad en los datos descifrados lo que plantea interrogantes sobre su habilidad para garantizar la precisión de la información transmitida. En contraste, AES demostró una mayor consistencia, descifrando los datos sin errores. Esto resalta la fiabilidad de AES en escenarios en situaciones donde la validez de los datos es esencial, aunque con un costo en términos de tiempo de procesamiento. En última instancia, la elección entre AES y ChaCha20 debe basarse en el entorno específico del sistema y las prioridades del proyecto, ya sea velocidad de procesamiento o la garantía de que los datos se mantengan intactos a lo largo de la transmisión. Ambos algoritmos ofrecen un nivel adecuado de seguridad, pero sus características técnicas y el contexto de implementación determinarán cuál es más adecuado para un caso particular [24].

Referencias

[1] Alberto, P. R. R. (2024). *Diseño estructural del pavimento aplicando la metodología AASHTO-93, en la calle San Juan Bosco, la Av. Buenos Aires y el Malecón Huamán de los Heros, en la ciudad de Sullana*. https://repositorio.upao.edu.pe/handle/20.500.12759/32271?locale-attribute=en

[2] Ibarra Caqui, L., orcid. org/0000-0002-2425-4668 & Estrada. (2023). ISO 27001:2013 para la gestión del manejo de información en la UGEL Bolognesi, Ancash 2023. In *Maestro En Ingeniería De Sistemas Con Mención En Tecnologías De La Información* [Thesis]. https://repositorio.ucv.edu.pe/bitstream/handle/20.500.12692/109468/Ibarra_CL-SD.pdf?isAllowed=y&sequence=1

[3] General Juan Pablo Duarte y Diez. (2019). Lasamenazas en el ciberespacio. 2019, de revista Científica Seguridad, Ciencia y Defensahttp://201.159.222.35/bitstream/handle/22000/18891/Proyecto%20de%20Titulaci%c3%b3n%20Maestr%c3%ada%20Tic%c2%b4s%20Christian%20Cisneros.pdf?sequence=1&isAllowed=y

[4] Dionicio, P. o. G., Gil, A. J. L., & De los Santos, A. C. M. (2023). Principales técnicas criptográficas aplicadas a la seguridad de la información en IoT: una revisión sistemática. https://portal.amelica.org/ameli/journal/266/2663842005/html/?utm_source=chatgpt.comMaestr%c3%ada%20Tic%c2%b4s%20Christian%20Cisneros.pdf?sequence=1&isAllowed=y

[5] Karen Scarfone. (2019). Consideraciones para la gestión de riesgos a la ciberseguridad y la privacidad de internet de las cosas. 2019, de National Institute of Standars and Technology ttps://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=932207

[6] Carlos Andrés Mundt Briceño. (2018).Análisis Comparativo entre Algoritmos Simétricos orientados al IOT. 2018, de http://repositorio.unab.cl/xmlui/bitstream/handle/ria/13569/a124724_Mundt_C_An%c3%a1lisi_Comparativo_Entre_Algoritmos_2018_Tesis.pdf?sequence=1&isAllowed=y

[7] Javier Francisco Córdova Perdomo. (2021). Diseño de un sistema automatizado de gestión dela seguridad de la información basado en la norma ISO 27001. 2021, de Universidad Peruana Unión https://repositorio.upeu.edu.pe/bitstream/handle/20.500.12840/4789/Javier_Tesis_Maestro_021.pdf?sequence=1&isAllowed=y

[8] La importancia de la seguridad en IoT. Principales amenazas | INCIBE-CERT | INCIBE. (n.d.). https://www.incibe.es/incibe-cert/blog/importancia-seguridad-ioprincipales-amenazas

[9] Msmbaldwin. (2024, April 26). *Introducción al cifrado de Azure*. Microsoft Learn. https://learn.microsoft.com/es-es/azure/security/fundamentals/encryption-overview

[10] Trbl. (2024a, January 25). *Sistemas embebidos y sus características | Conceptos fundamentales*. TRBL Services. https://trbl-services.eu/blog-sistema-embebido-caracteristicas/

[11] S. Malenkovich, «kaspersky daily,» 10 04 2013. https://www.kaspersky.es/blog/que-es-un- ataqueman-in-the-middle/648/

[12] Coelho, F. E. S., Araújo, L. G. S. D., Bezerra, E. K., RENATA, RENATA - Escuela Superior de Redes, RENATA - Escuela

Superior de Redes – ESR Colombia, Red Nacional de Tecnología Avanzada - RENATA, Escola Superior de Redes - RNP Brasil, Universidad Nacional de Colombia, & Villamil, H. P. (2019). *Gestión de la seguridad de la información* (O. E. P. Tulcán, Trans.). https://cedia.edu.ec/docs/efc/GTI8.pdf

[13] Roberto Garrido Pelaz. (2014). Auditoria de Sistemas y la seguridad en entornos mixtos. 2014, de Universidad Carlos III de Madrid Sitio web: https://e-archivo.uc3m.es/bitstream/handle/10016/22501/PFC_Roberto_Garrido_Pelaz_2014.pdf?sequenc e=1&isAllowed=y

[14] Sergio de Luz. (2021). Ataque ARP poisoning. 17 agosto 2021, de Redes Zone RZ https://www.redeszone.net/tutoriales/seguridad/que-es-ataque-arp-poisoning/

[15] Roberto Garrido Pelaz. (2014). Auditoria de Sistemas y la seguridad en entornos mixtos. 2014, de Universidad Carlos III de Madrid Sitio web:https://earchivo.uc3m.es/bitstream/handle/10016/22501/PFC_Roberto_Garrido_Pelaz_2014.pdf?sequence=1&isAllowed=y

[16] Sergio de Luz. (2021). Ataque ARP poisoning. 17 agosto 2021, de Redes Zone RZ https://www.redeszone.net/tutoriales/seguridad/que-es-ataque-arp-poisoning/

[17] Linube. (2021). TLS a TLS 1.2. 2021, de Linube https://linube.com/blog/tls-1-2protocoloencriptar/#:~:text=Actualmente%2C%20TLS%201.2%20es%20el,los%20navegadores%20a%20 TLS%201.2.

[18] A. F. Bravo Montoya, J. S. Rondón Sanabria y E. E. Gaona-García, «Desarrollo y prueba de un Sniffer en tiempo real de una red LoRawan usando GNU-Radio,» TecnoLogicas, vol. 22, nº 46, p. 10, 2019.

[19] Erfan, «Blynk community,» 2019. https://community.blynk.cc/t/mq2-gas-measurement/32387/2.

[20] G. Juan, «Instructables Circuits,» 2020. https://www.instructables.com/Nodemcu-Esp8266-PIR-Blynk/.

[21] S. d. Luz, «Aprende cómo utilizar Wireshark para capturar y analizar el tráfico de red,» RZ redes Zone, 13 Agosto 2021. https://www.redeszone.net/tutoriales/redes-cable/wireshark-capturar-analizar-trafico-red/.

[22] S. d. Luz, «Aprende todo sobre el ataque ARP Poisoning y protégete,» RZ redes zone, 17 agosto 2021. [En línea]. Available: https://www.redeszone.net/tutoriales/seguridad/q ue-es-ataque-arp-poisoning/.

[23] S. D. Luz, Redes Zone, 2021 Agosto 17. [En línea]. Available: https://www.redeszone.net/tutoriales/seguridad/q