UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Diseño e implementación de un mecanismo basado en principios de Blockchain para detectar Sybil Attack en redes IoT homogéneas

TESIS

Que para obtener el título de

Ingeniero en Telecomunicaciones

P R E S E N T A N

Alejandro Hiram Ramírez Martínez

Jorge Orlando González Guzmán

DIRECTOR DE TESIS

Dr. José Jaime Camacho Escoto

Ciudad Universitaria, Cd. Mx., 2022

Índice general

Capítulo 1

Introducción

1.1. Planteamiento del problema.

En la actualidad, la vanguardia tecnológica ha revolucionado las aplicaciones de las redes de telecomunicaciones en la vida cotidiana del ser humano. Sin embargo, dicho crecimiento también ha propiciado que los avances para alterar, dañar o robar información representen un serio peligro para los datos que están presentes en las redes. Uno de los ataques que generan mayor problemática es el denominado "Sybil Attack", dicho ataque consiste en una entidad maligna ajena a los elementos originales de la red que se proclama parte de esta con múltiples identidades, debido a sus características, las redes inalámbricas ad hoc son el blanco ideal para este tipo de ataque.

En el peor de los escenarios, el ataque antes mencionado puede alterar varias funcionalidades de la red, entre las que más destacan mecanismos basados en evaluación por votos, esquemas de asignación de recursos, la detección de otros ataques y mecanismos de encaminamiento [1]. Los elementos listados anteriormente representan un peligro inminente a la funcionalidad y confidencialidad de la red, por lo que un mecanismo para la detección temprana del "Sybil Attack" debe ser implementado en todas las WANETs (wireless ad hoc network).

En la actualidad existen diversas propuestas para detectar un "Sybil Attack" en una WANET. Algunas soluciones proponen validar a los nodos de la red por medio de una entidad certificadora [1] (que tiene alcance en toda la red). Sin embargo, algunas de las principales desventajas y limitaciones que se pueden presentar son que la red debe ser de un tamaño reducido, que un nodo debería tener los recursos computacionales para realizar todo el procesamiento necesario en el envío y recepción de mensajes.

Otros esquemas proponen que se distribuyan diversos canales a lo largo de la red, posteriormente, de forma aleatoria se escoja un nodo para enviar un desafío en ese canal de la red. El desafío se elige de forma que su cálculo sea lo suficientemente complejo para que requiera un tiempo determinado. Se asume que si un nodo no contesta el desafío en ese tiempo, entonces se tratará de un nodo malicioso [1]. La falla de este esquema es que si el nodo elegido tuviera más de una interfaz de red, no podría ser detectado como malicioso y por otro lado, si un nodo llegase a quedar sin energía suficiente sería tachado como malicioso aunque no lo sea.

Otros esquemas analizan las distancias entre nodos haciendo uso del RSSI (received signal strength indicator), potencia de recepción, SNR o algún otro parámetro de medición de la señal [1]. En estos esquemas, si un nodo malicioso está enviando diversos mensajes con diferentes identidades, la recepción de estos mensajes se tendrían valores iguales de los parámetros mencionados. Se puede presentar que el nodo malicioso durante la transmisión de los mensajes varíe su potencia con la que transmite, por lo que estaría vulnerando este esquema provocando que el nodo malicioso no se pueda detectar.

Los esquemas anteriores son algunos ejemplos de las diversas propuestas que se tienen para poder detectar una identidad Sybil pero las limitantes que se presentan ante estos hacen que el problema no esté resuelto en su totalidad.

1.2. Justificación.

Ante las diversas problemáticas que surgen debido al “Sybil Attack”, es importante que se tenga un esquema suficientemente robusto que cuide la integridad y confidencialidad de los datos que pudieran estar en una red IoT (internet of things). Se estima que en un lapso de tiempo menor a tres años este tipo de redes tengan un gran crecimiento, con un número estimado de 22,000 millones de dispositivos [2], al grado de estar presentes en hogares, ciudades y oficinas, todo lo anterior a nivel mundial. Dicho esto, prevenir un “Sybil Attack” es un asunto que involucra los intereses tanto de empresas como los de sus consumidores que deseen mantener la confidencialidad de su información, o incluso usuarios que deseen implementar su propia red de IoT.

Por otra parte, además de los escenarios que fueron mencionados en el planteamiento del problema, existe una consecuencia que se puede tener por un “Sybil Attack” exitoso, y es que surjan nuevos y diferentes ataques basados en el anterior que afecten de maneras distintas a la

red [1]. Por lo tanto, se debe desarrollar un algoritmo que pueda detectar el "Sybil Attack". El desarrollo de este algoritmo pudiera ser una propuesta que tenga implicaciones más allá de este esquema en IoT (por ejemplo otro tipo de WANETs). Sin embargo, será necesario que se lleven a cabo pruebas para predecir el comportamiento del algoritmo en la red que se desea proteger ante un ataque.

1.3. Objetivos.

1.3.1. Objetivo General.

Diseñar un algoritmo inteligente que detecte cuándo una entidad maligna se declara con múltiples identidades en una red (Sybil Attack), tomando como referencia los aspectos útiles de otros mecanismos junto con un esquema de blockchain. El mecanismo se implementará en varios dispositivos conectados a una red que tendrá las características y comportamiento de una red IoT homogénea con la intención de verificar su fiabilidad en un ambiente real.

1.3.2. Objetivos específicos.

1. Profundizar en las soluciones ya existentes para la detección de Sybil Attack con el fin de recopilar las deficiencias de los mismos para estructurar una solución más robusta.

2. Buscar un primer mecanismo que permita detectar identidades falsas en base a la distancia a la que se encuentran.

3. Mediante pruebas de campo, obtener el conjunto de parámetros ideales para el correcto funcionamiento del algoritmo.

4. Diseñar una metodología que opere con principios de Blockchain lo suficientemente inteligente para minimizar el error en la detección de nodos malignos.

5. Caracterizar la metodología propuesta por medio de pruebas en términos de fiabilidad.

6. Probar la eficiencia del algoritmo en una red compuesta por varios dispositivos que compartan características y posean un comportamiento que simule una red IoT homogénea.

1.4. Metodología.

Para lograr cumplir con los objetivos, se plantea la siguiente metodología:

1. Encontrar algún dispositivo que permita medir los parámetros requeridos(RSSI, SNR, etc) de un paquete recibido.
2. Realizar un programa que permita medir los parámetros mencionados anteriormente para un enlace punto a punto.
3. Repetir las mediciones anteriores con distintas distancias y documentar los resultados.
4. Procesar la información anteriormente obtenida para obtener valores que puedan ser utilizados por el algoritmo.
5. Realizar un programa que permita descartar a una identidad falsa utilizando los valores obtenidos en el paso anterior, en un enlace punto a punto.
6. Repetir el algoritmo de descarte para verificar la fidelidad del programa.
7. Escribir un programa que genere un desafío para los nodos detectados como malignos.
8. Generar pruebas con distintas dificultades para los desafíos.
9. Generar pruebas con diferentes dificultades de desafíos para todas las identidades de la red.
10. Escribir un programa que realice un consenso.
11. Generar pruebas para el programa de consenso con distintas dificultades.

1.5. Contribuciones

Este trabajo utiliza un esquema de tres fases para la detección de un "Sybil Attack", que son la detección basada en el RSSI y la implementación de conceptos de Blockchain (pruebas de trabajo y consensos). La adición de las pruebas de trabajo y del consenso permiten aumentar la precisión de detección del ataque con respecto a utilizar únicamente la detección por RSSI. El uso del esquema de detección por RSSI previo a las pruebas de trabajo disminuye ya carga de trabajo

del procesador del dispositivo que envía los desafíos. Por último, la etapa del consenso permite mejorar la precisión de la detección y que esta se realice por "mayoría de votos".

Se prevé que este esquema multi-fase pueda ser modificado para detectar otros ataques cibernéticos que aprovechen las vulnerabilidades de los algoritmos de enrutamiento de paquetes en redes inalámbricas. Lo anterior con la finalidad de fijar un antecedente que permita robustecer la seguridad en redes que no solo entren en la categoría de IoT.

La implementación presentada en este trabajo es la primera que pretender usar los tres esquemas antes mencionados en conjunto y de forma secuencial para tener una detección más selectiva al mismo tiempo que se reduce la carga de procesamiento en los dispositivos.

Capítulo 2

Sustento teórico

2.1. Introducción

El sustento teórico permite realizar un análisis conceptual del problema que se desea solucionar y el escenario donde aplicará el esquema de detección diseñado, unificando la base teórica necesaria que sustenta este trabajo.

Con la finalidad de lograr el entero entendimiento del contenido de este trabajo, en este capítulo se explicarán los conceptos necesarios para entender como las bases que fueron tomadas de la tecnología Blockchain se usaron para elaborar un esquema de detección para "Sybil Attack" en una red IoT; además de presentar los dispositivos utilizados para modelar la red antes mencionada y los parámetros físicos que sirvieron como referencia en el esquema de detección.

2.2. Blockchain

Blockchain es una estructura de datos o registro único, implementado con software, que además cumple con las características de ser consensuado, descentralizado y distribuido en varios nodos dentro de una red [3], que permite realizar transacciones punto a punto con criptomonedas sin ayuda de terceros de forma segura y confiable, mediante procesos de validación, verificación y confirmación. En la figura (2.1) se muestra un ejemplo de una red descentralizada y en la figura (2.2) un red centralizada.

Una red centralizada como la mostrada en la figura (2.2), es aquella que es controlada por una entidad central, la cual, tiene conocimiento de todos los nodos en la red. En consecuencia, una red

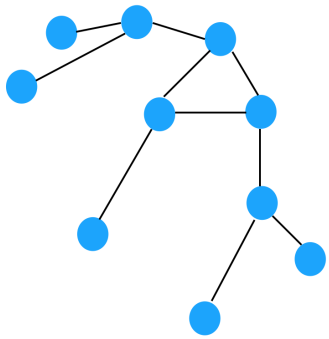

Figura 2.1: Red descentralizada. **Figura elaborada por los autores de este trabajo**.

centralizada almacena la información de todas las entidades en un solo nodo central que a su vez procesa esta información y toma decisiones en base a esta información.

Una red descentralizada (2.1), por el contrario, tiene dispersa la información y datos en cada uno de sus nodos. Esto se presta a que las decisiones en la red sean un consenso de los nodos que forman parte de ella.

La idea principal de blockchain es aplicada para realizar pagos digitalmente sin la necesidad de que este dinero pase por una institución bancaria, por lo que se logra un sistema descentralizado en el que los pagos se hacen "peer-to-peer". Un sistema "peer-to-peer.º también conocido como P2P, es aquel en el que los nodos de la red tienen un rol client/servidor al mismo tiempo, de tal forma que todos participan prestando sus recursos, ya sean CPU, memoria o espacio en disco [4]. Para ello, existe un consenso en el que otros nodos de la red participan y verifican la validez de las transacciones que se están realizando, se debe aclarar que estos nodos no tienen la conciencia de las identidades de quienes realizan y reciben una transacción, por lo que la confidencialidad está asegurada.

En Blockchain, la información de la transacción es almacenada en bloques contables, dentro de dichos bloques se almacena la siguiente información:

- Cantidad de registros o transacciones válidas.
- Información referente a ese bloque.
- Vinculación con el siguiente bloque a través de un hash.

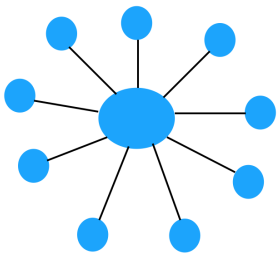

Figura 2.2: Red centralizada. **Figura elaborada por los autores de este trabajo**.

Cada uno de los bloques posee un lugar inamovible en la cadena, una vez que un elemento ingresa a la cadena no puede ser modificado por lo que al registro de blockchain se le atribuye la característica de inmutabilidad. La cadena completa se guarda en cada nodo de la red. Este mecanismo no exige un tercero que certifique la información, sino que se distribuye en múltiples nodos que la examinan sin necesidad de que se conozcan entre ellos [5], haciendo la información prácticamente imposible de falsificar. Conforme avance el tiempo, y se tengan que añadir nuevos registros, estos deben ser verificados y validados utilizando certificados y firmas digitales por los nodos de la red para ser añadidos como bloques dentro de la cadena.

En blockchain los datos están distribuidos en todos los nodos de la red. Al no haber un nodo central, todos participan por igual, almacenando y validando toda la información [3]. Lo que lo convierte en una gran herramienta para almacenar información de forma confiable.

Para la funcionalidad de blockchain existe un subgrupo de nodos a los que se les denomina mineros, dichos nodos realizan las operaciones para que las transacciones de criptomonedas se realicen de forma exitosa. Blockchain tiene tres aspectos muy importantes, en los que hace énfasis cuando se requiere implementar este mecanismo[6]:

- Validación
- Verificación
- Consenso

Estos conceptos son usados en una red de blockchain para tener un sistema de confianza y seguridad cuando se habla de realizar transacciones entre dos nodos.

2.2.1. Validación

Debido a la gran cantidad de transacciones que se realizan en la actualidad, la implementación del concepto de bloque dentro del registro único fue realizado para optimizar el proceso de validación de las transacciones que se llevan a cabo [7].

La información del bloque es una referencia a través de una hash, que es el resultado de una operación criptográfica que genera identificadores únicos e irrepetibles a partir de una cierta entrada [8], para poder hacerla más manejable y más eficiente, un bloque es marcado con su respectivo hash para marcar la validez y autenticidad del mismo.

2.2.2. Consenso

En una red de Blockchain el consenso es una parte fundamental, ya que garantiza la integridad de la cadena de bloques, haciendo que todos los nodos que participen en la red acepten la información que la cadena contiene [9]. Por lo tanto, el consenso es el proceso que se lleva a cabo para que los elementos que pertenecen a la red acepten que la información no contiene manipulaciones, ni datos erróneos o modificados.

Entre los objetivos principales del consenso, es evitar que se añadan bloques erróneos en la cadena, por lo que cada uno de ellos necesita de una revisión y una confirmación [9]. Es importante mencionar, que para lograr un consenso exitoso, es relevante que existan el mayor número de nodos posibles dentro de la red Blockchain.

El consenso es el sistema de sincronización entre todos lo nodos en una red de Blockchain, pues gracias a esta particularidad se puede asegurar que la información no esta dañada ni alterada mediante un acuerdo entre las entidades antes mencionadas, para que esto se pueda llevar a cabo, los bloques necesitan de la confirmación de todos lo nodos de la red, es decir, que estos validen los datos que e encuentran en cada uno de los bloques.

2.2.3. Desafío

Una de las operaciones que realizan los mineros es resolver un desafío que es generado cuando se quiere tomar una de las transacciones validadas para añadirse a la cadena, los mineros compiten por resolver este desafío para determinar quién gana el derecho de crear el siguiente bloque de la cadena, así que una vez que uno de los mineros haya finalizado el desafío, éste deberá de anunciarlo

en toda la red.

El desafío anterior fue implementado con el fin de evitar que dos o más mineros tomen la transacción validada, lo que tendría como consecuencia que se crearan múltiples bloques haciendo que la cadena perdiera su comportamiento, generando múltiples problemas asociados a la accesibilidad y seguridad de la información.

Algunas criptomonedas implmentan sus propios desafíos, en el caso del bitcoin, se usa Proof of Work. Debido a que este mecanismo requiere de muchos recursos computacionales, existen otros desafíos alternativos a Proof of Work que ayuden a realizar el consenso en blockchain, por ejemplo: Proof of Stake, Leased Proof of Stake, Proof of Importance, etc [10].

Proof of Work (PoW)

Este es un mecanismo para lograr el consenso en la red blockchain que se implementó inicialmente en la criptomoneda bitcoin. En PoW se hace que cada minero o computadora de la red resuelva operaciones matemáticas que resulten sencillas para una computadora, pero siendo repetitivas y de un costo (en recursos computacionales) lo suficientemente elevado para el ordenador. Por ende, nodos de la red blockchain están buscando constantemente un valor que está generado con una función hash [11]. El principio de PoW está diseñado para que sea difícil incluir un bloque a la cadena, pero fácil de verificar si las transacciones son válidas.

Proof of Stake(PoS)

En PoS, se define una probabilidad para cada moneda que esté en la red, de esta forma se plantea minar aquel bloque que tenga la mayor probabilidad en su red . En la red cada nodo se asocia a una dirección, cuantas más monedas tenga mayor es la probabilidad de que el bloque sea minado o añadido a la cadena de bloques.

Función Hash

Complementando lo mencionado en la sección (2.2.1), una función hash es una función criptográfica que tiene como objetivo codificar datos para formar una cadena de caracteres única, sin importar la cantidad de datos que entren como argumento en dicha función [8].

El algoritmo sha-256 que proviene de la familia sha-2 es uno de los algoritmos más utilizados para cumplir el objetivo de una función hash en múltiples aplicaciones de confidencialidad, en su

caso particular esta versión siempre tendrá como salida 256 bits [12].

La metodología con la que el algoritmo funciona, incluye operaciones de bajo nivel como son la AND y la OR, sin embargo estas operaciones son una pequeña parte de la complejidad de esta metodología, que sigue los siguientes paso:

1. Se inicializan dos vectores estándar necesarios para la función del algoritmo, el primero al se le denomina "k" y posee una extensión de 64 elementos, por otra parte, el segundo al que se le denomina "hash", que posee una extensión de 8 elementos, los elementos de ambos vectores tienen un tamaño de 32 bits

2. Se realiza un padding o relleno de la entrada de la función, con la intención de que esta quede en múltiplos de 64 bits.

3. La entrada ya rellenada se separa en bloques de 64 bits.

4. Los bloques de 64 bits que entran son procesados con operaciones estándar del algoritmo que incluyen operaciones de bajo nivel, condicionales y actualizaciones de variables, dichas operaciones estándar son ejecutadas 64 veces.

5. Se actualiza el vector hash con las salidas del punto anterior, la concatenación de los elementos de dicho vector es la salida del algoritmo (256 bits).

2.3. Ataques en redes

Los ataques en redes han crecido a la par que nuevas tecnologías surgen (que impliquen la interconexión de dispositivos). En redes ad-hoc, la vulnerabilidad para perpetuar ataques es aún mayor debido a la ausencia de una entidad central. Existen dos clasificaciones para los ataques en una red : ataques pasivos y ataques activos[13]. Los ataques que pueden ocurrir en una red ad-hoc son diversos, algunos ejemplos de ellos son :

- Sybil Attack.
- Sinkhole Attack.
- Wormhole Attack.
- BlackHole.

- Rank Attack
- Broadcast Attack.
- Selective forwarding.

2.3.1. Ataques pasivos

Los ataques pasivos son aquellos en los un nodo de la red está espiando o viendo información que circula a través él[13], por lo que el nodo malicioso tiene acceso a información ajena a él. Esto puede traer consecuencias en las que por ejemplo, se puedan obtener mensajes, contraseñas e información relevante de un usuario.

2.3.2. Ataques activos

Esta clasificación de los ataques involucra la acción realizada por algún nodo malicioso de la red, ya sea que esté replicando, modificando, o eliminando datos que circulan entre los nodos [13]. Este tipo de ataque suele tener efectos y consecuencias en la red en los que se puede enviar información personal o delicada a nodos de la red que no son requeridos siendo que se puede dar un desvío de tráfico de datos en la red. Además, este tipo de ataques degrada el rendimiento de una red.

2.3.3. Sybil Attack

Un Sybil Attack se da en una red cuando se realiza el intercambio de mensajes entre nodos para presentarse y así identificarse mediante una etiqueta o ID y entonces identificar quienes son vecinos de un nodo. Es entonces, cuando un nodo malicioso comienza generar identidades que pudieran estar o no presentes en la red, el nodo malicioso las dispersa a lo largo de la red, haciendo que los nodos que si son reales identifiquen etiquetas que son falsas como vecinos [1]. Un Sybil Attack pudiera hacer que los nodos reales estén enviando información a través de nodos maliciosos y así desviar el tráfico de toda la red hacía el nodo malicioso. En la figura (2.3), se muestra un diagrama de como funciona Sybil Attack. En donde el nodo “A” es malicioso y está suplantando identidades, mientras que s1, s2, s3, s4, y s5 son las identidades falsas que está suplantando el nodo “A”. A las falsas identidades que se dispersan por la red, como se observa en la figura (2.4), se les llama Sybil nodes[13].

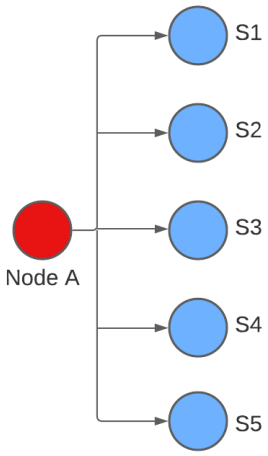

Figura 2.3: Generación de falsas identidades por un atacante.**Figura elaborada por los autores de este trabajo**.

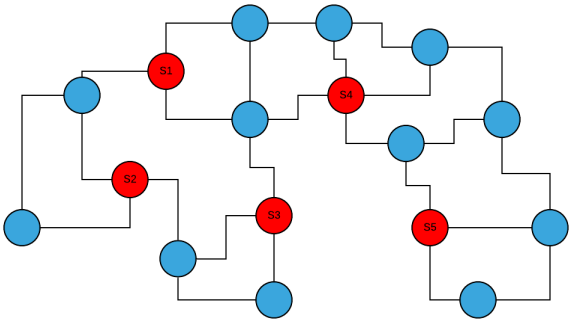

Figura 2.4: Sybil nodes dispersos en una red inalámbrica. **Figura elaborada por los autores de este trabajo**.

Existen variantes de "Sybil Attack" que se dan bajo diferentes circunstancias, como por ejemplo, el origen de las identidades que proclama el nodo malicioso o como se da la comunicación entre los nodos Sybil con los nodos buenos. Estos se dividen en tres ejes [14] que pueden efectuar un "Sybil Attack":

- Tipo de comunicación : directa o indirecta.
- Origen de las identidades : robada o fabricada.
- Tiempo en la participación de un nodo Sybil : simultánea o no simultánea.

Se debe de observar que un "Sybil Attack" puede darse en situaciones en los que se tengan

ataques en estos tres ejes de forma simultánea, por ejemplo, podría tener comunicación directa y tener una participación no simultánea en la red.

Comunicación directa

En este tipo de interacción, un nodo malicioso que está falsificando identidades dentro de la red comienza a difundir mensajes de tipo "HELLO", con los cuales se presenta ante sus vecinos más cercanos, ocasionando que estos vecinos le respondan el mensaje "HELLO". En este punto el nodo malicioso puede difundir mensajes de tipo "HELLO" con otras identidades (correspondientes a identidades Sybil) a sus vecinos más cercanos, lo que provocará que estos tengan en sus registros de la vecindad algunas identidades que son falsas correspondientes a nodos Sybil, ya que tienen vista directa con el nodo malicioso las identidades falsas de los nodos Sybil también podrán quedar registradas como vecinos próximos a un nodo legítimo [1]. Se le conoce como comunicación directa porque el nodo verdadero intenta comunicarse con estos nodos falsos sin la necesidad de un intermediario.

Comunicación indirecta

A diferencia de la comunicación directa, la comunicación indirecta está diseñada para que un nodo malicioso proclame en su vecindad identidades falsas, haciéndose pasar como el vecino más próximo a los nodos . Esto provoca que en un momento dado los nodos legítimos que pretendan enviar un mensaje a los nodos tengan que rutear sus mensajes a través del nodo malicioso, ya que sería la única ruta o la más barata para hacer llegar un mensaje a los nodos falsos [1].

Identidades fabricadas

En esta modalidad, el nodo malicioso genera identidades falsas de las que no se tiene registro alguno en la red, es decir, son nuevas y no son asociadas a algún dispositivo en particular. Comúnmente, las identidades generadas son aleatorias.

Identidades robadas

El origen de las identidades que genera un nodo malicioso son existentes dentro de la red y le pertenecen a dispositivos reales que están o estuvieron presentes en la red.

Tiempo en la participación de un nodo

Un nodo malicioso puede hacer que sus identidades falsas (nodos Sybil) participen en la red de diversas formas. Se considera como participación simultánea si todos los nodos Sybil participan a la vez en la red, es decir están diseminando información en la red al mismo tiempo, esto pudiera pasar en el escenario en el que un nodo malicioso cuenta con un dispositivo de red capáz de transmitir y recibir al mismo tiempo. Por el contrario, es no simultáneo cuando los nodos Sybil deben comunicarse uno a la vez. Esto correspondería a dispositivos más comunes en el ámbito de internet de las cosas.

Afectaciones en la red causadas por Sybil Attack

En particular, Sybil Attack vulnera una red en muchos aspectos. Esto es debido a la capacidad que tiene un mismo nodo para generar desviaciones en el tráfico de una red, crear spam o incluso robar recursos asignados en la red. Algunos de las afectaciones relatadas son:

- Almacenamiento distribuido.
- Enrutamiento.
- Datos agregados.
- Votación.
- Alojamiento de recursos.

2.4. Internet de las cosas

El término Internet de las Cosas (Internet of Things, IoT por sus siglas en inglés) fue utilizado por primera vez por el informático de origen británico Kevin Ashton en el año de 1999, que durante seis meses, trató de persuadir a la empresa Procter and Gamble, mundialmente conocida como P&G, para que pusiera etiquetas de identificación de radiofrecuencia y otros sensores en los productos de la cadena de suministro [15].

IoT es una arquitectura emergente basada en la Internet, que facilita el intercambio de bienes y servicios entre redes y que tiene un impacto importante en la seguridad y privacidad de los actores involucrados [16]. Describe la red de objetos físicos que llevan sensores integrados, software y otras

tecnologías con el fin de conectar e intercambiar datos con otros dispositivos y sistemas a través de Internet [2]. Los dispositivos que conforman la red pueden ser desde objetos domésticos cotidianos hasta sofisticados nodos industriales. La coexistencia e interconexión de estos dispositivos se logra sobre muchos tipos de redes, entre las que destacan IP e Internet, dichas redes permiten que los dispositivos intercambien mensajes por un determinado protocolo, este modelo de comunicación es usualmente utilizado en sistemas de automatización en el hogar, donde se utilizan pequeños paquetes de datos para lo comunicación entre los dispositivos por lo que los requisitos en cuanto a tasa de transmisión son relativamente bajos [17]. Otra característica importante de esos dispositivos es el envío de pequeñas cantidades de información en lapsos de tiempo que pueden llegar a durar hasta horas, por lo que el consumo energético de los mismos es muy bajo. Adicionalmente, los dispositivos IoT se ven limitados por factores técnicos como el procesamiento y memoria, ya que los fabricantes y proveedores de estas tecnologías se ven obligados a reducir el costo unitario para mantenerse competitivos ante el gran mercado que existe actualmente y que crecerá de forma exponencial en los siguientes años.

En Marzo de 2015, el Comité de Arquitectura de Internet publicó un documento para guiar la creación de redes de objetos inteligentes (RFC 7452), que describe cuatro modelos de comunicación utilizados en dispositivos IoT [17].

- Dispositivo a dispositivo.- Representa dos o más dispositivos que se conectan y se comunican entre sí de forma directa, sin necesidad de un intermediario. Esto implica el uso de un protocolo que puede no ser compatible con otros dispositivos subyacentes.

- Dispositivos a la nube.- Representa a los dispositivos IoT que se conectan directamente a un servicio en la nube. De igual forma que en el modelo de dispositivo a dispositivo si se utilizan protocolos de datos propietarios, se limitaría el uso de proveedores de servicio alternativos.

- Dispositivo a puerta enlace.- El dispositivo IoT se conecta a través de un servicio de enlace de capa de aplicación para poder llegar a un servicio en la nube.

- Intercambio de datos a través del backend.- Se refiere a una arquitectura que permite que los usuarios exporten y analicen datos de objetos inteligentes de un servicio en la nube [17].

IoT abre las posibilidades de conectar cualquier dispositivo a Internet independientemente de que sea portátil o no, convirtiéndolo en una red gigante que permite la interacción entre personas

y objeto, ofreciendo cambio radical en la calidad de vida de las personas, otorgando oportunidades de acceso a datos en la educación, en seguridad o en el transporte. Por otra parte, esta implementación será clave para aumentar la productividad de las empresas, ya que ofrece redes locales de dispositivos inteligentes y nuevos servicios que pueden ser personalizados según las necesidades del cliente. La IoT aumenta la cantidad de datos de información, ofreciendo la oportunidad de crear nuevos dispositivos interconectados inteligentes y explorar nuevos modelos de negocio [16].

La confiabilidad, la superación frente a la adversidad y la estabilidad de las aplicaciones y servicios que puede llegar a brindar IoT es fundamental para fomentar la confianza y su uso [17]. Si los usuarios de este tipo de redes no creen que sus dispositivos conectados y la información que contienen se encuentran protegidos y mantienen su confidencialidad, se puede provocar una caída severa en las aplicaciones de servicios que ofrece Internet de las cosas.

A medida de que aumenten la cantidad de dispositivos conectados a una red IoT, aumenta el número de oportunidades potenciales para que se de un ciberataque, que puede llegar a provocar la programación o daño al funcionamiento de los dispositivos conectados, exponiendo los datos del usuario. Por esta razón, la seguridad de los dispositivos y de los servicios IoT se vuelven temas críticos, en una sociedad que depende de dispositivos inteligentes para realizar sus tareas comunes. Las empresas creadoras de tecnología y proveedoras de servicios inteligentes enfrentan una gran adversidad para mantener la seguridad en sus dispositivos.

2.5. Aspectos físicos en las redes inalámbricas

Cuando se habla de redes inalámbricas se tienen que considerar diversos factores que ayudan a lograr una mayor comprensión y enfoque en el estudio de las redes y las aplicaciones que se les podría dar. Una clasificación muy usual es la que se da en orden del tamaño de la célula.

- Macro célula.
- Micro célula.
- Pico célula.

Estas células se ven afectadas por diversos fenómenos de propagación en mayor o menor medida. Una de ellas es la atenuación debido a la distancia, en donde a mayor sea la distancia que hay en-

tre las antenas de los dispositivos, mayor será la atenuación de la potencia en la antena del receptor.

Existen varios modelos para la atenuación por distancia. Uno de ellos es el modelo clásico para el caso de la propagación en el espacio libre, aunque es poco preciso :

$$P_R = \frac{P_T}{d^2} \tag{2.1}$$

En donde

- P_R es la potencia recibida en el receptor.
- P_T es la potencia transmitida por el transmisor.
- d es la distancia entre las antenas.
- $n = 2$ es el factor de atenuación

Una ecuación más completa involucra parámetros de las antenas transmisoras y receptoras.

$$P_R = P_T G_T G_R (\frac{\lambda}{4\pi d})^2 \tag{2.2}$$

En donde

- G_T es la ganancia de la antena transmisora.
- G_R es la ganancia de la antena receptora.
- λ es la longitud de onda.

La ecuación (2.2) también es conocida como ecuación de Friis.

2.5.1. Indicador de fuerza de la señal recibida

El parámetro conocido como Indicador de fuerza de la señal recibida (Received signal strength indicator, RSSI por sus siglas en inglés) es una medida de las pérdidas que se presentan en una antena receptora respecto a una potencia de referencia. Considerando la ecuación (2.2), se tiene que el RSSI puede ser representado con la ecuación:

$$RSSI[dBm] = 10log_{10}[\frac{P_T}{P_{ref}}] \tag{2.3}$$

En donde la potencia de referencia es $P_{ref} = 1[mW]$.

2.5.2. Relación señal a ruido

La relación señal a ruido (Signal Noise Ratio, SNR por sus siglas en inglés) se define como el cociente del valor de la potencia de la señal deseada, es decir, la señal transmitida por el origen, sobre el valor de la potencia del ruido que genera el medio [18]. Este parámetro es expresado con la ecuación:

$$SNR[dB] = 10log_{10}(\frac{P_S}{P_N}) \tag{2.4}$$

Donde P_S es la potencia de la señal deseada recibida y P_N es la potencia del ruido introducido por el medio recibida en el receptor.

2.6. Características de LoRa Stick Lite

LoRa Stick Lite es un dispositivo usado para LPWAN (low power wide area networks) por lo que se ajusta a las características que debe tener un dispositivo de IoT. Para lograrlo, LoRa Stick Lite implementa LoRa (Long Range) y puede ser utilizado con un protocolo llamado LoRaWAN, el cual usa el estándar de LoRa Alliance [19]. Visto desde el modelo OSI, LoRaWAN está orientado a las capas física y MAC mientras que LoRa únicamente comprende la capa física. Algunos aspectos generales de LoRaWAN que se tienen en la capa física se muestran en la tabla (2.1).

Capa física	Cantidad
Estándar	LoRa Alliance
Modulación	CSS
Frecuencias	ISM : 433$[MHz]$, 868$[MHz]$, 915$[MHz]$
Anchos de banda	125$[KHz]$, 250$[KHz]$
Tasa de transmisión	Dependiente del ancho de banda y spreading factor
Consumo de energía	bajo

Tabla 2.1: Aspectos generales de LoRaWAN en capa física [20]

Como se puede observar en la tabla (2.1), LoRaWAN usa como modulación CSS (Chirp Spread Spectrum), con lo que se permite escoger entre diversos anchos de banda. En aspectos más generales, la modulación CSS funciona enviando cada símbolo en un ancho de banda de frecuen-

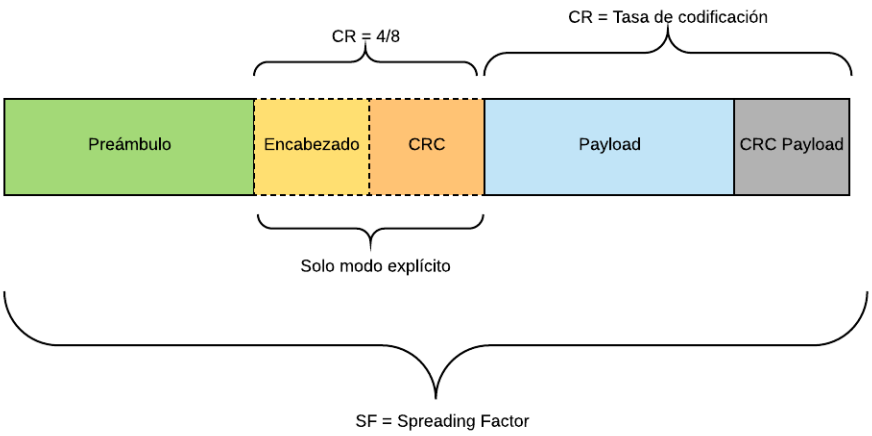

Figura 2.5: Encabezado de la paquetería de LoRaWan. **Figura elaborada por los autores de este trabajo**.

cia [21], [22]. LoRaWAN logra esta modulación, haciendo variar la cantidad de símbolos por frecuencia con un parámetro (spread factor). Donde un símbolo está asiganado a una frecuencia f_0 y esta varía en un rango de frecuencias (f_{min} y f_{max}), la frecuencia que se le asigna a un símbolo cambia de forma que se vaya abarcando el ancho de banda y se termina cuando regresa a la frecuencia inicial f_0.

2.6.1. Tiempo de transmisión de un frame

Un frame que se requiere enviar con LoRaWAN tiene la estructura mostrada en la figura (2.5). La longitud de todo este frame es lo que se considera como el spread factor (SF). Con lo que en base a esta estructura del frame, se puede realizar un cálculo teórico del tiempo de transmisión de un símbolo en LoRaWAN. La tasa de un símbolo está dada por ecuación (2.5), en donde se puede observar que depende del spread factor. Considerando que el empaquetado, tiene dos escenarios, implícito y explícito, este tendrá un impacto en el tiempo de transmisión.

$$R = \frac{BW}{2^{SF}} \tag{2.5}$$

En donde :

- BW es el ancho de banda
- R es la tasa del símbolo

- SF es el spread factor ($[7, 12]$)

El tiempo que tarda en transmitirse un paquete, se muestra en la ecuación (2.6). Representado por la suma del preámbulo y payload.

$$T_{frame} = T_{pre} + T_{payload} \tag{2.6}$$

En donde :

- T_{frame} es el tiempo por frame.
- T_{pre} es el tiempo por preámbulo.
- $T_{payload}$ es el tiempo por payload.

Para obtener la ecuación (2.6), es necesario obtener T_{pre} , el cual depende del tiempo por símbolo, calculado en la ecuación (2.7) para posteriormente sustituir en la ecuación (2.8).

$$T_{sym} = \frac{1}{R} \tag{2.7}$$

$$T_{pre} = (n_{pre} + 4{,}25) * T_{sym} \tag{2.8}$$

De las ecuaciones (2.7) y (2.8):

- R es la tasa de un símbolo.
- n_{pre} la longitud de los registros modem, LoRaWAN 1.0 lo define como 8.

Finalmente, el tiempo del payload, depende proporcionalmente del tamaño de del payload ($n_{payload}$) y del tiempo por símbolo T_{sym}, como se muestra en la ecuación (2.9).

$$T_{payload} = n_{payload} * T_{sym} \tag{2.9}$$

Para obtener el tamaño del payload,se calcula usando la ecuación (2.10), en donde :

- PL longitud del paquete en bytes.
- IH Header implicito, varía entre 0 y 1 según esté habilitado.
- DE Define si usa una baja tasa de datos, varía entre 0 y 1 según esté habilitado.

- CR Tasa de codificación, $1 : 4/5 : 4/8$.

$$n_{payload} = 8 + max(ceil[\frac{8PL - 4SF + 28 + 16CRC - 20IH}{4(SF - 2DE)}](CR + 4), 0) \tag{2.10}$$

La función ceil() recibe valores en bits, por lo que se hacen las multiplicaciones adecuadas para los parámetros que recibe. Esta función regresa el número de nibbles desde que se divide por SF. Si se activa IH, se le agregarán 20 bits al paquete, poniéndose en modo explícito.

Capítulo 3

Estado del arte

3.1. Introducción

Con el propósito de tener una profundización de antecedentes en la detección de "Sybil Attack", se realizó una investigación de artículos publicados por diferentes autores que abordan el tema de la detección de esta clase de ataque en redes con diferentes características, utilizando diversos métodos para hallar una solución a esta problemática; para así contar con una base teórica mucho más solida para el desarrollo de este trabajo.

Los artículos que fueron recopilados para la elaboración de este capítulo fueron divididos en tres categorías, donde cada una de ellas comparte características únicas para dar solución a la detección de identidades falsas generadas por un "Sybil Attack" en diferentes tipos de redes.

3.2. Soluciones basadas en un parámetro físico

Miden parámetros físicos que se encuentran incluidos en el envío y recepción de paquetes en una red, independientemente si el medio de propagación es cableado o inalámbrico. Entre los parámetros más utilizados se encuentran: RSSI, SNR, potencia de recepción y tasa de transmisión

Para este tipo de soluciones, la técnica más simple es la identificación de los nodos por medio de la distancia. De esta forma, si dos nodos vecinos están a una distancia muy similar, estos nodos podrían asumirse como uno mismo. En [23] se presenta una propuesta basada en la medición de RSSI que permite estimar la localización de los nodos que pertenecen a la red, de modo que si existen dos o más nodos que no cumplen con cierto margen de posición serán etiquetados como

identidades falsas. Por otra parte, la propuesta descrita en [24] incluye una etapa de pruebas de ubicación que vienen ligadas con ciertas etiquetas de posición, por lo que se requiere que las unidades móviles cuenten con receptor GPS, sistema que realiza operaciones satélites para triangular una posición, dichas operaciones están fuertemente ligadas a los parámetros físicos en las señales de recepción. Por último, en [14] se describen 5 diferentes esquemas de detección, en el que uno de ellos propone que un nodo verifique las posiciones de sus vecinos, sin embargo, se especifica que el esquema sólo es válido cuando la red no es móvil.

3.3. Soluciones basadas en una prueba de trabajo.

Usan desafíos o pruebas de trabajo con el fin de comprobar la identidad de un vecindario u obstaculizar la respuesta de entidades maliciosas.

Dentro de este tipo de soluciones se plantea un esquema básico que consiste en que una entidad genuina dentro de la red elabore pruebas que requieran de cierto procesamiento, todo esto con la intención de dar una carga de trabajo lo suficientemente grande para detectar si un nodo es malicioso o no. Los autores en [24] plantean un esquema de detección en donde se incluye la solicitud de pruebas de trabajo, pruebas que deben realizarse en un tiempo previamente estimado para concederle cierta etiqueta a un vehículo; un vehículo no se considerará genuino dentro de la red hasta que cumpla con cierto número de etiquetas. Por otra parte, el esquema de detección propuesto en [25] utiliza las pruebas de trabajo con el objetivo de que un nodo malicioso no pueda intervenir en las votaciones debido a una saturación de recursos que es producto de realizar múltiples pruebas de trabajo.

3.4. Soluciones basadas en consenso o votación.

Utilizan un sistema de votación, en donde los nodos que participan en dicho sistema, argumentan su respuesta basándose en el comportamiento de un nodo o nodos que son previamente señalados. Esto se realiza con el objetivo de poder detectar nodos maliciosos dentro de la red, tomando como base el criterio del vecindario.

Para este tipo de solución se identificaron dos artículos que toman el fundamento teórico antes mencionado para poder determinar si un nodo es Sybil o no; el primero de ellos está en el esquema planteado en [14], en donde se genera un registro de todos las identidades presentes en la red

mediante una autoridad central y así diseminar este registro en toda la red. Por otra parte, en [26] plantean la detección de un "Sybil Attack" mediante un consenso, en donde se evalúa si el comportamiento de ciertos nodos es fuera de lo normal, dicho esquema de detección está diseñado para evitar la participación de nodos maliciosos en dicha votación, ya que para poder participar en dicho esquema se debe ser seleccionado de forma aleatoria, en donde la prioridad la tienen los nodos con un mejor nivel de reputación.

3.5. Soluciones que no entraron en alguna categoría

A pesar de la categorización utilizada para segmentar los artículos analizados en el desarrollo de este capítulo, existieron dos excepciones, es decir, un par de esquemas de detección propuestos que no entraron en ninguna de las tres categorías, sin embargo, su contenido tuvo una gran aportación para el desarrollo de este trabajo.

Los autores de [27] y [28] plantean esquemas de detección que no fue posible relacionar con alguna de las categorías mencionadas en este capítulo, el primero de ellos, basa su éxito en la evaluación del comportamiento de los nodos de toda la red, mientras que el segundo utiliza un modelo basado en firmas en donde se pone a prueba una de las grandes propiedades de las funciones hash.

3.6. Conclusiones

La recopilación e investigación de los documentos mencionados en este capítulo aportaron herramientas imprescindibles para el desarrollo de este proyecto dado que se analizaron ventajas, desventajas, errores y aciertos de los esquemas de detección y mitigación de "Sybil Attack" que plantearon diferentes autores, todo lo anterior con el fin de robustecer el algoritmo de detección del ataque antes mencionado.

En el artículo [14] se propone el esquema de prueba de recursos de radio (radio resource testing), en el que los dispositivos en una red deben responder un desafío por un canal asignado por el nodo que emite el desafío. Mostrando que el concepto de desafíos puede ser aplicado para la detección de identidades Sybiles, incluso si se tuvieran pocas identidades como Sybiles. Debido a que múltiples identidades Sybiles pueden vivir en un mismo nodo maligno (quien las esté generando) y este mismo no podría responder desafíos para todas las identidades. Por otro lado este esquema puede

ser roto si el nodo malicioso tiene la capacidad de responder más de un desafío a la vez. En el caso del esquema propuesto en [14], para que esto ocurriera, un atacante debería tener un radio capaz de transmitir en más de un canal a la vez (o en su defecto, múltiples radios).

El artículo [23] plantea un esquema de detección en una red de sensores apoyado en la lectura del parámetro RSSI. Durante su desarrollo, los autores lograron demostrar su inestabilidad, ya que la desviación estándar de una muestra de N paquetes recibidos en un sensor receptor fue demasiado grande. El artículo presenta en sus resultados un comportamiento muy lejano a una campana Gaussiana. Sin embargo, demuestra que la variable puede ser utilizada en un esquema de detección si se obtiene la diferencia de un par de lecturas. Por otra parte, las características de este esquema de detección provocan una limitante para los tipos de redes donde puede ser implementado. Por ejemplo, una red móvil es un escenario complejo para una comparativa de parámetros físicos debido a la varianza de las mediciones asociada al movimiento de los nodos involucrados. Sin embargo, para una red estática como es una red de sensores, representa un escenario ideal para una implementación de este tipo, ya que las mediciones de los parámetros físicos son idealmente comparativas. Por otro lado, un nodo malicioso pudiera tener la habilidad de generar y enviar paquetes con diferentes potencias, lo que requeriría de un algoritmo complejo y varias identidades genuinas para detectar la posición exacta de un nodo, tal como se realiza en el articulo [23].

Los artículos [24] y [25] presentan una solución a la detección de un “Sybil Attack” realizando pruebas de trabajo, en el primer artículo una autoridad confiable previamente desplegada sobre la red es la que asigna y verifica la solución proporcionada por los nodos. En el segundo artículo las identidades en la red realizan sus pruebas de trabajo a partir de una constante previamente obtenida de forma aleatoria y así tener un sistema de votaciones seguras, evitando un “Sybil Attack”. Estos esquemas prueban ser muy efectivos para detectar y mitigar un “Sybil Attack”, sin embargo, se requiere una gran capacidad de procesamiento en los dispositivos adicional a una serie de condiciones, como contar con una red homogénea. Debido a que si nodos maliciosos logran tener una capacidad de procesamiento mayor al del resto de la red, estos nodos maliciosos pudieran calcular pruebas por adelantado.

En [26], los autores utilizan un sistema de reputaciones basado en el éxito obtenido de un consenso, de tal forma que maliciosos no logran hacerse del control de dichos consensos. Bajo el supuesto de que un atacante busca frustrar consensos para evitar que transacciones sean completa-

das, este logre ser penalizado mediante una baja reputación y por lo tanto sea menos elegible para futuros consensos. Sin embargo no garantiza que atacantes Sybiles no se hagan del control de primeros consensos, especialmente si se considera una probabilidad uniforme para cada participante de ser elegido.

El artículo [28] utiliza un esquema de firmas para determinar si un nodo falsifica una identidad de la red, lo anterior a partir del concepto de que la salida de una función hash, así como la creación de una llave privada puede tener diferentes entradas o llaves públicas. Con el concepto anterior, los nodos maliciosos pueden llegar a calcular una entrada que no corresponde a la firma del nodo genuino y así ser detectados automáticamente.

El esquema de detección propuesto en [27] utiliza el monitoreo de comportamiento como elemento principal para detectar un nodo malicioso de un "Sybil Attack", por lo que su eficiencia únicamente aplica bajo la premisa de que la intención del atacante es ganar las recompensas por minar bloques.

Los resultados obtenidos del análisis de los artículos mencionados en este capítulo muestran que las soluciones basadas en parámetros físicos, como es el SNR y RSSI, pueden tener un margen de error considerablemente grande si no se les hace un tratamiento adecuado.

Los esquemas de detección basados en pruebas de trabajo presentan una fiabilidad sumamente alta además de que pueden ser implementados en un mayor número de tipos de redes, sin embargo, el costo del procesamiento y consumo de recursos puede llegar a ser muy alto. Por ello, un nodo genuino que requiera validar su identidad deberá concentrar muchos de sus recursos, lo que representa una limitante importante en redes que tengan un bajo nivel de procesamiento o que se encuentren energizados a través de baterías, como sería en el caso de IoT.

Para la categoría de esquemas de detección basadas en consenso, los resultados demuestran que la efectividad de este tipo de implementaciones depende directamente del criterio que toman los nodos al momento de tomar una decisión para el consenso. Por ello, la implementación de una solución de este tipo requiere conocer el objetivo de la red para establecer un criterio basado en el comportamiento de los nodos o en su defecto que este pueda ser complementado con otro de los tipos de esquemas mencionados anteriormente. Adicionalmente, el número de nodos genuinos que participan dentro de la red representa un parámetro importante para la efectividad del algoritmo, pues el atacante al poder replicar múltiples identidades dentro de la red podría afectar en su totalidad los resultados del consenso.

Capítulo 4

Desarrollo

4.1. Introducción

"Sybil Attack" es uno de los ataques más perjudiciales en redes descentralizadas, por lo que la importancia de definir todos los elementos del esquema de detección constituirán la base necesaria para que este trabajo cumpla su objetivo de sembrar un antecedente para el diseño de esquemas de detección rápidos y eficientes en cualquier tipo de red inalámbrica, inclusive, para cualquier tipo de ataque que aproveche las vulnerabilidades del enrutamiento de paquetes.

En este capítulo se presenta el esquema de detección diseñado a partir de la investigación previa que fue explicada en los capítulos de sustento teórico y estado del arte. Esta presentación incluye los algoritmos diseñados para cada una de las fases de este esquema, así como la estructura de los empaquetados de los mensajes que se intercambiaron durante la ejecución.

4.2. Descripción del esquema

Como se ha visto en el capítulo anterior, existen diversas metodologías y esquemas con los que se pueden detectar, evitar o mitigar la participación de nodos maliciosos que pretenden diseminar identidades falsas en este tipo de redes. Éstas metodologías se basan en tres conceptos: detección de posición, pruebas de trabajo PoW y consenso.

El esquema desarrollado en esta tesis contiene una hibridación de estos conceptos para la detección de identidades falsas presentes en un Sybil Attack. Por consiguiente, el esquema propuesto está compuesto por tres fases que contemplan los puntos mencionados anteriormente:

- Fase 1: Detección de posibles identidades falsas por medio de la estimación de su posición.
- Fase 2: Validación de la identidad mediante una prueba de trabajo (PoW).
- Fase 3: Consenso de aquellos nodos que no aprobaron las fases 1 y 2.

Las fases anteriormente listadas llevan un estricto orden que fue determinado con el fin de disminuir el costo computacional y de recursos asociados. Adicionalmente, el esquema de detección finaliza su ejecución con el resultado de la fase 3. En este punto, el esquema determina cuáles identidades están siendo suplantadas. El diagrama de la figura 4.1 presenta las fases que se llevan a cabo en el esquema de detección, en dicha figura es posible apreciar como el inicio de cualquier fase depende del resultado de la fase anterior.

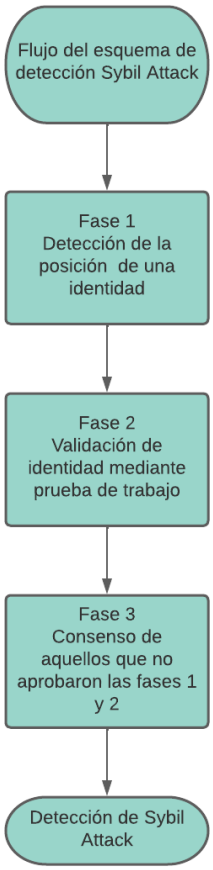

Figura 4.1: Fases que se llevan a cabo dentro del esquema de detección.**Figura elaborada por los autores de este trabajo**.

4.3. Descripción del algoritmo

4.3.1. Algoritmo principal

En el algoritmo (1) se muestra el esquema general de la propuesta de detección de identidades falsas. Es importante mencionar que la lista de identidades falsas o *ListaOscura* es inicializada como un conjunto vacío, es decir, ninguna identidad está dentro de la categoría de sospechoso cuando el algoritmo inicia su proceso. Posterior a la inicialización, se ejecuta un algoritmo al que se le denominará *Algortimo RSSI*, el cual tendrá como salida los subconjuntos en donde se encuentran segmentados las identidades que adquieren la categoría de nodos sospechosos, dicha segmentación es igualmente realizada por el algoritmo *Algoritmo RSSI*, el listado resultante del algoritmo antes mencionado quedará almacenado en la *ListaGris*.

Para cada uno de los subconjuntos de la *ListaGris* se enviarán pruebas de trabajo en base a un algoritmo que lleva como nombre *Algoritmo PoW*, dicho algoritmo volverá a evaluar el estado de sospecha de las identidades de la *ListaGris*, lo que dará como origen una *ListaOscura*.

Una vez que se obtiene el listado de sospechosos que no superaron las dos primeras pruebas, se enviará un mensaje de consenso que incluirá las identidades de la *ListaOscura*. Con el objetivo de que las identidades genuinas puedan proporcionar su posición en esta fase.

Algoritmo 1: Algoritmo principal.

Result: Conjunto de identidades falsas

$ListaOscura \leftarrow \emptyset$;

$ListaGris \leftarrow AlgRSSI()$;

for $Group \in ListaGris$ **do**

  $ListaOscura = ListaOscura \cup AlgPoW(\text{Group})$;

end

for $Falso_{id} \in ListaOscura$ **do**

  Enviar mensaje_consenso($Falso_{id}$);

end

return $Consenso()$;

El código mostrado en el apéndice (A), muestra la implementación del algoritmo (1). Este código es ejecutado en arduino, de forma que se tuvo que agregar la biblioteca de Heltec con el objetivo de poder comunicar esta plataforma con las tarjetas LoRa Lite Stick.

El algoritmo (2) describe el procedimiento que realiza cada uno de los dispositivos con identi-

dades genuinas que pertenecen a la red con el fin de recolectar la información necesaria para que puedan ejecutar los algoritmos de detección de Sybil Attack. Cada vez que un dispositivo recibe un mensaje de cualquier tipo, independientemente de que sea el destino del paquete o no, obtiene el valor del RSSI del mismo, al igual que el identificador asociado a la fuente del mensaje, es decir, la identidad del dispositivo que transmitió el paquete. La implementación de este algoritmo se muestra en el código del apéndice (A). Para poder implementar este código se requirió del diseño de estructuras de mensajes y así poder tener comunicación entre nodos. La estructura cambia entre cada tipo de mensaje, de forma que se tienen los siguientes tipos de mensajes :

- Tipo 0 : Este tipo representa un mensaje genérico en la red y su empaquetado está estructurado como se muestra en la figura (4.2).
- Tipo 1: Representa un mensaje de solicitud de PoW y su empaquetado está estructurado como se muestra en la figura (4.3).
- Tipo 2: Representa un mensaje de respuesta de PoW y su empaquetado está estructurado como se muestra en la figura (4.4).
- Tipo 3: Representa un mensaje de consenso y su empaquetado está estructurado como se muestra en la figura (4.5).

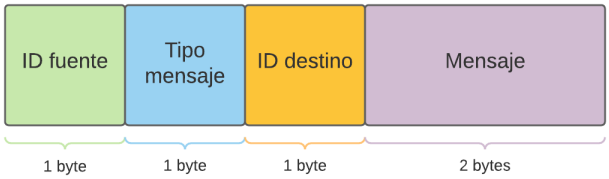

Figura 4.2: Empaquetado mensaje tipo 0. **Figura elaborada por los autores de este trabajo**.

Figura 4.3: Empaquetado mensaje tipo 1. **Figura elaborada por los autores de este trabajo**.

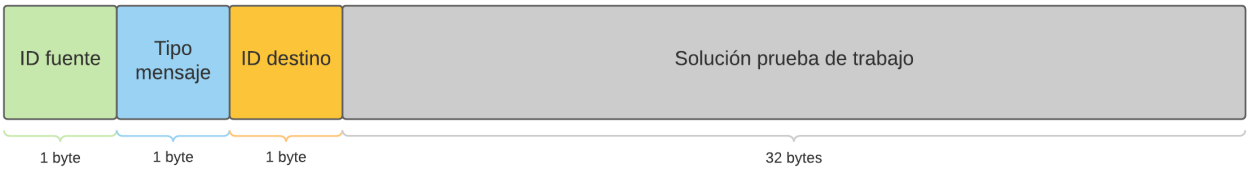

Figura 4.4: Empaquetado mensaje tipo 2. **Figura elaborada por los autores de este trabajo**.

Figura 4.5: Empaquetado mensaje tipo 3. **Figura elaborada por los autores de este trabajo**

El dispositivo realiza una consulta en su base de datos para validar si se tiene registro de la identidad que mandó el mensaje, es decir, se verifica si el identificador ya se encuentra dentro del historial de mensajes recibidos. En caso de que así sea, se almacenarán sólo los últimos 10 valores de RSSI recibidos de dicha identidad en el registro.

Algoritmo 2: Algoritmo de almacenamiento en base de datos.

Result: Algoritmo sin salida

if $MensajeRecibido() = True$ **then**
- $rssi \leftarrow RSSI\_paquete()$;
- $id \leftarrow ID\_paquete()$;
- **if** $RSSI_{id} \in RSSI$ **then**
  - **if** $|RSSI_{id}| \geq 10$ **then**
    - $RSSI_{id} \leftarrow RSSI_{id} - \{Older(RSSI_{id})\}$;
  - **end**
- **else**
  - $RSSI \leftarrow RSSI \cup RSSI_{id}$;
- **end**
- $RSSI_{id} \leftarrow RSSI_{id} \cup rssi$;

end

4.3.2. Fase 1: RSSI

El algoritmo (3) describe el procedimiento que se lleva a cabo durante la primera fase de detección de identidades falsas. Para que esta fase pueda llevarse a cabo, es necesario obtener un arreglo de promedios y desviaciones estándar, por lo que ambas listas se inicializan con valor nulo, es decir, como conjunto vacío, debe de existir por lo menos una identidad que contenga diez mensajes dentro de la base de datos del nodo. Para cada identidad que cumpla con la característica anterior en cuanto a número de RSSI guardados en el historial, se obtendrá su promedio de dicho valor y la desviación estándar.

Algoritmo 3: Algoritmo fase RSSI

Result: Conjunto de conjuntos de nodos sospechosos

$Promedios \leftarrow \emptyset$;
$desv \leftarrow \emptyset$;
for $RSSI_{id} \in RSSI$ **do**
  if $|RSSI_{id}| \geq 10$ **then**
    $Promedios \leftarrow Promedios \cup Promedio(RSSI_{id})$;
    $desv \leftarrow desv \cup desv(RSSI_{id})$;
  end
end
for $Promedios_{id} \in Promedios$ **do**
  $ListaGris_a \leftarrow \emptyset$;
  for $Promedios_{id'} \in Promedios - \{Promedios_{id}\}$ **do**
    if $Promedios_{id} - desv_{id} \leq Promedios_{id'} \leq Promedios_{id} + desv_{id}$ **then**
      $ListaGris_a \leftarrow ListaGris_a \cup \{id', id\}$;
    end
  end
  $ListaGris \leftarrow ListaGris \cup \{ListaGris_a\}$;
end
return $ListaGris$;

Con los promedios obtenidos, se recorre la lista antes mencionada en busca de dos promedios que su diferencia sea menor que cierto margen dependiente de la desviación estándar. Una vez que se encuentran dos valores que cumplan con la condición antes mencionada, ambos identificadores asociados a dichos promedios serán agregados a una lista de nombre *ListaGris*. El procedimiento anterior se realiza con todos los elementos de la lista de promedios compárandose con todos los valores que de igual forma conforman dicha lista.

La implementación de este algoritmo se encuentra en el apéndice (B). El mensaje de tipo 0, el cual su estructura puede observarse en la figura (4.2) es el usado en esta fase, recordando que se almacenan RSSI y el ID del mensaje recibido, por lo que no es necesario mandar alguna información en especifico y basta con transmitir un mensaje genérico.

4.3.3. Fase 2: Pruebas de trabajo

El algoritmo (4) describe el proceso que debe llevar a cabo un dispositivo para crear n pruebas de trabajo y así poder inicializar la segunda fase del esquema de detección híbrido.

Para que el proceso pueda llevarse a cabo, es necesario apoyarse de ciertos conjuntos que son inicializados en cero, y tienen como objetivo almacenar tiempos de ejecución de pruebas de trabajo y las soluciones a dichas pruebas; con la inicialización anterior, el nodo que genera los desafíos debe generar una cadena aleatoria. Para cada identidad que pertenece al subconjunto de sospechosos, se asigna una semilla como entrada a la prueba de trabajo, dicha semilla es producto de la concatenación de la cadena aleatoria y el identificador que resultó ser sospechoso. Con la información anterior, el nodo que genera las pruebas debe medir el tiempo que le lleva a cabo realizar la prueba de tabajo y almacenar dicho valor en uno de los conjuntos que fue inicializado con el algoritmo, así como la solución del desafío.

Algoritmo 4: Algoritmo creación de PoW básico

Result: Conjunto de identidades sospechosos

$(Tiempo \leftarrow \emptyset;$

$Respuesta \leftarrow \emptyset;$

$ID = \emptyset;$

$cadena = Random();$

for $id \in Group$ **do**

  $semilla \leftarrow cadena + id;$

  $tiempo\_inicial \leftarrow Time();$

  $respuesta \leftarrow PoW(semilla);$

  $tiempo\_final \leftarrow Time();$

  $tiempo \leftarrow tiempo\_final - tiempo\_inicial;$

  $Tiempo \leftarrow Tiempo \cup tiempo;$

  $Respuesta \leftarrow Respuesta \cup respuesta;$

  $ID \leftarrow ID \cup id;$

end

$Mensaje\_desafio(ID, cadena);$

$tiempo\_desafio \leftarrow Time();$

Una vez registrada la solución y su tiempo de elaboración para cada desafío, el nodo genuino deberá enviar por mensaje la lista de identificadores que deberán responder el desafío y la cadena

que deben usar como semilla. Por otra parte, el nodo que elabora la fase 2 debe empezar a tomar el tiempo de resolución de desafío una vez que envía el mensaje.

La implementación de este algoritmo está en el código del apéndice (B). Se usa el mensaje de tipo 1, cuya estructura de muestra en la figura (4.3), que tiene como mensaje el envío del conjunto para el cual se generó la prueba de trabajo y un número aleatorio de 4 dígitos. Gracias a esta información, es posible que cualquier nodo en la red pueda ejecutar la prueba de trabajo y responder en un tiempo igual al estimado por el nodo honesto.

El algoritmo descrito en el pseudocódigo (5) describe una versión optimizada del algoritmo (4), el cual crea las pruebas de trabajo necesarias para un grupo de identidades sospechosas y evita colisiones durante la recepción de resultados en caso de que existan dos o más nodos genuinos dentro de la lista de sospechosos.

Algoritmo 5: Algoritmo creación de PoW optimizado

Result: Conjunto de nodos sospechosos

$Tiempo \leftarrow \emptyset$;

$Respuesta \leftarrow \emptyset$;

$ID \leftarrow \emptyset$;

$cadena \leftarrow Random()$;

while $Group \neq \emptyset$ **do**

  $semilla \leftarrow cadena + id$;

  $tiempo\_inicial \leftarrow Time()$;

  $respuesta \leftarrow PoW(semilla)$;

  $tiempo\_final \leftarrow Time()$;

  $tiempo \leftarrow tiempo\_final - tiempo\_inicial$;

  if $\nexists i, j \in Tiempo \mid i - j < margen\_tiempo$ **then**

    $Tiempo \leftarrow Tiempo \cup tiempo$;

    $Respuesta \leftarrow Respuesta \cup respuesta$;

    $ID \leftarrow ID \cup id$;

  end

end

$Mensaje\_desafio(ID, cadena)$;

$tiempo\_desafio = Time()$;

Para que esta versión del algoritmo de creación de pruebas de trabajo puede llevarse a cabo, es

necesario inicializar tres conjuntos vacíos, mismos que servirán de apoyo para el nodo en la entrega de resultados, los conjuntos antes mencionados llevan como nombre Tiempo, Respuesta e ID. Una vez terminada la inicialización de conjuntos, el nodo genera una cadena pseudoaleatoria.

Como siguiente paso, para cada identidad que se encuentre dentro de la lista de sospechosos, se crea la semilla para la prueba de trabajo, dicha semilla se compone de la cadena y del identificador de la identidad sospechosa, posteriormente se realiza la prueba de trabajo y el nodo genuino toma el tiempo que le toma llevar a cabo la prueba de trabajo. Una vez que obtiene el tiempo, analiza en el conjunto existente de tiempos si es posible que la prueba sea respondida evitando colisión, es decir, que el tiempo que tarde en realizar la prueba un nodo genuino este fuera de cierto margen de colisión con el resto de los tiempos del conjunto; si esta condición se cumple, se añade el tiempo, respuesta e identificador a su conjunto correspondiente.

Una vez que se hayan finalizado todas las pruebas de trabajo, el nodo genuino envía el mensaje de desafío e inicializa el tiempo de respuesta.

4.3.4. Algoritmo Recolección y Validación de Pruebas de trabajo o PoW

El algoritmo (6) forma parte de la fase 2 del esquema de detección planteado en la figura (4.1). Éste describe el proceso que lleva a cabo el nodo genuino que generó las pruebas de trabajo. El algoritmo (6) tiene como objetivo recolectar las respuestas de pruebas de trabajo de las identidades que fueron incluidas en la lista de sospechosos.

Algoritmo 6: Recolección respuesta PoW

Result: Conjunto de conjuntos de variables en respuesta a PoW

if $MensajeRecibidoPoW() = True$ **then**

- $marca\_t \leftarrow Time()$;
- $r\_pow, id \leftarrow ExtraerInformacion()$;
- $PoW \leftarrow PoW \cup (r\_pow, id, marca\_t)$;

end

El proceso que lleva a cabo el algoritmo se realiza cada vez que se recibe un mensaje de tipo respuesta de prueba de trabajo; en caso de que se reciba un mensaje con las características antes mencionadas, el nodo genuino toma el tiempo en que recibe el mensaje. Posteriormente extrae del paquete recibido la respuesta al desafío e identificador fuente para almacenarlos en un conjunto que utilizará durante la validación de resultados.

El algoritmo (7) plantea el procedimiento que sigue el nodo genuino que realizó la recolección de las respuestas de la pruebas de trabajo, este algoritmo es ejecutado inmediatamente después de la ejecución del algoritmo de recolección (6).

El proceso que se ejecuta en este algoritmo realiza un recorrido en el conjunto PoW, el cual contiene la respuesta de la prueba de trabajo, la identidad que mando la respuesta y la marca de tiempo en la que se recibió. Para cada elemento de este conjunto se obtiene el tiempo adicional que tardo la identidad en responder el desafío, esto se logra restando la marca de tiempo de recepción con el tiempo que le tomo al nodo genuino realizar la prueba de trabajo; si el tiempo adicional no cumple con el margen previamente establecido (delimitado por el tiempo que se requiere para resolver la prueba de trabajo, el tiempo de transmisión, el tiempo de propagación y el tiempo necesario para evitar colisiones), la identidad que respondió a dicho desafío es añadido a un conjunto que representa la lista oscura.

Este algoritmo da como resultado un conjunto de identidades que no pasaron la fase de prueba de trabajo.

Algoritmo 7: Validación de PoW

Result: Conjunto de conjuntos de identidades sospechosas

$ListaOscura \leftarrow \emptyset$;

for $(r\_pow, id, marca\_t) \in PoW$ **do**

  $tiempo \leftarrow marca_t - tiempo\_desafio$;

  if $r\_pow = Respuesta_{id} \& tiempo > Tiempo_{id} + T_{extra}$ **then**

    $ListaOscura \leftarrow ListaOscura - id$;

  end

end

return $ListaOscura$;

Para la implementación de los algoritmos (6) y (7), se realizó en el código mostrado en el apéndice (B). En la figura (4.4) se muestra el mensaje de tipo 2, dicho contiene los primeros 32 dígitos del hash obtenido por el nodo que realizó la prueba. Esto permite al nodo honesto comparar de forma sencilla con la hash obtenida cuando generó la prueba.

4.3.5. Fase 3: Consenso

El algoritmo (8) describe el procedimiento que sigue un nodo en el inicio de la fase 3, el cual consiste en recolectar la *ListaOscura* de nodos genuinos vecinos (los no incluidos en la *ListaOscura*).

El objetivo consiste en obtener conjuntos de identidades con las que el nodo genuino comprare con su *ListaOscura* y así pueda realizar un consenso en base a la opinión de nodos genuinos. El proceso empieza cuando se recibe una *ListaOscura*, es entonces cuando el nodo genuino almacena las identidades recibidas en ID_s.

El siguiente paso en el algoritmo descrito por (8) es recorrer el registro ID_s y agregar cada identidad a un conjunto de identidades llamado $Consenso_{id}$. Dicho conjunto sirve para realizar la comparación contra las identidades de la *ListaOscura* generada en la fase 2 por el nodo genuino.

Algoritmo 8: Recolección de identidades del consenso

Result: Conjunto de identidades para el consenso

$Consenso_{id} \leftarrow \emptyset$;

$ID_s \leftarrow RecibeConsensoID()$;

for $id \in ID_s$ **do**

  $Consenso_{id} \leftarrow Consenso_{id} \cup id$;

end

return $Consenso_{id}$;

El algoritmo (9) define la comparación que realiza un nodo genuino entre *ListaOscura* y $Consenso_{id}$. El proceso comienza cuando se recorre la lista oscura y por cada identidad se busca si pertenece al $Conjunto_{id}$. Las identidades que si fueron encontradas son agregadas son al conjunto ID_f que define las identidades falsas que fueron encontradas con el algoritmo.

Algoritmo 9: Consenso.

Result: Conjunto de identidades para el consenso

$ID_f \leftarrow \emptyset$;

for $id \in ListaOscura$ **do**

  if $id \in Consenso_{id}$ **then**

    $ID_f \leftarrow ID_f \cup id$;

  end

end

return ID_f;

La implementación de (9) y (8) se tiene en el apéndice (B). Para ello se generó una estructura de paquete como se muestra en la figura (4.5) correspondiente al mensaje de tipo 3, de forma que el mensaje contiene las listas negras generadas en la fase 2 del nodo que envíe el mensaje de consenso.

Capítulo 5

Resultados

5.1. Introducción

Con el objetivo de validar el funcionamiento del algoritmo diseñado y explicado en el capítulo del desarrollo, se llevaron a cabo experimentos en el que se simularon Sybil Attacks en un a red con características IoT, al exponer los resultados se sienta una base para posibles mejoras en el algoritmo, así como posibles integraciones para su uso en otros ambientes.

En el cuerpo de este capítulo se presentan los fundamentos, resultados y su interpretación de todos los experimentos realizados para construir el esquema de detección "Sybil Attack" para una red IoT, la información es presentada en tablas y gráficas comparativas que proporcionan el recurso necesario para llevar a cabo un análisis completo de la eficiencia de todas las etapas del esquema de detección.

5.2. Pruebas punto a punto para RSSI y SNR

Utilizando un par de dispositivos Heltec Wireless Stick Lite que son compatibles con el IDE de Arduino, y la biblioteca HELTEC ESP32 se realizó el envío de paquetes a diferentes distancias. La biblioteca antes mencionada permitió obtener el nivel de los parámetros físicos RSSI y SNR de los paquetes en recepción.

Los dispositivos y sus antenas fueron aseguradas con cinta, como se puede apreciar en la figura (5.1) con el objetivo de garantizar que las pruebas estuvieran sujetas a las mismas condiciones. Por otra parte, las computadoras que tomaron roles de transmisor y receptor de los mensajes fueron

Figura 5.1: Antena de la Wireless Stick Lite asegurada con cinta.

colocadas en una superficie plana, que se muestra en la figura (5.2), que facilitó que los dispositivos no se movieran durante el envío y recepción de mensajes. Para este experimento, se enviaron 1000

Figura 5.2: Superficie donde fueron realizadas las pruebas punto a punto.

paquetes con un SF de 7, un ancho de banda de 250kHz y un CR de 4/5 desde un dispositivo hacia el otro variando la distancia entre ellos desde 1 hasta 32 metros. La tabla (5.1) muestra los datos recopilados durante el experimento. La primera columna de la tabla muestra el promedio del RSSI de las 1000 transmisiones por experimento. La segunda columna, la desviación estándar de los 1000 RSSI obtenidos. La tercera y cuarta columna muestran también el promedio y la desviación estándar pero para el SNR.

Distancia[m]	**Promedio RSSI**	**Desviación Estándar RSSI**	**Promedio SNR**	**Desviación Estándar SNR**
1	-78.554	4.0208	15.486	2.2683
2	-76.198	0.9637	15.2372	2.319
4	-83.3463	1.0804	13.0385	4.2594
8	-91.5125	1.3518	15.521	1.8395
16*	-104.4334	2.1859	15.1071	2.1145
32*	-107.125	1.301	14.8437	2.6981

Tabla 5.1: Promedios y desviaciones estándar de RSSI y SNR. **Tabla elaborada por los autores de este trabajo con los resultados obtenidos**.

Las pruebas en 16 y 32 metros no entregaron el total de paquetes porque los porcentajes de paquetes recibidos fueron de 41.3 % y 2.4 % respectivamente.

Los resultados obtenidos en la tabla (5.1) reflejaron inconsistencia en ambos parámetros físicos, es decir, valores muy variables de desviación estándar cuando la distancia cambió. El comportamiento antes mencionado se asemeja con el presentado en [23], donde se realizaron experimentos que guardaron condiciones muy similares. En el artículo antes mencionado fue evaluado el parámetro RSSI, obteniendo un comportamiento muy impredecible del RSSI para una misma distancia.

La inconsistencia de los valores de RSSI y SNR durante las pruebas de punto a punto fueron atribuidas a un factor externo, ya que se observó cierta tendencia a una disminución del valor de los parámetros físicos a lo largo del flujo de paquetes, es decir, se observaron valores más bajos en la última serie de paquetes recibidos en una misma distancia, esto provocó que la desviación estándar tomara valores muy grandes. El efecto observado durante los experimentos fue atribuido al aumento de temperatura de las antenas durante la transmisión de paquetes; pues la transmisión de los mismos se realizó uno después del otro, por lo que el dispositivo en conjunto con su antena tuvieron lapsos de tiempo considerablemente grandes de operación, provocando un aumento de temperatura.

Con el fin de utilizar la información recabada y que ésta fuera de utilidad a pesar de las desviaciones estándar tan variables, se aplicó un procedimiento basado en el principio utilizado por los autores de [23], este principio fue la diferencia entre dos valores. Se aplicó la siguiente

metodología:

1. Dividir los 1000 valores de RSSI y SNR en dos secciones, una para representar los valores recibidos por un nodo A y la otra parte para el nodo B. Para dicha división, los valores pares fueron asignados al primer nodo y los valores impares para el segundo, asegurando que la distribución fuera de uno y uno.

2. Tomar las mediciones de diez en diez y calcular un promedio para cada dispositivo.

3. Obtener la diferencia entre los dos promedios (nodo A y nodo B), es decir, realizar una resta de estos y obtener el valor absoluto del resultado.

4. Almacenar el resultado en un archivo.

Una vez obtenidas todas las diferencias de RSSI y SNR de todos los promedios de las diferencias en todas las distancias, fueron procesados los resultados y documentados en las tablas (5.2) para RSSI y (5.3) para SNR.

Distancia[m]	**Promedio Diferencia RSSI**	**Desviación Estándar Diferencia RSSI**
1	0.1768	0.1593
2	0.0802	0.0882
4	0.1	0.1265
8	0.1263	0.1003
16*	0.1589	0.1511
32*	0.1667	0.0471

Tabla 5.2: Resultados de las diferencias de RSSI. **Tabla elaborada por los autores de este trabajo con los resultados obtenidos**.

Con los resultados obtenidos en las tablas (5.2) y (5.3) logró combatirse el fenómeno observado en los resultados de la tabla (5.1). Las diferencias entre muestra y muestra reflejaron la consistencia necesaria para poder seleccionar el parámetro físico de mayor utilidad para el esquema de detección. Para ambos parámetros físicos la desviación estándar tuvo mucho mayor consistencia cuando se varió la distancia. Con el nuevo análisis de resultados fue posible seleccionar el RSSI como referencia para el esquema de detección, debido a que a que con el procesamiento realizado, los valores de

Distancia[m]	Promedio Diferencia SNR	Desviación Estándar Diferencia SNR
1	0.7763	0.7193
2	0.7273	0.6876
4	0.8	1.2319
8	0.5513	0.58
16*	0.5642	0.5879
32*	0.5583	0.1829

Tabla 5.3: Resultados de las diferencias de SNR. **Tabla elaborada por los autores de este trabajo con los resultados obtenidos**.

desviación estándar fueron muy parecidos en todas las distancias, como puede observarse en la tabla (5.2).

5.3. Pruebas de desafío

Para el primer experimento de la segunda fase se utilizaron las tarjetas Wireless Stick Lite de la figura (5.3), el objetivo del experimento consistió en probar el algoritmo de desafió (PoW), el cual tuvo como objetivo implementar el algoritmo estándar SHA256 de forma retroalimentada a cierta entrada hasta obtener el resultado para una dificultad dada, todo lo anterior con la intención de obtener el tiempo promedio que le tomaría al dispositivo concluir con el proceso para responder a un desafío y así poder seleccionar la dificultad adecuada para el algoritmo de la segunda fase.

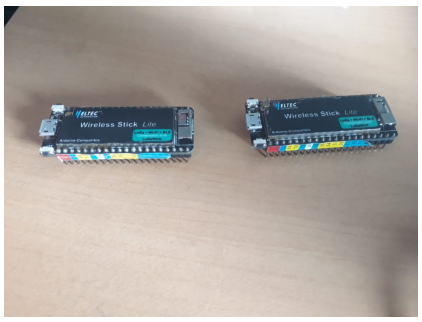

Figura 5.3: Dispositivos utilizados para las pruebas de trabajo. **Fotografía tomada por los autores de este trabajo**.

Las pruebas fueron realizadas de forma individual para dos dispositivos con las mismas carac-

terísticas, bajo los mismos niveles de dificultad y con la misma entrada. La replicación de escenario para ambas tarjetas se realizó con la intención de corroborar que las tarjetas al tener mismas características de procesamiento, realizaran la misma prueba en la misma cantidad de tiempo.

Fueron realizados 1000 desafíos para cada una de las tres dificultades, y el tiempo que le llevó a la tarjeta realizarlos fue documentado en un archivo de texto. Estos valores fueron procesados para obtener los resultados de la tabla (5.4).

Dificultad	**Tiempo promedio A [ms]**	**Tiempo promedio B [ms]**	**Desviación estándar**
1	7.526	7.526	7.2303
2	123.5	123.5	123.6955
3	1927.417	1927.417	1872.2804

Tabla 5.4: Resultados de los tiempos de procesamiento para concluir el desafío. **Tabla elaborada por los autores de este trabajo con los resultados obtenidos**.

Con los resultados registrados en la tabla (5.4) es posible concluir que ambas tarjetas realizaron cada uno de los 1000 desafíos en el mismo margen de tiempo. Adicionalmente, se observa la gran variación que existe en el tiempo de solución de desafíos al variar la dificultad. También se puede observar un comportamiento en el que la desviación estándar tuvo un valor similar al tiempo promedio. Dicho comportamiento se presentó en los tres niveles de dificultad.

Debido a los grandes valores de desviación estándar presentes en la tabla (5.4) fueron documentados los histogramas para los tres niveles de dificultad.

El histograma de la figura (5.4) representa los 1000 ensayos realizados por las tarjetas para una dificultad de nivel 1, en dicha figura logra observarse un comportamiento similar al Gaussiano en donde la mayor cantidad de ensayos se concentran entre los 0 y 10 milisegundos. Sin embargo, existió un porcentaje considerable de pruebas de trabajo que duraron más de 10 milisegundos. El histograma de la figura (5.5) representa los 1000 ensayos realizados por las tarjetas para una dificultad de nivel 2. En dicha figura logra observarse un comportamiento lejano a una campana Gaussiana, en donde los tiempos de pruebas de trabajo fueron mucho más variables, y a pesar de concentrar un porcentaje considerable de ensayos entre los valores 0 y 200 en milisegundos, existieron muchas pruebas que fueron realizadas en mucho mayor tiempo, presentando valores superiores a los 900 milisegundos.

El histograma de la figura (5.6) representa los 1000 ensayos realizados por las tarjetas para

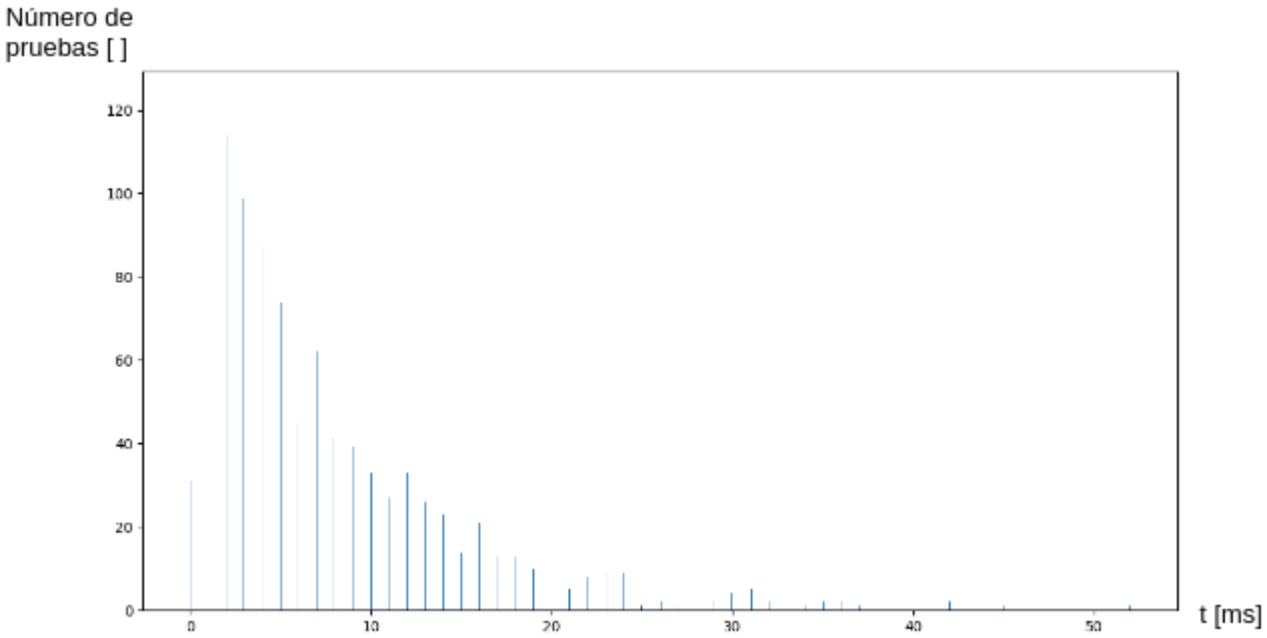

Figura 5.4: Histograma de tiempo en milisegundos para la dificultad 1. **Figura elaborada por los autores de este trabajo**.

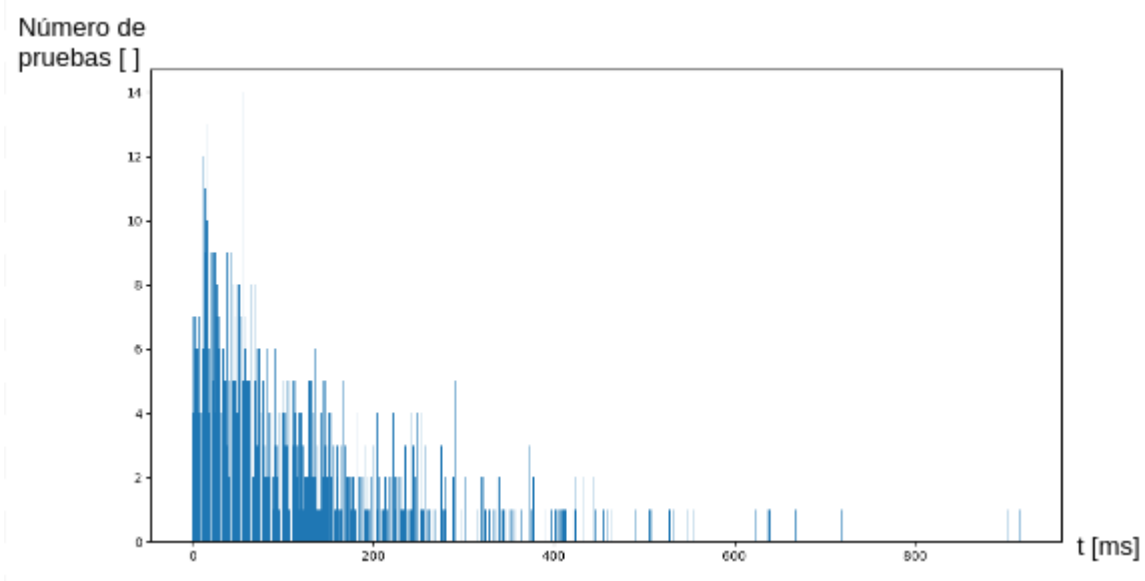

Figura 5.5: Histograma de tiempo en milisegundos para la dificultad 2. **Figura elaborada por los autores de este trabajo**.

una dificultad de nivel 3. En la figura se observa una distribución de tiempos bastante extensa en donde se concentraron la mayor parte de los desafíos entre 0 y 4000 milisegundos. A diferencia de la dificultad anterior, fuera de este rango de tiempo, los ensayos presentes fueron una gran minoría, pero a pesar de esto, logra observarse una gran varianza considerablemente mayor para este nivel de dificultad.

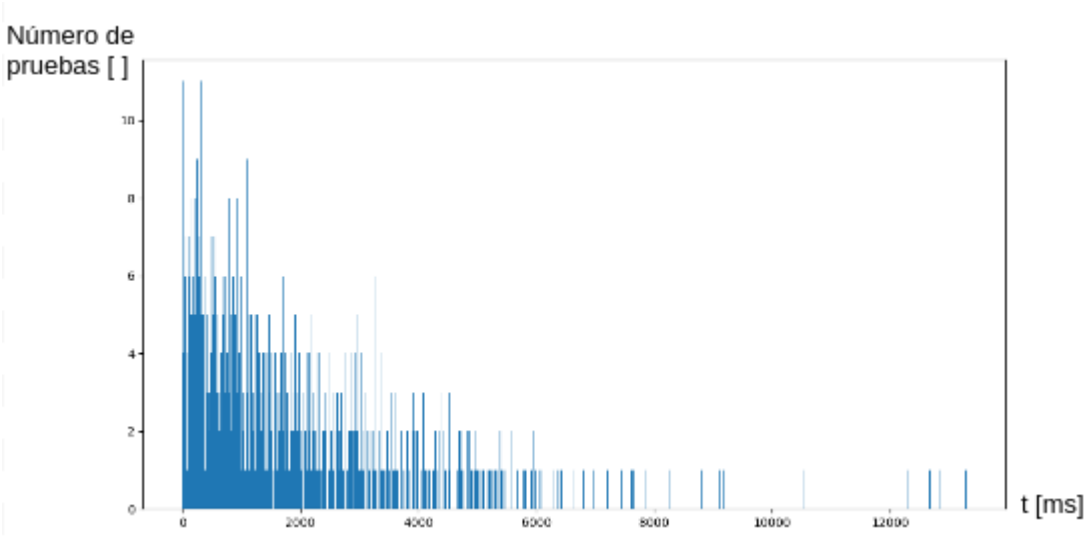

Figura 5.6: Histograma de tiempo en milisegundos para la dificultad 3. **Figura elaborada por los autores de este trabajo.**

La caracterización lograda con las figuras (5.4) a (5.6) permitió concluir que a mayor sea la dificultad del desafío, mayor será la varianza que se tendrá en el tiempo de resolución de los mismos, lo cual representa una ventaja en el esquema de detección, pues al tener un valor de varianza superior, se reducen las colisiones que pueden llegar a generarse en la segunda fase cuando dos o más nodos genuinos son catalogados como sospechosos responden sus respectivas pruebas de trabajo.

Las dificultades 2 y 3 pueden ajustarse, por su tiempo promedio y desviación estándar, a la segunda fase del esquema de detección, por lo que serán consideradas para la simulación del escenario Sybil Attack.

5.4. Algoritmo de búsqueda de identidades Sybil mediante RSSI

Los resultados obtenidos durante el primer experimento permitieron seleccionar el parámetro físico RSSI para la primera fase del esquema de detección. La fase descrita en el pseudocódigo (3) en conjunto con el envío de paquetes fueron programados en tarjetas Wireless Stick Lite utilizando la biblioteca HELTEC ESP32.

Para lograr replicar un escenario Sybil Attack, fue programada una segunda versión del algoritmo para esta fase de detección, esta versión tuvo como único objetivo la transmisión de paquetes con múltiples identidades, adoptando el comportamiento de un nodo atacante o nodo Sybil.

Utilizando ambas versiones del algoritmo, se simuló el ambiente descrito en la figura (5.7). En la figura (5.7), los 3 nodos de color azul representan a los nodos genuinos cargados con el programa que sigue el procedimiento descrito en el pseudocódigo (3). Por otro lado, el nodo de color rojo representa al nodo malicioso cargado con el programa que transmite paquetes con múltiples identidades. En este escenario, el nodo malicioso replicó 4 identidades.

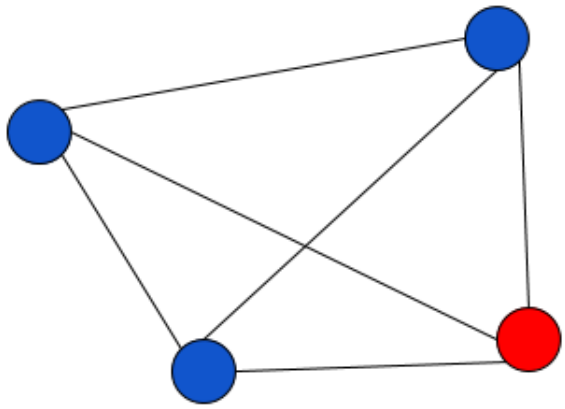

Figura 5.7: Red utilizada para el experimento de la primera fase del esquema de detección. **Figura elaborada por los autores de este trabajo**.

El algoritmo cargado en las tarjetas representadas en color azul en la figura (5.7) fue programado para obtener una lista con las identidades que no cumplieran con el criterio de desviación estándar multiplicada por un factor, es decir, una lista de las identidades sospechosas a la que se le denominó como *lista gris*. Con la intención de obtener un rango de valores de forma experi-

mental que permitieran hallar la mayor efectividad en esta fase, se varió la magnitud del margen de cumplimiento, el cual es definido como el rango de valores RSSI para los que se considera una identidad es falsa o genuina. La ecuación (5.1) describe como se genera este margen. De este modo dos mensajes con diferentes identificadores de fuente son considerados como mensajes enviados de un mismo nodo. De igual forma también se varió el número de muestras de RSSI que fueron tomados para obtener el promedio de los mismos y compararlos con un mensaje nuevo, todo lo anterior, como se describe en el algoritmo (3).

$$RSSI_i - F * \sigma < RSSI_j < RSSI_i + F * \sigma \tag{5.1}$$

En donde :

- $RSSI_i$ es el promedio de RSSI de la identidad con la que se compara
- $RSSI_j$ es el RSSI de la identidad comparada
- F Es un factor que multiplica las desviaciones estándar
- σ es la desviación estándar

Se realizaron combinaciones en función de diferentes valores de los dos parámetros antes mencionados. En cada una de estas combinaciones, Se obtuvieron 10000 mensajes recibidos por el nodo genuino y a partir de este se generaron listas grises.

Para poder evaluar la eficiencia de la primera fase del esquema de detección, los resultados fueron documentados y procesados para obtener 5 parámetros utilizados por los autores del artículo [27], los cuales permitieron medir la eficiencia de esta fase del esquema de detección. Los parámetros seleccionados fueron:

- Falsos positivos FPR.
- Falsos negativos FNR.
- Verdaderos positivos TPR.
- Verdaderos negativos TNR.
- Exactitud Acc.

El parámetro de falso negativo describe de forma porcentual las identidades Sybiles que fueron considerados como nodos genuinos durante la ejecución del algoritmo, mientras que el parámetro de falso positivo indica el porcentaje de nodos genuinos que fueron considerados como identidades Sybiles. Por otro lado, verdaderos positivos representa el porcentaje de identidades Sybiles que sí fueron encontradas en la lista gris y los verdaderos negativos son los nodos genuinos que no aparecen en esta lista. Finalmente, la efectividad se define de acuerdo con [27] utilizando la ecuación (5.2).

$$Exactitud = \frac{S_D + G_C}{N} \tag{5.2}$$

Donde:

Exactitud: efectividad

S_D: Número de identidades Sybiles detectadas.

G_C: Número de nodos genuinos detectados.

N: Total de identidades en la red.

Los parámetros descritos anteriormente fueron procesados para cada una de las variaciones de los experimentos. Los resultados obtenidos de dicho proceso fueron documentadas en las tablas (5.5) a (5.7).

En las tablas antes mencionadas se muestran los niveles porcentuales de falsos negativos, falsos positivos, verdaderos positivos, falsos positivos y exactitud. La tabla (5.5) muestra los resultados de los experimentos realizados para un promedio comparativo de 10 RSSI para diferentes factores de desviación estándar. La tabla (5.6) representa los resultados para un promedio comparativo de 20 RSSI y la tabla (5.7) para un promedio de 5 RSSI para la comparación de paquetes recibidos.

Factor Desviación Estándar	FNR[%]	FPR[%]	TNR[%]	TPR [%]	Exactitud[%]
0.25	47.2	22.3	77.7	52.8	67.03
0.5	38.7	29.3	70.7	61.3	66.66
1	36.2	39.6	60.4	63.8	61.85
2	38.7	51.7	48.3	61.3	53.89
3	36.1	69.7	30.3	63.9	44.69
4	27.7	84.6	15.4	72.3	39.78

Tabla 5.5: Niveles porcentuales de efectividad para un promedio de 10 RSSI. **Tabla elaborada por los autores de este trabajo con los resultados obtenidos**.

Factor Desviación Estándar	FNR[%]	FPR[%]	TNR[%]	TPR [%]	Exactitud[%]
0.25	39.1	20.9	79.1	60.9	71.29
0.5	32.1	30.5	69.5	67.9	68.82
1	35.1	41.0	59.0	64.9	61.54
2	39.8	51.7	48.3	60.2	53.38
3	37.6	71.0	29.0	62.4	43.30
4	27.6	88.8	11.2	72.4	37.41

Tabla 5.6: Niveles porcentuales de efectividad para un promedio de 20 RSSI. **Tabla elaborada por los autores de este trabajo con los resultados obtenidos**.

Factor Desviación Estándar	FNR[%]	FPR[%]	TNR[%]	TPR [%]	Exactitud[%]
0.25	53.8	23.4	76.6	46.2	63.55
0.5	48.3	29.0	71.0	51.7	62.70
1	40.6	37.3	62.7	59.4	61.30
2	37.8	51.2	48.8	62.2	54.55
3	34.8	67.0	33.0	65.2	46.80
4	28.7	79.5	20.5	71.3	42.23

Tabla 5.7: Niveles porcentuales de efectividad para un promedio de 5 RSSI. **Tabla elaborada por los autores de este trabajo con los resultados obtenidos**.

Para obtener una mejor visualización de datos, los resultados obtenidos de las tablas anteriores fueron graficados en las figuras (5.8), (5.9) y (5.10).

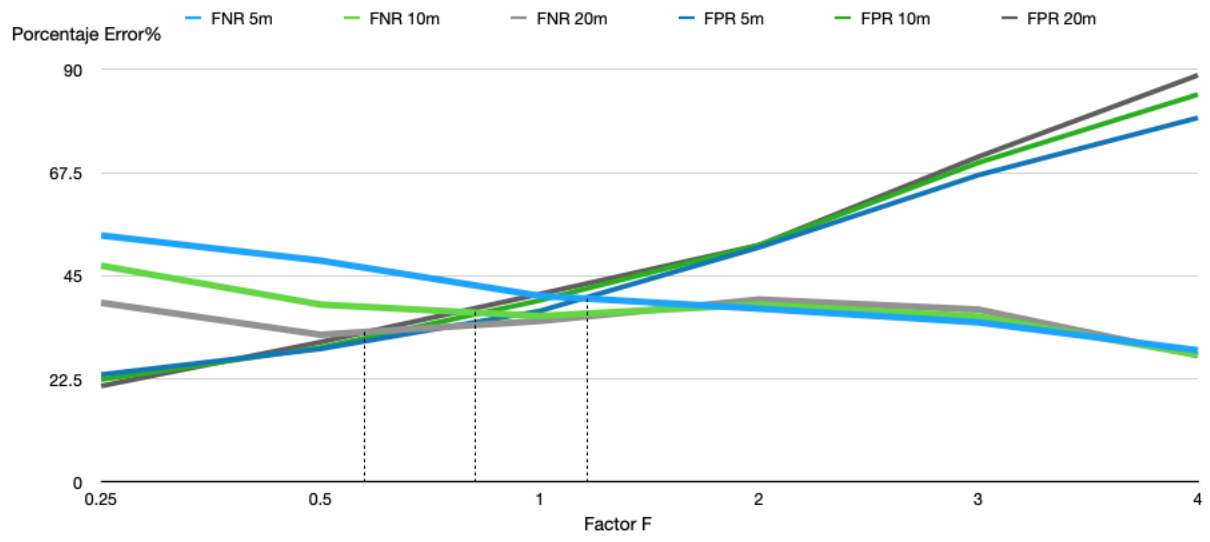

Figura 5.8: Comparativa de falsos negativos vs falsos positivos de la primera fase del algoritmo para promedios de RSSI con 5, 10 y 20 números de muestras.**Figura elaborada por los autores de este trabajo**.

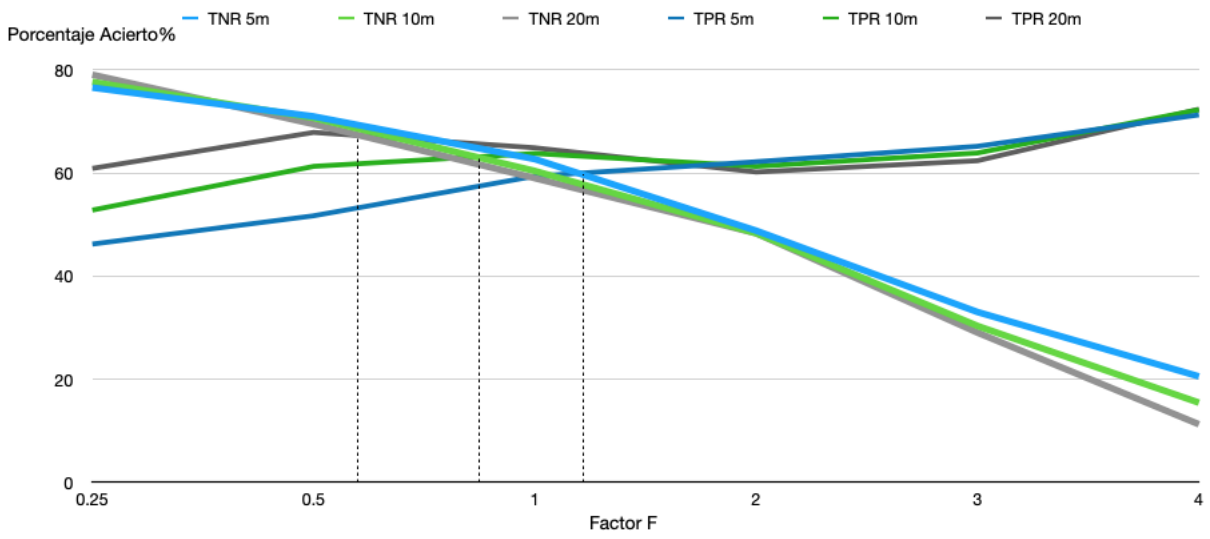

Figura 5.9: Comparativa de falsos negativos vs falsos positivos de la primera fase del algoritmo para promedios de RSSI con 5, 10 y 20 números de muestras. **Figura elaborada por los autores de este trabajo**.

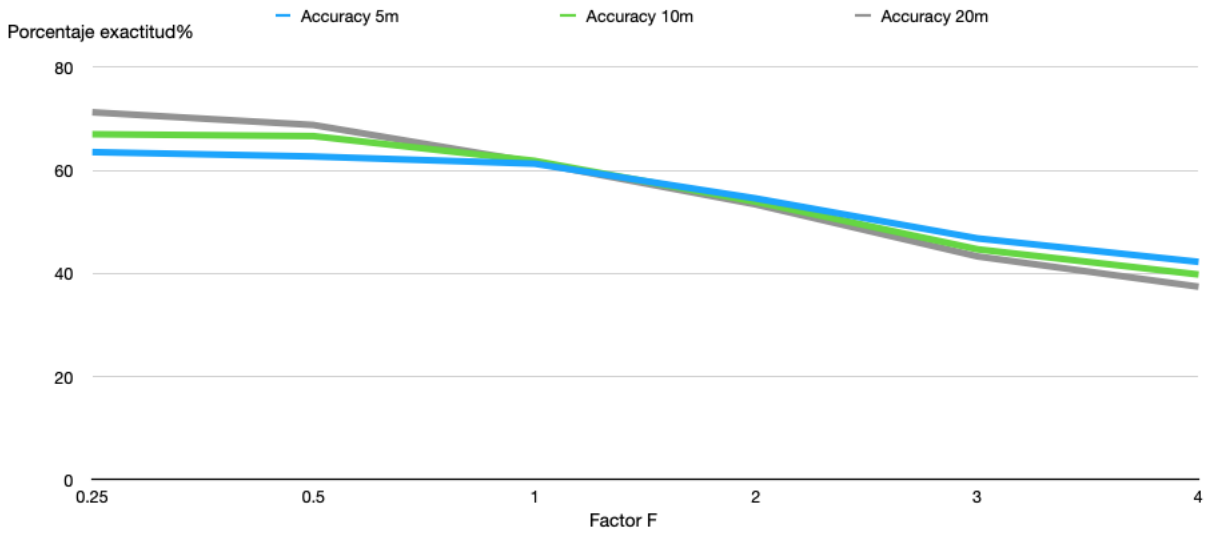

Figura 5.10: Comparativa de falsos negativos vs falsos positivos de la primera fase del algoritmo para promedios de RSSI con 5, 10 y 20 números de muestras.**Figura elaborada por los autores de este trabajo**.

Es posible observar con los resultados de las figuras (5.8), (5.9) que para promedios comparativos de 5, 10 y 20 muestras se encuentran diversos puntos de equilibrio entre los campos analizados para cada caso, es decir, un punto donde identidades genuinas son incluidas menos veces en la lista gris sin dejar de agregar identidades Sybil a ésta lista.

Los resultados graficados en la figura (5.8), (5.9) presentaron ciertas diferencias respecto a los demás experimentos similares. Aunque de igual forma, la intersección de los campos de falso negativo y falso positivo continuó presentándose aunque con un Factor F más grande.

Los resultados obtenidos a partir de las tablas (5.8), (5.9) y (5.10), muestra que a pesar de que un tamaño de muestras $Muestras < 10$ RSSI, la exactitud del algoritmo decrementa hasta un 42 %, el comportamiento de muestra igual para los tres casos considerados. Debido al decremento de la exactitud para el caso de $Muestras = 5$ RSSI, se determinó que el número de muestras utilizadas para el promedio de RSSI debe ser superior a 10 muestras para así reducir el porcentaje de los campos analizados durante éste experimento.

Por otra parte, el valor de desviación estándar ideal para adoptar en el esquema de detección debe estar entre 0.5 y 1, valor cercano al equilibrio de FNR y FPR en los experimentos para promedios de 10 y 20 muestras.

5.5. Algoritmo de descarte de nodos malignos mediante pruebas de trabajo.

Los resultados obtenidos con el experimento de pruebas de desafío permitieron establecer un nivel de desafío entre 2 y 3, mientras que los resultados del algoritmo de búsqueda de nodos malignos mediante RSSI sugieren usar un factor de desviación estándar entre 0.5 y 1. Con la combinación de los resultados anteriores fue posible implementar la segunda fase del esquema de detección en tarjetas Wireless Stick Lite para que ésta se ejecutara inmediatamente después de terminar la primera fase del esquema de detección, como lo describe el algoritmo (1).

Con el objetivo de replicar un escenario Sybil Attack, se implementó la red descrita en la figura (5.11), la cual se encuentra constituida por tres nodos genuinos (nodos azules), los cuales guardan las identidades 1,3 y 4 respectivamente, y un nodo maligno (nodo rojo), que se encarga de replicar las identidades 2,6,7 y 8.

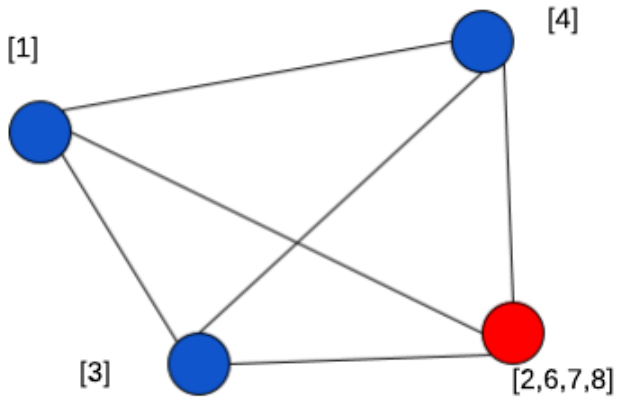

Figura 5.11: Red implementada para el experimento de la segunda fase del esquema de detección. **Figura elaborada por los autores de este trabajo**.

El nodo genuino con el identificador 1 fue utilizado para recopilar los resultados del experimento, por lo que en este nodo fue cargado un programa que contenía el procedimiento seguido en los algoritmos (3) para la primera fase y (4), (6) y (7) para la segunda fase. El resto de los nodos genuinos, es decir, el 3 y 4, fueron programados para enviar paquetes con sus respectivos

identificadores y responder sus pruebas de trabajo. Finalmente, el nodo maligno fue programado para enviar mensajes con sus múltiples identidades y responder las pruebas de trabajo que sean asignadas a a sus identidades sin tener prioridad sobre alguna de ellas.

Una vez planteado el escenario de la figura (5.11), se propuso la ecuación (5.3) para que el nodo con identificador 1 tuviera un tiempo de referencia adecuado para categorizar las respuestas de los nodos que realizan su prueba de trabajo.

$$T_{Total} = T_{prueba} + F(2T_{aire} + T_{margen}) \tag{5.3}$$

Donde:

T_{Total}: Tiempo que el nodo genuino dará como máximo para recibir la respuesta de un nodo.

T_{prueba}: Tiempo que le llevó al nodo genuino realizar la prueba de trabajo.

T_{aire}: Tiempo que dura la propagación del paquete en transmisión y recepción.

T_{margen}: Margen de tiempo adicional establecido por el tiempo de ejecución menor de prueba de trabajo entre dos.

F: Factor que lleva un valor variable en cada experimento.

El experimento que se llevó a cabo para determinar la eficiencia de la segunda prueba consistió en obtener el listado de sospechosos que proporciona el nodo genuino después de ejecutar la segunda fase del esquema de detección a partir de la lista gris recibida de la primera fase para diferentes dificultades.

Para ésta fase del esquema de detección se realizaron experimentos de 1000 ensayos donde el factor F tomó distintos valores entre 0,25 y 4. En cada uno de los 1000 ensayos se obtuvo una *lista gris*, es decir, la lista de sospechosos proporcionada por el primer esquema de detección. Además, se obtuvo la *lista negra*, es decir, las identidades que no lograron pasar la prueba de la segunda fase.

Las Tablas (5.8), (5.9) y (5.12) muestran los resultados obtenidos para esta segunda fase considerando una dificultad 2 para los parámetros FNR, FPR, TNR, TPR y Exactitud. Adicionalmente, se obtuvo la fase 1, teniendo una comparativa entre ambas fases.

Factor F	FNR fase 1	FNR fase 2	FPR fase 1	FPR fase 2
0.25	32.9	32.9	28.8	27.8
0.5	35.7	35.7	32.3	32.0
1	21.6	21.6	43.4	43.1
2	3.1	3.1	47.6	47.2
3	4.0	4.0	51.7	51.4

Tabla 5.8: Resultados obtenidos para falsos negativos y falsos positivos de la fase 2 comparado con la fase 1. **Tabla elaborada por los autores de este trabajo con los resultados obtenidos**.

La gráfica en la figura (5.12) muestra la comparación de los falsos negativos obtenidos en la fase 1 contra los obtenidos en la fase dos. Respectivamente esta misma gráfica muestra la comparación de falsos positivos de la fase 1 contra la fase 2.

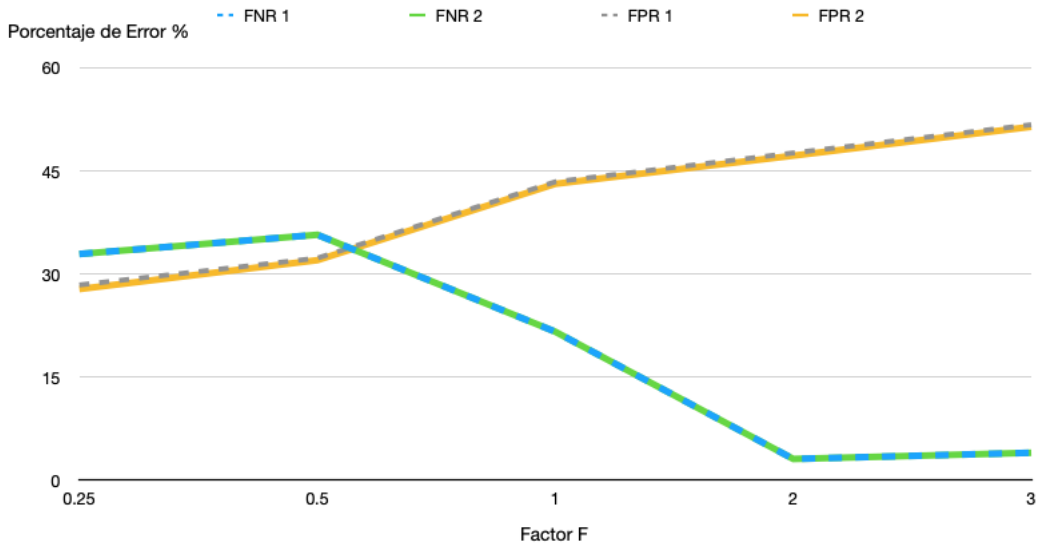

Figura 5.12: Comparativa de falsos negativos y falsos positivos para una dificultad de 2. **Figura elaborada por los autores de este trabajo**.

De la misma forma, la tabla (5.9) muestra los resultados para verdaderos negativos y verdaderos positivos para una dificultad 2 en comparativa con la fase 1.

Factor F	TNR fase 1	TNR fase 2	TPR fase 1	TPR fase 2
0.25	71.6	72.2	67.1	67.1
0.5	67.7	68.0	64.3	64.3
1	56.6	56.9	78.4	78.4
2	52.4	52.8	96.9	96.9
3	48.3	48.6	96	96

Tabla 5.9: Verdaderos negativos y verdaderos positivos de la fase 2 comparado con la fase 1. **Tabla elaborada por los autores de este trabajo con los resultados obtenidos**.

La gráfica de la figura (5.13) muestra una comparativa de los verdaderos negativos y verdaderos positivos obtenidos en la fase 1 contra la fase 2.

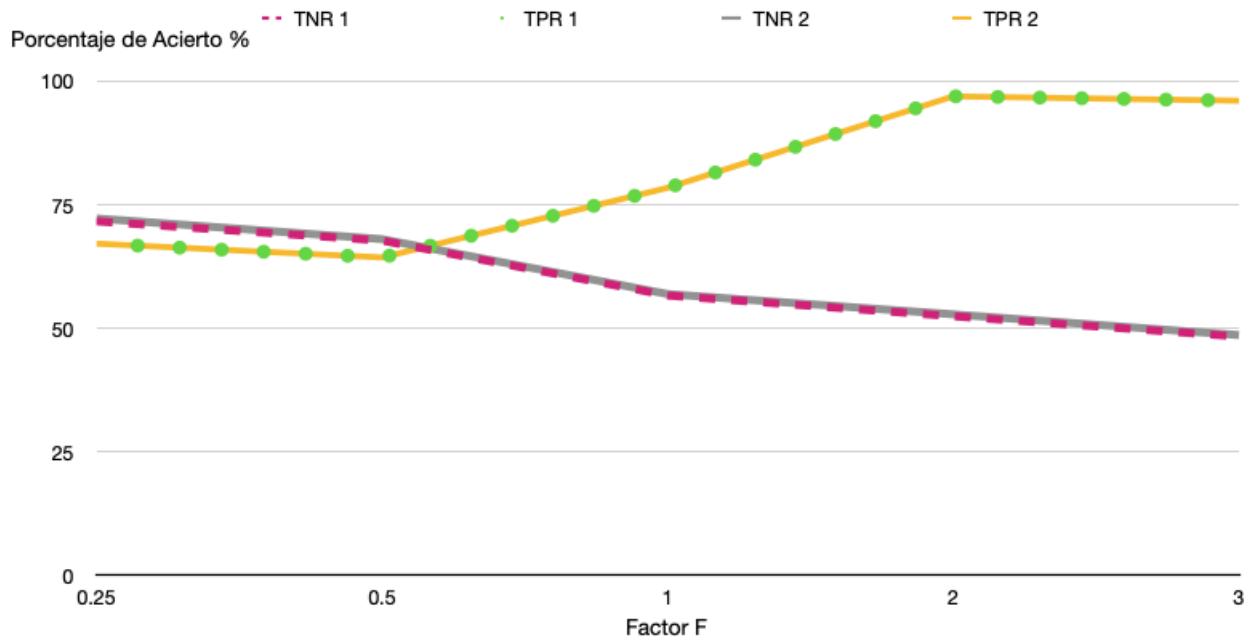

Figura 5.13: Comparativa de verdaderos negativos y verdaderos positivos para una dificultad de 2. **Figura elaborada por los autores de este trabajo**.

De la misma forma se obtienen los resultados para la fase dos, considerando una dificultad 3 en las pruebas de trabajo. Estos resultados se muestran en las Tablas (5.10),(5.11),(5.13).

Factor F	FNR fase 1	FNR fase 2	FPR fase 1	FPR fase 2
0.25	33.8	33.8	25.8	24.7
0.5	34.9	34.9	30.6	29.6
1	26.6	26.6	41	37.1
2	17.3	17.3	44.0	36.7

Tabla 5.10: Falsos negativos y falsos positivos de la fase 2 comparado con la fase 1 con una dificultad 3. **Tabla elaborada por los autores de este trabajo con los resultados obtenidos**.

La gráfica en la figura (5.14) muestra la comparación de los falsos negativos obtenidos en la fase 1 contra los obtenidos en la fase dos. Respectivamente esta misma gráfica muestra la comparación de falsos positivos de la fase 1 contra la fase 2. La Tabla (5.11) muestra los resultados para

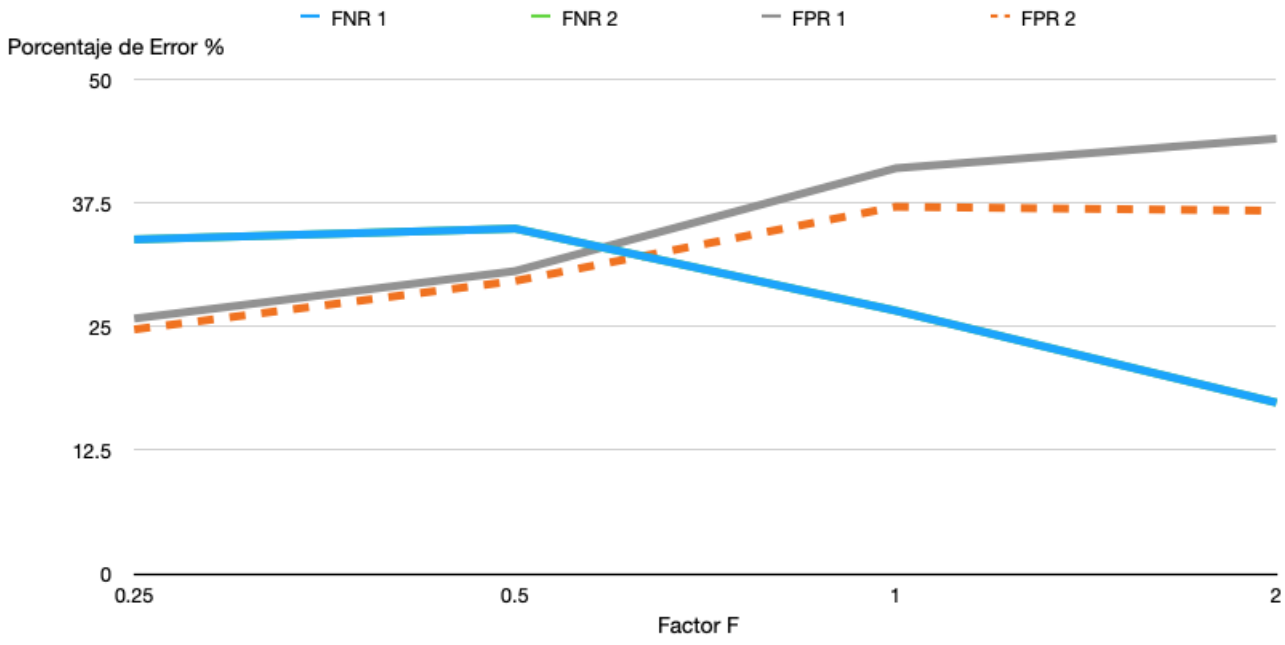

Figura 5.14: Gráfica comparativa de la exactitud de la fase 1 contra la obtenida en fase 2 con dificultad 3. **Figura elaborada por los autores de este trabajo**.

verdaderos negativos y verdaderos positivos para una dificultad 2 en comparativa con la fase 1. La gráfica de la figura (5.15) muestra una comparativa de los verdaderos negativos y verdaderos positivos obtenidos en la fase 1 contra la fase 2.

Factor F	TNR fase 1	TNR fase 2	TPR fase 1	TPR fase 2
0.25	74.2	75.3	66.2	66.2
0.5	69.4	70.4	65.1	65.1
1	59.0	62.9	73.4	73.4
2	56.0	63.3	82.7	82.7

Tabla 5.11: Verdaderos negativos y verdaderos positivos de la fase 2 comparado con la fase 1 considerando una dificultad 3. **Tabla elaborada por los autores de este trabajo con los resultados obtenidos**.

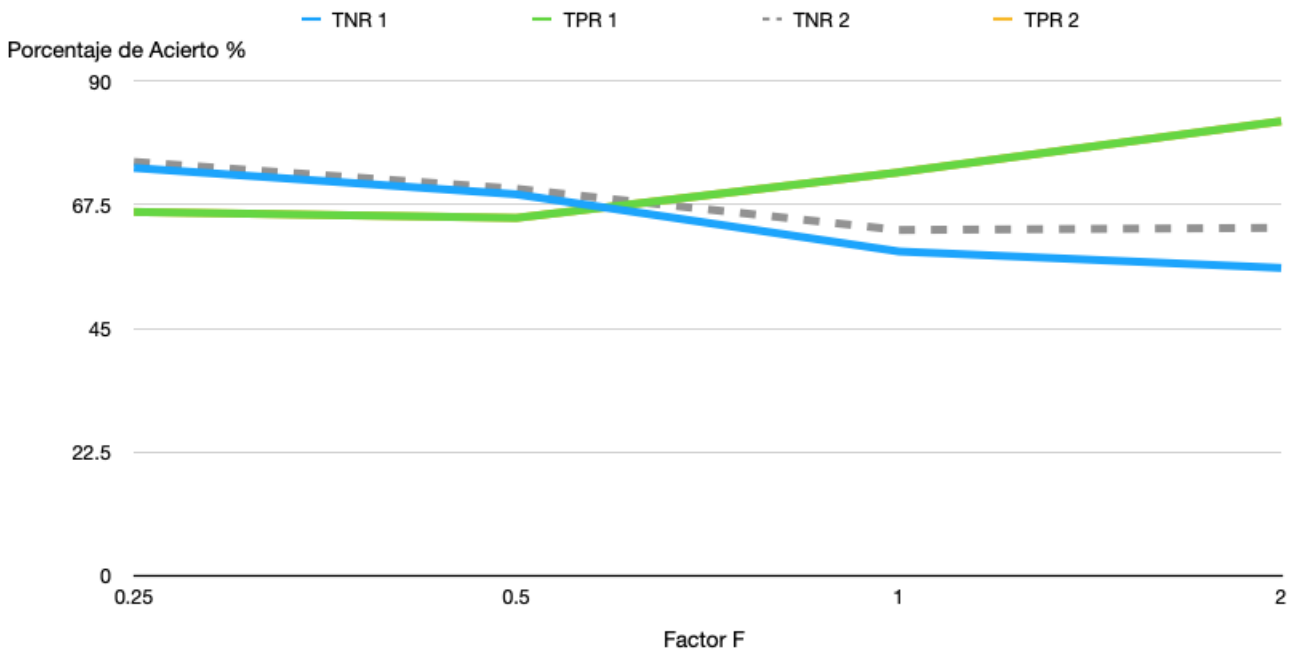

Figura 5.15: Comparativa de verdaderos negativos y verdaderos positivos para una dificultad de 3. **Figura elaborada por los autores de este trabajo**.

Las Tablas (5.12) y (5.13) muestran la comparación de la exactitud para una dificultad 2 y 3 en esta fase contra una dificultad 1.

En las gráficas de la figura (5.16) se muestra la comparación de la exactitud de la fase 1 contra la de la fase 2 para dificultades de 2 y 3.

Aproximadamente una tercera parte de las identidades Sybil no fueron detectadas durante los ensayaos de éstos experimentos, mientras que para un factor de 1 y 2 aproximadamente este porcentaje se ve mayormente reducido debido a la fase 1. Se puede observar en las gráficas de las

Factor F	Exactitud fase 1	Exactitud fase 2
0.25	69.37	69.65
0.5	66	66.15
1	67.48	67.75
2	74.6	74.8
3	72.14	72.31

Tabla 5.12: Exactitud de la fase 2 con dificultad 2 respecto a la fase 1. **Tabla elaborada por los autores de este trabajo con los resultados obtenidos**.

Factor F	Exactitud fase 1	Exactitud fase 2
0.25	70.19	70.74
0.5	67.21	67.74
1	66.20	68.18
2	69.37	72.99

Tabla 5.13: Exactitud de la fase 2 con dificultad 3 respecto a la fase 1. **Tabla elaborada por los autores de este trabajo con los resultados obtenidos**.

figuras (5.12) y (5.14) que no hay un incremento de falsos negativos respecto a la fase 1, es decir, ninguna identidad falsa deja de ser detectada en la lista negra. Esto se puede interpretar como que ninguna identidad Sybil logra responder el desafío impuesto en la fase dos.

En el campo de falsos positivos de la *lista negra*, se mantuvieron porcentajes relativamente bajos, probando la efectividad del esquema de detección para recuperar nodos genuinos que fueron señalados como sospechosos durante el primer esquema de detección. Las gráficas de las figuras (5.12) y (5.14) se observa que la cantidad de falsos positivos en la fase dos disminuye considerablemente respecto a la fase 1. Esta observación sustenta que identidades honestas logran resolver desafíos y superar la fase 2.

La gráfica de la figura (5.15) muestra que la cantidad de verdaderos negativos aumenta durante la fase dos, es decir, una mayor cantidad de identidades honestas son consideradas como honestas debido a que logran contestar la prueba de trabajo impuesta. Las gráficas de las figuras (5.13) y (5.15) muestran que los verdaderos positivos se mantienen constante en la fase dos respecto a la

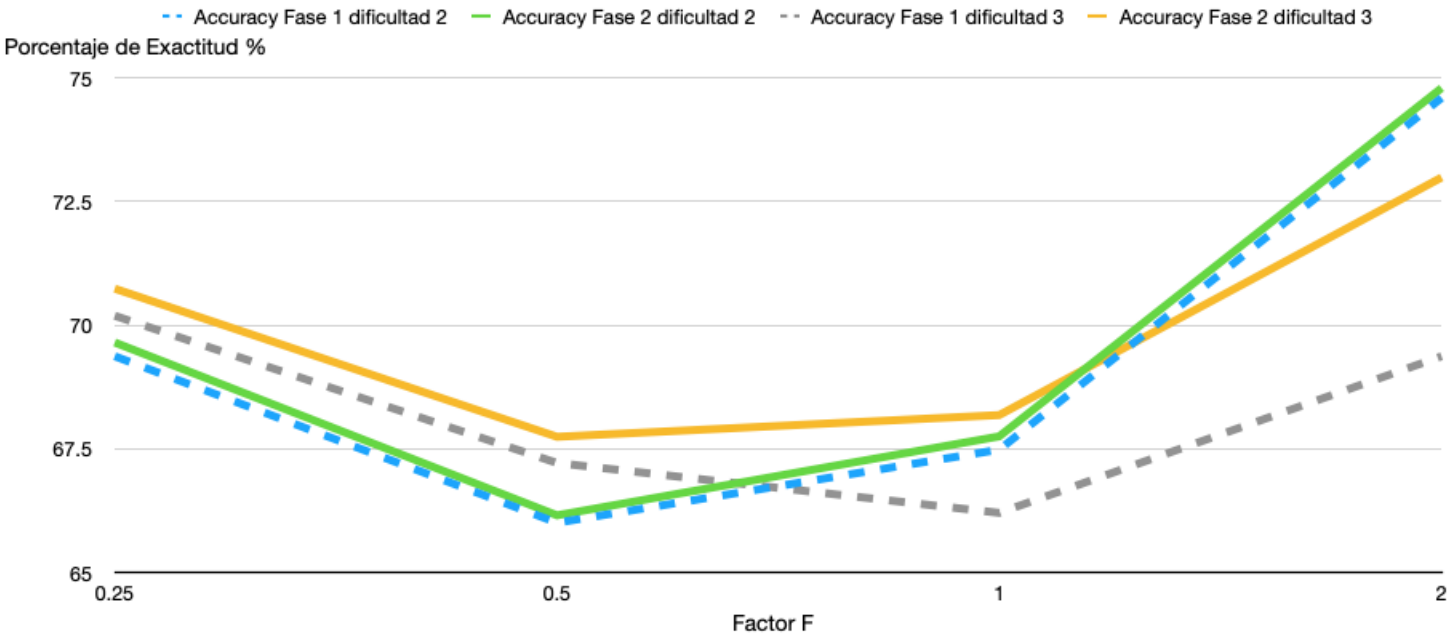

Figura 5.16: Comparativa de la exactitud de la fase 1 contra la obtenida en fase 2 con dificultades de 2 y 3. **Figura elaborada por los autores de este trabajo**.

fase 1.

En las gráficas de la figura (5.16) se puede observar que la exactitud se incrementa durante la fase 2. Esto debido a que durante esta fase las identidades Sybil son incapaces de responder desafíos(debido a que el nodo que genera estas identidades no es capaz de contestar más de un desafío a la vez), mientras que identidades honestas logran resolver estos desafíos. De acuerdo con la ecuación (5.2) se observa que si el número de identidades honestas aumenta, la exactitud aumenta.

Pese a que el factor F aumente más allá del rango $[0{,}5, 1]$ generé una lista gris con mayor FPR (como se observa en los resultados de la fase 1), los resultados en esta fase, sugieren que nodos genuinos logran pasar esta fase porque que no tienen que resolver n pruebas de trabajo, a diferencia de las identidades Sybil. Para sustentar esto, se generaron pruebas sin considerar el efecto de la fase 1 para las dificultades 2 y 3. Lo que implica que se genere una lista gris con todas las identidades, tanto falsas como honestas.

Los resultados de la Tabla (5.14) muestran los campos FNR, FPR, TNR, TPR y $Exactitud$ para estas pruebas con una dificultad 2 y 3. Se debe notar, que ya que se consideran todas las identidades, los falsos negativos de la fase 1 tienen que ser de $FNR = 0\,\%$ y los falsos positivos

$FPR = 100\,\%$, por lo que la exactitud $Exactitud = 50\,\%$.

Dificultad	FNR	FPR	TNR	TPR	Exactitud
2	0.0	98.6	1.4	100.0	50.7
3	0.0	80.1	19.9	100.0	59.975

Tabla 5.14: Resultados obtenidos cuando no se considera la fase 1. **Tabla elaborada por los autores de este trabajo con los resultados obtenidos**.

En éstas mismas pruebas se obtuvo el promedio de identidades encontradas por lista negras en ambas dificultades. Teniendo que cuando se tiene una dificultad 2, la lista negra en promedio tiene $ListaNegra$ =4.97 identidades por ejecución en promedio y para la dificultad 3 $ListaNegra$ =4.6010 identidades por ejecución en promedio.

A pesar de que la segunda fase logró demostrar que con su implementación es posible descartar las acusaciones de la primera fase, se tuvieron valores muy altos en los promedios de porcentajes de falsos negativos provenientes de la fase 1, que provocaron que el esquema perdiera efectividad. Sin embargo, durante el análisis de los resultados se llegó a una importante conclusión: Si dentro de la primera fase no se incluye una identidad Sybil dentro de la lista de sospechosos, éste no será detectado por las siguientes dos fases. Por esta razón se obtuvieron porcentajes tan altos del campo de Falso Negativo en la segunda fase ya que desde un principio hubo algunas identidades Sybil que no fueron incluidas durante las pruebas de trabajo.

Los resultados obtenidos demuestran que para que la segunda fase del esquema de detección logre cumplir con su objetivo, será necesario asegurar un porcentaje mínimo de falsos negativos durante la primera fase, aunque esto implique que el porcentaje de falsos positivos incremente, pues durante la segunda fase se lograrán omitir la mayor parte de los nodos genuinos que fueron acusados como maliciosos debido al resultado de la primera fase.

Los porcentajes de FNR, FPR, TNR, TPR y $Exactitud$ para la fase 2 aumentaron cuando el nodo que genera las pruebas de trabajo proporcionó una dificultad mayor y con ello más tiempo para que el resto de la red respondiera a sus desafíos, por lo que el beneficio con una dificultad 2 es menor en comparación con la dificultad 3.

5.6. Pruebas de consenso para detección final de Sybil Attack

Con los resultados obtenidos en el experimento del algoritmo de descarte de identidades Sybil a través de pruebas de trabajo se logró visualizar la importancia de establecer un valor de factor de desviación estándar que logre ya sea menos falsos positivos o menos falsos negativos.

A partir de esta observación se generó el ambiente de la figura (5.17) en donde los nodos señalados en color azul correspondieron a nodos genuinos y el rojo a un nodo maligno que replicó 3 identidades además de la propia, dentro de éste ambiente fue implementado todo el esquema de detección, es decir, fue añadida la tercera fase.

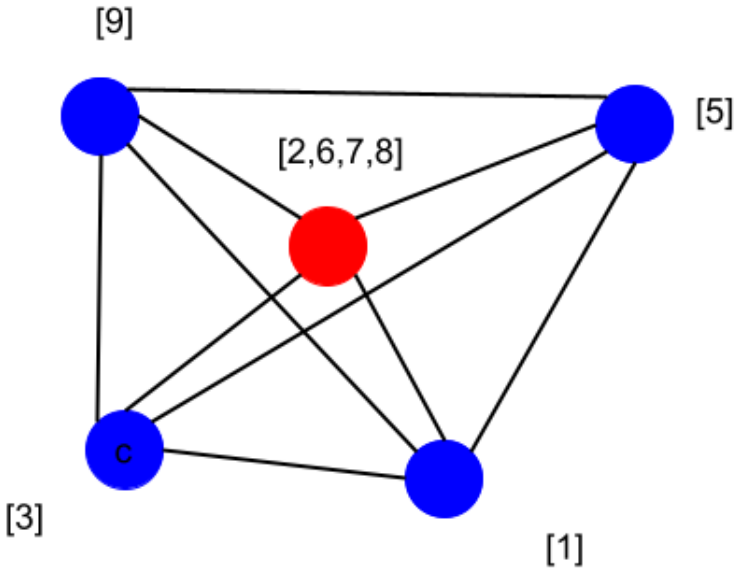

Figura 5.17: Red implementada para el experimento de la tercera fase del esquema de detección. **Figura elaborada por los autores de este trabajo**.

Con el fin de asegurar la combinación de las dos variables (desviación estándar y dificultad de prueba de trabajo) que proporcionará la mayor efectividad del esquema de detección, se realizaron 14 experimentos en donde la desviación estándar fue multiplicada por un factor que varió desde los 0.25 hasta 3 y la dificultad de la prueba de trabajo tomo como valores 1 y 2. En cada uno de estos experimentos se ejecutó el esquema de detección completo 1000 veces asegurando tener una muestra lo suficientemente grande para obtener resultados confiables.

Los resultados asociados a una dificultad de prueba de trabajo de 2 quedaron documentados

en las tablas (5.15) y (5.16).

En las tablas antes mencionadas se muestran los niveles porcentuales de falso negativo (FNR), falso positivo (FPR), verdadero negativo (TNR), verdadero positivo (TPR) y eficiencia. La tabla (5.15) muestra los resultados de los experimentos realizados con un dificultad de prueba de trabajo de 2. Por otra parte, la tabla (5.16) muestra los resultados para una dificultad de 3.

Factor F	FNR lista final	FPR lista final	TNR lista final	TPR lista final	Exactitud
0.25	55.88	15.36	84.63	44.11	67.27
0.5	46.42	17.9	82.09	53.57	69.87
1	39.92	27.32	72.67	60.07	67.27
1.25	45.8	31.08	68.91	54.2	62.6
1.5	46.02	30.74	69.25	53.97	62.7
2	46.35	37.93	62.06	53.64	58.45
3	38.16	49.55	50.44	61.83	55.33

Tabla 5.15: Tercera fase del esquema de detección para una dificultad 2. **Tabla elaborada por los autores de este trabajo con los resultados obtenidos**.

Factor F	FNR lista final	FPR lista final	TNR lista final	TPR lista final	Exactitud
0.25	47.76	11.29	88.79	52.23	73.12
0.5	41.38	10.6	89.39	58.61	76.2
1	32.41	19.63	80.36	67.58	74.88
1.25	31.8	20.89	79.1	68.19	74.42
1.5	31.54	23.92	76.07	68.45	72.81
2	35.19	20.58	79.41	64	73.15
3	34.61	27.4	72.6	65.38	69.49

Tabla 5.16: Tercera fase del esquema de detección para una dificultad 3. **Tabla elaborada por los autores de este trabajo con los resultados obtenidos**.

Con el objetivo de obtener una mejor visibilidad de los resultados obtenidos en este experimento, los valores de la tabla (5.15) fueron graficados en la figura (5.18) donde es posible observar que entre mayor sea el factor que multiplica a la desviación estándar, menor serán los verdaderos negativos,

es decir, el número de nodos genuinos descartados de la acusación del consenso será menor mientras que el número de identidades falsas detectadas como maliciosas aumenta (verdadero positivo). Por otra parte, las identidades falsas que fueron consideradas genuinas, es decir, los falsos negativos, tiende a un decremento con el aumento del factor que multiplica a la desviación estándar así como el número de nodos genuinos que fueron considerados como auténticos aumenta (falsos positivos).

Las conclusiones anteriormente obtenidas, fueron constantes desde la primera fase, pues esta misma tendencia se observó en los experimentos anteriores. Sin embargo, es importante destacar que la mayor eficiencia presente en todo el esquema de detección para una dificultad de prueba de trabajo de 2 se dio cuando el factor que multiplica a la desviación de estándar fue de 0.5 con un valor muy cercando al 70 %; este valor mostró un a tendencia a decrementar cuando el factor aumentó.

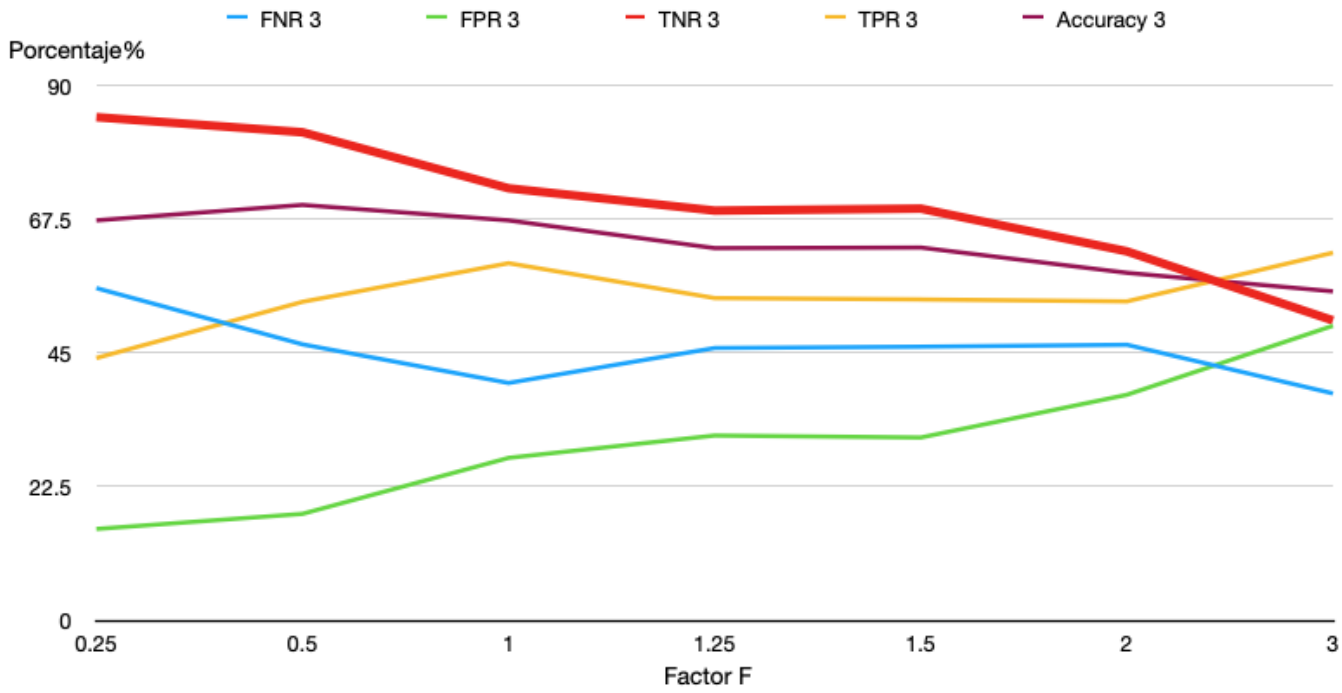

Figura 5.18: Exactitud esquema de detección completo con dificultad de prueba de trabajo igual a 2. **Figura elaborada por los autores de este trabajo**.

Los valores de la tabla (5.16) fueron graficados en la figura (5.19) donde es posible observar una tendencia bastante similar en los resultados para una dificultad de prueba de trabajo de 2, con una clara diferencia en los falsos negativos y falsos positivos, porcentajes que fueron considerablemente más bajos.

La eficiencia del esquema de detección se vio beneficiada con el aumento de la dificultad de la prueba de trabajo, el mayor valor de este parámetro se dio para un factor que multiplica a la desviación estándar de 0.5. Sin embargo, los valores 1 y 1.25 tuvieron un valor similar, por lo que esta dificultad proporciona un mayor rango de valores que pueden multiplicar a la desviación estándar, lo que permite una mayor flexibilidad en la implementación desde la primera fase.

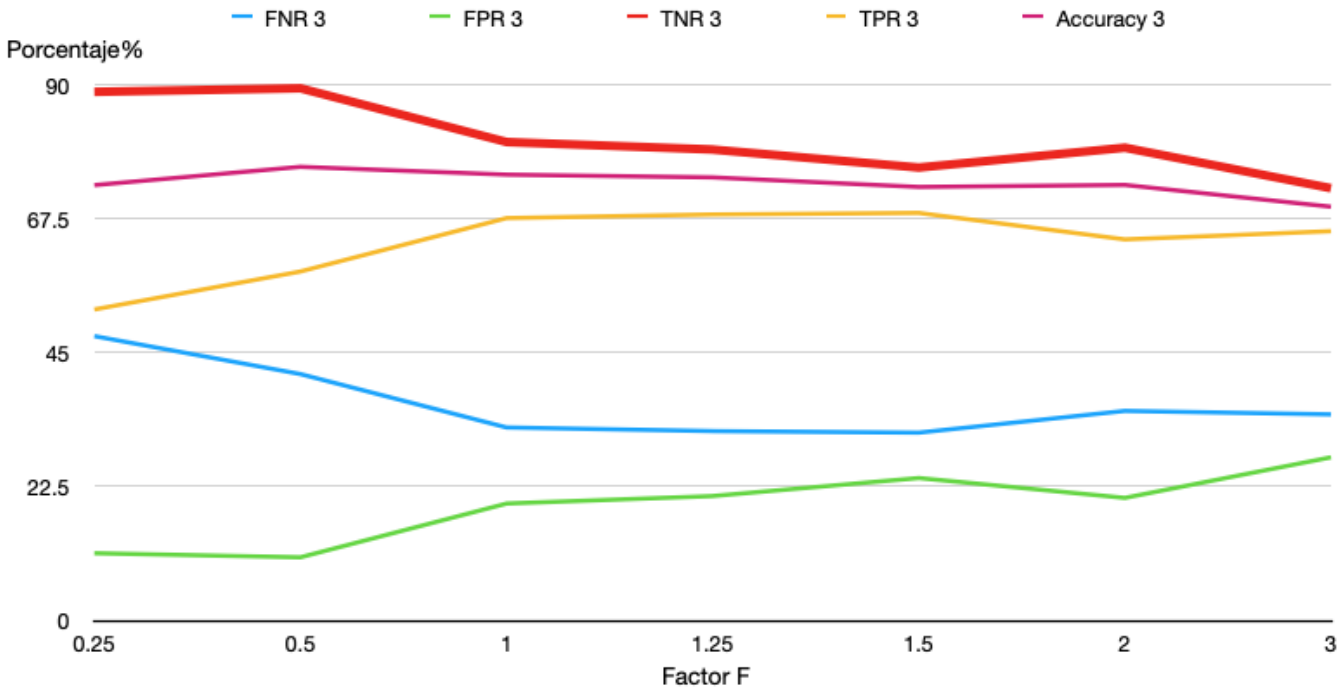

Figura 5.19: Exactitud del esquema de detección completo con dificultad de prueba de trabajo igual a 3. **Figura elaborada por los autores de este trabajo**.

Capítulo 6

Conclusiones

El esquema de detección de identidades falsas para un Sybil Attack implementado en este trabajo consistió en el desarrollo de 3 fases; la primera de ellas obtiene el RSSI de paquetes recibidos para poder detectar mensajes enviados con diferentes identidades que puedan provenir de una misma fuente y su objetivo principal es el reducir el procesamiento realizado en la segunda fase. La segunda etapa consiste en realizar pruebas de trabajo a cada una de las identidades catalogadas como sospechosas en la primera etapa. La última fase consiste en que un dispositivo con una identidad genuina que realiza una acusación con base en los resultados de las primeras dos fases pueda conocer la opinión del resto de los dispositivos con identidades que no entran como sospechosos.

Durante el desarrollo de este trabajo se realizaron una serie de experimentos que pueden englobarse en dos grandes bloques; durante el primer bloque se realizaron experimentos que tuvieron como objetivo hallar las principales características que se integrarían al esquema de detección. El segundo bloque de experimentos consistieron en probar la eficiencia del algoritmo implementado en cada una de sus fases, tomando como referencia 4 campos porcentuales: verdadero positivo, falso positivo, verdadero negativo y falso negativo.

Durante la primera etapa fue hallado el parámetro físico que se utilizaría en el esquema, pues se hizo una comparación entre RSSI y SNR. Durante este experimento obtuvimos resultados que señalaron una fuerte inconsistencia en ambos parámetros. Por ello, para su resolución adoptamos el uso de la diferencia en la lectura de paquetes, incorporando este principio en el algoritmo de detección, lo que permitió que la primera fase cumpliera con su objetivo de hallar las primeras identidades sospechosas en base a la diferencia entre dos promedios. De acuerdo con los resultados

obtenidos, para distancias de $d = 1[m]$, hay una desviación estándar en la diferencia de RSSI de $\sigma = 0{,}1593$, en comparación de la misma distancia con el caso de SNR con $\sigma = 0{,}7193$, esta comparación permitió concluir que RSSI era el mejor parámetro para el objetivo de este esquema. El segundo experimento de la primera etapa consistió en reconocer el tiempo que le toma a las tarjetas Wireless Stick Lite realizar pruebas de trabajo con diferentes dificultades, gracias a los resultados obtenidos en este experimento se concluyó que uno de los valores ajustables dentro del esquema de detección sería la dificultad de la prueba de trabajo.

El segundo bloque de experimentos consistió en probar la eficiencia del algoritmo implementado en cada una de sus fases, tomando como referencia cuatro parámetros: verdadero positivo, falso positivo, verdadero negativo y falso negativo.

En la implementación de la primera fase del esquema de detección se llegaron a resultados que influyeron en el resto de la ejecución, pues la efectividad de esta etapa influiría de forma directa a los resultados finales. Durante las pruebas, el algoritmo logró localizar en todos sus ensayos identidades falsas, sin embargo, también llegó a categorizar de esta forma a identidades genuinas. Este fenómeno fue representado por el valor porcentual de falso positivo, el cual incrementó de forma proporcional al aumento del factor de desviación estándar. Tener un factor de desviación estándar bajo, como el del valor de desviación estándar σ =0.25, permitiría que la menor cantidad posible de identidades genuinas sean catalogadas como sospechosas teniendo un resultado del 22 % en el parámetro de falsos positivos. Sin embargo, existe el riesgo de que pase lo mismo con identidades falsas, haciendo que el algoritmo pierda su efectividad desde la primera fase. En contraparte, aumentar el factor de desviación estándar, por ejemplo para un valor de $\sigma = 4$, provoca que una mayor cantidad de identidades genuinas sean catalogadas como sospechosas teniendo un resultado de el 84 % en el parámetro de falsos positivos. Sin embargo, la detección de identidades falsas durante esta primera fase se volvería mucho más efectiva pues el valor porcentual del campo de verdadero positivo se elevó del 52.8 % para un desviación estándar σ =0.25 al 72.3 % para una desviación $\sigma = 4$. Aunque la eficacia del algoritmo se elevaría, las identidades genuinas pagarían el costo de esta efectividad con procesamiento adicional en la segunda fase.

Con los resultados obtenidos en la implementación de la segunda fase se encontró que si la lista de identidades sospechosas que entrega la primera fase contiene a todas todas las identidades falsas, el porcentaje de falsos negativos se mantendría constante en la salida de la segunda fase. De igual forma, el segundo esquema de detección asegura disminuir los porcentajes de falsos positivos. Si se

incluyen identidades genuinas dentro de la lista de identidades sospechosas, estas serán omitidas de la lista en la salida de la segunda fase con una efectividad muy alta que se comprobó con los resultados documentados en la tabla (5.13), donde para un factor $F = 2$ se incrementa la eficiencia del 69,37 % para la primera fase al 72,99 % para la tercera fase.

Durante los experimentos realizados para la fase 2, se obtuvieron comportamientos muy similares en los resultados tanto para dificultad 2 y 3 considerando todos los factores de desviación estándar. Esto permite que en en una implementación sea posible seleccionar una amplia gama de valores ya sea el factor F o una dificultad que pueden ser determinados por las características de la aplicación, dándole al algoritmo una amplia flexibilidad en su implementación, pudiéndose considerar para redes con otras características.

Al poder implementar el esquema de detección con sus 3 fases, se obtuvieron resultados descriptivos sobre la eficiencia del algoritmo. Se realizaron experimentos variando la dificultad de la prueba de trabajo y el factor de desviación estándar. La implementación de la tercera fase tuvo un bajo impacto, pues la mayor eficiencia para un factor F =0.5 y una dificultad de 3 tuvo un valor 76.2 %. Este porcentaje fue muy cercano al obtenido en la eficiencia de las pruebas para la segunda fase del algoritmo. Sin embargo, la ejecución de la tercera fase requiere de poco procesamiento, lo que justifica su implementación en el esquema de detección diseñado en este trabajo.

Con los resultados mostrados en el capítulo anterior, se pueden determinar algunas áreas de mejora en cada una de las fases propuestas en este esquema. Por parte de la primera fase, se puede observar que conforme el número de muestras N aumenta para promediar el RSSI, el punto de cruce ente falsos positivos y negativos es en un factor F más bajo, por lo que podría optimizar el resultado de *Lista Gris* y por tanto mejorando la fase 2, sin embargo, esta mejora no es posible en este trabajo por la cantidad de memoria que tienen las tarjetas LoRa, limitando el procesamiento de datos que puede hacerse. Por otro lado, la mejora propuesta en la fase 2 pudiera ser referente a la dificultad empleada, pues con una dificultad 3 las tarjetas tardan demasiado tiempo en ejecutar varias pruebas encoladas, lo que provoca a su vez que nodos genuinos no contesten por completo las pruebas encoladas, la propuesta sería implementar un algoritmo que genere la fase dos con menos ocasiones para nodos genuinos que pasaron pruebas anteriores. Por parte de la fase 3, si bien los resultados demostraron no ayudar en mucho a la detección de identidades Sybil, si es posible notar que el algoritmo que implementa el consenso es basado en una simple comparación, por lo que una propuesta de mejora puede ser un esquema similar al propuesto en [26] en donde a

base de reputación es que se obtiene el conjunto de identidades participantes en un consenso, dicha reputación pudiera ser obtenida en base al número de veces que se pasó con éxito la fase 2, sin embargo implementar este esquema puede llevar a un procesamiento más pesado para la tarjeta y a consumir el ancho de banda existente en la red, pues paquetes de la fase 1 y 2 se transmiten aleatoriamente.

Apéndice A

Código principal del nodo honesto

```
/*
-----------------------------------------------------------------------------
                        Dispositivo honesto
-----------------------------------------------------------------------------
*/
#include "heltec.h"
#include "Node.hpp"

#define BAND    433E6 // Usa la banda 433 MHz
long lastSendTime = 0;        // ltimo paquete enviado
int interval = 1000; //Intervalo de tiempo
Node n; // Objeto que representa al nodo actual
vector<char> payload{'1','2'}; // Inicializa el payload
unsigned char id = '5'; //cambiar por cualquier ID
unsigned char dst='d';// Al inicio los paquetes son default y sin dst
unsigned char type = 0x00;//Default message
int isgl=0; //Describe si hay lista gris
int isPoW=0; // Describe si recibe un PoW
int lastpow=0;
//int counter;
vector<char> rnum;
vector<vector<char>> proofs;
```

```
vector<int> thresholds;
int T=0;
int lastbl;
int isbl=0;
int nmsj=0;

void setup() {
  // Inicializamos LoRa
    Heltec.begin(true, true, true, true , BAND);
    LoRa.setSpreadingFactor(7);
    LoRa.setCodingRate4(6);
    LoRa.onReceive(onReceive);//Interrupcion para recepcion
    LoRa.receive();
    Serial.println("Heltec.LoRa init succeeded.");
    //inicializa el nodo
    n.setID(id);//Configuramos la clase nodo
    n.setFactor(3); // Configura el Factor
    n.setDifficulty(2); // Configura dificultad
    n.setTime_interval(1000); //Intervarlo de tiempo
    n.setPoW_t();
}

void loop() {
  //n.setTm(type);
  /*
  Solo ejecutamos el main cada intervalo de tiempo aleatorio para evitar
  colisiones en la transmision de paquetes
  */
  unsigned char dst;
  vector<char> solution;
  vector<char> bl;
  vector<char> master;
  int j,i,k;
```

```
int ti,tf,tt,f;
int lastgl=0;
vector<char> honest;
if (isPoW==1)
{
  solution = n.solvePoW(rnum);
  type=0x02;
  Pack(type,dst,solution);
  sendMessage(n);
  //delay(1000);
  rnum.clear();
  solution.clear();
  isPoW=0;
  isgl=0;
}
if (millis() - lastSendTime > interval)
{
  if(isPoW==0){
    if((millis()-lastbl)>T && isbl ==1)
    {
      int isblist = n.SybilDetection();
      //Serial.println("Generated!!");
      bl = n.getBlackList();
      type = 0x03;
      n.HonestList();// get honest
      //Serial.println("get honest");
      Pack(type,0x00,bl); // Broadcast
      //Serial.println("packed");
      sendMessage(n);// send*/
      if (isblist==1)
      {
        //Serial.println("Generated");
      PrintBlackList(bl);
```

```
    n.clearBlackList();
    }
    else{
      Serial.println("**--BlackList--**");
      Serial.println("No blacklist");
    }
    lastbl=millis();
    thresholds.pop_back();
    isbl=0;
    n.Consensus(bl);
    master=n.getMasterBlackList();
    PrintMaster(master);
    n.ClearMaster();
  }
  if(proofs.size()>0)
  {
    dst = 'd';
    type =0x01;
    int proof_tam = proofs.size();
    if(millis()-lastpow>thresholds.back())
    {
      payload = proofs.back();
      Pack(type,dst,payload);
      sendMessage(n);
      lastpow=millis();
      //Serial.println(String(lastpow-lastbl));
      proofs.pop_back();
      lastbl=millis();
      T = thresholds.back();
      isbl=1;
    }
  }
  if(isgl==1 && thresholds.size()==0 ) // Si se genero la lista gris
```

```
      //entonces se genera PoW
      {
        proofs.clear();
        GL_pow(); // Genera PoW
        thresholds = n.calcThreshold();
        n.calcTmin();
        lastpow=0;
        lastbl=millis();
        isgl=0;
      }
      if( isgl==0 && isPoW==0)
      {
        type=0x00;
        dst = 'd';
        Pack(type,dst,payload);
        sendMessage(n);
      }
      lastSendTime = millis();
      interval = 1000;
      LoRa.receive(); //Activa una interrupci n si recibe un mensaje
    }
  }
}

void sendMessage(Node n)
{
  /*
  Este codigo envia mensajes de diferentes tipos
  Tipo 0 : mensaje generico o de cualquier clase
  Tipo 1 : mensaje de solicitud de PoW
  Tipo 2 : mensaje de respuesta de PoW
  Tipo 3 : mensaje de consenso
  */
```

```
  int j=0;
  int i=0;
  LoRa.beginPacket();
  /*ID src*/
  LoRa.write(n.getID());
  /*Type message*/
  LoRa.write(n.getTm());
  /*ID dst*/
  LoRa.write(dst);
  /*Payload*/
  for (i=0;i<payload.size();i++)
  {
    //Serial.print(payload.at(i));
    LoRa.print(payload.at(i));
  }
  LoRa.endPacket();
  payload.clear();
}


void onReceive(int packetSize)
{
  /*Interrupcion*/
  if (packetSize == 0) return;
  char IDE = (char)LoRa.read(); // Recibe ID
  unsigned char type_me = LoRa.read(); // Recibe tipo de mensaje
  /*ID dst*/
  char dst = (char)LoRa.read();
  //Serial.println("Destination ID : "+String(dst));
  /*Payload*/
  String incoming="";
  while(LoRa.available())
  {
```

```
    /*
    Recibe todo el payload del mensaje
    */
    incoming += (char) LoRa.read();
  }
  /*Reading RSSI*/
  int rssi = (int )LoRa.packetRssi(); // Obtiene rssi del mensaje
  //Serial.println("RSSI: "+String(rssi));
  /*Storage RSSI*/
  storageRSSI(IDE,type_me,rssi); // Almacenamos el ID y rssi recibido
  /*Phase 1*/
  if(nmsj>=100){
    isgl= n.Discard();// Algoritmo de descarte de nodos maliciosos
  }
  /*Unpack content*/
  Unpack(type_me,dst,incoming,IDE);
}
void storageRSSI(char IDE, char type, int rssi)
{
  /*
  Almacena un ID en una estructura de datos y su correspondiente rssi
  en una cola de rssi
  */
  int a =n.IsinHist(IDE);
  if (a==1)
  {
    /*Si el ID existe en el historial de IDs revisa que la cola de rssi no este
    llena*/
    if(n.isQueueFull(IDE)==true)
    {
      /*
      Si esta llena la cola de rssi, elimina el rssi mas viejo.
      */
```

```
      int r = n.RemoveRSSI(IDE);
    }
  }
  else
  {
    /*
    Si no existe el ID en el historial de IDs, se agrega el nuevo ID en el
    historial y se crea su cola de rssi.
    */
    n.AddIDtoHist(IDE);
  }
  n.AddRSSI(IDE, rssi); //Agregamos el rssi del mensaje a sus correspondiente ID
  nmsj ++;
}
```

Apéndice B

Implementación de la clase nodo

```
//
//  Node.cpp
//  PI_SybilAttack_defense
//
//  Created by <Jorge O. Gonzalez> on 05/09/2020.
//
//
/*
--------------------------------------------------------------------------------

                        Class Node implementation

--------------------------------------------------------------------------------

*/
/*-------------------------------Phase 1 : RSSI-------------------------------*/
int Node::Discard()
{
    int i,j;
    vector<queue> id_test;
    vector<char> suspected;
    int sup,inf;
    int ans;
    int resp;
    int sump,sumdesv,var;
```

```
for (i=0;i<this->Hist.size();i++)
{
  if (this->Hist.at(i).end>9)
  {
    sump =0;
    sumdesv=0;
    var=0;
    for (j=0;j<Hist.at(i).end;j++)
    {
      sump = sump + this->Hist.at(i).RSSI[j];
    }
    this->Hist.at(i).prom = sump/10;
    for (j=0;j<Hist.at(i).end;j++)
    {
      sumdesv= sumdesv +
      ((this->Hist.at(i).RSSI[j]-this->Hist.at(i).prom)*
      (this->Hist.at(i).RSSI[j]-this->Hist.at(i).prom));
    }
    var = sumdesv/10;
    this->Hist.at(i).desv = sqrt(var);
    id_test.push_back(this->Hist.at(i));
  }
}
if(id_test.size()>1)
{
  for(i=0;i<id_test.size();i++)
  {
    suspected.clear();
    for(j=0; j<id_test.size();j++)
    {
      if(id_test.at(i).ID!=id_test.at(j).ID)
      {
        inf = id_test.at(i).prom-(this->fact*(id_test.at(i).desv));
```

```
            sup =  id_test.at(i).prom+(this->fact*(id_test.at(i).desv));
            if(id_test.at(j).prom>inf && id_test.at(j).prom<sup)
            {
              suspected.push_back(id_test.at(j).ID);
            }
          }
        }
        if(suspected.size()>0){
          suspected.push_back(id_test.at(i).ID);
          if(inGraylist(suspected)!=1)
          {
            this->graylist.push_back(suspected);
          }

        }

      }
    }
    if(this->graylist.size()>0)
    {
      ans=1;
    }
    else{
      ans=0;
    }
    return ans;
}
void Node::removesubset()
{
  this->graylist.clear();
}
/*-----------------------------Phase 2 : PoW-----------------------------*/
void Node::genPoW(vector<char> subset,vector<char> rand_n)
```

```
{
  int i,j;
  string number ="";
  string input="";
  string solution;
  int i_t,f_t,t_pow;
  vector<vector <char>> solutions;
  vector<string> pow_solutions;
  vector<int> pow_time;
  number = randNumAdapter(rand_n);
  for (i=0;i<subset.size();i++)
  {
    input = number + subset.at(i);
    i_t = clock();
    solution=ProofOfWork(input,this->difficulty);
    f_t = clock();
    t_pow = f_t-i_t;
    solution=solution.substr(0,32);
    //cout<<subset.at(i)<<endl;
    //cout<<solution<<endl;
    pow_time.push_back(t_pow);
    pow_solutions.push_back(solution);
  }
  this->pow.push_back(pow_solutions);
  this->pow_ti.push_back(pow_time);
  this->id_tested.push_back(subset);
}
string Node::ProofOfWork(string input,int dif)
{
  string to_hash;
  string target;
  string hash="";
  //string solution;
```

```
  to_hash = toHash(input,hash);
  target = GenerateTarget(dif);
  do
  {
    hash = sha256(to_hash);
    to_hash = toHash(to_hash,hash);
  }while(hash.substr(0,dif)!=target);
  //cout<<"Mined"<<endl;
  return hash;
}

void Node::calcTmin()
{
  int i,j;
  int tmin,act,tole;
  for(i=0;i<this->pow_ti.size();i++)
  {
    tmin =0;
    for(j=0;j<this->pow_ti.at(i).size();j++)
    {
      if(j==0)
      {
        tmin = this->pow_ti.at(i).at(j);
      }
      else{
        act = this->pow_ti.at(i).at(j);
        if(tmin>act)
        {
          tmin = act;
        }
      }
    }
```

```
    tole =(this->pow_t+40+tmin)*1;//300 -> dif 2, 20000 -> dif 3
    this->tol.push_back(tole);
  }
}


/*-----------------------------------Phase 3 : PoW-----------------------------------------*/

void Node::Consensus(vector<char> bl)
{
  int i,j,k;
  vector<vector<char>> consensus;
  for(i=0;i<this->honest.size();i++)
  {
    for(j=0;j<this->Hist.size();j++)
    {
      if(this->honest.at(i)==this->Hist.at(j).ID)
      {
        //cout<<"ID cons:"<<this->Hist.at(j).ID<<endl;
        if(this->Hist.at(j).bl.size()>=0)
        {
          consensus.push_back(this->Hist.at(j).bl);
          this->Hist.at(j).bl.clear();
        }
      }
    }
  }
  //cout<<"Compare"<<endl;
  for(i=0;i<bl.size();i++)
  {
    //cout<<"ID : "<<bl.at(i)<<endl;
    for(j=0;j<consensus.size();j++)
    {
```

```
        //cout<<"Blacklist"<<j<<endl;
        for(k=0;k<consensus.at(j).size();k++)
        {
          //cout<<"with ID :"<<consensus.at(j).at(k)<<endl;
          if(bl.at(i)==consensus.at(j).at(k))
          {
            //cout<<"ID:"<<bl.at(i)<<endl;
            //cout<<"Is sybil!!!!!"<<endl;
            this->MasterBlackList.push_back(bl.at(i));
          }
        }
      }
    }
    if(this->MasterBlackList.size()==0)
    {
      //cout<<"Consensus was not reached!!"<<endl;
      this->MasterBlackList=bl;
    }
    this->honest.clear();
}
```

Referencias

[1] Amol Vasudeva y Manu Sood: *Survey on sybil attack defense mechanisms in wireless ad hoc networks.* Elsevier, 2018.

[2] *¿Qué es IoT?* Oracle México. https://www.oracle.com/mx/internet-of-things/what-is-iot/.

[3] Pastorino, Cecilia: *Blockchain: qué es, cómo funciona y cómo se está usando en el mercado.* We live security, 2018.

[4] Bauwens, Michel, Vasilis Kostakis y Alex Pazaitis: *Peer to peer.* University of Westminster Press, 2019.

[5] Molano, Natalia Andre: *Claves para entender la tecnología Blockchain.* BBVA, 2019.

[6] Noah Weston, Jared Willard y Peng Wang: *Performance of blockchain technology on DoD tactical networks.* En Blowers, Misty, Russell D. Hall y Venkateswara R. Dasari (editores): *Disruptive Technologies in Information Sciences II*, volumen 11013, páginas 109 – 119. International Society for Optics and Photonics, SPIE, 2019. https://doi.org/10.1117/12.2520541.

[7] *¿Qué es un bloque en Blockchain?* https://academy.bit2me.com/que-es-un-bloque-dentro-de-la-bl Consultado 01-02-2020.

[8] *¿Qué es un hash?* bit2me Academy. https://https://academy.bit2me.com/que-es-hash/, Consultado 01-02-2020.

[9] *¿Qué es el consenso en criptomonedas?* https://academy.bit2me.com/consenso-criptomonedas/, Consultado 01-02-2020.

[10] Sachin Shetty, Val Red, Charles Kamhoua, Kevin Kwiat y Laurent Njilla: *Data provenance assurance in the cloud using blockchain.* En Hall, Russell D., Misty Blowers y Jo-

nathan Williams (editores): *Disruptive Technologies in Sensors and Sensor Systems*, volumen 10206, páginas 125 – 135. International Society for Optics and Photonics, SPIE, 2017. https://doi.org/10.1117/12.2266994.

[11] Nakamoto, Satoshi: *Bitcoin: A Peer-to-Peer Electronic Cash System*. Cryptography Mailing list at https://metzdowd.com, Marzo 2009.

[12] Pino, Jorge Daniel: *SIMD sobre algoritmosde hash SHA-1 y SHA-256*. Universidad de Buenos Aires, 24 de Febrero del 2016.

[13] Mishra, Amitabh: *Threats and attacks*, página 43–60. Cambridge University Press, 2008.

[14] Newsome, J., E. Shi, D. Song y A. Perrig: *The Sybil attack in sensor networks: analysis defenses*. En *Third International Symposium on Information Processing in Sensor Networks, 2004. IPSN 2004*, páginas 259–268, April 2004.

[15] Elder, Jeff: *Cómo Kevin Ashton nombró El Internet de las Cosas*. avast blog, 2 de Septiembre del 2019. https://www.avast.com/es-es/c-what-is-the-internet-of-things.

[16] Jordi Salazar y Santiago Silvestre: *Internet de las Cosas*. Universidad Técnica Checa, Technická 2, Praga, República Checa, versión de prueba edición.

[17] Karen Rose, Scott Eldridge y Lyman Chapin: *La Internet de las Cosas - Una breve reseña*, 2015.

[18] Tomasi, Wayne: *Sistemas de Comunicaciones Electrónicas*. Prentice Hall, México, 4a edición, 2003.

[19] Ertürk MA, Aydın MA, Büyükakkaşlar MT y Evirgen H: *A Survey on LoRaWAN Architecture, Protocol and Technologies*. future internet, 11:216, Octubre 2019.

[20] Kais Mekki, Eddy Bajic, Frederic Chaxel y Fernand Meyer: *A comparative study of LPWAN technologies for large-scale IoT deployment*. ICT Express, 5(1):1 – 7, 2019, ISSN 2405-9595. http://www.sciencedirect.com/science/article/pii/S2405959517302953.

[21] Reynders, B., W. Meert y S. Pollin: *Range and coexistence analysis of long range unlicensed communication*. En *2016 23rd International Conference on Telecommunications (ICT)*, páginas 1–6, 2016.

[22] Reynders, B. y S. Pollin: *Chirp spread spectrum as a modulation technique for long range communication*. En *2016 Symposium on Communications and Vehicular Technologies (SCVT)*, páginas 1–5, 2016.

[23] Demirbas, M. y Youngwhan Song: *An RSSI-based scheme for sybil attack detection in wireless sensor networks*. En *2006 International Symposium on a World of Wireless, Mobile and Multimedia Networks(WoWMoM'06)*, páginas 5 pp.–570, 2006.

[24] Mohamed Baza, Mahmoud Nabil, Mohamed Mahmoud, Niclas Bewermeier, Kemal Fidan, Waleed Alasmary y Mohamed Abdallah: *Detecting Sybil Attacks using Proofs of Work and Location in VANETs*. En *IEEE Transactions on Dependaple and Secure Computing*, página 16 pp, 2018.

[25] Klonowski, Marek y Koza, MichaB: *Countermeasures against Sybil Attacks in WSN Based on Proofs-of-Work*. En *Proceedings of the Sixth ACM Conference on Security and Privacy in Wireless and Mobile Networks*, WiSec '13, página 179–184, New York, NY, USA, 2013. Association for Computing Machinery, ISBN 9781450319980. https://doi.org/10.1145/2462096.2462125.

[26] Alex Biryukov y Daniel Feher: *ReCon: Sybil-resistant consensus from reputation*. Pervasive and Mobile Computing, 61:101109, 2020, ISSN 1574-1192. http://www.sciencedirect.com/science/article/pii/S1574119219304742.

[27] Swathi P, Chirag Modi y Dhiren Patel: *Preventing Sybil Attack in Blockchain using Distributed Behavior Monitoring of Miners*. IEEE, Julio 2019.

[28] JongBeom Lim, HeonChang Yu y Joon-Min Gil: *Detecting Sybil Attacks in Cloud Computing Envirotnments Based on Fail-Stop Signature*. página 12 pp, 2017.

[29] Asfia, U., V. Kamuni, S. Sutavani, A. Sheikh, S. Wagh y N. M. Singh: *A Blockchain Construct for Energy Trading against Sybil Attacks*. En *2019 27th Mediterranean Conference on Control and Automation (MED)*, páginas 422–427, 2019.