Universidad de La Laguna

Trabajo de Fin de Máster

Máster Universitario en Ciberseguridad e Inteligencia de Datos

---

Infraestructura de clave pública en IoT

Public Key Infrastructure in IoT

Iris Estefanía Pereira Domínguez

La Laguna, 13 de mayo de 2024

Dña. Mª Candelaria Hernández Goya, con N.I.F 45.441.714-Q, Profesora Titular de Universidad adscrita al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutora

C E R T I F I C A (N):

Que la presente memoria titulada:

"Infraestructura de clave pública en IoT"

ha sido realizada bajo su dirección por Dña. **Iris Estefanía Pereira Domínguez**, con N.I.F. 43.388.253-H.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos, firman la presente en La Laguna a 13 de mayo de 2024

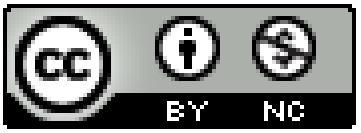
Agradecimientos

A mis padres, mi hermana y mis amigos y amigas, que han estado ahí, pese a todo, apoyándome incondicionalmente.

A María Candelaria Hernández Goya y Jose Ignacio Estévez Damas por su tiempo, paciencia y dedicación a lo largo del proyecto. Ha sido un honor tener la oportunidad de aprender de ustedes y trabajar bajo su guía. Gracias por su constante apoyo y por hacer de este viaje una experiencia enriquecedora.

Gracias por creer en mí en cada paso del camino.

Licencia



Resumen

El propósito de este trabajo ha sido abordar la seguridad en las comunicaciones del estándar OPC UA (Open Platform Communications - Unified Architecture) en entornos industriales. OPC UA es un estándar de comunicación de gran relevancia en la industria diseñado para facilitar la interoperabilidad entre sistemas heterogéneos. Su función principal reside en hacer posible el intercambio de datos de forma eficiente y segura entre dispositivos utilizados en el ámbito industrial. Este estándar desempeña un papel fundamental en la automatización de procesos y la evolución en la industria.

Así pues, este proyecto está enfocado principalmente en el empleo de Infraestructura de Clave Pública (PKI) para securizar las comunicaciones dentro del contexto del IIoT (Internet Industrial de las Cosas). De este modo, se proporciona un método robusto y seguro para verificar la identidad de los dispositivos y sistemas, así como para cifrar las comunicaciones, lo que impide que estos datos puedan ser interceptados e incluso modificados.

La PKI proporciona las tecnologías y procedimientos necesarios para garantizar la seguridad en la gestión de certificados, desplegando mecanismos que engloban desde la generación inicial y distribución hasta la revocación de los mismos. Esta capacidad de las PKI de revocar certificados es uno de los aspectos críticos, ya que permite mitigar potenciales riesgos en los casos en los que se viera comprometida la seguridad o la confianza en la autoridad emisora, invalidando certificados comprometidos o expirados, asegurando así la integridad y la confiabilidad de la infraestructura de seguridad.

El estándar OPC UA y, por ende, las aplicaciones que lo implementan, por lo general hacen uso de certificados digitales como mecanismo para proporcionar autenticación, firma y confidencialidad, lo cual garantiza la identidad de los sistemas y dispositivos involucrados en la comunicación, así como la confidencialidad de los datos. Sin embargo, la mayoría de estas aplicaciones recurren a certificados autofirmados que, debido a que no están respaldados por una autoridad de confianza reconocida, pueden suponer un riesgo en entornos poco controlados, dado que son susceptibles de ser interceptados por un atacante, lo cual le permitiría suplantar un servidor legítimo y obtener información sensible o confidencial.

Para cumplir con el objetivo de proteger las comunicaciones y mitigar posibles ataques de intermediarios, se ha generado e implantado en el laboratorio Beckhoff, situado en la Escuela Técnica Superior de Ingeniería Informática de la Universidad de La Laguna, una autoridad certificadora dedicada, así como los procedimientos y componentes necesarios para establecer una PKI robusta y plenamente funcional para los PLCs Beckhoff CX5010. Al hacer esto, se ha intentado, en el caso de intercambiar certificados autofirmados por cada aplicación y/o dispositivo, reemplazarlos por otros creados sobre la base de dicha autoridad de confianza, autenticando los extremos de la comunicación y configurando además otros parámetros como método de autenticación, longitud de clave, algoritmo de firma y demás políticas de seguridad.

Palabras clave: IOT, IIOT, Infraestructura de Clave Pública, PLC, OPC UA, Certificado.

Abstract

The purpose of this work has been to address the security of OPC UA (Open Platform Communications - Unified Architecture) communications in industrial environments. OPC UA is a communication standard of great relevance in the industry designed to facilitate interoperability between heterogeneous systems. Its main function is to enable efficient and secure data exchange between devices used in industry. This standard plays a fundamental role in process automation and evolution in industry.

This project is primarily focused on the use of Public Key Infrastructure (PKI) to secure communications within the context of the IIoT (Industrial Internet of Things). This provides a robust and secure method for verifying the identity of devices and systems, as well as encrypting communications, which prevents this data from being intercepted and even modified.

PKI provides the technologies and procedures necessary to guarantee security in certificate management, deploying mechanisms that cover everything from initial generation and distribution to certificate revocation. This ability of PKIs to revoke certificates is one of the critical aspects, since it allows mitigating potential risks in cases where security or trust in the issuing authority is compromised, invalidating compromised or expired certificates, thus ensuring the integrity and reliability of the security infrastructure.

The OPC UA standard and, therefore, the applications that implement it, generally make use of digital certificates as a mechanism to provide authentication, signature and confidentiality, which guarantees the identity of the systems and devices involved in the communication, as well as the confidentiality of the data. However, most of these applications rely on self-signed certificates which, because they are not backed by a recognized trusted authority, can pose a risk in poorly controlled environments, since they are susceptible to interception by an attacker, which would allow him to impersonate a legitimate server and obtain sensitive or confidential information.

To meet the objective of protecting communications and mitigating possible attacks by intermediaries, a dedicated certificate authority has been generated and implemented at the Beckhoff laboratory located at the School of Computer Engineering of the University of La Laguna, as well as the procedures and components necessary to establish a robust and fully functional PKI for the Beckhoff CX5010 PLCs. In doing so, we have tried, in the case of exchanging self-signed certificates for each application and/or device, to replace them with others created on the basis of such trusted authority, authenticating the communication ends, and also configuring other parameters such as authentication method, key length, signature algorithm and other security policies.

Keywords: IOT, IIOT, Public Key Infrastructure, PLC, OPC UA, Certificate.

Índice general

Índice de figuras

Índice de tablas

1. Introducción

El propósito de este documento es recopilar toda la información relevante para la realización de este Trabajo de Fin de Máster. Concretamente, el primer capítulo consta de una introducción al mismo, plasmando los objetivos planteados para su implementación y las etapas que se han seguido hasta su conclusión. Finalmente, se presenta en detalle la estructura de este documento y sus apéndices, así como una breve descripción del contenido de cada apartado.

1.1. Motivación

En la actualidad, el Internet Industrial de las Cosas (Industrial Internet of Things, IIoT) está continuamente transformando la manera en que operan las industrias, ya que facilita la interconexión de dispositivos industriales a través de Internet. Esta conectividad permite la recopilación e intercambio de datos en tiempo real entre equipos, sistemas de control y aplicaciones más fácilmente, que da como resultado mayor eficiencia operativa, optimización de los procesos industriales y mejora en la capacidad de tomar decisiones. Sin embargo, junto con los beneficios, han aparecido importantes desafíos en materia de seguridad, particularmente debido a amenazas como ataques de malware, intrusiones maliciosas y manipulación de datos.

En este contexto, los Controladores Lógicos Programables (Programmable Logic Controller, PLC) tienen funciones cruciales, dado que controlan y monitorizan múltiples procesos en el entorno de producción. Estos dispositivos son responsables de recibir datos de entrada, ejecutar la lógica de control y emitir datos para controlar equipos industriales, sistemas de fabricación y otros equipos. Dada su relevancia, la seguridad de los PLCs es esencial para garantizar la integridad de los procesos industriales y protegerlos contra potenciales amenazas. En el artículo [1] se realiza un análisis detallado de vulnerabilidades y modelos de ataque relacionados con estos dispositivos.

Varios ataques han puesto de manifiesto recientemente la debilidad de los PLC. Un buen ejemplo ocurrió cuando ciberdelincuentes apoyados por Irán atacaron a la Autoridad del Agua de EE.UU. con esta tecnología [2]. En el ataque, los PLC de Unitronics se utilizaron activamente para atacar la Autoridad Municipal del Agua de Aliquippa, en Pensilvania, lo que provocó la intervención de la Agencia de Ciberseguridad e Infraestructuras (CISA) de Estados Unidos. Además, el grupo Cyber Av3ngers ha afirmado haber infiltrado estaciones de tratamiento de agua en Israel, lo que indica un patrón preocupante de ataques dirigidos a infraestructuras críticas.

Estos acontecimientos ponen de manifiesto la necesidad de proteger los PLC de las ciberamenazas. Otro ejemplo de esto son los PLC maliciosos. Los *Evil PLC* [3] son un tipo de dispositivos que podría atacar un sistema PLC a través de sus puntos débiles, lo que provocaría una pérdida de protección en los procesos industriales. Es importante aplicar medidas de seguridad estrictas, como cambiar las contraseñas predeterminadas, utilizar

autenticación multifactor y aislar los PLC de Internet en lugares donde la seguridad del agua y otros servicios críticos esté amenazada, para ayudar a reducir los riesgos y garantizar que las operaciones industriales cruciales no se paralicen.

Shodan es un motor de búsqueda que mantiene un registro de todos los dispositivos conectados a Internet y genera un informe detallado sobre cada uno de ellos. Realizando una búsqueda en Shodan con el término *"authentication disabled" "RFB 003.008" country:es port:"5900"*, revela en la figura 1 una serie de dispositivos con la autenticación desactivada y utilizando el protocolo de escritorio remoto VNC en España. Este hallazgo subraya la preocupante falta de seguridad en la configuración de estos dispositivos, lo que los hace vulnerables a posibles intrusiones y ataques cibernéticos. Estos equipos, una herramienta crucial utilizada para el control de procesos industriales, son susceptibles de sufrir brechas de seguridad como consecuencia de la falta de medidas de seguridad. Esta despreocupación por la seguridad de los PLC pone de manifiesto la necesidad inmediata de hacer comprender la importancia de aplicar medidas de seguridad sólidas para proteger la integridad de los sistemas industriales críticos.

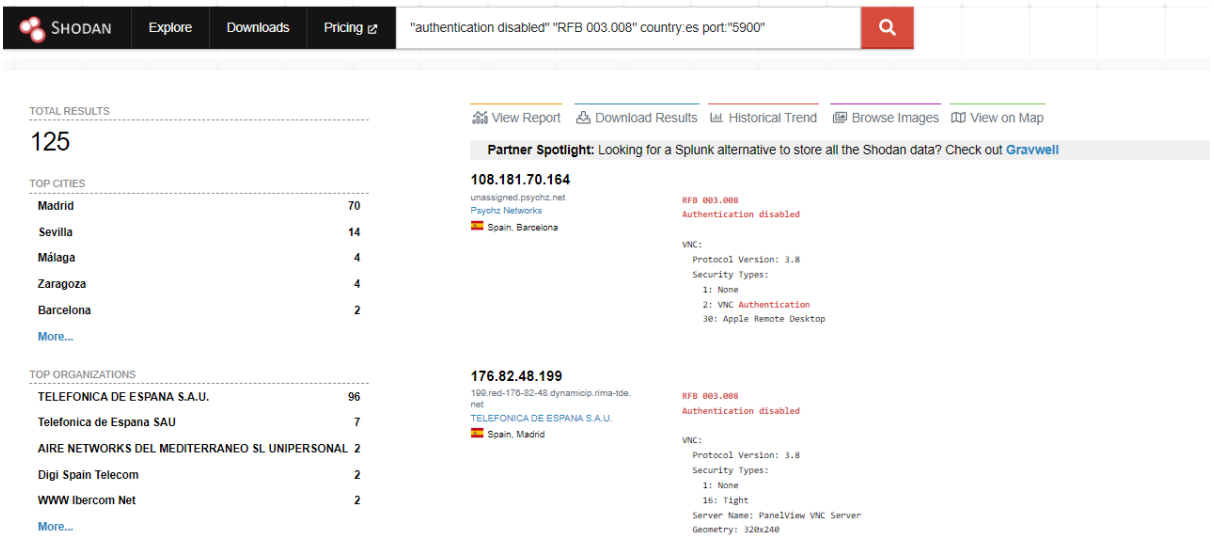


Figura 1: Búsqueda de dispositivos sin autenticación en Shodan.

En el ámbito industrial, OPC UA (Open Platform Communications - Unified Architecture) destaca como el estándar de comunicación más ampliamente aceptado, debido a que promueve la interoperabilidad entre sistemas heterogéneos, es decir, sistemas que utilizan diferentes tecnologías, protocolos o plataformas [4]. OPC UA crea un marco robusto y seguro para el intercambio de datos entre equipos industriales y software, garantizando la conexión y la integración entre ellos. No obstante, aunque OPC UA ya integra medidas de seguridad, la implementación adicional de una infraestructura de clave pública puede consolidar aún más la seguridad y fiabilidad de las comunicaciones en entornos IIoT.

La infraestructura de clave pública (Public Key Infrastructure, PKI) es un conjunto de tecnologías y protocolos utilizados para gestionar de forma segura la creación, distribución y autenticación de certificados digitales. Estos certificados se utilizan para autenticar

dispositivos, sistemas y usuarios en una red, y para cifrar y firmar digitalmente los datos transmitidos. Además, los certificados digitales contribuyen al principio de autenticación, uno de los pilares fundamentales de la ciberseguridad, al garantizar la identidad legítima de los dispositivos y usuarios en el entorno IIoT. La incorporación de PKI en un entorno industrial permite crear una base de seguridad sólida para anticipar amenazas como el acceso no autorizado, la manipulación de datos o la interceptación de comunicaciones, lo cual es especialmente importante de cara a garantizar la continuidad y la seguridad de operaciones críticas.

Así pues, mediante la combinación de estas tecnologías se puede lograr una solución completa que permita hacer frente a los retos actuales de la ciberseguridad, como son la integridad, la fiabilidad y la protección de los sistemas y datos en el contexto de entornos industriales en continua evolución.

1.2. Definición del problema

En la actualidad, encontrar aplicaciones de ámbito industrial que emplean el estándar OPC UA es cotidiano. Sin embargo, el principal inconveniente radica en que la mayoría de ellos emplean certificados autofirmados para asegurar las comunicaciones entre cliente y servidor. Este tipo de certificados, al no estar firmados digitalmente por una autoridad certificadora de confianza que valide su autenticidad, están sujetos a vulnerabilidades. Entre ellas podemos citar:

- Ausencia de verificación de la identidad: Los certificados autofirmados no proporcionan garantía de autenticidad, por lo que esto posibilita que un atacante genere un certificado autofirmado y suplante una entidad legítima.
- Ataques de intermediarios maliciosos: En un ataque de tipo *person-in-the-middle*, un atacante podría interceptar la comunicación y presentar un certificado autofirmado como si fuese legítimo, lo cual le permitiría leer, modificar o manipular la información transmitida sin levantar sospechas.
- Generación insegura: Los certificados autofirmados generados automáticamente por aplicaciones podrían, en algunos casos, ser generados con claves débiles o configuraciones incorrectas, lo que los haría susceptibles a ataques.

Por ello, en entornos críticos no es adecuado emplear los certificados generados por defecto. Es recomendable utilizar certificados generados por autoridades de confianza como Let’s Encrypt, DigiCert o GlobalSign, con una configuración apropiada para garantizar una comunicación segura.

1.3. Estado del arte

El grupo de estandarización formado por Fisher-Rosemount, Intellution, Opto 22 y Rockwell Software, entre otros, se unió para crear un estándar para el intercambio de datos. Este estándar de comunicación, conocido como OPC (OLE for Process Control), fue inicialmente definido en 1995. En 10 años, OPC ha crecido hasta convertirse en el estándar

de comunicación industrial gracias a su adaptabilidad y conformidad con casi todos los equipos y sistemas de automatización del mundo.

A lo largo de los años, han surgido varias versiones de OPC, como la especificación DA para datos en tiempo real, la especificación AE para alarmas y la HDA para datos históricos extensos, entre otras. Cada una de estas versiones lee datos o los escribe a través de su propio conjunto de comandos. Todas estas versiones se basan en el marco de Microsoft Windows y utilizan COM/DCOM (Distributed Component Object Model) como modelo de comunicación para el intercambio de datos entre componentes de software.

El protocolo OPC Data Access (DA) es uno de los más conocidos del estándar OPC, ya que fue pionero en devolver información adicional por cada solicitud de lectura o escritura. Además del nombre y el valor de una etiqueta, OPC DA también les asigna marcas de tiempo y valores de calidad. Las marcas de tiempo informan de cuándo se ha producido un cambio en un valor concreto, mientras que la calidad informa de cualquier contratiempo en la comunicación o de si los datos introducidos son erróneos.

Cada especificación OPC (OPC DA, también conocida como OPC Classic y otras) tiene su propio grupo objetivo. Sin embargo, estos enfoques con el tiempo se han adaptado a los contextos modernos. Para responder a ellos, la Fundación OPC ideó el nuevo estándar OPC Unified Architecture (UA).

OPC UA ofrece varias mejoras y diferencias funcionales importantes, incluidas mejoras en la seguridad y la capacidad de agrupar etiquetas, y optimización de modelos de comunicación, lo que facilita la configuración para el usuario. Además, han surgido nuevas tecnologías que incluyen, entre otras, blockchain, que permite compartir información de forma precisa y segura. A su vez, las plataformas de seguimiento y gestión han accedido a más información de procesos respecto a años anteriores gracias a este desarrollo [5].

Desde las primeras versiones del protocolo hasta la actualidad, OPC ha empleado una arquitectura cliente-servidor, en la que ambos elementos interactúan para realizar intercambios de información. Esto implica que cada servidor espera a que uno o varios clientes le soliciten información. Una vez el servidor recibe una petición, la contesta y se queda esperando la siguiente. En OPC es el cliente quien decide cuándo y cómo va a interactuar con el servidor [6]:

- El cliente OPC está compuesto por dos partes, siendo la primera el *Client Application*, donde se configura y se escribe el código del cliente OPC, y la segunda el *Communication Stack*, donde se definen las solicitudes de datos. Además, hay dos formas de realizar solicitudes: solicitud directa, donde el cliente solicita y el servidor responde, y suscripción, donde el cliente se suscribe y el servidor envía datos si hay un cambio, se supera un umbral o se cumple una condición especificada por el cliente y aprobada por el servidor.
- El servidor OPC se divide en tres partes: *Server Application, Server Api* y *Communication Stack*. Hay diferentes secciones dentro de la arquitectura de un

servidor OPC: la primera incluye el código implementado para la función del servidor y, además, hay diferentes objetos reales, es decir, variables internas del servidor o dispositivos utilizados por el servidor.

La base de la comunicación entre clientes y servidores OPC son los nodos. Son elementos que representan datos o funcionalidades. Cada nodo tiene una identidad única y puede contener información como valores de datos, objetos y variables. Los clientes OPC utilizan estos nodos para interactuar con el servidor, como leer datos o invocar métodos.

La configuración entre el cliente y el servidor puede realizarse de múltiples formas, lo que permite personalizar en gran medida las comunicaciones. Dicha configuración permite que pueda ser leyendo y escribiendo datos de forma directa a través del cliente en los nodos del servidor, monitorizando posibles cambios en los valores de los nodos del servidor o utilizando suscripciones para recibir actualizaciones automáticas cuando estos valores se modifican.

1.4. Objetivos

El objetivo principal de este proyecto consiste en garantizar una comunicación segura entre clientes y servidores OPC UA mediante el uso de certificados digitales previamente firmados por una autoridad certificadora de confianza.

Los objetivos de este trabajo se desglosan en detalle a continuación:

- **Familiarización con el estándar OPC UA:** Adquirir conocimientos fundamentales sobre el estándar OPC UA, explorando su estructura, funcionalidades y aplicaciones prácticas.

- **Análisis de primitivas de seguridad del estándar:** Investigar y analizar las primitivas de seguridad proporcionadas por OPC UA, incluyendo autenticación, cifrado y firma digital, para comprender cómo garantizan la integridad y confidencialidad de la comunicación.

- **Familiarización con las herramientas de simulación:** Indagar en las herramientas de simulación disponibles para OPC UA, como UAExpert y Prosys OPC UA Simulation Server, explorando sus características y capacidades.

- **Análisis y despliegue de la PKI en el entorno de simulación:** Estudiar y desplegar una Infraestructura de Clave Pública (PKI) en el entorno de simulación, comprendiendo cómo se generan, gestionan y utilizan los certificados digitales para garantizar la autenticidad y seguridad en la comunicación OPC UA.

- **Análisis y despliegue de la PKI en el entorno de PLC de Beckhoff:** Investigar y aplicar los principios de la PKI en el entorno de los controladores lógicos programables (PLC) de Beckhoff, comprendiendo cómo se integran los certificados digitales en estos dispositivos para asegurar la comunicación segura y fiable con sistemas compatibles con OPC UA.

1.5. Conceptualización de la propuesta

El planteamiento llevado a cabo está pensado para su implantación en una planta industrial o un entorno de esas características, donde se trabaja con servidores y clientes OPC UA. Estos son susceptibles de ser atacados debido a que llevan a cabo operaciones críticas y manejan datos de gran relevancia, por lo que es fundamental asegurarlos.

Así pues, se pretende obtener un entorno en el que las comunicaciones sean plenamente seguras. Para ello es fundamental tener en cuenta los perfiles de seguridad configurables en las aplicaciones OPC UA, además de administrar y gestionar correctamente los certificados digitales que estas emplean de cara a garantizar la autenticación de los dispositivos que intervienen en el entorno industrial.

1.6. Fases

En este apartado se ofrece una visión general de las fases llevadas a cabo a lo largo de la evolución del proyecto.

- **Marco teórico:**
  - Explicación detallada del estándar OPC UA.
  - Conceptos fundamentales de la seguridad en OPC UA.
  - Fundamentos de la Infraestructura de Clave Pública (PKI).

- **Diseño de una PKI para OPC UA:**
  - Definición de los requisitos de seguridad para el despliegue de OPC UA.
  - Diseño de la infraestructura de la PKI, incluyendo autoridades de certificación, políticas de certificación, y mecanismos de gestión de claves.
  - Integración de la PKI en el contexto de OPC UA.

- **Simulación del uso de certificados:**
  - Configuración del entorno de simulación utilizando Prosys OPC Simulation Server, UA Expert u otros simuladores compatibles.
  - Generación de certificados a través de la PKI diseñada e implementada.
  - Configuración de los servidores y clientes en el simulador para utilizar los certificados generados.
  - Realización de escenarios de comunicación entre los servidores y clientes simulados, utilizando la autenticación y seguridad proporcionadas por los certificados.

- **Implementación:**
  - Desarrollo de la PKI según el diseño propuesto.
  - Configuración de servidores OPC UA y clientes para utilizar la PKI.
  - Pruebas y validación del funcionamiento de la PKI en un entorno de laboratorio.

- **Resultados y Análisis**
  - Presentación de los resultados obtenidos durante la implementación y

simulación.
- Análisis de la eficacia y seguridad de la PKI desplegada en OPC UA.

1.7. Estructura del documento

Este documento se divide en dos partes: la primera de ellas contiene la memoria que resume el proyecto, mientras que la segunda incluye los anexos que especifican en detalle algunos aspectos del mismo.

La memoria consta de los capítulos que se presentan a continuación:

Capítulo 1 – Introducción. En el primer capítulo se presenta brevemente la motivación, definición del problema, estado del arte y objetivos generales del proyecto, así como sus distintas fases. Como último apartado, se presenta la estructura de la memoria.

Capítulo 2 – Tecnologías empleadas. En este capítulo se comentan las tecnologías empleadas tanto hardware como software.

Capítulo 3 – Especificaciones de seguridad. Trata de describir concretamente las diferentes medidas y protocolos implementados en OPC UA para garantizar la seguridad en las comunicaciones, así como las prácticas recomendadas para su implementación efectiva en aplicaciones que hacen uso del estándar.

Capítulo 4 – Implementación. Este capítulo está dedicado a detallar el desarrollo del proyecto y las diferentes fases que se han atravesado a lo largo del mismo, así como las dificultades encontradas.

Capítulo 5– Presupuesto. En este apartado se detalla el presupuesto para este proyecto en cuanto a costes de material y de recursos humanos.

Capítulo 6 – Conclusiones y líneas futuras. En esta sección se finaliza la descripción del proyecto con una serie de conclusiones, así como un conjunto de objetivos a futuro que incluir y/o mejorar en el proyecto.

Capítulo 8 – Bibliografía. Finalmente, se presenta una lista detallada de las fuentes bibliográficas utilizadas durante la investigación y el desarrollo del proyecto. Incluye aquellos recursos consultados que respaldan la fundamentación del trabajo realizado.

Este documento contiene, además, una serie de anexos que completan y amplían la información ya presentada en los apartados anteriores.

Anexo 1 – Enlace al repositorio de código. Este anexo presenta el enlace al repositorio donde se aloja el código del script para crear una CA raíz autofirmada y los certificados OPC UA para las aplicaciones de simulación.

2. Tecnologías empleadas

En este capítulo se efectúa un análisis de las tecnologías utilizadas en el desarrollo de este proyecto, comenzando con un examen detallado de los equipos y otras tecnologías empleadas.

2.1. IoT e IIoT

El término IoT, o Internet de las Cosas, se refiere a la red colectiva de dispositivos conectados y a la tecnología que facilita la comunicación entre los dispositivos y la nube, así como entre los propios dispositivos [7]. Esto les otorga la capacidad de recopilar y compartir datos y de ejecutar ciertas operaciones de forma autónoma o bajo supervisión humana. Estos dispositivos pueden incluir desde electrodomésticos, sistemas wearables, hasta complejos sistemas industriales, equipados con sensores, actuadores y tecnologías de conectividad, todos ellos capaces de comunicarse entre sí y con otros sistemas.

IoT está basado en la capacidad de estos dispositivos para obtener datos de su entorno con la ayuda de sensores integrados, procesar la información recibida y tomar decisiones en función de los resultados obtenidos. Por otro lado, pueden comunicarse con servidores remotos, aplicaciones móviles u otros dispositivos de IoT para transferir datos, recibir instrucciones o realizar determinadas acciones.

El Internet de las Cosas tiene una amplia gama de aplicaciones en manufactura, salud, agricultura, transporte, domótica, energía, logística, etc. y, debido a su rápido crecimiento, este plantea importantes desafíos en el ámbito de seguridad, privacidad, interoperabilidad y gestión de datos.

Por otra parte, el Internet Industrial de las Cosas (IIoT) representa una extensión del concepto de Internet de las Cosas, particularmente enfocado en entornos industriales y procesos de fabricación. Su enfoque principal se centra en conectar equipos industriales, instalaciones de producción y sistemas de control utilizando Internet y otras redes de comunicación con el objetivo de mejorar la eficiencia, la productividad y la seguridad en ese ámbito específico.

Esta evolución del IoT utiliza técnicas y tecnologías avanzadas como sensores inteligentes, sistemas integrados, análisis de datos en tiempo real y conexiones de red de alta velocidad para crear un entorno industrial automatizado y conectado. Esta integración facilita la recopilación, el análisis y la aplicación inmediata de datos para optimizar la producción, prevenir fallos en los equipos, mejorar la calidad del producto y reducir los costes operativos.

Además, IIoT facilita la implantación de otros conceptos como la fabricación inteligente (Industria 4.0) y el mantenimiento predictivo, que tienen como objetivo optimizar la producción y reducir el tiempo de inactividad de los equipos a través del uso de datos y análisis avanzados. Por ejemplo, los sensores conectados a equipos industriales pueden

recopilar datos sobre el rendimiento y el estado del equipo, lo que permite identificar de forma proactiva potenciales problemas y la programación del mantenimiento para reducir tiempos de inactividad.

A pesar de los numerosos beneficios, la implementación industrial de IoT plantea desafíos importantes en términos de ciberseguridad, interoperabilidad de sistemas, gestión de datos y privacidad, los cuales requieren especial atención y soluciones robustas para garantizar su éxito a largo plazo.

2.2. PLC

Un controlador lógico programable (Programmable Logic Controller, PLC), es un dispositivo electrónico fundamental en entornos industriales, utilizado para monitorear y coordinar operaciones en sistemas automatizados. Actúa como un "cerebro", es decir, como el elemento central programable que controla y coordina diversos procesos del sistema, que van desde la automatización de procesos de fabricación, el seguimiento y control de equipos industriales, hasta la gestión de sistemas de seguridad. Los PLC se utilizan ampliamente en la industria para automatizar procesos de fabricación, mejorar la eficiencia operativa, controlar la calidad del producto y garantizar la seguridad operacional. Estos sistemas automatizados permiten que los procesos automáticos reemplacen las tareas manuales, lo que resulta en una producción más precisa, rápida y consistente. Para su correcto funcionamiento se deben programar previamente las funciones a realizar. Esto se realiza mediante un software específico que corresponde a la marca del PLC y al lenguaje de programación a utilizar. Un PLC consta principalmente de tres partes:

- Unidad Central de Procesamiento (CPU): Es el núcleo de un PLC, que ejecuta programas almacenados y procesa señales de entrada para determinar las acciones a tomar.
- Módulos de entrada: Son dispositivos que reciben señales del entorno externo, como interruptores, sensores, codificadores, entre otros, y las convierten en señales digitales que la CPU puede procesar.
- Módulos de Salida: Reciben señales de la CPU y las convierten en acciones físicas como motores de arranque, válvulas, luces, etc.

2.2.1. Beckhoff CX5010

Los dispositivos CX5010 son computadoras integradas diseñadas para un rendimiento confiable en entornos industriales. Estos dispositivos cuentan con procesadores Intel Atom® de bajo consumo que brindan el mejor equilibrio entre potencia de procesamiento y consumo de energía. El modelo CX5010 está equipado con un procesador Intel Atom® Z510 de 1,1 GHz.

Estas computadoras integradas son altamente versátiles y pueden adaptarse para cubrir una amplia gama de necesidades en automatización industrial. Dependiendo de la configuración del entorno de ejecución, la aplicación TwinCAT puede ser empleada en proyectos que involucren PLCs, control de movimiento, o una combinación de ambos, ya sea con o sin visualización. Desarrollado por Beckhoff Automation GmbH & Co. Kg, TwinCAT es un completo paquete de software que convierte prácticamente cualquier equipo compatible en un controlador PLC y CNC (Control Numérico por Computador), además de servir como dispositivo operativo [8].

Además de la potencia de procesamiento, el modelo CX5010 ofrece una amplia gama de opciones de conectividad y E/S, lo que los hace adecuados para una variedad de aplicaciones industriales, tales como automatización de maquinaria, control de procesos, monitorización y gestión de energía, etc. Su diseño robusto y compacto lo hace ideal para instalación en entornos hostiles donde el espacio es limitado.

Figura 2: Beckhoff CX5010

La tabla 1 muestra la ficha técnica de estos dispositivos.

PROCESADOR	Intel Atom® Z510, frecuencia de reloj de 1,1 GHz
NÚMERO DE NÚCLEOS	1
MEMORIA FLASH	Ranura para tarjeta Compact Flash, incluye 128 MB (ampliable)
MEMORIA PRINCIPAL	512 MB de RAM (no ampliable)
MEMORIA PERSISTENTE	UPS integrado de 1 segundo (1 MB en tarjeta Compact Flash)
INTERFACES	2 x RJ45 10/100/1000 Mbit/s, 1 x DVI-D, 4 x USB 2.0, 1 x interfaz opcional
LED DE DIAGNÓSTICO	1 x alimentación, 1 x estado TC, 1 x acceso flash, 2 x estado del bus
RELOJ	Reloj interno con batería respaldada para hora y fecha (batería intercambiable)

SISTEMA OPERATIVO	Windows Embedded CE 6, Windows Embedded Standard 2009
SOFTWARE DE CONTROL	TwinCAT 2 runtime, TwinCAT 3 runtime (XAR)
CONEXIÓN E/S	E-bus o K-bus, reconocimiento automático
SUMINISTRO DE ENERGÍA	24 V CC (-15%/+20%)
SUMINISTRO DE CORRIENTE E-BUS/K-BUS	2 A
CONSUMO MÁXIMO DE ENERGÍA	12 W
DIMENSIONES (AN X AL X PR)	106 mm x 100 mm x 92 mm
PESO APROX.	575 g
HUMEDAD RELATIVA	95%, sin condensación
TEMPERATURA DE OPERACIÓN/ALMACENAMIENTO	-25…+60°C/-40…+85°C
INMUNIDAD/EMISIÓN EMC	Cumple con EN 61000-6-2/EN 61000-6-4
RESISTENCIA A VIBRACIONES/CHOQUES	Cumple con EN 60068-2-6/EN 60068-2-27
CLASIFICACIÓN DE PROTECCIÓN	IP20
APROBACIONES/MARCAS	CE, UL, con opción de pedido CX1900-0105: ATEX
MARCADO EX	II 3 G Ex nA IIC T4 Gc
NIVEL DE PLATAFORMA TWINCAT 3	Performance (40)

Tabla 1: Ficha técnica del modelo CX5010

2.3. OPC UA

OPC UA (Open Platform Communications - Unified Architecture) es un estándar internacional (IEC62541) que permite comunicaciones de datos seguras, eficientes e interoperables entre dispositivos, sistemas y aplicaciones para la Industria 4.0. Es un estándar industrial, multiplataforma y de código abierto para entornos industriales desarrollado por la Fundación OPC en el año 2008.

OPC UA nace por la necesidad de obtener una herramienta independiente del sistema operativo subyacente manteniendo las características del OPC Clásico, pero evitando la dependencia de los sistemas Microsoft y su tecnología COM/DCOM. Por el contrario, emplea el protocolo TCP para comunicaciones en redes internas (intranet), y el protocolo HTTPS para realizar comunicaciones seguras a través de internet mediante el cifrado SSL/TLS. Este enfoque permite a OPC UA adaptarse a una variedad de entornos de red y necesidades de seguridad.

La arquitectura unificada OPC UA ha sido diseñada para adaptarse a las necesidades de la Industria 4.0 y ofrece varias ventajas significativas [9]:

- **Independencia de la plataforma:** OPC UA es independiente de la tecnología Windows, lo que permite su utilización en una variedad de sistemas operativos. Esto facilita la implementación de aplicaciones OPC UA nativas en dispositivos móviles y PLCs, lo que resulta fundamental en la era del IIoT y la Industria 4.0.

- **Mayor seguridad:** OPC UA incluye características de seguridad avanzadas como cifrado de mensajes de extremo a extremo utilizando AES-128 y AES-256, y la capacidad de implementar comunicación SSL y HTTPS lo cual garantiza la protección de datos durante la transferencia.

- **Arquitecturas simplificadas:** OPC UA integra las especificaciones de OPC Clásico (DA, HDA y A&E), eliminando la necesidad de componentes adicionales para la comunicación entre ellas. Además, OPC UA ha sido diseñado para coexistir con OPC Clásico, lo que permite aprovechar la infraestructura existente y acceder a una amplia gama de dispositivos y protocolos de comunicación.

- **Escalabilidad y ahorro de costes:** OPC UA permite a las organizaciones conectar puntos finales existentes y futuros de manera escalable aprovechando los recursos disponibles y simplificando las arquitecturas de automatización. Esto permite ahorrar significativamente los costes de mantenimiento y ampliación del sistema de automatización.

Sin embargo, el estándar OPC UA también presenta algunas limitaciones, como:

- **Complejidad:** Debido a que OPC UA tiene una amplia gama de características y capacidades, la implementación y configuración pueden resultar complejas, especialmente para usuarios no familiarizados con el estándar.

- **Sobrecarga:** En entornos con recursos limitados, la sobrecarga asociada a la implementación completa de OPC UA puede suponer un coste adicional de recursos, especialmente en dispositivos con capacidad de procesamiento limitada.
- **Coste:** Algunas implementaciones de OPC UA pueden requerir inversiones significativas en hardware, software y capacitación para conseguir un despliegue efectivo.

3. Especificaciones de seguridad

3.1. Seguridad del estándar OPC UA

La Oficina Federal de Seguridad de la Información (BSI) llevó a cabo en el año 2015, bajo la dirección del consorcio de TÜV SÜD Rail, el estudio de análisis de seguridad OPC UA [10]. El análisis de especificaciones ha demostrado que OPC UA, a diferencia de muchos otros protocolos industriales, ofrece un alto nivel de seguridad, ya que BSI no detectó errores sistemáticos.

OPC UA es una tecnología de comunicación industrial estándar utilizada en diversos niveles de una instalación industrial, desde la gestión empresarial hasta el control directo de procesos [11]. Su uso implica la interacción con clientes y proveedores, lo que lo convierte en un potencial objetivo de espionaje o sabotaje industrial. Además, puede estar expuesto a amenazas de malware no dirigido, como gusanos, que se propagan en redes públicas. La interrupción de las comunicaciones en el control de procesos podría incurrir en pérdidas financieras, afectar la seguridad pública y de los empleados, o causar daños ambientales. Por tanto, la seguridad en la implementación y operación del estándar es crucial para evitar estos riesgos.

OPC UA se implementa en una amplia gama de entornos operativos con diferentes suposiciones sobre amenazas y accesibilidad, y con una variedad de políticas de seguridad y regímenes de cumplimiento. Por lo tanto, OPC UA proporciona un conjunto flexible de mecanismos de seguridad que pueden variar en función de los protocolos y tecnologías de comunicación que se utilicen, así como el entorno en el que se apliquen. Estos mecanismos contribuyen a la protección en cada una de las dimensiones de la ciberseguridad.

3.1.1. Autenticación

En OPC UA, la autenticación [12] es fundamental tanto para las aplicaciones como para los usuarios.

Las aplicaciones se autentican entre sí durante el establecimiento de la comunicación mediante certificados digitales. La versión clásica de OPC utiliza la tecnología DCOM

(Distributed Component Object Model) de Microsoft para la comunicación entre componentes de software distribuidos en red. En contraste con OPC clásico, donde las restricciones de DCOM pueden complicar la configuración, en OPC UA la autenticación mediante certificados digitales simplifica enormemente su implementación.

Cuando un cliente se conecta a un servidor, se establece una conexión segura de inmediato. Esto implica que el cliente y el servidor deben intercambiar información segura para poder comunicarse correctamente. Para lograr esto, es necesario configurar tanto el cliente como el servidor para que confíen en el certificado digital del otro. Este proceso garantiza que la aplicación en ambos extremos de la comunicación sea autenticada, es decir, que se verifique su identidad y se asegure de que sean quienes dicen ser.

En cuanto a la autenticación del usuario, OPC UA requiere su cumplimiento y ofrece varios métodos, como la combinación de usuario y contraseña, certificados digitales y tickets Kerberos. Estos mecanismos son más flexibles y seguros que en OPC clásico, donde la autenticación no está habilitada por defecto y su configuración puede ser costosa, especialmente en entornos de dominio.

3.1.2. Autorización

El acceso para leer, escribir o ejecutar recursos debe autorizarse sólo para aquellas entidades que tengan la necesidad de ese acceso dentro de los requisitos del sistema. La autorización puede ser tan generalizada como permitir o no permitir que un cliente acceda a un servidor o podría ser mucho más detallada, como permitir acciones específicas sobre elementos de información específicos por parte de usuarios específicos. Por tanto, al usuario se le debe otorgar acceso solo a la información que necesita para la función que está realizando. La autorización puede proporcionarse a través de roles o ser gestionada por un GDS (Global Discovery Server). El GDS facilita a los usuarios la localización de servidores OPC UA en redes distribuidas. Esto permite a los clientes encontrar los servidores específicos que necesitan para acceder a determinados datos y funciones, así como conectarse a ellos.

3.1.3. Confidencialidad

OPC UA utiliza cifrado simétrico y asimétrico para proteger la confidencialidad como objetivo de seguridad. De este modo, el cifrado asimétrico se utiliza para el acuerdo de claves y el cifrado simétrico para proteger todos los demás mensajes enviados entre aplicaciones OPC UA.

3.1.4. Integridad

OPC UA utiliza firmas simétricas y asimétricas para abordar la integridad como objetivo de seguridad. Las firmas asimétricas se utilizan en la fase de acuerdo de claves durante el establecimiento del canal seguro. Las firmas simétricas se aplican a todos los demás mensajes, incluidos los mensajes de PubSub del paradigma de publicación/suscripción.

3.1.5. Auditabilidad

Las acciones tomadas por un sistema deben registrarse para proporcionar evidencia a las partes interesadas:

- Que este sistema funcione según lo previsto. Para ello se realiza un seguimiento de las acciones exitosas.
- Que identifican al iniciador de determinadas acciones. Por ello se realiza un seguimiento de la actividad del usuario.
- Que los intentos de comprometer el sistema fueron rechazados, por lo que se rastrean las acciones fallidas.

OPC UA admite el registro de auditoría al proporcionar una traza de las actividades a través de las entradas de registro de los múltiples clientes y servidores que inician, reenvían y manejan la actividad. OPC UA depende de los productos de aplicación OPC UA para proporcionar un esquema de registro de auditoría eficaz o una manera eficiente de recopilar los eventos de auditoría de todos los nodos.

3.1.6. Disponibilidad

OPC UA minimiza la pérdida de disponibilidad causada por la inundación de mensajes reduciendo la cantidad de procesamiento que se realiza en un mensaje antes de que el mismo sea autenticado. De esta manera, se evita que un atacante consiga que una aplicación legítima de OPC UA dedique una cantidad excesiva de tiempo y recursos a responder a mensajes maliciosos.

3.2. Seguridad en aplicaciones OPC UA

En cuanto a medidas de seguridad, las aplicaciones que funcionan bajo el estándar OPC UA, como método más utilizado y seguro, poseen certificados digitales que comparten durante la fase de conexión como mecanismo de autenticación. En cambio, en OPC clásico, no todos los clientes y servidores OPC soportan este tipo de restricciones, por lo que normalmente no se encuentra configurado.

La utilización de certificados sigue la estructura planteada en la figura 3:

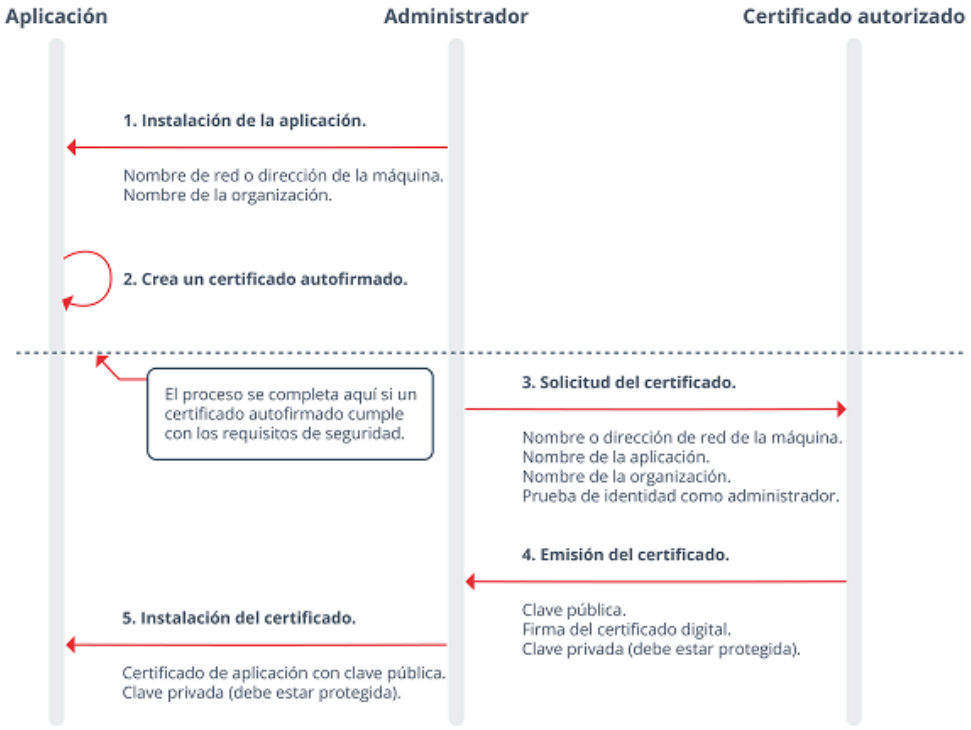


Figura 3: Seguridad con certificado autofirmado OPC UA [13]

Además, existen otros tipos de autenticación, como:

- Anonymous: sin ninguna autenticación del usuario.
- Usuario y contraseña: requiere de la creación de un usuario y de una contraseña.

Por otro lado, se configuran las políticas de seguridad, mediante las cuales OPC UA permite definir hasta tres perfiles de seguridad:

- None: no requiere seguridad.
- Sign: permite utilizar la estructura de PKI mediante Basic256. El perfil Basic256 de OPC UA emplea el cifrado AES (Advanced Encryption Standard) con una clave de 256 bits para garantizar la confidencialidad de las comunicaciones. Además, utiliza certificados digitales y firmas digitales, basadas en algoritmos como RSA, para la autenticación de los participantes.
- Sign & Encrypt: también requiere de una infraestructura PKI e incluye diferentes variantes, tanto Basic256Sha256 como Aes128-Sha256. El algoritmo AES (Advanced Encryption Standard), que utiliza una longitud de clave de 256 bits, se emplea junto con el algoritmo hash de 256 bits a través de SHA-256 en Basic256Sha256 para garantizar la integridad de los datos. Aes128-Sha256, por su parte, aplica el algoritmo AES, que tiene claves de 128 bits junto con hashes de 256 bits, para ofrecer un 100% de autenticidad y confidencialidad entre dos partes que se comunican a través de un canal abierto.

4. Implementación

4.1. Análisis de los certificados OPC UA

La OPC Foundation proporciona información detallada sobre los certificados de instancia de aplicación en su documentación oficial. Se puede consultar en el apartado correspondiente en [14].

Un certificado de instancia de aplicación es una cadena de bytes que contiene la forma codificada DER (Distinguished Encoding Rules) de un certificado X.509v3. Este certificado lo emite una autoridad certificadora e identifica una instancia de una aplicación que se ejecuta en un único host. Los campos X.509v3 contenidos en un certificado de instancia de aplicación se describen en la tabla 2.

Parámetro	Descripción
Certificado de instancia de aplicación	Un certificado X.509v3
Versión	V3
Número de serie	El número de serie asignado por el emisor.
Algoritmo de firma	El algoritmo utilizado para firmar el certificado.
Firma	La firma creada por el emisor.
Emisor	El nombre distinguido del certificado utilizado para crear la firma.
Validez	Cuándo el certificado entra en vigor y cuándo caduca.
Sujeto	El nombre completo de la instancia de la aplicación . Se especificará el atributo de nombre común y deberá ser el nombre del producto o un equivalente adecuado. El atributo Nombre de la organización será el nombre de la organización que ejecuta la instancia de la aplicación. Esta organización normalmente no es el proveedor de la aplicación. Se pueden especificar otros atributos.
Nombre alternativo del sujeto	Los nombres alternativos para la instancia de la aplicación. Deberá incluir un uniformResourceIdentifier que sea igual a applicationUri. El URI deberá ser una URL válida o una URN válida.

	Los servidores deberán especificar un nombre DNS parcial o completo o un nombre estático Dirección IP que identifica la máquina donde se ejecuta la instancia de la aplicación. Se pueden especificar nombres DNS adicionales si la máquina tiene varios nombres.
Clave pública	La clave pública asociada al certificado.
Uso de clave	Especifica cómo se puede utilizar la clave del certificado. Para claves RSA, el uso de claves incluirá firma digital, no repudio, cifrado de claves y cifrado de datos. Para claves ECC, el uso de claves incluirá firma digital. Se permiten otros bits de uso de claves, pero no se recomiendan. Los certificados autofirmados también incluirán keyCertSign.
Uso extendido de clave	Especifica límites adicionales sobre cómo se puede utilizar la clave del certificado . Para perfiles RSA, extendKeyUsage especificará serverAuth para servidores y clientAuth para clientes . ExtendedKeyUsage también debe especificar clientAuth para servidores. Para perfiles ECC, serverAuth y clientAuth son opcionales. Se permiten otros bits de uso de clave extendida.
Identificador de clave de entidad	Proporciona más información sobre la clave utilizada para firmar el certificado. Se debe especificar para certificados firmados por una CA. Se recomienda especificar para certificados autofirmados.
Restricciones básicas	El indicador CA identifica si el certificado pertenece a una Autoridad de Certificación (CA) y la longitud de la ruta especifica cuántas CAs intermedias pueden seguirlo. La extensión basicConstraints debe estar presente y será validada, pero marcarla como crítica no afecta su procesamiento. El indicador CA debe ser FALSO para los certificados de instancia de aplicación, a menos que se necesite para garantizar la interoperabilidad. Si se acepta un certificado de instancia de aplicación

	con el indicador CA VERDADERO, se emitirá una advertencia. Para certificados de instancia de aplicación autofirmados con el indicador CA VERDADERO, la longitud de la ruta debe ser 0.

Tabla 2: Certificado de Instancia de Aplicación

Para verificar que los certificados contienen la información que indica la OPC Foundation, es posible inspeccionar uno de los certificados de instancia de aplicación autofirmados creados automáticamente por uno de los simuladores en OpenSSL. Esto se presenta en la figura 4:

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            01:8e:f5:ae:a0:24
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: CN=SimulationServer@DESKTOP-KU7FIQQ, O=Prosys OPC, DC=DESKTOP-KU7FIQQ
        Validity
            Not Before: Apr 19 08:27:25 2024 GMT
            Not After : Apr 17 09:27:25 2034 GMT
        Subject: CN=SimulationServer@DESKTOP-KU7FIQQ, O=Prosys OPC, DC=DESKTOP-KU7FIQQ
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:c9:94:bf:70:3d:51:75:21:bd:fb:96:4f:1b:4a:
                    7f:d3:d3:32:a4:18:bd:32:3b:f1:2a:58:8f:0b:96:
                    45:4d:e0:4b:70:30:ed:a2:44:bc:eb:c7:29:61:b7:
                    3b:ad:e9:0e:d2:17:5d:52:71:d5:00:b6:cb:45:24:
                    3a:5b:8e:50:9c:a6:7a:63:d1:0c:c6:90:80:e4:64:
                    a5:c4:93:38:3d:61:5f:c6:82:73:d3:59:83:97:b4:
                    47:ec:39:7a:6c:8b:de:5f:37:a8:2c:c0:d6:79:28:
                    09:2d:50:e7:ed:ca:76:45:9e:48:2e:97:a0:0d:8e:
                    c4:e0:0d:c6:c1:2a:1e:e1:5a:d4:e3:70:43:9b:f7:
                    f5:41:02:51:f2:e3:1b:ab:f7:39:71:8d:4e:f0:00:
                    36:9c:31:7d:2a:01:30:cf:23:bf:49:5d:72:4e:41:
                    32:eb:a0:b3:da:29:ea:0f:6f:0b:87:81:7d:83:b0:
                    be:8f:24:d7:7c:85:45:37:f5:a2:4e:d1:77:12:b7:
                    82:08:12:7d:a3:f7:c1:03:6e:3b:1d:05:b8:cd:a6:
                    b0:d8:be:64:0e:d4:47:14:00:e5:9c:35:03:31:cc:
                    bf:ed:ba:a5:18:3d:a0:83:f2:a4:32:a1:4b:ef:63:
                    d3:26:ef:c8:6b:3a:b7:fd:e7:93:56:f4:e4:51:1d:
                    36:49
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Authority Key Identifier:
                BA:0F:DE:87:03:75:B1:CA:88:BA:75:26:C0:6F:E7:7A:DD:1D:5C:66
            X509v3 Subject Key Identifier:
                BA:0F:DE:87:03:75:B1:CA:88:BA:75:26:C0:6F:E7:7A:DD:1D:5C:66
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Key Usage:
                Digital Signature, Non Repudiation, Key Encipherment, Data Encipherment, Certificate Sign
            X509v3 Extended Key Usage:
                TLS Web Server Authentication, TLS Web Client Authentication
            X509v3 Subject Alternative Name:
                URI:urn:DESKTOP-KU7FIQQ:OPCUA:SimulationServer, DNS:DESKTOP-KU7FIQQ
    Signature Algorithm: sha256WithRSAEncryption
    Signature Value:
        63:0b:a3:c3:72:b0:5c:6d:75:b5:3d:81:82:93:ca:85:2f:9f:
        0b:c7:38:8d:6e:8a:ca:76:fb:2b:2c:f3:94:7b:b9:80:b9:f8:
```

Figura 4: Certificado de instancia de aplicación autogenerado por Prosys OPC

4.2. Generación de certificados OPC UA para entornos de simulación

4.2.1. Simuladores OPC UA

➢ **Prosys OPC UA Simulation Server**

Prosys OPC UA Simulation Server es una herramienta de simulación del lado del servidor que ha sido desarrollada por Prosys OPC de acuerdo con el estándar OPC UA. Permite simular un entorno industrial para propósitos de pruebas y desarrollo. Con esta herramienta es posible crear y configurar diferentes elementos como nodos y variables que representan dispositivos y datos en un sistema industrial. Esto facilita a los desarrolladores de aplicaciones cliente OPC UA, así como a los ingenieros de sistemas de monitoreo y control, y otros usuarios de dispositivos electrónicos industriales que utilizan OPC UA como protocolo de comunicación, probar cómo están funcionando los productos o sistemas en situaciones prácticas.

➢ **TOP Server 6**

Top Server 6 UA Simulator es una herramienta de simulación desarrollada por Software Toolbox que puede utilizarse para experimentación, revisión y capacitación de servidores OPC UA en situaciones industriales y de automatización. Esta herramienta se puede utilizar para modelar la respuesta de sistemas de control industrial (ICS), dispositivos de automatización de máquinas y promover la interoperabilidad de dispositivos de automatización industrial compatibles con el estándar.

➢ **UaExpert**

UaExpert es una herramienta de cliente OPC UA desarrollada por Unified Automation GmbH. UaExpert está diseñado como un cliente de prueba de uso general que es capaz de soportar una variedad de métodos de prueba como acceso a datos, alarmas y condiciones, acceso histórico y funciones de llamada de métodos UA. Este software es ampliamente utilizado en entornos industriales para la supervisión y el control de procesos, así como para el desarrollo y la depuración de aplicaciones basadas en OPC UA.

4.2.2. Herramientas de generación de certificados

➢ **OpenSSL**

OpenSSL es un conjunto de herramientas de código abierto que ofrece implementaciones de diversas funciones criptográficas (cifrado, creación de claves, capa de comunicaciones, etc.). OpenSSL es muy versátil y se utiliza en diversas aplicaciones y plataformas. Permite realizar tareas como la generación de claves, la creación de solicitudes de certificados, la firma de certificados, la gestión de autoridades de certificación y la configuración de servicios de seguridad.

➢ **UA Configuration Tool**

UA Configuration Tool es una herramienta utilizada para configurar y administrar sistemas que implementan el estándar OPC UA. Esta herramienta ofrece una interfaz de usuario amigable con la que el administrador puede personalizar varias características de los servidores OPC UA, incluida la adición de puntos de conexión, la configuración de seguridad y la gestión de usuarios y roles, así como la implementación de servicios específicos.
Dentro de las posibles configuraciones que ofrece esta aplicación, se encuentra la gestión de certificados. Esta función permite la importación de certificados de autoridades de certificación, la generación de certificados autofirmados y su asignación a un servidor OPC UA específico.

➢ **XCA**

XCA es una herramienta de código abierto con una interfaz de usuario amigable que permite gestionar de manera organizada los certificados digitales y establecer una infraestructura de clave pública. XCA es capaz de proporcionar un entorno para crear y firmar certificados, llevar a cabo las funciones de las autoridades de certificación y realizar tareas como la revocación de certificados y la configuración de políticas de seguridad.

4.2.3. Análisis de certificados de instancia de aplicación

Al iniciar por primera vez cada una de las aplicaciones, estas emiten automáticamente los certificados autofirmados que cada una requiere:

En el caso de UaExpert, se genera un certificado autofirmado con los parámetros tal como se refleja en la tabla 3:

Versión	V3
Número de serie	El número de serie asignado por el emisor.
Algoritmo de firma	sha256RSA
Algoritmo hash de firma	sha256
Emisor	CN = UaExpert@[nombre del equipo] O = Asignado por el usuario
Validez	5 años a partir de su generación.
Sujeto	Mismos parámetros que el emisor: CN = UaExpert@[nombre del equipo] O = Asignado por el usuario
Nombre alternativo del titular	Dirección URL=urn:[nombre del equipo]:UnifiedAutomation:UaExpert Nombre DNS=[nombre del equipo]

Clave pública	RSA (2048 bits)
Uso de clave	Firma digital, Sin repudio, Cifrado de clave, Cifrado de datos, Firma de certificados
Uso extendido de clave	Autenticación del servidor (1.3.6.1.5.5.7.3.1) Autenticación del cliente (1.3.6.1.5.5.7.3.2)
Identificador de clave de entidad	Id. de clave Emisor de certificado: Dirección del directorio: CN=UaExpert@[nombre del equipo] O=Asignado por el usuario Número de serie del certificado
Restricciones básicas	Tipo de asunto=Entidad de certificación (CA) Restricción de longitud de ruta=0

Tabla 3: Parámetros del certificado de instancia de aplicación autofirmado de UaExpert

En el caso de Prosys OPC Simulation Server, genera un certificado de instancia de aplicación similar al que se puede ver en la tabla 4 para identificar el servidor OPC UA en la red. Este certificado está firmado por una Autoridad de Certificación (CA) interna, que en este caso es la SimulationServerCA:

Versión	V3
Número de serie	El número de serie asignado por el emisor.
Algoritmo de firma	sha256RSA
Algoritmo hash de firma	sha256
Emisor	CN = SimulationServerCA
Validez	10 años a partir de su generación.
Sujeto	Mismos parámetros que el emisor: CN = SimulationServerCA
Clave pública	RSA (2048 bits)
Uso de clave	Firma digital, Firma de certificados, Firma CRL sin conexión, Firma de lista de revocación de certificados (CRL)
Identificador de clave de entidad	Id. de clave: Coincide con el Id. de clave del titular.
Restricciones básicas	Tipo de asunto=Entidad de certificación (CA)

	Restricción de longitud de ruta=Ninguno

Tabla 4: Parámetros del certificado de CA autofirmado de Prosys OPC

El certificado expedido con el nombre "SimulationServer@[nombre del equipo]_https_2048" es utilizado para establecer conexiones seguras a través del protocolo HTTPS, permitiendo una comunicación segura entre el servidor OPC UA y los clientes que se conectan a través de este protocolo:

Versión	V3
Número de serie	El número de serie asignado por el emisor.
Algoritmo de firma	sha256RSA
Algoritmo hash de firma	sha256
Emisor	CN = SimulationServerCA
Validez	10 años a partir de su generación.
Sujeto	CN = [nombre del equipo].mshome.net
Nombre alternativo del titular	Dirección URL=urn:[nombre del equipo].mshome.net:OPCUA:SimulationServer Nombre DNS=[nombre del equipo].mshome.net
Clave pública	RSA (2048 bits)
Uso de clave	Firma digital, Sin repudio, Cifrado de clave, Cifrado de datos, Firma de certificados
Uso extendido de clave	Autenticación del servidor (1.3.6.1.5.5.7.3.1) Autenticación del cliente (1.3.6.1.5.5.7.3.2)
Identificador de clave de entidad	Id. de clave: Coincide con el Id de clave del titular de la CA. Emisor de certificado: Dirección del directorio: CN=SimulationServerCA Número de serie del certificado
Restricciones básicas	Tipo de asunto=Entidad final Restricción de longitud de ruta=Ninguno

Tabla 5: Parámetros del certificado de instancia de aplicación firmado por la CA de Prosys OPC

Por otro lado, el certificado "SimulationServer@[nombre del equipo]_2048" es un certificado autofirmado que también puede ser utilizado para la comunicación OPC UA, pero no necesariamente garantiza la misma seguridad que el certificado firmado por la CA interna. Este certificado autofirmado puede ser útil para pruebas o para entornos donde la

seguridad no sea una preocupación primordial. En el caso de un entorno de simulación es este el certificado con el que el servidor se presenta al cliente UaExpert:

Versión	V3
Número de serie	El número de serie asignado por el emisor.
Algoritmo de firma	sha256RSA
Algoritmo hash de firma	sha256
Emisor	DC = [nombre del equipo].mshome.net O = Prosys OPC CN = SimulationServer@[nombre del equipo]
Validez	10 años a partir de su generación.
Sujeto	Coincide con los parámetros del emisor: DC = [nombre del equipo].mshome.net O = Prosys OPC CN = SimulationServer@[nombre del equipo]
Nombre alternativo del titular	Dirección URL=urn:[nombre del equipo].mshome.net:OPCUA:SimulationServer Nombre DNS=[nombre del equipo].mshome.net
Clave pública	RSA (2048 bits)
Uso de clave	Firma digital, Sin repudio, Cifrado de clave, Cifrado de datos, Firma de certificados
Uso extendido de clave	Autenticación del servidor (1.3.6.1.5.5.7.3.1) Autenticación del cliente (1.3.6.1.5.5.7.3.2)
Identificador de clave de entidad	Id. de clave: Coincide con el Id de clave del titular de este mismo certificado.
Restricciones básicas	Tipo de asunto=Entidad final Restricción de longitud de ruta=Ninguno

Tabla 6: Parámetros del certificado de instancia de aplicación autofirmado de Prosys OPC

La elección de explorar otra opción de servidor OPC UA surgió del interés en ampliar la implementación del proyecto en diferentes entornos. Al probar los certificados en una alternativa distinta, se busca confirmar su efectividad en una variedad de configuraciones y verificar la adaptabilidad a diferentes contextos. Este servidor denominado TOP Server utiliza también un certificado autofirmado con los parámetros indicados en la tabla 7:

Versión	V3

Número de serie	El número de serie asignado por el emisor.
Algoritmo de firma	sha256RSA
Algoritmo hash de firma	sha256
Emisor	CN = TOP Server/UA Server O = Unknown C = ES DC = [nombre del equipo]
Validez	3 años a partir de su generación.
Sujeto	Coincide con los parámetros del emisor: CN = TOP Server/UA Server O = Unknown C = ES DC = [nombre del equipo]
Nombre alternativo del titular	Dirección URL=urn:[nombre del equipo]:SWToolbox.TOPServer.V6:UA%20Server Nombre DNS=[nombre del equipo]
Clave pública	RSA (2048 bits)
Uso de clave	Firma digital, Sin repudio, Cifrado de clave, Cifrado de datos, Firma de certificados
Uso extendido de clave	Autenticación del servidor (1.3.6.1.5.5.7.3.1) Autenticación del cliente (1.3.6.1.5.5.7.3.2)
Identificador de clave de entidad	Emisor de certificado: Dirección del directorio: CN=TOP Server/UA Server O=Unknown C=ES DC=[nombre del equipo] Número de serie del certificado
Restricciones básicas	Tipo de asunto=Entidad final Restricción de longitud de ruta=Ninguno

Tabla 7: Parámetros del certificado de instancia de aplicación autofirmado de TOP Server

4.2.4. Generación y sustitución de certificados

Una vez se conoce esta información, el primer paso es tratar de replicar estos certificados con las herramientas descritas anteriormente y asegurar que las aplicaciones los reconocen y son capaces de establecer comunicaciones seguras haciendo uso de ellos.

En primer lugar se consideró emitir estos certificados mediante la herramienta UA Configuration Tool. Estos certificados se crean por defecto con un algoritmo de firma SHA-1 (de 160 bits) en lugar de SHA-256. SHA-1 se considera débil y vulnerable a ataques de colisión, donde dos conjuntos de datos diferentes pueden generar el mismo valor hash. Debido a esto, su uso se desaconseja en aplicaciones donde se requiere una alta seguridad. Dado que no es posible modificar este parámetro ni acceder a las claves privadas asociadas, se decidió emplear otra alternativa para generar los certificados.

Seguidamente, se trató de replicar estos certificados con la herramienta XCA. Con esta herramienta es posible generar un **certificado autofirmado** para UaExpert, que la aplicación acepta como válido y permite usar para realizar conexiones con el servidor a pesar de un warning *BadCertificateHostNameInvalid*, mostrado en la figura 5 que es posible ignorar para continuar con la conexión.

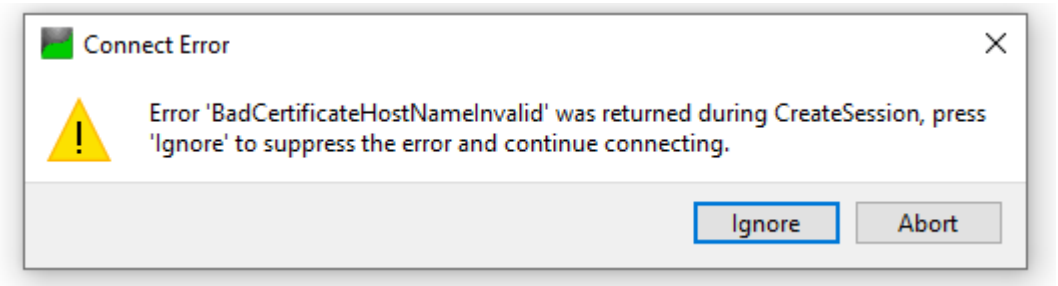


Figura 5: Error emitido por el cliente OPC UA al establecer conexión con el servidor.

En el caso de Prosys OPC, en primer lugar, se emite el certificado para la CA, el cual la aplicación acepta y utiliza para crear a su vez el certificado para conexiones HTTPS. Sin embargo, como escapa del alcance de este proyecto, no es posible publicar este servicio para probar que la comunicación se establece correctamente con estos certificados.

En cuanto al certificado autofirmado, a pesar de que no es posible configurar el parámetro DC (Domain Component) en XCA, se genera un certificado con el que es posible conectarse y enviar datos a un cliente UaExpert.

De la misma manera, es posible expedir un certificado autofirmado con los parámetros necesarios para el servidor OPC UA TOP Server 6 con el cual es posible entablar comunicación con el cliente.

Estos certificados autofirmados no son considerados fiables porque no están respaldados por una autoridad de certificación confiable y reconocida. En un certificado autofirmado, la entidad que emite el certificado se autentica a sí misma mediante la firma digital del certificado, en lugar de ser autenticada por una entidad externa de confianza. Por tanto, el próximo paso a seguir es crear una autoridad certificadora propia para firmar estos certificados. Para ello se recurre a OpenSSL, ya que es sencillo especificar los parámetros de cada certificado en un fichero de configuración.

Paso 1: Definición de las especificaciones de la CA y los certificados a emitir.

En primera instancia se genera una autoridad certificadora para firmar el certificado del cliente UaExpert. Para ello, se usa el fichero de configuración *openssl.cnf* mostrado en la figura 6:

```
CA_DIR = .
RANDFILE = $ENV::CA_DIR/.rnd
KEY_SIZE = 2048

[ ca ]
default_ca = CA_default

[ CA_default ]
dir = $ENV::CA_DIR
certs = $dir/certs
database = $dir/index.txt
new_certs_dir = $certs
certificate = $certs/rootCA.der
serial = $dir/serial
crl_dir = $dir/crl
crlnumber = $dir/crlnumber
crl = $dir/crl.pem
private_key = $dir/private/rootCA_key.pem
RANDFILE = $dir/private/.rand
x509_extensions = v3_ext
default_days = 365
crl_extensions = crl_ext
default_crl_days = 30
default_md = default
preserve = no
policy = policy_match

[ req ]
default_bits       = 2048
distinguished_name = req_distinguished_name
prompt             = no

[ req_distinguished_name ]
O  = TFM_Iris
CN = rootCA

[ v3_ext ]
basicConstraints     = critical,CA:true,pathlen:0
subjectKeyIdentifier = hash
authorityKeyIdentifier=keyid:always,issuer:always
keyUsage = digitalSignature, nonRepudiation, keyEncipherment, dataEncipherment,
keyCertSign
extendedKeyUsage = serverAuth, clientAuth

[ crl_ext ]
```

```
authorityKeyIdentifier=keyid:always, issuer:always
```

Figura 6: Sección del fichero de configuración para generar certificados en OpenSSL

El archivo de configuración de OpenSSL anterior abarca las secciones esenciales para la configuración de una autoridad certificadora y la emisión de certificados digitales. Las secciones *[ca]* y *[CA_default]* especifican detalles importantes como el directorio de la CA, la ubicación de los certificados, la base de datos de la CA, el directorio para los nuevos certificados emitidos, el archivo de certificado de la CA raíz, entre otros. La sección *[req]* define los parámetros para las solicitudes de certificados. Aquí se especifica el tamaño predeterminado de la clave y se establece la configuración para los nombres distintivos de las solicitudes de certificados. El apartado *[req_distinguished_name]* se utiliza para configurar los campos del nombre distintivo que se incluirán en las solicitudes de certificados. La sección *[v3_ext]* se refiere a las extensiones X.509 del certificado. Aquí se definen configuraciones adicionales para el certificado, como restricciones básicas, identificadores de clave y usos permitidos del certificado. Finalmente, el fragmento *[crl_ext]* define las extensiones adicionales que se incluirán en la CRL generada, lo que garantiza que la CRL contenga la información necesaria para identificar la autoridad de certificación y el emisor del certificado revocado.

Paso 2: Creación de la estructura de directorios

A continuación se crea la estructura de directorios para la CA y un fichero index.txt que actúa como base de datos, almacenando información sobre los certificados emitidos. También se crea un fichero denominado crlnumber para llevar un registro del número de serie de la lista de revocación de certificados más reciente emitida por la CA.

Paso 3: Generación de la clave privada de la CA raíz

Seguidamente, se crea una clave privada de 2048 bits para la CA raíz:

```
openssl genrsa -out private/rootCA.key 2048
```

Paso 4: Generación del certificado de la CA raíz

Y se emite el certificado correspondiente a partir de esta clave:

```
openssl req -x509 -new -nodes -key private/rootCA.key -days 3650 -out
certs/rootCA.crt -config rootCA.cnf -extensions v3_ext
```

Paso 5: Generación de certificados de instancia de aplicación

Una vez creada la CA raíz, se crea una nueva clave para el certificado de instancia de aplicación del cliente:

```
openssl genrsa -out private/uaexpert.key 2048
```

Esta clave se emplea para crear una solicitud de firma del certificado para la CA raíz:

```
openssl req -new -key private/uaexpert.key -out uaexpert.csr -config
uaexpert.cnf
```

Y se firma el certificado con la CA raíz:

```
openssl x509 -req -in uaexpert.csr -CA certs/rootCA.crt -CAkey
private/rootCA.key -CAcreateserial -out certs/uaexpert.crt -days 3650
-sha256 -extfile uaexpert.cnf -extensions v3_ext
```

Paso 6: Generación de la lista de revocación de certificados

Adicionalmente, se genera la CRL (Certificate Revocation List) de la CA raíz:

```
openssl ca -gencrl -config rootCA.cnf -out crl/crl.pem
```

Paso 7: Despliegue de los certificados

Se colocan estos certificados en la PKI de UaExpert, el de la rootCA en el directorio issuers, y el de UaExpert en el directorio own. Prosys OPC devuelve un error ya que no puede validar el certificado de la CA raíz:

INFO [OPC-UA-Stack-Non-Blocking-Work-Executor-4] com.prosysopc.ua.stack.cert.d [] - Certificate '5B43F4F8BF2306D3E3B00752742D5473799E0DA7' added to rejected certificates.
WARN [OPC-UA-Stack-Non-Blocking-Work-Executor-1] com.prosysopc.ua.stack.transport.tcp.nio.g [] - Remote certificate not accepted: Bad_CertificateChainIncomplete (0x810D0000) "The certificate chain is incomplete."

Figura 7: Error *Bad_CertificateChainIncomplete* en los logs de Prosys OPC

Al añadir el certificado y la CRL de la CA raíz al directorio *issuers* de la PKI de Prosys OPC, los logs muestran que se ha inicializado correctamente la CRL y que el certificado se ha añadido como rechazado, por lo que hay que aceptarlo manualmente en la aplicación para que funcione correctamente.

INFO [OPC-UA-Stack-Non-Blocking-Work-Executor-4] com.prosysopc.ua.stack.cert.d [] - CRL initialized from C:\Users\Usuario\.prosysopc\prosys-opc-ua-simulation-server\PKI\CA\issuers\crl\crl.crl: no revoked certificates
INFO [OPC-UA-Stack-Non-Blocking-Work-Executor-4] com.prosysopc.ua.stack.cert.d [] - Certificate '5B43F4F8BF2306D3E3B00752742D5473799E0DA7' added to rejected certificates.
INFO [OPC-UA-Stack-Non-Blocking-Work-Executor-4] fcom.prosysopc.ua.stack.cert.d [] - Certificate '5B43F4F8BF2306D3E3B00752742D5473799E0DA7' added to trusted certificates.

Figura 8: Logs de Prosys OPC

Sabiendo que esto funciona a la perfección, se trata de repetir este proceso para generar los certificados del servidor OPC UA a partir de la CA creada previamente:

```
# Crear la Clave Privada de la CA de Prosys OPC
openssl genrsa -out private/prosysCA.key 2048 &&

# Crear una solicitud de firma del certificado (CSR) para la CA
openssl req -new -key private/prosysCA.key -out prosysCA.csr -config
prosys_ca.cnf &&

# Firmar el certificado con la CA raíz
openssl x509 -req -in prosysCA.csr -CA certs/rootCA.crt -CAkey
private/rootCA.key -CAcreateserial -out certs/prosysCA.crt -days 3650
-sha256 -extfile prosys_ca.cnf -extensions v3_ext &&

# Generar la clave privada del servidor Prosys OPC
openssl genrsa -out private/prosys.key 2048 &&

# Generar la CSR
openssl req -new -key private/prosys.key -out prosys.csr -config
prosys.cnf &&

# Firmar el certificado con la CA raíz
openssl x509 -req -in prosys.csr -CA certs/rootCA.crt -CAkey
private/rootCA.key -CAcreateserial -out certs/prosys.crt -days 3650
-sha256 -extfile prosys.cnf -extensions v3_ext
```

Figura 9: Código para generar los certificados del servidor OPC UA con OpenSSL

Sin embargo, al tratar de entablar conexión desde el cliente y a pesar de que se indica que la CA raíz es un emisor de certificados reconocido, se obtienen los siguientes errores:

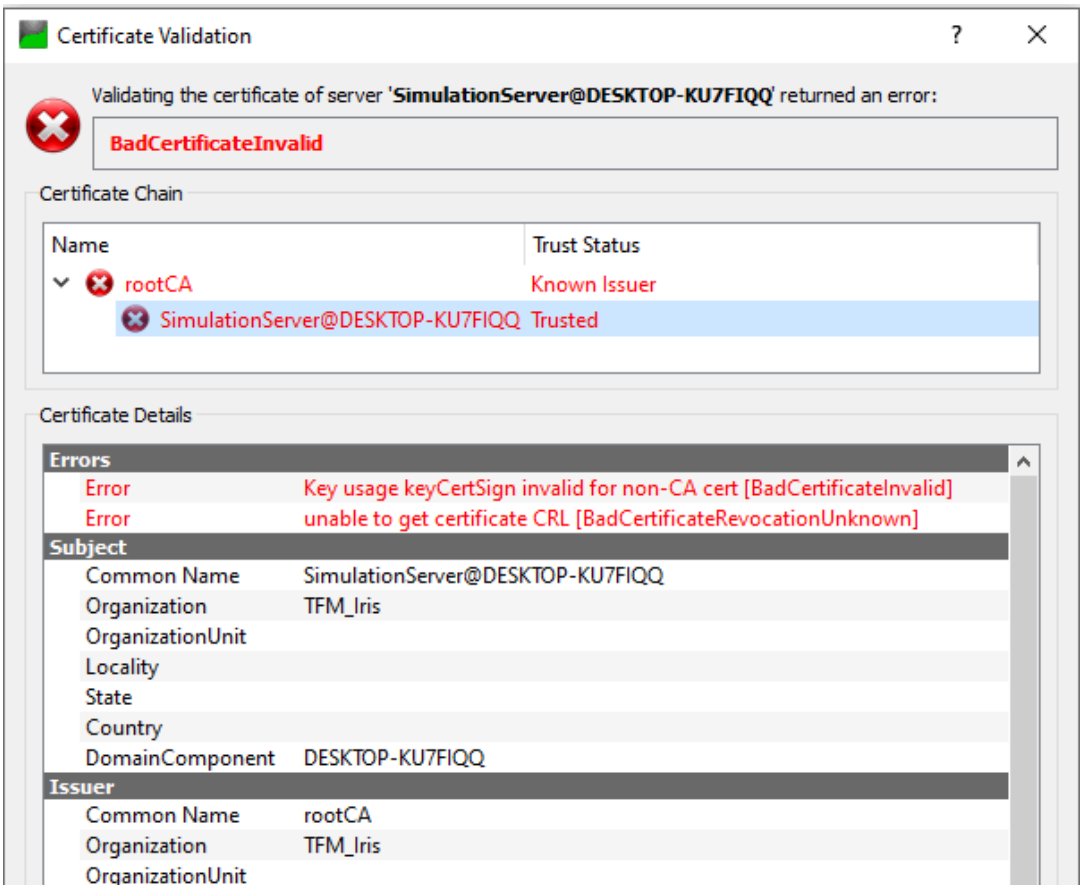


Figura 10: Errores devueltos por el cliente relativos al certificado del servidor

El primero de ellos indica que el certificado tiene habilitado el bit *keyCertSign* a pesar de no ser una CA. Este error no se mostraba en el certificado anterior a pesar de estar este parámetro contemplado en el uso de clave. No obstante, al haber sustituido un certificado autofirmado por otro firmado por una CA raíz, pueden haber cambiado las condiciones del mismo. Para solventarlo, se eliminó del fichero de configuración este bit en el apartado *keyUsage*.

El segundo de ellos indica que no se encuentra la CRL asociada a la CA raíz. Esto es debido a que esta lista se guardó en la PKI en formato *.der*. De acuerdo a la documentación oficial de UaExpert [15], este error se solventa cambiando el formato de la CRL a *.crl* o *.pem*.

Finalmente, es posible comunicar ambos simuladores utilizando las políticas de seguridad *SignAndEncrypt* y *Basic256Sha256*, como se muestra en la figura 13, que corresponde a los logs producidos por el servidor OPC UA, y es posible extraer datos de este y visualizarlos en el cliente UaExpert, como puede verse en la figura 14. Sumado a esto, se puede apreciar en las figuras 15 y 16 que los certificados que se están empleando en ambas aplicaciones están firmados por el emisor *rootCA* y pertenecen a la organización

TFM_Iris. Estos parámetros coinciden con los especificados en el fichero de configuración y son el distintivo que indica que efectivamente son los que se han generado durante este procedimiento

Además, se puede observar la estructura de directorios final de cada una de las aplicaciones en las figuras 11 y 12 una vez se han colocado los certificados en las ubicaciones adecuadas

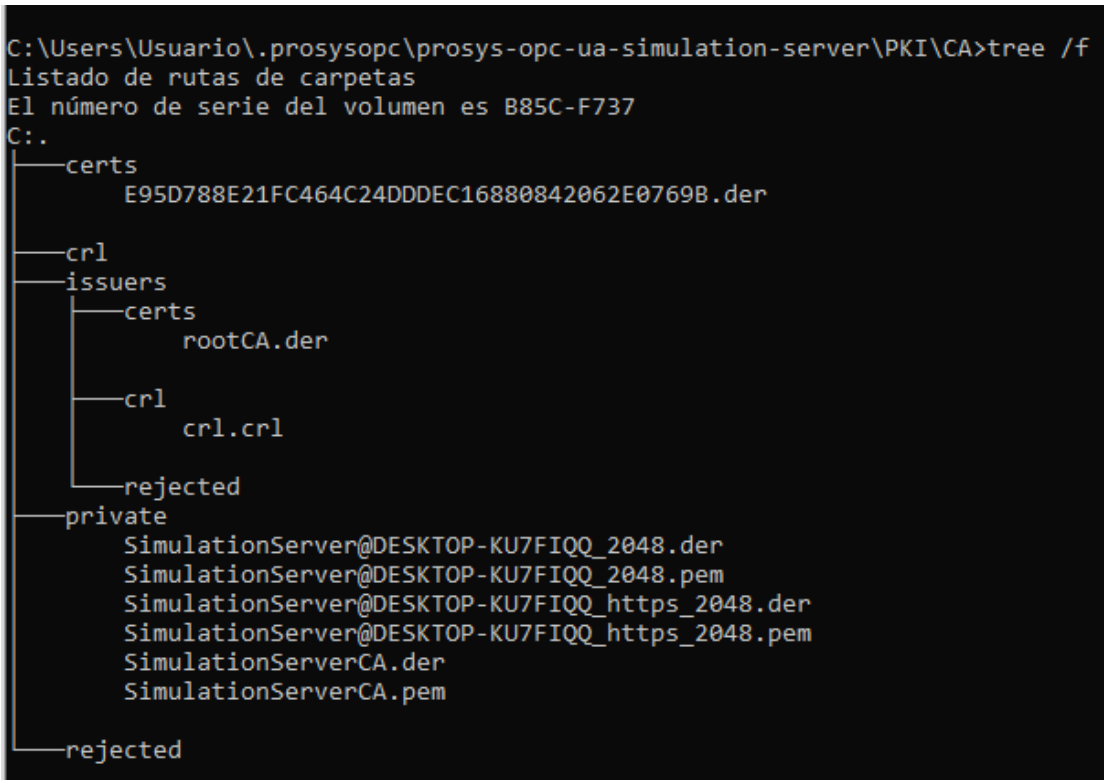


Figura 11: Estructura de la PKI de la aplicación Prosys OPC

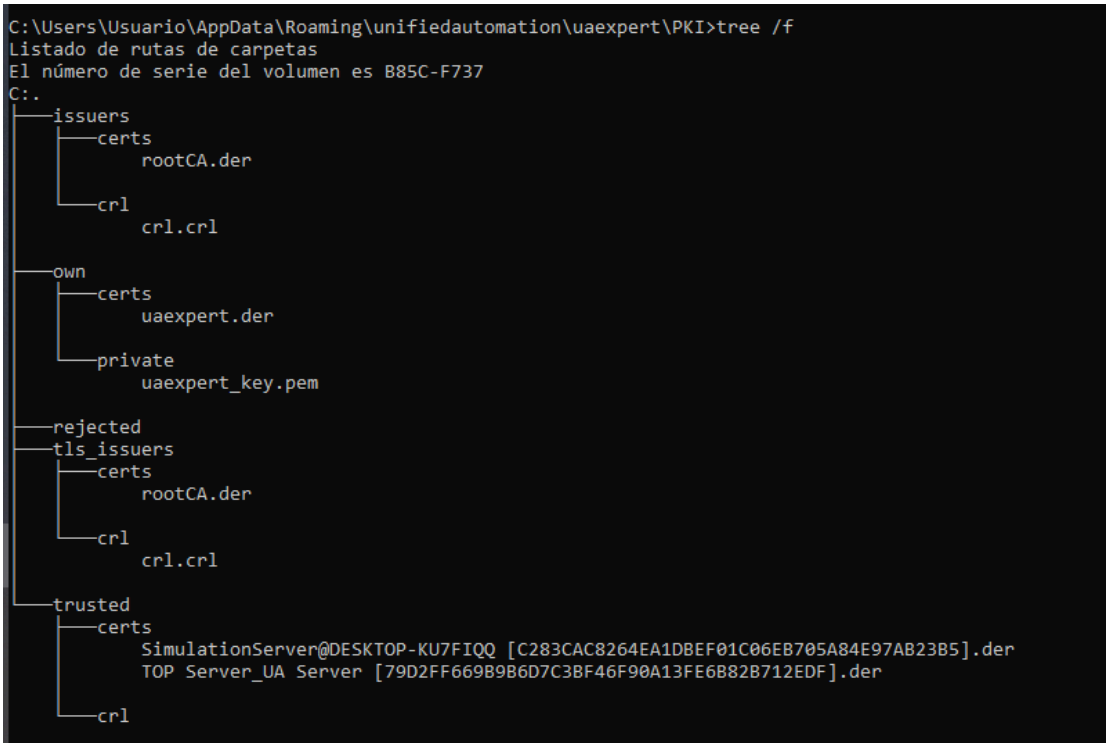


Figura 12: Estructura de la PKI de la aplicación UaExpert

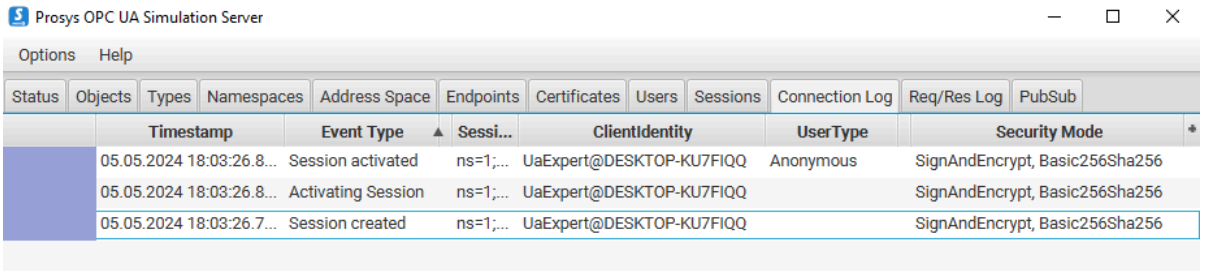


Figura 13: Logs de conexión del servidor Prosys OPC

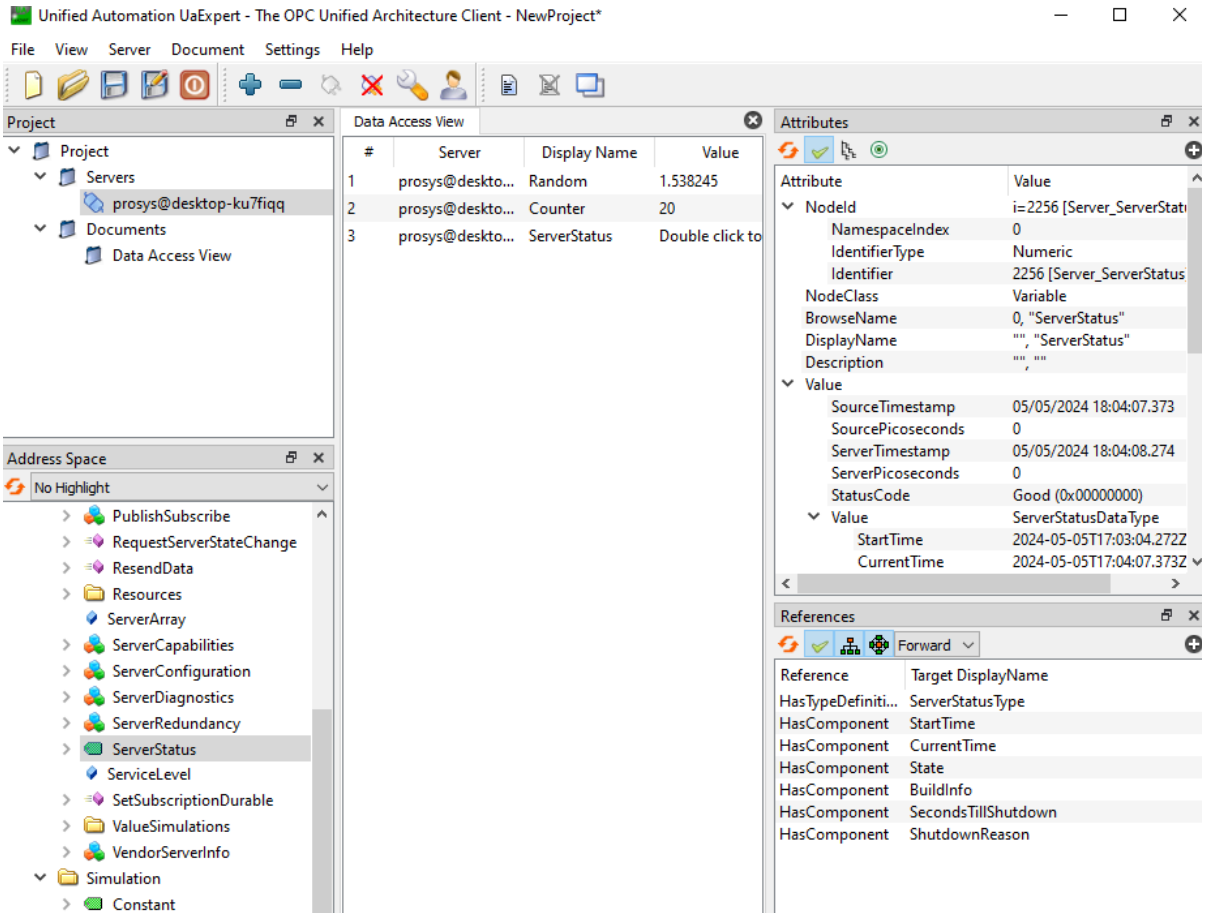


Figura 14: Captura de datos desde el servidor Prosys OPC

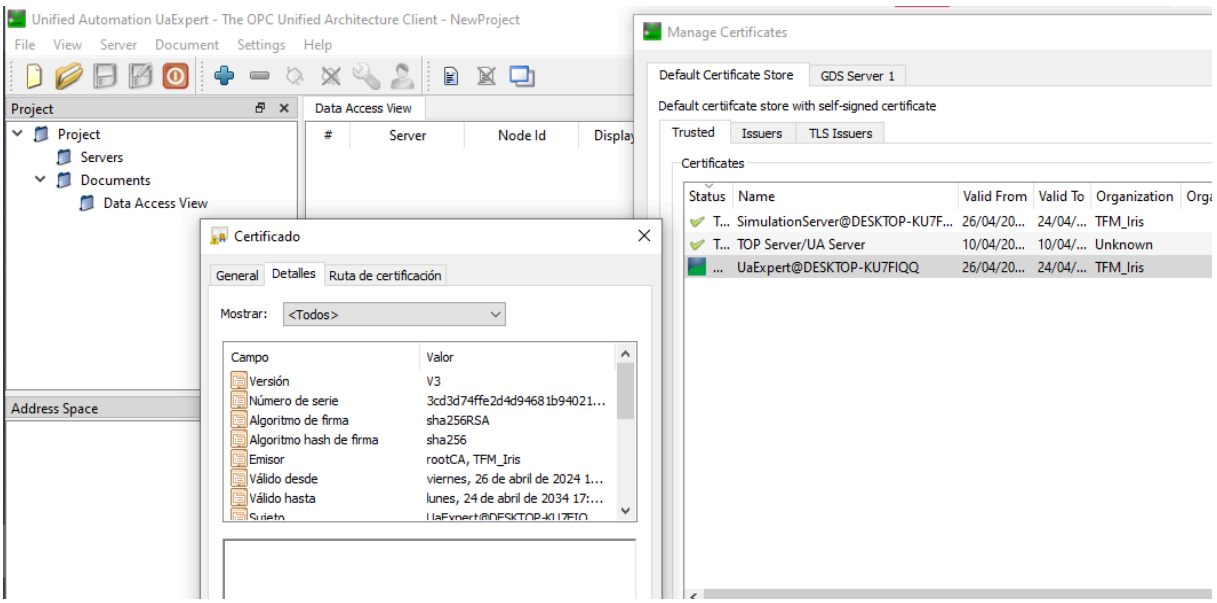


Figura 15: Certificado empleado por UaExpert tras la sustitución

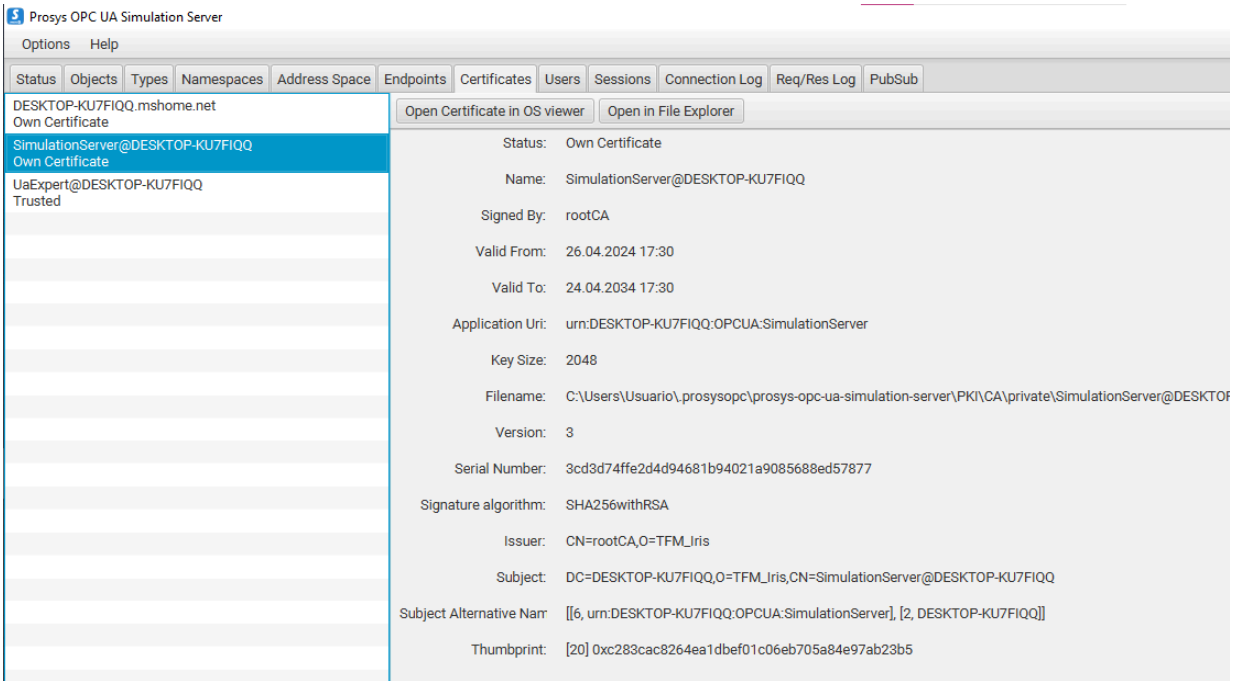


Figura 16: Certificado empleado por Prosys OPC tras la sustitución

En el caso del servidor TOP Server, se trató de expedir un certificado de instancia de aplicación a partir de una CA raíz del mismo modo que para el servidor anterior. Una vez se obtiene este certificado, es necesario importarlo en la aplicación manualmente, como se ve en la figura 17.

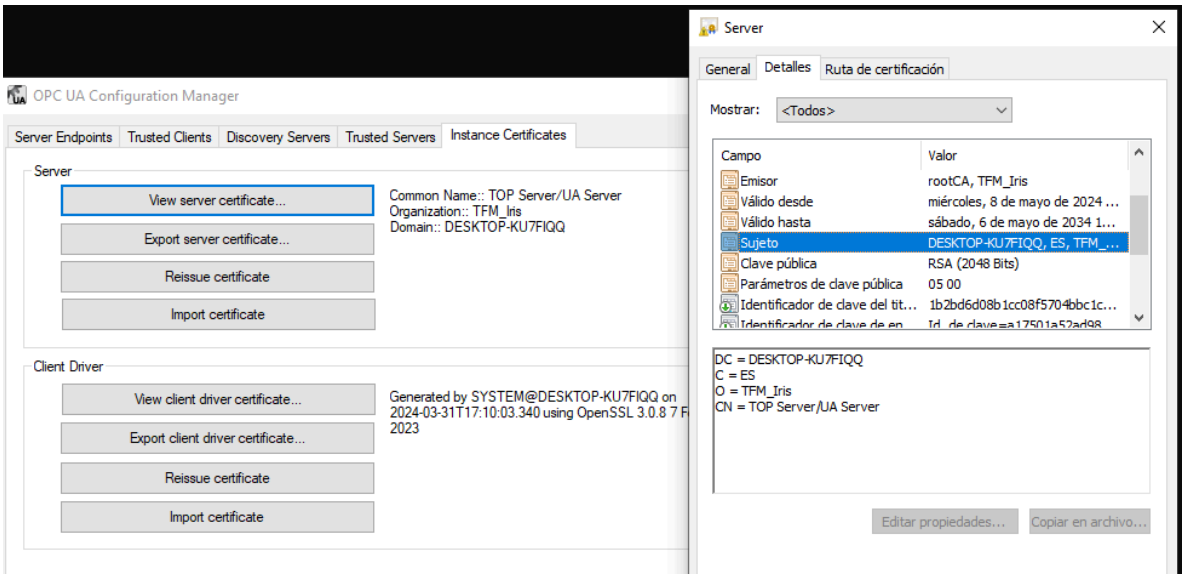


Figura 17: Certificado del servidor TOP Server firmado por una CA

Al establecer la comunicación con el cliente, del mismo modo que en el caso anterior, es necesario aceptar los certificados manualmente. Sin embargo, se obtiene este mensaje de

advertencia que se ve en la figura 18, que indica que se debe importar el certificado de la autoridad certificadora del cliente de forma manual para poder proceder.

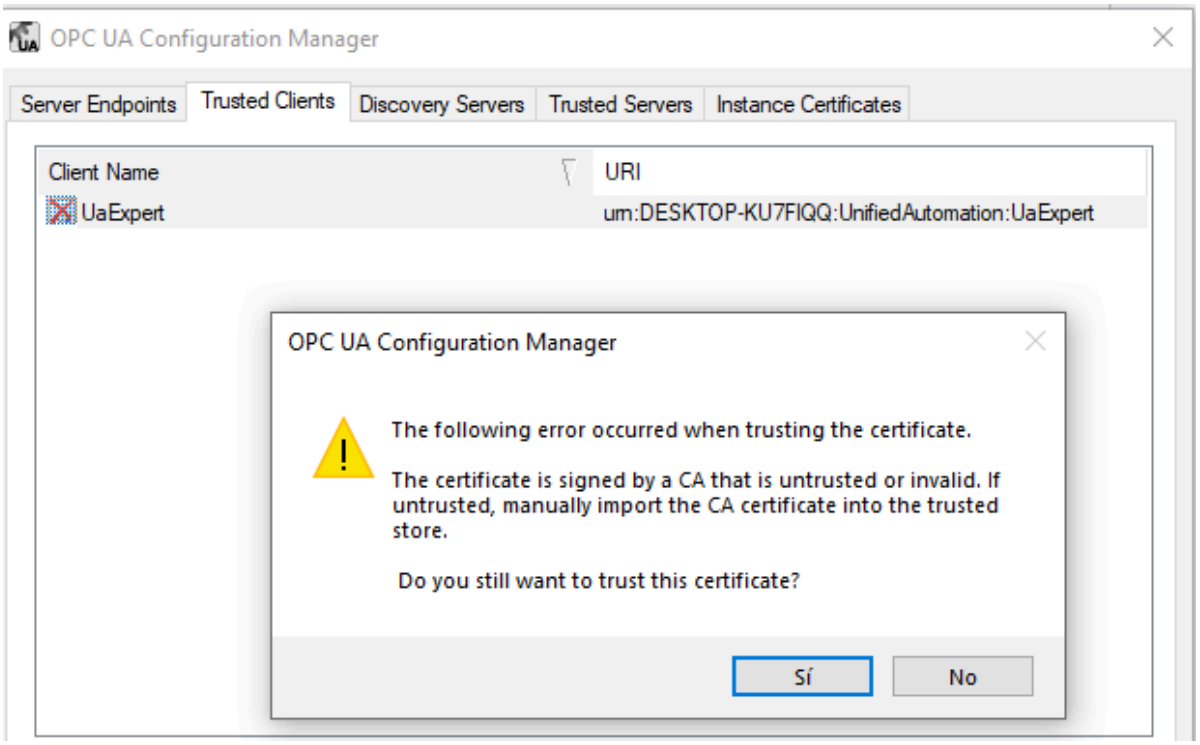


Figura 18: Warning que insta a importar el certificado de la CA raíz

Una vez se importa este certificado, la pestaña *Trusted Clients* en el servidor se debe ver del mismo modo que en la figura 19.

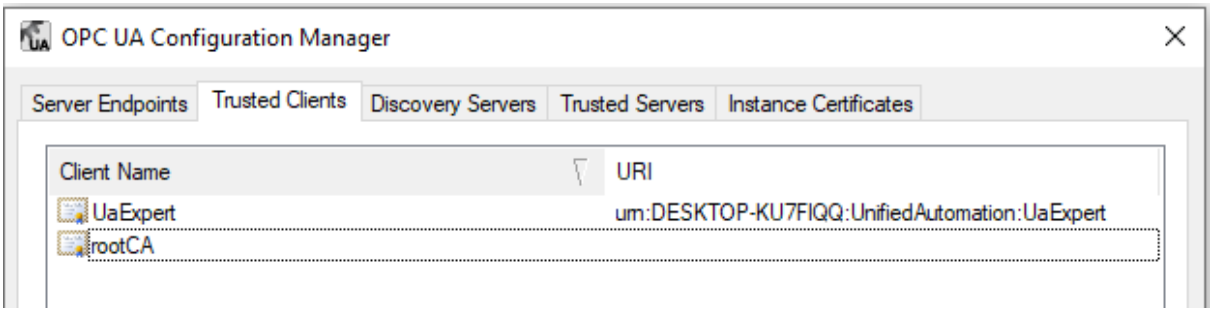


Figura 19: Pestaña *Trusted Clients* de TOP Server con los certificados del cliente

A continuación, se debe reinicializar el servidor para aplicar los cambios. Al hacerlo y tratar de realizar la conexión, el cliente pide aceptar manualmente el certificado como se aprecia en la figura 20. Cabe destacar que anteriormente se incluyó el certificado de la autoridad que emite el certificado del servidor en el directorio *issuers* de la PKI de UaExpert.

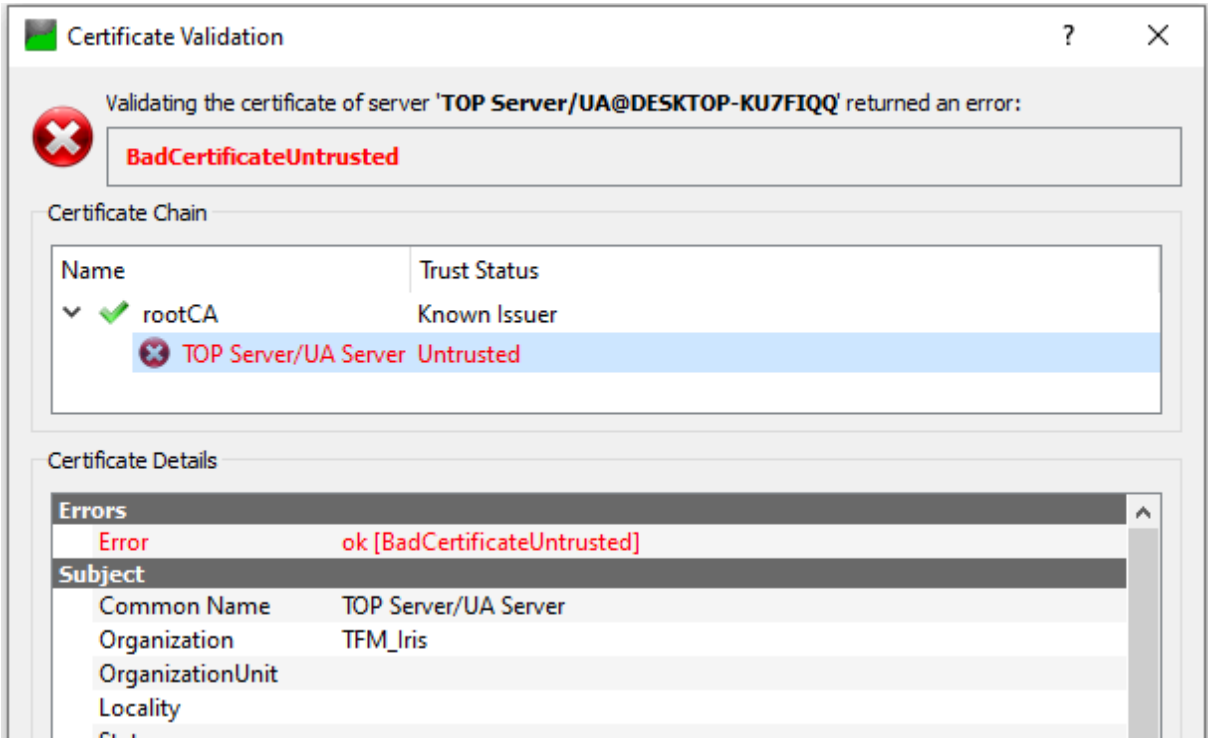


Figura 20: Certificado de TOP Server en el cliente UaExpert.

Finalmente, como se ilustra en la figura 21, tras aceptar manualmente el certificado, es posible realizar la conexión y extraer datos del servidor OPC UA.

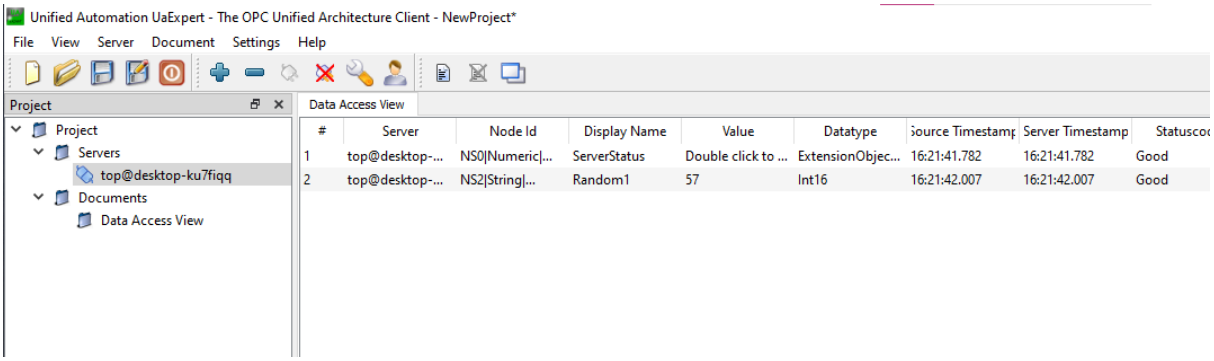


Figura 21: Captura de datos desde el servidor TOP Server

4.3. Generación de certificados OPC UA en un entorno real

Una vez confirmado que estos certificados permiten establecer una conexión segura entre cliente y servidor, se lleva a cabo un proceso para trasladar esta funcionalidad desde el entorno de simulación al laboratorio de pruebas de ámbito académico, el cual cuenta con varios PLCs Beckhoff CX5010.

Para este propósito, en primer lugar se extrae la PKI de uno de estos PLC, con el fin de asegurar que la estructura de esta es la misma que la observada en las aplicaciones

utilizadas en las simulaciones. Cuando se trabaja con certificados en entornos industriales, la uniformidad en la estructura es fundamental para asegurar la compatibilidad y la funcionalidad adecuada en diferentes sistemas y dispositivos. Se puede visualizar que la estructura se corresponde con las vistas anteriormente en la figura 22 mostrada a continuación:

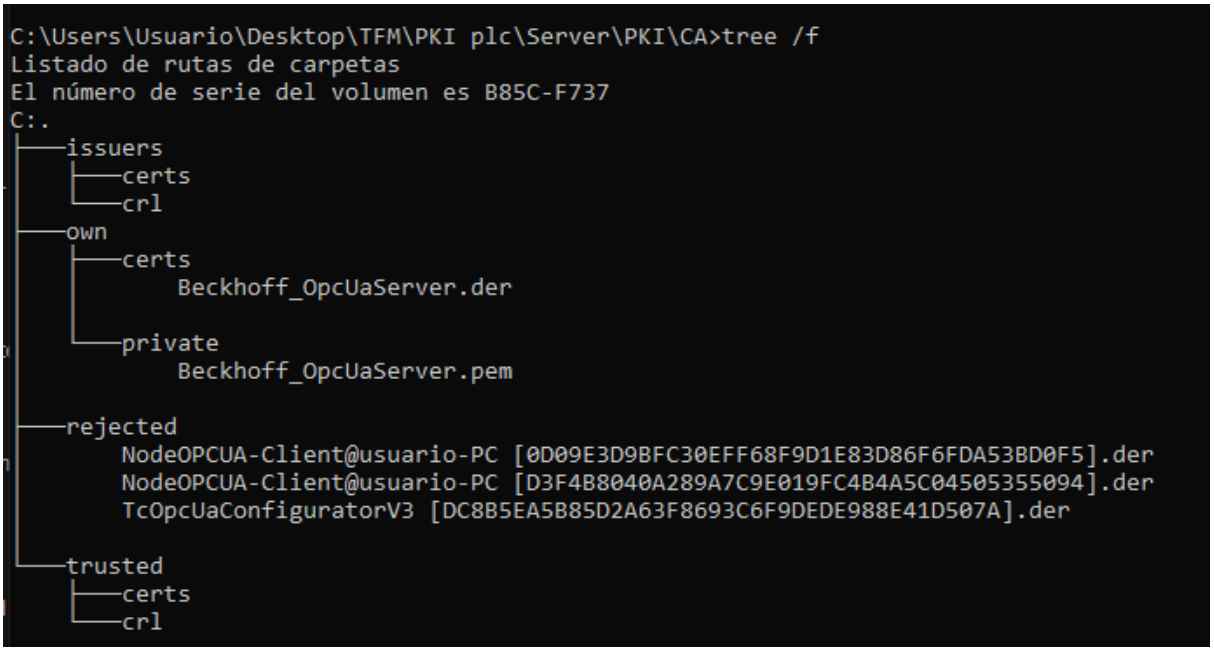


Figura 22: Estructura de la PKI de un PLC Beckhoff CX5010

Además, durante este proceso de verificación, también se confirma que los certificados extraídos del PLC presentan la misma estructura que aquellos utilizados en la simulación. Esta consistencia en la estructura de los certificados es fundamental para facilitar su correcto funcionamiento y garantizar que los certificados faciliten la interoperabilidad de varios sistemas y dispositivos dentro del entorno industrial real. La estructura de dicho certificado se especifica en la tabla 8.

Versión	V3
Número de serie	El número de serie asignado por el emisor.
Algoritmo de firma	sha256RSA
Algoritmo hash de firma	sha256
Emisor	CN = TcOpcUaServer@[nombre del equipo] OU = Unit O = Organization L = LocationName S = C = DE DC = [nombre del equipo]

Validez	20 años a partir de su generación.
Sujeto	Mismos parámetros que el emisor: CN = TcOpcUaServer@[nombre del equipo] OU = Unit O = Organization L = LocationName S = C = DE DC = [nombre del equipo]
Nombre alternativo del titular	Dirección URL= urn:BeckhoffAutomation:TcOpcUaServer Nombre DNS=[nombre del equipo]
Clave pública	RSA (4096 bits)
Uso de clave	Firma digital, Sin repudio, Cifrado de clave, Cifrado de datos, Firma de certificados
Uso extendido de clave	Autenticación del servidor (1.3.6.1.5.5.7.3.1) Autenticación del cliente (1.3.6.1.5.5.7.3.2)
Identificador de clave de entidad	Id. de clave Emisor de certificado: Dirección del directorio: CN=TcOpcUaServer@[nombre del equipo] OU=Unit O=Organization L=LocationName S="" C=DE DC=[nombre del equipo] Número de serie del certificado
Restricciones básicas	Tipo de asunto=Entidad de certificación (CA) Restricción de longitud de ruta=0

Tabla 8: Parámetros del certificado de instancia de aplicación autofirmado de un PLC Beckhoff CX5010

Dado que algunos parámetros como la longitud de la clave, el nombre alternativo del titular o los apartados del nombre distintivo del sujeto varían ligeramente respecto a los certificados anteriores, se modificó el fichero de configuración de OpenSSL para emitir un certificado adecuado a partir de una CA de confianza, incluyendo en este el nombre de la máquina en cuestión.

Una vez hecho esto, se emitieron todos los certificados necesarios para el cliente y el servidor, así como la CA raíz. Seguidamente, se instalaron los certificados

correspondientes en la PKI del servidor OPC UA, tal y como se indica en la figura 23.

```
C:\Users\Usuario\Desktop\TFM\certs PLC\CA (2)\CA>tree /f
Listado de rutas de carpetas
El número de serie del volumen es B85C-F737
C:.
├───issuers
│   ├───certs
│   │       rootCA.der
│   │
│   └───crl
│           crl.crl
│
├───own
│   ├───certs
│   │       Beckhoff_OpcUaServer.der
│   │
│   └───private
│           Beckhoff_OpcUaServer.pem
│
├───rejected
└───trusted
    ├───certs
    │       UaExpert@usuario-PCLabBeckhoff [E803A55B52FA2553577DD6B624991CFBA2FCB9BF].der
    │
    └───crl
```

Figura 23: Estructura de la PKI de un PLC tras instalar los nuevos certificados

Haciendo uso de nuevo del cliente UaExpert, se trató de establecer una conexión con el servidor, y se obtuvo el error mostrado en la figura 24.

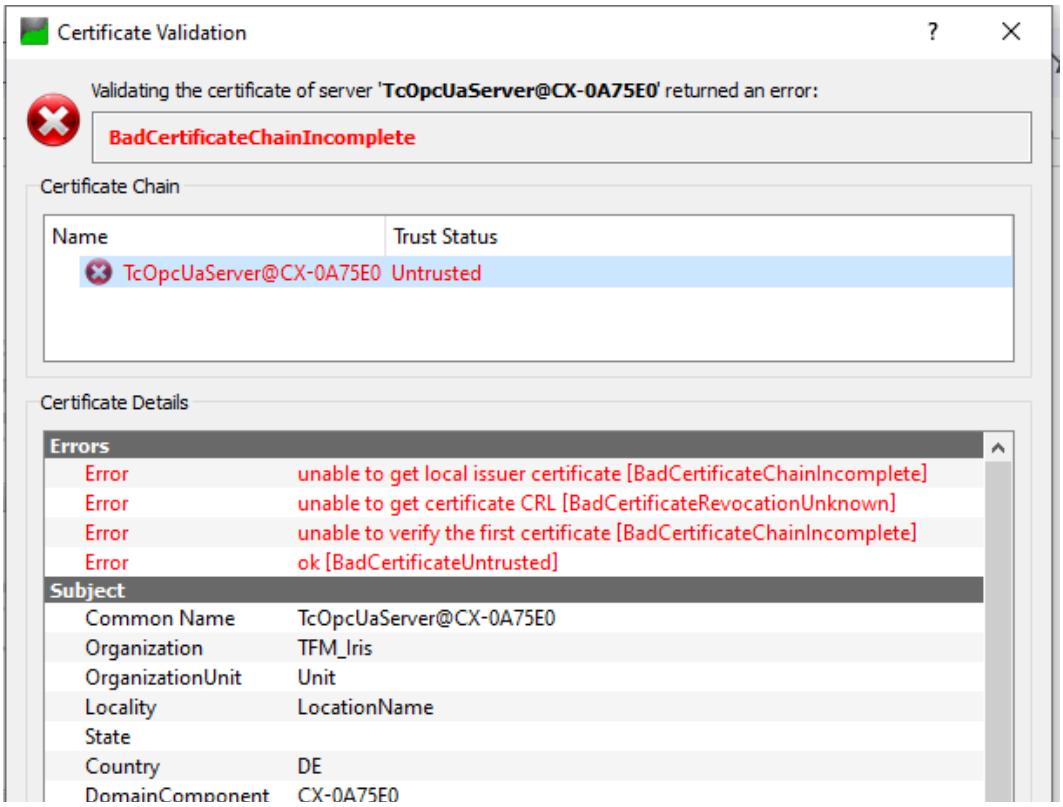


Figura 24: Error *BadCertificateChainIncomplete* emitido por el cliente OPC UA.

Este error indica que no es posible verificar la cadena de confianza y la CRL de la autoridad de confianza, y se debe a que aún no se ha instalado el certificado de la misma en la PKI del cliente, por lo que al realizar este paso, se permite aceptar el certificado manualmente como puede verse en la figura 25.

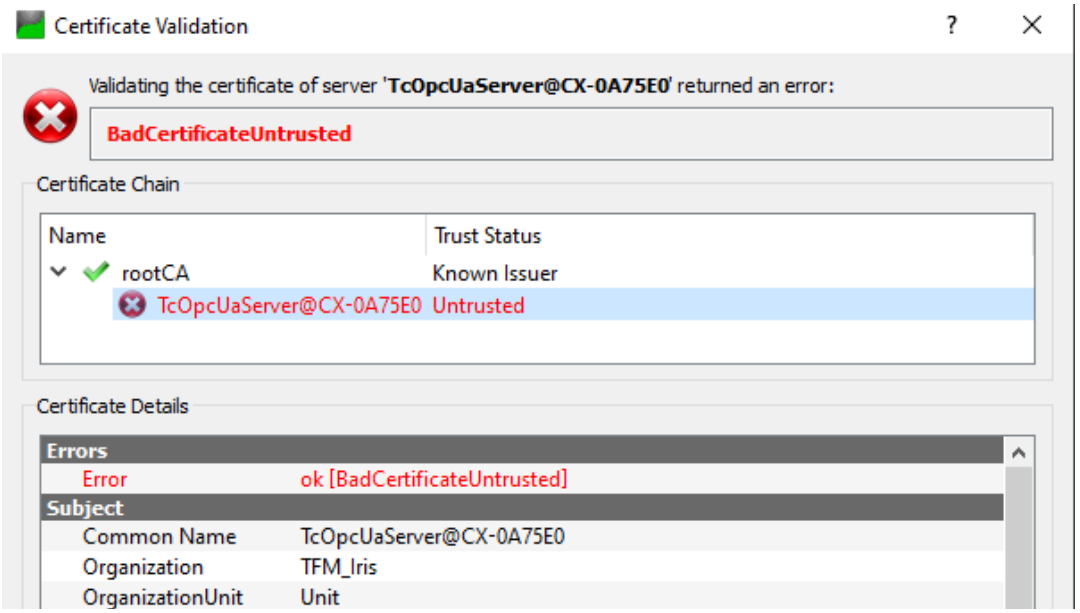


Figura 25: CA raíz reconocida como emisor de confianza en el cliente OPC UA

Sin embargo, el servidor OPC UA no requiere que el certificado sea aceptado manualmente. Al acceder a los ficheros de configuración destaca encontrar la opción *AutomaticallyTrustAllClientCertificates* habilitada como se puede ver en la figura 26. Esto indica que el servidor está configurado para confiar automáticamente en todos los certificados de cliente sin realizar ninguna validación adicional. Esto significa que el servidor aceptará y confiará en cualquier certificado presentado por un cliente sin verificar su autenticidad o validez.

```
<IsVisible>true</IsVisible>
<IsDiscoveryUrl>true</IsDiscoveryUrl>
<AutomaticallyTrustAllClientCertificates>true</AutomaticallyTrustAllClientCertificates>
▼<CertificateStore>
```

Figura 26: Configuración para aceptar automáticamente los certificados de clientes

Permitir a un servidor OPC UA que acepte automáticamente todos los certificados de los clientes puede poner en peligro la seguridad de la comunicación. Esto se debe a que la autenticación por certificado es la forma en que el cliente y el servidor establecen la confianza mutuamente. Si el servidor acepta automáticamente cualquier certificado sin validar su autenticidad, se vuelve vulnerable a ataques de suplantación de identidad o *spoofing*, donde un atacante podría presentar un certificado falso y hacerse pasar por un cliente legítimo.

Así pues, se deshabilitó esta opción, por lo que, al intentar establecer una nueva conexión, el certificado del cliente se colocó automáticamente en el directorio *rejected* de la PKI,

como se puede ver en la figura 27, por lo que en este caso debe ser trasladado al directorio *trusted* para aceptarlo como certificado confiable.

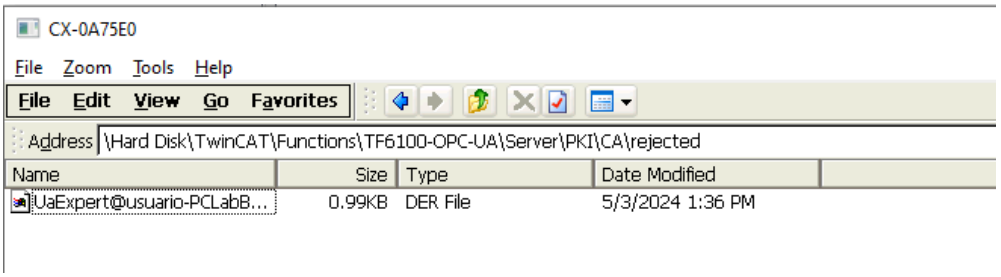


Figura 27: Certificado de cliente rechazado en un servidor OPC UA real

Finalmente, en la figura 28 se puede apreciar cómo, a través del cliente, es posible obtener datos de posición y velocidad tomados directamente del motor en movimiento con el que cuenta el PLC que está actuando como servidor OPC UA.

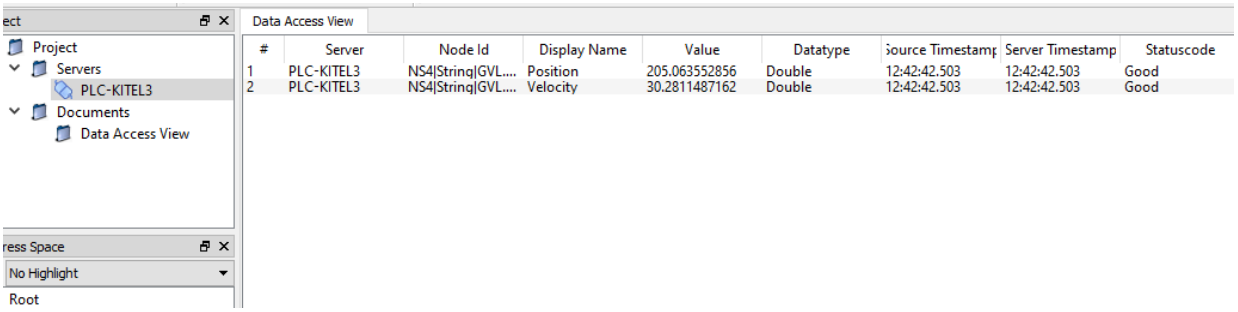


Figura 28: Captura de datos de un servidor OPC UA real

4.4. Script para automatizar la generación de certificados

En entornos que requieren una infraestructura de clave pública robusta, la generación manual de certificados y claves puede ser propensa a errores y consumir mucho tiempo. Por lo tanto y para simplificar este proceso, se ha desarrollado un script de automatización cuyo flujo de trabajo es:

- Creación de la estructura de directorios y ficheros necesarios para el correcto funcionamiento de la PKI.
- Generación de la clave privada y creación de certificado y CRL para la CA raíz.
- Generación de claves privadas y solicitudes de firma de certificado (CSR) para los certificados subyacentes.
- Firma de los certificados subyacentes con la CA raíz.
- Conversión de formatos de certificados y claves según sea necesario.

```
#!/bin/bash

# Crear la estructura de directorios
mkdir -p private certs crl
touch index.txt
echo 01 > crlnumber

# Obtener el nombre del equipo desde la variable de entorno
COMPUTER_NAME=$(echo $COMPUTERNAME) &&

# Generar el contenido de los archivos de configuración dinámicamente
cat <<EOF >rootCA.cnf
CA_DIR = .
RANDFILE = \$ENV::CA_DIR/.rnd
KEY_SIZE = 2048

[ ca ]
default_ca = CA_default

[ CA_default ]
dir = \$ENV::CA_DIR
certs = \$dir/certs
database = \$dir/index.txt
new_certs_dir = \$certs
certificate = \$dir/certs/rootCA.crt
serial = \$dir/serial
crl_dir = \$dir/crl
crlnumber = \$dir/crlnumber
crl = \$dir/crl.pem
private_key = \$dir/private/rootCA.pem
RANDFILE = \$dir/private/.rand
x509_extensions = v3_ext
default_days = 365
crl_extensions = crl_ext
default_crl_days = 30
default_md = default
preserve = no
policy = policy_match

[ req ]
default_bits       = 2048
distinguished_name = req_distinguished_name
prompt             = no

[ req_distinguished_name ]
O  = TFM_Iris
CN = rootCA

[ v3_ext ]
basicConstraints     = critical,CA:true,pathlen:0
subjectKeyIdentifier = hash
```

```
authorityKeyIdentifier=keyid:always,issuer:always
keyUsage = critical, digitalSignature, nonRepudiation, keyEncipherment,
dataEncipherment, keyCertSign, cRLSign
extendedKeyUsage = serverAuth, clientAuth

[ crl_ext ]
authorityKeyIdentifier=keyid:always, issuer:always
EOF

cat <<EOF >prosys_ca.cnf
[ req ]
default_bits       = 2048
distinguished_name = req_distinguished_name
prompt             = no

[ req_distinguished_name ]
O  = TFM_Iris
CN = SimulationServerCA

[ v3_ext ]
basicConstraints     = critical,CA:true
subjectKeyIdentifier = hash
authorityKeyIdentifier=keyid:always,issuer:always
keyUsage = critical, digitalSignature, keyCertSign, cRLSign
EOF

cat <<EOF >prosys.cnf
[req]
default_bits = 2048
distinguished_name = req_distinguished_name
prompt = no

[req_distinguished_name]
CN = SimulationServer@$COMPUTER_NAME
O = TFM_Iris
DC = $COMPUTER_NAME

[v3_ext]
basicConstraints = critical,CA:false
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always
keyUsage                                                                    =
critical,digitalSignature,nonRepudiation,keyEncipherment,dataEncipherment
extendedKeyUsage = serverAuth,clientAuth
subjectAltName       =       URI:urn:$COMPUTER_NAME:OPCUA:SimulationServer,
DNS:$COMPUTER_NAME
EOF

cat <<EOF >uaexpert.cnf
[ req ]
```

```
default_bits       = 2048
distinguished_name = req_distinguished_name
prompt             = no

[ req_distinguished_name ]
O  = TFM_Iris
CN = UaExpert@$COMPUTER_NAME

[ v3_ext ]
basicConstraints     = critical,CA:true,pathlen:0
subjectKeyIdentifier = hash
authorityKeyIdentifier=keyid:always,issuer:always
keyUsage     =     digitalSignature,     nonRepudiation,     keyEncipherment,
dataEncipherment, keyCertSign
extendedKeyUsage = serverAuth, clientAuth
subjectAltName = @alt_names

[alt_names]
URI.1 = urn:$COMPUTER_NAME:UnifiedAutomation:UaExpert
DNS.1 = $COMPUTER_NAME

[ crl_ext ]
authorityKeyIdentifier=keyid:always, issuer:always
EOF

# Crear la Clave Privada de la CA raíz
openssl genrsa -out private/rootCA.key 2048 &&

# Crear un certificado para la CA raíz
openssl  req  -x509  -new  -nodes  -key  private/rootCA.key  -days  3650  -out
certs/rootCA.crt -config rootCA.cnf -extensions v3_ext &&

# Convertir los certificados y claves a formato .der y .pem
openssl x509 -outform der -in certs/rootCA.crt -out certs/rootCA.der &&
openssl rsa -in private/rootCA.key -outform pem -out private/rootCA.pem &&

##################################################

# Crear la CRL de la CA raíz
openssl ca -gencrl -config rootCA.cnf -out crl/crl.pem  &&

# Cambiar la CRL a formato .crl (Prosys OPC)
openssl crl -inform PEM -outform DER -in crl/crl.pem -out crl/crl.crl &&

# Cambiar la CRL a formato .der (UaExpert)
openssl crl -inform PEM -outform DER -in crl/crl.pem -out crl/crl.der &&

##################################################

# Crear la Clave Privada de la CA de UaExpert
openssl genrsa -out private/uaexpert.key 2048 &&
```

```
# Crear una solicitud de firma del certificado (CSR) para la CA
openssl req -new -key private/uaexpert.key -out uaexpert.csr -config
uaexpert.cnf &&

# Firmar el certificado con la CA raíz
openssl x509 -req -in uaexpert.csr -CA certs/rootCA.crt -CAkey
private/rootCA.key -CAcreateserial -out certs/uaexpert.crt -days 3650
-sha256 -extfile uaexpert.cnf -extensions v3_ext &&

# Convertir los certificados y claves a formato .der y .pem
openssl x509 -outform der -in certs/uaexpert.crt -out certs/uaexpert.der
&&
openssl rsa -in private/uaexpert.key -outform pem -out
private/uaexpert_key.pem &&

#####################################################

# Crear la Clave Privada de la CA de Prosys OPC
openssl genrsa -out private/prosysCA.key 2048 &&

# Crear una solicitud de firma del certificado (CSR) para la CA
openssl req -new -key private/prosysCA.key -out prosysCA.csr -config
prosys_ca.cnf &&

# Firmar el certificado con la CA raíz
openssl x509 -req -in prosysCA.csr -CA certs/rootCA.crt -CAkey
private/rootCA.key -CAcreateserial -out certs/prosysCA.crt -days 3650
-sha256 -extfile prosys_ca.cnf -extensions v3_ext &&

# Convertir los certificados y claves a formato .der y .pem
openssl x509 -outform der -in certs/prosysCA.crt -out
certs/SimulationServerCA.der &&
openssl rsa -in private/prosysCA.key -outform pem -out
private/SimulationServerCA.pem &&

#####################################################

# Generar la clave privada del servidor Prosys OPC
openssl genrsa -out private/prosys.key 2048 &&

# Generar la CSR
openssl req -new -key private/prosys.key -out prosys.csr -config
prosys.cnf &&

# Firmar el certificado con la CA raíz
openssl x509 -req -in prosys.csr -CA certs/rootCA.crt -CAkey
private/rootCA.key -CAcreateserial -out certs/prosys.crt -days 3650
-sha256 -extfile prosys.cnf -extensions v3_ext &&

# Convertir los certificados y claves a formato .der y .pem
openssl x509 -outform der -in certs/prosys.crt -out
certs/SimulationServer@$COMPUTER_NAME\_2048.der &&
```

```
openssl rsa -in private/prosys.key -outform pem -out
private/SimulationServer@$COMPUTER_NAME\_2048.pem &&

##################################################

# Limpiar archivos temporales
rm uaexpert.csr prosysCA.csr prosys.csr
```

Figura 29: Código del script para la generación automatizada de certificados

El script se puede utilizar ejecutando un simple comando en la consola *Git Bash*. Además, es posible personalizarlo mediante un IDE para, por ejemplo, omitir la creación de la estructura de directorios o la CA raíz en el caso de haber ejecutado estos pasos anteriormente, o añadir otros formatos a los diferentes ficheros en función de los requerimientos específicos de cada entorno.

5. Presupuesto

En este capítulo se presentan las estimaciones económicas del Trabajo Fin de Máster. Esta estimación se divide en dos secciones: costos relacionados con materiales y costos relacionados con el desarrollo, es decir. los recursos humanos involucrados.

5.1. Costes tecnológicos

La tabla 9 presenta los costes que corresponden a los materiales y/o equipamiento necesario para realizar un despliegue ficticio de este proyecto.

Tipos	Descripción	Producto	Coste unitario
PLC (Programmable Logic Controller)	Compra de autómatas para el despliegue del proyecto en fase de implementación.	PLC Beckhoff CX5010	2.712 € aprox.
Certificados emitidos por Let’s encrypt	Generación de certificados a partir de la autoridad certificadora de confianza Let’s encrypt	Certificado	0 €

Tabla 9: Costes tecnológicos

5.2. Costes de desarrollo

La tabla 10 presenta los costes correspondientes al desarrollo del proyecto y recursos humanos necesarios para el mismo, así como el tiempo empleado en cada uno de los periodos que este comprende.

Tipo de actividad	Cantidad	Coste unitario	Coste total
Estudio del estándar	15 horas	10€/h	150€
Diseño de la PKI	10 horas	10€/h	100€
Configuración del entorno de simulación	5 horas	10€/h	50€
Despliegue de la PKI en el entorno de simulación	12 horas	10€/h	120€
Pruebas de funcionamiento en el entorno de simulación	10 horas	8€/h	80€
Despliegue de la PKI en el entorno de laboratorio	3 horas	12€/h	36€
Documentación del proyecto	25 horas	9€/h	225€
Total:	80 horas		761€

Tabla 10: Costes de desarrollo

6. Conclusiones y líneas futuras

A lo largo del desarrollo de este proyecto, se ha logrado un avance significativo en materia de seguridad respecto al estándar OPC UA, al poder reemplazar los certificados autofirmados del mismo con certificados emitidos por una Autoridad de Certificación (CA) propia. Este hito se ha alcanzado tanto en entornos de servidores simulados como reales permitiendo confirmar su funcionamiento en ambas configuraciones.

En los sistemas basados en OPC UA, el uso de certificados proporcionados por una Autoridad de Certificación (CA) interna supone una mejora considerable en lo que respecta tanto a la seguridad como a la autenticación. Sin embargo, se debe tener en cuenta que su fiabilidad es cuestionable debido a su carácter. Los certificados expedidos por esta CA local satisfacen todos los requisitos de autenticación necesarios para el entorno de pruebas, pero no cuentan con ninguna validación de terceros de confianza.

Aunque los certificados autofirmados son menos seguros, no son totalmente inútiles en el contexto de la seguridad OPC UA. De una manera más amplia y más alineada con la industria, la idea es alojar servidores OPC UA bajo un único dominio mediante el cual estos servidores pueden ser asegurados usando certificados de CAs de terceros de confianza como Let's Encrypt. Estas organizaciones están respaldadas por sólidas políticas de seguridad y, por ende, afirman la autenticidad de los certificados y mantienen la veracidad en la comunicación. Esta vía aportará una forma asequible y fácilmente personalizable de gestionar los certificados, así como confianza e interoperabilidad con otros sistemas.

Teniendo en cuenta las conclusiones, se contempla combinar la Criptografía Post-Cuántica (PQC) en OPC UA [16] en un trabajo posterior. Como tal, se consideran dos enfoques: híbrido y totalmente post-cuántico. Ambos tienen como objetivo garantizar la seguridad de las comunicaciones OPC UA, ya sea mediante combinaciones de algoritmos clásicos y post-cuánticos o mediante la conversión a post-cuántica únicamente.

En el enfoque híbrido, la confidencialidad, autenticidad e integridad de los datos se garantizan mediante una combinación híbrida de esquemas criptográficos clásicos y post-cuánticos. Se emplean firmas híbridas para la autenticación mutua entre el cliente y el servidor, mientras que distintos algoritmos de cifrado combinados con MAC (Códigos de Autenticación de Mensajes) ayudan a establecer canales seguros. Se emplea un certificado X.509 con distintas extensiones para permitir la inclusión de claves públicas junto con firmas digitales post-cuánticas.

Alternativamente, en el enfoque post-cuántico más puro, los esquemas post-cuánticos sustituyen por completo a los algoritmos criptográficos clásicos. Se define una nueva política de seguridad para OPC UA en la que las claves públicas y las firmas digitales post-cuánticas se incluyen en los certificados del servidor. El intercambio de claves se realiza con un esquema post-cuántico KEM (Key Encapsulation Mechanism), mientras que la autenticación se realiza con certificados post-cuánticos. Una vez establecida la conexión segura, se crean los componentes necesarios para su uso en sesiones de comunicación posteriores.

7. Conclusions and future works

Throughout the development of this project, a significant advance in security has been achieved with respect to the OPC UA standard by being able to replace its self-signed certificates with certificates issued by a Certification Authority (CA) of its own. This milestone has been achieved in both simulated and real server environments allowing to confirm its operation in both configurations.

In OPC UA-based systems, the use of certificates provided by an internal Certification Authority (CA) is a considerable improvement in terms of both security and authentication. However, it should be noted that their reliability is questionable due to their nature. The certificates issued by this local CA satisfy all the authentication requirements necessary for the test environment, but do not have any trusted third-party validation.

Although self-signed certificates are less secure, they are not totally useless in the context of OPC UA security. In a broader and more industry-aligned way, the idea is to host OPC UA servers under a single domain whereby these servers can be secured using certificates from trusted third-party CAs such as Let's Encrypt. These organizations are backed by strong security policies and thus affirm the authenticity of the certificates and maintain veracity in communication. This approach will provide an affordable and easily customizable way to manage certificates, as well as trust and interoperability with other systems.

Given the findings, it is contemplated to combine Post-Quantum Cryptography (PQC) in OPC UA [16] in further work. As such, two approaches are considered: hybrid and fully post-quantum. Both aim to ensure the security of OPC UA communications, either by combinations of classical and post-quantum algorithms or by conversion to post-quantum only.

In the hybrid approach, confidentiality, authenticity and data integrity are guaranteed by a hybrid combination of classical and post-quantum cryptographic schemes. Hybrid signatures are used for mutual authentication between client and server, while different encryption algorithms combined with MAC (Message Authentication Codes) help establish secure channels. An X.509 certificate with different extensions is used to allow the inclusion of public keys along with post-quantum digital signatures.

Alternatively, in the purest post-quantum approach, post-quantum schemes completely replace classical cryptographic algorithms. A new security policy is defined for OPC UA in which public keys and post-quantum digital signatures are included in the server certificates. Key exchange is performed with a post-quantum KEM (Key Encapsulation Mechanism) scheme, while authentication is performed with post-quantum certificates. Once the secure connection is established, the necessary components are created for use in subsequent communication sessions.

8. Bibliografía

[1] Wang, Z. et al., "A Survey on Programmable Logic Controller Vulnerabilities, Attacks, Detections, and Forensics," Processes, vol. 11, p. 918, Mar. 2023. https://doi.org/10.3390/pr11030918

[2] XPoint. “Ciberdelincuentes Iraníes Explotan PLC en Ataque Contra la Autoridad del Agua en EE. UU.” Nov. 29, 2023. [Online]. Available: Ciberdelincuentes Iraníes Explotan PLC en Ataque Contra la Autoridad del Agua en EE. UU. - XPoint Cybersecurity

[3] INCIBE. "Evil PLC, la amenaza sigilosa" Nov. 24, 2022. [Online]. Available: Evil PLC, la amenaza sigilosa | INCIBE-CERT

[4] M. Damm, S.-H. Leitner, y W. Mahnke, "OPC Unified Architecture", en OPC Unified Architecture, Springer, 2009. https://doi.org/10.1007/978-3-540-68899-0

[5] Becolve Digital, "OPC: de sus orígenes a OPC UA", Sep. 20, 2019. [Online]. Available: OPC: de sus orígenes a OPC UA

[6] INCIBE (INCIBE), "OPC UA, equilibrio entre ciberseguridad y rendimiento," Jan. 11, 2024. [Online]. Available: OPC UA, equilibrio entre ciberseguridad y rendimiento | INCIBE-CERT

[7] Amazon Web Services, "¿Qué es el Internet de las cosas (IoT)?" [Online]. Available: ¿Qué es IoT? - Explicación del Internet de las cosas - AWS

[8] “Beckhoff TwinCAT” [Online]. Available: BECKHOFF TwinCAT

[9] Satoshi, "5 razones por las que elegir OPC UA", Jan. 5, 2017. [Online]. Available: OPC UA, 5 razones para elegirlo como habilitador de la Industria 4.0

[10] U. Pohlmann and Prof. Dr.-Ing. A. Sikora, "Practical Security Guidelines for Building OPC UA Applications," The Industrial Ethernet Book. [Online]. Available: Practical Security Guidelines for Building OPC UA Applications.

[11] OPC Foundation. "Part 2: Security Model. OPC Unified Architecture Specification”. [Online]. Available: UA Part 2: Security - 4 OPC UA security architecture

[12] INCIBE, "Estandarización y seguridad en el protocolo OPC UA," Nov. 15, 2018. [Online]. Available: Estandarización y seguridad en el protocolo OPC UA | INCIBE-CERT

[13] INCIBE, "OPC UA, equilibrio entre ciberseguridad y rendimiento," Jan. 11, 2024. [Online]. Available: OPC UA, equilibrio entre ciberseguridad y rendimiento | INCIBE-CERT

[14] OPC Foundation. "OPC Unified Architecture Specification Part 6: Mappings". [Online]. Available: UA Part 6: Mappings - 6.2.2 Application Instance Certificate.

[15] Unified Automation. “UA SDK C++ Reference Guide: Level 2 Client SDK Security". [Online]. Available: C++ Based OPC UA Client/Server SDK: Security.

[16] L. Chen, N. Li, K. Liang, and S. Schneider, Eds., “Computer Security – ESORICS 2020, ser. Lecture Notes in Computer Science”, vol. 12346. Cham, Switzerland: Springer International Publishing, 2020, pp. 301–302. https://doi.org/10.1007/978-3-030-59013-0.

[17] Github. https://github.com/

Apéndice 1. Script para la creación automática de certificados

Repositorio en GitHub [17]: https://github.com/alu0101205953/script_OPC_UA.git