

Securing communication on the field: Protecting geo-distributed computing in an untrusted environment

Olivier Gilles¹, David Faura¹ and Daniel Gracia Pérez¹

¹Thales Research & Technology, 1 avenue Augustin Fresnel, 91767 Palaiseau, France

Abstract

As the number of geo-distributed connected sensors and the need to perform complex functionality increase in Industrial IoT-based systems, so does the transferred data volume. Mainframes and even edge computing show their limits when facing this intensive computing. To answer this challenge, an emerging trend, called Industrial Continuous Computing (ICC), advocates for truly distributed computation in the network, from the Smart Peripheral Devices (SPD) to the end-server or actuators. This, in turn, raises the issue of communication security. In this context, protecting communications in a dynamic architecture is challenging, as attackers may have physical access to a legitimate device. Surface attack on the communication includes confidentiality and integrity of the data, which in turn requires integrity of the software stack manipulating them. Breaches on these properties are likely to lead to secret exposure or sabotage.

We present a novel approach based on open source software and secure hardware aiming to ensure end-to-end security for communication as well as securing devices authentication, enabling the confidentiality and integrity of data exchanges between the different SPDs and the end-server. Our approach uses publish-subscribe communication protocols (i.e. many-to-many) to build scalable and dynamic computing architecture. Amongst them, OPC UA PubSub provides security and interoperability to the ICC, allowing end-to-end encryption. This security however relies on securely embedded secrets on the node. We further secure authentication against specific threats to ICC, by using a Trusted Platform Module (TPM) as a secure element to protect the secrets embedded on devices, and relying on secure boot to protect this secure element against attackers. By doing so, we were able to set up fully secure a geo-distributed industrial computing infrastructure dedicated to the monitoring of railway facilities, connecting sensors, edge devices and data server in order to perform predictive maintenance.

1. Introduction


Industrial systems are constituted of four kind of nodes: sensors, actuators, controllers and computers. The most common architecture is a system where a supervisor component implementing both control and computing functions is connected to a set of slave devices, which can be either sensors or actuators. With the important increase of data and computation needs, supervisor components become unable to provide enough computation power to perform the needed operations.

C&ESAR 2023 by DGA, November 21–23, 2023, Rennes, France

✉ olivier.gilles@thalesgroup.com (O. Gilles); david.faura@thalesgroup.com (D. Faura); daniel.gracia-perez@thalesgroup.com (D. Gracia Pérez)

🆔 0000-0002-3776-2071 (O. Gilles); 0009-0004-9416-8855 (D. Faura); 0000-0002-5364-8244 (D. Gracia Pérez)

© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

In traditional Industrial Control Systems such as SCADA systems, none of these components are connected to open networks, and most of them rely on field buses, using wired communications and proprietary protocols (e.g. PROFINET, EtherCAT).

In Industrial Internet of Things (IIoT), as opposed to the more generic IoT context, the things (devices, but also supervisors) are rarely directly connected to open networks such as Internet. Indeed, the compromise of any device of the system could lead to catastrophic results [1, 2]. In the other hand, computing nodes necessary to perform computing-intensive functions, but also access to third-party sensors, monitoring nodes or even actuators demand data to cross open networks. Indeed, in order to ensure data security during this odyssey, one must ensure (1) the path safety and (2) whether it actually goes to its intended destination.

While these issues are well-treated in traditional IT systems, IIoT differs from those in that nodes can typically be physically accessed by users, or even by any bystanders, including potential attackers. This leads to a whole new class of possible attacks where the attacker must be assumed to have total control on memory (through memory dumping or fault injection [3]), and that we must protect the system against.

In this context that requires *security* for both data and software, we focus our analysis on existing *open protocols*. We present in this paper a secure system architecture based on such open protocols, namely MQTT, OPC UA PubSub, and on a hardware security element, the TPM, and how we can articulate them to protect the nodes against attackers benefiting from physical access to the device and reasonable hacking skills.

The remainder of this paper is organized as follows: Section 2 introduces key concepts needed for distributed computing in critical systems and their implications in terms of security needs on the system's equipments, as well as the industry challenge of predictive maintenance; Section 3 describes the OPC UA protocol, which is commonly used in industrial systems and candidates for IIoT, as well as our contribution, a solution to secure authentication by ensuring secret confidentiality and integrity in the SPD; in Section 4, we describe the industry challenge of predictive maintenance, as well as the prototype relying on our solution to address its security and connectivity issues; finally, we conclude about further enhancement in the conclusion.

2. Distributing Computation in Critical Systems

In recent years, the rising popularity of smart systems (Smart City, Autonomous System, Industry 4.0) and the proliferation of remote connected sensors and IoTs has generated massive and varied amounts of data, greatly increasing the processing and storage capacities required to extract valuable information. This evolution of usage has shown the limitations of a remote centralized processing architecture, which typically increases the complexity and limits the monitoring and interaction with smart systems in real time [4].

2.1. Geo-distributed industrial architectures

The Compute Continuum is an emerging distributed architecture, which brings back heterogeneous capabilities such as processing or storage close to the IoT devices. In that architecture, the underlying infrastructure is composed of a cloud datacenter connected through a set of

heterogeneous computing networks to a vast number of geo-distributed computing nodes forming the Edge domain.

The interest of this architecture is the benefits to extract in real time valuable information from the IoT devices, to reduce reaction times of the actuators and to enable the reuse of IT-domain components. Furthermore, this infrastructure can orchestrate and monitor the system's functions within the computing continuum. However, it increases the overall cyber security issues because the geo-distributed computing nodes interact with an uncontrolled environment, increasing the possibility of attacks on physical nodes and communication medium.

These issues are especially crippling when the architecture is applied to OT systems, within an Industrial Compute Continuum (ICC, see Figure 1), which have drastic safety requirements, including real-time constraints. Regarding security in particular, it implies (1) the ability to process data and perform analyses as close as possible to the data sources while ensuring data privacy and data security, (2) efficient isolation of critical and non-critical functions collocated on the same computing nodes and (3) secure communications along all the computing continuum, including authentication of all actors (nodes and functions).

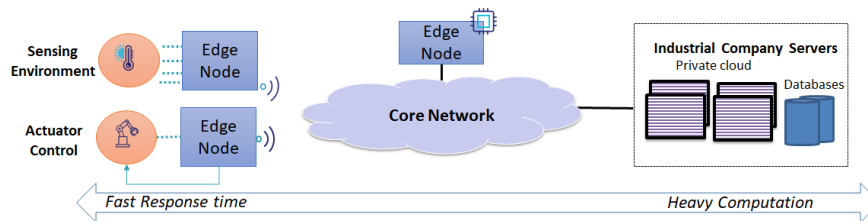


Figure 1: Industrial Compute Continuum

2.2. Designing the Industrial Compute Continuum

2.2.1. ICC communication topology

While many topologies may be used to implement the ICC, we advocate for publish-subscribe (or PubSub) communication pattern. In this pattern, publishers and subscribers are connected to an agent commonly known as *broker*, which may be constituted of a single component or distributed. The broker receives messages from publishers and dispatches them to subscribers subscribed to the topic of the messages. In other words, publishers send messages to potentially many subscribers. This loosely coupling facilitates network scalability and versatility. Recognized protocols enforcing PubSub mechanisms as MQTT¹ or AMQP² are widely used in the industry.

In a network topology like the IIoT, where numerous (typically several hundreds) captors and actuators need to communicate with supervisors, a classical client-server approach requires a dense connection mesh to gather and distribute information. In this perspective, the PubSub mechanism is interesting because the supervisors only need to send one message to communicate information to all the remote devices, thus limiting the bandwidth usage on the source network.

¹<http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/errata01/os/mqtt-v3.1.1-errata01-os-complete.html>

²<http://docs.oasis-open.org/amqp/core/v1.0/amqp-core-complete-v1.0.pdf>

In the ICC context, where the information goes through successive computing nodes, the PubSub topology also allows factorizing the global computation power used.

2.2.2. IIoT communication security

Properly signed and encrypted messages are hard to modify and decrypt. Yet in the long term no system is immune to compromise, even for mature, market-proven systems³. This is particularly true for long-life *safe* systems (for instance automatic train controllers), which are rarely updated: such systems must be thoroughly tested for systematic bugs before being deployed [5, 6]. Since new vulnerabilities can be revealed after the device deployment, a defense-in-depth strategy must be adopted to ensure persistent security throughout the whole system's lifecycle.

If the keys used to sign and encrypt messages of a device get compromised, the attacker would be able to forge messages as if they were produced by this device. Furthermore, it would allow a rogue client to obtain the same access as the device, leading to potential leaks of industrial secrets, or offering leverage to physical sabotage actions to the attacker.

IIoT often relies on a gateway in order to assert security. IIoTs such as remote sensors and a gateway are an important segment of an ICC. As gateways (1) can be hierarchical and (2) can perform edge computing, they are sound candidates to build the ICC backbone.

2.2.3. Implementing a gateway function

Because of the limited computing (or available) power of end-devices on some use cases, it is not always possible to secure all communications between all devices. One typical case is when end-devices such as remote sensors must run on limited batteries for long periods, and thus reduce their computation to the bare minimum.

In this case, devices that must communicate without encryption must belong to the same isolated private network. To overcome this limitation and enable communications with devices outside of this network, a gateway may be used to bridge communications to other isolated networks. The gateway function ensures the confidentiality and the integrity of the data it relays, from the edge (OT-to-IT border) to the subscriber system. It can use secure and interoperable protocols to communicate these data to/from other gateways or other remote controllers, even through public infrastructures encrypting/authenticating the communications when necessary. The gateway function can be distributed amongst multiple physical devices.

However, this approach implies that compromising a device implementing the gateway function may give access to the whole subnetwork. In this case, it is possible to forge messages on behalf of the compromised gateway. Such attacks may lead to catastrophic results.

In order to ensure security of the industrial system, we present in this document a hardware-based approach to secure the gateway communications, and give in Section 4.2 an example of its usage in the predictive maintenance for the railway industry.

³<https://us-cert.cisa.gov/ics/advisories/icsa-19-274-01>

2.2.4. End-to-End Encryption

It is not always an option to rely on a secure gateway for very sensitive systems, for example when subject to state security regulations, or target of industrial spying. In these cases, confidentiality must involve only the ends of the communication, without third party. As such, end-to-end encryption is an important need for IIoT. Only the final receiver of a message should be able to access to its clear text data. It encompasses the capability to cross networks through as many gateways as needed, while the data keeps its confidentiality and integrity, both being guaranteed by cryptographic means. This latter feature is important, as devices in IIoT are usually embedded into multiple layers of subnetworks.

Using end-to-end encryption is only possible when using a data transmission protocol that implements it, and it is usually not applicable for legacy systems.

2.2.5. Software integrity

Software integrity of the communicating devices is a mandatory foundation for security. This applies to both system (firmware, kernel, drivers) and application software stacks.

Devices integrity must be ensured by the device themselves through Secure Boot mechanisms. Secure Boot ensures that the device will only boot the device constructor software. Furthermore, some Secure Boot implementations can provide decryption functionalities ensuring that the protected software remains confidential while at rest, i.e. the protected software will be encrypted in the storage memory like MMC, flash, etc.

3. Securing geo-distributed systems communications

In order to design trustable geo-distributed systems in adversary condition, it is mandatory to ensure communication security across all the system lifetime. Two critical needs to achieve that result are (1) secure communication and (2) secure authentication. In this section, we show how we can use OPC UA to met these needs in a protected environment, and how we can leverage on hardware-based measures (Secure Boot and TPM) to extend security to untrusted environment.

3.1. OPC UA PubSub

OPC UA is a standard [7] for data exchange in industrial communications. It provides safe and secure means to connect supervision systems (SCADA) with programmable logic controller (PLC), actuators, and sensors. Its publisher-subscriber variant (OPC UA PubSub) is a good candidate for implementing IIoTs-based systems, including ICC, as it is platform-independent, offers great inter-operability between standard IT and OT protocols and has been through thorough security assessment [8, 9].

OPC UA PubSub is structured in two layers enabling the inclusion of new underlying protocols as technology evolves: the Transport layer and the Message layer. It currently supports a set of well-established communication protocols in order to convey its payload between two clients, ranging from raw Ethernet to MQTT/AMQP. The most light-weight of

these protocols (e.g. Ethernet or UDP) are used to implement field buses, and typically broadcast messages on a local loop or take advantage of a physical switch to emulate said broker.

The two latter transport protocols are in fact using existing publish-subscribe protocols, which are only used for their transport capabilities. These are better suited to Cloud Integration as they are supported by the main cloud providers. MQTT lower overhead makes it more adapted to numerous IIoT scenarios.

3.1.1. OPC UA PubSub End-to-End Encryption

Messages are composed of headers and a payload. When using encryption, only the payload, which bears data values, is encrypted. The whole Message is then signed (including sequence number as to prevent replay attacks) and the signature is appended at the end of the message. In this way, regardless of the Transport layer, end-to-end encryption is achieved.

OPC UA PubSub supports three *security modes*: encryption and signature, signature only, or none of them. There are multiple *levels* of encryption which are called *security policies*. They describe which algorithms are used, with which key lengths, signature scheme, ...

For now, PubSub encryption always uses block ciphers in counter mode (AES in CTR mode [10]), and the signature algorithm is a message authentication code (HMAC) based on the SHA-256 hash algorithm. As these algorithms need a secret to work with (i.e. a symmetric key), publishers and subscribers must possess the same secrets to securely exchange messages. No other node in the communication path (including the broker or a potential man-in-the-middle attacker) must to know any of these secrets, ensuring end-to-end encryption.

3.1.2. OPC UA PubSub key distribution

The solution defined in the specification (see Section 5.4.3 of [7] Part 14) is to use a server entity named Security Key Services (SKS). This OPC UA server is in charge of the authentication of the agents of the network, and the distribution of the security keys.

As a consequence, publishers and subscribers must connect to the server using the classic OPC UA protocol to fetch the security keys. They must do so in a secure way, so that the security group keys are transferred in a confidential way on the (maybe) public network.

In classic OPC UA, the authentication of the client is handled through a Public Key Infrastructure (PKI) with Certificate Authorities (CA). It uses the X.509 [11] public key certificate and Certificate Revocation List (CRL) standard. This implies that the client embeds a key pair to prove its identity. This pair is composed of a so-called public key that is signed by a certificate authority (this signed key is also called a certificate), and a private key which must stay secret at all times. The client and server shall also embed the trust chain to be able to check respectively the client and server certificates (see [12] for details).

Figure 2 describes the two main OPC UA PubSub phases, where communications using OPC UA Client/Server paradigm (i.e. authentication and key distribution) are colored in red, while communications using the PubSub paradigm (i.e. encrypted messaging) are colored in blue.

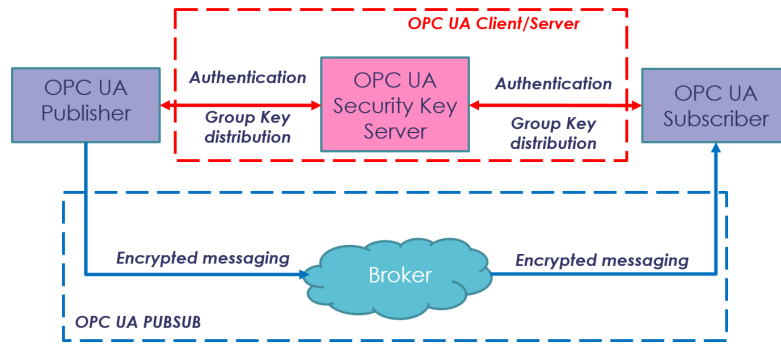


Figure 2: OPC UA PubSub

3.1.3. Publisher authentication

The OPC UA client-server protocol has been analyzed for security vulnerabilities in the past (for instance, see [8, 9]) and is considered secure. Moreover, as of today, no vulnerability has been identified in the OPC UA PubSub protocol, which security relies on modern cryptographic means. However, as mentioned in Section 3.1.1, all agents participating in a given security group share the same encryption and signature keys. This holds true whether the clients are subscribers or publishers, making a malicious client holding valid group keys able to compromise a whole group, as it could either feed some forged data (from publisher, attack on data integrity), or read secret data (from subscriber, attack on data confidentiality). Hence, the group keys handling and their distribution is a vulnerable process, and protecting the group keys is thus critical to ensure the group security. As the group keys are securely exchanged on a secure OPC UA client-server connection, their confidentiality is based on two factors: (1) their storage in PubSub agents and (2) the authenticity of the client certificate used to fetch the group keys from the SKS server.

In OPC UA PubSub specification, agents connect to the SKS for authentication using the OPC UA client/server protocol. In order to assert its authenticity, the OPC UA client will (1) encrypt a message containing its nonce with the SKS's private key and (2) send its encrypted and signed certificate to the SKS. On its side, the SKS will (1) check the validity of the certificate through its Public Key Infrastructure (PKI), (2) decrypt the client's nonce and (3) encrypts its own nonce with the client's public key, and sign it with its private key. Upon reception of this response, the client will check its validity through the SKS's public key, and decrypt it with its own private key. If all these operations succeeded, double authentication is performed, and both are deemed as legitimate. Exchanged nonces are then used to generate temporary *session keys*, which will be used in further exchanges until a key renewal is needed.

Now that the client has established a secure way to communicate with the server, it can securely fetch the PubSub group keys. As the server has authenticated the client, it can either authorize or deny the client request to fetch the group keys, depending on whether or not the client is part of the security group. This process relies on the fact that the client private key cannot be recovered, so that attackers cannot impersonate legitimate clients.

3.2. Securing secrets on the field

Security measures must be taken all along the life cycle of the client device, as many opportunities may be used to steal the client's private key. For instance, if the private key is provisioned by an external actor (distributed over the network, brought physically to the device on an external storage device, ...), it could be compromised even before its provisioning. Even if the keys are provisioned securely, it is still possible to compromise the device's private key. For instance, if the private key is stored physically unprotected on the device, an attacker could either compromise it at rest or at runtime.

3.2.1. Asserting Authenticity with a Secure Cryptoprocessor

The Trusted Platform Module (TPM) is a cryptographic standard [13] for cryptoprocessor providing services such as random number generation, generation and storage of cryptographic keys, encryption and decryption of data. The TPM standard is currently in its second iteration and it's frequently referred as TPM2. The benefit of using a TPM is that it is in charge of protecting the secrets and of doing the encryption/signature process without loading the secrets to the system's memory. While doing so, the secrets never get out of the TPM hardware module. Moreover, some TPM implementations can be tamper resistant to physical intrusion and reset themselves in such event, protecting the secrets.

TPMs are usually available as discrete hardware components integrated into the target device. Other times, TPMs are also available as software components [14] executed in a Trusted Executed Environment (TEE). In all cases, the communication between the device and the TPM is standardized by the cryptographic standard.

3.2.2. TPM and OPC UA

In the OPC UA client-server protocol, a TPM can be used to securely store the asymmetric keys and realize signature operations (robust authentication in Section 3.1.3) and decryption (to obtain the nonce). This requires two additional configuration steps: (1) the TPM is first provisioned with an asymmetric key pair, i.e. the TPM generates an asymmetric key pair for which the public key part can be retrieved but the private key part cannot, and (2) the public key part of the created key pair is retrieved from the TPM and signed by a trusted certificate authority.

The signed public key part is the certificate that uniquely authenticates an OPC UA client or server in the network architecture. An OPC UA PubSub agent that has a configured TPM is now able to recover the security group keys securely, and the private key used in the OPC UA client-server protocol cannot be extracted from the device.

3.2.3. Secure TPM configuration at boot time

In order to ensure the confidentiality of the TPM secrets it must be assured that the secrets in the TPM can only be used in a trusted environment to avoid impersonation of our TPM secured device. To achieve so, the TPM secrets are protected by another secret only known to trusted environments. In our case, this can be ensured by cryptographically coupling the Secure

Boot mechanism of the device SoC and the TPM. The TPM standard provides the Platform Configuration Registers (PCRs) mechanisms for this purpose. The same mechanism can be used to implement a Measured Boot (always on top of the SoC Secured Boot mechanism).

3.2.4. Improving authentication security with trusted secrets

As illustrated by the previous example, the TPM provides the following properties: (1) the private key is not readable on non-volatile storage (hard-disk, ROM, ...), (2) the private key is not loaded in live memory of the device and (3) no human operator ever access or manipulates the private key.

Additionally to the mere security insurance increase, using a TPM significantly reduces the attack surface of the system. Without a TPM (or other kinds of Hardware Security Modules, HSM), the private key appears in static and/or live memory. It is also generated and provisioned by different actors, human and/or automated. At any point the private key may be the target of an attacker. In our solution, the only point of failure is the TPM, and thus the security assessment is much easier and less error-prone.

Having a single point of protected storage also helps monitoring the security of all the communications. This facilitates the identification of the potential leak of the key and its revocation, keeping safe the other agents of the network more efficiently.

3.2.5. Limits and Remaining Vulnerabilities

As it is an additional component, a TPM module must be integrated into the target board. Although technically not challenging since the TPM suppliers generally make this integration as easy as possible, it is likely to lead to a re-certification in critical domains (e.g. avionics).

A second possible issue is the limited computational power of the TPM. In the example of the OPC UA client-server connection, the TPM is used to sign a message to be sent (i.e. to encrypt a 32B hash of the message, which is computed by the host) and to decrypt another message (the AES256 nonce, so 16B of payload). In our benchmarks with ST33TPM/I²C, this process takes around 1 second, using a rather naive implementation. Experience in TPM usage tells us that a more optimized implementation would take around 100 ms (performances can change according to the TPM version and supplier). In the PubSub protocol, the client-server connection is only done to fetch the group keys, so the impacts establishing the connection will strongly depend on the group key lifetime. In some real-time systems with strict needs in terms of worst-case latency, it may be an issue.

While the TPM reinforces security, it cannot prevent all type of attacks. Remote Code Execution attacks on the software interacting with the TPM would allow an attacker to turn a legitimate node into a malicious agent. This calls for a defense-in-depth approach, with use of threat protection and detection on software in addition to the solution presented in this paper.

In a production system, the TPM can additionally be used to perform *measured boot* coupled with a root of trust, and the use of the created key pair to ensure that the device is only used when the system integrity can be ensured, reinforcing the TPM usage to protect the group keys.

Finally, decommissioning a TPM should also be done with care, as the private key may still be embedded in the device. Ensuring total and irreversible wipe out of the key is possible, but

this action must be clearly planned in the device life cycle. In all cases, it is recommended to also revoke the device certificate from the certificate authority, preventing potential use of an incorrectly cleared TPM.

4. Industrial Application: Securing Railway Predictive Maintenance

We demonstrated the effectiveness and feasibility of our solution on a real-life industrial challenge involving geo-distributed computation: predictive maintenance of catenaries, and we ported our solution to industrial-grade equipments, demonstrating operational feasibility.

In this section we first describe the industrial challenge then we present our solution to address it and ensuring communication security by applying solutions presented in Section 3.

4.1. Predictive Maintenance

Usage of IIoT can bring benefits along the industrial systems' lifecycle, including but not limited to phases such as provisioning, deployment, production or operational use. Amongst these different stages, we explored the maintenance phase as a promising target for introduction of IIoTs.

Predictive Maintenance is a technique aiming to optimize the maintenance time by using information remotely harvested from sensors – typically IIoT devices – monitoring the industrial system of interest. In this approach, monitoring the industrial equipment's state allows to predict a short-term failure, and thus to dispatch a maintenance operator just-in-time. Predictive maintenance also allows implementing an optimization feedback loop, by harvesting knowledge on the industrial equipment behaviour and refining its states definition and thresholds, for instance through machine learning. Predictive maintenance is a popular way to introduce IIoT technologies into industry as (1) it is purely additive to the industrial system, without impacting the core critical functionalities and (2) it has a fast Return On Investment, leading to drastic decrease on cost of ownership if done correctly.

In order to define the security needs for the use case, we performed a risk analysis following the EBIOS [15] methodology involving cybersecurity and safety experts from the railway industry. We defined three critical assets: (1) the analytics datalake, (2) the train service availability and (3) the safety-critical network (i.e. the system in charge for real-time signaling).

Regarding data send to the analytics, the main need is integrity, as forged data may trigger emergency counter-measures, generally involving interruption of service on the whole track. Thus, forged sensor data can be used to perform business-critical denial of service, impacting asset (2). Furthermore, allowing unknown amount of untrusted data to be integrated to the analytics datalake defeats its purpose and prevents its smart exploitation, impacting assets (1).

Availability must be ensured in order to guarantee persistence of service (asset 2), although considering the average duration of a maintenance operation comparatively to information sampling, the only challenge will be ensuring that security measures taken will not deter from this objective.

Confidentiality is also needed on data, since sensors may indicate real-time position of the trains, which is a sensitive information since trains are national strategical assets. Hence the exploiting company must ensure protection of such information.

From a connection perspective, authentication and authorization must be strictly enforced in order to ensure security of the safety-critical network (asset 3). Although the maintenance and safety networks are currently strictly separated, it may not be the case in the future, since signaling / critical sensors information may be used to refine predictive maintenance. Furthermore, it cannot be ruled out that some hidden path already exists between the two networks, for instance through corporate IT network. Secure authentication ensures that no rogue device can connect into the network, while authorization ensures that any legitimate device will be used the way it is intended to (e.g. that an subscriber will not begin to publish data).

4.2. Catenaries Monitoring Infrastructure

We built a prototype of light SPD implementing our secure communication solution, and we tested it on an industrial equipment to ensure secure communications for predictive maintenance in the railway industry, which is at the same time a critical asset of the supply chain and an industrial system itself. Predictive maintenance however is not limited to this specific industry, but can and indeed should be applied in any industry where critical assets are costly to access and complex enough for failure to be difficult to predict on a strict time-based. Since modern manufacturing processes are typically distributed over large area and involving multiple stakeholders [16] - and indeed use complex equipment, they typically fit both requirements.

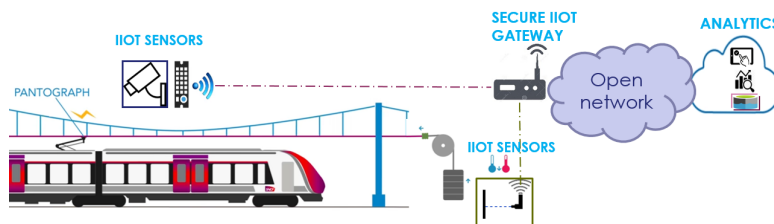


Figure 3: Catenary predictive maintenance

Specifically, we monitored wireless trackside sensors (connected thermometers and unwinder) to get information on the catenaries (respectively heating and physical tension), and exploiting them in order to infer catenaries state in real-time. These data are then transmitted to an analytics center that decides and prioritizes the maintenance operations according to different business factors, such as maintenance team availability and current position, relative severity of the catenary state, or per-hour cost of the line disruption; thus saving cost on maintenance while minimizing service disruption for train users. Figure 3 illustrates the use case.

Strict OT zone protection was not covered in this document, as it is heavily dependent on the actual use case - whether because of (1) physical protections impeding access to the OT network, (2) physical limitations on the OT devices or network prevent cryptographic security measures to be applied, or (3) relative low value of the data samples comparatively to the correlated data

produced by the secure gateway. Nevertheless, if possible and needed, the approach we use to secure the IT zone could be applied to the OT zone as well.

4.3. System architecture

We tested two sensors: an unwinder and a thermometer, both equipped with a STIMIO Railnode module, allowing to embed their outputs into LoRa frames. The gateway was based on a STIMIO Railnet⁴, an ARM Cortex-A7-based gateway supporting both LoRaWAN and 2G/4G communications. The LoRaWAN Join Server responsible for sensors enrollment was hosted by the gateway. The Railnet gateway is railway-certified and used in actual railway infrastructures.

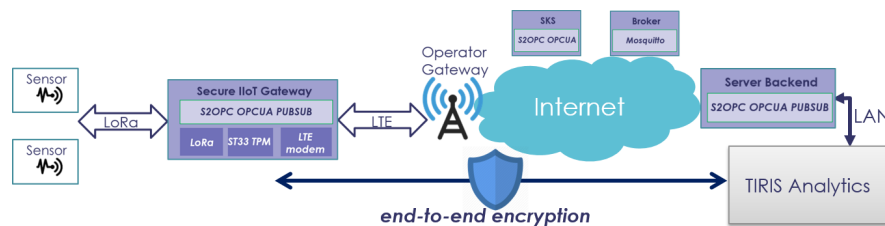


Figure 4: Smart Catenaries architecture

We applied our approach to secure a trackside gateway for connected sensors, in order to ensure connectivity and security of the predictive maintenance. As illustrated in Figure 4, sensors are connected through LoRaWAN to the gateway, which is in turn connected to open networks (such as the Internet) through LTE.

The communication protocol between the gateway and open network subscribers (in our case a monitoring server) is OPC UA PubSub with end-to-end encryption enabled. The MQTT protocol is used to transport the encrypted OPC UA PubSub messages, as it is more convenient to use a TCP broker to connect to remote devices. The gateway of the private subnetwork is then represented by two entry points: the MQTT broker and the SKS server used to distribute the group keys. Systemel's S2OPC⁵ library is used to build the OPC UA PubSub subscriber on the gateway, but also for the SKS server. The MQTT broker was the off-the-shelves, open-source tool Mosquitto⁶, while the Secure Key Service was implemented through a S2OPC server, this time working in client/server OPC UA mode. Both services were run on the same physical remote server, although in actual deployment we would encourage hosting them in two distinct machines.

Finally, the analytics center was made of a back-end S2OPC subscriber, in charge of establishing connection, getting OPC UA messages and transferring them to the analytics server. The latter has been simulated through simple display of the results in a remote machine, although connectivity to TIRIS⁷, the actual analytics server used by Thales railway division has been demonstrated in previous experiments.

⁴<https://stimio.fr/documentation/railnet/>

⁵<https://opcfoundation.org/products/view/safe-and-secure-opc/>

⁶Eclipse's Mosquitto broker: <https://mosquitto.org/>

⁷<https://www.thalesgroup.com/en/markets/transport/railways-digitalisation/tiristm-smart-maintenance>

Certificates from both SKS and clients embed their public key used for the authentication process described in Section 3.2.2 (associated with the related private key). In our use case, these certificates are all signed by the same Certificate Authority (CA), using SHA256 hash and a RSA4096 key pair. All end-users owned the CA certificate, and thus were able to check the validity of others equipment's certificates. While we did not experiment on it, the TPM could be used to ensure the CA certificate integrity, to ensure that an attacker did not tamper the file. In order to do it, a hash of the certificate can be stored within the TPM, and compared to the certificate file on-disk before any usage.

In our experiment, the CA was not connected to clients and SKS, so certificate revocation was not directly possible. In most operationally deployed use cases, a complete PKI should be set up, although the actual implementation of this PKI might differ a lot depending on the use case needs and constraints.

4.4. Implementing the gateway function in an industrial-grade SPD

As shown in Figure 5, we integrated a TPM2 ST33 from ST Microelectronics (the STPM4RasPI⁸ extension board) as a discrete component into STIMIO Railnet module. Its Common Criteria evaluation reached EAL 4+ [17], hence offering a satisfying level of security for this prototype.

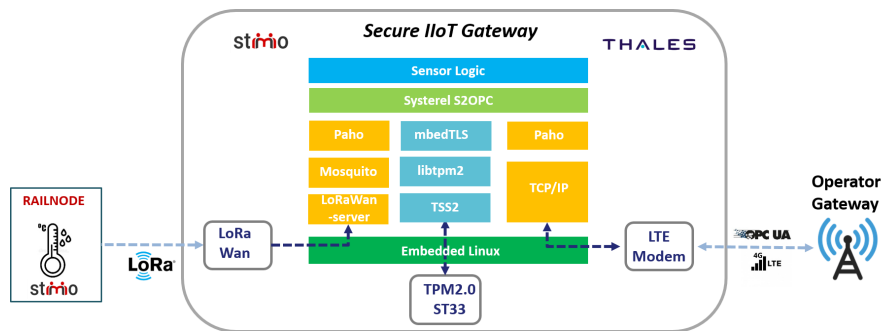


Figure 5: Secure IIoT Gateway implementation

The gateway operational communication with the IT zone was done through an OPC UA PubSub Publisher, while its authentication was performed through a OPC UA client. Both were built with Systemel S2OPC library. Cryptographic services for both OPC UA components relied on the open-source, light-weight cryptography library mbedtls⁹, which we adapted to communicate with the TPM¹⁰ through the TPM library from the TSS2 open-source stack¹¹, as well as the Thales-proprietary library *libtpm2*.

An OPC UA server was also added to the PubSub modules to expose the published data in a client-server manner. This access is mostly for convenience and tests, as data transfers between devices are only done in PubSub. The Paho¹² library, a light-weight MQTT client

⁸<https://www.st.com/en/evaluation-tools/stpm4raspi.html>

⁹<https://github.com/ARMmbed/mbedtls>

¹⁰<https://gitlab.com/systemel/S2OPC/-/tree/pab-tpm2>

¹¹<https://github.com/tpm2-software>

¹²Eclipse's Paho library: <https://www.eclipse.org/paho/>

implementation, is used by S2OPC as the library for the MQTT Transport of the OPC UA PubSub messages. All software used in this experiment is open-source, with the exception of libtpm2, a proprietary library developed by Thales and used as high-level API above TSS2. During the authentication phase, the S2OPC stack will use the customized cryptography stack to perform hardware-based secure authentication, as described in Section 3.2.1, communicating with the SKS with OPC UA Client/Server directly over TCP/IP.

During the communication phase, data received in LoRa from the remote IIoT sensor (railnode) are processed by the LoRAWan-server, which dispatches messages to the local MQTT broker (mosquitto), in turn queried by the S2OPC stack through the Paho library. Once the data are received and processed by the gateway sensor logic, a new information is sent through Paho to the remote MQTT broker.

Using these software and hardware components, we were able to implement a prototype of secure gateway for IIoT systems. More specifically, we ensured hardware-based authentication of the gateway, and the end-to-end signature and encryption for communications, as intended.

Some very basic level of edge computing was also performed in the gateway, such as computing relative heating and timestamping the data. In more realistic use cases, more extensive computation should be performed at edge (i.e. in the gateway).

Successful connection, persistent connectivity and effective signature and encryption were demonstrated through the use of forged messages and rogue gateways, which were both rejected by the SKS and unable to log in and access any secret or data from the system.

5. Related works

Organisms such as German IUNO or French GIMELEC are mainly promoting methods and tools in order to ensure national companies conformance to Industry4.0 standards [18]. While they may also support research activities, they mainly propose high-level guidelines for the industry. In the case of IUNO, usage of TPM for authentication is mentioned in [19], but no implementation is provided, neither its usage within an existing communication protocol is described. In [20], hardware properties (SRAM PUFs) are exploited to ensure secure authentication by guarantying integrity and confidentiality of the secret key, similarly to our usage of the TPM solution. This work, however, relies on a specific architecture which may not be usable in actual use cases. Furthermore, no indication on how to integrate the proposed protocol into established standards is proposed in the article.

Regarding communication protocols of industrial use cases, Data Distributed Services [21] (DDS) is an open standard for real-time distributed communications. While the standard is open regarding the actual communications implementation, it supports a brokerless publish/subscribe pattern of communications, and so can be compared to OPC UA which offers such possibility in its PubSub version. DDS is a rich standard offering fine-grain control of the Quality of Service. A security standard has been published, allowing security patterns similar to OPC UA PubSub [22]. Either base or security DDS specifications, however, only describe APIs when OPC UA provides in-depth protocol specifications, hence drastically increasing the tools interoperability. Furthermore, in our knowledge no national cybersecurity authority has performed a thorough review of DDS Security, as it was the case for OPC UA by the German BSI [9].

The OPC Foundation also mentions the possibility of using a TPM [23] or other secure storage solutions. However, it does not discuss its potential usage or benefits for authentication.

The wolfTPM library from wolfSSL¹³ allows integration of a TPM into the wolfSSL library, a cryptographic library implementing the TLS1.3 standard [24]. This approach is quite similar to ours, but it only applies to client/server connections. Its usage into the publish/subscribe communications schemes is not mentioned in the literature. Another example of such approach is presented by authors of [25], where benefits of integration of a TPM within the TLS cryptography standard [26] is exploited not only for authentication but also for payload encryption. However, as in the former case it does not cover the publish/subscribe topology, and in case of multiple publishers may lead to multiple unnecessary encryptions.

6. Conclusion

We propose in this paper a consolidation of the authentication of devices that are part of a distributed system, with both legacy devices that cannot be updated because of safety requirements, and remote devices that must use the public network and require end-to-end encryption. In [27], we assessed our solution with ISA/IEC 62443 - a set of standards relative to the security of industrial communication networks and systems.

The proposed solution uses open protocols to communicate its data (MQTT, OPC UA PubSub), enhancing the interoperability of the system, hence its maintainability. It also uses a TPM hardware module to conceal secret identifiers and guarantee the authenticity of the remote modules. We use secure boot in order to ensure the integrity of the device software. The developed prototype shows that the remote data distribution works efficiently and securely. Functional tests included in the S2OPC suite all passed successfully, and operational communications were working as intended. Key distribution was performed in a relatively long time (around 10 seconds), probably because of the low bandwidth of the SPI bus connecting the TPM to the CPU. Considering the timing needs of our application, such duration was acceptable. More constrained applications may require a closer integration of the TPM.

On a final note, while performances of the remote component are proven satisfying on its industrial prototype, complete industrialization of the solution requires derisking of the whole architecture, and more specifically further testing of the performances, since the solution's domain of application is Industrial IoT where a high number of devices provide situations similar to data lakes (this is particularly the case in the railway industry). In the current architecture, we have identified two elements that deserve further testing for the complete industrialization of the solution: the broker and the SKS server.

References

- [1] R. Langner, Stuxnet: Dissecting a cyberwarfare weapon, IEEE Security Privacy (2011).
- [2] J. Nazario, BlackEnergy DDoS Bot Analysis, Arbor (2007).
- [3] C. H. Kim, J.-J. Quisquater, Faults, injection methods, and fault attacks, IEEE Design & Test of Computers (2007).

¹³<https://www.wolfssl.com>

- [4] P. Tedeschi, S. Sciancalepore, Edge and fog computing in critical infrastructures: Analysis, security threats, and research challenges, in: 2019 IEEE European Symp. on Security and Privacy Workshops (EuroS&PW), 2019.
- [5] C. Metayer, P. Humbert, P.-A. Brameret, Vers une Implémentation Sûre et Cybersécurisée du protocole OPC UA, in: Congrès Lambda Mu 21, “Maîtrise des risques et transformation numérique: opportunités et menaces”, 2018.
- [6] M. Wolf, D. Serpanos, Safety and security in cyber-physical systems and internet-of-things systems (2018).
- [7] OPC Unified Architecture Specification, Specification, OPC Foundation, 2006-.
- [8] M. Puys, P. M.-L., P. Lafourcade, Formal Analysis of Security Properties on the OPC-UA SCADA Protocol, in: Computer Safety, Reliability, and Security, 2016.
- [9] Damm, Gappmeier, Zugfil, Plöb, Fiat, Störckuhl, OPC UA Security Analysis, Technical Report, BSI, 2017.
- [10] Advanced encryption standard, NIST FIPS PUB 197 (2001).
- [11] S. Boeyen, S. Santesson, T. Polk, R. Housley, S. Farrell, D. Cooper, Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, 2008.
- [12] Systerel, OPC UA Certificate Validation, <https://blog.systerel.fr/fr/posts/2020-11/opcu-certification-validation/>, 2020.
- [13] IEC, Trusted platform module library – Part 1: Architecture, Standard, 2015.
- [14] H. Raj, et al., fTPM: A Software-Only Implementation of a TPM Chip, in: Proceedings of the 25th USENIX Conf. on Security Symp., 2016.
- [15] Expression des Besoins et Identification des Objectifs de Sécurité - EBIO, Agence nationale de la sécurité des systèmes d’information (ANSSI), 2010.
- [16] W. Bauer, M. Hämmerle, S. Schlund, C. Vocke, Transforming to a hyper-connected society and economy – towards an “industry 4.0”, *Procedia Manufacturing* (2015).
- [17] ANSSI, Rapport de certification ANSSI-CC-2018/42 ST33TPHF20, https://www.ssi.gouv.fr/uploads/2018/10/anssi-cc-2018_42fr.pdf, 2018.
- [18] M. Waidner, M. Kasper, Security in industrie 4.0 - challenges and solutions for the fourth industrial revolution, in: 2016 Design, Automation & Test in Europe (DATE), 2016.
- [19] D. A., et al., Putting things in context: Securing industrial authentication with context information, *Int. Journal on Cyber Situational Awareness (IJCSA)* (2019).
- [20] C. Lipps, et al., Proof of concept for iot device authentication based on sram pufs using atmega 2560-mcu, in: 1st Int. Conf. on Data Intelligence and Security (ICDIS), 2018.
- [21] DDS Security 1.1, Data Distribution Service - Version 1.4, Specifications, OMG, 2015.
- [22] DDS Security 1.1, DDS Security - Version 1.1, Specifications, OMG, 2018.
- [23] S. W. Group, Practical Security Recommendations for building OPC UA Applications, White paper, OPC Foundation, 2018.
- [24] R. E., The Transport Layer Security (TLS) Protocol Version 1.3, RFC 8446, 2018.
- [25] K. Li, M. Mass, M. Ralph, A Type-safe, TPM Backed TLS Infrastructure, Technical Report, Carnegie Mellon University, 2012.
- [26] T. Polk, S. Turner, Prohibiting Secure Sockets Layer (SSL) Version 2.0, 2011.
- [27] O. Gilles, D. Gracia Pérez, P.-A. Brameret, V. Lacroix, Securing IIoT communications using OPC UA PubSub and Trusted Platform Modules, *Journal of Systems Architecture* (2022).