ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

EVALUACIÓN DE VULNERABILIDADES DEL PROTOCOLO DISPLAY AUTHENTICATED ASSOCIATION, DEFINIDO EN EL ESTÁNDAR IEEE 802.15.6, MEDIANTE LA IMPLEMENTACIÓN DE UN ESCENARIO DE PRUEBAS

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

GUSTAVO ANTONIO DÁVILA VINTIMILLA

OSCAR DAVID TORRES SÁNCHEZ

DIRECTOR: MSc. PABLO WILLIAM HIDALGO LASCANO

CODIRECTOR: MSc. JORGE EDUARDO RIVADENEIRA MUÑOZ

Quito, noviembre 2018

AVAL

Certificamos que el presente trabajo fue desarrollado por Gustavo Antonio Dávila Vintimilla y Oscar David Torres Sánchez, bajo nuestra supervisión.

_____

MSc. Pablo William Hidalgo Lascano
DIRECTOR DEL TRABAJO DE TITULACIÓN

_____

MSc. Jorge Eduardo Rivadeneira Muñoz
CODIRECTOR DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

Nosotros, Gustavo Antonio Dávila Vintimilla y Oscar David Torres Sánchez, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedemos nuestros derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Gustavo Antonio Dávila Vintimilla	Oscar David Torres Sánchez

DEDICATORIA

Dedico el presente trabajo a mis padres, ya que ellos han sido el pilar fundamental para culminar este proyecto, siempre me han apoyado y me han dado fuerzas para seguir adelante, los quiero mucho.

A mi hermana Daniela, ella siempre ha estado cuando la he necesitado, te quiero mucho ñaña.

A toda mi familia y en especial a mis abuelitos Hilda, Bolívar y Carlos, espero tenerlos mucho más tiempo conmigo, los quiero y admiro demasiado.

Oscar David Torres Sánchez.

Dedico este trabajo de titulación a mis padres, quienes son las personas más felices y orgullosas con la culminación de esta etapa.

En general, dedicado a todas las personas que comparten conmigo esta emoción.

Gustavo Antonio Dávila Vintimilla.

AGRADECIMIENTO

Quiero primero agradecer a Dios, él me dio la vida, la salud y me bendijo con una gran familia y unos amigos excelentes.

A mis padres Paquita y Marcelo, gracias por todo, no tengo con que agradecerles todo lo que han hecho por mí, gracias por cada una de las enseñanzas dadas las mismas que me han hecho el hombre que soy ahora, son mi más grande bendición y siempre estaré agradecido con la vida por darme padres tan maravillosos.

A mi ñaña Daniela gracias por tus consejos, por tu tiempo, por compartir conmigo todo este tiempo, por todas las aventuras juntos y las que vendrán.

A mis abuelitos, a mis primos, a mi tías y tíos por siempre darme palabras de aliento y hacerme saber que cuento con su apoyo.

A mi compañero Gustavo ya que juntos hemos logrado culminar este objetivo, gracias por tu tiempo a lo largo de toda la carrera y en especial para culminar este proyecto.

Al Ing. Pablo Hidalgo principalmente por todos los consejos, enseñanzas y guía que me ha dado a lo largo de este proyecto.

Al MSc. Jorge Rivadeneira ya que sin él este trabajo posiblemente no se hubiera realizado de la forma en la que se hizo, gracias por su tiempo y dedicación desinteresada.

A mis compañeros y amigos de la Universidad, muchas cosas hemos vivido y las volvería a vivir junto a ustedes, cuantas dificultades las pude sobrellevar gracias a ustedes los quiero mucho.

Oscar David Torres Sánchez.

AGRADECIMIENTO

En primer lugar, agradezco a Dios por darme la oportunidad de estar aquí y cumplir con esta meta.

Agradezco infinitamente a mi familia, quienes se han encargado de darme su apoyo constante a lo largo de toda mi vida, y han depositado toda su confianza en mí; gracias por todo el amor que recibo de ustedes. Agradezco a mis padres por inculcarme valores primordiales para poder ser una persona íntegra por encima de todas las cosas. Gracias a mis hermanos por ser un apoyo en las diferentes circunstancias vividas a lo largo de esta etapa.

Agradezco especialmente a mis amigos y compañeros David Torres y Juan José López, con quienes pude disfrutar toda esta época universitaria. Les agradezco por su amistad, solidaridad, confianza y apoyo en los momentos más importantes y cruciales; sin ustedes a lado, este camino no hubiera sido el mismo. Son excelentes seres humanos y espero que puedan seguir así y brillen a lo largo de su vida.

Un agradecimiento muy sentido para mis tutores de tesis, Pablo Hidalgo y Jorge Rivadeneira. Les agradezco por toda su colaboración desinteresada, su tiempo invertido y por apersonarse de este compromiso sobre otras cosas no menos importantes.

Gracias Ing. Pablo Hidalgo por haberme mentalizado primero ser una buena persona y después un buen profesional. Por recordarme que el camino más fácil no siempre es el mejor, que las cosas más difíciles y que más sacrificio cuestan son aquellas que más satisfacción nos brindan. Gracias por ser una excelente persona y gracias por su amistad.

Le agradezco mucho al Ing. Jorge Rivadeneira por todo su interés en nosotros y su constante ayuda para juntos poder alcanzar esta meta tan importante. Gracias por haber sido un excelente guía y consejero en todo sentido, no solo el académico. Gracias por su confianza y gran amistad.

Gracias a todos mis docentes, de quienes en su momento aprendí cosas para la vida y la carrera profesional. Toda mi admiración y respeto hacia ustedes, gracias por hacer de esta universidad lo que es y conservar su prestigio.

Amigos con los que compartí diferentes etapas y supimos disfrutar de esta hermosa época universitaria, para ustedes gracias totales.

Gustavo Antonio Dávila Vintimilla

ÍNDICE DE CONTENIDO

RESUMEN

Las Redes Inalámbricas de Área Corporal (WBAN) están revolucionando el mundo debido a la gran importancia de sus aplicaciones. En cuanto a cuidado de la salud, las WBAN pueden mejorar la calidad de vida de un paciente que se somete a procesos de monitoreo, diagnóstico y tratamiento. Más allá de que una WBAN sea capaz de comunicarse remotamente mediante el uso de otras tecnologías de comunicación, y proporcionar gran variedad de datos en tiempo real, los grandes beneficios entre otros, radican esencialmente en acelerar el tiempo de diagnóstico de los pacientes.

Es necesario que las WBAN puedan ser utilizadas fácilmente y que crezca el índice de penetración de esta tecnología. Este objetivo será muy difícil de lograr si no se garantiza que las WBAN puedan mantener privacidad de la información en todas las etapas de su proceso de comunicación. Por tal razón, en este trabajo de titulación se evalúan vulnerabilidades en el protocolo *Display Authenticated Association* (DAA) estandarizado en el IEEE 802.15.6 y se presentan soluciones que eviten posibles ataques de Hombre en el medio o Impersonalización de entidades.

En el primer capítulo se presenta el estado del arte de las WBAN, características y requerimientos del Estándar IEEE 802.15.6, servicios de seguridad y procesos de asociación del mencionado estándar. En el segundo capítulo, se explican las fases de diseño e implementación de cada uno de los procesos necesarios para: implementar el protocolo DAA, evaluar vulnerabilidades en dicho protocolo, y presentar soluciones que logren mitigar las vulnerabilidades encontradas. En el tercer capítulo se muestran los resultados obtenidos al implementar el protocolo DAA, evaluarlo con ejecución de ataques de Hombre en el medio e Impersonalización de entidades; también se expone el análisis de los resultados de los ataques al variar algunos de los parámetros de las soluciones propuestas. En el cuarto capítulo se presentan las conclusiones y recomendaciones a las que se llegó después de analizar los resultados obtenidos. Finalmente, en Anexos, se encuentran el código de programación desarrollado para alcanzar los objetivos de este trabajo, y el Estándar IEEE 802.15.6 en el cual se basa gran parte de este estudio.

PALABRAS CLAVE: CMAC-AES**,** Curva Elíptica, *Display Authenticated Association*, IEEE 802.15.6, Protocolos de Asociación, WBAN.

ABSTRACT

Wireless Body Area Networks (WBAN) are revolutionizing the world due to the great importance of their applications. Regarding the health care, WBANs can improve the quality of life of patients that undergo processes of monitoring, diagnosis and treatment. Beyond that, a WBAN is able to communicate remotely through the use of other communication technologies, and provide a variety of data in real time, great benefits above all, are focused in accelerating the time of diagnosis.

It is necessary that the WBANs can be easily used and the penetration rate of this technology grows significantly. This objective will be very difficult to achieve if it is not guaranteed that the WBANs can assure information privacy in all stages of its communication process.

For this reason, in this work, vulnerabilities are evaluated in the Display Authenticated Association protocol (DAA) standardized in IEEE 802.15.6 and solutions are presented to avoid possible Man in the middle Attack or Impersonation Attack.

In the first chapter is presented the state-of-the-art of the WBAN, characteristics and requirements of IEEE 802.15.6, security services and association processes of the standard. In the second chapter, we explain the design and implementation phases of each of the processes necessary to implement the DAA Protocol, evaluate vulnerabilities in that protocol, and present solutions that mitigate the vulnerabilities found. The third chapter shows the results obtained by implementing the DAA protocol, evaluating it with execution of Man in the middle Attack or Impersonation Attack and also present an analysis of attacks' results when are changed some parameters of the proposed solutions. The fourth chapter presents the conclusions and recommendations reached after analyzing the results obtained. Finally, in Annexes it is possible to find the script developed for reaching the objectives of this research and IEEE 802.15.16 Standard, which is the main source of information of this document.

KEYWORDS: CMAC-AES, Elliptic Curve Cryptography, Display Authenticated Association, IEEE 802.15.6, Association Protocols, WBAN.

1 INTRODUCCIÓN

El estándar IEEE 802.15.6 de *Wireless Body Area Networks* (WBAN)[1], define cuatro protocolos de Asociación entre un Nodo y un Hub. Es importante que los procesos de Asociación sean seguros, y exista una adecuada autenticación entre las entidades que van a comunicarse. Si el proceso de Asociación se ve comprometido por falta de seguridad, toda la comunicación será vulnerable a ataques realizados por terceros. Como se puede ver en [2] y [3], los protocolos de Asociación utilizados en WBANs son vulnerables a varios tipos de ataques, lo cual hace que la comunicación sea insegura.

Debido al tipo de aplicaciones que se utiliza en WBANs [4], la información está obligada a ser confidencial e íntegra. Si las vulnerabilidades que se detallan teóricamente en [2] y [3] no son corregidas, el uso de WBANs dejará de ser una opción para aplicaciones que contienen información confidencial como monitoreo ambulatorio, médico, etc. Además, el estándar IEEE 802.15.6 promete aplicaciones que pueden cambiar la forma de llevar a cabo actividades muy importantes en el diario vivir de los seres humanos [5]; pero si no se provee de comunicación segura a las WBANs, éstas dejarán de ser interesantes para el usuario y no entrarán en auge como se espera.

Uno de los protocolos de Asociación definidos en el estándar es el *Display Authenticated Association* (DAA), el cual se pretende desarrollar y evaluar en este Trabajo de Titulación. Una vez evaluado este proceso de Asociación, se comprobará la existencia de vulnerabilidades, y se presentará una solución que haga de este proceso un método seguro para la Asociación entre el Nodo y el Hub. De esta forma se logrará comunicación segura en WBANs IEEE 802.15.6, mediante el uso del protocolo de Asociación modificado que se propondrá.

1.1 Objetivos

El objetivo general de este Proyecto Técnico es:

- Evaluar las vulnerabilidades en el protocolo de Asociación *Display Authenticated Association* (DAA) del estándar IEEE 802.15.6, mediante la creación de una aplicación en un escenario de pruebas WBAN.

Los objetivos específicos de este Proyecto Técnico son:

- Describir el estado del arte del Proceso de Asociación DAA definido en el estándar IEEE 802.15.6, para desarrollar la aplicación en Java que permita la implementación de dicho protocolo.
- Implementar un escenario de comunicaciones para emular la WBAN.
- Desarrollar el protocolo DAA sobre el cual se realizarán las pruebas de vulnerabilidades.
- Evaluar vulnerabilidades en el proceso de Asociación, mediante la ejecución de ataques de Hombre en el medio sobre el escenario de pruebas.
- Analizar los resultados de las pruebas de vulnerabilidades del protocolo DAA y proponer una solución que mejore la integridad de la información en el proceso de Asociación.

1.2 Alcance

En este proyecto se investigará el protocolo DAA definido en [1], que se lleva a cabo entre los nodos y el Hub de una WBAN. Este proceso debe contener un enlazamiento de entidades mediante tramas *Beacon*; luego se procederá a la simulación del proceso de Asociación, el cual empleará el procedimiento de Diffie-Hellman para el intercambio de llaves públicas entre entidades, sobre un canal inseguro. Estas llaves públicas serán generadas a partir de llaves privadas propias de las entidades, utilizando Criptografía de Curva Elíptica [6]. Según el estándar IEEE 802.15.6 se utilizará la curva elíptica P-256, estandarizada y publicada en FIPS Pub 186-3 [7].

Además, es importante investigar sobre funciones criptográficas que se llevan a cabo en este proceso de Asociación, tales como Estándar de Encriptación Avanzada (AES) disponible en FIPS Pub 197 [8], y Código de Autenticación de Mensajes basado en Cifrado (CMAC) disponible en el RFC 4493 [9], con el fin de entender el funcionamiento de éstos y plasmarlos en código de programación que es parte del DAA. Con la codificación de todo el procedimiento llevado a cabo, y mediante el uso de lenguaje Java [10], se desarrollará la aplicación que implemente el protocolo DAA en el escenario de comunicaciones.

Se creará un ambiente de pruebas con el escenario de comunicaciones mostrado en la Figura 1.1, para evaluar en cada etapa las vulnerabilidades y el funcionamiento del ataque mencionado. Se tratará de determinar si un atacante es capaz de asociarse de manera efectiva con las entidades, lo cual afectaría la integridad de la información que se intercambia entre las entidades. Pero, no solo es necesario comprobar si el atacante puede asociarse, es necesario probar la solución a ese ataque con el objetivo de evitar que este proceso de Asociación se vea afectado por fallas de concepto dentro del estándar [1].

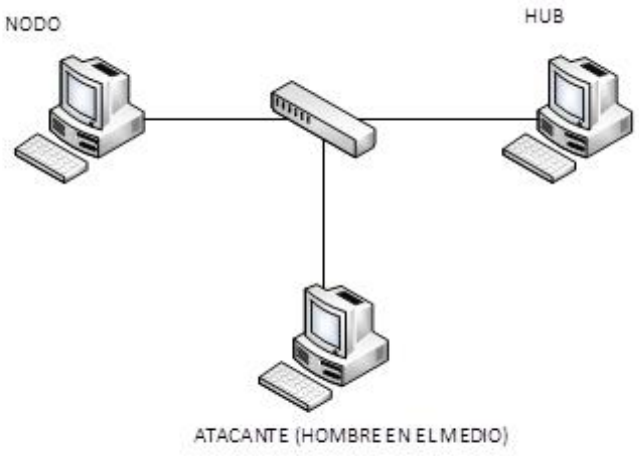


Figura 1.1. Escenario de comunicaciones a implementar.

Es importante aclarar que en este trabajo no se implementarán protocolos de comunicación propios de WBANs como de capa Física (PHY) y de subcapa de Control de Acceso al Medio (MAC), ya que el objetivo de estudio es el protocolo DAA. Además, no se recreará un ambiente WBAN, sino que se utilizará un escenario de comunicación que emule la WBAN y permita probar el protocolo de Asociación. Debido a que actualmente no se dispone de dispositivos 802.15.6, se aclara que se realizará una emulación del proceso de *Broadcast Beacon* tomando campos importantes y necesarios de la trama *Beacon*, acorde al objetivo de estudio, pero no se implementará todo el proceso como se define en el estándar por lo extenso de su alcance.

Debido al amplio contenido teórico, este trabajo de titulación, se limita a la implementación del protocolo DAA, prueba de vulnerabilidades y presentación de una solución teórica. No contempla el Proceso de Desasociación. Además, este trabajo recomendará varios temas futuros que surgirán a partir de esta investigación, los cuales son necesarios para lograr implementar el estándar IEEE 802.15.6 en su totalidad.

1.3 Marco Teórico

1.3.1 INTRODUCCIÓN A WBAN

a. Historia

A partir de los estudios realizados sobre Redes Inalámbricas de Área Personal (*Wireless Personal Area Networks* - WPAN), las cuales apuntaban a transmitir información entre dispositivos cercanos al cuerpo humano, incrementa el interés en investigar sobre redes que tengan mayor velocidad efectiva de transmisión con un menor consumo energético y dentro de un rango más corto [4]. Además, surge la necesidad de cumplir con requerimientos tecnológicos más demandantes, con el fin de prolongar la vida de los dispositivos y lograr una comunicación más próxima al cuerpo humano. De esta manera, el grupo de trabajo IEEE 802 empieza su carrera en buscar un estándar que logre crear Redes Inalámbricas de Área Corporal (*Wireless Body Area Networks* - WBAN).

En enero del año 2006 se creó el grupo de trabajo WG15, dedicado a investigar sobre las posibles vías que alcancen los objetivos planteados para las WBAN [11]. Después, en mayo del 2006 se establece el grupo de interés en WBANs (*Interest Group Wireless Body Area Networks* – IG-WBAN), el cual luego pasaría a ser aceptado formalmente dentro del WG15 del IEEE 802 como un grupo de estudio SW-WBAN. En enero del año 2008 este último sería certificado como un grupo de tarea bajo el 802.15 (TG6). En abril del 2010, se planteó el primer borrador del estándar de comunicaciones de WBANs, el cual logró alcanzar los objetivos planteados de reducción de consumo energético y comunicaciones dentro y sobre el cuerpo humano para aplicaciones médicas y no médicas. Finalmente, la versión aprobada fue revisada y publicada como *"IEEE Standard for Local and Metropolitan Area Networks- Part 15.6: Wireless Body Area Networks "* [1], la cual es una fuente de investigación primordial en este proyecto técnico.

Por otro lado, las WBAN surgen de la evolución de las conocidas Redes Inalámbricas de Sensores (*Wireless Sensor Networks* – WSN). A partir de la creación de las WSN, se realizaron varios estudios que analizaron las características a mejorar en estas redes, con el fin de que sea posible implementar las mismas en un entorno corporal. La gran demanda existente para aplicaciones de monitoreo del cuerpo humano, dieron lugar a investigaciones exhaustivas de requerimientos, indispensables a cumplirse al momento de implementar redes de este tipo en un ambiente corporal [5].

Es así como, las WBAN son un punto clave en el desarrollo de sistemas de monitoreo personal, ligado a varios tipos de aplicaciones que utilizan sensores para detectar

señales relacionadas a la salud de la persona, el ambiente que lo rodea, la posición en la que se encuentra y actividades que realiza. Estos sensores pueden estar ubicados como implantes, parches corporales o adheridos a la vestimenta del usuario y deben ser capaces de detectar, muestrear, procesar y transmitir una o más señales propias del cuerpo humano o de su entorno inmediato [12].

b. Aplicaciones

Como se mencionó anteriormente, la necesidad de crear las WBAN surgió por la gran demanda de implementar aplicaciones de sensores en el cuerpo humano que se comuniquen entre sí, y que puedan mejorar la calidad de vida de los seres humanos. Es importante tomar en cuenta que, debido a la existencia de una interfaz inalámbrica es posible tener aplicaciones más eficientes en costo e implementación [13]. Las WBAN permiten una amplia gama de aplicaciones entre las cuales se pueden tener aplicaciones médicas, monitoreo de signos vitales, militares, entretenimiento, deportivas, etc. En la Tabla 1.1 se detallan algunas aplicaciones de las WBAN.

Tabla 1.1. Aplicaciones WBAN[4].

Aplicaciones WBAN	**Médicas**	**Wearables**	Asma
			Monitoreo de salud
			Puesta en escena de sueño
			Ayuda para entrenamiento deportivo profesional y amateur
			Evaluación de fatiga y preparación para batalla en soldados
		Implantables	Enfermedades cardiovasculares
			Detección de cáncer
		Control remoto de dispositivos médicos	Ambiente asistido para vivir AAL[1]
			Monitoreo de pacientes
			Sistemas de telemedicina
	No médicas		*Streaming* en tiempo real
			Entretenimiento
			Emergencias (no médicas)

En las WBAN se pueden tener dos tipos de nodos principales: los sensores y actuadores. En las aplicaciones, los sensores se encargan de medir parámetros específicos externos o internos del cuerpo humano, como por ejemplo tomar mediciones de temperatura, presión sanguínea, humedad en la piel, ritmo cardiaco, etc. En la Figura 1.2, se muestra un ejemplo de WBAN para monitoreo de pacientes. Por otro lado, se tienen los actuadores, que se encargan de realizar acciones específicas de acuerdo a las

[1] **AAL:** *Ambient Assisted Living*.

mediciones tomadas por los nodos sensores; por ejemplo, enviar una alarma de urgencia, dosificar un medicamento específico y otras que tengan que ver con la interacción directa con el ser humano. Otro elemento importante en las WBAN es el Hub o coordinador, conocido como *Gateway Personal Device,* el cual cumple la función de colectar toda la información transmitida desde los sensores y transmitir a redes remotas a través de otras tecnologías de comunicación tales como Bluetooth, WiFi, Redes celulares 3G, 4G, etc. El coordinador puede ser un elemento próximo al ser humano como por ejemplo *SmartPhone*, Asistente Personal Digital (*Personal Digital Assistant* – PDA), Laptop, etc. En la Figura 1.3 se muestra la infraestructura de un sistema de telemedicina para monitoreo de pacientes.

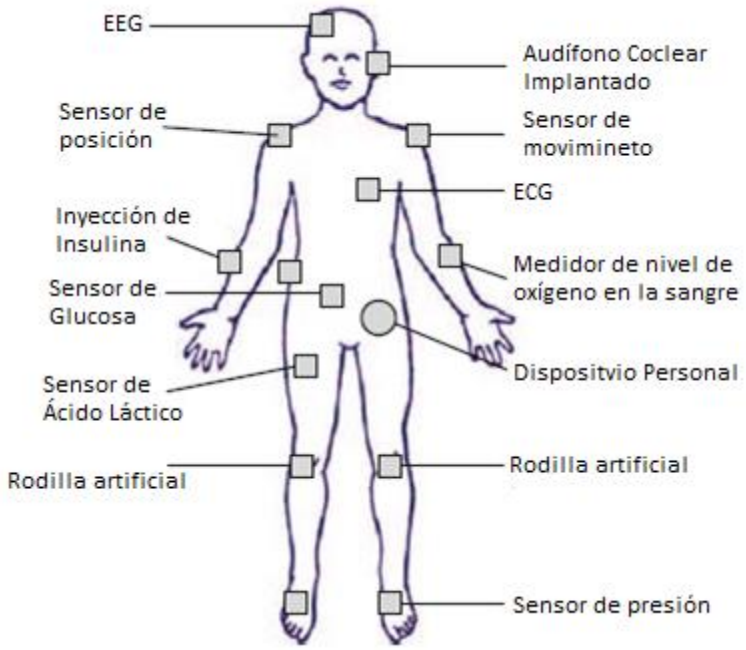


Figura 1.2. WBAN para monitoreo de pacientes[13].

De acuerdo a lo mencionado en esta sección, se realizará un breve análisis de cada una de las aplicaciones indicadas en la Tabla 1.1. Como se observa en esta tabla, se pueden clasificar las aplicaciones en dos grandes grupos: médicas y no médicas.

b.1 Aplicaciones Médicas

Enfocados en que las WBAN cumplan su principal objetivo de mejorar la calidad de vida de los usuarios; gran parte de las investigaciones realizadas en este campo se dirigieron hacia aplicaciones que revolucionen la forma de monitorear las condiciones de salud y síntomas de pacientes de forma remota y en tiempo real [4]. Con el fin de acelerar los diagnósticos sobre enfermedades que afectan gravemente al ser humano, se buscó aplicaciones para tener un control remoto sobre la salud de los pacientes; con ello es

posible que al estar inmersos actuadores en las WBAN se pueda tener una respuesta rápida ante condiciones críticas de salud en tiempo real como nivel elevado de glucosa, ritmo cardiaco anormal, detección de células cancerígenas, presión sanguínea elevada, etc.

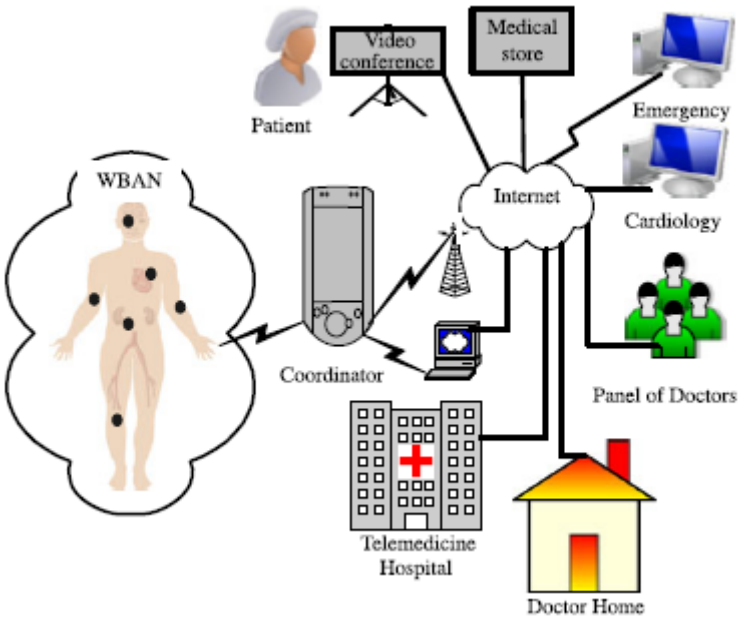


Figura 1.3 Infraestructura para sistema de telemedicina utilizando WBAN[5].

Generalmente, los sensores se encargan de monitorear parámetros fisiológicos de los pacientes; en casos donde estos datos son detectados como condiciones anormales, se envían al *Gateway Personal Device* (un *Coordinator* en el caso de la Figura 1.3*),* el cual a su vez por medio de la red celular o la red inalámbrica a la que pertenezca (*Wireless Local Área Network* - WLAN*)* enviará los datos a través de Internet hacia un centro de monitoreo o el médico de cabecera, quien dará un *feedback* y tomará las acciones pertinentes.

Dentro de las aplicaciones médicas [5], se pueden tener las siguientes:

- **Control de diabetes**

Por medio del monitoreo en tiempo real del nivel de glucosa en la sangre, al identificar un nivel peligroso se puede suministrar una dosis adecuada de insulina por medio de un actuador. De esta forma se previene riesgos de que la persona se desmaye, pierda la circulación o quede ciega para toda la vida.

- **Detección de cáncer**

Algunas células cancerígenas pueden ser detectadas por medio de sensores muy pequeños que sean parte de una WBAN; de esta forma se podrían diagnosticar tumores cancerígenos sin la necesidad de intervenciones quirúrgicas o biopsias. Así, con el diagnóstico continuo y a tiempo se puede tener un periodo más largo para las etapas de análisis y tratamiento [4].

- **Monitoreo cardiovascular**

Las enfermedades cardiovasculares están generalmente asociadas a eventos impredecibles, de anormalidad abrupta de parámetros como presión sanguínea o ritmo cardiaco, mas no a anormalidades continuas. Mediante un apropiado monitoreo ambulatorio y análisis de estos episodios, se puede reducir notablemente la ocurrencia de un infarto de miocardio.

- **Prevención de ataques de Asma**

Gran cantidad de personas que sufren de Asma pueden ser expuestas a condiciones ambientales que contengan agentes alergénicos que les causen ataques de Asma. Esto puede ser evitado gracias a que se puede monitorear agentes alérgenos presentes en el aire, desencadenando una alarma que prevenga a la persona de la presencia de estos agentes en el lugar donde se encuentra.

- **Monitoreo de pacientes**

Las WBAN tienen amplia gama de aplicaciones enfocadas en el monitoreo de signos vitales de pacientes y la provisión de un *feedback* en tiempo real de la información recibida en el lado remoto. Mediante mediciones continuas de presión sanguínea, temperatura corporal, ritmo cardiaco, frecuencia respiratoria, sonidos del pecho y parámetros desde sensores implantados, se puede realizar monitoreo continuo de calidad, que haga posible el procesamiento de los datos fisiológicos a tiempo, logrando así que se pueda suministrar la medicación necesaria cuando se requiera y se ayude en la rehabilitación.

Lo detallado no solo sería de gran ayuda en el monitoreo de pacientes, sino también para aplicaciones de seguimiento pos-tratamiento, investigación médica de traumas y enfermedades crónicas, asistencia médica remota, etc.; todo esto con el propósito de mejorar la calidad de vida del ser humano y la prevención de la muerte.

b.2 Aplicaciones No-Médicas

Entre las aplicaciones No-Médicas, se tiene un conjunto de aplicaciones que ayudan a facilitar el diario vivir de las personas en diferentes ámbitos, entre las cuales se tienen:

- **Ayuda para entrenamiento deportivo**

Se puede utilizar las WBAN para monitorear parámetros de rendimiento físico en entrenamiento, los cuales pueden ser procesados para analizar el progreso del rendimiento, mejorar rutinas de entrenamiento, prevenir lesiones, estimar tiempos de rehabilitación, etc.

- **Usos militares**

En el campo de batalla, mediante el uso de cámaras y sensores biométricos, se pueden tomar mediciones sobre movimientos realizados por soldados, su estado físico, y capacidad de estar listo para el combate. Además, estos datos podrían ser compartidos entre el equipo de batalla, utilizando comunicación entre WBAN con un canal seguro para transmitir dichos datos entre el equipo y su comandante, previniendo así emboscadas.

- **Detección de emociones**

Las emociones expresadas por el ser humano se pueden predecir por la combinación de un conjunto de mediciones que determinen el estado anímico de la persona. Por ejemplo, si se monitorea que la frecuencia respiratoria y el ritmo cardiaco han aumentado, y que el nivel de sudor en las palmas de la mano ha incrementado, se podría interpretar que la persona siente miedo.

- **Entretenimiento**

Este subconjunto de aplicaciones abarca una variedad interesante de aplicaciones enfocadas a juegos interactivos, redes sociales, reproducción de música y video. Se puede utilizar las WBAN para comunicar el dispositivo personal (PD), con micrófonos, auriculares y cámaras que interactúen con las aplicaciones mencionadas.

- **Monitoreo del hogar para emergencias**

Mediante el uso de sensores externos al cuerpo, posibles en las WBAN, se puede monitorear el nivel de monóxido de carbono en el hogar, presencia de llamas, gas propano, etc.; al tener un nivel alarmante se comunicarán con dispositivos próximos al cuerpo humano y notificarán de esta emergencia.

- **Autenticación segura**

Por medio de sensores biométricos o fisiológicos se puede lograr la identificación única de la persona, gracias a funciones de reconocimiento de comportamiento, huellas, iris y patrones faciales. Se puede implementar esta aplicación para contrarrestar la duplicidad y falsificación de identidad.

c. Diferencias de las WBAN con las WSN

Como se explicó al inicio de esta sección, las WBAN surgieron a partir de la necesidad de aplicaciones de WSN dirigidas expresamente al cuerpo humano. Por esta razón, las WBAN deben alcanzar objetivos más exigentes que las WSN, debido a las condiciones propias de propagación dentro (*In-Body*), sobre (*On-Body*) y fuera del cuerpo humano (*Off-Body*). Los canales de comunicaciones para cada uno de los tipos de comunicación mencionados en las WBAN, tienen sus propios requerimientos y cada uno tendrá diferentes limitaciones.

Para comunicaciones *Off-Body*, los dispositivos que se encuentran sobre el cuerpo humano (generalmente el Hub) deben comunicarse con dispositivos que están a distancia de algunos metros del cuerpo humano; esto implica que las pérdidas de propagación en este canal tendrán su modelo propio y dependerán de la distancia a la que se encuentran los dispositivos lejanos, así como a la frecuencia que se transmita. Generalmente se utiliza el canal de 2.4 GHz [14].

Por otro lado, en comunicaciones *On-Body*, los dispositivos que se encuentran sobre la piel del cuerpo humano deberán transmitir información entre ellos; de esta manera, el coordinador o Hub recolectará toda la información transmitida por los sensores o nodos. En este tipo de comunicaciones el canal de comunicaciones tendrá diferentes barreras a superar, teniendo como principales limitantes la ubicación del sensor sobre el cuerpo humano y la posición del cuerpo humano como tal (de pie, sentado, acostado, etc.). En la Tabla 1.2 se pueden observar diferentes valores de pérdidas por propagación según la posición en la que se encuentran los sensores en el cuerpo.

Tabla 1.2. *Path Loss* típico en comunicaciones *On-Body* a *On-Body* [15].

Ubicación del sensor	*Path Loss* [dB]
Pecho	65.3
Mano derecha	44.5
Tobillo derecho	60.9

Finalmente, se tiene la comunicación *In-Body*, la más retadora de todas, debido a las condiciones propias del ser humano y a la limitación de potencia de transmisión de los sensores. El cuerpo humano que contiene fluidos, tejidos y órganos, cada uno de estos elementos con su propio valor de conductividad, constante dieléctrica y características de impedancia, convierten a este canal de comunicaciones en un modelo difícil de estandarizar. Además, debido a su proximidad al cuerpo humano la potencia de transmisión no puede ser elevada, y ésta se encuentra normalizada por la *International Telecommunications Union* (ITU) y la *Federal Communications Commission* (FCC) a un valor máximo de 25 uW de Potencia Isotrópica Radiada Equivalente (PIRE).

Cuando se visualiza a los sensores como una antena transmisora y al cuerpo humano como el canal de comunicaciones, se puede inferir que existirá una potencia radiada por los sensores, la cual será transmitida a través de ondas de radiofrecuencia a través del cuerpo (tejidos, órganos, fluidos, etc.). Es lógico deducir que, por los tejidos del cuerpo humano, la impedancia característica del canal de comunicaciones varía y aumenta, reduciendo así la eficiencia de la transmisión, tal como se menciona en [5].

La potencia de transmisión también está limitada debido a la capacidad del cuerpo (los tejidos) para absorber la energía cuando éste es expuesto a ondas de radiofrecuencia (RF). Esta sensibilidad es medida por la Tasa de Absorción Especifica (*Specific Absorption Rate* - SAR*)*, la cual expresa la capacidad de absorber un watt de potencia por unidad de kilogramo de tejido W/kg y se encuentra normalizada por la FCC a un valor de 1.6 W/kg [4].

Los principales retos para tener sistemas de monitoreo de la salud del cuerpo humano será cumplir con aspectos relacionados al usuario como la capacidad de usar (llevar puesto) los sensores, fácil uso, dar retroalimentación de importancia al usuario, precio, privacidad y seguridad. El cumplimiento de estos aspectos será importante para lograr gran aceptación de estos sistemas; además, estará ligado a cumplir condiciones tecnológicas como operación de bajo consumo energético, integración de sensores, integración del sistema con otras tecnologías, localización e identificación del usuario. En la Figura 1.4 se puede observar la interacción entre los aspectos relacionados al usuario y los retos tecnológicos.

Tomando en cuenta lo detallado anteriormente, se puede identificar que para cumplir con los requerimientos propios de WBAN, éstas se diferencian de las WSN en los siguientes aspectos:

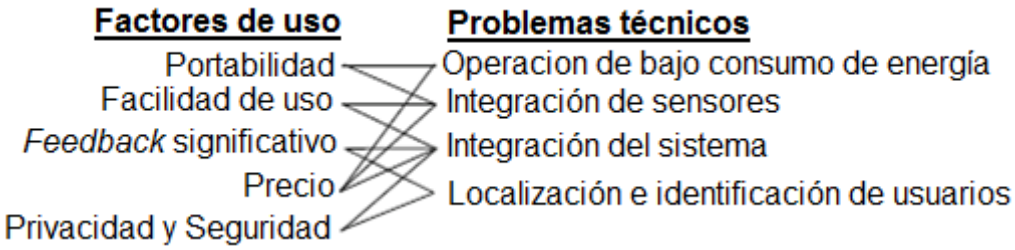


Figura 1.4. Interacción entre factores de uso y problemas técnicos [12].

- **Cobertura**

Refiriéndose a la cobertura de cada una de las redes; las WSN están diseñadas para monitorear el ambiente, por lo cual requiere que tengan cobertura de metros a pocos kilómetros. Por el contrario, las WBAN requieren monitorear el cuerpo humano, esto demanda una cobertura de centímetros a pocos metros (generalmente valores típicos de 1 a 2 metros) [13].

- **Número de nodos**

Las WSN requieren de grandes cantidades de nodos, ya que deben cubrir un área más amplia, mientras que las redes inalámbricas de área corporal tienen un área limitada que cubrir lo que hace que el número de nodos sea menor. Un WBAN típica de uso médico tienen un valor típico de 6 nodos, con capacidad de hasta 256 nodos [16]. Sin embargo, el estándar 802.15.6 establece un número máximo de 64 nodos para una WBAN [1].

- **Exactitud de resultados**

En las redes de sensores inalámbricas este parámetro se basa en la redundancia de nodos, mientras que en las WBAN este parámetro depende de la precisión de los nodos y su robustez.

- **Tareas de los nodos**

Los nodos de las WSN realizan tareas dedicadas dentro de la red, mientras que los sensores corporales pueden ser multitarea. Un nodo WBAN *On-Body*, al estar sobre la piel puede tomar mediciones de temperatura, humedad o nivel de transpiración y presión sanguínea, cantidad de oxígeno en la sangre (SpO2), etc.

- **Tamaño de nodos**

En WSN el tamaño no se encuentra limitado, aunque se prefiere utilizar nodos pequeños. En WBAN es primordial que los nodos sensores sean muy pequeños, de lo contrario, los

sensores no podrían ser utilizados fácilmente como parches, implantes o como parte de la vestimenta, lo cual limitaría la expansión del uso de este tipo de redes de sensores de área corporal.

- **Topología de red**

Generalmente la topología de una WSN tiende a ser fija o estática, mientras que, por la naturaleza propia de las aplicaciones de las WBAN, en el cuerpo humano, debido al movimiento de éste, la topología de red será variable.

- **Tasa de datos**

Este parámetro es homogéneo en la mayoría de casos de WSN. En las redes de área corporal, la tasa de datos será heterogénea, debido a que este parámetro está sujeto a la ubicación de los sensores en el cuerpo humano (*In-Body, On-Body,* y *Off-Body*) y a la distancia a la que se encuentran los nodos entre sí, distribuidos alrededor del cuerpo.

- **Reemplazo de nodos**

En las WSN los nodos pueden ser reemplazados fácilmente, e incluso, la mayoría de veces estos nodos son desechables. Este aspecto cambia totalmente cuando se trata de las WBAN, ya que los nodos implantados no pueden ser reemplazados con facilidad, debido a que no se puede someter al usuario a intervenciones médicas constantemente. De este importante concepto parten características esenciales de las WBAN, como tiempo de vida y consumo energético, las cuales se detallan más adelante.

- **Tiempo de vida de los nodos**

En los dos tipos de redes los sensores deben tener un tiempo de vida de algunos meses hasta años. Pero en WBAN se tiene la limitación del tamaño de las baterías, lo cual disminuye su capacidad. Es así como ésta se convierte en una característica retadora de lograr, pero importante para el éxito de las WBAN.

- **Suministro de energía**

El suministro de energía en las WSN no es una tarea difícil de realizar, pues, generalmente las baterías pueden ser sustituidas fácil y frecuentemente. Pero, al enfocarse en sensores implantados en el cuerpo humano, se puede deducir fácilmente que el acceso a los nodos se complica, lo cual dificulta el intercambio de baterías para suministrar energía a los nodos.

- **Demanda de energía**

Ésta es mucho mayor en WSN debido a la mayor distancia a la que se debe transmitir y a la facilidad que se tiene para suministrar energía. Sin embargo, en WBAN no se cuenta con la misma facilidad para suministrar energía a los nodos, por tal razón, es deseable que el consumo energético (la demanda de energía del nodo para su funcionamiento) sea el menor posible.

- **Búsqueda de fuentes de energía**

Las WSN utilizan fuentes de energía solares y eólicas, debido a que estos sensores son utilizados para monitoreo del medio ambiente, o parámetros que se encuentran en el medio externo (*outdoor*). Los sensores WBAN al estar en contacto con el cuerpo humano, y no necesariamente expuestos al ambiente externo, sólo pueden obtener energía a partir de las vibraciones y el calor del cuerpo.

- **Bio-compatibilidad**

No considerado en la mayoría de aplicaciones WSN. Éste es un factor importante a considerar en nodos implantados y en algunos externos de las WBAN. Es esencial que los sensores a utilizarse en el cuerpo humano no causen reacciones negativas, de otra manera su uso no tendría la acogida esperada.

- **Nivel de seguridad**

Las WSN no requieren de un nivel de seguridad elevado; por la naturaleza propia de la información que transmiten no se requiere un nivel de privacidad tan elevado como en las WBAN. Estas últimas necesitan proteger la privacidad de la información del usuario, lo cual demanda de niveles superiores de seguridad.

- **Impacto de la pérdida de datos**

La información perdida en las WSN es compensada por el alto grado de redundancia que poseen estas redes, pues cuentan con un gran número de nodos que forman parte de la red. En las WBAN, será importante medir este parámetro de transmisión, debido a que se requiere que la comunicación sea en tiempo real. Además, se necesitará implementar mediciones adicionales para asegurar que se cumpla con la calidad de servicio (*Quality of Service – QoS*) requerida.

- **Tecnología inalámbrica**

Bluetooth, ZigBee, GPRS (*General Packet Radio Service*), WLAN, entre otras, son algunas de las tecnologías que se han utilizado para implementar WSNs. Para implementar WBAN, será necesario utilizar tecnologías de bajo consumo energético.

En la Figura 1.5 se muestran diagramas radiales de las características principales en WBAN, WSN y WLAN, con su respectivo nivel requerido, con el fin de comparar y dar una visión más clara de los estrictos requerimientos de las WBAN.

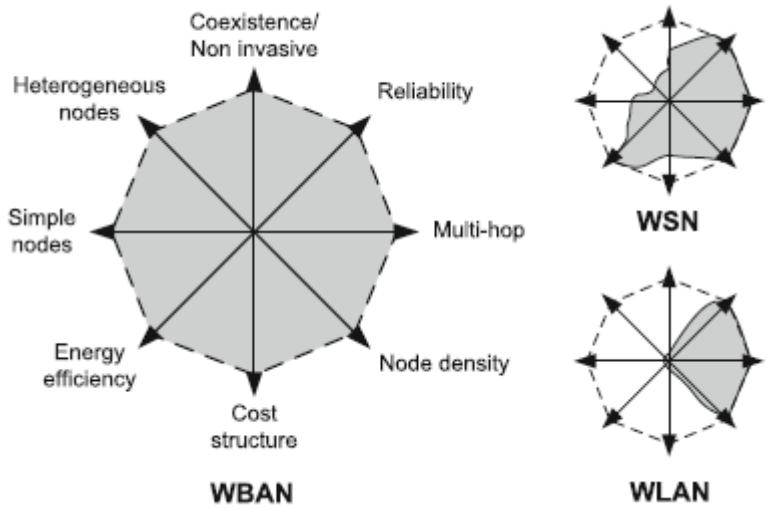


Figura 1.5. Características de WBAN comparadas con WSN y WLAN [13].

1.3.2 WBAN SEGÚN EL ESTÁNDAR IEEE 802.15.6

Esta sección se enfocará específicamente en las WBAN del estándar IEEE 802.15.6, donde se explicará en forma concisa los requerimientos técnicos que deben tener estas redes y sus principales características. El estándar IEEE 802.15.6 en el cual se basa este trabajo se encuentra disponible en el Anexo I.

a. Requerimientos

Los principales requerimientos para el estándar IEEE 802.15.6 serán mencionados a continuación [4], [17], [18].

En primer lugar, la tasa de bits entre los enlaces de las WBAN debe estar dentro del rango de 10 kbps a 10 Mbps; estos valores dependerán de la aplicación y la ubicación de los nodos. Segundo, la tasa de paquetes erróneos (*Packet Error Rate – PER)* debe ser menor al 10%, para la mayoría (95 %) de los mejores enlaces basando su rendimiento en

el PER. Debe existir rapidez en la asociación y disociación de los nodos a la WBAN; se define que estos procesos deberían tomar un tiempo menor a 3 segundos. Además, una WBAN debe ser capaz de soportar hasta 256 nodos en su red, en donde comunicaciones *In-Body* y *On-Body* deben ser capaces de coexistir en la red.

Los nodos deben ser capaces de brindar una comunicación confiable, aun cuando la persona se encuentre en movimiento o realice cualquier actividad como sentarse, caminar, correr, voltear, saltar, agacharse e incluso si la persona se encuentra realizando cualquier actividad con otras personas. Las WBAN deben proporcionar confiabilidad en los enlaces, en donde la capacidad de éstos puede disminuir debido a efectos propios del canal, como desvanecimiento del canal o sombra; de ninguna manera se puede perder comunicación de datos a causa de las condiciones inestables del canal de comunicaciones o interferencia.

Las WBAN requieren soportar pequeñas variaciones al retardo (*jitter*), bajo retardo total (latencia) y alta confiabilidad. Los valores requeridos son:

- Latencia menor a 125 ms para aplicaciones médicas y menor a 250 ms para otras aplicaciones.
- *Jitter* menor a 50 ms.

La capa física debe soportar hasta 10 WBAN co-localizadas, distribuidas en forma aleatoria en un volumen de 6 m^3. Todos los dispositivos deben ser capaces de transmitir a una potencia de 0.1 mW y radiar una máxima potencia de transmisión menor a 1 mW; con lo cual se estaría cumpliendo con la normalización de la FCC para el SAR de 1.6 W/kg [4].

Con el propósito de convergencia con otras redes, las WBAN deben ser capaces de operar en ambientes heterogéneos, donde otras tecnologías de red (estándares de comunicación) colaboren entre sí para recibir la información monitoreada a partir de los nodos sensores.

Las WBAN deben administrar características de QoS, con el fin de que puedan manejar servicios de prioridad, ser seguras y proporcionar confiabilidad por medio de la asignación inteligente de sus enlaces idóneos para un tipo de tráfico determinado. Además, es muy importante que se implemente mecanismos de ahorro de energía, ya que como se mencionó anteriormente, el suministro de energía es una de las mayores limitaciones en las WBAN; esto hace que sea mandatorio que estas redes sean capaces de operar correctamente en condiciones restringidas de energía.

Finalmente, estas redes deben implementar tecnología *Ultra Wide Band* (UWB) con transmisión *Narrow-band,* para de esta forma alcanzar altas tasas de transmisión y cubrir diferentes tipos de ambientes. Por ejemplo, existen aplicaciones médicas en WBAN que demandan tasas de transmisión bastante altas; uno de estos casos es el monitoreo de Electrocardiogramas.

b. Características

En esta sección se detallarán las características que deben tener las WBAN, según el estándar IEEE 802.15.6. Se dará una visión general sobre las características de los nodos que trabajan en estas redes según sus tipos, el número de nodos máximo estandarizado, la topología de red, la arquitectura de comunicaciones, capas estandarizadas y finalmente la seguridad. Todas las características mencionadas serán expuestas según el análisis del estándar mencionado.

b.1 Tipos de nodos

Un nodo es cualquier dispositivo independiente capaz de comunicarse con otros. Los nodos que trabajan en las redes WBAN pueden ser clasificados por varios criterios, como: funcionalidad, implementación en la red y rol que desempeñan. En esta sección se tomarán en cuenta estos tres criterios, para de esta forma, tener una visión más clara de lo que es un nodo, sus características y las funciones que puede realizar.

- **Clasificación por su funcionalidad**

Los nodos que forman parte de una WBAN, tienen diferentes funciones en ella, por esta razón puede ser clasificados de la siguiente forma:

- Dispositivo Personal (*PD*): Llamado también g*ateway* corporal, fuente o *Body Control Unit* (BCU), debido a las funciones que realiza. Se encarga de recolectar toda la información obtenida por medio de los nodos sensores y actuadores, así como también de comunicarse con redes externas, lo cual le da su nombre de *Gateway*. En resumen, transmite la información recolectada hacia el centro de monitoreo remoto.

- Sensores: Su principal función es medir parámetros específicos en el cuerpo humano, sean externos o internos. Lo llevan a cabo por medio de respuestas a estímulos físicos, que son convertidos en datos que son procesados y finalmente adecuados para transmitirse por un medio inalámbrico. Existe un gran número de sensores para diferentes tipos de aplicaciones como, por ejemplo: electrocardiograma,

electroencefalograma, electromiógrafo, glucosa, presión arterial, nivel de oxígeno, movimiento, posición, etc.

- Actuadores: Su función principal es interactuar con el usuario a partir de la información recolectada por los sensores; es decir, proporciona una retroalimentación al sistema según los datos recibidos por lo sensores, desencadenando una acción específica para los datos recibidos.

- **Clasificación por su lugar de ubicación**

Al implementar una WBAN, los nodos pueden ser ubicados en diferentes partes del cuerpo humano. Por ello, se tiene la siguiente clasificación de nodos según el lugar donde se implementan:

- Nodos implantados: Como su nombre lo indica, estos nodos se encuentran implantados en el cuerpo humano, de tal forma que, desempeñan comunicación *In-Body*. Se encuentran dentro de los tejidos, adjuntos a los órganos de monitoreo o simplemente debajo de la piel. Un ejemplo claro de este tipo de nodos, son los marcapasos actuales, que son implantados sin necesidad de cirugía invasiva.

- Nodo para superficie corporal: Este tipo de sensores son los más comunes y simples de utilizar, debido a su facilidad de implementarlos. Se ubican sobre la piel o a una distancia máxima de dos centímetros de ésta. Utilizados para aplicaciones de monitoreo de temperatura, humedad, glucosa, etc.

- Nodos externos: Son nodos que no están en contacto directo con el cuerpo humano, se ubican desde pocos centímetros del cuerpo hasta 5 metros de distancia. Más utilizados para monitoreo del ambiente que rodea al ser humano; por ejemplo: medir niveles de CO, medir la existencia de elementos alergénicos en el aire, existencia de flamas, etc.

- **Clasificación por el rol que desempeñan**

Los nodos de una WBAN se pueden clasificar por el rol que desempeñan dentro de la red. Según el estándar IEEE 802.15.6, se reconocen tres tipos de nodos según su rol, y son los siguientes:

- Nodo: Es la entidad que contiene la arquitectura de capa física (PHY), subcapa de Control de Acceso al Medio (MAC) y que opcionalmente

puede proporcionar servicios de seguridad mencionados en el estándar. Estos nodos están limitados a ejecutar una aplicación específica y comunicarse directamente con el coordinador, pero, no son capaces de retransmitir mensajes recibidos por otros nodos. Sin embargo, se tienen los *Relay nodes* (*relayed*, si se comunica a través de un nodo intermedio; o *relaying*, si es el nodo retransmisor intermedio) los cuales pueden desempeñar la función de retransmisión.

- Nodos retransmisores: Conocidos como *relaying nodes*; en el estándar, son nodos intermedios entre los nodos finales y el Hub, capaces de retransmitir mensajes de otros nodos hacia el PD. Estos nodos también son capaces de detectar datos como un nodo común. Su función es importante en las WBAN, debido a que en algunos casos el Hub se encontrará a distancias que requieran tener un nodo intermedio que transmita la información del nodo final. Por ejemplo, un sensor ubicado en el pie (*relayed node*) necesitará de un *relaying node* que transmita su información al Hub o PD.

- Hub o coordinador: Según el estándar IEEE 802.15.6, se define al Hub como una entidad capaz de tener funcionalidades de nodo y, a su vez, pueda coordinar el acceso al medio y tener control sobre la administración de energía de todos los nodos dentro de una red de área corporal (*Body Area Network – BAN*). Generalmente, es él quien tiene la función de comunicarse con redes externas y ser el g*ateway*. El Hub es un PD, a través del cual todos los nodos de la red se comunicarán.

b.2 Número de nodos

En documentos previos a la publicación del estándar, se consideró un número de nodos en orden de decenas hasta centenas. Pero, como se mencionó anteriormente una WBAN típica tiene 6 nodos, con configuración escalable que posibilita tener hasta 256 nodos. Debido a que se permite tener un solo hub en la WBAN, el estándar IEEE 802.15.6 definió un número máximo de 64 nodos por WBAN. Es importante anotar que en una misma persona pueden coexistir de 2 a 4 WBANs, lo cual hace posible un total de 256 nodos en la red. Las limitaciones no vienen dadas debido a la arquitectura de red, técnicas de transmisión o protocolos de comunicación del estándar, sino que están ligadas al número de canales ortogonales (*slots* de tiempo) que se deberá establecer a partir del tiempo de trama.

b.3 Topología de red

Los elementos de la red, nodos y Hub, están organizados como conjuntos lógicos denominados en el estándar como BANs; dentro de ésta el Hub se encarga de controlar el acceso al medio para cada uno de los nodos y además administrar la energía en la misma. Como se mencionó anteriormente, solo puede existir un único Hub dentro de una BAN y un número máximo de 64 nodos.

Se define en el estándar una topología de red tipo estrella, la cual puede ser de un solo salto o de dos saltos. En la topología de un solo salto (*one-hop star*), los nodos se comunican directamente con el Hub, dándose el intercambio de tramas Nodo-Hub. Por otro lado, en topología de dos saltos (*two-hop extended star*) existe la opción de que los nodos se comuniquen con un nodo intermedio (*relaying node*) antes de comunicarse con el Hub. Es deseable que el Hub se encuentre en una ubicación céntrica, como la cintura, para que la topología estrella sea proporcionalmente distribuida a lo largo del cuerpo. En la Figura 1.6 se puede apreciar la topología tipo estrella de las BAN.

Es importante considerar según la topología de red que se utilice, criterios como interferencia, consumo de energía, retardo en la transmisión, falla de nodo y movilidad. En la Tabla 1.3 se presenta un cuadro comparativo de la variación de estos criterios según la topología que se utilice. Es claro inferir que, una topología de dos saltos puede significar mayor retardo de transmisión comparado con la de un salto; así como también, representará un menor consumo de energía debido a menor potencia de transmisión necesaria para alcanzar el nodo coordinador, pues se tiene nodos intermedios que disminuyen la distancia de transmisión.

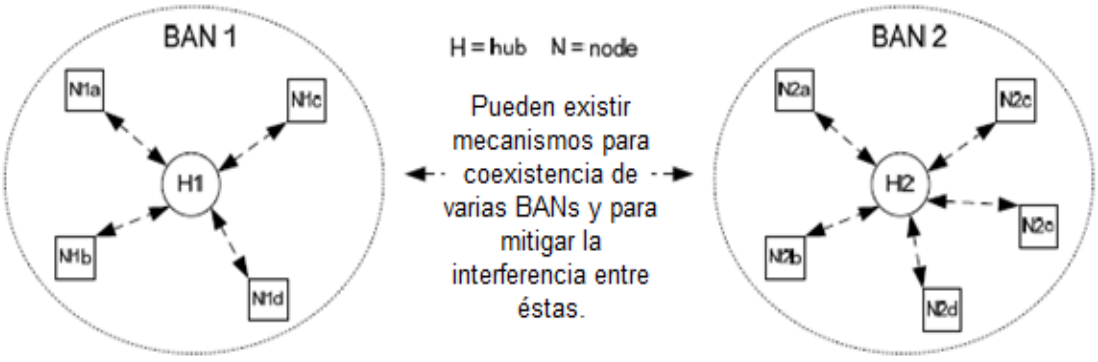


Figura 1.6. Topología WBAN [1].

Tabla 1.3. Cuadro comparativo de topologías WBAN [4].

Criterio	Topología estrella de un salto	Topología estrella de dos saltos
Consumo de energía	Los nodos más alejados al PDA necesitarán mayor cantidad de energía para transmitir la información que los que se encuentran cerca.	En este caso, los nodos lejanos transmitirán a través de los nodos intermedios que se encuentran cerca al PDA, por lo tanto, los nodos lejanos no demandan alto consumo de energía.
Retardo de transmisión	Representan el menor retardo debido a la existencia de un solo salto.	Depende de la configuración de la red. Sin embargo, los nodos cercanos al PDA tienen menor retardo que los que deben transmitir a través de un nodo intermedio.
Interferencia	Nodos más alejados requieren mayor potencia de transmisión lo cual incrementa la interferencia en el sistema.	Debido a la cercanía de los nodos, la potencia de transmisión es mínima. Esto mantiene el nivel de interferencia bajo.
Falla en nodos y movilidad	Solo el nodo que falla será afectado, mas no el resto de la red.	Los nodos que forman parte del nodo con falla serán afectados en la red.

Además, dentro de las topologías mencionadas existen dos métodos de comunicación, modo *Beacon* y modo *non-Beacon*. En el modo *Beacon*, el coordinador de la red transmite periódicamente tramas *Beacon* para definir el inicio y fin de una super trama; de esta forma habilita el control de asociación a la red y el sincronismo de los nodos. En este modo, el ciclo de trabajo está dado por la longitud del período *Beacon*, el cual puede ser configurado por el usuario o utilizar el estandarizado en el IEEE 802.15.6. En el modo *non-Beacon*, los nodos de la red son capaces de enviar tramas al coordinador y utilizar acceso al medio CSMA/CA (*Carrier Sense Multiple Access with Collision Avoidance*) cuando se requiera. En este último modo los nodos necesitan encender e invitar a transmitir al coordinador para que éste reciba los datos que transmiten; pero, debido a que no todos pueden comunicarse a la vez, los nodos deben esperar a que el coordinador los invite a ser parte de la comunicación.

b.4 Arquitectura de comunicación

La arquitectura de comunicación necesaria para implementar una aplicación de WBAN completa, con monitoreo corporal y envío de datos a lugares remotos, puede ser

separada en tres niveles o *Tiers*. Como se menciona en [4] y se ilustra en la Figura 1.7 estos niveles son:

- *Tier I:* Comunicaciones *Intra-WBAN* – Este nivel representa la WBAN propiamente dicha, con los sensores distribuidos a lo largo del cuerpo humano comunicándose entre sí a una distancia máxima de aproximadamente 2 metros. Los datos recolectados por estos sensores y posteriormente procesados por el coordinador, son transmitidos a un Punto de Acceso (*Access Point* - AP) ubicado en el *Tier II*.

- *Tier II:* Comunicaciones *Inter-WBAN* - En este nivel se tienen comunicaciones entre el coordinador y uno o varios APs, los cuales pueden ser parte de la infraestructura o pueden ubicarse de manera que se logre un entorno dinámico para gestión de emergencias. El principal objetivo de este nivel es interconectar las WBAN con redes externas como Internet y redes celulares, logrando así fácil acceso remoto a esta información.

- *Tier III:* Comunicaciones *Beyond-WBAN* – Este nivel de comunicación depende de la aplicación específica que se vaya a implementar. Por ejemplo, un *Gateway* como el PDA pueda unir la comunicación entre el *Tier II* y este *Tier,* con el fin de que sea una interfaz de comunicación entre Internet y una base de datos de pacientes. Puede servir para aplicaciones como almacenamiento y *feedback* de los datos recibidos desde el *Tier I*; además, estas aplicaciones dependerán principalmente de las tecnologías de comunicación que contenga el PS (*Personal Server*) del *Tier I*.

b.5 Capas

El estándar IEEE 802.15.6 define dos capas principales: Capa Física PHY y Subcapa MAC *(Medium Access Control*); de esta manera, asegura comunicaciones inalámbricas alrededor o dentro del cuerpo humano con características de bajo costo, baja complejidad, bajo consumo energético, corto alcance y alta confiabilidad. Estas capas serán obligatorias para la comunicación tanto de Nodo como de Hub, y serán útiles en un canal operativo de comunicaciones en cualquier tiempo.

La Capa PHY se encarga de funciones como activación y desactivación del radio *transceiver*, detección de uso del canal (*Clear Channel Assesment* - CCA), transmisión y recepción de datos. Es importante aclarar que los servicios de seguridad del mensaje ocurrirán en la subcapa MAC, mientras que la generación de llaves seguras se llevará a cabo dentro y/o fuera de dicha subcapa. Más adelante se detallarán tramas PHY las

cuales son necesarias para establecer la comunicación entre Nodo y Hub, así como tramas MAC necesarias para llevar a cabo la asociación de ambas entidades.

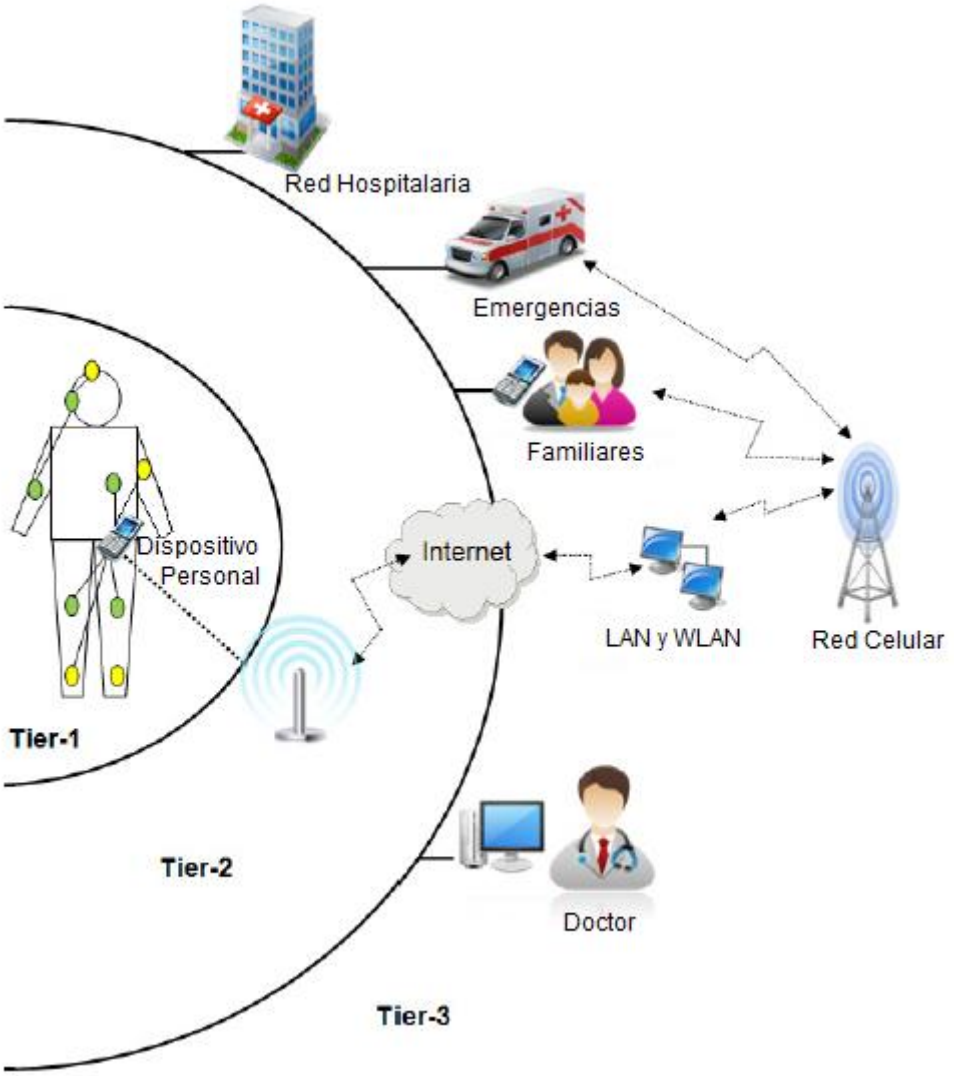


Figura 1.7. Niveles de comunicación en WBAN [4].

b.6 Seguridad

Debido a los requerimientos técnicos y características propias de este estándar, se tiene que las especificaciones de seguridad propuestas para otros tipos de redes no han podido ser implementadas en las WBAN. Los requerimientos mínimos de seguridad serán:

- Gestión segura de las operaciones de encriptación y desencriptación previa a compartir llaves dentro de la WBAN; y adición y remoción de nodos en forma segura por medio de los procesos de Asociación y Desasociación.

- La disponibilidad de red debe ser garantizada, ya que el hecho de que la información no pueda acceder al medio físico podría incurrir en la muerte de un paciente. Se debe asegurar la operación, mantenimiento y capacidad de cambio a otra WBAN en el caso de que la disponibilidad sea esencial.
- La Autenticación de datos es primordial, debido a que las entidades deben asegurarse de que la información recibida proviene de una fuente confiable.
- La integridad de los datos será esencial para asegurar la salud del paciente. Puede existir el caso que los datos sean alterados por un adversario, lo cual pondrá en riesgo la vida del paciente. Para este fin también se utilizan protocolos de autenticación de datos.
- Confidencialidad de datos para garantizar la no divulgación de toda la información recolectada del paciente o usuario.
- Los datos deben ser lo más actualizados posible, garantizando que no han sido retransmitidos por un atacante y las tramas se encuentran en orden.

Debido a que esta característica es el punto donde se enfoca este Proyecto Técnico, en la siguiente sección se tratará a detalle la seguridad en el estándar IEEE 802.15.6. A pesar de que, la seguridad de la información es uno de los puntos más importantes en las redes de comunicaciones, no se ha investigado a profundidad en esta área para las WBAN [4].

1.3.3 NIVELES DE SEGURIDAD EN EL ESTÁNDAR IEEE 802.15.6

La seguridad dentro del estándar IEEE 802.15.6 busca cumplir con un proceso para poder intercambiar mensajes de manera segura. El paradigma de seguridad que sigue el estándar es el mostrado en la Figura 1.8, en el cual se definen tres niveles de seguridad:

- Nivel 0 - Comunicación insegura: El nivel más bajo de seguridad donde los datos son transmitidos a través de tramas inseguras; en este nivel no se aseguran parámetros de integridad, autenticación, confidencialidad y privacidad.
- Nivel 1 - Autenticación, pero no encripción: Los datos son transmitidos a través de tramas autenticadas, pero no encriptadas, de tal forma que garantiza integridad y autenticación de las tramas, pero no privacidad y confidencialidad.

- Nivel 2 - Autenticación y encripción: Nivel más alto de seguridad que garantiza todos los parámetros que no lo hace el nivel 0. Las tramas son transmitidas de forma autenticada y encriptada.

Los nodos y Hub, pueden comunicarse de manera segura o insegura, para lo cual deberán pasar por diferentes estados en la subcapa MAC, al nivel de seguridad seleccionado, como se muestra en la Figura 1.9.

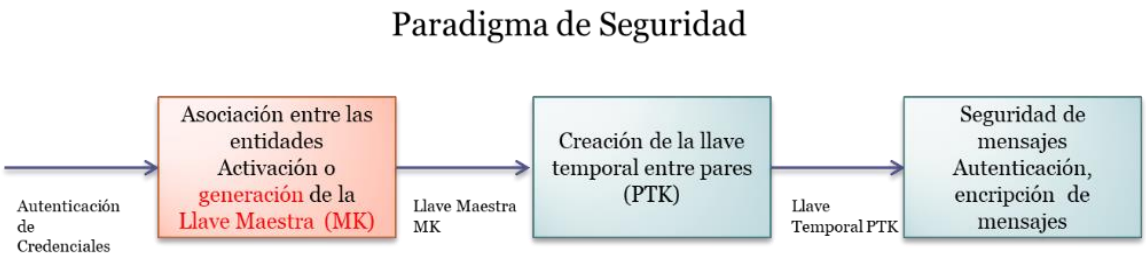


Figura 1.8. Paradigma de Seguridad IEEE 802.15.6 [1].

Los servicios de seguridad que buscarán las entidades será:

- Negociación del conjunto de seguridad deseado
- Generación de la llave maestra (MK).
- La MK generada no se deroga hasta un proceso de Desasociación de las entidades.
- La MK sirve para creación de la llave temporal entre pares (PTK), con su consecuente uso en autenticación y encripción de mensaje.

Dentro de un proceso de Asociación se utilizarán algunos servicios adicionales como:

- Protocolo Diffie-Hellman
- Encripción basada en Curva Criptográfica Elíptica
- Estándar de Encripción Avanzada AES
- Codificación basada en bloques CMAC

Todo esto para cumplir con los requisitos entre dos entidades y poder cumplir con el paradigma de seguridad establecido en el estándar.

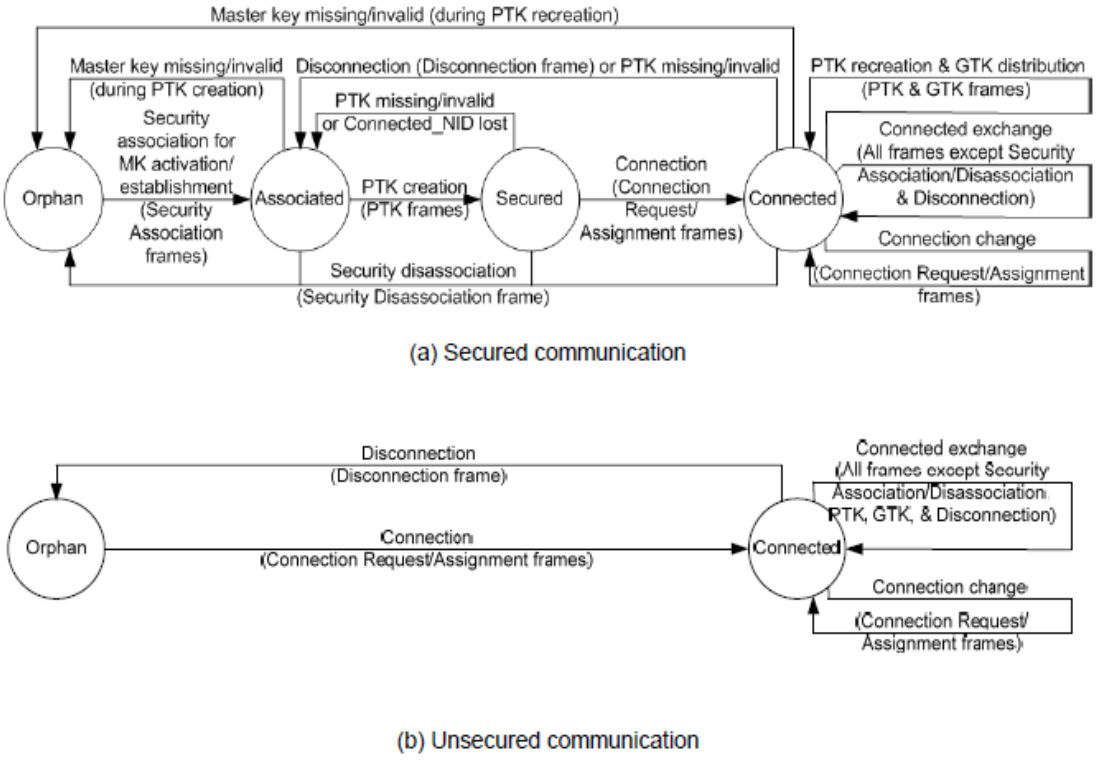


Figura 1.9. Diagrama de estados MAC y seguridad [1].

1.3.4 SERVICIOS DE SEGURIDAD SEGÚN EL ESTÁNDAR IEEE 802.15.6

a. Protocolo Diffie-Hellman

La criptografía asimétrica fue propuesta por primera vez en 1976 por Whitfield Diffie y Martin Hellman, pioneros en el desarrollo de algoritmos criptográficos [6]. La propuesta de criptografía asimétrica, también llamada criptografía de llave pública, se convirtió en un evento histórico en el desarrollo de los sistemas criptográficos. El cifrado propuesto por Diffie y Hellman basa su complejidad en el uso de funciones matemáticas y no en operaciones de patrones de bits. El uso de operaciones sobre los bits y sus patrones ya no eran suficientes para mantener segura la información.

La principal característica de un cifrado asimétrico es el uso de dos claves o llaves separadas y el intercambio de una de ellas. En la criptografía simétrica se utiliza una sola llave entre las dos entidades a comunicar. Esto hace que la criptografía asimétrica tenga importantes implicaciones en la integridad, autenticación y no repudio [19].

Para desarrollar un cifrado asimétrico, primero, cada entidad a comunicarse debe generar sus llaves públicas y privadas. Cuando entre la entidad A y la entidad B se quiere

compartir un mensaje, tanto la entidad A como la entidad B deben compartir primero sus llaves públicas. Las llaves públicas estarán a disponibilidad de cualquier entidad que tenga acceso al canal de comunicación. Una vez compartidas las llaves públicas si la entidad A debe cifrar el contenido de ese mensaje hace uso de la llave pública de la entidad B. Pero se podría pensar que como las llaves públicas se comparten a cualquier entidad se podría descifrar el mensaje [20]. Ese mensaje únicamente podrá ser descifrado por la entidad que tenga la llave privada, tal como se ilustra en la Figura 1.10.

La complejidad y la seguridad de la criptografía asimétrica radica en la obtención de estas llaves criptográficas. La llave privada es una clave secreta que solo la conoce la entidad que la generó. La llave pública es la que puede ser obtenida por cualquier entidad. Esta llave pública se genera a partir de la llave privada. La generación de la llave pública se hace en base a funciones matemáticas, evitando que la entidad que tenga la llave pública pueda descubrir cuál es la llave privada, con lo que se evita que los atacantes descubran los mensajes cifrados con este método. Esto permite obtener un método *one way* de generación de claves. Para la generación de llaves públicas se tienen algunos métodos como Diffie-Hellman, DSA, ElGamal, Merkle-Hellman, Goldwasser-Micalli, entre otros [21]. Los principales métodos para la generación de llaves públicas son RSA y curva elíptica.

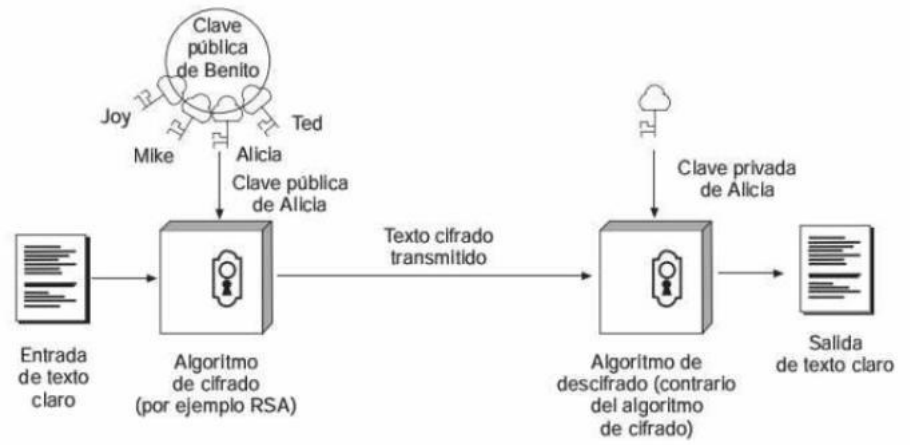


Figura 1.10. Cifrado Asimétrico [20].

b. Curva Criptográfica Elíptica

Las curvas elípticas fueron desarrolladas como funciones algebraicas y por sus características especiales han sido estudiadas por más de un siglo. Han sido utilizadas en la creación de llaves públicas para que dos entidades cumplan con el protocolo Diffie-Hellman.

La primera vez que las curvas elípticas se utilizaron para criptografía fue en 1985. El primer modelo de criptografía utilizando curva elíptica lo propusieron Neal Koblitz y Victor Miller [6] y [22]. Después de estudios y comprobaciones, la curva elíptica fue estandarizada por múltiples organizaciones alrededor de 1990, con su posterior aparición comercial.

La principal razón de la aparición y aceptación de los sistemas criptográficos utilizando curva elíptica se debió a que otorgaba el mismo nivel de seguridad con menor cantidad de bits que otros sistemas como RSA. La computación de las llaves utilizando curva elíptica es menor, siendo ésta otra de las razones de la superioridad de la curva elíptica con respecto al algoritmo RSA. La curva elíptica está basada en el problema del logaritmo discreto (*Elliptic Curve Discrete Logarithm Problem* - ECDLP) [6], y [23] que permite el uso de la curva elíptica en sistemas criptográficos asimétricos.

La familia de curvas elípticas tiene entre sus principales características la de ser curvas regulares[2] no singulares[3] [6] y [23]. Las curvas elípticas geométricamente no tienen auto intersecciones. El conjunto de puntos pertenecientes o soluciones a la curva elíptica forman un grupo Abeliano [6] y [23]. Un grupo es Abeliano si cumple la propiedad conmutativa.

La curva elíptica se define en base a la Ecuación 1.1.

$$y^2 = x^3 + ax + b$$

Ecuación 1.1. Ecuación general de una curva elíptica.

Una curva elíptica válida debe asegurar que la Ecuación 1.1 contenga tres distintas soluciones [6], [23] y [24]. Esta condición se la controla con el discriminante que describe la Ecuación 1.2.

$$4a^3 + 27b^2 \neq 0$$

Ecuación 1.2. Discriminante de una curva elíptica.

Dentro de las operaciones válidas sobre puntos soluciones a la curva elíptica están únicamente la suma de un punto P y un punto Q, y el doble de un punto P.

[2] **Curva elíptica regular:** Tiene forma geométrica elíptica.
[3] **Curva elíptica no singular:** Un punto de coordenada X puede tener varias soluciones en Y.

La suma de dos puntos, siendo P diferente de $-Q$ se la puede analizar geométricamente y matemáticamente. Geométricamente la suma de dos puntos se la consigue trazando una recta que atraviese por los puntos P y Q. Debido a las características de la curva elíptica esta línea trazada a través de P y Q debe intersecar en un punto R también solución a la curva elíptica [23] y [24]. La solución de la suma de los dos puntos es la reflexión en el eje x del punto R. Un ejemplo del proceso de la suma de dos puntos está dado en la Figura 1.11.

El análisis matemático para la suma de dos puntos está definido en las Ecuaciones 1.3, 1.4 y 1.5.

$$x_R = \lambda^2 - x_1 - x_2$$

Ecuación 1.3. Ecuación de reflexión en el eje x del punto R.

$$y_R = -y_1 + \lambda(x_1 - x_R)$$

Ecuación 1.4. Ecuación de reflexión en el eje y del punto R.

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1}$$

Ecuación 1.5. Ecuación de la pendiente de la recta.

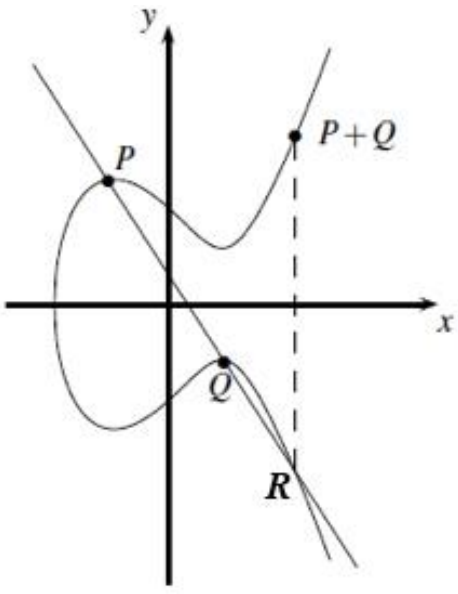


Figura 1.11. Ejemplo de la suma de dos puntos soluciones a la curva elíptica.

Donde x_1, y_1, x_2, y_2 son coordenadas cartesianas de los puntos P y Q respectivamente, y λ es la pendiente de la recta.

La segunda operación definida es el doble de un punto. Geométricamente el doble del punto difiere del análisis realizado en la suma de dos puntos. Para realizar el doble del punto se traza una tangente desde el punto a doblar con el objetivo que interseque en un nuevo punto solución a la curva elíptica. El punto doble será la reflexión en el eje x de esa intersección [23]. Un ejemplo del proceso realizado para doblar el punto viene dado en la Figura 1.12.

Para encontrar el doble de un punto matemáticamente se aplican las Ecuaciones 1.6, 1.7 y 1.8.

$$x_R = \lambda^2 - 2x_1$$

Ecuación 1.6. Ecuación de reflexión en el eje x del punto R.

$$y_R = -y_1 + \lambda(x_1 - x_R)$$

Ecuación 1.7. Ecuación de reflexión en el eje y del punto R**.**

$$\lambda = \frac{3x_1^2 + a}{2y_1}$$

Ecuación 1.8. Ecuación de la pendiente de la recta.

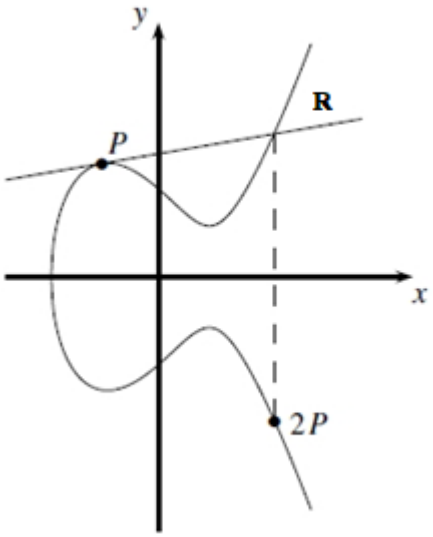


Figura 1.12. Ejemplo del doble de un punto.

Existe un punto que permitiría cumplir con las propiedades mencionadas en las Ecuaciones 1.9 y 1.10.

$$P + O = P$$

Ecuación 1.9. Propiedad del elemento neutro en curva elíptica.

$$2P = 0$$

Ecuación 1.10. Propiedad cancelativa en curva elíptica.

La curva elíptica se define sobre un campo Z_p con la Ecuación 1.11 [6], [23] y [25], en la que p es un número primo mayor a 3 ($p > 3$). El conjunto finito Z_p para curva elíptica está definido en la Ecuación 1.12.

$$y^2 = x^3 + ax + b \quad (mod\ p)$$

Ecuación 1.11. Ecuación general de una curva elíptica sobre campos finitos.

$$Z_p = \{0,1,2, \dots \dots, p-1\}$$

Ecuación 1.12. Conjunto finito sobre el cual estará definida la ECC.

Una curva elíptica sobre un campo finito al igual que en la curva elíptica sobre los números reales tiene que cumplir con el discriminante definido en la Ecuación 1.13.

$$4a^3 + 27b^2 \neq 0 \quad (mod\ p)$$

Ecuación 1.13. Discriminante de una curva elíptica sobre campos finitos.

Al operar con números pertenecientes a un conjunto finito, su valor podría exceder o disminuir dentro del rango del campo, lo que le dejaría fuera del conjunto. Esto no es ningún problema ya que se puede encontrar un equivalente de ese número en el conjunto Z_p.

Dentro de una curva elíptica sobre Z_p, los números soluciones a la curva también forman un conjunto finito. Para que un valor en la coordenada Y sea solución tiene que ser elevado al cuadrado y generar un residuo cuadrático. Se dice que un número es residuo

cuadrático si y solo si tiene raíz cuadrada dentro del campo Z_p [6]. Dentro de un campo Z_p la cantidad de residuos cuadráticos viene dada por la Ecuación 1.14.

$$n = \frac{p-1}{2}$$

Ecuación 1.14. Cantidad de residuos cuadráticos para soluciones de ECC.

Donde n es la cantidad de residuos cuadráticos que tiene el conjunto finito Z_p. Esta expresión no determina qué valores son residuos cuadráticos. Un número es residuo cuadrático si cumple con la Ecuación 1.15.

$$x^{\frac{p-1}{2}} = 1 \quad en\ Zp$$

Ecuación 1.15. Condición para hallar un residuo cuadrático.

Esto implica que no todos los valores dentro del conjunto Z_p pueden ser coordenadas (x,y) soluciones de la curva elíptica.

Todo esto conlleva a que la generación de llave pública utilice todos estos procesos y se obtenga basándose en la Ecuación 1.16.

$$S_k \times G = P_k$$

Ecuación 1.16. Generación de llave pública.

Donde:

- S_k: Llave privada.
- G: Punto solución de la ecuación de la curva elíptica.
- P_k: Llave pública.

Como se había mencionado, las únicas operaciones válidas para puntos pertenecientes a la curva elíptica son la suma de dos puntos y el doble de un punto. Para poder encontrar esa llave pública se debe realizar una serie de operaciones consecutivas hasta llegar a la relación que se tiene en la Ecuación 1.16.

Hay que recalcar que las ecuaciones definidas desde la Ecuación 1.2 a la Ecuación 1.8, son válidas para curvas elípticas sobre campos finitos siempre y cuando se opere en base a matemática modular [23] y [25] y sobre un campo finito Z_p. Dentro de las

Ecuaciones 1.5 y 1.8 se visualizan divisiones existentes, pero dentro de matemática modular no existe la figura de división [24] y [26], por esta razón es necesario la operación de multiplicación de un valor por el inverso del otro. El inverso de un número dentro de matemática modular se define en base a la Ecuación 1.17, válido siempre y cuando p sea un valor primo mayor a tres.

$$\frac{1}{x} = x^{p-2} \quad en \ Z_p$$

Ecuación 1.17. División dentro de números finitos primos.

Si se requeriría generar una P_k, bajo el método de la curva elíptica, para el caso de una llave privada $S_k = 7$. Las operaciones para llegar a este equivalente podrían ser $G,\ 2G,\ 4G,\ 5G,\ 6G,\ 7G$. En esta opción se realizarían dos operaciones doblando el punto y tres operaciones de suma de dos puntos. Otra opción podría ser $G,\ 2G,\ 3G,\ 6G,\ 7G$. En esta segunda opción se puede ver que es menor la cantidad de iteraciones para llegar a la relación de la llave privada. El número de iteraciones tendrá repercusiones directas en el costo de procesamiento que necesitarán los sistemas para la generación de llaves públicas. Cualquier método utilizado para llegar a esa relación da exactamente el mismo resultado y por consiguiente la misma llave pública.

Las soluciones a la curva elíptica en un campo finito Z_p, también forman un grupo Abeliano. Esto implica que cualquier operación realizada en un punto definido de la curva elíptica da como resultado otro punto en la curva elíptica que incluye el punto al infinito. La seguridad proporcionada por la curva elíptica se debe a que, a pesar del conocimiento de la clave pública y el punto de partida de la curva elíptica, conocer la clave privada es muy complejo. Con un punto inicial de la curva elíptica, las posibles formas de alcanzar la clave pública son innumerables. Esto, junto con la gran cantidad de combinaciones proporcionadas por la cantidad de bits de una clave privada, ofrece un alto nivel de seguridad [26].

Para el estándar IEEE 802.15.6 el número de bits de la clave privada es 256. El cálculo de la clave pública a partir de una clave privada es relativamente simple con respecto al proceso inverso; siendo un sistema prácticamente irrompible por los sistemas actuales. Los parámetros utilizados en el estándar IEEE 802.15.6 se especifican para Curve P-256 que están estandarizados en FIPS Pub 186-3[1].

- $p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$

- $r = 1157920892103562487626974469494075735299969552241357603424222506$ 1068512044369
- $a = p - 3$
- $b = 5ac635d8aa3a93e7b3ebb$d55769886bc651d06b0cc53b0f63bce3c3e27d2604b
- $Gx =$ 6b17d1f2e12c4247f8bce6e563a440f277037d812deb33a0f4a13945d898c296
- $Gy =$ 4fe342e2fe1a7f9b8ee7eb4a7c0f9e162bce33576b315ececbb6406837bf51f5
- $1 < Sk < r - 1$

Donde:

- p: Número primo, valor del campo finito.
- r: Orden del punto base G.
- a y b: coeficientes de la curva P-256.
- Gx y Gy: coordenadas iniciales de la curva

c. Estándar de Encripción Avanzada (AES)

AES pertenece a una clasificación de cifrado en bloque simétrico al igual que DES, 3DES e IDEA. La aparición de AES fue producto del concurso organizado por el NIST (Instituto Nacional de Estándares y Tecnología) en 1997, luego de que el NIST no haya estandarizado 3DES por su lentitud [27]. El concurso público solicitaba la creación de un algoritmo de encripción. Las condiciones del concurso incluían:

- Ser tan robusto o más que 3DES (llave de 112 bits).
- Debe ser cifrador simétrico de bloques de 128 bits de datos.
- Permitir longitudes de llave mayores que DES y 3DES: 128, 192 y 256 bits.
- Eficiencia computacional e idoneidad para hardware y software.

Después de algunas etapas de selección, en octubre del año 2000, el NIST elige al algoritmo de Rijndael. El estándar de encripción publicada en FIPS PUB 197 de encripción procesa bloques de longitud de 128 bits, utilizando llaves de longitudes estandarizadas como 128, 192 y 256 bits [27] y [28]. Los creadores del algoritmo le dieron el nombre con el cual se conoció dentro del concurso, el algoritmo se llamaba Rijndael por Joan Daemen y Vincent Rijmen.

Las principales características de este algoritmo de encripción son:

- No es de tipo Feistel[4].
- Tamaño del bloque de texto: 128 bits.
- Tamaño de clave: 128, 192 y 256 bits.
- Número de etapas: depende del tamaño de la clave.
- Usa una Caja S, similar a las empleadas en el DES.

Tomando en cuenta estas características, el algoritmo funciona en base a lo indicado en la Figura 1.13, donde cada etapa y ronda conlleva una serie de procesos tales como:

- Sustitución de *bytes*
- Desplazamiento de filas
- Mezcla de columnas
- Suma de clave de etapa

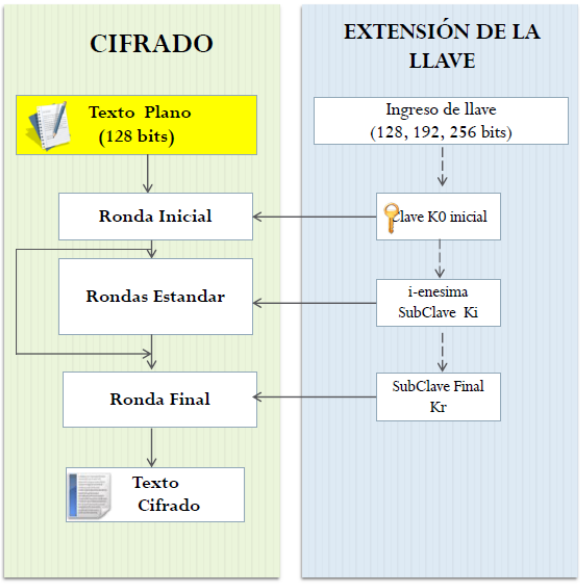


Figura 1.13. Proceso de codificación con AES [29].

Estos procesos se realizan de manera iterativa hasta obtener el texto cifrado; el texto ingresado será dividido en bloques de 128 bits y tratado de la misma forma. Las claves generan palabras, estas palabras deben ser múltiplos de 4 con el propósito de formar la caja S llamada extensión de llave. La sustitución se encarga de tomar los *bytes* del texto plano sumado a la clave y reemplazarlos con algún valor de las llaves expandidas de la

[4] **Feistel:** Algoritmo simétrico por rondas, realiza las mismas operaciones en igual cantidad tanto para la encripción como desencripción.

caja S. El desplazamiento de filas se realiza como una operación de permutación para obtener el efecto deseado [20]. La mezcla de columnas consiste en multiplicar cada columna de la matriz de entrada por una matriz constante. En la suma de clave de etapa se realiza una operación entre el resultado de la Mezcla de columna, y las palabras correspondientes a la ronda provenientes de la extensión o expansión de llaves.

Todo este proceso se lo puede visualizar a lo largo de las 10 etapas en la Figura 1.14.

Dentro de los modos de operación estandarizados y más comunes que tiene la encripción con AES están:

- *Electronic Codebook* (ECB)
- *Cipher Block Chaining* (CBC)
- *Propagating Cipher Block Chaining* (PCBC)
- *Cipher Feedback* (CFB)
- *Output Feedback* (OFB)
- *Counter* (CTR)

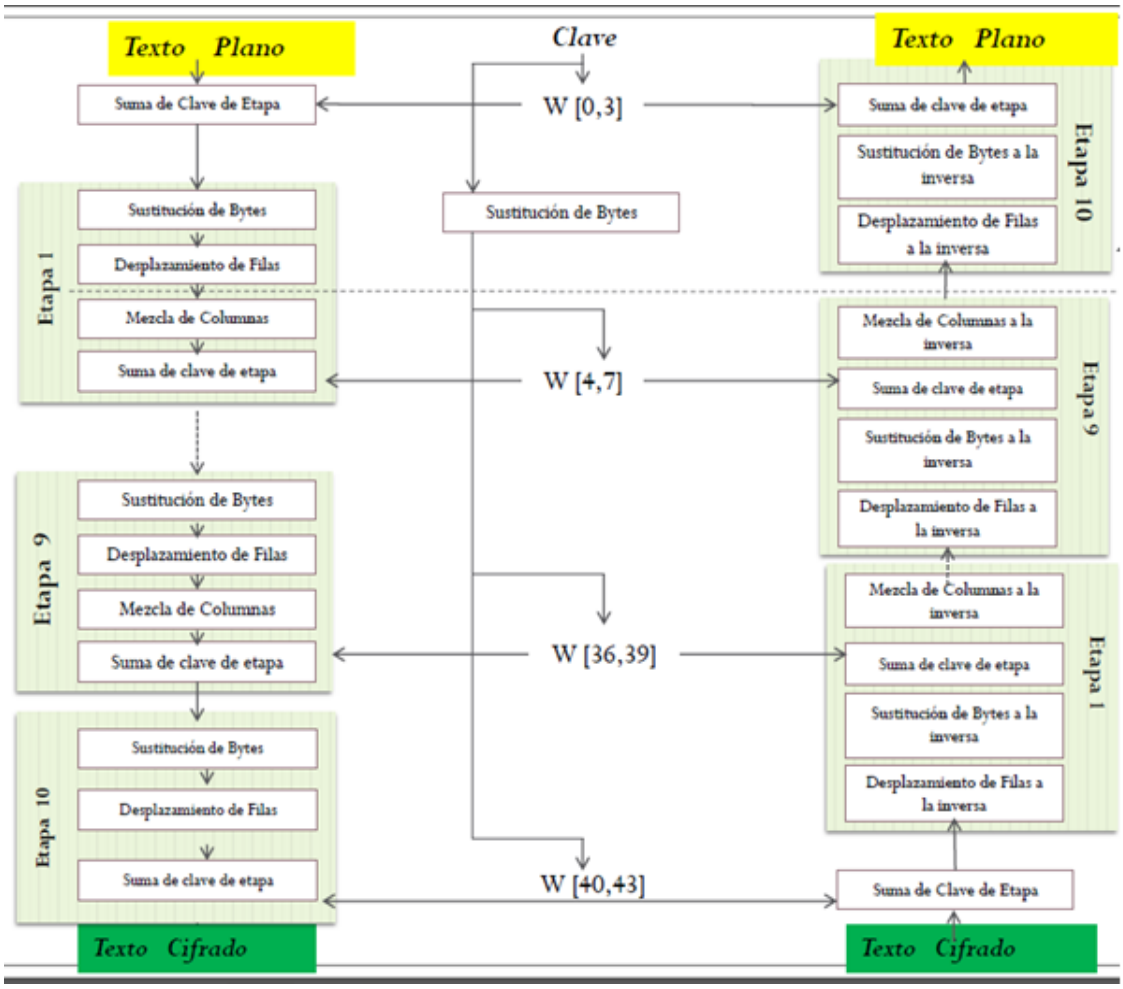


Figura 1.14. Etapas codificación AES [29].

Para este trabajo será necesario entender el funcionamiento del modo de cifrado en cadena (CBC) sin *Padding*. El concepto de *Padding* indica que cuando existan mensajes incompletos y no múltiplos de 128 bits, el algoritmo de encripción autocompletará los mensajes. Con el objetivo de que todos los mensajes sean codificados de la misma forma, independientemente de que se codifiquen varias veces, es necesario que no se realice autocompletación de mensajes. Además, dentro de CMAC como se verá a continuación, existe un manejo particular para mensajes incompletos siendo innecesario colocar *Padding* dentro de los modos de encripción [30].

El modo CBC fue creado por Ehrsam, Meyer, Smith y Tuchman. Este método principalmente otorga seguridad al hacer que sus mensajes se operen con XOR con el texto cifrado anterior [28] y [30]. Esto se realiza para que cada nuevo mensaje dependa el anterior dando fortaleza al ya seguro método de encripción AES. Para el primer bloque se utiliza un vector de inicialización reemplazando a un mensaje encriptado inicial, como se observa en la Figura 1.15. Si este vector de inicialización no es seteado de manera fija podría ser aleatorio, este efecto no es deseado para los objetivos de este trabajo.

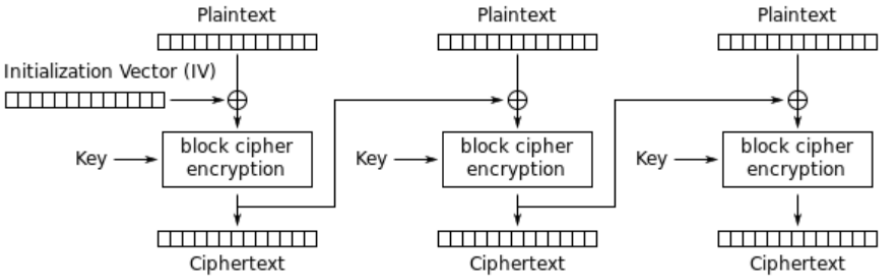


Figura 1.15. Encripción en modo CBC [28].

CBC ha sido el modo de operación más comúnmente utilizado. Sus principales inconvenientes son que la encriptación es secuencial, al ser secuencial es por ende más lento y no se puede realizar procesos de encripción en paralelo.

d. AES-CMAC

Codificación creada por el NIST basándose en su predecesor el *One-Key* CBC MAC1. (OMAC1) es una mejora del modo de encadenamiento *extended Cipher Block* (XCBC) presentado por el vector de constantes [XCBCa, XCBCb], cuyas componentes toman el nombre de "*Black and Rogaway*" gracias a los autores[5] que ayudaron a la mejora de este algoritmo como se detalla en [31]. XCBC a su vez es una mejora del *Cipher Block*

[5] **Phil Rogaway** proporcionó los valores de las constantes de XCBC-MAC. **John Black** detalló las especificaciones para la corrección del algoritmo y proporcionó casos de prueba.

Chaining-Message Authentication Code (CBC-MAC). XCBC aborda de manera eficiente las deficiencias de seguridad de CBC-MAC, y OMAC1 reduce de manera eficiente el tamaño de clave de XCBC. Todas estas mejoras conllevaron al resultado de AES-CMAC, donde su principal diferencia es el método de encripción. AES provee una seguridad mucho más fuerte que sus predecesoras[32].

AES-CMAC otorga mayor garantía en la integridad de datos que lo que ofrece un *ckecksum* o un código de detección de errores. Estos últimos verifican si hubo alteración de datos no intencional o accidental, debido a problemas con el canal de comunicaciones. Mientras que AES-CMAC detecta la modificación intencional que tiene como propósito el corromper los datos existentes en una comunicación.

AES-CMAC logra un objetivo de seguridad de igual o mejor rendimiento del que otorgan la seguridad de códigos que utilizan funciones *hash* como HMAC[6]. AES-CMAC se basa en un cifrado simétrico de bloque de claves, AES y HMAC se basan en una función *hash*, por ello AES-CMAC es más apropiado para algunos sistemas de información por las ventajas que tiene sobre los mencionados [32] y [33].

AES-CMAC utiliza el estándar de encriptación avanzada [NIST-AES] como un bloque de construcción. Para generar un MAC, AES-CMAC toma una clave secreta, un mensaje de longitud variable y la longitud del mensaje en octetos como entradas y devuelve una cadena de bits fijos.

Los procesos de AES-CMAC se basan en CBC-MAC básico. En un mensaje M, para ser autenticado, el CBC-MAC se aplica a M. Hay dos casos de operación en CMAC. Las Figuras 1.16 y 1.17 ilustran el funcionamiento de AES-CMAC en ambos casos. Si el tamaño del bloque de mensaje de entrada es igual a un múltiplo de 128 bits, el último bloque se operará junto a K1 con OR exclusivo antes de pasar por la encripción final en AES. A su vez, si el último bloque no es múltiplo de 128 bits éste se rellenará con 10 ^ i hasta completar el tamaño del bloque, después pasará a operarse junto a K2 con OR exclusivo. El resultado de la operación entregará el mensaje M codificado con AES-CMAC.

- AES_K es la encripción con AES y llave K
- Mensaje M, dividido en bloques M_1, M_2, M_n
- K1 subllave para mensajes completos.
- K2 subllave para mensajes incompletos.

[6] **HMAC:** Código de Autentificación de Mensajes en clave *Hash*

- T resultado de codificación con CMAC_AES de longitud específica.

K1 se usa para el caso en que la longitud del último bloque sea igual a 128 bits. K2 se usa para el caso donde la longitud del último bloque sea menor que la longitud del bloque, esto es menor a 128 bits [32] y [33].

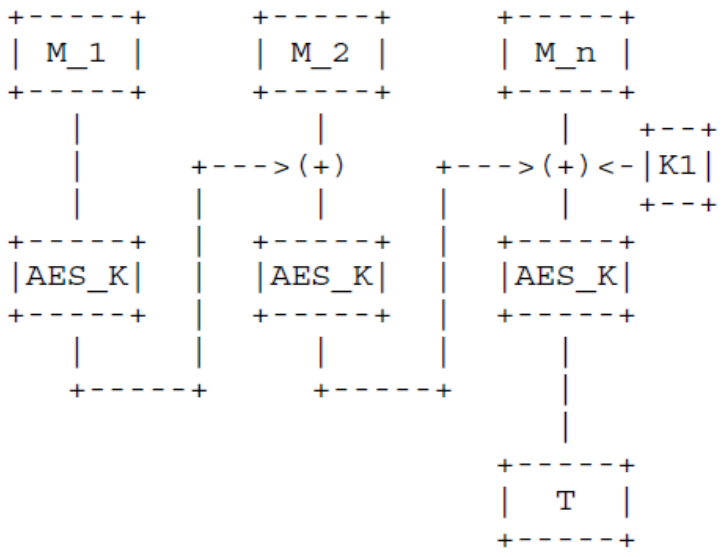


Figura 1.16. AES-CMAC para mensajes completos[32].

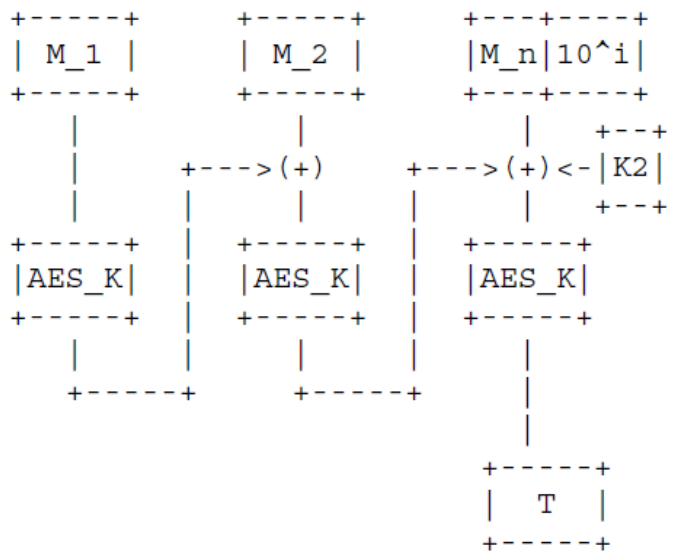


Figura 1.17. AES-CMAC para mensajes incompletos [32]**.**

1.3.5 PROTOCOLOS DE ASOCIACIÓN EN IEEE 802.15.6

El proceso de Asociación en el estándar involucra la compartición de la *Master Key* (MK), la cual se genera a partir del protocolo de Asociación. En el estándar se definen cuatro protocolos de Asociación, los cuales son: *Unauthenticated Association, Public Key hidden Association, Password Authenticated Association, Display Authenticated Association*. El principal objetivo de estos protocolos es compartir la MK entre las entidades Nodo y Hub. Todos los protocolos de Asociación hacen uso de la ECC, CMAC y los demás servicios de seguridad mencionados anteriormente; y a pesar de que son similares, cada uno de estos protocolos tendrán sus requerimientos particulares para poder llevar a cabo su propio proceso. A continuación, se citan los requerimientos propios de cada uno de los protocolos de Asociación [1].

- ***Unauthenticated Association:*** Este protocolo no tiene un requerimiento específico que lo diferencie de los demás protocolos ya que es el más básico.
- ***Public Key hidden Association:*** Se requiere de la compartición previa de la PK del Nodo hacia el Hub, enviando esta información por un canal fuera de banda seguro, que no está relacionado con la comunicación principal. Adicionalmente, el Hub debe poder almacenar dicha POK (*Pre-shared Out-of-band Key*) de los diferentes nodos.
- ***Password Authenticated Association:*** Requiere la compartición previa de la llave PW citada en el proceso de este protocolo de Asociación en [1].
- ***Display Authenticated Association:*** Para este protocolo se requiere que las entidades Nodo y Hub tengan una pantalla que muestre los números decimales creados una vez que el proceso de Asociación haya sido exitoso. Además, el usuario deberá verificar la igualdad de dichos números entre las entidades que se asocian.

a. Display Authenticated Association Protocol (DAA)

En este Proyecto Técnico, se tratará específicamente este protocolo de Asociación. Se detallará el proceso, las vulnerabilidades de éste y se presentarán soluciones para mitigar las vulnerabilidades descritas. El Nodo y Hub deben tener un *display* para números de cinco dígitos decimales, como requisito para ejecutar este protocolo con el fin de obtener la MK, con la cual se creará la PTK (*Pairwise Temporal Key*). Es importante aclarar que

es el Nodo quien inicia el proceso de Asociación con el envío de la primera trama y no el Hub, como también sucede en los demás protocolos de Asociación.

a.1 Proceso del Protocolo DAA

El proceso de Asociación es la primera etapa del paradigma de seguridad establecido en el estándar IEEE 802.15.6; se puede decir que el protocolo de Asociación se divide en tres etapas:

- Fase de creación de elementos de asociación y proceso Diffie-Hellman.
- Fase de creación de elementos de autenticación con los elementos del proceso Diffie-Hellman.
- Fase de autenticación de entidades.

Para la primera fase de creación se tiene el cálculo de algunos elementos y la ejecución de algunos procedimientos como los enumerados a continuación y reflejados en la Figura 1.18:

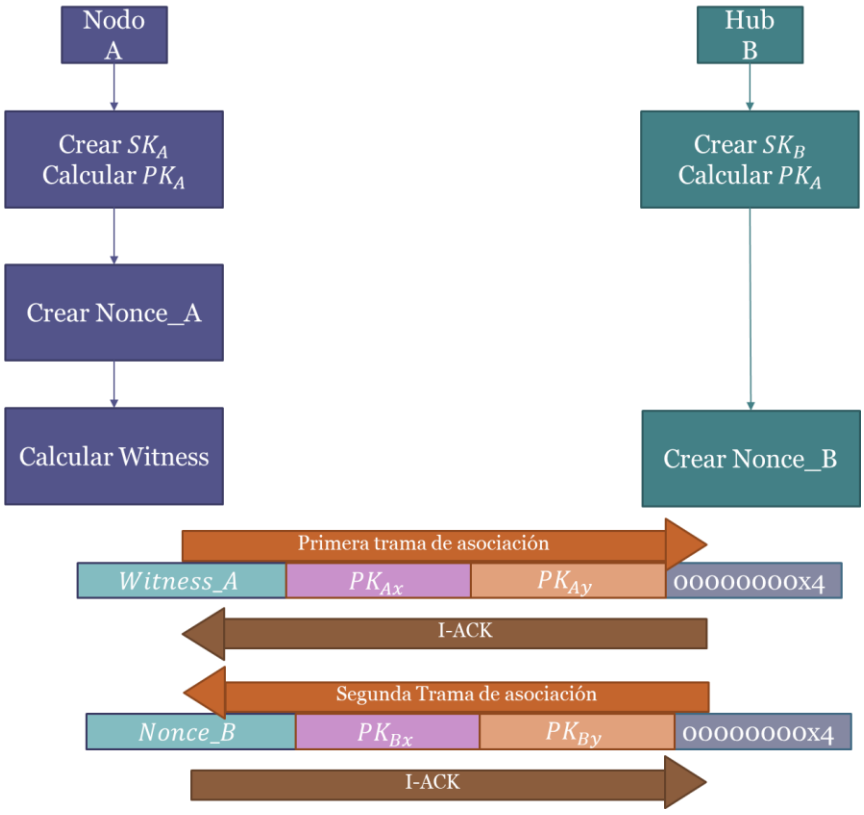


Figura 1.18. Fase de creación elementos de asociación y proceso Diffie-Hellman.

- **Creación llave secreta SK:** Obtención de un número randómico seguro de 256 bits.
- **Cálculo llave pública PK:** Obtención de dos puntos pertenecientes a una curva criptográfica elíptica en base a la Ecuación 1.16.
- **Creación *Nonce*:** Obtención de un número randómico seguro de 128 bits.
- **Cálculo *Witness*:** Obtención de un elemento seguro parte de la autenticación creado por el nodo, basado en codificación CMAC y en la Ecuación 1.18

$$Witness = CMAC(NonceNodo, DireccionNodo \,||DireccioHub\,||\,PKx\;||\;PKy, 128)$$

Ecuación 1.18. Cálculo *Witness.*

- **Intercambio primera y segunda trama de Asociación:** El Nodo envía la primera trama de Asociación hacia el Hub. El Hub envía el I-ACK de la primera trama, y posteriormente envía la segunda trama de Asociación.

Para la segunda fase de creación de elementos de autenticación, se tiene el cálculo de algunos elementos y la ejecución de algunos procedimientos como los enumerados a continuación y reflejados en la Figura 1.19:

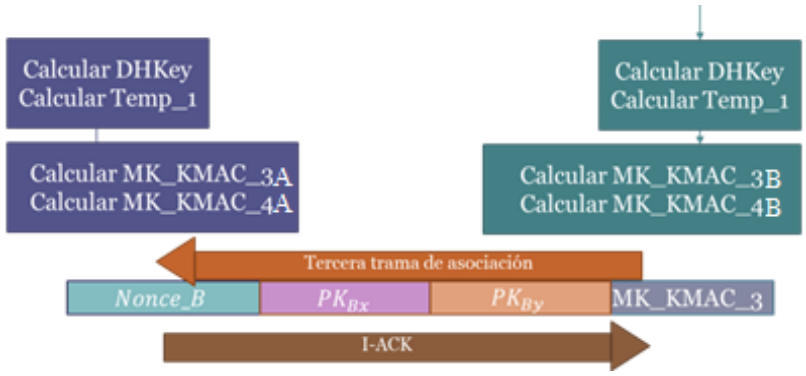


Figura 1.19. Fase de creación de elementos de autenticación con los elementos del proceso Diffie-Hellman.

- **Llave Diffie-Hellman:** Llave necesaria para el cálculo de elementos de autenticación. Es necesario extraer la coordenada en X del proceso. Se basa en la Ecuación 1.19.

$$DHKey = X(SKa \times PKb) = X\,(SKb \times PKa)$$

Ecuación 1.19. Obtención llave Diffie-Hellman.

- **Temp:** Los 128 bits más significativos de la llave Diffie-Hellman.
- **Creación MK_KMAC_3A:** Obtención de elemento seguro parte de la autenticación creado por el Nodo. Basado en codificación CMAC y en la Ecuación 1.20:

$$MK\_KMAC\_3A = CMAC(Temp, DireccionNodo \,||DireccionHub \,||\, Witness \,||\, \; NonceHub \,||CampoSS, 64)$$

Ecuación 1.20. Cálculo MK_KMAC_3A.

- **Creación MK_KMAC_3B**: Obtención de elemento seguro parte de la autenticación creado por el Hub. Basado en codificación CMAC y en la Ecuación 1.21:

$$MK\_KMAC\_3B = CMAC(Temp, DireccionNodo \,||DireccionHub \,||\, Witness \,||\, \; NonceHub \,||CampoSS, 64)$$

Ecuación 1.21. Cálculo MK_KMAC_3B.

- **Creación MK_KMAC_4A:** Obtención de elemento seguro parte de la autenticación creado por el Nodo. Basado en codificación CMAC y en la Ecuación 1.22:

$$MK\_KMAC\_4A = CMAC(Temp, DireccionHub \,||DireccionNodo||\, NonceHub||\; Witness \,||CampoSS, 64)$$

Ecuación 1.22. Cálculo MK_KMAC_4A.

- **Creación MK_KMAC_4B:** Obtención de elemento seguro parte de la autenticación creado por el Hub. Basado en codificación CMAC y en la Ecuación 1.23:

$$MK\_KMAC\_4B = CMAC(Temp, DireccionHub \,||DireccionNodo||\, NonceHub||\; Witness \,||CampoSS, 64)$$

Ecuación 1.23. Cálculo MK_KMAC_4B.

- **Intercambio tercera trama de Asociación:** Envío de la tercera trama de Asociación desde el Hub hacia el Nodo.

Para la tercera fase de autenticación se tiene el cálculo de algunos elementos y la ejecución de algunos procedimientos como los enumerados a continuación y reflejados en la Figura 1.20:

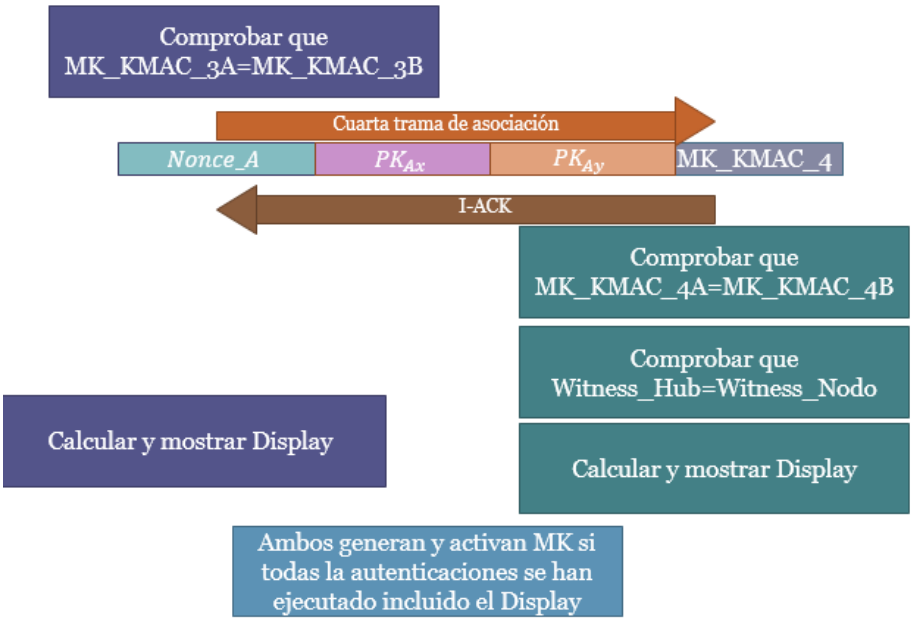


Figura 1.20. Fase de autenticación de entidades.

- **Autenticación MK_KMAC_3 en Nodo:** Si MK_KMAC_3A y MK_KMAC_3B son iguales.
- **Autenticación MK_KMAC_4 en Hub:** Si MK_KMAC_4A y MK_KMAC_4B son iguales.
- **Autenticación *Witness* en Hub:** Si el *Witness* calculado en Nodo coincide con el calculado por el Hub al final del proceso.
- **Creación de *Display***: Elemento que da al nombre al proceso y permite autenticar en pantalla utilizando la Ecuación 1.24 y transformándolo en número decimal.

$$D = CMAC(NonceNodo\ ||NonceHub, NonceHub\ ||\ NonceNodo||\ Temp, 16)$$

Ecuación 1.24. Cálculo elemento D

- **Autenticación con *Display*:** Si los *displays* en las entidades coinciden.
- **Creación MK:** Creación de llave para activación de procesos. Basada en la Ecuación 1.25, utilizando los bits menos significativos de la llave Diffie-Hellman llamado temporal 2.

$$MK = CMAC(Temp2\ , NonceNodo\ ||NonceHub, 128)$$

Ecuación 1.25. Cálculo *Master Key* (MK).

- **Activación MK:** Si ambas partes tiene el mismo *display* por consiguiente se considera activa a la llave maestra.

El primer objetivo de este trabajo es entender todo el proceso de Asociación e implementarlo para poder comprobar una asociación entre entidades, luego ejecutar ataques sobre este proceso evaluando sus vulnerabilidades.

A continuación, se explica cada detalle para poder formar las tramas usadas a lo largo del proceso de Asociación DAA.

a.1.1 Formato general de una trama MAC

Una trama MAC es una secuencia ordenada de campos entregados a o desde el punto de acceso de servicio de la capa física; permitirá intercambiar información entre las diferentes entidades a nivel de Subcapa MAC [1].

La Figura 1.21 muestra el formato general de una trama MAC la cual consiste de:

- MAC *Header* o Cabecera de 7 octetos o 56 bits
- MAC *Frame Body* que cuenta con una longitud variable máxima de 255 octetos; este campo será el que le da el funcionamiento a cada trama.
- *Frame Check Sequence* de 2 octetos

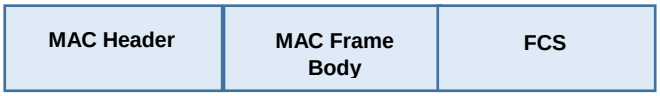


Figura 1.21. Formato general de trama en IEEE802.15.6.

El campo MAC *Header* se estructura como se muestra en la Figura 1.22 y está conformado por los sub campos:

- Control de trama de cuatro octetos o 32 bits
- Identificador de Receptor a nivel de MAC *Header* de un octeto
- Identificador de Remitente a nivel de MAC *Header* de un octeto
- Identificador de BAN de un octeto

Figura 1.22. Formato MAC *Header.*

El campo de control de trama permite identificar qué tipo de información se va a transmitir dentro de la trama MAC. Además del tipo de información a ser enviada, se puede conocer el comportamiento que tendrán los nodos de manera general en el intercambio de esa información. El campo de control está compuesto por:

- Versión del protocolo (1 bit)
- Política de ACK (2 bits)
- Nivel de Seguridad (2 bits)
- Índice TK[7] (1 bit)
- Seguridad de BAN/ Retransmisión (1 bit)
- Tiempo ACK/ Indicador de EAP/ Primera Trama en Tiempo (1 bit)
- Sub tipo de Trama (4 bits)
- Tipo de trama (2 bits)
- Más datos (1 bit)
- Última Trama/ Modo de Acceso / B2 (1 bit)
- Número de Secuencia/ Ventana de *Poll-Post* (8 bits)
- Número de Fragmento/ Siguiente / Coexistencia (3 bits)
- Fragmento no Final / Cancelación/ Escala / Inactivo (1 bit)
- Espacio Reservado (4 bits)

Si se suman todos los bits de cada uno de los sub campos del campo de control de trama se obtienen 32 bits. Como se puede apreciar existen subcampos que contienen más de una función. Las combinaciones de los valores de los subcampos darán el tipo de trama que se enviará por parte de las entidades. La combinación y la utilización de lo subcampos para las tramas de administración de Asociación, administración *Beacon* e IACK serán explicados en secciones posteriores.

a.1.2 MAC *Frame Body* de tramas de Asociación

El campo *MAC Frame Body* de las tramas de Asociación contiene principalmente el campo *Frame Payload*. El formato de cada *Frame payload* se diferenciará por el tipo de

[7] **TK**: Clave temporal.

trama de administración utilizada. Este campo contiene la información y el comportamiento de cada una de las tramas de manera mucho más específica de lo encontrado dentro del campo MAC *Header*. Este campo permite almacenar la información de las credenciales utilizadas en los procesos de los servicios de seguridad de una trama de Asociación [1].

Para una trama de Asociación la estructura se muestra en la Figura 1.23, en la cual se tienen los siguientes campos:

- Dirección de Receptor (6 octetos)
- Dirección del remitente (6 octetos)
- *Security Suite Selector* (2 octetos)
- Control de Asociación (2 octetos)
- Datos de Asociación de seguridad (88 octetos)

Figura 1.23. Formato MAC *Frame Body* de las tramas de Asociación.

La dirección de Receptor corresponde a un valor seteado a partir de un código EUI-48 y consiste de una concatenación de 24 bits asignados para un OUI (Identificador Único de Organización) y otros 24 bits que asigna esa organización conocidos como NIC (*Network Interface Card*). Esta dirección en forma es muy similar a la obtenida en una dirección MAC-48. La principal diferencia es la organización a la cual se le asigna esa dirección, aunque los términos MAC y MAC *address* se mantienen por históricos [1] y [34]. Si la dirección del receptor es desconocida este valor se seteará en ceros [1] y [34] .

La dirección de Remitente corresponde a la dirección del remitente seteado a partir de un código EUI-48.

El campo *Security Suite Selector* básicamente proporciona la información del tipo de protocolo de Asociación que se empleará y los datos pertinentes para cada protocolo. Este campo está estructurado tal como se muestra en la Figura 1.24 y cuenta con los campos de:

- Protocolo de Asociación de seguridad (3 bits)
- Nivel de Seguridad (2 bits)
- Autenticación de Trama de Control (1 bit)

- Función de cifrado (4 bits)
- Reservado (6 bits)

Figura 1.24. Campo *Security Suite Selector*

El campo protocolo de Asociación de Seguridad puede tomar distintos valores dependiendo del tipo de protocolo de Asociación a emplearse entre las entidades. Estos valores están acordes a la Tabla 1.4.

Tabla 1.4. Valores campo Protocolo de Asociación de seguridad.

Valor decimal	Protocolo de Asociación de Seguridad
0	*Master Key pre-shared Association*
1	*Unauthenticated Association*
2	*Public key hidden Association*
3	*Password Authenticated Association*
4	*Display Authenticated Association*
5-7	Reservado

El nivel de seguridad requerido se escogerá acorde a la Tabla 1.5.

Tabla 1.5. Valores campo Nivel de Seguridad.

Valor decimal	Nivel de Seguridad
0	Nivel 0 – Comunicación insegura.
1	Nivel 1- Autenticación, pero no encripción.
2	Nivel 2- Autenticación y encripción.
3	Reservado.

El campo de Autenticación de tramas de Control especifica que se establecerá en un valor de uno para los casos en que las tramas de control necesiten autenticación o encripción para los niveles uno y dos de seguridad. Se colocará un valor de cero para cualquier otro caso.

El tipo de cifrado que utilizará la trama de Asociación estará especificado dentro de la función de cifrado basándose en la Tabla 1.6.

Tabla 1.6. Valores campo Función de cifrado.

Valor decimal	Función de Cifrado
0	AES-128
1	Camellia[8]-128
2-15	Reservado

El campo de control de Asociación especificado en la Figura 1.23 contiene los campos para controlar el proceso de Asociación. Estos campos incluyen y se los muestra en la Figura 1.25, siendo los siguientes:

- Número de secuencia de Asociación (4 bits)
- Estado de Asociación (4 bits)
- Reservado (8 bits)

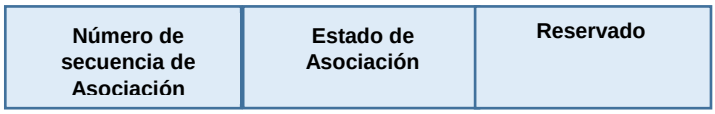


Figura 1.25. Formato campo control de Asociación.

El número de secuencia indica el número de la trama que está enviando en este momento la entidad. Dentro del proceso de Asociación DAA existirán los valores de 1 al 4.

El estado de Asociación indica el estado en el cual se encuentra determinada entidad al momento de enviar la respectiva trama de Asociación. Todos los estados posibles que podría reportar una entidad se indican en la Tabla 1.7 [1].

El último campo restante de la Figura 1.23 es el campo de Datos de Asociación de Seguridad. Este campo es principalmente importante por el hecho de contar con los valores calculados para que el proceso de Asociación DAA se complete. Los valores dependerán de cada trama enviada, pero en términos generales los campos están formados como en la Figura 1.26 y son:

[8] **Camellia:** Cifrado en bloque de llave simétrica, de tipo Feistel con rondas de 18 o 24. Posee tamaño de bloque de 128 bits y posibles tamaños de llave de 128, 192 y 256 bits

Tabla 1.7. Valores campo Estado de Asociación.

Valor decimal	Estado de asociación
0	Unirse al procedimiento de Asociación de seguridad.
1	Cancelación del procedimiento de Asociación de seguridad con un *Security Suite Selector* de seguridad diferente.
2	Anulando el procedimiento de Asociación de seguridad debido a la falta de credencial de seguridad necesaria.
3	Abortar el procedimiento de Asociación de seguridad debido a la falta temporal de recursos.
4	Anulando el procedimiento de Asociación de seguridad debido a restricciones de política de seguridad impuestas por el administrador / propietario de este Hub en las comunicaciones en su BAN.
5-15	Reservado.

- *Nonce* de remitente (16 octetos)
- PKx de remitente (32 octetos)
- PKy de remitente (32 octetos)
- MK_KMAC (8 octetos)

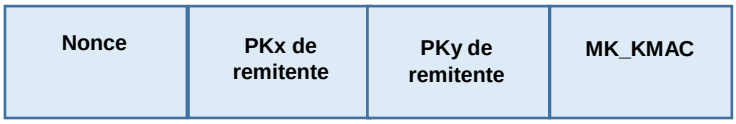


Figura 1.26. Formato campo Datos de Asociación de Seguridad.

Los valores que tomarán estos campos depende de la trama que está enviando en ese momento la entidad. Los valores se detallan dentro del proceso de Asociación DAA como tal. Por ejemplo, dentro del campo *Nonce* para un Nodo se puede enviar el elemento *Witness* y un *Nonce* dentro de la primera y cuarta trama de Asociación respectivamente.

Un Nodo dentro de la primera trama de Asociación enviará el *Witness,* las llaves PKx, PKy y una secuencia de 16 caracteres de cero. Para la cuarta trama de Asociación los valores a enviar serán el *Nonce*, las llaves PKx, PKy y el valor de MK_KMAC_4A.

Un Hub dentro de la segunda trama de Asociación enviará el *Nonce*, las llaves PKx, PKy y una secuencia de 16 caracteres de cero. Para la tercera trama de Asociación los valores a enviar serán el *Nonce*, las llaves PKx, PKy y el valor de MK_KMAC_3B.

a.1.3 MAC *Frame Body* de tramas *Beacon*

Una trama *Beacon* entra también en la categoría de trama de Administración, con un formato similar de cabecera al de la trama de Asociación.

Para este caso en el MAC *Frame Body* se encuentran los siguientes campos:

- Dirección remitente (6 octetos)
- *Beacon period length* (1 octeto)
- *Allocation slot numbering* (1 octeto)
- RAP1 *end* (1 octeto)
- RAP2 *start* (1 octeto)
- RAP2 *end* (1 octeto)
- MAC *Capability* (3 octetos)
- PHY *Capability* (1 octeto)
- RAP1 *start* (1 octeto)

La dirección de remitente corresponde a la dirección establecida a partir de un código EUI-48. Esta dirección permite que el nodo el cual empezará el proceso de Asociación conozca la dirección de la entidad a la cual va enviar la primera trama de Asociación. Si el nodo no recibiera una super trama *Beacon* no contará con la información necesaria para iniciar el proceso de Asociación.

Para el valor de *Beacon period length s*e debe considerar que para este trabajo se usará escenario sin "*shifting*" activado; entonces este valor define la longitud del actual período de *Beacon*, el cual está enviando en ese momento el Hub, en términos de *slots* de asignación. Se podrán utilizar valores entre 00_H y FF_H. Para efectos prácticos es posible utilizar el mayor valor que está codificado en el valor 00_H que representa 256 *slots* [1] , [35] y [36].

El campo *Allocation slot numbering* y para el escenario mencionado, determina que los *slots* asignados empezarán en 00_H e irán aumentando hasta FF_H. Este valor aumentará únicamente cuando se asigne un nuevo *slot*.

RAP1 (*Random Access Phase*) *end* toma el valor de asignación $E1_H$, indicando que, si existe acceso randómico en el proceso, éste terminará en el valor $E1_H$. $E1_H$ podrá tomar valores a lo largo de 00_H y FF_H. Se sugiere un valor intermedio debido a que este valor se colocará más bajo que cualquier valor antes establecido en $E1_H$, pero muy alto evitará que llegue de manera pronta a él [1] , [35] y [36].

RAP2 *end* se utiliza cuando más de un acceso randómico está en acción. Como objetivo de este trabajo, el escenario utilizado no emplea este campo. El valor RAP1 es el único utilizado para definir accesos randómicos; si este campo no se utiliza se establecerá en un valor con una cadena de ceros [1] , [35] y [36].

RAP2 *start* se utiliza cuando más de un acceso randómico está en acción. Como objetivo de este trabajo, el escenario utilizado no cuenta con esto, de la misma forma que en RAP2 *end*. Por ello este valor también contará con una cadena de ceros en su campo.

Dentro del campo MAC *Capability* se establecen campos que permiten controlar la capacidad a nivel de MAC que tendrá la entidad en modo *Beacon*. Los valores con los que cuenta este campo son:

- CSMA/CA
- *Slotted Aloha*
- *Type I Poll*
- *Type II Poll*
- *Scheduled Access*
- *Unscheduled Access*
- *Fragmentation*
- *Command Frames*
- *Node always active*
- *Guard Time provisioning*
- L-BACK
- G-ACK
- *Relaying Node*
- *Relayed Hub*
- *Beacon Shifting*
- *Channel Hopping*
- *Data Subtypes*
- *Reserved*

Si se activan todos los campos en 1, permitirá a la entidad trabajar bajo cualquier escenario posible en capa MAC. Que todos los valores estén colocados en uno no implica que serán utilizados o que deben ser implementados todos. Los valores en uno indican únicamente que si es necesario y si las condiciones lo requieren serán utilizados, facilitando el control y uso de las tramas *Beacon* para los propósitos de este trabajo [1] , [35] y [36].

Dentro del campo PHY *Capability* se establecen campos que permiten controlar la capacidad a nivel físico que tendrá la entidad en modo *Beacon*. Los valores con los que cuenta este campo son:

- Rate 0
- Rate 1
- Rate 2
- Rate 3
- Rate 4
- Rate 5
- Rate 6
- Rate 7

Los valores correspondientes a las distintas velocidades con respecto al tipo de modulación dentro de la red, se encuentran especificadas en la Tabla 15 del Anexo I.

De la misma forma que en la capa MAC, en la capa física se podrán colocar todos los campos en 1, dándole la capacidad al dispositivo de que trabaje en cualquiera velocidad y cualquier banda, dando cerca de 60 posibles combinaciones. Nuevamente esto no significa que se estén implementando todas, significa que puede soportar la velocidad y banda deseada, la entidad que esté enviando las tramas en ese momento.

a.1.4 Estructura Trama *Beacon*

Como se había mencionado en literales anteriores el campo MAC *Header* indicado en la Figura 1.21, varía y depende mucho de la trama que se va a enviar. Para el caso de una super trama *Beacon* los valores utilizados en el campo MAC *Header* corresponden a los indicados en la Tabla 1.8.

La continuación de la trama, es decir, el MAC *Frame Body* de una trama *Beacon* será con los valores que se muestran en la Tabla 1.9, en base a los campos mencionados en el literal a.1.3 de esta sección.

Dentro de MAC *Capability* de una trama *Beacon* se colocarán los valores que se muestran en la Tabla 1.10, en base a los campos mencionados en el literal a.1.3 de esta sección.

Tabla 1.8. Valores campo MAC *Header* trama *Beacon*.

Campo	Valor (bits)	Razón
Versión del protocolo	0	El valor de la versión del protocolo será cero para la revisión del protocolo manejado.
Política de ACK	00	Correspondiente a una política de No respuesta N-ACK; para tramas *Beacon* no es necesaria.
Nivel de Seguridad	00	Correspondiente a tramas no seguras. Ningún tipo de encripción para tramas *Beacon.*
Índice TK	0	Se coloca en cero debido a que solo se implementa en tramas seguras con llaves PTK (Llave temporal privada) / GTK (llave temporal grupal).
Seguridad de BAN	0	Ya que la entidad aceptará comunicaciones inseguras.
Indicador de EAP	1	Valor establecido en uno como indicador de EAP (*Exclusive Access Phase*) para tramas *Beacon* ya sea en modo *shifting o no shifting.*
Subtipo de Trama	0000	Valor determinado para tramas *Beacon.*
Tipo de trama	00	Valor determinado para tramas de Administración.
Más datos	1	Se establece en uno si el Nodo tiene al menos una trama de Administración o de tipo de datos pendiente para su transmisión o retransmisión al Hub.
B2	0	Este campo no se utilizará debido a que no se necesitan tramas del tipo B2 para nuestros fines, por ello se establece en un valor de cero.
Número de Secuencia	*secuencia	Para las tramas *Beacon* se utilizará como un valor de secuencia dependiendo del número de trama enviada.
Coexistencia	000	Primer cero ya que no está habilitado el modo *shifting.* Segundo cero ya que no está habilitado el modo *channel hopping.* Tercer cero ya que la entidad no soporta *interleaving.*
Inactivo	0	Cero ya que no existe ninguna super trama adicional en el escenario BAN.
Espacio Reservado	0000	Espacio reservado para ajustar el tamaño de cabecera.
ID Receptor	00_H	Valor para *broadcast* para nodos no conectados.
ID Remitente	01_H	Valor para *unicast* desde nodos no conectados.
ID BAN	ID	Valores entre 0x02–0xF5 para entidades conectadas para envío *unicast.*

Tabla 1.9. Valores MAC *Frame Body* trama *Beacon*

Campo	Valor (Hex)	Razón
Dirección del Remitente	MAC	Se utiliza la dirección MAC del dispositivo por su similitud con la dirección EUI-48.
Beacon period length	00_H	La cadena de ceros indica que se utilizará la máxima cantidad de *slots* posibles de asignar.
Allocation Slot Numbering	00_H	Se utilizará el primer período de asignación para el Nodo dentro del escenario.
RAP1 *end*	81_H	Se tomó este valor debido a que se recomienda. En general este valor no se colocará más bajo que cualquier valor antes establecen en E1.
RAP2 *start*	00_H	Se ocupa cuando más de un acceso randómico está en acción; para el escenario se utilizan los valores RAP1 para definir los accesos randómicos, si estos campos no se utilizan se establecen valores en 0.
RAP2 *end*	00_H	Se ocupa cuando más de un acceso randómico está en acción. Para el escenario se utilizarán los valores RAP1 para definir los accesos randómicos, si estos campos no se utilizan se establecen valores en 0.
RAP1 *start*	80_H	Para que el inicio y el final del *slot* de acceso randómico sean a un *slot* de diferencia.
MAC *Capability*		Referirse a la Tabla 1.10.
PHY *Capability*		Referirse a la Tabla 1.11.

Dentro de PHY *Capability* de una trama *Beacon* se colocarán los valores que se muestran en la Tabla 1.11, en base a los campos mencionados en el literal a.1.3 de esta sección. Los valores de la tasa o velocidad de transmisión dependerán de la banda de frecuencia de operación, estos posibles valores se especifican en la Tabla 15 del Anexo I.

Tabla 1.10. Valores campo MAC *Capability*.

Campo	Valor (bits)	Razón
CSMA/CA	1	La entidad podrá aceptar contención CSMA/CA.
Slotted Aloha	1	La entidad podrá aceptar contención Aloha.
Type-1 Poll	1	La entidad podrá aceptar poleo de tipo 1.
Type-2 Poll	1	La entidad podrá aceptar poleo de tipo 2.
Scheduled Access	1	La entidad podrá aceptar acceso programado.
Unscheduled Access	1	La entidad podrá aceptar acceso no programado.
Fragment	1	La entidad podrá aceptar fragmentación.
Command Frames	1	La entidad podrá aceptar Tramas de Comando.
Node Always active	1	La entidad estará siempre en modo activo.
Guard Time provisioning	1	La entidad podrá aceptar aprovisionamiento de tiempo.
L-ACK	1	La entidad podrá admitir L-ACK(ACK retrasado).
G-ACK	1	La entidad podrá admitir G-ACK (ACK grupal).
Relaying Node	1	La entidad podrá soportar la funcionalidad de *relaying node* (nodo retransmisor intermedio) para realizar comunicación en una configuración tipo estrella.
Relayed Hub/Node	1	La entidad podrá soportar la funcionalidad de *relayed node* (nodo que se comunica a través de un nodo intermedio) para realizar comunicación en una configuración tipo estrella.
Beacon Shifting	0	La entidad no admitirá modo *shifting* en *Beacon.*
Channel Hopping	0	La entidad no admitirá modo *channel hopping* en *Beacon.*
Data Subtypes	0000	La entidad admitirá los 16 subtipos de trama, que es el máximo para este campo.
Reserved	0000	Reservado para ajustar longitud de campo en trama.

a.1.5 Estructura Trama Asociación

Como se había mencionado en literales anteriores, el campo MAC *Header* indicado en la Figura 1.21 varía y depende mucho de la trama que se va a enviar. Para el caso de una

trama de Asociación los valores utilizados en el campo MAC *Header* corresponden a los indicados en la Tabla 1.12.

El *Payload* de las tramas de Administración para Asociación de Seguridad está descrita en el literal a.1.2 y los valores se describen en la Tabla 1.13.

Tabla 1.11. Valores campo PHY *Capability*.

Campo	Valor (bits)	Razón
Rate 0	1	La entidad admitirá velocidad 0
Rate 1	1	La entidad admitirá velocidad 1
Rate 2	1	La entidad admitirá velocidad 2
Rate 3	1	La entidad admitirá velocidad 3
Rate 4	1	La entidad admitirá velocidad 4
Rate 5	1	La entidad admitirá velocidad 5
Rate 6	1	La entidad admitirá velocidad 6
Rate 7	1	La entidad admitirá velocidad 7
Rate 8	1	La entidad admitirá velocidad 8

Tabla 1.12. Valores MAC *Frame Body* trama Asociación.

Campo	Valor	Razón
Dirección de receptor	MAC receptor	Dirección MAC por su similitud a la dirección EUI-48.
Dirección de remitente	MAC remitente	Dirección MAC por su similitud a la dirección EUI-48.
Security Suite Selector		Referirse a tabla 1.14.
Control de Asociación		Referirse a tabla 1.15.
Datos de Asociación de Seguridad		Referirse a tabla 1.16.

Tabla 1.13. Valores MAC *Header* trama Asociación.

Campo	Valor (bits)	Razón
Versión del protocolo	0	El valor de la versión del protocolo será cero para la revisión del protocolo manejado.
Política de ACK	10	Correspondiente a una política de respuesta inmediata I-ACK.

Campo	Valor (bits)	Razón
Nivel de Seguridad	00	Correspondiente a tramas no seguras. Ningún tipo de encripción para tramas de Asociación.
Índice TK	0	Se coloca en cero debido a que solo se implementa en tramas seguras con llaves PTK /GTK (*Group Temporal Key*).
Seguridad de BAN	0	Reservada para tramas de administración para asociación.
Primera Trama en Tiempo	1	Se establece en uno si ésta es la primera trama enviada por el Hub al nodo al comienzo de un intervalo de asignación.
Sub tipo de Trama	0100	Valor determinado para tramas de Asociación de seguridad.
Tipo de trama	00	Valor determinado para tramas de Administración.
Más datos	1	Se establece en uno si el Nodo tiene al menos una trama de Administración o de tipo de datos pendiente para su transmisión o retransmisión al Hub.
Última trama	0	Se establece en cero si es probable que el remitente envíe otra trama en el intervalo de asignación actual después de la trama actual.
Número de Secuencia	*secuencia	Para las tramas de Asociación se utilizará como un valor de secuencia dependiendo del número de trama enviada.
Número de Fragmento	000	Se establece en cero si la trama actual contiene una carga no fragmentada o el primer fragmento de una carga de trama fragmentada.
Fragmento no Final	0	Se establece en cero si la trama actual contiene una carga no fragmentada o el primer fragmento de una carga de trama fragmentada.
Espacio Reservado	0000	Espacio reservado para ajustar el tamaño de cabecera.
ID Remitente	ID	Valores entre 0x02–0xF5 para entidades conectadas para envío *unicast.*
ID Receptor	ID	Valores entre 0x02–0xF5 para entidades conectadas para envío *unicast.*
ID BAN	ID	Valores entre 0x02–0xF5 para entidades conectadas para envío *unicast.*

Tabla 1.14. Valores del campo *Security Suite Selector*.

Campo	Valor (decimal)	Razón
Protocolo de Asociación de Seguridad	4	Correspondiente al protocolo DAA.
Nivel de seguridad	0	Comunicación insegura.
Control de autenticación de trama	0	Para tramas que utilicen el nivel uno de seguridad.
Función de cifrado	0	Cifrado con AES-128.
Reservado	0	Valor para completar campo *Security Suite Selector.*

Tabla 1.15. Valores del campo de Datos de Asociación de Seguridad

Campo	Longitud en bits	Razón
Nonce	128	Nodo • *Witness*[9] primera trama • *Nonce*[10] cuarta trama Hub • *Nonce* segunda trama • *Nonce* tercera trama
PKx	256	Nodo • PKx nodo Hub • PKx Hub
PKy	256	Nodo • PKy nodo Hub • PKy Hub
MK_KMAC	64	Nodo • Ceros primera trama • MK_KMAC_4A cuarta trama Hub • Ceros segunda trama • MK_KMAC_3B tercera trama

[9] **Witness:** Elemento testigo, será utilizado en la autenticación de entidades.
[10] **Nonce**: Número aleatorio seguro.

Tabla 1.16. Valores del campo Control de Asociación.

Campo	Valor (decimal)	Razón
Secuencia de Asociación	Secuencia	Número de trama de Asociación que se envía.
Estado de Asociación	0	Cuando en la trama se avisa que la entidad se une al procedimiento de Asociación.
Reservado	0	Valor para completar campo Control de Asociación.

a.1.6 Estructura Trama IACK

La estructura de una trama de ACK inmediato no cuenta con *payload,* únicamente dispone de MAC *Header* y sus valores se explican en la Tabla 1.17.

Tabla 1.17. Valores MAC *Header* IACK.

Campo	Valor (bits)	Razón
Versión del protocolo	0	El valor de la versión del protocolo será cero para la revisión del protocolo manejado.
Política de ACK	10	Correspondiente a una política de respuesta inmediata I-ACK.
Nivel de Seguridad	00	Correspondiente a tramas no seguras. Ningún tipo de encripción para tramas de Asociación.
Índice TK	0	Se coloca en cero debido a que solo se implementa en tramas seguras con llaves PTK /GTK.
Seguridad de BAN	0	Reservada para tramas de administración para asociación.
Tiempo de ACK	0	La trama IACK no incluye una estampilla de tiempo.
Subtipo de Trama	0000	Valor determinado para tramas IACK.
Tipo de trama	10	Valores para tramas de control.
Más datos	1	Se establece en uno si el nodo tiene al menos una trama de administración o de tipo de datos pendiente para su transmisión o retransmisión al Hub.

Campo	Valor (bits)	Razón
Última trama	0	Se establece en cero si es probable que el remitente envíe otra trama en el intervalo de asignación actual después de la trama actual.
Número de Secuencia	*secuencia	Para las tramas de asociación se utilizará como un valor de secuencia dependiendo el número de trama enviada.
Número de Fragmento	000	Reservado para tramas IACK sin poll. Este tipo de trama no se utilizará de la misma forma que este campo. Los campos reservados se colocan en cero.
Fragmento no Final	0	Reservado para tramas IACK sin poll. Este tipo de trama no se utilizará de la misma forma que este campo. Los campos reservados se colocan en cero.
Espacio Reservado	0000	Espacio reservado para ajustar el tamaño de cabecera.
ID Remitente	ID	Valores entre 0x02–0xF5 para entidades conectadas para envío *unicast.*
ID Receptor	ID	Valores entre 0x02–0xF5 para entidades conectadas para envío *unicast.*
ID BAN	ID	Valores entre 0x02–0xF5 para entidades conectadas para envío *unicast.*

a.2 Vulnerabilidades en el proceso DAA

Todos los protocolos de asociación presentados en el estándar IEEE 802.15.6 son vulnerables a ataques KCI[11] (*Key Compromise Impersonation* – KCI) por lo cual se puede perder el secreto de las llaves en el avance del proceso. Además, el DAA es vulnerable a ataques de impersonalización, debido a que se envía PK, *Nonce* y otros elementos de autenticación en texto plano. Al no tener seguridad en el envío de estos elementos, un atacante podría apropiarse de estos elementos y replicarlos para afectar la fase de autenticación, comprometiendo todo el proceso de Asociación como se demuestra teóricamente en [2] y [3]. El estándar IEEE 802.15.6 debería trabajar en asegurar la integridad y confidencialidad en el envío de los elementos de autenticación, ya que actualmente el Hub no puede determinar que los elementos que recibe provienen de la entidad con la que está intentado asociarse.

[11] **KCI:** Ataques de impersonalización que comprometen la seguridad del traspaso de información como claves.

Si bien es cierto, el proceso de cálculo del *Witness* es robusto, pero al no garantizar el secreto de los elementos que se utilizan para su cálculo hace al proceso de Asociación vulnerable. En esta investigación se demostrará de manera práctica lo expuesto teóricamente en la referencia mencionada.

a.3 Solución teórica a vulnerabilidades en el DAA

Para contrarrestar los efectos de los ataques analizados en la sección anterior, se recurrirán a conceptos utilizados en otro proceso de Asociación del estándar IEEE802.15.6.

Dentro del proceso de Asociación *Password Authenticated Association* (PAA), se utiliza una llave precompartida para implementar la autenticación. PAA modifica sus argumentos calculados con la inserción de esta llave. La modificación de los argumentos hace que en el proceso de Asociación se intercambien valores modificados como PK'. Esta llave precompartida no se intercambia a través de mensajes, más bien un dispositivo con el estándar IEEE802.15.6 deberá ya contar de fábrica con este valor en su memoria [1] y [36].

Para las vulnerabilidades de DAA se utilizará la llave precompartida para poder contrarrestar de manera sencilla lo realizado por el atacante. Como se analiza, las vulnerabilidades del proceso se dan debido a que una entidad no puede descubrir que las credenciales y los mensajes de asociación vienen procedentes de un atacante. Para esto la modificación se realizará a nivel del parámetro *Witness*. Entre las entidades Nodo y Hub existe autenticación a nivel de testigo, previo a la autenticación por *display*. Es decir que, si no se consigue una autenticación correcta por parte del Hub con el parámetro *Witness*, el proceso de Asociación será abortado. El proceso se abortará con un atacante cuando éste no calcule los valores, utilizando la llave precompartida para calcular las credenciales y valores para las tramas de asociación.

Para poder solucionar las vulnerabilidades se han considerado dos alternativas:

- Modificar el mensaje *Witness* y enviar un *Witness* modificado.
- Utilizar una llave modificada para que en el proceso CMAC que calcula el *Witness*, éste sea alterado y enviar un *Witness* modificado.

a.3.1 Modificación del mensaje *Witness*

Dentro de un proceso CMAC el mensaje partirá independientemente de la longitud de éste. El concatenar un nuevo valor no afectará el proceso CMAC. Por esto, la primera solución para un ataque es concatenar una llave precompartida al mensaje que se

codificará. Con esto el atacante si no conoce la llave precompartida no podrá realizar el proceso CMAC. La seguridad que otorga CMAC y AES permitirá confirmar que si un atacante recibe el nuevo valor de *Witness* difícilmente podrá conocer el valor de la llave precompartida entre las entidades. El cálculo del nuevo *Witness* se basa en la Ecuación 1.26.

$$Witness = CMAC(NonceNodo, DireccionNodo \,||DireccioHub\,||\, PKx \;||\; PKy\,||LlavePrecompartida, 128)$$

Ecuación 1.26. Cálculo nuevo *Witness.*

a.3.2 Modificación de la clave para codificación CMAC

La modificación de la clave afectará a todo el proceso CMAC que calculará el *Witness*. Para afectar la clave del cálculo de CMAC se realizará una encripción AES utilizando el *Nonce* A y la llave precompartida. La encripción de AES con el *Nonce* entregará una clave de la misma longitud deseable pero modificada de tal forma que el proceso será seguro contra un ataque. De la misma forma, con el nuevo *Witness* enviado, para un atacante prácticamente será imposible saber cuál fue la llave precompartida. La modificación de llave se obtiene a partir de la Ecuación 1.27.

En ambos casos el nodo enviará un *Witness* modificado y el Hub dentro de la autenticación de *Witness* deberá aplicar el mismo procedimiento para el cálculo de *Witness* e identificará si ambos son iguales procediendo con el cálculo de MK.

$$K = AES_{128}(llavePrecompartida, NonceNodo)$$

Ecuación 1.27. Cálculo clave modificada *Witness.*

2 METODOLOGÍA

En este Proyecto Técnico, se realizó investigación bibliográfica. Se tuvo que cubrir varias áreas de estudio, previo a entender de forma completa el protocolo de Asociación DAA y su posterior implementación práctica. Es así, que como se muestra en el capítulo 1 se realizó una investigación teórica desde los más general hacia lo particular, haciendo uso del método de investigación deductivo. Una vez con el conocimiento pleno del tema específico, fue necesario abarcar otras áreas de investigación, con el fin de entender de mejor manera los procesos llevados a cabo dentro de éste. Todo el conocimiento teórico investigado en las áreas de WBANs, Seguridad de la Información y junto a los conocimientos básicos acerca de Programación Orientada a Objetos, Sistemas Distribuidos, etc. fueron importantes y necesarios para alcanzar el objetivo de este proyecto.

Para la implementación práctica del tema propuesto, se dividió el proyecto en pequeñas partes que debían ser llevadas a cabo, con el fin de que al integrarse con sus precedentes puedan conformar el trabajo final. En la parte de desarrollo, se utilizó el método iterativo incremental; cada vez que se cumplió una iteración, se realizaron las pruebas necesarias para dar de alta a la misma. Posteriormente, se integraba la nueva iteración con sus precedentes, donde se verificaba la funcionalidad de la misma y de ser el caso se mejoraban sus características. La aplicación de este método, facilitó el llevar a cabo tan complejo proyecto; pues, en inicios se enfocó en lograr las características mínimas, las cuales iban siendo mejoradas según el avance y requerimiento de iteraciones futuras, para finalmente lograr el entregable final de calidad.

Para la implementación práctica de este proyecto técnico, se dividió en etapas esenciales de desarrollo, las cuales se identificaron como indispensables para alcanzar el objetivo. Estas etapas de desarrollo son:

- **Modelo Cliente – Servidor:** Programación en red para lograr la comunicación entre dispositivos que se utilizarán para emular los Nodos y Hub; en este caso se emplearán PCs.
- **Protocolo DAA:** La implementación del protocolo DAA por medio de la codificación del mismo.
- **Ataques al DAA y soluciones:** Codificación de ataques al proceso de Asociación DAA y prueba de soluciones ante las vulnerabilidades encontradas.

- **GUI:** Integración de todo e implementación de la Interfaz Gráfica de Usuario (GUI), con el fin de lograr una mejor visualización del proceso en una interfaz gráfica, en lugar de la consola.

Las subsecciones de diseño e implementación que se detallarán a continuación, estarán centradas en estas etapas y sus partes que las comprenden.

2.1 Fase de diseño

2.1.1 REQUERIMIENTOS FUNCIONALES

a. Emulación de intercambio de mensajes en WBAN

Será indispensable que se tenga una conexión establecida entre los actores previo a iniciar el proceso de Asociación DAA. Con este propósito, se crea un modelo Cliente - Servidor mediante el uso de programación en red. El uso de *sockets* TCP, logra que se pueda establecer comunicación hacia un servidor con determinada dirección IP y número de puerto TCP.

Es importante que se puedan obtener las direcciones MAC de los *hosts* que forman parte del modelo Cliente – Servidor, ya que estos datos serán útiles para emular las tramas *Beacon* de una comunicación WBAN.

La interacción del sistema deberá ser controlada a tal punto que su resultado final sea una viva imagen de lo que realmente sucede en comunicaciones WBAN, donde a partir de las consideraciones planteadas y exceptuando las limitaciones que se encuentran fuera del alcance de este proyecto, se logre cumplir con el objetivo planteado.

b. Implementación del protocol DAA

El sistema debe tener la capacidad de emular el proceso de Asociación DAA entre dos entidades, Nodo y Hub, tal como lo hacen dentro de una WBAN.

c. Implementación de ataques al protocol DAA

Una vez que se logre implementar el DAA en su totalidad, será necesario recrear los escenarios de ataques a las vulnerabilidades existentes en este proceso, Impersonalización de Nodo y ataque de Hombre en el medio.

d. Implementación de soluciones a vulnerabilidades

De comprobar la existencia de dichas vulnerabilidades, mediante la emulación de las mismas, se procederá a implementar soluciones dentro del proceso DAA, las cuales fortalezcan el proceso de Asociación y eviten ataques como los mencionados.

e. Mensajes de error

Es importante que, durante todo el proceso, desde el establecimiento de la comunicación hasta la ejecución del proceso de Asociación DAA, se muestren los errores que pueden suscitarse. Por esta razón, todo el proceso ha sido limitado a ejecutar el resultado esperado, y caso contrario, mostrar los errores con sus respectivos mensajes. De esta forma el usuario conocerá las fallas que se han desencadenado durante la ejecución, siendo informado de la causa de éstas.

f. Interfaz Gráfica de Usuario

Como se mencionó anteriormente, se implementarán interfaces gráficas GUI para que el usuario pueda visualizar de una manera más organizada y amigable el proceso en general. De esta forma se evita que el usuario deba recurrir a la consola para visualizar el proceso, donde precisamente no se encuentra el proceso de manera clara y visible como en la GUI. Para cada una de las entidades que forman parte del proceso, Nodo y Hub, se han desarrollado GUIs que muestren su proceso propio y discriminen todos los mensajes replicados por el servidor que son ajenos a su función. Adicionalmente, el Atacante posee una GUI más compleja, donde se pueden observar por separado los procesos, cuando éste toma las funciones de Nodo o de Hub, e incluso muestra el ataque que ha logrado ejecutar.

2.1.2 REQUERIMIENTOS NO FUNCIONALES

a. Número de usuarios concurrentes y desempeño

Para emular una WBAN, será necesario que el servidor pueda tener 64 conexiones concurrentes, donde en el caso ideal, uno de los clientes deberá tener funcionalidades de Hub y los demás de Nodo. Como parte del proceso, será indispensable que se discrimine los mensajes replicados por el servidor, ya que una entidad no deberá mostrar sus propios mensajes transmitidos. Por ejemplo, la difusión de tramas *Beacon*, llevada a cabo por el Hub, solamente deberá ser visualizada por los nodos que forman parte de la red. Estos aspectos garantizarán que el modelo Cliente – Servidor, logre emular correctamente el establecimiento de comunicación de una WBAN.

b. Hardware y software

Los requerimientos de *software* y *hardware*, para el desarrollo y pruebas de aplicación serán los detallados a continuación. En cuanto a hardware, será necesario la utilización de mínimo tres *hosts* en la red del modelo Cliente – Servidor, los cuales para estar conectados entre sí harán uso de un *Switch* de capa 2; también se requerirán 3 *Patch Cords* UTP. Los requerimientos de los *hosts* en los que se realizará las pruebas son: procesador Intel Core i5 de 2.4 GHz, memoria RAM de 4 GB instaladas con 2.43 GB utilizables, y 500 GB de almacenamiento. Sin embargo, como es lógico suponer, la aplicación no estará limitada a ser ejecutada en equipos de alta capacidad de procesamiento.

En cuanto a software, será necesario la utilización de un Entorno Integrado de Desarrollo (*Integrated Development Environment* – IDE), que sea capaz de trabajar con programación orientada a objetos. Para este trabajo de titulación, se escogió Eclipse Versión Neon.3 *Release* (4.6.3) ó NetBeans IDE 8.2, los cuales utilizan *Java SE Development Kit* 1.8 (JDK). En este entorno se instaló Eclipse *WindowBuilder*, el mismo que facilita la implementación de una GUI de manera gráfica, autogenerando el código propio de la GUI. Sin embargo, será necesario programar sobre este código todas las características funcionales para lograr la GUI deseada. Estos requerimientos de software harán posible el desarrollo y pruebas de ejecución del aplicativo propuesto.

2.1.3 DIAGRAMAS

Como se indicó en la parte introductoria a este capítulo, existen cuatro etapas esenciales en el desarrollo. En base a éstas se definirán los respectivos diagramas de secuencia, de clases y de actividades, según sea el caso, de forma global, ya que en el subcapítulo siguiente se abordará a detalle cada proceso de las mencionadas etapas.

a. Modelo Cliente – Servidor

a.1 Diagrama de secuencia

En la Figura 2.1 se puede observar el diagrama de secuencia UML del modelo Cliente-Servidor implementado para establecer la comunicación entre entidades.

a.2 Diagrama de clases

Para la implementación del modelo mencionado, se utilizó programación orientada a objetos, donde se codificó una serie de clases y métodos detallados en la Figura 2.2. La

interacción de cada una de estas clases hace posible el establecimiento de conexión entre los *hosts* conectados en red, quienes emulan el funcionamiento de Nodo y Hub de las redes WBAN.

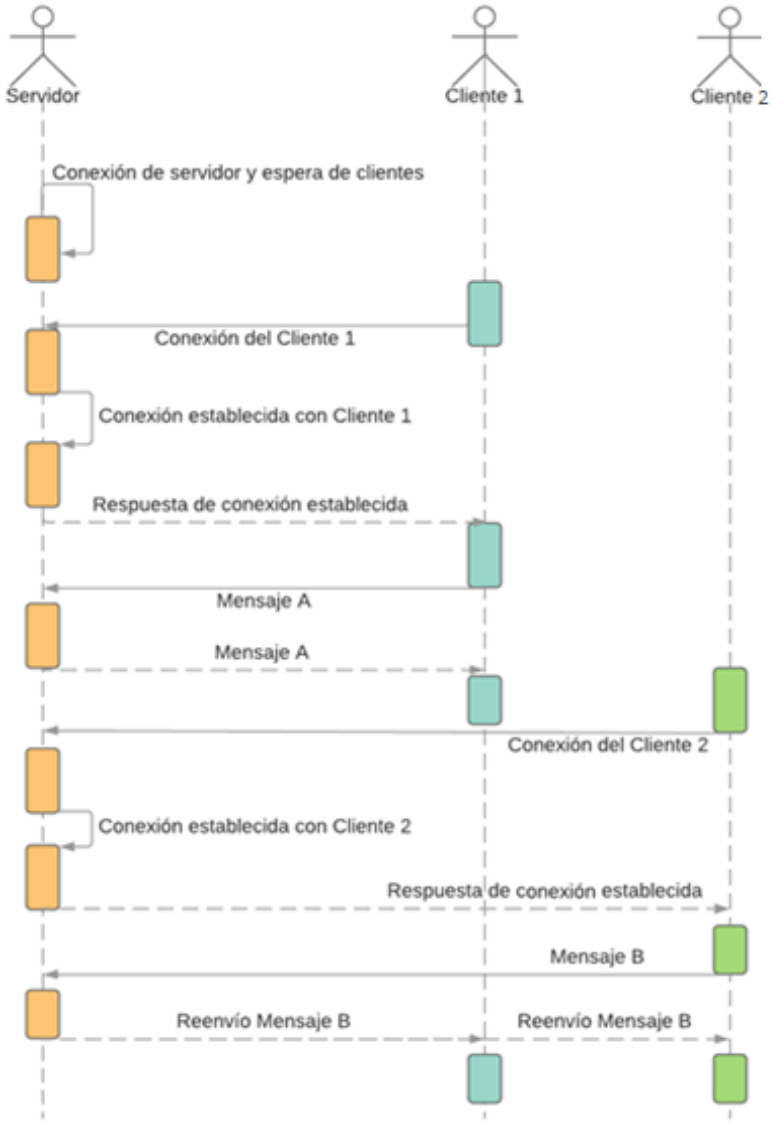


Figura 2.1. Diagrama de secuencia UML del modelo Cliente-Servidor.

a.3 Diagrama de actividades

En la Figura 2.3 se muestra el diagrama de actividades, tanto para el Cliente como para el Servidor del modelo implementado.

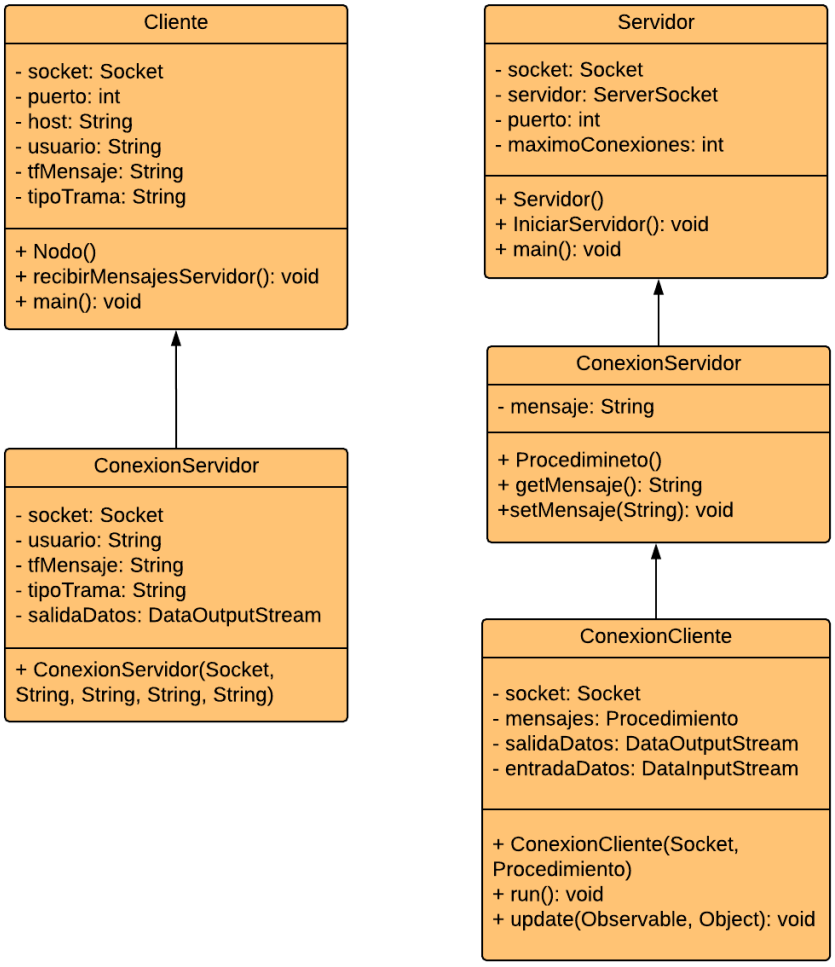


Figura 2.2. Diagramas de clases UML del modelo Cliente-Servidor.

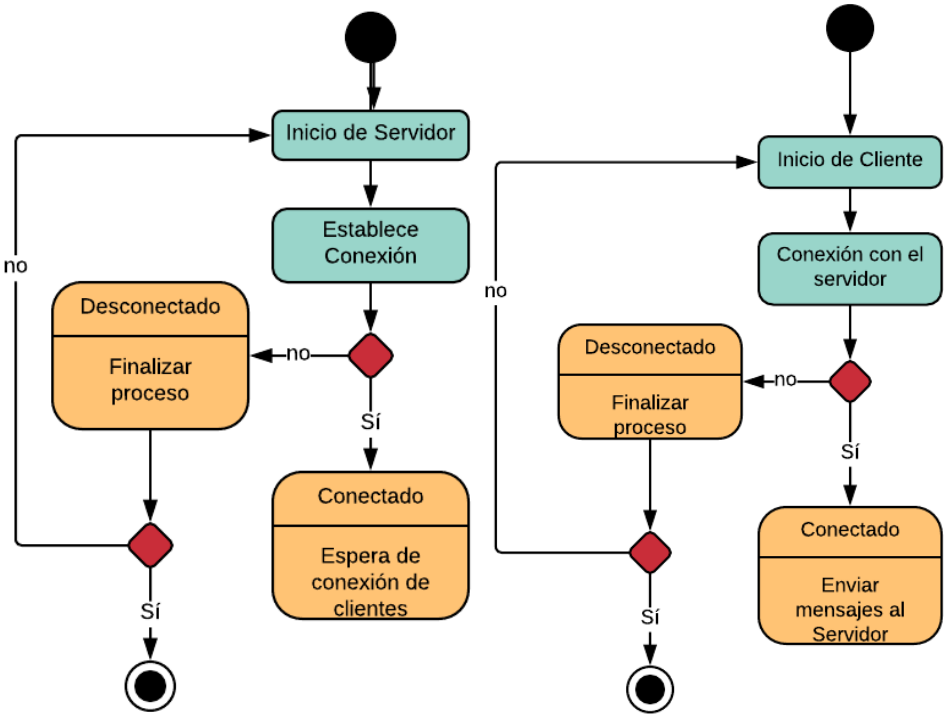


Figura 2.3. Diagramas de actividades UML de Cliente y Servidor.

b. Interfaz Gráfica de Usuario GUI

b.1 Diagrama de clases

Las GUI deberán tener un lugar donde se pueda mostrar el procedimiento DAA a implementar y el resultado de los ataques a evaluar. En la Figura 2.4 se muestra el diagrama de clases para las entidades Nodo y Hub de las WBAN.

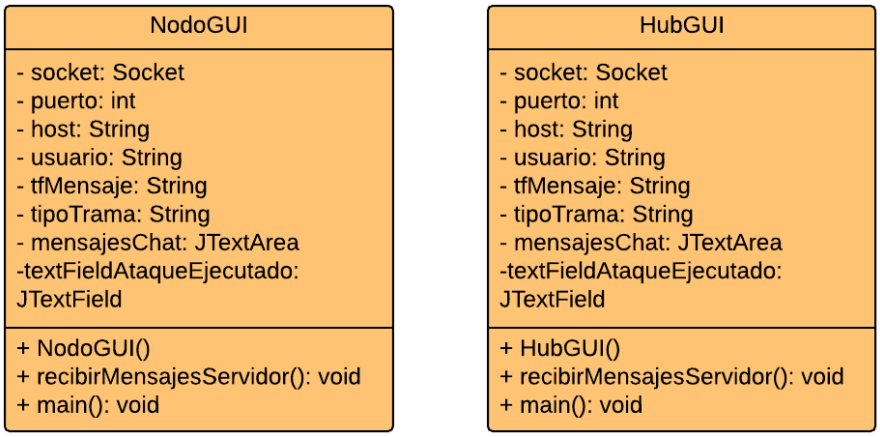


Figura 2.4. Diagramas de clases UML de GUI de Nodo y Hub

b.2 Diagrama de actividades

La apertura y cierre de GUI, así como los procesos entre estas dos actividades, se muestran en el diagrama de actividades UML de la Figura 2.5.

b.3 Bosquejo de la GUI

La GUI deberá contener elementos mínimos como: título de la entidad que representa, cuadro de texto para visualizar el intercambio de tramas, y un *Scroll bar* que permita visualizar mensajes previos. El bosquejo de la GUI se muestra en la Figura 2.6.

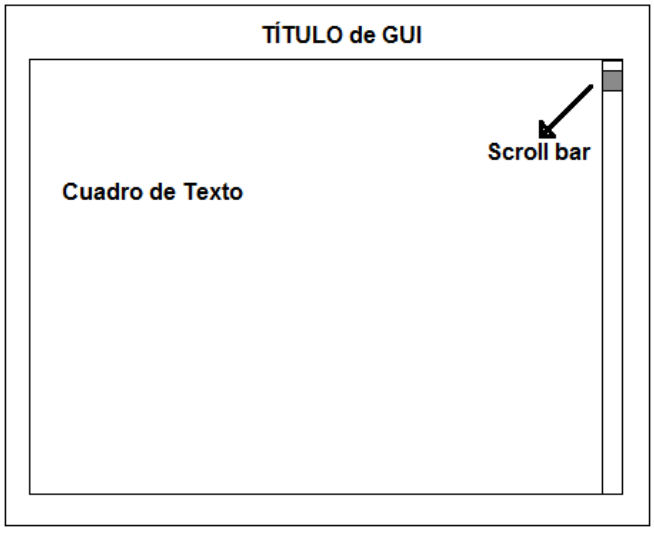


Figura 2.5. Bosquejo de la GUI.

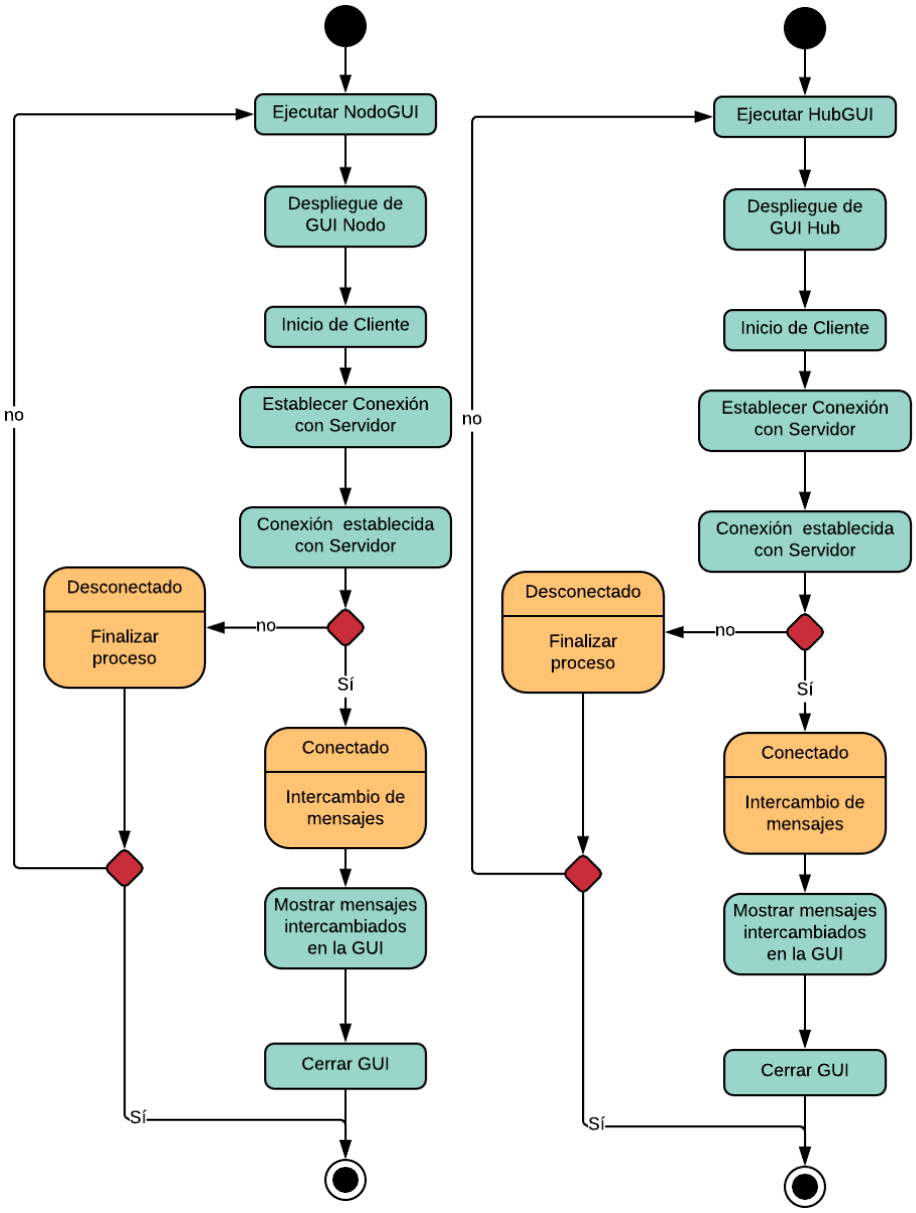


Figura 2.6. Diagrama de actividades UML de las GUI.

c. Protocolo DAA

c.1 Diagrama de secuencia

En la Figura 2.7 se muestra el diagrama de secuencia UML del proceso de Asociación DAA.

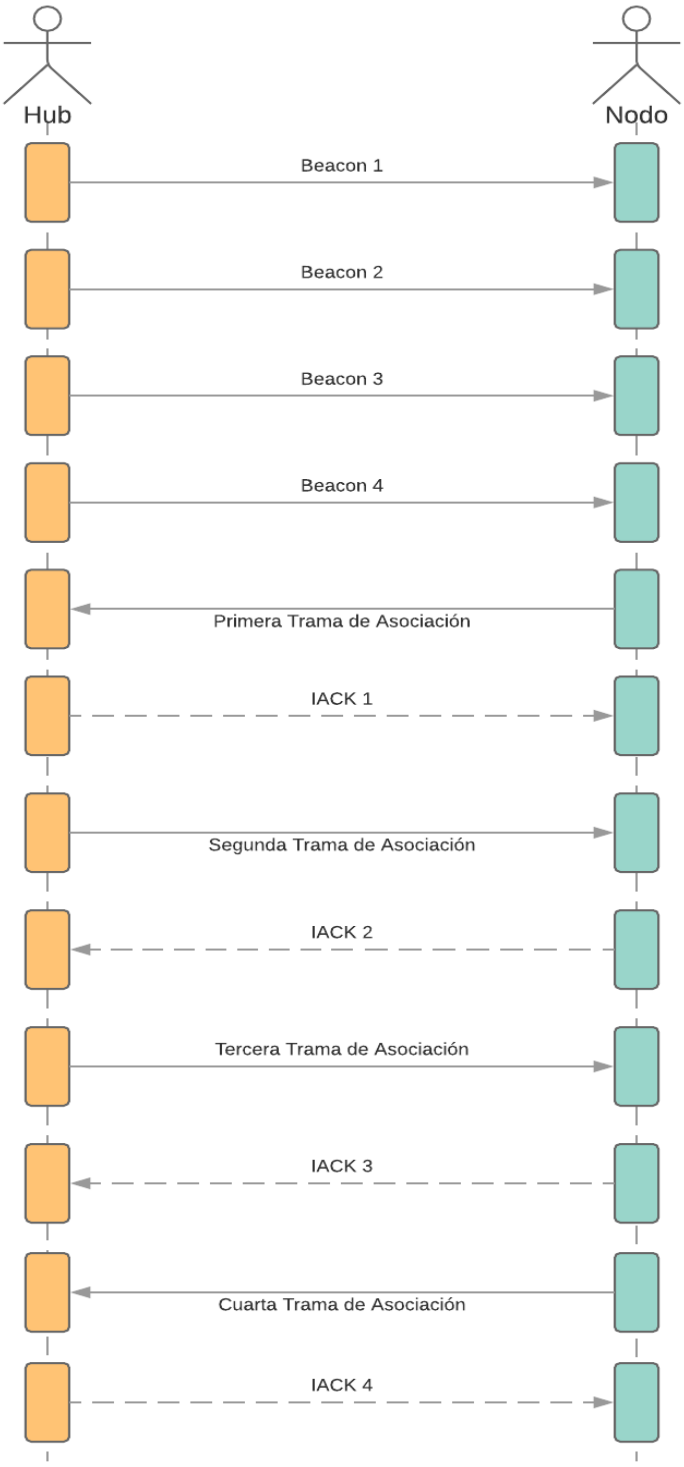


Figura 2.7. Diagrama de secuencia UML de intercambio de tramas entre Nodo y Hub

d. Ataques al DAA

En las Figuras 2.8 y 2.9, se puede visualizar cómo cambia la secuencia del proceso DAA al efectuarse ataques sobre éste, tanto en ataque de Hombre en el medio como en ataque de Impersonalización del Nodo.

d.1Diagrama de secuencia

d.1.1 Impersonalización del Nodo

En la Figura 2.8 se muestra se muestra el diagrama de secuencia UML del ataque Impersonalización del Nodo**.**

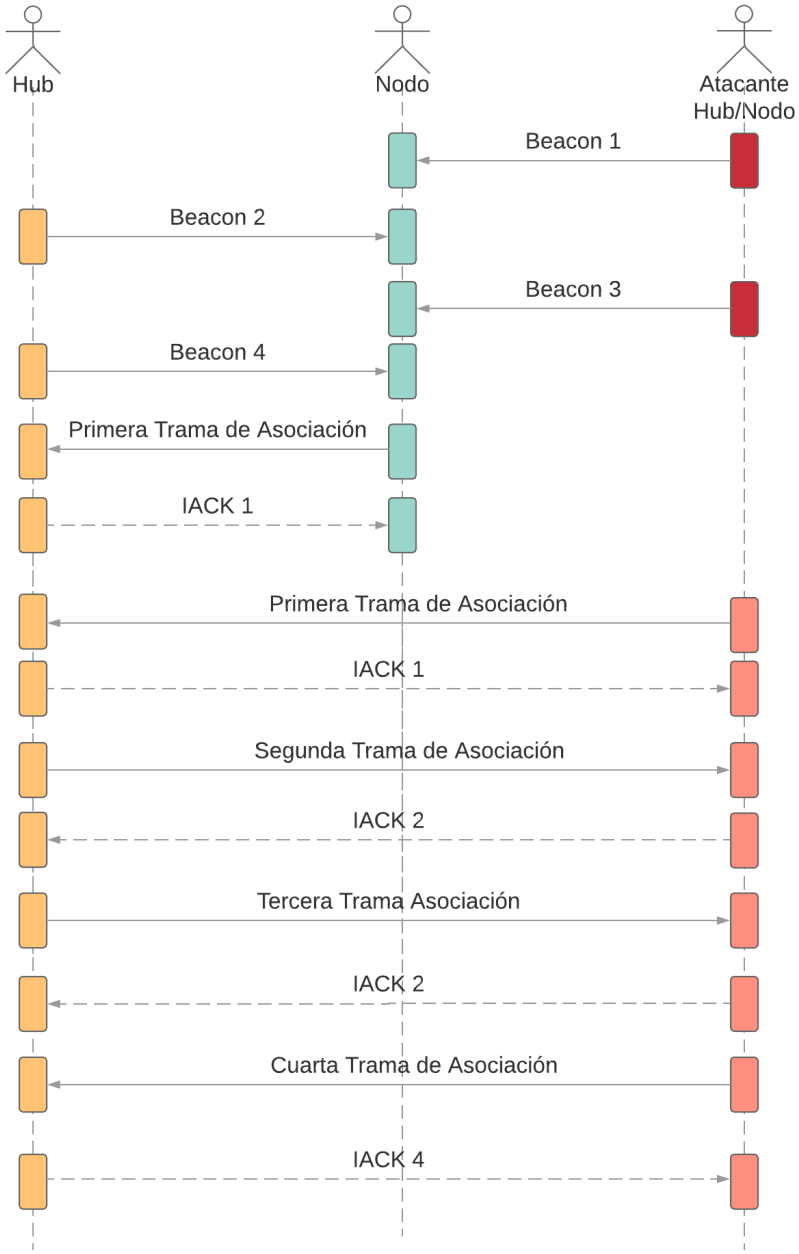


Figura 2.8**.** Diagrama de secuencia UML Impersonalización del Nodo.

d.1.2 Hombre en el medio

En la Figura 2.9 se muestra se muestra el diagrama de secuencia UML del ataque Hombre en el medio**.**

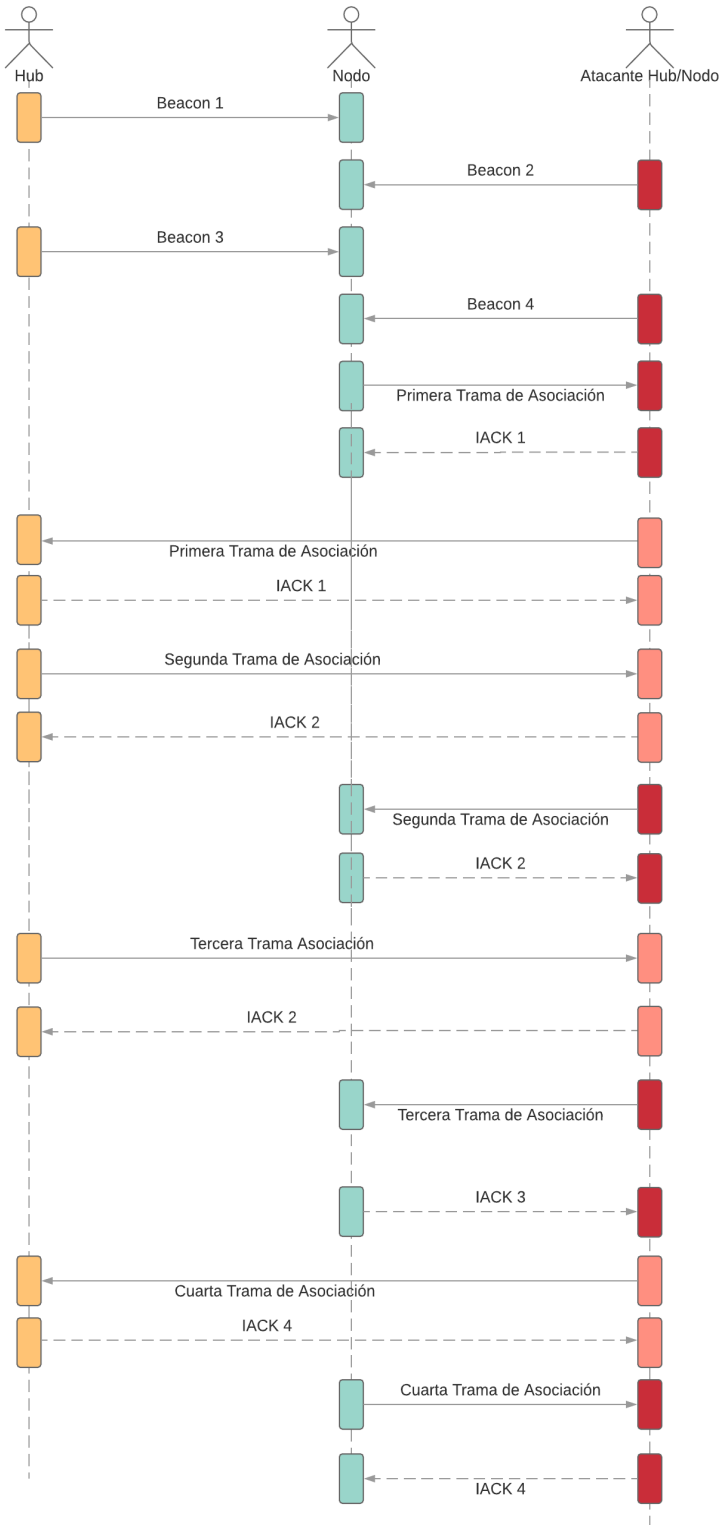


Figura 2**.**9**.** Diagrama de Secuencia UML para Ataque de Hombre en el medio.

2.2 Fase de implementación y simulación

2.2.1 ESCENARIO DE COMUNICACIONES

a. Programación en red modelo Cliente-Servidor

Para la implementación del modelo Cliente –Servidor, se emplea programación en red en lenguaje Java, haciendo uso de la librería “java.net”. Como su modelo lo indica esta parte del desarrollo está dividida en dos secciones importantes, el Servidor y el Cliente. En ambos casos será necesario hacer uso de programación en red con *sockets,* lo cual establece la combinación de dirección IP y número de puerto a nivel de Capa de Transporte, como un punto de acceso al servicio (*Service Access Point* -SAP) entre la Capa de Aplicación y la Capa de Transporte del modelo TCP/IP. También, será necesario manejar la entrada y salida de datos por medio de librerías Java que se encargan de leer corrientes de datos como “java.io”.

a.1 Servidor

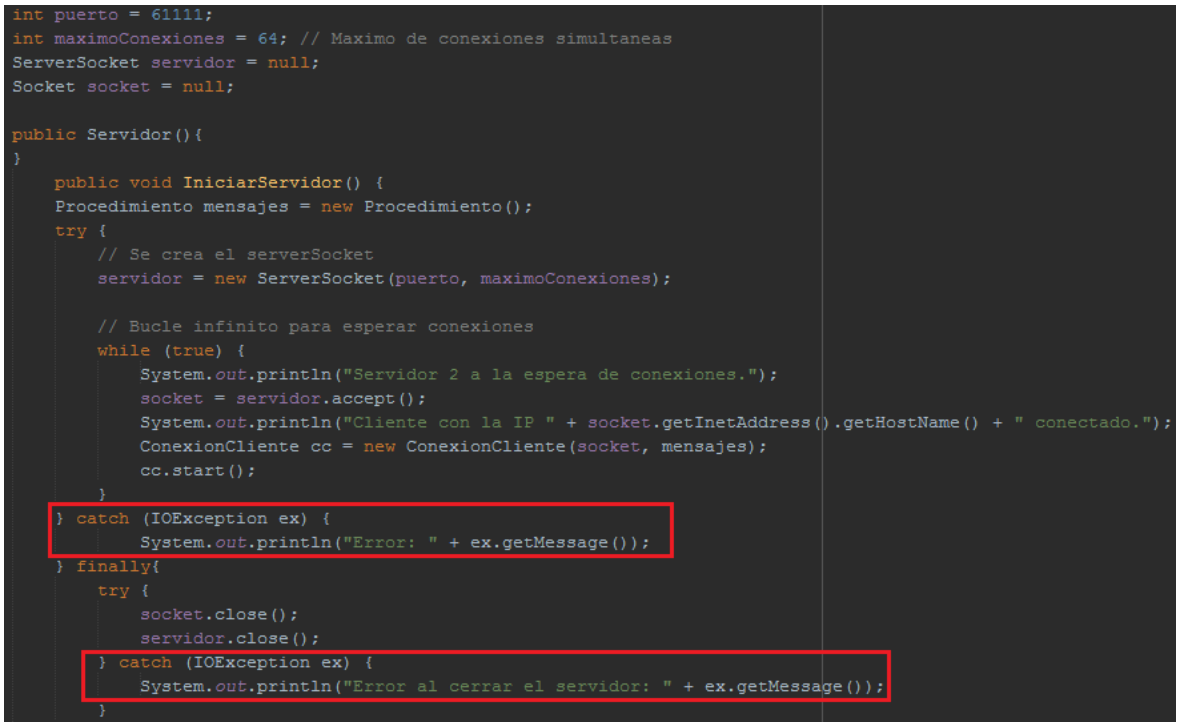
Para la implementación del Servidor, se utiliza un “*ServerSocket”* que permita un número máximo de conexiones. Este “*ServerSocket”* se crea a partir de dos parámetros: la dirección IP que posee el host donde se ejecuta el programa, dato que se toma automáticamente, y el número de puerto seleccionado. Este número de puerto debe estar fuera del rango de los puertos de red bien conocidos y de los registrados; es decir, un número comprendido entre 49151 y 65535.

El Servidor posee las clases detalladas en el diagrama de clases de la Figura 2.2, cada una de las cuales posee sus respectivos atributos y métodos.

a.1.1 Clase Servidor

La clase “Servidor” es la clase principal del Servidor dentro del modelo mencionado. Al momento de ejecutar la clase “Servidor”, por medio del método “IniciarServidor”, se crea un objeto de la clase “Procedimiento”. La clase lleva este nombre debido a que representa todos los mensajes que se intercambiarán entre Cliente y Servidor, los cuales principalmente representan las tramas intercambiadas durante el proceso DAA. Además, el “*ServerSocket”* se crea con los parámetros mencionados anteriormente. También, en el instante que un Cliente solicita conexión con el servidor, éste recibe el mensaje de notificación, de donde extrae la dirección IP del Cliente que requiere conectarse. Para esto, se utiliza la clase “ConexionCliente” la cual apertura un *socket* para el Cliente que se va a conectar tomándolo como parámetro, así como los mensajes que éste enviará; el detalle de este proceso se explicará más adelante a.1.3.

Además, esta clase posee el adecuado manejo de excepciones, para de esta forma procesar de correctamente los errores que se pueden suscitar durante la ejecución de la misma. Es decir que en el caso de que la conexión no sea exitosa, el programa ejecutará las sentencias que se muestran en el cuadro rojo de la Figura 2.10, con el fin de que la ejecución no sea abortada por dichos errores. Concretamente, de existir un error, la clase procede a obtener el error suscitado, mostrarlo en pantalla y finalmente a cerrar los *Sockets* abiertos al inicio de la ejecución.



```
int puerto = 61111;
int maximoConexiones = 64; // Maximo de conexiones simultaneas
ServerSocket servidor = null;
Socket socket = null;

public Servidor(){
}
    public void IniciarServidor() {
    Procedimiento mensajes = new Procedimiento();
    try {
        // Se crea el serverSocket
        servidor = new ServerSocket(puerto, maximoConexiones);

        // Bucle infinito para esperar conexiones
        while (true) {
            System.out.println("Servidor 2 a la espera de conexiones.");
            socket = servidor.accept();
            System.out.println("Cliente con la IP " + socket.getInetAddress().getHostName() + " conectado.");
            ConexionCliente cc = new ConexionCliente(socket, mensajes);
            cc.start();
        }
    } catch (IOException ex) {
            System.out.println("Error: " + ex.getMessage());
    } finally{
        try {
            socket.close();
            servidor.close();
        } catch (IOException ex) {
            System.out.println("Error al cerrar el servidor: " + ex.getMessage());
        }
```

Figura 2.10. Manejo de excepciones de creación y cierre de *ServerSocket* con sentencia *try catch*.

a.1.2 Clase Procedimiento

Como se mencionó anteriormente, su nombre se debe a que cargará todos los mensajes intercambiados entre el "*ServerSocket*" Servidor y el *Socket* Cliente, los cuales, más adelante representarán las tramas que son necesarias transmitir para llevar a cabo el procedimiento de asociación del protocolo DAA. En forma sintetizada, esta clase tiene un atributo tipo *String* que representa los mensajes; como los mensajes serán dinámicos durante todo el procedimiento, será imperativo que estos cambios sean registrados. La forma de llevar a cabo lo mencionado, es por medio del uso de la clase "*Observable*" ya existente en Java, la cual notificará a los observadores que el objeto ha sido modificado, con el fin de que cada que esto suceda se puedan tener los nuevos valores. Por lo

descrito, “Procedimiento” hereda los métodos de la clase “*Observable*” y los utiliza dentro de su propio método “setMensaje”.

Esta clase posee dos métodos propios, que son “getMensaje” y “setMensaje”. El primero, sirve para obtener el valor actual del atributo “mensaje” al momento de utilizar este método. En el segundo, se setea un valor en “mensaje”; y para que esta actualización tenga efecto se deben utilizar dos métodos de la clase “*Observable*”. Para notificar a los observadores que el mensaje ha cambiado se utiliza el método *“setChanged”*, en tanto que “*notifyObservers*” se usa para que cada observador llame internamente al método “*update*”; de esta forma tanto el Cliente como el Servidor podrán tener el mensaje actualizado.

a.1.3 Clase Conexión Cliente

En esta clase se introducen todos los procedimientos para que el servidor pueda aceptar la conexión del cliente por medio de la creación de un *socket* dedicado a su comunicación. La creación de *sockets* dedicados es posible gracias a la clase “*Thread*”, la cual hace posible que se ejecuten varias tareas a la vez, conocidas comúnmente como hilos que comparten los datos del programa y manejan sus propios datos. Es decir, se podrá tener varias conexiones concurrentes sin que una afecte a la otra en la ejecución del programa.

También, será necesario implementar la clase “*Observable*” para poder utilizar los métodos de dicha clase, los cuales servirán para que cada vez que cambia el mensaje en el flujo de datos intercambiados entre el Cliente y el Servidor, todos los observadores puedan leer el nuevo mensaje mediante el método “*run*”; y para que cada vez que una entidad cambie el mensaje, notifique a las demás por medio del método “*update*”.

Como el objetivo de esta etapa de desarrollo es obtener los datos del intercambio de mensajes, será mandatorio utilizar el paquete “*java.io.DataInputStream*” para leer corrientes de datos de entrada y “*java.io.DataOutputStream*” para los datos de salida. Estos paquetes servirán para declarar los atributos de entrada de datos y salida de datos, en donde su valor será el obtenido desde el *socket* por medio de los métodos “*getInputStream*” y “*getOutputStream*” respectivamente.

En el método “*run*” declarado en esta clase, en síntesis, se lleva a cabo el proceso de lectura de los datos que entran por el *socket*. En primer lugar, se setea a “mensajes”, objeto de la clase “Procedimiento”, como un objeto que será observado, por medio del método “*addObserver*”, para que cada vez que éste cambie se pueda leerlo. Posteriormente, se procede a leer el mensaje cargado a entrada de datos propiamente,

por medio del método "*readUTF"* y se lo muestra en pantalla. Finalmente, se procede establecer el mensaje leído en "mensajes" por medio del método "*setMensaje*"; con esto todas las entidades implicadas en el procedimiento sabrán que el mensaje de entrada ha cambiado y podrán obtener sus datos. Cabe recalcar que, este proceso se llevará a cabo en un lazo que no terminará a menos que la conexión sea finalizada.

Por otro lado, se declara el método "*update*" el cual servirá para notificar a los observadores que los datos de salida cargados por el *socket* han cambiado. Además, dentro de este método se envían los datos cargados al *socket* en su salida de datos, mediante el método "*writeUTF*", que tiene como parámetro un *String* que ha sido modificado y notificado por el método "*notifyObservers*". En síntesis, cada vez que se carga un mensaje nuevo a la salida de datos, esto es notificado a los observadores, transformado a código UTF (*Unicode Transformation Format*) y enviado por medio del *socket* a los demás *host* conectados.

a.2 Cliente

En esta parte del desarrollo será importante configurar la dirección IP y el puerto del servidor al que el cliente hará su requerimiento. Adicionalmente, para cada cliente se deberá tener un usuario que diferencie de manera única a éste dentro de la red; así, se identifica el cliente que se conecta y su función de Nodo o de Hub.

a.2.1 Clase Cliente

En esta clase se crea el *socket* para el cliente, a partir de los parámetros de dirección IP del servidor y el puerto. Además, se asigna el usuario para identificarlo en la red, definición de tipo de mensaje que envía y los mensajes a enviar. Se hace uso de la clase "ConexionServidor" para la conexión del Cliente con el Servidor y envío de todos los parámetros mencionados por medio del *socket* hacia el Servidor; el detalle de esto se explica en la siguiente subsección.

Adicionalmente, se crea el método "recibirMensajesServidor" donde se ejecutan los procedimientos de carga y lectura de los datos transmitidos por el Servidor hacia el Cliente; haciendo uso de un atributo del tipo "*DataInputStream*" y los métodos "*getInputStream*" y "*readUTF*" que desarrollan las funciones anteriormente explicadas. Será importante que el Cliente esté recibiendo los datos transmitidos en el sistema durante todo el tiempo que esté conectado, lo cual será considerado en el desarrollo de esta etapa.

Finalmente, se toma en cuenta el uso adecuado del manejo de excepciones, debido a errores que se pueden producir durante el tiempo de ejecución. Se consideró limitar

excepciones debido a errores suscitados por la creación fallida del *socket* y del flujo en entrada de datos como se muestra en la Figura 2.11.

```
host = "192.168.0.1";
puerto = 61111;
usuario = "N1";

System.out.println("Quieres conectarte a " + host + " en el puerto "
        + puerto + " con el nombre de ususario: " + usuario + ".");

// Se crea el socket para conectar con el Sevidor del Chat
try {
    socket = new Socket(host, puerto);
    System.out.println(socket.toString());
} catch (UnknownHostException ex) {
        System.out.println("No se ha podido conectar con el servidor (" + ex.getMessage() + ").");
} catch (IOException ex) {
        System.out.println("No se ha podido conectar con el servidor (" + ex.getMessage() + ").");
}
```

Figura 2.11. Manejo de excepciones de creación de *Socket* y de flujo de entrada de datos con sentencia *try catch*.

a.2.2 Clase Conexión Servidor

Esta clase se enfoca en la conexión del Cliente con el Servidor para el envío de datos hacia el Servidor. Por esta razón se hace uso de un atributo del tipo "*DataOutputStream*" y los métodos "*getOutputStream*" y "*writeUTF*" que desarrollan las funciones explicadas anteriormente. En esta clase se considera limitar los errores en tiempo de ejecución con el debido manejo de excepciones. Se consideran errores debido a la creación incorrecta del *socket* para conexión con el Servidor y a la creación errónea del flujo de datos de salida.

2.2.2 INTERFAZ GRÁFICA DE USUARIO

Una vez implementadas todas las etapas previas de desarrollo, los resultados de la ejecución de la herramienta no serían visibles al usuario final sin la instanciación de una de las clases y el despliegue de información a través de la pantalla. El caso más sencillo permitiría la salida del resultado a través de consola; sin embargo, para conseguir una herramienta mucho más amigable para el usuario se desarrolló en conjunto su propia interfaz gráfica.

Para su implementación se utilizó *WindowBuilder*, donde se puede seleccionar los elementos del *JFrame* a utilizar, así como diseñar el espacio de manera interactiva. Los elementos principales fueron un *JScrollPane*, necesario para montar sobre él un

JTextArea que es el lugar donde se irán mostrando los datos transmitidos por los *sockets*. Adicionalmente, se utiliza un *JTextLabel*, para mostrar la entidad a la que pertenece, Nodo o Hub, y el ataque logrado. En la Figura 2.12 se muestra las GUI implementadas para las entidades con función de Nodo y de Hub, mientras que en la Figura 2.13 se observa la GUI implementada para un atacante que desempeña funciones de Nodo y de Hub simultáneamente.

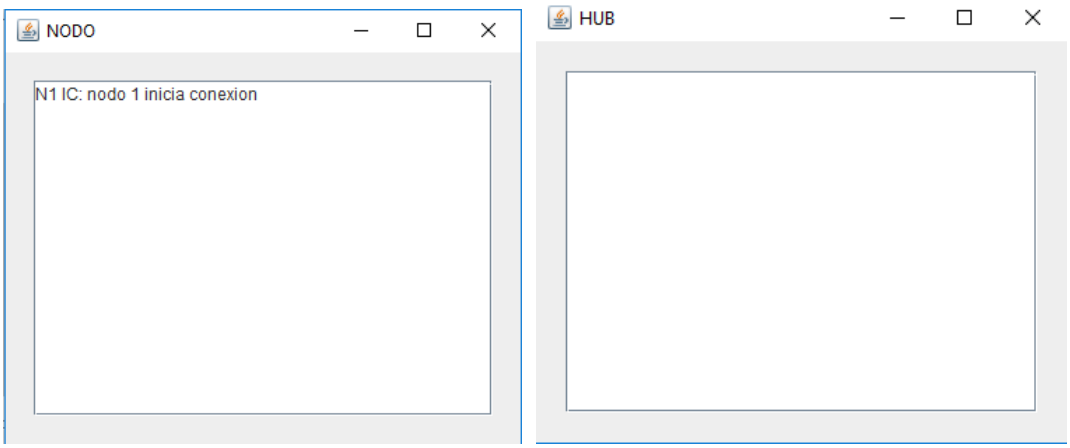


Figura 2.12. Interfaces gráficas de entidades Nodo y Hub.

Figura 2.13. Interfaz gráfica de un atacante con funcionalidades de Nodo y Hub

2.2.3 CRIPTOGRAFÍA DE CURVA ELÍPTICA

a. Generación de llaves privadas SK

Las llaves privadas SK deben ser únicas y aleatorias, con un valor menor a la constante *r* definida en el subcapítulo 7.1 de [1], y con longitud de 256 bits. Por esta razón se implementó el método "*getRandomSK()"*, que genera un valor entero pseudorandómico S_k cumpliendo con los requerimientos de ser menor que la constante *r* y tener una longitud de 256 bits. Este método fue implementado en la clase "*StandardCurve"*, la cual se detallará más adelante en la sección c.3, y su codificación se muestra en la Figura 2.14.

```
public BigInteger getRandomSK(){
        Random rnd = new SecureRandom();
        BigInteger r;
        BigInteger private_key= new BigInteger("1");;

        if (this.id.compareTo("P-256")==0){
                r= new BigInteger("115792089210356248762697446949407573 5"
                        + "2999695522413576034242225906106851204436 9");
                private_key = r;
                do {
                private_key = new BigInteger(256,rnd);
                        }
                        while (private_key.compareTo(r) == 1 ||
                        private_key.compareTo(r) == 0 || private_key.bitLength()!=256);
```

Figura 2.14. Método "*getRandomSK()"* implementado para generación de llaves privadas.

b. Algoritmo de Generación de llaves públicas PK

La fase de diseño e implementación del generador de llaves asimétricas mediante la evaluación de curvas elípticas empieza definiendo el requerimiento en base a la limitante en el cálculo de puntos sobre la curva. Como se indicó anteriormente, las únicas operaciones permitidas son el doble del punto y la suma de un punto con otro. Por ello se debe definir un algoritmo de iteraciones con el fin de optimizar la búsqueda de un punto requerido.

b.1 Algoritmo de Iteraciones

El Algoritmo de Iteraciones tiene como objetivo alcanzar un punto requerido $R = P_k$ partiendo de un punto G y minimizar el número de pasos hasta obtener el valor deseado. Como se expresa en la Ecuación 1.16, el valor de P_k es igual al resultado de la multiplicación escalar entre el punto G y un valor entero S_k sobre un campo finito p. El algoritmo toma como argumento el valor de la llave privada (S_k) y encuentra el camino

más rápido partiendo de $S_k = 1$. La Figura 2.15 ilustra, mediante un diagrama de actividades UML, el correspondiente algoritmo.

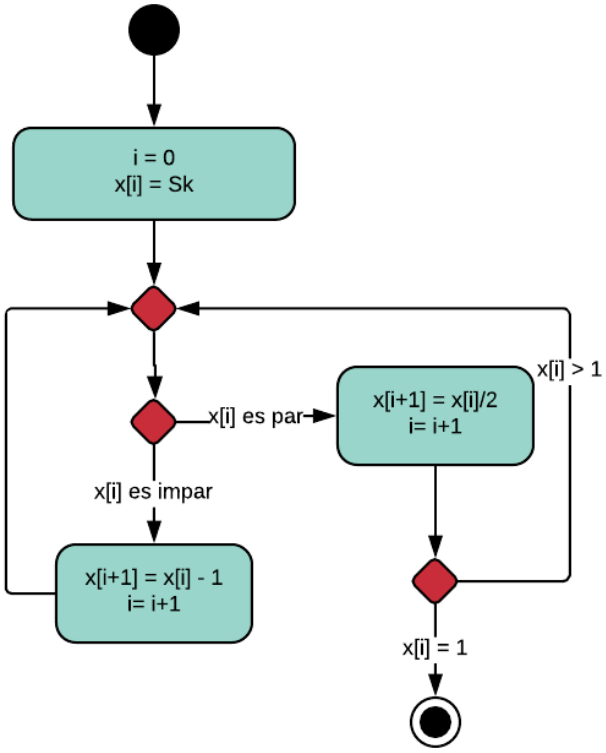


Figura 2.15. Diagrama de Actividades UML del Algoritmo de Iteraciones.

Una vez determinados todos los valores para alcanzar al entero S_k, se procede al cálculo de cada uno de los puntos. Este procedimiento involucra computación modular. Mediante Java el tipo de dato *BigInteger*, definido dentro de su clase, contiene un conjunto de métodos que trabajan tanto en aritmética clásica como en aritmética modular como por ejemplo "*mod()", "modInverse()", "multiply()", "pow()*", etc., lo cual facilita la implementación de un método de cálculo.

c. Clases

Para el generador de llaves, se han definido tres clases como se muestra en el diagrama de clases ilustrado en la Figura 2.16.

c.1 Clase Curve

La primera clase "*Curve"* es abstracta y contiene los valores de los parámetros de la curva como atributos, además contiene cinco métodos que se heredarán en las subclases y al método abstracto *"getPublicKey()".*

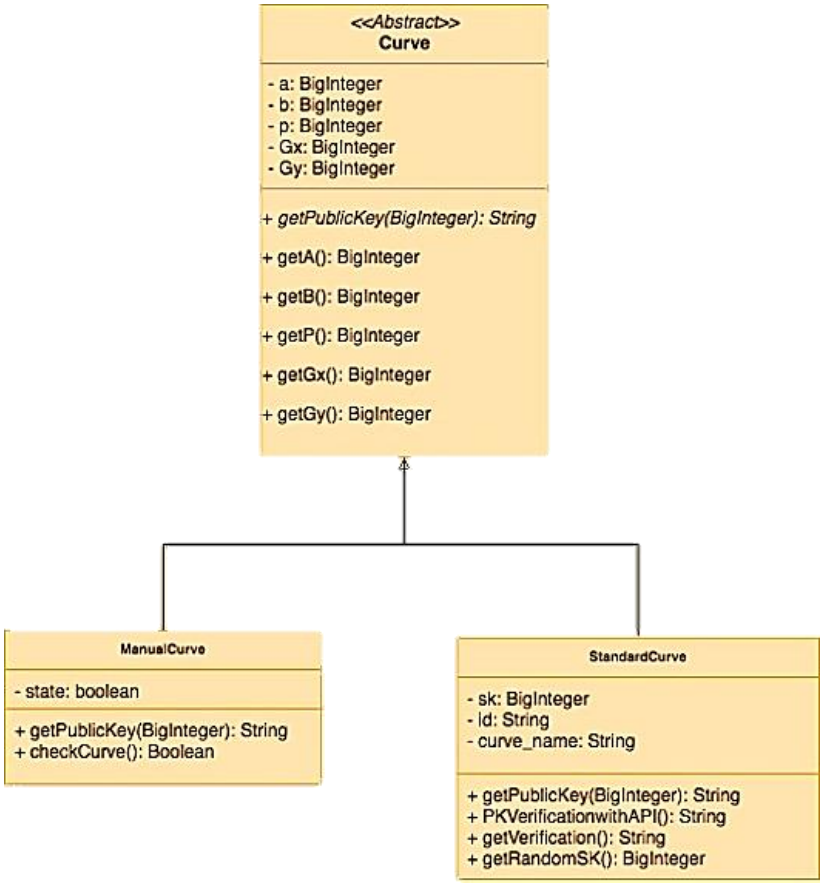


Figura 2.16. Diagrama de Clases UML del Generador de $\boldsymbol{P_k}$.

c.2 Clase Manual Curve

La segunda clase "*ManualCurve"* heredará los atributos de la clase *Curve* y los métodos ahí definidos. La clase se verá obligada a implementar el método *getPublicKey(),* el cual retornará el valor de la clave pública P_k. La clase "*ManualCurve"* a más del método heredado implementa un método de comprobación de curva "*checkCurve()"*; este método determinará si se cumple la condición de los parámetros de la curva elíptica definidos en la Ecuación 1.13. Esta clase es muy importante para el cálculo de la llave *Diffie-Hellman*, necesaria dentro del proceso del protocolo DAA.

c.3 Clase Standard Curve

La clase "*StandardCurve", detallada* en el Anexo II, heredará los atributos de la clase *Curve* y los métodos definidos en esta. También, implementa tres métodos a más del "*getPublicKey()"*; éstos son "*PKVerification()", "getVerification()" y "getRandomSK()".* El método "*getRandomSK()"* genera un valor entero pseudorandómico S_k, como se mencionó anteriormente; mientras que los métodos "*PKVerification()"* y "*getVerification()"* invocan a algunas clases de la *API Bouncy Castle Provider* con el fin de comprobar el

valor devuelto por el método abstracto implementado. Esta clase es sumamente importante para la generación de llaves públicas P_k a partir de la llave privada S_k y el punto de la curva *G*, utilizando la curva estandarizada P-256, según se menciona en el estándar 802.15.6[1].

2.2.4 IMPLEMENTACIÓN Y SIMULACIÓN DE SERVICIOS DE SEGURIDAD SEGÚN EL ESTÁNDAR IEEE 802.15.6

a. Algoritmo AES

Dentro de las clases mencionadas, se puede observar que existe la clase AES con tres métodos que permiten el cálculo de una encripción AES de 128 bits; estos tres métodos son:

- *EncriptAES*
- *encryptText*
- *bytesToHex*

El método *"EncriptAES"* devuelve el valor en hexadecimal del mensaje encriptado con AES-128. El método "*encryptText"* devuelve el mensaje encriptado, mientras que el último método transforma de formato *bytes* a hexadecimal.

Para obtener un mensaje encriptado, una vez creado un objeto de la clase AES, se hace el llamado del método "*EncriptAES"*, con el mensaje a encriptar y la llave para el proceso como atributos.

Este método realiza una comprobación previa de la llave que ingresa al método; si las llaves no tienen una longitud par, éstas serán modificadas para que no existan mensajes de error dentro del proceso. Con el método "*parseHexBinary"* de la clase "*DatatypeConverter"* la llave ingresada en forma de *String* se convertirá en un *array* de *bytes*, necesarios para continuar con el proceso. Dentro de los procesos de las librerías "*javax.crypto"* la llave ingresada requiere no únicamente la transformación de ésta en *array* de *bytes*, es necesario que ésta sea construida como llave secreta. La construcción de la llave secreta necesaria para la encripción con AES se realiza mediante el establecimiento de un objeto de la clase "*SecretKeySpec"*; este objeto admitirá la llave en *array* de *bytes* y el tipo de algoritmo, los cuales serán utilizados como atributos dentro del método constructor de la clase.

Realizada la preparación de la llave ingresada dentro del método, es necesario hacer el llamado del método que realiza la encripción con AES; debido a eso dentro del método "*EncriptAES"* se encuentra el llamado del método "*encryptText"*. Este método recibe como atributos de entrada el mensaje en forma de *String* y la clave secreta construida.

Dentro del método se hacen dos procedimientos previos a la encripción en AES. El primer procedimiento es la creación y acondicionamiento de un vector de inicialización; como se ha expuesto en el capítulo 1, para la encripción usando AES la manera más segura de realizarlo es con AES-CBC; este método es aquel que utiliza la llave correspondiente y el vector de inicialización. El segundo procedimiento necesario es la conversión de *String* a *array* de *bytes* por parte del mensaje a encriptar.

Para la creación y acondicionamiento del vector de inicialización se empieza con la creación de una cadena de *String* que representa la cadena hexadecimal de la llave a ser utilizada como vector de inicialización para AES. Esta cadena será convertida en *array* de *bytes* utilizando de igual forma el método "*parseHexBinary"* de la clase "*DatatypeConverter".* Una vez transformado en *array* de *bytes*, esta cadena dará lugar a un objeto creado en forma de vector de inicialización utilizando la clase "*ivParameterSpec".* Para la creación de un objeto del tipo "*ivParameterSpec"* se debe colocar como atributo del método constructor el *array* de *bytes* ya transformado. El objeto que se crea servirá para hacer la encripción del mensaje.

Una vez preparados todos los elementos se crea e inicializa un encriptador. Para la creación del encriptador es necesario un objeto de la clase "*Cipher"* y utilizar para ellos el método "*getInstance"*. Dentro del método *"getInstance"* se coloca el tipo de encriptador a utilizar como atributo del método; entre los tipos de encriptadores que se pueden crear están:

- AES/CBC/NoPadding (128)
- AES/CBC/PKCS5Padding (128)
- AES/ECB/NoPadding (128)
- AES/ECB/PKCS5Padding (128)
- DES/CBC/NoPadding (56)
- DES/CBC/PKCS5Padding (56)
- DES/ECB/NoPadding (56)
- DES/ECB/PKCS5Padding (56)
- DESede/CBC/NoPadding (168)
- DESede/CBC/PKCS5Padding (168)
- DESede/ECB/NoPadding (168)
- DESede/ECB/PKCS5Padding (168)
- RSA/ECB/PKCS1Padding (1024, 2048)
- RSA/ECB/OAEPWithSHA-1AndMGF1Padding (1024, 2048)

- RSA/ECB/OAEPWithSHA-256AndMGF1Padding (1024, 2048)

Para propósitos de este trabajo, se utilizará una encripción *AES/CBC/NoPadding.* Esta encripción implica que se utilizará encripción AES con un modo de operación CBC (*Cipher Block Chaining*) sin relleno de bits (*No Padding*). Este modo de operación utiliza el vector de inicialización ya mencionado y no realiza un relleno de bits; asegura que su salida sea de 128 bits independientemente de la entrada que tenga. Otro modo de operación que serviría para los propósitos de este trabajo es *AES/ECB/NoPadding.* Este modo de operación presenta algunas desventajas en relación a lo que ofrece el modo CBC, su seguridad es menor debido a que no posee vector de inicialización y tampoco encripción secuencial de bloques.

Una vez creado el objeto de la clase "*Cipher"* es importante iniciarlo en modo encripción, para esto se usa el método *init*. Este método "*init"* es un método "*void"* que no retorna ningún valor, pero prepara al objeto para realizar la encripción del mensaje. Dentro de los atributos para el método "*init"* se encuentran el modo de funcionamiento, la llave y el vector de inicialización. El modo de funcionamiento corresponde a si se desea que el objeto mencionado trabaje en modo encripción (*Object.ENCRYPT_MODE)* o en modo desencripción *(Object.DECRYPT_MODE).* Para efectos de este trabajo únicamente será necesario iniciar el objeto en modo encripción.

Cabe recalcar que el método no entregará un error si el vector de inicialización es olvidado como atributo, el mismo escogerá un vector de inicialización randómico de 128 bits para suplir las necesidades del objeto. Esta opción no es válida para este trabajo debido a la necesidad de obtener siempre el mismo resultado de 128 bits si un mismo mensaje se ingresa al encriptador.

Para finalizar la encripción, al mensaje a encriptar tal y como se ha realizado a lo largo de esta sección para los elementos dentro de la clase "*java.crypto",* es necesario convertirlo de *String* a *array* de *bytes*. Además, se utiliza el método "*doFinal".* El método "*doFinal"* retorna en forma de *array* de *bytes* el mensaje encriptado con las condiciones iniciadas con el método "*init"*; si se desea cambiar alguna clase o vector de inicialización se requiere una nueva inicialización del encriptador. Únicamente siendo necesario la trasformación de este *array* de *bytes* en forma de *String* hexadecimal para su uso. El mensaje a encriptar se lo considera como texto plano y la salida como texto encriptado utilizando AES-128.

En *Java Platform SE 8* y su clase "*java.crypto"*, la que se ha utilizado para desarrollar este procedimiento, se acepta una longitud máxima de 128 bits tanto para la llave como para el vector de inicialización. Pero de lo desarrollado teóricamente en el capítulo 1, uno de los requerimientos del proceso de Asociación es generar llaves de 256 bits para el proceso CMAC, el que a su vez requiere imperiosamente utilizar AES-128 bits. Como se puede observar el programa entraría en conflicto si se hace un llamado del método encriptador con una llave de 128 bits. Esto se soluciona instalando una extensión de las políticas de seguridad para Java. La instalación consiste en descargar y descomprimir los archivos Java *Cryptography Extension* (*JCE*) *Unlimited Strength Jurisdiction Policy Files* y copiarlos dentro de la dirección *%JAVA_HOME%\jre1.8.0_131\lib\security*.

La instalación de las nuevas políticas de seguridad muestra que la longitud máxima de llave que se puede ingresar dentro del encriptador AES es 2147483647 bits, pero esto no se cumple ya que AES tienen estandarizadas las longitudes de llave como se muestra en el capítulo 1. Si se ingresan llaves de longitud no estandarizada dentro del programa, se generará una excepción debido a que la librería con la cual está desarrollado cumple con las normas y estándares determinados para el proceso.

b. Algoritmo CMAC

El código de autenticación de mensajes basado en cifrado (CMAC) corresponde a una de las principales herramientas, con la cual será desarrollado todo el proceso del presente trabajo a lo largo de todas sus etapas. La presente sección explicará el desarrollo de este código con sus diferentes algoritmos. CMAC partirá de los conceptos generados en el capítulo 1 y utilizará como base la encripción AES descrita en la sección anterior.

El método CMAC buscará desarrollar el procedimiento descrito en el capítulo 1; para ello este método tendrá 3 atributos de entrada para realizar las operaciones necesarias. Los atributos que admitirá este método son la llave, el mensaje a codificar y el número de bits de salida. La salida de este método será una cadena o *String* hexadecimal equivalente en octetos al número de bits especificados en la entrada para el mensaje codificado con CMAC.

b.1 Generación de Sub llaves K1 y K2

Previo a realizar el algoritmo CMAC como tal, es necesario tener preparadas las subllaves (*Subkey*) K1 y K2 del procedimiento antes mencionado. Las llaves secundarias son derivadas de la llave principal K la cual es utilizada para la codificación con AES. Las llaves secundarias servirán para trabajar en la última iteración del algoritmo de CMAC dependiendo de si el mensaje tiene una longitud múltiplo exacta de 16 octetos; para este

caso se utilizará la llave secundaria K1 ya que no es necesario completar el mensaje. Para el caso donde sea necesario completar el mensaje, debido a que no es múltiplo de 16 octetos, será necesaria la utilización de la llave secundaria K2.

El método "*Subkey"* es un método que acepta como atributo de entrada la llave principal K en forma de *String* y retorna un *array de* elementos *BigInteger* que contendrá a las dos llaves secundarias.

El primer paso del método "*Subkey"* es realizar una encripción AES de un mensaje de ceros con longitud 16 octetos y la llave K mediante el llamado del método "*EncriptAES"* de la clase "AES". Debido a que el retorno del método "*EncriptAES"* es en *String*, es necesario generar un elemento objeto de la clase *BigInteger* para hacer el respectivo almacenamiento de ese dato; este dato será llamado L.

De L se debe obtener el estado del bit más significativo (msb(L)). Si msb(L) es igual a cero L deberá sufrir un desplazamiento hacia la izquierda de 1 bit y éste corresponderá al valor de la llave secundaria K1. Si msb(L) no es igual a cero, el dato L sufrirá un desplazamiento hacia la izquierda de un bit y posteriormente se aplicará una operación *XOR* con la constante Rb=00000000000000000000000000000087 en valor hexadecimal, y este corresponderá al valor de la llave secundaria K1. Las operaciones de desplazamiento se realizan con el método "*shifLeft"* y la operación con el método *XOR.* Debido a problemas generados por los desplazamientos y los bits sobrantes que resultan, se realiza una comprobación adicional. Se debe comprobar si la longitud de L corresponde a la misma longitud de K1, si esto no es así se debe eliminar el primer bit de K1 ya que el proceso ha dado lugar a un bit excedido con valor en 1.

K2 se deriva del valor K1 ya hallado. Si msb(K1) es igual a cero K1 deberá sufrir un desplazamiento hacia la izquierda de 1 bit y éste corresponderá al valor de la llave secundaria K2. Si msb(K1) no es igual a cero, el dato K1 sufrirá un desplazamiento hacia la izquierda de un bit y posteriormente se aplicará una operación *XOR* con la constante Rb y corresponderá al valor de la llave secundaria K2*.* Debido a problemas generados por los desplazamientos y los bits sobrantes que resultan, se realiza una comprobación adicional tal y como se realiza en la llave K1. Se debe comprobar si la longitud de K2 corresponde a la misma longitud de K1, si esto no es así se debe eliminar el primer bit de K2, ya que el proceso ha dado lugar a un bit excedido con valor en 1. El algoritmo de obtención de llaves secundarias se presenta en la Figura 2.17.

Una vez obtenidas las llaves secundarias K1 y K2 dentro de un *array de* elementos *BigInteger* llamado K1_K2[ ], en el que K1_K2[0] correspondería a K1 y K1_K2[1] correspondería a K2, se prosigue con el algoritmo CMAC.

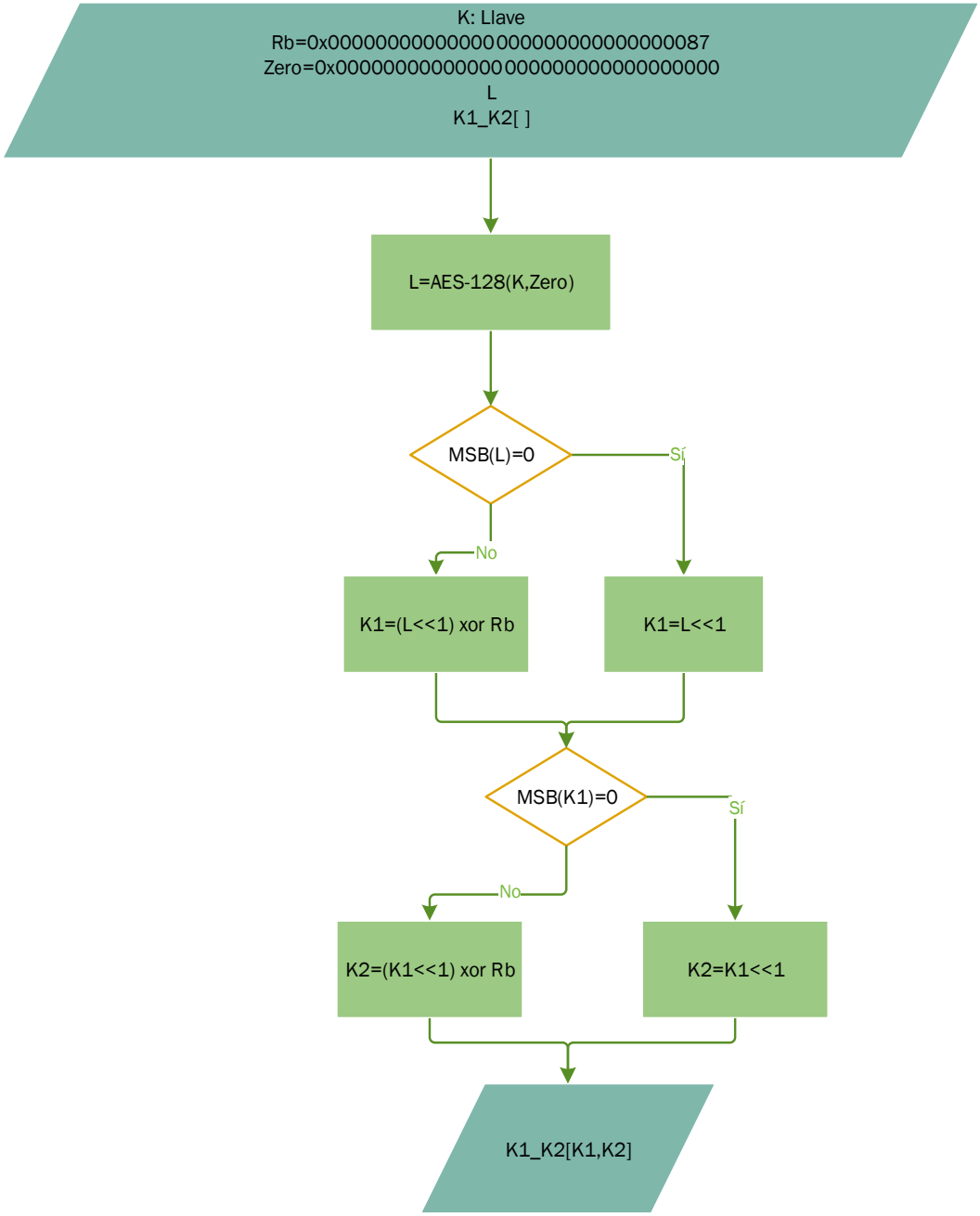


Figura 2.17. Algoritmo de cálculo llaves secundarias.

b.2 Selección del tipo de codificación en CMAC

Para proseguir con el algoritmo es necesario comenzar determinando la cantidad de octetos que el mensaje a codificar tiene. Este valor es importante ya que dará la cantidad

de iteraciones a realizar dentro del algoritmo y también entregará información para conocer si se utiliza K1 o K2 como llaves en el último bloque.

Para determinar la cantidad de octetos que contiene el mensaje a codificar, se toma la entrada *String* y con el método *length* se establece la cantidad de caracteres de esa cadena y se los divide para dos. Se divide debido a que en un formato hexadecimal un carácter corresponde a 4 bits teniendo como resultado dos caracteres por octeto en el mensaje. También es importante conocer la cantidad de mensajes que se tendrán; esto se consigue aproximando hacia arriba la división entre la cantidad de octetos del mensaje para 16, todo esto con el método *ceil* de la clase "*Math".* Adicionalmente se determina la cantidad de octetos sobrantes, si es que el mensaje no tiene una longitud múltiplo de 128 bits (16 octetos). Para esto se utiliza el operador % para tener el residuo entre la cantidad de mensajes y 16; si este residuo es cero se usará K1, si tiene un valor diferente de cero se utilizará K2.

Con la información de la cantidad de mensajes, se deberá activar una bandera, esta bandera es un dato *booleano* que permitirá escoger si se utiliza K1 o K2. Para esto y tal como describe el RFC 4493[9] si el número de mensajes es igual a cero es necesario colocar un valor de n=1 y la bandera en falso. El valor de la bandera se colocará en falso debido a que este valor determina si es necesario completar el mensaje o no; de la misma forma lo necesitarán los mensajes a pesar de que tengan la cantidad de mensajes diferente a cero y no sean múltiplos de 16 octetos. Adicionalmente a esto se deberá colocar el número de mensajes en 1 y el valor de n será utilizado en posteriores procedimientos.

Si el número de mensajes es diferente de cero implica que la longitud del mensaje sobrepasa al menos una vez el valor de 16 octetos; sin embargo, es propicio verificar el número de octantes sobrantes para continuar con el procedimiento. Si el número de octantes es igual a cero la bandera se colocará en verdadero y n en 32. Mientras que, si el número de octantes sobrantes es diferente de cero, implica que es necesario completar los octetos y la bandera se pondrá en falso y n en cero. Todo este procedimiento se visualiza en la Figura 2.18.

b.3 Tipo de mensajes posibles a codificar en CMAC

Los mensajes marcados como verdadero corresponden a bloques completos dentro de CMAC, a diferencia de los mensajes marcados como falso que son considerados como bloques incompletos. Esto implica de manera concluyente que existen 3 tipos de mensajes a codificar con CMAC:

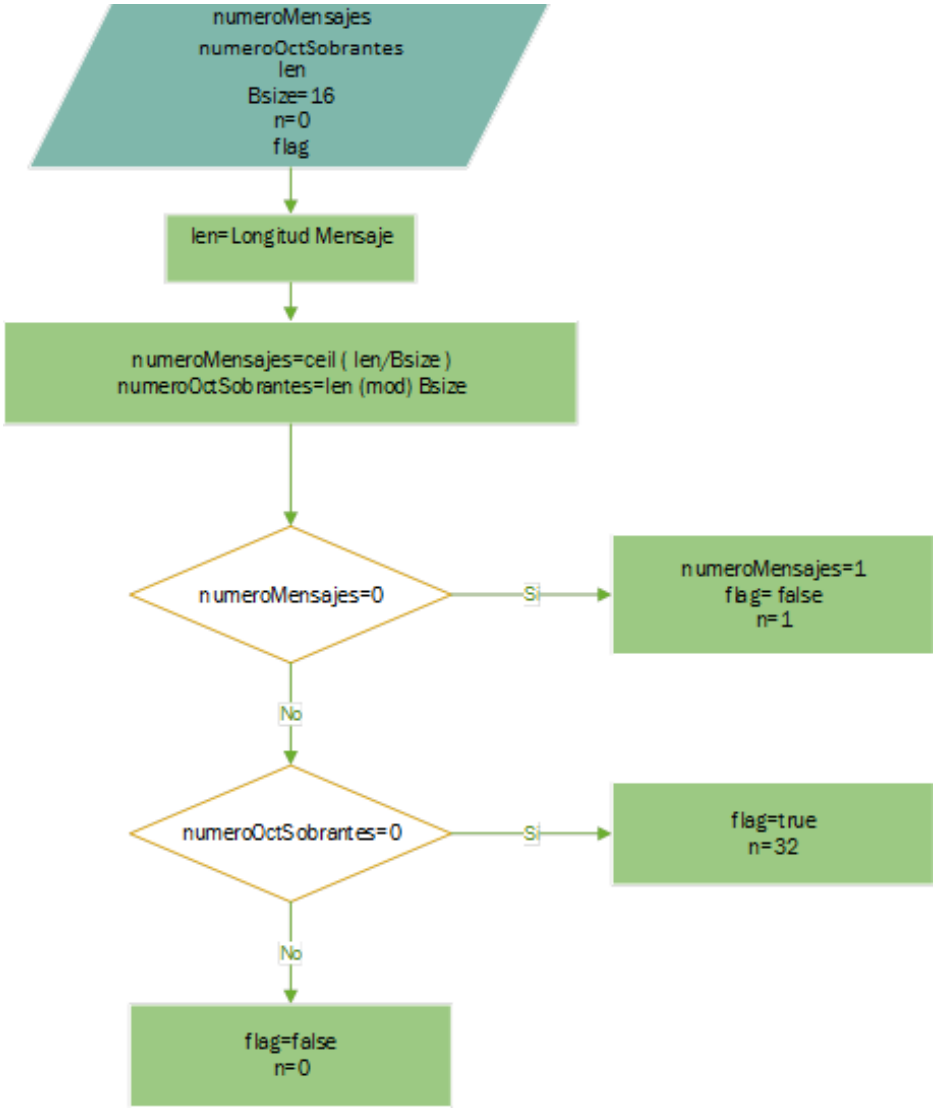


Figura 2.18. Fase de selección de tipo de codificación.

- Mensajes que no superan los 16 octetos que deberán llevar el estado falso para ser completados llevándolos a un mensaje de 16 octeto completos.
- Mensajes que superan al menos una vez la cantidad de 16 octetos, pero no son múltiplos exactos significando también que su estado deberá estar en falso y el último bloque será tratado de la misma forma que el caso anterior.
- Mensajes que superan al menos una vez los 16 octetos y que si son múltiplos exactos de 16; éstos deberán estar marcados como verdaderos ya que el último bloque de éste no necesitará completación y será tratado de manera diferente.

b.4 Procesos sobre el último mensaje en CMAC

Una vez entendidos los tres casos que se pueden presentar en una codificación CMAC, es necesario obtener el último bloque a codificar, considerando que para el primer caso es el último y único bloque. Con la idea de generalizar el proceso de extracción del último bloque conocido como *Mensaje_n* y considerando que el método *"subString"* de la clase "*String"* necesita el número de carácter de inicio incluido y el número de carácter final sin incluir para la extracción de la cadena, este método recibirá como carácter de inicio el valor de la Ecuación 2.1 y como argumento final el de la Ecuación 2.2.

$$(numeroMensajes - 1) * 32$$

Ecuación 2.1. Obtención valor inicial en método *"subString"* para el último mensaje.

$$((numeroMensajes - 1) * 32) + n + numeroOctSobrantes * 2$$

Ecuación 2.2. Obtención valor final en método "*subString"* para el último mensaje.

Por ello, dentro de los condicionales se colocan no solo los valores de verdadero o falso para los estados de los bloques, también se colocan los valores de n que permitan que las ecuaciones cumplan con su objetivo. Por ello n:

- En el primer tipo de mensaje tomará de cero al valor uno más el número de octetos sobrantes por dos.
- En el segundo tipo de mensaje tomará desde el número de mensajes menos uno por 32, esto debido a que se consideran 32 caracteres por bloque, hasta ese mismo valor sumado la cantidad de octantes sobrantes por 2.
- Para el tercer caso se tomará desde el número de mensajes menos uno por 32, hasta el valor final que se considera el valor de inicio más 32 ya que no existe valor en el número de octetos sobrantes.

Una vez considerado el *subString*, se debe tomar acción con respecto a la bandera determinada en los procesos anteriores. Si la bandera es verdadera (*true*), el último bloque llamado *M_last* un elemento *BigInteger* creado a partir del *subString Mensaje_n* formará parte de la operación *XOR* con la llave K1 (K1_K2[0]), esto utilizando el método *XOR* de la clase "*BigInteger"*.

Si la bandera es falsa (*false*), el *subString* de *Mensaje_n* deberá concatenarse con una cadena de uno y seguido de tantos ceros en términos de bits hasta que el tamaño del

mensaje *Mensaje_n* sea de 128 bits o 32 caracteres. Para esto, una variable de tipo *String* con un ocho en el primer carácter y seguido de 15 caracteres cero representaría lo mencionado al convertir a hexadecimal, esto permitiendo tomar la subcadena a concatenarse cumpliendo la condición del mensaje *Mensaje_n.* Esto se realiza en base a la información de número de octetos sobrantes. Este proceso se denomina *Padding*, es decir un bloque con *Padding* contará con los octetos sobrantes y una cadena de ocho seguido de ceros contando con una longitud total 32 caracteres en forma hexadecimal.

Para la concatenación se tomará una subcadena desde cero hasta 16 menos la cantidad de octetos sobrantes por dos y el mensaje *Mensaje_n.* Para ello se utiliza el método *concat* de la clase "*String".* Posteriormente se crea el último bloque llamado *M_last*, elemento *BigInteger* creado a partir del *Mensaje_n* con *Padding*; este elemento formará parte de la operación *XOR* con la llave K2 (K1_K2[1]), utilizando el método *XOR* de la clase "*BigInteger*" de la forma en la que el algoritmo de la Figura 2.19 lo describe.

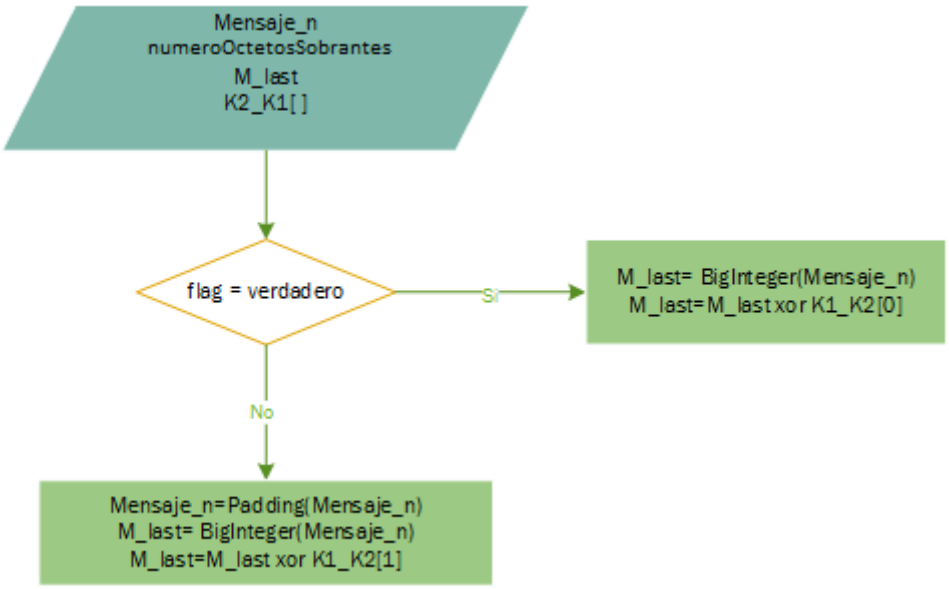


Figura 2.19. Proceso sobre último mensaje en CMAC.

b.5 Proceso iterativo en CMAC

Todo lo realizado antes, forma parte de la preparación de los bloques previo a su encripción con AES, tanto los bloques parciales como el último bloque, el cual como se ha visualizado tiene un tratamiento especial. Por ello, adicionalmente a eso se crean dos variables *BigInteger* llamadas X e Y, seteadas en valor inicial de cero para que vayan tomando los valores de los bloques a codificar en CMAC. Además, existirá una variable

llamada “Xaes” la cual será la salida del método AES conteniendo el bloque encriptado seteada originalmente en vacío(*null*).

Dentro de un lazo *for* donde se inicializa en uno hasta el valor del número de mensajes, se colocará todo el proceso de partición y encripción de los bloques CMAC. Del mensaje a codificar se toman bloques de 32 caracteres en cada iteración. Los bloques se irán tomando en función de la variación del valor de iteración y evitarán que se codifique el último bloque *M_last.* Cada bloque crea un elemento *BigInteger* llamado Y, este elemento como lo indica las Figuras 1.16 y 1.17 debe formar parte de una operación *XOR* con su mensaje codificado predecesor llamado X. Este elemento X en cada iteración será encriptado con AES-128, con la misma llave en forma de cadena e ingresada como atributo en el método CMAC descrito anteriormente. En la iteración inicial la operación se realizará con una cadena de ceros mientras que en las siguientes iteraciones ya contará con un valor proveniente de una encripción AES-128 previa para poder ser operado con un nuevo bloque del mensaje.

El último mensaje o bloque *M_last,* el cual ya formó parte de una operación *XOR* ya sea con K1 o K2 como se indica en las Figuras 1.16 y 1.17, se deberá operar con el último mensaje encriptado dentro del lazo *for*. El proceso iterativo se describe en la Figura 2.20.

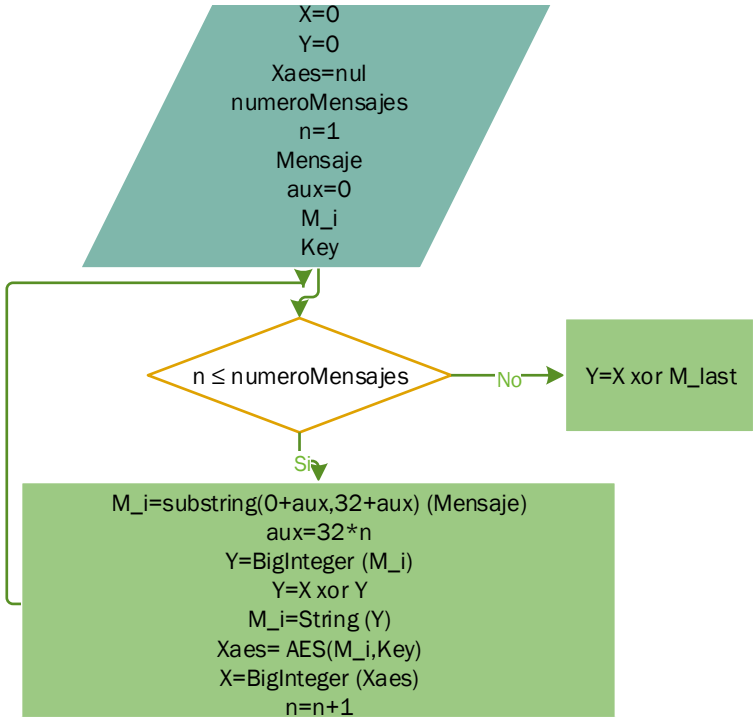


Figura 2.20. Iteración y partición de mensajes en CMAC.

Este mensaje deberá ser encriptado por AES-128, pero debido a la imperfección de la operación *XOR* de la clase "*BigInteger"* es necesario manejar los errores que se puedan presentar en las longitudes de los mensajes. Se verifica la longitud del último mensaje a encriptar y si sobrepasa en uno la longitud de 32 caracteres se eliminará ese valor inicial, el cual si sucede siempre será de un valor de uno. Si la longitud es menor en uno se añadirá un valor de cero al inicio de la cadena.

Ya encriptado el último mensaje, será considerado como el mensaje codificado y listo para ser retornado por el método CMAC. Este método tiene como atributo la cantidad de bits necesarios del mensaje codificado. Los bits tomados serán los más significativos o los del extremo derecho de la cadena resultante. Este valor al estar expresado en bits necesita ser dividido entre cuatro y será el valor final de la subcadena. Los valores típicos son 128, 64, 32 o 16.

c. Testigo o "*Witness*"

El testigo o "*Witness*" en el protocolo DAA se puede considerar como el elemento más importante de éste en términos de autenticación. Además de combinar todos los algoritmos de codificación y encripción descritos hasta el momento, permite autenticar el proceso y las entidades que lo componen. Para el testigo se necesitará una llave, un mensaje y la cantidad de bits de salida todo esto para que sean codificados a través de CMAC.

Los elementos del testigo para ser codificados se componen básicamente de una llave formada por un número randómico de 128 bits o 32 caracteres llamado *Nonce*; para crear este *Nonce* se utiliza la clase "*SecureRandom"*. Además de esto el mensaje a codificar con CMAC será la concatenación entre la dirección del Nodo, la dirección del Hub, la llave pública en coordenada X y la llave pública en coordenada Y, estas últimas creadas a partir de codificación con curva criptográfica elíptica. Para el testigo se necesita una salida de 128 bits.

Cabe recalcar que la única entidad que envía dentro de sus tramas de comunicación al testigo es el Nodo, tanto el original como el impersonalizado por el atacante. Este elemento debería garantizar que se eviten los ataques que se han descrito, pero como se menciona no lo hace por completo. Modificaciones a este testigo permitirán garantizar la autenticación en el proceso de Asociación objetivo de este estudio.

En el Anexo II se encuentra el código implementado para llevar a cabo todos los procesos y algoritmos descritos en esta sección, así como también los comentarios que lo describen.

2.2.5 IMPLEMENTACIÓN Y SIMULACIÓN DE ELEMENTOS DEL PROTOCOLO DAA

a. Codificación de tramas

Dentro de un paquete especial se concentrarán todas las funciones para unificar la información y elementos en las diferentes tramas. Los elementos son creados dentro del proceso de Asociación y se pasan como atributos a las clases específicas, para que se formen las cadenas de caracteres correspondientes a las tramas de Asociación.

Dentro de las diferentes tramas existen procesos en común que serán utilizados en estas tramas; entre estos métodos existen el cálculo de la dirección MAC y el cálculo del código de redundancia cíclico. La obtención de la dirección MAC emula la dirección EUI-48 necesaria en las tramas de Administración IACK y *Beacon*. El cálculo del código de redundancia cíclico será necesario para las tramas de Administración y *Beacon*, tal como se explica en el capítulo 1 de este trabajo.

a.1 Cálculo CRC

El método que contendrá el cálculo del CRC se llama *"CalculoCRC"*, este método aceptará como atributo de entrada un *String* el cual contendrá la información a la cual el método realizará el cálculo. Este método retorna un *String* de cuatro caracteres pertenecientes al CRC del mensaje de entrada.

Un código de redundancia cíclico puede ser calculado bajo operaciones de modificación de bits, pero siempre los CRC forman parte de un grupo de 256 distintos valores. Debido a esa condición se crea una tabla con los valores indicados en la Tabla 2.1. La tabla contendrá todos los valores posibles de CRC y será formada por un vector de enteros (*int*) llamado *table*. Después de creado ese vector, se debe transformar el *String* que corresponde al mensaje a ser calculado su CRC a un *array* de *bytes* llamado *bytes*, esto se consigue con el método *"getBytes"* de la clase "*String"*, es decir tendrá un elemento byte correspondiente a cada carácter del mensaje *String*.

Dentro de un lazo *for* el cual recorrerá toda la dimensión del *array y* con un elemento entero llamado *crc* inicialmente seteado en cero, se operará hasta conseguir el CRC del mensaje. A este entero crc es necesario que se le ejecute un desplazamiento de 8 bits a la derecha con inclusión de ceros, esta operación se representa con >>> 8. Una vez que el desplazamiento se haya ejecutado se realizará la operación *XOR* con un elemento de la tabla antes mencionada. El elemento a seleccionar será el que corresponda a la

posición representada por la operación *XOR* entre el elemento crc y el *byte* de la cadena transformada. Adicionalmente para que esta operación corresponda a una posición válida del vector se aplicará un *AND* con el valor 15 o FF en hexadecimal. Toda esta operación volverá a ser almacenada en la variable crc. Estas operaciones se repetirán hasta que todos los elementos del *array* de *bytes* se cumplan, es decir tantas veces como caracteres del mensaje estén.

Tabla 2.1. Valores posibles en la obtención de CRC.

0x0000	0xC0C1	0xC181	0x0140	0xC301	0x03C0	0x0280	0xC241	0xC601
0x0500	0xC5C1	0xC481	0x0440	0xCC01	0x0CC0	0x0D80	0xCD41	0x0F00
0xCAC1	0xCB81	0x0B40	0xC901	0x09C0	0x0880	0xC841	0xD801	0x18C0
0xDA81	0x1A40	0x1E00	0xDEC1	0xDF81	0x1F40	0xDD01	0x1DC0	0x1C80
0x1540	0xD701	0x17C0	0x1680	0xD641	0xD201	0x12C0	0x1380	0xD341
0xF001	0x30C0	0x3180	0xF141	0x3300	0xF3C1	0xF281	0x3240	0x3600
0x35C0	0x3480	0xF441	0x3C00	0xFCC1	0xFD81	0x3D40	0xFF01	0x3FC0
0x3B80	0xFB41	0x3900	0xF9C1	0xF881	0x3840	0x2800	0xE8C1	0xE981
0xEA41	0xEE01	0x2EC0	0x2F80	0xEF41	0x2D00	0xEDC1	0xEC81	0x2C40
0x2700	0xE7C1	0xE681	0x2640	0x2200	0xE2C1	0xE381	0x2340	0xE101
0x60C0	0x6180	0xA141	0x6300	0xA3C1	0xA281	0x6240	0x6600	0xA6C1
0x6480	0xA441	0x6C00	0xACC1	0xAD81	0x6D40	0xAF01	0x6FC0	0x6E80
0xAB41	0x6900	0xA9C1	0xA881	0x6840	0x7800	0xB8C1	0xB981	0x7940
0xBE01	0x7EC0	0x7F80	0xBF41	0x7D00	0xBDC1	0xBC81	0x7C40	0xB401
0xB7C1	0xB681	0x7640	0x7200	0xB2C1	0xB381	0x7340	0xB101	0x71C0
0x9181	0x5140	0x9301	0x53C0	0x5280	0x9241	0x9601	0x56C0	0x5780
0x5440	0x9C01	0x5CC0	0x5D80	0x9D41	0x5F00	0x9FC1	0x9E81	0x5E40
0x9901	0x59C0	0x5880	0x9841	0x8801	0x48C0	0x4980	0x8941	0x4B00
0x8EC1	0x8F81	0x4F40	0x8D01	0x4DC0	0x4C80	0x8C41	0x4400	0x84C1
0x4680	0x8641	0x8201	0x42C0	0x4380	0x8341	0x4100	0x81C1	0x8081
0x06C0	0x0780	0xC741	0x4040	0x21C0	0x2080	0xE041	0xA001	0x8581
0xCFC1	0xCE81	0x0E40	0x0A00	0xA781	0x6740	0xA501	0x65C0	0x4540
0x1980	0xD941	0x1B00	0xDBC1	0xAE41	0xAA01	0x6AC0	0x6B80	0x8701
0xDC41	0x1400	0xD4C1	0xD581	0xBB01	0x7BC0	0x7A80	0xBA41	0x47C0
0x1100	0xD1C1	0xD081	0x1040	0x74C0	0x7580	0xB541	0x7700	0x5B40
0xF6C1	0xF781	0x3740	0xF501	0x7080	0xB041	0x5000	0x90C1	0x4E00
0x3E80	0xFE41	0xFA01	0x3AC0	0x9741	0x5500	0x95C1	0x9481	0x8BC1
0x2940	0xEB01	0x2BC0	0x2A80	0x5A00	0x9AC1	0x9B81	0x8A81	0x4A40
0xE401	0x24C0	0x2580	0xE541					

Para retornar el valor entero que resulta de toda la operación se transforma a un *String* hexadecimal con el método *"toHexString"* de la clase "*Integer"*. Además, para tener la misma presentación de las tramas es necesario pasar todos los caracteres a mayúsculas con el método *"toUpperCase"* de la clase "*String"*. El proceso de obtención del código CRC se visualiza en la Figura 2.21, el cual explica el funcionamiento del código implementado que se encuentra en el Anexo II.

a.2 Cálculo MAC

Como se menciona, el cálculo de MAC emula la dirección EUI-48 que deberían tener los dispositivos con el estándar IEEE 802.15.6. Por ello es necesario obtener la dirección del dispositivo donde se está ejecutando las pruebas de vulnerabilidad para poder formar de manera correcta las diferentes tramas.

El método MAC no admite atributos de entrada, pero retorna un *String* de 12 caracteres, con la dirección de la interfaz de red que está involucrada en el traspaso de información para la Asociación.

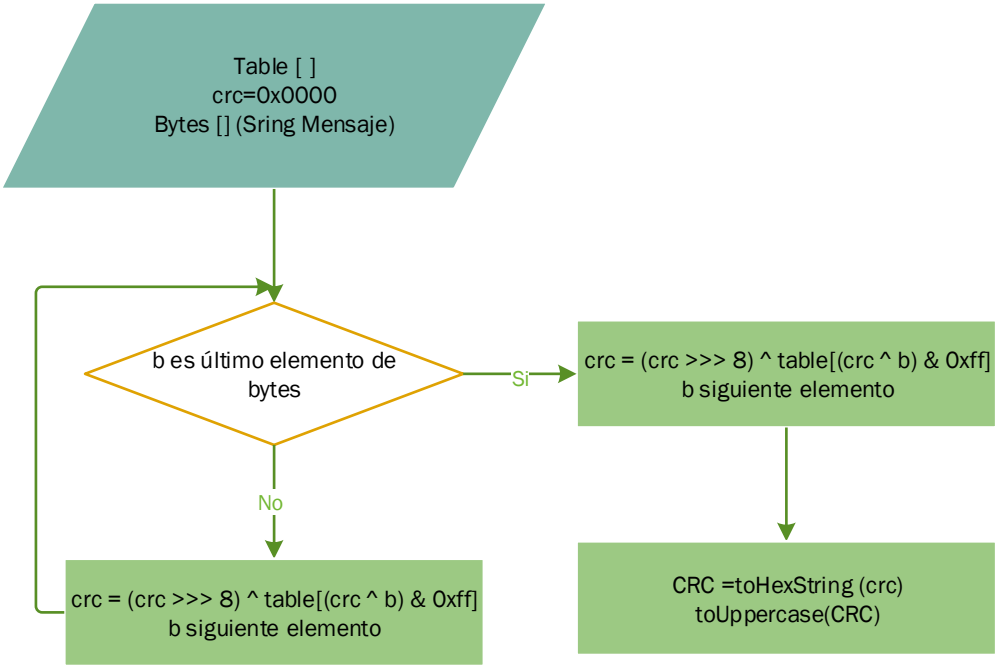


Figura 2.21. Proceso cálculo CRC.

Con la importación de paquetes *java.net* se tienen algunas clases y métodos útiles para la obtención de la dirección antes mencionada. Por ejemplo, para la obtención de la dirección IP de la máquina local donde se está ejecutando el proceso, se utiliza el método

"getLocalHost" de la clase *"InetAddress"* para obtener el nombre de la interfaz del dispositivo donde se está ejecutando el proceso, para posteriormente con ello crear un objeto "*NetworkInterface"* añadido el método *"getByInetAddress".* Este objeto contiene entonces toda la información de la interfaz de red antes mencionada; la información requerida podrá ser obtenida con diferentes métodos que la clase *"NetworkInterface"* contiene. Por ello para obtener la dirección física de esa interfaz se utiliza el método "*getHardwareAddress",* este método devolverá un *array* de bytes con la dirección.

Esta dirección contenida dentro del *array* de *bytes* no está en el formato deseado para ser insertada en las tramas, debido a esto es necesario la construcción de un *String* con la dirección. Con el método *append* de la clase *"StringBuilder"* se irán adjuntando los caracteres que sí correspondan a un carácter válido de la dirección MAC; para poder verificar todos los caracteres se irá recorriendo uno a uno los elementos del *array* de *bytes* que contiene la dirección. Al final ese *String* construido se retornará para continuar con el proceso, tal como se muestra en la Figura 2.22.

```
public class CalculoMac {
	public String Mac(){
		InetAddress ip;
		String Mac = "aa";
		try {

			ip = InetAddress.getLocalHost();
			NetworkInterface network = NetworkInterface.getByInetAddress(ip);

			byte[] mac = network.getHardwareAddress();
			StringBuilder sb = new StringBuilder();
			for (int i = 0; i < mac.length; i++) {
				sb.append(String.format("%02X%s", mac[i], (i < mac.length - 1) ? "" : ""));
			}
			Mac = sb.toString();

		} catch (UnknownHostException e) {
			e.printStackTrace();
		} catch (SocketException e) {
			e.printStackTrace();
		}
		return Mac;
```

Figura 2.22. Codificación implementada para cálculo MAC.

a.3 Tramas *Beacon*

La clase "*Beacon*" contendrá las operaciones, llamadas de métodos y concatenaciones necesarias para conformar la trama *Beacon* a ser enviada dentro del proceso. El método *Beacon1* no admite ningún valor como atributo y devuelve un *String* con la conformación de la trama *Beacon* detallada su estructura con minuciosidad en el capítulo 1.

El primer elemento que se crea es la cabecera de la trama, es decir se crea un *String* con la secuencia 01020000000102. Posteriormente esta cadena se concatena con la

dirección MAC, la cual el procedimiento se mencionó en el literal anterior. Estos dos elementos a su vez se concatenan con el *Frame Payload* es decir con la secuencia 00000181000080FFF400FF. Con todos estos elementos concatenados se crea un objeto para llamar al método "*CalculoCRC"* pasando como argumento todas las concatenaciones hasta ese momento. Por último, concatena una cadena adicional que es el CRC obtenido para ese respectivo mensaje, dando como resultado la trama *Beacon* que utilizará el Hub para el proceso de Asociación, tal como se muestra en la Figura 2.23.

```
public class Beacon {
    public String Beacon1(){
        CalculoMac MACTrama=new CalculoMac(); //llamo a Calculo MAC para la direccion de la trama
        String MAC=MACTrama.Mac();
        String BeaconTrama = "01020000000102";// Mac Header
        BeaconTrama = BeaconTrama.concat(MAC);
        String BeaconTrama1="00000181000080FFF400FF"; //Frame Payload
        BeaconTrama=BeaconTrama.concat(BeaconTrama1);
        String CalCRC = "0x" +BeaconTrama;
        CalculoCRC CRCTramaBeacon=new CalculoCRC();//CAlculo del CRC
        String CRC=new String();
        CRC=CRCTramaBeacon.Crc(CalCRC);
        BeaconTrama=BeaconTrama.concat(CRC); //TramaFinal
        return BeaconTrama;
    }
```

Figura 2.23. Implementación trama *Beacon*.

a.4 Primera Trama de Asociación

Para que se requiera la creación de la primera trama de Asociación se deben haber ejecutado los siguientes procedimientos:

- Creación de una llave privada.
- Cálculo de las coordenadas X e Y de la llave pública con el método de Curva Elíptica
- Creación de un elemento randómico de 128 bits llamado *Nonce*.
- Creación del testigo o *Witness.*

Por ello el método Asociacion1 admite como atributos de entrada cinco cadenas conformadas por:

- Dirección del Hub (tomada de la trama *Beacon*)
- Dirección del Nodo (utilizando el método *"CalculoMAC"*)
- *String* con la llave pública en X.

- *String* con la llave pública en Y.
- *String* con el testigo o *Witness*.

Se retorna toda la primera trama de Asociación como *String*, con los elementos necesarios en el orden correcto y preciso.

La cabecera de igual forma que en la trama *Beacon* se crea, pero para este caso la cabecera estará conformada por la secuencia 4142000002030200. Esta cabecera se concatena con el *String* "DireccionHub", con el *String* "DireccionNodo", estos últimos ingresados como atributos al método, y con la cadena 80000000; que finalmente se guardarán en un *String* llamado "MgnTrama".

Para la primera trama de Asociación el elemento MK no se calcula y en su lugar se coloca una cadena conformada por 16 ceros. El *String* "MgnTrama" se concatenará con el *String "Witness"* que contiene al testigo, el *String* "PkxA" que contiene la llave pública en X, el *String* "PkyA" que contiene la llave pública en Y, el *String* "MK" que contiene la cadena de 16 ceros; todo esto se guardará en un *String* "TramaAsociacion".

Todo este mensaje se pasará como argumento para que se calcule el CRC. Para esto como se lo realiza en la trama *Beacon*, se genera el objeto y posteriormente se realiza el llamado del método para el cálculo respectivo. Por último, al *String* "TramaAsociacion" se concatena el CRC calculado y se lo retorna para el uso dentro del proceso, como se muestra en la Figura 2.24.

```
public class PrimeraTramaAsociacion {
        public String Asociacion1(String RecipientAdd,String SenderAdd,
                String PKxA,String PKyA,String Witness_A ){
                CalculoMac MACTrama=new CalculoMac();
                String MAC=MACTrama.Mac();
                String MgnTrama = "4142000002030200"; /* Mac Header */
                MgnTrama = MgnTrama.concat(RecipientAdd).concat(SenderAdd).concat("80000000");
                String MK="0000000000000000";
                String TramaAsociacion=MgnTrama.concat(Witness_A).concat(PKxA).concat(PKyA).concat(MK);
                String CalCRC = "0x" +TramaAsociacion;
                String asociacion = CalCRC;
                CalculoCRC CRCTramaManagment=new CalculoCRC();
                String CRC=new String();
                CRC=CRCTramaManagment.Crc(asociacion);
                TramaAsociacion=TramaAsociacion.concat(CRC);
                return TramaAsociacion;
        }
```

Figura 2.24. Codificación de la Primera Trama de Asociación.

a.5 Segunda Trama de Asociación

Para que se requiera la creación de la segunda trama de Asociación en el Hub se deben haber ejecutado los siguientes procedimientos:

- Haber recibido la primera trama de Asociación por parte del Nodo
- Creación de una llave privada.
- Cálculo de las coordenadas X e Y de la llave pública con el método de Curva Elíptica
- Creación de un elemento randómico de 128 bits llamado *Nonce*.

Por ello el método Asociacion2 admite como atributos de entrada cuatro cadenas conformadas por:

- Trama de Asociación uno.
- *String* con la llave pública en X.
- *String* con la llave pública en Y.
- *String* con el *Nonce*.

Con la información antes descrita, el Hub realiza las siguientes acciones, con el objetivo de formar la segunda trama de Asociación.

- En el quinto carácter de la primera trama de Asociación se cambia el valor cero por el valor uno, indicando que la secuencia ha aumentado un valor por ser la segunda trama de Asociación.
- Se intercambian la dirección de *Recipient ID* con la de *Sender ID* y viceversa de la primera trama de Asociación. Estos elementos están en los caracteres 9,10,11,12 de la trama. En la segunda trama de Asociación se mantendría el valor de *BAN ID.*
- Se intercambian las Direcciones de Hub y Nodo con el objetivo de evitar que se necesite calcular nuevamente. Estas secuencias están ubicadas desde el carácter 17 al 28 para la dirección del Nodo y del 29 al 40 para la dirección del Hub.
- Reemplazar el valor cero que se tuvo en el carácter 45 de la primera trama de Asociación, por un valor uno.
- Guardar los elementos que no han sido modificados provenientes de la trama de Asociación uno, en el mismo orden.

Después de realizar estas acciones se dispone de 10 secuencias tanto de los elementos cambiados como de los que no lo han hecho. Para el caso de la segunda trama de

Asociación el elemento MK aún sigue sin ser calculado por ello también, mantendrá una cadena de 16 ceros.

Las 10 secuencias se irán concatenando una a una en el orden correcto. Adicionalmente, éstas se concatenarán con el *String "Nonce"*, el *String* "PkxB" que contiene la llave pública en X, el *String* "PkyB" que contiene la llave pública en Y, el *String* "MK" que contiene la cadena de 16 ceros, todo esto se guardará en un *String* "TramaAsociacion2".

Todo este mensaje se pasará como argumento para que se calcule el CRC. Para esto, como se lo realiza en la trama Asociación uno, se genera el objeto y posteriormente se realiza el llamado del método para el cálculo respectivo. Por último, al *String* "TramaAsociacion2" se concatena el CRC calculado y se lo retorna para el uso dentro del proceso, como se muestra en la Figura 2.25.

```
public class SegundaTramaAsociacion {
    public String Asociacion2(String TramaAsociacion_1,String PKxB, String PKyB,String NonceB ){
        String seq1=TramaAsociacion_1.substring(0,4);
        String seq2= TramaAsociacion_1.substring(4,6).replace("00","01");
        String seq3= TramaAsociacion_1.substring(6,16).replace("0002030200","0003020200");
        String seq4= TramaAsociacion_1.substring(16,28).replace(TramaAsociacion_1.substring(16,28),
                TramaAsociacion_1.substring(28,40));
        String seq5= TramaAsociacion_1.substring(28,40).replace(TramaAsociacion_1.substring(28,40),
                TramaAsociacion_1.substring(16,28));
        String seq6= TramaAsociacion_1.substring(40,44);
        String seq7= TramaAsociacion_1.substring(44,45).replace("0", "1");
        String seq8= TramaAsociacion_1.substring(45,48);
        String MK="0000000000000000";
        String seq9= NonceB;
        String seq10= PKxB;
        String seq11= PKyB;
        String seq12= MK;
        String TramaAsociacion2= seq1.concat(seq2).concat(seq3).concat(seq4).concat(seq5).concat(seq6).
                concat(seq7).concat(seq8).concat(seq9).concat(seq10).concat(seq11).concat(seq12);
        String CalCRC = "0x" +TramaAsociacion2;
        String asociacion = CalCRC;
        CalculoCRC CRCTramaManagment=new CalculoCRC();
        String CRC=new String();
        CRC=CRCTramaManagment.Crc(asociacion);
        TramaAsociacion2=TramaAsociacion2.concat(CRC);
        return TramaAsociacion2;
    }
}
```

Figura 2.25. Codificación de la Segunda Trama de Asociación.

a.6 Tercera Trama de Asociación

Para que se requiera la creación de la tercera trama de Asociación en el Hub se deben haber ejecutado los siguientes procedimientos:

- Haber enviado como Hub la segunda trama de Asociación.
- Cálculo de la DHKey
- Cálculo de la llave Temp_1

- Cálculo de MK_KMAC_3B

Por ello el método “Asociacion3” admite como atributos de entrada dos cadenas conformadas por:

- Trama de Asociación dos
- MK_KMAC_3B

Con la información antes descrita el Hub realiza las siguientes acciones, con el objetivo de formar la tercera trama de Asociación.

- En el quinto carácter de la segunda trama de Asociación se cambia el valor uno por el valor dos, indicando que la secuencia ha aumentado un valor por ser la tercera trama de Asociación.
- Reemplazar el valor uno que se tuvo en el carácter 45 de la segunda trama de Asociación, por un valor dos.

Después de realizar estas acciones se dispone de 10 secuencias tanto de los elementos cambiados como de los que no lo han hecho. Para el caso de la tercera trama de Asociación el elemento MK se acepta como atributo del método.

Las 10 secuencias se irán concatenando una a una en el orden correcto. Adicionalmente éstas se concatenarán con el *String “Nonce”*, el *String* “PkxB” que contiene la llave pública en X, el *String* “PkyB” que contiene la llave pública en Y, estas últimas extraídas de la cadena Asociacion2, y el *String* “MK_3B” ingresado como atributo; todo esto se guardará en un *String* “TramaAsociacion3”.

Todo este mensaje se pasará como argumento para que se calcule el CRC. Para esto como se lo realiza en la trama Asociación dos, se genera el objeto y posteriormente se realiza el llamado del método para el cálculo respectivo. Por último, al *String* “TramaAsociacion3” se concatena el CRC calculado y se lo retorna para el uso dentro del proceso, como se muestra en la Figura 2.26.

a.7 Cuarta trama de Asociación

Para que se requiera la creación de la cuarta trama de Asociación en el Nodo se deben haber ejecutado los siguientes procedimientos:

- Haber recibido del Hub la tercera trama de Asociación.
- Cálculo de la DHKey

- Cálculo de la llave Temp_1.
- Cálculo de MK_KMAC_3A
- Que MK_KMAC_3A sea igual que MK_KMAC_3B
- Cálculo de MK_KMAC_4ª

```
public class TerceraTramaAsociacion {
    public String Asociacion3(String TramaAsociacion_2,String MK_3B ){
        String seq1=TramaAsociacion_2.substring(0,4);
        String seq2= TramaAsociacion_2.substring(4,6).replace("01","02");
        String seq3= TramaAsociacion_2.substring(6,44);
        String seq4= TramaAsociacion_2.substring(44,45).replace("1", "2");
        String seq5= TramaAsociacion_2.substring(45,48);
        String NonceB=TramaAsociacion_2.substring(48,80);
        String PKxB=TramaAsociacion_2.substring(80,144);
        String PKyB=TramaAsociacion_2.substring(144,208);
        String seq6= NonceB;
        String seq7= PKxB;
        String seq8= PKyB;
        String seq9= MK_3B;
        String TramaAsociacion3= seq1.concat(seq2).concat(seq3).concat(seq4).concat(seq5).
                concat(seq6).concat(seq7).concat(seq8).concat(seq9);
        String CalCRC = "0x" +TramaAsociacion3;
        String asociacion = CalCRC;
        CalculoCRC CRCTramaManagment=new CalculoCRC();
        String CRC=new String();
        CRC=CRCTramaManagment.Crc(asociacion);
        TramaAsociacion3=TramaAsociacion3.concat(CRC);
        return TramaAsociacion3;
    }
```

Figura 2.26. Codificación de la Tercera Trama de Asociación.

Por ello el método Asociacion4 admite como atributos de entrada tres cadenas conformadas por:

- Trama de Asociación uno
- MK_KMAC_4A
- *Nonce*_A

Con la información antes descrita el Nodo realiza las siguientes acciones, con el objetivo de formar la cuarta trama de Asociación.

- En el quinto carácter de la primera trama de Asociación se cambia el valor cero por el valor tres, indicando que la secuencia ha aumentado un valor por ser la cuarta trama de Asociación.

- Reemplazar el valor cero que se tuvo en el carácter 45 de la primera trama de Asociación, por un valor tres.

Después de realizar estas acciones se dispone de 10 secuencias tanto de los elementos cambiados como de los que no lo han hecho. Para el caso de la cuarta trama de Asociación el elemento MK se acepta como atributo del método.

Las 10 secuencias se irán concatenando una a una en el orden correcto. Adicionalmente éstas se concatenarán con el *String "Nonce"*, el *String* "PkxB" que contiene la llave pública en X, el *String* "PkyB" que contiene la llave pública en Y, estas últimas extraídas de la cadena Asociacion1, y el *String* "MK_4A" ingresado como atributo; todo esto se guardará en un *String* "TramaAsociacion4". En la cuarta trama de Asociación el valor donde irá el *Nonce*, es ocupado por el testigo en la primera trama, permitiendo algún tipo de vulnerabilidad en este punto.

```
public class CuartaTramaAsociacion {
    public String Asociacion4(String TramaAsociacion_1,String Nonce_A,String MK_4A ){
        String seq1=TramaAsociacion_1.substring(0,4);
        String seq2= TramaAsociacion_1.substring(4,6).replace("00","03");
        String seq3= TramaAsociacion_1.substring(6,44);
        String seq4= TramaAsociacion_1.substring(44,45).replace("0", "3");
        String seq5= TramaAsociacion_1.substring(45,48);
        String PKxA=TramaAsociacion_1.substring(80,144);
        String PKyA=TramaAsociacion_1.substring(144,208);
        String seq6= Nonce_A;
        String seq7= PKxA;
        String seq8= PKyA;
        String seq9= MK_4A;
        String TramaAsociacion4= seq1.concat(seq2).concat(seq3).concat(seq4).concat(seq5).
                concat(seq6).concat(seq7).concat(seq8).concat(seq9);
        String CalCRC = "0x" +TramaAsociacion4;
        String asociacion = CalCRC;
        CalculoCRC CRCTramaManagment=new CalculoCRC();
        String CRC=new String();
        CRC=CRCTramaManagment.Crc(asociacion);
        TramaAsociacion4=TramaAsociacion4.concat(CRC);
        return TramaAsociacion4;
    }
```

Figura 2.27. Codificación de la Cuarta Trama de Asociación.

Todo este mensaje se pasará como argumento para que se calcule el CRC. Para esto como se lo realiza en la trama Asociación dos, se genera el objeto y posteriormente se realiza el llamado del método para el cálculo respectivo. Por último, al *String* "TramaAsociacion4" se concatena el CRC calculado y se lo retorna para el uso dentro del proceso, como se muestra en la Figura 2.27.

a.8 Tramas IACK

La trama IACK, se envía justo después de una trama de Asociación. Esta trama dispone de *payload,* solo cuenta con su cabecera y posteriormente un valor de CRC. La diferencia entre cada trama IACK únicamente serán los identificadores de la cabecera y los valores que representan una secuencia, por ello:

- En la primera trama a la secuencia 4007000003020200, se calcula el CRC y se concatena.
- A la segunda IACK se reemplazará por la secuencia 4007010002030200 concatenada al CRC que resulte de este mensaje.
- La tercera IACK tendrá la secuencia 4007020002030200 concatenada con el CRC.
- La cuarta y última trama IACK se concatena el CRC a la secuencia 4007040003020200.

Para la segunda y cuarta trama IACK, se colocará en el método el atributo IACK1, mientras que para la tercera trama IACK3 se colocará como atributo la segunda trama IACK.

b. Estructuración del proceso DAA

La estructuración del proceso DAA corresponde a una combinación de todos los procesos anteriores, es decir mezcla procesos como CMAC, Criptografía de Curva Elíptica, formación de tramas, entre otros. La formación del DAA existe para las entidades Nodo y Hub, necesitando una coordinación entre cada proceso para que el protocolo de Asociación se cumpla de manera efectiva.

El objetivo del proceso DAA es activar la llave MK entre las dos entidades; si y solo si las dos entidades han activado la MK y el valor decimal mostrado es el mismo, se puede decir que la Asociación entre las dos entidades se ha hecho efectiva, tal como se lo explica en el capítulo 1 de este trabajo. Esta sección busca detallar de manera metodológica los algoritmos y mecanismos necesarios para que se ejecute el proceso de Asociación entre las dos entidades.

Además, los desarrollos de los diferentes procesos para cada entidad no han sido modificados en base a lo descrito en el capítulo 1, de tal forma que las vulnerabilidades encontradas y detalladas en la misma sección se puedan evaluar. La formación de cada proceso conlleva una serie de condicionales y estructuras de control con el objetivo de ir

cumpliendo a cabalidad cada parte del proceso. Por ello se puede considerar que el proceso es semicontrolado, debido a que de forma parcial se puede controlar la asociación entre las entidades y posteriormente el tipo de ataque que se ejecutará.

Los procesos se estructuran por separado para cada entidad, pero mantienen siempre relación directa en función de cumplir el proceso dentro del escenario de comunicaciones.

b.1 Proceso Hub

El Hub es el encargado de ejecutar la primera acción para que el protocolo de Asociación se ejecute; el Hub envía de manera repetitiva y cíclica tramas *Beacon* hasta que halle un Nodo que quiera asociarse. El envío de tramas *Beacon* se realiza cada tres segundos, esto último con la clase *"Thread"* y el método *"sleep",* emulando los *slots* de tiempo como se indica en el capítulo 1. Para la creación de tramas se hace el llamado del método *Beacon*1 de la clase "*Beacon*" que ensambla la trama a ser enviada. El Hub está diseñado para enviar tantas tramas como sean necesarias hasta que reciba un mensaje de Asociación uno, por parte de una entidad Nodo. El identificador con el que la trama *Beacon* se envía es "H1 BB:" acompañada del *String* con los caracteres de la trama *Beacon*.

Debido a que en el escenario de comunicaciones el servidor replica cada mensaje enviado hacia todos los elementos conectados de esa red, hay que discriminar los mensajes que lleguen por parte del Hub. La discriminación se realiza por medio de etiquetas o identificadores de mensaje. Por esto, cada trama *Beacon* se enviará por ejemplo con el identificador "H1 BB:". No es necesario para el Hub imprimir las tramas *Beacon* que está enviando, debido a ello toda trama que empiece con este indicador no será impresa dentro de la interfaz gráfica del Hub. La extracción de cada identificador únicamente se realizará con el método "*startsWith"*

b.1.1 Fase de envío de tramas *Beacon*

Para que el Hub decida terminar con el envío de tramas *Beacon,* existe un elemento *booleano* que se activa, esto es, se coloca en *true* cuando una trama de Asociación uno ha llegado, es decir que contenga dentro de su etiqueta de identificación "A1". Adicionalmente, una vez obtenida la primera trama de Asociación, el Hub guarda la etiqueta de la entidad con la cual se va asociar para evitar que otros identificadores se tomen y detengan el proceso de Asociación. La única opción para que el Hub interrumpa el proceso de Asociación y lo recorra, es que un nuevo Nodo envíe una trama de Asociación uno. La reejecución del protocolo de Asociación, como se explica en las referencias teóricas de este trabajo es posible si antes de activar una *Master Key* (MK) un

Nodo con nuevas credenciales envía una nueva primera trama de Asociación que reinicie el proceso de Asociación.

La reejecución del protocolo de Asociación, será especialmente útil cuando se desee realizar impersonalizaciones o ataques dentro del proceso de Asociación.

Una vez recibida por parte del Hub la primera trama de Asociación, se continua con el proceso de Asociación y todos estos procedimientos quedan anulados y no serán tomados en cuenta por parte de la ejecución del programa. La Fase inicial y envío de tramas *Beacon* se aprecia en la Figura 2.28, cuyo algoritmo representa el código implementado para este fin, detallado en el Anexo II.

Una vez recibida la primera trama de Asociación, el controlador de envío de tramas *Beacon* se desactiva, es decir el dato *booleano* que permite el ingreso a la estructura de control y envío de las tramas *Beacon* en un lapso de 3 segundos se desactiva. El Hub para evitar asociarse con alguna entidad que no sea la que activó el proceso, toma la identidad y lo comprueba siempre que vaya a entrar al proceso como tal. A pesar de que la comprobación se realiza enseguida de la llegada de la primera trama de Asociación, puede darse el caso en el que llegue otra trama proveniente de un Nodo que no sea la de asociación uno u otra trama con identificador diferente, siendo necesario siempre realizar la comprobación. Por ello para que la estructura de control permita el ingreso al cálculo de credenciales para el proceso de Asociación, se verifica que el mensaje haya sido enviado por un Nodo y tenga el identificador que accionó el proceso de Asociación.

b.1.2 Fase de creación de elementos de Asociación

Ya dentro del proceso y continuando con la secuencia, el Nodo verifica si el mensaje identificador de trama sea A1, para eliminar el identificador de mensaje e identificador de trama dejando únicamente la trama como tal. A continuación, el Hub debe crear la dirección con la que se comunicará, esto con el método *“CalculoMAC”.* Otra opción para obtener la dirección con la que el Hub está intentado asociarse es guardar la trama *Beacon* que se envía periódicamente y extraer la dirección de allí. De la primera trama de Asociación se extrae la dirección MAC enviada dentro de la primera trama de Asociación. La dirección del Nodo se encuentra desde el carácter 29 al carácter 40 de la primera trama de Asociación. Toda esta extracción se realiza con el método *“subString”* de la clase *“String”*.

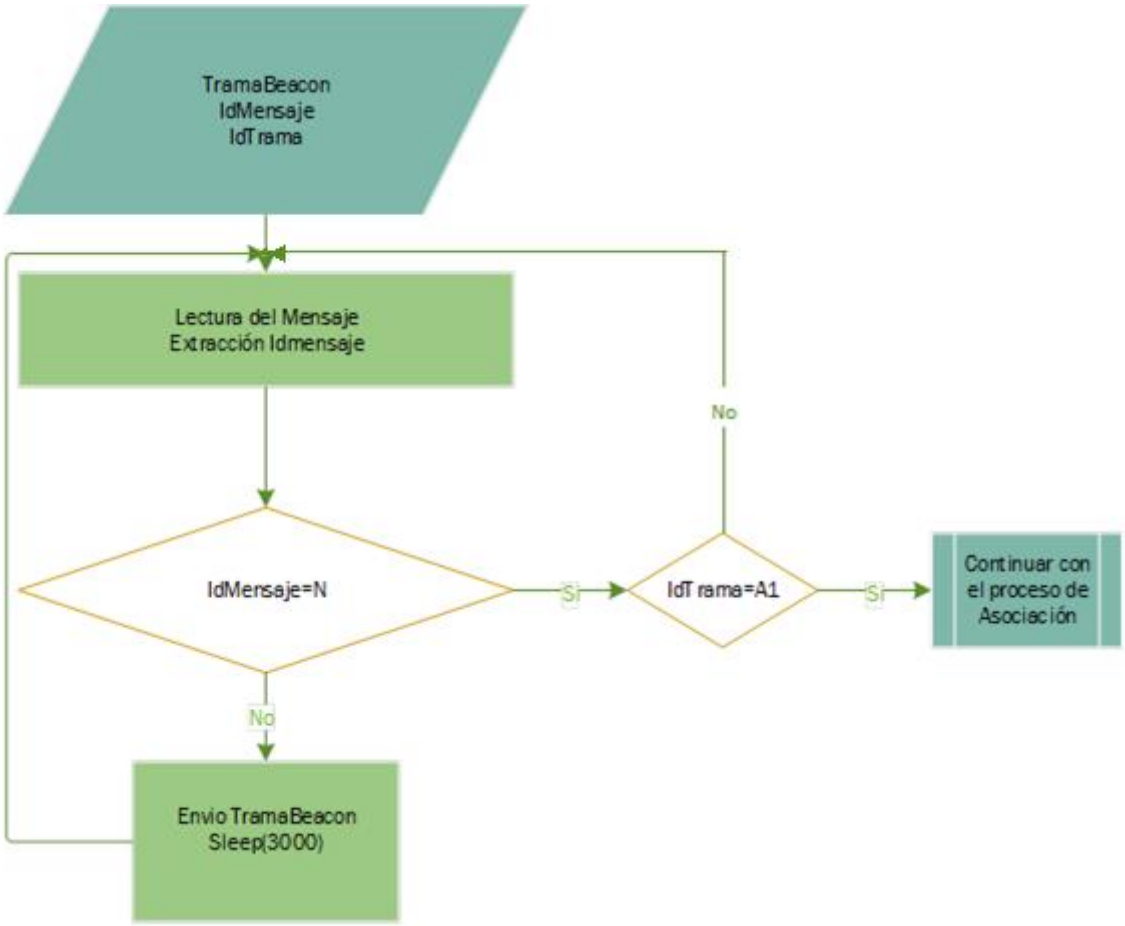


Figura 2.28. Fase inicial proceso de Asociación y tramas *Beacon.*

El Hub apenas extrae la dirección del Nodo, llama al método IACK de la clase "PrimerIack" procediendo a enviar el *String* generado por ese método. El envío de la primera trama IACK por parte del Hub indica al Nodo que se ha recibido de manera correcta la primera trama de Asociación; esto pone al Nodo en espera de la segunda trama de Asociación. La primera trama IACK enviada por el Hub, tendrá el identificador "H1 I1:" acompañado del *String* correspondiente a la trama IACK.

Uno de los primeros elementos que debe crear el Hub para la Asociación, es la llave secreta la cual será utilizada para criptografía de curva elíptica. Dentro de la clase *"StandardCurve"* la cual permite especificar dentro del método constructor qué tipo de curva estandarizada se utilizará, se establece la curva "P-256". Dentro de esta clase se encuentra el método "*getRandomSK".* Este método devuelve un valor *BigInteger* con una clave pseudo randómica segura de 256 bits o 64 caracteres. Para guardar en formato *String* hexadecimal, con el método *format* de la clase *"String*" se coloca el tipo de formato "%x". Adicionalmente se requiere estandarizar la presentación de las credenciales que se

van obteniendo. En función de esto, todos los valores deberán ser presentados y guardados en mayúsculas, esto se lo realiza con el método *"toUpperCase".*

La llave secreta formará parte del proceso de creación de las llaves públicas usando criptografía de Curva Elíptica. Esta llave únicamente no debe ser guardada en forma de *String* para ser impresa dentro de la interfaz gráfica, deberá conservar su valor *BigInteger* permitiendo hacer el llamado del método que crea las llaves públicas.

El método *"getPublicKey"* de la clase *"StandardCurve"* acepta como parámetro un *BigInteger* conformado por la llave privada creada con anterioridad. Este método retorna un *array* de *BigInteger* de dos elementos, uno para la llave pública en coordenada X y otro con la llave pública en coordenada Y. Para poder presentar en la interfaz gráfica, las llaves públicas obtienen formato hexadecimal *String*. El método de generación puede producir claves de una longitud menor a 32 caracteres, siendo necesario controlar este tipo de casos. Para controlarlo, se debe añadir un cero al inicio si la longitud es de 31 caracteres. Esto no implica que el método no funcione, corresponde más al hecho de que la transformación a hexadecimal desde un *BigInteger* provoca estas situaciones. El proceso de almacenamiento e impresión en la interfaz gráfica se debe realizar por cada elemento de la llave pública (X e Y).

Otro elemento importante para la creación de la segunda trama de Asociación es el *Nonce*. El *Nonce* al igual que lo que se realiza dentro del método "*getRandomSK",* para la creación, utiliza la clase *"SecureRandom"*. Con la posterior creación de un elemento *BigInteger* asociado con la clase mencionada, dentro de la creación se especifica la cantidad de bits que tendrá ese elemento randómico. Este valor deberá ser guardado como *String* para poder hacer la impresión dentro de la interfaz gráfica.

La trama de Asociación dos podrá ser creada una vez calculados y obtenido los siguientes elementos:

- Llave privada Sk
- Cálculo de las coordenadas X e Y de la llave pública con el método de Curva Elíptica Pkx y Pky.
- Creación de un elemento randómico de 128 bits, *Nonce*.

El Hub debe hacer la creación de un objeto de la clase "SegundaTramaAsociacion" y con el llamado del método "Asociacion2" se ensamblará la trama a enviar al servidor. El envío de esta trama se realizará con la etiqueta "H1 A2:", cerrando así la estructura de control

que se inició cuando se recibió la trama de Asociación uno, tal como se visualiza en el proceso de la Figura 2.29, cuyo código implementado se encuentra en el Anexo II.

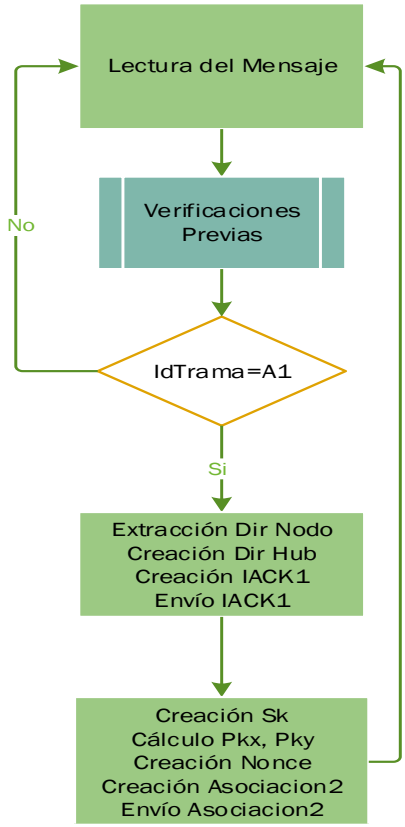


Figura 2.29. Fase de creación de elementos de Asociación.

Entregada la segunda trama de Asociación, el Hub queda en expectativa de recibir el segundo IACK por parte del Nodo para continuar con el procedimiento de Asociación. Después de recibir la segunda trama IACK, el Hub entra en fase de creación de elementos de autenticación, para la posterior verificación de los mismos.

b.1.3 Fase de creación de elementos de autenticación

Una vez recibido el IACK2 éste se guardará para su uso en posteriores tramas IACK. Además, el siguiente paso del proceso de Asociación corresponde a la creación de la llave *Diffie-Hellman* (DHKey).

La llave *Diffie-Hellman* es aquella que resulta de la multiplicación de la llave pública del Nodo, obtenida de la trama de Asociación uno, con la llave privada del Hub. Las llaves se extraen como subcadenas del carácter 81 al 144 para la llave pública X y del elemento 145 al 208 para la llave pública en Y. Todo esto con el objetivo que entre las dos entidades se comparta la misma llave *Diffie-Hellman*.

Para realizar esta multiplicación dentro del campo de la curva criptográfica elíptica es necesario utilizar una curva diferente a la correspondiente P-256. Esta nueva curva tendrá como parámetros los mismos valores a, b, p de la curva P-256, con la diferencia que los puntos iniciales Gx y Gy corresponderán a las coordenadas de la llave pública. Todo esto se puede realizar debido a las propiedades de curva criptográfica elíptica, donde la multiplicación de un escalar por un punto de la curva elíptica retorna otro punto dentro de ésta. Para realizar la creación de esta curva se utiliza la clase *"MCurve",* donde el método constructor permite ingresar los 5 parámetros mencionados.

Una vez creada la curva se debe obtener una nueva llave pública resultante de la llave privada del Hub. Esto se hace de igual manera con el método "*getPublicKey".* Este método acepta como atributo la llave secreta del Hub y entregará un nuevo punto perteneciente a la curva elíptica. Dentro del proceso se determina que la llave *Diffie-Hellman* corresponderá a la coordenada X que devuelve el método. Para extraer la coordenada X se utiliza el valor cero del *array BigInteger* que retorna del método "*getPublicKey".* Esta llave *Diffie-Hellman* se guardará en formato *String* para poder ser impresa dentro de la interfaz gráfica.

El objetivo de la creación de la llave *Diffie-Hellman* es la transformación de esta llave en una llave temporal uno para los procesos CMAC que prosiguen a éstos. Para obtener esta llave temporal uno, se toman los primeros 128 bits o 32 caracteres de la llave *Diffie-Hellman*. Como se ha indicado a lo largo de este trabajo la obtención de una subcadena se realiza con el método "*subString"* de la clase *"String".*

La llave temporal uno servirá para el cálculo de los elementos MK_KMAC_3B y el MK_KMAC_4B. El cálculo de estos elementos corresponde a un procedimiento CMAC con llave Temp1, mensaje KMAC y salida de 64 bits. El mensaje por codificar en MK_KMAC_3B será la concatenación de:

- Dirección del Nodo
- Dirección del Hub
- *Witness* del Nodo
- *Nonce* del Hub
- *Security Suite Selector* de las tramas de Asociación

El mensaje por codificar para el elemento KMAC_4B es muy similar a lo que requiere el mensaje de KMAC_3B con la diferencia que el orden del mensaje es:

- Dirección del Hub
- Dirección del Nodo
- *Nonce* del Hub
- *Witness* del Nodo
- *Security Suite Selector* de las tramas de Asociación.

Cabe recalcar que los elementos pertenecientes al Nodo para los mensajes a codificar se extraen de la trama de Asociación uno. Por ejemplo, la dirección del Nodo se guarda una vez recibida la trama de Asociación uno. El *Witness* se extrae igualmente de la trama de Asociación uno, con el método *"subString"* desde el carácter 49 al 80. El *Security Suite Selector* se lo puede extraer tanto de la trama de Asociación uno, como de la trama de Asociación dos desde el carácter 41 al carácter 44.

Para la creación de ambos elementos se crea un objeto de la clase *"CMAC"* y se colocan los parámetros antes mencionados dentro del método *"CalcCMAC"*. Las variables *String* que se crean esperarán una cadena de longitud 16 caracteres.

A pesar de que se calculan ambos elementos, el único que será enviado en la tercera trama de Asociación es el elemento MK_KMAC_3B. Para realizar el envío de la tercera trama de Asociación se crea un objeto del tipo "TerceraTramaAsociacion" y con el método "Asociacion3" pasando como atributo la primera trama de Asociación y el nuevo elemento MK_KMAC3B se ensamblará la tercera trama de Asociación. El envío de la tercera trama de Asociación se realizará con la etiqueta "H1 A3:", cerrando además la estructura de control que se activó cuando se recibió el segundo IACK. La creación de credenciales de autenticación y su proceso se lo visualiza en la Figura 2.30, el código que implementa dicho proceso se encuentra en el Anexo II.

Cuando el Hub ha enviado la tercera trama de Asociación estará a la espera de recibir dos tramas, la trama IACK 3 y la cuarta trama de Asociación. Al recibir la última, el Hub comenzará el proceso de autenticación de credenciales, pero primero enviará una respuesta al Nodo con el objetivo de que éste también comience el mismo proceso.

b.1.4 Fase de autenticación

Para comenzar con el proceso de autenticación de la trama de Asociación cuatro, se extrae el *Nonce* que envía el Nodo y el MK_KMAC_4A, este último extraído desde el carácter 209 al 224. Por parte del Hub deberá con el *Nonce* extraído desde el carácter 49

hasta el 80 de la cuarta trama de Asociación, crear el *Witness* con las mismas credenciales que el Nodo debió haberlo realizado.

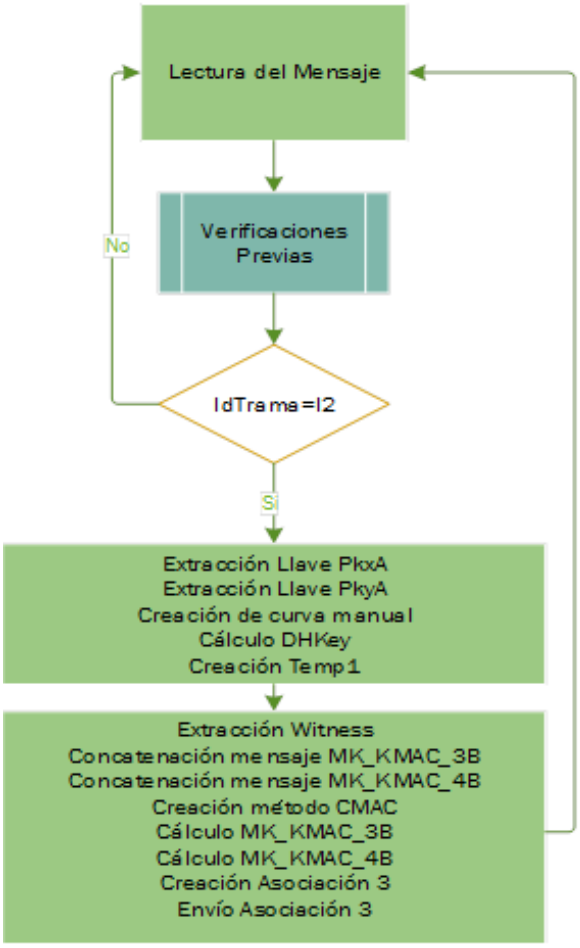


Figura 2.30. Creación de credenciales de Autenticación.

Para el cálculo del *Witness*, el Hub deberá codificar el mensaje obtenido de la concatenación de:

- Dirección Nodo
- Dirección Hub
- PKx Nodo
- PKy Nodo

Cuando el Hub ha calculado estos elementos de autenticación, debe realizar el proceso para verificar si las entidades pueden asociarse. El Hub hará dos comprobaciones

seguidas. El Hub deberá comprobar si el MK_KMAC_4B es igual al MK_KMAC_4A; si estos dos no son iguales el procedimiento se aborta hasta que se ejecuta un nuevo proceso. Se realiza dentro de una estructura de control *if* recordando que para comparar cadenas se utiliza el método "*equals"*. La segunda verificación se realiza sobre el elemento *Witness*; si el *Witness* calculado, con el extraído de la cuarta trama de Asociación, no son iguales se aborta el proceso.

El último proceso de verificación que se realiza para este protocolo de Asociación es la comparación de los números D mostrados en pantalla. El cálculo de este elemento se da a partir de la codificación con CMAC utilizando una concatenación entre el *Nonce* del Nodo y el *Nonce* del *Hub* como llave; además el mensaje será una concatenación del *Nonce* del *Hub, Nonce* del Nodo más la llave temporal uno. Todo esto con una salida de 16 bits, esta salida será transformada en el número decimal que será presentado. La codificación se realiza con el método "*CalcCMAC"* de la misma forma que el utilizado en el cálculo de los MK_KMAC. En la concatenación de las llaves *Nonce* A y *Nonce* B radica el principal motivo por el cual AES-128 debía permitir llaves de 256 bits como se indicó en la descripción del método AES.

Esta transformación de bits a decimal y la presentación de este número en la interfaz gráfica da el nombre a este proceso de Asociación. Si se verifica que lo números mostrados en pantalla son los mismos se puede proseguir a asociar a las entidades.

La asociación de las entidades se da cuando se calcula la llave maestra MK; la creación de esta llave maestra se da con una codificación CMAC nuevamente. En este caso la codificación se la realizará con una llave conformada por 128 bits o 32 caracteres últimos de la llave *Diffie-Hellman* llamada Temp_2. El mensaje a codificar será la concatenación del *Nonce* del Nodo y el *Nonce* del Hub con una salida de 128 bits. Todo esto realizado por el método "*CalcCMAC"*. La autenticación se da en base al proceso de la Figura 2.31, implementado en codificación Java como se indica en el Anexo II.

b.2 Proceso Nodo

b.2.1 Fase inicial, recibimiento de tramas *Beacon*

El Nodo es el encargado de empezar o iniciar el protocolo de Asociación; el Hub envía de manera repetitiva y cíclica tramas *Beacon* hasta que halle un Nodo que quiera asociarse. El envío de tramas *Beacon* se realiza cada tres segundos, siendo necesario programar al Nodo para que en una cierta cantidad de mensajes recibidos decida asociarse.

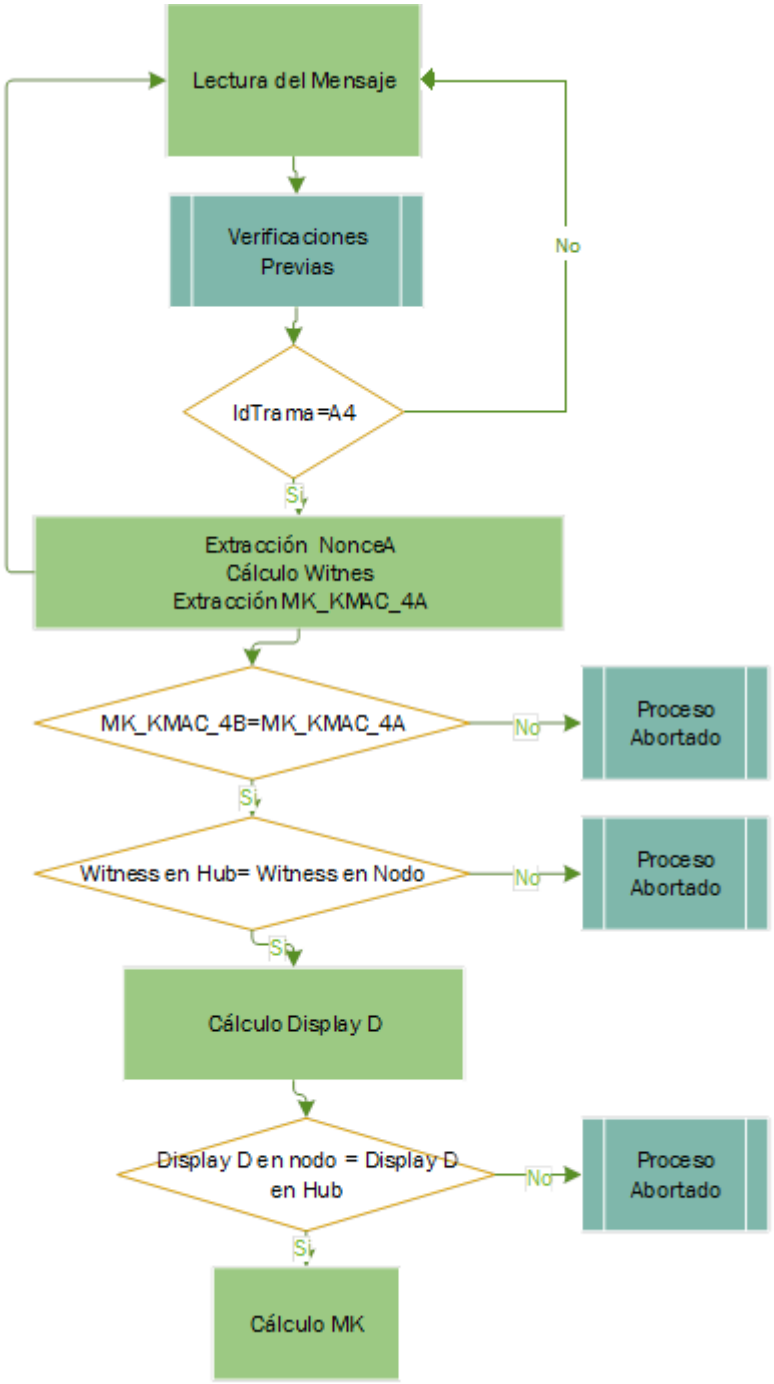


Figura 2.31. Fases de Autenticación y Asociación.

Cada que llegue un mensaje con el Identificador de trama “BB” el Nodo aumentará el valor del contador. El valor determinado puede ser cambiado, pero se ha colocado en un valor referencial de cuatro. Este valor de cuatro otorga el tiempo necesario para poder comprobar los elementos que van llegando sin tener el problema de una asociación tan

rápida. Además, este valor es especialmente útil cuando sea necesario probar las vulnerabilidades, dándole tiempo al atacante para que ejecute el ataque correspondiente.

Cuando el Nodo recibe la cuarta trama *Beacon*, según lo configurado para este proceso, un *booleano* llamado asociación se coloca en verdadero, indicando que desea iniciar un proceso de Asociación con el Nodo. Adicionalmente a eso se toma el identificador de mensaje con el cual se activó el proceso de Asociación. Antes de empezar a crear los elementos de la primera trama de Asociación el Nodo siempre comprueba que el elemento que activó la Asociación sea el que envió el último mensaje.

En el Nodo existe la necesidad de ignorar los mensajes de *Beacon* que llegan a pesar de que sea del mismo elemento que activó el proceso de Asociación. La diferencia de tiempos que existe entre que un Hub siga enviando tramas *Beacon* y que el Nodo realice la creación de la primera trama de Asociación y la envíe al Hub provoca esta necesidad. Recordando que el Hub deja de emitir tramas *Beacon* cuando recibe la primera trama de Asociación. Debido a todo esto si el Nodo ya identificó que el proceso de Asociación se ha iniciado y recibe nuevamente una trama *Beacon*, este mensaje se colocará con caracteres vacíos.

De la misma forma que el Hub, uno de los primeros elementos que debe crear para la asociación es la llave secreta, la cual será utilizada para criptografía de curva elíptica. Dentro de la clase *"StandardCurve"*, igualmente se utiliza la curva "P-256" en el constructor de la clase, y dentro de ésta se encuentra el método *"getRandomSK".* Este método devuelve un valor *BigInteger* con una clave pseudo randómica segura de 256 bits o 64 caracteres; estos valores se deberán guardar en formato *String* y deberán ser impresos en la interfaz gráfica. El procedimiento inicial del Nodo se visualiza en la Figura 2.32, y su codificación implementada se encuentra en el Anexo II.

b.2.2 Fase de creación de elementos de autenticación para Asociación

Una vez que el Nodo activó el proceso para ingresar al proceso de Asociación, verifica que el identificador de trama con el que llega el mensaje sea el que activó el proceso. Ya dentro del proceso de Asociación verifica nuevamente si el último mensaje corresponde a una trama *Beacon*, si es así comienza con la creación de la primera trama de Asociación.

A continuación, el Nodo debe crear la dirección con la que se comunicará. Esto se realizará con el método "*CalculoMAC".* De la trama *Beacon* se extrae la dirección MAC del Hub. La dirección del Hub se encuentra desde el carácter 29 al carácter 40 de la trama *Beacon*. Toda esta extracción se realiza con el método "*subString"* de la clase *String*.

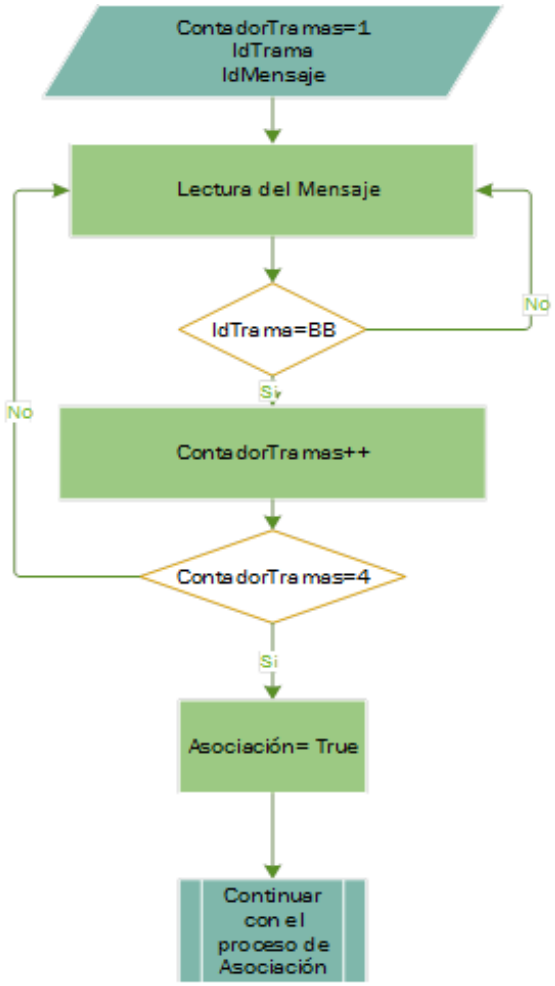


Figura 2.32. Fase inicial, recibimiento de *Beacon.*

El método "*getPublicKey"* de la clase "*StandardCurve"* acepta como parámetro un *BigInteger* conformado por la llave privada creada con anterioridad. Este método retorna un *array* de *BigInteger* de dos elementos, uno para la llave pública en coordenada X y otro con la llave pública en coordenada Y dentro del Nodo. Para poder presentar en la interfaz gráfica, las llaves públicas obtienen formato hexadecimal *String*. El método de generación puede producir claves de una longitud menor a 32 caracteres, siendo necesario controlar este tipo de casos. Para controlarlo, se debe añadir un cero al inicio si la longitud es de 31 caracteres. Esto no implica que el método no funcione, corresponde más al hecho de que la transformación a hexadecimal desde un *BigInteger* provoca estas situaciones. El proceso guardado e impresión en la interfaz gráfica se debe realizar por cada elemento de la llave pública (X e Y).

Otro elemento importante para la creación de la primera trama de Asociación es el *Nonce*. El *Nonce* al igual que lo que se realiza dentro del método "*getRandomSK",* utiliza

la clase "*SecureRandom"* para la creación. Con la creación posterior de un elemento *BigInteger* asociado con la clase mencionada, dentro de la creación se especifica la cantidad de bits que tendrá ese elemento randómico; para este caso se utilizarán 128 bits. Este valor deberá ser guardado como *String* para poder hacer la impresión en la interfaz gráfica; en la cadena se contará con 32 caracteres.

Uno de los principales elementos que se envían a través de la primera trama de Asociación es el testigo o *Witness*. El *Witness* será parte del proceso de autenticación que el Hub realiza para continuar con el proceso. El *Witness* se crea con la codificación en CMAC. La llave para esta codificación será el *Nonce* creado. El mensaje corresponderá a la concatenación de la dirección del Nodo, la dirección del Hub, la llave pública en X y la llave pública en Y. La salida necesaria será de 128 bits o 32 caracteres. Para realizar la codificación se crea un objeto de la clase CMAC y con el método "CalcCMAC" con los tres parámetros devolverá un *String* con el mensaje codificado.

Los elementos que debieron ser creados para que el Nodo envíe la primera trama de Asociación son:

- Llave privada Sk
- Cálculo de las coordenadas X e Y de la llave pública con el método de Curva Elíptica Pkx y Pky.
- Creación de un elemento randómico de 128 bits, *Nonce*.
- Creación del testigo o *Witness*

Para ensamblar la primera trama de Asociación se crea un objeto de la clase "PrimeraTramaAsociacion" y con el método "Asociacion1" se crea la trama. Este método admite como atributos la dirección del Nodo, la dirección del Hub, la llave pública en X, la llave pública en Y y el *Witness*.

El Nodo enviará la primera trama de Asociación y quedará a la expectativa de recibir la primera trama IACK seguida de la segunda trama de Asociación. Todos estos procedimientos se evidencian en la Figura 2.33 correspondientes a la fase de creación de elementos de asociación.

b.2.3 Fase de creación de elementos de autenticación

Una vez recibida la trama de Asociación tres, ésta será guardada para su uso en posteriores tramas y para extraer los elementos necesarios. El siguiente paso del proceso de Asociación corresponde a la creación de la llave *Diffie-Hellman* (DHKey).

La llave *Diffie-Hellman* es aquella que resulta de la multiplicación de la llave pública del Hub obtenida de la trama de Asociación tres, con la llave privada del Nodo. Las llaves se extraen como subcadenas del carácter 81 al 144 para la llave pública X y del elemento 145 al 208 para la llave pública en Y. Todo esto con el objetivo que entre las dos entidades se comparta la misma llave *Diffie-Hellman*.

Para realizar esta multiplicación dentro del campo de la curva criptográfica elíptica es necesario utilizar una curva diferente a la correspondiente P-256. Esta nueva curva tendrá como parámetros los mismos valores a, b, p de la curva P-256, con la diferencia que los puntos iniciales Gx y Gy corresponderán a las coordenadas de la llave pública. Todo esto se puede realizar debido a las propiedades de curva criptográfica elíptica, donde la multiplicación de un escalar por un punto de la curva elíptica retorna otro punto dentro de ésta. Para realizar la creación de esta curva se utiliza la clase "*MCurve",* donde el método constructor permite ingresar los 5 parámetros mencionados.

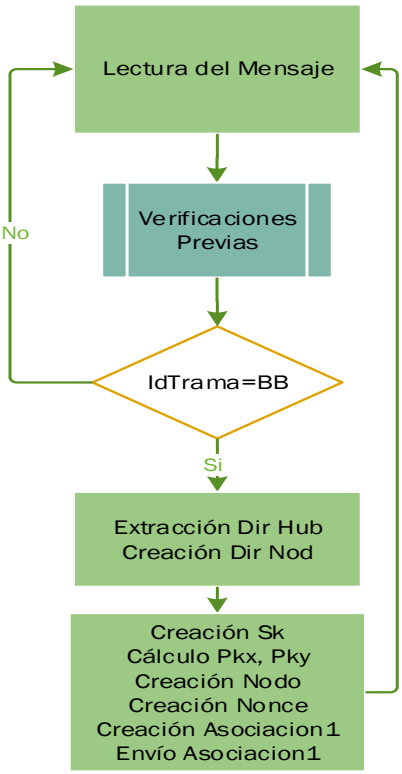


Figura 2.33. Fase de creación de elementos de Asociación.

Una vez creada la curva se debe obtener una nueva llave pública resultante de la llave privada del Nodo. Esto se lo realiza de igual manera con el método "*getPublicKey".* Este método acepta como atributo la llave secreta del Hub y entregará un nuevo punto perteneciente a la curva elíptica. Dentro del proceso se determina que la llave *Diffie-*

Hellman corresponderá a la coordenada X que devuelve el método. Para extraer la coordenada X se utiliza el valor cero del *array BigInteger* que retorna del método "*getPublicKey".* Esta llave *Diffie-Hellman* se guardará en formato *String* para poder ser impresa dentro de la interfaz gráfica.

El objetivo de la creación de la llave *Diffie-Hellman* es la transformación de esta llave en una llave temporal uno para los procesos CMAC que prosiguen a éstos. Para obtener esta llave temporal uno, se toma los primeros 128 bits o 32 caracteres de la llave *Diffie-Hellman*. Como se ha realizado a lo largo de este trabajo la obtención de una subcadena se realiza con el método "*subString"* de la clase "*String".*

La llave temporal uno servirá para el cálculo de los elementos MK_KMAC_3A y el MK_KMAC_4A. El cálculo de estos elementos corresponde a un procedimiento CMAC con llave Temp1, mensaje KMAC y salida de 64 bits, realizado por el método "CalcCMAC" de la clase CMAC. El mensaje por codificar en MK_KMAC_3A será la concatenación de:

- Dirección del Nodo
- Dirección del Hub
- *Witness* del Nodo
- *Nonce* del Hub
- *Security Suite Selector* de las tramas de Asociación.

El mensaje a codificar para el elemento KMAC_4A es muy similar a lo que requiere el mensaje de KMAC_3B con la diferencia que el orden del mensaje es:

- Dirección del Hub
- Dirección del Nodo
- *Nonce* del Hub
- *Witness* del Nodo
- *Security Suite Selector* de las tramas de Asociación.

Cabe recalcar que los elementos pertenecientes al Nodo para los mensajes a codificar se extraen de la trama de Asociación uno. Por ejemplo, la dirección del Nodo se guarda una vez recibida la trama de Asociación uno. El *Witness* utilizará el calculado para el envío de la trama de Asociación uno. El *Security Suite Selector* se lo puede extraer tanto de la

trama de Asociación uno como de la trama de Asociación dos desde el carácter 41 al carácter 44.

Para la creación de ambos elementos se crea un objeto de la clase CMAC y se colocan los parámetros antes mencionados dentro del método "*CalcCMAC*". Las variables *String* que se crean esperarán una cadena de longitud 16 caracteres.

El Nodo al recibir la segunda trama de Asociación realiza el cálculo de los elementos antes mencionados sin antes enviar la trama IACK2 con el objetivo de que en el lado del Hub también comience el proceso de cálculo de MK_KMAC3B y MK_KMAC_4B. La fase de creación de elementos de autenticación se muestra en la Figura 2.34.

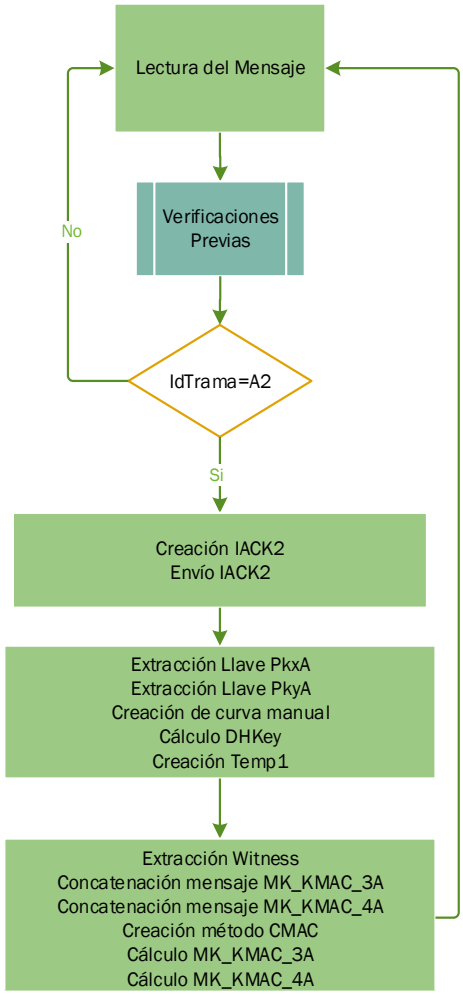


Figura 2.34. Fase de creación de elementos de autenticación.

b.2.4 Fase de autenticación del proceso

Una vez que el Nodo realiza el cálculo de los componentes MK_KMAC tendrá que esperar que el Hub envíe la tercera trama de Asociación para realizar las verificaciones

correspondientes y enviar la cuarta trama de Asociación. Esto significa que el Nodo es la entidad que primero realiza la autenticación utilizando MK_KMAC.

De la tercera trama de Asociación se extrae el elemento MK_KMAC_3B que se encuentra desde el carácter 209 al 224. Este elemento se compara con el elemento MK_KMAC_3A, creado por el Nodo. Si estos elementos son iguales el Nodo llamará a la creación de la cuarta trama de Asociación. El Nodo deberá crear un objeto de la clase "TramaAsociacionCuatro" y con el método "Asociacion4" ensamblará la cuarta trama de Asociación. Una vez creada la trama se envía al servidor para su posterior replicación hacia el Hub tal y como se mira en la Figura 2.35.

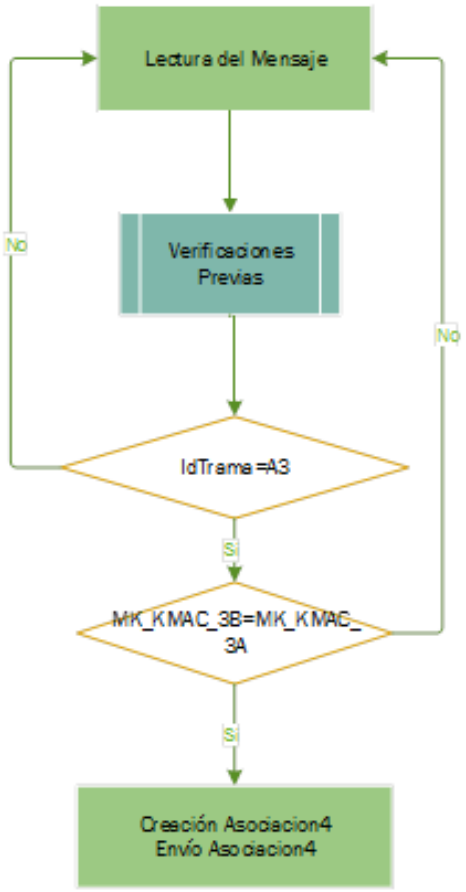


Figura 2.35. Autenticación MK_KMAC

El último proceso de verificación que se realiza para este protocolo de Asociación por parte del Nodo es la comparación de los números D mostrados en pantalla. El cálculo de este elemento se da a partir de la codificación con CMAC utilizando una concatenación entre el *Nonce* del Nodo y el *Nonce* del Hub como llave; además el mensaje será una concatenación del *Nonce* del Hub, *Nonce* del Nodo más la llave temporal uno. Todo esto

con una salida de 16 bits; esta salida será transformada en el número decimal que será presentado en la interfaz gráfica. La codificación se realiza con el método “CalcCMAC” de la misma forma que el utilizado en el cálculo de los MK_KMAC.

Esta trasformación de bits a decimal y la presentación de este número en la interfaz gráfica da el nombre a este proceso de Asociación. Si se verifica que lo números mostrados en pantalla son los mismos se puede proseguir a asociar a las entidades.

La asociación de las entidades se da cuando se calcula la llave maestra MK; la creación de esta llave maestra al igual que en el Hub se da con una codificación CMAC. En este caso la codificación se la realizará con una llave conformada por los 128 bits o 32 caracteres últimos de la llave *Diffie-Hellman* llamado temp_2. El mensaje a codificar será la concatenación del *Nonce* del Nodo y el *Nonce* del Hub con una salida de 128 bits. Todo esto realizado por el método “*CalcCMAC*”.

b.3 Integración de procesos sobre escenario de comunicaciones

Para la integración del proceso de Asociación se deben cumplir algunos requisitos como:

- El sistema debe contener un equipo que cumpla la función de servidor
- Dos equipos diferentes deben contener los ejecutables de cada entidad
- Los equipos que contengan los ejecutables y el servidor deben tener acceso de red
- Deben tener acceso a los métodos de envío y recibo de mensajes

Cuando los dispositivos cumplan con las condiciones mencionadas, cada entidad dentro del proceso creará un objeto del tipo conexión servidor donde se especificará tipo de trama colocando el identificador y el tipo de mensaje con su respectivo identificador.

La creación del objeto enviará al servidor, el cual replicará y dentro del proceso como se ha explicado se discriminará para que cada entidad cumpla con los pasos respectivos del proceso de Asociación. Una vez completado e integrados los procesos, la interacción de mensajes entre Nodo y Hub se hace en base al diagrama de secuencia de la Figura 2.7 presentada anteriormente en la sección c.1 del numeral 2.1.3 de este documento.

c. Estructuración de ataques sobre el proceso DAA

c.1 Ataque

c.1.1 Impersonalización del Nodo

Para que una nueva entidad realice impersonalización se deberá programar exactamente con los mismos pasos que contiene un proceso Nodo, haciendo que el atacante pueda asociarse con el Hub de manera efectiva.

Existen dos principales diferencias entre lo que se programa para un Nodo y lo que se programaría dentro de un atacante con impersonalización. La primera diferencia es el tipo de etiqueta que llevará cada mensaje que se envía por parte del atacante. Dentro de un proceso de Asociación normal se puede verificar que la etiqueta de mensaje que lleva un Nodo es “N1”, para que cuando un atacante quiera asociarse se pueda observar la etiqueta “NA”. Dentro de un Hub los procesos se discriminan básicamente por el identificador de trama y por el tipo de mensaje, haciendo que a pesar de que cambie el identificador de mensaje exista un proceso de Asociación.

La segunda diferencia entre los dos programas es que el contador de tramas en un proceso de Asociación normal está establecido en cuatro, mientras que para un Nodo atacante se establece en uno. La diferencia de este valor radica en que para un atacante no es necesario esperar que lleguen las cuatro tramas *Beacon* para empezar a asociarse. Un atacante empezará la Asociación apenas se haya corrido el programa. Además, un Nodo atacante está programado para que no reciba ninguna trama del Hub atacante.

Para que un Nodo no reciba ninguna trama del Hub atacante, dentro del Nodo se discrimina con la etiqueta “HA”; cualquier trama que llegue con esa etiqueta será descartada. En cambio, se ha determinado que un Nodo atacante esté programado para asociarse inmediatamente con un Hub normal y de manera inmediata.

La impersonalización se puede realizar debido a que como se explicó en el capítulo 1, nada impide que un atacante programado con funciones de Nodo pueda correr un protocolo de Asociación. La fase de creación de elementos de autenticación y la fase de autenticación de las entidades se hace con la entidad con la cual se ha empezado un proceso de Asociación. Por ello la impersonalización relativamente se vuelve sencilla bajo estas premisas.

La Impersonalización de Nodo se puede dar en un ambiente donde a pesar de que un Nodo normal haya ejecutado un proceso de Asociación, el Nodo atacante puede impersonalizar. Al ser programado para ejecutar un proceso apenas ejecute el programa,

si existe ya un proceso en marcha, el enviar la primera trama de Asociación hará que el Nodo atacante recorra el proceso de Asociación sobre el Hub. Es decir, la impersonalización se puede dar de forma solitaria o en un ambiente de Hombre en el medio. El diagrama de secuencias que muestra un Impersonalización del Nodo se presentó en la Figura 2.7, ubicada en la sección d.1.1 del numeral 2.1.3 de este documento.

c.1.2 Impersonalización de Hub

Para la Impersonalización del Hub es aún más sencillo ya que únicamente dentro del Hub se deberá tener como identificador de mensaje la etiqueta “HA”. Los procesos se mantendrán y el envío de tramas *Beacon* de igual manera será periódico.

Así como en un Hub normal, en el Hub atacante se empezará el protocolo de Asociación cuando reciba una trama de Asociación por parte de un Nodo normal. Como se mencionó un Nodo atacante con un Nodo Normal no podrán asociarse. En conclusión, un Hub atacante se comporta de la misma manera que un Hub normal.

Dentro de un Nodo normal se podrá verificar que llegan efectivamente las tramas de manera alternada tanto de un Hub normal como de un Hub atacante, esperando que sea la cuarta trama la que active el proceso. Si un Hub atacante no inició el proceso, a partir de ahí el atacante no podrá hacer la respectiva impersonalización. Debido a esto una Impersonalización del Hub solo se puede realizar en un escenario de Hombre en el medio. Esto también es debido a que un Nodo normal no está diseñado para recorrer el proceso de Asociación a menos que se ejecute el aplicativo nuevamente.

En conclusión, se tendrían únicamente dos ataques disponibles: una Impersonalización del Nodo o Ataque de Hombre en el medio, que lleva implícito una Impersonalización del Hub. Todo esto cuando se ejecutan los tres aplicativos al mismo tiempo, tomando en consideración que si se ejecutan únicamente un Hub atacante y un Nodo, existirá la respectiva impersonalización.

c.1.3 Programación Hombre en el medio

Para que exista una programación de Hombre en el medio, únicamente dentro del aplicativo del atacante se contarán con ambas impersonalizaciones. Para que coexistan ambas impersonalizaciones las variables deberán estar bien separadas e identificadas para que se ejecuten ambas sin ningún problema.

Pero para exista ataque de Hombre en el medio no es suficiente con ejecutar un ataque, ya que si dentro del envío de tramas *Beacon* el Nodo se activó con una trama *Beacon*

perteneciente a un Hub normal no podrá existir ataque en el medio por lo motivos ya mencionados. Para que exista ataque de Hombre en el medio el Nodo debió activar su proceso de Asociación con la trama *Beacon* del atacante. Además, esto se da ya que un Nodo atacante es más versátil y siempre se atacará si existen las condiciones necesarias y no se ha alterado el proceso de Asociación.

Si un Nodo no eligió asociarse con el Hub atacante, éste calculará las credenciales, pero no podrá autenticar ni con MK_KMAC ni con *Witness* ya que por denotadas razones no serán las mismas. Todo este proceso depende mucho de las primeras tramas de Asociación y para la creación de estas tramas de asociación se utilizan los identificadores de las entidades que han activado el proceso. El diagrama de secuencias que muestra un ataque de Hombre en el medio se presentó en la Figura 2.9, ubicada en la sección d.1.2 del numeral 2.1.3 de este documento.

d. Implementación de soluciones prácticas para vulnerabilidades

En base a lo presentado de manera teórica, se han desarrollado dos soluciones que evitan que un atacante pueda ingresar y proceder con el protocolo de Asociación. Como se expone teóricamente, la clave para que no exista ataque es utilizar conceptos de otro de los protocolos de Asociación que se encuentra dentro de IEEE802.15.6. Una llave precompartida y existente en las entidades a asociarse permitirán que las credenciales no dependan únicamente de las tramas enviadas.

d.1 Solución en base a modificación de mensaje

La primera solución se aplica concatenando en el mensaje del cálculo del elemento *Witness* la llave precompartida. El concatenar hará que la codificación CMAC resultante del proceso dependa de una llave adicional. Solo entidades que conozcan el método de solución sabrán que deberán añadir al final del mensaje del *Witness* esta llave precompartida.

Por ejemplo, lo que haría un Nodo sería calcular este nuevo *Witness* y enviarlo a través de la primera trama de Asociación uno, mientras que el Hub deberá tomar este elemento y compararlo con la creación de su propio *Witness* tomando en cuenta que debe añadir al final la clave precompartida. Un Hub atacante no podrá saber que el *Witness* creado es en base de una llave precompartida, haciendo que en el momento de autenticación con *Witness* falle y aborte el proceso. Un Nodo atacante enviará mal el *Witness* haciendo que en el Hub normal no se pueda realizar la autenticación lo que hará que este último aborte el proceso. Los nodos al no realizar autenticación con *Witness* pueden llegar a calcular

credenciales que si llegan a ser mostradas no serán válidas, ya que tanto del lado del Hub como del Nodo deberán ser iguales para que la Asociación esté completa.

d.2 Solución en base a modificación de clave

La segunda solución altera de manera diferente el cálculo de *Witness*. Esta solución plantea que se altere la clave de codificación en CMAC. Se realizará una encripción entre el *Nonce* y la llave precompartida. Pero no se intercambian las claves modificadas, sino las llaves normales que deberá tener un proceso de Asociación. Por eso solo la entidad que conozca de la operación cambiará la clave para realizar la codificación del *Witness*, y de la misma forma si no existe autenticación los procesos se abortan.

Todos estos procesos se lo realizan dentro de los cálculos y operaciones de Nodo y Hub normales, asegurando que si se ejecuta un proceso de Asociación entre éstos se complete de manera efectiva. Solo si se introduce un atacante, ninguna entidad se asociará y todos los procesos estarán abortados. Todo lo mencionado se podrá verificar en la interfaz gráfica.

Todos los procesos indicados se pueden ver codificados en lenguaje de programación Java en el Anexo II de este documento.

3 RESULTADOS Y DISCUSIÓN

En el presente capítulo se pondrán a disposición todas las pruebas y resultados obtenidos a partir de lo diseñado, programado y emulado en el capítulo 2. Las pruebas a realizar involucran:

- Emulación del proceso de Asociación.
- Pruebas de ingreso de un Atacante en el proceso de Asociación.
- Análisis de vulnerabilidades presentes en el proceso de Asociación.
- Verificación de la solución implementada dentro del proceso.

Para realizar la emulación del proceso de Asociación se necesita el escenario mostrado en la Figura 3.1, en el cual se incluyen dos computadoras de escritorio, donde la primera computadora deberá tener:

- Direccionamiento IP: 192.168.0.1.
- Un IDE de Java para realizar la ejecución del proceso.
- La clase Servidor, la cual realizará el reenvío de los mensajes hacia los otros equipos.
- Las clases necesarias para ejecutar el proceso de Asociación como Nodo, incluyendo la clase que ejecuta la interfaz gráfica, la misma que permite visualizar el proceso.

La segunda computadora será necesario que tenga:

- Direccionamiento IP: 192.168.0.2.
- Un IDE de Java para realizar la ejecución de las clases.
- Las clases necesarias para ejecutar el proceso de Asociación como Hub, incluyendo la clase que ejecuta la interfaz gráfica la misma que permite visualizar el proceso.

Para la inclusión del Atacante en el escenario es necesario disponer sus elementos tal y como se visualiza en la Figura 3.2.

En este caso, la tercera computadora que actuará como Atacante, será necesario que tenga:

- Direccionamiento IP: 192.168.0.3.
- Un IDE de Java para realizar la ejecución de las clases.
- Las clases necesarias para ejecutar el proceso de Asociación como Atacante incluyendo la clase que ejecuta la interfaz gráfica que permitirá visualizar el proceso. Cabe recalcar que, dentro de las clases que ejecutan el proceso de un Atacante se ejecutan procesos tanto de Hub como de Nodo. La interfaz gráfica para este equipo cuenta con una división, en la cual una parte estará destinada para el Atacante como Nodo y otra para el Atacante como Hub.

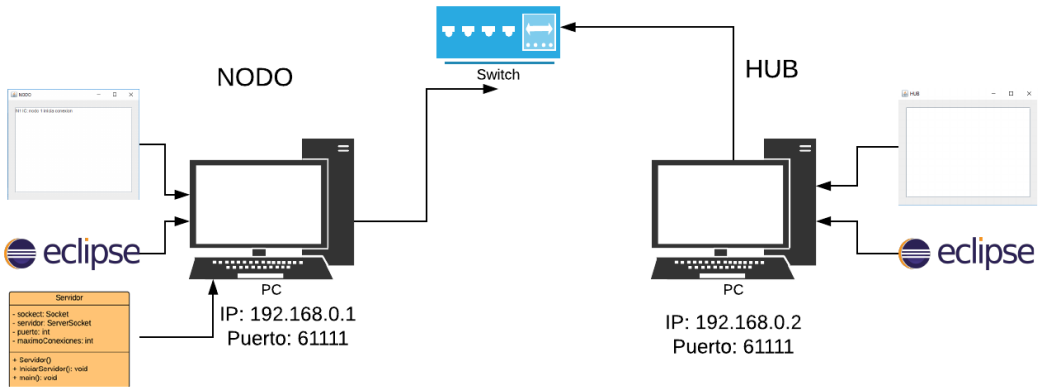


Figura 3.1. Escenario necesario para realizar la emulación del proceso de Asociación.

Para las pruebas de emulación del proceso de Asociación sobre el escenario mostrado en la Figura 3.1, se buscará verificar que el proceso de Asociación entre dos entidades (refiriéndose a entidades como Nodo, Hub o Atacante) se cumpla con éxito; en donde todas las entidades (en este caso PCs) se encuentran interconectadas en red a través de un *Switch* de capa 2. Para realizar esta verificación es necesario que se cumplan ciertos pasos tales como:

- Ejecución de la clase Servidor exitosa; es decir, el programa deberá estar listo para aceptar conexiones de las entidades y tener la capacidad de recibir y reenviar los mensajes que lleguen.
- Entidades conectadas de manera correcta hacia el Servidor a través de los *sockets* creados en la ejecución de las clases.
- Intercambio de mensajes entre las entidades Nodo y Hub a través de la conexión con el Servidor.

- Verificación en el Servidor de la llegada de los mensajes del proceso de Asociación de las dos entidades. Cabe recalcar que los mensajes tendrán identificadores de entidad tales como:
    - N1: Entidad 1 Nodo.
    - H2: Entidad 2 Hub.

  Además, se tendrán identificadores de trama tales como:
    - IC: Inicio de Conexión.
    - BB: *Broadcast Beacon.*
    - A1: Primera trama de Asociación.
    - A2: Segunda trama de Asociación.
    - A3: Tercera trama de Asociación.
    - A4: Cuarta trama de Asociación.
    - I1: Primera trama de IACK.
    - I2: Segunda trama de IACK
    - I3: Tercera trama de IACK.
    - I4: Cuarta trama de IACK.
- Culminación del proceso de Asociación, donde se mostrará la llave maestra en cada una de las interfaces gráficas.

Para la inmersión del Atacante dentro del proceso de Asociación se deberá verificar que se cumpla con lo siguiente:

- Se deberá comprobar que las conexiones sean exitosas, de igual forma que para el proceso de Asociación sin Atacante.
- Se deberá verificar que dentro del Servidor se impriman los mensajes pertenecientes al Atacante, los cuales tendrán los siguientes identificadores de trama:
    - NA: Atacante que ejecuta un proceso como Nodo.
    - HA: Atacante que ejecuta un proceso como Hub.

Cabe recalcar que los identificadores de trama son los mismos ya mencionados anteriormente.

- Además, se verificará si se logra tener un ataque exitoso o no; y si el ataque existe dentro del proceso de Asociación, se mostrará qué tipo de ataque se ejecutó.

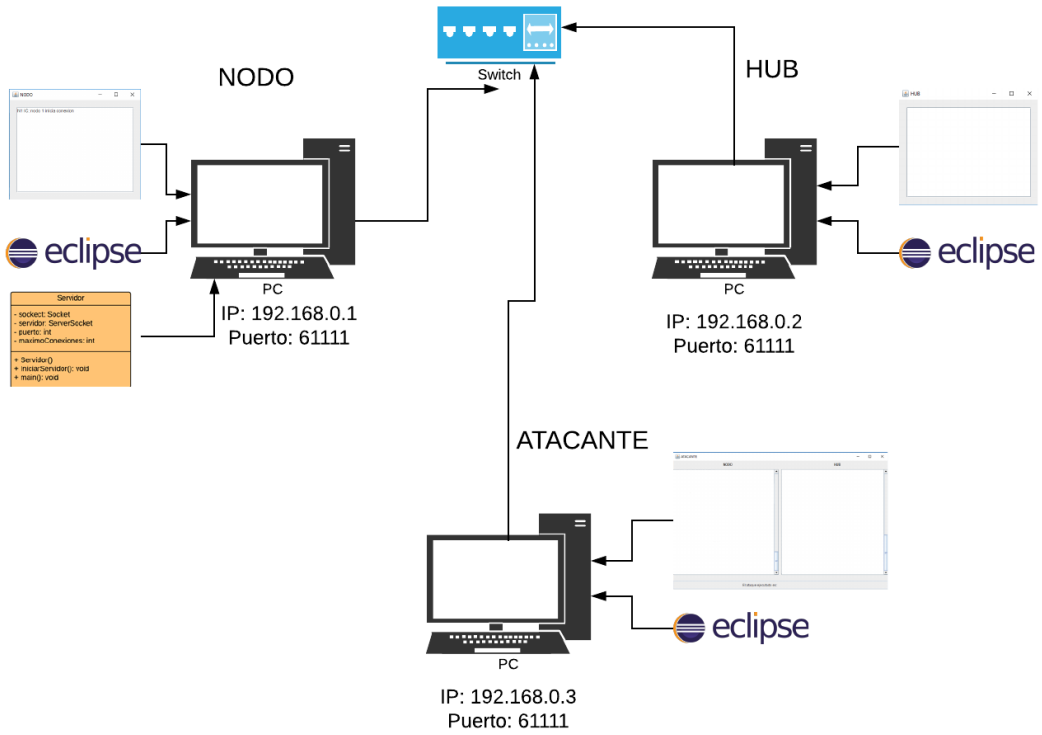


Figura 3.2. Escenario para ingreso de Atacante dentro del proceso de Asociación.

El orden de ejecución de los programas y el intervalo de tiempo que transcurre entre la ejecución de éstos influye mucho en el éxito o no de un ataque y el tipo de ataque que se obtiene; el ataque no podrá ser exitoso si éste se ejecuta después que las entidades culminen la fase de creación de elementos de Asociación. Por esta razón, en este capítulo se analizarán las vulnerabilidades del proceso cuando se ejecutan las siguientes secuencias y modificaciones:

- Ejecución secuencial de los programas en el orden Nodo - Hub - Atacante.
- Ejecución secuencial de los programas en el orden Nodo - Atacante - Hub.
- Ejecución con tiempo aleatorio de ingreso del Atacante:

- ✓ El Atacante trabajando como Hub reduce el intervalo de tiempo de envío de tramas *Beacon*.
- ✓ El Atacante trabajando como Hub aumenta el intervalo de tiempo de envío de tramas *Beacon*.
- ✓ El Atacante trabajando como Nodo es modificado para esperar hasta la cuarta trama *Beacon*, antes de que inicie el proceso de Asociación.

Al finalizar este capítulo, se mostrarán los resultados obtenidos cuando se realizan las pruebas de vulnerabilidades indicadas anteriormente, una vez que el Nodo y el Hub han sido modificados con la implementación de las soluciones mencionadas en el capítulo 2.

3.1 Pruebas de escenario de comunicaciones

Para iniciar con las pruebas del proceso de Asociación se debe ejecutar el programa Servidor a través del IDE instalado en el primer computador. Ya que como se mencionó en el capítulo 2, la clase Servidor permitirá recibir y reenviar los mensajes transmitidos por las entidades al ejecutar en el IDE sus respectivas clases.

La ejecución exitosa de la clase Servidor se puede evidenciar en la consola del IDE en la cual se ejecuta la clase, debido a que aparece el mensaje “Servidor a la espera de conexiones” tal y como se visualiza en la Figura 3.3. Este mensaje indicará que el Servidor podrá admitir las conexiones que se crearán de la ejecución de las clases respectivas de cada entidad que forman parte del proceso de Asociación. En la consola no se visualizarán cambios hasta que otra clase se haya ejecutado con la intención de que la entidad requiera conectarse hacia el Servidor. Si las conexiones se establecen con el Servidor, éste tendrá la capacidad de recibir los mensajes y reenviarlos hacia todas las entidades que se han conectado.

Para proseguir con las pruebas es necesario ejecutar las clases que contienen los procesos de las entidades Nodo y Hub. Cuando una entidad se va a conectar con el Servidor, en la consola del IDE correspondiente a la propia entidad se verificará que ésta requiere conectarse, es decir se evidenciará la dirección IP y puerto del Servidor junto con el puerto propio del equipo desde el cual se está conectando, tal y como se observa en la Figura 3.4. En la consola del Servidor se mostrarán las conexiones creadas a partir de la ejecución de las entidades, permitiendo visualizar la dirección IP del equipo que se

conectó y posteriormente el nombre con el que va a ser identificado dentro del Servidor, como se muestra en la Figura 3.5.

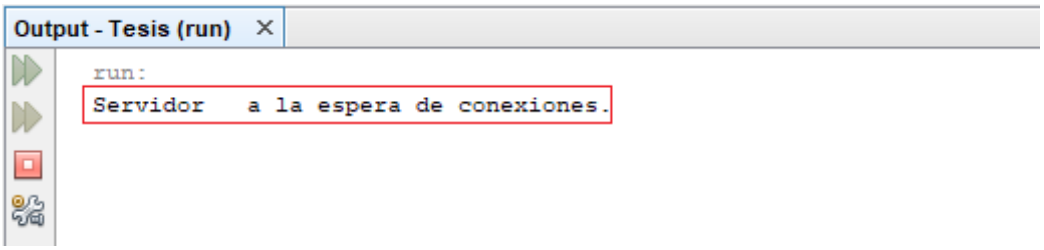


Figura 3.3. Servidor a la espera de conexiones.

A pesar de que ya se estén intercambiando mensajes entre entidades conectadas, el Servidor permanentemente aceptará nuevas conexiones, es decir el Servidor siempre se encontrará a la espera de nuevas conexiones, hasta alcanzar el límite de 64 conexiones establecido en el estándar IEEE 802.15.6. Por lo tanto, si un Atacante ingresa después de algún intercambio de mensajes entre entidades, el Servidor no presentará obstáculo para que esa conexión se realice.

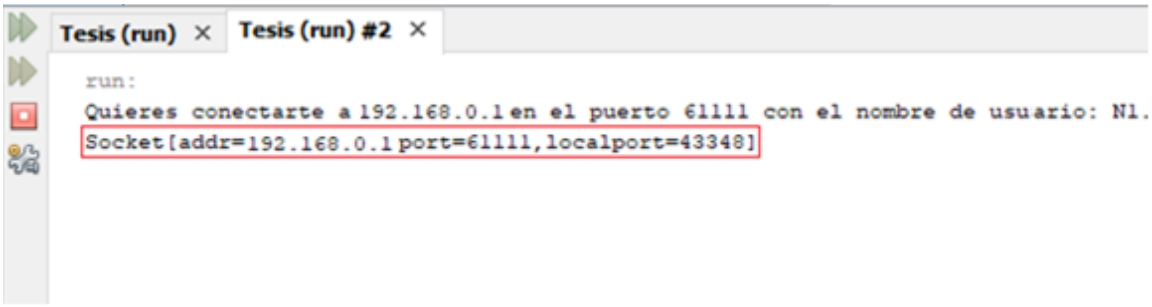


Figura 3.4. Entidades conectadas.

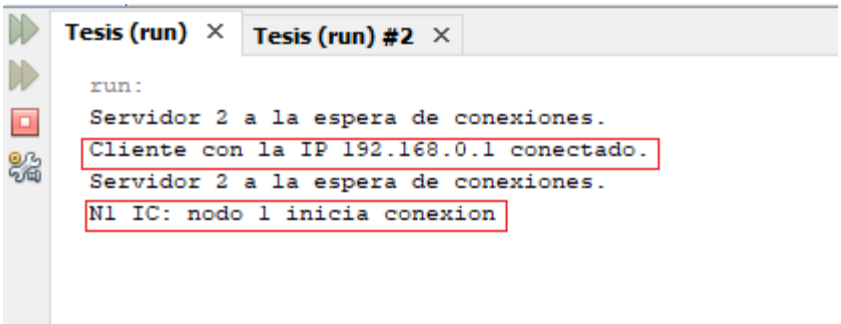


Figura 3.5. Conexiones dentro de Servidor.

Toda entidad a partir de que se conecta empezará a recibir todos los mensajes que retransmite el Servidor, inclusive aquellos que han sido enviados por la propia entidad. El Servidor se encarga de recibir y reenviar los mensajes, pero debido a la correcta programación y uso de identificadores para cada trama, cada entidad tomará el mensaje que le corresponde para continuar el proceso de Asociación. Dentro del Servidor se podrá evidenciar todo el intercambio de mensajes provenientes de las entidades. Los mensajes pueden ser provenientes de un Nodo, un Hub o un Atacante; todos los mensajes se distinguirán por los identificadores de entidad y trama como se nombraron en el inicio de este capítulo. Para el escenario de la Figura 3.1 y una vez ejecutado de manera exitosa el proceso de Asociación, en la consola del Servidor se pueden observar todos los mensajes que formaron parte del proceso, similar a lo que se observa en la Figura 3.6.

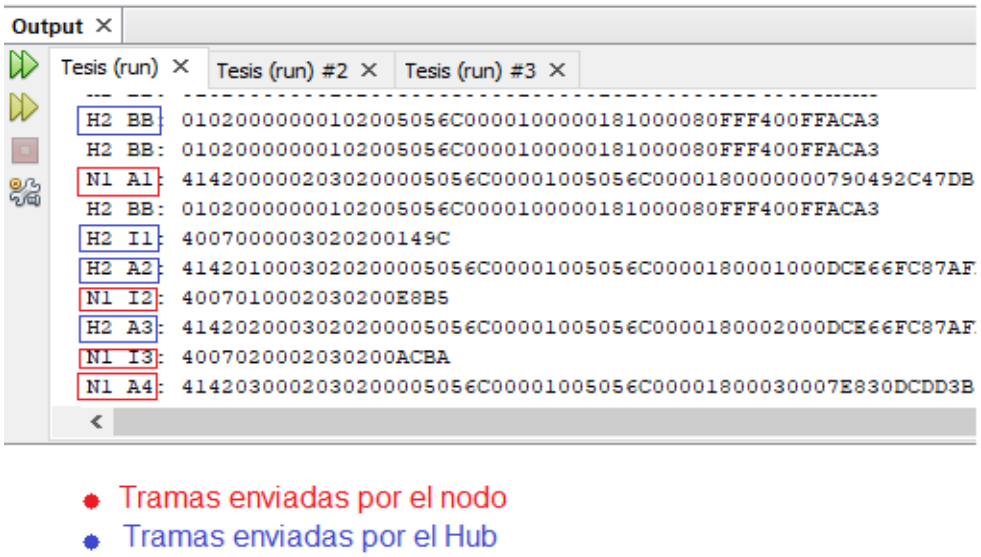


Figura 3.6. Intercambio de mensajes entre entidades visualizadas en la consola del Servidor.

3.2 Pruebas de Protocolo de Asociación

Una vez evidenciado que dentro del Servidor se realiza de manera correcta el intercambio de mensajes, es necesario verificar para cada entidad que el protocolo de Asociación se lleva a cabo exitosamente. Las interfaces gráficas pertenecientes a cada una de las entidades sirven de herramienta para ir comprobando el proceso de Asociación, intercambio de mensajes, creación de llaves, entre otros de manera interactiva. La Figura 3.7 muestra la interfaz que utilizará el Nodo, mientras que la Figura

3.8 muestra la interfaz usada por el Hub. Como se puede apreciar, ambas interfaces son muy similares, las diferencias se podrán corroborar una vez ejecutado el proceso de Asociación.

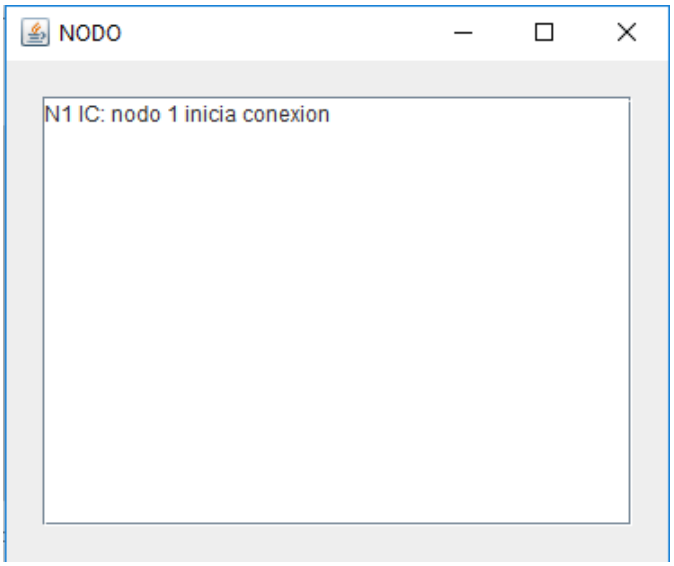


Figura 3.7. Interfaz gráfica de Nodo.

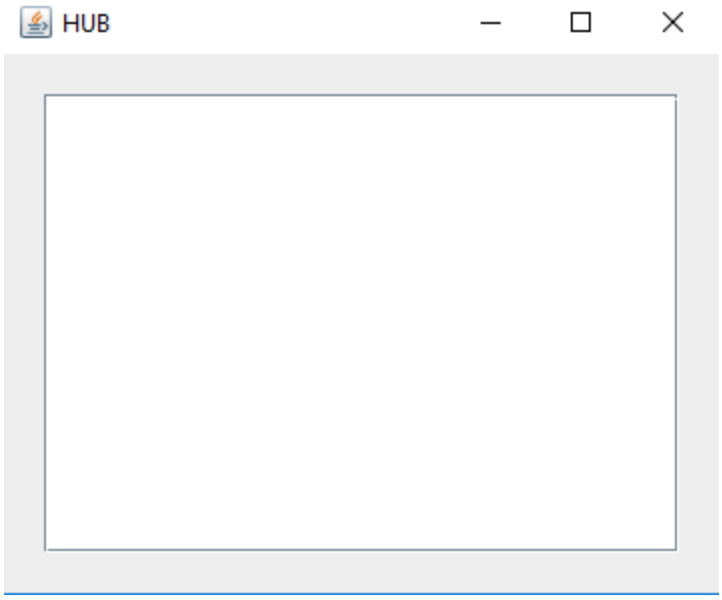


Figura 3.8. Interfaz gráfica del Hub.

De lo visto en el capítulo 1, el proceso de Asociación es una concatenación de pequeños procedimientos que cada entidad deberá ir ejecutando para crear las diferentes tramas que se irán intercambiando hasta lograr la Asociación; cada uno de esos pasos se podrán visualizar a través de las interfaces. Dentro del escenario de la Figura 3.1 y basados en la

Figura 2.7, el proceso de Asociación se podrá evidenciar en las interfaces gráficas con los siguientes procesos:

- Una vez establecidas las conexiones de las entidades con el Servidor, el Hub empezará a enviar tramas *Beacon*. El Nodo recibe estas tramas de manera secuencial, las mismas que se irán imprimiendo en la interfaz gráfica Nodo. El Nodo una vez que recibe la cuarta trama *Beacon* activa el proceso de Asociación; por tal razón, como se observa en la Figura 3.9, el Nodo imprimirá 5 veces la trama *Beacon*: 4 veces antes de activar el proceso de Asociación y una vez activo el proceso imprime la última trama válida que llega al Nodo, en este caso una trama *Beacon* que será la utilizada para armar la primera trama de Asociación.

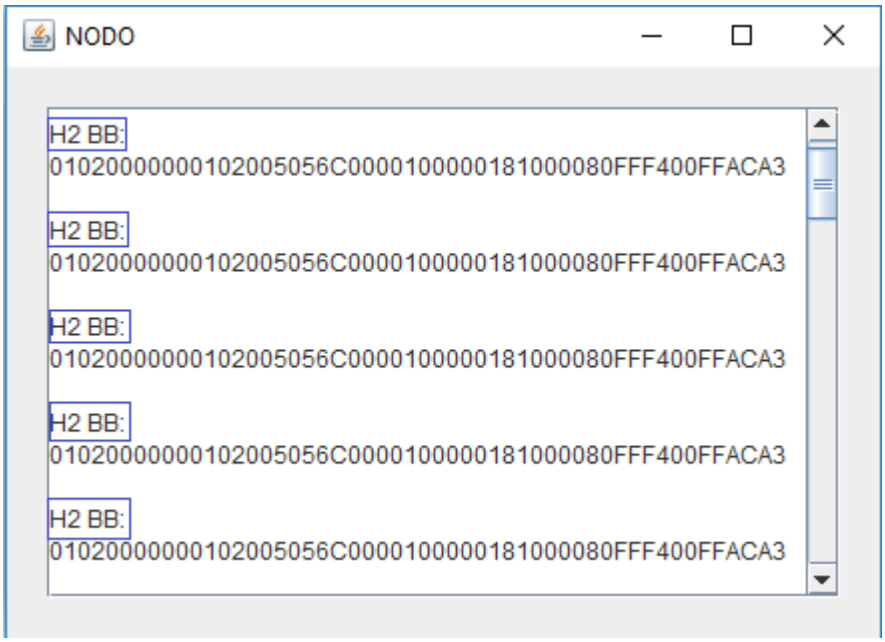


Figura 3.9. Nodo recibe tramas *Beacon.*

- Cuando el Nodo recibe las tramas *Beacon* y se activa el proceso de Asociación, comienza la fase de creación de elementos de Asociación; en la interfaz gráfica se verificará paso a paso cada uno de los elementos creados en esta fase, tales como Dirección Nodo, Dirección Hub, Llave privada, Llaves públicas, *Nonce*, Witness y primera trama de Asociación. A pesar de que no todos los elementos van a ser enviados dentro de la primera trama de Asociación, todos son necesarios para calcular esta primera trama. Cabe recalcar que estos valores no son fijos, ya que en cada proceso de Asociación que se ejecute se crearán nuevos elementos de

Asociación. La fase de creación de elementos de Asociación dentro del Nodo, mostrada a través de la interfaz gráfica, se puede observar en la Figura 3.10.

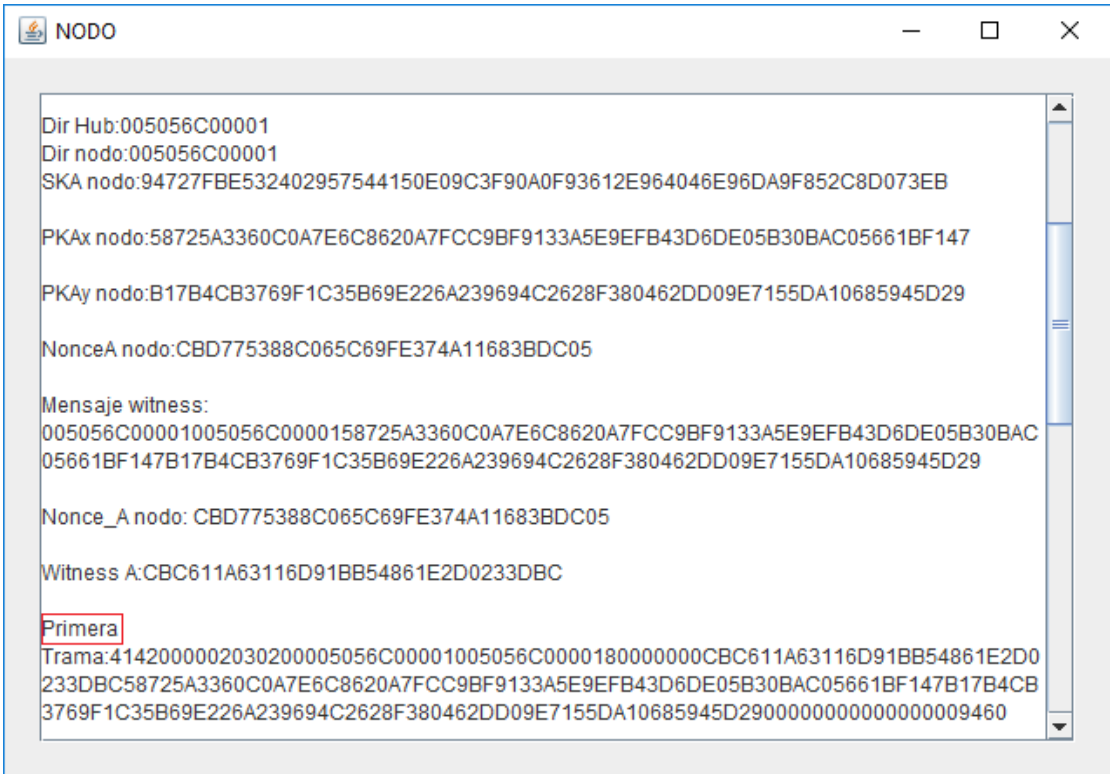


Figura 3.10. Creación de elementos de Asociación.

- Cuando el Nodo calcula todos los elementos de la primera fase y envía la primera trama de Asociación, el Hub envía el correspondiente IACK e inmediatamente inicia el proceso de Asociación. Una vez iniciado el proceso de Asociación, el Hub comienza la fase de creación de los elementos de Asociación, calculando e imprimiendo elementos como Dirección Nodo, Dirección Hub, Llave privada, Llaves públicas, *Nonce* y segunda trama de Asociación, tal como se ve en la Figura 3.11.

- Después de que el Hub calcula los elementos de Asociación y envía la segunda trama de Asociación, tanto el Nodo como el Hub comienzan la fase de creación de elementos de autenticación. Entre los elementos que se irán calculando en las entidades están clave Diffie-Hellman, llaves temporales, MK_KMAC_3B, MK_KMAC_4B, MK_KMAC_3A, MK_KMAC_4A y tercera trama de Asociación; estos elementos se irán imprimiendo sobre la interfaz gráfica tal y como se observa en las Figuras 3.12 y 3.13. El Hub enviará la tercera trama de Asociación para continuar con el proceso.

Figura 3.11. Creación de elementos de Asociación Hub.

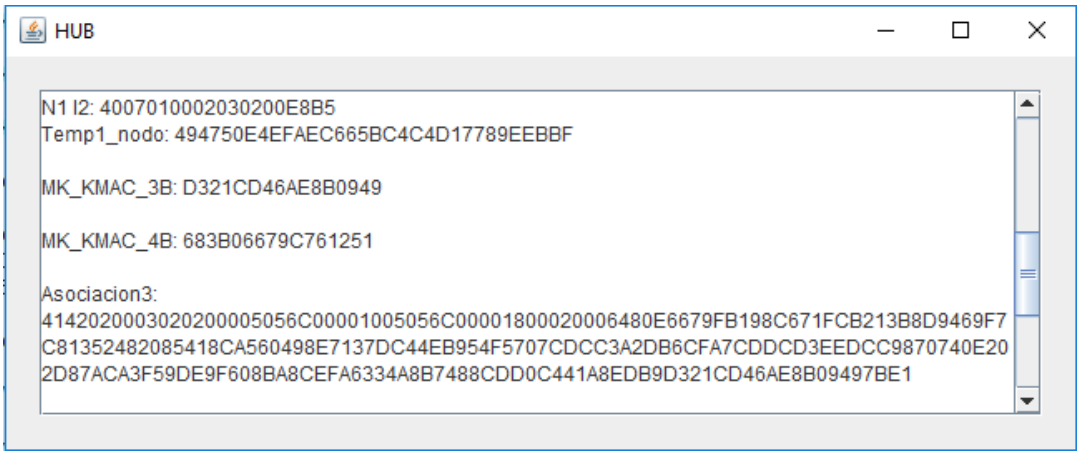


Figura 3.12. Creación de elementos de autenticación Hub.

- El Nodo al recibir la tercera trama, autentica y envía la cuarta trama de Asociación, con los elementos de autenticación para que el Hub realice la respectiva comprobación y culmine la Asociación entre las entidades.
- Una vez intercambiadas todas las tramas de Asociación entre las entidades, éstas comienzan la comprobación final. La autenticación por *display* determina que por parte de las entidades se muestre el número decimal en *display,* el cual debe ser el mismo en ambas entidades; la verificación de esto se realiza dentro del proceso propio de las entidades, pero también se puede realizar con la observación a través

de la interfaz gráfica. Una vez comprobados estos parámetros, se calculará la llave maestra MK quedando activada para ambas entidades.

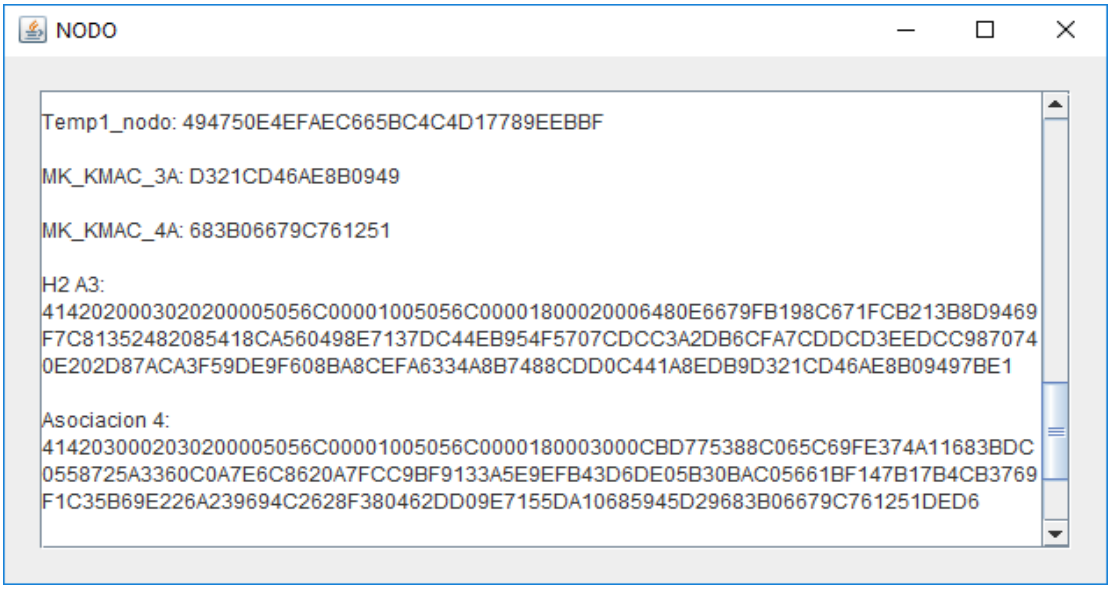


Figura 3.13. Creación de elementos de autenticación en el Nodo.

Como se observa en las Figuras 3.14 y 3.15, los números *display* y MK son exactamente iguales para ambas entidades, quedando completa la Asociación entre las entidades Nodo y Hub. Si ambas entidades cuentan con la misma llave MK, las entidades están listas para cumplir con el resto del proceso de seguridad, especificado en el paradigma del estándar IEEE 802.15.6.

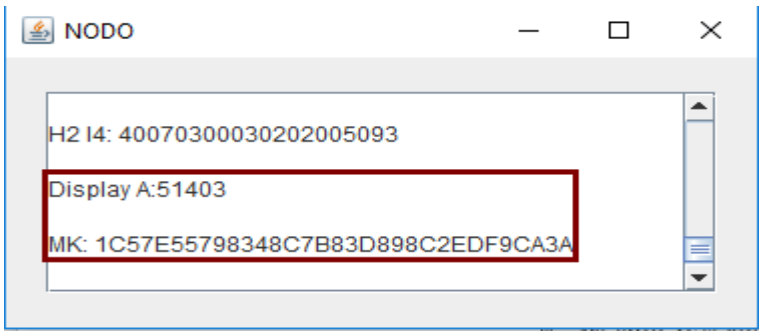


Figura 3.14. Asociación completada en el Nodo.

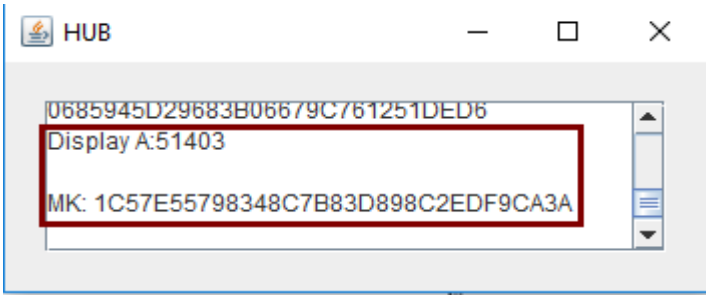


Figura 3.15. Asociación completada en el Hub.

3.3 Pruebas del Ataque

Esta sección tiene como objetivo mostrar las distintas formas en las que un ataque se puede ejecutar y las posibles vulnerabilidades que existen dentro del proceso de Asociación DAA. La Figura 3.2 muestra el escenario en el cual se podrá introducir un ataque ya sea, Hombre en el medio o Impersonalización del Nodo, dentro del proceso de Asociación entre Nodo y Hub.

Un Atacante, como se indicó en el capítulo 2, está programado con las funcionalidades tanto de Nodo como de Hub, pero eso no implica que el Atacante siempre ejecutará ataque Hombre en el medio. Como se mencionó al inicio de este capítulo, las secuencias en la que se inician los programas (Nodo, Hub y Atacante) influyen en la obtención o no de un ataque sobre el proceso, y en el caso que se presente un ataque, el tipo de ataque que se ejecutará.

A continuación, se explican cada una de las secuencias y sus resultados.

3.3.1 NODO - HUB – ATACANTE

En esta primera opción se considera que el Servidor ya está activo y los programas se irán ejecutando de manera continua y sincronizada bajo esta combinación, es decir se correrá el programa Nodo seguido del programa Hub y finalmente el correspondiente al Atacante. El Nodo comenzará a recibir las tramas de Asociación de manera alternada hasta completar la cuarta e iniciará el proceso de Asociación; las tramas se alternan entre las que envía un Hub y un Atacante/Hub[12]. Bajo esta secuencia de ejecución de los programas, el Nodo recibirá dos tramas del Hub y dos tramas del Atacante/Hub; la cuarta trama pertenecerá al Atacante/Hub y por consiguiente esta trama dará inicio a la

[12] **Atacante/Hub:** Cuando el programa atacante está utilizando las funciones de un Hub.

Asociación entre el Nodo y el Atacante. En cuanto al Atacante/Nodo[13], éste no esperará a la cuarta trama para asociarse, únicamente activará el proceso de Asociación con la primera trama que llegue por parte del Hub. Bajo esta combinación se comprueba que el Atacante empezará un proceso de Asociación tanto con el Hub como con el Nodo, por ende, se ejecutará un ataque de Hombre en el medio.

La Asociación se ejecutará entre el Atacante y las demás entidades, bajo la misma secuencia que la analizada en la sección 3.2. La Figura 3.16 muestra el valor de *display* y llave maestra calculada en el Hub; al compararlos con los valores mostrados en la Figura 3.18 (Nodo) se observa que se tienen los mismos valores de *display* y MK. Del mismo modo en las Figuras 3.17 y 3.18 (Hub) se observan los mismos valores en las interfaces Nodo y Atacante/Hub.

Es evidente que entre el Nodo y el Hub no comparten la misma llave maestra, produciéndose ataque de Hombre en el medio; el Nodo y el Hub creen haberse asociado con la entidad correspondiente, pero consiguen asociarse con un Atacante. Analizando las vulnerabilidades, las entidades Nodo o Hub no tienen la capacidad de autenticar si es un Atacante o una entidad con parámetros IEEE 802.15.6. Los procesos se corren impersonalizando a ambas entidades por parte del Atacante, siendo este último un elemento intruso en la red conformada.

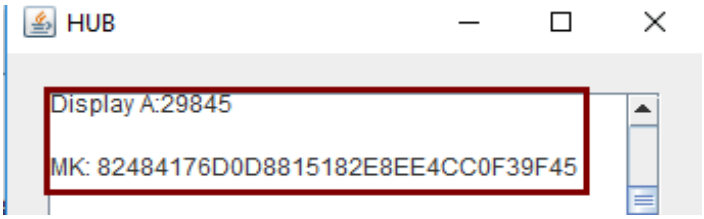


Figura 3.16. Hub en un Ataque de Hombre en el medio.

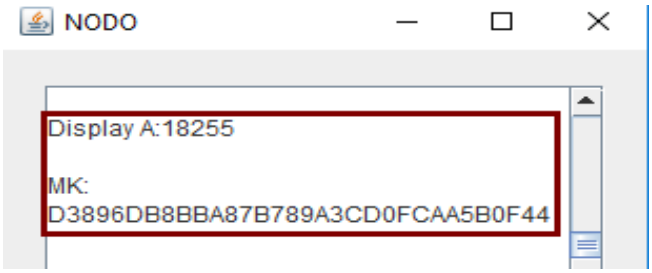


Figura 3.17. Nodo en Ataque de Hombre en el medio.

[13] **Atacante/Nodo**: Cuando el programa atacante está utilizando las funciones de un Nodo.

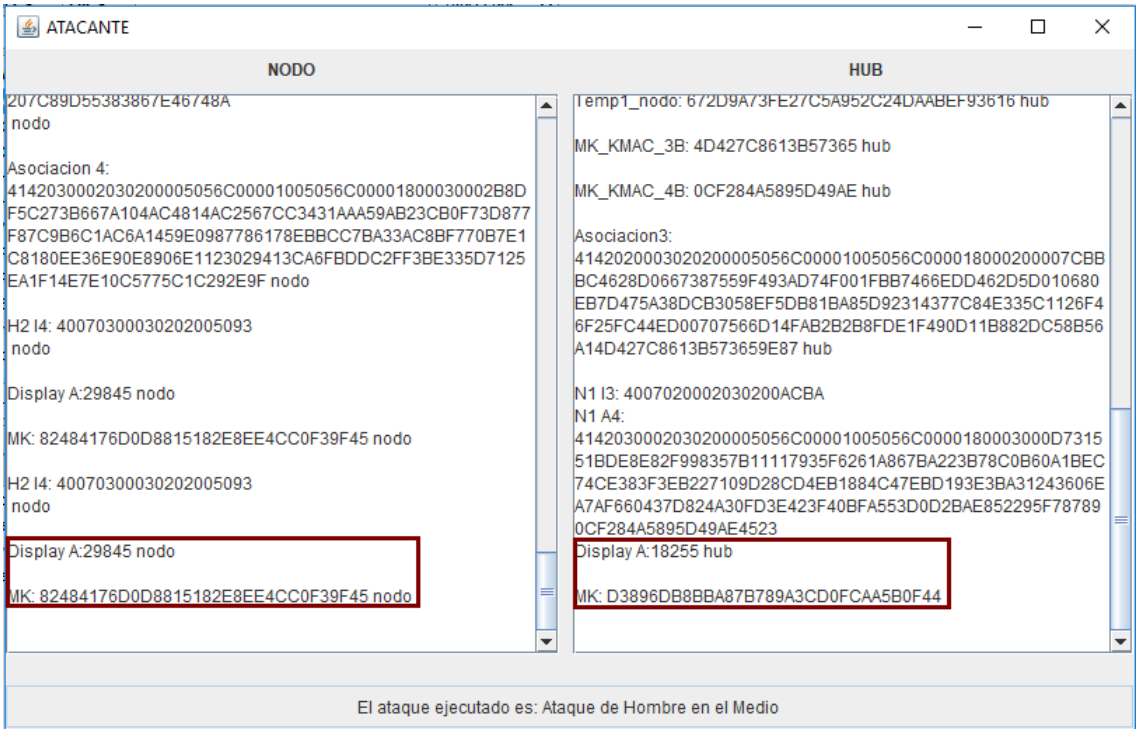


Figura 3.18. Ataque de Hombre en el medio.

3.3.2 NODO - ATACANTE – HUB

En esta segunda opción los programas se irán ejecutando de manera continua y sincronizada bajo esta segunda combinación, es decir se correrá el programa Nodo seguido del programa Atacante y finalmente el correspondiente al Hub. Ahora el Nodo no recibirá la cuarta trama *Beacon* proveniente del Atacante, sino será el propio Hub el que active el proceso de Asociación con su trama *Beacon*. Cuando el proceso de Asociación se acciona, las entidades Nodo y Hub intentarán asociarse, pero el Atacante realizará una denegación de servicio impidiendo que el proceso de Asociación se complete. El Atacante/Nodo recibe una única trama *Beacon* y enviará la primera trama de Asociación hacia el Hub, esto provoca una re-ejecución del proceso, quedando inválida la fase de creación de elementos de Asociación entre el Nodo y el Hub. En el nuevo proceso, los nuevos elementos de Asociación se crearán únicamente sobre el Hub y el Atacante/Nodo.

El Atacante/Nodo comienza a asociarse con el Hub mientras que el Nodo quedaría en un estado de huérfano hasta que otra entidad envíe tramas *Beacon* para reejecutar el proceso. El Atacante/Hub queda sin la posibilidad de enviar nuevamente tramas *Beacon* colocándose de igual manera en estado huérfano. El Nodo estará recibiendo las tramas

enviadas por el proceso, creyendo que sigue asociándose con el Hub, pero este último ya se encuentra en proceso de Asociación con el Atacante/Nodo. De manera didáctica se muestran las credenciales creadas por parte del Nodo, pero se indicará que el proceso ha fallado por problemas en la autenticación. Mientras que el Atacante/Hub indica que su proceso no continuó ya que no se han intercambiado tramas con él, impidiéndose continuar con el proceso.

El hecho de que se asocien únicamente Atacante/Nodo con Hub da un ataque de Impersonalización de Nodo. Las Figuras 3.19, 3.20 y 3.21 muestran cómo quedarían los valores de *display* y MK en un proceso de Impersonalización de Nodo. La secuencia mencionada en esta sección da como resultado un 100% de ejecución de este tipo de ataque.

Como se puede apreciar en las Figuras 3.19, 3.20 y 3.21, los elementos en el Nodo no deberían crearse, pero se muestran para indicar que la autenticación ha fallado. Las llaves maestras entre un Hub y el Atacante/Nodo coinciden, mientras que el Atacante/Hub no completa exitosamente su Asociación.

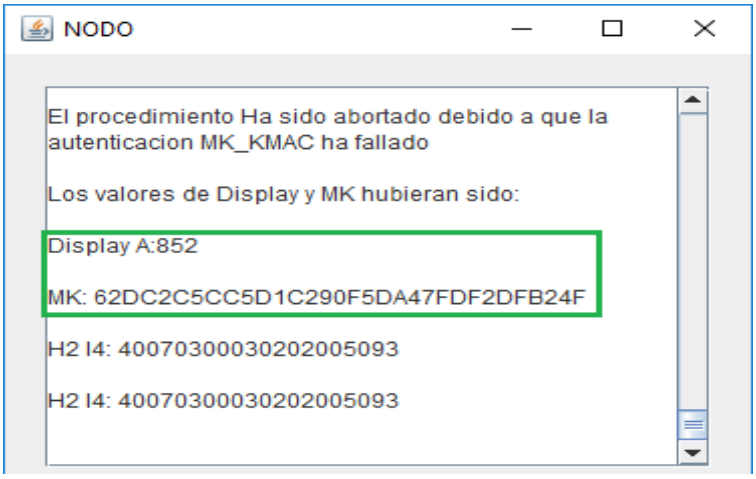


Figura 3.19. Falla de autenticación en Nodo.

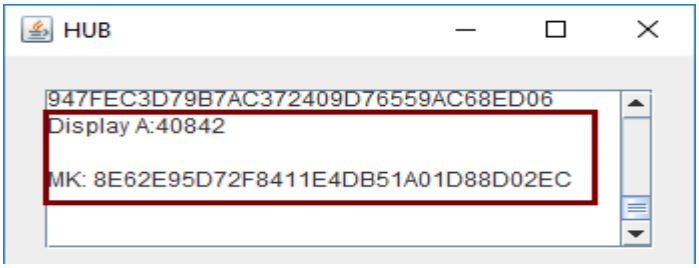


Figura 3.20. Asociación en Hub con Impersonalización de Nodo.

Figura 3.21. Ataque de Impersonalización del Nodo.

3.3.3 ACCESO EN TIEMPO ALEATORIO DEL ATACANTE

Para que el Atacante tenga un ingreso exitoso y pueda interrumpir el proceso de Asociación DAA entre dos entidades se deben cumplir ciertas condiciones. Entre las condiciones para que un ataque se dé con éxito, es que éste se ejecute antes de que entre el Nodo y el Hub culmine la fase de creación de elementos de Asociación. Esto último implica que, si un Nodo y un Hub ya intercambiaron la primera y segunda trama de Asociación, el Atacante ya no podrá ingresar al sistema. Por esta razón, es importante que el tiempo de ingreso del atacante una vez que se ha ejecutado Nodo y Hub, sea menor al tiempo de 12 segundos; en donde, en base a mediciones se determinó que culmina la fase de creación de elementos de Asociación. Para realizar estas pruebas, el ingreso aleatorio del Atacante debe estar en un rango de tiempo de 0 a 11 segundos, una vez que se ha ejecutado el proceso Nodo y Hub.

El no ingreso del Atacante se debe a que un Atacante/Nodo necesita las tramas *Beacon* por parte del Hub para ingresar; el Hub al empezar a asociarse detiene el envío de las tramas *Beacon* impidiendo el ingreso del Atacante/Nodo. Para el caso del Atacante/Hub, éste necesita la primera trama de Asociación por parte del Nodo; si ya ha sido enviada no tendrá cómo interrumpir el proceso.

Para el ingreso aleatorio de un Atacante se han considerado algunos casos, en los cuales se evaluarán la cantidad y porcentaje del tipo de ataque que se ejecutó, tomando en cuenta las premisas descritas en los dos primeros casos de ataques.

a. Atacante sin modificación

Para este literal se considera que el Atacante no ha sufrido ninguna modificación en su código, siendo el mismo utilizado para las secciones 3.3.1 y 3.3.2, pero con un ingreso en tiempo aleatorio y secuencia no controlada como se realizó en dichas secciones. Además, se considera que la ejecución del Atacante se dará antes del intercambio de la primera y segunda trama de Asociación, evitando que el Atacante quede fuera del proceso, para poder evaluar las vulnerabilidades.

Para este caso se obtienen los resultados mostrados en la Tabla 3.1.

Tabla 3.1. Resultados Caso a.

Ataque ejecutado	Número de veces
Ataque de Hombre en el medio	13
Ataque de Impersonalización de Nodo	9
Total	22

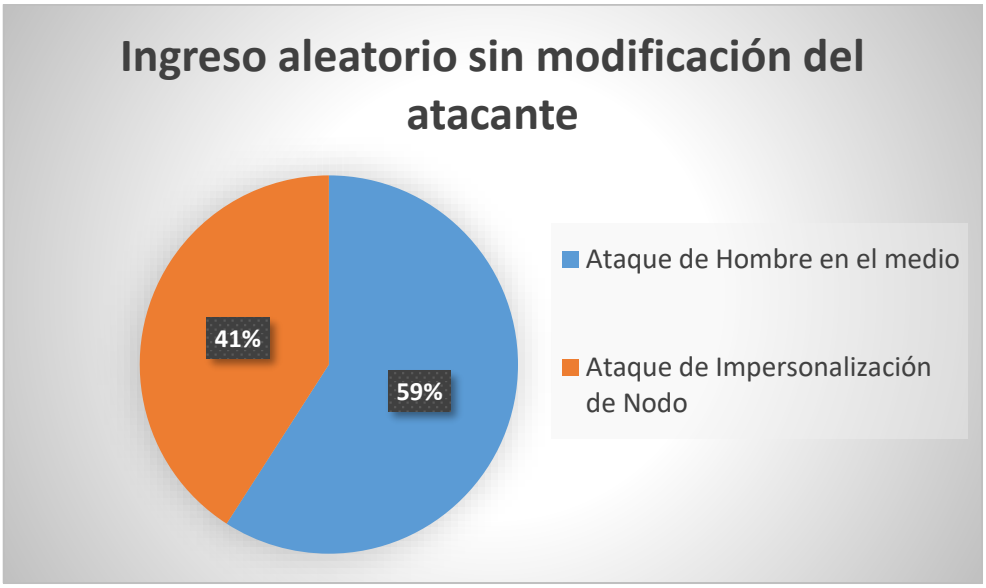


Figura 3.22. Porcentaje de ataques Caso a.

La Figura 3.22 muestra el porcentaje de tipos de ataques logrados para estas condiciones. Como se puede apreciar en esta figura, y gracias a la programación con la que cuenta el Atacante, la tendencia es que se ejecute un ataque de Hombre en el medio

en mayor porcentaje. Como se explicó en las secciones 3.3.1 y 3.3.2, las condiciones para que se ejecuten los dos tipos de ataques se cumplen, pero con mayor predisposición para ataques de Hombre en el medio. Adicional a eso, se puede evidenciar que el tipo de ataque que se ejecuta en el proceso se ve influenciado directamente por la trama *Beacon* que hizo al Nodo empezar el proceso de Asociación.

b. Atacante modificado, envío de tramas en menor tiempo

Para este caso, el ingreso del Atacante se realizará de la misma forma que en el literal a, pero ahora con modificación en la programación del Atacante. Para el envío de tramas *Beacon* por parte del Atacante/Hub originalmente se contaba con un temporizador de valor 3 segundos o 3000 milisegundos; para este caso se considera una reducción en el temporizador que controla el envío de tramas *Beacon* a un valor de 1 segundo o 1000 milisegundos. Los resultados producto de la reducción del tiempo con el que se envían tramas *Beacon* en el Atacante/Hub se muestran en la Tabla 3.2.

Tabla 3.2. Resultados Caso b.

Ataque ejecutado	Número de veces
Ataque de Hombre en el medio	18
Ataque de Impersonalización de Nodo	4
Total	22

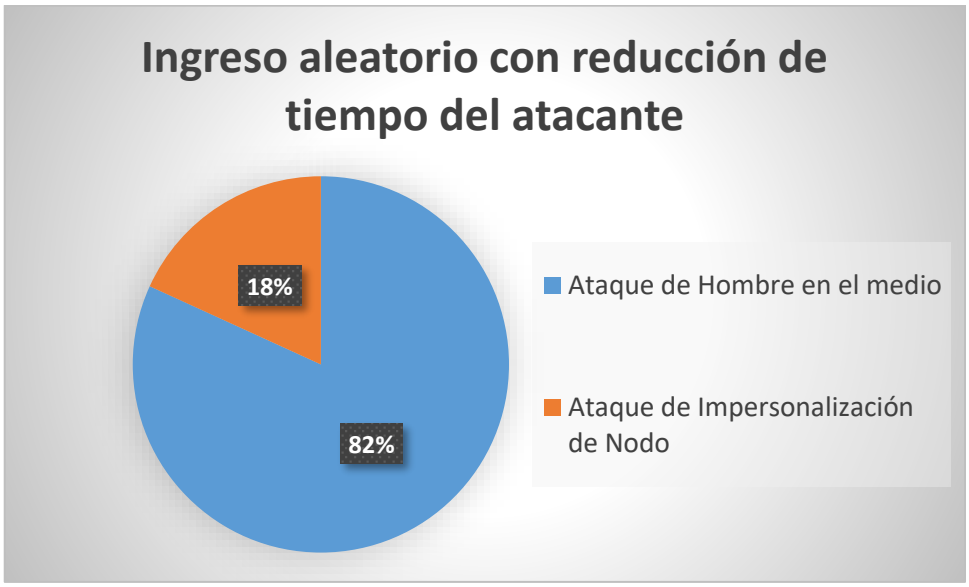


Figura 3.23. Porcentaje de ataques Caso b.

La Figura 3.23 muestra el porcentaje de tipos de ataques logrados para estas condiciones. Como se puede apreciar en la Figura 3.23, el porcentaje de ataques de Hombre en el medio aumenta. El aumento del porcentaje de ataque de Hombre en el medio se debe a que el envío de tramas por parte del Atacante/Hub aumenta en frecuencia, incrementando la probabilidad de que la trama tomada por el Nodo para asociarse sea la que envía el Atacante/Hub.

c. Atacante modificado, envío de tramas en mayor tiempo

Para este caso el ingreso del Atacante se realizará de la misma forma que en el literal b, pero con modificación en el tiempo de tramas *Beacon*. Si bien es cierto el Atacante/Hub originalmente contaba con un temporizador del mismo valor (3 segundos o 3000 milisegundos), ahora para este caso se considera un aumento en el temporizador que controla el envío de tramas *Beacon,* el mismo que será de 5 segundos o 5000 milisegundos. Los resultados producto del aumento del tiempo con el que se envían las tramas *Beacon* en el Atacante/Hub se muestran en la Tabla 3.3.

Tabla 3.3. Resultados Caso c.

Ataque ejecutado	Número de veces
Ataque de Hombre en el medio	6
Ataque de Impersonalización de Nodo	16
Total	22

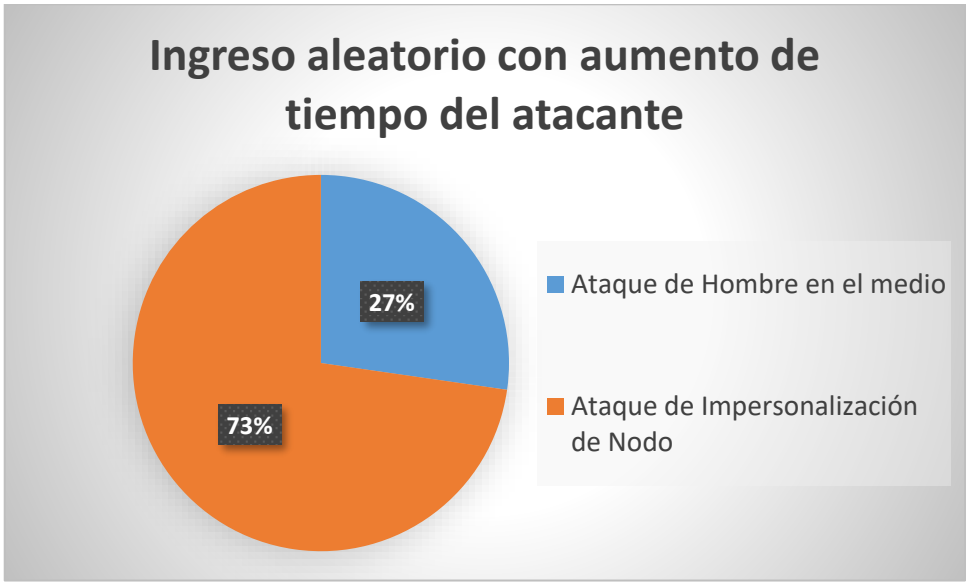


Figura 3.24. Porcentaje de ataques Caso c.

La Figura 3.24 muestra el porcentaje de tipos de ataques logrados para estas condiciones. Como se puede apreciar en esta figura, el porcentaje de ataques de Impersonalización de Nodo aumenta, si se compara con los resultados obtenidos en los literales a y b de esta sección. El aumento del porcentaje de Impersonalización de Nodo se debe a que el envío de tramas por parte del Atacante/Hub reduce en frecuencia, incrementando la probabilidad de que la trama tomada por el Nodo para asociarse sea la que envía el Hub. Se debe recordar que, si la trama tomada por el Nodo para comenzar con el proceso de Asociación es la del Hub existirá una cancelación de ese proceso, provocando que al final el ataque ejecutado sea de Impersonalización de Nodo.

d. Atacante modificado, Nodo/Atacante modificado

Como se mencionó en las secciones 3.3.1 y 3.3.2 el Atacante/Nodo empezará la Asociación apenas reciba una trama *Beacon*, a diferencia de un Nodo que necesitará de una cuarta trama para empezar la Asociación. Para este caso, se han modificado las condiciones haciendo que un Atacante/Nodo tenga que esperar también a la cuarta trama *Beacon* para empezar el proceso de Asociación, igualando el comportamiento presentado en el Nodo.

Para esta modificación, a pesar de que la ejecución del atacante se realiza previo a la terminación de la fase de creación de elementos de Asociación de las entidades, existe la posibilidad que el Atacante no logre ingresar a la WBAN para vulnerarla. Todo esto debido a que, si el Atacante/Nodo no recibe a tiempo o simplemente no recibe la cuarta trama *Beacon* no logrará ingresar a la red, produciéndose una Asociación entre entidades Nodo y Hub de manera normal. Los resultados producto de la modificación del Atacante/Nodo se muestran en la Tabla 3.4.

Tabla 3.4. Resultado Caso d.

Ataque ejecutado	Número de veces
Ataque de Hombre en el medio	6
Ataque de Impersonalización de Nodo	4
No Ataque	12
Total	22

La Figura 3.25 muestra que en efecto el porcentaje de ataques no ejecutados cuando el Atacante/Nodo ha sido modificado es evidentemente mayor. Esto pone en consideración que un Atacante no puede ingresar en cualquier tiempo ni en cualquier circunstancia al

sistema para vulnerarlo, tomando en cuenta también que el objetivo del Atacante es poder impersonalizar las entidades en el sistema.

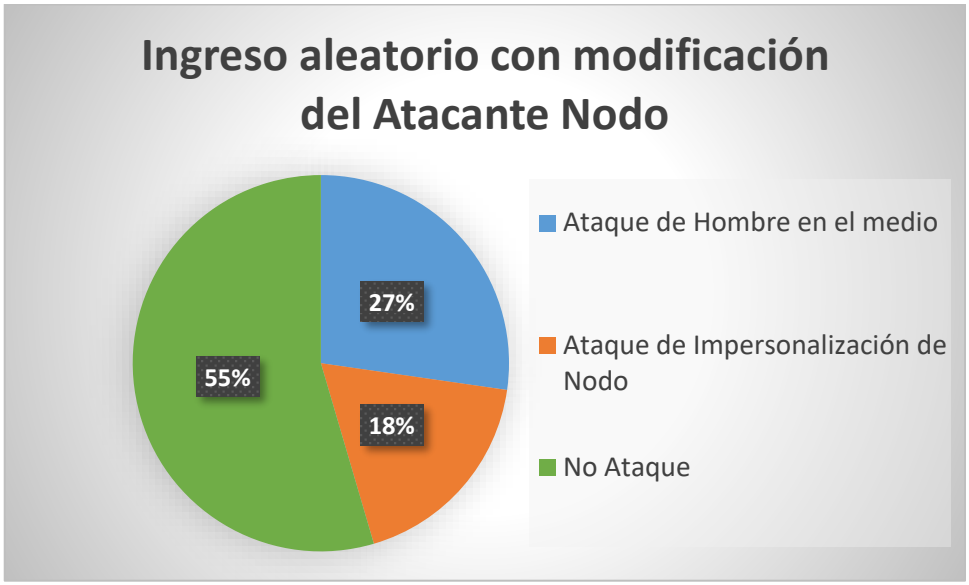


Figura 3.25. Porcentaje de ataques Caso d.

3.4 Pruebas de la Solución

Las soluciones implementadas se basarán en lo analizado en la sección 2.2.5 literal d; esas soluciones como se ha mencionado afectan directamente al cálculo del testigo o *Witness*. La primera solución se basa en la modificación del mensaje *Witness*, mientras que la segunda solución se basa en la modificación de la llave para el *Witness.* Esta sección busca analizar lo que ocurre dentro del proceso de Asociación cuando se han implementado estas distintas soluciones y un atacante desea ingresar a la WBAN. Cabe aclarar que estas soluciones no afectan el proceso de Asociación entre dos entidades cuando no se ha ejecutado un ataque sobre el mismo.

Para las pruebas de la solución implementada se considerarán las mismas pruebas de ataque de las secciones 3.3.1, 3.3.2 y 3.3.3. Tomando en cuenta las mismas premisas, como que, para que un ataque sea posible, un Atacante debe ingresar antes de la fase de creación de elementos de Asociación entre las entidades que buscan asociarse. Las Tablas 3.5 y 3.6 muestran los resultados obtenidos al realizar las pruebas con las dos soluciones implementadas.

Tabla 3.5. Resultados obtenidos en pruebas utilizando solución de modificación de mensaje *Witness.*

Resultados utilizando solución de modificación de mensaje *Witness*			
	Ataque de Hombre en el medio	**Impersonalización de Nodo**	**No ataque**
Nodo – Hub- Atacante	No se ejecuta el ataque debido a la solución implementada.	No aplica.	No aplica.
Nodo – Atacante – Hub	No aplica.	No se ejecuta el ataque debido a la solución implementada.	No aplica.
Ingreso en tiempo aleatorio, Atacante sin modificación	No se ejecuta el ataque debido a la solución implementada.	No se ejecuta el ataque debido a la solución implementada.	No aplica.
Ingreso en tiempo aleatorio, Atacante modificado, envío de tramas en menor tiempo	No se ejecuta el ataque debido a la solución implementada.	No se ejecuta el ataque debido a la solución implementada.	No aplica.
Ingreso en tiempo aleatorio, Atacante modificado, envío de tramas en mayor tiempo	No se ejecuta el ataque debido a la solución implementada.	No se ejecuta el ataque debido a la solución implementada.	No aplica.
Ingreso en tiempo aleatorio, Atacante modificado, Nodo/Atacante modificado	No se ejecuta el ataque debido a la solución implementada.	No se ejecuta el ataque debido a la solución implementada.	Si el atacante no ingresó al sistema existirá probabilidad de que un ataque no se ejecute.

Tabla 3.6. Resultados obtenidos en pruebas utilizando solución de modificación de clave para *Witness.*

Resultados utilizando solución de modificación de clave para *Witness*			
	Ataque de Hombre en el medio	**Impersonalización de Nodo**	**No ataque**
Nodo – Hub- Atacante	No se ejecuta el ataque debido a la solución implementada.	No aplica.	No aplica.
Nodo – Atacante –Hub	No aplica.	No se ejecuta el ataque debido a la solución implementada.	No aplica.
Ingreso en tiempo aleatorio, Atacante sin modificación	No se ejecuta el ataque debido a la solución implementada.	No se ejecuta el ataque debido a la solución implementada.	No aplica.
Ingreso en tiempo aleatorio, Atacante modificado, envío de tramas en menor tiempo	No se ejecuta el ataque debido a la solución implementada.	No se ejecuta el ataque debido a la solución implementada.	No aplica.
Ingreso en tiempo aleatorio, Atacante modificado, envío de tramas en mayor tiempo	No se ejecuta el ataque debido a la solución implementada.	No se ejecuta el ataque debido a la solución implementada	No aplica.
Ingreso en tiempo aleatorio, Atacante modificado, Nodo/Atacante modificado	No se ejecuta el ataque debido a la solución implementada.	No se ejecuta el ataque debido a la solución implementada.	Si el atacante no ingresó al sistema existirá probabilidad de que un ataque no se ejecute.

Como se puede evidenciar en las Tablas 3.5 y 3.6 los comportamientos en las dos soluciones no difieren, todo esto basado en la premisa de que la autenticación se realizará sobre el mismo elemento; por lo tanto, a pesar de ofrecer distinta seguridad el resultado final es el mismo. Tanto para el caso de Ataque de Hombre en el medio e Impersonalización de Nodo se busca que los valores se sigan calculando, a pesar de que

la autenticación en el proceso ha fallado, con el objetivo de evidenciar que las credenciales en efecto serán distintas. Como se analizó en el capítulo 2 de este trabajo, dentro de un Nodo la autenticación en *Witness* no puede ser utilizada debido a que la única entidad que realiza autenticación de *Witness* es el Hub; sin embargo, *Witness* al ser un elemento para la obtención de MK_KMAC en el Nodo, se verá como una falla de autenticación MK_KMAC. Las Figuras 3.26, 3.27 y 3.28 muestran lo que ocurre en las entidades con cualquiera de las soluciones implementadas después ejecutar un proceso de Asociación con ataque.

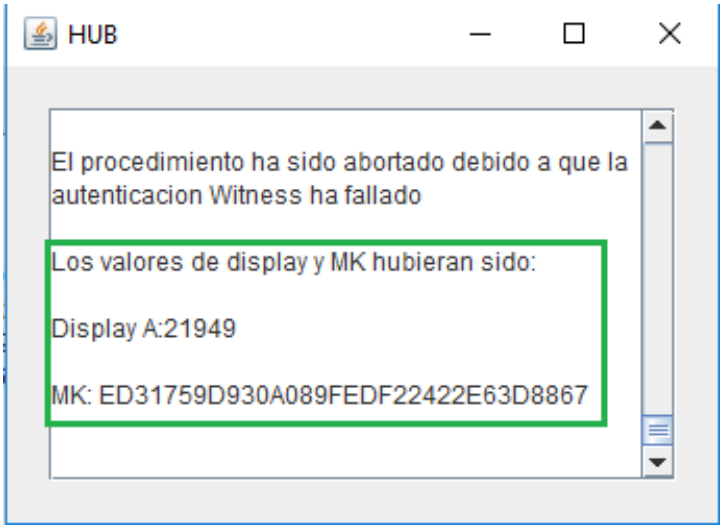


Figura 3.26. Hub falla Asociación por solución implementada.

En la Figura 3.26 se puede observar que la entidad Hub muestra directamente que el proceso de autenticación ha fallado por autenticación de W*itness* y emite los valores que hubiera calculado con la información errónea. La Figura 3.27 no evidencia un claro fallo de autenticación en *Witness*, pero se puede dar un fallo en autenticación MK_KMAC, además de poder corroborar estos resultados con el hecho de que en el Atacante/Hub falla la autenticación y los valores mostrados en *display* no son iguales. La Figura 3.28 muestra tanto el fallo de Atacante/Hub como las credenciales de Atacante/Nodo, que comparadas con la Figura 3.27 se evidencia que los valores son erróneos.

Esto puede dar como conclusión que las soluciones implementadas evitan que un Atacante logre asociarse con las entidades que conforman una WBAN establecida, es decir logra contrarrestar la principal vulnerabilidad detectada en estos sistemas, la cual es no tener la capacidad de detectar si la Asociación se realiza con una entidad correcta o con un Atacante. El evitar la Asociación impide que un Atacante dentro de los procesos de paradigma de seguridad, continúe intercambiando información sensible como se ha

indicado anteriormente. Ya sea en un ataque de Impersonalización de Nodo, como de Ataque de Hombre en el medio, el Atacante creerá que ejecutó una Impersonalización del Nodo, pero en todos los casos se puede evidenciar que las credenciales calculadas *display* y MK son totalmente diferentes. Si se desea realizar una Asociación exitosa será necesario ejecutar nuevamente el proceso de Asociación enviando una nueva primera trama de Asociación por parte del Nodo.

Sin embargo, se podría tener inconvenientes con la distribución de la llave pre compartida si no se establece métodos que aseguren la privacidad y disponibilidad del canal a utilizar. De acuerdo al Estándar IEEE 802.15.6, se puede utilizar un canal fuera de banda para la distribución de esta llave, como lo indica en el protocolo *Public Key hidden Association*. Adicionalmente, otra opción será que los fabricantes puedan establecer estas llaves al momento de la fabricación de los sensores. Finalmente, se podría utilizar cualquier técnica de transmisión *Covert Channe*[14]*l*.

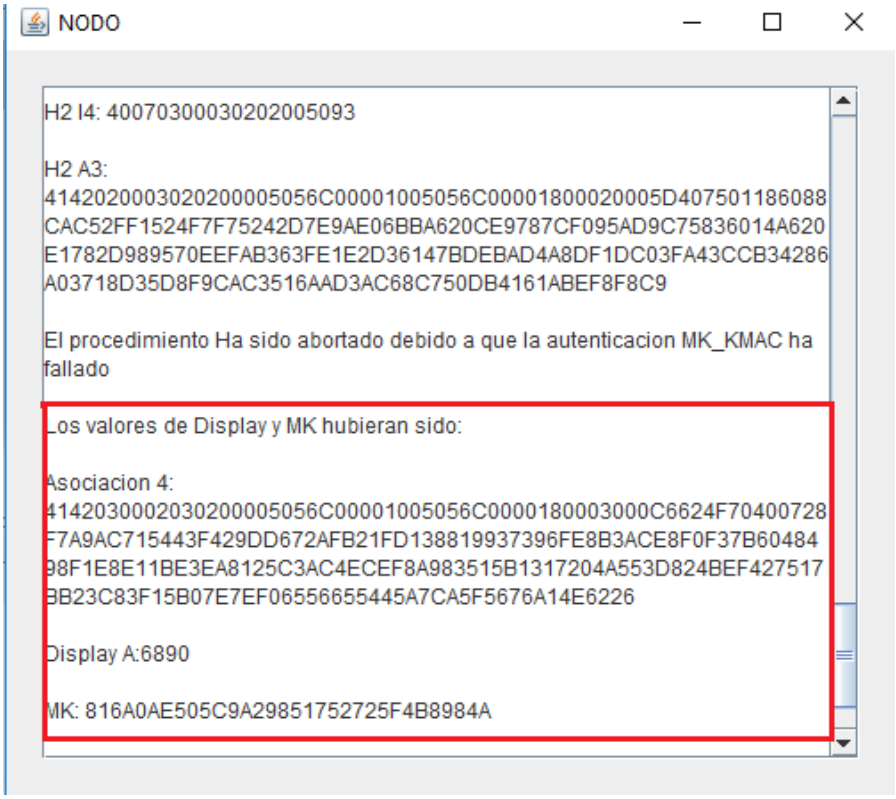


Figura 3.27. Nodo falla Asociación por solución implementada.

[14] **Covert Channel:** Denominado canal encubierto, debido a que es un canal que puede ser utilizado para transferir información de un usuario a otro dentro de un sistema, haciendo uso de medios no destinados para este fin. Es necesario que exista un acuerdo previo a la comunicación que detalle la codificación a utilizar entre ambas partes.

Figura 3.28. Ataque falla por soluciones implementadas en las entidades.

4 CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

Las conclusiones que se detallan a continuación, se enfocan en cuatro aspectos importantes: estado del arte del protocolo DAA, implementación del DAA, resultados de la evaluación de vulnerabilidades en el DAA y resultados obtenidos al implementar las soluciones propuestas.

En cuanto al estado del arte del protocolo DAA se concluye lo siguiente:

- En el estándar IEEE 802.15.6 se afirma que no existen vulnerabilidades dentro del proceso DAA, específicamente vulnerabilidades a Ataque de Hombre en el medio. Sin embargo, en la referencia [2] ya se demuestra teóricamente vulnerabilidades del DAA debido a la posibilidad de impersonalización de entidades y ausencia de seguridad al compartir las llaves públicas que se generan en el inicio del proceso. Por lo tanto, según el estudio realizado, el protocolo DAA no es capaz de mantener la confidencialidad de sus elementos de Asociación (Llaves Públicas PK y Nonce, específicamente); y al no implementar mecanismos más robustos de autenticación de entidades a asociarse, es vulnerable a Impersonalización de entidades, desencadenando un Ataque de Hombre en el medio.

- El estándar IEEE 802.15.6 indica que la forma de autenticación de este método es mediante el valor del *display* mostrado en las interfaces, pero esta no es la única forma con la que el proceso DAA realiza la autenticación, inclusive se puede afirmar que si la autenticación MK_KMAC falla en cualquiera de las entidades la autenticación por *Witness* y *display* fallarán.

De acuerdo a lo analizado en la implementación del protocolo DAA, se concluye que:

- Existe el reto de generar Llaves Públicas PK de 128 bits utilizando el método de criptografía de Curva Elíptica, donde utilizando solamente sumas y duplo de un punto se debe llegar al resultado de SK multiplicado por el punto inicial de la curva G; por lo cual, el algoritmo de iteración implementado encuentra el camino más corto y de menor número de operaciones para dicho cálculo, descomponiendo eficientemente el escalar SK hasta que su valor sea igual a uno. Con el algoritmo desarrollado se

puede disminuir notablemente los requerimientos de procesamiento necesarios para la implementación del DAA.

- Para la implementación de AES la mejor opción es elegir el modo de cifrado CBC sin *Padding*. Esto se debe a que el modo CBC cuenta con un vector de inicialización, lo cual aumenta la seguridad del cifrado, y el relleno de mensajes incompletos (*Padding*) puede ser controlado de mejor manera en el procedimiento posterior AES-CMAC, el cual tiene un manejo particular de mensajes incompletos.

- CMAC tiene gran ventaja sobre el correcto manejo de los mensajes y la partición que puede hacer sobre éstos. La automatización del manejo de los mensajes para CMAC permitió que se implemente una sola clase que abarque cada uno de los casos. La clase CMAC puede tomar tanto mensajes completos, incompletos y únicos obteniendo en todos los casos el resultado correcto de codificación, sin necesidad de que este proceso se realice en la codificación de AES previa.

- El algoritmo de CMAC es el más utilizado en cada una de las fases del proceso de Asociación, un buen entendimiento de su algoritmo evita que el proceso de Asociación se vea afectado; ya que al ser utilizado para el cálculo de los elementos de autenticación *Witness* y MK_KMAC, una incorrecta implementación puede fallar en la fase de autenticación y por ende abortar la Asociación.

- El cálculo del CRC para las tramas de Administración puede estar basado en un conjunto finito de valores posibles, evitando la realización de las divisiones binarias para la obtención del Código de Redundancia Cíclico. Por tal razón, para la implementación de CRC en este trabajo se utilizó un arreglo de 256 valores posibles, lo cual optimiza este proceso.

En referencia a los resultados de la evaluación de vulnerabilidades en el DAA, se puede concluir:

- Cada fase del proceso de Asociación tiene objetivos particulares, para este trabajo se logró demostrar que la principal vulnerabilidad se presenta antes de la culminación de la fase de creación de elementos de Asociación; es decir un Atacante no podrá ingresar al proceso de Asociación cuando la fase de creación de elementos de Asociación de un Nodo y un Hub ha terminado. Debido a esto y analizando situaciones más reales, si un Atacante no detecta a tiempo que existe un proceso de Asociación iniciado, no logrará irrumpir en ese sistema, haciendo menos evidente las vulnerabilidades encontradas en el proceso DAA.

- La principal vulnerabilidad encontrada en este método de Asociación es que una entidad no tiene la capacidad de discriminar si el dispositivo que quiere asociarse con él es una entidad correcta o es un atacante, cosa que no ocurre en el proceso *Password Authenticated Association*, en el cual se basan las soluciones que fueron implementadas en las entidades.

- La creación del Atacante previo a la realización de las pruebas fue pensada con el objetivo de que logre un Ataque de Hombre en el medio, pero se evidenció que la secuencia de ejecución de los diferentes programas puede dar como resultado una denegación de servicio impidiendo una Asociación y desencadenando en Impersonalización de Nodo por parte del Atacante.

- Luego de la realización de seis tipos de pruebas para los ataques, se puede decir que la predisposición en la mayoría de éstos es la de un Ataque de Hombre en el medio, obteniendo en cuatro de los seis casos mayor porcentaje sobre Impersonalización de Nodo.

Finalmente, se tuvo resultados favorables al implementar las soluciones propuestas para mitigar las vulnerabilidades antes evaluadas y demostradas, a partir de estos resultados se establece lo siguiente:

- Las soluciones implementadas obtienen un 100% de efectividad en cuanto a denegar la Asociación de un Atacante con entidades que se encuentran en medio de un proceso de Asociación; pero no consigue que estas entidades puedan culminar normalmente su proceso de Asociación, sin que este no se vea afectado por el intento de ataque, necesitando para el caso de este trabajo volver a ejecutar los programas para que la Asociación se complete exitosamente.

- La idea de realizar una compartición previa de una llave para modificar los parámetros dentro del proceso de Asociación fue tomada del protocolo *Password Authenticated Association*, la compartición de esta llave evita que un atacante que no conozca esta llave logre asociarse con una de estas entidades. La extracción de la llave con la información que se envía a través de las tramas de Asociación es prácticamente imposible, ya que el nivel de seguridad lo entregan el algoritmo CMAC y la encripción AES. De esta manera se mejora la autenticación en el proceso DAA.

4.2 Recomendaciones

A continuación, se detallan recomendaciones generales para ejecutar las pruebas prácticas de lo implementado, oportunidades de mejora de este trabajo, y trabajos futuros que se podrían realizar a partir de éste.

Al momento de realizar las pruebas de la parte práctica de este trabajo, se recomienda:

- Para cada prueba de ataque se debe detener y ejecutar nuevamente el Servidor, ya que éste al momento de finalizar el proceso de Asociación comienza a percibir como fallas en el envío de mensajes, impidiendo visualizar de forma correcta el nuevo proceso con los nuevos mensajes que se envían.
- Las condiciones de red ayudan a lograr una ejecución exitosa de las pruebas, por eso es necesario verificar que las máquinas logran tener conexión, que el puerto a utilizar no esté ocupado por otro proceso, y desactivar *firewalls* únicamente en el transcurso de las pruebas.
- De ser posible contar con máquinas de características de procesador Intel Core i5 de 2.4 GHz y memoria RAM de 4 GB, ya que las pruebas a realizarse y los procesos que llevan dentro estas clases son de gran costo operacional para una máquina.

Según las oportunidades de mejora encontradas en este trabajo, se recomienda:

- Realizar esta implementación utilizando otro lenguaje de programación, ya que Java presenta problemas tanto en el consumo de recursos de una maquina como problemas en el diseño de la interfaz gráfica que en otros lenguajes puede llegar a ser más sencillo. Además, la programación en red para la implementación del modelo Cliente – Servidor puede ser más fácil al utilizar otros lenguajes de programación, por ejemplo, lenguaje C# ejecutado sobre .NET *Framework*.
- Replicar esta implementación haciendo uso de dispositivos *Raspberry PI*, con el fin de contar con recursos de procesamiento limitados y una interfaz de red inalámbrica, para que las condiciones de la red y sus dispositivos se asemejen más a lo que se tendría en una WBAN.

A partir del estudio realizado en este trabajo, surgen temas que se recomienda abordar o implementar, con el fin de que se pueda profundizar en esta área de investigación. Por tal razón se recomienda los siguientes trabajos futuros:

- Realizar pruebas prácticas de vulnerabilidad sobre los otros métodos de Asociación definidos en el Estándar IEEE 802.15.6, los cuales el estándar también los define como invulnerables.
- Desarrollar el proceso de Desasociación segura de entidades, ya que es el único proceso capaz de anular la MK compartida al culminar exitosamente un proceso de Asociación.
- Implementar el proceso de creación de la llave PTK, debido a que una vez que se ha compartido la MK, se deberá crear esta llave temporal para finalmente lograr intercambio seguro de tramas
- Según el objetivo de estudio, en este trabajo se implementa una versión simplificada y suficiente del proceso *Broadcast Beacon*. Se recomienda profundizar en el proceso *Broadcast Beacon*, con el fin de implementar dicho proceso tal como especifica el Estándar IEEE 802.15.6 del Anexo I.
- Investigar e implementar protocolos de comunicación de capa física (PHY) y subcapa de Control de Acceso al Medio (MAC), tal como se especifica en el Estándar IEEE 802.15.6 del Anexo I.
- Finalmente, como se indica en las conclusiones de este documento, las soluciones implementadas evitan que una entidad no deseada se asocie con una de las entidades que se encuentran en medio de un proceso de Asociación; pero, el ataque ejecutado impide que estas continúen su proceso exitosamente, siendo necesario reejecutar el proceso de Asociación. Se recomienda investigar si efectivamente esto sucedería en una WBAN y desencadenaría en un ataque de Denegación de Servicio, y de ser el caso, que soluciones se podrían plantear a este problema.

5 REFERENCIAS BIBLIOGRÁFICAS

[1] ASTRIN and A., “IEEE Standard for Local and metropolitan area networks part 15.6: Wireless Body Area Networks,” *I E EE Std 802.15.6*, 2012.

[2] M. Toorani, “On Vulnerabilities of the Security Association in the IEEE 802.15.6 Standard.” Proceedings of Financial Cryptography and Data Security Workshop, 2015.

[3] X. Huang, D. Liu, and J. Zhang, “An Improved IEEE 802 . 15 . 6 Password Authenticated Association Protocol,” IEEE/CIC ICCC 2015 Symposium on Selected Topics in Communications, 2015.

[4] S. Movassaghi, S. Member, M. Abolhasan, and S. Member, “Wireless Body Area Networks : A Survey,” vol. 16, no. 3, pp. 1658–1686, 2014.

[5] O. Phy *et al.*, “A Comprehensive Survey of Wireless Body Area Networks,” pp. 1065–1094, Springer Science+Business Media, pp. 1065-1094, Agosto 2010.

[6] C. Paar and J. Pelzl, *Understanding cryptography : a textbook for students and practitioners*. Springer, 2009.

[7] C. S. IEEE and ANSI, “IEEE Standard Specifications for Public-Key Cryptography,” *New York IEEE*, 2000.

[8] NIST, “Advanced Encryption Standard (AES), Springfield: Federal Information Processing Standards Publications,” 2001.

[9] J. H. Song, R. Poovendran, J. Lee, and T. Iwata, “The AES-CMAC Algorithm,” no. 4493, pp. 1–20, 2006.

[10] W. J. Savitch, *Absolute Java*, Global Edi. Pearson Education, 2009.

[11] A. W. Astrin, H.-B. Li, and R. Kohno, “Standardization for Body Area Networks,” *IEICE Trans. Commun.*, vol. E92–B, no. 2, pp. 366–372, 2009.

[12] E. Jovanov and A. Milenkovic, “Body Area Networks for Ubiquitous Healthcare Applications : Opportunities and Challenges,” pp. 1245–1254, 2011.

[13] B. Braem and I. Moerman, “A survey on wireless body area networks,” pp. 1–18, 2011.

[14] K. Y.-I. 15-07-0943-00-0ban and undefined 2007, “Channel model for body area

network (BAN)," *ci.nii.ac.jp*.

[15] J. T. Takahiro Aoyagi, "Channel Models for WBANs - NICT," *IEEE P802.15 Wirel. Pers. Area Networks*, pp. 1–57, 2008.

[16] H. Kaschel and H. K. Carcamo, "Redes de Area Corporal Inalámbricos : Requisitos , Desafíos e Interferencias," no. November 2014, 2015.

[17] J. Y. Khan, M. R. Yuce, G. Bulger, and B. Harding, "Wireless Body Area Network (WBAN) Design Techniques and Performance Evaluation," *J. Med. Syst.*, vol. 36, no. 3, pp. 1441–1457, Jun. 2012.

[18] D. Smith, D. Miniutti, … T. L.-I. A. and, and undefined 2013, "Propagation models for body-area networks: A survey and new outlook," *ieeexplore.ieee.org*.

[19] J. Der Derian, "The value of security: Hobbes, Marx, Nietzsche, and Baudrillard," pp. 161–178, Jan. 2009.

[20] W. Stallings, *Fundamentos de seguridad en redes : aplicaciones y estándares*. Pearson/Prentice Hall, 2004.

[21] "Yaksha, an improved system and method for securing communications using split private key asymmetric cryptography," Nov. 1994.

[22] F. Akhter, "A novel Elliptic Curve Cryptography scheme using random sequence," in *2015 International Conference on Computer and Information Engineering (ICCIE)*, 2015, pp. 46–49.

[23] N. Koblitz, *A course in number theory and cryptography*. Springer-Verlag, 1994.

[24] M. Rosing, "Implementing Elliptic Curve Cryptography," *Markov Chain Approach IWINAC ... Part I (Lecture Notes Comput. Sci.*

[25] I. Verbauwhede, *Secure integrated circuits and systems*. Springer, 2010.

[26] T. Güneysu and C. Paar, "Modular Integer Arithmetic for Public Key Cryptography," Springer, Boston, MA, 2010, pp. 3–26.

[27] J. Daemen and V. Rijmen, *The Design of Rijndael : AES - The Advanced Encryption Standard*. Springer Berlin Heidelberg, 2002.

[28] S. Frankel, R. Glenn, and S. Kelly, "The AES-CBC Cipher Algorithm and Its Use with IPsec," Sep. 2003.

[29] A. Pousa, "Algoritmo de cifrado simétrico AES," 2011.

[30] W. E. Burr, “Selecting the advanced encryption standard,” *IEEE Secur. Priv. Mag.*, vol. 1, no. 2, pp. 43–52, Mar. 2003.

[31] S. Frankel, “RFC 3566 - The AES-XCBC-MAC-96 Algorithm and Its Use With IPsec,” 2003.

[32] R. Poovendran, J. Lee, and T. Iwata, “The AES-CMAC Algorithm,” Jun. 2006.

[33] R. Poovendran and J. Lee, “The AES-CMAC-96 Algorithm and Its Use with IPsec,” Jun. 2006.

[34] D. Haskins and E. Allen, “IP Version 6 over PPP.”

[35] G. Bianchi, “Performance analysis of the IEEE 802.11 distributed coordination function,” *IEEE J. Sel. Areas Commun.*, vol. 18, no. 3, pp. 535–547, Mar. 2000.

[36] E. Jovanov *et al.*, “A WBAN System for Ambulatory Monitoring of Physical Activity and Health Status: Applications and Challenges,” in *2005 IEEE Engineering in Medicine and Biology 27th Annual Conference*, 2005, pp. 3810–3813.

ANEXOS

ANEXO I. Estándar IEEE 802.15.6 (Formato digital)

ANEXO II. Código de programación (Formato digital)

ORDEN DE EMPASTADO