Arquitectura de seguridad para un sistema ciberfísico de adquisición, procesamiento y análisis de datos médicos de las tripulaciones que realizan entrenamiento en cámara de altura

Maestría en Seguridad de la Información
Universidad de los Andes
Bogotá D.C, Colombia

Jennifer Aguirre Velandia
j.aguirrev@uniandes.edu.co

Sergio Baudin Cruz
sb.cruz@uniandes.edu.co

***Resumen*—El siguiente documento desarrolla la propuesta de seguridad para un sistema IoT, que se encarga de sensar datos médicos de tripulantes de vuelo durante los entrenamientos de cámara de altura en la Fuerza Aeroespacial Colombiana. En un principio, el sistema ciberfísico de adquisición, procesamiento y almacenamiento de datos médicos para la cámara de altura de la Fuerza Aeroespacial Colombiana, no se contempló con los controles de seguridad necesarios para su puesta en marcha, de modo que en su diseño y desarrollo no se tomaron en cuenta las medidas de seguridad para garantizar la confidencialidad, integridad, disponibilidad y privacidad de los datos privados o semiprivados de los tripulantes.**

De lo anterior, se pudo observar que el sistema era propenso a manipulación, indisponibilidad y fallos en la veracidad de la información sensada que luego era mostrada al personal médico; y quienes, a parir de su análisis, tomarían decisiones sobre las situaciones de hipoxia en los tripulantes en la cámara.

Así y con el fin de cumplir lo establecido en la Ley 1581 del 2012 de Protección de Datos Personales, se hace necesario establecer la arquitectura de seguridad para este sistema ciberfísico antes de implementarlo en la cámara de altura con tripulantes a bordo. Se establecieron y diseñaron los controles necesarios mitigar los riesgos de la información en tránsito, almacenamiento y procesamiento desde el momento de su adquisición por los sensores, envió hacia un centralizador que los procesa y habilita su consumo a través de un servicio web y finalmente, hace su almacenamiento seguro en la nube.

Keywords*— *ASCON, aviación, cámara de altura, confidencialidad, criptografía ligera, disponibilidad, hipobárica, hipoxia, integridad, IoT, tripulaciones de vuelo, protección de datos, privacidad, seguridad, sensado.

I. Contexto y Problema

En Colombia se han realizado entrenamientos de cámara de altura desde el año 2010, donde se ha entrenado no solo tripulantes de la Fuerza Pública sino también personal de aviación civil general y tripulaciones de otros países. Esto con el fin de reconocer los síntomas y signos que se pueden llegar a presentar en un vuelo con condiciones de hipoxia hipóxica, en el que el cuerpo pierde saturación de oxígeno en la sangre, como producto de la baja concentración de este en grandes altitudes.

El ejercicio de cámara de altura consiste en someter a tripulantes a una elevada altitud y pruebas de hipoxia para reconocer los síntomas causados por la falta de oxígeno, donde una persona (conocida como observador) hace el monitoreo, alerta y documenta los diferentes efectos que podrían presentar los individuos y también toma acción para evitar que pueda haber algún daño en su salud.

En el desarrollo del ejercicio de cámara de altura, se tiene una disposición de 8 personas a cargo como se puede apreciar en la Figura 1, donde cada miembro tiene una tarea específica.

Figura 1. Equipo Cámara de Altura
Tomada de https://www.sheppard.af.mil/News/Photos/igphoto/2000111071

Así, los tripulantes son asignados cada uno a una estación al momento de iniciar el entrenamiento y los observadores se encargarán de monitorear su estado a medida que se desarrolle el ejercicio e interviniendo en caso de ser necesario.

Para facilitar el entrenamiento, los estudiantes de la Universidad de los Andes desarrollaron un proyecto en el que se estructuró un sistema ciberfísico para la adquisición de signos vitales en el entrenamiento de Cámara de altura. Este permite

identificar en tiempo real, los tripulantes que entren en hipoxia y permite monitorear los datos de pulsioximetria (medición de saturación de oxígeno en sangre) para ver su estado de salud durante el entrenamiento, a través de un servicio web. Además, realiza un almacenamiento de los datos con propósito de ser parte de un historial médico a futuro, para que ayude a la caracterización de los estados físicos de acuerdo con los entrenamientos de los tripulantes.

El sistema ciberfísico cuenta con un dispositivo de IoT instalado en cada una de las estaciones, tipo guante, que los tripulantes se colocan; este utiliza los sensores para recopilar los datos de pulso y saturación de sangre que luego envía junto con los datos de identificación del individuo a un centralizador (conocido como Gateway/servidor). Este último procesa la información y en caso de que el tripulante entre en estado de hipoxia moderada o severa, gestiona una alerta visual a través de Leds (actuadores) y mensajes en una interfaz gráfica web para que el observador tome acción cuando sea necesario.

Figura 2. Dispositivo de sensado tipo guante.

Para garantizar los pilares de seguridad, como mínimo se debería hacer cifrado de los datos en trasmisión, identificación de las estaciones que estén autorizadas dentro del sistema, el acceso seguro al centralizador (Gateway/Servidor) del sistema, contar con métodos de autenticación y, por último, proteger la información almacenada tanto local como remotamente. Todo lo anterior, con el fin de proteger los datos privados y semiprivados del personal de tripulantes de vuelo cuando utilicen el sistema de identificación y medición de signos vitales.

II. Propuesta y Justificación

La ley 1581 de 2012 es la principal en cuanto a el manejo de datos personales a nivel Colombia y de acuerdo con el artículo 5, los datos relativos a la salud son de carácter sensible *[1]* y por tanto debe garantizarse que sean tratados como tales cuando están siendo procesados, almacenados o cuando se encuentran en tránsito, es decir se debe garantizar su confidencialidad y privacidad.

Además, hay decretos que modifican o extienden la ley mencionada, entre ellos se destaca el DECRETO 1377 DE 2013 que establece disposiciones específicas para la protección de datos personales y hace énfasis en que los responsables de la adquisición de los datos deben poder demostrar que han implementado las medidas necesarias para cumplir con todos los requisitos impuestos en la ley 1581.

Así mismo, el marco legal castiga fuertemente a través del código penal, con la ley 1273 del 2009 toda violación de datos personales incluyendo el articulo 269H, que hace un enfoque en los responsables de la administración, manejo o control de la información; de modo que, si hay alguna brecha o ataque relacionado con la información, también se podrá imponer una sentencia con agravante al responsable de la adquisición de la información por no tomar las medidas de protección necesarias.

Por otro lado, estudios realizados entre 1900 y los años 2000 por la Fuerza de Defensa Australiana, en el que participaron tripulantes que habían hecho previamente el entrenamiento, mostraron que el 76% de los casos de hipoxia fueron autodeterminados, un 10% fueron reconocidos por otro miembro de la tripulación y finalmente el 14% no se reconoció *[2]*. Con lo anterior se ve la importancia de tener un control sobre la información respecto a cada agente de la prueba.

El uso del sistema ciberfísico en el desarrollo del ejercicio de cámara de altura, está enfocado en obtener datos de forma objetiva, con el fin de ayudar a tomar decisiones de acuerdo los signos vitales de los tripulantes y de generar un registro de la capacidad de respuesta del tripulante en posibles situaciones de hipoxia. Por este motivo es necesario salvaguardar los datos sensados de la manipulación indebida, perdida o alteración no autorizada, que en consecuencia podría afectar directamente en la salud de los tripulantes de la cámara.

Por lo anterior, se requiere que el sistema ciberfísico no solo cumpla con la funcionalidad de medición y reporte de datos de pulsioximetría, sino que garantice el aseguramiento de la información (saturación de oxígeno en sangre y pulsaciones por minuto). Así se propone diseñar e implementar la arquitectura de seguridad del sistema ciberfísico de medición de pulsioximetría para cámaras de altura, mediante la identificación de los principales puntos vulnerables y aplicando los controles necesarios que garanticen almacenar, trasmitir y procesar los datos sensados, manteniendo el nivel de seguridad y privacidad adecuados.

III. Diseño

A. *Requerimientos*

1) *Requerimientos funcionales.*

a) Capa de aplicación:

- El sistema debe permitir autenticación de los usuarios del sistema.
- El sistema debe permitir generación de logs.

b) Capa de apoyo a servicios y aplicaciones:

- El sistema debe verificar la integridad de los datos sensados antes de procesarlos.

- El sistema debe procesar los datos sensados sin tener en cuenta datos privados (tales como historia clínica y antecedentes médicos) o semiprivados (tales como el documento de identificación) del tripulante.

c) Capa de red:

- Se debe tener autenticación de los diferentes dispositivos conectados a la red.

d) Capa de dispositivo:

- El sistema debe permitir la autenticación de los nodos sensores que están enviando los datos al Gateway.

2) Requerimientos no funcionales.

a) Capa de aplicación:

- El sistema debe brindar confidencialidad a la información que maneje cada usuario.
- El sistema debe brindar privacidad de la información asociada a cada tripulante.

b) Capa de red:

- El sistema debe soportar el envío de información cifrada.

3) Requerimientos técnicos.

a) Capa de red:

- El sistema debe garantizar conexión inalámbrica al interior de la cámara de altura.

b) Capa de dispositivo:

- El sistema debe ser compatible con los sensores MAX30100 y BMP180.

B. Arquitectura y Componentes

El sistema ciberfísico completo de medición de pulsioximetría para cámaras de altura cuenta con los siguientes componentes.

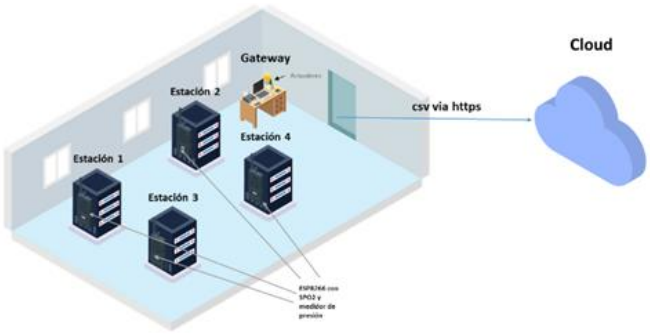

Figura 3. Configuración del sistema completo IoT de medición de pulsioximetría

El sistema tiene 4 estaciones, cada una con un nodo de sensado (sensor MAX30102 y tarjeta de desarrollo Raspberry Pi Pico) y con conexión inalámbrica. Estas envían los datos a un Gateway/Servidor (Raspberry pi a3+), que a su vez tiene incorporado un sensor de presión (BPM180) y los actuadores (leds verde, amarillo y rojo). Por último, el sistema ciberfísico tiene un componente en nube la nube (servicio de almacenamiento de Dropbox) que almacena los datos de identificación de tripulantes, junto con los datos médicos generados en el entrenamiento.

Para facilitar el aseguramiento, se diseña la arquitectura de seguridad dividiendo el sistema en 3 módulos basándose en la interacción entre sus componentes y el estado de la información.

1) Módulo 1 – Estación a Gateway.

El primer módulo está definido por la conexión entre la estación y el Gateway en la que el dispositivo de sensado "tipo guante" de la estación, obtiene la saturación de oxígeno en sangre y pulsaciones para enviarlas al Gateway usando MQTT por WiFi cada 2 segundos a partir del inicio del entrenamiento. El Gateway además de recibir estos datos, envía a las estaciones las órdenes de inicio o parada del ejercicio junto con sus tiempos y los datos de identificación del tripulante. Con la información que obtiene de cada estación, el Gateway procesa los datos para encender los diferentes actuadores de alerta (leds indicadores) en caso de que una estación haya entrado en hipoxia.

Así mismo, el Gateway cuenta con un sensor barométrico para adquirir la información de presión y temperatura a la que está la cámara de altura; esta información es almacenada localmente junto con la obtenida de la estación con el fin de obtener el panorama completo de en qué condiciones y cuándo entra el tripulante en hipoxia.

Teniendo en cuenta que en este módulo se adquieren y transmiten los datos, es necesario que éstos estén asegurados como se muestra en la figura 4. El sistema ciberfísico debe tener un método de identificación única de cada estación hacia el Gateway y además, el canal que se establece por MQTT o los mismo datos deben estar cifrados. Esto, para garantizar la autenticidad e integridad de los datos, además de implementar una forma para reconocer, mitigar y darle trazabilidad a acciones maliciosas como suplantaciones de estaciones, envío de datos no autorizados, modificaciones sobre los paquetes intercambiados o posibles fugas de información. Al mismo tiempo, la estación debería hacer una gestión de su propio estado para evitar que, por errores de lectura, no responda correctamente y no se puedan obtener los datos.

Otro de los puntos a tratar en este módulo, es el manejo de errores y pérdida de conexión lo cual afectaría él envió datos. El sistema no requiere un dispositivo que implemente redundancia en la estación, debido a que los tiempos son cortos y hacer que el tripulante cambie de estación, haría que se pierda el objetivo del entrenamiento y podría ser peligroso en caso de que el tripulante ya haya entrado en estado de hipoxia. Dadas estas condiciones, en vez de sustituir el dispositivo de la estación con falencias, esta necesita enviar su estado al Gateway para que el observador identifique que hay una particularidad con la estación y cambie al proceso manual, en el que el observador está pendiente de los síntomas de hipoxia del tripulante.

Se considera pertinente incluir un modo de verificación del tripulante que se encuentra en la estación que garantice que la persona que está tomando la prueba, sea de hecho el responsable y no un tercero que lo suplante. Para esto se considera como parte del trabajo a futuro, incluir al sistema un mecanismo de

autenticación biométrica en cada estación, que use la huella digital aprovechando que el dispositivo de sensado se conecta directamente a una de las yemas de los dedos del tripulante.

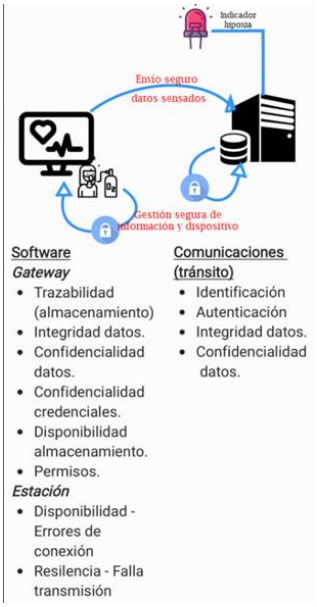

Figura 4. Arquitectura de seguridad módulo 1 del sistema IoT de medición de pulsioximetría

2) Módulo 2 – Gateway/Servidor a Red local (LAN).

El segundo módulo controla la conexión entre el Gateway y la red interna para el consumo de los datos a través de un servicio web. El procesamiento y almacenamiento de datos se hace internamente en el Gateway, el cual lee los datos recibidos desde la estación y los almacena en un archivo .csv de forma local. Al mismo tiempo, el Gateway en este módulo hace las veces de servidor de una página web, en la que se puede ingresar los datos del tripulante, consultar los valores de saturación de oxígeno y pulso cardiaco con respecto al tiempo de manera gráfica y recibir alarmas cuando el tripulante se encuentra en estado de hipoxia severa o moderada durante el ejercicio.

En este módulo, como se muestra en la figura 5, se quiere garantizar que el acceso a los datos y la autorización de consumo, edición y/o eliminación sea solo hecha por actores designados y con los permisos necesarios; por ende, habría que aplicar controles no solo con respecto a los usuarios sino en los canales y plataformas que se están utilizando sin que esto afecte la respuesta en tiempo real de tareas críticas como el reporte de las estaciones y el funcionamiento de los actuadores.

Teniendo en cuenta el funcionamiento y distribución de los módulos, se puede determinar que el sistema tiene un dispositivo centralizador (Gateway/servidor) receptor de las comunicaciones de datos y peticiones. En consecuencia, este debe ser el que autentique a los demás dispositivos, valide la información recibida filtrándola antes de realizar cualquier otra operación con ella y almacene los datos relacionados con las peticiones y acciones de los usuarios a manera de logs, con el fin de mantener la trazabilidad del sistema.

El servicio web debería tener restricciones para ser consumido solo desde la red interna y tener un control sobre la cantidad de dispositivos y peticiones que se puede recibir. Así una vez establecido un canal seguro para el tránsito de la información, la página mostrará los datos del tripulante solo a usuarios autenticados y que tengan los permisos necesarios; la autenticación se haría por medio de formularios con usuario y contraseña para contrastarlos contra los usuarios, roles y privilegios del directorio activo una vez se integre todo el proyecto en las instalaciones de la fuerza aérea; pero para efectos este proyecto, se cubrió el requisito con usuarios creados localmente dentro de la aplicación.

Figura 5. Arquitectura de seguridad módulo 2 del sistema IoT de medición de pulsioximetría

3) Módulo 3 – Gateway a cloud.

El último módulo es el que se encarga de la conexión del Gateway con un servicio en la nube. El Gateway envía el archivo .csv con todas las mediciones del ejercicio a un servidor remoto de Cloud, el cual actúa como almacenamiento permanente, permitiendo tener un histórico de los ejercicios. Este módulo fue creado para garantizar la trazabilidad y backups de los datos en caso de pérdida o eliminación, sin embargo, necesita medidas de protección en cuanto al envío y acceso de los datos en la nube. Las medidas de acceso deben ajustarse en el servidor web de manera que ayuden a mantener un control sobre la lectura y cambios de la información, al igual que un ajuste sobre los permisos y privilegios de los archivos a subir como se puede ver en la siguiente ilustración.

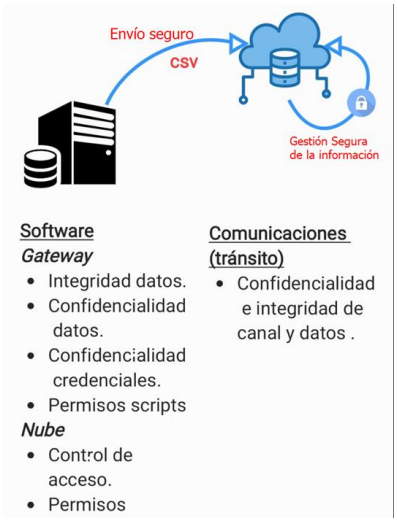

Figura 6. Arquitectura de seguridad módulo 3 del sistema IoT de medición de pulsioximetría.

El sistema ciberfísico en general debería tener controles de autenticación y validación de integridad de la información recibida en los diferentes módulos, haciendo una correcta verificación de que los datos sean los que se esperan y que no puedan ser alterados o afectados por terceras partes.

IV. Implementación.

En la arquitectura propuesta, se utilizó el algoritmo de cifrado y autenticación ligera ASCON, el cual fue un punto relevante para la implementación de controles de seguridad. Teniendo en cuenta la naturaleza del sistema ciberfísico y que se compone de placas de bajo procesamiento y almacenamiento para recibir y enviar datos sensibles y privados de tripulantes de vuelo, se requiere un mecanismo de cifrado que brinde la capacidad de cifrar datos y autenticar dispositivos, consumiendo pocos recursos de procesamiento y en un tiempo adecuado para la trasmisión de estos.

De acuerdo con lo investigado, ASCON provee una protección contra ataques laterales, que impide el uso de criptoanálisis para descifrar la información enviada o recibida, pero es muy importante tener en cuenta aspectos como la longitud de los mensajes que se envían y reciben, las cuales son muy cortas y los mensajes pueden muchas veces, repetirse en diferentes ocasiones, por esa razón es de gran relevancia el cambio del nonce y llaves con periodicidad *[5]*.

Además de ofrecer un cifrado y descifrado simétrico, ASCON cuenta con funciones de hashing y códigos de mensaje de autenticación, usando datos de entrada uniformes y con el mismo core de cifrado por bloques que utiliza funciones de permutación. Es importante mencionar que a pesar de ser un algoritmo basado en criptografía simétrica adiciona un nonce público único para la transacción y la generación de un tag, asemejando a una firma, que ayuda a que el remitente se autentique con el destinatario. De ahí que ASCON brinda las características de seguridad necesarias para mitigar los riesgos asociados a confidencialidad, integridad y privacidad *[5]*.

A. *Módulo 1: Estación a Gateway.*

En este módulo se realizaron pruebas funcionales de algoritmos de cifrado que pudieran ser soportados en el sistema ciberfísico propuesto, seleccionando el algoritmo con esquema ASCON-128 que tiene funcionalidades de cifrado de bloques con procesamiento ligero, utiliza función MAC (Message Autentication Code) que por medio de un mensaje y una llave secreta permite realizar autenticación entre dos entidades y cifrado a través de una llave simétrica pre compartida, para comunicación cifrada entre las estaciones y el Gateway *[17]*.

Como parte del proceso de selección del algoritmo, se realizaron pruebas implementándolo en Python y ejecutándolo en una Raspberry Pi Pico W. Esta placa tiene un procesador de 1GHz, single-core con una CPU de 512MB RAM y basados en las referencias del rendimiento de los procesadores.

TABLE I. Rendimiento de ASCON (bits/seg) con respecto al tamaño del mensaje

(a) Ascon-128

Message Length	1	8	16	32	64	1536	long
AMD Ryzen 7 1700*					14.5	8.8	8.6
Intel Xeon E5-2609 v4*					17.3	10.8	10.5
Cortex-A53 (ARMv8)*					18.3	11.3	11.0
Intel Core i5-6300U	367	58	35	23	17.6	11.9	11.4
Intel Core i5-4200U	521	81	49	32	23.9	16.2	15.8
Cortex-A15 (ARMv7)*					69.8	36.2	34.6
Cortex-A7 (NEON)	2705	250	150	99	73.2	48.8	47.9
Cortex-A7 (ARMv7)	1871	292	175	115	86.6	58.3	57.2
ARM1176JZF-S (ARMv6)	2189	340	202	133	97.9	64.4	65.3

*Results taken from eBACS [BL].

Considerando la información del procesador proporcionada en la tabla de Rendimiento de ASCON (table I), los que más se asemejan en rendimiento a los del sistema ciberfísico son aquellos pertenecientes a la serie Cortex-A53, con una velocidad de procesamiento de 1800 MHz. Estos procesadores requieren 18.3 ciclos para cifrar o descifrar un mensaje de 64 bits (0.286 ciclos/bit). Al ajustar esta cifra para un procesador con una velocidad de 1000 MHz (los utilizados en el sistema ciberfísico), la relación resultante es de 0.159 ciclos por cada bit a cifrar o descifrar, lo cual se considera apropiado para la implementación de este proyecto. Tras confirmar que tiene un buen rendimiento, se procedió a incorporar el algoritmo de ASCON en el Gateway y las estaciones, teniendo como base los ejemplos y código de las páginas https://github.com/eclipse/paho.mqtt.python y https://github.com/meichlseder/pyascon/tree/5ee786cdc8a74d9c0f7b3c81f99f5dcb5490ca00.

Para la conexión y autenticación entre la estación y el Gateway se implementó un protocolo de autenticación con llaves pre compartidas, usando el tag de la función MAC (Message Autentication Code) de ASCON (figura 7), validación de llave de cifrado de comunicación y finalmente envío y recepción de mensajes en MQTT cifrados, de la siguiente forma:

- Paso 1: Se genera un juego de llaves común para las estaciones y el Gateway, adicional se genera el tag del Message Autentication Code (MAC) con las direcciones MAC de las tarjetas Wifi de las placas que se van a usar, con el fin de crear el archivo mac_satation#.tag, estas llaves son guardadas en cada estación y en el Gateway junto con el archivo .tag de cada estación. Aunque las llaves de

cifrado se almacenan en cada dispositivo, el entorno en el que son resguardados cuenta con las medidas de seguridad física para proporcionar la protección de estos componentes.

- Paso 2: Con la creación y actualización de llaves y tags, la estación selecciona de forma aleatoria una de las llaves pre compartidas y crea un tag con la llave seleccionada y la dirección MAC de la tarjeta Wifi de la placa y publica el tag en el tópico hand_shake#.
- Paso 3: El Gateway se suscribe al tópico hand_shake# y toma el tag para realizar una comparación con los tags guardados en los archivos mac_satation#.tag de cada tarjeta y las llaves pre compartidas, la comparación se realiza realizando la operación de MAC con todas las llaves guardas y las direcciones MAC de las tarjetas autorizadas para el ejercicio, una vez se encuentra el tag que coincida con el publicado en el al tópico hand_shake#, se establece la llave de cifrado simétrico. Posterior a esto se crea un MAC con la palabra ESTABLISH para hacer saber a cada estación que el Gateway está listo para recibir información, este MAC se publica en el tópico hand_shake#ask, adicional se publica en el tópico workStation-#/state "3", significa que la estación está bien y está lista para enviar datos o en caso que no se logre realiza la autenticación se publica en el tópico workStation-#/state "1" que significa que la estación está mal y no está lista para enviar datos.
- Paso 4: Una vez se verifica el MAC_ESTABLISH en la estación se termina el proceso de autenticación, que se espera no tarde más de 2 segundos, y a partir de este punto se establece un método de envió cifrado de datos entre la estación y el Gateway, se inicia la publicación de información cifrada en los tópicos workStation-#/spo2 (para datos de oximetría), workStation-#/bpm (para datos de frecuencia cardiaca) workStation-#/mac (para dirección MAC del dispositivo) y workStation-#/data (para la información del archivo csv tiempo | MAC | id_tripulante | nombre_tripulante | oximetría | frecuencia cardiaca).
- Paso 5: Con la llave de cifrado establecida el Gateway inicia la suscripción a los tópicos de trabajo para procesar los datos sensados.

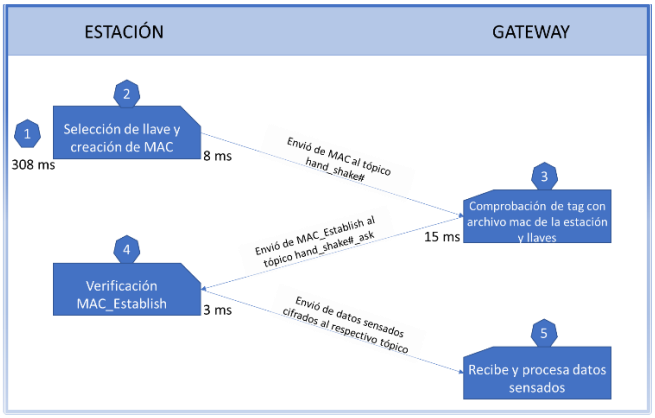

Figura 7. Protocolo de autenticación sin certificado.

Para realizar la implementación exitosa y segura del algoritmo de ASCON en el sistema ciberfísico, se tomaron varias medidas tanto en el Gateway como en la estación. Dentro de ellas, está el uso de variables de entorno para llamar los datos sensibles (llaves de cifrado y autenticación, contraseñas y usuarios) en scripts, evitando que queden quemadas en el código y haya una fuga que afecte a la confidencialidad de las mismas y en últimas la seguridad de todo el sistema.

1) Gateway.

Por otro lado, para realizar la validación del tag del algoritmo de cifrado se requiere un archivo. tag similar al que se muestra en la figura. Este junto con las variables de entorno deben cambiarse en un periodo inferior a 1 mes, con el fin de mantener las credenciales controladas por el personal técnico del proyecto.

Figura 8. Contenido archivo mac_station1.tag

Es importante aclarar que la dirección MAC de los dispositivos y la clave secreta, no se comparte en texto plano en ningún tópico de MQTT ya que se establece entre las estaciones y el Gateway, como se observó previamente. Una vez el Gateway autentique cada estación, va a publicar los tópicos workStation-#/start (inicio o parada del ejercicio), workStation-#/interval (que establece cada cuanto se realiza el envío de información al Gateway y se actualiza la información en Node-red), workStation-#/id (número identificación del tripulante), workStation-#/name (nombre del tripulante) a la estación correspondiente.

2) Estación.

Adicionalmente, cada estación tiene configurado el intento de conexión de forma reiterativa en caso de falla, garantizando que regrese al último punto exitoso de la autenticación. De la misma manera, si el fallo de conexión ocurre después de la autenticación, la llave de cifrado se guarda de forma temporal en una variable de entorno, lo que le va a permitir seguir enviando información cifrada con la llave pre acordada. Este es un mecanismo de protección contra los fallos de conexión o errores que pueden generar indisponibilidad.

3) Comunicaciones en tránsito.

De acuerdo con la información dada, se logra una trasmisión de los paquetes de forma segura al cifrarlos con ASCON-128. Al hacer una interceptación de los datos se observa que no se puede recuperar la información rápidamente y no se puede modificar ya que esto generaría errores de descifrado en los siguientes módulos. Se cumple el objetivo de garantizar la confidencialidad e integridad de la información en tránsito desde los sensores hasta el Gateway para su posterior procesamiento.

B. Módulo 2 – Gateway/Servidor a Red local (LAN).

En este módulo el sistema ciberfísico cuenta con un servicio web, a modo de GUI (Graphical User Interface), para presentar la información del tripulante por medio de un dashboard. El servicio es gestionado por el mismo Gateway, convertido en servidor, a través de Node-red y es consumido por los usuarios conectados a la red interna a través del navegador.

Node-red es una herramienta de programación visual basada en JavaScript que permite hacer la recepción, procesamiento y transmisión de información por medio de funciones representadas en bloques, llamados nodos. Estos se emplearon para crear la lógica de aseguramiento usando JavaScript y Python como lenguajes base y con el fin de no tener que cambiar las tecnologías ya instaladas en el Gateway.

Mediante esta herramienta se gestionó la seguridad de los datos recibidos de las estaciones por MQTT, los ingresados por el usuario desde el servicio web y las configuraciones del servicio web, peticiones y conexiones.

1) Comunicaciones en tránsito Gateway/Servidor.

Entre las cosas más relevantes se encontraba asegurar el canal por el que se estaba presentando el dashboard ya que este era consumido por http, dejando toda la información que viajaba en texto plano. Para esto se generaron e instalaron certificados de servidor web, enviando la información de los servicios web a través de un canal seguro como HTTPS/TLS y hacer redirección en caso de que se reciba http.

Gracias a la herramienta OpenSSL se generaron las llaves y se firmaron los certificados con algoritmos de cifrados seguros como SHA256 y RSA2048; así mismo, estos archivos se generaron con un usuario de altos privilegios para que los permisos fueran lo más restringidos posibles y que su modificación o consulta solo se hicieran por personas autorizadas.

Las rutas del certificado y llaves fueron agregadas al settings.js de Node-red para que éste despliegue la interfaz gráfica por https, y se habilita la opción “RequireHttps” para redireccionar las posibles conexiones por http. Con estas medidas se fuerza a que los datos viajen cifrados y de ahí que no pueda ser leída o modificada por terceros en la red.

```
root@raspberrypi:~# ls -al .node-red/tls/
total 20
drwxr-xr-x 2 root root 4096 Nov 17 16:45 .
drwxr-xr-x 5 root root 4096 Nov 21 04:04 ..
-rw-r--r-- 1 root root 1302 Nov 17 16:45 certificate_node-red.crt
-rw-r--r-- 1 root root 1045 Nov 17 16:43 certificate_node-red.csr
-rw------- 1 root root 1675 Nov 17 16:41 node-red.key
root@raspberrypi:~#

/** Option 1: static object */
https: {
  key: require("fs").readFileSync('/root/.node-red/tls/node-red.key'),
  cert: require("fs").readFileSync('/root/.node-red/tls/certificate_node-red.crt')
},

/** The following property can be used to cause insecure HTTP connections to
 * be redirected to HTTPS.
 */
requireHttps: true,
```

Figura 9. Habilitar TLS/HTTPs en node-red.

Como medida extra se configuraron los límites de conexión de modo que el sistema no sea propenso a denegaciones de servicio por consumo de recursos cuando un usuario quiere hacer múltiples peticiones en un lapso muy corto. Modificando el archivo de ajustes de Node-red, en la opción de “httpNodeMiddleware” se puede ajustar estas opciones para que acepte no más de 10 peticiones cada 3 segundos.

Por otro lado, con el fin de evitar vulnerabilidades de configuración por defecto y reducir el riesgo de acceso no autorizado a recursos, se realizaron modificaciones en las rutas por defecto que Node-Red utiliza para la consola administrativa (en la que está la lógica del código) y el dashboard. El que se pudiera acceder con tan solo digitar la ip/dominio y el puerto, facilitaba el hallazgo de los endpoints para consumir servicios administrativos y obtener información de las estaciones sin autorización.

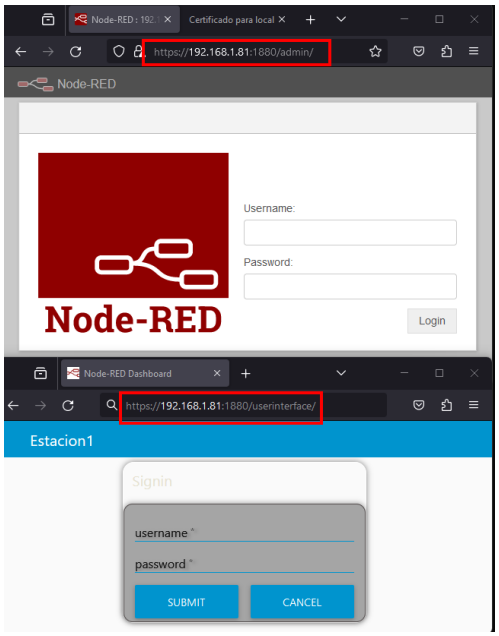

Figura 10. Cambio de rutas por defecto en endpoints.

Los cambios implementados fueron modificar las líneas "httpAdminRoot" con '/admin' y "ui" con {path: "userinterface"} en el archivo settings.js de Node-Red. Así, como se puede ver en la ilustración anterior, el acceso a estas rutas es más indirecto y gracias a que este control se complementa con el de límite de peticiones, se mitigan ataques de escaneo de rutas por aumento en los tiempos de descubrimiento.

2) Gateway/Servidor.

De acuerdo a lo expuesto en la arquitectura de seguridad del Gateway para este módulo, luego de recibir los datos, requerían descifrado para hacer su procesamiento y mostrarlos en el servicio web, sin embargo, en primera instancia se revisa que el usuario tenga una sesión activa dentro de la aplicación web y que el ejercicio haya comenzado. Así, individuos externos no puedan hacer la consulta o modificación de datos sin autorización. Luego de esto se ejecuta un código de Python que contiene la función de descifrado de ASCON (explicado con anterioridad) mediante el nodo "exec" de Node-red; este recibe como entrada lo obtenido por MQTT a través del nodo "mqtt in" una vez se haya validado la sesión.

Si se encuentra alguna falla al momento de descifrar, Node-red genera una notificación con el error en la interfaz gráfica; de lo contrario, luego de obtener el mensaje en texto plano (salida del nodo exec) se pasa el dato al nodo de validación de datos. Esto se realizaba por cada uno de los tópicos recibidos por MQTT: workStation-#/spo2, workStation-#/bpm, workStation-#/mac, a excepción del tópico workStation-#/data ya que hace referencia a los datos cifrados a almacenar en el archivo .csv y se envía a cloud (módulo 3). A continuación, se ve un ejemplo del tópico workStation-1/spo2 que transmite la información relacionada a la saturación de oxígeno en la sangre.

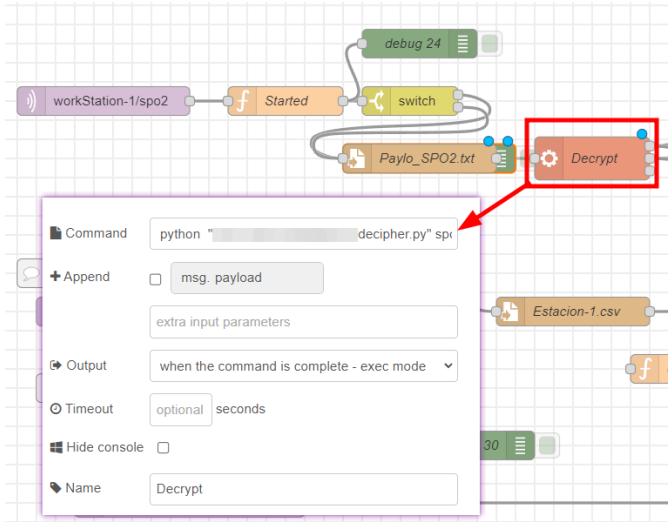

Figura 11.Proceso de datos cifrados recibidos por MQTT

El nodo "*función*" contiene el código encargado de la validación de los datos de entrada, tomando la salida o "*payload*" de los nodos anteriores que no generaron errores, es decir, los datos descifrados. A este se le hace una verificación de longitud, tipo y de caracteres mediante un RegExp dependiendo de la estructura deseada y generando un error que se presenta al usuario en caso de que los datos no fueran los esperados.

De lo contrario, se pasan los datos al dashboard para ser mostrados al usuario final y ser guardados o procesados de acuerdo a su funcionalidad dentro del sistema. Esto garantiza que los datos sean los esperados y no generen comportamientos inesperados dentro del Gateway o la aplicación del servicio web.

A continuación, se observa el flujo para el tópico workStation-1/spo2 que solo acepta números con una longitud máxima de 3 dígitos (Porcentaje de saturación entre 0 y 100). Si los datos recibidos no seguían este patrón o se corrompían durante el envío o descifrado, se generaba un error predeterminado que se mostraba en el dashboard. Finalmente, aquellos que se ajustaban a los requerimientos eran mostrados en el dashboard y analizados/procesados para activar el actuador en caso de ser necesitarlo.

Figura 12. Validación de datos descifrados o ingresados por usuario.

Los datos recibidos desde el dashboard de la página web, son verificados de una forma similar. Luego de ingresados y verificar que hay una sesión activa, pasan a un nodo "*función*" que valida que se ajusten a la estructura esperada, después son cifrados con un nodo "exec" usando un script de Python "*chiper.py*" y se envían por mqtt para ser recibidos por la estación. Ya que estos datos son ingresados antes de empezar el entrenamiento en la cámara de altura, se genera un error preestablecido y se muestra en la GUI, esto evita iniciar el ejercicio de recolección con información errónea o con contenido sospechoso.

Como ejemplo, en la siguiente imagen se ve el flujo para una de las entradas de usuario que se completa con el nombre del tripulante. Node-red lo recibe con el nodo "text input" del módulo dashboard, hace la validación con el RegExp "^(?:[a-zA-Z](?:[,.'-]?[a-zA-Z])?){1,25}". Si cumple con las condiciones, cifra la data y la envía por mqtt en el tópico "workStation-1/name".

Figura 13. Validación de entradas por usuario, cifrado y envío a estación

Para garantizar trazabilidad de los datos del piloto durante el ejercicio en la cámara de altura, estos se almacenan automáticamente cifrados en el Gateway una vez son recibidos (figura 14). Esto se realiza con el fin de que los datos no sean alterados, accedidos o vistos por usuarios no autorizados. El archivo queda guardado con permisos de root.

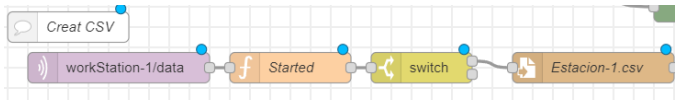

Figura 14. Flujo de Node-red para generar archivo con logs.

Los datos sensados de cada estación se almacenan dentro del Gateway. No obstante, tras la realización de múltiples ejercicios se podrían presentar inconvenientes en términos de almacenamiento y procesamiento debido al espacio utilizado. Por esta razón y para evitar problemas de disponibilidad de recursos, se necesita optimizar el almacenamiento local del gateway. El mecanismo para resolver esto, fue realizar un backup de los datos en la nube (módulo 3) y liberar el espacio con una tarea automatizada, que corre cada lunes a media noche para eliminar los archivos luego de subirlos a la nube.

Por otro lado, también para garantizar la trazabilidad de los datos y acciones de los usuarios, se obtienen todos los mensajes de error y eventos, agregando los tiempos en que sucedieron y se agregan al archivo de logs "logs.txt". La función de estos logs es ayudar a identificar comportamientos anómalos como lo son múltiples intentos de login fallidos o inclusión de código en las entradas del usuario. Cabe resaltar que los mensajes fueron predefinidos para evitar que información sensible quedara almacenada en bitácoras de los errores producidos, como se ve en la figura 15.

Por otro lado, para Node-red también se habilitaron los logs en el archivo settings.js, esto para obtener información de la herramienta relacionada con errores, información general de la aplicación, características de ejecución y los eventos generados por la API de admin que va por http. La habilitación se dejó con la opción "info" que incluía información de la aplicación, errores y alertas, pero no incluía información de contraseñas o información confidencial como identificadores de sesión.

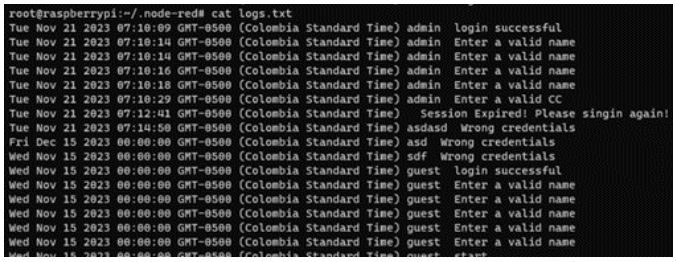

Figura 15.Almacenamiento de log generados por la aplicación web.

3) Servidor - Cliente.

Para el aseguramiento entre el servidor y el cliente era necesario implementar controles de acceso y autorización para la página web que mostraba el dashboard y para el servicio administrador, ya que estos eran mostrados directamente sin ningún tipo de autenticación o validación de identidad.

Para esto, se implementaron los formularios y la lógica de login en:

- Dashboard: Usado para consultar los datos de la estación (GUI)
- Consola de administrador: Usada para acceder y/o modificar la lógica del código con los nodos.

El primer servicio a asegurar es el de administración porque es el que contiene toda la lógica del código, así como todos los flujos de Node-red que administran y procesan tanto datos como los servicios del sistema ciberfísico del Gateway. Node-red ofrece una forma de autenticarse haciendo login con usuario y contraseña y configurando con los permisos de acuerdo al rol a desempeñar.

Esto se realiza modificando una vez más el archivo settings.js, que contiene en la propiedad adminAuth, una sección para definir los usuarios con sus respectivas características y la duración de la sesión del servicio administrador. Para este caso, la sesión estaría activa por 3 horas, luego de esto el usuario deberá volver a autenticarse.

Figura 16. Habilitar control de acceso para servicio administrativo de node-red.

Para evitar brechas de seguridad por exponer contraseñas en texto plano en archivos de configuración, la inclusión de contraseñas dentro del settings.js se debe hacer aplicándoles una función de hashing con antelación. De acuerdo a la documentación de Node-red el algoritmo soportado para generar el hash es bcrypt, que es un algoritmo robusto y al día de hoy considerado como seguro desde que sea aplicado con una contraseña lo suficientemente fuerte. Así que se construyó una contraseña alfanumérica de 30 caracteres que aumentaría el tiempo y costo de procesamiento si alguien intenta hacer brute-forcing sobre el hash.

Node-red cuenta con otra propiedad “httpNodeAuth” para generar una autenticación, pero sobre los endpoints http de Node-red, incluyendo el dashboard. Ya con esto, tanto para el servicio administrador como para los servicios que van por http solicitarán usuario y contraseña antes de logearse al dashboard. Sin embargo, la propiedad de httpNodeAuth no provee un mecanismo para limitar el tiempo de sesión después de la autenticarse en el portal web, por lo que como mitigación se decide incluir una página de login exclusivamente para el dashboard, además del control previamente explicado.

Así, en el flujo de Node-red se incluye una ventana que solicitaba el usuario y contraseña, los compara contra los usuarios almacenados localmente y si éste existía dentro de los registros, se inicia la sesión con un client id que duraba por 3 horas antes de cerrarse y volver a pedir credenciales.

El tiempo de la sesión empieza solo cuando el usuario se ha autenticado correctamente (ítem 1 de la imagen) y se asigna el id de sesión. El tiempo de vida se ajusta a 10800000 milisegundos (3 horas) y desciende por segundo hasta llegar a 0 (ítem 2 de la imagen). Luego de esto hay un timeout de la sesión, revocando el id de sesión y redirigiendo a la página de inicio para pedir credenciales de nuevo. [El código de login fue tomado de https://github.com/phyunsj/node-red-dashboard-login y ajustado de acuerdo a las necesidades para este proyecto]

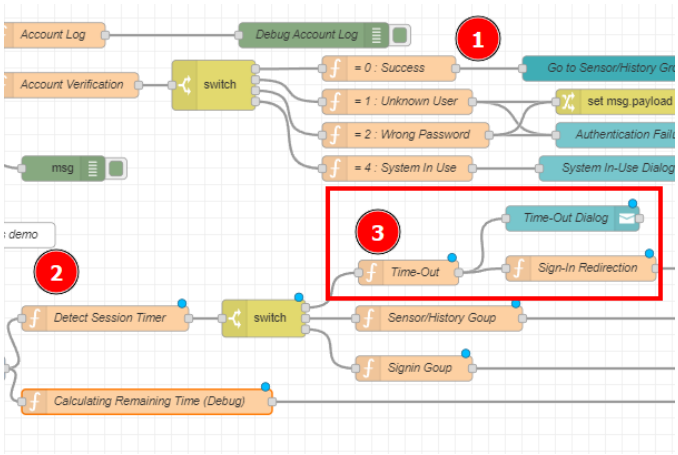

Figura 17. Flujo para crear control de acceso y sesión del dashboard

Con esto, el módulo cumple con todo lo planteado en la sección de arquitectura.

C. Módulo 3: Gateway a Cloud.

Este módulo se encarga de transportar el archivo con los datos de la estación a cloud de manera segura. Para esto se usó el gestor de almacenamiento en la nube Dropbox y su documentación, que incluía un script en Python [https://raw.githubusercontent.com/dropbox/dropbox-sdk-python/master/example/updown.py] para hacer la carga y descarga de archivos y carpetas a una aplicación de Dropbox.

1) Nube.

En primera instancia se creó la aplicación en Dropbox y se generó el token de acceso (requerido por el script updown.py) y se le asignaron los permisos más restringidos posibles. Como el consumo de los datos para incluirlos en la historia clínica esta por fuera del alcance del presente proyecto y solo es necesario almacenarlos, se marcaron los permisos a solo escritura en la carpeta en la nube y los datos solo pueden ser accedidos por el usuario creador de la aplicación en Dropbox, como se ve en la siguiente imagen.

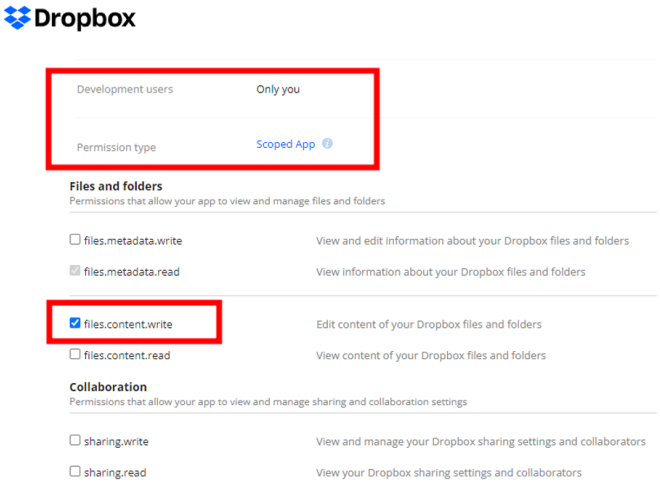

Figura 18. Ajustar permisos de aplicación de Dropbox.

2) Gateway.

Gracias a la implementación del cifrado de ASCON en los módulos pasados, los datos son almacenados cifrados al llegar a este módulo. Lo que se necesita para este segmento es verificar que las tareas o scripts que utilizan el archivo de datos cifrado, no reciba entradas de usuarios o se pueda modificar de alguna forma por un usuario no autorizado.

Teniendo en cuenta lo anterior, se modifica el script de subida de archivos a Dropbox, modificando los parámetros que eran requeridos por consola o entradas de usuarios para dejarlos estáticos dentro del código. Además, aquellos datos sensibles como el token de acceso de Dropbox se reemplazaron por variables de entorno para evitar dejar información sensible en texto plano dentro de los scripts de los dispositivos.

Para culminar con la configuración de los scripts, se dejan los permisos únicamente para ser leídos, modificados o ejecutados por un usuario privilegiado, de modo que solo root o una tarea a su nombre puede correr el comando que sube el csv con los datos cifrados de la estación a la nube.

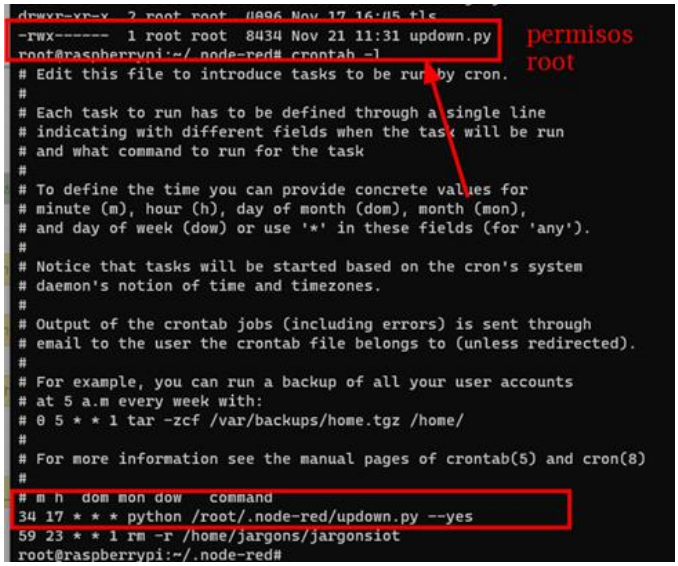

Figura 19. Automatización de backup de datos cifrados de estaciones a la nube.

3) Comunicaciones en tránsito.

El aseguramiento entre el Gateway y la nube se da a partir del envío de los datos. Como se ha mencionado con anterioridad, ahora el contenido del archivo está cifrado y los endpoints para subirlo a la nube se comunican por un canal seguro (HTTPS/TLS) haciendo uso del api oficial de Dropbox, que a su vez, solicita un token de acceso como mecanismo de autenticación.

V. ANÁLISIS DE RESULTADOS.

Al efectuar pruebas del tiempo que tarda el proceso de cifrado y descifrado del mensaje junto con su envío al tópico y construcción del archivo .csv (figura 20), se puede observar que el tiempo de la operación criptográfica no tarda más de 100 milisegundos, ajustándose a lo que se había planteado de realizar esta operación con un tiempo inferior a los 2 segundos.

Figura 20. Prueba de tiempos de ejecución de funciones de cifrado y descifrado

Una vez desarrollados estos dos códigos de programación, se realizaron capturas de paquetes con la herramienta WireShark, simulando un ataque de hombre en el medio, con el fin de comparar la comunicación entre las estaciones y el Gateway, para esto se crearon 2 laboratorios, el primero se realizó verificando la publicación sin seguridad en el tópico workStation-1/spo2, con los siguientes resultados:

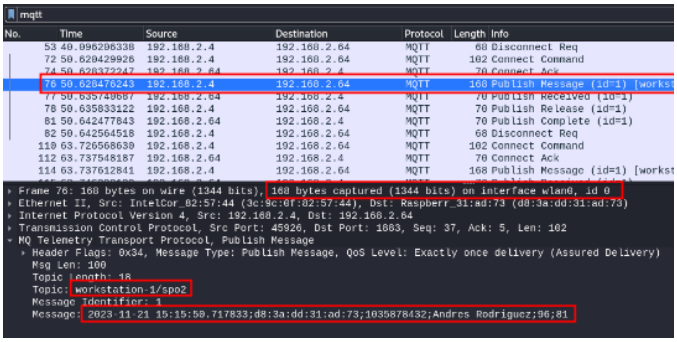

Figura 21. Captura de los datos sin cifrado.

Se observa el payload de la publicación en texto claro, quedando completamente expuesto, al no contar con ningún tipo de seguridad en caso de un ataque de hombre en el medio.

Por otro lado, al implementar el cifrado con ASCON, los datos del paquete capturado ocupa 387 bytes, duplicando el peso del paquete sin cifrar de 168 bytes; pese a esto, el tamaño está contemplado para la capacidad de almacenamiento de los dispositivos planteada en el levantamiento de requerimientos y adicionalmente, se tiene una tarea semanal que elimina estos datos del Gateway luego de pasarlos a la nube.

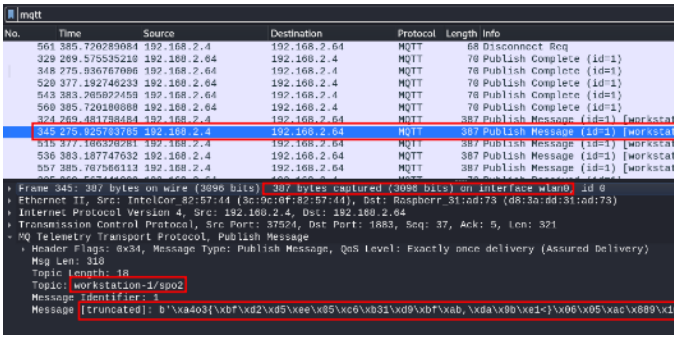

Figura 22.Captura de los datos con cifrado.

Lo mencionado sirve como control para garantizar la confidencialidad de los datos y detectar cuando estos sean modificados a causa de un actor malicioso en el medio de las comunicaciones. De la misma forma, manejar el almacenamiento ayuda a evitar problemas de disponibilidad a causa de consumo excesivo de memoria.

De igual forma se realizó captura del inicio de comunicación entre las estaciones y el Gateway para intentar encontrar información que ayude a descifrar el contenido del paquete, pero lo único que se puede ver es el valor del tag. Lo anterior gracias a que las operaciones de cifrado, descifrado y creación del MAC se realizan directamente en el Gateway y las estaciones, sin enviar ningún dato sensible en texto plano.

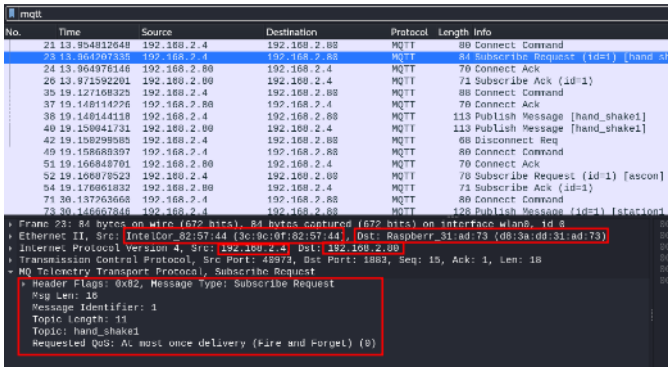

Figura 23. Captura de hand_shake

Con el fin de tener trazabilidad de los diferentes eventos que se puedan dar en MQTT, se establece como gestor de logs el que brinda la aplicación de Mosquitto, configurándolo en el Gateway para que guarde todos los eventos en un archivo .log, como se observa a continuación:

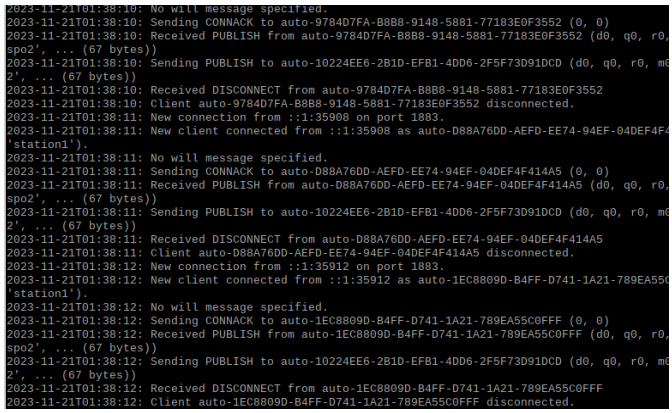

Figura 24. Ejemplo del archivo .log de las acciones guardas con fecha, hora y descripción del evento.

Por lo que se refiere al aseguramiento del módulo 2, tuvo retos en cuanto a buscar alternativas que se ajustaran a la tecnología que hay actualmente en el proyecto. Node-red permite una versatilidad en la configuración de ciertas características como certificados e implementación de TLS sobre los canales; pero al mismo tiempo es poco flexible con los módulos o librerías que permite utilizar para el manejo y modificación de datos cifrados.

En cuanto al descifrado y cifrado en Node-red de la información recibida por MQTT, no se logró utilizar la dependencia ascon-js porque este no soporta módulos ES6 de JavaScript; gracias a esto, se necesitó importar y llamar comandos de terminal para hacer el descifrado y cifrado con scripts de Python.

A pesar de lo anterior, luego de obtener la información, Node-red facilita la creación, diseño y despliegue de aplicaciones web ya que cuenta con módulos que se encargan de la parte gráfica y algunos controles. Y los controles que aún no existen, se pueden desarrollar con scripts JavaScript como se realizó con el formulario de login y manejo de sesión del dashboard.

Basado en esto, se pudieron incluir controles de tiempo y validación de usuarios que antes no tenía el sistema. En un principio el dashboard solo necesitaba del endpoint para cargar y permitía a cualquier persona ingresar, obtener, modificar y comunicar datos a o desde las estaciones y el Gateway.

Los cambios propuestos no solo dejan identificar al usuario, sino que le da acceso a la interfaz gráfica, controlando el tiempo que tiene una sesión activa y evitando que esta quede abierta para ser consumida por cualquier persona dentro de la red interna con tan solo un navegador, por un tiempo indefinido.

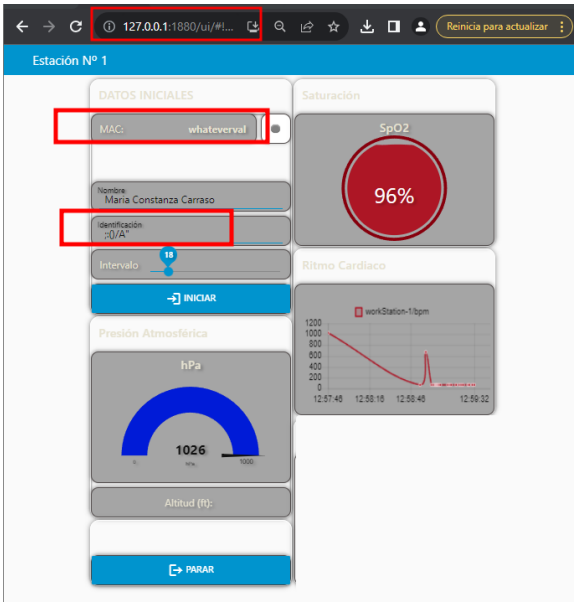

Figura 25. Dashboard del sistema original para presentar los datos.

Como se ve a continuación, con la arquitectura propuesta ahora se solicita usuario y contraseña para ingresar tanto al servicio web que muestra el dashboard, como a el servicio administrativo y los endpoints que se consumen por https. Las contraseñas fueron ajustadas de acuerdo a los requerimientos de seguridad (Mayúsculas, minúsculas, caracteres especiales, números y como mínimo 8 caracteres de longitud).

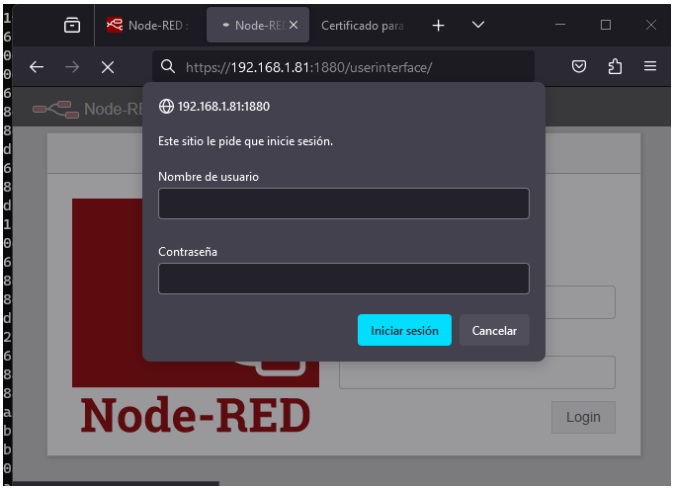

Figura 26.Control de acceso para servicio administrativo.

En caso de que los usuarios no existieran o la contraseña estuviera incorrecta, se modificó la lógica del código de login para evitar la enumeración de usuarios o de contraseñas. Gracias a que el sistema siempre devuelve el mismo mensaje sin importar el error de autenticación que se tenga, se protege confidencialidad de usuarios.

Entre las pruebas que se realizaron para comprobar los controles de autenticación y autorización, se encuentran el envío de información sin sesión iniciada, modificación de datos desde estaciones falsas, reúso de los tokens de sesión luego de expirar, solicitud de información a través de peticiones de http sin identificadores de sesión y sin haber iniciado el entrenamiento. Todas las anteriores generaban correctamente los errores definidos para cada caso y no se obtuvo ni modificó información de forma no autorizada.

Otro punto a favor de la implementación hecha es que la verificación de tipos de variables y la validación de entradas por estructura y longitud se hace en todos los datos recibidos ya sea desde los sensores o desde el servicio web (figura 27). Esto ayuda a hacer un manejo de errores y previene ataques de inyección que pueden resultar en denegaciones de servicio o comportamientos imprevistos, en el peor de los casos, ejecución de comandos que a su vez podría escalar y comprometer todo el sistema.

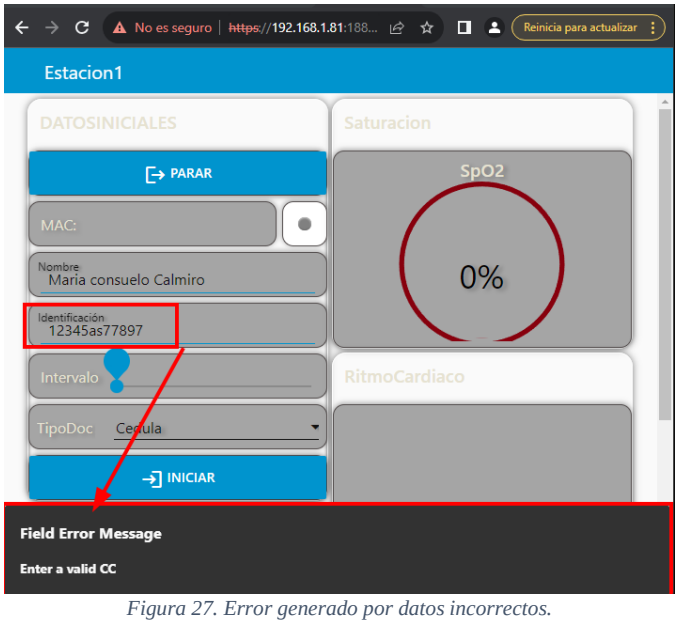

Figura 27. Error generado por datos incorrectos.

Como parte del prototipo de este proyecto se realizó la instalación de certificados en servidor web para asegurar que la conexión se estableciera con los dominios específicos y garantizar que el canal estuviera cifrado a través de la comunicación por HTTPS/TLS. Después de implementado, los navegadores seguían marcando el sitio como inseguro, pero esto se debe a que para efectos de este proyecto los certificados fueron auto firmados y en consecuencia el CN no concuerda con el dominio. Se tiene pensado que cuando haga la integración del proyecto con las tecnologías de la entidad (Fuerza Aérea Colombiana), se adquiera un certificado de una autoridad de certificación (CA) por lo que por ahora el riesgo puede asumirse.

A pesar del error por el certificado, al interceptar las peticiones enviadas por la aplicación web (figura 28), se puede observar que la opción de TLS está habilitada y de ahí que estas viajan por un canal seguro garantizando la confidencialidad e integridad de los datos ante agentes externos.

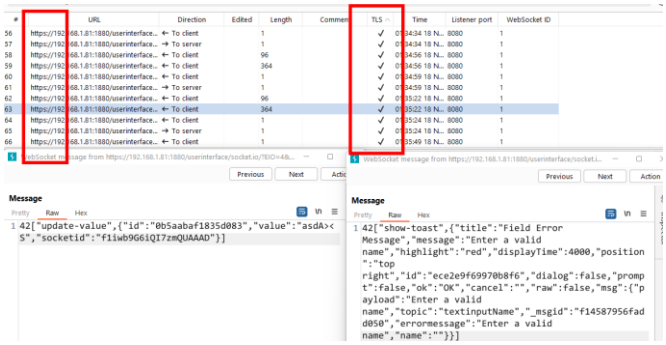

Figura 28. Interceptación local de peticiones para verificar canal(TLS).

Por último, en cuanto al módulo 3, de acuerdo con el esquema de comunicación propuesto de la estación al Gateway original, se disponía de un tópico que recibía información para almacenar en un archivo de texto plano el historial de cada ejercicio realizado en la cámara de altura (con la información de tiempo | dirección MAC | id_tripulante | nombre_tripulante | oximetría | frecuencia cardiaca). Con los cambios propuestos, esta información se recibe cifrada desde el tópico workStation-#/data y es almacenada de la misma manera localmente y en la nube, garantizando su integridad y confidencialidad como se puede apreciar en la siguiente imagen.

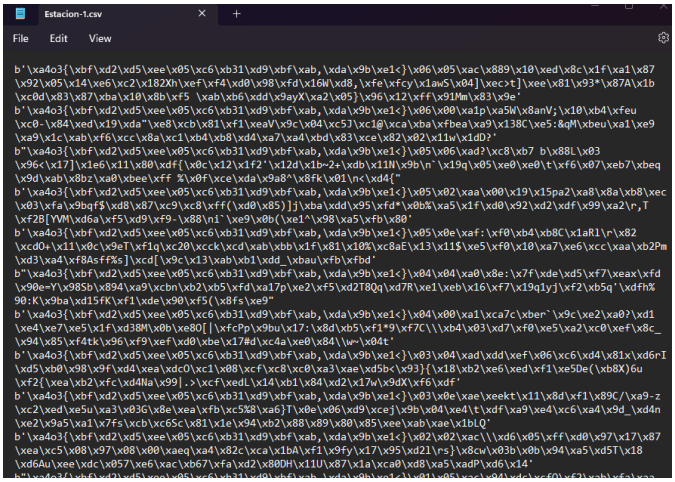

Figura 29. Archivo generado de la estación que se sube a cloud.

VI. Conclusiones y Trabajo a Futuro

Teniendo en cuenta los controles propuestos, se considera que los requerimientos de seguridad en este documento fueron completados luego de identificar y atacar los problemas que existían en el procesamiento, almacenamiento y envío de la información en cada uno de los módulos. Así mismo, se explicó su impacto teniendo en cuenta los tres pilares principales de información (confidencialidad, integridad y disponibilidad).

Teniendo en cuenta que es un sistema IoT, hay restricciones en la implementación de algoritmos robustos o librerías específicas, sin embargo, se logró adaptar el algoritmo de cifrado ligero ASCON, para autenticación y cifrado, con resultados muy significativos, logrando tiempos de procesamiento menores a 1 segundo, no alcanza a tener connotación de trasmisión en tiempo real, pero si se cumplió con lo requerido por el personal médico, que era lograr un tiempo de sensado entre 2 a 5 segundos, es relevante para trabajos futuros verificar el protocolo de autenticación que permita autodescubrimiento de nodos nuevos con certificación.

También resaltar que todos los puntos importantes de la aplicación quedaron con un mecanismo de autenticación, basados en contraseña y usuario; estos datos son almacenados localmente por lo que a largo plazo podrían provocar un problema con respecto a los recursos consumidos por el Gateway a medida que el número de empleados aumente. Por lo anterior, se recomienda hacer una integración del mecanismo de autenticación con un sistema centralizado de autenticación o directorio activo, lo que ayudaría a que el sistema ciberfísico se enfoque en el procesamiento y manejo de la data recolectada durante las simulaciones de vuelo en la cámara de altura.

Por último, recordar que dentro de este proyecto no se están considerando requerimientos de seguridad física ni de autodescubrimiento o autoconfiguración de los dispositivos ya que la estructura física y de los dispositivos tiene que permanecer estable; cualquier cambio generado en los dispositivos tiene que ser hecho manualmente y supervisado por el interesado considerando que esto se encuentra por fuera del alcance.

VII. References

[1] Congreso de la República de Colombia, "Régimen legal de Bogotá D.C.," Ley 1581 de 2012. [En línea]. Disponible: https://www.alcaldiabogota.gov.co/sisjur/normas/Norma1.jsp?i=49981. [Accedido: 18 de agosto de 2023].

[2] Ramírez Madrid D., "Metodología para la implementación de IoT," 2018. [En línea]. Disponible: https://repository.udistrital.edu.co/bitstream/handle/11349/13742/RamirezMadridDavidAndres2018.pdf?sequence=1&isAllowed=y. [Accedido: 20 de agosto de 2023].

[3] Congreso de la República de Colombia, "Ley 1581 de 2012." [En línea]. Disponible: https://www.alcaldiabogota.gov.co/sisjur/normas/Norma1.jsp?i=49981. [Accedido: 26 de septiembre de 2023].

[4] Digital Talent Agency, "Modelo de Gestión de Proyectos." [En línea]. Disponible: https://www.dtagency.tech/cursos/metodologias_gestion_proyectos/tema_1-ModeloWaterfall.pdf. [Accedido: 12 de noviembre de 2023].

[5] Dobraunig, C., Eichlseder, M., Mendel, F., & Schläffer, M., "ASCON Lightweight Authenticated Encryption & Hashing." [En línea]. Disponible: https://ascon.iaik.tugraz.at/. [Accedido: 22 de noviembre de 2023].

[6] Función Pública, "Decreto 911/23." [En línea]. Disponible: https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=211210. [Accedido: 2 de junio de 2023].

[7] Oficina de Prensa Fuerza Aérea Colombiana, "Fuerza Aérea Colombiana adquiere Cámara de Altura, para entrenamiento de sus tripulaciones." [En línea]. Disponible: https://www.fac.mil.co/es/noticias/fuerza-aerea-colombiana-adquiere-camara-de-altura-para-entrenamiento-de-sus-tripulaciones. [Accedido: 10 de junio de 2010].

[8] Sarmiento, J. A., & Correa, C. H., "Gestión de proyectos aplicada al PMBOK 6ED," Editorial de la Universidad Pedagógica y Tecnológica de Colombia, 2020.

[9] Unión Internacional de Telecomunicaciones, "Recomendación Y.4000/Y.2060 (06/12).", 2014. [En línea]. Disponible: https://www.itu.int/rec/dologin_pub.asp?lang=s&id=T-REC-Y.2060-201206-I!!PDF-S&type=items. [Accedido: 20 de octubre de 2023].

[10] National Institute of Standards and Technology, "Considerations for Managing Internet of Things (IoT)," Arlington, Virginia, 2019.

[11] National Institute of Standards and Technology, "IoT Non-Technical Supporting Capability Core Baseline," National Institute of Standards and Technology, 2021.

[12] Dover T. P, "Evaluating Medical IoT (MIoT) Device Security using NISTIR-8228 Expectation," 2021.

[13] S. Thapa, Bello A., Maurushat A. y Farid F., "Security Risks and User Perception towards Adopting Wearable Internet of Medical Things," Int. J. Environ. Res., vol. 5519, p. 20, 2023.

[14] Serpanos D. y Wolf M. "Internet-of-Things (IoT) Systems," Atlanta, GA: Springer International Publishing, 2018.

[15] Martínez M. C., Sauro del Valle S., Brox P. y Sánchez S., "Hardware Implementation of Authenticated Ciphers for Embedded Systems," IEEE LATIN AMERICA TRANSACTIONS, vol. 18, nº 9, pp. 1581-1591, 2020.

[16] Sanaz A. and Khairul A. B., "Lightweight Security Mechanism over MQTT Protocol for IoT Devices" International Journal of Advanced Computer Science and Applications (IJACSA), 11(7), 2020. http://dx.doi.org/10.14569/IJACSA.2020.0110726.

[17] Dobraunig, C., Eichlseder, M., Mendel, F., & Schläffer, M. 2021. "Ascon PRF, MAC, and Short-Input MAC". IACR Cryptol. ePrint Arch., 2021, 1574