BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN

“ANÁLISIS Y DESARROLLO DE LOS SERVICIOS DE SEGURIDAD POST-CUÁNTICOS EN IOT”

TESIS

PARA OBTENER EL GRADO DE:

MAESTRA EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA:

ICC. ESTEFANIA SERRANO HERNÁNDEZ

DIRECTOR DE TESIS:

DR. MIGUEL ANGEL LEÓN CHAVEZ

16 de Noviembre 2022

Agradecimientos

Quisiera comenzar agradeciendo al Consejo Nacional de Ciencia y Tecnología (CONACyT) por la beca que me brindó durante el período en que estuve en la Maestría.

Estoy muy agradecido con el Dr. Miguel Angel León Chávez por supervisarme, por su apoyo continuo y compartir su conocimiento. Fue un gran ambiente de trabajo y de aprendizaje, estoy particularmente agradecido por haber tenido la oportunidad de trabajar y estudiar de tal manera que pudiera alcanzar mis objetivos.

Me gustaría agradecer a mis sinodales Dra. Mireya Tovar Vidal, Dr. Manuel Martín Ortiz, Dr. Mario Rossainz y Dra. Maya Carrillo por sus comentarios sobre esta tesis las cuales considero que enriquecen el contenido y logra hacer más claro lo que se quiere transmitir en esta.

Por último, pero no menos importante, me gustaría agradecer a mi familia, mi madre, padre, hermana y amigos quienes me apoyaron durante todo el camino y me brindaron mucha motivación y amor para lograr conseguir la meta que tanto deseaba desde el inicio de la Maestría.

Resumen

En este proyecto de tesis se propone desarrollar un análisis para identificar la criptografía asimétrica usada en dos arquitecturas del IoT, la propuesta por el IETF y la propuesta por el IEEE, la cual será vulnerable a ataques realizados en computadoras cuánticas masivas.

Se propone sustituir la criptografía asimétrica en el IoT por algoritmos post-cuánticos los cuales están siendo estandarizados por el Instituto Nacional de Estándares y Tecnología (NIST) y que satisfagan los requerimientos y las restricciones de seguridad del IoT.

Abstract

In this thesis project, it is proposed to develop an analysis to identify the asymmetric cryptography used in two IoT architectures, the one proposed by the IETF and the one proposed by the IEEE, which will be vulnerable to attacks carried out on massive quantum computers.

It is proposed to replace asymmetric cryptography in the IoT with post-quantum algorithms which are being standardized by the National Institute of Standards and Technology (NIST) and that satisfy the requirements and security restrictions of the IoT.

Índice general

AGRADECIMIENTOS 1
RESUMEN 2
ABSTRACT 3
CAPÍTULO 1 INTRODUCCIÓN 6
1.1 Criptografía de llave Simétrica 6
1.2 Criptografía de llave asimétrica 6
1.2.1 Algoritmo de Shor 7
1.2.2 Algorítmo de Grover 7
1.3 Criptografía Post-Cuántica 7
1.4 Definición del problema 8
1.5 Objetivos y preguntas de investigación 8
1.7 Estado del Campo del Arte 9
CAPÍTULO 2. IOT 10
2.1 Arquitecturas protocolarias en el IoT 10
2.1.1 Servicios 11
2.2 IEEE 802.15.4 12
2.2.1 Subcapa MAC del IEEE 802.15.4 12
2.2.2 Seguridad en la Capa MAC. 14
2.3 Capa IETF 6TOP 16
2.4 Capa IETF 6LoWPAN 16
2.4.1 Seguridad en 6LoWPAN 17
2.5 IPsec 18
2.5.1 Amenazas de seguridad 19
2.6 Capa IETF RPL 19
2.6.1 Seguridad en RPL 19
2.7 Subcapa IETF CoAP 20
2.7.2 DTLS 21
2.8 PILA ZIGBEE SE 22
2.8.1 Seguridad de la capa de red 23
2.8.2 La clave de red 23
2.9 Capa Aplicación 23

2.9.1Autenticación y cifrado 23
2.10 Conclusión del capítulo 24
CAPÍTULO 3. ALGORITMOS CRIPTOGRÁFICOS **25**
3.1 Criptografía post-cuántica 25
3.2 Criptografía basada en código 29
3.2.1 Criptografía basada en lattice 29
3.2.3 Criptografía basada en isogenia 30
3.3 Niveles de seguridad NIST 30
3.3.1 Espacio 30
3.3.2 Fallos 31
3.4 Evaluación usando ENCRYPT 32
CAPÍTULO 4. ANÁLISIS DE RENDIMIENTO **36**
Módulo ESP32 ..... **Error! Bookmark not defined.**
4.1 Análisis de Rendimiento Arduino 36
CAPÍTULO 5. CONCLUSIONES **40**
BIBLIOGRAFÍA **41**

Capítulo 1 Introducción

No cabe duda que los avances tecnológicos y en particular las comunicaciones electrónicas se han convertido en uno de los principales pilares tecnológicos de la era moderna. La necesidad de los servicios de seguridad en las comunicaciones, tales como: confidencialidad, integridad, autenticidad, control de acceso y no rechazo en la transmisión y el almacenamiento de datos hace que la ciencia de la criptografía sea una de las disciplinas más importantes de la tecnología de la información.

Hay dos tipos de criptosistemas: simétrico y asimétrico. En la primera dos usuarios comparten una llave o clave que mantienen en secreto y su fortaleza se basa en la longitud de la llave medida en bits; en la segunda cada usuario es propietario de dos llaves, una privada que mantiene en secreto y otra que hace pública a todos los usuarios del sistema y su fortaleza se basa en problemas computacionales difíciles de resolver usando computadoras clásicas.

La teoría de la computación cuántica, introducida por primera vez como concepto en 1982 por Richard Feynman, ha sido investigada extensamente y se considera la destructora de la criptografía asimétrica moderna actual. Además, es un hecho que la criptografía simétrica también puede verse afectada por algoritmos cuánticos específicos; sin embargo, su seguridad se puede aumentar con el uso de espacios de claves más grandes. Además, se han introducido algoritmos que pueden romper los actuales algoritmos criptográficos asimétricos cuya seguridad se basa en la dificultad de factorizar números primos grandes y el problema del logaritmo discreto. Parece que incluso la criptografía de curva elíptica, que actualmente se considera el esquema más seguro y eficiente, es débil frente a las computadoras cuánticas. En consecuencia, surgió la necesidad de diseñar algoritmos criptográficos resistentes a ataques usando conputadoras clásicas y cuánticas. [1]

1.1 Criptografía de llave Simétrica

En la criptografía simétrica, el remitente y el receptor utilizan la misma clave secreta y el mismo algoritmo criptográfico para cifrar y descifrar datos. Por ejemplo, Alicia puede cifrar un mensaje de texto claro sin formato utilizando su clave secreta compartida y Roberto puede descifrar el mensaje utilizando el mismo algoritmo criptográfico que utilizó Alicia y la misma clave secreta compartida. La clave debe mantenerse en secreto, lo que significa que solo Alicia y Roberto deben saberlo; por lo tanto, se exige una forma eficiente de intercambiar claves secretas a través de redes públicas inseguras. [1]

La criptografía asimétrica se introdujo para resolver el problema de la distribución de claves en la criptografía simétrica. Los algoritmos simétricos populares incluyen el estándar de cifrado avanzado (AES) y el estándar de cifrado de datos (3DES).

1.2 Criptografía de llave asimétrica

La fortaleza en seguridad de la criptografía asimétrica se basa en problemas computacionales como la dificultad de factorizar números enteros muy grandes (IFP) y el problema del logaritmo discreto (DLP). Este tipo de algoritmos se denominan funciones unidireccionales porque son fáciles de calcular en una dirección pero la inversión es difícil brevemente descritos a continuación.

- Problema de la factorización de números enteros (IFP): este problema se puede enunciar de la siguiente manera: dado un entero positivo n, encuentre sus factores primos, es decir escriba n = p1^e1 p2^e2 … pi^ei, donde pi son números primos diferentes para cada ei $\geq$1.

- Problema del logaritmo discreto (DLP): encuentre x de la ecuación y ≡ g^x mod p dados los parámetros del dominio (p, q, g) e y; donde p es un número primo, q es un divisor de p-1, y g ∈ [1, p-1] tiene un orden q (es decir, t=q es el entero positivo más pequeño que satisface q ≡ 1 mod p).

- DLP en Curvas Elípticas (EC): este problema se enuncia de la siguiente manera: encuentre k de la ecuación Q = kP, donde P y Q son dos puntos en una curva elíptica y k es un escalar.

Diffie-Hellman es un algoritmo de llave asimétrico que utiliza la propiedad antes mencionada para transmitir claves de forma segura a través de una red pública. Recientemente, se recomiendan claves mayores o iguales a 2048 bits para el intercambio seguro de claves. Además, se utiliza ampliamente otra familia de algoritmos de clave pública conocida como criptografía de curva elíptica. ECC proporciona el mismo nivel de seguridad que los sistemas RSA y DLP con operandos clave más cortos, lo que lo hace conveniente para ser utilizado por sistemas de bajos recursos computacionales.

1.2.1 Algoritmo de Shor

En 1994, el matemático Peter Shor en su artículo "Algorithms for Quantum Computation: Discrete Logarithms and Factoring" [2], demostró que la factorización de números enteros grandes cambiaría fundamentalmente con una computadora cuántica. El algoritmo de Shor puede hacer colapsar la criptografía de llave asimétrica moderna ya que resuelve los problemas IFP y DLP en tiempo polinomial.

1.2.2 Algorítmo de Grover

El algoritmo de Grover es uno de los principales algoritmos de la computación cuántica, el cual explota al máximo el principio fundamental de superposición, mostrando la superioridad de las computadoras cuánticas sobre las clásicas. Este algoritmo se conoce por su poder de búsqueda sobre un conjunto de datos no estructurados, es decir imaginemos que tenemos un conjunto con N datos de estudiantes, pero necesitamos encontrar uno en específico el cual para este algoritmo lo llamaremos target (w), si realizamos la acción de búsqueda sobre estos N estudiantes, con un algoritmo clásico debería mínimo buscarlo en $N/2$ operaciones. En un sistema cuántico dado la superposición de estados, este problema se puede resolver examinando simultáneamente todas las posibles combinaciones. Como resultado, el número de pasos para encontrar el nombre del estudiante que deseamos se puede obtener en solo O ($\sqrt{N}$) pasos. [3]

1.3 Criptografía Post-Cuántica

En 2016 el Instituto Nacional de Estándares y Tecnología (NIST) del Departamento de Comercio del gobierno de los Estados Unidos de Norteamérica convocó a proponer algoritmos criptográficos de llave pública post-cuánticos (Public-Key Post-Quantum Cryptographic, PQC) es decir resistentes a ataques por computadoras clásicas y cuánticas, que serán estandarizados por dicho organismo.

En 2017, se recibieron 69 algoritmos en la primera ronda, de los cuales en 2019 se seleccionaron 17 para la segunda ronda. En 2020 se anunció 4 algoritmos finalistas y 5 algoritmos alternos de la tercera ronda. Se espera que en 2024 se publique el primer borrador del estándar para su revisión final.

Los algoritmos de llave pública post-cuánticos de la tercera ronda son: Classic McEliece, CRYSTALS-KYBER, NTRU y SABER.
Y los algoritmos alternos son: BIKE, FrodoKEM, HQC, NTRU Prime y SIKE.

1.4 Definición del problema

El Protocolo de Internet versión 6 (IPv6) especifica direcciones de 128 bits, es decir 2^128 = 3.4 x 10^38 posibles direcciones IP, por lo que es posible conectar cualquier cosa a la red Internet. Existen diversas definiciones del Internet de la Cosas (IoT) y todas coinciden en la conexión a la red Internet de objetos simples con bajo o nulo poder de procesamiento (CPU), baja capacidad de almacenamiento (memoria) y bajo consumo de energía.

Existen varias arquitecturas del IoT pero todas usan criptografía de llave simétrica para implementar el servicio de seguridad de confidencialidad y usan criptografía de llave asimétrica para generar la llave secreta.

Este proyecto de tesis analiza los algoritmos criptográficos post-cuánticos finalistas de la tercera ronda del NIST para determinar cuál o cuáles son los más convenientes para implementar en el IoT tomando en cuenta por un lado las restricciones de los objetos simples y por otro lado el rendimiento de los algoritmos post-cuánticos.

Los resultados de este estudio serán útiles para ayudar al IoT a seleccionar las soluciones criptográficas adecuadas para la computación postcuántica.

1.5 Objetivos y preguntas de investigación

A partir de esta investigación se planea generar las bases para una herramienta que pueda ser utilizada en las pilas protocolarias en el IoT.

El objetivo general de este trabajo de tesis puede establecerse como sigue:

Análisis y desarrollo de los servicios de seguridad post-cuánticos en IoT

Para lograrlo, debe trabajarse en los siguientes objetivos específicos:

1. Analizar las arquitecturas de seguridad del IoT y determinar qué servicios de seguridad implementan y qué algoritmos criptográficos utilizan.
2. Identificar los algoritmos criptográficos de llave asimétrica que ya no serán seguros una vez que existan grandes computadoras cuánticas.
3. Analizar qué algoritmos post-cuánticos del concurso del NIST que satisfacen los requerimientos y las restricciones del IoT.
4. Analizar el rendimiento de los algoritmos post-cuánticos en el IoT.

Analizando los objetivos de la investigación se propone la siguiente pregunta en general: ¿Es posible encontrar un algoritmo post-cuántico con el uso de computadoras clásicas, que cumpla los requerimientos de seguridad confidencialidad, integridad, autenticación, control de acceso y no rechazo en el IoT, y así sustituir los algoritmos clásicos por uno de ellos?

En relación con los objetivos particulares, se trabajan las siguientes preguntas de investigación.

1. ¿Cuáles son los algoritmos pos-cuánticos a trabajar más adecuadas?

2. ¿Qué herramienta de software y hardware es el más adecuado para trabajar dichos algoritmos?

3. ¿Cuáles son las restricciones que se nos pueden presentar al trabajar con algoritmos post-cuánticos?

1.6 Limitaciones

Los sistemas ciberfísicos tienen consecuencias en el mundo real si se ven comprometidos. Incluso si un atacante no puede obtener el control, si puede acceder a la información, puede aprender, por ejemplo, cuando el propietario está ausente y otros hábitos potencialmente comprometedores [2] [3]. En este caso, los sistemas ciberfísicos incluyen no solo equipos como cámaras o controles de puerta, sino también infraestructuras inteligentes como SmartThings de Samsung y AWS IoT de Amazon.

La vida útil del dispositivo también debe tenerse en cuenta al evaluar los riesgos. Esto es especialmente cierto con los sistemas ciberfísicos, que a menudo pasan años entre actualizaciones [3]. Si un dispositivo se implementará durante 20 años, su hardware actual debe admitir todos los parches de software futuros. A pesar de que la aceleración del hardware logra un rendimiento excelente (especialmente el consumo de energía) [3], se desaconseja para los dispositivos integrados por esta razón, ya que proporciona una falta de flexibilidad en caso de que el algoritmo deba cambiarse o reemplazarse en el futuro. Es esencial considerar la duración del tiempo que el cifrado necesita para permanecer seguro y cuántos años llevará convertir nuestra infraestructura de clave pública (PKI) a cuántica segura. Si la suma de estos períodos es mayor que el tiempo que tardará en desplegarse una computadora cuántica viable, entonces esos datos están en riesgo [3]. Si no se mantiene el secreto hacia adelante, entonces también se debe agregar a este período la duración en la que los datos permanecerán vulnerables.

1.7 Estado del Campo del Arte

“RLizard: Post-Quantum Key Encapsulation Mechanism for IoT Devices”

Propone el mecanismo de encapsulación de claves (KEM) de RLizard, cuya seguridad depende del aprendizaje del anillo con errores y del aprendizaje del anillo con problemas de redondeo. Debido a que RLizard opera en un tipo especial de anillo, es más eficiente en términos de los ciclos de reloj necesarios para la generación de claves y el tamaño de la clave en comparación con el esquema original de Lizard. Para demostrar la superioridad del método propuesto sobre otros KEM conocidos, este trabajo compara sus rendimientos en el entorno ARM Internet of Things (IoT) de 32 bits. El análisis de rendimiento mostró que RLizard KEM requiere la menor cantidad de ciclos de reloj para la generación, encapsulación y desencapsulación de claves cuando los parámetros se configuran para admitir un nivel de seguridad comparable con el de AES-128. En resumen, se espera que RLizard KEM se utilice para la comunicación y la autenticación seguras entre los dispositivos de punto final de IoT, cuyo poder de cómputo generalmente es limitado.

“Saber Post-Quantum Key Encapsulation Mechanism (KEM): Evaluating Performance in Mobile Devices and Suggesting Some Improvements”

Saber es uno de los cuatro finalistas en el Proceso de estandarización de criptografía post-cuántica del NIST en curso. Es uno de los tres finalistas que se basan en problemas de retículas. Este trabajo pretende mostrar los resultados de un análisis del desempeño de Saber en arquitecturas x64 y ARM. Saber se probó siguiendo un escenario en el que primero se generaron un par de claves públicas y privadas. Luego, los datos que representan una clave de sesión se cifran con la clave pública generada y se descifran con la clave privada generada. Se evaluó el desempeño del algoritmo ejecutando estos pasos en cada arquitectura propuesta. Según los datos recopilados, se verifica si Saber es adecuado para dispositivos móviles o no. Se encontraron cuellos de botella al ejecutar el código de Saber[3].

Capítulo 2. IoT

El concepto de Internet de las Cosas (Internet of Things, IoT) fué introducido por primera vez por Kevin Ashton en 1999 como “associating physical objects with RFID tags, each object could be given an identity for generating data on such things”. [4]

El ITU lo define como “A dynamic global network infrastructure with self-configuring capabilities based on standard and interoperable communication protocols where physical and virtual “things” have identities, physical attributes and virtual personalities, use intelligent interfaces and are seamlessly integrated into the information network”.

Por su parte el IETF [4] lo define como “the network of physical objects or "things" embedded with electronics, software, sensors, and connectivity to enable objects to exchange data with the manufacturer, operator and/or other connected devices”.

Lo cierto es que han hecho realidad la idea de Internet de las cosas (IoT) con aplicaciones en la automatización de la industria, los edificios y del hogar, en las ciudades inteligentes, en la administración de la energía, en la agricultura, en la salud, etc.

El IoT permite que los objetos físicos vean, escuchen, piensen y realicen trabajos haciéndolos "hablar" juntos, compartir información y coordinar decisiones. El IoT transforma estos objetos de tradicionales a inteligentes mediante la explotación de sus tecnologías subyacentes, como la computación ubicua y generalizada, los dispositivos integrados, las tecnologías de comunicación, las redes de sensores, los protocolos y las aplicaciones de Internet [4][5] .

En este capítulo se analizará las dos pilas protocolarias en el Internet de las cosas, la especificada por el IETF y la propuesta por el IEEE 802.15.4 o ZigBee, determinar qué servicios de seguridad implementan y qué algoritmos criptográficos utilizan en cada una de sus capas, así como identificar los algoritmos criptográficos de llave asimétrica.

2.1 Arquitecturas protocolarias en el IoT

Las pilas de protocolos de IoT contienen varias capas, como se muestra en la Fig1. El objetivo del este capítulo es identificar los servivicios de seguridad y los algoritmos criptográficos utilizados en las capas de ambas pilas protocolarias del IoT.

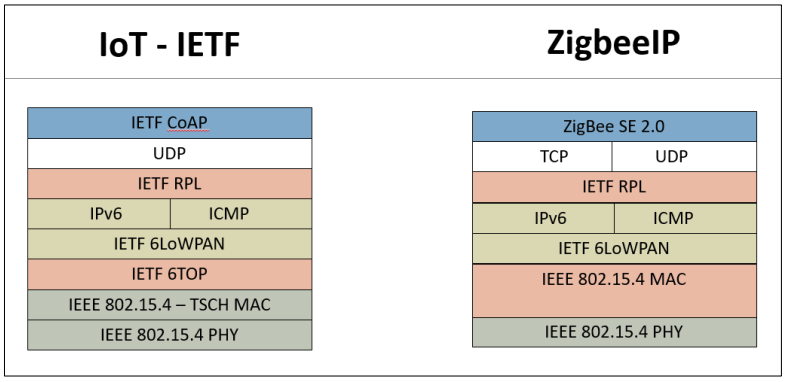


Fig 1. Pilas protocolarias del Internet de las Cosas a) Pila de IETF b) Pila Zigbee SE

2.1.1 Servicios

A continuación se describen brevemente los servicios que ofrece cada capa.

a) Capa Física
1) Transformar datos en una señal electromagnética, decodifica y transmitirlos en ciertos rangos de frecuencia.

b) Capa Mac
1) Provee servicio de control de acceso al medio (CSMA/CA)

c) 6LoWPAN
1) Fragmentación y re ensamblaje
2) Compresión de encabezado
3) Autoconfiguración de direcciones
4) Protocolo de enrutamiento de malla
5) Administración de redes
6) Consideraciones de implementación
7) Consideraciones sobre la aplicación y la capa superior
8) Resuelve el problema de direcciones IPV4.
9) Implementa servicios de seguridad.

d) ICMP
1) Permite reportar errores en la transmisión de un paquete.

e) IETF RPL
1) Protocolo de ruteo de paquetes IP para las redes de baja potencia.
2) Construye grafo acíclico dirigido orientado al destino (DODAG).
3) Define objetos de información.
4) Objetos que anuncian el destino en el grafo.
5) Define reconocimientos.
6) Los ruteadores en cada nivel registran la información.
7) Construcción del grafo de arriba hacia abajo.

f) CoAP
1) Protocolo a nivel aplicación para ambientes restringidos.
2) Transformar el contenido (REST).
3) Usa UDP
4) Descubrir recursos.
5) Objetos de seguridad (CoRE).
6) Define subcapas.
7) Se monta sobre DTLS para proveer servicios de seguridad y confidencialidad.

2.2 IEEE 802.15.4

Como se puede observar en la Fig. 1, ambas arquitecturas de IoT especifican el uso de la capa física y la subcapa MAC del IEEE 802.15.4. El cual especifica las redes inalámbricas de área personal y de baja velocidad (Low-Rate Wireless Personal Area Network, LR-WPAN) y sus principales características son las siguientes:

a) Velocidades de datos de 250kb/s, 100kb/s, 40kb/s y 20kb/s.
b) Topologías de red en estrella y operación punto a punto (peer-to-peer).
c) Asignación de direcciones cortas de 16 bits o extendida de 64 bits.
d) A nivel de la subcapa MAC el protocolo de control de accesos por sensado de portadora y evitando colisiones (carrier sense multiple access with collision avoidance, CSMA-CA).
e) Opcionalmente permite reservar ranuras de tiempo garantizando para garantizar la transmisión (guaranteed time slots, GTS).
f) Además permite confirmar la transferencia confiables de los datos por medio de tramas de reconocimiento (acknowledged) .
g) Bajo consumo de energía e indicación de la energía.
h) Ofrece indicación de la calidad del enlace.
i) A nivel de capa física ofrece 16 canales en la banda de los 2450 MHz, 30 canales en la banda de 915 MHz, y 3 canales en la banda de los 868 MHz.

A continuación se identifica los servicios de seguridad y los algoritmos criptográficos de este estándar.

2.2.1 Subcapa MAC del IEEE 802.15.4

El estándar IEEE 802.15.4 soporta los siguientes servicios de seguridad:

- Confidencialidad de los datos.
- Autenticidad de los datos.
- Protección contra repetición.

La subcapa MAC es la responsable de proporcionar estos servicios de seguridad a las tramas salientes y entrantes cuando se demanda por los niveles superiores. No obstante, la implementación de seguridad de un dispositivo es opcional, comportándose de una de las siguientes formas:

- **Cuando un dispositivo no implementa seguridad**, no debe proporcionar un mecanismo a la subcapa MAC para realizar ninguna transformación criptográfica en tramas salientes o entrantes ni requerir ningún atributo asociado con seguridad.
- **Si un dispositivo implementa seguridad**, debe proporcionar un mecanismo a la subcapa MAC para proporcionar transformaciones criptográficas con seguridad sólo si el atributo *macSecurityEnabled* está puesto a TRUE.

Así, cuando se ofrece servicios de seguridad, existen tres campos en la trama MAC que están relacionados con esta, como se muestra en la Fig. 2:

- **Control de trama** (*Frame Control*), ubicado en el encabezado MAC.
- **Control de Seguridad Auxiliar** (*Auxiliary Security Control*, también en el encabezado MAC).
- **Carga útil de datos** (*Data Payload*), en el campo de carga útil del MAC

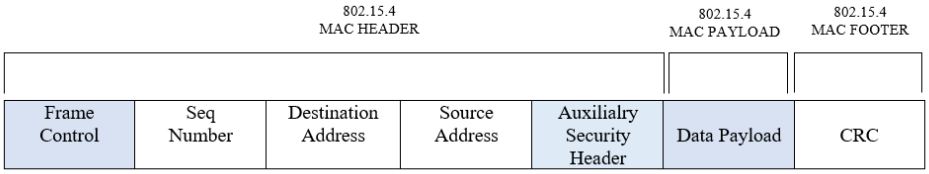


Fig 2.Trama subcapa MAC con información de seguridad

La Trama de Seguridad Auxiliar (*Auxiliary Security Frame*) sólo se habilita si el subcampo *Security Enabled* de la Trama de Control está activo. Este encabezado especial tiene tres campos, mostrados en la Fig. 3:

- **Security Control** (1 byte), especifica qué clase de protección se usa (ver más abajo).
- **Frame Counter** (4 bytes), es un contador dado por el origen de la trama actual para proteger el mensaje de ataques de repetición. Por esta razón, cadamensaje tiene un ID de secuencia único representado por este campo.
- **Key Identifier** (0 - 9 bytes), especifica la información necesaria para conocer qué clave se está usando con el nodo con el que se comunica.

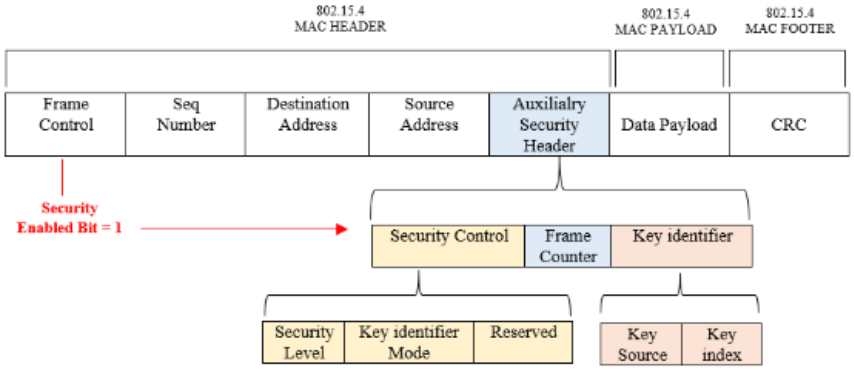


Fig. 3 Campos de Cabecera de Seguridad Auxiliar IEEE 802.15.

El *Security Control* es el lugar donde se especifica la Política de Seguridad (*Security Policy*) global. Usando los primeros 2 bits (campo *Security Level*) se elige qué se cifra y cuál será la longitud de la clave: el valor 0x00 significa sin cifrado, de manera que los datos ni se cifran (ausencia de confidencialidad de los datos) ni se valida su autenticidad (ausencia de validación de integridad de los datos). Desde 0x01 a 0x03, los datos se autentican usando el Código de Autenticación de Mensaje (MAC). El valor 0x04 cifra la carga útil asegurando la confidencialidad de los datos. Los valores en el rango 0x05 a 0x07 aseguran tanto confidencialidad de los datos como su integridad (autenticidad).

El campo de datos de carga útil puede tener tres configuraciones diferentes, dependiendo de los campos de seguridad definidos:

- **AES-CTR**: todos los datos se cifran usando el Estándar de Cifrado Avanzado (AES) con una clave definida de 128 bits, el contador de trama establece un ID de mensaje único, y el Key Counter (campo Key Control) se usa por la capa de aplicación si se alcanza el máximo valor de

Frame Counter.

- **AES-CBC-MAC**: el Código de Autenticación de Mensaje (MAC) se añade al final del payload, con una longitud que depende del nivel de seguridad especificado en el campo Security Policy. El MAC se crea cifrando la información de la cabecera MAC 802.15.4 y la carga útil de datos.

- **AES-CCM**: Es la mezcla de los dos modos de operación anteriores, donde los subcampos se corresponden con el modo AES-CTR más el subcampo AES-CBC-MAC cifrado.

Aparte de proporcionar los servicios de seguridad formateando las tramas salientes e interpretando y gestionando las entrantes, cada transceptor 802.15.4 tiene que controlar su propia lista de control de acceso (ACL, *Access Control List*) para controlar los pares "de confianza", junto con la política de seguridad correspondiente para cada uno. Esta lista, entre otras cosas, contiene la siguiente información:

- **Dirección**: del nodo con el que se quiere comunicar.

- **Suite de seguridad** (*Security Suite*): la política de seguridad que está siendo aplicada ((AES-CTR, AES-CCM-64, AES-CCM-128, etc.).

- **Clave**: la clave de 128 bits usada en el algoritmo AES.

- **Último vector de inicialización** (IV, *Initialization Vector*) y contador de repeticiones (*Replay Counter*), para evitar ataque de repetición.

Cuando un nodo quiere recibir un mensaje de otro nodo (o quiere enviar un paquete a un nodo específico), primeramente, comprueba que dicho nodo se encuentra en su ACL (se trata de un nodo de confianza). En caso afirmativo, utiliza los datos de configuración específicos extraídos de la lista y aplica las medidas de seguridad adecuadas; en caso contrario (no es un nodo de confianza), o bien rechaza el mensaje o comienza un proceso de autenticación con el nodo remoto.

2.2.2 Seguridad en la Capa MAC.

La seguridad de la capa MAC se basa en la seguridad de IEEE 802.15.4 aumentada con CCM*. CCM (Cipher Block Chaining - Message Authentication Code) es un modo de operación de los algoritmos criptográficos de llave simétrica que combina dos modos de operación: el modo contador (CTR) y el modo de encadenamiento por bloque (CBC). CCM ofrece los servicios de seguridad de confiabilidad e integridad. CCM* es CCM con capacidades de sólo cifrado y sólo integridad. La capa MAC usa una sola clave para todos los niveles de seguridad CCM*.

Como parte del modelo de confianza abierta, la capa MAC es responsable de su propio procesamiento de seguridad, pero las capas superiores determinan qué claves o niveles de seguridad usar. La capa superior hace que coincida la clave por defecto para la capa MAC con la clave de red activa, y las claves de enlace de la capa MAC con cualquiera de las claves de enlace de la capa superior.

*Modo de operación CCM**

Se introduce un modo CCM*, en el que se utiliza un contador determinado a partir del contador de tramas del dispositivo fuente para proporcionar “frescura” de la trama y evitar el ataque de repetición. Para cada nodo al que un dispositivo envía o recibe tramas protegidas, se crea una entrada ACL en el PIB de MAC, que contiene la dirección implícita o explícita de la entidad y el material de seguridad correspondiente asociado, incluida una clave AES, un contador de tramas para tramas salientes, y un contador de tramas externo para tramas entrantes. Si es explícito, contiene un identificador de clave.

Entradas

La transformación directa del modo CCM * toma lo siguiente como entradas:

a) Una clave de cadena de bits de longitud keylen bits que se utilizará como clave. Cada entidad deberá tener evidencia de que el acceso a esta clave está restringido a la propia entidad y a los miembros del grupo de intercambio de claves previstos.

b) Un número aleatorio (nonce) N de 15 - L octetos. Dentro del alcance de cualquier clave de cifrado, el valor nonce será único.

c) Una cadena de octetos m de longitud l (m) octetos, donde $0 \leq$ l (m) $< 2^{8L}$

d) Una cadena de octetos a de longitud l (a) octetos, donde $0 \leq$ l (a) $< 2^{64}$

Transformación de cifrado

a) Formar el campo Banderas de 1 octeto que consta de dos campos reservados de 1 bit y representaciones particulares de 3 bits de los enteros 0 y L, como se indica a continuación:

Flags = Reservado || Reservado || 0 || L

Aquí, los dos campos reservados de 1 bit están reservados para futuras expansiones y se pondrán a '0'. El campo '0' es la representación de 3 bits del entero 0, en el orden de primer bit más significativo. El campo L es la representación de 3 bits del entero L - 1, en el orden de primer bit más significativo.

b) Defina el campo Ai de 16 octetos que consiste en el campo Flags de 1 octeto definido en el paso a) en esta subcláusula, el campo Nonce de (15 - L) octetos N, y la representación de L octetos del entero i, como sigue:

Ai = Flags || Nonce N || Contador i, para i = 0, 1, 2,…

Tenga en cuenta que esta definición garantiza que todos los campos Ai sean distintos de los campos B0 que se utilizan realmente, ya que tienen un campo Flags con una codificación de M distinta de cero en las posiciones donde todos los campos Ai tienen una codificación todo cero del entero 0.

c) Analizar el mensaje PlaintextData como M1 || … || Mt, donde cada bloque de mensajes Mi es una cadena de 16 octetos.

d) Los bloques de texto cifrado C1,…, Ct se definen de la siguiente manera:

Ci: = E (Clave, Ai) + Mi para i = 1, 2,..., t

La cadena Ciphertext es el resultado de omitir todos los octetos l (m) más a la izquierda de la cadena C1 || … || Connecticut. (¿??)

f) Defina el bloque de cifrado de 16 octetos S0 como sigue:

S0: = E (clave, A0)

g) La etiqueta de autenticación cifrada U es el resultado de aplicar XOR a la cadena que consta de los octetos M más a la izquierda de S0 y la etiqueta de autenticación.

2.3 Capa IETF 6TOP

6P forma parte de la subcapa de operación 6TiSCH (6TOP), la capa que está justo por encima de la capa de control de acceso al medio TSCH del estándar IEEE 802.15.4.

Las funciones de 6TOP son

- Terminar el 6P, que permite a los nodos vecinos comunicarse para añadir/eliminar celdas entre sí.
- Ejecutar uno o varios SF (antes de usar abreviaciones escribirlo completo, qué es SF?) de 6TOP, que definen las reglas que deciden cuando añadir/borrar celdas.

Los mensajes 6P se transportan dentro de los elementos de información de carga útil (IEs) de 802.15.4. Estos IEs de carga útil se cifran y autentifican en la capa de enlace mediante CCM*. 6P se beneficia del mismo nivel de seguridad que cualquier otro IE de carga útil. 6P no define sus propios mecanismos de seguridad. En particular, 6P se beneficiará de la solución de gestión de claves utilizada en la red. Esto es relevante, ya que los ataques a la seguridad, como los de falsificación y atribución errónea, se vuelven más perjudiciales cuando se comparte una sola clave entre un grupo de más de dos participantes.

6P no proporciona protección contra los ataques de negación de servicio (DoS). Algunos ejemplos de ataques son el no envío de mensajes de confirmación en las transacciones de 3 pasos y el envío de solicitudes con un formato incorrecto. Estos casos deberían ser manejados por una política apropiada, como la limitación de la tasa o la inclusión del atacante en una lista negra después de varios intentos [6].

El efecto sobre la red en general se localiza sobre todo en los dos nodos en cuestión, ya que la comunicación se produce en celdas dedicadas.

2.4 Capa IETF 6LoWPAN

El concepto 6LoWPAN fue originado por la idea de que "el Protocolo de Internet puede y debe ser aplicada incluso a los más pequeños dispositivos", y que los dispositivos de baja potencia con capacidades de procesamiento limitadas deben ser capaces de participar en el Internet de las Cosas [7].

El objetivo inicial fue definir una capa de adaptación para hacer frente a las exigencias impuestas por IPv6, tales como el aumento del tamaño de la dirección y la MTU byte de 1280. Se han definido mecanismos de encapsulación y compresión de cabecera que permiten a los paquetes IPv6 ser enviados y recibidos en redes basadas en de IEEE 802.15.4 con MTU más pequeñas. La compresión produce cabeceras tan pequeñas de sólo 4 bytes, mientras que al mismo tiempo permite el uso de diferentes tipos de redes de malla y gestiona la fragmentación y re ensamblaje donde sea necesario.

Esta especificación espera que la capa de enlace esté suficientemente protegida, ya sea por medio de seguridad física o IP para el enlace principal, o con criptografía de subcapa MAC. Sin embargo, es posible que el cifrado y la autenticación de la capa de enlace no sean suficientes para brindar confidencialidad, autenticación, integridad y actualización tanto a los datos como a los paquetes de protocolo de enrutamiento. La sincronización horaria, la auto organización y la localización segura para el enrutamiento de varios saltos también son fundamentales para el soporte.

En consecuencia, la seguridad de tales dispositivos puede depender en gran medida de los mecanismos definidos en la subcapa MAC del IEEE 802.15.4. Sin embargo, este último sólo define AES-CCM* para la

autenticación o el cifrado de tramas IEEE 802.15.4 y, en particular, no especifica la gestión de claves (presumiblemente orientada a grupos). Otros problemas que se deben abordar en implementaciones reales se relacionan con la configuración y la administración seguras.

2.4.1 Seguridad en 6LoWPAN

El aspecto de la seguridad, sin embargo, parece una pequeña compensación en el 6LoWPAN, ya que la seguridad es siempre una función costosa. Esto es particularmente cierto para WPAN de baja tasa. Obviamente, agregar seguridad hace que el problema sea aún más desafiante.

Por ejemplo, al colocar IPv6 en la parte superior de 6LoWPAN puede parecer posible utilizar el protocolo de seguridad de IPv6 [RFC4301] y desactivar el mecanismo de seguridad definido por el IEEE 802.15.4. Pero, por otro lado, IPsec es relativamente maduro para servicios en IP o capas superiores. Además, debido a sus propiedades inherentes y / o restricciones mencionadas anteriormente, 6LoWPAN plantea desafíos únicos a los que las técnicas de seguridad tradicionales no se pueden aplicar directamente [8][9].

Por ejemplo, las primitivas de criptografía de clave pública generalmente se evitan (por ser demasiado caras, en procesamiento y almacenamiento) al igual que los métodos de cifrado convencionales relativamente pesados. En consecuencia, resulta cuestionable si los dispositivos 6LoWPAN pueden admitir IPsec tal cual [10].

Sin embargo, la seguridad de la capa 2 se debe utilizar para todas las operaciones asociadas, como la asociación de subcapa MAC, etc. 6LoWPAN puede necesitar definir sus propios métodos de gestión de claves que requieran una sobrecarga mínima en el tamaño del paquete y en el número de intercambio de mensajes de señalización [11]. IPsec proporcionará autenticación y confidencialidad entre los nodos finales y a través de múltiples enlaces LoWPAN y puede ser útil solo cuando dos nodos desean aplicar seguridad a todos los mensajes intercambiados.

Sin embargo, en la mayoría de los casos, la seguridad puede solicitarse en la capa de aplicación según sea necesario, mientras que otros mensajes pueden fluir en la red sin seguridad.

Los requisitos de seguridad para LoWPAN: son los siguientes:

- Confidencialidad de los datos: hacer que la información sea inaccesible para usuarios no autorizados. Por ejemplo, un nodo no debería filtrar algunos de sus datos recopilados a las redes vecinas [12][13].

- Autenticación de datos: dado que un adversario puede inyectar mensajes fácilmente, el receptor debe asegurarse de que los datos se originen en fuentes confiables.

- Integridad de datos: asegura que los datos recibidos no sean alterados en tránsito por un atacante.
- Actualización de los datos: esto podría significar la actualización de los datos y la actualización de las claves. De manera informal, la actualización de los datos implica que cada dato es reciente y asegura que ningún adversario reproduzca mensajes antiguos.

- Disponibilidad: garantiza la supervivencia de los servicios de red a (solo) partes autorizadas cuando sea necesario, a pesar de los ataques DoS.

- Robustez: asegura la continuidad de la operación a pesar de anomalías, como ataques, nodos fallidos, etc.

- Resistencia: es la capacidad de la red para evitar que el atacante obtenga el control total de la red mediante un ataque de replicación de nodos en caso de que algunos nodos estén comprometidos.

2.5 IPsec

La seguridad del protocolo de Internet (IPsec) es una de las principales capas de protocolos que los desarrolladores utilizan para asegurar el intercambio de datos a través de una red. Con IPsec, el propio dispositivo no necesita gestionar el cifrado, por lo que los intercambios de datos utilizan menos potencia de procesamiento y consumen menos datos. Esto es especialmente valioso para los dispositivos que no tienen los recursos para cifrar las transmisiones con Transport Layer Security y Secure Sockets Layer (TLS/SSL) [RFC 2401, 4301, 3138].

IPsec [RFC 2401, 4301, 3138] incluye tres protocolos distintos que aseguran diferentes aspectos de las comunicaciones de red:

- Cabeceras de autenticación (AH) [RFC 2402, 4302, 4305]
- Carga útil de seguridad encapsulada (ESP) [RFC 2406, 4303, 4305]
- Protocolo de Asociación de Seguridad de Internet y Gestión de Claves (ISAKMP) [RFC 2408, 4306]

Las cabeceras de autenticación y la carga útil de seguridad encapsuladas pueden utilizarse juntas o por separado, e ISAKMP reacciona a cuál de estos protocolos se utiliza para la transmisión.

IPsec requiere que dos pares que se comuniquen compartan una clave secreta que normalmente se establece dinámicamente con el protocolo de intercambio de claves de Internet (IKEv2). Por lo tanto, tiene una sobrecarga de paquetes adicional incurrida por el intercambio de paquetes IKEv2.

Esta clase de esquema de gestión de claves depende de la criptografía asimétrica, como los certificados de clave pública. Todo esto tiene un precio, a menudo impuesto por los recursos limitados de computación y energía de los nodos 6lowpan. Sin embargo, estos son los criptoanálisis más difíciles. Algunos de los ejemplos más populares incluyen, entre otros, el acuerdo de claves Diffie-Hellman, RSA o ECC [RFC2631]. Trabajos recientes sobre la implementación de ECC para dispositivos de baja potencia han demostrado su viabilidad para redes de sensores. ECC proporciona seguridad con un tamaño de clave más pequeño que es comparable a la seguridad proporcionada por RSA con un tamaño de clave mucho mayor.

ISAKMP define los procedimientos y formatos de paquetes para establecer, negociar, modificar y eliminar las SA (asociaciones de seguridad). Una Asociación de Seguridad contiene toda la información necesaria para la ejecución de diversos servicios de seguridad de red, como a nivel IP (IPsec AH o ESP), transporte o servicios de la capa de aplicación. ISAKMP define el formato para el intercambio de generación de claves y datos de autenticación. Proporciona un marco coherente para la transferencia de claves y datos de autenticación, que es independiente de la técnica de generación de claves, el algoritmo de cifrado y el mecanismo de autenticación [RFC2408].

ISAKMP se distingue del protocolo Internet Key Exchange (IKE) de intercambio de claves por separar los detalles de la seguridad de administración de la asociación (y de gestión de claves) de los detalles de intercambio de claves. [RFC2408]. IKE emplea un intercambio secreto de claves de tipo Diffie-Hellman para establecer el secreto compartido de la sesión. Se suelen usar sistemas de clave pública o clave pre-compartida.

2.5.1 Amenazas de seguridad

6lowpan es muy susceptible a los ataques físicos, es decir, amenazas debidas a la reubicación y el enmascaramiento de la destrucción de nodos físicos. Mediante ataques físicos, uno o varios nodos de 6lowpan pueden quedar fuera de servicio de forma permanente, por lo que las pérdidas son irreversibles.

El ataque físico puede extraer secretos criptográficos de los circuitos asociados, modificar la programación en los nodos y puede permitir que el nodo malintencionado tome el control sobre ellos. Estos compromisos pueden resultar en la modificación del código dentro del nodo y cambiar el rol orientado a la misión de las redes completas, y mucho menos de los sensores. [RFC2408]

2.6 Capa IETF RPL

Los protocolos de enrutamiento son un componente esencial de las redes convencionales, y esto también se aplica a las redes 6LoWPAN.

RPL [RFC 6550] es un protocolo de enrutamiento optimizado para IPv6 diseñado por el IETF especialmente para redes de baja potencia y con pérdidas (LLN) y que se utiliza principalmente en las redes 6LoWPAN. RPL es un protocolo de enrutamiento vectorial de distancia, y su topología de mapeo se basa en una estructura de gráfico acíclico dirigido orientado al destino (DODAG)[14].

En el árbol DODAG, es esencial que los nodos seleccionen sus nodos padres correctos, ya que cada nodo, excepto la raíz, debe tener un nodo padre. El rango RPL se utiliza para describir la posición de un nodo en la topología del árbol. Los autores presentan un esquema de selección seguro para ayudar a un nodo hijo a elegir un nodo padre auténtico. En su algoritmo de selección, el valor del umbral de un nodo se calculará en base a los valores de rango medio y máximo de sus nodos vecinos para excluir a los nodos falsos de convertirse en su padre [16]. Así, las soluciones existentes pueden garantizar la generación de tablas de enrutamiento seguras en las redes domésticas inteligentes.

2.6.1 Seguridad en RPL

Cada mensaje RPL tiene una variante segura. Las variantes seguras ofrecen protección contra la integridad y la repetición, así como protección opcional contra la confidencialidad y el retraso. Dado que la seguridad abarca tanto el mensaje base como las opciones, en los mensajes seguros la información de seguridad se encuentra entre la suma de comprobación y la base. El nivel de seguridad y los algoritmos utilizados se indican en los mensajes del protocolo, como se describe a continuación:

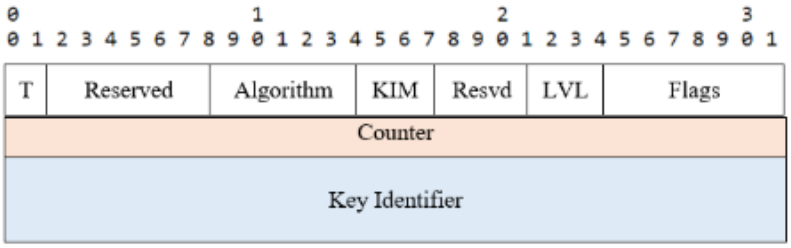


Fig. 4 Seguridad en RPL

Los códigos de autenticación de mensajes (MAC) y las firmas proporcionan autenticación sobre todo el mensaje de control RPL ICMPv6 no seguro, incluida la sección de seguridad con todos los campos definidos, pero con la suma de comprobación ICMPv6 puesta a cero temporalmente [17][18]. El cifrado proporciona la confidencialidad del mensaje RPL ICMPv6 seguro comenzando por el primer byte después de la sección de seguridad y continuando hasta el último byte del paquete. La transformación de seguridad produce un mensaje ICMPv6 con la inclusión de los campos criptográficos (MAC, firma, etc.). En otras palabras, la propia transformación de seguridad (por ejemplo, la Firma y/o el Algoritmo en uso) detallará cómo incorporar los campos criptográficos en el paquete seguro.

Algoritmo de seguridad: El campo Algoritmo de seguridad especifica el esquema de cifrado, MAC y firma que la red utiliza . Los valores admitidos en este campo son los siguientes:

Algorithm	Encryption/MAC	Signature
0	CCM with AES-128	RSA with SHA-256
1-255		

Fig 5. Algoritmo de seguridad.

2.6.2Modo de funcionamiento criptográfico

El modo de funcionamiento criptográfico descrito en esta especificación (Algoritmo = 0) se basa en CCM y el cifrado en bloque AES-128 [RFC3610]. Este modo de operación es ampliamente soportado por las implementaciones existentes. El modo CCM requiere un nonce (CCM nonce).

Nonce CCM

Un nodo RPL construye un nonce CCM como sigue:

Identificador de origen: 8 bytes. El identificador de origen se establece como el identificador lógico del originador del paquete protegido.

Contador: 4 bytes. El contador se establece con el valor (sin comprimir) del campo correspondiente en la opción de seguridad del mensaje de control RPL.

Modo de identificador de clave (KIM): 2 bits. El KIM se ajusta al valor del campo correspondiente en la opción de seguridad del mensaje de control RPL.

Nivel de seguridad (LVL): 3 bits. El nivel de seguridad se ajusta al valor del campo correspondiente en la opción de seguridad del mensaje de control RPL.

Los bits no asignados del nonce de CCM están reservados. Deben ponerse a cero al construir el nonce CCM. Todos los campos del nonce de CCM se representan en el octeto más significativo y el bit más significativo en primer orden [RFC3610].

2.7 Subcapa IETF CoAP

El protocolo de aplicación restringido (Constrained Application Protocol, CoAP) [RFCs 7252, 7390, 7641, 7959, 8613] sigue un enfoque de mejor esfuerzo para mantener sincronizados el estado observado por un

cliente y el estado real del recurso en un servidor. Esto puede hacer que el cliente y el servidor no estén sincronizados en ocasiones. Dependiendo de la sensibilidad del recurso observado, operar en un estado antiguo podría ser una amenaza para la seguridad. Por lo tanto, el cliente debe tener cuidado de no utilizar una representación después de que su Max -Age expire, y el servidor debe establecer la opción Max-Age a un valor razonable.

Los recursos pueden ser observados a través de CoAP que está asegurado por la Seguridad de la Capa de Transporte de Datagramas (DTLS) utilizando cualquiera de los modos de seguridad. El uso de DTLS se indica mediante el esquema URI "coaps". Todas las notificaciones resultantes de una solicitud GET con una opción de observación deben ser devueltas dentro de la misma época de la misma conexión que la solicitud [RFCs 7252, 7390, 7641, 7959, 8613].

Hay que destacar que el estado cifrado es creado por el nodo emisor y descifrado por el mismo nodo al recibir una respuesta. La clave no se comparte con ningún otro sistema. Por lo tanto, la elección del esquema de cifrado y la generación de la clave para este sistema es una cuestión puramente local.

Cuando se utiliza el cifrado, se recomienda el uso de **AES-CCM** [RFC3610] con una etiqueta de 64 bits, combinada con un número de secuencia y una ventana de repetición [19]. Esta elección se basa en la aceleración de hardware disponible en muchos sistemas con restricciones. Si se dispone de un algoritmo diferente acelerado en el emisor, con una fuerza similar o mayor, entonces debería preferirse. Cuando la privacidad del estado no es requerida, y el cifrado no es necesaria, HMAC-SHA-256 [RFC6234], combinado con un número de secuencia y una ventana de repetición, puede ser utilizado.

Cuando se utiliza un modo de cifrado que depende de un nonce, como AES-CCM, el uso repetido del mismo nonce bajo la misma clave hace que el cifrado falle de forma catastrófica. Si un nonce se utiliza alguna vez para más de una operación de cifrado con la misma clave, entonces el mismo flujo de claves se utiliza para cifrar ambos textos planos, y las garantías de confidencialidad quedan anuladas. Los dispositivos con fuentes de entropía de baja calidad -como es el caso de los dispositivos con recursos limitados [RFC 7252].

2.7.2 DTLS

Durante la fase de aprovisionamiento, un dispositivo CoAP recibe la información de seguridad que necesita, incluyendo materiales de clave y listas de control de acceso. Al final de la fase de aprovisionamiento, el dispositivo estará en uno de los cuatro modos de seguridad con la siguiente información para el modo dado. Los modos NoSec y RawPublicKey son obligatorios para esta especificación.

NoSec: No hay seguridad a nivel de protocolo (DTLS está desactivado). Deberían utilizarse técnicas alternativas para proporcionar seguridad de capa inferior cuando sea apropiado. El uso de IPsec se discute en [IPsec-CoAP]. Algunas capas de enlace en uso con nodos restringidos también proporcionan seguridad en la capa de enlace, lo que puede ser apropiado con una gestión adecuada de las claves [20][21].

PreSharedKey: DTLS está habilitado, hay una lista de claves pre compartidas [RFC4279], y cada clave incluye una lista de los nodos con los que se puede utilizar para comunicarse. En el extremo, puede haber una clave por cada nodo con el que este nodo CoAP necesite comunicarse con él (proporción 1:1 de nodo/clave). Por el contrario, si más más de dos entidades comparten una clave específica pre compartida, esta clave sólo permite a las entidades autenticarse como miembros de ese grupo y no como un par específico.

RawPublicKey: DTLS está activado y el dispositivo tiene un par de claves asimétricas sin certificado (una clave pública en bruto) que se valida usando un mecanismo fuera de banda [RFC7250]. El dispositivo

también tiene una identidad calculada a partir de la clave pública y una lista de identidades de los nodos con los que puede comunicarse.

Certificado: DTLS está habilitado y el dispositivo tiene un par de claves asimétricas con un certificado X.509 [RFC5280] que lo vincula a su sujeto y está firmado por alguna raíz de confianza común.

En el modo "NoSec", el sistema simplemente envía los paquetes a través de UDP normal sobre IP y se indica con el esquema "coap" y el puerto CoAP por defecto. El sistema está asegurado únicamente impidiendo que los atacantes puedan enviar o recibir paquetes de la red con los nodos CoAP.

Hay que tener en cuenta la seguridad cuando se trata de protocolos IoT. Por ejemplo, CoAP utiliza UDP para transportar información. CoAP se basa en las características de seguridad de UDP para proteger la información. Así como HTTP utiliza TLS sobre TCP, CoAP utiliza Datagram TLS (DTLS) sobre UDP. DTLS soporta RSA, AES, etc. [RFC4279].

El mensaje CoAP más pequeño tiene una longitud de 4 bytes, si se omiten el token, las opciones y la carga útil. CoAP utiliza dos tipos de mensajes, peticiones y respuestas, utilizando un formato de cabecera base simple y binario. La cabecera base puede ir seguida de opciones en un formato optimizado de Tipo-Longitud-Valor. CoAP está vinculado por defecto a UDP y opcionalmente a DTLS, lo que proporciona un alto nivel de seguridad en las comunicaciones.

Cualquier byte después de las cabeceras en el paquete se considera el cuerpo del mensaje. La longitud del cuerpo del mensaje está implícita en la longitud del datagrama. Todo el mensaje debe caber en un solo datagrama cuando se vincula a UDP. Cuando se utiliza con 6LoWPAN, tal y como se define en el RFC 4944, los mensajes también deberían caber en una única trama IEEE 802.15.4 para minimizar la fragmentación [RFC4279].

2.8 PILA ZIGBEE SE

La seguridad es una preocupación importante en la arquitectura Zigbee. Aunque Zigbee utiliza los elementos de seguridad básicos en IEEE 802.15.4 (por ejemplo, estándar de cifrado avanzado (AES) y modos de seguridad Contador con CBC-MAC (CCM) [20][21]), amplía esto con:

• Algoritmos de cifrado AES de 128 bits

• Seguridad sólida, aprobada por el Instituto Nacional de Estándares y Tecnología (NIST) de EE. UU.

• Tipos de claves definidos (enlace, red)

• Configuración y mantenimiento de claves definidas

• Las claves se pueden conectar a una aplicación

• CCM * (modo de funcionamiento unificado / más sencillo)

• Centros de confianza

• Seguridad que se puede personalizar para la aplicación

2.8.1 Seguridad de la capa de red

Esta sección describe cómo Zigbee implementa la seguridad en la capa de red, que se aplica a la seguridad estándar. La seguridad de la red proporciona seguridad independientemente de las aplicaciones que puedan estar ejecutándose en un nodo Zigbee [23]. Todos los dispositivos certificados por Zigbee deben utilizar la seguridad de la capa de red. Proporciona el control de acceso básico para controlar qué nodos pueden participar en una red Zigbee en particular.

2.8.2 La clave de red

La seguridad de la red utiliza una clave para toda la red para el cifrado y el descifrado. Todos los dispositivos autorizados a unirse a la red tienen una copia de la clave y la utilizan para cifrar y descifrar todos los mensajes de la red. La clave de red también tiene un número de secuencia asociado para identificar una instancia particular de la clave [23] [24]. Cuando se actualiza la clave de red, el número de secuencia se incrementa para permitir que los dispositivos identifiquen qué instancia de la clave de red se ha utilizado para proteger los datos del paquete. El número de secuencia varía de 0 a 255. Cuando el número de secuencia llega a 255, vuelve a 0. Nota: Todas las claves Zigbee tienen una longitud de 128 bits. Todos los dispositivos que forman parte de una red Zigbee segura tienen una copia de la clave de red**.**

2.9 Capa Aplicación

Para garantizar el cumplimiento del producto final de ZSE, los dispositivos ZSE tienen una serie de requisitos de diseño especiales que incluyen los siguientes, que los hacen únicos de otros dispositivos Zigbee:

• Soporte para Chord-Based Key Establishment Schemes for Wireless Sensor Networks (CBKE) incluido el algoritmo ECC subyacente utilizado para la generación de claves [24].

• Compatibilidad con la autenticación de firma digital de curva elíptica (ECDSA), que es necesaria para validar los datos de imagen de firmware transferidos en el clúster de carga de arranque ZSE Over-the-Air (OTA).

• Certificados preinstalados emitidos por la autoridad de certificación correspondiente.

• Códigos de instalación preinstalados elegidos por el fabricante.

• Acceso al código de instalación e identificador único extendido (EUI649 la configuración de la red (para facilitar la comunicación fuera de banda de estos datos al centro de confianza).

• Los mensajes de datos APS para ciertos clústeres (la mayoría de los clústeres ZSE) requieren seguridad APS con una clave de enlace de aplicación (o centro de confianza).

El marco de la aplicación Zigbee garantiza que se cumplan los requisitos anteriores (cuando corresponda) cuando se utiliza la herramienta Simplicity Studio AppBuilder [] para configurar un dispositivo ZSE. Para obtener más información sobre el desarrollo basado en el marco de aplicación de Zigbee, consulte el documento UG391 [25] [26]: Guía del desarrollador del marco de aplicación de Zigbee

2.9.1Autenticación y cifrado

Zigbee usa una clave simétrica de 128 bits para cifrar todas las transmisiones en la capa de red usando AES-128. Los encabezados de red y auxiliares se envían de forma clara pero autenticada, mientras que la carga

útil de la red se autentica y cifra. AES-128 se usa para crear un hash de toda la porción de red del mensaje (encabezado y carga útil), que se agrega al final del mensaje [26].

El Código de integridad del mensaje (MIC) y se utiliza para autenticar el mensaje asegurándose de que no se haya modificado [27] [28]. Un dispositivo receptor procesa el mensaje y verifica el MIC calculado con el valor agregado al mensaje. Las alteraciones del mensaje invalidan el MIC y el nodo receptor descartará el mensaje por completo.

Nota: Zigbee usa un MIC de 4 bytes.

2.10 Conclusión del capítulo

Como resultado del estudio de ambas pilas protocolarias, se pudo identificar que algoritmos criptográficos simétricos y asimétricos se encuentran en cada una de las capas definidas en cada pila que a continuación se muestran en la fig.6 y fig.7. Los algoritmos marcados en rojo indican los algoritmos de llave asimétrica y que son vulnerables a ataques usando computadoras cuánticas.

Pila IoT – IETF

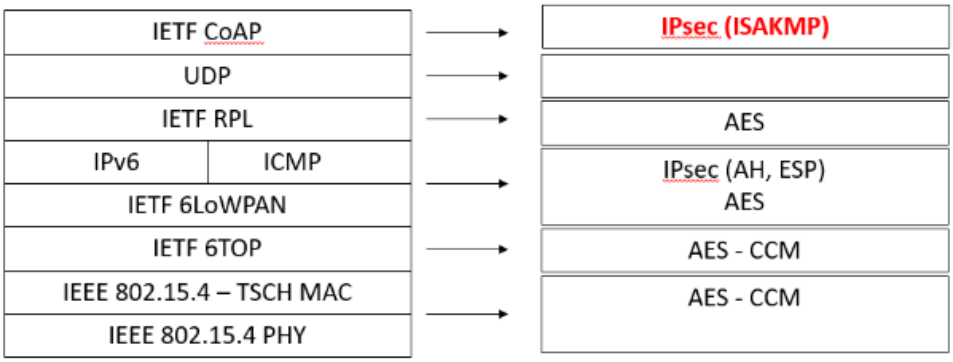


Fig 6. Algoritmos criptográficos usados en la pila protocolaria de IETF

Pila ZigBee

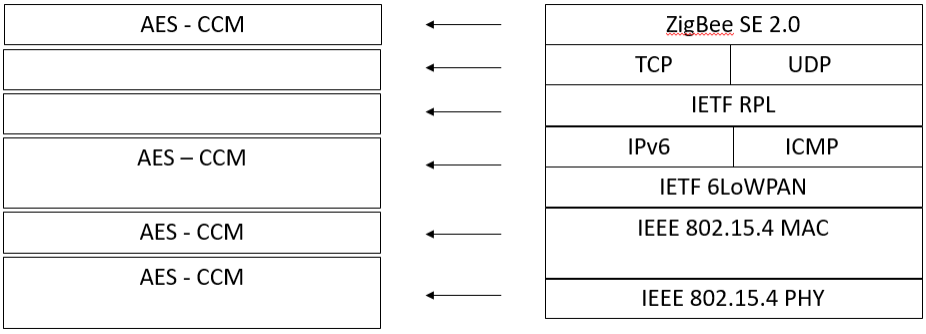


Fig 7 .Algoritmos criptográficos usados en la pila protocolaria de Zigbee.

Capítulo 3. Algoritmos Criptográficos

3.1 Criptografía post-cuántica

La historia de la criptografía post-cuántica ya está bien documentada en la literatura [27]; sin embargo, se introduce brevemente en esta sección, la cual se centra principalmente en la historia del desarrollo y las relaciones entre los algoritmos en lugar de las construcciones matemáticas. No es necesario tener conocimientos matemáticos para comprender la criptografía post-cuántica que se explora aquí, una comprensión básica de la criptografía es suficiente.
La computación cuántica desafía la línea divisoria entre problemas de computación tratables e intratables. Los ejemplos más significativos de esto son los algoritmos cuánticos eficientes para romper sistemas criptográficos que se cree que son seguros para las computadoras clásicas. En 1994, Shor encontró algoritmos cuánticos para la factorización y el registro discreto, y estos pueden usarse para romper el criptosistema RSA ampliamente utilizado y el intercambio de claves Diffie-Hellman usando una computadora cuántica. La pregunta más obvia que esto plantea es qué criptosistemas usar después de que se construyan las computadoras cuánticas. Una vez que se encuentre un buen sistema de reemplazo, aún habrá problemas con la logística de cambiar cada criptosistema en uso, y llevará tiempo hacerlo. Además, la información cifrada más sensible de la actualidad debería permanecer segura incluso después de que se construyan las computadoras cuánticas. Por lo tanto, estos datos ya deben estar encriptados con criptosistemas resistentes cuánticos. [28]

La criptografía clásica consta de problemas y herramientas que incluyen cifrado, distribución de claves, firmas digitales, generación de números pseudoaleatorios, pruebas de conocimiento cero y funciones unidireccionales.

El problema de la factorización

El problema de la factorización de n de la siguiente forma: Dado un entero positivo *n*, encontrar su factorización en números primos. Es decir, hallar una expresión tal que $n = p_1^{e1} p_2^{e2} \ldots p_k^{ek}$ donde *pi* son parejas de primos distintos y cada *ei* > 0.

Este problema es la base del algoritmo *RSA*, que permite cifrar/descifrar un mensaje. Su proceso, en términos generales, se describe a continuación. Se toman dos números primos *p* y *q* tan grandes como sea posible, y se calcula su producto *n* = *pq*. Sea *e* un número entero impar, y que no sea un factor del resultado del producto de $m = (p - 1)(q - 1)$. [28]

Finalmente, sea *d* un número tal que $e \cdot d \equiv 1 \bmod ´ m$. La llave pública se compone del par de números (*e*, *n*) y la llave privada por el par de números (*d*, *n*). El cifrado de un mensaje se obtiene mediante la expresión:

$$E(M) = M_e (\bmod n).$$

Resolver este problema mediante una computadora clásica resulta una tarea sumamente compleja, que podría tomar incluso años antes de ser completada. Sin embargo, una computadora cuántica tiene la capacidad de hacerlo en un tiempo aceptable mediante el algoritmo de Shor, el cual se centra en las potencias de un número al aplicarles la operación módulo y buscar predecir superíodo. [28]

El problema del logaritmo discreto

Sea *G* un grupo cíclico de orden *n*. Sea α un generador de *G* y sea β *2 G*. El logaritmo discreto de *beta* con base en α, denotado por $log\alpha\beta$, es un entero único *x*, $0 \leq x \leq n$ - 1, tal que $\beta = \alpha x$.

Luego, el problema del logaritmo discreto consiste en: Dado un número primo *p*, un generador α de $Z*p$, y un elemento $\beta \in Z*n$, encontrar un entero *x*, $0 \leq x \leq p$ - 2 tal que $a^x \equiv \beta(\text{mod } p)$.

La forma inmediata de resolver este problema consiste en una búsqueda exhaustiva, al calcular sucesivamente $\alpha 0, \alpha 1, \alpha 2, \ldots$, hasta obtener el valor β, lo que implicaría $O(n)$ multiplicaciones, donde *n* es del orden de α, por lo que resulta costoso e ineficiente para valores grandes de *n.* [28]

Las curvas elípticas son grupos, por lo que cumplen la propiedad de cerradura, asociatividad, tienen elemento de identidad y elemento inverso para cada elemento. En particular, el elemento de identidad es el punto al infinito, 0.

Criptografía de curvas elípticas

La seguridad de la criptografía de curvas elípticas se basa en el problema del logaritmo discreto generalizado, que se enuncia de la siguiente forma: Dado un grupo cíclico finito *G* de orden *n*, un generador α de *G*, y un elemento $\beta \in G$, encontrar el entero *x*, $0 \leq x \leq n$ - 1 tal que $a^x = \beta$. la llave privada es un entero que se elige al azar entre los valores 1 y el orden seleccionado.

El inverso del punto *P* es el punto simétrico respecto al eje *x*. La adición se define dados tres puntos colineales diferentes de cero que intersecten la curva, *P, Q, R*, entonces $P + Q + R = 0$. Si sólo hay dos puntos de intersección, es decir, una recta tangente es la que intersecta a la curva, entonces se dice que $P = Q$ y por tanto, $P + P = R$ o bien, $2P = R$, lo que se denomina *doblado de un punto*.

La llave pública se calculará por medio del producto escalar entre el entero que representa la llave privada y el punto generador *P*, lo que da como resultado un punto (*x, y*). Aunque resulta fácil calcular la llave pública a partir de la privada, hacerlo en sentido inverso es computacionalmente difícil, por lo que se considera una función de una sola vía. [28]

3.1.2 Concurso de estandarización NIST

El Proceso de estandarización de criptografía postcuántica del NIST es un concurso organizado por el NIST para actualizar sus estándares e incluir Criptografía postcuántica en dos categorías: para firma digital y para cifrado e intercambio de claves de mecanismo de encapsulación de clave pública (PKE/KEM). El proceso actualmente está en la tercera ronda y tiene se han elegido 7 algoritmos finalistas, 4 para PKE/KEM y 3 para firma digital. Además, se han elegido 8 candidatos de reserva, 5 para PKE/KEM y 3 para firma digital. De los 4 finalistas de mecanismo de encapsulación de clave pública, 3 son basados en retículos y 1 está basado en código. De los 3 finalistas de firma digital, 2 están basados en retículos y 1 en esquemas multivariable. En la fig.8 se muestra la clasificación de los algoritmos seleccionados para la tercera ronda. [29]

*Candidatos para la **tercera** ronda para KEM:*

- *Classic McEliece*
- *CRYSTALS–KYBER*
- *NTRU*
- *Saber*

Candidatos alternos:

- *BIKE*
- *FrodoKEM*
- *HQC*
- *NTRU Prime*
- *SIKE*

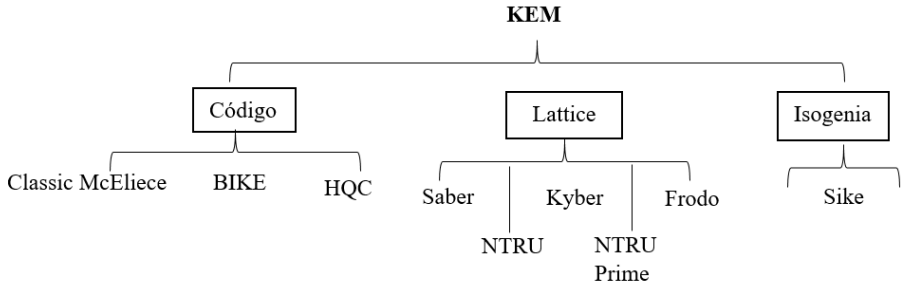


Fig 8. Alternativas de la Ronda 3 del NIST

La convocatoria de propuestas del NIST identificó tres aspectos generales de los criterios de evaluación que utilizarse para comparar algoritmos candidatos a lo largo del proceso de estandarización de NIST PQC: 1) seguridad, 2) costo y rendimiento, y 3) algoritmo y características de implementación.

Una de las decisiones difíciles que enfrentó el NIST fue decidir entre KYBER, NTRU y Saber. Los tres fueron seleccionados como finalistas y eran muy comparables entre sí. NIST confía en la seguridad que proporciona cada uno. La mayoría de las aplicaciones podrían usar cualquiera de ellos sin penalizaciones de rendimiento significativas. Como se indicó al final de la segunda ronda, el NIST tenía la intención de estandarizar solo uno de estos finalistas, ya que los tres se basaban en redes estructuradas. Los problemas relacionados con las patentes fueron un factor en la decisión del NIST durante la tercera ronda cuando el NIST se dio cuenta de varias patentes de terceros. [29]

El resto de candidatos KEM seleccionados (BIKE, Classic McEliece, HQC, SIKE) seguirán siendo evaluados en la cuarta ronda. Tanto BIKE como HQC se basan en códigos estructurados y serían adecuados como un KEM de uso general que no se basa en redes. NIST puede seleccionar como máximo uno de estos dos candidatos para la estandarización al finalizar la cuarta ronda. SIKE sigue siendo un candidato atractivo para la estandarización debido a sus pequeños tamaños de clave y texto cifrado. El NIST espera que continúen los estudios sobre SIKE durante la cuarta ronda. Classic McEliece fue finalista, pero el NIST no lo está estandarizando en este momento. Aunque se considera que es seguro, el NIST aún no prevé que se utilice mucho debido a su gran tamaño de clave pública. Por lo tanto, no hay urgencia para estandarizar ClassicMcEliece todavía. [29]

*Candidatos que avanzan a la **cuarta** ronda para KEM:*

- *BIKE*

BIKE (Bit Flipping Key Encapsulation) es un KEM basado en códigos binarios lineales cuasicíclicos de verificación de paridad de densidad moderada (QC-MDPC). El criptosistema BIKE se diseñó inicialmente para el uso de claves efímeras, pero ahora se afirma que también admite el uso de claves estáticas. BIKE tiene el rendimiento más competitivo entre los KEM sin celosía. El reciente y explícito reclamo de seguridad de IND-CCA por parte del equipo de BIKE es alentador. NIST anticipa que el tiempo adicional en la cuarta ronda permitirá una mayor investigación por parte de la comunidad de los reclamos de seguridad de BIKE.

NIST tiene la intención de seleccionar al menos un KEM adicional para la estandarización al final de la cuarta ronda. BIKE sigue bajo consideración debido a su rendimiento general y su suposición de seguridad sustancialmente diferente del KEM seleccionado actualmente.

- *ClassicMcEliece*

El NIST confía en la seguridad de Classic McEliece y se sentiría cómodo estandarizando los conjuntos de parámetros enviados (en algunos casos, bajo una fuerza de seguridad reclamada diferente). Sin embargo, no está claro si Classic McEliece representa la mejor opción para suficientes aplicaciones como para justificar su estandarización en este momento. Para sistemas de propósito general que deseen basar su seguridad en códigos en lugar de entramados, BIKE o HQC pueden representar una opción más atractiva. Para aplicaciones que necesitan un texto cifrado muy pequeño, SIKE puede resultar más atractivo. NIST, por lo tanto, considerará a Classic McEliece en la cuarta ronda junto con BIKE, HQC y SIKE. Al NIST le gustaría recibir comentarios sobre casos de uso específicos para los que Classic McEliece sería una buena solución.

- *HQC*

HQC ofrece sólidas garantías de seguridad y un análisis maduro de la tasa de fallas de descifrado. Aunque la estructura cuasicíclica de HQC permite tamaños razonables para las claves públicas y los textos cifrados, las claves públicas y los textos cifrados de HQC son más grandes que todos los demás KEM basados en código estructurado y celosía estructurada. El rendimiento general de HQC es aceptable, aunque no óptimo. NIST tiene la intención de seleccionar al menos un KEM adicional para la estandarización al final de la cuarta ronda. HQC sigue bajo consideración debido al riguroso análisis de seguridad y la suposición de seguridad sustancialmente diferente del KEM actualmente seleccionado.

- *SIKE*

El rendimiento de SIKE en dispositivos integrados puede ser un problema porque el tiempo para realizar una encapsulación/desencapsulación de una sola clave (en un procesador ARM de 32 bits de gama baja, por ejemplo) puede ser notable. La implementación de SIKE en FPGA puede ser una buena ruta para lograr un mejor rendimiento en dispositivos integrados [224, 225]. Además, puede resultar atractivo construir protocolos híbridos que utilicen SIKE junto con ECDH (intercambio de claves Diffie-Hellman de curva elíptica) seguro precuántico, ya que SIKE y ECDH pueden compartir algunas subrutinas comunes.

3.2 Criptografía basada en código

El principal problema de este cifrado, es que para generar una clave pública suficientemente segura para la criptografía postcuántica, tendría que estar por encima de 8 millones de bits. El mensaje cifrado, es el resultado de la multiplicación de esta matriz por el mensaje en claro. Aun así, el cifrado y el descifrado se consideran eficientes y la principal ventaja para aplicar criptografía basada en código es que para deshacer la multiplicación que genera la matriz (que necesitaría un potencial atacante para descubrir el mensaje en claro) no se conoce ningún método para deshacer estas matrices, con lo que parece viable usarlos de forma segura para cripto post-cuántica [29].

La criptografía basada en código se refiere a la criptografía que se basa en cualquier clase de código de corrección de errores. El cifrado funciona introduciendo errores en un mensaje y el descifrado recuperando el mensaje original. Creado en 1978 por Robert J. McEliece, el criptosistema McEliece fue la primera instancia de criptografía basada en código y fue el primer criptosistema resistente a los ataques cuánticos desarrollados. El criptosistema Niederreiter de 1986 era un criptosistema de clave pública tipo mochila que luego se demostró que tenía una prueba de seguridad tan estricta como McEliece. El beneficio de Niederreiter fue que se estimó que tenía claves más pequeñas que McEliece, aunque más tarde se demostró que tenía vulnerabilidades.

3.2.1 Criptografía basada en lattice

La criptografía post-cuántica basada en retículas basa su seguridad en la presunción de la dificultad que supone resolver diversos problemas definidos sobre éstas. Entre los principales problemas difíciles se encuentran el problema del vector más Corto, el Problema del vector más cercano, y el problema de los vectores independientes más cortos. Se considera que los tres problemas tienen el mismo grado de complejidad computacional.

Uno de los primeros criptosistemas basados en retículas que ha sido objeto de mucha investigación y criptoanálisis es el criptosistema GGH, que es muy similar a McEliece pero que utiliza retículas en lugar de códigos. NTRU es una instancia de GGH (y su variante HNF) que aplica una estructura algebraica adicional. La unidad de anillo polinomial truncado de grado N (NTRU) es la técnica utilizada por NTRU y NTRU Prime. Se basa en el cálculo limitado de polinomios restringidos sobre anillos polinomiales [29][30].

El aprendizaje con errores (LWE) es la base de la mayoría de los candidatos NIST basados en retículas. Se ha demostrado que LWE es al menos tan difícil como los problemas de retícula del peor de los casos (como el problema del vector más corto NP-hard) pero muy ineficiente.

El algoritmo Frodo es un ejemplo de algoritmo basado en LWE. Agregar las estructuras algebraicas adicionales, como se ve en otros algoritmos basados en Lattice, mejora la eficiencia tanto de la velocidad como del tamaño de la clave [31]. Sin embargo, al igual que la estructura adicional introducida en alguna criptografía basada en código, esto puede potencialmente introducir vulnerabilidades.

El algoritmo SABER utiliza el aprendizaje de módulos con redondeo (MLWR). A diferencia de LWE, que utiliza muestreo a partir de distribuciones de ruido, MLWR utiliza una "aleatoriedad" obtenida de forma determinista. Debido a esto, no es necesario incluir el ruido en el texto cifrado, lo que reduce el uso del ancho de banda [31].

El algoritmo Kyber utiliza el aprendizaje de módulos con errores (MLWE). Se ha demostrado que es equivalente en tiempo a MLWR. Debido a las similitudes entre estas dos opciones, logran parámetros, rendimiento y seguridad muy similares. Debido a esta similitud y su similitud con NTRU, que también es un esquema de retículas estructurado, como máximo, NIST estandarizará uno de los tres [31].

3.2.3 Criptografía basada en isogenia

La criptografía de isogenia utiliza curvas elípticas, pero a diferencia de la ECC tradicional (no post-cuántica), que utiliza logaritmos discretos, se utilizan curvas elípticas supersingulares. Algoritmo post cuántico cuyo objetivo es el intercambio de claves, y que busca sustituir a los algoritmos de Diffie-Hellman Exchange (DHE) y Elliptic Curve Diffie Hellman (ECDHE) en la era cuántica. Fue creado en 2011 por De Feo, Jao, and Plut. Debido a que funciona de forma similar a los mismos actualmente se considera una opción muy interesante [31].

Además de ser la base matemática menos estudiada, Isogenia también es la más nueva. También tiene una de las barreras de entrada más altas de los mecanismos de encapsulación de clave post-cuántica y sólo lo utiliza un candidato del NIST, el algoritmo SIKE.

3.3 Niveles de seguridad NIST

NIST está buscando uno o más algoritmos de cifrado de clave pública o de encapsulación de clave que sean seguros con respecto a la distinción en los ataques de texto cifrado elegido adaptativo (IND-CCA2 seguro). Esto significa que aunque varios algoritmos pueden alcanzar el mismo nivel de seguridad con respecto a un nivel dado del algoritmo AES, es posible que no brinden seguridad contra los mismos ataques.

NIST ha definido cinco niveles de seguridad, denominados niveles N1-N5 Un algoritmo debe ser al menos tan difícil de romper como un ataque de búsqueda de claves contra AES-128 para lograr la seguridad L1. La seguridad N2 es equivalente a la búsqueda de colisiones SHA3-256 (u otra función de hash criptográfica ideal). La seguridad N3 es equivalente a la búsqueda de claves AES-192. La seguridad L4 es equivalente a la búsqueda de colisiones SHA3-384. La seguridad L5 es equivalente a la búsqueda de claves AES-256.

Existe una cierta superposición entre estos niveles. Un ataque clásico de búsqueda de clave de fuerza bruta contra AES es O (2^n), donde *n* es el tamaño de la clave. Para realizar una búsqueda clásica de colisión de fuerza bruta contra SHA3, la complejidad de tiempo es O ($\sqrt{2^n}$), = O ($2^{n/2}$), donde *n* es la longitud del resumen. Esto significa que tanto AES-128 como SHA3-256 tienen una complejidad de tiempo clásica de O (2^{128}). Sin embargo, el NIST los considera diferentes niveles de seguridad [37].

Los requisitos de seguridad de todos los niveles se pueden ver en la Tabla 1

Nivel NIST	Clásica
AES-128 (N1)	128
SHA3-256 (N2)	128
AES-192 (N3)	192
SHA3-384 (N4)	192
AES-256 (N5)	256

Tabla 1. Seguridad clásica para cinco niveles NIST

3.3.1 Espacio

Hay un grado de variación en los tamaños de clave pública y texto cifrado de los candidatos NIST. El tamaño de la llave pública es el tamaño de la clave del cliente para realizar la encapsulación (para KEM antas de usar abreviaciones escríbelo completo) o para verificar firmas (para firmas digitales). Esta llave pública debe transferirse a través de la red para los mecanismos de encapsulación de claves efímeras, lo que agrega un costo de ancho de banda adicional. Para esquemas no efímeros, debe almacenarse en el cliente, lo que puede ser un factor limitante para dispositivos integrados específicos con almacenamiento flash limitado. Los tamaños de las llaves tanto públicas como privadas para cada algoritmo candidato se presentan en las tablas 2 y 3.

Candidatos Finalistas

Algoritmos		AES 128	SHA3 256	AES 192	SHA 384	AES 256
Classic McEliece	PK	678 B		926 B		1887 B
	SK	995 B		1056 B		2019 B
Kyber	PK	800 B		1184 B		1568 B
	SK	1632 B		2400 B		3168 B
NTRU	PK	699 B		931 B		1230 B
	SK	935 B		1235 B		1592 B
SABER	PK	672 B		992 B		1312 B
	SK	1568 B		2304 B		3040 B

Tabla2. Longitud en bytes de la Clave Pública (PK) y Clave Secreta (SK) reportada por los Autores.

Candidatos Finalistas Alternos

Algoritmos		AES 128	SHA3 256	AES 192	SHA 384	AES 256
BIKE	PK	12,323 bits		24,659 bits		40,973 bits
	SK	2,244 bits		2,246 bits		4.640 bits
FRODO KEM	PK	9616 B		15632 B		21520 B
	SK	19888 B		13294 B		43088 B
HQC	PK	2249 B		4522 B		7245 B
	SK	40 B		40 B		40 B
NTRU-PRIME	PK	1125 B		1463 B		2231 B
	SK	897 B		1184 B		1847 B
SIKE	PK	330 B	378 B	462 B		564 B
	SK	374 B	434 B	524 B		644 B

Tabla 3. Longitud en bytes (o bits) de la Clave Pública (PK) y Clave Secreta (SK) reportada por los Autores.

3.3.2 Fallos

Los algoritmos seguros, indistinguibilidad bajo el ataque de texto plano elegido (IND-CPA) con altas tasas de falla siguen siendo aceptables para KEM efímeros sin caché de claves. Las tasas de falla tan altas como 2-64 pueden ser aceptables para dispositivos integrados que generan relativamente pocas negociaciones de clave durante su vida, incluso cuando no se usan de manera efímera. Si bien esto no es útil para servidores grandes, permite utilizar en situaciones algoritmos más eficientes que podrían no tener pruebas de seguridad tan estrictas. Para los dispositivos integrados que pueden adaptarse a este caso de uso y no necesitan tanta seguridad, vale la pena considerar los algoritmos seguros IND-CPA eficientes con altas tasas de falla. [32]

Existe un pequeño peligro accidental por tener una tasa de fallas de este tipo, solo una pequeña posibilidad de que el intercambio de claves demore más. Sin embargo, ha habido ataques teorizados que implican forzar fallas. Los valores que hacen que falle el intercambio de claves dependen parcialmente de la clave secreta utilizada. Un atacante puede emitir muchas negociaciones de clave cuidadosamente diseñadas contra un servidor en el que está buscando qué parámetros hacen que la negociación falle. Conocer estos valores podría permitirles descubrir la clave privada del servidor, permitiéndoles hacerse pasar por el servidor. [32,33]

3.4 Evaluación usando ENCRYPT

Evaluaremos los datos usando ENCRYPT [34] Evaluación comparativa de sistemas criptográficos (eBACS).AMD64, también llamado Intel 64 o x86_64, fue seleccionado debido a su desempeño y ubicuidad. Es la arquitectura que se utiliza en la mayoría de los procesadores de escritorio de uso general, incluidas las series Xeon e i7 de Intel. Su inclusión de la aceleración de hardware AES en muchos modelos modernos es especialmente relevante para esta evaluación, ya que algunos algoritmos de llave asimétrica, como Kyber y Frodo, pueden usar AES para ciertas operaciones. AArch64, también llamado ARM64, es una extensión de 64 bits de ARM que utiliza el conjunto de instrucciones A64. Al igual que AMD64, agrega instrucciones para AES, así como para SHA-1, SHA-224 y SHA-256. Sus principales casos de uso incluyen dispositivos móviles. Se utiliza para la CPU Cortex-A.

En cada operación, para cada conjunto de parámetros, se utilizan los mejores tiempos estables recientes de cualquier máquina de la arquitectura respectiva. Todos los ciclos de reloj informados para la Tabla 4, por ejemplo, son los tiempos medios de la computadora con el tiempo medio de generación de claves más corto. Esto corresponde aproximadamente a la mejor mediana (razonable) para la operación dada en esa arquitectura dada. Cada conjunto de parámetros o incluso cada nivel de seguridad puede provenir de una computadora diferente. Todas las computadoras consideradas en esta evaluación tienen recursos aproximadamente comparables;

En las siguientes tablas 4 y 5 se muestran los resultados de la encapsulación en ciclos de reloj para AMD64, AArch64 y ARM32, respectivamente. En estas tablas se puede ver que mientras que para la encapsulación Kyber es el más rápido por un gran margen para la arquitectura AMD64 y el más rápido por un pequeño margen para la arquitectura ARM32, SABER supera a Kyber por un gran margen en AArch64. NTRU, Classic McEliece, HQC y BIKE son significativamente más lentos que Kyber y SABER en todas las arquitecturas, y Frodo y SIKE son increíblemente lentos.

ALGORITMO	NIST N1	NIST N2	NIST N3	NIST N4	NIST N5
Frodo	3174934	-	6416940	-	10484177
Kyber	29888	-	51297	-	72953
Saber	54787	-	107490	-	135090
NTRU	1130530	-	1970948	-	2945383
Classic McEliece	121892	-	1093795	-	1195623
HQC	1022648	-	1121130	-	1738750
BIKE	3544650	-	10926158	-	-
SIKE	172450573	-	-	-	-

Tabla 4 Tiempos de encapsulación de KEM en AMD64 en ciclos de reloj.

ALGORITMO	NIST N1	NIST N2	NIST N3	NIST N4	NIST N5
Frodo	12282758	-	27049733	-	48059878
Kyber	219346	-	349287	-	1120191
Saber	166660	-	295200	-	461262
NTRU	1724194	-	1136575	-	12443430
Classic McEliece	1227472	-	1797596	-	1957320
HQC	3161025	-	10017759	-	11675500
BIKE	-	-	-	-	-
SIKE	253270596	-	-	-	-

Tabla 5. Tiempos de encapsulación de KEM en AArch64 en ciclos de reloj.

ALGORITMO	NIST N1	NIST N2	NIST N3	NIST N4	NIST N5
Frodo	2968856	-	14443912	-	10315449
Kyber	21719	-	42679	-	59599
Saber	39780	-	116261	-	109957
NTRU	151722	-	12733562	-	14796922
Classic McEliece	147020528	-	118674450	-	1002321374
HQC	169152	-	338906	-	534928
BIKE	428543	-	1269728	-	-
SIKE	10725996	-	-	-	-

Tabla 6.Tiempos de generación de claves KEM en AMD64 en ciclos de reloj.

ALGORITMO	NIST N1	NIST N2	NIST N3	NIST N4	NIST N5
Frodo	10979270	-	23901203	-	42975638
Kyber	134282	-	240053	-	374061
Saber	100097	-	200100	-	335932
NTRU	10180300	-	17183700	-	24099950
Classic McEliece	1071215391	-	1590199275	-	11123074875
HQC	1183622	-	2026900	-	3615525
BIKE	-	-	-	-	-
SIKE	144228707	-	-	-	-

Tabla 7. Tiempos de generación de claves KEM en AArch64 en ciclos de reloj.

ALGORITMO	NIST N1	NIST N2	NIST N3	NIST N4	NIST N5
Frodo	134182311	-	176597519	-	125703807
Kyber	235857	-	1327819	-	1509019
Saber	190163	-	1118747	-	1553413
NTRU	14522613	-	122935366	-	128470576

Classic McEliece	1100117808	-	10198729429	-	12019682306
HQC	1306454	-	2875319	-	11117645
BIKE	-	-	-	-	-
SIKE	156082603	-	-	-	-

Tabla 8 Tiempos de generación de claves KEM en ARM32 en ciclos de reloj.

El algoritmo Saber funciona muy bien. Sus tiempos son generalmente competitivos con Kyber, a veces incluso lo superan. Sus tiempos de desencapsulación son muy buenos, sus tiempos de encapsulación son justos, sin embargo, sus tiempos de generación de claves son, con mucho, los más lentos de todos los algoritmos. Esto indica que McEliece es prometedor en entornos donde no tiene que regenerar claves constantemente, que es el caso de uso principal que NIST está considerando.

Kyber, que se basa en cierta aceleración de hardware y requiere 34 veces más ciclos de reloj para la encapsulación a 128 bits de seguridad. Para la misma operación con el mismo nivel de seguridad, McEliece tiene una desaceleración de 19 y Saber tiene una desaceleración de 25. NTRU tiene un factor de desaceleración de solo 2, por lo que si bien sigue siendo la opción más lenta para generación de llaves.

En ARM de 32 bits, NTRU es solo 8 veces más lento, mucho menos que la diferencia de 18 veces con la arquitectura AMD64. En un entorno TLS en este ARM, la brecha probablemente será aún más estrecha. Suponiendo que la brecha se vea afectada de manera similar a la instancia AMD64, NTRU podría ser casi tan rápido como Kyber, incluso a velocidades de red sin restricciones. Además, esto es con un uso efímero. Mirando sólo la generación de llaves, ARM NTRU es menos del doble de lento que ARM Kyber.

Al observar el tamaño, la velocidad y la seguridad de la clave pública, hay algunos claros favoritos. Sin embargo, diferentes algoritmos tienen diferentes pros y contras.

McEliece clásico

Pros: muchos conjuntos de parámetros pueden ser rápidos, bien estudiados, altamente seguros y el tamaño de texto cifrado muy pequeño significa que funcionará bien en entornos no efímeros con restricciones de ancho de banda. Debido en parte a su antigüedad y al gran volumen de investigación, los criptógrafos pueden confiar mucho en su seguridad. Sin tasa de fracaso.

Contras: claves públicas masivas, se necesita más trabajo para integrarlo en TLS. El alto consumo de memoria hace que no sea excelente para dispositivos restringidos o entornos con subprocesos.

Casos de uso: Criptografía de uso general en dispositivos sin requisitos estrictos de memoria.

Los textos cifrados pequeños significan que también funcionaría bien en redes limitadas para KEM no efímero/siempre que se reutilicen claves McEliece. Especialmente valioso cuando se requiere un alto grado de confianza en el criptosistema debido a las estrictas pruebas de seguridad y madurez de McEliece.

Kyber

Pros: Muy rápido en máquinas x86_64. Claves y textos cifrados bastante compactos

Contras: Tasas de falla mayores a las preferidas. Las claves públicas y los textos cifrados más pequeños se pueden encontrar en otros lugares.

Casos de uso: logra algunos de los mejores rendimientos siempre que no se utilice en un entorno muy restringido. Gran opción para criptografía eficiente de propósito general.

HQC

Pros: la alternativa NIST más rápida, competitiva con los finalistas; detrás de Kyber y Saber y por encima de NTRU y McEliece

Contras: Claves públicas bastante grandes. Textos cifrados muy grandes.

Casos de uso: criptografía de propósito general cuando Kyber o Saber no son viables y las redes no están restringidas.

BIKE

Pros: La encapsulación rápida significa que la mayor parte del trabajo lo realiza el servidor

Desventajas: Desencapsulación bastante lenta

Casos de uso: Bueno para clientes limitados, aunque incluso así es superado por otras opciones.

Frodo

Pros: Seguro, resistente al aumento de fallas.

Contras: claves grandes, textos cifrados grandes y rendimiento lento

Casos de uso: el rendimiento no es relevante, pero la confianza en la seguridad sí lo es.

Saber

Pros: Rápido, incluso en hardware limitado.

Desventajas: Tasas de falla bastante altas, aunque se cree que es resistente al aumento de fallas.

Casos de Uso: Criptografía de propósito general. Logra un rendimiento comparable al de Kyber y lo supera ligeramente en entornos restringidos

NTRU

Pros: Bien estudiado, seguro. Sin tasa de fracaso. Se desempeña bien en ciertas situaciones.

Contras: Más lento que la mayoría de las alternativas pero no prohibitivamente lento

Casos de uso: como McEliece, bueno cuando la confianza en la seguridad es valiosa. A diferencia de McEliece, NTRU es más lento y tiene textos cifrados más grandes, pero tiene claves públicas mucho más pequeñas y consume menos memoria, lo que lo hace más equilibrado. Buena opción cuando la confianza en la seguridad es valiosa pero McEliece no es viable. Bueno para entornos limitados.

SIKE

Pros: Teclas increíblemente compactas. Sin tasa de fracaso por naturaleza. Posibles aceleraciones son posibles. Textos cifrados increíblemente compactos.

Contras: muy lento en comparación con los esquemas basados en código y en retícula (más de un orden de magnitud). Se cree que esto se debe en parte a que Isogeny no está tan bien desarrollado para la criptografía.

Casos de uso: cuando el almacenamiento/memoria flash y el ancho de banda son muy limitados y la velocidad de encapsulación/desencapsulación es aceptable. También puede ser más fácil de desarrollar para los criptógrafos existentes, ya que utiliza principios ya vistos en la criptografía de curva elíptica existente.

Capítulo 4. Análisis de Rendimiento

Para este último capítulo, realizaremos el análisis de los algoritmos post-cuánticos para IoT con un procesador pequeño, en este caso usaremos ARDUINO. La placa tiene CP2102 USB estable a chips TTL, para garantizar la comunicación normal del puerto serie. Es compatible con la descarga automática, sin necesidad de cambiar manualmente entre modo de descarga y ejecución. Es compatible con el desarrollo en Windows (entorno de simulación cygwin y msys32) y el sistema Linux. ESP32-D0WDQ6 tiene dos MCU LX6 de 32 bits de baja potencia incorporadas que es compatible con el estándar requerido para verificar el rendimiento de cada uno de estos algoritmos para KEM. Este módulo tiene un chip ESP32-D0WDQ6, que es extensible y adaptable. Dos núcleos de CPU pueden ser controlados individualmente o alimentados. El rango de ajuste de la frecuencia del reloj es de 80 MHz a 240 MHz.

Github

GitHub es una página que ofrece un grupo de servicios que facilitan el uso de Git, como por ejemplo hosting de proyectos, facilidades de colaboración, reviews de código, perfiles personales, pull requests, issues, etc. Es la plataforma de «hosting» de los proyectos. Una comunidad de personas que desarrollan y comparten, usando GIT. En definitiva, Github es un sitio web pensado para hacer posible el compartir el código de una manera más fácil y al mismo tiempo darle popularidad a la herramienta de control de versiones en sí, que es Git.

4.1 Análisis de Rendimiento Arduino

Para este análisis, sólo se tomara en cuenta el nivel más bajo de cada uno de estos algoritmos. Al trabajar con Arduino es importante que conozcamos que es github y cómo hacer ciertas operaciones sencillas. Para este análisis usamos las librerías de los algoritmos que ya están previamente en github (https://github.com/mupq/pqm4/tree/master/crypto_kem) (pqm4 es la biblioteca criptográfica poscuántica para ARM Cortex-M4) para poder “ligarlo” a la tarjeta Arduino. (Fig.9)

Biblioteca pqm4

Los objetivos de diseño de la biblioteca son ofrecer pruebas funcionales automatizadas en una placa de desarrollo ampliamente disponible; generación automatizada de vectores de prueba y comparación con la salida de una implementación de referencia que se ejecuta en el lado del host (es decir, en la computadora a la que está conectada la placa de desarrollo); evaluación comparativa automatizada de la velocidad, el uso de la pila y el tamaño del código; perfiles automatizados de ciclos gastados en primitivas simétricas (SHA-2, SHA-3, AES); integración de implementaciones limpias de PQClean; y fácil integración de nuevos esquemas e implementaciones en el marco.

```
#include <crypto_kem>
void setup() {

int crypto_kem_keypair(unsigned char *pk, unsigned char *sk);
int crypto_kem_enc(unsigned char *ct, unsigned char *ss, const unsigned char *pk);
int crypto_kem_dec(unsigned char *ss, const unsigned char *ct, const unsigned char *sk);
  }

void loop() { }
```

Fig 9. Llamada de la librería de github

Primero es necesario copiar el ***sketch*** a la carpeta del proyecto. Si ya tenemos *git*, para instalar la librería basta con abrir un terminal, ir a la carpeta ~/sketchbook/libraries/ y hacer un clone de la *URL* del proyecto que puedes encontrar justo encima del botón de descargar ZIP. En esta ocasión no habrá que renombrar nada ya que *GitHub* no modificará el nombre de la carpeta del proyecto como en el caso de la instalación manual del *ZIP*. En definitiva ya estaría instalada la librería y la veríamos en nuestro *sketchbook* al abrir *Arduino IDE*.

Afortunadamente, el **microcontrolador** de las **placas Arduino** ya cuenta con un módulo que es capaz de realizar esto por sí solo. Estos módulos son denominados **contadores** o **temporizadores** y su tarea es muy simple: mantener un registro con los pulsos generados por el **hardware clock**. Todo esto se encuentra en el archivo de código del core de Arduino que está en la carpeta de Windows

C:/Program Files(x86)/Arduino/hardware/arduino/avr/cores/arduino/wiring.c

Cronometrar tiempo en Arduino

```
void wait(unsigned long milliseconds) {
  unsigned long timeout = millis() + milliseconds;
  while (millis() < timeout) {
    yield();
  }

  uint16_t msgLen = sizeof(readBuffer);
  memcpy(enc_iv, enc_iv_to, sizeof(enc_iv_to));
  uint16_t encLen = encrypt_to_ciphertext((char*)cleartext, msgLen, enc_iv);

  Serial.println((char*)ciphertext);
   Serial.println("Encrypted. KEM...");   memcpy(enc_iv, enc_iv_from, sizeof(enc_iv_from));
  uint16_t decLen = decrypt_to_cleartext(ciphertext, encLen , enc_iv);

  Serial.print("Decrypted cleartext:\n"); Serial.println((char*)cleartext);

  if (strcmp((char*)readBuffer, (char*)cleartext) == 0) {
    Serial.println("Generate KEM correctly.");
  } else {
    Serial.println("KEM test failed.");
  }

  Serial.println("---");
```

Fig.10 Función para medir el tiempo del algoritmo KEM

```
unsigned long us, ms, ck;
unsigned long _us, _ms, _ck;
unsigned long __us, __ms, __ck;

void setup()
{ Serial.begin(9600);
}
boolean firstloop=1;
void loop() {
currentMicros = micros(); // capture current time
elapsedMicros = currentMicros - previousMicros; // see how much time has passed
  if (elapsedMicros >= oneHundredth){ // 0.01S passed, so:
  previousMicros = previousMicros + oneHundredth; // set up for next time interval
  hundredths = hundredths +1; // increment the 0.01S counter
    if (hundredths == 50){ // 0.5S passed?
    hundredths = 0; // reset the counter
    PIND = PIND | 0b00100000; // toggle LED pin by writing 1 to Input register
    } // 0.5S check
  }
} // end loop


if (firstloop) {
  Serial.print("clocks for 1us:");
  ck=microsecondsToClockCycles(1);
  Serial.println(ck,DEC); firstloop--;
  Serial.println("runtime us, ms, ck :: elapsed tme us, ms ck"); }
  ck=microsecondsToClockCycles(us);
  Serial.print(us,DEC);
  Serial.print("\t");
  Serial.print(ms,DEC);
  Serial.print("\t"); Serial.print(ck,DEC);
  Serial.print("\t::\t"); __us = us - _us; __ms = ms - _ms; __ck = ck - _ck;
  Serial.print(__us,DEC); Serial.print("\t");
  Serial.print(__ms,DEC); Serial.print("\t");
  Serial.println(__ck,DEC); }
```

Fig.11 Función para mostrar en pantalla el tiempo de cada uno de los algoritmos

```
 bool testCipher_N(AuthenticatedCipher *cipher, const struct TestVector *test, size_t inc)
 {
     size_t posn, len;
     uint8_t tag[16];

     crypto_feed_watchdog();

     cipher->clear();
     if (!cipher->setKey(test->key, cipher->keySize())) {
         Serial.print("setKey ");
         return false;
     }
     if (!cipher->setIV(test->iv, test->ivsize)) {
         Serial.print("setIV ");
         return false;
     }

     memset(buffer, 0xBA, sizeof(buffer));

     for (posn = 0; posn < test->authsize; posn += inc) {
         len = test->authsize - posn;
         if (len > inc)
             len = inc;
         cipher->addAuthData(test->authdata + posn, len);
     }

     for (posn = 0; posn < test->datasize; posn += inc) {
         len = test->datasize - posn;
         if (len > inc)
             len = inc;
         cipher->encrypt(buffer + posn, test->plaintext + posn, len);
     }
```

Fig.12 Función para generación de llaves

En la siguiente tabla, se muestran los resultados obtenidos del total de ciclos de reloj para cada uno de los algoritmos seleccionados para el nivel más bajo.

ALGORITMO	NIST N1
Frodo	10826540
Kyber	753621
Saber	9716250
NTRU	1827690
Classic McEliece	887261
HQC	998162
BIKE	459281
SIKE	118258

Tabla 10 Tiempos de generación de claves KEM en ARDUINO en ciclos de reloj.

De acuerdo a los resultados, podemos darnos cuenta de que algoritmos como, Bike y Sike, son los que menos ciclos de reloj tardan al terminar el algoritmo. Por el contrario los algoritmos como Frodo y Saber, son los dos algoritmos con mayor tiempo al terminar su ejecución.

NIST celebró la tercera Conferencia de estandarización de NIST PQC del 7 al 9 de junio de 2021 para discutir varios aspectos de estos candidatos y obtener comentarios valiosos para la(s) selección(es) final(es). Ahora ya tiendo los resultados que se obtuvo en ARDUINO, haremos una comparación de los mismos con los resultados de algunas evaluaciones de algunos autores que trabajaron con dichos algoritmos en microcontrolador CORTEX M4 en el nivel 1 [40]. En la siguiente tabla, podemos observar que los resultados en M4 soy demasiado grandes para el Internet de las cosas, ya que el objetivo de este análisis es trabajar con dispositivos pequeños.

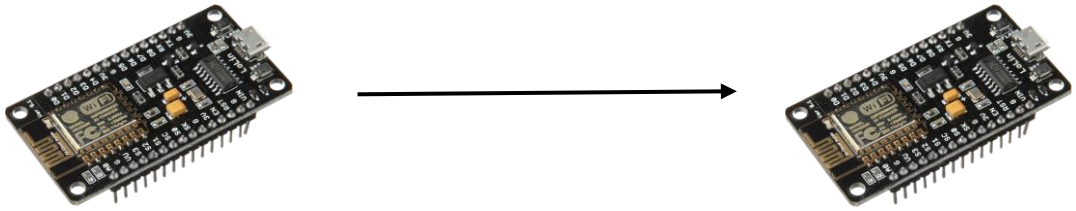
ALGORITMO	NIST N1
Frodo	349 123 987
Kyber	2 520 913
Saber	2 833 348
NTRU	1 236 867
Classic McEliece	430 811 294
HQC	x
BIKE	3 489 412
SIKE	489 378 234

Tabla 10 Tiempos de generación de claves KEM en CORTEX M4 en ciclos de reloj.

Tomaremos en cuenta los dos algoritmos cuyo resultado fue el más bajo de todos los demás, Saber y Kyber, y haremos el mismo procedimiento para ambos calculando el tiempo en que tarda de un dispositivo a otro. Para eso, se utilizara dos tarjetas Arduino. A continuación en las tablas 11 y 12 se muestran los resultados obtenidos en cada uno de los experimentos.

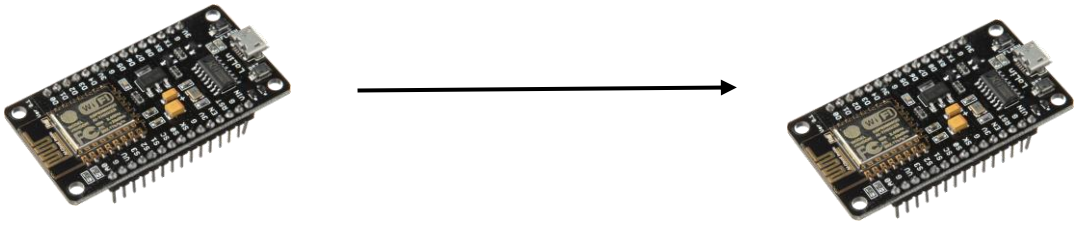
Algoritmo BIKE

459281 ciclos de reloj	665141 ciclos de reloj



Algoritmo SIKE

118258 ciclos de reloj	13001 ciclos de reloj



Capítulo 5. Conclusiones

Se hizo un amplio estudio acerca de las pilas protocolarias en el IoT para comprender mejor el cómo están conformados, los servicios de cada una de las capas y que protocolos de seguridad las conformas y con esto, identificar si existe o no algoritmos criptográficos de llave asimétrica ya que se han establecido los peligros de las computadoras cuánticas, que suponen una amenaza para el cifrado de llave asimétrica, que es vital para una Internet segura, y se han desarrollado algoritmos resistentes a la computación cuántica.

Una vez que se tuvo en claro que algoritmos criptográficos están dentro de las pilas protocolarias del IoT, se investigó acerca del concurso de NIST, en la búsqueda de nuevos algoritmos criptográficos post cuánticos (en este caso para KEM). Se evaluaron todos los candidatos, leyendo cada uno de los documentos que están disponibles en la página oficial del NIST, para poder tener en cuenta la generación de llaves de cada uno de estos algoritmos candidatos y con esto, comprender cuales de esos algoritmos puedan satisfacer los requerimientos y las restricciones para el IoT.

Una vez teniendo en claro cuáles fueron los candidatos finales y alternos para KEM, procedimos a evaluar cada uno de estos usando la plataforma para algoritmos criptográficos ENCRYP y fue posible identificar, con bastante confianza, algoritmos con un rendimiento y una seguridad razonablemente aceptables. Uno de los objetivos de esta tesis fue la evaluación de esos algoritmos criptográficos en dispositivo

pequeños/limitados ya que en las evaluaciones del NIST, todos los algoritmos solo han sido probados en computadoras con procesadores amplios.

Como dispositivo limitado/pequeño, se eligió la tarjeta ARDUINO esp32, modulo compatible para el IoT y el software ARDUINO IDE. Para la evaluación de cada uno de los algoritmos para conocer sus tiempos de reloj en este tipo de dispositivos, usamos las bibliotecas de pqm4 que podemos encontrar en github. Descargamos esas librerías en nuestra computadora, identificamos los archivos de las librerías y las subimos en el software Arduino ide. Pudimos darnos cuenta en los tiempos de cada uno de los algoritmos criptográficos en un dispositivo pequeño, en las que hay una gran diferencia en números comparado con una computadora y procesador grande.

Algunos de los resultados de los algoritmos si coincide con los algoritmos seleccionados en la ronda 3, así como los alternos. Y aún queda por esperar cuáles serán los próximos candidatos que pasaran a la ronda 4 del concurso NIST.

Esta investigación se podría mejorar en la búsqueda de cómo podríamos evaluar esos algoritmos en algún otro dispositivo móvil del IoT como algún teléfono móvil, y poder observar si los resultados obtenidos en el Arduino lleguen a coincidir con los resultados que obtendríamos en un celular. Todavía hay mucha evaluación que se debe realizar en los próximos años. Se deben considerar más casos de uso. Específicamente, mirando diferentes opciones de hardware y red. Incluso más allá de la arquitectura, la cantidad de recursos informáticos disponibles para el cliente y el servidor puede marcar la diferencia no solo en las velocidades sino también en qué anchos de banda el tamaño del texto cifrado se convierte en un factor limitante. Finalmente, siempre vale la pena explorar los esquemas de control de acceso adicionales y las optimizaciones generales de los algoritmos. Independientemente de las formas que tome el trabajo futuro, es valioso considerar la imagen completa en el análisis y evitar la visión de túnel, como se ha demostrado en esta tesis.

BIBLIOGRAFÍA

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2401] Kent, S. and R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

[RFC2402] Kent, S. and R. Atkinson, "IP Authentication Header", RFC 2402, November 1998.

[RFC2406] Kent, S. and R. Atkinson, "IP Encapsulating Security Payload (ESP)", RFC 2406, November 1998

[RFC2407] Piper, D., "The Internet IP Security Domain of Interpretation for ISAKMP", RFC 2407, November 1998.

[RFC2410] Glenn, R. and S. Kent, "The NULL Encryption Algorithm and Its Use With IPsec", RFC 2410, November 1998.

[RFC2412] Orman, H., "The OAKLEY Key Determination Protocol", RFC 2412, November 1998.

[RFC2451] Pereira, R. and R. Adams, "The ESP CBC-Mode Cipher Algorithms", RFC 2451, November 1998

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC2464] Crawford, M., "Transmission of IPv6 Packets over Ethernet Networks", RFC 2464.

[RFC2521] Karn, P. and W. Simpson, "ICMP Security Failures Messages", RFC 2521, March 1999.

[RFC3447] Jonsson, J. and B. Kaliski, "Public-Key Cryptography tandards (PKCS) RSA Cryptography Specifications. Version 2.1", RFC 3447, February 2003.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291.February 2006.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

[RFC4443] Conta, A., Deering, S., and M. Gupta, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, March 2006.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861.

[RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, September 2007.

[RFC5480] Turner, S., Brown, D., Yiu, K., Housley, R., and T. Polk, "Elliptic Curve Cryptography Subject Public Key Information", RFC 5480, March 2009.

[RFC6066] Eastlake, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, January 2011.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

[RFC6690] Shelby, Z., "Constrained RESTful Environments (CoRE) Link Format", RFC 6690, August 2012.

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980

[1] K. S. Roy and H. K. Kalita, “A Survey on Post-Quantum Cryptography for Constrained Devices,” *International Journal of Applied Engineering Research*, vol. 14, no. 11, pp. 2608–2615, 2019.

[2] J. Norman. Formulation of Shor's Algorithm for Quantum Computers, Apr. 30, 2020, Accessed on: June. 13, 2020. [Online]. Available: http://www.historyofinformation.com/detail.php?id=3877

[3] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz, "Comparing Elliptic Curve Cryptography and RSA on 8-Bit CPUs," in *International workshop on cryptographic hardware and embedded systems*, pp. 119–132, Springer, 2004.

[4] H. Nejatollahi, N. Dutt, S. Ray, F. Regazzoni, I. Banerjee, and R. Cammarota, "Post-Quantum Lattice-Based Cryptography Implementations: A Survey," *ACM Computing Surveys (CSUR)*, vol. 51, no. 6, p. 129, 2019.

[5] L. K. Grover, "A Fast Quantum Mechanical Algorithm for Database Search," in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pp. 212–219, 1996.

[6] G. Brassard, P. Hoyer, and A. Tapp, "Quantum Algorithm for the Collision Problem," *arXiv preprint quant-ph/9705002*, 1997.

[7] L. Chen, L. Chen, S. Jordan, Y.-K. Liu, D. Moody, R. Peralta, R. Perlner, and D. Smith-Tone, *Report on post-quantum cryptography*, vol. 12. US Department of Commerce, National Institute of Standards and Technology, 2016.

[8] M. Campagna, L. Chen, Ö. Dagdelen, J. Ding, J. K. Fernick, N. Gisin, D. Hayford, T. Jennewein, N. Lütkenhaus, M. Mosca, *et al.*, "Quantum safe cryptography and security," *ETSI White Paper*, vol. 8, 2015.

[9] D. Ott, C. Peikert, *et al.*, "Identifying Research Challenges in Post Quantum Cryptography Migration and Cryptographic Agility," *arXiv preprint arXiv:1909.07353*, 2019.

[10] E. Fujisaki and T. Okamoto, "Secure integration of asymmetric and symmetric encryption schemes," in *Annual International Cryptology Conference*, pp. 537– 554, Springer, 1999.

[11] A. J. Menezes, V. O. P. C., and S. A. Vanstone, *Handbook of Applied Cryptography*. CRC Press, 2001.

[12] J. Jonsson, "An oaep variant with a tight security proof," 2002.

[13]T. Okamoto, "Authenticated key exchange and key encapsulation in the standard model," in *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 474–484, Springer, 2007.

[14] D. Aggarwal, G. Brennen, T. Lee, M. Santha, and M. Tomamichel, "Quantum attacks on bitcoin, and how to protect against them, 2017," *arXiv preprint arXiv:1710.10377*.

[15] "Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process," June 2020.

[16] "PQC Standardization Process: Third Round Candidate Announcement ," July 2020.

[17] C. Chen, O. Danba, J. Hoffstein, A. Hulsing, J. Rijneveld, J. M. Schanck, P. Schwabe, W. Whyte, and Z. Zhang, "NTRU: Algorithm Specifications and Supporting Documentation (2019)," vol. 1.

[18] D. J. Bernstein, C. Chuengsatiansup, T. Lange, and C. van Vredendaal, “Round 2 (2019),” vol. 1, pp. 7–3.

[19] “Cybersecurity Vulnerabilities Affecting Medtronic Implantable Cardiac Devices, Programmers, and Home Monitors: FDA Safety Communication,” Mar 2019.

[20] S. Gray, “Always on: privacy implications of microphone-enabled devices,” in *Future of privacy forum*, 2016.

[21] L. Wu, X. Du, M. Guizani, and A. Mohamed, “Access Control Schemes for Implantable Medical Devices: A Survey,” *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1272–1283, 2017.

[22] Food, D. Administration, *et al.*, “Postmarket management of cybersecurity in medical devices,” *Silver Spring: Food and Drug Administration*, 2016.

[23] Food, D. Administration, *et al.*, “Content of premarket submissions for management of cybersecurity in medical devices,” 2019.

[24] Z. B. Celik, E. Fernandes, E. Pauley, G. Tan, and P. McDaniel, “Program analysis of commodity iot applications for security and privacy: Challenges and opportunities,” *ACM Computing Surveys (CSUR)*, vol. 52, no. 4, pp. 1–30,2019.

[25] C. W. Axelrod, “Managing the risks of cyber-physical systems,” in *2013 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*, pp. 1–6, IEEE, 2013.

[26] M. Fagan, K. N. Megas, K. Scarfone, and M. Smith, “Iot device cybersecurity capability core baseline,” tech. rep., National Institute of Standards and Technology, 2020.

[27] Bernstein, D.J., Buchmann, J., Dahmen, E.: Post Quantum Cryptography. Springer Publishing Company, Incorporated, 1st edn. (2008)

[28] Academia mexicana de computación: Introducción a la Ciberseguridad y sus aplicaciones en México (2020)

[29]Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Processhttps://nvlpubs.nist.gov/nistpubs/ir/2022/NIST.IR.8413.pdf

[30] U. Banerjee, A. Pathak, and A. P. Chandrakasan, “2.3 an energy-efficient configurable lattice cryptography processor for the quantum-secure internet of things,” in *2019 IEEE International Solid-State Circuits Conference-(ISSCC)*, pp. 46–48, IEEE, 2019.

[31] K. Basu, D. Soni, M. Nabeel, and R. Karri, “Nist post-quantum cryptography-a hardware evaluation study.,” *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 47, 2019.

[32] E. Alkim, R. Avanzi, J. Bos, L. Ducas, A. de la Piedra, T. Poppelmann, P. Schwabe, D. Stebila, M. R. Albrecht, E. Orsini, *et al.*, “NewHope: Algorithm Specifications and Supporting Documentation (2020),” vol. 1, no. 7.5, 2020.

[33] R. Overbeck and N. Sendrier, “Code-Based Cryptography,” in *Post-quantum cryptography*, pp. 95–145, Springer, 2009.

[34] eBACS: ECRYPT Benchmarking of Cryptographic Systems https://bench.cr.yp.to/

[35] N. Aragon, P. Barreto, S. Bettaieb, L. Bidoux, O. Blazy, J.-C. Deneuville, P. Gaborit, S. Gueron, T. Guneysu, C. A. Melchor, *et al.*, "BIKE: Bit Flipping Key Encapsulation," 2017.

[36] M. Baldi, A. Barenghi, F. Chiaraluce, G. Pelosi, and P. Santini, "LEDAcrypt: LowDensity Parity-Check Code-Based Cryptographic Systems," *NIST round*, vol. 2, 2019.

[37] C. A. Melchor, N. Aragon, S. Bettaieb, L. Bidoux, O. Blazy, J.-C. Deneuville, P. Gaborit, E. Persichetti, G. Zémor, and I.-C. Bourges, "Hamming quasi-cyclic (hqc)," *NIST PQC Round*, vol. 2, pp. 4–13, 2018.

[38] C. A. Melchor, N. Aragon, M. Bardet, S. Bettaieb, L. Bidoux, O. Blazy, J.-C. Deneuville, P. Gaborit, A. Hauteville, A. Otmani, O. Ruatta, J.-P. Tillich, and G. Zémor, "ROLLO-Rank-Ouroboros, LAKE & LOCKER," 2019.

[39] NIST https://csrc.nist.gov/

[40] NIST https://csrc.nist.gov/events/2021/third-pqc-standardization-confe