Computers in Industry 144 (2023) 103802

Contents lists available at ScienceDirect

Computers in Industry

journal homepage: www.sciencedirect.com/journal/computers-in-industry

Improving efficiency and security of IIoT communications using in-network validation of server certificate

Asier Atutxa [a,*,1], Jasone Astorga [a,2], Marc Barcelo [b,3], Aitor Urbieta [b,4], Eduardo Jacob [a,5]

[a] Department of Communications Engineering, University of the Basque Country UPV/EHU, Faculty of Engineering of Bilbao, Plaza Ingeniero Torres Quevedo, n. 1, 48013 Bilbao, Spain
[b] Ikerlan Technology Research Centre, Basque Research and Technology Alliance (BRTA), Po J.M. Arizmendiarrieta 2, Arrasate/Mondragon, 20500 Gipuzkoa, Spain

ARTICLE INFO

Keywords:
DTLS
In-network computing
IoT
P4

ABSTRACT

The use of advanced communications and smart mechanisms in industry is growing rapidly, making cybersecurity a critical aspect. Currently, most industrial communication protocols rely on the Transport Layer Security (TLS) protocol to build their secure version, providing confidentiality, integrity and authentication. In the case of UDP-based communications, frequently used in Industrial Internet of Things (IIoT) scenarios, the counterpart of TLS is Datagram Transport Layer Security (DTLS), which includes some mechanisms to deal with the high unreliability of the transport layer. However, the (D)TLS handshake is a heavy process, specially for resource-deprived IIoT devices and frequently, security is sacrificed in favour of performance. More specifically, the validation of digital certificates is an expensive process from the time and resource consumption point of view. For this reason, digital certificates are not always properly validated by IIoT devices, including the verification of their revocation status; and when it is done, it introduces an important delay in the communications. In this context, this paper presents the design and implementation of an in-network server certificate validation system that offloads this task from the constrained IIoT devices to a resource-richer network element, leveraging data plane programming (DPP). This approach enhances security as it guarantees that a comprehensive server certificate verification is always performed. Additionally, it increases performance as resource-expensive tasks are moved from IIoT devices to a resource-richer network element. Results show that the proposed solution reduces DTLS handshake times by 50–60 %. Furthermore, CPU use in IIoT devices is also reduced, resulting in an energy saving of about 40 % in such devices.

1. Introduction

Industrial systems are more connected than ever. Thanks to information and communication technologies, industrial processes are becoming smarter, more efficient and more sustainable. Industrial Internet of Things (IIoT) devices have emerged as key elements for the pervasive monitoring of any industrial system. These data are then communicated and combined together to gain business-level intelligence and to develop more autonomous processes and decision-making (Liu et al., 2022; Tchoffa et al., 2021. In this context, information and communication technologies provide the grounds to support the timely transmission and processing of gathered data and commands.

Nevertheless, the use of communication networks to transmit critical industrial information increases also the exposure of the industrial systems and processes to information leakage and a wide new range of security attacks Corallo et al. (2022). Additionally, the amount and diversity of IIoT devices are rapidly growing. Although there is no consensus on a general definition for IIoT, usually, they are considered to have small memories and CPUs, and to communicate over low-power and low-bit rate data networks Ojo et al. (2018). These devices are

* Corresponding author.
E-mail addresses: asier.atutxa@ehu.eus (A. Atutxa), jasone.astorga@ehu.eus (J. Astorga), mbarcelo@ikerlan.es (M. Barcelo), aurbieta@ikerlan.es (A. Urbieta), eduardo.jacob@ehu.eus (E. Jacob).
[1] ORCID: 0000-0003-2041-8423.
[2] ORCID: 0000-0002-5532-004X.
[3] ORCID: 0000-0001-5503-074X.
[4] ORCID: 0000-0001-5836-4198.
[5] ORCID: 0000-0001-7093-0586.

https://doi.org/10.1016/j.compind.2022.103802
Received 30 June 2022; Received in revised form 28 September 2022; Accepted 31 October 2022
Available online 13 November 2022
0166-3615/© 2022 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

frequently deployed in a massive way in order to pervasively monitor the manufacturing processes and the environment, and later send gathered data to a cloud or a centralized data storage service. Taking into account the low-performance and massive deployment characteristics of IIoT devices, they are usually designed to be cheap, and in this effort of saving costs, security is sometimes neglected. Additionally, IIoT devices are not usually connected to a keyboard and a display, making it more difficult to update configurations, install patches and updated software versions, etc. By way of an example of the special vulnerability of IoT devices, the latest attacks (Millman, 2021; Muncaster, 2021 have been launched by taking advantage of poorly protected IoT devices.

For these reasons, all of the industrial communication protocols used today (AMQP 1.0, MQTT, XMPP, OPC UA, Modbus TCP, CoAP, etc.) rely on an underlying security layer that guarantees the confidentiality and integrity of the communications. This layer is commonly implemented by the standard and well-known Transport Layer Security (TLS) Rescorla (2018) or Datagram Transport Layer Security (DTLS) Rescorla and Modadugu (2012) protocols, which usually rely on X.509 digital certificates for strong and scalable authentication. DTLS has been designed as an equivalent to TLS, but tailored to the specificities of UDP-based communications frequently used in lossy wireless environments. However, the use of public key cryptography and the management of X.509 digital certificates involve resource-expensive tasks, which are not always affordable for resource-deprived IIoT devices. Even in the case of IIoT devices with higher capacities, the execution of these tasks usually implies important delays, which hinder the real-timeness expected from today's communications.

In this regard, the validation of the server certificate is one of the most costly tasks involved in the establishment of a certificate-based DTLS communication. This validation implies different operations, including, (1) verifying that the certificate is within its validity time-range and has not expired yet; (2) verifying that the certificate is signed by a trusted Certificate Authority (CA); and (3) verifying that the certificate has not been revoked. With respect to this last step, two main mechanisms are currently used to check if a certificate has been revoked before its expiration time. The most common mechanism is considering Certificate Revocation Lists (CRLs) (Request For Comments RFC 5280). A CRL consists of a list of digital certificates issued by a CA which have been revoked before their actual expiration time. However, CRLs are just downloaded periodically and not online for each received certificate. This opens a vulnerability window for certificates revoked after the last CRL download to be accepted as valid. In order to face this security issue, the use of the Online Certificate Status Protocol (OCSP) (RFC 6960) is usually presented as a more robust alternative. In short, OCSP defines a request-response message exchange to allow clients to query CAs about the revocation status of a given certificate. This query is performed online for each and every received certificate. Therefore, OCSP stands as a more secure mechanism compared to the periodic download of CRLs. However, OCSP presents some disadvantages too. Overall, the execution of the OCSP message exchange is more costly than checking a local CRL and it introduces a greater delay in the DTLS negotiation, as it implies communicating with a remote CA.

Taking into account the resource limitations of IIoT devices and the performance overhead introduced by strong security mechanisms, the manufacturers and developers of these devices frequently refuse to implement costly certificate validation mechanisms. Even more so in the case of brokered communication mechanisms, common in industrial contexts, where communications are short-lived but very frequently established. Therefore, the TLS/DTLS session handshake introduces a high overhead per every transmitted message.

In this context, this paper presents an innovative in-network certificate validation system, which is broadly represented in Fig. 1. In short, the proposal consists of pulling out certificate validation operations from resource-deprived IIoT devices to a resource-richer networking device, more specifically, to the IIoT border router that all packets directed to the IIoT network must traverse. For this aim, an approach based on In-Network Computing (INC) is used by leveraging data plane programming (DPP) in the IIoT border router. As the operations involved in certificate validation do not require knowledge of any private or secret key, the proposed solution does not hinder end-to-end security.

Therefore, the main contributions of this paper are the following: .

- An in-network certificate validation system that allows to transparently check the validity of all the certificates received by IIoT devices.
- Increased security level of the protected IIoT devices: as the validation is performed in the edge of the network, it is guaranteed that a reliable validation of all received certificates is actually performed, without depending on the capacities of the end IIoT devices to perform it.
- Resource and time savings: as the certificate validation is performed in a powerful network device, it takes less time to complete it. Additionally, it is no longer necessary for the IIoT device to carry out

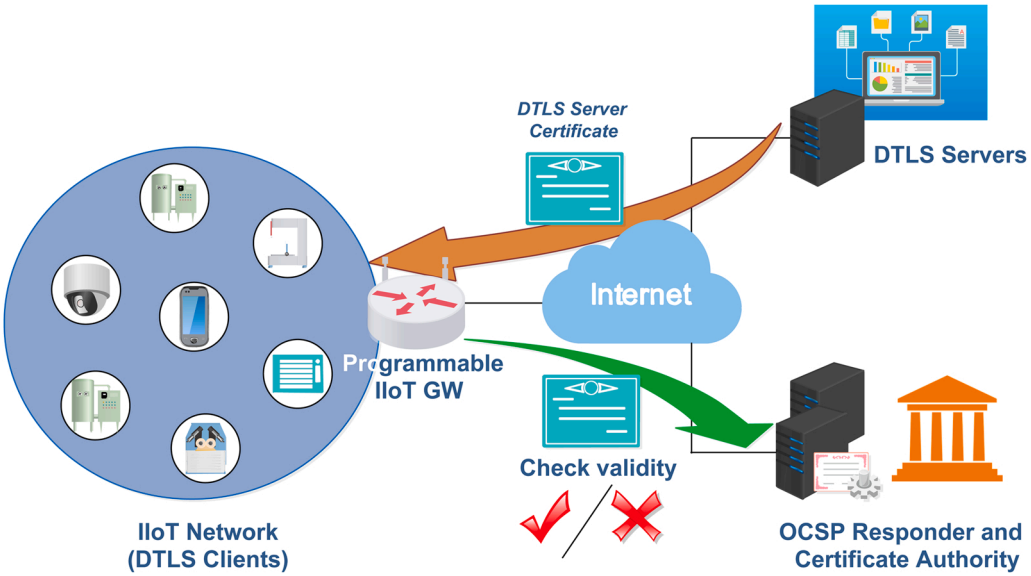


Fig. 1. Considered reference scenario for the proposed system.

the validation, reducing CPU and battery use in the resource-deprived device.
- Performance evaluation of the proposed system by means of a real testbed implementation.

It is also worth mentioning that the proposed system is totally transparent for the communicating endpoints and it does not require any modification or special configuration of the software already running in the protected IIoT devices.

The solution proposed in this work contributes significantly to the state of the art of industrial communication security, specifically in terms of performance, which simplifies the implementation of security protocols while avoiding drawbacks in efficiency and having full compatibility with current standards.

The rest of the paper is organized as follows: Section 2 reviews the related work in the field of IIoT security, and Section 3 analyses the main concepts and technologies related to the work. Then, Section 4 presents the proposed system, and Section 5 describes the implementation and configuration of the testbed for the performance tests. Section 6 shows and interprets the obtained results. Finally, Section 7 summarizes the main conclusions of the paper.

2. Related work

The design of efficient security mechanisms for IIoT devices is a widely studied research topic during the latest years. Some authors propose the use alternative encryption mechanisms such as Attribute-Based Encryption (ABE) Bao et al. (2022) or lightweight cryptography Deebak et al. (2022), while other research works propose to leverage novel technologies such as Distributed Ledger Technologies (DLTs) Xu et al. (2022).

On the other hand, DPP is gaining momentum as an enabler for the efficient implementation of different types of networking solutions in industrial networks. In Chang et al. (2021), the authors propose to use Programming Protocol-independent Packet Processors (P4) Bosshart et al. (2014) to satisfy traffic isolation and performance for network slicing in industrial environments, while the work in Rodriguez et al. (2019) proposes to use DPP to reduce latency when sending a stop command to a robot arm and in Kannan et al. (2019) the aim is to implement the Data-Plane Time-synchronization Protocol (DPTP).

More focused on the implementation of security services, in the latest years, the offloading of security features to the network by means of INC and Software Defined Networking (SDN) is getting increasingly popular and there are multiple proposals to implement access control and firewall functionalities (Almaini et al., 2021; Zaballa et al., 2020; Datta et al., 2018; Ricart-Sanchez et al., 2019; Grigoryan and Liu, 2018; Yousefi et al., 2020.

Data plane programming is also widely used to classify and sample traffic in order to identify malicious traffic frequently caused by Distributed Denial of Service (DDoS) attacks. Then, switching devices are dynamically programmed to block the identified malicious flows as close to the entry point to the network as possible (Zuo et al., 2020a, 2020b; Lapolli et al., 2019; Dimolianis et al., 2020; Kuka et al., 2019; Mahrach et al., 2018.

INC and P4 in particular are also highly attractive solutions to implement Virtual Private Networks (VPNs) and secure tunnels (Hauser et al., 2020; Zuo et al., 2019.

In the recent years, many authors have targeted making the popular DTLS protocol suitable for its execution in highly constrained IoT devices. For this aim, most proposals have focused on delegating the DTLS handshake to a resource-richer trusted third party (Granjal et al., 2013; Hummen et al., 2014; Moosavi et al., 2015; Park and Kang, 2014; Kang et al., 2015; Park et al., 2016; Ma et al., 2020. In this way, the DTLS session is negotiated between a trusted powerful entity and the remote peer, without the intervention of the IoT device. Then, the DTLS session resumption feature is leveraged to convey the session negotiated by the resource-rich entity to the IoT device. From this point on, the DTLS record protocol is executed end-to-end between the IoT device and the remote peer.

These proposals achieve the establishment of a DTLS session between an IoT device and a remote peer in the most efficient way. However, such approaches require the delegation of the IoT device's private key, which implies important security concerns. Additionally, the third party gets to know all the cryptographic material negotiated during the handshake, resulting in privacy concerns associated with the information protected with that material thereafter. Additionally, most proposals assume the existence of a trust relationship and a secure communication channel between the IoT device and the third party, which must be established offline.

In order to avoid all the security issues associated with the delegation of the IoT devices' private keys, some works propose to delegate only those DTLS handshake tasks that can be performed without the knowledge of the corresponding private key (Falk and Fries, 2014; Cho et al., 2019, which result in more secure but less efficient solutions. Some other works (Fouladgar et al., 2006; Marino et al., 2019 aim to find a balance between performance and security depending on the specific application context. For this goal, they propose adaptive solutions in which the network administrator can decide on the specific tasks to be delegated depending on the use case and the characteristics of the involved IoT devices.

An alternative line of work to make DTLS affordable for constrained IoT devices consists of reducing the size of DTLS messages, mainly based on the compression of DTLS headers and X.509 certificates. This is achieved by following the IPv6 over Low power Wireless Personal Area Networks (6LoWPAN) approach to compress DTLS headers and certificates (Raza et al., 2013, 2017; Chavan and Nighot, 2016, and by replacing human-readable Abstract Syntax Notation One (ASN.1)) codification with a more Concise Binary Object Representation (CBOR) (Schukat and Cortijo, 2015; Kwon and Raza, 2018; Kwon and Ahn, 2019; Höglund et al., 2020.

In general, all these proposals allow to reduce the energy consumption of IoT devices, as they reduce the amount of time devices spend in reception/transmission mode. However, these solutions imply that IoT devices no longer use standard message and certificate formats, which can result in compatibility issues. In fact, they are fully dependent on the format conversion mechanism, usually performed in the IoT network gateway. Besides, this conversion is executed by means of tailored software which frequently is not properly updated and maintained. Therefore, the solution presented in this paper provides an efficient security mechanism leveraging function offloading in the edge of the network while complying with the DTLS protocol standard, which to our knowledge, has not been done to date.

3. Fundamental technologies

3.1. DTLS

DTLS provides the same security features as the well-known TLS, but adapted to the characteristics of non-reliable datagram-based communications, such as UDP, Datagram Congestion Control Protocol (DCCP), etc. Therefore, DTLS is similar to TLS, but it has to solve some problems inherent to datagram-based communications such as packet losses and out-of-order arrival of packets. In order to achieve these goals, DTLS implements packet retransmission and repetition detection mechanisms.

In order to deal with packet losses, DTLS implements a mechanism based on timers and retransmissions if the corresponding response has not been received before the timer expires. With respect to reordering, this issue is addressed by the use of sequence numbers and maintaining sequence-related states in the communicating peers. When one of the peers receives a message, compares the sequence number within the message with the expected sequence number. If the received message is the next message expected by the peer, it processes it. If it is a future

message, it stores it for later processing, once all previous messages have been received.

DTLS follows the layered approach of TLS with two main sub-protocols: DTLS handshake and DTLS record. The handshake protocol is executed whenever a new connection is established, and it is responsible for authenticating the communicating peers and negotiating the cryptographic suites and keys that will be used afterwards by the record protocol to protect the actual application layer traffic. The record protocol is based on using symmetric-key cryptography to encrypt and add a Message Authentication Code (MAC) to each message, in order to protect the confidentiality and integrity of data coming from the application layer.

As shown in Fig. 2, the DTLS handshake consists of six flights of messages. Initially, the DTLS handshake is started by the client with a *ClientHello* message, in which it specifies the list of cryptographic suites supported by the client. When receiving this message, the server responds with a *HelloVerifyRequest* message, which contains a fresh cookie used to avoid DDoS attacks with spoofed IP addresses, by forcing the client to resend the initial *ClientHello* message embedding the cookie received from the server. Once the server receives the second *ClientHello* message, it checks the validity of the received cookie and in a successful case, it responds back with a *ServerHello* message, which specifies the selected cipher suite.

The following flight of messages depends on the selected authentication mechanism. DTLS supports three different authentication mechanisms: Pre-Shared Keys (PSK), Raw Public Keys (RPK) and digital certificates. The preferred one among the three is the use of digital certificates, since it provides the greatest scalability and robustness. When digital certificates are used for peer authentication, the *ServerHello* message is followed by a message containing the server's digital certificate. After the *Certificate* message, the *ServerKeyExchange* message is sent, which contains authenticated information about the key exchange. Optionally, the server might also request a certificate from the client by means of a client's *CertificateRequest* message. Finally, the server sends a *ServerHelloDone* message to end the flight of messages and to confirm the security of the handshake and the encrypted channel.

When the client receives these messages, it first verifies the server certificate by checking that the certificate is within its validity period, the CA signature is correct and the certificate has not been revoked. In order to manage the revocation of digital certificates, each CA maintains a list of the certificates it has issued and that have been revoked before their expiration time. This list contains a timestamp with the last time it

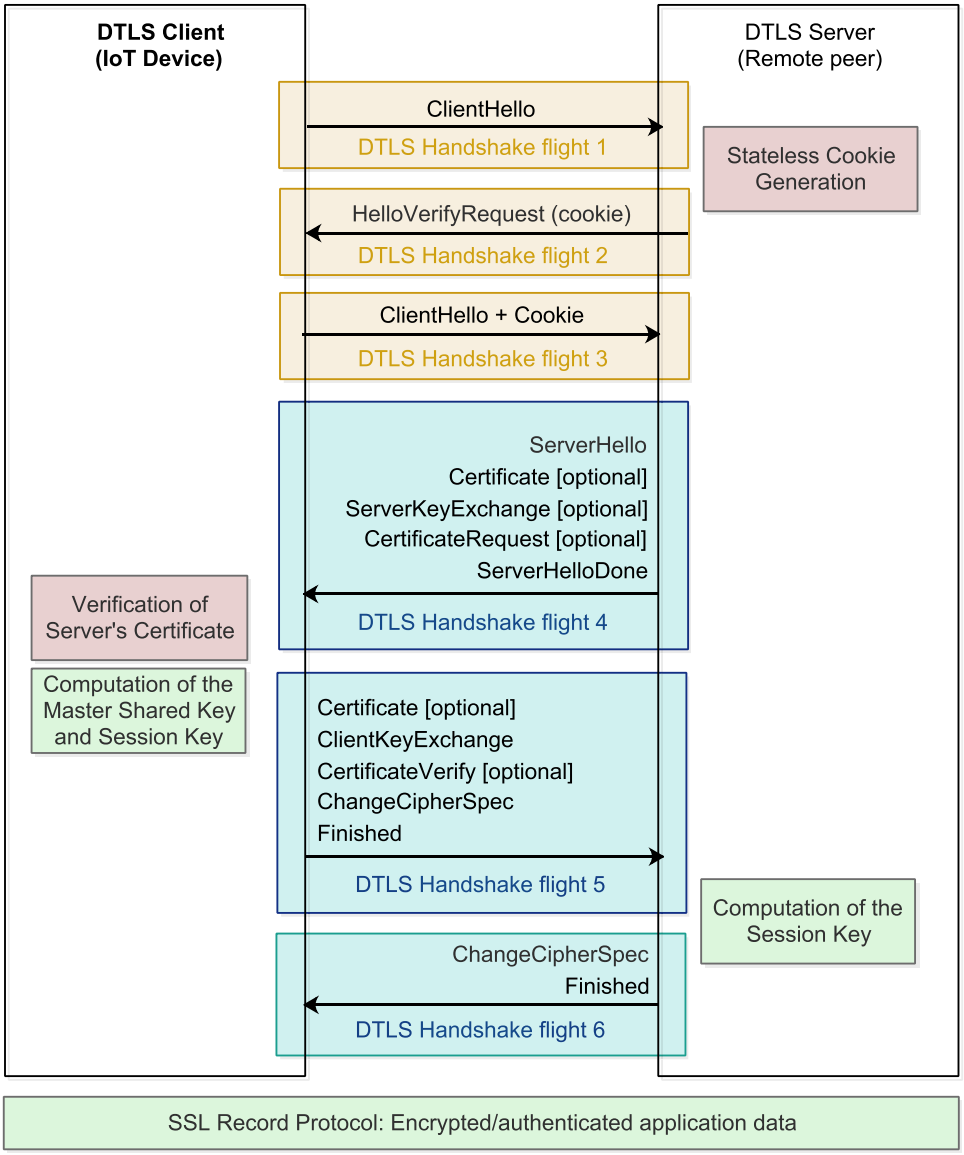


Fig. 2. DTLS 1.2 Handshake message exchange.

has been modified and the serial numbers of all revoked certificates. Then, this list is signed by the CA. In order to allow clients to check if a certificate has been revoked, there are two main mechanisms available: CRLs and OCSP. The use of CRLs is based on clients downloading the list of revoked certificates from each CA at specific time intervals. However, this could prove to be a heavy process for resource-constrained devices and not completely accurate for recently revoked certificates. In contrast, OCSP is a much more robust mechanism that performs online requests for each certificate. The request includes the serial number of the certificate to be validated and the response specifies if the status of this certificate is good, revoked or unknown. In order to guarantee the integrity of this response message, it is signed by the CA who issued the certificate or by a trusted responder.

If the received server certificate is valid, the client uses the server's public key included in the certificate to verify the authenticity of the signed *ServerKeyExchange* message and proceeds to compute the Master Shared Key that will be used afterwards by the record protocol to protect the application data by means of symmetric-key encryption.

If a client certificate was requested in the previous flight of messages sent by the server, the client proceeds to send its own certificate. Then, depending on the key exchange method agreed on the initial *ClientHello* and *ServerHello* message exchange, the client computes a pre-master secret. It encrypts this secret with the server's public key and sends it to the server through the *ClientKeyExchange* message. Then, the client sends a *ChangeCipherSpec* message to let the server know that the following messages will be encrypted with the computed session key. To end the flight of messages, the client sends a *Finished* message, to inform the server that the handshake is finished from the client-side. This last message is already encrypted with the negotiated session key. Eventually, the server decrypts the pre-master secret and computes the session key. At this point, the server sends its *ChangeCipherSpec* message, telling the client that it will encrypt the following messages with the computed session key. Finally, the server sends its *Finished* message encrypted with the computed session key.

Once the DTLS handshake is finished, the DTLS record protocol starts protecting the application layer information exchanged by both peers. More specifically, the computed session keys are used to encrypt application layer data and authenticate it by means of MAC codes.

3.2. In-network computing and programmable networks

In-Network Computing (INC), also known as in-network computation or netcompute, is an emerging research area that consists of offloading computations usually performed at the endpoints, to the networking devices. Among the benefits of INC, three stand out from the rest: (1) the possibility of providing shorter response times, when INC is used to generate responses in-network, before the actual messages reach the corresponding endpoints Atutxa et al. (2021); (2) the reduction of network traffic, when INC is used to filter or aggregate messages (Wang et al., 2020; Madureira et al., 2020; (3) and the reduction of the workload to be performed by the endpoint devices (Cesen et al., 2020; Kunze et al., 2021. This last advantage is specially critical in the cases where endpoints are resource-deprived devices that communicate over wireless, low-bandwidth networks with high packet loss rates. This is the case of IIoT devices communicating over Low-Power Wireless Area Networks (LoWPANs).

Both Smart Network Interface Cards (SmartNICs) and programmable switch-ASICs (application-specific integrated circuits) have been fundamental for INC, allowing network administrators to program their data planes by means of high-level languages, such as P4. P4 stands for Programming Protocol-independent Packet Processors, and it implements a domain-specific language that specifies how the data plane of networking devices processes packets. P4 is target independent, which means that it does not depend on specific hardware or low-level placement details, as long as the data plane is programmable. A typical example of a programmable switch used for testing and development is the Behavioural Model version 2 (BMv2) software switch Anon (2022).

The P4 architecture consists of the following elements, which are graphically depicted in Fig. 3:

- Parser: the parser defines a state machine where the programmer specifies the packet format accepted by the switch in its packet processing pipeline. Typically, there is one state for each different protocol accepted by the switch. In each state, the switch extracts information from the packet and then, it makes a transition decision based on that information.
- Ingress/Egress Match-Action Pipelines: these programmable match-action pipelines are used to implement packet processing algorithms. They take as input the extracted packet headers and run a sequence of Match-Action tables, where most typical actions include inserting, removing and modifying headers, dropping/cloning packets, forwarding packets to specific ports, etc. It is worth noting that from one Match-Action to the other metadata can be retained. Metadata consists of some extra fields included in the packet which are useful to determine further packet processing.
- Traffic queue manager: this component executes packet buffering and replication actions, in order to control the packet flow between the ingress and egress phases throughout the switch.
- Deparser: the deparser takes the previously modified headers, turns those headers back into a packet stream and appends the corresponding payload.

4. Proposed system

As already mentioned, the system proposed in this paper aims at offloading time- and resource-consuming tasks of the DTLS handshake from the IIoT devices to the network. Specifically, the selected tasks to be performed in the network are: (1) checking that the certificate has not expired, (2) validation of certificate signature, and (3) verification of the certificate revocation status by means of OCSP. We have selected these three tasks because they do not require knowledge of any private or secret key, and they can be performed in-network in a transparent way for the communicating endpoints.

The proposed solution allows enhancing the DTLS communication in two critical aspects: (1) first of all, the security of the communication is enhanced because it is guaranteed that all security validations are performed, including the verification of the certificate revocation status using OCSP. It is worth noting that this latter verification is frequently omitted by IIoT devices, which operate over low power and low bandwidth wireless networks, where message transmissions are minimized.

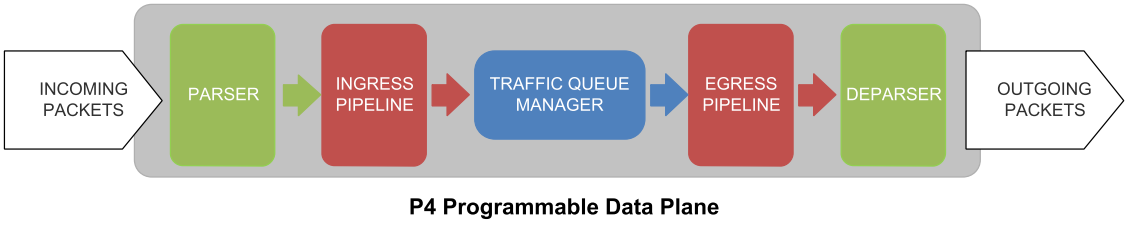


Fig. 3. Elements of the P4 architecture.

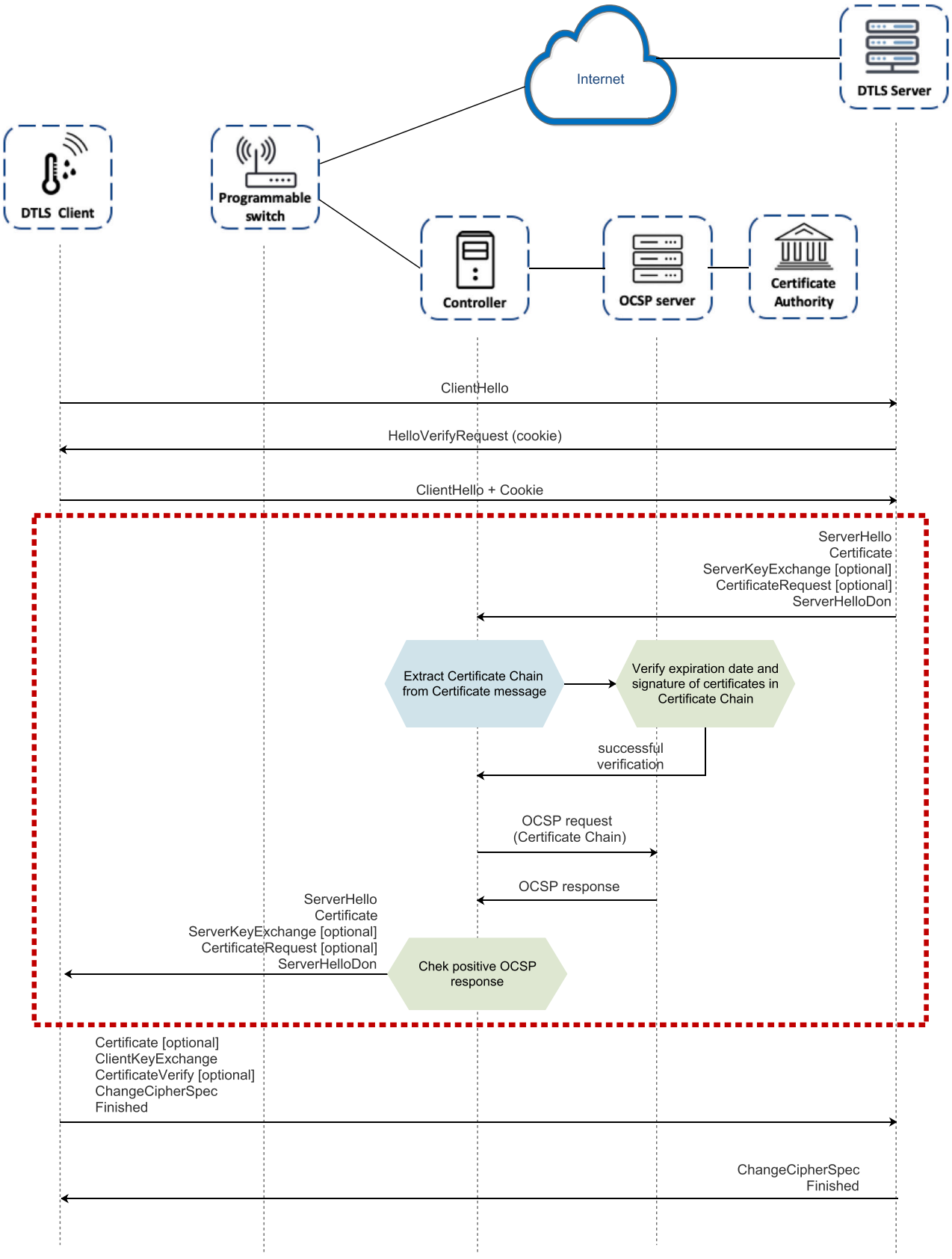
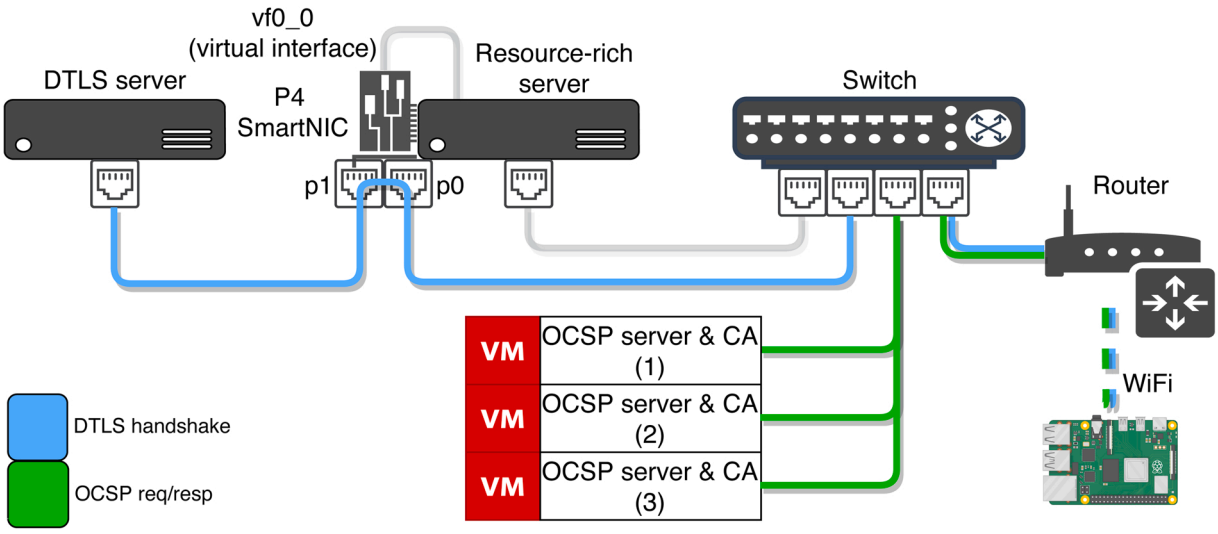
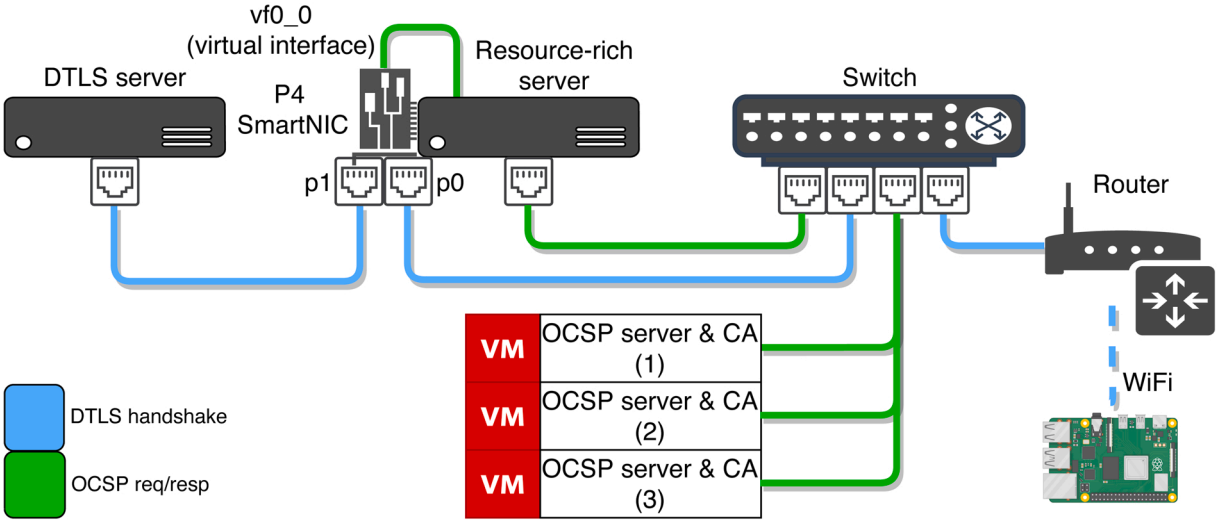


Fig. 4. Considered reference scenario for the proposed system.



(a) Conventional scenario, IIoT device performs the DTLS handshake, server certificate validation and OCSP verification.



(b) INC-aided scenario, IIoT device performs the DTLS handshake and the resource-rich server performs the server certificate validation and OCSP verification.

Fig. 5. Comparison of both architectures and the communication flows involved in the DTLS handshake and OCSP requests in each case (a) Conventional scenario, IIoT device performs the DTLS handshake, server certificate validation and OCSP verification. (b) INC-aided scenario, IIoT device performs the DTLS handshake and the resource-rich server performs the server certificate validation and OCSP verification.

(2) On the other hand, the performance of the DTLS handshake is enhanced since performing resource-intensive operations in an IIoT device with limited CPU and memory is more costly than performing these operations on a resource-rich device with a broadband connection.

The the proposed INC-aided mechanism is depicted in Fig. 4. Following the DTLS handshake message exchange explained in Section 3.1, the IIoT client starts the connection by sending a *ClientHello* message. The server responds with a *HelloVerifyRequest* message including a fresh cookie and the client replies with a new *ClientHello* message, this time including the cookie just received from the server.

Next, the server proceeds to send the *ServerHello*, *Certificate*, *ServerKeyExchange* and *ServerHelloDone* application messages. All these messages travel in a single message flight, but can be fragmented in multiple datagrams depending on the MTU of the network and the size of the *Certificate* message, which conveys the certificate chain. The programmable switch scans all the traffic that traverses it, and it is capable of detecting the specific message that carries the server certificate by executing the ingress Algorithm 1. It detects DTLS messages and checks the type field for all the different application messages of the flight sent by the server, as it can be fragmented at any point and resulting datagrams can start with any of the four mentioned application messages.

If the analyzed datagram corresponds to the targeted flight, it is forwarded to the controller for further processing. On the contrary, if it is not part of the flight, the datagram is forwarded as usual.

Regarding the complexity of Algorithm 1, all the actions are executed once and considered to require constant time. However, searching for the output port of the switch corresponding to the client is considered to be O(n), as the number of executions is proportional to the size of the address list, so the total complexity of the algorithm is defined as O(n).

Algorithm 1. Ingress processing of the packet in the programmable switch.

```
Input: Packet
Output: Forward Decision
if hdr.dtls.sh == isValid then
    if ingress port == controller then
        egressPort ← controllerPort
    else
        if header.dtls.sh == 0x02 ||
          header.dtls.sh == 0x1B ||
          header.dtls.sh == 0x1C ||
          header.dtls.sh == 0x1E then
            egressPort ← controllerPort
        else
            if header.ethernet.dstAddr ==
              clientAddr then
                egressPort ← clientPort
            else
                egressPort ← serverPort
else
    if header.ethernet.dstAddr == clientAddr
      then
        egressPort ← clientPort
    else
        egressPort ← serverPort
```

Algorithm 2. Server certificate verification in the controller (CertificateVerify).

```
Input: Cert_S, CertChainList =
       [Cert_CA1, ..., Cert_CAn], i
Output: K+
if i == 0 then
    Cert ← Cert_S
    CA == CAi + 1
else
    Cert ← Cert_CAi
    CA == CAi + 1
if curTime < Cert.notBefore &&
  curTime > Cert.notAfter then
    K+ = NULL
else
    if !exists(K+_CA) then
        K+_CA ←
          CertificateVerify(Cert_CA, i + 1)
    if K+_CA == NULL then
        K+ = NULL
    else
        if H(Cert.payload)! =
          K+_CA{Cert.signature} then
            K+ = NULL
        else
            revocationStatus ←
              OCSPquery(Cert)
            if revocationStatus! = good then
                K+ = NULL
            else
                K+ ← Cert.PubKey
```

After executing Algorithm 1 in the switch, the whole message is forwarded to the control plane, instead of continuing its path towards the client. At the controller level, the server certificate ($Cert_S$) is extracted and its validity is verified. To this end, three verifications are performed, following the process defined in Algorithm 2. This algorithm returns the server's public key if the verification is successful, and NULL otherwise. First of all, the controller verifies that the server certificate is within its validity period and has not expired yet. This verification is performed first as it is the less costly one from the communication and computing point of view. In this way, it is possible to quickly detect messages containing expired certificates, without further processing.

If the certificate is within its validity period, the process continues to verify the certificate signature. In order to perform this verification, it is necessary to obtain in a reliable way the public key of the CA that signed the certificate (K^+_{CA}). The controller might already own locally this public key or it might need to get it from the CA's certificate. In this last case, the controller must verify the validity of the CA's certificate ($Cert_{CA}$), following the same procedure used for the verification of the server certificate. In the same way, when verifying the CA's certificate it might be necessary to verify also the parent CA's certificate and in the end, to verify recursively the whole certificate chain, as detailed in

Table 1
Mapping between KPIs, testing scenarios and measurement mechanisms.

KPI	Testing Scenario		Measurement Mechanism
	Baseline Scenario	INC-enabled certificate validation	
DTLS handshake delay (ms)	✓	✓	Subtraction of timestamp values measured at the IIoT DTLS client: time when Finished message is received minus time when ClientHello message is sent
Certificate processing delay (ms)	✓	✓	Subtraction of timestamp values measured at the device where certificate verification is performed (IoT device or controller depending on the test scenario): time when certificate validation ends minus time when Certificate message is received
OCSP exchange delay (ms)	✓	✓	Subtraction of timestamp values measured at the device where certificate verification is performed (IoT device or controller depending on the test scenario): time when OCSP response is received minus time when OCSP request is sent
Controller processing time (ms)		✓	Subtraction of timestamp values measured at the controller: time when server Certificate message is forwarded to DTLS client minus time when server Certificate message is received at the P4 switch
Energy consumption	✓	✓	Measurement of energy and electric charge used by the IIoT device (Raspberry Pi 3 Model B) during the tests, measured in mWh. The process includes 100 repetitions of a DTLS handshake in each certain test scenario.
CPU use (%)	✓	✓	Measurement of CPU using an external Linux "time" package, which provides the percentage of the CPU that a job got, divided in user, system and total. The process includes 100 repetitions of a DTLS handshake in each certain test scenario.

Algorithm 2.

The complexity of Algorithm 2 is defined in a similar way to the previous one. In this case, the computation that determines the complexity of the algorithm is the validation of the signature, which has a different complexity depending on the cryptosystem in use. For example, for RSA (Rivest, Shamir and Adleman) implementations, as signature validations make use of the public key, it is assumed to require $O(n^2)$. The other actions involved in this algorithm have a proportional complexity to the size of the key, so are considered to require linear time O(n). Therefore, the choice of cryptosystem has a significant impact on the time complexity of the complete alorithm.

Depending on the result of the server certificate verification, the controller acts in the following way: if the verification was successful, the controller sends to the data plane of the switch the same *Certificate* message initially obtained from it. The ingress at the data plane identifies that the message comes from the control plane and not from the port where the server is accessible, and therefore, it proceeds to forward the message to the client, so that the DTLS handshake can continue normally. If on the contrary, the verification of the server certificate fails, the controller creates a DTLS alert message specifying the code *bad_certificate* and sends it to the IIoT client. In this way, the DTLS handshake fails and the DTLS session with the offending server is not established.

5. Testbed implementation

This section explains the testbed used to demonstrate and evaluate the proposed solution. The testbed has been designed to reliably resemble an actual industrial deployment, in order to obtain truthful and close to reality results.

The implemented testbed is graphically depicted in Fig. 5, where Fig. 5a depicts the baseline scenario, where a DTLS client and server perform a conventional DTLS handshake, and Fig. 5b shows the operation of the proposed INC-aided DTLS handshake in the same testbed. In both cases, the client and the server are the two main hosts in the network, in order to enable a communication including a DTLS handshake. The client is deployed on a Raspberry Pi 3 Model B, which is a representative example of a device with scarce resources, operating in an industrial environment. This version, Model 3B, has an ARM Cortex-A53 (v7 architecture) quad-core 1.2 GHz 64-bit CPU, 1 GB RAM and several connectivity options, with Raspbian OS version 11. Two interfaces are used in this device, one IEEE 802.11 WiFi and one IEEE 802.3 Ethernet. The WiFi interface is used for the communication involved in the tests, to be as close to a typical IIoT environment as possible. When it comes to the Ethernet interface, it is used for remote control and management purposes. The DTLS server is implemented in a high-performance server in our local network, with an 8 core Intel Xeon D-214NT CPU @ 2.30 GHz, 32 GB RAM, Ubuntu 20.04.3 LTS, and 10 Gbps Ethernet interfaces.

Another server is used for the deployment of the programmable network device, which has an octa-core Intel Xeon D-1541 CPU @ 2.10 GHz and 64 GB RAM, using Ubuntu 18.04.5 LTS. The programmable device is implemented through PCIe using a Netronome NFP-4000 SmartNIC with 2×10 Gbps Ethernet interfaces. This NIC is chosen mainly due to the flexibility and fast implementation it provides. In fact, instead of deploying a full switch, a NIC allows the definition of a programmable data plane in a network card, communicated through virtual interfaces to the control plane located in the host server.

The whole structure is implemented in hardware equipment and connected through physical links, except for the servers that implement the CA hierarchy, which are deployed as Virtual Machines (VMs) in an OpenStack node. Each of these VMs contains the CA application and the OCSP responder. The three VMs have been created with the same resources, 8 virtual CPUs and 16 GB RAM working with Ubuntu 20.04.1 LTS, and they are connected to the testbed network through a 10 Gbps Ethernet interface. Apart from these VMs, the rest of the testbed is implemented in hardware equipment in order to obtain more relevant results.

Our testbed considers two different operational scenarios, shown in Fig. 5: (1) Fig. 5a depicts the normal DTLS handshake operation between the client and the server, and (2) Fig. 5b shows the P4-enhanced DTLS handshake using DPP and an intermediate entity (controller). In the first scenario, the client and the server are connected using static network devices, more specifically a wireless access point and a layer 2 switch. However, the second scenario leverages a third component between the DTLS server and the layer 2 switch, which consists of the aforementioned controller server with the programmable NIC, where the in-network validation of the server certificate is performed.

5.1. Definition of KPIs and testing methodology

This section describes the Key Performance Indicators (KPIs) that have been selected to assess the performance of the proposed system. The baseline performance corresponds to the traditional operation of the DTLS handshake where all certificate validations are carried out by the IIoT device acting as the DTLS client. The selected KPIs are the following:

- DTLS handshake time (ms): it denotes the time taken for the secure session establishment between the DTLS client and server. That is, the time spent since the DTLS client sends the first ClientHello message until it receives the last Finished message sent by the DTLS

server. However, as the DTLS handshake is a complex process with multiple interactions, we have defined specific KPIs for the different phases of the process, in order to have a better understanding of the aspects that have a greater contribution to the overall handshake time:

- Certificate processing time (ms): this parameter stands for the time elapsed since the server Certificate message is received at the client or the controller, until the certificate format, validity period and signature are verified.
- OCSP exchange time (ms): this parameter measures the time elapsed since the client or controller sends the OCSP request message, until the corresponding response is received from the OCSP responder.
- Controller processing time (ms): this parameter represents the time that each Server Certificate message spends at the intermediate P4 SmartNIC and controller. It gathers the certificate processing time, the OCSP exchange time, the message processing time at the P4 SmartNIC and the message transmission time between the P4 SmartNIC and the controller. Given the nature of this parameter, it is only measured when the proposed INC system is used, as in the baseline architecture no P4 SmartNIC and controller exists.

- Energy consumption: it represents the energy consumed by the DTLS client IIoT device for the execution of the whole DTLS handshake exchange.
- CPU used: percentage of the CPU used by the DTLS handshake process in the IIoT device.

Table 1 summarizes the different situations in which each of the aforementioned KPIs are measured, along with the mechanism used to measure it:

6. Obtained results and discussion

First, we measure the different times incurred by the process and we evaluate the impact of two parameters in these times: (1) the signature algorithm used by the CA to sign the server certificate, more specifically assessing the impact of RSA and Elliptic Curve Digital Signature Algorithm (ECDSA) with NIST-based curves; and (2) the length of the certificate chain conveyed in the *Certificate* message.

Fig. 6 shows the results obtained for the different time KPIs when different values of the certificate signing algorithm are used. Four security levels have been defined, which correspond to the length of the signatures that have been tested. Shorter signatures are considered as a lower security level, while larger signatures are more secure. The mapping for each level and length for RSA and Elliptic Curve Cryptography (ECC) is the following:

- Security Level 1: RSA 2048 and ECC 224 bits.
- Security Level 2: RSA 3072 and ECC 256 bits.
- Security Level 3: RSA 7680 and ECC 384 bits.
- Security Level 4: RSA 8192 and ECC 521 bits.

In this context, the green and blue bars represent the INC-aided and conventional methods for RSA algorithm, respectively, whereas the red and magenta bars show the equivalent for ECC-based signature schemes. When it comes to patterns and shades, each one indicates the time needed for a different sub-process:

- **The darkest shade** (or solid color) indicates the time since the DTLS packet carrying the server's certificate is received, until the certificate is extracted and ready to send the OCSP request. This is called the *processing time*.

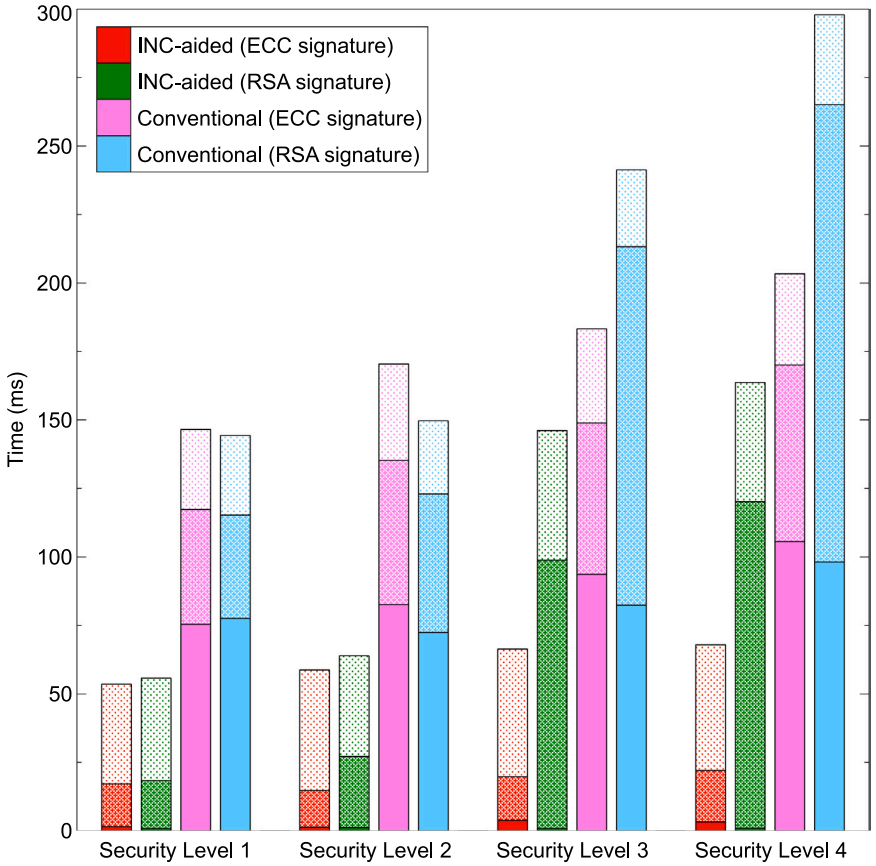


Fig. 6. Assessment of the impact of the RSA-based CA signature algorithms on process times (Server Public Key: RSA 2048, Certificate Chain Lenght: 2).

- **The intermediate shade** shows the required time for the **OCSP query**, since the request is sent to the CA and until the response is received.
- **The lightest shade** comprehends all the other sub-processes involved in the DTLS handshake, such as the transmission times.

In all cases, the values of the server public key and the certificate chain length are kept constant (server public key is RSA-2048 and certificate chain length is 2).

The duration of the whole DTLS handshake varies between 150 ms and 300 ms when RSA keys are used for signature and all the processes are performed by the IIoT device; instead, when ECC keys are used, the DTLS handshake varies between 150 ms and 200 ms. This performance difference corresponds to the fact that ECC cryptography is based on the algebraic structure of elliptic curves over finite fields, while RSA is based on the prime factorization method. As a result, ECC cryptography allows achieving the same security level as RSA with shorter keys. This also impacts scalability, specially when large RSA keys must be used.

The use of the proposed mechanism where certificate validations are performed at the controller allows to significantly reduce these times between 50% and 65% depending on the specific case. It can also be seen that the main contributor to the overall DTLS handshake time is the OCSP exchange, as it implies communicating with the OCSP responder. It is also noticeable the significant reduction of the certificate processing time when this task is performed by the controller. In fact, when this process is performed by the controller, the incurred time is almost negligible. In contrast, when this task is performed by the IIoT device, the incurred time is greater in many cases than the whole DTLS handshake time when the proposed system is used.

Therefore, regarding the execution time, we can conclude that the INC-aided solution clearly improves the operation of the DTLS handshake. Besides, due to the superior performance of the mathematical structure of elliptic curves against RSA, the INC-aided ECC scheme has been demonstrated to be the fastest solution among the presented four.

Finally, in Fig. 6, a big difference can be seen between using 3072-bit and 7680-bit RSA keys. The reason for this effect is that the use of 7680-bit RSA keys implies the necessity of additional packet fragmentation, which in turn derives in a considerable increase of the times that imply packet transmissions.

In Fig. 7, the impact of the certificate chain length on the different measured times has been assessed. The green and blue bars consider the situation where certificates are signed with an RSA 8192 bit key, while red and magenta bars consider the case where 521 bit ECC keys are used. Both types of keys provide a similar security level, and the different shades of each bar indicate the same measured sub-process times as the previous figure: darkest shade processing time, intermediate shade OCSP query time, and lightest shade encompasses the rest. We have evaluated realistic situations in which the certificate chain consists of 2 certificates (server certificate and root CA certificate), 3 certificates (server certificate, intermediate CA certificate and root CA certificate), and 4 certificates (server certificate, two intermediate CA certificates and root CA certificate). The length of the conveyed certificate chain length has a direct impact on the length of the sent message, which translates into an almost linear increase of the DTLS handshake time with the certificate chain length. As it can be seen, the use of our proposed in-network certificate validation system allows to substantially reduce all evaluated times for all the considered cases, between 59 % and 68 % for ECC-based signatures and between 28 % and 45 % for RSA-based signatures. The difference between RSA and ECC signatures is again significant, for the same reason explained before. That is, the mathematical foundations that lay under both cryptographic constructions require that RSA signatures use longer keys to achieve the same security level as ECC signatures. Therefore, the certificate length will also be smaller for ECC than RSA. This issue has also an impact on

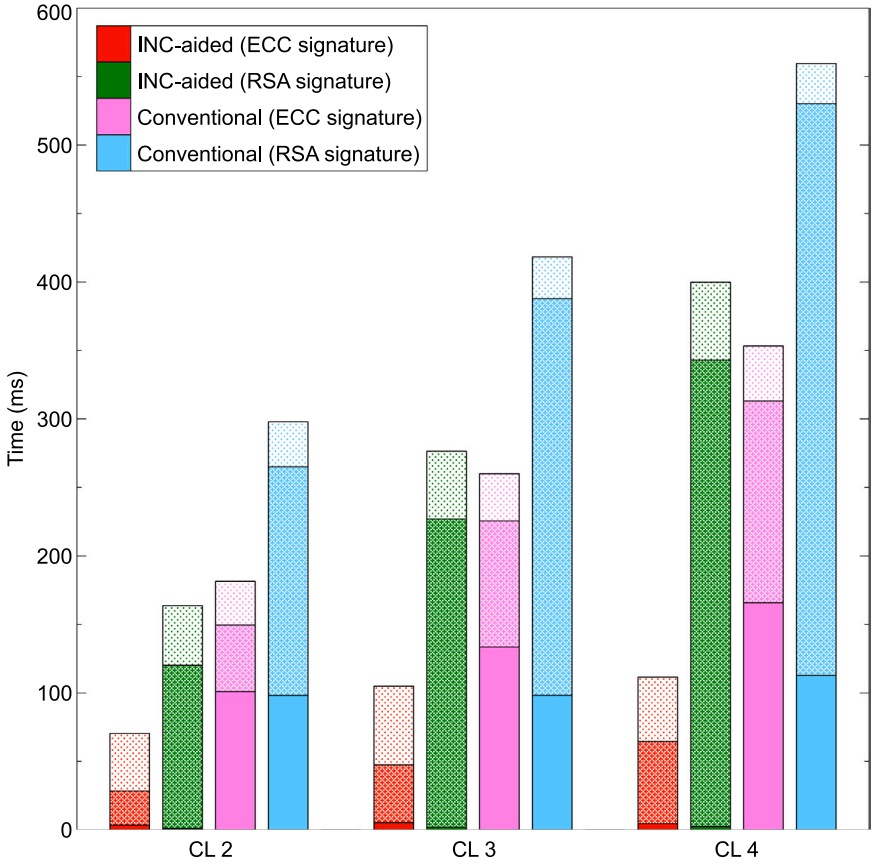


Fig. 7. Assessment of the impact of the Chain Length on process times (Server Public Key: secp256r1/RSA8192, CA Signature Algorithm: secp521r1/RSA8192).

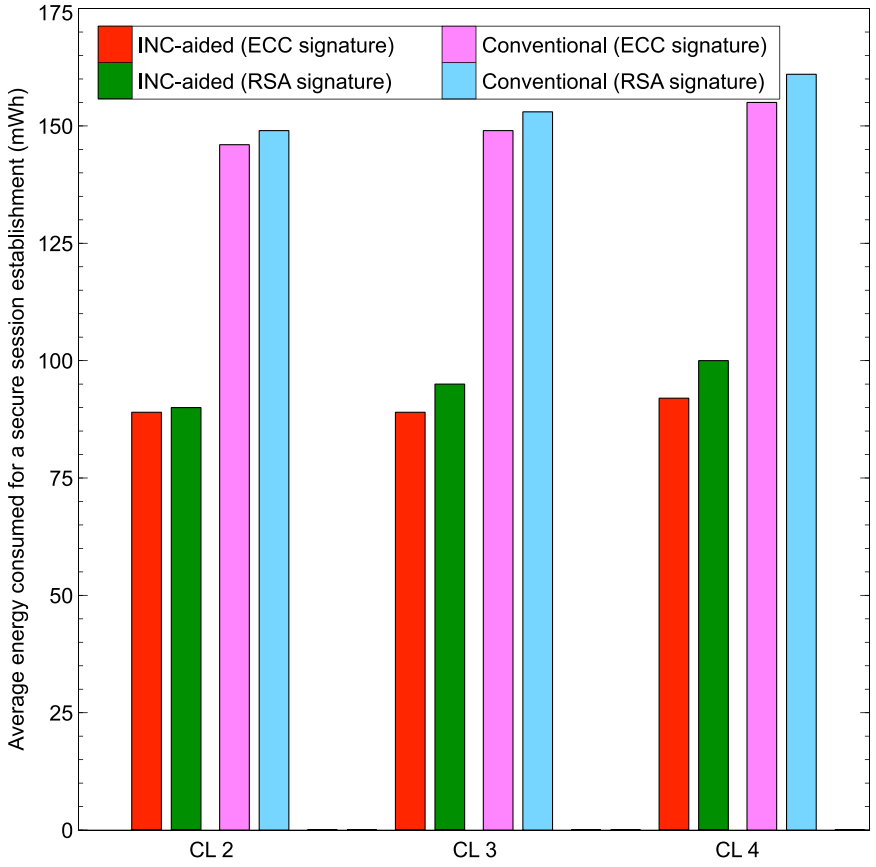


Fig. 8. Assessment of the impact of the Chain Length on energy consumption in the IIoT device (Server Public Key: RSA 2048).

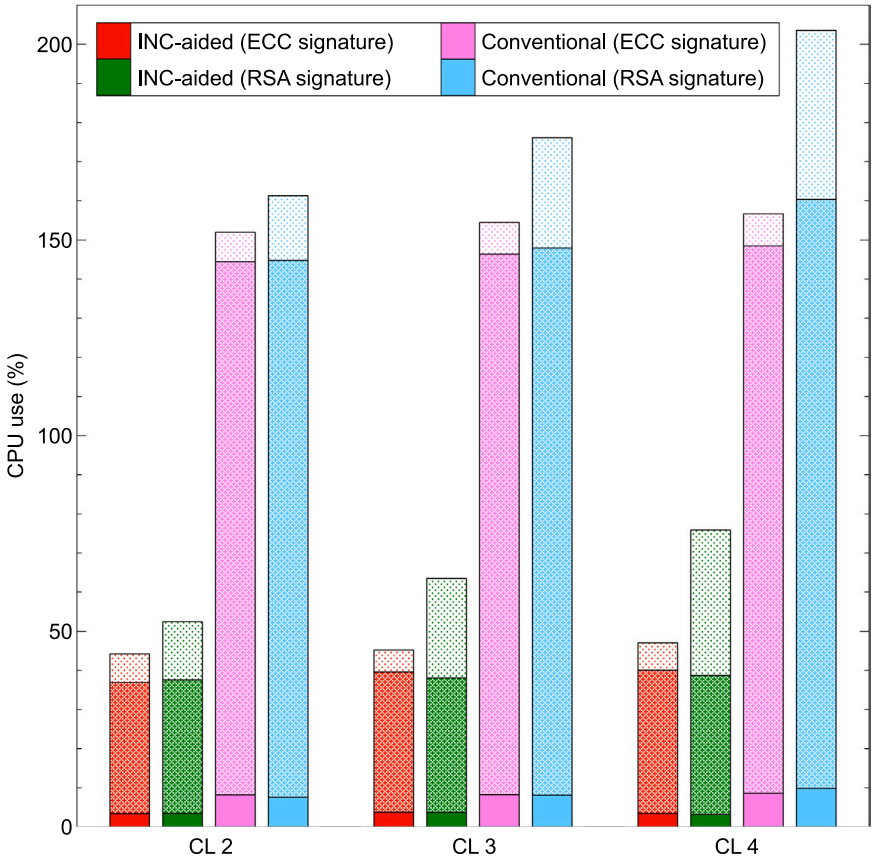


Fig. 9. Assessment of the impact of the Chain Length on CPU use in the IIoT device (Server Public Key: RSA 2048).

scalability when it comes to longer certificate chains. As a result, the performance of the tests that use RSA-based signatures is more affected by the length of the certificate chain.

The reduction of the different process times corresponds to less CPU use and message transmissions at the IIoT device, which in turn results in a reduction of the energy consumed by the IIoT device. Following the same color patterns in the bar graphs, the measured values for energy consumption are summarized in Fig. 8 for different lengths of the certificate chain using RSA or ECC keys for certificate signatures (server public keys are always RSA). The use of the proposed in-network certificate validation system allows reducing the energy consumption of the IIoT device by about 40 %, which is a noteworthy result, specially in the case of IIoT devices that operate on batteries. This reduction corresponds mainly to a less intensive use of the radio to transmit/receive messages, as for example, the IIoT device does not need to execute the OCSP message exchange. In fact, transmission and reception of bits over the air is one of the most consuming tasks for IIoT devices, being the energy used to transmit 1 bit similar to the energy used to execute about 1000 processor instructions Hill et al. (2000).

Fig. 9 shows the CPU use in the IIoT device following a similar bar color and chain length representation. In this case, different color shades indicate the CPU times for system (dark), user (intermediate), and total (light). The proposed INC-aided system clearly shows a reduction in CPU use time (red and green bars) compared to the conventional process times (magenta and blue bars), reducing from a 200 % to 70 % in the worst case scenario of a four certificate chain. It must be said that CPU use values above 100 % indicate the use of multiple cores, as the full use of a single core is represented as 100 %. This reduction is closely related to the time required to complete the DTLS handshake process and, as mentioned before, is one of the reasons for the reduction of energy consumption. Our solution requires less CPU use, mainly due to the delegation of the computation-heavy certificate validation process to a resource-richer device, so that the IIoT devices are freed from this task. Therefore, IIoT devices have fewer processes to run, which translates to lower CPU usage, which in turn means lower power consumption. Besides the specific contributions of our proposal, the mathematical method behind each encryption protocol is also relevant, as calculations of elliptic curves over finite fields are easier than prime factorization, even more when keys or certificate chains become larger and require more computing.

7. Conclusions

As the relevance of network security increases, DTLS is becoming one of the most used protocols in IIoT networks. However, performance and efficiency of this protocol is limited due to the constraints of IIoT devices. Therefore, we present a system to delegate server certificate validation within the DTLS handshake process to a resource-richer server leveraging data plane programming in an IIoT border router, reducing the total time of the handshake by 50–60 % and reducing CPU and battery consumption in the resource-deprived device.

It must be noted that these values have been obtained in a little testbed deployed in a laboratory environment. In a real industrial scenario, where interferences are present and distances are greater, communications through the wireless links will experience greater latencies. In this context, the improvement achieved with the proposed INC-aided mechanisms will be even more noticeable.

In conclusion, the obtained results demonstrate the suitability of the proposed solution for the defined scenario, and overall, prove the validity of the proposed system. Additionally, it is noteworthy that the results can be extrapolated to other environments, such as the Internet of Vehicles (IoV) or electronic health systems, which also make use of scarce resource devices and benefit from the increased security and performance enhancement. In fact, with the quick growth of TLS and DTLS, any resource-deprived scenario that implements these protocols can benefit from security and performance improvement provided by the proposed solution.

Author statement

The article was conceived and structured by all the authors. **A. Atutxa, J. Astorga** and **M. Barcelo** conceived the evaluation model and analysed the data. **A. Atutxa** wrote the paper and all the authors revised it and provided feedback.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

This work was financially supported by the Spanish Ministry of Science and Innovation through the TRUE-5G project PID2019-108713RB-C54/AEI/10.13039/501100011033. It was also partially supported by the Ayudas Cervera para Centros Tecnológicos grant of the Spanish Centre for the Development of Industrial Technology (CDTI) under the project EGIDA (CER-20191012), and by the Basque Country Government under the ELKARTEK Program, project REMEDY - Real tiME control and embeddeD securitY (KK-2021/00091).

References

Almaini, A., Al-Dubai, A., Romdhani, I., Schramm, M., Alsarhan, A., 2021. Lightweight edge authentication for software defined networks. Computing 103 (2), 291–311. https://doi.org/10.1007/s00607-020-00835-4.

AnonBehavioral Model (bmv2).⟨https://github.com/p4lang/behavioral-model⟩ 2022.

Atutxa, A., Franco, D., Sasiain, J., Astorga, J., Jacob, E., 2021. Achieving low latency communications in smart industrial networks with programmable data planes. Sensors 21 (15). https://doi.org/10.3390/s21155199.

Bao, Y., Qiu, W., Cheng, X., Sun, J., 2022. Fine-grained data sharing with enhanced privacy protection and dynamic users group service for the iov. IEEE Trans. Intell. Transp. Syst. 1–15. https://doi.org/10.1109/TITS.2022.3187980.

Bosshart, P., Daly, D., Gibb, G., Izzard, M., McKeown, N., Rexford, J., Schlesinger, C., Talayco, D., Vahdat, A., Varghese, G., Walker, D., 2014. P4: programming protocol-independent packet processors, SIGCOMM. Comput. Commun. Rev. 44 (3), 87–95. https://doi.org/10.1145/2656877.2656890.

F.E.R. Cesen, L. Csikor, C. Recalde, C.E. Rothenberg, G. Pongrácz, Towards low latency industrial robot control in programmable data planes. In: Proceedings of the Sixth IEEE Conference on Network Softwarization (NetSoft), 2020, 165–169.10.1109/NetSoft48620.2020.9165531.

Chang, C.-Y., Ruiz, T.G., Paolucci, F., Jiménez, M.A., Sacido, J., Papagianni, C., Ubaldi, F., Scano, D., Gharbaoui, M., Giorgetti, A., Valcarenghi, L., Tomakh, K., Boddi, A., Caparrós, A., Pergolesi, M., Martini, B., 2021. Performance isolation for network slices in industry 4.0: the 5growth approach. IEEE Access 9, 166990–167003. https://doi.org/10.1109/ACCESS.2021.3135827.

Chavan, A.A., Nighot, M.K., 2016. Secure andcost-effective application layer protocolwith authentication interoperability foriot. In: Proceedings of the First International Conference on Information Security & Privacy 2015, Procedia Computer Science 78, 646–651. ⟨http://www.sciencedirect.com/science/article/pii/S1877050916001149⟩.

E. Cho, M. Park, H. Lee, J. Choi, T.T. Kwon, D2TLS: delegation-based DTLS for cloud-based IoT services. In: Proceedings of the International Conference on Internet of Things Design and Implementation, IoTDI '19, Association for Computing Machinery, New York, NY, USA, 2019, 190–201.10.1145/3302505.3310081.

Corallo, A., Lazoi, M., Lezzi, M., Luperto, A., 2022. Cybersecurity awareness in the context of the industrial internet of things: a systematic literature review. Comput. Ind. 137, 103614 https://doi.org/10.1016/j.compind.2022.103614.

R. Datta, S. Choi, A. Chowdhary, Y. Park, P4guard: Designing p4 based firewall. In: Proceedings of the MILCOM 2018 - 2018 IEEE Military Communications Conference (MILCOM), 2018, 1–6.10.1109/MILCOM.2018.8599726.

Deebak, B.D., Memon, F.H., Cheng, X., Dev, K., Hu, J., Khowaja, S.A., Qureshi, N.M.F., Choi, K.H., 2022. Seamless privacy-preservation and authentication framework for iot-enabled smart ehealth systems. Sustain. Cities Soc. 80, 103661 https://doi.org/10.1016/j.scs.2021.103661. ⟨https://www.sciencedirect.com/science/article/pii/S2210670721009240⟩.

M. Dimolianis, A. Pavlidis, V. Maglaris, A multi-feature ddos detection schema on p4 network hardware. In: Proceedings of the Twenty Third Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)), 2020, 1–6.10.1109/ICIN48450.2020.9059327.
R. Falk, S. Fries, Managed certificate whitelisting - a basis for internet of things security in industrial automation applications. In: Proceedings of the Eighth International Conference on Emerging Security Information, Systems and Technologies - SECURWARE 2014, 2014.
Fouladgar, S., Mainaud, B., Masmoudi, K., Afifi, H., 2006. Tiny 3-tls: a trust delegation protocol for wireless sensor networks. In: Buttyán, L., Gligor, V.D., Westhoff, D. (Eds.), Security and Privacy in Ad-Hoc and Sensor Networks. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 32–42.
Granjal, J., Monteiro, E., Silva, J.S., 2013. Application-layer security for the wot: Extending coap to support end-to-end message security for internet-integrated sensing applications. In: Tsaoussidis, V., Kassler, A.J., Koucheryavy, Y., Mellouk, A. (Eds.), Wired/Wireless Internet Communication. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 140–153.
G. Grigoryan, Y. Liu, Lamp: Prompt layer 7 attack mitigation with programmable data planes. In: Proceedings of the Seventeenth International Symposium on Network Computing and Applications (NCA), IEEE 2018, 1–4.10.1109/NCA.2018.8548136.
Hauser, F., Häberle, M., Schmidt, M., Menth, M., 2020. P4-ipsec: site-to-site and host-to-site vpn with ipsec in p4-based sdn. IEEE Access 8, 139567–139586. https://doi.org/10.1109/ACCESS.2020.3012738.
Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D., Pister, K., 2000. System architecture directions for networked sensors. SIGARCH Comput. Archit. News 28 (5), 93–104. https://doi.org/10.1145/378995.379006.
Höglund, J., Lindemer, S., Furuhed, M., Raza, S., 2020. Pki4iot: Towards public key infrastructure for the internet of things. Comput. Secur. 89, 101658 https://doi.org/10.1016/j.cose.2019.101658. ⟨http://www.sciencedirect.com/science/article/pii/S0167404819302019⟩.
R. Hummen, H. Shafagh, S. Raza, T. Voig, K. Wehrle, Delegation-based authentication and authorization for the IP-based internet of things. In: Proceedings of the Eleventh Annual IEEE International Conference on Sensing, Communication, and Networking (SECON), 2014, 284–292.
Kang, N., Park, J., Kwon, H., Jung, S., 2015. ESSE: efficient secure session establishment for internet-integrated wireless sensor networks. Int. J. Distrib. Sen. Netw. 2015 https://doi.org/10.1155/2015/393754 (Jan.).
P.G. Kannan, R. Joshi, M.C. Chan, Precise time-synchronization in the data-plane using programmable switching asics. In: Proceedings of the ACM Symposium on SDN Research, SOSR '19, Association for Computing Machinery, New York, NY, USA, 2019, 8–20.10.1145/3314148.3314353.
M. Kuka, K. Vojanec, J. Kuera, P. Benáek, Accelerated ddos attacks mitigation using programmable data plane. In: Proceedings of the ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), 2019, 1–3.10.1109/ANCS.2019.8901882.
I. Kunze, R. Glebke, J. Scheiper, M. Bodenbenner, R.H. Schmitt, K. Wehrle, Investigating the applicability of in-network computing to industrial scenarios, In: Proceedings of the Fourth IEEE International Conference on Industrial Cyber-Physical Systems (ICPS), 2021, 334–340.10.1109/ICPS49255.2021.9468247.
Kwon, H., Ahn, J., 2019. On designing a lighter certificate for resource-limited Internet-of-Things devices. Trans. Emerg. Telecommun. Technol. 30 (10) https://doi.org/10.1002/ett.3740.
H. Kwon, S. Raza, On compressing pki certificates for resource limited internet of things devices. In: Proceedings of the Asia Conference on Computer and Communications Security, ASIACCS '18, Association for Computing Machinery, New York, NY, USA, 2018, 837–839.10.1145/3196494.3201591.
A.C. Lapolli, J. AdilsonMarques, L.P. Gaspary, Offloading real-time ddos attack detection to programmable data planes. In: Proceedings of the IFIP/IEEE Symposium on Integrated Network and Service Management (IM), 2019, 19–27.
Liu, Z., Sampaio, P., Pishchulov, G., Mehandjiev, N., Cisneros-Cabrera, S., Schirrmann, A., Jiru, F., Bnouhanna, N., 2022. The architectural design and implementation of a digital platform for industry 4.0 sme collaboration. Comput. Ind. 138, 103623 https://doi.org/10.1016/j.compind.2022.103623.
Ma, Y., Yan, L., Huang, X., Ma, M., Li, D., 2020. Dtlshps: Sdn-based dtls handshake protocol simplification for iot. IEEE Internet Things J. 7 (4), 3349–3362. https://doi.org/10.1109/JIOT.2020.2967464.
Madureira, A.L.R., Araújo, F.R.C., Sampaio, L.N., 2020. On supporting iot data aggregation through programmable data planes. Comput. Netw. 177, 107330 https://doi.org/10.1016/j.comnet.2020.107330.
Mahrach, S., Mjihil, O., Haqiq, A., 2018. Scalable and dynamic network intrusion detection and prevention system. In: Abraham, A., Haqiq, A., Muda, A.K., Gandhi, N. (Eds.), Innovations in Bio-Inspired Computing and Applications. Springer International Publishing, Cham, pp. 318–328 (pp).
Marino, F., Moiso, C., Petracca, M., 2019. PKIoT: A public key infrastructure for the Internet of Things. Trans. Emerg. Telecommun. Technol. 30 (10), 3681. https://doi.org/10.1002/ett.3681.
R. Millman, Iot Devices are More Vulnerable Than Ever. 2021. ⟨https://www.itpro.co.uk/network-internet/internet-of-things-iot/360850/iot-devices-are-more-vulnerable-than-ever⟩.
S.R. Moosavi, T.N. Gia, E. Nigussie, A. Rahmani, S. Virtanen, H. Tenhunen, J. Isoaho, Session resumption-based end-to-end security for healthcare internet-of-things. In: Proceedings of the IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing, 2015, 581–588.
P. Muncaster, Attacks on Iot Devices Double Over Past Year. 2021. ⟨https://www.infosecurity-magazine.com/news/attacks-iot-devices-double-past/⟩.
Ojo, M.O., Giordano, S., Procissi, G., Seitanidis, I.N., 2018. A review of low-end, middle-end, and high-end iot devices. IEEE Access 6, 70528–70554. https://doi.org/10.1109/ACCESS.2018.2879615.
J. Park, N. Kang, Lightweight secure communication for CoAP-enabled internet of things using delegated DTLS handshake, In: Proceedings of the International Conference on Information and Communication Technology Convergence (ICTC), 2014, 28–33.
Park, J., Kwon, H., Kang, N., 2016. Iot-cloud collaboration to establish a secure connection for lightweight devices. Wirel. Netw. 23 https://doi.org/10.1007/s11276-015-1182-y, 01.
Raza, S., Helgason, T., Papadimitratos, P., Voigt, T., 2017. Securesense: end-to-end secure communication architecture for the cloud-connected internet of things. Future Gener. Comput. Syst. 77, 40–51. https://doi.org/10.1016/j.future.2017.06.008. ⟨http://www.sciencedirect.com/science/article/pii/S0167739×17312360⟩.
Raza, S., Shafagh, H., Hewage, K., Hummen, R., Voigt, T., 2013. Lithe: lightweight secure coap for the internet of things. IEEE Sens. J. 13 (10), 3711–3720.
E. Rescorla, The transport layer security (tls) protocol version 1.3. Tech. Rep., IETF, RFC 8446 (2018).
E. Rescorla, N. Modadugu, Datagram transport layer security version 1.2. Tech. Rep., IETF, RFC 6347 (2012).
R. Ricart-Sanchez, P. Malagon, J.M. Alcaraz-Calero, Q. Wang, Netfpga-based firewall solution for 5g multi-tenant architectures. In: Proceedings of the IEEE International Conference on Edge Computing (EDGE), 2019,132–136.10.1109/EDGE.2019.00037.
F. Rodriguez, C.E. Rothenberg, G. Pongrácz, In-network p4-based low latency robot arm control. In: Proceedings of the Fifteenth International Conference on Emerging Networking EXperiments and Technologies, CoNEXT '19 Companion, Association for Computing Machinery, New York, NY, USA, 2019, 59–61.10.1145/3360468.3368178.
M. Schukat, P. Cortijo, Public key infrastructures and digital certificates for the internet of things. In: Proceedings of the Twenty Sixth Irish Signals and Systems Conference (ISSC), 2015, 1–5.
Tchoffa, D., Figay, N., Ghodous, P., Panetto, H., ElMhamedi, A., 2021. Alignment of the product lifecycle management federated interoperability framework with internet of things and virtual manufacturing. Comput. Ind. 130, 103466 https://doi.org/10.1016/j.compind.2021.103466.
Wang, S.-Y., Li, J.-Y., Lin, Y.-B., 2020. Aggregating and disaggregating packets with various sizes of payload in p4 switches at 100 gbps line rate. J. Netw. Comput. Appl. 165, 102676 https://doi.org/10.1016/j.jnca.2020.102676.
Xu, G., Bai, H., Xing, J., Luo, T., Xiong, N.N., Cheng, X., Liu, S., Zheng, X., 2022. Sg-pbft: a secure and highly efficient distributed blockchain pbft consensus algorithm for intelligent internet of vehicles. J. Parallel Distrib. Comput. 164, 1–11. https://doi.org/10.1016/j.jpdc.2022.01.029. ⟨https://www.sciencedirect.com/science/article/pii/S0743731522000363⟩.
F. Yousefi, A. Abhashkumar, K. Subramanian, K. Hans, S. Ghorbani, A. Akella, Liveness verification of stateful network functions, In: Proceedings of the Seventeenth USENIX Symposium on Networked Systems Design and Implementation (NSDI 20), USENIX Association, Santa Clara, CA, 2020, 257–272.⟨https://www.usenix.org/conference/nsdi20/presentation/yousefi⟩.
E.O. Zaballa, D. Franco, Z. Zhou, M.S. Berger, P4knocking: offloading host-based firewall functionalities to the network. In: Proceedings of the Twenty Third Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN), 2020, 7–12.10.1109/ICIN48450.2020.9059298.
Zuo, Z., Chang, C., Zhang, Y., He, R., Qin, X., Yung, K.L., 2020. P4label: packet forwarding control mechanism based on p4 for software-defined networking. J. Ambient Intell. Humaniz. Comput. https://doi.org/10.1007/s12652-020-01719-3.
Zuo, Z., Chang, C., Zhu, X., 2020. A software-defined networking packet forwarding verification mechanism based on programmable data plane. J. Electron. Inf. Technol. 42 (190381), 1110. https://doi.org/10.11999/JEIT190381. ⟨http://article/id/d413fb02-35a6-4421-a135-9f60306d781d⟩.
Zuo, Z., He, R., Zhu, X., Chang, C., 2019. A novel software-defined network packet security tunnel forwarding mechanism. Math. Biosci. Eng. 16 (5), 4359–4381. https://doi.org/10.3934/mbe.2019217.

Asier Atutxa received his BSc degree in Telecommunication Engineering in 2018 and his MSc degree in Telecommunication Engineering in 2020 from the University of the Basque Country (UPV/EHU). He joined the Communications Engineering Department of the UPV/EHU as a researcher in the I2T Research Lab (Engineering and Research on Telematics) in 2018. His research interests include software -defined networking, data plane programming and virtualization. Currently, he is a PhD student in Mobile Network Information and Communication Technologies at the UPV/EHU.

Jasone Astorga received her BSc and MSc degrees in Telecommunication Engineering in 2004 and her PhD (cum laude) in 2013 from the University of the Basque Country (UPV/EHU). From 2004–2007 she worked at Nextel S. A., a Telecommunications enterprise. In 2007 she joined the UPV/EHU as a lecturer and as a researcher in the I2T Research Lab (https://i2t.ehu.eus/es). Currently she is an Assistant Professor on the same Department. She has participated in several research projects at the local, national and European levels, being the main researcher in several of them, and she has supervised two Ph.D. theses. She is co-author of more than 20 papers published in relevant scientific journals and more than 30 contributions to recognized national and international conferences in her research field. Currently, her research interests include cybersecurity in industrial environments, Software Defined Networking and Network Function Virtualization for industrial communications.

Marc Barcelo is with IKERLAN since 2016, where he is currently a researcher in the Cybersecurity for Digital Platforms working group. He received his M.Sc. degree in electrical engineering (Hons.) and Ph.D. degree (cum laude) from Universitat Autonoma de Barcelona (UAB) in 2010 and 2015, respectively. From 2009–2015 he was with the UAB's Telecommunications and Systems Engineering Department participating in several international R&D projects. During 2014, he was an International Research Intern at Nokia Bell Labs, New Jersey, USA. He has published over 20 papers in recognized international journals and conferences. Currently, his research efforts are focused on the design of secure communication architectures for the Internet of Things and Industry 4.0.

Aitor Urbieta is with IKERLAN since 2007. He studied Informatics Engineering in the University of Mondragon (Spain), where he obtained his Ph.D. degree in Computer Science in 2010. He is currently a research fellow and belongs to the Digital Platform Cybersecurity team as the team leader. His current research interests include Cybersecurity, Internet of Things (IoT), Machine to Machine (M2M), Industrial Control Systems, Digital Platforms, Fog Computing, Edge Computing, IoT environments validation, IoT environments testing and Middleware. He is author or co-author of more than 20 peer-reviewed scientific publications in the field of Internet of Things, Service Oriented Architectures and Semantic Web. During the last ten years he has been working on the development and validation of several IoT platforms for various domains (transport, energy and smart manufacturing), with emphasis on Cybersecurity and Safety issues.

Eduardo Jacob (Senior Member, IEEE) received the B.Sc. degree in industrial engineering and the M.Sc. degree in industrial engineering, and industrial communications and electronics respectively in 1987 and 1991, and the Ph.D. degree in communications engineering in 2001 from the University of the Basque Country (UPV/EHU). During two years he worked in a public telecommunications research and development enterprise (currently Tecnalia). He spent several years as the IT Director in the private sector. Since 1994 then he has been at full time with the Faculty of Engineering in Bilbao, UPV/EHU, where he was elected as Head of the Department of Communications Engineering from 2012 to 2016. He is currently Full Professor and leads the I2T (Engineering and Research on Telematics) Research Laboratory. He also has directed several Ph.D. theses and managed several research projects at the local, national, and European levels. His research interests include applying software-defined networks to industrial communications, cybersecurity in distributed systems, software-defined wireless sensor networks, and in-network processing.