Facultad 5

Título: Sistema para la Seguridad de las Comunicaciones del SCADA “Guardián del ALBA”.

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Autor: Daisy Diana Vargas Vento

Tutor(es): Ing. Alejandro Manuel Rubinos Carvajal.

Ing. Enrique Reyes Bermúdez.

Ciudad de la Habana

2011

Declaración de Autoría

Declaro que soy el único autor de este trabajo y autorizo al Centro de Informática Industrial de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Daisy Diana Vargas Vento

_____

Ing. Alejandro Manuel Rubinos Carvajal

Ing. Enrique Reyes Bermúdez

_____

_____

Datos de contacto

Ing. Alejandro Manuel Rubinos Carvajal (amrubinos@uci.cu**)**

Graduado con el título de Ingeniero en Ciencias Informáticas en el 2009 en la Universidad de las Ciencias Informáticas. Tiene 4 años de experiencia en la especialidad y en el proyecto Guardián del ALBA.

Ing. Enrique Reyes Bermúdez (ereyes@uci.cu**)**

Graduado con el título de Ingeniero en Ciencias Informáticas en el 2009 en la Universidad de las Ciencias Informáticas. Tiene 3 años de experiencia en la especialidad y en el proyecto Guardián del ALBA.

Dedicatoria

Quiero dedicar esta tesis y con ella todos los triunfos de mi vida:

En primer lugar a mis padres por ser mi ejemplo, mi guía y por haberme dado todo ese cariño necesario para ser una buena persona.

A mis abuelitos que son la razón de todo lo que hago, todas las metas que me propongo en la vida son para merecer su orgullo.

A mi hermanito por quien siento un amor interminable.

Agradecimientos

- ✓ *A mi mamita por estar siempre ahí y tener un cariño infinito.*
- ✓ *A mi papá por su ejemplo y exigencias para ser mejor.*
- ✓ *A mis abuelos Lorgio, Sarah, Matilde y Oslidio los quiero muchísimo.*
- ✓ *A Normando por ayudarme siempre y a Tania por sus atenciones.*
- ✓ *A mi hermano, mis primas, mi tía Ana Elena, mi tío Osly por su apoyo incondicional y su confianza.*
- ✓ *A Gladys, mi tío Jorge, a toda mi familia, gracias.*
- ✓ *A mi novio David por brindarme su amor todos estos 5 años y seguirme amando a pesar de las dificultades.*
- ✓ *A mis amigos de siempre Jean Carlos, Dayné, Tony, Dunia y Yurisel por su paciencia para escucharme y darme sus excelentes consejos.*
- ✓ *A mis tutores por su apoyo y ayuda para convertirme finalmente en una Ingeniera.*
- ✓ *A mis abuelitos Allo y Nena por tantos años de cariño.*
- ✓ *A Danilo, Ignaís, Prevot, Sancho y todos aquellos que han creído en mí y me han abiertos sus puertas.*
- ✓ *A los amigos que nunca olvidaré Stanley, Omar, Fabio, Amílcar, Wendy, Adianis, Yune, Frank, Yoandra, Lisbeth , Kenia y tantos otros que aunque no los nombre saben que tienen un espacio en mi corazón.*
- ✓ *A la decana por ser la mejor de la universidad y enorgullecerme de ser miembro de la Facultad 5.*
- ✓ *A la Revolución por darme la oportunidad de realizar mis sueños.*

Resumen

El proyecto SCADA “Guardián del ALBA” para la automatización de la producción de petróleo de la República Bolivariana de Venezuela, constituye un sistema distribuido de supervisión, control y adquisición de datos que realiza el intercambio de información entre sus módulos a través de canales de comunicaciones en la red, sin mecanismos para la protección de los datos que se transmiten. Debido a la importancia de este sistema en tiempo real, que además constituye un eslabón fundamental en la economía del hermano país, sería sumamente perjudicial si ocurriera la modificación o espionaje de alguno de estos datos, por lo que es necesaria la creación de un mecanismo que asegure que la información transmitida no podrá ser adquirida, alterada o utilizada por alguna entidad ajena al sistema.

Para dar solución a dicho problema se realiza el siguiente trabajo de diploma que tiene como objetivo principal desarrollar un sistema para la seguridad de las comunicaciones del SCADA “Guardián del ALBA”, sobre una plataforma de software libre, capaz de proteger el intercambio de información entre sus módulos del ataque de intrusos. La solución está compuesta por dos subsistemas, un cliente y un servidor, encargados de garantizar la seguridad a través de un sistema criptográfico híbrido, que utiliza un algoritmo simétrico para el intercambio de información entre los módulos y un asimétrico para garantizar el intercambio seguro de las llaves de cifrado. Obteniéndose como resultado la primera versión de una biblioteca de cifrado para el proyecto Seguridad del Centro de Informática Industrial.

Palabras clave

Cifrado, Comunicaciones, SCADA, Seguridad, Sistema, Software Libre.

Summary

The SCADA “Guardián del ALBA” project for oil production automatization from the Bolivarian Republic of Venezuela is a distributed monitoring, control and data acquisition system based on the exchange of information between the modules via communication channels in the network, without mechanisms for the protection of the transmitted data. Due to the importance of this real-time system, which also constitutes an essential link in the neighboring countries economy, it would be extremely damaging if a tampering or intelligence leak of these data occurred, because of which it is necessary to create a mechanism that ensures that the information transmitted cannot be acquired, altered or used by any entity outside the system.

The main objective of this work is to develop a system to secure the communications of “Guardián del ALBA”, over a free software platform, able to protect the exchange of information between different modules from the attack of intruders. The solution consists of two subsystems, a client and a server, responsible for ensuring security through a hybrid cryptographic system that uses a symmetric algorithm for the exchange of information between the modules and an asymmetric for secure exchange of encryption keys. As the main result, we obtain the first version of a library of encryption for Industrial Computing Center (CEDIN).

Keywords:

Communication, Encryption, Free Software, SCADA, Security, System.

Índice de contenido

Índice de tablas

Índice de figuras

Introducción

La Seguridad Informática tiene como objetivo fundamental mantener la protección de los bienes informáticos, así sea datos, archivos o software, de la divulgación, manipulación o destrucción no autorizada. Los datos constituyen generalmente el activo informático más valorado, puesto que su pérdida, transformación o espionaje por parte de personas mal intencionadas puede ocasionar grandes daños, sobre todo si la información tiene un carácter urgente o sensible.

Este campo ha ido desarrollándose crecientemente a medida que han ido evolucionando las ciencias informáticas, pero tan rápida como ha sido su evolución ha sido también el crecimiento de la calidad y sofisticación de los ataques desarrollados por intrusos, que han generado formas cada vez más creativas de vulnerar la seguridad implantada. Este desarrollo ha alcanzado un grado tal, que ya no es necesario tener grandes conocimientos de computación o programación para penetrar un sistema informático; en lugares tan accesibles para todos como Internet, se pueden encontrar herramientas automatizadas que realizan todo el trabajo. Por lo tanto, tan rápido como se avanza en prestaciones y capacidad de respuesta ante fenómenos y problemas de todo tipo, también lo hacen aquellos que se dedican a encontrar huecos de seguridad en la arquitectura, diseño e implementación de programas, protocolos de comunicación y sistemas de software en sentido general, que les permitan violarlos de alguna forma.

La Universidad de las Ciencias Informáticas (UCI) es un proyecto de la Revolución Cubana cuya misión fundamental es formar profesionales comprometidos con su Patria y calificados en la rama de la Informática, a partir de un modelo pedagógico flexible, que vincula dinámica y coherentemente el estudio con la producción y la investigación, acorde con las necesidades sociales del país y de otros pueblos hermanos. (Rubinos Carvajal, 2009)

En la Universidad la producción se encuentra estructurada por centros productivos que integran la formación y la producción en torno a una temática para convertirla en

productos de software. En la Facultad 5 se encuentra el Centro de Informática Industrial (CEDIN) y entre sus proyectos el SCADA "Guardián del ALBA" o GALBA, como una colaboración entre la Universidad y empresas venezolanas. Este proyecto consta de varios módulos que en su conjunto conforman el sistema y que son desarrollados por distintas líneas de trabajo.

Durante todo el proceso de desarrollo del "Guardián del ALBA" se ha incluido entre los objetivos a alcanzar la seguridad del sistema ante ataques de usuarios maliciosos, logrando la reducción de vulnerabilidades con la implantación de mecanismos de seguridad que constituyen funcionalidades brindadas por un servidor de Seguridad implementado por la línea de desarrollo Seguridad del proyecto SCADA.

SCADA es un sistema distribuido que realiza la comunicación de sus módulos a través de la red y el intercambio de datos entre las distintas partes se realiza de forma plana, es decir, el texto que se transmite no está cifrado. Si alguien obtuviese acceso a esta red y lograse, pretendiendo ser parte del sistema, modificar o simplemente "escuchar" la información altamente sensible que se transmite, las consecuencias serían muy perjudiciales. En sistemas de este tipo, cuyo funcionamiento se realiza en tiempo real, la alteración del valor de una variable podría provocar que no se notificara la ocurrencia de una alarma crítica o la denotación de alguna cuando no lo amerita. (Rubinos Carvajal & Reyes Bermúdez, 2010) Todo lo antes mencionado constituye la **situación problémica** de esta investigación.

Es por esta razón que el presente Trabajo de Diploma se plantea como **problema científico** ¿Cómo lograr un intercambio de información de forma segura entre los módulos que componen el SCADA "Guardián del ALBA"?.

Esta investigación tiene como **objeto de estudio** los mecanismos de seguridad para la transmisión de datos.

El **campo de acción** queda enmarcado dentro del uso de sistemas de cifrado de datos en el intercambio de información a través del canal de comunicaciones del SCADA

"Guardián del ALBA".

Por tanto, se puede enunciar como **idea a defender**: El cifrado de los datos que se intercambian entre los distintos módulos del SCADA "Guardián del ALBA", incrementa la seguridad en el canal de comunicaciones del sistema.

Proponiendo como **objetivo general** de la investigación: Desarrollar un componente de cifrado de datos para incrementar la seguridad en el canal de comunicaciones del SCADA "Guardián del ALBA".

Con los siguientes **objetivos específicos**:

- Estudiar los mecanismos de seguridad de la información y las comunicaciones y cómo estos están siendo utilizados en los sistemas SCADA desarrollados en Cuba y el mundo.
- Estudiar el estado actual de los algoritmos y técnicas de cifrado de datos que gestionan el intercambio seguro de información.
- Modelar y diseñar un sistema para el cifrado de los datos que se intercambian entre los módulos del SCADA "Guardián del ALBA".
- Implementar, sobre una plataforma libre una herramienta que posibilite la comunicación segura del sistema SCADA "Guardián del ALBA".
- Probar los componentes del sistema implementado para validar su correcto funcionamiento.
- Documentar el sistema para la Seguridad de las Comunicaciones del SCADA "Guardián del ALBA" para que pueda servir de consulta durante la realización de posteriores versiones.

Conforme a este planteamiento se derivan las siguientes **tareas a desarrollar**:

- Estudio de las herramientas y métodos de seguridad en las comunicaciones de mayor uso a nivel mundial.
- Estudio de los métodos y operaciones existentes para la encriptación de datos.

- Selección de las tecnologías y herramientas libres a utilizar para el desarrollo del componente de cifrado.
- Análisis y diseño de los subsistemas que componen el sistema para la seguridad de las comunicaciones del SCADA "Guardián del ALBA".
- Implementación y prueba de los componentes que integran los subsistemas para la seguridad en las comunicaciones del SCADA "Guardián del ALBA", aplicando las normas internacionales que garantizan la calidad requerida del producto.

Los **Métodos de Investigación Científica** empleados son:

Métodos Teóricos:

1) **Análisis histórico y lógico:** Con el objetivo de analizar la situación actual sobre los mecanismos de cifrado de datos, su funcionamiento y principales características, así como seleccionar aquellos de más utilidad para la investigación.
2) **Análisis y Síntesis:** Para realizar el análisis y estudio de las bibliografías existentes sobre el tema en cuestión y lograr obtener de manera sintetizada el contenido necesario y suficiente para la realización del presente trabajo.
3) **Método Comparativo:** Este método resulta de gran utilidad para realizar la selección de los sistemas y algoritmos de cifrado a utilizar, así como las herramientas y tecnologías para el desarrollo de la solución.

Métodos Empíricos:

1) **Criterios de especialista:** Con el fin de consultar a las personas que poseen dominio sobre el tema, y que sus opiniones ayuden al perfeccionamiento de la investigación.

Al concluir este trabajo se espera obtener como **principales resultados**:

- Análisis, diseño e implementación de un sistema de cifrado de datos para el intercambio de información entre los diferentes módulos que conforman el

SCADA “Guardián del ALBA”.

- Primera versión de una biblioteca para ser integrada al proyecto Seguridad del Centro de Informática Industrial.

El Trabajo de Diploma está estructurado en 4 capítulos donde se detalla el proceso investigativo realizado:

- **Capítulo 1:** Fundamentación Teórica

  Describe el estado del arte actual. Se hace referencia a la seguridad en los sistemas SCADA y a los diferentes tipos y algoritmos de cifrado de datos existentes en el mundo y cuáles de ellos son utilizados en la realización de la solución.

- **Capítulo 2:** Definición de la solución y de las herramientas y tecnologías utilizadas.

  Se definen los requisitos con los que debe cumplir el sistema, así como la arquitectura sobre la cual se realizará el diseño e implementación de la solución. Se explican las metodologías, tecnologías y lenguajes seleccionados para ser utilizados en el desarrollo de la aplicación, tomando como principio la selección de herramientas libres.

- **Capítulo 3:** Descripción de la solución propuesta

  Se describe la modelación de la solución propuesta y el diseño del sistema. Se plantea el modelo de datos y se describe la forma es que son utilizados los diferentes patrones que enriquecen la arquitectura del sistema.

- **Capítulo 4:** Implementación y pruebas del sistema

  Se describe la implementación de la aplicación, así como la forma en que esta quedará integrada en el SCADA “Guardián del ALBA” y las pruebas realizadas a las funcionalidades desarrolladas. Además, se describen los estándares de codificación y documentación utilizados, destacándose que estos forman parte de los estándares aplicados por el Centro de Informática Industrial.

Capítulo 1: Fundamentación Teórica

1.1 Sistemas SCADA

Los SCADA son sistemas de control, supervisión y adquisición de datos empleados en la distribución de gas, electricidad, datos (comunicaciones), petróleo y sus derivados, etc. Se trata de sistemas que permiten supervisar desde una estación o red central lo que ocurre en distintos sitios estratégicos de una red de distribución y tomar acciones, automáticas o manuales, que permitan ajustar los parámetros del proceso controlado.

En la actualidad la mayoría de estos sistemas realizan sus funciones de forma distribuida, es decir, la ejecución no ocurre en una única estación de trabajo. Entre sus componentes se encuentra el encargado de gestionar la comunicación entre los diferentes nodos, con vista a garantizar el paso de mensajes y que el sistema responda como un servicio integrado.

1.1.1 Aplicaciones de los sistemas SCADA

Actualmente los SCADA son empleados en una gran variedad de aplicaciones en diversas industrias: pesadas, ligeras o de bienes (Figura 1).

Figura 1: Aplicaciones de los sistemas SCADA

1.1.2 SCADAS disponibles en el mundo y en Cuba.

En Cuba, se trabaja en Holguín en el desarrollo de sistemas SCADA para la línea del níquel en el control de exportaciones, con el Sistema de Supervisión y Control EROS, desarrollado en 1991 por especialistas de la "Unión del Níquel" de la provincia. EROS, es un sistema de supervisión y control de procesos industriales, en el que se concentran múltiples facilidades para operar y dirigir cualquier proceso productivo.

La Universidad de las Ciencias Informáticas, desarrolla sistemas SCADA, entre otras aplicaciones para:

- La Empresa de Telecomunicaciones de Cuba (ETECSA) con el objetivo de supervisar los equipos del departamento de Energía y Climatización.
- El Instituto de Meteorología para supervisar los parámetros de sus equipos para realizar cálculos predictivos acerca del comportamiento del tiempo.
- La Oficina del Historiador con el objetivo de lograr edificios inteligentes, capaces de supervisar los consumos de grandes corporaciones como hoteles y centros comerciales.
- Para las empresas productoras de petróleo (PDVSA) en la República Bolivariana de Venezuela, para el cual está siendo desarrollado específicamente este trabajo de diploma.

El grupo italiano Progea, ha desarrollado el Movicom, el cual representa la tercera generación innovadora de las plataformas industriales de supervisión y control. Este SCADA ofrece la posibilidad de realizar potentes y compactos sistemas de visualización para la interfaz hombre-máquina (HMI).

WinCC, es un sistema de supervisión sobre computadoras, ejecutable bajo Microsoft Windows 95 y Windows NT, desarrollado por la empresa Siemens. Está concebido para la visualización y el manejo de procesos, en líneas de fabricación de máquinas e instalaciones. (Organización Autómatas, 2009)

1.1.3 Seguridad en los sistemas SCADA.

Una de las características ignoradas durante años en los sistemas SCADA ha sido el tema de garantizar un sistema seguro ante ataques por parte de usuarios o entes malintencionados. Esto se debe principalmente a que se consideraban sistemas aislados, a los cuales no se tenía acceso desde la red de comunicación mundial. Esta característica, aunque garantizaba la seguridad ante ataques externos, los hacía de difícil actualización e incapaces de interactuar con otros sistemas homólogos y no los hacía invulnerables ante ataques internos.

Hoy en día es bastante común, que las redes SCADA estén mezcladas con las redes de las empresas sin ningún tipo de separación o control de acceso. Esta situación crea una puerta de enlace potencial entre ambas redes, ya que basta comprometer alguna máquina para poder acceder de una red a otra. Los atacantes no necesitan ser expertos en redes de control de procesos. Basta con atacar una máquina de la red local de la empresa que tenga acceso a la red SCADA, para poder causar daños.

Los sistemas SCADA se encuentran expuestos al mundo exterior y hay mucha información sobre ellos disponible, incluyendo especificaciones y vulnerabilidades, lo que los hace un blanco fácil para un ataque.

Los principales aspectos de seguridad en sistemas SCADA se pueden resumir en cuatro grandes grupos (Penín, 2007):

- **Características físicas:** Fiabilidad brindada por los equipos y su disponibilidad.
- **Sistemas Operativos:** El tener sistemas operativos considerados como estándares (Linux y Windows principalmente) conlleva a adoptar los riegos de los propios sistemas, de dominio público.
- **Comunicaciones:** Al utilizar protocolos abiertos se añaden más vulnerabilidades a los sistemas de control, pudiéndose acceder a las redes de información de estos sistemas a través de estos.
- **Aplicaciones:** La falta de medidas de seguridad (contraseñas, privilegios,

limitación de tiempo) hace que los sistemas sean vulnerables a amenazas, tales como virus o piratas informáticos.

Actualmente en el desarrollo de este tipo de sistemas se ha impulsado la creación de mecanismos que garanticen su seguridad ante usuarios maliciosos y las acciones dañinas que estos puedan llevar a cabo, utilizando métodos de identificación, autenticación y control de acceso de los usuarios a los recursos. Pero aún constituye una carencia de estos sistemas el hecho de que en muchos de ellos, no se han implementado medidas de seguridad básicas, tales como cifrado y redundancia; e incluso hay implementaciones cuya pila TCP/IP[1] es defectuosa, lo que los hace vulnerables a escaneos de red y a un sinfín de ataques plenamente conocidos. (S21sec, 2007)

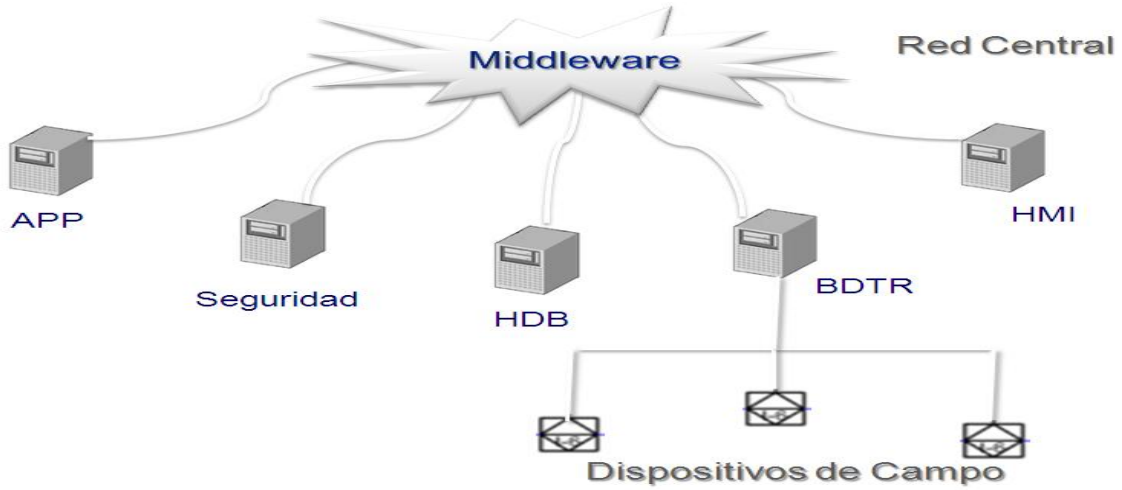
1.1.4 SCADA "Guardián del ALBA"

El SCADA "Guardián del ALBA", es un sistema distribuido para la producción de petróleo de la República Bolivariana de Venezuela. Este sistema está compuesto por varios módulos que realizan su comunicación a través de una infraestructura de comunicación llamada middleware; mientras la comunicación con los dispositivos de campo se realiza a través del servidor de base de datos en tiempo real (Figura 2).

La seguridad del sistema ante ataques de usuarios maliciosos ha constituido durante todo el proceso de desarrollo del "Guardián del ALBA" uno de los objetivos a cumplir. En este sentido se ha logrado reducir las vulnerabilidades del sistema con la implantación de mecanismos de autenticación, control de acceso, detección de intrusos e identificación de amenazas, todas estas, funcionalidades brindadas por un servidor de Seguridad implementado por la línea de desarrollo Seguridad del proyecto SCADA. Pero aunque se ha ganado seguridad en muchos sentidos, aún quedan aspectos por garantizar como la seguridad de la información intercambiada por los módulos a través del middleware de comunicaciones.

[1] TCP/IP opera a través del uso de una pila, la cual es la suma total de todos los protocolos necesarios para completar una transferencia de datos entre dos máquinas.

Figura 2: SCADA "Guardián del ALBA"



1.2 Seguridad de la información y las comunicaciones

Tanto la seguridad de la información como la seguridad de las comunicaciones tienen como objetivo fundamental garantizar tres aspectos:

- **Confidencialidad de la información:** Establece que la información o los activos informáticos sólo podrán ser accedidos por el personal autorizado.
- **Integridad de la información:** La información o los activos informáticos sólo podrán ser modificados por las personas autorizadas y de la forma autorizada.
- **Disponibilidad de la información:** La información o los activos informáticos deberán estar disponibles para el personal autorizado.

Se puede considerar una amenaza o ataque a la seguridad, la violación de cualquiera de los aspectos antes mencionados. Los ataques pueden servir a varios objetivos incluyendo fraude, extorsión, robo de información, venganza o simplemente el desafío de penetrar un sistema. Un ataque puede ser realizado por empleados internos que abusan de sus permisos de acceso, o por atacantes externos que acceden remotamente o interceptan el tráfico de red.

Las intervenciones o ataques a la seguridad de la información se pueden clasificar en

dos tipos: pasivos y activos. (Ramió, 2006)

Un interventor se dice que es **pasivo**, cuando su irrupción dentro del medio de comunicación se hace sin dejar rastro alguno de su existencia, lo cual no significa que este hecho no sea perjudicial, puesto que de esta forma se viola uno de los principios de la seguridad informática, la confidencialidad de la información.

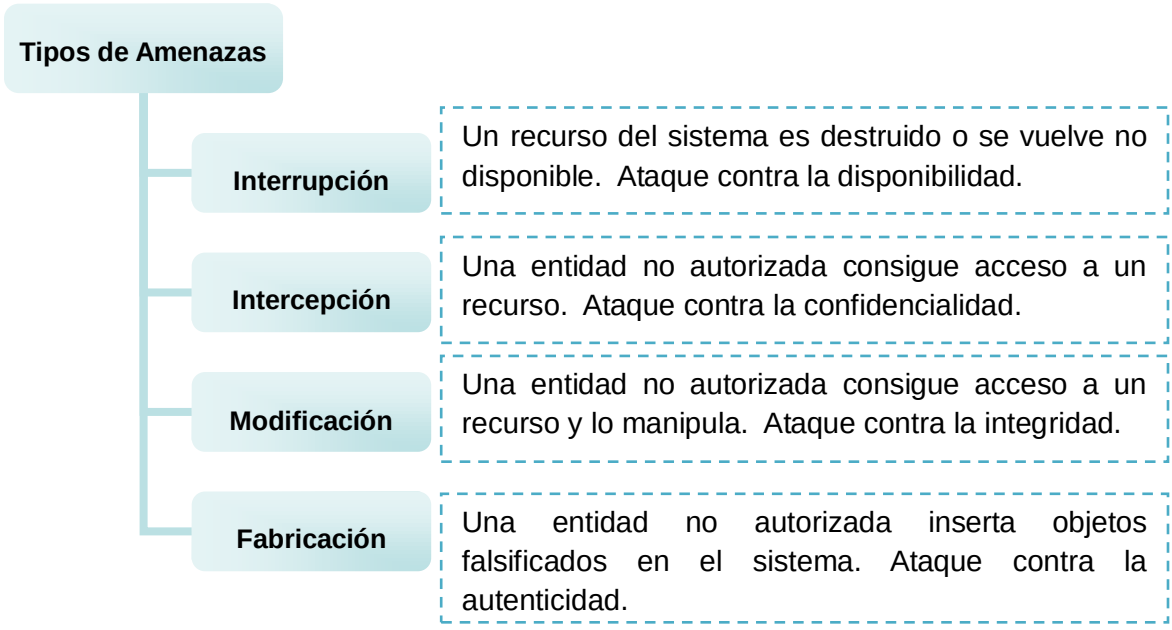
Las intervenciones **activas** son aquellas donde además de ganar el acceso a la información, se viola la integridad de los datos y se modifica o altera su contenido.

Ambos tipos de ataque pueden verse frustrados con técnicas de aseguramiento de la información como la criptografía.

1.2.1 Categorías generales de amenazas o ataques

Atendiendo a una clasificación por las características del ataque, existen cuatro categorías generales de amenazas, reflejadas en la siguiente figura. (Figura 3). (Vega Cutiño, 2010):

Figura 3: Categorías generales de amenazas



1.2.2 Mecanismos de seguridad

No existe un único mecanismo capaz de garantizar totalmente la seguridad en la información y las comunicaciones, pero la mayoría de ellos hacen uso de técnicas criptográficas basadas en el cifrado de la información. Los más importantes son los siguientes (Veyrat Marqués, 2007):

- **Intercambio de autenticación**: Corrobora que una entidad, ya sea origen o destino de la información, es la deseada.
- **Cifrado:** Garantiza que la información no sea inteligible para individuos, entidades o procesos no autorizados.
- **Firma digital:** Garantiza la integridad del mensaje y autenticación del emisor. Juega un papel esencial en el servicio de no repudio.
- **Integridad de datos:** Este mecanismo implica el cifrado de una cadena comprimida de datos a transmitir, llamada generalmente valor de comprobación de integridad.
- **Control de acceso:** Esfuerzo para que sólo aquellos usuarios autorizados accedan a los recursos del sistema o a la red.
- **Tráfico de relleno:** Consiste en enviar tráfico espurio[2] junto con los datos válidos para que el atacante no sepa si se está enviando información, ni qué cantidad de datos útiles se está transmitiendo.
- **Control de encaminamiento:** Permite enviar información por determinadas zonas consideradas clasificadas. Asimismo posibilita solicitar otras rutas, en caso que se detecten persistentes violaciones de integridad en una ruta determinada.
- **Unicidad:** Consiste en añadir a los datos un número de secuencia, la fecha y hora, un número aleatorio, o alguna combinación de los anteriores. De esta forma se evitan amenazas como la reactuación o resecuenciación de mensajes.

[2] Tráfico falso.

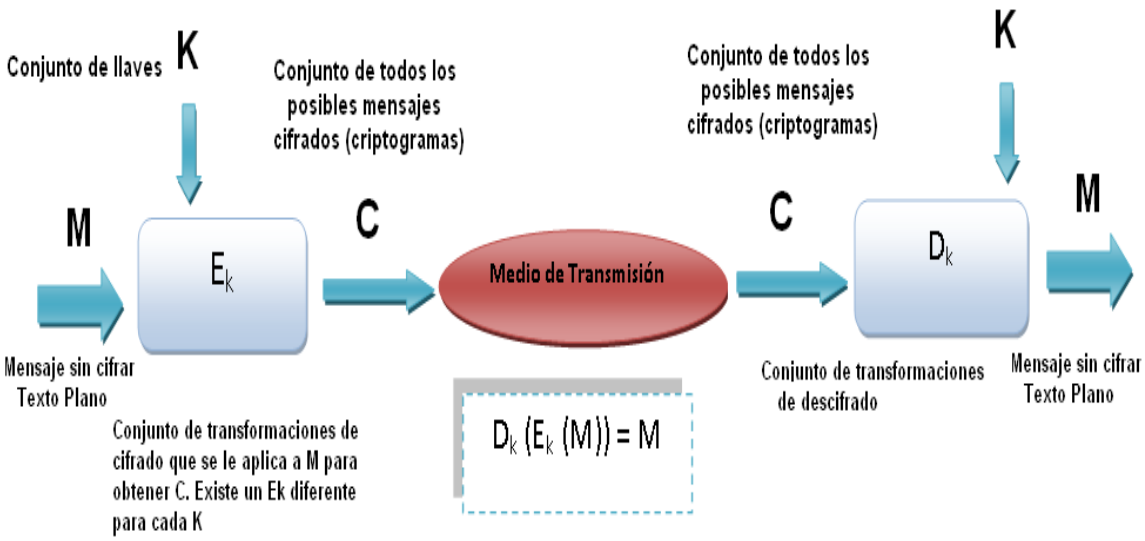
La protección de la información utilizando como mecanismo de seguridad el cifrado de datos, permitirá el acceso a la información sólo al personal autorizado, garantizando de esta forma la confidencialidad de los datos que se intercambian entre los módulos que componen el SCADA "Guardián del ALBA".

1.3 Criptografía

El término criptología (del griego krypto: lo oculto, lo escondido y logos: estudio) es el estudio de los criptosistemas. Sus áreas de investigación principales son la criptografía y el criptoanálisis, pero también se incluye la esteganografía[3] como parte de esta ciencia aplicada. La criptografía es el estudio de técnicas de cifrado seguras, mientras el criptoanálisis es el estudio de las técnicas orientadas a hallar vulnerabilidades en los sistemas de cifrado. (López López, 2005)

Se define un criptosistema como una quíntupla (M, C, K, E, D) (Figura 4), donde:

Figura 4: Quíntupla que define un criptosistema



La criptografía puede definirse como una rama inicial de las Matemáticas y en la

[3] Ocultación de la verdadera información dentro de algo que parezca completamente inofensivo.

actualidad también de la Informática y la Telemática, que hace uso de métodos y técnicas con el objeto principal de cifrar, y por tanto proteger, un mensaje o archivo por medio de un algoritmo, usando una o más claves. (Ramió, 2006)

Su aplicación resuelve problemas de seguridad como la confidencialidad o privacidad de la información y su integridad, pero también permite confirmar quien es realmente el autor del mensaje recibido y la seguridad de que este no ha sido alterado (autenticación), así como que no se pueda negar su autoría (no repudio).

El procedimiento utilizado para cifrar datos se realiza por medio de un algoritmo, al cual se le puede considerar como una función matemática. Por lo tanto, un algoritmo de cifrado es una fórmula para desordenar una información de manera que ésta se transforme en incomprensible, usando un código o clave (en ocasiones, más de una). Los mensajes que se tienen que proteger, denominados texto claro o texto plano, se transforman mediante esta función, y a la salida del proceso se obtiene el texto cifrado o cifrograma. (Pastor Franco & Sarasa López, 2005)

En muchos casos existe un algoritmo de descifrado encargado de reordenar la información y volverla inteligible, pero no siempre es así. Cuando existen ambas funciones, una para cifrar y otra para descifrar, se dice que el sistema criptográfico es de dos vías o reversible, mientras que cuando no existe una función para descifrar, se dice que el sistema es de una sola vía o irreversible.

1.3.1 Tipos de criptografía

La criptografía se divide en clásica y moderna de forma poco formal. Dos métodos engloban lo más relevante de la criptografía clásica:

La **sustitución,** consistente en cambiar los caracteres componentes del mensaje original en otros según una regla determinada de posición natural en el alfabeto.

La **transposición,** consistente en cambiar los caracteres componentes del mensaje original en otros según una regla determinada de posición en el orden del mensaje.

Prácticamente todos los algoritmos de cifrado se basan en uno de esto métodos o en combinaciones de ambos. Sin embargo, estos sistemas son vulnerables al criptoanálisis y presentan una dificultad en cuanto a la relación complejidad/longitud de la llave y el tiempo necesario para cifrar y descifrar el mensaje.

En la criptografía moderna, las llaves de cifrado constituyen su base fundamental. Estas nuevas técnicas de cifrado de datos cuentan con una serie de elementos que las hacen superior a las técnicas llamadas clásicas.

Un criptosistema que haga uso de claves criptográficamente débiles será él mismo débil. La gestión de claves abarca generación, distribución, almacenamiento, tiempo de vida, destrucción y aplicación de las claves de acuerdo con una política de seguridad.

1.3.2 Ventajas de la criptografía moderna

Con la aparición de las computadoras se dispone de una potencia de cálculo muy superior a la presente en los métodos clásicos, lo cual constituye la primera ventaja, mayor **velocidad de cálculo**.

Estos nuevos métodos de la criptografía moderna constituyen sistemas criptográficos **más estables y seguros** debido a los avances alcanzados por las ciencias matemáticas.

Surgieron muchas actividades nuevas que precisaban la ocultación de datos, con lo que la criptografía experimentó un **fuerte avance debido a las nuevas necesidades de seguridad**.

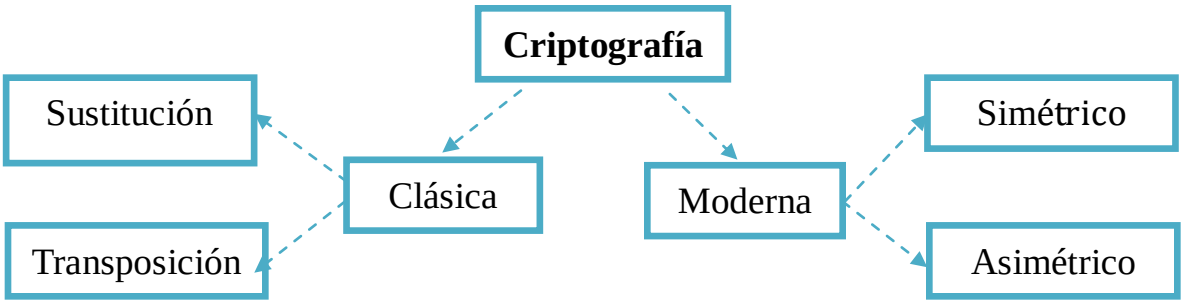
1.3.2.1 Tipos de criptografía moderna

Los sistemas criptográficos modernos se clasifican en dos tipos o familias generales, los simétricos o de llave única, también conocidos como de llave privada, y los sistemas asimétricos o de llave pública. Esta clasificación es realizada según el tipo de claves que utilizan estos sistemas para realizar las operaciones de cifrado y descifrado.

(Ramió, 2006)

En el siguiente gráfico (Figura 5) se puede observar un resumen de las distintas clasificaciones definidas hasta el momento dentro de la criptografía.

Figura 5: Clasificaciones dentro de la criptografía



Según el tratamiento que se le da al mensaje para realizar las operaciones de cifrado y descifrado, los criptosistemas simétricos también pueden ser clasificados en cifradores de bloque (dividen el mensaje en bloques de 64 ó 128 bits) o cifradores de flujo (realizan las operaciones bit a bit). (Ramió, 2006)

1.3.3 Requisitos de seguridad de los algoritmos criptográficos

Los algoritmos criptográficos deben cumplir con una serie de requisitos que garanticen la efectividad y el cumplimiento de los objetivos que se persiguen con su aplicación, estos requisitos son: (Ramió, 2006)

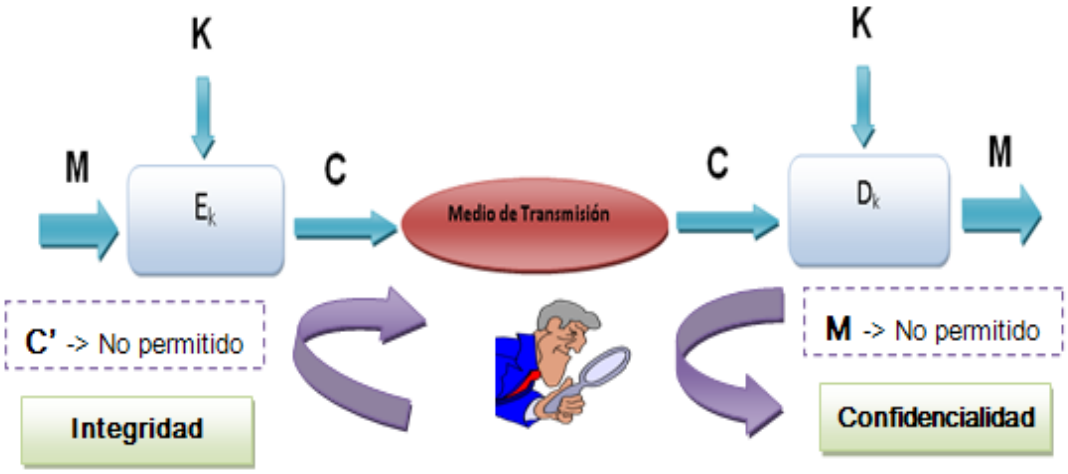
- El algoritmo de cifrado y descifrado deberá ser rápido y fiable.
- Debe ser posible transmitir ficheros por una línea de datos, almacenarlos o transferirlos.
- No debe existir retardo debido al cifrado o descifrado.
- La seguridad del sistema deberá residir solamente en el secreto de la clave y no en las funciones de cifrado.
- La fortaleza del sistema se entenderá como la imposibilidad computacional (tiempo de cálculo en años que excede cualquier valor razonable) de romper el

cifrado o encontrar una clave secreta a partir de otros datos de carácter público.

1.3.4 Cifrado simétrico

En el cifrado simétrico o de clave privada, se utiliza la misma clave para las operaciones de cifrado y descifrado y solamente aquellos entes que toman parte en la comunicación tienen conocimiento de esta. La seguridad de estos sistemas reside mayormente en mantener dicha clave en secreto.

Figura 6: Cifrado simétrico



Estos sistemas cifran bloques de texto del mensaje original y son más sencillos que los sistemas de llave pública, por lo que sus procesos de cifrado y descifrado son más rápidos y resultan apropiados para el cifrado de grandes volúmenes de datos. Su utilización garantiza la integridad y confidencialidad del mensaje a transmitir. Además, con claves de sólo unas centenas de bits obtendremos una alta seguridad pues la no linealidad del algoritmo hace que en la práctica el único ataque factible sea por fuerza bruta.

Las principales desventajas de los métodos simétricos son la distribución de las llaves, el peligro de que muchas personas deban conocer una misma llave y la dificultad de almacenar y proteger muchas llaves diferentes.

Todos los algoritmos de cifrado clásicos se pueden considerar simétricos, pero los principales algoritmos de este tipo en la actualidad son DES, TDES, AES e IDEA.

DES (Data Encryption Standard): Fue desarrollado como un estándar de cifrado de datos en 1997 por IBM, aunque actualmente ya no constituye un estándar. Fue el algoritmo más utilizado en el mundo hasta la década del 90. Utiliza una llave simétrica de 64 bits, de los cuales 56 son usados para el cifrado, mientras que los 8 restantes son de paridad, y se usan para la detección de errores en el proceso.

TDES (3DES, Triple-DES): Constituye una variante de DES, basado en su uso tres veces, para solventar el problema de la corta longitud de su llave. Normalmente se utiliza una secuencia de cifrado−descifrado−cifrado con tres claves diferentes y no relacionadas entre sí, con lo que se logra longitudes de llave de 192 bits, de los cuales 168 son efectivos.

AES (Advanced Encryption Standard): También conocido como Rijndael, es un esquema de cifrado por bloques destinado a remplazar al DES como estándar. AES tiene un tamaño de bloque fijo de 128 bits y tamaños de llave de 128, 192 ó 256 bits. La mayoría de los cálculos de este algoritmo se realizan en un campo finito determinado. Opera en una matriz de 4×4 bytes, llamada state. Este algoritmo es uno de los más populares utilizados en criptografía simétrica.

IDEA (International Data Encryption Algorithm): Es un cifrador por bloques diseñado por Xuejia Lai y James L. Massey de la Escuela Politécnica Federal de Zúrich y descrito por primera vez en 1991. Cifra bloques de texto de 64 bits, en 8 rondas, operando siempre con números de 16 bits. Utiliza operaciones como XOR, suma y multiplicación de enteros.

1.3.4.1 Comparación de algoritmos simétricos

Para realizar una comparación entre los principales algoritmos simétricos en la actualidad y poder establecer cual nos brindaría una óptima seguridad en el cifrado de las comunicaciones del SCADA “Guardián del ALBA”, se deben tener en cuenta

elementos como: potencialidad de la llave para las operaciones de cifrado y descifrado, velocidad con la que realizan estas operaciones y las debilidades que se han hecho públicas de dichos algoritmos y que pudieran afectar o no la seguridad del sistema.

La velocidad de cifrado de los algoritmos contiene valores indicativos sólo para una comparación entre algoritmos.

Tabla 1: Comparación entre algoritmos simétricos

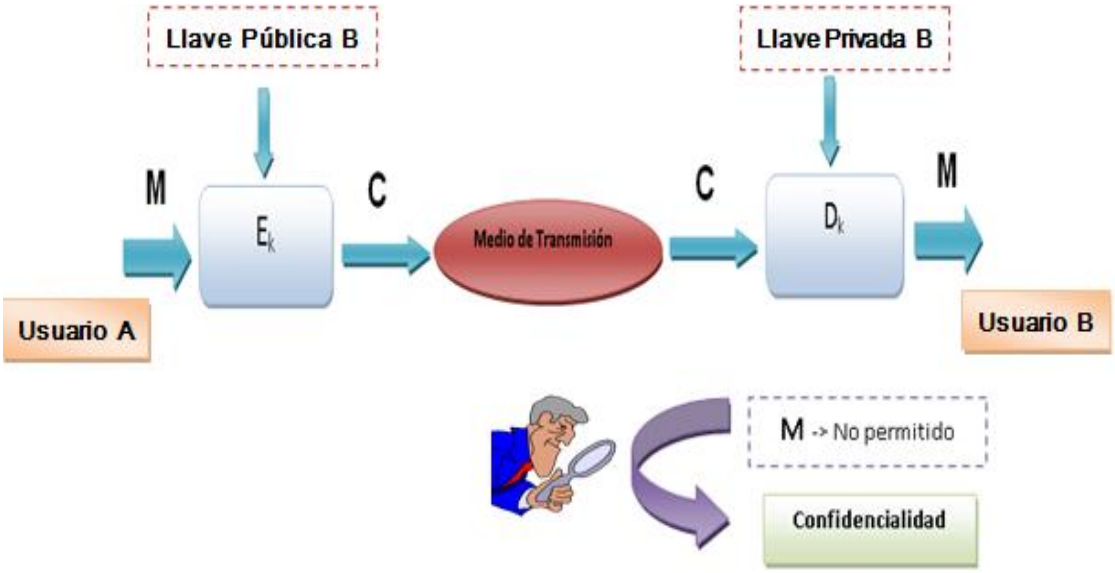
	DES	TDES	AES	IDEA
Potencialidad de la llave	Llave corta y de longitud fija, por lo que no asegura una fortaleza adecuada	Llaves fijas de 128 y 192 bits.	Longitud de llave variable.	Longitud de llave de 128 bits
Velocidad	35 Kbytes/seg	12 Kbytes/seg	55 Kbytes/seg	53 Kbytes/seg
Complejidad de implementación	Fácil de implementar	Fácil de implementar	Fácil de implementar	Fácil y rápido de implementar
Vulnerabilidad del algoritmo	Ha sido roto. Sus claves han sido vulneradas en menos de 24 horas.	Considerado mucho más seguro que el DES al basarse en sus debilidades.	Para ser vulnerado requiere que el atacante pueda ejecutar programas en el mismo sistema.	Su única debilidad conocida es un conjunto de 251 claves débiles, que dentro del universo de llaves no constituyen un peligro.

Un algoritmo roto es aquel donde se ha detectado alguna vulnerabilidad que permite conocer el texto claro o la llave de cifrado, sin embargo se sigue considerando un algoritmo computacionalmente seguro, si el tiempo o los recursos que se emplearon para vulnerarlo no ponen en peligro el valor de la información.

1.3.5 Cifrado asimétrico

En los sistemas asimétricos o de llave pública se crea para cada usuario un par de claves, una privada y otra pública, inversas dentro de un cuerpo finito. Lo que se cifra en emisión con una clave, se descifra en recepción con la clave inversa. La seguridad del sistema reside en la dificultad computacional de descubrir la clave privada a partir de la pública. Para ello, se utilizan funciones matemáticas de un solo sentido o con trampa.

Figura 7: Cifrado asimétrico



Los criptosistemas de clave pública, aunque más lentos que los simétricos, resultan adecuados para las funciones de autenticación, distribución de claves y firmas digitales. El cifrado asimétrico se destaca por ser seguro para el intercambio de las claves aún a través de medios públicos inseguros (como por ejemplo, Internet). Los principales algoritmos de cifrado asimétrico existentes en la actualidad y que son utilizados para el intercambio de claves son Diffie-Hellman, El Gamal y RSA.

Diffie – Hellman (DH): Ideado por los matemáticos Whitfield Diffie y Martín Hellman con el informático Ralph Merkle a mediados de los 70, este algoritmo ha demostrado su factibilidad en comunicaciones inseguras como Internet. Está basado en las

propiedades y en el tiempo necesario para calcular el valor del logaritmo de un número extremadamente grande y primo. En la práctica sólo es válido para el intercambio de llaves simétricas.

RSA (Rivest, Shamir y Adleman): Fue desarrollado en 1977 y en la actualidad es el primer y más utilizado algoritmo de este tipo, es válido tanto para cifrar como para firmar digitalmente. Constituye el algoritmo de cifrado más utilizado en conexiones seguras en línea. La seguridad de este algoritmo radica en el problema de la factorización de números enteros y su funcionamiento se basa en el producto conocido, de dos números primos grandes elegidos al azar y mantenidos en secreto. Por su velocidad suele usarse en sistemas híbridos para cifrar y enviar la llave simétrica que se usa posteriormente en la comunicación cifrada.

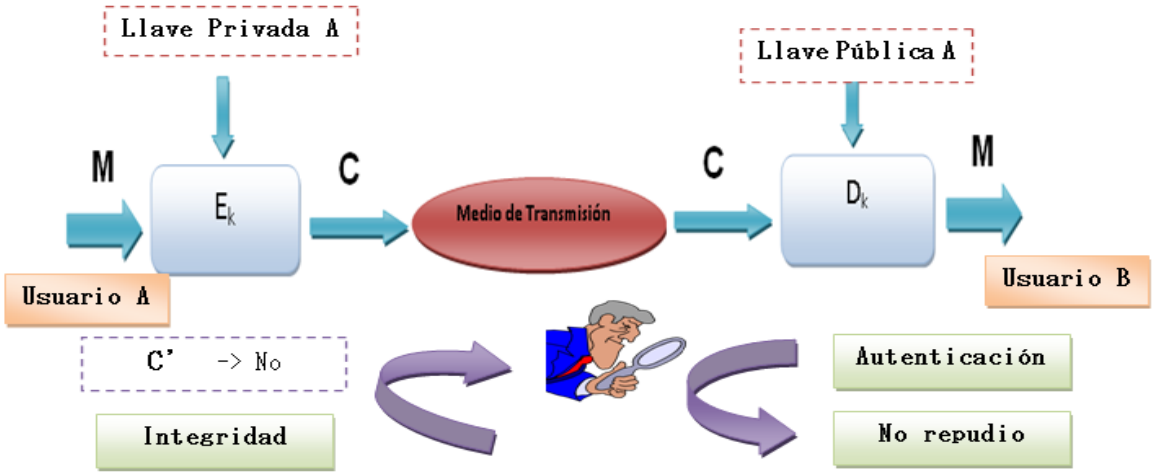
El Gamal: Se refiere a un esquema de cifrado basado en problemas matemáticos de logaritmos discretos. Es un algoritmo de criptografía asimétrica basado en la idea de Diffie-Hellman y que funciona de una forma parecida a este algoritmo. Puede ser utilizado tanto para generar firmas digitales como para cifrar o descifrar.

1.3.5.1 Firma digital

La firma digital constituye un mecanismo de cifrado de los algoritmos asimétricos, utilizado para garantizar la identidad del firmante. Este mecanismo implica el cifrado, mediante la clave privada del emisor, del resumen de los datos que serán transferidos junto con el mensaje; este se procesa en el receptor, para verificar su integridad.

En la práctica, debido a que los algoritmos de clave pública son muy ineficaces a la hora de cifrar documentos largos, los protocolos de firma digital se implementan junto con funciones unidireccionales de resumen (hash), de manera que en vez de firmar un documento, se firma un resumen del mismo.

Figura 8: Firma digital



De esta forma se ofrecen conjuntamente los servicios de no repudio, ya que nadie excepto A podría haber firmado el documento; y de autenticación, ya que si el documento viene firmado por A, podemos estar seguros de su identidad, dado que sólo él ha podido firmarlo. Mediante la firma digital se garantiza asimismo la integridad del documento, ya que en caso de ser modificado, resultaría imposible hacerlo de forma tal que se generase la misma función de resumen que había sido firmada.

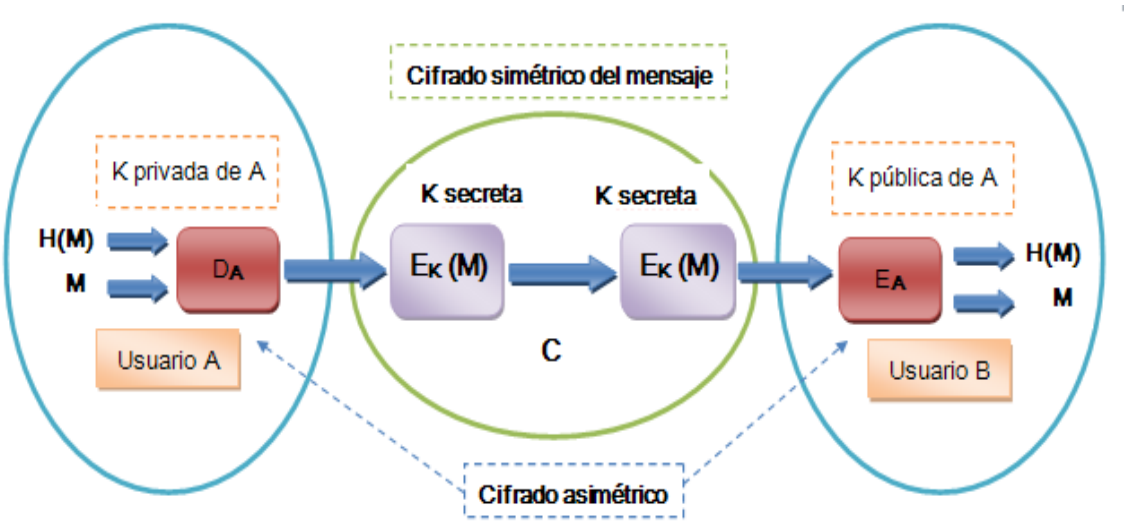
1.3.6 Sistemas híbridos

Los sistemas asimétricos o de clave pública son lentos pero tienen un intercambio de claves fácil de realizar y además cuentan con la firma digital que constituyen un factor importante para garantizar la autenticidad del usuario que envía el mensaje. Sin embargo, los sistemas simétricos o de llave privada son más rápidos pero el intercambio de la llave constituye su principal inconveniente, así como no cuentan con mecanismos de autenticación como la firma digital.

Después del análisis realizado, orientado entre otros aspectos, a la búsqueda de algoritmos de cifrado que permitan la implementación de un sistema para la gestión de la seguridad de las comunicaciones del "Guardián del ALBA", se decide implementar un sistema híbrido ya que permite aprovechar las ventajas de los sistemas simétricos y los

asimétricos, priorizando ante todo la velocidad con que deben realizarse las operaciones de cifrado y descifrado para no afectar la disponibilidad del SCADA.

Figura 9: Funcionamiento de un sistema híbrido



Por sus características de velocidad y variabilidad de las longitudes de llave, elementos que propician la seguridad en el tipo de sistema que se quiere implementar, sin afectar la disponibilidad, se decide utilizar para las operaciones de cifrado y descifrado de los datos que intercambian entre los módulos del SCADA el algoritmo simétrico Advanced Encryption Standard, AES.

Para el intercambio de las llaves utilizadas en el cifrado de las comunicaciones entre los módulos del SCADA se propone utilizar el algoritmo asimétrico RSA, cuya velocidad permitirá realizar dichas operaciones sin afectar el funcionamiento en tiempo real del sistema.

Capítulo 2: Definición de la solución y de las herramientas y tecnologías utilizadas.

2.1 Definición de la solución

El Sistema para la Seguridad de las Comunicaciones del SCADA "Guardián del ALBA" debe ser diseñado para garantizar el intercambio seguro de datos entre los distintos módulos que componen el SCADA sin afectar la disponibilidad del sistema, así como brindar fiabilidad, confidencialidad y rapidez de las comunicaciones.

2.1.1 Requisitos del sistema

Los requisitos con que debe cumplir un sistema se clasifican en funcionales y no funcionales. Los funcionales son capacidades o condiciones que el sistema debe cumplir, mientras que los no funcionales son propiedades o cualidades que hacen al producto atractivo, usable, rápido y confiable.

2.1.1.1 Requisitos funcionales

El Sistema para la Seguridad de las Comunicaciones del SCADA "Guardián del ALBA" debe brindar funcionalidades que permitan el intercambio seguro de datos entre los distintos módulos del SCADA. Estas funcionalidades son:

RF 1. Cifrar texto claro: El sistema debe ser capaz de cifrar datos, utilizando para ello el algoritmo simétrico AES y el asimétrico RSA.

RF 2. Descifrar texto encriptado: El sistema debe ser capaz de descifrar datos, utilizando para ello el algoritmo simétrico AES y el asimétrico RSA.

RF 3. Generar llave: El sistema debe ser capaz de generar las llaves pública y privada para el cifrado y descifrado asimétrico y la llave para el cifrado y descifrado simétrico.

RF 4. Eliminar llave: El sistema debe ser capaz de eliminar una llave almacenada.

RF 5. Almacenar llave: El sistema debe ser capaz de almacenar todas las llaves

generadas.

RF 6. Cargar llave temporal: El sistema debe ser capaz de cargar la llave para una primera comunicación segura de un archivo.

RF 7. Chequear validez de la llave: El sistema debe ser capaz de chequear la validez de las llaves a utilizar en el cifrado y descifrado según un tiempo de caducidad definido.

2.1.1.2 Requisitos no funcionales

Los requisitos no funcionales son propiedades o cualidades que el producto debe cumplir. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido y confiable.

- **Requisitos de rendimiento y disponibilidad del sistema.**

  RNF 1. El sistema debe garantizar su disponibilidad en todo momento para no interferir con la característica de funcionamiento en tiempo real de los sistemas SCADA.

  RNF 2. El sistema debe garantizar la rapidez de las comunicaciones para la validez de la información en tiempo real.

- **Restricciones en el diseño y la implementación.**

  RNF 3. El sistema debe contar con un conjunto de patrones que permita la reutilización del código así como facilidad en la actualización del mismo.

  RNF 4. El código debe cumplir con los estándares de codificación establecidos por el cliente.

  RNF 5. El sistema debe contar con dos módulos, cliente y servidor, que deben funcionar como bibliotecas.

- **Requisitos de seguridad**

  RNF 6. Las llaves para el cifrado/descifrado de datos tienen un tiempo de caducidad definido, lo que las hace válidas por un intervalo restringido de tiempo.

- **Requisitos de confiabilidad**

  RNF 7. Se debe garantizar que el sistema no afecte la fiabilidad e integridad de la información que se transmite.

  RNF 8. Se debe garantizar que el sistema no afecte la disponibilidad de los módulos del SCADA "Guardián del ALBA".

  RNF 9. Las actividades de mantenimiento, supervisión y edición del sistema no deben interferir en el correcto desempeño del resto del sistema, ni afectar la ejecución de la aplicación.

- **Requisitos de apariencia o interfaz externa**

  RNF 10. El sistema debe proveer una interfaz de acceso única tanto en el cliente como en el servidor.

- **Requisitos de usabilidad**

  RNF 11. Las bibliotecas deben poder ser usadas para el intercambio de datos seguro del sistema SCADA "Guardián del ALBA".

- **Requisitos de soporte.**

  RNF 12. Se debe ofrecer servicios de mantenimiento y actualización.

- **Requisitos de portabilidad.**

  RNF 13. El sistema debe funcionar en el sistema operativo GNU/Linux con las distribuciones Ubuntu, Debian y Nova.

- **Requisitos de documentación de usuario.**

  RNF 14. Se debe garantizar que el sistema cuente con una correcta documentación.

- **Requisitos asociados al licenciamiento.**

  RNF 15. Se debe garantizar que el sistema se desarrolle bajo los principios del software libre y por tanto cualquier componente de software que se utilice también debe cumplir son esta premisa.

2.1.2 Definición de la Arquitectura

La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución.

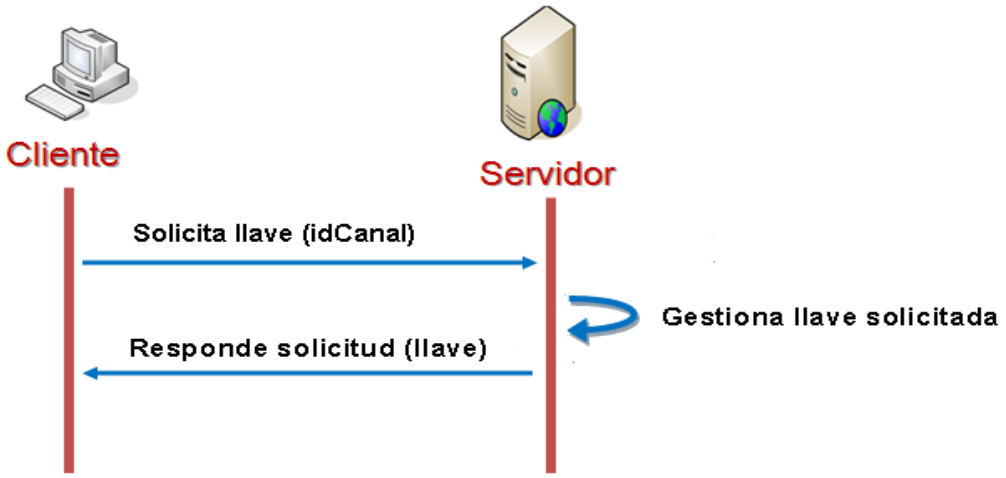
Los patrones arquitectónicos son patrones de software que ofrecen soluciones a problemas de arquitectura en ingeniería de software. Dan una descripción de los elementos y el tipo de relación que tienen, junto con un conjunto de restricciones sobre cómo pueden ser usados. Un patrón arquitectónico expresa un esquema de organización estructural esencial para un sistema de software, que consta de subsistemas, sus responsabilidades e interrelaciones.

2.1.2.1 Arquitectura Cliente-Servidor

La tecnología Cliente-Servidor es el procesamiento cooperativo de la información por medio de un conjunto de procesadores, en el cual múltiples clientes, distribuidos geográficamente, solicitan requisitos a uno o más servidores centrales. Este modelo provee usabilidad, interoperabilidad, flexibilidad y escalabilidad en las comunicaciones.

En esta arquitectura el cliente envía un mensaje al servidor solicitando un servicio (en el caso de nuestro sistema las llaves para la comunicación) y este envía uno o varios mensajes con la respuesta. La siguiente figura (Figura 10) muestra el esquema de comunicación Cliente-Servidor empleado en el Sistema para la Seguridad de las Comunicaciones del SCADA “Guardián del ALBA”.

Figura 10: Esquema de comunicación Cliente-Servidor



2.1.2.2 Patrón arquitectónico 3 capas

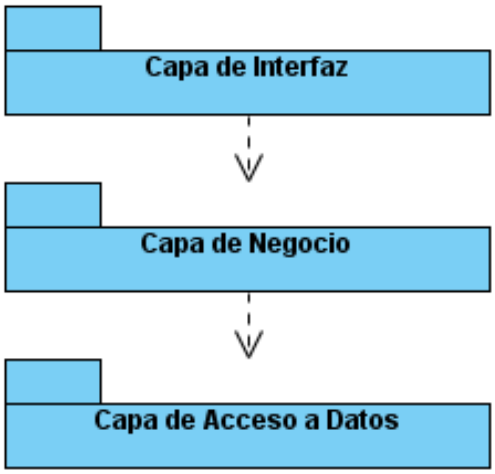
El desarrollo del sistema está basado en la estructura del patrón arquitectónico **N-Capas**, especializándose en tres capas. La utilización de este patrón, proporciona aplicaciones más flexibles y robustas debido al encapsulamiento, con desarrollos paralelos en cada capa, añadiendo un mantenimiento y soporte más sencillos.

1.- Capa de interfaz: Es el nivel que se encuentra al alcance del usuario, establece la comunicación, envía y recibe los datos cifrados. Esta capa se comunica únicamente con la capa de negocio y oculta funcionalidades que son transparentes al usuario.

2.- Capa de negocio: Es el nivel donde residen las funcionalidades que se ejecutan. Se denomina capa de negocio (e incluso de lógica del negocio) porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de interfaz, para recibir y enviar los datos de la comunicación, y con la capa de datos, para el almacenamiento o recuperación de datos.

3.- Capa de acceso a datos: Es la capa que sirve de puente entre la capa de negocio y el proveedor de datos, además se encarga de acceder a estos y recibe solicitudes de inserción, eliminación o carga de los mismos, desde la capa de negocio.

Figura 11: Patrón arquitectónico tres capas



2.2 Definición de las herramientas y tecnologías utilizadas

Basándose en el estudio realizado de las diferentes metodologías, lenguajes, herramientas y tecnologías existentes y su desarrollo actual, se decide utilizar las listadas en este capítulo por su peculiaridad de pertenecer a la categoría de software libre. Además, con el uso de las mismas se está siendo consecuente con las políticas trazadas por el Centro de Informática Industrial, y más específicamente el Proyecto Seguridad.

2.2.1 Sistema Operativo

Un sistema operativo puede ser contemplado como una colección organizada de extensiones software del hardware, consistente en rutinas de control que hacen funcionar al computador y proporcionan un entorno para la ejecución de programas. Estos programas utilizan las facilidades proporcionadas por el sistema operativo para obtener acceso a recursos del sistema informático como el procesador, archivos y dispositivos de entrada/salida. De esta forma, el sistema operativo constituye la base sobre la cual pueden escribirse los programas de aplicación, los cuales invocarían sus servicios por medio de llamadas al sistema. Estos actúan como interfaz entre los usuarios/aplicaciones y el hardware de un sistema informático. (Ruiz Múzquiz, 2004).

2.2.1.1 GNU/Linux

GNU/Linux es un sistema operativo que ha sido desarrollado bajo los principios del software libre. Software libre se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software. De modo más preciso, se refiere a cuatro libertades de los usuarios del software (HispaLinux, 2007):

- La libertad de usar el programa, con cualquier propósito (libertad 0).
- La libertad de estudiar cómo funciona el programa, y adaptarlo a tus necesidades (libertad 1).
- La libertad de distribuir copias, con lo que puedes ayudar a tu vecino (libertad 2).
- La libertad de mejorar el programa y hacer públicas las mejoras a los demás, de modo que toda la comunidad se beneficie (libertad 3).

Se selecciona el sistema operativo GNU/Linux, en sus distribuciones[4] Ubuntu, Debian y Nova, por ser un sistema operativo robusto, estable y rápido, que por su característica de ser software libre tiene un costo cero o muy bajo y disponibilidad del código fuente que permite aprender, modificar o ayudar al desarrollo del sistema.

2.2.2 Lenguaje de modelado: UML

El lenguaje de modelado es la notación (principalmente gráfica) utilizada para expresar un diseño, el proceso indica los pasos que se deben seguir para llegar a este. (González Cornejo, 2009).

El Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, Unified Modeling Language) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software.

Este constituye un lenguaje para especificar y no para describir métodos o procesos. Se

[4] Recopilación de programas y ficheros, organizados y preparados para su instalación.

puede aplicar en una gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como el Proceso Unificado de Rational), pero no especifica en sí mismo qué metodología o proceso usar. (Proyecto Zero, 2010)

2.2.3 Lenguaje de programación: C++

Un lenguaje de programación es un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Es utilizado para controlar el comportamiento físico y lógico de una máquina. (Florentino, 2007)

C++ es un lenguaje de programación diseñado con la intención de extender al exitoso lenguaje de programación C con mecanismos que permitiesen la manipulación de objetos. Posteriormente se añadieron facilidades de programación genérica, que se sumaron a los otros dos paradigmas que ya estaban admitidos (programación estructurada y la programación orientada a objetos). Por esto se suele decir que el C++ es un lenguaje multiparadigma. C++ permite trabajar tanto a alto como a bajo nivel. (Calvo, 2006)

Se decide utilizar C++ como lenguaje de programación para el desarrollo del sistema debido a que:

- Es un lenguaje orientado a objetos.
- Es muy potente y práctico, debido a su robustez, para la creación de sistemas complejos.
- Es independiente de la plataforma.
- Es muy utilizado en el mundo por lo que existe abundante información sobre su uso.
- Existen muchos algoritmos ya desarrollados en C++, de manera que pueden tomarse y amoldarse a la solución deseada.

2.2.4 Metodología de desarrollo: RUP

Una metodología de desarrollo es una colección de documentación formal referente a los procesos, las políticas y los procedimientos que intervienen en el desarrollo del software. La finalidad de una metodología de desarrollo es garantizar la eficacia y la eficiencia en el proceso de generación de software. (Blanco Cuaresma, 2008)

El Proceso Unificado de Rational, (RUP, por sus siglas en inglés, Rational Unified Process) es una metodología de procesos de desarrollo de software que brinda guías consistentes y personalizadas para todo el equipo de proyecto. Está publicado, distribuido y soportado en forma abierta y documentado en forma coherente y completa. Es una metodología flexible al momento de emplear los artefactos que se generan. Su objetivo fundamental es obtener productos de software de alta calidad, que satisfagan los requisitos y en un tiempo y presupuesto predecibles.

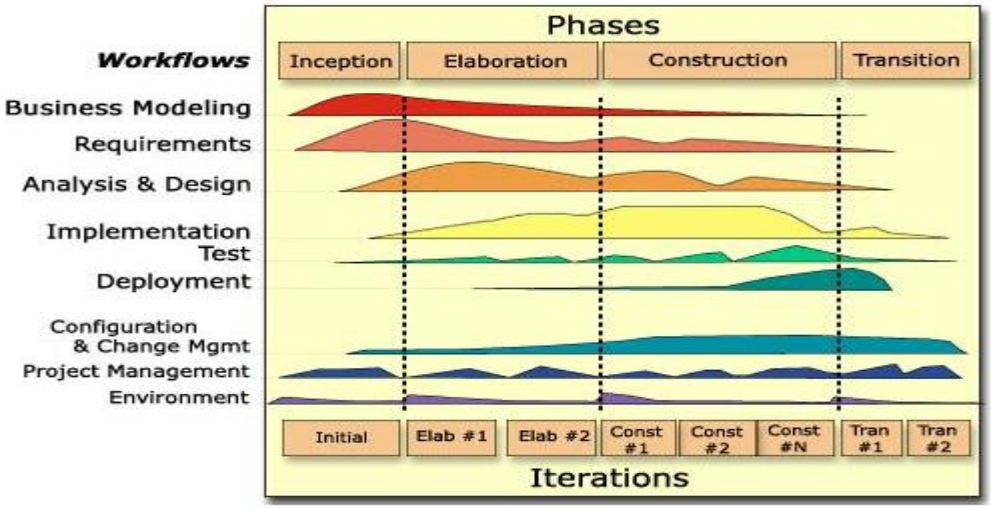
Por ser una metodología centrada en la arquitectura, facilita la visión del sistema completo. La arquitectura describe los elementos del modelo que son más importantes para su construcción, o sea, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente.

Una de las mejores prácticas centrales de RUP es la noción de desarrollar iterativamente. Esta metodología organiza los proyectos en términos de disciplinas y fases, consistiendo cada una en una o más iteraciones. (Figura 10).

Se decide utilizar RUP como metodología de desarrollo de software debido a que:

- Provee un entorno de proceso de desarrollo configurable, basado en estándares.
- Permite tener claro y accesible el proceso de desarrollo que se sigue.
- Permite ser configurado a las necesidades de la organización y del proyecto.
- Contiene un enfoque orientado a objetos.

Figura 12: Metodología de desarrollo de software: RUP



2.2.5 Herramienta CASE[5]: Visual Paradigm

Una herramienta CASE es un conjunto de métodos, utilidades y técnicas que facilitan la automatización del ciclo de vida del desarrollo de sistemas de información, completamente o en alguna de sus fases. (Bello Lobato, 2008) Las herramientas CASE abarcan todos los pasos del desarrollo del software, y también aquellas actividades generales que se aplican a lo largo del proceso.

Visual Paradigm es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. (Download Manager)

Se decide utilizar Visual Paradigm como herramienta CASE debido a que:

- Posee capacidades de ingeniería directa e inversa.

[5] Siglas en inglés: Computer Aided Software Engineering (Ingeniería de Software Asistida por Ordenador).

- Puede integrarse a los principales entornos de desarrollo.
- Posee disponibilidad de múltiples versiones para cada necesidad.

2.2.6 Entorno de desarrollo: Eclipse

Un entorno de desarrollo integrado (Integrated Development Environment, IDE) es un programa compuesto por una serie de herramientas que utilizan los programadores para desarrollar código. Esta herramienta puede estar pensada para su utilización con un único lenguaje de programación o bien puede dar cabida a varios de éstos.

Los entornos de desarrollo proporcionan un marco de trabajo para la mayoría de los lenguajes de programación existentes en el mercado (por ejemplo C, C++, C#, Java, Python y Visual Basic, entre otros). Además, es posible que un mismo entorno de desarrollo tenga la posibilidad de utilizar varios lenguajes de programación, como es el caso de Eclipse. (Universidad de Oviedo, 2009)

Se decide utilizar Eclipse debido a que es un IDE gratuito, libre, potente, tiene gran soporte en la comunidad de software libre y permite instalar complementos o plugins[6] para lograr un entorno de desarrollo ampliado con posibilidades variadas de uso.

2.2.7 Base de datos: SQLite

SQLite es una biblioteca que implementa un proceso autónomo, sin servidor, sin necesidad de configuración, con motor de base de datos transaccional[7]. El código para SQLite es de dominio público y por lo tanto su uso es libre para cualquier propósito, comercial o privado.

SQLite es un motor de base de datos SQL[8] embebido. A diferencia de la mayoría de otras bases de datos SQL, no tiene un proceso de servidor independiente. SQLite lee y escribe directamente en archivos de disco normal y el formato de archivo de base de

[6] Aplicación que se relaciona con otra para aportarle una función nueva y específica.
[7] Base de datos cuyo único fin es el envío y recepción de datos a grandes velocidades.
[8] Lenguaje de Consulta Estructurado, por sus siglas en inglés Structured Query Language.

datos es multiplataforma. (Sitio Oficial SQLite)

Se decide utilizar SQLite para el almacenamiento de las llaves debido a que no tiene necesidad de un servidor, haciendo al sistema más ligero; constituye una herramienta libre y utiliza código SQL para la gestión de la base de datos.

2.2.8 Generador de documentación: Doxygen

El paquete Doxygen permite generar la documentación correspondiente a una aplicación. Puede generar archivos en HTML[9] y/o un manual de referencia a partir de un grupo de ficheros fuente documentados. Se decide utilizar Doxygen ya que contiene un sistema de documentación para C++ que es el lenguaje de programación que se utiliza en el sistema. Además, brinda la facilidad de extraer la documentación directamente de las fuentes, lo que hace mucho más fácil mantener la consistencia con el código fuente.

2.2.9 Bibliotecas de cifrado

En ciencias de la computación, una biblioteca (del inglés library) es un conjunto de subprogramas utilizados para desarrollar software. Las bibliotecas contienen código y datos, que proporcionan servicios a programas independientes, es decir, pasan a formar parte de éstos. Esto permite que el código y los datos se compartan y puedan modificarse de forma modular.

La biblioteca seleccionada para la implementación de las funcionalidades de cifrado del componente a desarrollar debe ser una herramienta libre, de código abierto, que permita el uso de los algoritmos seleccionados para realizar el cifrado y descifrado de datos en un sistema híbrido a altas velocidades. Luego de un análisis basado en estas características, fundamentales para el desarrollo del sistema, se decide realizar una selección entre las bibliotecas de cifrado SSL y Cripto++ para elegir la que será utilizada en la implementación del sistema para la seguridad de las comunicaciones del SCADA "GALBA".

[9] Siglas en inglés: HyperText Markup Language (Lenguaje de Marcado de Hipertexto), empleado para elaborar páginas web.

Crypto++: también conocida como CryptoPP, libcrypto++ y libcryptopp, Es una biblioteca para el desarrollo de programas en C++ que requieren algoritmos criptográficos. Permite utilizar algoritmos cifradores por bloque (IDEA, AES, DES, RC5); funciones hash (SHA-1, SHA-256, SHA-512) y sistemas de clave pública (RSA y DSA).

SSL: Implementa los protocolos Secure Sockets Layer (Capa de Conectores de Red Segura) (SSL v2/v3) y Transport Layer Security (Capa de Transporte Segura) (TLS v1). Permite generar claves RSA, DH y DSA, calcular resúmenes de mensajes (SHA-256); cifrar y descifrar con distintos algoritmos de cifrado simétricos (DES, IDEA, AES) y asimétricos (RSA). Con esta biblioteca es posible realizar pruebas cliente/servidor utilizando el protocolo SSL y crear certificados X.509[10], CSRs[11] y CRLs[12].

2.2.9.1 Matriz de decisión para bibliotecas de cifrado

La Matriz de Decisión es una técnica aplicable a distintos campos, dentro y fuera de la ingeniería, para la toma de decisiones racionales, entre distintas alternativas posibles. Mejora la objetividad del proceso de selección por ser estructurado, de metodología sistemática, repetible y con resultados en idioma universal (números).

Los criterios de selección son cualidades y condiciones a satisfacer por las alternativas propuestas y su determinación se basa en el cumplimiento de las necesidades del sistema a desarrollar. Cada uno de estos factores cuenta con un valor de acuerdo a su importancia relativa para la decisión. Cada una de las alternativas es puntuada en correspondencia con el factor evaluado, la evaluación puede ser de:

- 0 – No aceptable
- 1 – Poco aceptable
- 2 - Aceptable

[10] Estándar para infraestructuras de claves públicas.

[11] Lista de Certificados Revocados, certificados que ya no son válidos y en los que no debe confiar ningún usuario del sistema.

[12] Solicitud de Firma de Certificado, mensaje enviado por el solicitante a una Autoridad de Certificación con el fin de solicitar un Certificado de Identidad Digital.

- 3 - Sobresaliente
- 4 - Excelente

Las puntuaciones de cada uno de los factores son multiplicadas por los pesos relativos de cada opción. La suma de los valores ponderados de cada opción nos permite decidir, asumiendo que la opción de mayor valor es la más conveniente según los criterios seleccionados.

2.2.9.2 Criterios de evaluación

Los criterios de evaluación seleccionados según las características que debe cumplir el sistema son:

1. **Facilidad de implementación:** Este criterio pretende evaluar cuán difícil sería implementar la solución, para cumplir con las restricciones de tiempo y fiabilidad.
2. **Portabilidad:** Las tecnologías deben ser multiplataforma, para permitir la utilización del sistema de cifrado en otro tipo de aplicaciones.
3. **Documentación:** Cantidad y claridad de la documentación encontrada.

2.2.9.3 Importancia

La siguiente tabla (Tabla 2) muestra los valores de importancia relativa asignados a cada uno de los anteriores criterios de evaluación. Esta importancia ha sido asignada teniendo en cuenta que ambas bibliotecas seleccionadas cumplen con bastante equidad con las características fundamentales que debe cumplir el sistema, por lo que puede basarse la selección en características que faciliten la implementación de las funcionalidades de cifrado.

Tabla 2: Importancia de los criterios de decisión

Criterios	Importancia
Facilidad de implementación	3
Portabilidad	3
Documentación	4

2.2.9.4 Puntajes y selección

La siguiente tabla (Tabla 3) muestra los resultados de la evaluación de las bibliotecas de cifrado según los pesos relativos de los criterios de decisión seleccionados. La evaluación está realizada cuantitativamente de 0 a 4 puntos, donde 0 resulta una evaluación no aceptable y 4 de excelente.

Tabla 3: Puntajes de las bibliotecas de cifrado

		Alternativas			
Modelo de decisión		**SSL**		**Crypto++**	
Criterios	**Importancia**	**Calificación**	**Puntaje**	**Calificación**	**Puntaje**
Facilidad de implementación	3	5	15	4	12
Portabilidad	3	5	15	4	12
Documentación	4	5	20	5	20
Total			**50**		**44**

Luego de evaluadas ambas alternativas y según los resultados obtenidos se selecciona la biblioteca SSL para la implementación de las funcionalidades de cifrado por brindar, además de las características definidas con anterioridad, la facilidad de implementación y cantidad y claridad de la documentación necesarias para culminar en tiempo y con la calidad esperada el componente de cifrado para la seguridad de las comunicaciones del SCADA "Guardián del ALBA".

2.2.10 Bibliotecas Boost

Boost es un conjunto de bibliotecas de código abierto que permite extender las capacidades que oferta C++. Su licencia posibilita que sea empleada en cualquier tipo de proyecto, ya sea comercial o no. Está diseñada e implementada de manera tal que puede utilizarse en un amplio espectro de aplicaciones y plataformas. (Sorio, 2010)

Este conjunto de bibliotecas brinda funcionalidades, entre las que se encuentran la posibilidad de representar períodos de tiempo y de realizar casteo de diferentes tipos de datos primitivos (string, int, float), que resultan muy útiles para el desarrollo del sistema para la seguridad de las comunicaciones del SCADA "Guardián del ALBA".

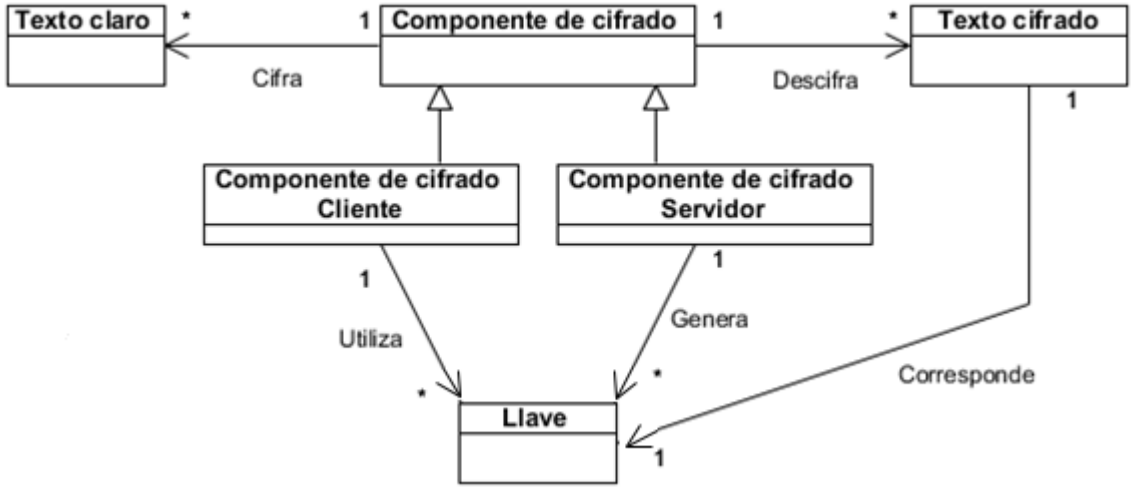
Capítulo 3: Descripción de la solución propuesta.

3.1 Modelo del dominio

Un modelo del dominio captura los tipos más importantes de objetos que existen o los eventos que suceden en el entorno donde estará el sistema; se considera en RUP un subconjunto del llamado modelo de objetos del negocio.

Un modelo del dominio es una representación de las clases conceptuales del mundo real, no de componentes de software. Este tipo de modelo tiene como objetivo principal ayudar a comprender los conceptos con los que deberá trabajar nuestra aplicación. Estos conceptos se representan en el siguiente diagrama (Figura 13):

Figura 13: Modelo del dominio



3.1.1 Glosario de términos del dominio

Componente de cifrado: Herramienta que realiza las operaciones de cifrado y descifrado de datos. El sistema cuenta con un componente de cifrado cliente y un componente de cifrado servidor.

Componente de cifrado Cliente: Utiliza llaves para realizar las operaciones de cifrado y descifrado de texto.

Componente de cifrado Servidor: Genera las llaves para las operaciones de cifrado y descifrado de texto.

Texto claro: Texto que no está cifrado y es inteligible[13].

Texto cifrado: Texto ininteligible[14], solamente podrá ser leído luego de ser descifrado.

Llave: Pieza de información que controla las operaciones de cifrado y descifrado en los algoritmos criptográficos.

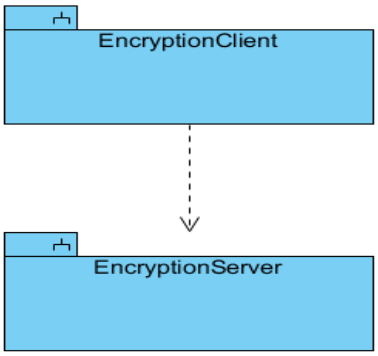
3.2 Descripción del sistema

El Sistema para la Seguridad de las Comunicaciones del SCADA "Guardián del ALBA" está compuesto por dos subsistemas, como se muestra en la figura (Figura 14):

EncryptionClient: Biblioteca adjunta a todos los módulos del SCADA, encargada de la administración de las llaves de comunicación entre módulos. Este subsistema realiza los pedidos de claves para el cifrado y descifrado de datos al servidor.

EncryptionServer: Este subsistema "escucha" las peticiones de los clientes y les brinda las llaves para el intercambio de información entre ellos. Administra las llaves de la comunicación entre los clientes y entre el cliente y el servidor.

Figura 14: Diagrama de subsistemas de la solución propuesta



[13] Comprensible, claro, legible.
[14] Incomprensible, indescifrable, incognoscible.

3.2.1 Descripción de las funcionalidades

El Sistema para la Seguridad de las Comunicaciones se encarga del cifrado y descifrado de todos los datos que se intercambian entre los módulos que componen el SCADA "Guardián del ALBA".

Para realizar las operaciones de cifrado y descifrado (RF 1. y RF 2.) necesarias para el intercambio seguro de datos, los clientes deben solicitar al servidor de cifrado las llaves correspondientes a cada uno de los canales por los que se va a transmitir la información. Para el envío de las solicitudes y la obtención de las respuestas es necesario establecer una comunicación entre los clientes de cifrado y el servidor.

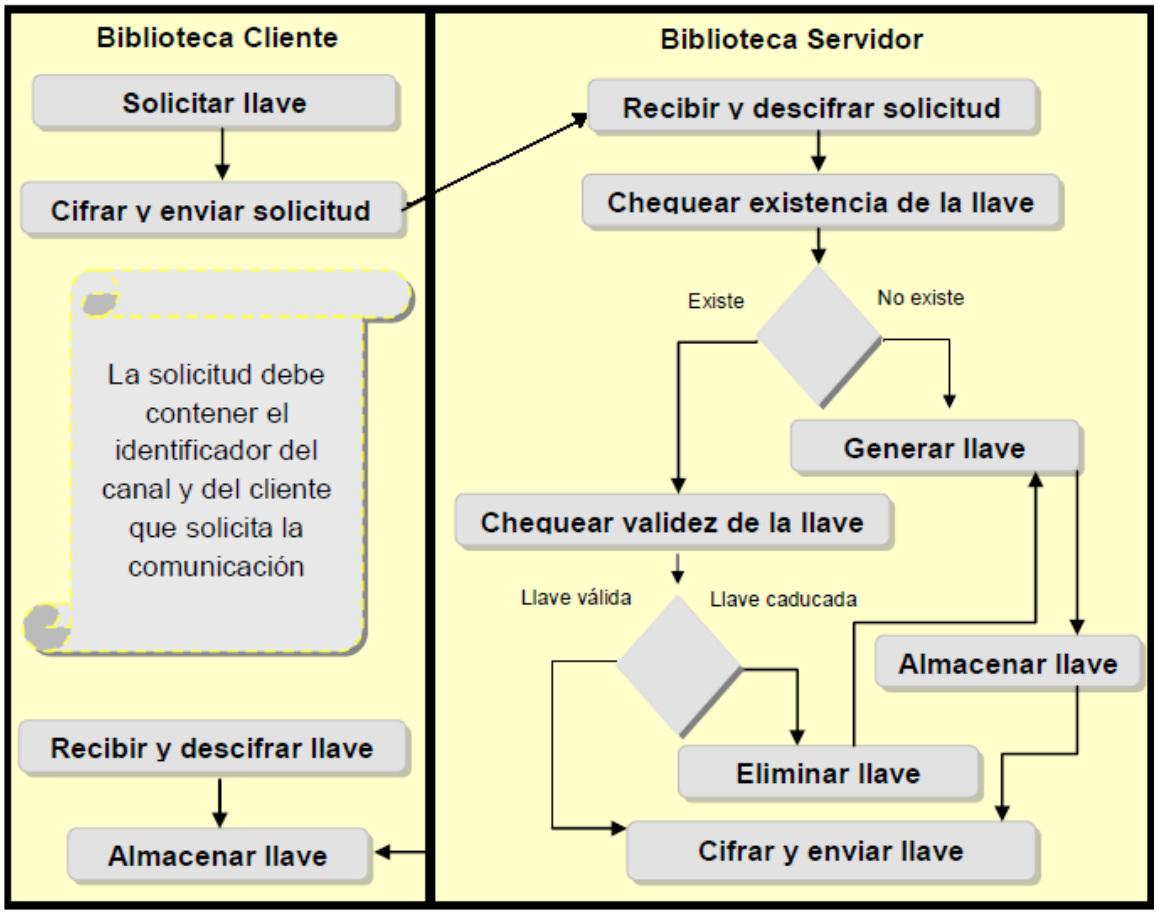
El servidor atiende a cada solicitud generando una llave para cada canal (RF 3), puesto que el cifrado para el intercambio de datos entre clientes se realiza mediante un algoritmo simétrico. En caso de que dicha llave ya se encuentre almacenada (RF 5), pues ha sido generada para una comunicación anterior, se chequea que la misma sea válida (RF 7). Cada llave generada tiene un período de validez definido, como requisito de seguridad, y luego de transcurrido este tiempo debe ser eliminada (RF 4) y generada una nueva llave.

El cifrado y descifrado de las solicitudes enviadas por los clientes de cifrado al servidor y de las respuestas de este, se realizan mediante un algoritmo asimétrico. Las llaves para estas comunicaciones Cliente-Servidor también serán generadas y administradas por el servidor, que de igual forma es el encargado de garantizar que cumplan con el período de validez establecido.

El servidor antes del inicio de las comunicaciones generará una llave temporal, necesaria para la primera comunicación segura Cliente-Servidor y la almacenará para su distribución manual a cada uno de los módulos.

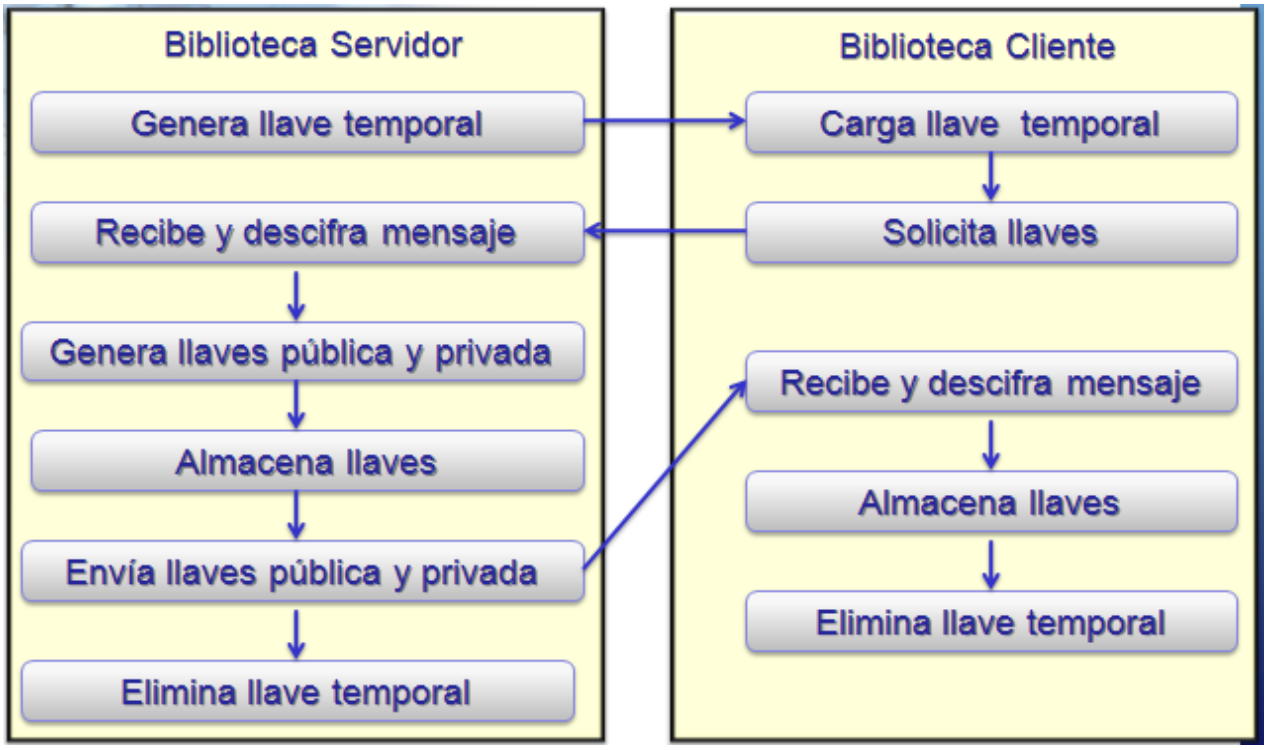
La siguiente figura (Figura 15) muestra la descripción de las principales funcionalidades anteriormente descritas, mediante la interacción entre las bibliotecas cliente y servidor.

Figura 15: Descripción de las principales funcionalidades



La funcionalidad cargar primera llave (RF 6), tiene como propósito garantizar la seguridad en las comunicaciones del sistema desde el primer intercambio de datos. El subsistema EncryptionServer encargado de la producción de llaves, generará una llave temporal que será copiada, en una dirección de memoria específica, en todos los módulos que componen el SCADA "Guardián del ALBA". Cada uno de estos módulos usará esta llave temporal para realizar el pedido al servidor de las llaves para la comunicación entre los clientes y el servidor. El EncryptionServer dará respuesta a estos pedidos utilizando la llave temporal, la cual será destruida luego de finalizado el proceso. La siguiente figura (Figura 16) muestra como se realiza la primera comunicación entre los subsistemas que componen la solución.

Figura 16: Primera comunicación Cliente-Servidor



3.3 Análisis y diseño del sistema

Esta disciplina explica cómo transformar los requisitos en los productos de trabajo que especifiquen el diseño del software que se va a desarrollar.

El propósito de esta disciplina es:

- Transformar los requisitos en un diseño de lo que será el sistema.
- Evolucionar una arquitectura robusta para el sistema.
- Adaptar el diseño para que se ajuste al entorno de implementación, con un diseño pensado para el rendimiento.

El análisis consiste en obtener una visión del sistema a partir de los requisitos funcionales. Por otro lado, el diseño es un refinamiento del análisis que tiene en cuenta

los requisitos no funcionales, para determinar de qué forma deberá cumplir el sistema sus objetivos.

3.3.1 Clases persistentes

La persistencia es la capacidad de un objeto de mantener su valor en el espacio y en el tiempo. Las clases persistentes se definen para conocer la información real representada en las tablas de la base de datos. (Mesa Valiente & Castillo Pérez, 2008)

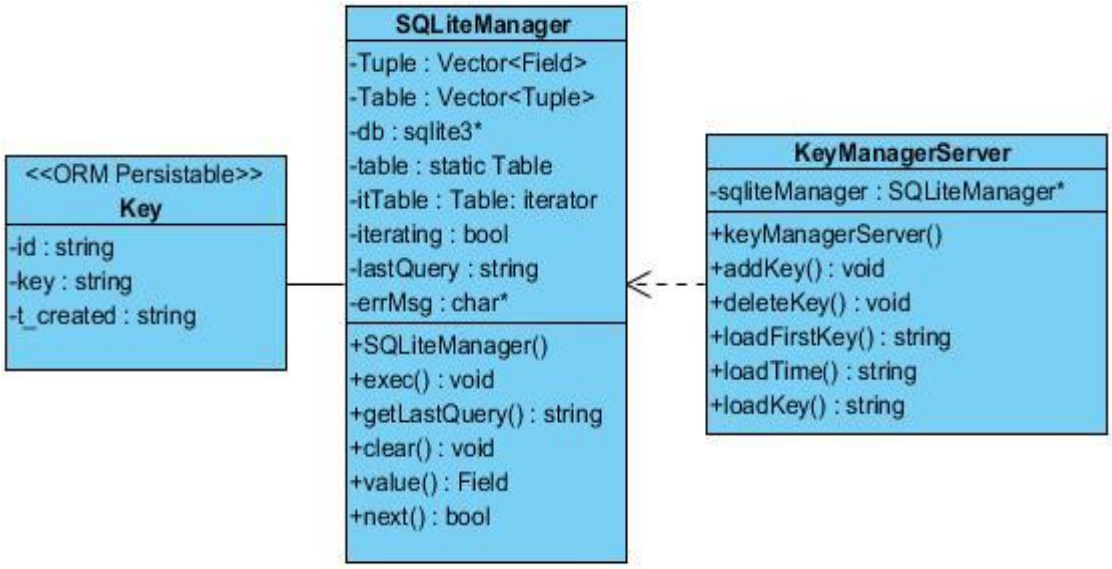
El sistema debe almacenar las llaves de cifrado para la comunicación entre clientes y para la comunicación Cliente-Servidor y chequear que su tiempo de vida esté dentro del tiempo de validez especificado, en la Tabla 4 se muestra una descripción de la clase persistente para su mejor comprensión.

Tabla 4: Descripción de clase persistente Key

Nombre	**Key**	
Descripción	Almacena los datos relacionados con las llaves de cifrado y descifrado de datos.	
Atributo	**Tipo**	**Descripción**
id	Text	Se refiere al identificador para los canales de comunicación y las llaves públicas y privadas de comunicación con el servidor.
key	Text	Se refiere a las llaves para el cifrado y descifrado.
tCreated	Text	Se refiere al momento de creación de la llave y se almacena en el formato AAAA-MM-DD HH:MM:SS

La siguiente figura (Figura 17) muestra el diagrama de clases persistentes que se realizó para modelar la estructura lógica de la base de datos.

Figura 17: Diagrama de clases persistentes



La clase KeyManagerServer es la encargada de realizar las operaciones relacionadas con el almacenamiento de las llaves en el servidor, a través de la clase SQLiteManager la cual gestiona el uso de la base de datos SQLite, pudiendo de esta manera agregar, eliminar y cargar las llaves necesarias para las operaciones de cifrado y descifrado de datos. En los clientes el almacenamiento de las llaves se realiza a través de mapas que contienen la información en el mismo formato que en el servidor.

3.3.2 Diagramas de clases

El diagrama de clases captura la estructura lógica del sistema y las clases que constituyen el modelo. Es un modelo estático, describiendo lo que existe y qué atributos y comportamiento tiene. Los diagramas de clases son los más útiles para ilustrar las relaciones entre las clases e interfaces (Architect G. d, 2007).

Las siguientes figuras (Figuras 18-19) muestran los diagramas de clases que componen los subsistemas de la solución siguiendo la estructura definida por la arquitectura utilizada.

Figura 18: Diagrama de clases del EncryptionClient

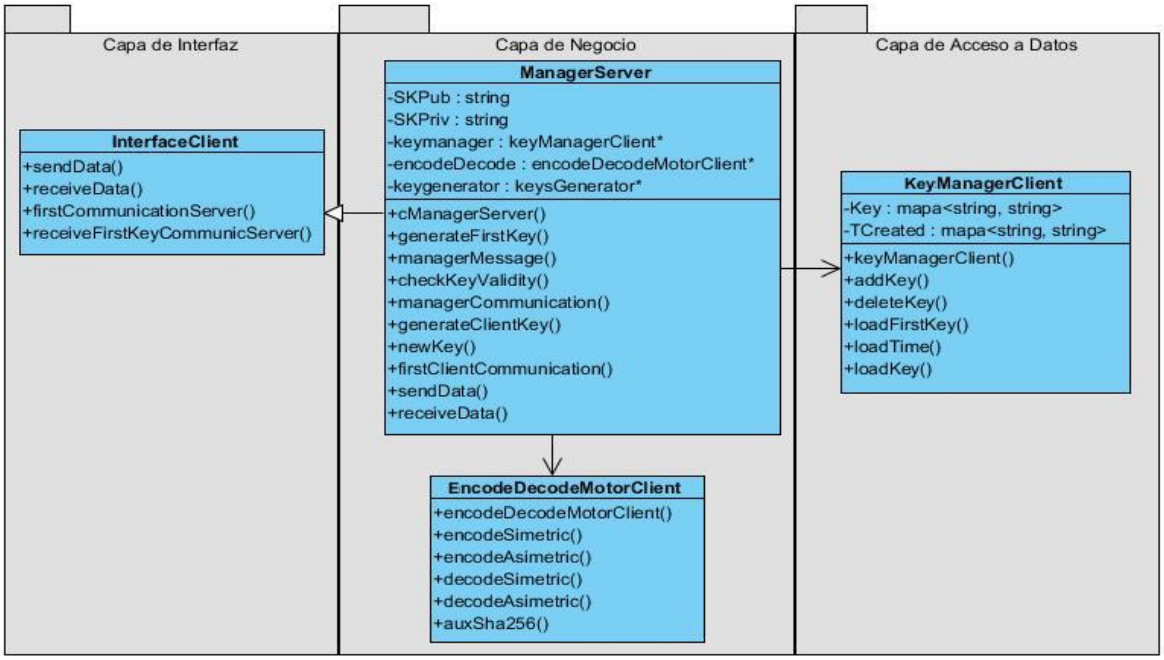
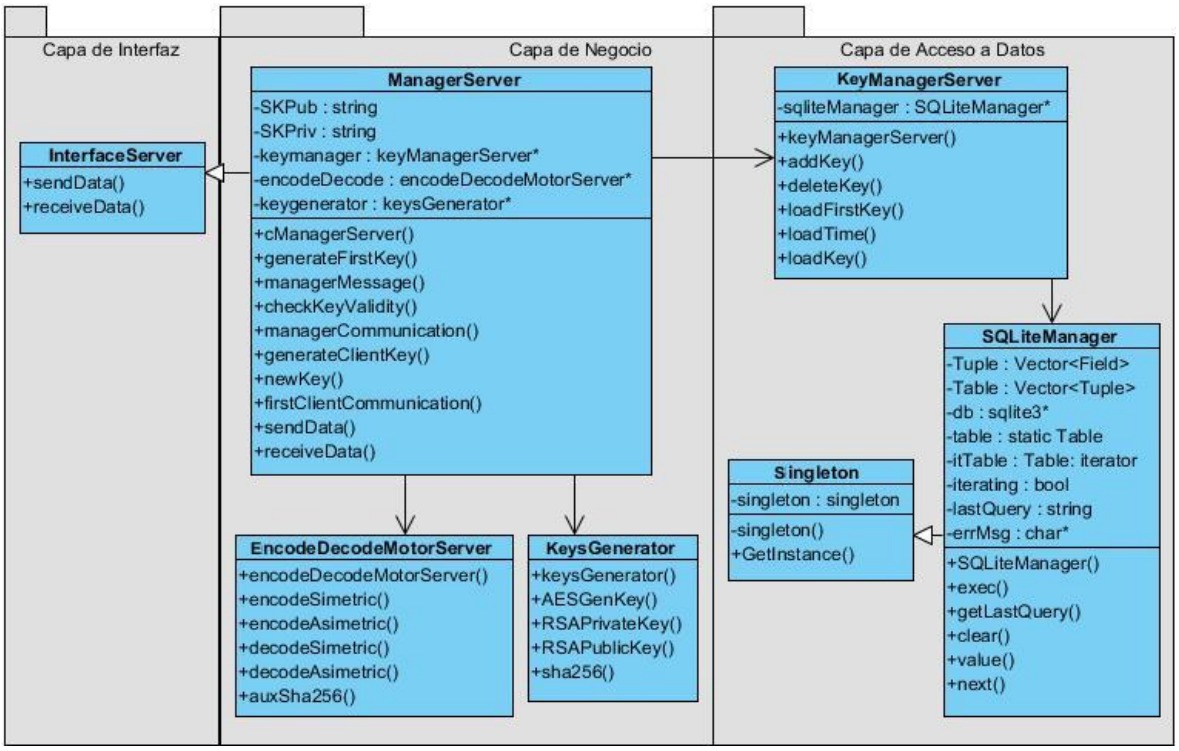


Figura 19: Diagrama de clases del EncryptionServer



3.3.2.1 Descripción de las principales clases

En los Anexos de esta investigación pueden consultarse las descripciones de las clases más importantes para el desarrollo del Sistema para la Seguridad de las Comunicaciones del SCADA "Guardián del ALBA", donde se detallan cada una de las funcionalidades que las componen.

3.3.3 Diagramas de interacción

El diagrama de secuencia es un tipo de diagrama de interacción que muestra el intercambio de mensajes en un determinado espacio de tiempo resaltando el orden y momento en que se envían los mensajes a los objetos. En las siguientes figuras se muestra la interacción entre las clases que conforman los subsistemas.

3.3.3.1 Diagramas de secuencia del EncryptionClient

Figura 20: Diagrama de secuencia para procesar la entrada de datos cifrados

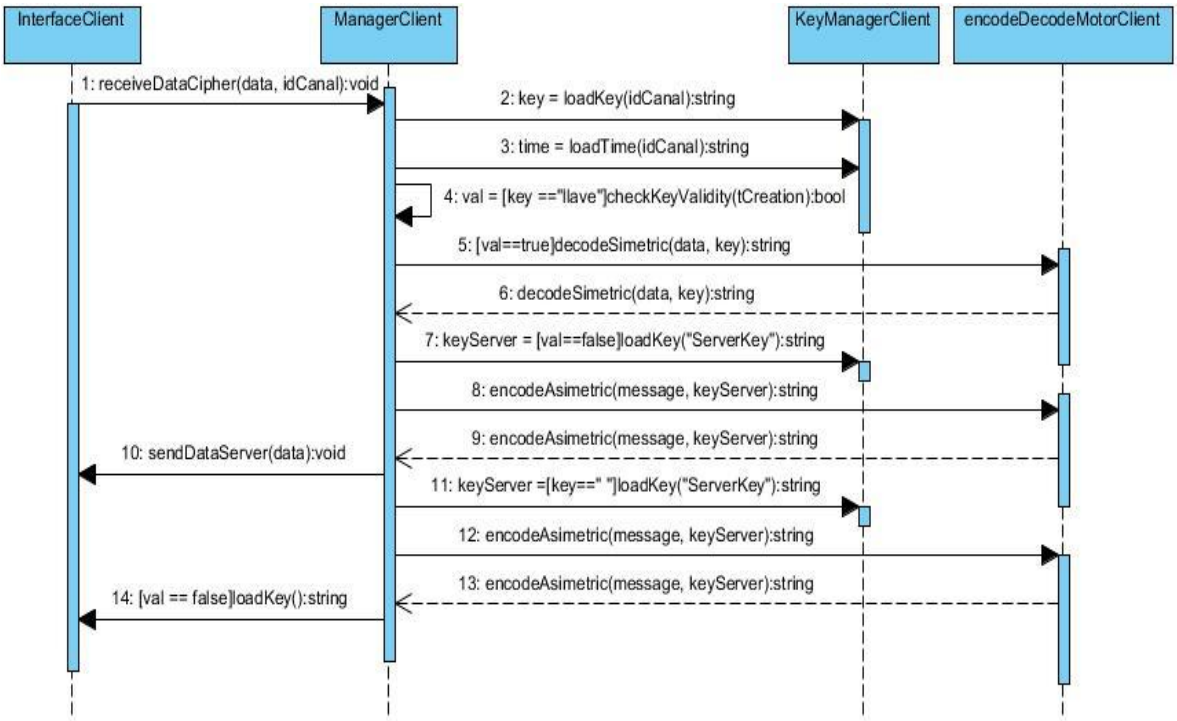


Figura 21: Diagrama de secuencia para la salida de datos cifrados

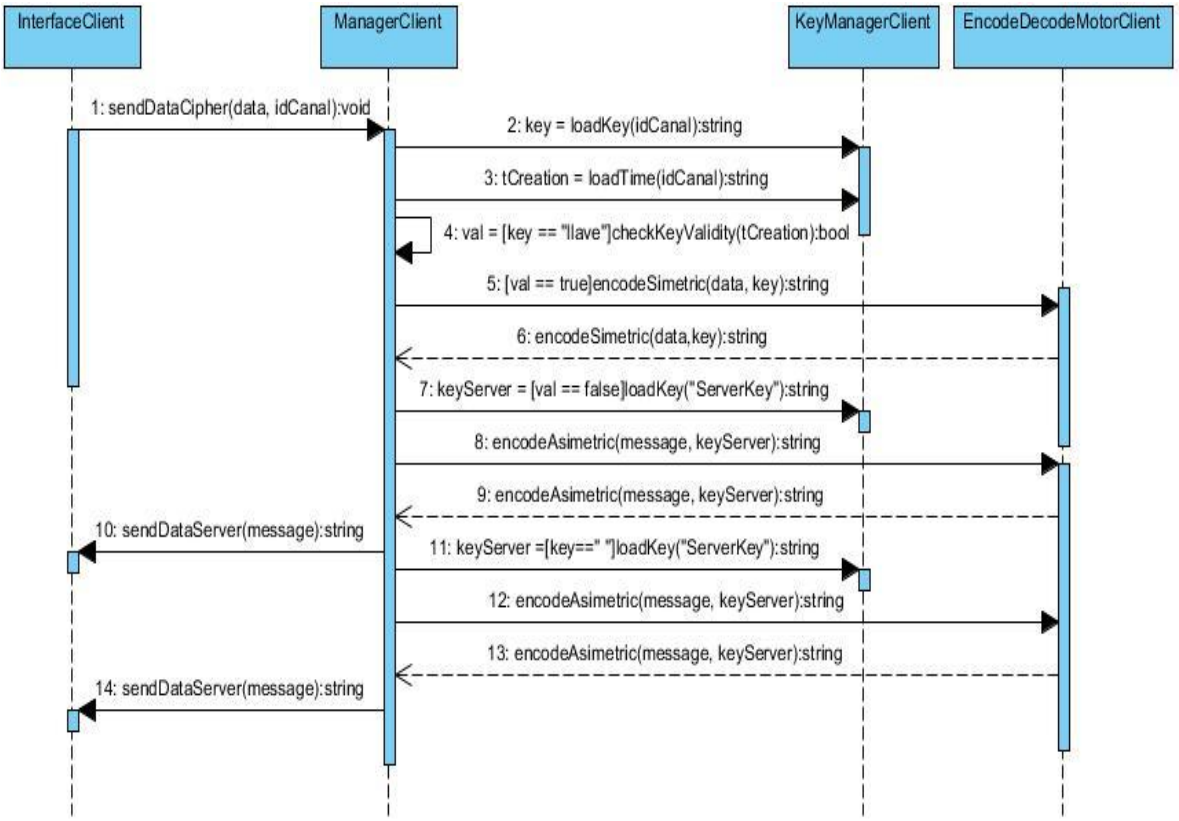
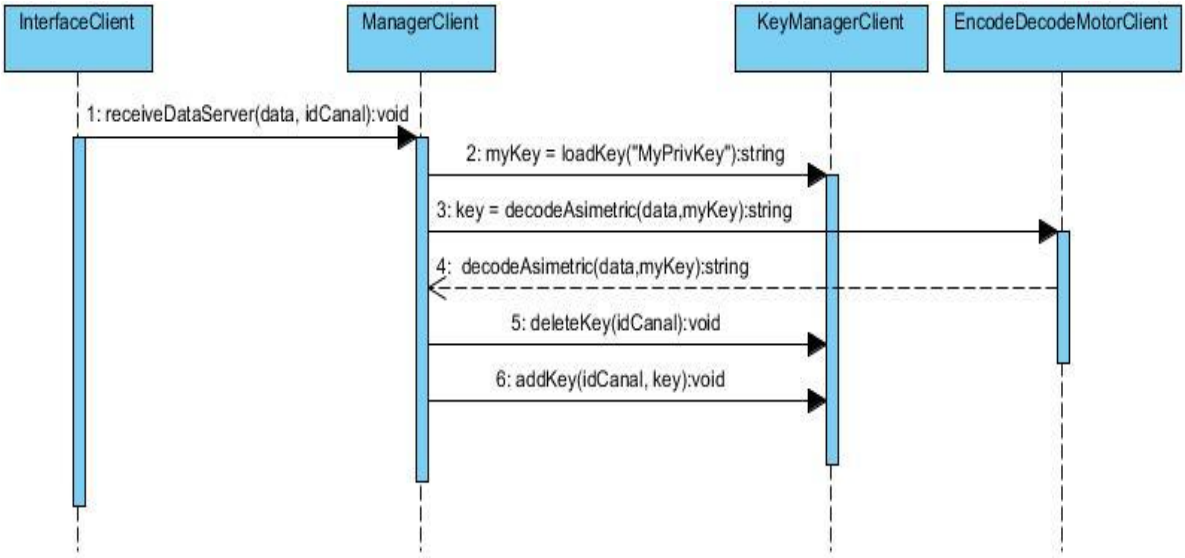
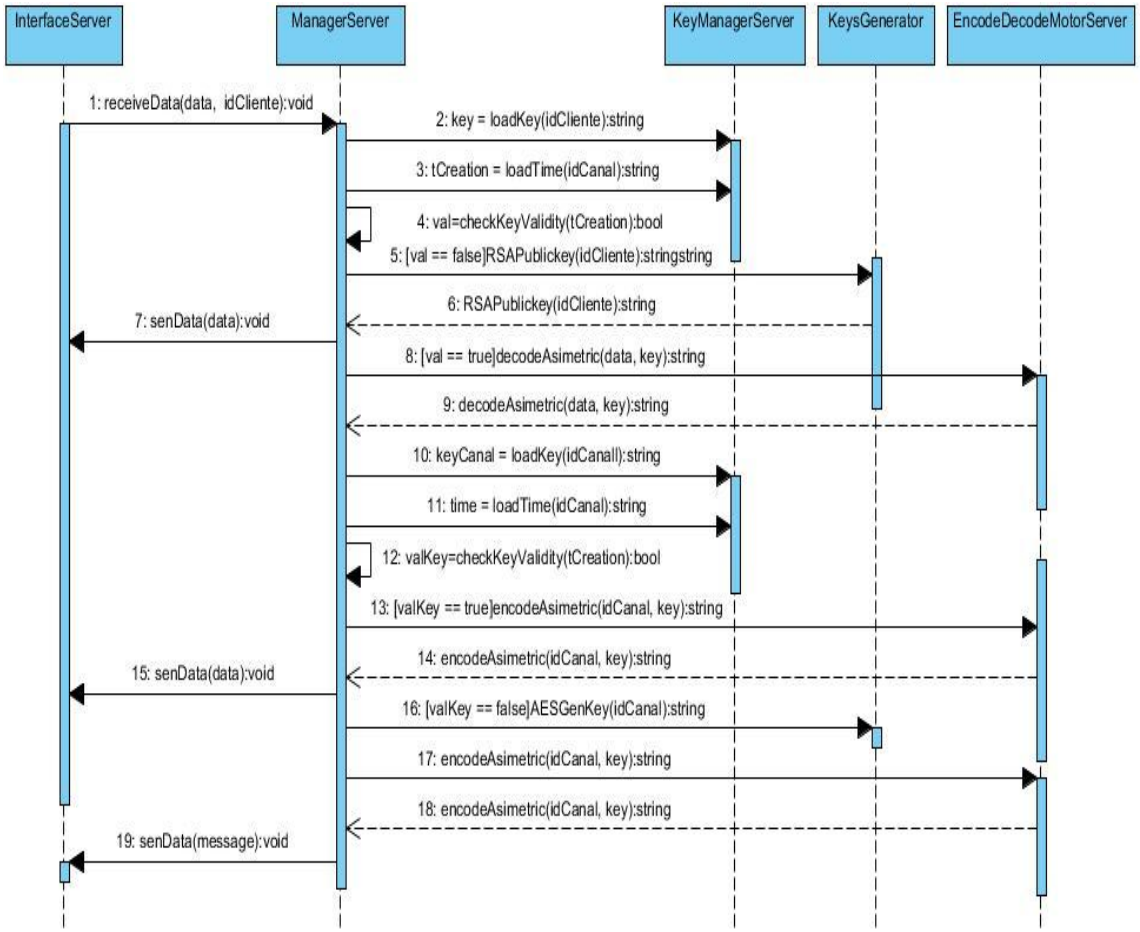


Figura 22: Diagrama de secuencia para procesar respuesta de solicitud de llave



3.3.3.2 Diagramas de secuencia del EncryptionServer

Figura 23: Diagrama de secuencia para procesar solicitud de llave del cliente



3.4 Patrones de diseño

Los patrones de diseño tienen diferentes clasificaciones en dependencia de su funcionalidad.

- **Patrones Creacionales:** Muestran la guía de cómo crear objetos cuando sus creaciones requieren tomar decisiones. Estas decisiones, normalmente son resueltas dinámicamente, decidiendo qué clases instanciar o sobre qué objetos u objeto delegar responsabilidades.
- **Patrones estructurales:** Describen la forma en que diferentes tipos de objetos

pueden ser organizados para trabajar unos con otros.

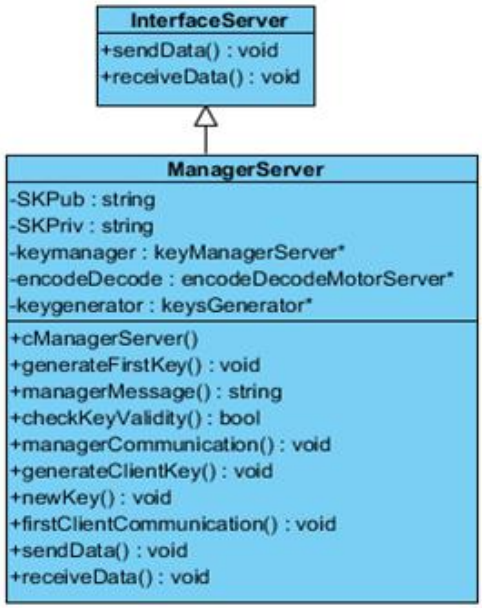
- **Patrones de comportamiento:** Se utilizan para organizar, manejar y combinar comportamientos.

En ambas bibliotecas que componen la aplicación, con el objetivo de garantizar una interfaz de acceso al sistema, se utilizó el patrón estructural Facade o Fachada. Como patrón creacional se utilizó el Singleton o instancia única para restringir la instanciación del acceso a la base de datos.

3.4.1 Patrón Facade

El patrón de diseño Facade proporciona una interfaz unificada que, representando a todo un subsistema, facilita su uso. En este caso, como se trata de bibliotecas, se hace necesario restringir el acceso de usuarios a funcionalidades propias del sistema, por lo tanto, se brinda una fachada para limitar el acceso a funcionalidades que no son de su interés, ocultando así detalles de implementación.

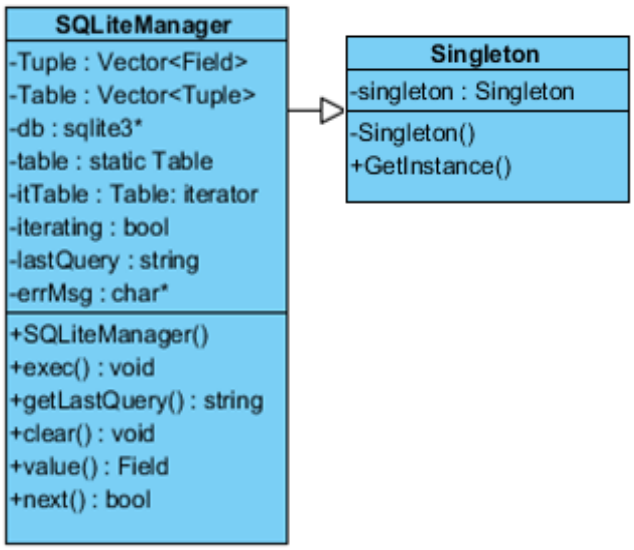
Figura 24: Patrón de diseño Facade



3.4.2 Patrón Singleton

El patrón de diseño Singleton se encarga de restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. Su intención consiste en garantizar que una clase sólo tenga una instancia y proporcionar un punto de acceso global a ella. Varias funcionalidades distintas pudieran necesitar el acceso a la base de datos, con el uso de este patrón aseguramos de que se acceda a sólo una instancia de esta.

Figura 25: Patrón de diseño Singleton



Capítulo 4: Implementación y pruebas del sistema.

4.1 Estándar de codificación

Los estilos de codificación y documentación explicados en este capítulo son aplicados en el Centro de Informática Industrial, esto justifica el empleo de los mismos en el Sistema para la Seguridad de las Comunicaciones del SCADA "Guardián del ALBA". Los estilos utilizados evidencian la intención de lograr una aplicación que pueda ser comprendida sin necesidad de tener un amplio conocimiento respecto al tema.

4.1.1 Nombres

- Los nombres de las clases son sustantivos singulares.
- Los nombres de clases y objetos deben reflejar qué hacen y no cómo lo hacen.
- Escoger nombres lo suficientemente largos que expresen correctamente el sentido de lo que se quiere, pero evitando manejar nombres que dificulten la labor de implementación.
- Evitar nombres que permitan una interpretación subjetiva (evitar ambigüedad y asegurar abstracción).
- Evitar redundancia no repitiendo nombres de clases en sus elementos.
- Dado que los nombres generalmente son el producto de concatenar varias palabras, se debe emplear la primera palabra en minúscula, mayúscula para denotar la letra de inicio de cada una de las palabras restantes por las que esté formado y minúscula para las letras intermedias en el caso de los nombres de métodos y funciones. Para el caso de los nombres de variables y atributos debe aplicarse la misma convención.
- Las variables booleanas deben contener la palabra is en su nombre.
- Los nombres de constantes deben contener solo letras mayúsculas.
- Minimizar el uso de abreviaturas. En caso de ser requeridas, se debe ser consistente en su uso y cada abreviatura debe significar solo una cosa.

- Los nombres de los métodos son frases que incluyen verbos.
- Los nombres de los atributos y parámetros son frases con sustantivos.

4.1.2 Codificación

- Se establece un tamaño de indentación[15] estándar de tres espacios, sin tabulaciones.
- Alinear secciones del código.
- Alinear verticalmente llaves de apertura y cierre.
- Usar espacios antes y después de los operadores que el lenguaje de programación permita.
- Emplear líneas en blanco para organizar el código, permitiendo crear párrafos de código para una mejor lectura.
- Evitar colocar más de una sentencia por línea.
- Emplear constantes en sustitución de números o cadenas de caracteres literales.
- Minimizar el alcance de las variables para evitar confusión y facilitar el mantenimiento.
- Emplear cada variable y rutina solo para un propósito.
- Evitar el uso de variables públicas, sustituirlas por variables privadas y métodos que provean el valor de tal variable, para mantener el encapsulamiento.
- Emplear las letras i, j, k, l, m, p, q, r para contadores en ciclos.
- Mantener la modularidad del código bajo el criterio de la lógica que encierra, no exagerar la modularidad.
- Emplear correctamente los tipos de ciclos: si es al menos una vez usar do-while, si es ninguna o más veces usar while-do, y si se conoce el número exacto de ciclos usar for.

[15] Espacio o sangría que se pone a la derecha de cada línea de código.

- Inicializar todas las variables.
- Emplear líneas en blanco para separar pasos lógicos (Ej.: declaraciones, lazos).

4.2 Estándar de documentación.

Se utilizó la herramienta Doxygen para generar la documentación del código de las distintas clases utilizadas para la implementación de las bibliotecas, los formatos brindados por esta herramienta se explican a continuación.

4.2.1 Estilo de bloques de documentación.

Se adopta el estilo de bloques de documentación JavaDoc, el cual consiste en un bloque de comentario de estilo C. Este bloque de comentario comienza con dos asteriscos, los asteriscos que se encuentran en la línea de la mitad son opcionales. Ejemplo:

```
/**
 * Texto
 */
```

4.2.2 Descripción breve con el comando \brief o @brief.

Para hacer una descripción breve se adopta el uso del comando \brief o @brief en el bloque de comentarios ya descrito.

La acción de este comando termina al final de un párrafo, de tal manera que la descripción detallada sigue después de una línea vacía, tal como lo muestra el ejemplo:

```
/**
 * @brief descripción breve.
 * Continuación de la descripción breve.
 *
```

```
* La descripción detallada comienza aquí, nótese
* que se debe dejar una línea en blanco para lograr
* tener las dos descripciones (breve y detallada)
*/
```

Nota: Si no se utiliza el comando @brief, Doxygen toma la descripción hecha en el bloque como una descripción detallada y no como una descripción breve.

4.2.3 Descripción de argumentos y métodos.

La descripción de argumentos de métodos y funciones se encuentra en línea, es decir, luego de la declaración de cada uno de los argumentos se da una breve descripción de cada uno de ellos. En el siguiente ejemplo se muestra el formato a utilizar:

```
int multFunction (      int c , /**<Primer operando de la operación */
                        int d , /**<Segundo operando de la operación */
                        int e /**<Tercer operando de la operación */ );
```

4.2.4 Documentación de tipos de datos.

Para describir la función y los elementos que componen los tipos de datos definidos se pueden utilizar los formatos especificados en los apartados anteriores. A continuación se muestra un ejemplo:

```
/**
* @brief Enumerado de los tipos de datos admitidos
*
* Descripción más detallada de la función de este tipo de dato.
*/ enum DataTypes{  INTEGER, /**<Puede ser un valor de tipo entero */
                   DOUBLE, /**<Puede ser un valor de tipo double */
```

STRING /**<Puede ser un valor de tipo string */ };

Observamos que se genera una descripción breve seguida de una descripción más detallada para la función, luego se explica brevemente el valor de retorno de la misma y la descripción de los parámetros usando el formato de documentación en línea.

4.2.5 Comandos @author y @date.

Es importante que se especifique el nombre del autor y la fecha de creación de cualquier estructura en un código, para ello se utilizan los comandos @author y @date, para el nombre del autor y la fecha respectivamente, en el siguiente ejemplo se puede ver la acción de estos comandos:

```
/**@brief Descripción breve
*
* Aquí comienza la descripción detallada
* @author Daisy Diana Vargas
* @date 13-04-2011
*/
```

4.2.6 Comando @see.

Existe otro comando útil que permite hacer referencias a otras clases o métodos cuando esto sea necesario, es importante usar este comando en la documentación a la hora de implantar los métodos o funciones propias de alguna clase, este comando es @see y su uso se muestra en el siguiente ejemplo:

```
/**
* Constructor de la clase
* @see Test::Test()
```

```
*/
Test1();
```

Esto genera en la documentación para el constructor de la clase de prueba Test1 una referencia hacia la documentación existente para el constructor de la clase de prueba Test.

Nota: Si se quiere hacer una referencia desde cualquier estructura hacia la documentación completa de una clase, se debe utilizar el comando @see seguido del nombre de la clase de esta forma:

```
/**
 * Constructor de la clase
 * @see Clase
 */
Test1();
```

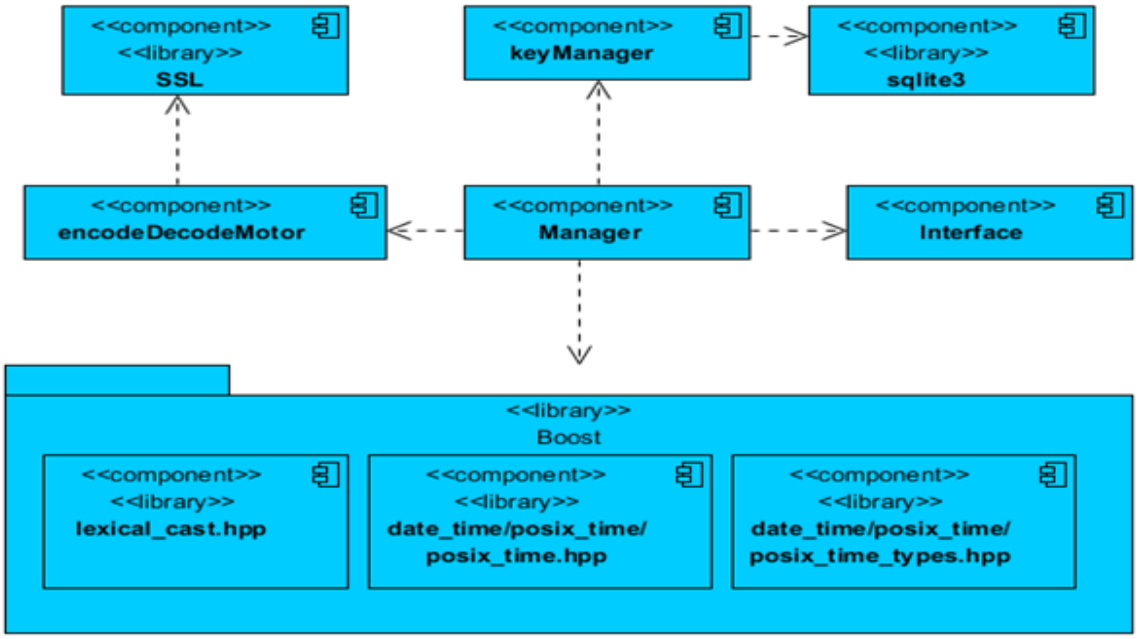
4.3 Implementación.

4.3.1 Diagrama de componentes.

Un diagrama de componentes ilustra los fragmentos de software, controladores embebidos, etc. que conforman un sistema. Este tipo de diagrama tiene un nivel de abstracción más elevado que un diagrama de clases. Usualmente un componente se implementa por una o más clases (u objetos) en tiempo de ejecución. (Guía de Usuario de Enterprise Architect.)

Un componente es una pieza de software que describe un conjunto de servicios que son usados sólo por medio de interfaces bien definidas. El siguiente diagrama (Figura 29) describe las interacciones existentes entre los principales componentes que integran la solución, así como sus dependencias más significativas.

Figura 26: Diagrama de componentes



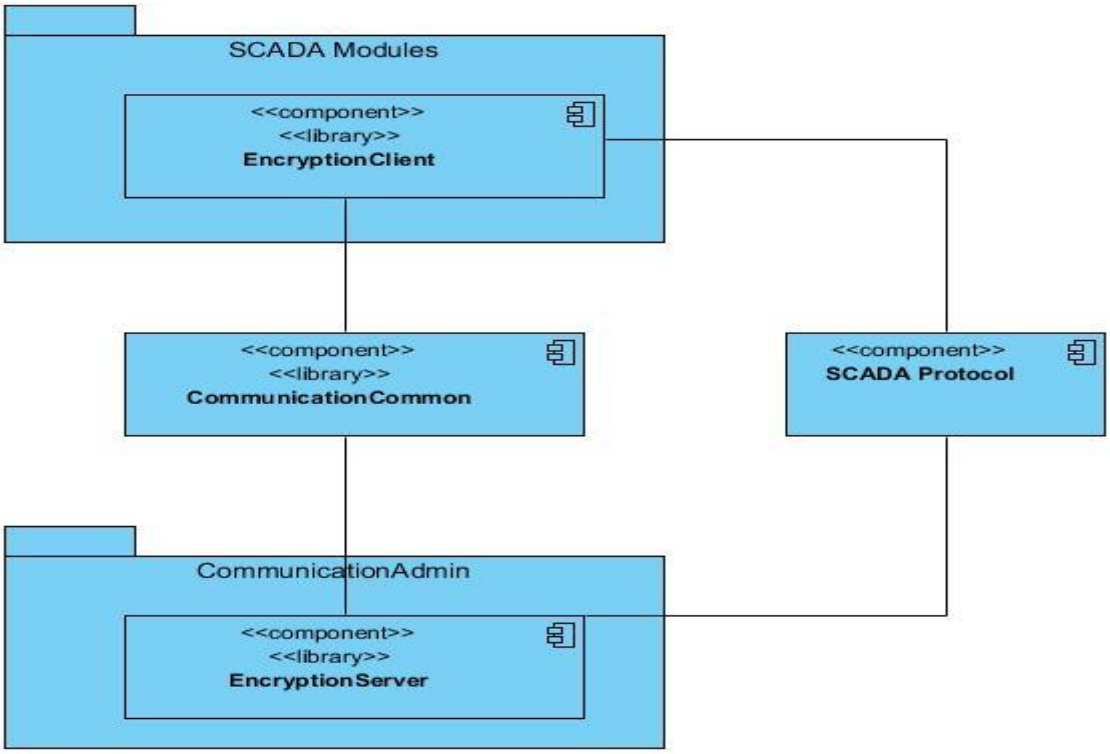
4.3.2 Diagrama de integración

La biblioteca EncryptionServer se encontrará empotrada en el componente CommunicationAdmin del SCADA "Guardián del ALBA", garantizando así que la comunicación y la distribución de las listas de servidores y clientes se haga en modo seguro. Los clientes y servidores del SCADA tendrán empotrado un EncryptionClient que se encargará del cifrado y descifrado de la información que llega o es emitida.

Ambas bibliotecas para el cifrado de datos se relacionarán directamente con la biblioteca CommunicationCommon quien será la encargada de gestionar la comunicación entre los clientes y el servidor. El cifrado de los datos se realizará justo después de que sea estandarizada la información a enviar en el SCADAProtocol. De la misma forma el cliente de cifrado que recibe la información descifrará esta justo antes de ser manipulada por el SCADAProtocol.

El siguiente diagrama muestra la integración de la solución en el SCADA "GALBA".

Figura 27: Diagrama de integración de la solución propuesta



4.3.3 Diagrama de despliegue.

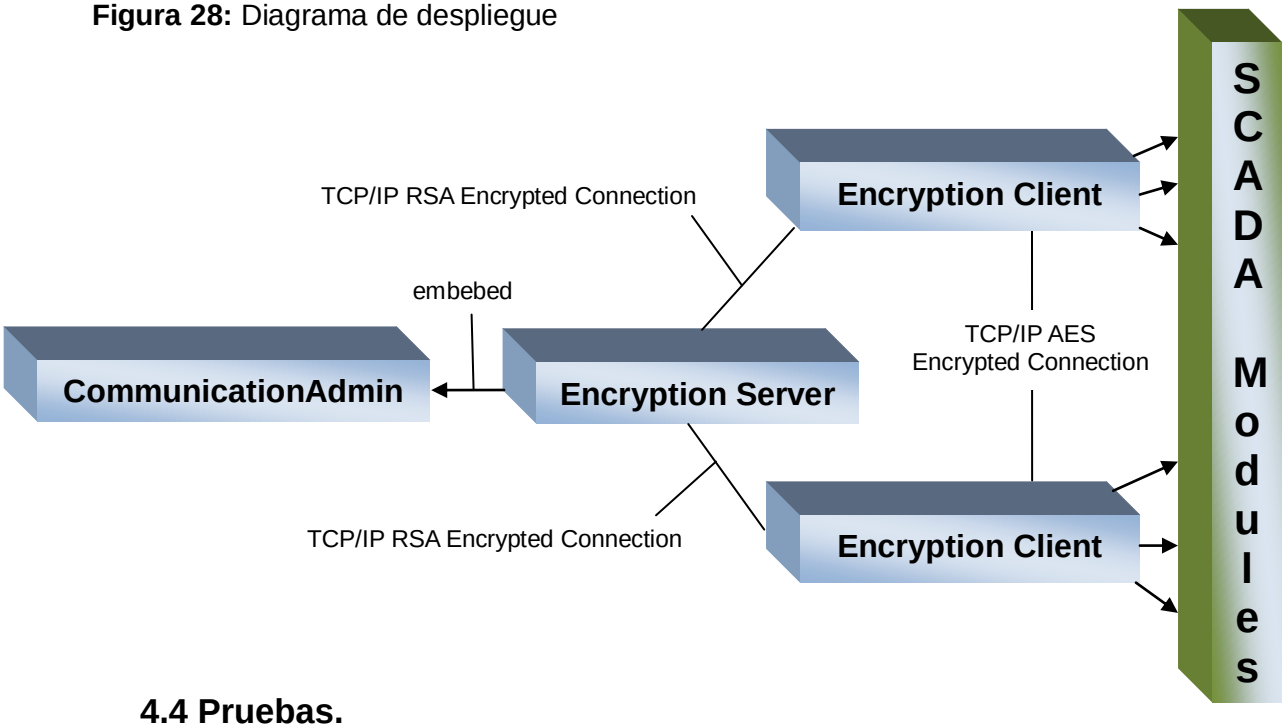
Es un tipo de diagrama del Lenguaje Unificado de Modelado (UML) que se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes.

En el diagrama de despliegue del sistema se muestra como tanto los servidores como los clientes del SCADA tendrán embebida la biblioteca EncryptionClient, estos se comunicarán con el EncryptionServer vía TCP/IP usando RSA como método de cifrado para la implementación de la conexión segura (cifrado asimétrico). De la misma forma el EncryptionServer se encuentra embebido en el componente CommunicationAdmin del SCADA.

La comunicación entre los clientes de cifrado se realizará mediante conexión TCP/IP,

cifrada con AES como algoritmo de cifrado (cifrado simétrico).

Figura 28: Diagrama de despliegue



4.4 Pruebas.

La realización de pruebas es una actividad en la cual un sistema o componente es ejecutado bajo ciertas condiciones o requisitos específicos, cuyos resultados son observados, registrados y evaluados. Estas técnicas ayudan a definir un conjunto de casos de prueba aplicando un cierto criterio, estos son determinados por los valores a asignar a las entradas en su ejecución, realizando de esta forma un conjunto de pruebas unitarias a cada uno de los componentes del sistema atendiendo a las funcionalidades a cumplir por este.

Las pruebas unitarias en su enfoque funcional o de caja negra permiten comprobar el correcto funcionamiento de cada uno de los componentes del sistema, analizando las entradas y salidas y verificando que el resultado es el esperado. Se consideran exclusivamente las entradas y salidas del sistema sin tener en cuenta la estructura interna del mismo. Este método permite validar el software, entendiendo como validación el proceso que determina si el software satisface los requisitos anteriormente

definidos.

4.4.1 Descripción de los casos de prueba.

A continuación se detallan los casos de prueba (CP) elaborados por cada una de los requisitos funcionales para comprobar el correcto funcionamiento del sistema bajo las diferentes condiciones a las que puede someterse.

4.4.1.1 Descripción de los casos de prueba del EncryptionClient.

Tabla 5: CP 1. Cifrar datos

Escenario	Parámetros		Respuesta del Sistema	Resultados de la Prueba
	data	idCanal		
CP 1. Cifrar un texto claro entrado al sistema.	Alarma activada	Alarma	Texto cifrado[16]	Prueba exitosa

Tabla 6: CP 2. Descifrar datos

Escenario	Parámetros		Respuesta del Sistema	Resultados de la Prueba
	data	idCanal		
CP 2. Hallar el texto claro a partir de un texto cifrado entrado al sistema.	Texto cifrado	Alarma	Alarma activada	Prueba exitosa.

[16] Secuencia de caracteres ininteligible que representan el texto cifrado.

Tabla 7: CP 3. Almacenar llave

Escenario	Parámetros		Respuesta del Sistema	Resultados de la Prueba
	id	key		
CP 3. Almacenar una llave en el mapa con su fecha y hora de almacenamiento.	Alarma	8defb3dc0afe988e0ae4ca855e025b881b105d96231ff1832497daac8b663609	Llave almacenada correctamente.	Prueba exitosa.

Tabla 8: CP 4. Eliminar llave

Escenario	Parámetros	Respuesta del Sistema	Resultados de la Prueba
	id		
CP 4. Eliminar una llave almacenada.	Alarma	Llave eliminada correctamente.	Prueba exitosa.

Tabla 9: CP 6. Cargar primera llave

Escenario	Respuesta del Sistema	Resultados de la Prueba
CP 5. Cargar llave de un fichero de texto y almacenarla en el mapa.	Llave cargada y almacenada correctamente.	Prueba exitosa.

Tabla 10: CP 5. Chequear validez de la llave

Escenario	Parámetros	Respuesta del Sistema	Resultados de la Prueba
	id		
CP 6.1 Chequear validez de una llave con tiempo de creación de 2 horas.	Alarma	Llave válida	Prueba exitosa.
CP 6.2 Chequear validez de una llave con tiempo de creación de 2 días.	Punto	Llave válida	Prueba fallida.

4.4.1.2 Descripción de los casos de prueba del EncryptionServer.

Tabla 11: CP 7. Cifrar datos

Escenario	Parámetros		Respuesta del Sistema	Resultados de la Prueba
	data	key		
CP 7. Cifrar un texto claro.	Punto	8defb3dc0afe988e0ae4ca855e025b881b105d96231ff1832497daac8b663609	Texto cifrado	Prueba exitosa.

Tabla 12: CP 8. Descifrar datos

Escenario	Parámetros		Respuesta del Sistema	Resultados de la Prueba
	data	key		
CP 8. Descifrar texto.	Texto cifrado	8defb3dc0afe988e0ae4ca855e025b881b105d96231ff1832497daac8b663609	Punto	Prueba exitosa.

Tabla 13: CP 9. Generar llave

Escenario	Respuesta del Sistema	Resultados de la Prueba
CP 9.1 Genera llave simétrica	Llave generada correctamente	Prueba exitosa.
CP 9.2 Genera llave asimétrica pública.	Llave generada correctamente	Prueba exitosa.
CP 9.3 Genera llave asimétrica privada.	Llave generada correctamente	Prueba exitosa.

Tabla 14: CP 10. Almacenar llave

Escenario	Parámetros		Respuesta del Sistema	Resultados de la Prueba
	id	key		
CP 10. Almacenar una llave en la base de datos con su fecha y hora de almacenamiento.	Alarma	8defb3dc0afe988e0ae4ca855e025b881b105d96231ff1832497daac8b663609	Llave almacenada correctamente.	Prueba exitosa.

Tabla 15: CP 11. Eliminar llave

Escenario	Parámetros	Respuesta del Sistema	Resultados de la Prueba
	id		
CP 11. Eliminar una llave almacenada.	Alarma	Llave eliminada correctamente.	Prueba exitosa.

Tabla 16: CP 12. Chequear validez de la llave

Escenario	Parámetros	Respuesta del Sistema	Resultados de la Prueba
	id		
CP 12.1 Chequear validez de una llave con tiempo de creación de 2 horas.	Alarma	Llave válida	Prueba exitosa.
CP 12.2 Chequear validez de una llave con tiempo de creación de 2 días.	Punto	Llave inválida	Prueba exitosa.

De los resultados obtenidos a partir de la aplicación de los diferentes casos de prueba efectuados a las funcionalidades del sistema se deriva la solidez del mismo. De un total de 16 pruebas realizadas al software, 15 resultaron satisfactorias y sólo 1 reveló vulnerabilidades. El error detectado fue solucionado, lo cual demuestra que el proceso de validar la primera versión funcional del sistema concluye de manera exitosa.

Conclusiones generales

Luego de terminado el proceso de investigación, desarrollo y validación de la aplicación realizada, se arriba a las siguientes conclusiones:

- Mediante la aplicación de técnicas criptográficas se logra reducir el nivel de vulnerabilidades en las comunicaciones del sistema de seguridad del SCADA "Guardián del ALBA".

- La aplicación de un sistema criptográfico híbrido, entre el algoritmo simétrico AES y el asimétrico RSA, permite alcanzar seguridad en las comunicaciones del SCADA "Guardián del ALBA" sin afectar su disponibilidad y funcionamiento en tiempo real.

- La aplicación del algoritmo AES para el cifrado de la información intercambiada entre los módulos que componen el SCADA y del algoritmo RSA para la seguridad de las llaves de comunicación, garantiza un nivel adecuado de seguridad aprovechando las ventajas de los sistemas criptográficos existentes.

- Se obtuvo la primera versión de un componente de cifrado de datos bien documentado que provee al SCADA "Guardián del ALBA" de seguridad en el intercambio de información entre los módulos que lo integran.

Recomendaciones

- Ampliar las funcionalidades del sistema con el desarrollo de una Infraestructura de Llave Pública (PKI) que permita la autenticación de usuarios y módulos utilizando Certificados Digitales.

- Incorporar al sistema la posibilidad de configurar elementos, para el incremento de la seguridad, como el tamaño y el período de validez de las llaves de cifrado.

Referencias bibliográficas

1. **[Rubinos Carvajal, 2009].** A. M. Carvajal. (2009). Subsistema de Seguridad para el SCADA Nacional "Guardián del ALBA". Universidad de las Ciencias Informáticas. Cuba.
2. **[Rubinos Carvajal & Reyes Bermúdez, 2010].** A. M. Rubinos Carvajal, E. Reyes Bermúdez. (2010). Arquitectura del software Seguridad en las Comunicaciones. Universidad de las Ciencias Informáticas. Cuba.
3. **[Organización Autómatas, 2009].** Organización Autómatas. Sistemas SCADA. [Citado el: 21 de diciembre de 2009]. [En línea] 5 de Noviembre de 2010. Obtenido de: http://www.automatas.org/redes/scada.html
4. **[Penín, 2007].** A. R. Penín. (2007). Sistemas SCADA. Macombo: 2da edición. S.L. Barcelona, España.
5. **[S21sec, 2007].** S21sec, Web de Seguridad. Sugerencias para mejorar la seguridad en SCADA. [Citado el: 10 de Enero de 2007]. [En línea] 7 de Noviembre de 2010. Obtenido de: http://www.s21sec.com
6. **[Ramió, 2006].** J. A. Ramió. (2006). Libro electrónico de seguridad informática y criptografía. Madrid, España. Universidad Politécnica de Madrid. Sexta edición.
7. **[Vega Cutiño, 2010].** R. Y. Vega Cutiño. (2010). Fundamentos básicos de la criptografía. Universidad de las Ciencias Informáticas. Cuba. Disponible en el Entorno Virtual de Aprendizaje (EVA).
8. **[Veyrat Marqués, 2007].** A. Veyrat Marqués. (2007). Las medidas de seguridad de nivel alto. Manaca Consulting S.L.
9. **[López López, 2005].** E. E. López López. (2005). Análisis de métodos de encriptación y seguridad en redes. Guatemala: Trabajo de tesis para optar por el título de Ingeniero.
10. **[Pastor Franco & Sarasa López, 2005].** J. Pastor Franco, M. Sarasa López. (2005). Criptografía digital. Fundamentos y aplicaciones. Prensas Universitarias de Zaragoza. 4ta Edición.
11. **[Ruiz Múzquiz, 2004]** P. Ruiz Múzquiz. (2004). Sistemas Operativos. Versión 0.5. Madrid.

12. **[HispaLinux, 2007].** HispaLinux, Sociedad del Conocimiento Libre. ¿Qué es el software libre? [Citado el: 27 de Marzo de 2007]. [En línea] 24 de Febrero de 2011. Obtenido de: http://hispanilinux.es/SoftwareLibre

13. **[González Cornejo, 2009].** J. E. González Cornejo. ¿Qué es UML? [Citado el: 16 de Octubre de 2009]. [En línea] 27 de Marzo de 2011. Obtenido de: http://www.docirs.cl/uml.htm

14. **[Proyecto Zero, 2010].** José Alberto Fuentes Sánchez, María de los Ángeles Santos Oliva. Proyecto Zero. Modelado UML. [En línea] 2 de Marzo de 2011. Obtenido de: https://forja.rediris.es/docman/view.php/282/444/uml20.pdf.

15. **[Florentino, 2007].** Deidry Florentino. (Marzo, 2007). Historia del computador y evolución, arquitectura, lenguajes de programación y algoritmos. [En línea] 18 de Febrero de 2011. Obtenido de: http://nectacellcomputer.jimdo.com/historia-del-computador-arquitectura-lenguajes-de-programaci%C3%B3n-y-algoritmos/

16. **[Calvo, 2006].** Luis Calvo. Lenguaje de Programación. [Citado el: 22 de Abril de 2006]. [En línea] 23 de Febrero de 2011. Obtenido de: http://www.scribd.com/doc/Lenguaje-de-Programacion.

17. **[Blanco Cuaresma, 2008].** Sergi Blanco Cuaresma. Marble Station. Metodologías de Desarrollo. [Citado el: 14 de Febrero de 2008]. [En línea] 19 de Febrero de 2011. Obtenido de: http://www.marblestation.com/?p=644

18. **[Bello Lobato, 2008].** Víctor Bello Lobato. Ingeniería de software I. [Citado el: 5 de junio de 2008]. [En línea] 23 de Febrero de 2011. Obtenido de: http://www.scribd.com/doc/3062020/Capitulo-I-HERRAMIENTAS-CASE

19. **[Sambayón Group, 2008].** Sambayón Group. (Abril, 2008). Introducción a RUP. Versión 0.1. Obtenido de: http://svn2.assembla.com/svn/tdp2/Introduccion%20a%20RUP.doc

20. **[Download Manager].** Free Download Manager. Visual Paradigm for UML. [En línea] 17 de Febrero de 2011. Obtenido de: http://www.freedownloadmanager.org/ParadigmaUML

21. **[Universidad de Oviedo, 2009].** Sitio web de la E.U. de Ingeniería Técnica Informática de Oviedo. (2009). Entornos de Desarrollo Integrado. [En línea] 01 de Marzo de 2011.

Obtenido de: http://petra.euitio.uniovi.es/Entornos%20de%20Desarrollo%20Integrado.pdf

22. **[Sitio oficial SQLite].** Sitio oficial SQLite. [En línea] 17 de Febrero de 2011. Obtenido de: http://www.sqlite.org

23. **[Sorio, 2010].** R. P. Sorio. (2010). Sistema de Vigilancia por Cámaras Cancerbero (SVCC). Trabajo de Diploma. Universidad de las Ciencias Informáticas. Cuba.

24. **[Mesa Valiente & Castillo Pérez, 2008].** R. Mesa Valiente, I. Castillo Pérez. (2008). LIMS de Calidad del Centro de Ingeniería Genética y Biotecnología: Desarrollo de la Base de Datos del Módulo Sección de Mejoramiento de la Calidad. Trabajo de Diploma. Universidad de las Ciencias Informáticas, Cuba.

25. **[Architect, G. d, 2007].** Architect, G. d. (2007) Guía de usuario de Enterprise Architect. [En línea] 6 de Abril de 2011. Obtenido de: http://www.sparxsystems.com.ar/index.php.

Bibliografía

1. E. Reyes Bermúdez (2009). Sistemas de autenticación para el módulo Seguridad del proyecto Guardián del ALBA. Trabajo de Diploma. Universidad de las Ciencias Informáticas. Cuba.
2. A. M. Rubinos Carvajal, E. Reyes Bermúdez, H. A. Nuevo León. (Septiembre, 2010). Variante para garantizar la seguridad en las comunicaciones del sistema SCADA. Universidad de las Ciencias Informáticas. Cuba.
3. Idaho National Laboratory. INL increases infrastructure security at SCADA. [En línea] 11 de Diciembre de 2010. Obtenido de: http://www.inl.gov/featurestories/2006-09-28.shtml
4. IT Security Expert Advisory Group. SCADA Security-Advice for CEOs. [En línea] 7 de Diciembre de 2010. Obtenido de: http://www.ag.gov.au/agd/WWW/rwpattach.nsf/VAP
5. National Infrastructure Security Co-ordination Centre (NISCC). (Febrero, 2005). Good Practice Guide on Firewall Deployment for SCADA and Process Control Networks. British Columbia Institute of Technology (BCIT). [En línea] 28 de Noviembre de 2010. Obtenido de: http://www.niscc.gov.uk/niscc/docs/re-20050223-00157.pdf
6. Keith Stouffer, Joe Falco, Karen Kent. Guide to Supervisory Control and Data Acquisition (SCADA) and Industrial Control Systems Security. [En línea] 28 de Noviembre de 2010. Obtenido de: http://csrc.nist.gov/publications/drafts/800-82/Draft-SP800-82.pdf
7. Ronald L. Krutz. Securing SCADA systems. [Citado el: 31 de Octubre de 2007]. [En línea] 21 de Enero de 2011. Obtenido de: http://ebookee.org/Securing-SCADA-Systems-by-Ronald-L-Krutz_137037.html
8. Sergio Hernando. Más sobre seguridad en sistemas SCADA. [Citado el: 4 de diciembre de 2006]. [En línea] 21 de Enero de 2011. Obtenido de: http://www.sahw.com/ mas-sobre-seguridad-en-sistemas-scada
9. Steven S. Smith. The SCADA Security Challenge: The Race Is On. [Citado el: 25 de noviembre de 2006]. [En línea] 15 de Enero de 2011. Obtenido de: http://www.infosecwriters.com/texts.php

10. Ricardo Cañizares Sales. (Mayo 2008). Revista Hoy Hablamos De. La seguridad en las comunicaciones.

11. Colectivo de autores. (2007). Redes. Tema 8: Seguridad en las comunicaciones. Universidad de Oviedo

12. Kioskea. Cifrado por medio de RSA. [Citado el: 16 de octubre de 2008]. [En línea] 20 de Enero de 2011. Obtenido de: http://es.kioskea.net/crypto/Cifrado+por+medio+de+RSA.

13. Angélica García. Channel Planet. Nuevo estándar de cifrado AES de 128-bit, 192-bit y 256-bit de longitud. [Citado el: 3 de enero de 2006]. [En línea] 20 de Diciembre de 2011. Obtenido de: http://www.channelplanet.com/index.php?idcategoria=9749

14. Carlos Neira. Kriptápolis. Comparativa de AES vs. 3DES. [Citado el: 8 de julio de 2006]. [En línea] 5 de Enero de 2011. Obtenido de: http://www.kriptopolis.org/comparativa-de-aes-vs-3des

15. Antonio Paredes. Tecnología 21. Cifrado RSA de 1024 bits, roto. [Citado el: 5 de marzo de 2010]. [En línea] 5 de Enero de 2011. Obtenido de: http:\\www.tecnologia21.com\cifrado-rsa-1024-bits.htm

16. Cecilia Saia. Logran romper cifrado RSA de 1024 bits. [Citado el: 5 de marzo de 2010]. [En línea] 6 de Enero de 2011. Obtenido de: http://www.biteliat.com

17. Marcos Dacosta Balboa. Cifrado RSA de 1024 bits crakeado, [Citado el: 6 de marzo de 2010]. [En línea] 6 de Enero de 2011. Obtenido de: http://www.linwind.com

18. W. Stallings. (1997). Comunicaciones y redes de computadoras. 5ta Edición. Prentice Hall Iberia.

19. A. S. Tanenbaum. (1996).Computer Networks. (3ra Edición). Prentice-Hall.

20. F. Halsall. (1995). Data Encryptation, Computer Networks and Open Systems. Addison-Wesley Iberoamericana.

21. J. Freer. (1988). Introducción a la tecnología y diseño de Sistemas de Comunicaciones y Redes de Ordenadores. Anaya Multimedia.

22. J. M. Alonso. (1996). Protocolos de comunicaciones para sistemas abiertos. Addison-

Wesley Iberoamericana.

23. Aurora Vizcaíno, Félix O. García, Ismael Caballero. (2009). Ingeniería del Software 1. Práctica 1. Trabajando con Visual Paradigm para UML. Universidad de Cantabria. Facultad de Ciencias. María Sierra.

24. Iván Tapia Moreno. (2009). Flujo de Trabajo: Análisis y Diseño. Administración de Proyectos de Desarrollo de Software. Universidad de las Ciencias Informáticas, Cuba.

25. Daniel Perovich, Andrés Vignaga. (2004). Rational Unified Process. Curso Arquitectura de Software. Instituto de Computación. Facultad de Ingeniería. Universidad de la República, Uruguay.

26. Karenny Brito Acuña. (2008). Selección de metodologías de desarrollo para aplicaciones web en la Facultad de Informática de la Universidad de Cienfuegos. Biblioteca virtual de derecho, economía y ciencias sociales.

27. Oscar Hernando Guzmán Cortés. (2004). Aplicación práctica del diseño de pruebas de software a nivel de programación. Universidad ICESI. Sistemas & Telemática.

Anexos

Anexo 1. Descripción de las clases del subsistema EncryptionServer.

Tabla 17: Descripción de la clase EncodeDecodeMotorServer

Nombre: EncodeDecodeMotorServer	
Descripción: Es la clase donde residen las funcionalidades de cifrado y descifrado de datos.	
Nombre:	EncodeDecodeMotorServer()
Descripción:	Constructor por defecto de la clase.
Nombre:	~EncodeDecodeMotorServer()
Descripción:	Destructor de la clase.
Nombre:	encodeSimetric(std::string text, std::string key)
Descripción:	Permite el cifrado simétrico de los datos utilizando el algoritmo AES. Esta operación la delega en la clase ManagerServer.
Nombre:	encodeAsimetric(std::string text, std::string publicKey)
Descripción:	Permite el cifrado asimétrico de los datos utilizando el algoritmo RSA. Esta operación la delega en la clase ManagerServer.
Nombre:	decodeSimetric(std::string text, std::string key)
Descripción:	Permite el descifrado simétrico de los datos utilizando el algoritmo AES. Esta operación la delega en la clase ManagerServer.
Nombre:	decodeAsimetric(std::string text, std::string privateKey)
Descripción:	Permite el descifrado asimétrico de los datos utilizando el algoritmo RSA. Esta operación la delega en la clase ManagerServer.
Nombre:	auxSha256(std::string text)
Descripción:	Método auxiliar de la clase que devuelve una función resumen del texto utilizando el algoritmo SHA 256.

Tabla 18: Descripción de la clase KeysGenerator

Nombre: KeysGenerator	
Descripción: Es la clase donde residen las funcionalidades de generación de llaves para el cifrado y descifrado de los datos.	
Nombre:	KeysGenerator()
Descripción:	Constructor por defecto de la clase.
Nombre:	~KeysGenerator()
Descripción:	Destructor de la clase.
Nombre:	AESGenKey()
Descripción:	Permite la generación de la llave para el cifrado y descifrado de datos utilizando el algoritmo AES. Esta operación la delega en la clase ManagerServer.
Nombre:	RSAPrivateKey()
Descripción:	Permite la generación de la llave privada para el cifrado y descifrado de datos utilizando el algoritmo RSA. Esta operación la delega en la clase ManagerServer.
Nombre:	RSAPublicKey(std::string sk)
Descripción:	Permite la generación de la llave pública para el cifrado y descifrado de datos utilizando el algoritmo RSA, a partir de la llave privada. Esta operación la delega en la clase ManagerServer.
Nombre:	sha256(std::string text)
Descripción:	Método auxiliar de la clase que devuelve una función resumen del texto utilizando el algoritmo SHA 256.

Tabla 19: Descripción de la clase KeyManagerServer

Nombre: KeyManagerServer	
Descripción: Es la clase donde residen las funcionalidades para el almacenamiento de las llaves de cifrado y descifrado de datos.	
Atributo	**Tipo**
sqliteManager	SQLiteManager *
Nombre:	KeyManagerServer()
Descripción:	Constructor por defecto de la clase.
Nombre:	~KeyManagerServer()
Descripción:	Destructor de la clase.
Nombre:	addKey(std::string id, std::string key, std::string tCreated)
Descripción:	Permite almacenar una llave con los datos pasados por parámetro. Esta operación la delega en la clase ManagerServer.
Nombre:	deleteKey(std::string id)
Descripción:	Permite eliminar una llave según el identificador pasado por parámetro. Esta operación la delega en la clase ManagerServer.
Nombre:	loadFirstKey()
Descripción:	Permite cargar la primera llave para la comunicación cifrada, esta llave se encuentra almacenada en un fichero de texto. Esta operación la delega en la clase ManagerServer.
Nombre:	loadKey(std::string id)
Descripción:	Permite cargar una llave según el identificador pasado por parámetro. Esta operación la delega en la clase ManagerServer.
Nombre:	loadTime(std::string id)
Descripción:	Permite cargar el momento de creación de una llave según el identificador pasado por parámetro. Esta operación la delega en la clase ManagerServer.

Tabla 20: Descripción de la clase SQLiteManager

Nombre: SQLiteManager	
Descripción: Es la clase donde residen las funcionalidades para el manejo de la base de datos con la herramienta SQLite.	
Atributo	**Tipo**
Tuple	Vector<Field>
Table	Vector<Tuple>
Db	sqlite3 *
Table	static Table
itTable	Table::iterator
iterating	bool
lastQuery	string
*errMsg	char
Nombre:	SQLiteManager()
Descripción:	Constructor por defecto de la clase, en él se crea la base de datos con las tablas y campos que la componen.
Nombre:	~SQLiteManager()
Descripción:	Destructor de la clase.
Nombre:	Exec(std::string query)
Descripción:	Permite ejecutar una consulta, pasada por parámetros, en la base de datos. Esta operación la delega en la clase KeyManagerServer.
Nombre:	getLastQuery()
Descripción:	Permite devolver la última consulta ejecutada en la base de datos. Esta operación la delega en la clase KeyManagerServer.
Nombre:	Next()
Descripción:	Permite moverse a través de los resultados obtenidos de una consulta a la base de datos. Esta operación la delega en la clase KeyManagerServer.
Nombre:	Clear()
Descripción:	Permite limpiar los vectores para la realización de nuevas consultas a la base de datos. Esta operación la delega en la clase KeyManagerServer.
Nombre:	Value(unsigned int pos)
Descripción:	Permite obtener los resultados de una consulta en una posición específica moviéndose a través del método Next(). Esta operación la delega en la clase KeyManagerServer.

Tabla 21: Descripción de la clase ManagerServer

Nombre: ManagerServer.	
Tipo de Clase: Controladora	
Atributo	**Tipo**
keysManager	KeyManagerServer *
encodeDecode	EncodeDecodeMotorServer *
keyGenerator	KeysGenerator *
Nombre:	ManagerServer()
Descripción:	Constructor por defecto de la clase.
Nombre:	~ManagerServer()
Descripción:	Destructor de la clase.
Nombre:	generateFirstKey()
Descripción:	Permite generar la llave para la primera comunicación entre los clientes y el servidor y almacenarla en un fichero.
Nombre:	checkKeyValidity(std::string tCreation)
Descripción:	Permite chequear la validez de la llave según un período de tiempo definido y el momento de la creación de esta pasado por parámetros.
Nombre:	newKey(std::string id, std::string key)
Descripción:	Permite la creación de una nueva llave con el identificador y el valor de la llave pasado por parámetro.
Nombre:	generateClientKey(std::string client)
Descripción:	Permite generar el par de llaves, para la comunicación Cliente-Servidor, correspondiente a un cliente pasado por parámetros y almacenarla.
Nombre:	managerMessage(std::string message)
Descripción:	Permite manipular el mensaje recibido por parámetros.
Nombre:	managerCommunication(std::string message, std::string client)
Descripción:	Permite obtener la llave del canal solicitada para la comunicación y cifrarla.
Nombre:	firstClientCommunication(std::string client)
Descripción:	Permite obtener la llave para la primera comunicación Cliente-Servidor y cifrarla con la llave temporal.

Anexo 2. Descripción de las clases del subsistema EncryptionClient.

Tabla 22: Descripción de la clase EncodeDecodeMotorClient

Nombre: EncodeDecodeMotorClient	
Descripción: Es la clase donde residen las funcionalidades de cifrado y descifrado de datos.	
Nombre:	EncodeDecodeMotorClient()
Descripción:	Constructor por defecto de la clase.
Nombre:	~EncodeDecodeMotorClient()
Descripción:	Destructor de la clase.
Nombre:	encodeSimetric(std::string text, std::string key)
Descripción:	Permite el cifrado simétrico de los datos utilizando el algoritmo AES. Esta operación la delega en la clase ManagerClient.
Nombre:	encodeAsimetric(std::string text, std::string publicKey)
Descripción:	Permite el cifrado asimétrico de los datos utilizando el algoritmo RSA. Esta operación la delega en la clase ManagerClient.
Nombre:	decodeSimetric(std::string text, std::string key)
Descripción:	Permite el descifrado simétrico de los datos utilizando el algoritmo AES. Esta operación la delega en la clase ManagerClient.
Nombre:	decodeAsimetric(std::string text, std::string privateKey)
Descripción:	Permite el descifrado asimétrico de los datos utilizando el algoritmo RSA. Esta operación la delega en la clase ManagerClient.
Nombre:	auxSha256(std::string text)
Descripción:	Método auxiliar de la clase que devuelve una función resumen del texto utilizando el algoritmo SHA 256.

Tabla 23: Descripción de la clase KeyManagerClient

Nombre: KeyManagerClient	
Descripción: Es la clase donde residen las funcionalidades para el almacenamiento de las llaves de cifrado y descifrado de datos para la comunicación entre los módulos del SCADA.	
Atributo	**Tipo**
Key	map< std::string, std::string >
TCreated	map< std::string, std::string >
key	Key
tCreated	TCreated
Nombre:	KeyManagerClient()
Descripción:	Constructor por defecto de la clase.
Nombre:	~KeyManagerClient()
Descripción:	Destructor de la clase.
Nombre:	addKey(std::string id, std::string key, std::string tCreated)
Descripción:	Permite almacenar una llave con los datos pasados por parámetro. Esta operación la delega en la clase ManagerClient.
Nombre:	deleteKey(std::string id)
Descripción:	Permite eliminar una llave según el identificador pasado por parámetro. Esta operación la delega en la clase ManagerClient.
Nombre:	loadFirstKey()
Descripción:	Permite cargar la primera llave para la comunicación cifrada, esta llave se encuentra almacenada en un fichero de texto. Esta operación la delega en la clase ManagerClient.
Nombre:	loadKey(std::string id)
Descripción:	Permite cargar una llave según el identificador pasado por parámetro. Esta operación la delega en la clase ManagerClient.
Nombre:	loadTime(std::string id)
Descripción:	Permite cargar el momento de creación de una llave según el identificador pasado por parámetro. Esta operación la delega en la clase ManagerClient.

Tabla 24: Descripción de la clase ManagerClient

Nombre: ManagerClient	
Tipo de Clase: Controladora	
Atributo	**Tipo**
keysManager	KeyManagerClient *
encodeDecode	EncodeDecodeMotorClient *
Nombre:	ManagerClient()
Descripción:	Constructor por defecto de la clase.
Nombre:	~ManagerClient()
Descripción:	Destructor de la clase.
Nombre:	firstCommunicationServer()
Descripción:	Permite establecer la primera comunicación con el servidor de forma segura mediante una llave temporal cargada de un fichero.
Nombre:	receiveFirstKeyCommunicServer(std::string data)
Descripción:	Permite recibir de forma segura las llaves de comunicación con el servidor.
Nombre:	receiveDataServer(std::string data, std::string idCanal)
Descripción:	Permite gestionar las llaves de los canales de comunicación, brindadas por el servidor.
Nombre:	sendDataServer(std::string data)
Descripción:	Permite solicitar al servidor la llave para un canal de comunicación.
Nombre:	receiveDataCipher(std::string data, std::string idCanal)
Descripción:	Permite obtener en claro los datos adquiridos de un canal de comunicación.
Nombre:	sendDataCipher(std::string data, std::string idCanal)
Descripción:	Permite publicar de forma segura datos en un canal de comunicación.
Nombre:	checkKeyValidity(std::string tCreation)
Descripción:	Permite chequear si una llave es válida según el tiempo establecido para el vencimiento.

Glosario de Términos

A

- **ALBA:** La Alternativa Bolivariana para los Pueblos de Nuestra América o ALBA es una propuesta de integración enfocada para los países de América Latina y el Caribe que pone énfasis en la lucha contra la pobreza y la exclusión social con base en doctrinas de izquierda.

- **Aplicación:** Terminología técnica para referirse a un programa de software.

- **Arquitectura:** Indica la estructura, funcionamiento e interacción entre las partes del software.

C

- **CEDIN:** Siglas que identifican el Centro de Informática Industrial de la facultad 5 de la Universidad de las Ciencias Informáticas.

- **Cifrado:** Método que permite aumentar la seguridad de un mensaje o de un archivo mediante la codificación del contenido, de manera que sólo pueda leerlo la persona que cuente con la clave de cifrado adecuada para descodificarlo.

- **Comunicación:** Proceso mediante el cual se puede transmitir información de una entidad a otra.

- **Confiabilidad:** Un sistema es confiable en la medida que posea un alto nivel de seguridad.

- **Control de acceso:** Mecanismo que en función de la identificación ya autenticada permite acceder a datos o recursos.

F

- **Fiabilidad:** Referido al comportamiento de un sistema o dispositivo, se define como la "probabilidad de que el dispositivo desarrolle una determinada función, bajo ciertas condiciones y durante un período de tiempo determinado".

H

- **HMI:** Interfaz Hombre-Máquina (Human Machine Interface).

I

- **IDE:** Siglas en inglés que identifican un Entorno de Desarrollo Integrado (Integrated Development Environment). Es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica.

- **Inteligible:** Que puede ser entendido. Comprensible, claro, accesible, asequible, descifrable, legible, penetrable, fácil, sencillo.

- **Ininteligible:** No inteligible. Incomprensible, indescifrable, enrevesado, difícil, incognoscible.

R

- **Recurso:** Cualquier parte componente de un sistema de información.

M

- **Middleware:** Software de conectividad que ofrece un conjunto de servicios que hacen posible el funcionamiento de aplicaciones distribuidas sobre plataformas heterogéneas. Funciona como una capa de abstracción de software distribuida, que se sitúa entre las capas de aplicaciones y las capas inferiores (sistema operativo y red).

- **Multiplataforma:** Término utilizado frecuentemente en informática para indicar la capacidad o características de poder funcionar o mantener una interoperabilidad de forma similar en diferentes sistemas operativos o plataformas.

S

- **SCADA:** Supervisión Control y Adquisición de Datos (SCADA): Aplicación de software especialmente diseñada para funcionar sobre computadores en el control de producción, proporcionando comunicación con los dispositivos de campo y controlando el proceso de forma automática desde la pantalla del operador, brindando toda la información asociada al proceso.

- **Seguridad:** Proviene de la palabra securitas del latín. Cotidianamente se puede referir a la seguridad como la ausencia de riesgo o también a la confianza en algo o alguien.

- **Sistema:** Compuesto cuyos componentes se relacionan con al menos algún otro componente, puede ser material o conceptual.

- **Software:** Conjunto de programas, documentos, procesamientos y rutinas asociadas con la operación de un sistema de computadoras, es decir, la parte intangible o lógica de una computadora.

- **Software Libre:** Se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software.