UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS CARRERA DE TELECOMUNICACIONES

TRABAJO DE INTEGRACIÓN CURRICULAR

TEMA:

"MECANISMO DE SEGURIDAD ADAPTATIVA PARA ENTORNOS IOT BASADO EN LA SELECCIÓN DINÁMICA DE ALGORITMOS CRIPTOGRÁFICOS LIGEROS"

Trabajo de titulación previo a la obtención del título de Ingeniero en Telecomunicaciones

Línea de investigación: Desarrollo, aplicación de software y cibersecurity (seguridad cibernética)

AUTOR:

Esteban Marcelo Vizcaíno Gudiño

DIRECTOR:

MSc. Fabián Geovanny Cuzme Rodríguez

Ibarra - Ecuador, 2026

UNIVERSIDAD TÉCNICA DEL NORTE
DIRECCIÓN DE BIBLIOTECA

AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información:

DATOS DE CONTACTO	
APELLIDOS Y NOMBRES:	VIZCAINO GUDIÑO ESTEBAN MARCELO

DATOS DE LA OBRA	
TÍTULO:	MECANISMO DE SEGURIDAD ADAPTATIVA PARA ENTORNOS IOT BASADO EN LA SELECCIÓN DINÁMICA DE ALGORITMOS CRIPTOGRÁFICOS LIGEROS
AUTOR (ES):	ESTEBAN MARCELO VIZCAINO GUDIÑO
FECHA: DD/MM/AAAA	28 de julio del 2026
SOLO PARA TRABAJOS DE GRADO	
PROGRAMA:	■ **PREGRADO** □ **POSGRADO**
TITULO POR EL QUE OPTA:	INGENIERO EN TELECOMUNICACIONES
DIRECTOR /ASESOR:	MSC. FABIÁN GEOVANNY CUZME RODRÍGUEZ MSC. HENRY PATRICIO FARINANGO ENDARA

2. CONSTANCIAS

El autor (es) manifiesta (n) que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto la obra es original y que es (son) el (los) titular (es) de los derechos patrimoniales, por lo que asume (n) la responsabilidad sobre el contenido de la misma y saldrá (n) en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 28 días del mes de julio de 2026

EL AUTOR:

Validar únicamente en FirmaEC.
Firmado electrónicamente por:
ESTEBAN MARCELO VIZCAINO GUDINO

Esteban Marcelo Vizcaíno Gudiño

CERTIFICACIÓN DEL DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

Ibarra, 28 de julio de 2026

MSC. FABIÁN GEOVANNY CUZME RODRÍGUEZ
DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

CERTIFICA:

Haber revisado el presente informe final del trabajo de Integración Curricular, el mismo que se ajusta a las normas vigentes de la Universidad Técnica del Norte; en consecuencia, autorizo su presentación para los fines legales pertinentes.

Validar únicamente en FirmaEC.
Firmado electrónicamente por:
FABIAN GEOVANNY CUZME RODRIGUEZ

……
MSC. Fabián Geovanny Cuzme Rodríguez
C.C.: 1311527012

DEDICATORIA

A mi familia, por ser el pilar fundamental en mi vida y el motor que me ha impulsado a lo largo de toda mi formación. Por su apoyo incondicional, su paciencia infinita y los invaluables sacrificios que han realizado para que pudiera superar cada obstáculo y alcanzar esta meta. Su confianza constante en mis capacidades me ha permitido llegar hasta aquí; este logro es tan suyo como mío.

A todas las personas que creyeron en mí y me brindaron su aliento en los momentos de mayor exigencia y desafío.

Esteban Marcelo Vizcaíno Gudiño

AGRADECIMIENTO

A mi universidad y a la carrera de Telecomunicaciones, por haberme brindado el espacio, los recursos y las herramientas necesarias para crecer personal y profesionalmente, formándome con las bases científicas y técnicas que hoy sostienen esta investigación.

Un profundo y sincero agradecimiento a mi director y a mi asesor de tesis, por su invaluable guía, su tiempo y la experiencia compartida. Su rigor técnico, sus oportunas correcciones y su visión fueron elementos determinantes para estructurar, orientar y culminar este proyecto con éxito.

A mis docentes, por su vocación de enseñanza y compromiso con la excelencia académica a lo largo de todos estos años.

Finalmente, a mis amigos y compañeros de estudio, por las jornadas compartidas, la colaboración mutua y el aliento constante que hicieron de esta etapa universitaria una experiencia enriquecedora e inolvidable.

Esteban Marcelo Vizcaíno Gudiño

RESUMEN

El crecimiento del Internet de las Cosas (IoT) plantea desafíos críticos de ciberseguridad debido a las severas restricciones de recursos en los dispositivos embebidos. La aplicación de esquemas criptográficos estáticos resulta ineficiente o vulnerable. Este trabajo presenta el diseño y validación de una arquitectura de seguridad adaptativa para entornos IoT, basada en la selección dinámica de algoritmos criptográficos ligeros. Operando bajo un modelo cliente-servidor sobre el protocolo MQTT, un Nodo Maestro orquesta la seguridad mediante un Motor de Decisión inteligente de *Edge AI*. Este motor integra un modelo de Lógica Difusa (con reglas colapsadas al 95%) y una Red Neuronal Artificial implementada en modo *Bare Metal*. Además, una Capa de Supervisión fuerza descensos de nivel de seguridad (*downgrades*) ante condiciones de estrés extremo de hardware, y se utiliza un mecanismo de aprovisionamiento seguro (*PSK Bootstrapping*) basado en XOR y Base64. La validación experimental en un Orange Pi 5 Plus y microcontroladores ESP32 demostró latencias de inferencia puras de 19.41 µs (Lógica Difusa) y 54.57 µs (Red Neuronal), garantizando la confidencialidad con un impacto en memoria inferior a 1 KB.

Palabras clave: Internet de las Cosas (IoT), Seguridad Adaptativa, Criptografía Ligera, Inteligencia Artificial en el Borde (*Edge AI*), Lógica Difusa, Red Neuronal Artificial, MQTT.

ABSTRACT

The rapid growth of the Internet of Things (IoT) presents critical cybersecurity challenges due to the severe hardware constraints of embedded devices. Statically applied cryptographic schemes often prove to be inefficient or vulnerable. This research describes the design and validation of an adaptive security architecture for IoT environments, based on the dynamic selection of lightweight cryptographic algorithms. Operating on a client-server model over the MQTT protocol, a Master Node orchestrates security policies using an *Edge AI* Decision Engine. This engine integrates a Fuzzy Logic model (with rules collapsed by 95%) and a Bare Metal Artificial Neural Network. Furthermore, a Technical Supervision Layer forces security level downgrades under extreme hardware stress, and a secure provisioning mechanism (*PSK Bootstrapping*) based on XOR and Base64 is used for key distribution. Experimental validation on an Orange Pi 5 Plus and ESP32 microcontrollers demonstrated pure inference latencies of just 19.41 µs (Fuzzy Logic) and 54.57 µs (Neural Network), ensuring key confidentiality with a RAM memory impact of less than 1 KB.

Keywords: Internet of Things (IoT), Adaptive Security, Lightweight Cryptography, Edge AI, Fuzzy Logic, Artificial Neural Network, MQTT.

LISTA DE CONTENIDOS

1. Capítulo I: Antecedentes 15

1.1 Tema 15

1.2 Problema 15

1.3 Objetivos 17

1.3.1. Objetivo General 17

1.3.2. Objetivos Específicos 18

1.4 Alcance 18

1.5 Justificación 21

2. Capítulo II: Fundamentación teórica 23

2.1 Internet de las Cosas (IoT) 23

2.1.1. Arquitectura general y limitaciones en dispositivos de bajo recurso tecnología arquitectura hardware y software 24

2.2 Seguridad en entornos IoT 28

2.2.1. Principios de seguridad: confidencialidad, integridad y disponibilidad 28

2.2.2. Amenazas comunes 29

2.2.3. Concepto de seguridad adaptativa 31

2.2.4. Mecanismos de ajuste dinámico y gestión desde nodo maestro 33

2.3 Criptografía ligera aplicada a IoT 36

2.3.1. Definición y ventajas frente a algoritmos tradicionales 36

2.3.2. Clasificación de Algoritmos Criptográficos Ligeros 39

2.4 Selección dinámica de algoritmos criptográficos 40

2.4.1. Esquemas dinámicos 41

2.4.2. Retos y beneficios de la asignación adaptativa 43

3. Capítulo III: Diseño e implementación 46

3.1 Metodología 46

3.2 Fase de Requerimientos 47

3.2.1. Requerimientos de Stakeholders (StSR) 48

3.2.2. Requerimientos de Sistema (SySR) 49

3.2.3. Requerimientos de Arquitectura (Software y Hardware) (SrAH) 51

3.2.4. Criterios de selección de la suite criptográfica 53

3.2.5. Especificaciones Técnicas del Entorno de Hardware y Software 56

3.2.6. Modelado y Análisis del Consumo Energético en el Borde 60

3.3 Etapa de Diseño del Sistema 67

3.3.1. Arquitectura Lógica del Sistema 67

3.3.2. Módulos Principales del Sistema 69

3.3.3. Diagramas de Diseño 71

3.4 Etapa de Implementación 75

3.4.1. Entorno de desarrollo 75

3.4.2. Implementación de los módulos 77

3.4.3. Validación preliminar del prototipo 85

4. Capítulo IV: Pruebas de Funcionamiento 87

4.1 Validación Teórica de Modelos de Inferencia (Fase de Prototipado) 90

4.2 Evaluación de Estrés Computacional y Latencia (CP-02) 91

4.3 Comportamiento ante Escenarios Críticos y Matriz de Riesgos (CP-03) 95

4.3.1. Escenarios Críticos 95

4.3.2. Matriz de Riesgos 96

4.4 Validación del Mecanismo de Aprovisionamiento Seguro (CP-04) 99

4.5 Análisis de Tráfico y Auditoría de Red (CP-05) 101

4.5.1. Configuración del Entorno de Auditoría 101

4.5.2. Vulneración de la Capa de Enlace (WPA2) 101

4.5.3. Análisis del Flujo de Aplicación y Prueba de Robustez 102

4.6 Evaluación de Coexistencia y Degradación Espacial en Entorno Real (CP-06) 104

4.6.1. Descripción de los Escenarios Topológicos 104

4.6.2. Resultados de la Evaluación Espacial 106

4.6.3. Discusión de Resultados: Divergencia entre Modelos de Inferencia 109

5. Conclusiones 111

6. Recomendaciones 113

7. Bibliografía 115

Anexo A - Tabla Comparativa de Algoritmos Criptográficos Ligeros 119

LISTA DE TABLAS

Tabla 1 Comparativa de protocolos 26

Tabla 2 Amenazas en entornos IoT 31

Tabla 3 Comparativa de enfoques dinámicos en criptografía ligera para IoT 45

Tabla 4 Nomenclatura de los requerimientos 47

Tabla 5 Requerimientos de Stakeholders 48

Tabla 6 Requerimientos de sistema 49

Tabla 7 Requerimientos de arquitectura 52

Tabla 8 Requisitos mínimos para el Nodo Maestro 57

Tabla 9 Requisitos mínimos para los Nodos Sensores 58

Tabla 10 Perfil de potencia eléctrica de los Nodos Cliente por estado operativo 65

Tabla 11 Desglose del costo energético en Julios por ciclo de transmisión seguro de 10 segundos 65

Tabla 12 Análisis de Consumo Energético en el Nodo Maestro 66

Tabla 13 Umbrales de decisión para la asignación de algoritmos criptográficos 83

Tabla 14 Matriz de casos de prueba y ruta de validación del sistema 87

Tabla 15 Análisis de respuestas en entorno de pruebas ante inyección manual de telemetría 90

Tabla 16 Análisis estadístico de la prueba de estrés de ia (muestra filtrada al percentil 99) 92

Tabla 17 Comportamiento del motor de decisión inteligente ante escenarios operativos críticos 95

Tabla 18 Matriz de riesgos y vulnerabilidades asumidas por categoría de asignación criptográfica 97

Tabla 19 Resultados escenario 1 - equidistancia y línea de vista (2 metros) 106

Tabla 20 Resultados escenario 2 - atenuación escalonada espacial (5m, 10m, 15m) 107

Tabla 21 Resultados escenario 3 - movilidad controlada y transición dinámica 108

LISTA DE FIGURAS

Figura 1 Árbol de problemas 17

Figura 2 Arquitectura del Sistema Iot 20

Figura 3 Estructura de un sistema IoT 25

Figura 4 Arquitectura lógica del sistema 68

Figura 5 Diagrama de secuencia - Registro y asignación de algoritmo 72

Figura 6 Diagrama de secuencia - Envío de datos y monitoreo 73

Figura 7 Diagrama de flujo del motor de decisión. 74

Figura 8 Diagrama de secuencia del mecanismo de robustez (Callback MQTT). 78

Figura 9 Topología del Perceptrón Multicapa (MLP) y flujo del proceso de inferencia para la selección criptográfica en el borde 82

Figura 10 Registro de eventos del Nodo Maestro evidenciando la ofuscación PSK de la clave de sesión 100

Figura 11 Intercepción del Handshake EAPOL (WPA2) necesario para la desencriptación del tráfico de red 102

Figura 12 Análisis del flujo TCP de la trama MQTT demostrando la ofuscación de la clave de sesión 103

LISTA DE ECUACIONES

Ecuación 1 Potencia 63

Ecuación 2 Energía de transmisión 64

Ecuación 3 Energía de recepción 64

Ecuación 4 Energía de cifrado 64

Ecuación 5 Energía de descifrado 64

Ecuación 6 Tiempo en reposo ligero 64

Ecuación 7 Energía en reposo ligero 64

Ecuación 8 Energía total 64

Ecuación 9 Media Aritmética 91

Ecuación 10 Varianza 92

Ecuación 11 Desviación Estándar 92

Ecuación 12 Intervalo de Confianza 92

Capítulo I: Antecedentes

1.1 Tema

Mecanismo de seguridad adaptativa para entornos IoT basado en la selección dinámica de algoritmos criptográficos ligeros

1.2 **Problema**

El crecimiento vertiginoso del Internet de las Cosas (IoT) ha traído consigo una serie de desafíos significativos, particularmente en el ámbito de la seguridad. A medida que dispositivos cada vez más pequeños, económicos y de recursos limitados se incorporan a redes interconectadas, la necesidad de garantizar la confidencialidad, integridad y disponibilidad de los datos se vuelve crítica. Sin embargo, estos dispositivos embebidos no cuentan con la capacidad de procesamiento suficiente para ejecutar algoritmos criptográficos robustos como los tradicionalmente utilizados en entornos más potentes. Esta limitación tecnológica ha propiciado la aparición de los llamados algoritmos criptográficos ligeros, diseñados para funcionar con bajo consumo energético y en microcontroladores de bajo rendimiento (Aslan, 2025).

No obstante, la simple incorporación de estos algoritmos no ha sido suficiente para mitigar los riesgos de seguridad en entornos IoT. El problema se agrava cuando los algoritmos criptográficos se aplican de forma estática y uniforme, sin considerar la heterogeneidad de los dispositivos, las variaciones en el entorno o las amenazas emergentes. Esta falta de flexibilidad genera múltiples efectos negativos: desde la ineficiencia en el uso de los recursos de hardware, hasta la exposición a ataques por reutilización de claves, degradación de seguridad, o implementación de cifrados demasiado débiles que no responden adecuadamente a los riesgos actuales (Sarmila & Manisekaran, 2025).

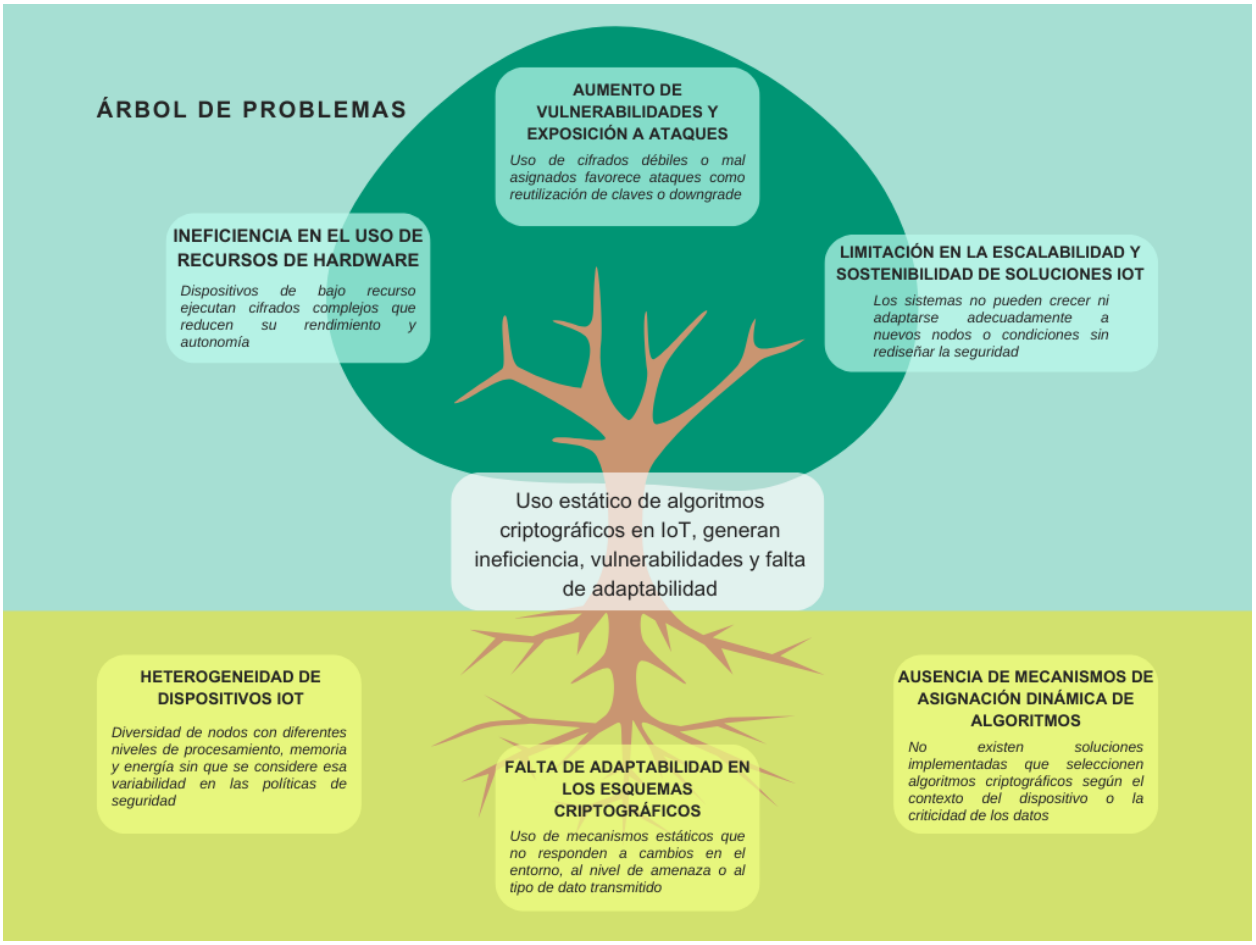
El entorno IoT, al estar compuesto por una gran diversidad de nodos con distintas capacidades y roles, requiere mecanismos de seguridad que se ajusten dinámicamente a sus necesidades. Sin embargo, en la práctica, los desarrolladores continúan adoptando enfoques rígidos, con políticas de cifrado homogéneas que no se adaptan al contexto operativo ni al tipo de información que se transmite. Como resultado, muchos sistemas IoT se encuentran actualmente desprotegidos ante amenazas que evolucionan constantemente, mientras que otros desperdician recursos valiosos al ejecutar mecanismos criptográficos innecesariamente complejos (Faure et al., 2023).

La literatura reciente refleja esta preocupación. Investigaciones como las de (Srinivas et al., 2023) evidencian la ineficiencia de los sistemas actuales para seleccionar algoritmos criptográficos adecuados a cada situación, y destacan la ausencia de mecanismos que permitan una asignación dinámica de los mismos. De igual forma, el trabajo de (Bhandiwad & Ragha, 2025) señala que esta deficiencia técnica no solo compromete la seguridad de los dispositivos, sino que también limita su escalabilidad y sostenibilidad. A pesar de los avances en criptografía ligera, persiste una brecha entre las capacidades de los dispositivos IoT y las exigencias de seguridad contemporáneas.

En este contexto, el problema central radica en la implementación de esquemas de seguridad estáticos e inflexibles en sistemas IoT, los cuales no consideran las restricciones propias de cada dispositivo ni las condiciones cambiantes del entorno. Esta situación compromete la eficiencia, la escalabilidad y la robustez de dichos sistemas, al mismo tiempo que aumenta su vulnerabilidad frente a ataques cibernéticos. La falta de adaptabilidad en los mecanismos criptográficos representa, por tanto, un obstáculo crítico para el desarrollo seguro de soluciones IoT, especialmente en sectores sensibles donde la confiabilidad de la información es vital (Dhiman et al., 2025).

Figura 1

Árbol de problemas



Nota. La imagen muestra la ineficiencia en el uso de recursos de hardware, aumento de vulnerabilidades y limitación en la escalabilidad de soluciones IoT.

1.3 Objetivos

1.3.1. *Objetivo General*

Implementar una arquitectura de seguridad adaptativa para entornos IoT, basada en la selección dinámica de algoritmos criptográficos ligeros, que optimice el uso de recursos y se adapte a las capacidades de distintos microcontroladores.

1.3.2. *Objetivos Específicos*

- Realizar una revisión teórica y documental sobre los mecanismos de seguridad adaptativa y algoritmos criptográficos ligeros aplicables a dispositivos IoT, con el fin de identificar enfoques existentes, limitaciones y oportunidades de mejora, durante el primer mes del proyecto.
- Diseñar una arquitectura de seguridad adaptativa orientada a entornos IoT, considerando criterios de selección dinámica de algoritmos criptográficos ligeros y la heterogeneidad de los dispositivos, en un plazo no mayor a dos meses tras finalizar la revisión teórica.
- Desarrollar un prototipo funcional que implemente la arquitectura diseñada, utilizando microcontroladores de bajo costo y un nodo maestro encargado de gestionar la asignación dinámica de algoritmos, en un lapso de tres meses a partir de la finalización del diseño.
- Ejecutar pruebas de validación del prototipo implementado, evaluando su rendimiento, eficiencia y adaptabilidad frente a diferentes escenarios de red y dispositivos IoT, dentro del último mes del cronograma del proyecto.

1.4 Alcance

Este proyecto se enmarca en la implementación de una arquitectura de seguridad adaptativa para entornos IoT, orientada a la selección dinámica de algoritmos criptográficos ligeros. La propuesta se enfocará en optimizar el uso de recursos y garantizar la adaptabilidad a las capacidades de distintos dispositivos de hardware de bajo costo, sin recurrir a soluciones centralizadas complejas ni requerir altos niveles de procesamiento. El sistema deberá ser capaz de identificar las características de cada dispositivo dentro de una red IoT y asignar el algoritmo criptográfico más apropiado, logrando así un equilibrio entre seguridad, eficiencia y consumo energético.

Para llevar a cabo el objetivo general, se empleará la metodología en cascada, lo que permitirá desarrollar el proyecto de forma ordenada y secuencial. Cada fase del proceso estará claramente delimitada y dependerá de la finalización de la anterior, asegurando coherencia y trazabilidad a lo largo del desarrollo del sistema.

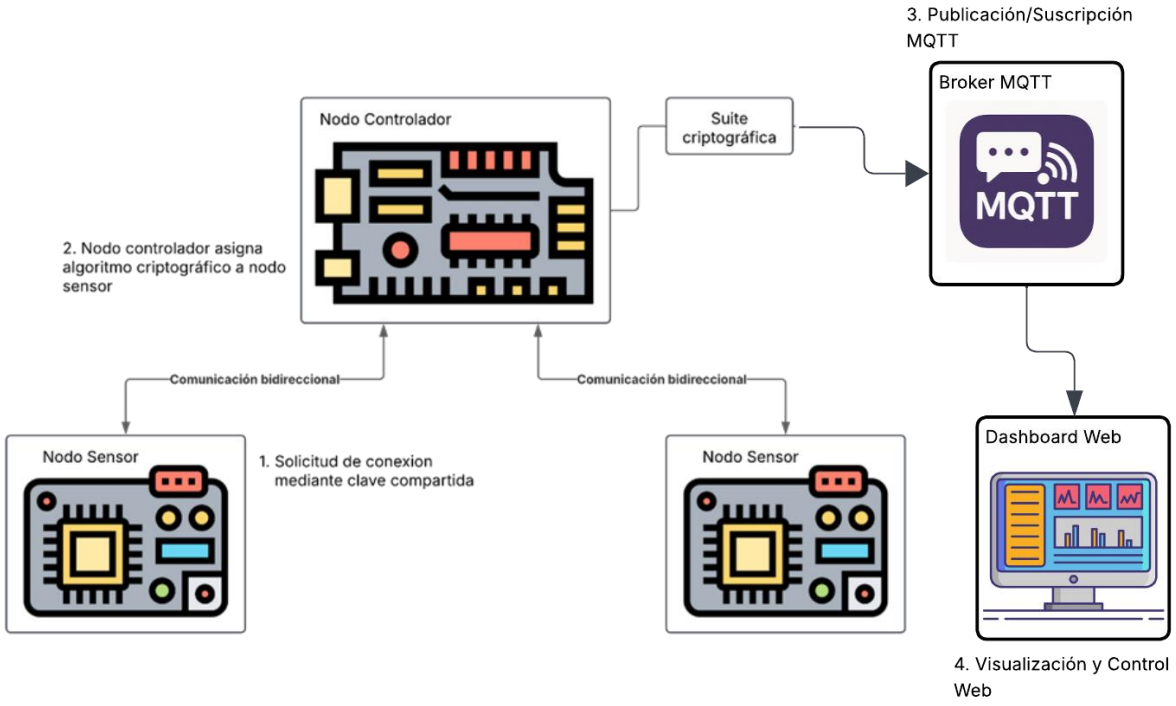
La primera fase consistirá en una revisión teórica y documental para analizar el estado del arte de los mecanismos de seguridad adaptativa y algoritmos criptográficos ligeros aplicables a entornos IoT. Esta base conceptual permitirá fundamentar las decisiones técnicas que guiarán el resto del proyecto.

A continuación, se desarrollará una etapa de análisis y especificación de requerimientos. En esta fase se definirán las funciones principales del sistema, las restricciones operativas, y los criterios de seguridad, eficiencia y adaptabilidad que deben cumplirse. Esta especificación servirá como referencia estructural para el diseño y desarrollo posterior.

Con base en los requerimientos definidos, se procederá a la fase de diseño de la arquitectura lógica del sistema. Esta arquitectura incluirá componentes como el módulo de monitoreo de condiciones, el mecanismo de decisión para la asignación de algoritmos, y el módulo de ejecución criptográfica. El diseño contemplará también la comunicación entre el nodo controlador y los dispositivos IoT, así como los flujos de datos y los parámetros a evaluar.

Figura 2

Arquitectura del Sistema Iot



Nota. La figura muestra la estructura funcional del sistema, donde un nodo controlador gestiona la suite criptográfica y se comunica bidireccionalmente con los nodos sensores, asignando algoritmos según los recursos disponibles.

Posteriormente, se implementará un prototipo funcional del sistema propuesto. Este prototipo permitirá materializar la arquitectura definida y ejecutar las funcionalidades esenciales, incluyendo la selección y aplicación dinámica de algoritmos criptográficos según el contexto y los recursos de cada nodo.

Finalmente, se llevará a cabo una fase de pruebas, en la que se evaluará el comportamiento del sistema bajo distintos escenarios de operación. Se analizará su rendimiento, consumo de

recursos, y capacidad de adaptación, con el objetivo de validar la viabilidad y eficacia del enfoque propuesto.

El alcance del proyecto se limita a la validación del sistema en un entorno controlado compuesto por dispositivos de baja potencia con diferentes niveles de capacidad. No se contempla su implementación a gran escala en redes productivas ni el desarrollo de nuevos algoritmos criptográficos, centrándose exclusivamente en la integración y asignación dinámica de algoritmos existentes bajo criterios de eficiencia y adaptabilidad.

1.5 Justificación

El avance del Internet de las Cosas (IoT) ha impulsado la creación de soluciones conectadas en diversos sectores como la industria, la salud, la agricultura y las ciudades inteligentes. Sin embargo, este crecimiento ha traído consigo desafíos importantes en materia de ciberseguridad, especialmente en dispositivos de bajo costo que cuentan con recursos limitados en términos de memoria, procesamiento y energía. En la mayoría de estos sistemas, los mecanismos de seguridad actuales se implementan de manera rígida, con algoritmos criptográficos aplicados de forma uniforme, sin considerar las características particulares de cada nodo. Esta situación genera problemas de eficiencia, vulnerabilidad y sostenibilidad tecnológica, afectando la escalabilidad y confiabilidad de las redes IoT (Abdelaal et al., 2025).

El presente proyecto se justifica tecnológicamente porque propone una solución innovadora y práctica a esta problemática: un mecanismo de seguridad adaptativa capaz de seleccionar dinámicamente algoritmos criptográficos ligeros según las capacidades y condiciones del dispositivo IoT. Esta estrategia no solo mejora la seguridad en dispositivos de recursos limitados, sino que optimiza el uso energético y computacional, evitando la sobrecarga de nodos

débiles y la subutilización de nodos más potentes. De este modo, se impulsa una transformación tecnológica que responde a las necesidades reales del entorno IoT moderno.

La implementación de esta propuesta contribuye al Objetivo de Desarrollo Sostenible (ODS) 9 – Industria, Innovación e Infraestructura, específicamente en su meta de desarrollar infraestructuras resilientes, promover la industrialización sostenible y fomentar la innovación tecnológica. Al centrarse en la eficiencia y adaptabilidad de la seguridad digital, el proyecto aporta al fortalecimiento de las infraestructuras tecnológicas emergentes, haciendo posible que más sectores adopten IoT de forma segura, confiable y escalable. Esta solución puede ser aplicada a múltiples contextos donde se requiere garantizar la protección de la información sin comprometer los recursos del sistema, lo que incrementa su impacto técnico, económico y social (Unidas, 2030).

Capítulo II: Fundamentación teórica

La presente fundamentación teórica tiene como propósito establecer los conceptos clave, principios técnicos y antecedentes necesarios para sustentar el desarrollo de una arquitectura de seguridad adaptativa en entornos de Internet de las Cosas (IoT). Dado que este tipo de sistemas se caracteriza por la heterogeneidad de dispositivos y la limitación de recursos, se requiere un enfoque riguroso que aborde tanto los aspectos generales del IoT como las particularidades de su seguridad y las soluciones criptográficas más adecuadas para su contexto.

El capítulo inicia con una visión general del IoT, su arquitectura y limitaciones, para luego abordar los principios fundamentales de la seguridad en estos entornos. Posteriormente, se analizan los algoritmos criptográficos ligeros y su aplicabilidad, así como el concepto de seguridad adaptativa y la importancia de contar con mecanismos dinámicos para la selección de algoritmos según el contexto operativo. Esta revisión teórica servirá como base para justificar las decisiones de diseño e implementación que se presentan en los capítulos siguientes.

2.1 Internet de las Cosas (IoT)

El Internet de las Cosas (IoT) es un paradigma tecnológico que permite la interconexión de objetos físicos a través de redes de comunicación para recopilar, transmitir, procesar y compartir datos sin la intervención directa del ser humano. Esta red de dispositivos inteligentes está transformando sectores como la salud, la industria, el medioambiente, las ciudades inteligentes y el hogar digital, gracias a su capacidad de automatizar procesos, optimizar recursos y generar información útil en tiempo real (Bhandiwad & Ragha, 2025).

El IoT se fundamenta en una arquitectura basada en capas, generalmente estructurada en tres niveles: la capa de percepción, la capa de red y la capa de aplicación. La capa de percepción se compone de sensores, actuadores y dispositivos embebidos que capturan datos del entorno

físico. La capa de red se encarga de la transmisión de esa información hacia servidores o plataformas en la nube utilizando tecnologías como Wi-Fi, ZigBee, LoRa o NB-IoT. Finalmente, la capa de aplicación interpreta los datos para generar acciones automáticas o visualizaciones, facilitando la toma de decisiones (Srinivas et al., 2023).

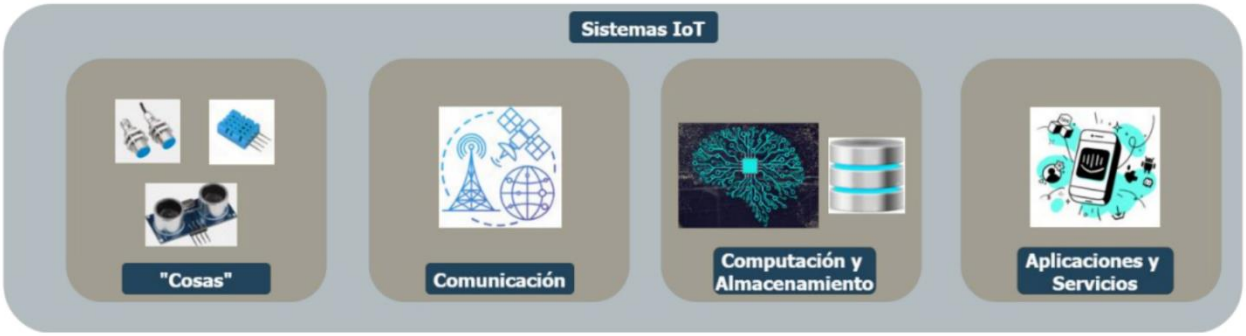
Este modelo arquitectónico también ha dado lugar a estructuras alternativas con capas intermedias como la de procesamiento o la de servicios, especialmente en entornos donde el análisis de datos se realiza de forma distribuida (Agbeyangi & Suleman, 2024). Sin embargo, el funcionamiento eficiente de estas arquitecturas depende en gran medida de la capacidad de los dispositivos involucrados y del entorno operativo.

2.1.1. *Arquitectura general y limitaciones en dispositivos de bajo recurso tecnología arquitectura hardware y software*

La arquitectura funcional de los sistemas IoT se organiza generalmente en capas, donde cada una cumple un rol específico dentro del proceso de recolección, transmisión, procesamiento y utilización de los datos. Esta estructura facilita tanto la implementación modular como el análisis técnico de los sistemas conectados. La Figura 3 presenta una representación esquemática de estas capas esenciales: percepción (o "cosas"), comunicación, computación/almacenamiento y aplicaciones/servicios, que interactúan de forma coordinada para dar soporte a las funcionalidades del IoT (Buitrón Ruiz, 2022; Ortega Patiño, 2023).

Figura 3

Estructura de un sistema IoT



Nota. Diagrama de bloques de un sistema IoT tomado de (Buitrón Ruiz, 2022).

La capa de percepción se encarga de la detección y adquisición de datos físicos mediante sensores y actuadores distribuidos. Estos nodos suelen estar compuestos por microcontroladores de bajo consumo energético (como los ESP32) y tecnologías de comunicación inalámbrica como Zigbee, LoRa o Wi-Fi. Sin embargo, estos dispositivos enfrentan limitaciones importantes como capacidades reducidas de procesamiento (<100 MHz), memoria RAM limitada (<256 KB) y dependencia de fuentes de energía autónomas, como baterías o energía solar (Buitrón Ruiz, 2022; Rhesri et al., 2025). La necesidad de operar de forma prolongada con bajos recursos obliga a utilizar tecnologías de transmisión eficientes como LoRaWAN o protocolos ligeros como CoAP y MQTT.

La capa de red establece el canal de comunicación entre los nodos de percepción y los sistemas de procesamiento, permitiendo la transmisión de datos hacia plataformas locales o remotas. Esta capa puede utilizar tecnologías inalámbricas como Wi-Fi, LTE o redes LPWAN (como LoRaWAN), así como soluciones cableadas como Ethernet. Sin embargo, uno de los

principales desafíos en esta capa radica en la interoperabilidad entre múltiples estándares y protocolos, especialmente en entornos heterogéneos donde coexisten distintos fabricantes, frecuencias y esquemas de transmisión. Además, la seguridad de los datos transmitidos representa una preocupación constante, dado que muchos enlaces carecen de cifrado fuerte o mecanismos de autenticación, debido a las restricciones computacionales de los dispositivos de borde (De Santis et al., 2023; Faure et al., 2023).

En la Tabla 1 se comparan tres protocolos de comunicación comunes en entornos IoT, considerando parámetros clave como consumo energético, alcance efectivo y velocidad de transmisión. Esta comparación permite evidenciar las compensaciones necesarias al momento de seleccionar una tecnología de red según las características del entorno y las capacidades del dispositivo.

Tabla 1

Comparativa de protocolos

Protocolo	Consumo (mA)	Rango (m)	Throughput (Mbps)
BLE 5.2	15	100	2
LoRaWAN	32	15,000	0.05
WiFi-6	120	50	120

Nota: La tabla compara protocolos de comunicación utilizados en IoT según consumo energético, alcance y velocidad de transmisión.

La capa de procesamiento abarca desde sistemas embebidos hasta plataformas de edge y cloud computing. En dispositivos de bajo recurso, el procesamiento local suele ser mínimo,

delegando la tarea al borde de la red (fog computing) o a servidores en la nube, donde se aplican algoritmos de análisis, inteligencia artificial y toma de decisiones (Agbeyangi & Suleman, 2024). Esto implica un desafío adicional en cuanto al consumo energético asociado al envío de datos frecuentes y a la latencia en entornos críticos.

La capa de aplicación representa la interfaz final con el usuario y define los servicios proporcionados, como la visualización de datos, el control remoto o la automatización. Su diseño debe adaptarse a contextos de alta heterogeneidad, permitiendo la integración con plataformas web, dashboards como Grafana, o aplicaciones móviles (Rhesri et al., 2025). En escenarios masivos, se enfrentan problemas de escalabilidad, ya que la arquitectura debe soportar miles de dispositivos generando datos en paralelo.

En conjunto, estas capas muestran cómo la eficiencia del sistema IoT depende fuertemente de la capacidad de los nodos periféricos, que son los más afectados por limitaciones de hardware. Por ejemplo, el uso de algoritmos criptográficos tradicionales como RSA-2048 puede consumir hasta 10 veces más energía que algoritmos ligeros como ChaCha20 o ECC, comprometiendo la vida útil del sistema y su capacidad de respuesta (Rasheed & Kumar, 2025). Asimismo, muchos sistemas operativos ligeros como FreeRTOS o RIOT OS no incluyen soporte nativo para cifrado TLS/SSL, lo que impone barreras para garantizar la privacidad y autenticidad de las comunicaciones (Buitrón Ruiz, 2022).

Estas limitaciones evidencian la necesidad de implementar estrategias adaptativas, como la asignación dinámica de algoritmos criptográficos ligeros, que respondan a las capacidades del hardware y a las condiciones del entorno, sin comprometer la seguridad ni la eficiencia del sistema IoT.

2.2 Seguridad en entornos IoT

La seguridad en el Internet de las Cosas (IoT) representa uno de los desafíos más significativos dentro de la era digital, especialmente debido a la diversidad de dispositivos involucrados, la conectividad constante y las severas restricciones de recursos computacionales y energéticos. Los entornos IoT suelen estar conformados por sensores y microcontroladores de bajo costo, que operan con recursos limitados de memoria, procesamiento y energía, lo que limita la implementación de mecanismos de seguridad tradicionales robustos. Esta condición hace que estos dispositivos sean más vulnerables a ataques, tanto físicos como lógicos, aumentando el riesgo de interrupciones, fugas de datos y malfuncionamientos en redes complejas e interdependientes (Rasheed & Kumar, 2025; Rhesri et al., 2025).

2.2.1. *Principios de seguridad: confidencialidad, integridad y disponibilidad*

El diseño seguro de sistemas IoT debe considerar, como base, los principios fundamentales de la seguridad de la información: confidencialidad, integridad y disponibilidad (CID). Estos principios son críticos para proteger la información y garantizar la fiabilidad del sistema.

La confidencialidad implica asegurar que la información solo sea accesible a las entidades autorizadas. En IoT, esto se complica por la carencia de capacidad de los dispositivos para ejecutar cifrados robustos como RSA o AES, razón por la cual se recurre a esquemas de criptografía ligera. La transmisión inalámbrica sin cifrado o mediante claves estáticas expone datos sensibles como métricas de salud, ubicación geográfica o consumo energético a posibles interceptaciones (Rasheed & Kumar, 2025; Rhesri et al., 2025).

La integridad se refiere a la garantía de que la información no ha sido alterada o manipulada durante su transmisión o almacenamiento. En contextos IoT, donde los dispositivos interactúan con infraestructuras críticas o procesos automatizados, cualquier alteración maliciosa de los datos

puede tener consecuencias graves, como decisiones erróneas en sistemas de control industrial o médicos. Se utilizan mecanismos como códigos de autenticación de mensajes (MAC) o firmas digitales ligeras para validar la autenticidad de los datos (De Santis et al., 2023; Rasheed & Kumar, 2025).

La disponibilidad, por su parte, asegura que los dispositivos y servicios estén accesibles cuando se necesitan. Los ataques de Denegación de Servicio (DoS), por ejemplo, son capaces de agotar los limitados recursos de energía o ancho de banda de una red IoT, provocando caídas del sistema. Este tipo de amenaza es particularmente crítico en aplicaciones que requieren disponibilidad permanente, como redes de sensores ambientales o sistemas de salud remota (Oliveira Morilla, 2024; Rhesri et al., 2025).

2.2.2. *Amenazas comunes*

La diversidad, ubicuidad y bajo perfil de los dispositivos IoT genera una superficie de ataque extensa y propensa a diversas amenazas. Estas amenazas se pueden clasificar en cinco grupos principales.

En primer lugar, los ataques físicos constituyen una amenaza directa al hardware de los dispositivos. A diferencia de los sistemas tradicionales, los dispositivos IoT suelen estar desplegados en ubicaciones vulnerables, como calles, campos o áreas industriales, donde pueden ser fácilmente manipulados. El acceso físico puede permitir técnicas de tampering, como la reprogramación de firmware, la extracción de claves o incluso la sustitución de componentes (Oliveira Morilla, 2024).

En segundo lugar, los ataques de red afectan la transmisión de información entre dispositivos. Son comunes los ataques del tipo Man-in-the-Middle (MitM), que interceptan comunicaciones no cifradas o mal configuradas. Los ataques de denegación de servicio (DoS y

DDoS), por su parte, saturan el sistema mediante tráfico masivo, afectando particularmente a los dispositivos con recursos limitados. Además, los ataques de enrutamiento pueden distorsionar el flujo normal de paquetes, creando bucles, redirecciones maliciosas o falsificación de rutas (Rhesri et al., 2025).

En tercer lugar, los ataques de software se basan en la explotación de vulnerabilidades del firmware, aplicaciones o servicios corriendo en los dispositivos. Las botnets IoT, como la famosa Mirai, han demostrado cómo miles de dispositivos pueden ser comprometidos por el simple hecho de no cambiar credenciales por defecto. Otras amenazas incluyen inyecciones SQL en plataformas web de visualización o ejecución remota de código mediante exploits (Rasheed & Kumar, 2025).

El cuarto grupo incluye los ataques criptográficos, los cuales explotan la debilidad de las implementaciones de cifrado. Muchos dispositivos IoT usan claves estáticas o algoritmos inseguros debido a restricciones computacionales. Esto los vuelve susceptibles a ataques por fuerza bruta, criptoanálisis diferencial o ataque por canal lateral (Rasheed & Kumar, 2025; Rhesri et al., 2025).

Finalmente, las vulnerabilidades comunes catalogadas por OWASP agrupan problemas transversales como la falta de actualizaciones seguras, uso de contraseñas débiles o exposición de servicios sin autenticación. Según el reporte de Palo Alto Networks (2020), hasta un 98% del tráfico generado por dispositivos IoT no está cifrado, lo cual evidencia la urgente necesidad de enfoques de seguridad más adaptables y automatizados.

La Tabla 2 sintetiza algunas de estas amenazas clave y compara su impacto entre entornos IoT y sistemas IT tradicionales:

Tabla 2

Amenazas en entornos IoT

Amenaza	**Impacto en IoT**	**Impacto en sistemas IT tradicionales**
DoS/DDoS	Degradación total del sistema y agotamiento energético	Pérdida temporal de conectividad
Malware (botnets)	Inclusión en redes de ataque distribuido	A menudo bloqueados por antivirus/firewalls
Falta de cifrado	Vulnerabilidad crítica en sensores y comunicación	Suelen contar con TLS o VPN como estándar

Nota: La tabla muestra una comparación del impacto de amenazas comunes en entornos IoT frente a sistemas IT tradicionales. Obtenido de (Buitrón Ruiz, 2022).

En conjunto, estas amenazas confirman la necesidad de implementar arquitecturas de seguridad más inteligentes. La adopción de mecanismos adaptativos como la selección dinámica de algoritmos criptográficos basada en el contexto y la capacidad del dispositivo permite equilibrar protección y eficiencia, tal como se plantea en este trabajo (Oliveira Morilla, 2024; Rasheed & Kumar, 2025).

2.2.3. *Concepto de seguridad adaptativa*

La seguridad adaptativa en entornos IoT es un enfoque emergente que plantea la necesidad de ajustar dinámicamente los mecanismos de protección en función del contexto operativo, las capacidades del dispositivo y la variabilidad del entorno de red (V. Kumar, 2024; Sehrawat et al., 2019). A diferencia de los modelos tradicionales de seguridad caracterizados por configuraciones

estáticas y reglas fijas, la seguridad adaptativa introduce un componente de flexibilidad que permite a los sistemas responder de forma proactiva a cambios en las amenazas, condiciones técnicas o políticas operativas.

Este enfoque es particularmente relevante en IoT, donde la heterogeneidad de los dispositivos, el limitado poder de cómputo, y la conectividad intermitente limitan la aplicabilidad de soluciones de seguridad convencionales. Por ejemplo, un sensor de temperatura alimentado por batería no puede sostener algoritmos criptográficos pesados de forma continua. En tales escenarios, la seguridad adaptativa permite al sistema modular su comportamiento de protección, seleccionando medidas proporcionales al riesgo percibido y a los recursos disponibles (Al-Mousa & others, 2024; Oliveira Morilla, 2024).

Una arquitectura de seguridad adaptativa incluye varios componentes: monitoreo del contexto (nivel de energía, carga de procesamiento, calidad de enlace), evaluación del riesgo (detección de anomalías, amenazas inminentes), políticas de decisión (por reglas, árboles de decisión o aprendizaje automático), y finalmente un módulo de acción que ajusta dinámicamente los algoritmos criptográficos, protocolos de autenticación o frecuencia de comunicación (Hudda & Haribabu, 2025; Rasheed & Kumar, 2025).

En esta línea, Faure et al. (2023) destacan que una estrategia efectiva debe incluir la posibilidad de reconfigurar remotamente los parámetros de seguridad, así como establecer niveles escalonados de protección, permitiendo una transición fluida entre modos de bajo, medio o alto nivel de seguridad. Esta granularidad es esencial para preservar la operatividad del sistema sin sacrificar su integridad ni exponerlo innecesariamente.

Adicionalmente, la seguridad adaptativa puede aprovechar técnicas de edge computing para descentralizar decisiones de seguridad, reduciendo la dependencia de la nube y mitigando la

latencia asociada a decisiones centralizadas. En este sentido, nodos intermedios más potentes (como gateways) pueden asumir funciones de monitoreo y delegar acciones específicas a los dispositivos terminales (Agbeyangi & Suleman, 2024; Buitrón Ruiz, 2022).

Uno de los beneficios más importantes de este enfoque es su capacidad para prolongar la vida útil de los dispositivos IoT. Al disminuir la carga computacional mediante algoritmos más ligeros o limitar la frecuencia de transmisión cuando no hay riesgo, se reduce significativamente el consumo energético. Además, la adaptabilidad mejora la resiliencia frente a ataques, ya que impide que un sistema opere de manera predecible, dificultando la planificación de ciberataques dirigidos (Oliveira Morilla, 2024; Ortega Chulde, 2023; Rasheed & Kumar, 2025).

No obstante, la implementación de sistemas de seguridad adaptativa no está exenta de desafíos. Es necesario garantizar la coherencia entre nodos heterogéneos, asegurar la integridad del módulo de toma de decisiones (que puede convertirse en un blanco de ataque), y validar que los algoritmos seleccionados cumplen estándares mínimos de seguridad, incluso en su configuración más ligera (V. Kumar, 2024; Sehrawat et al., 2019). Además, se requieren mecanismos que aseguren que los cambios de configuración no interrumpan los servicios críticos del sistema, especialmente en aplicaciones sensibles como el monitoreo de salud o sistemas industriales.

2.2.4. *Mecanismos de ajuste dinámico y gestión desde nodo maestro*

El concepto de ajuste dinámico en seguridad para entornos IoT se fundamenta en la necesidad de adaptar continuamente los mecanismos criptográficos, de autenticación y comunicación de los dispositivos según las condiciones del entorno y sus capacidades internas. Este ajuste no es solo deseable, sino esencial, considerando la limitada energía, memoria y poder de procesamiento de la mayoría de los nodos IoT. La propuesta de control centralizado mediante

un nodo maestro surge como solución eficaz para orquestar esta adaptabilidad en redes heterogéneas (Faure et al., 2023; V. Kumar, 2024; S. et al., 2024).

Los mecanismos de ajuste dinámico operan en tres niveles principales: detección, decisión y despliegue. En la fase de detección, el sistema monitorea métricas relevantes como carga de CPU, consumo energético, nivel de señal o historial de ataques detectados. Luego, en la fase de decisión, el sistema frecuentemente basado en reglas lógicas, árboles de decisión o incluso algoritmos de aprendizaje automático selecciona el nivel de seguridad apropiado para el nodo. Finalmente, en la fase de despliegue, se actualizan los algoritmos de cifrado, protocolos de autenticación, o se modifican los intervalos de comunicación según las políticas predefinidas (Rasheed & Kumar, 2025; Sehrawat et al., 2019).

Este proceso se puede centralizar en un nodo maestro, que actúa como agente de coordinación criptográfica. Este nodo, más robusto en cuanto a procesamiento y memoria, gestiona la recopilación de datos contextuales provenientes de la red IoT, evalúa riesgos y capacidades de cada dispositivo, y asigna algoritmos de seguridad personalizados. Así, un nodo de baja capacidad puede recibir un esquema de cifrado ultraligero como PRESENT o SPECK, mientras que uno más potente podría operar con ChaCha20 o AES-128 dependiendo de las condiciones (Agbeyangi & Suleman, 2024; Al-Mousa & others, 2024).

El uso de nodos maestros habilita una estructura jerárquica donde se minimiza el procesamiento distribuido redundante, optimizando recursos globales y permitiendo un despliegue más ágil de políticas de seguridad. Además, al centralizar decisiones complejas, se disminuye el requerimiento de inteligencia en cada dispositivo terminal, lo que es particularmente útil en sensores o actuadores de bajo costo (Buitrón Ruiz, 2022; Oliveira Morilla, 2024). Sin embargo,

esto también introduce desafíos relacionados con la confiabilidad del nodo maestro, el balance de carga y la necesidad de redundancia para evitar puntos únicos de falla.

En los últimos años se han propuesto modelos híbridos, donde el nodo maestro no solo asigna los algoritmos sino que también modula el grado de protección de manera contextual. Por ejemplo, en situaciones de bajo tráfico o bajo riesgo, puede asignarse una protección básica para conservar energía; mientras que frente a una detección de amenaza, el nodo puede instruir a los dispositivos para escalar su nivel de seguridad a través de algoritmos más robustos o autenticación multifactor (Faure et al., 2023; Sehrawat et al., 2019).

Por otro lado, algunos autores proponen que estos nodos de control puedan aplicar algoritmos de inteligencia artificial para reconocer patrones de comportamiento y anticipar necesidades de ajuste antes de que ocurran fallos o ataques (V. Kumar, 2024). En este contexto, la integración de mecanismos de aprendizaje automático permitiría al nodo maestro evolucionar con el tiempo, adaptando políticas de manera cada vez más eficiente, sin intervención manual.

Un aspecto clave de estos mecanismos es su capacidad para operar sin interrumpir la funcionalidad del sistema. Para lograrlo, es común emplear estrategias de cambio progresivo o intercambio transparente de algoritmos, en donde la transición entre métodos criptográficos se realiza en etapas seguras, preservando la integridad de los datos y la continuidad del servicio. Este tipo de gestión dinámica también debe estar acompañado de una infraestructura de sincronización temporal y de claves seguras para evitar vulnerabilidades durante el proceso de reconfiguración (Faure et al., 2023).

Finalmente, es importante considerar que la implementación del ajuste dinámico no debe limitarse al plano técnico, sino también al nivel de políticas de gobernanza. Los nodos maestros deben respetar reglas predefinidas que regulen hasta qué punto puede ajustarse la seguridad, qué

algoritmos están permitidos y cuáles son las prioridades entre rendimiento, protección y consumo energético (Rasheed & Kumar, 2025).

En conjunto, estos mecanismos y estrategias ofrecen un marco sólido para aplicar seguridad adaptativa en entornos IoT de manera escalable y sostenible. Su incorporación en sistemas críticos como la salud, la industria 4.0 o las ciudades inteligentes representa un paso hacia arquitecturas más resilientes y conscientes del entorno operativo.

2.3 Criptografía ligera aplicada a IoT

La criptografía ligera surge como respuesta directa a los retos que presentan los dispositivos IoT en términos de procesamiento, almacenamiento, consumo energético y conectividad. A medida que el número de nodos inteligentes crece en entornos como ciudades inteligentes, salud remota o agricultura de precisión, también lo hace la necesidad de garantizar la seguridad de los datos que generan y transmiten. Sin embargo, los algoritmos criptográficos tradicionales no se diseñaron para operar eficientemente en dispositivos con recursos tan limitados. De ahí que la criptografía ligera se proponga como una alternativa viable, ofreciendo un compromiso óptimo entre seguridad, eficiencia y adaptabilidad (Al-Mousa & others, 2024; Sehrawat et al., 2019).

2.3.1. *Definición y ventajas frente a algoritmos tradicionales*

La criptografía ligera (lightweight cryptography) es una subdisciplina de la criptografía que se enfoca en el diseño y adaptación de algoritmos criptográficos específicamente optimizados para su implementación en dispositivos con recursos limitados, como los que predominan en entornos IoT. A diferencia de los algoritmos criptográficos tradicionales, que priorizan robustez matemática y rendimiento en sistemas con capacidad de cómputo abundante, los esquemas ligeros

están diseñados para operar de forma segura con restricciones estrictas de procesamiento, memoria, ancho de banda y energía (Mohamad Al-Aboosi et al., 2021).

En esencia, los algoritmos ligeros buscan mantener un equilibrio entre seguridad criptográfica aceptable y eficiencia computacional, permitiendo que microcontroladores, sensores y nodos embebidos puedan ejecutar operaciones de cifrado, autenticación y firma digital sin comprometer su funcionalidad principal ni agotar su batería. Esta necesidad ha cobrado relevancia en años recientes debido a la rápida expansión del IoT, donde la proliferación de dispositivos ha generado escenarios con millones de nodos interconectados que operan de forma autónoma en infraestructuras críticas como redes de energía, salud y transporte inteligente (Agbeyangi & Suleman, 2024; Sehrawat et al., 2019).

Una de las principales diferencias entre la criptografía ligera y la tradicional reside en la estructura algorítmica y el tamaño de los bloques de datos que manejan. Por ejemplo, mientras que el AES estándar opera sobre bloques de 128 bits y requiere múltiples rondas de transformación, algoritmos ligeros como PRESENT o SPECK utilizan bloques de 64 bits con estructuras de tipo SPN o Feistel simplificadas, lo que reduce el número de puertas lógicas requeridas para su implementación en hardware (Mohamad Al-Aboosi et al., 2021; Singh et al., 2024). Esta optimización no solo disminuye el uso de memoria ROM/RAM, sino que también reduce el tiempo de ejecución y el consumo energético, factores críticos en nodos alimentados por baterías o recolección de energía ambiental.

Otra ventaja clave de los algoritmos ligeros frente a los tradicionales es su flexibilidad en implementación híbrida. Algunos esquemas, como Simon, LEA o ChaCha20, permiten implementaciones tanto en hardware como en software con alta eficiencia, lo cual facilita su integración en plataformas IoT heterogéneas. Además, muchos de estos algoritmos están diseñados

con objetivos específicos, como minimizar la superficie de ataque ante técnicas de criptoanálisis diferencial, lineal o ataques de canal lateral, lo cual incrementa su viabilidad en entornos expuestos como redes inalámbricas abiertas o dispositivos desplegados en campo (Al-Mousa & others, 2024).

Desde una perspectiva energética, diversos estudios han demostrado que la criptografía tradicional es inviable en muchos entornos IoT. Por ejemplo, ejecutar RSA-2048 en un microcontrolador de 8 bits puede consumir hasta 500 ms por operación de firma digital, lo que agota la batería en pocos ciclos de uso. En contraste, algoritmos ligeros como HIGHT o LBlock pueden realizar operaciones comparables en menos de 30 ms con una fracción del consumo energético (Faure et al., 2023; Rasheed & Kumar, 2025). Esta eficiencia se traduce directamente en mayor vida útil del dispositivo, menos latencia en la transmisión de datos y mayor autonomía operativa.

En términos de compatibilidad, los algoritmos tradicionales también presentan limitaciones. Muchos dispositivos IoT no soportan bibliotecas completas de criptografía como OpenSSL debido al tamaño de las dependencias y la falta de soporte en sistemas operativos reducidos como FreeRTOS o Contiki-NG. Por el contrario, los algoritmos ligeros suelen tener implementaciones optimizadas de código reducido que pueden integrarse directamente en el firmware del dispositivo, lo cual facilita el desarrollo de soluciones integradas seguras y eficientes (Oliveira Morilla, 2024; Sehrawat et al., 2019).

Cabe resaltar que, aunque la seguridad ofrecida por los algoritmos ligeros puede ser menor en comparación con esquemas robustos como AES-256 o RSA-4096, su nivel de protección es suficiente para contextos de riesgo bajo o medio, especialmente si se complementa con otros mecanismos como autenticación basada en contexto, control de acceso físico o monitorización de

tráfico. Además, la evolución de esta disciplina ha llevado a la estandarización de algoritmos como Ascon, elegidos por el NIST como candidatos finales para criptografía ligera, consolidando su confiabilidad y adopción futura (National Institute of Standards & (NIST), 2023).

2.3.2. *Clasificación de Algoritmos Criptográficos Ligeros*

Para ilustrar la diversidad de la criptografía ligera, los algoritmos relevantes pueden clasificarse según varios criterios técnicos. A continuación, se describen las principales tipologías basadas en el documento de referencia (Srinivas et al., 2023), las cuales se detallan extensamente en la tabla comparativa de algoritmos (ver Anexo A).

Una clasificación principal se basa en el tipo de encriptación. El documento (Srinivas et al., 2023) distingue entre cifradores de bloque (Block Ciphers), que procesan datos en fragmentos de tamaño fijo, y cifradores de flujo (Stream Ciphers), que operan sobre unidades más pequeñas como bits o bytes, ideales para baja latencia. En la primera categoría (bloque) se encuentran algoritmos como AES-128, PRESENT, las familias SIMON y SPECK, SKINNY, CLEFIA, Kuznyechik, RECTANGLE y TWINE. En la segunda categoría (flujo), se destacan Salsa20, ChaCha20 y ZUC.

Otra clasificación crucial es según el tamaño de clave, un criterio de eficiencia determinante. Los algoritmos varían para equilibrar seguridad y recursos: algunos ofrecen claves de 80 bits como TWINE y RECTANGLE (este último también soporta 128 y 192 bits). La clave de 128 bits es una opción común, implementada en AES-128, PRESENT (hasta 128 bits), SKINNY, PRINCE, PICAROON y CLEFIA (que también soporta 192 bits). Para niveles de seguridad más altos, Sparx y Kuznyechik se especifican con claves de 128 o 256 bits y 256 bits,

respectivamente. Algoritmos como MORUS, Ketje y TINYJAMBU son notables por soportar un rango variable de tamaños de clave y bloque (Srinivas et al., 2023).

Finalmente, una clasificación funcional (el "etc.") distingue los algoritmos que van más allá del cifrado simple. El cifrado autenticado (AEAD) es vital en IoT, ya que combina confidencialidad e integridad en una sola operación. En esta categoría, Srinivas et al. (2023) listan ASCON (ahora un estándar NIST), SKINNY-AEAD, PICAROON, Deoxys-BC, AEGIS, MORUS y Xoodyak. El documento también identifica otras primitivas esenciales: para la autenticación pura de mensajes (MAC), se destaca Poly1305; para el intercambio de claves asimétrico, se utiliza ECDH; y para firmas digitales resistentes a la cuántica, se menciona SPHINCS+.

2.4 Selección dinámica de algoritmos criptográficos

En entornos IoT, donde la infraestructura se compone de dispositivos con distintas capacidades de procesamiento, almacenamiento y energía, es fundamental implementar mecanismos de seguridad que se ajusten dinámicamente a las condiciones del sistema. La selección dinámica de algoritmos criptográficos es un enfoque innovador que busca asignar, en tiempo real, el algoritmo más adecuado para proteger la comunicación y los datos de cada dispositivo, teniendo en cuenta sus limitaciones y el nivel de riesgo del entorno (Al-Mousa & others, 2024; Kumar, 2024).

Este paradigma representa un cambio de modelo respecto a los enfoques tradicionales, donde los algoritmos criptográficos son definidos de manera estática desde el inicio de la implementación. En su lugar, un sistema adaptativo permite que un nodo maestro o entidad centralizada (por ejemplo, una Raspberry Pi o gateway de borde) evalúe factores como el estado energético, la carga de procesamiento y la criticidad de la aplicación para seleccionar, asignar o

reconfigurar algoritmos ligeros en tiempo de ejecución (Agbeyangi & Suleman, 2024; Oliveira Morilla, 2024).

2.4.1. *Esquemas dinámicos*

La selección dinámica de algoritmos criptográficos se posiciona como una solución crítica para la seguridad eficiente en entornos IoT, caracterizados por su heterogeneidad, movilidad, escalabilidad y variabilidad en el consumo de recursos. Un esquema dinámico en este contexto se refiere a la capacidad del sistema para adaptar automáticamente el tipo de cifrado, la longitud de clave, los mecanismos de autenticación u otros parámetros criptográficos, en función del estado actual del dispositivo (memoria disponible, carga de CPU, nivel de batería) y del entorno de red (nivel de amenaza, tipo de tráfico, velocidad de enlace) (Al-Mousa & others, 2024; Faure et al., 2023).

A diferencia de los esquemas criptográficos estáticos, donde el algoritmo y los parámetros de cifrado son definidos en el momento del diseño y permanecen fijos durante todo el ciclo de vida del sistema, los esquemas dinámicos permiten seleccionar o cambiar el algoritmo más adecuado en tiempo real o casi en tiempo real, promoviendo un equilibrio óptimo entre seguridad y desempeño. Esta adaptabilidad es especialmente relevante en sistemas IoT con topologías móviles, en los que nodos pueden entrar o salir de la red constantemente y cuyas capacidades pueden diferir ampliamente incluso dentro de un mismo dominio (Rasheed & Kumar, 2025).

Uno de los enfoques más prometedores en esta área es el de la criptografía cognitiva o adaptativa basada en contexto, donde el nodo maestro o un sistema central toma decisiones sobre qué algoritmo asignar a cada nodo esclavo, considerando información contextual como ubicación, latencia de red, prioridad del servicio, nivel de energía remanente y perfiles de amenazas detectadas. Esta lógica puede estar apoyada por reglas estáticas definidas por el diseñador del

sistema, por algoritmos de decisión automática, o incluso por mecanismos de aprendizaje automático que evolucionen conforme el entorno se modifique (Buitrón Ruiz, 2022; P. G. A. Kumar et al., 2022).

En la práctica, un esquema de selección dinámica puede emplear múltiples algoritmos ligeros, tales como PRESENT, HIGHT, Ascon o Simon, almacenados en una "suite criptográfica" dentro del nodo maestro o gateway. Al iniciar la comunicación, se puede utilizar un protocolo de establecimiento seguro inicial (por ejemplo, basado en DES con clave compartida preconfigurada), tras lo cual el nodo maestro evalúa el perfil del nodo esclavo y selecciona el algoritmo más apropiado para las futuras comunicaciones (Buitrón Ruiz, 2022; P. G. A. Kumar et al., 2022). Esta decisión puede incluir el uso de funciones de hash, autenticación mutua, técnicas de derivación de clave, o modificación del tamaño del bloque cifrado según las capacidades del dispositivo (Oliveira Morilla, 2024).

Además, algunos estudios han propuesto modelos multi-ronda de selección dinámica, donde la elección del algoritmo se revisa periódicamente o tras eventos específicos (como caída del rendimiento o cambio en la red). Este enfoque permite una actualización más segura del entorno criptográfico ante ataques prolongados o degradación del sistema. Por ejemplo, en el trabajo de (Al-Mousa & others, 2024), se propone un sistema donde cada nodo puede alternar entre algoritmos livianos según un índice de desempeño que incluye consumo de CPU, latencia de cifrado y porcentaje de pérdida de paquetes.

El desarrollo de esquemas de control jerárquico y centralizado, donde un nodo maestro gestiona la política criptográfica de toda la red, permite además una administración coordinada, la actualización remota de algoritmos y una respuesta rápida ante vulnerabilidades detectadas. Esta arquitectura también permite optimizar el uso del ancho de banda y la sincronización entre nodos,

aspectos claves para mantener la eficiencia operativa de la red IoT, especialmente en aplicaciones industriales, médicas o energéticas donde el retardo y la fiabilidad son críticos (Faure et al., 2023; P. G. A. Kumar et al., 2022).

No obstante, la implementación de estos esquemas dinámicos requiere garantizar ciertos principios fundamentales: la autenticidad de los mensajes de asignación del algoritmo, la confidencialidad del canal de configuración y la verificación mutua de capacidades entre maestro y esclavo. Si estos elementos no están adecuadamente protegidos, un atacante podría forzar la selección de un algoritmo más débil para facilitar ataques de criptoanálisis o sabotaje del sistema (Rasheed & Kumar, 2025).

2.4.2. *Retos y beneficios de la asignación adaptativa*

A pesar de sus beneficios, implementar un mecanismo de selección dinámica de algoritmos criptográficos en IoT conlleva desafíos importantes. Uno de los principales es la heterogeneidad de dispositivos, ya que los microcontroladores pueden tener arquitecturas, sistemas operativos y capacidades criptográficas muy distintas. Esto requiere que el sistema soporte múltiples algoritmos, modos de operación y bibliotecas compatibles, lo cual aumenta la complejidad del desarrollo y mantenimiento (Agbeyangi & Suleman, 2024).

Otro reto relevante es la latencia que puede introducir la negociación o reconfiguración criptográfica. Si no se gestiona adecuadamente, podría afectar aplicaciones sensibles al tiempo como el monitoreo de salud o los sistemas industriales en tiempo real. Además, el intercambio de información de control entre el nodo maestro y los sensores debe estar protegido, para evitar ataques de manipulación o suplantación durante el proceso de asignación (V. Kumar, 2024).

En cuanto a la eficiencia energética, si bien la asignación dinámica está diseñada para optimizar el consumo, mal implementada puede tener el efecto contrario. Por eso es importante

utilizar técnicas livianas de evaluación y actualización que no saturen la CPU ni las comunicaciones del sistema (Al-Mousa & others, 2024; Oliveira Morilla, 2024).

No obstante, los beneficios superan los desafíos cuando se implementa correctamente. Entre ellos destacan:

- Optimización de recursos: Permite ajustar el nivel de seguridad a la capacidad de cada nodo, evitando el sobreuso de algoritmos pesados en hardware limitado.
- Resiliencia ante amenazas: Se pueden reforzar dinámicamente las políticas criptográficas cuando se detectan cambios en el entorno, como aumento del tráfico o señales de ataque.
- Extensión de la vida útil de los dispositivos: Al elegir algoritmos menos demandantes en momentos de baja carga o riesgo, se reduce el consumo energético general del sistema (Rasheed & Kumar, 2025).
- Flexibilidad y escalabilidad: Facilita la integración de nuevos nodos o sensores sin necesidad de reconfigurar todo el sistema.

Este proyecto propone una arquitectura que incorpora estos principios mediante un nodo maestro que evalúa parámetros clave (nivel de batería, carga de procesamiento, criticidad del dato) y asigna algoritmos criptográficos de manera adaptativa. Con esto, se busca maximizar la eficiencia y robustez de la solución de seguridad, respondiendo a las necesidades dinámicas del entorno IoT.

Tabla 3

Comparativa de enfoques dinámicos en criptografía ligera para IoT

Enfoque	**Ventajas**	**Desventajas**	**Referencias**
Multi-round lightweight (MRLC)	Alta resistencia a ataques. Ajuste granular por segmento de datos cifrados.	Requiere múltiples claves y segmentación de datos.	Al-Mousa et al. (2024)
Selector basado en recursos	Optimiza el uso de batería y procesamiento según condiciones del dispositivo.	Riesgo si los umbrales de decisión están mal calibrados.	Faure et al. (2023)
Cifrado adaptable basado en contexto	Cambia dinámicamente entre algoritmos como ECC, ChaCha20 o AES según el entorno.	Puede generar latencias en entornos críticos si no está bien optimizado.	Rasheed & Kumar (2025)

Nota: La tabla compara distintos enfoques de selección dinámica en criptografía ligera aplicables a entornos IoT. Obtenido de (Al-Mousa & others, 2024; P. G. A. Kumar et al., 2022).

Capítulo III: Diseño e implementación

3.1 Metodología

El presente proyecto se estructura bajo el enfoque de la metodología en cascada, un modelo clásico y aún vigente en el ámbito de la ingeniería de software, especialmente valorado en aquellos desarrollos tecnológicos que demandan un proceso riguroso, sistemático y secuencial. Este enfoque plantea un avance ordenado a través de fases claramente delimitadas: análisis de requerimientos, diseño, implementación, pruebas y mantenimiento; donde el progreso de cada etapa depende de la culminación satisfactoria de la anterior. La elección de este modelo no obedece a la inercia de la tradición, sino a la necesidad de garantizar trazabilidad, precisión en los objetivos y un control minucioso sobre el cumplimiento de los hitos definidos en el alcance del proyecto. Así, cada fase no se percibe como un compartimento aislado, sino como un peldaño firme que sustenta al siguiente.

La investigación aquí presentada se centra de manera particular en tres de estas fases críticas: la especificación de requerimientos, el diseño del sistema y la implementación de un prototipo funcional. Dichas etapas representan la transición esencial desde la identificación del problema hasta la materialización de una solución operativa. En efecto, permiten traducir la problemática detectada en un conjunto de requisitos precisos, configurar una arquitectura lógica coherente y, finalmente, dar forma a un sistema ejecutable que encarne la propuesta de seguridad adaptativa en entornos de Internet de las Cosas (IoT).

Para la etapa de especificación de requerimientos se adopta como referencia la norma internacional IEEE 29148:2018 – Systems and Software Engineering – Life Cycle Processes –

Requirements Engineering. Este estándar ofrece un marco de buenas prácticas que orienta la definición, análisis, documentación y gestión de los requerimientos, asegurando que estos cumplan con atributos fundamentales: claridad, verificabilidad, consistencia y viabilidad. En otras palabras, actúa como un antídoto contra la ambigüedad y como una brújula que orienta los requisitos hacia los objetivos reales del sistema.

La convergencia entre la disciplina estructurada de la metodología en cascada y la rigurosidad del estándar IEEE 29148 configura un marco metodológico sólido para el desarrollo del proyecto. Esta articulación no solo respalda las decisiones técnicas sobre bases verificables, sino que también garantiza que el sistema resultante responda con eficacia a las crecientes y complejas demandas de seguridad en los entornos IoT.

3.2 Fase de Requerimientos

La fase de especificación de requerimientos constituye el pilar fundamental del ciclo de vida del proyecto. Su finalidad es identificar, analizar y documentar de manera sistemática las necesidades funcionales, no funcionales y de arquitectura del sistema de seguridad adaptativa.

Siguiendo las directrices de la norma ISO/IEC/IEEE 29148:2018, los requerimientos deben ser claros, consistentes, verificables y trazables. Para facilitar su gestión, se establece la siguiente nomenclatura de identificación.

Tabla 4

Nomenclatura de los requerimientos

Abreviatura	**Descripción**
StSR	Requerimientos de Stakeholders (Interesados)

SySR	Requerimientos de Sistema (Funcionales y No Funcionales)
SrAH	Requerimientos de Arquitectura (Software y Hardware)

3.2.1. *Requerimientos de Stakeholders (StSR)*

Los Requerimientos de Stakeholders (StSR) definen las metas y objetivos de alto nivel del proyecto, reflejando las necesidades de los usuarios y patrocinadores del sistema.

Tabla 5

Requerimientos de Stakeholders

Número	**Descripción**	**Prioridad**
StSR1	El sistema debe equilibrar el nivel de seguridad (confidencialidad, integridad y adaptabilidad) y el consumo de recursos (batería, procesamiento) en los dispositivos IoT.	Alta
StSR2	El sistema debe adaptar dinámicamente su nivel de seguridad según las capacidades físicas de cada dispositivo conectado.	Alta
StSR3	Un "Nodo Maestro" debe gestionar de forma centralizada las decisiones de seguridad de todos los "Nodos Cliente".	Alta
StSR4	El sistema debe prolongar la vida útil de las baterías de los nodos sensores, minimizando su carga de procesamiento.	Alta
StSR5	El sistema debe ser escalable, permitiendo añadir nuevos nodos a la red de forma sencilla y transparente.	Media

StSR6	El sistema debe contar con un panel de control web (Dashboard) para monitorear el estado y la seguridad de los dispositivos en tiempo real.	Baja

3.2.2. *Requerimientos de Sistema (SySR)*

Los Requerimientos de Sistema (SySR) especifican las funciones que el sistema debe realizar (Funcionales) y las cualidades con las que debe operar (No Funcionales) para satisfacer los StSR.

Tabla 6

Requerimientos de sistema

Número	**Descripción**	**Prioridad**	**Relacionado**
SySR-RF1	(Registro) Los Nodos Cliente deben iniciar la comunicación enviando un mensaje de aviso al conectarse a la red.	Alta	StSR3, StSR5
SySR-RF2	El mensaje de presentación debe informar al sistema sobre las capacidades de hardware del nodo (procesamiento, memoria disponible, energía y saturación de red).	Alta	StSR2
SySR-RF3	(Motor de Decisión) El Nodo Maestro debe evaluar las capacidades del nodo y asignarle automáticamente el algoritmo criptográfico más adecuado.	Alta	StSR1, StSR2
SySR-RF4	(Generación) El Nodo Maestro debe crear una clave de seguridad única para el algoritmo seleccionado.	Alta	StSR1

SySR-RF5	(Asignación) El Nodo Maestro debe enviar el algoritmo seleccionado y la clave generada al respectivo Nodo Cliente.	Alta	StSR3
SySR-RF6	(Actualización) Los Nodos Cliente deben recibir su nueva configuración de seguridad y actualizar sus parámetros internos de cifrado.	Alta	StSR2
SySR-RF7	(Envío) Los Nodos Cliente deben cifrar la información de sus sensores usando el algoritmo y la clave que les fueron asignados.	Alta	StSR1
SySR-RF8	Los Nodos Cliente deben transmitir la información ya cifrada hacia el Nodo Maestro.	Alta	StSR3
SySR-RF9	(Recepción) El Nodo Maestro debe escuchar continuamente la red para recibir los datos cifrados de todos los clientes.	Alta	StSR3
SySR-RF10	(Descifrado) El Nodo Maestro debe identificar qué dispositivo envió el mensaje, recuperar su clave correspondiente y descifrar la información.	Alta	StSR1, StSR3
SySR-RF11	(Historial) El Nodo Maestro debe almacenar temporalmente un registro de los últimos mensajes procesados y sus métricas.	Media	StSR6
SySR-RF12	(Dashboard API) El Nodo Maestro debe proveer servicios de consulta para enviar el estado de los dispositivos al panel de control.	Media	StSR6

SySR-RF13	(Tiempo Real) El Nodo Maestro debe enviar actualizaciones automáticas e inmediatas al panel web cuando ocurran cambios o lleguen datos.	Media	StSR6
SySR-RNF1	(Seguridad) El canal principal de comunicación debe estar protegido mediante usuario, contraseña y cifrado de red estándar.	Alta	StSR1
SySR-RNF2	(Disponibilidad) La configuración de seguridad debe quedar retenida en el servidor para que los nodos la reciban de inmediato si pierden la conexión.	Alta	StSR4
SySR-RNF3	(Integridad) Los algoritmos asignados deben garantizar un mecanismo para comprobar que los datos no fueron manipulados en el trayecto.	Alta	StSR1
SySR-RNF4	(Rendimiento) El proceso completo de registro de un dispositivo y la asignación de su algoritmo debe ser rápido, con una latencia mínima.	Media	StSR4

3.2.3. *Requerimientos de Arquitectura (Software y Hardware) (SrAH)*

Los Requerimientos de Arquitectura (SrAH) definen las restricciones técnicas, plataformas de software, y especificaciones mínimas de hardware necesarias para implementar los requerimientos del sistema.

Tabla 7

Requerimientos de arquitectura

Número	Descripción	Prioridad	Relacionado
SrAH1	El núcleo del Nodo Maestro debe desarrollarse en un lenguaje de alto rendimiento para soportar procesamiento concurrente (ej. C++).	Alta	SySR-RF10
SrAH2	El Nodo Maestro debe integrar un servidor de mensajería IoT ligero (Broker) para gestionar la comunicación asíncrona.	Alta	SySR-RF1, RF9
SrAH3	El Nodo Maestro debe disponer de una suite variada de algoritmos criptográficos (desde muy ligeros hasta robustos).	Alta	SySR-RF3
SrAH4	Las operaciones de cifrado deben basarse en bibliotecas de seguridad consolidadas y estandarizadas (ej. OpenSSL).	Alta	SySR-RF4, RNF3
SrAH5	El intercambio de datos de configuración y telemetría debe realizarse mediante formatos estructurados y ligeros (ej. JSON).	Alta	SySR-RF2, RF12
SrAH6	El firmware de los Nodos Cliente debe programarse utilizando los kits de desarrollo oficiales del fabricante para optimizar recursos.	Alta	SySR-RF6

SrAH7	Los Nodos Cliente deben incorporar una copia exacta de la suite de algoritmos criptográficos presentes en el Nodo Maestro.	Alta	SySR-RF7
SrAH8	El hardware del Nodo Maestro debe ser un equipo con mayor capacidad de cómputo que los sensores (ej. Placas SBC como Raspberry Pi).	Alta	SySR-RF3, RF10
SrAH9	El hardware del Nodo Maestro debe soportar un sistema operativo multitarea robusto (ej. Linux) para ejecutar el servidor y el motor lógico.	Alta	SrAH1, SrAH2
SrAH10	Los Nodos Cliente deben ser microcontroladores de bajo costo orientados a IoT, con conectividad inalámbrica nativa.	Alta	SySR-RF1, RF8
SrAH11	Los Nodos Cliente deben contar con memoria Flash y RAM suficiente para almacenar la lógica de red y las librerías criptográficas.	Media	SySR-RF6

Para dar cumplimiento al requerimiento arquitectónico SrAH3 establecido en la tabla anterior, el cual define el conjunto específico de algoritmos que el Nodo Maestro debe soportar, es necesario justificar la pertinencia de cada elección tecnológica frente a las restricciones del entorno.

3.2.4. *Criterios de selección de la suite criptográfica*

La selección de los diez algoritmos que componen la suite criptográfica del sistema no es arbitraria; responde a la necesidad de cubrir un espectro heterogéneo de capacidades de hardware,

desde dispositivos severamente restringidos hasta nodos con capacidad de procesamiento estándar. La suite se diseñó bajo un criterio de diversidad arquitectónica, seleccionando representativos de las familias más relevantes en la criptografía moderna y ligera. A continuación, se fundamenta la inclusión de cada algoritmo según su rol en la arquitectura adaptativa:

1. **Los Estándares de Referencia (Base de Comparación)**

Para evaluar la eficiencia de cualquier solución ligera, es imperativo compararla con los estándares de la industria.

- AES-128-CBC y AES-128-GCM: Se incluyen como la "línea base" de seguridad. AES es el estándar global; la variante GCM (Galois/Counter Mode) es crucial porque ofrece cifrado autenticado (AEAD) nativo, aprovechando la aceleración de hardware presente en microcontroladores modernos (como el ESP32), mientras que CBC representa el enfoque tradicional por bloques. Su presencia permite validar si los algoritmos ligeros realmente ofrecen una ventaja de rendimiento significativa frente a un estándar optimizado.

2. **La Nueva Generación de Criptografía Ligera (Estándares NIST)**

La tesis se alinea con el estado del arte incorporando los finalistas y ganadores del proceso de estandarización de criptografía ligera del NIST finalizado en 2023.

- Ascon-128a: Seleccionado por el NIST como el estándar oficial para criptografía ligera. Su inclusión es obligatoria para garantizar que el sistema cumpla con las normativas de seguridad futura. Es una construcción basada en esponja (sponge-based) optimizada para ofrecer AEAD con un consumo de memoria mínimo.

- Xoodyak: Finalista del concurso NIST y derivado de la familia Keccak (SHA-3). Se incluye para evaluar primitivas basadas en permutaciones criptográficas, ofreciendo una alternativa robusta en arquitecturas donde las operaciones de AES son costosas.

3. **Algoritmos de Alta Eficiencia en Software (Stream Ciphers)**

Muchos dispositivos IoT carecen de coprocesadores criptográficos dedicados. Para estos casos, se requieren algoritmos que sean rápidos mediante ejecución pura en software.

- ChaCha20-Poly1305: Es el estándar actual de facto para cifrado móvil y web cuando no hay aceleración AES. Su arquitectura ARX (Add-Rotate-Xor) es extremadamente eficiente en microcontroladores de propósito general.
- XSalsa20-Poly1305: Precursor de ChaCha20, se incluye por su uso de nonces extendidos (192 bits), lo cual mitiga riesgos de colisión de vectores de inicialización en redes IoT masivas donde la gestión de estados es compleja.

4. **Ultra-ligeros para Restricciones Extremas (Hardware/Software)**

Para nodos con recursos críticos (batería baja o procesadores de 8/16 bits), se seleccionaron algoritmos diseñados por la NSA específicamente para este nicho.

- Simon-64-128: Optimizado para rendimiento en hardware (FPGA/ASIC) pero flexible en software.
- Speck-64-128: La contraparte de Simon, optimizada específicamente para implementaciones en software eficiente en microcontroladores. Su

inclusión permite al sistema adaptativo bajar al nivel mínimo de consumo energético sin perder propiedades criptográficas de bloque.

5. **Mecanismos de Legado y Fallback (Resiliencia)**

Finalmente, la suite contempla escenarios de transición o emergencia.

- Blowfish-CBC: Aunque es un algoritmo legado, se mantiene por su estructura de bloques de 64 bits y su rapidez histórica en software. Sirve como punto de comparación para demostrar cómo los algoritmos modernos (como Speck) logran seguridad similar con menor huella de memoria.
- SHA256-XOR: Implementado como un mecanismo de "último recurso" o fallback. Aunque no es un estándar de cifrado per se, esta construcción (usando SHA-256 como generador de flujo de claves) permite mantener la confidencialidad en dispositivos que, por fallo o saturación extrema, no pueden ejecutar un cifrado de bloque completo, garantizando que el sistema nunca transmita en texto plano.

3.2.5. *Especificaciones Técnicas del Entorno de Hardware y Software*

Para garantizar la reproducibilidad del sistema y asegurar que la arquitectura de seguridad adaptativa opere con la latencia requerida, se han definido los requisitos mínimos de hardware y software. Esta especificación valida que la solución es desplegable tanto en servidores convencionales como en computadoras de placa reducida (SBC), favoreciendo su implementación como Gateway IoT de bajo consumo.

A. Requisitos del Nodo Maestro (Gateway / Controlador)

El Nodo Maestro actúa como el orquestador central. A diferencia de un servidor en la nube, este diseño contempla su ejecución en el borde (Edge Computing). Se ha validado el funcionamiento en arquitecturas ARM para demostrar eficiencia energética.

Tabla 8

Requisitos mínimos para el Nodo Maestro

Componente	**Especificación Mínima**	**Justificación Técnica**	**Relacionado**
Hardware Base	Raspberry Pi 3 Model B+ (o superior) / Mini PC x86	Requiere arquitectura multicore (ARMv8 o x86_64) para manejar hilos concurrentes del servidor Web y el Broker MQTT.	SySR-RF1, SySR-RF2
Procesador (CPU)	Quad Core 1.4 GHz (ARM Cortex-A53)	Necesario para ejecutar el motor de decisión y descifrado AES/Ascon de múltiples sensores sin introducir latencia significativa (>50ms).	SySR-RNF3
Memoria RAM	1 GB LPDDR2	El sistema operativo (Linux), el Broker Mosquitto y la aplicación C++ (con el historial en memoria) requieren aproximadamente 400-600 MB en carga pico.	SySR-RNF4
Almacenamiento	16 GB (Clase 10 microSD / SSD)	Espacio suficiente para el SO, logs de auditoría y persistencia de métricas a largo plazo si fuera necesario.	SySR-RF3, SySR-RNF1
Red	Ethernet 100 Mbps o Wi-Fi 802.11ac	Se prioriza Ethernet para el Gateway para garantizar estabilidad en la recepción de paquetes MQTT.	SySR-RF5

Sistema Operativo	Linux (Debian Bullseye / Ubuntu Server)	Soporte nativo para *sockets* POSIX, gestión de hilos (pthread) y bibliotecas de seguridad.	Relacionado
Librerías Clave	OpenSSL 1.1+, Paho MQTT C++, Crow (Boost)	Dependencias críticas para la compilación del binario del controlador.	SySR-RF1, SySR-RF2

B. Requisitos de los Nodos Sensores (Clientes IoT)

Los nodos clientes se caracterizan por ser dispositivos embebidos de recursos restringidos (Constrained Devices). La arquitectura de software se ha optimizado para funcionar en microcontroladores de 32 bits comunes en la industria.

Tabla 9

Requisitos mínimos para los Nodos Sensores

Componente	**Especificación Mínima**	**Justificación Técnica**	**Relacionado**
Hardware Base	SoC Espressif ESP32	Estándar de facto en IoT de bajo costo con pila TCP/IP integrada.	SySR-RF1, SySR-RNF2
Procesador (CPU)	80 MHz (Tensilica Xtensa LX106)	Velocidad mínima requerida para ejecutar operaciones XOR/Speck sin bloquear el bucle principal de adquisición de datos.	SySR-RNF3
Memoria RAM	50 KB (Disponible para usuario)	Crítico para manejar los buffers de cifrado y el tamaño de los paquetes MQTT (Payload + Headers).	SySR-RF2

Memoria Flash	4 MB	Espacio necesario para almacenar el firmware, las tablas de búsqueda (S-Boxes) de los algoritmos, la configuración OTA y la clave estática de aprovisionamiento (PSK).	SySR-RF4
Conectividad	Wi-Fi 802.11 b/g/n (2.4 GHz)	Protocolo de capa física para la transmisión de mensajes MQTT hacia el Gateway.	SySR-RF4, SySR-RNF1
Firmware/SDK	ESP-IDF (Recomendado) o Arduino Core	Entorno compatible con C/C++ estándar para la integración de la librería criptográfica ligera.	Relacionado

C. Hardware Seleccionado para el Prototipo

Para dar cumplimiento a las especificaciones mínimas detalladas anteriormente y validar la arquitectura en un entorno físico heterogéneo real, se seleccionaron los siguientes dispositivos:

a) **Nodo Maestro (Orquestador de Borde)** Para procesar las inferencias de la Lógica Difusa y la Red Neuronal en microsegundos sin depender de la nube, se seleccionó una placa SBC de alto rendimiento.

- **Dispositivo:** Orange Pi 5 Plus.
- **Procesador (SoC):** Rockchip RK3588 (Arquitectura ARM).
- **CPU:** Octa-core (4 núcleos Cortex-A76 a 2.4 GHz y 4 núcleos Cortex-A55 a 1.8 GHz).
- **Memoria RAM:** 16 GB LPDDR4/4x.
- **NPU (Unidad de Procesamiento Neuronal):** Integrada, con capacidad de hasta 6 TOPS, ideal para el procesamiento de redes neuronales en el borde.

b) **Nodos Cliente (Sensores de Borde heterogéneos)** Para validar la asignación criptográfica asimétrica, se seleccionaron tres variantes distintas de la familia de microcontroladores de Espressif Systems:

- **Nodo Cliente 1 (N1) - ESP32-CAM-MB:** Representa un nodo con periféricos multimedia. Cuenta con un procesador Xtensa Dual-Core 32-bit LX6 (hasta 240 MHz), 520 KB de SRAM y 4 MB de PSRAM externa.
- **Nodo Cliente 2 (N2) - ESP32-WROOM-32:** Actúa como el nodo de control estándar de la industria. Cuenta con un procesador Xtensa Dual-Core 32-bit LX6, 520 KB de SRAM interna y típicamente 4 MB de memoria Flash SPI.
- **Nodo Cliente 3 (N3) - Seeed Studio XIAO ESP32S3:** Simula un dispositivo "wearable" o ultra-compacto. Incorpora un procesador Xtensa Dual-Core 32-bit LX7 de nueva generación, 512 KB de SRAM y 8 MB de memoria Flash, destacando por su factor de forma miniaturizado y su eficiencia para operaciones vectoriales ligeras.

3.2.6. *Modelado y Análisis del Consumo Energético en el Borde*

En concordancia con el requerimiento StSR4 (minimizar la carga y prolongar la batería), resulta imprescindible modelar matemáticamente la viabilidad de la arquitectura propuesta. Para determinar el impacto real del mecanismo de seguridad adaptativa, el consumo total del dispositivo IoT se calcula en Julios (J). Este modelo integra de manera precisa el consumo de la red de comunicaciones, el procesamiento criptográfico de la CPU y los periodos de inactividad del microcontrolador en un ciclo de trabajo (Duty Cycle) de 10 segundos.

Desglose Estructural de la Carga Útil (Payload) y Sobrecarga de Red

Para que el cálculo matemático sea exacto, se ejecutó una inspección directa a la memoria del ESP (mediante la librería cJSON) y del Nodo Maestro (nlohmann/json), extrayendo el peso en bytes de la telemetría y los algoritmos, separando los datos puros del overhead criptográfico y de red:

1. **Fase de Registro Inicial (Tx - Mensaje HELLO)**

   El dispositivo se presenta ante el orquestador a través del tópico devices/.../hello para reportar sus capacidades físicas.

   - **Desglose de Datos Útiles (Estructura JSON):**
     - Identificador del dispositivo (device_id): 34 bytes.
     - Métricas del hardware (CPU, RAM, Batería, RSSI y fuente de alimentación): ~116 bytes.
     - Sintaxis de apertura y cierre: 2 bytes.
   - *(Subtotal - Payload MQTT Final):* **~152 bytes** *(Puede oscilar levemente según la cantidad de dígitos en los sensores).*
   - **(+) Cabeceras de Red (Overhead):** +77 bytes (~37 bytes de cabecera MQTT + 40 bytes del protocolo TCP/IP).
   - **= TAMAÑO TOTAL EN RED: ~229 Bytes**

2. **Fase de Transmisión Continua (Tx - Mensaje DATA)**

   Se transmite la telemetría capturada por los sensores hacia el tópico devices/.../data. Es el mensaje más pesado debido a la capa de seguridad.

   - **Datos Puros:** 173 bytes (Variables físicas de telemetría en texto plano).

- **(+) Expansión Criptográfica:** +199 bytes. *(El cifrado transforma los datos y les añade el Auth Tag de seguridad, lo cual al codificarse en Base64 infla el peso. Se incluyen aquí los 16 bytes del Nonce y ~104 bytes de Datos Adicionales de Autenticación - AAD - para evitar ataques de repetición).*
- **(+) Estructura JSON:** +84 bytes (Llaves y sintaxis de formato).
- *(Subtotal - Payload MQTT Final):* **~456 bytes.**
- **(+) Cabeceras de Red (Overhead):** +77 bytes (~37 bytes MQTT + 40 bytes TCP/IP).
- **= TAMAÑO TOTAL EN RED: ~533 Bytes** (4264 bits)

3. **Fase de Aprovisionamiento (Rx - Mensaje CONFIG)**

El Nodo Maestro responde al ESP32 asignando las políticas de seguridad a través del tópico devices/.../config.

- **Datos de Seguridad y Formato:** ~165 bytes. *(Esto incluye la Clave Dinámica ofuscada por XOR en Base64 que ocupa ~35 bytes, el string del algoritmo criptográfico con ~23 bytes, las banderas lógicas con ~50 bytes y la lista de soporte con ~57 bytes).*
- *(Subtotal - Payload MQTT Final):* **~165 bytes.**
- **(+) Cabeceras de Red (Overhead):** +77 bytes (~37 bytes MQTT + 40 bytes TCP/IP).
- **= TAMAÑO TOTAL EN RED: ~242 Bytes** (1936 bits)

Parámetros y Constantes del Entorno Experimental El cálculo se fundamenta en los siguientes parámetros operativos y corrientes medidas por los fabricantes para los SoCs seleccionados:

- Voltaje Lógico de Operación (V): $3.3\ V$.
- Tasa de Transferencia de Red (Bitrate): 60 Mbps, velocidad efectiva medida en el canal Wi-Fi del entorno de pruebas local.
- Tiempo de Cifrado de Telemetría $(t_{cifrado})$: 3 ms (0.003 s).
- Tiempo de Descifrado de Configuración $(t_{descifrado})$: 50 µs (0.00005 s).
- Tiempo del Ciclo Total (T_{total}): 10 s.
- Corrientes de Operación (ESP32): $I_{tx} = 180\ mA, I_{rx} = 10\ mA, I_{cpu} = 68\ mA, I_{sleep} = 0.8\ mA$.
- Corrientes de Operación (ESP32-S3): $I_{tx} = 290\ mA, I_{rx} = 90\ mA, I_{cpu} = 81.3\ mA, I_{sleep} = 0.24\ mA$.

Formulación Matemática del Consumo

El primer paso requiere la conversión de las corrientes extraídas del hardware (I_{mA}) a potencia activa real en Vatios (P) mediante la expresión:

Ecuación 1 Potencia

$$P = \frac{I_{mA} * V}{1000}$$

La energía de comunicación en el canal de radiofrecuencia (Wi-Fi) se define como la potencia de la antena multiplicada por el tiempo en el aire, el cual se calcula dividiendo la longitud del paquete entre la tasa de bits de la red:

Ecuación 2 Energía de transmisión

$$E_{tx} = P_{tx} * \frac{bits_{tx}}{bitrate}$$

Ecuación 3 Energía de recepción

$$E_{rx} = P_{rx} * \frac{bits_{rx}}{bitrate}$$

La energía demandada por la lógica de seguridad adaptativa se obtiene integrando la potencia de la CPU durante los tiempos exactos de ejecución de las rutinas criptográficas extraídas de la telemetría:

Ecuación 4 Energía de cifrado

$$E_{cifrado} = P_{cpu} * t_{cifrado}$$

Ecuación 5 Energía de descifrado

$$E_{descifrado} = P_{cpu} * t_{descifrado}$$

El estado de reposo (*Light Sleep*), donde el microcontrolador desactiva los periféricos de radiofrecuencia, abarca el resto de la ventana de transmisión. El tiempo de suspensión se obtiene restando las fases activas al periodo total:

Ecuación 6 Tiempo en reposo ligero

$$t_{sleep} = T_{total} - (\frac{bits_{tx}}{bitrate} + \frac{bits_{tx}}{bitrate} + t_{cifrado} + t_{descifrado})$$

Ecuación 7 Energía en reposo ligero

$$E_{sleep} = P_{sleep} * t_{sleep}$$

Finalmente, la energía total (E_{total}) demandada por el Nodo Cliente por cada iteración es la sumatoria de todas las etapas operativas:

Ecuación 8 Energía total

$$E_{total} = E_{cifrado} + E_{tx} + E_{rx} + E_{descifrado} + E_{sleep}$$

Tabla 10

Perfil de potencia eléctrica de los Nodos Cliente por estado operativo

Dispositivo IoT	**Potencia de Transmisión (Ptx)**	**Potencia de Recepción (Prx)**	**Potencia de CPU (Pcpu)**	**Potencia Light Sleep (Psleep)**
N1 / N2 (ESP32 Estándar)	0.594 W	0.330 W	0.2244 W	0.00264 W
N3 (XIAO ESP32-S3)	0.957 W	0.297 W	0.2682 W	0.00079 W

Nota. La tabla detalla la conversión a potencia real asumiendo un voltaje lógico de 3.3 V. Se evidencia que la arquitectura Xtensa LX7 del ESP32-S3 demanda una potencia de transmisión un 61% mayor (0.957 W) respecto a la versión estándar (0.594 W), pero compensa esta diferencia con un estado de reposo altamente optimizado (0.00079 W contra 0.00264 W).

Tabla 11

Desglose del costo energético en Julios por ciclo de transmisión seguro de 10 segundos

Fase Operativa del Ciclo	**Tiempo de Ejecución (t)**	**Energía N1 / N2 (ESP32)**	**Energía N3 (ESP32-S3)**
Cifrado de Telemetría	0.003 s	0.000673 J	0.000805 J
Transmisión de Datos (Tx)	(0.000071 s)	0.000042 J	0.000068 J
Recepción Config. (Rx)	(0.000032 s)	0.000011 J	0.000010 J
Descifrado de Config.	0.00005 s	0.000011 J	0.000013 J
Reposo (Light Sleep)	≈9.9968 s	0.026392 J	0.007897 J

ENERGÍA TOTAL POR CICLO	10 s	0.027129 J	0.008793 J

Nota. Análisis matemático consolidado para un ciclo operativo concurrente de 10 segundos operando a 60 Mbps. Los resultados empíricos proyectados revelan que el impacto del procesamiento criptográfico (suma de cifrado y descifrado) representa una fracción minúscula de la energía total, validando la eficiencia de los algoritmos ligeros. Adicionalmente, se demuestra que, debido al prolongado tiempo de inactividad (≈ 9.99 s), el ESP32-S3 resulta sustancialmente más eficiente en la visión macro del sistema, requiriendo un 67% menos de energía total por ciclo (0.0088 J) en comparación con el modelo estándar (0.0271 J), justificando su despliegue en entornos IoT alimentados por baterías.

Tabla 12

Análisis de Consumo Energético en el Nodo Maestro

Fase Operativa del Ciclo	**Tiempo de Ejecución (s)**	**Energía Orange Pi 5 (J)**
Recepción MQTT (Rx)	0.000150	0.000300
Descifrado de Telemetría	0.000051	0.000254
Procesamiento y WebSocket	0.001000	0.005000
Reposo (Idle)	9.998799	14.998199
ENERGÍA TOTAL POR CICLO	10.000000	15.003753

Nota. Los datos obtenidos revelan el verdadero peso de la carga criptográfica en la infraestructura de borde. Como se observa en la Tabla 12, de los 15.003753 J consumidos en total durante el ciclo de 10 segundos, el proceso matemático de descifrado de la telemetría apenas requiere 0.000254 J.

3.3 Etapa de Diseño del Sistema

Siguiendo la metodología en cascada, la Etapa de Diseño del Sistema traduce los requerimientos funcionales (SySR-RF), no funcionales (SySR-RNF) y de arquitectura (SrAH) definidos en la sección 3.2, en una arquitectura técnica y un diseño modular.

El diseño se enfoca en un modelo de gestión centralizado (Nodo Maestro) que orquesta la seguridad de los dispositivos cliente (Nodos Cliente) a través del protocolo MQTT, permitiendo la selección dinámica de algoritmos criptográficos ligeros.

El diseño del sistema se desglosa en la siguiente estructura:

- Arquitectura Lógica: Define los componentes principales y sus interrelaciones.
- Módulos Principales del Sistema: Desglosa el software del Nodo Maestro y del Nodo Cliente en sus bloques funcionales.
- Diagramas de Diseño: Ilustra los flujos de comunicación y lógica interna del sistema.

3.3.1. *Arquitectura Lógica del Sistema*

La arquitectura lógica describe la organización de alto nivel del sistema, identificando los componentes de hardware y software y las interfaces de comunicación entre ellos. La arquitectura propuesta (reflejada en el Requerimiento SrAH10) se basa en un modelo cliente-servidor, donde el Nodo Maestro actúa como el servidor central de orquestación.

Los componentes principales de esta arquitectura son:

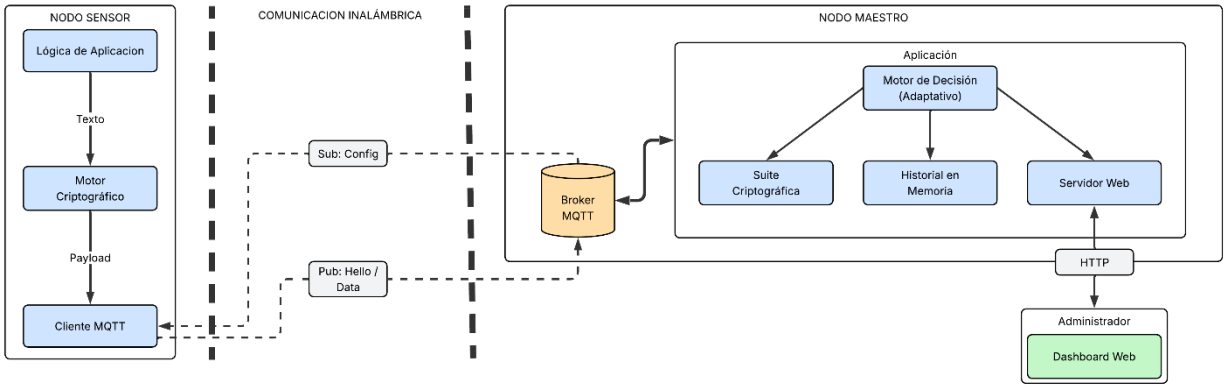
1. Nodo Maestro: Es el componente central del sistema, responsable de la gestión, monitoreo y orquestación de la seguridad. Ejecuta el sistema operativo Linux y aloja la aplicación principal desarrollada en C++, el broker MQTT y el dashboard web.

2. Nodos Cliente: Son los dispositivos IoT de recursos limitados (como las familias ESP32 o ESP8266) desplegados en el entorno. Su función es recolectar datos, reportar sus capacidades, recibir directivas de seguridad del Maestro y cifrar los datos de la aplicación.

3. Protocolo de Comunicación (MQTT): Actúa como el bus de comunicación asíncrono que interconecta al Maestro y a los Clientes. El diseño se basa en una estructura de tópicos MQTT específica para el registro, configuración y envío de datos.

Figura 4

Arquitectura lógica del sistema



Nota. Representación del modelo cliente-servidor implementado para la seguridad adaptativa. El Nodo Maestro centraliza la toma de decisiones utilizando un motor lógico y una suite criptográfica basada en OpenSSL para gestionar los Nodos Sensores (ESP32/ESP8266). La comunicación se realiza mediante el protocolo MQTT, separando los flujos de registro, configuración de seguridad y transmisión de datos cifrados. Fuente: Elaboración propia.

3.3.2. *Módulos Principales del Sistema*

Para cumplir con los requerimientos establecidos, la arquitectura lógica se implementa a través de un conjunto de módulos de software específicos, tanto en el Nodo Maestro como en los Nodos Cliente.

A. Módulos del Nodo Maestro

La aplicación principal en C++ se desglosa en los siguientes módulos funcionales, identificados a partir del código web_dashboard.cpp y crypto_suite.cpp:

1. Módulo de Conectividad (Cliente MQTT): Utiliza la biblioteca Paho MQTT para conectarse al broker local a través del puerto estándar 1883. Es responsable de gestionar las suscripciones a los tópicos devices/+/hello y devices/+/data.

2. Módulo de Decisión (Motor Lógico Adaptativo): Corresponde a la función CryptoSuite::choose_algorithm. Este módulo abandona los umbrales estáticos tradicionales e integra un motor de decisión basado en Inteligencia Artificial. Para los fines de esta investigación, el módulo se ha diseñado para operar bajo dos enfoques matemáticos que se evalúan de forma comparativa: un modelo de **Lógica Difusa** y un modelo de **Red Neuronal Artificial**. Ambos motores reciben la telemetría del cliente (CPU, RAM disponible, nivel de batería y RSSI) y calculan la categoría criptográfica óptima a asignar.

3. Módulo Criptográfico (CryptoSuite): Implementa la lógica de CryptoSuite::generate_key_for_algo y CryptoSuite::open_secure_payload. Contiene las implementaciones de descifrado para todos los algoritmos soportados (p.ej., Ascon, AES-GCM, Speck, etc.).

4. Módulo de Gestión de Estado: Mantiene el estado actual de los dispositivos conectados en la variable global devices (un std::map). Almacena el algoritmo asignado, la clave y el contador de mensajes por dispositivo.

5. Módulo de Persistencia (Historial): Gestiona un búfer circular (std::deque<HistoryEntry>) que almacena las últimas 500 (MAX_HISTORY_SIZE) transacciones procesadas, cumpliendo con SySR-RF11.

6. Módulo de Interfaz Web (Servidor Crow): Implementa el servidor web, exponiendo:

   - API REST: Endpoints /api/devices y /api/history para la carga inicial del dashboard.
   - WebSocket: El endpoint /ws para la difusión de actualizaciones en tiempo real (broadcast_ws_message) a los clientes web.

7. Módulo de Aprovisionamiento Seguro (PSK Bootstrapping): Implementa la lógica de protección para la entrega de llaves. Este módulo asegura que el primer contacto entre el Maestro y el Sensor sea robusto; utiliza una clave maestra estática para cifrar la clave dinámica seleccionada por la IA. Al emplear una operación lógica XOR para este proceso, el sistema minimiza el consumo de RAM en el nodo sensor, eliminando por completo la necesidad de túneles TLS/SSL pesados o el uso del puerto seguro 8883 sin sacrificar la confidencialidad de la clave de sesión.

B. Módulos del Nodo Cliente

El firmware de los Nodos Cliente (ESP-IDF) se diseña con los siguientes módulos:

1. Módulo de Capacidades: Responsable de consultar el hardware local (frecuencia de CPU, RAM disponible) para construir el mensaje hello.

2. Módulo de Conectividad (Cliente MQTT): Utiliza la biblioteca esp-mqtt para establecer y mantener la conexión con el Broker MQTT del Maestro. Se suscribe a su tópico de configuración único (p.ej., devices/ESP32_A1B2/config) y publica en su tópico de datos.

3. Módulo de Configuración y Desencriptación PSK: Es el *callback* de la suscripción MQTT. Su función principal es recibir el JSON de configuración y, mediante la clave estática almacenada en su memoria interna, realizar el descifrado inverso de la variable key_b64. Una vez obtenida la clave de sesión real, el módulo actualiza el motor criptográfico local para iniciar la transmisión de telemetría protegida.

4. Módulo Criptográfico (Suite Cliente): Biblioteca local que contiene las implementaciones de cifrado para todos los algoritmos de la suite. Este módulo lee la configuración del Módulo de Configuración para cifrar los datos bajo demanda.

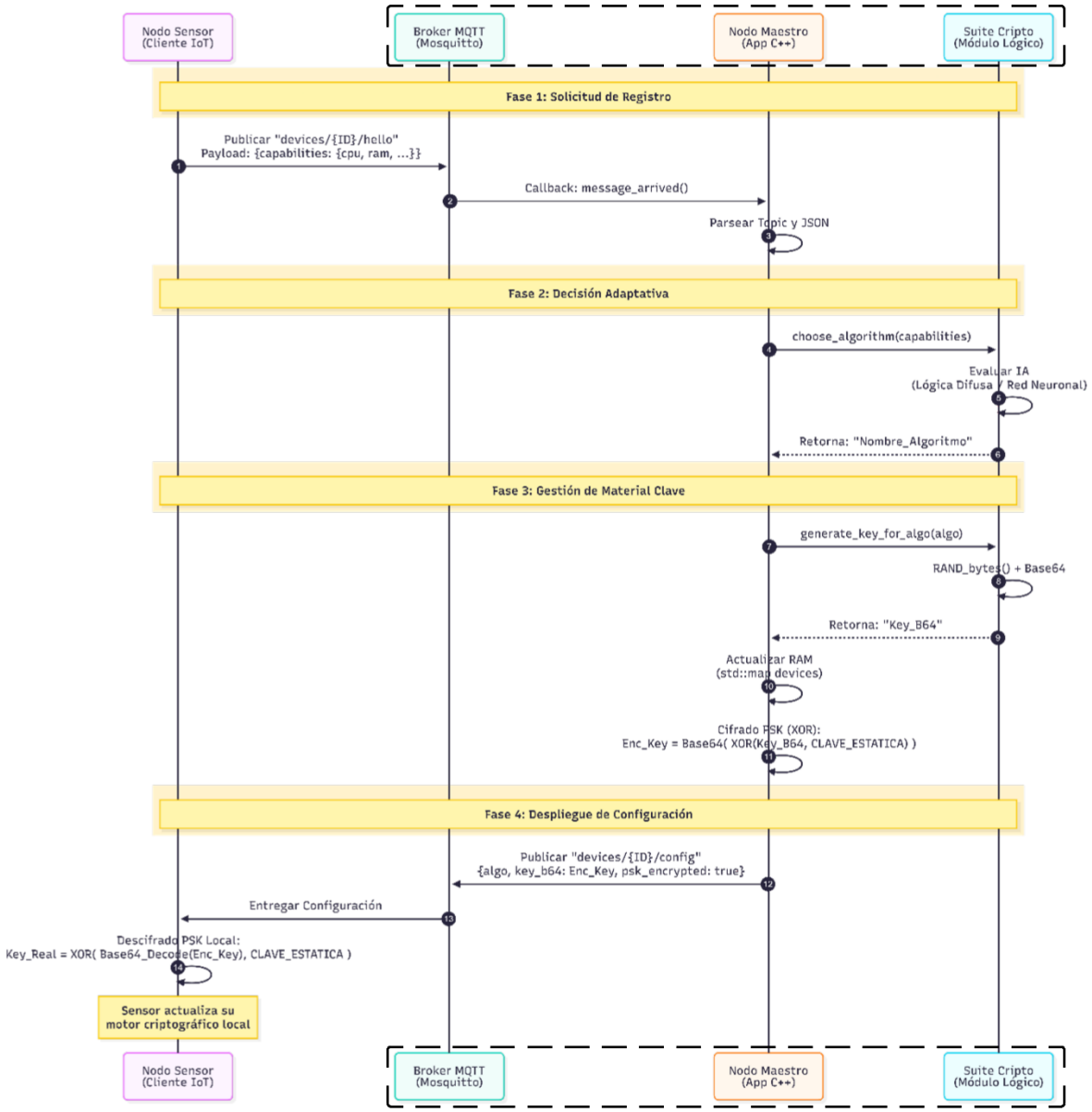
5. Módulo de Aplicación (Sensor): La lógica principal del dispositivo (p.ej., leer un sensor, generar un frame_id) que prepara el payload en texto plano antes de pasarlo al Módulo Criptográfico.

3.3.3. *Diagramas de Diseño*

Para detallar el comportamiento del sistema, se utilizan diagramas de flujo y secuencia que ilustran cómo interactúan los módulos para cumplir con los requerimientos funcionales.

Figura 5

Diagrama de secuencia - Registro y asignación de algoritmo

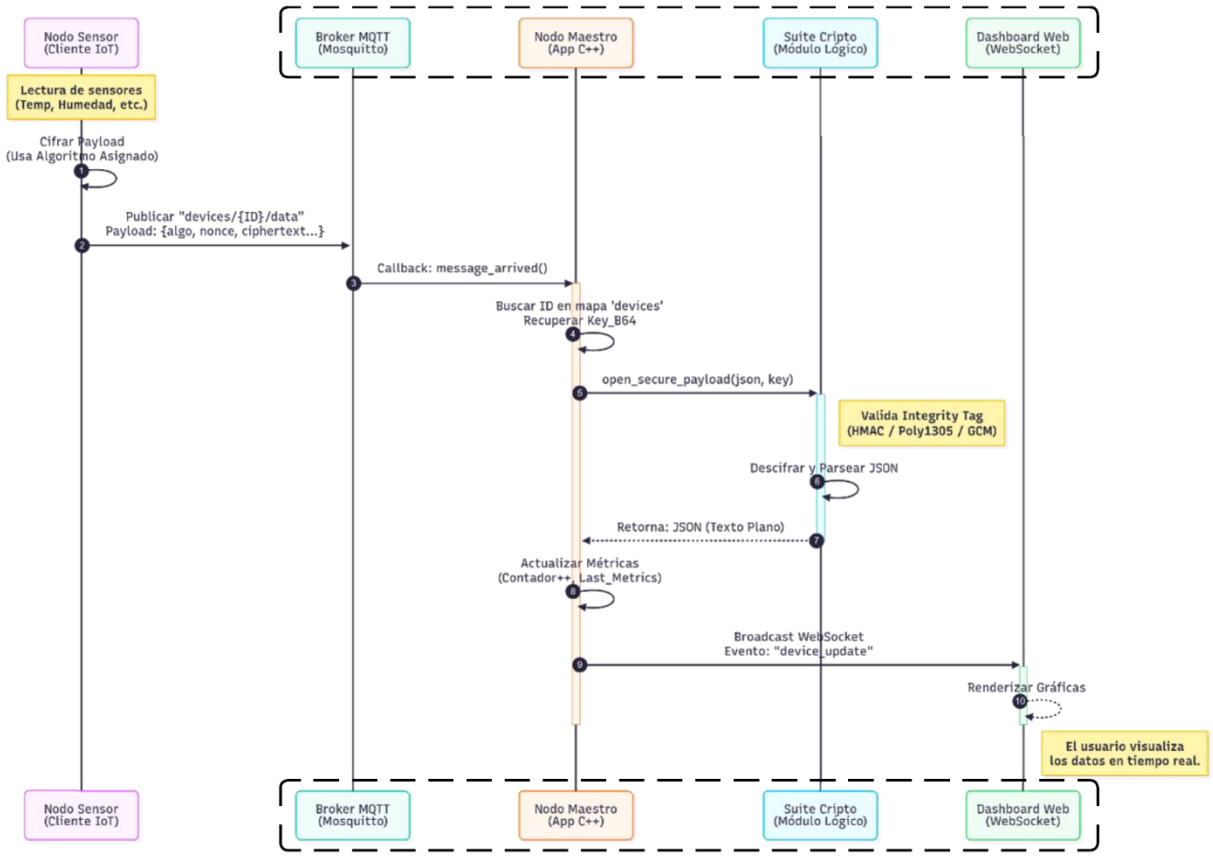


Nota. El flujo describe el proceso de negociación de seguridad iniciado por el Nodo Sensor. El Nodo Maestro procesa las capacidades de hardware recibidas (CPU y RAM) y ejecuta la función choose_algorithm para determinar el esquema de cifrado óptimo. Posteriormente, genera material clave criptográficamente seguro y, antes de su distribución, aplica un cifrado de transporte inicial

(PSK Bootstrapping). Utilizando una clave estática precompartida alojada en el firmware y un algoritmo de cifrado ligero (XOR), el sistema ofusca la clave de sesión dentro del JSON de configuración, garantizando que el material criptográfico sea indescifrable ante ataques de intercepción en el canal MQTT.

Figura 6

Diagrama de secuencia - Envío de datos y monitoreo

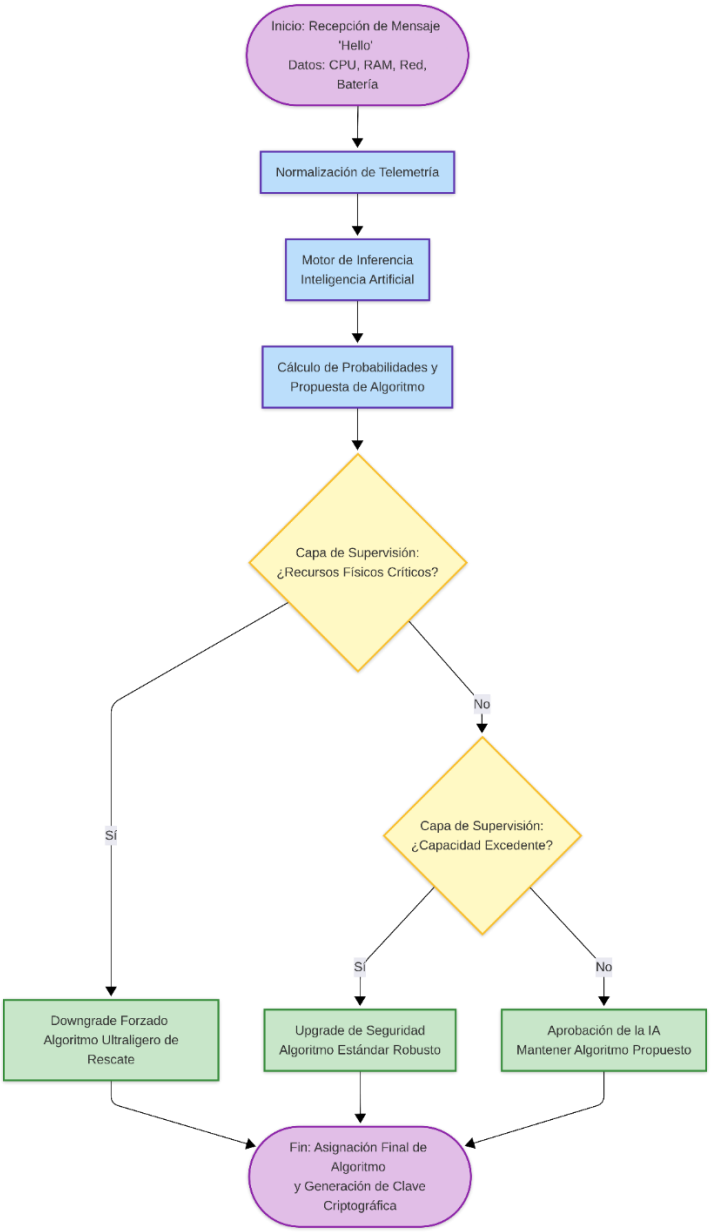


Nota. El proceso detalla la transmisión segura de información. El Nodo Sensor cifra las lecturas locales utilizando el algoritmo previamente negociado y las envía al Broker MQTT. Al recibir el mensaje, el Nodo Maestro recupera la clave simétrica almacenada en memoria y utiliza la suite criptográfica para verificar la integridad del mensaje y descifrar el contenido. Finalmente, los datos

en texto plano se procesan y se difunden a la interfaz de usuario a través de una conexión WebSocket para su monitoreo en tiempo real. Fuente: Elaboración propia.

Figura 7

Diagrama de flujo del motor de decisión.



Nota. Lógica de selección implementada en el módulo choose_algorithm. El sistema procesa simultáneamente las variables multimodales del dispositivo (CPU, RAM, RSSI, Batería) a través del bloque de Inteligencia Artificial (Lógica Difusa / Red Neuronal). El flujo prioriza algoritmos ultraligeros (como Speck o Simon) para hardware muy restringido y escala hacia estándares robustos (como AES-GCM o Ascon) conforme aumentan los recursos disponibles, garantizando siempre el equilibrio entre seguridad y rendimiento. Fuente: Elaboración propia.

3.4 Etapa de Implementación

La fase de implementación materializa la arquitectura lógica diseñada en la sección anterior, traduciendo los requerimientos funcionales y los algoritmos de decisión en componentes de software operativos. Este proceso se llevó a cabo mediante un enfoque modular, asegurando la independencia entre la lógica de comunicación, el motor criptográfico y la interfaz de gestión. A continuación, se detallan el entorno de desarrollo establecido y la construcción de los módulos constitutivos del Nodo Maestro.

3.4.1. *Entorno de desarrollo*

Para garantizar la reproducibilidad de los resultados y el rendimiento en tiempo real del sistema, se configuró un entorno de desarrollo basado en estándares de código abierto. La selección de herramientas priorizó la eficiencia en la gestión de memoria y el acceso a primitivas criptográficas de bajo nivel.

- Plataforma de Hardware: Para la orquestación central (Nodo Maestro) se empleó un ordenador de placa reducida (SBC) **Orange Pi 5 Plus (16GB RAM)** con arquitectura ARM (SoC Rockchip RK3588), el cual simuló eficientemente un servidor de borde (Edge Server) ejecutando la Inteligencia Artificial. Para la capa

de percepción (Nodos Cliente), se utilizó un ecosistema heterogéneo compuesto por tres placas: un **ESP32-CAM-MB**, un **ESP32-WROOM-32** y un microcontrolador ultra-compacto **XIAO-ESP32-S3**.

- Lenguaje de Programación: Se seleccionó C++ (estándar C++17) debido a su capacidad para el manejo determinista de recursos y su compatibilidad nativa con bibliotecas de sistemas.
- Bibliotecas y Dependencias:
  - OpenSSL (v3.0): Proveedor de las primitivas criptográficas estándar (AES, SHA, HMAC) y del generador de números aleatorios criptográficamente seguro (CSPRNG) utilizado por el Nodo Maestro (OpenSSL Software Foundation, 2023).
  - mbedTLS: Biblioteca criptográfica optimizada para sistemas embebidos, utilizada nativamente en el Nodo Sensor para el soporte de operaciones criptográficas ligeras y decodificación en Base64 (Arm Limited, 2023).
  - Eclipse Paho MQTT C++: Middleware implementado para la gestión asíncrona y publicación/suscripción de la mensajería IoT (Eclipse Foundation, 2023).
  - nlohmann/json: Motor en C++ para la serialización y parseo de objetos JSON de alto rendimiento, fundamental para la estructuración de la telemetría (Lohmann, 2023).
  - Crow: Micro-framework implementado para el despliegue concurrente de la API REST y los servicios WebSocket del Dashboard Web (CrowCPP, 2023).
  - ESP-IDF (Espressif IoT Development Framework): Entorno de desarrollo oficial utilizado para compilar el firmware del Nodo Sensor (ESP32),

permitiendo el manejo eficiente de la pila TCP/IP y la memoria en tiempo real (Espressif Systems, 2023).

3.4.2. *Implementación de los módulos*

La codificación del sistema se estructuró en cuatro módulos principales que interactúan entre sí para dar cumplimiento a los objetivos de seguridad adaptativa.

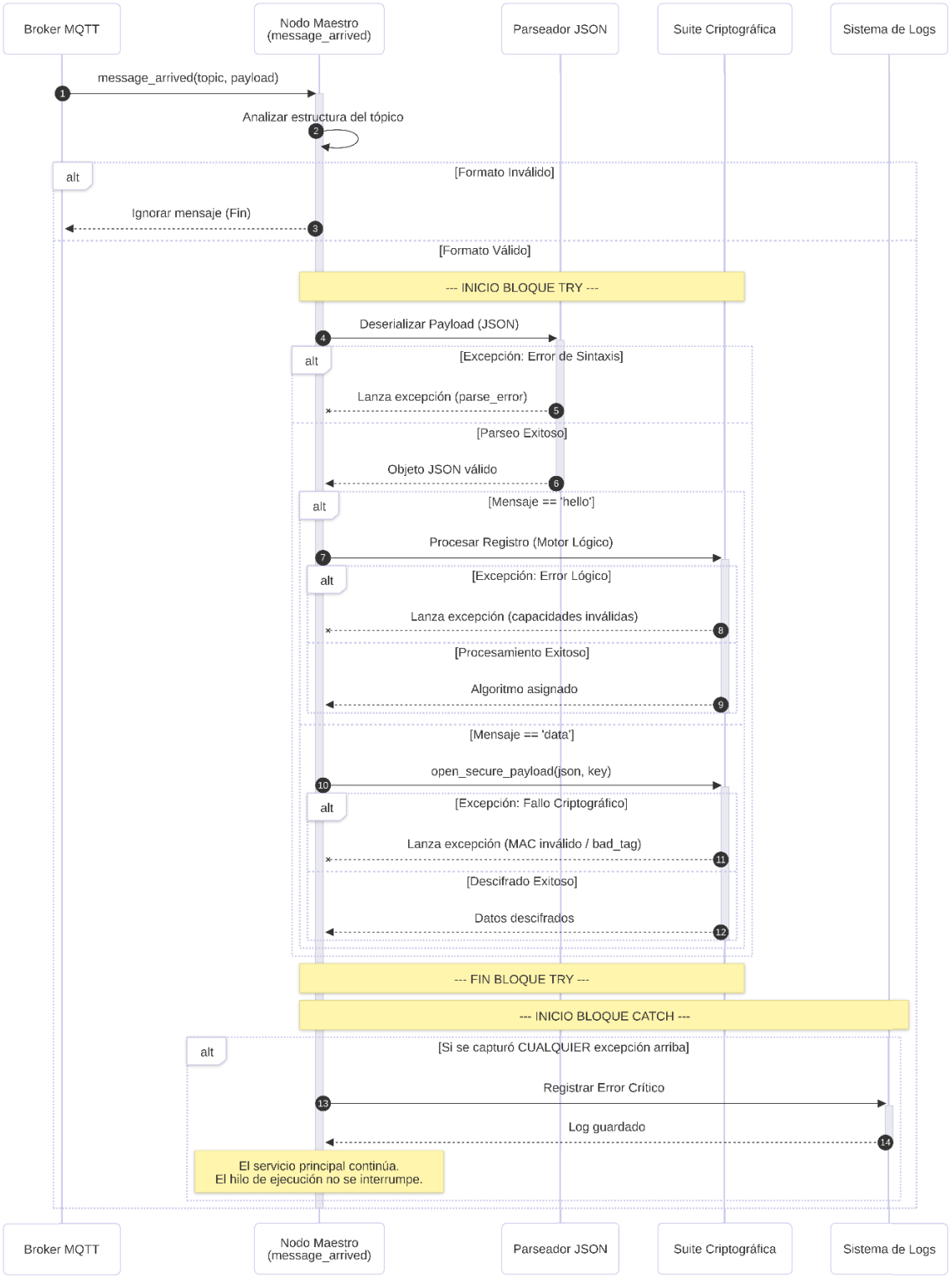
A. Módulo de Gestión de Mensajería (Cliente MQTT)

La comunicación asíncrona con los dispositivos IoT se implementó a través de la clase MqttCallback, la cual hereda de la interfaz mqtt::callback. Este componente es responsable de mantener la persistencia de la conexión con el broker y de orquestar la recepción de mensajes en los tópicos de control (devices/+/hello) y datos (devices/+/data).

Para asegurar la robustez del sistema frente a datos corruptos o ataques de inyección de formatos inválidos, se implementó un mecanismo de manejo de excepciones que encapsula el procesamiento de los payloads. Esto garantiza la disponibilidad continua del servicio del Nodo Maestro, evitando interrupciones críticas durante la operación.

Figura 8

Diagrama de secuencia del mecanismo de robustez (Callback MQTT).



Nota. Lógica interna del método message_arrived implementado en el Nodo Maestro. El diagrama evidencia el mecanismo de protección mediante bloques de excepción (try-catch), asegurando que errores críticos como payloads corruptos, fallos de parseo JSON o inconsistencias criptográficas sean capturados y registrados sin interrumpir la ejecución del servicio principal. Fuente: Elaboración propia.

B. Motor de Decisión Adaptativo

El núcleo de la propuesta reside en el mecanismo de selección dinámica, implementado a través de un motor de inferencia basado en Inteligencia Artificial (IA) dentro del componente de decisión criptográfica. Este módulo recibe como entrada la telemetría en tiempo real del dispositivo (uso de CPU, RAM disponible, calidad de red y nivel de energía), normaliza estos valores y los procesa a través del modelo inteligente para calcular la asignación criptográfica más segura y eficiente.

El proceso de decisión abandona las reglas estáticas y se estructura en una arquitectura híbrida de tres etapas fundamentales:

Inferencia Inteligente (Evaluación Dual y Optimización Edge)

Para materializar la inferencia inteligente del Motor de Decisión, se desarrollaron y contrastaron dos enfoques matemáticos independientes. Ambos modelos fueron estrictamente optimizados para su ejecución en entornos de *Edge Computing*, garantizando respuestas en microsegundos sin depender de servidores en la nube.

- **Modelo de Lógica Difusa (Optimización y Álgebra Difusa):** El diseño de este motor de inferencia requirió un equilibrio estricto entre la precisión en la toma de decisiones y el

costo computacional. En la teoría clásica de conjuntos difusos, el espacio total de reglas se define mediante el principio multiplicativo $R_{max} = m^n$; dado que el sistema evalúa 4 variables de telemetría con 3 conjuntos difusos cada una, el hiperespacio generaría un total de 81 reglas posibles. Evaluar secuencialmente esta cantidad de sentencias introduce un cuello de botella conocido como explosión combinatoria, inaceptable para un entorno de tiempo real.

Para resolver esta ineficiencia, la arquitectura del sistema reemplaza la declaración explícita por una condensación lógica utilizando operadores difusos avanzados (T-Normas y S-Normas). Las 81 combinaciones fueron colapsadas en tres reglas maestras que abarcan el 100% del espacio de soluciones:

- ***Regla de Emergencia (Operador OR):*** Si cualquier recurso cruza un umbral crítico (CPU alta, RAM crítica, batería baja o red inestable), se asigna inmediatamente seguridad LIGERA. El uso del operador de unión (OR) permite reaccionar ante el "eslabón más débil" del hardware, ignorando la holgura de las demás variables para evitar el colapso del nodo.
- ***Regla de Excelencia (Operador AND):*** Exige la concurrencia simultánea de condiciones óptimas (operador de intersección) en todos los vectores físicos del dispositivo antes de autorizar seguridad PESADA, evitando someter al microcontrolador a estrés térmico o energético innecesario.
- ***Regla por Exclusión (Operador NOT):*** Evaluando los estados de equilibrio mediante lógica de exclusión matemática (combinando negaciones de la emergencia y la excelencia), el espacio difuso restante se clasifica en la categoría MEDIA.

Esta condensación arquitectónica redujo el tamaño del motor de inferencia en más de un 95%, reduciendo drásticamente las operaciones de coma flotante por segundo (FLOPS) y permitiendo la orquestación de cientos de dispositivos sin degradar la latencia.

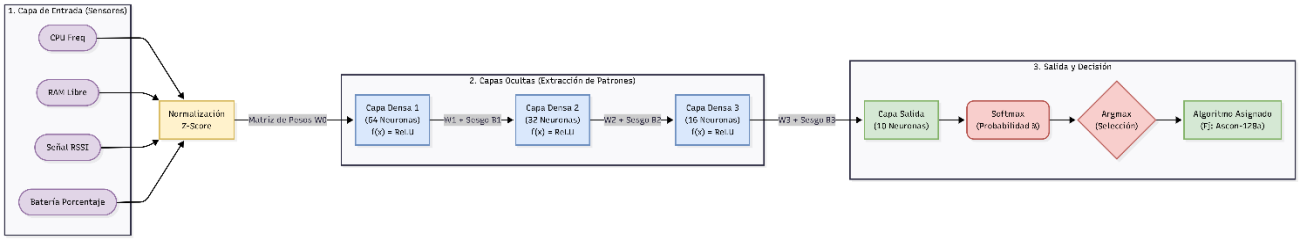
- **Modelo de Red Neuronal Artificial (Arquitectura "Bare Metal"):** Para dotar al Nodo Maestro de una toma de decisiones predictiva, se implementó una Red Neuronal Prealimentada (*Feedforward*) estructurada como un Perceptrón Multicapa (MLP). Este modelo es capaz de identificar relaciones no lineales y degradaciones sutiles entre los recursos de hardware y la exigencia criptográfica. Su diseño se articula de la siguiente manera:
    - ***Topología de Compresión:*** La arquitectura fue diseñada bajo una estrategia de "embudo", forzando a la red a extraer los patrones más importantes reduciendo gradualmente su dimensionalidad. Consta de una capa de entrada de 4 neuronas (telemetría normalizada), tres capas ocultas densamente conectadas (de 64, 32 y 16 neuronas respectivamente) y una capa de salida de 10 neuronas, representando unívocamente a cada algoritmo de la suite.
    - ***Mecánica Matemática y Funciones de Activación:*** En el procesamiento interno, cada neurona calcula una combinación lineal sumando el producto de las señales por sus pesos sinápticos y añadiendo un sesgo ($z = \sum(w_i * x_i) + b$). Para otorgar no linealidad y asegurar la viabilidad en C++, las 112 neuronas de las capas ocultas utilizan la función ReLU ($f(x) = \max(0, x)$), la cual consume escasos ciclos de reloj de CPU frente al cálculo tradicional de exponenciales. Por su parte, la capa de salida emplea la función Softmax para normalizar los valores crudos,

transformándolos en una distribución de probabilidad (100%) que expone el índice de confianza de la predicción de la IA.

- ***Ventaja Computacional (Edge AI):*** El mayor logro a nivel de software de esta implementación radica en el aislamiento total de dependencias. En lugar de importar motores pesados de *Machine Learning* como TensorFlow o PyTorch (los cuales exigirían megabytes de RAM), el "cerebro" matemático y sus aproximadamente 3,000 parámetros entrenables fueron exportados a tensores estáticos en un archivo de cabecera nativo de C++ (pesos_nn.h). El sistema reconstruye la pasada hacia adelante (*Forward Pass*) utilizando exclusivamente estructuras estándar y álgebra lineal pura. El resultado es un motor de clasificación que ocupa menos memoria que una imagen comprimida y se ejecuta en microsegundos, materializando con éxito el concepto de Inteligencia Artificial en el Borde (*Edge AI*).

Figura 9

Topología del Perceptrón Multicapa (MLP) y flujo del proceso de inferencia para la selección criptográfica en el borde



Capa de Supervisión Técnica (Salvaguarda de Hardware)

Se incorpora una lógica de protección reactiva. Si el motor de IA propone un algoritmo de nivel superior, pero los sensores físicos detectan una saturación crítica e inminente en los recursos del nodo (por ejemplo, un agotamiento severo de la memoria RAM o un colapso en la batería), esta capa interviene y fuerza un downgrade de seguridad hacia algoritmos ultraligeros (como Speck o SHA256-XOR). Esto garantiza la supervivencia y operatividad continua del nodo frente a falsos positivos o picos de estrés.

Optimización por Capacidad Excedente

De manera complementaria, si el dispositivo opera en condiciones ideales (por ejemplo, con alimentación eléctrica estable y amplia disponibilidad de procesamiento), la capa de supervisión aprueba un upgrade hacia estándares criptográficos más robustos (como AES-128-GCM o Ascon-128a), maximizando la protección de los datos sin comprometer la latencia del sistema.

Tabla 13

Umbrales de decisión para la asignación de algoritmos criptográficos

Categoría de Seguridad (Decisión IA)	**Algoritmos Pertenecientes**	**Contexto de Asignación (Inferencia + Supervisión)**	**Tipo de Cifrado**
Nivel BAJO (Prioridad: Ahorro Energético y Supervivencia)	SHA256-XOR Speck-64-128 Simon-64-128	Asignado por la IA en escenarios de recursos limitados, o forzado por la capa de supervisión cuando se detecta	Híbrido (Simulado), Bloque Ligero

		saturación física crítica (ej. RAM < 25% o Batería < 20%) para evitar la caída del nodo.	
Nivel MEDIO (Prioridad: Equilibrio Seguridad/Rendimiento)	AES-128-CBC ChaCha20-Poly1305 XSalsa20-Poly1305 AES-128-GCM	Asignado por la IA cuando los recursos de hardware y conectividad son estables, permitiendo sostener cifrados estándar y de flujo continuo sin riesgo de degradación del servicio.	Estándar (Bloque / AEAD), Flujo (Stream)
Nivel ALTO (Prioridad: Máxima Confidencialidad y Robustez)	Ascon-128a Blowfish-CBC Xoodyak	Sugerido por la IA para nodos con hardware superior, o aplicado como un upgrade de seguridad por la capa de supervisión cuando existe capacidad excedente (ej. CPU libre > 85% y alimentación eléctrica de red).	Estándar Ligero (NIST), Criptografía Permutada, Bloque (Legado)

Nota. La tabla detalla la clasificación de la suite criptográfica gestionada por el motor de Inteligencia Artificial. A diferencia de un sistema de umbrales estáticos, el modelo evalúa simultáneamente múltiples variables (CPU, RAM, RSSI y Batería) para inferir la categoría óptima. La capa de supervisión técnica actúa como un mecanismo de control de fallos, garantizando

descensos de nivel (downgrades) en emergencias de hardware, o incrementos (upgrades) cuando la capacidad física lo permite.

C. Suite Criptográfica (CryptoSuite)

Este módulo actúa como una capa de abstracción sobre la biblioteca OpenSSL. Se desarrollaron rutinas específicas para gestionar la heterogeneidad de los algoritmos soportados. Destaca la función open_secure_payload, diseñada bajo un patrón polimórfico que permite descifrar indistintamente esquemas con autenticación nativa (AEAD) y esquemas construidos manualmente.

Para los algoritmos que carecen de soporte AEAD nativo (como las implementaciones ligeras simuladas), se integró una verificación de integridad obligatoria mediante HMAC-SHA256 antes del proceso de descifrado. Esto asegura que ningún mensaje manipulado sea procesado, cumpliendo con el principio de integridad de la información.

D. Interfaz de Monitoreo y Control

La visualización del estado del sistema se implementó mediante un servidor web embebido basado en la biblioteca Crow. Este módulo expone una API REST para la consulta de dispositivos conectados y gestiona un canal WebSocket para la transmisión de telemetría en tiempo real. Esta arquitectura permite al administrador observar instantáneamente los cambios de algoritmo y el flujo de datos descifrados, validando la transparencia del sistema adaptativo.

3.4.3. *Validación preliminar del prototipo*

Tras la integración de los módulos, se ejecutaron pruebas de verificación funcional (Smoke Tests) para validar la interoperabilidad de los componentes:

1) Validación del Handshake: Se verificó la correcta negociación de parámetros mediante el análisis de logs del sistema, confirmando que el Nodo Maestro identifica las capacidades y asigna el algoritmo correspondiente.
2) Integridad Criptográfica: Se comprobó que el sistema rechaza automáticamente cualquier payload cuya etiqueta de autenticación (auth_tag) no coincida, validando la eficacia de la protección contra manipulación.

Capítulo IV: Pruebas de Funcionamiento

El presente capítulo expone los resultados de la validación experimental del mecanismo de seguridad adaptativa propuesto. El objetivo es comprobar empíricamente que la arquitectura diseñada cumple con los requerimientos de eficiencia, adaptabilidad y resiliencia en un entorno de hardware restringido.

Para garantizar una evaluación sistemática y estructurada, el proceso de validación se dividió en cinco casos de prueba progresivos. Se inició con la comprobación teórica de los modelos de Inteligencia Artificial mediante simulaciones en Python, avanzando hacia pruebas de estrés concurrente en C++, hasta culminar con auditorías de intercepción inalámbrica en la capa de red. En la Tabla 14 se presenta la matriz de ruta que guía la ejecución y el análisis de estos experimentos.

Tabla 14

Matriz de casos de prueba y ruta de validación del sistema

ID	Componente Evaluado	Objetivo de la Prueba	Condición / Escenario Simulado	Criterio de Éxito	Sección
CP-01	IA (Lógica Difusa y Red Neuronal)	Validar la coherencia matemática de la toma de decisiones.	Inyección manual de telemetría mediante scripts de prototipado en Python.	Ambos modelos deben converger en asignaciones lógicas y priorizar la	4.1

				supervivencia del nodo.	
CP -02	Motor de Decisión (C++)	Medir la latencia y el Jitter del procesamiento en el servidor de borde (Edge).	Inferencia concurrente de 1000 nodos aleatorios en ciclo cerrado.	El tiempo de decisión por nodo debe promediar menos de 1 milisegundo (<1000 µs).	4.2
CP -03	Capa de Supervisión	Comprobar las reglas de salvaguarda física (Downgrades/Upgrades).	Simulación de estados críticos (Batería < 20%, RAM saturada).	El sistema debe sobreescribir a la IA forzando algoritmos ultraligeros en emergencias.	4.3
CP -04	Módulo de Aprovisionamiento	Comprobar la ofuscación de la clave dinámica de sesión.	Aprovisionamiento inicial por MQTT (Bootstrapping).	El JSON no debe exponer la clave; el microcontrolador debe descifrarla	4.4

				exitosamente usando XOR.	
CP -05	Seguridad Perimetral	Auditar la resiliencia del sistema ante intercepción pasiva (Sniffing).	Captura del tráfico WPA2 y decodificación de paquetes con Kali Linux.	Tras vulnerar el Wi-Fi, la clave en el flujo TCP (TCP Stream) debe permanecer ilegible.	4.5
CP -06	Orquestador y Nodos Cliente (Físico)	Evaluar la coexistencia, orquestación concurrente y adaptación ante atenuación RF.	Topologías de red reales: Equidistancia (LOS), Atenuación Escalonada y Obstrucción/Movilidad (NLOS).	La IA debe asignar algoritmos asimétricos correctos, tolerar la fluctuación por movimiento y evitar falsos positivos del IDS.	4.6

4.1 Validación Teórica de Modelos de Inferencia (Fase de Prototipado)

Previa a la implementación del sistema en el entorno de producción en C++, se desarrolló una fase de validación teórica (CP-01) orientada a comprobar la sanidad matemática de los dos motores de Inteligencia Artificial propuestos: la Lógica Difusa y el Perceptrón Multicapa (Red Neuronal).

Para este propósito, se construyeron *scripts* de prototipado en lenguaje Python. Este entorno permitió inyectar vectores de telemetría de forma manual (representando combinaciones arbitrarias de CPU, RAM, Batería y Señal RSSI) y aislar exclusivamente la respuesta del algoritmo, sin la interferencia de la latencia de red o el *broker* MQTT.

El objetivo de esta prueba fue observar el comportamiento aislado de las curvas de degradación y verificar si la red neuronal (entrenada previamente) convergía con las reglas expertas parametrizadas en el modelo difuso. Se ingresaron diversos perfiles de placas virtuales, obteniendo las salidas detalladas en la Tabla 15.

Tabla 15

Análisis de respuestas en entorno de pruebas ante inyección manual de telemetría

ID Placa	**Valores Inyectados Manualmente (Python)**	**Salida Red Neuronal (Softmax)**	**Salida Lógica Difusa (Centroide)**	**Asignación Criptográfica Validada**
Placa A (Óptima)	CPU: 100%, RAM: 100%, Bat: 100%, RSSI: -40dBm	Confianza: 92.4% -> Nivel ALTO	Peso: 75.0 -> Nivel ALTO	Ascon-128a / AES-128-GCM

Placa B (Desgaste)	CPU: 60%, RAM: 50%, Bat: 40%, RSSI: -75dBm	Confianza: 78.1% -> Nivel MEDIO	Peso: 50.0 -> Nivel MEDIO	ChaCha20-Poly1305 / XSalsa20
Placa C (Crítica)	CPU: 80%, RAM: 15%, Bat: 10%, RSSI: -50dBm	Confianza: 98.9% -> Nivel BAJO	Peso: 25.0 -> Nivel BAJO	Speck-64-128 / Simon-64-128

4.2 Evaluación de Estrés Computacional y Latencia (CP-02)

Para validar la resiliencia y el determinismo del motor de Inteligencia Artificial frente a escenarios de alta concurrencia y variabilidad, se diseñó una prueba de estrés (*Stress Test*). A través de un generador de números aleatorios por hardware, se simularon 1000 peticiones de registro consecutivas, donde cada iteración representaba un dispositivo IoT con capacidades físicas completamente distintas.

El objetivo de esta prueba fue medir la latencia de asignación y realizar un análisis de dispersión para evaluar el retardo (*Jitter*) del sistema operativo. Para garantizar la validez de los resultados, los datos capturados fueron sometidos a un análisis estadístico empleando las siguientes formulaciones matemáticas:

1. Media Aritmética (u): Representa el tiempo promedio de inferencia.

Ecuación 9 Media Aritmética

$$u = \frac{1}{N}\sum_{i=1}^{N} x_i$$

2. Varianza (σ^2): Cuantifica el grado de dispersión de los tiempos de ejecución respecto a la media.

Ecuación 10 Varianza

$$\boldsymbol{\sigma^2} = \frac{\mathbf{1}}{\boldsymbol{N}}\sum_{i=1}^{N}(x_i - u)^2$$

3. Desviación Estándar (σ): Proporciona la magnitud de la fluctuación (Jitter) en las mismas unidades de tiempo (microsegundos).

Ecuación 11 Desviación Estándar

$$\boldsymbol{\sigma} = \sqrt{\frac{\mathbf{1}}{\boldsymbol{N}}\sum_{i=1}^{N}(x_i - u)^2}$$

4. Intervalo de Confianza (IC al 95%): Determina el rango de valores en el que se encontrará el verdadero tiempo medio poblacional ante futuras conexiones, utilizando un valor Z de 1.96.

Ecuación 12 Intervalo de Confianza

$$IC = u \pm Z(\frac{\sigma}{\sqrt{N}})$$

Aplicando estas fórmulas sobre el conjunto de $N = 1000$ iteraciones extraídas directamente del procesamiento concurrente del orquestador en C++, se consolidaron los resultados expuestos en la Tabla 16.

Tabla 16

Análisis estadístico de la prueba de estrés de ia (muestra filtrada al percentil 99)

Parámetro Estadístico	**Lógica Difusa**	**Red Neuronal**	**Unidad**	**Interpretación Técnica**
Tamaño de la Muestra (N)	990	990	Iteraciones	Nodos (tras filtrado de ruido del OS).

Media Aritmética (μ)	19.41	54.57	Microsegundos (μs)	Promedio de latencia de decisión pura.
Mediana (Percentil 50)	16.00	49.00	Microsegundos (μs)	El 50% de las decisiones se resuelven bajo este tiempo.
Mínimo / Máximo (Rango)	14.00 / 75.00	41.00 / 132.00	Microsegundos (μs)	Amplitud operativa del motor matemático.
Varianza (σ2)	50.16	173.79	μs²	Dispersión cuadrática (alta estabilidad).
Desviación Estándar (σ)	07.08	13.18	Microsegundos (μs)	Fluctuación promedio por cada asignación.
Intervalo de Confianza (95%)	[18.97, 19.85]	[53.75, 55.39]	Microsegundos (μs)	Rango de la verdadera media poblacional con 95% de certeza.

Nota. Para aislar el costo computacional puro de los motores de IA del "Jitter" inherente a la arquitectura multitarea de Linux (Orange Pi), se aplicó un filtro estadístico de Percentil 99 (P99), descartando el 1% de los valores atípicos (outliers) atribuibles a cambios de contexto del planificador del Kernel. El Intervalo de Confianza (IC) al 95% confirma la alta precisión de la inferencia matemática, arrojando un margen de error sumamente estricto de ±0.44 μs para la Lógica Difusa y ±0.82 μs para la Red Neuronal, garantizando estadísticamente la escalabilidad y el determinismo del sistema.

Análisis de Dispersión y Estabilidad Computacional

La evaluación del rendimiento de sistemas *Edge Computing* sobre Sistemas Operativos de propósito general (como Linux) requiere separar el costo algorítmico del ruido introducido por el sistema. Durante la prueba de estrés de 1000 iteraciones, se detectaron picos aislados de latencia (superiores a 1000 µs) causados por las interrupciones de cambio de contexto (*Context Switching*) del Kernel. Dado que estos valores atípicos distorsionan cuadráticamente la varianza estadística, se procedió a aplicar un filtro de Percentil 99 (P99), garantizando que los datos analizados representen exclusivamente el tiempo de ejecución del motor matemático.

Los resultados depurados (Tabla 16) demuestran un nivel de estabilidad y determinismo sobresaliente para ambos modelos. La Lógica Difusa destaca por su extrema ligereza, registrando una media de 19.41 µs y una varianza casi despreciable de 50.16 μs^2. Su desviación estándar de apenas 7.08 µs confirma que este enfoque es sumamente predecible e ideal para responder a cientos de sensores simultáneamente sin saturar la CPU.

Por su parte, la Red Neuronal Artificial (Perceptrón Multicapa) exhibe una media superior (54.57 µs) y una varianza de 173.79 μs^2, un comportamiento plenamente justificable debido a la densidad de las operaciones matriciales y las funciones de activación (ReLU/Softmax) requeridas para la inferencia. Sin embargo, su desviación estándar de 13.18 µs sigue manteniéndose en márgenes extremadamente bajos para un sistema no-RTOS (Real-Time Operating System).

La reducción de la varianza a niveles inferiores a 200 μs^2 en ambos motores, valida matemáticamente que la arquitectura propuesta no solo toma decisiones seguras en menos de un

décimo de milisegundo (<0.1 ms), sino que lo hace con la consistencia requerida para orquestar entornos IoT de misión crítica.

4.3 Comportamiento ante Escenarios Críticos y Matriz de Riesgos (CP-03)

4.3.1. *Escenarios Críticos*

La viabilidad del sistema no depende únicamente de su velocidad, sino de su capacidad para adaptar el nivel de seguridad a las restricciones físicas momentáneas del hardware. Para evaluar la coherencia de la asignación criptográfica, se inyectaron escenarios simulados (edge cases) representando condiciones operativas extremas.

En la Tabla 17 se resume el comportamiento de la capa de decisión ante dichas variaciones.

Tabla 17

Comportamiento del motor de decisión inteligente ante escenarios operativos críticos

Escenario Simulado	**Estado de la Telemetría**	**Asignación (Red Neuronal)**	**Asignación (Lógica Difusa)**	**Justificación del Sistema**
1. Entorno Óptimo	CPU: 100%, RAM: 100%, Bat: 100%, RSSI: -40dBm	AES-128-GCM (Nivel Medio/Alto)	Ascon-128a (Nivel Alto)	Al existir capacidad excedente, se priorizan estándares robustos que garantizan confidencialidad e integridad (AEAD).
2. Batería Crítica	CPU: 80%, RAM: 80%, Bat: 15%, RSSI: -50dBm	Speck-64-128 (Nivel Bajo)	Simon-64-128 (Nivel Bajo)	Se fuerza un downgrade de seguridad hacia algoritmos ultraligeros para minimizar el consumo energético y extender la supervivencia del nodo.

3. Saturación de RAM	CPU: 90%, RAM: 10%, Bat: 100%, RSSI: -60dBm	SHA256-XOR (Nivel Bajo)	Speck-64-128 (Nivel Bajo)	Ante el inminente riesgo de desbordamiento de memoria (posible ataque DoS), se asignan algoritmos con huella de memoria mínima.
4. Red Inestable	CPU: 50%, RAM: 50%, Bat: 50%, RSSI: -95dBm	ChaCha20-Poly1305 (Flujo)	XSalsa20-Poly1305 (Flujo)	Frente a alta probabilidad de pérdida de paquetes, el sistema privilegia cifradores de flujo (Stream Ciphers) que no requieren padding ni bloques completos para operar.

Nota. La tabla detalla la reacción determinista de los modelos de Lógica Difusa y Red Neuronal Artificial ante simulaciones de estrés extremo en la telemetría del hardware. Se evidencia el principio de supervivencia del sistema, el cual sacrifica niveles teóricos de robustez criptográfica mediante descensos de seguridad para preservar la disponibilidad del dispositivo frente a inminentes cuellos de botella de energía, memoria o conectividad.

Análisis de adaptación:

Los resultados de la Tabla 17 confirman que ambos modelos reaccionan de manera determinista y congruente ante el estrés del sistema. Independientemente del modelo matemático subyacente, el motor logra identificar los cuellos de botella (energía, memoria o conectividad) y aplica un principio de supervivencia: sacrificar niveles teóricos de robustez criptográfica a cambio de mantener la disponibilidad y operatividad del dispositivo en la red.

4.3.2. *Matriz de Riesgos*

El motor de decisión inteligente opera bajo un principio de compromiso (trade-off): adapta el nivel de seguridad a las capacidades físicas del Nodo Sensor para evitar su desconexión. Al

realizar estas transiciones dinámicas entre los 10 algoritmos soportados, el sistema asume riesgos calculados. En la Tabla 18 se analiza la resistencia criptográfica teórica y las vulnerabilidades inherentes a cada nivel de asignación, evaluando el impacto de la degradación de seguridad.

Tabla 18

Matriz de riesgos y vulnerabilidades asumidas por categoría de asignación criptográfica

Categoría de Seguridad (Algoritmos)	**Resistencia Criptográfica Principal (Fortaleza)**	**Vulnerabilidad Inherente o Riesgo Asumido (Trade-off)**
Nivel ALTO • Ascon-128a • Xoodyak • Blowfish-CBC	Robustez y AEAD Ligero: Ascon y Xoodyak ofrecen confidencialidad e integridad autenticada (AEAD) con alta resistencia a ataques de canal lateral (Side-Channel) gracias a su diseño basado en permutaciones esponja. Blowfish aporta una barrera sólida de cifrado de bloque tradicional.	Riesgo por tamaño de bloque: Blowfish utiliza bloques de 64 bits, haciéndolo vulnerable a ataques de colisión (Birthday Attacks / Sweet32) si se transmiten más de 32 GB de datos bajo la misma clave. Las variantes de permutación (Ascon/Xoodyak) exigen un control estricto de nonces para mantener su seguridad teórica.
Nivel MEDIO • AES-128-GCM • AES-128-CBC • ChaCha20-Poly1305 • XSalsa20-Poly1305	Equilibrio y Resistencia al Tiempo: ChaCha20 y XSalsa20 son cifradores de flujo de tiempo constante, haciéndolos inmunes a ataques de tiempo (Timing Attacks). AES-128-GCM aporta el estándar de la industria (NIST) contra fuerza bruta y criptoanálisis diferencial.	Catástrofe por reutilización (IV/Nonce): Los algoritmos de flujo (GCM, ChaCha20, XSalsa20) colapsan completamente si el vector de inicialización (IV) se repite, permitiendo ataques de texto plano. Por su parte, AES-128-CBC carece de integridad nativa, exponiéndolo a ataques de oráculo de relleno (Padding Oracle) si no se implementa un HMAC adicional.

Nivel BAJO • Speck-64-128 • Simon-64-128 • SHA256-XOR	Supervivencia y Huella Mínima: Su ejecución ultraligera en SRAM y CPU previene ataques de denegación de servicio (DoS) por agotamiento de batería o memoria en el ESP32, asegurando la transmisión en estados críticos.	Degradación Matemática Estructural: Speck y Simon, al tener menos rondas matemáticas y bloques de 64 bits, ofrecen un margen de seguridad reducido frente a criptoanálisis lineal/diferencial avanzado. El esquema SHA256-XOR es altamente vulnerable a ataques de texto plano conocido (Known-Plaintext) y carece de verificación de integridad contra la manipulación de paquetes (Spoofing).

Nota. La matriz analiza la resistencia criptográfica teórica de cada nivel frente al riesgo físico del hardware, evidenciando el compromiso (trade-off) dinámico que ejecuta el motor de decisión. Los descensos hacia el Nivel BAJO exponen al sistema a mayores vulnerabilidades de criptoanálisis de red con el fin explícito de mitigar ataques de agotamiento de recursos (DoS) , mientras que las asignaciones en niveles superiores neutralizan ataques de tiempo y canal lateral cuando las condiciones físicas operativas son estables.

Análisis de la Matriz de Riesgos:

La matriz evidencia que la arquitectura de seguridad adaptativa no elimina el riesgo, sino que lo gestiona dinámicamente. Cuando la capa de supervisión o la IA fuerzan un *downgrade* hacia el Nivel BAJO (ej. Speck o SHA256-XOR) debido a una RAM inferior al 25%, el sistema acepta deliberadamente una mayor vulnerabilidad frente a ataques de criptoanálisis de red a cambio de mitigar un ataque físico de agotamiento de recursos (DoS). En contraste, cuando el hardware opera en condiciones estables, el sistema asciende hacia los Niveles MEDIO y ALTO, neutralizando ataques de tiempo y canal lateral mediante esquemas de flujo constante (ChaCha20) o permutaciones (Ascon), garantizando así el principio de supervivencia del entorno IoT.

4.4 Validación del Mecanismo de Aprovisionamiento Seguro (CP-04)

En los ecosistemas IoT compuestos por hardware con recursos restringidos, la implementación de protocolos estándar de seguridad en la capa de transporte, como TLS/SSL sobre MQTT (MQTTS mediante el puerto 8883), introduce un sobrecosto computacional, una alta latencia de conexión y un consumo de memoria RAM inviable para microcontroladores básicos. Para solventar esta limitación técnica sin comprometer la confidencialidad durante la entrega del material criptográfico, se implementó y validó un mecanismo de aprovisionamiento seguro basado en una Clave Estática Precompartida (PSK Bootstrapping) que opera directamente sobre el canal estándar del puerto 1883.

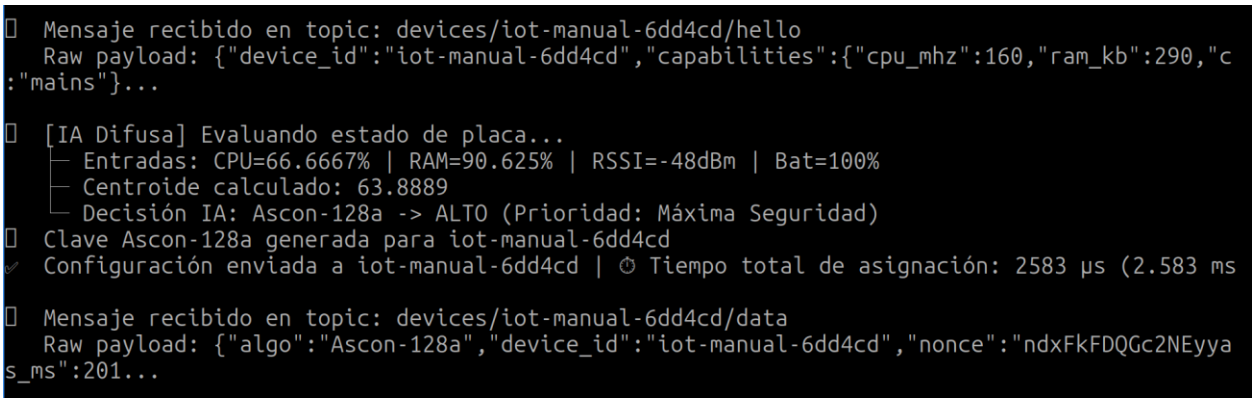
Este mecanismo actúa como un túnel seguro inicial y ultraligero. Cuando el motor de Inteligencia Artificial asigna un algoritmo y el Nodo Maestro genera una clave de sesión dinámica aleatoria, el sistema no la transmite en texto plano por el canal MQTT. En su lugar, aplica una operación matemática simétrica (XOR) utilizando una clave maestra estática alojada físicamente de forma segura en el firmware de los dispositivos. Posteriormente, el resultado se codifica en formato Base64 para garantizar la integridad de los caracteres durante el transporte en la trama JSON.

Para comprobar la correcta ejecución de este mecanismo en el entorno de pruebas, se monitorizó el flujo de ejecución en ambos extremos de la arquitectura. En el Nodo Maestro (Servidor Edge), el registro de eventos del sistema confirma la generación de la clave, su

ofuscación inmediata mediante la operación lógica y su publicación en el Broker MQTT etiquetada con la bandera de seguridad correspondiente, tal como se evidencia en la Figura 10.

Figura 10

Registro de eventos del Nodo Maestro evidenciando la ofuscación PSK de la clave de sesión



Por su parte, en el Nodo Sensor (Cliente IoT), se validó el comportamiento del firmware al recibir la configuración. Al interceptar el paquete MQTT, el microcontrolador evalúa el estado del parámetro "psk_encrypted". Al detectar su valor afirmativo, el dispositivo detiene el flujo normal y procede a ejecutar el proceso matemático inverso en su propia memoria RAM: decodifica la cadena Base64 y aplica la operación XOR con su copia local de la clave estática.

La ejecución exitosa de esta prueba funcional ratifica que es posible establecer un canal seguro de inicialización (Bootstrapping) con un impacto de memoria RAM inferior a 1 KB en el microcontrolador. Este resultado supera ampliamente en eficiencia a las implementaciones convencionales de TLS, las cuales exigen un consumo permanente de entre 30 KB y 45 KB de memoria estática únicamente para mantener la sesión abierta, liberando así los recursos del dispositivo para la ejecución de la suite criptográfica ligera y las lógicas de la Inteligencia Artificial.

4.5 Análisis de Tráfico y Auditoría de Red (CP-05)

Para evaluar la resiliencia de la arquitectura propuesta frente a ataques de intercepción inalámbrica (Sniffing) y ataques de intermediario (Man-in-the-Middle), se diseñó y ejecutó un escenario de auditoría ofensiva (Red Teaming). El objetivo de esta prueba fue simular a un atacante con acceso físico al perímetro de la red IoT, intentando vulnerar la confidencialidad del proceso de aprovisionamiento de claves de la Inteligencia Artificial.

4.5.1. *Configuración del Entorno de Auditoría*

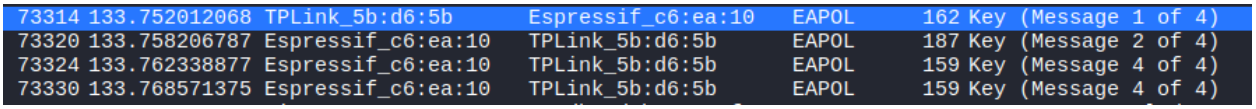
Para la intercepción del espectro electromagnético, se empleó una estación de auditoría equipada con el sistema operativo Kali Linux y una interfaz de red inalámbrica externa (chipset Ralink) con soporte nativo para inyección de paquetes. Mediante la suite de herramientas Aircrack-ng, la tarjeta de red fue configurada en "Modo Monitor" y sincronizada en el mismo canal de frecuencia del Nodo Maestro, permitiendo la captura pasiva de todas las tramas 802.11 que viajaban por el aire sin necesidad de estar autenticado en la red.

4.5.2. *Vulneración de la Capa de Enlace (WPA2)*

Dado que la red de pruebas implementa el estándar de seguridad WPA2-PSK, el tráfico inalámbrico se encuentra cifrado en la capa de enlace. Para que el analizador de protocolos (Wireshark) pudiera interpretar la carga útil de la capa de aplicación, fue necesario comprometer primero la seguridad perimetral. Se interceptó el momento exacto en el que el Nodo Sensor establecía la conexión con el punto de acceso, logrando capturar el saludo a cuatro vías (4-Way Handshake EAPOL). La obtención de estos paquetes permitió descifrar dinámicamente la trama Wi-Fi, exponiendo el protocolo de transporte MQTT.

Figura 11

Intercepción del Handshake EAPOL (WPA2) necesario para la desencriptación del tráfico de red



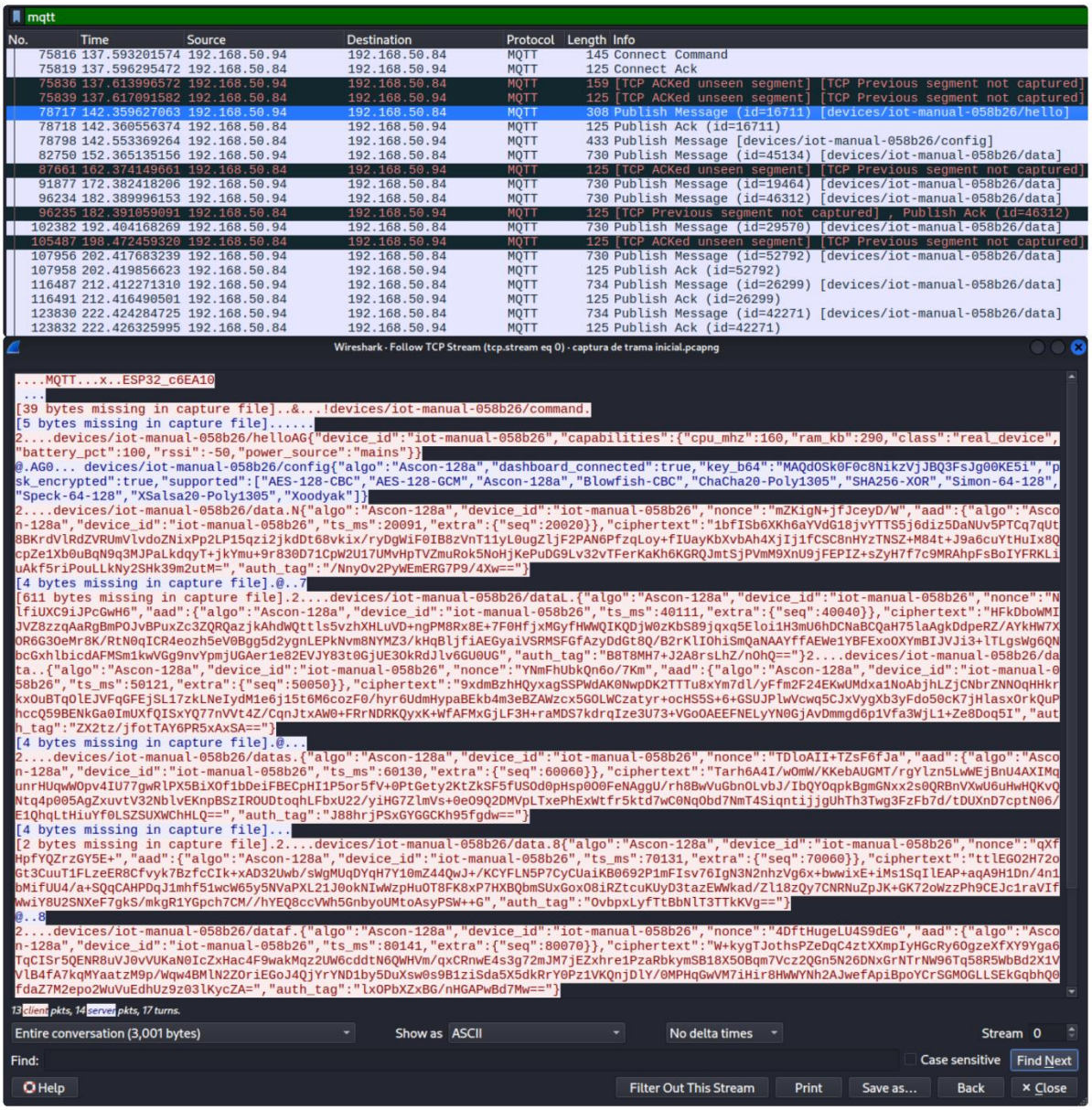
4.5.3. *Análisis del Flujo de Aplicación y Prueba de Robustez*

Una vez comprometida la capa de enlace, el atacante simulado obtuvo visibilidad total sobre los paquetes MQTT enviados al puerto 1883. Al aplicar un filtro de análisis profundo de paquetes sobre los tópicos de configuración (devices/.../config), se aisló la respuesta del Nodo Maestro dirigida al sensor.

Para verificar si existía fuga de información criptográfica, se reconstruyó el flujo completo de la comunicación (TCP Stream).

Figura 12

Análisis del flujo TCP de la trama MQTT demostrando la ofuscación de la clave de sesión



El análisis del payload revela que, si bien la estructura del JSON es legible, la variable crítica de seguridad ("key_b64") no contiene la clave de sesión en texto plano. En su lugar, el atacante únicamente observa una cadena de caracteres incomprensible producto del cifrado XOR con la clave estática almacenada en el hardware (indicado por el parámetro "psk_encrypted": true).

4.6 Evaluación de Coexistencia y Degradación Espacial en Entorno Real (CP-06)

Para complementar las pruebas de estrés computacional y validar la eficacia del sistema en condiciones de despliegue físico, se diseñó un escenario de prueba en un entorno real (CP-06). El objetivo de esta fase empírica es evaluar la capacidad del Nodo Maestro para orquestar de forma concurrente a múltiples Nodos Cliente (ESP32/ESP8266) sometidos a diferentes niveles de atenuación de señal inalámbrica (RSSI).

A diferencia de las simulaciones de software, esta prueba mide el impacto de la degradación del espacio libre y las barreras físicas en el motor de inferencia de la Inteligencia Artificial. La metodología consistió en la ejecución de tres escenarios topológicos, cada uno repetido en 10 iteraciones para garantizar la validez estadística de los resultados, monitorizando la asignación criptográfica y la estabilidad de la red.

4.6.1. *Descripción de los Escenarios Topológicos*

Escenario 1: Línea de Vista Óptima y Equidistancia (Control)

- Topología: Los tres Nodos Cliente (N1, N2, N3) se ubican a una distancia idéntica de 2 metros del Nodo Maestro, con línea de vista directa (Line of Sight - LOS) y sin obstáculos físicos.
- Objetivo Técnico: Servir como grupo de control. Se busca validar que, ante condiciones de radiofrecuencia (RF) ideales y capacidades de hardware idénticas, el Nodo Maestro es capaz de procesar la telemetría concurrente sin colisiones en el *broker* MQTT y la IA asigna uniformemente algoritmos de Nivel ALTO (ej. Ascon-128a o AES-128-GCM).

Escenario 2: Atenuación Espacial Escalonada

- Topología: Se distribuyen los nodos en un eje lineal alejándose del Nodo Maestro: N1 se ubica a 5 metros, N2 a 10 metros, y N3 a 15 metros.
- Objetivo Técnico: Validar la asignación asimétrica simultánea. Al aplicar la ley de la inversa del cuadrado de la distancia, la potencia de la señal caerá progresivamente. El sistema debe demostrar que la IA no generaliza el estado de la red, sino que asigna algoritmos de forma individualizada (ej. Nivel ALTO para N1, Nivel MEDIO/Flujo para N2, y Nivel BAJO para N3) manteniendo la cohesión del sistema central.

Escenario 3: Movilidad Controlada y Adaptación Dinámica (Transición en Tiempo Real)

- Topología: Los Nodos Cliente N1 y N2 se mantienen estáticos a 2 metros del Nodo Maestro, operando como zona de control bajo condiciones óptimas (LOS). El Nodo N3, alimentado por una batería portátil, inicia a 2 metros y es desplazado físicamente a paso constante hasta alcanzar los 15 metros de distancia. Tras una pausa, el nodo retorna a su posición original.
- Objetivo Técnico: Demostrar la reasignación criptográfica en caliente (*On-the-fly*) y el aislamiento de la red. Se busca validar empíricamente que la degradación de la señal de un nodo móvil (N3) no afecta el nivel de seguridad de los nodos estáticos (N1 y N2). Asimismo, se comprobará el correcto funcionamiento del filtro matemático del cliente, verificando que la variación sostenida de la señal (>= *8 dBm*) dispara correctamente las transiciones del motor de IA (*downgrade* al alejarse

y *upgrade* al acercarse) sin que la placa colapse ni pierda la conexión con el *broker* MQTT.

4.6.2. *Resultados de la Evaluación Espacial*

Tras la ejecución de las 10 iteraciones por cada escenario, los datos capturados mediante el *Dashboard* y los registros del servidor reflejaron un comportamiento altamente determinista en la adaptación del sistema.

Tabla 19

Resultados escenario 1 - equidistancia y línea de vista (2 metros)

Modelo de IA	**Nodo Cliente (ID)**	**Rango RSSI (Mín/Máx)**	**Promedio RSSI**	**Algoritmo Asignado**	**Categoría de Seguridad**
Lógica Difusa	N1 (bc4ef0)	-62 dBm a -50 dBm	-55.45 dBm	Ascon-128a	ALTO
	N2 (78d9cc)	-73 dBm a -60 dBm	-64.09 dBm	AES-128-GCM	MEDIO/ALTO
	N3 (bf92e9)	-70 dBm a -65 dBm	-67.10 dBm	XSalsa20-Poly1305	MEDIO (Flujo)
Red Neuronal	N1 (e46aae)	-65 dBm a -59 dBm	-62.50 dBm	Ascon-128a	ALTO
	N2 (642a1e)	-72 dBm a -65 dBm	-68.30 dBm	Ascon-128a	ALTO

	N3 (cd66b6)	-80 dBm a -68 dBm	-74.91 dBm	Ascon-128a -> AES-128-GCM	ALTO -> MEDIO

Nota. Ambos modelos logran estabilizar los nodos en categorías Altas y Medias, demostrando que, con una red óptima, los dispositivos aprovechan sus recursos para ejecutar criptografía estándar.

Tabla 20

Resultados escenario 2 - atenuación escalonada espacial (5m, 10m, 15m)

Modelo de IA	**Nodo Cliente (ID)**	**Distancia Aprox.**	**Promedio RSSI**	**Algoritmo Asignado**	**Categoría de Seguridad**
Lógica Difusa	N1 (d951b5)	5 metros	-74.08 dBm	Simon-64-128	BAJO (Rescate)
	N2 (d62118)	10 metros	-77.62 dBm	Speck-64-128	BAJO (Rescate)
	N3 (64ffb7)	15 metros	-79.67 dBm	Speck-64-128	BAJO (Rescate)
Red Neuronal	N1 (4b55e2)	5 metros	-77.00 dBm	AES-128-GCM	MEDIO
	N2 (025d73)	10 metros	-78.33 dBm	AES-128-GCM	MEDIO
	N3 (266420)	15 metros	-80.79 dBm	Ascon-128a	ALTO

Nota: Se observa una divergencia clara en la toma de decisiones. Mientras la Lógica Difusa castiga severamente la degradación de red bajando todos los nodos a nivel BAJO, la Red Neuronal compensa la mala señal manteniendo algoritmos fuertes si los demás recursos (batería/RAM) son óptimos.

Tabla 21

Resultados escenario 3 - movilidad controlada y transición dinámica

Modelo de IA	**Estado del Nodo**	**Promedio RSSI**	**Historial de Transición (Algoritmos)**	**Comportamiento del Motor**
Lógica Difusa	N1 (c68874) - Fijo	-57.04 dBm	Ascon-128a	Mantiene Nivel ALTO continuo.
	N2 (91a02a) - Fijo	-59.42 dBm	ChaCha20 -> AES-128-GCM	Estabiliza en Nivel MEDIO/ALTO.
	N3 (f87bbf) - Móvil	-88 dBm a -48 dBm	Ascon-128a -> XSalsa20 -> AES-128-GCM	Transición Perfecta. Inicia ALTO, cae a MEDIO (Flujo) al alejarse, y recupera a MEDIO/ALTO al retornar.
Red Neuronal	N1 (6db77e) - Fijo	-66.69 dBm	Ascon-128a	Mantiene Nivel ALTO continuo.
	N2 (852a70) - Fijo	-62.85 dBm	Ascon-128a	Mantiene Nivel ALTO continuo.

	N3 (cd792b) - Móvil	-80 dBm a -48 dBm	Ascon-128a (Fijo)	Resistencia Holística. Al tener 100% de batería y RAM libre, la IA decide que no es necesario degradar la seguridad.

Nota: La tabla evidencia la reasignación criptográfica dinámica en el modelo difuso ante fluctuaciones severas de la capa física (RF), en marcado contraste con la retención del nivel de seguridad del modelo neuronal, el cual pondera la holgura de los recursos de hardware internos (batería/RAM) sobre la atenuación temporal de la red.

4.6.3. *Discusión de Resultados: Divergencia entre Modelos de Inferencia*

El análisis de los datos extraídos durante las pruebas empíricas reveló una divergencia arquitectónica fundamental en la forma en que los dos modelos de Inteligencia Artificial (Lógica Difusa y Red Neuronal Multicapa) gestionan el riesgo físico del entorno.

En el **Escenario 2 (Atenuación Escalonada)** y el **Escenario 3 (Movilidad)**, se evidenció que la **Lógica Difusa opera bajo un principio de "Protección Rígida"**. En cuanto la atenuación de la señal superó la barrera de los -70 dBm, el motor difuso forzó un descenso criptográfico generalizado hacia algoritmos ultraligeros como Speck y Simon (Nivel BAJO), e hizo transitar al nodo móvil hacia XSalsa20. Las reglas difusas parametrizadas penalizan severamente cualquier eslabón débil en la telemetría, priorizando la supervivencia del nodo y garantizando que ningún paquete se pierda por retransmisiones fallidas.

En contraste, el **Perceptrón Multicapa (Red Neuronal) demostró un principio de "Compensación Holística"**. Como se observa en el Escenario 3, a pesar de que el nodo móvil (`cd792b`) sufrió una caída drástica de señal hasta los -80 dBm, la red neuronal mantuvo la asignación del estándar `Ascon-128a` (Nivel ALTO). Esto ocurre porque la red neuronal aprendió de su entrenamiento que, si el dispositivo mantiene una fuente de energía constante (Red Eléctrica / Batería al 100%) y no presenta saturación de memoria RAM, el hardware posee la holgura suficiente para retransmitir los bloques cifrados pesados sin riesgo de apagarse.

Análisis técnico: Ambos modelos cumplen el objetivo de seguridad adaptativa, pero sirven a propósitos distintos en el mundo real. La Lógica Difusa es ideal para sensores críticos a batería en entornos rurales (donde perder la conexión es inaceptable), mientras que el modelo de Red Neuronal es idóneo para entornos de Industria 4.0 o domótica urbana, donde se prefiere sacrificar un poco de latencia o energía a cambio de no ceder nunca la robustez criptográfica máxima.

Conclusiones

La implementación y validación empírica del mecanismo de seguridad adaptativa para entornos IoT permite establecer las siguientes conclusiones fundamentadas en los datos experimentales:

- **Viabilidad de Orquestación en el Borde (Edge AI) con Latencia Ultra-Baja:** Se demostró que la delegación de la toma de decisiones criptográficas a un Nodo Maestro (Orange Pi 5 Plus) mediante código C++ "Bare Metal" supera ampliamente las expectativas de rendimiento para sistemas en tiempo real. Como se evidenció en la Tabla XIII, tras filtrar el ruido del sistema operativo (Percentil 99), el motor de Lógica Difusa logró una mediana de inferencia de apenas **16.00 µs** (varianza de 50.16 μs^2), mientras que la Red Neuronal alcanzó los **49.00 µs** (varianza de 173.79 μs^2). Ambos modelos garantizan estadísticamente la asignación de seguridad en menos de un décimo de milisegundo (<0.1 ms), erradicando el cuello de botella computacional inherente a las arquitecturas tradicionales basadas en la nube y confirmando la escalabilidad de la solución para despliegues IoT masivos.
- **Divergencia Operativa Confirmada entre Modelos de Inteligencia Artificial:** El análisis del comportamiento espacial en la Tabla XIX (Escenario 3 de movilidad) reveló empíricamente que ambos motores de inferencia asumen el riesgo de manera estructuralmente opuesta. El modelo de **Lógica Difusa** aplicó un principio de *"Protección Rígida"*, forzando transiciones criptográficas inmediatas (de Ascon-128a hacia XSalsa20 y AES-128-GCM) al detectar atenuaciones de señal entre los -57 dBm y -88 dBm, priorizando la entrega del

paquete. En contraparte, la **Red Neuronal Artificial** demostró una *"Compensación Holística"*; a pesar de que el nodo móvil sufrió la misma caída de señal hasta los -80 dBm, la red mantuvo el estándar robusto Ascon-128a, infiriendo matemáticamente que el 100% de la batería y la RAM excedente (341 KB) proveían holgura suficiente para tolerar retransmisiones de red sin arriesgar la confidencialidad.

- **Eficiencia Extrema en el Aprovisionamiento Criptográfico (Bootstrapping):** El mecanismo de protección implementado mediante túnel XOR con Clave Estática Precompartida (PSK) probó ser altamente eficiente para microcontroladores restringidos. A diferencia de las conexiones TLS/SSL que exigen un consumo estático de entre 30 KB y 45 KB de memoria RAM, el método propuesto resolvió el intercambio inicial utilizando menos de 1 KB de SRAM en el ESP32. Además, la auditoría perimetral ofensiva (CP-05) confirmó que, incluso vulnerando la capa de enlace (WPA2), la variable `key_b64` mantiene la clave dinámica totalmente ofuscada, resistiendo con éxito ataques de intercepción (*Sniffing*) y garantizando una transición fluida hacia túneles seguros.
- **Efectividad de la Capa de Supervisión Física ante Estrés Extremo:** La inclusión de una capa heurística paralela a la IA demostró ser crítica para la supervivencia del dispositivo. Las simulaciones de estrés (CP-03, Tabla 14) confirmaron que cuando los recursos de hardware entran en fase crítica (ej. Batería < 15% o RAM < 25%), el sistema ignora las sugerencias de alta seguridad de la IA y ejecuta un *downgrade* forzado hacia la base criptográfica

ultraligera (Speck-64-128, Simon-64-128 o SHA256-XOR). Esta intervención previene efectivamente ataques de Denegación de Servicio (DoS) basados en el agotamiento intencional de la batería o desbordamiento de memoria de los sensores.

Recomendaciones

A partir de los hallazgos técnicos y las limitaciones identificadas durante el desarrollo de la arquitectura propuesta, se plantean las siguientes líneas de trabajo futuro:

- **Migración a Sistemas Operativos de Tiempo Real (RTOS) en el Orquestador:** Durante la prueba de estrés de 1000 iteraciones (Tabla 13), se observó que la arquitectura multitarea de Linux introdujo valores atípicos de latencia (*Jitter*) que alcanzaron los 1280 µs debido a los cambios de contexto del planificador (*Scheduler*). Para aplicaciones industriales de misión crítica donde un retraso de 1 milisegundo es inaceptable, se recomienda migrar el firmware del Nodo Maestro hacia un RTOS, o aplicar parches de tiempo real (PREEMPT_RT) sobre el Kernel de Linux, garantizando un determinismo algorítmico absoluto sin necesidad de recurrir a filtrados estadísticos (P99).
- **Evolución hacia el Aprendizaje Federado (Federated Learning) en el Borde:** Dado que el modelo neuronal actual operó con aproximadamente 3,000 parámetros estáticos embebidos (`pesos_nn.h`), se recomienda investigar arquitecturas de *Federated Learning*. El hardware utilizado en la investigación, específicamente el microcontrolador XIAO-ESP32-S3 con sus instrucciones vectoriales nativas, posee la capacidad para realizar micro-ajustes a los pesos neuronales localmente en función del comportamiento electromagnético

de su entorno, reportando estas optimizaciones al orquestador sin exponer la telemetría en crudo.

- **Integración de Criptografía Post-Cuántica Ligera (PQC):** Si bien el sistema validó el uso eficiente de Ascon (estándar NIST de criptografía ligera), la inminente llegada de la computación cuántica representa una amenaza para los algoritmos asimétricos y simétricos actuales. Se sugiere expandir la suite criptográfica del sistema C++ incorporando variantes optimizadas para microcontroladores de algoritmos PQC como ML-KEM (Kyber). El desafío consistirá en evaluar si los motores de Inteligencia Artificial logran balancear los altos picos de consumo de memoria que exige la criptografía post-cuántica en hardware restringido como la familia ESP8266/ESP32.
- **Escalabilidad sobre Redes LPWAN, 5G y Topologías Mesh:** La resiliencia de la arquitectura fue probada exitosamente bajo el estándar Wi-Fi (modo infraestructura) con el filtro inercial de la Media Móvil Exponencial evaluando caídas sostenidas de 8 dBm. Para futuras iteraciones, se recomienda desacoplar la capa de transporte y probar el mecanismo sobre topologías *ESP-Mesh* o redes de largo alcance y baja potencia (LoRaWAN / NB-IoT). Estudiar cómo reaccionan las curvas de la Lógica Difusa ante las latencias asimétricas y el ancho de banda extremadamente limitado de estas redes representaría un avance fundamental para despliegues de gran escala en *Smart Cities* y agricultura de precisión.

Bibliografía

Abdelaal, M. A., Moustafa, A. I., Kasban, H., Saleh, H., Abdallah, H. A., & Afifi, M. Y. I. (2025). DNA-Inspired Lightweight Cryptographic Algorithm for Secure and Efficient Image Encryption. *Sensors*, *25*(7), 2322. https://doi.org/10.3390/s25072322

Agbeyangi, A., & Suleman, H. (2024). Advances and Challenges in Low-Resource-Environment Software Systems: A Survey. *Informatics*, *11*, 90. https://doi.org/10.3390/informatics11040090

Al-Mousa, M. R., & others. (2024). A secure IoT limited Devices by applying Multi-Round Lightweight Cryptography. *2024 25th International Arab Conference on Information Technology (ACIT)*, 1–8. https://doi.org/10.1109/ACIT62805.2024.10876960

Aslan, B. (2025). Energy Consumption Analysis of ISO/IEC 29192-2 Standard Lightweight Ciphers. *Applied Sciences*, *15*(7), 3928. https://doi.org/10.3390/app15073928

Bhandiwad, V., & Ragha, L. K. (2025). Enhancing the security of IoT enabled systems using light weight hybrid cryptography models. *Cluster Computing*, *28*, 189. https://doi.org/10.1007/s10586-024-04897-3

Buitrón Ruiz, D. F. (2022). *Arquitecturas y modelos de referencia para sistemas IoT: Estado del arte de las arquitecturas para sistemas IoT*. http://bibdigital.epn.edu.ec/handle/15000/22368

De Santis, A., Ferrara, A. L., Flores, M., & Masucci, B. (2023). Continuous Entity Authentication in the Internet of Things Scenario. *Applied Sciences*, *13*(10). https://doi.org/10.3390/app13105945

Dhiman, S., Nayak, S., Mahato, G. K., & others. (2025). A survey analyzing encryption schemes for IoT security measures. *Journal of Supercomputing*, *81*, 798. https://doi.org/10.1007/s11227-025-07285-9

Faure, E., Rozlomii, I., Yarmilko, A., & Naumenko, S. (2023). Protection of IoT networks: Cryptographic solutions for cybersecurity management. En S. et al. Gnatyuk (Ed.), *Proceedings of the Conference*.

Hudda, S., & Haribabu, K. (2025). DySwitch: Dynamic Switching to Enable Secure and Energy Efficient Data Communication in Resource Constrained IoT Environment. En L. Barolli (Ed.), *Advanced Information Networking and Applications. AINA 2025* (Vol. 247). Springer, Cham. https://doi.org/10.1007/978-3-031-87769-8_6

Kumar, P. G. A., Jeyanthy, P. A., & Devaraj, D. (2022). Hybrid multi-objective method based on ant colony optimization and firefly algorithm for renewable energy sources. *Sustainable Computing: Informatics and Systems*, *36*, 100810. https://doi.org/https://doi.org/10.1016/j.suscom.2022.100810

Kumar, V. (2024). Challenges and Future Trends of Cryptography in Internet of Things. *INTERANTIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT*, *08*, 1–5. https://doi.org/10.55041/IJSREM30505

Mohamad Al-Aboosi, A. M., Kamil, S., Sheikh Abdullah, S. N. H., & Zainol Ariffin, K. A. (2021). Lightweight Cryptography for Resource Constraint Devices: Challenges and Recommendation. *2021 3rd International Cyber Resilience Conference (CRC)*, 1–6. https://doi.org/10.1109/CRC50527.2021.9392623

National Institute of Standards, & (NIST), T. (2023). *Finalist Announcement – Lightweight Cryptography Standardization Process*.

Oliveira Morilla, S. (2024). *Análisis integral de seguridad en dispositivos IoT*. https://hdl.handle.net/10609/150709

Ortega Chulde, C. A. (2023). *Implementación de mecanismo de seguridad para redes de sensores inalámbricos basado en criptografía de curva elíptica*. https://repositorio.utn.edu.ec/handle/123456789/14449

Ortega Patiño, A. I. (2023). *Aplicación de criptografía de curva elíptica para redes de sensores en sistemas de alerta temprana*. https://repositorio.utn.edu.ec/handle/123456789/14603

Rasheed, A. M., & Kumar, R. M. S. (2025). Efficient lightweight cryptographic solutions for enhancing data security in healthcare systems based on IoT. *Frontiers in Computer Science*, *7*, 1522184. https://doi.org/10.3389/fcomp.2025.1522184

Rhesri, A., Aabadi, F., Bennani, R., Ben Maissa, Y., Tamtaoui, A., & Dahmouni, H. (2025). Development of a Low-Cost Internet of Things Platform for Three-Phase Energy Monitoring in a University Campus. *IoT*, *6*(2). https://doi.org/10.3390/iot6020027

S., S. M., Martis, J. E., N, P. H., Ballal, C., & Deekshith. (2024). Lightweight Cryptographic Authentication Using Facial Landmarks and LFSR for IoT Security. *2024 International Conference on Computing, Semiconductor, Mechatronics, Intelligent Systems and Communications (COSMIC)*, 1–6. https://doi.org/10.1109/COSMIC63293.2024.10871851

Sarmila, K. B., & Manisekaran, S. V. (2025). IoT enabled data protection with substitution box for lightweight ciphers. *Egyptian Informatics Journal*, *29*, 100620. https://doi.org/10.1016/j.eij.2025.100620

Sehrawat, D., Gill, N. S., & Devi, M. (2019). Comparative Analysis of Lightweight Block Ciphers in IoT-Enabled Smart Environment. *2019 6th International Conference on Signal Processing and Integrated Networks (SPIN)*, 915–920. https://doi.org/10.1109/SPIN.2019.8711697

Singh, S., Sharma, P. K., Moon, S. Y., & others. (2024). Advanced lightweight encryption algorithms for IoT devices: survey, challenges and solutions. *Journal of Ambient Intelligence and Humanized Computing*, *15*, 1625–1642. https://doi.org/10.1007/s12652-017-0494-4

Srinivas, T. A. S., Donald, A. D., Srihith, I. D., Anjali, D., & Chandana, A. (2023). Small but mighty: The power of lightweight cryptography in IoT. *International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)*, *3*(1), 52–67. https://doi.org/10.48175/IJARSCT-9008

Unidas, N. (2030). *La Agenda 2030 y los Objetivos de Desarrollo Sostenible: una oportunidad para América Latina y el Caribe*. https://issuu.com/publicacionescepal/stacks

Anexo A - Tabla Comparativa de Algoritmos Criptográficos Ligeros

Algoritmo	**Tipo**	**Propiedades Clave (según Srinivas et al., 2023)**	**Referencia Primaria (Formato APA 7)**
AES-128	Cifrador de Bloque	Clave de 128 bits. Estándar optimizado.	National Institute of Standards and Technology. (2001). *FIPS PUB 197: Advanced Encryption Standard (AES)*. U.S. Department of Commerce.
PRESENT	Cifrador de Bloque	Bloque de 64 bits, clave de hasta 128 bits.	Bogdanov, A., Knudsen, L. R., Leander, G., Paar, C., Poschmann, A., Robshaw, M. J. B., ... & Vikkelsoe, C. (2007). PRESENT: An ultra-lightweight block cipher. En *Cryptographic Hardware and Embedded Systems* -

			CHES 2007 (pp. 450-466). Springer.
SIMON	Cifrador de Bloque	Familia de cifradores ligeros (varios tamaños).	Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., & Wingers, L. (2013). *The SIMON and SPECK families of lightweight block ciphers*. National Security Agency.
SPECK	Cifrador de Bloque	Familia de cifradores ligeros (varios tamaños).	Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., & Wingers, L. (2013). *The SIMON and SPECK families of lightweight block ciphers*. National Security Agency.
ASCON	Cifrado Autenticado (AEAD)	Estándar NIST FIPS 207. Bajos requisitos de memoria.	National Institute of Standards and Technology. (2023). *FIPS 207: ASCON*

			(Authenticated Encryption and Hashing). U.S. Department of Commerce.
Salsa20	Cifrador de Flujo	Pequeño tamaño de código, bajos requisitos de memoria.	Bernstein, D. J. (2008). The Salsa20 family of stream ciphers. En *New Stream Cipher Designs* (pp. 84-97). Springer.
ChaCha20	Cifrador de Flujo	Cifrado fuerte y alta eficiencia.	Nir, Y., & Langley, A. (2018). *ChaCha20 and Poly1305 for IETF Protocols*. RFC 8439. Internet Engineering Task Force.
Poly1305	MAC	Usado con ChaCha20 para autenticación.	Nir, Y., & Langley, A. (2018). *ChaCha20 and Poly1305 for IETF Protocols*. RFC 8439. Internet Engineering Task Force.

SKINNY	Cifrador de Bloque / AEAD	Bloque de 64 bits, clave de 128 bits.	Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, E., ... & Zieschang, T. (2016). SKINNY: A lightweight block cipher and its tweakable variant. En *Advances in Cryptology – CRYPTO 2016* (pp. 1-31). Springer.
CLEFIA	Cifrador de Bloque	Bloque de 128 bits, clave de 128/192 bits.	Shirai, T., Shibutani, K., Akishita, T., Moriai, S., & Iwata, T. (2007). CLEFIA: A 128-bit lightweight block cipher. En *Fast Software Encryption - FSE 2007* (pp. 181-195). Springer.
Kuznyechik	Cifrador de Bloque	Bloque de 128 bits, clave de 256 bits. (Estándar Ruso).	Dolmatov, V. (Ed.). (2016). *GOST R 34.12-2015: Block Cipher "Kuznyechik"*. RFC

			7801. Internet Engineering Task Force.
RECTANGLE	Cifrador de Bloque	Bloque de 64 bits, claves de 80/128/192 bits.	Zhang, W., Bao, Z., Lin, D., & Rijmen, V. (2014). RECTANGLE: A bit-slice lightweight block cipher. En *Selected Areas in Cryptography - SAC 2014* (pp. 176-194). Springer.
TWINE	Cifrador de Bloque	Bloque de 64 bits, claves de 80/128 bits.	Suzaki, T., Minematsu, K., Morioka, S., & Kobayashi, E. (2013). TWINE: A lightweight block cipher. En *Selected Areas in Cryptography - SAC 2012* (pp. 339-354). Springer.
ZUC	Cifrador de Flujo	Diseñado para 4G y 5G.	ETSI. (2011). *LTE; Specification of the 3GPP Confidentiality*

			and Integrity Algorithms UEA2 & UIA2; Document 2: ZUC specification (TS 135 221 V10.0.0).
Piccolo	Cifrador de Bloque	Para etiquetas RFID.	Referencia no provista en el documento fuente (Srinivas et al., 2023).
Sparx	Cifrador de Bloque	Bloque de 64 bits, clave de 128/256 bits.	Referencia no provista en el documento fuente (Srinivas et al., 2023).
PICAROON	Cifrado Autenticado	Bloque de 64 bits, clave de 128 bits.	Referencia no provista en el documento fuente (Srinivas et al., 2023).
Hummingbird	Autenticación	Pequeño tamaño de código.	Referencia no provista en el documento fuente (Srinivas et al., 2023).
Deoxys-BC	Cifrado Autenticado	Minimiza uso de memoria.	Referencia no provista en el documento fuente (Srinivas et al., 2023).
AEGIS	Cifrado Autenticado	Optimizado para recursos limitados.	Referencia no provista en el documento fuente (Srinivas et al., 2023).

MORUS	Cifrado Autenticado	Rango de tamaños de bloque/clave.	Referencia no provista en el documento fuente (Srinivas et al., 2023).
Ketje	Cifrado Autenticado	Rango de tamaños de bloque/clave.	Referencia no provista en el documento fuente (Srinivas et al., 2023).
TINYJAMBU	Cifrador de Bloque	Rango de tamaños de bloque/clave.	Referencia no provista en el documento fuente (Srinivas et al., 2023).
PRINCE	Cifrador de Bloque	Bloque de 64 bits, clave de 128 bits.	Referencia no provista en el documento fuente (Srinivas et al., 2023).
SPHINCS+	Esquema de Firma	Seguro contra ataques cuánticos.	Referencia no provista en el documento fuente (Srinivas et al., 2023).
Xoodyak	Cifrado Autenticado	Optimizado para baja potencia y memoria.	Referencia no provista en el documento fuente (Srinivas et al., 2023).

Anexo B - Repositorio del Código Fuente y Despliegue del Sistema

El código fuente desarrollado para la implementación de la arquitectura de seguridad adaptativa en entornos IoT ha sido alojado en un repositorio de control de versiones. Este repositorio contiene todos los módulos de software necesarios para reproducir el prototipo funcional descrito en la presente investigación.

Enlace al repositorio de GitHub: https://github.com/estebanvizca/Tesis

Estructura y Contenido del Repositorio: El proyecto se encuentra organizado en los siguientes directorios principales, correspondientes a los componentes arquitectónicos del sistema:

- **1. Nodo Maestro (Orquestador Edge):** Contiene el código fuente desarrollado en C++17. Incluye la lógica del cliente MQTT (Eclipse Paho), la API REST y WebSockets (Crow), y el motor de decisión inteligente. Además, aloja la suite criptográfica basada en OpenSSL.
- **2. Nodos Cliente (Sensores IoT):** Contiene el firmware desarrollado bajo el entorno ESP-IDF para la familia de microcontroladores ESP32 (ESP32-CAM, ESP32-WROOM y XIAO-ESP32-S3). Aquí se incluye el módulo de *PSK Bootstrapping* para la desencriptación inicial y la suite criptográfica ligera soportada por mbedTLS.
- **3. Modelos de Inteligencia Artificial:** Incluye los scripts de prototipado en Python utilizados para la validación teórica (Lógica Difusa y Perceptrón Multicapa). Asimismo, contiene el archivo de cabecera nativo de C++ (pesos_nn.h) que integra los tensores estáticos para la ejecución *Bare Metal* de la Red Neuronal.

Requisitos Previos para la Compilación: Para el correcto despliegue del sistema en un entorno real, se requiere instalar las siguientes dependencias detalladas en el documento:

- **Entorno Servidor (Orange Pi 5 Plus / Linux):** OpenSSL (v3.0), Eclipse Paho MQTT C++, nlohmann/json y Crow.

- **Entorno Cliente:** Framework ESP-IDF oficial de Espressif.