



UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE INGENIERÍA EN TELECOMUNICACIONES

**TRABAJO DE GRADO PREVIO A LA OBTENCIÓN DEL TÍTULO DE
INGENIERÍA EN ELECTRÓNICA Y REDES DE COMUNICACIÓN**

TEMA:

**“ESQUEMA DE AUTENTICACIÓN PARA REDES DE SENSORES (WSN) BASADA
EN TECNOLOGÍA DE CADENA DE BLOQUES Y TANGLE”**

AUTOR: CARLA IBETH VALLEJO ROSERO

DIRECTOR: MSC. FABIÁN GEOVANNY CUZME RODRÍGUEZ

ASESOR: MSC. HERNÁN MAURICIO DOMÍNGUEZ LIMAICO

Ibarra-Ecuador

2023



UNIVERSIDAD TÉCNICA DEL NORTE

BIBLIOTECA UNIVERSITARIA

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

1. IDENTIFICACIÓN DE LA OBRA

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

| DATOS DE CONTACTO | | | |
|-----------------------------|---|------------------------|------------|
| CÉDULA DE IDENTIDAD: | 0401875257 | | |
| APELLIDOS Y NOMBRES: | Vallejo Rosero Carla Ibeth | | |
| DIRECCIÓN: | Carchi-Bolívar -Barrio la Esperanza-Vía San Joaquín | | |
| EMAIL: | civallejo@utn.edu.ec | | |
| TELÉFONO FIJO: | 062287-618 | TELÉFONO MÓVIL: | 0987766387 |

| DATOS DE LA OBRA | |
|--------------------------------|--|
| TÍTULO: | ESQUEMA DE AUTENTICACIÓN PARA REDES DE SENSORES (WSN) BASADA EN TECNOLOGÍA DE CADENA DE BLOQUES Y TANGLE |
| AUTOR (ES): | Vallejo Rosero Carla Ibeth |
| FECHA: DD/MM/AAAA | 15/05/2023 |
| SOLO PARA TRABAJOS DE GRADO | |
| PROGRAMA: | <input checked="" type="checkbox"/> PREGRADO <input type="checkbox"/> POSGRADO |
| TÍTULO POR EL QUE OPTA: | Ingeniera en Electrónica y Redes de Comunicación |
| ASESOR /DIRECTOR: | MSc. Fabian Cuzme |

CONSTANCIAS

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

Ibarra, a los 16 días del mes de mayo de 2023

LA AUTORA:



Carla Ibeth Vallejo Rosero
0401875257



UNIVERSIDAD TÉCNICA DEL NORTE
FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CERTIFICACIÓN

MAGÍSTER FABIAN GEOVANNY CUZME RODRIGUEZ, CON CÉDULA DE IDENTIDAD Nro. 1311527012, DIRECTOR DEL PRESENTE TRABAJO DE TITULACIÓN CERTIFICA:

Que, el presente trabajo de Titulación denominado: “ESQUEMA DE AUTENTICACIÓN PARA REDES DE SENSORES (WSN) BASADA EN TECNOLOGÍA DE CADENA DE BLOQUES Y TANGLE”, ha sido desarrollado por la señorita Vallejo Rosero Carla Ibeth bajo mi supervisión.

Es todo en cuanto puedo certificar en honor a la verdad.

A handwritten signature in blue ink, appearing to read "Fabián Geovanny Cuzme Rodríguez", is written over a horizontal line.

MSc. Fabián Geovanny Cuzme Rodríguez

C.C. 1311527012

DIRECTOR

DEDICATORIA

Esta dedicatoria está dirigida a todas las personas que, gracias a sus conocimientos y apoyo, me han ayudado a alcanzar mi meta.

En particular, quisiera agradecer a mis padres, Amilcar Vallejo y Rosa Rosero, quienes con su arduo trabajo y sabios consejos me han guiado en mi lucha por alcanzar mis objetivos e inculcándome la importancia de recordar siempre mis raíces y valores independientemente del futuro que se me presente.

También deseo expresar mi gratitud a mi familia, quienes siempre han tenido fe en mí y me han brindado palabras de aliento para mantenerme motivada.

A mis amigos, quienes me han ayudado a crecer tanto personal como profesionalmente, y a todas aquellas personas que he conocido a lo largo de mi vida, quienes con su sabiduría y honestidad me han permitido aprender de mis errores y mejorar cada día como ser humano.

Carla Ibeth Vallejo Rosero

AGRADECIMIENTO

Quiero expresar mi profunda gratitud a mi familia por su guía, apoyo, enseñanzas y esfuerzo que me ayudaron a perseverar en cada uno de los semestres de mi carrera. Agradezco en particular a mis padres y hermanos por darme el valor necesario para no rendirme ante los desafíos académicos

A mis padrinos Fredy Godoy, Angelica Vallejo y Patricia Vallejo que con sus consejos y apoyo me han permitido mejorar como persona y cumplir mis objetivos.

A mi abuelita, tías y novio que siempre estuvieron para mí en todo momento, con su apoyo incondicional, sembrando valores de honestidad, puntualidad, respeto y humildad que me han permitido mejorar en el camino y en el transcurso de mi vida.

No puedo olvidar a mis amigos y compañeros, Alejandro Ortega, Johanna Pozo, María Jose Cauja, William Ortega, Edwin Solano, Angel Velazco, Armando Chandi, Marlon Moncayo, Javier Aguilar y Gabriel Pozo quienes compartieron conmigo momentos inolvidables y me brindaron su apoyo, palabras de aliento y enseñanzas valiosas que me ayudaron a superar las dificultades que enfrenté en mi carrera.

También, quiero agradecer a mis profesores, Msc. Omar Oña, Msc. Stefany Flores, Msc. Luis Suarez y Msc. Alejandra Pinto, por su asesoramiento y orientación en la toma de decisiones cruciales en mi trayectoria estudiantil, lo que me ayudó a mejorar mi mentalidad y enfoque logrando así alcanzar mis objetivos académicos y a la Sra. ex secretaria de la carrera Silvita Molina por su ayuda desinteresada que me brindo cuando yo más lo necesite.

Finalmente, a mi director Msc. Fabian Cuzme y a mi Asesor Msc. Mauricio Domínguez por su disposición para responder a mis preguntas, brindarme retroalimentación y motivarme en momentos de incertidumbre ha sido fundamental en mi formación como estudiante y profesional. Su compromiso con mi desarrollo académico y personal es algo que siempre valoraré y apreciaré.

RESUMEN

Esta tesis se enfoca en la seguridad de las redes de sensores inalámbricos (WSN), las cuales son cada vez más utilizadas en nuestra vida diaria y en una amplia variedad de aplicaciones. A pesar de su utilidad, las limitaciones inherentes de los dispositivos que componen estas redes hacen que la seguridad siga siendo un desafío importante. Para solucionar este problema, se realizó el estudio de dos tecnologías emergentes como son la Cadenas de bloques (Blockchain) y Tangle que nos han permitido crear un esquema de autenticación seguro para las WSN. En el estudio se analizaron las limitaciones y desafíos de la aplicación de estas tecnologías en los entornos WSN, y se presentaron posibles soluciones para garantizar la escalabilidad y seguridad de las redes. Cabe destacar que la aplicación de esquemas de autenticación en WSN puede presentar un gran desafío debido a las limitaciones de los dispositivos, como la capacidad de procesamiento, la memoria y la vida útil de la batería.

Uno de los problemas mayores en la aplicación de esquemas de autenticación en las WSN es el equilibrio entre la seguridad y la eficiencia. Los esquemas de autenticación suelen requerir cálculos criptográficos complejos que consumen una gran cantidad de recursos de computación. En una WSN, donde los dispositivos son de baja potencia y la transmisión de datos es limitada, la realización de estos cálculos puede ser prohibitiva en términos de consumo de energía y uso de recursos de la red. Además, los esquemas de autenticación complejos también pueden aumentar el costo de los dispositivos y la complejidad de la implementación.

Además, se examinaron diferentes técnicas de autenticación existentes y se compararán con los esquemas de autenticación propuestos basados en Cadena de bloques (Blockchain) y Tangle. En definitiva, se explorará cómo la combinación de estas tecnologías puede ayudar a mejorar la seguridad y privacidad de los datos

transmitidos en las redes de sensores inalámbricos, abriendo nuevas oportunidades para su uso en aplicaciones críticas.

La solución que se propone en este trabajo de investigación es la creación de un nuevo esquema de autenticación que permita tomar las características de la prueba de trabajo de Tangle, así como también su aumento de peso propio para mejorar su seguridad, creando así un canal seguro para el envío de las transacciones. Todas ellas se almacenarán en las estaciones base. Para evitar que la red se vuelva centralizada, se creó una cadena de bloques entre las estaciones base, de esta manera se pudo manejar una copia de la cadena principal en todas las estaciones base y evitar la pérdida de información. Así mismo, se asegurará la seguridad de la información, ya que la tecnología de cadena de bloques se destaca por su inmutabilidad, tras la aplicación del esquema de autenticación se observó que a mayor cantidad de nodos e islas existe un aumento en los tiempos de simulación, además de existir una gran resistencia a ataques de diccionario y DoS.

ABSTRACT

This thesis focuses on the security of wireless sensor networks (WSN), which are increasingly used in our daily lives and a wide variety of applications. Despite their utility, the inherent limitations of the devices that make up these networks make security an important challenge. To address this problem, two emerging technologies, Blockchain and Tangle, were studied, allowing us to create a secure authentication scheme for WSN. The limitations and challenges of applying these technologies in WSN environments were analyzed, and possible solutions were presented to ensure the scalability and security of the networks. It should be noted that applying authentication schemes in WSN can present a major challenge due to device limitations such as processing capacity, memory, and battery life. One of the major problems in the application of authentication schemes in WSN is balancing security and efficiency. Authentication schemes often require complex cryptographic calculations that consume a large amount of computing resources. In a WSN, where devices are low-powered and data transmission is limited, performing these calculations can be prohibitive in terms of energy consumption and network resource usage. Additionally, complex authentication schemes can also increase device cost and implementation complexity. Different existing authentication techniques were examined and compared with the proposed authentication schemes based on Blockchain and Tangle. Ultimately, the combination of these technologies was explored to improve the security and privacy of data transmitted in wireless sensor networks, opening new opportunities for their use in critical applications. The solution proposed in this research work is the creation of a new authentication scheme that takes advantage of Tangle's proof-of-work characteristics as well as its self-weight increase to improve security, thus creating a secure channel for sending transactions. All transactions will be stored at the base

stations. To prevent the network from becoming centralized, a blockchain was created between the base stations, allowing for a copy of the main chain to be maintained at all base stations and avoiding the loss of information. Furthermore, the security of the information will be ensured, as blockchain technology is known for its immutability. After the authentication scheme was applied, an increase in simulation times was observed with higher numbers of nodes and islands, as well as resistance to dictionary and DoS attacks.

ÍNDICE DE CONTENIDOS

| | |
|--|----|
| DEDICATORIA | 4 |
| AGRADECIMIENTO | 6 |
| RESUMEN | 7 |
| CAPITULO I: ANTECEDENTES | 22 |
| 1.1. Introducción al Capítulo I | 22 |
| 1.2. Planteamiento del Problema | 22 |
| 1.3. Objetivos | 25 |
| <i>1.3.1. Objetivo General</i> | 25 |
| <i>1.3.2. Objetivos Específicos</i> | 25 |
| 1.4. Alcance | 25 |
| 1.5. Justificación | 28 |
| CAPITULO II: FUNDAMENTO TEÓRICO | 31 |
| 2.1. Seguridad en redes y su importancia | 31 |
| <i>2.1.1. Escenarios de Posibles Ataques</i> | 32 |
| 2.2. Redes De Sensores Inalámbricas (WSN) | 34 |
| <i>2.2.1. Arquitectura</i> | 35 |
| <i>2.2.2. Tipos de Redes de Sensores Inalámbricos</i> | 37 |
| 2.3. Criptográfica , Cadena de Bloques y Tangle | 40 |
| <i>2.3.1. Cadena de Bloques, concepto y funcionamiento</i> | 40 |
| <i>2.3.2. Tipos de Cadenas de Bloques</i> | 47 |
| <i>2.3.3. Características de la Cadena de Bloques</i> | 49 |

| | |
|--|----|
| 2.3.4. <i>Mecanismo de Autenticación para Cadena de Bloques</i> | 51 |
| 2.3.5. TANGLE..... | 51 |
| 2.3.6. Protocolos y elementos de Seguridad en el Esquema de Autenticación | |
| 55 | |
| 2.4. Seguridad en Redes de Sensores Inalámbricos | 60 |
| 2.4.1. Autenticación | 60 |
| 2.4.2. Dificultades de la seguridad en redes de Sensores | 62 |
| 2.5. Trabajos Relacionados | 63 |
| Una autenticación potenciada por Blockchain esquema para la detección de | |
| gusanos en la red inalámbrica red de sensores | 63 |
| Una identidad híbrida basada en BlockChain Esquema de autenticación para | |
| Multi-WSN | 64 |
| Un esquema escalable basado en Blockchain para intercambio de datos | |
| relacionados con el tráfico en VANET | 65 |
| 3. CAPITULO III: Definir el Esquema de Autenticación | 67 |
| 3.1. Metodología de Diseño | 67 |
| 3.2. Fase 1: Requisitos y Requerimientos | 68 |
| 3.2.1. <i>Situación Actual y Tipo de Investigación</i> | 69 |
| 3.2.2. <i>Descripción general del Sistema</i> | 70 |
| 3.2.3. <i>Diagrama de Bloques del Sistema</i> | 73 |
| 3.2.4. Stakeholders del Proyecto | 84 |
| 3.2.5. Elección de Hardware para simulación | 88 |

| | |
|--|-----|
| 3.2.6. Elección de Software para Simulación | 90 |
| 3.3. Fase 2: Diseño del Sistema | 91 |
| 3.3.1. Diagrama de Bloques del Sistema | 91 |
| 3.3.2. Diagrama de Funcionamiento del Sistema | 92 |
| 3.3.3. Diagrama Lógico de la Red | 95 |
| 3.4. Fase 3: Implementación de la Red | 96 |
| 3.4.1. Inicialización de nodos | 97 |
| 3.4.2. Mensajes Indicadores de procesos | 97 |
| 3.4.3. Caso updatePASS | 98 |
| 3.4.4. Petición de Asociación | 101 |
| 3.4.5. Respuesta de Asociación | 102 |
| 3.4.6. Mensaje de Autenticación | 104 |
| 3.4.7. Función powHASH | 105 |
| 3.4.8. Función checkSalt | 107 |
| 3.4.9. ACK de Confirmación | 108 |
| 3.4.10. Mensaje de UPDATEASO | 108 |
| 3.4.11. Función startTX | 110 |
| 3.4.12. Función caseSensor Data | 111 |
| 3.4.13. Función CompeteBlock | 113 |
| 3.4.1. Función caseDataNonce | 114 |
| 3.4.2. Función caseBlockchain | 117 |

| | |
|---|------------|
| 4. CAPITULO IV: PRUEBAS DE FUNCIONAMIENTO Y RESULTADOS | 119 |
| 4.1. Fase 4: Validación y Verificación de los Resultados | 119 |
| 4.1.1. Validación de resultados de funcionamiento | 129 |
| 4.1.1.1.Prueba 1: Verificar el funcionamiento del Esquema de autenticación | 130 |
| 4.1.1.2.Prueba 2: Se evaluará el rendimiento del esquema de autenticación en el contexto del Modelo Urbano de LoRAWAN | 144 |
| 4.1.1.3.Prueba 3: Simulación de ataques de diccionario y denegación de servicio | 150 |
| 4.1.1.4.Prueba 4: Creación de diferentes ambientes (Aumento del número de nodos e islas), para evaluar la escalabilidad de la red. | 159 |
| 4.1.1.5.Prueba 5: Evaluación del comportamiento del esquema propuesto al unirse a una Cadena de Bloques Real | 169 |
| 4.2. DISCUSIÓN | 190 |
| 5. CONCLUSIONES | 192 |
| 6. RECOMENDACIONES | 193 |

Índice de Figuras

| | |
|---|----|
| Figura 1. <i>Arquitectura del Esquema de Autenticación</i> | 28 |
| Figura 2. <i>Esquema de una WSN</i> | 35 |
| Figura 3 . <i>Arquitectura de redes de Sensores</i> | 36 |
| Figura 4 . <i>Topologías de WSN</i> | 37 |
| Figura 5. <i>Plataformas de nodos sensores</i> | 38 |
| Figura 6. <i>WSN para detección de incendios en zonas forestales</i> | 39 |
| Figura 7. <i>Aplicación de una red de sensores para monitorizar gases químico</i> | 39 |
| Figura 8 . <i>Proceso de Funcionamiento de una Cadena de Bloques</i> | 42 |
| Figura 9. <i>Proceso de funcionamiento del algoritmo de consenso POW</i> | 46 |
| Figura 10. <i>Árbol Hash de Merkle</i> | 47 |
| Figura 11 <i>Tipos de Blockchain</i> | 48 |
| Figura 12 <i>Estructura de DAG</i> | 52 |
| Figura 13. <i>Pesos dentro de una transacción</i> | 53 |
| Figura 14 . <i>Diagrama de Flujo de SEECR</i> | 56 |
| Figura 15 . <i>Creación de Contraeñas Randómicas</i> | 58 |
| Figura 16. <i>Salt Criptográfica</i> | 59 |
| Figura 17. <i>Proceso de Interaccion en esquema para la deteccion de gusanos</i> | 64 |
| Figura 18 . <i>Diagrama de Flujo de esquema de autenticacion para Multi WSN</i> | 65 |
| Figura 19. <i>Modelo en Cascada</i> | 68 |

| | |
|---|-----|
| Figura 20 . <i>Diseño General del Sistema</i> | 72 |
| Figura 21 <i>Diagrama de Bloques del Esquema de Autenticación</i> | 73 |
| Figura 22 . <i>Diagrama de secuencia de Envió de mensajes y Autenticación</i> .. | 76 |
| Figura 23 . <i>Aumento de pesos</i> | 78 |
| Figura 24. <i>Establecimiento de la red Tangle</i> | 79 |
| Figura 25 . <i>Diagrama de Flujo del Funcionamiento del protocolo SEECR</i> ... | 80 |
| Figura 26 . <i>Creación del Bloque candidato</i> | 82 |
| Figura 27 <i>Prueba de Trabajo y Creación de Cadena Central</i> | 84 |
| Figura 28. <i>Características LoRaWAN</i> | 89 |
| Figura 29. <i>Diagrama de Bloques del Sistema</i> | 92 |
| Figura 30. <i>Diagrama de Funcionamiento del Sistema</i> | 94 |
| Figura 31 <i>Diagrama lógico de la Red</i> | 96 |
| Figura 32 <i>Inicialización de los nodos</i> | 97 |
| Figura 33 <i>Mensajes indicadores de procesos</i> | 98 |
| Figura 34 <i>Método updatePASS()</i> | 99 |
| Figura 35. <i>Obtención de pass aleatoria</i> | 100 |
| Figura 36. <i>Petición de Asociación</i> | 101 |
| Figura 37 <i>Metodo send ASO</i> | 102 |
| Figura 38. <i>Respuesta de Asociación</i> | 103 |
| Figura 39 <i>Enviamos la respuesta al mensaje de Autenticación</i> | 104 |
| Figura 40 <i>Función caseAUTH</i> | 105 |
| Figura 41 <i>Método powHash</i> | 106 |
| Figura 42 <i>Implementación de la Salt</i> | 107 |
| Figura 43. <i>Código ACK de Confirmación</i> | 108 |
| Figura 44 <i>Condiciones para realizar la asociación de un nodo</i> | 108 |

| | |
|--|-----|
| Figura 45 <i>Código UPDATE ASO</i> | 109 |
| Figura 46 <i>Cálculo de peso actual del nodo</i> | 110 |
| Figura 47 <i>Función StartTX</i> | 111 |
| Figura 48 <i>Creación de Información del Sensor</i> | 113 |
| Figura 49 <i>Código para el minado de la cadena de Bloques</i> | 114 |
| Figura 50 <i>Código para recuperación del mensaje</i> | 116 |
| Figura 51 <i>Datos que se muestran dentro de la Cadena de Bloques</i> | 117 |
| Figura 52 <i>Código para Actualización de la Cadena de Bloques</i> | 118 |
| Figura 53 <i>Ejecución por consola de pruebas unitarias</i> | 120 |
| Figura 54 <i>Test creado para verificación de código de la Cadena de Bloques</i> | 121 |
| Figura 55 <i>Test realizado a la red Tagle</i> | 123 |
| Figura 56 <i>Segunda parte del test realizada a la red Tangle</i> | 124 |
| Figura 57 <i>Creación de conexión y retardo de los módulos</i> | 125 |
| Figura 58 <i>Posiciones de los sensores al establecer la red</i> | 126 |
| Figura 59 <i>Resultado Obtenido al realizar el Test a la Cadena de Bloques</i> .. | 127 |
| Figura 60 <i>Resultados de Test realizado a Red Tangle</i> | 128 |
| Figura 61 <i>Inicialización de las puertas de salida</i> | 131 |
| Figura 62 <i>Obtención de Contraseña dinámica en los nodos y estación base</i> | 132 |
| Figura 63 <i>Mensaje de petición de Asociación a través de consola</i> | 132 |
| Figura 64 <i>Envío de petición de Asociación de los nodos a la Estación Base</i> | 133 |
| Figura 65 <i>Contenido del mensaje de Respuesta de Asociación</i> | 134 |

| | |
|---|-----|
| Figura 66 <i>Envío de Respuesta de Asociación de la Estación base a los nodos</i> | 134 |
| Figura 67 <i>Inicia el proceso de Autenticación</i> | 135 |
| Figura 68 <i>Mensaje de Autenticación y comparación de Hash</i> | 136 |
| Figura 69 <i>Envío de mensaje de Autenticación de los nodos a la Estación base</i> | 136 |
| Figura 70 <i>Mensaje de ACK enviado por consola</i> | 137 |
| Figura 71 <i>Envío de ACK de asociación</i> | 137 |
| Figura 72 <i>Envío por consola del mensaje de Actualización</i> | 138 |
| Figura 73 <i>Envío de mensajes de Actualización en la Simulación</i> | 138 |
| Figura 74 <i>Envío de datos desde los nodos a la estación base</i> | 139 |
| Figura 75 <i>Envío por consola del mensaje de datos</i> | 139 |
| Figura 76 <i>Recepción de transacciones enviadas por los nodos</i> | 140 |
| Figura 77 <i>Minado del Bloque Génesis</i> | 141 |
| Figura 78 <i>Generación de nonce y competencia entre estaciones base</i> | 142 |
| Figura 79 <i>Envío de nonce entre estaciones base</i> | 143 |
| Figura 80 <i>Envío de mensaje de actualización de Blockchain entre las estaciones base</i> | 143 |
| Figura 81 <i>Actualización de Blockchain Principal con el nuevo Bloque Agregado</i> | 144 |
| Figura 82 <i>Valores referenciales programadas dentro del archivo .ini</i> | 146 |
| Figura 83 <i>Fórmula para la obtención de la perdida de trayectoria y coeficiente de perdida</i> | 147 |
| Figura 84 <i>Cálculo de la distancia</i> | 147 |
| Figura 85 <i>Valor de Frecuencia</i> | 148 |

| | |
|---|-----|
| Figura 86 Resultados de la aplicación de las fórmulas de Perdida Logarítmica | 149 |
| Figura 87 Envío de mensajes de Autenticación desde el nodo Atacante a la Estación Base | 151 |
| Figura 88 Envío de Petición de Asociación desde el Atacante a los nodos Sensores..... | 151 |
| Figura 89 Envía mensaje por consola de la Petición de Asociación | 152 |
| Figura 90 Mensaje Respuesta de Asociación de los Nodos a el Atacante | 152 |
| Figura 91 Aplicación del Protocolo SEECR..... | 153 |
| Figura 92 Envío de mensaje de Autenticando del Atacante a los nodos..... | 153 |
| Figura 93 Intentos realizados desde el Atacante a los nodos sensores | 154 |
| Figura 94 Indicamos que las estaciones base no lo validan | 155 |
| Figura 95 Diagrama de secuencia del envío de mensajes | 155 |
| Figura 96 Envío de mensaje informativo de la existencia de un nodo malicioso | 156 |
| Figura 97 Informamos el ID del nodo atacante | 156 |
| Figura 98 Envío de Respuesta de Asociación del nodo a el Atacante | 157 |
| Figura 99 Inicio de Ataque DOS..... | 157 |
| Figura 100 Mensaje de respuesta de Asociación | 158 |
| Figura 101 Realiza diferentes ataques | 158 |
| Figura 102 Diagrama de secuencia de Ataque DOS | 159 |
| Figura 103 Ambiente con cuatro islas | 160 |
| Figura 104 Tiempo de Inicio de Obtención de Contraseñas..... | 161 |
| Figura 105. Tiempo de Inicio de Envío de Transacciones | 162 |
| Figura 106 Cadena de Bloques actualizada entre las diferentes islas. | 163 |

| | |
|---|-----|
| Figura 107 <i>Escenario con una isla aumentada nodos</i> | 164 |
| Figura 108 <i>Corriente Consumida en los nodos</i> | 165 |
| Figura 109 <i>Comparación de pesos en las estaciones base</i> | 166 |
| Figura 111 <i>Desactivación de las Islas</i> | 167 |
| Figura 112 <i>Control para desactivación de Islas</i> | 168 |
| Figura 113 <i>Mensaje informativo de que se a excedido el mínimo de islas desactivadas</i> | 169 |
| Figura 114 <i>Creación del objeto Block</i> | 170 |
| Figura 115 <i>Clase Blockchain</i> | 171 |
| Figura 116 <i>Métodos que componen la clase Blockchain</i> | 173 |
| Figura 117 <i>Métodos de la clase Blockchain</i> | 174 |
| Figura 118 <i>Definición de funciones de la cadena de Bloques</i> | 175 |
| Figura 119 <i>Función de consenso</i> | 177 |
| Figura 120 <i>Código para la administración de transacciones</i> | 178 |
| Figura 121 <i>Simulación de Red Tangle en computadoras</i> | 179 |
| Figura 122 <i>Escenario Red Tangle 1 Isla</i> | 180 |
| Figura 123 <i>Archivos csv. que almacenan las transacciones</i> | 181 |
| Figura 124 <i>Transformación de datos .csv a formato Python</i> | 181 |
| Figura 125 <i>Recuperación de información de la función mine_unconfirmed_transactions ()</i> | 183 |
| Figura 126 <i>Incorporación de IPs de las diferentes islas a el código Python</i> | 184 |
| Figura 127 <i>Dirección IP y puerto de la página web que mostrara cada uno de los bloques de la Cadena</i> | 184 |
| Figura 128 <i>Archivos para la visualización y diseño de página web</i> | 185 |

| | |
|--|-----|
| Figura 129 <i>Página web de la Cadena de Bloques</i> | 186 |
| Figura 130 <i>Bloque Génesis</i> | 186 |
| Figura 131 <i>Elementos del Bloque Génesis</i> | 187 |
| Figura 132 <i>Aumento de Bloques en la Cadena Principal</i> | 188 |
| Figura 133 <i>Proceso de Consenso</i> | 189 |
| Figura 134 <i>Detalle de los Bloques que se agregan a la cadena principal</i> ... | 190 |

Índice de Tablas

| | |
|---|-----|
| Tabla 1. <i>Algoritmos de Consenso usados en Blockchain</i> | 43 |
| Tabla 2 <i>Características de TI</i> | 50 |
| Tabla 3. <i>Tipos de Autenticación y Características</i> | 61 |
| Tabla 4. <i>Actores Involucrados</i> | 85 |
| Tabla 5. <i>Nomenclatura de los requerimientos</i> | 86 |
| Tabla 6. <i>Requerimientos de Stakeholders</i> | 86 |
| Tabla 7. <i>Requerimientos de Sistema</i> | 87 |
| Tabla 8. <i>Requerimientos de Arquitectura</i> | 88 |
| Tabla 9. <i>Selección de Hardware para el nodo sensor</i> | 89 |
| Tabla 10. <i>Elección de Plataforma de Simulación</i> | 90 |
| Tabla 11 <i>Explicación de Patrón de tres A</i> | 119 |
| Tabla 12 <i>Listado de Pruebas realizadas</i> | 129 |
| Tabla 13 <i>Valores referenciales usados en el modelo Hot Zone</i> | 146 |
| Tabla 14 <i>Campos del método to_dict</i> | 171 |
| Tabla 15 <i>Métodos necesarios para el funcionamiento de la Blockchain</i> | 172 |

CAPITULO I: ANTECEDENTES

1.1. Introducción al Capítulo I

En este capítulo se hace una descripción del problema, objetivos, alcance y justificación, donde se sustenta la importancia de desarrollar el presente proyecto.

1.2. Planteamiento del Problema

En los últimos años se ha visto como las redes de sensores inalámbricos (WSN) han logrado introducirse dentro de diferentes campos tecnológicos, alcanzando a establecerse como una infraestructura elemental para el desarrollo del Internet de las cosas (IoT), los nodos de una WSN tienen características de bajo consumo, costo y tamaño que cooperan entre sí para realizar el monitoreo del entorno en el cual se encuentran dispersos por medio de transceptores de baja potencia; a diferencia de “las redes inalámbricas convencionales que usan un patrón de comunicación extremo a extremo las redes WSN trabajan bajo un patrón dominante de comunicación muchos a uno lo cual provoca que muchos nodos comuniquen sus lecturas de sensores hasta una estación base central” (Segovia Ferreira, 2014,p. 69). o en el caso de la “topología de malla, cada nodo puede actuar como relé para otros nodos, donde si un enlace se congestiona o falla un nodo, los datos se desviarán automáticamente a través de rutas alternativas” (Guy, 2006,p. 2). Este tipo de comunicación a provocado varios problemas de seguridad y privacidad.

Las WSNs se encuentran expuestas a diferentes amenazas debido a que usan un medio de transmisión broadcast y los nodos se ubican en medios hostiles, lo cual dificulta la implementación de un esquema de seguridad adecuado. Chenyu et al. (2020) afirma que “Se ha intentado crear protocolos de autenticación de usuario para WSN, pero la mayoría de las propuestas han demostrado ser inseguras o incapaces de

proporcionar atributos de seguridad importantes como lo son; el uso de una password Friendly, la autenticación mutua y el mal manejo de tablas verificadoras de contraseñas” (p. 3). Según indica Benenson et al. (2005) cuando se introdujeron por primera vez ataques de captura de nodos en los esquemas de autenticación de usuarios remotos, facilito a los atacantes conseguir comprometer algunos de los nodos sensores y llevar a cabo una serie de ataques posteriores. Después de esto, los ataques de captura de nodos comienzan a ser aceptados como una forma práctica de vulnerar los esquemas de autenticación de usuarios para WSN.

La seguridad en las WSNs es un campo que ha generado varias investigaciones teniendo como resultado el desarrollo de varios esquemas de autenticación simétricos y asimétricos los cuales tienen algunas limitaciones como autenticación retrasada, mala distribución de compromisos de cadena clave y alto consumo de energía (Arreaga,2020). Desde este punto se ve la necesidad de adaptar mecanismos de autenticación que permitan, asegurar que los nodos se autenticuen antes de realizar un envío de datos , ya que las nuevas tecnologías como las cadenas de bloques y Tangle en los últimos años han mostrado un gran potencial en diversas aplicaciones ; esto se debe a que “las transacciones en una red de cadena de bloques se agrupan en bloques y se ejecutan en todos los nodos participantes por lo cual cada bloque contiene una lista de transacciones, el estado más reciente, un número de bloque y un valor de dificultad “(Bahga & Madiseti, 2016, p. 1). Las cadenas de bloques también cuenta con una base de datos segura la cual hace uso de diferentes algoritmos hash para garantizar la integridad de los datos, consiguiendo así controlar la singularidad de cada bloque ,de tal forma que si se determinaran anomalías , el bloque se volvería invalido de forma inmediata ; también hace uso de contratos inteligentes que se presentan como un fragmento de código, la cual se identifica con una dirección única , permitiendo así la

verificación de nuevos bloques (Bahga & Madiseti, 2016). Una tecnología que ha dado también mucho que hablar es Tangle, que se basa en el funcionamiento de un “gráfico acíclico dirigido (DAG) en donde cada vértice es una transacción enviada a la red y los enlaces dirigidos representan una verificación de transacciones” (Bu et al., 2019, p.645), para esto el DAG maneja “un peso acumulativo para cada transacción, definiéndose como el propio peso de una transacción particular más la suma de los pesos de todas las transacciones que aprueben directa o indirectamente esta transacción” (Popov et al., 2019). Según Silvano & Marcelino (2020) “Esto se da debido a que un nodo elige otras dos transacciones para aprobar, a través del manejo de un algoritmo de selección Markov Chain Monte Carlo (MCMC), logrando así que el nodo compruebe si las dos transacciones están, o no, en conflicto; para que el nodo emita una transacción válida, primero se debe resolver un tipo de prueba de trabajo (PoW), con el cual su hash está concatenado revisando así algunos datos de las transacciones aprobadas anteriormente” (p.308).

La autenticación es una operación clave que se realiza para evitar la participación de entidades externas no autorizadas en la red. En 2015, Cuzme indica en su investigación sobre Internet de las Cosas (IoT) y las consideraciones de seguridad que “Las implicancias de no comprender la importancia en la seguridad o privacidad podrían ser devastador en el diseño de una arquitectura IoT” (p.65), considerando esta afirmación en la aplicabilidad de las WSNs, se busca la implementación y combinación de tecnologías emergentes como Blockchain y Tangle para su aplicación en el campo de seguridad de las WSN, lo cual permitirá el desarrollo de nuevas aplicaciones ya que cuentan con un “tipo de estructura de datos flexible y de amplio alcance que operan bajo los principios de una tecnología de libro mayor distribuido por lo que permite grabar y compartir datos” (Rocamora & Amellina, 2018, p.11). Consiguiendo de esta

manera brindar así una mayor seguridad en autenticación al hacer uso de contratos inteligentes. El presente esquema que sea resultado de esta investigación busca ser considerado y aplicable en el desarrollo de futuros proyectos relacionados a las WSNs de la academia.

1.3. Objetivos

1.3.1. Objetivo General

Establecer un esquema de autenticación basado en cadena de bloques y Tangle para redes de sensores inalámbricos (WSN), mediante pruebas de verificación y validación.

1.3.2. Objetivos Específicos

- Revisar el contexto de diferentes investigaciones realizadas sobre las redes de sensores inalámbricos, las tecnologías basadas en cadena de bloques y Tangle, su arquitectura y funcionamiento.
- Definir un esquema de autenticación de nodos inalámbricos que se base en el funcionamiento de Tangle y la seguridad que proporcionan las cadenas de bloques
- Establecer un escenario de simulación que permita realizar las pruebas de funcionamiento del esquema de autenticación realizado.
- Realizar pruebas de verificación y validación que permitan demostrar la utilización del esquema propuesto en despliegues de WSNs de la academia.

1.4. Alcance

El alcance del estudio contempla investigar el entorno de las tecnologías basadas en cadena de bloques y Tangle, su valor estratégico y el enfoque del funcionamiento del libro mayor distribuido; que cada una de estas tecnologías maneja, lo cual permitirá identificar la forma en cómo se permite a los usuarios, obtener un consenso eventual sobre el estado del libro mayor, de manera descentralizada. En su primera etapa se realizara un análisis de las características de las redes de sensores inalámbricos (WSN), en la cual se

indagara el nivel de protección con el que cuentan cada uno de los dispositivos que la componen; una característica importante a considerar en estas redes es la capacidad energética, cuya dependencia está marcada por el consumo realizado por el nodo sensor durante la comunicación; permitiendo así determinar el tiempo de vida de la batería que realiza el suministro de energía a la red, otro aspecto que se debe tener presente es la comunicación que manejan las redes (WSN) la cual es de corto alcance con un canal estrecho de banda ancha. Al ser una red con alta aplicabilidad se debe examinar su tolerancia a fallos, errores y la capacidad de autocomprobarse y auto calibrarse, permitiendo así determinar los riesgos y restricciones que se deben tener presentes al realizar un esquema de autenticación.

El esquema de autenticación que se busca crear deberá contar con una alta eficiencia energética que permita manejar la movilidad de los nodos y sus rutas cambiantes, otro aspecto importante es la escalabilidad de la red que permitirá gestionar una gran cantidad de nodos sin provocar una sobrecarga excesiva. La capacidad de respuesta de la red no debe verse afectada por los parámetros de escalabilidad y latencia debido a que el tiempo que debe tomar cada nodo en enviar una respuesta debe ser relativamente bajo, para poder realizar una autenticación óptima entre nodos.

El diseño del esquema de autenticación se basará en el uso de una estación base la cual será la encargada de realizar la autenticación de los primeros nodos que formen parte de la red, de ahí en adelante se buscará generar un modelo de Grafo acíclico dirigido con el cual el primer nodo que forme parte de la red será el encargado de encontrar el hash correcto, que indique que los nodos que intentan formar parte de la red son auténticos, después de que se haya realizado este proceso, los nodos que deseen formar parte de la red deberán ser autenticados por dos nodos anteriores que ya formen parte de la red y que cuenten con un alto nivel de confianza, logrando así crear un tipo de malla; en la cual los

nodos que se encuentren en una altura y profundidad más cercana al final de la red deberá encargarse de la autenticación del resto de nodos.

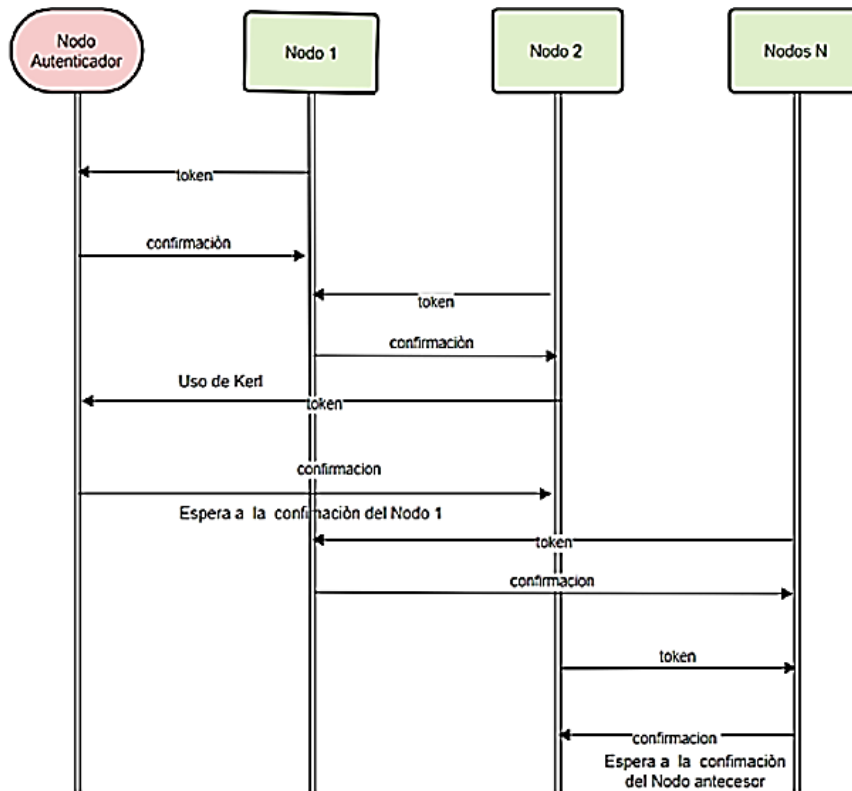
Para lograr observar el comportamiento del esquema a desarrollar se pretende hacer uso de diferentes herramientas de simulación; que permitan crear en primera instancia un ambiente ideal, el cual cuente con una cantidad mínima de nodos, donde cada nodo podrá recibir una rápida respuesta de la estación base y en segunda instancia un ambiente hostil en el cual se exponga a la red WSN a diferentes interferencias provocadas por un dispositivo desconocido, el cual al emitir señales de radio en la misma frecuencia que la red este usando para su comunicación, provocara que la capacidad de respuesta entre la estación base y los nodos tome mayor tiempo.

Los resultados obtenidos de las diferentes simulaciones permitirán observar el comportamiento del esquema de autenticación tanto en ambientes ideales como hostiles en consecuencia se podrá determinar si el esquema planteado puede usarse en diversos ambientes y bajo ciertas condiciones. La verificación y validación de resultados se lo realizará en base a la ejecución de pruebas unitarias a través de la aplicación de las 3A's del Unit Testing logrando así realizar una definición correcta de requisitos en su primera etapa se definirán los requisitos que debe cumplir el código de cada uno de los nodos y la estación base que formaran parte de la red, en la fase de actuación se realizara un test de comprobación el cual permitirá observar los diferentes resultados de la simulación y así dar paso a la realización de un análisis y por último se aplicara la fase de afirmar dentro de la cual se realizara la comprobación de los resultados obtenidos tras la ejecución del esquema de autenticación propuesto, consiguiendo así comprobar si los resultados que se recolecten estén acorde a lo que se planteó dentro de la investigación para lo cual se generan diferentes pruebas de rendimiento que nos permitan determinar la latencia,

capacidad de respuesta y tiempo que se demora en realizar la autenticación el esquema propuesto.

Figura 1.

Arquitectura del Esquema de Autenticación



1.5. Justificación

En la actualidad el tema de la seguridad dentro de las redes de sensores inalámbricos (WSN) se ha vuelto muy controversial, debido a la gran cantidad de limitantes que poseen estas redes, Roman et al. (2005) indica que cualquier adversario puede acceder a la información procedente de una red de sensores, debido a que los nodos están (normalmente) distribuidos en un entorno de fácil acceso, y los canales de comunicación inalámbricos son inherentemente inseguros.

Dentro de los criterios de seguridad que se tiene en la actualidad para las redes WSN la autenticación es un factor clave, ya que “permite asegurarse de que el inicio de un mensaje que se comunica de un nodo a otro nodo es correctamente identificado “ (Vikas et al, 2021, p.1419) consiguiendo así evitar dentro de la capa de red ataques como Backhole, Ataque Sybil, Sinkhole, replicación de nodos e inundación de paquetes Hello.

Debido a los diferentes campos en los que se busca implementar las redes WSN, se ha intentado crear diferentes métodos para conseguir una autenticación segura entre los nodos y así poder realizar el envío de datos. Según Riaz et al., (2019) “debido a que es difícil para un nodo sensor realizar una autenticación segura puede ocurrir la fuga de información sensible además de un agotamiento innecesario de los recursos de la red, como son la potencia del nodo y el ancho de banda de la red”(p.2) pero muchas veces estos métodos no han generado los resultados esperados, es aquí donde se pretende realizar la implementación de tecnologías emergentes como lo son las cadenas de bloques y el Tangle que al manejar “una base de datos que es independientemente construida, mantenida y actualizada por cada uno de los nodos que conforman la red”(Živi et al., 2019,p1),nos permitan crear ciertas primitivas de seguridad, en la cual la red después de realizar la autenticación de los nodos que la componen pueda proceder a el envío de datos.

Debido a que parte de los problemas de autenticación y seguridad en las redes (WSN) se ven afectadas por la escalabilidad de la red, se pretende hacer uso de un Grafo acíclico dirigido propio de Tangle “el cual permite que la red se vuelva más rápida a medida que van apareciendo más transacciones, logrando así mantener una cantidad de transacciones por segundo ilimitada.” (Miranda Palacios, 2018, p.44). No se puede asegurar el 100% de seguridad en la red ya que como indica Wasimi (2019) “la seguridad absoluta es inalcanzable tanto en el entorno real como en el virtual, es posible crear un

nivel de seguridad que sea suficientemente adecuado en casi todas las condiciones ambientales” (p.3)

Este trabajo de grado tiene un impacto a nivel académico y tecnológico dentro de lo académico servirá como una base para futuras investigaciones y desarrollo de soluciones para la seguridad de las redes WSN y en la parte tecnológica abrirá las puertas para la implementación de redes WSN en nuevos ambientes mejorando así sus prestaciones y su confiabilidad.

CAPITULO II: FUNDAMENTO TEÓRICO

En este capítulo se realiza la recopilación de la fundamentación teórica que sustenta el desarrollo del presente proyecto. Algunos de los temas relevantes que se mencionaran será la evolución de la seguridad en las redes de sensores inalámbricos, las limitaciones de las redes WSN, esquemas de autenticación implementados en las redes (WSN), funcionamiento de las cadenas de bloques, características de las cadenas de bloques.

2.1.Seguridad en redes y su importancia

La seguridad de la red comienza con la autorización, comúnmente con un nombre de usuario y una contraseña, consiguiendo así la creación de disposiciones y políticas adoptadas por un administrador de red, para prevenir y supervisar el acceso no autorizado, la modificación no consentida del sistema, el mal uso o la denegación de una red informática y los recursos accesibles a la red. (Pawar & Anuradha, 2015,)

Para Xiao & Guo (2020) la seguridad de las redes no se compone de un solo aspecto, sino que contiene cuatro eslabones esenciales siendo su principal reto la protección de hardware, software y recursos de datos en los sistemas informáticos para que no se destruyan, alteren o filtren ya sea por razones accidentales o maliciosas (p.439).

Dentro de la importancia de la seguridad Vangala (2017) indica lo siguiente:

Al considerar la seguridad de la red, se espera que toda la red sea segura. La seguridad de la red no solo afecta a la seguridad en los ordenadores de cada extremo de la cadena de comunicación. Al transmitir datos, el canal de comunicación no debe ser vulnerable al ataque. Un posible hacker podría apuntar al canal de comunicación, obtener los datos, descifrarlo y volver a insertar un mensaje falso.

Asegurar la red es tan importante como asegurar los ordenadores y cifrando el mensaje, para lo cual se desarrolla un plan eficaz de seguridad de la red que comprende los problemas de seguridad, los posibles atacantes, el nivel de seguridad necesario y los factores que hacen que una red sea vulnerable a los ataques. Los pasos involucrados en la comprensión de la composición de una red segura, Internet o de otro tipo, se sigue a lo largo de este esfuerzo de investigación. (p. 6)

2.1.1. Escenarios de Posibles Ataques

Actualmente, existen numerosos tipos de ataques que pueden afectar a una red. Estos ataques pueden ser causados por una variedad de factores, tanto directos como indirectos, y pueden perpetrarse mediante el uso de diversas técnicas y métodos. Estos ataques pueden poner en peligro la seguridad y la privacidad de los datos alojados en la red. Aunque pueden observarse estos ataques en cada una de las capas del modelo OSI y de la arquitectura de redes de sensores, es importante destacar que estos ataques tienen efectos diferentes en cada una de estas capas y arquitecturas como se muestra a continuación:

Capa Física: La interferencia y la manipulación son los principales tipos de ataques físicos. La defensa estándar contra interferencias involucra varias formas de comunicación de espectro ensanchado o salto de frecuencia. Un atacante también puede manipular los nodos físicamente, interrogarlos y comprometerlos. La protección contra manipulaciones cae en dos categorías: pasivo y activo. (X. Chen et al., 2009)

Capa Enlace de Datos : Chen et al. (2009) indica que la colisión, el agotamiento y la injusticia son los principales ataques en esta capa. El código de corrección de errores puede facilitar el ataque de colisión, sin embargo, el resultado es limitado porque los nodos maliciosos aún pueden dañar más datos que la red puede

corregir. La TDMA (Time División Múltiple Access o Acceso Múltiple por División de Tiempo) es otro método para prevenir colisiones. Pero requiere más recursos de control y todavía es susceptible a colisiones. Los adversarios pueden permitir que los nodos del sensor ejecuten un gran número de tareas para agotar la batería de estos nodos. (p. 4)

Capa de Red: Messai (2014) señala que en las redes de sensores inalámbricas (WSNs, por sus siglas en inglés), se utiliza una comunicación de saltos múltiples para enrutar los paquetes hacia su destino. Esta capa de comunicación es vulnerable a varios tipos de ataques, como el agujero negro, el reenvío selectivo, el ataque Sybil, el ataque de inundación HELLO, el agujero de gusano y el ataque de replicación de identidad. Para protegerse de estos ataques, los nodos de la red actúan como "perros guardianes" para supervisar la transmisión del paquete y detectar comportamientos anormales. Si se detecta un comportamiento sospechoso, los nodos actualizan la información de enrutamiento para evitar el uso del nodo comprometido.

Capa de Transporte: Según Sinha et al. (2017), los protocolos de red, UDP y TCP son propensos a varios tipos de ataques de seguridad, como las inundaciones de TCP, las inundaciones de UDP y los ataques de predicción de TCP. El ataque de inundación TCP, también conocido como inundación de ping, se lleva a cabo enviando numerosas solicitudes de ping ICMP a los nodos de una víctima, lo que puede resultar en la inundación de los búferes de entrada y salida de la víctima y retrasar su conexión a la red objetivo. La técnica de predicción de TCP implica predecir el índice de secuencia y crear paquetes desde el nodo transmisor. Por otro lado, el ataque de inundación de UDP se realiza enviando una gran cantidad de paquetes UDP, lo que obliga a los nodos víctimas a enviar un gran número de paquetes de respuesta. (p.4)

Capa de Aplicación Sinha et al. (2017), afirma que la capa de aplicación de una red soporta varios protocolos como HTTP, SMTP y FTP, que se utilizan para habilitar servicios web, transferir correo y transferir archivos, respectivamente. Estos protocolos son vulnerables a diferentes ataques de seguridad. El ataque de malware¹ es un tipo de ataque HTTP que incluye caballos de Troya, gusanos, keyloggers, virus y puertas traseras. El malware es un software malicioso que interrumpe o intercepta los datos confidenciales legítimos (p.4)

2.2.Redes De Sensores Inalámbricas (WSN)

Las redes de sensores inalámbricos por sus siglas en inglés (WSNs) se han popularizado en esta época, lo cual se debe a la convergencia de tecnologías tales como inalámbrica, procesamiento de datos, almacenamiento de información, algoritmos, hardware y las capacidades de los sensores, tal como se expone en (Shahzad, 2013). Las WSNs están conformadas por un grupo de pequeños ordenadores también llamados “nodos” que se comunican de manera inalámbrica y colaboran en una o más tareas comunes. Normalmente los nodos de sensores están compuestos por uno o más dispositivos sensores que monitorizan entornos y envían información a través de la red como humedad, presión, temperatura, velocidad, vibración, entre otros comunicándose dentro de su contexto. (Álvarez, 2018). Sus múltiples usos y aplicaciones han impactado positivamente en diferentes aspectos de la vida y, por consiguiente, en la sociedad. Es utilizado ampliamente en la monitorización ambiental, así como en la biometría, óptica, salud, e incluso, en la monitorización de las estructuras civiles. (Muhammad A. A., 2016)

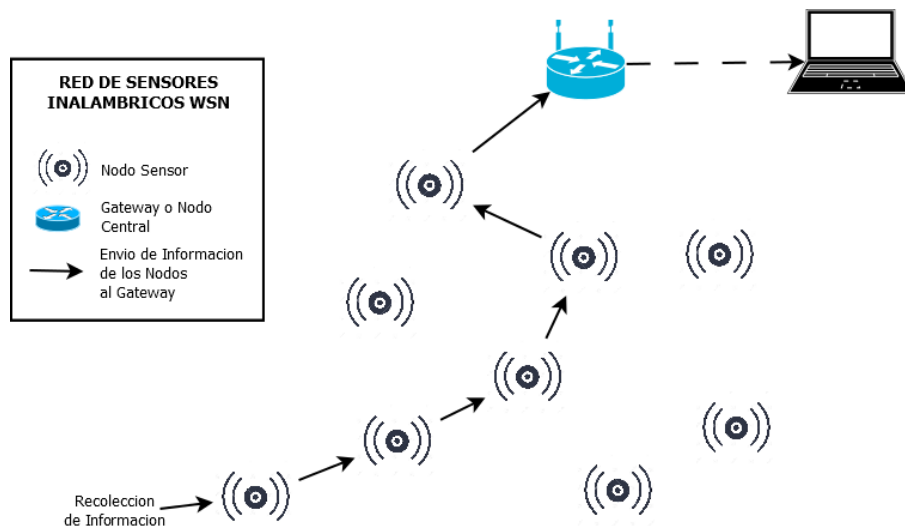
¹ **Malware:** Software Malicioso que realiza funciones en el sistema que son perjudiciales para el usuario

Las redes de sensores inalámbricas se caracterizan por su facilidad de despliegue y por ser auto configurables. Esto significa que pueden convertirse en emisores, receptores u ofrecer servicios de encaminamiento entre nodos sin visión directa en cualquier momento, y también pueden registrar datos referentes a los sensores locales de cada nodo. A pesar de tener limitados recursos, como procesamiento, memoria, consumo de energía y ancho de banda, estos sistemas son muy utilizados y eficientes, y proporcionan datos veraces y confiables.

En la Figura 2, se puede ver cómo se despliega una red de sensores. Después de recopilar información, estos sensores envían los datos entre sí hasta llegar al Gateway, que es el encargado de procesar la información y enviarla a una computadora como dispositivo de almacenamiento. Así, se puede ver cómo se realiza la comunicación entre los sensores y cómo la información llega a su destino final.

Figura 2.

Esquema de una WSN



2.2.1. Arquitectura

La arquitectura de las redes de sensores inalámbricos (WSN) se presenta como una estructura abierta y sencilla que se basa en cinco capas del modelo OSI: la capa de

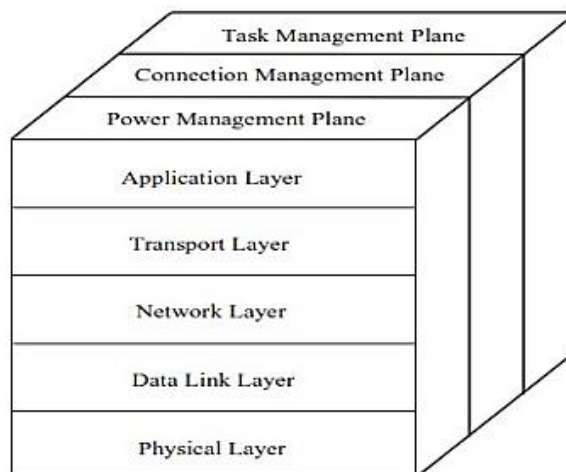
aplicación, la capa de transporte, la capa de red, la capa de enlace de datos y la capa física. Además, estas capas se complementan con tres planos centrados en las necesidades de las redes WSN: el plano de administración de energía, el plano de gestión de conexiones y el plano de gestión de tareas.

Estas capas se utilizan para administrar la conectividad de la red y permiten que los nodos trabajen juntos para aumentar la eficiencia global de la red.”(Kaur et al., 2014, p.6)

La Figura 3 muestra tres planos: el plano de energía, el plano de conexión y el plano de gestión de tareas. El plano de energía se encarga de controlar el nivel de potencia, la detección y la comunicación de los sensores. El plano de conexión se centra en la configuración o reconfiguración de los nodos sensores para mantener la conectividad de la red. El plano de gestión de tareas distribuye las tareas entre los nodos sensores para maximizar la vida útil y la eficiencia energética de la red.

Figura 3

Arquitectura de redes de Sensores



Fuente :(Kaur et al., 2014)

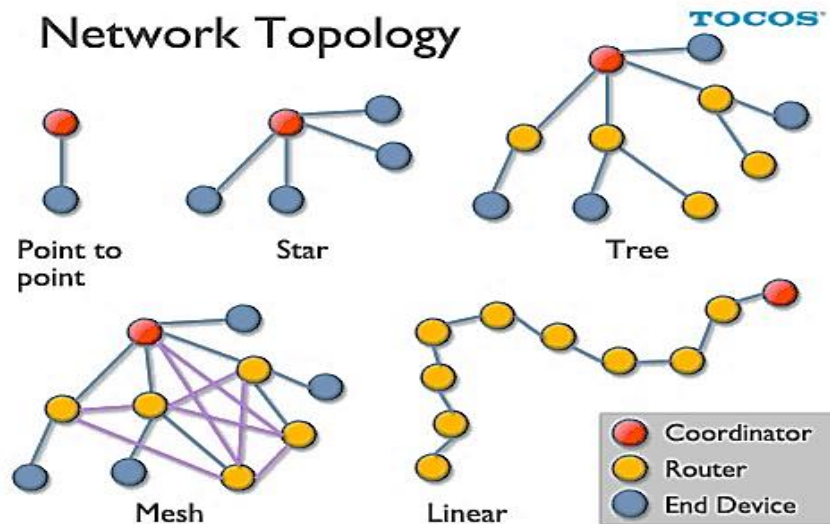
2.2.2. Tipos de Redes de Sensores Inalámbricos

Existen diferentes tipos de redes de sensores, las transmisiones de datos son realizadas, dependiendo de la configuración y complejidad de los componentes (hardware) estos se pueden clasificar según su topología, plataforma o aplicación como se indica a continuación:

Según su Topología: Además de la clásica topología de red mallada de WSN, existen dos topologías más. Una de ellas es la topología de redes en estrella, en la que los nodos inalámbricos se comunican con un dispositivo de pasarela (gateway) que actúa como puente de comunicación con una red cableada. En la Figura 4, se pueden observar diferentes topologías de redes de sensores inalámbricos WSN, mostrando diferentes disposiciones de nodos, elementos y enlaces de conexión. . (Gensei Corporation, 2019).

Figura 4

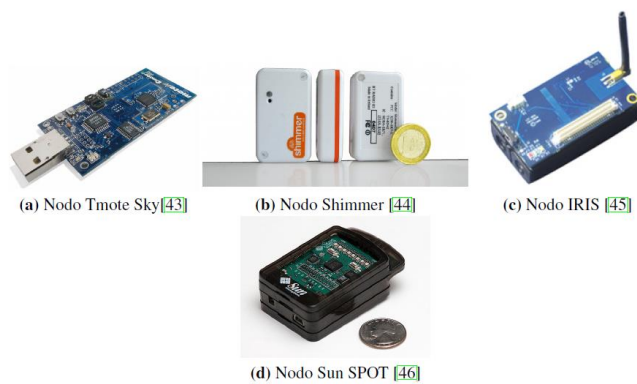
Topologías de WSN



Según su Plataforma: Los nodos de una red de sensores pueden variar en forma y tamaño. Generalmente tienen forma rectangular, con un tamaño aproximado de 5 cm por lado, o forma de disco con un diámetro inferior a 1 cm. En la Figura 5 se muestran algunos ejemplos de nodos sensores, los cuales pueden utilizarse para actividades que requieran comunicación a corta o larga distancia, mayor resistencia a interferencias y otros aspectos, dependiendo de sus tamaños y características.

Figura 5.

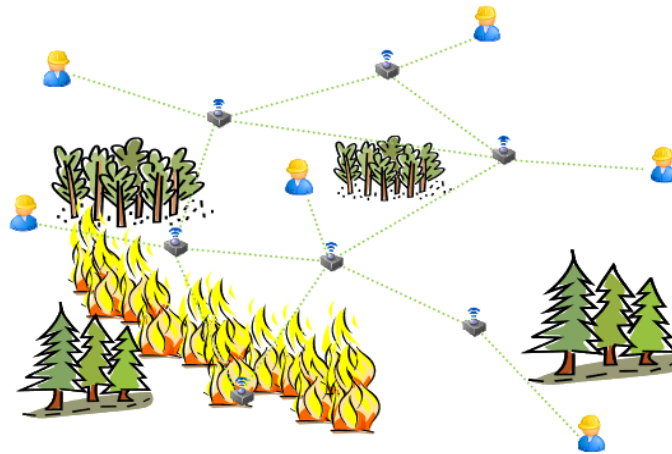
Plataformas de nodos sensores



Según su Aplicación: Las WSN pueden recoger datos de los nodos de tres formas diferentes: captura de eventos, capturas periódicas e informes bajo demanda. En el modo de presentación de informes por captura de eventos, los nodos sensores solo informan de los datos si se produce un evento en su entorno. Un ejemplo de esto se indica en la Figura 6, que muestra una WSN que se despliega en un bosque para controlar los incendios forestales. Los nodos sensores se distribuyen en una topología estrella extendida, y solo reportan a la estación base si se produce un incendio en el bosque. (Segovia, 2014) .

Figura 6

WSN para detección de incendios en zonas forestales



Fuente : (Garcia et al., 2009)

Otras aplicaciones de redes de sensores utilizan la captura bajo demanda, en la que se envían solicitudes de información a los nodos sensores para que envíen sus datos detectados. Un ejemplo de esto se muestra en la Figura 8, que muestra una aplicación que supervisa la presencia de contaminantes químicos en el agua. El nodo principal puede enviar una petición a los nodos de sensores para detectar el nivel de contaminantes y reportar los valores a la estación base cuando se requiera. (Segovia, 2014)

Figura 7.

Aplicación de una red de sensores para monitorizar gases químicos



2.3.Criptográfica, Cadena de Bloques y Tangle

La criptografía en un sistema de redes distribuido, también conocido como Cadena de Bloques, es el lenguaje codificado utilizado por los participantes de la red, principalmente los nodos, para comunicarse, entenderse y llegar a acuerdos de manera segura. Todos los proyectos de Cadenas de Bloques utilizan criptografía para proteger la información y evitar el acceso no autorizado. En resumen, la criptografía en Cadena de Bloques es un medio para garantizar la seguridad y privacidad de la información y la comunicación entre los nodos de la red (Martínez, 2020)

Dentro del desarrollo de nuevas herramientas que brinden seguridad, se encuentra la creación de tecnologías como Cadena de Bloques y Tangle, las cuales buscan facilitar la realización de diferentes transacciones de forma segura, a través del uso de algoritmos y hash, que permiten ejecutar un envío eficiente de los diferentes datos, consiguiendo así generar de forma exitosa una transacción.

El proceso de validación de transacciones en IOTA, también conocido como Tangle, es diferente al utilizado en la cadena de bloques. En lugar de depender de mineros para seleccionar y agregar transacciones a una cadena de bloques, el Tangle utiliza un mecanismo de consenso denominado "validación de pares". Este sistema requiere que cada nueva transacción confirme al menos dos transacciones previas antes de ser aceptada en la red.(Fundación IOTA, 2022)

Este enfoque permite una mayor escalabilidad y eficiencia en comparación con el modelo de minería utilizado en la cadena de bloques ya que no se requiere un gran poder de procesamiento para validar las transacciones.

2.3.1. Cadena de Bloques, concepto y funcionamiento

Una cadena de bloques se puede pensar como una tabla con tres columnas, donde cada fila representa una transacción distinta, la primera columna almacena la

marca de tiempo, la segunda columna conserva los detalles , y la tercera columna guarda el hash de la transacción actual más sus detalles, e incluyen el hash de la transacción anterior.(Di Pierro, 2017)

Musleh et al. (2019) indica que el funcionamiento de una cadena de bloques comienza con una solicitud de transacción; la cual puede ser iniciada por cualquier usuario. A continuación, la transacción se difunde a todos los clientes de la red. Seguidamente, se lleva a cabo el proceso de verificación; en esta etapa todos los nodos se encargan de constatar los intercambios generados a través de los hashes. Una vez finalizada la verificación, la transacción se incluye en un nuevo bloque, el cual se encontrará conectado a la cadena de bloques anterior, lo que la hace permeable e inalterable.

Los algoritmos de consenso utilizados en las redes Cadena de Bloques han experimentado un proceso de evolución en el que se ha buscado mejorar la seguridad de estas redes y resolver el problema de la falla de consenso. Estos algoritmos también se han adaptado de acuerdo con las necesidades específicas del servicio que se ofrece y al tipo de red utilizada.(Martínez, 2020)

El proceso de consenso incluye la verificación de las transacciones antes de que se añadan a la Cadena de Bloques, dando lugar a la creación de diferentes intervalos de tiempo predefinidos discretos, los cuales se encargan de representar los tiempos desde el inicio de las transacciones, hasta el momento de su adición a la cadena de bloques. El tiempo de confirmación dependerá de diferentes factores como son: el tamaño del bloque, los volúmenes de transacción y los algoritmos de consenso utilizados (Musleh et al., 2019).

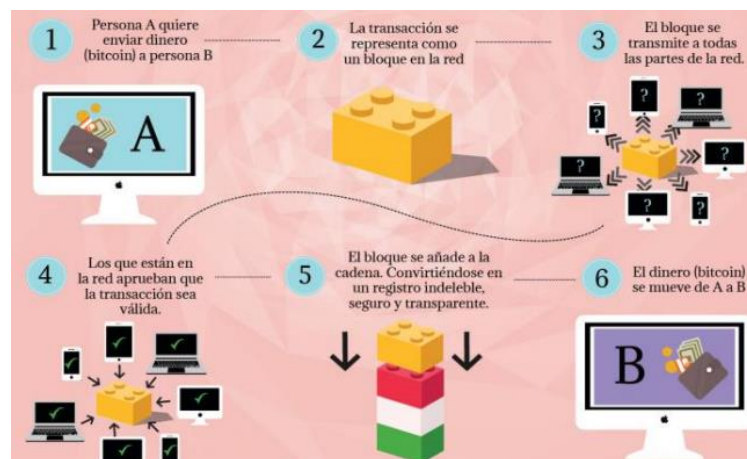
En la Figura 10 se observa que cuando se realiza una transacción, se crea un bloque de datos encriptado que se añade a la cadena de bloques. Esta cadena refleja la

secuencia y el tiempo exacto de las transacciones y garantiza la inmutabilidad de la información almacenada. Cada bloque incluye la huella de la transacción y el usuario que la ha llevado a cabo.

La transacción es verificada por la red de nodos sin la intervención de terceros y, una vez verificada, el bloque se añade de manera permanente a la cadena, creando un registro completamente transparente e inmutable.

Figura 8

Proceso de Funcionamiento de una Cadena de Bloques



2.3.1.1. Algoritmos de Consenso

Cuando se trata de validar cada transacción para evitar la penetración de la Cadena de Bloques, se añade cada uno de los bloques a través de un algoritmo de consenso, los cuales según indica Guo & Yu (2022) “son usados para construir su confianza y almacenar correctamente las transacciones en los bloques, siendo considerados el corazón de todas las transacciones de Blockchain”.

Estos algoritmos determinan cómo se llega a un acuerdo sobre el estado actual de la cadena y cómo se añaden nuevos bloques a la misma. Existen diferentes tipos de algoritmos de consenso los cuales se analizan dentro de la Tabla 2, cada uno con sus

propias características y ventajas. Algunos de los algoritmos de consenso más comunes son:

- Proof of Work (PoW) - Prueba de Trabajo.
- Proof of Stake (PoS) - Prueba de Participación.
- Delegated Proof of Stake (DPoS) – Prueba de Participación Delegada
- Proof of Elapsed Time (PoET) – Prueba de Tiempo Transcurrido.
- Practical Byzantine Fault Tolerance (PBFT) - Tolerancia práctica a fallas bizantinas
- Directed Acyclic Graph (DAG) - Gráfico Acíclico Dirigido

A continuación, se examinan los principales algoritmos de consenso y se proporciona una tabla comparativa de las características que los definen.

Tabla 1.

Algoritmos de Consenso usados en Blockchain

| ALGORITMOS DE CONSENSO | | | | | | |
|--------------------------------------|--|--|--|---|--|----------------------------------|
| Características | PoW | PoS | DPoS | PoET | PBFT | DAG |
| Configuración | Sin autorización pública/ Blockchain privada | Sin autorización pública/ Blockchain privada | Sin autorización pública/ Blockchain Privada | Privado autorizado sin permiso Blockchain | Blockchain privado con permiso | Publico no autorizado Blockchain |
| Costo de entrada y devolución | Costo de entrada relativamente alto, pero altos retornos | Bajo costo de entrada, pero bajos retornos | Menor costo y menores retornos que POS | Muy bajo costo de entrada, pero bajos retornos | Todos participan sin retorno | Todos participan sin retorno |
| Incentivos | El minero ganador recibe nuevas monedas con las tarifas de bloque y transacción en el bloque que él/ ella valida | El ganador recibe tarifas de transacción con el nuevo bloque. Si un ganador de bloque intenta agregar un bloque inválido, él o ella pierden su apuesta | La amenaza de pérdida de reputación e ingresos proporciona incentivos para que los delegados actúen honestamente y mantengan la red segura | El minero ganador recibe las tarifas de transacción con el nuevo bloque que él/ ella valida | Ninguna | Ninguna |
| Finalidad | Probabilístico | Probabilístico | Probabilístico | Probabilístico | Inmediata | Probabilístico |
| Escalabilidad en la red | Alto | Medio | Medio | Medio | Bajo Crezca rápidamente en un costo de | Alto |

| | | | | | | | |
|---------------------------------------|--|---|--|---|--|---|--|
| | | | | | | comunicación enorme como la cantidad de nodos escala hacia arriba | |
| Eficiencia Energética | Muy bajo (cálculos intensivos de energía, por ejemplo. Bitcoin consume alrededor de 121,36 teravatios-hora (TWh) al año) | Alto | Alto no se requieren mineros | Alto | | Medio Algunos sistemas PBFT usan POW para prevenir ataques Sibil, pero solo después de un número determinado de bloques (es decir, 100) y no para cada bloque) | Medio Una pequeña operación PoW cuando un nodo envía una transacción para asegurar que la red no está siendo spam y también valida transacciones anteriores |
| Ataque de mayoría o 51 | el número de nodos maliciosos > 25% de todos los nodos para el ataque | Reducción de la probabilidad de ataque del 51 | Más fácil organizar un ataque del 51% si los delegados combinan su poder | Reducción de la probabilidad de ataque del 51 | | el número de nodos maliciosos > un tercio de todos los nodos para el ataque | no probado a escala |
| Susceptible al ataque de Sibil | No | yes | yes | no | | yes | No |

Nota. Se realiza una comparación de las características de cada uno de los algoritmos de consensos para identificar su funcionamiento y analizar cuál es el más óptimo.

2.3.1.2. Mecanismo de Prueba de Trabajo (PoW)

El algoritmo de consenso conocido como Proof-of-Work (PoW) es el método original utilizado en las redes de Cadena de Bloques para garantizar la seguridad y la confiabilidad de las transacciones. En este proceso, los mineros o validadores contribuyen activamente para verificar y confirmar las transacciones en la red, a cambio de una recompensa. Es importante tener en cuenta que la validación y creación de nuevos bloques en la cadena requieren la colaboración de todos los mineros en la

red para asegurar una verificación minuciosa de las transacciones.(Rizwan et al., 2018)

Según Mingxiao et al. (2017) el mecanismo de prueba de trabajo (PoW), se basa en la asignación de derechos de contabilidad y recompensas, por medio de la competencia de poder Hashing; donde los diferentes nodos calculan la solución específica de un problema matemático, apoyándose en la información del bloque anterior. Además, se encontró que, para la realización de este proceso el primer nodo que resuelva el problema matemático podrá dar origen al siguiente bloque, y así tener una recompensa.

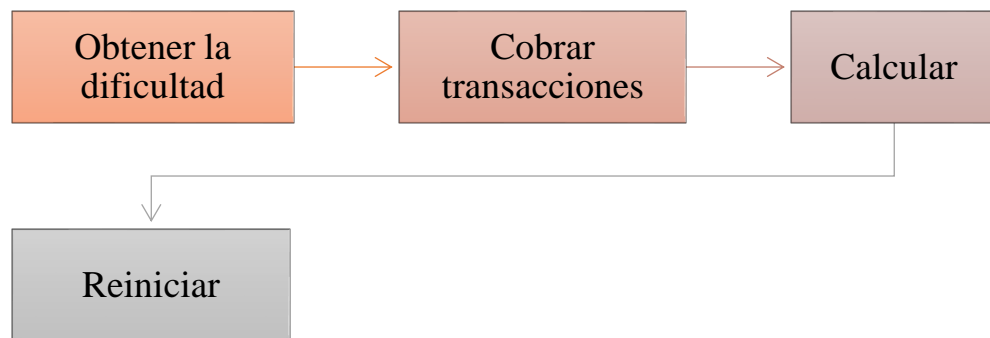
Para el cálculo del algoritmo se siguen diferentes pasos los cuales se presentan de forma simplificada en la Figura 11, estos son:

1. **Obtener la dificultad:** La cadena de bloques tiene un mecanismo de ajuste automático de dificultad para regular la tasa de creación de bloques en la red. Este mecanismo se activa después de la producción de cada 2016 bloques, y ajusta la dificultad en función de la tasa de hash total de la red.
2. **Recopilar transacciones:** Una vez que se ha producido el último bloque en la red de Bitcoin, el proceso de minería comienza con la recopilación de todas las transacciones pendientes en la red. Estas transacciones son agrupadas y se calcula su Raíz de Merkle, una forma de resumir todas las transacciones de un bloque en un solo valor hash.
3. **Cálculo:** Una vez que se han recolectado y procesado todas las transacciones y se han rellenado los campos del bloque, el próximo paso es el cálculo para resolver el problema criptográfico y generar un nuevo bloque.

Reinicio: Si el minero no logra averiguar el valor hash en un cierto período de tiempo, se reinicia el proceso desde el paso 2, volviendo a recorrer el Nonce y volviendo a calcular el valor hash del bloque. Esto se repite hasta que se encuentra una solución válida.

Figura 9

Proceso de funcionamiento del algoritmo de consenso POW



2.3.1.3. Operaciones Criptográficas Hash

Para realizar consultas sin tener que descargar toda la información almacenada. Blockchain hace uso de un árbol hash de Merkle, el cual permite almacenar diversas piezas de información independiente, en las hojas de una estructura en árbol.

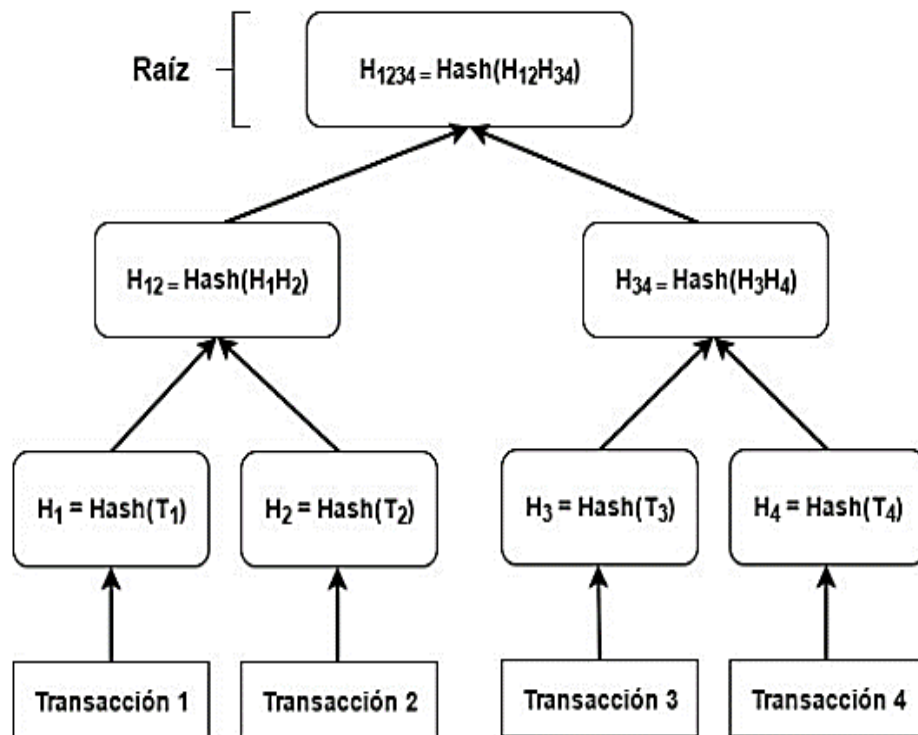
Para la creación del árbol, se hace un hash de la información contenida en cada nodo hoja, para luego pasar a generar los nodos de cada nivel superior del árbol, se procede a concatenar diversos valores hash del nivel inferior, y se le aplica la función hash a esta concatenación.

Este proceso se repite hasta llegar a un nivel donde hay un sólo nodo, denominado la "raíz" del árbol el cual se muestra en la Figura 12, en el cual se puede consultar de forma autenticada cualquier contenido del árbol, con una cantidad de valores hash; proporcional al logaritmo del número de nodos del árbol, para realizar la validación de un contenido se procede a calcular el valor raíz a partir de los nodos

adyacentes, y se comprueba si existe coincidencia con el valor raíz autenticado. La estructura es segura; porque no se puede generar un conjunto de nodos adyacentes a voluntad, que dé como resultado el valor del nodo raíz autenticado. (Retamal et al., 2017)

Figura 10

Árbol Hash de Merkle

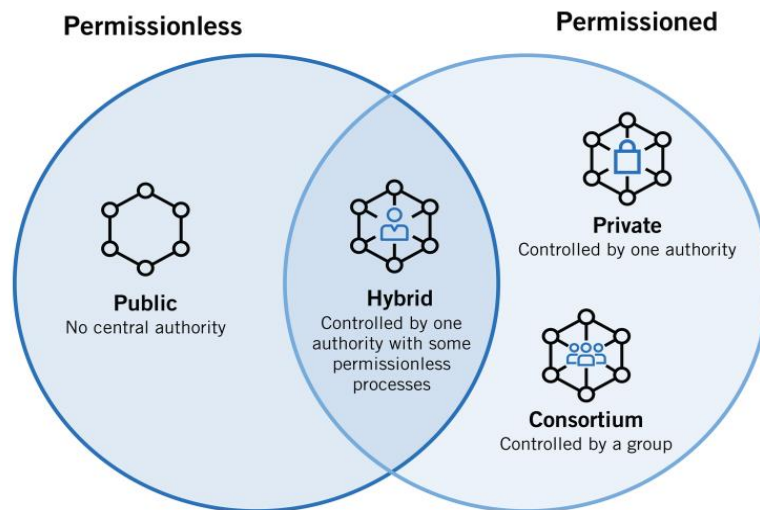


2.3.2. Tipos de Cadenas de Bloques

En la actualidad encontramos diferentes tipos de Cadenas de Bloques como se muestran en la Figura 13, cada una de ellas se diferencia por sus capacidades y características, las cuales dependen del uso que se les desee otorgar, lográndose adaptar a las necesidades que tenga la red.

Figura 11

Tipos de Blockchain



a) Cadenas de Bloques Pública

“Las cadenas de bloques públicas se presentan como grandes redes distribuidas, cuya ejecución se da a través de un token nativo” (Laurence, 2017) .Están abiertas para que cualquier persona pueda registrarse como usuario , pero ninguno pueda censurar eficazmente la actividad dentro del sistema. Los clientes no pueden confiar en mecanismos "off-chain" (como contratos legales) para protegerse contra el fraude y el abuso(Guegan, 2017).

b) Cadenas de Bloques Privada

Las cadenas de bloques privadas también conocidas como cadenas de bloques administrativas, se encargan de restringir el acceso a un círculo seleccionado de personas e instituciones, permitiendo que solo partes seleccionadas pueden acceder y hacer cambios en el libro mayor distribuido(Guegan, 2017) .” En una Blockchain

privada, la autoridad central determina quién puede ser un nodo, por lo cual no concede necesariamente a cada nodo derechos iguales, para desempeñar funciones”(Wegrzyn & Wang, 2021 ,p.3)

c) Cadenas de Bloques Híbrida

La Blockchain Híbrida se centra en mantener una estructura distribuida, al mismo tiempo que fortalece la seguridad mediante una participación limitada; con lo cual se logra resolver el problema de la baja velocidad de transacción y los problemas de escalabilidad existentes en la Blockchain Pública, en este tipo de Blockchain el grado de apertura de los datos varía, por medio de controles de acceso definidos por el consorcio, para controlar el ingreso de participantes y la información dentro de la Blockchain. (Rojas, 2018, 19)

2.3.3. Características de la Cadena de Bloques

Parrondo (2018) indica que la Cadena de Bloques ha estado revolucionando la tecnología, adoptando el potencial necesario para cambiar la forma en que las empresas se enfrentan al futuro. Según (Raikwar et al., 2020) esta tecnología ofrece una mayor seguridad y calidad en los datos mediante las siguientes ventajas:

- **Descentralización:** no hay una entidad de confianza que controle la red, lo que significa que no hay un único punto de fallo.
- **Consistencia:** los nodos de la cadena de bloques leen los mismos datos al mismo tiempo, evitando el doble gasto y garantizando la coherencia del sistema a través del consenso.
- **Escalabilidad:** el rendimiento de la cadena de bloques aumenta con el incremento del número de pares y recursos informáticos asignados.
- **Disponibilidad:** un cliente siempre recibe una respuesta independientemente de si es la última escritura en el sistema distribuido

La tecnología de cadena de bloques también requiere una ciertas Características de TI para brindar varias ventajas tecnológicas clave a los usuarios, como se indica en la Tabla 3

Tabla 2

Características de TI

| Características de Cadena de Bloques | Ventajas | Desventajas |
|---|---|---|
| P2P | Las transacciones P2P son posibles sin la confianza de un tercero como proveedor de servicio. Reducción de tarifas innecesarias | Cuando Ocorre un problema, no se sabe quién es el responsable de ello. |
| Escalabilidad | Fácilmente establecido, conectado y expandido por fuente distribuida. El costo de desarrollo del sistema es reducido. | El número posible de transacciones que se pueden manejar con el pago es reducido en comparación con la escala de transacción dentro de la economía real |
| Transparencia | Es posible acceder públicamente a todos los registros de transacciones. Legalización de transacciones y reducción de los costos de regulación. | Dado que los detalles de la transacción se revelan, todas las transacciones se pueden rastrear. La garantía perfecta de pseudo-anonimato puede ser difícil, y la reidentificación mediante la combinación es posible. |
| Seguridad | El libro mayor es de propiedad conjunta (integridad) El costo relacionado con la seguridad se reduce | Cuando se pierde la clave privada o es hackeada, no existe una solución general. No proporciona confidencialidad. |
| Estabilidad del Sistema | No hay un único punto de falla. Si se producen errores o disminución de la función en ciertos sistemas participantes, el efecto en toda la red es muy leve. | Focalizada en grandes grupos mineros. Es difícil ejecutar en tiempo real manejo de gran volumen. |

Nota : (A. Rojas & Rodrigo, 2018)

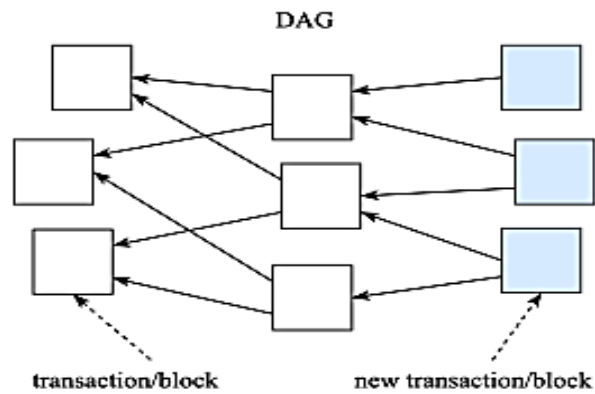
2.3.4. Mecanismo de Autenticación para Cadena de Bloques

La Cadena de Bloques realiza el manejo de un libro mayor el cual es inmutable y se encarga de verificar y asegurar que los usuarios, las transacciones y los mensajes sean legítimos. Ismail (2017) indica que en una cadena de bloques la autenticación es realizada mediante contratos inteligentes, que se escriben y se implementan en cadena de bloques (p. 1737). Es de gran importancia el uso de un generador para la creación de contratos inteligentes, “ el cual será el encargado de activarse cada vez que una de las partes requiera una autenticación, y autogobernarse dentro de un ámbito de acciones predefinido; consiguiendo así eliminar la necesidad de que un tercero autentifique las transacciones”(Lim et al., 2018,p.1737).

2.3.5. TANGLE

IOTA aparece con una novedosa arquitectura distribuida basada en un grafo acíclico dirigido DAG, el cual es un sistema de almacenamiento cuya característica principal es que los enlaces creados entre los elementos de la red siempre van en una dirección, además estos no pueden crear bucles dentro de la estructura (Sorrius Martí, 2018), si un nodo no contribuye a la red, puede ser eliminado. Por esta razón, un nodo es incentivado a participar en la red, incluso en momentos en que no emite transacciones(Bachmann, 2019).

El libro mayor distribuido de DAG, es un Distributed Ledger Technology (DLT) se encarga de almacenar los datos de las diferentes transacciones, en forma de vértices de un gráfico acíclico dirigido. El DAG permite agregar nuevas transacciones a diferentes vértices al mismo tiempo, como se indica en la Figura 14. Los libros mayores de DAG se dividen en dos categorías como son: basados en bloques (blockDAG) y basados en transacciones (TDAG)(Fan et al., 2021).

Figura 12*Estructura de DAG*

2.3.5.1. Consenso

El término "consenso" se refiere al proceso que lleva a todos los nodos a ponerse de acuerdo sobre el mismo estado que el DLT. El consenso es intrínseco en el proceso de emisión de transacciones, y es una de las principales características del DLT. Se dice que una transacción se confirma cuando es validada directa o indirectamente por todas las transacciones que aún no están aprobadas. (Cotugno et al., 2020)

Según Silvano & Marcelino (2020) el proceso para adjuntar una transacción a Tangle va de la siguiente manera:

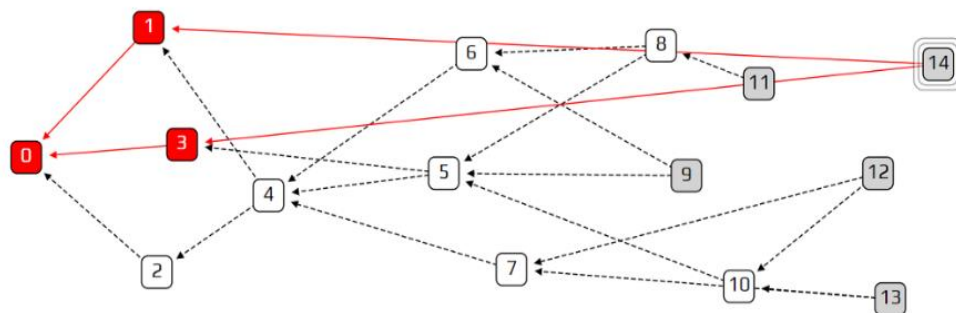
- 1) Un nodo elige otras dos transacciones para aprobar, de acuerdo con un algoritmo de selección de punta Markov Chain Monte Carlo (MCMC).
- 2) El nodo comprueba si las dos transacciones están, o no, en conflicto y desaprueba las transacciones conflictivas (doble gasto).
- 3) En secuencia, para que el nodo emita una transacción válida, debe resolver una especie de prueba de trabajo (PoW), esto se logra encontrando un nonce, de modo que su hash se concatena con algunos datos de la transacción aprobada.

- 4) Después de eso, el usuario envía su transacción a la red, y se convierte en una punta (transacción no aprobada).
- 5) La punta espera la confirmación, a través de la aprobación directa o indirecta, hasta que su peso acumulado alcanza el umbral predefinido.

El nivel de confianza de una transacción se decide a través del uso de un algoritmo de selección de propinas, el cual se encarga de seleccionar las transacciones para la confirmación, dependiendo del peso acumulado, tomando en cuenta que entre más alto sea el peso acumulado, más fácil será la elección de la punta. (Fan et al., 2021) “El peso de una transacción es proporcional a la cantidad de trabajo que el nodo emisor invirtió en ella como se indica en la Figura 15; en la práctica, el peso sólo puede asumir los valores $3n$, donde n es un número entero positivo y pertenece a algún intervalo no vacío de valores aceptables”(Popov, 2018).

Figura 13

Pesos dentro de una transacción



2.3.5.2. Seguridad en Tangle

Tangle, al igual que otras tecnologías nuevas, tiene sus propios desafíos de seguridad debido a que no se basa en el protocolo de cadena de bloques. Esto genera problemas específicos, como los relacionados con el protocolo de verificación basado en DAG y los exclusivos a su implementación. Los principales ataques en un sistema

basado en Tangle incluyen los relacionados con la selección de peso y cómo se manejan los conflictos en las nuevas transacciones. Además, hay vulnerabilidades críticas en la función hash de Tangle y otros posibles ataques que se han presentado anteriormente (Sorrius Martí, 2018).

Para mejorar la seguridad en el Tangle, se utiliza el algoritmo MCMC (Markov Chain Monte Carlo) con el objetivo de proteger contra ataques de usuarios malintencionados y aumentar y garantizar la seguridad de la red. La seguridad del algoritmo frente a los ataques puede ajustarse mediante el parámetro α : valores altos de α aumentan la probabilidad de que la partícula se dirija a las transacciones con el mayor peso acumulativo (a medida que α se acerca a ∞ las partículas se mueven de manera determinista), mientras que valores más bajos de α hacen que el algoritmo y su resultado sean más impredecibles (ya que los pesos acumulativos importan menos y la probabilidad de salto tiende a ser uniforme a medida que α se acerca a cero). (Silvano & Marcelino, 2020).

Según Brooks (1998) dentro de los métodos de MCMC se conoce la distribución estacionaria, y se procede a identificar la distribución de transición, aunque en la práctica puede haber infinitas distribuciones para elegir. El teorema principal que sustenta el método MCMC, es que cualquier cadena que sea irreducible y aperiódica tendrá una distribución estacionaria única, y que el núcleo de transición t-step convergerá a esa distribución estacionaria (p,7)

2.3.5.3. Características de Tangle

Es vital contar con un sistema que pueda procesar una gran cantidad de transacciones de forma eficiente, transparente y sin cuellos de botella. IOTA se esfuerza por abordar estos desafíos y busca solucionar problemas relacionados con estas problemáticas específicas por lo cual cuenta con diferentes características como son:

a) Escalabilidad

IOTA no utiliza bloques, por lo cual se procede a agregar una a una las transacciones a la red, permitiendo así tener mejor escalabilidad y una menor latencia. Para que se pueda publicar una transacción, se deben validar en primera instancia dos transacciones, mientras más transacciones se vayan validando, el Tangle será óptimo y más seguro, debido a que irá certificando las transacciones una por una (Santos Ramírez, 2020).

b) Eficiencia Energética

La tecnología IOTA permite que la prueba de trabajo (PoW), se externalice a un dispositivo más potente, para así poder reducir el consumo de energía de los dispositivos IoT restringidos. (Alsboui et al., 2020)

c) Seguridad y privacidad

La tecnología IOTA ha desarrollado Masked Authenticated Messaging (MAM), el cual se presenta como un protocolo de comunicación de datos de segunda capa; tiene la capacidad de transmitir y acceder a flujos de datos cifrados a través del enredo (Alsboui et al., 2020). En IOTA solo el propietario de un canal, podrá publicar datos dentro de otro, haciendo que cualquier intento de escribir datos falsos sobre el canal, o la toma de control del mismo pueda ser detectado de forma fácil por los dispositivos autorizados (Carelli et al., 2022). El cifrado MAM está habilitado por tres modos, lo cual ayuda a controlar la visibilidad y el acceso a los canales públicos, privados y restringidos. En consecuencia se podrá, cifrar, autenticar y transmitir datos a la red IOTA. (Alsboui et al., 2020)

2.3.6. Protocolos y elementos de Seguridad en el Esquema de Autenticación

La seguridad en una red de sensores inalámbricos (WSN) es esencial para garantizar la protección de la información transmitida, los recursos y evitar

comportamientos inadecuados de los nodos. Para lograrlo, se utilizan diversos protocolos y elementos de seguridad. En este tema, se discutirán los protocolos y elementos de seguridad utilizados en el esquema de autenticación.

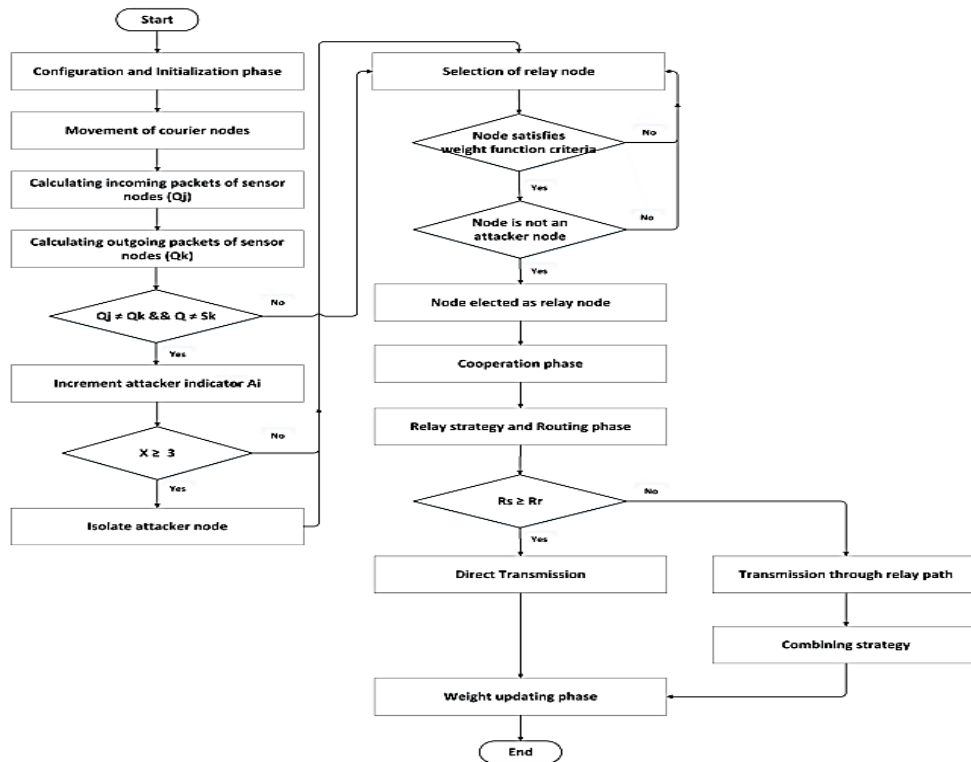
a) SEECR: Protocolo de enrutamiento cooperativo y de eficiencia energética segura para redes de sensores inalámbricos

Según indica Saeed et al.(2020) El protocolo propuesto del SEECR usa una red de múltiples saltos en el entorno de las UWSNs (Redes de sensores submarinos); por medio de la aplicación de la técnica de cooperación. Además, explica que en este esquema los paquetes de datos generados desde el nodo de origen son reenviados al nodo de destino, a través de salto por salto, logrando detectar ataques comunes de enrutamiento activo, así como también la eliminación de nodos atacantes.

Para detectar y eliminar nodos atacantes, cada nodo sensor almacena paquetes que son enviados y recibidos por nodos vecinos, como se muestra en la Figura 16, los paquetes se almacenan en Q_j y Q_k . Los paquetes entrantes P_{in} del nodo sensor se almacenan en Q_j , y los paquetes salientes P_{out} se almacenan en Q_k . Después de almacenar los paquetes, se compararán los valores Q_j y Q_k . Si ambos valores no son iguales y el nodo no se considera un nodo fregadero, entonces hay posibilidades de sea un nodo atacante.(Saeed et al., 2020, pag 6-7)

Figura 14

Diagrama de Flujo de SEECR



b) Contraseñas Randómicas

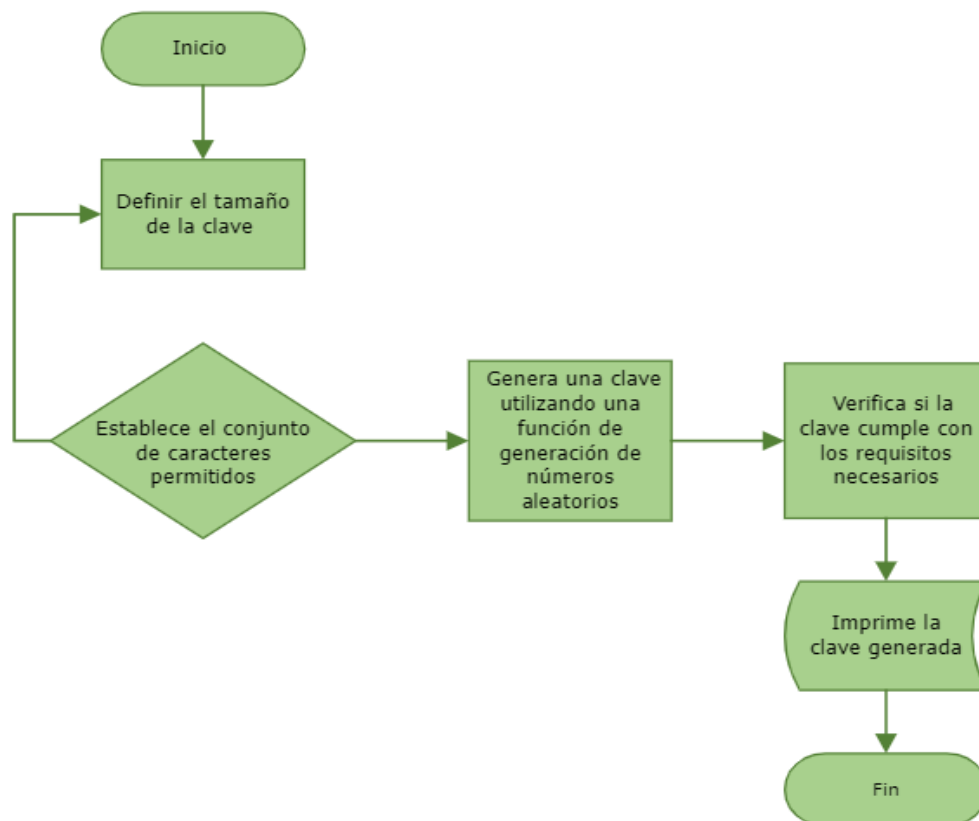
Para generar contraseñas aleatorias, según lo sugerido por, Chan et al., (2003), se puede elegir un conjunto de claves aleatoriamente a partir de un archivo de contraseñas. Luego, cada nodo del sensor recibirá un subconjunto de claves que incluye palabras con dígitos y símbolos especiales. Solo los dos nodos que pueden encontrar una instancia en común en sus subconjuntos respectivos podrán usar esta clave compartida como su contraseña para iniciar la comunicación.

La Figura 9 muestra cómo se crea una contraseña aleatoria. Para ello, se comienza definiendo el tamaño de la contraseña, que puede estar compuesta por diferentes cantidades de caracteres. Luego, se establece el conjunto de caracteres permitidos, que pueden incluir números, letras o caracteres especiales, o una combinación de ellos. A continuación, se utiliza una función de generación de números aleatorios para seleccionar un carácter del conjunto de caracteres permitidos. Este

proceso se repite hasta que se hayan seleccionado todos los caracteres necesarios para formar la contraseña de la longitud deseada. Finalmente, se verifica que la contraseña generada cumpla con cualquier requisito adicional que se haya establecido, como la presencia de al menos una letra mayúscula y una minúscula, o un número.

Figura 15

Creación de Contraseñas Randómicas



c) Contraseñas Saladas (Salted Passwords)

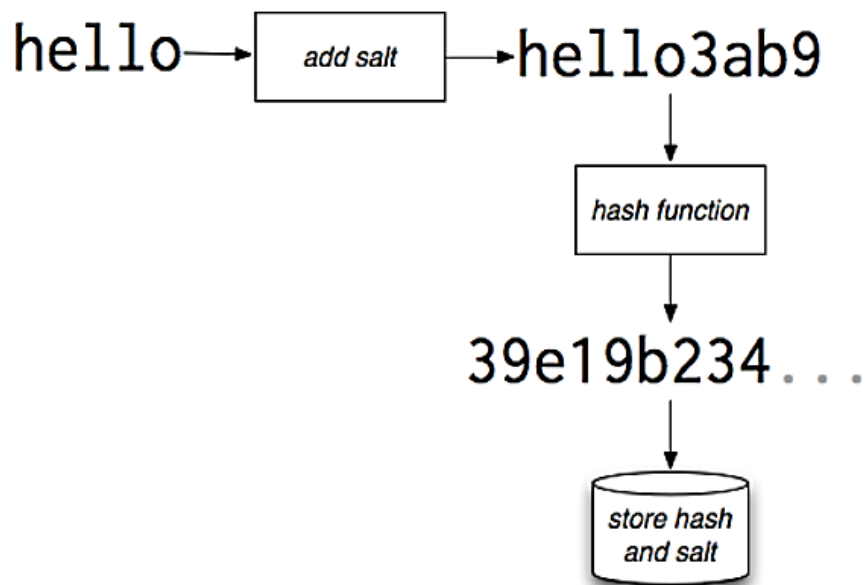
En esencia, la sal es un bloque de datos aleatorios, ya sea en forma de cadenas o bytes, que se combina con el texto plano para generar el valor de sal y con ello un hash salado. Según (Patel et al., 2013) el proceso para generar un hash salado consiste en utilizar el valor de la sal como prefijo del texto plano o agregarlo antes de calcular el hash. Los pasos para generar una contraseña hash salado incluyen:

1. Recolectar la contraseña original
2. Generar un valor de sal mediante funciones/método aleatorio de confianza
3. Añadir la sal a la contraseña original
4. Generar la contraseña hash salado usando la función hash apropiada
5. Almacenar la sal y el hash en una base de datos.

Una salt como se muestra en la Figura 17 es una cadena aleatoria de letras o números que se agrega al principio o al final de una contraseña antes de aplicarle un algoritmo hash. En otras palabras, en lugar de solo cifrar la contraseña $h(\text{contraseña})$, se calcula $h(\text{sal}||\text{contraseña})$. La ubicación de la salt puede ser al principio o al final de la contraseña, pero es importante tener en cuenta esto para que las contraseñas sean más resistentes y difíciles de descifrar. (Boonkrong & Somboonpattanakit, 2016).

Figura 16

Salt Criptográfica



2.4.Seguridad en Redes de Sensores Inalámbricos

Las redes de sensores inalámbricos a diferencia de las redes convencionales presentan una mayor cantidad de vulnerabilidades de seguridad, esto se debe a que son implementados dentro de medios hostiles sin protección física, además las redes WSN cuentan con recursos limitados que dificultan la protección de la información transmitida entre los diferentes sensores.

Según Keerthika & Shanmugapriya (2021) se considera que para que las redes de sensores funcionen de forma correcta y de forma segura se debe tomar en cuenta aspectos como:

- **Disponibilidad:** Es de vital importancia que los recursos estén disponibles en la red operativa para que el mensaje pueda transmitirse sin problemas y los nodos puedan utilizar el recurso y la red.
- **Autorización:** Busca que solo los sensores autorizados proporcionen información a los servicios en la red operativa.
- **Autenticación:** Implica que los nodos del sensor en la comunicación son genuinos y tienen acceso adecuado a la red.
- **Confidencialidad:** Asegura que el mensaje en la red de comunicación no puede ser leído y entendido por los atacantes.
- **Integridad:** Se refiere a que el mensaje no es alterado o manipulado mientras estaba en la comunicación de red. Simplemente inyectando paquetes adicionales, se puede cambiar todo el paquete. (p.2)

2.4.1. Autenticación

La autenticación es un proceso mediante el cual se verifica la identidad de un nodo en una red y se garantiza que los datos o los mensajes de control se originan de

una fuente autenticada. Rajeswari & Seenivasagam (2016) presentan varios tipos y procedimientos de autenticación, que se pueden entender mejor a través de la Tabla 1.

Tabla 3.

Tipos de Autenticación y Características

| Tipo de Autenticación | Conceptos |
|--|---|
| Autenticación unidireccional | Solo se transmite un mensaje desde el nodo remitente al nodo receptor. |
| Autenticación bidireccional o mutua | Ambas entidades pueden autenticarse entre sí en un enlace de comunicación. En entornos WSN, este esquema no solo significa la autenticación entre los nodos normales y la estación base, sino que también menciona las dos contrapartes que son seguras de la identidad del otro |
| Autenticación tripartita | Un tercer mensaje del remitente al receptor se envía una vez que los relojes de los nodos no se pueden sincronizar |
| Autenticación implícita | La autenticación implícita no solo se realiza como un proceso independiente, sino que también es el subproducto de otros procesos como el establecimiento de claves. En las WSN, este tipo de autenticación puede minimizar tanto la complejidad operativa como el consumo de energía |

2.4.2. Dificultades de la seguridad en redes de Sensores

De acuerdo con Sinha et al. (2017), entre las dificultades que se presentan al tratar de implementar un alto nivel de seguridad en redes de sensores se encuentran:

- La seguridad en las redes de sensores se complica por las limitadas capacidades del hardware del nodo del sensor y las propiedades del despliegue.
- Los nodos del sensor son susceptibles a la captura física, pero debido a su bajo costo, es poco probable que prevalezca el hardware resistente a manipulaciones.
- Un aspecto a tomar en consideración es el despliegue de las redes de sensores WSN en áreas peligrosas, la capacidad del WSN para soportar las duras condiciones ambientales donde se despliegan se encuentran desatendidas (Keerthika & Shanmugapriya, 2021).
- Los nodos sensores utilizan la comunicación inalámbrica, lo que los hace más propensos a ser escuchados sin que sepan las personas involucradas, ya que esto permite interceptar las comunicaciones sin su conocimiento o consentimiento.
- Las técnicas avanzadas anti-jamming tales como: espectro extendido de frequency-hopping y prueba física de la manipulación de nodos; son generalmente imposibles en una red de sensores; debido a los requisitos que necesita un diseño complejo, debido a su mayor tamaño y consumo de energía más alto.
- La seguridad también debe ampliarse a despliegues a gran escala. La mayoría de los protocolos de seguridad estándar actuales se diseñaron para configuraciones de dos partes y no se escalan a un gran número de participantes.
- Uno de los desafíos que se busca solventar dentro de las redes de sensores ,es prolongar la vida útil de los nodos, para así disminuir el consumo de energía de la red, consiguiendo mejorar su ciclo de vida (Keerthika & Shanmugapriya, 2021).

2.5.Trabajos Relacionados

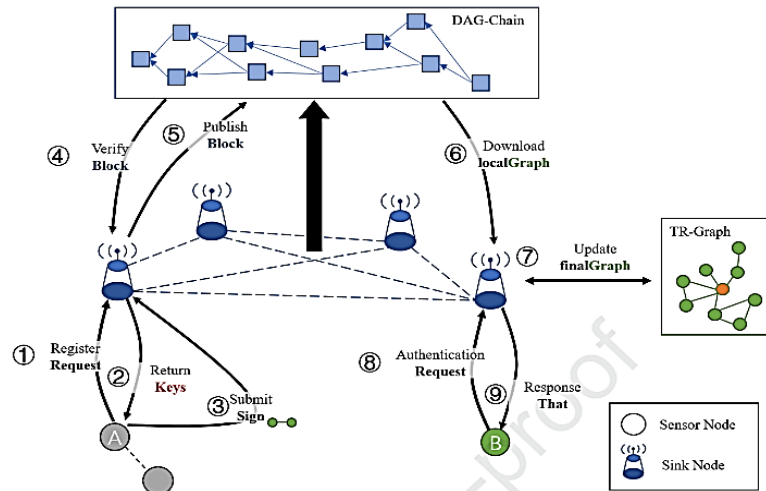
En este apartado se muestran diferentes investigaciones que tienen la misma temática del trabajo de tesis a desarrollar; y que también muestran diferentes esquemas de autenticación que se han propuesto, para dar solución al problema de seguridad en las redes de sensores inalámbricos WSN.

Una autenticación potenciada por Blockchain esquema para la detección de gusanos en la red inalámbrica red de sensores

Dentro del artículo de Y. Chen et al.(2022) se propone un esquema basado en Blockchain para el nodo de autenticación en WSN, el cual consta de cinco elementos primordiales que son estación, nodo sumidero, nodo sensor, red de sensores inalámbricos y red Blockchain. Este esquema cuenta con dos procesos: generación de información de identidad e interproceso de acción con Blockchain. El esquema divide la WSN en múltiples redes locales independientes, como se observa en la Figura 17 cada nodo fregadero realiza la autenticación de los nodos del sensor en la red local, para que después la cadena de bloques se encargue de almacenar la información de la identidad del sensor, consiguiendo de esta forma garantizar la seguridad, después se realiza la simulación de la base de capacidades de gestión de nodos (BAS), logrando realizar la detección de gusanos. De esta forma se prueba el potencial de seguridad y del esquema creado. Los resultados que arroja este esquema indica, que existen ciertas limitaciones para lograr equilibrar la sobrecarga de almacenamiento y comunicación, además de que aparecen ciertas dificultades para conseguir agrupar nodos dinámicos.

Figura 17

Proceso de Interacción en esquema para la detección de gusanos



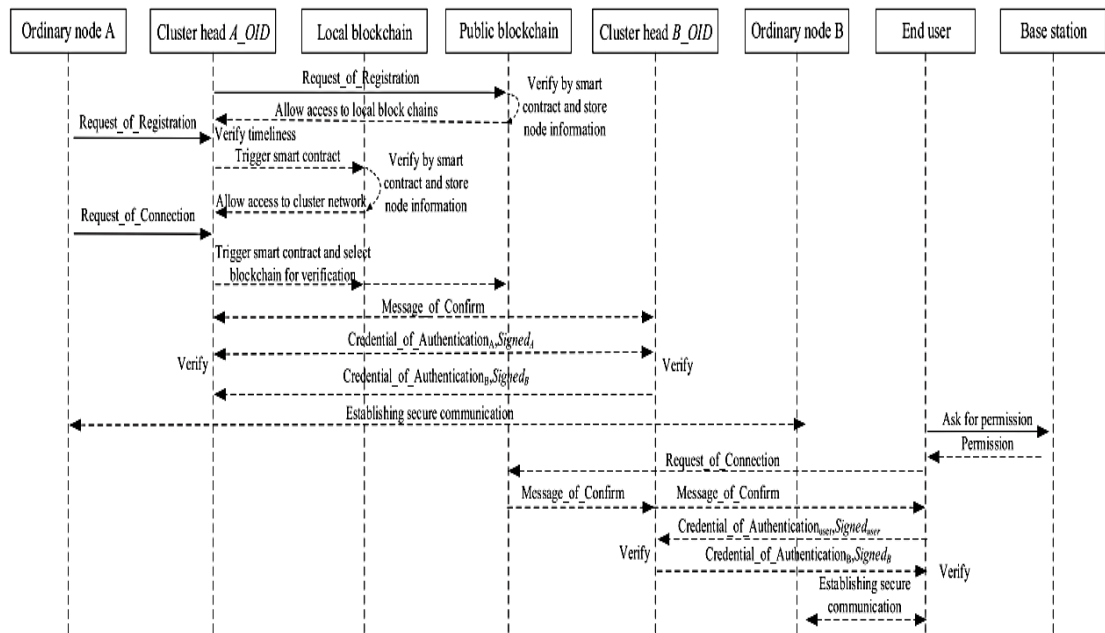
Fuente: Y. Chen et al.(2022)

Una identidad híbrida basada en Blockchain Esquema de autenticación para Multi-WSN

Este esquema de autenticación de identidad multi-WSN se basa en las características de descentralización que maneja la cadena de bloques, y las combina con la estructura distribuida de los nodos de IoT, para lo cual, como etapa inicial se construye una cadena de bloques privada entre las cabezas de clúster, formando una sola red WSN como se muestra en la Figura 18. A la cual, se agrega una cadena de bloques publica a las diferentes estaciones base, de las redes de sensores WSN. En este esquema se completa el registro de información de identidad entre nodos de cabecera de clúster, nodos ordinarios y la autenticación dentro de la comunicación. Después de realizar diferentes pruebas, se consigue demostrar que el esquema cuenta con una buena seguridad y eficiencia(Cui, 2020).

Figura 18

Diagrama de Flujo de esquema de autentificación para Multi WSN



Fuente: (Cui, 2020).

Un esquema escalable basado en Blockchain para intercambio de datos relacionados con el tráfico en VANET

El presente esquema propone el uso de una cadena de bloques autorizada para la gestión segura de datos del tráfico vial, para lo cual se aprovecha la proximidad de las RSU a lugares de eventos, creando dinámicamente grupos de consenso buscando mantener el diseño centralizado, para evaluar el rendimiento, la latencia, la comunicación y los costos de almacenamiento, así como también la efectividad de las micro transacciones. La simulación creada arroja como resultados, que el mejor desempeño para el protocolo propuesto se obtiene al trabajar con esquemas basados en Prueba de trabajo; además se logró visibilizar que el esquema es capaz de minimizar costos de comunicación y almacenamiento además de que supera la replicación completa de la cadena de bloques. (Diallo et al., 2022)

Existen diferentes investigaciones enfocadas al desarrollo de esquemas de autenticación, que buscan mejorar el desempeño de las redes de sensores inalámbricas en el área de la seguridad, sin embargo, debido a una gran cantidad de limitaciones, muchos esquemas terminan siendo obsoletos; lo que ha llevado a la inclusión de tecnologías emergentes en la creación de nuevos esquemas.

Una de las investigaciones analizadas se basa en la fusión de Tangle y Cadena de Bloques, para lo cual en primera instancia se crea múltiples redes locales, las cuales para formar una sola red se autentican con el nodo central, en este esquema la cadena de bloques se encarga de guardar la información obtenida de toda la red para luego hacer uso de una base de capacidades de gestión de nodos (BAS) que será la encargada de la detección de gusanos dentro de la red, otros esquemas se basan en las características de descentralización, efectividad de las micro transacciones y prueba de trabajo (POW) que maneja la cadena de bloques para luego combinarlas con diferentes estructuras como IoT lo cual permite crear un esquema liviano y seguro capaz de trabajar de modo eficiente.

Al revisar estas investigaciones se llegó a la conclusión de que el manejo de Blockchain y Tangle dentro de una red de sensores es factible, pero que para su correcto diseño se deben tener claros el funcionamiento y limitaciones de estas, ya que solo así se podrá crear un esquema que brinde la seguridad necesaria dentro de la red.

3. CAPITULO III: Definir el Esquema de Autenticación

En esta sección, se indica como se hace uso del modelo en cascada , de manera que a través de un proceso secuencial nos permitirá realizar un correcto diseño del esquema de autenticación, basándose en el funcionamiento de tecnologías emergentes como son la Cadena de Bloques y el Tangle, siendo esta última la encargada de realizar los procesos de autenticación entre los nodos , seguido de esto se abordará el estudio de los diferentes requisitos y requerimientos que necesita la red para su funcionamiento; dentro de este capítulo también se especifica la ubicación de los nodos y el funcionamiento que tendrá la red de sensores inalámbricos.

3.1. Metodología de Diseño

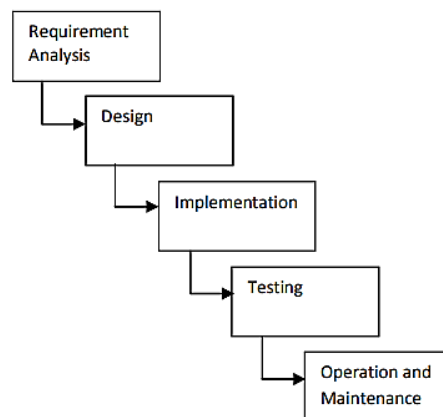
La metodología es un factor clave que permite desarrollar un conjunto de pasos y procedimiento de investigación que ayudan a desarrollar de forma adecuada este trabajo de titulación, en primera instancia se optó por realizar la recolección de información a través del uso de diferentes fuentes bibliográficas, por ello se procedió a la búsqueda de artículos científicos y tesis relacionadas con el tema, como resultado de la investigación , se identificó diferentes aspectos fundamentales, entre los cuales se encuentran: la comprensión del funcionamiento y características de las redes WSN, así como también los ataques que pueden sufrir; los algoritmos de consenso que definen el funcionamiento de las tecnologías de Cadenas de Bloques y Tangle; las operaciones criptográficas y las técnicas matemáticas que utilizan para realizar la autenticación.

Estos parámetros ayudan a crear una propuesta concisa que permite integrar las tecnologías de Cadena de Bloques y Tangle para la creación de un esquema de autenticación. Los aspectos fundamentales identificados en la investigación permiten, el desarrollo del diseño y simulación del esquema propuesto, con el fin de mejorar el funcionamiento y características de las redes WSN.

Al utilizar el modelo en cascada como referencia, es importante considerar que esta metodología se presenta como un procedimiento lineal, el cual se divide en cinco etapas fundamentales: Requisitos, Diseño, Implementación, Verificación y Mantenimiento que se ejecutan de forma única. Cabe destacar que el modelo de cascada presentado en la Figura 19 es estático y aborda el desarrollo de sistemas de manera secuencial, se debe comprender que dentro del modelo en cascada cada paso se congela, lo que implica que se debe asegurar que cada fase esté completamente terminada antes de continuar al siguiente paso, por lo cual es recomendable realizar diferentes revisiones hasta tener claro si el resultado obtenido es el que se desea. (Adenowo & Adenowo, 2020).

Figura 19.

Modelo en Cascada



Fuente .(Adenowo & Adenowo, 2020)

3.2.Fase 1: Requisitos y Requerimientos

En la Fase 1 se da a conocer los aspectos necesarios para realizar el proyecto, entre ellos tenemos el análisis; este nos ayuda a entender la problemática existente dentro de las redes de sensores; permitiendo determinar los diferentes parámetros para el establecimiento de una solución; otro elemento importante es la situación actual, en esta se expone los trabajos y medidas tomadas para mejorar la seguridad de las redes de

sensores. Dentro de esta sección también se expone los desafíos que presenta la creación de un esquema de autenticación seguro para redes de sensores inalámbricos (WSN), a través de la integración de dos tecnologías diferentes como son la cadena de Bloques y el Tangle.

3.2.1. Situación Actual y Tipo de Investigación

En la actualidad las redes de Sensores Inalámbricos (WSN) sufren ataques pasivos y activos, en el caso de ataques pasivos los espías pueden realizar el monitoreo del canal de comunicación que se establece entre los nodos; consiguiendo descubrir información confidencial de los mismos, sin la necesidad de perturbar la comunicación. En los ataques activos se pueden diferenciar dos tipos: externos e internos; en el caso de los externos, un nodo que no forma parte de la red manipula o interfiere en las funcionalidades de la red; a diferencia de estos, los ataques internos se generan por miembros comprometidos que pertenecen a la red de sensores, siendo más difíciles de proteger que los ataques externos (Li, 2010).

La investigación propuesta en este trabajo de tesis es de tipo cuantitativo y cualitativo ya que realiza un análisis de los elementos, procesos, arquitectura y mecanismos de autenticación que manejan las tecnologías de cadena de Bloques y Tangle para luego determinar que parámetros de seguridad se pueden emplear dentro del diseño del esquema de autenticación.

Investigación Documental: Para dar veracidad a la investigación se hace uso de diferentes fuentes entre las cuales se tiene libros, artículos y revistas científicas las cuales ayudan a crear las bases del marco teórico de esta investigación, así como también el estudio de diferentes esquemas de autenticación propuestos que permitieron desarrollar los fundamentos principales para el diseño del esquema planteado.

Investigación Bibliográfica: Este trabajo se apoya en diferentes artículos cuya investigación se centran en solucionar el mismo problema lo que ayuda a tomar en cuenta diferentes parámetros al realizar el diseño del esquema de autenticación. Además de aportar con elementos importantes como son los conceptos actuales y criterios científicos .(Burgos Yar, 2021)

3.2.2. Descripción general del Sistema

La solución propuesta se enfoca en la creación de diferentes islas compuestas por un conjunto de sensores inalámbricos, cuya comunicación se basa en la arquitectura de Tangle, la cual es unidireccional. En cada una de las islas, los nodos envían información a un nodo central denominado "Estación Base". Este último será el punto de partida para la autenticación y asociación de los nodos en cada una de las islas.

El proceso de funcionamiento inicia cuando el primer nodo se autentica con la estación base mediante el uso de un hash creado a partir de una contraseña más un "salt" aleatorio, el cual es verificado por la estación base para permitir su asociación. Los siguientes nodos que deseen unirse a la red, deben autenticarse tanto con la estación base como con los nodos que formen parte de la red tomando en cuenta que a partir del cuarto nodo la autenticación solo se realiza entre los nodos sensores.

Para verificar si un nodo es malicioso o no, el esquema de autenticación hace uso del protocolo SEECR. Este protocolo, a través de la comprobación de paquetes de entrada y salida y de los intentos de autenticación, determina si un nodo es malicioso. En caso de ser un nodo malicioso, los nodos sensores lo marcan como tal e indican el ID del nodo malicioso a todos los nodos de la red. De esta forma, se procede a cortar los caminos de comunicación que existen con este nodo, aumentando así la seguridad de la red.

Es importante destacar que cada vez que un nuevo nodo se una a la red, la estación base y los nodos sensores que ya formen parte de la red Tangle aumentaran de peso. En el esquema de autenticación, los nodos con mayor profundidad² y peso³ realizarán un cambio de contraseña periódico para fortalecer la seguridad de la red.

La Figura 20 ilustra la arquitectura de la red, donde se observa cómo se realiza la comunicación entre los nodos, así como también las características y protocolos utilizados dentro de cada una de las islas que componen la red. A través de las flechas verdes, se puede seguir el camino que seguirá la información hasta llegar al punto central conocido como Estación Base. Además, se observa que las estaciones base de las diferentes islas están conectadas entre sí mediante una cadena de bloques, lo que garantiza mayor seguridad en la transmisión de información debido a que es un sistema descentralizado.

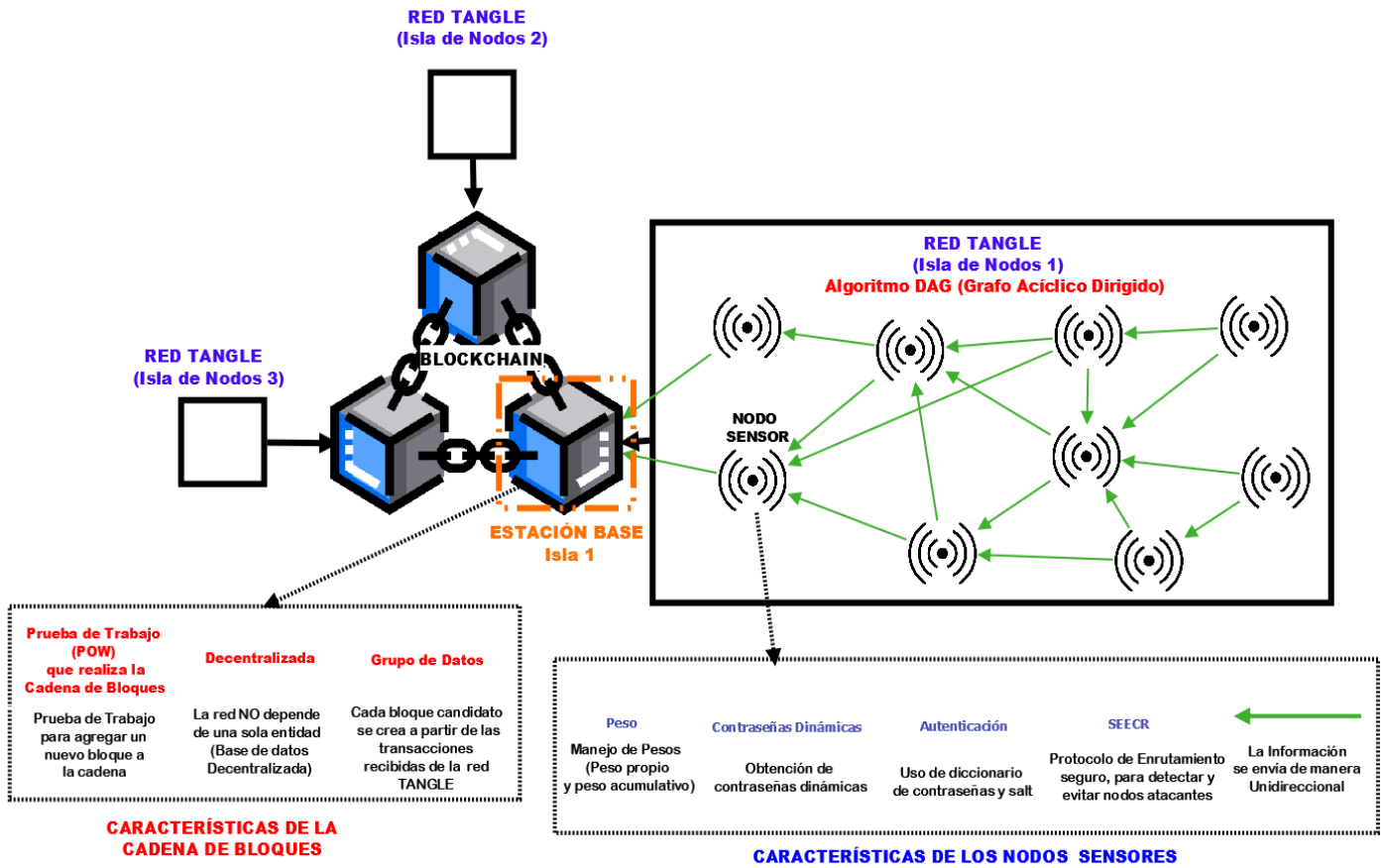
Con el fin de facilitar la comprensión, se han identificado algunas características de los nodos y las estaciones base en la imagen presentada en la Fig 20. En cuanto a los nodos, se destaca su autenticación mediante el uso de un diccionario de contraseñas y Salt, la implementación del protocolo SEECR y el peso que manejan en la red. Por otro lado, en relación a las estaciones base, se destaca su capacidad para crear un bloque candidato y llevar a cabo una prueba de trabajo.

² Profundidad: El nodo más profundo es aquel que se encuentra más cerca a la estación base.

³ Peso: Valor propio de los nodos que aumenta al unirse un nuevo nodo a la red.

Figura 20

Diseño General del Sistema



1. Limitaciones del Sistema Propuesto

El esquema de autenticación a desarrollar poseerá algunas restricciones, entre las cuales se encuentran:

- La estación base deberá tener una fuente de energía fija, debido a que realiza una prueba de trabajo que consume gran cantidad de recursos y energía.
- Es esencial tener en cuenta que las islas de nodos deben tener un tamaño limitado, ya que una cantidad excesiva de nodos puede generar un aumento en los pesos de la red y, como resultado, en el tiempo necesario para que la estación base recolecte todos los datos y genere el bloque candidato. Por esta razón, se

recomienda que cada isla no contenga más de 100 nodos. De esta forma, se asegura una eficiencia en el proceso de creación de bloques candidatos y se mejora el rendimiento de la red.

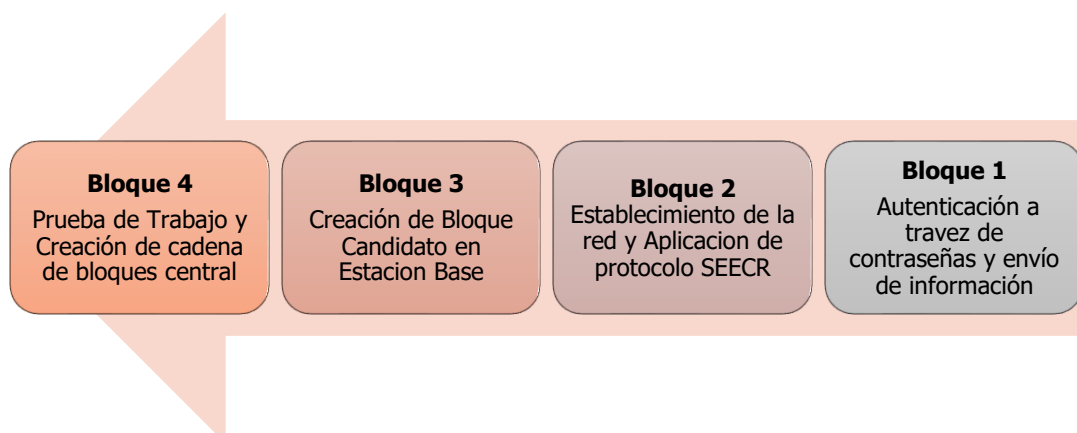
- Es necesario contar con al menos dos islas dentro de la red para poder utilizar el esquema de autenticación, ya que si se trabajara con una única isla no sería posible crear la cadena de bloques.
- Para la realización de la autenticación se hace uso de un diccionario de contraseñas, siendo un limitante ya que este método resulta vulnerable ante ataques de fuerza bruta, debido a que los atacantes pueden probar diferentes combinaciones de contraseñas hasta encontrar la correcta.

3.2.3. Diagrama de Bloques del Sistema

Para una comprensión de la estructura de la red, se presenta el diagrama de bloques del sistema en la Figura 21. Este diagrama ilustra las etapas del esquema de autenticación mediante el uso de bloques que se muestran de derecha a izquierda, indicando los aspectos principales a considerar dentro del esquema de autenticación desarrollado.

Figura 21

Diagrama de Bloques del Esquema de Autenticación



3.2.3.1. Bloque 1: Autenticación a través de contraseñas y envío de información

Para garantizar la autenticación en la red de sensores WSN, se utilizarán contraseñas y Salt dinámicas obtenidas a través de un diccionario. El proceso se inicia cuando el primer nodo de la red envía una “*Solicitud de Asociación*” a la estación base. La estación base responde con un mensaje de “*Respuesta de Asociación*” que incluye un hash generado a partir de una contraseña y una Salt. Este mensaje indica que el nodo debe autenticarse para formar parte de la red. Al recibir este mensaje, el nodo intentará descifrar el hash recibido de la estación base en un período de tiempo. Para ello, se realizará una prueba de trabajo que consiste en probar varios hash generados a partir de la mezcla de diferentes contraseñas más Salt (clave adicional) , las cuales se crean con la ayuda de un diccionario de contraseñas programadas dentro de los nodos.

Una vez determinada la clave junto con la Salt, el nodo retirará la contraseña y utilizará el hash de la Salt para su autenticación con la estación base. Si la verificación es correcta, el nodo será agregado a la red y podrá enviar y recibir datos, pero si es incorrecto, este nodo no formará parte de la red.

A diferencia del primer nodo, en la red IOTA, a partir del segundo nodo se requiere la autenticación mediante la intervención de dos entidades distintas. Esto puede lograrse de dos formas diferentes: la primera consiste en que el nodo se autentique con la estación base y con otro nodo que ya formó parte de la red. La segunda forma implica que el nodo se autentique con dos nodos distintos que ya se encuentren autenticados en la red. Esto implica que deben realizar dos pruebas de trabajo para descifrar las contraseñas correspondientes, esto se lo realiza debido a que solo los nodos que han sido autenticados por al menos dos nodos de la red pueden formar parte de la red IOTA.

Para mejorar la seguridad en la red IOTA, las contraseñas cambiarán periódicamente. El tiempo en el cual las contraseñas deben cambiarse dependerá del peso que tenga cada nodo en la red. A mayor peso del nodo, menor será el tiempo para cambiar la contraseña. Esto se realiza con el objetivo de evitar que los nodos seguros de la red sean atacados con facilidad.

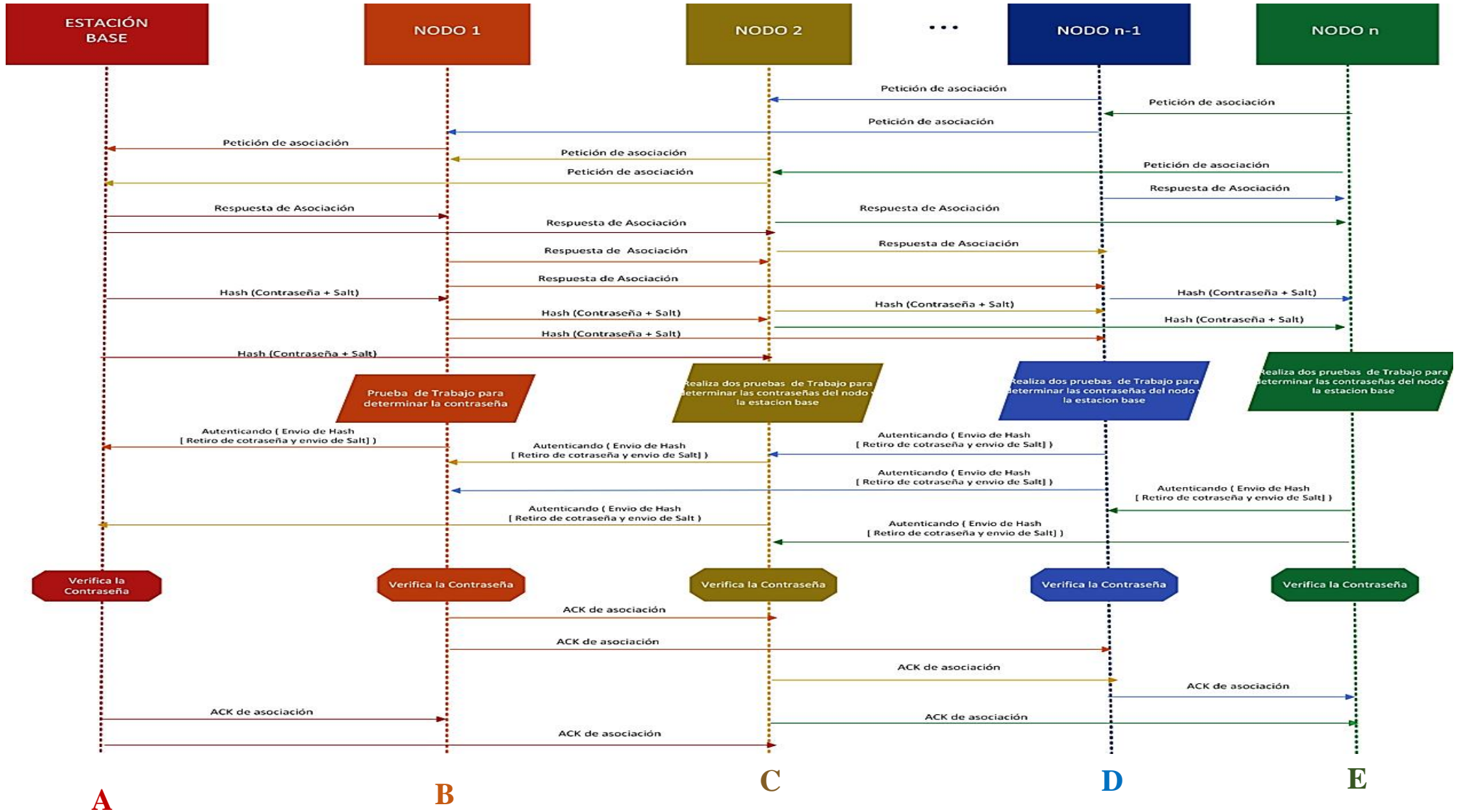
En la Figura 22 se muestra el diagrama de secuencia del proceso de autenticación de contraseñas hasta la verificación y envío de datos. En primer lugar, el Nodo 1 envía una Petición de asociación a la Estación base. A esta solicitud, la estación base responde con un mensaje de respuesta de asociación el cual contiene un hash. El Nodo 1, al recibir este mensaje, procede a realizar una prueba de trabajo para determinar el hash y enviar un mensaje de Autenticando en el cual se encuentra incluido el hash que determina para proceder a su validación.

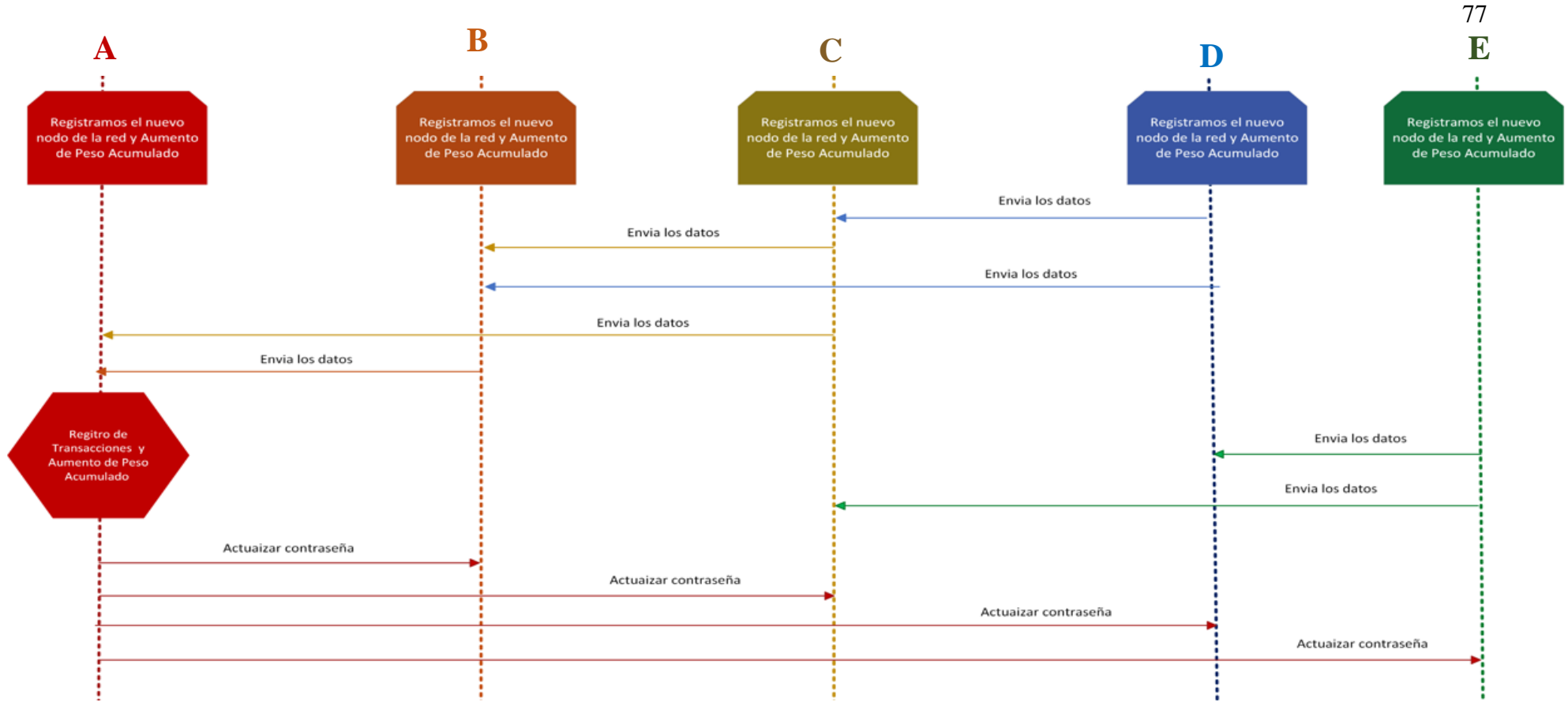
Al recibir la estación base el mensaje de autenticación, proceda a realizar su validación. Si el hash recibido del Nodo 1 es correcto, la estación base envía un ACK de asociación y se procede a aumentar el peso de la estación base ya que se ha unido un nuevo nodo a la red, así como también al envío de un mensaje de actualización de contraseñas a los nodos de la red para evitar que se maneje una contraseña estática.

A continuación, el Nodo 1 podrá realizar el envío de datos, lo cual la estación base lo registrará como una nueva transacción.

Figura 22 .

Diagrama de secuencia de Envió de mensajes y Autenticación



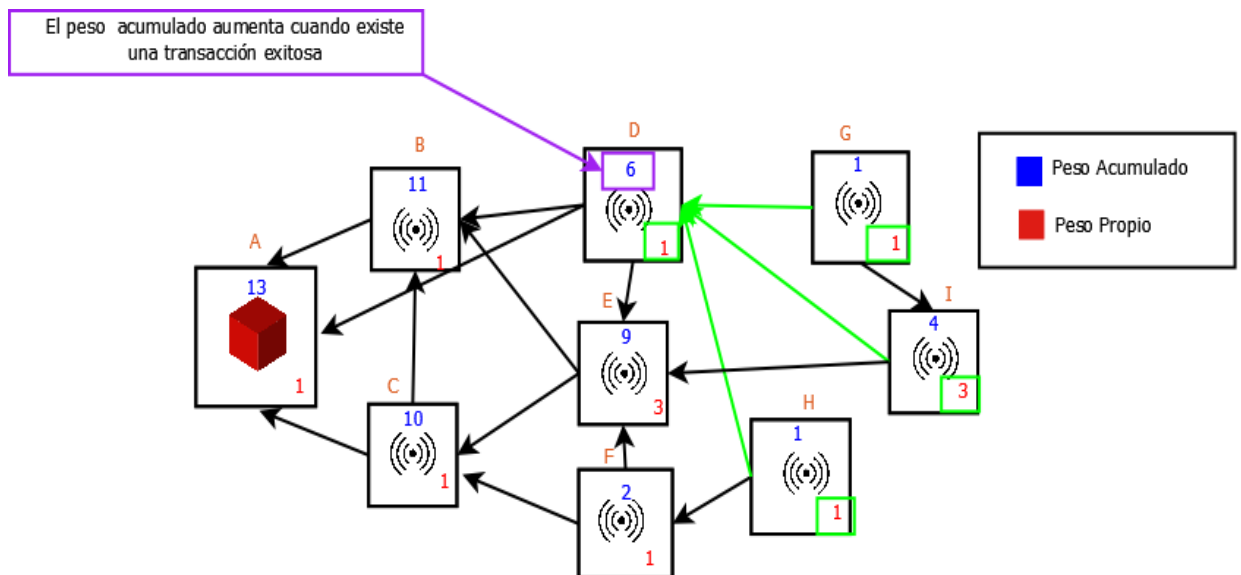


a) Registro de Transferencia y Aumento de Peso Acumulado

Después de realizar una autenticación exitosa, se procede a registrar el nuevo nodo que se a unido a la red y con el las transacciones generadas por el mismo. Esto provoca que el peso acumulado del nodo aumente. El aumento de peso resulta del peso propio de los nodos que están directamente conectados a él. En la Figura 23 se puede observar cómo la estación base y los nodos sensores tienen un peso propio y un peso acumulado. El peso acumulado se calcula sumando el peso propio del nodo con el peso de los nodos conectados directamente a él. Por ejemplo, en el caso del nodo D, el peso acumulado es igual a la suma de $G + I + H +$ peso propio del nodo D los cuales corresponden a los valores 1, 3, 1 y 1 respectivamente dando como resultado un valor de 6 correspondiente al peso del Nodo D.

Figura 23

Aumento de pesos

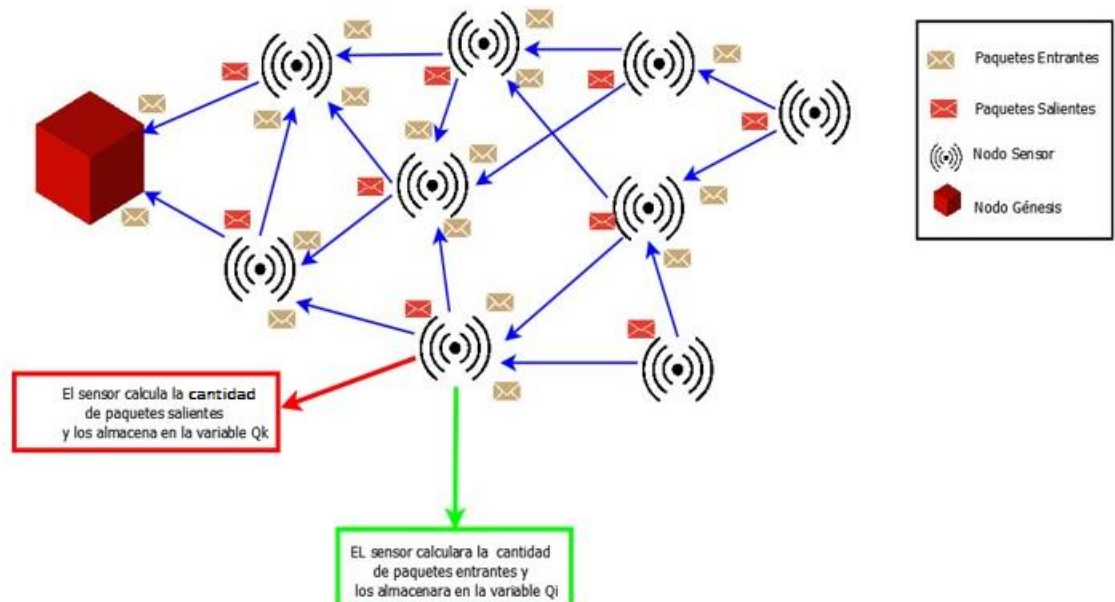


3.2.3.2. Bloque 2: Establecimiento de la red y Aplicación del protocolo SEECR

Para aplicar el protocolo SEECR, es necesario contar con una red que incluya al menos tres o cuatro nodos, ya que solo en ese momento es posible llevar a cabo el conteo de mensajes entrantes y salientes. Como se muestra en la Figura 24, siempre existen diferentes mensajes tanto de entrada como de salida, por lo que el número de mensajes entrantes y salientes debe ser igual. En base a esta lógica, se aplica el protocolo SEECR, que en caso de detectar una cantidad diferente de mensajes entrantes y salientes en conjunto con solicitudes de autenticación insistentes por parte de un nuevo nodo sensor, procederá a cortar la vía de comunicación hacia el sensor, marcándolo como un nodo malicioso e indicando a toda la red a través de un mensaje “Attacker Info” en forma de emisión que no se deben comunicar con este nodo.

Figura 24.

Establecimiento de la red Tangle



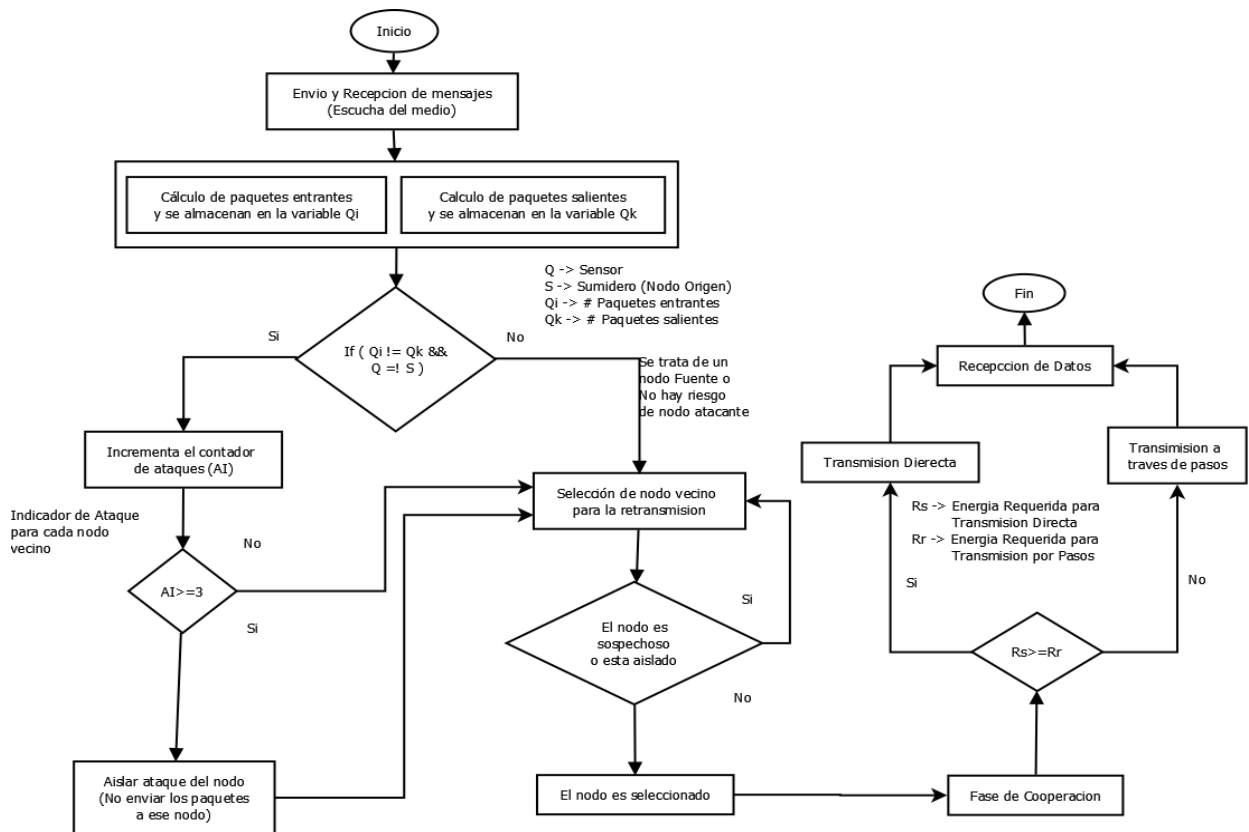
La aplicación del protocolo inicia con el envío y recepción de mensajes, en esta etapa los nodos escucharán al medio en el cual se encuentran. A continuación, se realizará el cálculo de paquetes entrantes; este dato se almacenará en la variable Q_i , de la misma forma se determinará el número de paquetes salientes, almacenando este dato en la variable Q_k .

A continuación, se realiza la verificación de paquetes indicando, que, si el número de paquetes entrantes es diferente al número de paquetes salientes, y además el sensor (Q), no es un sumidero (S) se realizará el incremento del contador de ataques. Si el contador es mayor a tres, automáticamente se corta el envío de mensajes y se aísla el nodo.

En el caso de que los paquetes entrantes y salientes sean iguales, se selecciona un nodo vecino para su retransmisión. Si al momento de buscar el nodo vecino se escoge un nodo aislado o sospechoso, se realizará la búsqueda de un nuevo nodo vecino; pero si se ha seleccionado un nodo con las condiciones adecuadas, se pasa a una fase de cooperación. En esta fase, si la energía requerida para transmisión directa (R_s) es mayor o igual a la energía requerida para transmisión por pasos (R_r), se procede a realizar una transmisión directa, caso contrario se procede a realizar una transmisión por pasos hasta llegar a la etapa final que es la recepción de datos.

Figura 25

Diagrama de Flujo del Funcionamiento del protocolo SEECR



3.2.3.3. Bloque 3: Creación de Bloque Candidato en Estación Base

Los datos obtenidos ‘Transacciones’ de cada uno de los nodos se guardarán dentro de la estación base, que será la encargada de ordenar cada uno de los datos, para formar un bloque candidato. A esta información se le agrega dos elementos esenciales clave a la información existente: el hash del bloque N y el hash del bloque anterior (N-1). Estos elementos se obtuvieron a través del proceso de minado del bloque génesis, que es el primer bloque de la cadena de bloques.

El hash del bloque N es un valor criptográfico que está formado por un conjunto de números y letras únicas. Este valor se genera a partir de la información contenida en el bloque N y desempeña la función de servir como un identificativo de la información que contiene el bloque, así como garantizar su integridad y seguridad.

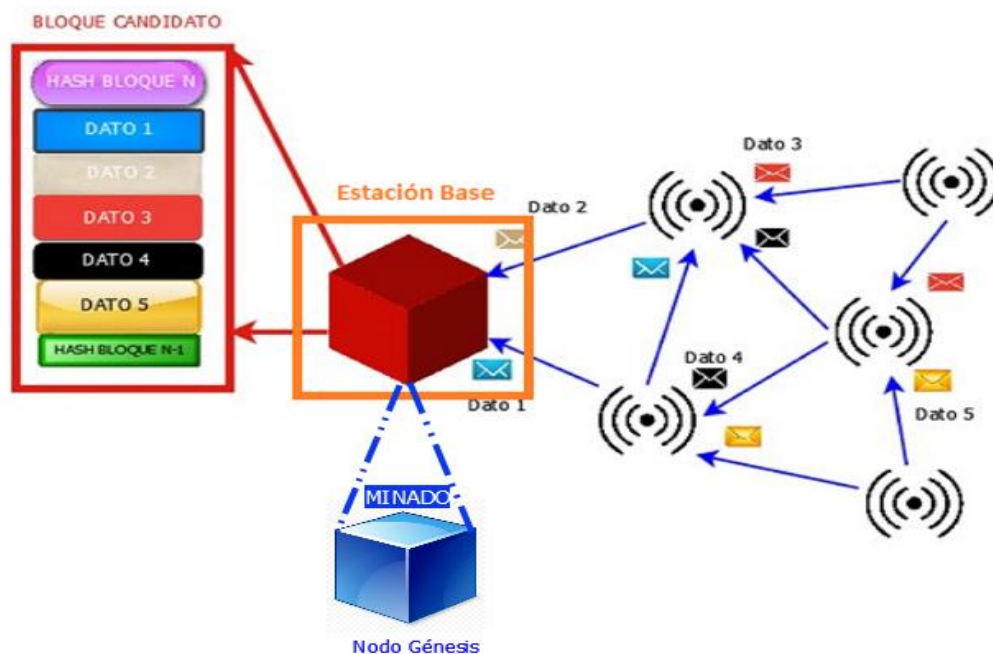
El hash del bloque N-1 se agrega a cada bloque para crear una conexión inmutable entre los bloques de la cadena. Este hash ayuda a identificar cada bloque de tal forma que, si es alterado, la cadena de bloques dejará de reconocer ese bloque y todos los bloques que dependen de él.

Por lo tanto, es crucial que tanto el hash del bloque N como el hash del bloque N-1 se mantengan seguros e inalterados para garantizar la confiabilidad y la inmutabilidad de la cadena de bloques.

La Figura 26 ilustra cómo cada sensor envía sus datos respectivos, representados con colores distintos, a la estación base. Estos datos son organizados en orden de recepción ya cada conjunto de datos se le agrega un hash de bloque N. Al final del conjunto de datos, se agrega el hash del bloque N-1 siendo este dato obtenido del minado realizado al bloque génesis.⁴

Figura 26

Creación del Bloque candidato



⁴ **Bloque génesis:** es el primer bloque de una cadena de bloques por lo cual usa un hash especial que se utiliza para la creación del hash de los bloques siguientes.

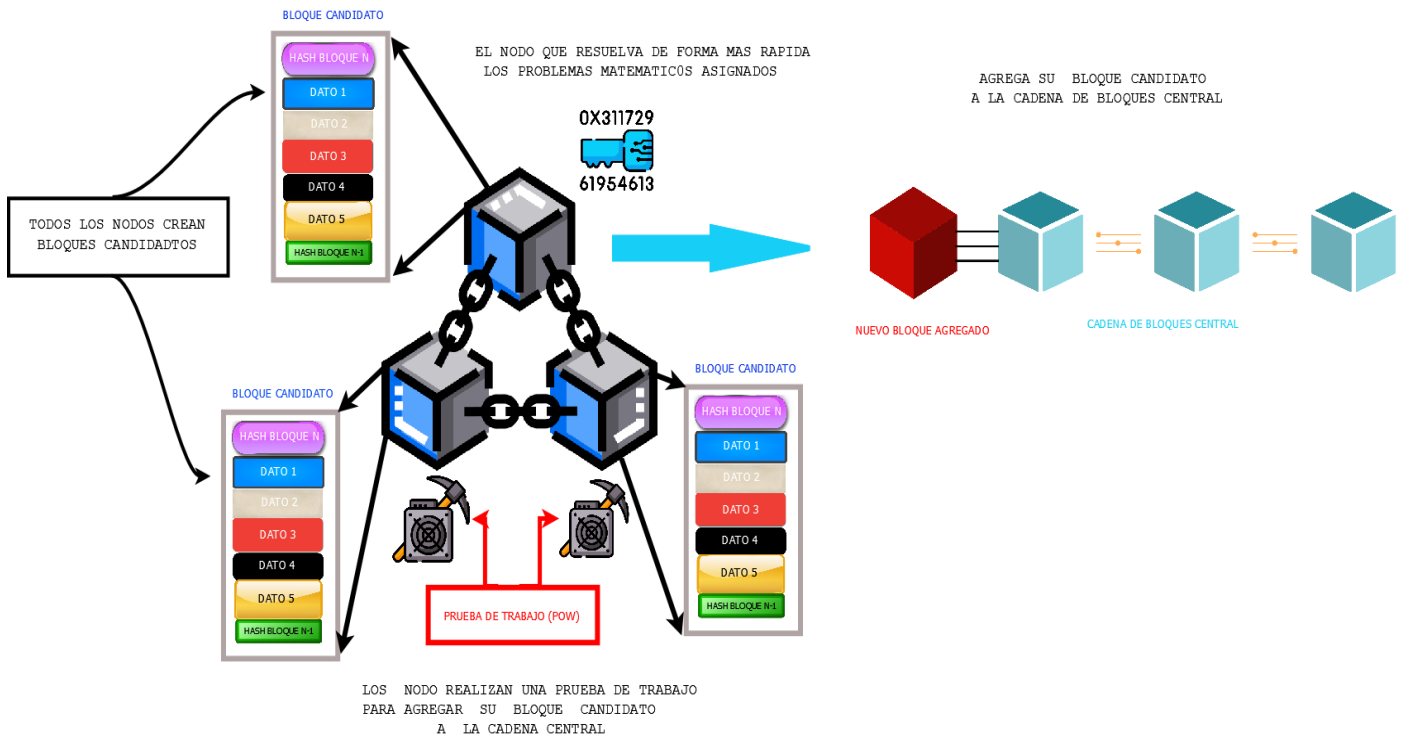
3.2.3.4. Bloque 4: Prueba de Trabajo y Creación de cadena de bloques central

En esta etapa, las estaciones base de las tres islas generan su propio bloque candidato con los datos obtenidos de los nodos sensores de su isla, tal como se indica en la Figura 27. Para que el bloque candidato pueda formar parte de la cadena central, la estación base deberá realizar una prueba de trabajo. Esta prueba consiste en que las estaciones base compitan entre sí para resolver un problema matemático complejo utilizando una gran cantidad de potencia de procesamiento. El problema matemático se basa en un algoritmo hash criptográfico que requiere que la estación base encuentre un número específico, conocido como nonce, que, cuando se agrega a la información del bloque, produce un hash que cumple con ciertas condiciones específicas.

Solo la estación base que logre resolver de manera eficiente la prueba de trabajo y venza en velocidad a los otros nodos, podrá agregar su bloque candidato a la cadena central. En el momento en que se logra agregar el bloque candidato a la cadena central, se procede a actualizar la cadena central mediante la difusión de una copia de esta a todas las estaciones base. De esta forma, las estaciones base de todas las islas tendrán a su disposición la misma información.

Figura 27

Prueba de Trabajo y Creación de Cadena Central



3.2.4. Stakeholders del Proyecto

Con el análisis de situación actual y tomando como referencia el estándar ISO/IEC/IEEE 29148 que indica los lineamientos específicos para el desarrollo de procesos y productos relacionados con el desarrollo de trabajos de Ingeniería. Se indican los siguientes aspectos, los Stakeholders que son los encargados de describir los individuos y/o elementos importantes que afectan de gran manera a los resultados y objetivos que se establecen dentro del trabajo de grado a realizar.

Este estándar internacional es el encargado de proveer un tratamiento unificado de los diferentes procesos y productos, necesarios para el establecimiento de requisitos de ingeniería que influyen dentro del ciclo de vida de un sistema y software.

Los requerimientos se presentan como un elemento importante dentro de este proyecto, debido a que es de gran importancia conocer los diferentes actores, necesidades, condiciones y elementos para que el diseño de la red sea óptimo y aplicable en un medio físico.

Tabla 4

Actores Involucrados

| Actores Involucrados | | |
|-----------------------------|-----------------------|----------------|
| N° | Actor | Función |
| 1 | Carla Vallejo | Desarrolladora |
| 2 | Msc. Fabián Cuzme | Director |
| 3 | Msc. Hernán Domínguez | Asesor |

a) Requerimientos de Diseño

Los criterios considerados para el establecimiento de los diferentes requerimientos, se encuentran determinados de acuerdo a su prioridad, este conjunto de requisitos expresan aspectos claves que se deben tener presentes para el correcto diseño del esquema, como son los requerimientos del esquema de autenticación y los requerimientos de arquitectura; para que de esta forma se pueda cumplir de forma satisfactoria con los objetivos planteados, y lograr así dar solución a algunos problemas existentes en los esquemas de autenticación actuales.

Se ha definido diferentes requerimientos, por ello, se ha fijado distintas abreviaturas que permitirán una mejor comprensión y procesamiento de la información, estableciendo dentro de la Tabla 5 los requerimientos de Stakeholders, requerimientos del esquema de autenticación y arquitectura.

Tabla 5

Nomenclatura de los requerimientos.

| Descripción | Abreviatura |
|--------------------------------|-------------|
| Requerimientos de Stakeholders | StSR |
| Requerimientos de Sistema | SySR |
| Requerimientos de Arquitectura | SrSH |

3.2.6.3. Requerimientos de Stakeholders

Antes de realizar el diseño del sistema se debe tener claro los requerimientos operacionales, los cuales indican el funcionamiento total del sistema, a partir de la especificación de requisitos generales; los mismos que debe cumplir el esquema de autenticación en una red de sensores inalámbricos (WSN).

Tabla 6.

Requerimientos de Stakeholders

| StRS Requerimientos Operacionales | | | | |
|--------------------------------------|--|-----------|-------|------|
| Número | Descripción | Prioridad | | |
| | | Alta | Media | Baja |
| StRS1 | Prueba de Trabajo realizada por los nodos para determinación de contraseñas. | x | | |
| StRS2 | Verificación de nodo honesto por dos nodos de la red. | x | | |
| StRS3 | Envío de paquete de actualización de contraseñas | x | | |
| StRS4 | Creación de canal seguro para envío de datos | x | | |
| StRS5 | Comprobación de paquetes de entrada y salida a través del protocolo SEECR | x | | |
| StRS6 | Mensaje de aviso de nodo malicioso | x | | |
| StRS7 | Tiempo de actualización de contraseñas máximo (1 min) | | x | |
| StRS8 | Verificación de bloque candidato agregado a la cadena central | | | |
| StRS9 | Tiempo de duración de batería del nodo génesis mínimo (3 horas) | x | | x |

3.2.6.4. Requerimientos de Sistema

En el estándar ISO/IEC/IEEE/29148 se indica que los requerimientos del sistema son una parte esencial para el diseño de la red, ya que muestra cual es la función

de cada uno de los elementos que la conforman, e indica cuál será su función dentro del esquema de autenticación.

Tabla 7

Requerimientos de Sistema

| SySR | | | | |
|--|---|------------------|--------------|-------------|
| Requerimientos del Sistema | | | | |
| Abreviatura | Descripción | Prioridad | | |
| | | Alta | Media | Baja |
| Requerimientos de Performance | | | | |
| SySR1 | Los dos primeros nodos sensores se autenticarán con la estación base y los demás se autenticarán entre ellos. | x | | |
| SySR2 | Al realizar una transacción exitosa el peso acumulado de los nodos sensores aumentara. | x | | |
| SySR3 | La estación base o nodo génesis recibirá los datos de los sensores y los registrará | x | | |
| SySR4 | Los nodos génesis realizaran la prueba de trabajo para determinar el hash del bloque anterior | x | | |
| SySR5 | El nodo génesis que resuelva más pronto la prueba de trabajo agregara su bloque candidato a la cadena principal | | x | |
| SySR6 | Todos los nodos génesis de las islas tendrán una copia de la cadena principal. | x | | |
| Requerimientos de Interfaz | | | | |
| SySR7 | Las Estaciones base debe tener conexión a Internet | x | | |
| SySR8 | Comunicación inalámbrica entre nodos de la red. | x | | |
| Requerimientos Físicos | | | | |
| SySR9 | Los nodos de la red IOTA debe tener un bajo consumo energético. | | x | |
| Requerimientos de Seguridad | | | | |
| SySR10 | Los datos enviados por los nodos se encriptarán usando SHA256. | x | | |
| SySR11 | Cada bloque de la cadena contara con una hash anterior y propio. | x | | |
| SySR12 | Aumento de Salt a la contraseña generada por los nodos | x | | |
| Requerimientos de Modo / Estado | | | | |
| SySR13 | La plataforma de simulación proporciona todas las características para la creación de una red de sensores. | | x | |

3.2.6.5. Requerimientos de Arquitectura

Dentro de esta sección se describe las propiedades generales, con las cuales debe contar el hardware y el software; para que el funcionamiento del esquema propuesto sea efectivo.

Tabla 8.

Requerimientos de Arquitectura

| SrSH | | | | |
|---------------------------------------|--|-----------|-------|------|
| Requerimientos de Arquitectura | | | | |
| Abreviatura | Descripción | Prioridad | | |
| | | Alta | Media | Baja |
| Requerimientos de Diseño | | | | |
| SrSH1 | Los nodos de la red IOTA deben poseer un alto nivel de cobertura | X | | |
| SrSH2 | La estación base debe contar con características robustas de memoria y procesamiento | X | | |
| Requerimientos De Hardware | | | | |
| SrSH3 | Bajo consumo de Energía al realizar el proceso de autenticación de nodos | | X | |
| SrSH4 | Mayor alcance con tasas de transmisión bajas | | X | |
| SrSH5 | Duración de Batería | | X | |
| SrSH6 | Tolerancia a la interferencia | X | | |
| SrSH7 | Escalabilidad | X | | |
| SrSH8 | Comunicación Inalámbrica | X | | |
| Requerimientos de Software | | | | |
| SrSH9 | Lenguaje orientado a objetos | X | | |
| SrSH10 | Funcionamiento en sistemas operativo Linux y Windows | | X | |
| SrSH11 | Rápido Procesamiento | X | | |
| SrSH12 | Manejo de librerías y protocolos enfocadas en telecomunicaciones | X | | |
| SrSH12 | Software Libre | | | X |
| SrSH13 | Compatibilidad con Solidity | X | | |

3.2.5. Elección de Hardware para simulación

Las mejores opciones de hardware y software simulado se escogerán de acuerdo con el análisis de requerimientos previamente indicados en la sección anterior, ya que a través de este análisis se buscará diferentes elementos de hardware y software que se acoplen a los requerimientos señalados.

a) Elección de Tecnología inalámbrica para Simulación

Escoger de forma correcta la tecnología utilizada dentro de los nodos sensores que compondrán la red IOTA es de vital importancia, ya que, si existe algún daño o fallo de esta, toda la red de sensores (isla) terminara colapsando.

Tabla 9

Selección de Hardware para el nodo sensor

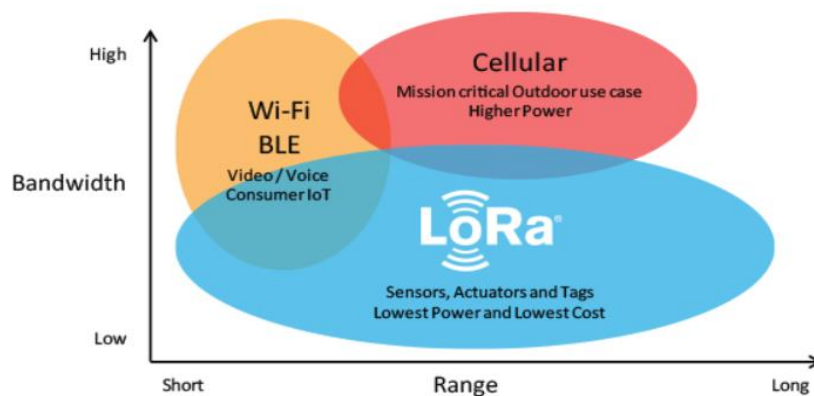
| Hardware Nodo Sensor | Requerimientos | | | | | | Valoración Total |
|-------------------------|----------------|------|------|------|-------|-------|---------------------|
| | STRS8 | SRH1 | SRH4 | SRH5 | SrSH6 | SrSH7 | |
| LoRaWAN | 1 | 1 | 1 | 1 | 1 | 1 | 6 |
| NB-IoT | 1 | 1 | 0 | 1 | 1 | 0 | 4 |
| ZigBee | 1 | 1 | 0 | 1 | 1 | 0 | 4 |

1= cumple 0= no cumple

De acuerdo con los resultados obtenidos en la Tabla 9, se escoge a la tecnología de LoRAWAN como Hardware para la simulación del nodo sensor debido a que obtiene la puntuación de 6 cumpliendo con todos los requerimientos planteados y cuyas características son aptas para la creación de la red IOTA a diferencia de las tecnologías de NB-IoT y ZigBee que no cumplen con el requerimiento SrSH7 indicado en la Tabla 8.

Figura 28.

Características LoRaWAN



Según Quimbita & Salvador (2018) LoRaWAN es una tecnología que garantiza velocidades de datos de 0,3 kbps hasta 50 kbps, que se consideran aceptables para el envío de datos de sensores en tiempo real en IoT. El uso de bajas velocidades de datos asegura el bajo consumo de energía de los dispositivos finales, permitiendo el uso de baterías con una vida útil de varios años, siendo diseñada para sensores y aplicaciones que necesitan enviar pequeñas cantidades de datos a largas distancias.

3.2.6. Elección de Software para Simulación

Debido a que se está realizando una simulación, es de gran importancia escoger el lenguaje de programación y las plataformas de simulación; que permitan desarrollar de la mejor forma el diseño de la red, así como también la implementación de todas las funciones indicadas dentro de la arquitectura.

Tabla 10.

Elección de Plataforma de Simulación

| Plataforma de Simulación | Requerimientos | | | | | | Valoración Total |
|--------------------------|----------------|------|------|------|-------|-------|------------------|
| | STRS8 | SRH1 | SRH4 | SRH5 | SrSH6 | SrSH7 | |
| Omnet ++ 6.0 | 1 | 1 | 1 | 1 | 1 | 1 | 6 |
| NS3 | 1 | 1 | 1 | 1 | 1 | 0 | 5 |
| NS2 | 1 | 1 | 1 | 1 | 1 | 0 | 5 |
| Contiki | 1 | 1 | 0 | 1 | 1 | 0 | 4 |

1= cumple 0= no cumple

De acuerdo con los resultados obtenidos en la Tabla 10, se escoge a Omet++ 6.0 como el Software para la simulación del esquema de autenticación, cumpliendo con una puntuación de 6, cumpliendo así con todos los requerimientos planteados a diferencia de las plataformas de simulación NS3, NS2 y Contiki que no nos permite cumplir con el Requerimiento SrSH7 indicado en la Tabla 8 por lo cual tienen una puntuación de 5 y 4.

OMNeT6.0 ++ en sí mismo no es un simulador de nada concreto, sino que más bien proporciona la infraestructura y herramientas para crear simuladores. Una de las características fundamentales de esta infraestructura es la arquitectura de componentes modular de simulación.(López Lecumberri, 2011). La versión usada para la realización de este trabajo de Investigación se usa Omnet ++ 6.0, el cual cuenta con diferentes características entre las que se aplican correcciones para el visor de gráfico de secuencia en el IDE que no muestra algunos archivos de registro de eventos, además de solucionar fallas de gráficas en Linux en el IDE.

3.3. Fase 2: Diseño del Sistema

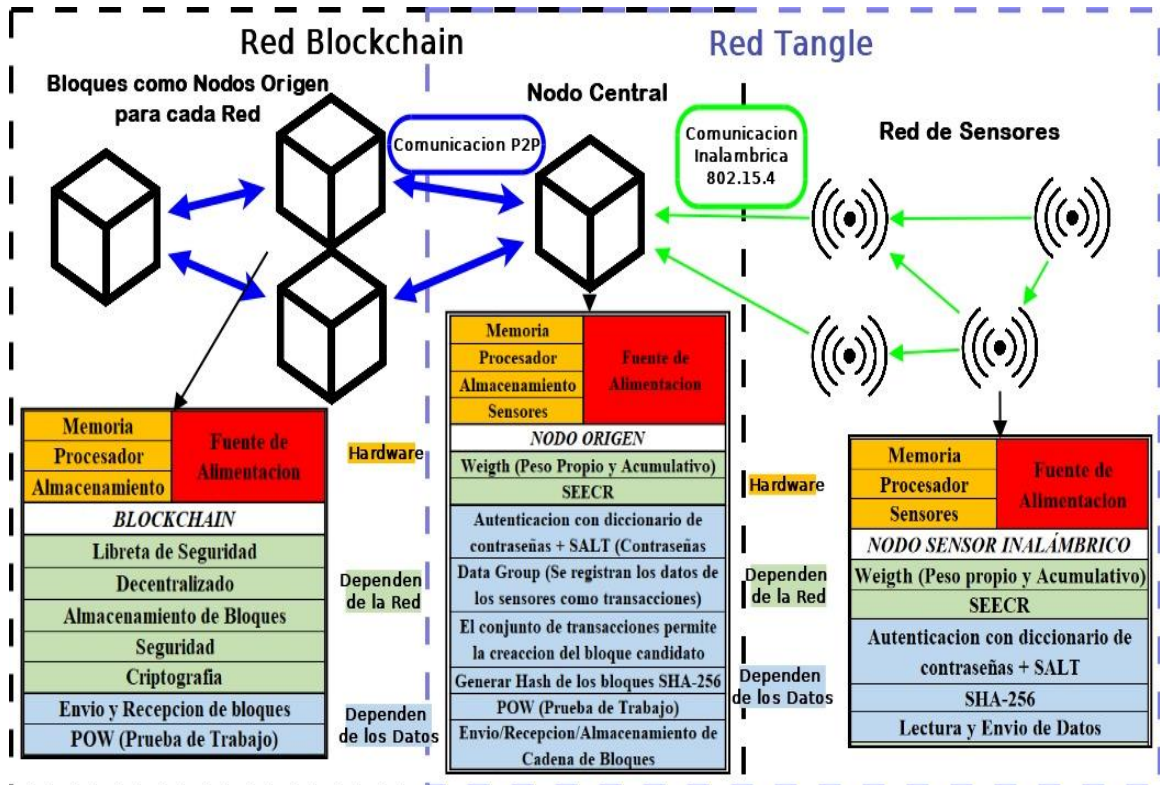
Una vez realizada la elección del software y hardware para la simulación del esquema de autenticación, se establece el diseño del esquema, el cual se explica de manera general a través de las características que tendrá la red y de su funcionamiento. Para explicar de mejor manera los mensajes y procesos que realizan el esquema de autenticación, se explica en detalle uno a uno los mensajes que se envían en primera instancia entre la estación base y los nodos, y en segunda instancia entre los nodos.

3.3.1. Diagrama de Bloques del Sistema

El esquema de autenticación se encuentra conformado por tres elementos importantes: los nodos inalámbricos, la estación base y la cadena de bloques. Cada una de estas partes como se indica en la Figura 29 cuentan con componentes clave como memoria, procesador, batería, entre otros, propios del hardware. También se tiene en cuenta otros aspectos propios de la red, como el manejo de pesos y la aplicación del protocolo SEECR, y otros que trabajan para dar seguridad al envío de datos. Entre estas características se encuentra la autenticación a través de un diccionario de contraseñas y salt, y para evitar el envío de datos en texto plano, estos se encriptan.

Figura 29.

Diagrama de Bloques del Sistema



3.3.2. Diagrama de Funcionamiento del Sistema

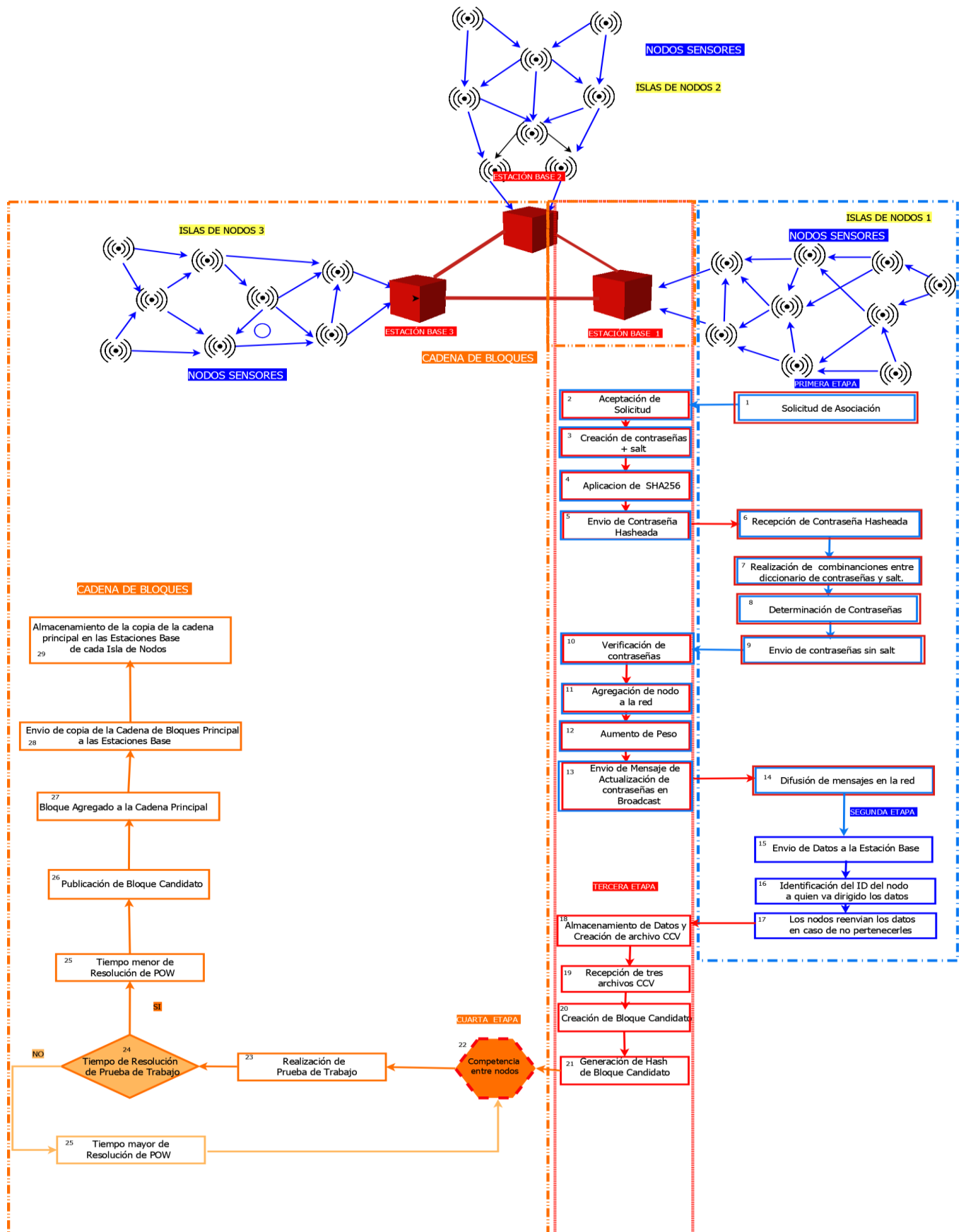
La Figura 30 muestra el funcionamiento de todo el sistema en cada una de sus etapas, desde la autenticación de los nodos hasta la creación del bloque candidato que se desea añadir a la cadena de bloques principal. Para facilitar la comprensión, el proceso se divide en dos etapas. La primera está relacionada con la comunicación y autenticación de cada uno de los nodos hasta llegar a establecer la red Tangle. En la segunda etapa, se centra en el envío de datos desde cada uno de los nodos hasta la estación base para la creación del bloque candidato. Este bloque debe resolver una prueba de trabajo para ser publicado y formar parte de la cadena de bloques central.

Dentro de la imagen, se puede observar cómo se encuentran divididas las funciones que realizan cada elemento de la red dentro del esquema de autenticación.

- El recuadro azul señala todas las funciones que realizan los nodos, algunas de las cuales son realizadas por la estación base.
- El recuadro rojo muestra las funciones que realizan cada una de las estaciones base que forman parte de la red.
- Por último, el recuadro naranja muestra el proceso a seguir para la creación de la cadena de bloques, mismo que se genera al competir las tres estaciones base.

Figura 30.

Diagrama de Funcionamiento del Sistema



3.3.3. Diagrama Lógico de la Red

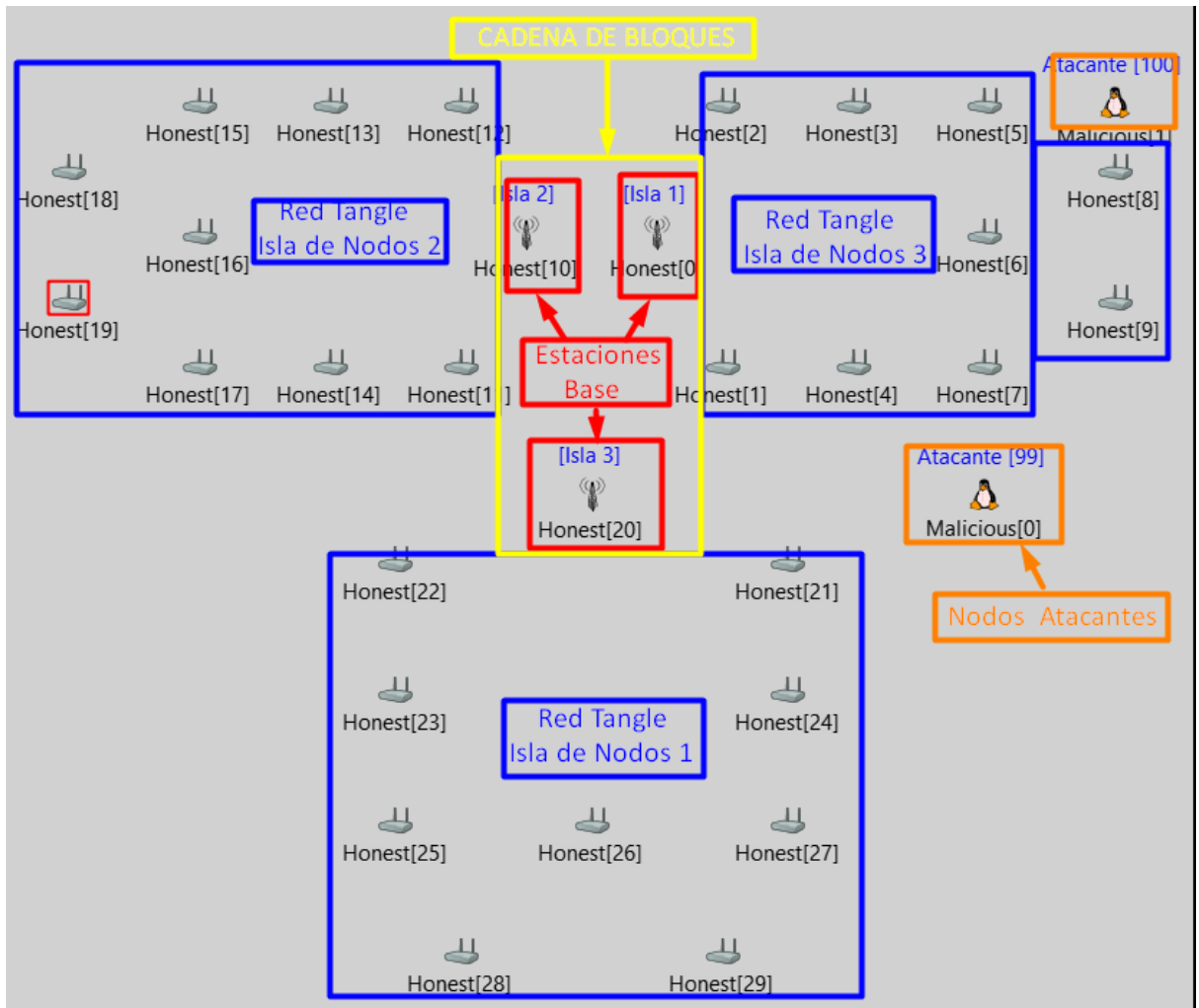
La Figura 31 muestra el diagrama lógico de la red que consta de diferentes componentes, de los cuales tenemos estaciones base e islas de nodos, las islas Tangle están representadas dentro de los recuadros azules, cada una de las cuales consta de nueve nodos sensores, para formar parte de la red cada uno de los nodos deberá autenticarse. Solo los nodos validados pueden enviar datos, el envío de información propia del nodo es unidireccional y de los mensajes de confirmación y petición es bidireccional.

La parte central (marcada con un recuadro amarillo) muestra dentro de la red la parte que representa la cadena de bloques, la misma que se crea a partir de la competencia entre las estaciones base y cuya función será la de recibir la información de cada uno de los nodos y registrarla como transacciones. Al tener un conjunto de transacciones se formará el Bloque candidato, el cual para publicarse dentro de la cadena de bloques central deberá realizar una prueba de trabajo, la misma que consistirá en que la estación base que determine del nonce en el menor tiempo podrá publicar su bloque candidato.

Dentro de los recuadros naranja se muestran los nodos atacantes. Cada uno de ellos genera un ataque distinto. Para el desarrollo de pruebas, se tomó en cuenta el ataque de diccionario, que es un tipo de ataque de contraseñas en el que se utiliza una lista de palabras comunes para intentar adivinar la contraseña de un usuario, y el ataque de denegación de servicio, que busca hacer que un servicio recurso en línea deje de estar disponible para los usuarios legítimos.

Figura 31

Diagrama lógico de la Red



3.4. Fase 3: Implementación de la Red

Para implementar una red, se necesitan varios pasos. Primero, se debe diseñar la topología de la red, es decir, cómo están conectados los diferentes dispositivos y nodos de la red. Esto puede implicar diseñar la disposición física de los dispositivos y determinar qué dispositivos estarán conectados entre sí y cómo se transmitirán los datos entre ellos.

Una vez que se ha diseñado la topología de la red, el siguiente paso es seleccionar y configurar los dispositivos necesarios para implementar la red. Cada

dispositivo debe configurarse correctamente para asegurarse de que funcionen adecuadamente y se comuniquen correctamente con los demás dispositivos de la red.

3.4.1. Inicialización de nodos

En el fragmento del código indicado en la Figura 32 se realiza la inicialización de los procesos necesarios para que cada uno de los nodos determine las conexiones de acuerdo con el archivo de simulación, así como también las posiciones de cada

elemento. Esto se representa a través del nodo configurator



Figura 32

Inicialización de los nodos

```
void HonestModule::initialize()
{
    _initialize();

    EV << "Initialization complete\n";
    scheduleAt(simTime() + exponential(rateMean), msgUpdatePass);
    scheduleAt(simTime() + exponential(rateMean) + 0.6 , msgBLOQ11);
    scheduleAt(simTime() + exponential(rateMean) + 25 , msgStartTx);

    //scheduleAt(simTime() + exponential(rateMean) + 25 , msgSensorData);
}
```

3.4.2. Mensajes Indicadores de procesos

Dentro de la función *HonestModule:handleMessage*, el *switch case* se encarga de procesar diferentes tipos de mensajes que se envían dentro de la red. La función tiene un parámetro *msg* que corresponde al mensaje que se desea procesar, indicando cada uno de los procesos que se están realizando. Entre ellos tenemos la autenticación, las actualizaciones de contraseñas y las notificaciones a todos los nodos de la existencia de un nodo malicioso para su descarte. Esto se logra gracias a que para cada uno de los mensajes se llama a una función específica que se encarga de procesar ese tipo de mensaje. Si el tipo de mensaje no coincide con ninguno de los casos del *switch*, no se realiza ninguna acción adicional.

En la Figura 33 se puede observar que la función *handleMessage* se encarga de manejar los diferentes mensajes que se envían al módulo de simulación *HonestModule*. A través del objeto *cMessage*, este módulo representa los diferentes mensajes dentro de la simulación. El manejo de los diferentes tipos de mensajes se realiza mediante el uso de una estructura *switch* que comprueba el valor de *msg->getKind()* y llama a una función diferente para cada tipo de mensaje.

Figura 33

Mensajes indicadores de procesos

```

void HonestModule::handleMessage(cMessage* msg)
{
    EV << "MessageType: " << msg->getKind() << endl;
    //EV << "getDisplayString: " << msg->getDisplayString() << "." << endl;
    switch (msg->getKind())
    {
        case MessageType::ISSUE:
            caseISSUE();
            break;
        case MessageType::POW:
            casePOW(msg);
            break;
        case MessageType::UPDATE:
            caseUPDATE(msg);
            break;
        case MessageType::AUTH:
            caseAUTH(msg);
            break;
        case MessageType::BLOQ11:
            caseBLOQ11();
            break;
        case MessageType::ASO:
            caseASO(msg);
            break;
        case MessageType::ASORESP:
            caseASORESP(msg);
            break;
        case MessageType::ASOACK:
            caseASOACK(msg);
            break;
        case MessageType::UPDATEASO:
            caseUPDATEASO(msg);
            break;
        case MessageType::UPDATEPASS:
            updatePASS();
            break;
        case MessageType::ATTACKER:
            caseATTACKFR(msg);
    }
}

```

3.4.3. Caso updatePASS

La Figura 34 muestra dentro del código como la función *updatePASS* se encarga de actualizar una variable de clase llamada *actualPassHash*, en la cual se guarda la contraseña actual, con un valor generado por la función *getPassDinamic*, encargada de generar las contraseñas de forma dinámica. Cuando esto se realiza, se incrementa la variable de class *passLimite* en 1, lo cual ayuda a hacer más compleja la

contraseña. La función también comprueba si *passLimite* es menor o igual a *txLimite*. Si esta condición se cumple, se establece una programación para que la función se ejecute de nuevo en el futuro utilizando la función *scheduleAt*.

En general, la función *updatePASS* es usada para la generación de la contraseña y para indicar la ID del nodo estación base a cada una de las islas según corresponda. Este proceso se realiza una sola vez. Esta ID será usada luego por cada nodo para saber hacia dónde enviar los datos de los sensores.

Figura 34

Método *updatePASS()*

```
void HonestModule::updatePASS()
{
    if (passlimite <= txlimite)
    {
        if (firstPass == false)
        {
            pastSaltHash = actualSaltHash;
            EV << "****PastSaltHash: " << pastSaltHash << endl;

            //Proceso de asignar BSTA
            if (ID == "[1]" || ID == "[2]" || ID == "[3]" || ID == "[4]" || ID == "[5]" || ID == "[6]" || ID == "[7]" || ID == "[8]" || ID == "[9]" ) {
                nodoBSTA = "[0]";
            } else if (ID == "[11]" || ID == "[12]" || ID == "[13]" || ID == "[14]" || ID == "[15]" || ID == "[16]" || ID == "[17]" || ID == "[18]" ||
            } else if (ID == "[21]" || ID == "[22]" || ID == "[23]" || ID == "[24]" || ID == "[25]" || ID == "[26]" || ID == "[27]" || ID == "[28]" ||
                nodoBSTA = "[20]";
            }
        }
        actualPassHash = getPassDinamic();
        firstPass = false;

        scheduleAt(simTime() + 5 - peso*0.01, mslUpdatePass);
        EV << "Siguiente actualizacion en t= " << simTime() + 5 - peso*0.1 << " (s)." << endl;
        passlimite++;
        currentSleep += sx1272_sleep;
    }
    else {
        EV << "Actualizaciones de pass terminadas..." << endl;
    }
}

```

Indicamos el limite de la contraseña

Verificamos si es la primera contraseña

Enviamos a los nodos honestos el ID de la Estacion Base

Generamos una contraseña dinámica

Disminuimos el tiempo en el cual se crea una nueva contraseña dinámica

La Figura 35 muestra la función llamada *getPassDinamic* misma que tiene como objetivo generar una contraseña de forma dinámica. Para ello, abre dos archivos *CSV* llamados *passDict.csv* y *salDict.csv*. Luego, genera un número aleatorio entre 1 y la longitud de las contraseñas en *passDict.csv*. Para ello, lee las líneas del archivo *passDict.csv* hasta que encuentra la línea correspondiente al número aleatorio generado. Realiza el mismo proceso con el archivo *salDict.csv*. A continuación, procede a cerrar los archivos y a concatenar la *contraseña* y la *salt* leídas de los archivos *CSV*. Luego,

genera un hash de la *salt* utilizando la función *sha256* y devuelve el hash de la concatenación de la contraseña y la *salt*.

Figura 35.

Obtención de pass aleatoria

```
string AbstractModule::getPassDinamic()
{
    EV << "Obteniendo la pass dinamica..." << endl;

    //leer contraseñas del diccionario
    //std::string path = "./data/pass" + ID + ".csv";

    //***** Se obtiene primero una de pass aleatoria de las 200 disponibles *****
    std::string path = "./data/passDict.csv";
    std::fstream file;
    file.open(path, std::ios::in);

    if(!file.is_open()) throw std::runtime_error("No se pudo abrir el CSV!");
    std::string line;
    int i = 1;
    int randPass = intrand(passDictLength) + 1; // Genera un entero aleatorio desde 0 hasta Length
    string input1, output1;

    while(getline(file, line))
    {
        if (i == randPass)
        {
            input1 = line;
            break;
        }
        i++;
    }
    file.close();
    EV << "Pass: " << input1 << endl;

    //***** Se obtiene una salt aleatoria de las 50 disponibles *****
    std::string path1 = "./data/salDict.csv";
    std::fstream file1;
    file1.open(path1, std::ios::in);

    if(!file1.is_open()) throw std::runtime_error("No se pudo abrir el CSV!");
    std::string line1;
    int j = 1;
    int randSalt = intrand(saltDictLength) + 1;
    string input2, output2;

    while(getline(file1, line1))
    {
        if (j == randSalt)
        {
            input2 = line1;
            break;
        }
        j++;
    }
    file1.close();
    EV << "Salt: " << input2 << endl;
}
```

3.4.4. Petición de Asociación

En la Figura 36 se observa que la función *caseBLOQ11* se utiliza para realizar un proceso de asociación entre diferentes nodos de una red. La función tiene varios *if* anidados, cada uno de los cuales comprueba una condición en relación con un nodo específico de la red. Si se cumple la condición, se ejecuta una acción que envía una petición de asociación *sendASO* al nodo correspondiente. La acción también actualiza una variable local llamada *currentTx* y establece la variable *nodoActAso* en "true". Por último, muestra un mensaje en pantalla que indica que se ha enviado la petición de asociación.

En general, la función *caseBLOQ11* se utiliza para enviar peticiones de asociación a diferentes nodos de una red, y lleva un registro de los nodos a los que se les ha enviado una petición, mostrando también un mensaje en pantalla.

Figura 36.

Petición de Asociación

```

void HonestModule::caseBLOQ11()
{
    //EV << "Bloque 1 paso 1 con ID->" << ID << " y con flag-> " << nodo1Aso << endl;
    if (!nodo1Aso && ID == "[1]")
    {
        EV << "Nodo " << ID << " envia peticion de asociacion!" << endl;
        sendASO(ID);
        currentTx += sx1272_tx;
    }
    if (nodo1Aso && !nodo2Aso && ID == "[2]")
    {
        EV << "Nodo " << ID << " envia peticion de asociacion!" << endl;
        sendASO(ID);
        nodo2Aso = true;
        currentTx += sx1272_tx;
    }
    if (nodo1Aso && !nodo3Aso && ID == "[3]")
    {
        EV << "Nodo " << ID << " envia peticion de asociacion!" << endl;
        sendASO(ID);
        nodo3Aso = true;
        currentTx += sx1272_tx;
    }
    if (nodo1Aso && !nodo4Aso && ID == "[4]")
    {
        EV << "Nodo " << ID << " envia peticion de asociacion!" << endl;
        sendASO(ID);
        nodo4Aso = true;
        currentTx += sx1272_tx;
    }
    if (nodo1Aso && !nodo5Aso && ID == "[5]")
    {
        EV << "Nodo " << ID << " envia peticion de asociacion!" << endl;
        sendASO(ID);
        nodo5Aso = true;
        currentTx += sx1272_tx;
    }
}

```

La función se encarga de enviar un mensaje de tipo *msgASO* a través de la puerta de salida *out* del módulo. En la Figura 35 se evidencia que antes de enviar el mensaje, la función establece el contenido del mensaje a una estructura de datos llamada *dataASO*, que contiene el valor del parámetro *ID*. Luego, la función envía un mensaje de broadcast sobre todas las salidas disponibles en la puerta de salida *out*, enviando una copia del mensaje a cada una de ellas.

Figura 37

Metodo send ASO

```
//Broadcast para el proceso de Autenticación - Bloque 1
void AbstractModule::sendASO(string ID)
{
    auto data = new dataASO;
    data->ID = ID;

    for(int i = 0; i < gateSize("out"); i++)
    {
        auto copyData = new dataASO;
        copyData->ID = data->ID;

        msgASO->setContextPointer(copyData);
        send(msgASO->dup(), "out", i);
    }
    currentTx += sx1272_tx;

    delete data;
}

```

Indica cuantas salidas tiene el nodo

Envía el Mensaje de Petición de Asociacion

3.4.5. Respuesta de Asociacion

En la función *caseASO* indicada en la Figura 38 se recibe un parametro llamado *msg*, que corresponde a un mensaje recibido por el módulo. La función se encarga de procesar el mensaje *msgASO* y determinar si el módulo debe enviar una respuesta o no.

La función tiene varios *if* anidados, cada uno de los cuales comprueba una condición en relación con el ID del módulo y el ID del nodo que envió el mensaje *msgASO*. Si se cumple la condición, se establece la variable *validID* en *true*.

Por ejemplo, el primer *if* comprueba si el ID del módulo es "[0]". Si es así, establece la variable *validID* en *true*, indicando que el módulo debe enviar una respuesta.

De manera similar, el segundo *if* comprueba si el ID del módulo es "[1]" y, si es así, también establece la variable *validID* en *true*.

En general, el *caseASO* es el encargado de determinar si un módulo debe enviar una respuesta a una petición de asociación enviada por otro nodo de la red.

Figura 38

Respuesta de Asociación

```
void HonestModule::caseASO(cMessage* msg){
    auto data = (dataASO*)msg->getContextPointer();
    bool validID = false;
    //EV << "RECIBIDO MSG_ASO con ID: " << data->ID << endl;

    //Responde a todos los conectados
    if(ID == "[0]")
    {
        validID = true;
    }
    //Responde a todos los conectados
    if(ID == "[1]")
    {
        validID = true;
    }
    //Respuesta a nodo [3]
    if(ID == "[2]" && data->ID == "[3]")
    {
        validID = true;
    }
    //Respuesta a nodo [4]
    if(ID == "[3]" && data->ID == "[4]")
    {
        validID = true;
    }
    //Respuesta a nodo [5]
    if(ID == "[6]" && data->ID == "[5]")
    {
        validID = true;
    }
    //Respuesta a nodo [6]
    if(ID == "[3]" && data->ID == "[6]")
    {
        validID = true;
    }
}
```

El siguiente código mostrado en la Figura 39 se ejecuta solo si la variable *validID* es *true*. Realiza acciones como mostrar un mensaje en la consola con información sobre el *ID* del nodo que envió el mensaje *msgASO* y el *ID* del módulo que lo recibió. También se muestra en pantalla un mensaje llamado *Recibida ASO!*. Además, se envía un mensaje de tipo *msgAuthReq* al nodo que envió el mensaje *msgASO*, con el ID del módulo que lo recibió.

Este código se utiliza para procesar un mensaje de petición de asociación y enviar una respuesta de autenticación al nodo que envió la petición. La respuesta de

autenticación incluye información sobre el ID del módulo y se utiliza para establecer una conexión segura entre los nodos.

Figura 39

Enviamos la respuesta al mensaje de Autenticación

```

if (validID)
{
    bubble("Recibida ASO!");
    EV << "RECIBIDO MSG ASO con ID: " << data->ID << endl;
    EV << "Enviando hash para PoW: " << actualPassHash << "\nhacia el nodo " << data->ID << "; desde el nodo " << ID << endl;
    sendAuthReq(data->ID, ID),
    validID = false;
    currentTx += sx1272_tx;
}

```

3.4.6. Mensaje de Autenticación

En el caso de autenticación exitosa, se mostrará una burbuja indicando que la autenticación se ha realizado de forma correcta.

A continuación, se procederá a validar el hash dentro de la condición *If* en la cual se toma en cuenta el hash actual y el anterior para empezar a realizar diferentes combinaciones de contraseñas y salt hasta encontrar el hash correcto, si esto se cumple de forma adecuada el nodo enviara la salt encontrada para su validación, y se indicará que se ha validado correctamente, caso contrario se mostrara el mensaje de que no se valida. Si la respuesta es positiva se usará el método *sendAuthAck* para realizar el envío de un ACK de confirmación.

En la Figura 40 se observa que la primera línea de código obtiene los datos del mensaje y los almacena en una variable llamada *data*. Estos datos están almacenados en el contexto del mensaje como una estructura de datos *dataAuth*.

La segunda línea de código es una condición que evalúa si el ID del módulo es igual al ID del nodo que envió el mensaje. Si se cumple esta condición, se muestra un mensaje en una burbuja llamada *Recibida AUTH*.

Luego, se ejecutan sentencias *EV* que muestran en la salida información sobre el mensaje *AUTH* recibido, incluyendo el ID del nodo que lo envió y el hash del mensaje. En la séptima línea de código, se evalúa si el hash del mensaje es igual al valor de la variable *actualSaltHash* o a la variable *pastSaltHash*. Si se cumple esta condición, se ejecuta la octava línea de código, que muestra en la salida un mensaje indicando que el hash del mensaje ha sido recibido y validado.

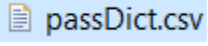
La novena línea de código llama a la función *sendAuthAck* pasándole como argumentos el ID del nodo que envió el mensaje y el ID del nodo actual. Esta función envía un mensaje de confirmación de autenticación (*AUTH_ACK*) al nodo que envió el mensaje, si esto se realiza de forma correcta se procede incrementar el valor de la variable *peso* en 1.

Figura 40

Función *caseAUTH*



3.4.7. Función *powHASH*

El código indicado en la Figura 41 se encarga de realizar un proceso de autenticación. La función comienza imprimiendo un mensaje en pantalla y luego abriendo un archivo CSV llamado *passDict.csv*  y leyendo su contenido línea a línea en un bucle *while*. Cada línea se almacena en la variable *line*. Luego, se asigna el valor de *line* a la variable *input1*.

Dentro del bucle, se llama a la función *checkSalt* pasándole los valores de *input1* y *passHash* como parámetros. Esta función devuelve una cadena de caracteres que se almacena en la variable *output1*. Luego, se concatenan *input1* y *output1* y se calcula el hash SHA-256 de la concatenación. Si el valor del hash es igual al valor de *passHash*, se llama a la función *sendAuth* y se sale del bucle. Si no es así, se sigue iterando hasta que se encuentra una coincidencia o se acaba el archivo.

Para entenderlo de manera clara, esta función busca en un archivo CSV una línea que, cuando se concatena con el valor devuelto por la función *checkSalt*, produce un hash SHA-256 igual al valor de *passHash*. Si se encuentra una coincidencia, se llama a la función *sendAuth*. En caso contrario, no se hace nada.

Figura 41

Método *powHash*

```

void HonestModule::powHash(string passHash, string IDRx, string IDTx)
{
    EV << "Proceso de autenticacion..." << endl;
    // Leer diccionario de pass
    std::string path = "./data/passDict.csv";
    std::fstream file;
    file.open(path, std::ios::in);

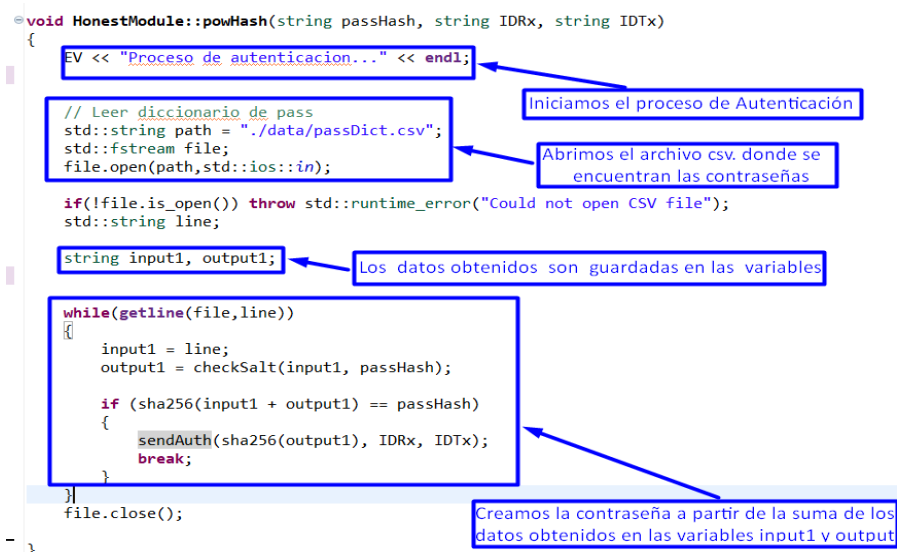
    if(!file.is_open()) throw std::runtime_error("Could not open CSV file");
    std::string line;

    string input1, output1;

    while(getline(file,line))
    {
        input1 = line;
        output1 = checkSalt(input1, passHash);

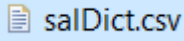
        if (sha256(input1 + output1) == passHash)
        {
            sendAuth(sha256(output1), IDRx, IDTx);
            break;
        }
    }
    file.close();
}

```



- Iniciamos el proceso de Autenticación**: Points to the initial message print statement.
- Abrimos el archivo csv. donde se encuentran las contraseñas**: Points to the file opening code.
- Los datos obtenidos son guardadas en las variables**: Points to the declaration of *input1* and *output1*.
- Creamos la contraseña a partir de la suma de los datos obtenidos en las variables *input1* v *output1***: Points to the concatenation and hashing logic inside the *while* loop.

3.4.8. Función checkSalt

La función comienza abriendo un archivo CSV llamado *salDict.csv*  y leyendo su contenido línea a línea en un bucle while. Cada línea se almacena en la variable *line1*. Luego, se asigna el valor de *pass* a la variable *input1* y se asigna el valor de *line1* a la variable *input2*.

Dentro del bucle, se concatenan *input1* y *input2* y se calcula el hash *SHA-256* de la concatenación. Si el valor del hash es igual al valor de *passHash*, se imprime un mensaje en pantalla y se sale del bucle. Si no es así, se incrementa un contador y se sigue intentando hasta que se encuentra una coincidencia o se acaba el archivo.

En conclusión, esta función busca en un archivo CSV una línea que, cuando se concatena con el valor de *pass*, produce un hash SHA-256 igual al valor de *passHash*. Si se encuentra una coincidencia, se devuelve la línea encontrada. En caso contrario, se devuelve el valor de la última línea del archivo.

Figura 42

Implementación de la Salt

```
string HonestModule::checkSalt(string pass, string passHash){
    // Leer diccionario de salt
    std::string path1 = "../data/salDict.csv";
    std::fstream file1;
    file1.open(path1, std::ios::in);
    if(!file1.is_open()) throw std::runtime_error("Could not open CSV file");
    std::string line1;
    int i = 1;
    string input1, input2, output1;
    input1 = pass;
    while(getline(file1, line1))
    {
        input2 = line1;
        //EV << "pass: " << input1 << " salt: " << input2 << endl;
        output1 = input1 + input2;
        //EV << "pass + salt: " << sha256(output1) << " hash: " << passHash << endl;
        if (sha256(output1) == passHash)
        {
            EV << "Hash encontrado en el intento: " << i << "\nEnviando salt hacia la STA para validacion " << endl;
            EV << "Pass: " << input1 << "\nSalt: " << input2 << endl;
            break;
        }
        i++;
    }
    file1.close();
    return input2;
}
```

Abrimos el archivo salDict.csv y leemos su contenido

Guardamos el valor de la contraseña en la variable input

Concatenamos input1 e input2 y se calcula el hash

3.4.9. ACK de Confirmación

El funcionamiento del código mostrado en la Figura 43 consiste en que, cuando se recibe un mensaje *ASO_ACK*, se extrae la información del mensaje y se almacena en una variable llamada *data*. Luego, se comprueba si el ID del nodo actual coincide con el ID almacenado en *data*. Si es así, se muestra un mensaje en la consola y se establece una bandera llamada *nodo1Aso* que indica que el nodo está asociado.

Figura 43

Código ACK de Confirmación

```

void HonestModule::caseASOACK(cMessage* msg){
    auto data = (dataAuthAck*) msg->getContextPointer();
    if(ID == data->ID)
    {
        EV << "RECIBIDO MSG ASO ACK para ID: " << data->ID << endl;
        bubble("Recibida ASO ACK!");
        EV << "Nodo asociado correctamente!" << endl;
        //nodo1Aso = true; //flag que indica que nodo 1 esta asociado
    }
}

```

Diagrama de anotaciones para el código de la Figura 43:

- Creemos el mensaje que se va a enviar (punto a `msg->getContextPointer()`)
- Guardamos la ID del nodo dentro del metodo (punto a `data->ID`)
- Si el nodo cumple la condición (punto a `if(ID == data->ID)`)
- Mostramos el mensaje ASO ACK (punto a `bubble("Recibida ASO ACK!");`)
- Se indica el nodo asociado y la entrada por la cual a sido asociado (punto a `EV << "Nodo asociado correctamente!" << endl;`)

A continuación, en la Figura 44 se muestra una serie de condicionales que comprueban el ID del nodo actual y el ID almacenado en *data*, y establecen la bandera *nodo1Aso* y programan un mensaje llamado *msgBLOQ11* para enviarse en el futuro.

Figura 44

Condiciones para realizar la asociación de un nodo

```

// ISLA 1
if (ID == "[2]" && !nodo1Aso)
{
    nodo1Aso = true; //flag que indica que nodo 1 esta asociado
    scheduleAt(simTime() + exponential(rateMean), msgBLOQ11);
}

```

Diagrama de anotaciones para el código de la Figura 44:

- Identificamos el ID y vemos que sea diferente del nodo 1 (punto a `if (ID == "[2]" && !nodo1Aso)`)
- Se indica el nodo asociado y la entrada por la cual a sido asociado (punto a `nodo1Aso = true;`)

3.4.10. Mensaje de UPDATEASO

Cuando se recibe un mensaje *UPDATEASO* los nodos deben actualizar sus pesos propios debido a que se ha asociado un nuevo nodo, para lo cual el código creado extrae la información del mensaje para luego almacenarlo en una variable llamada *data*.

Luego, como se muestra en la Figura 45 se busca en el vector *tangleTable* si ya existe una entrada que combina el ID del nodo principal y el ID del nodo asociado teniendo dos opciones:

- Si ya existe, se muestra un mensaje en la consola y se eliminan el mensaje y la información almacenada en *data*.
- Si no existe, se muestra un mensaje en la consola y se agrega la entrada al vector *tangleTable*.

A continuación, se busca en el vector *tangleTableInd* si existe el ID del nodo principal, se aumenta el peso del nodo y se agrega el ID del nodo asociado al vector. También se tiene una respuesta por consola misma que indica el peso actual del nodo, pero no se llama a la función *spreadAso*, ni se actualizan las variables *currentRx* y *currentTx*.

Figura 45

Código UPDATE ASO

```

void HonestModule::caseUPDATEASO(cMessage* msg){
    auto data = (dataUpdateAso*) msg->getContextPointer();
    //if (std::find(tangleTable8.begin(), tangleTable8.end(), data->ID) != tangleTable8.end() && std::find(tangleTable.begin(), tangleTable.end(), data->IDmaster + data->ID) != tangleTable.end())
    {
        EV << "El nodo " << data->ID << " asociado por " << data->IDmaster << " ya se registro!\n";
        delete data;
        delete msg;
        return;
    }
    EV << "Registrando nodo asociado " << data->ID << "\n";
    bubble("Nuevo nodo asociado!");
    tangleTable.push_back(data->IDmaster + data->ID);
    // Actualizar Peso
    for (auto nodeID : tangleTableInd)
    {
        if (nodeID == data->IDmaster)
        {
            peso++;
            tangleTableInd.push_back(data->ID);
        }
    }
}

```

Determinamos si el mensaje de actualización a sido registrado

Indicamos el nodo que se a registrado y cual nodo lo registrar

Creamos un mensaje informativo en la simulación

Si no existe el registro lo creamos y actualizamos el peso

Este código muestra en la consola el peso actual del nodo y crea una copia de la información almacenada en *data*. Luego, llama a la función *spreadAso* para enviar mensajes en modo broadcast junto con el módulo remitente del mensaje y la copia de

la información. También se aumentan las variables *currentRx* y *currentTx* con las variables *sx1272_rx* y *sx1272_tx*, respectivamente.

Figura 46

Cálculo de peso actual del nodo

```
EV << "Peso del nodo " << ID << " es de: " << peso << endl;
auto copyData = new dataUpdateAso;
copyData->ID = data->ID;
copyData->IDmaster = data->IDmaster;
spreadAso(msg->getSenderModule(), copyData);
currentRx += sx1272_rx;
currentTx += sx1272_tx;
```

Guardamos los datos del nuevo dispositivo asociado

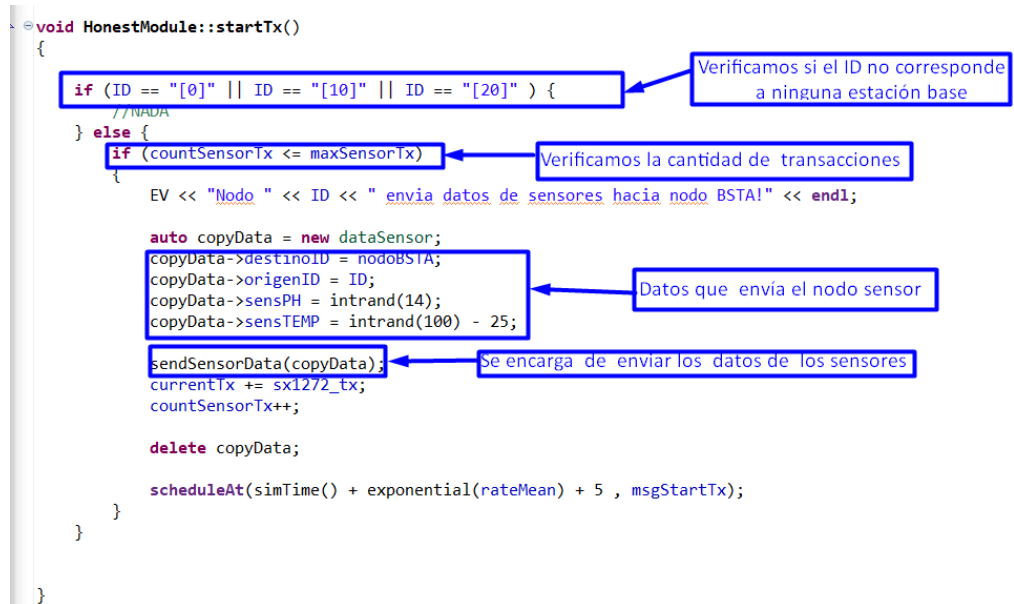
Enviamos mensajes de Broadcast

3.4.11. Función startTX

El objetivo del código mostrado en la Figura 47 es enviar datos desde los sensores a la Estación Base. La función *startTx* se ejecuta en un intervalo regular, controlado por la función *scheduleAt* y el parámetro *rateMean*.

La función comienza verificando si el ID del dispositivo es "[0]", "[10]" o "[20]". Si es así, no se hace nada debido a que son los IDs de las estaciones base de las diferentes islas. Si no es así, se verifica si el contador de transmisiones de datos de sensores *countSensorTx* es menor o igual al máximo número permitido de transmisiones *maxSensorTx*. Si esto es cierto, se envían los datos de los sensores a través de la función *sendSensorData*. Los datos incluyen el ID del destino *nodo BSTA*, el ID del origen (ID del dispositivo actual) y los valores aleatorios generados para dos sensores: *sensPH* y *sensTEMP*. Después de enviar los datos, se aumenta el contador de transmisiones *countSensorTx* y se programa una nueva ejecución de *startTx* para un tiempo en el futuro, utilizando el parámetro *rateMean* como tasa de transmisión.

Figura 47

Función *startTx*

3.4.12. Función *caseSensor Data*

Este código se encarga de procesar mensajes que contienen datos de un sensor y llevar a cabo diferentes acciones dependiendo de si el destinatario del mensaje es el módulo que está ejecutando este código o no.

Si el destinatario del mensaje no es el módulo que está ejecutando el código, se crea una copia del mensaje y se reenvía a su destinatario original.

Si el destinatario del mensaje es el módulo que está ejecutando el código, se almacenan los datos del sensor en una lista y se cuenta el número de veces que se ha recibido un mensaje de este tipo. Si se han recibido suficientes mensajes de este tipo, se considera que se ha completado una transacción y se llama a una función para almacenar esta transacción.

En la Figura 48 se observa a detalle como la función *caseSensorData* se encarga de procesar mensajes de tipo *SENSORDATA*. Esto se realiza en varios pasos:

1. Se crea un puntero data a un objeto *dataSensor* que es extraído del contexto del mensaje mediante *msg->getContextPointer ()*. Este objeto *dataSensor* contiene datos de un sensor, como el ID del nodo que envió los datos, el ID del destinatario de los datos, y los valores de PH y temperatura medidos por el sensor.
2. Se comprueba si el ID del destinatario de los datos (almacenado en *data->destinoID*) coincide con el ID del módulo que está procesando el mensaje (almacenado en ID). Si no coincide, se reenvía el mensaje a su destinatario creando una copia del objeto *dataSensor* y utilizando la función *sendSensorData* para enviarla.
3. Si el ID del destinatario de los datos coincide con el ID del módulo que está procesando el mensaje, se procesan los datos de sensor. Se imprimen estos datos y se aumenta en uno la variable *countNodeTrans*. Si *countNodeTrans* es menor que 9, se almacenan los datos en la lista *actualTransaction* y en la lista *actualTransactionData*. Si *countNodeTrans* es igual a 9, se llama a la función *storeTransaction* y se reinicia *countNodeTrans* a 0. La función *storeTransaction* se encarga de almacenar la transacción actual (que consta de los datos de sensor de 9 nodos diferentes) en algún lugar (probablemente en una base de datos o en una cadena de bloques).

Figura 48

Creación de Información del Sensor

```

void HonestModule::caseSensorData(cMessage* msg)
{
    auto data = (dataSensor*) msg->getContextPointer();

    if (data->destinoID != ID)
    {
        EV << "Reenviando a ID DESTINO: " << data->destinoID << endl;
        auto copyData = new dataSensor;
        copyData->destinoID = data->destinoID;
        copyData->origenID = data->origenID;
        copyData->sensPH = data->sensPH;
        copyData->sensTEMP = data->sensTEMP;

        sendSensorData(copyData);
        currentTx += sx1272_tx;
        delete copyData;
    } else {
        bubble("Recibida SENSOR_DATA!");
        EV << "Datos recibidos del sensor " << data->origenID << "\nPH: " << data->sensPH << "\nTEMP: " << data->sensTEMP << endl;
        EV << "ID DESTINO: " << data->destinoID << endl;

        if (countNodeTrans < 9) / Se almacenan los datos que componen una sola transaccion
        {
            string row = "" + data->origenID + ";" + std::to_string(data->sensPH) + ";" + std::to_string(data->sensTEMP) + ";";
            actualTransaction.push_back(row);

            //Datos mas legibles para Blockchain
            string row1 = "NodoOrigen: " + data->origenID + " SensorPH: " + std::to_string(data->sensPH) + " SensorTEMP: " + std::to_str:
            actualTransactionData.push_back(row1);
        }

        countNodeTrans++;
        if (countNodeTrans == 9) / Cantidad de nodos max, sin contar Genesis ni Attacker
        {
            bubble("Transaccion completa!");
            storeTransaction(); /Al completar los datos para la transaccion se llama al metodo
            countSensorTx++;
            countNodeTrans = 0;
        }
    }
}

```

3.4.13. Función CompeteBlock

Esta función es llamada cuando se recibe un mensaje de tipo *cMessage* y se utiliza para simular el proceso de minería de bloques en una cadena de bloques.

En la Figura 49 se logra observar el código encargado de realizar el minado dentro de la cadena de bloques, donde la función inicia verificando si el identificador (ID) del módulo es "[0]", "[10]" o "[20]".

Luego, se llama a la función *addBlockchain* para agregar un nuevo bloque a la cadena de bloques. La información para el nuevo bloque se obtiene de la variable *tempData*, que se inicializa como "datos de " seguido del ID del módulo.

Finalmente, se aumenta el índice del bloque y se incrementa la variable *transCounter*. También se programa la función para que se vuelva a llamar a sí misma en un tiempo futuro utilizando la función *scheduleAt*. El tiempo de espera se determina utilizando la función *exponential* con una tasa media especificada en la variable *rateMean*, más 10.

Figura 49

Código para el minado de la cadena de Bloques

```

void HonestModule::caseCompeteBlock(cMessage* msg)
{
    if (ID == "[0]" || ID == "[10]" || ID == "[20]") {
        if (transCounter < txLimite){
            EV << "*** NODO " << ID << " REALIZA MINADO PARA COMPETIR!" << endl;
            if (ID == "[20]" && simTime() > 35.0 && simTime() < 50) {
                blockIndex++;
            } else if (ID == "[0]" && simTime() > 35.0 && simTime() < 50) {
                blockIndex += 1;
            } else if (ID == "[10]" && simTime() > 50) {
                blockIndex++;
            } else if (ID == "[0]" && simTime() > 50) {
                blockIndex += 2;
            }
            EV << "*** INDEX " << blockIndex << endl;
            //auto data = (dataCompete*) msg->getContextPointer();
            //string tempData = readTransaction(blockIndex);
            string tempData = "datos de " + ID;
            //addBlockChain(blockIndex, data->temporalData);
            addBlockChain(blockIndex, tempData);
            blockIndex++;
            transCounter++;

            scheduleAt(simTime() + exponential(rateMean) + 10, msgCompeteBlock);
        }
    }
}

```

Verificamos si el valor de ID corresponde a las estaciones Base

Verificamos si el contador de transacciones es menor al limite de texto

Añadimos un bloque a la cadena a través de addBlockChain

3.4.1. Función caseDataNonce

Este método se llama al recibir un mensaje que contenga el parámetro nance (número de intentos necesarios para minar el bloque) y que se utilizará para determinar quién es el ganador de una competencia de minería.

La función comienza recuperando la información del mensaje y almacenándola en una variable *msgData* de tipo *dataNonce*. Luego, verifica si el ID del módulo es igual al *destinoID* especificado en el mensaje. Si esta condición se cumple, el código sigue su ejecución.

Luego, se almacena el *origenID* y el nonce del mensaje en un vector de dos dimensiones llamado *competeVec*. La variable *competeCounter* se utiliza para llevar un registro de cuántos nonces se han recibido hasta el momento.

Cuando *competeCounter* alcanza el valor de 3, significa que se han recibido todos los nonces necesarios para determinar al ganador de la competencia. El código utiliza un algoritmo de ordenamiento para ordenar el vector *competeVec* por el valor del nonce. Luego, se verifica si el ID del módulo es igual al valor de la primera fila y columna del vector ordenado *competeVec [0][0]*. Si esta condición se cumple, significa que el módulo es el ganador de la competencia y envía el bloque a los demás módulos utilizando la función *callSendBlock*.

En la Figura 50 se observa como la primera línea de código recupera la información del mensaje y la almacena en una variable llamada *msgData* de tipo *dataNonce*. Esta variable es un puntero a una estructura de datos que contiene información sobre el nonce y el ID del módulo que envió el mensaje.

La segunda línea de código verifica si el ID del módulo es igual al *destinoID* especificado en el mensaje. Si esta condición se cumple, significa que el módulo es el destinatario del mensaje y debe procesarlo. La tercera línea de código muestra un mensaje por pantalla con información sobre el nonce y el ID del módulo que envió el mensaje.

La cuarta y quinta líneas de código almacenan el *origenID* y el nonce del mensaje en un vector de dos dimensiones llamado *competeVec*. La variable *competeCounter* se utiliza para llevar un registro de cuántos nonces se han recibido hasta el momento.

La sexta línea de código incrementa el valor de *competeCounter* en 1. La siguiente sección de código se ejecuta cuando se han recibido todos los nonces necesarios para determinar al ganador de la competencia (cuando *competeCounter* alcanza el valor de 3).

El primer bloque de código implementa un algoritmo de ordenamiento para ordenar el vector *competeVec* por el valor del nonce. Luego, se verifica si el ID del módulo es igual al valor de la primera posición del vector. Si esta condición se cumple, significa que el módulo es el ganador de la competencia y debe enviar el bloque a los demás módulos. El código utiliza una serie de sentencias if anidadas para determinar a qué estación base se debe enviar el bloque.

Finalmente, se establece el valor de *competeCounter* en 0 para indicar que la competencia ha terminado. La última línea de código elimina la variable *msgData* y libera la memoria que ocupa.

Figura 50

Código para recuperación del mensaje

```

void HonestModule::caseDataNonce(cMessage* msg)
{
    auto msgData = (dataNonce*) msg->getContextPointer();
    if (ID == msgData->destinoID) {
        EV << "*** NONCE " << msgData->nonce << " recibido de " << msgData->origenID << endl;
        competeVec[0][competeCounter] = msgData->origenID;
        competeVec[1][competeCounter] = std::to_string(msgData->nonce);
        competeCounter++;

        if (competeCounter == 3) {
            for (int i = 0; i < 3; i++){ //Algoritmo de ordenamiento
                for (int j = 0; j < 3; j++){
                    if (std::stoi(competeVec[1][i]) >= std::stoi(competeVec[1][j])) {
                        //nada
                    } else {
                        string temp1 = competeVec[1][i];
                        string temp2 = competeVec[0][i];
                        competeVec[1][i] = competeVec[1][j];
                        competeVec[0][i] = competeVec[0][j];
                        competeVec[1][j] = temp1;
                        competeVec[0][j] = temp2;
                    }
                }
            }

            EV << "***COMPETEVEC\n" << competeVec[0][0] << " " << competeVec[0][1] << " " << competeVec[0][2] << "\n" << competeVec[1][0] << " " << competeVec[1][1] << " " << competeVec[1][2] << "\n";
            if (competeVec[0][0] == ID) {
                EV << "*** PRIMER NODO GANADOR " << ID << "\nEnviando con index: " << blockIndex - 1 << "\ncompeteVec[0][0]: " << competeVec[0][0] << "\ncompeteVec[1][0]: " << competeVec[1][0] << "\n";
                if (ID == "[0]") {
                    callSendBlock(blockIndex - 1, "[10]");
                    callSendBlock(blockIndex - 1, "[20]");
                } else if (ID == "[10]") {
                    callSendBlock(blockIndex - 1, "[0]");
                    callSendBlock(blockIndex - 1, "[20]");
                } else if (ID == "[20]") {
                    callSendBlock(blockIndex - 1, "[0]");
                    callSendBlock(blockIndex - 1, "[10]");
                }
            }
        }
    }
}

```

Recuperamos la información y la guardamos en la variable msgData

Verificamos si el ID del modulo es igual ID del Destino

Almacenamos el origen del ID y el nonce

Recibimos todos los nonces y determinamos el ganador de la competencia

Ejecutamos el algoritmo de ordenamiento

Se determina a que estaciones Base se debe enviar el Bloque

3.4.2. Función *caseBlockchain*

La función *caseBlockchain* se encarga de actualizar la cadena de bloques local y, si corresponde, reenviar el bloque recibido a los demás módulos en la lista *competeVec*.

Cuando se recibe un mensaje de tipo BLOCK, la función extrae los datos del mensaje y los guarda en la variable *msgData*. El mensaje contiene información sobre un bloque de la cadena de bloques, esto se lo puede evidenciar en la Figura 51 la cual muestra la estructura del *dataBlock* el cual es un puntero al mensaje que se ha recibido y se ha obtenido a través del método *getContextPointer ()*, este puntero está formado por diferente información como es el hash propio, hash del bloque anterior ,el ID del nodo origen , el ID del nodo destino entre otros.

Figura 51

Datos que se muestran dentro de la Cadena de Bloques

```
struct dataBlock
{
    int index;
    int nonce;
    string data;
    string hash;
    string prevHash;
    string time;
    string origenID;
    string destinoID;
};
```

Luego, se establece el valor de la variable *blockIndex* en el índice del bloque contenido en el mensaje y se inicializa la variable *competeOrder* en 0. Si el destino del mensaje es este módulo (identificado por ID), entonces:

- Muestra un mensaje en la pantalla indicando que se ha recibido un bloque.
- Muestra un mensaje en la pantalla con información sobre el bloque recibido.
- Llama a la función *updateBlock* con los datos del bloque recibido para actualizar la cadena de bloques local.

- Incrementa el índice del bloque en 1

La función *caseBlockChain* itera sobre el vector *competeVec* para encontrar el índice del módulo que envió el bloque en el mensaje. A continuación, se establece el valor de *competeOrder* en el índice encontrado. Si el valor de *competeOrder* no es igual a 2, se verifica si el módulo que sigue en la lista *competeVec* es este módulo. Si es así, se llama a la función *callSendBlock* para enviar el bloque a los demás módulos en la lista *competeVec*. La acción para realizar depende del valor de ID, teniendo en cuenta las siguientes acciones:

- Si ID es "[0]", entonces se llama a la función *callSendBlock* para enviar el bloque a los módulos "[10]" y "[20]".
- Si ID es "[10]", entonces se llama a la función *callSendBlock* para enviar el bloque a los módulos "[0]" y "[20]".
- Si ID es "[20]", entonces se llama a la función *callSendBlock* para enviar el bloque a los módulos "[0]" y "[10]".

Figura 52

Código para Actualización de la Cadena de Bloques

```

oid HonestModule::caseBlockChain(cMessage* msg)
{
    auto msgData = (dataBlock*) msg->getContextPointer();
    blockIndex = msgData->index;
    int competeOrder = 0;

    if (msgData->destinoID == ID)
    {
        bubble("BLOCK recibido!");
        //newBlock(msgData->data, msgData->index, -1, msgData->time);
        EV << "*** ACTUALIZANDO BLOCKCHAIN!" << "ENVIADO POR " << msgData->origenID << "\nMI INDEX " << blockIndex + 1 << endl;
        updateBlock(blockIndex, msgData->nonce, msgData->data, msgData->hash, msgData->time, msgData->prevHash);
        blockIndex++;

        for (int i = 0; i < 3; i++) {
            if (msgData->origenID == competeVec[0][i]) {
                competeOrder = i;
                break;
            }
        }
        //EV << "competeOrder: " << competeOrder << endl;

        if (competeOrder != 2) {
            //Enviar 3 por variable global que maneja en N° de islas
            //Nada
            if (competeVec[0][competeOrder + 1] == ID) {
                if (ID == "[0]") {
                    callSendBlock(blockIndex, "[10]");
                    callSendBlock(blockIndex, "[20]");
                } else if (ID == "[10]") {
                    callSendBlock(blockIndex, "[0]");
                    callSendBlock(blockIndex, "[20]");
                } else if (ID == "[20]") {
                    callSendBlock(blockIndex, "[0]");
                    callSendBlock(blockIndex, "[10]");
                }
            }
        }
    }
}

```

Guardamos los datos obtenidos del metodo getContextPointer en el msgData

Establecemos el valor de la variable blockIndex

Verificamos el ID de destino

Mostramos un mensaje en la pantalla y consola

Actualiza la cadena de Bloques local

Se maneja segun el numero de islas

Enviamos los bloques a las Estaciones Base

4. CAPITULO IV: PRUEBAS DE FUNCIONAMIENTO Y RESULTADOS

En esta sección se detallan las pruebas de funcionamiento y los resultados obtenidos, en relación con el esquema de autenticación propuesto. Se describe con precisión los métodos y procedimientos utilizados para evaluar el rendimiento y características del esquema, así como validar los objetivos del trabajo planteado. Además, se presentan y analizan los resultados obtenidos mediante gráficos, tablas y códigos necesarios para comprender de manera completa el trabajo realizado.

4.1.Fase 4: Validación y Verificación de los Resultados

Una vez completado el diseño e implementación del esquema de autenticación, se procede a describir en detalle las pruebas y procedimientos utilizados para validar y verificar los resultados obtenidos. Esto incluye, el uso del patrón de las tres A (Arrange, Act, Assert) las cuales proponen la división de una prueba unitaria en las tres secciones indicadas en la Tabla:

Tabla 11

Explicación de Patrón de tres A

| Nombre | Significado | Descripción |
|----------------|----------------------|--|
| Arrange | Organizar/Inicializa | Es la configuración inicial que se debe realizar antes de ejecutar la prueba, en este apartado se establecen los valores de los datos y se preparan los objetos necesarios para el funcionamiento del código |
| Act | Actuar | En esta sección se realiza la prueba para lo cual se invoca los métodos o funciones utilizados dentro de la simulación. |
| Assert | Confirmar/Comprobar | En esta parte se verifica que el resultado es el esperado esto se lo realiza a través de la comprobación de los métodos ejecutados. |

Estas prácticas se utilizan para diseñar y ejecutar pruebas unitarias efectivas y confiables. El objetivo es asegurar que las pruebas sean coherentes, fáciles de entender y mantenibles, y que proporcionen información valiosa sobre el comportamiento del código. Al seguir estas prácticas, se puede identificar rápidamente y corregir errores en el código, lo que ayuda a mejorar la calidad del software y reducir los errores.

a) Realización de test a través del Shell de Omnet ++

Para realizar las pruebas unitarias, se hace uso de la herramienta *opp_test* de OMNET++. Esta herramienta es versátil y útil para diferentes situaciones de prueba, permitiendo demostrar funciones, clases, módulos y simulaciones completas. Como resultado de la ejecución de esta herramienta, se muestra en la Figura 53 la obtención de un archivo *.test* que muestra el resultado de la prueba del código NED y de los archivos *.ini*.

El funcionamiento de estas pruebas se basa en tres componentes: el archivo de prueba *".test"*; un script de control que depende de *opp_test* para la ejecución de las pruebas; y un programa de prueba que genera una carpeta *work*, que es la salida del programa de simulación.

Figura 53

Ejecución por consola de pruebas unitarias

```

/c/omnetpp/omnetpp-6.0.1/test/core$ ./runtest iota_batch.test
make: se entra en el directorio '/c/omnetpp/omnetpp-6.0.1/test/core/signalreg'
make: No se hace nada para 'all'.
make: se sale del directorio '/c/omnetpp/omnetpp-6.0.1/test/core/signalreg'
opp_test.py: extracting files from *.test files into ./work...
make: No se hace nada para 'all'.
opp_test: running tests using work_dbg...
*** iota_batch.test: PASS
=====
PASS: 1  FAIL: 0  ERROR: 0  EXPECTEDFAIL: 0  SKIPPED: 0
Aggregate result: PASS
Results can be found in ./work
  
```

b) Test aplicado a la Cadena de Bloques

El código mostrado en la Figura 54 permite visualizar diferentes bibliotecas como *OMNeT++*, *SHA-256* y *blockChain* que brindan funciones y clases para cifrado y creación de una cadena de bloques. Al utilizar el comando *using namespace omnetpp*,

se permite el acceso a las funciones y clases de la biblioteca OMNeT++. La clase *Test* se define a través de *class Test: public cSimpleModule*, lo que permite su accesibilidad en una simulación y garantiza la correcta ejecución de la actividad del módulo "Test" durante la simulación.

Las líneas *Blockchain bChain = Blockchain()*, *"EV << "Minando bloque 1... << \n y bChain.AddBlock(Block(1, "Datos Bloque 1"))* crean un objeto de la clase Blockchain, lo minan y agrega un bloque a la cadena de bloques.

Figura 54

Test creado para verificación de código de la Cadena de Bloques

```

#include <omnetpp.h>
#include <iotalib/sha256.h>
#include <iotalib/blockChain.h>
#include <iostream>

using namespace omnetpp;

namespace blockChain {

class Test : public cSimpleModule
{
public:
    Test() : cSimpleModule(65536) {}
    virtual void activity();
};

Define_Module(Test);

void Test::activity()
{
#define PRINT(x, i)  EV << "Bloque [" << i << "]: " << x << "\n";

Blockchain bChain = Blockchain();

EV << "Minando bloque 1..." << "\n";
bChain.AddBlock(Block(1, "Datos Bloque 1"));

EV << "Minando bloque 2..." << "\n";
bChain.AddBlock(Block(2, "Datos Bloque 2"));

EV << "Minando bloque 3..." << "\n";
bChain.AddBlock(Block(3, "Datos Bloque 3"));

for (int i = 0; i < 4; i++)
    PRINT(bChain.stringBlockChain[i], i);
}
}

```

Librerías Incluidas

Inicializamos el módulo

Minado de los Bloques

c) Test realizado a la red Tangle

El código indicado en la Figura 55 incluye bibliotecas externas, tales como *iotalib/json.hpp*, *iotalib/blockChain.h*, *iotalib/sha256.hy* OpenMP útiles para el correcto desempeño del código, así como también bibliotecas para la generación de contraseñas aleatorias mediante el diccionario de contraseñas. Además, para su correcto funcionamiento, depende de varios módulos, entre los cuales se incluyen:

- ***connectModules***: este método se encarga de conectar dos módulos en la red.
- ***positionModules***: este método establece la posición de los módulos en la red.
- ***getEdgeList***: este método devuelve la lista de aristas (conexiones entre nodos) de la topología utilizada a partir de un archivo CSV en una carpeta de topologías.
- ***getModules***: este método devuelve un vector que contiene punteros a todos los módulos (honestos y maliciosos) en la red.
- ***checkError***: este método arroja un error si existe un conflicto entre el archivo CSV y los parámetros del archivo NED.
- ***initialize***: este método es una función virtual que se sobrescribe para inicializar el módulo ***ConfiguratorModule***.

La clase ***ConfiguratorModule*** utilizada en el código es una clase derivada de ***cSimpleModule***, que es esencial para la simulación de la red. Esta clase implementa métodos para leer datos de un archivo CSV y utilizarlos para configurar una simulación de red.

La clase proporciona métodos para dividir una línea de un archivo CSV en una lista de números enteros, conectar dos módulos, colocar módulos y devolver la lista de

bordes de la topología utilizada. Además, también permite la generación de un vector que contiene punteros⁵ a todos los módulos, tanto los honestos como los maliciosos.

Figura 55

Test realizado a la red Tangle

```

#include <cmath>
#include <math.h>

#include <algorithm>
#include <numeric>
#include <iotalib/comp.h>
#include <iotalib/Structures.h>
#include <iotalib/sha256.h>

//Semilla de contraseña
#include <cstdlib>
#include <ctime>
#include <string.h>

using namespace cmmnetpp;
using std::string;

namespace iota_batch {
#pragma GCC diagnostic push
#pragma GCC diagnostic ignored "-Wunused-parameter"

    //##### INICIO - ConfiguratorModule #####

    class ConfiguratorModule : public cSimpleModule
    {
    public:
        //Dividir una línea (del fichero CSV) en una lista de int según el delimitador ','
        std::vector<int> split(const std::string& line) const;
        //Conectar dos módulos
        void connectModules(cModule* moduleOut, cModule* moduleIn);
        //Posición de los módulos
        void positionModules(cModule* moduleOut, int edgeFirst);

        //Devuelve la lista de aristas (utilizando un mapa :
        // [Nodo (clave)] -> Nodos adyacentes (valor)) de la topología utilizada a partir del archivo csv de la carpeta de topologías.
        std::map<int, std::vector<int>> getEdgeList() const;
        //Devuelve un vector que contiene punteros para todos los módulos (honestos y maliciosos)
        std::vector<cModule*> getModules() const;

        //Arrojar un error si existe un conflicto entre el archivo CSV y los parámetros del archivo NED (por ejemplo, el número de módulos)
        void checkError(int nodeID, int totalModules) const;

        //Sobreescribir la función initialize() de la clase cSimpleModule
        void initialize() override;

        std::vector<std::string> positionsVec;
    };
}

```

El código indicado en la Figura 56 define el módulo *ConfiguratorModule* y dos funciones que forman parte de él. La función *split()* se encarga de separar una cadena de números enteros en tokens usando comas como separador. La función *getEdgeList()* devuelve un mapa que representa una topología, donde la clave es un número entero y el valor es un vector de números enteros. Esta función lee la topología actual a partir del parámetro *topology* del módulo padre, abre un archivo CSV correspondiente a esa

⁵ **Punteros:** Tipo de dato en programación que apunta a una dirección de memoria en la que se encuentra almacenado un valor

topología y devuelve un mapa que representa los nodos y sus respectivos vecinos en la topología.

Figura 56

Segunda parte del test realizada a la red Tangle

```

Define_Module(ConfiguratorModule);

std::vector<int> ConfiguratorModule::split(const std::string& line) const
{
    std::vector<int> tokens;
    std::string token;
    std::istringstream tokenStream(line);

    while(std::getline(tokenStream, token, ','))
    {
        tokens.push_back(stoi(token));
    }

    return tokens;
}

std::map<int, std::vector<int>> ConfiguratorModule::getEdgeList() const
{
    std::string currentTopo = getParentModule()->par("topology");
    std::cout << "Topología actual: " << currentTopo << "\n";
    EV << "***** TOPO: " << currentTopo << " parent: " << getParentModule() << "\n";
    std::string path = "./topologies/CSV/" + currentTopo + ".csv";
    std::fstream file;
    file.open(path, std::ios::in);

    if(!file.is_open()) throw std::runtime_error("Could not open CSV file");

    std::string line;
    std::map<int, std::vector<int>> myTopo;

    while(getline(file, line))
    {
        auto edgeList = split(line);
        myTopo[edgeList[0]] = std::vector<int>(edgeList.begin() + 1, edgeList.begin() + edgeList.size());
    }

    file.close();
}

```

Annotations in the image:

- A box labeled "Función split separa cadenas en tokens" points to the `split` function.
- A box labeled "Crea el mapa de la topología con sus respectivos nodos" points to the `getEdgeList` function.

El código que se muestra en la Figura 57 se encarga de conectar dos módulos⁶, llamados `moduleOut` y `moduleIn`, a través de un canal de retardo. El retardo puede ser fijo como un valor (`delay`) o generado aleatoriamente (a través de la función `uniform(minDelay, maxDelay)`) dependiendo del valor del parámetro "ifRandDelay" en el módulo principal.

El tamaño de las puertas de salida de `moduleOut` y la puerta de entrada de `moduleIn` se incrementan en 1 y luego se conectan a través del canal creado (`channel`).

⁶ **Módulos:** Unidad de software dentro de Omnet++ que se utilizan para simular el comportamiento de diferentes entidades en un sistema

Además, se configura la cadena de visualización de la conexión para que no sea visible (ancho=0) mediante el uso de `gOut->getDisplayString().setTagArg("ls",1,"0")`.

Figura 57

Creación de conexión y retardo de los módulos

```
void ConfiguratorModule::connectModules(cModule* moduleOut, cModule* moduleIn)
{
    double delay = 0.0;

    if(getParentModule()->par("ifRandDelay"))
    {
        double minDelay = getParentModule()->par("minDelay");
        double maxDelay = getParentModule()->par("maxDelay");
        delay = uniform(minDelay,maxDelay);
    }

    else
    {
        delay = getParentModule()->par("delay");
    }

    auto channel = cDelayChannel::create("Channel");
    channel->setDelay(delay);

    moduleOut->setGateSize("out",moduleOut->gateSize("out") + 1);
    moduleIn->setGateSize("in",moduleIn->gateSize("in") + 1);

    auto gOut = moduleOut->gate("out",moduleOut->gateSize("out") - 1);
    auto gIn = moduleIn->gate("in",moduleIn->gateSize("in") - 1);

    EV << "confModule-getDisplayString: " << moduleOut->getDisplayString().str() << endl;

    gOut->connectTo(gIn,channel);
    gOut->getChannel()->callInitialize();

    /*
    cDisplayString & disp = gOut->getDisplayString();
    disp.setTagArg("ls", 1, "0"); // 1 indicates argument of the Tag, 0 - width
    */
    //conexiones entre gates no son visibles (width=0)
    gOut->getDisplayString().setTagArg("ls",1,"0");
    std::cout << "ConfiguratorModule::connectModules -> Pass" << "\n";
}

```

Este código implementa la función `ConfiguratorModule::positionModule` que tiene como objetivo posicionar un módulo llamado `moduleOut` en una posición específica dentro de una topología determinada. La topología se selecciona en base a la propiedad `topology` en el módulo principal, que puede tener uno de los siguientes valores: *SinAtaques*, *ConAtaques*, *IslaAumentada* o *NodosAumentados*.

La sentencia switch indicada en la Figura 58 que compara `currentTopoInt` con diferentes valores y determina la posición (posX, posY) de `moduleOut`. Además, la función establece la etiqueta y el ícono del módulo a través de la función `setTagArg`. El

parámetro *edgeFirst* se utiliza como un índice para seleccionar la posición correcta en función de la topología seleccionada.

Figura 58

Posiciones de los sensores al establecer la red

```

//Posiciones
int posX, posY, currentTopoInt;

std::string currentTopo = getParentModule()->par("topology");
if (currentTopo == "SinAtaques") currentTopoInt = 1;
else if (currentTopo == "ConAtaques") currentTopoInt = 2;
else if (currentTopo == "IslaAumentada") currentTopoInt = 3;
else if (currentTopo == "NodosAumentados") currentTopoInt = 4;

switch(currentTopoInt)
{
case 1:
switch(edgeFirst)
{
case 0:
posY = 300;
posX = 1000;
moduleOut->getDisplayString().setTagArg("t", 0, "[Isla 1]");
moduleOut->getDisplayString().setTagArg("i", 0, "device/antennatower");
break;
case 1:
posY = 500;
posX = 1100;
break;
case 2:
posY = 100;
posX = 1100;
break;
case 3:
posY = 100;
posX = 1300;
break;
case 4:
posY = 500;
posX = 1300;
break;
case 5:
posY = 100;
posX = 1500;
break;
case 6:
posY = 300;
posX = 1500;
break;
case 7:

```

Compara y determina las posiciones de cada uno de los nodos sensores

d) Resultado que Genera la Cadena de Bloques

Este resultado se da al cargar diferentes archivos. **NED**, provenientes del directorio actual y del directorio *lib*. La simulación comienza con la inicialización del módulo *Test*, que se encarga de minar tres bloques y agregarlos a una cadena de bloques.

La información relevante de cada bloque minado se imprime de la forma indicada en la Figura 59, incluyendo su índice, nonce, contenido, hash y el hash del

bloque anterior. Después de completar la minería de los tres bloques, la simulación se completa y se llama a la función *finish()* para finalizar la ejecución.

Figura 59

Resultado Obtenido al realizar el Test a la Cadena de Bloques

```

OMNeT++ Discrete Event Simulation (C) 1992-2022 Andras Varga, OpenSim Ltd.
Version: 6.0, build: 220413-71d8fab425, edition: Academic Public License -- NOT FOR COMMERCIAL USE
See the license for distribution terms and warranty disclaimer

Setting up Cmdenv...

Loading NED files from .: 2
Loading NED files from ..\lib: 8

Preparing for running configuration General, run #0...
Assigned runID=General-0-20230117-12:09:48-8144
Setting up network "Test"...
Initializing...
Initializing module Test, stage 0
Running simulation...
** Event #1 t=0 Test (Test, id=1)
Minando bloque 1...
*** Bloque minado con exito! ***
Minando bloque 2...
*** Bloque minado con exito! ***
Minando bloque 3...
*** Bloque minado con exito! ***
Bloque [0]: {"index": 0,"nonce": -1, "data": "Genesis Block", "hash": 0, "time": 0, "prevHash": 0}
Bloque [1]: {"index": 1,"nonce": 200, "data": "Datos Bloque 1", "hash": 00735b4ad589ad252fbf123150dd4c74e703f48d3089bfff32ecb70f30592f06, "time": , "prevHash":
Bloque [2]: {"index": 2,"nonce": 177, "data": "Datos Bloque 2", "hash": 007b930475087f9a2a73013355c3d7123035f6862e85b50dacfae0cb091df87b, "time": , "prevHash":
Bloque [3]: {"index": 3,"nonce": 113, "data": "Datos Bloque 3", "hash": 00f971816fd3806ed3630be715a214edf37f0abdb418b37be44544e7f5c4daaa, "time": , "prevHash":
No more events, simulation completed at t=0, event #1

Calling finish() at end of Run #0...
End.

```

Realizamos la inicialización del módulo Test

Minado del Bloque

Resultado del Bloque Minado

e) Resultado de la red de sensores

En este caso, cada uno de los nodos se evalúa a través de un criterio "PASS" el cual es un indicador de que un módulo o una parte de la simulación se ha completado con éxito y ha cumplido con los requisitos esperados. En la Figura 60 se ve que el módulo *ConfiguratorModule* está realizando varias operaciones y cada vez que se completa una operación, se registra una línea con *ConfiguratorModule:[nombre de la operación] -> Pass*. Las operaciones incluyen:

- *getModules_Honest*: obtener módulos honestos
- *checkError*: verificar errores

4.1.1. Validación de resultados de funcionamiento

Para garantizar la seguridad del sistema, se llevarán a cabo pruebas de validación en diferentes escenarios, tanto en condiciones normales como en situaciones de ataques simulados. Para ello, se usó la tecnología LoRAWAN misma que posee características de bajo consumo de energía, larga distancia de transmisión, escalabilidad y compatibilidad necesarias para la implementación del esquema propuesto. Así se podrá evaluar la capacidad de la red para resistir futuros ataques y comprender el nivel actual de seguridad del esquema de autenticación, para un desarrollo más ordenado de las pruebas de funcionamiento, se ha creado la Tabla 12 misma que muestra los tipos de pruebas que se realizarán con su respectivo Ambiente y resultados que se espera obtener.

Tabla 12

12

Listado de Pruebas realizadas

| LISTADO DE PRUEBAS | | |
|--|--|--|
| Tipos de Pruebas | Características del Ambiente | Resultados esperados de las Pruebas |
| Prueba 1: Verificar el funcionamiento del esquema de autenticación | Se verifica el envío de datos desde los nodos sensores a la estación base, sin utilizar una tecnología específica. | Se mostrarán mensajes detallando cada paso del proceso de autenticación y hasta el envío de datos a través del esquema de autenticación |
| Prueba 2: Evaluación del rendimiento del esquema de autenticación en el contexto del Modelo Urbano de LoRAWAN | Se evaluará el rendimiento de la potencia LoRAWAN tanto del nodo sensor emisor como del receptor, se determinarán los coeficientes de pérdida y se medirá la distancia. | Se evaluará la eficiencia del esquema al verificar si no consume en exceso energía y tiene bajos niveles de pérdida al enviar los datos. |
| Prueba 3: Simulación de ataques (Ataque de diccionario y denegación de servicio (DoS)) para evaluar la resistencia del esquema de autenticación a estos tipos de amenazas. | Se utilizarán dos nodos atacantes para evaluar si la red es resistente a los ataques de diccionario y denegación de servicio (DoS), para verificar la vulnerabilidad de la red ante estos tipos de amenazas. | Se evaluará si el protocolo SEECR detecta el nodo malicioso e indica a la red el ID del nodo malicioso, y si la red evita enviar mensajes a través de este camino, para garantizar la seguridad de la red. |
| Prueba 4: Creación de diferentes ambientes (Aumento del número de nodos e islas), para evaluar la escalabilidad de la red. | El ambiente tendrá una mayor cantidad de islas y nodos que el esquema original, para evaluar la capacidad de la red para adaptarse a una mayor complejidad | Se evaluará si existe retraso en el envío de mensajes o si hay un aumento en el tiempo requerido para completar la PoW dentro de cada isla, para medir el rendimiento de la red en ambientes con una mayor cantidad de nodos e islas |
| Prueba 5: Evaluación del comportamiento del esquema | Se realizará la conexión de una simulación con una cadena de Bloques Real, para lo cual dentro de | Observar el comportamiento referente a tiempos de creación y |

propuesto al unirse a una Cadena de Bloques Real. tres computadores se correrá la simulación de la red de sensores. cargado de bloques a la página web.

4.1.1.1.Prueba 1: Verificar el funcionamiento del Esquema de autenticación

Durante la simulación, se utilizan diversos mensajes para visualizar el proceso de autenticación que se lleva a cabo entre la estación base y los nodos, así como entre los mismos nodos. A fin de comprobar el correcto funcionamiento de la simulación, se presenta de manera detallada los resultados obtenidos tanto en consola como en la simulación, indicando cada uno de los pasos que conforman el esquema de autenticación, hasta la fase final de creación de la cadena de bloques.

Inicialización de Nodos y Obtención de Contraseña Dinámica

Al iniciar la simulación, los nodos y las estaciones base que conforman la red pasan por un proceso de inicialización. Como se muestra en la Figura 61 en este proceso se establecen las conexiones entre los nodos y se establecen las puertas de salida, que se pueden reconocer en la imagen con los nombres "output". Estas puertas se usan para recibir la información.

Figura 61

Inicialización de las puertas de salida

```

Initializing channel Network.Honest[0].out[0].Channel, stage 0
INFO (ConfiguratorModule)Network.Configurator:Module 1 ---> Module 2
INFO (ConfiguratorModule)Network.Configurator:confModule-getDisplayString: p=400,300;t=Ger
Initializing channel Network.Honest[0].out[1].Channel, stage 0
INFO (ConfiguratorModule)Network.Configurator:Module 1 ---> Module 3
INFO (ConfiguratorModule)Network.Configurator:confModule-getDisplayString: p=400,300;t=Ger
Initializing channel Network.Honest[0].out[2].Channel, stage 0
INFO (ConfiguratorModule)Network.Configurator:Module 1 ---> Module 11
INFO (ConfiguratorModule)Network.Configurator:confModule-getDisplayString: p=400,300;t=Ger
Initializing channel Network.Honest[0].out[3].Channel, stage 0
INFO (ConfiguratorModule)Network.Configurator:Module 1 ---> Module 12
INFO (ConfiguratorModule)Network.Configurator:confModule-getDisplayString: p=500,500;i=de
Initializing channel Network.Honest[1].out[0].Channel, stage 0
INFO (ConfiguratorModule)Network.Configurator:Module 2 ---> Module 1
INFO (ConfiguratorModule)Network.Configurator:confModule-getDisplayString: p=500,500;i=de
Initializing channel Network.Honest[1].out[1].Channel, stage 0
INFO (ConfiguratorModule)Network.Configurator:Module 2 ---> Module 3
INFO (ConfiguratorModule)Network.Configurator:confModule-getDisplayString: p=500,500;i=de
Initializing channel Network.Honest[1].out[2].Channel, stage 0
INFO (ConfiguratorModule)Network.Configurator:Module 2 ---> Module 4
INFO (ConfiguratorModule)Network.Configurator:confModule-getDisplayString: p=500,500;i=de
Initializing channel Network.Honest[1].out[3].Channel, stage 0
INFO (ConfiguratorModule)Network.Configurator:Module 2 ---> Module 5
INFO (ConfiguratorModule)Network.Configurator:confModule-getDisplayString: p=500,500;i=de
Initializing channel Network.Honest[1].out[4].Channel, stage 0
INFO (ConfiguratorModule)Network.Configurator:Module 2 ---> Module 6
INFO (ConfiguratorModule)Network.Configurator:confModule-getDisplayString: p=500,100;i=de
Initializing channel Network.Honest[2].out[0].Channel, stage 0
INFO (ConfiguratorModule)Network.Configurator:Module 3 ---> Module 1
INFO (ConfiguratorModule)Network.Configurator:confModule-getDisplayString: p=500,100;i=de
Initializing channel Network.Honest[2].out[1].Channel, stage 0
INFO (ConfiguratorModule)Network.Configurator:Module 3 ---> Module 2
  
```

A continuación, en la Figura 62 se muestra cómo cada uno de los nodos realiza el proceso de obtención de una contraseña dinámica. Esta contraseña se genera a partir de la unión de una contraseña y una salt, las cuales son buscadas dentro del diccionario de contraseñas y salt que contiene cada una de las estaciones base. Una vez creada la contraseña, se realiza un proceso de hashing de contraseñas usando la función SHA-256 para evitar el envío de texto plano. También se muestra el tiempo en el que se realizará la siguiente actualización de contraseña mismo que se obtiene a través de la formula $simTime() + 5 - peso * 0.1 \ll '' (s)$. La función $simTime()$ proporciona el tiempo real de la simulación, al cual se le suma el valor resultante de la operación de 5 segundos (tiempo de retraso), menos el peso propio que tiene el nodo multiplicado por el valor de 0.1. De esta manera, a medida que aumenta el peso, la actualización de contraseñas se realiza en menos tiempo. Esta práctica se lleva a cabo porque si los nodos mantienen una contraseña estática, será más fácil de descifrar.

Figura 62

Obtención de Contraseña dinámica en los nodos y estación base

```

** Event #1 t=0.033148700741 Network.Honest[20] (HonestModule, i
INFO:MessageType: 13
INFO:Obteniendo la pass dinamica...
INFO:Pass: a12345
INFO:Salt: hobllly11
INFO:Pass + salt: a12345hobllly11
INFO:Con Hash igual a: 42d63a847560df45e99d62501e9728541f5c8d1504
INFO:Siguiente actualizacion en t= 4.933148700741 (s).
** Event #2 t=0.07066535678 Network.Honest[19] (HonestModule, i
INFO:MessageType: 13
INFO:Obteniendo la pass dinamica...
INFO:Pass: azerty
INFO:Salt: maktub
INFO:Pass + salt: azertymaktub
INFO:Con Hash igual a: 4bf86936c820577605ad096b9fcbaa6556bf5928af
INFO:Siguiente actualizacion en t= 4.97066535678 (s).
** Event #3 t=0.109440298254 Network.Honest[1] (HonestModule, i
INFO:MessageType: 13
INFO:Obteniendo la pass dinamica...
INFO:Pass: 456789
INFO:Salt: reper
INFO:Pass + salt: 456789reper
INFO:Con Hash igual a: eaf6815c8df1093c304308d367b8c68a45f0e062ca
INFO:Siguiente actualizacion en t= 5.009440298254 (s).
** Event #4 t=0.126209996446 Network.Honest[23] (HonestModule,
INFO:MessageType: 13
INFO:Obteniendo la pass dinamica...
INFO:Pass: 686584
INFO:Salt: blbysek
INFO:Pass + salt: 686584blbysek
INFO:Con Hash igual a: 86e5e63e4783a83c471225dd0cff5f9379431fa37C
INFO:Siguiente actualizacion en t= 5.026209996446 (s).
** Event #5 t=0.143010486254 Network.Honest[21] (HonestModule,

```

a. Petición de Asociación

Al ejecutar el código indicado en la Figura 36 de la **sección 3.4.3** se puede observar cómo en la consola se muestra un mensaje de petición de asociación, como se indica en la Figura 63 hacia la estación base. En este mensaje se muestra el ID del nodo que está realizando la petición de asociación para que la estación base pueda identificar a quién deberá enviar el mensaje de respuesta de asociación.

Figura 63

Mensaje de petición de Asociación a través de consola

```

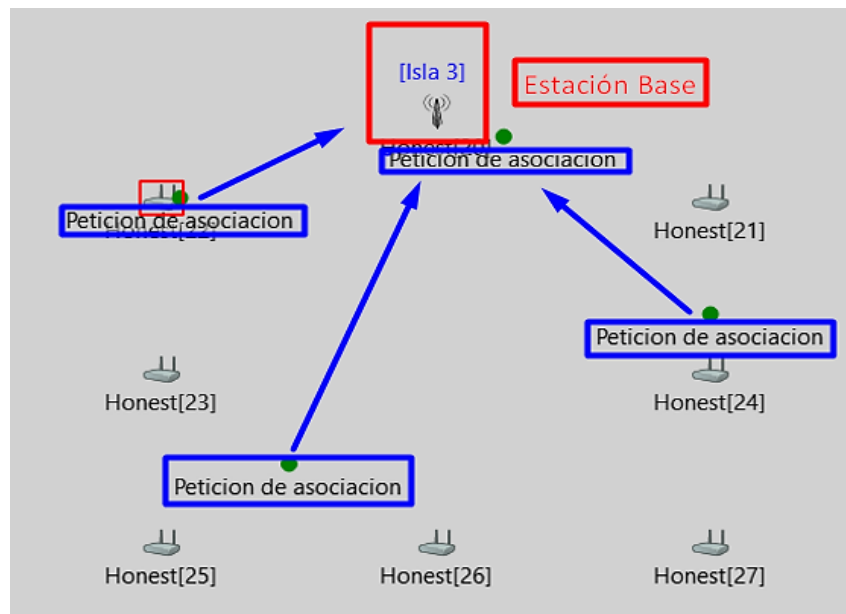
** Event #49 t=1.305512204647 Network.Hones
INFO:MessageType: 8
INFO:Nodo [21] envia peticion de asociacion!
INFO:***** timeIni: 1.305512204647

```

El envío del mensaje de petición de asociación también se puede observar dentro de la simulación, como se observa en la Figura 64. Los nodos con ID [24] y [25] hacen el envío del mensaje *Petición de Asociación* identificados con un círculo de color verde hacia la estación base.

Figura 64

Envío de petición de Asociación de los nodos a la Estación Base



b. Respuesta de Asociación

Al recibir la estación base el mensaje de petición de asociación en la consola se mostrará el mensaje indicado en la Figura 65, que indica que se ha recibido un mensaje de petición de asociación del nodo con ID [21], A continuación, también se muestra el hash usado para la prueba de trabajo el cual es enviado desde la estación base con ID [20] hacia el nodo ID [21]. Este hash está formado por la contraseña y la Salt que generó la estación base al momento de inicializarse.

Figura 65

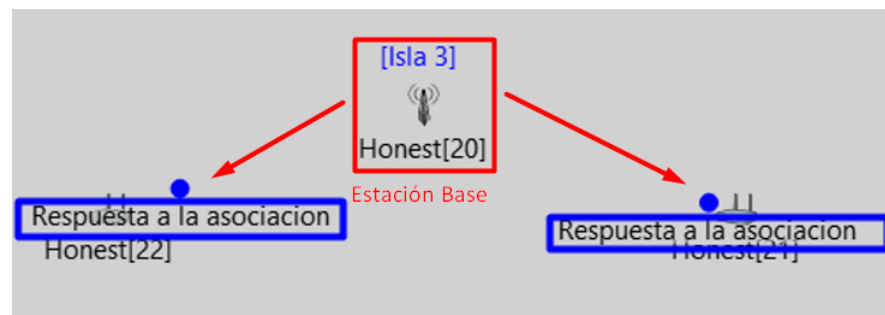
Contenido del mensaje de Respuesta de Asociación

```
INFO:Potencia LORA recibida: -101.41 dBm a 721.11 metros!
** Event #53 t=1.575275324189 Network.Honest[20] (HonestModule,
INFO:MessageType: 9
INFO:RECIBIDO MSG_ASO con ID: [21]
INFO:Enviando hash para PoW: 48d3488b29e2dda2b78c2350a41fdb349501
INFO:hacia el nodo [21]; desde el nodo [20]
INFO:pathLoss: 74.6698dB coefLoss: 104.67dB rxPowerdBm: -90.6698dB
INFO:Potencia LORA recibida: -90.6698 dBm a 316.228 metros!
```

En la Figura 66 se observa el envío de mensajes de respuesta de asociación desde la estación base ID [20] a los nodos sensores con ID [21] y [22]. Estos mensajes se representan con un círculo de color azul y, como se envían a través de un medio inalámbrico, son escuchados por ambos nodos. Sin embargo, al verificar el ID del nodo al que está dirigido el mensaje, solo el nodo sensor al que pertenece lo acepta, mientras que el otro lo rechaza.

Figura 66

Envío de Respuesta de Asociación de la Estación base a los nodos



c. Realización de Prueba de Trabajo

Cuando el nodo recibe el mensaje de respuesta de asociación, ejecuta el código indicado en la Figura 41 y 42 de la **sección 3.4.7** y **sección 3.4.8** respectivamente para luego mostrar en la consola de simulación el mensaje observado en la Figura 67, que

indica que el nodo con ID [21] ha recibido el mensaje de la estación base representada con el ID [20].

La prueba de trabajo se lleva a cabo internamente en el nodo sensor con ID [21], mismo que a través de los diccionarios de contraseñas y salt que tiene, crea diferentes combinaciones y las hashea, cuando obtiene el hash lo compara con el recibido de la estación base o nodo dependiendo del caso y comprueba si su resultado es el mismo, si es así lo valida y el nodo sensor procederá a eliminar la contraseña y solo enviará el hash correspondiente de la salt a la estación base, que se encargará de realizar la validación, si es lo contrario indica que no es válido. Como el nodo debe realizar diferentes intentos la información mostrada por consola corresponde al número de intentos que realizó el nodo hasta encontrar el hash, el cual para este caso es 44.

Figura 67

Inicia el proceso de Autenticación

```
INFO:MessageType: 10
INFO:El nodo [21] ha recibido el hash enviado por el nodo [20]
INFO:Proceso de autenticación...
INFO:Hash encontrado en el intento: 44
INFO:Enviando salt hacia la STA para validacion
INFO:Pass: 1111
INFO:Salt: Vandal1982
```

d. Autenticación

Cuando el nodo encuentra el hash correcto, envía un mensaje de autenticación a la estación base, que incluye la salt para su validación. La estación base emitirá un mensaje por consola como se indica en la Figura 68, que muestra el ID del nodo del que ha recibido el mensaje de autenticación, además del hash que envió y el hash que recibió. Una vez validado, el nodo se asocia a la estación base, lo que aumenta su peso de 1 a 2. La información de que el nodo se ha asociado se envía a todos los nodos de la

red, lo que reduce el tiempo de actualización de las contraseñas que contienen cada nodo.

Figura 68

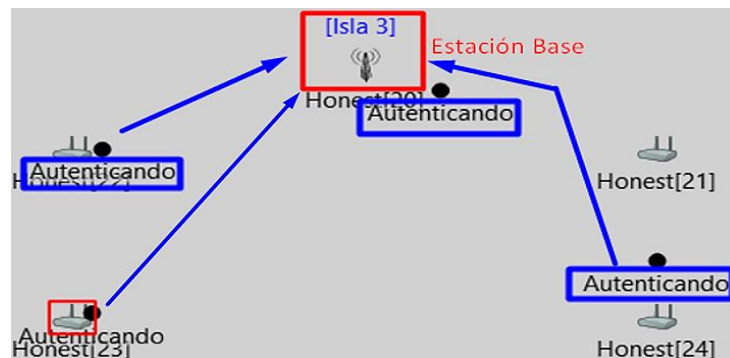
Mensaje de Autenticación y comparación de Hash

```
INFO:MessageType: 7
INFO:RECIBIDO MSG_AUTH con ID: [21]
INFO:HASH: 138e7461d245c2fd0feef68d14663fcd50680c89c8fdbabcf01228.
INFO:actualHash: 138e7461d245c2fd0feef68d14663fcd50680c89c8fdbabc.
INFO:La STA ha recibido el hash y lo valida!
INFO:Peso del nodo [20] es de: 2
INFO:Informando la asociacion de [21] por el nodo [20]
INFO:Enviando a todos los nodos!
```

En la Figura 69 se muestra cómo se realiza el envío de los mensajes de autenticación desde los nodos hacia la estación base, estos son identificados a través de un círculo de color negro y de la burbuja que muestra el nombre del mensaje

Figura 69

Envío de mensaje de Autenticación de los nodos a la Estación base



e. ACK de Asociación

Si la autenticación se realiza correctamente, la estación base genera un mensaje de ACK que confirma al nodo que ya se encuentra asociado y que ya puede enviar información. Este mensaje se muestra en la consola como se indica en la Figura 70, con el ID del nodo al que va dirigido y la indicación "Nodo asociado correctamente".

Figura 70

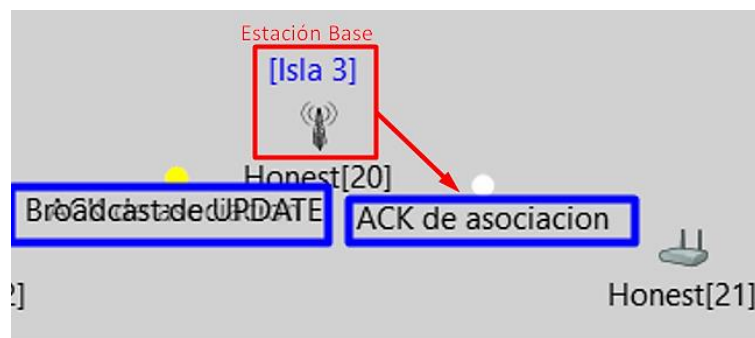
Mensaje de ACK enviado por consola

```
INFO:MessageType: 11
INFO:RECIBIDO MSG_ASO_ACK para ID: [21]
INFO:Nodo asociado correctamente!
```

En la Figura 71 se puede observar cómo, durante la simulación, se envían mensajes desde la estación base hacia los nodos sensores utilizando círculos blancos con el mensaje "ACK de Asociación". Estos mensajes se utilizan para confirmar que la asociación del nodo sensor a la red ha sido exitosa.

Figura 71

Envío de ACK de asociación



f. Mensaje de Actualización

Cuando se lleva a cabo la autenticación y validación de un nodo que desea unirse a la red, se muestra un mensaje en la consola de la simulación (como se ve en la Figura 72) que informa a todos los nodos de la red sobre el ID del nuevo nodo y el ID del nodo que lo asoció. Además, se solicita a todos los nodos que permitieron esta asociación de manera indirecta (asociando previamente a alguno de los nodos involucrados) que aumenten su peso para fortalecer la red, como se indica en el mensaje de la consola. Además, se solicita a todos los nodos que permitieron esta asociación de manera

indirecta (asociando previamente a alguno de los nodos involucrados) que aumenten su peso para fortalecer la red, como se indica en el mensaje de la consola.

Figura 72

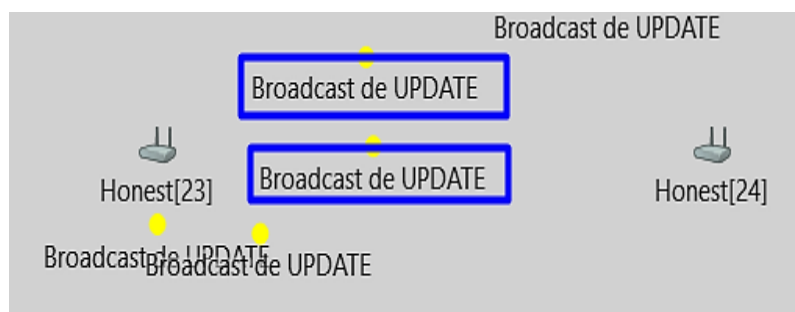
Envió por consola del mensaje de Actualización

```
INFO:MessageType: 12
INFO:Registrando nodo asociado [21]
INFO:Peso del nodo [22] es de: 1
INFO:Informando la asociacion de [21] por el nodo [20]
INFO:Enviando a todos los nodos!
INFO:pathLoss: 74.6698dB coefLoss: 104.67dB rxPowerdBm: -90.6698d
INFO:Potencia LORA recibida: -90.6698 dBm a 316.228 metros!
```

Los mensajes de actualización en la simulación se pueden identificar en la Figura 73, el mensaje **Broadcast de UPDATE** representado por círculos amarillos. Estos mensajes se envían mediante un Broadcast para que todos los nodos en la red puedan recibir la información y actualizar sus contraseñas dinámicas y aumentar su peso. Es importante que todos los nodos actualicen sus contraseñas de forma oportuna para mantener la seguridad de la red, por lo que el tiempo de actualización de contraseñas debe ser lo más breve posible.

Figura 73

Envió de mensajes de Actualización en la Simulación



g. Transacciones del Bloque Candidato

Al realizar correctamente el proceso de autenticación y asociación de un nodo, se puede observar que cada uno de los nodos de la red envía datos a la estación base. La Figura 74 ilustra los mensajes enviados por consola que indican el inicio del proceso de envío de datos. En estos mensajes se incluye la identificación del nodo y una indicación del proceso de envío de datos.

Figura 74

Envío de datos desde los nodos a la estación base

```
INFO:MessageType: 15
INFO:El nodo [24] envia los datos de sus sensores hacia la BSTA [20]!
Event #2137 C=8.881624098878 Network.Honest[7] (HonestModule, Id=12)

INFO:MessageType: 15
INFO:El nodo [14] envia los datos de sus sensores hacia la BSTA [10]!
```

En la Figura 75 se puede observar que la estación base con ID [10] recibe los paquetes de datos enviados por el nodo sensor con ID [13] y emite un mensaje por consola que incluye información del emisor y receptor, así como también información sobre el PH, temperatura y número de transacción .

Figura 75

Envío por consola del mensaje de datos

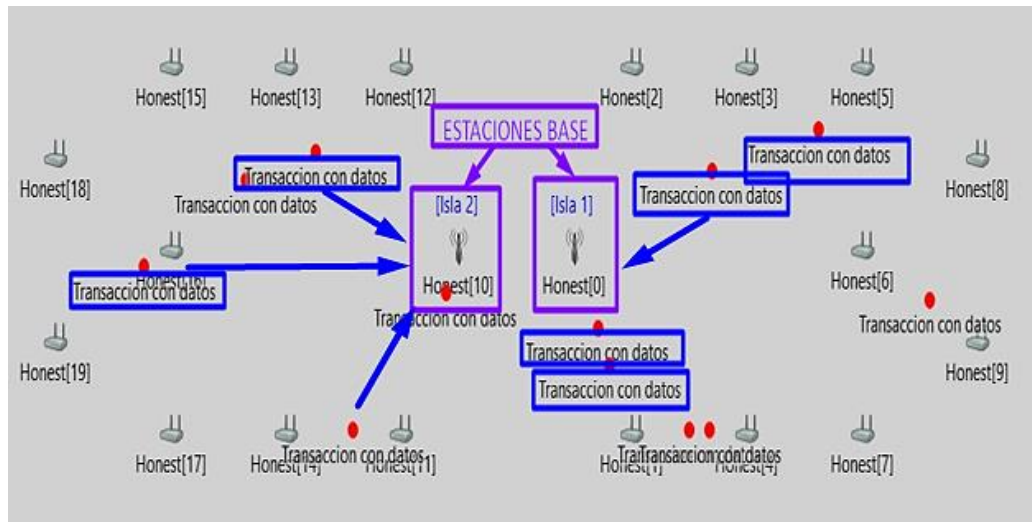
```
INFO:MessageType: 16
INFO:Datos recibidos del sensor [13]
INFO:PH: 3
INFO:TEMP: -18
INFO:ID DESTINO: [10]
INFO:Creando Transaccion N. 1
```

En la Figura 76 se observa que, al iniciar el proceso de transmisión de datos, es posible observar cómo cada uno de los nodos sensores envía una señal que se refleja en la simulación, representada con el color rojo y con el título "*Transacción de datos*".

Estas señales son dirigidas hacia la estación base, donde, mediante el procesamiento de las transacciones recibidas, se genera un bloque candidato en cada una de las estaciones base.

Figura 76

Recepción de transacciones enviadas por los nodos



Para iniciar con el proceso de minería en la cadena de bloques, se crea el primer nodo conocido como el "nodo génesis". Este será el punto de partida para la construcción de toda la cadena de bloques entre las diferentes estaciones base de los nodos. A diferencia de los demás bloques, el nodo génesis no tiene un hash anterior, sino solo un hash propio. Como se puede observar en la Figura 77, el primer nodo de la cadena tiene los parámetros de hash, índice, nonce y, especialmente, el hash anterior con valores de 0. Dentro de cada una de las islas, se generará una versión específica de la cadena de bloques, ya que se supone que los datos de esa isla serán los primeros en agregarse, como se señala en la sección resaltada en azul.

Figura 77

Minado del Bloque Génesis

```

MessageType: 18
** NODO [20] REALIZA MINADO PARA COMPETIR!
*** INDEX 1
** Bloque minado! 000db43b9623bbc268862218c7c07e030edfa86b2203cf2abf19f2a04001082a
** NODO ** [20]newBlock
[[
  {
    "data": "Bloque Genesis",
    "hash": "0",
    "index": 0,
    "nonce": -1,
    "prevHash": "0",
    "time": "0s"
  }, {
    "data": "datos de [20]",
    "hash": "000db43b9623bbc268862218c7c07e030edfa86b2203cf2abf19f2a04001082a",
    "index": 1,
    "nonce": 2920,
    "prevHash": "0",
    "time": "30.041555780644s"
  }
]]
+++ TEMPORAL
1
2920
datos de [20]
000db43b9623bbc268862218c7c07e030edfa86b2203cf2abf19f2a04001082a
30.041555780644s
0

```

Para permitir que los bloques compitan entre sí, se utiliza un nonce, que es un número aleatorio generado por los mineros, que se agrega a un bloque grabado o encriptado de la cadena de bloques. Este nonce se calcula mediante la resolución de un problema matemático específico conocido como "problema de prueba de trabajo" (Proof of Work, PoW). El objetivo de este problema es encontrar un valor nonce que cuando se combina con la información del bloque, produce un hash con un cierto nivel de complejidad, mismo que está determinado por la cantidad de ceros que se agregan al hash. En la Figura 78 se observa cómo cada una de las estaciones base, denominadas con los ID [20], [0], y [10], ha generado su propio nonce para la competición. Solo la estación base que encuentre un nonce válido en menor tiempo se considerará como la ganadora, y podrá agregar su bloque candidato en primer lugar a la cadena principal. En el mensaje publicado por consola también se puede observar el orden en que se agregarán los

bloques en la cadena, siendo la estación base con ID [20] la que iniciará el proceso de publicación de bloques.

Figura 78

Generación de nonce y competencia entre estaciones base

```

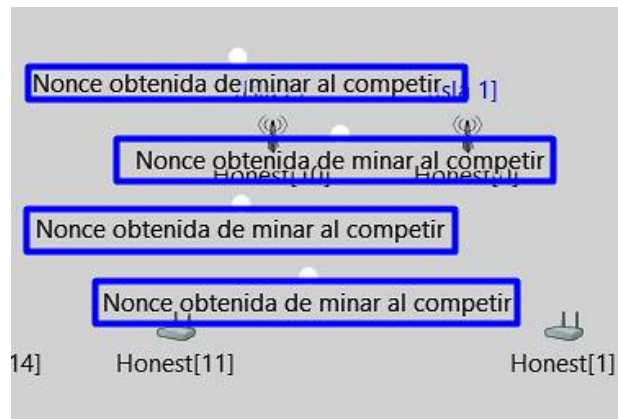
:** NONCE 10899 recibido de [0]
**COMPETEVEC
[20] [0] [10]
4784 10899 11634
*** PRIMER NODO GANADOR [20]
:Enviando con index: 1
:competeVec[0][0]: [20]

```

La Figura 79 muestra el mensaje "*Nonce obtenido de minar al competir*", que se representa dentro de la simulación con círculos de color blanco, mismo que se envía como complemento al mensaje mostrado en la Figura 78. Este mensaje se genera al determinar cuál estación base resuelve la prueba de trabajo en menor tiempo. Por lo tanto, su bloque candidato se agregará en primer lugar a la cadena de bloques. A continuación, las demás estaciones base competirán entre sí para agregar sus bloques candidatos a la cadena principal. A diferencia de una cadena de bloques convencional, aquí no se permite descartar ningún bloque de datos, por lo que se fomenta la competencia entre las estaciones y el orden en que se agregarán los bloques dependerá de la rapidez con que se resuelva la Prueba de Trabajo (POW).

Figura 79

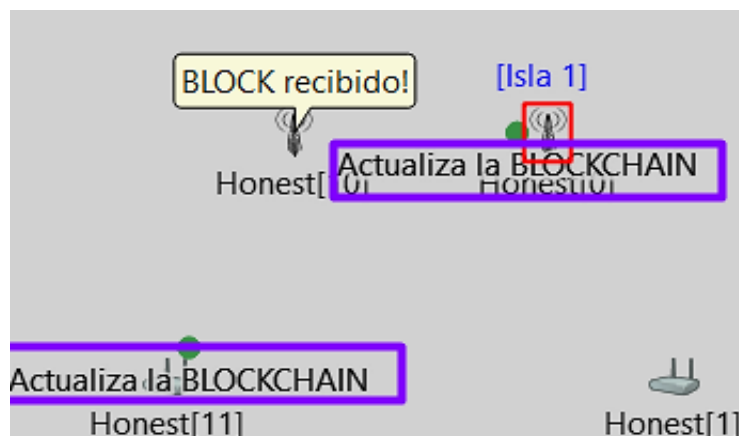
Envío de nonce entre estaciones base



Para actualizar la cadena de bloques central en la simulación, la estación base ganadora envía un mensaje en forma de multicast a todas las estaciones base de las islas como se muestra en la Figura 80. Cada estación actualiza su copia de la cadena al recibir el mensaje "Actualiza la Blockchain". La estación base que recibe el mensaje genera una burbuja informativa que indica que el bloque se ha recibido.

Figura 80

Envío de mensaje de actualización de Blockchain entre las estaciones base



El mensaje indicado en la Figura 80 contiene los siguientes datos, mostrados en la Figura 81. La cual muestra inicialmente el mensaje de "Actualiza la Blockchain" por la estación base con ID [20]. A continuación, se puede observar que se mantiene el

bloque Génesis, cuál es el punto inicial de toda la cadena. A continuación, se agrega el bloque de la estación base con ID [20]. A diferencia del bloque Genesis, este tendrá un hash propio y un hash anterior, ya que tomará como hash anterior el hash del bloque Genesis, el nonce que calculó y el tiempo en el cual se demoró encontrar la respuesta a la prueba de trabajo

Figura 81

Actualización de Blockchain Principal con el nuevo Bloque Agregado

```

:MessageType: 1/
:*** ACTUALIZANDO BLOCKCHAIN!ENVIADO POR [0]
:MI INDEX 3
: NODO [10]
: [
:   {
:     "data": "Bloque Genesis",
:     "hash": "4c7a9577badb8f2ff123e078a9dc7bfcee92a28db4e5a755e57",
:     "index": 0,
:     "nonce": -1,
:     "prevHash": "0",
:     "time": "0s"
:   }, {
:     "data": "datos de [20]",
:     "hash": "0004707c152601aaeb587bdfd93d3521657d18caedc556772886",
:     "index": 1,
:     "nonce": 4784,
:     "prevHash": "4c7a9577badb8f2ff123e078a9dc7bfcee92a28db4e5a755",
:     "time": "30.041555780644s"
:   }, {
:     "data": "datos de [0]",
:     "hash": "000f40eb767b5e77b39239cfc964c3d59e741dedf1f40c439a77",
:     "index": 2,
:     "nonce": 10899,
:     "prevHash": "4c7a9577badb8f2ff123e078a9dc7bfcee92a28db4e5a755",
:     "time": "30.057940290144s"
:   }
: ]
:callSendBlock
:callSendBlock

```

BLOQUE GENESIS

NUEVOS BLOQUES AGREGADOS

4.1.1.2. Prueba 2: Se evaluará el rendimiento del esquema de autenticación en el contexto del Modelo Urbano de LoRAWAN

Esta prueba se realiza con el objetivo de determinar el comportamiento y rendimiento del esquema de autenticación en ambientes más cercanos a la realidad, es decir, sin condiciones ideales. Para ello, se hace uso del Modelo de Pérdida de

Trayectoria por Distancia Logarítmica bajo los parámetros del Modelo Hot Zone Urbano (University George Mason, 2007).

La fórmula indicada en la Ecuación 1 está limitada para el cálculo de la pérdida de trayectoria a una frecuencia de operación de 900 MHz, para utilizar esta expresión en un rango de frecuencias distinto será necesario aplicar un factor de corrección como se señala en la Ecuación 2.

$$[P_L(d)]dB = [P_L(d_0)]dB + 10 n \log_{10} \left(\frac{d}{d_0} \right) \quad (1)$$

Donde:

d = Distancia en metros

n = Exponente de pérdida de trayectoria

X = Variable aleatoria distribuida gaussiana

d_0 = Pérdida del trayecto de referencia

Para la complementación de la prueba realizada se utiliza los parámetros del Modelo Hot Zone Urbano (3GPP, 2018), mismo que se encuentran en la Tabla 13, este modelo asume que las antenas se encuentran a altura de techo (15 m). Además, omite posibles pérdidas del sistema respecto al transmisor/receptor, de manera que los valores referenciales expresados son los más relevantes para el cálculo del Pathloss (Kocan et al., 2017).

$$\delta = 21 * \log_{10} \left(\frac{f}{900MHz} \right) \quad (2)$$

Para la complementación de la prueba realizada se utiliza los parámetros del Modelo Hot Zone Urbano mismo que se encuentran en la Tabla 13.

Tabla 13

Valores referenciales usados en el modelo Hot Zone

| Valores Referenciales | Descripción |
|----------------------------|---------------|
| Perdida | 30 (dB) |
| Frecuencia | 900 (MHz) |
| Distancia | 1m |
| Perdida Exponencial | 3 coeficiente |

Los valores referenciales indicados en la Figura 82 se encuentra especificados dentro del archivo .ned, estos son utilizados dentro de la simulación como base para el cálculo de la pérdida de potencia , el coeficiente de pérdida y la potencia de recepción.

Figura 82

Valores referenciales programadas dentro del archivo .ini

```

int referenceLoss = default(30); //30dB de perdida a 1m
int referenceFreq = default(900); //MHz
int referenceDistance = default(1); //metros
int pathLossExponent = default(3); //coeficiente

```

Dentro de la simulación se aplica la fórmula del Modelo de Pérdida de Trayectoria por Distancia Logarítmica como se indica en la Ecuación 1, en la cual se encuentra aplicado el factor de corrección para que su funcionamiento sea correcto con frecuencias distintas a 900MHz , para el cálculo de la pérdida de Trayectoria el código toma los valores establecidos del *pathLossExponent* ,*referenceDistance* y *referenceFreq*, así como también valores obtenidos del funcionamiento de la simulación como son el valor de frecuencia (freq) y distancia (distante).

Para la obtención del coeficiente de pérdida se tomará el valor de la pérdida de trayectoria previamente calculado más el valor de *referenceLoss* que se muestra en la Figura 83 y que está definido dentro de la simulación.

Figura 83

Fórmula para la obtención de la pérdida de trayectoria y coeficiente de pérdida

```
float pathLoss = 10*pathLossExponent*std::log10(distance/referenceDistance) + 21 * std::log10(freq/referenceFreq);
float coefLoss = pathLoss + referenceLoss;
```

Pérdida de Trayectoria

Coeficiente de Pérdida

La Figura 84 muestra la parte del código que permite realizar el cálculo de la distancia se utiliza el archivo .csv de posiciones, dentro del cual se encuentran las coordenadas de cada uno de los nodos, para que Omnet ++ obtenga cada una de las posiciones del archivo .csv se lee fila por fila y se almacenan los datos en vectores de donde se extraerán las coordenadas x1, x2, y1, y2 respectivas, cuando ya se han obtenido estos datos se realiza el cálculo de la distancia entre dos puntos para conocer la distancia total entre los dos nodos.

Figura 84

Cálculo de la distancia

```
float AbstractModule::calcularDistancia(string origenID, string destinoID) {
    std::string path = "./data/positions.csv";
    std::fstream file;
    file.open(path, std::ios::in);

    if(!file.is_open()) throw std::runtime_error("No se pudo abrir el CSV!");
    std::string line, word;

    std::vector<std::vector<string>> content;
    std::vector<string> row;

    int x1,x2,y1,y2; //Formula distancia entre dos puntos

    while(getline(file,line))
    {
        row.clear();
        std::stringstream str(line);
        while(getline(str, word, ','))
            row.push_back(word);
        content.push_back(row);
    }
    file.close();
}
```

Se selecciona la ruta del archivo CSV

Comprobación en caso de problemas con el CSV

Definición de variables auxiliares

Lectura del CSV fila por fila guardando los datos separados en un vector de vectores

```

for (int i = 0; i < content.size(); i++) {
    for (int j = 0; j < content[i].size(); j++) {
        if (content[i][j] == origenID) {
            x1 = std::stoi(content[i][j + 1]);
            y1 = std::stoi(content[i][j + 2]);
        }
        if (content[i][j] == destinoID) {
            x2 = std::stoi(content[i][j + 1]);
            y2 = std::stoi(content[i][j + 2]);
        }
        //EV << content[i][j] << " ";
    }
    //EV << endl;
}

float distCalc = std::sqrt(std::pow((x2 - x1),2) + std::pow((y2 - y1),2));
//EV << "Distancia = " << distCalc << endl;
return distCalc;
}

```

Obtiene las posiciones de los nodos origen/destino y almacena las coordenadas X e Y respectivas

Distancia entre dos puntos

Para el cálculo indicado en la Figura 85 se establece el valor de la frecuencia en 868 MHz debido a que la mayoría de los dispositivos LoRAWAN trabajan en esta frecuencia. Para obtener el valor de la potencia de Recepción (*rxPowerdBm*) se aplica la fórmula indicada en la Figura 85 la cual realiza la sustracción entre el coeficiente de pérdida (*coefLoss*) y la potencia de transmisión (*txPowerdBm*). El valor de la potencia de transmisión (*txPowerdBm*) mostrado en la Figura 85 muestra cómo se encuentra establecido dentro del código con el valor de 14 dBm el cual cumple con la asignación de frecuencias y los requisitos reglamentarios para ISM usados en Europa, estos valores son usados de forma frecuente dentro de módulos LoRaWAN Clase A como indica Andreu (2018) mismos que presentan exigencias energéticas mínimas y manejan el protocolo LoRaWAN 1.0.

Figura 85

Valor de Frecuencia

```
float loraTxFreq = 868; //MHz
```

```
float rxPowerdBm = txPowerdBm - coefLoss; Potencia de Recepción
```

```
int loraTxPower = 14; //14dBm
```

Al utilizar la tecnología LoRAWAN y las fórmulas indicadas anteriormente, se obtienen los valores de pérdida de potencia *pathloss*, entre otros indicados la Figura 86 valores esenciales para el diseño y optimización de la red, conocer la pérdida de potencia ayuda a comprender las limitaciones del sistema. Además, se envía información sobre el *coefLoss*, información utilizada para calcular la pérdida de potencia en función de la distancia entre el transmisor y el receptor, lo que permite optimizar la configuración de la red para obtener el mejor rendimiento y alcance posible, otro valor que se muestra *rxPowerdBm*, que es la cantidad de potencia de la señal recibida por el dispositivo receptor. Este valor es importante ya que determina la calidad de la señal recibida, ayudando a detectar posibles problemas de cobertura o interferencia, también se indica a la distancia a la cual se encuentra cada uno de los nodos

Figura 86

Resultados de la aplicación de las fórmulas de Perdida Logarítmica

```
pathLoss: 104.67dB rxPowerdBm: -90.6698dBm
Potencia LORA recibida: -90.6698 dBm a 316.228 metros!
```

Cada uno de los valores tanto de PathLoss como de la Potencia de Recepción obtenidos durante los distintos escenarios de simulación fueron analizados, a fin de determinar si el modelo de autenticación propuesto funcionaba según lo esperado en un ambiente no ideal. De acuerdo con lo antedicho, ninguno de los valores de la potencia en la recepción calculados, fue inferior al límite de -137 dBm (threshold) establecido por el datasheet del chipset LoRa SX1272 fabricado por Semtech Corporation (2019), siendo este último un modelo de referencia usado para abstraer las características principales de LoRa a la simulación en OMNET++.

4.1.1.3. Prueba 3: Simulación de ataques de diccionario y denegación de servicio

Para evaluar la resistencia del esquema de autenticación propuesta, se llevaron a cabo dos tipos de ataques: el ataque de diccionario y el ataque de denegación de servicio (DoS). El primer ataque se realizó para verificar el funcionamiento del protocolo SEECR⁷ y determinar si es posible descubrir las contraseñas de los nodos mediante el uso de un diccionario. El segundo ataque se realizó con el fin de evaluar si es posible afectar el funcionamiento de los nodos de la red a través de una sobrecarga de solicitudes de autenticación.

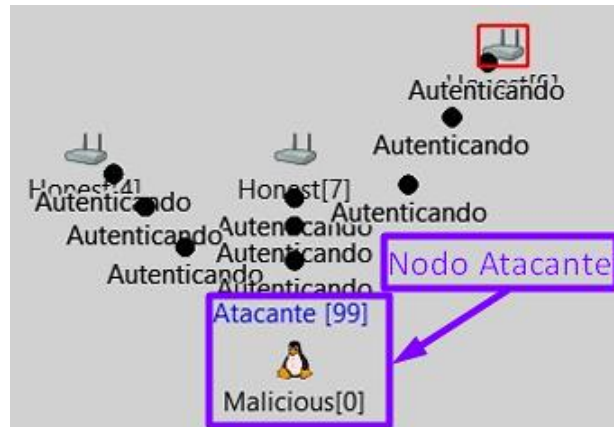
a) Ataque de Diccionario

Para identificar el nodo atacante en la simulación, se utiliza un icono diferente a los usados en la red Tagle. En la Figura 87 se muestra como el nodo atacante realiza una solicitud de autenticación a tres nodos diferentes de la red, los cuales responderán con un mensaje que contiene una contraseña más salt hasheada. El nodo atacante deberá descifrar esta contraseña para poder formar parte de la red. Si el nodo atacante envía una contraseña incorrecta tres veces, el nodo se identifica como atacante y se informa a la red que no deben enviar datos, a través de esa ruta.

⁷ SEECR : Protocolo de enrutamiento para redes de sensores inalámbricas submarinas Seguro Energético Eficiente y Cooperativo

Figura 87

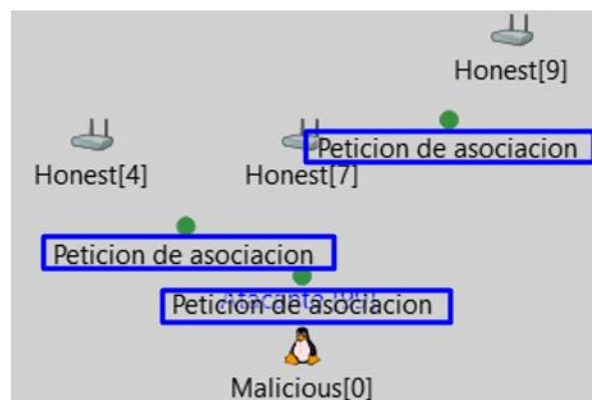
Envío de mensajes de Autenticación desde el nodo Atacante a la Estación Base



El ataque comienza con el envío de una solicitud de asociación desde el nodo atacante hacia los nodos sensores, tal como se muestra en la Figura 88 el mensaje “Petición de Asociación” denotado con puntos de color verde tienen como origen el nodo Atacante y como destino los nodos sensores con ID [4], [7], [9]. De esta manera, el nodo atacante busca integrarse a la red Tangle.

Figura 88

Envío de Petición de Asociación desde el Atacante a los nodos Sensores



Al iniciar con el ataque dentro de la consola se observa el mensaje “Atacando “ indicando que el nodo atacante con ID [99] está realizando una petición de asociación como se indica en la Figura 89.

Figura 89

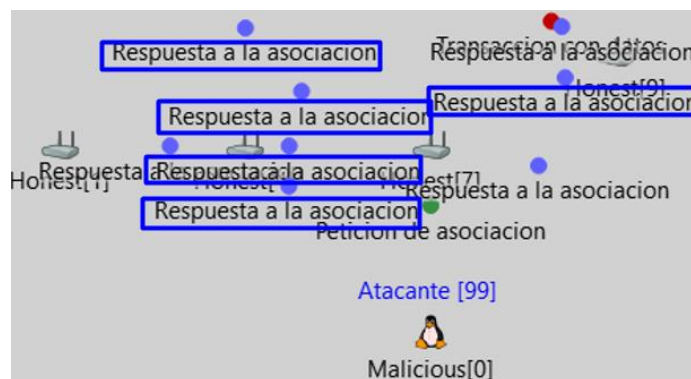
Envía mensaje por consola de la Petición de Asociación

```
INFO:*Atacando...
INFO:*Nodo [99] envia peticion de asociacion!
```

A continuacion los nodos sensores enviaron un mensaje denominado “Respuesta de Asociacion “ el cual se puede visibilizar en la Figura 90 dentro de la simulacion con circulos de color celeste hacia el nodo atacante , en este mensaje se encontrara la contraseña mas salt hasheada que el nodo atacante debera decifrar si desea formar parte de la red.

Figura 90

Mensaje Respuesta de Asociación de los Nodos a el Atacante



Dentro de la consola de Omnet ++ se observara que los ID [99] envian un hash para la prueba de trabajo que debera realizar el nodo atacante, dentro de la Figura 91 se observa que los hash enviados desde los nodos sensores hacia el nodo atacante son diferentes , por lo cual el nodo atacante tendra que probar diferentes contraseñas y realizar un hash de las mismas para poder enviar a cada uno de los nodos sensores para su autenticacion , cada uno de los paquetes enviados por el nodo atacante seran registrados en los nodos sensores dentro de la variable Qk para luego realizar su comparacion con los paquetes salientes que se almacenan en la variable Qj y de esta forma determinar la existencia de un nodo atacante.

Figura 91

Aplicación del Protocolo SEECR

```

:MessageType: 9
RECIBIDO MSG_ASO con ID: [99]
Enviando hash para PoW: c9c859a70d9c003aeea59c02c9dceb33e9340bca065b8a343dc28f3e14468ca
hacia el nodo [99]; desde el nodo [9]
Qk: 1 Qj: 0
:PathLoss: 271.522dB coefLoss: 301.522dB rxPowerdBm: -287.522dBm
:Potencia LORA recibida: -287.522 dBm a 1.15273e+009 metros!
vent #2468 t=16.282920263701 Network.Honest[1] (HonestModule, id=9) on Petición de asociacio
:MessageType: 9
RECIBIDO MSG_ASO con ID: [99]
Enviando hash para PoW: 3dcd447a714026264d1543793212aded435b5c1141ceea8258b07a24db8665cc
hacia el nodo [99]; desde el nodo [4]
Qk: 1 Qj: 0
:PathLoss: 271.522dB coefLoss: 301.522dB rxPowerdBm: -287.522dBm
:Potencia LORA recibida: -287.522 dBm a 1.15273e+009 metros!
vent #2469 t=16.320828715496 Network.Honest[7] (HonestModule, id=12) on Petición de asociaci
:MessageType: 9
RECIBIDO MSG_ASO con ID: [99]
Enviando hash para PoW: 015caf04d90af794770f11715ed6a4429dbb3e554dc07d0496133b57b6bfcdb5
hacia el nodo [99]; desde el nodo [7]
Qk: 1 Qj: 0
:PathLoss: 271.522dB coefLoss: 301.522dB rxPowerdBm: -287.522dBm
:Potencia LORA recibida: -287.522 dBm a 1.15273e+009 metros!

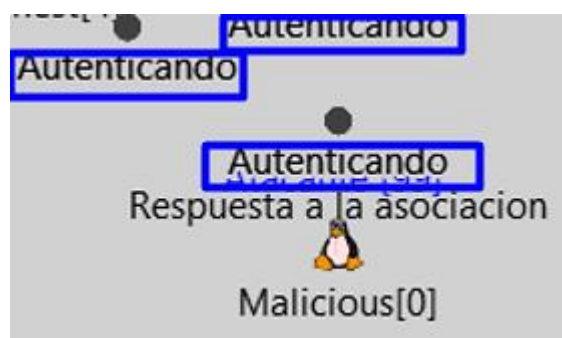
```

ID NODO ATACANTE
 Hash para Prueba de Trabajo
 Nodo de la red IOTA que recibe el mensaje
 Activación de Variables Qk y Qj

Al realizar el proceso de autenticación entre el nodo atacante y los nodos sensores como se muestra la Figura 92 se observa dentro de la simulación que el nodo atacante representado con un pingüino envía los mensajes “Autenticando” los cuales se pueden distinguir a través de círculos de color negro, estos mensajes contienen las posibles contraseñas hash para su autenticación.

Figura 92

Envío de mensaje de Autenticando del Atacante a los nodos



Dentro de la consola se muestra que el nodo [99] ha iniciado el ataque conocido como "Ataque de asociación" y a continuación en la Figura 93 se presentan los hashes utilizados para intentar penetrar la red. Se observa que el nodo malicioso realiza cuatro intentos de autenticación a cada uno de los nodos sensores, sin embargo, estos intentos no son exitosos debido a que las contraseñas utilizadas no son correctas, a esto también se suma la detección de la presencia del nodo malicioso, para lo cual se aplica el protocolo SEECR.

Figura 93

Intentos realizados desde el Atacante a los nodos sensores

```

INFO:*El nodo [99] ha recibido el hash enviado por el nodo [9]
INFO:*Ataque de asociacion...
INFO:*Ataque de asociacion N. 1
INFO:*Atacando con Hash: 97ba48c6089637ff0a9bbdbb5d02b92f441e71dca8ba3f102f715c71162b6e10
INFO:*Ataque de asociacion N. 2
INFO:*Atacando con Hash: dd92e5f11a9df5631c8143e56c763d904c03a5670cdb70d66d4bcb9d87089b36
INFO:*Ataque de asociacion N. 3
INFO:*Atacando con Hash: e71287c63f586f2925d36cd301ec6c79bd827b2c517918d21bb640ab2e3a9acb
INFO:*Ataque de asociacion N. 4
INFO:*Atacando con Hash: c26c2d78854bb24135a44e73273629d602c275bba8daf8b9fe88c872b457a253
** Event #2473 +=16 522850555408 Network Honest[4] (HonestModule id=9) on Respuesta a la
Intentos realizados

INFO:*El nodo [99] ha recibido el hash enviado por el nodo [4]
INFO:*Ataque de asociacion...
INFO:*Ataque de asociacion N. 1
INFO:*Atacando con Hash: 0cc1eec133322bb85194792826c5478a0e6bdcbca798c57893dedec508be309c
INFO:*Ataque de asociacion N. 2
INFO:*Atacando con Hash: 04cac1474f8912961f5d9bd62351d84fafdcae314f7d1d11b455265c9a8d2c26
INFO:*Ataque de asociacion N. 3
INFO:*Atacando con Hash: fe83cad24254f77442924e086097f781d17c6abb5567c28f02d100c4ae53d6b3
INFO:*Ataque de asociacion N. 4
INFO:*Atacando con Hash: 6b57a5197508dbc13c39f8e8ff9a3c98653018fac5572f90436a090c76eee9f1

```

Tras recibir el hash enviado por el nodo atacante, el nodo sensor procede a su validación. En la Figura 94 se muestra el mensaje por consola, que muestra el hash enviado por el nodo malicioso, identificado como *HASH*, y el hash actual del nodo sensor, identificado como *actualHash*. Si no se encuentran coincidencias entre ambos hashes, ¡la estación base emite el mensaje "*El nodo ha recibido el hash y no lo valida!*", lo que indica que el nodo atacante se ha identificado como tal. Además, el mensaje

incluye un análisis de los paquetes entrantes y salientes del nodo sensor que detectó al nodo malicioso.

Figura 94

Indicamos que las estaciones base no lo validan

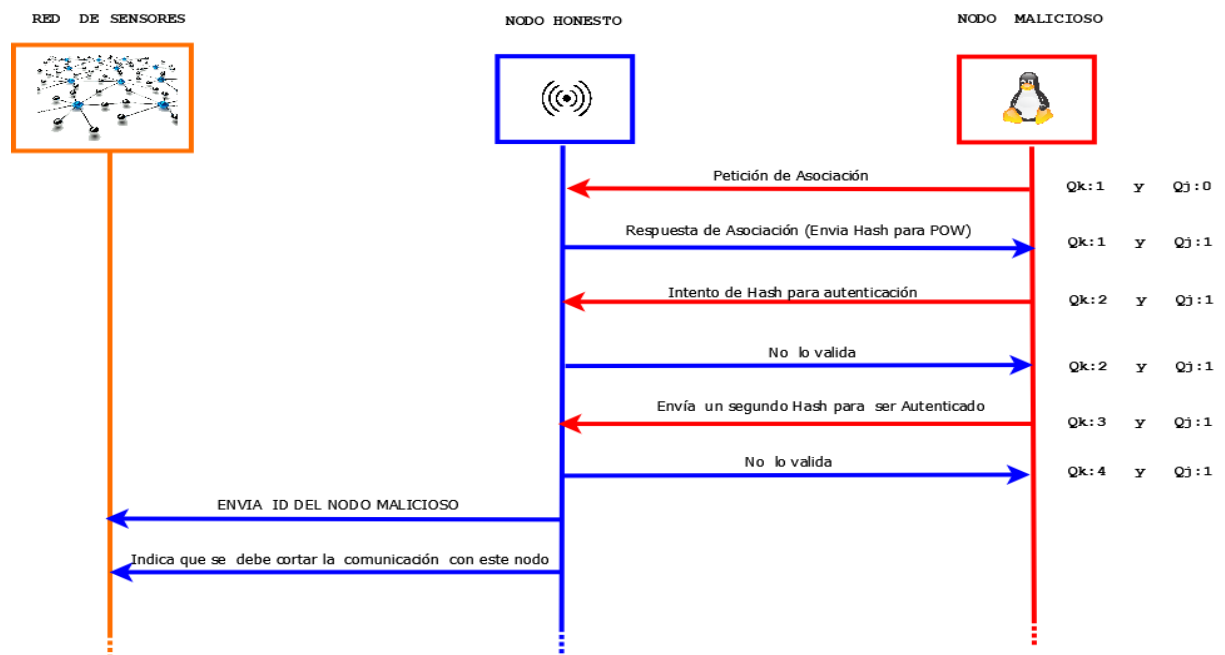
```

):MessageType: 7
):RECIBIDO MSG AUTH con ID: [99]
):HASH: b51afaa799cc0aceb8bd2c50599d87461df7e3aef1fa100095991e918e537811
):actualHash: 138e7461d245c2fd0feef68d14663fcd50680c89c8fdbabcf012285343fcc6ec pastHash: 6b57a5197508dbc13c39f8e8ff9a3c98653018fac
):El nodo ha recibido el hash y NO lo valida!
):*PathLoss: 310.605dB & RxPowerdBm: -296.605dBm
):Potencia LORAWAN recibida: -296.604980 dBm a 2.31475e+09 metros!
):Nodo [99] ya registrado en el vector de control!
):Nodo [4] actualiza los valores de Qj: 1 y Qk: 6 para el nodo [99]
    
```

El conteo de paquetes realizado a través de la aplicación del protocolo SEECR para la detección del nodo malicioso se realiza como se indica en la Figura 95 en el cual se indica como se toma en consideración cada uno de los mensajes, para el aumento en el número de paquetes en las variables Qk y Qj correspondientes a los paquetes de entrada y salida respectivamente.

Figura 95

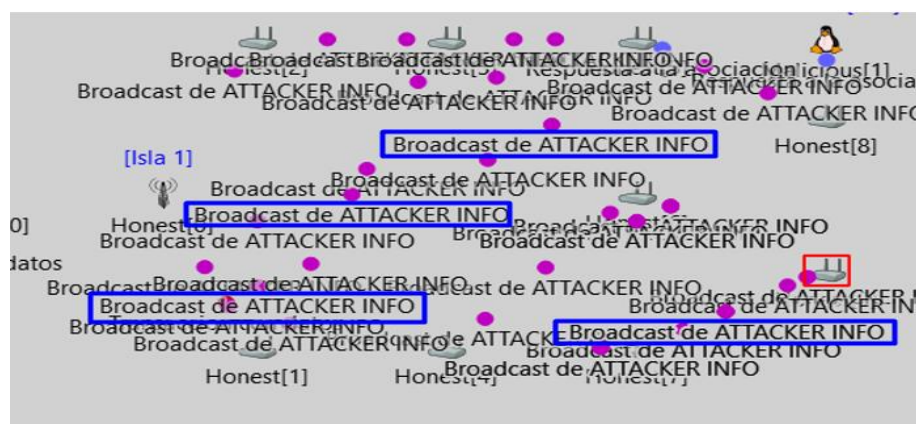
Diagrama de secuencia del envío de mensajes



Cuando ya se ha identificado el nodo malicioso como se indica dentro de la simulación de la Figura 96 los nodos sensores envían el mensaje **“Broadcast de Atacker Info”** “en forma de Broadcast para que toda la red conozca cual es el ID del nodo malicioso y de esta forma corta la ruta de envío de mensajes hacia el mismo, tomando otras rutas para el envío de información de forma segura.

Figura 96

Envío de mensaje informativo de la existencia de un nodo malicioso



Finalmente, en la Figura 97 se indica que el nodo atacante con ID [99] ha sido registrado y marcado como "ATACANTE". Además, se proporciona el ID del nodo sensor que detecta la actividad sospechosa y que inicia con el envío de los mensajes informativos en forma de broadcast.

Figura 97

Informamos el ID del nodo atacante

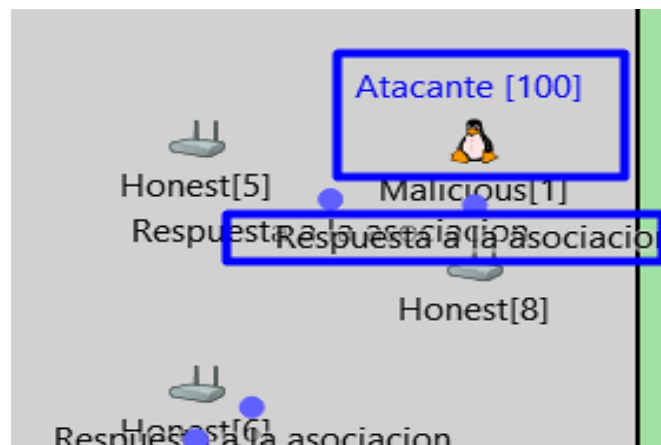
```
INFO:MessageType: 14
INFO: Registrando nodo atacante [99]
INFO: Informando que el nodo [99] se ha marcado como ATACANTE! gracias al nodo [9]
INFO: pathLoss: 271.655dB coeLoss: 301.655dB rxPowerdBm: -287.655dBm
INFO:Potencia LORA recibida: -287.655 dBm a 1.16458e+09 metros!
```

b) Ataque DOS (Distributed Denial of Service, Negación de Servicio Distribuido)

El ataque de DOS se realiza a través del nodo atacante [100] el cual envía varias solicitudes de autenticación hacia un nodo sensor de la red Tangle para agotar sus recursos como se indica en la Figura 98, en la simulación se puede observar como el Atacante envía varias peticiones de asociación a las cuales el nodo sensor solo responde con respuesta de asociación sin pasar al paso de autenticación hasta el punto de determinar que este nodo es malicioso y cortar la comunicación con el mismo.

Figura 98

Envío de Respuesta de Asociación del nodo a el Atacante



Al iniciar el ataque, se muestra en la Figura 99 el mensaje emitido por consola "*Atacando...*" y también se indica el tipo de ataque que se está realizando. En este primer mensaje, el nodo atacante con ID [100] realiza una solicitud de asociación, como se indica en la Figura 98

Figura 99

Inicio de Ataque DOS

```
INFO:*Atacando...
INFO:*Nodo [100] realiza ataque DOS N. 1
```

En la Figura 100, se puede distinguir que cuando el nodo sensor recibe la solicitud de asociación, envía un mensaje de respuesta de asociación que incluye el hash de una combinación de contraseña y sal para la realización de la prueba de trabajo. Este mensaje generado en la consola también nos muestra la dirección del mensaje, que en este caso es "desde el nodo [8] hacia el nodo [100]"

Figura 100

Mensaje de respuesta de Asociación

```
NFO:MessageType: 9
NFO:RECIBIDO MSG ASO con ID: [100]
NFO:Enviando hash para PoW: 34950c8ca6391cala3cbb509955882b547bd96f5a0f4ef470cc34c197ee952f5
NFO:hacia el nodo [100]; desde el nodo [8]
NFO:pathLoss: 280.236dB coefLoss: 310.236dB rxPowerdBm: -296.236dBm
NFO:Potencia LORA recibida: -296.236 dBm a 2.25011e+09 metros!
```

En la Figura 101 el nodo malicioso está enviando repetidas solicitudes de autenticación al nodo sensor, a pesar de que el nodo sensor envía el hash para la realización de la prueba de trabajo y la autenticación, el nodo malicioso no hace caso y continúa enviando solicitudes de asociación con el objetivo de agotar los recursos del nodo sensor. Por lo tanto, el nodo atacante no deja de enviar múltiples mensajes al nodo sensor.

Figura 101

Realiza diferentes ataques

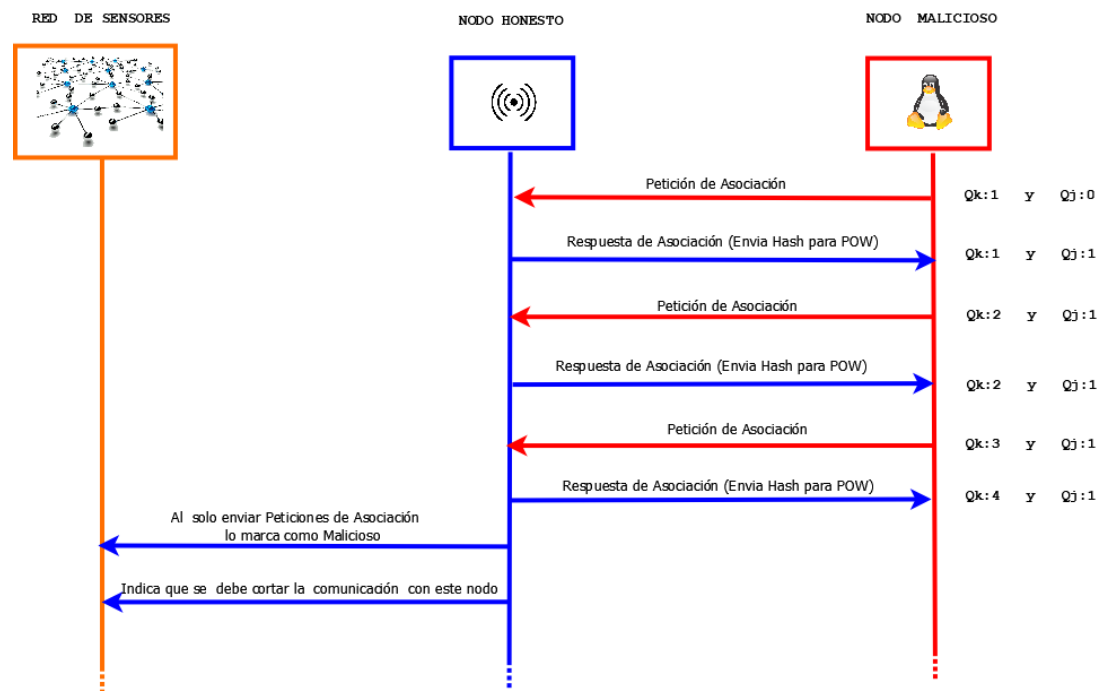
```
*El nodo [100] ha recibido el hash enviado por el nodo [8]
*Nodo [100] realiza ataque DOS N. 2

*El nodo [100] ha recibido el hash enviado por el nodo [8]
*Nodo [100] realiza ataque DOS N. 8
```

En el caso de un ataque de Denegación de Servicio (DOS), el nodo atacante realiza múltiples solicitudes de asociación tal como se muestra en la Figura 102. El nodo receptor de estas solicitudes responde con una confirmación de asociación. Sin embargo, dado que el nodo atacante no busca autenticarse, sino más bien agotar los recursos del nodo atacado, seguirá enviando solicitudes de asociación. Después de cuatro intentos, el nodo receptor asumirá que el nodo atacante es malicioso y lo marcará como tal, enviando un mensaje informativo a todos los nodos de la red.

Figura 102

Diagrama de secuencia de Ataque DOS



4.1.1.4. Prueba 4: Creación de diferentes ambientes (Aumento del número de nodos e islas), para evaluar la escalabilidad de la red.

a) Aumento de número de Islas

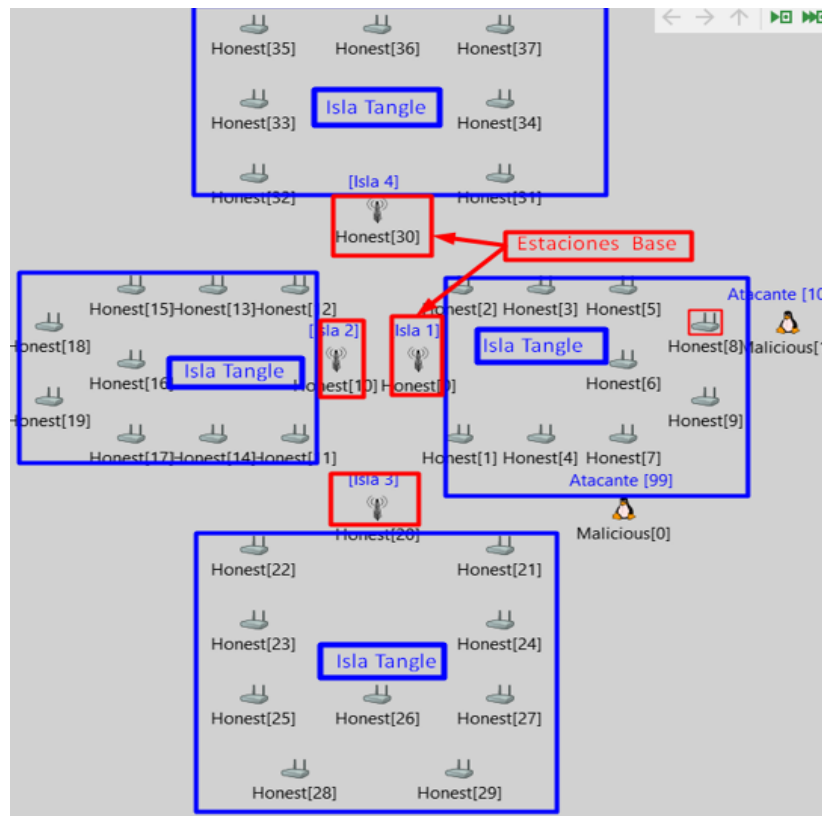
Esta prueba se centra en revisar la escalabilidad del esquema de autenticación y su comportamiento ante una mayor cantidad de islas Tangle (islas de nodos sensores). De igual modo, se examinará el aumento de nodos en las islas con el fin de detectar

posibles retrasos o cambios en el proceso de autenticación y generación de la cadena de bloques.

La Figura 103 muestra un sistema conformado por cuatro islas, cada una de las cuales está compuesta por nueve nodos. En la figura también se puede observar un aumento en el número de estaciones, lo cual puede afectar el proceso de generación de la cadena de bloques.

Figura 103

Ambiente con cuatro islas



Al realizar el aumento de islas dentro de la topología de red establecida, se puede observar que hay un aumento en los tiempos de obtención de contraseñas aspecto que se puede observar en la Figura 104. En la topología formada por tres islas (Figura A), los nodos sensores inician el proceso de obtención de contraseñas a los 423 us. En cambio, en la topología que cuenta con cuatro islas (Figura B), el proceso comienza a

partir de los 3 ms, existiendo una diferencia de 2,577 us entre ambos. A nivel general del funcionamiento de la red, esta diferencia no genera afectación.

Figura 104

Tiempo de Inicio de Obtención de Contraseñas

Figura A. Topología con 3 islas

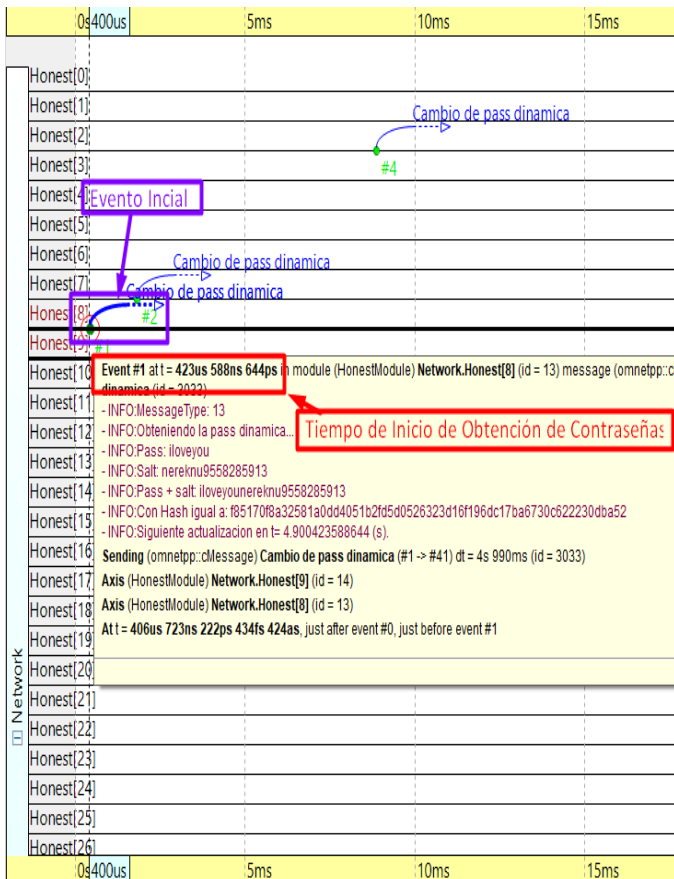
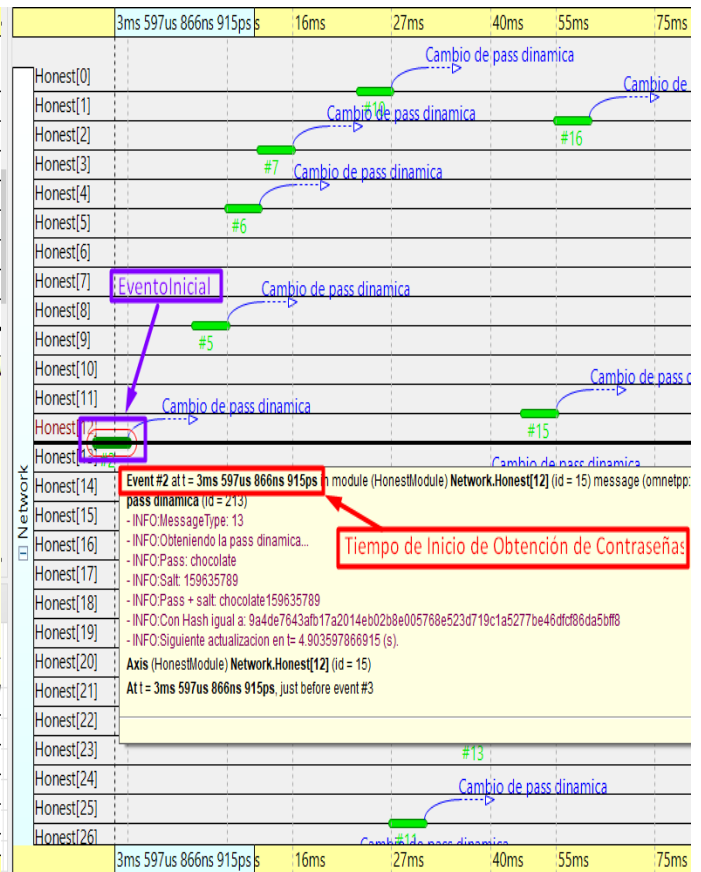


Figura B. Topología con 4 islas



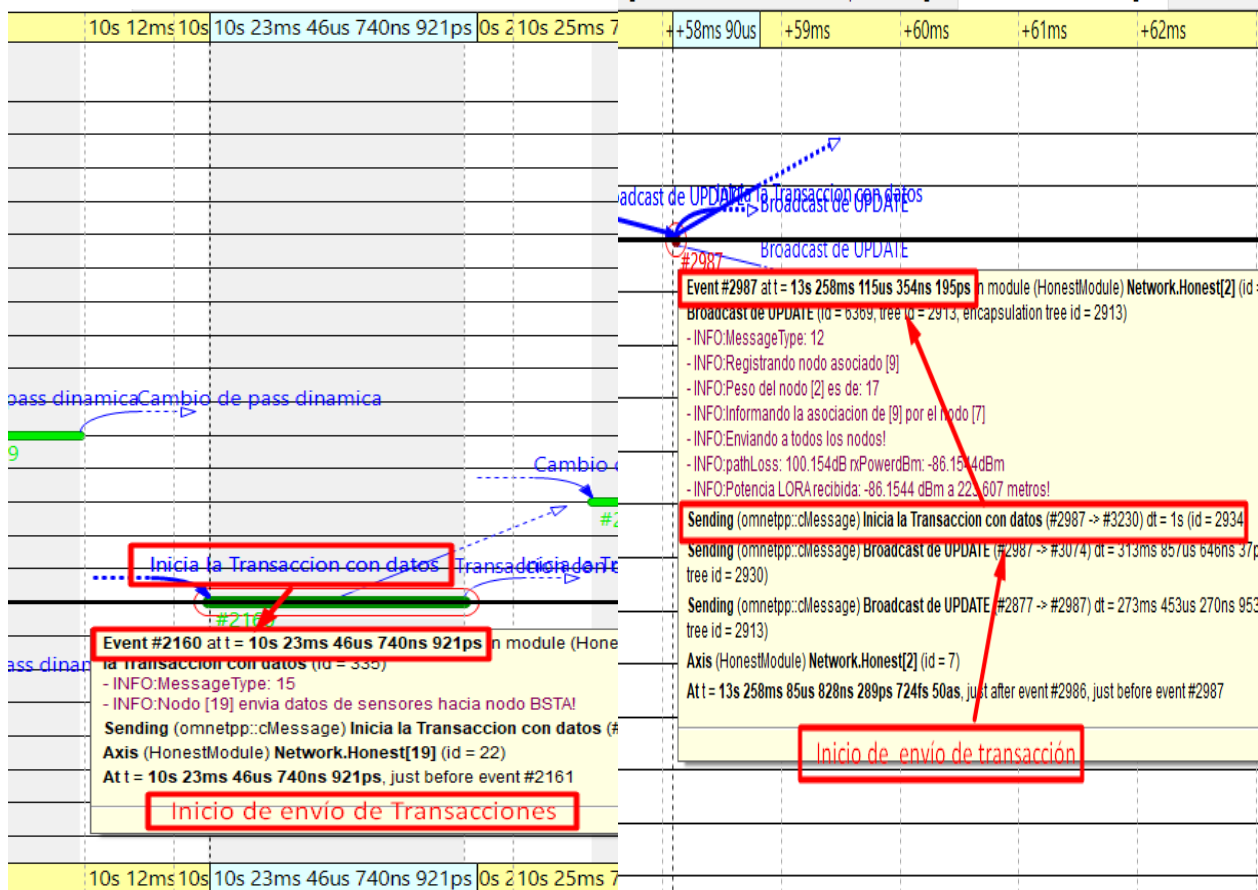
Al culminar todo el proceso de autenticación de los nodos, se inicia el envío de datos, proceso que dentro de la simulación se conoce como envío de transacciones proceso que se maneja en un tiempo determinado como se indica en la Figura 105. A diferencia de la comparación anterior, en esta ocasión, la topología de tres islas (Figura C) tarda 10 segundos en iniciar el envío de transacciones, mientras que la topología de cuatro islas (Figura D) tarda 13 segundos en hacerlo. Esto indica que, a medida que se van cumpliendo cada uno de los procesos dentro de la simulación, el tiempo entre las dos simulaciones aumenta, mostrando una diferencia más marcada.

Figura 105.

Tiempo de Inicio de Envío de Transacciones

Figura C. Topología con 3 islas

Figura D. Topología con 4 islas



Uno de los aspectos más visibles al agregar una isla es que la cadena de bloques se vuelve mucho más larga. Esto se debe a que una nueva isla competirá y producirá su propio candidato. Es por lo que, en algunos casos, las estaciones base que realizan la prueba de trabajo de forma más rápida pueden enviar su bloque candidato en menos tiempo, lo que se visualiza de forma consecutiva dentro de la cadena de bloques central, como se muestra en la Figura 106.

Figura 106

Cadena de Bloques actualizada entre las diferentes islas.

```

*** ACTUALIZANDO BLOCKCHAIN!ENVIADO POR [20]
MI INDEX 14
** NODO ** [10]
{
  "data": "Bloque Genesis",
  "hash": "4c7a9577badb8f2ff123e078a9dc7bfcee92a28db4e5a755e57ffafad25176e8",
  "index": 0,
  "nonce": -1,
  "prevHash": "0",
  "time": "0s"
}, {
  "data": "datos de [20]",
  "hash": "000bbcf352f36880f1efa0dbea6b70f9e9e2a1bdc365478cb79a17e25b0bdb84",
  "index": 1,
  "nonce": 34440,
  "prevHash": "4c7a9577badb8f2ff123e078a9dc7bfcee92a28db4e5a755e57ffafad25176e8",
  "time": "50.086800820894s"
}, {
  "data": "datos de [10]",
  "hash": "00087c8b3c70e1e060d1460e2a064e9a589392a1080cf92c397c21b8f7cc56bd",
  "index": 2,
  "nonce": 22647,
  "prevHash": "000bbcf352f36880f1efa0dbea6b70f9e9e2a1bdc365478cb79a17e25b0bdb84",
  "time": "50.126992196836s"
}, {
  "data": "datos de [30]",
  "hash": "000e1b6af45a668d5f52ec84c5be73f0589fd46d7c96d422f54b74f549ba5522",
  "index": 3,
  "nonce": 26600,
  "prevHash": "4c7a9577badb8f2ff123e078a9dc7bfcee92a28db4e5a755e57ffafad25176e8",
  "time": "50.069306917359s"
}, {
  "data": "datos de [20]",
  "hash": "000adf9b54d091b7c301be07811d50e616a7667253a720e2b5bebe1b703d9bbd",
  "index": 5,
  "nonce": 42907,
  "prevHash": "000bbcf352f36880f1efa0dbea6b70f9e9e2a1bdc365478cb79a17e25b0bdb84",
  "time": "60.266264349009s"
}, {
  "data": "datos de [0]",
  "hash": "000c6465925eb3f7669b6686fa8b9c9d116f493a0f72cb58f24626743212cc83",
  "index": 6,
  "nonce": 42603,
  "prevHash": "0000ee8a79858072e79451a84b1c4c3af453d25e96c7d3cf34437ef5fdcfa391",
  "time": "60.191544580119s"
}, {

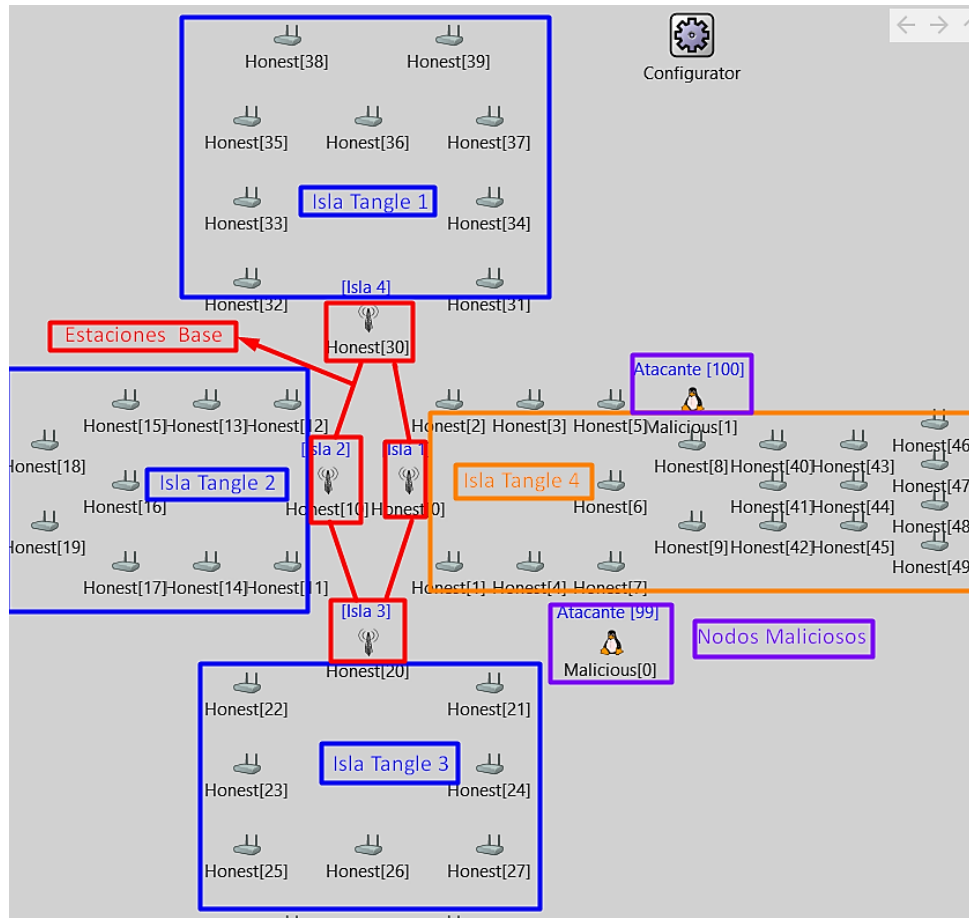
```

b) Aumento de nodos en la Isla

Dentro de esta prueba se busca revisar que tan escalable es la red y su comportamiento al tener una gran cantidad de nodos, para lo cual se ha añadido una cantidad de diecinueve nodos en una de las islas y se mantiene una cantidad de nueve nodos en las tres islas restantes como se muestra en la Figura 107.

Figura 107

Escenario con una isla aumentada nodos

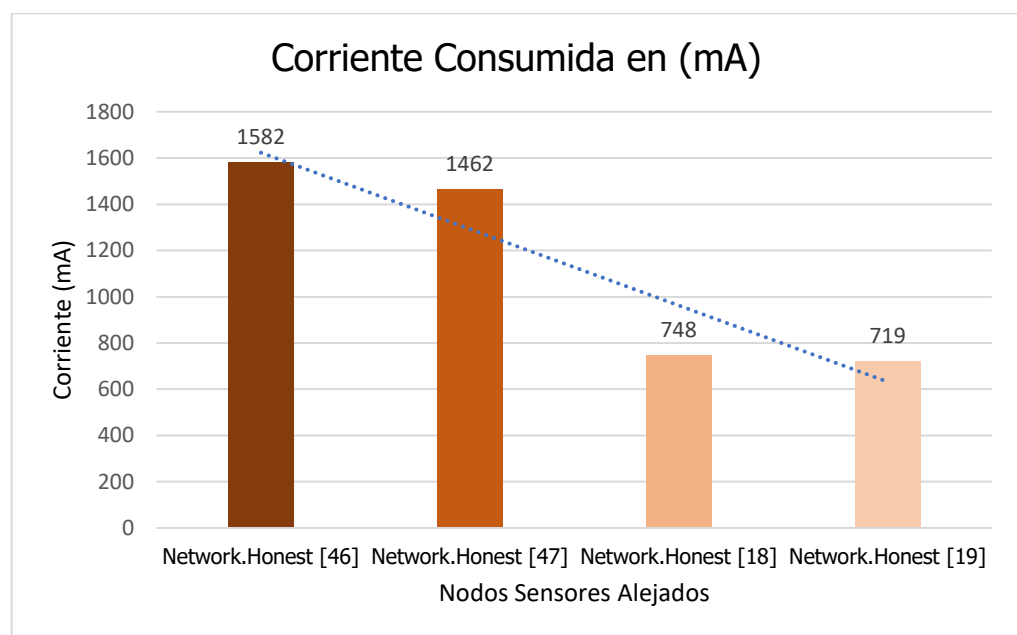


Uno de los factores que se puede observar que cambian entre la isla con mayor cantidad de nodos es la diferencia en el consumo de corriente que tienen los nodos que se encuentran más alejados de la estación base. Como se muestra en la Figura 108, la corriente total consumida por los nodos de ID [46] y [47] pertenecientes a la *Isla Tangle 3* es de 1,582 y 1,462 [mA] respectivamente, las cuales son muy altas debido a que tienen una diferencia que va desde 714 a 834 [mA] en comparación con las consumidas en los nodos con ID [18] y [19] de la *Isla Tangle 2* cuyos valores son de 748 y 719[mA], considerándose relativamente bajas ya que los nodos más cercanos a la estación base consumen 730 [mA]. Esto indica que, al hacer uso de una gran cantidad de nodos en las islas, los nodos que se encuentran más alejados de la estación base tendrán un

consumo mayor de corriente, llegando a afectar la duración de la batería de los nodos sensores, teniendo en cuenta que los nodos que tienen que realizar el reenvío de paquetes de datos de forma constante muestran un consumo alto de energía siendo uno de estos casos el nodo [1] que a pesar de estar cerca consume una cantidad de 1132 [mA].

Figura 108

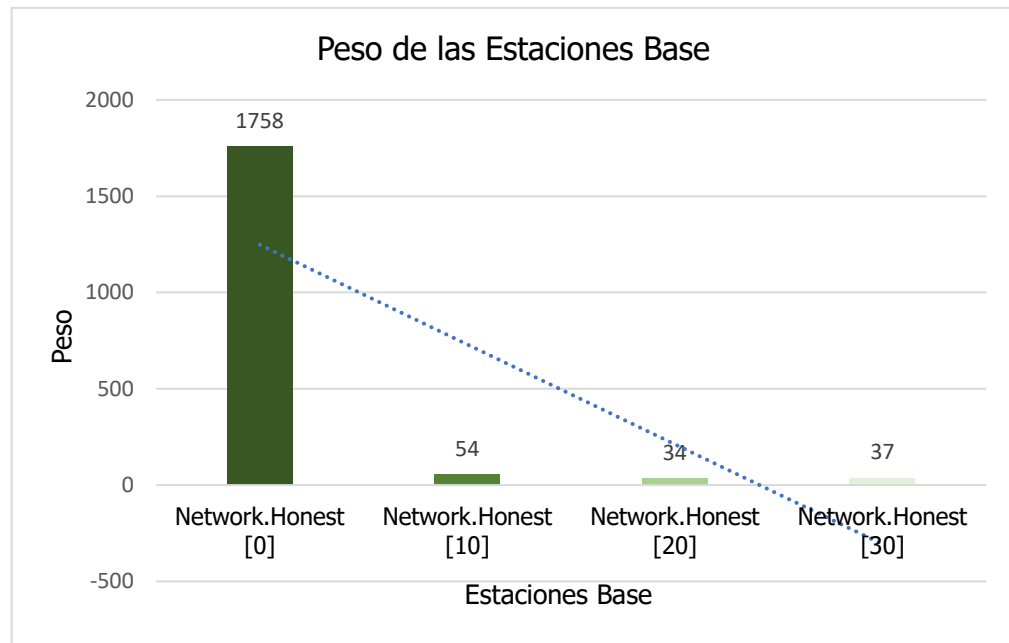
Corriente Consumida en los nodos



En el caso de los pesos, se puede observar en la Figura 109 que la estación base de la *Isla Tangle 3* mostrada en la Figura 107 que contiene la mayor cantidad de nodos presenta un peso propio mucho más alto que las estaciones base del resto de las islas. Esto se debe a que, al tener una mayor cantidad de nodos autenticados, tanto el peso de las estaciones base como el de los nodos cercanos a ella van a tener un peso mucho más alto que las otras estaciones base pertenecientes a las Islas Tangle 1,2,3. Esto es un gran beneficio para la seguridad de la red, ya que, a mayor peso, mayor seguridad de la red.

Figura 109

Comparación de pesos en las estaciones base



c) Desactivación de las Islas

Para asegurar el correcto funcionamiento del esquema de autenticación en la cadena de bloques, se ha desarrollado un código que permite condicionar el número de islas necesarias para su creación. La Figura 110 muestra un fragmento de código que se ejecuta durante la inicialización de la simulación mismo que comparan la variable `ID` con los valores de "[0]", "[10]", "[20]", "[30]" IDs que corresponden a las estaciones base de cada una de las islas donde si ambas condiciones se cumplen, la variable *pruebaIslaCaída* se establece en verdadero (true) permitiendo que una de las islas se encuentre caída.

Figura 109

Desactivación de las Islas

```

//***** Prueba de isla caída *****
if (ID == "[0]" && islaCaída1) pruebaIsLaCaída = true;
if (ID == "[10]" && islaCaída2) pruebaIsLaCaída = true;
if (ID == "[20]" && islaCaída3) pruebaIsLaCaída = true;
if (ID == "[30]" && islaCaída4) pruebaIsLaCaída = true;

if (topology == "ConAtaques" || topology == "SinAtaques" || topology == "UnaIsla") {

    scheduleAt(simTime() + exponential(rateMean), msgUpdatePass);
    scheduleAt(simTime() + 0.6, msgBLOQ11);
    scheduleAt(simTime() + exponential(rateMean) + 12, msgStartTx); //25
    if (!pruebaIsLaCaída) scheduleAt(simTime() + exponential(rateMean) + 15, msgCompeteBlock);

} else {

    scheduleAt(simTime() + exponential(rateMean), msgUpdatePass);
    scheduleAt(simTime() + exponential(rateMean) + 5.0, msgBLOQ11);
    scheduleAt(simTime() + exponential(rateMean) + 45.0, msgStartTx);
    if (!pruebaIsLaCaída) scheduleAt(simTime() + exponential(rateMean) + 50, msgCompeteBlock);

}

EV << "\xb1mProceso de inicializacion completado!\xb10m\n";

```

Controlamos que isla no funcionara

Si se cumple las condiciones indicadas en el (if) se actualiza las contraseñas, se realiza el bloqueo del canal de comunicación tambien se procede a inicializar la transmisión de mensajes.

El segundo bloque de código indicado en la Figura 110 se basa en la variable *topology*, que también es una cadena de caracteres, el cual donde si el valor de *topology* es igual a *ConAtaques*, *SinAtaques* o *UnaIsla*, se realiza la programación de cuatro eventos usando la función *scheduleAt ()* misma que se encarga de programar la ejecución de un mensaje en un momento específico de la simulación.

Es así que el primer evento se programa dependiendo de la topología de la red simulada y de si se ha detectado una isla caída, mismo que se determina en un tiempo aleatorio (*exponential(rateMean)*) más el tiempo real de simulación (*simTime()*), utilizando el mensaje *msgUpdatePass* para actualiza las contraseñas de los nodos vecinos.

El segundo evento tiene como función bloquear el canal de comunicación. Este evento se programa para ejecutarse después de un tiempo fijo de 0.6 unidades de tiempo.

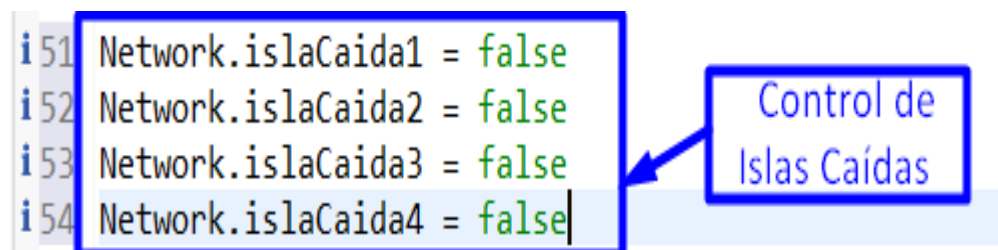
El tercer evento se genera cuando inicia la transmisión de un mensaje. Este evento se programa para ejecutarse en un momento específico calculado como

$simTime() + exponential(rateMean) + 12$ o $simTime() + exponential(rateMean) + 45.0$, mismo que determina el momento exacto en el que se programará el evento, dependiendo de la topología de la red simulada. La programación de estos eventos en momentos específicos de la simulación es necesaria para que se puedan simular las diferentes interacciones entre los nodos de la red

La activación y desactivación de las islas de nodos se lo realiza desde el archivo .ini mismo que muestra en cada uno de los escenarios la cantidad de islas con las que cuenta, como se muestra en la Figura 111 el parámetro *Network.Islas* es aquel que nos permite modificar la cantidad de islas con la que se podrá trabajar dentro de cada escenario, tomando en cuenta que el número mínimo de islas para que funcione la topología es dos, si se rebasa el límite mínimo automáticamente la simulación muestra el mensaje “*Número de islas caídas no permitido!*” indicado en la Figura 112, dentro de este mensaje también se indica el archivo en el cual se debe realizar el cambio para poder ejecutar normalmente la simulación.’

Figura 110

Control para desactivación de Islas



```

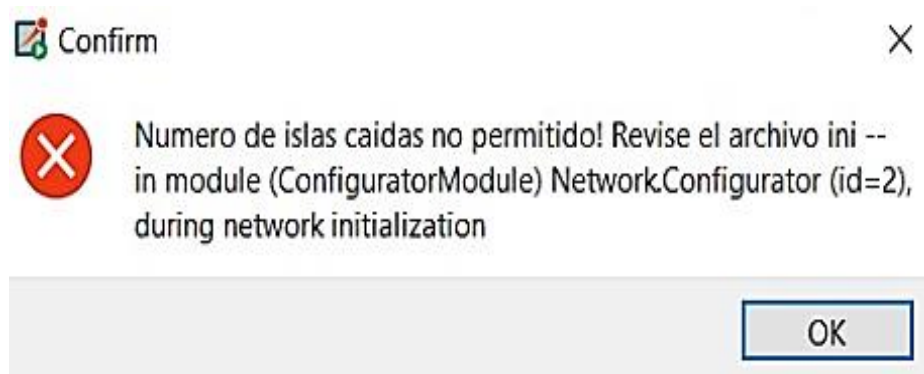
i 51 Network.islaCaida1 = false
i 52 Network.islaCaida2 = false
i 53 Network.islaCaida3 = false
i 54 Network.islaCaida4 = false

```

Control de Islas Caídas

Figura 111

Mensaje informativo de que se a excedido el mínimo de islas desactivadas



4.1.1.5. Prueba 5: Evaluación del comportamiento del esquema propuesto al unirse a una Cadena de Bloques Real

a) Código de creación de la Cadena de Bloques

En la Figura 113 el constructor de la clase *init* toma cinco argumentos que se utilizan para inicializar los atributos del objeto Block, incluyendo el índice del bloque (*index*), la identificación de la red (*red_id*), las transacciones incluidas en el bloque (*transacciones*), el tiempo de creación del bloque (*tiempo*) y el hash del bloque anterior (*hash_previo*). Además, se inicializa el atributo nonce en cero.

Figura 112

Creación del objeto *Block*

```

import json
import hashlib
import datetime
from urllib_
import flask
from flask import Flask, jsonify, request, render_template
import requests

class Block:
    def __init__(self, index, red_id, transacciones, tiempo, hash_previo):
        self.index = index
        self.red_id = red_id
        self.transacciones = transacciones
        self.tiempo = tiempo
        self.hash_previo = hash_previo
        self.nonce = 0

    def compute_hash(self):
        block_string = json.dumps(self.__dict__, sort_keys=True)
        return hashlib.sha256(block_string.encode()).hexdigest()

```

Importamos las librerías

Creamos el formato del Bloque

Realizamos el cálculo del hash

Además, en la figura anterior se observa que el método *compute_hash* de la clase *Block* toma los atributos del objeto *Block*, los convierte en una cadena JSON y luego aplica la función de hash SHA-256 de la biblioteca *hashlib* para calcular el hash del bloque. El hash se utiliza para verificar la integridad del bloque y su posición en la cadena de bloques. En resumen, este código implementa la funcionalidad básica de un bloque en una cadena de bloques y proporciona una forma de calcular su hash.

El código mostrado en la Figura 114 define una clase llamada *Blockchain* que representa una cadena de bloques. El atributo *dificultad* se establece en 4, lo que significa que cada bloque en la cadena debe tener un hash que comience con cuatro ceros. El método *init* es el constructor de la clase. Crea una lista vacía para transacciones sin confirmar, una lista vacía para la cadena de bloques, llama a la función *create_genesis_block()* para crear el primer bloque de la cadena (el bloque génesis) y crea un conjunto vacío de nodos.

Figura 113

Clase Blockchain

```

class Blockchain:
    dificultad = 4

    def __init__(self):
        self.transacciones_sin_confirmar = []
        self.chain = []
        self.create_genesis_block()
        self.nodos = set()

    def to_dict(self):
        chain_dict = []
        for block in self.chain:
            block_dict = {
                "index": block.index,
                "red_id": block.red_id,
                "tiempo": block.tiempo,
                "hash_previo": block.hash_previo,
                "hash": block.hash,
                "nonce": block.nonce,
                "transacciones": block.transacciones
            }
            chain_dict.append(block_dict)
        #return {"chain": chain_dict}
        return {"Longitud de la Cadena": len(self.chain),
                "cadena": chain_dict}

    def to_json(self):
        return json.dumps(self.to_dict(), indent=4)

```

El método *to_dict* crea un diccionario que representa la cadena de bloques. Recorre la lista de bloques de la cadena y crea un diccionario para cada bloque con los campos que se detalla en la Tabla 14:

Tabla 14

Campos del método *to_dict*

| CAMPO | DESCRIPCIÓN |
|--------------------|------------------------------------|
| <i>índice</i> | El número de bloque en la cadena |
| <i>red_id</i> | Un identificador único para la red |
| <i>tiempo</i> | La hora en que se creó el bloque |
| <i>hash_previo</i> | El hash del bloque anterior |

| | |
|----------------------|--|
| <i>hash</i> | El hash del bloque actual |
| <i>nonce</i> | Un número aleatorio utilizado en el proceso de prueba de trabajo para generar un hash que cumpla con la dificultad |
| <i>transacciones</i> | Una lista de transacciones incluidas en el bloque |

Finalmente, devuelve un diccionario que contiene la longitud de la cadena de bloques y una lista de diccionarios que representan cada bloque de la cadena.

A continuación, en la Tabla 15 se describe detalladamente las funciones específicas de cada uno de los métodos indicados en la Figura 115, con el fin de comprender mejor su papel en el funcionamiento de la cadena de bloques y en el desempeño de la clase Blockchain.

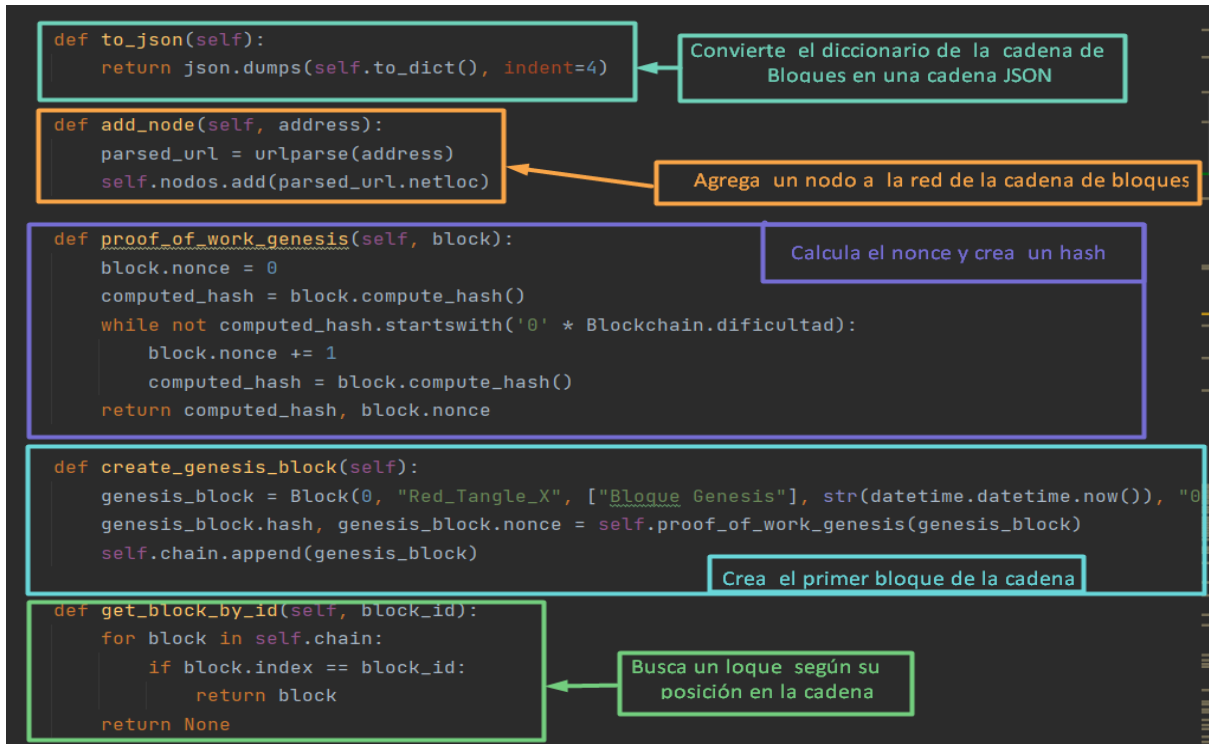
Tabla 15

Métodos necesarios para el funcionamiento de la Blockchain

| Método | Función | Descripción |
|------------------------------|-------------------|---|
| <i>to_json</i> | <i>json.dumps</i> | Convierte el diccionario de la cadena de bloques en una cadena JSON con formato legible para lo cual utiliza la función <i>json.dumps</i> de la biblioteca estándar de Python, que toma un objeto de Python y lo convierte en una cadena JSON |
| <i>add_node</i> | <i>urlparse</i> | Tiene como objetivo agregar las direcciones de los nodos conectados a la cadena de bloques. Estas direcciones se especifican en el formato <i>http://ipv4:puerto</i> . Este método toma una dirección de nodo como argumento y utiliza la función <i>urlparse</i> de la biblioteca estándar de Python para analizar la dirección y extraer el nombre de host (el nodo) y el número de puerto. Luego, agrega el nombre de host a la lista de nodos de la cadena de bloques |
| <i>proof_of_work_genesis</i> | | Se utiliza para calcular el nonce necesario para crear un hash válido para el bloque génesis |
| <i>create_genesis_block</i> | | Crea el primer bloque (el bloque génesis) de la cadena de bloques. Crea un objeto de bloque con valores para el índice específico, la <i>red_id</i> , la lista de transacciones, el tiempo y el hash anterior. Luego, llama al método <i>proof_of_work_genesis</i> para calcular el nonce y el hash para el bloque génesis y los asigna al bloque. |
| <i>get_block_by_id</i> | | Busca un bloque en la cadena de bloques por su índice (su posición en la cadena). Toma un número de índice como argumento y registra la lista de bloques de la cadena de bloques, este método se usa para la parte visual de la blockchain, permitiendo encontrar el bloque que necesitamos. |

Figura 114

Métodos que componen la clase Blockchain



La clase Blockchain también cuenta con otros métodos, como *last_block(self)*, el cual devuelve el último bloque de la cadena de bloques, que se almacena en una lista llamada *chain* y *self.chain[-1]*. Otro método que se presenta en la Figura 116 es *print_block(self, n)*, que recibe un número entero *n* como argumento y devuelve una cadena de texto que describe el *bloque en la posición n*⁸ de la cadena de bloques. La cadena de texto que se devuelve incluye el índice del bloque, la ID de la red, las transacciones que contiene, el tiempo en que se creó, el hash del bloque anterior, el hash del bloque y el nonce.

Por último, está el método *proof_of_work(self, block)*, el cual toma un objeto *block* y devuelve el hash del bloque después de encontrar un nonce que cumpla ciertos requisitos. El proceso para encontrar un nonce se basa en el uso de un bucle *while* para

⁸ Bloque en posición *n* : se accede mediante `self.chain[n]`

iterar sobre diferentes valores de nonce y calcular el hash del bloque con cada uno de ellos, hasta encontrar un nonce que produzca un hash válido. Los requisitos para que el hash del bloque sea válido se basan en la variable dificultad de la clase Blockchain.

Figura 115

Métodos de la clase Blockchain

The image shows a code editor with three Python methods from the Blockchain class, each with a descriptive annotation in a colored box:

- last_block(self):** A green box highlights the code `return self.chain[-1]`. An annotation box with a blue border and arrow points to it, containing the text "Devuelve el último bloque de la cadena".
- print_block(self, n):** A green box highlights the entire method code. An annotation box with a blue border points to it, containing the text "Describe la posición en la que se colocara la Tesis".
- proof_of_work(self, block):** A yellow box highlights the entire method code. An annotation box with a blue border points to it, containing the text "Proceso para encontrar el nonce de todos los bloques menos del génesis".

Para que la cadena de bloques pueda realizar diferentes actividades, como la verificación del hash o agregar una nueva transacción, entre otras, es necesario crear diferentes funciones, como se muestra en el código de la Figura 117. Entre ellas se encuentra la función *add_block*, que toma un objeto *block* y una prueba de trabajo *proof*, y comprueba si el hash anterior del bloque coincide con el hash del último bloque en la cadena. Si es así, se verifica que la prueba de trabajo sea válida y luego se agrega el bloque a la cadena de bloques, actualizando su hash. También se tiene la función *is_valid_proof*, que se encarga de comprobar si el hash del bloque y la prueba de trabajo dada satisfacen los requisitos para que el hash del bloque sea válido. Para agregar una

nueva transacción a la lista de transacciones no confirmadas, se utiliza la función *add_new_transaction*. Y para la creación de un nuevo bloque a partir de las transacciones no confirmadas, se usa la función *mine*, la cual incrementa en uno el índice con respecto al último bloque en la cadena. En base a esto, se realiza la prueba de trabajo en el nuevo bloque. Si la prueba de trabajo es válida, se agrega el nuevo bloque a la cadena de bloques y se devuelve el índice del nuevo bloque.

Figura 116

Definición de funciones de la cadena de Bloques

The image shows a code editor with four Python functions for a blockchain, each with a corresponding annotation box:

```
def add_block(self, block, proof):
    hash_previo = self.last_block.hash
    if hash_previo != block.hash_previo:
        return False
    if not self.is_valid_proof(block, proof):
        return False
    block.hash = proof
    self.chain.append(block)
    return True
```

Verificamos si el hash anterior coincide con el ultimo hash de la cadena

Valida el hash del Bloque

```
def is_valid_proof(self, block, block_hash):
    return (int(block_hash, 16) < 2 ** (256 - Blockchain.dificultad) and block_hash
```

```
def add_new_transaction(self, transaction):
    self.transacciones_sin_confirmar.append(transaction)
```

Agregamos una nueva transaccion

```
def mine(self, red_id):
    if not self.transacciones_sin_confirmar:
        return False
    last_block = self.last_block
    new_block = Block(
        index=last_block.index + 1,
        red_id=red_id,
        transacciones=self.transacciones_sin_confirmar,
        tiempo=str(datetime.datetime.now()),
        hash_previo=last_block.hash
    )
    proof = self.proof_of_work(new_block)
    self.add_block(new_block, proof)
    self.transacciones_sin_confirmar = []
```

Creamos un nuevo bloque a partir de las transacciones que no han sido confirmadas

A diferencia de la cadena de bloques que se adaptó para el desarrollo de la simulación, la cadena de bloques real realiza un proceso de consenso para la determinación de la cadena más larga, lo cual lo realiza a través de la función *def consenso(self)* cuyo objetivo es verificar si la cadena de bloques actual de una estación base es la más larga y actualizada en comparación con las cadenas de otras estaciones base en la red. Si se encuentra una cadena más larga y válida, la cadena actual se reemplaza por la cadena más larga.

En el código mostrado en la Figura 118, la función primero comprueba si la estación base actual tiene vecinos conectados a la red. Si no hay vecinos, devuelve Falso y se mantendrá solo la cadena perteneciente a la estación base. Si hay estaciones base vecinas, la función itera sobre cada una de ellas para solicitar su cadena de bloques utilizando la *API/chain2*⁹. Si se recibe una respuesta exitosa (código de estado HTTP 200), se constata si la longitud de la cadena recibida es mayor que la longitud de la cadena actual del nodo. Si se encuentra una cadena más larga y válida, se actualiza la variable *nueva_cadena* con la nueva cadena encontrada y la longitud máxima de la cadena actual se actualiza con la longitud de la cadena encontrada.

⁹ **API /chain2:** Ruta de la aplicación basada en blockchain que permite a las estaciones base de la red solicitar y recibir la cadena de bloques de otra estación base.

Figura 117

Función de consenso

```

def consenso(self):
    if len(blockchain.nodos) <= 1:
        return False
    else:
        vecinos = self.nodos
        nueva_cadena = None

        # Busca una cadena más larga que la nuestra
        longitud_maxima = len(self.chain)
        print(longitud_maxima)
        for nodo in vecinos:
            try:
                response = requests.get(f'http://{nodo}/chain2')
            except:
                print(f'Nodo {nodo} está desconectado')
                continue
            if response.status_code == 200:
                longitud = response.json()['longitud de la Cadena']
                cadena = response.json()['cadena']
                print(longitud)
                print(cadena)
                print(blockchain.is_valid_chain(cadena))
                # Comprueba si la longitud es mayor y la cadena es válida
                if longitud > longitud_maxima and blockchain.is_valid_chain(cadena):
                    longitud_maxima = longitud
                    nueva_cadena = cadena
                    print("Si es valida")

```

Verificamos la existencia de Estaciones Base en la red

Si existe Estaciones Base vecinas o una cadena más larga se almacena en estas variables

Se inicializa con la longitud de la cadena actual

Realizamos una solicitud HTTP GET para obtener la cadena de bloques de la estación base

Se actualiza con la cadena más larga y válida

Si se cumple con la condición se imprime el mensaje *Si es valida*

El código mostrado en la *Figura 119* utiliza un bucle *while* para monitorear constantemente un archivo CSV y enviar solicitudes POST a un servidor web para cada fila de datos nuevos en el archivo. El código también envía una solicitud GET para que el servidor mío los datos después de que se hayan enviado todas las solicitudes POST. El código comienza obteniendo la marca de tiempo actual del archivo *filename1* utilizando *os.path.getmtime*. Luego, el código compara esta marca de tiempo con la marca de tiempo anterior *last_modified1*. Si la marca de tiempo es diferente, esto significa que el archivo ha sido modificado y el código procede a leer los nuevos datos del archivo CSV utilizando una *with instrucción* para abrir el archivo y un *csv.reader* para leer los datos.

Figura 118

Código para la administración de transacciones

```

while True:
    Comprueba si la marca de tiempo de los archivos a cambiad
    current_modified1 = os.path.getmtime(filename1)
    if current_modified1 != last_modified1:
        print("El archivo1 ha sido modificado")
        El archivo 1 ha sido modificado, así que leemos los nuevos datc
        with open(filename1, newline='') as csvfile:
            reader = csv.reader(csvfile, delimiter=';', quotechar='')
            next(reader)
            Omitimos la primera fila que contiene los encabezados
            for row in reader:
                print(row)
                data = {"Nodo": row[0], "sensorPH": row[1],
                        "sensorTEMP": row[2]}
                Usamos los valores de la fila actual
                headers = {"Content-type": "application/json"}
                response = requests.post("http://127.0.0.1:5000/new_transaction", json=data, headers=headers)
                print(response.status_code)
                Esperamos 1 segundo antes de enviar la siguiente solicitud POST
            last_modified1 = current_modified1
            Actualizamos la marca de tiempo del archivo 1
            Enviar una solicitud GET despues de enviar todas la solicitudes POST
            response = requests.get("http://127.0.0.1:5000/mine")
            print(response.status_code)
            print("El archivo1 ha sido minado")
            time.sleep(1)
            Esperar 10 segundos antes de enviar la siguiente solicitud GET

```

El código indicado en la Figura 119 también realiza una iteración a través de cada fila en el archivo *CSV*, creando un diccionario *data* para cada fila que contiene los valores de la fila actual. Luego, el código usa el módulo *requests* para enviar una solicitud *POST* al servidor web con el diccionario *data* como carga útil y los encabezados especificados en *headers*.

Después de enviar todas las solicitudes *POST*, el código envía una solicitud *GET* al servidor web para que mine los datos. Luego, el código espera 1 segundo antes de continuar el bucle *while True*.

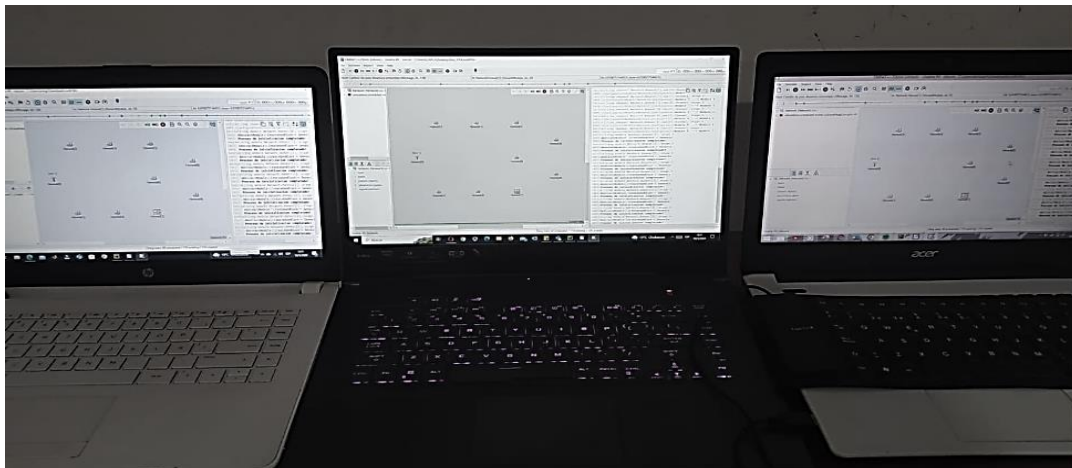
En resumen, este código se utiliza para monitorear un archivo *CSV* en busca de nuevas filas de datos y enviar las filas de datos nuevas a un servidor web utilizando solicitudes *POST*. También se utiliza para enviar una solicitud *GET* al servidor web para que mine los datos después de que se hayan enviado todas las solicitudes *POST*.

b) Conexión entre la Simulación de Omnet++ y la Blockchain Real

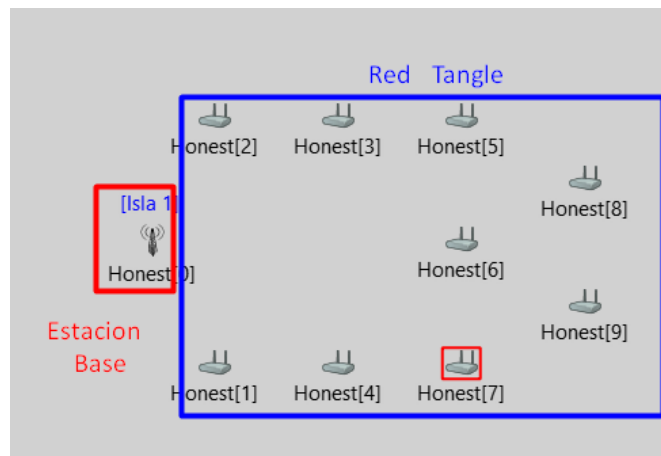
Para la simulación de las redes de sensores inalámbricos, se procede a simular dentro de cada una de las PC el escenario formado por una sola isla como se muestra en la Figura 120. De esta forma, cada una de las PC recibirá un archivo CSV de cada una de las estaciones base de la simulación que están en ejecución. Este archivo se utilizará para la creación del bloque candidato dentro de cada PC. Luego, se realizará una prueba de trabajo y se publicará en la cadena de bloques principal.

Figura 119

Simulación de Red Tangle en computadoras



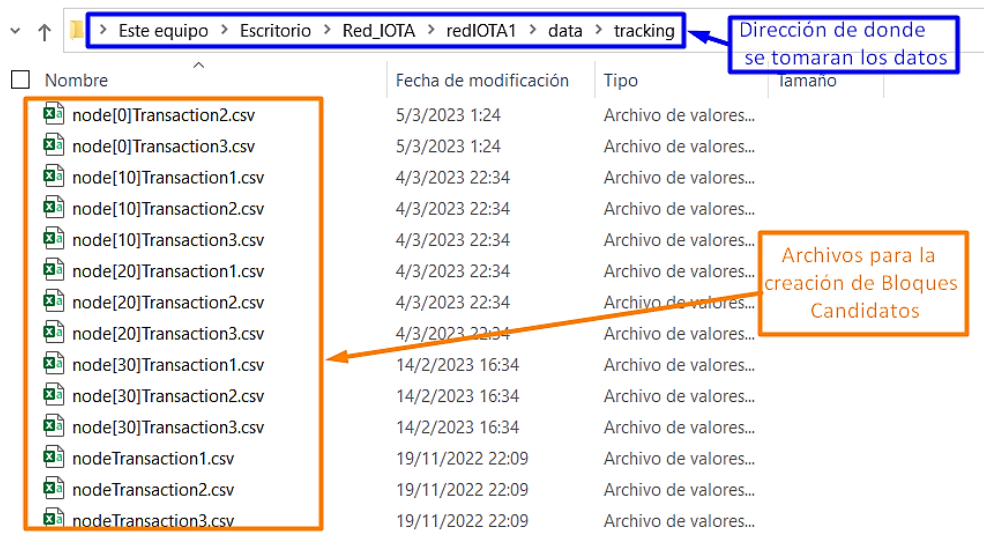
Para la conexión de la cadena de bloques real y la simulación de Omnet++, se ha creado un escenario con una sola isla, como se muestra en la Figura 121. La misma está compuesta por 9 nodos sensores y una estación base. Es importante destacar que la estación base de la simulación será la encargada de realizar el envío de las transacciones a la cadena de bloques reales.

Figura 120*Escenario Red Tangle 1 Isla*

Cuando la estación base recibe una transacción generada por los nodos sensores, se generan varios archivos .csv como se muestra en la Figura 122 que almacenan los datos de PH, humedad y temperatura recolectados por los sensores, cambiando cada vez que se ejecuta la simulación. La importancia de estos archivos radica en que la cadena de bloques reales los utiliza para recolectar información de las transacciones generadas por los nodos sensores por lo cual debe acceder a la ruta donde se encuentran guardados. Estos archivos se usarán para la creación del bloque candidato en cada una de las estaciones base.

Figura 121

Archivos .csv. que almacenan las transacciones



Para importar los archivos .csv al código Python responsable de generar la cadena de bloques, se utilizan varias librerías, incluyendo *csv*, *os* y *request*, que permiten obtener la última versión de los archivos y enviarlos al servidor. Para la obtención de estos archivos es de gran importancia indicar la dirección exacta de la simulación que contiene los archivos .csv, seleccionando solo los correspondientes a una estación base específica como puede ser la [0],[10][20], tal como se muestra en la Figura 123. A continuación, dentro del código se especifica la dirección IP de la PC que se ejecutará la simulación misma que será la encargada de realizar las solicitudes de minado y nuevas transacciones.

Figura 122

Transformación de datos .csv a formato Python

```

import csv
import os
import time
import requests

filename1 = 'C:/Users/CARLA/Desktop/Red_IOTA/redIOTA1/data/tracking/node[0]Transaction1.csv'
filename2 = 'C:/Users/CARLA/Desktop/Red_IOTA/redIOTA1/data/tracking/node[0]Transaction2.csv'
filename3 = 'C:/Users/CARLA/Desktop/Red_IOTA/redIOTA1/data/tracking/node[0]Transaction3.csv'

transacciones = "http://192.168.0.104:5000/new_transaction"
minado = "http://192.168.0.104:5000/mine"

# Obtener la marca de tiempo inicial de los archivos
last_modified1 = os.path.getmtime(filename1)
last_modified2 = os.path.getmtime(filename2)
last_modified3 = os.path.getmtime(filename3)

```

Annotations in the image:

- Se lee los archivos .csv** (yellow box) points to `import csv`.
- Obtener la última actualización del archivo .csv** (purple box) points to `import os` and `import time`.
- Enviamos las transacciones al servidor** (green box) points to `import requests`.
- Ubicación de los archivos de la Isla Tangle 1** (green box) points to the filename variables.
- Dirección de envío de peticiones** (orange box) points to the `transacciones` variable.
- Guardamos la fecha del archivo y esperamos actualizaciones** (red box) points to the `last_modified` variables.

El código mostrado en la Figura 124 utiliza un método **GET** para realizar la recuperación de información de la función `mine_unconfirmed_transactions()` misma que define en su interior una variable `ID_redtangle` que se utiliza para identificar la IP de la estación base que está minando. A continuación, se llama a la función `mine` del objeto `blockchain` para intentar minar el siguiente bloque de transacciones no confirmadas en la cadena de bloques.

Si no hay transacciones pendientes para ser minadas, la función devuelve el mensaje *Nada que minar*. Si se ha minado un bloque, se llama a la función `consenso` en todos los nodos vecinos utilizando la biblioteca `requests`.

Figura 123

Recuperación de información de la función `mine_unconfirmed_transactions()`

```

@app.route('/mine', methods=['GET'])
def mine_unconfirmed_transactions():
    ID_redtangle = 'Red Tangle A -- 192.168.0.104 - ISLA 1'
    result=blockchain.mine(ID_redtangle)
    vecinos=blockchain.nodos
    if not result:
        return "Nada que minar"
    for nodo in vecinos:
        response = requests.get(f'http://{nodo}/consenso')
        if response.status_code == 200:
            print("Se ha llamado al Consenso de todos los Nodos")
    return "El bloque #{} es minado y se ha llamado al Consenso de los demás nodos".format(result)

```

Annotations in the image:

- Red box: `ID_redtangle = 'Red Tangle A -- 192.168.0.104 - ISLA 1'`
- Green box: `for nodo in vecinos:` block containing the `requests.get` call and the `print` statement.
- Text box 1: "Para realizar el minado se requiere especificar el ID de la red" (with an arrow pointing to the ID variable).
- Text box 2: "Buscamos en las estaciones base y les indicamos que deben realizar el consenso" (with an arrow pointing to the `for` loop).
- Comment: `## cambiar dependiendo del nodo...`

La función *consenso* que se indica en la Figura anterior es una función definida en la clase *blockchain* que se utiliza para verificar que la cadena de bloques en todos los nodos sea la misma. Si la llamada a la función *consenso*¹⁰ es exitosa, la función imprime un mensaje en la consola indicando que se ha llamado a la función de consenso de todas las estaciones base. Finalmente, se devuelve un mensaje que indica que se ha minado el bloque y se ha llamado a la función de consenso de las estaciones base vecinas.

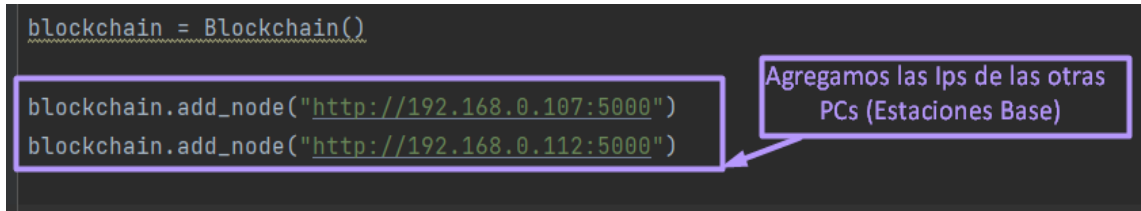
Para poder realizar la conexión con las otras islas Tangle se procede a agregar las direcciones IP de las estaciones base como se muestra en la Figura 125, para lo cual se hace uso de la función *add_node()* que toma como parámetro una cadena de texto que representa la dirección URL de una estación base de la red. Una vez que se agreguen las estaciones base a la red, la estación base original (la que ejecuta este código) podrá comunicarse con todas las estaciones base y compartir información sobre la cadena de bloques. Esto es importante para asegurar que todas las copias de la cadena

¹⁰ **Consenso:** Proceso mediante el cual se logra un acuerdo entre todos los nodos de la red sobre el estado actual de la cadena de bloques.

de bloques en la red sean idénticas y para permitir el proceso de consenso entre los nodos de la red.

Figura 124

Incorporación de IPs de las diferentes islas a el código Python



```

blockchain = Blockchain()
blockchain.add_node("http://192.168.0.107:5000")
blockchain.add_node("http://192.168.0.112:5000")

```

Agregamos las Ips de las otras PCs (Estaciones Base)

En este caso, la Figura 126 muestra la aplicación web Flask¹¹ que está escuchando al host 192.168.0.104 y en el puerto 5000. El host es la dirección IP de la máquina en la que se está ejecutando la aplicación web, y el puerto es el número de puerto en el que la aplicación web escucha las solicitudes entrantes. Una vez que se ejecuta esta línea de código, la aplicación web estará en ejecución y se podrá acceder a ella en la dirección URL " http://192.168.0.104:5000 ". Esto permitirá a los usuarios acceder a la aplicación web y utilizar sus funcionalidades, como visualizar información de la cadena de bloques, realizar transacciones y agregar nodos a la red.

Figura 125

Dirección IP y puerto de la página web que mostrara cada uno de los bloques de la Cadena



```

app.run(host='192.168.0.104', port=5000)

```

Dirección donde se ejecutara la Blockchain

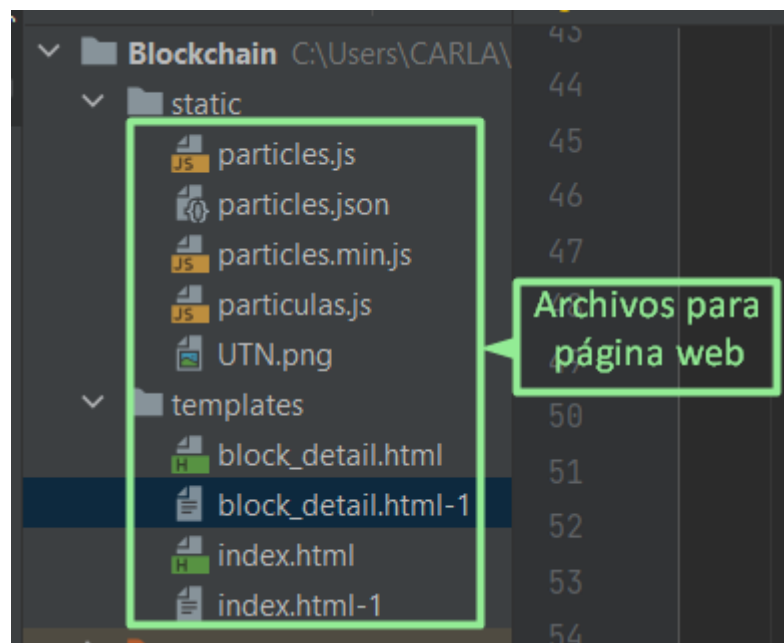
Para poder visualizar la creación de la cadena de bloques dentro de la página web, es necesario agregar diferentes archivos en Python, tales como *particles.js*,

¹¹ **Flask:** Es un framework de aplicaciones web de Python que permite crear aplicaciones web de forma sencilla y rápida.

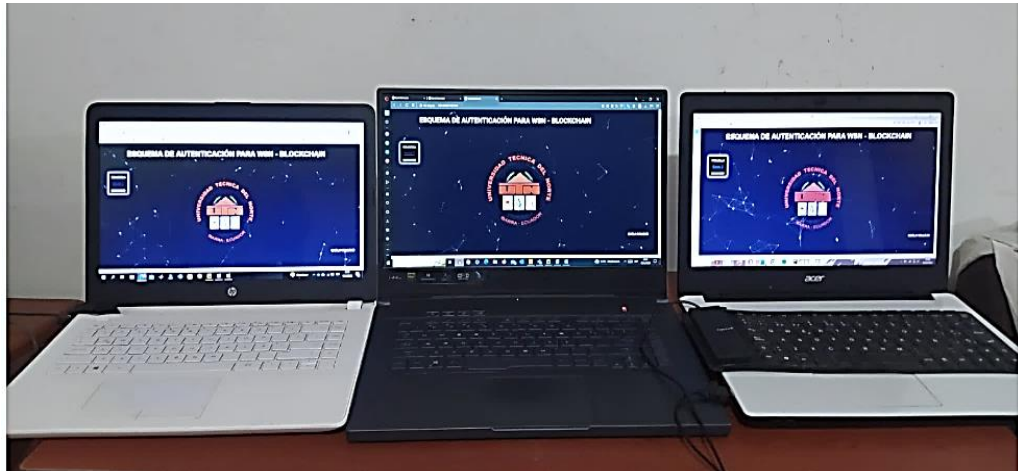
particles.json y *UTN.png* mismos que se indican en la Figura 127 . Estos archivos permiten crear una página web más amigable. Además, es necesario incluir el *index.html* para crear la página principal y el *block_detail.html* para agregar la información que contiene cada uno de los bloques de la cadena.

Figura 126

Archivos para la visualización y diseño de página web



Cuando ponemos en el navegador la dirección IP estática fijada a cada una de las PC y especificamos el puerto 5000, se podrá ingresar a la página web de la Cadena de Bloques como se indica en la Figura 128. En esta página se podrá ver cómo se suben cada uno de los bloques generados por las diferentes islas. Todas las PC se muestran en su página web el bloque génesis, que es el mismo para todas las islas al inicio del minado.

Figura 127*Página web de la Cadena de Bloques*

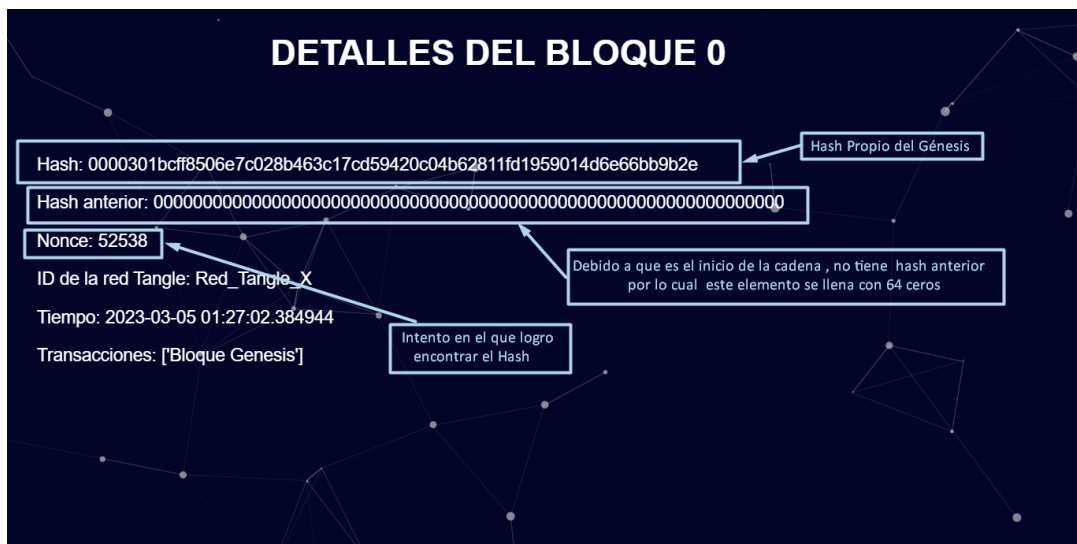
Al ejecutar el código que se muestra en las Figuras 112 a 118, se podrá visualizar el Bloque Génesis de la red al ingresar a la URL de la página web creada como se observa en la Figura 129. Cabe destacar que cada estación base tiene su propio bloque Génesis, por lo que será diferente en cada una de ellas. Sin embargo, al agregar un nuevo bloque a la cadena, el Bloque Génesis cambiará y se volverá el mismo en todas las estaciones base, ya que será compartido como una copia de la cadena de bloques principal.

Figura 128*Bloque Génesis*

El Bloque Genesis es el primer bloque de una cadena de bloques como se indica en la Figura 130, por lo tanto, no tiene ningún bloque anterior al que se pueda hacer referencia en su hash. El hash de un bloque se genera a partir de la información contenida en el bloque anterior y en la información del bloque actual, por lo que el Bloque Genesis no puede tener un hash anterior por lo cual en su lugar se llena este campo por una gran cantidad de 0, ya que no hay ningún bloque anterior del que pueda derivarse. En su lugar, el hash propio del Bloque Genesis se genera únicamente a partir de la información que contiene, como el nonce y los datos del bloque.

Figura 129

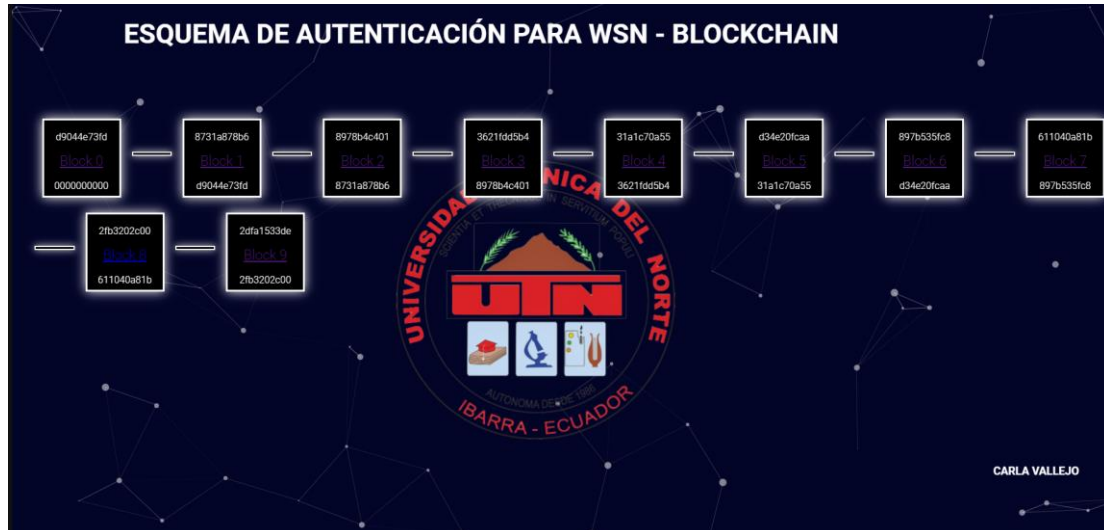
Elementos del Bloque Génesis



Cuando todas las estaciones base han receptado las diferentes transacciones que generan los nodos sensores, y se haya realizado la prueba de trabajo de forma rápida se procederá a agregar cada uno de los bloques a la cadena principal como se indica en la Figura 131, donde cada uno de los bloques cuenta con ID o Nombre para su reconocimiento, y parte de los hash anterior y propio que le componen.

Figura 130

Aumento de Bloques en la Cadena Principal



El proceso que se muestra en la Figura 132 se realiza a través de un proceso de consenso el cual se encarga de la validación de los nuevos bloques que se agregan a la cadena de bloques principal. Para agregar un nuevo bloque, los nodos de la red deben validar que las transacciones incluidas en el bloque son válidas y que el bloque se ajusta a las reglas y protocolos del sistema. Si la mayoría de los nodos están de acuerdo en que el nuevo bloque es válido, se agrega a la cadena de bloques y se propaga a la red, este proceso se lo puede observar dentro del terminal de Python en el cual se indica la dirección IP 192.168.0.112 llama al consenso y la IP 192.168.0.107 indica el estado de la cadena en ese momento, obteniendo como resultado lo que se indica en la Figura 132.

Figura 132

Detalle de los Bloques que se agregan a la cadena principal



4.2. DISCUSIÓN

A diferencia otros esquemas de autenticación que utilizan la cadena de bloques para el almacenamiento de datos obtenidos por sensores como en el caso del trabajo “Un sistema híbrido de autenticación de identidad basado en cadenas de bloques”, en el cual se maneja diferentes etapas para el registro de la información en el cual se determina dos tipos de autenticación intradominio¹² e interdominio¹³, el esquema de autenticación propuesto crea un entorno seguro a través de la implementación del protocolo SEECR y el uso de diferentes contraseñas más salt, permitiendo así que los nodos pasen por un proceso de autenticación antes de unirse a la red. De esta manera, se dificulta la infiltración de nodos maliciosos en la comunicación debido a la complejidad que presenta la resolución de la prueba de trabajo que deben superar para

¹² **Intradominio:** Se refiere al proceso de verificar la identidad de un usuario dentro del mismo dominio de seguridad.

¹³ **Interdominio:** Proceso de verificar la identidad de un usuario que pertenece a un dominio de seguridad diferente al recurso que se desea acceder

formar parte de la red, así como el número limitado de intentos disponibles para descubrir el hash correcto.

El esquema propuesto realiza un proceso de autenticación utilizando una Cadena de Bloques personalizada en la cual todos los bloques se publican según la velocidad en la que las estaciones base resuelvan la prueba de trabajo para de esta forma se logren publicar cada uno de los bloques de información, en comparación con el trabajo de Y. Chen et al.(2022) en el cual el nuevo bloque que sea creado se añadirá al final del Tangle mediante el reconocimiento del formato y la fuente por parte de los participantes de la red y después de varias rondas de consenso entre los nodos se demostrara la credibilidad del bloque .

Después de analizar algunas de las características de los esquemas de autenticación citados, se ha propuesto dentro del nuevo esquema de autenticación planteado el manejo de dos pruebas de trabajo la primera es realizada específicamente por los nodos sensores para autenticarse y la segunda es ejecutada por las estaciones base para la publicación de su bloque candidato en la cadena de bloques central permitiendo de esta forma dar mayor seguridad a toda la red.

5. CONCLUSIONES

Al culminar con el trabajo de investigación planteado y realizar un análisis exhaustivo, así como también el estudio de la problemática planteada, se puede afirmar que se han logrado alcanzar los objetivos propuestos y se han obtenido resultados significativos en relación con el tema abordado mismos que se indican a continuación:

- Al revisar el contexto de diferentes investigaciones sobre las redes de sensores inalámbricos, las tecnologías basadas en cadena de bloques y Tangle, se afirma que estas nuevas tecnologías son viables pueden usarse para mejorar la seguridad y eficiencia de las redes WSN debido a sus características de descentralización y eficacia en la gestión de datos y transacciones.
- El esquema de autenticación de nodos inalámbricos basado en el funcionamiento de Tangle y cadenas de bloques ha demostrado ser una solución efectiva para mejorar la seguridad y la integridad de las redes inalámbricas. Al aplicar pruebas de trabajo, aumento de peso en los nodos y estaciones base, así como la aplicación del protocolo SEECR, se puede proporcionar un mayor nivel de confianza en la autenticidad y seguridad de las redes inalámbricas sin aumentar significativamente el consumo de energía y recursos, representando una mejora en la eficiencia de la autenticación de nodos inalámbricos.
- La creación de un escenario simulado de ataques es un paso fundamental en el proceso de implementación del esquema de autenticación diseñado, demostrando el funcionamiento y comportamiento del esquema frente a ataques de diccionario y denegación de servicio. Los resultados obtenidos fueron positivos debido al reconocimiento y aislamiento de nodos maliciosos que lleva a cabo el esquema de autenticación a través de la aplicación del protocolo SEECR, el cual a través del número de intentos de autenticación limitado que maneja y al nivel de

complejidad requerido para revertir un hash, resulta difícil la penetración de nodos maliciosos en la red.

- Al realizar la prueba de aplicación del patrón de las "3A" del Unit Testing mediante la herramienta opp_test, se pudo verificar la creación del bloque candidato en cada estación base, así como la inicialización, uso y desempeño de los nodos maliciosos y honestos en la red, obteniendo así resultados precisos en la verificación del establecimiento de conexiones dentro de la Red Tangle.

- La creación de un escenario que consta de cuatro islas permitió observar que el proceso de inicialización de cada una de las islas lleva más tiempo, esto se debe a que, al contar con más islas, existe una mayor cantidad de dispositivos, lo que prolonga los procesos de autenticación, envío de transacciones, creación de bloque candidato y establecimiento de la cadena de bloques. En comparación, con una red formada por tres islas la cual realiza todos los procesos mucho más rápido, permitiendo así entender que el número de islas en el escenario tiene un impacto significativo en el tiempo de inicialización y en los procesos de la red. Es importante considerar este factor al implementar el esquema de autenticación propuesto en redes de sensores inalámbricos de gran escala.

- La tecnología de cadena de bloques real, a diferencia de la configurada para la simulación, requiere que todas las estaciones base lleguen a un consenso para validar el bloque candidato que se unirá a la cadena central. Además, se utiliza para verificar la cadena más grande que puede formarse a través de la publicación de dos bloques candidatos al mismo tiempo.

6. RECOMENDACIONES

En base a el funcionamiento del esquema de autenticación se han establecido diferentes recomendaciones que ayudaran al correcto funcionamiento de este, así como

también permitirá evidenciar el aporte que proporciona a la seguridad de las redes de sensores WSN.

- Para evitar ser una víctima de un ataque de diccionario y aportar a la seguridad del esquema es recomendable cargar dentro de los nodos sensores un diccionario personalizado cuyas claves sean únicas y no se encuentren dentro de ninguna base de datos de claves.

- Un aspecto importante dentro de las redes Tangle es la cantidad de nodos con la cual funcionan por lo cual para tener un alto nivel de seguridad en los nodos sensores y en la estación base es importante tener una gran cantidad de nodos en las islas ya que de esta forma el peso propio de los nodos y de la estación base aumentara creando actualizaciones mensajes de actualización de contraseñas de forma más frecuente evitando de esta forma tener claves estáticas dentro de cada uno de los elementos que conforman la red Tangle.

- Debido a que la estación base de cada una de las islas es la encargada de generar el bloque candidato y guardar una copia de la cadena de bloques principal es de gran importancia que sus características a nivel de procesamiento, memoria y energía sean robustos para que de esta forma el proceso de la creación de la cadena de bloques no se vea afectada volviéndose más lenta.

- El esquema de autenticación propuesto solo podrá crear la cadena de Bloques si se tiene un mínimo de dos islas, en caso de no cumplir esta condición, solo se podría minar el bloque génesis propio, por lo cual el bloque candidato que genere la estación base solo puede agregarse a su propio bloque génesis, este fenómeno se da debido a que no existiría una cadena de bloques central.

- Se recomienda que en un Trabajo futuro se realice una mejora en el proceso de autenticación a través del uso de base de datos de contraseñas o de

la creación de contraseñas de forma automática a base de una semilla permitiendo de esta forma reforzar la seguridad de la red y reducir la probabilidad de descubrir las contraseñas.

BIBLIOGRAFÍA

- 3GPP. (2018). Universal Mobile Telecommunications System (umts); Spatial Channel Model for Multiple Input Multiple Output (mimo) Simulations (3gpp Tr 25.996 Version 15.0.0 Release 11) Intellectual Property Rights. [https://www.semanticscholar.org/paper/Universal-Mobile-Telecommunications-System-\(umts\)%3B/003eed0b33285b5e3afa62a381e485093e34d09](https://www.semanticscholar.org/paper/Universal-Mobile-Telecommunications-System-(umts)%3B/003eed0b33285b5e3afa62a381e485093e34d09)
- Adenowo, A., & Adenowo, B. (2020). Software Engineering Methodologies: A Review of the Waterfall Model and Object- Oriented Approach. *International Journal of Scientific and Engineering Research*, 4, 427-434.
- Alazzawi, L. K., Elkateeb, A. M., Ramesh, A., & Aljuhar, W. (2008). Scalability Analysis for Wireless Sensor Networks Routing Protocols. *22nd International Conference on Advanced Information Networking and Applications - Workshops (aina workshops 2008)*, 139-144. <https://doi.org/10.1109/WAINA.2008.178>
- Alsboui, T., Qin, Y., Hill, R., & Al-Aqrabi, H. (2020). Towards a Scalable IOTA Tangle-Based Distributed Intelligence Approach for the Internet of Things. En K. Arai, S. Kapoor, & R. Bhatia (Eds.), *Intelligent Computing* (Vol. 1229, pp. 487-501). Springer International Publishing. https://doi.org/10.1007/978-3-030-52246-9_35
- Andreu, S. V. (2018). GRADO EN INGENIERÍA DE TECNOLOGÍAS Y SERVICIOS DE TELECOMUNICACIÓN.
- Avila-Campos, P., Astudillo-Salinas, F., Vazquez-Rodas, A., & Araujo, A. (2019). Evaluation of LoRaWAN Transmission Range for Wireless Sensor Networks in Riparian Forests. *Proceedings of the 22nd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems - MSWIM '19*, 199-206. <https://doi.org/10.1145/3345768.3355934>

- Bachmann, S. (2019). Analysis of the Tangle in the IoT Domain [Theses: Master Basic Module (MBM), Universität Zürich].
https://files.ifi.uzh.ch/CSG/staff/Rafati/Simon_MBM_IOTA.pdf
- Balaji, S. (2012). WATEERFALLVs V-MODEL Vs AGILE: A COMPARATIVE STUDY ON SDLC. . . Vol., 1, 5.
- Boonkrong, S., & Somboonpattanakit, C. (2016). Dynamic Salt Generation and Placement for Secure Password Storing. *IAENG International Journal of Computer Science*, 43, 27-36.
- Brooks, S. (1998). Markov chain Monte Carlo method and its application. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 47(1), 69-100.
<https://doi.org/10.1111/1467-9884.00117>
- Burgos Yar, V. V. (2021). Cifrado de datos usando Cadena de Bloques (BlockChain) como tecnología de convergencia para dispositivos móviles asociados con IoT (Internet of Things), en la capa de aplicación del modelo de capas IoT.
<http://repositorio.utn.edu.ec/handle/123456789/11459>
- BusinessTech. (2022, 04). How long it takes hackers to crack your password based on how many characters it has. <https://businesstech.co.za/news/it-services/572976/how-long-it-takes-hackers-to-crack-your-password-based-on-how-many-characters-it-has/>
- Carelli, A., Palmieri, A., Vilei, A., Castanier, F., & Vesco, A. (2022). Enabling Secure Data Exchange through the IOTA Tangle for IoT Constrained Devices. *Sensors*, 22(4), Article 4. <https://doi.org/10.3390/s22041384>
- Chan, H., Perrig, A., & Song, D. (2003). Random key predistribution schemes for sensor networks. 2003 Symposium on Security and Privacy, 2003., 197-213.
<https://doi.org/10.1109/SECPRI.2003.1199337>

- Chen, X., Makki, K., Yen, K., & Pissinou, N. (2009). Sensor network security: A survey. *IEEE Communications Surveys Tutorials*, 11(2), 52-73. <https://doi.org/10.1109/SURV.2009.090205>
- Chen, Y., Yang, X., Li, T., Ren, Y., & Long, Y. (2022). A blockchain-empowered authentication scheme for worm detection in wireless sensor network. *Digital Communications and Networks*. <https://doi.org/10.1016/j.dcan.2022.04.007>
- Cotugno, L., Franci, S., Galli, G., Vizzarri, A., & Perrone, G. (2020). Blockchain and IoTA Systems for Automotive sector: A Comparative Analysis. 7.
- Cui, Z., XUE, F., Zhang, S., Cai, X., Cao, Y., Zhang, W., & Chen, J. (2020). A Hybrid BlockChain-Based Identity Authentication Scheme for Multi-WSN. *IEEE Transactions on Services Computing*, 13(2), 241-251. <https://doi.org/10.1109/TSC.2020.2964537>
- Di Pierro, M. (2017). What Is the Blockchain? *Computing in Science Engineering*, 19(5), 92-95. <https://doi.org/10.1109/MCSE.2017.3421554>
- Diallo, E., Dib, O., & Al Agha, K. (2022). A scalable blockchain-based scheme for traffic-related data sharing in VANETs. *Blockchain: Research and Applications*, 100087. <https://doi.org/10.1016/j.bcra.2022.100087>
- Elshrkawey, M., Elsherif, S. M., & Elsayed Wahed, M. (2018). An Enhancement Approach for Reducing the Energy Consumption in Wireless Sensor Networks. *Journal of King Saud University - Computer and Information Sciences*, 30(2), 259-267. <https://doi.org/10.1016/j.jksuci.2017.04.002>
- Eterovic, J., Uran Acevedo, J., Rusticcini, A., & Gigante, N. (2021). Desarrollo de una DApp académica en la red Blockchain Federal Argentina. XXIII Workshop de Investigadores en Ciencias de la Computación (WICC 2021, Chilecito, La Rioja). <http://sedici.unlp.edu.ar/handle/10915/120517>

- Fan, C., Ghaemi, S., Khazaei, H., Chen, Y., & Musilek, P. (2021). Performance Analysis of the IOTA DAG-Based Distributed Ledger. *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, 6(3), 1-20. <https://doi.org/10.1145/3485188>
- Fundación IOTA. (2022). *IOTA_for_Business.pdf*. 28.
- Garcia, E., Serna, M., Bermúdez, A., & Casado, R. (2009). Simulación de un sistema de apoyo en la extinción de incendios forestales basado en WSN.
- Guegan, D. (2017). Public Blockchain versus Private blockchain. 8.
- Guo, H., & Yu, X. (2022). A Survey on Blockchain Technology and its security. *Blockchain: Research and Applications*, 100067. <https://doi.org/10.1016/j.bcra.2022.100067>
- Hassan, G., Subramaniam, S., Zukarnain, Z., & Samian, N. (2022). Fault Tolerance Structures in Wireless Sensor Networks (WSNs): Survey, Classification, and Future Directions. *Sensors*, 22, 38. <https://doi.org/10.3390/s22166041>
- Ismail, R. (2017). Enhancement of Online Identity Authentication Through Blockchain Technology. 4.
- Karunathilake, T., Udugama, A., & Förster, A. (s. f.). Simulation Speedup in OMNeT++ Using Contact Traces. 32-19. <https://doi.org/10.29007/s5q1>
- Kaur, K., Kaur, P., & Singh, E. S. (2014). Wireless Sensor Network: Architecture, Design Issues and Applications. 2(11), 6.
- Keerthika, M., & Shanmugapriya, D. (2021). Wireless Sensor Networks: Active and Passive attacks - Vulnerabilities and Countermeasures. *Global Transitions Proceedings*, 2(2), 362-367. <https://doi.org/10.1016/j.gltp.2021.08.045>
- Kocan, E., Domazetovic, B., & Pejanovic-Djurisic, M. (2017). Range Extension in IEEE 802.11ah Systems Through Relaying. *Wireless Personal Communications*, 97(2), 1889-1910. <https://doi.org/10.1007/s11277-017-4334-9>

- Laurence, T. (2017). *Blockchain*. John Wiley & Sons.
- Li, C. (2010). Security of Wireless Sensor Networks: Current Status and Key Issues. En *Smart Wireless Sensor Networks*. IntechOpen. <https://doi.org/10.5772/13158>
- Lim, S. Y., Fotsing, P., Almasri, A., Musa, O., Mat Kiah, M. L., Ang, T., & Ismail, R. (2018). Blockchain Technology the Identity Management and Authentication Service Disruptor: A Survey. *International Journal on Advanced Science, Engineering and Information Technology*, 8, 1735. <https://doi.org/10.18517/ijaseit.8.4-2.6838>
- López Lecumberri, G. (2011). Simulación de redes de computadores con OMNeT++4. <https://academica-e.unavarra.es/xmlui/handle/2454/4179>
- Martínez, A. (2020). *Blockchain, algoritmos de consenso [Universidad Oberta de Catalunya]*. <https://openaccess.uoc.edu/bitstream/10609/127926/6/aamoresmTFM1220memoria.pdf>
- Mingxiao, D., Xiaofeng, M., Zhe, Z., Xiangwei, W., & Qijun, C. (2017). A review on consensus algorithm of blockchain. 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2567-2572. <https://doi.org/10.1109/SMC.2017.8123011>
- Musleh, A. S., Yao, G., & Muyeen, S. M. (2019). Blockchain Applications in Smart Grid—Review and Frameworks. *IEEE Access*, 7, 86746-86757. <https://doi.org/10.1109/ACCESS.2019.2920682>
- Nair, P. R., & Dorai, D. R. (2021). Evaluation of Performance and Security of Proof of Work and Proof of Stake using Blockchain. 2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV), 279-283. <https://doi.org/10.1109/ICICV50876.2021.9388487>
- Oudjaout, A., Bagaa, M., Bachir, A., Challal, Y., Lasla, N., & Khelladi, L. (2009). Information Security in Wireless Sensor Networks. En *Encyclopedia on Ad Hoc and Ubiquitous*

- Computing (pp. 427-472). World Scientific. <https://hal.archives-ouvertes.fr/hal-00543985>
- Park, C., & Lahiri, K. (2005). Battery discharge characteristics of wireless sensor nodes: An experimental analysis. In Proceedings of the IEEE Conf. on Sensor and Ad-hoc Communications and Networks (SECON).
- Park, S., Savvides, A., & Srivastava, M. (2001). Simulating networks of wireless sensors. 2, 1330-1338 vol.2. <https://doi.org/10.1109/WSC.2001.977453>
- Parrondo, L. (2018). Tecnología blockchain, una nueva era para la empresa. *Revista de contabilidad y dirección*, 27, 11-31.
- Patel, P. N., Patel, J. K., & Virparia, P. V. (2013). A Cryptography Application using Salt Hash Technique. 2(6).
- Pawar, M. V., & Anuradha, J. (2015). Network Security and Types of Attacks in Network. *Procedia Computer Science*, 48, 503-506. <https://doi.org/10.1016/j.procs.2015.04.126>
- Popov, S. (2018). The Tangle. Version 1.4.3, 28.
- Porat, A., Pratap, A., Shah, P., & Adkar, V. (2008). Blockchain Consensus: An analysis of Proof-of-Work and its applications. 6.
- Quimbita, M. A. M., & Salvador, C. E. P. (2018). Evaluación de pasarela LoRa/LoRaWAN en entornos urbanos. 40.
- Raikwar, M., Gligoroski, D., & Velinov, G. (2020). Trends in Development of Databases and Blockchain. 177-182. <https://doi.org/10.1109/SDS49854.2020.9143893>
- Rajeswari, S. R., & Seenivasagam, V. (2016). Comparative Study on Various Authentication Protocols in Wireless Sensor Networks. *The Scientific World Journal*, 2016, e6854303. <https://doi.org/10.1155/2016/6854303>
- Retamal, C. D., Roig, J. B., & Tapia, J. L. (2017). La blockchain: Fundamentos, aplicaciones y relación con otras tecnologías disruptivas. Undefined.

<https://www.semanticscholar.org/paper/La-blockchain-%3A-fundamentos%2C-aplicaciones-y-con-Retamal-Roig/cfb7ffb23fd3cd8bab147f271affcffe66e4a993>

Rizwan, F. R. F., Hamid, H. S. S., Azmathullah, R. M. R., & Haque, F. R. R. (2018). Proof-of-Work Vs Proof-of-Stake: A Comparative Analysis and an Approach to Blockchain Consensus Mechanism. <http://repository.psau.edu.sa:80/jspui/handle/123456789/4749>

Rojas, A., & Rodrigo, L. (2018). ANÁLISIS DE LA TECNOLOGÍA BLOCKCHAIN, SU ENTORNO Y SU IMPACTO EN MODELOS DE NEGOCIOS. <https://repositorio.usm.cl/handle/11673/47346>

Rojas, L. R. Á. (2018). ANÁLISIS DE LA TECNOLOGÍA BLOCKCHAIN, SU ENTORNO Y SU IMPACTO EN MODELOS DE NEGOCIOS. 94.

Saeed, K., Khalil, W., Ahmed, S., Ahmad, I., & Khattak, M. N. K. (2020). SEECR: Secure Energy Efficient and Cooperative Routing Protocol for Underwater Wireless Sensor Networks. *IEEE Access*, 8, 107419-107433. <https://doi.org/10.1109/ACCESS.2020.3000863>

Santos Ramírez, J. R. (2020). Criptomoneda IOTA utilizando tecnología Tangle, una alternativa Blockchain [Other, Universidad de San Carlos de Guatemala]. <http://biblioteca.ingenieria.usac.edu.gt/>

Semtech Corporation. (2019, enero). Semtech SX1712 [Semtech SX1712-Datasheet]. Semtech SX1712.

https://semtech.my.salesforce.com/sfc/p/#E0000000JelG/a/440000001NCE/v_VBhk1IolDgxwwnOpcS_vTFxPfSEPQbuneK3mWsXIU

Silvano, W. F., & Marcelino, R. (2020). Iota Tangle: A cryptocurrency to communicate Internet-of-Things data. *Future Generation Computer Systems*, 112, 307-319. <https://doi.org/10.1016/j.future.2020.05.047>

- Smith, B. (2011). Object-Oriented Programming. En B. Smith (Ed.), *AdvancED ActionScript 3.0: Design Patterns* (pp. 1-25). Apress. https://doi.org/10.1007/978-1-4302-3615-3_1
- Sorrius Martí, M. (2018). Seguridad en la Internet de las cosas. Estudio de IOTA para el Internet of Things. Universitat Oberta de Catalunya.
- Stallman, R. (2020). La definición de software libre. <https://idus.us.es/handle/11441/100711>
- Turkanovic, M., & Holbl, M. (2013). An Improved Dynamic Password-based User Authentication Scheme for Hierarchical Wireless Sensor Networks. *Elektronika Ir Elektrotechnika*, 19(6), Article 6. <https://doi.org/10.5755/j01.eee.19.6.2038>
- University George Mason. (2007). Simulation of Wireless Communication Systems using MATLAB. Monte Carlo Simulation. <https://www.pdfdrive.com/simulation-of-wireless-communication-systems-using-matlab-e3820723.html>
- Vangala, S. (2017). Network Security: History, Importance, and Future SOUTHERN NEW HAMPSHIRE UNIVERSITY. https://www.academia.edu/34310842/Network_Security_History_Importance_and_Future_SOUTHERN_NEW_HAMPSHIRE_UNIVERSITY
- Wegrzyn, K. E., & Wang, E. (2021). Types of Blockchain: Public, Private, or Something in Between. 5.
- Xiao, M., & Guo, M. (2020). Computer Network Security and Preventive Measures in the Age of Big Data. *Procedia Computer Science*, 166, 438-442. <https://doi.org/10.1016/j.procs.2020.02.068>

ANEXOS

ANEXO 1. Ficha de Requerimiento

Proyecto de titulación: “Esquema de autenticación para redes de sensores (WSN) basada en tecnología de Cadena de Bloques y Tangle”

Objetivo de análisis: Entender de forma clara todos los requerimientos que debe contar el Esquema de Autenticación para brindar seguridad y cumplir con los objetivos planteados dentro del trabajo de investigación.

Fecha de realización: 22 de septiembre del 2022

Artículos consultados:

1. An Improved Dynamic Password-based User Authentication Scheme for Hierarchical Wireless Sensor Networks
2. The Tangle
3. Information Security in Wireless Sensor Networks
4. SEECR: Secure Energy Efficient and Cooperative Routing Protocol for Underwater Wireless Sensor Networks.
5. How long it takes hackers to crack your password based on how many characters it has
6. Simulating networks of wireless sensors

7. The Tangle
8. Equilibria in the Tangle
9. Evaluation of Performance and Security of Proof of Work and Proof of Stake using Blockchain.
10. Blockchain Consensus: An analysis of Proof-of-Work and its applications.
11. Blockchain Technology the Identity Management and Authentication Service Disruptor: A Survey.
12. An Enhancement Approach for Reducing the Energy Consumption in Wireless Sensor Networks
13. Evaluation of LoRaWAN Transmission Range for Wireless Sensor Networks in Riparian Forests.
14. Battery discharge characteristics of wireless sensor nodes: An experimental analysis.
15. Fault Tolerance Structures in Wireless Sensor Networks (WSNs): Survey, Classification, and Future Directions.
16. Scalability Analysis for Wireless Sensor Networks Routing Protocols.
17. Object-Oriented Programming.
18. Simulation Speedup in OMNeT++ Using Contact Traces.
19. La definición de software libre.
20. Seguridad en internet de las cosas usando soluciones Blockchain.

Lista de Stakeholders

| Descripción | Abreviatura |
|--------------------------------|--------------------|
| Requerimientos de Stakeholders | StSR |
| Requerimientos de Sistema | SySR |
| Requerimientos de Arquitectura | SrSH |

| Requerimiento de Stakeholders | | | |
|--------------------------------------|---|--|------------------|
| Nomenclatura | Requerimiento | Descripción | Prioridad |
| StRS1 | Envío de paquete de actualización de contraseñas | La autenticación de usuarios es un tema importante en las redes de sensores inalámbricos, es por lo cual que la creación de un esquema de autenticación basado en contraseña dinámicas brinda alta seguridad ya que estas cambian después de un tiempo determinado brindando un enfoque de autenticación simple pero muy segura. (Turkanovic & Holbl, 2013) | Alta |
| StRS2 | Verificación de nodo honesto por dos nodos de la red | Esta característica es propia de Tangle, esto se debe a que el nodo elige otras dos transacciones para aprobar; de esta forma el nodo verifica si las dos transacciones no están en conflicto para pasar a su aprobación.(Popov, 2018) | Alta |
| StRS3 | Creación de canal seguro para envío de datos | El medio de comunicación inalámbrico y los recursos limitados del sensor, generan un conjunto de vulnerabilidades, las cuales hacen a las redes de sensores propensas a diversos ataques,tanto en la capa física como en la de enlace (Ouadjaout et al., 2009), por lo cual es de gran importancia crear un canal seguro; por medio de la autenticación de nodos, para que de esta forma la información enviada a través de este canal , pueda llegar a su destino sin cambios o alteraciones. | Alta |
| StRS4 | Comprobación de paquetes de entrada y salida a través del protocolo SEECR | Para evitar el ataque de nodos maliciosos, se compararán tanto los valores de paquetes de entrada como de salida donde, si ambos valores no son iguales y el nodo sensor no es un nodo sumidero entonces hay posibilidades de que el nodo sensor sea un nodo atacante.(Saeed et al., 2020) | Alta |
| StRS5 | Tiempo de actualización de contraseñas máximo (1 min) | Según BusinessTech (2022) en la actualidad una contraseña compuesta por números, letras y símbolos, cuya longitud sea de siete caracteres es descifrada en treinta y nueve segundos , por lo cual la actualización de contraseñas parte con un tiempo máximo de un minuto , el cual disminuirá conforme aumente el peso de los nodos; consiguiendo de esta forma dar mayor seguridad al esquema. | Media |

| | | | |
|--------------|---|--|-------------|
| StRS6 | Tiempo de duración de batería del nodo génesis mínimo (3 horas) | Según Park et al.(2001) la capacidad de la batería se puede modelar en función de la energía y comportamiento de los nodos consumidores, es por tal razón que para tener un mejor funcionamiento de los nodos en cada una de las islas se estima que la duración de la batería debería ser como mínimo tres horas. | Baja |
|--------------|---|--|-------------|

| Requerimiento de Sistema | | | |
|---------------------------------|---|--|------------------|
| Nomenclatura | Requerimiento | Descripción | Prioridad |
| SySR1 | Los dos primeros nodos sensores se autenticarán con la estación base y los demás se autenticarán entre ellos. | Para tener un diseño basado en Tangle, se adaptó el modelo de aprobación de transacciones a una forma de autenticación de nodos; ya que como indica Popov, (2018) cuando llega una nueva transacción, debe ser aprobada por dos transacciones previas las cuales están representadas por aristas dirigidas, logrando así realizar las transacciones de forma segura. | Alta |
| SySR2 | Al realizar una transacción exitosa el peso acumulado de los nodos sensores aumentara. | Popov (2018) indica, sí una transacción es lo suficientemente antigua y con un gran peso acumulativo, entonces el peso acumulado crece a gran velocidad, creando así una red central confiable. | Alta |
| SySR3 | La estación base o nodo génesis recibirá los datos de los sensores y los registrará. | En esta parte se implementa el funcionamiento de libro mayor distribuido, propio de Blockchain, en el cual mediante el manejo de una red peer-to-peer que colabora con una red compartida; se forma el protocolo de validación de nuevos bloques de información.(Nair & Dorai, 2021) | Alta |
| SySR4 | El nodo génesis que resuelva más pronto la prueba de trabajo agregara su bloque candidato a la cadena principal | La idea principal detrás del protocolo es hacer que los nodos resuelvan un problema computacionalmente costoso antes de que puedan sugerir un nuevo bloque. El nodo que primero resuelve el problema extrae el nuevo bloque y transmite el mensaje a los otros nodos en la red (Porat et al., 2008). | Media |
| SySR5 | Todos los nodos génesis de las islas tendrán una copia de la cadena principal. | Lim et al. (2018) nos indica que Blockchain busca tener cada transacción registrada con cada miembro de la red para esta manera se elimine la intermediación , creando así una red descentralizada | Alta |

| Requerimiento de Sistema | | | |
|-----------------------------------|--|--|------------------|
| Requerimientos de Hardware | | | |
| Nomenclatura | Requerimiento | Descripción | Prioridad |
| SrSH1 | Bajo consumo de Energía al realizar el proceso de autenticación de nodos | Cuando el tamaño de la red aumenta, los nodos que están ubicados lejos de la estación base consumen más rápidamente su energía (Elshrkawey et al., 2018), por lo cual al ser una red escalable se debe contar con una fuente de energía duradera. | Media |
| SrSH2 | Mayor alcance con tasas de transmisión bajas | Este apartado permite crear un compromiso entre la tasa de datos y la cobertura, consiguiendo así optimizar el rendimiento de la red con un ancho de banda constante.(Avila-Campos et al., 2019) | Media |
| SrSH3 | Duración de Batería | La extensión de la vida útil de la batería, es el principal impulsor del diseño de las redes de sensores inalámbricos (WSN) de bajo consumo(C. Park & Lahiri, 2005), ya que de esta forma se evita que los nodos se agoten al realizar la resolución de algoritmos complicados. | Media |
| SrSH4 | Tolerancia a la interferencia | Hassan et al., (2022) afirma que la estructura subyacente de tolerancia a fallos es un requisito crítico que debe tenerse en cuenta al diseñar cualquier algoritmo en WSNs, debido a que se crea un esquema de autenticación el tener un sistema con alta tolerancia a interferencia evitara el robo y pérdida de información. | Alta |
| SrSH5 | Escalabilidad | La escalabilidad es una propiedad esencial para la desempeño exitoso de cualquier red que implica un gran número de nodos, los cuales manejan una capacidad de computación limitada, comunicación inalámbrica y detección, tomando en cuenta que si el número de estos nodos aumenta, se debe determinar si la degradación en la red y el rendimiento puede ser tolerado.(Alazzawi et al., 2008) | Alta |
| SrSH6 | Comunicación Inalámbrica | Todos los dispositivos que formen parte de la red deben poseer la característica de ser inalámbricos ya que caso contrario no serían de gran utilidad para la demostración del esquema propuesto. | Alta |
| Requerimientos de Software | | | |
| SrSH8 | Lenguaje orientado a objetos | Smith, (2011) afirma que, este lenguaje crea una arquitectura de software, que permite flexibilidad a través del diseño modular, además de que ayuda a prevenir | Alta |

| | | | |
|---------------|--|--|--------------|
| | | el código inmanejable, por lo cual es un lenguaje muy útil para el esquema diseñado. | |
| SrSH9 | Funcionamiento en sistemas operativos Linux y Windows | Este apartado es de gran importancia ya que nos permitirá trabajar en ambos sistemas operativos y adaptar la simulación de IOTA al ambiente de Blockchain. | Media |
| SrSH10 | Rápido Procesamiento | Este ítem es de gran importancia para realizar de una forma rápida la toma de decisiones de reenvío de datos, así como también para encontrar la ubicación de sus nodos vecinos. (Karunathilake et al., s. f.) | Alta |
| SrSH11 | Manejo de librerías y protocolos enfocadas en telecomunicaciones | Debido a que se está realizando una simulación de una red de sensores, el hecho de que el programa cuente con los protocolos y las características de estos dispositivos inalámbricos, nos ayuda a crear una simulación más cercana a la realidad y con datos certeros. | Alta |
| SrSH12 | Software Libre | Este elemento es de gran utilidad ya que como usuarios podremos ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software, teniendo siempre la libertad para ejecutar el programa sea cual sea nuestro propósito; así como también adaptarlo a nuestras necesidades en caso de ser necesario. (Stallman, 2020) | Baja |
| SrSH13 | Compatibilidad con Solidity | Solidity se presenta con un lenguaje de programación orientado a objetos utilizado para escribir contratos inteligentes en la plataforma Ethereum que cuenta con muchas similitudes con C y C++ (Eterovic et al., 2021) , por lo cual es de gran importancia que el programa usado para el desarrollo de la red IOTA sea compatible con Solidity para así poder crear el Blockchain entre los nodos génesis. | Alta |

Elaborado por: Carla Vallejo

ANEXO 2

REPOSITORIO DE GITHUB

<https://github.com/CarlaVallejo/Esquema-de-autenticaci-n-Blockchain-y-Tangle> basado en el código del repositorio <https://github.com/T-amairi/IOTA>