Deusto
Facultad de Ingeniería
Universidad de Deusto
Ingeniaritza Fakultatea
Deustuko Unibertsitatea

Grado en Ingeniería en Tecnologías de Telecomunicación

Telekomunikazio Teknologien Ingeniaritzako Gradua

Proyecto fin de grado
Gradu amaierako proiektua

Desarrollo e implementacion de un nuevo esquema de abstraccion que garantice los conceptos fundamentales de la seguridad de la informacion en entornos heterogéneos interoperables

Iñigo Alonso Gonzalez

Director: Pablo García Bringas

Bilbao, septiembre de 2014

Resumen

Internet de las Cosas es un concepto que define la incorporación de inteligencia y conexión a los objetos del mundo real. A día de hoy, uno de los mayores obstáculos que deben superarse en este campo es la falta de mecanismos de seguridad y de privacidad en su comunicación. Dado que las diferentes herramientas de seguridad de la información actuales no están diseñadas para operar en entornos heterogéneos y de recursos limitados, es necesario desarrollar un esquema de abstracción capaz de garantizar los conceptos de confidencialidad, integridad, disponibilidad, autenticidad y no repudio en la interacción entre dispositivos ubicuos. Para ello, se ha desarrollado un sistema capaz de dar servicio a software de terceros para habilitar la comunicación transparente y segura con otras entidades. En concreto, los elementos que componen este sistema son: un supernodo encargado de la infraestructura de red que interconecta de forma segura los diferentes nodos del sistema además de analizar el comportamiento de los nodos de la red y de gestionar las diferentes relaciones de confianza generadas por los supernodos que componen dicha red. Por otro lado, un servicio capaz de establecer los pasos necesarios para la comunicación segura con un supernodo. Por último, una librería controlador que permite, a software de terceros, comunicarse con el servicio instalado para transmitir la información deseada. Los resultados han demostrado que gracias al sistema desarrollado, es posible establecer una infraestructura de red segura compuesta por nodos con capacidades heterogéneas y recursos limitados.

Descriptores

Seguridad, Internet of Things, comunicaciones, privacidad.

Índice

Índice de ilustraciones

Índice de tablas

1. INTRODUCCIÓN

El Internet de las Cosas o Internet of Things (IoT, por sus siglas en inglés) es un concepto que define la incorporación de inteligencia y conexión a los objetos del mundo real. Este concepto está transformando la forma de hacer negocios, la organización del sector público y el día a día de millones de personas. No obstante, uno de los mayores obstáculos que deben superarse en este campo es el la falta de mecanismos de seguridad y privacidad.

Dado que los mecanismos de ciberseguridad actuales no están diseñados para operar en entornos heterogéneos y de recursos limitados, se propone implementar un nuevo esquema de abstracción que incluya los conceptos principales de seguridad en dichos entornos heterogéneos interoperables. En concreto, el objetivo es garantizar los conceptos de confidencialidad, integridad, disponibilidad, autenticidad y no repudio (ver anexo 1) en la interacción entre dispositivos ubicuos. Para ello, se propone diseñar un nuevo esquema de abstracción que tenga en cuenta todos estos conceptos fundamentales de la seguridad de la información.

Para garantizar la confidencialidad de las comunicaciones, es importante que éstas se realicen de forma cifrada. No obstante, debido a la naturaleza heterogénea del entorno en el que se pretende implantar este esquema, el cifrado de dichas comunicaciones deberá poder adaptarse a las diferentes capacidades de cómputo de los dispositivos que en él se encuentran. Del mismo modo, para garantizar la integridad de los datos, se deberá utilizar un servicio de confidencialidad que permita evitar que cualquier otro dispositivo no autorizado pueda acceder o alterar la información transmitida. Por otro lado, se garantizará la disponibilidad del sistema replicando los servidores e infraestructura necesaria para proporcionar este servicio. También, se implementará un sistema de certificados que garanticen la autenticidad tanto del emisor como del receptor. Al igual que con el cifrado, el mecanismo de autenticidad deberá poder adaptarse a las diferentes capacidades de cómputo de los dispositivos. Por último, para garantizar el no repudio se implementará un sistema de confianza que establecerá la comunicación entre una red de servidores verificados para que compartan el conocimiento que tienen a cerca de los dispositivos que se comunican a través de ellos.

Esta documentación expone, en primer lugar, un breve resumen sobre el estado de la técnica que ha permitido definir la motivación a la hora de realizar este proyecto. Del mismo modo,

muestra los motivos por los cuales existe una demanda de investigación y desarrollo de técnicas de securización para entornos heterogéneos y de recursos limitados. Para satisfacer esta demanda, se han establecido unos objetivos expuestos junto con el alcance más adelante en este proyecto. Una vez claros los objetivos y los límites del trabajo a realizar, esta memoria muestra con detalle las características del diseño del sistema desarrollado para cumplir dichos objetivos y permitir su posterior desarrollo. Este desarrollo también queda plasmado en esta memoria mostrando la arquitectura software utilizada durante la creación del prototipo posteriormente publicado [1]. A continuación, se muestran las pruebas realizadas a este prototipo y los consiguientes resultados que han validado el cumplimiento de los objetivos establecidos. Debido a que este proyecto únicamente constituye una primera aproximación a la creación de una solución que permita satisfacer la demanda de seguridad previamente mencionada, esta memoria recoge las múltiples líneas de trabajo futuro que deben seguirse para poder desarrollar un producto final que satisfaga esta demanda en un complejo entorno real. Finalmente, se muestra un capítulo de conclusiones en el que se resume de forma breve y concisa todo el conocimiento generado como consecuencia de la realización de este proyecto. Además, como en todo proyecto, se muestra la planificación establecida así como el presupuesto establecido para la realización del mismo.

2. ESTADO DE LA TÉCNICA

La electrónica vive tiempos de gran expansión y cambio. Funciones que décadas atrás requerían de un ordenador del tamaño de una habitación son hoy día realizadas con facilidad por simples dispositivos electrónicos que caben en la palma de la mano. La electrónica se está fragmentando para crear múltiples dispositivos que realizan tareas específicas. Como decía Mark Weiser, director científico del Xerox Palo Alto Research Center, este concepto de computación ubicua aboga por un futuro en el que la computación desaparecería de nuestra vista y formará parte integral de nuestra vida diaria resultando transparente para nosotros [2]. Es por ello por lo que al interconectar este tipo de dispositivos entre sí se crea una red de naturaleza heterogénea en la que destaca la gran variedad características y capacidades de cómputo de los dispositivos que la forman. Si proporcionamos a estas redes la posibilidad de comunicarse a través de internet, logramos lo que se denomina como Internet de las Cosas.

Internet de las Cosas o Internet of Things (IoT, por sus siglas en inglés) consiste en la integración de sensores y dispositivos en objetos cotidianos que quedan conectados a Internet a través de redes fijas e inalámbricas.

Dado su tamaño y coste, los sensores son fácilmente integrables en hogares, entornos de trabajo y lugares públicos. De esta manera, cualquier objeto es susceptible de ser conectado y «manifestarse» en la Red. Esto permite medir desde la temperatura de una habitación hasta el tráfico de taxis en una ciudad. A diario, cámaras de vigilancia velan por la seguridad en los edificios y los paneles del metro nos indican el tiempo que falta hasta la llegada del siguiente tren. Incluso en las multas de tráfico existe poca intervención humana. Esto está empezando a transformar la forma de hacer negocios, la organización del sector público y el día a día de millones de personas. En sectores como el sanitario, el agrícola, la logística o el de suministros ya está extendido el uso de este tipo de dispositivos.

A medida que la información y las personas están cada vez más conectadas, la tecnología se ha convertido en una herramienta principal de colaboración y toma de decisiones en el mundo. Dentro de la posibilidad de estar permanentemente conectado y localizable, está surgiendo una nueva generación de consumidores que da por hecho contar con acceso a internet y cualquier avance técnico que permita la movilidad.

Es por esta razón por la que se están empezando a plantear preocupaciones respecto a la privacidad y la seguridad de estos dispositivos. Estas preocupaciones vienen infundadas por el hecho de que en la actualidad no existen técnicas o herramientas que garanticen los conceptos

de confidencialidad, integridad, disponibilidad, autenticidad y no repudio propios de un entorno seguro en la interacción entre dispositivos ubicuos.

Actualmente existe una amplia variedad de protocolos criptográficos para garantizar la confidencialidad de los mensajes. En concreto: el cifrado de bloques para la confidencialidad Advanced Encryption Standard (AES)[3]; el algoritmo asimétrico para la firma digital y el transporte de claves Rivest-Shamir-Adelman (RSA)[4]; el algoritmo asimétrico de generación de claves Diffie-Hellman (DH)[5]; y los algoritmos d regeneración de hash seguros SHA-1 y SHA-256[6]. Este conjunto de algoritmos está siendo reemplazado por un grupo de algoritmos asimétricos conocidos como Criptografía de Curva Elíptica (del inglés: Elliptic curve cryptography, ECC).

Este grupo de algoritmos fueron diseñados para dispositivos con una alta disponibilidad de recursos, como por ejemplo, velocidad del procesador y memoria. La viabilidad de estas técnicas criptográficas en IoT no está clara y requiere un análisis más profundo para asegurar que los algoritmos pueden implementarse correctamente dada la limitada memoria y velocidad de procesado presente en los dispositivos que conforman el IoT.

Para el desarrollo inicial de un protocolo de IoT, los desarrolladores acuden a algoritmos como AES-GCM, que es un modo combinado que soporta autenticación y cifrado, y a algoritmos asimétricos como los ECC. A medida que los recursos comunes en los dispositivos del IoT se vuelven más claros, los investigadores han determinado que estos algoritmos no son óptimos [7] y que es necesario el avance en la investigación de nuevos algoritmos de cifrado más adecuados.

Esto ha dado como resultado un conflicto con los desarrolladores implementadores: cualquier técnica que sea fácil de implementar será fácil de romper por usuarios con más recursos computacionales ya que la asunción de que los atacantes contarán con las mismas limitaciones de recursos es incorrecta.

Para garantizar que los primeros desarrolladores tienen disponibles las características de seguridad necesarias, es esencial que los protocolos de IoT contemplen de forma obligatoria la implementación de mecanismos de seguridad de uso opcional. Esto proporcionará seguridad en todas las implementaciones de estos protocolos y la posibilidad de no hacer uso de ella cuando no sea necesaria. Aunque en una primera instancia, a posibilidad de no utilizar estos mecanismos de securización dé como resultado un primero uso de estos protocolos sin hacer uso de su versión segura, la experiencia lograda con casos similares en el pasado, como el uso del cifrado de conexión inalámbrica Web, permitirá concienciar a los usuarios de estos protocolos de que un sistema de cifrado débil es fácilmente descubierto y explotado.

Uno de los aspectos más difíciles de la seguridad criptográfica es la gestión de claves. Mientras que muchos protocolos de internet se han desplegado con una gestión manual de claves (por ejemplo claves pre-compartidas), la configuración manual de claves en el gran número de dispositivos presentes en el IoT no es favorable a la escalabilidad. Además del gran número de dispositivos, las limitadas interfaces de usuario de las que se dispone hace difícil la implementación de una seguridad significativa. Incluso si las claves de los dispositivos pueden

configurarse al inicio de la puesta en producción, es necesaria la resignación de claves automática una vez pasada la fase de despliegue.

La identificación de usuarios y dispositivos presenta un desafío significativo en el actual uso de Internet, y se verán exacerbadas por el gran número de dispositivos y las limitaciones previstas en las interfaces de usuario. Esto hace necesario el uso de mecanismos de securización que combinen técnicas tanto manuales como automáticas para el despliegue inicial de estos dispositivos. Ya existen protocolos que hacen uso de este tipo de mecanismos, en particular, los denominados “protocolos de emparejamiento” utilizados en tecnologías como Bluetooth.

En IoT se espera que muchos dispositivos no estén asociados únicamente a un usuario. Por ejemplo, una casa solo necesita una tostadora incluso si proporciona servicio a una familia de cuatro miembros. Por ello, surge la necesidad de crear identidades para grupos de personas de una forma nunca antes desarrollada.

El tema de la usabilidad también proporcionará un desafío significativo. Independientemente de la naturaleza del dispositivo, es esencial que la dificultad de la experiencia de uso del dispositivo no se vea incrementada. Del mismo modo, la dificultad de adaptación al protocolo por parte del desarrollado implementado no debe verse aumentada por la incorporación de mecanismos de securización. Este será un reto importante incluso para los aspectos de redes; proporcionar seguridad de uso sencillo se está convirtiendo en un tema de investigación.

Los problemas de privacidad también se están convirtiendo en un gran reto. La experiencia con la red eléctrica inteligente (smart grid en inglés) demuestra la importancia que supone no exponer el uso de electricidad asociada a una casa o negocio. IoT tiene el potencial para exponer datos importantes permitiendo la violación de la privacidad de sus usuarios.

Por otro lado, la migración de los sistemas de información centralizada a las aplicaciones basadas en Internet significa que las transacciones abarcan una amplia gama de dominios y organizaciones, no se puede confiar en la misma medida en todos ellos. Las inconsistencias en las relaciones de confianza actuales subrayan la necesidad de un sistema flexible, los sistemas de gestión de confianza de uso general que pueden navegar por estos dominios de confianza complejos.

Como respuesta a la insuficiencia de los mecanismos de autorización tradicionales se desarrolló el enfoque de Trust Management para la seguridad de sistemas distribuidos. Este enfoque, se define como "un enfoque unificado para la especificación y la interpretación de las políticas de seguridad, las credenciales, las relaciones que permiten la autorización directa de las acciones de seguridad críticos”. Los motores de gestión de confianza evitan la necesidad de resolver “identidades” en un conflicto de autorización. En cambio, expresan privilegios y restricciones en un lenguaje de programación. Esto permite una mayor flexibilidad y expresibilidad en sistemas de seguridad escalables. Otras ventajas del enfoque incluyen pruebas de que las transacciones solicitadas cumplen con las políticas locales y las arquitecturas del sistema que animan a los desarrolladores y administradores a considerar la política de una aplicación de seguridad con cuidado y especificarla explícitamente [8].

TSM (Trusted Service Manager) es un medio para establecer un canal de comunicación, contractual, y tecnológico de confianza entre las empresas que actualmente despliegan los sistemas de telefonía móvil (tanto dispositivos o terminales, como redes) [9]

Desde el punto de vista del consumidor, TSM opera en segundo plano [10], haciendo el rol de business-to-business. Estos servicios no son visibles para el usuario. El papel que desempañan en este ecosistema los TSM es el de proporcionar un canal de confianza entre las distintas entidades que participan, a nivel técnico, contractual y de comunicación.

Se han implementado varios sistemas automatizados de gestión de la confianza: de PolicyMaker, KeyNote y REFEREE. Un defecto común a todas estas soluciones es que se utilizan para identificar una forma estática de confianza, por lo general, a discreción del programador de aplicaciones (es decir, el programador inserta código para evaluar la confianza, a menudo en el inicio de una sesión). Sin embargo, la confianza puede cambiar con el tiempo. Si bien es cierto que la mayoría de los sistemas de gestión de la confianza se centran en los protocolos para establecer la confianza en un contexto particular, en la actualidad, no existen soluciones que aborden de forma dinámica este problema.

En definitiva, existen grandes desafíos en lo que a seguridad de IoT se refiere. Es esencial que los protocolos de IoT incluyan desde sus origines características de seguridad, incluso si estas superan las capacidades de dichos dispositivos en la actualidad. Además, dado el amplio rango de capacidades presentes en los dispositivos que conforman Internet of Things, es necesario que los protocolos desarrollados tengan en cuenta estas capacidades a la hora de diseñarse. Dentro de estos mecanismos deben contemplarse los conceptos fundamentales de la seguridad de la información; confidencialidad, autenticidad, integridad, disponibilidad y no repudio. La garantización de cada uno de ellos es fundamental para la adopción de estos protocolos.

.

3. JUSTIFICACIÓN

Más de mil millones de usuarios de todo el mundo utilizan Internet tanto en su vida laboral como en la social y gracias a la tecnología wireless se han ampliado las posibilidades de interacción con la Red a cualquier lugar, en cualquier momento. En este contexto, el Internet de las Cosas constituye un avance con gran impacto sobre la sociedad y los negocios.

Además, dado que se ha reducido drásticamente el tamaño, coste y consumo de energía del hardware que se encuentra en los dispositivos que constituyen el Internet de las Cosas, es posible fabricar dispositivos electrónicos diminutos a un coste muy reducido. Estos pequeños dispositivos, junto con la expansión de las redes de comunicación, permiten incorporar inteligencia y conexión a los objetos del mundo real y están transformando lo que era una red global de personas en una red global de «todas las cosas». El hecho de que Internet esté presente al mismo tiempo en todas partes permite que la adopción masiva de esta tecnología sea factible. Esto está empezando a transformar la forma de hacer negocios, la organización del sector público y el día a día de millones de personas.

Tal y como dice el Doctor Rolf H. Weber en su artículo Internet of Things – New security and privacy challenges “Con la aparición de la Internet of Things, es necesario desarrollar nuevos enfoques que aseguren la seguridad y privacidad en este entorno.”[11] o se muestra en la divulgación El Internet de las Cosas - En un mundo conectado de objetos inteligentes, “Uno de los mayores obstáculos que debe superar El Internet de las Cosas son las lagunas en cuanto a seguridad y privacidad se refiere”[12]. Con esto podemos concluir con que la seguridad en Internet of Things no se está tomando con la seriedad que debiera ya que los agentes involucrados en él son de extrema importancia.

Veo por ello apremia la necesidad de implementar un esquema de abstracción que garantice los conceptos fundamentales de la seguridad de la información en la Internet of Things. Esto proporcionará al IoT protección frente a ataques, integridad de la información transmitida, una alta disponibilidad del sistema y autenticidad de los dispositivos involucrados.

Por lo tanto, considero que este proyecto proporciona tanto un bien necesario para la sociedad como una oportunidad de negocio debido a que en el IoT se encuentran involucrados agentes diversos como empresas, personas o servicios cuyos intereses se verían comprometidos por la vulneración de la seguridad de la red.

4. OBJETIVOS Y ALCANCE

4.1 OBJETIVOS

Para la correcta implementación de este proyecto se han establecido los siguientes objetivos:

Objetivo principal:

Implementar un nuevo esquema de abstracción que garantice los conceptos fundamentales de la seguridad de la información en entornos heterogéneos interoperables.

Con el objetivo de dar cumplimiento a la misión fundamental de este proyecto, se plantean siete objetivos específicos que buscan resolver las carencias de seguridad anteriormente planteadas:

Objetivo específico 1:
Desarrollar un módulo de cifrado de comunicaciones que se adapte a las diferentes capacidades de cómputo de los dispositivos que se comuniquen dentro del sistema.

Objetivo específico 2:
Implementar un mecanismo que permita detectar alteraciones en la integridad de la comunicación.

Objetivo específico 3:
Desarrollar un mecanismo que permita la autenticación inequívoca de cada uno de los dispositivos del sistema.

Objetivo específico 4:
Desarrollar un mecanismo que registre toda acción acaecida dentro del sistema permitiendo un futuro procesado de la misma.

Objetivo específico 5:
Desarrollar un mecanismo que garantice la alta disponibilidad del sistema.

Objetivo específico 6:
Integrar todos los mecanismos de securización desarrollados en un sistema de comunicación.

Objetivo específico 7:
Desarrollar una herramienta que permita al software de terceros hacer uso del sistema desarrollado.

4.2 ALCANCE

El propósito general del software desarrollado es el de aportar una solución a la demanda de un sistema que permita la comunicación segura en un entorno heterogéneo interoperable. Para delimitar el alcance de este proyecto se han establecido los siguientes ámbitos.

Se va a desarrollar una API en Java que permitirá:
• Establecer la comunicación entre dos dispositivos.
• Transmitir a través de un canal seguro la información deseada.
• Finalizar la comunicación entre los dos dispositivos.
• Proveer todas las solicitudes de comunicación por parte de otros dispositivos.

También, se va a desarrollar un conjunto de programas Java permitirán:
• Establecer un canal de comunicación entre dos dispositivos de la red.
• Garantizar los conceptos fundamentales de la seguridad de la información durante el establecimiento del canal de comunicación y la transmisión de la información a través de este.
• Ofrecer múltiples alternativas de securización.

5. METODOLOGÍA

En el siguiente capítulo se muestra la metodología a seguir para cumplir los objetivos previamente enunciados. Esta metodología se compone de seis fases que se repiten de forma cíclica.

Con el fin de mostrar de manera clara y sencilla la visión global del alcance de este proyecto, se han representado las fases a seguir en el siguiente diagrama.

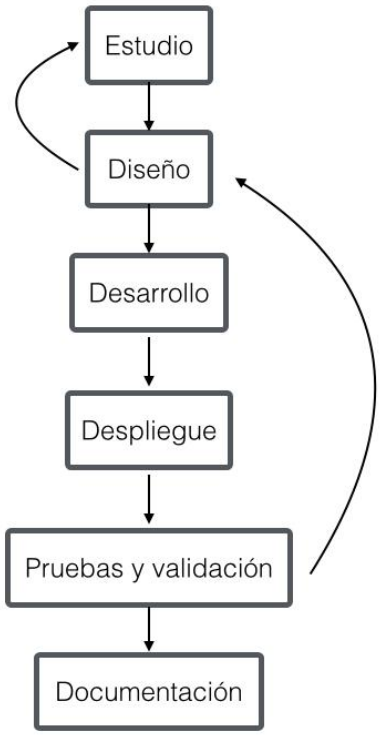

Ilustración 1 Diagrama de metodología

Fase 1:

Estudio inicial de tecnologías existentes.

Llevar a cabo un proceso de documentación y análisis sobre el estado del arte de las tecnologías que abarca el ámbito del proyecto a través de la consulta de publicaciones de la comunidad científica en dicho ámbito.

Fase 2:
Diseño.
Planificar con detalle los pasos a seguir y su modo de ejecución a la hora de desarrollar el proyecto. Si los conocimientos de los que se dispone no permiten superar esta fase se debe regresar a la fase de estudio.

Fase 3:
Desarrollo.
Desarrollar la solución software que permita obtener un prototipo del sistema diseñado.

Fase 4:
Despliegue.
Desplegar el sistema desarrollado en un entorno de desarrollo.

Fase 5:
Pruebas y validación.
Validar el correcto funcionamiento del sistema desarrollado haciendo uso de los objetivos establecidos para verificar su validez. El resultado de las pruebas realizadas debe proporcionar unos resultados que satisfagan los objetivos establecidos para pasar a la siguiente fase, en caso contrario de deberá volver a la fase de desarrollo.

Fase 6:
Documentación.
Plasmar el conocimiento obtenido durante la elaboración del proyecto en una memoria para que futuros investigadores partan de esta base.

6. DISEÑO DEL SISTEMA

Para garantizar todos los criterios necesarios para establecer una red segura compuesta por diferentes elementos con recursos limitados y capacidades diversas, es necesario diseñar una capa de abstracción que permita tener control absoluto en cada una de las fases de cualquier comunicación que en ella se realicen.

Este capítulo se centra en el diseño de cada uno de los elementos que componen el sistema desarrollado. En primer lugar se explicará la infraestructura específicamente diseñada para permitir que todos los integrantes de la red se comuniquen entre sí de forma transparente manteniendo el control de todas las acciones que se realizan en la red. También se enumeran los diferentes elementos que componen esta infraestructura mostrando cuál es su papel en el mantenimiento de ésta.

Una vez especificada la infraestructura sobre la que opera el sistema, se analizará en detalle las diferentes medidas adoptadas para garantizar los estándares de confidencialidad, integridad disponibilidad, autenticidad y no repudio establecidos en el alcance del proyecto. Por último se muestra el protocolo de comunicación diseñado para permitir la puesta en práctica de las medidas de seguridad desarrolladas para la comunicación dentro de la infraestructura desarrollada.

6.1 LA INFRAESTRUCTURA DE RED

Con el fin de obtener el máximo control sobre las comunicaciones del sistema, se ha diseñado una infraestructura de red que permite que todos los elementos que la componen puedan comunicarse entre sí de forma segura. Esta infraestructura se ha diseñado para ejecutarse en la capa de aplicación de la pila de protocolos OSI. Por otro lado, aunque se ha diseñado con vistas a realizar grandes despliegues, es igualmente posible desplegarla dentro de una red de área local sin necesidad de realizar ninguna clase de modificación en su diseño.

La infraestructura está compuesta por Nodos y Supernodos (Ver ilustración 2). Definimos Nodo como un integrante desarrollado por una entidad externa a la que la red proporciona la capacidad de comunicarse con otros semejantes dentro de sí misma. Por otro lado, un Supernodo es un integrante cuyo único fin es dar servicio a los Nodos que componen la red ofreciéndoles la capacidad de establecer una comunicación entre ellos a través de sí mismo. La capacidad de colocarse como intermediario en una comunicación permite a los Supernodos analizar todo el

tráfico trasmitido a través del sistema y ofrecer una tercera parte de confianza durante la comunicación entre dos nodos desconocidos entre sí.

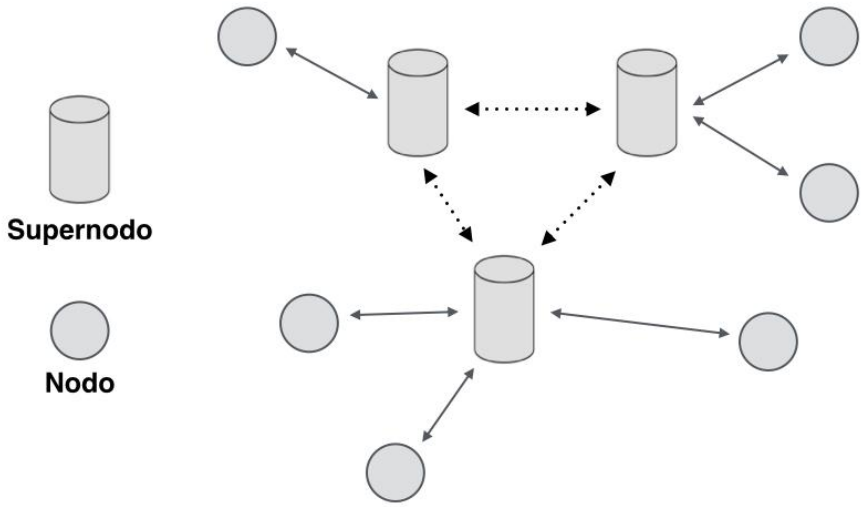

Ilustración 2 Esquema general de la red

A continuación se muestran en detalle las características de cada uno de ellos:

6.1.1 Nodo

Un Nodo es un integrante de la red cuyo hardware y software ha sido desarrollado por una entidad externa. El Nodo hace un uso convencional de la red conectándose a otros Nodos o dando servicio a estos, ejerciendo de forma simultánea la función de cliente y de servidor.

6.1.1.1 Características Hardware de un Nodo

Dentro de la categoría Nodo podemos encontrar una gran cantidad de hardware diferente como por ejemplo ordenadores convencionales, teléfonos inteligentes, referidos a partir de ahora como smartphones por su nombre en inglés, placas de capacidades reducidas y sensores con acceso a internet. Esta amplia gama de capacidades de cómputo convierte al Nodo el elemento cuya naturaleza es la más heterogénea de la red.

6.1.1.2 Características Software de un Nodo

En cuanto al software se refiere, la mayor parte de los Nodos están compuestos por dos módulos (Ver ilustración 4), un programa denominado Servicios de Nodo, referido a partir de ahora como Node Services por su traducción al inglés, y una librería adoptada por en el software de terceros denominada Node Library. Del mismo modo, existen Nodos que por su naturaleza solo están

formados por un módulo (Ver ilustración 3), no obstante, este módulo no es más que la unificación de los dos anteriormente mencionados por lo que, de cara a los otros elementos de la red su funcionamiento es idéntico al de los Nodos compuestos por dos módulos. Las razones

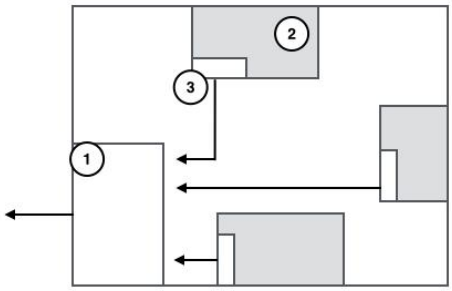

Ilustración 4 Estructura interna de un Nodo de dos módulos

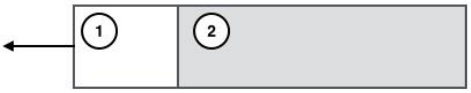

Ilustración 3 Estructura interna de un Nodo de un módulo

1. Node Services
2. Software de terceros
3. Node Library

1. Librería
2. Software de terceros

por las que un Nodo debería estar compuesto por un único módulo son entre otras, la limitación por parte del hardware a la hora de ejecutar múltiples programas en paralelo, los escasos recursos de los que pueda disponer dicho Nodo o restricciones por parte del sistema operativo a la hora de comunicarse dos programas diferentes.

Los módulos de los nodos convencionales, aquellos compuestos por dos módulos, están diseñados para trabajar completamente desacoplados, lo que permite incluso que, a la hora de diseñarlos, sean implementados en dos módulos de hardware independientes.. Sin embargo, en este tipo de diseños es especialmente necesario proporcionar un canal seguro de comunicación entre ambos módulos ya que el diseño inicial de este sistema no contempla la securización de la comunicación entre dichos módulos.

A continuación se analiza en detalle la función de ambos módulos:

6.1.1.2.1 Node Services

Es un programa cuya función es hacer de intermediario entre la Node Library y el Supernodo. Con el fin de poder ofrecer este servicio a las librerías Node Library del Nodo, este programa está continuamente ejecutándose a la espera de peticiones de comunicación, tanto por parte de las librerías como por parte de Supernodos de la red, pudiendo ser esta última función deshabilitada a través de la configuración del Node Services.

Gracias a este diseño es posible no replicar recursos de forma innecesaria y establecer una identidad única para el Nodo y común para todas sus aplicaciones (Ver ilustración 6). Del mismo modo, permite no replicar código durante su desarrollo.. Además, el módulo Node Services

asume la mayor parte de la carga computacional necesaria para establecer una conexión con el Supernodo, securizarla y transmitir información. De esta forma, es posible proporcionar una librería Node Library ligera.

Se han establecido los puertos TCP 1170 y 1171 para la escucha de peticiones por parte de Supernodos externos (1170) y software de terceros que utilicen la librería Node Library (1171).

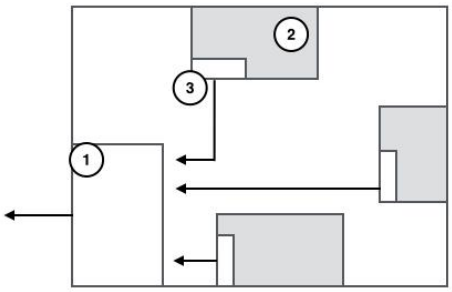

Ilustración 6 Node Services distribuido

1. Node Services
2. Software de terceros
3. Node Library

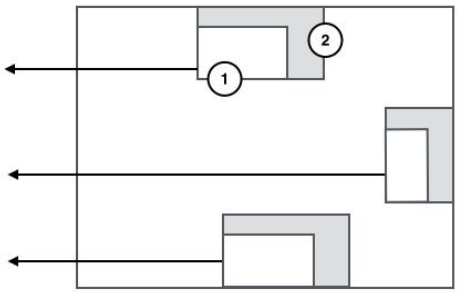

Ilustración 5 Node Services no distribuido

1. Node Services
2. Software de terceros

6.1.1.2.1.1 Recursos de Node Services

Todos los recursos necesarios para que el Nodo desempeñe su función correctamente se almacenan en una carpeta debidamente protegida, denominada data, que únicamente es accesible por el programa Node Services y el usuario administrador del sistema. Esto se debe a que la exposición de dichos recursos a terceros comprometería la identidad y seguridad del Nodo. En caso de un ataque y que dichos recursos quedarán expuestos ante una entidad maliciosa, únicamente se vería comprometida la seguridad del Nodo poseedor de los mismos, dejando así intacta la seguridad los demás elementos del sistema. Estos recursos están compuestos por un archivo xml de configuración, el cual posee la clave de autenticación del Nodo, un almacén de llaves en el que se almacenarán los certificados propios del Nodo, referido a partir de ahora como keystore por su nombre en inglés, y un archivo de extensión .pem que almacenará la clave pública de la entidad certificadora del sistema. De los recursos mencionados, el único recurso presente en todos los Nodos es el archivo de configuración, lo demás varían en función de las características del Nodo analizadas junto a la composición de estos archivos más adelante en esta documentación.

6.1.1.2.1.2 Configuración

A continuación se muestran los aspectos configurables del programa Node Services. La configuración representada en este archivo, determina la forma de comportarse del módulo Node Services y, por lo tanto, la del software de terceros instalado dentro del Nodo que haga uso de este sistema. Para que el módulo Node Services asimile de forma correcta esta configuración ha de almacenarse como un archivo de formato xml con el nombre config dentro de la carpeta data mencionada anteriormente.

Son varias las razones por las que esta configuración debe asignarse a través de un archivo y no a través de una interfaz de usuario. En primer lugar, uno de los objetivos del diseño de este módulo es hacerlo lo más ligero posible y una interfaz de usuario añadiría una gran cantidad de código y consumiría recursos únicamente para este fin.Por otro lado, al tratarse de un sistema orientado al uso por parte de personal cualificado es innecesario dotar de interfaz de usuario al panel de configuración debido a la facilidad con la que un desarrollador cualificado es capaz de configurarlo por medio de un archivo de configuración. Finalmente, dado que muchos Nodos carecen periféricos capaces de representar interfaces de usuario configurar de esta forma sería inviable.

La elección del formato xml se debe a que se trata de un formato de definición de contenido estructurado estándar, fácil de procesar y muy extendido.
Los aspectos configurables son los siguientes:

Security

class=""
level=“”
Explicación:
Especifican la clase y el nivel de seguridad bajo los que actuará el Nodo. Las diferentes clases y niveles de seguridad vienen especificados en el capítulo x TITULO
Valor por defecto:
class="B"
level=“0"
Repeticiones
Este elemento de la configuración puede especificarse de 0 a 1 veces. Aunque no es obligatorio, es altamente recomendable especificar estos valores.

Sessions

external=“”
internal=“”
Explicación:
Específica el número de sesiones que el Nodo podrá mantener activas tanto solicitadas por un Supernodo externo como por una librería Node Library interna. Ambos valores son

independientes por lo que en el caso de indicar tres sesiones externas y 3 internas el Nodo sería capaz de tratar un total de 6 sesiones al mismo tiempo.

Valor por defecto:
external="3"
internal="3"
Repeticiones
Este elemento de la configuración puede especificarse de 0 a 1 veces.

Keystore

password=" "
Explicación:
Especifica el valor de la contraseña del almacén de llaves en el que se almacenan los certificados del Nodo.
Valor por defecto:
No existe
Repeticiones
Este elemento de la configuración puede especificarse de 0 a 1 veces. Para todos aquellos casos en los que el Nodo vaya a utilizar certificados es obligatorio especificar este valor.

Certificate

alias=" "
password=" "
signed=" "
Explicación:
Elemento contenido dentro del parámetro de configuración Keystore. Especifica el alias, la contraseña y si está firmado por la Entidad Certificadora del sistema de un certificado almacenado en la keystore.
Valor por defecto:
No existe
Repeticiones
Este elemento de la configuración puede especificarse de 0 a n veces. Para todos aquellos casos en los que el Nodo vaya a utilizar certificados es obligatorio especificar este valor al menos una vez.

Supernode-list

fixed=" "
Explicación:
Especifica si la lista de Supernodos que se definen en su interior debe ser la única lista de Supernodos a los que el Nodo deberá conectarse. Esta configuración sirve para cuando el Nodo debe operar dentro de una red interna como puede ser de área empresarial o gubernamental. En caso de especificarse como "false" el Nodo será capaz de conectarse a cualquier Supernodo de la red de Supernodos.
Valor por defecto:
No existe

Repeticiones
Este elemento de la configuración puede especificarse de 0 a 1 veces.

Supernode

ip=""
Explicación:
Elemento contenido dentro del parámetro de configuración Supernode-list. Especifica toda la información acerca del Supernodo al que deberá conectarse el Nodo configurado. En esta versión solamente se especificará la dirección IP del Supernodo.
Valor por defecto:
No existe
Repeticiones
Este elemento de la configuración puede especificarse de 0 a n veces.

Ejemplo de configuración en formato xml:

```
<?xml version="1.0" encoding="utf-8"?>

<Profile>

<Security class="A" level="1" />

<Sessions internal="3" external="5" />

<Keystore password="demopassword123">

        <Certificate alias="demoalias" password="demopassword123" signed="true"/>

        <Certificate alias="demoalias" password="demopassword123" signed="false"/>

</Keystore>

<Supernode-list fixed="true">

        <Supernode ip="192.168.1.1" />

</Supernode-list>

</Profile>
```

6.1.1.2.2 Node Library

Node Library es una librería software diseñada para proporcionar al software de terceros la capacidad de comunicarse con otros Nodos de la red a través del sistema de comunicación segura desarrollado en este proyecto.

Las características principales de esta librería son su ligereza y simplicidad. Esto se logra gracias a que la mayor parte de la carga computacional necesaria para establecer una conexión con el sistema es realizada por el módulo Node Services. Para ello, esta librería realiza la función de controlador de este módulo por medio de comandos a través de un canal socket al puerto 1171 de la dirección IP local 127.0.0.1.

Esta librería proporciona las herramientas necesarias para gestionar interacciones de comunicación tanto en modo servidor como en modo cliente. El software de terceros deberá utilizar unas herramientas u otras en función de su naturaleza, cliente o servidor.

6.1.1.2.2.1 Herramientas de cliente

Independientemente del lenguaje de programación en el que se haya desarrollado la librería, aunque en este caso se Java, el usuario de esta librería podrá generar un objeto Session. Éste objeto representa un canal abierto a través del sistema hacia otro Nodo. Estas sesiones proporcionan al usuario de la librería los siguientes métodos:

Connect

Explicación:
Este método inicia el proceso de conexión con el Nodo de destino.
Parámetros:
Recibe como parámetros la dirección IP del Nodo de destino y el puerto bajo el que estará escuchando la aplicación servidora del mismo Nodo.
Resultado:
Si la conexión es establecida exitosamente el método connect no retornará valor alguno, en caso contrario hará uso del mecanismo de notificación de errores propio del lenguaje de programación en el que esté implementado, en este caso a través de excepciones.

Transmit

Explicación:
Una vez realizada la conexión con éxito, este método permitirá transmitir la información deseada al Nodo de destino.
Parámetros:
Recibe como único parámetro el array de bytes generado a partir de la información que se desee transmitir al Nodo de destino. Dado que el sistema no necesita interpretar dicha información, puede tratarse de cualquier tipo de datos.
Resultado:
Si la información es transmitida correctamente, este método retornará en forma de array de bytes la información que el software del Nodo destino haya estimado como respuesta. En caso

contrario, se hará uso del mecanismo de notificación de errores propio del lenguaje de programación en el que esté implementado, en este caso a través de excepciones.

Disconnect

Explicación:
Este método permite iniciar el proceso de desconexión de una sesión abierta.
Parámetros:
No recibe ningún parámetro.
Resultado:
Si la desconexión se realiza exitosamente el método disconnect no retornará valor alguno, en caso contrario hará uso del mecanismo de notificación de errores propio del lenguaje de programación en el que esté implementado, en este caso a través de excepciones.

6.1.1.2.2.2 Herramientas de Servidor

El usuario de esta librería podrá generar un objeto ServerSession. Este objeto recibe como parámetro una instancia de una interfaz cuyos métodos permitirán al programa gestionar los eventos producidos a lo largo de una sesión con un Nodo cliente. Es importante tener en cuenta que este objeto solo podrá hacerse cargo de una única sesión haciendo necesaria la implementación por parte del software de terceros de un gestor de sesiones en modo servidor. Una vez instanciado el objeto ServerSession el programa únicamente tendrá que especificar el porto TCP al que deberá escuchar para gestionar la petición de una nueva sesión. Los métodos de los que dispone el software de terceros para gestionar los eventos en una sesión en modo servidor son:

OnConnectionRequest

Explicación:
Este método se ejecutará cada vez que un Nodo solicite establecer una conexión.
Parámetros:
Recibe por parámetro la dirección IP del Nodo que solicita la conexión.
Resultado:
En caso de aceptar dicha solicitud de conexión, el programa deberá retornar el valor boolean true, en caso contrario false.

OnTransmit

Explicación:
Este método se ejecutará cada vez que se reciba una transmisión por parte del Nodo cliente.
Parámetros:
Recibe por parámetro un array de bytes que constituirá la información transmitida desde el Nodo cliente.
Resultado:
Se deberá retornar un array de bytes que constituirá la respuesta por parte del Nodo servidor a la transmisión recibida.

6.1.2 Supernodo

Un Supernodo es un elemento del sistema cuyo único fin es dar servicio a los Nodos que componen la red ofreciéndoles la capacidad de establecer una comunicación entre ellos a través de sí mismo. La razón por la que es necesaria la intervención del Supernodo en todas las comunicaciones es que la red que se tiene como objetivo securizar está compuesta por Nodos desconocidos, para el sistema y entre sí. Esto hace necesario establecer una tercera parte de confianza durante las comunicaciones Nodo a Nodo.

De este modo, un Nodo que quiera comunicarse con otro deberá solicitar dicha comunicación al Supernodo que considere oportuno. Este negociará un nivel de seguridad con el Nodo que ha iniciado la comunicación para que toda la información que se envíen ambos se transmita de forma segura.

Una vez securizado el canal Nodo Emisor - Supernodo, el Supernodo enviará una solicitud de conexión al Nodo destinatario, iniciando así un nuevo proceso de securización que mantendrá seguro el canal de comunicación Supernodo - Nodo Destinatario (Ver ilustración 7).

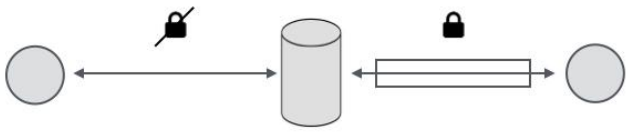

Ilustración 7 Comunicación Nodo a Nodo

Este modo de comunicación introduce una tercera parte de confianza entre dos Nodos desconocidos. No obstante, la comunicación Nodo a Supernodo pasa a ser vulnerable a la suplantación de identidad de un Supernodo. Es por esto por lo que cada uno de los Supernodos deberá tener un certificado firmado por la entidad certificadora del sistema, puede verse información más detallada en el capítulo 6.2.1.

Otra de las razones por las que es importante la utilización de Supernodos en el sistema es que, de no existir estos, el nivel de seguridad de las comunicaciones se tiene que adecuar a las capacidades de cada uno, si dos Nodos se comunican sin un Supernodo el nivel de seguridad de la comunicación se verá limitado al nivel más alto que sea capaz el Nodo más débil. De esta forma, la existencia del Supernodo permite a cada Nodo comunicarse con el máximo nivel de seguridad que sea capaz. Del mismo modo, para garantizar el no repudio, es necesario llevar un registro de todas las acciones que realizan los Nodos en el sistema y por ello, es vital que un elemento de confianza monitorice y registre dichas acciones.

Si bien es cierto que todas estas razones hacen imprescindible la utilización de Supernodos en el sistema, es necesario destacar que la utilización de dichos elementos supone un gran aumento de la carga de tiempo de las comunicaciones dentro de la red. Además, supone uno de los puntos vulnerables a la hora de atacar el sistema, por lo que tiene que ser debidamente protegido.

Por todo ello, se propone como trabajo futuro estudiar la forma de eliminar los Supernodos sin perder los beneficios que aportan.

6.1.2.1 Características Hardware de un Supernodo

El hardware del Supernodo debe ser lo suficientemente potente como para permitirle gestionar múltiples sesiones con el nivel de seguridad más alto posible. Se trata de un dispositivo potente cuyas capacidades varían en función del número de Nodos a los que se espera que de servicio. Sus especificaciones varían desde en función del tamaño de la arquitectura que se presente.

6.1.2.2 Características Software de un Supernodo

En cuanto al Software se refiere, el Supernodo está formado por un programa principal denominado Supernode Proxy. Este programa se encarga de esperar solicitudes de comunicación por parte de alguno de los Nodos de la red a la que da servicio con el fin de permitirles establecer una comunicación a través de sí mismo. El puerto TCP establecido para la escucha es el 1170.

Durante su funcionamiento, este programa almacena los registros obtenidos en una base de datos interna. Esta base de datos está preparada para que, en un futuro desarrollo de este proyecto, proporcionar los datos necesarios a otros programas del Supernodo con el objetivo de realizar analíticas y compartir sus resultados con otros Supernodos (Ver ilustración 8).

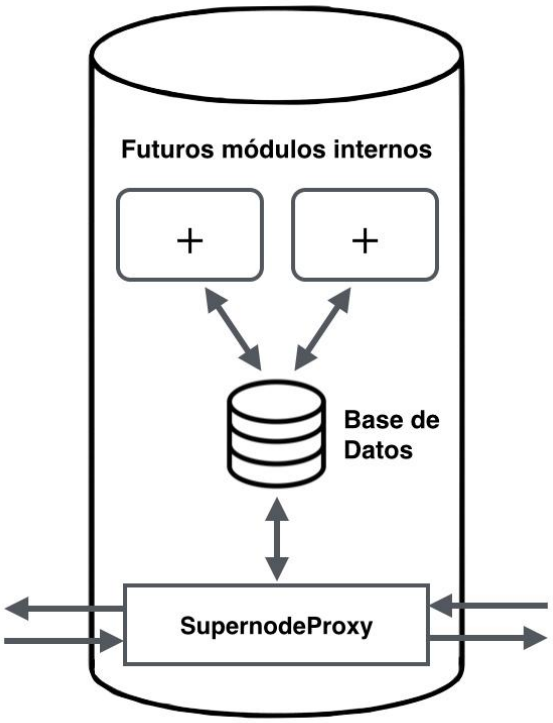

Ilustración 8 Estructura interna de un Supernodo

6.1.2.2.1 Recursos del Supernode Proxy

Al igual que el módulo Node Services del Nodo, el Supernode Proxy posee una serie de recursos almacenados en una carpeta denominada data. Estos recursos los componen un almacén de llaves donde se encuentra el certificado del Supernodo y un archivo de extensión .pem que almacenará la clave pública de la entidad certificadora del sistema.

Dada la importancia de dichos recursos, la carpeta data únicamente debe proporcionar permisos de lectura y escritura a la aplicación Supernode Proxy y al usuario administrador del sistema. En caso de que resulten comprometidos, la entidad certificadora del sistema se encargará de anular el certificado que posee el Supernodo afectado y será necesaria la solicitud de un nuevo certificado firmado por parte del personal administrador del Supernodo.

6.2 ASPECTOS DE SEGURIDAD GARANTIZADOS

La infraestructura de comunicación descrita en el capítulo anterior permite tener un gran control sobre todas las fases y aspectos de la comunicación de los Nodos del sistema pudiendo establecerlas, analizarlas y denegarlas. En este capítulo se muestran los métodos diseñados para garantizar los conceptos básicos de la seguridad de la información haciendo uso de dicha infraestructura. Para ello, y dada la relación existente entre algunos de estos conceptos, se han desarrollado soluciona conjuntas para algunos de ellos como son autenticidad, confidencialidad e integridad por un lado y disponibilidad y no repudio por otro. A continuación se profundiza en estos conceptos y en cómo se aplican al proyecto.

6.2.1 Autenticidad, Confidencialidad e Integridad

Debido a la naturaleza heterogénea de la red que se tiene como objetivo securizar, se ha desarrollado un conjunto de modelos de seguridad diseñados específicamente para garantizar los aspectos de autenticidad, confidencialidad e integridad dentro de la amplia gama de capacidades propias de los Nodos del sistema. Estos modelos definen el comportamiento del Nodo durante las fases de verificación, autenticación, registro y cifrado de la información. A continuación se muestra en detalle cada una de las fases.

Verificación
Fase en la que el Supernodo demuestra, ante el Nodo, ser un Supernodo autorizado y verificado por el sistema garantizando ser un Supernodo de confianza al que poder conectarse conectarse. Esta fase protege a los Nodos de ataques de suplantación de identidad del Supernodo, ataques de hombre en el medio, denominados a partir de ahora como MITM por sus siglas en inglés (Man-in-the-middle) y de Supernodos no autorizados por el sistema.
Esta fase, junto con la de autenticación, garantiza el concepto de autenticidad.

Autenticación
En esta fase el Nodo debe verificar su identidad ante el Supernodo. Esta fase permite así al Supernodo atribuir al Nodo la autoría de todos actos que del Nodo durante la comunicación.

Esta fase, junto con la de verificación, garantiza el concepto de autenticidad además de contribuir al no repudio.

Registro
Esta fase permite a Nodos hasta ahora desconocidos en el sistema a dotarse de una identidad. Dicha identidad quedará registrada en la red de Supernodos para poder identificarse en futuras comunicaciones. El proceso de registro debe realizarse de forma que ninguna entidad no autorizada pueda obtener el elemento identificador generado.
Esta fase contribuye a garantizar el concepto de autenticidad.

Cifrado de la información
Esta fase es la encargada de cifrar la información transmitida durante la comunicación Nodo a Supernodo. La información debe ir cifrada de forma que ninguna entidad no autorizada pueda descifrarla en un periodo de tiempo lo suficientemente largo como para considerarse perjudicial.
Esta fase contribuye a garantizar el concepto de autenticidad.

Los modelos de seguridad que definen el comportamiento durante estas fases se dividen en diferentes clases representadas por una letra del abecedario. Esta letra es asignada en función del nivel de seguridad y robustez ofrecido por la tecnología utilizada en cada clase. Del mismo modo, cada clase posee un nivel de seguridad representado por un valor numérico comprendido entre 0 y 9 siendo 0 el nivel de seguridad más alto. El nivel de seguridad de una clase especifica variaciones en los parámetros utilizados durante la implementación de la tecnología atribuida a dicha clase.

Aunque en un principio se había considerado que cada clase de seguridad debía implementar de manera específica y ajena al resto de clases todos y cada uno de los métodos utilizados para garantizar los diferentes conceptos, se ha determinado que no es necesario desarrollar nuevas técnicas para garantizar todos los conceptos mencionados debido a que algunas técnicas desarrolladas en clases inferiores ofrecían la misma robustez con un menor consumo de recursos que en las definidas en clases superiores. Por ello se ha optado por desarrollar un modelo más avanzado que establece como técnica para garantizar un concepto aquella perteneciente a la clase de seguridad más alta definida dentro de las clases de igual o menor robustez que la implementada por el Nodo (Ver tabla 1).

	Verificación	Autenticación	Registro	Cifrado de la información
A0	↓	A1 + Verificación de Nodo por certificado firmado	↓	↓
A1	↓	Auth_key cifrado con RSA	↓	↓
A2	Verificación de Supernodo por certificado firmado	↓	↓	↓
B0	-	Auth_key SHA	Diffie-Hellman	Variante de MTProto

Tabla 1 Modelos de seguridad

En este proyecto se han diseñado las clases de seguridad A y B con un nivel de seguridad comprendido entre 0 y 2 para la clase A y un nivel de 0 para la clase B. No obstante, la escalabilidad de este diseño permite la implementación de nuevos modelos de seguridad llegando a introducir múltiples letras y números por cada clase y nivel si fuera necesario. A continuación, se muestra en detalle las características propias de cada clase.

6.2.1.1 Clase A

La clase de seguridad A tiene como base el uso del sistema criptográfico de clave pública RSA[4]. Esta clase está caracterizada por el uso de certificados tanto por parte del Nodo como del Supernodo. Dichos certificados realizan una tarea fundamental durante los procesos de verificación y autenticación. Es por esto por lo que la clase de seguridad A solo define los métodos utilizados durante estas dos fases y hereda de la clase B los métodos utilizados para las fases de registro y cifrado de la información. Durante el diseño de esta clase, se consideró definir un método propio para la fase de cifrado de la información que consistía en cifrar todo el contenido con la clave pública del destinatario en ambos sentidos de la comunicación Nodo a Supernodo. Pese a tratarse de un método poco convencional a la hora de cifrar la información de forma constante debido a su alto coste computacional, se consideraba adecuado proporcionar un modelo de máxima seguridad para aquellos Nodos que la necesitasen. No obstante, este método fue rechazado debido a que en RSA, el tamaño en bits del contenido a cifrar debía ser igual o menor que el tamaño en bits de la clave de cifrado menos los bits de relleno (once en la mayor parte de los casos). Esto hacía inviable cifrar la información que se debía transmitir.
Para esta clase de seguridad se han diseñado los niveles 0, 1 y 2.

6.2.1.1.1 Nivel 0

El nivel 0 de la clase A añade un proceso de verificación del Nodo a la fase de autenticación del modelo A1. Este proceso se basa en la posesión por parte del Nodo de un certificado firmado

por la entidad certificadora responsable del sistema. Esta entidad es la misma que evalúa los Supernodos firmando sus certificados para su posterior verificación. Del mismo modo, para que un Nodo pueda actuar bajo un modelo de seguridad A0 ha de pasar por un proceso de verificación que, de resultar exitoso, proporcionará a este, una copia de su certificado firmado por la entidad certificadora del sistema. Haciendo uso de este certificado durante la fase de autenticación, el Nodo podrá demostrar al Supernodo que ha sido previamente analizado y verificado por la entidad administradora del sistema. En un futuro desarrollo de este proyecto, este modelo de seguridad influirá positivamente en el nivel de reputación del Nodo que lo implemente.

6.2.1.1.2 Nivel 1

El nivel 1 de la clase A define la fase de autenticación a través de cifrado RSA. En este nivel, la fase de autenticación consta de dos pasos. En el primero, el Nodo envía al Supernodo la clave pública de su certificado. En el segundo, el Nodo envía al Supernodo su elemento identificador, denominado Auth_key, cifrado en primer lugar con la clave privada de su certificado y en segundo con la clave pública del Supernodo, obtenida durante la fase de verificación. La Auth_key es obtenida previamente durante la fase de registro. El cifrado de la Auth_key con la clave privada del Nodo permite garantizar al Supernodo la integridad y autenticidad de dicha clave. Por otro lado, el cifrado con la clave pública del certificado del Supernodo garantiza la confidencialidad de dicha clave ya que la única forma de descifrar el mensaje transmitido es a través de la clave privada que solamente debería poseer el Supernodo. El tamaño mínimo establecido para la clave de todos los certificados del sistema es 1024 bits ya que cualquier tamaño inferior a este es, a fecha de hoy (agosto de 2014), considerado débil debido al corto periodo de tiempo necesario para descifrarlo. Los certificados de los Nodos deberán ser generados por el personal encargado de la puesta en funcionamiento de estos y deberán ser introducidos en un keystore almacenado en la carpeta data con los demás recursos. Por último, deberá especificarse en el archivo de configuración de la librería Node Services del Nodo la contraseña del keystore, el alias del certificado, su contraseña y si está firmado o no por la entidad certificadora del sistema (Para más información véase el capítulo 6.1.1.2.1.2.).

6.2.1.1.3 Nivel 2

El nivel 2 de la clase A es el nivel de seguridad más bajo definido para esta clase en este proyecto. Este nivel está diseñado para aquellos Nodos que carezcan de certificado propio pero sean capaces de cifrar y descifrar con RSA. Este nivel hereda la fase de autenticación del modelo de seguridad inferior más cercano, B0. El nivel 2 define el comportamiento del Nodo durante la fase de verificación. En este nivel, la fase de verificación consta de dos pasos. En el primero, el Supernodo envía al Nodo la clave pública de su certificado firmado por la entidad certificadora del sistema. Una la haya recibido, el Nodo comprueba la valides del cifrado de la entidad certificadora con la clave pública de esta almacenada en la carpeta data del Nodo. Si la verificación resulta ser correcta, el Nodo solicitará al Supernodo una prueba de que la clave pública recibida le pertenece. En ese momento, el Supernodo envía un mensaje de validación simbólico cifrado con la clave privada de su certificado, si el Nodo consigue descifrar el mensaje

de validación con la clave pública obtenida en el paso anterior da por finalizado correctamente el proceso de validación.

6.2.1.2 Clase B

La clase de seguridad B tiene como tecnología principal el esquema de cifrado por bloques AES.

El motivo por el que se ha utilizado AES para esta clase de seguridad es que, dada la falta de recursos para el cifrado propia de los Nodos que adoptan esta clase de seguridad, es necesario la utilización de un mecanismo de cifrado rápido, sencillo de implementar y que requiera poca memoria. Para ello, Mickaël Cazorla, Kevin Marquet y Marine Minier analizaron en su artículo Survey and Benchmark of Lightweight Block Ciphers for Wireless Sensor Networks [13] el consumo de recursos por parte de los diferentes mecanismos ligeros de cifrado existentes del cual se obtiene que AES es el mecanismo más ligero y eficiente de los analizados.

No obstante, el uso de AES solo constituye una parte del conjunto de tecnologías utilizadas por esta clase ya que se encarga de definir las diferentes técnicas utilizadas para las fases de autenticación, registro y cifrado de la información.

La limitación de esta clase de seguridad en esta primera versión del proyecto es la falta de una técnica para implementar la fase de verificación. Dado que carece de una clase inferior de la que heredar una técnica para esta fase, un Nodo que implemente un modelo de seguridad de clase B no podrá comprobar si el Supernodo al que se conecta ha sido analizado y verificado por el administrador del sistema.
Para esta clase de seguridad se ha diseñado el nivel 0.

6.2.1.2.1 Nivel 0

Este nivel define las técnicas para la implementación de las fases de autenticación, registro y cifrado de la información.

6.2.1.2.1.1 Autenticación

A diferencia de otras técnicas de implementación de la fase de autenticación, el modelo B0 no garantiza la identidad del Nodo durante esta fase. La identidad del Nodo se garantiza con la combinación de las fases de autenticación y cifrado de la información. Esto se debe a que durante esta fase, el Nodo con un modelo de seguridad B0 envía al Supernodo el SHA-256 generado a partir de su elemento identificador, denominado Auth_key. Una vez recibido, el Supernodo obtiene de su base de datos el Auth_key correspondiente a dicho SHA. El Auth_key obtenido será almacenado en memoria para utilizarlo como clave de cifrado y descifrado durante la fase de cifrado de la información, momento en el cual, el Nodo podrá garantizar ser el poseedor de dicho Auth_key. En caso de no existir un Auth_key asociado, el proceso de autenticación resultará fallido y el Nodo deberá proceder con la fase de registro.

6.2.1.2.1.2 Cifrado de la información

Para el cifrado de la información en este nivel se ha desarrollado una variante del protocolo de comunicación segura MTProto. El MTProto es un protocolo creado por Nicolai Durov para definir el sistema de cifrado utilizado en la aplicación de mensajería móvil Telegram [14]. Este protocolo tiene como base el cifrado simétrico AES de 256 bits y el cifrado RSA de 2048 bits. Si bien es cierto que ha sido ampliamente criticado por un gran número de expertos en seguridad debido al bajo número de pruebas a las que se ha sometido y la cuestionada calidad de las mismas, este protocolo se adecuaba a las necesidades de este nivel de seguridad. A continuación se muestra un gráfico que define el funcionamiento de este protocolo.

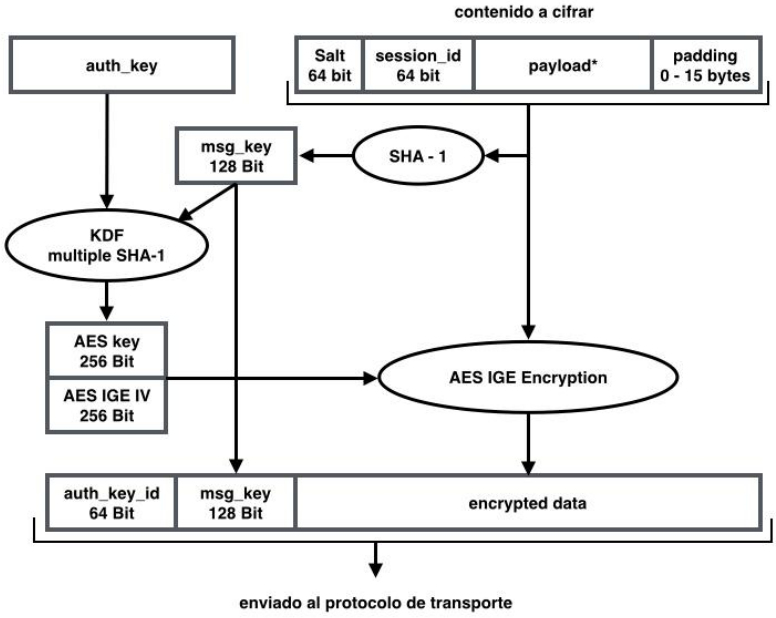

Ilustración 9 Diagrama MTProto

Como puede verse en la ilustración 9 durante el proceso de cifrado, el MTProto genera un paquete compuesto por el contenido del mensaje a enviar, un salt de 64 bits, un id de sesión de 64 bits y de cero a quince bytes de relleno. El paquete generado pasa por un proceso de resumen SHA-1 que da como resultado una clave de 128 bits, denominada msg_key. Esta clave es utilizada junto con una clave compartida denominada authe_key para generar, a través de un proceso de resumen KDF, la clave AES con la que finalmente se cifrará el paquete de información original. Junto con este mensaje cifrado se envían tanto la msg_key como el identificador de 64 bits de la clave compartidas auth_key. Gracias a la información recibida en estos mensajes, el servidor es capaz de descifrar el contenido del mensaje cifrado obteniendo la clave AES a partir del msg_key enviado y la clave compartida auth_key a la que hace referencia el id enviado. Para más información sobre el funcionamiento de MTProto consultar la documentación del protocolo [15].

Como se ha mencioando anteriormente, el mecanismo de cifrado desarrollado para el modelo B0 es una variante del MTProto ya que este protocolo está diseñado a medida para la aplicación de mensajería instantánea Telegram. La variante desarrollada modifica aspectos de MTProto como el paquete inicial del cual se genera el msg_key, el proceso de generación de la clave AES y el mecanismo de referencia de la clave compartida Auth_key. La arquitectura de dicha variante se ve representada en la siguiente figura.

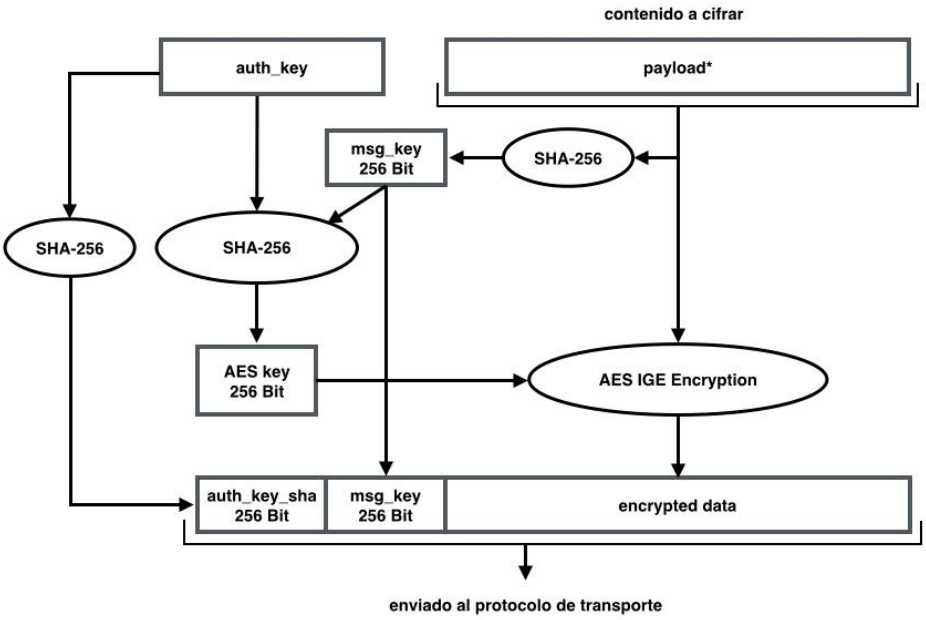

Ilustración 10 Diagrama variante MTProto

Como puede verse en la ilustración 10, durante el proceso de cifrado, el contenido del mensaje a enviar pasa por un proceso de resumen SHA-256 que da como resultado una clave de 256 bits, denominada msg_key. Para añadir una menor tasa de colisión y por lo tanto una mayor seguridad, se el msg_key ha pasado de generarse una función de resumen SHA-1 a una SHA-256.

A diferencia del MTProto, el msg_key generado pasa a concatenarse con el auth_key a través de un proceso convencional de concatenado de bytes. Los bytes resultantes pasan por una función de resumen SHA-256 dando como resultado la clave AES de 256 bits con la que se cifrará el mensaje original. El auth_key toma el valor de la clave generada durante el proceso de registro del Nodo. Junto con el mensaje original cifrado se envían tanto el msg_key como el resultado de la función SHA-256 del auth_key. Gracias a la información recibida en estos mensajes, el servidor es capaz de descifrar el contenido del mensaje cifrado obteniendo la clave AES a partir del msg_key enviado y la clave compartida auth_key a la que hace referencia el SHA enviado.

La razón por la que se envía el SHA del auth_key y no su identificador se debe a que el sistema está diseñado para que la base de datos en la que se almacenan los auth_keys esté distribuida a a lo largo de la red de Supernodos del sistema. Aunque el contenido de cada una de las bases de datos se actualizará con frecuencia, es muy difícil que durante el proceso de registro los Supernodos puedan asignar un identificador numérico sin que este entre en conflicto con otro ya generado por cualquier otro Supernodo del sistema. Por lo tanto, debido a su bajo indice de colisiones [6] se ha establecido como identificador de los auth_keys el resultado de su propia función resumen SHA-256.

Gracias a este protocolo de cifrado es posible garantizar la integridad del mensaje enviado ya que, de ser modificado el msg_key enviado originalmente, pasaría a ser defectuoso a la hora de descifrar el nuevo mensaje. Por otro lado este protocolo completa el proceso de autenticación proporcionando una prueba solida de que el Nodo posee realmente el auth_key al que referencia el SHA enviado tato durante el proceso de autenticación como en el mensaje.

6.2.1.2.1.3 Registro

El proceso de registro se realiza en los casos en los que la identidad de un Nodo carece de identificación o esta no es reconocida durante el proceso de autenticación. Para la fase de registro, el modelo de seguridad B0 realiza un intercambio de claves seguro Diffie-Hellman. Este protocolo criptográfico permite realizar un establecimiento de claves entre partes que no han tenido contacto previo, utilizando un canal inseguro, y de manera anónima, es decir, no autentificada [5].

El proceso de registro consta de dos partes. En la primera, el Nodo genera un par de claves asimétricas con un módulo de 512 bits. Envía la clave pública al Supernodo y este genera un nuevo par en función de esta. De este nuevo par, el Supernodo envía de vuelta la clave pública generada. Una vez que ambos posean la clave pública del otro generarán una clave común secreta de 512 bits denominada auth_key. Dicha clave es la que se utilizará durante el proceso de autenticación del Nodo.

6.2.1.3 Clase C

Esta clase tiene como objetivo garantizar los conceptos de autenticidad, confidencialidad e integridad en los Nodos con menos recursos del sistema. Es por esto por lo que en un principio se le asignó como tecnología principal el algoritmo de cifrado DES. El diseño original de clase C contemplaba el cifrado DES utilizando el auth_key como clave. No obstante, como se muestra en el artículo A Performance Comparison of Data Encryption Algorithms [16], AES es más eficiente, consume menos recursos, tiene un mayor grado de seguridad y soporta claves de hasta 256 bits frente a los 56 bits de DES. Por lo tanto, dado que no existe ningún tipo de ventaja en usar DES frente a AES, la clase C queda inservible y todos aquellos dispositivos que implementasen C pasarán a utilizar un modelo se seguridad de clase B.

No obstante, el alto nivel de recursos requeridos para la implementación del modelo de seguridad de clase B hace necesario el diseño de un modelo de clase C adecuado a las capacidades de

los Nodos con menos recursos. Para un futuro desarrollo de este proyecto, se propone orientar la clase C a tecnologías de cifrado ligero como MIFARE DESFire.

6.2.2 Disponibilidad

Para garantizar la alta disponibilidad del sistema se ha diseñado un mecanismo que permite gestionar un número determinado de sesiones en ejecución, encolar futuras sesiones y redirigir o rechazar aquellas que no es capaz de encolar. Este mecanismo protegerá al sistema frente a sobrecargas de tráfico y ataques de denegación de servicio. Se ha implementado este mecanismo en el software principal del Supernodo, denominado Supernode Proxy, y en el módulo de software interno del Nodo denominado Node Services tanto para las peticiones internas como externas.

El sistema diseñado está preparado para que en un futuro desarrollo de este proyecto se pueda crear una red de Supernodos capaz de gestionar de forma conjunta la carga de sesiones activas en el sistema.

6.2.3 No repudio

Para garantizar el no repudio dentro del sistema desarrollado, se ha diseñado un mecanismo de registro de analíticas que permite al Supernodo recopilar en una base de datos todos los datos relevantes generados durante las comunicaciones a las que da servicio. En concreto, en esta versión del proyecto, los datos registrados son los siguientes:

- Tiempo de inicio y de fin de cada sesión.
- La identificación de los Nodos involucrados en la sesión especificando cuál de ellos actúa en modo cliente y cual en modo servidor.
- Los certificados utilizados por cada uno de los Nodos.
- El puerto TCP de la aplicación del Nodo servidor.
- La versión del protocolo de comunicación y el modelo de seguridad utilizado por ambos Nodos.
- El tamaño del contenido de cada transmisión y el SHA del auth_key enviado.

En un futuro desarrollo del proyecto, un módulo software del Supernodo se alimentará de estos datos para generar perfiles de reputación y confianza de los Nodos. Los Supernodos harán uso de estos perfiles a la hora de limitar las capacidades de comunicación de los Nodos en la red. Dado que el alcance de este proyecto no contempla la elaboración de estos perfiles de reputación, no se han definido unas reglas fijas para la interpretación de las analíticas. No obstante, como primera aproximación se propone partir de las siguientes reglas:

Modificar negativamente la reputación del Nodo cuando:

- Se considere que su seguridad se ha visto comprometida como consecuencia de:
    - Un error en el cifrado.
    - Fallo en la comprobación del checksum (en este caso denominado msg_key).
    - Fallo en la verificación del certificado.

- El Nodo sufra alteraciones en su conducta habitual como:
    - Comunicarse con un Nodo diferente al que se ha comunicado siempre.
    - Cambio de certificado.
    - Pertenezca una red conflictiva.
    - El puerto de la aplicación de destino sea sensible o pertenezca a una lista de puertos conflictivos.

Del mismo modo, como trabajo futuro, los Supernodos compartirán estos perfiles entre sí para generar una base de conocimiento colectiva, de esta forma, cuando un Nodo se traslade de una red del sistema a otra, el Supernodo al que se conecte sabrá de qué Nodo se trata y cuál es su reputación. Durante la compartición de los perfiles, además de tener que verificar su identidad a través de los certificados de los que disponen, los Supernodos tendrán en cuenta al reputación de los Nodos en un nivel proporcional a la reputación que tengan sobre el Supernodo que les ha enviado la información.

6.3 PROTOCOLO DE COMUNICACIÓN

Para llevar a cabo todos los procesos analizados en el capítulo anterior, es necesario diseñar un protocolo de comunicación que permita el intercambio de información entre los diferentes módulos del sistema de manera eficiente. Dado que ninguno de los protocolos existentes permite realizar este tipo de comunicaciones, se ha desarrollado uno propio para este sistema.

Este protocolo se denomina Fennec Protocol (haciendo referencia al nombre interno del proyecto). Este protocolo está diseñado para utilizarse en canales de transmisión de bytes. Los bytes enviados representarán texto codificado en utf-8, estándar de codificación más usado en la actualidad.

Se trata de un protocolo multilínea cuyos órdenes constan, en primer lugar, de un comando principal seguido de cero a n atributos. Las líneas posteriores son interpretadas como cabeceras representadas en forma de clave - valor. Los caracteres que se han establecido para delimitar estas líneas son CR (retorno de carro) y LF (salto de línea) representados a través de los caracteres \r y \n respectivamente. Juntos, estos caracteres definirán el final de una línea y el comienzo de otra. Por otra parte, se ha establecido el carácter espacio en blanco ‘ ’ para delimitar cada uno de los elementos que componen una línea. Puesto que en muchos casos el número de cabeceras es variable, se ha establecido como delimitador de órdenes dos CRLFs seguidos.

Para facilitar la comprensión de este protocolo se han utilizado mecanismos similares a los empleados en el protocolo HTTP, como por ejemplo el formato de los mensajes y los códigos de estado.

Este protocolo es utilizado durante los cuatro tipos de comunicaciones que existen en una sesión. Identificaremos dichas comunicaciones asignando un número a cada una, como se puede observar en la ilustración 11.

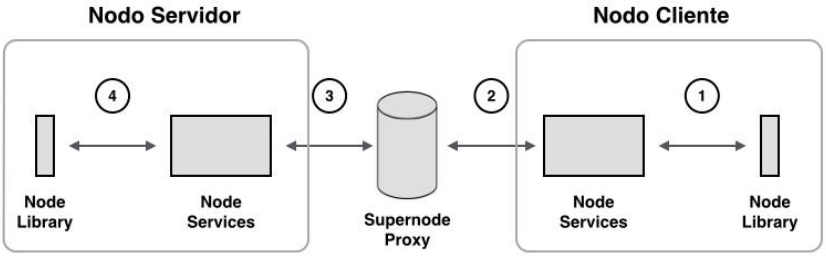

Ilustración 11 Comunicaciones dentro de una sesión

A continuación se muestran los comandos y cabeceras definidas para cada una de las fases de la comunicación. Se ha utilizado el estándar de representación de protocolos de comunicaciones bidireccionales ABNF, (Augmented Backus–Naur Form) para la representación de este protocolo.

El índice de reglas ABNF utilizadas es el siguiente:

Regla	Definición ABNF	Descripción
ALPHA	%x41-5A / %x61-7A	Letra ASCII mayúscula y minúscula (A–Z, a–z)
DIGIT	%x30-39	Digito decimal (0–9)
DOUBLE	1*DIGIT "." 1*DIGIT	Número real
SP	%x20	Espacio en blanco
VCHAR	%x21-7E	Caracteres visibles
CR	%x0D	Retorno de carro
LF	%x0A	Salto de línea
CRLF	CR LF	Estándar de Internet para definir nueva línea
BIT	"0" / "1"	Digito binario
STAUS_CODE	%x30-35 2DIGIT	Código de estado
STATUS_MSG	1*VCHAR	Mensaje de estado
DEC_OCTET	DIGIT / %x31-39 DIGIT / "1" 2DIGIT / "2" %x30-34 DIGIT / "25" %x30-35	Valor entero comprendido entre 0 y 255

Regla	Definición ABNF	Descripción
IPV4	3(DEC_OCTET ".") DEC_OCTET	Dirección IP de versión 4
PORT	1*DIGIT	Puerto TCP
PROTOCOL_VERSION	"FENNEC/" DOUBLE	Versión del protocolo Fennec
SEC_CLASS	%x41-5A	Clase del modelo de seguridad
SEC_LEVEL	DIGIT	Nivel del modelo de seguridad
CONTENT_LENGTH	1*DIGIT	Tamaño de un mensaje en Bytes
CONTENT	1*BIT	Contenido de un mensaje en bits
SHA	1*(ALPHA / DIGIT)	Resultado de una función de HASH

Tabla 2 Índice de reglas ABNF para el protocolo Fennec

Códigos de estado

A continuación se muestra la lista de códigos de respuesta del protocolo y las frases estándar asociadas, destinadas a dar una descripción corta del estado de la comunicación. Aunque el sistema está diseñado para actuar conforme a cada uno de estos códigos, aquellas ocasiones en las que ocurra un error inesperado durante la comunicación y el protocolo no contemple recibir un código de estado, el sistema está diseñado para enviar un comando de desconexión que cerrará la sesión independientemente el punto en el que se encuentre.

Código	Mensaje
400	Bad Request
401	Unauthorized
406	Not Acceptable
407	Proxy Authentication Required. Este código lo devuelven todos aquellos métodos que se ejecuten sin haber realizado previamente los pasos de securización
409	Conflict
412	Precondition Failed
500	Internal Server Error
505	Fennec Version Not Supported

Tabla 3 Códigos de estado utilizados en el protocolo Fennec

6.3.1 Negociación de versión

Explicación

Para garantizar la escalabilidad del protocolo, se ha introducido una fase de negociación de versión para que ambos módulos acuerden una versión común del protocolo nada más iniciar la comunicación. Todos los elementos del sistema deberán ser capaces de operar con otros elementos que implementen una versión igual o inferior del protocolo. Esta es la única parte del protocolo que no puede modificarse en ninguna de las versiones posteriores.

Petición

Comando	HELLO
Parámetros	versión de protocolo
Cabeceras	-
Representación ABNF	"HELLO" SP PROTOCOL_VERSION 2CLRF

Tabla 4 Petición de negociación de versión del protocolo Fennec

Respuesta

Comando	HELLO_RESULT
Parámetros	versión de protocolo, código de estado, mensaje de estado
Cabeceras	-
Representación ABNF	"HELLO_RESULT" SP PROTOCOL_VERSION SP STAUS_CODE SP STAUS_MSG 2CLRF
Posibles códigos de estado	- **200**: Todo hay ido bien, la versión del protocolo se ha establecido correctamente - **400**: Sintaxis incorrecta, el protocolo no se ha implementado como debiera - **500**: Fallo interno del Supernodo

Tabla 5 Respuesta de negociación de versión del protocolo Fennec

6.3.2 Iniciación

Explicación

La fase de iniciación está diseñada para utilizarse únicamente durante la comunicación 3. Establece el periodo de tiempo en el que el Supernodo pasa a modo servidor esperando que el Nodo con el que se comunica lleve la iniciativa durante las fases de negociación de modelo de seguridad, verificación, registro y autenticación. Una vez finalizadas dichas fases, el Nodo enviará un comando de finalización de iniciación para volver a ponerse en modo servidor permitiendo así llevar la iniciativa al Supernodo. Esta fase permite utilizar sin alteraciones de protocolo las mismas fases de negociación de modelo de seguridad, verificación, registro y autenticación definidos tanto para la comunicación 2 como en la 3.

Petición

Comando	INIT
Parámetros	-
Cabeceras	-
Representación ABNF	"INIT" 2CLRF

Tabla 6 Petición de iniciación del protocolo Fennec

Respuesta

Comando	FINISH_INIT
Parámetros	-
Cabeceras	-
Representación ABNF	"FINISH_INIT" 2CLRF
Posibles códigos de estado	-

Tabla 7 Respuesta de iniciación del protocolo Fennec

6.3.3 Negociación de modelo de seguridad

Explicación

La fase de negociación de modelo de seguridad está diseñada para utilizarse únicamente durante las comunicaciones 2 y 3. Durante esta fase se negocia el modelo de seguridad utilizado durante el resto de las comunicaciones de la sesión. Si una vez establecido el modelo de seguridad alguno de los dos elementos no actúa conforme a lo esperado bajo ese nivel de seguridad el otro elemento considerará sospechosa su conducta y finalizará la sesión de inmediato. Durante la negociación, el Nodo envía al Supernodo tanto la clase como el nivel de seguridad del modelo

bajo el cual está configurado para funcionar, si el Supernodo acepta dicho modelo pasan a la siguiente fase.

Petición

Comando	SECURITY
Parámetros	-
Cabeceras	Security-Class, Security-Level
Representación ABNF	"SECURITY" CRLF "Security-Class:" SP SEC_CLASS CRLF "Security-Level:" SP SEC_LEVEL 2CLRF

Tabla 8 Petición de negociación de modelo de seguridad del protocolo Fennec

Respuesta

Comando	SECURITY_RESULT
Parámetros	código de estado, mensaje de estado
Cabeceras	-
Representación ABNF	"SECURITY_RESULT" SP STAUS_CODE SP STATUS_MSG 2CLRF
Posibles códigos de estado	- **200**: Todo hay ido bien, el modelo de seguridad se ha establecido correctamente - **400**: Sintaxis incorrecta, el protocolo no se ha implementado como debiera - **406**: El supernodo no acepta este modelo de seguridad - **500**: Fallo interno del Supernodo

Tabla 9 Respuesta de negociación de modelo de seguridad del protocolo Fennec

6.3.4 Verificación

Explicación

La fase de verificación está diseñada para utilizarse únicamente durante las comunicaciones 2 y 3. El proceso de verificación consta de dos partes. En la primera, el Nodo solicita al Supernodo su certificado público. Una vez recibido, en la segunda parte, el Nodo responde mostrando el resultado de la validación del certificado, en caso de ser correcto, el Supernodo concluirá enviando un comando cifrado con su clave privada.

Petición 1

Comando	VERIFY_A_0
Parámetros	-
Cabeceras	-
Representación ABNF	"VERIFY_A_0" 2CLRF

Tabla 10 Petición 1 de verificación del protocolo Fennec

Respuesta 1

Comando	VERIFY_A_0_RESULT
Parámetros	-
Cabeceras	Content-Length
Representación ABNF	"VERIFY_A_0_RESULT" CLRF "Content-Length:" SP CONTENT_LENGTH 2CLRF CONTENT
Posibles códigos de estado	-

Tabla 11 Respuesta 1 de verificación del protocolo Fennec

Petición 2

Comando	VERIFY_A_1
Parámetros	código de estado, mensaje de estado
Cabeceras	-
Representación ABNF	"VERIFY_A_1" SP STAUS_CODE SP STATUS_MSG 2CLRF
Posibles códigos de estado	- **100**: La validación ha sido un éxito, continuar. - **400**: Sintaxis incorrecta o protocolo mal interpretado. - **412**: La clave pública es incorrecta o esta mal formada. - **401**: Error de certificado, no valido - **500**: Fallo interno del Nodo

Tabla 12 Petición 2 de verificación del protocolo Fennec

Respuesta 2 (Cifrado con clave privada del Supernodo)

Comando	VERIFY_A_1_RESULT
Parámetros	código de estado, mensaje de estado
Cabeceras	-
Representación ABNF	"VERIFY_A_1_RESULT" SP STAUS_CODE SP STATUS_MSG 2CLRF
Posibles códigos de estado	- **200**: Todo correcto - **500**: Fallo interno del Supernodo

Tabla 13 Respuesta 2 de verificación del protocolo Fennec

6.3.5 Registro

Explicación

La fase de registro está diseñada para utilizarse únicamente durante las comunicaciones 2 y 3 solo en aquellos casos en los que el Nodo carezca de identificador o el proceso de autenticación haya fallado. Durante esta fase el Nodo envía un primer comando indicando el tamaño en bytes del contenido que enviará a continuación. Este contenido constituye los bytes de la clave pública del Nodo generada durante el proceso de generación de claves Diffie-Hellman. Una vez enviado, el Supernodo responderá mostrando el resultado de la generación de un segundo par de claves en función de la clave pública recibida, en caso de ser correcto, el Supernodo transmitirá el contenido asociado a su clave pública generada. Si el identificador generado es igual a otro ya registrado se recomienda repetir el proceso de registro una vez más.

Petición

Comando	REGISTER
Parámetros	-
Cabeceras	Content-Length
Representación ABNF	"REGISTER" CLRF "Content-Length:" SP CONTENT_LENGTH 2CLRF CONTENT

Tabla 14 Petición de registro del protocolo Fennec

Respuesta

Comando	REGISTER_RESULT
Parámetros	código de estado, mensaje de estado
Cabeceras	Content-Length

Representación ABNF	"REGISTER_RESULT" SP STATUS_CODE SP STATUS_MSG CLRF "Content-Length:" SP CONTENT_LENGTH 2CLRF CONTENT
Posibles códigos de estado	- **200**: Proceso correcto - **400**: Sintaxis incorrecta o protocolo mal interpretado. - **409**: El identificador generado es igual a otro ya registrado. - **412**: La clave pública es incorrecta o está mal formada. - **500**: Fallo interno del Supernodo

Tabla 15 Respuesta de registro del protocolo Fennec

6.3.6 Autenticación

La fase de autenticación está diseñada para utilizarse únicamente durante las comunicaciones 2 y 3. Existen dos protocolos de actuación a la hora de implementar la fase de autenticación. Dicha implementación va en función del modelo de seguridad adoptado durante el proceso de negociación de modelo de seguridad.

6.3.6.1 Clase A

Explicación

El proceso de autenticación del modelo A consta de dos partes. En la primera, el Nodo envía un primer comando indicando el tamaño en bytes del certificado público que enviará a continuación. Tras la respuesta del Supernodo, en caso de ser correcta, el Nodo enviará su identificador en un comando cifrado en primero lugar, con la clave privada del Nodo y en segundo con la clave pública del Supernodo adquirida en el proceso de verificación. Finalmente, el Supernodo enviará un código de estado representando el resultado del proceso de autenticación.

Petición 1

Comando	AUTHENTICATE_A_0
Parámetros	-
Cabeceras	Content-Length
Representación ABNF	"AUTHENTICATE_A_0" CLRF "Content-Length:" SP CONTENT_LENGTH 2CLRF CONTENT

Tabla 16 Petición 1 de clase A de autenticación del protocolo Fennec

Respuesta 1

Comando	AUTHENTICATE_A_0_RESULT

Parámetros	código de estado, mensaje de estado
Cabeceras	-
Representación ABNF	"AUTHENTICATE_A_0_RESULT" SP STAUS_CODE SP STATUS_MSG 2CLRF
Posibles códigos de estado	- **100**: La validación ha sido un éxito, continuar. - **400**: Sintaxis incorrecta o protocolo mal interpretado. - **412**: La clave pública es incorrecta o esta mal formada. - **401**: Error de certificado, no valido - **500**: Fallo interno del Supernodo

Tabla 17 Respuesta 1 de clase A de autenticación del protocolo Fennec

Petición 2 (Cifrado en primer lugar, con la clave privada del Nodo y en segundo con la clave pública del Supernodo)

Comando	AUTHENTICATE_A_1
Parámetros	-
Cabeceras	-
Representación ABNF	"AUTHENTICATE_A_1" 2CLRF CONTENT ;

Tabla 18 Petición 2 de clase A de autenticación del protocolo Fennec

Respuesta 2

Comando	AUTHENTICATE_A_1_RESULT
Parámetros	código de estado, mensaje de estado
Cabeceras	-
Representación ABNF	"AUTHENTICATE_A_1_RESULT" SP STAUS_CODE SP STATUS_MSG 2CLRF
Posibles códigos de estado	- **200**: La autenticación ha sido un éxito. - **400**: Sintaxis incorrecta, protocolo mal interpretado o cifrado incorrecto. - **401**: Autenticación incorrecta. - **500**: Fallo interno del Supernodo

Tabla 19 Respuesta 2 de clase A de autenticación del protocolo Fennec

6.3.6.2 Clase B

Explicación

Durante el proceso de autenticación del modelo B el Nodo envía un comando con el resultado de la función HASH de su identificador. En respuesta a esta petición el Supernodo indica si existe un identificador asociado a este HASH registrado en el sistema.

Respuesta

Comando	AUTHENTICATE_B
Parámetros	-
Cabeceras	SHA
Representación ABNF	"AUTHENTICATE_B" CLRF "SHA:" SP SHA 2CLRF

Tabla 20 Petición de clase B de autenticación del protocolo Fennec

Respuesta

Comando	AUTHENTICATE_B_RESULT
Parámetros	código de estado, mensaje de estado
Cabeceras	-
Representación ABNF	"AUTHENTICATE_B_RESULT" SP STAUS_CODE SP STATUS_MSG 2CLRF
Posibles códigos de estado	- **200**: La autenticación ha sido un éxito. - **400**: Sintaxis incorrecta o protocolo mal interpretado. - **401**: Autenticación incorrecta. No existe ningún identificador asociado a ese HASH. - **500**: Fallo interno del Supernodo

Tabla 21 Respuesta de clase B de autenticación del protocolo Fennec

6.3.7 Conexión

Explicación

La fase de conexión está diseñada para utilizarse únicamente las comunicaciones 1 y 2. Durante esta fase se envía un comando indicando en los parámetros la dirección IP del Nodo y el puerto TCP de la aplicación a la que se desea conectar. Como respuesta se recibe el estado de la conexión. En el caso de llevar esta fase a cabo durante la comunicación 2, el módulo Node Services cifrará la petición y descifrará la respuesta con el mecanismo de cifrado asociado al módulo de seguridad acordado durante la fase de negociación.

Petición (Cifrado solo en la comunicación 2)

Comando	CONNECT
Parámetros	dirección IP, puerto
Cabeceras	-
Representación ABNF	"CONNECT" SP IPV4 SP PORT 2CLRF

Tabla 22 Petición de conexión del protocolo Fennec

Respuesta (Cifrado solo en la comunicación 2)

Comando	CONNECT_RESULT
Parámetros	código de estado, mensaje de estado
Cabeceras	-
Representación ABNF	"CONNECT_RESULT" SP STAUS_CODE SP STATUS_MSG 2CLRF
Posibles códigos de estado	- **200**: La conexión se ha establecido correctamente. - **400**: Sintaxis incorrecta, protocolo mal interpretado o cifrado incorrecto. - **401**: La aplicación servidora no ha aceptado la conexión. - **407**: No se ha realizado alguno de los pasos de securización. - **500**: Fallo interno del Supernodo.

Tabla 23 Respuesta de conexión del protocolo Fennec

6.3.8 Solicitud de conexión

Explicación

La fase de solicitud de conexión está diseñada para utilizarse únicamente en las comunicaciones 3 y 4. Esta fase es muy similar a la de conexión, la diferencia radica en que esta se realiza en sentido contrario, el Supernodo recibe un comando de conexión e inicia la fase de solicitud de conexión con el Nodo objetivo. Durante esta fase se envía un comando indicando en los parámetros la dirección IP del Nodo y el puerto TCP de la aplicación a la que se desea conectar además de una cabecera indicando la dirección IP del Nodo que solicita dicha conexión. Como respuesta se recibe el estado de la conexión. Al igual que en la fase de conexión, en el caso de llevar esta fase a cabo durante la comunicación 3, el Supernodo cifrará la petición y descifrará la respuesta con el mecanismo de cifrado asociado al módulo de seguridad acordado durante la fase de negociación.

Petición (Cifrado solo en la comunicación 3)

Comando	CONNECT_REQUEST
Parámetros	dirección IP, puerto
Cabeceras	Source-Host
Representación ABNF	"CONNECT_REQUEST" SP IPV4 SP PORT CLRF "Source-Host:" SP IPV4 2CLRF

Tabla 24 Petición de solicitud de conexión del protocolo Fennec

Respuesta (Cifrado solo en la comunicación 3)

Comando	CONNECT_REQUEST_RESULT
Parámetros	código de estado, mensaje de estado
Cabeceras	-
Representación ABNF	"CONNECT_REQUEST_RESULT" SP STAUS_CODE SP STATUS_MSG 2CLRF
Posibles códigos de estado	- **200**: La conexión se ha establecido correctamente. - **400**: Sintaxis incorrecta, protocolo mal interpretado o cifrado incorrecto. - **401**: La aplicación servidora no ha aceptado la conexión. - **500**: Fallo interno del Nodo.

Tabla 25 Respuesta de solicitud de conexión del protocolo Fennec

6.3.9 Transmisión

Explicación

La fase de transmisión está diseñada para utilizarse en todas las comunicaciones. Durante esta fase se envía un comando indicando en una de sus cabeceras el tamaño en bytes del contenido que enviará a continuación. Este contenido constituye los bytes del mensaje que la aplicación del Nodo origen desea enviar al Nodo destino. Como respuesta se recibe un código de estado indicando el resultado de la transmisión y el tamo del contenido de la respuesta a dicho mensaje que se enviará a continuación. En el caso de tratarse de las comunicaciones 2 y 3 tanto los comandos como el contenido de los mensajes se cifrarán con el mecanismo de cifrado asociado al módulo de seguridad acordado durante la fase de negociación.

Petición (Cifrado solo en las comunicaciones 2 y 3)

Comando	TRANSMIT
Parámetros	-
Cabeceras	Content-Length
Representación ABNF	"TRANSMIT" CLRF "Content-Length:" SP CONTENT_LENGTH 2CLRF CONTENT

Tabla 26 Petición de transmisión de conexión del protocolo Fennec

Respuesta (Cifrado solo en las comunicaciones 2 y 3)

Comando	TRANSMIT_RESULT
Parámetros	código de estado, mensaje de estado
Cabeceras	Content-Length
Representación ABNF	"TRANSMIT_RESULT" SP STAUS_CODE SP STATUS_MSG CLRF "Content-Length:" SP CONTENT_LENGTH 2CLRF CONTENT
Posibles códigos de estado	- **200**: Transmisión correcta. - **400**: Sintaxis incorrecta, protocolo mal interpretado o cifrado incorrecto. - **407**: No se ha establecido una conexión con el Nodo objetivo. - **500**: Fallo interno.

Tabla 27 Respuesta de transmisión del protocolo Fennec

6.3.10 Desconexión

Explicación

La fase de transmisión está diseñada para utilizarse en todas las comunicaciones. Durante esta fase se envía un comando sin parámetros ni cabeceras solicitando la desconexión y cierre de la sesión. Como respuesta se recibe el resultado de la desconexión.

Respuesta

Comando	DISCONNECT
Parámetros	-
Cabeceras	-
Representación ABNF	"DISCONNECT" 2CLRF

Tabla 28 Petición de desconexión del protocolo Fennec

Respuesta

Comando	DISCONNECT_RESULT
Parámetros	código de estado, mensaje de estado
Cabeceras	-
Representación ABNF	"DISCONNECT_RESULT" SP STATUS_CODE SP STATUS_MSG 2CLRF
Posibles códigos de estado	- **200**: Desconexión correcta. - **400**: Sintaxis incorrecta o protocolo mal interpretado. - **407**: No se ha establecido una conexión con el Nodo objetivo. - **500**: Fallo interno.

Tabla 29 Respuesta de desconexión del protocolo Fennec

6.3.11 Cifrado

Explicación

Para poder transmitir mensajes cifrados a durante una sesión, es necesario que el destinatario conozca el tamaño en bytes del contenido cifrado que va a recibir. Por ello, se ha diseñado un comando de cifrado que notifica, en claro, la llegada de un mensaje cifrado y su contenido. De esta manera, el destinatario sabe cuántos bytes debe encapsular en un array de bytes para descifrarlo posteriormente. Los comandos del protocolo mencionados en las anteriores fases y el contenido enviado tras ellos se encapsula en dos comandos de cifrado (Ver ilustración 12).

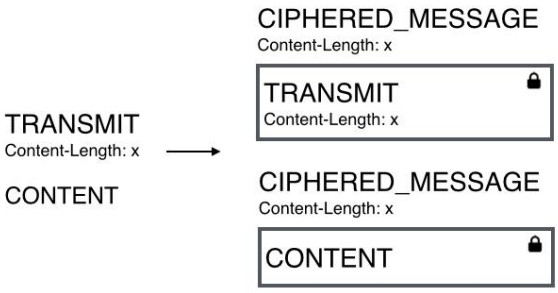

Ilustración 12 Encapsulado de una transmisión cifrada

Petición

Comando	CIPHERED_MESSAGE
Parámetros	-
Cabeceras	Content-Length
Representación ABNF	"CIPHERED_MESSAGE" CLRF "Content-Length:" SP CONTENT_LENGTH 2CLRF

Tabla 30 Petición 1 de envío de contenido cifrado del protocolo Fennec

Respuesta

Comando	CIPHERED_MESSAGE_RESULT
Parámetros	código de estado, mensaje de estado
Cabeceras	-
Representación ABNF	"CIPHERED_MESSAGE_RESULT" SP STAUS_CODE SP STATUS_MSG 2CLRF
Posibles códigos de estado	- **200**: Desconexión correcta. - **400**: Sintaxis incorrecta o protocolo mal interpretado. - **500**: Fallo interno.

Tabla 31 Respuesta 1 de envío de contenido cifrado del protocolo Fennec

Respuesta

Comando	-
Parámetros	-
Cabeceras	-
Representación ABNF	CONTENT

Tabla 32 Petición 2 de envío de contenido cifrado del protocolo Fennec

Respuesta

Comando	CIPHERED_MESSAGE_RESULT
Parámetros	código de estado, mensaje de estado
Cabeceras	-
Representación ABNF	"CIPHERED_MESSAGE_RESULT" SP STAUS_CODE SP STATUS_MSG 2CLRF
Posibles códigos de estado	- **200**: Desconexión correcta. - **400**: Sintaxis incorrecta o protocolo mal interpretado. - **500**: Fallo interno.

Tabla 33 Respuesta 2 de envío de contenido cifrado del protocolo Fennec

7. DESARROLLO

Para demostrar la viabilidad del diseño expuesto en el capítulo anterior, se ha desarrollado un sistema capaz de garantizar los conceptos fundamentales de la seguridad de la información en entornos compuestos por diferentes elementos con recursos limitados y capacidades diversas siguiendo las pautas mencionadas en dicho diseño. Este capítulo analiza los detalles de dicho desarrollo.

Aunque es perfectamente funcional, el sistema software desarrollado no alcanza el nivel de producto final, se trata únicamente de un prototipo desarrollado para demostrar la viabilidad del diseño mencionado. No obstante, se encuentra publicado en un repositorio [1] accesible para todo aquel que desee modificarlo, contribuir a en su mejora o simplemente hacer uso de él.

Debido a que se trata de un prototipo y a que la metodología establecida contempla múltiples iteraciones sobre el proceso de diseño y reimplementación, se ha determinado Java, en concreto su versión 1.7, como lenguaje de programación principal para su desarrollo. La motivación principal para la elección de este lenguaje ha sido su capacidad de ejecutarse en diferentes sistemas operativos, así como su alto nivel de abstracción que permite un rápido desarrollo ideal para trabajos de prototipado y la amplia experiencia del desarrollador en dicho lenguaje, que son otros factores que han inclinado la balanza hacia este lenguaje.

7.1 MÓDULOS SOFTWARE DEL SISTEMA

Para desarrollar los diferentes módulos software que componen el sistema se han creado diferentes componentes, denominados Fennec Libray, Node Library, Node Services y Supernode Proxy. Los tres últimos representan el software principal propio de cada uno de los módulos mencionados en el diseño y el primero es una librería que aúna todas las funcionalidades propias del protocolo, del sistema de comunicación y securización comunes en los tres últimos. En este apartado se analiza en detalle la composición y características de estos módulos.

7.1.1 Fennec Library

Este módulo tiene como objetivo agrupar todas las funcionalidades propias del protocolo, del sistema de comunicación y securización comunes en los módulos Node Library, Node Services y Supernode Proxy. Mediante esta librería se consigue una buena abstracción del código, lo que permite no replicar código y ampliar, en un futuro, sus funcionalidades de forma sencilla. Para dotar a los demás módulos de todas estas funcionalidades se ha exportado en formato jar y

añadido posteriormente como librería Java a cada uno de ellos. No obstante, durante el proceso de desarrollo, se ha creado en cada uno de los módulos una dependencia al proyecto en el que ha desarrollado esta librería para facilitar su implementación.

Las siguientes figuras muestran la composición y arquitectura de esta librería.

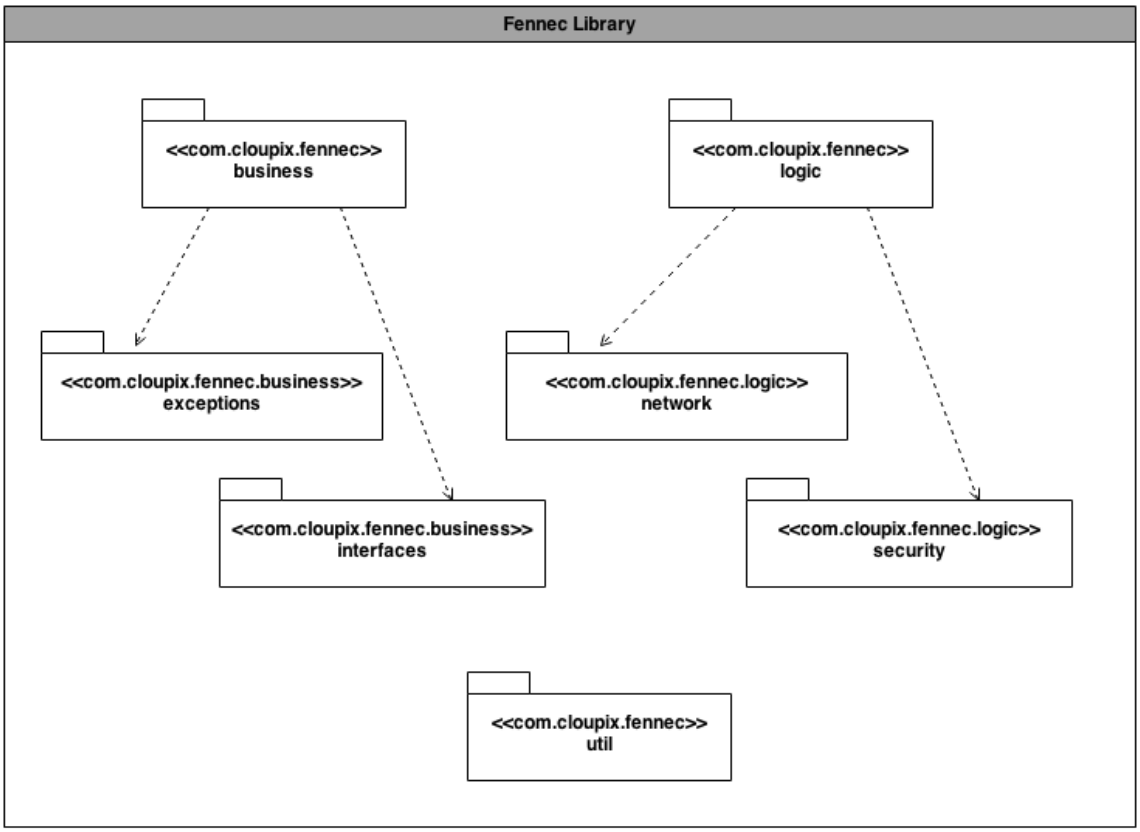

Ilustración 13 Diagrama de paquetes de Fennec Library

Debido al tamaño de las figuras correspondientes al diagrama de clases de Fennec Library, se encuentran ubicadas en anexo 2 de este documento. Si se desea acceder a estas figuras en formato digital estas se encuentran, juneto con el conjunto de módulos software en un repositorio público [1].

7.1.1.1 Utilidades

El paquete util contiene una clase abstracta denominada R de la que heredan otras tres denominadas LibraryR, ServicesR y SupernodeR. El nombre de R viene dado por la inicial de la palabra Resources, “recursos” en inglés. Esta jerarquía de clases permite a cada uno de los módulos definir valores específicos a los atributos comunes utilizados a lo largo de la librería. La instancia de esta clase se crea durante el proceso de carga de configuración durante el inicio del programa. La configuración almacenada en disco pasa a almacenarse en memoria gracias a esta clase. La clase R utiliza el patrón singleton para tener una única instancia a lo largo de la

ejecución. Esto permite determinar, gracias a la clase a la que pertenece la instancia creada, en qué módulo se está ejecutando dentro de la librería.

7.1.1.2 Logic

El paquete logic está diseñado para almacenar todas las clases de carácter operacional, es decir, todas aquellas clases que almacenen las operaciones y métodos utilizados por la clase principal y no constituyan un objeto contenedor de información. Este paquete contiene a su vez dos paquetes denominados network y security. Network almacena las clases que realizan operaciones de red y security aquellas que realizan operaciones de cifrado y seguridad.

7.1.1.2.1 Network

El paquete network contiene dos tipos de clases, clases de protocolo y clases de infraestructura de comunicación. Las clases de protocolo permiten, a cada módulo que implemente la librería, comportarse acorde a las directrices especificadas en la versión del protocolo de comunicación especificada en la configuración. Por otro lado, las clases de infraestructura permiten gestionar la comunicación independientemente del protocolo utilizado.

7.1.1.2.1.1 Clases de infraestructura de comunicación

Existen dos clases orientadas a la gestión de la infraestructura de comunicación, ActiveRequestManager y PassiveRequestManager. La clase ActiveRequestManager se encarga de gestionar las comunicaciones en modo cliente, es decir, aquellas iniciadas por el modulo que implementa la librería. Por otro lado, la clase PassiveRequestManager se encarga de gestionar las comunicaciones en modo servidor, es decir, aquellas cuyo iniciador de la comunicación es otro elemento de la red. Ambas clases contienen como atributo una instancia de una clase encargada de implementar el comportamiento del protocolo de comunicación. Cada petición que realizan o reciben es redirigida a esta clase para determinar, y ejecutar las acciones correspondientes. Estas clases son las únicas que tienen acceso a las clases de protocolo, toda clase que desee interactuar a través de la red deberá utilizar el método correspondiente implementado en una de estas clases.

7.1.1.2.1.2 Clases de protocolo de comunicación

Las clases de protocolo de comunicación son aquellas que determinan el protocolo de actuación a seguir para cada interacción en la red. Estas conforman una jerarquía que tiene como clase principal FennecProtocol, de ella heredan las clases donde se reimplementan los métodos específicos de cada versión del protocolo. Puesto que en esta primera versión del proyecto solo se ha desarrollado una única versión del protocolo de comunicación, la única clase que hereda de FennecProtocol es FennecProtocolV1. A su vez, FennecProtocolV1 pasa a ser la clase padre de aquellas clases que implementen funcionalidades específicas para cada módulo software, en este caso las clases que heredan de FennecProtocolV1 son FennecProtocolV1Library, FennecProtocolV1Services y FennecProtocolV1Supernode. Esta jerarquía conforma un árbol como el que se puede contemplar en la ilustración 14. Para crear la instancia adecuada tanto la

clase FennecProtocol como la FennecProtocolV1 tienen un método factory denominado build que permite instanciar un objeto de la clase correcta a través de los parámetros definidos en R. Esta arquitectura de clases permite añadir nuevas versiones del protocolo y comportamientos para nuevos módulos del sistema de forma sencilla y sin modificar el resto de clases. Para añadir una nueva versión únicamente hay que crear una nueva clase denominada FennecProtocolVX donde X es sustituida por el número de versión que se desee implementar.

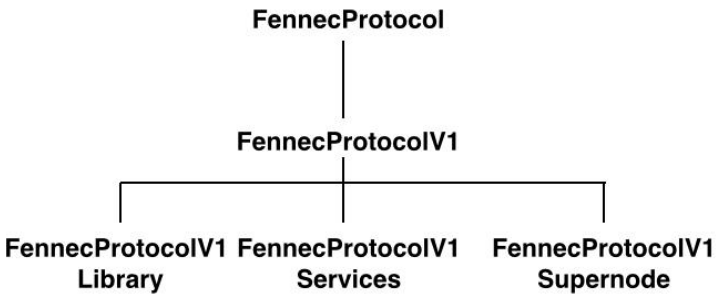

Ilustración 14 Árbol de clases del protocolo de comunicación

7.1.1.2.2 Security

El paquete security contiene dos tipos de clases, clases de cifrado y clases de modelo de seguridad. Las clases de cifrado contienen todos los métodos necesarios para cifrar y descifrar con diferentes tecnologías. Por otro lado, las clases de modelo de seguridad especifican las capacidades se securización propias de cada modelo de seguridad.

7.1.1.2.2.1 Clases de modelo de seguridad

Las clases de modelo de seguridad son aquellas que especifican las capacidades de securización propias de cada modelo de seguridad. Estas conforman una jerarquía que tiene como clase principal SecurityLevel. Debido a la naturaleza de los modelos de seguridad (Ver capitulo 6.2.1.) se ha implementado una jerarquía en cascada. En ella los modelos de seguridad de clase más alta (tratándose en este caso de A la más alta y B la más baja) heredan de los modelos de seguridad más baja. Esto permite reimplementar solo aquellos métodos que deban ser sustituidos por unos de mayor seguridad y así poder utilizar los métodos no reimplementados de la clase de seguridad más cercana (Ver tabla 1). Para crear la instancia adecuada, la clase SecurityLevel tiene un método factory denominado build que permite instanciar un objeto de la clase correcta en función de los parámetros securrityClass y securityLevel representados por una cadena de caracteres y un número entero respectivamente. Estas clases son instanciadas por la clase de protocolo de comunicación al finalizar el proceso de negociación del modelo de seguridad. La instancia generada es utilizada a lo largo de la sesión para cifrar y descifrar

contenido y para determinar el comportamiento del protocolo en función del modelo de seguridad instanciado.

7.1.1.2.2.2 Clases de cifrado

Las clases de cifrado contienen los métodos necesarios para cifrar y descifrar utilizando las tecnologías de cifrado asignadas a las diferentes clases de seguridad. Dentro de esta categoría encontramos las clases RSACipher, AESCipher, DHKeyAgreementAlice y DHKeyAgreementBob.

RSACipher contiene los métodos necesarios para cifrar y descifrar un array de bytes a través de cifrado asimétrico RSA.

AESCipher, al igual que RSACipher, contiene los métodos necesarios para cifrar y descifrar un array de bytes a través de cifrado simétrico AES. Durante el desarrollo de esta clase, ha sido necesario modificar los archivos correspondientes a las políticas de tamaño de clave del JDK (Java Development Kit) debido a que Oracle establece una restricción para las claves AES de 128 bits de tamaño máximo y el diseño de este sistema establecía un tamaño de clave de 256bits. Esto se debe a que varios países, entre otras restricciones, no permiten la importación de software que utilice un cifrado AES con claves superiores a 128 bits. Para modificar estas políticas Oracle ofrece los archivos necesarios para establecer un límite indefinido en el tamaño de clave AES. A priori, esta solución elimina la compatibilidad del módulo software con todas aquellas plataformas que no posean las políticas adecuadas. No obstante, esta restricción solo viene dada por el software que hace uso del JDK de Oracle. Por ello, se ha determinado utilizar OpenJDK ya que carece de esta limitación de clave y permite el desarrollo de este software sin la necesidad de modificar ningún archivo, proporcionando así la compatibilidad con otros sistemas que se había perdido inicialmente.

Por último, las clases DHKeyAgreementAlice y DHKeyAgreementBob proporcionan los métodos necesarios para realizar un intercambio de claves a través de un proceso de generación de claves Diffie-Hellman. DHKeyAgreementAlice proporciona los métodos necesarios para el iniciador del proceso de generación de claves y DHKeyAgreementBob proporciona los métodos necesarios para la otra parte.

7.1.1.3 Interfaces

El paquete interfaces contiene todas las interfaces utilizadas en la librería, que, en este caso, únicamente alberga aquellas encargadas de comunicar la librería con los diferentes módulos software que la implementan. Estas proveen a la librería los métodos que cada módulo deberá implementar permitiendo así transmitir el resultado necesario durante la ejecución del protocolo dentro de la librería, de operaciones ajenas a la esta, definidas internamente en cada uno de los módulos software. Para ello, se ha cerrado una jerarquía que tiene como clase principal ProtocolCallbacks, de ella heredan las interfaces donde se reimplementan los métodos específicos de cada versión del protocolo. Puesto que en esta primera versión del proyecto solo

se ha desarrollado una única versión del protocolo de comunicación, la única clase que hereda de ProtocolCallbacks es ProtocolV1Callbacks. A su vez, ProtocolV1Callbacks pasa a ser la interfaz padre de aquellas clases que implementen funcionalidades específicas para cada módulo software, en este caso las interfaces que heredan de ProtocolV1Callbacks son ProtocolV1CallbacksLibrary, ProtocolV1CallbacksServices y ProtocolV1CallbacksSupernode. De esta forma, el método build de la clase factory del protocolo de comunicación recibe como parámetro un objeto de la clase ProtocolCallbacks que más tarde realizará un casting de la interfaz correspondiente cuando se instancia la clase correspondiente al módulo software que implementa la librería. De esta forma, la librería podrá acceder al resultado de operaciones definidas dentro del módulo software.

7.1.1.4 Excepciones

El paquete exceptions contiene todas aquellas excepciones no definidas en el conjunto de sesiones de java que pueden ocurrir durante la sesión. Estas excepciones forman una jerarquía en función del tipo de excepción. La clase principal de esta jerarquía es SessionException y de ella heredan AuthenticationException, CommunicationException y ProtocolException. AuthenticationException abarca todas aquellas excepciones relativas a la autenticación como la denegación de acceso a ciertos recursos o el fallo del proceso de autenticación. Por otro lado, CommunicationException abarca todas aquellas excepciones relativas a la comunicación de la sesión definidas a través de un código de estado y un mensaje. Por último, ProtocolException abarca todas aquellas excepciones relativas al protocolo de comunicación como la mala implementación de este o el no reconocimiento de un comando entre otros.

7.1.1.5 Objetos

Además de los paquetes exceptions e interfaces, el paquete business contiene todas las clases utilizadas a lo largo de la librería para representar el dominio del sistema. Aunque la mayor parte de la lógica reside en el paquete logic, algunas de estas clases contienen ligeras operaciones asociadas al tipo de información que representan. Estas clases son:

LineParser: Esta clase permite dividir y extraer con facilidad el contenido de una linea de comando del protocolo lanzado las excepciones pertinentes en caso de que la línea no cumpla los requisitos propios del protocolo.

BlockParser: Esta clase permite dividir y extraer con facilidad el contenido, en formato LineParser, de múltiples líneas de comando del protocolo lanzado las excepciones pertinentes en caso de que la línea no cumpla los requisitos propios del protocolo.

CertificateInfo: Esta clase representa la información de un certificado RSA necesaria para obtenerlo de su almacén de llaves.

KeystoreInfo: Esta clase permite almacenar la contraseña de un almacén de llaves y las lista de claves, en formato CertificateInfo, que contiene en su interior.

CipheredContent: Esta clase representa el contenido de un mensaje cifrado indicando la clase de cifrado y almacenando su contenido en bytes y permitiendo extraer de este la authe_key y msg_key asociados a él.

Status: Esta clase define un estado de la comunicación, permite almacenar en el mismo objeto tanto el código de estado como su mensaje. En caso de no proporcionar mensaje de estado durante su creación, esta clase asigna el mensaje predefinido a ese código.

Supernode: Esta clase almacena la información necesaria para contactar con un Supernodo, en esta primera versión solo almacena su dirección IP.

7.1.2 Node Library

Este módulo tiene como objetivo proporcionar al software de terceros la capacidad de comunicarse con otros Nodos de la red a través del sistema de comunicación segura desarrollado. Las características principales de esta librería son su ligereza y simplicidad y gracias a la librería Fennec Library apenas son necesarias clases propias para su implementación (Para conocer todas las características de este módulo ver capitulo 6.1.1.2.2.).

Las siguientes figuras muestran la composición y arquitectura de este módulo.

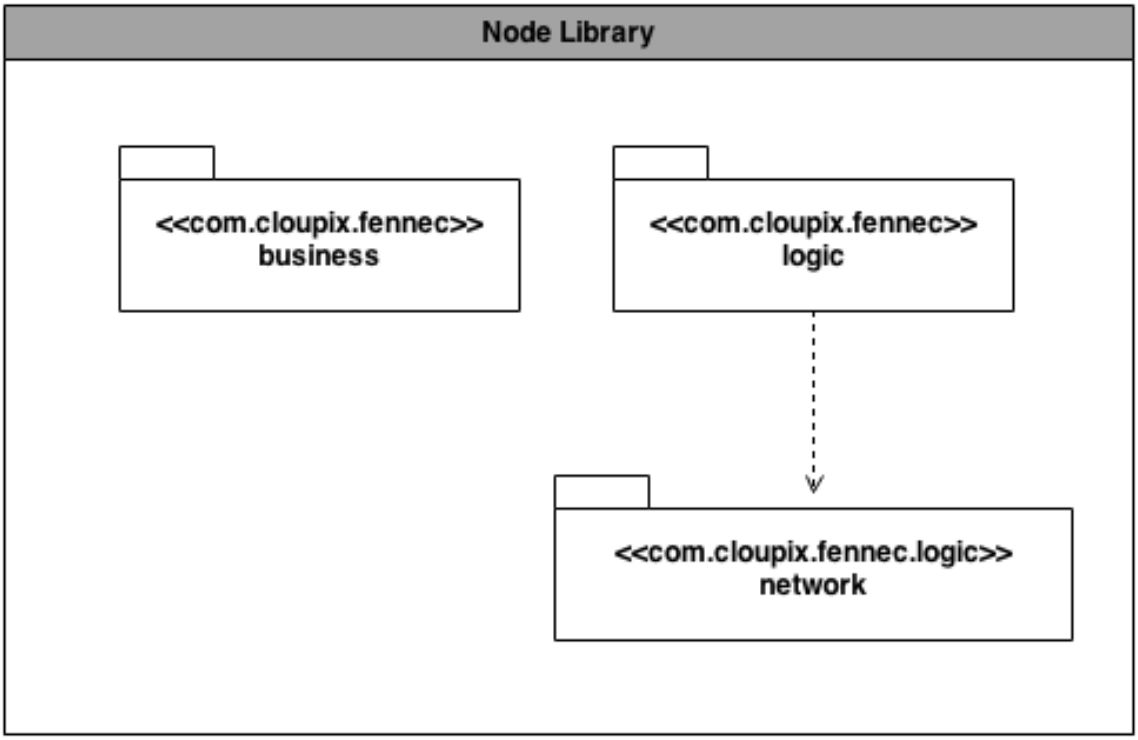

Ilustración 15 Diagrama de paquetes de Node Library

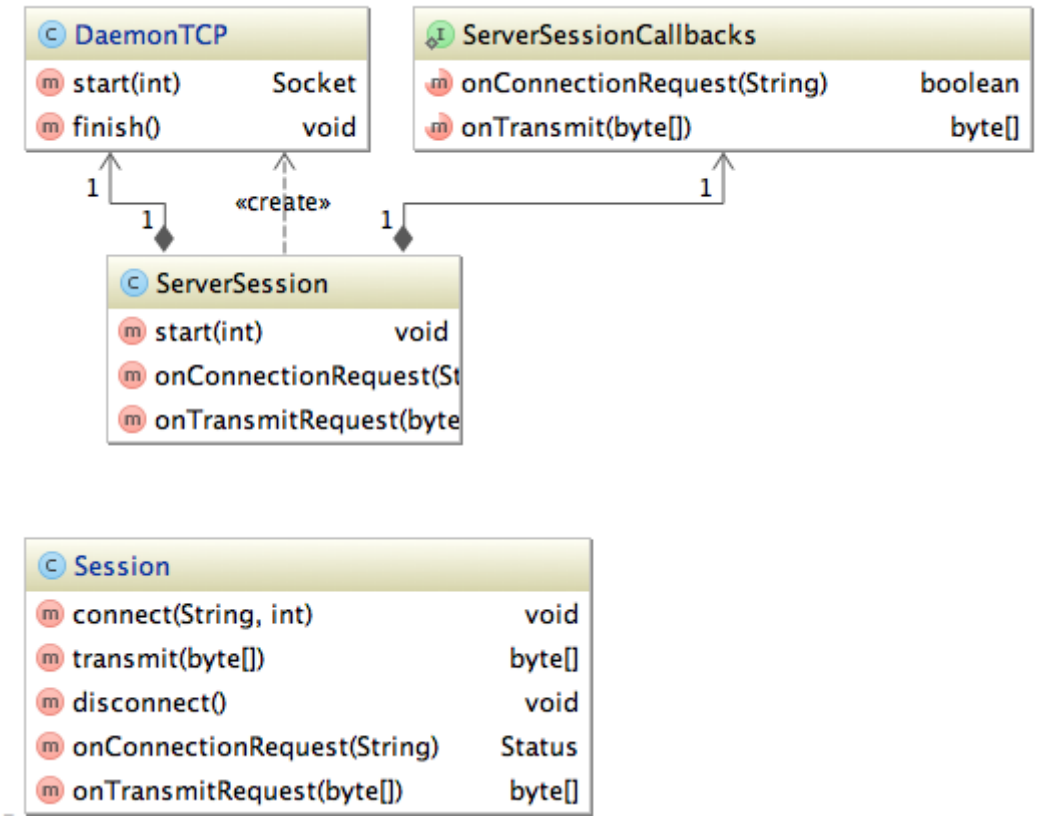

Ilustración 16 Diagrama de clases de Node Library

7.1.2.1 Logic

El paquete logic contiene únicamente un paquete denominado network, que, al igual que en el paquete homónimo de Fennec Library, está diseñado para almacenar todas las clases de carácter operacional. Del mismo modo, el paquete network agrupa todas las clases cuyas operaciones tienen que ver con la comunicación de red. En este caso únicamente contiene una clase denominada DaemonTCP que únicamente se utiliza cuando el software que hace uso de la librería es de carácter servidor. Esta clase permite obtener un objeto socket a partir de una solicitud de conexión a través del puerto TCP especificado para la escucha.

7.1.2.2 Business

El paquete business contiene dos clases denominadas Session y ServerSession. Estas son las clases que deberá instanciar el software de terceros para hacer uso del sistema de comunicación segura desarrollado.

Session: Esta clase es la que el software de terceros de tipo cliente únicamente deberá utilizar para comunicarse con otro Nodo de la red. Una vez instanciada, el objeto creado permitirá hacer uno de los métodos connect, transmit y disconnect. Para ver en detalle las características y funcionamiento de estos métodos ver el capítulo 1.1.1.2.2.1. Herramientas de cliente.
Internamente, la clase Session realizar todas las interacciones de red a través del objeto de la librería Fennec Libary ActiveRequestManager.

ServerSession: Esta clase es la que el software de terceros de tipo servidor únicamente deberá utilizar para gestionar las comunicaciones entrantes de la red. Durante la instanciación, esta clase recibe como parámetro una interfaz ServerSessionCallbacks (definida dentro de la propia clase ServerSession) cuyos métodos deberá implementar el desarrollado del software de terceros que haga uso de esta librería. Los métodos le permitirán gestionar los eventos en una sesión en modo servidor. Para ver en detalle las características y funcionamiento de estos métodos ver el capítulo 6.1.1.2.2.2. Herramientas de Servidor.

Internamente, la clase Session realizar todas las interacciones de red a través del objeto de la librería Fennec Libary PassiveRequestManager. Además, esta clase implementa la interfaz de la misma librería ProtocolV1CallbacksLibrary para recibir los eventos de solicitud de conexión y solicitud de transmisión.

7.1.2.3 Dependencias

Este módulo depende de la librería Fennec Library. Esta librería es utilizada para tener acceso a las funcionalidades propias del protocolo, del sistema de comunicación y securización del sistema desarrollado.

7.1.3 Node Services

Este módulo tiene como objetivo hacer de intermediario entre la librería Node Library y el Supernodo. Para más información sobre las características y funcionamiento de este módulo ver el capítulo 6.1.1.2.1. Node Services.

Las siguientes figuras muestran la composición y arquitectura de este módulo.

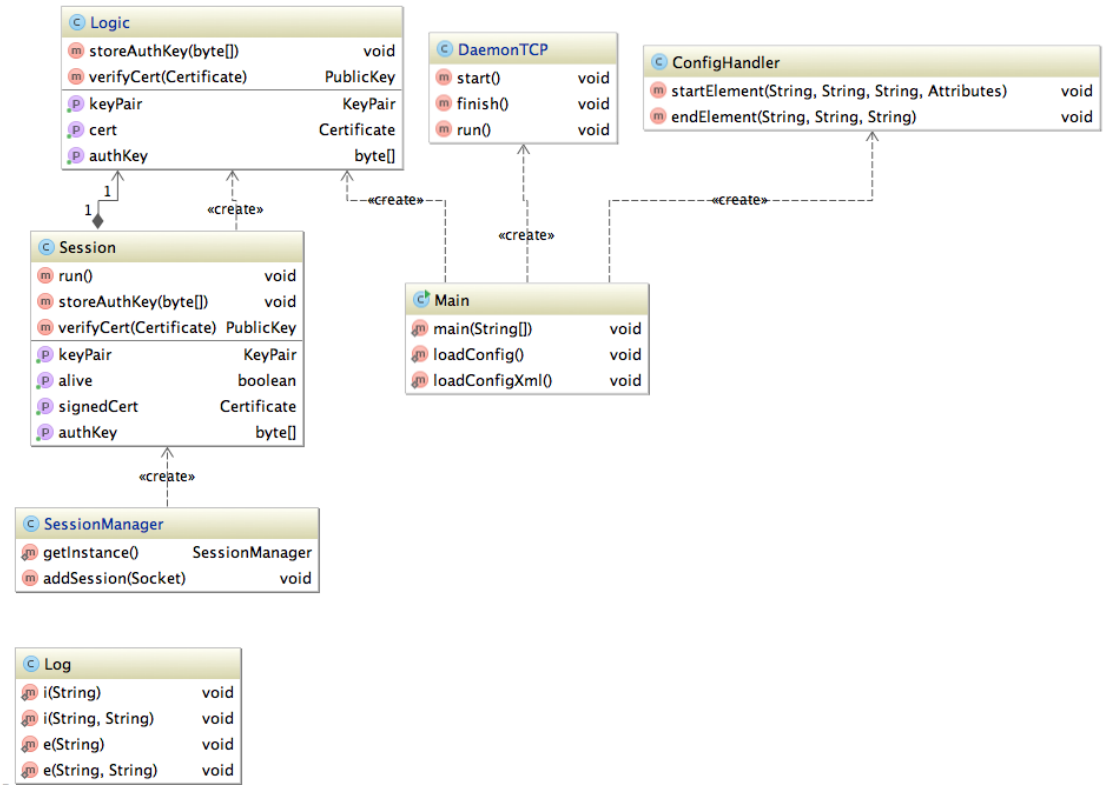

Ilustración 18 Diagrama de clases de Node Services

7.1.3.1 Logic

El paquete logic está diseñado para almacenar todas las clases de carácter operacional, es decir, todas aquellas clases que almacenen las operaciones y métodos utilizados por la clase principal y no constituyan un objeto contenedor de información. Este paquete contiene a su vez un paquete denominado network. Del mismo modo, este paquete también contiene tres clases denominadas Logic, ConfigHandler y SessionManager descritas a continuación:

Logic: Esta clase contiene métodos para el almacenado y obtención de la clave identificadora denominada authKey y métodos para obtención y verificación de certificados RSA. La clave authKey se almacena en claro en un archivo en la carpeta interna data. Debido al carácter sensible de la información almacenada en esta carpeta, únicamente debe tener permisos de lectura y escritura el administrador del sistema. Por otro lado, los certificados se extraen del almacén de llaves almacenado en esta misma carpeta. Estos son obtenidos a a través de las claves y aliases definidos en la configuración del módulo.

ConfigHandler: Esta clase permite obtener el contenido del archivo de configuración en formato xml. Esta información pasa a almacenarse en memoria en la única instancia del objeto R definido

en la librería Fennec Library. Hereda de la clase DefaultHandler y es asignado como atributo a una clase, ya definido en la librería java, denominada SAXParser que es la que realizará la obtención del contenido teniendo en cuenta las directrices definidas en el DefaultHandler.

SessionManager: Esta clase permite gestionar todas las sesiones del Nodo. Debido a que el objeto que representa una sesión implementa la interfaz Runnable, esta clase hace uso de un objeto de la clase ThreadPoolExecutor para encolar las diferentes sesiones establecidas a través de él. El objeto ThreadPoolExecutor almacena las sesiones en dos colas diferentes, una de ejecución y otra de espera. El tamaño de la cola de ejecución viene definido en el archivo de configuración, una vez alcanzado este tamaño, las sesiones comenzarán a almacenarse en la cola de espera. En esta cola las sesiones no son ejecutadas y por lo tanto no consumen apenas recursos del Nodo. En el momento en el que una de las sesiones finaliza, se obtiene la sesión que lleva más tiempo encolada y se comprueba que el canal de comunicación de esa sesión sigue abierto, de ser así pasa a ejecutarse en la cola de ejecución, en caso contrario se desechará y se pasará a la siguiente encolada. Para evitar múltiples instancias de esta clase, se ha seguido el patrón singelton para su instanciación.

7.1.3.2 Network

El paquete network agrupa todas las clases cuyas operaciones tienen que ver con la comunicación de red. En este caso únicamente contiene una clase denominada DaemonTCP utilizado para atender las peticiones externas (de cualquier Supernodo de la red) o internas (de la alguna de las Node Library del Nodo en el que se ejecuta). Esta clase permite obtener un objeto socket a partir de una solicitud de conexión a través de los puertos TCP 1170 y 1171 y enviárselo a la única instancia de la clase SessionManager para su tratamiento. Para permitir la escucha de múltiples puertos TCP en paralelo, esta clase implementa la clase Runnable lo que le permite ejecutarse en paralelo a otros hilos de ejecución.

7.1.3.3 Business

El paquete business contiene una clase denominada Session. Esta clase representa una sesión de comunicación y gestiona todos sus eventos haciendo uso del objeto de la librería Fennec Library de clase PassiveRequestManager. Para permitir la ejecución en paralelo a otras sesiones, la clase Session implementa la interfaz Runnable lo que permite al SessionManager ejecutar dicha sesión en su ThreadPoolExecutor. Además, esta clase implementa la interfaz de la misma librería ProtocolV1CallbacksServices para definir con su propia lógica los métodos de almacenamiento de clave de identidad, obtención de clave de identidad y solicitud y verificación de certificado RSA.

7.1.3.4 Main

La clase Main se encuentra definida en el paquete raíz de este módulo y define las funcionalidades a ejecutar nada más iniciar el programa. Su funcionamiento es sencillo, en primer lugar carga la configuración del archivo de configuración en memoria utilizando la clase R. En segundo lugar, inicia la escucha de la clase DaemonTCP al puerto definido en la configuración como puerto de comunicación interno. Y por último inicia la escucha de la clase DaemonTCP al puerto definido en la configuración como puerto de comunicación externo. Dado que este módulo se caracteriza por tener múltiples hilos de ejecución, se ha representado el conjunto de estos hilos en la siguiente figura.

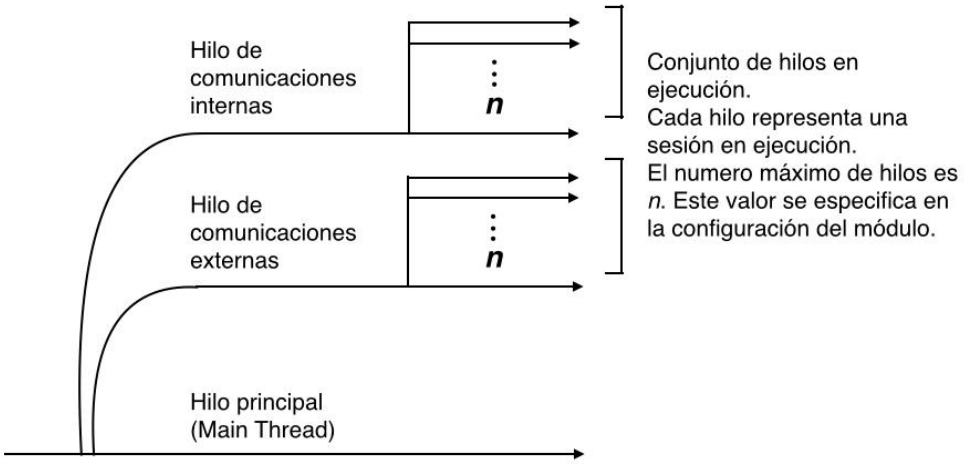

Ilustración 19 Diagrama de hilos de ejecución de Node Services

7.1.3.5 Dependencias

Este módulo depende de la librería Fennec Library. Esta librería es utilizada para tener acceso a las funcionalidades propias del protocolo, del sistema de comunicación y securización del sistema desarrollado.

7.1.4 Supernode Proxy

Este módulo tiene como objetivo hacer de intermediario entre la librería Node Library y el Supernodo. Para más información sobre las características y funcionamiento de este módulo ver el capítulo 1.1.1.2.1. Node Services.

Las siguientes figuras muestran la composición y arquitectura de este módulo.

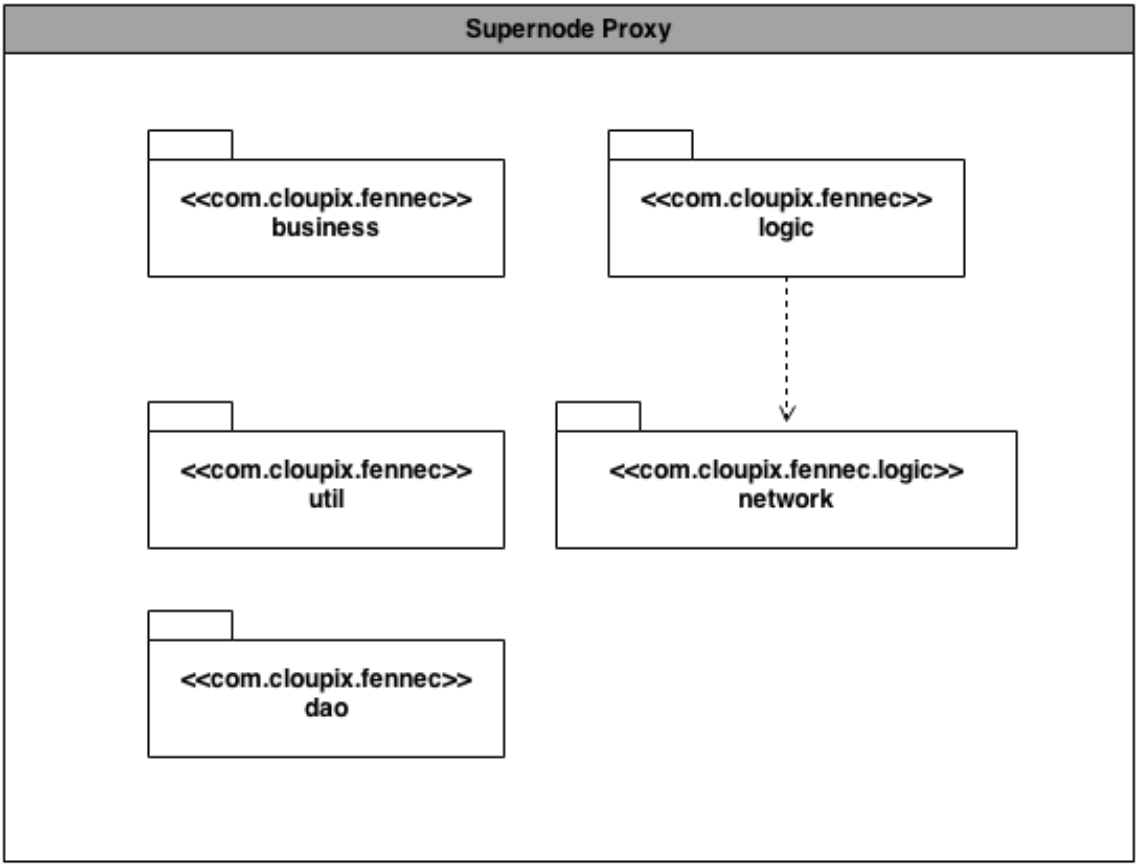

Ilustración 20 Diagrama de paquetes del Supernode Proxy

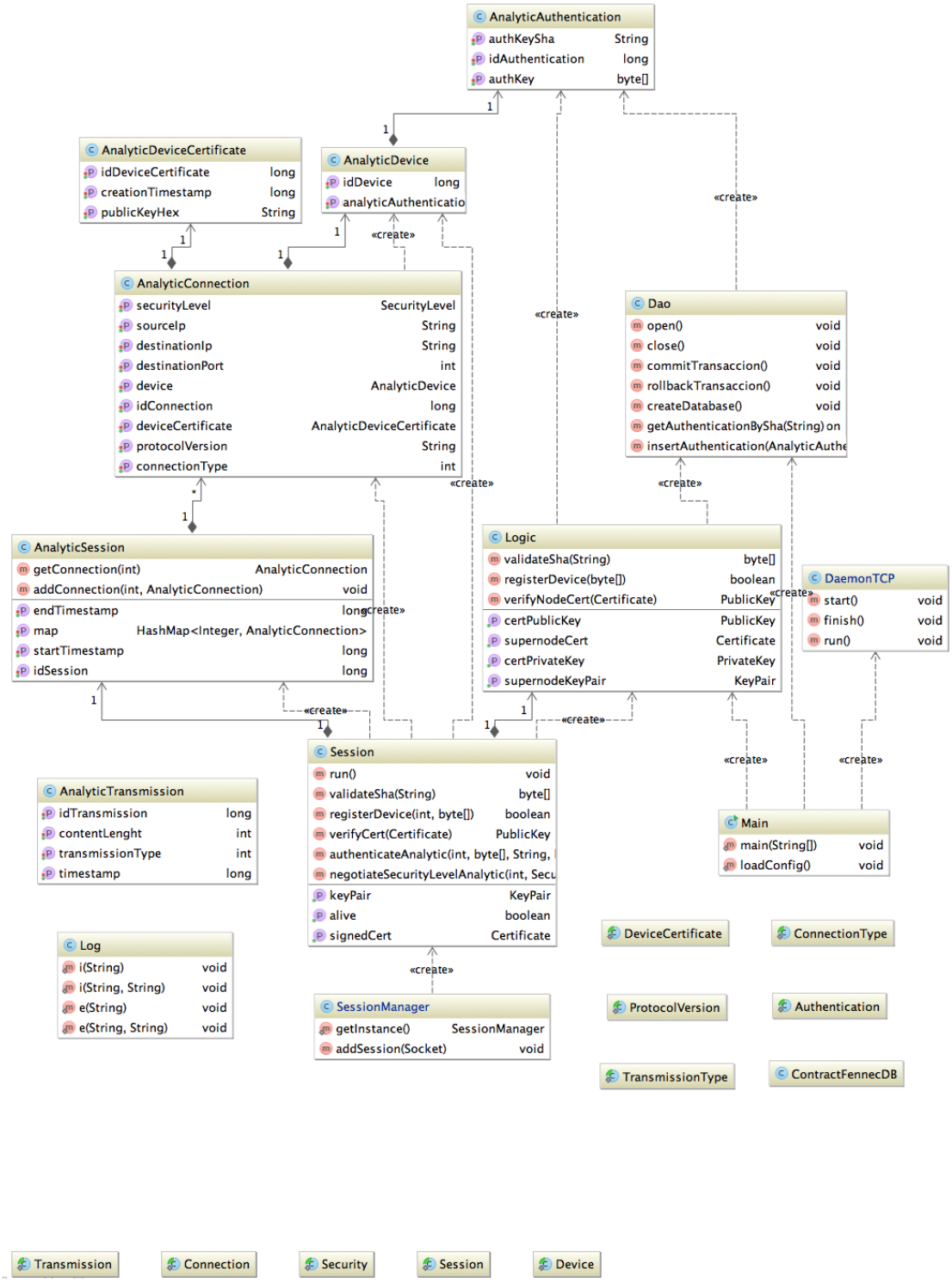

Ilustración 21 Diagrama de clases del Supernode Proxy

7.1.4.1 Logic

El paquete logic está diseñado para almacenar todas las clases de carácter operacional, es decir, todas aquellas clases que almacenen las operaciones y métodos utilizados por la clase principal y no constituyan un objeto contenedor de información. Este paquete contiene a su vez un paquete denominado network y dos clases denominadas Logic y SessionManager descritas a continuación:

Logic: Esta clase contiene métodos para la validación y registro de claves identificadoras de Nodos y métodos para obtención y verificación de certificados RSA. La validez de las claves identificadoras se contrasta con las almacenadas en la base de datos interna alojada en la carpeta data. Debido al carácter sensible de la información almacenada en esta carpeta, únicamente debe tener permisos de lectura y escritura el administrador del sistema. Por otro lado, los certificados se extraen del almacén de llaves almacenado en esta misma carpeta. Estos son obtenidos a a través de las claves y aliases definidos en la configuración del módulo.

SessionManager: Esta clase permite gestionar todas las sesiones del Supernodo. Debido a que el objeto que representa una sesión implementa la interfaz Runnable, esta clase hace uso de un objeto de la clase ThreadPoolExecutor para encolar las diferentes sesiones establecidas a través de él. El objeto ThreadPoolExecutor almacena las sesiones en dos colas diferentes, una de ejecución y otra de espera. El tamaño de la cola de ejecución viene definido en el archivo de configuración, una vez alcanzado este tamaño, las sesiones comenzarán a almacenarse en la cola de espera. En esta cola las sesiones no son ejecutadas y por lo tanto no consumen apenas recursos del Nodo. En el momento en el que una de las sesiones finaliza, se obtiene la sesión que lleva más tiempo encolada y se comprueba que el canal de comunicación de esa sesión sigue abierto, de ser así pasa a ejecutarse en la cola de ejecución, en caso contrario se desechará y se pasará a la siguiente encolada. Para evitar múltiples instancias de esta clase, se ha seguido el patrón singelton para su instanciación.

7.1.4.2 Network

El paquete network agrupa todas las clases cuyas operaciones tienen que ver con la comunicación de red. En este caso únicamente contiene una clase denominada DaemonTCP utilizado para atender las peticiones externas (de cualquier Nodo de la red). Esta clase permite obtener un objeto socket a partir de una solicitud de conexión a través del puerto TCP 1170 y enviárselo a la única instancia de la clase SessionManager para su tratamiento. Para permitir la escucha de múltiples puertos TCP en paralelo, esta clase implementa la clase Runnable lo que le permite ejecutarse en paralelo a otros hilos de ejecución.

7.1.4.3 Business

El paquete business contiene un conjunto de clases para realizar analíticas y una clase denominada Session.

Session: La clase Session representa una sesión de comunicación y gestiona todos sus eventos haciendo uso del objeto de la librería Fennec Library de clase PassiveRequestManager. Para permitir la ejecución en paralelo a otras sesiones, la clase Session implementa la interfaz Runnable lo que permite al SessionManager ejecutar dicha sesión en su ThreadPoolExecutor. Además, esta clase implementa la interfaz de la misma librería ProtocolV1CallbacksSupernode para definir con su propia lógica los métodos de analíticas, registro de clave de identidad de un Nodo, obtención de clave de identidad y solicitud y verificación de certificado RSA.

Clases de analíticas: Estas clases se utilizan para almacenar en memoria todos los datos generados durante una sesión para más tarde volcar dicha información en la base de datos. Las clases pertenecientes a este conjunto son AnalyticAuthentication, AnalyticConnection, AnalyticDevice, AnalyticDeviceCertificate, AnalyticSession y AnalyticTransmission.

7.1.4.4 Dao

El paquete DAO está diseñado para almacenar todas las clases relacionadas con el acceso a base de datos. En concreto, este paquete contiene las clases Dao y ContractFennecDB.

Dao: Esta clase contiene todos los métodos necesarios para gestionar una base de datos SQLite. Estos métodos abarcan la creación de la base de datos, la apertura y cierre de la conexión y la consulta, inserción y modificación de los datos.

La razón por la que se ha determinado utilizar este tipo de base de datos es la siguiente:

Se barajaron tres tipos de bases de datos. En primer lugar se tuvo que elegir entre utilizar una base de datos relacional o no relacional. Se determinó que aunque una base de datos no relacional ofrecía un rendimiento superior al proporcionado por una relacional, era vital estructurar los datos de forma rigurosa con el fin de poder obtener analíticas en un futuro. Por otro lado, dentro de las bases de datos relacionales, se barajó utilizar una base de datos MySQL externa y única para todo el sistema. No obstante, si bien es cierto que este tipo de base de datos proporcionaba un buen desacoplamiento dentro de cada módulo Supernode Proxy y unificación de la base de datos de todos los Supernodos del sistema, el objetivo principal era utilizar una base de datos ligera. Además, la base de datos de Supernodos no debería estar desacoplada debido al tiempo que añadiría a las comunicaciones entre Nodos a los que el Supernodo da servicio e instalar un motor de base de datos MySQL en cada uno de los nodos no era viable debido a su alto consumo de recursos. Por estas razones, se consideró idóneo utilizar una base de datos SQLIte.

ContractFennecDB: Esta clase define en cadenas de caracteres estáticas todos los nombres de tabla y columnas de cada tabla de la base de datos, así como la construcción de sentencias de creación y eliminación de las mismas. El proposito de esta clase es definir de forma segmentada cada uno de los aspectos del diseño de la base de datos (Ver ilustración 22) para facilitar su gestión.

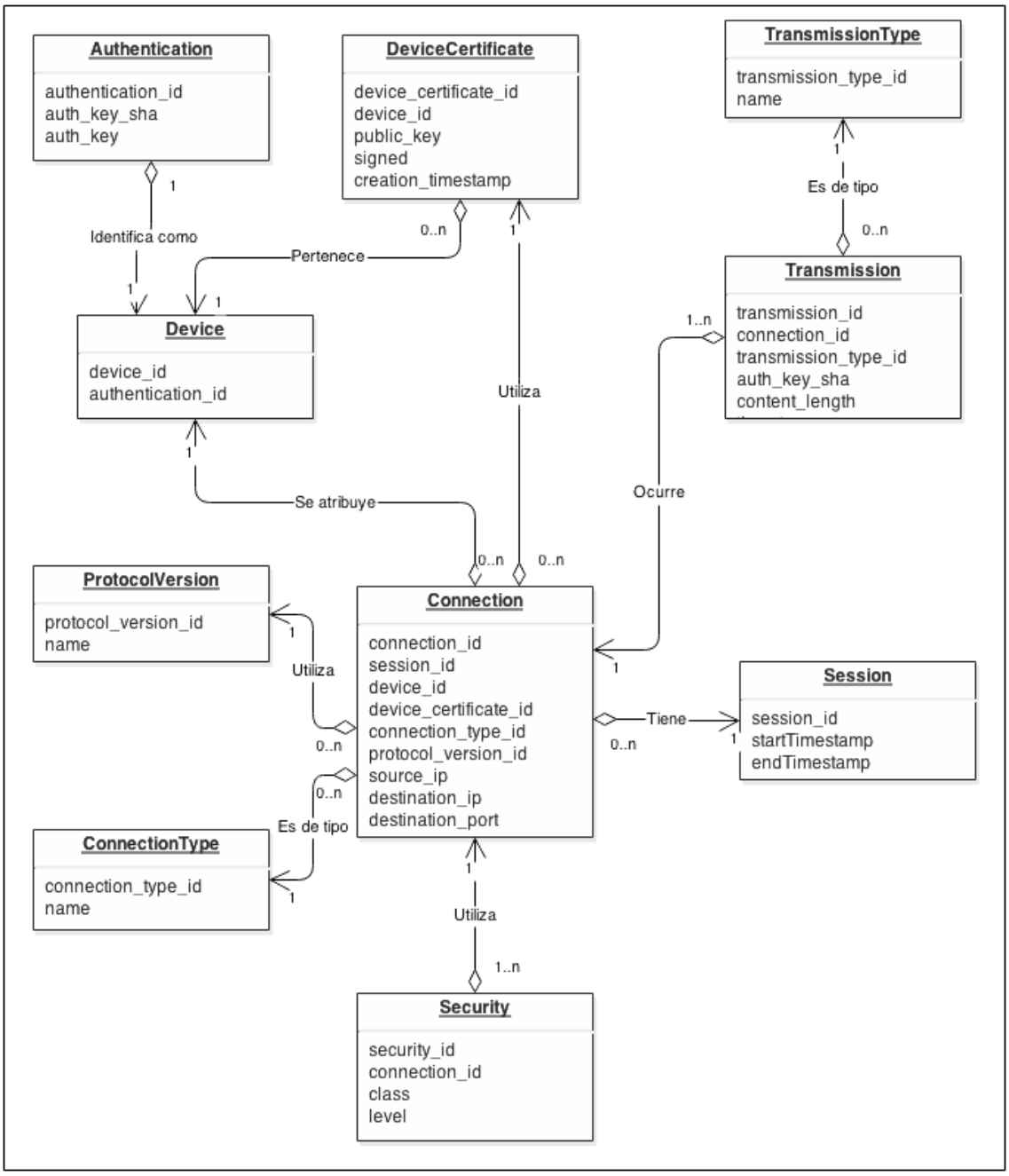

Ilustración 22 Diagrama relacional Base de datos Supernode Proxy

7.1.4.5 Main

La clase Main se encuentra definida en el paquete raíz de este módulo y define las funcionalidades a ejecutar nada más iniciar el programa. Su funcionamiento es sencillo, en primer lugar carga la configuración del archivo de configuración en memoria utilizando la clase R. En segundo lugar, inicia la escucha de la clase DaemonTCP al puerto definido en la configuración

como puerto de comunicación interno. Y por último inicia la escucha de la clase DaemonTCP al puerto definido en la configuración como puerto de comunicación externo. Dado que este módulo se caracteriza por tener múltiples hilos de ejecución, se ha representado el conjunto de estos hilos en la siguiente figura.

7.1.4.6 Dependencias

Este módulo depende de las librerías Fennec Library y sqlite jdbc versión 3.7.2. Fennec Library es utilizada para tener acceso a las funcionalidades propias del protocolo, del sistema de comunicación y securización del sistema desarrollado. Por otro lado, la libería sqlite jdbc versión 3.7.2 proporciona a este módulo las clases y métodos suficientes para para poder gestionar la base de datos sqlite interna.

8. PRUEBAS Y RESULTADOS

En este capítulo se muestran las pruebas realizadas para la validación de cada uno de los objetivos establecidos así como sus consiguientes resultados.

Durante la realización de las pruebas se ha creado una entidad certificadora propia que realiza la función de la entidad certificadora del sistema. Con esta entidad generada, se han firmado los certificados de los Supernodos utilizados durante las pruebas y los certificados de los Nodos con un modelo de seguridad A0. Del mismo modo, se ha entregado el certificado público de esta entidad a todos los Nodos y Supernodos para que sean capaces de verificar la validez de otros certificados. Para la puesta en producción de este sistema, esta entidad certificadora deberá ser sustituida por la entidad certificadora oficial del sistema.

Pruebas para verificar el cumplimiento de los objetivos

1. Desarrollar un módulo de cifrado de comunicaciones que se adapte a las diferentes capacidades de cómputo de los dispositivos que se comuniquen dentro del sistema.

Adaptación del modelo de seguridad

Prueba 1	
Descripción	Se ha especificado en la configuración de un Nodo el uso de un moldeo de seguridad A0 y se ha iniciado una solicitud de sesión con otro Nodo. Se ha repetido esta prueba para los modelos A1, A2 y B0.
Resultados	1. El Nodo adopta correctamente el modelo de seguridad y solicita al Supernodo el modelo adecuad A0 en este caso. 2. El Supernodo responde con un "SECURITY_RESULT 200 OK" durante la fase de negociación de seguridad. 3. Tanto el Nodo como el Supernodo han actuado conforme a lo esperado bajo el modelo de seguridad seleccionado.
Conclusión	Prueba finalizada con éxito.

Prueba 2	
Descripción	Se ha especificado en la configuración de un Nodo el uso de un moldeo de seguridad incorrecto y se ha iniciado una solicitud de sesión con otro Nodo.
Resultados	El Nodo rechaza el modelo de seguridad erróneo y asimila el modelo de seguridad por defecto B0
Conclusión	Prueba finalizada con éxito.

Prueba 3	
Descripción	Se ha forzado el envío del modelo incorrecto (C0 en este caso) en los parámetros de la negociación del protocolo.
Resultados	El Supernodo responde con un "SECURITY_RESULT 406 Not Acceptable"
Conclusión	Prueba finalizada con éxito.

Cifrado de la información

Prueba 1	
Descripción	Transmisión de un mensaje desde un Nodo con un modelo de seguridad B0 a uno con un modelo de seguridad A1. El mensaje transmitido ha sido la siguiente cadena de caracteres "Hola! Soy un mensaje de prueba". Se ha realizado una captura de tráfico para poder determinar la información obtenida por un posible atacante. Esta prueba se ha repetido con cada uno de los modelos de seguridad desarrollados.
Resultados	1. La información capturada durante la comunicación del Nodo de seguridad B0 al Supernodo está cifrada correctamente. 2. La información capturada durante la comunicación del Nodo de seguridad A1 al Supernodo está cifrada correctamente.
Conclusión	Prueba finalizada con éxito.

2. Implementar un mecanismo de que permita detectar alteraciones en la integridad de la comunicación.

Prueba 1	
Descripción	Transmisión de un mensaje desde un Nodo con un modelo de seguridad A0 a uno con un modelo de seguridad B0. El mensaje transmitido ha sido la siguiente cadena de caracteres "Hola! Soy un mensaje de prueba". Una vez cifrado y generado el mensaje completo se ha obtenido la parte que contiene el mensaje cifrado y se ha alterado para simular una alteración del mensaje por un atacante. Debido a que el contenido estaba cifrado no ha sido posible realizar una alteración coherente de los datos por lo que únicamente se ha alterado uno de los bytes que lo componían.
Resultados	El Supernodo ha respondido con un código de error 400 que determina un error "Sintaxis incorrecta, protocolo mal interpretado o cifrado incorrecto." debido a que el msg_key entregado no correspondía al contenido cifrado del mensaje dando como resultado un fallo durante el proceso de descifrado.
Conclusión	Prueba finalizada con éxito.

Prueba 2	
Descripción	Transmisión de un mensaje desde un Nodo con un modelo de seguridad A0 a uno con un modelo de seguridad B0. El mensaje transmitido ha sido la siguiente cadena de caracteres "Hola! Soy un mensaje de prueba". Este mensaje no ha sufrido alteración alguna durante la transmisión.
Resultados	El Supernodo ha descifrado correctamente el mensaje y lo ha redirigido al Nodo de destino.
Conclusión	Prueba finalizada con éxito.

3. Desarrollar un mecanismo que permita la autenticación inequívoca de cada uno de los dispositivos del sistema.

Verificación de la identidad de un Supernodo

Prueba 1	
Descripción	Se ha proporcionado a un Supernodo un certificado firmado por una entidad certificadora diferente a la especificada como oficial en la fase de pruebas. Se ha iniciado la comunicación con un Nodo con un modelo de seguridad A2.

Prueba 1	
Resultados	Durante la fase de verificación, el Nodo ha cerrado la sesión debido a que el proceso de verificación del certificado del Supernodo ha resultado erróneo.
Conclusión	Prueba finalizada con éxito.

Prueba 2	
Descripción	Se ha proporcionado a un Supernodo un certificado firmado por la entidad certificadora especificada como oficial en la fase de pruebas. Se ha iniciado la comunicación con un Nodo con un modelo de seguridad A2.
Resultados	La comunicación se ha establecido con éxito debido a que el proceso de verificación del certificado del Supernodo ha resultado correcto.
Conclusión	Prueba finalizada con éxito.

Verificación de la identidad de un Nodo

Prueba 1	
Descripción	Se ha proporcionado a un Nodo con un modelo de seguridad A0 un certificado firmado por una entidad certificadora diferente a la especificada como oficial en la fase de pruebas. Se ha iniciado la comunicación con un Supernodo.
Resultados	Durante la fase de verificación el Supernodo ha cerrado la sesión debido a que el proceso de verificación del certificado del Nodo ha dado un resultado negativo.
Conclusión	Prueba finalizada con éxito.

Prueba 2	
Descripción	Se ha proporcionado a un Nodo un certificado firmado por la entidad certificadora especificada como oficial en la fase de pruebas. Se ha iniciado la comunicación con un Supernodo.
Resultados	La comunicación se ha establecido con éxito debido a que el proceso de verificación del certificado del Nodo ha dado un resultado positivo.
Conclusión	Prueba finalizada con éxito.

Prueba 3	
Descripción	Para simular un atacante que ha obtenido el SHA del auth_key de otro Nodo durante el proceso de verificación, se ha proporcionado un SHA de un auth_key válido a un Nodo con un modelo de seguridad B0. Se ha iniciado la comunicación y se ha forzado al Nodo para enviar el SHA del Nodo ajeno.
Resultados	El proceso de autenticación se ha completado con éxito, no obstante, ninguno de los mensajes cifrados enviados por el Nodo han podido ser descifrados por el Supernodo por lo que nunca ha llegado a establecerse una concesión con el Nodo destino.
Conclusión	Prueba finalizada con éxito.

Prueba 4	
Descripción	Se ha proporcionado una clave de identificación auth_key falsa a un Nodo con modelo de seguridad B0 y se ha iniciado una comunicación con un Supernodo.
Resultados	Durante el proceso de autenticación, el Supernodo ha respondido con un código de error 401 que determina "Autenticación incorrecta."
Conclusión	Prueba finalizada con éxito.

Prueba 5	
Descripción	Se ha iniciado una comunicación con un Nodo con modelo de seguridad B0 y con una auth_key válida y asociada a él.
Resultados	La sesión se ha establecido correctamente debido a que el proceso de autenticación ha resultado correcto y los mensajes cifrados enviados han sido descifrados correctamente por el Supernodo.
Conclusión	Prueba finalizada con éxito.

Confidencialidad durante el registro de un Nodo

Prueba 1	
Descripción	Se ha simulado un ataque MITM, hombre en el medio, capturando toda la información transmitida durante la fase de registro de un Nodo con un modelo de seguridad A0.
Resultados	Se ha obtenido un mensaje cifrado que no ha sido posible descifrar debido a la falta de un elemento indispensable para su descifrado, la clave privada del certificado del Supernodo.
Conclusión	Prueba finalizada con éxito.

Prueba 2	
Descripción	Se ha simulado un ataque MITM, hombre en el medio, capturando toda la información transmitida durante la fase de registro de un Nodo con un modelo de seguridad B0.
Resultados	La información obtenida no permite generar la clave de identificación compartida por el Nodo y el Supernodo ya que con los datos capturados hará falta una gran cantidad de tiempo y capacidad de cómputo para deducirlo.
Conclusión	Prueba finalizada con éxito.

4. Desarrollar un mecanismo que registre toda acción acaecida dentro del sistema permitiendo un futuro procesado de la misma.

Prueba 1	
Descripción	Se ha establecido una sesión entre dos Nodos y se han realizado cuatro transmisiones antes del cierre de esta. Una vez finalizada la sesión, se ha accedido a la información de la base de datos SQLite almacenada en el Supernodo que ha gestionado la sesión con el programa SQLiteManager.
Resultados	Todos los datos asociados a esta sesión están almacenados y relacionados correctamente entre sí.
Conclusión	Prueba finalizada con éxito.

5. Desarrollar un mecanismo que garantice la alta disponibilidad del sistema.

Prueba 1	
Descripción	Se ha configurado un Supernodo para poder mantener en ejecución únicamente 2 sesiones. Se han establecido tres sesiones con dos Nodos, cada Nodo ha solicitado conexión para las tres sesiones y ha solicitado desconexión para una de ellas pasados 2 segundos.
Resultados	1. Las dos primeras sesiones se han establecido correctamente y sin ningún retardo. 2. Una vez pasados dos segundos, la tercera solicitud de sesión ha obtenido respuesta y se ha establecido correctamente.
Conclusión	Prueba finalizada con éxito.

Prueba 2	
Descripción	Se ha modificado el software de un Nodo para que solicite cien sesiones por segundo a un único Supernodo.
Resultados	Durante el ataque, las sesiones a las que el Supernodo daba Servicio no se han visto retardadas, alteradas ni anuladas.
Conclusión	Prueba finalizada con éxito.

6. Integrar todos los mecanismos de securización desarrollados en un sistema de comunicación.

Prueba 1	
Descripción	Se ha exportado cada uno de los módulos y se ha ejecutado.
Resultados	Todos los módulos hacían uso de todos los mecanismos de securización de forma correcta.
Conclusión	Prueba finalizada con éxito.

7. Desarrollar una herramienta que permita al software de terceros hacer uso del sistema desarrollado.

Prueba 1	
Descripción	Se han creado dos aplicaciones Java independientes al proyecto desarrollado. Ambas aplicaciones han añadido como dependencia la librería exportada Node Library. Una de las aplicaciones ha hecho uso de la clase ServerSession estableciendo como puerto de escucha el 1179 e implementando un método dentro del evento que permite hacerse cargo de la información recibida a través del sistema desarrollado. Este método permite decodificar el array de bytes recibido en texto y concatenarle, sea cual sea el texto recibido, la cadena de texto " qué tal?". Por otro lado, la otra aplicación se ha programado para utilizar la clase Session de la librería Node Library permitiéndole establecer una conexión con la dirección IP 127.0.0.1 y el puerto 1179. En la siguiente instrucción se ha hecho uso del método transmit y se ha introducido como parámetro el array de bytes perteneciente al texto "Hola!". Se ha programado esta aplicación para mostrar por pantalla el texto correspondiente a la decodificación del array de bytes recibido y finalmente hacer uso del método disconnect del objeto Session. Por último, se han ejecutado ambos programas junto con el resto de módulos del sistema en el entorno de desarrollo en el que se han programado.
Resultados	1. La conexión se ha realizado correctamente a través de todos los módulos. 2. El Nodo cliente ha transmitido correctamente la cadena de texto establecida. 3. El Nodo servidor ha recibido e interpretado correctamente la información. 4. El Nodo cliente ha mostrado por pantalla el texto correspondiente al array de bytes recibido en la respuesta, el mensaje mostrado por pantalla ha sido "Hola! qué tal?” 5. Tanto el Nodo cliente como el servidor se han desconectado correctamente.
Conclusión	Prueba finalizada con éxito.

9. TRABAJO FUTURO

El proyecto realizado constituye únicamente una primera aproximación a un sistema capaz de operar en un entorno real. Se trata de una prueba de concepto que ha permitido verificar la viabilidad del diseño elaborado, este proporciona una solución a la demanda de un esquema de abstracción que garantice los conceptos fundamentales de la seguridad de la información en entornos heterogéneos interoperables. Por lo tanto, existen mejoras que es necesario implementar para poder poner este sistema en producción en un entorno real. Para ello, se han categorizado los diferentes puntos de mejora en cuatro categorías diferentes:

9.1 SECURIZACIÓN

Se debe crear un mecanismo de securización para la comunicación entre los módulos Node Services y Node Library ya que en el sistema actual no lo contempla y supone un punto vulnerable del sistema que permite a atacantes internos, programas instalados dentro del Nodo, escuchar en claro y modificar el trafico transmitido entre estos dos módulos.

Por otra parte, el sistema actual carece de un mecanismo de verificación de identidad de un Supernodo para aquellos Nodos que haga uso de un modelo de seguridad de clase B o inferior. Esta limitación permite la suplantación de la identidad de los Supernodos dando cabida a ataques de hombre en el medio, MITM por sus siglas en inglés. Es necesario diseñar un mecanismo de verificación de Supernodos evitando el uso de certificados RSA ya que se trata de una clase de seguridad específicamente diseñada para Nodos que no pueden soportar este tipo de mecanismos.

El sistema actual identifica a cada nodo con una clave única pero no es capaz de determinar qué aplicación del Nodo es la que se está comunicando. Esto permite a aplicaciones maliciosas dañar la reputación del Nodo en el que se albergan, perjudicando así a otras aplicaciones del Nodo y al Nodo en general. Por ello, es necesario crear otro identificador para la aplicación que permita solucionar esta limitación. De esta forma, se podrá registrar las diferentes aplicaciones de la red asociándolas con la reputación de los Nodos a las que pertenecen. Esto permitirá sacar patrones de conducta de las mismas aplicaciones en otros Nodos y así determinar que aplicaciones son maliciosas pudiendo así bajar la reputación a una aplicación y no al Nodo en general.

9.2 NUEVAS FUNCIONALIDADES

El sistema actual no contempla la posibilidad de que un Nodo que utilice el sistema desarrollado pueda comunicarse a través de este sistema con otro elemento de la red que no lo implemente. Por lo tanto, es necesario modificar el módulo Supernode Proxy para que permita la comunicación con este tipo de elementos. Esto permitirá secularizar al menos una parte de la comunicación. No obstante, será posible indicar en la configuración de un Nodo que este rechace, para garantizar una mayor seguridad, todas las solicitudes de comunicación por parte de otro elemento de la red que no implemente el sistema.

Las sesiones del sistema únicamente permiten comunicaciones de uno a uno. Como trabajo futuro se propone actualizar el sistema para permitir a los Nodos realizar comunicaciones de tipo broadcast. No obstante, debido al poder que esto otorga al software malicioso, únicamente los Nodos con mejor reputación podrán realizar este tipo de comunicaciones.

9.3 ANALÍTICAS

Como ya se ha mencionado a lo largo de la memoria, en esta primera parte del proyecto únicamente se recopilan los datos generados en cada sesión, permitiendo así la generación de mecanismos de reputación y confianza. Por lo tanto, como tarea fundamental del trabajo futuro se propone desarrollar un modelo de reputación y confianza que permita determinar el grado de confianza de los Nodos de la red. Como punto de partida se propone establecer un nivel neutral de reputación para cada Nodo y en función de las reglas propuestas en el capítulo 6.2.3. No repudio variar este nivel.

Estableciendo el modelo de reputación y confianza como base, se propone atribuir, de forma indirecta y proporcional, la reputación de un Nodo a valores característicos del mismo como la red a la que pertenece, el certificado que utiliza y su dirección MAC y no únicamente a su clave identificadora, denominada auth_key. Esto se debe a que un nodo con baja reputación puede borrar su auth_key o generar un nuevo certificado para limpiar su identidad. Es por eso por lo que además de bajar la reputación de un nodo a través de su auth_key, es necesario reducir la reputación de (o ser más escéptico con) aquellos Nodos cuyas características sean similares a las de un Nodo de baja reputación, nunca por debajo del nivel neutral inicial para no perjudicar a otros Nodos.

Otra opción es la utilización de listas negras, que permita a los Nodos solicitar a los Supernodos que rechacen todas las peticiones de comunicación de otro Nodo hacia este.

El sistema actual trata el contenido de los mensajes como un conjunto de bytes. Los Supernodos desconocen el contenido de estos mensajes y únicamente deducen el mal comportamiento a través de las características de la comunicación. Se propone mejorar el sistema para permitir a los Supernodos ser capaces de interpretar el contenido de los mensajes transmitidos a través de él y así poder hacer un análisis más preciso de la actividad de los Nodos de la red. No obstante, dado que este desarrollo solo supondría un aumento de la eficacia de un sistema de analíticas

funcional, se propone desarrollar en último lugar dando más prioridad a otros desarrollos de trabajo futuro.

9.4 OPTIMIZACIÓN

Dentro de todas las tareas establecidas para el trabajo futuro consideramos que esta es en la que se debe poner un mayor énfasis. El sistema actual establece una base para el desarrollo de modelos de seguridad y proporciona un total de cuatro modelos diferentes. No obstante, si bien es cierto que estos modelos cumplen con los objetivos establecidos y permiten a una gran variedad de dispositivos de diferentes capacidades comunicarse de forma segura, todavía queda mucho margen de mejora a la hora de establecer modelos de seguridad que se adecúen a las capacidades de muchos elementos dentro de Internet of Things. Por lo tanto, se establece como tarea fundamental el estudio en profundidad de los límites, capacidades y recursos de todos los elementos que conforman Internet of Things y elaborar modelos de seguridad óptimos para estos.

Como ya se ha mencionado en la introducción del capítulo 7 Desarrollo, el lenguaje de programación en el que se ha desarrollado el sistema es Java debido a su versatilidad y facilidad de uso idóneo para el prototipado. No obstante, al tratarse de un lenguaje de alto nivel, es más ineficiente que otros lenguajes de bajo nivel. Por lo tanto, para mejorar la eficiencia del sistema, se propone desarrollar el sistema en un lenguaje de bajo nivel como C.

En el sistema actual, el software de cada Nodo que desee hacer uso de él debe adaptarse haciendo uso de la librería Node Library. Esta medida forma parte únicamente de la prueba de concepto ya que en un futuro desarrollo deberá eliminares esta dependencia para poder hacer más sencillo el uso del sistema de securización. Esto permitirá su fácil adaptación a la industria. Por lo tanto, se propone como trabajo futuro hacer este sistema transparente para todo el software que se ejecute dentro del Nodo adoptando un funcionamiento similar al de un proxy interno o cortafuegos.

Por último, aunque prestan un servicio fundamental, se propone estudiar la posibilidad de eliminar los Supernodos del sistema sin perder los beneficios que aportan. Esta tarea es de gran dificultad y supondría modificar por completo el diseño del sistema entero. No obstante, de lograrse, se reduciría en gran medida la dependencia del sistema de elementos externos a los Nodos, reduciendo el coste de implantación y aumentando la vellosidad de comunicación.

10. CONCLUSIONES

La fragmentación de la electrónica de consumo está dando lugar a un gran uso de dispositivos de capacidades muy diversas. Uno de los mayores obstáculos que debe superarse en este campo es la falta de mecanismos de seguridad y de privacidad en la comunicación entre estos dispositivos. Dado que las diferentes herramientas de seguridad de la información actuales no están diseñadas para operar en entornos heterogéneos y de recursos limitados, el mercado demanda la creación de un mecanismo que garantice los principios fundamentales de la seguridad de la información en la comunicación de estos dispositivos.

Para dar solución a este problema se ha desarrollado un sistema que garantiza que la información transmitida desde un dispositivo llegue al dispositivo correcto (autenticidad), que dicha información solo sea legible para dispositivo para el que iba destinada (confidencialidad), que la información recibida sea la misma que la transmitida (integridad), que toda acción quede registrada (no repudio) y que ninguna entidad maliciosa pueda inutilizar la red (disponibilidad).

Para llevar a cabo este proyecto se ha realizado una extensa y profusa actividad de documentación, tanto de artículos técnicos como de las investigaciones más avanzadas que, en este campo, se han realizado durante estos últimos años. Así, esta labor de recopilación de documentación forma una parte importante de la labor llevada a cabo durante este proyecto ya que se ha usado como base de conocimiento sobre la que se ha desarrollado el diseño del sistema.

Para ello se ha desarrollado una infraestructura de red compuesta por elementos denominados Nodos y Supernodos. Esta infraestructura permite a los Nodos, dispositivos de la red, comunicarse con otros Nodos a través de los Supernodos. Esto les permite establecer un canal seguro ofreciendo una tercera parte de confianza durante la comunicación entre dos nodos desconocidos entre sí.

De esta forma, esta infraestructura de comunicación permite tener absoluto control sobre todas las fases y aspectos de la comunicación de los Nodos del sistema permitiendo así el desarrollo de los mecanismos necesarios para garantizar los conceptos de seguridad de la información.

Para garantizar estos conceptos y debido a la naturaleza heterogénea de la red que se tiene como objetivo securizar, se ha desarrollado un conjunto de modelos de seguridad diseñados específicamente para garantizar los aspectos de autenticidad, confidencialidad e integridad dentro de una amplia gama de capacidades propias de los Nodos del sistema. Estos modelos

definen el comportamiento del Nodo durante las fases de verificación, autenticación, registro y cifrado de la información. En este proyecto se han diseñado las clases de seguridad A y B con un nivel de seguridad comprendido entre 0 y 2 para la clase A y un nivel de 0 para la clase B.

El modelo de seguridad de clase A está caracterizado por el uso de certificados tanto por parte del Nodo como del Supernodo.

El nivel 0 añade un proceso de verificación del Nodo a la fase de autenticación del modelo A1 haciendo uso de un certificado firmado por la entidad certificadora responsable del sistema que garantiza que el Nodo que lo posee ha pasado con éxito un análisis previo de seguridad.

El nivel 1 define la fase de autenticación a través de cifrado RSA.

El nivel 2 define el proceso de verificación permitiendo al Supernodo verificar su identidad a través de un certificado firmado por la entidad certificadora del sistema.

El modelo de seguridad de clase B tiene como tecnología principal el esquema de cifrado por bloques AES.

El nivel 0 define las técnicas para la implementación de las fases de registro, autenticación y cifrado de la información. Realiza la autenticación a través un proceso de generación de claves por Diffie-Hellman. Utiliza para ello una variación desarrollada en este proyecto del protocolo MTProto para las fases de autenticación y cifrado de la información.

Por otro lado, el concepto de disponibilidad es garantizado gracias a un mecanismo que permite a los Supernodos gestionar un número determinado de sesiones en ejecución, encolar futuras sesiones y redirigir o rechazar aquellas que no es capaz de encolar.

Por último, para garantizar el no repudio, se ha diseñado un mecanismo de registro de analíticas que permite al Supernodo recopilar todos los datos generados durante las comunicaciones a las que da servicio.

Por último, para llevar a cabo todos estos procesos, se ha desarrollado un protocolo de comunicación que permite el intercambio de información entre los diferentes módulos del sistema de manera eficiente.

Los resultados proporcionados en el capítulo x demuestran que se han cumplido todos los objetivos planteados al inicio del proyecto.

En este proyecto se ha elaborado un mecanismo de securización funcional para entornos heterogéneos. Se han establecido nuevos objetivos debido al amplio margen de mejora expuesto en el capítulo de 9 Trabajo Futuro. Gracias a la escalabilidad de su diseño será posible cumplir estos nuevos objetivos. Del mismo modo, gracias a este proyecto, se han sentado las bases para cubrir una necesidad cada vez más presente tanto en la sociedad como en la industria de proporcionar una solución a la falta de seguridad en las comunicaciones de dispositivos de capacidades reducidas.

La investigación realizada para este proyecto muestra la dificultad de crear un sistema que garantice el mismo nivel de seguridad en todos los dispositivos que conforman el Internet of Things debido a los pocos recursos de los que disponen. Sin embargo, debido a que esta tecnología se encuentra en su primera etapa de desarrollo, este proyecto demuestra que existe un gran abanico de posibilidades en esta materia. Aunque este proyecto ofrece un buen mecanismo de seguridad, la robustez de esta todavía es varias unidades de magnitud inferior a la proporcionada en dispositivos con grandes capacidades como pueden ser ordenadores de sobremesa o teléfonos móviles de alta gama. Para alcanzar este nivel, es necesario mucho trabajo de investigación y que las capacidades de estos dispositivos sigan creciendo. A medida que estas capacidades aumenten, el nivel de robustez de los sistemas de seguridad que puedan utilizar será mayor.

En cuanto a conclusiones de carácter personal, este proyecto me ha permitido conocer de cerca el funcionamiento de numerosos mecanismos de cifrado. Además, me ha servido como motivación para profundizar sobre el estado de la técnica de la seguridad en Internet of Things y sobre mecanismos de cifrado en sistemas que disponen de bajos recursos. Pero sobre todo me ha enseñado como afrontar un proyecto de grandes dimensiones y me ha otrogado una base, un punto de partida para afrontar este problema. Gracias a esta base puedo decir, que se trata de un área en la que podré contribuir, a través de futuras investigaciones y desarrollos, con nuevo conocimiento. Conocimiento que permitirá cubrir la necesidad, más vital cada día, de garantizar la seguridad y privacidad en las comunicaciones de los dispositivos que conforman Internet of Things.

11. PLANIFICACIÓN

En el siguiente capítulo se muestra la planificación de tareas establecida para la realización del proyecto. Las tareas establecidas parten como base de las fases planteadas para la metodología y constan a su vez de subtareas.

TAREA 1	**Estudio inicial de tecnologías existentes**
Objetivo	Llevar a cabo un proceso de documentación y análisis sobre el estado del arte de las tecnologías que abarca el ámbito del proyecto a través de la consulta de publicaciones de la comunidad científica en dicho ámbito.
Duración	14 días
Subtareas	1. Estudiar en profundidad las limitaciones de los diferentes tipos de dispositivos que conforman internet de las cosas. 2. Analizar los múltiples tipos de infraestructura de comunicación y seleccionar el que mejor se adapte a las necesidades de este proyecto. 3. Analizar los diferentes mecanismos de cifrado, especialmente de cifrado ligero, existentes en el estado del arte. 4. Estudiar los diferentes protocolos de comunicación segura presentes en el estado del arte. 5. Estudiar las diferentes técnicas de sistemas de alta disponibilidad.
Resultados esperados	Documentación que muestre el estado del arte de cada una de las tecnologías utilizadas. Selección de tecnologías apropiadas para el diseño el proyecto.

Tabla 34 Tarea 1 Planificación

TAREA 2	**Diseño del sistema**
Objetivo	Planificar con detalle los pasos a seguir y su modo de ejecución a la hora de desarrollar el proyecto
Duración	30 días

TAREA 2	Diseño del sistema
Subtareas	1. Diseñar una infraestructura de comunicación que permita tener control sobre todas las fases y aspectos de la comunicación. 2. Diseñar un conjunto de métodos de securización de la comunicación que se adapten a las capacidades de los dispositivos comunes en la red. 3. Diseñar un mecanismo de cifrado. 4. Diseñar un método de autenticación. 5. Diseñar un mecanismo que permita detectar la alteración no permitida de la información transmitida en el sistema. 6. Diseñar un sistema de registro de eventos. 7. Diseñar un mecanismo de balanceo de carga 8. Diseñar un protocolo de comunicación que permita el intercambio de información entre los diferentes dispositivos durante los procesos de securización y transmisión de la información. 9. Diseñar una arquitectura software que permite al desarrollo de un prototipo funcional.
Resultados esperados	Diagrama de bloques, diagrama de paquetes, diagrama de flujo, diagrama de casos de uso y diagrama de clases. Una primera versión del diseño que especifique todos los detalles del sistema que más tarde se desarrollará.

Tabla 35 Tarea 2 Planificación

TAREA 3	Desarrollo del sistema
Objetivo	Desarrollar la solución software que permita obtener un primer prototipo del sistema diseñado.
Duración	46 días
Subtareas	1. Desarrollar la infraestructura de red diseñada. 2. Desarrollar el mecanismo de balanceo de carga desarrollado. 3. Aplicar los diferentes modelos de seguridad desarrollados a la comunicación de los dispositivos haciendo uso del protocolo diseñado. 4. Desarrollar el mecanismo de registro de enveros diseñado. 5. Desarrollar una API que permita al software de terceros hacer uso del sistema desarrollado.
Resultados esperados	Un conjunto de módulos software que juntos permitan componer un sistema que cumpla con los requisitos expuestos en el alcance del proyecto. Una API que permita al software de terceros hacer uso del sistema desarrollado.

Tabla 36 Tarea 3 Planificación

TAREA 4	Despliegue
Objetivo	Desplegar el sistema desarrollado en un entorno de desarrollo.
Duración	2 días
Resultados esperados	Prototipo funcional del sistema.

Tabla 37 Tarea 4 Planificación

TAREA 5	Pruebas y validación
Objetivo	Validar el correcto funcionamiento del sistema desarrollado.
Duración	9 días
Resultados esperados	Prototipo funcional de un sistema que cumple todos los objetivos y requisitos establecidos en el apartado de Objetivos y alcance.

Tabla 38 Tarea 5 Planificación

TAREA 6	Documentación
Objetivo	Plasmar el conocimiento obtenido durante la elaboración del proyecto en una memoria para que futuros investigadores partan de esta base.
Duración	21 días

Tabla 39 Tarea 6 Planificación

A continuación se muestra un esquema en forma de diagrama de Gantt para la planificación de este proyecto. En él se muestra el tiempo de dedicación previsto para las diferentes tareas expuestas.

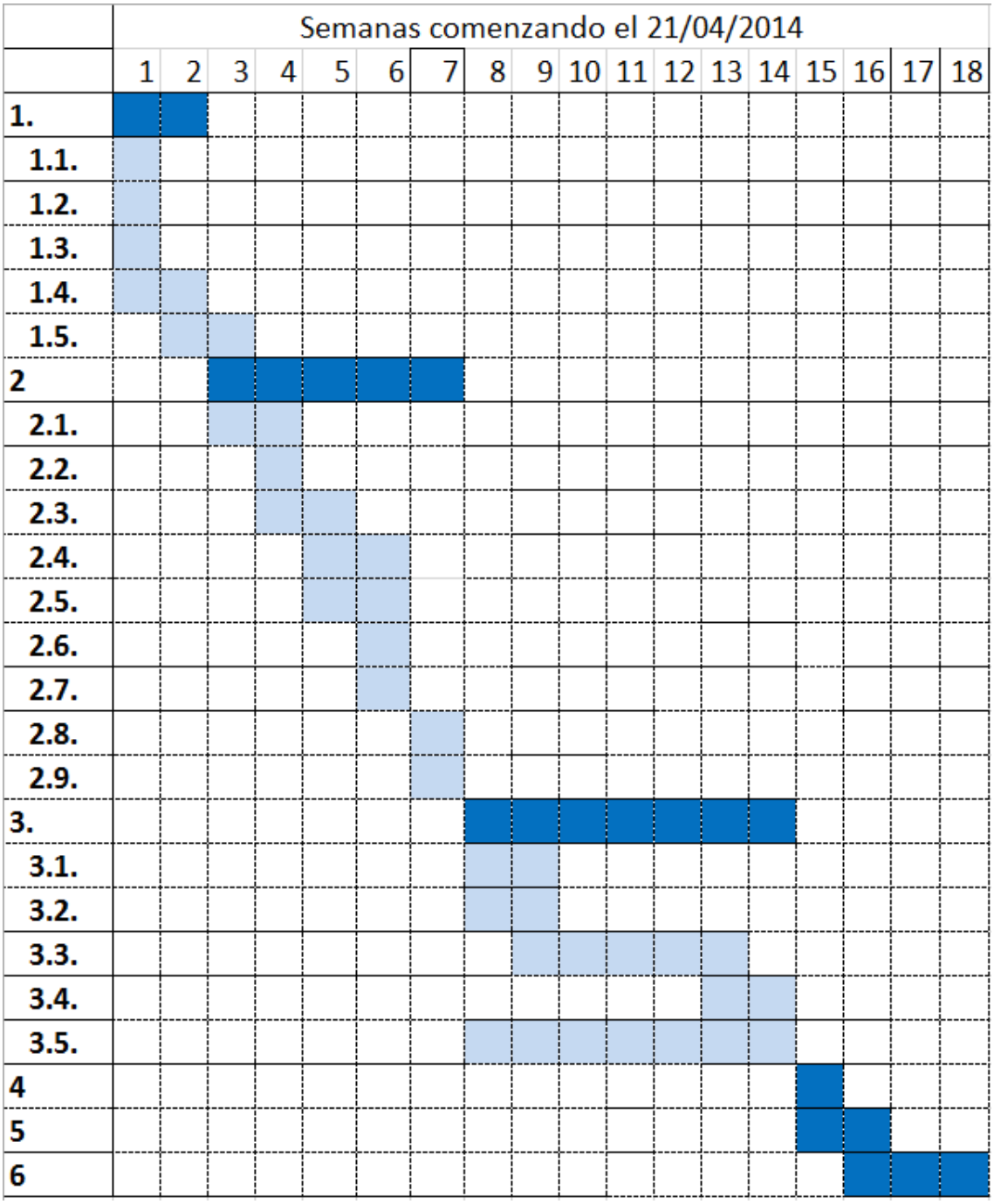

Ilustración 23 Diagrama de Gant de la planificación realizada

12. PRESUPUESTO

A continuación se desglosan las partidas presupuestarias del presente proyecto. Para una adecuada estimación de los costes se han dividido en tres grupos.

Gastos de personal

En primer lugar se muestran los gastos de personal. Este grupo determina el coste total en función del coste por hora de cada empleado multiplicado por el número de horas dedicadas. Como se puede ver, el investigador con cargo de jefe de proyecto, ha dedicado un total de 365 horas. Estas multiplicadas por un precio hora de 9,00 € dan un total de 3285 €.

GASTOS DE PERSONAL							
Nombre y Apellidos	DNI	Titulación	Categoría Profesional	Labor en el proyecto	2014		
					eur/h	horas	TOTAL
Iñigo Alonso Gonzalez	44346021C	Titulado Medio	Investigador	Jefe proyecto	9,00	365	3.285,00
SUBTOTAL						365	3.285,00

Ilustración 24 Gastos de personal presupuestados

Gastos de material

En segundo lugar se muestran los gastos de material. Este grupo de gastos representa el presupuesto establecido para los gastos del material adquirido para la realización del proyecto.

GASTOS DE MATERIAL		
Concepto	Cantidad	2014
Arduino Mega 2560	1	47,19
Xbee 1mW Trace Antenna	1	18,90
Raspberry Pi, Model B, 512 MB con Tarjeta SD de 8GB	1	38,40
IntelliJ IDEA 13.1 Community Edition	1	0,00
SQLiteManager	1	0,00
SUBTOTAL		104,49

Ilustración 25 Gastos de material presupuestados

Costes indirectos

En tercer lugar se muestran los gastos producidos por el uso de material previamente adquirido. En él se reflejan elementos como el coste de amortización del equipo informático, el coste d ella luz, limpieza de la oficina en la que se a desarrollado el proyecto, consumibles y el abono de transporte.

COSTES INDIRECTOS	
Concepto	2014
Informatico	240,00
Luz	190,00
Limpieza	160,00
Consumibles	90,00
Transporte	156,00
SUBTOTAL	836,00

Ilustración 26 Costes indirectos presupuestados

Total gastos

Por último se muestra la suma del total de todos los gastos mencionados dando como resultado la cantidad de euros presupuestada para el proyecto.

TOTAL GASTOS	
Concepto	2014
Gastos de personal	2.880,00
Gastos de material	104,49
Otros gastos	836,00
SUBTOTAL	4.225,49

Ilustración 27 Gastos totales presupuestados

13. BIBLIOGRAFÍA

1. *Bitbucket http://bitbucket.org/AlonsoApp/fennec*
   Accedido el 25 de Agosto de 2014

2. *Weiser, M.: The Computer for the 21st Century. Scientific American 265(9):66–75 (1991).*

3. *Gueron, Shay, and Michael E. Kounavis. "Intel® Carry-Less Multiplication Instruction and its Usage for Computing the GCM Mode." White Paper (2010).*

4. *Rivest, Ronald L., Adi Shamir, and Len Adleman. "A method for obtaining digital signatures and public-key cryptosystems." Communications of the ACM 21.2 (1978): 120-126.*

5. *Mahalanobis, Ayan. Diffie-Hellman Key Exchange Protocol, Its Generalization and Nilpotent Groups. Diss. Florida Atlantic University Boca Raton, Florida, 2005.*

6. *Mendel, Florian, et al. "Analysis of step-reduced SHA-256." Fast Software Encryption. Springer Berlin Heidelberg, 2006.*

7. *Polk, Tim, and Sean Turner. "Security challenges for the internet of things." Workshop on Interconnecting Smart Objects with the Internet, Prague. 2011.*

8. *Madlmayr, G.; Langer, J.; Kantner, C.; Scharinger, J.; Schaumuller-Bichl, I.; Risk Analysis of Over-the-Air Transactions in an NFC Ecosystem, Near Field Communication, 2009. NFC '09. First International Workshop on, 2009.*

9. *Tsutsui, A., Management Architecture and Distribution Framework for Home Network Services NGN Workshop, 2005*

10. *Akram, R. and Markantonakis, K. and Mayes, K., Application Management Framework in User Centric Smart Card Ownership Model, Information Security Applications, 2009.*

11. *Doctor Rolf H. Weber Internet of Things – New security and privacy challenges*

12. *El Internet de las Cosas - En un mundo conectado de objetos inteligentes Accenture – Fundación Bankinter*

13. *Cazorla, Mickaël, Kevin Marquet, and Marine Minier. "Survey and Benchmark of Lightweight Block Ciphers for Wireless Sensor Networks⋆." IDEA 64.128 (2013): 34.*

14. *Telegram http://telegram.org*
    Accedido el 25 de Agosto de 2014

15. *Telegram Core core.telegram.org/mtproto*
Accedido el 25 de Agosto de 2014

16. Nadeem, Aamer, and M. Younus Javed. "A performance comparison of data encryption algorithms." Information and communication technologies, 2005. ICICT 2005. First international conference on. IEEE, 2005.

17. Areitio, Javier. "Seguridad de la Información." Redes, Informática y Sistemas de Información. Editorial Paraninfo (2008).

18. Asociación Española para el Fomento de la Seguridad de la Información, ISMS Forum Spain

14. AGRADECIMIENTOS

La realización de este proyecto ha sido posible gracias a toda la experiencia en investigación, desarrollo de software y metodología de trabajo, adquirida a lo largo de los años que he pasado en el laboratorio s3lab de Deustotech. El equipo entero me ha apoyado siempre y me he sentido parte de él desde el primer día y es por ello que debo dar las gracias a Igor Santos por haberme proporcionado la oportunidad de formar parte de este gran equipo.

En concreto, en lo que respecta a este proyecto, he de destacar a Borja Sanz por la ayuda y el apoyo prestado en todo momento, a Javier Nieves por guiarme durante el desarrollo de este, de nuevo, a Igor Santos por su gran ayuda y a Xabier Ugarte por la motivación que me ha proporcionado.

Del mismo modo, quiero destacar la labor de asesoramiento recibida por los profesores Jonathan Ruiz e Iñaki Vázquez en los campos de computación ubicua y protocolos de comunicación.

Y por último, pero no por ello menos importante, debo agradecer a Pablo García Bringas su labor como director de este proyecto.

Gracias a todos.

15. ANEXOS

15.1 ANEXO 1

Confidencialidad

Es el requisito que intenta que la información privada o secreta no se revele a individuos no autorizados. La protección de la confidencialidad se aplica a los datos almacenados durante su procesamiento, mientras se transmiten y se encuentran en tránsito. Para muchas organizaciones, la confidencialidad se encuentra, frecuentemente, detrás de la disponibilidad y de la integridad, en términos de importancia. Para algunos sistemas y para tipos específicos de datos, como los autenticadores, la confidencialidad es de extrema importancia.[17]

Disponibilidad

Disponibilidad y accesibilidad de los sistemas y datos, sólo para su uso autorizado. Es un requisito necesario para garantizar que el sistema trabaje puntualmente, con prontitud y que no se deniegue el servicio a ningún usuario autorizado. La disponibilidad protege al sistema contra determinados problemas como los intentos deliberados o accidentales de realizar un borrado no autorizado de datos, de causar cualquier tipo de denegación del servicio o de acceso a los datos y de los intentos de utilizar el sistema o los datos para propósitos no autorizados. La disponibilidad, frecuentemente, es uno de los objetivos de seguridad más importante de toda organización.

Integridad

Se encarga de garantizar que la información del sistema no haya sido alterada por usuarios no autorizados, evitando la pérdida de consistencia. Presenta dos facetas:

- Integridad de datos. Es la propiedad de que los datos no hayan sido alterados de forma no autorizada, mientras se almacenan, procesan o transmiten.
- Integridad del sistema. Es la cualidad que posee un sistema cuando realiza la función deseada, de manera no deteriorada y libre de manipulación no autorizada. La integridad, normalmente, es el objetivo de seguridad más importante después de la disponibilidad.

Autenticidad

Es la propiedad que permite identificar el generador de la información. Por ejemplo al recibir un mensaje de alguien, estar seguro que es de ese alguien el que lo ha mandado, y no una tercera

persona haciéndose pasar por la otra (suplantación de identidad). En un sistema informático se suele conseguir este factor con el uso de cuentas de usuario y contraseñas de acceso.[18]

No repudio

Es la negativa de una entidad de sistema de haber estado involucrada en una asociación (especialmente en una asociación que implique transferencia de información) o de haber participado en dicha relación.

15.2 ANEXO 2

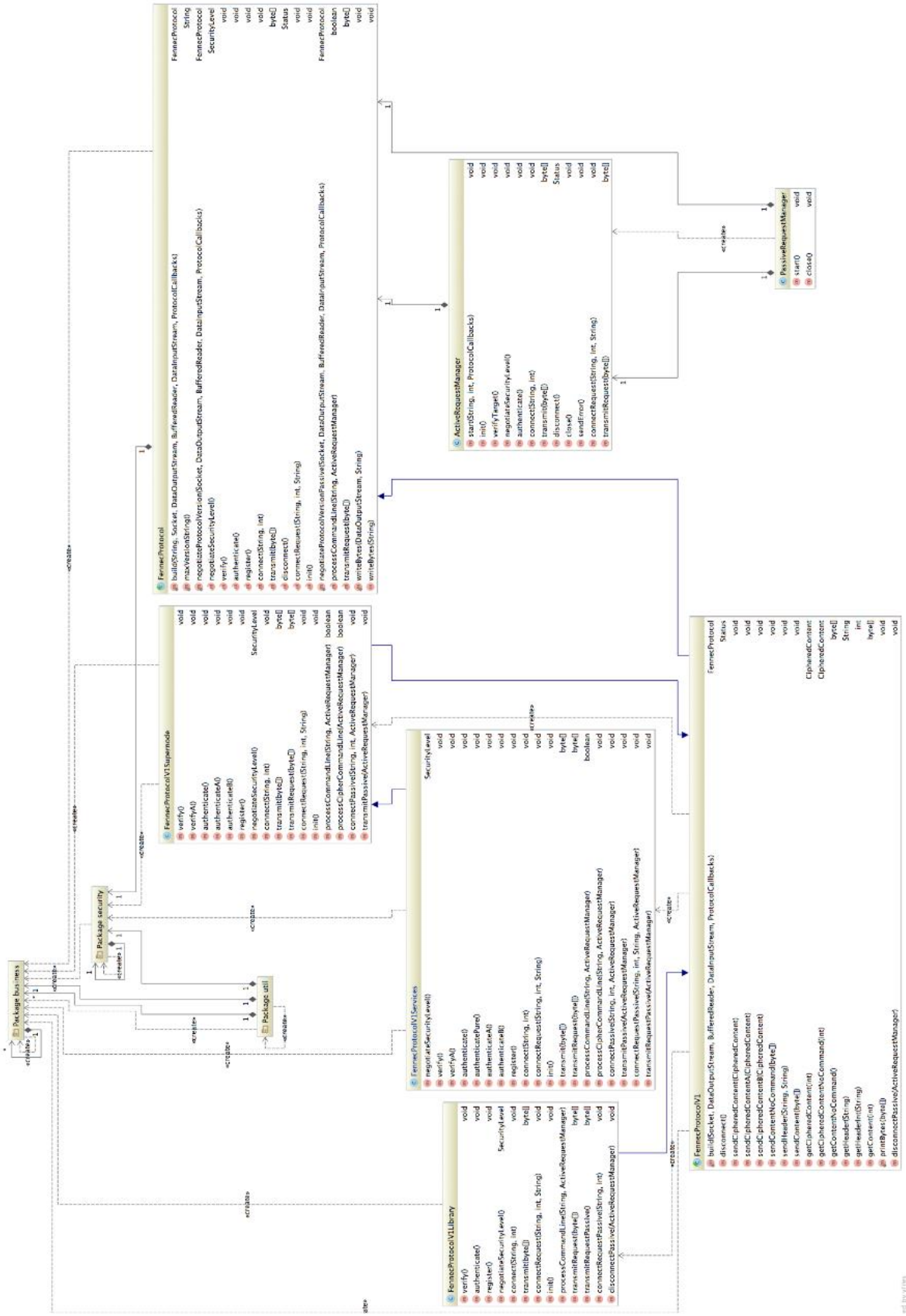

Ilustración 28 Diagrama de clases Fennec Library Parte 1

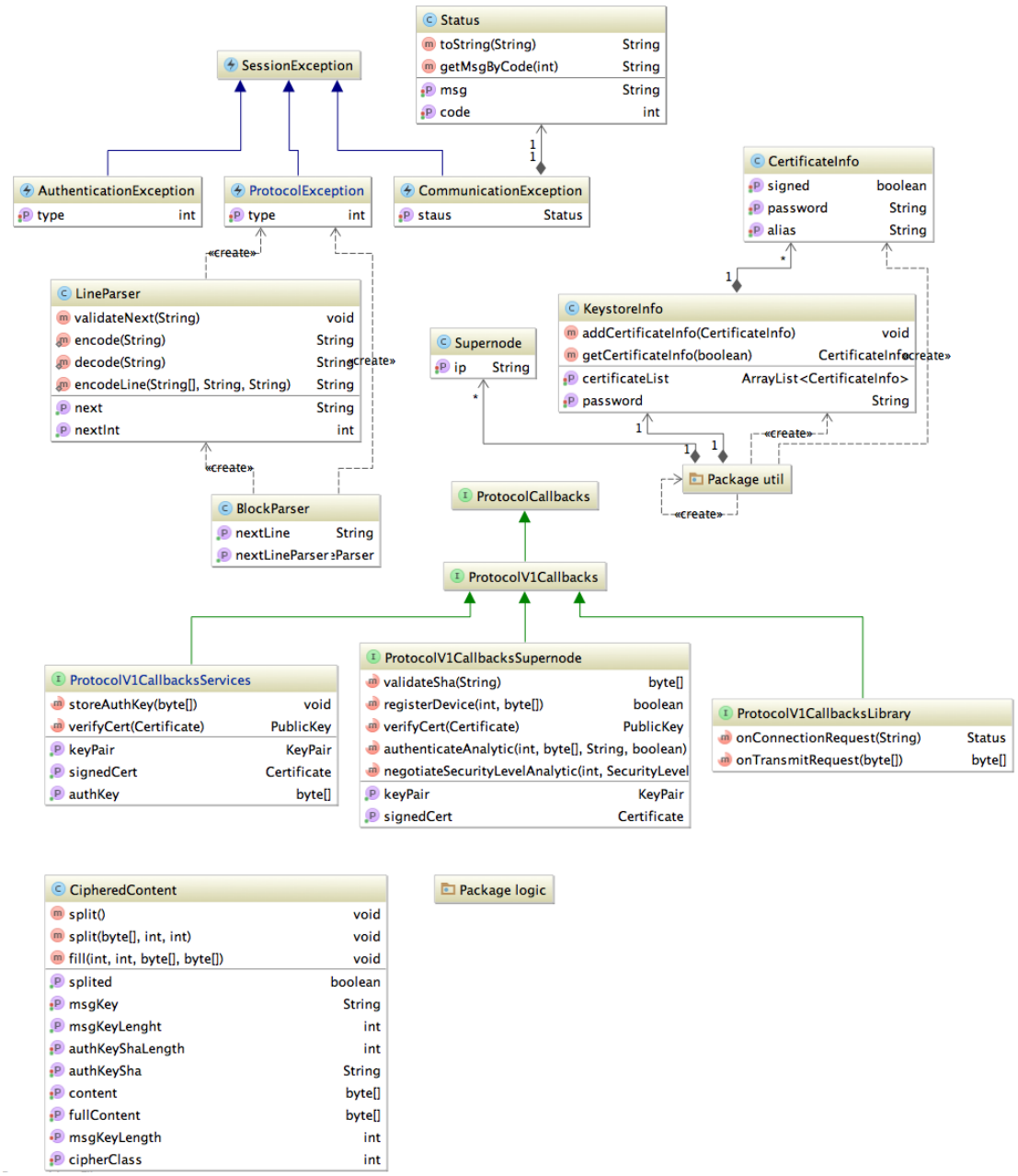

Ilustración 29 Diagrama de clases Fennec Library Parte 2

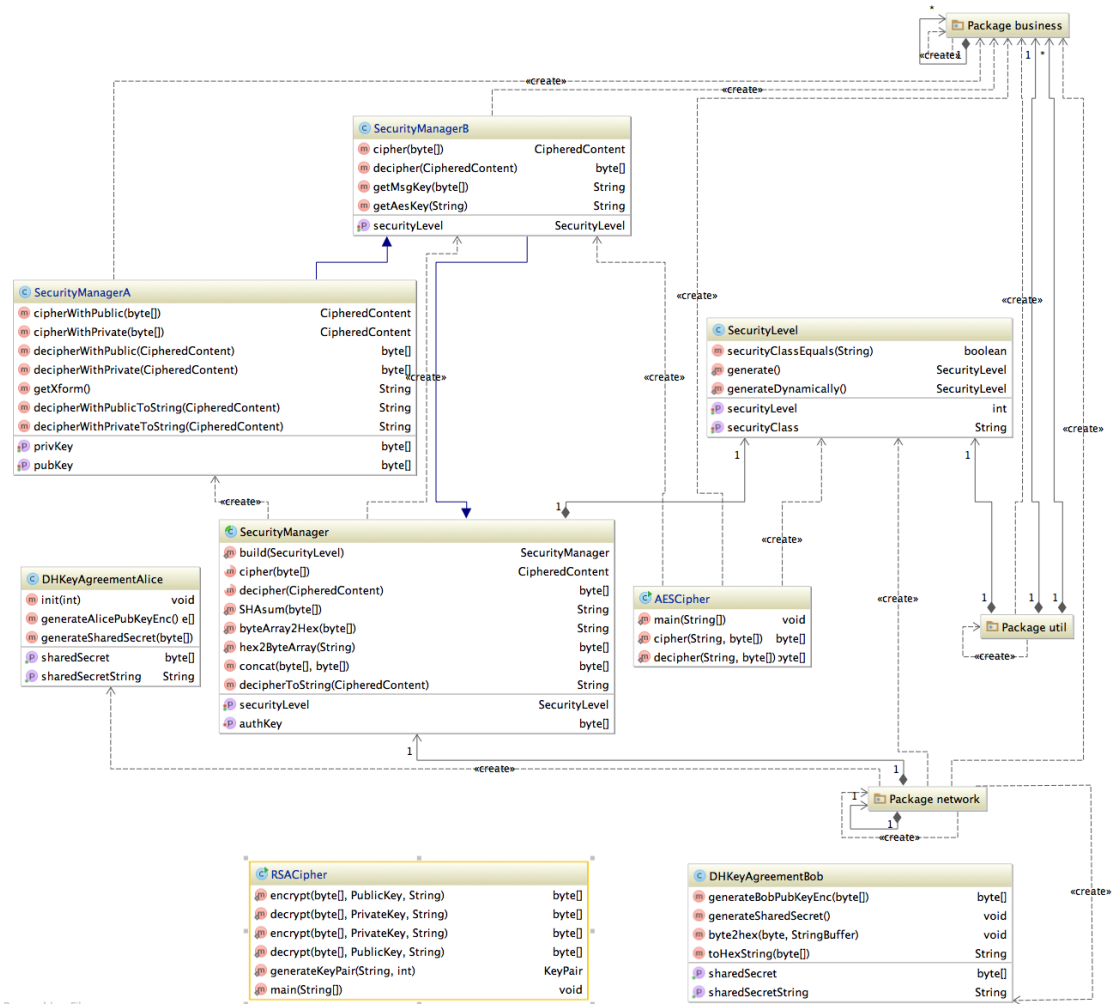

Ilustración 30 Diagrama de clases Fennec Library Parte 3