UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

Carrera de Ingeniería en Electrónica y Redes de Comunicación

IMPLEMENTACIÓN DE MECANISMO DE SEGURIDAD PARA REDES DE SENSORES INALÁMBRICOS BASADO EN CRIPTOGRAFÍA DE CURVA ELÍPTICA

Trabajo de Grado previo a la obtención del título de:

Ingeniero en Electrónica y Redes de Comunicación

AUTOR: CHRISTOPHER ALEJANDRO ORTEGA CHULDE

DIRECTOR: MSc. FABIÁN GEOVANNY CUZME RODRÍGUEZ

ASESOR: MSc. HERNÁN MAURICIO DOMÍNGUEZ LIMAICO

Ibarra – Ecuador

2023

UNIVERSIDAD TÉCNICA DEL NORTE

BIBLIOTECA UNIVERSITARIA

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

IDENTIFICACION DE LA OBRA

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

DATOS DE CONTACTO			
CÉDULA DE IDENTIDAD:	1004376503		
APELLIDOS Y NOMBRES:	Ortega Chulde Christopher Alejandro		
DIRECCIÓN:	Ibarra – Av. Víctor Manuel Guzmán y 13 de Abril		
EMAIL:	caortegac@utn.edu.ec		
TELÉFONO FIJO:	0997701688	**TELÉFONO MÓVIL:**	0997701688

DATOS DE LA OBRA	
TÍTULO:	IMPLEMENTACIÓN DE UN MECANISMO DE SEGURIDAD PARA REDES DE SENSORES INALÁMBRICOS BASADO EN CRIPTOGRAFÍA DE CURVA ELÍPTICA
AUTOR (ES):	Ortega Chulde Christopher Alejandro
FECHA: DD/MM/AAAA	16/06/2023
SOLO PARA TRABAJOS DE GRADO	
PROGRAMA:	**X PREGRADO** ☐ **POSGRADO**
TITULO POR EL QUE OPTA:	Ingeniero en Electronica Redes de Comunnicacion
ASESOR /DIRECTOR:	**MSc. Fabián Cuzme**

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCAS APLICADAS

CONSTANCIAS

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

Ibarra, a los 16 días del mes de junio de 2023.

Christopher A. Ortega Chulde

AUTOR

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCAS APLICADAS

CERTIFICACION

MAGÍSTER FABIÁN GEOVANNY CUZME RODRÍGUEZ, DIRECTOR DEL PRESENTE TRABAJO DE TITULACIÓN CERTIFICA:

Que el presente trabajo de Titulación **IMPLEMENTACIÓN DE UN MECANISMO DE SEGURIDAD PARA REDES DE SENSORES INALÁMBRICOS BASADO EN CRIPTOGRAFÍA DE CURVA ELÍPTICA**, ha sido desarrollado por el señor **CHRISTOPHER ALEJANDRO ORTEGA CHULDE** bajo mi supervisión.

Es todo cuanto puedo certificar en honor a la verdad.

MSc. Fabián G. Cuzme Rodríguez

DIRECTOR

"Siempre lleva más tiempo de lo esperado,
incluso cuando tienes en cuenta la Ley de Hofstadter."

Hofstadter's Law

DEDICATORIA

*A mi madre **Narciza**, por ser mi motivación desde el día en que nací, tu amor, esfuerzo y apoyo incondicional han sido un pilar fundamental en mi vida; Admiro tu lucha incansable y entrega infinita para brindar un futuro mejor a tus hijos. Gracias a ti, he logrado llegar hasta aquí y convertirme en la persona que soy, es un verdadero honor y privilegio ser tu hijo.*

*A **mi familia** por su apoyo en cada etapa de mi vida, por guiarme por el camino correcto y estar presentes en los momentos más difíciles.*

.

Christopher Ortega

AGRADECIMIENTO

Quiero expresar mi más sincero agradecimiento:

*A **mi madre y hermanos** por su amor y dedicación. Ustedes me han apoyado en todo momento. Nunca podré agradecerles lo suficiente por su amor y apoyo.*

*A mis amigos por haber compartido momentos inolvidables durante la trayectoria universitaria, me siento muy afortunado de conocerlos y de haber creado memorias duraderas con ustedes. En especial, quiero agradecer a **Alder, José Luis, Carla, Jean** y **Marisol**, quienes han sido mis amigos más cercanos y me han apoyado siempre. También, quiero expresar un agradecimiento especial a **Kathy** por todo el apoyo moral que me brindó durante el transcurso de la carrera. Les deseo lo mejor en la vida.*

*A mi director **MSc. Fabian Cuzme** y asesor **MSc. Mauricio Domínguez** por saber guiarme en la realización de este trabajo, su apoyo siempre permanente fue el motivo por el cual pude empezar y finalizar este proyecto.*

*A mi pareja **Jessica** por su apoyo y palabras de aliento; Tu presencia han sido un gran sostén en los momentos más difíciles. Gracias por tu constante apoyo en mis sueños y metas, siempre has estado ahí para impulsarme a seguir adelante y alcanzar mis objetivos.*

*A **G**enny **P**resley **T**ravis por ayudarme a encontrar respuestas y soluciones a los problemas complejos; Gracias por tu dedicación y ayuda incondicional.*

$$\boldsymbol{a} = 2, \boldsymbol{b} = -11, \boldsymbol{p} = 3847, \boldsymbol{K} = 10, \boldsymbol{S} = (343,3206)$$

(929,1651) (2157,1846) (2157,1846) (2882,2626) (3199,1629) (1800,3807) (2882,2626) (1535,2291)
(3199,1629) (177,3049) (1513,920) (2325,827) (3199,1629) (3482,3467) (2168,3304) (177,3049)
(2325,827) (3199,1629) (671,3562) (3199,1629) (2157,1846) (3199,1629) (2157,1846) (2325,827)
(2168,3304) (2882,2626) (2882,2626) (3199,1629) (2168,3304) (2882,2626) (1620,2524) (1215,1497)
(2670,1922) (1215,1497) (2882,2626) (1800,3807) (3465,455) (827,3023)
(2157,1846) (2882,2626) (3199,1629) (2168,3304) (1535,2291) (2882,2626)
(671,3562) (3199,1629) (2882,2626) (2670,1922) (308,1164) (1535,2291)

Christopher Ortega

RESUMEN

La tecnología en constante avance, junto con el Internet de las Cosas (IoT) y 6G, ha impulsado el desarrollo y la implementación de redes de sensores inalámbricos (WSN) en diversos campos, como la monitorización ambiental, la agricultura de precisión, la gestión de infraestructuras y la salud. Sin embargo, el uso de estas redes plantea desafíos en cuanto a la seguridad de la información transmitida y almacenada en ellas. La protección de los datos sensibles se ha convertido en una preocupación fundamental para garantizar la confidencialidad, la integridad y la autenticidad de la información en las WSN.

En este contexto, surge la necesidad de mejorar la seguridad en las redes de sensores inalámbricos mediante la implementación de técnicas criptográficas basadas en curvas elípticas (ECC). El objetivo principal es utilizar la criptografía de curva elíptica para proteger la comunicación y los datos en las WSN, aprovechando su capacidad para proporcionar un nivel adecuado de seguridad con un menor consumo de recursos computacionales y energéticos.

En esta tesis se realiza un estudio exhaustivo de los estándares de eficiencia criptográfica y se utiliza la metodología IEEE29148 para definir los elementos del sistema que los dispositivos de baja potencia deben cumplir para implementar el algoritmo ECC. Este enfoque garantiza que la implementación sea eficiente y cumpla con los requisitos de seguridad necesarios.

Una vez establecidas las bases teóricas, se lleva a cabo la aplicación de la librería desarrollada en diferentes despliegues de WSN utilizando el software OMNeT++. Se implementan los algoritmos ECDH y ElGamal Elíptico para realizar pruebas de intercambio de información, eficiencia de cifrado y descifrado, consumo de energía y tiempo de procesamiento. Estas pruebas permiten evaluar la viabilidad y eficacia de la criptografía de curva elíptica en entornos reales de WSN, demostrando su capacidad para fortalecer la seguridad de las comunicaciones y proteger los datos sensibles transmitidos por los sensores inalámbricos.

Palabras clave: criptografía, curvas elípticas, WSN, IEEE29148, OMNeT++, ECDH, ElGamal Elíptico

.

ABSTRACT

The constantly advancing technology, along with the Internet of Things (IoT) and 6G, has driven the development and implementation of wireless sensor networks (WSNs) in various fields such as environmental monitoring, precision agriculture, infrastructure management, and healthcare. However, the use of these networks poses challenges regarding the security of the transmitted and stored information. Protecting sensitive data has become a fundamental concern to ensure confidentiality, integrity, and authenticity of the information in WSNs.

In this context, the need arises to enhance security in wireless sensor networks through the implementation of cryptographic techniques based on elliptic curves (ECC). The main objective is to use elliptic curve cryptography to protect communication and data in WSNs, leveraging its ability to provide an adequate level of security with lower computational and energy resource consumption.

This thesis conducts a comprehensive study of cryptographic efficiency standards and utilizes the IEEE29148 methodology to define the system elements that low-power devices must comply with to implement the ECC algorithm. This approach guarantees an efficient implementation that meets the necessary security requirements.

Once the theoretical foundations are established, the developed library is applied in different WSN deployments using the OMNeT++ software. The elliptic curve Diffie-Hellman (ECDH) and elliptic curve ElGamal algorithms are implemented to perform tests on information exchange, encryption and decryption efficiency, energy consumption, and processing time. These tests allow evaluating the feasibility and effectiveness of elliptic curve cryptography in real WSN environments, demonstrating its capability to strengthen communication security and protect sensitive data transmitted by wireless sensors.

Keywords: cryptography, elliptic curves, WSN, IEEE29148, OMNeT++, ECDH, Elliptic ElGamal.

ÍNDICE DE CONTENIDO

Índice de Figuras

Índice de Tablas

Índice de Ecuaciones

Índice de Algoritmos

1 CAPÍTULO I. Antecedentes

En este capítulo se hace una descripción del problema, objetivos, alcance y justificación, donde se sustenta la importancia de desarrollar el presente proyecto. La criptografía está presente en todas partes, cuando se realiza una llamada o se envía un mensaje por teléfono, cuando se retira dinero del banco y en muchos de los casos se están aplicando algoritmos criptográficos. Actualmente existen varios algoritmos que proveen seguridad, sin embargo, algunos se vuelven obsoletos debido a los avances en la tecnología. Los últimos años las agencias de seguridad como la NSA1 (National Security Agency) aprueban el uso de ciertos algoritmos, entre ellos, los basados en curvas elípticas ECC (Elliptic curve cryptography). Este Trabajo de Titulación se enfoca en la explicación teórica, y el desarrollo del código de los algoritmos criptográficos enfocados a sensores con limitada potencia de hardware, estos nodos sensores usan curvas elípticas para proteger la información.

1.1 Planteamiento del Problema

Actualmente las redes de sensores inalámbricos (WSN, del inglés Wireless Sensor Network) son una herramienta tecnológica muy valiosa para operar en actividades de campo, recolectar información en tiempo real, realizar un monitoreo automático o incluso operar equipos a distancia. Siendo así, deben cumplir determinadas características para ser confiables y que cumplan con su funcionamiento. (Iacono L. 2010).

Para que una WSN pueda estar formada por decenas de “nodos” (dispositivos de recolección de datos), es fundamental el bajo costo de estos, de lo contrario; la red sería inviable o nada práctica. Actualmente estos dispositivos poseen capacidades reducidas diseñadas por los fabricantes para solventar costos. Algunas limitaciones son: capacidad de cómputo, recursos de procesamiento, memoria, demandas de energía, limitado ancho de banda y baja potencia. Generalmente delegan a otros dispositivos: tareas de gestión de red y enrutamiento de datos (Gateway). Por ello no disponen de un mecanismo de seguridad, debido a la demanda de procesamiento y energía que implica la criptografía estándar en estos dispositivos. (Mellado O. 2013)

Siempre está presente la necesidad de mecanismos que garanticen la autenticación, y confidencialidad, tanto de las comunicaciones, como de los propios dispositivos. Normalmente al diseñar una red WSN las personas no tienen en cuenta esto, especialmente en la

comunicación entre los nodos y el Gateway. Un ejemplo de esto es: los datos médicos que son recogidos por los sensores, o la comunicación que realizan nodos en la recolección de datos. (Guaña J. 2020) La seguridad y la privacidad se convierten en elementos fundamentales debido a que los datos se asocian directamente a una persona o a cierta estructura, Los protocolos de seguridad inalámbrica no sólo evitan que las partes no deseadas se conecten a la red inalámbrica, sino que también encriptan los datos. (Kumar V. 2017)

La implementación de seguridad se realiza con el fin de encontrar un sistema criptográfico ligero que sea ideal y funcional en redes de sensores inalámbricos WSN. Todo esto utilizando criptografía de curva elíptica (ECC) el cual es uno de los métodos criptográficos más potentes que requieren menos esfuerzo computacional y puede ahorrar hasta un 90% de los recursos utilizados por un sistema RSA similar. (Naranjo G. 2013).

1.2 Objetivos

1.2.1 Objetivo General

Implementar un mecanismo de cifrado ligero de datos en una red de sensores inalámbricos basado en criptografía de curva elíptica.

1.2.2 Objetivos Específicos

- Recopilar información de la base bibliográfica, documental y del estado del arte de algoritmos criptográficos ligeros basados en criptografía de curva elíptica y su aplicación en las WSN.
- Establecer requerimientos para el funcionamiento del algoritmo de cifrado de criptografía de curva elíptica en las WSN.
- Definir escenarios de simulación con una WSN que permita demostrar el funcionamiento del algoritmo de cifrado de curva elíptica.
- Analizar los resultados obtenidos de la simulación para considerar la implementación del algoritmo en desarrollos de WSNs.

1.3 Alcance

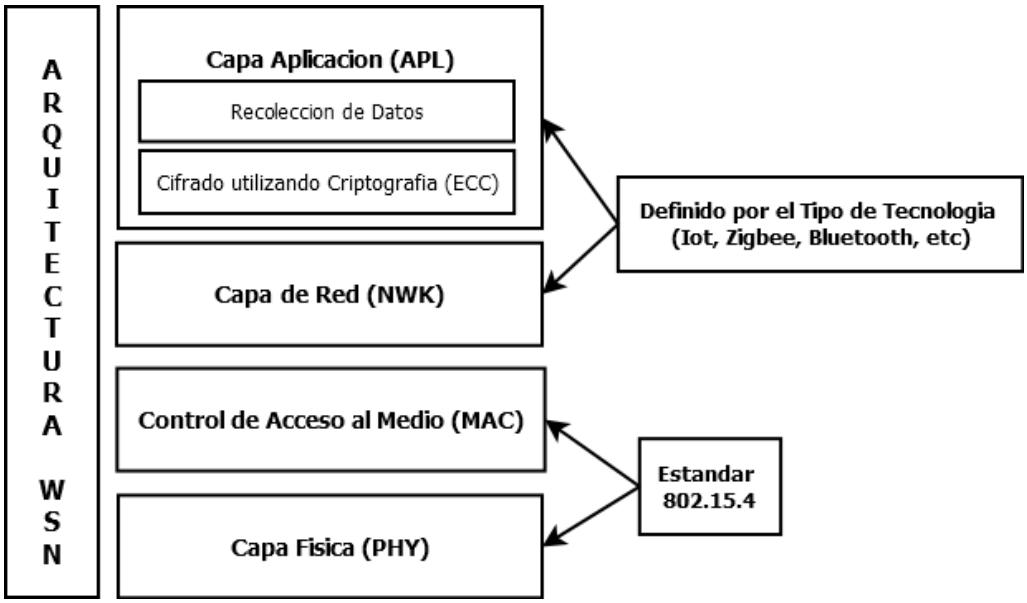
Mediante el siguiente trabajo se pretende implementar un mecanismo criptográfico ligero para dispositivos de bajo rendimiento, basado en matemáticas con curvas elípticas que cumpla requerimientos de seguridad para el despliegue en redes de sensores inalámbricos (WSN). Brindando así tiempos de cifrado más rápidos, generación de claves más cortas y un nivel de seguridad equivalente que los métodos tradicionales como RSA. Para el desarrollo de este, se utilizará la metodología en cascada definiendo cada una de las fases o ciclos de trabajo a realizar y analizando los requerimientos de cada fase de una manera exhaustiva, asegurando así la culminación de etapas anteriores antes del inicio de otras.

Para el cumplimiento del primer objetivo se realiza el levantamiento de la información de forma selectiva sobre el tema objeto de estudio. Realizando dos niveles: primero, desde la perspectiva de la seguridad en redes de sensores Inalámbricos y segundo, dirigida particularmente al tema de la Criptografía con curvas elípticas y sus algoritmos presentes en los últimos años, los mismos que están disponibles en el Estándar para la Eficiencia Criptográfica Versión 2 (SEC 2, del inglés Standards for Efficient Cryptography v2), donde se enumeran parámetros de dominio de curva elíptica en niveles de seguridad comúnmente requeridos para uso de otros estándares ECC como ANSI X9.62, ANSI X9.63, IEEE 1363 e IEEE 1363a. Se utilizará estos parámetros para fomentar la interoperabilidad y la implementación de soluciones basadas en ECC.

En consecuencia, para establecer requerimientos necesarios para el funcionamiento de los algoritmos, se utiliza las versiones de SEC 1 y SEC 2, donde se indican los esquemas criptográficos y los parámetros recomendados basado en criptografía de curva elíptica que dan como resultados la generación de claves de 192 bits, 224 bits, 256 bits, 384 bits y 521 bits. También se analiza los protocolos presentes en la arquitectura WSN, con el fin de implementar estos algoritmos en la capa superior de dicha arquitectura, como se muestra en la siguiente Figura.

Figura 1

Arquitectura de WSN usando ECC



Nota: El gráfico representa las capas de la arquitectura 802.15.4 para redes de sensores inalámbricos.

En contexto a lo anteriormente indicado se establecerán los requerimientos necesarios para determinar el esquema criptográfico de curva elíptica que mejor se adapte a una arquitectura WSN, utilizando la metodología IEEE29148 para definir los elementos del sistema que los dispositivos de bajo rendimiento deben cumplir para implementar el algoritmo ECC (requerimientos mínimos en hardware, memoria, procesamiento, etc.).

La elección de software de simulación también se realiza utilizando IEEE 29148, donde se especifica los procesos requeridos para actividades de ingeniería que dan como resultado productos de software; Se establece tres escenarios de simulación de WSN en los cuales se pretende verificar el funcionamiento del algoritmo ECC: Primero, para la **monitorización ambiental en lugares cerrados**, utilizando sensores para medir la temperatura, luz, estado de puertas, contaminantes en el aire, etc. Ideal para sistemas de alerta temprana (monitorización y prevención de incendios). Segundo, para la **agricultura** donde las WSN pueden ayudar a monitorizar cultivos y controlar los parámetros relevantes (humedad del suelo y temperatura del aire). Tercero, **monitorización de la salud**, donde las WSN recolectan información sobre el estado de un paciente y proveen resultados precisos, más detallados y en tiempo real para la toma de decisiones.

Finalmente, se procederá a las pruebas de funcionamiento en un ambiente simulado y al correspondiente análisis de resultados, realizando la comunicación entre nodos mediante el envío de información cifrada con criptografía de curva elíptica y se valida el funcionamiento de los algoritmos basados en métricas, como son: latencia, el tiempo de cifrado/descifrado, perdidas de paquetes, cantidad de nodos, distancias máximas de comunicación, y su aplicabilidad para diferentes despliegues de WSN.

1.4 Justificación

El mundo del futuro será un mundo conectado, con miles de millones de dispositivos (nodos sensores) interactuando para lograr un objetivo en común, entre los que se encuentran los dispositivos personales compartiendo información, con capacidad para analizar, diagnosticar y tomar decisiones (Cuzme F. 2015). El principal inconveniente y uno de los más importantes es asegurar la privacidad y seguridad de las tecnologías WSN, y conseguir estándares globalmente aceptados que hoy por hoy son demandados en diferentes campos de aplicación, cabe resaltar que uno de los retos planteados por los expertos es básicamente asegurar todos los niveles incluyendo los datos, protocolos y tipos de servicios, labor nada sencilla teniendo en cuenta que las plataformas desarrolladas son bastantes limitadas en recursos (Delgadillo E. 2008).

Como consecuencia en el campo de WSN al utilizar pequeños dispositivos con limitados recursos, trae consigo una amplia gama de nuevos desafíos de seguridad, la principal preocupación se presenta cuando estos “dispositivos limitados” gestionan información potencialmente sensible. (Rosales M., Sánchez G. 2010) La solución: “criptografía ligera” basada en curvas elípticas, esta no es sinónimo de criptografía débil, sino de eficiencia y robustez que ofrecen un compromiso razonable entre seguridad, rendimiento y coste computacional con respecto a la criptografía convencional. Actualmente esta rama de la criptografía está creciendo en uso y parece ser más segura con claves de menor tamaño, aunque se mantiene todavía a la sombra del famoso RSA. (Marrero Y. 2003).

La criptografía de curvas elípticas ha generado un gran impactado en el avance tecnológico, permitiendo la implementación de algoritmos de criptografía basados en problemas como la factorización entera y el problema del logaritmo discreto de una forma eficiente, la criptografía de curvas elípticas desarrollada por Koblitz y Miller en 1985, ha tomado adeptos debido a la

capacidad de proveer la misma funcionalidad comparada contra esquemas criptográficos como el RSA, los niveles de seguridad son bastantes interesantes, una llave de 160 bits en curvas elípticas equivale a una llave de 1024 bits en RSA (Zapata R., 2014).

El presente trabajo propone una alternativa para lograr un nivel confiable de seguridad, usando un algoritmo de cifrado ligero que pueda ser soportado por arquitecturas de hardware de bajo rendimiento, el proceso consiste en la generación de claves para la encriptación de datos sobre las redes de sensores inalámbricos, el cual puede ser aplicado en varios entornos, con el fin de establecer un precedente de seguridad.

1.4.1 Trabajos Relacionados

Guaña J. (2020) "Desarrollo de una aplicación interactiva utilizando algoritmos criptográficos basados en curvas elípticas" Quito, Escuela Politécnica Nacional.

Este Trabajo de Titulación presenta de forma didáctica toda la teoría referente a algunos algoritmos basados en criptografía de curva elíptica y su aplicación práctica queda plasmada en el desarrollo de una aplicación interactiva que consta de los siguientes módulos:

El primer módulo gráfica curvas elípticas sobre números reales, campos finitos primos, operaciones de suma y multiplicación. El segundo módulo muestra el procedimiento para generar un par de claves, privada y pública, mediante criptografía de curva elíptica y también describe el intercambio de estas. El tercer módulo presenta el proceso de cifrado y descifrado con el algoritmo de ElGamal elíptico. El cuarto módulo muestra el procedimiento para generar y verificar firmas digitales generadas a partir de una curva elíptica ingresada por el usuario. El quinto módulo permite a dos usuarios intercambiar, de manera segura, mensajes cortos usando la teoría de criptografía de curva elíptica, revisada en los módulos anteriores. (Guaña J.,2020)

Enríquez D. (2017) "Sistema de voto electrónico con protocolos de curvas elípticas aplicado en elecciones populares" Ibarra [Tesis para la obtención de título en Ingeniería en Sistemas Computacionales]

Este proyecto de tesis plantea una solución a los sistemas de votación tradicional, para ser reemplazados con sistemas de voto electrónico. Se enfoca principalmente al uso de seguridad web, utilizando protocolos de curvas elípticas, así mismo estos son más eficientes que los protocolos tradicionales caducos o más utilizados en sistemas web. Este trabajo de grado va orientado para despejar dudas acerca de los temas relaciones en seguridad web, tipos de encriptación, programación java y base de datos relacionados con el tema principal.

Además, puede considerarse una guía para elecciones populares de diferentes instituciones, ya que el sistema es versátil y se adapta a cualquier entorno. En este trabajo se implementa un algoritmo ECC, que da como resultado la generación de claves de 256 bits, utilizando en un equipo para la encriptación/desencriptación de información relacionada a votos electrónicos.

Castang G. Barbosa C. (2011) "Implementación del criptosistema de curva elíptica en entornos móviles" VINCULOS.

Este artículo presenta el trabajo realizado por los autores para implementar el criptosistema de curva elíptica en las transacciones presentes entre un dispositivo móvil y una aplicación Web. El objetivo de este artículo es explicar el esquema ECIES propuesto por Bouncy Castle para incorporar a nivel de aplicación el algoritmo de cifrado de curva elíptica.

Se desarrolló para el cifrado y descifrado del flujo de datos, y opera bajo un entorno e-commerce en el contexto de transacciones bancarias. El criptosistema de curva elíptica fue definido como opción de añadir seguridad a las aplicaciones dadas sus ventajas frente a otros criptosistemas de clave pública como RSA. Los autores presentan demostraciones del funcionamiento, resultados en base al análisis de tráfico, sus ventajas y desventajas.

Todos estos trabajos dan a conocer la utilidad e importancia de proteger la información utilizando la criptografía de curvas elípticas, así también las diferentes variaciones de ECC para diferentes usos como: cifrar datos (ElGamal EC, ECIES), firmar (ECDSA), y verificar/negociar claves (ECDH). El avance de los diferentes estudios en criptografía con curvas elípticas ha generado la flexibilidad para que las empresas o personas puedan implementar este sistema de seguridad en sus propias aplicaciones. Teniendo en cuenta el estado actual del arte con respecto a la implementación de seguridad criptográfica resulta que la ECC es un orden de magnitud más optimo que RSA y que otros estándares como DSA. Teniendo una gran eficiencia en las implementaciones, rapidez, bajo consumo, reducción de

los tamaños de claves y de la longitud del código transmitido. Siendo ideal para implementar en dispositivos con capacidades limitadas como son los nodos sensores ya que su característica principal es el bajo costo, dimensiones pequeñas y bajo consumo energético. Este trabajo proporciona una explicación clara de la ECC, los diferentes algoritmos utilizados, su implementación en un nodo WSN y las pruebas de funcionamiento en base a la captura de paquetes.

2 CAPITULO II. Fundamento Teórico

En este capítulo se recopila información disponible de bases bibliográficas de los últimos años, donde se ofrece un punto de vista general sobre la seguridad en redes de sensores inalámbricos (WSN), sus aplicaciones y la importancia de la criptografía. Se comenzará introduciendo conceptos básicos de seguridad y su valor en WSN, a continuación, se menciona a la criptografía como solución para proteger información valiosa, los diferentes algoritmos criptográficos de curva elípticas. Finalmente se presentan los softwares de simulación capaces de generar nodos sensores y donde, se pretende implementar la criptografía ligera.

2.1 Seguridad en Redes

Desde los inicios de las redes de comunicación e Internet en los años 60, el tema de seguridad en las redes no era un factor importante, no existía muchos conocimientos para atacar o vulnerar una red. Los primeros usuarios utilizaban estos medios para comunicarse u obtener información, no buscaban afectar en las actividades de otros usuarios. Internet no era un "lugar" seguro, ya que en ese momento no requería serlo.(Forero, 2017) Con el pasar de los años el servicio de Internet dejó de ser exclusivo para las grandes Empresas u organizaciones Gubernamentales; permitiendo así el ingreso a este servicio a otros sectores o personas, abriendo las puertas para poder explorar más allá y generar nuevas amenazas contra la red y la información que se mueve en ella. Desde este punto en 1988, es cuando se comenzó a tomar muy en serio el tema de la seguridad en la red. (Bahillo, 2021)

Para entender la seguridad en las diferentes redes, primero hay que entender cuál es el objetivo de la seguridad informática: "Mantener la Integridad, Disponibilidad, Privacidad, Control y Autenticidad de la información manejada por computadora. Los usuarios deben tener disponibles todos los componentes del sistema cuando así lo deseen". (Stallings, Comunicaciones y Redes de Computadoras, 1997)

a) **Integridad:** La información presente en el sistema permanece inalterada a menos que sean modificada por los usuarios autorizados.
b) **Disponibilidad:** Los usuarios deben tener disponibles toda la información que les competa cuando así lo deseen.
c) **Privacidad:** La información es accesible únicamente por los usuarios autorizados.

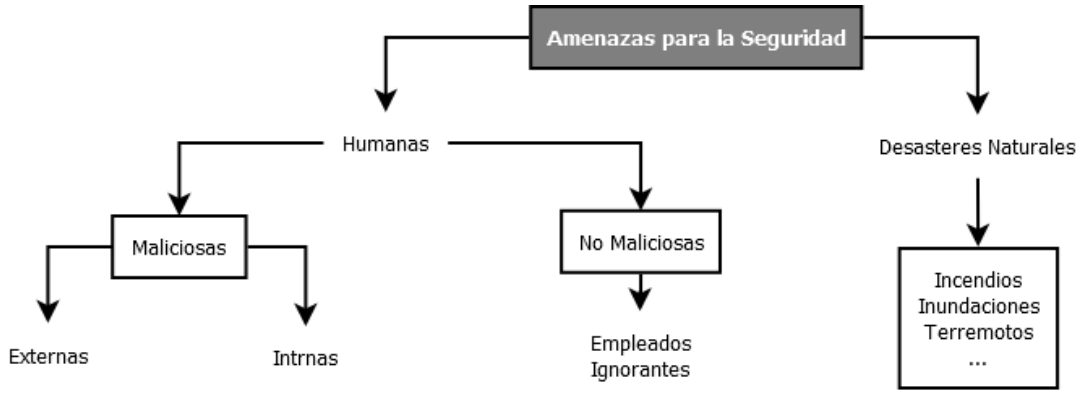
d) **Control:** Solo los usuarios autorizados deciden cuando y como permitir el acceso a la información.
e) **Autenticación:** Definir que la información requerida es válida y utilizable en tiempo, forma y distribución.

Extrapolando esta idea a las redes y al manejo de la información se entiende que el objetivo de la seguridad en redes es: Garantizar que se mantenga la información (de una empresa, entidad gubernamental, persona, etc.) de manera íntegra, privada (confidencial), controlada, autentificada y disponible (solo para el personal autorizado).(Forero, 2017)

La seguridad de redes trata de contrarrestar las amenazas presentes ya sean de tipo humano o natural. En la siguiente **Figura 2** se muestran las diferentes amenazas que se pueden dar en una organización siendo las más comunes las de tipo Humano y las ocasionales las de tipo desastres naturales.

Figura 2

Amenazas en la Seguridad



Nota: Diferentes tipos de amenazas en la seguridad. Adaptado de (Forero, 2017)

2.1.1 Tipos de Seguridad

Actualmente cuando las empresas u organizaciones disponen de sus propias redes internas a las que se brinda acceso a usuarios del exterior, los problemas de seguridad se plantean en cuatro áreas principales. (Forero, 2017) . En la **Figura 3** se muestra la distribución lógica de una red perteneciente a una organización, donde se indican los puntos de seguridad a tomar en cuenta.

1. **La seguridad del perímetro:** Es una línea de defensa importante en una red empresarial y cada organización tiene esta red perimetral, se utiliza como protección frente ataques de exterior generalmente usando **cortafuegos** (**firewalls)**. Un cortafuegos es una de las varias formas de proteger una red de otra red no fiable.

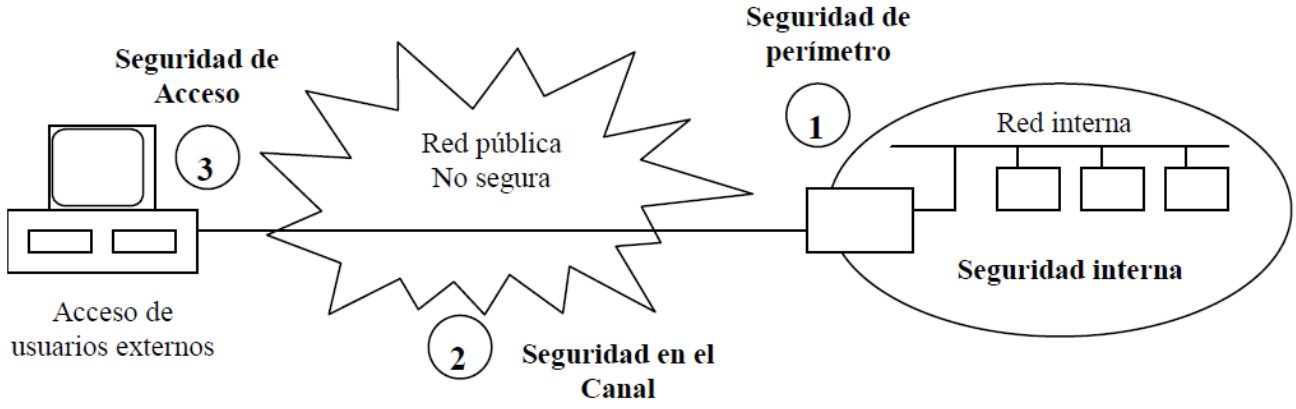
2. **La seguridad en el canal:** La red de comunicaciones siempre es un punto de desconfianza y la prevención contra estos ataques suele pasar siempre por el uso de técnicas de **criptografía** con el objetivo de proteger los datos.

3. **La seguridad de acceso:** Donde se contemplan tres aspectos, la **identificación** del usuario, la **autorización** del acceso y la **auditoría** de las operaciones o tareas realizadas en el sistema por la entidad que ha accedido.

4. **La seguridad Interna:** El problema de seguridad puede aparecer dentro de la propia empresa, provocando por empleados de la empresa o personas maliciosas. En ese caso cobra importancia el uso de técnicas como la **segmentación de la red** (VLANs) mediante el uso de conmutadores (switches), los sistemas de **monitorización de la red** y la **seguridad en servidores**.

Figura 3

Áreas de Seguridad



Nota: En la Figura se observa los diferentes tipos se seguridad que se pueden implementar en una organización, cada área es indispensable para crear una red segura en una empresa u organización.

2.1.2 Importancia de la seguridad

Con el paso de los años, se han hecho grandes esfuerzos, para proteger la información, pero paralelo a esto, personas maliciosas, piratas informáticos o hackers han crecido a la par en conocimientos, ayudados por los rápidos y constantes cambios en la tecnología computacional y por la red de internet. Ahora la información es mucho más sencilla de compartir y poseer, esto permite que muchas más personas tengan a su alcance el conocimiento para ingresar a una red privada y atacarla o ver su contenido siendo ese su objetivo o sólo por curiosidad.

Toda información que se maneje a través de una red inalámbrica o cableada está expuesta a ser vulnerada, los estándares actuales deniegan esto utilizando mecanismos de criptografía para que la información no pueda ser obtenida, ni tampoco vulnerada, esto está pensado para que los dispositivos de una empresa o persona realicen esto de manera automática. ¿Qué sucede cuando los dispositivos no tienen la capacidad para utilizar estos mecanismos de criptografía? Aquí aparece la importancia de la seguridad informática la cual radica esencialmente en el valor de la información, con el avance de la tecnología día tras día, el Internet de las Cosas (IoT) y 6G (*Sexta generación de conectividad móvil su objetivo es reducir aún más la latencia en las conexiones y aumentar notablemente la velocidad de transmisión*), se espera un elevado crecimiento en las redes de sensores inalámbricos, dispositivos personales o para nuestros hogares que no disponen de capacidad computacional para utilizar criptografía pero que son necesarios para el manejo de la información.

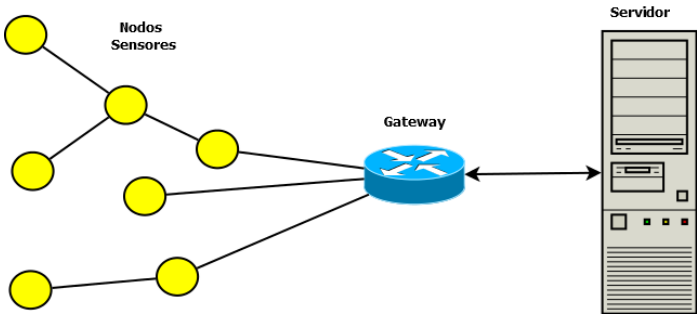
Por estos motivos, en ese entonces y en la actualidad, es requerido crear mecanismos de protección de los datos y lo más importante, se requiere de gente preparada que pueda manejar estos sistemas de seguridad. Por ende, la seguridad informática de las empresas y personas debe estar dirigida a prevenir las amenazas y los riesgos a los sistemas de información, así como los datos que manejan.

2.2 Redes de Sensores Inalámbricos

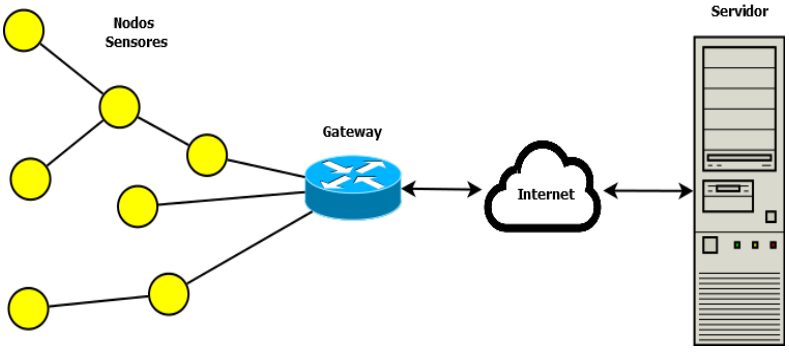
Una red de sensores inalámbricos (WSN - Wireless Sensor Network) es una red compuesta de nodos sensores interconectados entre sí, que intercambian datos a través de un medio inalámbrico, normalmente está compuesta por un gran número de nodos densamente desplegados con el objetivo de abarcar un área específica (el número de dispositivos puede ser del orden de cientos o miles de nodos). En la **Figura 4** se aprecia la estructura de una red de sensores y se da un ejemplo de dos opciones de despliegue. Además, estos dispositivos "nodos sensores" cuentan con reducidas dimensiones, bajo consumo de energía y bajo costo. Estos nodos son dispositivos inalámbricos auto configurables capaces de detectar eventos o capturar señales como por ejemplo contaminación del aire, temperatura ambiente, presión, señales cardíacas, etc.

Figura 4

Estructura de una red de sensores



a) Estructura de una red de sensores (Comunicación directa)



b) Estructura de una red de sensores (Comunicación a través de Internet)

Nota: En las ilustraciones se observa la estructura de una red de sensores inalámbricos de comunicación directa o a través de internet.

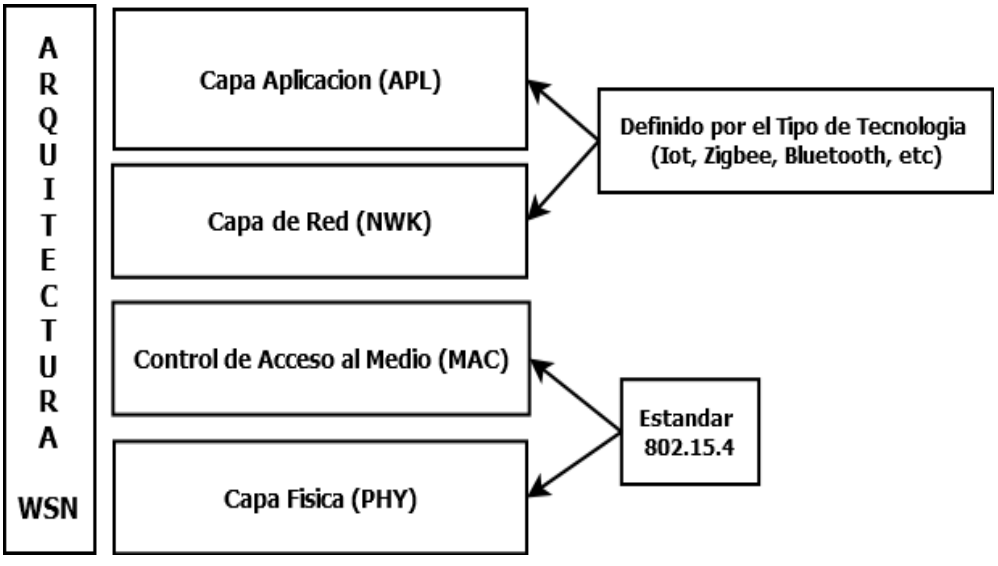
Los nodos se despliegan en distintos ambientes como la tierra, el aire, en vehículos, dentro de las edificaciones, etc., para captar datos y trasmitirlos utilizando un canal de comunicación inalámbrico hasta una estación base como se muestra en la **Figura 4** la estación base recolecta la información de todos los sensores, A su vez, esta información puede ser accedida a través de otro tipo de redes (como por ejemplo Internet) y utilizada para múltiples propósitos como el análisis de datos para tomar decisiones.

2.2.1 Arquitectura

Las aplicaciones de WSN, como todo sistema, tienen una arquitectura que define cómo es la estructura lógica y física de los componentes que la integran y cómo estas interactúan entre sí para lograr el objetivo de dicho sistema, generalmente está compuesta de cuatro capas como se muestra en la **Figura 5**: Capa de aplicación, capa de red y transporte, capa de acceso al medio y capa física.

Figura 5

Arquitectura de una red de sensores



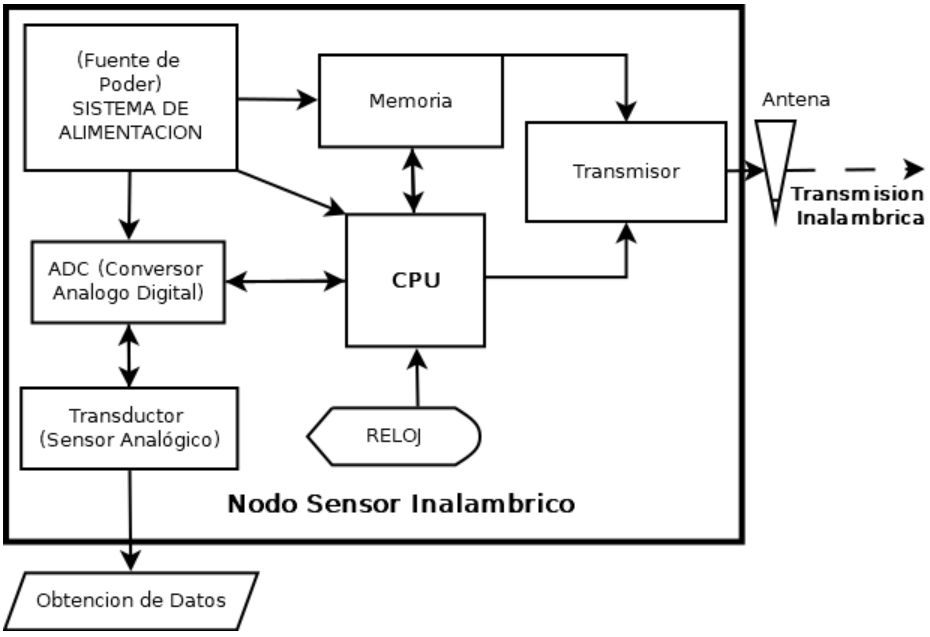
Nota: En la Figura se observa la arquitectura WSN de una red de sensores y sus distintas capas, definidas por un tipo de tecnología.

2.2.2 Elementos de una red de sensores Inalámbrica

Una red de sensores inalámbrica al igual que la red de datos o de comunicaciones tradicional, se compone de diferentes elementos, los cuales le permiten establecer una comunicación entre los nodos que la conforman y acorde con el fin para el cual se haya implementado. Según (Martínez E., 2009), este tipo de redes se componen de nodos sensores, puerta de enlace, estación base y canal de transmisión inalámbrico, etc. En la **Figura 6** se muestra un diagrama de bloques con los elementos que conforman un nodo sensor y como interactúan entre sí para él envió de información inalámbrica.

Figura 6

Nodo sensor inalámbrico



Nota: En la Figura se observa un nodo sensor inalámbrico, con sus diferentes elementos, los cuales trabajan entre sí para cumplir con el objetivo de transmitir información vía inalámbrica.

Transductor: Es el dispositivo capaz de transformar o convertir una determinada manifestación de energía (temperatura, humedad, luminosidad, voltaje, etc.) en otra diferente de salida, pero de valores muy pequeños, este dispositivo se encarga de obtener las señales analógicas y convertirlas en valores analógicos, pero en rangos utilizables.

Conversor Análogo Digital: Convierte las señales analógicas en señales digital, con el propósito de facilitar su procesamiento y hacer la señal resultante más inmune al ruido y otras

interferencias, es decir se encarga de convertir los datos del transductor en valores que pueden ser procesados.

Sistema de alimentación: El encargado de brindar el voltaje y corriente necesarios a todos los dispositivos que componen el sensor, normalmente contiene una batería, un sistema de carga y un estado de sueño.

CPU: Es la unidad central de proceso, es el componente más importante y se encarga de procesar los datos junto con la memoria.

Memoria: Cuando hablamos de "memoria" en los sensores, nos referimos a la memoria de acceso aleatorio (RAM) y se encarga de memorizar o almacena datos informáticos durante algún periodo de tiempo, es indispensable utilizar este recurso.

2.2.3 Aplicaciones de WSN

Las WSN pueden recoger datos de los nodos en tres modos diferentes: **captura de eventos, capturas periódicos** e **informes bajo demanda**. En el modo de ***presentación de informes por captura de eventos***, los nodos sensores detectan e informan datos solo si se produce un evento en su entorno. Por ejemplo, si una WSN se despliega en un bosque para controlar los incendios forestales, los nodos sensores se reportará a la estación base solo si se produce un incendio en el bosque. La ***presentación de informes periódicos*** se utiliza cuando los datos no son urgentes o cuando la información debe ser enviada cada cierto intervalo de tiempo, por ejemplo, un electrocardiograma que monitoriza un paciente. En esta modalidad, el informe de los datos detectados se presenta a la estación base cada período de tiempo predefinidos. En la ***captura bajo demanda***, se envían solicitudes de información a los nodos sensores para que envíen sus datos detectados. Por ejemplo, una aplicación que supervisa contaminantes químicos en el agua puede enviar una petición a los nodos de sensores para detectar el nivel de contaminantes y reportar los valores a la estación base cuando se requiera.

2.2.3.1 Monitorización ambiental en lugares cerrados

La capacidad de utilizar un sensor para medir la temperatura, la luz, el estado de puertas y ventanas, los flujos y la contaminación del aire puede ser utilizada para realizar un óptimo

control de ambientes cerrados. Se pueden utilizar sensores para ayudar en el uso de calentadores, ventiladores, entre otros equipos, de una manera razonable para evitar gastos innecesarios, Además las redes de sensores inalámbricas pueden ayudar en procesos de conducción de vehículos donde se recopile información del entorno. En la **Figura 7** se muestran ejemplos gráficos de las redes antes mencionadas.

Figura 7

Sensores para la monitorización en lugares cerrados

Nota: Las siguientes imágenes se definen como: **a)** Monitorización de Puertas con WSN; **b)** Sensor de movimiento y encendido de las luces WSN para encendido automático; **c)** WSN para monitorizar estados en un vehículo.

Hoy en día, otras aplicaciones en las que se pueden utilizar sensores son la detección de incendios, los detectores de humo son comunes en edificios y permiten disparar alertas. Sin embargo, una red de sensores inalámbricos (WSN, por sus siglas en inglés) podría en estos casos guiar a las personas atrapadas en el incendio a través de la ruta de escape más segura haciendo uso de luces en el techo y paredes u otro método. Por otro lado, una WSN podría ser integrada con otros componentes, como por ejemplo un sistema de información geográfica, para obtener más información sobre la situación real. Otra aplicación podría ser en la construcción o en la monitorización de edificaciones antiguas, donde la WSN podría obtener información estructural.

2.2.3.2 Monitorización ambiental en lugares abiertos (aire libre)

Otra área de aplicación de las WSN es la monitorización ambiental. Por ejemplo, para el rastreo de animales, para recabar información acerca de sus condiciones de vida y su comportamiento. El objetivo de la monitorización en animales es colocar sensores en distintas especies de animales terrestres para comprender la interacción e influencia de estas, comprender patrones de migración de los animales salvajes y como estos son afectados por el cambio climático y la influencia humana. También se utilizan estas redes, para el rastreo de animales en peligro de extinción. En la **Figura 8** se muestran ejemplos gráficos del uso de WSN en la monitorización de animales.

Figura 8

Monitorización ambiental en animales

Nota: En las siguientes imágenes se observa el rastreo de animales salvajes y en peligro de extinción. **a)** Se utiliza collares con diferentes sensores que monitorizan al animal, lo más utilizado es detectar la posición, la comunicación se realiza a grandes distancias; **b)** Los sensores se insertan directamente en la piel de los animales, evitando que se desprendan, normalmente se trata de dispositivos compactos con todos los componentes y comunicación acústica para redes de sensores inalámbricos bajo el agua (UWSN).

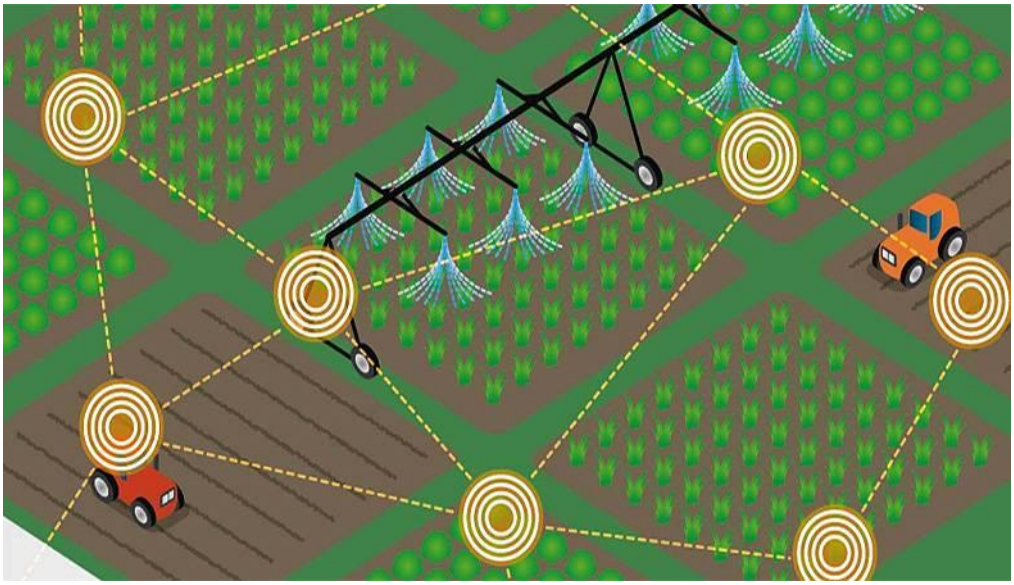
Otras aplicaciones relacionadas con la monitorización ambiental que se han desarrollado son aplicaciones para observaciones del medio ambiente, estudio de fenómenos naturales y pronóstico del clima.

2.2.3.3 Agricultura

Desplegar WSNs en la agricultura puede mejorar la eficiencia en el crecimiento de los cultivos, mejorar el trabajo, reducir los costos, reducir su impacto ambiental y aumentar la calidad de los productos. Las WSN pueden ayudar a monitorizar campos, viñedos y huertas, ayudando a los agricultores a prevenir daños en sus cosechas y aumentando la producción. La automatización en la agricultura provoca una contribución fundamental a lo que hoy se conoce como agricultura de precisión. En la **Figura 9** se muestra un ejemplo de agricultura de precisión que cumple con la utilización de una WSN capaz de controlar los parámetros relevantes (por ejemplo, humedad del suelo y temperatura del aire) y transmitir estos datos de forma inalámbrica a la ubicación agricultor para tomar las medidas adecuadas o integrándolo, por ejemplo, con un sistema de riego donde se libere la cantidad adecuada de agua para incrementar las cosechas basándose en los datos obtenidos.

Figura 9

WSN en la agricultura



Nota: En la Figura se observa la utilización de WSN en la agricultura, los sensores monitorizan las características actuales de los cultivos, esa información se transmite de manera inalámbrica, es procesada y el sistema toma las medidas adecuadas.

Falcone F. (2018) explica “Las prestaciones de este tipo de redes de sensores inalámbricos en entornos de agricultura inteligente dependen, de manera directa, tanto del entorno (tipos de

cultivos, terreno, etc.), como del propio diseño de los transceptores de comunicaciones, es decir, de los dispositivos que cuentan con un emisor y un receptor".

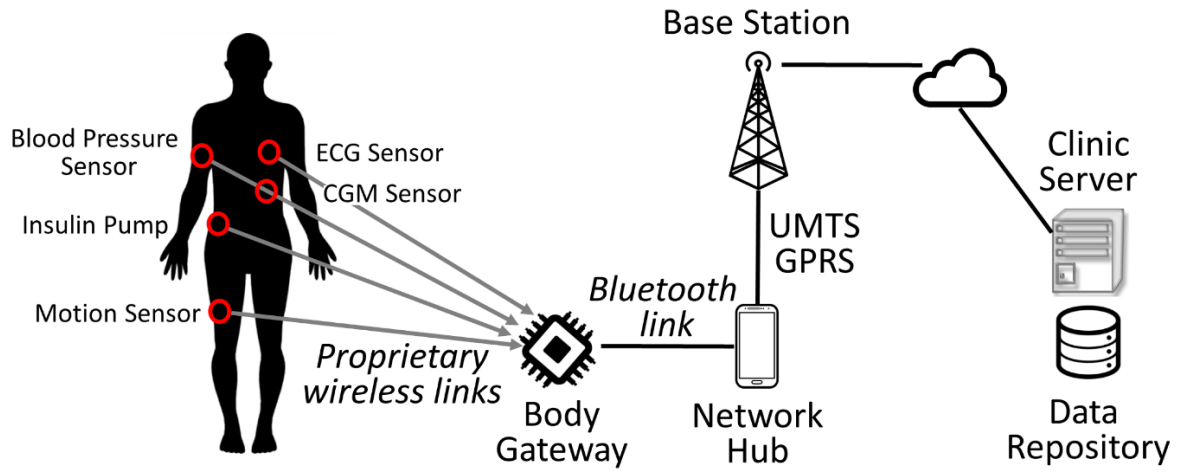
2.2.3.4 Monitorización de la Salud

Los sistemas para el cuidado de la salud e investigaciones científicas en el área médica se pueden beneficiar de las WSN ya que proveen información precisa, más detallada y en tiempo real para la toma de decisiones. Una de las aplicaciones particulares dentro del sector salud, recibe el nombre de BAN (Body Area Network) A gran escala, las redes WBAN pueden ser clasificadas como de uso médico y de uso no-médico.

En las redes WBAN de uso médico, se incluyen los sensores corporales utilizados para el monitoreo en tiempo real, de los signos vitales, de los pacientes, así como para procesos de terapias y rehabilitación, entre otros. En cambio, dentro de las redes WBAN de uso no-médico están las redes para monitoreo de animales, ciencia e investigación, aplicaciones militares, juegos y entretenimiento, monitoreo deportivo, entre otras, convirtiéndose en una herramienta de comunicaciones adaptable a las distintas necesidades de los usuarios. En la **Figura 10** se observa la topología, forma de conexión en el paciente, tipos de sensores y la distribución de la red para la obtención de datos de un paciente.

Figura 10

Red de área corporal inalámbrica



Nota: La Figura nos muestra los diferentes sensores adaptados a una persona para monitorizar su salud, esa información es enviada a través de medios inalámbricos a una base de datos.

Las WBAN son fundamentales para los propósitos de la medicina moderna y cuidado de la salud presente y futuro. En la actualidad se puede ver estos sensores adaptados en relojes inteligentes los cuales monitorean la salud del paciente, la almacenan y la procesan. Se espera que en el futuro este tipo de tecnología este más presente en las personas, previniendo así problemas de salud y mejorando la calidad de vida.

2.3 Seguridad en Redes de Sensores Inalámbricos

La seguridad en las redes de sensores ha sido estudiada de forma extensa por la comunidad científica, y aunque aún existen problemas de seguridad que deben ser resueltos (p. ej. manejo de nodos móviles, delegación de privilegios, privacidad, agentes seguros, actualización del código, etc.), actualmente es posible crear una red de sensores que cumpla un conjunto básico de propiedades de seguridad.

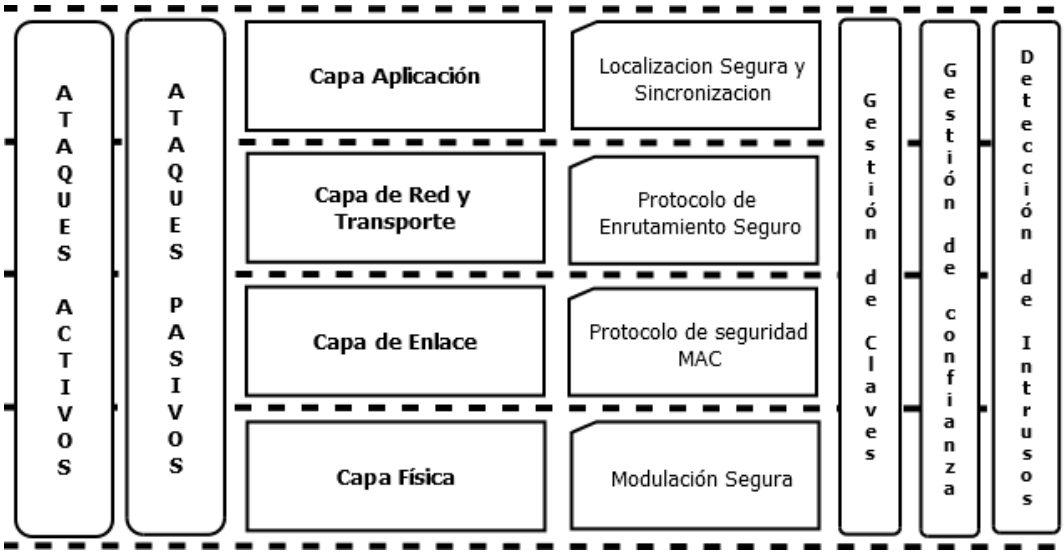
Los nodos sensores han sido normalmente considerados como dispositivos demasiado restringidos para soportar criptografía de clave pública, pero esta suposición ha cambiado utilizando criptografía de curvas elípticas (ECC), es posible tener soporte para cifrar datos, firmar, verificar, y negociar claves (ECDH) en un nodo sensor.

2.3.1 Arquitectura de seguridad en WSN

Los WSN son vulnerables a diversas amenazas y ataques. Como se muestra en la **Figura 11**, la arquitectura de seguridad de las redes de sensores inalámbricos se puede dividir en cuatro capas: capa física, capa de enlace, capa de red y capa de aplicación. Los problemas de seguridad incluyen principalmente: administración de claves, detección de intrusiones, administración de confianza, localización segura, sincronización segura y seguridad de enrutamiento. (Daian, 2018)

Figura 11

Arquitectura de seguridad en WSN



Nota: En la Figura se observa la arquitectura de seguridad de las redes de sensores inalámbricos y sus cuatro capas, adaptado de la arquitectura 802.15.4.

Detección de intrusiones: Estos mecanismos son para detectar, identificar y aislar a intrusos internos o externos de la red. Sin embargo, los mecanismos de detección de intrusiones suelen funcionar después de que los ataques maliciosos surtan efecto y se hayan descubierto. Es difícil detectar intrusos maliciosos en la primera vez que los ataques produjeron efecto. Alternativamente, los mecanismos de tolerancia a intrusiones se pueden utilizar para proteger las redes al tiempo que permiten la existencia de intrusos maliciosos.

Gestión de confianza: Los mecanismos de administración de confianza existentes se pueden clasificar en tres categorías: esquema centralizado, esquema distribuido y esquema jerárquico. En el esquema centralizado, un nodo raíz o una estación base suministran la gestión de confianza para cada nodo de la red. En el esquema distribuido, cada nodo necesita calcular y mantener los valores de confianza de toda la red. Los esquemas jerárquicos son apropiados para topologías basadas en clústeres que se utiliza ampliamente en WSN de gran cantidad de nodos.

Gestión de claves: Los principales objetivos de la criptografía y la administración de claves son la confidencialidad, la autenticación, la integridad y la no denuncia. La criptografía permite almacenar o entregar información confidencial en redes no seguras, para que no pueda ser leída

o modificada por usuarios no autorizados. Los mecanismos de cifrado y administración de claves para los WSN consumen ancho de banda y energía, y hacen que estos primitivos no sean adecuados para WSN con recursos muy limitados.

2.3.2 Ataques de seguridad en WSN

El tipo de ataque depende de las intenciones que tenga el atacante, estos ataques y amenazas se pueden clasificar en ataques pasivos y ataques activos; dentro de los activos se encuentran los ataques de denegación de servicio (DoS) y pueden estar presente en cualquier parte de la pila de protocolos.

Ataques pasivos: se definen como el número de intentos que realiza los nodos atacantes de percibir la naturaleza de las actividades y obtener los datos transmitidos en la red sin interrumpir la operación. El atacante puede predecir la naturaleza de la comunicación, analizando el tráfico de paquetes, observar el intercambio de paquetes, identificar los hosts que se comunican y determinar la ubicación de nodos, sin embargo, es difícil detectar ataques pasivos ya que no se ve afectado el funcionamiento de la red. (Catalán Gallach, 2019).

Ataques activos: Los ataques activos se definen por ser aquellos que alteran, elimina, destruyen los datos transmitidos por la red. Los ataques activos son capaces de modificar los datos o eliminarlos para interrumpir el funcionamiento de la red. Si los ataques realizados vienen de nodos externos a la red se llaman ataques externos, que son mucho más fáciles de detectar y defender. Por el contrario, si un ataque proviene de un nodo interno este puede causar severos daños en la red. En esta categoría también se puede encontrar a los:

> **Ataques DoS:** estos pueden interrumpir la comunicación, afectar a la cooperación entre los nodos y disminuir la disponibilidad de toda la red. Los ataques DoS son de bajo costo, mortales y lo que es peor, difíciles de detectar y defender.

2.3.3 Ataques de Seguridad Basados en la pila de protocolos

2.3.3.1 Capa física

La capa física de las WSN se encarga de la comunicación inalámbrica entre los nodos de la red. Y se pueden presentar varios ataques como: Ataques pasivos, Ataques DoS, ataques de espionaje y los ataques de interferencia.

- **Ataques de interferencia**: Es donde uno o varios nodos maliciosos emiten constantemente señales inútiles o no deseadas y, por lo tanto, se interferirá con la comunicación normal con otros nodos legítimos en la región afectada. El nodo malicioso puede paralizar la red atacando a los nodos especiales, el nodo raíz o estación base.

2.3.3.2 Capa enlace

En esta capa se presentan varios ataques DoS que se pueden dar en la capa para consumir la batería de los nodos legítimos o perturbar el funcionamiento normal de MAC, incluyendo el ataque de interferencia, ataque de colisión, el ataque de agotamiento y el ataque de negación del sueño.

- **Ataque de colisión:** Para el mecanismo de establecimiento de comunicación RTS/CTS, no debe transmitir paquetes durante el periodo de tiempo, pero un nodo malicioso puede violar el mecanismo y transmitir paquetes incluso después de detectar un paquete CTS, provocando colisión en el receptor e invalidando el paquete, Una forma de defensa ante esto es código de corrección de errores.
- **Ataque de negación de sueño:** Sirve para evitar que el nodo entre en modo de suspensión. El atacante podría optar por ejecutar un ataque de denegación de sueño, en este tipo de ataques puede deshabilitar la red permanentemente puede llevar meses agotar las baterías del nodo objetivo. Un ataque de negación de sueño más inteligente podría agotar las baterías en pocas semanas o días. Un atacante de negación de sueño puede ser difícil de detectar.

- **Ataque de agujero de gusano:** En un ataque de agujero de gusano se crea un enlace fuera de grupo formado por el oponente entre dos lugares físicos con menor retardo y mayor ancho de banda que el enlace común. En este ataque el nodo atacante mueve algunos paquetes reconocidos de un extremo a otro usando enlaces fuera de banda y los reinyecta en el sistema en un diferente punto. Por lo tanto, al diseñar un algoritmo de localización, se debe considerar este efecto de ataque.

2.3.3.3 Capa Red

La capa de red encargada de enrutar paquetes del nodo origen al nodo destino, esta capa es vulnerable a amenazas o ataques que tienen como objetivo perturbar el enrutamiento de la red, incluyendo el ataque de repetición, ataque de reenvió selectivo, negligencia y codicia, ataque de desvió, etc.

- **Ataque de reenvió selectivo:** Los nodos maliciosos se comportan como nodos legítimos, eliminan paquetes selectivamente y se niegan a reenviar los paquetes recibidos. Sin embargo, existe el riesgo de que el nodo vecino encuentre otras rutas para enrutar el paquete al nodo destino. Este ataque se puede detectar usando mecanismos de vigilancia predefinido para WSN, que tiene reglas predefinidas para generar alertas de intrusión. Con este mecanismo los ataques pueden detectarse permitiendo que los nodos escuchen a los nodos del siguiente salto en una transmisión de radiodifusión.
- **Ataque de desvió:** El atacante reenvía paquetes a rutas incorrectas modificando rutas o dirigiendo paquetes incorrectamente a un nodo malintencionado. Este ataque puede protegerse modificando la ruta de ruta que consta de rutas de origen en cada paquete. La autorización, el filtrado de salida, el enrutamiento con reconocimiento de confianza y la supervisión de rutas son técnicas factibles para defenderse de un ataque de desvió.
- **Ataque de sumidero:** Es un ataque de agujero negro particular que evita que el nodo sumidero legitimo obtenga los datos transmitidos por los nodos de sensores, causando serias amenazas a los protocolos y aplicaciones de capa superior.

2.3.3.4 Capa Transporte

La capa de transporte de las WSN es responsable del transporte confiable de paquetes. Los ataques DoS típicos en esta capa incluyen el ataque de desincronización, el ataque de inundación de sincronización, etc.

- **Ataque de desincronización:** En un ataque de desincronización, un atacante malintencionado interrumpe las conexiones activas entre nodos transmitiendo paquetes falsificados con números de secuencia falsos o indicadores de control que desincronizan los puntos finales. Un atacante malintencionado puede falsificar mensajes con números de secuencia incorrectos para perturbar la sincronización entre nodos, lo que puede afectar la precisión de los relojes sincronizados y la eficiencia de las operaciones programadas.
- **Ataque de inundación de sincronización:** Cuando se requiere un protocolo para mantener un estado, se vuelve vulnerable al agotamiento de la memoria a través de la inundación. Un atacante puede realizar repetidamente nuevas solicitudes de conexión hasta que los recursos requeridos por cada conexión se agoten o alcancen un límite máximo.
- **Ataque de G. Sybil:** Un atacante puede pretender estar varios puntos a la vez con múltiples personalidades. Debido a esto los Protocolos de Enrutamiento son engañados. Para solucionar este problema se utilizan métodos de autenticación y verificación de posición.
- **Ataque de inundación "Hello":** En caso de ataque de la inundación "hello" un nodo malicioso envía el paquete de saludo, y el nodo de recepción puede presumir que el opositor es el vecino. Si el oponente utiliza alta potencia para la transmisión, entonces esta suposición es falsa. Para defenderse de este ataque la confirmación de enlace bidireccional puede ayudar.
- **Suplantación de Acknowledgement:** Un paquete enviado al nodo vecino puede ser capturado por el nodo vengativo y puede utilizar este mensaje para suplantar el ACK, con el objetivo de convertir el enlace en un vínculo débil o un vínculo en un área geográfica de sombra. Esto da lugar a la falla de la conectividad y a la alta tasa de error de bits. Una respuesta a este incendio sería el cifrado de todos los paquetes transmitidos a través del sistema de comunicación.

2.3.4 Mecanismos de seguridad para WSN

La seguridad es uno de los temas más importantes en las redes de sensores inalámbricos, debido a la imposibilidad de utilizar los mecanismos convencionales contra ataques, ya que los sensores cuentan con recursos de procesamiento y almacenamiento limitados. No es posible formular un mecanismo infalible, es necesario realizar estudios, actualizaciones y evoluciones de los algoritmos que logren contrarrestar los continuos ataques emergentes (Aranzazu C., 2009).

A continuación, se resumen los principales mecanismos de seguridad de las redes de sensores inalámbricos:

2.3.4.1 Manejo de claves y cifrado

Gran parte de las implementaciones de seguridad en WSN utilizan encriptación y esquemas de manejo de claves. El principal problema por solucionar es lograr que el esquema de manejo sea suficientemente eficiente, de tal forma que, si un intruso logra capturar un nodo, no le sea posible acceder a todas las claves de la red y por tanto a la información confidencial del sistema.

2.3.4.2 Autenticación

La autenticación es un punto fundamental en la seguridad de las redes de sensores. Esto se debe a que un atacante puede clonar un nodo o sustraer la información de las claves de la red y enviar información maliciosa a la red. Es necesario entonces generar mecanismos que permitan a los nodos reconocer que la información recibida es auténtica, mediante la validación de la identidad del nodo transmisor. El mensaje de autenticación más utilizado es el MAC, el cual contiene diferente información que justifica la legitimidad de un nodo. (Aranzazu C., 2009)

2.3.4.3 Detección de intrusos

Debido al despliegue de los sensores en áreas abiertas, donde es posible que un atacante capture un nodo y acceda a toda su información, fácilmente se presentan ataques por clonación, por intrusos o DoS. A los nodos comprometidos se les suele denominar “Moles”. Se puede recurrir a dos soluciones como:

- ***Monitoreo de nodos*** - Para el monitoreo de la red, varios autores postulan el concepto de que los mejores candidatos para proteger el sistema y por ende a los sensores, son los mismos nodos.
- ***Sistema de detección de intrusos*** - Cuando los recursos de los nodos no son tan limitados, es posible utilizar un software especial para la detección de intrusos, denominado IDS (Intrusion Detection System).

2.3.4.4 Otros métodos

Diferentes autores desarrollan mecanismos donde se utilizan conceptos como votación, la huella dactilar del nodo, la huella social, árboles jerárquicos, entre otros, que permitan detectar e inclusive revocar nodos corruptos, todos estos métodos dependen del tipo de red, el nivel de seguridad que se requiera, así como el tipo de Hardware. (Aranzazu C., 2009).

2.4 Criptografía

La criptografía proviene de las palabras griegas *Kriptos* (ocultar) y *graphos* (escritura). Literalmente significa “escritura oculta” y es la ciencia que estudia el cifrado/descifrado y codificación de los datos con la finalidad de hacerlos ilegibles a personas y terceros para los que no está destinado el mensaje. Esta técnica se utiliza en el arte, la tecnología y la ciencia para conseguir que los mensajes sean confidenciales. (Three Points, 2021). Esta ciencia se remonta de la época romana hasta nuestros días, la cual es básica para mantener en secreto nuestros datos cuando usamos Internet, correo o comercio electrónico.

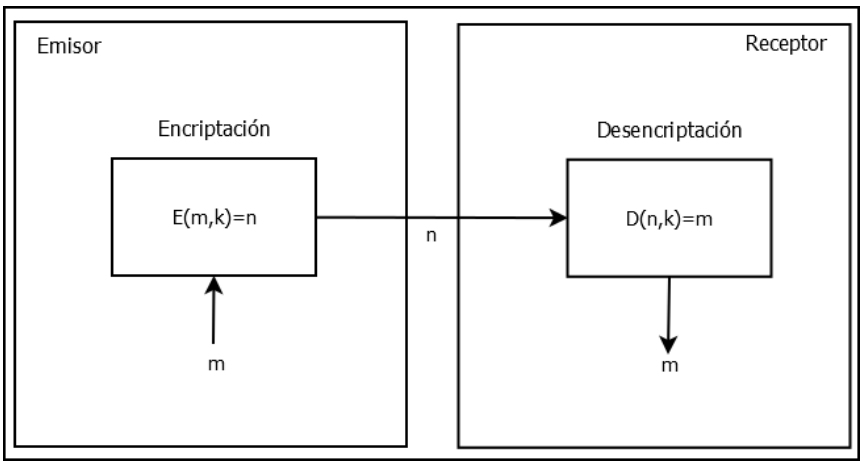
Para realizar una comunicación entre dos usuarios (persona o servidor), es necesario establecer un protocolo de actuación. Este protocolo se aplica en función de las necesidades a cumplir y puede ser simétrica o asimétrica.

2.4.1 Criptografía Simétrica (Clave Privada)

La criptografía simétrica utiliza la misma clave para cifrar y descifrar el mensaje de datos, es decir se basa en un secreto compartido. Es por esta razón que la seguridad de este proceso depende de la posibilidad de que una persona no autorizada consiga la clave de sesión o clave secreta. (Mendoza, 2019) En la **Figura 12** se indica el proceso de cifrado simétrico el cual consiste en aplicar una función matemática a un mensaje para hacerlo ilegible para cualquier persona que no tenga la clave privada. La desencriptación es el proceso inverso, en el que se aplica la función inversa para recuperar el mensaje original.

Figura 12

Encriptación simétrica



Nota: En la Figura se observa la representación de la criptografía simétrica, la cual utiliza la misma contraseña "k" para la encriptación y desencriptación de datos.

Donde:

- $\boldsymbol{m}$ → Mensaje para trasmitir
- $\boldsymbol{n}$ → Mensaje Encriptado
- $\boldsymbol{k}$ → Clave privada
- $\boldsymbol{E(m,k) = n}$ → Función para la Encriptación
- $\boldsymbol{D(n,k) = m}$ → Función para la Desencriptación

2.4.2 Criptografía Asimétrica (Clave Pública y Privada)

Los algoritmos asimétricos son diferentes a los simétricos en un sentido muy importante. Cuando se genera una clave simétrica, simplemente se escoge un número aleatorio de la longitud apropiada, a diferencia de las claves asimétricas en las cuales el proceso es más complejo. Los algoritmos asimétricos se llaman así debido a que en lugar de usar una sola clave para realizar la codificación y la decodificación, se utilizan dos claves diferentes: una para cifrar y otra para descifrar. Estas dos claves se encuentran asociadas matemáticamente, cuya característica fundamental es que una clave no puede descifrar lo que cifra. (Mendoza, 2019)

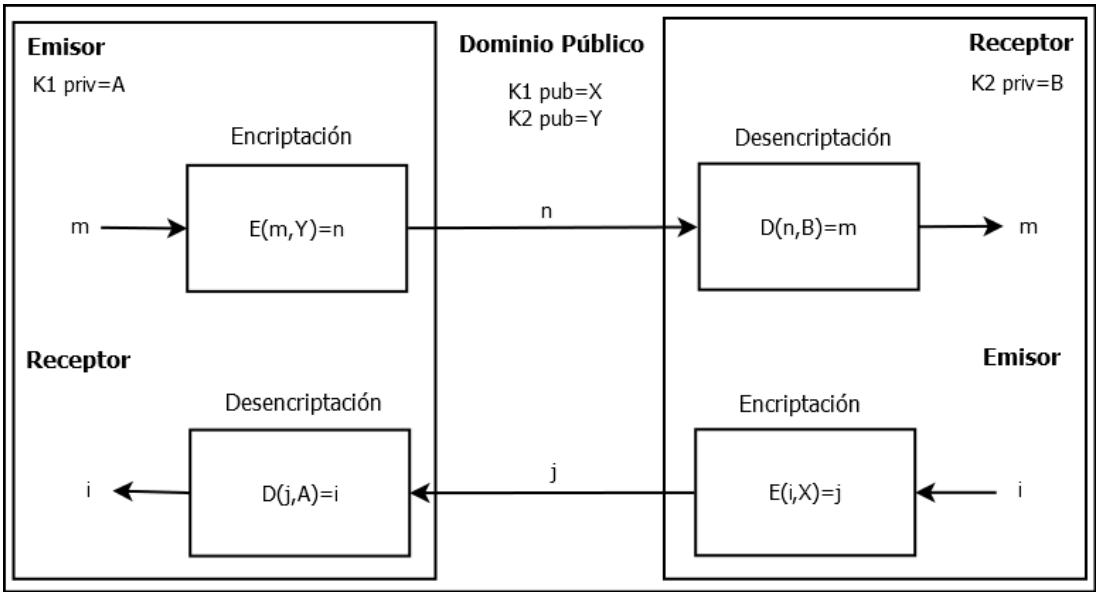
Cuando se completa la generación de una clave asimétrica se define una clave de cifrado (clave pública) y una clave de descifrado (clave privada); la primera puede ser conocida por todo el mundo, pero se debe tener mucho cuidado en ocultar la clave privada. Las claves asimétricas

tienen la sorprendente propiedad de que lo que se está cifrando con una clave sólo se puede descifrar con la otra. En la **Figura 13** se indica el proceso de cifrado donde un usuario puede encriptar un mensaje utilizando la clave pública de otro usuario, y solo ese usuario podrá descifrar el mensaje utilizando su clave privada correspondiente. Esto hace que la criptografía asimétrica sea muy segura, ya que incluso si alguien intercepta el mensaje encriptado, no podrá descifrarlo sin acceso a la clave privada correspondiente.

RSA es el primer y más utilizado algoritmo de este tipo, fue inventado y publicado en 1978 por tres investigadores del MIT. Rivest, Shamir y Adleman (De ahí proviene su nombre) puede cifrar un mensaje de máximo 116 bytes (58 caracteres) y se basa en la factorización de dos números primos.

Figura 13

Encriptación asimétrica



Nota: En la Figura se representa a la criptografía de clave privada, donde tanto el emisor como el receptor utilizan su propio par de claves, para generar un nivel alta seguridad.

Donde:

- $\boldsymbol{m}, \boldsymbol{i} \rightarrow$ Menajes para transmitir
- $\boldsymbol{n}, \boldsymbol{j} \rightarrow$ Mensajes Encriptados
- $\boldsymbol{A}, \boldsymbol{B} \rightarrow$ Claves privadas (Utilizadas para desencriptar mensajes)
- $\boldsymbol{X}, \boldsymbol{Y} \rightarrow$ Claves públicas (Utilizadas para encriptar mensajes)
- $\boldsymbol{E}(\boldsymbol{m}, \boldsymbol{k}) = \boldsymbol{n} \rightarrow$ Función para la Encriptación
- $\boldsymbol{D}(\boldsymbol{n}, \boldsymbol{k}) = \boldsymbol{m} \rightarrow$ Función para la Desencriptación

2.4.3 Criptografía con Curvas Elípticas

Las curvas elípticas (EC) son curvas matemáticas que surgen al graficar **ecuaciones de segundo grado**[1] en dos dimensiones. Al tomar una ecuación de segundo grado y graficarla en un plano cartesiano (dos dimensiones), se obtiene una curva que puede ser una parábola, una elipse o una hipérbola; si se restringen ciertos valores de la ecuación, se puede obtener una curva elíptica.

Las curvas elípticas tienen propiedades interesantes en términos de simetría, áreas y longitudes de arcos, y se pueden utilizar para resolver problemas en diversas áreas de las matemáticas, como la geometría, el álgebra y la teoría de números. En la criptografía asimétrica, las curvas elípticas se utilizan para generar claves públicas y privadas de manera segura.

En 1985, Neal Koblitz y Victor Miller proponen simultáneamente y de manera independiente que es posible montar sistemas criptográficos de clave pública y privada (Criptografía asimétrica) utilizando curvas elípticas. De manera simplificada la idea es proponer que el sistema criptográfico trabaje sobre un conjunto de puntos de una curva elíptica y no sobre un campo finito. En sus inicios no se le tomo la importancia que merecía dicho descubrimiento, con el pasar de los años los matemáticos se dieron cuenta que implementar este cifrado resultaba muy útil para minimizar tamaños de clave y mantener un nivel de seguridad optimo. Desde entonces las Curvas Elípticas es uno de los campos de mayor interés en las matemáticas y la Geometría Algebraica.

2.4.3.1 Expresión matemática de las curvas elípticas

De acuerdo con (Stallings, Cryptography and Network Security Principles and Practices, Fourth Edition, 2005) una curva elíptica (***E***) sobre un cuerpo ***K*** es una curva algebraica sin puntos singulares, lo que significa que no tiene puntos que se comporten de manera anormal o singular. viene definida por **(Ec. 1**) denominada la ecuación de Weierstrass normalizada.

$$E: y^2 + a_1xy + a_3y = x^3 + a_2x2 + a_4x + a_6, a_i \in K \qquad \text{(Ec. 1)}$$

[1] *Una ecuación de segundo grado es una ecuación matemática que tiene la forma:* $\boldsymbol{ax^2 + bx + c = 0}$ *donde* $\boldsymbol{a, b}$ *y* $\boldsymbol{c}$ *son constantes y* $\boldsymbol{x}$ *es una variable*

Donde $\boldsymbol{x}$ y $\boldsymbol{y}$ son variables, mientras que los términos $\boldsymbol{a_1, a_2, a_3, a_4}$, y $\boldsymbol{a_6}$ son constantes que deben pertenecer a $\boldsymbol{K}$.

En algunos contextos, la ecuación normalizada de Weierstrass puede ser difícil de manejar debido a la complejidad de la ecuación y a la cantidad de cálculos matemáticos que implica. Por ejemplo, para resolver esta ecuación, es necesario utilizar cálculo avanzado y conocimientos de álgebra y teoría de números. Mediante un proceso de reducción y usando **transformaciones lineales**[2] se pueden eliminar términos y hacer que la ecuación sea más simple y fácil de manejar. Convirtiendo asi la **ecuación normalizada** en la **ecuación reducida** de Weierstrass **(Ec. 2)**, donde $\boldsymbol{a}$ y $\boldsymbol{b}$ son los parámetros que definen a la curva. La única condición es que si la característica del cuerpo $\boldsymbol{K}$ es distinta de 2 y de 3, la ecuación de la curva $\boldsymbol{E}$ se puede expresar como la ecuación reducida, caso contrario se utiliza la ecuación normalizada.

$$E: y^2 = x^3 + ax + b \qquad \text{(Ec. 2)}$$

La ecuación reducida permite representar todo tipo de curvas elípticas, siempre y cuando cumplan con ciertas condiciones de "regularidad" y se comporten de manera similar a una elipse en geometría, esto significa que la curva no debe tener *cúspides* (Punta o extremo de la curva), no debe cruzarse a sí misma y no debe comportarse de manera anormal. Es posible establecer estas restricciones formalmente con la **(Ec. 3).**

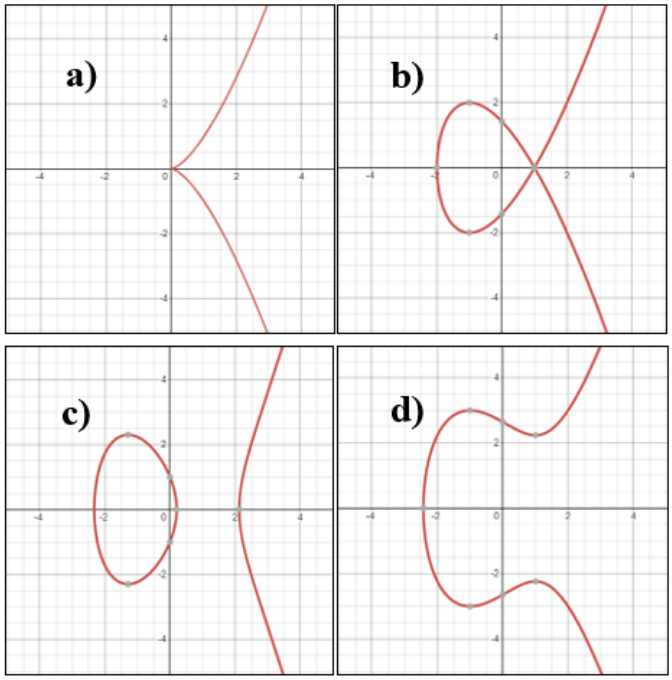
$$4a^3 + 27b^2 > 0 \qquad \text{(Ec. 3)}$$

En la **Figura 14** se muestra algunos ejemplos de curvas elípticas utilizando la **(Ec. 2)**, donde se modifican los valores correspondientes de $\boldsymbol{a}$ y $\boldsymbol{b}$, sin tomar en cuenta las restricciones de la **(Ec. 3)**.

[2] Las transformaciones lineales son operaciones matemáticas que se aplican a una o más variables de una ecuación, tienen la propiedad de que preservan la estructura lineal, haciendo posible utilizarlas en el proceso de reducción de ecuaciones para simplificarlas y conservar sus propiedades.

Figura 14

Diferentes curvas elípticas



Nota: La figura nos indica los tipos de curvas elípticas que podríamos tener, por ejemplo: **a)** $\boldsymbol{y^2 = x^3}$ es singular ya que tiene una cúspide; **b)** $\boldsymbol{y^2 = x^3 - 3x + 2}$ es singular ya que exhibe un punto de cruce; **c)** $\boldsymbol{y^2 = x^3 - 5x + 2}$ no es singular, tiene dos componentes y tres puntos en el eje de las abscisas; **d)** $\boldsymbol{y^2 = x^3 - 3x + 7}$ no es singular, tiene un solo componente y un punto en el eje de las abscisas. Solo se puede trabajar con los 2 últimos tipos de curvas elípticas.

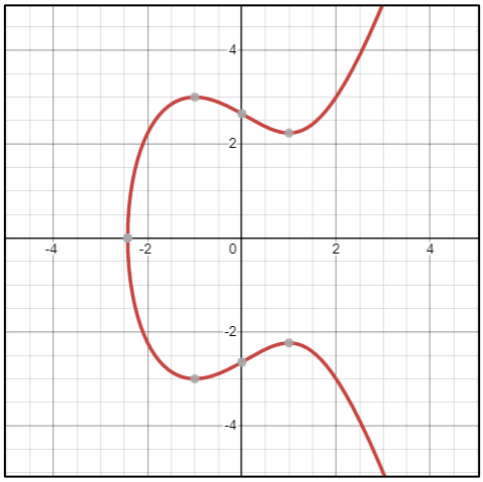
La curva elíptica $\boldsymbol{y^2 = x^3 + ax + b}$ debe estar definida sobre un cuerpo $\boldsymbol{K}$, la misma se puede representar en diferentes campos como:

2.4.3.2 Curvas Elípticas en un conjunto de números reales $\mathbb{R}$

Los valores de la curva no pueden utilizarse para criptografía en este conjunto debido a la ambigüedad de trabajar con números decimales y a la perdida de información al redondear dichos números. En la **Figura 15** se representa gráficamente una curva elíptica en este conjunto y aunque es posible trabajar con curvas elípticas sobre conjuntos de números reales $\mathbb{R}$, en la criptografía de curvas elípticas se utilizan principalmente sobre conjuntos de números finitos

Figura 15

Definir la ecuación sobre el conjunto de números reales R



Nota: En la Figura se observa la curva elíptica $\boldsymbol{y^2 = x^3 - 3x + 7}$ representada en el conjunto de números reales.

2.4.3.3 **Curvas Elípticas sobre cuerpos finitos primos $\mathbb{Z}_p$**

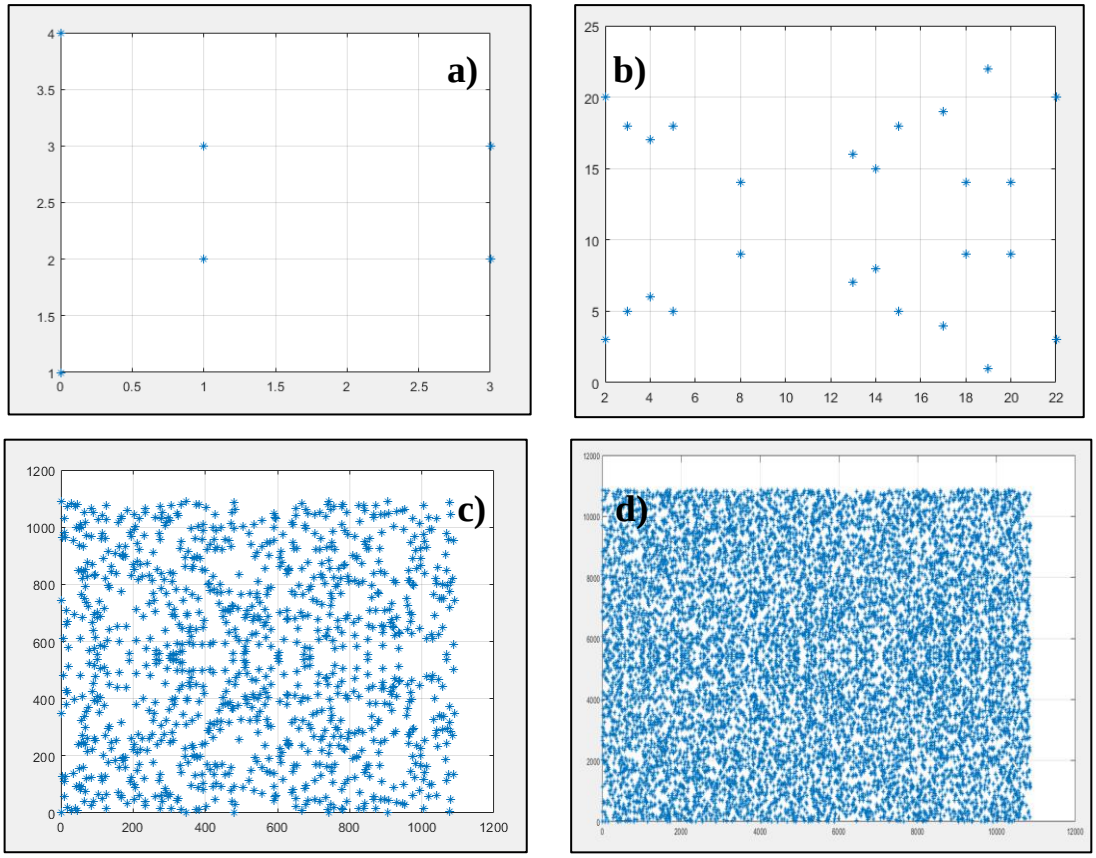
Las curvas para criptografía no usan números que se acerquen al infinito, debido a la capacidad limitada de los procesadores actuales, por eso, su rango se limita dentro de un cuerpo o campo finito primo, es decir, un conjunto con un número finito de elementos.

Un cuerpo finito $\mathbb{Z}_p$ con $\boldsymbol{p}$ elementos donde $\boldsymbol{p}$ debe ser un número primo $\boldsymbol{Z_p = Z/pZ}$, este grupo se utiliza en criptografía convencional con curvas elípticas, ya que se crea un conjunto de operaciones y un conjunto de puntos, dando como resultado un grupo finito cíclico donde se pueden crear variaciones de criptografía.

El uso de curvas elípticas sobre conjuntos de números finitos permite evitar ciertos ataques de seguridad que podrían ser posibles cuando se trabaja con curvas elípticas sobre conjuntos de números reales. En la **Figura 16** se muestra como es una curva elíptica expresada en un campo $\mathbb{Z}_p$ primo, no se puede determinar cuántos elementos tiene la curva $\boldsymbol{E}(\mathbb{Z}_p)$, y en general es un problema difícil de calcular ya que este depende totalmente de la curva, y no de $\mathbb{Z}_p$, es decir variando los valores a los parámetros de la curva $(\boldsymbol{a}\ y\ \boldsymbol{b})$ la curva generara diferente cantidad de puntos.

Figura 16

Definir la ecuación sobre cuerpos finitos, Ejemplo: $a)\ Z_5,\ b)\ Z_{23},\ c)\ Z_{1091}\ y\ d)\ Z_{10859}$



Nota: En la Figura se observa la curva elíptica $\boldsymbol{y^2 = x^3 - 3x + 7}$ representada en el conjunto finito de puntos $\boldsymbol{Z_5}$ donde se aprecia que la cantidad de puntos donde existe la curva es de **6** elementos, por otro lado para valores superiores como el conjunto finito de puntos $\boldsymbol{Z_{10859}}$ la cantidad de puntos no se puede calcular, solo se puede utilizar el **Teorema de Hasse** para encontrar la cantidad aproximada, más adelante se explica este teorema en el apartado **Parámetros de la Ecuación de la curva elíptica**, además nótese como la representación en un campo de números finitos primos es *unidimensional* ya que se utiliza un solo número para identificar un punto en el espacio, mientras que la representación en una curva elíptica es bidimensional ya que se necesitan dos números, uno para la coordenada **x** y otro para la coordenada **y**, con el fin de identificar un punto perteneciente a la curva.

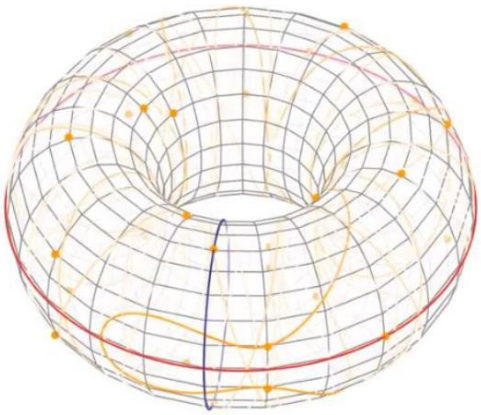
2.4.3.4 Curvas Elípticas sobre cuerpos de torsión

Las curvas elípticas sobre **cuerpos de torsión**[3] son una forma de representación de dichas curvas en el campo de la criptografía, la torsión de las curvas nos ayuda a la adición de un conjunto finito de puntos al final de la curva con el objetivo de mejorar su seguridad. Estos puntos de torsión se añaden para proteger la información cifrada de ataques específicos y para mejorar la eficiencia de los algoritmos de cifrado basados en curvas elípticas. El problema de trabajar en estos campos radica en estudiar los posibles grupos de torsión, esto implica estudiar otros objetos, llamados curvas modulares, y concretamente, saber cuándo estas curvas tienen puntos.

Este tipo de campo se utiliza para Criptografía Cuántica. Es importante destacar que la criptografía cuántica aún se encuentra en sus primeras etapas de desarrollo y que todavía hay muchos desafíos tecnológicos y científicos por resolver antes de que sea posible implementarla a gran escala. Sin embargo, es un área de investigación importante para garantizar la seguridad de la información en un mundo cada vez más tecnológico y conectado. Se espera que tenga aplicaciones en áreas como la criptografía, la seguridad de la información y la comunicación cuántica. En la **Figura 17** se muestra cómo se representa una curva elíptica en un cuerpo de torsión “en este caso un toroide”, la línea amarilla representa la curva y al final de esta se observa la adición de puntos.

Figura 17

Definir la ecuación sobre cuerpos de torsión.



Nota: En la Figura se observa un **toroide** y como se representa la curva elíptica más un conjunto finito de puntos, logrando un nivel mucho más complejo y completo de seguridad en criptografía, la figura ha sido generada en el software de MATLAB.

2.5 Matemática Modular

En criptografía y computación uno de los principales problemas es que cuerpos como los números reales o racionales son infinitos. Por este motivo, se trabaja con conjuntos cerrados de elementos finitos, utilizando como solución la aritmética modular ($\boldsymbol{A.Mod}$), esta proporciona un sistema donde cualquier número queda representado en un grupo, se le llama a veces aritmética de reloj ya que, al llegar a un cierto número da la vuelta y empieza de nuevo.

La Aritmética Modular se define dentro de un conjunto $\mathbb{Z}\boldsymbol{p}$ y es representado por el operador $\boldsymbol{mod}$ o tambien por el símbolo %, esta operación determina el residuo de una división de dos números enteros. Los pasos para desarrollar el cálculo de la aritmética modular son:

1. Dados dos números enteros $\boldsymbol{A}$ (un numero entero) y $\boldsymbol{B}$ (el módulo), se desea calcular ($\boldsymbol{A\ mod\ B}$) o ($\boldsymbol{A\%B}$).
2. Primero, se divide $\boldsymbol{A}$ entre $\boldsymbol{B}$ y se encuentra el cociente $\boldsymbol{C}$ y el residuo $\boldsymbol{R}$.
3. El resultado de la operación $\boldsymbol{A\ mod\ B}$ es el residuo $\boldsymbol{R}$.

Por ejemplo, para calcular $\mathbf{7}\ \boldsymbol{mod}\ \mathbf{3}$, se divide $\mathbf{7}$ entre $\mathbf{3}$ y se obtiene un cociente de $\mathbf{2}$ y un residuo de $\mathbf{1}$. Por lo tanto, $\mathbf{7}\ \boldsymbol{mod}\ \mathbf{3}\ =\ \mathbf{1}$.

Tambien, en la aritmética modular se pueden realizar las operaciones matemáticas básicas como *suma*, *resta*, *multiplicación* y *división*, donde los resultados siempre se encuentran dentro del conjunto $\boldsymbol{Zp}$ definido por números enteros empezando por el $\mathbf{1}$ y siendo el último $\boldsymbol{p}-\mathbf{1}$: $\mathbb{Z}\boldsymbol{p} = \{\mathbf{1}, \mathbf{2}, \mathbf{3}, \dots, \boldsymbol{p}-\mathbf{1}\}$ Al trabajar con estos conjuntos siempre se excluye el $\mathbf{0}$, esto se hace para tener elemento neutro en la operación de la multiplicación y también para asegurar que exista un inverso multiplicativo.

- **Suma:** Dados dos números enteros $\boldsymbol{A}$ y $\boldsymbol{B}$, y un módulo $\boldsymbol{p}$, se desea calcular $(\boldsymbol{A}+\boldsymbol{B})\ \boldsymbol{mod}\ \boldsymbol{p}$. Primero, se realiza la suma de $\boldsymbol{A}$ y $\boldsymbol{B}$, luego se divide el resultado entre $\boldsymbol{p}$ y se toma el residuo como el resultado de la operación. Por ejemplo: se desea calcular $(\mathbf{3}+\mathbf{4})\ \boldsymbol{mod}\ \mathbf{5}$, primero se suman $(\mathbf{3}+\mathbf{4}) = \mathbf{7}$, luego se divide $\mathbf{7}$ entre $\mathbf{5}$ y se obtiene un residuo de $\mathbf{2}$, por lo tanto $(\mathbf{3}+\mathbf{4})\ \boldsymbol{mod}\ \mathbf{5} = \mathbf{2}$.

- **Resta:** La resta en aritmética modular se realiza de la misma manera que en aritmética lineal, buscando un elemento congruente al resultado final, dados dos números enteros

$\boldsymbol{A}$ y $\boldsymbol{B}$, y un módulo $\boldsymbol{p}$, se desea calcular $(\boldsymbol{A}-\boldsymbol{B})\ \boldsymbol{mod}\ \boldsymbol{p}$. Primero, se realiza la resta de $(\boldsymbol{A}-\boldsymbol{B})$, y si el resultado es negativo se suma $\boldsymbol{p}$ hasta obtener un número positivo, luego se divide el resultado entre $\boldsymbol{p}$ y se toma el residuo como el resultado de la operación. Por ejemplo, se desea calcular $(\mathbf{5}-\mathbf{7})\ \boldsymbol{mod}\ \mathbf{3}$, primero se restan $(\mathbf{5}-\mathbf{7})=-\mathbf{2}$, luego se suma $\mathbf{3}$ hasta obtener $\mathbf{1}$ (un numero positivo), se divide $\mathbf{1}$ entre $\mathbf{3}$ y se obtiene un residuo de $\mathbf{1}$, por lo tanto $(\mathbf{5}-\mathbf{7})\ \boldsymbol{mod}\ \mathbf{3}=\mathbf{1}$.

- **Multiplicación:** La multiplicación se comporta de la misma forma que la suma, dados dos números enteros $\boldsymbol{A}$ y $\boldsymbol{B}$, y un módulo $\boldsymbol{p}$, se desea calcular $(\boldsymbol{A}*\boldsymbol{B})\ \boldsymbol{mod}\ \boldsymbol{p}$. Primero, se realiza la multiplicación de $\boldsymbol{A}$ y $\boldsymbol{B}$, luego se divide el resultado entre $\boldsymbol{p}$ y se toma el residuo como el resultado de la operación. Por ejemplo, se desea calcular $(\mathbf{4}*\mathbf{3})\ \boldsymbol{mod}\ \mathbf{5}$, primero se multiplican $(\mathbf{4}*\mathbf{3})=\mathbf{12}$, luego se divide $\mathbf{12}$ entre $\mathbf{5}$ y se obtiene un residuo de $\mathbf{2}$, por lo tanto $(\mathbf{4}*\mathbf{3})\ \boldsymbol{mod}\ \mathbf{5}\ =\ \mathbf{2}$.

- **División:** La división en aritmética modular es un poco más compleja que las anteriores operaciones, debido a la falta de una división convencional en $(\boldsymbol{A}.\boldsymbol{Mod})$. Sin embargo, es posible utilizar el *inverso modular* para encontrar el resultado de una división, se dice que todo número tiene su inverso, el cual al multiplicarse ambos, el resultado es $\mathbf{1}$, siendo este el *elemento neutro*. En la **(Ec. 4)** se denota al inverso modular como $\boldsymbol{a^{-1}}$ y como este se multiplica con su complemento $\boldsymbol{a}$ para obtener el elemento neutro "**1**":

$$(a \cdot a^{-1})\ mod\ (m) = 1 \qquad \text{(Ec. 4)}$$

De tal manera que podemos utilizar la multiplicación para resolver la división, cambiando de $(\boldsymbol{A}/\boldsymbol{B})\boldsymbol{mod}\ \boldsymbol{p}$ por $(\boldsymbol{A}*\boldsymbol{B^{-1}})\boldsymbol{mod}\ \boldsymbol{p}$ siendo $\boldsymbol{B^{-1}}$ el inverso modular de $\boldsymbol{B}$. Un método para encontrar el inverso modular es utilizar el Algoritmo Extendido de Euclides (AEE) detallado más adelante en la sección **Algoritmo Extendido de Euclides AEE**.

Por consiguiente, el proceso para resolver la división es la siguiente: dados dos números enteros $\boldsymbol{A}$ y $\boldsymbol{B}$, y un módulo $\boldsymbol{p}$, se desea calcular $(\boldsymbol{A}/\boldsymbol{B})\ \boldsymbol{mod}\ \boldsymbol{p}$. Primero se necesita encontrar el inverso modular de $\boldsymbol{B}$ (es decir, el número que al multiplicarse con $\boldsymbol{B}$ da como resultado $\mathbf{1}$ en módulo $\boldsymbol{p}$), ese inverso modular es llamado $\boldsymbol{B^{-1}}$ y se multiplica

con $\boldsymbol{A}$, para obtener el resultado, siguiendo los pasos de la multiplicación. Por ejemplo, se desea calcular $(\mathbf{2/3})\ \boldsymbol{mod}\ \mathbf{5}$, primero se busca el inverso modular de $\mathbf{3}$ tambien expresado como $(\mathbf{3^{-1}}\boldsymbol{mod}\ \mathbf{5})$ lo cual a través de operaciones sucesivas se va comprobando de que el resultado sea $\mathbf{1}$; Se determina que $\mathbf{3^{-1}}$ en $(\mathbf{3} * \mathbf{3^{-1}}\boldsymbol{mod}\ \mathbf{5})$ es $\mathbf{2}$ ya que cumple con la **(Ec. 4)** $[(\mathbf{3} * \mathbf{2})\ \boldsymbol{mod}\ \mathbf{5} = \mathbf{1}]$, ahora se remplaza este término y la operación cambia a la multiplicación $(\mathbf{2} * \mathbf{2})\ \boldsymbol{mod}\ \mathbf{5}$, luego se multiplica $(\mathbf{2} * \mathbf{2}) = \mathbf{4}$, se toma el resultado $\mathbf{4}\ \boldsymbol{mod}\ \mathbf{5} = \mathbf{4}$, por lo tanto $(\mathbf{2/3})\ \boldsymbol{mod}\ \mathbf{5} = \mathbf{4}$ o $(\mathbf{2} * \mathbf{3^{-1}})\ \boldsymbol{mod}\ \mathbf{5} = \mathbf{4}$.

2.6 Algoritmo ECDH

Esta sección muestra cómo se combinan las anteriores secciones (**Criptografía con Curvas Elípticas** y **Matemática Modular**) para generar el Algoritmo Diffie-Hellmann en Curvas Elípticas (ECDH); este algoritmo genera un intercambio de claves asimétricas que se utiliza para establecer una clave secreta compartida entre dos partes, de manera segura. Se basa en la matemática de curvas elípticas y permite asegurar la privacidad de las comunicaciones en aplicaciones de todo tipo. ECDH es uno de los algoritmos recomendados por **NIST**[4] y se considera más seguro y eficiente en comparación con otros algoritmos de intercambio de claves.(Lucena López, 2010.)

2.6.1 Parámetros de la Ecuación de la curva elíptica

Para el intercambio de claves, se utiliza una curva con la **(Ec. 5)**: Sea una curva $\boldsymbol{E}$ formada sobre un campo $\boldsymbol{Zp}$, formada por el conjunto de elementos $(\boldsymbol{x}, \boldsymbol{y}) \in \boldsymbol{Zp}$, tales que:

$$E(\mathbb{Z}p) = \{(x, y) \in [y^2 = x^3 + ax + b \; mod \; (m)]\} \cup \{\mathcal{O}\} \qquad \text{(Ec. 5)}$$

Donde $a, b \in \mathbb{Z}p$ y cumple con $4a^3 + 27 \cdot b^2 \neq 0 \; mod \; (m)$, además $(\mathcal{O})$ es un punto en el infinito, este es un requisito indispensable para cumplir con las características de **grupo abeliano**[5] y se decide de forma arbitraria.

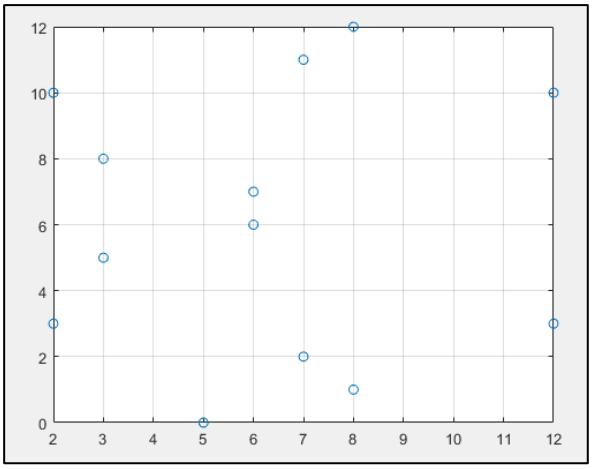
Como ejemplo la **Figura 18** toma los parámetros de la curva: $a = -3, b = 7 \; y \; p = 13$, Al definirla sobre $\boldsymbol{Z_{13}}$, se tiene un total de 13 puntos más el punto en el infinito, es decir 14 elementos.

[4] NIST es la sigla en inglés de National Institute of Standards and Technology. es una agencia del gobierno de los Estados Unidos encargada de establecer normas y tecnologías de medida, estandarización y seguridad en una amplia variedad de campos como la seguridad cibernética, la criptografía, etc.

[5] Un grupo abeliano es un tipo de grupo matemático que cumple con la propiedad de la conmutatividad, lo que significa que el orden de los elementos en una operación no importa. Son ampliamente utilizados en matemáticas, criptografía y teoría de números.

Figura 18

EC sobre el conjunto $\mathbb{Z}_{13}$



Nota: En la Figura se observa la cantidad de puntos generados en la curva $y^2 = -3x^3 + 7$ en el campo finito Z_{13}.

Para conocer el total de puntos de una EC, denotados como $\#\boldsymbol{E}$, que será crucial conocerlo para determinar la seguridad de una curva, se puede computar, en caso de que sean parámetros pequeños, pero normalmente se usará el **Teorema de Hasse**, en la **(Ec. 6)** se muestra la fórmula para el cálculo de la cantidad de puntos utilizando este teorema:

$$p - 2\sqrt{p} < \#\mathbb{E} < p + 2\sqrt{p} \qquad \text{(Ec. 6)}$$

Así se puede aproximar entre que valores estará el total de puntos de una curva. Por ejemplo: la **(Ec. 7)** utiliza los parámetros de la curva: $a = -3, b = 7\ y\ p = 13$, para limitar los valores entre un valor mínimo y un máximo.

$$13 - 2\sqrt{13} < \#\mathbb{E} < 13 + 2\sqrt{13} \qquad \text{(Ec. 7)}$$

$$5{,}788 < \#\mathbb{E} < 20{,}211$$

Y como se ha mostrado anteriormente en la **Figura 18** la cantidad de puntos para dicha curva es $\#\mathbb{E} = \mathbf{14}$, este Teorema es muy útil cuando utilizamos campos $\boldsymbol{Zp}$ con números primos muy grandes.

2.6.2 Punto Generador

Para trabajar con ECC, se necesita usar un grupo de elementos finitos, a fin de poder computarse en un tiempo razonable, cualquier elemento de nuestra curva elíptica puede ser un generador, pero interesa que incluya a todos los puntos de la curva para asegurar mayor seguridad. Este punto con coordenadas $\boldsymbol{x}$ y $\boldsymbol{y}$ será de donde parte el algoritmo para el intercambio de claves.

2.6.3 Orden del Grupo Cíclico

Una vez escogido el *punto generador* de la curva elíptica que funcionará como elemento primitivo, es necesario computar todos sus puntos hasta llegar al punto en el infinito. Para computar el grupo cíclico se utilizará la multiplicación escalar, aplicando $A.Mod$ a cada cálculo.

Cuando se tratan de dos puntos diferentes, suma de puntos:

$$Si\ x_1 \neq x_2\ entonces \begin{cases} x_3 = m^2 - x_1 - x_2\ mod\ (p) \\ y_3 = m(x_1 - x_3) - y_1\ mod\ (p) \end{cases}$$
$$con\ m = \frac{y_2 - y_1}{x_2 - x_1}\ mod\ (p) \qquad \text{(Ec. 8)}$$

Cuando se trata de puntos diferentes perpendiculares entre sí:

$$Si\ x_1 = x_2\ e\ y_1 \neq y_2\ entonces\ P + Q = \mathcal{O} \qquad \text{(Ec. 9)}$$

Cuando se trata del mismo punto, múltiplos de un punto:

$$Si\ x_1 = x_2\ e\ y_1\ = y_2\ entonces \begin{cases} x_3 = m^2 - 2x_1\ mod\ (p) \\ y_3 = m(x_1 - x_3) - y_1\ mod\ (p) \end{cases} \qquad \text{(Ec. 10)}$$

$$con\ \mathrm{m} = \frac{3x_1^2 + \mathrm{A}}{2y_1} \mathrm{mod}\ (\mathrm{p})$$

Para el cálculo de la pendiente (m): primero se obtiene el denominador y encontrando su inverso con $\boldsymbol{AEE}$ se multiplica con el numerador, siguiendo la propiedad de la división modular.

2.6.4 Cofactor

Es un parámetro usado para determinar la seguridad de la curva e idealmente debería ser 1 y no mayor de 4. Se calcula con la **(Ec. 11)** donde el cofactor es la cantidad de puntos hasta llegar al infinito dividido para el orden del punto generador, en la sección **Orden de la curva y de un punto** se detalla la obtención de estos valores.

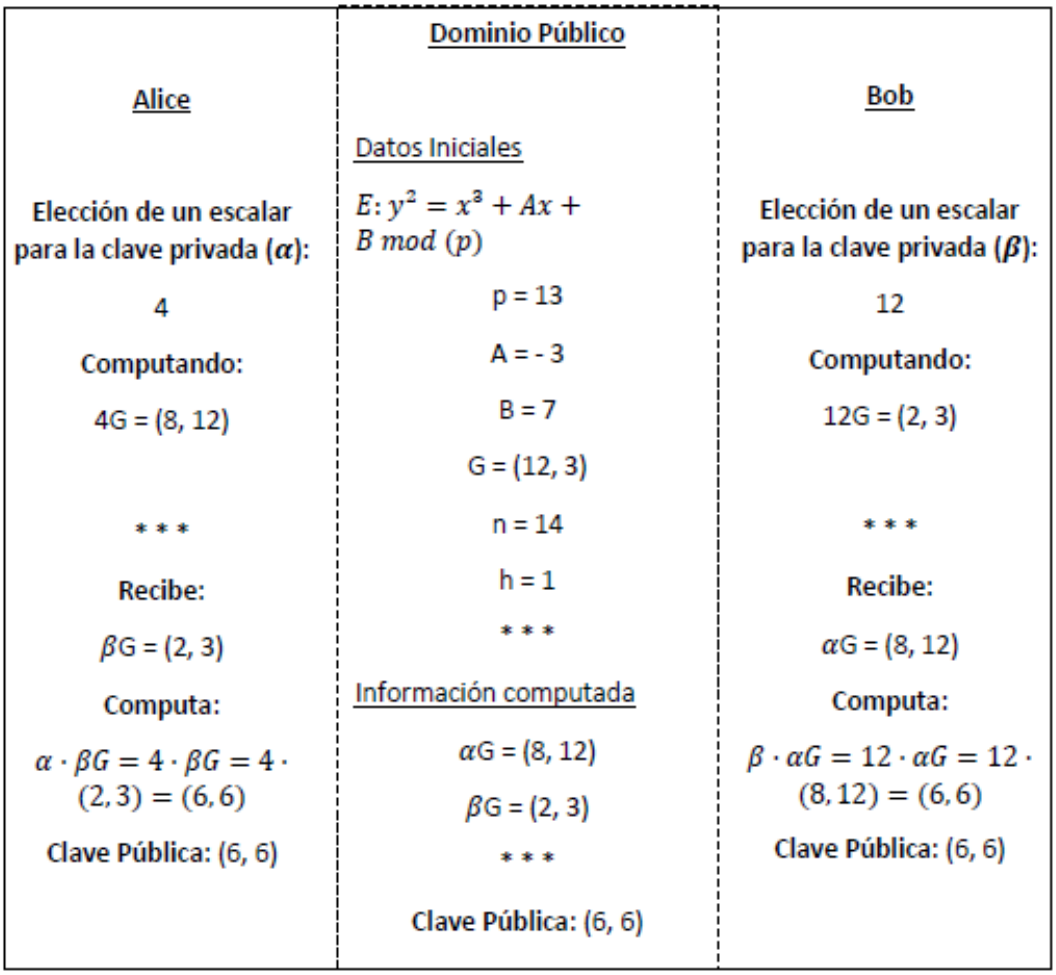
$$Cofactor\ (h) = \frac{\#E\left(\mathbb{Z}/\mathbb{Z}p\right)}{ord\ (G)} \qquad \text{(Ec. 11)}$$

2.6.5 Puntos Computados

Por último, será necesario que ambas partes (emisor y receptor) eligen de forma secreta e independiente un entero positivo, el cual será su clave privada. Una vez seleccionado de forma aleatoria, se realiza una multiplicación escalar con este entero y el generador, el cual dará un elemento del grupo cíclico. Ambos enviarán este elemento y será de dominio público. Cada uno cogerá el punto del otro, y lo multiplicará por su escalar, consiguiendo el mismo elemento del grupo cíclico y siendo éste la clave secreta compartida. En la **Figura 19** se muestra el esquema de intercambio de claves ECDH utilizando como ejemplo dos usuarios Alice y Bob los cuales aplican el protocolo de intercambio de claves para la obtención de una clave secreta compartida y segura.

Figura 19

Esquema del Protocolo de Intercambio de Claves Diffie-Hellman en Curvas Elípticas



Nota: En la Figura se observa la representación del protocolo de intercambio de claves Diffie-Hellman en Curvas Elípticas, adaptado de (Poincare, 2019).

Finalmente, así funcionaria el protocolo ECDH, en el capítulo siguiente se muestra todos los algoritmos que se implementaran en un sistema de cifrado de datos ligero usando las matemáticas con curva elípticas y su equivalente en algoritmos para ser implementados en una WSN.

2.7 Comparativa ECDH respecto RSA

De entre todos los algoritmos asimétricos, quizá **RSA** sea el más sencillo de comprender e implementar, sus claves sirven indistintamente tanto para cifrar como para autentificar. RSA se basa en la dificultad para factorizar grandes números. Las claves pública y privada se calculan a partir de un número que se obtiene como producto de dos primos grandes.

En la **Tabla 1** se puede observar la cantidad de instrucciones por Segundo (MIPS) durante un año necesarios para romper la seguridad comparado con los tamaños de clave.

Tabla 1. *Comparación de la Longitud. de clave con el Ratio entre RSA y ECC*

Tiempo de Ruptura (MIPS año)	*Tamaño de clave RSA (bits)*	*Tamaño de clave ECC (bits)*	*Relación entre claves RSA/ECC*
10^4	512	106	6:1
10^8	768	132	6:1
10^{11}	1024	160	7:1
10^{20}	2048	210	10:1
10^{78}	21000	600	35:1

Como se puede ver, la ratio de seguridad aumenta de forma muy notable, por lo que significaría que ECDH es un sistema más seguro y eficiente. Es importante analizar sus implicaciones en la eficiencia del sistema, ya que a mayor longitud de clave se exige:

- Mayor capacidad de almacenamiento y computación.
- Más energía para la computación.
- Más tiempo para realizar las operaciones.
- Mayor amplitud de banda para transportar la clave y el mensaje a posteriori.

Por todas estas repercusiones, se puede deducir que para tener una comunicación más eficiente se debe optar por la menor longitud de clave posible, siempre que certifique la seguridad necesaria.

2.8 Software para simulación de WSN

Se ha considerado varios softwares de simulación para representar o emular los diferentes despliegues de WSN y la implementación del algoritmo criptográfico. Los simuladores son una potente herramienta utilizada para facilitar la investigación, se los utiliza de manera constante para comparar y evaluar diferentes diseños de redes, probar la funcionalidad de los algoritmos, así como de nuevos y preexistentes protocolos. Se considera el programa basándonos en varias características como son: su complejidad o facilidad de manejo, bajo costo o versiones gratuitas, lenguaje de programación que utiliza, etc.

A continuación, se presenta una descripción general de los diversos simuladores actualmente disponibles para WSN.

El primer software por comparar es **Shawn**, este es un simulador para redes de sensores personalizable de código abierto, diseñado para soportar redes a gran escala utilizando un lenguaje de programación C++. Este consiste en características de perseverancia y desacople de las instancias de la simulación. Estas características hacen que sea más fácil la implementación. Con Shawn es posible simular diferentes tipos de redes WSN, y su diseño se enfoca en lograr una velocidad de simulación a gran escala, la curva de aprendizaje de Shawn dependerá en gran medida de los conocimientos previos y habilidades del usuario en programación y simulación de redes.

Por otra parte, **VisualSense** es otro software simulador para redes es está diseñado para representar requerimientos específicos de las redes inalámbricas, así como examinar los canales de comunicación. Está diseñado para mantener una base de componentes estructurados como modelos. Ofrece un exacto y extensible modelo de radio comunicación. El modelo de radio utilizado viene acompañado de un modelo de programación de energía general que puede ser reprogramado por los fenómenos físicos. (Ptolemy, 2021). VisualSense tambien utiliza el lenguaje C++, simular diferentes tipos de redes de sensores inalámbricos, incluyendo sensores de área amplia, sensores de baja potencia, sensores de baja tasa de datos y sensores basados en eventos. La curva de aprendizaje de VisualSense puede ser considerada moderada debido a la complejidad del software.

Otro programa simulador de redes es **Network Simulator (Ns2-Ns3)**, NS-3 es un simulador de red de eventos discretos, dirigido principalmente para investigación y uso educativo.

Permite a los usuarios simular el comportamiento de redes de varios tamaños y configuraciones, y a menudo se utiliza para estudiar el rendimiento de diferentes protocolos y algoritmos de red. NS2 y NS3 están escritos en el lenguaje de programación C++ y están disponible como software de código abierto. (Zarrad, 2017). El lenguaje de programación utilizado por Ns2-Ns3 es C++ y Otcl (Object-oriented Tool Command Language). La curva de aprendizaje de Ns2-Ns3 puede ser un poco empinada, especialmente para aquellos que no tienen experiencia previa en programación, pero hay mucha documentación disponible en línea que puede ayudar a los usuarios a comprender su funcionamiento.

Tambien, otro software es **OMNeT++**, este es un simulador modular de eventos discretos de redes orientado a objetos, usado habitualmente para modelar el tráfico de redes de telecomunicaciones, protocolos, sistemas multiprocesadores y distribuidos, validación de arquitecturas hardware, evaluación del rendimiento de sistemas software y, en general, modelar cualquier sistema que pueda simularse con eventos discretos. Utiliza un lenguaje de programación de alto nivel llamado C++. OMNeT++ puede simular una amplia variedad de redes, incluyendo redes de comunicaciones móviles, redes de sensores inalámbricos, redes de computadoras, redes de transporte, entre otras. La curva de aprendizaje de Omnet++ puede ser un poco inclinada al principio, pero una vez que se tiene una comprensión básica, se vuelve más fácil de usar.

El ultimo software por comparar es **MATLAB** este puede simular una amplia gama de sistemas, incluyendo sistemas dinámicos, sistemas de control, procesamiento de señales, análisis de datos, y más. MATLAB es un sistema de cómputo numérico que ofrece un entorno de desarrollo integrado con un lenguaje de programación propio. Está disponible para las plataformas Unix, Windows, macOS y GNU/Linux-MATLAB, utiliza el lenguaje de cálculo técnico desarrollado por MathWorks, tiene un entorno de programación para el desarrollo de algoritmos, análisis de datos, visualización y cálculo numérico, dispone de módulos para simular diferentes tipos de sensores. (PUCP, 2019) La curva de aprendizaje de MATLAB puede ser rápida para aquellos con experiencia en programación, pero puede ser un poco desafiante para los principiantes.

2.9 Hardware para WSN

A continuación, se presenta una descripción general de algunos hardware programables actualmente disponibles para WSN, detallando sus fortalezas y debilidades en diferentes campos como costo, facilidad de uso, utilidad para despliegues de WSN, etc.

En primer lugar **Arduino** es una plataforma de creación de electrónica de código abierto, Los costos de Arduino pueden variar de unos pocos dólares a cientos de dólares, dependiendo del tipo de placa y de los componentes que se incluyan como sensores. Es conocido por su facilidad de uso y su gran comunidad en línea. Se utiliza con frecuencia en pequeñas redes de sensores debido a su bajo costo y su facilidad de uso. Sin embargo, puede ser más difícil escalar a grandes redes de sensores debido a limitaciones en el rendimiento y la capacidad de procesamiento.

Otra alternativa es **Raspberry Pi**, este es un ordenador de bajo coste y formato compacto destinado al desarrollado para hacer accesible la informática a todos los usuarios. La Raspberry Pi también se caracteriza por ser muy utilizada para desarrollar pequeños prototipos y para la formación sobre informática y electrónica.(Roberto Solé, 2021) Los modelos más básicos de Raspberry Pi cuestan alrededor de $35-50, mientras que los modelos más avanzados pueden costar cientos de dólares. Es una plataforma más avanzada que Arduino, y puede ser un poco más difícil de usar para algunos usuarios. Sin embargo, también existe una gran comunidad en línea que brinda soporte y recursos para ayudar a los usuarios a aprender y usar la plataforma. Es adecuada para una amplia gama de proyectos de IoT, incluidas las redes de sensores.

Finalmente, **LoRa** es una tecnología de comunicación inalámbrica que se utiliza para conectar dispositivos a larga distancia, utiliza la banda de frecuencia ISM de 915 MHz y permite la transmisión de datos a larga distancia (hasta kilómetros) con baja potencia y un ancho de banda estrecho. LoRa es adecuado para proyectos de IoT y WSN que requieren la transmisión de datos a larga distancia, como por ejemplo sistemas de monitoreo de la fauna, sensores de agua subterránea, y monitoreo de la energía renovable. LoRa pueden ser más caros que otros componentes de IoT como Arduino o Raspberry Pi. Sin embargo, la tecnología LoRa puede ser más rentable a largo plazo debido a su capacidad de transmitir datos a larga distancia y su baja potencia de transmisión. En cuanto a la facilidad de uso, la implementación de una red LoRa puede requerir un poco más de conocimientos técnicos y recursos que otras tecnologías de IoT, pero existen muchos recursos y documentación para los usuarios.

3 CAPITULO III. Requerimientos y Diseño del esquema criptográfico adaptado a WSN

En esta sección se realizará un estudio de los Estándares para la eficiencia criptográfica versión 1 y 2, además del uso de la metodología IEEE29148 para definir los elementos del sistema que los dispositivos de bajo rendimiento deben cumplir para implementar el algoritmo ECC y la elección del software de simulación donde se especifica los procesos requeridos para actividades de ingeniería que dan como resultado productos de software y donde se establece los requerimientos de información, directrices y un ciclo de vida de trabajo.

3.1 Metodología

El presente Trabajo de Titulación se basa en una investigación descriptiva, experimental y práctica, ya que describe las características de la criptografía con curvas elípticas para comprender e implementar los algoritmos basados en la misma, con el fin de desarrollar un sistema criptográfico que se pueda utilizar en cualquier tipo de red Inalámbrica, enfocado en dispositivos con bajo rendimiento.

La elección de una metodología adecuada para el desarrollo de un proyecto depende de los requerimientos y propósitos del mismo, como también de las investigaciones previas; debido a que se necesita seguir procedimientos, se selecciona la metodología en cascada, puesto que, dicha metodología incluye fases de verificación en cada etapa de diseño, con esto se consigue tener un sistema que cumpla con los requerimientos planteados inicialmente, así como también dar cumplimiento a los objetivos propuestos.

3.1.1 Metodología en Cascada

La metodología en cascada, también llamado secuencial o ciclo de vida de un programa, es el enfoque metodológico que ordena rigurosamente las etapas del proceso para el desarrollo de software, de tal forma que el inicio de cada etapa debe esperar a la finalización de la etapa anterior. Al final de cada etapa, el modelo está diseñado para llevar a cabo una revisión final, que se encarga de determinar si el proyecto está listo para avanzar a la siguiente fase.

Un ejemplo de una metodología de desarrollo en cascada es:

- Análisis de requisitos.
- Diseño del sistema
- Implementación
- Pruebas
- Entrega

De esta forma, cualquier error de diseño detectado en la etapa de prueba conduce necesariamente al rediseño o modificación del sistema.

3.1.2 Metodología IEEE-29148.

La Metodología IEEE-29148 es un estándar de la organización **IEEE**[6] que establece los lineamientos para el desarrollo de software y la documentación de los requisitos de software. La metodología se centra en proporcionar un enfoque coherente y sistemático para la recopilación, análisis y validación de los requisitos de software. IEEE-29148 proporciona un marco de trabajo que incluye las siguientes actividades clave:

- **Definición de requisitos**: identificación y documentación de los requisitos del software.
- **Análisis de requisitos**: evaluación de los requisitos para asegurarse de que sean completos, coherentes y realistas.
- **Validación de requisitos**: verificación de que los requisitos cumplen con los objetivos y expectativas del negocio y los usuarios.
- **Gestión de requisitos**: seguimiento y control de los requisitos a lo largo del ciclo de vida del software.

El objetivo principal de la Metodología IEEE-29148 es mejorar la calidad y la eficacia del proceso de desarrollo de software y asegurar que los requisitos del software sean claros, precisos y adecuados para el uso previsto.

[6] El IEEE (Institute of Electrical and Electronics Engineers) es una organización profesional global dedicada al avance de la tecnología para el beneficio de la humanidad.

3.2 Fase 1: Análisis de Requisitos

Esta fase tiene como objetivo establecer los requerimientos del sistema en cuanto a requerimientos iniciales, de software, de hardware y de Stakeholders, aplicando la metodología IEEE-29148, los requisitos fueron determinados en base a un análisis bibliográfico que se encuentra en el **ANEXO A – Requerimientos**. Los acrónimos usados para la definición de cada uno de los requerimientos utilizados en este trabajo se detallan en la **Tabla 2**, en la **Tabla 3** se indica los diferentes tipos de prioridad que tendrán cada requerimiento independientemente si está enfocado al Sistema, Arquitectura o Stakeholders.

Tabla 2. *Acrónimos del estándar IEEE 29148:2022*

Requerimiento	Acrónimo
De Stakeholders	*StSR*
De Sistema	*SySR*
De Arquitectura	*SrSH*

Tabla 3. *Priorización de requerimientos del sistema*

Prioridad	***Descripción***
Alta	Se considera un nivel de alta importancia que debe de ser incluido en el desarrollo del sistema. Caso contrario afecta el funcionamiento de este.
Media	Se considera un nivel de media importancia, no incluirlos pueden afectar al funcionamiento del sistema.
Baja	Se considera un nivel de mínima importancia, es decir que se los puede excluir, y no representa un impacto significativo en el funcionamiento del sistema.

3.2.1 Estándares para la eficiencia criptográfica "SECG"

El **SECG**[7] fue creado por la empresa **CERTICOM**, fundada en 1985 para promover estándares de curva elíptica, así como la difusión de los mejores métodos para implementar este tipo de criptografía. Actualmente la empresa dispone de 350 patentes relacionadas con ECC, y varias patentes que se encuentran en desarrollo.

Los frutos de esta organización quedan reflejados en dos estándares principales. El primero de ellos, formado en el 2009 llamado "***SEC 1 – Elliptic Curve Cryptography***" el cual hace una descripción de los esquemas permitidos (**ECDSA**[8], **Algoritmo** ECDH, **ECMQV**[9] y **ECIES**[10]). Además, se describen todas las *primitivas criptográficas* (bloques fundamentales para la construcción de la criptografía, es decir operaciones matemáticas básicas) que se usan en estos esquemas para representar las estructuras necesarias (claves, certificados, contenidos cifrados, etc.).

El segundo documento, ***"SEC 2 – Recommended Elliptic Curve Domain Parameters"***, hace una propuesta sobre los parámetros a utilizar de los esquemas definidos en ***SEC 1***, El uso de estas recomendaciones aumentan en gran medida la interoperabilidad de las aplicaciones que los usen.

Estos estándares forman parte de los requerimientos del sistema ya que el trabajo de esta organización es uno de los principales promotores de la estandarización criptográfica con curvas elípticas. Convirtiéndose en una herramienta indispensable para encontrar nuevas soluciones en la optimización de protocolos, sobre todo en entornos donde el tamaño de la clave no puede ser demasiado grande, un claro ejemplo de esto son las WSN.

[7] **SECG** "Grupo de Estándares para Criptografía Eficiente", es un grupo de trabajo que se encarga de desarrollar y promover estándares criptográficos para sistemas con recursos limitados y de baja potencia.
[8] **ECDSA** "Algoritmo de Firma Digital de Curva Elíptica", es un algoritmo de firma digital de clave pública que se basa en el uso de curvas elípticas para garantizar la autenticidad y la integridad de un mensaje.
[9] **ECMQV** "Elliptic Curve Menezes-Qu-Vanstone" es un protocolo criptográfico de clave pública que se utiliza para el intercambio de claves secretas entre dos partes en una comunicación segura.
[10] **ECIES** "Elliptic Curve Integrated Encryption Scheme" es un esquema criptográfico de cifrado de clave pública, es utilizado para proporcionar seguridad en la transmisión de datos y garantizar la privacidad e integridad de la información transmitida.

3.2.2 Requerimientos de STAKEHOLDERS

En la **Tabla 4**, se muestra el listado de los **Stakeholders**[11], estos indican las personas que están involucradas directa e indirectamente en el desarrollo del proyecto con su respectiva función, y quienes se verán envueltos dentro de distintas etapas del proceso.

***Tabla 4**. Requerimiento de Stakeholders*

Actores Involucrados		
No.	***Actor***	***Función***
1	*Christopher Ortega*	*Desarrollador del Proyecto*
2	*Msc. Fabián Cuzme*	*Director*
3	*Msc. Hernán Domínguez*	*Asesor*

3.2.2.1 Requerimientos Operacionales

Los requerimientos operacionales (StSR), también conocidos como requisitos de rendimiento, se utilizan para definir el correcto funcionamiento del sistema y las prioridades que tendrán cada requerimiento. Estos requisitos se derivan de los stakeholders, que son las partes interesadas en el sistema, y se basan en las operaciones del sistema y las condiciones necesarias para su correcta implementación. En la **Tabla 5** se muestra todos los requerimientos los cuales son esenciales para el éxito del sistema, en su mayoría se basan en los requerimientos determinados por el autor, las operaciones del sistema, la capacidad de cifrar y descifrar datos de manera segura y eficiente; y las condiciones necesarias para implementar la criptografía ECC en WSN.

***Tabla 5.** Requerimientos de Stakholders - Operacionales*

StSR ***REQUERIMIENTOS OPERACIONALES***				
No.	***Requerimiento***	***Prioridad***		
		Alta	***Media***	***Baja***
StSR 1	*El algoritmo debe ser modificable para generar distintos tamaños de clave y tiempos de cifrado*		*X*	
StSR 2	*Establecer las pruebas de funcionamiento en tres escenarios de Simulación*	*X*		

[11] Los stakeholders (interesados, en español) son personas, grupos u organizaciones interesadas en el éxito o el fracaso de un proyecto.

StSR 3	*Se debe presentar altos conocimiento en lenguajes de programación (C++ y MATLAB)*	*X*		
StSR 4	*El sistema debe ser interoperable de manera efectiva con otros sistemas*			*X*
StSR 5	*Se debe demostrar conocimiento de Criptografía EC y como se aplica está a una red de sensores*	*X*		
StSR 6	*Verificación de la Encriptación / Desencriptación utilizando una comunicación IP y la captura de datos*	*X*		
StSR 7	*El tipo de cifrado de datos puede ser cifrado de flujo o cifrado de bloque*			*X*
StSR 8	*El sistema debe tener la capacidad de manejar grandes volúmenes de datos*			*X*

3.2.3 Requerimientos de Sistema

En la **Tabla 6**, se establecen los requerimientos del sistema (SySR), que son un conjunto de especificaciones técnicas que describen el funcionamiento del sistema de manera detallada. Estos requerimientos son críticos para asegurar que el sistema cumpla con sus objetivos propuestos y limitaciones.

***Tabla 6**. Requerimientos del sistema SySR*

SySR				
No.	***Requerimiento***	***Prioridad***		
		Alta	***Media***	***Baja***
SySR 1	*El sistema utilizara los algoritmos: ECDH y ELGamal Elíptico*	*X*		
SySR 2	*Cada escenario de simulación debe contar con los parámetros esenciales de cada WSN*		*X*	
SySR 3	*Gestionar ciclo de vida de las claves, donde se renueven cada cierto tiempo*		*X*	
SySR 4	*Notificaciones del Sistema ante eventos inesperados en los procesos de: Generación de claves, encriptación y desencriptación*	*X*		
SySR 5	*Implementación de medidas de seguridad adicionales, como autenticación y control de acceso a los datos*			*X*
SySR 6	*Menor tiempo de respuesta dependiendo de la aplicación y entorno de prueba*		*X*	
SySR 7	*Verificación de la seguridad utilizando análisis de tramas*	*X*		

SySR 8	*Longitud de clave equilibrando tiempo de procesamiento y de cifrado*		*X*	
SySR 9	*Compatibilidad con la simulación en OMNeT++ y capacidad de generar datos de simulación para el análisis de desempeño.*	*X*		
SySR 10	*Documentación y guía de usuario para el uso y configuración del sistema.*			*X*
SySR 11	*Monitoreo de parámetros de hardware como: nivel de procesamiento y consumo de energía*		*X*	

3.2.4 Requerimientos de Arquitectura

En la **Tabla 7** se establecen los requerimientos de arquitectura (SrSH), que definen la arquitectura que se debe utilizar para el correcto funcionamiento del sistema. Estos requerimientos son esenciales para garantizar que el sistema cumpla con las especificaciones técnicas requeridas. El objetivo de estos requerimientos está en conocer las características del sistema, sus limitaciones, el rendimiento con el algoritmo, la precisión de los cálculos, tiempo de respuesta, el tiempo de cifrado, etc.

Tabla 7. *Requerimientos de Arquitectura SrSH*

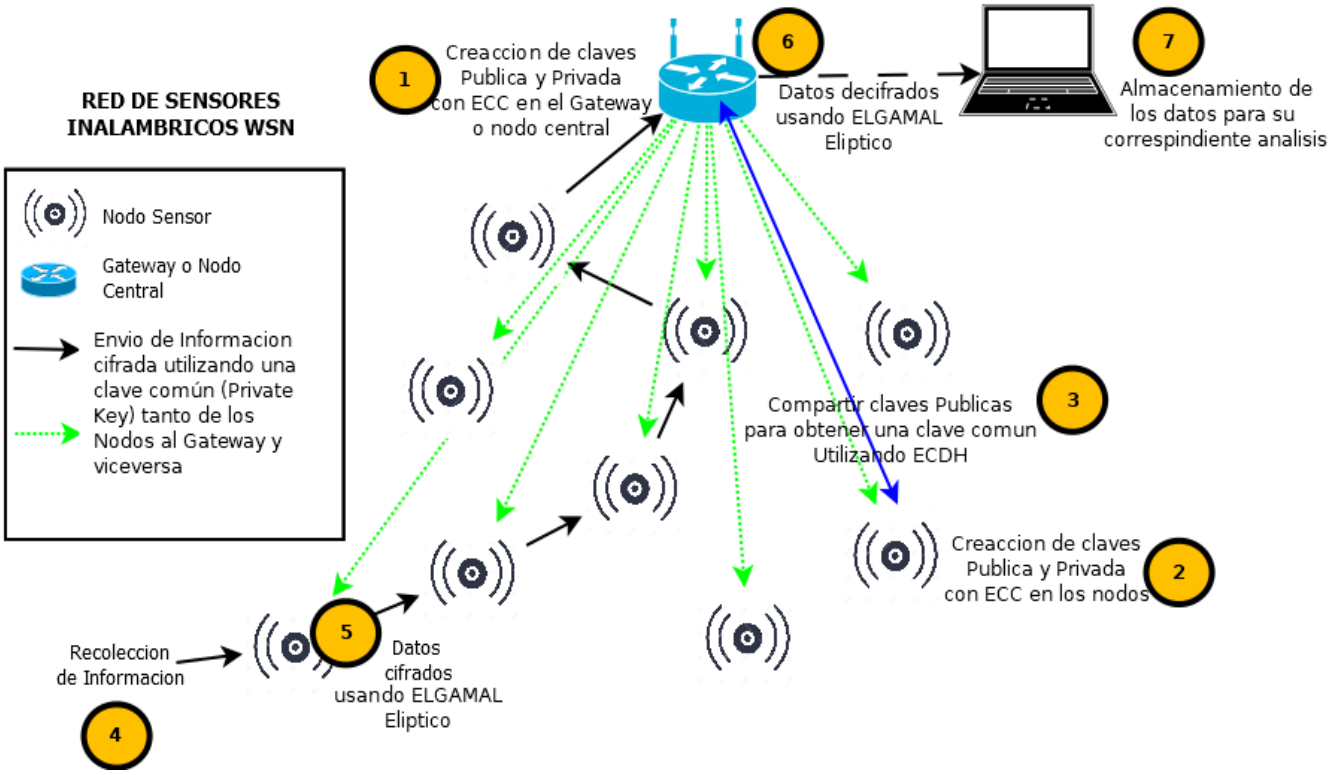
SrSH				
No.	***Requerimiento***	***Prioridad***		
		Alta	***Media***	***Baja***
SrSH 1	*Dispositivo base de alto rendimiento, equipo donde se implementa la simulación de las redes con ECC*		*X*	
SrSH 2	*Definir el tipo de comunicación de la red de sensores*		*X*	
SrSH 3	*Limitaciones de Hardware en base a el estándar 802.15.4 y el tipo de Criptografía soportada*	*X*		
SrSH 4	*Los nodos sensores debe tener un bajo consumo energético.*			*X*
SrSH 5	*La plataforma de simulación debe contar con varias características para la creación de una red de sensores*	*X*		

3.3 Fase 2: Diseño del Sistema

El sistema permitirá brindar seguridad criptográfica a cualquier red inalámbrica (WSN, WBAN, WLAN, etc.) o cualquier escenario donde sea necesario cifrar datos de manera ligera. Para ello, se permite generar un método de encriptación basado en curvas elípticas sobre campos finitos primos, estableciendo las restricciones necesarias para el hardware. En la **Figura 20**, se muestra un ejemplo de cifrado de datos utilizando una clave en común (Private Key), que es obtenida por cada dispositivo mediante diferentes procesos e intercambios de claves asimétricas. Esta clave se utilizará para cifrar los datos obtenidos de los sensores.

Figura 20

Red de sensores Inalámbricos con ECC



Nota: En la Figura se observa el envío de información utilizando una clave privada "criptografía simétrica" para el cifrado de los datos generados por los nodos sensores, esta clave será generado mediante criptografía de curva elíptica.

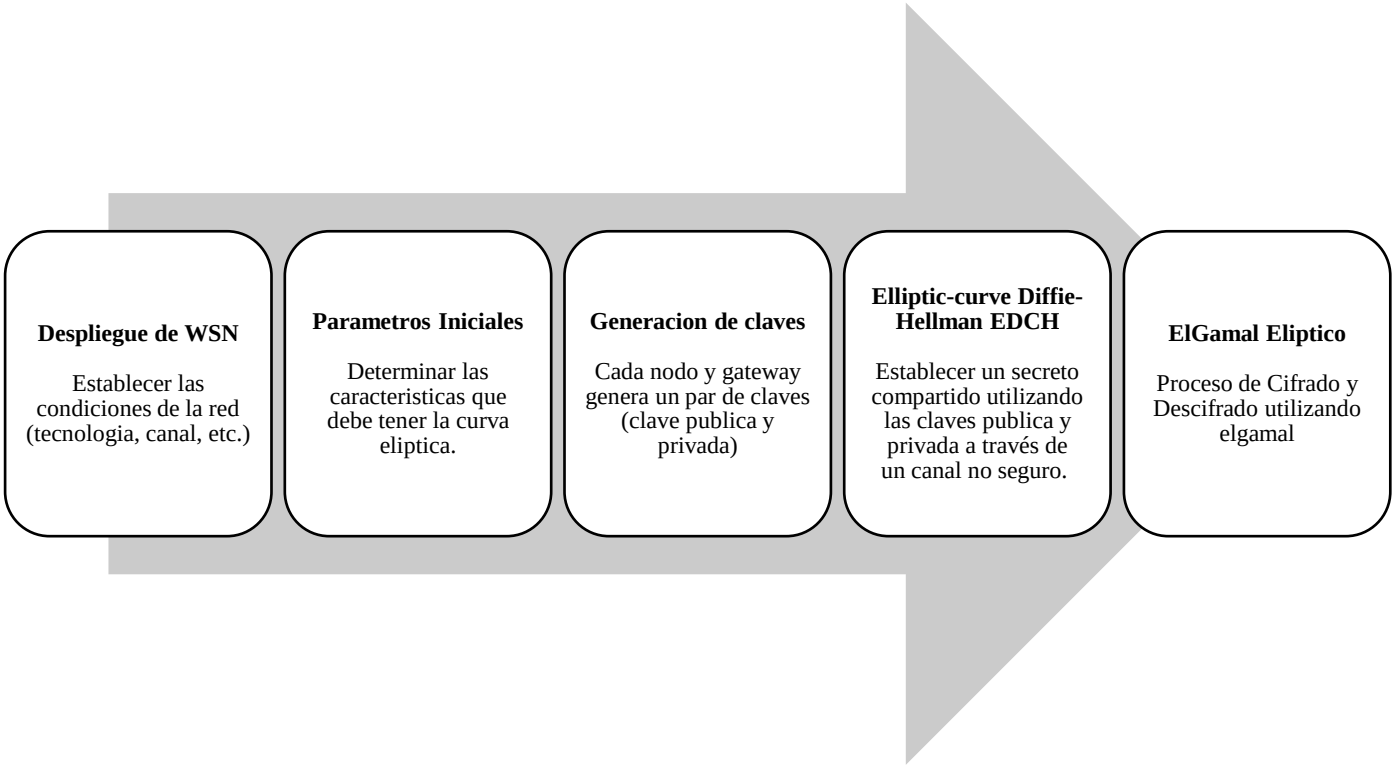
3.3.1 Diagrama de Bloques del sistema

El sistema basado en la criptografía de curvas elípticas se divide en varios procesos o etapas involucrando la generación de claves, intercambio de claves, consenso de clave común, la representación de caracteres en puntos de una curva elíptica y el proceso de cifrado y descifrado utilizando el Gamal elíptico.

La **Figura 21** representa el diagrama general del sistema, en donde se describen básicamente los procesos principales del sistema ya mencionados, posteriormente se profundizará a detalle cada uno de estos juntos con sus respectivos subprocesos.

Figura 21

Diagrama de bloques del sistema



Nota: En la Figura se observa los procesos Generales a utilizar para el establecimiento de una comunicación segura en una red de sensores, los procesos de generación de claves, ECDH y ElGamal Elíptico serán los mismos para cualquier despliegue y cualquier tipo de parámetros iniciales.

3.3.1.1 Despliegue de WSN

Un despliegue de WSN es la instalación y configuración de una red de sensores inalámbricos en un área específica, o representar dicha configuración en un software destinado a la simulación de estas. Se propone implementar tres ejemplos de despliegues aplicado a funciones tales como **monitoreo ambiental, monitoreo de lugares cerrados, y salud**. Una WSN puede ser desplegada en cualquier área, desde una ciudad hasta un parque natural, y puede incluir desde pocos sensores hasta miles de ellos, dependiendo de la complejidad del sistema y del objetivo del despliegue.

Para desplegar una red de WSN, es necesario tener en cuenta varios factores, como la ubicación de los dispositivos de sensores, la topología de la red, el protocolo de comunicación utilizado, y la forma en que se recopilan y procesan los datos. Además, es importante asegurarse de que la red tenga una buena fiabilidad y seguridad para garantizar la confiabilidad de los datos recopilados.

3.3.1.2 Parámetros de dominio de una curva elíptica

Los parámetros de dominio de una curva elíptica son un conjunto de valores que se utilizan para definir una curva elíptica y especificar cómo se debe trabajar con ella. Los parámetros de dominio se suelen representar mediante el conjunto $\boldsymbol{T} = (\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{G}, \boldsymbol{n}, \boldsymbol{h})$, donde:

- $\boldsymbol{p}$ es el tamaño del campo modular sobre el cual se define la **Curvas Elípticas sobre cuerpos finitos primos** $\mathbb{Z}_{\boldsymbol{p}}$. Este parámetro determina el número de puntos posibles en la curva elíptica y también define el tamaño del espacio de claves.
- $\boldsymbol{a}$ y $\boldsymbol{b}$ son los coeficientes del término lineal y cuadrático, respectivamente, en la ecuación de la curva elíptica **(Ec. 2)**.
- $\boldsymbol{G}$ es el **Punto Generador** de la curva elíptica, que se utiliza como punto de partida para realizar la multiplicación escalar y obtener el par de claves.
- $\boldsymbol{n}$ es el **Orden del Grupo Cíclico**, que es el número de veces que se debe multiplicar el punto generador por sí mismo para obtener el punto en el infinito.
- $\boldsymbol{h}$ es el coeficiente de fricción también llamado **Cofactor**, que se utiliza para mejorar la

seguridad de algunos algoritmos.

- Se suele agregar una **función de hash**[12], donde se toma un mensaje de entrada y se convierte en una cadena de caracteres de tamaño fijo. Esto se utiliza para asegurar la integridad del mensaje durante el proceso de cifrado y envió.

Es importante definir estos parámetros antes de iniciar cualquier proceso que involucre el uso de curvas elípticas, ya que son esenciales para la seguridad y la confiabilidad de los algoritmos criptográficos que se utilizan.

Los parámetros de dominio se utilizan en diferentes protocolos de criptografía con curvas elípticas, como el protocolo de intercambio de claves de Diffie-Hellman sobre curvas elípticas (ECDH) y el esquema de cifrado ElGamal sobre curvas elípticas (ElGamal Elíptico). Estos protocolos se utilizan para realizar el intercambio de claves y el cifrado/descifrado de mensajes, respectivamente, en un entorno seguro.

3.3.1.3 Generación de Claves

La generación de claves con ECC se basa en el uso de un par de claves: una **clave privada** y una **clave pública**. La clave privada es conocida solo por el propietario de la clave y se utiliza para generar la clave pública. La clave pública, por otro lado, es conocida por cualquiera. Para generar un par de claves con ECC, se sigue estos pasos:

- Se elige una curva elíptica y un punto base $\boldsymbol{G}$ en la curva. Estos **Parámetros de dominio de una curva elíptica** se utilizarán para definir el "espacio/campo" donde se obtendrán las claves.
- Se elige un número privado $\boldsymbol{d}$, este número debe ser un número entero aleatorio y conocido solo por el propietario de la clave.
- Se utiliza el número privado $\boldsymbol{d}$, el punto base $\boldsymbol{G}$ y la curva elíptica para calcular la clave pública $\boldsymbol{Q}$. La clave pública $\boldsymbol{Q}$ es un punto en la curva elíptica y se calcula utilizando la **(Ec. 10)**, obteniendo $\boldsymbol{Q = d * G}$.

[12] Una función de hash es una función matemática que se utiliza para convertir datos de cualquier tamaño o tipo en un valor de tamaño fijo que representa a los datos originales, esta transformación es irreversible y se utiliza en una amplia variedad de aplicaciones incluyendo la criptografía, la seguridad de la información, la integridad de los datos, etc.

- Se guarda la clave privada $\boldsymbol{d}$ y la clave pública $\boldsymbol{Q}$. Estos forman el par de claves ECC.

Es importante tener en cuenta que la seguridad de la criptografía con curvas elípticas depende del tamaño del número privado $\boldsymbol{d}$ y del número primo $\boldsymbol{p}$ utilizado para definir el espacio de claves. Cuanto mayor sea el tamaño de estos números, más segura será la criptografía.

3.3.1.4 Elliptic-Curve Diffie-Hellman (ECDH)

El protocolo (ECDH) tiene como objetivo el intercambio de claves para establecer una *clave secreta compartida* entre dos partes que se comunican a través de un canal inseguro; en este ejemplo las partes son "$\boldsymbol{A}$ y $\boldsymbol{B}$". ECDH se basa en la **Criptografía Asimétrica (Clave Pública y Privada)** basada en curvas elípticas. El método realiza los siguientes pasos:

- Los **Parámetros de dominio** de una curva elíptica deben ser los mismos entre ambas partes.
- Ambas partes generan sus *propias claves* usando la **Generación de Claves,** obteniendo así para $\boldsymbol{A}$ la clave privada $\boldsymbol{d_A}$ y la clave pública $\boldsymbol{Q_A = d_A * G}$ y para $\boldsymbol{B}$ la clave privada $\boldsymbol{d_B}$ y la clave pública $\boldsymbol{Q_B = d_B * G}$.
- Ambas partes intercambian sus *claves públicas* por el canal inseguro, es decir $\boldsymbol{A}$ envía $\boldsymbol{Q_A}$ hacia $\boldsymbol{B}$, y $\boldsymbol{B}$ envía $\boldsymbol{Q_B}$ hacia $\boldsymbol{A}$.
- $\boldsymbol{A}$ con su propia clave privada $\boldsymbol{d_A}$ y la clave pública de $\boldsymbol{B}$ $\boldsymbol{Q_B}$ calcula el punto $\boldsymbol{S_A = d_A * Q_B}$ obteniendo la clave secreta compartida que es $\boldsymbol{S_A = d_A * d_B * G}$.
- $\boldsymbol{B}$ con su propia clave privada $\boldsymbol{d_B}$ y la clave pública de $\boldsymbol{A}$ $\boldsymbol{Q_A}$ calcula el punto $\boldsymbol{S_B = d_B * Q_A}$ obteniendo la clave secreta compartida que es $\boldsymbol{S_B = d_B * d_A * G}$.
- $\boldsymbol{A}$ y $\boldsymbol{B}$ han obtenido el secreto compartido $\boldsymbol{S_A = S_B = d_A * d_B * G = d_B * d_A * G}$ ahora pueden intercambiar datos con algún tipo de cifrado.

El protocolo ECDH se detalla de mejor manera en el apartado **Algoritmo ECDH**, este protocolo es muy seguro, ya que se basa en la dificultad de determinar el **logaritmo discreto de una curva elíptica**[13], que es un problema matemático que se considera muy difícil de

[13] El logaritmo discreto es una función que es **difícil de calcular**, y **difícil de invertir**, lo que significa que si una persona conoce un punto y su logaritmo discreto, no puede calcular fácilmente el punto original. Esta propiedad hace que el logaritmo discreto sea útil para la implementación de sistemas criptográficos seguros y eficientes.

resolver. Además, ECDH es muy eficiente en términos de rendimiento y requisitos de memoria, lo que lo hace ideal para su uso en dispositivos con recursos limitados, como dispositivos móviles y sensores inalámbricos.

3.3.1.5 ElGamal Elíptico

ElGamal Elíptico es un esquema de cifrado de **Criptografía Asimétrica** (Clave Pública y Privada) basada en curvas elípticas, aunque existe una variante que utiliza una *clave secreta compartida* y es esta variante la que se implementa en el sistema. Este esquema se utiliza para cifrar y descifrar mensajes de forma segura entre dos partes que se comunican a través de un canal inseguro, aunque se basan en la misma tecnología que ECDH, este se utiliza para propósitos diferentes y los dos se utilizan en diferentes etapas de un proceso de comunicación segura. El proceso es el siguiente:

- Se necesita los **Parámetros de dominio de una curva elíptica**, la clave secreta compartida $\boldsymbol{S}$ obtenida tras usar el algoritmo **Elliptic-Curve** Diffie-Hellman (ECDH) y el mensaje a cifrar $\boldsymbol{msj}$.
- Se representa el mensaje a cifrar $\boldsymbol{msj}$ como puntos $\boldsymbol{M}$ de la curva, en este proceso se puede representar uno o más caracteres dentro de un mismo punto, todo depende del tamaño del campo.
- A cada punto del mensaje a cifrar $\boldsymbol{M}$ le sumamos el punto de la clave secreta compartida $\boldsymbol{S}$ obteniendo el mensaje cifrado en forma de puntos $\boldsymbol{C = M + S}$.
- Los puntos resultantes $\boldsymbol{C}$ son enviados por una de las partes para su descifrado.
- La segunda parte recibe $\boldsymbol{C}$ y utiliza $\boldsymbol{S}$ para encontrar $\boldsymbol{-S}$, el cual es $\boldsymbol{-S = (x, -y)\ (mod\ p)}$
- Una vez obtenido $\boldsymbol{-S}$, es posible encontrar $\boldsymbol{M}$, el cual es $\boldsymbol{M = C - S}$, o también expresado como $\boldsymbol{M = C + (-S)}$.
- Finalmente se convierte el conjunto de puntos $\boldsymbol{M}$ en el mensaje enviado $\boldsymbol{msj}$.

El esquema ElGamal Elíptico es muy seguro, ya que se basa en las mismas propiedades que ECDH, teniendo igual importancia en términos de seguridad y eficiencia. Además, ElGamal Elíptico es compatible con una amplia variedad de dispositivos y plataformas, lo que facilita su uso en diferentes entornos.

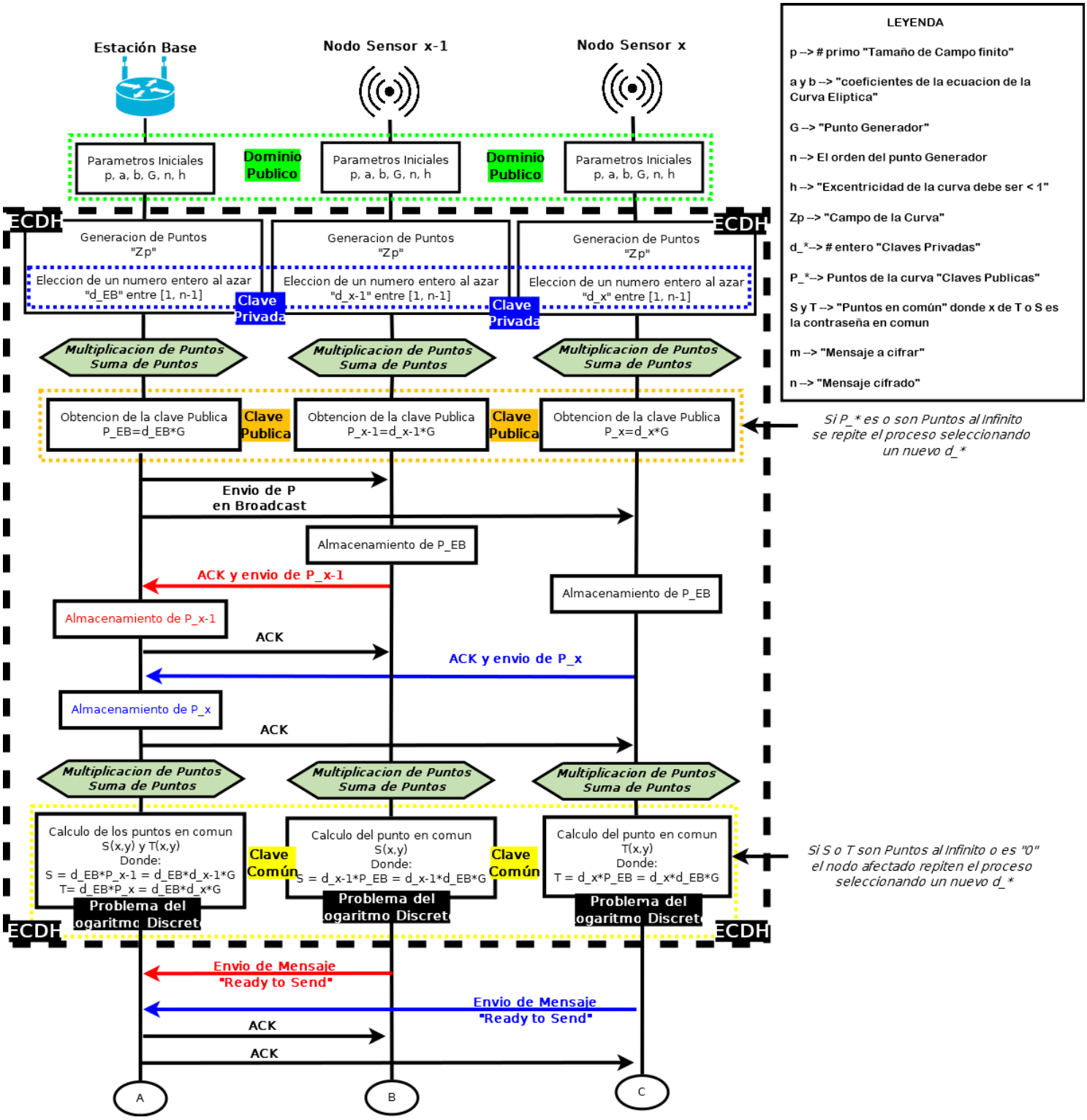
3.3.2 Diagrama de Funcionamiento

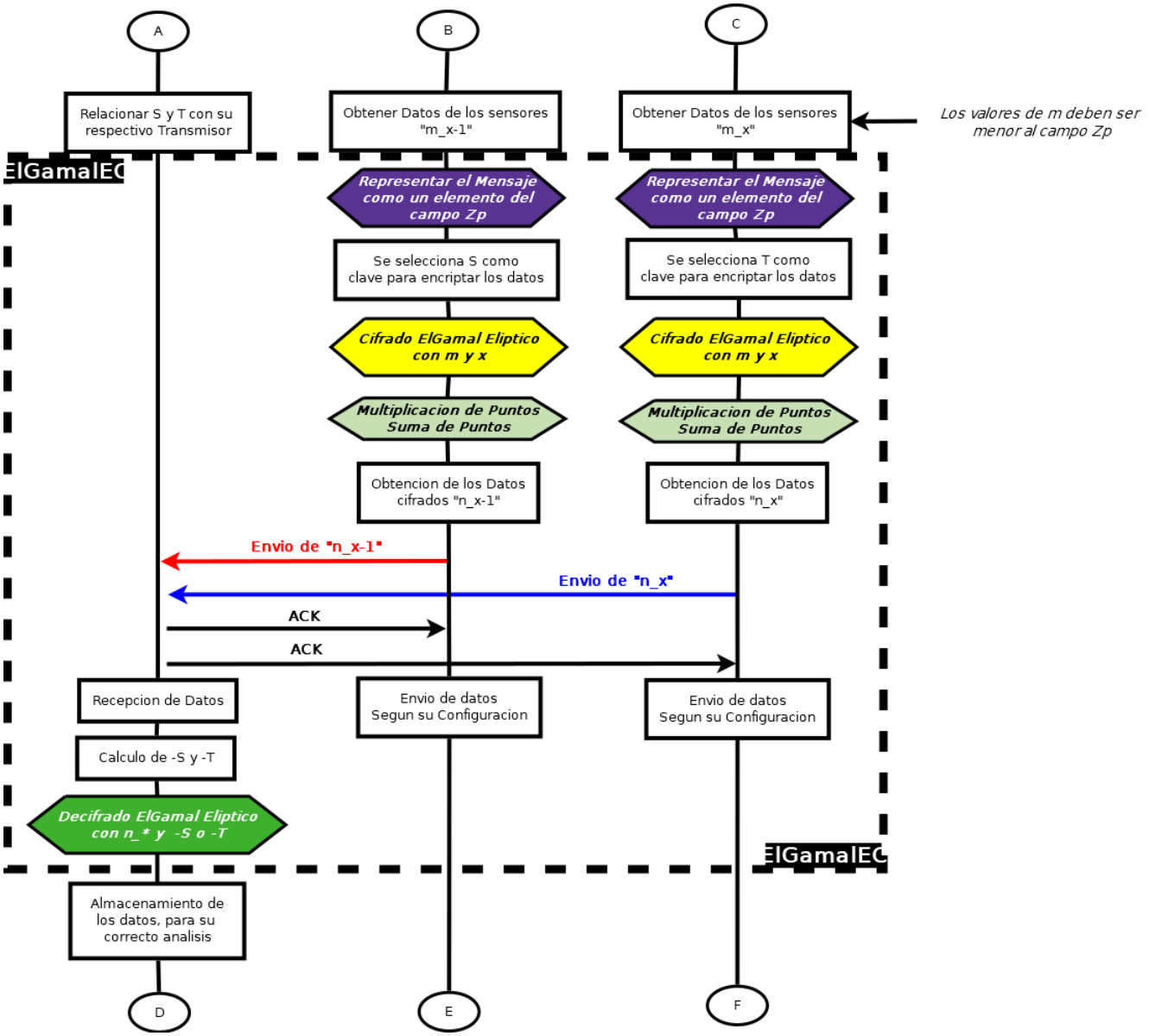
La **Figura 22** esta segmentada en partes y muestra el diagrama de funcionamiento de todo el sistema de cifrado para una red WSN, indicando de manera detallada todos los procesos que cumplirá el esquema. Los procesos esenciales se muestran en el apartado **Diagrama de Bloques del sistema** y estos son: Los parámetros de dominio de la curva elíptica, la generación e intercambio de claves ECDH y el cifrado/descifrado del mensaje utilizando ElGamal Elíptico. En el esquema se muestran 2 nodos sensores y una estación base cada uno deberá tener los mismos **Parámetros de dominio** de una curva elíptica **(recuadro verde)** y de estos depende todo el proceso del sistema, los tamaños de clave, el nivel de seguridad, etc. A continuación la estación base en conjunto con los nodos realizan la **Generación de Claves** públicas y privadas **(Recuadro azul y naranja)**, El proceso continúa implementando el algoritmo de intercambio de claves y estableciendo una clave en común usando **Elliptic-Curve Diffie-Hellman (ECDH) (Recuadro Negro ECDH)**.

Una vez establecido la clave en común, se utiliza esta para el cifrado de datos usando la variante de **ElGamal Elíptico (Recuadro Negro ElGamalEC)**. Hay muchas funciones que realizan múltiples procesos **(Hexágonos irregulares alargados)**, estos representan todos los algoritmos y funciones matemáticas presentes en el proceso, se detallan más adelante en las siguientes secciones, finalmente se observa el envío de datos **(Flechas de colores)** las cuales nos ayuda a comprender el orden de cada proceso y cada una de las secuencias. Finalmente la parte de la actualización de contraseña **(Recuadro Negro Actualización)** se realiza con el objetivo de mantener un sistema seguro de datos utilizando propiedades matemáticas.

Figura 22

Diagrama de Funcionamiento del esquema de seguridad para una red de sensores utilizando ECC


Estación Base
Nodo Sensor x-1
Nodo Sensor x
Parametros Iniciales p, a, b, G, n, h
Dominio Publico
Parametros Iniciales p, a, b, G, n, h
Dominio Publico
Parametros Iniciales p, a, b, G, n, h
ECDH
Generacion de Puntos "Zp"
Eleccion de un numero entero al azar "d_EB" entre [1, n-1]
Clave Privada
Generacion de Puntos "Zp"
Eleccion de un numero entero al azar "d_x-1" entre [1, n-1]
Clave Privada
Generacion de Puntos "Zp"
Eleccion de un numero entero al azar "d_x" entre [1, n-1]
Multiplicacion de Puntos Suma de Puntos
Multiplicacion de Puntos Suma de Puntos
Multiplicacion de Puntos Suma de Puntos
Obtencion de la clave Publica P_EB=d_EB*G
Clave Publica
Obtencion de la clave Publica P_x-1=d_x-1*G
Clave Publica
Obtencion de la clave Publica P_x=d_x*G
Envio de P en Broadcast
Almacenamiento de P_EB
ACK y envio de P_x-1
Almacenamiento de P_EB
Almacenamiento de P_x-1
ACK
ACK y envio de P_x
Almacenamiento de P_x
ACK
Multiplicacion de Puntos Suma de Puntos
Multiplicacion de Puntos Suma de Puntos
Multiplicacion de Puntos Suma de Puntos
Calculo de los puntos en comun S(x,y) y T(x,y) Donde: S = d_EB*P_x-1 = d_EB*d_x-1*G T= d_EB*P_x = d_EB*d_x*G
Clave Común
Calculo del punto en comun S(x,y) Donde: S = d_x-1*P_EB = d_x-1*d_EB*G
Clave Común
Calculo del punto en comun T(x,y) Donde: T = d_x*P_EB = d_x*d_EB*G
Problema del ogaritmo Discret
Problema del ogaritmo Discret
Problema del ogaritmo Discret
ECDH
Envio de Mensaje "Ready to Send"
Envio de Mensaje "Ready to Send"
ACK
ACK
A
B
C
LEYENDA
p –> # primo "Tamaño de Campo finito"
a y b –> "coeficientes de la ecuacion de la Curva Eliptica"
G –> "Punto Generador"
n –> El orden del punto Generador
h –> "Excentricidad de la curva debe ser < 1"
Zp –> "Campo de la Curva"
d_*–> # entero "Claves Privadas"
P_*–> Puntos de la curva "Claves Publicas"
S y T –> "Puntos en común" donde x de T o S es la contraseña en comun
m –> "Mensaje a cifrar"
n –> "Mensaje cifrado"
Si P_* es o son Puntos al Infinito se repite el proceso seleccionando un nuevo d_*
Si S o T son Puntos al Infinito o es "0" el nodo afectado repiten el proceso seleccionando un nuevo d_*



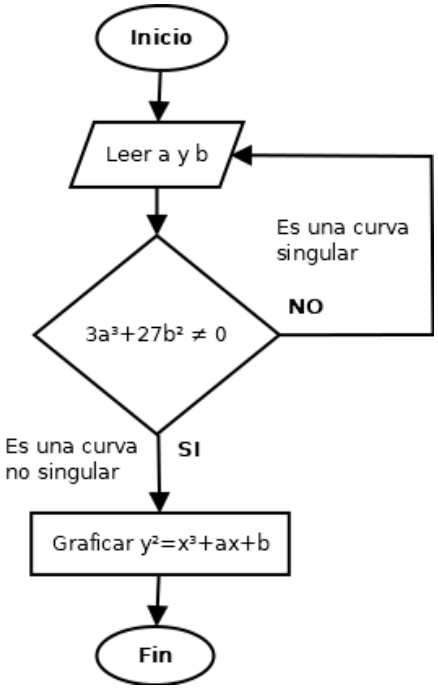
A continuación, se muestran los bloques de trabajo realizados para este capítulo con su respectiva evolución e integración entre ellos.

3.3.3 Generación de curvas elípticas

Este paso se utiliza para visualizar los diferentes tipos de curvas elípticas sobre los números reales vistos en la **Figura 15**, utilizando "la Ecuación reducida de Weierstrass" **(Ec. 2)** la cual no es suficiente para determinar si una curva es elíptica, por lo que es necesario considerar también la **(Ec. 3)** "Discriminante de la ecuación reducida de Weierstrass". En la **Figura 23** se muestra el diagrama de flujo para la generación de curvas elípticas, este diagrama junto con el ***Algoritmo 1*** deberán ser interpretados en un software para la generación de gráficos a partir de funciones, como **GeoGebra**[14] o MATLAB .

Figura 23

Diagrama de flujo para la generación de la curva elíptica sobre $\mathbb{R}$



Nota: En la Figura se observa el diagrama de flujo para la generación de una curva elíptica en el dominio de los números reales, a partir de la ecuación reducida de Weierstrass, con la respectiva comprobación de la singularidad de la curva.

A continuación, se indica la respectiva interpretación y el ***Algoritmo 1*** , donde:

[14] GeoGebra es un software matemático gratuito y de código abierto que permite a los usuarios crear y analizar objetos matemáticos en dos y tres dimensiones

1. Se ingresa los coeficientes a y b, de la ecuación $y^2 = x^3 + ax + b$, los valores de a, $b \in \mathbb{R}$.
2. Verificar el valor de $4a^3 + 27b^2 \neq 0$. Si efectivamente es diferente de cero se trata de una curva elíptica caso contrario se trata de una curva singular y se vuelve a pedir los datos de inicio.
3. Se grafica la ecuación con los parámetros ingresados y muestra el aviso con el tipo de curva identificada.

```
Input: (int) a, b
Output: (String) aviso, (Draw) gráfico

(1)  gráfico ← y^2 = x^3 - a*x - b
(2)  if (4*a^3 + 27*b^2) != 0
(3)     aviso ← (Es una curva elíptica no singular)
(4)     plot (gráfico)
(5)  else
(6)     aviso ← (Es una curva singular)
(7)     Return Input
(8)  print (aviso)
```

Algoritmo 1. *Generación de una curva elíptica sobre R*

Este ***Algoritmo 1*** solo se utiliza para representar a las curvas elípticas de una manera visual, no se utiliza en el funcionamiento del esquema de cifrado ligero, los gráficos han sido utilizados con el fin de ayudar a comprender las curvas elípticas y sus diferentes campos en los cuales pueden ser expresadas.

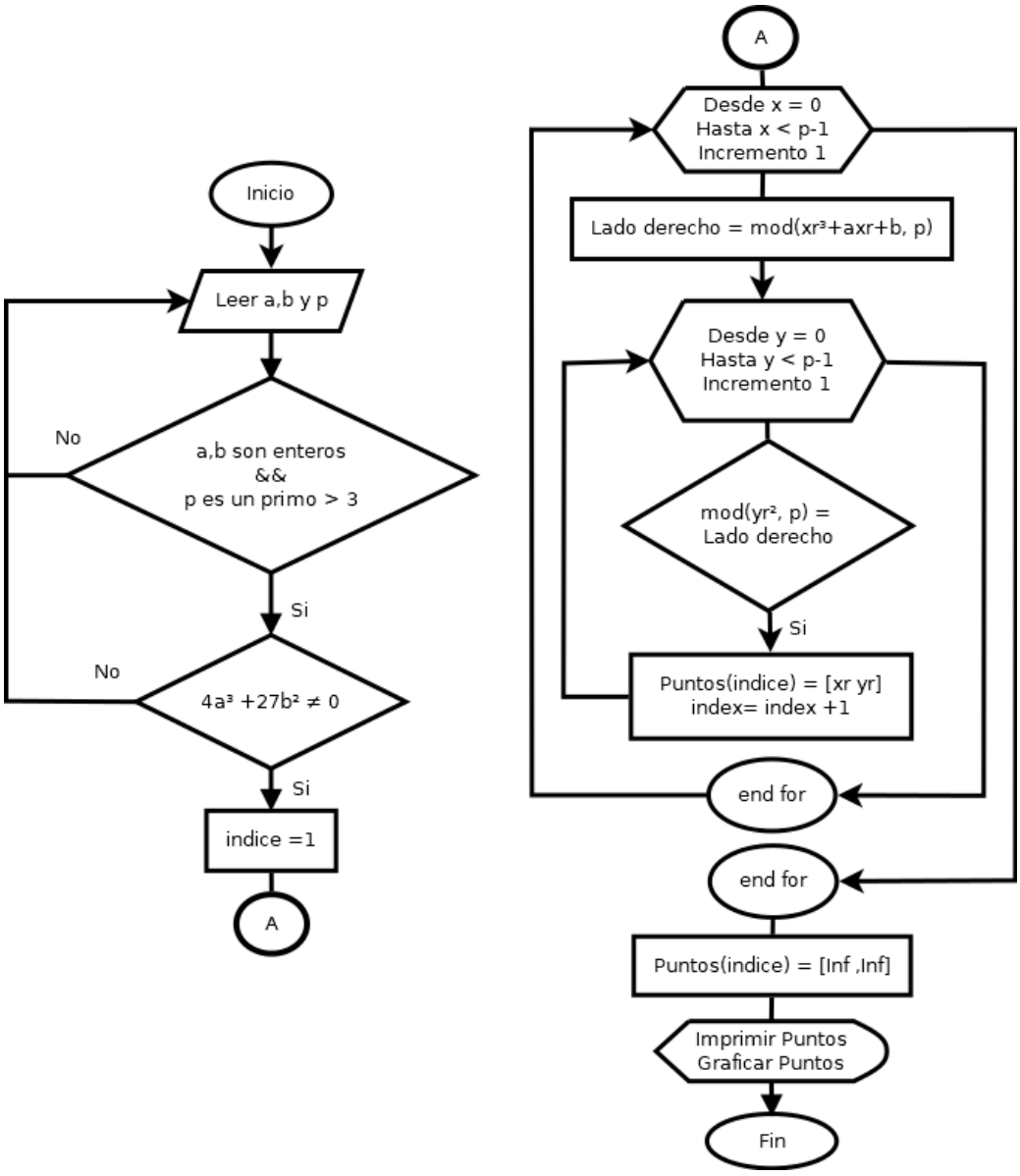
3.3.4 Generación de puntos de la curva elíptica

Este tipo de criptografía se implementa sobre cuerpos finitos primos descrita en la **(Ec. 5)**, para obtener los puntos y el gráfico de estos, se utiliza una función específica, la **Figura 24** muestra el diagrama de flujo de esta función, la figura se ha dividido en dos partes donde la primera parte determina si la curva es singular y si el tamaño del campo $\boldsymbol{p}$ es un numero primo mayor que **3**, esto se realiza para que la curva no pierda su propiedad de no-singularidad; si los parámetros iniciales son correctos, la segunda parte se encarga de la obtención de los puntos probando cada valor y remplazándolo en la curva hasta que este pertenezca a dicha curva,

haciendo que se obtenga una matriz con la cantidad total de puntos computados, más el punto al infinito.

Figura 24

Diagrama de Flujo para Generar puntos en un Zp primo > 3.



Nota: En la Figura se observa el diagrama de flujo para la generación de puntos de una curva elíptica en un campo Zp, la primera parte está enfocada a la verificación de los parámetros iniciales (números enteros, el valor primo, o a detectar la singularidad de la curva); La segunda parte calcula los puntos comparando los valores en la ecuación de Weierstrass y guardando estos en una matriz.

A continuación, se detalla la implementación del ***Algoritmo 2*** llamado "**puntosCurvaEliptica(…)**" que devuelve el conjunto de puntos de la curva en forma de vector y en forma gráfica.

1. Se ingresa los coeficientes a, b y el primo p de la ecuación $y^2 = x^3 + ax + b\ (mod\ p)$ como parámetros de entrada. Se verifica que $a, b\ \in\ \mathbb{Z}$ con la función mod.
2. Luego se verifica que p sea primo y mayor a 3, también si $4a^3 + 27b^2 = 0$, se trata de una curva singular.
3. Se inicializa el contador $index\ = 1$ y se genera un lazo con el ciclo for para los valores de x que va desde 0 hasta $p-1$ en donde se calcula todos los posibles valores de x en el lado derecho de la ecuación de la curva: $x^3 + ax + b\ (mod\ p)$.
4. Dentro del ciclo se genera un nuevo ciclo for para los valores de y que va desde 0 hasta $p - 1$ en donde se calcula todos los valores de y en la ecuación: $y^2\ (mod\ p)$.
5. Se compara con el lado derecho calculado previamente, si el lado derecho e izquierdo coinciden, en la matriz $Puntos$ [][] se guarda el par (x, y) y se incrementa la variable $index$ para el siguiente par ordenado, se repite este proceso hasta terminar los $p-1$ valores.
6. Al terminar ambos ciclos se añade el punto al infinito $\boldsymbol{O}$ a la matriz $Puntos(index, :)$.
7. Finalmente, se gráfica y se presenta el vector con todos los puntos de la curva más el punto al infinito.

```
Input: (int) a, b, p
Output: (int) Puntos[][]

(1)   if (mod(a,1)!=0 || mod(b,1)!=0)
(2)       print ('El valor de a y de b deben ser enteros');
(3)       Return Input
(4)   elseif (primo(p)!=1 || p<4)
(5)       print ('p debe ser un numero primo mayor que 3');
(6)       Return Input
(7)   elseif (4*a^3+27*b^2 == 0)
(8)       print ('Es una curva singular');
(9)       Return Input
(10)  else
(11)      indice = 1;
(12)      for x=0:p-1
(13)          ladoDerecho = mod(x^3 + a*x + b, p);
(14)          for y=0:p-1
(15)              if mod(y^2,p) == ladoDerecho
(16)                  Puntos(indice, :) = [x y];
(17)                  indice = indice + 1;
(18)              end
(19)          end
(20)      end
```

```
(21)        Puntos(indice, :) = [Inf Inf];
(22)    end
(23)    plot(Puntos)
```

Algoritmo 2. *generación de puntos de curva sobre $\mathbb{F}p$*

3.3.5 Suma de puntos en curvas elípticas

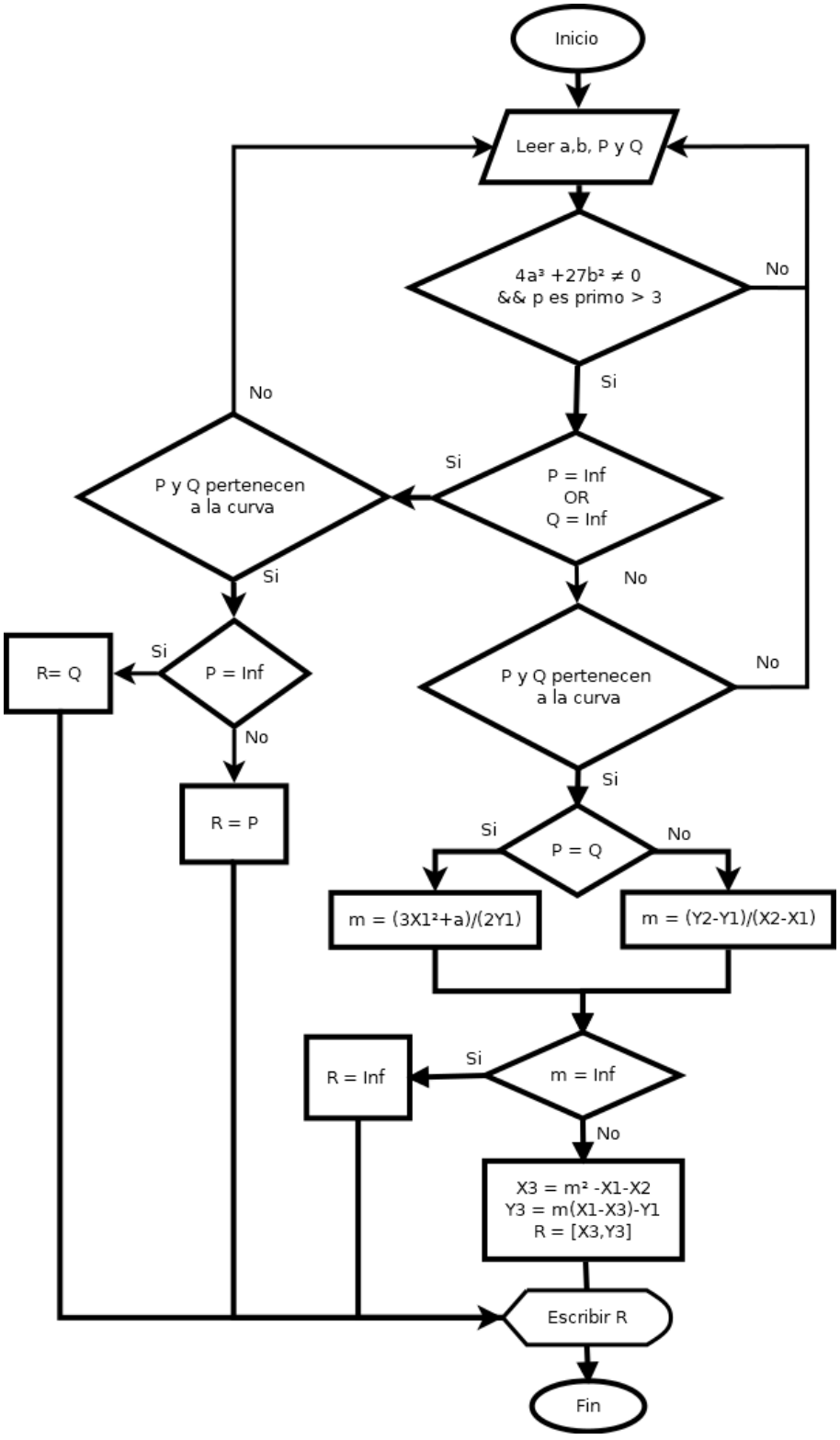
Las operaciones de grupo sobre curvas elípticas descritas en el apartado **Orden del Grupo Cíclico** contemplan la operación suma de puntos de una curva elíptica sobre los campos finitos primos. Para realizar la suma de dos puntos $P = (x1, y1)$ y $Q = (x2, y2)$ se usa la **(Ec. 8)**, que calcula las coordenadas $(x3, y3)$ del punto R y la pendiente de la cuerda o tangente m. Para aplicar esta aritmética se necesita una idea clara de lo que significa $\frac{y}{x} = y.x^{-1}$. En aritmética modular para realizar esta división se debe encontrar el inverso multiplicativo de un número, también llamado el "**Algoritmo Extendido de Euclides** AEE**)**", y luego realizar una sola multiplicación. En la **Figura 25** se muestra el diagrama de flujo para la operación de suma de dos puntos sobre campos finitos primos y en la **Figura 26** el diagrama de flujo para la implementación del AEE.

La función suma de puntos de una curva elíptica Zp expresada en el ***Algoritmo 3*** hace un llamado directo a la función AEE expresada en el ***Algoritmo 4***, para resolver la división modular, convirtiendo a estas dos funciones en una misma, mejorando asi la eficiencia del algoritmo criptográfico.

La suma de puntos en la curva elíptica es la ***base de la criptografía de curva elíptica***, sin embargo, la implementación de este algoritmo puede ser más complicado que otros algoritmos criptográficos como RSA. La suma de puntos en la curva elíptica es una operación matemática compleja y costosa en términos de tiempo y recursos de computación. A pesar de estas complicaciones, la criptografía de curva elíptica se ha convertido en una de las principales alternativas a los sistemas criptográficos, siendo más eficiente en términos de almacenamiento, ancho de banda y velocidad de procesamiento que otros sistemas criptográficos basados en el problema del logaritmo discreto.

Figura 25

Diagrama de flujo para la suma de puntos en curva elíptica sobre Zp



Nota: En la Figura se observa el diagrama de flujo que permite resolver la operación suma de puntos en curva elíptica sobre Zp, esta es la operación más importante y esta función será llamada múltiples veces para el proceso de intercambio de claves y cifrado de datos.

A continuación, se explica el ***Algoritmo 3*** llamado "**sumaPuntosCurvaEliptica(…)**" el cual permite resolver la operación de suma de dos puntos en curvas elípticas sobre Zp.

1. Se ingresa los coeficientes a, b y el primo p de la ecuación $y^2 = x^3 + ax + b\ (mod\ p)$. Se verifica que $p > 3$ sea primo, que $a, b\ \in\ \mathbb{Z}$ y si la curva es no singular.
2. Ingresados los puntos $P\ y\ Q$ junto con los parámetros de la curva, se verifica si $P\ o\ Q$ son puntos al infinito, de ser así se cumple la existencia del elemento neutro.
3. Se comprueba que los puntos $P\ y\ Q$ pertenezcan a la curva elíptica, sustituyendo los pares ordenados de los puntos en la ecuación $y^2 = x^3 + ax + b\ (mod\ p)$.
4. Si los puntos $P\ y\ Q$ son iguales para calcular el valor de m según el caso de duplicación de un punto descrito en la **(Ec. 10)**. Luego se comprueba que la pendiente no tienda al infinito para calcular las coordenadas del punto R. Si m tiende al infinito, R es el punto al infinito $\boldsymbol{O}$.
5. Si los puntos P y Q no son iguales se calcula el valor de m según el caso de suma de dos puntos y se codifica según corresponda la Ecuación **(Ec. 8)**. Después, se comprueba que la pendiente no tienda al infinito para calcular las coordenadas de R. Si m tiende al infinito, R es el punto al infinito $\boldsymbol{O}$.
6. Al revisar los pasos 4 y 5, se evidencia que la operación división no se puede aplicar de forma normal en aritmética modular ya que $\frac{y}{x} = y.x^{-1}$ o lo que se traduce multiplicar por el inverso multiplicativo. La implementación de la función $\boldsymbol{AEE(a, b)}$ permite buscar el inverso multiplicativo de un número, esta función se explica más adelante en la sección **Algoritmo Extendido de Euclides AEE**, y se observa su diagrama de funcionamiento en la **Figura 26**.

```
Input: (int) a, b, p, (Punto) P, Q
Output: (Punto) R

(1)   if (mod(a,1)!=0 || mod(b,1)!=0)
(2)       print ('a y/o b no son enteros')
(3)       Return Input
(4)   elseif (primo(p)!=1 || p<4)
(5)       print ('p debe ser mayor entero mayor que 3')
(6)       Return Input
(7)   elseif (4*a^3+27*b^2 == 0)
(8)       print ('Es una curva singular')
(9)       Return Input
(10)  end
(11)  if (P(1) == inf || P(2) == inf)
(12)      if (mod(Q(2)^2,p)!= mod(Q(1)^3+a*Q(1)+b, p))
(13)          print ('El punto Q debe pertenecer a la curva')
(14)      else
```

```
(15)            R = Q;
(16)            Return;
(17)        end
(18)    end
(19)    if (Q(1) == inf || Q(2) == inf)
(20)        if mod(P(2)^2,p)!= mod(P(1)^3 + a*P(1)+b, p)
(21)            print ('El punto P debe pertenecer a la curva')
(22)        else
(23)            R = P;
(24)            Return;
(25)        end
(26)    end
(27)    if (P == Q)
(28)        dy = mod(3*P(1)^2+a,p);
(29)        dx = mod(2*P(2),p);
(30)        if (maxCdiv(dx,p)==1)
(31)            m = mod(dy*AEE(dx,p),p);
(32)            x3 = mod(m^2 - P(1) - Q(1),p);
(33)            y3 = mod(m*(P(1) - x3) - P(2),p);
(34)            R = [x3 y3];
(35)        else
(36)            R = [inf inf];
(37)        end
(38)    else
(39)        dy = mod(Q(2)-P(2),p);
(40)        dx = mod(Q(1)-P(1),p);
(41)        if (maxCdiv(Q(1)-P(1),p) == 1)
(42)            m = mod(dy*AEE(dx,p),p);
(43)            x3 = mod(m^2 - P(1) - Q(1),p);
(44)            y3 = mod(m*(P(1) - x3) - P(2),p);
(45)            R = [x3 y3];
(46)        else
(47)            R = [inf inf];
(48)        end
(49)    end
```

Algoritmo 3. *Suma de puntos en Zp*

3.3.5.1 Algoritmo Extendido de Euclides AEE

El algoritmo extendido de Euclides es una técnica eficiente para encontrar el **máximo común divisor**[15] (MCD) de dos números enteros y también puede ser utilizado para encontrar el inverso multiplicativo de un número en un campo finito utilizando las **ecuaciones de Bezout**[16], que establecen que para cualquier par de números enteros $\boldsymbol{a}$ y $\boldsymbol{b}$, existe una combinación lineal de $\boldsymbol{a}$ y $\boldsymbol{b}$ que da como resultado su máximo común divisor. Es decir, existen enteros $\boldsymbol{x}$ e $\boldsymbol{y}$ tales

[15] MCD es un término utilizado en matemáticas para describir el número más grande que es un factor común de dos o más números enteros.

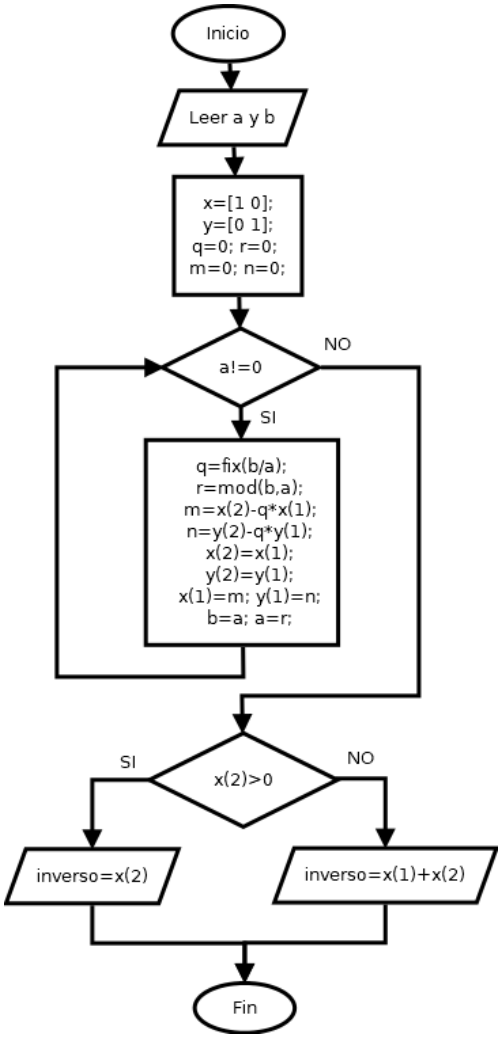
[16] Las ecuaciones de Bezout son una propiedad fundamental de los números enteros y tienen numerosas aplicaciones en diferentes áreas de las matemáticas, geometría algebraica y criptografía.

que $\boldsymbol{ax + by = MCD(a, b)}$. Estas ecuaciones se utilizan en el algoritmo extendido de Euclides; Se necesita encontrar los enteros $\boldsymbol{x}$ e $\boldsymbol{y}$ tales que $\boldsymbol{ax + my = 1}$, donde $\boldsymbol{a}$ es el número al que queremos encontrar el inverso multiplicativo, $\boldsymbol{m}$ es el módulo y el **MCD** de $\boldsymbol{a}$ y $\boldsymbol{m}$ debe ser $\boldsymbol{1}$, lo que asegura que el inverso multiplicativo exista.

Al utilizar el AEE, se pueden encontrar los valores de $\boldsymbol{x}$ e $\boldsymbol{y}$ que satisfacen la ecuación $\boldsymbol{ax + my = MCD(a, m)}$, y a partir de esto, se puede encontrar el inverso multiplicativo de $\boldsymbol{a}$ módulo $\boldsymbol{m}$ mediante iteraciones matemáticas. En particular, si $\boldsymbol{x}$ es positivo, entonces $\boldsymbol{x}$ es el inverso multiplicativo de $(\boldsymbol{a \bmod m})$. Si $\boldsymbol{x}$ es negativo, se puede calcular el inverso multiplicativo sumándolo $\boldsymbol{m}$ hasta que se vuelva positivo. La **Figura 26** muestra el diagrama de flujo para la generación del inverso multiplicativo, tomando como entrada los valores de $\boldsymbol{a}$ y $\boldsymbol{b}$; donde $\boldsymbol{a}$ es el número del cual se quiere obtener el inverso y $\boldsymbol{b}$ es el módulo.

Figura 26

Diagrama de flujo para la obtención del inverso multiplicativo



A continuación, se explica la implementación del ***Algoritmo 4*** llamado "**AEE(…)**" para calcular el inverso multiplicativo de un número $\boldsymbol{a}$ en un módulo $\boldsymbol{b}$. El algoritmo funciona de la siguiente manera:

1. Se inicializan los arreglos x e y con valores constantes y se inicializan las variables q, r, m y n con valores de cero.
2. Se ejecuta un bucle *while* hasta que a sea igual a cero.
3. Dentro del bucle se calcula el cociente q y el resto r de la división de b entre a.
4. Se calculan m y n utilizando las ecuaciones de Euclides.
5. Se almacenan temporalmente los valores de $x(1)$ e $y(1)$ en $x(2)$ e $y(2)$, respectivamente.
6. Se almacenan temporalmente los valores de m y n en $x(1)$ e $y(1)$, respectivamente.
7. Se actualizan los valores de b y a con a y r, respectivamente.
8. Al final del bucle, si $x(2)$ es mayor que cero, entonces el inverso es $x(2)$. Si $x(2)$ es menor o igual a cero, entonces el inverso es $x(1) + x(2)$.

```
Input: (int) a, b
Output: (int) inverso

(1)   x=[1 0];
(2)   y=[0 1];
(3)   q=0; r=0; m=0; n=0;
(4)   while a!=0
(5)      q=fix(b/a);
(6)      r=mod(b,a);
(7)      m=x(2)-q*x(1);
(8)      n=y(2)-q*y(1);
(9)      x(2)=x(1);
(10)     y(2)=y(1);
(11)     x(1)=m; y(1)=n;
(12)     b=a; a=r;
(13)  end
(14)  if (x(2)>0)
(15)     inverso=x(2)
(16)  else
(17)     inverso=x(1)+x(2)
(18)  end
(19)  end
```

Algoritmo 4. *Algoritmo Extendido de Euclides o Inverso Multiplicativo*

3.3.6 Multiplicación escalar en curvas elípticas

La multiplicación de un escalar $\boldsymbol{k}$ por un punto $\boldsymbol{P}$ de una curva elíptica se realiza mediante el algoritmo de duplicar y sumar, también conocido como **expansión de Horner**[17]. Este algoritmo es muy eficiente para operar en curvas elípticas sobre $\boldsymbol{Zp}$, ya que utiliza operaciones básicas como la duplicación y la suma para calcular el resultado de forma rápida.

Sean $\boldsymbol{E}$ una curva eliptica y $\boldsymbol{P}$ un punto sobre la curva. Si $\boldsymbol{k}$ es un entero positivo definimos a la potencia $\boldsymbol{k}$ de $\boldsymbol{P}$, denotada por $\boldsymbol{kP}$, la **(Ec. 12)** indica la cantidad de sumas, del punto $\boldsymbol{P}$ consigo mismo para obtener el valor de $\boldsymbol{kP}$:

$$kP = (P + P + \cdots P) \rightarrow k\ veces \qquad \text{(Ec. 12)}$$

Por lo tanto, se puede convertir al entero $\boldsymbol{k}$ en su expansión binaria para utilizar posteriormente la expansión de Horner, tal que $\boldsymbol{k} = (\boldsymbol{b_{m-1}}\ \boldsymbol{b_{m-2}}\ \boldsymbol{b_{m-3}}\ \ldots \boldsymbol{b_2}\ \ \boldsymbol{b_1}\ \boldsymbol{b_0})$, donde $\boldsymbol{m}$ es tal que $\boldsymbol{b_{m-1}} = \boldsymbol{1}$ representando al bit mas significativo de $\boldsymbol{k}$, Entonces se expresa como **(Ec. 13)**: (Garcia, 2010)

$$k = (b_{m-1}2^{m-1} + b_{m-2}2^{m-2} + \cdots + b_2 2^2 + b_1 2 + b_0) \qquad \text{(Ec. 13)}$$

Utilizando la expansión de Horner, se comprueba que los valores de toda esta operación hacen relación a una duplicación de $\boldsymbol{b_{m-1}}$ más la suma del término anterior $\boldsymbol{b_{m-2}}$, y el proceso continúa con todos los bits:

$$k = (\ldots((\boldsymbol{b_{m-1}}\boldsymbol{2} + \boldsymbol{b_{m-2}})2 + \cdots + b_1)2 + b_0) \qquad \text{(Ec. 14)}$$

Para representar la multiplicación de un punto por un escalar, se extrapola la idea obteniendo la **(Ec. 15)** donde se aprecia la duplicación y suma de los puntos.

[17] La expansión de Horner es una técnica para evaluar polinomios que permite reducir el número de multiplicaciones necesarias para calcular el valor de un polinomio en un punto dado.

$$kP = 2(\dots 2(2(2(bm_{-1}P) + bm_{-2}P) + \cdots + b_1P) + b_0) \quad \text{(Ec. 15)}$$

Por consiguiente, la **(Ec. 16)** indica que se debe aplicar una ***duplicación*** cuando el bit tratado es igual a **0,** o una ***duplicación y suma*** al anterior si se trata de un bit igual a **1**. Construyendo así la sucesión de puntos sobre la curva. (Garcia, 2010)

$$P_i = \begin{cases} 2P_{i-1} & si\ b_{m-i} = 0 \\ 2P_{i-1} + P & si\ b_{m-i} = 1 \end{cases} \quad \text{(Ec. 16)}$$

Para entender de mejor manera su funcionamiento se toma como ejemplo el valor de $\boldsymbol{k = 11}$ con su representación binaria $\boldsymbol{k = 1011_2}$. Esta representación binaria se puede convertir en una suma de potencias de dos:

$$11 = (1 * 2^3) + (0 * 2^2) + (1 * 2^1) + (1 * 2^0)$$

$$11 = 2^3 + 2^1 + 2^0$$

Ahora se toma el valor de $\boldsymbol{P}$ el cual se quiere multiplicar por $\boldsymbol{k = 11}$.

$$11P = 2^3P + 2^1P + 2^0P$$

Por lo que el algoritmo indica que:

Primer bit de $k = 1$ (Bit menos significativo)

- Siempre se empieza con el punto al infinito O
- Duplicar "O" obteniendo $2 * O = O$
- Sumar P a "O" obteniendo $P + O = P$, tambien expresado como $P = 2^0P$

Segundo bit de $k = 1$

- Duplicar "P" obteniendo $2 * P = 2P$. tambien expresado como $2P = 2^1P$.
- Sumar $2P$ a "P" para obtener el resultado de $2^1P + 2^0P$.

Tercer bit de $k = 0$

- Duplicar "$2P$" para obtener $2 * (2P) = 4P$, tambien expresado como $4P = 2^2P$

- No se realiza ninguna suma que implique 2^2P, asi que se mantiene $2^1P + 2^0P$.

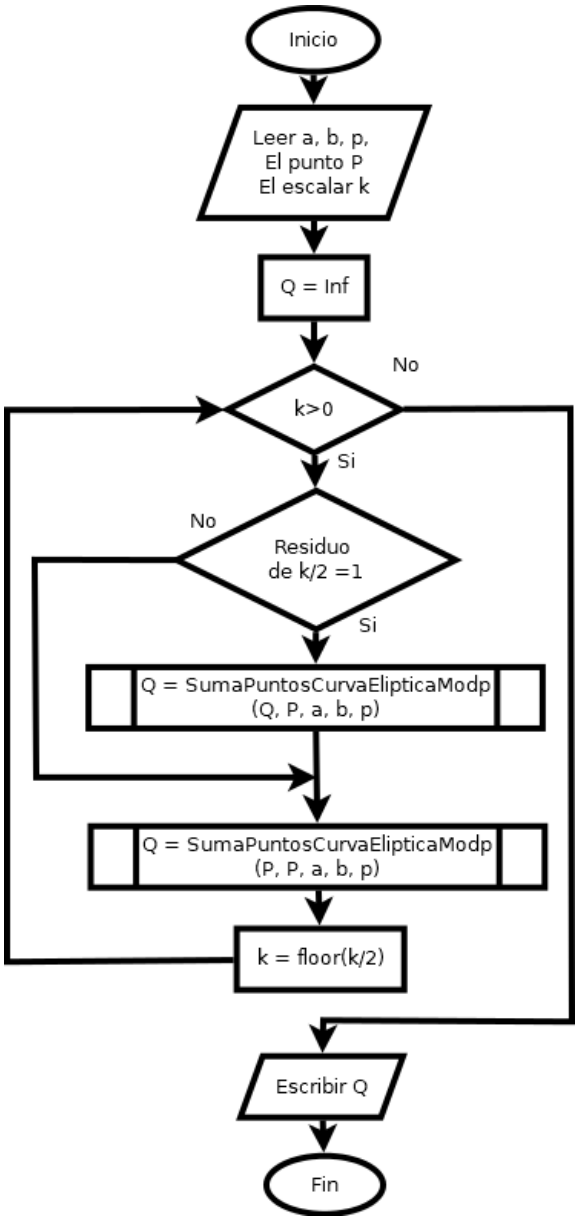
Cuarto bit de $k = 1$ (Bit más significativo)

- Duplicar "$4P$" para obtener $2 * (4P) = 8P$, tambien expresado como $8P = 2^3P$
- Sumar 2^3P a "$2^1P + 2^0P$" para obtener el resultado de $2^3P + 2^1P + 2^0P$.

En la **Figura 27** se muestra el diagrama de flujo de este algoritmo para las curvas sobre Zp, utilizando la duplicacion y suma dependiendo del escalar $\boldsymbol{k}$.

Figura 27

Diagrama de flujo del algoritmo duplicar y sumar en una curva elíptica.



A continuación, se muestra la implementación del **Algoritmo 5** llamado "**multEscalarCurvaEliptica(…)**" para la multiplicación en curvas elípticas.

1. La implementación, hace uso del algoritmo duplicar y sumar, la única diferencia es que se llama a la función **sumaPuntosCurvaEliptica(Q, P, a, b, p)** para sumar y **sumaPuntosCurvaEliptica(P, P, a, b, p)** para duplicar, esta función esta descrita en diagrama de la **Figura 25** y el ***Algoritmo 3***.
2. El escalar k determina cuántas veces se añadirá el punto P a sí mismo (es decir, "duplicado").
3. La función inicializa primero el resultado Q al punto en el infinito, que se representa por las coordenadas $[inf\ inf]$.
4. La función luego entra en un bucle que continúa mientras k sea mayor que 0. Si el resto de k dividido por 2 es igual a 1 (indica que el bit menos significativo de k es 1), la función realiza la adición de puntos en Q y P utilizando la función **sumaPuntosCurvaEliptica**. El resultado de la adición de puntos se asigna a continuación a Q.
5. Después de cada iteración del bucle, el punto P se duplica utilizando la función **sumaPuntosCurvaEliptica**, y el valor de k se divide por 2 utilizando la división entera (función $roundDown - floor$). Este proceso continúa hasta que k se vuelva 0, en cuyo punto el bucle termina y el valor final de Q se devuelve como resultado de la función.

```
Input: (int) a, b, p, k (Punto) P
Output: (Punto) Q

(1)   Q = [inf inf];
(2)   while k > 0
(3)      if mod(k,2) == 1
(4)         Q = sumaPuntosCurvaEliptica(Q, P, a, b, p);
(5)      end
(6)      P = sumaPuntosCurvaEliptica(P, P, a, b, p);
(7)      k = roundDown(k/2);
(8)   end
```

Algoritmo 5. *Multiplicación de un escalar por un punto en curva elíptica sobre Zp*

3.3.7 Orden de la curva y de un punto

En criptografía de curva elíptica, el orden de una curva elíptica y el orden de un punto son importantes porque se utilizan para asegurar la seguridad de las claves. Una curva elíptica y un punto con un orden alto son más difíciles de romper que un orden bajo.

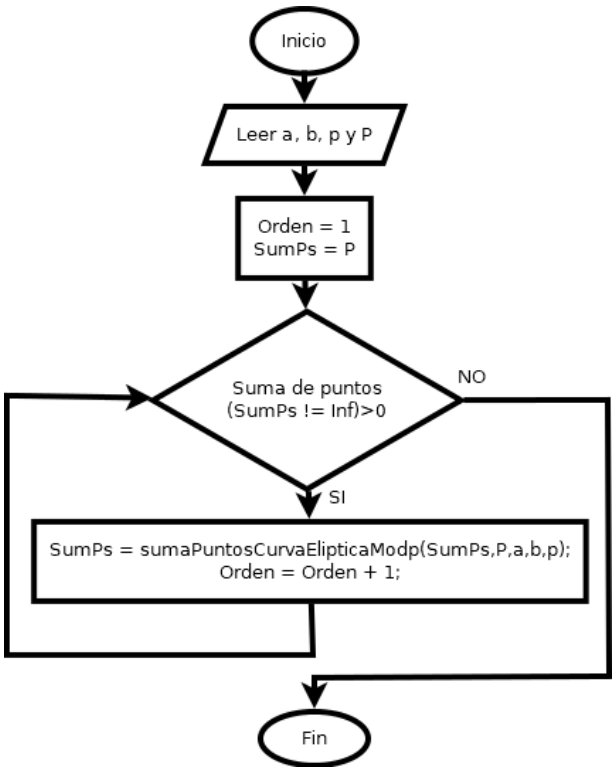
El orden de una curva elíptica es el número de puntos de la curva, esto se puede calcular utilizando el **Teorema de Hasse** visto en la sección **Parámetros de la Ecuación de la curva elíptica**. Por ejemplo, si una curva elíptica tiene un orden de 1000, significa que hay 1000 puntos diferentes en la curva.

El orden de un punto es el número de veces que se debe añadir el punto a sí mismo para obtener el punto en el infinito. Por ejemplo, si el orden de un punto es 1000, significa que se debe añadir el punto a sí mismo 1000 veces para obtener el punto en el infinito.

La **Figura 28** muestra el diagrama de flujo para calcular el orden de un punto en la curva elíptica, conocer este valor es crucial para el funcionamiento de ECC. El orden de un punto se relaciona directamente con el **Cofactor** de la curva elíptica; El cofactor es el *número de puntos* en la curva dividido por el *orden del punto* y afecta directamente a la seguridad de los algoritmos criptográficos, por ejemplo: si el cofactor es demasiado grande, el algoritmo para calcular la clave pública de ECC puede volverse muy lento, lo que puede tener un impacto en el rendimiento y la eficiencia del sistema, (Caiza, 2020) recomienda que es prefiere que el cofactor sea pequeño como 1, 2 o máximo 4. Donde la cantidad de puntos que se utiliza para el cifrado es aceptable y proporciona un buen equilibrio entre seguridad y eficiencia.

Figura 28

Diagrama de flujo para determinar el orden del punto en la curva



A continuación, se muestra la implementación del **Algoritmo 6** llamado "**ordenPuntoCurvaEliptica(…)**" para determinar el orden del punto en la curva eliptica.

1. Se ingresa el punto P junto con los parámetros de la curva, los valores de a,b y p.
2. Se establece el orden inicial en 1 y la suma de los puntos inicial en P.
3. Luego, utiliza un bucle $while$ para ir sumando el punto P a sí mismo, este es un proceso de fuerza bruta donde se calcula la suma de $P + P + P + \cdots$ hasta obtener como resultado el punto al infinito (que se representa como $[Inf\ Inf]$). Cada vez que se suma el punto, se aumenta el orden en 1.
4. El bucle $while$ continuará ejecutándose mientras la suma condicional de $SumPs$ sea mayor que cero. Esto se hace comparando $SumPs$ con el punto en el infinito utilizando la función $sum()$, que devuelve un valor mayor que cero si $SumPs$ no es igual al punto en el infinito.
5. Una vez que se ha llegado al punto en el infinito, se sale del bucle y se devuelve el orden final como resultado de la función.

```
Input: (int) a, b, p (Punto) P
Output: (int) Orden

(1)   Orden = 1;
(2)   SumPs = P;
(3)   while (sum(SumPs != [Inf Inf])>0))
(4)       SumPs = sumaPuntosCurvaEliptica(SumPs,P,a,b,p);
(5)       Orden = Orden + 1;
(6)   end
```

Algoritmo 6. *Orden de un punto en curva elíptica sobre Zp*

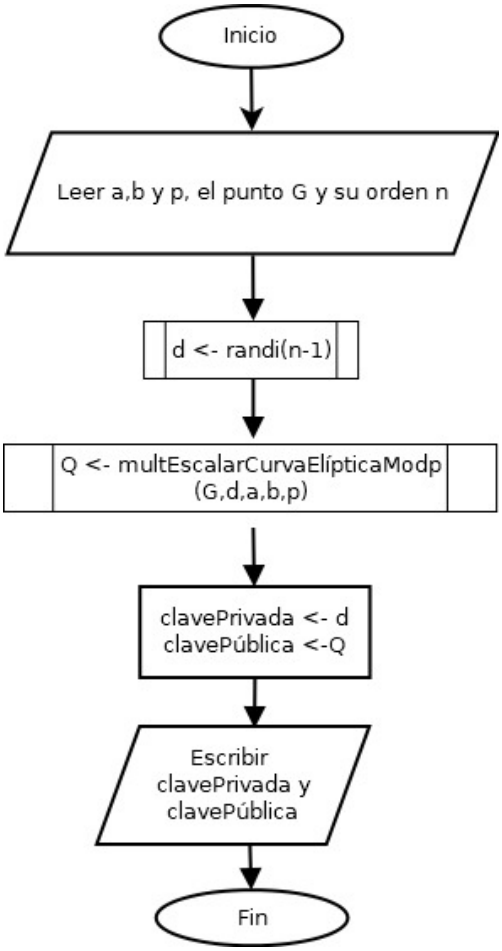
3.3.8 Generación de claves

La generación de un par de claves es parte de la **Criptografía Asimétrica (Clave Pública y Privada)** pero aplicado a las curvas elípticas, el funcionamiento e importancia se explica en el apartado **Generación de Claves**. El método de crear un par de claves, una pública a partir de una privada es sencillo ya que se dispone de las funciones necesarias para generarlas, es decir se puede generar las claves utilizando los algoritmos anteriores.

La **Figura 29** muestra cómo se generan las claves utilizando la multiplicación escalar de una curva elíptica, para ello se implementa el **Algoritmo 5. Multiplicación de un escalar por un punto en curva elíptica** *sobre Zp*; el punto por multiplicar será el **Punto Generador** y el escalar será un numero aleatorio.

Figura 29

Diagrama de flujo algoritmo de generación de claves, parte del esquema ECDH.



A continuación, se muestra la implementación del **Algoritmo 7** llamado "**generadorClavesEcc(…)**", el cual permite generar las claves pública y privada.

1. Se genera un entero aleatorio d entre 1 y $n-1$ utilizando la función $random$.
2. Utilizando la función **multEscalarCurvaEliptica(G,d,a,b,p)** descrita en el **Algoritmo 5** se calcula el punto Q en la curva elíptica dado por $Q = d * G$, donde G es un punto conocido en la curva elíptica y a y b son los parámetros de la curva.
3. Se asigna d como la clave privada y Q como la clave pública.
4. Finalmente, la función devuelve las claves generadas en los pasos anteriores.

```
Input: (int) a, b, p, n (Punto) G
Output: (int) clavePrivada (Punto) clavePublica

(1)   d = random(n-1);
(2)   Q = multEscalarCurvaEliptica(G, d, a, b, p);
(3)   clavePrivada = d;
(4)   clavePublica = Q;
(5)   end
```

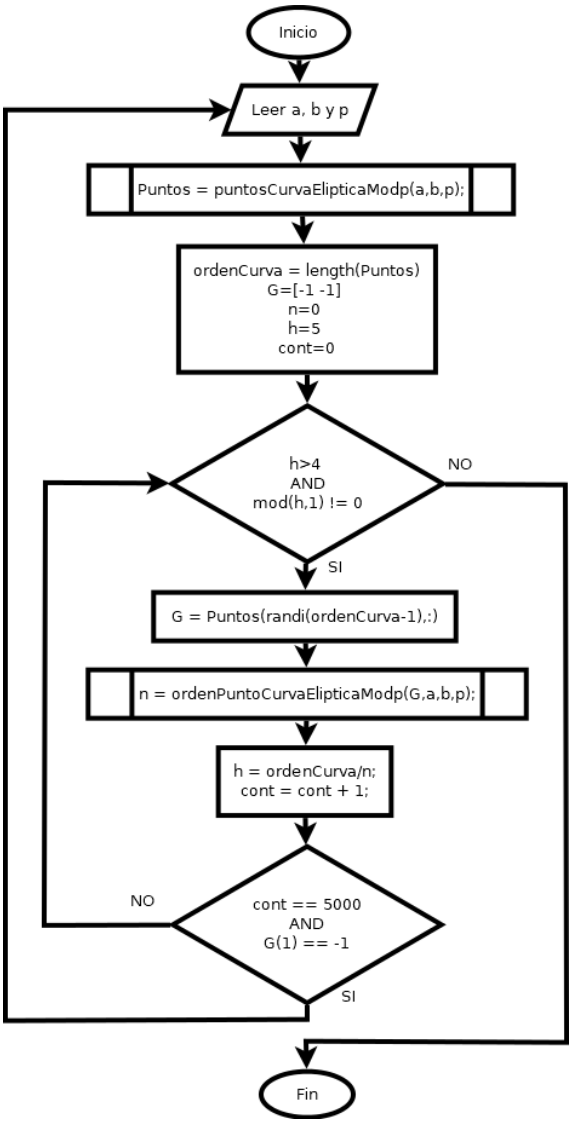
Algoritmo 7 *Generador de claves pública y privada*

3.3.9 Parámetros de dominio ECC

La elección adecuada de los **Parámetros de dominio de una curva elíptica** es fundamental para garantizar la seguridad y la eficiencia en la criptografía de curva elíptica. Los valores de a y b en la **(Ec. 2)** no siempre generan curvas adecuadas a pesar de estar en un cuerpo finito primo Zp, por ello se tiene como objetivo calcular los parámetros de una curva elíptica y un punto base para ser utilizados en todo el proceso del sistema. En la **Figura 30** se muestra el diagrama de funcionamiento el cual devuelve los parámetros necesarios como son: el **Orden de la curva y de un punto**, el **Punto Generador** ideal y el valor de **Cofactor** no mayor que 4 conforme lo explicado en **Orden de la curva y de un punto**.

Figura 30

Diagrama de flujo para la generación de los parámetros de la curva



A continuación, se muestra la implementación del **Algoritmo 8** llamado "**parametrosEcc(…)**" el cual utiliza el ***Algoritmo 2*** para la generación de puntos de la curva, la *generación aleatoria de puntos* y el **Algoritmo 6** para calcular el orden del punto; la comprobación de estas condiciones garantiza que los parámetros calculados sean seguros para su uso en el cifrado de datos.

1. Se ingresa los parámetros de entrada a, b y p, que corresponden a los coeficientes de la curva elíptica **(Ec. 2)** y el valor del primo $\boldsymbol{p}$.
2. Se utiliza un llamado al algoritmo **puntosCurvaEliptica** para calcular todos los puntos de la curva elíptica módulo p. A continuación, se calcula el *orden de la curva* y se inicializan las variables G, n y h.
3. Se ejecuta un bucle *while* que se repetirá hasta que se cumpla la condición determinada por el **Cofactor** con la variable h .
4. Dentro del bucle, se elige un punto aleatorio de la curva elíptica utilizando la función $random$, que elige un índice aleatorio entre 1 y el *orden de la curva - 1*.
5. A continuación, se calcula el orden del punto n elegido utilizando la función **ordenPuntoCurvaEliptica**.
6. El valor de h se calcula dividiendo el *orden de la curva* entre el orden del punto elegido n. El bucle continúa hasta que h sea menor que 4 y este sea un número entero. Si estas dos condiciones no se cumplen después de 5000 intentos, se regresa al paso 1.

```
Input: (int) a, b, p
Output: (Punto) G (int) ordenCurva, n, h

(1)   Puntos = puntosCurvaEliptica(a,b,p);
(2)   ordenCurva = length(Puntos);
(3)   G=[-1 -1];
(4)   n=0;
(5)   h=5;
(6)   cont=0
(7)   while ( h>4 || mod(h,1)!=0)
(8)       G = Puntos(random(ordenCurva-1),:);
(9)       n = ordenPuntoCurvaEliptica(G,a,b,p);
(10)      h = ordenCurva/n;
(11)      cont = cont + 1;
(12)      if (cont == 5000 || G(1) == -1)
(13)          disp('Se supero el límite de Intentos')
(14)          Return Input
(15)      end
(16)  end
```

Algoritmo 8 *Parámetros para ser utilizados en ECC*

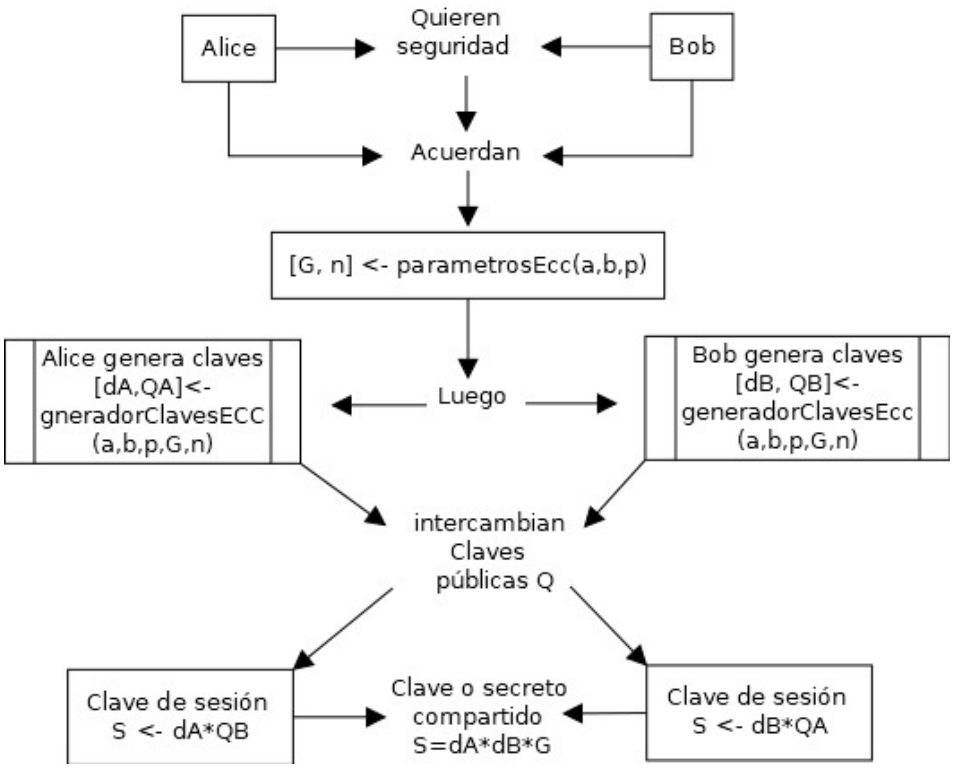
3.3.10 Intercambio de claves ECDH

El funcionamiento de ECDH se explica en los apartados **2.6** y **3.3.1.4** correspondientes a "**Algoritmo ECDH**" y "**Elliptic-Curve Diffie-Hellman (ECDH)**" respectivamente, hay que tener en cuenta que este algoritmo *no es un algoritmo de cifrado* y las claves que se intercambian son un paso intermedio y necesario para obtener una *clave secreta compartida*; Es decir en lugar de compartir la clave secreta directamente, las partes intercambian las claves públicas generadas a partir del algoritmo y utilizan estas claves para generar una clave secreta compartida que solo conocen las dos partes.

En la **Figura 31** se muestra el diagrama de flujo con los algoritmos que intervienen en el proceso ECDH, así como un ejemplo que sigue la secuencia del intercambio de claves para establecer una clave segura.

Figura 31

Diagrama de flujo del esquema ECDH.



A continuación, se muestra la implementación del **Algoritmo 9** llamado "**diffieHelmanEcc**":

1. Se ingresa los valores de a, b y p, con estos valores se llama a la función **parametrosEcc** para calcular el punto generador G y n, el orden del punto.

2. Ahora con los valores de a, b, p, G y n se usa el **Algoritmo 7** llamado **generadorClavesEcc** para obtener una clave pública y privada para Alice y para Bob.
3. Con las claves públicas y privadas de Alice y Bob, se calcula por separado la clave secreta compartida que viene representada por el punto S.
4. El intercambio de claves se usa para poder aplicar un criptosistema, utilizando una variante de **ElGamal Eliptico** donde se integran todas las funciones referentes a las operaciones con curva elíptica, la generación e intercambio de claves.

```
Input: (int) a, b, p
Output: (Punto) SA, SB

(1)   [ord, G, n, h] = parametrosEcc(a, b, p);
(2)   [dA, QA] = generadorClavesEcc(a, b, p, G, n);
(3)   print ('La clave privada de A es: dA);
(4)   print ('La clave publica de A es: QA(1),QA(2));
(5)   [dB, QB] = generadorClavesEcc(a, b, p, G, n);
(6)   print ('La clave privada de B es: dB);
(7)   print ('La clave publica de B es: QB(1),QB(2));
(8)   SA = multEscalarCurvaEliptica(QB,dA,a,b,p);
(9)   print ('La clave compartida A es: SA(1),SA(2));
(10)  SB = multEscalarCurvaEliptica(QA,dB,a,b,p);
(11)  print ('La clave compartida B es: SB(1),SB(2));
```

Algoritmo 9 *Calculo de clave secreta compartida S utilizando ECDH*

3.3.11 Representación de texto en Puntos de una curva elíptica

En 1987 Neal Koblitz propone un algoritmo para representar un **carácter imprimible**[18] en una curva elíptica, todo ello descrito en su artículo "*Elliptic curve cryptosystems*". En ese artículo, Koblitz se basa en el hecho de que para ciertas curvas elípticas no singulares $\boldsymbol{E}$: $\boldsymbol{y^2 = x^3 + ax + b}$ $(\boldsymbol{mod\ p})$, aproximadamente la mitad de los enteros módulo $\boldsymbol{p}$ tendrán raíces cuadradas, por lo que para representar un carácter en su **número decimal** $\boldsymbol{ASCII}$[19] en este caso representado como $\boldsymbol{m}$, se debe representar como una coordenada $\boldsymbol{x}$ de un punto en $\boldsymbol{E}$. Sin embargo, no todos los números enteros pueden ser representados como coordenadas $\boldsymbol{x}$ de un punto en la curva elíptica. En realidad, solo hay una probabilidad del $\boldsymbol{50\%}$ de que sea posible

[18] Los caracteres imprimibles son aquellos que se pueden representar como una cadena de bits o bytes y que se pueden mostrar o imprimir en una pantalla o en papel. Esto incluye letras, números, signos de puntuación y otros símbolos que se pueden representar mediante códigos de caracteres como ASCII.
[19] ASCII (American Standard Code for Information Interchange) es un código de caracteres estándar que se utiliza para representar texto y símbolos en la mayoría de los ordenadores y dispositivos electrónicos.

representar un carácter de esta manera. Esta probabilidad depende de que la ecuación $\boldsymbol{m^3 + am + b}$ tenga una raíz cuadrada módulo $\boldsymbol{p}$. Es decir, el valor de $\boldsymbol{m^3 + am + b \equiv {'Char'}(mod\ p)}$, tiene que ser congruente con el valor de carácter a representar en ASCII**.**

Por establecer una probabilidad no permitida para el cifrado, el algoritmo de Koblitz simplifica la fórmula general $\boldsymbol{m^3 + am + b \equiv {'Char'}(mod\ p)}$ utilizada para la codificación de coordenadas en la **(Ec. 17)**, con el mismo propósito y obteniendo una probabilidad dependiente de una variable, en este caso $\boldsymbol{K}$.

$$x = m * K + i \qquad \text{(Ec. 17)}$$

Donde, el entero positivo $\boldsymbol{K}$ es un parámetro de tolerancia al error, ya que representa el número de intentos en el algoritmo para encontrar la coordenada $\boldsymbol{x}$; Como la probabilidad de que cualquier entero $\boldsymbol{x}$ falle en cumplir la ecuación $\boldsymbol{y^2 = x^3 + ax + b\ (mod\ p)}$ es de $\frac{1}{2}$, la probabilidad de que cualquier entero $\boldsymbol{x}$ en $\boldsymbol{K}$ intentos falle se estima con la **(Ec. 18)**.

$$\left(\frac{1}{2}\right)^K \qquad \text{(Ec. 18)}$$

Además, en la **(Ec. 17)**, $\boldsymbol{m}$ es el valor del carácter en ASCII e $\boldsymbol{i}$, es un numero entero que va desde $\mathbf{0}$ hasta $\boldsymbol{K - 1}$. Para revisar la lista de enteros:

$$x0 = m * K + 0$$

$$x1 = m * K + 1$$

...

$$xK - 1 = m * K + K - 1$$

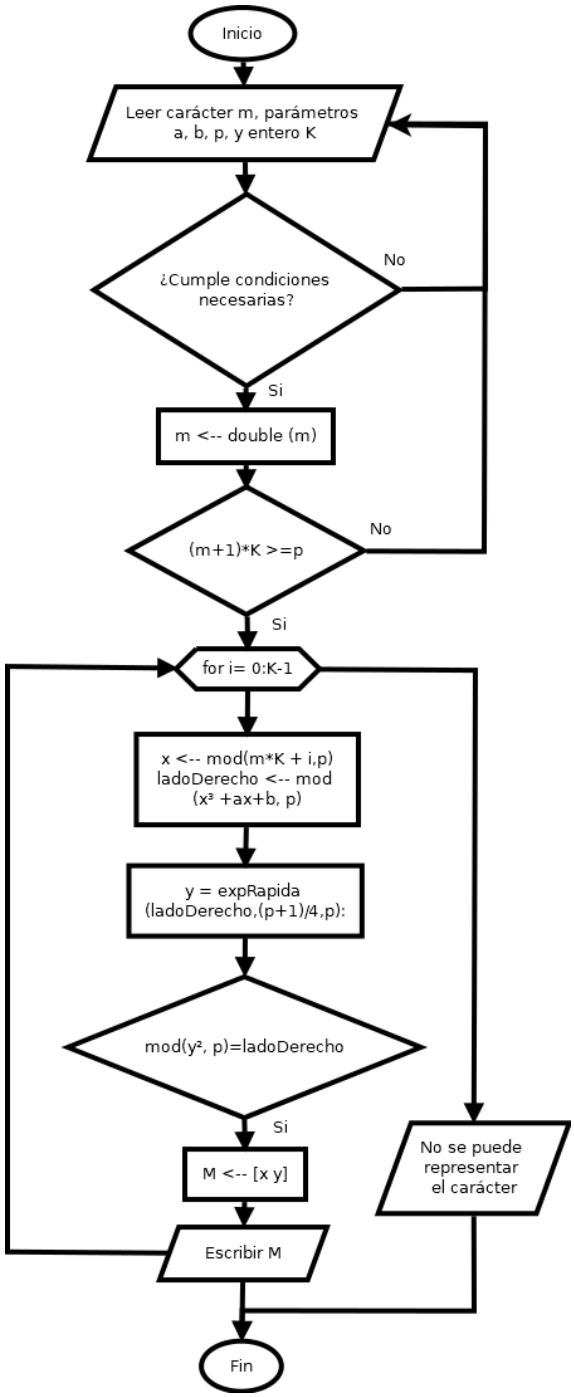
*S*i algún valor cumple como una coordenada $\boldsymbol{x}$ de un punto en $\boldsymbol{E}$. Finalmente devuelve el par ordenado $\boldsymbol{(x, \sqrt{x^3 + ax + b}\ )\ mod\ p}$ y cumpliendo con la **(Ec. 19)** la cual asegura que los valores de $\boldsymbol{x}$ generados, sean menores que el campo finito primo.

$$(m+1) * K < p \qquad \text{(Ec. 19)}$$

En la **Figura 32** se muestra el diagrama de flujo del algoritmo a implementar, el cual sirve para representar un carácter en una curva elíptica.

Figura 32

Diagrama de flujo para la representación de un carácter a puntos de una curva elíptica en el campo Zp.



A continuación, se muestra la implementación del **Algoritmo 10** llamado "**charToPuntoEcc(…)**"

1. Primero se verifica que los valores de a y b sean enteros y que p sea un número primo mayor que 3; condiciones necesarias para que la curva elíptica no sea singular.
2. Convierte el carácter en un valor numérico "código ASCII".
3. Verifica que el resultado de $(m + 1) * K$ sea menor a p.
4. Realiza K intentos para encontrar un punto (x, y) en la curva que represente el carácter. Para ello, realiza los siguientes cálculos en cada intento:
    a. Calcula los posibles valores de x con la fórmula: $x = (m * K + i)\ mod\ p$, donde i es un valor entre 0 y $K - 1$.
    b. Calcula el lado derecho de la ecuación de la curva elíptica $(x^3 + ax + b)\ mod\ p$.
    c. Calcula un posible valor de y con la fórmula: $y =$ **expRapida(ladoDerecho, (p + 1) / 4, p)**, explicada en el **Algoritmo 11**
    d. Verifica si (x, y) pertenece a la curva, es decir, si $y^2\ mod\ p$ es igual a ladoDerecho.
5. Si se encuentra un punto que represente el carácter, se guarda y se devuelve en la variable M. En caso contrario, se muestra un mensaje de error indicando que no se pudo representar el carácter y se sugiere intentar con un número primo p más grande.

```
Input: (int) a, b, p, K (Char) m
Output: (Punto) M

(1)   if (mod(a,1)!=0 || mod(b,1)!=0)}
(2)       print('El valor de a y de b deben ser enteros')
(3)   elseif (primo(p)!=1 || p<4)
(4)       print('El valor de p debe ser primo mayor que 3')
(5)   elseif (4*a^3+27*b^2 == 0)
(6)       print('Es una curva singular')
(7)   end
(8)   m = double(m);
(9)   print(m);
(10)  if ((m+1)*K >= p)
(11)      print('Error: debe usar un primo p más grande')
(12)      return Input
(13)  end
(14)  for (i = 0:K-1)
(15)      x = mod(m*K+i, p);
(16)      ladoDerecho = mod(x^3 + a*x + b, p);
(17)      y = expRapida(ladoDerecho,(p+1)/4,p);
(18)      if (mod(y^2,p) == ladoDerecho)
(19)          M = [x y];
(20)          return M
(21)      end
```

```
(22)  end
(23)  print('No se puede representar el carácter')
(24)  print('Intente usar un primo p más grande')
```

Algoritmo 10 *Representar un carácter en un punto de una curva*

3.3.11.1 Función exponencial rápida

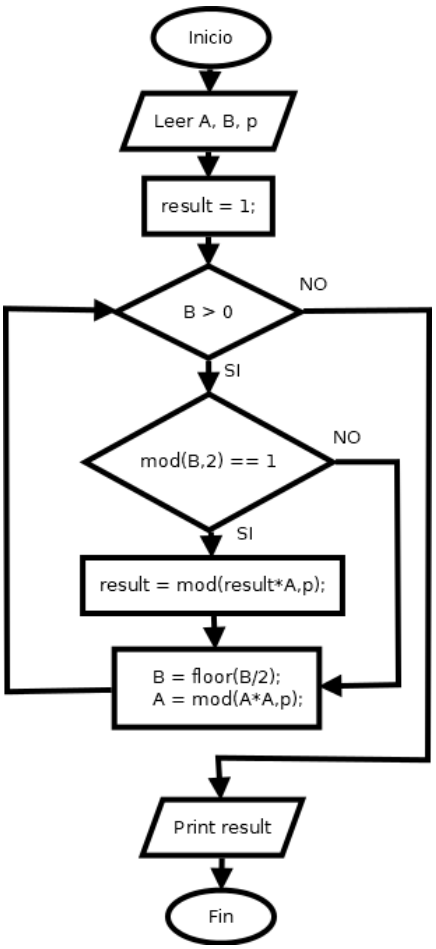
Este algoritmo proporciona una manera de calcular de forma rápida grandes potencias y se basa en la conversión binaria del exponente y en la operación $\boldsymbol{A^B}$ $(\boldsymbol{mod\ p})$.

1. En $\boldsymbol{A^B}$ $(\boldsymbol{mod\ p})$ se representa el exponente $\boldsymbol{B}$ en binario.
2. Se calculan los productos $\boldsymbol{A^{2^i}}$ en cada iteración desde $\boldsymbol{i = 0}$ hasta $\boldsymbol{n - 1}$, siendo $\boldsymbol{n}$ el número de bits que representan el valor $\boldsymbol{B}$ en binario.
3. Sólo se toman en cuenta los productos en los que en la posición $\boldsymbol{i}$ del valor $\boldsymbol{B}$ en binario aparece un $\boldsymbol{1}$.

En la **Figura 33** se observa el diagrama de flujo para la exponenciación modular rápida, nótese como el funcionamiento del algoritmo es igual que la *expansión de Horner*, solo que aplicado a la exponenciación.

Figura 33

Diagrama de flujo para la exponenciación modular rápida



A continuación, se explica la implementación y funcionamiento del algoritmo 11, donde:

1. Inicializa el resultado en 1.
2. Si B es mayor que 0, inicia un bucle para recorrer cada dígito binario de B.
3. Si el residuo de la división de B por 2 es 1, multiplica el resultado por la base A y lo reduce modulo p. Esto se hace para los dígitos binarios que son 1 en la representación de B.
4. Divide sucesivamente B entre 2 (elimina el dígito binario procesado).
5. Calcula los cuadrados sucesivos de A y reduce el resultado modulo p.
6. Se repite los pasos 3 a 5 hasta que B sea 0.
7. Devuelve el resultado.

```
Input: (int) A, B, p
Output: (int) result
(1)   result = 1;
(2)   while(B > 0)
(3)       if (mod(B,2) == 1)
(4)           result = mod(result*A,p);
(5)       end
(6)       B = roundDown(B/2);
(7)       A = mod(A*A,p);
(8)   end
```

Algoritmo 11 *Función exponencial rápida*

3.3.12 Representación Puntos de una curva elíptica a texto

Ahora para el descifrado se necesita una función que realice el proceso inverso al presentado previamente, es decir, que convierta un punto de la curva elíptica a su correspondiente carácter en texto claro. A continuación se muestra la implementación del **Algoritmo 12:**

1. Primero se verifica las condiciones necesarias: Comprueba que los valores de a y b sean enteros, que p sea un número primo mayor que 3 y que la curva elíptica no sea singular.
2. El algoritmo calcula los posibles valores de m en K intentos, usando el punto P y el valor de K.
3. Se calcula el carácter m a partir del punto P, usando el algoritmo: $m = mod((P(1) - i)/K, p)$.
4. El algoritmo verifica si Px es igual al x calculado, comparando $P(1)$ con x.
5. Si se puede representar el carácter, se guarda el carácter "m" y se devuelve el resultado. Si no es posible representar el carácter, se muestra el mensaje de error indicado en las líneas 18 y 19 del algoritmo 12.

```
Input: (int) a, b, p, K (Punto) P
Output: (char) m
(1)   if (mod(a,1)!=0 || mod(b,1)!=0)
(2)       print('El valor de a y de b deben ser enteros')
(3)   elseif (primo(p)!=1 || p<4)
(4)       print('p debe ser un numero primo mayor que 3')
(5)   elseif (4*a^3+27*b^2 == 0)
(6)       print('Es una curva singular')
(7)   elseif (mod(P(2)^2,p) != mod(P(1)^3 + a*P(1) + b, p))
(8)       print('P debe pertenecer a la curva elíptica')
```

```
(9)   end
(10)  for (i = 0:K-1)
(11)      m = mod((P(1)-i)/K,p);
(12)      x = mod(m*K + i,p);
(13)      if P(1) == x
(14)          m = char(m);
(15)          return m
(16)      end
(17)  end
(18)  print('No se puede representar el carácter')
(19)  print('Intente usar un primo p más grande')
```

Algoritmo 12 *Representar punto de una curva a un carácter*

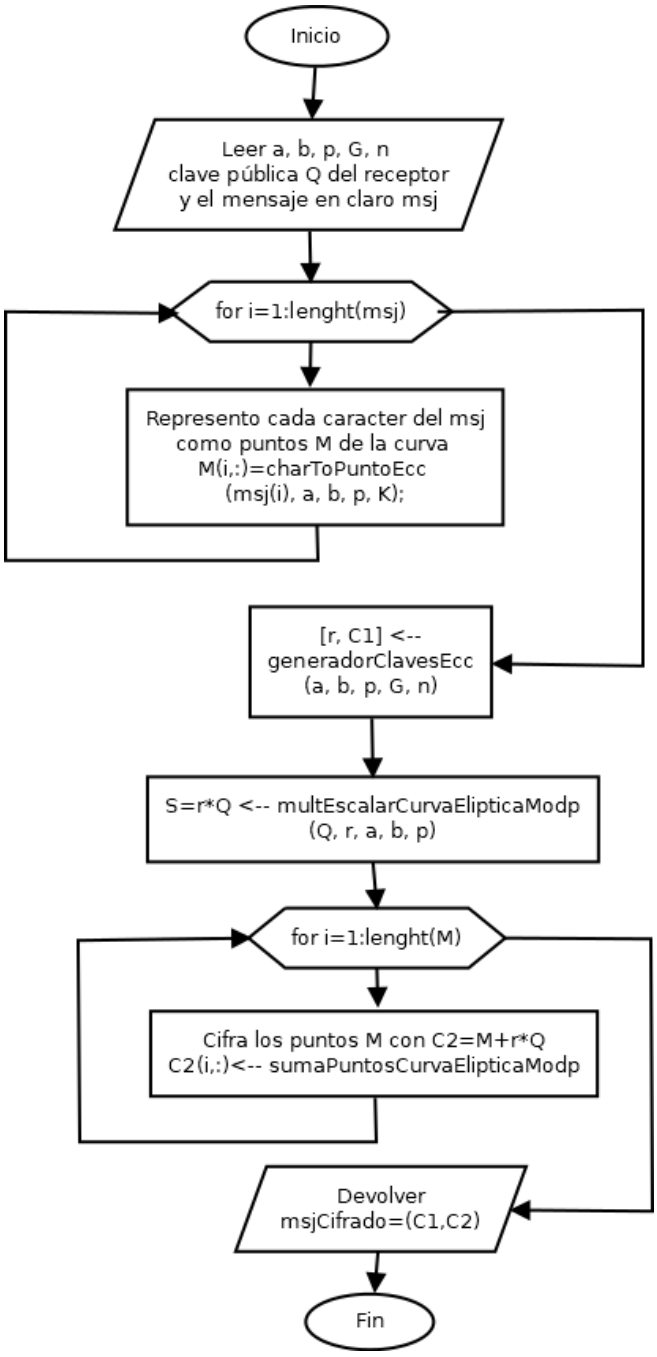
3.3.13 Algoritmo de cifrado ElGamal elíptico

Como se mencionó en el **Diagrama de Bloques del sistema,** el esquema de cifrado ElGamal basado en curvas elípticas fue propuesto por Koblitz y utiliza como grupo cíclico los puntos de una curva elíptica definida sobre Zp. El diagrama de flujo del algoritmo de cifrado ElGamal elíptico se muestra en la **Figura 34** y la lógica de este algoritmo se explica en los pasos siguientes:

1. Como entrada se tiene los siguientes parámetros (a, b, p, G, n), la clave publica Q y el mensaje en claro msj.
2. Representar el mensaje a cifrar msj como puntos M de la curva.
3. Escoger un entero aleatorio r en el intervalo de $[1, n-1]$, se le conoce como clave privada efímera o de sesión.
4. Calcular el punto $C1$ con la Ecuación, donde G es el punto generador. Se le conoce como clave pública efímera o de sesión.
5. Cifrar los puntos de M en $C2$ que son los puntos cifrados del mensaje. $C2=M+(r*Q)$
6. Devolver los valores de ($C1$,$C2$)

Figura 34

Diagrama de flujo del cifrado de datos utilizando ElGamal Eliptico.



A continuación, se muestra la implementación del algoritmo de cifrado ElGamal elíptico:

1. Se inicializa un vector vacío llamado ***M***, que se utilizará para guardar los puntos de la curva que representan cada carácter del mensaje.

2. Se recorre cada carácter del mensaje, utilizando un ciclo for, y para cada carácter, utiliza la función ***charToPuntoEcc*** para convertir el carácter en un punto de la curva. El resultado se guarda en el vector ***M***.
3. Obtiene el tamaño del vector ***M*** y lo guarda en una variable ***tam***.
4. Recorre cada punto del vector ***M*** utilizando otro ciclo for, y para cada punto, utiliza la función ***sumaPuntosCurvaElipticaModp*** descrita en el ***Algoritmo 3*** para sumar el punto con el punto de la clave compartida, que se recibe como parámetro. El resultado se guarda en un vector llamado ***C2***.
5. Asigna el vector ***C2*** a una variable llamada ***MsjCifrado*** y la función devuelve este valor.

```
Input: (int) a, b, p, K (Punto) S (String) msj
Output: (Matriz) Msjcifrado

(1)   M = [];
(2)   for i=1:length(msj)
(3)       M(i,:) = charToPuntoEcc(msj(i), a, b, p, K);
(4)   end
(5)   tam = size(M);
(6)   for i=1:tam(1)
(7)       C2(i,:)=sumaPuntosCurva…(M(i,:),S,a,b,p);
(8)   end
(9)   MsjCifrado = C2;
(10)  return
```

Algoritmo 13 *Cifrado ElGamal elíptico*

3.3.14 Algoritmo de descifrado ElGamal elíptico

El diagrama de flujo del algoritmo de descifrado ElGamal elíptico se muestra en la **Figura 35**, en este sentido, la lógica de funcionamiento de este se describe a continuación:

1. Como entrada se tienen los siguientes parámetros (a,b,p,G,n), la clave privada d, la clave pública de sesión y los puntos de cifrado ($C1$,$C2$).
2. Se descifran los puntos de la variable $C2$ para obtener los puntos M que representan el mensaje original con $M = C2 - dC1$

   Se reemplaza los valores de $C2$ y $C1$ para obtener el valor de M de la siguiente manera:

$$C2 - dC_1 = (M + rQ) - d(rG)$$
$$= M + rQ - rdG$$

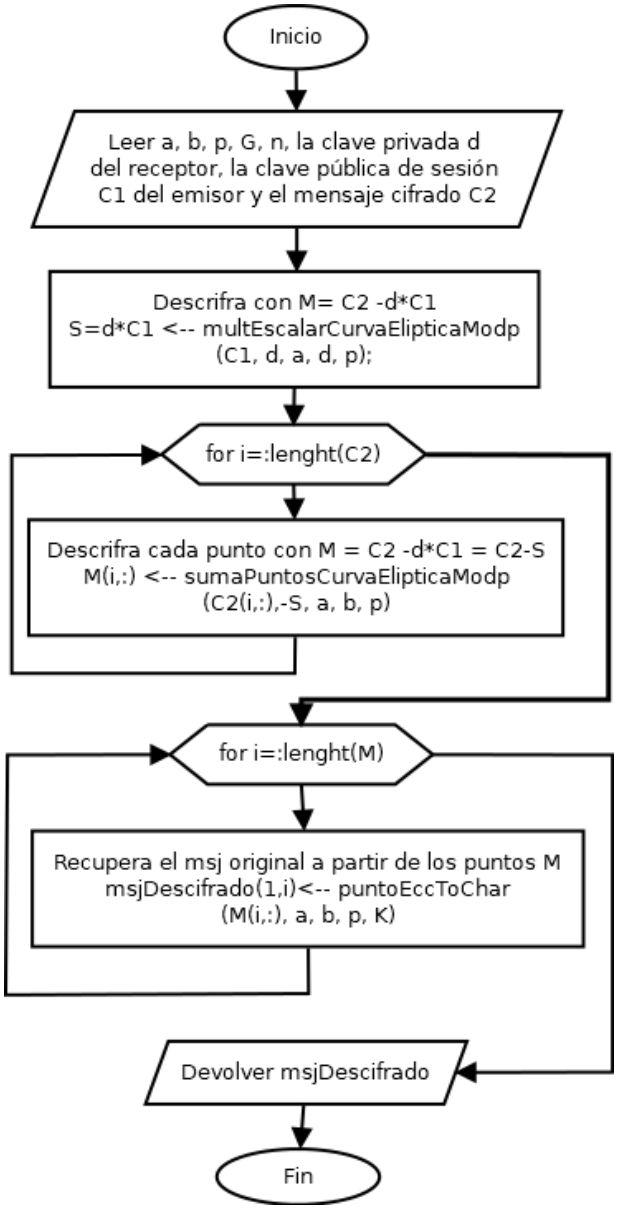
$$= M + rQ - rQ$$
$$= M$$

Esta operación se realiza con cada uno de los puntos de $C2$.

3. A partir de los puntos calculados M se recupera el mensaje msj original con el algoritmo de Koblitz inverso.

Figura 35

Diagrama de flujo del descifrado de datos utilizando ElGamal Elíptico.



A continuación, en el **Algoritmo 14** se presenta el seudocódigo sobre el descifrado ElGamal elíptico, cuya implementación se detalla en los pasos siguientes:

1. La función recibe como argumentos: a, b, p, K, S y $MsjCifrado$. "a" y "b" son los coeficientes de la curva elíptica, "p" es el número primo que se usa para definir el cuerpo finito en el que está definida la curva elíptica, "K" es el número de intentos, "S" es la clave compartida y "$MsjCifrado$" es el mensaje cifrado.
2. Se calcula $-S = (x, -y)\ (mod\ p)$ y se guarda en la variable "mS".
3. Se utiliza la función "***sumaPuntosCurvaElipticaModp***" descrita en el ***Algoritmo 3*** para calcular la suma de cada punto del vector "$C2$" con "mS". El resultado se guarda en la variable "M".
4. Se inicializa la variable "$msjDescifrado$" como un string vacío.
5. Se utiliza la función "***puntoEccToChar***" para convertir cada punto del vector "M" en un carácter. Los caracteres obtenidos se concatenan en la variable "$msjDescifrado$".
6. Finalmente, la función devuelve "$msjDescifrado$" como el mensaje descifrado.

```
Input: (int) a, b, p, K(Punto) S (Matriz) Msjcifrado
Output: (String) msj

(1)   C2 = MsjCifrado;
(2)   mS = [S(1) mod(-S(2),p)];
(3)   M = [];
(4)   tam = size(C2);
(5)   for (i=1:tam(1))
(6)       M(i,:)=sumaPuntosCurvaEliptica(C2(i,:),mS,a,b,p);
(7)   end
(8)   msjDescifrado='';
(9)   tam = size(M);
(10)  for (i=1:tam(1))
(11)      msjDescifrado(1,i)=puntoEccToChar(M(i,:),a,b,p,K);
(12)  end
(13)  return msjDescifrado
```

Algoritmo 14 *Descifrado ElGamal elíptico*

3.3.15 Elección de Software para la Simulación

La simulación es una herramienta valiosa que permite a los usuarios predecir el comportamiento y las consecuencias de sus acciones antes de tomar decisiones o realizar cambios en un sistema o proceso. Utilizando un software de simulación se puede:

- **Ahorrar tiempo y recursos**, la simulación permite probar diferentes escenarios y soluciones sin tener que implementarlos en el mundo real.
- **Aumentar la eficiencia** ya que esta identifica cuellos de botella y otras áreas de mejora en un sistema o proceso,
- **Mejorar la toma de decisiones** visualizando los resultados de la simulación y comparando diferentes opciones
- **Mitigar riesgos** y las consecuencias potenciales de sus acciones antes de implementarlas en el mundo real.

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

A continuación, se presenta una descripción general de los diversos simuladores actualmente disponibles para WSN. En general, la elección de un software depende de las necesidades específicas de cada proyecto y de las preferencias personales del usuario en cuanto a características, facilidad de uso y recursos disponibles, a continuación, se considera cuatro programas (Omnet++, Matlab, NS2 y NS3) basados en varias características como son: su curva de aprendizaje, versiones, lenguaje de programación que utiliza, etc. en la **Tabla 8** se muestra un resumen de cada software.

OMNeT++: es un entorno de simulación de sistemas discretos para la investigación y el desarrollo de sistemas complejos, especialmente en áreas de redes de computadoras, telecomunicaciones, y sistemas distribuidos. OMNeT++ es un software de código abierto, escrito en C++ y disponible en varios sistemas operativos.

MATLAB: es un software de cálculo técnico y visualización que permite resolver problemas complejos y analizar datos. también es capaz de simular sistemas dinámicos, incluyendo redes

de sensores, y ofrece una amplia biblioteca de algoritmos y funciones para realizar tareas de simulación.

NS2 (Network Simulator 2): es un software de simulación de redes de computadoras que permite modelar, simular y evaluar el comportamiento de redes de computadoras y protocolos. NS2 está escrito en C++ y es un software de código abierto disponible en varios sistemas operativos.

NS3 (Network Simulator 3): es una evolución de NS2, que tiene como objetivo mejorar la escalabilidad, la facilidad de uso y la flexibilidad de NS2. NS3 es un software de código abierto escrito en C++ y disponible en varios sistemas operativos.

Tabla 8. *Parámetros relevantes para cada software de simulación*

Plataforma de Simulación	***OMNET++***	***NS2***	***NS3***	***MATLAB***
Lenguaje	Inglés	Inglés	Inglés	Inglés, Español, etc.
Lenguaje de programación base	C++	C++ y TCL	C++	Lenguaje *m* (*propio del software*), es un lenguaje de alto nivel
Ultima versión	6.0.1	2.35	3.37	R2022b
Tipos de redes que puede simular	3G, 4G, 5G, LTE, Wi-Fi, redes de computadoras, redes de sensores WSN, redes de vehículos, redes de área amplia, redes de área local, redes móviles, redes satelitales, entre otras.	Redes de computadoras, redes de sensores WSN, redes de vehículos ad hoc, redes de área amplia, redes de área local, redes móviles, redes de comunicación satélite, redes de tiempo real, entre otras.	Redes de computadoras, redes de sensores WSN, redes de vehículos ad hoc, redes de área amplia, redes de área local, redes móviles, redes de comunicación satélite, entre otras.	5G, Wi-Fi, LTE, redes de comunicación satelital, Bluetooth, redes de sensores WSN, además de Evaluar el rendimiento y las métricas de una red inalámbrica.
Compatibilidad con otros lenguajes de programación	Python	NAM (OTCL)	Python	Las API del motor, C/C++, Fortran, Java, Python

Sistema Operativo	Windows, Linux, MacOS, UNIX	FreeBSD, Linux, SunOS, Solaris, Windows y MacOS X	Windows, Linux, MacOS, Free BSD	Windows, Linux, MacOS
Curva de aprendizaje	Alta	Alta	Alta	Media
Disponibilidad de documentación	Buena	Buena	Buena	Muy Buena
Soporte	Limitado	Limitado	Limitado	Bueno
Tipo de licencia	GPL - Open-source	BSD - Open-source	BSD - Open-source	MathWorks

Nota: La licencia *GPL* es esencialmente una garantía de propiedad intelectual, mientras que el tipo de licencia *BSD* está destinada a fomentar la comercialización del producto, finalmente para el software MATLAB es un producto de la empresa MathWorks y esta cuenta con todos los derechos sobre la licencia. La información de cada software está tomada del sitio oficial, así como del autor (Cuzme-Rodríguez, 2020).

A continuación, se establece la **Tabla 9** donde se realiza la comparativa de los softwares y los requerimientos a tomar en cuenta están relacionados con el software de simulación, estos están explicados detalladamente en la **Fase 1: Análisis de Requisitos**.

Tabla 9. *Características del software para cada requerimiento*

Plataforma de Simulación	***Requerimientos***						***Valoración Total***
	SySR 4	SySR 8	SrSH 5	StSR 1	StSR 2	StSR 5	
Omnet ++ 6.0	*1*	*1*	*1*	*1*	*1*	*1*	*6*
MATLAB	*1*	*1*	*1*	*1*	*1*	*1*	*6*
NS2	*1*	*0*	*1*	*1*	*1*	*0*	*4*
NS3	*1*	*0*	*1*	*1*	*1*	*0*	*4*
1= cumple 0= no cumple							

En base a la información contenida de las **Tabla 8 y Tabla 9** se da como resultado que el software principal a utilizar es Omnet++ ya que permite representar una WSN con muchas características, haciendo que sea lo más parecido a un entorno real y hacer pruebas del funcionamiento en base al envío de datos. También se utilizará como complemento para realizar pruebas, generar imágenes y buscar parámetros de eficiencia el software MATLAB.

3.3.16 Definición de características de Hardware a simular

Las características de hardware son indispensables para conocer las limitaciones en términos de memoria, procesamiento y energía. Por lo tanto, la simulación de una red de sensores debe reflejar estas limitaciones y características del hardware para proporcionar una representación precisa y realista del comportamiento de la red. El conocimiento de las características del hardware también es importante para ajustar los parámetros de la simulación para garantizar que la simulación sea precisa y realista. Por ejemplo, la frecuencia de radio, la tasa de transmisión de datos y la sensibilidad de la radio son características críticas del hardware que deben tenerse en cuenta al configurar la simulación. Si estos parámetros no se ajustan correctamente, la simulación puede no reflejar con precisión el comportamiento de la red de sensores.

Además, la comprensión de las características del hardware puede ayudar a identificar posibles cuellos de botella en la red y guiar la selección de algoritmos de enrutamiento y técnicas de transmisión de datos adecuados. Si, por ejemplo, el hardware de los sensores tiene una capacidad de procesamiento limitada, puede ser necesario seleccionar un algoritmo de enrutamiento que minimice la sobrecarga de procesamiento.

Las opciones de hardware se escogen de acuerdo con el análisis de software ya que al tratarse de un entorno virtual solo se puede permitir establecer las características de hardware que el software nos permita en este caso **Omnet++**; Este software cuenta con la librería **INET** la cual a su vez permite la simulación de nodos sensores utilizando el **estándar 802.15.4** encontrado en el módulo **Ieee802154NarrowbandRadio** permitiendo así definir las características de los nodos sensores que se utilizarán en la creación de la red de sensores inalámbricos. Este módulo Ieee802154NarrowbandRadio contiene los parámetros para especificar la *frecuencia central, ancho de banda, sensibilidad del receptor, potencia de transmisión* y otros detalles relacionados con la *comunicación inalámbrica*. Donde se puede hacer referencia a especificaciones técnicas de microcontroladores, o placas de desarrollo enfocadas a la transmisión de datos.

A continuación, se presenta una descripción general de los diversos hardware para redes de sensores los cuales cumplen características para ser simulados en Omnet++. En general, la elección del hardware depende de las necesidades específicas de cada proyecto y no es una norma que se utilice estos sensores específicos.

Digi XBee es un módulo de comunicación inalámbrica producido por Digi International. Opera en el rango de frecuencia de 2.4 GHz y utiliza protocolos como ZigBee, WiFi y Bluetooth. El rango de cobertura varía según el modelo y la velocidad de transferencia es de hasta 250 kbps.

MRF24J40MA es un módulo transceptor inalámbrico producido por Microchip Technology Inc. Opera en el rango de frecuencia de 2.4 GHz y es compatible con la norma IEEE 802.15.4. Ofrece una velocidad de transferencia de hasta 250 kbps y un rango de cobertura de hasta 400 metros en línea de visión.

NRF24L01+PA+LNA es un módulo transceptor inalámbrico producido por Nordic Semiconductor. Opera en el rango de frecuencia de 2.4 GHz y es compatible con la norma IEEE 802.15.4. Ofrece una velocidad de transferencia de hasta 2 Mbps y un rango de cobertura de hasta 1 km en línea de visión.

Moteino es un módulo transceptor inalámbrico de bajo consumo de energía que utiliza el procesador ATmega256RFR2. Opera en el rango de frecuencia de 915 MHz y ofrece una velocidad de transferencia de hasta 250 kbps. Tiene un rango de cobertura de hasta 300 metros y utiliza protocolos como ZigBee y Mesh Networking.

En la **Tabla 10** se detallan los nodos con sus respectivos atributos importantes, estos datos han sido extraídos de las hojas de documentación proporcionadas por cada fabricante. Es importante tener en cuenta que algunos ***módulos de radio frecuencia*** están destinados a WSN y requieren de un microcontrolador para poder funcionar, asi como de sensores para la recolección de datos. Asimismo, existen ***nodos*** que combinan módulos de RF y microcontroladores en una sola placa de desarrollo, lo que permite funcionar únicamente con sensores para la recopilación y transmisión de datos. Finalmente los ***nodos sensores*** integran tanto los módulos RF, el microcontrolador y los sensores necesarios para su operación.

Tabla 10. *Parámetros relevantes para cada hardware dentro de simulación*

Hardware	***Digi XBee Pro***	***MRF24J40MA***	***NRF24L01+PA+LNA***	***Moteino***
Compañía	Digi International	Microchip Technology	Nordic Semiconductor	LowPowerLab
Frecuencia de operación	2.4 GHz	2.4 GHz	2.4 GHz	915 MHz
Velocidad de transferencia	hasta 1Mbps	hasta 2 Mbps	hasta 2 Mbps	hasta 250 kbps

Modo de operación	varios, incluyendo ZigBee, WiFi y Bluetooth	punto a punto y punto a multipunto	punto a punto y punto a multipunto	varios, incluyendo punto a punto y punto a multipunto
Rango de cobertura	5 km en línea de visión	100 metros en línea de visión	1 km en línea de visión	300 metros
Procesador	ARM Cortex-M3	N/A	N/A	ATmega256RFR2
Memoria RAM	128 KB	Depende del microcontrolador a usar	Depende del microcontrolador a usar	32 KB
Almacenamiento	2 MB	Depende del microcontrolador a usar	Depende del microcontrolador a usar	256 KB
Potencia	250 mW	0 dBm a 10 dBm	7 dBm	3.3 V, 30 mA (transmisión), 15 mA (recepción)
Protocolos	ZigBee, Thread, Wi-Fi	IEEE 802.15.4	IEEE 802.15.4	ZigBee, Mesh Networking

Nota: Los módulos MRF24J40MA y NRF24L01+PA+LNA no cuentan con un procesador o microcontrolador integrado en el módulo. Por lo tanto, la cantidad de memoria RAM y el almacenamiento disponible dependerá del microcontrolador que se use en conjunto con el módulo, es compatible con una amplia variedad de microcontroladores y plataformas, como Arduino, Raspberry Pi, STM32, etc.

3.3.17 Escenarios de Pruebas

Es importante tener un mínimo de tres escenarios de pruebas donde las redes de sensores estarán enfocadas a la **monitorización ambiental en lugares cerrados**, **la agricultura y la monitorización de la salud**, conforme lo mencionado en el **Alcance** del presente trabajo de grado**.**

Al tener varios escenarios de prueba, se puede evaluar la capacidad de la red para proteger la privacidad y la integridad de los datos, prevenir el acceso no autorizado a la red y resistir diferentes tipos de ataques, como ataques de denegación de servicio, ataques de inundación, ataques de interceptación de mensajes, entre otros. Además, los escenarios de prueba también pueden ayudar a determinar los requisitos de energía, recursos de la red de sensores y a verificar el funcionamiento del algoritmo de cifrado/descifrado de datos con ECC, alguna de las verificaciones principales puede ser:

- **Validación del algoritmo**: Al probar el algoritmo de cifrado en diferentes escenarios, se puede verificar su validez y robustez. Es decir, se pueden probar diferentes configuraciones y entornos para asegurarse de que el algoritmo es eficaz y no se puede romper fácilmente.
- **Evaluación de rendimiento**: También se puede evaluar el rendimiento de la red de sensores inalámbricos con diferentes cargas de tráfico, diferentes tipos de nodos sensores, diferentes niveles de interferencia, etc.
- **Identificación de vulnerabilidades**: Al probar el algoritmo en diferentes escenarios, se pueden identificar posibles vulnerabilidades en la red de sensores inalámbricos. Si se encuentra una vulnerabilidad en un escenario, se pueden tomar medidas para corregirla antes de que se produzca un ataque real en la red.

Por lo tanto, se ha escogido los siguientes despliegues de WSN aplicadas a cada uno de los campos antes mencionados, a continuación, se detallan los siguientes escenarios:

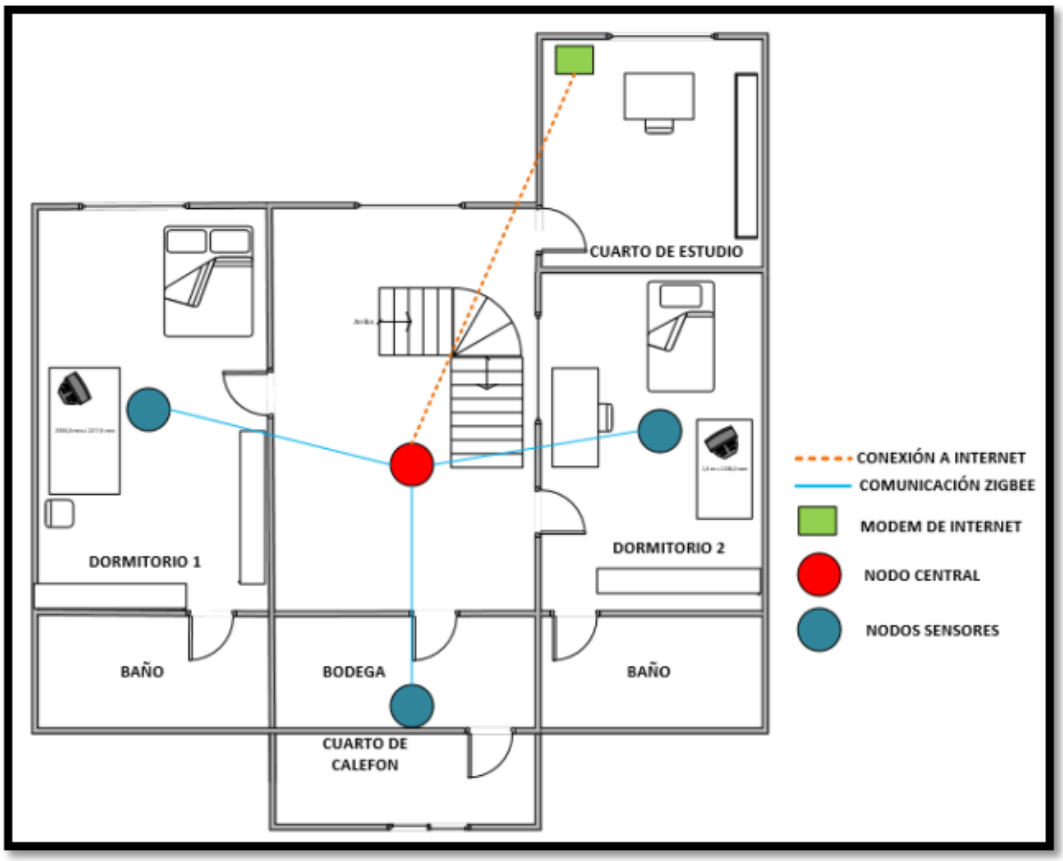
3.3.17.1 Monitorización ambiental en lugares cerrados

(Carrión, 2016) propone un sistema de monitoreo de monóxido de carbono mediante una red de sensores inalámbricos; Tratándose de un sistema con el objetivo de brindar seguridad ambiental en una residencia, que tiene como base una red de sensores inalámbricos con el fin

de cuidar las vidas humanas de la intoxicación por gases mortales. El sistema utiliza una red de sensores con tecnología Zigbee usando el estándar 802.15.4.

Figura 36

Diseño de WSN para la monitorización ambiental en lugares cerrados.



Nota: En la Figura se observa la topología de la red, usando un nodo sensor o "nodo central" como Gateway para los tres nodos que monitorean zonas importantes de un ambiente cerrado. Autor (Carrión, 2016).

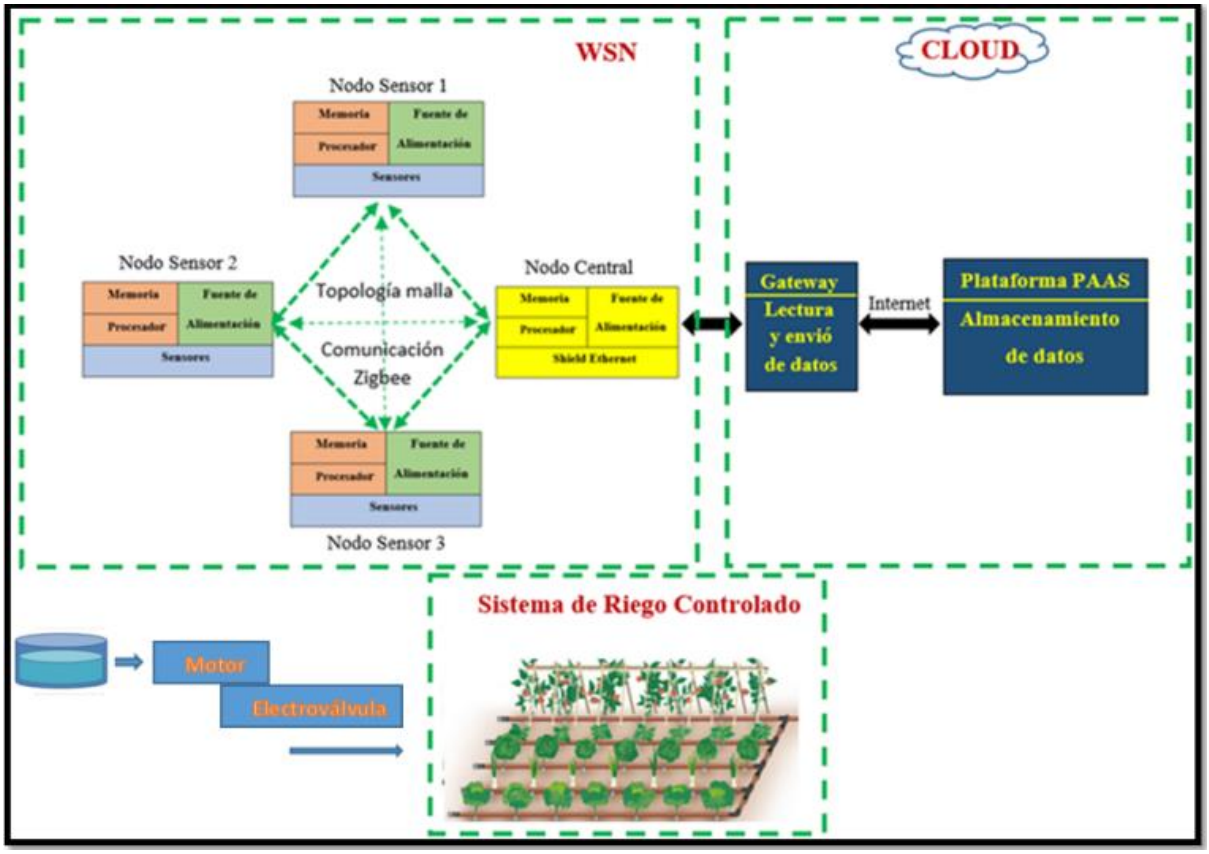
3.3.17.2 Monitorización en la agricultura

(Palacios, 2017) realizó un diseño de una red de sensores (WSN) con tecnología 802.15.4, basado en el concepto agricultura de precisión. La red WSN para la pradera de la Universidad Técnica del Norte, consiste en aplicar conceptos de agricultura de precisión para optimizar el control de un sistema de riego por goteo así como el monitoreo de agentes ambientales

involucrados en un cultivo de hortalizas bajo invernadero, así lograr un mejor aprovechamiento del agua de riego y los datos obtenidos de los parámetros estudiados.

Figura 37

Diseño de WSN para la monitorización en la agricultura.



Nota: En la Figura se observa la topología de la red, usando un "nodo central" como Gateway utilizado para el envío de datos, y tres nodos que monitorean parámetros importantes de las plantas, en base a eso se activa un sistema de riego. Autor (Palacios, 2017)

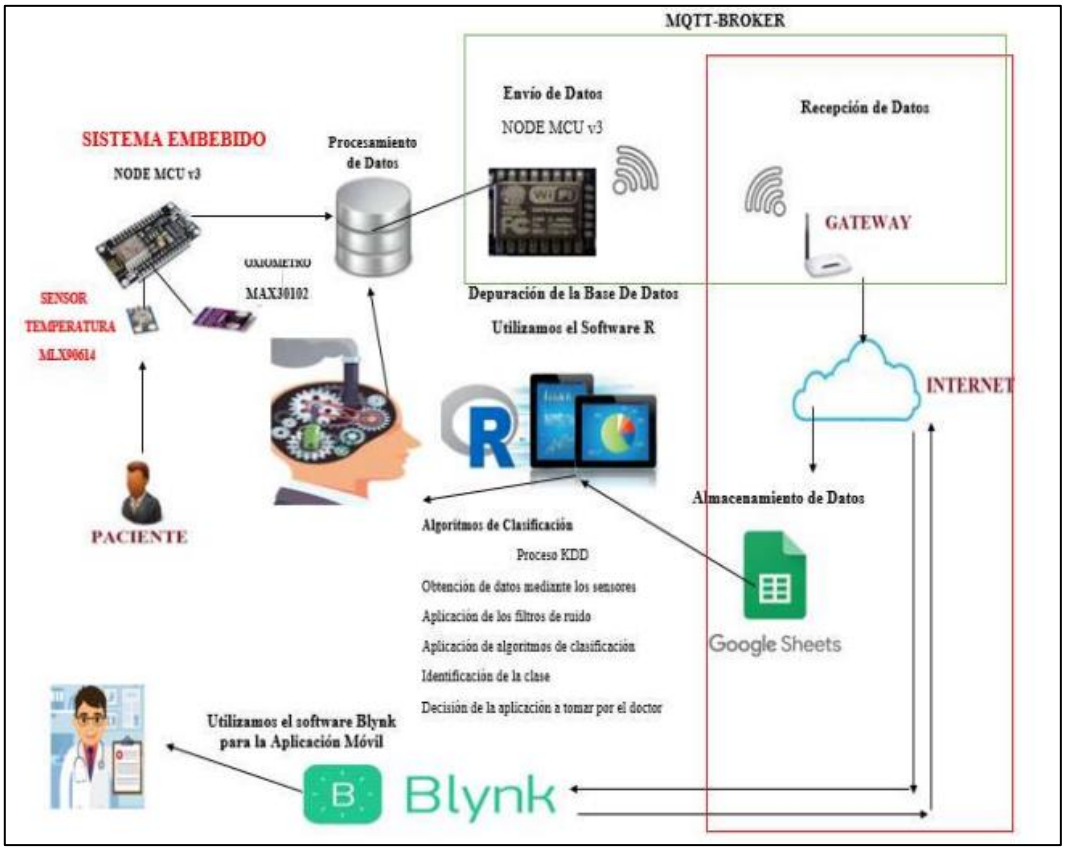
3.3.17.3 Monitorización de la salud

(Pineda, 2021) generó un medidor de temperatura y saturación de niveles de oxígeno en sangre para la monitorización de pacientes diagnosticados por Covid-19. Esta tesis se basa en el desarrollo de un sistema electrónico de detección temprana de anomalías en la salud de pacientes diagnosticados con COVID-19 denominado TEMPOXI-19. El sistema propuesto consiste en un detector de dos estados de salud de pacientes afectados por el SARS COV2 mediante la medición de la temperatura corporal y los niveles de saturación de oxígeno en

sangre desarrollado con técnicas aprendizaje supervisado y planteado con el uso de aplicaciones IoT.

Figura 38

Diseño de WSN para la monitorización de la salud.



Nota: En la Figura se observa la topología de la red, usando un el sistema embebido como nodo sensor enviando los datos directamente a un nodo MCU. Autor (Pineda, 2021)

3.4 Fase 3: Implementación

Para esta implementación se trabaja con el software Omnet++ el mismo que se puede descargar desde su página oficial https://omnetpp.org/

OMNET++ es una biblioteca y proporciona la infraestructura y herramientas para crear simuladores, utiliza un marco de simulación de C++ extensible, modular y basado en componentes, todas estas características son fundamentales para crear una infraestructura de red. La última versión a la fecha de redacción de este documento es la 6.0.1.

OMNET++ se caracteriza por tener una simulación de eventos discretos, es decir **el tiempo de simulación avanza según los eventos presentes** en la cola de eventos, Un evento sucede en momentos discretos de tiempo (un tiempo específico) y el estado del sistema puede cambiar durante estos eventos.

Las ventajas de trabajar con eventos en una cola de tiempos es que permite estudiar tiempos reales prolongados en tiempos de simulación reducidos, observar procesos que ocurren en tiempos muy pequeños (ms, us) y desarrollar protocolos y algoritmos definidos por eventos. Omnet++ utiliza tres tipos de archivos principales para su funcionamiento:

- **Archivos NED**: son utilizados para definir la estructura de la red. En ellos se definen los nodos de la red, sus conexiones y las interfaces de comunicación que utilizan.
- **Archivos INI**: se utilizan para especificar los parámetros de inicio de la simulación. En ellos se pueden definir diferentes opciones de configuración de la simulación, tales como el tiempo de simulación, el número de nodos en la red y los valores iniciales de los parámetros de los nodos.
- **Archivos de clases C++:** se utilizan para definir el comportamiento de los nodos de la red. Estos archivos contienen la implementación de las funciones que definen el comportamiento de los nodos, como el procesamiento de mensajes, la toma de decisiones y la generación de eventos.

  Los archivos de clases C++ constan de dos partes principales: los **archivos ".h"** que definen las clases y los métodos que se utilizarán en el **archivo ".cc"** que contiene la implementación de los métodos definidos en los archivos **".h"**. Estos archivos de clases C++ permiten a los usuarios definir el comportamiento personalizado de los nodos en

la red, lo que permite simular una amplia variedad de sistemas y protocolos de comunicación.

Otro punto importante son las variables que pertenecen a un módulo. Estos parámetros se pueden utilizar para construir la topología de la red, es decir, para definir el número de nodos, las conexiones entre ellos, etc. Además, se utilizan para proporcionar información al código C++ que implementa los módulos y canales simples; Existen varios tipos de parámetros que se pueden utilizar, entre ellos se encuentran:

- **double:** se utiliza para representar números de punto flotante.
- **int:** se utiliza para representar números enteros.
- **bool:** se utiliza para representar valores booleanos, es decir, verdadero o falso.
- **string:** se utiliza para representar cadenas de caracteres.
- **xml:** se utiliza para representar documentos XML.
- **csv:** se utiliza para representar archivos CSV.
- **object:** se utiliza para representar objetos complejos.

Los parámetros también pueden ser declarados volátiles, lo que significa que su valor puede cambiar en tiempo de ejecución. También se puede proporcionar un valor predeterminado utilizando la sintaxis **"default(...)"**, que se utilizará si el parámetro no se asigna de otra manera.

3.4.1 Desarrollo de la librería "Funciones ECC" para la implementación de ECC en Omnet++

Primero, es importante señalar que para utilizar las funciones de criptografía de curva elíptica en el software Omnet++, es necesario crear un archivo en lenguaje de programación C++ que contenga todas las funciones necesarias. Esto se logra mediante la conversión de las funciones diseñadas en la **Fase 2: Diseño del Sistema** a código C++, lo cual se describe en el **ANEXO B – Funciones ECC**. Para utilizar el archivo es necesario importar la librería usando *#include* "*funcionesECC.h*", Uno de los cambios a importantes que se realizó es definir el punto al infinito con el valor de nueves "999999" tanto para $\boldsymbol{x}$ como para $\boldsymbol{y}$; El punto al infinito puede ser cualquier valor que este fuera de la curva elíptica.

Es importante destacar que la librería creada a partir de estas funciones es esencial para la implementación de la criptografía de curva elíptica en diferentes aplicaciones, ya que permite la generación de claves, el intercambio seguro de información **Elliptic-Curve Diffie-Hellman (ECDH)**, el cifrado y descifrado de datos **ElGamal Elíptico**, la creación de firmas digitales, contratos inteligentes, entre otras aplicaciones.

Además, una ventaja importante de utilizar la criptografía de curva elíptica es que puede crear mecanismos de cifrado por bloques o por flujo, lo que aumenta la variedad de aplicaciones. También es posible generar tamaños de clave reducidos, lo que la hace ideal para mecanismos donde se utiliza la criptografía ligera.

Finalmente, la librería creada también incluye un modelo para la creación de hashes utilizando el algoritmo **sha-256**, lo que añade una capa adicional de seguridad a las aplicaciones que utilizan criptografía de curva elíptica.

3.4.2 Showcase de INET – "Coexistencia de IEEE 802.11 y 802.15.4"

Los *showcases*[20] de INET son pequeños estudios de simulación que muestran varios componentes y características de **INET Framework**[21]. Cada *showcase* consta de un modelo de simulación completamente configurado. Aunque no se diseñaron expresamente como tutoriales, los *showcases* se crearon con la esperanza de que fueran directamente útiles para los usuarios de INET que realizan simulaciones relacionadas.

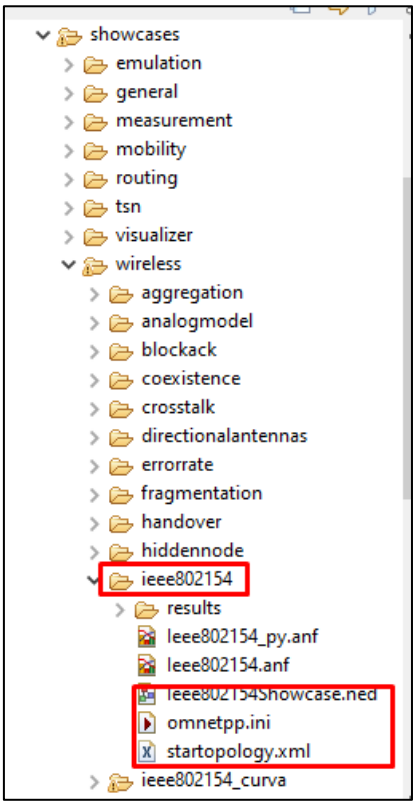
Se utiliza el ejemplo "*Coexistencia de IEEE 802.11 y 802.15.4*" disponible en https://inet.omnetpp.org/docs/showcases/wireless/coexistence/doc/index.html# y se lo modifica para trabajar solo con el estándar IEEE 802.15.4 y con $\boldsymbol{n}$ host. Este *showcase* viene incluido en la librería **inet4.4**. En la **Figura 39** se puede observar los archivos principales .***ini*** y .***ned*** necesarios para ejecutar el programa, estos archivos hacen el llamado de módulos y funciones para completar la lógica de transmisión inalámbrica, todos los ejemplos de *showcase* tiene su respectiva documentación disponible en **INET Framework**, se puede acceder a esta

[20] **showcase** es un término en inglés que se refiere a una exhibición o muestra de algo, en este caso la demostración de una simulación que demuestre cómo se comporta una red en 802.11 y 802.15.4.
[21] **INET Framework** es un conjunto de bibliotecas de modelos y herramientas de simulación de redes para OMNeT++.

documentación usando el enlace https://inet.omnetpp.org/docs/index.html, por otro lado si se desea acceder al ejemplo en OMNeT++ se utiliza la librería **inet4.4**. Estos *showcase* pueden ser modificables, agregándoles funcionalidades adicionales a los nodos o estableciendo algoritmos de seguridad para la red.

Figura 39

Ubicación del showcase coexistencia de 802.11 y 802.15.4 dentro de inet4.4



Nota: En la Figura se observa los archivos principales de este *showcase*; aun así, si existen más archivos los cuales contiene configuraciones como el medio de simulación, propiedades del nodo y propiedades de la red.

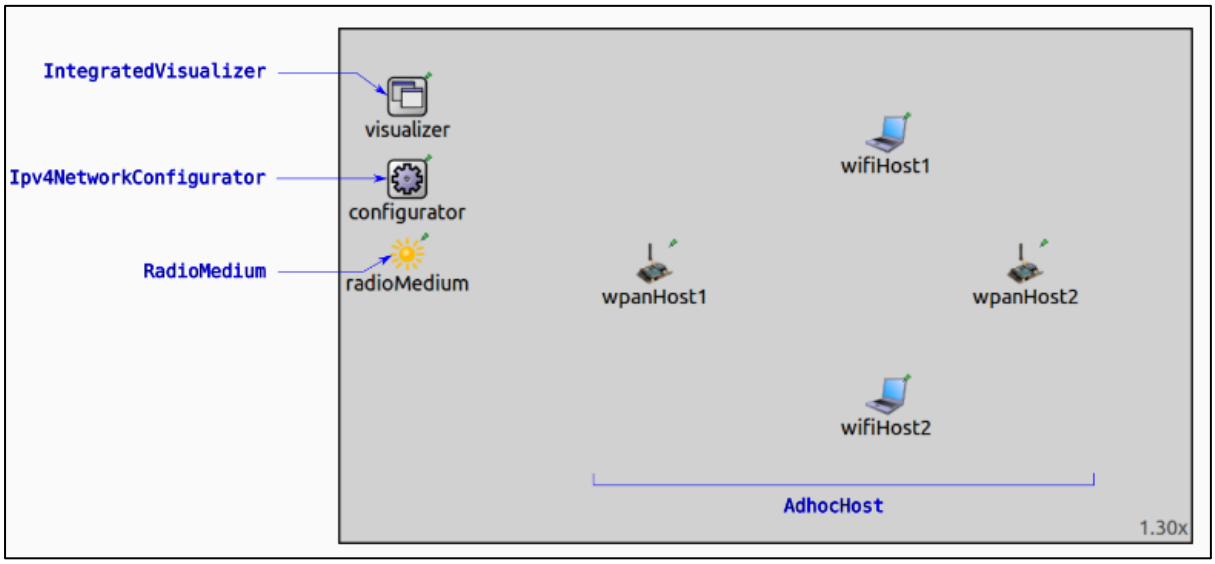
En la **Figura 40**, se puede apreciar una red que consta de cuatro ***AdhocHosts***. Dos de los hosts, **wifiHost1** y **wifiHost2**, se comunican a través de **802.11** en **modo AD-HOC**[22], mientras que los otros dos, **wpanHost1** y **wpanHost2**, utilizan **802.15.4**. Los cuatro hosts están dispuestos en un rectángulo, y se encuentran dentro del rango de comunicación entre sí (a una distancia

[22] El modo ad hoc, también llamado modo peer to peer, permite que los nodos se comunican directamente (punto a punto) sin la necesidad de un punto de acceso.

de 20 metros cada uno). En cada par de hosts, uno envía paquetes al otro (**wifiHost1** a **wifiHost2** y **wpanHost1** a **wpanHost2**).

Figura 40

Topología de la red coexistencia de 802.11 y 802.15.4



La implementación de estos *showcases* en **inet** es una aportación valiosa a la comunidad, siendo herramientas muy útiles para la simulación y evaluación de diferentes protocolos y tecnologías de redes, permitiendo recrear situaciones específicas en un entorno controlado. En este caso, la implementación de la librería de criptografía de curva elíptica y la implementación en 802.15.4 puede ser de gran ayuda para evaluar la seguridad de la red y detectar posibles vulnerabilidades.

3.4.3 Implementación de la red 802.15.4 para ECC

Una vez implementada la librería con las funciones ECC y el escenario de simulación a modificar, se procede a la ***configuración de los parámetros de la red***, en estos se describe el tipo de tecnología que tiene la red y los parámetros que pueden ser modificados como el canal, ancho de banda, bitrate, etc. también, a la ***configuración del escenario base*** este llevará todos los parámetros para simular el protocolo **802.15.4**. Además, la ***creación de los despliegues de WSN*** simulando un único escenario de prueba para tres diferentes despliegues de WSN. Finalmente se define el ***funcionamiento del nodo*** en el cual se implementa la criptografía de

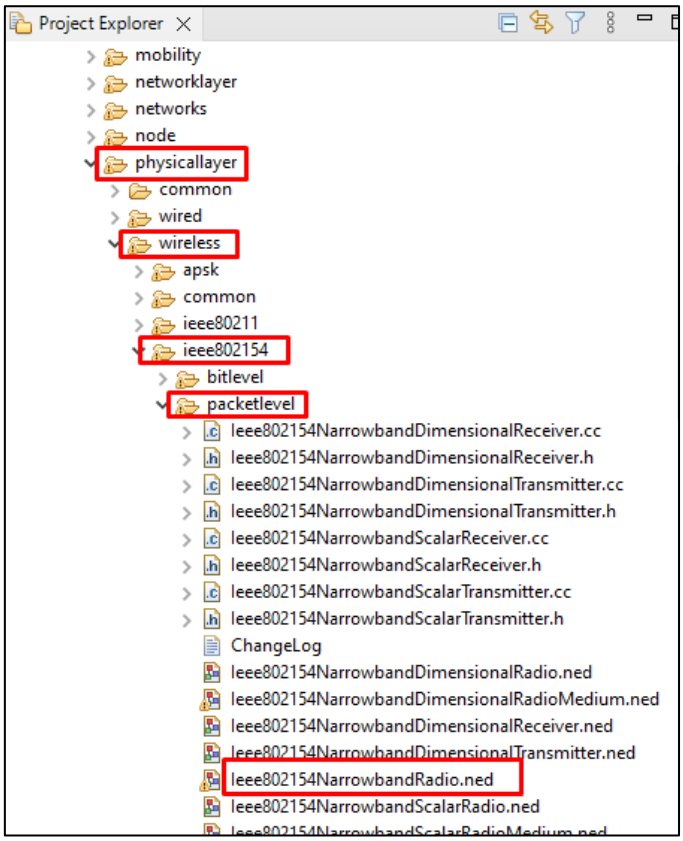
curva elíptica, además de indicar el funcionamiento del nodo, como envío, recepción y análisis de datos. Este estudio es de gran importancia y utilidad para la tecnología WSN y en general para todas las redes inalámbricas que utilicen dispositivos con baja capacidad de procesamiento y bajo consumo de energía; mejorando así la eficiencia en la comunicación y seguridad en estos dispositivos.

3.4.3.1 Configuración de los parámetros de la red

INET trabaja con modelos implementados para cada una de las capas, para este caso se necesita modificar los parámetros a nivel físico y de paquetes, es decir a nivel de las dos primeras capas (Física y Enlace de datos). En la **Figura 41** se muestra donde se encuentran los archivos necesarios para configurar el entorno de la red.

Figura 41

Archivos de configuración de los parámetros de la Red



El módulo llamado "**Ieee802154NarrowbandRadio.ned**", representa una radio de banda estrecha que opera de acuerdo con el estándar **IEEE 802.15.4**. En la **Figura 42** se muestra la sección de parámetros del código la cual define varias opciones de configuración para la radio, como la frecuencia central, el ancho de banda, la velocidad de bits, la longitud de la cabecera y la sensibilidad. Estos parámetros se utilizan para configurar los componentes transmisor y receptor del nodo sensor, que también se definen en el módulo.

Figura 42

Parámetros de configuración del canal y tramas

```
centerFrequency = default(2450 MHz);

// B_20dB ATmega256RFR2 (page 564)
bandwidth = default(2.8 MHz);

// 802.15.4-2006 (page 28)
*.bitrate = default(250 kbps);

// PHY Header (without preamble), 802.15.4-2006 (page 43)
// 1 octet SFD
// 7 bit Frame length
// 1 bit Reserved
*.headerLength = (1*8 + 7 + 1) * 1 b;

// Preamble
// 4 octets Preamble
// 1 symbol of 16us -> 4 bit
transmitter.preambleDuration = (4*8/4) * 16 us;

// RSSI sensitivity (ATmega256RFR2, page 566)
receiver.energyDetection = default(-90dBm);

// Receiver sensitivity (ATmega256RFR2, page 565)
// TODO That is not quite true, because sensitivity
//      is defined as the input signal power that yields
//      a PER < 1% for a PSDU of 20 octets, but INET handles it
//      as minimum reception power.
receiver.sensitivity = default(-100dBm);

// There is no fixed boundary, because of the
// DSSS and the capture effect. Taking the sensitivity minus some
// arbitrary value as an approximate guess.
receiver.minInterferencePower = default(-120dBm);
```

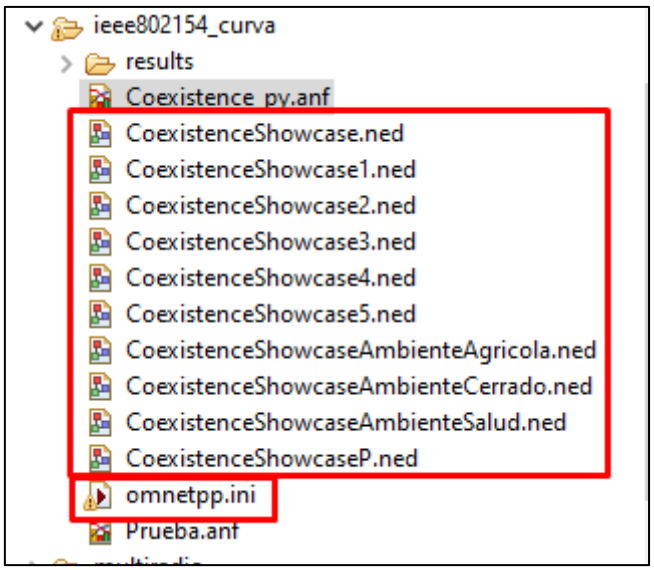
3.4.3.2 Despliegues de WSN

Para implementar la simulación de los diferentes despliegues de WSN, se trabaja creando y modificando 3 tipos de archivos. Para representar cada uno de los despliegues propuestos, se

necesita crear una nueva red, es decir un archivo .***ned*** para cada despliegue. La **Figura 43** muestra la creación de cada escenario de simulación.

Figura 43

Diferentes despliegues de WSN



Para poder llamar a cada uno de los escenarios, se realiza la configuración en el archivo .***ini*** este dispone de todas las configuraciones de cada uno de los escenarios, a continuación, se detalla el funcionamiento de los escenarios.

3.4.3.3 Configuración para el escenario base

El archivo .***ini*** indica el compartiendo de los nodos y de la red, para la configuración del escenario base se detalla el siguiente código. Las primeras líneas describen las características de la simulación como la red, que se va a visualizar y el tiempo límite para detener la simulación, la segunda parte describe las características del medio de comunicación, cuáles son los modelos que se implementan para genera un entorno **802.15.4**. La **Figura 44** muestra todo el código, todas las líneas utilizadas están comentadas para un mejor entendimiento.

Figura 44

Configuración inicial para el escenario base de WSN

```
[Config Base]
network = ieee802154_curvaShowcase  #Llama a la red principal 2 nodos transmisor y receptor
sim-time-limit = 5s # Limite maximo de la simulacion
#repeat = 8
description = "2 nodos" #Descripcion                                Caracteristicas de la simulacion

# radio medium settings
*.radioMedium.analogModel.typename = "DimensionalAnalogModel" #Se define el modelo analógico para el medio de radio
*.radioMedium.backgroundNoise.typename = "IsotropicDimensionalBackgroundNoise" #Se define el ruido de fondo isotrópico
*.radioMedium.backgroundNoise.powerSpectralDensity = -113dBmWpMHz #densidad espectral de potencia del ruido de fondo dBm/W/MHz

*.wpanHost*.wlan[*].radio.transmitter.frequencyGains = "left c-5MHz -40dB linear c-2MHz -20dB linear c-1MHz 0dB either c+1MHz 0dB
#Se define la ganancia de frecuencia del transmisor para la red inalámbrica personal (WPAN)
*.*Host*.wlan[*].radio.receiver.snirThresholdMode = "mean" #modo de umbral de relación señal/ruido
*.*Host*.wlan[*].radio.receiver.errorModel.snirMode = "mean" #relación señal/ruido

# default 2450MHz/2.8MHz bw
*.wpanHost*.wlan[*].typename = "Ieee802154NarrowbandInterface" #interfaz de red de la WPAN
*.wpanHost*.wlan[*].radio.typename = "Ieee802154NarrowbandDimensionalRadio" #tipo de radio para la WPAN    Parametros del medio
```

El escenario base también cuenta con el modelo de energía el cual simula el consumo de energía de los nodos de la red inalámbrica, empezando por un nivel de batería y perdiendo energía según los procesos que realiza el nodo. En la **Figura 45** se muestra las líneas de código para la implementación del modelo de energía, la primera parte determina el tipo de modelo a usar, la segunda parte establece los consumos del nodo y la tercera es un generador de energía, simula que el nodo se carga.

Figura 45

Configuración del modelo de energía para el escenario base de WSN

```
#************************************************ MODELO DE ENERGIA ***********************************************************
# Modelo de energia -> https://inet.omnetpp.org/docs/tutorials/wireless/doc/step8.html
*.*Host*.hasStatus = true #Cada host tendra un estado
*.*Host*.energyStorage.typename = "IdealEpEnergyStorage" #"IdealEpEnergyStorage" -Almacenamiento ideal #"SimpleEpEnergyStorage"
*.*Host*.wlan[*].radio.energyConsumer.typename = "StateBasedEpEnergyConsumer" #Tipo de consumo de energía
*.*Host*.energyManagement.typename = "SimpleEpEnergyManagement" #tipo de gestión de energía
#*.*Host*.energyStorage.nominalCapacity = 0.05J                                  Se define el tipo de estados, el
#*.*Host*.energyManagement.nodeShutdownCapacity = 0.1 * parent.energyStorage.nominalCapacity   almacenamieto de energia y el
#*.*Host*.energyManagement.nodeStartCapacity = 0.5 * parent.energyStorage.nominalCapacity      tipo de gestion
#*.*Host*.energyStorage.initialCapacity = uniform(0.1 * nominalCapacity, nominalCapacity)

*.*Host*.wlan[*].radio.energyConsumer.offPowerConsumption = 0mW # Consumo cuando esta pagado
*.*Host*.wlan[*].radio.energyConsumer.sleepPowerConsumption = 1mW # Consumo en modo sleep
*.*Host*.wlan[*].radio.energyConsumer.switchingPowerConsumption = 1mW # Consumo durante la transición de encendido a apagado o viceversa
*.*Host*.wlan[*].radio.energyConsumer.receiverIdlePowerConsumption = 2mW # Consumo durante Reposo
*.*Host*.wlan[*].radio.energyConsumer.receiverBusyPowerConsumption = 5mW # Consumo durante estado ocupado
*.*Host*.wlan[*].radio.energyConsumer.receiverReceivingPowerConsumption = 10mW # Consumo durante recepcion     Consumos del nodo
*.*Host*.wlan[*].radio.energyConsumer.transmitterIdlePowerConsumption = 2mW # Consumo durante recepcion       en cada etapa
*.*Host*.wlan[*].radio.energyConsumer.transmitterTransmittingPowerConsumption = 100mW # Consumo durante transmision

#*.*Host*.energyGenerator.typename = "AlternatingEpEnergyGenerator"
#*.*Host*.energyGenerator.powerGeneration = 4mW                  Generador de
#*.*Host*.energyGenerator.sleepInterval = exponential(25s)       Energia
#*.*Host*.energyGenerator.generationInterval = exponential(25s)

#*******************************************************************************************************************
```

El escenario base define las características del nodo estación base, En la **Figura 46** se observa todos los parámetros a ser configurado como puertos, tamaño del paquete, intervalos de envío, etc. El parámetro más importante es el que define el tipo de nodo, este hace referencia a un archivo *.cc* "**UdpBasicApp**" el cual controla todas las características del nodo y el cual se ha modificado para la implementación de ECC. En la sección **Configuración de los nodos** se muestra la modificación de este archivo.

Figura 46

Configuración del nodo estación base "BSTA" para el escenario base de WSN

```
#*************************************************************************************************************
# Configuracion basica del nodo BSTA
*.wpanHost1.numApps = 1 # Cantidad de nodos
*.wpanHost1.app[0].typename = "UdpBasicApp" # Tipo de nodo
*.wpanHost1.app[0].destAddresses = "wpanHost2" # Direccion de envio de datos
*.wpanHost1.app[0].destPort = 5000 # Puerto de destino
*.wpanHost1.app[0].localPort = 5000 # Puerto origen
*.wpanHost1.app[0].messageLength = 88byte # Tamaño del mensaje
*.wpanHost1.app[0].packetName = "UDPData-wpan-" # Nombre del paquete
*.wpanHost1.app[0].startTime = uniform(0.1us, 0.01s) # Tiempo de inicio aleatorio del nodo
*.wpanHost1.app[0].sendInterval = 0.1s # Intervalos de envio de datos
*.wpanHost1.app[0].stopTime = 0.09s # tiempo de finalización del nodo
*.wpanHost1.app[0].receiveBroadcast = true # El nodo puede recibir transmisiones de difusión
*.wpanHost1.app[0].numberOfNodes = 1 # Numero de nodos participantes en la red
*.wpanHost1.app[0].multicastInterface = "wlan0" # Nombre de la interfaz
*.wpanHost1.app[0].joinLocalMulticastGroups = true # Permite el nodo responda grupos de multidifusión
*.wpanHost1.app[0].isBSTA = true # Estacion Base
*.wpanHost1.app[0].nodoId = 1 # Identificador del nodo        Configuracion del funcionamiento de la Estacion Base

#*************************************************************************************************************
```

Finalmente, el escenario define las configuraciones de visualización y registro de eventos en la simulación, la **Figura 47** muestra el código para la visualización, la primera parte establece todos los eventos que se pueden visualizar en la simulación como el envío de datos, pérdida de paquetes, enlaces de comunicación, etc., la segunda parte determina el protocolo ARP.

Figura 47

Configuración de los eventos de visualización para el escenario base de WSN

```
#**************************************************************************************************
# visualizer settings
*.visualizer.physicalLinkVisualizer.displayLinks = true # Muestra los enlaces físicos entre los nodos
*.visualizer.dataLinkVisualizer.displayLinks = true # Muestra los enlaces de datos entre los nodos

*.visualizer.mediumVisualizer.displaySignals = true # muestra las señales en el medio de transmisión
*.visualizer.mediumVisualizer.displayTransmissions = true # muestra las transmisiones que ocurren en el medio
*.visualizer.mediumVisualizer.displayReceptions = true # muestra las recepciones que ocurren en el medio
*.visualizer.mediumVisualizer.displaySpectrums = false # muestra el espectro de frecuencia de la señal

*.visualizer.mediumVisualizer.signalRingSize = 5m # tamaño del anillo que se muestra en el visualizador
*.visualizer.mediumVisualizer.signalWaveLength = 5m # longitud de onda de la señal que se muestra en el visualizador

*.visualizer.infoVisualizer.format = "%t" # formato de visualización de la información en el visualizador
*.visualizer.infoVisualizer.placementHint = "topCenter" # Ubicación donde se muestra la información en el visualizador
*.visualizer.infoVisualizer.placementPriority = -10 # prioridad de la información que se muestra en el visualizador

*.visualizer.routingTableVisualizer.displayRoutingTables = false # muestra las tablas de enrutamiento

*.visualizer.packetDropVisualizer.displayPacketDrops = true # mostrar los paquetes que se han perdido
*.visualizer.packetDropVisualizer.labelFormat = "%r" # etiquetas de los paquetes perdidos      Visualizacion de enventos
#**************************************************************************************************
# arp type
*.*Host*.ipv4.arp.typename = "GlobalArp" # tipo de protocolo ARP      Porotocolo ARP

# for video recording
#*.visualizer.*.fadeOutMode = "animationTime" # modo de desvanecimiento de animación
```

3.4.3.4 Configuración para todos los escenarios

Para cada uno de los escenarios se extiende la configuración base, permitiendo así adaptar la configuración de **802.15.4** para cualquier despliegue de WSN. La **Figura 48** muestra la implementación del escenario 6 llamado "**Monitorización en Ambiente Cerrado**" la primera parte extiende la configuración del escenario base, la segunda parte determina los parámetros para la estación base (BSTA), especificando que tendrá comunicación con cada una de las *interfaces de los nodos*, la *cantidad de nodos* sensores que tendrá la red, los tiempos de intervalo de envio, etc. La tercera parte es la configuración específica para cada nodo sensor, especificando el *identificador del nodo* y la *cantidad de datos* que va a enviar; si los datos superan el tamaño máximo permitido por el paquete ***88bytes*** se segmentan y se envían utilizando varias tramas.

Figura 48

Configuración de los nodos para el escenario 6 llamado "Monitorización en Ambiente cerrado"

```
[Config Escenario_6]
extends = Base # Se exitende las configuraciones del escenario base
network = ieee802154_curvaShowcaseAmbienteCerrado # Llamado a la red
description = "Monitorizacion en Ambiente Cerrado" # Descripcion
```

Extender las configuraciones del escenario base al nuevo escenario

```
*.wpanHost1.app[0].destAddresses = "wpanHost2 wpanHost3 wpanHost4" #Direciones destino para el envio de datos de la BSTA
*.wpanHost1.app[0].numberOfNodes = 3 # cantidad de nodos en la red
*.wpanHost1.app[0].modeMulticast = false # No multicast
*.wpanHost1.app[0].sendInterval = 0.1s # Intervalo de envio de datos
*.wpanHost1.app[0].stopTime = 1.0s  # Tiempo de finalizacion del nodo
*.wpanHost1.app[0].startTime = uniform(0.5us, 0.01s) #Inicio del nodo aleatorio
```

Nodo Estacion Base "BSTA"

```
*.wpanHost2.numApps = 1 # Configuracion para el 2do nodo
*.wpanHost2.app[0].nodoId = 2 # Identificador del 2do nodo
*.wpanHost2.app[0].nDatosTxMax = 4 # Cantidad de datos que envia el nodo

*.wpanHost3.numApps = 1 # Configuracion para el 3cer nodo
*.wpanHost3.app[0].nodoId = 3 # Identificador del 3cer nodo
*.wpanHost3.app[0].nDatosTxMax = 4 # Cantidad de datos que envia el nodo

*.wpanHost4.numApps = 1 # Configuracion para el 4to nodo
*.wpanHost4.app[0].nodoId = 4 # Identificador del 4to nodo
*.wpanHost4.app[0].nDatosTxMax = 4 # Cantidad de datos que envia el nodo
```

Configuracion para los nodos sensores

Finalmente la nueva red, especifica las configuraciones generales para los nodos sensores, hay que aclarar que las configuraciones tienen un orden y van de lo particular hasta lo más general, es decir se toma primero las configuraciones iniciales de manera particular y después se puede especificar configuraciones generales para los demás nodos. La **Figura 49** muestra la configuración de los nodos sensores, especificando el *tipo de nodo* "**UdpBasicApp**", la *dirección de destino* de los nodos sensores; El *tamaño del paquete* de ***88bytes*** se utiliza este valor dado que la red maneja mensajes pequeños para minimizar la carga de tráfico en la red y mejorar la eficiencia de la transmisión de datos, además se especifica los *intervalos de tiempo* para el envio de los nodos.

Figura 49

Configuración general de los nodos sensores en el escenario "Monitorización en Ambiente cerrado"

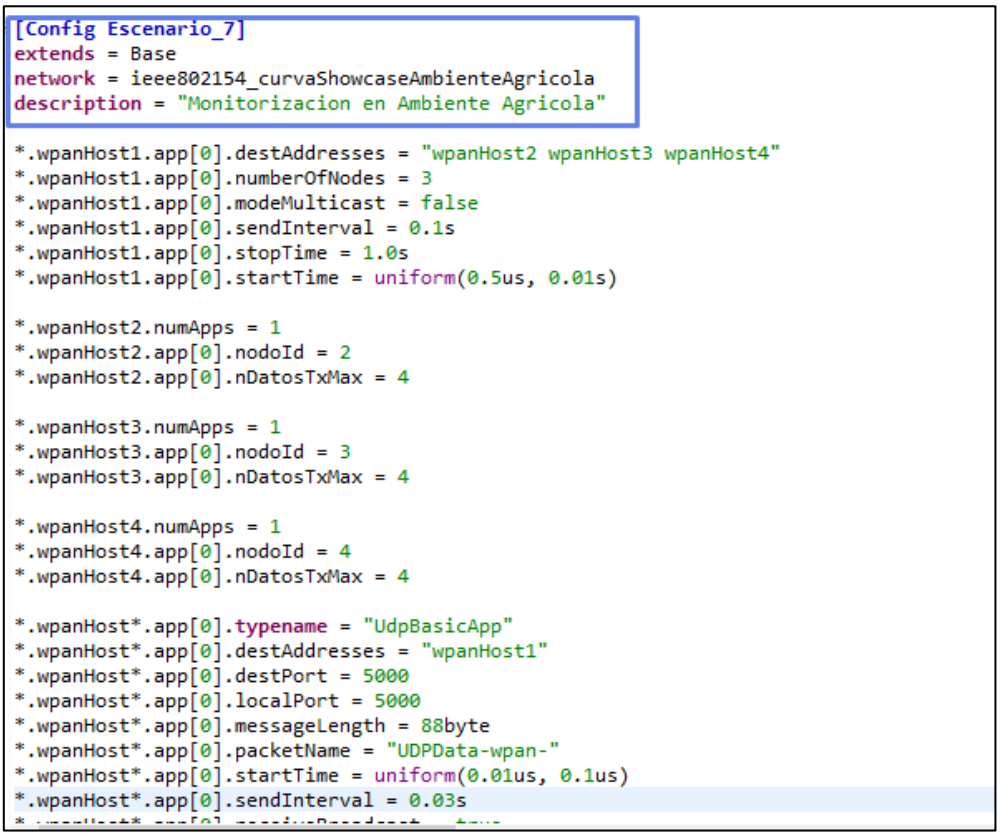
```
*.wpanHost*.app[0].typename = "UdpBasicApp" # Tipo de nodo Configuracion para todos los nodos
*.wpanHost*.app[0].destAddresses = "wpanHost1" #Direccion de destino para los nodos sensores
*.wpanHost*.app[0].destPort = 5000 # Puerto destino
*.wpanHost*.app[0].localPort = 5000 # Puerto Origen
*.wpanHost*.app[0].messageLength = 88byte # Tamaño del paquete
*.wpanHost*.app[0].packetName = "UDPData-wpan-" # Nombre del paquete
*.wpanHost*.app[0].startTime = uniform(0.01us, 0.1us) #Intervalo de inicio del nodo
*.wpanHost*.app[0].sendInterval = 0.03s # Intervalo de envio del nodo
*.wpanHost*.app[0].receiveBroadcast = true # Recibir transacciones broadcast
*.wpanHost*.app[0].multicastInterface = "wlan0" # Tipo de interfaz para los nodos sensores
*.wpanHost*.app[0].joinLocalMulticastGroups = true # Unirse a multidifusion
```

Configuracion General para todos los nodos sensores

Todas las redes presentan la misma implementación, En la **Figura 50** se muestra la implementación para la red llamada "**Monitorización en Ambiente Agrícola**", nótese como extiende los parámetros del escenario base e implementa la configuración de cantidad de nodos y envío de paquetes.

Figura 50

Configuración para el escenario 7 - "Monitorización en Ambiente Agrícola"



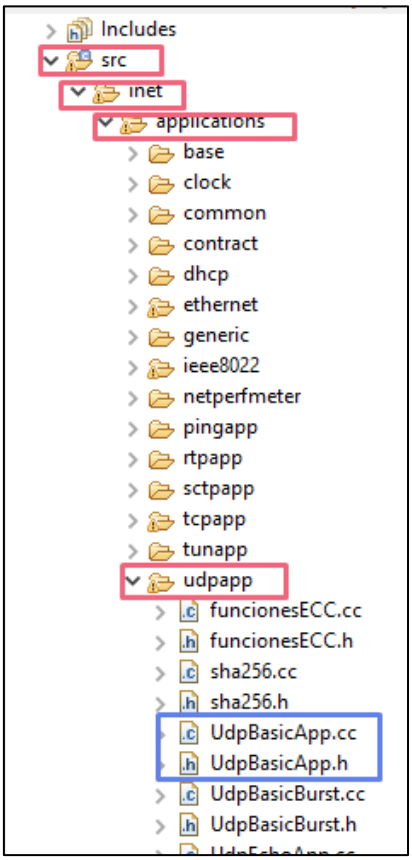
```
[Config Escenario_7]
extends = Base
network = ieee802154_curvaShowcaseAmbienteAgricola
description = "Monitorizacion en Ambiente Agricola"

*.wpanHost1.app[0].destAddresses = "wpanHost2 wpanHost3 wpanHost4"
*.wpanHost1.app[0].numberOfNodes = 3
*.wpanHost1.app[0].modeMulticast = false
*.wpanHost1.app[0].sendInterval = 0.1s
*.wpanHost1.app[0].stopTime = 1.0s
*.wpanHost1.app[0].startTime = uniform(0.5us, 0.01s)

*.wpanHost2.numApps = 1
*.wpanHost2.app[0].nodoId = 2
*.wpanHost2.app[0].nDatosTxMax = 4

*.wpanHost3.numApps = 1
*.wpanHost3.app[0].nodoId = 3
*.wpanHost3.app[0].nDatosTxMax = 4

*.wpanHost4.numApps = 1
*.wpanHost4.app[0].nodoId = 4
*.wpanHost4.app[0].nDatosTxMax = 4

*.wpanHost*.app[0].typename = "UdpBasicApp"
*.wpanHost*.app[0].destAddresses = "wpanHost1"
*.wpanHost*.app[0].destPort = 5000
*.wpanHost*.app[0].localPort = 5000
*.wpanHost*.app[0].messageLength = 88byte
*.wpanHost*.app[0].packetName = "UDPData-wpan-"
*.wpanHost*.app[0].startTime = uniform(0.01us, 0.1us)
*.wpanHost*.app[0].sendInterval = 0.03s
```

3.4.3.5 Configuración de los nodos

El funcionamiento del nodo, sus características y toma de decisiones esta especificado en el archivo "**UdpBasicApp**", La **Figura 51** muestra la ubicación del archivo dentro de la librería **inet4.4**.

Figura 51

Ubicación del archivo "UdpBasicApp" para controlar el funcionamiento de los nodos

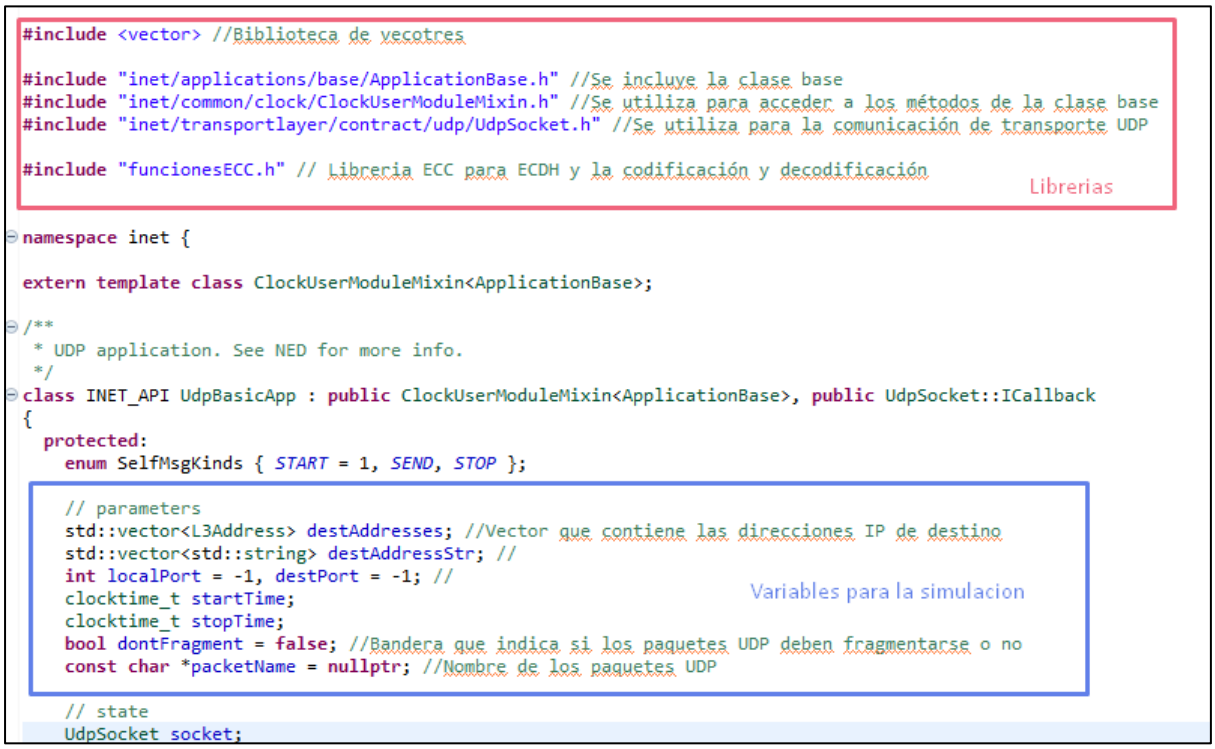


El archivo ha sido modificado para implementar criptografía de curvas elípticas, a continuación, se detalla todo el código utilizado para la toma de decisiones de los nodos. Como se trata de un archivo en $\boldsymbol{C++}$, este dispone de dos componentes el archivo $\boldsymbol{.h}$ para la declaración de variables y funciones que se van a utiliza en los nodos; El segundo archivo $\boldsymbol{.cc}$ el cual define las funciones, clases y método que utilizaran los nodos.

En la **Figura 52** se muestra las librerías utilizadas para el funcionamiento del nodo, en este caso se implementa la librería para el uso de vectores y matrices, la librería creada llamada "**funcionesECC.h**" la misma que implementa todas las primitivas matemáticas para implementar criptografía de curvas elípticas con campos finitos primos; Además especifica las variables utilizadas en la simulación como: variables para las direcciones IP, puertos, nombre de los paquetes, etc.

Figura 52

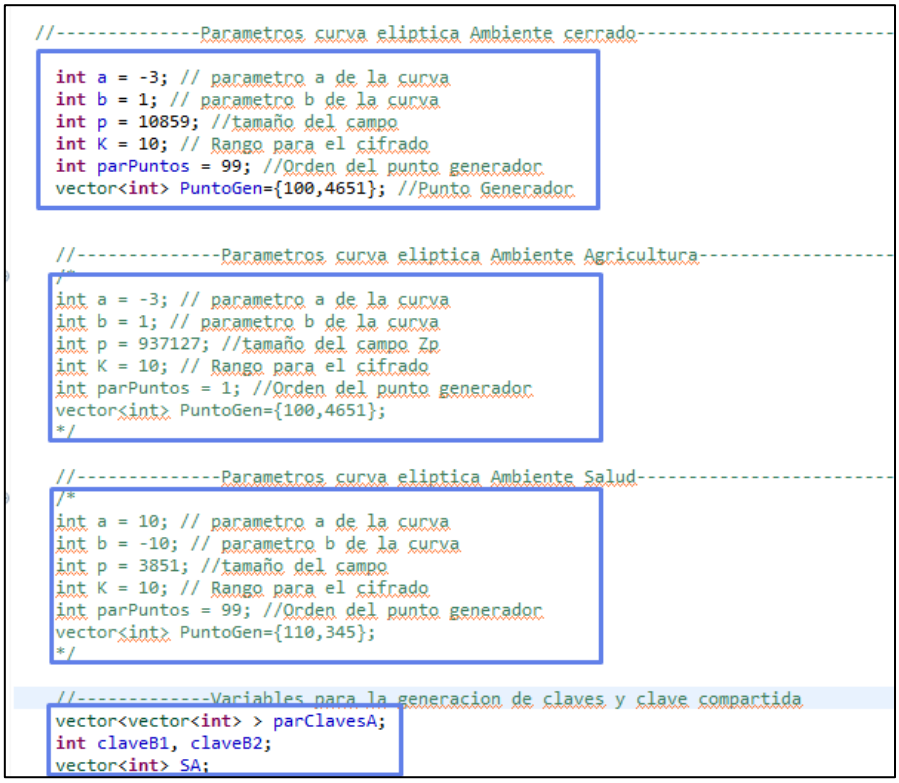
Librerías y variables utilizadas en el archivo UdpBasicApp.h



La **Figura 53** muestra también los parámetros de la curva elíptica y campo para cada uno de los escenarios, la elección de estas curvas se describe en la sección **Prueba 3: Elección de la curva elíptica para la WSN y demostración de los algoritmos ECDH y ElGamal**, además se establece las variables para la generación de claves y obtención de la clave secreta compartida.

Figura 53

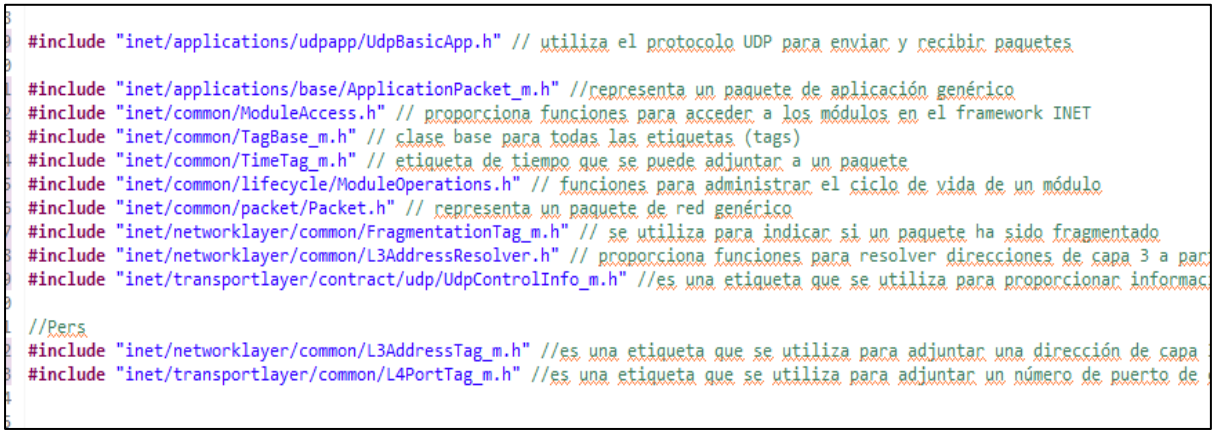
Parámetros de las curvas elípticas, campos finitos primos y orden del punto, para cada escenario de simulación en el archivo UdpBasicApp.h



El segundo archivo . ***cc*** se utiliza para definir las funciones, clases y método que utilizarán los nodos para enviar y recibir información. En la **Figura 54** se muestra las librerías utilizadas, la mayoría de estas son para el proceso de encapsulamiento de datos usando el protocolo TCP/IP.

Figura 54

Librerías para el encapsulamiento de datos y representación de paquetes en el archivo UdpBasicApp.cc

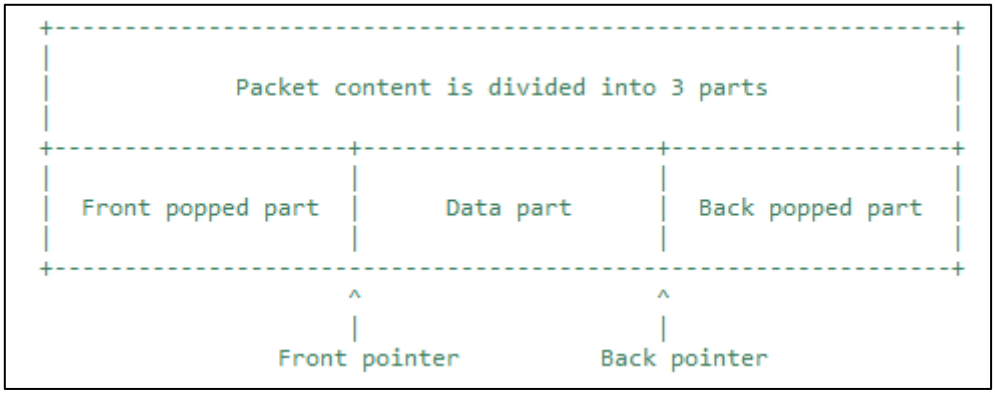


La representación de paquetes es fundamental para la simulación de redes de comunicación. Las aplicaciones y los protocolos de comunicación construyen, deconstruyen, encapsulan, fragmentan, agregan y manipulan paquetes de muchas maneras.

Durante el procesamiento de paquetes, a medida que el paquete es pasado a través de las capas de protocolo en el receptor, las cabeceras y los remates son eliminados desde el principio y el final. Esto reduce efectivamente la parte no procesada que se llama la parte de datos, pero no afecta los datos almacenados en el paquete. En la **Figura 55** se muestra la estructura básica de los paquetes, estos disponen de propiedades de división, donde cada paquete puede contener atributos al inicio, y al final, aparte de los datos; Se utiliza el frente del paquete (Front popped) para el direccionamiento y la parte del final (Back popped) para garantizar la autenticidad de los datos.

Figura 55

Encapsulación de datos en paquetes



El archivo "**UdpBasicApp.cc**" define la función para el inicio del nodo, en la **Figura 56** se muestra el código el cual permite que el nodo obtenga los parámetros del archivo . ***ini***, aquí se cargan las variables de direcciones IP, puertos, tiempo de simulación, tipo de interfaz, etc.

Figura 56

Método de inicio, cargar datos del archivo .ini

```
void UdpBasicApp::initialize(int stage)        Metodo de Incio
{
    ClockUserModuleMixin::initialize(stage);

    if (stage == INITSTAGE_LOCAL) {
        numSent = 0;
        numReceived = 0;
        WATCH(numSent);
        WATCH(numReceived);

        //Inicializacion de parametros (ver ini config.)
        localPort = par("localPort");
        destPort = par("destPort");
        startTime = par("startTime");
        stopTime = par("stopTime");
        packetName = par("packetName");
        dontFragment = par("dontFragment");
        if (stopTime >= CLOCKTIME_ZERO && stopTime < startTime)
            throw cRuntimeError("Invalid startTime/stopTime parameters");
        selfMsg = new ClockEvent("sendTimer");

        nodes = par("numberOfNodes");
        modeMulticast = par("modeMulticast");
        sinAlgoritmo = par("sinAlgoritmo");
        nTransmision = par("nTransmision");
        isBSTA = par("isBSTA");
        nodoId = par("nodoId");
        nDatosTxMax = par("nDatosTxMax");
        sumidero = par("sumidero");
    }                                          Inicio de las variables
                                               obteniendolas
                                               del archivo .ini
```

Además, el archivo "**UdpBasicApp.cc**" también define la función "**setSocketOptions()**" vista en la **Figura 57**, que se encarga de configurar las opciones del *socket* utilizado para enviar y recibir los paquetes UDP. La configuración de estas opciones es importante para ajustar el comportamiento del socket según las necesidades de la aplicación, como la dirección IP del grupo multicast al que se enviarán los paquetes, el tiempo de vida de los paquetes, entre otras opciones.

Figura 57

Configuración del socket

```
void UdpBasicApp::setSocketOptions()
{
    //valores para configurar el socket del nodo
    int timeToLive = par("timeToLive");
    if (timeToLive != -1)
        socket.setTimeToLive(timeToLive);

    int dscp = par("dscp");
    if (dscp != -1)
        socket.setDscp(dscp);

    int tos = par("tos");
    if (tos != -1)
        socket.setTos(tos);

    const char *multicastInterface = par("multicastInterface");

    if (multicastInterface[0]) { //Si el parámetro multicastInterface no está vacío se obtiene una tabla de interfaces-----
        IInterfaceTable *ift = getModuleFromPar<IInterfaceTable>(par("interfaceTableModule"), this);
        NetworkInterface *ie = ift->findInterfaceByName(multicastInterface);
        if (!ie)
            throw cRuntimeError("Wrong multicastInterface setting: no interface named \"%s\"", multicastInterface);
        socket.setMulticastOutputInterface(ie->getInterfaceId());
        EV << "multicastInterface: " << multicastInterface << endl;
    }
```

Valores para la configuracion del socket

Se obtiene las nterfaces de la red

En la **Figura 58** se muestra el método "**chooseDestAddr**" el cual se encarga de elegir una dirección de destino para los paquetes que serán enviados a través del socket UDP del nodo. La elección de la dirección de destino es aleatoria, y se elige una de entre varias direcciones posibles almacenadas en un vector llamado **destAddresses**.

Figura 58

Configuración de la elección de direcciones IPv4

```
L3Address UdpBasicApp::chooseDestAddr()
{
    int k = intrand(destAddresses.size());
    if (destAddresses[k].isUnspecified() || destAddresses[k].isLinkLocal()) {
        L3AddressResolver().tryResolve(destAddressStr[k].c_str(), destAddresses[k]);
    }
    return destAddresses[k];
}
```

Para la implementación del intercambio de claves se utiliza el método "**buildClave()**" **Figura 59**, este método se encarga de llamar a las funciones "**parametrosEcc**" y "**generadorClavesEcc**" obtenidos de la librería "**FuncionesEcc**", además se encarga de encapsular esta clave y etiquetarla con el tipo de dato, esto se realiza para el establecimiento de una clave en común entre la estación base y los nodos.

Figura 59

Método para la creación de claves

```
Packet *UdpBasicApp::buildClave()
{
    std::cout << "*SendPacket en t= " << simTime() << "\n";
    vector<vector<int> > Puntos = parametrosEcc(a, b, p, parPuntos);
    //#########################################################
    vector<int> G { Puntos[1][0], Puntos[1][1]};
    int n = Puntos[2][0];

    parClavesA = generadorClavesEcc(a, b, p, G, n); //Generamos claves para la estacion base
    EV << "La clave publica de A es: [" << parClavesA[1][0] << " " << parClavesA[1][1] << "]" << endl;

    std::ostringstream str;
    str << "BROADCAST-envio";
    Packet *packet = new Packet(str.str().c_str());
    if (dontFragment)
        packet->addTag<FragmentationReq>()->setDontFragment(true);
    const auto& payload = makeShared<ApplicationPacket>();
    payload->setChunkLength(B(par("messageLength")));
    payload->setSequenceNumber(numSent);

    char parClaves_str[128];
    sprintf(parClaves_str,"[%d %d]",int(parClavesA[1][0]),int(parClavesA[1][1]));
    payload->setStrMsg(parClaves_str);

    payload->appendIntMsg(int(parClavesA[1][0]));
    payload->appendIntMsg(int(parClavesA[1][1]));

    string strKind = "BROADCAST-envio";
    payload->setStrKind(strKind.c_str());

    payload->addTag<CreationTimeTag>()->setCreationTime(simTime());
    packet->insertAtBack(payload);

    return packet;
```

Leemos los parametros de la curva eliptica

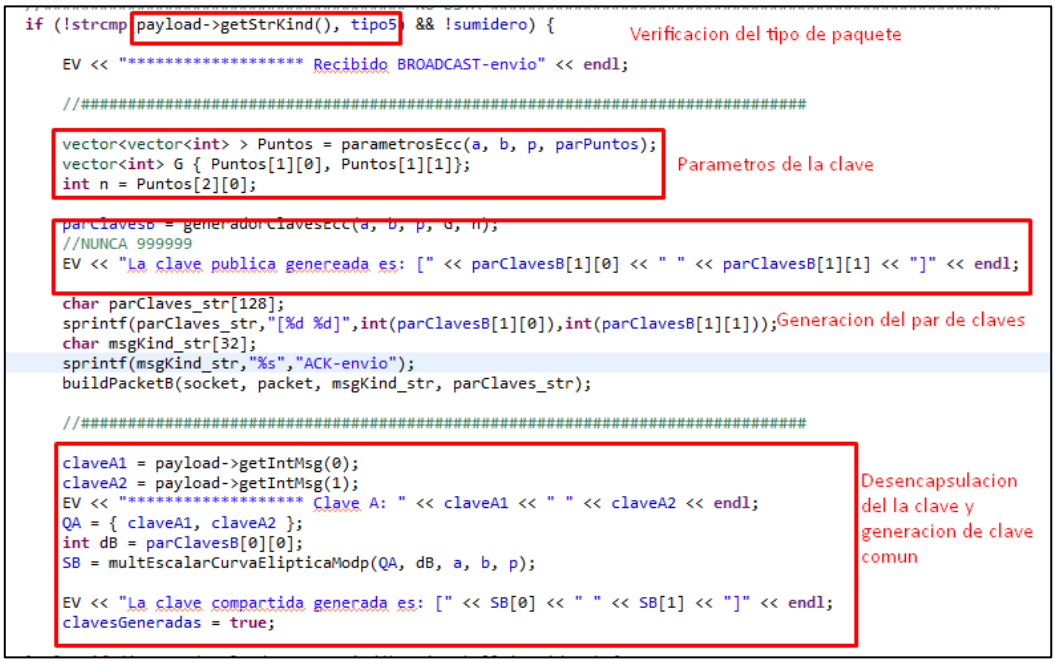
Creaccion de las claves

Encapsulacion de las claves

Todo se maneja a través de los tipos de datos, los mismos que representan a la trama; Así el otro nodo verifica el tipo de dato y realiza la función del establecimiento de una clave común, el proceso es similar: primero generando el par de claves propio y después desencapsulando el paquete y generando una clave común. En la **Figura 60** se muestra el establecimiento de la clave común primero obteniendo la clave pública del otro nodo y multiplicando la clave privada del nodo propio.

Figura 60

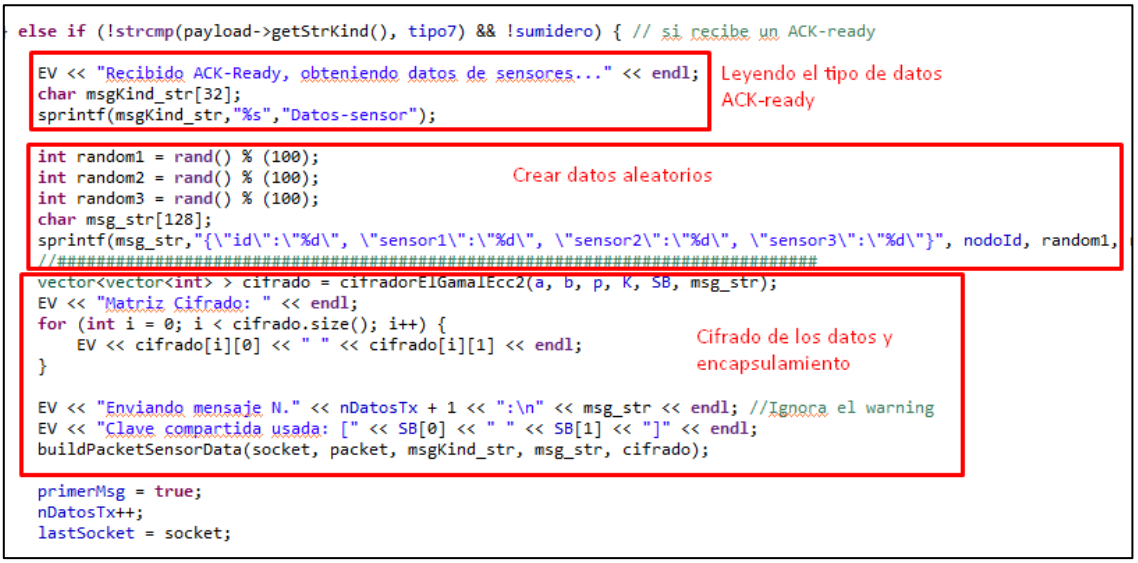
Establecimiento de la clave común



La **Figura 61** muestra el proceso de cifrado, el cual se activa cuando el nodo recibe el tipo de dato "**ACK-ready**" aquí se generan valores aleatorios simulando los datos de los sensores y se llama al método "**cifradorElGamalEcc2**" para generar una matriz que contenga los puntos del mensaje cifrado, finalmente se encapsula y se prepara el envio de la trama.

Figura 61

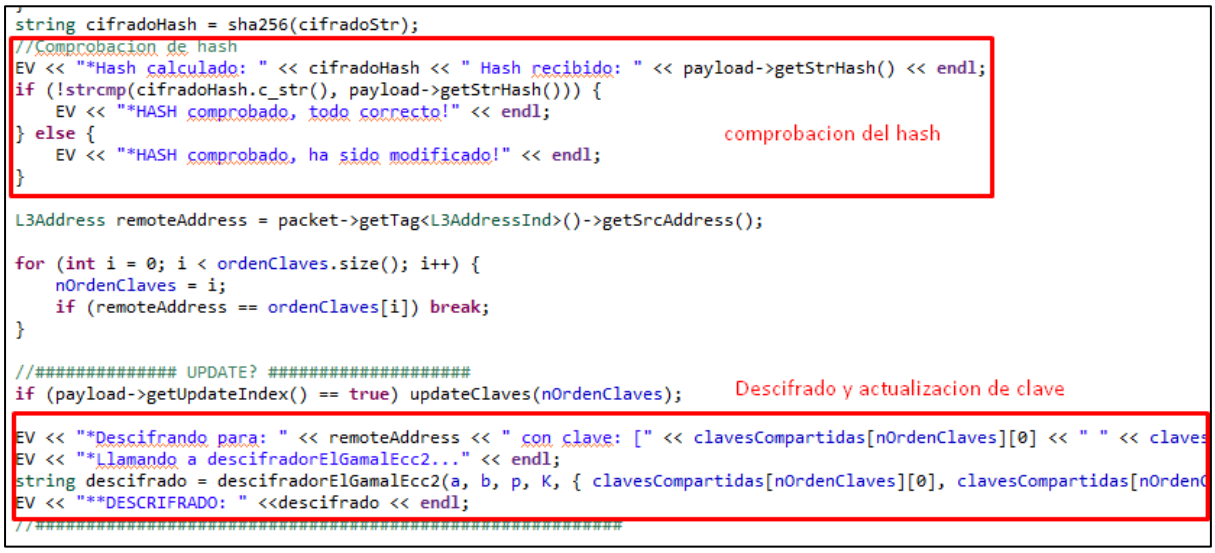
Proceso de cifrado de datos para los nodos sensores



La **Figura 62** muestra el proceso de descifrado de la estación base, esta primero comprueba el hash de los datos y después utiliza la calve compartida para descifrar el mensaje llamando a la función "**descifradorElGamalEcc2**" también llama al método "**updateClaves()**" para la actualización de la calve.

Figura 62

Proceso de descifrado de datos en la estación base



3.4.4 Modelo del consumo de energía de INET

El modelo de energía de INET está incluido en la versión ***inet4.4*** e incluye el *modelo de consumo de energía*, el *modelo de generación de energía* y el *modelo de almacenamiento temporal de energía*, en este caso solo se utiliza el modelo de consumo de energía, todos los modelos se encuentran disponible en https://inet.omnetpp.org/docs/users-guide/ch-power.html# este es un componente clave que será utilizado en la simulación de los diferentes despliegues de redes. Este modelo permite medir el consumo de energía en los dispositivos inalámbricos asignando valores de consumo a cada uno de los eventos, la disposición de valores es la siguiente y pueden estar sujetos a cambios, aun así se utiliza la configuración de los parámetros por defecto:

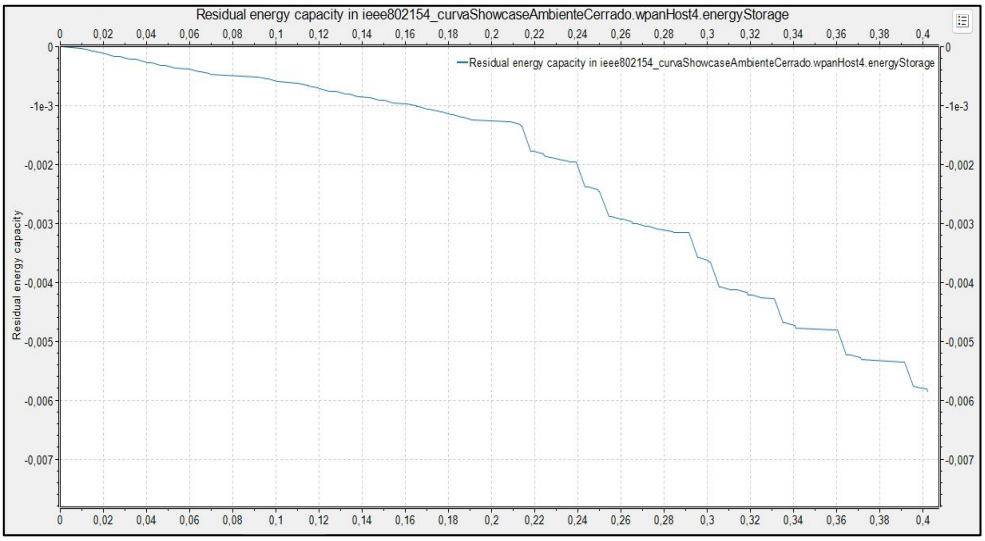
- **Consumo cuando el nodo esta apagado:** 0mW
- **Consumo cuando el nodo está en estado de suspensión "modo sleep":** 1mW
- **Consumo durante la transición de encendido a apagado:** 1mW

- **Consumo durante el reposo del nodo:** 2mW
- **Consumo durante el estado de ocupado:** 5mW
- **Consumo durante recepción:** 10mW
- **Consumo durante transmisión:** 100mW

En la **Figura 63** se muestra un ejemplo donde se observa todo el nivel de energía consumido durante toda la simulación. El nodo inicia con un nivel de energía de **0**, con el pasar del tiempo y los eventos generados este nivel va disminuyendo, exactamente después de $\mathbf{0.41}$ ($\boldsymbol{s}$), el nodo obtiene $\mathbf{-0.0058}$ ($\boldsymbol{j}$) pudiéndose comprobar cuál es el gasto de energía que realiza todo el nodo.

Figura 63

Grafica del consumo de energía del nodo 4, perteneciente a la red ambiente cerrado



En la **Figura 64** se establece el modelo de energía implementado, en el cual se establece el consumo de energía de diferentes componentes de la red inalámbrica, como la radio, el almacenamiento de energía y la gestión de energía.

Figura 64

Modelo de energía para todos los escenarios de simulación

```
#************************************** MODELO DE ENERGIA ***************************************
# Modelo de energia -> https://inet.omnetpp.org/docs/tutorials/wireless/doc/step8.html
*.*Host*.hasStatus = true
*.*Host*.energyStorage.typename = "IdealEpEnergyStorage" #"IdealEpEnergyStorage" #"SimpleEpEnergyStorage"
*.*Host*.wlan[*].radio.energyConsumer.typename = "StateBasedEpEnergyConsumer"
*.*Host*.energyManagement.typename = "SimpleEpEnergyManagement"
#*.*Host*.energyStorage.nominalCapacity = 0.05J
#*.*Host*.energyManagement.nodeShutdownCapacity = 0.1 * parent.energyStorage.nominalCapacity
#*.*Host*.energyManagement.nodeStartCapacity = 0.5 * parent.energyStorage.nominalCapacity
#*.*Host*.energyStorage.initialCapacity = uniform(0.1 * nominalCapacity, nominalCapacity)

*.*Host*.wlan[*].radio.energyConsumer.offPowerConsumption = 0mW
*.*Host*.wlan[*].radio.energyConsumer.sleepPowerConsumption = 1mW
*.*Host*.wlan[*].radio.energyConsumer.switchingPowerConsumption = 1mW
*.*Host*.wlan[*].radio.energyConsumer.receiverIdlePowerConsumption = 2mW
*.*Host*.wlan[*].radio.energyConsumer.receiverBusyPowerConsumption = 5mW
*.*Host*.wlan[*].radio.energyConsumer.receiverReceivingPowerConsumption = 10mW
*.*Host*.wlan[*].radio.energyConsumer.transmitterIdlePowerConsumption = 2mW
*.*Host*.wlan[*].radio.energyConsumer.transmitterTransmittingPowerConsumption = 100mW
```

Al establecer los consumos de energía, se puede simular el consumo energético de la red inalámbrica y analizar el impacto de diferentes configuraciones, así como su rendimiento. Este modelo de energía puede ser utilizado para optimizar el consumo energético de una red.

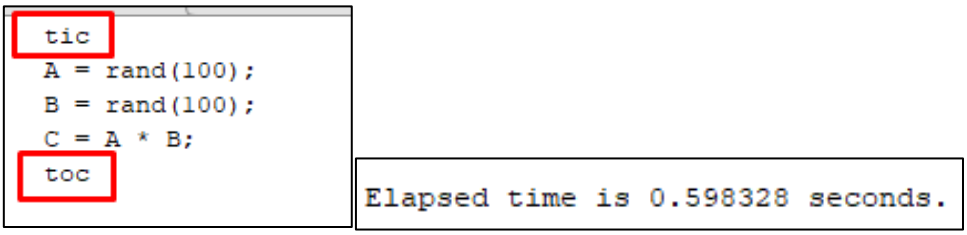
3.4.5 Análisis del procesamiento del sistema

En Omnet++ no existe una herramienta para comprobar la cantidad de procesamiento de cada uno de los nodos, ni tampoco para determinar el tamaño de las claves, por ello es necesario buscar otra herramienta para obtener estos valores. MATLAB dispone de varias herramientas que nos permiten monitorear y administrar el uso de memoria en un programa, medir el tiempo de ejecución de un bloque de código o una función e identificar los cuellos de botella en el rendimiento de un código. A continuación, se detalla el funcionamiento de estas herramientas:

- El comando "**tic - toc**" se utiliza para marcar el tiempo de ejecución del programa, es decir el tiempo en el que está en uso el procesador; "**tic**" inicia un temporizador interno en MATLAB, el comando "**toc**" se utiliza para detener el temporizador y medir el tiempo transcurrido. Es útil para medir el tiempo de ejecución de un bloque de código o una función. En la **Figura 65** se observa el resultado de haber implementado el comando "**tic-toc**" aplicado a un conjunto de funciones.

Figura 65

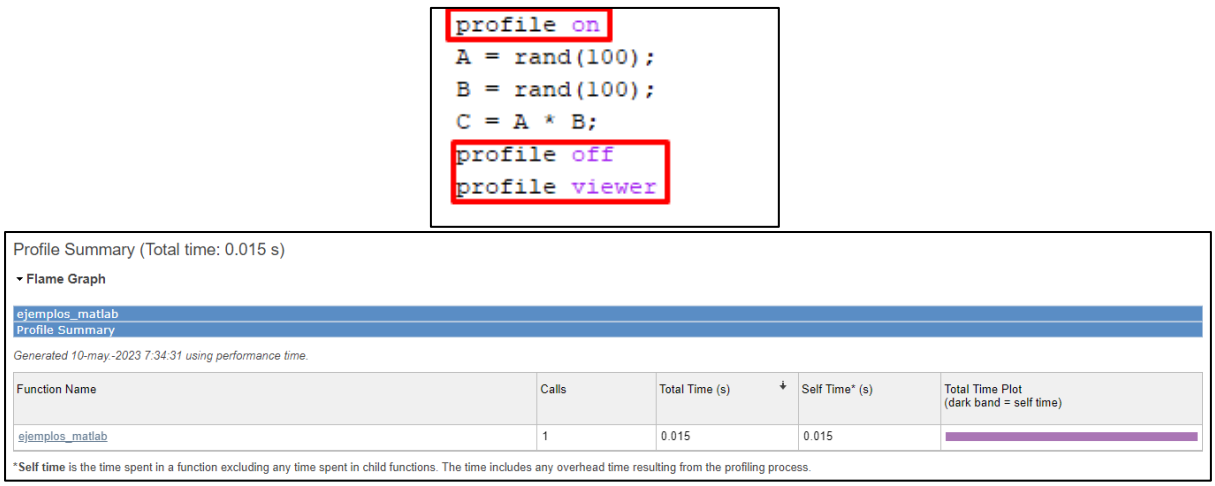
Comando "tic-toc" y resultado obtenido



- El comando "**profile on**" se utiliza para habilitar el perfilado de código. Esto es una técnica utilizada para medir el tiempo que tarda una función o un bloque de código en ejecutarse, y para identificar los cuellos de botella en el rendimiento. Una vez habilitado el perfilado de código, se puede ejecutar el código y obtener un informe detallado del tiempo de ejecución de cada función en el código. En la **Figura 66** se observa el resultado de haber implementado el comando "**profile on**" en un conjunto de operaciones, además para detener el perfilado se utiliza el comando "**profile off**" y para visualizar el contenido se utiliza el comando "**profile viewer**".

Figura 66

Comando "profile on" y resultado de la ejecución



- El comando "**memory**" se utiliza para mostrar información acerca de la memoria disponible y en uso en MATLAB. Al utilizar este comando, se mostrarán detalles como la cantidad total de memoria disponible, la cantidad de memoria que está siendo utilizada por MATLAB en ese momento, la cantidad de memoria disponible para usar por el usuario y la cantidad de memoria reservada para el sistema operativo. En la **Figura 70** se observa el resultado de haber implementado el comando "**memory**".

Figura 67

Comando memory y resultado de la ejecución

```
A = rand(100);
B = rand(100);
C = A * B;
memory
```

```
Maximum possible array:         6014 MB (6.306e+09 bytes) *
Memory available for all arrays:        6014 MB (6.306e+09 bytes) *
Memory used by MATLAB:         2375 MB (2.490e+09 bytes)
Physical Memory (RAM):         8077 MB (8.469e+09 bytes)
```

- El comando "**whos**" se utiliza para mostrar información detallada acerca de las variables que están actualmente en memoria. Este comando mostrará detalles como el nombre de la variable, su tamaño, su tipo de datos y la cantidad de memoria que está utilizando. En la **Figura 68** se observa el resultado de haber implementado el comando "**whos**" en una operación de multiplicación de matrices aleatorias, el comando "whos" analiza los tamaños en bytes de cada una de las variables, así como el tipo de dato.

Figura 68

Comando whos y resultado de ejecución

```
A = rand(100);
B = rand(100);
C = A * B;
whos
```

```
Name        Size              Bytes  Class     Attributes

A         100x100             80000  double
B         100x100             80000  double
C         100x100             80000  double
```

Se utilizan estas herramientas para analizar la implementación de la criptografía en WSN, lo que ayuda a determinar si los nodos tienen suficiente cantidad de memoria. El análisis más detallado de los resultados se presenta en la sección **Prueba 5: Análisis del cálculo de procesamiento** donde se determina si el nodo cuenta con la cantidad de memoria suficiente, si no hay suficiente memoria disponible, el nodo puede contener errores. Es importante asegurarse de que se tenga suficiente memoria disponible para utilizar el algoritmo criptográfico de manera eficiente.

4 CAPITULO IV. Resultados

En esta sección se presentan los resultados obtenidos de las simulaciones de redes de sensores en el software OMNeT++, donde se ha implementado criptografía de curva elíptica (ECC) con los algoritmos ECDH y ElGamal Elíptico; se realiza pruebas de intercambio de información, la eficiencia del algoritmo de cifrado y descifrado, el consumo de energía y el tiempo de procesamiento. Además, se discuten las ventajas y desventajas de la implementación de criptografía de curva elíptica, así como recomendaciones para utilizar ECC en diferentes redes que dispongan limitado procesamiento.

Para comprobar la funcionalidad y la calidad del sistema se presenta la siguiente **Tabla 11** con la lista de pruebas a realizar, estas son pruebas de ***caja blanca*** y ***caja negra*** brindando enfoques diferentes. Las pruebas de caja blanca permiten examinar el código interno del algoritmo y verificar su funcionamiento, por otro lado, las pruebas de caja negra permiten evaluar el comportamiento externo del algoritmo, así como el funcionamiento de la red, sin conocer los detalles internos de su implementación.

Tabla 11*. Pruebas por realizar*

Prueba	***Tipo***	***Descripción***	***Resultado Esperado***	***Herramientas Utilizadas***
Prueba 1*: Funcionamiento de los algoritmos*	**Caja Blanca**	Prueba del funcionamiento de los algoritmos: ECDH y ELGAMAL EC, además de todas las funciones utilizadas en el sistema.	Verificar que los algoritmos de intercambio de clave y cifrado/descifrado operen correctamente	MATLAB
Prueba 2: *Funcionamiento de la red 802.15.4 y diferentes despliegues de WSN.*	**Caja Negra**	Evaluación del funcionamiento de la red 802.15.4 y la verificación de los distintos despliegues de redes de sensores inalámbricos.	Comprobar que la red y sus despliegues funcionen correctamente según las especificaciones	OMNET++
Prueba 3: *Elección de la curva elíptica para la WSN y demostración de los algoritmos ECDH y ElGamal Elíptico para el cifrado y*	**Caja Negra**	Evaluación de la elección de la curva elíptica y los algoritmos de cifrado/descifrado utilizados en la WSN.	Verificar que la elección de la curva y los algoritmos de cifrado/descifrado sean adecuados y funcionen correctamente	OMNET++

descifrado de los datos.				
Prueba 4: *Demostración del modelo de energía.*	**Caja Negra**	Verificación del modelo de energía implementado en la WSN.	Validar que el modelo de energía utilizado en la WSN funcione adecuadamente y proporcione resultados sobre	OMNET++
Prueba 5: *Análisis del cálculo de procesamiento.*	**Caja Blanca**	Evaluación del rendimiento del algoritmo de cifrado en términos de procesamiento.	Evaluar la eficiencia y precisión del cálculo de procesamiento en el algoritmo de cifrado, obtener la cantidad de operaciones que se necesita para ejecutar el algoritmo.	MATLAB

4.1 Prueba 1: Funcionamiento de los algoritmos

Las pruebas de funcionamiento aplicadas a los algoritmos son esenciales para garantizar que funcionen correctamente, sean eficientes y se ajusten a los requisitos del sistema. Las pruebas también son importantes para mejorar el rendimiento y la calidad del algoritmo. Las siguientes pruebas, obtienen los resultados de las funciones codificadas utilizando el software MATLAB.

4.1.1 Generación de puntos

Esta prueba tiene como objetivo validar que la obtención de puntos que conforman la curva elíptica sobre campos finitos primos con la función descrita en la sección **Generación de puntos de la curva elíptica** y el ***Algoritmo 2*** llamado "**puntosCurvaEliptica**". Además, valida si cumple con los requerimientos necesarios para utilizar dicha curva elíptica.

En la **Figura 69** se muestra el error si una curva no cumple con las condiciones necesarias, como que no sea singular y que el valor de $\boldsymbol{p}$ sea un numero primo mayor que 3.

Figura 69

Mensaje de error en la generación de puntos

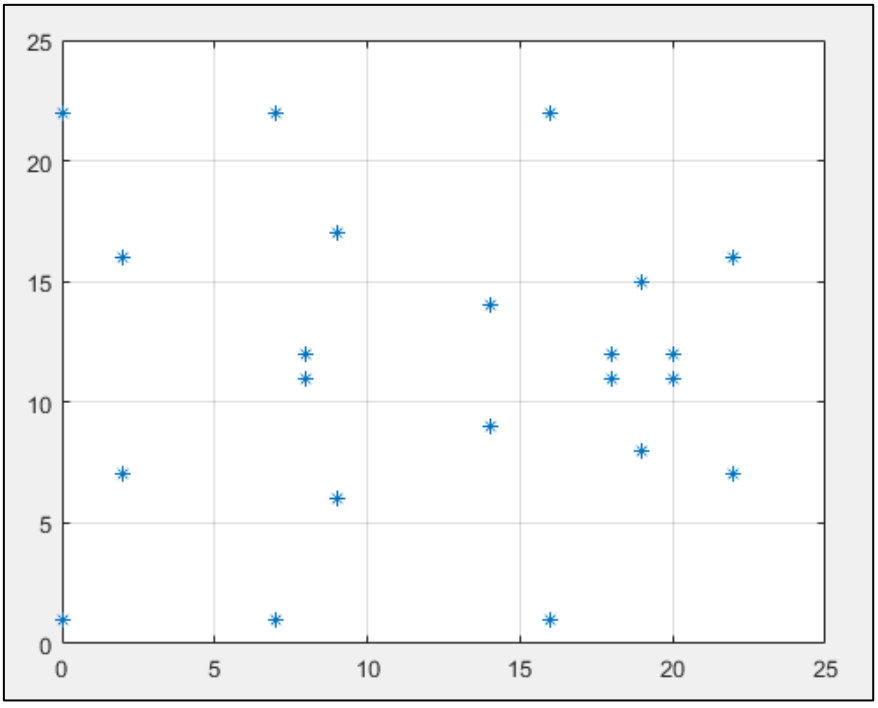
```
>> puntosCurvaElipticaModp(0, 0, 103)
Es una curva singular, Intenta con otra curva
>> puntosCurvaElipticaModp(-3, 1, 8)
El valor de p debe ser un numero primo mayor que 3
```

En la **Figura 70** se muestra los puntos generados de la curva elíptica $\boldsymbol{y^2 = x^3 - 3x + 1}$ $(\boldsymbol{mod\ 23})$, además del grafico correspondiente, nótese como el valor de $\boldsymbol{23}$ limita el campo en $\boldsymbol{x}$ e $\boldsymbol{y}$ del plano. Dado que se utiliza el módulo $\boldsymbol{23}$, los valores de $\boldsymbol{x}$ e $\boldsymbol{y}$ están restringidos a los números enteros entre $\boldsymbol{0}$ y $\boldsymbol{22}$. Esto significa que solo se consideran los valores de $\boldsymbol{x}$ e $\boldsymbol{y}$ en ese rango, y cualquier valor fuera de ese rango se "envuelve" dentro del mismo utilizando aritmética modular.

Figura 70

Generación de puntos para $\boldsymbol{Z_{23}}$

0	1
0	22
2	7
2	16
7	1
7	22
8	11
8	12
9	6
9	17
14	9
14	14
16	1
16	22
18	11
18	12
19	8
19	15
20	11
20	12
22	7
22	16
Inf	Inf

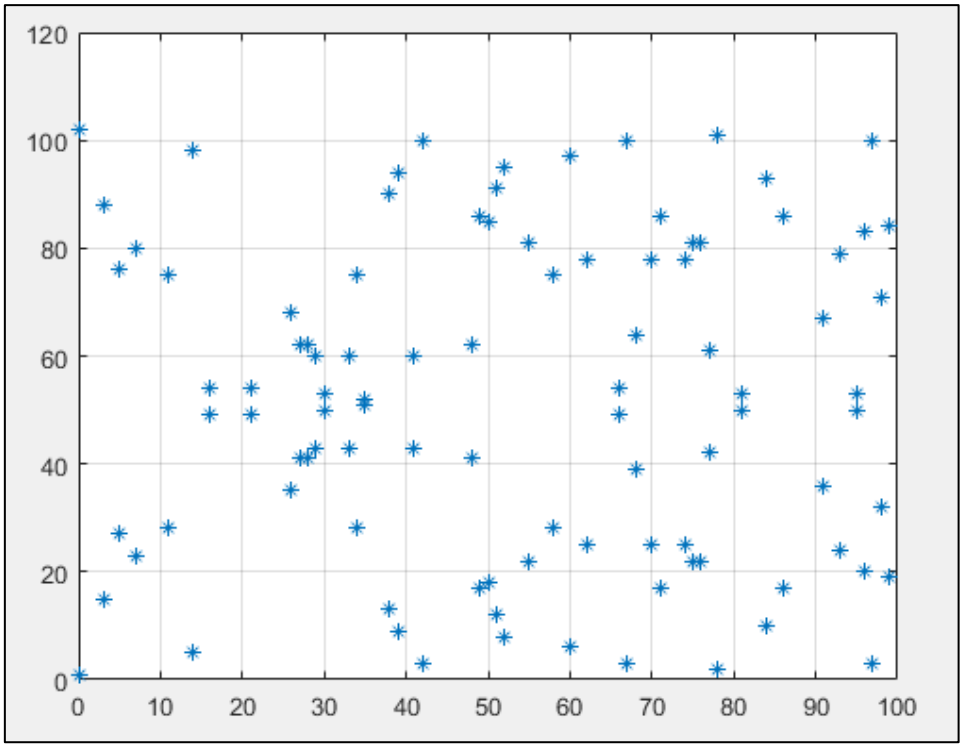


En la **Figura 71** se aprecia los puntos generados de la curva elíptica $\boldsymbol{y^2 = x^3 - 3x + 1}$ $(\boldsymbol{mod\ 103})$ junto con su correspondiente gráfico.

Figura 71

Generación de puntos para $\boldsymbol{Z_{103}}$

0	1
0	102
3	15
3	88
5	27
5	76
7	23
7	80
11	28
11	75
14	5
14	98
16	49
16	54
21	49
21	54
26	35
26	68
27	41
27	62
28	41
28	62
29	43
29	60
30	50
30	53
...	

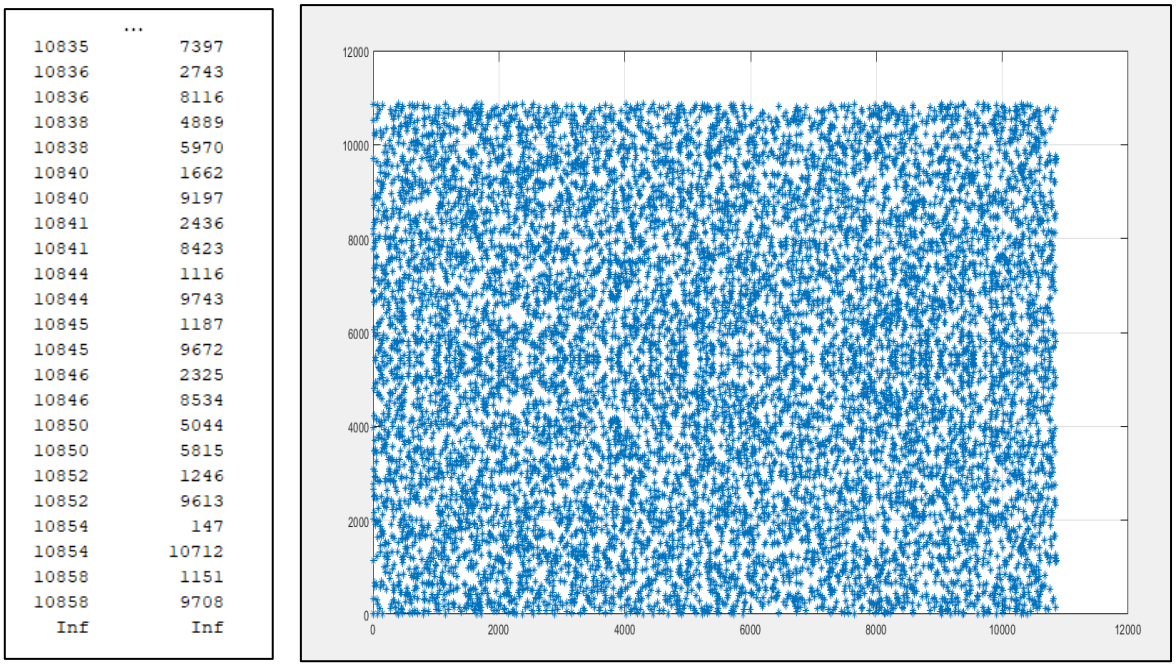


En la **Figura 72** se aprecia los puntos generados de la curva elíptica $\boldsymbol{y^2 = x^3 - 3x + 1}$ $\boldsymbol{(mod\ 10859)}$ junto con su correspondiente gráfico. La generación del cálculo de los puntos para $\boldsymbol{p = 10859}$ y la generación de la imagen utilizan todo el procesamiento del sistema y el tiempo para generar esta imagen es aproximadamente de $\boldsymbol{10min}$ utilizando un computador de gama media. El sistema registró y cronometró el tiempo empleado en el proceso. Es importante destacar que a medida que aumenta el tamaño del campo, es decir, en campos mayores a $\boldsymbol{10859}$, el tiempo necesario para generar los puntos y visualizarlos experimenta un crecimiento exponencial.

La cantidad de tiempo se debe a la complejidad computacional asociada con el cálculo de los puntos de la curva elíptica en un campo finito grande, así como los recursos adicionales necesarios para la visualización gráfica. El crecimiento exponencial del tiempo en campos mayores a $\boldsymbol{10859}$ es una consecuencia directa del aumento en la complejidad de los cálculos en campos finitos más grandes.

Figura 72

Generación de puntos para $\boldsymbol{Z_{10859}}$



La generación de puntos depende de la curva elíptica y del tamaño del campo, para la implementación de criptografía se debe ajustar el campo $\boldsymbol{p}$ y curva ($\boldsymbol{a}$, $\boldsymbol{b}$) para los cuales no deben superar el procesamiento del sistema.

4.1.2 Suma de puntos

En esta prueba se verifica el funcionamiento del ***Algoritmo 3*** el cual permite realizar la suma de puntos llamado "**sumaPuntosCurvaEliptica"**, para eso se utiliza los puntos del campo $\boldsymbol{Z_{23}}$ comprobando el resultado matemáticamente con el devuelto por la función, en 3 diferentes casos como son:

4.1.2.1 Suma de dos puntos diferentes

Se quiere sumar los puntos $\boldsymbol{P = (20, 11)}$ y $\boldsymbol{Q = (7, 1)}$ que pertenecen a la curva elíptica $\boldsymbol{y^2 = x^3 - 3x + 1}$ sobre $\boldsymbol{Z_{23}}$ para encontrar el punto $\boldsymbol{R}$. Se identifica el caso, donde $\boldsymbol{P = (x_1, y_1)}$, el valor de $\boldsymbol{x_1 = 20}$ y en $\boldsymbol{Q = (x_2, y_2)}$ el valor de $\boldsymbol{x_2 = 7}$, por lo que se trata de una suma de dos puntos distintos, $\boldsymbol{P \neq Q}$. Se calcula $\boldsymbol{m}$ según la **(Ec. 8)** y se opera utilizando aritmética modular $\boldsymbol{A.mod.}$

$$\boldsymbol{m = \frac{y_2 - y_1}{x_2 - x_1} (mod\ p)}$$

$$= \frac{1 - 11}{7 - 20} (mod\ 23)$$

$$= \frac{10}{13} (mod\ 23)\ \{\boldsymbol{division\ modular}\}$$

$$= 10 \times 13^{-1} (mod\ 23)$$

$$= 10 \times 16 (mod\ 23)$$

$$= 160 (mod\ 23)$$

$$= 22$$

Con el valor de $\boldsymbol{m}$ se calcula los valores de $\boldsymbol{R = (x_3\ , y_3)}$ también descrito en la **(Ec. 8)**.

$$\boldsymbol{x_3 = m^2 - x_1 - x_2\ (mod\ p)}$$

$$= (22)^2 - 20 - 7\ (mod\ 23)$$

$$= \ 484 - 20 - 7\ (mod\ 23)$$

$$= 457\ (mod\ 23)$$

$$= 20$$

$$\boldsymbol{y_3 = m(x_1 - x_3) - y_1\ (mod\ p)}$$

$$= 22(20 - 20) - 11\ (mod\ 23)$$

$$= -11\ (mod\ 23)$$

$$= 12$$

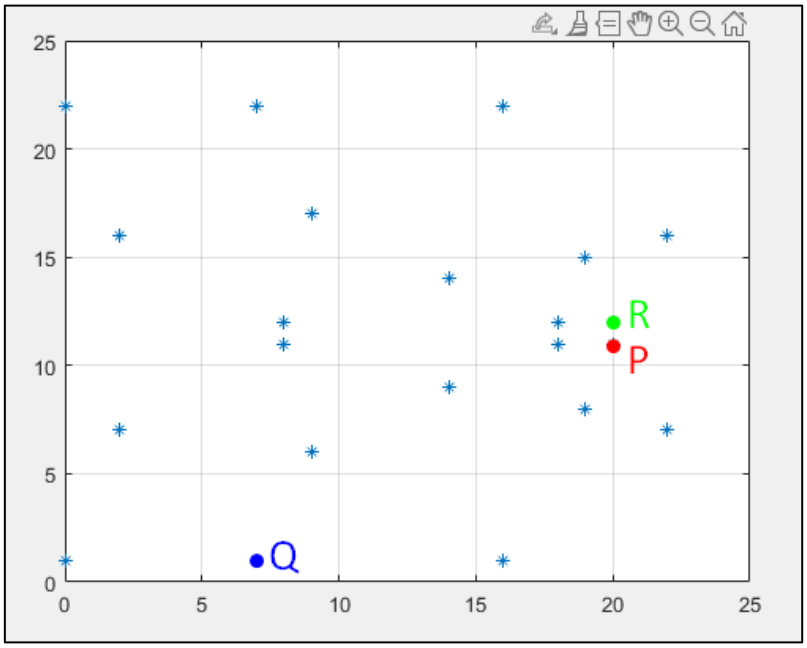
Obteniendo el resultado de $\boldsymbol{R = (x_3, y_3) = (20,\ 12)}$.

En la **Figura 73** se verifica el resultado del algoritmo el cual devuelve el mismo resultado que el obtenido mediante cálculos.

Figura 73

Suma de los puntos P y Q para obtener el resultado en el punto R en $\boldsymbol{Z_{23}}$

```
>> sumaPuntosCurvaElipticaModp([20 11],[7 1],-3,1,23)

ans =

    20    12
```



4.1.2.2 Suma de dos puntos iguales (Duplicación de un punto)

Se quiere sumar los puntos $\boldsymbol{P = (20, 11)}$ y $\boldsymbol{Q = (20, 11)}$ que pertenecen a la curva elíptica $\boldsymbol{y^2 = x^3 - 3x + 1}$ sobre $\boldsymbol{Z_{23}}$ para encontrar el punto $\boldsymbol{R}$. Se identifica el caso, donde $\boldsymbol{P = (x_1, y_1) = Q = (x_2, y_2)}$, por lo que se trata de una suma de dos puntos iguales, también llamado duplicación de un punto, $\boldsymbol{P = Q}$. Se calcula $\boldsymbol{m}$ según la **(Ec. 10)** y se opera utilizando aritmética modular $\boldsymbol{A.mod.}$

$$\boldsymbol{m = \frac{3x_1^2 + a}{2y_1} \, (mod\ p)}$$

$$= \frac{3(20)^2 + (-3)}{2(11)} \, (mod\ 23)$$

$$= \frac{1197}{22} (mod\ 23)\ \{\boldsymbol{division\ modular}\}$$

$$= 1197 \times\ 22^{-1} (mod\ 23)$$

$$= 1197 \times\ 22 (mod\ 23)$$

$$= 26334\ (mod\ 23)$$

$$= 22$$

Con el valor de $\boldsymbol{m}$ se calcula los valores de $\boldsymbol{R} = (\boldsymbol{x_3}, \boldsymbol{y_3})$, representándose como $\boldsymbol{R} = \boldsymbol{2P} = \boldsymbol{2Q}$, para ello se utiliza la **(Ec. 10)**.

$$\boldsymbol{x_3 = m^2 - 2x_1\ (mod\ p)}$$

$$= (22)^2 - 2(20) (mod\ 23)$$

$$= 444\ (mod\ 23)$$

$$= 7$$

$$\boldsymbol{y_3 = m(x_1 - x_3) - y_1\ (mod\ p)}$$

$$= 22(22 - 7) - 11\ (mod\ p)$$

$$= 319 (mod\ p)$$

$$= 22$$

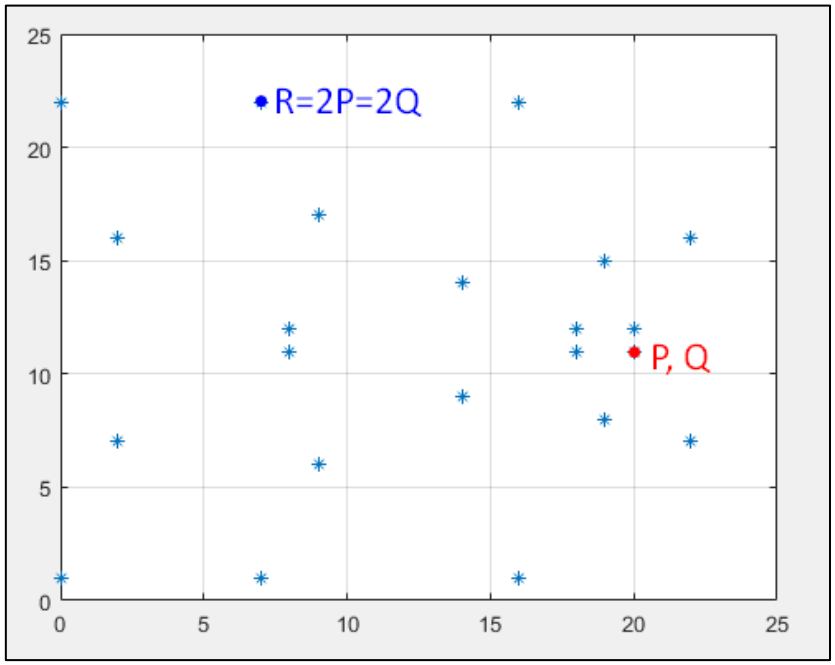
Obteniendo el resultado de $\boldsymbol{R} = (\boldsymbol{x_3}, \boldsymbol{y_3}) = (\boldsymbol{7},\ \boldsymbol{22})$.

En la **Figura 74** se verifica el resultado del algoritmo el cual devuelve el mismo resultado que el obtenido mediante cálculos.

Figura 74

Duplicación de puntos P = Q para obtener el resultado en el punto R=2P=2Q en $\boldsymbol{Z_{23}}$

```
>> sumaPuntosCurvaElipticaModp([20 11],[20 11],-3,1,23)

ans =

     7    22
```



4.1.2.3 Suma al infinito (sumar dos puntos opuestos)

Se quiere sumar los puntos $\boldsymbol{P = (9, 6)}$ y $\boldsymbol{Q = (9, 17)}$ que pertenecen a la curva elíptica $\boldsymbol{y^2 = x^3 - 3x + 1}$ sobre $\boldsymbol{Z_{23}}$ para encontrar el punto $\boldsymbol{R}$. Se identifica el caso, donde $\boldsymbol{P \neq Q}$, por lo que se trata de una suma de dos puntos distintos, además $\boldsymbol{x_1 = x_2}$ por lo que se calcula $\boldsymbol{m}$ según la **(Ec. 9)** y se opera utilizando aritmética modular $\boldsymbol{A.mod.}$

$$\boldsymbol{m = \frac{y_2 - y_1}{x_2 - x_1} (mod\ p)}$$

$$= \frac{17 - 6}{9 - 9} (mod\ 23)$$

$$= \frac{11}{0} (mod\ 23)$$

$$= inf\,(mod\ 23)$$

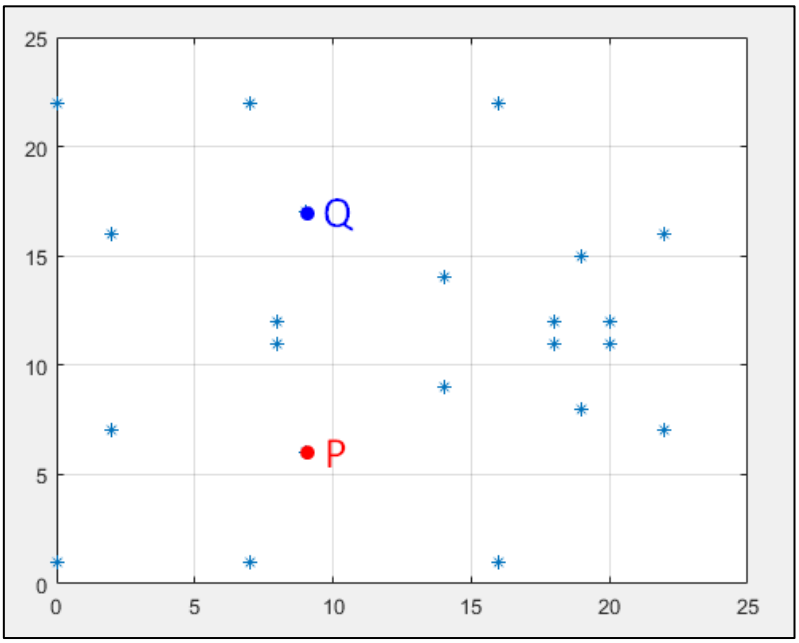
$$= inf$$

Por lo que la pendiente de la curva tiende al punto al infinito. Entonces tenemos que $\boldsymbol{R} = (\boldsymbol{x_3}, \boldsymbol{y_3}) = \boldsymbol{O}$, el punto al infinito. Sumar dos puntos opuestos genera el elemento neutro llamado **punto al infinito** ($\boldsymbol{O}$), En la **Figura 75** se verifica el resultado del funcionamiento del algoritmo el cual devuelve el mismo resultado que el obtenido mediante cálculos.

Figura 75

Suma de dos puntos opuestos que dan como resultado el punto al infinito $\boldsymbol{O}$ en $\boldsymbol{Z_{23}}$

```
>> sumaPuntosCurvaElipticaModp([9 6],[9 17],-3,1,23)

ans =

   Inf   Inf
```



4.1.3 Multiplicación de puntos

En esta prueba se verifica el funcionamiento del **Algoritmo 5** el cual permite generar la multiplicación de un escalar por un punto, llamado "**multEscalarCurvaEliptica"**, para eso se comprueba el resultado de todos los puntos en $\boldsymbol{Z_{23}}$.

En la **Tabla 12** se muestran todos los puntos computados a partir de la multiplicación de un escalar por el punto $\boldsymbol{P}$. El orden de los puntos depende del punto de inicio, es decir si cambiamos el punto $\boldsymbol{P} = (\mathbf{2}, \mathbf{7})$ por otro de la curva, el orden será diferente.

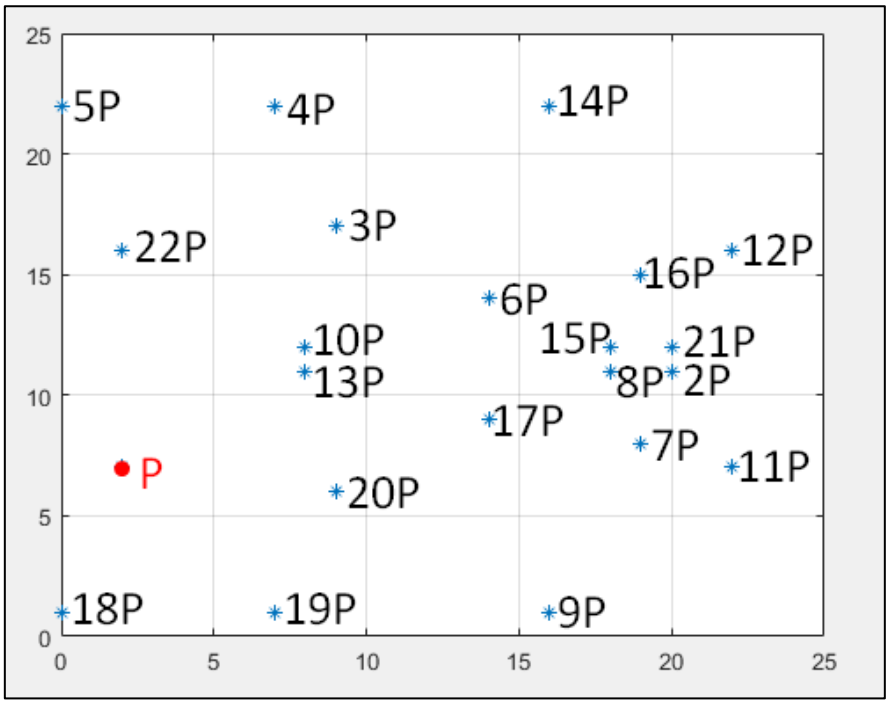
Tabla 12. *Puntos computados en la curva $y^2 = x^3 - 3x + 1$ en Z_{23} usando el algoritmo llamado "multEscalarCurvaEliptica"*

	x	y
P	*2*	*7*
2P	*20*	*11*
3P	*9*	*17*
4P	*7*	*22*
5P	*0*	*22*
6P	*14*	*14*
7P	*19*	*8*
8P	*18*	*11*
9P	*16*	*1*
10P	*8*	*12*
11P	*22*	*7*
12P	*22*	*16*
13P	*8*	*11*
14P	*16*	*22*
15P	*18*	*12*
16P	*19*	*15*
17P	*14*	*9*
18P	*0*	*1*
19P	*7*	*1*
20P	*9*	*6*
21P	*20*	*12*
22P	*2*	*16*
23P	***O***	***O***

En la **Figura 76** se muestra de manera grafica todos los múltiplos del punto $\boldsymbol{P} = (\mathbf{2}, \mathbf{7})$, hasta llegar a $\mathbf{22}\boldsymbol{P}$, también se añade el punto al infinito $\mathbf{23}\boldsymbol{P} = \boldsymbol{O}$ el cual se puede representar como cualquier punto que no pertenezca a la curva en $\boldsymbol{Z_{23}}$.

Figura 76

Multiplicación de un escalar por $\boldsymbol{P}$ en $\boldsymbol{Z_{23}}$



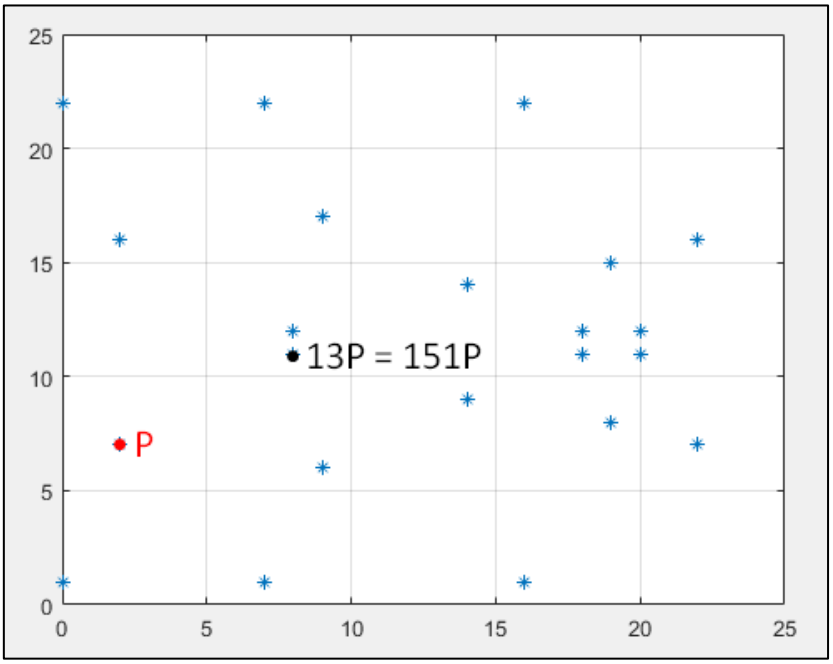
También se puede calcular la multiplicación de un escalar mayor al campo $\boldsymbol{p}$ con el punto $\boldsymbol{P}$, siendo posible obtener un punto en la curva. Por ejemplo: Se calcula el producto del escalar $\boldsymbol{k = 151}$ por el punto $\boldsymbol{P = (2, 7)}$ de la curva elíptica $\boldsymbol{y^2 = x^3 - 3x + 1\ (mod\ 23)}$.

En la **Figura 77** se muestra el resultado de obtener $\boldsymbol{151P}$, concluyendo así que $\boldsymbol{13P = (8, 11)}$ y $\boldsymbol{151P = (8, 11)}$ son el mismo punto.

Figura 77

Multiplicación del escalar $\boldsymbol{151}$ *por* $\boldsymbol{P = (2, 7)}$ *en* $\boldsymbol{Z_{23}}$

```
>> multEscalarCurvaElipticaModp([2 7],151,-3,1,23)

ans =

     8    11
```



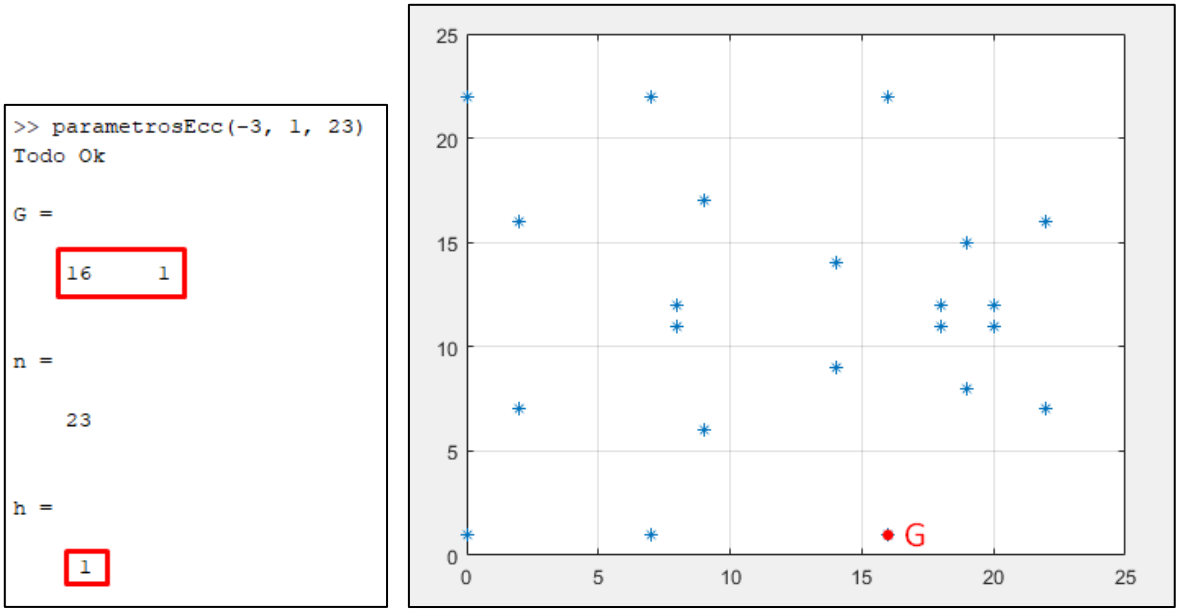
4.1.4 Parámetros ECC

En esta prueba se verifica el **Algoritmo 8** llamado "**parametrosEcc**", disponible en la sección **Parámetros de dominio ECC** el cual permite encontrar el orden de subgrupo, el orden del punto y el valor del cofactor. Este algoritmo nos ayuda a determinar si existe un *punto generador* ideal, con un *cofactor* entero y no menor que **4** para que se implemente los algoritmos de ECDH y ElGamal Eliptico.

La **Figura 78** muestra el resultado del algoritmo aplicado en la curva $\boldsymbol{y^2 = x^3 - 3x + 1}$ $(\boldsymbol{mod\ 23})$ sobre $\boldsymbol{Z_{23}}$, se pretende encontrar el punto generador del subgrupo, el orden del punto y el cofactor. El resultado del cofactor $\boldsymbol{h = 1}$ indica que existen $\boldsymbol{23}$ puntos en el grupo y el orden del punto es $\boldsymbol{23}$, obteniendo así $h = \frac{Puntos\ del\ grupo}{orden\ del\ punto}$.

Figura 78

Parámetros de dominio de la curva elíptica en $\boldsymbol{Z_{23}}$ obteniendo el punto generador $\boldsymbol{G}$ el cual permite la implementación de ECDH y El GamalEliptico de forma segura.



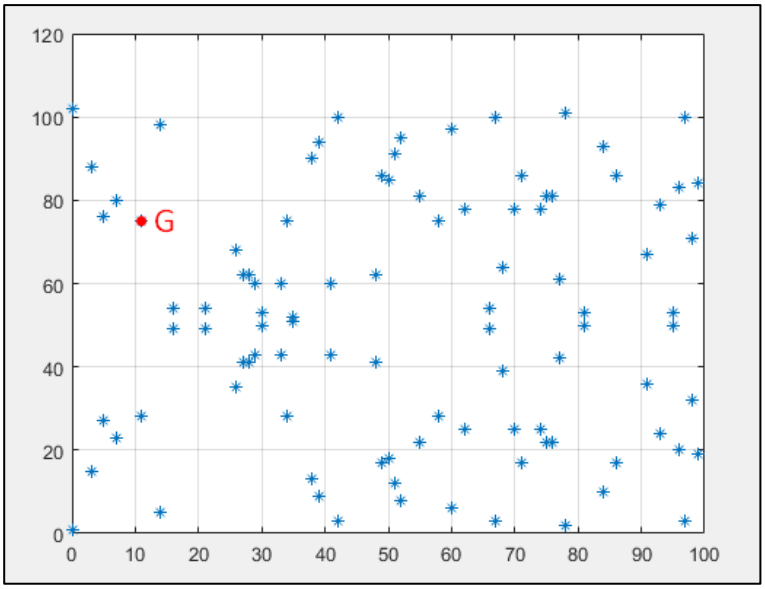
Para otro campo u otra curva elíptica el punto generador puede cambiar; ***Pueden existir muchos puntos que cumplan esta característica y también no existir un punto generador que tenga un cofactor*** $\boldsymbol{< 4}$; Es importante buscar curvas y campos que generen estos parámetros para ser implementados en otros algoritmos.

La **Figura 79** muestra el resultado del algoritmo aplicado en la curva $\boldsymbol{y^2 = x^3 - 3x + 1}$ $(\boldsymbol{mod\ 103})$ sobre $\boldsymbol{Z_{103}}$.

Figura 79

Parámetros de dominio de la curva elíptica en Z_{103} obteniendo $G = (11,75)$

```
>> parametrosEcc(-3, 1, 103)
Todo Ok

G =

    11    75


n =

    99


h =

     1
```



4.1.5 Intercambio de claves usando ECDH

En esta prueba se verifica el funcionamiento del **Algoritmo 9** ECDH para la generación de claves, el intercambio de estas y la obtención de una clave secreta compartida, el algoritmo se llama "**diffieHelmanEcc"**, y se encuentra descrito en el apartado **Intercambio de claves ECDH**.

Para verificar el funcionamiento se ingresa los valores de la curva ($\boldsymbol{a}$ y $\boldsymbol{b}$), los valores del campo ($\boldsymbol{p}$) y el punto generado ($\boldsymbol{G}$) de la curva elíptica $\boldsymbol{y^2 = x^3 - 3x + 1\ (mod\ 23}$) sobre $\boldsymbol{Z_{23}}$ con $\boldsymbol{G = (0, 22)}$.

Se tiene dos partes $\boldsymbol{A}$ y $\boldsymbol{B}$, las cuales quieren establecer una clave secreta compartida para ello, cada uno genera su par de claves:

- $\boldsymbol{A}$ elige la clave privada $\boldsymbol{d_A = 3}$ y parte de $\boldsymbol{G = (0, 22)}$ para obtener su clave publica $\boldsymbol{d_A G = 3G = (18, 12)}$.
- $\boldsymbol{B}$ elige la clave privada $\boldsymbol{d_B = 19}$ y parte de $\boldsymbol{G = (0, 22)}$ para obtener su clave publica $\boldsymbol{d_A G = 19G = (9, 17)}$.

- Ambas partes intercambian las claves $\boldsymbol{A}$ envía $\boldsymbol{(18,12)_A}$ hacia $\boldsymbol{B}$; y $\boldsymbol{B}$ envía $\boldsymbol{(9,17)_B}$ hacia $\boldsymbol{A}$
- $\boldsymbol{A}$ calcula $\boldsymbol{S_A = d_A(9,17)_B}$

$$= \boldsymbol{d_A d_B G}$$
$$= \boldsymbol{3 * 19 (mod\ 23) G}$$
$$= \boldsymbol{57 (mod\ 23) G}$$
$$= \boldsymbol{11G = (16,1)}$$

- $\boldsymbol{B}$ calcula $\boldsymbol{S_B = d_B(18,12)_A}$

$$= \boldsymbol{d_B d_A G}$$
$$= \boldsymbol{19 * 3 (mod\ 23) G}$$
$$= \boldsymbol{57 (mod\ 23) G}$$
$$= \boldsymbol{11G = (16,1)}$$

- Obteniendo así la clave secreta compartida en $\boldsymbol{11G = (16,1)}$

La **Tabla 13** muestra el orden de los puntos para una mejor comprensión, donde se parte del punto generador $\boldsymbol{G}$, se obtiene todos los múltiplos de este hasta llegar al punto al infinito.

Tabla 13. *Puntos computados en la curva $y^2 = x^3 - 3x + 1$ en Z_{23} usando el punto generador $\boldsymbol{G = (0,22)}$*

	X	Y
G	*0*	*22*
2G	*8*	*12*
3G	*18*	*12*
4G	*9*	*6*
5G	*20*	*11*
6G	*19*	*8*
7G	*22*	*6*

8G	*14*	*9*
9G	*2*	*16*
10G	*7*	*22*
11G	*16*	*1*
12G	*16*	*22*
13G	*7*	*1*
14G	*2*	*7*
15G	*14*	*14*

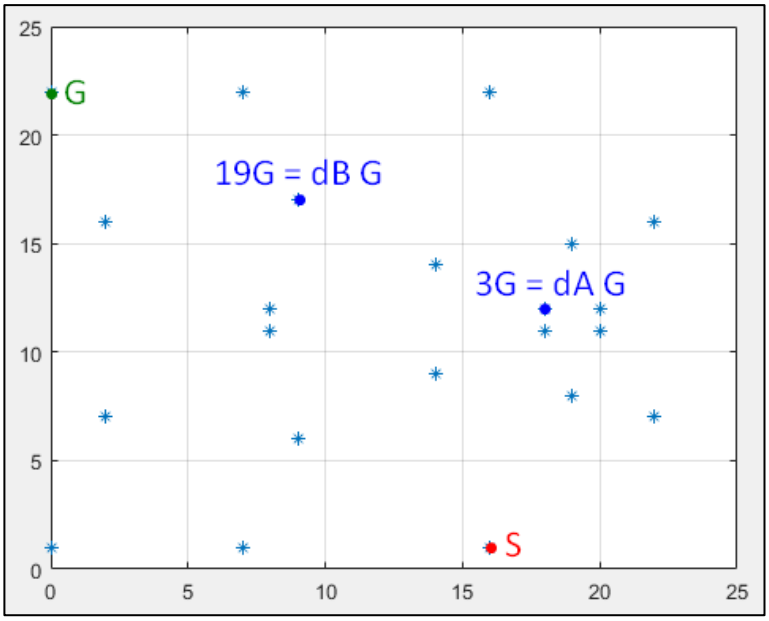
16G	*22*	*7*
17G	*19*	*15*
18G	*20*	*12*
19G	*9*	*17*
20G	*18*	*11*
21G	*8*	*11*
22G	*0*	*1*
23G	***O***	***O***

En la **Figura 80** se muestra de forma gráfica la ubicación de cada uno de los puntos (claves públicas y clave secreta compartida) que intervienen en este proceso, así como la demostración del funcionamiento del algoritmo en la curva elíptica $\boldsymbol{y^2 = x^3 - 3x + 1\ (mod\ 23)}$ sobre $\boldsymbol{Z_{23}}$ con $\boldsymbol{G = (0,22)}$.

Figura 80

Intercambio de claves en ECDH, en el campo Z_{23} y con $G = (0,22)$

```
Numero entero a = -3
Numero entero b = 1
Numero primo p = 23
Todo Ok

G =

     0    22


n =

    23


h =

     1

La clave privada de A es: [3]
La clave publica de A es: [18, 12]
La clave privada de B es: [19]
La clave publica de B es: [9, 17]
La clave compartida generada por A es: [16, 1]
La clave compartida generada por B es: [16, 1]
```



4.1.1 Cifrado de datos utilizando ElGamal Eliptico

Antes de implementar el cifrado con ElGamal Eliptico, se necesita representar el texto a cifrar en puntos de la curva elíptica; Aunque en la vida real se utilice un solo punto para cifrar todo el mensaje, este se puede dividir y representar varios puntos, aquí se implementa un cifrado de flujo.

Se necesita utiliza una curva elíptica con un campo lo suficientemente grande para representar cada carácter en puntos de la curva. La curva elíptica que cumple con las condiciones mínimas para representar todos los caracteres del código ASCII encontrada por el autor es $\boldsymbol{y^2 = x^3 - 3x + 1\ (mod\ 3851)}$ sobre $\boldsymbol{Z_{3851}}$.

En la **Figura 81** se observa la representación de diversos caracteres en la curva $\boldsymbol{y^2 = x^3 - 3x + 1\ (mod\ 3851)}$, esto se logra utilizando el **Algoritmo 10** llamado "**charToPuntoEcc**" el cual devuelve la representación de cada carácter como un punto en la curva. Para el ejemplo se utiliza la cadena de texto "**D@ToS 123**" la cual se quiere representar cada uno de sus caracteres en puntos de la curva elíptica.

Figura 81

*Representación de la cadena de texto "**D@ToS 123**" en la curva* $y^2 = x^3 - 3x + 1 \ (mod\ 3851)$

A continuación, se muestra la **Tabla 13** con todos los caracteres y su correspondiente punto en la curva.

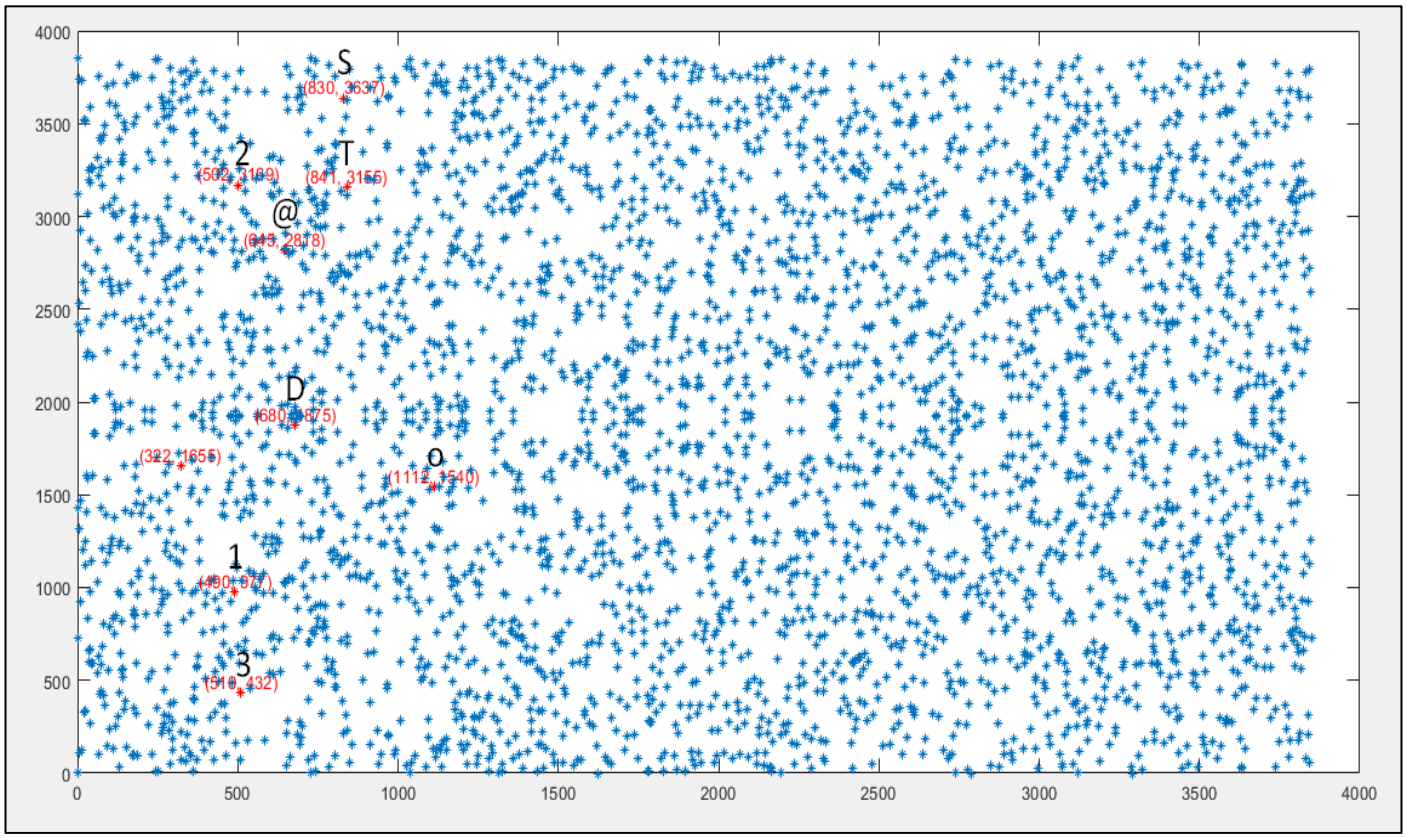
Tabla 14. *Representación del cada uno de los caracteres en puntos*

Carácter	ASCII	x	y
D	*68*	*680*	*1875*
@	*64*	*645*	*2818*
T	*84*	*841*	*3155*
o	*111*	*1112*	*1540*
S	*83*	*830*	*3637*
	32	*322*	*1655*
1	*49*	*490*	*977*
2	*50*	*502*	*3169*
3	*51*	*510*	*432*

En la **Figura 82** se observa cómo se representan estos puntos dentro de la curva, cada punto puede representar un solo carácter. Existiendo así suficientes puntos para representar cualquier tipo de carácter.

Figura 82

Gráfico de la curva $y^2 = x^3 - 3x + 1 \ (mod\ 3851)$*, en esta se obsera la informacion representada en puntos*



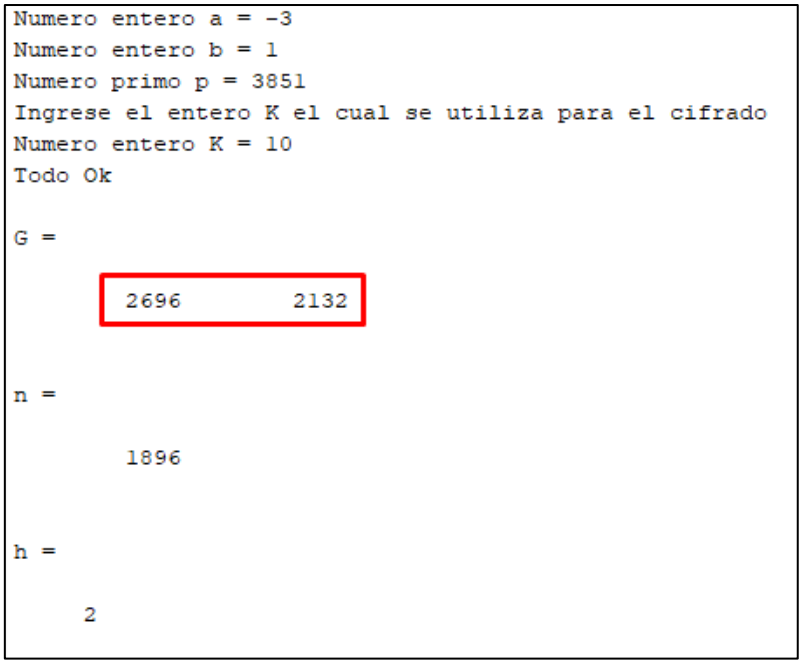
Una vez convertidos todos los caracteres de la información a cifrar "**D@ToS 123**" en puntos de la curva elíptica **Tabla 14** se procede al cifrado usando ElGamal Elíptico, para ello antes del proceso anterior se debe tener una clave secreta en común, implementada en el

Intercambio de claves usando ECDH, a continuación, se realiza todo el proceso de cifrado de los datos, utilizando el software MATLAB.

Primero se establece el punto generador de la curva elíptica $\boldsymbol{y^2 = x^3 - 3x + 1}$ $(\boldsymbol{mod\ 3851})$, con el campo $\boldsymbol{Z_{3851}}$**.** En la **Figura 83** se muestra la implementación de los **Parámetros ECC** el cual nos devuelve el punto generador con un cofactor $\boldsymbol{h < 4}$

Figura 83

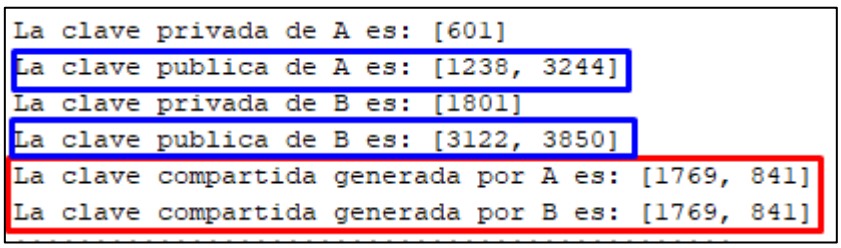
Obtención del punto generador G=(2696, 2132)



Se obtiene el par de claves, además de la clave compartida, en la **Figura 84** se muestra la generación de las claves a partir del cálculo multiplicación de puntos, obteniendo como clave compartida $\boldsymbol{S = (1769, 841)}$.

Figura 84

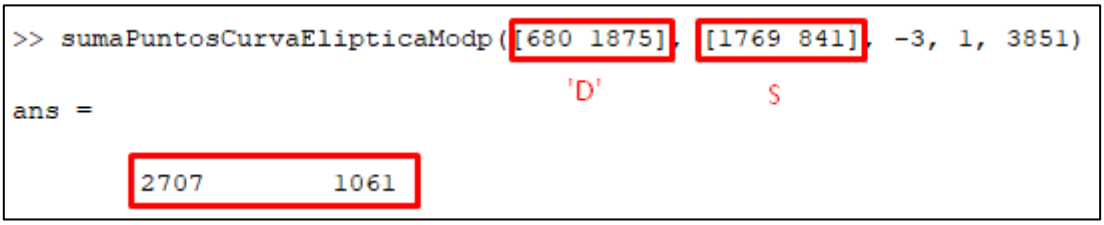
Generación de claves e intercambio para cada una de las partes



Con la clave compartida se procede al proceso de cifrado de datos, la clave compartida será el punto con el cual se tienen que sumar cada uno de los caracteres, para representarlos en un nuevo conjunto de puntos. La **Figura 85** muestra el resultado para el primer punto que representa al carácter ***'D'*** sumado con la clave secreta, esto se realiza para cada uno de los puntos del mensaje a cifrar.

Figura 85

Suma de un punto que representa un carácter con el punto que representa la clave secreta compartida.



En la **Tabla 15** se muestra el resultado de la operación suma de puntos entre el punto que representa el carácter con la calve compartida generada, obteniendo así otro punto dentro de la curva elíptica.

Tabla 15. *Cifrado de cada uno de los caracteres utilizando la clave secreta compartida*

Carácter	Datos x	Datos y	Cifrado x	Cifrado y	Carácter Cifrado
D	*680*	*1875*	*2707*	*1061*	*Ď*
@	*645*	*2818*	*3610*	*756*	*ũ*
T	*841*	*3155*	*721*	*3783*	*H*
o	*1112*	*1540*	*2397*	*2711*	*ï*
S	*830*	*3637*	*569*	*2808*	*8*
	322	*1655*	*2435*	*2541*	*ó*
1	*490*	*977*	*3463*	*3705*	*Ś*
2	*502*	*3169*	*825*	*391*	*R*
3	*510*	*432*	*2489*	*1626*	*ø*

En la **Figura 86** se puede observar cómo se convierte el nuevo conjunto de puntos a caracteres obteniendo el mensaje cifrado "***ĎũHï8óŚRø***";

Figura 86

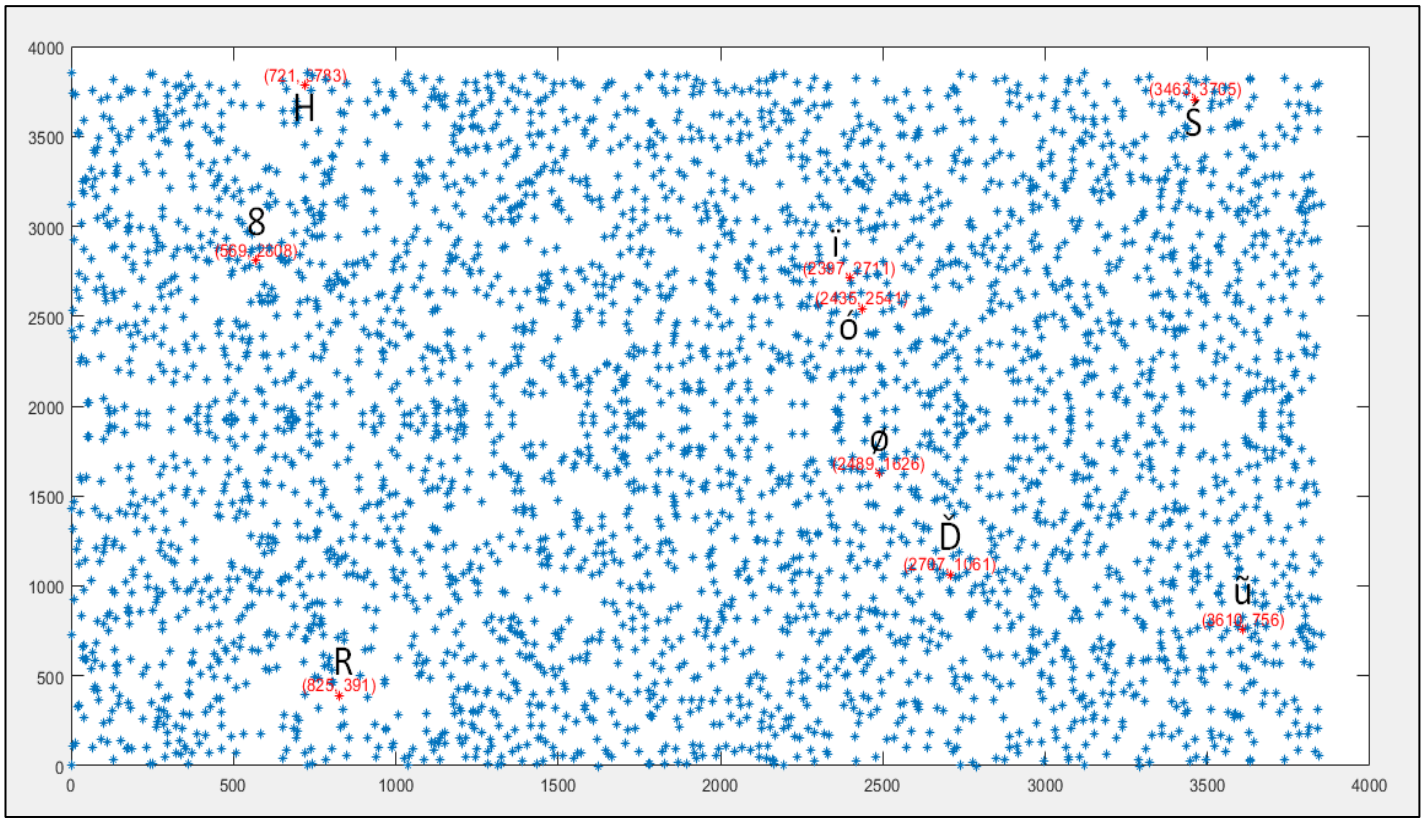
Representación de los puntos cifrados a texto

```
>> puntostotext([2707    1061
3610     756
721  3783
2397     2711
569  2808
2435     2541
3463     3705
825  391
2489     1626
],-3,1,3851,10)

ans =

    'ĎũHï8óŚRø'
```

También, en la **Figura 87** se puede observar los nuevos puntos que representan el mensaje cifrado en la curva elíptica $\boldsymbol{y^2 = x^3 - 3x + 1}$ $\boldsymbol{(mod\ 3851)}$.

Figura 87

Cifrado de datos en forma de punto de la curva $\boldsymbol{y^2 = x^3 - 3x + 1}$ $\boldsymbol{(mod\ 3851)}$.



Para comprobar que el algoritmo está funcionando se utiliza la función de cifrado, la cual devuelve una matriz de puntos, los cuales condicen con la **Tabla 15**. En la **Figura 88** se demuestra el funcionamiento del algoritmo, el cual requiere los parámetros de la curva $\boldsymbol{y^2 = x^3 - 3x + 1\ (mod\ 3851)}$, el tamaño del campo $\boldsymbol{Z_{3851}}$, la clave secreta compartida $\boldsymbol{S = (1769, 841)}$ y el mensaje a cifrar "**D@ToS 123**".

Figura 88

Comprobación del algoritmo para el cifrado de datos

```
>> cifradorElGamalEcc2(-3, 1, 3851, 10, [1769 841], 'D@ToS 123')

ans =

        2707        1061
        3610         756
         721        3783
        2397        2711
         569        2808
        2435        2541
        3463        3705
         825         391
        2489        1626
```

4.1.2 Descifrado de datos utilizando ElGamal Elíptico

En el proceso de descifrado también interviene la clave compartida, pero en este caso se desea encontrar el punto inverso de la clave compartida, el cual sumado a los puntos cifrados nos da como resultado los puntos del mensaje original. Para obtener el punto inverso de la clave compartida se realiza:

- Se parte del punto "clave secreta compartida" $\boldsymbol{S = (1769, 841)}$
- El inverso de este punto $\boldsymbol{-S}$ será el que sumando con el punto $\boldsymbol{S}$ de como resultado el elemento neutro, para ello se sabe que el valor de la coordenada $\boldsymbol{x}$ no cambia.
- Se encuentra el valor inverso de $\boldsymbol{y}$, con la siguiente formula $\boldsymbol{-S = (x, -y\ (mod\ p))}$, donde la coordenada que se quiere encontrar es $\boldsymbol{-y\ (mod\ p)}$, para ello se reemplazan

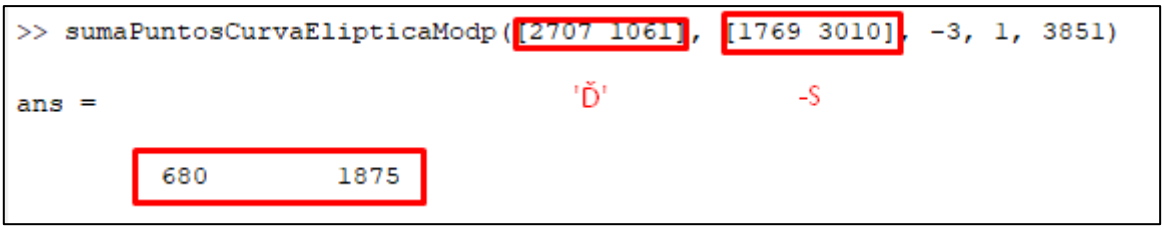
los valores $-\mathbf{841}$ $(\boldsymbol{mod}\ \mathbf{3851})$ y se aplica aritmética modular obteniendo $-\boldsymbol{y} = \mathbf{3010}$.

- El punto inverso es $-\boldsymbol{S} = (\mathbf{1769}, \mathbf{3010})$.

La **Figura 89** muestra el resultado del descifrado para el primer punto, donde se realiza la operación suma entre el punto que representa al carácter 'Ď' y el punto inverso de la clave secreta compartida $-\boldsymbol{S}$.

Figura 89

Suma de dos puntos, el primero el punto que representa al mensaje cifrado y el segundo el punto de la clave secreta.



En la siguiente **Tabla 16** se muestra todos los puntos descifrados con el objetivo de obtener el mensaje original.

Tabla 16. *Suma de los puntos cifrados con el inverso de la calve secreta compartida* $-\boldsymbol{s} = (\mathbf{1769}, \mathbf{3010})$

Carácter Cifrado	Cifrado x	Cifrado y	Datos x	Datos y	Carácter Descifrado
Ď	*2707*	*1061*	*680*	*1875*	*D*
Ũ	*3610*	*756*	*645*	*2818*	*@*
H	*721*	*3783*	*841*	*3155*	*T*
Ï	*2397*	*2711*	*1112*	*1540*	*o*
8	*569*	*2808*	*830*	*3637*	*S*
Ó	*2435*	*2541*	*322*	*1655*	
Ś	*3463*	*3705*	*490*	*977*	*1*
R	*825*	*391*	*502*	*3169*	*2*
Ø	*2489*	*1626*	*510*	*432*	*3*

En la **Figura 90** se puede observar cómo se convierte el conjunto de puntos computados a caracteres, obteniendo el mensaje descifrado "***D@ToS 123***".

Figura 90

Representación de los puntos descifrados a texto

```
>> puntostotext([680    1875
645 2818
841 3155
1112    1540
830 3637
322 1655
490 977
502 3169
510 432
],-3,1,3851,10)

ans =

    D@ToS 123'
```

Para verificar el funcionamiento del algoritmo llamado "**descifradorElGamalEcc2**", se ingresan los parámetros de la curva (***a***, ***b***), el tamaño del campo ***p***, la clave secreta compartida ***S*** y el mensaje cifrado "**MsjCifrado**", La **Figura 91** muestra el mensaje de salida esperado, es decir el mensaje original, esta función realiza todos los procesos antes mencionados, facilitando la implementación de ECC.

Figura 91

Comprobación del algoritmo para el descifrado de datos

```
>> descifradorElGamalEcc2(-3,1,3851,10,[1769 841],[2707 1061
3610    756
721 3783
2397    2711
569 2808
2435    2541
3463    3705
825 391
2489    1626
])

ans =

   'D@ToS 123'
```

4.2 Prueba 2: Funcionamiento de la red 802.15.4 y diferentes despliegues de WSN

El objetivo de las redes **802.15.4** es proporcionar una solución de bajo costo de procesamiento, bajo consumo de energía y bajo ancho de banda para la comunicación de dispositivos IoT, y WSN. En esta prueba, se examinará el funcionamiento de diferentes despliegues de redes **802.15.4** y WSN utilizando la herramienta de simulación Omnet++. El objetivo es explorar cómo estas redes pueden ser utilizadas en diferentes entornos y cómo la implementación de la criptografía puede mejorar tanto la seguridad como el rendimiento de la red.

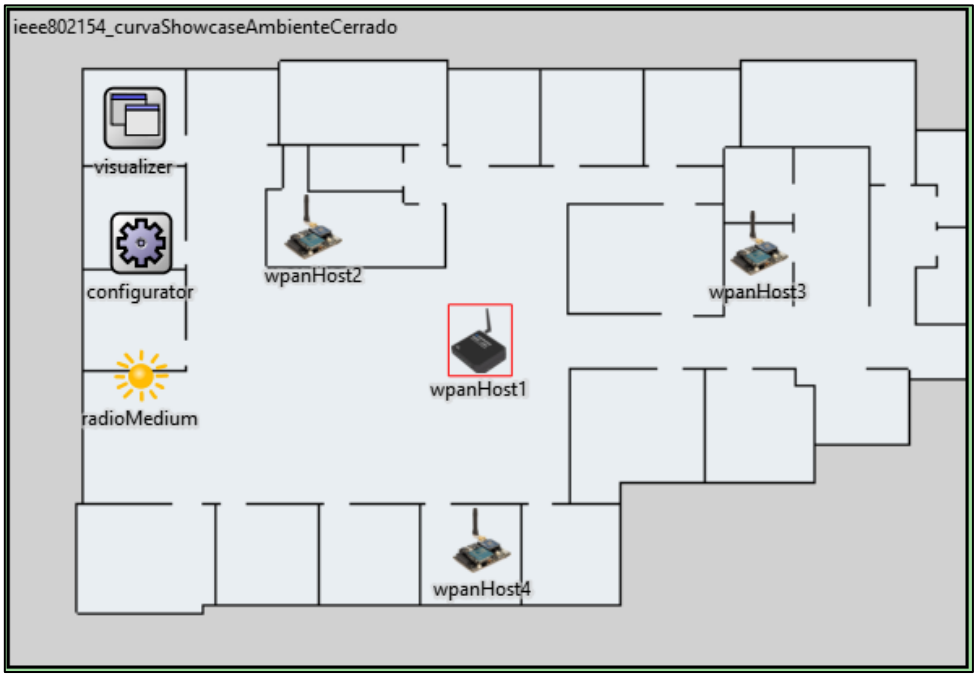
4.2.1 Ambiente Cerrado

El siguiente despliegue **Figura 92** trata de simular las características de la red vista en el escenario **Monitorización ambiental en lugares cerrados,** se trata de un despliegue de nodos para el monitoreo de monóxido de carbono en el aire, el sistema consta de:

- **Topología de la red:** estrella
- **Cantidad de Nodos:** 3 nodos sensores y 1 estación base
- **Protocolos de red utilizados:** TCP
- **Cantidad de datos a transmitir:** 1 dato

Figura 92

WSN para la Monitorización en ambiente cerrado



El primer proceso que se realiza es la inicialización de los nodos, donde se obtienen las direcciones IP y se especifican las rutas para cada uno de ellos. También se identifican cada uno de los dispositivos (**wpanHost1, wpanHost2, wpanHost3 y wpanHost4**) los cuales cuentan con un módulo de radio idéntico que consta de una antena isotrópica, un transmisor y receptor dimensional, usando ***IEEE 802.15.4.*** Estos parámetros se especifican en el escenario base **Figura 44**. Donde la **Figura 93** muestra los eventos producidos en consola, donde los módulos configurador (a) y radio (b), asignan valores a todos los nodos "**wpanHost***".

Figura 93

Mensajes por consola parámetros de inicio de los nodos

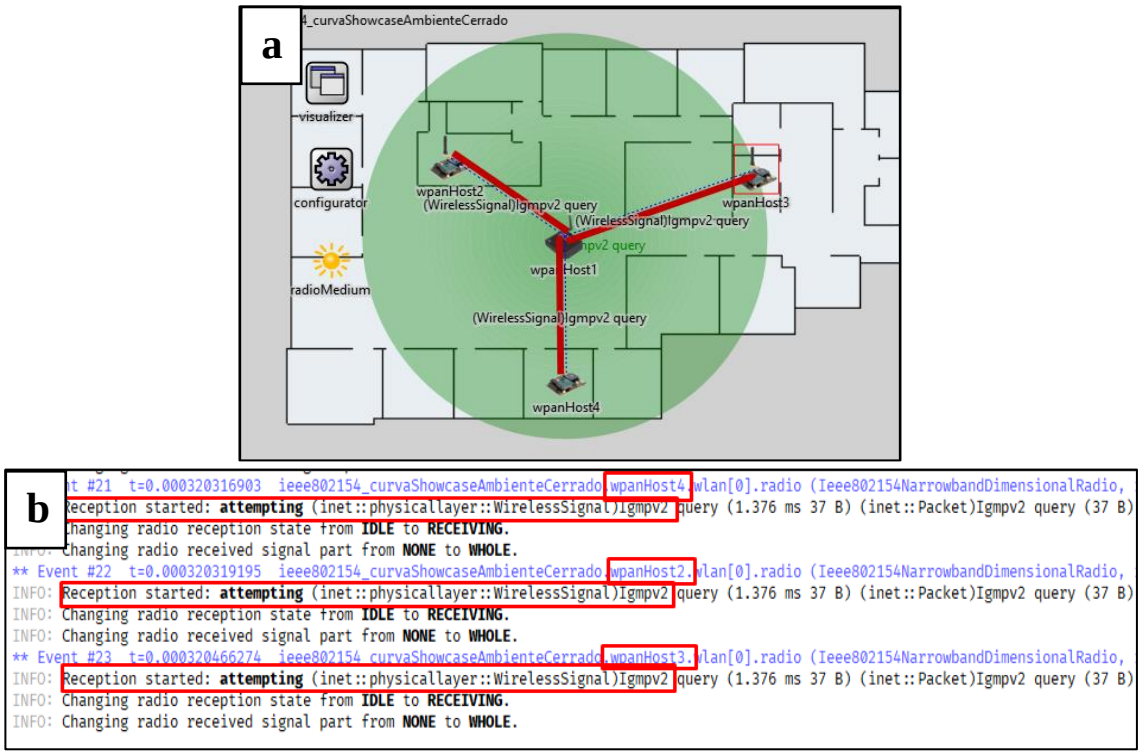
Una vez iniciado los nodos la estación base envía una trama **IGMPv2**[23] a todos los nodos simultáneamente, esto lo hace para permitir que los nodos sensores se comuniquen con la estación base y mantener un grupo multicast; es decir el nodo principal "**wpanHost1**" se comunicara directamente con todos los nodos sensores "**wpanHost2, wpanHost3 y wpanHost4**" enviando una sola trama, esto se lo hace para determinar el orden del envío y tambien se utiliza en el proceso de envío de contraseñas. La **Figura 94** muestra la trama *igmpv2* siendo enviada por el nodo principal hacia los demás dispositivos **(a)**, además se puede apreciar el mensaje en consola **(b)** donde se indica que los nodos sensores han recibido dicha trama.

[23] Internet Group Management Protocol version 2 es un protocolo de comunicación utilizado en redes de computadoras para gestionar y controlar el tráfico de multidifusión (multicast).

Finalmente se establece un grupo de multidifusión donde el nodo principal "**wpanHost1**" se puede comunicar con los demás nodos de la red.

Figura 94

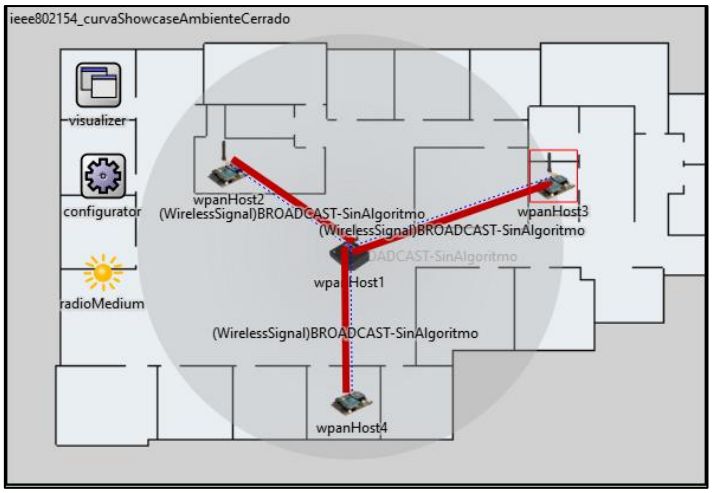
Envió de trama IGMPV2 del nodo principal hacia los nodos sensores



Una vez asignado el grupo de multidifusión, la estación base "**wpanHost1**" envía una trama llamada "**Boradcast-SinAlgoritmo**" destinada al nodo sensor "**wpanHost2**" el cual tiene la dirección IP **10.0.0.2**, esta trama se envía con el objetivo de indicar el orden de envio de datos, la estación base actúa como el nodo coordinador de los demás nodos. La **Figura 95** muestra el envío de la trama, así como en consola las direcciones IP y puertos de esta, tanto de destino como de origen respectivamente.

Figura 95

Envío de la trama "Broadcast-SinAlgoritmo" del nodo principal hacia los nodos sensores




```
, srcAddress = 10.0.0.1, destAddress = 10.0.0.2, options =  | UdpHeader, srcPort = 5000, destPort = 5000,
```

Todos los nodos sensores reciben la trama verifican la dirección MAC de destino, si no les pertenece descartan el paquete, la **Figura 96** muestra como todos los nodos pueden escuchar la trama dirigida a "**wpanHost2**", todos verifican la dirección MAC y solo el nodo sensor 2 acepta y desencapsula la trama.

Figura 96

Recepción de la trama "Broadcast-SinAlgoritmo"

```
** Event #68  t=0.014235201582  ieee802154_curvaShowcaseAmbienteCerrado.wpanHost3.wlan[0].mac (Ieee802154Narrowbanc
DETAIL: Received frame name= , myState=1 src=0A-AA-00-00-00-01 dst=0A-AA-00-00-00-02 myAddr=0A-AA-00-00-00-03
DETAIL: packet not for me, deleting...
```

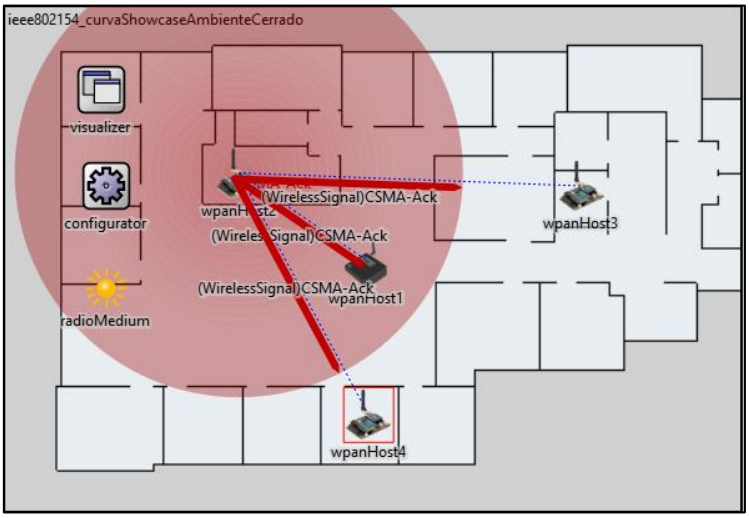
```
** Event #54  t=0.014235052211  ieee802154_curvaShowcaseAmbienteCerrado.wpanHost4.wlan[0].mac (Ieee802154Narrowbanc
DETAIL: Received frame name= , myState=1 src=0A-AA-00-00-00-01 dst=0A-AA-00-00-00-02 myAddr=0A-AA-00-00-00-04
DETAIL: packet not for me, deleting...
```

```
** Event #55  t=0.014235054503  ieee802154_curvaShowcaseAmbienteCerrado.wpanHost2.wlan[0].radio (Ieee802154NarrowbandDimensionalRa
INFO: Reception ended: successfully for (inet::physicallayer::WirelessSignal)BROADCAST-SinAlgoritmo (4.192 ms 125 B) (inet::Packet
INFO: Sending up (inet::Packet)BROADCAST-SinAlgoritmo (125 B) (inet::SequenceChunk) length = 125 B
INFO: Changing radio reception state from RECEIVING to IDLE.
** Event #56  t=0.014235054503  ieee802154_curvaShowcaseAmbienteCerrado.wpanHost2.wlan[0].mac (Ieee802154NarrowbandMac, id=109)  o
DETAIL: Received frame name= , myState=1 src=0A-AA-00-00-00-01 dst=0A-AA-00-00-00-02 myAddr=0A-AA-00-00-00-02
DETAIL: Received a data packet addressed to me, preparing an ack...
DETAIL: Adding a new child to the map of Sequence numbers:0A-AA-00-00-00-01
DETAIL: In executeMac
DETAIL: (15) FSM State IDLE_1, EV_FRAME_RECEIVED: setting up radio tx → WAITSIFS.
INFO (MessageDispatcher)ieee802154_curvaShowcaseAmbienteCerrado.wpanHost2.li: Dispatching packet to protocol, protocol = ipv4(50),
INFO (MessageDispatcher)ieee802154_curvaShowcaseAmbienteCerrado.wpanHost2.bl: Dispatching packet to protocol, protocol = ipv4(50),
INFO (MessageDispatcher)ieee802154_curvaShowcaseAmbienteCerrado.wpanHost2.cb: Dispatching packet to protocol, protocol = ipv4(50),
INFO (MessageDispatcher)ieee802154_curvaShowcaseAmbienteCerrado.wpanHost2.nl: Dispatching packet to protocol, protocol = ipv4(50),
INFO (MessageDispatcher)ieee802154_curvaShowcaseAmbienteCerrado.wpanHost2.ipv4.lp: Dispatching packet to protocol, protocol = ipv4
INFO (Ieee802154NarrowbandDimensionalRadio)ieee802154_curvaShowcaseAmbienteCerrado.wpanHost2.wlan[0].radio: Radio mode changed fro
INFO (Ieee802154NarrowbandDimensionalRadio)ieee802154_curvaShowcaseAmbienteCerrado.wpanHost2.wlan[0].radio: Changing radio recepti
INFO (Ieee802154NarrowbandDimensionalRadio)ieee802154_curvaShowcaseAmbienteCerrado.wpanHost2.wlan[0].radio: Changing radio transmi
INFO (Ieee802154NarrowbandDimensionalRadio)ieee802154_curvaShowcaseAmbienteCerrado.wpanHost2.wlan[0].radio: Changing radio receive
```

Ahora el nodo sensor "**wpanHost2**" envía una trama llamada "**CSMA-Ack**" con dirección al nodo "**wpanHost1"**, notificando que ha recibido el broadcast y pidiendo el acceso al medio, por lo tanto, los demás nodos también escuchan esta información y no usaran el medio hasta que termine de transmitir los datos de "**wpanHost2**". La **Figura 97** muestra el envio de la trama y en consola se aprecia la recepción del paquete por "**wpanHost1**", los demás nodos descartan la trama e inician el estado de suspensión "*modo sleep*" con el fin de minimizar el consumo de energía.

Figura 97

Envío de la trama "CSMA-Ack"



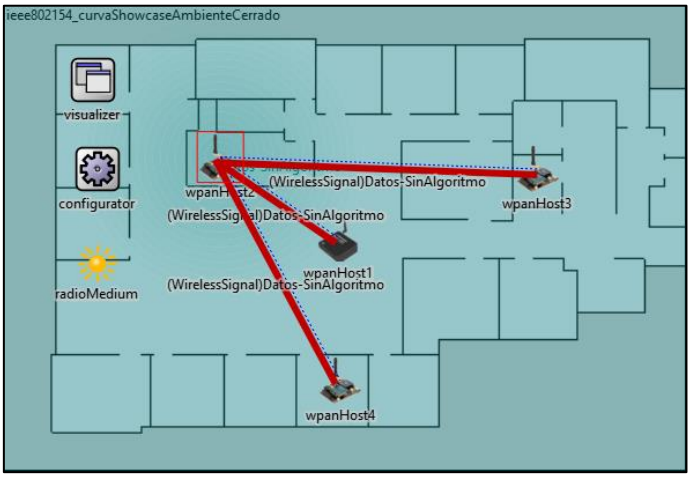

```
** Event #75  t=0.014779373698  ieee802154_curvaShowcaseAmbienteCerrado.wpanHost1.wlan[0].radio (Ieee802154NarrowbandDimensionalF
INFO: Reception ended: successfully for (inet::physicallayer::WirelessSignal)CSMA-Ack (352 us 5 B) (inet::Packet)CSMA-Ack (5 B) (
INFO: Sending up (inet::Packet)CSMA-Ack (5 B) (inet::Ieee802154MacHeader) srcAddr = 0A-AA-00-00-00-02, destAddr = 0A-AA-00-00-00-
INFO: Changing radio reception state from RECEIVING to IDLE.
** Event #76  t=0.014779373698  ieee802154_curvaShowcaseAmbienteCerrado.wpanHost1.wlan[0].mac (Ieee802154NarrowbandMac, id=63)  c
DETAIL: Received frame name= , myState=5 src=0A-AA-00-00-00-02 dst=0A-AA-00-00-00-01 myAddr=0A-AA-00-00-00-01
DETAIL: In executeMac
DETAIL: (5) FSM State WAITACK_5, EV_ACK_RECEIVED:  ProcessAck, manageQueue...
DETAIL: (manageQueue) no packets to send, entering IDLE state.
** Event #77  t=0.014779618621  ieee802154_curvaShowcaseAmbienteCerrado.wpanHost4.wlan[0].radio (Ieee802154NarrowbandDimensionalF
INFO: Reception ended: successfully for (inet::physicallayer::WirelessSignal)CSMA-Ack (352 us 5 B) (inet::Packet)CSMA-Ack (5 B) (
INFO: Sending up (inet::Packet)CSMA-Ack (5 B) (inet::Ieee802154MacHeader) srcAddr = 0A-AA-00-00-00-02, destAddr = 0A-AA-00-00-00-
INFO: Changing radio reception state from RECEIVING to IDLE.
** Event #78  t=0.014779618621  ieee802154_curvaShowcaseAmbienteCerrado.wpanHost4.wlan[0].mac (Ieee802154NarrowbandMac, id=201)
DETAIL: Received frame name= , myState=1 src=0A-AA-00-00-00-02 dst=0A-AA-00-00-00-01 myAddr=0A-AA-00-00-00-04
DETAIL: packet not for me, deleting...
** Event #79  t=0.014779758829  ieee802154_curvaShowcaseAmbienteCerrado.wpanHost3.wlan[0].radio (Ieee802154NarrowbandDimensionalF
INFO: Reception ended: successfully for (inet::physicallayer::WirelessSignal)CSMA-Ack (352 us 5 B) (inet::Packet)CSMA-Ack (5 B) (
INFO: Sending up (inet::Packet)CSMA-Ack (5 B) (inet::Ieee802154MacHeader) srcAddr = 0A-AA-00-00-00-02, destAddr = 0A-AA-00-00-00-
INFO: Changing radio reception state from RECEIVING to IDLE.
** Event #80  t=0.014779758829  ieee802154_curvaShowcaseAmbienteCerrado.wpanHost3.wlan[0].mac (Ieee802154NarrowbandMac, id=155)
DETAIL: Received frame name= , myState=1 src=0A-AA-00-00-00-02 dst=0A-AA-00-00-00-01 myAddr=0A-AA-00-00-00-03
DETAIL: packet not for me, deleting...
```

Una vez el nodo "**wpanHost2**" ha ganado el medio, empieza a transmitir los datos de los sensores, estos se encapsulan en una trama denominada como "**Datos-SinAlgoritmo**" y se envían con dirección hacia el nodo sensor "**wpanHost1**", la **Figura 98** muestra el envio de la

trama y en consola como todos los nodos, al estar en escucha del medio pueden conocer el contenido del paquete, ya que no existe un mecanismo de cifrado. El nodo principal almacena los datos y prepara una respuesta de ACK, los demás nodos descartan los mismos.

Figura 98

Envío de datos en la trama "Datos-SinAlgoritmo"




```
** Event #91  t=0.019931373698  ieee802154_curvaShowcaseAmbienteCerrado.wpanHost1.udp (Udp, id=54)  on Datos-SinAlgoritmo (inet::Packet, id=
INFO: Packet Datos-SinAlgoritmo received from network, dest port 5000
INFO: Sending payload up to socket sockId=0
INFO (MessageDispatcher)ieee802154_curvaShowcaseAmbienteCerrado.wpanHost1.at: Dispatching packet to socket, socketId = 0, inGate = (omnetpp::
** Event #92  t=0.019931373698  ieee802154_curvaShowcaseAmbienteCerrado.wpanHost1.app[0] (UdpBasicApp, id=57)  on Datos-SinAlgoritmo (inet::
INFO: ++++++++++++++++++ ProcesandoPaquete: (inet::Packet)Datos-SinAlgoritmo (88 B) (inet::SequenceChunk) length = 125 B
INFO: ++++++++++++++++ peekData: (inet::ApplicationPacket) sequenceNumber = 1, strMsg = {"id":"2", "sensor1":"41", "sensor2":"67", "sensor3"
INFO: ******************** intMsg: 0 0
INFO: ++++++++++++++++ Recibido Datos-SinAlgoritmo
INFO: *Modo Sin Algoritmo!
INFO: **** Enviando paquete con direccion → 10.0.0.2
INFO (MessageDispatcher)ieee802154_curvaShowcaseAmbienteCerrado.wpanHost1.at: Dispatching packet to service, protocol = udp(75), servicePrim
** Event #93  t=0.019931373698  ieee802154_curvaShowcaseAmbienteCerrado.wpanHost1.udp (Udp, id=54)  on ACK-datos (inet::Packet, id=214)
INFO: Sending app packet ACK-datos over ipv4.
INFO (MessageDispatcher)ieee802154_curvaShowcaseAmbienteCerrado.wpanHost1.tn: Dispatching packet to service, protocol = ipv4(50), servicePri
INFO (MessageDispatcher)ieee802154_curvaShowcaseAmbienteCerrado.wpanHost1.ipv4.up: Dispatching packet to service, protocol = ipv4(50), servi
INFO (MessageDispatcher)ieee802154_curvaShowcaseAmbienteCerrado.wpanHost1.ipv4.mp: Dispatching packet to service, protocol = ipv4(50), servi
** Event #94  t=0.019931373698  ieee802154_curvaShowcaseAmbienteCerrado.wpanHost1.ipv4.ip (Ipv4, id=80)  on ACK-datos (inet::Packet, id=214)
INFO: Received (inet::Packet)ACK-datos (96 B) (inet::SequenceChunk) length = 96 B from upper layer.
DETAIL: Sending datagram 'ACK-datos' with destination = 10.0.0.2
INFO: Routing (inet::Packet)ACK-datos (116 B) (inet::SequenceChunk) length = 116 B with destination = 10.0.0.2, output interface = wlan0, ne
INFO: Sending (inet::Packet)ACK-datos (116 B) (inet::SequenceChunk) length = 116 B to output interface = wlan0.
INFO (MessageDispatcher)ieee802154_curvaShowcaseAmbienteCerrado.wpanHost1.ipv4.lp: Dispatching packet to interface, interfaceId = 101, inGat
INFO (MessageDispatcher)ieee802154_curvaShowcaseAmbienteCerrado.wpanHost1.nl: Dispatching packet to interface, interfaceId = 101, inGate = (
INFO (MessageDispatcher)ieee802154_curvaShowcaseAmbienteCerrado.wpanHost1.cb: Dispatching packet to interface, interfaceId = 101, inGate = (
INFO (MessageDispatcher)ieee802154_curvaShowcaseAmbienteCerrado.wpanHost1.bl: Dispatching packet to interface, interfaceId = 101, inGate = (
INFO (MessageDispatcher)ieee802154_curvaShowcaseAmbienteCerrado.wpanHost1.li: Dispatching packet to interface, interfaceId = 101, inGate = (
```

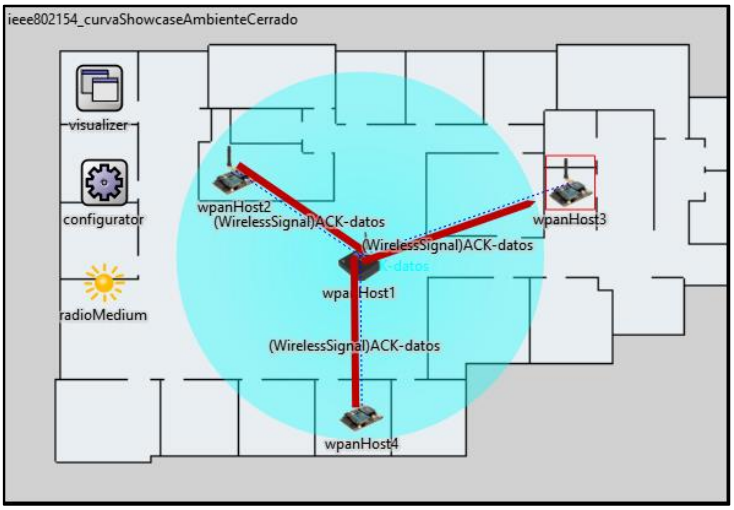
```
** Event #100  t=0.019931618621  ieee802154_curvaShowcaseAmbienteCerrado.wpanHost4.app[0] (UdpBasicApp, id=195)  on Datos-SinAlgoritmo (inet
INFO: ++++++++++++++++++ ProcesandoPaquete: (inet::Packet)Datos-SinAlgoritmo (88 B) (inet::SequenceChunk) length = 125 B
INFO: ++++++++++++++++ peekData: (inet::ApplicationPacket) sequenceNumber = 1, strMsg = {"id":"2", "sensor1":"41", "sensor2":"67", "sensor3"
INFO: ******************** intMsg: 0 0
** Event #101  t=0.019931758829  ieee802154_curvaShowcaseAmbienteCerrado.wpanHost3.wlan[0].radio (Ieee802154NarrowbandDimensionalRadio, id=1
INFO: Reception ended: successfully for (inet::physicallayer::WirelessSignal)Datos-SinAlgoritmo (4.192 ms 125 B) (inet::Packet)Datos-SinAlgo
INFO: Sending up (inet::Packet)Datos-SinAlgoritmo (125 B) (inet::SequenceChunk) length = 125 B
INFO: Changing radio reception state from RECEIVING to IDLE.
```

```
** Event #104  t=0.019931758829  ieee802154_curvaShowcaseAmbienteCerrado.wpanHost3.udp (Udp, id=146)  on Datos-SinAlgoritmo (inet::Packet, i
INFO: Packet Datos-SinAlgoritmo received from network, dest port 5000
INFO: Sending payload up to socket sockId=2
INFO (MessageDispatcher)ieee802154_curvaShowcaseAmbienteCerrado.wpanHost3.at: Dispatching packet to socket, socketId = 2, inGate = (omnetpp:
** Event #105  t=0.019931758829  ieee802154_curvaShowcaseAmbienteCerrado.wpanHost3.app[0] (UdpBasicApp, id=149)  on Datos-SinAlgoritmo (inet
INFO: ++++++++++++++++++ ProcesandoPaquete: (inet::Packet)Datos-SinAlgoritmo (88 B) (inet::SequenceChunk) length = 125 B
INFO: ++++++++++++++++ peekData: (inet::ApplicationPacket) sequenceNumber = 1, strMsg = {"id":"2", "sensor1":"41", "sensor2":"67", "sensor3"
INFO: ******************** intMsg: 0 0
```

El nodo sensor **wpanHost1** envía la verificación de haber recibido los datos con un "**Ack-datos**" **Figura 99** dirigido hacia nodo sensor "**wpanHost2**", en consola se aprecia como el nodo sensor 2 recibe el ACK y asi concluye el envio de los datos.

Figura 99

Confirmación de recepción de datos "ACK-Datos"

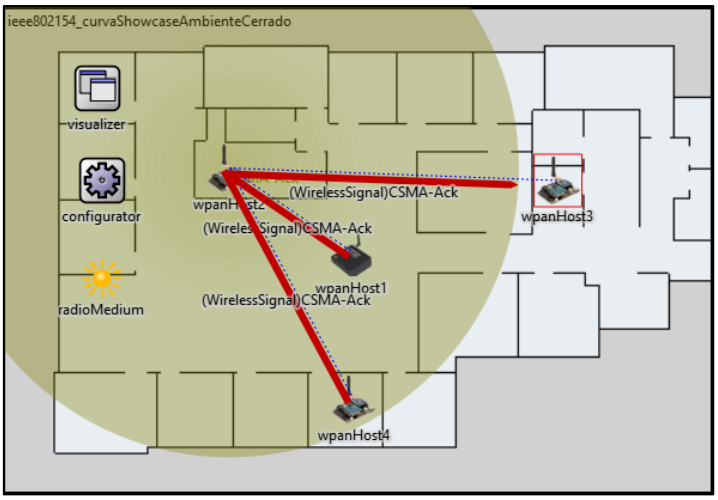



```
** Event #118  t=0.024763692893  ieee802154_curvaShowcaseAmbienteCerrado.wpanHost2.ipv4.ip (Ipv4, id=126)  on ACK-datos (inet::P
INFO: Received (inet::Packet)ACK-datos (116 B) (inet::SequenceChunk) length = 125 B from network.
DETAIL: Received datagram `' with dest=10.0.0.2
INFO: Delivering (inet::Packet)ACK-datos (116 B) (inet::SequenceChunk) length = 125 B locally.
INFO: Passing up to protocol udp(75)
INFO (MessageDispatcher)ieee802154_curvaShowcaseAmbienteCerrado.wpanHost2.ipv4.mp: Dispatching packet to protocol, protocol = ud
INFO (MessageDispatcher)ieee802154_curvaShowcaseAmbienteCerrado.wpanHost2.ipv4.up: Dispatching packet to protocol, protocol = ud
INFO (MessageDispatcher)ieee802154_curvaShowcaseAmbienteCerrado.wpanHost2.tn: Dispatching packet to protocol, protocol = udp(75)
** Event #119  t=0.024763692893  ieee802154_curvaShowcaseAmbienteCerrado.wpanHost2.udp (Udp, id=100)  on ACK-datos (inet::Packet
INFO: Packet ACK-datos received from network, dest port 5000
INFO: Sending payload up to socket sockId=1
INFO (MessageDispatcher)ieee802154_curvaShowcaseAmbienteCerrado.wpanHost2.at: Dispatching packet to socket, socketId = 1, inGate
** Event #120  t=0.024763692893  ieee802154_curvaShowcaseAmbienteCerrado.wpanHost2.app[0] (UdpBasicApp, id=103)  on ACK-datos (i
INFO: ++++++++++++++++++ ProcesandoPaquete: (inet::Packet)ACK-datos (88 B) (inet::SequenceChunk) length = 125 B
INFO: ++++++++++++++++ peekData: (inet::ApplicationPacket) sequenceNumber = 1, strMsg = ACK, datos recibidos!, strKind = ACK-dat
INFO: ****************** intMsg: 0 0
INFO: Recibido ACK-datos, proceso terminado...
```

Una vez ha terminado el proceso, el nodo sensor **"wpanHost2"** se encarga de comunicar a todos que ha terminado el proceso de envío y recepción de datos, dejando libre el medio para que sea utilizado por otro nodo. **Figura 100** muestra el envio de la trama CSMA-ACK, la cual indica que el medio esta libre; en consola se muestra como los nodos sensores cambian de estado, pasando del estado de suspensión al modo escucha, esperando a que el nodo coordinador les indique el momento para transmitir.

Figura 100

Liberación del medio para que otros nodos puedan enviar sus datos.

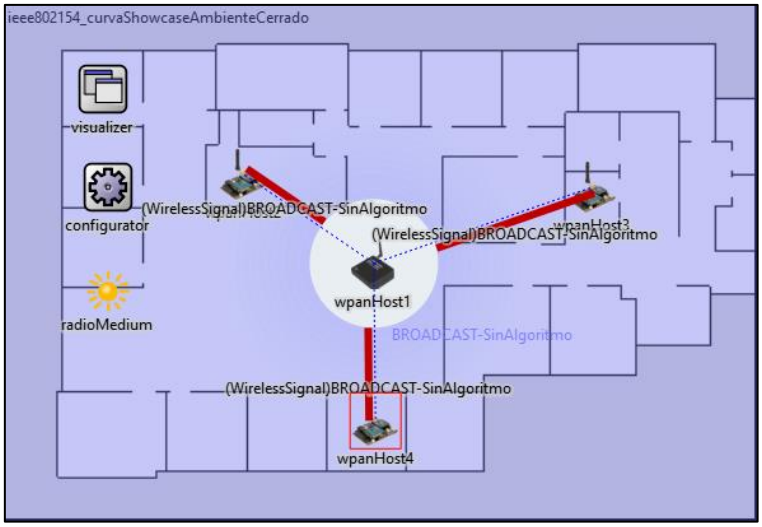



```
** Event #126  t=0.024956012088  ieee802154_curvaShowcaseAmbienteCerrado.wpanHost1.wlan[0].radio (Ieee802154NarrowbandDimen
INFO: Reception started: attempting (inet::physicallayer::WirelessSignal)CSMA-Ack (352 us 5 B) (inet::Packet)CSMA-Ack (5 B)
INFO: Changing radio reception state from IDLE to RECEIVING.
INFO: Changing radio received signal part from NONE to WHOLE.
** Event #127  t=0.024956257011  ieee802154_curvaShowcaseAmbienteCerrado.wpanHost4.wlan[0].radio (Ieee802154NarrowbandDimen
INFO: Reception started: attempting (inet::physicallayer::WirelessSignal)CSMA-Ack (352 us 5 B) (inet::Packet)CSMA-Ack (5 B)
INFO: Changing radio reception state from IDLE to RECEIVING.
** Event #128  t=0.024956397219  ieee802154_curvaShowcaseAmbienteCerrado.wpanHost3.wlan[0].radio (Ieee802154NarrowbandDimen
INFO: Reception started: attempting (inet::physicallayer::WirelessSignal)CSMA-Ack (352 us 5 B) (inet::Packet)CSMA-Ack (5 B)
INFO: Changing radio reception state from IDLE to RECEIVING.
```

El proceso se repite con el nodo principal "**wpanHost1**" enviando la trama de "**Broadcast-SinAlgoritmo**" **Figura 101** pero esta vez hacia el nodo sensor "**wpanHost3**" indicando que es su turno de transmitir los datos.

Figura 101

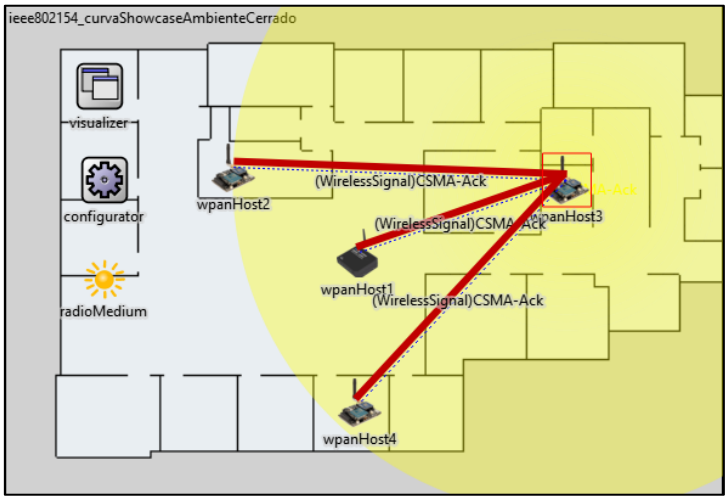
Liberación del medio para que otros nodos puedan enviar sus datos.



El nodo sensor "**wpanHost3**" envía un "**CSMA-Ack**", **Figura 102** hacia todos los nodos**,** donde el ACK es la confirmación que ha recibido la trama de **broadcast** y el **CSMA** indica que utilizara el medio y los demás nodos no podrán hacer uso de él.

Figura 102

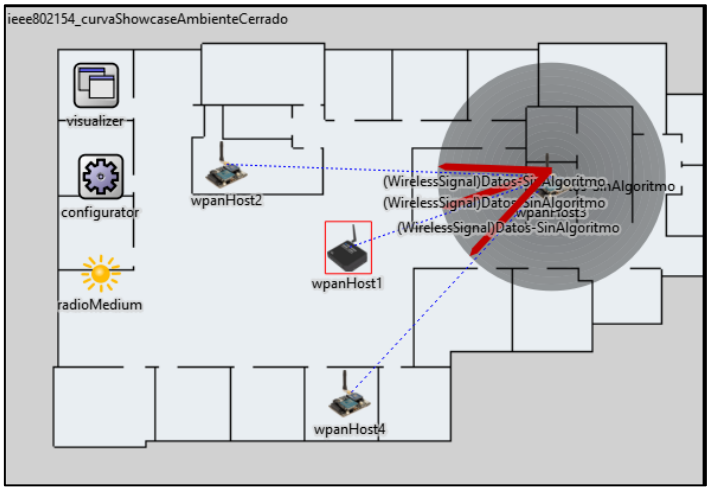
Envío de "CSMA-Ack" por parte de wpanHost3.



Una vez el nodo sensor "**wpanHost3**" ha ganado el medio empieza a transmitir los datos usando la trama "**Datos-SinAlgoritmo**" **Figura 103** hacia el nodo sensor "**wpanHost1**".

Figura 103

Envío de "CSMA-Ack" por parte de wpanHost3.



El proceso continúa para cada uno de los nodos que pertenecen a la red, y finaliza cuando los nodos han enviado la cantidad de datos previstos. En la **Tabla 17** se detalla las estadísticas de

la simulación obtenidas con Omnet++, donde se observa el tiempo de simulación, la cantidad de eventos generados, así como el número de transmisiones corresponde a todas las tramas enviadas donde se realizan 5 trasmisiones para el envío y recepción de los datos por cada uno de los 3 nodos sensores, más la trama inicial para el grupo de multidifusión dando un total de 16.

Tabla 17. *Resultados estadísticos del despliegue Ambiente cerrado*

Tiempo de simulación	0.225419694198 **s**
Cantidad de eventos	349
Cantidad de datos enviados por un nodo sensor	1
Cantidad de datos recibidos por **wpanHost1**	3
Número de transmisiones realizadas	16
Número de señales enviadas	48

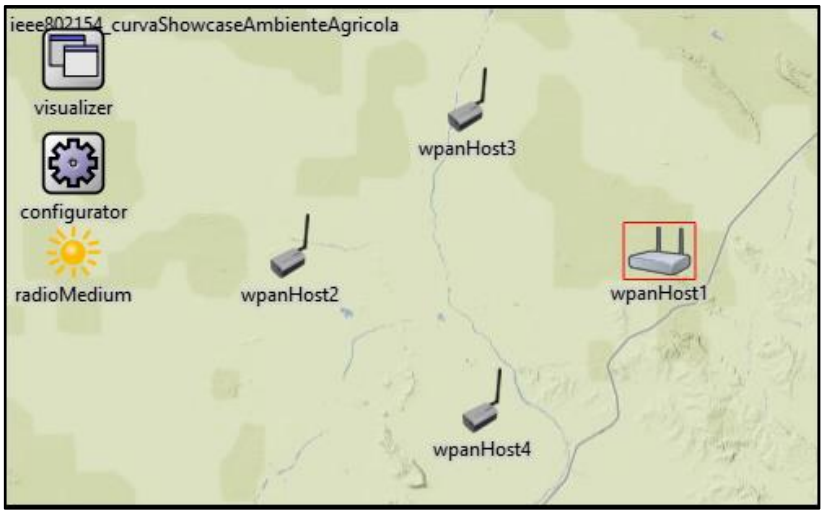
4.2.2 Ambiente Agrícola

El siguiente despliegue **Figura 104** trata de simular las características de la red vista en el escenario **Monitorización en la agricultura**, se trata de un despliegue de nodos basado en el concepto agricultura de precisión y la cual tiene como objetivo optimizar el control del sistema de riego por goteo, así como el monitoreo de agentes ambientales, el sistema consta de:

- **Topología de la red:** estrella
- **Cantidad de Nodos:** 3 nodos sensores y 1 estación base
- **Protocolos de red utilizados:** TCP
- **Cantidad de datos a transmitir:** 2 datos

Figura 104

WSN para la Monitorización en ambiente Agrícola



El funcionamiento de la red dentro de la simulación es el mismo que en **Ambiente Cerrado.** En la **Tabla 18** se detalla las estadísticas de la simulación obtenidas con Omnet++, donde se observa el tiempo de simulación, la cantidad de eventos generados, así como el número de transmisiones corresponde a todas las tramas enviadas. Nótese como al enviar el doble de datos los valores de la tabla coinciden con el doble de la **Tabla 17**.

Tabla 18. *Resultados estadísticos del despliegue Ambiente agrícola*

Tiempo de simulación	0.24530878044 **s**
Cantidad de eventos	701
Cantidad de datos enviados por un nodo sensor	2
Cantidad de datos recibidos por **wpanHost1**	6
Número de transmisiones realizadas	31
Número de señales enviadas	93

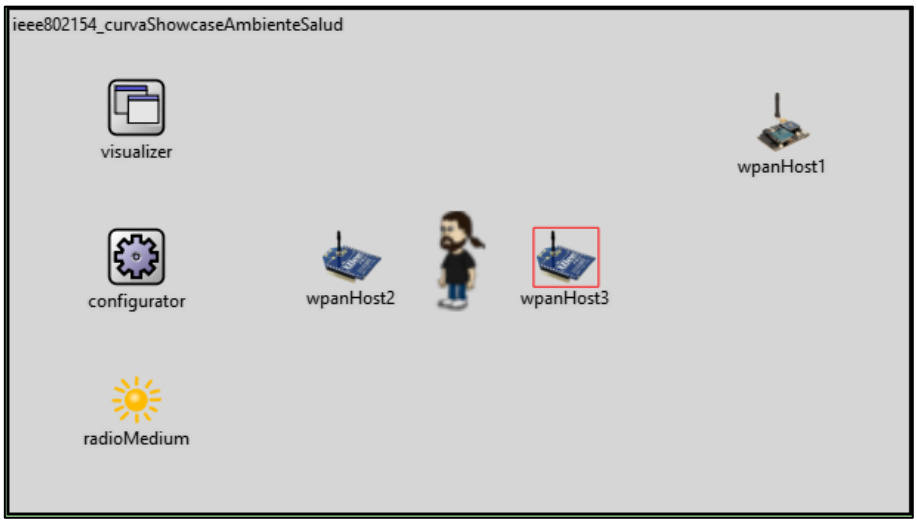
4.2.3 Ambiente Salud

El siguiente despliegue **Figura 105** trata de simular las características de la red vista en el escenario **Monitorización de la salud**, se trata de un despliegue de nodos basado en la detección temprana de anomalías en la salud de pacientes diagnosticados con COVID19, el sistema consta de:

- **Topología de la red:** estrella
- **Cantidad de Nodos:** 2 nodos sensores y 1 estación base
- **Protocolos de red utilizados:** TCP
- **Cantidad de datos a transmitir:** 1 dato

Figura 105

WSN para la Monitorización en ambiente salud



El funcionamiento de la red dentro de la simulación es el mismo que en **Ambiente Cerrado.** En la **Tabla 19** se detalla las estadísticas de la simulación obtenidas con Omnet++, donde se observa el tiempo de simulación, la cantidad de eventos generados, así como el número de transmisiones corresponde a todas las tramas enviadas.

Tabla 19. *Resultados estadísticos del despliegue Ambiente salud*

Tiempo de simulación	0.124988101536 **s**
Cantidad de eventos	194

Cantidad de datos enviados por un nodo sensor	1
Cantidad de datos recibidos por **wpanHost1**	2
Número de transmisiones realizadas	11
Número de señales enviadas	22

4.3 Prueba 3: Elección de la curva elíptica para la WSN y demostración de los algoritmos ECDH y ElGamal

En esta prueba se realiza una comparación entre las redes con el objetivo de determinar el tipo de curva elíptica sobre la cual realizaran el proceso de cifrado y la cual se implementa en la simulación. Los parámetros de las redes a tomar en cuenta son:

- **El nivel de seguridad** que deben tener los datos para proteger la privacidad y confidencialidad de estos.
- **El nivel de importancia de los datos** que se envían a través de la red de sensores.
- **El tiempo mínimo y máximo** requerido para el envío de los datos y así asegurar una respuesta oportuna.
- **La cantidad de energía** requerida para mantener y operar la red de sensores.
- **La eficacia de la implementación de medidas de seguridad criptográficas** en la red de sensores.

La **Tabla 20** muestra los resultados de los parámetros para cada una de las redes, se pretende aplicar una curva elíptica que no afecte al rendimiento de cada una de las redes.

Tabla 20. *Características de los diferentes despliegues de WSN*

Tipo de Red	***WSN para la monitorización en ambientes cerrados***	***WSN para la monitorización en la agricultura***	***WSN para la monitorización en la salud***
Descripción	Monitoreo de la calidad del aire, verificación de estados de puertas y ventanas en edificios, oficinas, escuelas, etc.	Monitoreo del suelo, humedad, nutrientes y condiciones climáticas para optimizar la producción agrícola.	Monitoreo de signos vitales, mediciones de glucosa, niveles de oxígeno en la sangre y otros datos para la prevención y diagnóstico de enfermedades.
Nivel de seguridad	**Alto** (datos confidenciales)	**Medio** (datos importantes, pero no confidenciales)	**Alto** (datos confidenciales)
Prioridad de envío de datos	**Alto**	**Alto**	**Alto**
Tiempo mínimo de envío de datos	Segundos	Minutos	Segundos

Tiempo máximo de envío de datos	2 o 3 minutos	Horas	1 minuto
Consumo de energía	**Medio**	**Bajo**	**Medio**
Eficacia de la red implementando criptografía	**Medio** (la criptografía tiene un impacto moderado en el rendimiento de la red)	**Alta** (la criptografía no tiene un impacto significativo en el rendimiento de la red)	**Bajo** (la criptografía reduce significativamente el rendimiento de la red)

Por consiguiente, se realiza la siguiente distribución de curvas, contemplando todas las restricciones y generan la cantidad de puntos necesarias para representar cualquier carácter.

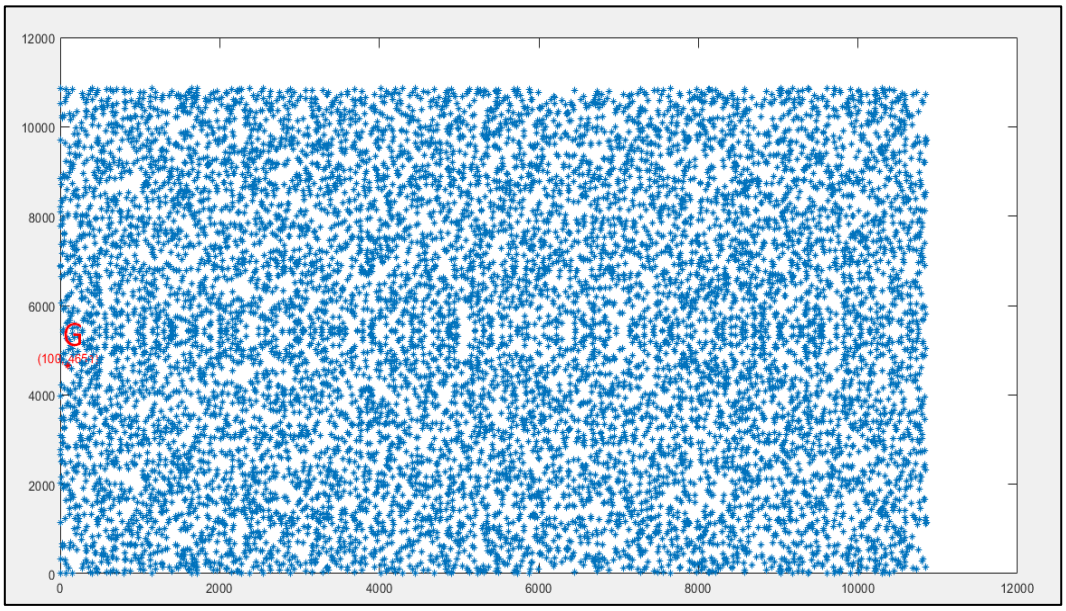
4.3.1 WSN para la monitorización en ambientes cerrados

Para la red de sensores destinada a la monitorización en ambientes cerrados se implementa la siguiente curva elíptica (**Figura 106**), la misma genera un impacto moderado, generando un pequeño retardo en la comunicación de la red y generando tamaños de clave de tipo medio. La prueba se realiza con los siguientes parámetros y de estos depende el funcionamiento de la criptografía y el correcto funcionamiento de la WSN.

- Parámetros de la curva ($\boldsymbol{a = -3, b = 1}$)
- Campo finito primo ($\boldsymbol{p = 10859}$)
- Parámetro de tolerancia al error ($\boldsymbol{k = 10}$)
- Punto Generador ($\boldsymbol{G = (100, 4651)}$)
- Cofactor ($\boldsymbol{h = 1}$)
- Cantidad de datos a enviar por nodo (2)

Figura 106

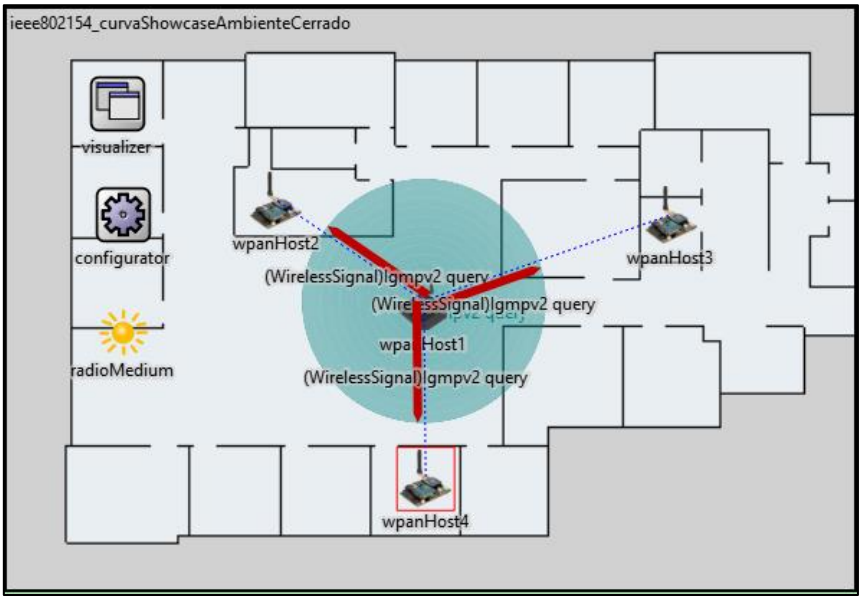
Curva elíptica $y^2 = x^3 - 3x + 1$ $(mod\ 10859)$ *en WSN para la Monitorización en ambiente cerrado*



El funcionamiento de la red cambia al utilizar ECC, ya que se utilizan diferentes tramas para el intercambio y envío de claves/datos. Aun así el primer paso que realiza la red es la inicialización de los nodos, seguido del envío de la trama "**IGMPv2**" **Figura 107** por parte del nodo coordinados "**wpanHost1**" para crear el grupo multicast.

Figura 107

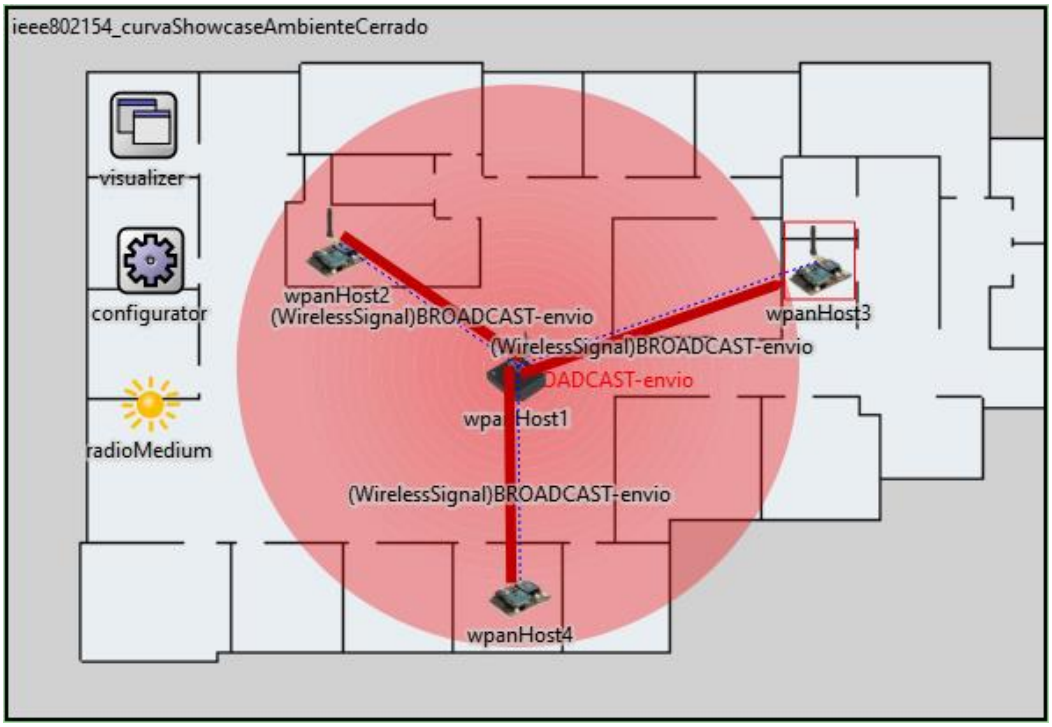
Envió de trama IGMPV2 del nodo principal hacia los nodos sensores



Una vez asignado el grupo; El nodo principal "**wpanHost1**" crea su clave publica (**10228 3886**), y genera la trama "**Broadcast-envio**" la cual esa destinada a la dirección IP **10.0.0.2** del nodo "**wpanHost2**". En la **Figura 108** muestra el envío de la trama, asi como el evento por consola en el cual indica la clave generada.

Figura 108

Envió de trama "Broadcast-envío" del nodo principal hacia el nodo sensor 2

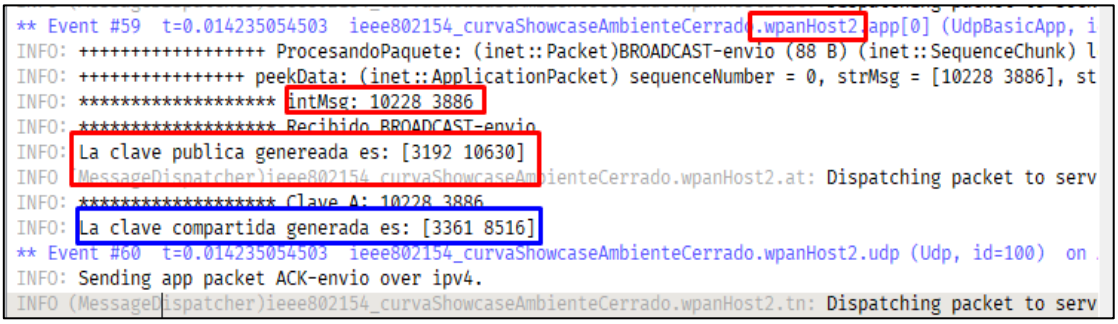



```
** Event #37  t=0.008442735308  ieee802154_curvaShowcaseAmbienteCerrado.wpanHost1.app[0]
INFO (MessageDispatcher)ieee802154_curvaShowcaseAmbienteCerrado.wpanHost1.at: Dispatchin
INFO (MessageDispatcher)ieee802154_curvaShowcaseAmbienteCerrado.wpanHost1.at: Dispatchin
INFO: multicastInterface: wlan0
INFO: receiveBroadcast: 1
INFO (MessageDispatcher)ieee802154_curvaShowcaseAmbienteCerrado.wpanHost1.at: Dispatchin
INFO (MessageDispatcher)ieee802154_curvaShowcaseAmbienteCerrado.wpanHost1.at: Dispatchin
INFO: joinLocalMulticastGroups: 1
INFO: La clave publica de A es: [10228 3886]
INFO: **** Enviando paquete con direccion → 10.0.0.2
INFO (MessageDispatcher)ieee802154_curvaShowcaseAmbienteCerrado.wpanHost1.at: Dispatchin
```

La trama es recibida por el nodo sensor "**wpanHost2**" **Figura 109**, obteniendo asi la clave publica de "**wpanHost1**" (**10228 3886**), Además este genera su clave publica (**3192 10630**) la cual va a ser enviada hacia "**wpanHost1**". Como el nodo sensor 2 ya dispone de una clave privada secreta y la clave publica de nodo 1, este puede generar la clave secreta compartida (**3361 8516**), esta se utilizará para cifrar el intercambio de datos solo entre estos nodos.

Figura 109

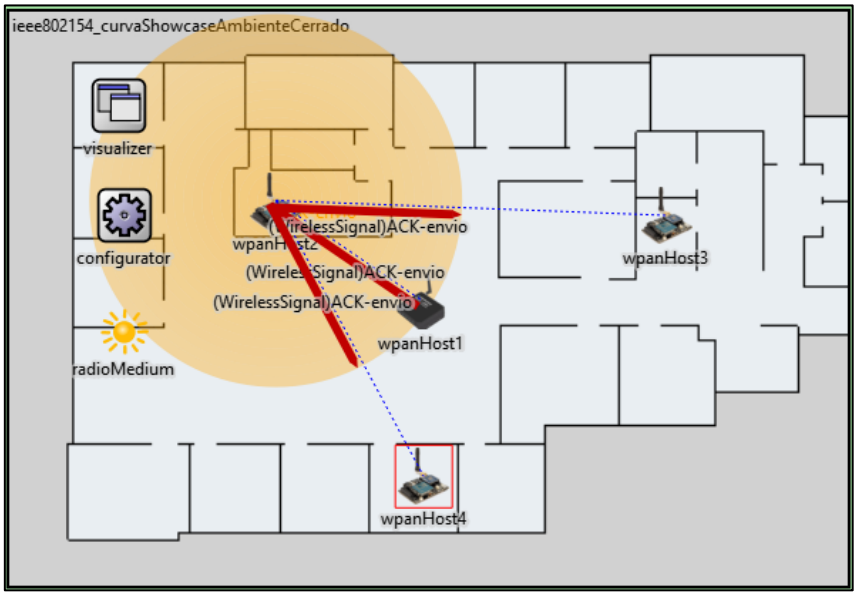
Cálculo de la clave secreta compartida entre "wpanHost1" y "wpanHost2"

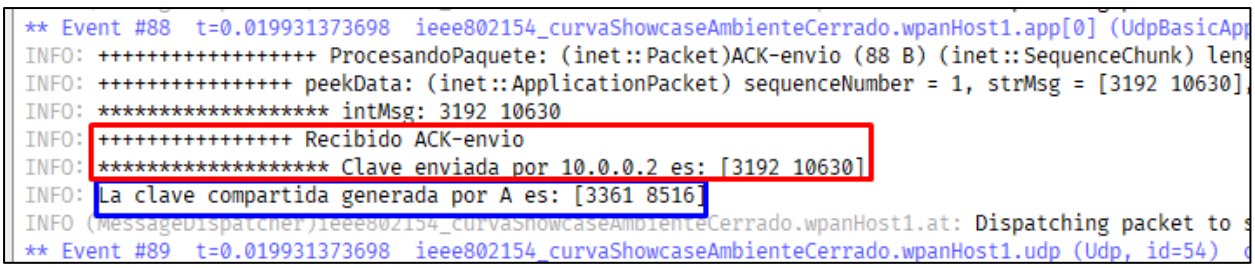


El nodo sensor "**wpanHost2**", envía un "**CSMA-Ack**" ganando el medio para después enviar la trama "**ACK-envio**" **Figura 110** el cual contiene la clave publica de "**wpanHost2**", con esto, ambos nodos han logrado obtener una clave secreta compartida (**3361 8516**), donde "**wpanHost2**" cifrara los datos utilizando esta clave y "**wpanHost1**" descifrara los mismos. Aquí finaliza el protocolo *ECDH* usado para el intercambio de claves.

Figura 110

Envío de la clave publica y obtención de clave secreta compartida

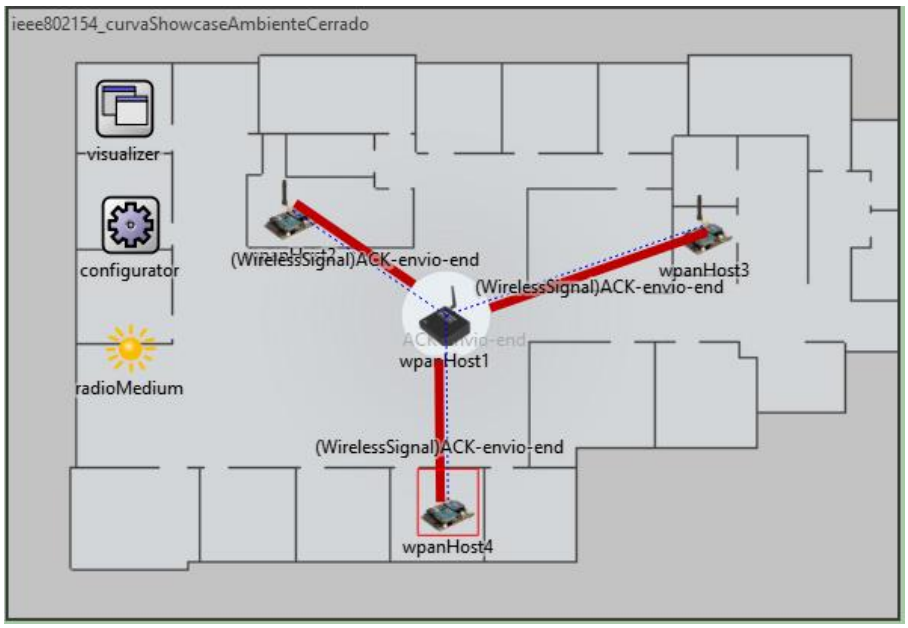




El nodo "**wpanHost1**" confirma la recepción de la clave enviando la trama "**ACK-envio-end**" **Figura 111**, anunciando que han finalizado el proceso de intercambio de claves.

Figura 111

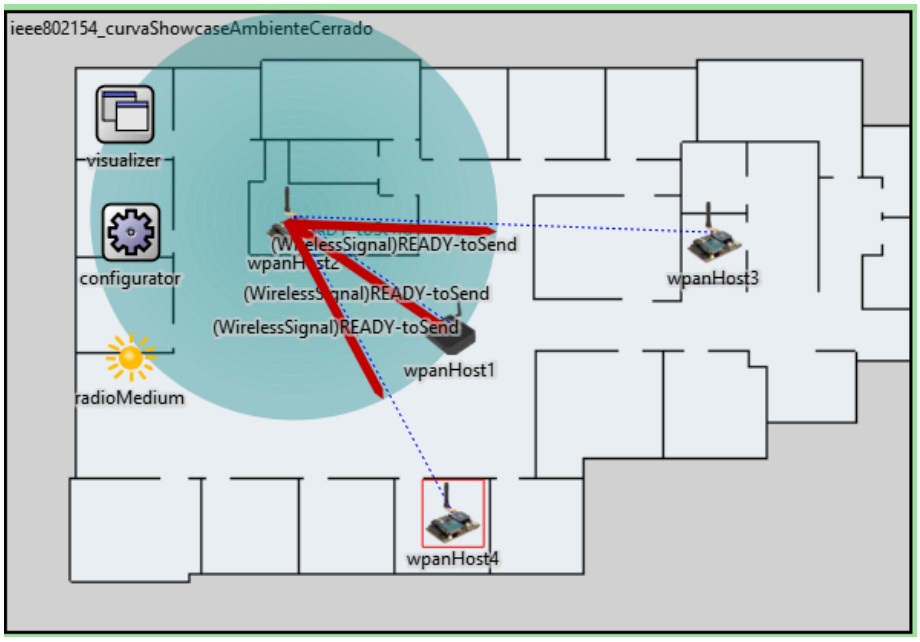
Envío de la clave publica y obtención de clave secreta compartida



El nodo wpanHost2 envía la trama "READY-toSend" **Figura 112**, indicando que ya tiene datos y está listo para enviar.

Figura 112

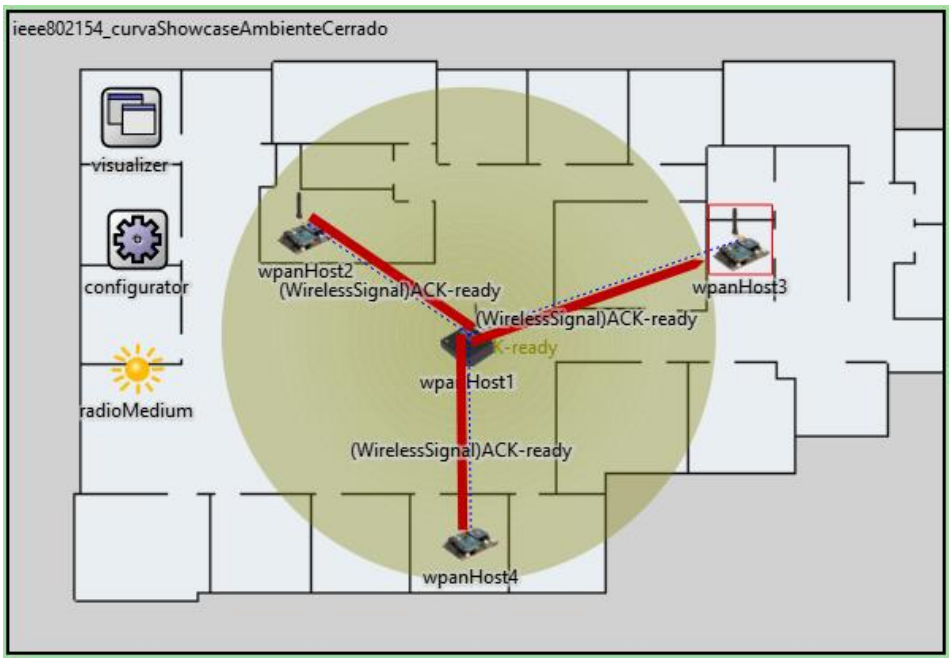
Envío de la trama "READY-toSend" de wpanHost2 hacia wpanHost1



El nodo "**wpanHost1**" confirma la recepción enviando la trama "**ACK-ready**" **Figura 113**.

Figura 113

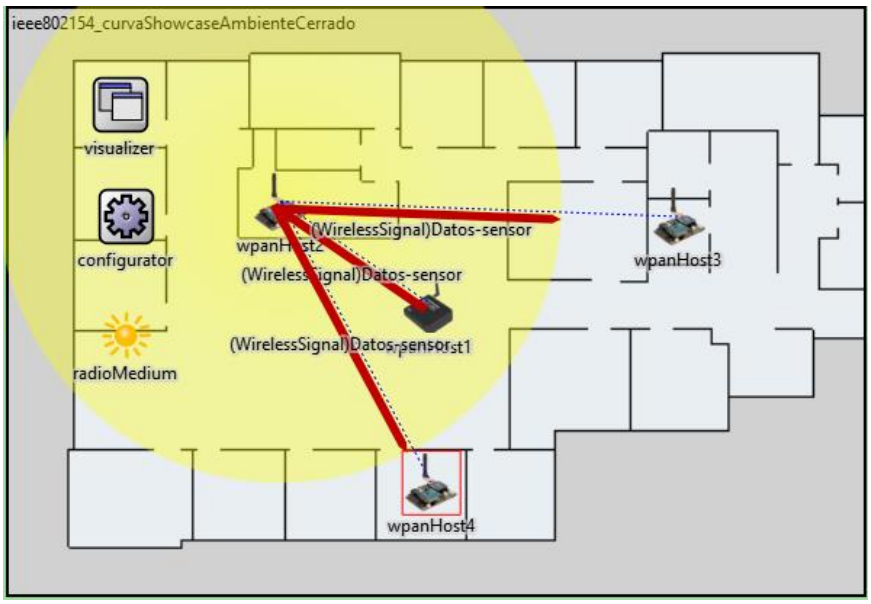
Envío de la trama "ACK-ready" de wpanHost1 hacia wpanHost2



Ahora, El nodo "**wpanHost2**" envía una trama "**CSMA-Ack**" indicando que va a utilizar el medio. Inmediatamente cifra los datos utilizando *ElGamal Elíptico* con la clave secreta compartida (**3361 8516**), además calcula el hash del conjunto de datos; Todo esto se encapsula en una misma trama llamada "**Datos-sensor**". La utilización del hash se realiza para que el nodo "**wpanHost1**" pueda verificar si existió alteración de los datos. La **Figura 114** muestra la trama, y los eventos ocurridos en consola donde el nodo1 obtiene los datos cifrados, realiza la comprobación del hash y descifra usando la misma clave compartida (**3361 8516**).

Figura 114

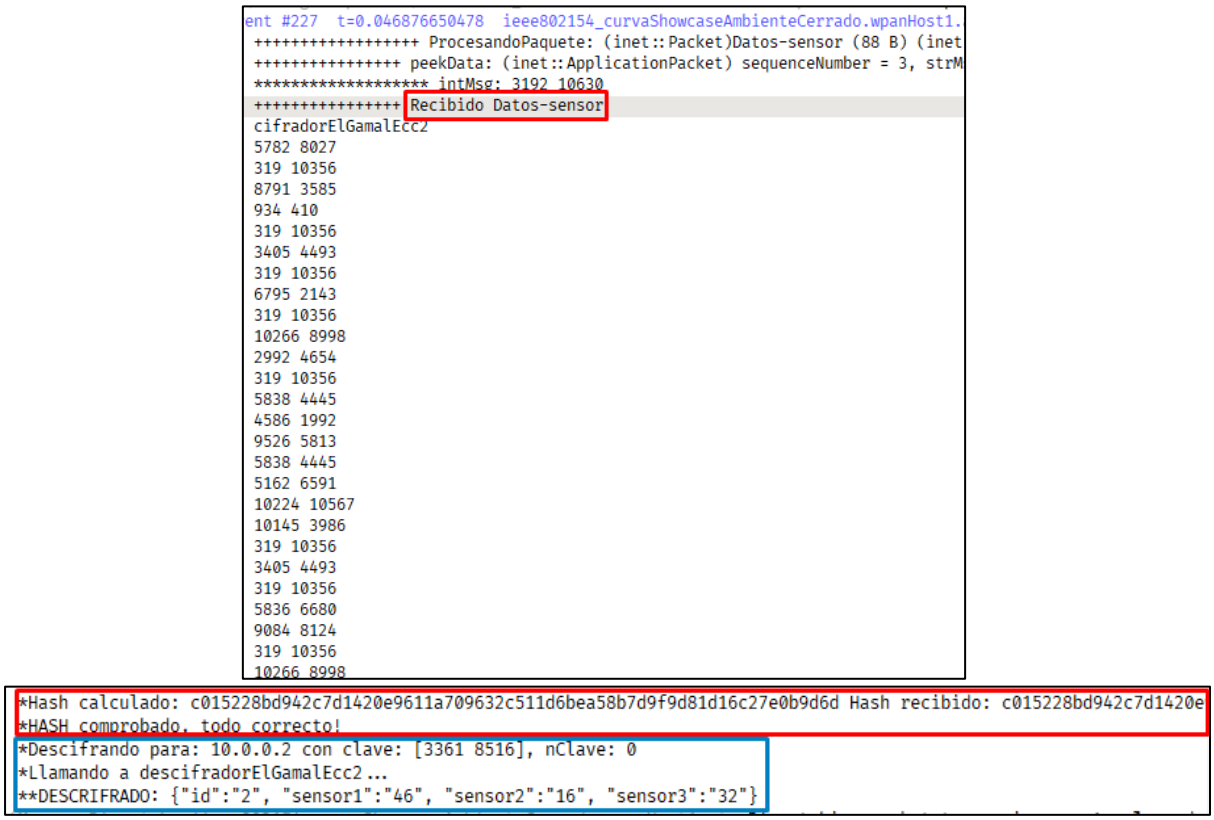
Envío de la trama "Datos-sensor" de wpanHost2 hacia wpanHost1



La **Figura 115** muestra el contenido de la trama al momento de recepción de los datos, el nodo wpanHost1 recibe la trama, en esta se observa todo el conjunto de puntos que forma el mensaje, además, el final de la trama tiene el hash calculado, esto para la verificación si el mensaje fue alterado. Si todo esta correcto desencripta los datos usando ***ElGamal Elíptico***.

Figura 115

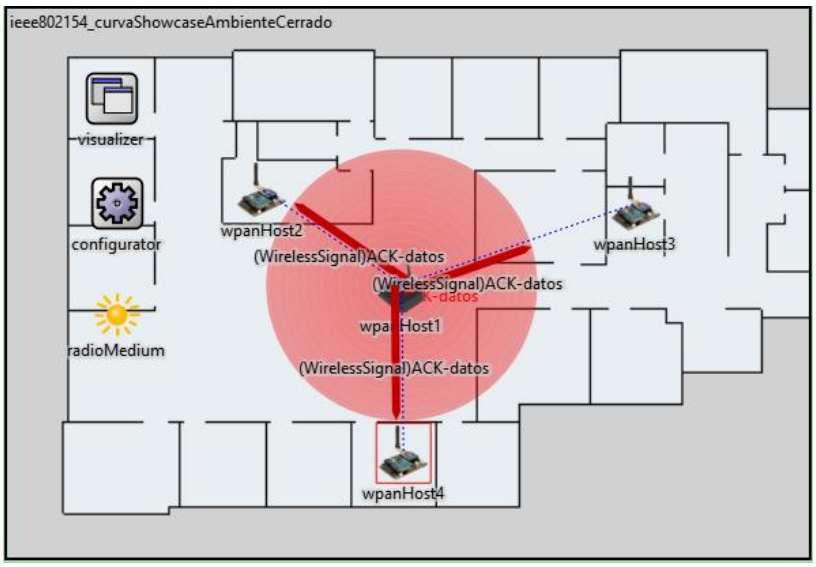
Contenido de la trama "Datos-sensor"



El nodo "**wpanHost1**" envía una trama "ACK-datos" **Figura 116** indicando la recepción de los datos

Figura 116

Envío de la trama "ACK-datos"



La red está configurada para que todos los nodos sensores envíen un total de 2 datos. Por ello el nodo "**wpanHost2**" inicial la actualización de claves **Figura 117**, actualizando matemáticamente la contraseña. El proceso es el siguiente: *Se seleccionan el punto* $\boldsymbol{x}$ *de la actual clave compartida* (**3361 8516**) *y se divide para dos, el resultado obtenido es redondeado hacia abajo, obteniendo asi un valor entero; Este valor será por el cual la clave compartida deba multiplicarse para obtener un punto diferente en la curva*. En este caso la ***nueva contraseña*** es (**3504 2067**). Esto se realiza en cada uno de los datos a enviar, el nodo cifra los datos, calcula el hash y envía la trama hacia "**wpanHost1**" **Figura 118**.

Figura 117

Evento: Actualización de contraseñas y cifrado de los datos No. 2.

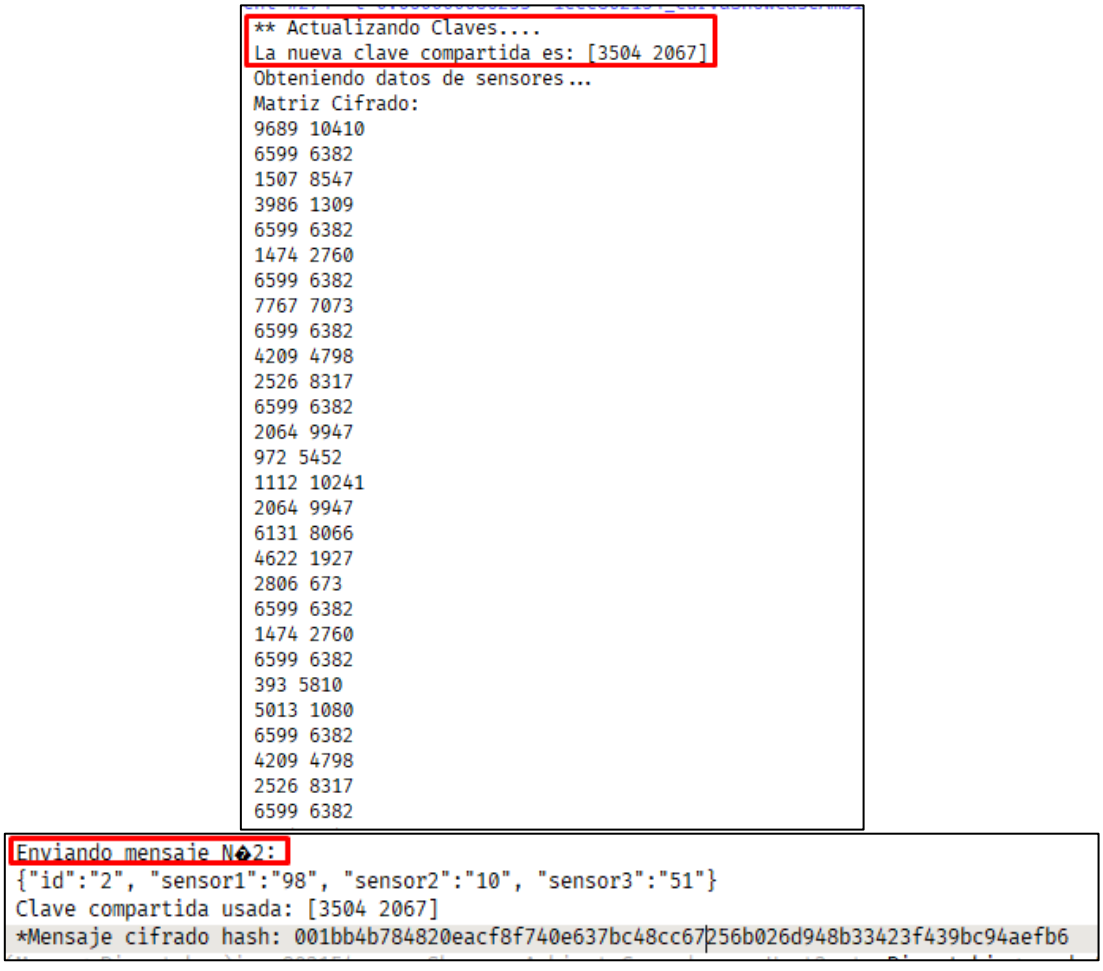
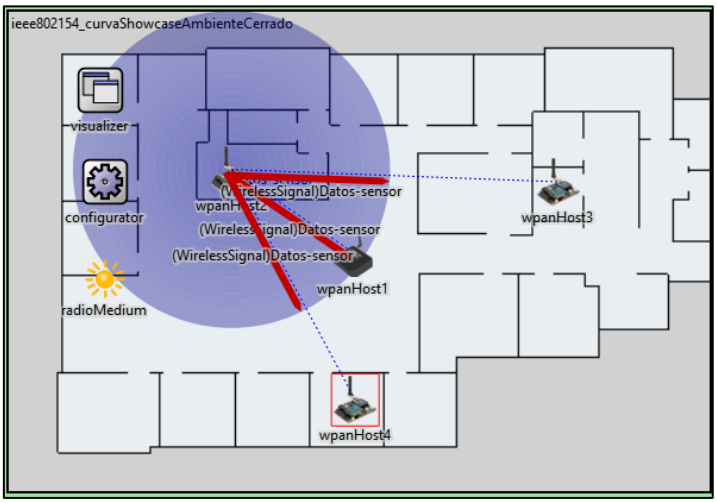


Figura 118

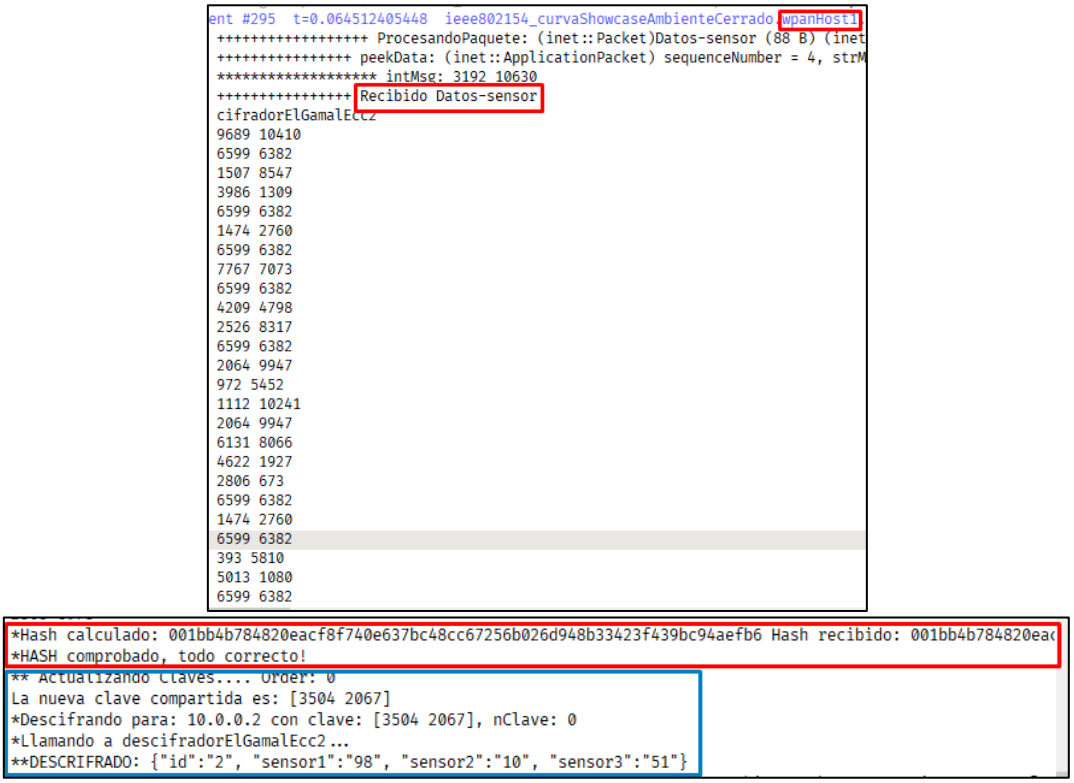
Envío de la trama "Datos-Sensor" con los datos No. 2.



El nodo "**wpanHost1**" recibe los datos, verifica que numero de paquete es y aplica el descifrado usando la nueva contraseña tambien obtenida matemáticamente. La **Figura 119** muestra la trama "**Datos-sensor**" obtenida por el nodo1, tambien se comprueba el hash en busca de alteraciones y descifra utilizando ***ElGamal Elíptico*** con la nueva clave (**3504 2067**).

Figura 119

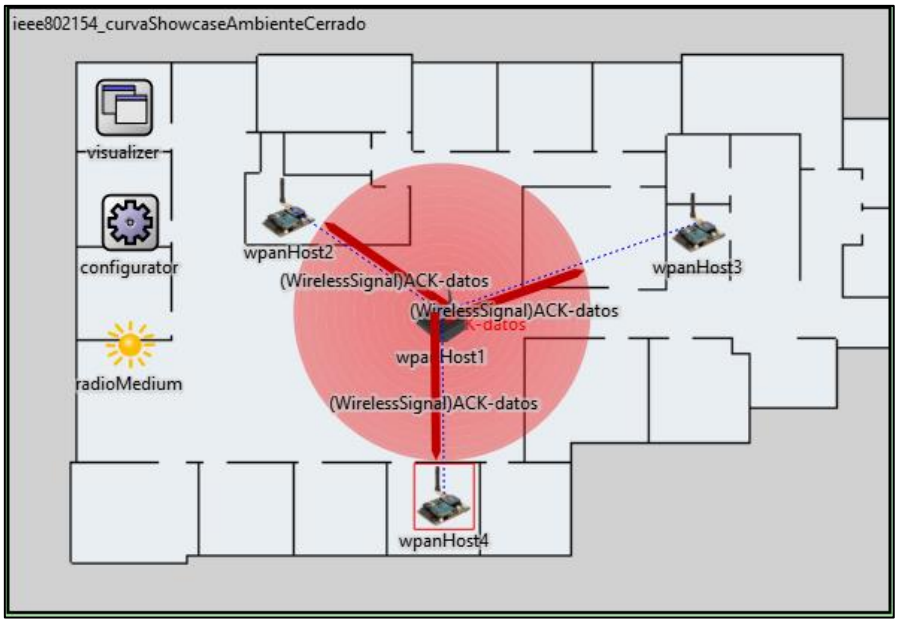
Contenido de la trama "Datos-Sensor" con los datos No. 2.



El nodo "**wpanHost1**" envía una trama "**ACK-datos**" **Figura 120** indicando la recepción de los datos No. 2 y la finalización del envio con el nodo "**wpanHost2**".

Figura 120

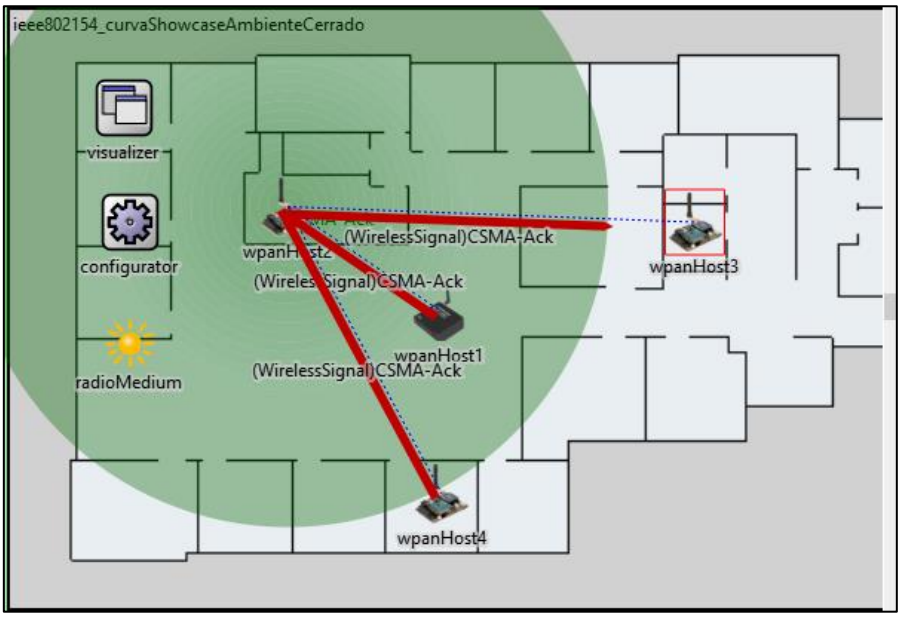
Envío de la trama "ACK-datos"



El nodo "**wpanHost2**" libera el medio enviando una trama "**CSMA-ACK**" **Figura 121**, permitiendo asi la comunicación de los demás nodos con el nodo coordinador "**wpanHost1**".

Figura 121

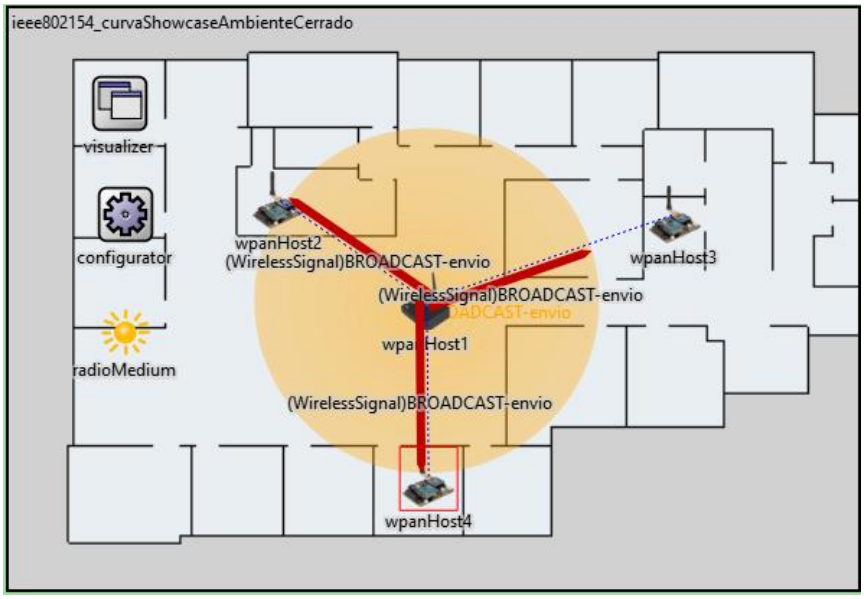
Envío de la trama "CSMA-ACK "



Finalmente, el nodo coordinador "**wpanHost1**" repite todo el proceso pero esta vez para otro de los nodos, primero enviando la trama "**BROADCAST-envio**" **Figura 122**, la cual contiene la clave publica, esta trama es enviada hacia "**wpanHost3**".

Figura 122

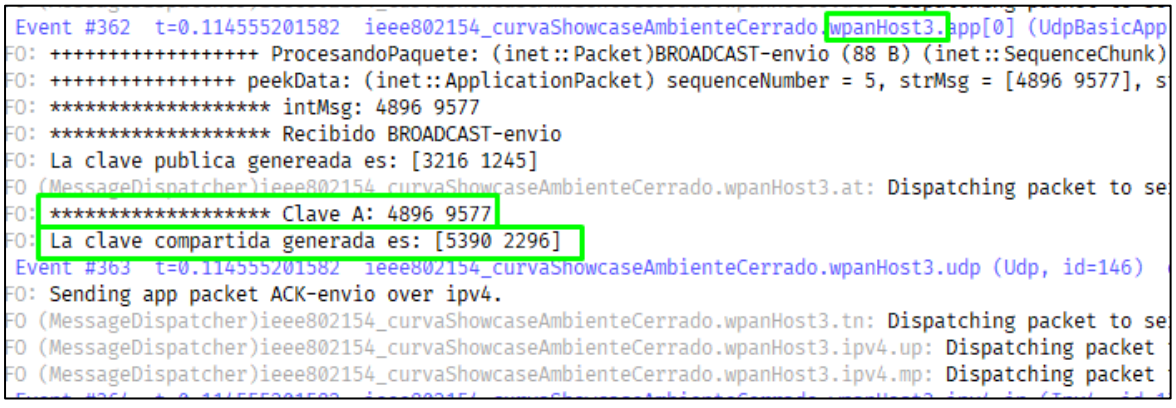
Envió de trama "Broadcast-envío" del nodo principal hacia el nodo sensor 3



El nodo "**wpanHost3**" recibe la clave y a partir de esta calcula la clave secreta compartida, el proceso se repite de la misma forma que el nodo "**wpanHost2**" la **Figura 123** muestra los eventos en consola generados por "**wpanHost3**".

Figura 123

Recepción de la trama "Broadcast-envío" obteniendo la clave publica de wpanHost1



En la **Tabla 21** se detalla las estadísticas de la simulación obtenidas con Omnet++, donde se observa el tiempo de simulación, la cantidad de eventos generados, así como el número de transmisiones corresponde a todas las tramas enviadas.

Tabla 21. *Resultados estadísticos del despliegue Ambiente cerrado con ECC*

Tiempo de simulación	0.279889290897 s
Cantidad de eventos	941
Cantidad de datos enviados por un nodo sensor	2
Cantidad de datos recibidos por **wpanHost1**	6
Número de transmisiones realizadas	49
Número de señales enviadas	147

Al analizar los datos de la **Tabla 17** y la **Tabla 21** se observa como el tiempo se simulación aumenta $\mathbf{0,54469}\ \boldsymbol{s}$, asi mismo la cantidad de transmisiones realizadas tiene una diferencia de $\mathbf{17}$ siendo mayor debido al intercambio de claves y cifrado de datos, la implementación de ECC en una red de sensores requiere el uso de más transmisiones así como tiempo de ejecución, debió a la generación de una contraseña común y al proceso de cifrado y descifrado.

4.3.2 WSN para la monitorización en ambientes Agrícolas

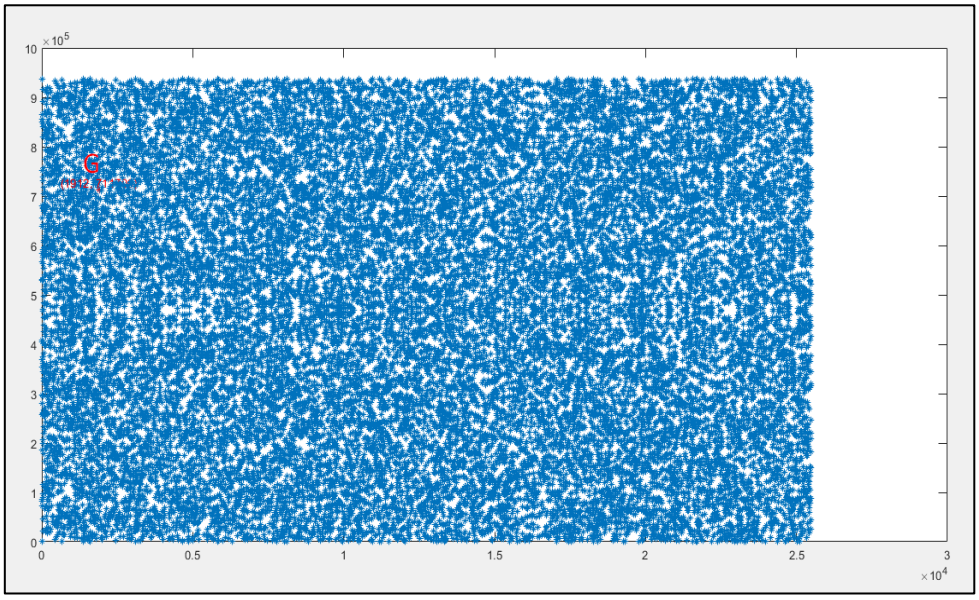
Para la red de sensores destinada la monitorización en la agricultura se implementa la siguiente curva elíptica **Figura 124**, la misma genera un impacto alto en la red, pero esta se puede permitirse ya que la prioridad de envio de los datos es baja, aun asi, se mantiene dentro del rango de tiempo mínimo de envio de datos.

- Parámetros de la curva ($\boldsymbol{a = -3, b = 1}$)
- Campo finito primo ($\boldsymbol{p = \ 937127}$)
- Parámetro de tolerancia al error ($\boldsymbol{k = 10}$)
- Punto Generador ($\boldsymbol{G\ = (1912, 711700)}$)

- Cofactor ($\boldsymbol{h = 1}$)

Figura 124

Curva elíptica $\boldsymbol{y^2 = x^3 - 3x + 1\ (mod\ 937127)}$ *en WSN para la Monitorización en ambiente agrícola*



El funcionamiento de la red dentro de la simulación es el mismo que en **WSN para la monitorización en ambientes cerrados.** En la **Tabla 22** se detalla las estadísticas de la simulación obtenidas con Omnet++, donde se observa el tiempo de simulación, la cantidad de eventos generados, así como el número de transmisiones corresponde a todas las tramas enviadas.

Tabla 22. *Resultados estadísticos del despliegue Ambiente agrícola con ECC*

Tiempo de simulación	0.279889584491 s
Cantidad de eventos	941
Cantidad de datos enviados por un nodo sensor	2
Cantidad de datos recibidos por **wpanHost1**	6
Número de transmisiones realizadas	49
Número de señales enviadas	147

Al analizar los datos de la **Tabla 18** y la **Tabla 22** se observa como el tiempo se simulación aumenta $\mathbf{0,3458}\ \boldsymbol{s}$, asi mismo la cantidad de transmisiones realizadas tiene una diferencia de **18** siendo mayor debido al intercambio de claves y cifrado de datos.

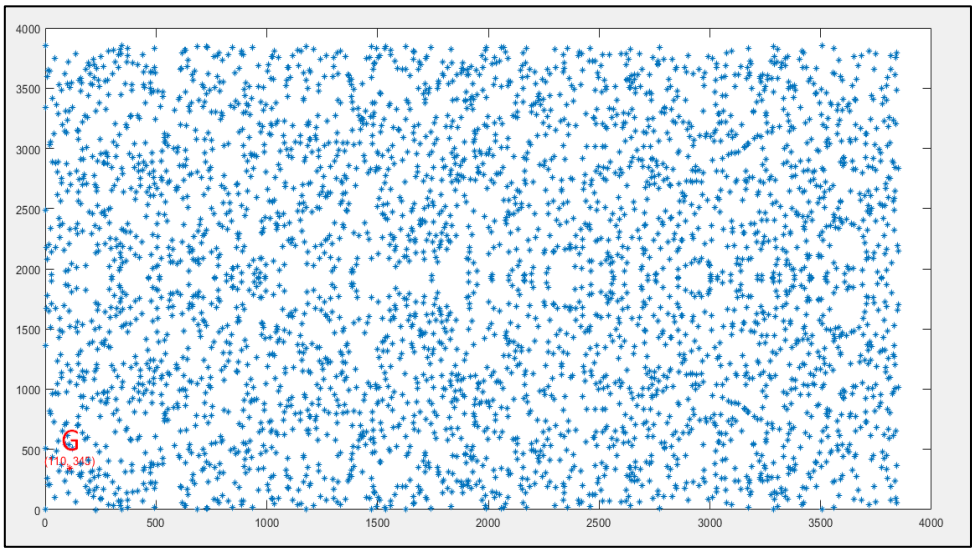
4.3.3 WSN para la monitorización en ambiente salud

Para la red de sensores destinada la monitorización en la salud se implementa la siguiente curva elíptica **Figura 125**, la misma genera un impacto bajo en la red, generando retardos despreciables de valores de ms, y aun manteniéndose dentro del rango para el tiempo mínimo de envío de datos.

- Parámetros de la curva ($\boldsymbol{a = 10, b = -10}$)
- Campo finito primo ($\boldsymbol{p = 3851}$)
- Parámetro de tolerancia al error ($\boldsymbol{k = 10}$)
- Punto Generador ($\boldsymbol{G = (110, 345)}$)
- Cofactor ($\boldsymbol{h = 2}$)
- Cantidad de datos a enviar por nodo (2)

Figura 125

Curva elíptica $\boldsymbol{y^2 = x^3 + 10x - 10\ (mod\ 3851)}$ *en WSN para la Monitorización en ambiente salud*



El funcionamiento de la red dentro de la simulación es el mismo que en **WSN para la monitorización en ambientes cerrados.** En la **Tabla 23** se detalla las estadísticas de la

simulación obtenidas con Omnet++, donde se observa el tiempo de simulación, la cantidad de eventos generados, así como el número de transmisiones corresponde a todas las tramas enviadas.

Tabla 23. *Resultados estadísticos del despliegue Ambiente salud con ECC*

Tiempo de simulación	0.163102150878 s
Cantidad de eventos	516
Cantidad de datos enviados por un nodo sensor	2
Cantidad de datos recibidos por **wpanHost1**	6
Número de transmisiones realizadas	33
Número de señales enviadas	99

Al analizar los datos de la **Tabla 19** y la **Tabla 23** se observa como el tiempo se simulación aumenta $\mathbf{0,38114049}\ \boldsymbol{s}$, asi mismo la cantidad de transmisiones realizadas tiene una diferencia de $\mathbf{11}$ siendo mayor debido al intercambio de claves y cifrado de datos.

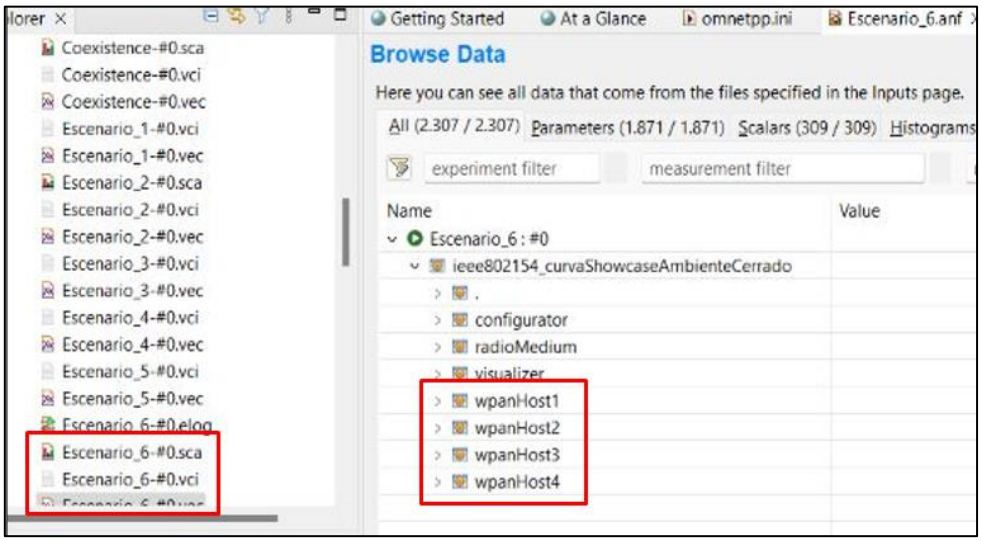
4.4 Prueba 4: Demostración del modelo de energía

Esta prueba se realiza con el objetivo de evaluar el consumo de energía de los dispositivos y el consumo de energía adicional requerido para realizar operaciones criptográficas complejas. Esta prueba permitirá determinar la cantidad de energía que se consume durante las operaciones criptográficas, así como evaluar la eficacia del algoritmo de ECC.

Los datos se obtienen realizando una grabación a los procesos durante la simulación y se analizan abriendo los archivos . $\boldsymbol{vec}$, **Figura 126** disponibles en la carpeta "**result**" del mismo proyecto.

Figura 126

Archivos para el análisis de resultados del consumo de energía

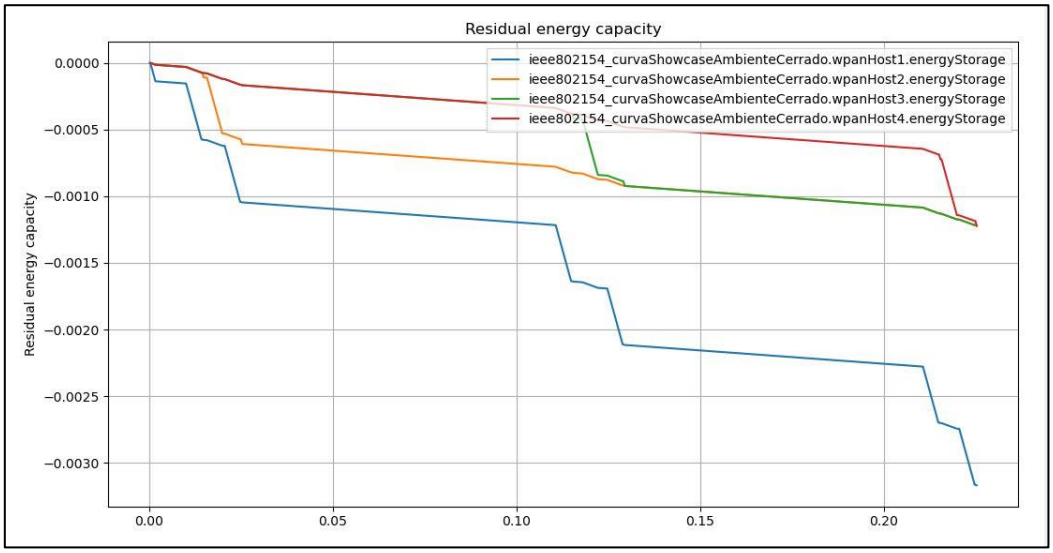


4.4.1 Ambiente Cerrado

Se realiza un análisis visual y comparativo con los datos anteriores La **Figura 127** indica el gasto de los nodos que participan en el despliegue monitorización en ambiente cerrado. El **eje horizontal** establece el *tiempo de simulación (s)* y el **eje vertical** *la energía consumida (J).* La figura indica la caída en el consumo de cada uno de los "**wpanHost***", estos datos pertenecientes a cada uno de los nodos sin algoritmo. Las caídas pronunciadas en el grafico representan mayor actividad en el nodo, además el nodo principal o "**wpanHost1**" es el que realiza un gato mayor de energía debido a la recepción de datos.

Figura 127

Grafica de consumo de energía para cada uno de los "wpanHost", en la red monitorización en ambiente cerrado – Sin el Algoritmo

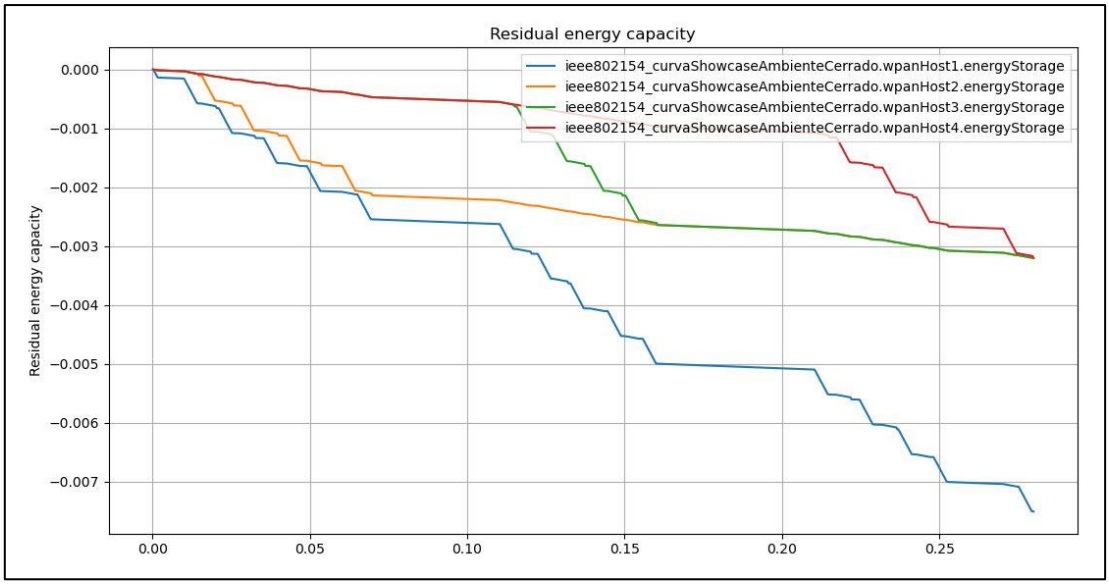


Nota: La figura muestra cómo cambia el consumo de energía en cada uno de los nodos de la red, es importante tener en cuenta que el nodo "**wpanHost1**" es quien tiene un mayor consumo energético, mientras que los otros nodos al enviar la misma cantidad de datos tienen un consumo igual y menor al nodo principal.

La **Figura 128** indica el gasto de los nodos que participan en el despliegue monitorización en ambiente cerrado, estos datos son pertenecientes a los nodos con el algoritmo, el grafico indica la caída en el consumo de cada uno de los "**wpanHost***"; A comparación de la **Figura 127** el consumo es mayor comprobando la existencia de más procesos debido a la implementación de criptográfica con curva elíptica, aun asi se mantiene el patrón donde el nodo "**wpanHost1**" supera a los demás nodos debido a la recepción de datos, del grafico se destaca **3** caídas pronunciadas correspondientes a la actividad de cada uno de los nodos sensores junto con el nodo principal.

Figura 128

Grafica de consumo de energía para cada uno de los "wpanHost", en la red monitorización en ambiente cerrado – con el Algoritmo



Para una mejor interpretación la **Tabla 24** compara los consumos de energía; Concluyendo así que el aumento en el consumo es pequeño pero notable en cada uno de los nodos, lo que sugiere que la implementación del algoritmo ECC tiene un impacto en la red. Indicando que puede ser significativo a largo plazo, especialmente si se considera que la red puede tener muchos nodos.

Los valores en la **Tabla 24** están expresados en **Joules**[24] **(J)**, para calcular la potencia consumida (**W**) se necesita conocer el tiempo de duración de la simulación ya que $\boldsymbol{Potencia}\ (\boldsymbol{W}) = \boldsymbol{Energía}\ (\boldsymbol{J})\ /\ \boldsymbol{Tiempo}\ (\boldsymbol{s})$, el tiempo de la simulación es $\boldsymbol{0.284512657091\ s}$ obteniendo asi los valores de la columna "**diferencia en mW"**.

__Tabla 24__. Consumo de energía de los nodos en un ambiente cerrado

Nodo sensor	***Consumo de energía sin el Algoritmo [J]***	***Consumo de energía con el Algoritmo ECC [J]***	***Diferencia en [J]***	***Diferencia en [mW]***
wpanHost1	0.008558	0.008782	0.00244	8.57
wpanHost2	0.007455	0.008161	0.00706	24.81
wpanHost3	0.007455	0.008050	0.0063	22.144
wpanHost4	0.007569	0.008081	0.00512	17.99

[24] Joule es una unidad de medida para la energía eléctrica consumida o generada en un circuito eléctrico en un período de tiempo.

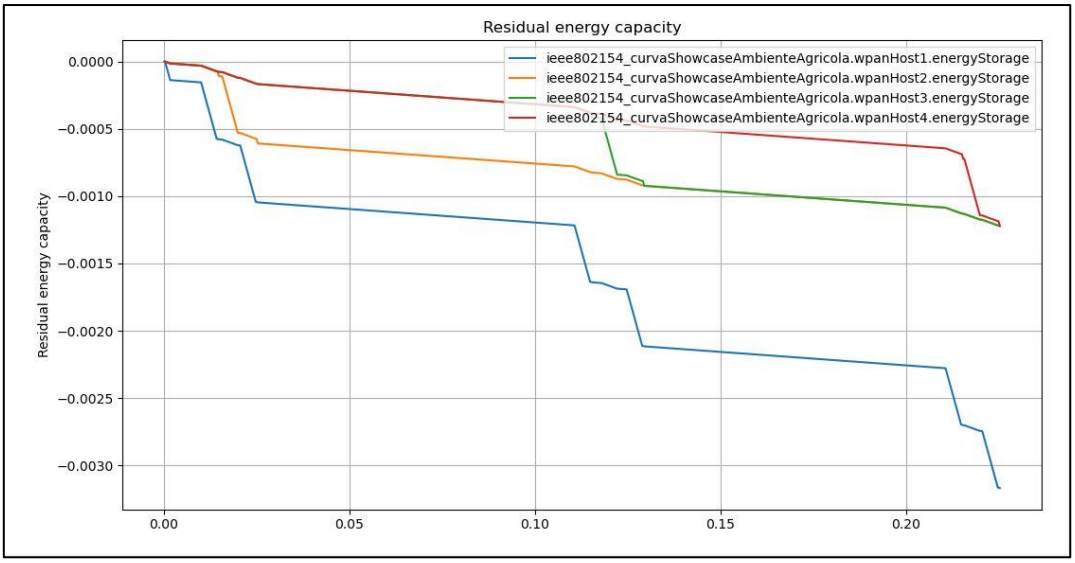
Realizando un análisis a la tabla se puede apreciar que existe una diferencia de $\mathbf{8,57}\ \boldsymbol{mW}$ entre implementar o no criptografía en el nodo principal, este valor se consideraría relativamente bajo, además el valor para los nodos sensores ronda entre los $\mathbf{20}\ \boldsymbol{mW}$, este valor tambien se considera bajo y este aumento es aceptable dentro del contexto de redes de sensores inalámbricos.

4.4.2 Ambiente Agrícola

La red diseñada para este ambiente utiliza la curva con más coste computacional, los cálculos de procesamiento del nodo no se toman en cuenta para el modelo de energía, impidiendo asi que se pueda obtener un valor real de consumo energético, tan solo una aproximación basada en los gastos de energía que realizan los nodos al enviar los datos; La **Figura 129** muestra los datos pertenecientes de los nodos sin el algoritmo para cada uno de los "**wpanHost***", la figura es similar al anterior escenario ya que se realiza una prueba con las mismas consideraciones físicas de los nodos, el cambio significativo se realiza al estudiar la implementación de la criptografía.

Figura 129

Grafica de consumo de energía para cada uno de los "wpanHost", en la red monitorización en ambiente agrícola – Sin Algoritmo

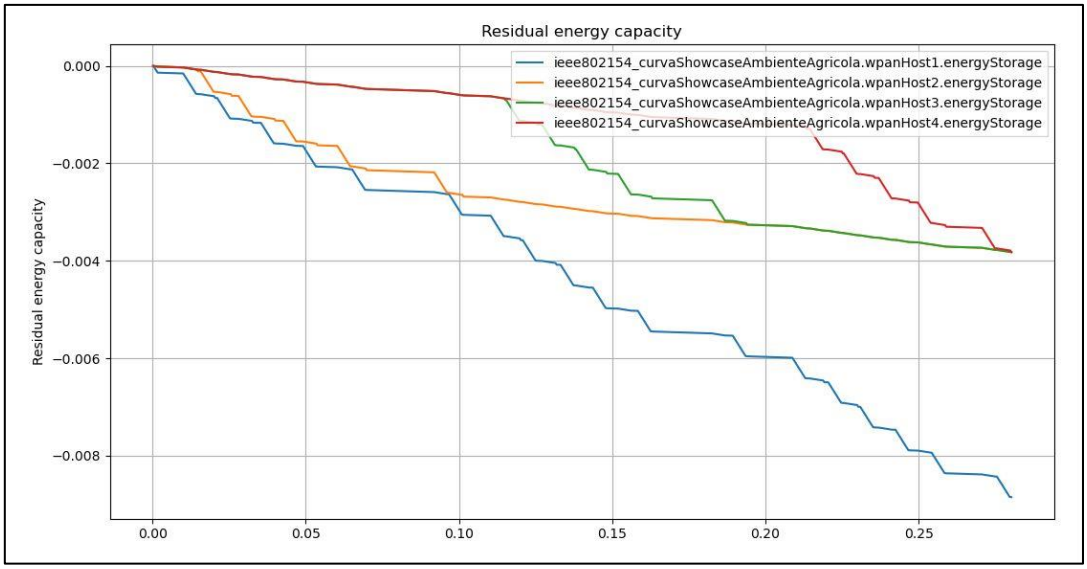


La **Figura 130** muestra los datos pertenecientes de los nodos con el algoritmo para cada uno de los "**wpanHost***". La implementación de la criptografía en este escenario hace que exista

un mayor tiempo de simulación asi como mayor consumo dentro de la red. En el grafico se observa como el nodo principal "**wpanhost1**" tiene pocos intervalos de descanso ya que recepta los datos de los demás nodos sensores y a la vez realiza el proceso de descifrado de datos.

Figura 130

Grafica de consumo de energía para cada uno de los "wpanHost", en la red monitorización en ambiente agrícola – Con Algoritmo



Para una mejor interpretación la **Tabla 25** compara los consumos de energía; Concluyendo asi que el aumento en el consumo es pequeño pero notable en cada uno de los nodos, lo que sugiere que la implementación del algoritmo ECC tiene un impacto en la red. El tiempo de simulación es $\mathbf{0.279889584491}\ \boldsymbol{s}$

Tabla 25*. Consumo de energía de los nodos en ambiente agrícola*

Nodo sensor	***Consumo de energía sin el Algoritmo [J]***	***Consumo de energía con el Algoritmo ECC [J]***	***Diferencia en [J]***	***Diferencia en [mW]***
wpanHost1	0.008558	0.008876	0.00318	11.36
wpanHost2	0.007455	0.008161	0.00706	25.22
wpanHost3	0.007455	0.007972	0.00517	18.47
wpanHost4	0.007569	0.008113	0.00544	19.43

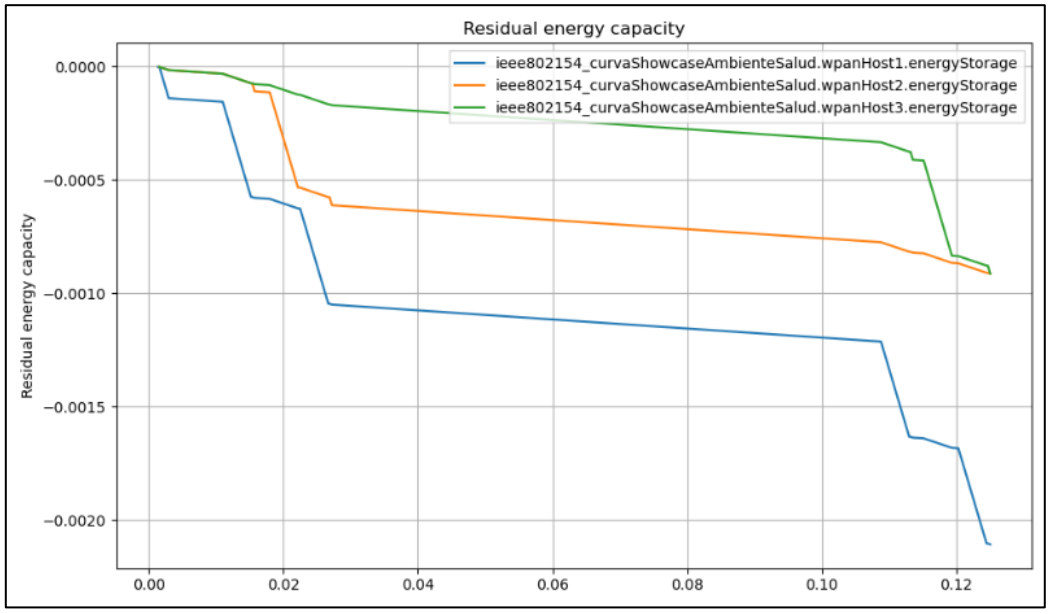
Realizando un análisis a la **Tabla 25** se puede apreciar que existe una diferencia de $\mathbf{11,36}\ \boldsymbol{mW}$ entre implementar o no criptografía en el nodo principal, además el valor para los nodos sensores ronda entre los $\mathbf{20}\ \boldsymbol{mW}$, este valor tambien se considera bajo y el consumo es similar a la **Tabla 24** este aumento es aceptable dentro del contexto de redes de sensores inalámbricos enfocadas a la agricultura donde no se necesita priorizar el envio de datos.

4.4.3 Ambiente Salud

La red diseñada para este ambiente utiliza la curva con menos coste computacional, a continuación, se realiza el análisis de la aproximación basada en los gastos de energía que realizan los nodos al enviar los datos. La **Figura 131** muestra los datos pertenecientes de los nodos sin el algoritmo para cada uno de los "**wpanHost***", la figura es diferente a los otros escenarios y aunque comparten las mismas características físicas, la diferencia se encuentra en la cantidad de nodos que pertenecen a la red, el grafico muestra como existen **2** caídas pronunciadas correspondiendo con la actividad de los **2** nodos al enviar datos al nodo principal.

Figura 131

Consumo de energía de los nodos sin el algoritmo

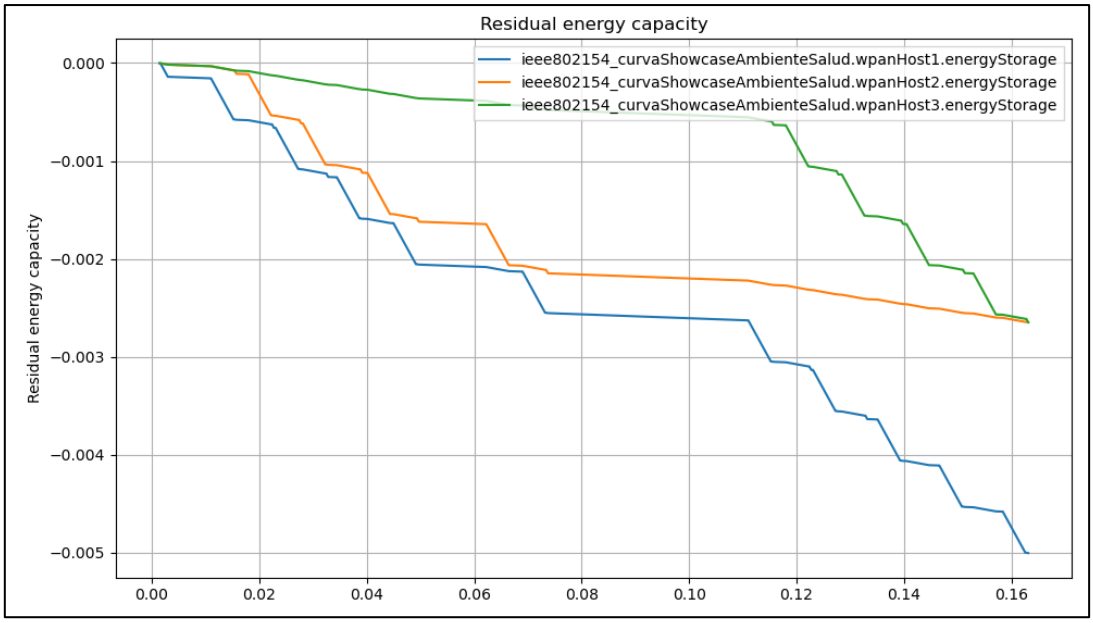


La **Figura 132** muestra los datos pertenecientes de los nodos con el algoritmo para cada uno de los "**wpanHost***". El consumo de energía es inferior a los demás escenarios asi como el

tiempo de simulación; Los procesos de cifrado y descifrado con la curva elíptica aplicada a este escenario requieren de poco tiempo del sistema, haciendo que los eventos se produzcan en menos tiempo.

Figura 132

Consumo de energía de los nodos con el algoritmo



Para una mejor interpretación la **Tabla 26** compara los consumos de energía en el tiempo de $\mathbf{0.163102150878\ s}$, La columna "**Diferencia en mW**" nos indica que el consumo para el nodo principal es de $\mathbf{27.95\ mW}$, este valor difiere de los demás escenarios ya que en estos el valor promedio ronda entre los $\mathbf{10\ mW}$, además la diferencia para los nodos sensores es alrededor de $\mathbf{45\ mW}$ siendo el escenario con mayor consumo de energía al implementar el algoritmo, para implementar la criptografía en este tipo de escenarios es necesario realizar un análisis exhaustivo del consumo de energía de los dispositivos a implementar ECC.

Tabla 26*. Consumo de energía de los nodos en ambiente salud*

Nodo sensor	***Consumo de energía sin el Algoritmo [J]***	***Consumo de energía con el Algoritmo ECC [J]***	***Diferencia en [J]***	***Diferencia en [mW]***
wpanHost1	0.008514	0.008970	0.00456	27.95
wpanHost2	0.007714	0.008468	0.00756	46.35
wpanHost3	0.007855	0.008552	0.00697	42.73

4.5 Prueba 5: Análisis del cálculo de procesamiento

Esta prueba tiene como objetivo conocer los tamaños de clave de cada una de las curvas implementadas en la simulación; La longitud de la clave determina la cantidad de bits utilizados para cifrar los datos y, por lo tanto, su fuerza criptográfica. Un tamaño de clave más pequeño hace que sea más fácil para un atacante romper la clave mediante técnicas de fuerza bruta o mediante la utilización de algoritmos criptográficos débiles.

La prueba consta de la comunicación entre un trasmisor y un receptor, específicamente el nodo trasmisor enviara los datos cifrados al receptor usando ECC. Se utiliza los protocolos **ECDH** y **ElGamal Eliptico**; Los datos a enviar son 3 y en cada uno se utiliza una diferente contraseña. El script para la ejecución del código se encuentra en el **ANEXO C – Script para el análisis de procesamiento del sistema ECC**.

El código calcula el número de ***ciclos de reloj*** que se necesitaron para realizar el proceso de cifrado de 3 datos. Esto se hace mediante la fórmula: $tiempo * frecuencia$ / 1000. Además, se calcula la cantidad de ***operaciones por segundo*** realizadas por el procesador, utilizando la fórmula: $frecuencia * 10\text{^}6 / ciclos\_reloj$. Y finalmente se calcula el número de ***MIPS*** (Millones de Instrucciones Por Segundo) que realiza el procesador, usando la siguiente formula $operaciones\ por\ segundo$ / 10^6.

4.5.1 Ambiente Cerrado

Para este escenario, se utiliza la curva indicada en la sección **WSN para la monitorización en ambientes cerrados.** Los valores se ingresan en la ejecución del script, el cual lleva a cabo el proceso de intercambio de claves y la generación de una clave común "**ECDH**", además se realiza el cifrado y descifrado de 3 datos usando "**ElGamal Elíptico**" y se obtienen los resultados indicados en la **Tabla 27**, aquí se puede apreciar que la cantidad de **MIPS** es de 3.1 siendo una cifra muy buena para la implementación en sensores inalámbricos, en la sección **Comparación entre los escenarios** se realiza un análisis detallado sobre los **MIPS** y el hardware empleado para WSN.

Tabla 27. Operaciones por segundo para la WSN " ambiente cerrado"

Frecuencia del procesador	*Tiempo de ejecución procesador*	*Ciclos de reloj*	*Operaciones por segundo*	*MIPS*
2.5 GHz	0.2314 s	555453840	31257691	3.125

La **Figura 133** detalla los tamaños de clave que ocupan todas las variables que intervienen en el proceso de cifrado y descifrado, las más importantes a destacar son: para el mensaje cifrado "**MsjCifrado**" con un tamaño de **480 bytes**, y para tamaño de clave "**SA**" con un tamaño de **16 bytes**. El mensaje cifrado tiene una variable de **tipo double** de tamaño **30x2** (30 filas x dos columnas) que almacena el mensaje cifrado. La clave también se almacena en una variable de **tipo double** de tamaño (**1x2**) representando un vector de dos valores numéricos.

Figura 133

Tamaño de clave para los parámetros de la WSN "ambiente cerrado"

```
*  Limited by System Memory (physical + swap file) available.
  Name                Size             Bytes  Class     Attributes

  EnteroxmedioA       1x1                  8  double
  EnteroxmedioB       1x1                  8  double
  G                   1x2                 16  double
  K                   1x1                  8  double
  MsjCifrado         30x2                480  double
  QA                  1x2                 16  double
  QB                  1x2                 16  double
  SA                  1x2                 16  double
  SB                  1x2                 16  double
  a                   1x1                  8  double
  b                   1x1                  8  double
  dA                  1x1                  8  double
  dB                  1x1                  8  double
  data1               1x30                60  char
  data2               1x30                60  char
  data3               1x30                60  char
  msj                 1x30                60  char
  msjDescifrado       1x30                60  char
  n                   1x1                  8  double
  p                   1x1                  8  double
```

Finalmente, la **Figura 134** muestra la información detallada acerca del tiempo de ejecución de las funciones, incluyendo el tiempo total de ejecución y el tiempo de ejecución de cada línea de código dentro de la función, permitiendo identificar cuellos de botella o áreas de mejora en el código. En este caso las funciones con más tiempo de procesamiento relacionadas con ECC

son: "**sumaPuntosCurvaEliptica**" la cual es la función más importante que interviene en todas las etapas del sistema y "**gcd**" esta función se utiliza para encontrar el máximo común divisor y está relacionada con la representación de texto a puntos y viceversa.

Figura 134

Análisis de las funciones utilizadas en script "ambiente cerrado"

Function Name	Calls	Total Time (s)	Self Time* (s)	Total Time Plot (dark band = self time)
isprime	514	0.034	0.017	
primes	514	0.016	0.016	
sumaPuntosCurvaElipticaModp	334	0.083	0.017	
charToPuntoEcc	90	0.014	0.006	
puntoEccToChar	90	0.007	0.003	
gcd	326	0.032	0.032	
AEE	326	0.008	0.008	
multEscalarCurvaElipticaModp	8	0.047	0.003	
cifradorElGamalEcc2	3	0.039	0.008	
descifradorElGamalEcc2	3	0.039	0.010	
generadorClavesEcc	2	0.028	0.002	
Mips	1	0.211	0.085	
expRapida	166	0.003	0.003	

4.5.2 Ambiente Salud

Para este escenario se utiliza la curva indicada en la sección **WSN para la monitorización en ambiente salud** los valores se ingresan en la ejecución del script y se obtienen los siguientes resultados.

Frecuencia del procesador	***Tiempo de ejecución procesador***	***Ciclos de reloj***	***Operaciones por segundo***	***MIPS***
2.5 GHz	0.2314 s	555453840	4320791	4.320

En la **Figura 135** se proporcionan detalles sobre los tamaños de clave y variables relacionadas con el proceso de cifrado y descifrado. Se destaca la presencia de dos variables importantes: "**MsjCifrado**", que corresponde al mensaje cifrado y ocupa **480 bytes**, y "**SA**", que representa

el tamaño de clave y ocupa **16 bytes**. Los tamaños de las variables donde se almacenan los mensajes a transmitir y las claves son similares al escenario anterior, ya que las curvas no distan mucho, es decir son similares.

Figura 135

Tamaño de clave para los parámetros de la WSN "ambiente salud"

```
*  Limited by System Memory (physical + swap file) available.
  Name                Size              Bytes  Class     Attributes

  EnteroxmedioA       1x1                   8  double
  EnteroxmedioB       1x1                   8  double
  G                   1x2                  16  double
  K                   1x1                   8  double
  MsjCifrado         30x2                 480  double
  QA                  1x2                  16  double
  QB                  1x2                  16  double
  SA                  1x2                  16  double
  SB                  1x2                  16  double
  a                   1x1                   8  double
  b                   1x1                   8  double
  dA                  1x1                   8  double
  dB                  1x1                   8  double
  data1               1x30                 60  char
  data2               1x30                 60  char
  data3               1x30                 60  char
  msj                 1x30                 60  char
  msjDescifrado       1x30                 60  char
  n                   1x1                   8  double
  p                   1x1                   8  double
```

La **Figura 136** muestra todas las funciones implementadas, así como la cantidad de llamadas realizadas a estas, las más implementas coinciden con el escenario anterior siendo estas: La función "**sumaPuntosCurvaEliptica**" la cual juega un papel fundamental en el procesamiento de la curva elíptica, ya que permite realizar la suma de dos puntos en dicha curva. Esta función es esencial para la mayoría de las operaciones criptográficas con EC. Además, la función "**isprime**" es utilizada para verificar si un valor ingresado en el campo es un número primo. Finalmente, la función "**gcd**" es empleada para encontrar el máximo común divisor entre dos números, y es utilizada en la representación de los mensajes a puntos.

Figura 136

Análisis de las funciones utilizadas en la WSN "ambiente salud"

Generated 02-jun.-2023 5:33:20 using performance time.

Function Name	Calls	Total Time (s)	Self Time* (s)	Total Time Plot (dark band = self time)
isprime	478	0.032	0.017	
primes	478	0.015	0.015	
sumaPuntosCurvaElipticaModp	298	0.073	0.019	
charToPuntoEcc	90	0.017	0.009	
puntoEccToChar	90	0.009	0.005	
gcd	290	0.026	0.026	
AEE	290	0.006	0.006	
multEscalarCurvaElipticaModp	8	0.051	0.006	
cifradorElGamalEcc2	3	0.048	0.016	
descifradorElGamalEcc2	3	0.038	0.016	
generadorClavesEcc	2	0.031	0.002	

4.6 Comparación entre los escenarios

Cada uno de los escenarios utiliza una diferente curva elíptica, así como diferente cantidad de dispositivos y condiciones propias para cada ambiente, en la **Tabla 28** se detalla los aspectos utilizados, así como los resultados obtenidos. Estos datos nos proporcionan información sobre el rendimiento y características de cada configuración de la curva elíptica en términos de tiempo, consumo de energía, tamaños de clave y mensajes. También se incluye la métrica MIPS la cual indica la cantidad de instrucciones de coma flotante por segundo que el sistema es capaz de ejecutar.

***Tabla 28**. Parámetros y comparación entre los escenarios y sus diferentes curvas elípticas*

Parámetros	***Ambiente Cerrado***	***Ambiente Agrícola***	***Ambiente Salud***
Parámetros de la curva	$a = -3, b = 1$	$a = -3, b = 1$	$a = 10, b = -10$
Tamaño del campo	10859	937127	3851
Punto Generador	(100,4651)	(1912,711700)	(110 ,345)
Tiempo de simulación	0.22143 [s]	0.27988 [s]	0.16310 [s]
Consumo de energía	≈ 20 [mW] c/n	≈ 20 [mW] c/n	≈ 40 [mW] c/n
Tamaño de la clave	16 [bytes]	32 [bytes]	16 [bytes]
Tamaño del mensaje a transmitir	60 [bytes]	60 [bytes]	60 [bytes]

Tamaño del mensaje cifrado	480 [bytes]	960 [bytes]	480 [bytes]
Cantidad de llamados a la función Suma de Puntos	334	492	298
Tiempo de la función Suma de puntos	0,016 [s]	0,023 [s]	0,015 [s]
MIPS	3.125	6.331	4.320

En este sentido, se realiza una comparación de acuerdo con los MIPS de cada WSN, tomando como referencia el hardware utilizado para estas redes. Primero el "**microcontrolador Atmega-328**" es un dispositivo de la familia AVR de Atmel y tiene una *velocidad de reloj* de hasta **20 MHz**. Según la hoja de especificaciones técnicas de Atmel, es capaz de ejecutar **1 MIPS** por cada MHz de velocidad de reloj. Por lo tanto, si el Atmega-328 está funcionando a su velocidad máxima de **20 MHz**, puede ejecutar hasta **20 MIPS**.

Así también otro ejemplo es "**el módulo XBee de Digi International**" utiliza un chip de la familia ARM Cortex-M3, que tiene una *velocidad de reloj* de hasta **64 MHz**. Según la hoja de datos de ARM, el Cortex-M3 es capaz de ejecutar **1.25 MIPS** por cada MHz de velocidad de reloj. Por lo tanto, si el chip del módulo XBee está funcionando a su velocidad máxima de **64 MHz**, puede ejecutar hasta **100 MIPS**.

La cantidad real de MIPS que puede lograr estos dispositivos dependerá del código específico que se esté ejecutando, al implementar la librería ECC, los datos obtenidos pueden cambiar, no obstante, cada ambiente anteriormente probado funcionará correctamente en cada hardware.

Conclusiones y Recomendaciones

Para culminar la presente investigación, se lleva a cabo la sección de conclusiones y recomendaciones, la cual permitirá detallar los resultados obtenidos, las soluciones e inconvenientes encontrados, así como las contribuciones realizadas durante el desarrollo del proyecto propuesto. Además, se proporcionarán recomendaciones para futuros estudios en el ámbito profesional y académico.

Conclusiones

- Durante el desarrollo de este Trabajo de Titulación, se logró exitosamente la implementación del algoritmo de cifrado basado en curvas elípticas, diseñado específicamente para su implementación en una red de sensores inalámbricos (WSN). Este enfoque criptográfico aprovecha operaciones matemáticas complejas, como la suma y multiplicación en curvas elípticas, además del intercambio de claves ECDH y el sistema de cifrado ElGamal elíptico.

- Los requerimientos necesarios para el funcionamiento del algoritmo de cifrado de criptografía de curva elíptica en las WSN han sido definidos de manera precisa. Estos requerimientos establecen las condiciones y características que deben cumplirse para garantizar una implementación efectiva y segura del algoritmo en entornos de redes de sensores inalámbricos.

- Los parámetros de dominio requeridos para el funcionamiento del algoritmo incluyen, obtener un valor optimo del cofactor, el cual desempeña un papel crucial en la determinación del nivel de seguridad proporcionado por una curva elíptica. Este valor influye en la cantidad de puntos utilizados en la criptografía, ya que su cercanía a **1** es determinante. Si el cofactor de la curva es igual a uno, se utilizarán todos los puntos de la curva para la criptografía. Por otro lado, si el cofactor es igual a dos, se empleará la mitad de los puntos, y así sucesivamente.

- Dado que la generación de todos los puntos de las curvas elípticas es un proceso computacionalmente costoso, es importante limitar el campo primo de dichas curvas. En otras palabras, se busca aumentar la seguridad de la información sin comprometer

los tiempos de ejecución ofrecidos por los sistemas, manteniéndolos en niveles aceptables.

- Normalmente, el algoritmo ElGamal elíptico cifra un mensaje completo en un único punto de la curva, ya que los campos finitos son demasiado extensos. No obstante, al implementar este algoritmo con el objetivo de utilizarlo en dispositivos de baja potencia, se decidió representar cada carácter del mensaje como un punto en la curva. De esta manera, se logra una representación más eficiente y adecuada para dispositivos con limitaciones de rendimiento.

- La criptografía de curvas elípticas (ECC) se destaca al ofrecer una ventaja significativa para la implementación de aplicaciones seguras en redes de sensores inalámbricos (WSN). La capacidad de reducir el tamaño de las claves en ECC se vuelve especialmente beneficiosa en este escenario. Esto se debe a que el tamaño reducido de las claves contribuye a optimizar los recursos limitados de los nodos sensores, permitiendo una implementación más eficiente y segura en las WSN.

Recomendaciones

- El estándar para la eficiencia criptográfica sobre curvas elípticas **SEC1.v2**. es de mucha ayuda para comprender las primitivas criptográficas es decir toda la parte matemática que interviene en los procesos de criptografía, aunque la matemática a implementar es complicada ya que intervienen conceptos como aritmética modular, campos finitos, inverso de un punto, etc. El **SEC1.v2** indica el proceso de estos cálculos de manera detalla.

- El uso de un sistema de control de versiones se recomienda como una práctica fundamental para garantizar una gestión eficiente de los cambios en el código, facilitando la detección y resolución de errores, así como la colaboración en proyectos de desarrollo de aplicaciones.

- Cuando se manejan datos de gran sensibilidad o valor durante la transmisión, se puede considerar el uso de longitudes de clave más largas para aumentar aún más la seguridad. Para lograr esto, es posible aumentar el tamaño del campo Z_p al buscar una curva elíptica que genere una mayor cantidad de puntos. Asimismo, se puede seleccionar un punto generador que abarque todos los puntos de la curva, lo que resulta en un cofactor igual a 1.

- Para brindar una protección adicional a los datos transmitidos, se recomienda implementar el uso de ECC en conjunto con otros protocolos y técnicas de seguridad, como la autenticación y el cifrado de extremo a extremo. Además, es fundamental llevar a cabo pruebas exhaustivas de seguridad y realizar actualizaciones periódicas para garantizar la efectividad de la implementación de ECC y otros protocolos de seguridad en la prevención de posibles amenazas y vulnerabilidades en la red de sensores.

Trabajos Futuros

A lo largo de este trabajo se ha desarrollado la base matemática necesaria para la implementación de curvas elípticas en cualquier tipo de redes, pero enfocadas a dispositivos de bajo rendimiento. Sin embargo, existen varias líneas de trabajo futuro que se podrían expandir, haciendo aún más el alcance de este proyecto. Algunas de estas líneas de trabajo incluyen:

Ampliando el banco de pruebas de los algoritmos criptográficos se puede verificar la corrección de estos de manera más exhaustiva. Esto incluiría la realización de pruebas adicionales que examinen diferentes escenarios y situaciones, como datos de entrada extremos, casos límite y situaciones de error; también asegurarse de que siempre los resultados sean consistentes en comparación con el trabajo de otros autores.

Implementar los algoritmos de ataque al logaritmo discreto es una excelente manera de ampliar la funcionalidad de la librería criptográfica. Permitiría a los usuarios experimentar con los algoritmos utilizados por atacantes para intentar romper la seguridad de los sistemas criptográficos y comprender la importancia de elegir parámetros de seguridad adecuados.

Implementar los algoritmos de firma digital con curvas elípticas que permitan la autenticación y la verificación de integridad de los datos transmitidos en una red WSN. La implementación de estos algoritmos es importante para asegurar que los datos transmitidos no han sido modificados y que provienen de una fuente confiable.

Estas son solo algunas de las posibles líneas de trabajo futuro para la expansión de este proyecto. Con el tiempo y los recursos adecuados, la librería criptográfica podría crecer y desarrollarse aún más para satisfacer las necesidades de una amplia variedad de usuarios y aplicaciones.

BIBLIOGRAFÍA

Bahillo, L. (2021). Historia de Internet: cómo nació y cuál fue su evolución. M4rketing Ecommerce, 3-4.

Caiza, J. C. (2020). Desarrollo de una aplicaccion intercativa utilizando algoritmos criptograficos basados en curvas elipticas. Quito: ESCUELA POLITÉCNICA NACIONAL.

Carrión, E. (2016). Sistema de monitoreo de monóxido de carbono mediante una red de sensores inalámbricos y una plataforma como servicio en la nube para una residencia. Ibarra: Universidad Tecnica del Norte.

Cuzme-Rodríguez, F. U.-C.-Z.-E.-L.-V. (2020). Simulation Tools for Solving Engineering Problems. Case Study. In Applied Technologies: First International Conference. Quito, Ecuador.

Forero, E. B. (2017). Seguridad en redes. Bogotá D.C.: Universitaria del Área Andina.

Garcia, J. M. (2010). Capítulo 8: Criptografía de curvas elípticas. Morelia: ITC del ITESM.

Mendoza, J. C. (2019). Demostracion de Cifrado Cimetrico y Asimetrico. Quito: INGENIUS.

Palacios, J. (2017). Diseño de una red de sensores (WSN) con tecnología 802.15.4, basado en el concepto agricultura de precisión para el control y monitoreo de cultivos de hortalizas bajo invernadero en la granja la pradera de la Universidad Técnica del Norte. Ibarra: Universidad Técnica del Norte.

Pineda, S. (2021). Medidor de temperatura y saturación de niveles de oxígeno en sangre para la monitorización de pacientes diagnosticados por Covid-19 y su cerco epidemiológico, mediante el uso de Algoritmos de aprendizaje de máquina, con el uso de Aplicaciones del IoT. Ibarra: Universidad Técnica del Norte.

Stallings, W. (1997). Comunicaciones y Redes de Computadoras. Prentice Hall: Iberia.

Stallings, W. (2005). Cryptography and Network Security Principles and Practices, Fourth Edition. United States of America: Pearson Education.

Three Points. (24 de 08 de 2021). ThreePoints the school for digital susiness. Obtenido de introducción a la Criptografía: https://www.threepoints.com/blog/breve-introduccion-a-la-criptografia#:~:text=La%20criptograf%C3%ADa%20es%20la%20t%C3%A9cnica,que%20los%20mensajes%20sean%20confidenciales.

El Sevier. (2005)" Koblitz curve cryptosystems" Information-Security and Cryptography, Ruhr-University of Bochum, Universitätsstrasse 150, D-44780 Bochum, Germany.

Castang G. Barbosa C. (2011) "Implementación del criptosistema de curva elíptica en entornos móviles" VINCULOS.

Castang G. Barbosa C. Tibaquira Y. (2011) "Implementación del criptosistema de curva elíptica en un prototipo de aplicación móvil para E-Commerce" Revista Tekhnê 2011, Vol. 8, 5–12

Iacono L. Godoy P. Marianetti O. García C. (2010)" Estudio de Plataformas de Hardware Empleadas en Redes de Sensores Inalámbricas" CORE.

Ahamed T. (2019) "Internet of Things: A Comprehensive Study of Security Issues and Defense Mechanisms" IEEE Access.

Alcaraz C. Román R. López J. (2008) "Análisis de la Aplicabilidad de las Redes de Sensores para la Protección de Infraestructuras de Información Críticas"

Moghadam M. Nikooghadam M. Baqer M. Alishahi M. Mortazavi L. Mohajerzadeh A. (2020) "An Efficient Authentication and Key Agreement Scheme Based on ECDH for Wireless Sensor Network" IEEE Access.

Guaña J. (2020) "Desarrollo de una aplicación interactiva utilizando algoritmos criptográficos basados en curvas elípticas" "ESCUELA POLITÉCNICA NACIONAL.

Manuel J. López L (2022) "Criptografía y Seguridad en Computadores" Creative Commmons.

Martínez R. Ordieres J. Martínez de Pisón F. González A. Elías F. Lostado R. Pernía A. (2009) "Redes Inalámbricas de Sensores: Teoría y Aplicación Práctica" UNIVERSIDAD DE RIOJA.

Chinchay I. Kaschel H. Abarzua R. (2014)" Implementación de una curva elíptica aplicada a una WSN limitada en hardware" Departamento de Electrónica y automática, Facultad de Ingeniería, Universidad de Piura.

Bekkali A. Boulmalf M. Essaaidi M. Mezzour G. (2018) "Securing the Internet of Thing (IoT)" IEEE.

Ferrández F. (2005) "Sistemas criptográficos de curva elíptica basados en matrices" [Tesis doctoral de la Universidad de Alicante]

Henriques M. (2017) "Using symmetric and asymmetric cryptography to secure communication between devices in IoT" Departmentof Computer Engineering.

Chunqiang H. Yuwen P. Feihong Y. Ruifeng Z. Arwa A. Tao X. (2020) "Secure and Efficient Data Collection and Storage of IoT in Smart Ocean" IEEE INTERNET OF THINGS JOURNAL.

Young T. (2015) "Task Scheduling Algorithm to Reduce the Number of Processors using Merge Conditions" International Journal on Computer Science and Engineering (IJCSE).

Daniel R. (2009) "SEC 1: Elliptic Curve Cryptography" Certicom Research, "Standards for Efficient Cryptography 1".

Daniel R. (2010) "SEC 2: Recommended Elliptic Curve Domain Parameters" Certicom Research, "Standards for Efficient Cryptography 2".

Stallings W. (2005) "Cryptography and Network Security" Principles and Practices, Fourth Edition.

Delgadillo E. (2008) "Diseño de un criptosistema para redes de sensores inalámbricos WSN basado en MPSOC" Bogota, Colombia [UNIVERSIDAD DE LOS ANDES FACULTAD DE INGENIERIA DEPARTAMENTO DE INGENIERIA ELÉCTRICA Y ELECTRÉNICA]

Santamarín K. Álvarez K. (2018) "Diseño e implementación de una red con sensores inalámbrica (WSN) con un protocolo abierto de comunicación basado en IEEE 802.15.4 (XBEE) para prácticas" UNIVERSIDAD POLITÉCNICA SALESIANA.

caortegac@utn.edu.ec

ANEXO A

FICHA DE REQUERIMIENTOS

Objetivo de la ficha: Establecer y mantener un acuerdo con los involucrados en el sistema, proporcionar bases para que el desarrollador tenga un mejor entendimiento de los requerimientos del sistema.

TEMA: "IMPLEMENTACIÓN DE UN MECANISMO DE SEGURIDAD PARA REDES DE SENSORES INALÁMBRICOS BASADO EN CRIPTOGRAFÍA DE CURVAS ELÍPTICAS"

Fecha inicio: 2022-10-03

Fecha final: 2022-10-21

NOMENCLATURA DE REQUERIMIENTOS:

Requerimiento	*Nomenclatura*
De Stakeholders	**StSR**
De Sistema	**SySR**
De Arquitectura	**SrSH**

caortegac@utn.edu.ec

Lista de artículos revisados:

- El Sevier. (2005) “Koblitz curve cryptosystems” Information-Security and Cryptography, Ruhr-University of Bochum, Universität Strasse 150, D-44780 Bochum, Germany.
- Castang G. Barbosa C. Tibaquira Y. (2011) “Implementación del criptosistema de curva elíptica en un prototipo de aplicación móvil para E-Commerce” Revista Tekhnê 2011, Vol. 8, 5–12
- Iacono L. Godoy P. Marianetti O. García C. (2010)” Estudio de Plataformas de Hardware Empleadas en Redes de Sensores Inalámbricas” CORE.
- Ahamed T. (2019) “Internet of Things: A Comprehensive Study of Security Issues and Defense Mechanisms” IEEE Access.
- Alcaraz C. Román R. López J. (2008) “Análisis de la Aplicabilidad de las Redes de Sensores para la Protección de Infraestructuras de Información Críticas”
- Moghadam M. Nikooghadam M. Baqer M. Alishahi M. Mortazavi L. Mohajerzadeh A. (2020) “An Efficient Authentication and Key Agreement Scheme Based on ECDH for Wireless Sensor Network” IEEE Access.
- Guaña J. (2020) “Desarrollo de una aplicación interactiva utilizando algoritmos criptográficos basados en curvas elípticas” “ESCUELA POLITÉCNICA NACIONAL.
- Manuel J. López L (2022) “Criptografía y Seguridad en Computadores” Creative Commmons.
- Martínez R. Ordieres J. Martínez de Pisón F. González A. Elías F. Lostado R. Pernía A. (2009) “Redes Inalámbricas de Sensores: Teoría y Aplicación Práctica” UNIVERSIDAD DE La RIOJA.
- Chinchay I. Kaschel H. Abarzua R. (2014)” Implementación de una curva elíptica aplicada a una WSN limitada en hardware” Departamento de Electrónica y automática, Facultad de Ingeniería, Universidad de Piura.

- Bekkali A. Boulmalf M. Essaaidi M. Mezzour G. (2018) “Securing the Internet of Thing (IoT)” IEEE.
- Ferrández F. (2005) “Sistemas criptográficos de curva elíptica basados en matrices” [Tesis doctoral de la Universidad de Alicante]
- Henriques M. (2017) “Using symmetric and asymmetric cryptography to secure communication between devices in IoT” Departmentof Computer Engineering.
- Chunqiang H. Yuwen P. Feihong Y. Ruifeng Z. Arwa A. Tao X. (2020) “Secure and Efficient Data Collection and Storage of IoT in Smart Ocean” IEEE INTERNET OF THINGS JOURNAL.
- Young T. (2015) “Task Scheduling Algorithm to Reduce the Number of Processors using Merge Conditions” International Journal on Computer Science and Engineering (IJCSE).
- Daniel R. (2009) “SEC 1: Elliptic Curve Cryptography” Certicom Research, “Standards for Efficient Cryptography 1”.
- Daniel R. (2010) “SEC 2: Recommended Elliptic Curve Domain Parameters” Certicom Research, “Standards for Efficient Cryptography 2”.
- Stallings W. (2005) “Cryptography and Network Security” Principles and Practices, Fourth Edition.
- Delgadillo E. (2008) “Diseño de un criptosistema para redes de sensores inalámbricos WSN basado en MPSOC” Bogota, Colombia [UNIVERSIDAD DE LOS ANDES FACULTAD DE INGENIERIA DEPARTAMENTO DE INGENIERIA ELÉCTRICA Y ELECTRÉNICA]
- Santamarín K. Álvarez K. (2018) “Diseño e implementación de una red con sensores inalámbrica (WSN) con un protocolo abierto de comunicación basado en IEEE 802.15.4 (XBEE) para prácticas” UNIVERSIDAD POLITÉCNICA SALESIANA.

caortegac@utn.edu.ec

REQUERIMIENTOS DE STAKEHOLDERS:

Nomenclatura:	StSR 1
Requerimiento:	El algoritmo debe ser modificable para generar distintos tamaños de clave y tiempos de cifrado
Descripción:	Permitir al usuario modificar ciertos parámetros en las curvas elípticas, con el fin de que se adapten a la capacidad del Hardware. Notificar al usuario cuando una curva es apta o no para criptografía, el proceso debe ser amigable con el usuario, donde solo se tenga que cambiar parámetros iniciales y no algoritmos matemáticos.
Prioridad:	Media

Nomenclatura:	StSR 2
Requerimiento:	Establecer las pruebas de funcionamiento en tres escenarios de Simulación
Descripción:	Los escenarios donde se implemente el algoritmo ECC deben ser tres: 1. Para la monitorización ambiental en lugares cerrados, 2. para la agricultura, y 3. la monitorización de la salud. La red debe ser estable y funcionar adecuadamente en el envío y recepción de datos. Se debe comprobar la información del envió de tramas antes de implementar el algoritmo de encriptación.
Prioridad:	Alta

Nomenclatura:	StSR 3
Requerimiento:	Se debe presentar altos conocimiento en lenguajes de programación (C++ y MATLAB)
Descripción:	La simulación de redes de Sensores Inalámbricos en los diferentes simuladores, así como el desarrollo del algoritmo criptográfico, requiere un alto nivel de programación por lo tanto se necesita que el autor del presente trabajo esté capacitado.
Prioridad:	Alta

Nomenclatura:	StSR 4
Requerimiento:	El sistema debe ser interoperable de manera efectiva con otros sistemas

caortegac@utn.edu.ec

Descripción:	El sistema debe ser capaz de intercambiar información y trabajar de manera efectiva con otros sistemas. Es decir, tener la capacidad para comunicarse, compartir datos, recursos, y coordinar sus actividades de manera coherente y sin problemas.
Prioridad:	Baja

Nomenclatura:	StSR 5
Requerimiento:	Se debe demostrar conocimiento de Criptografía ECC y como se aplica a una red de sensores
Descripción:	El autor debe demostrar un alto conocimiento en el funcionamiento de la Criptografía de Curvas Elípticas, y como se deben aplicar a las redes de sensores inalámbricos, Además. Indicar cuales son los parámetros iniciales y como se implementa toda la criptografía sobre campos finitos muy grandes, si el Hardware lo soporta.
Prioridad:	Alta

Nomenclatura:	StSR 6
Requerimiento:	Verificación de la Encriptación / Desencriptación utilizando una comunicación TCP y la captura de datos
Descripción:	El autor debe verificar que los datos enviados se transportan de forma segura desde los nodos sensores hacia la puerta de enlace o nodo principal, la encriptación debe estar presente en la capa aplicación, y debe comprobarse mediante la comunicación TCP/IP y usando la captura de tramas.
Prioridad:	Alta

Nomenclatura:	StSR 7
Requerimiento:	El tipo de cifrado de datos puede ser cifrado de flujo o cifrado de bloque
Descripción:	Se puede elegir entre las dos técnicas de cifrado, el cifrado de flujo para aplicaciones en tiempo real y el cifrado de bloque para proteger datos que no requieren transmisión que se utilizan comúnmente es decir para proteger la confidencialidad de los datos.

caortegac@utn.edu.ec

Prioridad:	Baja

Nomenclatura:	StSR 8
Requerimiento:	El sistema debe tener la capacidad de manejar grandes volúmenes de datos
Descripción:	El sistema debe ser capaz de procesar y almacenar una gran cantidad de datos sin que esto afecte negativamente su rendimiento
Prioridad:	Baja

REQUERIMIENTOS DE STAKEHOLDERS – TABLA DE RESUMEN:

StSR ***REQUERIMIENTOS OPERACIONALES***				
No.	***Requerimiento***	***Prioridad***		
		Alta	***Media***	***Baja***
StSR 1	*El algoritmo debe ser modificable para generar distintos tamaños de clave y tiempos de cifrado*		*X*	
StSR 2	*Establecer las pruebas de funcionamiento en tres escenarios de Simulación*	*X*		
StSR 3	*Se debe presentar altos conocimiento en lenguajes de programación (C++ y MATLAB)*	*X*		
StSR 4	*El sistema debe ser interoperable de manera efectiva con otros sistemas*			*X*
StSR 5	*Se debe demostrar conocimiento de Criptografía EC y como se aplica está a una red de sensores*	*X*		
StSR 6	*Verificación de la Encriptación / Desencriptación utilizando una comunicación IP y la captura de datos*	*X*		
StSR 7	*El tipo de cifrado de datos puede ser cifrado de flujo o cifrado de bloque*			*X*
StSR 8	*El sistema debe tener la capacidad de manejar grandes volúmenes de datos*			*X*

caortegac@utn.edu.ec

REQUERIMIENTOS DEL SISTEMA:

Nomenclatura:	SySR 1
Requerimiento:	El sistema utilizara los algoritmos: ECDH y ELGamal Eliptico
Descripción:	Implementar los diferentes algoritmos para la generación de claves, intercambio de claves ECDH y cifrado/descifrado ElGamal. Por cada algoritmo se requiere una función que recibe varias entradas y devuelve una o más salidas.
Prioridad:	Alta

Nomenclatura:	SySR 2
Requerimiento:	Cada escenario de simulación debe contar con los parámetros esenciales de cada WSN
Descripción:	Es preciso configurar la red WSN en los tres ambientes de simulación, configurando sus parámetros esenciales, como canal, ancho de banda, potencia, etc.
Prioridad:	Media

Nomenclatura:	SySR 3
Requerimiento:	Gestionar ciclo de vida de las claves, donde se renueven cada cierto tiempo
Descripción:	Las claves deben cambiar cada cierto periodo de tiempo, dependiendo de la longitud de clave, complejidad de la contraseña y el tiempo de descifrado.
Prioridad:	Media
Nomenclatura:	SySR 4
Requerimiento:	Notificaciones del Sistema ante eventos inesperados en los procesos de: Generación de claves, encriptación y desencriptación
Descripción:	El sistema debe ser capaz de notificar si existe algún evento inesperado durante el transcurso de la encriptación. Además, se debe controlar los parámetros iniciales, como el tipo de curva, punto generador y tamaño del campo, para que no se produzca algún error inesperado.
Prioridad:	Alta

caortegac@utn.edu.ec

Nomenclatura:	SySR 5
Requerimiento:	Implementación de medidas de seguridad adicionales, como autenticación y control de acceso a los datos
Descripción:	El sistema debe contar con mecanismos de protección que permitan asegurar la privacidad, confidencialidad e integridad de la información almacenada y transmitida en el sistema.
Prioridad:	Baja

Nomenclatura:	SySR 6
Requerimiento:	Menor tiempo de respuesta dependiendo de la aplicación y entorno de prueba
Descripción:	Dependiendo de la aplicación y entorno de prueba, se debe esperar un tiempo de respuesta prudente después de implementar el algoritmo ECC. El retardo es un parámetro de suma importancia en redes WBAN o las enfocadas a alertas tempranas, por otro lado, el retardo no es un factor muy clave para redes en la agricultura o enfocadas a la monitorización de lugares cerrados.
Prioridad:	Media

Nomenclatura:	SySR 7
Requerimiento:	Verificación de la seguridad utilizando análisis de tramas
Descripción:	Se debe hacer un análisis de las tramas cifradas con ECC para indagar que ocurren con los datos, si bien los cálculos para ejecutar el algoritmo son complejos, difíciles de descifrar, y basados en el problema del logaritmo discreto donde no existen algoritmos eficientes para desencriptar.
Prioridad:	Alta

Nomenclatura:	SySR 8
Requerimiento:	Longitud de clave equilibrando tiempo de procesamiento y de cifrado
Descripción:	Las WSN tienen recursos limitados de memoria, haciendo que el tamaño de clave sea un factor de alta prioridad, por lo tanto al implementar un algoritmo

caortegac@utn.edu.ec

	criptográfico se debe procurar que utilice una longitud de clave ideal equilibrando tiempos de procesamiento y seguridad.
Prioridad:	Media

Nomenclatura:	SySR 9
Requerimiento:	Compatibilidad con la simulación en OMNeT++ y capacidad de generar datos de simulación para el análisis de desempeño.
Descripción:	La capacidad del sistema para funcionar correctamente en el entorno de simulación en OMNeT++ y para generar datos que puedan ser utilizados para analizar el rendimiento del sistema.
Prioridad:	Alta

Nomenclatura:	SySR 10
Requerimiento:	Documentación y guía de usuario para el uso y configuración del sistema
Descripción:	El proyecto debe contar con la disponibilidad de información y recursos que permitan a los usuarios comprender y utilizar el sistema de manera eficaz. Siendo útil esta documentación para solucionar problemas si se producen errores o problemas de funcionamiento.
Prioridad:	Baja

Nomenclatura:	SySR 11
Requerimiento:	Monitoreo de parámetros de hardware como: nivel de procesamiento y consumo de energía
Descripción:	Cuando la red ya esté funcionando adecuadamente, se necesita aplicar técnicas de monitoreo de parámetros principales de la red con la finalidad de obtener información sobre el funcionamiento de la red antes y después de aplicar criptografía.
Prioridad:	Media

REQUERIMIENTOS DEL SISTEMA – TABLA DE RESUMEN:

caortegac@utn.edu.ec

SySR				
No.	**Requerimiento**	**Prioridad**		
		Alta	**Media**	**Baja**
SySR 1	*El sistema utilizara los algoritmos: ECDH y ELGamal Elíptico*	*X*		
SySR 2	*Cada escenario de simulación debe contar con los parámetros esenciales de cada WSN*		*X*	
SySR 3	*Gestionar ciclo de vida de las claves, donde se renueven cada cierto tiempo*		*X*	
SySR 4	*Notificaciones del Sistema ante eventos inesperados en los procesos de: Generación de claves, encriptación y desencriptación*	*X*		
SySR 5	*Implementación de medidas de seguridad adicionales, como autenticación y control de acceso a los datos*			*X*
SySR 6	*Menor tiempo de respuesta dependiendo de la aplicación y entorno de prueba*		*X*	
SySR 7	*Verificación de la seguridad utilizando análisis de tramas*	*X*		
SySR 8	*Longitud de clave equilibrando tiempo de procesamiento y de cifrado*		*X*	
SySR 9	*Compatibilidad con la simulación en OMNeT++ y capacidad de generar datos de simulación para el análisis de desempeño.*	*X*		
SySR 10	*Documentación y guía de usuario para el uso y configuración del sistema.*			*X*
SySR 11	*Monitoreo de parámetros de hardware como: nivel de procesamiento y consumo de energía*		*X*	

caortegac@utn.edu.ec

REQUERIMIENTOS DE ARQUITECTURA:

Nomenclatura:	SrSH 1
Requerimiento:	Dispositivo base de alto rendimiento, equipo donde se implementa la simulación de las redes con ECC
Descripción:	El investigador requiere un equipo sobre el cual instalar el simulador, este debe tener buenas características en cuanto a procesamiento y capacidad de computación para que la simulación con todos sus componentes pueda funcionar adecuadamente y lograr los objetivos propuestos para esta investigación.
Prioridad:	Media

Nomenclatura:	SrSH 2
Requerimiento:	Definir Hardware para WSN utilizando el estándar 802.15.4 para definir el tipo de comunicación
Descripción:	Definir el tipo de Sensores, Puerta de Enlace y módulos de comunicación, donde se implemente el estándar 802.15.4, en donde se va a implementar en los diferentes despliegues de WSN.
Prioridad:	Media

Nomenclatura:	SrSH 3
Requerimiento:	Limitaciones de Hardware en base a tipos de Criptografía aplicados al nodo sensor
Descripción:	Determinar tamaños de claves soportados por el Hardware, y tiempos de procesamiento.
Prioridad:	Alta

Nomenclatura:	SrSH 4
Requerimiento:	Los nodos sensores debe tener un bajo consumo energético.
Descripción:	La extensión de la vida útil de la batería es el principal impulsor del diseño de las redes de sensores inalámbricos (WSN) de bajo consumo. Se debe encontrar una relación entre batería y procesamiento para tener un óptimo desempeño del nodo sensor.
Prioridad:	Bajo

caortegac@utn.edu.ec

Nomenclatura:	SrSH 5
Requerimiento:	La plataforma de simulación proporciona todas las características para la creación de una red de sensores
Descripción:	El programa debe contar con los protocolos y las características de los dispositivos inalámbricos, ayudándonos así a crear una simulación más cercana a la realidad y con datos certeros.
Prioridad:	Alta

SrSH				
No.	***Requerimiento***	***Prioridad***		
		Alta	***Media***	***Baja***
SrSH 1	*Dispositivo base de alto rendimiento, equipo donde se implementa la simulación de las redes con ECC*		*X*	
SrSH 2	*Definir Hardware para WSN utilizando el estándar 802.15.4 para definir el tipo de comunicación*		*X*	
SrSH 3	*Limitaciones de Hardware en base a tipos de Criptografía aplicados al nodo sensor*	*X*		
SrSH 4	*Los nodos sensores debe tener un bajo consumo energético.*			*X*
SrSH 5	*La plataforma de simulación proporciona todas las características para la creación de una red de sensores*	*X*		

REVISADO POR	ELABORADO POR
MSC. Fabián Cuzme Rodríguez **TUTOR DE TRABAJO DE GRADO**	Christopher Alejandro Ortega **INVESTIGADOR**

ANEXO B – Funciones ECC

```
#include <cstring>
#include <fstream>
#include <iostream>
#include <vector>
#include <cmath>
#include <algorithm>
#include <string>
#include "funcionesECC.h"

using namespace std;

const unsigned int SHA256::sha256_k[64] = //UL = uint32
        {0x428a2f98, 0x71374491, 0xb5c0fbcf, 0xe9b5dba5,
         0x3956c25b, 0x59f111f1, 0x923f82a4, 0xab1c5ed5,
         0xd807aa98, 0x12835b01, 0x243185be, 0x550c7dc3,
         0x72be5d74, 0x80deb1fe, 0x9bdc06a7, 0xc19bf174,
         0xe49b69c1, 0xefbe4786, 0x0fc19dc6, 0x240ca1cc,
         0x2de92c6f, 0x4a7484aa, 0x5cb0a9dc, 0x76f988da,
         0x983e5152, 0xa831c66d, 0xb00327c8, 0xbf597fc7,
         0xc6e00bf3, 0xd5a79147, 0x06ca6351, 0x14292967,
         0x27b70a85, 0x2e1b2138, 0x4d2c6dfc, 0x53380d13,
         0x650a7354, 0x766a0abb, 0x81c2c92e, 0x92722c85,
         0xa2bfe8a1, 0xa81a664b, 0xc24b8b70, 0xc76c51a3,
         0xd192e819, 0xd6990624, 0xf40e3585, 0x106aa070,
         0x19a4c116, 0x1e376c08, 0x2748774c, 0x34b0bcb5,
         0x391c0cb3, 0x4ed8aa4a, 0x5b9cca4f, 0x682e6ff3,
         0x748f82ee, 0x78a5636f, 0x84c87814, 0x8cc70208,
         0x90befffa, 0xa4506ceb, 0xbef9a3f7, 0xc67178f2};

void SHA256::transform(const unsigned char *message, unsigned int block_nb)
{
   uint32 w[64];
   uint32 wv[8];
   uint32 t1, t2;
   const unsigned char *sub_block;
   int i;
   int j;
   for (i = 0; i < (int) block_nb; i++) {
      sub_block = message + (i << 6);
      for (j = 0; j < 16; j++) {
         SHA2_PACK32(&sub_block[j << 2], &w[j]);
      }
      for (j = 16; j < 64; j++) {
         w[j] =  SHA256_F4(w[j -  2]) + w[j -  7] + SHA256_F3(w[j - 15]) + w[j - 16];
      }
      for (j = 0; j < 8; j++) {
         wv[j] = m_h[j];
      }
      for (j = 0; j < 64; j++) {
         t1 = wv[7] + SHA256_F2(wv[4]) + SHA2_CH(wv[4], wv[5], wv[6])
            + sha256_k[j] + w[j];
```

```
            t2 = SHA256_F1(wv[0]) + SHA2_MAJ(wv[0], wv[1], wv[2]);
            wv[7] = wv[6];
            wv[6] = wv[5];
            wv[5] = wv[4];
            wv[4] = wv[3] + t1;
            wv[3] = wv[2];
            wv[2] = wv[1];
            wv[1] = wv[0];
            wv[0] = t1 + t2;
        }
        for (j = 0; j < 8; j++) {
            m_h[j] += wv[j];
        }
    }
}

void SHA256::init()
{
    m_h[0] = 0x6a09e667;
    m_h[1] = 0xbb67ae85;
    m_h[2] = 0x3c6ef372;
    m_h[3] = 0xa54ff53a;
    m_h[4] = 0x510e527f;
    m_h[5] = 0x9b05688c;
    m_h[6] = 0x1f83d9ab;
    m_h[7] = 0x5be0cd19;
    m_len = 0;
    m_tot_len = 0;
}

void SHA256::update(const unsigned char *message, unsigned int len)
{
    unsigned int block_nb;
    unsigned int new_len, rem_len, tmp_len;
    const unsigned char *shifted_message;
    tmp_len = SHA224_256_BLOCK_SIZE - m_len;
    rem_len = len < tmp_len ? len : tmp_len;
    memcpy(&m_block[m_len], message, rem_len);
    if (m_len + len < SHA224_256_BLOCK_SIZE) {
        m_len += len;
        return;
    }
    new_len = len - rem_len;
    block_nb = new_len / SHA224_256_BLOCK_SIZE;
    shifted_message = message + rem_len;
    transform(m_block, 1);
    transform(shifted_message, block_nb);
    rem_len = new_len % SHA224_256_BLOCK_SIZE;
    memcpy(m_block, &shifted_message[block_nb << 6], rem_len);
    m_len = rem_len;
    m_tot_len += (block_nb + 1) << 6;
}
```

```
void SHA256::final(unsigned char *digest)
{
   unsigned int block_nb;
   unsigned int pm_len;
   unsigned int len_b;
   int i;
   block_nb = (1 + ((SHA224_256_BLOCK_SIZE - 9)
                < (m_len % SHA224_256_BLOCK_SIZE)));
   len_b = (m_tot_len + m_len) << 3;
   pm_len = block_nb << 6;
   memset(m_block + m_len, 0, pm_len - m_len);
   m_block[m_len] = 0x80;
   SHA2_UNPACK32(len_b, m_block + pm_len - 4);
   transform(m_block, block_nb);
   for (i = 0 ; i < 8; i++) {
      SHA2_UNPACK32(m_h[i], &digest[i << 2]);
   }
}

std::string sha256(std::string input)
{
   unsigned char digest[SHA256::DIGEST_SIZE];
   memset(digest,0,SHA256::DIGEST_SIZE);

   SHA256 ctx = SHA256();
   ctx.init();
   ctx.update( (unsigned char*)input.c_str(), input.length());
   ctx.final(digest);

   char buf[2*SHA256::DIGEST_SIZE+1];
   buf[2*SHA256::DIGEST_SIZE] = 0;
   for (int i = 0; i < SHA256::DIGEST_SIZE; i++)
      sprintf(buf+i*2, "%02x", digest[i]);
   return std::string(buf);
}


//PERSO
bool primo(int number){

   if(number < 2) return false;
   if(number == 2) return true;
   if(number % 2 == 0) return false;
   for(int i=3; (i*i)<=number; i+=2){
      if(number % i == 0 ) return false;
   }
   return true;

}

//################################################################
int sum(vector<int> sumArray){
```

```
    vector<int> resArray;
    int total;

    if (sumArray[0] != 999999) {
        resArray.push_back(1);
    } else {
        resArray.push_back(0);
    }
    if (sumArray[1] != 999999) {
        resArray.push_back(1);
    } else {
        resArray.push_back(0);
    }

    total = resArray[0] + resArray[1];
    resArray.clear();

    return total;
}

//#################################################################
//Función parametrosEcc
vector<vector<int> > puntosCurvaEliptica(int a, int b, int p) {

    vector<vector<int> > Puntos;
    if ((a%1 != 0) || (b%1 != 0)) {
        cout << "El valor de a y de b deben ser enteros" << endl;
    } else if (primo(p) != 1 || p < 4) {
        cout << "p debe ser un numero primo mayor que 3" << endl;
    } else if ((pow(4*a,3) + pow(27*b,2)) == 0) {
        cout << "Es una curva singular, Intenta con otra curva" << endl;
    } else {
        cout << "Todo OK" << endl;
        int indice = 1;
        for (int x = 0; x < p; x++) {
            int ladoDerecho = fmod((pow(x,3) + a*x + b), p);
            for (int y = 0; y < p; y++) {
                if (fmod(pow(y,2),p) == ladoDerecho) {
                    Puntos.push_back({ x, y });
                }
            }
        }
        Puntos.push_back({ 999999, 999999 });
    }

    return Puntos;
}

//#################################################################
//Función Algoritmo Extendido de Euclides
int AEE(int a, int b) {
    vector<int> x{ 1, 0 };
    vector<int> y{ 0, 1 };
```

```
    int q = 0;
    int r = 0;
    int m = 0;
    int n = 0;
    int inverso = 0;

    while (a != 0) {
        q = trunc(b/a);
        r = b%a;
        m = x[1] - (q*x[0]);
        n = y[1] - (q*y[0]);
        x[1] = x[0];
        y[1] = y[0];
        x[0] = m;
        y[0] = n;
        b = a;
        a = r;

        if (x[1] > 0) {
            inverso = x[1];
        } else {
            inverso = x[0] + x[1];
        }
    }

    return inverso;
}

//#####################################################################
//Función sumaPuntosCurvaEliptica
vector<int> sumaPuntosCurvaEliptica(vector<int> P, vector<int> Q, int a, int b, int p) {

    vector<int> R;
    int m = 0;

    if (a%1 != 0 || b%1 != 0) {
        cout << "El valor de a y de b deben ser enteros" << endl;
    } else if (primo(p) != 1 || p < 4) {
        cout << "p debe ser un numero primo mayor que 3" << endl;
    } else if ((4*pow(a,3) + 27*pow(b,2)) == 0) {
        cout << "Curva singular, Intenta otra curva" << endl;
    }


    if (P[0] == 999999) {
        if (fmod(pow(Q[1],2),p) != fmod(((pow(Q[0],3)) + a*Q[0] + b),p)) {
            cout << "El punto Q debe pertenecer a la curva eliptica" << endl;
        } else {
            R = Q;
            return R;
        }
    }
```

```
    if (Q[0] == 999999) {
        if (fmod((pow(P[1],2)),p) != fmod(((pow(P[0],3)) + a*P[0] + b),p)){
            cout << "El punto P debe pertenecer a la curva eliptica" << endl;
        } else {
            R = P;
            return R;
        }
    }

    if (fmod((pow(P[1],2)),p) != fmod(((pow(P[0],3)) + a*P[0] + b),p)){
        cout << "*El punto P debe pertenecer a la curva eliptica" << endl;
    } else if (fmod(pow(Q[1],2),p) != fmod(((pow(Q[0],3)) + a*Q[0] + b),p)) {
        cout << "*El punto Q debe pertenecer a la curva eliptica" << endl;
    }

    if (P[0] == Q[0] && P[1] == Q[1]) {
        int dy = fmod(((3*pow(P[0],2)) + a),p);
        int dx = (2*P[1])%p;
        if (abs(__maxCdiv(dx, p)) == 1) {
            m = fmod(dy*AEE(dx, p),p);
            int x3 = ((int(pow(m,2)) - P[0] - Q[0])%p + p)%p;
            int y3 = ((m*(P[0] - x3) - P[1])%p + p)%p;
            R = {x3,y3};
        } else {
            m = 999999;
            R = {999999,999999};
        }
    } else {
        int dy = ((Q[1] - P[1])%p + p)%p; // (a%b + b)%b
        int dx = ((Q[0] - P[0])%p + p)%p;
        if (abs(__maxCdiv((Q[0] - P[0]),p)) == 1) {
            m = dy*AEE(dx, p)%p;
            int x3 = ((int(pow(m,2)) - P[0] - Q[0])%p + p)%p;
            int y3 = ((m*(P[0] - x3) - P[1])%p + p)%p;
            R = {x3,y3};
        } else {
            m = 999999;
            R = {999999,999999};
        }
    }

    return R;
}

//#####################################################################
//Función ordenPuntoCurvaEliptica
int ordenPuntoCurvaEliptica(vector<int> P, int a, int b, int p) {

    int Orden = 1;
    vector<int> SumPs = P;
    while(sum(SumPs) > 0) {
        SumPs = sumaPuntosCurvaEliptica(SumPs, P, a, b, p);
        Orden++;
```

```
    }

    return Orden;
}

//#################################################################
//Función parametrosEcc
vector< vector<int>> parametrosEcc(int a, int b, int p, int parPuntos) {

    vector< vector<int>> resultado;

    vector<vector<int> > Puntos = puntosCurvaEliptica(a, b, p);

    /*
    for (int i = 0; i < Puntos.size(); i++) {
        for (int j = 0; j < Puntos[i].size(); j++)
          cout << " " << Puntos[i][j];
      cout << endl;
    }
    */

    int ordenCurva = Puntos.size();

    vector<int> G{ -1, -1 };
    int n = 0;
    int h = 5;
    int cont = 0;

    while ((h > 4) || ((h%1) != 0)) {
        //int random = rand() % (ordenCurva - 1);
        int randomP = parPuntos; //99

        G.clear();
        G.push_back(Puntos[randomP][0]);
        G.push_back(Puntos[randomP][1]);
        //cout << "G: " << G[0] << " " << G[1] << endl;
        n = ordenPuntoCurvaEliptica(G,a,b,p);
        h = ordenCurva/n;

        cont++;
        if ((cont == 5000) || G[0] == -1) {
            cout << "Error" << endl;
        }
    }

    vector<int> ordenCurva_vector{ ordenCurva, 0 };
    resultado.push_back(ordenCurva_vector);
    resultado.push_back(G);
    vector<int> n_vector{ n, 0 };
    resultado.push_back(n_vector);
    vector<int> h_vector{ h, 0 };
    resultado.push_back(h_vector);
```

```
    return resultado;
}

//#################################################################
//Función multEscalarCurvaEliptica
vector<int> multEscalarCurvaEliptica(vector<int> P, int k, int a, int b, int p) {

    vector<int> Q{ 999999, 999999 };

    do {
        int kt = k;
        while (kt > 0) {
            if (kt%2 == 1) {
                Q = sumaPuntosCurvaEliptica(Q, P, a, b, p);
            }
            P = sumaPuntosCurvaEliptica(P, P, a, b, p);
            kt = roundDown(kt/2);
        }
    } while (Q[0] == 999999);

    return Q;
}

//#################################################################
//Función generadorClavesEcc
vector<vector<int> > generadorClavesEcc(int a, int b, int p, vector<int> G, int n) {

    vector<int> Q;
    int d;
    do {
        srand(time(NULL));
        d = rand() % (n - 1);
        //d = 10000; //AQUI
        //int d = intrand(n); // Genera un entero aleatorio desde 0 hasta Length
        //cout << "d: " << d << endl;
        Q = multEscalarCurvaEliptica(G, d, a, b, p);
    }
    while (Q[0] == 999999);

    int clavePrivada = d;
    vector<int> clavePublica = Q;
    vector< vector<int>> resultado;
    vector<int> clavePrivada_vector{ clavePrivada, 0 };
    resultado.push_back(clavePrivada_vector);
    resultado.push_back(clavePublica);

    return resultado;
}

//#################################################################
//funcion puntoEccToChar
char puntoEccToChar(vector<int> P, int a, int b, int p, int K) {
    int m = 0;
```

```
    char m_char;

    if (a%1 != 0 || b%1 != 0) {
        cout << "El valor de a y de b deben ser enteros" << endl;
    } else if (primo(p) != 1 || p < 4) {
        cout << "p debe ser un numero primo mayor que 3" << endl;
    } else if ((4*pow(a,3) + 27*pow(b,2)) == 0) {
        cout << "Curva singular, Intenta otra curva" << endl;
    } else if (int(pow(P[1],2))%p != (int(pow(P[0],3)) + a*P[0] + b)%p) {
        cout << "P debe pertenecer a la curva eliptica" << endl;
    }

    for (int i = 0; i < (K - 1); i++) {
        m = (((P[0] - i)/K)%p + p)%p;
        int x = ((m*K + i)%p + p)%p;
        if (P[0] == x) {
            m_char = char(m);
            return m_char;
        }
    }
    cout << "No se puede representar el caracter" << endl;
    cout << "Intente usar un primo p más grande" << endl;
}
//#########################################################
//funcion expRapida
int expRapida(int A, int B, int p) {
    int result = 1;
    while(B > 0) {
        if (B%2 == 1) {
            result = (result*A)%p;
        }
        B = roundDown(B/2);
        A = (A*A)%p;
    }
    return result;
}

//#########################################################
//funcion charToPuntoEcc
vector<int> charToPuntoEcc(char m, int a, int b, int p, int K) {
    vector<int> M;
    if (a%1 != 0 || b%1 != 0) {
        cout << "El valor de a y de b deben ser enteros" << endl;
    } else if (primo(p) != 1 || p < 4) {
        cout << "p debe ser un numero primo mayor que 3" << endl;
    } else if ((4*pow(a,3) + 27*pow(b,2)) == 0) {
        cout << "Curva singular, Intenta otra curva" << endl;
    }

    double m_d = double(m);
    cout << m_d << endl;

    if ((m_d + 1)*K >= p) {
```

```
        cout << "Error: Con los valores de K y m, debe usar un primo p más grande o un K menor" <<
endl;
        vector<int> error { 0, 0 };
        return error;
    }

    for (int i = 0; i < K - 1; i++) {
        double x = fmod(((m_d*K) + i),p);
        double ladoDerecho = fmod(((pow(x,3)) + a*x + b),p);
        double y = expRapida(ladoDerecho, (p + 1)/4, p);
        if (fmod((pow(y,2)),p) == ladoDerecho) {
            M.clear();
            M.push_back(x);
            M.push_back(y);
            return M;
        }
    }
}

//#########################################################
//funcion cifradorElGamalEcc2
vector<vector<int> > cifradorElGamalEcc2(int a, int b, int p, int K, vector<int> S, string msj) {
    vector<vector<int> > M;
    vector<vector<int> > C2;
    const int msj_length = msj.length();
    char* char_array = new char[msj_length + 1];
    strcpy(char_array, msj.c_str());
    //cout << "M: " << endl;
    for (int i = 0; i < msj_length; i++) {
        M.push_back(charToPuntoEcc(char_array[i], a, b, p, K));
        //cout << M[i][0] << " " << M[i][1] << endl;
    }

    int tam = M.size();

    for (int j = 0; j < tam; j++) {
        vector<int> M_temp { M[j][0], M[j][1] };
        C2.push_back(sumaPuntosCurvaEliptica(M_temp, S, a, b, p));
    }

    return C2; //MsjCifrado
}

//#########################################################
//funcion descifradorElGamalEcc2
string descifradorElGamalEcc2(int a, int b, int p, int K, vector<int> S, vector<vector<int> >
MsjCifrado) {
    vector<vector<int> > C2 = MsjCifrado;
    vector<vector<int> > M;
    vector<int> mS { S[0], ((-S[1]%p)+ p)%p};
    int tam = C2.size();
    for (int i = 0; i < tam; i++) {
        vector<int> C2_temp { C2[i][0], C2[i][1] };
```

```
        M.push_back(sumaPuntosCurvaEliptica(C2_temp, mS, a, b, p));
    }

    string msjDescifrado = "";
    tam = M.size();
    for (int j = 0; j < tam; j++) {
        vector<int> M_temp { M[j][0], M[j][1] };
        char c = puntoEccToChar(M_temp, a, b, p, K);
        string s(1, c);
        msjDescifrado = msjDescifrado + s;
    }

    return msjDescifrado;
}
```

ANEXO C – Script para el análisis de procesamiento del sistema ECC

```
%% DiffieHelmanECC and ElGamalEliptico
close all; clear; clc;
data1 = 'SensotN1{temp: 810, hum: 21, }';
data2 = 'SensotN1{temp: 30, hum: 468, }';
data3 = 'SensotN1{temp: 313, hum: 81, }';
disp('*******************************************')
disp('INTERCAMBIO DE CLAVES DIFFIE HELLMAN ELIPTICO')
disp('*******************************************')
disp(' ')
disp('Elliptic Curve Diffie-Hellman Key Exchange')
disp(' ')
disp('Ingrese los enteros a, b y el primo p > 3 de la curva Fp');
disp('y^2 = x^3 + ax + b (mod p)');
a = input('Numero entero a = ');
b = input('Numero entero b = ');
p = input('Numero primo p = ');
disp('Ingrese el entero K el cual se utiliza para el cifrado');
K = input('Numero entero K = ');
disp('Ingrese el punto generador G');
G = input('Punto G = ');
n = ordenPuntoCurvaElipticaModp(G,a,b,p);
disp(n);


% Definir los valores de entrada
frecuencia = 2.4 * 1e9; % en GHz

%------------------------
tic
profile on
memory
whos
%------------------------

[dA, QA] = generadorClavesEcc(a, b, p, G, n);
fprintf('La clave privada de A es: [%d] \n',dA);
fprintf('La clave publica de A es: [%d, %d] \n',QA(1),QA(2));
[dB, QB] = generadorClavesEcc(a, b, p, G, n);
fprintf('La clave privada de B es: [%d] \n',dB);
fprintf('La clave publica de B es: [%d, %d] \n',QB(1),QB(2));
SA = multEscalarCurvaElipticaModp(QB, dA, a, b, p);
fprintf('La clave compartida generada por A es: [%d, %d]
\n',SA(1),SA(2));
SB = multEscalarCurvaElipticaModp(QA, dB, a, b, p);
fprintf('La clave compartida generada por B es: [%d, %d]
\n',SB(1),SB(2));
%-------------------ELGAMAL-----------------------------------------
-
disp('*******************************************')
disp('   CIFRADO DE DATOS CON EL GAMAL ELIPTICO    ')
disp('*******************************************')
disp(' ')
disp('EL GAMAL ELIPTICO ECC')
disp(' ')
disp('El mensaje a cifrar es: ')
msj = data1
```

```
MsjCifrado = cifradorElGamalEcc2(a, b, p, K, SA, msj)
disp('El mensaje decifrado es: ')
msjDescifrado = descifradorElGamalEcc2(a, b, p, K, SB, MsjCifrado)
%--------------------Actualizar clave---------------------------------
disp('*******************************************')
disp('          ACTUALIZAR CLAVE COMUN            ')
disp('*******************************************')
disp(' ')
EnteroxmedioA=floor(SA(1)/2); %Dividimos para 2 el valor de x de la clave
compartida
EnteroxmedioB=floor(SB(1)/2);
SA = multEscalarCurvaElipticaModp(SA, EnteroxmedioA, a, b, p);
fprintf('La nueva clave compartida generada por A es: [%d, %d]
\n',SA(1),SA(2));
SB = multEscalarCurvaElipticaModp(SB, EnteroxmedioB, a, b, p);
fprintf('La nueva clave compartida generada por B es: [%d, %d]
\n',SB(1),SB(2));
%--------------------ELGAMAL-----------------------------------------
-
disp('*******************************************')
disp('   CIFRADO DE DATOS CON EL GAMAL ELIPTICO    ')
disp('*******************************************')
disp(' ')
disp('EL GAMAL ELIPTICO ECC')
disp(' ')
disp('El mensaje a cifrar es: ')
msj = data2
MsjCifrado = cifradorElGamalEcc2(a, b, p, K, SA, msj)
disp('El mensaje decifrado es: ')
msjDescifrado = descifradorElGamalEcc2(a, b, p, K, SB, MsjCifrado)
%--------------------Actualizar clave---------------------------------
disp('*******************************************')
disp('          ACTUALIZAR CLAVE COMUN            ')
disp('*******************************************')
disp(' ')
EnteroxmedioA=floor(SA(1)/2); %Dividimos para 2 el valor de x de la clave
compartida
EnteroxmedioB=floor(SB(1)/2);
SA = multEscalarCurvaElipticaModp(SA, EnteroxmedioA, a, b, p);
fprintf('La nueva clave compartida generada por A es: [%d, %d]
\n',SA(1),SA(2));
SB = multEscalarCurvaElipticaModp(SB, EnteroxmedioB, a, b, p);
fprintf('La nueva clave compartida generada por B es: [%d, %d]
\n',SB(1),SB(2));
%--------------------ELGAMAL-----------------------------------------
-
disp('*******************************************')
disp('   CIFRADO DE DATOS CON EL GAMAL ELIPTICO    ')
disp('*******************************************')
disp(' ')
disp('EL GAMAL ELIPTICO ECC')
disp(' ')
disp('El mensaje a cifrar es: ')
msj = data3
MsjCifrado = cifradorElGamalEcc2(a, b, p, K, SA, msj)
disp('El mensaje decifrado es: ')
msjDescifrado = descifradorElGamalEcc2(a, b, p, K, SB, MsjCifrado)

%------------------------
```

```
tiempo = toc
profile viewer
memory
whos
%------------------------

% Calcular los ciclos de reloj y la cantidad de operaciones por segundo
ciclos_reloj = tiempo * frecuencia;
ops_segundo = (frecuencia * 10^6) / ciclos_reloj;
MIPS = ops_segundo / 10^6;

% Mostrar los resultados
fprintf('Ciclos de reloj: %f\n', ciclos_reloj);
fprintf('Operaciones por segundo: %f\n', ops_segundo);
fprintf('MIPS: %f\n', MIPS);
```