Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional

Unidad Zacatenco

Departamento de Computación

Capa de seguridad para balizas basada en criptografía ligera

Tesis que presenta

Karla Jocelyn Campos Cruz

para obtener el Grado de

Maestro en Ciencias en Computación

Codirectores de la Tesis

Dra. Brisbane Ovilla Martínez
Dr. Cuauhtemoc Mancillas López

Ciudad de México **Agosto 2021**

Resumen

La creciente interconexión de dispositivos cotidianos en el llamado internet de las cosas, plantea una serie de preocupaciones en relación con la seguridad y la privacidad que se debe tener para proteger la información emitida por ellos y las aplicaciones que los utilizan. Tecnologías de comunicación emergentes, como el Bluetooth de baja energía (BBE), han sido diseñadas para la conexión de dispositivos con poca disponibilidad energética. Muchos de estos dispositivos están equipados con microcontroladores restringidos que carecen de sistemas de seguridad eficientes. Particularmente en este trabajo se analizan las balizas, las cuales, son dispositivos que emiten identificadores a través de señales BBE. Las aplicaciones que utilizan balizas son novedosas y en la actualidad se consideran vulnerables a ataques de seguridad, debido a la especificación de baja energía de Bluetooth y los recursos limitados de las balizas.

En este trabajo de tesis se analiza la tecnología de balizas, sus aplicaciones y principales vulnerabilidades. En el inicio del desarrollo se eligieron las primitivas criptográficas que se integran a la capa de seguridad, posteriormente, se configuró un entorno de emulación de balizas empleando una Raspberry Pi y una computadora para probar la implementación del protocolo de seguridad propuesto, este se diseñó con base en algoritmos de criptografía ligera para el establecimiento de una llave común y la generación de identificadores efímeros como contramedidas ante las posibles amenazas de seguridad en este tipo de tecnologías.

Abstract

The growing interconnection of everyday devices in the internet of things presents a series of concerns concerning security and privacy that must be taken into account to protect the information emitted and the applications that use them. Emerging communication technologies, such as Bluetooth Low Energy (BLE), have been designed to connect devices with little energy availability. Many of these devices are equipped with restricted microcontrollers that lack efficient security systems. In particular, beacons are analyzed, which are devices that emit identifiers through BLE signals. Applications which use beacons are novel and are now considered vulnerable to security attacks due to the low energy specification of Bluetooth and the limited resources of beacons. This thesis work analyzes the beacon technology, its applications, and its main vulnerabilities. At the beginning of development, the cryptographic primitives that are integrated into the security layer were chosen; subsequently, a beacon emulation environment was configured using a Raspberry Pi and a computer to test the implementation of the proposed security protocol; this was designed based on lightweight cryptography algorithms for the establishment of a shared key and the generation of ephemeral identifiers as countermeasures against possible security threats in this type of technology.

Agradecimientos

En primer lugar agradezco a mis padres, abuelas y hermanas, por todo su cariño, apoyo y motivación en cada decisión y proyecto de mi vida.

Agradezco enormemente a mis codirectores, el Dr. Cuauhtemoc Mancillas López y la Dra. Brisbane Ovilla Martínez por todo su apoyo, tiempo y esfuerzo que dedicaron durante el desarrollo de mi trabajo de tesis.

Agradezco al CINVESTAV (Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional), en particular al Departamento de Computación, aceptarme en su programa de Maestría en Ciencias en Computación.

De igual manera, agradezco al CONACyT (Consejo Nacional de Ciencia y Tecnología) por el apoyo económico brindado durante mis estudios de maestría. Este trabajo de tesis derivó del proyecto SRE-AMEXCID titulado "Generación y aplicación de herramientas serológicas, moleculares y rastreo de contacto y movilidad, en 6 hospitales de 3 entidades de México, para estudio, mitigación y contención de la epidemia de COVID-19".

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Existe una gran diversidad de dispositivos interconectados en áreas de tecnología emergentes, entre las que destacan, el Internet de las cosas (IdC) y el Bluetooth de baja energía (BBE), por lo general estos dispositivos se comunican de forma inalámbrica entre sí, trabajan en conjunto para realizar alguna tarea de comunicación usualmente de forma descuidada y sin ningún protocolo de seguridad de por medio. Debido al creciente número de dispositivos restringidos del IdC, la estandarización de la seguridad debe lograrse en un futuro cercano, porque una vez que muchos dispositivos sean implementados, es posible que no tengan la capacidad de actualizarse, lo que puede generar vulnerabilidades y amenazas futuras. Las limitaciones de recursos en estos dispositivos hacen que sea poco práctico utilizar algoritmos de cifrado/descifrado muy complejos y que consuman mucho tiempo para la comunicación segura. Esto provoca que los sistemas del IdC sean altamente susceptibles a diversos tipos de ataques como el seguimiento no autorizado, suplantación de identidad, falsificación de información, entre otros. Una solución ahorradora (con respecto a la cantidad de recursos utilizados) que proporcione seguridad a los dispositivos reducidos, necesita de un sistema que pueda ser implementado en ellos a pesar de sus recursos limitados. Dicha necesidad ha motivado el desarrollo de un área de investigación denominada como criptografía ligera. Por este motivo el *National Institute of Standards and Technology* (NIST), ha iniciado un proceso para solicitar, evaluar y estandarizar algoritmos criptográficos ligeros que sean adecuados para su uso en entornos restringidos donde el rendimiento de los estándares criptográficos actuales del NIST son inaceptables [1].

Las tecnologías del IdC y BBE se han movido más rápidamente que los mecanismos disponibles para protegerlas, particularmente a los dispositivos más pequeños. Salvaguardar estos dispositivos, los datos y las redes a las que se conectan puede ser un desafío debido a la variedad de artefactos y proveedores, así como a la dificultad de agregar seguridad a los aparatos con capacidades limitadas. Así, en esta tesis planeamos estudiar algoritmos criptográficos ligeros para balizas que usan BBE para sus comunicaciones.

1.1. Objetivos

General

Mejorar la seguridad en aplicaciones basadas en balizas BBE utilizando criptografía ligera.

Particulares

1. Comprender la tecnología de balizas BBE, sus características y las aplicaciones que tiene en la actualidad.

2. Analizar las amenazas y vulnerabilidades de seguridad que implica la utilización de balizas BBE en aplicaciones del IdC.

3. Implementar un protocolo de seguridad para las balizas BBE y proveer mayor seguridad en sus aplicaciones utilizando algoritmos de criptografía ligera.

1.2. Propuesta de solución

En este trabajo se propone un análisis de la tecnología Bluetooth de baja energía y la seguridad en los dispositivos que la utilizan, particularmente de las balizas BBE. Se propone un protocolo de seguridad diseñado con base en algoritmos de criptografía ligera para un acuerdo de llaves en común y la generación de identificadores efímeros, ambos integrados como capas del protocolo a manera de contramedidas ante posibles amenazas de seguridad como los ataques de suplantación y falsificación de información. Este trabajo de tesis derivó del proyecto SRE-AMEXCID titulado "Generación y aplicación de herramientas serológicas, moleculares y rastreo de contacto y movilidad, en seis hospitales de tres entidades de México, para estudio, mitigación y contención de la epidemia de COVID-19". El protocolo de seguridad esta basado en el esquema utilizado en la aplicación Applacovid [2], modificando las primitivas criptográficas implementadas con algoritmos de criptografía ligera.

1.3. Organización de la Tesis

En el capítulo 2 se presenta el contexto necesario para comprender la tecnología de los dispositivos BBE, se describen las capas de la arquitectura BBE, las topologías de conexión y las características más importantes en comparación al Bluetooth clásico. De igual manera se aborda la tecnología de balizas BBE, los protocolos de comunicación más usados, las áreas de aplicación actuales y las principales amenazas de seguridad a las que se enfrenta durante su implementación.

En el capítulo 3 se exponen los preliminares matemáticos y las primitivas criptográficas utilizadas en las capas del protocolo de seguridad que se propone en esta

tesis, las cuales, se basan en la implementación de un protocolo de acuerdo de llaves con criptografía de curva elíptica, así como en la generación y transmisión de identificadores efímeros creados con un algoritmo de cifrado ligero.

En el capítulo 4 se analizan las características físicas de un tipo de baliza BBE, luego, se describe el proceso para la emulación de balizas BBE con distintos protocolos de comunicación, con dichas emulaciones podrían realizarse ataques de suplantación en aplicaciones con balizas BBE, en este caso se utilizan para comprobar el funcionamiento del protocolo propuesto.

En el capítulo 5 se describe el diseño general del protocolo de seguridad, se plantea una arquitectura en capas con la integración de algoritmos criptográficos de llave pública para la comunicación segura de balizas BBE y la generación de identificadores efímeros pseudoaleatorios como mecanismos de seguridad, además, se realizan programas para las pruebas de implementación del protocolo de seguridad de balizas BBE.

En el capítulo 6 se muestran los resultados obtenidos durante las pruebas de implementación del protocolo de seguridad en un entorno emulado. Adicionalmente, se expone el proceso de establecimiento de llaves en distintos lenguajes de programación, se realiza una comparación de sus tiempos de ejecución con lo cual se eligió el entorno y lenguaje de programación más eficiente.

Por último, en el capítulo 7 se presentan las conclusiones, contribuciones y algunas ideas para el trabajo futuro.

Capítulo 2

Contexto Tecnológico

Hace algunos años, cuando se hablaba del futuro hacia el que nos conducía el avance de las tecnologías de la información y comunicaciones, se mencionaba al internet de las cosas (IdC) como una interconexión de artefactos cotidianos en una red que contaría con un alto nivel de inteligencia. A finales de 2017 se conectaron aproximadamente 8,400 millones de dispositivos del IdC y se estima que habrá alrededor de 30 mil millones de dispositivos a finales de este año [3], la rápida transición de plataformas, el aumento del uso de nuevas tecnologías y la adopción intensa de sensores impulsa esa trayectoria. El IdC hace posible que se creen nuevos niveles de relación, interacción y de dar mejor servicio a los usuarios, prácticamente de forma automática. Una de las tecnologías de comunicación altamente utilizadas por IdC es el Bluetooth de baja energía (BBE), fue diseñado especialmente para los dispositivos con recursos limitados (como la disponibilidad de energía). Las balizas BBE son uno de los nuevos tipos de dispositivos de interacción del IdC, estas emiten señales BBE con identificadores que son captados por aplicaciones compatibles y que ejecutan acciones en un dispositivo móvil.

En este capítulo se explica en detalle la tecnología BBE, las capas de su arquitectura, casos de uso y las vulnerabilidades de seguridad en su implementación. De igual manera se describe la tecnología de balizas BBE, sus principales aplicaciones, protocolos de comunicación más importantes y amenazas de seguridad actuales.

2.1. Bluetooth de baja energía

BBE es la nueva especificación a partir de la versión 4.0 de la tecnología Bluetooth desarrollado por el SIG (*Bluetooth Special Interest Group*). Se ha diseñado como una tecnología complementaria al Bluetooth clásico para garantizar un consumo de energía bajo y menor tiempo de establecimiento de conexión. A pesar del uso de la misma banda de frecuencia y las similitudes compartidas, BBE debe considerarse un nuevo estándar con objetivos y aplicaciones diferentes. BBE está hecho para la transmisión de pequeñas cantidades de datos y por lo tanto de ultra-bajo consumo de energía. Está pensado para mantener una conexión entre dispositivos por tiempos pequeños.

Esto permite que los dispositivos estén activos sólo cuando se les pide la transmisión de datos.

2.1.1. Datos técnicos de BBE

Algunos de las características técnicas más importantes del BBE son las siguientes:

- El espectro de frecuencia que ocupa es de 2.400 - 2.4835 GHz. Está segmentado en 40 canales de 2 MHz de ancho [4].
- La velocidad máxima de datos soportada (introducida en la versión 5 de Bluetooth) es de 2 Mbps.
- La distancia de comunicación varía significativamente según el entorno que rodea a los dispositivos BBE que se comunican, una distancia común es de 10 a 30 metros.
- El consumo de energía también varía mucho, depende de la implementación de la aplicación, los diferentes parámetros BBE y el el hardware utilizado. El consumo del BBE durante la transmisión suele ser inferior a 15 mA.
- La seguridad es opcional en la comunicación BBE, depende del dispositivo y de los desarrolladores de las aplicaciones implementarla. Sin embargo, también hay diferentes niveles de seguridad que se pueden implementar.
- BBE está diseñado para aplicaciones de transferencia de datos de bajo ancho de banda, ya que la implementación de BBE para aplicaciones de gran ancho de banda comprometería significativamente el bajo consumo de energía.
- Las conexiones, anuncios primarios, descubrimiento de servicios y características, así como la lectura y escritura de estas características es posible entre dos dispositivos BBE independientemente de su versión de Bluetooth compatible.

2.1.2. Bluetooth clásico y BBE

Existe una gran diferencia entre el Bluetooth clásico y el BBE, en términos de especificaciones técnicas, implementación y tipos de aplicaciones para las que se adapta cada uno. Algunas de las principales diferencias se resumen en la tabla 2.1.

Bluetooth clásico	**BBE**
Se utiliza en aplicaciones de transmisión de audio y transferencia de archivos.	Se utiliza para datos de sensores y control de dispositivos.
Velocidad de 3 Mbps.	Velocidad de 2 Mbps.
Opera sobre 79 canales de RF.	Opera sobre 40 canales de RF.
La detección ocurre en 32 canales.	La detección ocurre en 3 canales, las conexiones son más rápidas que con el Bluetooth clásico.

Tabla 2.1: Comparación de Bluetooth clásico y BBE.

BBE ha presentado algunas ventajas y beneficios significativos en tecnologías similares del IdC. Algunas de estas ventajas incluyen el menor consumo de energía, el menor costo de módulos y conjuntos de chips, el acceso gratuito a los documentos oficiales y su existencia en la mayoría de teléfonos móviles del mercado, esta es probablemente la mayor ventaja que tiene BBE sobre sus competidores como ZigBee, Z-Wave y Thread. Con los beneficios mencionados anteriormente, existe una serie de casos de uso en los que tiene más sentido utilizar el BBE:

- Datos de ancho de banda bajo: En los casos en que un dispositivo transfiere pequeñas cantidades de datos que representan lecturas de sensores o para controlar actuadores.
- Configuración del dispositivos: Se puede utilizar como interfaz secundaria para configurar un dispositivo antes de que se establezca la conexión inalámbrica principal.
- Uso de teléfono inteligente como interfaz: Debido a que los dispositivos pequeños de baja energía generalmente no tienen pantallas grandes y sólo pueden mostrar cantidades limitadas de datos al usuario final. Otro beneficio de usar un teléfono inteligente es que los datos se pueden transmitir a la nube.
- Dispositivos personales y portátiles: Para casos en los que un dispositivo es portátil y se puede ubicar en áreas donde ninguno otro puede.
- Dispositivos de sólo transmisión: Estos dispositivos tienen la tarea simple de transmitir datos para que otros dispositivos puedan leerlos.

Bluetooth 4.0 y BBE son soportados en la mayoría de plataformas a partir de las versiones listadas a continuación:

- iOS5+ (iOS7+ preferible)
- Android 4.3+ (numerosas correcciones de errores en 4.4+)
- Apple OS X 10.6+

- Windows 8 (XP, Vista y 7 sólo soportan Bluetooth 2.1)
- GNU/Linux Vanilla BlueZ 4.93+

2.1.3. Topologías BBE

El diagrama de la figura 2.1 explica la forma en que funcionan los dispositivos en una topología conectada uno a muchos. Un periférico sólo se puede conectar a un dispositivo central (como un teléfono móvil) a la vez, pero el dispositivo central se puede conectar a varios periféricos BBE (como balizas). Si es necesario intercambiar datos entre dos periféricos, será necesario implementar un sistema de buzón de correo personalizado donde todos los mensajes pasen por el dispositivo central. Sin embargo, una vez que se establece una conexión entre un periférico y un dispositivo central, la comunicación puede tener lugar en ambas direcciones, lo que es diferente al enfoque de transmisión unidireccional que utiliza sólo datos de anuncio [4].

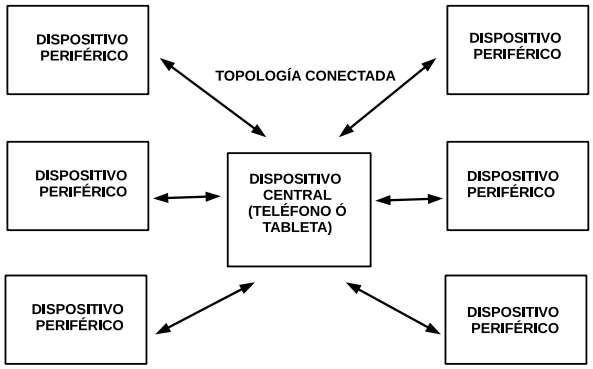


Figura 2.1: Topología uno a muchos en BBE.

Existen otras dos topologías soportadas por BBE mostradas en la figura 2.2. La topología uno a uno es cuando dos dispositivos BBE están conectados directamente entre sí, por otro lado, la topología muchos a muchos es cuando varios dispositivos se encuentran en una malla y pueden conectarse con todos los que estén en ella.

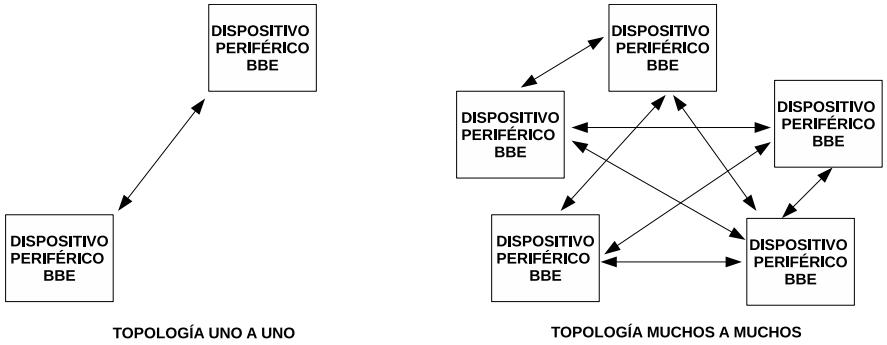


Figura 2.2: Otras topologías en BBE.

Una característica que BBE carecía desde el principio es la capacidad de admitir una topología de muchos a muchos (a menudo denominada red de malla), donde varios dispositivos BBE pueden enviarse mensajes entre sí y retransmitir mensajes a otros dispositivos dentro de una red. Todo esto cambió en julio de 2017 cuando SIG lanzó el estándar Bluetooth malla. El objetivo de la malla Bluetooth es aumentar el alcance de las redes BBE y agregar soporte para más aplicaciones industriales que utilizan la tecnología BBE. Antes del lanzamiento de la malla Bluetooth, BBE sólo admitía dos topologías (uno a uno y uno a muchos).

2.1.4. Arquitectura BBE

La figura 2.3 muestra las diferentes capas dentro de la arquitectura de BBE. Los tres bloques principales en la arquitectura de un dispositivo BBE son: la aplicación, el anfitrión y el controlador. La descripción de las capas de la arquitectura BBE detalladas a continuación se obtuvo de [4].

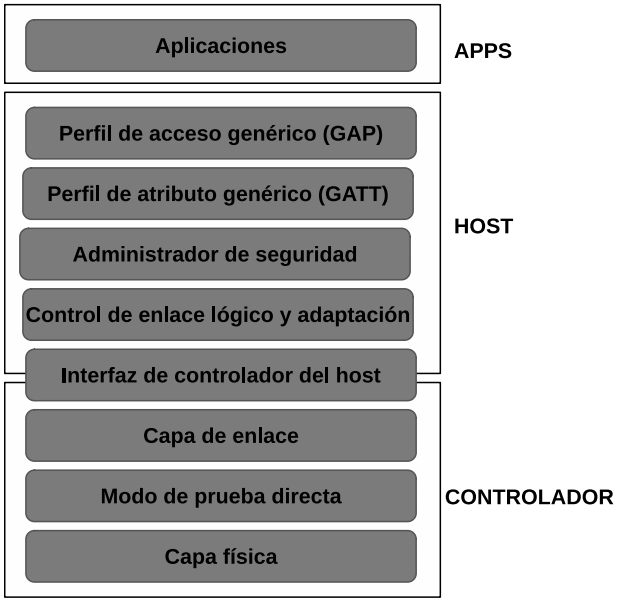


Figura 2.3: Arquitectura BBE.

La capa de aplicación depende del caso de uso que se le dé, se refiere a la implementación en la parte superior del Perfil de acceso genérico y el Perfil de atributo genérico. Es la forma en que una aplicación maneja los datos recibidos y enviados a otros dispositivos y la lógica detrás de ellos.

El controlador es el componente más básico de la arquitectura, contiene la capa física (PHY), la capa de enlace y el modo de prueba directa del BBE. El HCI (*anfitrión Controller Interface*) es el protocolo estándar que permite la comunicación entre el anfitrión y el controlador, se lleva a cabo a través de un interfaz serie. La capa de anfitrión es la parte que contiene a los protocolos fundamentales que permiten el acceso e intercambio de información en BBE. En el nivel más alto de la capa se encuentran de

forma paralela las capas GAP (*Generic Access Profile*) y GATT (*Generic Attribute Profile*), después se encuentra el administrador de seguridad y el protocolo L2CAP, este último es el encargado de dar acceso y soporte a los protocolos fundamentales, los cuales se describen con más detalle a continuación.

Perfil de acceso genérico (GAP)

GAP es el acrónimo para el perfil de acceso genérico (*Generic Access Profile*), se encarga de controlar las conexiones y los anuncios BBE. GAP es lo que permite que su dispositivo sea público hacia el exterior y determina como dos dispositivos pueden (o no) interactuar entre ellos. GAP define varios roles para los dispositivos, pero los dos conceptos clave a tener en cuenta son los dispositivos centrales y los dispositivos periféricos.

- Los periféricos son dispositivos pequeños, de bajo consumo de energía y con recursos limitados que pueden conectarse a un dispositivo central mucho más potente. Los dispositivos BBE son artefactos como un monitor de frecuencia cardíaca, sensor de proximidad habilitada para BBE, una baliza BBE, etc.

- Los dispositivos centrales suelen ser teléfonos móviles o tabletas a los que se conectan los dispositivos periféricos, estos tienen mucha más capacidad de procesamiento y memoria.

Hay dos maneras de transmitir información a través de GAP: Los datos de anuncio (*Advertising data payload*) y los datos de respuesta de escaneo (*Scan response payload*). Ambos tipos de datos son idénticos y pueden contener hasta 31 bytes, pero sólo los datos de anuncio son obligatorios, ya que se transmite continuamente desde el periférico, para permitir que los nodos centrales en el alcance sepan de su presencia. Los datos de respuesta del escaneo son opcionales y pueden ser pedidos desde un dispositivo central. De este modo los periféricos BBE pueden transmitir información extra como el nombre del dispositivo o alguna característica especial definida por el fabricante.

Un periférico emite los datos de anuncio a intervalos regulares configurables. Cada vez que el intervalo pasa, el periférico emite sus datos de anuncio. Intervalos altos, permiten ahorrar batería, mientras que intervalos cortos, permiten ser más reactivos. Si un dispositivo central necesita más datos y el periférico lo permite, puede solicitar adicionalmente la carga de respuesta de escaneo, y este contestará con la información adicional.

Si bien la mayoría de los periféricos se anuncian para que se pueda establecer una conexión y se puedan utilizar los servicios y las características del GATT (lo que permite intercambiar mucha más información y en ambos sentidos), hay situaciones en las que solo se desea enviar datos de anuncio. El principal caso de uso, es cuando se desea que un periférico transmita datos a más de un dispositivo a la vez. Esto sólo es posible utilizando los datos de anuncio ya que los datos enviados y recibidos en modo conectado solo pueden ser vistos por los dos dispositivos conectados.

Incluyendo una pequeña cantidad de datos personalizados en los 31 bytes de anuncio o de respuesta de escaneo, podemos usar un periférico BBE para enviar datos unidireccionalmente a dispositivos centrales dentro de la distancia de alcance. Esto es lo que se conoce como *Broadcasting* en BBE. Una vez que se establece la conexión entre un periférico y un dispositivo central, el proceso de anuncio suele detenerse y usará los servicios y características del GATT para comunicarse en ambas direcciones.

Perfil de atributos genéricos (GATT)

GATT es una especificación para enviar pequeños fragmentos de datos, denominados "atributos", a través de un vínculo BBE, es decir, define la forma en que dos dispositivos Bluetooth de baja energía transfieren datos de un lado a otro utilizando conceptos llamados servicios y características. Todos los perfiles de aplicaciones de bajo consumo actuales se basan en GATT. El SIG define muchos perfiles para dispositivos de bajo consumo. Un perfil es una especificación que describe cómo funciona un dispositivo en una aplicación determinada. Cabe señalar que un dispositivo puede implementar más de un perfil. Por ejemplo, un dispositivo puede incluir un monitor de ritmo cardíaco y un detector de nivel de batería.

GATT entra en juego una vez que se establece una conexión dedicada entre dos dispositivos, lo que significa que ya ha pasado por el proceso de anuncio regido por GAP. Lo más importante a tener en cuenta con el perfil GATT es que las conexiones son exclusivas. Lo que se significa que un periférico BBE sólo se puede conectar a un dispositivo central (un teléfono móvil, tableta, etc.) a la vez, tan pronto como un periférico se conecte a un dispositivo central, dejará de anunciarse y otros dispositivos ya no podrán verlo o conectarse a él hasta que se interrumpa la conexión existente. Establecer una conexión también es la única forma de permitir la comunicación bidireccional, donde el dispositivo central puede enviar datos significativos al periférico y viceversa.

GATT está basado en el protocolo de atributos (ATT). Por eso, también se denomina GATT/ATT. ATT está optimizado para funcionar en dispositivos BBE. Por esa razón, usa la menor cantidad de bytes posible. Cada atributo se identifica de forma exclusiva mediante un identificador único universal (UUID), que es un formato estandarizado de 128 bits para un ID empleado para identificar información de manera exclusiva. Los atributos transportados por ATT tienen formato de características y servicios. Un concepto importante que debe entenderse con el GATT es la relación servidor/cliente. El periférico se conoce como el servidor GATT , que contiene los datos de búsqueda de ATT y las definiciones de servicios y características, y el cliente GATT (el teléfono / tableta), que envía solicitudes a este servidor. Todas las transacciones son iniciadas por el dispositivo principal, el cliente GATT, que recibe respuesta del dispositivo secundario, el servidor GATT.

Al establecer una conexión, el periférico sugerirá un "Intervalo de conexión" al dispositivo central, y el dispositivo central intentará reconectar cada intervalo de conexión para ver si hay nuevos datos disponibles, etc. Es importante tener en cuenta que esta conexión, sin embargo, el intervalo es sólo una sugerencia, es posible que su

dispositivo central no pueda cumplir con la solicitud porque está ocupado hablando con otro periférico o porque los recursos del sistema requeridos simplemente no están disponibles.

El diagrama de la figura 2.4 ilustra el proceso de intercambio de datos entre un periférico (el servidor GATT) y un dispositivo central (el cliente GATT), con el dispositivo principal iniciando cada transacción, se observa los datos publicitarios o anuncios emitidos por la baliza BBE y la solicitud de respuesta de escaneo enviada por el dispositivo central:

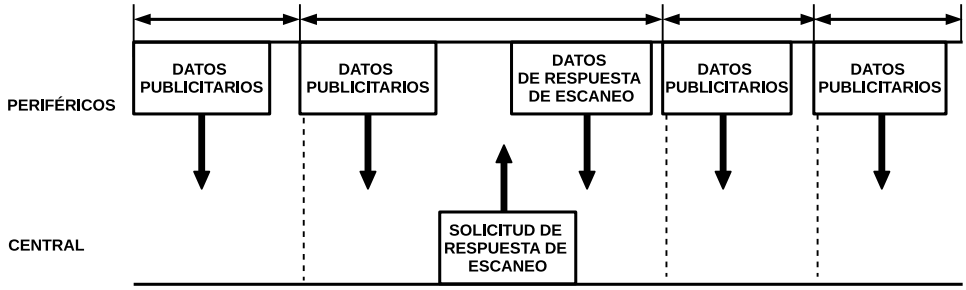


Figura 2.4: Transacción GATT.

Las transacciones GATT en el BBE se basan en objetos anidados de alto nivel llamados Perfiles, Servicios y Características, la figura 2.5 muestra la la estructura anidada de los objetos, donde los servicios engloban a las características y el perfil contiene a los servicios, a continuación se describirán cada uno de estos objetos.

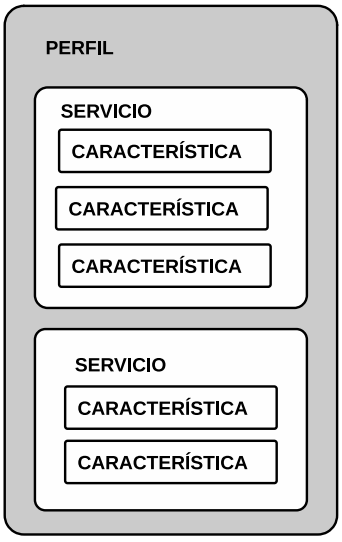


Figura 2.5: Estructura de objetos GATT.

- Perfil: Es una colección predefinida de servicios que ha sido compilada por Bluetooth SIG o por los diseñadores de periféricos. El perfil de frecuencia cardíaca, por ejemplo, combina el servicio de frecuencia cardíaca y el servicio de

información del dispositivo. La lista completa de perfiles basados en el GATT adoptados oficialmente se puede ver en el sitio: https://www.bluetooth.com/specifications/specs/.

- Característica: Una característica contiene un solo valor y 0-n descriptores que describen el valor de la característica. Una característica puede interpretarse como un tipo, que es similar a una clase.
- Descriptor: Los descriptores son atributos definidos que describen el valor de una característica. Por ejemplo, un descriptor puede especificar una descripción en lenguaje natural, un rango aceptable para el valor de una característica o una unidad de medida específica del valor de una característica.
- Servicio: Un servicio es una colección de características. Por ejemplo, podría tener un servicio denominado "Monitor de ritmo cardíaco" que incluya características como "medición del ritmo cardíaco". Puede consultar una lista de los perfiles y servicios basados en GATT en el sitio: https://www.bluetooth.com/.

Interfaz de controlador del anfitrión (HCI)

La capa HCI es un protocolo estándar definido por la especificación de Bluetooth que permite que la capa de anfitrión se comunique con la capa del controlador. En el caso de que el anfitrión y el controlador estén en conjuntos de chips separados, la capa HCI se implementará sobre una interfaz de comunicación física. Las tres interfaces de hardware oficialmente admitidas por la especificación son: UART, USB y SDIO (salida de entrada digital segura). En el caso de que las dos capas anfitrión y controlador) vivan en el mismo microcontrolador, la capa HCI será una interfaz lógica.

Protocolo de Adaptación y Control de Enlace Lógico (L2CAP)

La capa de Protocolo de Adaptación y Control de Enlace Lógico (L2CAP) actúa como una capa de multiplexación de protocolo. Se toma prestado del estándar Bluetooth clásico y realiza las siguientes tareas en el caso de BBE:

- Toma múltiples protocolos de las capas superiores y los coloca en paquetes BBE estándar que se pasan a las capas inferiores debajo de él.
- Maneja la fragmentación y la recombinación.
- Toma los paquetes más grandes de las capas superiores y los divide en trozos que se ajustan al tamaño máximo de carga útil BBE admitido para la transmisión.
- En el lado del receptor, toma varios paquetes y los combina en un paquete que puede ser manejado por las capas superiores.
- La capa L2CAP maneja dos protocolos principales: el protocolo de atributos (ATT) y el protocolo de administrador de seguridad (SMP).

Administrador de seguridad

El administrador de seguridad define los protocolos y algoritmos para generar e intercambiar llaves entre dos dispositivos. Incluye cinco características de seguridad:

1. Emparejamiento: el proceso de crear llaves secretas compartidas entre dos dispositivos.

2. Vinculación: el proceso de crear y almacenar llaves secretas compartidas en cada lado (central y periférico) para su uso en conexiones posteriores entre los dispositivos.

3. Autenticación: el proceso de verificar que los dos dispositivos comparten las mismas llaves secretas.

4. Cifrado: el proceso de cifrar los datos intercambiados entre los dispositivos.El cifrado en BBE utiliza el estándar de cifrado AES de 128 bits, que es un algoritmo de llave simétrica (lo que significa que se utiliza la misma llave para cifrar y descifrar los datos en ambos lados).

5. Integridad del mensaje: el proceso de firmar los datos y verificar la firma en el otro extremo. Esto va más allá de la simple verificación de integridad de un CRC calculado.

El administrador de seguridad fue contemplado a partir de la versión 4.2 de Bluetooth, sin embargo, su implementación depende de los dispositivos, las aplicaciones y sus administradores, en la mayoría de los casos no se utilizan las características de seguridad debido a los recursos limitados de los dispositivos BBE y a que la criptografía implementada por la especificación es clásica (poco ligera).

2.1.5. Vulnerabilidades en sistemas BBE

El protocolo BBE permite de manera ubicua la comunicación inalámbrica energéticamente eficiente entre dispositivos con recursos limitados. Para facilitar su adopción, BBE requiere una interacción limitada o nula del usuario para establecer una conexión entre dos dispositivos. Desafortunadamente, esta simplicidad es la causa principal de varios problemas de seguridad. Los dispositivos BBE para conectarse, deben emparejarse primero, y aquí es donde reside la principal vulnerabilidad de los sistemas BBE. Durante la primera etapa de emparejamiento, los dispositivos intercambian información básica sobre sus capacidades para descubrir cómo proceder con la conexión. Es decir, se identifican en la red y explican qué son (una baliza, un teclado, un auricular, etc.) y qué pueden hacer. Este intercambio no está cifrado. La segunda fase de emparejamiento está dedicada a generar e intercambiar llaves. Es en este punto que las conexiones BBE pueden ser manipuladas: si la conexión no está asegurada adecuadamente, los atacantes pueden tomar el control de los dispositivos y los datos

que transmiten. Por último, la vinculación es el proceso durante el cual los dispositivos almacenan los datos de autenticación que intercambiaron durante el primer emparejamiento, lo que les permite recordarse mutuamente como seguros cuando se vuelven a conectar en el futuro.

Existen métodos para asegurar la segunda fase, como el uso de llaves temporales para autorizar la conexión o el uso de BBE seguro, en la versión 4.2 de Bluetooth se implementa el algoritmo Diffie-Hellman para la generación de llaves, además de introducir un proceso más complejo de autenticación. Pero lo cierto es que muchos de los dispositivos IdC que se encuentran en el mercado no implementan estos métodos y son fácilmente vulnerables. Hay dos tipos de ataques comúnmente asociados a los módulos BBE: escuchas pasivas y el hombre intermedio.

- En la escucha pasiva un atacante puede escuchar los datos que los periféricos BBE envían a la unidad central y luego usarlos para descubrir otras vulnerabilidades de seguridad en el sistema.

- Un ataque de hombre intermedio involucra un dispositivo malicioso que pretende ser central y periférico al mismo tiempo y engaña a otros dispositivos de la red para que se conecten a él. En este caso, un dispositivo externo puede no sólo ver el tráfico de los equipos conectados, sino también inyectar datos falsos en la secuencia y provocar el mal funcionamiento de cadenas de producción completas. La figura 2.6 ilustra de manera sencilla un ataque de hombre intermedio en un sistema BBE, en el cual un adversario intercepta la información de un dispositivo BBE.

Figura 2.6: Ataque de hombre intermedio.

2.2. Tecnología de balizas BBE

Se trata de una de las industrias tecnológicas de más rápido crecimiento en los últimos años. En 2015 se detectaron cerca de cuatro millones de balizas BBE desple-

gadas en todo el mundo, y se tiene un estimado de 60 millones de balizas BBE que serán distribuidas para este año [5].

Las balizas (en inglés conocidas como *beacons*) son dispositivos que transmiten un identificador a través de una señal Bluetooth de baja energía, estos permiten transmitir mensajes o avisos directamente a un dispositivo móvil, es decir, actúan como un pequeño faro digital que puede "despertar" a otros dispositivos que estén escuchando. Para que los dispositivos se "despierten" se debe tener instalada una aplicación que esté escuchando y que reconozca la señal de esa baliza, entonces se realizará algún tipo de acción, por ejemplo, mostrar una notificación con un mensaje. Las balizas BBE vienen en diferentes tamaños y formas, desde pequeñas monedas planas hasta imitaciones de rocas de mayor tamaño, como tarjetas, pulseras y más. El factor de forma depende de su caso de uso y de los requisitos específicos del entorno en el que se utilizará la baliza. Básicamente el hardware de las balizas consiste en un microcontrolador con un chip de radio BBE y una batería, generalmente de botón, con la que pueden durar más de un año, aun así existen otro tipo de balizas que funcionan externamente y se pueden instalar en un enchufe o en un puerto USB, con la desventaja que no pueden ser ubicadas en cualquier sitio.

La trama de la señal transmitida por las balizas BBE se crea siguiendo las directrices marcadas por un protocolo de baliza, todos los protocolos de comunicación cumplen con la especificación de Bluetooth de baja energía. Dentro de cada trama, está la información que utiliza el dispositivo para filtrar las señales BBE de anuncios que cumplan con el estándar buscado. Los dispositivos móviles con soporte BBE tienen, en los principales sistemas operativos (iOS y Android), la posibilidad de detectar los protocolos compatibles siempre que el Bluetooth esté encendido. Aun así, es necesario desarrollar siempre una aplicación que haga uso de las balizas detectadas por el sistema, y en función de si es una baliza asociada a la aplicación o no, interactuar con el usuario.

2.2.1. Aplicaciones de balizas BBE

En la figura 2.7 se muestra la interacción de los usuarios con diversas aplicaciones que implementan la tecnología de balizas BBE, algunas de estas aplicaciones colocan balizas en lugares estratégicos para enviar mensajes a los usuarios que pasen cerca, ya sea que les den información del lugar o incluso activen notificaciones como técnica de marketing.

Algunos de los principales sectores en donde la tecnología de balizas BBE se ha hecho presente en los últimos años son los siguientes:

- Museos: Las aplicaciones con balizas BBE en los museos realizan un rápido análisis de grandes cantidades de datos, implementan la navegación interna en espacios cerrados, por lo cual es posible saber la productividad casi exacta de cada área del museo, permitiendo la mejor toma de decisiones, como la gestión del tiempo en que mantendrán una obra de arte en exposición, evitando gastos innecesarios.

Figura 2.7: Aplicaciones de balizas BBE.

- Acuarios y zoológicos: En esta área las aplicaciones con balizas BBE proporcionan a los usuarios información interactiva (como juegos y vídeos) de la especie por donde se este pasando, audio guías y geolocalización precisa dentro del lugar.

- Mercadotecnia: La función de las aplicaciones con balizas BBE en este sector es ofrecer una mejor experiencia de compra a los usuarios. La aplicación de una tienda provee fácilmente la ubicación exacta del consumidor, mediante el uso de las señales de balizas BBE. De esa forma, se le envían al teléfono del usuario, alertas y notificaciones con promociones y descuentos de servicios y/o productos.

- Turismo: En muchas categorías de aplicaciones móviles turísticas, la personalización y la ubicación exacta son claves para la experiencia excepcional del usuario. Con este tipo de aplicaciones se pueden recomendar sitios de interés, eventos próximos y dar infografías a los turistas, esto se puede realizar colocando balizas BBE en lugares estratégicos de la zona, donde al pasar un usuario (con la aplicación turística) por alguna de ellas se activen las recomendaciones.

- Educación: Las aplicaciones con balizas BBE en el campo de la educación pueden enfocarse en el control de acceso a aplicaciones como WhatsApp o Facebook, o incluso bloquear inmediatamente el acceso a internet en los teléfonos inteligentes. Además, La presencia de los alumnos en el aula se podría comprobar mediante las balizas BBE.

- Servicios de localización: En este sector se utilizan balizas móviles en servicios de localización y seguimiento de mascotas o artículos. Por ejemplo, el rastreo de equipaje en los aeropuertos, ya que el usuario no cuenta con información de cuando llegará su equipaje a la cinta transportadora. Con una baliza BBE dentro del equipaje, es posible realizar el seguimiento del mismo y recibir notificaciones al respecto.

- Control de acceso e identificación automática: Se pueden utilizar las balizas BBE para ingresar a un hotel, omitir el proceso de registro de entrada, ir directamente

a la habitación y abrir la puerta sin necesidad de llave o tarjeta, todo esto para evitar el contacto con otras personas (importante en estos tiempos de pandemia) y agilizar el proceso de *check-in*, gracias a las balizas BBE y una aplicación para teléfonos inteligentes.

- Rastreo de contactos: Actualmente en estos tiempos de pandemia, se han desarrollado múltiples aplicaciones que apoyan a la lucha contra el COVID-19, este tipo de aplicaciones emplean balizas BBE y teléfonos inteligentes para realizar un rastreo de contactos que se hayan registrado a través de identificadores emitidos por señales BBE, y posteriormente dar una alerta en caso de que alguno de ellos haya dado positivo a una prueba de la enfermedad. Un ejemplo de aplicación de este tipo es Applacovid [2].

2.2.2. Amenazas de seguridad en aplicaciones con balizas BBE

Existen diversas amenazas y ataques a la seguridad en las aplicaciones que utilizan balizas BBE, entre las que destacan las siguientes:

- Seguimiento no autorizado. En este caso los identificadores (ID) de la transmisión de una baliza BBE están disponibles para que los lea cualquier receptor cercano. Esto permite muchos abusos, incluido el seguimiento no autorizado. La figura 2.8 ilustra un esquema básico del ataque de seguimiento no autorizado, el cual consiste en que un espía pueda usar una aplicación móvil para recopilar identificadores y después usarlos para inferir la ubicación histórica de las balizas. Esta amenaza es un problema en las aplicaciones donde las balizas se encuentran estáticas y al alcance del público, como en los museos, estadios, acuarios y zoológicos, debido a que podrían ser ubicadas para posteriormente ser robadas o suplantadas.

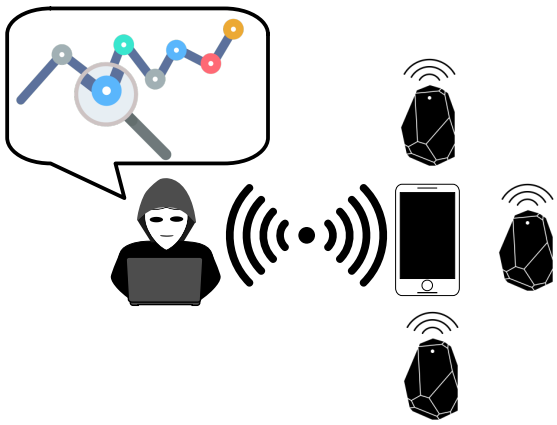


Figura 2.8: Ataque de seguimiento no autorizado.

Para este tipo de amenaza la contramedida de seguridad sugerida es la generación y transmisión de identificadores temporales a través de algoritmos de cifrado confiables, con ello la aplicación que recolecte estos identificadores no sabrá a que dispositivo corresponde cada identificador, imposibilitando de esta forma el rastreo sin autorización.

- Suplantación o *(Spoofing)*. Bajo el ataque de suplantación, un adversario puede espiar o interceptar el mensaje de solicitud de inicio de sesión de sesiones anteriores a través del canal público/abierto durante la ejecución del protocolo de autenticación [5]. Por ejemplo, en las aplicaciones de control de acceso y rastreo de contactos mencionadas anteriormente, esta amenaza es muy preocupante debido a que la suplantación de identidad sería un problema grave para la seguridad de las aplicaciones con balizas y sus usuarios. En la figura 2.9 se ilustra un esquema sencillo del ataque de suplantación de balizas BBE, en el cual se observa un atacante que intercepta los identificadores emitidos por una baliza y posteriormente los utiliza para hacerse pasar por ella en la interacción con la aplicación móvil.

Figura 2.9: Ataque de suplantación.

Una consecuencia del ataque de suplantación es el puede darse en las visitas a tiendas, debido a que las balizas BBE se utilizan a menudo para identificar la micro ubicación de los usuarios, un adversario puede usar la información de las balizas para ofrecer servicios que compitan con los que se brindan en las ubicaciones de los clientes registrados [6]. En donde podría darse la suplantación es en las aplicaciones de publicidad de los centros comerciales, ya que los negocios se verían afectados al enviarse otras opciones de compra y ofertas a los usuarios, ocasionando el robo de clientes a las empresas que emplean las aplicaciones. En este tipo de ataque se sugiere implementar servicios de seguridad como la autenticación a través de un protocolo de acuerdo de llaves, además la generación de identificadores efímeros disminuye la probabilidad de ataques de suplantación de balizas BBE.

- Falsificación de información. Un adversario podría falsificar los mensajes de cualquier baliza que emita una identificación coherente. Esta amenaza está presente en prácticamente todas las aplicaciones que utilizan balizas BBE, por ejemplo,

la figura 2.10 ilustra un ejemplo de ataque de falsificación en una aplicación del sector turístico, en donde los mensajes con información falsa podrían causar que un usuario se pierda, no encuentre los lugares o eventos anunciados y por lo tanto decida dejar de utilizar la aplicación.

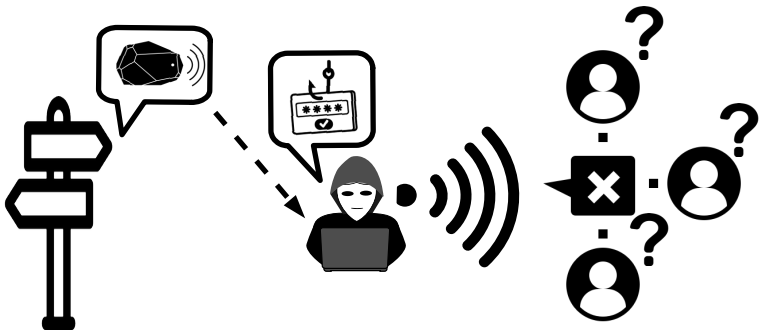


Figura 2.10: Ataque de falsificación de información.

En este caso los mecanismos de seguridad que se proponen son el cifrado y autenticado de los mensajes emitidos por las aplicaciones con balizas BBE, así como también la trasmisión de identificadores efímeros generados a partir de funciones pseudoaleatorias.

2.2.3. Protocolos de balizas BBE

Así como el Wi-Fi y el Bluetooth son estándares de comunicación por radio, los protocolos de balizas son estándares de comunicación BBE. Son perfiles que muestran la forma en que una baliza está transmitiendo información, esta es manejada por un dispositivo (generalmente un teléfono celular) con el mismo protocolo. Ambos extremos deben conocer la definición de los datos de transmisión si desean una transmisión exitosa, esta es la razón por la cual el tipo de datos de difusión está definido por el tipo de protocolo que está usando la baliza. La figura 2.11 muestra la forma general de un paquete de anuncio transmitido a través de una señal BBE.

En las secciones siguientes se describen los principales protocolos paran la comunicación entre las aplicaciones y las balizas BBE.

iBeacon

iBeacon es un estándar de tecnología desarrollado por Apple [7] que permite a las aplicaciones móviles escuchar las señales recibidas desde balizas que se encuentran bien colocadas en el mundo físico. En función de las señales, las aplicaciones móviles reaccionan de diversas maneras. La tecnología permite a las aplicaciones móviles

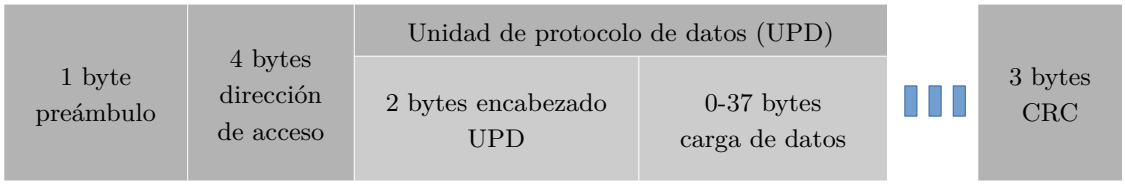


Figura 2.11: Paquete de anuncio BBE.

iBeacon entender la posición del dispositivo, en una escala microlocal y entregar contenido hipercontextual.

El protocolo iBeacon también incluye un campo que permite el cálculo de la distancia relativa desde la baliza. Las aplicaciones que coinciden con el identificador universal de la baliza, registrado en su base de datos pueden realizar tareas específicas basándose en el conocimiento de que la baliza está cerca. En la mayoría de las aplicaciones, esto se utiliza para proporcionar el concepto de monitoreo de región, donde ciertas funciones dependen de la ubicación del usuario dentro de la "región" creada por una baliza o una red de balizas. Al asignar un conjunto de balizas a un conjunto predeterminado de regiones, las aplicaciones conocerán la ubicación real del usuario sin el uso de Sistema de Posicionamiento Global. La figura 2.12 muestra la configuración del paquete de datos transmitido por el protocolo iBeacon.

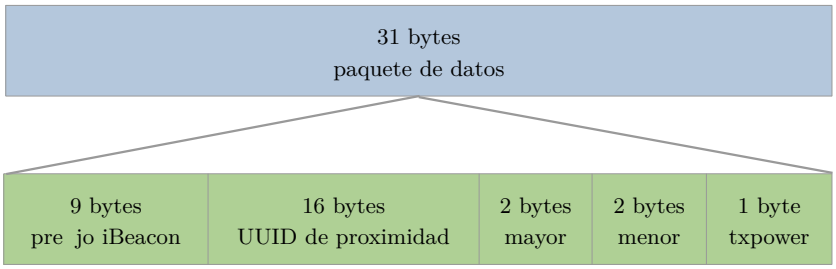


Figura 2.12: Paquete de anuncio iBeacon.

AltBeacon

AltBeacon es el protocolo desarrollado por la compañía Radius Networks [8]. Fue creado como una alternativa al protocolo cerrado iBeacon, ofreciendo las mismas funcionalidades pero siendo capaz de entregar más información en cada mensaje emitido. La especificación AltBeacon define el formato del mensaje publicitario que emiten las balizas de proximidad BBE, está destinada a crear un mercado abierto y competitivo para las implementaciones de balizas de proximidad. La funcionalidad de la baliza de

proximidad AltBeacon no se limita a los dispositivos de una sola función, sino que se puede incorporar como una característica de cualquier dispositivo que sea compatible con BBE y que cumpla con los requisitos definidos en la especificación de Bluetooth versión 4.0. El anuncio de AltBeacon se compone de un campo de 1 byte de longitud, un campo de 1 byte y un identificador de empresa de dos bytes, según lo prescrito por el formato de estructura de datos publicitarios específicos del fabricante, seguido de 24 bytes adicionales que contienen los datos del anuncio de la baliza BBE como se observa en la figura 2.13.

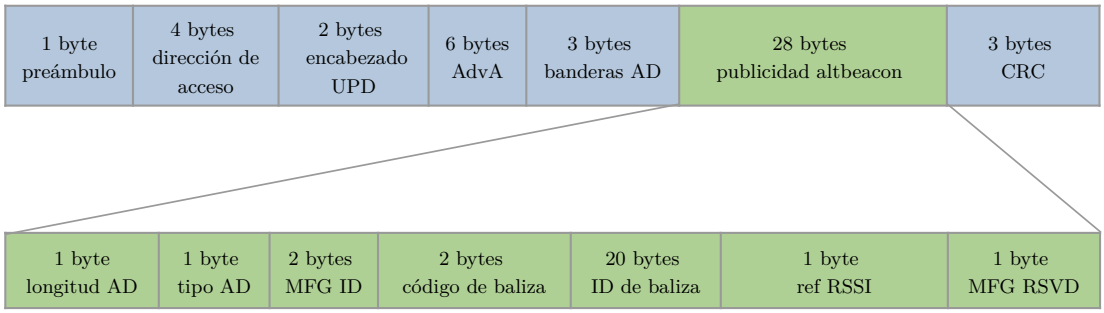


Figura 2.13: Paquete de anuncio AltBeacon.

Eddystone

Eddystone es un formato de baliza abierto desarrollado por Google [6] y diseñado teniendo en cuenta la transparencia y la solidez. Eddystone puede ser detectado tanto por dispositivos Android como iOS. La figura 2.14 muestra la configuración general del paquete de datos transmitido por el protocolo de Eddystone. El formato Eddystone

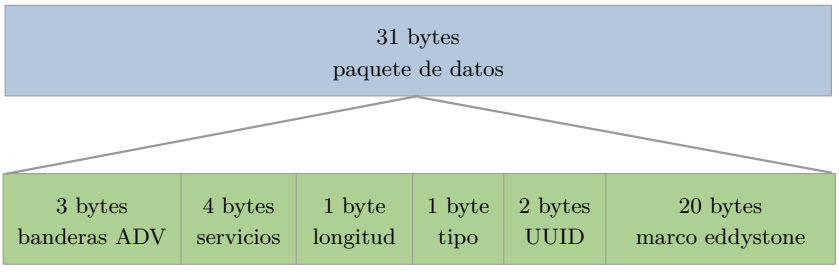


Figura 2.14: Paquete de anuncio Eddystone.

se basa en las lecciones aprendidas al trabajar con socios de la industria en implementaciones existentes, así como en la comunidad de balizas en general. Una baliza configurada con este protocolo puede emitir uno de los siguientes tipos de paquetes:

1. Eddystone-UID: Un ID estático único con un componente de espacio de nombres de 10 bytes y un componente de instancia de 6 bytes.

2. Eddystone-URL: Una URL comprimida que, una vez analizada y descomprimida, el cliente puede utilizarla directamente.

3. Eddystone-TLM: Datos de estado de la baliza que son útiles para el mantenimiento de la flota de balizas y potencia el punto final de diagnóstico de la API de balizas de proximidad de Google. TLM debe estar intercalado con un marco de identificación como Eddystone-UID o Eddystone-EID (para el cual la versión cifrada de eTLM preserva la seguridad).

4. Eddystone-EID[1]: Es un componente de la especificación Eddystone para balizas BBE. Las balizas Eddystone-EID emiten un identificador que cambia cada pocos minutos. El identificador puede resolverse en información útil mediante un servicio que comparte una llave (la llave de identidad efímera) con la baliza individual. Los IDE se generan utilizando medios criptográficos y sólo se pueden vincular a la baliza o información asociado a la baliza, por usuarios autorizados. Por lo tanto, el propietario de la baliza puede identificar el IDE mientras para otros dispositivos, se vería como una identificación aleatoria [6].

Eddystone-EID

El protocolo Eddystone-EID cuenta con las características siguientes:

1. Cada baliza BBE tiene una llave secreta con la que se identifica. La llave la conoce el propietario de la baliza.

2. La información de identificación en los anuncios de baliza consiste en una función pseudoaleatoria determinista de la hora actual (con la precisión correcta) tecleada con la llave secreta de baliza. A este valor lo llamamos “ID efímero”.

3. Siempre que el propietario de una baliza observe un anuncio, puede calcular su propia versión del ID efímero y compararlo con el ID recibido. De esta manera, el propietario de la baliza puede identificar la baliza, un proceso que llamamos “resolución local del ID efímero”.

4. El propietario de la baliza puede registrar la baliza con un resolvedor global confiable de ID efímeros. Durante el proceso de registro, el propietario envía la llave compartida con la baliza al resolvedor.

5. Cualquier dispositivo que observe un anuncio puede transferir los datos observados al resolvedor global. Luego, el resolvedor intenta identificar la baliza usando todas las llaves registradas. A este proceso lo llamamos “resolución en la nube global de la identificación efímera”.

[1]Eddystone Ephemeral Identifier

6. El resolvedor global puede notificar al dispositivo que observó la baliza y/o al propietario con información derivada de la identidad de la baliza, de acuerdo con la política de permisos de uso compartido correspondiente. Cualquier información de este tipo se transmite a través de canales seguros.

Los métodos de seguridad del protocolo utilizan un esquema criptográfico simple, construido principalmente alrededor del uso de AES-128 (Advanced Encryption Standard) en varios modos de operación y combinaciones de los mismos. Se elige AES debido a su alta ubicuidad en los dispositivos Bluetooth (dado que ya se usa en partes de la especificación Bluetooth), así como por el alto nivel de confianza en su seguridad. Para mayor eficiencia, AES se usa casi únicamente en una dirección (cifrado), ya sea como una función pseudoaleatoria simple (PRF) o en un modo de cifrado autenticado (EAX) [6]. Para dar una solución eficiente a las amenazas de seguridad y privacidad en las balizas BBE, el protocolo Eddystone-EID propone que se deben proteger tres tipos de datos:

1. Direcciones MAC de Bluetooth.

2. Identificadores de dispositivos a nivel de aplicación.

3. Estado mutable de transmisión específico del dispositivo, es decir, las características de conexión en la capa de enlace del Bluetooth de baja energía.

Este intento de solucionar la falta de mecanismos de seguridad en aplicaciones con balizas BBE propuesto por los desarrolladores de Google, en el que las balizas emitían un identificador que cambiaba cada pocos minutos; el proceso de generación de IDEs se realizaba a través de un resolvedor global y un servicio de Google (Proximity Beacon API) que lamentablemente ha quedado obsoleto el 7 de diciembre de 2020, y se cerró definitivamente el 1 de abril de 2021.

Los datos generados a partir de la tecnología de baliza BBE que se producirán a partir de miles de interacciones entre estos dispositivos y sus usuarios, no sólo serán masivos, sino también complejos y sufrirán muchos problemas de seguridad y privacidad, especialmente en lo que respecta a la autenticación entre dispositivos debido a los ataques de falsificación y suplantación. Para resolver estos problemas de seguridad, los investigadores en el campo de la seguridad informática proponen el desarrollo de protocolos de seguridad aplicados en este contexto.

Capítulo 3

Primitivas criptográficas

El surgimiento de diversas redes de comunicación, desde las de alcance mundial como el internet, hasta las de área personal como el Bluetooth, así como las aplicaciones que las utilizan, han abierto nuevas posibilidades para el intercambio de información entre los dispositivos. Al mismo tiempo, son cada vez mayores las amenazas a la seguridad de la información que se transmite. Por eso, es necesario crear diferentes mecanismos de seguridad, dirigidos a garantizar la confidencialidad y autenticidad de los dispositivos y sus datos, todos estos mecanismos o servicios de seguridad son proporcionados por la criptografía. La criptografía se divide en algoritmos de llave secreta o simétricos y de llave pública o asimétricos. Los algoritmos de llave secreta necesitan acordar una llave previamente entre las partes que se van a comunicar, lo cual se puede hacer mediante el uso de un algoritmo de llave pública. A continuación se abordará específicamente la variante de llave pública.

La criptografía asimétrica, es el método criptográfico que usa un par de llaves para el envío de mensajes. Las dos llaves pertenecen a la misma entidad que recibirá el mensaje, una llave es pública y se puede entregar a cualquiera, la otra llave es privada y el propietario debe guardarla de modo que nadie tenga acceso a ella, dichas llaves mantienen una relación matemática que permite el funcionamiento del algoritmo. Además, los métodos criptográficos garantizan que esa pareja de llaves sólo se puede generar una vez, de modo que se puede asumir que no es posible que dos entidades hayan obtenido casualmente la misma pareja de llaves. Este tipo de algoritmos criptográficos garantizan el resto de propiedades que los algoritmos simétricos no pueden garantizar: permiten establecer una llave a través de un canal inseguro, identificar a cada entidad que interviene, cifrar mensajes y asegurar la integridad de los mismos. Uno de los métodos de llave pública más utilizados en la actualidad son los criptosistemas de llave pública basados en curvas elípticas. A continuación se definen algunas estructuras algebraicas utilizadas en este Capítulo.

3.1. Preliminares

Definición 3.1.1 *Un grupo consiste de un conjunto $\mathbb{G}$ y una operación binaria $*$, para todo $a, b \in G$, $a * b \in \mathbb{G}$. Dicha operación cumple las siguientes propiedades:*

- ***Elemento neutro.*** *Existe e tal que para todo $a \in \mathbb{G}$, $e * a = a * e = a$.*
- ***Unicidad del inverso.*** *Para cada $a \in \mathbb{G}$ existe un valor único $a^{-1} \in \mathbb{G}$ tal que $a * a^{-1} = e$.*
- ***Asociatividad.*** *$a * (b * c) = (a * b) * c$ para todos $a, b, c \in \mathbb{G}$.*

Si además $a * b = b * a$, es decir $*$ es conmutativa, el grupo es llamado abeliano.

Definición 3.1.2 *Un grupo $\mathbb{G}$ se dice cíclico si existe $x \in \mathbb{G}$ tal que todos los valores $b_i = x^i$ para $i \in \mathbb{N}$ son diferentes y el valor i tal que $x^i = e$ coincide con el número de elementos en $\mathbb{G}$.*

Definición 3.1.3 *Un anillo es un conjunto $\mathbb{A} \neq \emptyset$ dotado de dos operaciones binarias internas "$+$" y "$\cdot$" verificando: A es un grupo abeliano con la operación "$+$" y para todo $a, b, c \in A$.*

Definición 3.1.4 *Un campo $\mathbb{K}$ es un anillo conmutativo con identidad en el que todo elemento no nulo tiene inverso.*

Definición 3.1.5 *Sea $p > 0$ una característica esencial de los campos finitos, específicamente cuando $p = 2$ son llamados campos binarios o de característica dos.*

Definición 3.1.6 *Sea p primo. Al anillo entero $\mathbb{Z}_p$ se llama campo primo. Notar que además de anillo es un campo finito.*

Es fácil ver que $(\mathbb{Z}_p, +, \cdot)$ donde " $+$ " denota la suma ordinaria modulo p y " $\cdot$ " denota el producto ordinario modulo p, es un anillo conmutativo con elemento neutro $0_{\mathbb{Z}_1}$ y elemento identidad $1_{\mathbb{Z}_1}$. Además como p es primo, entonces $(\mathbb{Z}_p, +, \cdot)$ es campo primo.

Definición 3.1.7 *Sea $\mathbb{K}$ un campo finito, entonces $|\mathbb{K}| = p^n$ para algún primo p y algún entero n. Recíprocamente para todo primo p y entero positivo n, existe un único campo, salvo isomorfismos, de p^n elementos. Al campo de p^n elementos se le denota $GF(p^n)$.*

3.2. Criptografía de curva elíptica

La criptografía de curva elíptica (ECC) pertenece a la criptografía asimétrica, debido a que se utilizan dos tipos de llaves distintas, una pública y una privada, en la que el conocimiento de la llave pública no permite determinar la llave privada. La criptografía de curvas elípticas fue propuesta en la década de los ochenta por Neal

Koblitz [9] y Víctor Miller [10]. Desde entonces una gran cantidad de investigaciones se han realizado para tener implementaciones eficientes y seguras de estos esquemas criptográficos. Uno de los principales beneficios al emplear ECC es que se pueden usar llaves más pequeñas que las utilizadas en otros criptosistemas clásicos como el RSA y ElGamal, manteniendo la misma seguridad. De igual manera ofrece un abanico de grupos en el mismo campo base. La implementación de ECC puede llegar a ser hasta 15 veces más rápida que los criptosistemas clásicos, dependiendo de la plataforma sobre la que se aplique, estas ventajas son importantes en los dispositivos inalámbricos restringidos, en los que la potencia de cálculo, la memoria y la duración de la batería son limitados.

3.2.1. Curva elíptica

Se define de manera general a las curvas elípticas de la siguiente forma. Sea $\mathbb{K}$ un campo, una curva elíptica sobre $\mathbb{K}$, se define mediante la ecuación de Weierstrass 3.1:

$$y^2 = x^3 + ax + b \tag{3.1}$$

donde $x, y, a, b \in \mathbb{K}$. En la práctica, por lo general $\mathbb{K}$ es un campo finito primo $GF(p)$ o un campo binario $GF(2^m)$. En este trabajo se utilizan las curvas elípticas sobre un campo finito primo $GF(p)$. El campo de $GF(p)$ contiene los elementos $0, \ldots, p-1$, si se trabaja solamente con el grupo multiplicativo, los elementos son $1, \ldots, p-1$. Las operaciones de suma y multiplicación deben ser reducidas modulo p. La definición formal de una curva elíptica es como sigue:

Definición 3.2.1 *Una curva elíptica* $\mathbf{E}$ *sobre un campo* $\mathbb{K}$ *es el conjunto de puntos* $(x, y) \in \mathbb{K} \times \mathbb{K}$ *que satisfacen la ecuación 3.1. Además de un punto especial llamado punto en el infinito y denotado como* P_∞.

3.2.2. Operaciones sobre curvas elípticas

Se define una operación para los puntos de $\mathbf{E}$ que se denota por $+$. Para visualizarlo mejor se considera primero que $\mathbb{K} = \mathbb{R}$, de manera que la curva elíptica es ahora una curva ordinaria en el plano. Sea l una recta del plano, entonces la recta l y la curva $\mathbf{E}$ tienen tres puntos de intersección P, Q, R [11]. A continuación, se definen las operaciones de suma y doblado de puntos sobre una curva elíptica, así como el producto de un punto por un escalar.

Suma de puntos

Dados $P, Q \in \mathbf{E}$ con $P \neq Q$, se define la suma de dos puntos $P + Q$ como el simétrico al punto de corte entre la recta que une a P y Q y a la curva $\mathbf{E}$, como se muestra en la figura 3.1. En el caso particular de que la recta que conecta P y Q

sea tangente a la curva en P, se define $P + Q$ como el simétrico a P. Se obtiene la ecuación 3.2 para la suma de puntos:

$$P + Q = (x_3, y_3) = \left(\left(\frac{y_2 - y_1}{x_2 - x_1} \right)^2 - x_1 - x_2, -y_1 + \left(\frac{y_2 - y_1}{x_2 - x_1} \right) (x_1 - x_3) \right) \tag{3.2}$$

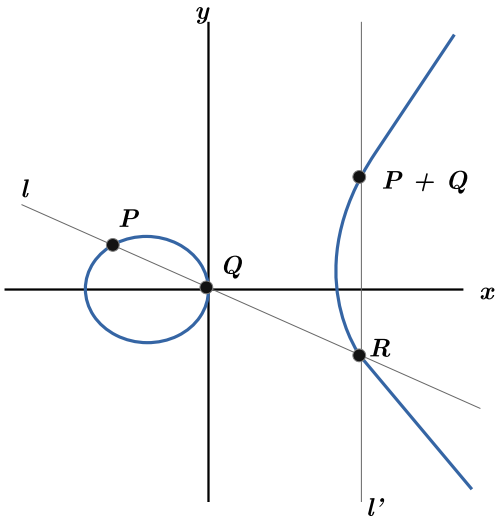


Figura 3.1: Gráfica de suma de dos puntos en $\mathbb{R}$.

Doblado de punto

Dado P, se define doblar un punto $P + P$ como el simétrico al punto de corte entre la recta tangente a P, que se denota por d, y la curva $\mathbf{E}$, como se muestra en la figura 3.2. En el caso concreto de que el punto P sea un punto de inflexión de la curva, se define $P + P$ como el simétrico a P. De igual manera que para la suma, para el doblado de puntos se obtiene la ecuación 3.3.

$$P + P = (x_2, y_2) = \left(\left(\frac{3x_1^2 + a}{2y_1} \right)^2 - 2x_1, -y_1 + \left(\frac{3x_1^2 + a}{2y_1} \right) (x_1 - x_3) \right) \tag{3.3}$$

En el caso general, es decir, sea cual sea la curva elíptica $\mathbf{E}$ (ecuación generalizada, ecuación reducida, etc.), se tiene que $\mathbf{E}$ con la operación $+$ forma un grupo abeliano.

Multiplicación escalar

Una de las operaciones más utilizadas en criptografía es la multiplicación escalar, la cual consiste en multiplicar un escalar por un punto. Por ejemplo: $2 \times P = P + P$. Por lo tanto, el escalar dice cuántas veces debe sumarse un mismo punto. Para hacerlo, existe el siguiente método analítico:

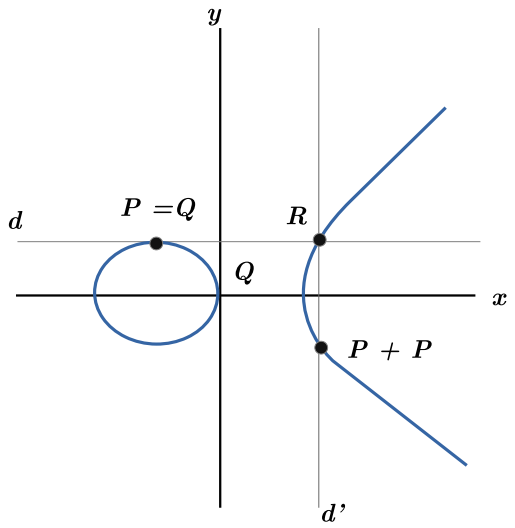


Figura 3.2: Gráfica de doblado de dos puntos en $\mathbb{R}$.

1. Trazar la recta tangente al punto, por este motivo no puede ser una recta singular.

2. Encontrar el segundo punto de intersección entre la tangente y la curva definida.

3. Hacer reflexión del punto respecto eje de las abscisas.

Dado $n \in \mathbb{Z}$ se tiene que, si n es positivo,

$$\begin{aligned} [n] : \mathbf{E} &\longrightarrow \mathbf{E} \\ P &\longmapsto [n]P = \underbrace{P + P + \ldots + P}_{n \text{ veces}} \end{aligned} \tag{3.4}$$

En criptografía, se necesita de grupos abelianos finitos. Sin embargo, las curvas elípticas sobre los reales no forman un grupo abeliano finito, por lo que no se pueden utilizar de esta manera. Es necesario entonces que el campo sobre el que se define la curva sea finito, para que el grupo abeliano generado por la curva elíptica sobre dicho campo sea también finito. Además, es muy común en criptografía el uso de curvas sobre campos primos finitos ($\mathbb{K} = \mathbb{Z}_p$ con p primo). Por seguridad computacional el primo p debe ser muy grande.

A partir de ahora se considera $\mathbb{K} = \mathbb{Z}_p$ con $p > 3$ y aunque gráficamente no se puede expresar la operación de grupo como se ha hecho antes, sí se le tiene expresada algebraicamente con las fórmulas obtenidas en 3.2 y 3.3, que para $\mathbb{K} = \mathbb{Z}_p$ pasarían a ser:

- Suma de puntos

$$x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2 \; mod\, p \qquad y_3 = -y_1 + \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x_3) \; mod\, p \tag{3.5}$$

- Doblado de un punto

$$x_2 = \left(\frac{3x_1^2 + a}{2y_1}\right)^2 - x_1 - x_2 \; mod\, p \qquad y_2 = -y_1 + \left(\frac{3x_1^2 + a}{2y_1}\right)(x_1 - x_3) \; mod\, p \tag{3.6}$$

3.2.3. Forma de Montgomery

Una curva elíptica **E** puede expresarse en diferentes sistemas, por lo que resulta de gran interés estudiar la rapidez de la operación de grupo (suma de puntos y doblar un punto) en diferentes sistemas de coordenadas y ecuaciones. Esta rapidez se mide según el número de operaciones elementales necesarias para calcular la suma de puntos y doblar un punto. Dentro de todos los tipos de sistemas que se encuentran, cabe destacar la mejora en cuanto a rapidez de cálculo que aporta la forma de Montgomery . Una curva elíptica **E** está en forma de Montgomery si la ecuación de la curva es del tipo:

$$By^2 = x^3 + Ax^2 + x. \tag{3.7}$$

3.2.4. Problema del Logaritmo Discreto

Definición 3.2.2 *Sea $(\mathbb{G}, \cdot)$ grupo cíclico finito y sea $|\mathbb{G}| = n$. Se toma un elemento primitivo $\alpha \in \mathbb{G}$ y otro elemento $\beta \in \mathbb{G}$, el problema del logaritmo discreto (PLD) consiste en encontrar el entero x, con $1 \leq x \leq n$ tal que*

$$\beta = \alpha \cdot \alpha \cdot \ldots \cdot \alpha = \alpha^x$$

Este entero x se llama logaritmo discreto de β en base α y se denota como $x = log_\alpha \beta$.

Calcular $\beta = \alpha^x$, es muy sencillo, pero invertir la aplicación resulta casi imposible debido a la aleatoriedad de la operación de grupo, ya que no se espera encontrar el x tal que $\beta = \alpha^x$ hasta haber comprobado todas las posibilidades. La dificultad varía según el grupo $\mathbb{G}$ seleccionado.

PLD en $\mathbb{Z}_p^*$.

Sea p primo y sea el grupo cíclico finito $\mathbb{Z}_p^*$ de orden $p-1$. Dado un elemento primitivo $\alpha \in \mathbb{Z}_p^*$ y otro elemento $\beta \in \mathbb{Z}_p^*$, el PLD consiste en determinar el entero $1 \leq x \leq p-1$ tal que $\alpha^x \equiv \beta \bmod p$. Si el primo p es suficientemente grande, calcular logaritmos discretos módulo p es muy complicado, mientras que la exponenciación $\alpha^x \equiv \beta \bmod p$ es muy sencilla.

3.3. Protocolo Diffie-Hellman

El Intercambio de llaves Diffie-Hellman (DHKE) fue propuesto por Whitfield Diffie y Martin Hellman en 1976 con la publicación del artículo "New directions in Cryptography" [12], influenciados por el trabajo anterior de Ralph Merkle, siendo el primer método asimétrico publicado en la literatura. Este método, sencillo pero muy eficaz, permite establecer una llave común entre dos partes que se comunican a través de un canal inseguro (de ahí que todos los parámetros que se envían sean públicos), aplicando el PLD. Con esa llave común establecida, las dos partes pueden cifrar y descifrar, es decir, esta llave funciona como llave simétrica. Originalmente fue propuesto para grupos multiplicativos de enteros módulo p, con p un número primo, el esquema se adaptó más tarde a grupos de puntos aditivos en curvas elípticas por Koblitz y Miller. Comúnmente es conocido como protocolo de curva elíptica Diffie-Hellman (ECDH), esta variante se describe de manera concisa en el Algoritmo 1.

Algoritmo 1 Protocolo de curva elíptica Diffie-Hellman

Parámetros públicos: Primo p, Curva $\mathbf{E}$, Punto $Q(x, y) \in \mathbf{E}$

Fase 1: Generación del par de llaves

Alicia	**Beto**
1: Genera llave privada $a = k_{priv,A} < q$.	1: Genera llave privada $b = k_{priv,B} < q$.
2: Genera lave pública $k_{pub,A} = [a]Q$.	2: Genera lave pública $k_{pub,B} = [b]Q$.

Fase 2: Calculo del secreto compartido

Alicia	**Beto**
3: Envía su llave pública $k_{pub,A}$.	3: Envía su llave pública $k_{pub,B}$.
4: Alicia tiene $(\mathbb{G}, Q, a, b[Q])$ y calcula $k_{AB} = [a]([b]Q)$.	4: Beto tiene $(\mathbb{G}, Q, b, [a]Q)$ y calcula $k_{BA} = [b]([a]Q)$.

3.3.1. DHKE con curvas elípticas

Definición 3.3.1 *Sea* $\mathbf{E}$ *una curva elíptica sobre* $\mathbb{Z}_p$*, dados dos puntos* $Q, P \in \mathbf{E}$ *con* $P \in \langle Q \rangle$ *y* $\langle Q \rangle$ *un subgrupo generado por* Q*, el PLD en* $\mathbf{E}$ *consiste en encontrar el entero positivo* x *tal que* $[x]Q = P$*.*

Para utilizar el grupo generado por una curva elíptica sobre $\mathbb{Z}_p$ para algún primo p como grupo para el criptosistema DHKE, es necesario que p sea extremadamente grande para que se trate de un método de cifrado seguro, hasta tal punto que se considera seguro si el grupo $\mathbb{G}$ generado contiene un subgrupo cíclico de orden al menos 2^{160}. Además, sólo es necesario encontrar un elemento de $\mathbf{E}$ que tenga orden

suficientemente grande, y no es necesario en ningún caso hallar todos los elementos de **E**.

El proceso general del protocolo DHKE con curva elíptica sería el siguiente:

1. Alicia escoge su llave privada $a = k_{priv,A} < q$, a partir de la cual genera su llave pública $k_{pub,A} = [a]Q$ y envía $k_{pub,A}$ a Beto.

2. Simultáneamente Beto escoge su llave privada $b = k_{priv,B} < q$, genera su llave pública $k_{pub,B} = [b]Q$ y envía $k_{pub,B}$ a Alicia.

3. Alicia dispone de $(\mathbb{G}, Q, a, b[Q])$ y calcula $k_{AB} = [a]([b]Q)$. 4. Beto dispone de $(\mathbb{G}, Q, b, [a]Q)$ y calcula $k_{BA} = [b]([a]Q)$.

Se prueba fácilmente que Alicia y Beto calculan el mismo secreto compartido.

$$k_{AB} = [a]k_{pub,B} = [a]([b]Q) = [a][b]Q$$
$$k_{BA} = [b]k_{pub,A} = [b]([a]Q) = [a][b]Q$$

3.3.2. Curva 25519

En criptografía, la curva 25519 es una curva elíptica que ofrece 256 bits de seguridad y está diseñada para su uso con el esquema de intercambio de llaves Diffie-Hellman de curva elíptica. Es una de las curvas elípticas más rápidas y no está cubierta por ninguna patente conocida.

La implementación de referencia de la curva 25519 es software de dominio público, además, es una de las curvas más utilizadas y eficientes, considerada muy segura y por ello se ha convertido en un estándar. Fue propuesta por Daniel J. Bernstein [13] en 2006 y se popularizó a partir del año 2013 por las sospechas sobre la National Security Agency (NSA) en su familia de curvas utilizadas hasta el momento. La figura 3.3 muestra una gráfica de la curva elíptica 25519 definida sobre $\mathbb{R}$. La curva 25519 pertenece a la familia llamada *curvas de Montgomery*, y esta definida por la ecuación 3.8 de tipo Weierstrass:

$$y^2 = x^3 + 486662x^2 + x \tag{3.8}$$

sobre el campo finito definido por el primo $2^{255} - 19$, y toma como punto base $x = 9$.

Función X25519

La función X25519 según el RFC 7748 [14] realiza multiplicaciones escalares en la forma de Montgomery de la curva 25519, se usa siempre que se esté implementando el algoritmo Diffie-Hellman. La función toma un escalar y una coordenada u como entradas y produce una coordenada u como salida. Aunque la función trabaja internamente con enteros, la entrada y la salida son cadenas de 32 bytes y esta especificación define su codificación.

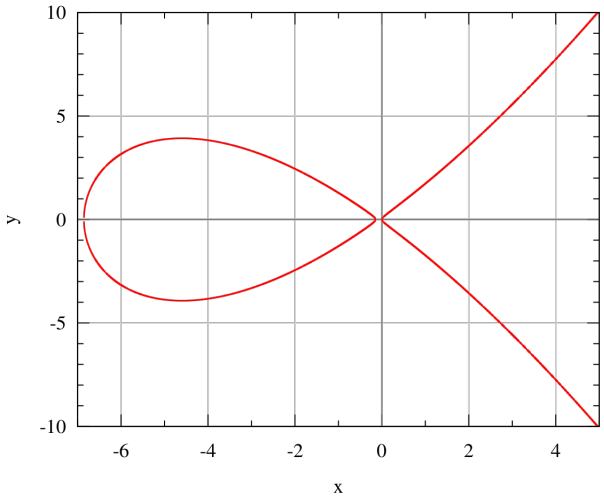


Figura 3.3: Gráfica de la curva 25519 en $\mathbb{R}$.

El algoritmo 2 describe las operaciones de bajo nivel que realiza la multiplicación escalar en forma de Montgomery en la curva 25519 para $n = 255$. La notación $\oplus$ representa el operador exclusivo o lógico, mientras que los símbolos $+, -, \times, ^2$ y $^{-1}$ representan el campo de operaciones aritméticas $\mathbb{F}_p$ de suma, resta, multiplicación, elevación al cuadrado e inversión, respectivamente.

En cada iteración i del algoritmo 2, la función de intercambio condicional (cswap) intercambia los valores de las coordenadas R_0 y R_1 cuando los bits k_i -1 y k_i son diferentes. Esta función es una contramedida para posibles ataques basados en caché, que podrían revelar los dígitos escalares (la llave privada en el algoritmo 1), determinando el orden de acceso de los puntos R_0 y R_1. La función cswap consta solo de operaciones lógicas simples, por lo que su costo es mínimo.

La coordenada u es un elemento del campo subyacente $GF(2^{255} - 19)$ y está codificado como una matriz de bytes, u, en orden *little-endian* tal que $u[0] + 256 * u[1] + 256^2 * u[2] + \ldots + 256^{(n-1)} * u[n-1]$ es congruente con el valor módulo p y $u[n-1]$ es mínimo. Al recibir una matriz de este tipo, las implementaciones de X25519 deben enmascarar el bit más significativo en el byte final. Esto se hace para preservar la compatibilidad con formatos de puntos que reservan el bit de signo para su uso en otros protocolos y para aumentar la resistencia a la implementación de huellas digitales.

Las implementaciones deben aceptar valores no canónicos y procesarlos como si se hubiera reducido el modulo del campo primo. Los valores no canónicos son $2^{255} - 19$ a $2^{255} - 1$. La función X25519 se utiliza en un protocolo ECDH como lo ilustra la figura 3.4.

Alicia genera 32 bytes aleatorios de $a[0]$ a $a[31]$ y transmite su llave pública $K_A = X25519(a, 9)$ a Beto, donde 9 es la coordenada u del punto base y se codifica como un byte con el valor 9, seguido de 31 bytes de cero. De manera similar, Beto genera 32 bytes aleatorios de $b[0]$ a $b[31]$, calcula su llave pública $K_B = X25519(b, 9)$ y la

Algoritmo 2 Descripción de la multiplicación escalar en forma Montgomery con la curva 25519

Entrada: $P(u_P, v_P) \in \mathbf{E}, k = (k_{n-1} = 1, k_{n-2}, \ldots, k_1, k_0)_2, a24 = (A+2)/4$
Salida: $u_Q = k_P$
Inicialización: $U_{R_0} \leftarrow 1, Z_{R_0} \leftarrow 0, U_{R_1} \leftarrow u_{P_1} \leftarrow u_P, Z_{R_1} \leftarrow 1, s \leftarrow 0$

1: **for** $i \leftarrow n-1 \ldots 0$ **do**
2: $\quad s \leftarrow s \oplus k_i$
3: $\quad U_{R_0}, U_{R_1} \leftarrow cswap(s, U_{R_0}, U_{R_1})$
4: $\quad Z_{R_0}, Z_{R_1} \leftarrow cswap(s, Z_{R_0}, Z_{R_1})$
5: $\quad s \leftarrow k_1$
6: $\quad A \leftarrow U_{R_0} + Z_{R_0}; B \leftarrow U_{R_0} - Z_{R_0}$ ▷ Operaciones comunes
7: $\quad C \leftarrow U_{R_1} + Z_{R_1}; D \leftarrow U_{R_1} - Z_{R_1}$ ▷ Suma
8: $\quad C \leftarrow C \times B; D \leftarrow D \times A$
9: $\quad U_{R_1} \leftarrow D + C; U_{R_1} \leftarrow U_{R_1}^2$
10: $\quad Z_{R_1} \leftarrow D - C; Z_{R_1} \leftarrow Z_{R1}^2; Z_{R_1} \leftarrow u_P \times Z_{R_1}$
11: $\quad A \leftarrow A^2; B \leftarrow B^2$ ▷ Doblado
12: $\quad U_{R_0} \leftarrow A \times B$
13: $\quad A \leftarrow A - B$
14: $\quad Z_{R_0} \leftarrow a_{24} \times A; Z_{R_0} \leftarrow +B; Z_{R_0} \leftarrow Z_{R_0} \times A$
15: **end for**
16: $U_{R_0}, U_{R_1} \leftarrow cswap(s, U_{R_0}, U_{R_1})$
17: $Z_{R_0}, Z_{R_1} \leftarrow cswap(s, Z_{R_0}, Z_{R_1})$
18: $Z_{R_0} \leftarrow Z_{R_0}^{-1}$
19: $u_{R_0} \leftarrow U_{R_0} \times Z_{R_0}$
20: **return** $u_Q \leftarrow u_{R_0}$

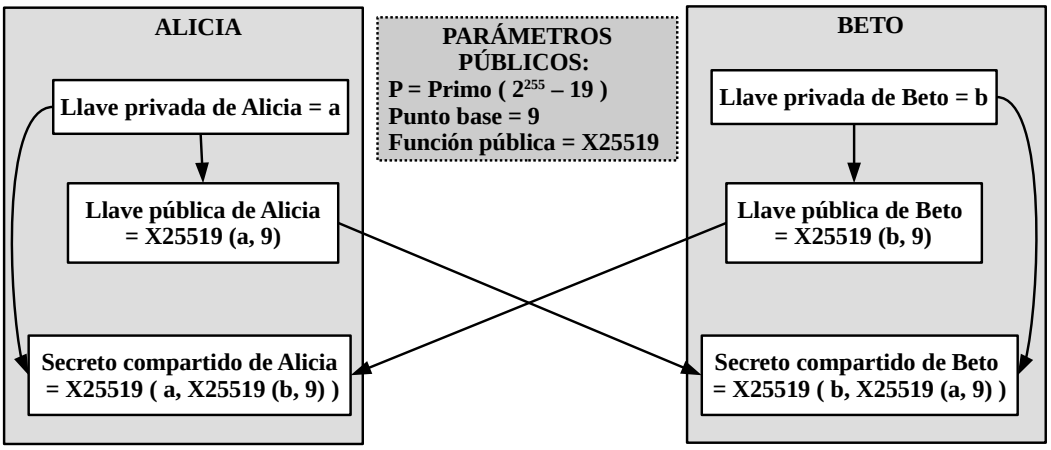


Figura 3.4: Diagrama DHKE con curva elíptica 25519.

transmite a Alicia. Al usar los valores generados y la entrada recibida, Alicia calcula el secreto compartido $X25519(a, K_B)$ y por lo tanto Beto calcula $X25519(b, K_A)$. Ahora ambos comparten el secreto $K = X25519(a, X25519(b, 9))$ que es igual a $X25519(b, X25519(a, 9))$. Al finalizar ambos pueden verificar el secreto, sin filtrar información adicional sobre el valor de K.

Diversos protocolos como el utilizado por la aplicación de mensajería instantánea WhatsApp [15] utilizan la curva elíptica 25519 y la función X25519, debido a sus características de seguridad y eficiencia.

3.4. Identificadores efímeros

La generación de identificadores efímeros es un mecanismo de seguridad propuesto para la utilización de balizas BBE, para realizar este proceso de generación se optó por usar el algoritmo de cifrado autenticado Ascon conocido por ser muy ligero y sencillo, por lo cual se ha distinguido en competencias internacionales[1] de cifrado autenticados y *hashing*.

3.5. Algoritmo Ascon

Ascon es una familia de algoritmos de cifrado autenticado y generación de *hash* que emplean la misma función de permutación, fueron diseñados para ser ligeros y fáciles de implementar, incluso con contramedidas adicionales contra ataques de canal lateral. Ascon utiliza una función esponja en configuración dúplex para el cifrado autenticado[2] Ascon ha sido seleccionada como la opción principal para el cifrado autenticado ligero en la competencia CAESAR (2014–2019) y actualmente participa en la competencia NIST *Lightweight Cryptography* que comenzó en el año 2019 para mas detalles en [16].

Ascon fue diseñado en el año por un equipo de criptógrafos de la Universidad de Tecnología de Graz, Infineon Technologies y la Universidad de Radboud, ellos son Christoph Dobraunig, Maria Eichlseder, Florian Mendel y Martin Schläffer.

3.5.1. Modo de operación Ascon

Ascon utiliza un modo de operación basado en una función esponja en configuración dúplex para el cifrado autenticado. La longitud recomendada de llave, etiqueta y *nonce* es de 128 bits. La esponja opera en un estado de 320 bits, con bloques de mensajes de 64 o 128 bits. Donde la tasa r y la capacidad $c = 320 - r$ dependen de la variante de Ascon. El proceso de cifrado se divide en cuatro fases:

1. Inicialización: Inicializa el estado con la llave K y *nonce* N.

[1] Ganador en la competencia CAESAR y es finalista en NIST

[2] El valor recomendado para las longitudes de llave, etiqueta y *nonce* es de 128 bits.

2. Procesamiento de datos asociados : Actualiza el estado con los bloques de datos asociados A_i.

3. Procesamiento de texto sin formato: Inyecta bloques de texto sin formato P_i en el estado y extrae bloques de texto cifrado C_i.

4. Finalización: Vuelve a inyectar la llave K y extrae la etiqueta T para la autenticación.

Inicialización

El estado inicial de 320 bits de Ascon está formado por la llave secreta K de k bits y *nonce* N de 128 bits, así como un IV que especifica el algoritmo (incluido el tamaño de la llave k, la tasa r, el número de rondas de inicialización y finalización a, y el número de rondas intermedias b, cada uno escrito como un entero de 8 bits. En la inicialización, se aplican a rondas de la transformación de ronda p al estado inicial, seguido de un xor de la llave secreta K:

$$S \leftarrow p^a(S) \oplus (0^{320-k}||K)$$

Procesamiento de datos asociados

Ascon procesa los datos asociados A en bloques de r bits. Agrega un solo 1 y el menor número de ceros a A para obtener un múltiplo de r bits y dividirlo en s bloques de r bits, $A_1||\ldots||A_s$. En caso de que A esté vacío, no se aplica ningún relleno y $s = 0$. A cada bloque A_i con $i = 1, \ldots, s$ se le aplica xor con los primeros r bits S_r del estado S, seguido de una aplicación de la permutación b-round p^b a S:

$$S \leftarrow p^b((S_r \oplus A_i)||S_c), 1 \leq i \leq s$$

Después de procesar A_s (también si $s = 0$), después a una constante de separación de dominio de 1 bit se le aplica xor con S:

$$S \leftarrow S \oplus (0^{319}||1)$$

Procesamiento del mensaje

Ascon procesa el mensaje P en bloques de r bits. El proceso de relleno agrega un solo 1 y el menor número de ceros al texto plano P, de modo que la longitud del texto plano rellenado es un múltiplo de r bits. El texto sin formato relleno resultante se divide en t bloques de r bits, $P_1||\ldots||P_t$:

$$P_1, \ldots, P_t \leftarrow r - bit \text{ bloques de } P||1||0^{r-1-(|P| mod r)}$$

En cada iteración, un bloque de texto plano relleno P_i con $i = 1, \ldots, t$ se aplica a los primeros r bits S_r del estado interno S, seguido de la extracción de un bloque

de texto cifrado C_i. Para cada bloque, excepto el último, todo el estado interno S se transforma mediante la permutación p^b utilizando b rondas:

$$Ci \leftarrow S_r \oplus P_i$$

El último bloque de texto cifrado C_t se trunca a la longitud del último fragmento de bloque del texto plano sin relleno para que su longitud esté entre 0 y $r-1$ bits, y la longitud total del texto cifrado C sea exactamente la misma que para el texto plano original P:

$$\widetilde{C}_t \leftarrow \lfloor Ct \rfloor_{|P| \bmod r}$$

Finalización

En la finalización, se le aplica un xor a la llave secreta K con el estado interno, el cual se transforma mediante la permutación p^a utilizando a rondas. La etiqueta T consta de los últimos 128 bits (menos significativos) del estado aplicando la xor con los últimos 128 bits de la llave K:

$$S \leftarrow p^a(S \oplus (0^r||K||0^{c-k}))$$
$$T \leftarrow \lceil S \rceil^{128} \oplus \lceil K \rceil^{128}$$

El algoritmo de cifrado Ascon devuelve la etiqueta T junto con el texto cifrado $C_1 || \ldots ||\widetilde{C}_t$. El algoritmo de descifrado devuelve el texto plano $P_1|| \ldots ||\widetilde{P}_t$ solo si el valor de etiqueta calculado coincide con el valor de etiqueta recibido. El modo de operación dúplex del cifrado autenticado Ascon completo se ilustra en la figura 3.5.

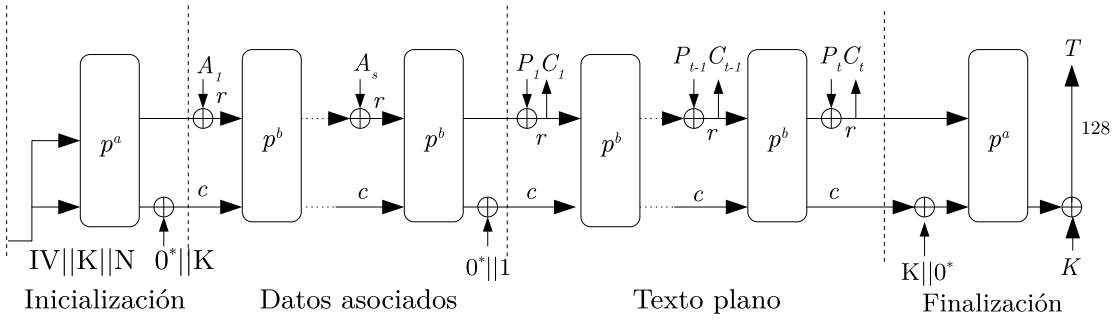


Figura 3.5: Diagrama de cifrado autenticado Ascon.

3.5.2. Permutación Ascon

Todos los miembros de la familia Ascon operan en un estado de 320 bits y utilizan la misma permutación ligera. La permutación aplica iterativamente una ronda de

transformación basada en SPN[3] donde $a = 12$ veces (para p^a) y $b \in \{6, 8\}$ veces (para p^b). La ronda de transformación consta de los siguientes tres pasos que operan en un estado de 320 bits dividido en cinco palabras x_0, x_1, x_2, x_3, x_4 de 64 bits cada una, como lo muestra la figura 3.6:

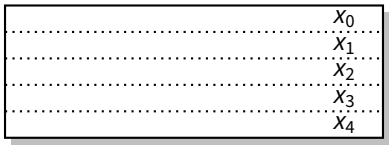


Figura 3.6: División del estado S.

- Suma de constantes de ronda: El paso p_C suma una constante de ronda c_r para registrar la palabra x_2 del estado S en la ronda i (ver figura 3.7). Ambos índices r e i parten de cero y se usa $r = i$ para p^a y $r = i + a - b$ para p^b.

$$x_2 \leftarrow x_2 \oplus c_r.$$

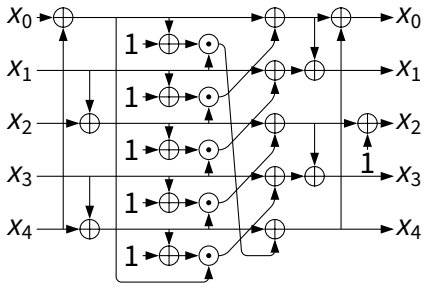


Figura 3.7: $S(x)$ (caja S de 5 bits) de Ascon.

- Capa de sustitución no lineal: La capa de sustitución p_S actualiza el estado S con 64 aplicaciones paralelas de la caja S de 5 bits $S(x)$ (definida en la figura 3.7) para cada segmento de bits de los cinco registros $x0 \ldots x4$ (ver la Tabla 3.1).

  Por lo general, se implementa en esta forma de división de bits, con operaciones realizadas en las palabras completas de 64 bits. La Tabla de búsqueda de S se muestra en la Tabla 3.2, donde x_0 es el MSB y x_4 el LSB.

[3] substitution–permutation network

$$
\begin{aligned}
x_0 &\leftarrow \Sigma_0(x_0) = x_0 \oplus (x_0 \gg 19) \oplus (x_0 \gg 28) \\
x_1 &\leftarrow \Sigma_1(x_1) = x_1 \oplus (x_1 \gg 61) \oplus (x_1 \gg 39) \\
x_2 &\leftarrow \Sigma_2(x_2) = x_2 \oplus (x_2 \gg 1) \oplus (x_2 \gg 6) \\
x_3 &\leftarrow \Sigma_3(x_3) = x_3 \oplus (x_3 \gg 10) \oplus (x_3 \gg 17) \\
x_4 &\leftarrow \Sigma_4(x_4) = x_4 \oplus (x_4 \gg 7) \oplus (x_4 \gg 41)
\end{aligned}
$$

Tabla 3.1: Capa lineal de Ascon con funciones de 64 bits $\Sigma i(x_i)$.

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	10	11	12	13	14	15	16	17	18	19	1a	1b	1c	1d	1e	1f
$S(x)$	4	b	1f	14	1a	15	9	2	1b	5	8	12	1d	3	6	1c	1e	13	7	e	0	d	11	18	10	c	1	19	16	a	f	17

Tabla 3.2: Caja S de 5 bits de Ascon como tabla de búsqueda.

- Capa de difusión lineal: La capa de difusión lineal p_L proporciona una difusión dentro de cada palabra de registro de 64 bits x_i. Aplica una función lineal $\Sigma_i(x_i)$ (definida en la Tabla 3.1) a cada palabra x_i.

3.5.3. Modo *hash* de Ascon

La familia Ascon incluye las funciones *hash*, Ascon-Hash y Ascon-Hasha , así como las funciones de salida extensibles Ascon-Xof y Ascon-Xofa, con modos de operación basados en funciones esponja. Ambos proporcionan seguridad de 128 bits con un tamaño de *hash* de al menos 256 bits. Los modos *hash* utilizan la misma permutación ligera de 320 bits que los modos de cifrado autenticado como se muestra en la figura 3.8.

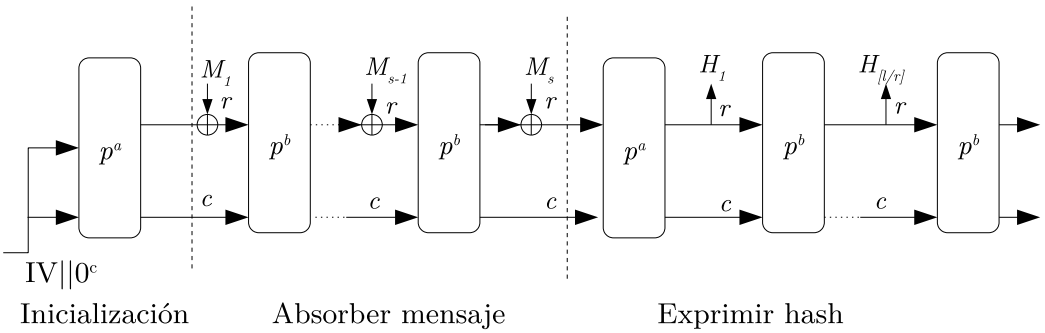


Figura 3.8: Diagrama de modo *hash* de Ascon.

Los modos *hash* absorben el mensaje M en bloques M_i de 64 bits y finalmente comprimen el valor *hash* H en bloques H_i de 64 bits. Después de que cada bloque es absorbido o exprimido, a excepción del último, se aplica la permutación p^b al estado. La permutación completa p^a se aplica en la inicialización y finalización, después del último bloque del mensaje.

Inicialización

El estado inicial de 320 bits de Ascon-Hash está definido por una constante IV que especifica los parámetros del algoritmo en un formato similar al de cifrado autenticado Ascon (incluyendo $k = 0$, la tasa r, y los números de rondas a y $b = 0$, cada uno escrito como un entero de 8 bits), seguido de la longitud máxima de salida de h bits como un entero de 32 bits (con $h = l = 256$) y un valor cero de 256 bits. La ronda de permutación p^a se aplica para inicializar el estado S:

$$S \leftarrow p^a(IV_{h,r,a}||0^{256})$$

El estado inicial S de 320 bits se puede calcular previamente para cada instancia obteniendo lo que se muestra en la Tabla 3.3.

	ee9398aadb67f03d
	8bb21831c60f1002
$S \leftarrow$	b48a92db98d5da62
	43189921b8f8e3e8
	348fa5c9d525e140

Tabla 3.3: Precálculo de instancias.

Absorción de mensaje

Ascon-Hash procesa el mensaje M en bloques de r bits. El proceso de relleno es el mismo que para el texto sin formato de Ascon: agrega un solo 1 y el menor número de ceros a M, de modo que la longitud del mensaje rellenado es un múltiplo de r bits. El mensaje resultante se divide en s bloques de r bits, $M_1||\ldots||M_s$:

$$M_1, \ldots, M_s \leftarrow \text{r-bit bloques de } M||1||0^{r-1-(|M| mod r)}$$

Los bloques del mensaje M_i con $i = 1, \ldots, s$ se procesan de la siguiente manera. Cada bloque M_i se asocia a los primeros r bits S_r del estado S, seguido de una aplicación de la ronda de permutación p^a a S:

$$S \leftarrow p^a((S_r \oplus M_i)||S_c), 1 \leq i \leq s$$

Exprimido (*Squeezing*)

La salida del *hash* se extrae del estado en bloques de H_i hasta que la longitud de salida solicitada $l \leq h$ se completa después de $t = \lceil l/r \rceil$ bloques. Después de cada extracción, el estado interno S se transforma mediante la ronda de permutación p^a:

$$\begin{aligned} &H_i \leftarrow S_r \\ &S \leftarrow p^a(S), \qquad 1 \leq i \leq t = \lceil l/r \rceil \end{aligned}$$

El último bloque de salida H_t se trunca a $l \bmod r$ bits y $H = H_1||\ldots||\widetilde{H_t}$ devuelve:

$$\widetilde{H_t} \leftarrow \lfloor H_t \rfloor_{l \bmod r}$$

Capítulo 4

Emulación de balizas BBE

El término emulación se refiere a la capacidad de reproducir el comportamiento de una determinada plataforma de hardware y software, es adecuada para usar en objetos digitales dinámicos e interactivos.

En este capítulo se explica en detalle la forma de realizar emulaciones de balizas tipo iBeacon y Eddystone, con las cuales se pueden realizar ataques de suplantación de balizas BBE. Un ataque de suplantación es un término que describe el tipo de comportamiento mediante el cual un adversario se hace pasar por un usuario o dispositivo de confianza, en este caso, se haría pasar por una baliza BBE de alguna aplicación, generalmente para realizar algo que le sea de beneficio y en ocasiones perjudicial para los propietarios de las balizas y usuarios de la aplicación.

4.1. Descripción de baliza BBE

Para el análisis de la estructura y comportamiento general de las balizas BBE, se utilizaron las balizas iBKS 105 de la marca Accent. Oficialmente fue el primer modelo de baliza compatible con los protocolos iBeacon y Eddystone al mismo tiempo. iBKS 105 es una baliza BBE basada en un chipset semiconductor modelo Nordic nrf51822, utiliza el protocolo de radio BBE y una pila de botón tipo CR2477 con un tiempo de vida de 30 a 46 meses(dependiendo de la potencia Tx a intervalo de 1 segundo). El circuito impreso de la baliza está preparado para implementar diferentes sensores que pueden ser ensamblados para pedidos grandes. La Figura 4.1 ilustra el exterior de una baliza iBKS 105 de la marca Accent, la cual cuenta con un chasis de plástico redondo que cubre el microcontrolador y su batería, además, puede abrirse fácilmente sin necesidad de herramientas y anclarse sobre cualquier superficie gracias a su cinta de doble cara.

Las especificaciones técnicas de la baliza iBKS 105 se listan a continuación:

- Tamaño: ancho de 11.3 mm, diámetro de 52.6 mm
- Peso: 24 gramos

Figura 4.1: Baliza iBKS 105.

- Distancia de detección: 50 metros máximo
- Batería: CR2477, 3 V - 1000 mAh
- Consumo de corriente (inactiva): 2.4 μA
- Temperatura de operación: -25 a +60°C

La empresa *Accent Advanced Systems* desarrolló la aplicación móvil (para iOS y Android) *iBKS Config Tool*, herramienta con la que se configuran todas sus balizas BBE, el manual de usuario de la aplicación se encuentra disponible en el sitio oficial de Accent [17].

Generalmente la arquitectura de este tipo de dispositivos comerciales es restringida por sus fabricantes, razón por la cual las modificaciones necesarias para implementar los mecanismos de seguridad que se proponen en esta tesis, resultan irrealizables. Por este motivo se decidió emplear un dispositivo como la Raspberry Pi; ya que al ser una plataforma abierta para ser utilizada en diversas aplicaciones, es posible usarla para emular una baliza BBE y configurarla libremente. Los recursos de hardware para realizar la emulación de balizas BBE son los siguientes:

- Raspberry Pi 4
- Tarjeta micro SD de 16 GB o 32 GB
- Fuente de alimentación
- Balizas BBE
- Computadora
- Teléfono inteligente

A continuación se muestra el proceso de preparación para emular una baliza BBE a través de una Raspberry Pi.

4.2. Preparación de Raspberry Pi

El primer paso para utilizar la Raspberry Pi es la instalación del sistema operativo Raspberry Pi OS, esto se hace mediante una unidad externa de almacenamiento disponible en la placa con una tarjeta micro SD, el principal beneficio que ofrece su instalación es que tiene precargados los módulos necesarios para utilizar los periféricos con los que cuenta la tarjeta, como el módulo de Bluetooth. Los detalles de instalación y configuración del sistema Raspberry Pi OS están disponibles en el Apéndice A, en la página 81.

Con el sistema Raspberry Pi OS funcionando correctamente, otro aspecto importante es el acceso a la Raspberry Pi, esto se puede realizar a través de diferentes maneras, usando periféricos de E/S como monitor, teclado y ratón o utilizando la conexión de red vía Ethernet o Wifi. Como parte del sistema operativo Raspberry Pi OS se instalaron los módulos para hacer uso de distintas conexiones, en este caso se elige por practicidad utilizar el Wifi, lo que permite asignar un dirección IP y utilizar el servicio SSH para el acceso remoto a la Raspberry Pi por medio de un canal seguro.

La conexión con SSH requiere del nombre de usuario (el usuario por defecto es `pi`) y de la dirección IP de la Raspberry Pi, con estos datos se puede conectar la plataforma a una red, ya sea de manera alámbrica o inalámbrica (ver Apéndice B, en la página 87, para más detalles). Un ejemplo de comando para realizar la conexión SSH es el siguiente:

```
ssh pi@192.168.1.79
```

Se ingresa la contraseña predeterminada (la contraseña por defecto es `raspberry`), en la primera solicitud de conexión, el protocolo preguntará si se desea continuar con la conexión SSH. Por medio de este servicio se podrá trabajar directamente en la terminal de la Raspberry Pi a través de la computadora portátil sin necesidad de otros periféricos. En la sección siguiente se dan detalles sobre la configuración de la Raspberry Pi como un periférico BBE.

4.3. Configuración de Raspberry Pi como periférico BBE

La Raspberry Pi cuenta con un módulo Bluetooth integrado y controlado por Raspberry Pi OS, este puede funcionar como Bluetooth de baja energía configurándolo a través de comandos o programas especializados.

Existen diversas formas de realizar emulaciones de periféricos BBE, ingresando comandos Bluetooth directamente en la terminal o con programas en lenguajes como Python, C y JavaScript. En particular se explicará una configuración programada en JavaScript usando el entorno Nodejs y su módulo de emulación Bleno. Debido al amplio soporte de la plataforma, la facilidad de instalación y la menor curva de aprendizaje con JavaScript, el módulo Bleno presenta un caso sólido como la mejor opción

para imitar a los periféricos BBE. La Figura 4.2 ilustra la relación de contención del software utilizado para la emulación de periféricos BBE.

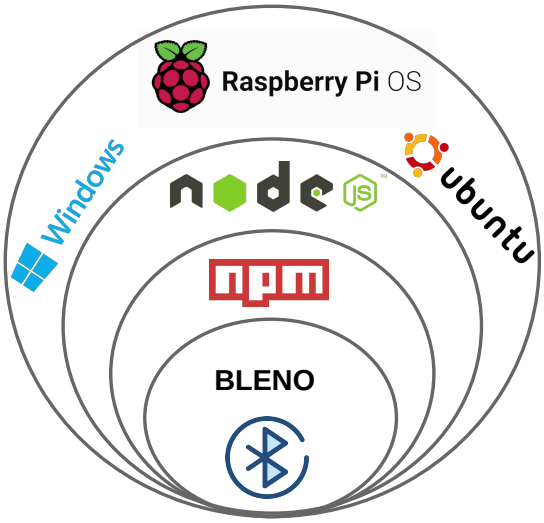


Figura 4.2: Capas de software.

El primer paso es instalar el entorno de ejecución Nodejs y su administrador de archivos `npm`, después se utiliza `npm` para instalar el módulo Bleno, el cual es un software especializado para emular periféricos BBE programado en JavaScript. A continuación se dan más detalles acerca de la instalación y configuración de estos programas.

4.3.1. Nodejs

Nodejs es un entorno de ejecución multiplataforma para la capa del servidor basado en el lenguaje JavaScript. Recordando que JavaScript era un lenguaje exclusivo del lado del cliente y solo se podía ejecutar con un navegador web, esto cambió en el año 2009 con la creación de Nodejs y la implementación del motor V8 de código abierto para JavaScript desarrollado por Google. Nodejs incorpora en su núcleo el motor V8 para ejecutar código JavaScript sin necesidad de algún navegador. Otra de las ventajas de Nodejs es que usa un modelo de operaciones asíncrono y orientado a eventos, es decir, puede ejecutar tareas de manera más eficiente, además Nodejs es un entorno especializado para aplicaciones en tiempo real.

Un sistema operativo basado en Linux contiene una versión de Nodejs en sus repositorios predeterminados que puede utilizarse para proporcionar una experiencia uniforme en varios sistemas. En el momento en que se redacta este trabajo, la versión en los repositorios es la 10.19. En caso de no contar con Nodejs, las opciones de descargas para distintos sistemas operativos se encuentran en el sitio oficial:

```
https://nodejs.org/es/download/current/
```

A continuación se describe la instalación de Nodejs en un sistema Linux a través del administrador de archivos `apt`. Como primer paso se actualiza el índice de paquetes locales mediante el comando:

```
sudo apt update
```

Al terminar la actualización se instala Nodejs con el comando siguiente:

```
sudo apt install nodejs
```

Para comprobar que la instalación se haya realizado de forma correcta, se consulta el número de versión de Nodejs como se muestra a continuación.

```
nodejs -v
Output
v10.19.0
```

Si el paquete de los repositorios se ajusta a los requerimientos, es todo lo que se necesita para la configuración de Nodejs. Nodejs cuenta con un gestor de paquetes llamado npm (Node Package Manager), este permite gestionar las dependencias de un proyecto, distribuir código e instalar diversos módulos y paquetes para utilizarlos con el entorno Nodejs. Se procede a instalar `npm`, con ayuda del administrador de archivos `apt` de la manera siguiente:

```
sudo apt install npm
```

Con los pasos anteriores se debe de tener instalado correctamente Nodejs y el administrador `npm`, lo cual se hizo con ayuda de los repositorios de software predeterminados de Linux.

Para usar de manera más óptima los módulos Nodejs para la emulación de dispositivos BBE, se opta por utilizar una versión más estable de Nodejs, se puede cambiar la versión actual a la versión anterior 8.9.0. Con este fin se instala el administrador de versiones de Nodejs con el comando siguiente:

```
sudo npm install -g n
```

A continuación se cambia a la versión 8.9.0 de Nodejs mediante el comando que sigue:

```
sudo n 8.9.0
```

Posteriormente se instala la herramienta `bluetooth-hci-socket` , utilizada para la gestión de la conexión Bluetooth con Nodejs, se usa el comando de instalación de `npm` como se muestra a continuación:

```
sudo npm install bluetooth-hci-socket --unsafe-perm
```

La preparación anterior es necesaria para la correcta instalación de Bleno, el módulo de emulación de periféricos BBE desarrollado en JavaScript.

4.3.2. Bleno

Bleno es un módulo de Nodejs, puede ejecutarse en sistemas MacOS, Windows o Linux. Bleno permite escribir lógica adicional en el comportamiento de un periférico virtual BBE, como computar datos para enviar a lectura y notificar o reaccionar a eventos de escritura. Se puede agregar fácilmente el código JavaScript a un sistema de control de versiones, como Git, para que todo un equipo pueda ejecutar fácilmente su propia instancia del periférico virtual.

A continuación se procede a instalar el módulo Bleno, este emula un dispositivo BBE y crea una interfaz con el controlador BBE de Raspberry Pi OS.

```
sudo npm install bleno
```

Finalmente se instala el paquete `onoff`, el cual permite la interfaz con el GPIO[1] de la Raspberry Pi.

```
sudo npm install onoff
```

Para la configuración básica del "dispositivo BBE" se crea un archivo JavaScript donde se establece el servicio principal, como se muestra a continuación:

```
const bleno = require('bleno');
bleno.on('stateChange', function(state) {
    console.log('on stateChange: ' + state);
    if (state === 'poweredOn') {
      bleno.startAdvertising('RaspberryPi', ['1803']);
    } else {
      bleno.stopAdvertising();
    }
});
```

El archivo se guarda con la extensión `.js`, por ejemplo, `bbe.js`, después se ejecuta en el entorno Nodejs con el comando siguiente:

```
sudo node bbe.js
```

[1] General Purpose Input/Output

Posteriormente se realiza un escaneo de dispositivos BBE para comprobar la configuración anterior, se debe visualizar el nombre dado al dispositivo (*RasberryPi*) y su dirección MAC como se muestra en el ejemplo siguiente:

```
sudo hcitool lescan
LE Scan ...
DC:A6:32:A4:8B:E4 (unknown)
DC:A6:32:A4:8B:E4 RaspberryPi
```

4.4. Emulación de baliza BBE

Existe una serie de bibliotecas y herramientas de código abierto que se pueden utilizar para enviar datos de tipo baliza BBE desde una Raspberry Pi. Se pueden utilizar programas desarrollados en lenguaje C o Python utilizando sus respectivas bibliotecas de bluetooth, otra forma es a través de comandos del sistema BlueZ ingresados directamente en la terminal de la Raspberry Pi. A continuación se describe el proceso de emulación de balizas BBE con comandos de BlueZ.

Una de las herramientas principales para la configuración y programación Bluetooth y BBE es la pila de protocolos BlueZ. A continuación se muestra la forma de instalación de las bibliotecas necesarias para el funcionamiento del sistema BlueZ.

```
sudo apt-get install libusb-dev
sudo apt-get install libdbus-1-dev
sudo apt-get install libglib2.0-dev --fix-missing
sudo apt-get install libudev-dev
sudo apt-get install libical-dev
sudo apt-get install libreadline-dev
```

Luego se puede realizar la instalación del sistema BlueZ, el cual proporciona el soporte para las capas y protocolos centrales de Bluetooth y BBE.

4.4.1. BlueZ

BlueZ es el sistema Bluetooth programado en Linux, en este caso permite que una Raspberry Pi se comunique con dispositivos Bluetooth clásicos y Bluetooth de baja energía. Sus características principales son su flexibilidad, eficiencia e implementación modular. Para utilizar el BlueZ se debe de descargar, compilar y luego instalar otras bibliotecas complementarias. El manual de instalación del sistema BlueZ está disponible en el Apéndice C, en la página 89.

Después de tener instalado BlueZ, ya es posible transmitir una señal BBE, la configuración depende del tipo de baliza que se quiera emular, recordando que existen balizas que son compatibles con distintos protocolos como iBeacon, Eddystone y Altbeacon.

A continuación se explica como emular balizas BBE de distintos protocolos a través de una Raspberry Pi. Se pretende que la baliza BBE emulada se comporte de la siguiente manera:

- Inicialize y transmita una señal BBE.
- Emita paquetes de anuncios.
- Emita un identificador único y permita tener más de una baliza identificable.

4.4.2. Emulación de baliza iBeacon

En particular con las balizas tipo iBeacon, se sabe que su transmisión tiene tres identificadores importantes:

- proximityUUID: un UUID único que distingue sus balizas iBeacons de otras iBeacons.
- major: se utiliza para agrupar conjuntos de iBeacons relacionadas.
- minor: se utiliza para identificar una iBeacon dentro de un grupo.

La emisión del paquete de datos de la baliza iBeacon la conforman estos identificadores, originando que la transmisión sea única y que permita tener más de una baliza iBeacon identificable. Para lograr emular una baliza iBeacon, se realiza el proceso de configuración siguiente. Primero se debe levantar el servicio BBE, para hacerlo se ejecutan en la terminal de la Raspberry Pi los comandos en el orden que se muestra a continuación:

```
sudo hciconfig hci0 up
sudo hciconfig hci0 leadv 3
sudo hciconfig hci0 noscan
```

Se ingresa el comando `hciconfig` para verificar que se ejecutan correctamente las instrucciones anteriores, la terminal debe mostrar una configuración parecida a la que se muestra en el ejemplo siguiente:

```
hci0: Type: Primary  Bus: UART
BD Address: DC:A6:32:A4:8B:E4  ACL MTU: 1021:8  SCO MTU: 64:1
UP RUNNING
RX bytes:1590 acl:0 sco:0 events:100 errors:0
TX bytes:3071 acl:0 sco:0 commands:100 errors:0
```

En la configuración se observan las características del módulo Bluetooth `hci0`, como la dirección (`BD Address`), MTU[2], estado actual (`UP RUNNING`), entre otras.

[2]Unidad de transmisión máxima

Después se utiliza el comando `hcitool` para configurar la transmisión del UUID. Para este ejemplo, se establecen los valores de major y minor como `0` adjuntándolos como `00 00 00 00` al final del UUID. Finalmente, el último byte que se agrega es el valor RSSI (*Received Signal Strength Indicator*), que es `C8`. El comando completo tiene el formato siguiente:

```
sudo hcitool -i hci0 cmd 0x08 0x0008 1E 02 01 1A 1A FF 4C 00 02
15 63 6F 3F 8F 64 91 4B EE 95 F7 D8 CC 64 A8 63 B5 00 00 00 00 C8
```

Los bytes de la trama en este ejemplo de configuración de baliza iBeacon se describe en la Tabla 4.1.

Valor de byte	**Descripción**
`0x08`	Define el OGF como el grupo de comandos de Bluetooth
`0x0008`	Define el OCF para la configuración de los datos de anuncio
`1E`	Define la longitud de datos de todo el paquete (30 bytes)
`02`	Número de bytes en la primera estructura AD
`01`	Bandera tipo AD
`1A`	valor 0x1A = 1010 bit 0 (APAGADO) LE (*Low Energy*) Modo limitado bit 1 (ON) LE Modo detectable general bit 2 (OFF) BR / EDR No admitido bit 3 (ON) LE y BR / EDR simultáneo (controlador) bit 4 (ON) LE y BR / EDR simultáneo (host)
`1A`	Número de bytes en la segunda estructura AD
`FF`	Datos específicos del fabricante (Tipo de AD)
`4C00`	Código de identificación de la empresa (0x004C = Apple)
`02`	Byte 0 del indicador de anuncio iBeacon
`15`	Byte 1 del indicador de anuncio iBeacon

`636F3F8F6491`
`4BEE95F7D8`
`CC64A863B5`
	UUID específico de iBeacon

`0000`	Major
`0000`	Minor
`C800`	Potencia Tx calibrada

Tabla 4.1: Descripción de los bytes de la trama de iBeacon.

Definiendo las siglas utilizadas en la Tabla 4.1, el OGF (*OpCode Group Field*) es el campo del grupo de código abierto en este caso del Bluetooth, el OCF (*Open Connectivity Foundation*) es la Fundación de conectividad abierta para configuración de anuncios, AD son las estructuras de los protocolos de anuncio como iBeacon y

Eddystone, por último BR / EDR (*Basic Rate/Enhanced Data Rate*) hace referencia a características del Bluetooth clásico.

Prueba de funcionamiento de baliza iBeacon

La manera más sencilla de comprobar la emulación de balizas BBE, es a través de una aplicación móvil con escáner BBE, en este caso se optó por utilizar la aplicación *iBKS Config Tool*, la cual es la aplicación de configuración de balizas de la marca *Accent*, se encuentra disponible en la tienda de aplicaciones y es compatible con los sistemas operativos Android y MacOS.

En la Figura 4.3 se observa el panel principal al ejecutar la aplicación iBKS Config Tool, se muestra la lista de dispositivos BBE encontrados con el escaneo y dentro de esta se visualiza a la Raspberry Pi configurada como una baliza con el protocolo iBeacon, se observa el nombre, identificador único, dirección Bluetooth y potencia de transmisión.

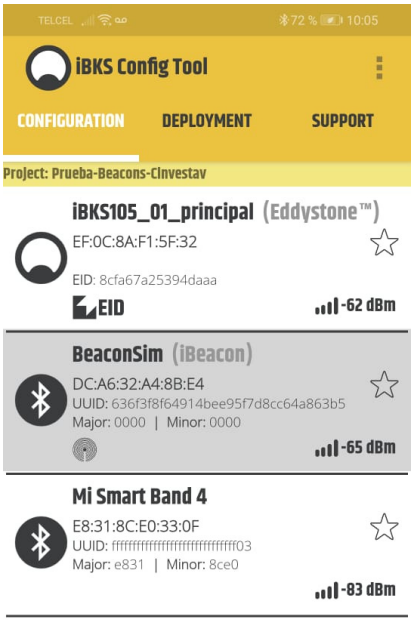


Figura 4.3: Escaneo con iBKS Config Tool.

4.4.3. Emulación de baliza Eddystone

De la misma forma que con el protocolo iBeacon, se puede configurar la Raspberry Pi para que se comporte como una baliza que emita anuncios del protocolo Eddystone desarrollado por Google. En este ejemplo de configuración se utilizará un mensaje con la especificación Eddystone-URL, donde se permite al usuario abrir la URL que se está transmitiendo. Los dispositivos que detecten la baliza emulada deben ejecutar un escaneo que pueda comprender el protocolo Eddystone-URL. Además, la baliza Eddystone emulada también debe mostrar su nombre y datos de potencia.

El primer paso es encender el servicio Bluetooth con el comando `sudo hciconfig hci0 up`, con el Bluetooth de la Raspberry Pi encendido, se activa el modo de anuncio de bajo consumo y se configura en el modo de anuncio no dirigido y no conectable. Esto se hace ejecutando el comando siguiente:

```
sudo hciconfig hci0 leadv 3
```

El argumento `leadv` activa el modo de anuncio de baja energía y el número que sigue especifica el modo que se quiere usar (como se mostró en la Tabla 4.1), en este caso se activa el modo de baja energía y el modo clásico de Bluetooth. Después de tener configurada la Raspberry Pi en el modo correcto, se utiliza el comando `hcitool`, esta herramienta nos permitirá enviar un comando a nuestro dispositivo Bluetooth para transmitir por ejemplo, una URL específica. Para el comando de ejemplo se utiliza la URL de la página del departamento de computación del Cinvestav: https://www.cs.cinvestav.mx/. Posteriormente se inicia el proceso para generar fácilmente el comando que transmitirá la URL como una baliza de tipo Eddystone, para poder ingresarla en el comando, la URL debe primero convertirse a su representación en código hexadecimal ASCII. Después de tener configurado el paquete de datos de Eddystone para que transmita la Raspberry Pi, se ingresa como un comando en la terminal (el comando debe ingresarse en una sola línea) de la manera siguiente:

```
sudo hcitool -i hci0 cmd 0x08 0x0008 1e 02 01 06 03 03 aa fe 16 16
aa fe 10 00 01 63 73 2e 63 69 6e 76 65 73 74 61 76 2e 6d 78 2f 00
```

La mayoría de los valores del paquete de datos deben de mantenerse como están, ya que el protocolo Eddystone lo requiere de esa manera. Una restricción a tener en cuenta es que la URL que se configure puede tener una longitud máxima hasta de 17 bytes.

En la Tabla 4.2 se describe detalladamente un ejemplo de paquete de datos con el protocolo Eddystone-URL.

Prueba de funcionamiento de baliza Eddystone-URL

Al ejecutar la aplicación iBKS Config Tool se realiza automáticamente el escaneo de dispositivos BBE cercanos. En la Figura 4.4 se observan los datos de la baliza emulada con la Raspberry Pi y configurada con el protocolo Eddystone-URL.

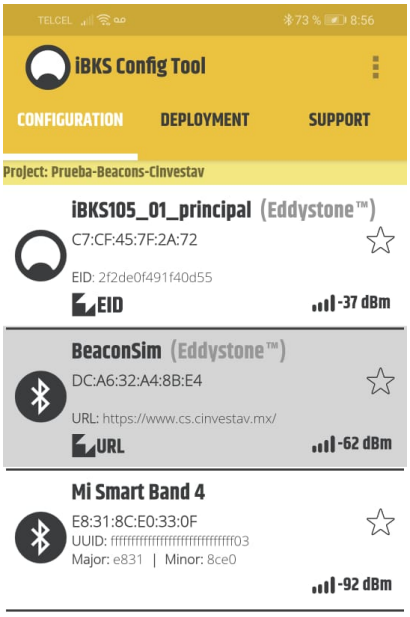


Figura 4.4: Escaneo con iBKS Config Tool.

En los datos de configuración de la baliza Eddystone emulada se puede observar la URL: `https://www.cs.cinvestav.mx/`, además de otros datos importantes como su nombre, dirección Bluetooth y potencia de transmisión.

El proceso de emulación de balizas BBE descrito en este capítulo se utilizará en la sección de resultados para realizar las pruebas de funcionamiento del protocolo de seguridad que se propone, específicamente en la etapa de generación y transmisión de identificadores efímeros a través de las "balizas BBE".

Valor de byte	**Descripción**
0x08	Define el OGF como el grupo de comandos de Bluetooth
0x0008	Define el OCF para configurar los datos de anuncio de Bluetooth
1e	Define la longitud de datos de toda la carga útil, 30 bytes
02	Longitud de datos de la siguiente sección, 2 bytes
01	Define que esta sección son datos de bandera
06	Define la bandera necesaria para el modo de anuncio de baja energía
03	Longitud de datos de la siguiente sección, 3 bytes
03	Define la lista completa de UUID de servicio, 16 bytes
AA	UUID de Eddystone
FE	UUID de Eddystone
16	Define la longitud de los datos de servicio
16	Define que los siguientes datos son “datos de servicio” para Eddystone
AA	UUID de Eddystone de 16 bits
FE	Define el tipo de marco que en este caso es URL
10	TX Power, utilizado para calcular la distancia a la baliza Eddystone
00	Esquema de URL, se establece que es “https: //”
01	Inicio de encabezado
63	‘c’ en hexadecimal (ASCII)
73	‘s’ en hexadecimal (ASCII)
2E	‘.’ en hexadecimal (ASCII)
63	‘c’ en hexadecimal (ASCII)
69	‘i’ en hexadecimal (ASCII)
6E	‘n’ en hexadecimal (ASCII)
76	‘v’ en hexadecimal (ASCII)
65	’e’ en hexadecimal (ASCII)
73	‘s’ en hexadecimal (ASCII)
74	‘t’ en hexadecimal (ASCII)
61	‘a’ en hexadecimal (ASCII)
76	‘v’ en hexadecimal (ASCII)
2E	‘.’ en hexadecimal (ASCII)
6D	‘m’ en hexadecimal (ASCII)
78	‘x’ en hexadecimal (ASCII)
2F	‘/’ en hexadecimal (ASCII)
00	‘nulo’

Tabla 4.2: Descripción de la trama de bytes de Eddystone-URL.

Capítulo 5

Desarrollo del protocolo de seguridad

En este capítulo se explica de manera detallada el desarrollo del protocolo de seguridad propuesto para las balizas BBE. Primero se describe el diseño general del protocolo, el cual está compuesto por tres capas principales, luego, se aborda la integración de las primitivas criptográficas elegidas en cada capa y su implementación en diferentes versiones según las necesidades de los dispositivos cliente y servidor.

5.1. Diseño del protocolo de seguridad

Con el fin de proveer servicios de seguridad a las balizas BBE y las aplicaciones que las utilizan; se diseñó un protocolo de seguridad basado en tres capas, en ellas se realizan las configuraciones de conexión, compartir un secreto en común y la generación de identificadores efímeros (IDEs) a través de un algoritmo de cifrado ligero. La figura 5.1 describe las tres capas que conforman el protocolo de seguridad que se propone para las balizas BBE.

En la primera capa se establecen todas las configuraciones de los conectores y puertos de enlace entre el cliente (dispositivo móvil) y el servidor (baliza BBE), además, se configura el servidor en modo de emisión de señales BBE y el cliente en modo de escaneo de dispositivos BBE. La segunda capa incorpora el protocolo de acuerdo de llaves Diffie-Hellman utilizando la curva elíptica 25519, durante este protocolo se realiza la generación de llaves, el intercambio de llaves públicas y el cálculo del secreto compartido, este proceso se realiza en la primera interacción de los dispositivos, es decir, cuando se encienden y configuran por primera vez las balizas BBE, además, se asume que no existen atacantes en el medio en el momento del intercambio de llaves. La última capa realiza la generación de identificadores efímeros a través del algoritmo ligero Ascon, se propone que los IDEs sean generados a partir de la derivación de llaves usando una función *hash* en el secreto compartido (llave diaria) y posteriormente sean transmitidos por las balizas BBE cada 15 minutos, estos IDEs se verifican por el dispositivo cliente al recibirlos. En las secciones siguientes se describe detalladamente la programación de cada capa del protocolo de seguridad de balizas BBE.

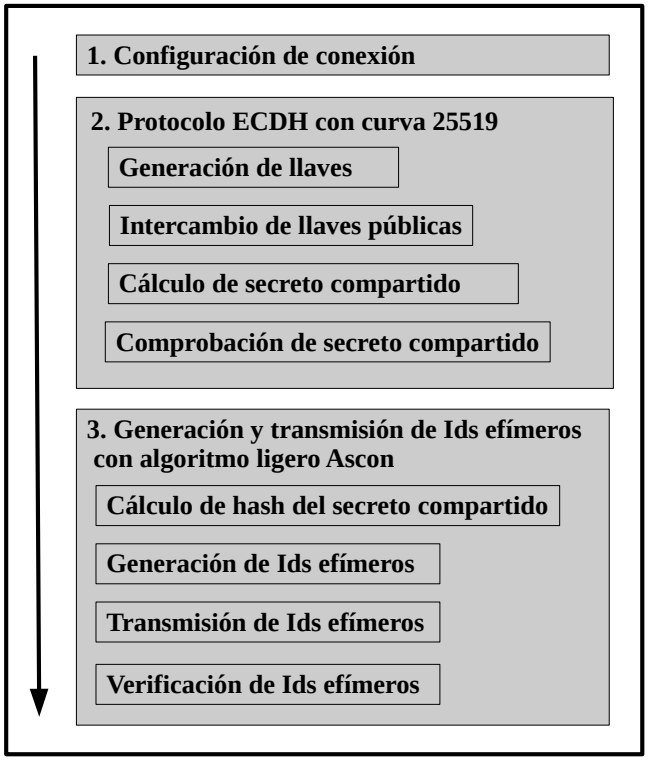


Figura 5.1: Capas del protocolo de seguridad.

5.2. Configuración de conexión

Cada módulo Bluetooth fabricado está impreso con una dirección de 48 bits única a nivel mundial, a la que se conoce como la dirección de Bluetooth. Esto es de origen idéntico a las direcciones MAC de Ethernet y Wifi, ya que ambos espacios de direcciones son administrados por la misma organización, la autoridad de registro IEEE [18]. Estas direcciones se asignan en el momento de la fabricación y están destinadas a ser únicas y permanecer estáticas durante la vida útil del módulo, además sirve convenientemente como la unidad de direccionamiento básica en toda la programación de Bluetooth y BBE.

Para lograr que un dispositivo Bluetooth se comunique con otro, se debe tener alguna forma de determinar la dirección Bluetooth del otro dispositivo. Esta dirección se utiliza en todas las capas del proceso de comunicación. Una vez que el cliente ha determinado la dirección del servidor al que desea conectarse, debe determinar qué protocolo de transporte utilizar. En este caso se utiliza el protocolo de transporte RFCOMM (*Radio Frequency Communication*), debido a que proporciona un servicio y garantías de confiabilidad muy similares al protocolo de red TCP (*Transmission Control Protocol*). En general, las aplicaciones que utilizan RFCOMM se preocupan por tener una conexión punto a punto a través de la cual puedan intercambiar flujos

de datos de manera confiable. Si una parte de esos datos no se puede entregar dentro de un límite de tiempo fijo, la conexión se interrumpe y se envía un error. A diferencia de TCP, RFCOMM sólo permite 30 puertos abiertos en un solo dispositivo. Esto tiene un impacto significativo en cómo elegir los números de puerto para aplicaciones de servidor. Una vez que se conoce una dirección numérica y un protocolo de transporte, se elige el número de puerto. En RFCOMM, los canales o puertos del 1-30 están disponibles para su uso, en la implementación del protocolo de seguridad que se propone se utiliza el puerto número 5.

En el modelo de programación de conectores o en inglés *sockets*, un conector representa un punto final de un canal de comunicación. Los conectores no están enlazados cuando se crean por primera vez y son inútiles hasta que una llamada para conectar (en una aplicación cliente) o aceptar (en una aplicación del servidor) se complete correctamente. Una vez que se conecta, se puede usar para enviar y recibir datos hasta que la conexión falle debido a un error de enlace o a la orden de terminación del usuario.

La estructura básica de datos utilizada para especificar una dirección de dispositivo Bluetooth es `bdaddr_t`, en la programación del protocolo de seguridad se utiliza la estructura de direccionamiento `sockaddr_rc`. Todas las direcciones de Bluetooth en el sistema BlueZ se almacenan y manipulan como estructuras de este tipo [18]. BlueZ proporciona dos funciones para convertir cadenas a estructuras tipo `bdaddr_t` y viceversa, las cuales se muestran a continuación.

```
int str2ba (const char * str, bdaddr_t * ba);
int ba2str (const bdaddr_t * ba, char * str);
```

La función `str2ba` toma una cadena de la forma "`XX:XX:XX:XX:XX:XX`", donde cada `XX` es un número hexadecimal que especifica un octeto de la dirección de 48 bits, y lo empaqueta en un `bdaddr_t` de 6 bytes. La función `ba2str` hace exactamente lo contrario.

A los adaptadores Bluetooth locales se les asignan números de identificación que comienzan con 0, el programa especifica el adaptador que se desea usar al asignar los recursos del sistema. Por lo general, solo hay un adaptador o no importa cuál se use, por lo que al pasar el parámetro `NULL` a la función `hci_get_route`, se recupera el número del primer adaptador Bluetooth disponible.

```
int hci_get_route (bdaddr_t * bdaddr);
int hci_open_dev (int dev_id);
```

La mayoría de las operaciones de Bluetooth requieren el uso de un conector abierto. En este sentido se utiliza la instrucción `hci_open_dev`, esta función abre un conector Bluetooth con el número de recurso especificado, es decir, el conector abierto por `hci_open_dev` representa una conexión al adaptador Bluetooth local, y no una conexión a un dispositivo Bluetooth remoto. Realizar operaciones de Bluetooth de bajo nivel implica enviar comandos directamente al microcontrolador con este conector.

Después de elegir el adaptador Bluetooth local para usar y asignar los recursos del sistema, el programa está listo para buscar dispositivos Bluetooth cercanos. La función `hci_inquiry` realiza un descubrimiento de dispositivos Bluetooth y devuelve una lista de dispositivos detectados e información básica sobre ellos en la variable `inquiry_info`. La estructura básica de la función se muestra a continuación.

```
int hci_inquiry(int dev_id, int len, int max_rsp, const uint8_t
*lap, inquiry_info **ii, long flags);
```

La instrucción `hci_inquiry` es una de las pocas funciones que requiere el uso de un número de recurso en lugar de un conector abierto, por lo que se usa el `dev_id` devuelto por la función `hci_get_route`. Además, se sugiere utilizar un `max_rsp` (máximo tiempo de respuesta) de 255 para una consulta estándar de 10.24 segundos.

Una vez que se tiene la lista de dispositivos cercanos y sus direcciones, el programa determina los nombres "fáciles de usar" asociados con esas direcciones. La función `hci_read_remote_name` es utilizada para este propósito.

```
int hci_read_remote_name (int sock, const bdaddr_t * ba, int len,
char * name, int timeout)
```

La figura 5.2 muestra el esquema general de las configuraciones de conexión entre los dispositivos que actúan como **cliente** y **servidor** en el protocolo de seguridad.

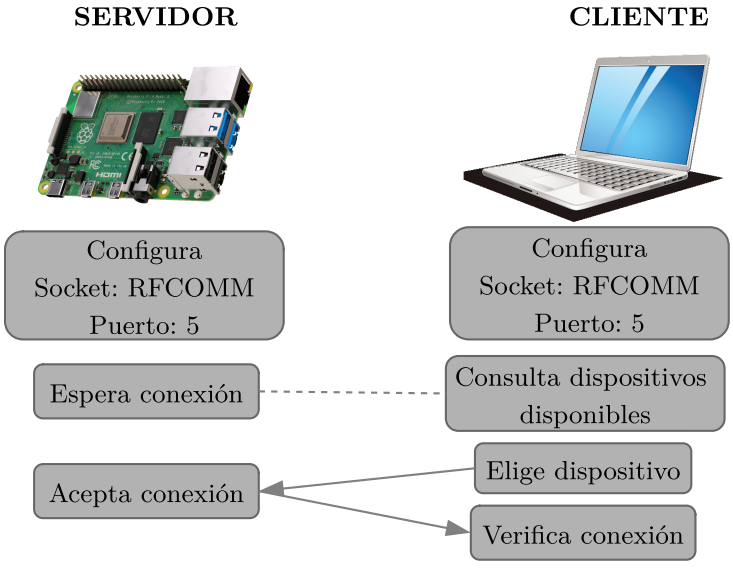


Figura 5.2: Diagrama de conexión.

El comando `hci_read_remote_name` intenta durante un tiempo de espera máximo (en milisegundos) utilizar el conector (`sock`) para consultar el nombre "fácil" del dispositivo con la dirección de Bluetooth (`ba`) . Si se tiene éxito, `hci_read_remote_name`

devuelve 0 y copia los primeros bytes del nombre del dispositivo en el `name`. En caso de error, devuelve el valor `-1`.

Para aceptar conexiones entrantes con un conector, se usa la función `bind`, para reservar el recurso del sistema operativo, después se usa el comando `listen` para ponerlo en modo de escucha/emisión y la función `accept` para bloquear y aceptar una conexión entrante. Crear una conexión saliente en el lado del cliente también es algo simple, solamente implica una llamada para conectarse. Una vez que se ha establecido una correcta conexión, el estándar de Bluetooth cuenta con las funciones `write`, `read`, `send` y `recv`, las cuales son utilizadas para la transferencia de datos entre los dispositivos.

5.3. Protocolo ECDH con curva 25519

La razón principal por la que se decidió desarrollar el protocolo de seguridad en el lenguaje de programación C, es debido a que los entornos de lenguajes de más alto nivel (como Python) no se ajustan del todo bien a los dispositivos de destino (las balizas BBE) del protocolo; los estrictos requisitos sobre el tamaño del programa, la velocidad y el uso de la memoria en dispositivos limitados, podrían llegar a impedir el uso de un lenguaje interpretado, aunado a que el control del adaptador Bluetooth local en lenguaje C es mejor que el que proporcionan otros módulos como PyBlueZ.

Antes de comenzar a integrar los algoritmos de seguridad, se deben de realizar las configuraciones de conexión según lo indica la Sección 5.2. La solución que se propone para el acuerdo de llaves en común en la primera interacción de los dispositivos clientes y las balizas BBE (servidores), es un secreto compartido con el protocolo ECDH utilizando la curva elíptica 25519, el cual usa métodos matemáticos para que las dos partes puedan derivar simultáneamente una llave en común sin haber transmitido ni un solo byte de información confidencial entre ellos. Se decidió utilizar la criptografía de curva elíptica debido a que es más eficiente en este tipo de aplicaciones, en el capítulo 3) se explican los principales beneficios de este tipo de criptografía a comparación de sistemas más antiguos como el RSA, el cual puede resultar computacionalmente costoso por sus longitudes de llaves mayores.

El proceso general del protocolo comienza con la generación de llaves privadas y públicas en cada dispositivo, cada llave cuenta con 32 bytes de longitud. En el desarrollo del protocolo de seguridad, primero, el *cliente* escoge su llave privada $a = k_{priv,C}$, a partir de la cual genera su llave pública $k_{pub,C} = [a]Q$ y la envía al *servidor*, simultáneamente el servidor escoge su llave privada $b = k_{priv,S}$, genera su llave pública $k_{pub,S} = [b]Q$ y la envía al cliente. El cliente dispone de los parámetros $(\mathbb{G}, Q, a, b[Q])$ y calcula $k_{CS} = [a]([b]Q)$. El servidor dispone de los parámetros $(\mathbb{G}, Q, b, [a]Q)$ y calcula $k_{SC} = [b]([a]Q)$. Al final se comprueba fácilmente que ambas entidades calculan el mismo secreto compartido de 32 bytes como se observa en la ecuación 5.1.

$$\frac{K_{CS} = [a]k_{pub,S} = [a]([b]Q) = [a][b]Q}{K_{SC} = [b]k_{pub,C} = [b]([a]Q) = [a][b]Q} \Longrightarrow K_{SS} \tag{5.1}$$

La figura 5.3 ilustra el proceso del protocolo ECDH que se describió en el Algoritmo 1, en el cual las entidades Alicia y Beto representan a los dispositivos que actúan como cliente y servidor, estos generan su par de llaves, intercambian sus llaves públicas y calculan su secreto compartido a través de la función X25519.

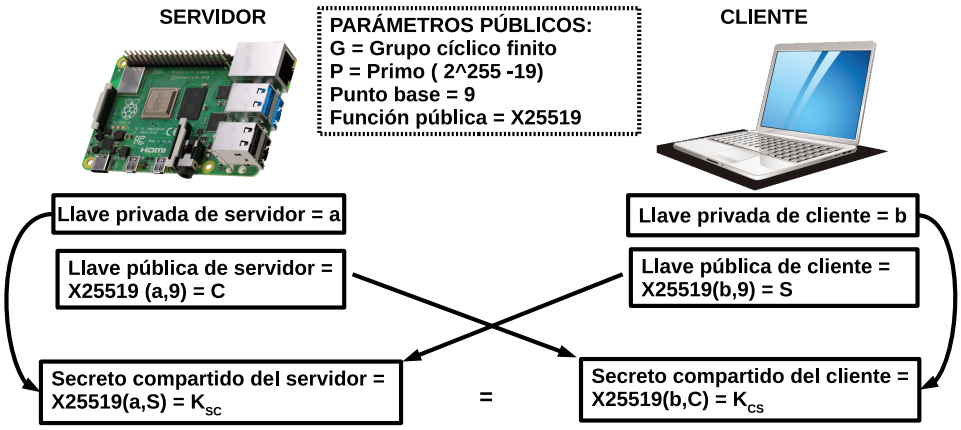


Figura 5.3: Protocolo ECDH con la curva 25519.

El protocolo ECDH con la curva 25519 se implementa con base en dos diferentes versiones del protocolo, la primera (para la parte del cliente) fue tomada de un programa que usa un procedimiento de escalera de Montgomery para curvas elípticas primarias y que admite el uso extensivo de precálculo, obteniendo de esa forma importantes aceleraciones para las implementaciones de software.

En la segunda versión (para el servidor) del protocolo ECDH, se utiliza un programa optimizado para dispositivos con microcontroladores ARM, esta implementación integra la operación de multiplicación escalar con instrucciones en lenguaje ensamblador que son llamadas por una función en lenguaje C para la generación de llaves públicas. A continuación se describen las funciones que integran ambas versiones del protocolo ECDH con curva 25519 en los dispositivos cliente y servidor.

5.3.1. Protocolo ECDH en el cliente

La publicación de Oliveira et al. [19] propone una implementación optimizada del protocolo ECDH, describe una variante de la escalera de Montgomery que permite acelerar la función de multiplicación de punto fijo inherente en la fase de generación del par de llaves Diffie-Hellman.

La propuesta combina una versión de derecha a izquierda de la escalera de Montgomery junto con el cálculo previo de valores constantes derivados directamente del punto base y sus múltiplos. Todo esto a cambio de recursos de memoria muy modestos y un pequeño esfuerzo extra de programación, la escalera propuesta obtiene importantes aceleraciones para las implementaciones de software, de igual manera cumple

totalmente con la especificación RFC 7748 [14] de las funciones X25519 y X448. Para ejecutar esta versión del protocolo ECDH optimizada se utilizan las funciones descritas a continuación.

La función `random_X25519_key (private_key_client)` genera la llave privada del protocolo ECDH en el lado del cliente. Luego de obtener la llave privada del cliente (`private_key_client`) se genera la llave pública con la función siguiente:

```
X25519_KeyGen(public_key_client, private_key_client);
```

En esta función se implementa la multiplicación escalar acelerada, al "multiplicar" la llave privada del cliente (`private_key_client`) por el punto base de la curva 25519, generando la llave pública intercambiable (`public_key_client`).

Se realiza el intercambio de las llaves públicas a través de los conectores definidos entre el cliente y el servidor, en el proceso de envío y recepción de las llaves públicas se utilizan las funciones de transferencia de datos `write` y `read` respectivamente. A continuación, se muestra la forma en que son utilizadas estas funciones definidas por el estándar Bluetooth.

```
read(socket, public_key_server, sizeof(public_key_server));

write(socket, public_key_client, sizeof(public_key_client));
```

La variable `socket` es el conector configurado en la primera capa del protocolo, ambas funciones de transferencia deben de contar con un conector, la variable con la información que se envía o recibe y el tamaño de los datos en bytes. Una vez que se recibe la llave pública enviada por el servidor, se puede calcular el secreto compartido con la función siguiente:

```
X25519_Shared(shared_client,public_key_server,private_key_client);
```

El secreto compartido del lado del cliente se almacena en la variable `shared_client` para su comprobación posterior. Para emplear de manera adecuada el secreto compartido, en lugar de usarlo directamente, se debe de implementar siempre una función de derivación de llaves en ambos dispositivos. Sería incorrecto utilizar como "llave raíz" el secreto compartido resultante, es necesario pasarlo por un bloque KDF (*key derivation function*), el cual encadena una o más llaves secretas de un valor como el secreto compartido, usando una función pseudoaleatoria. La derivación de llaves garantiza que las llaves utilizadas cuenten con propiedades deseables de llave segura como la extensión de longitud y variación de formato. Estas propiedades pueden evitar que un atacante obtenga una llave derivada consiguiendo información útil sobre el secreto compartido o las demás llaves derivadas. Las funciones *hash* son las funciones pseudoaleatorias más populares usadas para la derivación de llaves, proporcionan mayor seguridad ya que el protocolo ECDH se aplicaría solo una vez y después se realizaría la derivación a través del modo *hash* del algoritmo ligero Ascon.

5.3.2. Protocolo ECDH en el servidor

La versión del protocolo ECDH con curva 25519 [20] que se utiliza en el lado del servidor, implementa instrucciones en lenguaje ensamblador altamente optimizadas de la función X25519 definida para un microcontrolador ARMv7. Dicha implementación está optimizada para un grupo de microprocesadores sencillos como los Cortex-M4, aunque también funciona en otros procesadores de arquitectura ARM (ARMv7 y arquitecturas más nuevas de 64 bits) como la que utiliza una Raspberry Pi. En esta versión del protocolo ECDH con curva 25519 se utiliza la función siguiente para generar la llave privada.

```
random_X25519_key(private_key_server,32);
```

Esta función genera un valor aleatorio de 32 bytes de llave privada usando un generador de números criptográficamente seguro. Luego, la llave privada obtenida (`private_key_server`) se introduce en la función siguiente para generar la llave pública del servidor (`public_key_server`) que se envía al cliente.

```
X25519_calc_public_key(public_key_server, private_key_server);
```

Después de enviar la llave pública del servidor, se recibe la llave pública del cliente (`public_key_client`), esta se introduce junto con la llave privada del servidor en la función que calcula el secreto compartido (`shared_server`) como se muestra a continuación.

```
X25519_calc_shared_secret(shared_server,public_key_client,
private_key_server);
```

El secreto compartido calculado por el servidor se almacena en la variable `shared_server`, para finalmente compararlo con el secreto compartido calculado del lado del cliente y autenticar la conexión de los dispositivos.

El programa utilizado para el protocolo ECDH del lado del servidor usa solo 1892 bytes de espacio de código en forma compilada, además utiliza 368 bytes de pila y ejecuta una multiplicación escalar en 548,873 ciclos en un microcontrolador Cortex-M4, el cual es un récord de velocidad para un procesador de 64 MHz, eso significa que le toma menos de 9 milisegundos ejecutar una operación. También existe una versión aún más optimizada que utiliza la unidad de punto flotante (FPU), esta se ejecuta en 476,275 ciclos en un microntrolador ARM Cortex-M4F [20]. La versión básica se ejecuta en tiempo constante y utiliza un patrón de acceso a memoria igualmente constante, independientemente de la llave privada que protege contra ataques de canal lateral. Sin embargo, la versión FPU lee datos de la RAM en un patrón no constante, por lo tanto, esa versión solo es adecuada para dispositivos sin caché de datos, como los microcontroladores Cortex-M4 y Cortex-M33.

La implementación que se utiliza para el protocolo de seguridad es la versión básica, ya que esta puede ejecutarse en dispositivos sencillos con microcontroladores

ARM Cortex-M4, así como en otros más complejos como el microcontrolador ARMv7 de la Raspberry Pi.

5.4. Generación de identificadores efímeros

En la generación de los IDEs que serán asignados a las balizas BBE (servidores), se optó por utilizar el algoritmo de cifrado ligero Ascon, así como su modo *hash* para la generación de llaves diarias.

Como primer paso se debe generar una llave diaria de la forma $K_d = H(K_{d-1})$ (usando el modo *hash* de Ascon) para $d > 0$, tomando $d_0 = K_{SS}$ (secreto compartido), cada llave diaria se analiza como $d_i \rightarrow (k_d, n_d)$. K_d tiene una longitud de 32 bytes, mientras que k_d y n_d tienen una longitud de 16 bytes. Para generar los IDEs, se utiliza el algoritmo ligero Ascon como un generador pseudoaleatorio con datos asociados que actúan como sal [1]. En la fase de inicialización, los parámetros de entrada n_d y k_d provienen de la llave diaria K_d, y se usa una cadena de corrección adicional como datos asociados para Ascon (similar a la sal del *hashing*).

En la figura 5.4 se ilustra la generación de IDEs; primero se observa como se ingresa la llave diaria como las entradas K y N en la fase de inicialización, después se establece la cadena de sal, esta se maneja como datos asociados A_1, A_s, posteriormente se pasa por un bloque de permutación adicional y se emite el primer IDE. Después del tiempo establecido se genera el siguiente IDE, retroalimentando el resultado anterior al bloque de rondas de permutación p^b. Por lo tanto, las salidas de 128 bits del bloque p^b son los IDEs que que se asignarán a las balizas BBE cada 15 minutos aproximadamente.

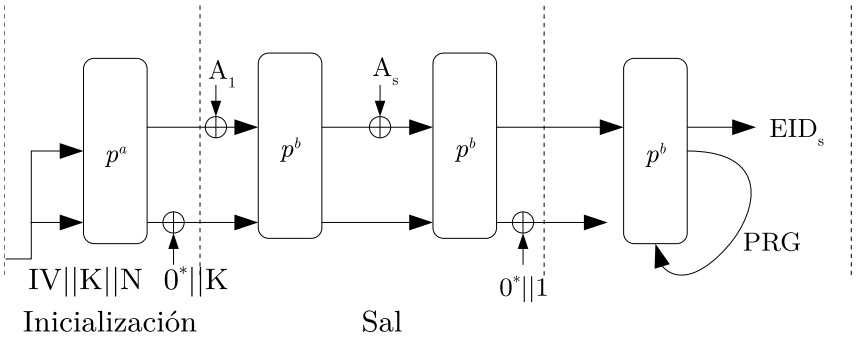


Figura 5.4: Diagrama de generación de identificadores efímeros.

En la ecuación 5.2 se observa que el primer IDE se obtiene directamente del proceso del algoritmo Ascon, pasando por las fases de inicialización (I) y Sal (S) con los datos asociados, mientras que los siguientes IDEs serán las salidas de la retroalimentación de identificadores anteriores dadas al último bloque de p^b, el cual implementa ocho rondas de permutaciones con operaciones ligeras y es por ello que se utiliza como un

[1] Bits aleatorios que se usan como entrada en una función de derivación de llaves

generador pseudoaleatorio de IDEs.

$$\begin{aligned} IDE_1 &\leftarrow p^b \leftarrow S \leftarrow A_1, A_S \leftarrow I \leftarrow k_d, n_d \\ IDE_S &\leftarrow p^b \leftarrow IDE_1 \\ IDE_S &\leftarrow p^b \leftarrow IDE_{S-1} \end{aligned} \tag{5.2}$$

Se pretende que los IDEs sean generados y transmitidos por el servidor, con un intervalo de tiempo de 15 minutos, hasta que un cliente que ha sido autenticado por el protocolo ECDH recoja un IDE y lo verifique. El algoritmo 3 describe el proceso de generación de IDEs a través del generador pseudoaleatorio basado en los bloques de permutación del algoritmo ligero Ascon.

Algoritmo 3 Descripción de la generación de IDEs

Entrada: DA (datos asociados), k_d (llave Ascon), n_d (*nonce* Ascon), num_IDEs (número de IDEs generados en un día), min (minutos de espera).
Salida: IDEs (identificadores efímeros)
Inicialización: DA $= 0 \ldots 0$, num_IDEs $= 96$, min $= 15 * 60$, $i = 0$

1: Generar_IDE (IDE, DA, k_d, n_d)
2: **for** $i \leq num\_IDEs$ **do**
3: PRG (IDE)
4: Emitir (IDE)
5: Sleep (min)
6: $i++$
7: **end for**

Los datos de entrada del algoritmo anterior se basan en el *hash* del secreto compartido (K_d) generado por el modo *hash* del algoritmo Ascon, esta llave diaria tiene una longitud de 32 bytes que son repartidos de forma equitativa en las variables k_d y n_d (con 16 bytes de longitud cada una), además, los datos asociados (A_1, A_S) que actúan como sal en el algoritmo propuesto, se inicializan con una cadena de bytes con valores 0. La cantidad de IDEs que se generan en un día es el resultado de multiplicar los IDEs generados en un hora (son 4) por las 24 horas, dando como resultado un total de 96 IDEs diarios que se transmitirán a través de una señal BBE y posteriormente captados por un cliente que los verifique. Las pruebas de funcionamiento de la transmisión de IDEs a través de señales BBE se exponen en el capítulo 6 de resultados, en la página 71.

Capítulo 6

Resultados

En las pruebas de implementación del protocolo de seguridad se utilizaron principalmente dos dispositivos con adaptadores BBE: una computadora portátil y una Raspberry Pi 4, a las cuales llamamos *cliente* y *servidor* respectivamente. En la tabla 6.1 se muestran las características principales de los dispositivos utilizados durante las pruebas de implementación.

Característica	**Computadora portátil**	**Raspberry Pi 4**
Procesador	AMD A10-8700p	ARM Cortex-A72
Frecuencia	4 núcleos a 1.8 GHz	4 núcleos a 1.5 GHz
GPU	AMD Radeon R6	VideoCore VI
RAM	12 GB	4 GB
Caché	2 MB	32 KB de datos, 48 KB de caché L1 y 1 MB de caché L2
Capacidad de almacenamiento	1 TB	16 GB
Conectividad	Bluetooth 4.0, Wi-Fi y Gigabit Ethernet	Bluetooth 5.0, Wi-Fi y Gigabit Ethernet
Puertos	Alimentación, USB 2.0, HDMI, Lector SD y Ethernet	GPIO, micro HDMI, USB 2.0 y 3.0 Micro SD, USB-C y Ethernet
Sistema operativo	Ubuntu 20.04.2 LTS	Raspberry OS

Tabla 6.1: Características de dispositivos utilizados.

La computadora portátil actuó como un cliente ya que realiza escaneos en busca de dispositivos BBE disponibles, en este caso se conectó a la Raspberry Pi 4. La placa se comportó como un servidor esperando la solicitud de conexión de algún cliente, se configuró a la Raspberry Pi en modo de escucha debido a que el estado general de las balizas BBE que se desea emular es su modo de emisión BBE permanente.

Lo primero que se integró al programa del protocolo de seguridad fue la biblioteca Bluetooth y sus componentes necesarios para la conexión BBE, después, se realizó la configuración de los puertos y conectores para el enlace BBE entre la computadora portátil y la Raspberry Pi.

Con el fin de integrar las primitivas criptográficas elegidas para el protocolo de seguridad de las balizas, se desarrollaron un par de programas, en un principio se hicieron pruebas de conexión BBE y la generación de llaves del protocolo ECDH en el lenguaje de programación Python, aunque posteriormente se optó por implementar el protocolo completamente en lenguaje C debido a su velocidad de cálculo y su eficiencia en aplicaciones de este tipo. En la sección siguiente se describe el proceso y los resultados obtenidos del programa desarrollado en lenguaje Python con el protocolo ECDH del lado del servidor.

6.1. Programación en lenguaje Python

En primera instancia se decidió utilizar el lenguaje de programación Python por sus características de flexibilidad y portabilidad, además de las diversas bibliotecas especializadas que se pueden incorporar, como el módulo PyBluez para las conexiones Bluetooth.

6.1.1. PyBluez

PyBluez es una biblioteca escrita en Python, utiliza los recursos Bluetooth y BBE con el objetivo de permitir a los desarrolladores poder crear aplicaciones que hagan uso de Bluetooth de una forma mucho más sencilla y rápida. Se puede hacer uso de esta biblioteca en los sistemas operativos Windows, Linux y MacOS, además está disponible de forma gratuita bajo la Licencia pública general de GNU en su sitio oficial[1].

Antes de ejecutar el comando de instalación se deben tener las dependencias necesarias para cada sistema operativo, en el caso de Linux, se requiere de Python 2.7 o una versión más reciente, de igual forma se debe tener instalado el sistema BlueZ para el manejo de conexiones BBE (el manual de instalación esta disponible en el Apéndice C, en la página 89). Para instalar PyBluez se utilizó el comando siguiente:

```
pip install pybluez
```

Después de tener instalado el módulo PyBluez se creó un programa en lenguaje Python, en este programa se importó el módulo PyBluez y se establecieron los conectores y puertos que se utilizan para el enlace Bluetooth entre los dispositivos que actúan como cliente y servidor.

[1] https://pypi.org/project/PyBluez/

6.1.2. Cython

Para optimizar la generación de llaves públicas del protocolo ECDH con curva 25519, se empleó la extensión Cython para usar funciones más eficientes hechas en lenguaje C en el programa codificado en Python, debido a que este realiza la conexión principal entre los dispositivos. En el desarrollo de este programa se probó el uso del compilador Cython para mandar a llamar funciones en lenguaje C, diseñadas específicamente para los dispositivos con arquitectura tipo ARM, dichas funciones implementan una multiplicación escalar en lenguaje ensamblador para acelerar el proceso de generación de llaves públicas.

Cython es un compilador estático de optimización para el lenguaje de programación Python, permite que escribir extensiones C para Python sea tan fácil como el propio Python. Cython brinda el poder combinado de Python y C para permitir diversas acciones. El objetivo de Cython es convertirse en un superconjunto del lenguaje Python que le proporcione una programación dinámica, funcional, orientada a objetos y de alto nivel. Su característica principal es el soporte para declaraciones de tipos estáticos opcionales como parte del lenguaje. El código fuente se traduce en código C/C++ optimizado y se compila como módulos de extensión de Python. Esto permite una ejecución muy rápida del programa y una estrecha integración con las bibliotecas de C externas, al tiempo que mantiene la alta productividad del programador por la que el lenguaje Python es bien conocido [21]. A diferencia de la mayoría de los programas de Python, Cython requiere que haya un compilador de C en el sistema. Los detalles para obtener un compilador de C varían según el sistema utilizado:

- Linux: El compilador GNU (gcc) suele estar presente o fácilmente disponible a través del sistema de paquetes. En Ubuntu o Debian, por ejemplo, el comando buscará todo lo que necesita con el comando siguiente:

  ```
  sudo apt-get install build-essential
  ```

- Mac OS: Para utilizar gcc, una opción es instalar el programa XCode de Apple, que se puede obtener de los DVD de instalación de Mac OS X o de su sitio oficial[2].

- Windows: Una opción popular es utilizar MinGW de código abierto (una distribución de Windows de gcc). Otra alternativa es usar Visual C de Microsoft. Luego, se debe utilizar la misma versión con la que se compiló el Python instalado.

La forma más sencilla de instalar Cython en un sistema Linux es mediante el sistema de paquetes `pip` mediante el comando:

```
pip install Cython
```

[2]https://developer.apple.com/

La versión más reciente de Cython se puede descargar desde la página oficial[3]. Se desempaqueta el archivo tarball o zip, se ingresa al directorio y luego se ejecuta con el comando siguiente:

```
python setup.py install
```

Existen varias formas de crear una biblioteca con Cython, la más recomendada, es aquella que necesita de un *setuptools* llamado `setup.py` y de un archivo `pyx` donde se declaran las funciones en C que se compilarán con Cython. Para construir el archivo Cython, se utilizó el comando siguiente:

```
python setup.py build_ext --inplace
```

El comando anterior generó un archivo de extensión `.so` en Linux o un `.pyd` en Windows, después, para utilizar la biblioteca simplemente se importó como un módulo de Python normal en un programa en lenguaje Python.

La primera versión del protocolo de seguridad desarrollado en Python con una biblioteca Cython realizó la generación de llaves del protocolo ECDH y el cálculo del secreto compartido, igualmente se comprobó que al utilizar la biblioteca Cython el tiempo de ejecución durante el protocolo ECDH es menor que al usar solamente Python. La figura 6.1 muestra el tiempo de ejecución del programa con el protocolo ECDH en el lenguaje Python sin el módulo de optimización Cython.

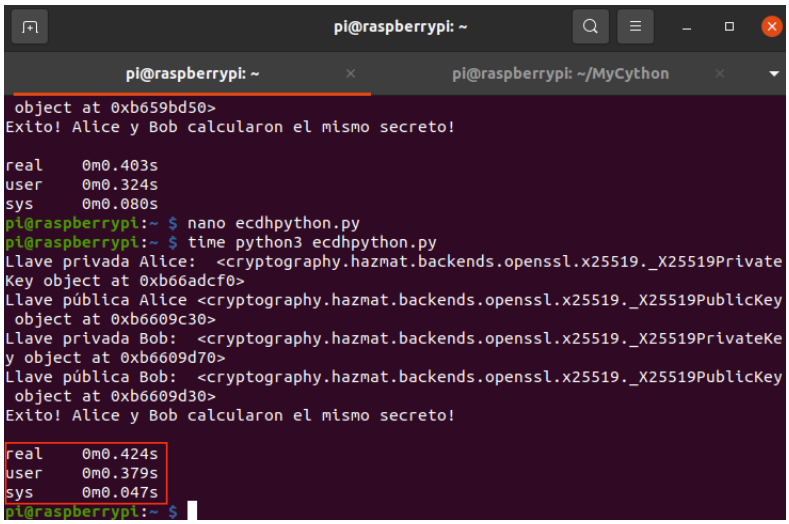


Figura 6.1: Tiempo de ejecución del protocolo ECDH en Python.

En la figura 6.2 se observa el tiempo de ejecución del programa con el protocolo ECDH en lenguaje Python utilizando el módulo Cython para la optimización de la generación de llaves públicas con funciones programadas en C. Al hacer una comparación de los tiempos de ejecución de ambos programas se puede apreciar que el

[3]https://cython.org/

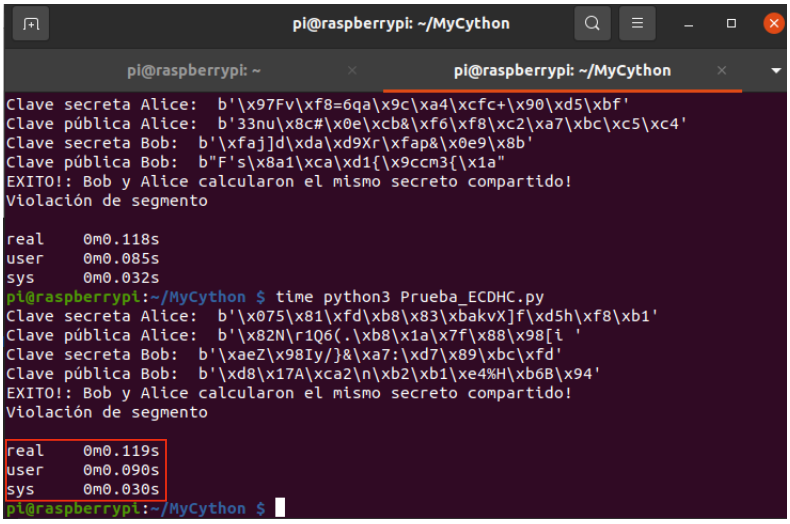


Figura 6.2: Tiempo de ejecución del protocolo ECDH con Cython.

tiempo del programa que incluye el módulo Cython es cuatro veces más rápido que el programa hecho en solo Python. Sin embargo, a pesar del aporte en velocidad de procesamiento y ejecución que brindan los módulos Cython a los programas en Python, este no deja de ser un lenguaje interpretado y por ello no se compara a la eficiencia que puede tener un programa implementado totalmente en lenguaje C.

En la figura 6.3 se muestra el tiempo de ejecución de un programa codificado en lenguaje C donde se implementa el protocolo ECDH con la curva 25519, se puede ver que el tiempo de ejecución es nueve veces más rápido que tiempo del programa desarrollado en Python con el módulo Cython.

La tabla 6.2 ilustra la comparación de los tiempos de ejecución de los tres programas que implementan el protocolo ECDH con la curva 25519, desarrollados en distintos lenguajes de programación.

Lenguaje	**Programa**	**Tiempo (seg)**
Python	ECDH	0.424
Python/Cython	ECDH	0.119
C	ECDH	0.012

Tabla 6.2: Comparación de tiempos de ejecución.

En la tabla 6.2 se observó que el tiempo de ejecución más eficiente de las implementaciones del protocolo ECDH, es el desarrollado en lenguaje C, con un tiempo 9 veces más rápido que el programa en Python con el módulo Cython y aproximadamente 40 veces más rápido que el programa codificado solamente en Python.

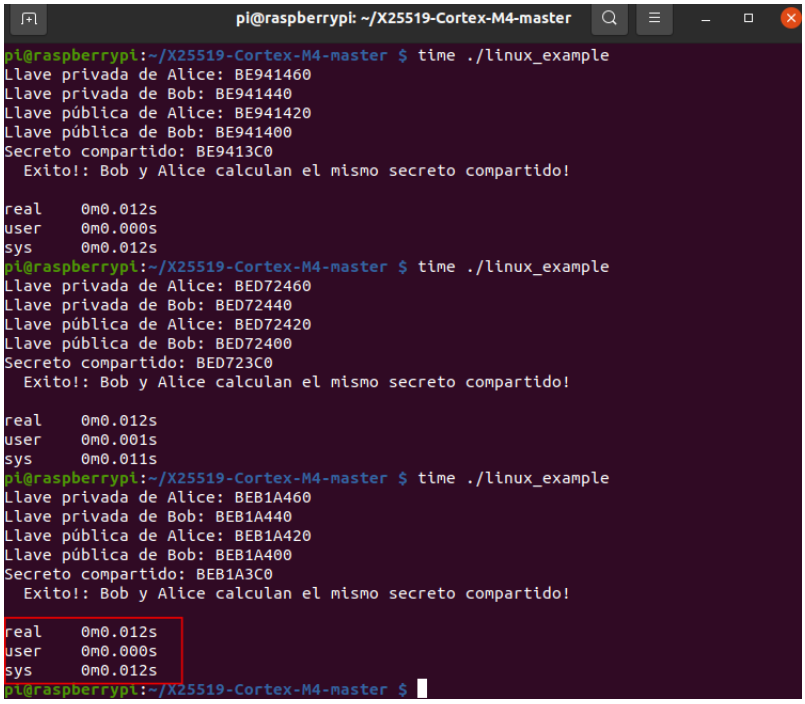
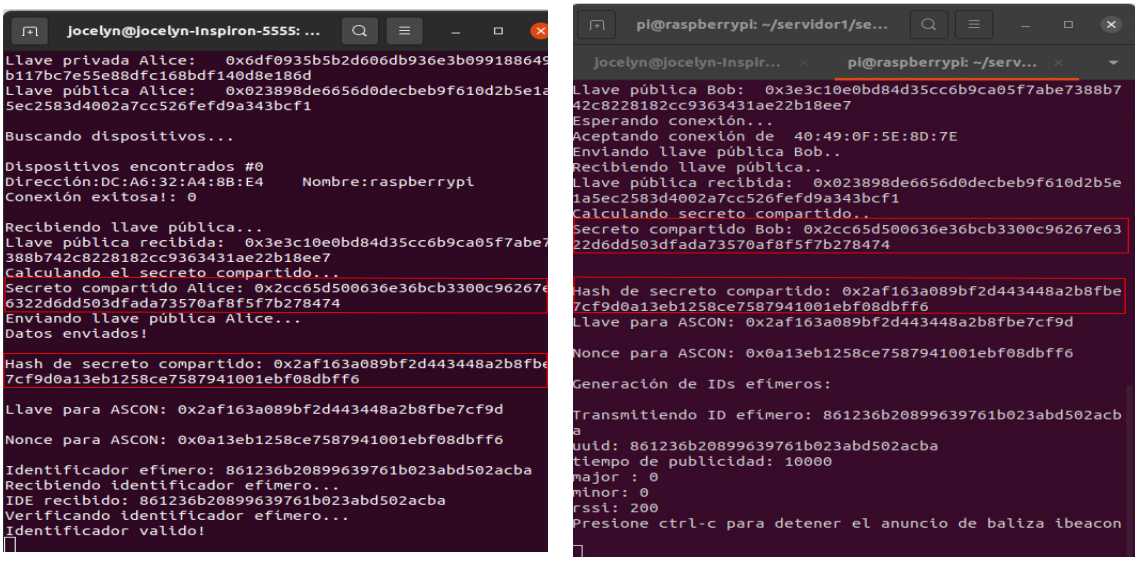


Figura 6.3: Tiempo de ejecución del protocolo ECDH en lenguaje C.

6.2. Pruebas de funcionamiento del protocolo ECDH integrado

Se decidió utilizar una Raspberry Pi 4 en las pruebas de implementación del protocolo debido a que puede emularse fácilmente una baliza BBE, además, se tiene el control absoluto para modificar sus protocolos y controladores. De igual manera al trabajar en una Raspberry Pi no se tienen las restricciones de las balizas comerciales donde no se pueden acceder a sus microcontroladores internos para su programación. Luego de que se realiza la configuración de la conexión Bluetooth en ambos dispositivos, según lo indica la primera capa del protocolo de seguridad; se implementó el protocolo de acuerdo de llaves con curva elíptica (ECDH) en cada lado de la conexión, de esta manera se llevó a cabo el cálculo del secreto compartido que proporcionó un servicio de autenticación durante la primera interacción de los dispositivos.

El protocolo de establecimiento de llaves en común se realizó con base en dos diferentes versiones del protocolo ECDH que utilizan la curva elíptica 25519, la primera (para el lado del cliente) es una variante que realiza precómputo [20] en la operación de multiplicación escalar en forma de Montgomery, la cual es utilizada durante la generación de llaves públicas. La segunda versión del protocolo ECDH con la curva 25519 (para el lado del servidor) es un programa optimizado para dispositivos con microcontroladores ARM [19], ya que esta implementación realiza la multiplicación escalar en lenguaje ensamblador para obtener una mayor eficiencia durante el proceso de generación de llaves públicas. Después de comprobar que el secreto compartido



(a) Terminal del cliente. (b) Terminal del servidor.

Figura 6.4: Ejecución del protocolo de seguridad.

calculado por ambos dispositivos haya sido el mismo, se obtuvo una llave diaria (K_d) utilizando el modo *hash* de Ascon sobre el secreto compartido, este *hash* debe ser calculado cada 24 horas debido a los ataques que podrían ocurrir durante el intercambio de llaves del protocolo ECDH, por tal razón es implementado una sola vez aminorando esta amenaza de seguridad. Posteriormente se tomó la llave diaria y se repartió en las entradas (llave y *nonce*) del algoritmo de Ascon diseñado para la generación de identificadores efímeros (IDEs).

En la figura 6.4 se observa la ejecución de los programas con la implementación del protocolo de seguridad en ambos dispositivos, en la figura 6.4a se puede apreciar la ejecución del protocolo ECDH del lado del cliente, así como la derivación de llaves aplicando la función *hash* al secreto compartido para la generación de los IDEs esperados. La figura 6.7b muestra la terminal del servidor durante la ejecución del protocolo ECDH, el cálculo del mismo secreto compartido y el respectivo *hash* para la generación de los IDEs que serán transmitidos.

6.3. Transmisión y verificación de IDEs

Después de realizar la generación de IDEs en el lado del servidor, se comprobó su funcionamiento transmitiéndolos a través de una baliza BBE emulada con la Raspberry Pi (proceso descrito en el capítulo 4). Los IDEs se generaron cada 15 minutos y se transmitieron por el servidor hasta que el cliente los recogió y verificó con los IDEs calculados por él, esto provoca que cualquier usuario malicioso que espíe la red en busca de identificadores validos para utilizar, esté imposibilitado de calcular el IDE

siguiente ni el anterior, ni tampoco emparejarse con una baliza de confianza, disminuyendo de esta forma algunas amenazas de seguridad como el ataque de suplantación de balizas BBE.

El identificador único (UUID) de una baliza con el protocolo iBeacon es generalmente estático y por lo tanto más vulnerable a ataques de suplantación; en la figura 6.5 se observa el cambio del UUID de la baliza iBeacon emulada (actúa como un IDE), de esta manera se comprueba el funcionamiento de la tercera capa del protocolo de seguridad propuesto, la cual se basa en la generación, transmisión y verificación de IDEs.

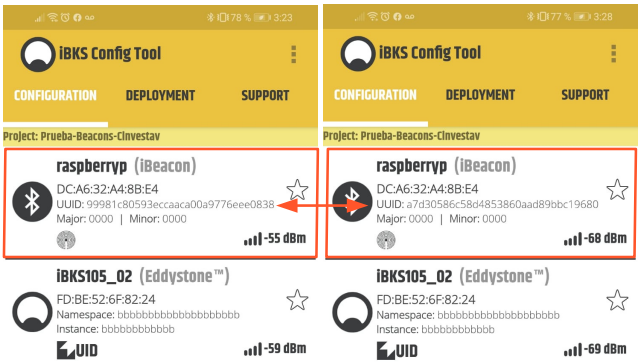


Figura 6.5: Transmisión de identificadores efímeros.

La figura 6.6 ilustra el proceso completo de interacción del protocolo de seguridad entre el cliente y el servidor. El diagrama presenta la integración de las principales funciones de las capas del protocolo de seguridad que se propuso para las balizas BBE, se comenzó con las configuraciones de conexión BBE, siguiendo con el proceso del protocolo ECDH y posteriormente el cálculo del *hash* del secreto compartido para la generación de IDEs a través del algoritmo ligero Ascon, finalmente se transmitieron los IDEs y se verificaron del lado del cliente.

La figura 6.7 presenta de forma más detallada el proceso de generación, transmisión y verificación de cuatro IDEs, resultantes de la implementación de la tercera capa del protocolo de seguridad; se resalta en color rojo el último IDE transmitido por el servidor y verificado por el cliente. En la figura 6.7b se observa la terminal del servidor configurado con las características de una baliza iBeacon donde el identificador (UUID) es modificado después de un intervalo de tiempo establecido previamente. La figura 6.7a muestra la terminal del lado del cliente donde son recibidos, comparados y verificados los cuatro IDEs con los que se esperan obtener (generados del lado del cliente), realizando de esta manera una autenticación por conocimientos basada en el secreto compartido que sólo conoce el cliente.

En la figura 6.8 se muestra el panel principal de la aplicación iBKS Config Tool, donde se realizó un escaneo de dispositivos BBE y se observó a la baliza iBeacon emulada (como se describe en el capítulo 4), con el identificador efímero resaltado en la figura 6.7, generado por el protocolo de seguridad.

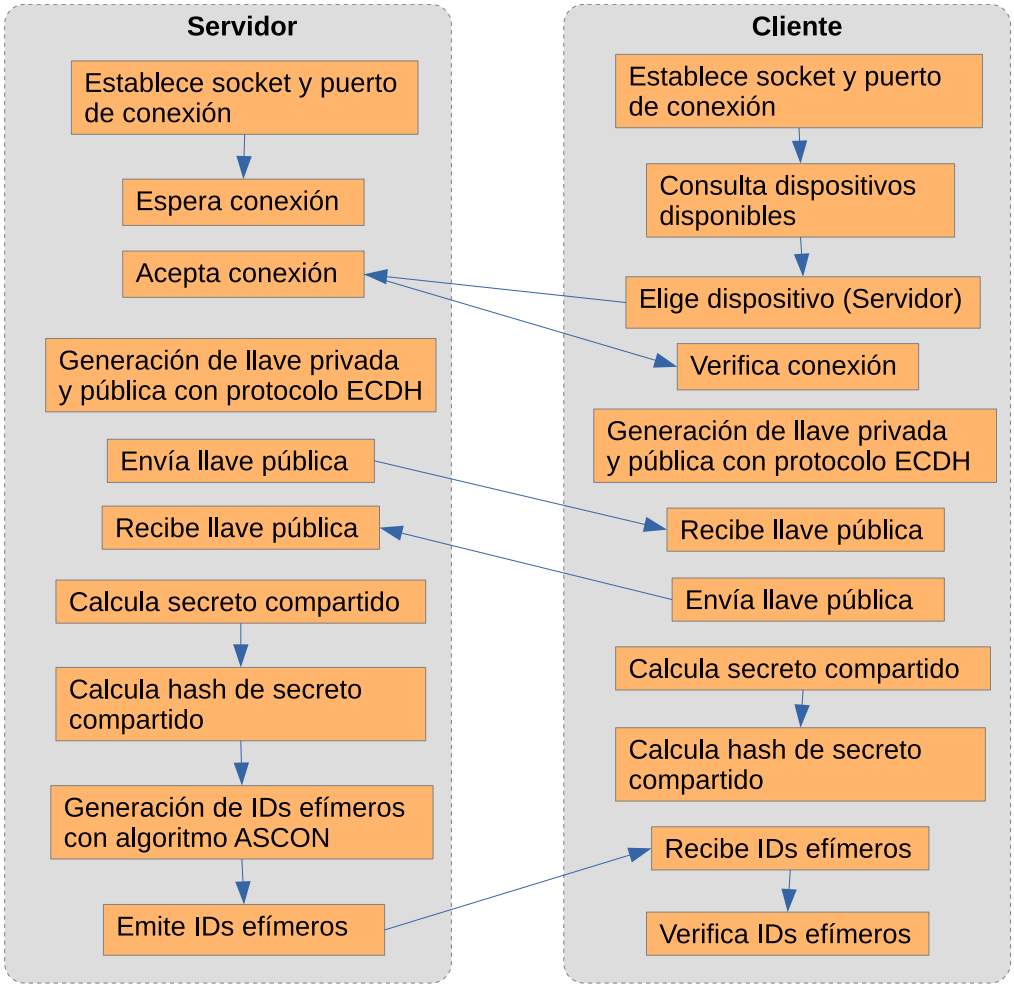
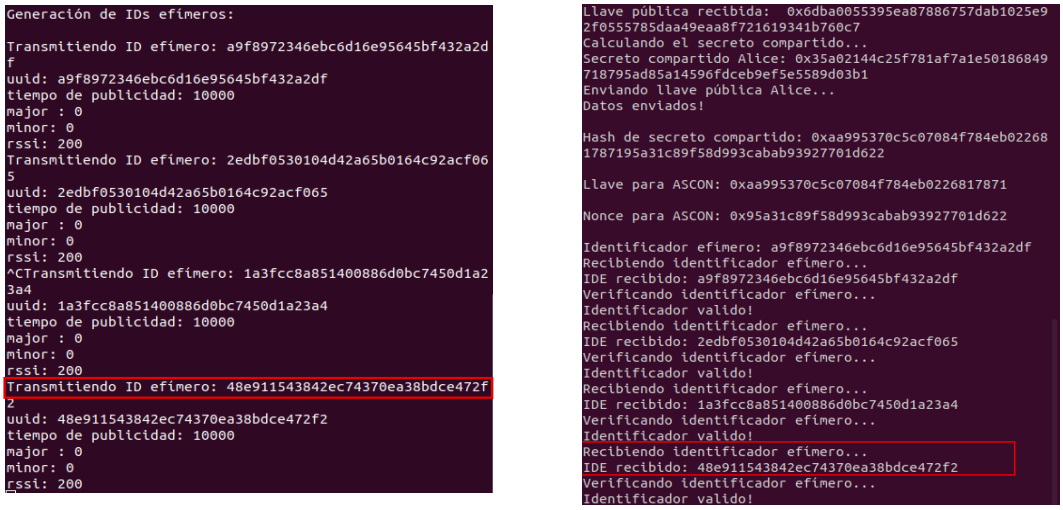


Figura 6.6: Diagrama de interacción del protocolo de seguridad.



(a) Terminal del cliente. (b) Terminal del servidor.

Figura 6.7: Generación y transmisión de IDEs

El código completo de la implementación del protocolo de seguridad de balizas BBE se encuentra disponible en el enlace siguiente:

https://gitlab.com/jocycampos1996/proyecto-ecdh-y-eid.git.

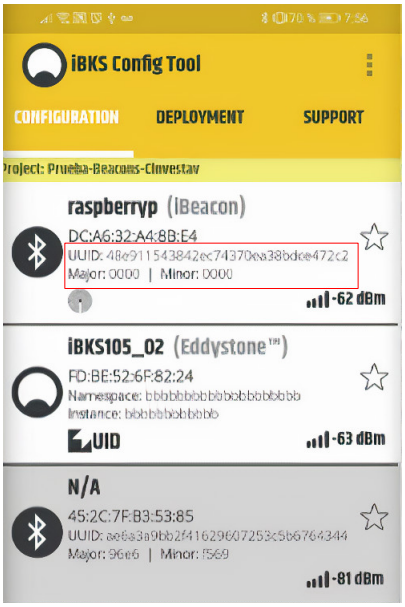


Figura 6.8: Verificación de transmisión de IDEs.

Capítulo 7

Conclusiones

En el presente trabajo se realizó un análisis de la tecnología Bluetooth de baja energía y los dispositivos restringidos que utilizan este tipo de señales, particularmente se abordaron las balizas BBE como casos de estudio. De igual forma se analizaron las principales amenazas de seguridad en las aplicaciones que utilizan balizas BBE y las posibles contramedidas ante los diversos ataques. Si bien la mayoría de los microcontroladores actuales tienen integrado el estándar de cifrado AES, este requiere de recursos suficientes para implementarlo. En la integración de algoritmos ligeros que se proponen usar en el protocolo de seguridad, se probó que funciona en un microcontrolador ARM Cortex-A7, aunque se puede implementar a partir de dispositivos con microcontroladores más sencillos como los Cortex-M4.

En este capítulo se presentan las principales contribuciones y el trabajo futuro relacionado con la propuesta del protocolo de seguridad de balizas BBE.

7.1. Contribuciones

El presente trabajo de tesis tiene como contribución principal la propuesta de un protocolo de seguridad implementado en un entorno de emulación de dispositivos BBE restringidos, en el cual se aplicaron algoritmos de criptografía ligera como los de la familia Ascon (para la generación de identificadores efímeros), así como versiones optimizadas del protocolo ECDH usando la curva 25519, todo esto para brindar servicios de seguridad en aplicaciones que implementan la tecnología de balizas BBE y que actualmente carecen de mecanismos de seguridad eficientes debido a la especificación BBE y a los recursos limitados de las balizas. Además, el proceso de configuración del protocolo podría aplicarse a otros dispositivos que tengan integrado un módulo Bluetooth con la especificación de baja energía, como los sensores deportivos, periféricos de computadora, etc.

7.2. Trabajo futuro

El protocolo de seguridad propuesto se implementó en un dispositivo que emulaba a una baliza BBE, esto se hizo debido a las restricciones de acceso y programación en las balizas comerciales. Como trabajo futuro se podría implementar el protocolo de seguridad como un módulo en la programación de los microcontroladores de las balizas BBE. Además, como sucede en la mayoría de los protocolos de seguridad emergentes, la propuesta realizada en este trabajo ofrece varias oportunidades de mejora como la integración de algoritmos criptográficos más optimizados para dispositivos restringidos.

La implementación del protocolo podría aplicarse también a la comunicación segura, ya que la base de derivación podría tener diferentes usos, como al envío de mensajes cifrados con el algoritmo de cifrado autenticado Ascon.

Otra mejora sería agregar un servicio de autenticación antes del intercambio de las llaves públicas del protocolo ECDH, esta se podría realizar a través de una entidad certificadora, de esta manera se erradicaría una de las principales vulnerabilidades en los sistemas de conexión BBE como es la suplantación de identidad.

Bibliografía

[1] Centro de recursos de seguridad informática(CSRC). Criptografía ligera. https://csrc.nist.gov/projects/lightweight-cryptography, 2017. [accedido el 25 de Octubre de 2020].

[2] F. Rodríguez-Henríquez, C. Mancillas López, and B. Ovilla Martínez. Applacovid. https://pakal.cs.cinvestav.mx/, 2020. [accedido 15 Diciembre 2020].

[3] Anca Jurcut, Pasika Ranaweera, and Lina Xu. *Introduction to IoT Security*, chapter 2. John Wiley Sons, 12 2019.

[4] Mohammad Afaneh. *Intro to bluetooth low energy.* NovelBits, Estados Unidos, 2018.

[5] Mohamed Amine Ferrag, Leandros Maglaras, Helge Janicke, Jianmin Jiang, and Lei Shu. Authentication protocols for internet of things: A comprehensive survey. *Security and Communication Networks*, 2017, 09 2017.

[6] Avinatan Hassidim, Moti Yung Yossi Matias, and Alon Ziv. Ephemeral identifiers:mitigating tracking & spoofing threats to ble beacons. *Google, Inc*, 04 2016.

[7] Hui Jun Tay Jiaqi Tan and Priya Narasimhan. A survey of security vulnerabilities in bluetooth low energy beacons. Technical report, Laboratorio de datos paralelos de la Universidad Carnegie Mellon, 2016.

[8] Radius Network. Altbeacon especifications. https://altbeacon.org/, 2015. [accedido 13 Enero 2021].

[9] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of computation*, 48:203–209, 1987.

[10] Victor S. Miller. Use of elliptic curves in cryptography. In Hugh C. Williams, editor, *Advances in Cryptology — CRYPTO '85 Proceedings*, pages 417–426, Berlin, Heidelberg, 1986. Springer Berlin Heidelberg.

[11] Ayerra Balduz I., Vázquez Lapuente M., and Jiménez Seral P. Criptografía y curvas elípticas. la curva de whatsapp. Master's thesis, Universidad de Zaragoza, España, 2018.

[12] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[13] Daniel J. Bernstein. Curve25519: New diffie-hellman speed records. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *Public Key Cryptography - PKC*, 2006.

[14] Adam Langley, Mike Hamburg, and Sean Turner. Curvas elípticas para la seguridad. https://rfc-editor.org/rfc/rfc7748.txt, 2016. [accedido 25 Enero 2021].

[15] Whatsapp encryption overview. technical security whitepaper. https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf, 2016. [accedido 02 Febrero 2021].

[16] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schl .ª ffer. Ascon v1.2, ronda 1 del proyecto nist lightweight cryptography. https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/round-1/spec-doc/ascon-spec.pdf, 2019. [accedido 15 Marzo 2021].

[17] Manual de usuario de la aplicación ibks config tool. https://accent-systems.com/es/support/knowledge/ibks-config-tool-user-manual/, 2020. [accedido 18 Febrero 2021].

[18] Albert S. Huang and Larry Rudolph. *Bluetooth Essentials for Programmers*. Cambridge University Press, 2007.

[19] Thomaz Oliveira, J. C. López-Hernández, H. Hisil, Armando Faz-Hernández, and F. Rodríguez-Henríquez. How to (pre-)compute a ladder - improving the performance of x25519 and x448. In *SAC*, 2017.

[20] Emil Lenngren. Optimized x25519 for arm cortex-m4 microcontrollers. https://github.com/Emill/X25519-Cortex-M4.4, 2018. [accedido 20 Abril 2021].

[21] G. Ewing, RW Bradshaw, and DS Seljebotn et al. S. Behnel. Cython c-extensions for python. https://cython.readthedocs.io/en/latest/src/quickstart/overview.html, 2020. [accedido 01 Abril 2021].

Apéndice A

Instalación de Raspberry Pi OS

Existen numerosas imágenes del sistema operativo Raspberry Pi OS (anteriormente llamado Raspbian) que se pueden utilizar con la Raspberry Pi 4. La imagen elegida se puede descargar de la página oficial de Raspberry Pi (se recomienda descargar la versión más reciente):

```
http://www.raspberrypi.org/downloads/
```

La computadora que se este usando debe contar con un lector de tarjetas SD para formatear y escribir el SO (Sistema Operativo). Para grabar el SO en una tarjeta micro SD de preferencia de 16 GB o 32 GB (de mayor capacidad podría causar problemas de instalación), se debe formatear la tarjeta en un formato Fat32.

Después de tener formateada la tarjeta se podrá instalar el sistema operativo Raspberry Pi OS, en este caso se utiliza el software de asistencia Raspberry Pi Imager, siendo este una las formas más rápidas y fáciles de instalar el Raspberry Pi OS y otros sistemas operativos en una micro SD, debido a que deja la tarjeta lista para ser utilizada.

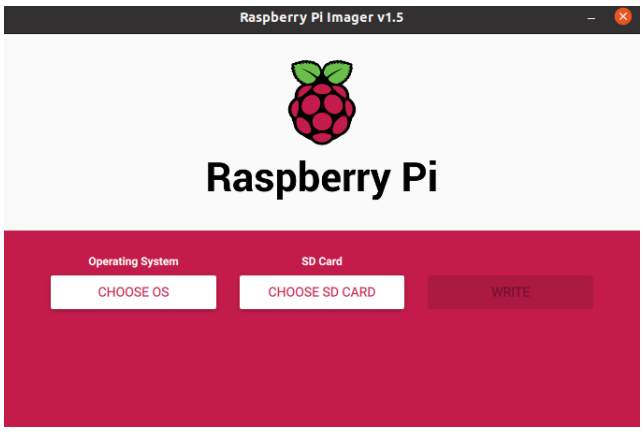


Figura A.1: Raspberry Pi Imager.

Para descargar e instalar Raspberry Pi Imager, se coloca la tarjeta micro SD en

su respectivo adaptador y después en el lector SD de la computadora, se ejecuta el Raspberry Pi Imager como se muestra en la figura A.1.

Posteriormente se elige la versión de Raspberry Pi OS recomendada por Raspberry Pi Imager y la tarjeta SD en donde se instalará, con esto se habilita el botón *WRITE* y se procede a escribir el sistema operativo. Una vez instalado el sistema operativo en la tarjeta, se coloca en la ranura correspondiente del Raspberry Pi antes de encenderla.

Configuración inicial de la Raspberry Pi

Para encender el dispositivo basta con conectarlo a una fuente de alimentación. La configuración inicial se puede realizar de manera gráfica o a través de la terminal usando el comando `raspi-config`, aunque se pueden configurar muchos aspectos del sistema, a continuación solo se muestran los pasos para configurar el lenguaje, el teclado y la habilitación del protocolo SSH.

1. El primer paso es actualizar el sistema con los cambios más recientes. Al lanzar los siguientes comandos con el prefijo `sudo` pedirá la contraseña, se introduce la contraseña por defecto `raspberry`. Se espera a que finalice el primer proceso para proceder con el segundo.

   ```
   sudo apt-get update
   sudo apt-get upgrade
   ```

   Este proceso puede durar varios minutos. Se recomienda ejecutar estos comando de vez en cuando para mantener el sistema actualizado.

2. Una vez finalizado el proceso anterior, se lanza la utilidad de configuración de Raspberry Pi con el comando siguiente:

   ```
   sudo raspi-config
   ```

Al ejecutar el comando aparece la pantalla de configuración mostrada en la figura A.2.

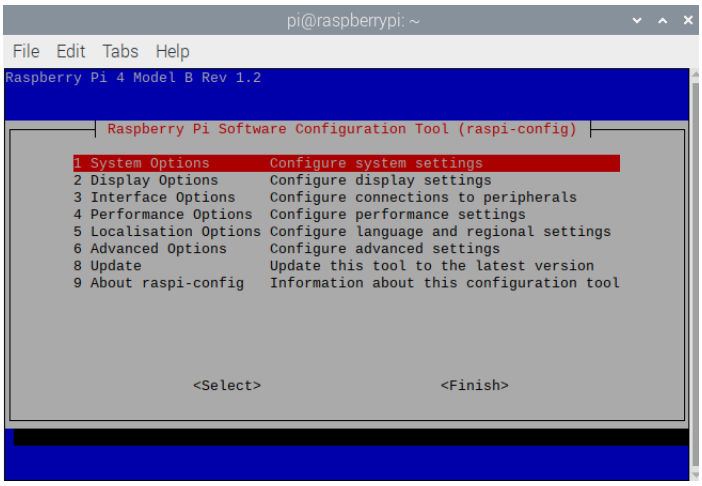


Figura A.2: Herramienta de configuración de Raspberry Pi.

3. En las opciones del sistema que se muestran en la figura A.3 se puede cambiar el nombre del *host* y la contraseña si se desea, por defecto aparece "`raspberry`", como se observa en la figura A.4.

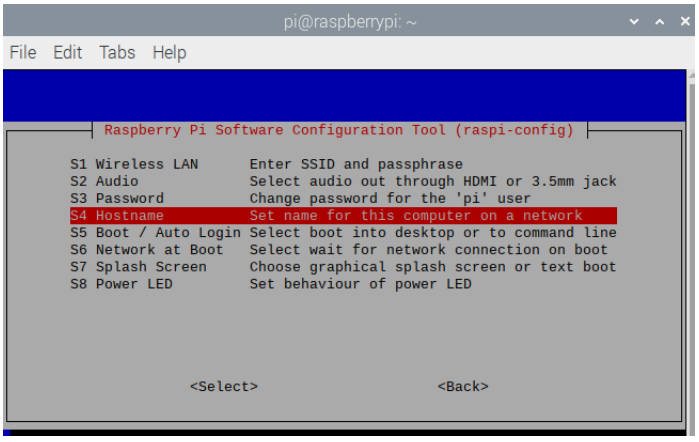


Figura A.3: Configuración de opciones del sistema.

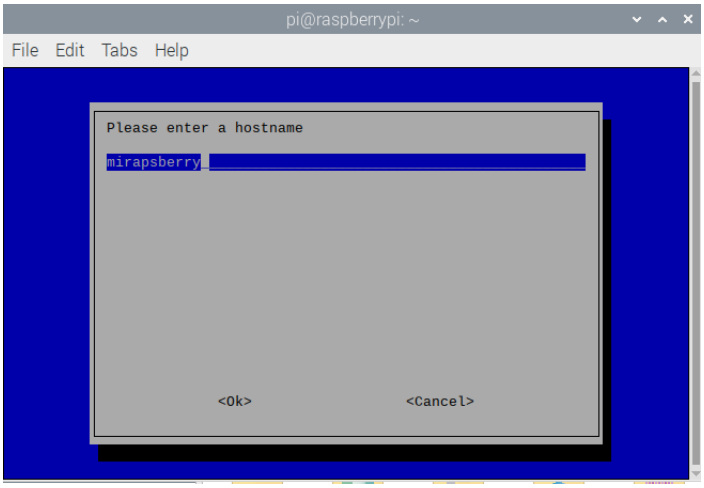


Figura A.4: Configuración de *hostname*.

4. Para configurar el idioma, localización, zona horaria y teclado, se ingresa a opciones de localización como se muestra en la figura A.5.

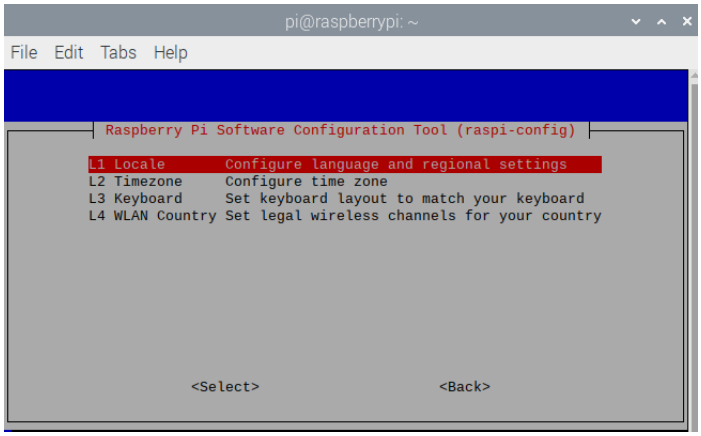


Figura A.5: Configuración de localización

- Localización: Cuando aparezca el menú como se ve en la figura A.6, se desplaza con las flechas del teclado y se selecciona es_ES.UTF-8 UTF-8 (español de España) y es_ES como el idioma por defecto del sistema.
- Zona horaria: Se selecciona el área geográfica (Continente) y capital de estado para ajustar la zona horaria.
- Teclado: Se elige el modelo de teclado en la lista. Para la mayoría de los casos se selecciona Generic 105-key (Intl) PC y Spanish.

En la sección de opciones de interfaces se puede habilitar el protocolo SSH como se muestra en la figura A.7.

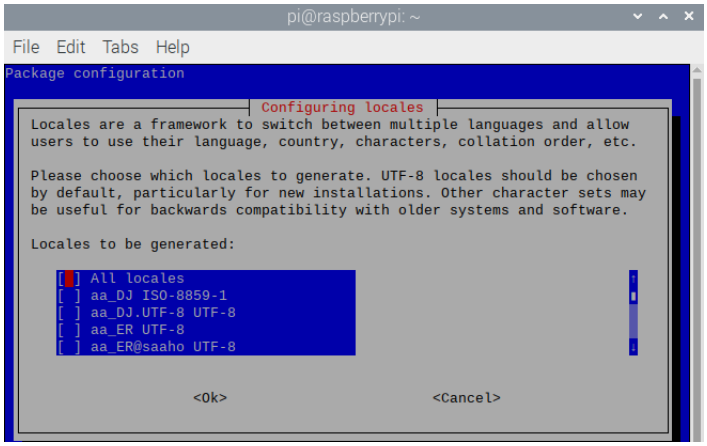


Figura A.6: Configuración de localización.

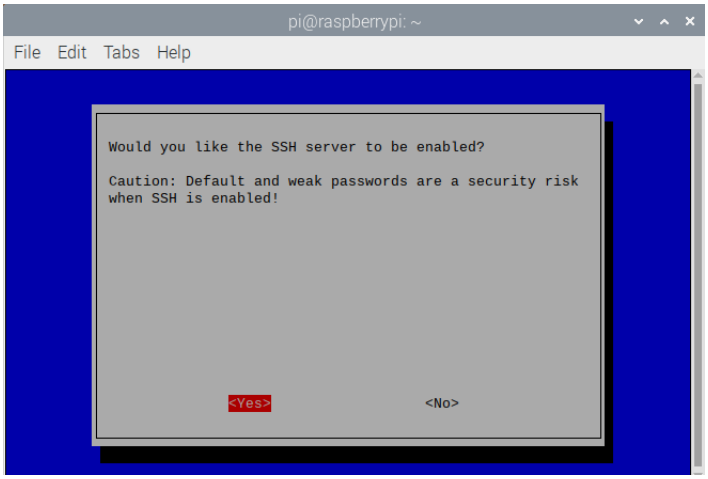


Figura A.7: Habilitación de SSH.

5. En caso de que no se haya habilitado el SSH en el paso anterior, se puede hacer después mediante los comandos siguientes:

```
sudo systemctl enable ssh
sudo systemctl start ssh
```

Una vez que se han hecho los cambios pertinentes, se da finalizar a la configuración y se reinicia la Raspberry Pi.

Apéndice B

Conexión a la Raspberry Pi

Para trabajar con la Raspberry Pi se optó por usar la conexión remota SSH a través de Wifi ó Ethernet, aunque existen otras opciones más directas de utilizar la tarjeta, como conectar periféricos de E/S como un monitor, teclado y ratón. El acceso con SSH utiliza la dirección IP de la Raspberry Pi, para ello se debe tener la tarjeta conectada a la red, en este caso se conectará de manera inalámbrica. Para conectar la Raspberry Pi a la red local se puede hacer de manera gráfica o a través de la terminal con los pasos siguientes:

1. Primero se listan las redes disponibles con el comando:

   ```
   sudo iwlist wlan0 scan
   ```

   El resultado es una larga lista con todos los datos e información de las redes Wifi disponibles.

2. Una vez que se reconozca la red deseada y su contraseña, se edita el fichero: /etc/wpa_supplicant/wpa_supplicant.conf. Para ello, se utiliza el comando:

   ```
   sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
   ```

3. En este fichero, se añade lo siguiente al final del archivo, según los datos de la red a la que se conectará.

   ```
   network={
           ssid="nombre-de-tu-wifi"
           psk="password-de-tu-wifi"
           key_mgmt=WPA-PSK
   }
   ```

   Por lo que el fichero quedaría al final de la siguiente manera:

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=ES

network={
        ssid="nombre-de-wifi"
        psk="password-de-wifi"
        key_mgmt=WPA-PSK
}
```

4. Se reinicia la Raspberry Pi con el siguiente comando:

```
sudo reboot
```

5. Para comprobar la conexión y obtener la IP que facilita el *router* se usa el siguiente comando:

```
ifconfig wlan0
```

   Si todo ha salido correctamente se verá la IP asignada. A continuación se podrá establecer una conexión remota a través del comando `ssh` con la siguiente estructura.

```
ssh pi@[Dirección IP]
```

6. La primera vez que se establece la conexión SSH entre el Raspberry Pi y el dispositivo externo se obtiene una sugerencia de seguridad/autenticación. Una vez que se haya tomado nota de ello, se introduce "si" en la terminal para continuar. Por último, se ingresa la contraseña para conectarse a la línea de comandos de la Raspberry Pi.

Apéndice C

Instalación de BlueZ

En este Apéndice se describen los pasos necesarios para descargar, compilar, instalar y configurar el sistema Bluez en la Raspberry Pi. Antes de comenzar, se debe de comprobar que la Raspberry Pi tenga acceso a internet, ya sea a través de una conexión por cable o inalámbrica. En caso de que no se cuente con los periféricos necesarios para el acceso a la placa, se puede ingresar a una terminal de comandos en la Rapberry Pi con la herramienta SSH (ver Apéndice B).

Prerequisitos

Esta guía funciona en los modelos de tarjetas siguientes:

- Raspberry Pi 3B+
- Raspberry Pi 4B

Se deben ejecutar previamente los comandos a continuación:

```
sudo apt-get update
sudo apt-get upgrade
```

Los dos comandos garantizarán que la tarjeta tenga las últimas actualizaciones. Las nuevas versiones de Raspberry Pi OS vienen con BlueZ instalado, para comprobarlo se ingresa el comando siguiente:

```
dpkg --status bluez | grep '^Version:'
```

En caso de no obtener respuesta, se requerirá instalar el sistema Bluez, como se explica a continuación.

Como primer paso se deben instalar las dependencias necesarias para la ejecución de BlueZ.

```
sudo apt-get install -y libusb-dev libdbus-1-dev libglib2.0-dev
libudev-dev libical-dev libreadline-dev
```

Después de que se ejecuten los comandos anteriores, se debe observar la instalación de las dependencias sin ningún mensaje de error. Para instalar BlueZ, se descarga y compila la última versión de su código fuente. Se puede instalar desde un paquete precompilado en el repositorio de Raspberry Pi OS, sin embargo, es casi seguro que la versión en el repositorio esté desactualizada. Si se desean utilizar funciones de Bluetooth de baja energía, se debe ejecutar la versión de BlueZ más reciente para obtener las últimas correcciones de errores y funciones BBE. La compilación de BlueZ desde la fuente asegurará que se tenga el último lanzamiento. Se accede a la página oficial de descarga de BlueZ (http://www.bluez.org/download/) y se copia el enlace a la última versión de la fuente. Por ejemplo, en el momento de redactar esta guía, la última versión de BlueZ es la 5.54 y se puede descargar desde el enlace:

```
http://www.kernel.org/pub/linux/bluetooth/bluez-5.54.tar.xz
```

Después de conectarse a una terminal en la Raspberry Pi, se ejecuta el siguiente comando para descargar y abrir el archivo fuente de BlueZ, se verifica la versión BlueZ descargada y se hacen los cambios pertinentes en la instalación.

```
cd ~
wget http://www.kernel.org/pub/linux/bluetooth/bluez-5.54.tar.xz
tar xvf bluez-5.54.tar.xz
cd bluez-5.54/
```

El comando anterior descomprime la fuente BlueZ en una nueva carpeta, como `bluez-5.54`.

Compilar e instalar BlueZ

Para compilar BlueZ, se usan los comandos estándar `configure`, `make` y `sudo make install`. Desde dentro del directorio de origen de BlueZ, se ejecuta el *script* de configuración de la manera siguiente:

```
./configure --enable-library
make
sudo make install
```

Configurar servicio BlueZ

Después de instalar BlueZ desde la fuente, se debe habilitar el servicio `bluetoothd`. Este servicio se comunica con una parte de BlueZ en el kernel y expone los dispositivos bluetooth a los programas del usuario. El servicio BlueZ debe estar ejecutándose. Con Raspberry Pi OS Jessie, el servicio BlueZ se ejecuta con `systemd`. Systemd es un servicio que controla otros procesos en la Raspberry Pi, como el sistema BlueZ.

Primero se verifica que el servicio BlueZ esté instalado y en buen estado ejecutando el comando:

```
systemctl status bluetooth
```

Probablemente se observe que el servicio está cargado pero no activo como se muestra en el ejemplo siguiente.

```
bluetooth.service - Bluetooth service
Loaded: loaded (/lib/systemd/system/bluetooth.service; disabled)
Active: inactive (dead)
Docs: man:bluetoothd(8)
```

Se puede activar manualmente el servicio con el comando:

```
sudo systemctl start bluetooth
```

Después de ejecutar el comando anterior, el `status`del servicio debería verse en modo activo ("`Running`") como se observa a continuación.

```
bluetooth.service - Bluetooth service
Loaded: loaded (/lib/systemd/system/bluetooth.service;
enabled; vendor preset
Active: active (running) since
Fri 2021-02-19 18:11:42 CST; 11min ago
Docs: man:bluetoothd(8)
Main PID: 1391 (bluetoothd)
Status: "Running"
Tasks: 1 (limit: 4915)
CGroup: /system.slice/bluetooth.service
           1391 /usr/lib/bluetooth/bluetooth --experimental
```

Para habilitar las funciones de Bluetooth de baja energía en el sistema BlueZ se debe modificar la configuración agregando la bandera `--experimental`. Estas son *APIs* especiales que permiten que BlueZ interactúe con dispositivos Bluetooth de baja energía, sin embargo, todavía están en desarrollo y tienen una bandera experimental que debe habilitarse primero. Para habilitar las funciones experimentales de BlueZ, se edita la configuración del servicio BlueZ ejecutando el comando:

```
sudo nano /lib/systemd/system/bluetooth.service
```

Después de agregar la bandera `--experimental` al archivo de configuración, se guardan los cambios y se reinicia el servicio.

Una vez realizadas la configuraciones necesarias, se está listo para comenzar a usar las herramientas de BlueZ como `bluetoothctl` y `hcitool` .

La herramienta `bluetoothctl` es el comando principal para configurar dispositivos Bluetooth en Linux, `bluetoothctl` no es parte de `systemd`, más bien es un simple conjunto de opciones para configurar dispositivos Bluetooth. Mientras que la mayoría de otros comandos modifican el comando básico con opciones, la ejecución de `bluetoothctl` inicia su propio entorno para ingresar opciones.

Hcitool hace uso de la interfaz del controlador de `host` (en una computadora) para comunicarse y leer/escribir cambios en los dispositivos BBE. Por lo tanto, `hcitool` es útil para encontrar el dispositivo BBE que se anuncia y luego cambiar los valores después de la conexión. Los datos solo se pueden cambiar si se conoce el servicio y la característica de la que provienen los datos. Para conocer los servicios y características relevantes, se puede utilizar la herramienta `gatttool`. Si no se le da algún comando, o si se usa la opción `-h`, `hcitool` imprime todas sus opciones de uso.

Comprobación de instalación

El adaptador Bluetooth integrado del Raspberry Pi se llama `hci0`. Se puede comprobar que funciona correctamente ingresando el comando `hciconfig`, se obtendrá una respuesta como el ejemplo siguiente:

```
hci0: Type: BR/EDR  Bus: UART
BD Address: DC:A6:32:A4:8B:E4  ACL MTU: 1021:8  SCO
MTU: 64:1
UP RUNNING
RX bytes:1987 acl:0 sco:0 events:91 errors:0
TX bytes:1647 acl:0 sco:0 commands:57 errors:0
```

Si por alguna razón se muestra como `DOWN`, se puede habilitar con el comando que sigue:

```
sudo hciconfig hci0 up
```

Con el comando `hciconfig`, se verifica de nuevo el estado del adaptador, en caso de que se muestre que el servicio esta corriendo pero con las banderas `PSCAN` e `ISCAN` se ingresan los comandos siguientes para configurar el modo BBE.

```
sudo hciconfig hci0 leadv 3
sudo hciconfig hci0 noscan
```

El comando `leadv` activa el modo de anuncio de baja energía y `noscan` configura el modo de anuncio no dirigido. Para iniciar la búsqueda de los dispositivos BBE se utiliza el comando siguiente:

```
sudo hcitool lescan
```

Se ingresa `Ctrl-C` para detener la búsqueda de dispositivos. Los doce dígitos hexadecimales (`B0:B4:48:ED:44:C3` por ejemplo) son la dirección Bluetooth del dispositivo. Se necesita conocer esta dirección cuando se realicen conexiones Bluetooth de baja energía.

Conexión con Gatttool

Gatttool es una herramienta del sistema BlueZ. Puede conectar, leer, escribir y escuchar fácilmente las notificaciones de los dispositivos Bluetooth de baja energía. Para conectar un dispositivo con BBE, como una computadora o una Raspberry Pi, se ingresa el comando siguiente:

```
gatttool -I -b [dirección Bluetooth]
```

La dirección Bluetooth es el valor informado por `hcitool`. La bandera `-I` indica que se usa un modo interactivo. Posteriormente Gatttool habilitará la terminal para poder escribir el comando `connect`, después de un momento de espera se mostrará el mensaje de "conexión exitosa" como se ve en el ejemplo siguiente.

```
pi@raspberrypi: ~ $ gatttool -I -b
 B0:B4:48:ED:44:C3
 [B0:B4:48:ED:44:C3] [LE]> connect
 Intentando conectarse a B0:B4:48:ED:44:C3
 Conexión exitosa
 [B0:B4:48:ED:44:C3] [LE]>
```

A continuación, se pueden utilizar los comandos de la herramienta Gatttool. El comando siguiente es uno de los principales.

```
Primary
```

Este comando numera los "servicios" disponibles, que son grupos que contienen "características", según la arquitectura BBE explicada en el capítulo 2, estos son elementos de datos que se pueden leer o escribir en el dispositivo. El comando siguiente sirve para visualizar las características de los servicios del dispositivo BBE.

```
characteristics
```

Después de averiguar los servicios y las características del dispositivo BBE, se necesita conocer los identificadores mediante los cuales se pueden leer o escribir datos usando el comando siguiente:

```
char-desc
```

Después de encontrar el identificador deseado, se leen los datos usando el comando:

```
char-read-hnd <handle>
```

Para escribir en un identificador específico, se necesita saber cuál es un identificador de escritura. Para esto, se opta por un método *hit and try*, es decir, se intenta leer todos los identificadores uno por uno hasta que se encuentre un error de lectura. Un error de lectura significa que el identificador específico no es un identificador de escritura (los identificadores de escritura no se pueden leer). Alternativamente, aplicaciones como *nrf connect*[1] pueden descubrir automáticamente los identificadores de escritura. Entonces, por ejemplo, para leer el nombre del dispositivo, se puede usar el comando `char-read-uuid`, dándole el ID de la característica "Nombre" del dispositivo BBE como se muestra a continuación:

```
[B0:B4:48:ED:44:C3] [LE]> char-read-uuid
00002a00-0000-1000-8000-00805f9b34fb
identificador: 0x0003
valor: 53 65 6e 73 6f 72 54 61 67 20 32 2e 30
```

Después de conectarse, se escribe un valor aleatorio para las diferentes características. En la mayoría de los casos, la escritura de valores aleatorios no funcionará como se esperaba. Para escribir los valores correctos en el identificador, se descifra el protocolo de datos, que se puede encontrar usando herramientas de rastreo como *Wireshark*[2] y *Ubertooth*[3]. Después de descifrar el protocolo de datos, se escriben los valores en el identificador usando el comando:

```
char-write-req <handle> <value>
```

Si el comando `char-write-req` informa un error, se puede usar de igual manera el comando siguiente:

```
char-write-cmd
```

[1] https://www.nordicsemi.com/
[2] https://www.wireshark.org
[3] http://ubertooth.sourceforge.net/