Journal of Network and Systems Management (2023) 31:32
https://doi.org/10.1007/s10922-023-09723-6

A Secure IIoT Gateway Architecture based on Trusted Execution Environments

Antônio Augusto Fröhlich[1] · **Leonardo Passig Horstmann**[1] · **José Luis Conradi Hoffmann**[1]

Received: 2 August 2022 / Revised: 2 August 2022 / Accepted: 20 January 2023 /
Published online: 3 February 2023

Abstract

Industrial Internet of Things (IIoT) gateways are affected by many cybersecurity threats, compromising their security and dependability. These gateways usually represent single points of failure on the IIoT infrastructure. When compromised, they can disrupt the entire system, including the security of the IIoT devices and the confidentiality and privacy of the data. This paper introduces a Secure IIoT Gateway Architecture that encompasses Trusted Execution Environment concepts and consolidated security algorithms to achieve a secure IIoT environment. Sensitive procedures of the IIoT, like device admission, bootstrapping, key management, authentication, and data exchange among operational technology (OT) and information technology (IT) are handled by the gateway inside the secure execution domain. The bootstrapping does not require devices to have any pre-stored secret or a pre-established secure channel to any trusted third party. Moreover, our architecture includes mechanisms for IIoT devices to safely interact with the Cloud without assuming the integrity of the gateways between them, enabling continuous verification of gateway integrity. A formal proof of the proposed solution security is provided. Finally, the security of the proposed architecture is discussed according to the specified requirements.

Keywords Device-to-device communication · Secure communications · Network architecture · Security and privacy · Security protocol

Antônio Augusto Fröhlich, Leonardo Passig Horstmann and José Luis Conradi Hoffmann have contributed equally to this work.

✉ José Luis Conradi Hoffmann
hoffmann@lisha.ufsc.br

Extended author information available on the last page of the article

1 Introduction

IIoT gateways connect OT devices to the IT infrastructure and, eventually, to the Cloud. These gateways are often based on ordinary operating systems and communication protocols, usually Linux and TCP/IP, and are therefore affected by the same sort of threats that afflict the Internet so intensely. However, while IIoT gateways have inherited many cybersecurity threats from general-purpose computing systems, the solutions usually applied to these systems are not directly applicable to them, mainly because of specific vulnerabilities and constraints associated with IIoT devices and applications [1, 2]. The range of IIoT attacks is much broader than on the Internet today. It includes bricking devices, jamming communication channels, denial of service, data infiltration and exfiltration, and even direct physical attacks, which have been demonstrated in several scenarios, including autonomous vehicles, smart homes, and infrastructure providers [3, 4].

Despite being widely applied to guarantee secure authentication in network communication, SSL/TLS solutions are too heavy for IoT device communication [5]. Widespread solutions adopt symmetric key algorithms for communication, such as AES or lightweight asymmetric solutions [6]. For instance, in Naoui et al. [7], the authors propose a secure communication protocol that relies on ECC and OTP algorithms to guarantee security to promote a lightweight security solution. On the other hand, Trusted Execution Environments (TEEs) provide additional protection and isolation guarantees for the software running on the platform, which can be used by gateways to improve the security and reliability of critical components such as temporary clear-text data handling and authentication management [8–10].

This paper introduces a Secure IIoT Gateway Architecture that explores TEE concepts to build a secure domain for the execution of sensitive gateway code and combines consolidated security algorithms such as ECDH, AES, Poly1305, and SHA-256, to provide security for the interaction between the OT and the IT worlds. Security sensitive operations, like bootstrapping, key management, and data exchange and marshaling, are always handled inside the TEE. Devices admission is handled on the fly through an ESA, expediting authentication tokens that enable mutual authentication between the gateway and the device without any further interaction with the ESA. Further gateway integrity checking is achieved through a secure channel established between the ESA and the devices during bootstrap. After bootstrap, devices communicate with the gateway using the accorded session key and OTP. The main contributions of the paper are:

- A Secure IIoT Gateway Architecture based on TEEs that encompasses device admission and bootstrapping, key management, local group communication, secure data exchange among OT and IT entities, and gateway integrity verification.
- The design of a protocol for authentication and key-establishment that does not rely on pre-stored secret or trusted third-party during device bootstrap.
- The design of a mechanism to allow IIoT devices to safely interact with the Cloud without assuming the integrity of the gateways in between them.

The remaining of this paper is organized as follows: Sect. 2 presents the related works and their solutions. Section 3 describes the design requirements that guided the development of the proposed gateway. Section 4 presents an in-depth description of the proposed architecture. Section 5 presents a formal proof of the security of the proposed solution. Section 6 presents a performance analysis of the secure marshaling procedure in a real implementation using ARM TRUSTZONE. Section 7 discusses the properties of the proposed solution in regards to the design requirements listed in Sect. 3. Finally, Sect. 8 concludes this paper.

2 Related Works

The interoperability required by IIoT infrastructures makes industrial applications susceptible to several attacks, raising security concerns based on the criticality of such scenarios [11]. An IIoT architecture design should address several security challenges to prevent exposing the infrastructure and data to vulnerabilities.

2.1 Gateway Architectures and Secure Communication Channel

According to Toğay et al. [12], IoT devices are getting attractive to attackers, which can use them for DDOS attacks or to attack the overall system directly. The authors focus on MQTT and propose the design of IoT gateways empowered with a secure element with storage for keys, a truly random generator, and AES 128-bit encryption capability. Their architecture envisions gateway security based on authentication at device bootstrap, which is performed considering pre-shared AES symmetric key, also used to generate session keys.

According to Toğay et al. [12], IoT devices are getting attractive to attackers, which can use them for DDOS attacks or to attack the overall system directly. The authors focus on MQTT and propose the design of IoT gateways empowered with a secure element with storage for keys, a truly random generator, and AES 128-bit encryption capability. Their architecture envisions gateway security based on authentication at device bootstrap, which is performed considering pre-shared AES symmetric key, also used to generate session keys.

Navarro-Ortiz et al. [13] address IIoT security with network and application session keys derived from the very own unique identifiers of the devices and the applications and a pre-stored symmetric application key. The authors recommend the use of secure storage technology to improve the security of the derived keys. The pre-stored symmetric application key is used to authenticate devices and encrypt messages. The confidentiality of the data exchanged on the IIoT is achieved by encrypting messages Payloads with the application session key. The messages are authenticated with the network session key, which is also used for message integrity checking. Similarly, but focusing on RFID supply chain networks and access control to Cloud services, Lin et al. [14] presents a mutual authentication protocol that bases on pre-stored keys, enabling a secure communication between devices and Cloud. Our solution, on the other hand, does not rely on pre-stored symmetric

keys, which beyond not requiring the customization of the devices, increases the communication security [15] and promotes the IIoT devices independence from the Cloud. Moreover, these solutions do not address the vulnerabilities imposed by an eventual gateway compromising. Other than that, our solution differs since a communication establishment implies on both the device registering and a gateway implicit authentication to an external security agent, which allow us to establish independent encryption keys and authentication tokens for the communication between devices, devices with the security agent, devices with the gateway, and gateways to the security agent.

Bienhaus et al. [16] proposed a secure gateway architecture for IIoT scenarios based on trusted platform module (TPM). The proposed approach aims to isolate the cryptographic keys and avoid illegitimate firmware manipulation. In their implementation, the TPM-based Gateway Architecture considers an OPC-UA core for multi-connectivity between OT protocols, where the OPC-UA format works as an intermediary abstract data model used for data transformation. Similar to our approach, the Firmware Hash is also considered by the key generation process. However, no isolation is guaranteed for the execution of the marshaling procedure itself, where the IIoT information is handled in clear text during the conversion between OT-IT messages.

2.2 Trusted Execution Environment and Secure Bootstrap

Ukil et al. [9] present an overview of the security of Embedded Systems in IIoT. The authors discussed the current solutions and pointed to TEEs as a solution to counter many physical and execution attacks by providing hardware isolation for the IIoT devices. Sebastian et al. [17] proposed a TEE-based solution to protect the integrity of measurements and commands of a smart inverter against attacks. Seungyeon et al. [18] propose a data security-sensitive classification model to reduce the overhead of TEE-based storage for an IoT gateway. However, their solution does not consider the possibility of a gateway being compromised. Pinto et al. [8] proposed a TRUSTZONE-based architecture to implement a TEE for Industrial IoT Edge Devices, claiming the TEE solution can meet both the security requirements and the real-time constraints of the IIoT. The authors reinforce the characteristics of TEE, such as memory and hardware isolation, to conclude that a TEE solution can provide the desired data confidentiality and resource availability. They mention the TRUSTZONE's absence of hardware mechanisms to assure data integrity and point to the necessity of a multi-level solution. Such concern is also addressed by Lesjak et al. [10], who point to vulnerabilities on ARM TRUSTZONE in respect of physical attacks to validate the necessity of a multi-level solution for the gateway security.

Ling et al. [19] proposed a hybrid booting approach with a secure boot for the Secure World and a trusted boot for the Normal World using ARM TRUSTZONE capabilities to guarantee the integrity of the IoT System during load time. In addition to that, their solution also performs periodic verification on the code running on the Normal World and a comparison to a pre-calculated reference value is done in order to attest runtime integrity. Their approach focuses on IoT Nodes and the device's

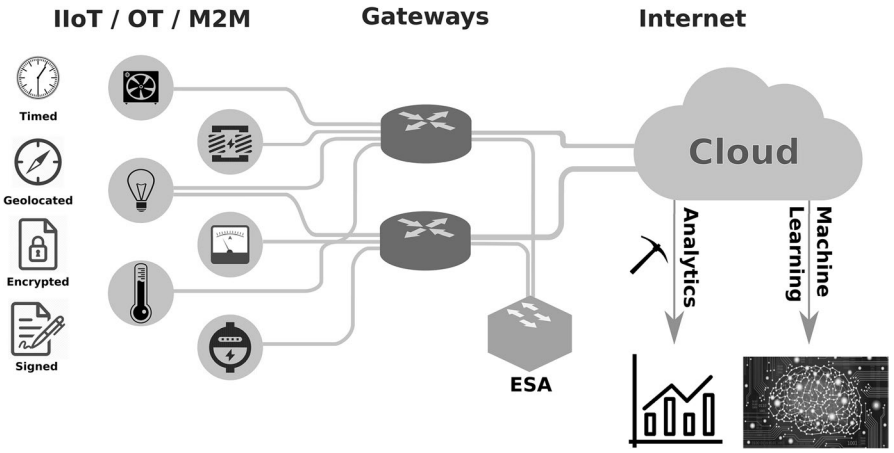

Fig. 1 The envisioned IIoT Architecture

functionalities are provided in the non-secure domain, which differs from ours since we are dealing with gateway security and security-sensitive operations are meant to be held on the secure domain.

3 Design Requirements

To fulfill the envisioned Secure IIoT Architecture depicted in Fig. 1 we based its design on the following requirements:

3.1 IIoT Operation Independent from the Cloud

IIoT devices, particularly those purely in the OT domain, usually communicate with each other using low-latency, time-triggered, cross-layer protocols that are specifically designed or customized for particular application domains. Creating dependencies with external processes running on the Cloud for the sake of security could compromise their operation, and therefore, should be avoided. This requirement aims at guaranteeing the availability premise of the well-known CIA security triad [20] not compromising the operation of the OT devices.

3.2 IIoT Data on the Cloud

Respected the Cloud-independence of IIoT devices operation, a functional requirement that arises is the collection of the data used for control operations in the IIoT Segment at the Cloud, enabling operation maintenance and planning optimizations [21]. In fact, Gateway Integrity checking Protocols like GIP, proposed by

Lucena et al. [22], bases on the property that data from the IIoT segment, and of interest to the Cloud, should be available in the Cloud.

3.3 Secure Communication

- *Security threats considered* The communication inside the IIoT, and between the IIoT and the Cloud, should consider the possible attacks that can be targeted to the proposed architecture. From this perspective, the threat model proposed by Needham-Schroeder [23], and formalized by Dolev and Yao [24], covers the main threats faced by communication systems and thus, is assumed as the Threat model in this paper. The powers of a Dolev-Yao attacker include: interfering with messages by blocking, eavesdropping, breaking down or modifying, sending new messages in the network by fabricating, re-ordering and replaying, and communicating with other entities in the network by initiating message exchanges and spoofing other entities.
- *Cross-domain encryption* Data exchanged on the IIoT, and between the IIoT and the Cloud, must be encrypted, with the IIoT segment, which usually operates under strict timing requirements, preferring symmetric algorithms [6], and the Cloud using ordinary PKIs with a combination of symmetric and asymmetric algorithms. This requirement aims at guaranteeing data confidentiality, one of the main aspects of security and one of the premises of the CIA [20, 25].
- *Strong authentication* Devices within the IIoT segment must be strongly authenticated so their identities can be validated as they exchange data on the IIoT, and between the IIoT and the Cloud. According to Tange et al. [20], authentication is one of the main mechanisms to ensure and assess the integrity of the IIoT devices and the data they exchange.

3.4 IIoT Devices

- *Precise time synchronization* Devices in an IIoT segment must keep their clocks synchronized, both for the sake of security and to precisely timestamp the data collected and stored on the Cloud for posterior analytics and ML. For instance, timeliness application requirements and protocol message routing policies rely on time synchronization, like TSTP [26] and TDMA-based protocols.
- *Geo-referencing* Devices in an IIoT segment should be aware of their locations, so the data they exchange can be consistently tagged with spatial coordinates, enabling additional analytics and ML processes. Moreover, geo-referencing can be applied in routing optimizations and fault-tolerance policies [27].

3.5 IIoT Gateways

- *Fault-tolerance* Gateways within the IIoT provide hardware and software replication mechanisms to enable fault-tolerance, avoiding single-point-of-failure issues, and consequently increasing availability. Fault-tolerance of the IIoT gateways is one of the main aspects to guarantee availability and ensure

the IIoT correct operation, which is seen as one of the main aspects of security considering the CIA principles [25]. In fact, Tange et al. [20] considers software redundancy and decentralization as topics of concern for IIoT availability.

- *Execution and storage security* Gateways in the IIoT must provide a Trusted Execution Environment (TEE) that ensures the confidentiality and the integrity of the loaded code and its data, enabling isolated execution. In this context, we see the security of the execution and storage as a key factor to ensure the IIoT data confidentiality, one of the main aspects of security systems [20].
- *Automatic update* Gateways in the IIoT must provide support to remote software updates, enabling remote firmware updates, remote reboots, and issue replicas to take over the control of the system. Tange et al. [20] state that the ability to configure, reconfigure, and update devices and gateways is crucial in Industry 4.0, as software and configuration of IIoT systems must have such ability to protect against previously unknown security threats.

3.6 IIoT Auditing

- *IIoT admission regulation* The IIoT architecture must hold an external security agent (ESA) that regulates the admission of new devices and gateways and grants their authenticity. This requirement aims to take advantage of the ESA as a secure entity to guarantee the control of the admission of devices into the IIoT network. Moreover, it creates an access control mechanism that relies on authentication to avoid breaking the integrity and confidentiality of data or eventually making it unavailable, which would disrupt all three CIA security premises [20, 25].
- *IIoT integrity checking* The ESA should provide means to audit gateways by checking the integrity of data exchanged in the IIoT segments. Tange et al. [20] state that verifying data integrity and authenticity is an essential feature. Moreover, integrity is part of the CIA security properties [25].

4 Proposed Architecture

In this section, we describe our IIoT Secure Gateway Architecture in detail. Figure 1 presents an overview of the envisioned IIoT architecture. In this context, we will adopt the term *IIoT segment* (also known as the OT domain) to designate a set of interconnected devices that communicate in a Machine to Machine (M2M) fashion and cooperate using a specific protocol (IIoT Protocol) with light symmetric encryption algorithms. In order to meet the requirements of the IIoT domain, the IIoT Protocol is usually supported by time synchronization features (e.g., PTP [28] and SPTP [29]) and geolocation. Moreover, signature mechanisms are also expected to guarantee trustfulness inside the OT domain. The secure *gateway* is an entity that connects OT devices to the IT infrastructure. Any interaction with the world outside the IIoT segment goes through the gateway, which uses ordinary operating systems and IP-based protocols typical for the Cloud (e.g., HTTPS with TLS).

The Cloud infrastructure (Information Technology (IT)) features storage, analytics, and machine learning models to support monitoring, optimizations, and planning. An External Security Agent (ESA) owned by the IIoT network owner resides at the IT where it continuously verify the integrity of the gateways and gives support to additional security operations. In terms of security, an External Security Agent is expected to provide the same level of protection as a Certificate Authority (CA). Critical components in the architecture are meant to be executed inside a Trusted Execution Environment (TEE), particularly in the gateways.

4.1 OT-IT Interoperability

The OT devices usually communicate with each other using low-latency, time-triggered, cross-layer protocols that are specifically designed or customized for particular application domains (e.g., TSTP [26], CAN (http://can-cia.org/), Ethercat (http://ethercat.org/), Profibus, and Modbus [30]). These protocols either do not address security or do so using light symmetric algorithms [6, 26]. We will refer to such protocols as the *IIoT Protocol*. Conversely, IT components communicate using the ordinary IP protocol stack, which addresses security using asymmetric algorithms and PKIs. Therefore, in order to promote the interoperability between OT devices and IT services, gateways must reach beyond the marshaling of the messages across two different protocols: they must also handle timeliness, trustfulness, and any sort of metadata relevant to each of the domains.

Figure 2 depicts such concepts, where the gateway must attend to the particular requirements from the IIoT scenario, for instance, Energy management, trickle, timing, security, authentication, and location. From the IT perspective, which follows an ordinary IP protocol stack, the gateway must marshal relevant features from the OT domain, like Trust, Time, and Location, at the application level alongside data.

Since both domains, OT and IT, usually approach security from different perspectives, the marshaling of messages that cross domains must inevitably pass through a clear text stage. Since gateways are directly connected to the IT infrastructure, and hence are subject to the same sort of attacks faced by computers on the Internet, this clear text stage becomes extremely critical for the security of the IIoT. Properly handling this phase is one of the ultimate goals of the architecture proposed here.

4.2 Typical Platform

Figure 3 presents a typical platform featuring TEE with the modeling of the secure gateway. In this architecture, the gateway procedures are split into secure and non-secure domains. The non-secure domain of a gateway hosts traditional OS services at both privileged and non-privileged levels, networking in particular. The secure domain emerges from a TEE, with its crypto-core, isolation mechanisms, and key storage being the foundations of security mechanisms used in the gateway software. Secure-sensitive data and procedures, like the Secure Marshaling, are held in the isolated secure world to ensure confidentiality. In our architecture, these mechanisms

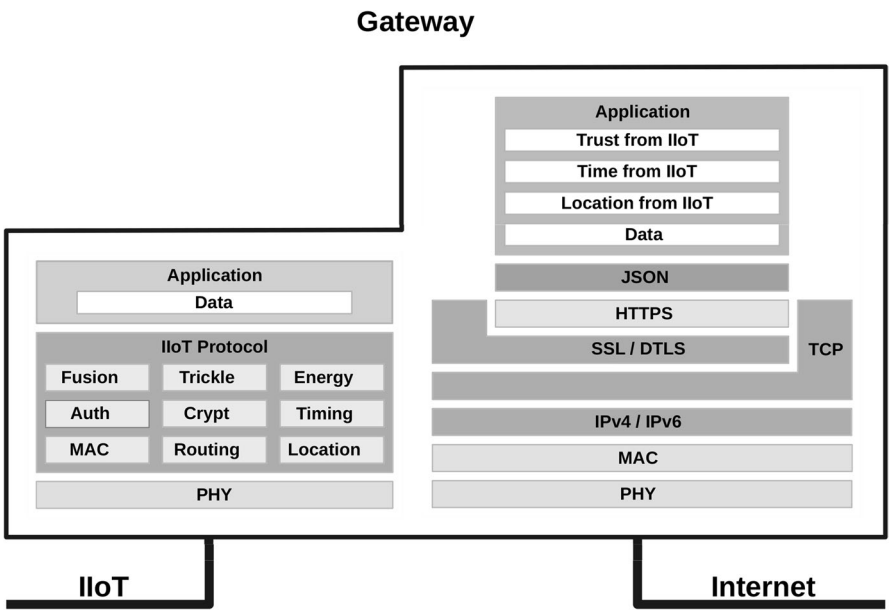

Fig. 2 OT and IT security protocols in a typical IIoT gateway scenario, where the gateway functionalities include marshaling messages between both domains

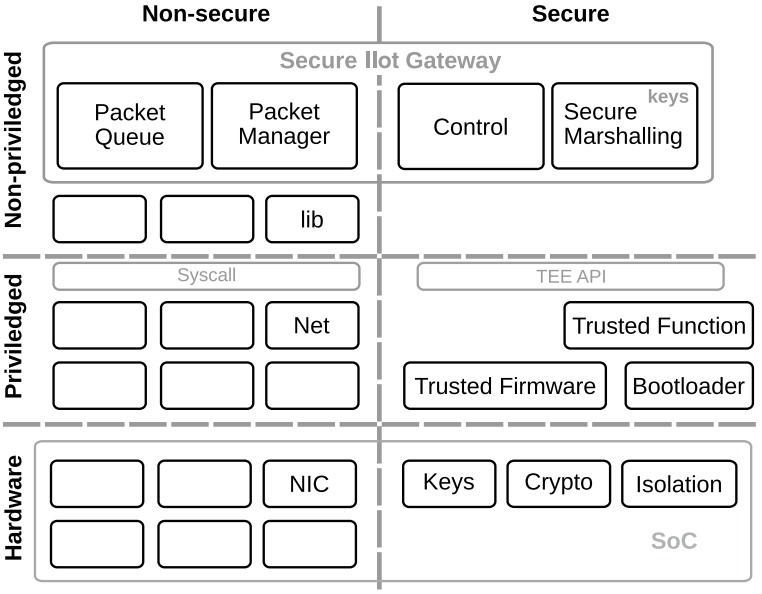

Fig. 3 Overview of a typical platform featuring a Trusted Execution Environment for the Secure IIoT Gateway Architecture

are two: a RoT for the gateway and its isolation from the non-secure world. The acceleration provided by the crypto-core is desirable, but not fundamental for the proposed architecture since all the associated algorithms can be implemented in software.

4.3 Root of Trust

The gateway RoT relies on the TEE kernel to be properly initialized during the secure boot of the system. In this way, we assume the TEE initialization to precede the OS boot in the non-secure domain and restrict the access to the hardware components on the secure domain (i.e., provide hardware isolation). A typical secure boot procedure starts with the TEE checking the integrity and authenticity of the trusted firmware using its pre-stored key and some sort of CA.

After the validation of the firmware, the bootloader is responsible for loading the OS in the non-secure domain. Here, we assume that, before initializing the OS, the TEE's secure boot procedure provides a RoT in the form of a *trusted function* to be used by the gateway software. This function contains all the code for the secure components of the gateway, along with a public key certificate for interactions with an ESA. Moreover, the non-secure part of the gateway software, which runs as an ordinary process on top of the OS[1], will later activate this function.

4.4 IIoT Gateways

In our architecture, we model the Secure IIoT Gateway based on four main components: Packet Queue, Packet Manager, Control, and Secure Marshaling. Gateway functionalities encompass the control of devices admission, bootstrapping, key management, and communication with the IT services (i.e., Cloud storage, ESA, and CA), which involves a marshaling process that enables cross-domain communication. In the non-secure domain, due to the connection to IT services in a standard IP stack, the gateway is exposed to several vulnerabilities. Internet attacks can compromise gateway integrity, availability, and data confidentiality, especially if using a general purpose OS such as Linux, making a gateway as vulnerable as a regular computer. In this way, to secure critical operations dealing with sensitive information, like marshaling and gateway control, we envision the embedding of these operations inside the TEE, which, in contrast to REEs, provides isolation for code execution and data handling.

Incoming packets from both IT services and IIoT devices are stored inside the Packet Queue and are consumed by the gateway Packet Manager component. The Packet Manager invokes the secure domain components by dispatching the packets according to their type and the protocol policy, like timeliness. Moreover, any

[1] A platform like this, with the same assumptions and mechanisms, can also be used for OT devices. However, since OT are isolated from the IT by the gateway, trustfulness is often regarded as a function of design, implementation, and operation of such devices and their coordination in an IIoT segment.

security-sensitive content is expected to be encrypted, as those components are set in the non-secure domain.

Packets that encompass the security bootstrap of the devices, the bootstrap of the gateway itself, or control messages incoming from the ESA, are dispatched to the Control component inside the secure domain, handling accorded keys and authentication procedures without impairing their confidentiality or integrity. Lastly, packets incoming from the IIoT devices to the IT services through the OT protocol, and vice-versa, are dispatched to the Secure Marshaling inside the secure domain, which marshals packets from one protocol to another while accounting for the timeliness, trustfulness, and any sort of metadata relevant to each of the domains. The proposed Architecture is agnostic of OT protocol, solely modeling operations to support a secure execution of an IIoT Gateway.

For authentication purposes, the gateway derives a certificate, issued by a CA, using a Trusted Function loaded by the Trusted Firmware, only accessible via the TEE API.

4.4.1 IIoT Gateway Bootstrapping

The gateway bootstrap starts by establishing HTTPS sessions with the Cloud and the ESA, thus ensuring mutual authentication through a CA. During this procedure, both the gateway and the ESA store the public key of each other's certificates (i.e., CY_g and CY_e), for further interactions. Every forthcoming communication between the gateway and the ESA, or the Cloud, is done through the established HTTPS sessions. The HTTPS session establishment is depicted by arrow 2 of Fig. 4.

4.5 IIoT Devices Bootstrapping

While specifying the elements of the proposed architecture that are related to IIoT devices, we took into consideration the fact that the diversity of types and associated standards is very large and might be even more diverse in the future. Therefore, we make minimal assumptions about IIoT devices in the architecture. We assume that IIoT devices:

- Are able to directly communicate with each other using any sort of network, wired or wireless.
- Are never directly connected to the Internet, so any external communication takes place via a gateway;
- Are able to execute a function that is trusted, either by design (e.g. isolated devices, with non-conventional operating systems, using non-conventional communication protocols) or via a TEE.

The activation of IIoT devices in the proposed architecture includes a phase dedicated to the establishment of secure communication channels with: (a) a gateway, (b) the local group of devices (i.e., an IIoT segment), and (c) the ESA. Channels (a) and

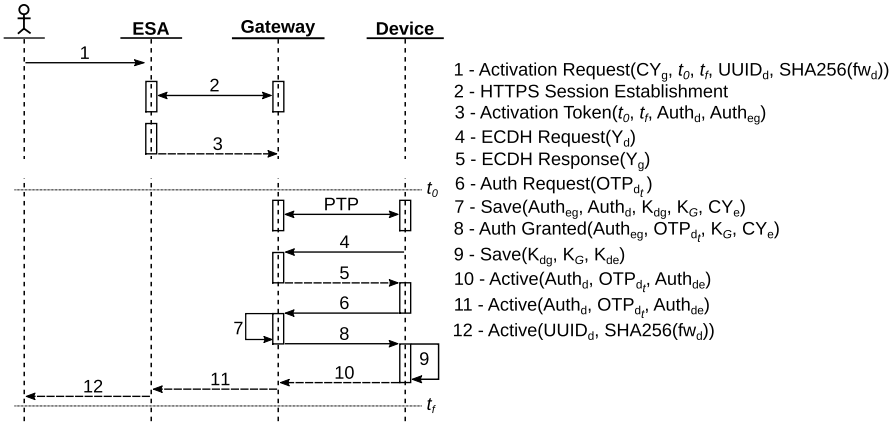

Fig. 4 Security bootstrapping of IIoT devices

(b) are modeled as time-triggered, low-latency, low-jitter channels used for real-time control, while (c) is an ordinary Internet channel. Channels (a) and (b) are therefore secured with symmetric-key algorithms, while (c) uses ordinary PKIs. For the three cases, we also avoid the known vulnerabilities associated with pre-stored symmetric keys and key distribution procedures [15] through the adoption of a key-agreement protocol. The whole procedure is summarized in Fig. 4 and will be detailed next.

The proposed architecture focus on providing the IIoT devices means to establish both a secure channel with ESA and a trust relationship with respective IIoT segment gateways. The trust relationship allows IIoT devices and gateways to authenticate without depending on pre-stored secrets or costly communication to CA's, which may not fit inside a time-triggered, low-latency, low-jitter channels used for real-time control. The secure channel established with the ESA allows the construction of integrity checking mechanisms to verify the correct operation of the IIoT gateway based on the messages exchanged on its respective IIoT segment.

4.5.1 Establishment of Secure Channels with Gateway and ESA

The activation of an IIoT device begins with a human operator sending a message, using a sane computer in the IT infrastructure, to an ESA in order to signalize the intention to activate such a device. The message

$$Activation\_Request(CY_g, t_0, t_f, UUID_d, SHA256(fw_d)) \quad (1)$$

which is represented in Fig. 4 by arrow 1, contains the public key CY_g of the certificate of a gateway in the target IIoT segment, the activation window $[t_0, t_f]$, a manufacturing-time identification of the device, represented here as $UUID_d$, and the SHA256 of its firmware $SHA256(fw_d)$. CY_g enables the ESA to identify the target gateway and also to confirm that it has previously authenticated itself to that ESA or to another cooperating ESA monitoring the same target IIoT segment. $UUID_d$ can be

extracted from labels on the device or its shipment documents and made available to the operator on secure storage. $SHA256(fw_d)$ can be automatically generated at compile-time and made available to the operator on secure storage.

The activation window $[t_0, t_f]$ parameter is an application-dependent feature that must correspond to the criticality of the system. During this time window, the security and confidentiality of the authentication parameters should be guaranteed. For instance, in high-criticality applications, the activation window granularity may be set to milliseconds, following the expected end-to-end latency of the authentication process. On the other hand, the activation window granularity may be set to hours or days for low-criticality applications.

The ESA reacts to the activation request by sending the corresponding gateway a message

$$Activation\_Token(t_0, t_f, Auth_d, Auth_{eg}) \tag{2}$$

which is represented in Fig. 4 by arrow 3, containing the time window for the activation of the device and two authentication tokens: $Auth_d$ and $Auth_{eg}$. The first will be used by the gateway to authenticate the device, and the second will be used by the device to authenticate the gateway. These tokens are obtained from the device's $UUID_d$ and $SHA256(fw_d)$, which are only known to the human operator and to the device itself. The malicious guessing of such information, either by a compromised gateway or by a rogue device, is limited by the activation window, as the information is not used in the future for authentication purposes (see Corollary 3 at Sect. 5). So, the ESA and the device independently derive the tokens using an intermediate implicit identifier Id_d calculated as:

$$Id_d = SHA256(UUID_d + SHA256(fw_d)) \tag{3}$$

Id_d is used exclusively to derive the authentication tokens. It is not further used for communication. The tokens are then derived by using Id_d as a key to encrypt the very own identifier with AES:

$$Auth_d = AES(Id_d, Id_d) \tag{4}$$

and also the ESA's public key CY_e:

$$Auth_{eg} = AES(Id_d, CY_e) \tag{5}$$

Other than that, the confidentiality of Id_d is assured by AES resistance to the plain-text attack. Nonetheless, $UUID_d$ and $SHA256(fw_d)$ confidentiality rely on the security property of the first preimage resistance of SHA256 [31] along with the activation window.

The gateway keeps a record of all devices in the IIoT segment it mediates. Prior to the activation of any new device, a record must be entered into the gateway's local database of devices, thus specifying a time window for its activation. Whenever a gateway receives an *Activation_Token* from an ESA, it becomes aware of a possible device activation and inserts a record (t_0, t_f, $Auth_d$, $Auth_{eg}$) into its database. If no activation takes place until t_f, the record is simply deleted.

From the side of a device, the activation begins with a procedure based on the IEEE 1588 standard for PTP to synchronize the device's clock with the gateway's. High-resolution timestamps are subsequently used to match the activation window chosen by the operator activating the device and also for message encryption and authentication.

After synchronizing its clock with the gateway using PTP, a booting IIoT device performs an ECDH key exchange with the gateway.[2] Initially, the device d generates a key pair (Y_d, X_d) using a curve such as the SECP256K1 [34], where Y_d is the public key, and X_d is the private key. The device then sends its public key Y_d to the gateway through a $ECDH\_Request(Y_d)$ message (represented in Fig. 4 by arrow 4). The gateway reacts to the message generating its own key pair (Y_g, X_g) and sending back a $ECDH\_Response(Y_g)$ message containing its public key Y_g to the device (represented in Fig. 4 by arrow 5). Both then compute the master secret K_{dg} independently, the gateway using

$$K_{dg} = Y_d^{X_g} \ mod \ q \tag{6}$$

and the node using

$$K_{dg} = Y_g^{X_d} \ mod \ q \tag{7}$$

where q represents the ECDH parameters agreed at design or deployment time.

This key exchange is subject to man-in-the-middle attacks [35], so it is followed by an authentication phase.

To authenticate itself to the gateway, the device uses the $Auth_d$ computed using (3) and (4), and a One-Time Password OTP_{d_t} created with the Poly1305-AES MAC [36] as follows:

$$OTP_{d_t} = Poly1305(K_{dg}, Auth_d \oplus t, m) \tag{8}$$

The Poly1305 takes the previously generated ECDH master secret as one of its inputs and a combination of $Auth_d$ and a timestamp t representing the current time as the other to produce a MAC on an arbitrary message m. The device then sends the following message to the gateway for validation:

$$Auth\_Request(OTP_{d_t}) \tag{9}$$

which is represented in Fig. 4 by arrow 6. Whenever a gateway receives an *Auth_Request* message, it searches the database of devices pending activation.

[2] The ECDH's key pair is generated using an elliptic curve, which makes it 10 times more efficient than the traditional Diffie-Hellman for 256 bits ECDH keys [32]. Moreover, this difference increases whenever we increase the size of the key. In [33], the performance of the ECDH algorithm was assessed on an embedded platform featuring a 32-bit, 26MHz ARM7TDMI-S processor with 128kB of flash memory and 96kB of RAM, with execution time in the granularity of a few seconds, which demonstrates the ability of this simple platform to perform such operations. Finally, this key generation is only required once in the proposed bootstrap process, therefore, further key generation will happen very sporadically given the key management procedure.

It then computes OPT_{d_t} for each combination of ECDH master secrets not yet authenticated and $Auth_d$ pending activation using (8). If a match is found, then the device's status is changed to active, and K_{dg} is bound to it through $Auth_d$. Since neither $Auth_d$ nor $UUID_d$ and $SHA256(fw_d)$ were ever transmitted over the IIoT segment, the authenticity of the device is established. Whenever an activation does not take place within the specified time frame, $Auth_d$ is marked as compromised and the device d is excluded from the IIoT segment. If the OTP fails verification, the message is ignored. If the result matches, the session parameters K_{dg}, K_g, $Auth_d$, $Auth_{eg}$, and CY_e are saved to a non-volatile, secure, memory to be restored in the eventuality of a gateway reboot so that it can recover the parameters and the bootstrapping procedure does not need to be repeated. The timing of the activation procedure can be relaxed by truncating the timestamp t to express time with an arbitrary resolution based on time synchronization accuracy and typical network latency [37], yet preserving PTP-grade precision, and running OTP verification with the truncated timestamp ± 1.

Next, the gateway sends a message to the device to confirm the establishment of a secure communication channel based on K_{dg} between them. The message:

$$Auth\_Granted(Auth_{eg}, OTP_{d_t}, K_G, CY_e) \tag{10}$$

which is represented in Fig. 4 by arrow 8, also confirms that the device is connected to a trusted gateway since the Id_d of the device was not sent to the gateway and the only way it could have to get to know $Auth_{eg}$ is via an ESA that trusts it. The device validates $Auth_{eg}$ from its own Id_d, the ESA's public key CY_e provided by the gateway and using (3) and (5). If the result matches, the session parameters K_{dg} and K_G are saved to a non-volatile, preferably secure, memory. In the eventuality of a reset by a watchdog timer or a similar transient failure event, the device can recover the parameters and the (relatively expensive) bootstrapping procedure does not need to be repeated. K_G is a group key that will be explained later.

At this point, device and gateway trust each other, the device has evidence that the gateway is also trusted by the ESA, and the requirement for device operation independent from the Cloud is satisfied. However, an inexpensive additional step can be of great use for further interactions with the Cloud. Therefore, a final message is sent from the device to the operator, via the ESA and the gateway, to confirm the activation and, at the same time, establish a secure communication channel between the device and the ESA. Initially, the message:

$$Active(Auth_d, OTP_{d_t}, Auth_{de}) \tag{11}$$

is sent to the gateway, which rencapsulates it with the Cloud protocol (i.e. HTTPS) as:

$$Active(Auth_d, OTP_{d_t}, Auth_{de}) \tag{12}$$

which are represented in Fig. 4 by arrows 10 and 11, respectively.

$Auth_{de}$ is an additional authentication token, similar to the $Auth_d$ and $Auth_{eg}$ described earlier. It uses the same Id_d from (3) as an AES key to encrypt a hash including the AES's public key as follows:

$$Auth_{de} = AES(Id_d, SHA256(UUID_d + SHA256(fw_d) + CY_e)) \tag{13}$$

Since only the device and the ESA know the involved parameters, the validation of $Auth_{de}$ yields both a shared secrete for further interactions, besides confirming the activation of a new device. In this way, to establish a secure channel, both the ESA and the device computes a symmetric key K_{de} as follows:

$$K_{de} = SHA256(Id_d + Auth_d + Auth_{de} + OTP_{d_t}) \tag{14}$$

Note that K_{de} relies on the OTP_{d_t} that will be forwarded in message *Active* to the gateway and ESA. This feature enables K_{de} confidentiality to be not reliant on the confidentiality of $UUID_d$ and $SHA256(fw_d)$ after the activation window ends (proved in Theorem 2 in Sect. 5). Finally, K_{de} is saved to a non-volatile, preferably secure, memory alongside the other sensitive information saved by arrow 9 in Fig. 4 for recovery after an eventual reset.

The ESA concludes the activation process by acknowledging the operator with the following message:

$$Active(UUID_d, SHA256(fw_d)) \tag{15}$$

which is represented in Fig. 4 by arrow 12.

4.5.2 Symmetric Group Key

Requiring all communication in an IIoT segment to go through the gateway can be too restrictive for many application scenarios, so, in our architecture, each node can be made part of a communication group during initialization. Devices within a group can directly communicate with each other by encrypting their messages using the K_G symmetric key received during the bootstrapping. Similarly, gateways can send multicast messages to a whole group at once using the same key. Authentication is not restricted in this scheme, because messages encrypted with the group's key are still authenticated with an OTP derived from the device's individual key. This scheme avoids more expensive alternatives, such as Group ECDH or key negotiations among peers. Moreover, group formation is not dictated by the architecture and can follow any policy that makes sense for the application scenario, including the geographic location of devices, their types, the role they assume in the plan, etc. Each device, however, can be associated with only one group.

4.5.3 Secure Communication

After completing the bootstrap, devices communicate by sending messages that are authenticated through their individual keys using (4) and (8) and encrypted with their group keys as follows:

$$Message(AES(K_G, Data), Auth_d, OTP_{d_i}) \tag{16}$$

4.5.4 Key Management

Key management after initialization is a major problem for IIoT since the bootstrapping procedures can rarely be executed during normal operation without disrupting the timing of the adjoining processes. Defining expiration times for keys and key revocation protocols for this architecture is straightforward in functional terms. For example, a timestamp can be added to the *Auth_Granted* message to determine that a device must renew its master secret before that instant. A *Renew_Key* message could also be easily added to the protocol to indicate that a key has been revoked and that a new negotiation must be started. Replay attacks may be countered by the verification of the message timestamp with the previous version of the same kind of message. The real issue pertains to the non-functional requirements of an IIoT system, including the delay incurred by key management operations directly on the affected nodes and indirectly on the network. For generality, we do not explicitly define key management procedures in our architecture. Instead, we model them as re-initialization events.

4.5.5 Gateway Integrity Verification

Although the security-sensitive operations and assets of the gateway are isolated into the secure domain, its non-secure domain counterpart can still be corrupted, denying access to resources like the Packet Queue. However, neither the integrity of messages nor their confidentiality will be affected by a non-secure domain corruption, as every message sent by the devices will be authenticated and encrypted using (4) and (8) as in (16). To challenge the gateway, a Gateway Integrity Check Protocol can be employed, for instance, Gateway Integrity checking Protocol [22], which promotes the integrity checking at the ESA by collecting information from the sensors network through challenges to the devices.

To prevent the gateway from interfering with the results of the challenges of the Integrity Checking Protocol, the ESA and the devices use the symmetric key K_{de}, which is generated after the device bootstrap according to (14). To improve security, at every message exchanged between ESA and the device, the value of K_{de} is updated as follows:

$$K_{de} = SHA256(K_{de} + SHA256(m)) \tag{17}$$

with m as the content of the message. After the establishment of K_{de}, each message exchanged between the ESA and the devices is encrypted and authenticated with K_{de} (i.e., an encrypted message m is concatenated with $AES(K_{de}, SHA256(M))$). Whenever a message is sent, K_{de} is updated at the sender. Whenever a message is received, the signature is verified by comparing the SHA256 hash of the received message and the encrypted SHA256 hash. Lastly, K_{de} is updated at the receiver. To avoid errors on the verification, the K_{de} keys used on the previous messages within

the truncated timestamp ± 1 as the received message timestamp is accepted. The truncation parameter is an application-defined arbitrary resolution. At every update, the oldest K_{de} key previously stored in the non-volatile (secure) memory is also updated. In this way, whenever a device reboots, the communication can be safely restored.

4.5.6 Rebooting

When an active device reboots, by a watchdog timer or a similar transient failure event, after having concluded the bootstrap procedure described in Sect. 4.5.1), any new $Auth_Request(Auth_d, OTP_d)$ messages incoming from that same device will be discarded by the gateway for security reasons. Thus, to restore a device after a reboot, time synchronization must be reestablished through PTP and the sensitive information saved during initialization (i.e. K_{dg}, K_G, and K_{de}, represented in Fig. 4 by arrow 9) must be recovered from a (secure) non-volatile memory so the previously accorded keys, and $Auth_d$ can be recomputed according to (4).

4.5.7 Revoking

When an active device is removed from the network, once it contains sensitive information saved during initialization (i.e. K_{dg}, K_G, and K_{de}), a *Revocation_Request*(CY_g, $UUID_d$, $SHA256(fw_d)$) must be issued by the operator to the ESA, which will then forward the *Revocation_Token*($Auth_d$) to the respective Gateway. On reception, the Gateway triggers a *Renew_Key* procedure except for the one with the given $Auth_d$ and removes it from the active devices list. In the case of *Revocation_Request*s issued for devices not yet authenticated, the Gateway removes the respective $Auth_d$ from the pending activation list. Since the operator is an employee of the company, in the scenario where he leaves the company, all open *Activation_Request*s for devices not yet activated must be revoked.

4.6 Gateway Replacement and Reboot

The reboot of a gateway can be triggered by a watchdog timer or a similar transient failure event. To restore its previous state, time synchronization must be reestablished through PTP. Moreover, sensitive information saved during devices bootstrapping ($Auth_d$, K_{dg}, and K_G for active devices, and $Auth_{eg}$ and CY_e of the *ESA*, represented in Fig. 4 by arrow 7) must be recovered from non-volatile, secure, memory. This enables the recovery of secure communication between the OT, gateway, and IT.

The replacement of a compromised gateway, however, is a more complex activity. Whenever an ESA detects a compromised gateway, it dispatches a maintenance request to replace it. The new gateway will start without any knowledge about the previous state of the IIoT segment, since it will not have access to the previously established keys K_{dg}s and K_Gs. So repeating the whole bootstrap procedure described in Sect. 4.4 is necessary. The ESA is responsible to re-trigger the bootstrap

procedures for each active device. The devices become aware of the procedure due to the unresponsiveness of the previous gateway and re-trigger their bootstrapping accordingly.

5 Security Analysis

This section provides a formal analysis of the security of the bootstrapping protocol. Before a node can interact with the other nodes inside the IIoT network, it must first authenticate with the IIoT gateway, which is represented by arrows 4 onward in Fig. 4. Moreover, a successful attack implies that the attacker can obtain the group key K_G for an IIoT segment or have a device (or set of devices) operating with a fabricated (forged) group key K'_G (i.e., the attacker authenticated itself with the device as a valid gateway).

The security analysis presented here relies on the following assumptions:

1. The ESA is secure, and so is the communication channel established between the ESA and the IIoT Gateway.
2. The Gateway marshaling and control code does not contain malicious functions when compiled and deployed to the Trusted Execution Environment.
3. The devices' authentication information ($UUID_d$ and $SHA256(fw_d)$) confidentiality should be guaranteed prior to the expiration of the activation window.

Theorem 1 *If the gateway authenticated a device, it must have done so with an IIoT device whose activation was requested by the ESA. Likewise, if an IIoT device authenticates with another device, it must have done so with a trusted Gateway.*

Proof The adversary, a malicious device, try to insert himself into the authentication process between the IIoT device and the trusted IIoT gateway, configuring a man-in-the-middle attack. This attack has two parts: First, the adversary intercepts the IIoT device message to the gateway. After this, the adversary attempts to authenticate itself with the gateway as the device and the to device as a gateway. This process can be modeled as a game where A is the adversary device, D is the IIoT device, and G is the IIoT gateway:

1. A intercepts the message *ECDH Request* from D to G (arrow 4).
2. A tries authenticating with G and with D.

Whenever A intercepts a message $ECDH\_Request(Y_d)$ from D to G, it generates a key pair (Y_a, X_a), and sends an $ECDH\_Response(Y_a)$ to D, according a symmetric key K_{da} with the device. Simultaneously, A sends a $ECDH\_Request(Y_a)$ to G, which will respond with $ECDH\_Response(Y_g)$, according a symmetric key K_{ag}. By assumptions 1 and 2, A is not able to retrieve confidential information from devices through attacks to the ESA and the gateway, respectively. By assumption 3, A is not

able to retrieve authentication information of devices prior to the expiration of the activation window. Thus, A cannot generate a valid $Auth_d$ through (3,4) within the respective activation window of device D. Moreover, since $Auth_d$ is never exchanged in any message inside the IIoT segment, nor are Id_d and its components, A must rely on a brute force attack on $Auth_d$ generation. However, the probability of generating an $Auth_d$, by brute force, that matches a SHA256 digest (4) is equal to $\frac{1}{2}^{256}$, which makes it infeasible to generate such a string within the strict time limitations imposed by the activation window expressed in message *Activation_Request* (arrow 1), subsequently not being able to generate a valid OTP_{a_t}. Therefore, G will refuse the authentication since no $Auth_d$ pending activation will match the OTP_{a_t} provided by A, and A will not receive K_G, which prevents it to decrypt group messages, nor $Auth_{eg}$ and CY_e.

After successfully generating K_{da}, D will forward *Auth_Request*(OTP_{d_t}) to A. Note that, in this scenario, OPT_{d_t} was generated using K_{da}, and if A forwards the received OPT_{d_t} to G, G will fail to authenticate A since $\forall K_{dg}$ pending in G $K_{dg} \neq K_{da}$. Moreover, A is able to synchronize time using *PTP*, however, it cannot extract the content of OPT_{d_t}, since A does not possess $Auth_d$ to generate the nonce required by the Poly1305-AES algorithm. Thus, it cannot generate a valid OPT_{a_t} with K_{ag} using OTP_{d_t}. Therefore, A will never be able to authenticate itself with G, and so forth, have access to K_G, proving the first part of this Theorem.

On the other hand, using the previously established K_{da}, A may pursue to conclude an authentication with D, according a fabricated K'_G, by sending *Auth_Granted*$(Auth_{ea}, OTP_{d_t}, K'_G, CY'_e)$ to D. However, this will fail at two levels when D tries to authenticate A. First, since A does not possess $Auth_d$, it will not produce an OTP_{d_t} that will match the one calculated by the device. Nevertheless, depending on the truncation of t in the OTP_{d_t} generation, A may attempt to simply forward the one received in the *Auth_Request*(OTP_{d_t}). Thus, the second level of verification is done to ensure that the gateway is indeed trusted, which is done by verifying the $Auth_{eg}$, once A does not possess Id_d, and by (5) a fabricated $Auth_{ea}$ will never match $Auth_{eg}$ produced by D without breaking the security of the AES algorithm, which is resistant to known-plaintext attack. □

Corollary 1 *Messages exchanged in an IIoT segment are resistant to eavesdropping attacks from unauthorized devices.*

Proof Following Theorem 1, an unauthorized device A attacking the network will not be able to obtain K_G while valid, nor will it have success in making a device to use a fabricated K'_G. Thus, even on a non-secure medium, A will not be successful on an eavesdropping attack since it will not be able to obtain the plain-text format of the messages without breaking the encryption mechanism (*AES*). □

Theorem 2 *K_{de} is secure as long as the time frame between a message exchange between the ESA and the device is smaller than the time necessary to perform a brute force attack either at SHA*256, *Poly*1305-*AES*, *and ECDH. Moreover, K_{de} does*

not rely on the confidentiality of $UUID_d$ and fw_d after the end of the device activation window.

Proof Note that, by its construction rule (14), K_{de} is generated as the SHA256 of the combination of Id_d. $Auth_d$, $Auth_{de}$, and OTP_{d_t}. Considering the worst case of assumption 3, an attacker A acquires the information of $UUID_d$ and $fw - d$ as soon as the activation window ends after successful device activation. Using (3), (4), and (13), the attacker successfully derives Id_d. $Auth_d$, $Auth_{de}$. Thus, to obtain K_{de}, A needs to generate an $OTP'_{d_t} = OTP_{d_t}$, and to do so, A requires the key K_{dg}, $Auth_d$, t, and the message m. However, even though A is aware of $Auth_d$ and can perform brute force over the $\oplus$ with an educated guess of the value of t, A is not able to obtain K_{dg} as long as ECDH is secure, and will not obtain m since the *Active* message m is encrypted with K_{dg}. So, to generate $OTP'_{d_t} = OTP_{d_t}$, A would require to break Poly1305-AES security, which will be secure as long as AES is secure [36]. Alternatively, A may attempt to brute-force attack the SHA256 digest that yields K_{de} directly. However, the probability of generating an $Auth_d$, by brute force, that matches a SHA256 digest (4) is equal to $\frac{1}{2}^{256}$. Nevertheless, K_{de} is updated at every message exchange between the ESA and the device (17). In this sense, as long as a message exchange between ESA and device is performed before A can guess by brute-force the exact Poly1305-AES MAC after the generation of the first K_{de}, K_{de} is secure, and we can conclude that K_{de} does not rely solely on the confidentiality of $UUID_d$ and $fw - d$. Finally, a worst-case scenario, in which we assume the gateway to be compromised, will also fail to obtain K_{de}. This case is true since in message *Active*, OTP_{d_t} is sent encrypted with K_{dg}, and thus, this information is only accessible inside the secure-world of the TEE, which by assumption 2, does not contain any malicious code. So, in the worst case, the non-secure world can act by denying access to the messages coming from the IIoT segment, which will trigger an unresponsive behavior during devices' activation, which will require an inspection over the gateway integrity or will be detected by the gateway integrity checking protocol, but no sensitive information will be leaked. □

Corollary 2 *The bootstrap process is resistant to fabricating, breaking down, or modifying messages.*

Proof Following Theorem 1, an unauthorized device A attacking the network will not be able to obtain K_G while valid, nor will it have success in making a device to use a fabricated K'_G. Moreover, every message exchanged in the IIoT segment is supported by a MAC OTP_{d_t}. Thus, as demonstrated in Theorem 2, A will not be able to generate a valid OTP_{d_t} since it cannot break K_{dg}, and by Corollary 1, A will not be able to obtain the plain text of messages exchanged in the IIoT segment. Thus A will not be able to modify or break down messages and still produce a valid MAC. Finally, since A possesses neither K_G nor K_{dg}, it will not be able to produce a valid MAC for a fabricated message, as by decrypting a message using K_G and using a key K_{dg} to generate OTP_{d_t}, a valid device will not obtain the same MAC sent by A, as by

definition, a different key over the same text unlikely produces the same result neither by *AES* (using K_G) nor by Poly1305-AES (using K_{dg}). □

Corollary 3 *After the activation window of a successful activation, the* $UUID_d$ *of the device does not require to be confidential anymore.*

Proof By Theorems 1 and 2, and by design of the proposed bootstrap protocol, none of the accorded encryption keys K_G, K_{dg}, and K_{de} rely on the $UUID_d$ for their renew, and neither does the security of Poly1305-AES MAC OTP_{d_t}. Thus, the leakage of the $UUID_d$ after the activation window of a successful activation does not impact the security of the IIoT segment security. □

5.1 Improving Gateway Security through TEE Properties

The aforementioned security analysis presumes that the gateway authenticated with the ESA is trusted and, thus, will not disrupt security by itself. Nevertheless, this may not always be true since gateways are prone to attacks that can compromise the security of the entire IIoT Segment once sensitive operations, such as marshaling and control, must deal with data and keys in plain text.

Following Theorem 2 and Corollary 1, considering a compromised gateway under *Dolev-Yao's Threat Model* [23, 24, 38], eavesdropping attacks will not be possible in the proposed architecture, as the compromised gateway will not have access to confidential information handled inside the secure world, once they are isolated by TEE capabilities.

Neither will it be able to break down, modify, or create valid messages without breaking the encryption algorithm itself. Following the same ideas presented in Corollary 2, considering the non-secure counterpart of the gateway as the attacker, it will not be able to extract sensitive information from the secure world (i.e., K_G, K_{dg}, and K_{de}).

Re-ordering or replaying messages will also not affect the integrity of the IIoT Segment due to the high-resolution timestamp used by the OTP in each message. Spoofing and initializing new communications will not be possible, as compromised gateways do not have access to the necessary authentication tokens, corroborating the assumptions addressed in Theorems 1 and 2, and Corollary 2, where the gateway secure-sensitive operations are assumed to be free of containing malicious code, which is assured by TEE isolation capabilities.

6 Secure Marshaling with Intel SGX

The SGX is an extension of Intel Instruction Set Architecture (ISA) to enable the secure execution of user-level applications, even in non-trusted platforms. The SGX instruction set provides the developer with an enclave, a reserved memory region inside the platform's memory.

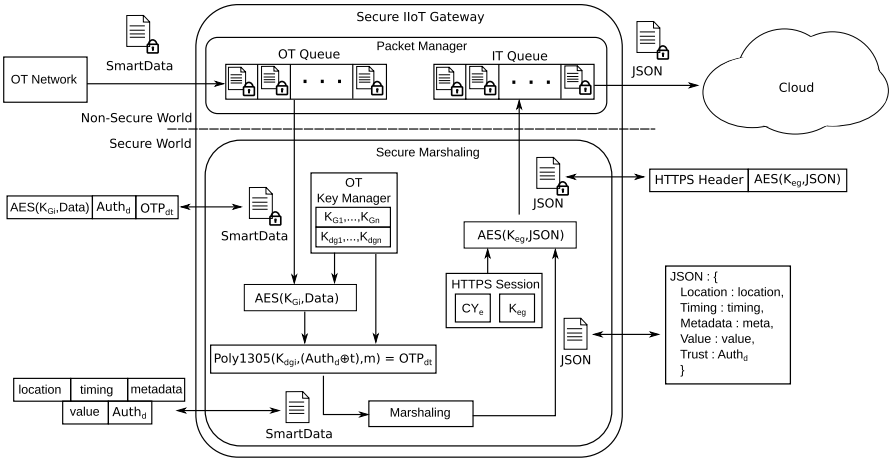

Fig. 5 Marshaling Procedure Prototype. Data from IIoT devices are handled by the Secure World on the marshaling process that outputs a JSON message that is sent to the Cloud

The enclave properties include memory isolation, with the SGX setting aside a memory region called the Processor Reserved Memory (PRM). The CPU protects the PRM from all non-enclave memory accesses. The PRM holds the Enclave Page Cache (EPC), consisting of 4 KB pages that store enclave code and data. The system software, which is untrusted, is responsible for assigning EPC pages to enclaves. The CPU tracks each EPC page's state in the Enclave Page Cache Metadata (EPCM) to ensure that each EPC page belongs to precisely one enclave [39].

This isolation is done by hardware, checking whether a page belongs to the running process, similar to a regular MMU permission checking but with some differences. The isolation works as another layer of hardware, checking if the physical address requested by the CPU is in the memory region reserved for the enclaves and if the requested page belongs to the enclave that requested it. If the requested region does not belong to the enclave, a fault signal is generated, and the process is aborted. A depiction of the verification flowchart is presented below.

Enclave data is only protected from unauthorized access by the hardware that performs the access checks, as data is set in plain text inside the silicon wafer in the CPU caches. In this architecture, the most used data is quickly accessed and does not hinder the performance of a process running in an enclave, but it makes it susceptible to side-channel type attacks [40]. Thus, in SGX, an enclave only contains the private data in computation and the code that operates on it.

The implementation for the proposed secure gateway architecture is depicted in Fig. 5. Messages exchanged in the IIoT are encapsulated as SmartData [41]. A SmartData is a piece of data enriched with enough metadata to make it self-contained in terms of semantics, spatial location, timing, and intrinsic notions of confidence and error. For this case-study, the SmartData are communicated using TSTP [26]. TSTP is an application-oriented, cross-layer protocol for WSN.

TSTP focuses on efficiently delivering functionality recurrently needed by WSN applications: trusted, timed, geo-referenced.

In our case-study implementation, the data sent from the IIoT devices to the gateway are encapsulated in the format described in (16). The non-secure domain counterpart of the gateway encompasses the Packet Manager and Packet Queue. Packet Queue considers Queues for both OT and IT messages. Messages are received encrypted and signed in the non-secure world and are stored at the Packet Queue. The interactions between secure and non-secure domains are handled by the Intel SGX OCALL and ECALL interfaces. While inside the secure world, the message is decrypted using K_G following (16) described in Sect. 4.5.3 and is authenticated using the OTP key (8). Next, the gateway marshals the content from TSTP to HTTPS. The received data and its respective metadata (e.g., timing, location, and trust from the OT protocol) are packed as a JSON message that will be used as the HTTPS POST content.

Next, the session with the Cloud is established and the data is sent through this secure channel using the MBEDTLS library while inside the secure domain application. Here, the IT Queue is directly managed by the MBEDTLS library, which implements OCALLs and ECALLs internally to handle the secure usage of Network I/O.

6.1 Performance Considerations

To investigate the impact of the execution time of utilizing TEE to isolate the marshaling operation, we evaluate the throughput of the marshaling process. The evaluation considers the following operations: packet reception, authentication, data decryption, HTTPS session establishment (once every 100 messages exchanged), and the gateway-Cloud interaction (HTTPS POST).

The chosen evaluation platform is an Intel(R) Core(TM) i7-8750 H processor with support for the Intel SGX extension. The Intel Core i7-8750 H has 6 homogeneous cores with hyperthreading (12 cores total) and up to 4.1GHz clock and 9MB cache. Additionally, the adopted platform has a built-in wireless LAN connectivity through a Qualcomm Atheros QCA6174 802.11ac Wireless Network Adapter The TLS support is obtained through the usage of the MBEDTLS library, and the code is compiled for the SGX by using the Intel SGX SDK.

Figure 6 presents the measured results. The average turn-around time to receive, marshal, and send a message to the cloud through the trusted gateway ranges from 13518.83 μs for 32 bytes of data to 24167.50 μs for a $64KB - 1B$ of data[3], yielding a throughput of 73 and 41 messages per second, respectively.

The turn-around time for the marshaling procedure (i.e., following the platform architecture depicted in Fig. 5, from the *OT Queue* at the non-secure world to the *Secure Marshaling* at the secure world, and then back to the *IT Queue* at the non-secure world) was isolated and is depicted in Fig. 6, ranging from 18.397 μs for 32 bytes of data to 4307.39 μs for $64KB - 1B$ of data. This measure represents

[3] $64KB - 1B$ is the maximum Data length supported by *TSTP* [26].

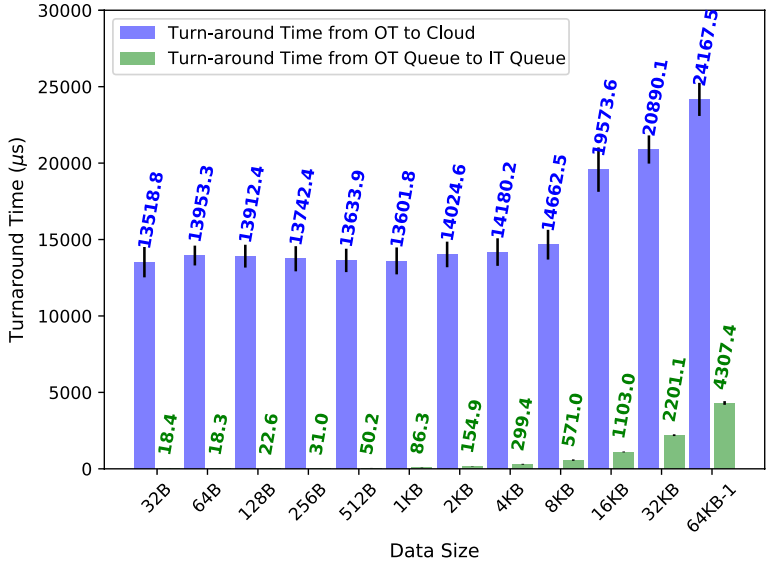

Fig. 6 Average Turn-Around Time measured for the gateway implementation in an Intel SGX. The measurements represent partial and complete marshaling and sending procedures inside the gateway, implemented according to Fig. 5

the low impact of the enclave into the gateway performance, where the marshaling procedure takes on average 17.82% of the complete turn-around time for $64KB - 1B$ data, and 0.1361% for $32B$ data.

The performance of the experiments was measured under the a network connection with an upload speed of $120Mbps$. We did not consider the bootstrapping procedure since it occurs only once for every device. Instead, we focus on the processing overhead introduced by the TEE. Security parameters (e.g., keys and certificates) used for encryption, decryption, and communication were manually configured prior to deployment.

7 Discussion

In this section, we discuss how the proposed IIoT Gateway architecture addresses security and fulfills each of the requirements established in Sect. 3. First, every security-sensitive operation of the gateway is modeled to be handled inside a secure domain enabled by the TEE, which addresses the *Execution and Storage Security* requirement. Consequently, the confidentiality and authenticity of the code and the gateway assets rely on the TEE capabilities of the hardware and memory isolation. This feature, combined with hardware and software replication mechanisms and automatic update capabilities, which can be implemented by taking advantage of the architecture design and secure communication channels, fulfills the requirements of the *IIoT Gateway*. The bootstrap ensures the mutual authentication of the gateway and the ESA using certificates, which are validated with a CA.

A device's bootstrap is only possible through the ESA interaction with the gateway, which satisfies the *IIoT Admission Regulation* requisite. By using the authentication tokens during the bootstrap, no confidential information about the device is provided to the gateway, which enables the establishment of a secure symmetric key between the devices and the ESA and provides support for the *IIoT Integrity Checking* to take place on the IIoT segment. This means that even though the gateway gets compromised, it will not be able to tamper with further interactions between the ESA and the devices, and its integrity can be continuously assured through this channel. Thus, with both IIoT Admission Regulation and IIoT Integrity checking ensured, the *IIoT Auditing* requirement is fulfilled.

The *IIoT independence from Cloud services* is ensured by only requiring an interaction with the Cloud services during the devices bootstrap, which enables the establishment of a trusted relationship between the gateway and the devices through the usage of authentication tokens and a time span. The bootstrap of the devices also incurs into the *Precise Time Synchronization* through PTP, which, combined with Geo-referencing support from the communication protocol, fulfills the *IIoT Devices* requirement.

All messages exchanged between the gateway and the devices are encrypted with a group symmetric key and authenticated with a singular symmetric accorded with each device. The messages exchanged between the ESA and the devices use a symmetric key defined at the end of the device's bootstrap, while the marshaling between the OT and IT domains is handled by the gateway. These operations, along with the key storage, are secured by the TEE isolation capabilities at each entity.

In this way, both *Cross-domain Encryption* and *Strong Authentication* requirements are addressed. As analyzed and proven in Sect. 5 and Sect. 5.1, the proposed architecture and bootstrap procedure is secure against message eavesdropping, messages breaking down, modification and fabrication, message reordering and replay attacks, spoofing and initializing fake communications, even considering scenarios where the gateway becomes compromised.

Furthermore, message blocking is countered by the Integrity checking capabilities granted during the bootstrapping by creating a secure channel between the ESA and the devices in the IIoT segment. Thus, the proposed solution fulfills all requirements of Dove-Yao's Threat Model and, subsequently, the *Secure Communication* requirements. In this sense, after bootstrap, all messages sent by the devices are forwarded to the Cloud, which fits the *IIoT Data on the Cloud* requirement. Thus, the proposed approach fulfills every requirement listed in Sect. 3.

8 Conclusion

This paper presented a Secure IIoT Gateway Architecture based on Commodity Trusted Execution Environment. The envisioned architecture introduces an External Security Agent (ESA) in the IIoT. The ESA regulates the admission of new devices and Gateways, enables authentication tokens for each device without lending to the gateway confidential information, and audits the Gateways by continuously checking the integrity of the data exchanged in the IIoT segment.

By combining consolidated security algorithms, the proposed architecture models the interaction between the OT and the IT worlds. These interactions encompass security bootstrapping with support to authentication, secure communication, secure marshaling, key management, devices rebooting management, and integrity verification. Moreover, every security-sensitive operation is modeled inside a secure domain of execution provided by TEEs. So, by combining the hardware isolation and security provided by the TEE with the consolidated security algorithms in the proposed architecture, every threat considered by *Dolev-Yao's Threat Model* can be countered.

An experiment was conducted considering the execution time of the gateway operations to analyze the impacts of using a Trusted Execution Environment. The TEE implementation used Intel SGX on an Intel(R) Core(TM) i7-8750 H. The experiment considered marshaling from TSTP to HTTPS. The analysis has shown that the TEE operations represented an average of 0.1361% of the total gateway turn-around time for $32B$ data, up to 17.82% for $64KB - 1B$ data, indicating a low impact in the utilization TEEs. Moreover, the total gateway turn-around time varied from 13518.83 μs for 32 bytes of data up to 24167.50 μs for 64KB–1B of data (respecting the limits of TSTP).

Future works include extending the current proof-of-concept to accommodate the entirety of the gateway functionalities in order to obtain a complete analysis of its overhead and verify the security of the proposed gateway architecture against other attack models.

Author contributions AAF: Conceptualization, Writing—Review & Editing, Supervision. LPH: Conceptualization, Formal analysis, Writing—Review & Editing, Investigation. JLCH: Conceptualization, Formal analysis, Writing - Review & Editing, Investigation.

Funding This study was financed in part by grants 2020/05142-1, 2021/02384-7, and 2021/02385-3, São Paulo Research Foundation (FAPESP).

Declarations

Conflict of interest The authors have declared no conflicts of interest.

References

1. Diro, A.A., Chilamkurti, N., Kumar, N.: Lightweight cybersecurity schemes using elliptic curve cryptography in publish-subscribe fog computing. Mobile Netw. Appl. **22**(5), 848–858 (2017). https://doi.org/10.1007/s11036-017-0851-8
2. Cionca, V., Newe, T., Dădârlat, V.T.: Configuration tool for a wireless sensor network integrated security framework. J. Netw. Syst. Manage. **20**(3), 417–452 (2011). https://doi.org/10.1007/s10922-011-9219-8
3. Kolias, C., Kambourakis, G., Stavrou, A., Voas, J.: DDoS in the IoT: Mirai and other botnets. Computer **50**(7), 80–84 (2017). https://doi.org/10.1109/mc.2017.201
4. Lyu, M., Sherratt, D., Sivanathan, A., Gharakheili, H.H., Radford, A., Sivaraman, V.: Quantifying the reflective DDoS attack capability of household IoT devices. In: Proceedings of the 10th ACM

Conference on Security and Privacy in Wireless and Mobile Networks—WiSec '17, pp. 46–51. ACM Press (2017)
5. Bali, R.S., Jaafar, F., Zavarasky, P.: Lightweight authentication for MQTT to improve the security of IoT communication. In: Proceedings of the 3rd International Conference on Cryptography, Security and Privacy. ICCSP '19, pp. 6–12. Association for Computing Machinery, New York, NY (2019)
6. The Things Network.: LoRaWan security, sponsored by The Things Industry. Retrieved from https://www.thethingsnetwork.org/docs/lorawan/security.html. Accessed 03 Nov 2020
7. Naoui, S., Elhdhili, M.E., Saidane, L.A.: Lightweight and secure password based smart home authentication protocol: LSP-SHAP. J. Netw. Syst. Manage. **27**(4), 1020–1042 (2019). https://doi.org/10.1007/s10922-019-09496-x
8. Pinto, S., Gomes, T., Pereira, J., Cabral, J., Tavares, A.: IIoTEED: an enhanced, trusted execution environment for industrial IoT edge devices. IEEE Internet Comput. **21**(1), 40–47 (2017). https://doi.org/10.1109/mic.2017.17
9. Ukil, A., Sen, J., Koilakonda, S.: Embedded security for Internet of Things. In: 2011 2nd National Conference on Emerging Trends and Applications in Computer Science, pp. 1–6. IEEE (2011)
10. Lesjak, C., Hein, D., Winter, J.: Hardware-security technologies for industrial IoT: TrustZone and security controller. In: IECON 2015—41st Annual Conference of the IEEE Industrial Electronics Society. IEEE, p. 2589–2595 (2015)
11. Panchal, A.C., Khadse, V.M., Mahalle, P.N.: Security issues in IIoT: a comprehensive survey of attacks on IIoT and its countermeasures. In: 2018 IEEE Global Conference on Wireless Computing and Networking (GCWCN), pp. 124–130. IEEE (2018)
12. Togay, C., Mutlu, G., Kurtulus, D., Özgür, F.: Secure gateway for the internet of things. Avrupa Bilim ve Teknol. Dergisi (2019). https://doi.org/10.31590/ejosat.524783
13. Navarro-Ortiz, J., Sendra, S., Ameigeiras, P., Lopez-Soler, J.M.: Integration of LoRaWAN and 4G/5G for the industrial internet of things. IEEE Commun. Mag. **56**(2), 60–67 (2018). https://doi.org/10.1109/mcom.2018.1700625
14. Lin, I.C., Hsu, H.H., Cheng, C.Y.: A cloud-based authentication protocol for RFID supply chain systems. J. Netw. Syst. Manage. **23**(4), 978–997 (2015). https://doi.org/10.1007/s10922-014-9329-1
15. Kuo, F.C., Tschofenig, H., Meyer, F., Fu, X.: Comparison studies between pre-shared and public key exchange mechanisms for transport layer security. In: Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications, pp. 1–6. IEEE (2006)
16. Bienhaus, D., Ebner, A., Jäger, L., Rieke, R., Krauß, C.: Secure gate: secure gateways and wireless sensors as enablers for sustainability in production plants. Simul. Model. Pract. Theory **109**, 102282 (2021). https://doi.org/10.1016/j.simpat.2021.102282
17. Sebastian, D.J., Agrawal, U., Tamimi, A., Hahn, A.: DER-TEE: secure distributed energy resource operations through trusted execution environments. IEEE Internet Things J. **6**(4), 6476–6486 (2019). https://doi.org/10.1109/JIOT.2019.2909768
18. Lee, S., Heo, M., Park, K., Kim, B., Hong, J.: Enhancing the security of IoT gateway based on the classification of user security-sensitive data. In: Proceedings of the Conference on Research in Adaptive and Convergent Systems. RACS '19, pp. 241–243. Association for Computing Machinery, New York, NY (2019)
19. Ling, Z., Yan, H., Shao, X., Luo, J., Xu, Y., Pearson, B., et al.: Secure boot, trusted boot and remote attestation for ARM TrustZone-based IoT Nodes. J. Syst. Architect. **119**, 102240 (2021). https://doi.org/10.1016/j.sysarc.2021.102240
20. Tange, K., De Donno, M., Fafoutis, X., Dragoni, N.: A systematic survey of industrial internet of things security: requirements and fog computing opportunities. IEEE Commun. Surv. Tutor. **22**(4), 2489–2520 (2020). https://doi.org/10.1109/COMST.2020.3011208
21. Li, J., Tang, X., Wei, Z., Wang, Y., Chen, W., An Tan, Y.: Correction to: Identity-based multi-recipient public key encryption scheme and its application in IoT. Mobile Netw. Appl. (2020). https://doi.org/10.1007/s11036-020-01512-8
22. Lucena, M., Scheffel, R.M., IoT, Fröhlich. A.A..: Protocol, gateway integrity checking. In: IX Brazilian Symposium on Computing Systems Engineering (SBESC), vol. 2019, pp. 1–8. IEEE (2019)
23. Needham, R.M., Schroeder, M.D.: Using encryption for authentication in large networks of computers. Commun. ACM **21**(12), 993–999 (1978). https://doi.org/10.1145/359657.359659
24. Dolev, D., Yao, A.C.: On the security of public key protocols. In: 22nd Annual Symposium on Foundations of Computer Science (sfcs 1981), pp. 350–357. IEEE (1981)

25. Hu, P., Ning, H., Qiu, T., Song, H., Wang, Y., Yao, X.: Security and privacy preservation scheme of face identification and resolution framework using fog computing in internet of things. IEEE Internet Things J. **4**(5), 1143–1155 (2017). https://doi.org/10.1109/JIOT.2017.2659783
26. Resner, D., Fröhlich, A.A.: Design rationale of a cross-layer, trustful space-time protocol for wireless sensor networks. In: 2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA), pp. 1–8. IEEE (2015)
27. Scheffel, R.M., Fröhlich, A.A.: FT-TSTP: a multi-gateway fully reactive geographical routing protocol to improve WSN reliability. In: 2018 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), pp. 1–6. IEEE (2018)
28. IEEE: IEEE standard for a precision clock synchronization protocol for networked measurement and control systems. In: IEEE Std 1588–2002, pp.1–154, 31 Oct. 2002. https://doi.org/10.1109/IEEESTD.2002.94144
29. Resner, D., Fröhlich, A.A.: Speculative precision time protocol: submicrosecond clock synchronization for the IoT. In: 21st IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2016), pp. 1–8. Berlin, Germany (2016)
30. IEC. Industrial Communication Networks—Fieldbus Specifications—Part 1: Overview and Guidance for the IEC 61158 and IEC 61784 Series. International Electrotechnical Commission, Geneva (2019)
31. Isobe, T., Shibutani, K.: Preimage Attacks on Reduced Tiger and SHA-2. In: Fast Software Encryption, pp. 139–155. Springer, Berlin (2009)
32. National Security Agency: The case for elliptic curve cryptography (2005, October 13). Retrieved from https://web.archive.org/web/20051013062853/http://www.nsa.gov/ia/industry/crypto_elliptic_curve.cfm. Accessed November 3, 2020
33. Resner, D., Augusto, Fröhlich, A.: Key establishment and trustful communication for the Internet of Things. In: Proceedings of the 4th International Conference on Sensor Networks—SENSORNETS,. INSTICC, pp. 197–206. SciTePress (2015)
34. Certicom Research: SEC 2: recommended elliptic curve domain parameters (2010, January 27). Retrieved from https://www.secg.org/sec2-v2.pdf. Accessed November 3, 2020
35. Aziz, B., Hamilton, G.: Detecting man-in-the-middle attacks by precise timing. In: 2009 Third International Conference on Emerging Security Information, Systems and Technologies, pp. 81–86. IEEE (2009)
36. Bernstein, D.J.: The Poly1305-AES message-authentication code. In: Proceedings of Fast Software Encryption, pp. 32–49. Paris, France (2005)
37. Resner, D.: Performance Evaluation of the Trustful Space-Time Protocol [M.Sc. Thesis]. Federal University of Santa Catarina. Florianópolis (2018). https://repositorio.ufsc.br/handle/123456789/189296
38. Carlos, M.C., Martina, J.E., Price, G., Custódio, R.F.: An updated threat model for security ceremonies. In: Proceedings of the 28th Annual ACM Symposium on Applied Computing. SAC '13, pp. 1836–1843. Association for Computing Machinery, New York, NY (2013). https://doi.org/10.1145/2480362.2480705
39. Costan, V., Devadas, S.: Intel SGX explained. IACR Cryptol. ePrint Arch. **2016**, 86 (2016)
40. Götzfried, J., Eckert, M., Schinzel, S., Müller, T.: Cache Attacks on Intel SGX. In: Proceedings of the 10th European Workshop on Systems Security. EuroSec'17, pp. 1–6. Association for Computing Machinery, New York, NY (2017)
41. Fröhlich, A.A.: SmartData: an IoT-ready API for sensor networks. Int. J. Sens. Netw. **28**(3), 202 (2018). https://doi.org/10.1504/ijsnet.2018.096264

Antônio Augusto Fröhlich is a full professor at the UFSC, where leads the Software/Hardware Integration Lab (LISHA) since 2001. With a Ph.D. in Computer Engineering from TU-Berlin, he has coordinated several R&D projects on embedded systems. Significant contributions from these projects materialized within the Brazilian digital television system and IoT technology for energy distribution, smart cities, and autonomous systems. He is a senior member of ACM, IEEE, and SBC.

Leonardo Passig Horstmann is a Ph.D. candidate at Federal University of Santa Catarina (UFSC), where he is a member of the Software/Hardware Integration Lab (LISHA) since 2018. He received M.Sc. degree in Computer Science in 2021, from the UFSC, Brazil, and his research interests include safety-critical systems, mixed-criticality systems, autonomous systems, real-time embedded systems, multicore processors, machine learning, IoT, misbehavior detection, and security protocols.

José Luis Conradi Hoffmann is a Ph.D. student at Federal University of Santa Catarina (UFSC), where he is a member of the Software/Hardware Integration Lab (LISHA) since 2018. He received his M.Sc. degree in Computer Science in 2021, from the UFSC, Brazil. His research interests include data-driven design of critical systems, safety verification, formal nethods, real-time embedded systems, machine learning, IoT, and security protocols.

Authors and Affiliations

Antônio Augusto Fröhlich[1] · Leonardo Passig Horstmann[1] · José Luis Conradi Hoffmann[1]

Antônio Augusto Fröhlich
guto@lisha.ufsc.br

Leonardo Passig Horstmann
horstmann@lisha.ufsc.br

[1] Software/Hardware Integration Lab, Federal University of Santa Catarina, Florianópolis, Santa Catarina 88040-900, Brazil