Empowering Resource-Constrained WoT Devices With Lightweight Self-Sovereign Identity (SSI) Using Delegation

Biagio Boi
dept. of Computer Science
University of Salerno
Fisciano, Salerno, Italy
bboi@unisa.it

Marco De Santis
dept. of Computer Science
University of Salerno
Fisciano, Salerno, Italy
mdesantis@unisa.it

Christian Esposito
dept. of Computer Science
University of Salerno
Fisciano, Salerno, Italy
esposito@unisa.it

***Abstract*—The Web of Things (WoT) represents a complex ecosystem of interconnected devices that exchange vast amounts of data, enabling advanced applications essential for various industrial and social processes. These applications face stringent security requirements due to the open nature of the Internet of Things (IoT) ecosystem and the widespread deployment of devices in public environments, posing significant challenges in safeguarding against malicious activities. FIWARE has established itself as a leading IoT infrastructure standard, offering robust security and access control through key components that facilitate authentication, access management, and secure data transmission. However, traditional authentication methods cannot be implemented at end-device level, posing significant risks. This study bridges critical gaps by seamlessly integrating Self-Sovereign Identity (SSI) into FIWARE by incorporating an innovative delegate node designed to enhance the computational capabilities of resource-constrained IoT devices while adhering to SSI design principles. By deploying an SSI-compliant agent on a gateway node and utilizing keys stored on devices with minimal memory—requiring only 520 KB of SRAM—via MQTT, this approach demonstrates its feasibility both in terms of performance and security. The results indicate an average session key generation and authentication time of 2.573 seconds, enabling mutual authentication between the application and the end device, making it suitable for real-world scenarios. Also, we provide a formal verification for the proposed protocol using the ProVerif model checking tool to check and validate our approach.**

***Index Terms*—IoT, Self-Sovereign Identity, Authentication, FIWARE, Decentralized Systems, MQTT**

I. Introduction

Networks including Internet of Things (IoT) devices are increasing the complexity by including interconnected devices that exchange large volumes of data, enabling advanced applications based on information and communication technologies (ICT). These applications, which are essential for various industrial and social processes, have particularly stringent security requirements due to their critical role in mission-critical contexts. The open nature of the IoT ecosystem and the pervasive distribution of devices in public spaces present significant challenges in terms of protection against malicious activities. Physical layer attacks [1], as well as, attacks to low-energy communication protocols [2], can potentially disrupt the entire architecture. Ensuring the security and privacy of the entire sensing loop at edge and end devices is crucial to prevent potential attacks from compromising local applications and propagating them to cloud services or external systems [3].

In the context of secure device and data management, FIWARE [4] has emerged as a standard reference for IoT infrastructure, providing an open-source platform designed to ensure security and access control. FIWARE integrates various key components that work in synergy to enable authentication, access control, and secure information transmission. A typical scenario involves IoT device enrollment by application and then the device sending the collected data to FIWARE. This process ensures that only authorized devices can interact with the system, delegating trust over-enrolled devices to the application, thereby providing effective protection against unauthorized access and enabling secure information management.

Open and Agile Smart Cities (OASC) demonstrates the expansion of FIWARE by developing one of the most famous global communities that is able to connect cities worldwide to enhance standardization and the usage of IoT-based technologies, by leveraging FIWARE as a central framework [5]. FIWARE authentication is splitted between user and device authentication. If by user side the credentials such as username and password are exchanged for a token according to OAuth2 protocol, creating an almost secure authentication; the approach is slightly different for the IoT devices. FIWARE provides certain security mechanisms, such as the Wilma PEP-Proxy, to enhance application security and protect against malicious users. Typically, the PEP-Proxy filters incoming requests; if the token in the request is valid, it forwards the request to the backend. Otherwise, it responds with an unauthorized status code. Anyway, applications are required to register devices, handle their authentication, and ensure secure data collection. This imposes excessive responsibilities on the application, creating a potential single point of failure. Furthermore, there are no assurances regarding the reliability of authentication performed by the application.

Due to the lack of standardization in how applications implement security in their interactions with IoT devices, this manuscript proposes a novel approach based on decentralized authentication. Self-Sovereign Identity (SSI)

is gaining interest in user-centric and device-centric authentication by empowering entities with a Decentralized Identifier (DID) used for public key-based authentication. The DID is typically associated with a set of attributes, creating the so-called Verifiable Credentials (VC) used to assess authorization and authentication in a decentralized context. SSI significantly enhances system security by enabling entities to directly own and manage their credentials, thus reducing the risks associated with password-based attacks. DIDComm is the protocol used for the exchange of DID-related information in a decentralized environment. Anyway, low-power devices struggle to keep up with the overhead introduced by DIDComm.

In this work, we propose the integration of SSI with memory-constrained devices, leveraging the concepts of DIDComm within an MQTT-aided communication framework. This approach ensures accessibility for devices that currently struggle to support asymmetric encryption due to limited energy and memory resources. The main contributions of this paper are summarized as follows:

- Propose a novel lightweight decentralized FIWARE-compliant authentication and data encryption mechanism that enhances security at both the application and, consequently, the IoT device levels.
- Formally evaluate the protocol for exchanging credentials between the application and IoT devices against authentication, secrecy, and forward secrecy attacks.
- Analyze the performance on constrained IoT devices, demonstrating the protocol's capabilities and effectiveness in a typical data collection process.

The manuscript is organized into five sections. Section 2 reviews the state-of-the-art in decentralized authentication within the Internet of Things (IoT) domain. Section 3 presents the proposed architecture, detailing its design principles, components, and communication protocol. In Section 4, we conduct a comprehensive analysis of the proposed architecture, evaluating its performance and security. Finally, Section 5 concludes the manuscript by summarizing the key findings and outlining potential directions for future research in decentralized authentication for IoT devices.

II. Related Work

Decentralized authentication in the IoT domain is a relevant topic, and different research directions have opened up for solving the problem of energy- and memory-constrained devices. Fan et al. describe DIAM-IoT [6], a decentralized Identity and Access Management (IAM) framework for IoT, designed to address interoperability and data-sharing challenges in a global ecosystem characterized by a large number of devices and users. While DIAM-IoT promotes seamless interaction among heterogeneous IoT platforms, it does not define a clear mechanism for the implementation of the protocol in constrained IoT devices, which cannot directly access the blockchain. Moreover, DIDs and their keys are directly saved on the device, potentially leading to misuse in case of physical attacks. On the same line, Gebresilassie et al. [7] focus on developing a decentralized identity management system for IoT devices. Their approach enhances resilience by distributing trust and mitigating single points of failure. However, the framework primarily focuses on user identity management and does not sufficiently consider the unique requirements of IoT devices operating in untrusted environments, leaving questions about its suitability for highly constrained devices.

Abubakar et al. [8] propose a blockchain-based approach for identity and authentication schemes with MQTT protocol integration, utilizing Ethereum smart contracts to automate authentication and authorization. Their evaluation demonstrates reduced resource usage, with memory utilization approximately 200MB less than TLS and CPU usage during authentication reduced to 24% compared to 81% for TLS. However, the reliance on Ethereum's block mining process introduces significant delays, requiring two blockchain transactions for authentication (challenge generation and verification), resulting in 26–30 seconds total delay. While effective for enhancing trust and accountability, this latency makes the scheme impractical for real-time IoT applications. Khalid et al. [9] also leverage blockchain but introduce fog computing to distribute the computational load, achieving improved efficiency for IoT devices. Their experimental setup involved a combination of laptops and Raspberry Pi systems acting as fog and IoT nodes. The framework demonstrated superior performance on resource-constrained devices, with Raspberry Pi nodes requiring only 24.77ms for registration requests and 0.09ms for data message transmission. However, like in [8], the reliance on Ethereum introduces approximately 14 seconds latency per transaction due to the Proof of Work (PoW) consensus mechanism, highlighting scalability concerns for real-time or large-scale deployments.

Firstly formulation of decentralised authentication using SSI has been proposed by Dixit et al. [10] in the context of Industrial IoT (IIoT), integrating DIDs and VCs using Ethereum and Hyperledger Indy. Their evaluation revealed efficient credential verification times of 0.2–0.3s on a Hyperledger Indy setup with four nodes. However, the hybrid approach introduces significant complexity, with Ethereum integration requiring substantial storage and resource demands, including 950KB for the DID registry smart contract and 350KB for the Verifiable Claims Registry.

Fathalla et al. [11] also propose a lightweight SSI framework, combining blockchain and secret sharing techniques to improve security while reducing computational overhead. Their evaluation on an emulated IoT network with Raspberry Pi 4 devices achieved a latency in identity issuance of between 0.4 and 0.6 seconds for 2 to 9 shares of credentials. However, identity verification exhibited linear increases in latency with more edge nodes, raising concerns about scalability. Furthermore, encryption overhead was excluded from the evaluation, leaving questions about the performance of the framework under real-world conditions.

A more recent contribution is DAXiot by Philipp et al. [12], which implements a decentralized authentication scheme using DIDs and Selective Disclosure JSON Web Tokens (SD-JWTs) within the MQTT 5.0 protocol. The evaluation, conducted with a broker and subscriber on an Intel Core i5 notebook and a publisher on a Raspberry Pi 3 Model B, demonstrated connection establishment times

of 115.8ms (21% slower than mutual TLS) and message publishing times of 6.3ms (10% slower than mutual TLS). Despite its strong focus on privacy and flexibility, DAXiot relies on centralized components, such as a revocation registry, and its suitability for highly resource-constrained devices remains unexplored.

Despite the technological benefits of these solutions, several limitations persist. Blockchain-based approaches, while decentralized, often introduce high computational and storage overhead, making them unsuitable for lightweight IoT devices [13]. SSI frameworks frequently rely on blockchain, which increases deployment costs and complicates their adoption in resource-constrained environments. Furthermore, many existing solutions lack formal security verification to ensure the robustness of their protocols against confidentiality, integrity, and authenticity threats. Limited research has focused on integrating these advanced authentication mechanisms with widely adopted IoT frameworks such as FIWARE, as highlighted in [14] and [15].

These challenges underline the need for decentralized authentication mechanisms that address the performance and security issues of existing solutions while ensuring compatibility with real-world IoT constraints.

III. System Model

This section outlines the architecture of the proposed SSI-enabled authentication framework for constrained IoT environments integrated with FIWARE. The system introduces a delegated identity agent that decouples authentication responsibilities from applications, reducing their complexity and enhancing overall security. As depicted in Fig. 1, the architecture consists of three primary layers: (1) the FIWARE core and application layer, which in this work is considered as a whole and only represented by the PEP Proxy as an entry point, but it can be further exploited; (2) the SSI gateway layer, identified as Aries Multi-Tenant Agent (AMA); and (3) the constrained IoT Device layer.

Currently, FIWARE's architecture places full responsibility for IoT device authentication and identity management upon the applications, lacking standardized protocols. The integration of SSI principles into this architecture addresses this gap by enabling decentralized authentication processes. In this enhanced workflow, the external service, through the PEP Proxy initiates an authentication request forwarded to the AMA. The AMA then performs the SSI-based authentication steps necessary for securely validating IoT device identities. The proposed architecture employs MQTT communication with IoT end-devices and utilizes DIDComm to secure applications authentication. The classical FIWARE framework remains the core system used to manage data and device interactions aligned with the WoT standards.

In the following sections, we provide detailed insights into each component's roles, interactions, and responsibilities, clarifying how their coordinated functioning ensures secure, efficient, and SSI-compliant authentication within the extended FIWARE ecosystem.

A. Architecture

1) PEP Proxy: In the proposed architecture, FIWARE interacts with the AMA through the Wilma PEP Proxy,

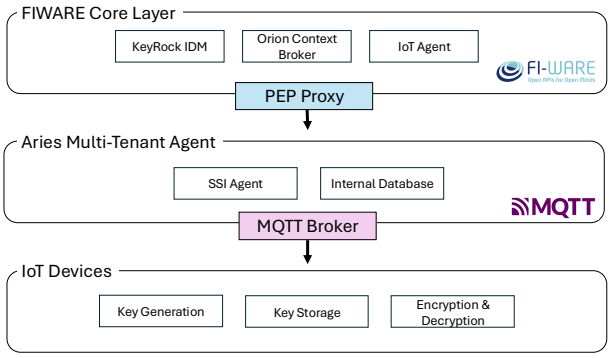

Fig. 1. Overview of the three-layer SSI-enhanced architecture for resource-constrained IoT devices within a FIWARE environment.

in line with the conventional FIWARE framework. The primary enhancement over the traditional model is the introduction of new REST endpoints to facilitate secure communication between the PEP Proxy and AMA.

Assuming that the IoT device has already been enrolled and can be authenticated via its interaction with the AMA, the PEP Proxy operates by forwarding authentication requests originating from the FIWARE core to the AMA. To support this functionality, the proxy is extended with additional HTTPS-secured endpoints that manage both the initial authentication phase and the responses issued by the IoT device. In particular, the following endpoints are exposed:

- `initAuthentication`: initializes the authentication procedure initiated by the FIWARE core (e.g., KeyRock).
- `deviceKeyReceive`: receives the session encryption key generated by the IoT device.
- `deviceVPReceive`: receives the Verifiable Presentation (VP) issued by the IoT device.

Authentication is completed only after the Verifiable Presentation is received and, ideally, validated by the FIWARE Identity Manager (KeyRock), which is responsible for assessing the presented attributes. Once the device is successfully authenticated, it is authorized to participate in secure data sensing and transmission, ensuring that only verified devices contribute to the system, thus improving its overall integrity and security.

The interaction between the AMA and the PEP Proxy is secured via a TLS session, which guarantees the confidentiality and integrity of the exchanged data, and prevents unauthorized access from untrusted domains.

As part of its standard functionality, the Wilma PEP Proxy filters incoming requests targeting IoT devices. It verifies the Proof of Possession (PoP) token, issued by KeyRock, to ensure that each request is authorized. If validation succeeds, the request is forwarded to the respective IoT device, encrypting the request using the pre-shared session key established during the authentication. Once received the request, the device processes the message and returns a response. This response propagates through the Context Broker and is ultimately delivered to the external application.

While KeyRock may represent a potential single point of failure, this risk can be mitigated by configuring short-lived PoP tokens, thereby reducing the attack surface and aligning with application-specific security requirements. The establishment of session key between PEP and IoT device prevent the AMA to know the message exchanged while being able to authenticate the device succesfully.

2) Aries Multi-Tenant Agent: Typical applications interact with the Wilma PEP Proxy through a backend managed within the FIWARE core. While the PEP is responsible for receiving authentication-related data from the IoT device, the actual authentication procedure is delegated to the AMA. In this revised design, the PEP sends a request to the AMA specifying the resource the user intends to access, identified by a URI. The AMA processes this request as a Verifiable Presentation Request (VPR), following the SSI paradigm. Unlike conventional SSI Agent implementations, the proposed architecture introduces an extended version tailored to the specific requirements of resource-constrained environments. A transport plugin based on the MQTT protocol has been developed to enable communication with IoT devices, emphasizing both efficiency and scalability. The AMA dynamically establishes MQTT topics based on the requested URI. Each endpoint is internally mapped to a corresponding Device ID and topic using the AMA database. These values are passed as parameters during MQTT client initialization and are used to define device-specific communication channels. This functionality is implemented by integrating a plugin into the SSI agent framework, which registers a callback to manage MQTT transport. The agent publishes authentication requests on the designated topic and listens for responses on the same channel. Once data is received, it is processed as part of the SSI authentication flow. An advanced configuration enables setting the Quality of Service (QoS) level to 2, ensuring that each message is delivered exactly once and acknowledged. This level of reliability is especially suited for critical IoT applications, such as healthcare systems, where message loss could compromise functionality or safety.

To emulate the DIDComm handshake over MQTT, the standard protocol roles of *requester* and *responder* are maintained. Typically, the requester initiates the session, and the responder acknowledges and completes it. However, in this implementation, the final *complete* (ACK) message is omitted, as it carries no additional information and serves only as a confirmation. In our architecture, the gateway (AMA) acts as the requester and MQTT broker, while the IoT device functions as the responder. Communication is considered complete upon the creation of the MQTT topic and the reception of a valid message from the device.

This dual-role architecture allows the AMA to function simultaneously as an SSI agent and as an MQTT broker, managing multiple IoT device sessions independently. For each device, a dedicated MQTT client is instantiated, effectively bridging the constrained device with the FIWARE infrastructure. Given the limited memory available on typical IoT devices (e.g., ESP32 with 520 KB of SRAM), this model enables the device to securely use its locally stored private key to encrypt credentials and generate proofs without exceeding its resource limitations.

3) IoT Devices: Communication between IoT devices and the Agent is established using the MQTT protocol in its unsecured variant, which does not rely on TLS. However, as outlined in the following section, the proposed authentication protocol ensures confidentiality and integrity through a combination of asymmetric and symmetric encryption mechanisms, alongside the DIDComm messaging protocol.

The generation of a VP on a resource-constrained IoT device requires a secure and lightweight cryptographic stack. To meet this requirement, the proposed solution employs the Edwards-curve Digital Signature Algorithm (EdDSA) using `ed25519` key pairs [16]. From the same key material, `x25519` keys are derived to support Elliptic Curve Diffie-Hellman (ECDH) key exchange, allowing the device and the agent to establish secure session keys [17]. This unified key strategy significantly reduces the memory footprint, as it avoids the need to store and manage multiple key types.

Once the session keys are negotiated through ECDH, according to DIDComm handshake, the IoT device proceeds to encrypt the VC received by the AMA, and sign the presentation data. To perform authenticated encryption, the protocol uses the ChaCha20-Poly1305 algorithm [18]. ChaCha20 offers both energy efficiency and enhanced resistance to certain cache-timing attacks compared to traditional ciphers like AES, making it a robust choice for heterogeneous IoT environments.

The Poly1305 component complements ChaCha20 by providing message authentication. It generates a Message Authentication Code (MAC) that ensures the integrity of the encrypted payload. This guarantees that any modification of the ciphertext will be detected, a critical requirement for the trustworthy generation and transmission of VPs.

Altogether, this cryptographic pipeline enables the IoT device to securely construct, sign, and transmit a Verifiable Presentation over an insecure MQTT channel, fulfilling the requirements of SSI while remaining compatible with the device's resource constraints.

B. Authentication Protocol

The message flow required for delegate authentication and encryption is illustrated in Fig. 2. In this context, we adopt the reduction $DID_x \rightarrow pk_x = pk(sk_x)$, which is known to all participants, including the IoT Device. The Aries Multi-Tenant Agent (AMA), identified by its public DID_{AMA}, is equipped with an MQTT Broker and serves as both the Issuer and Delegator of devices identity.

The PEP, as defined in the FIWARE Architecture, is enhanced with an SSI Wallet and identified by DID_P. It acts as the Verifier of device identity. Each device is represented by the AMA and possesses a private key sk_D. Unlike the other actors, IoT Device DID_D is stored in the AMA's wallet, while its private key sk_D remains under the device control, by saving it in its firmware. This design ensures that sk_D is directly managed by the IoT device, addressing the constraints that limit wallet implementation.

Communication between IoT Device and the AMA relies on the MQTT protocol, but in its insecure version—without

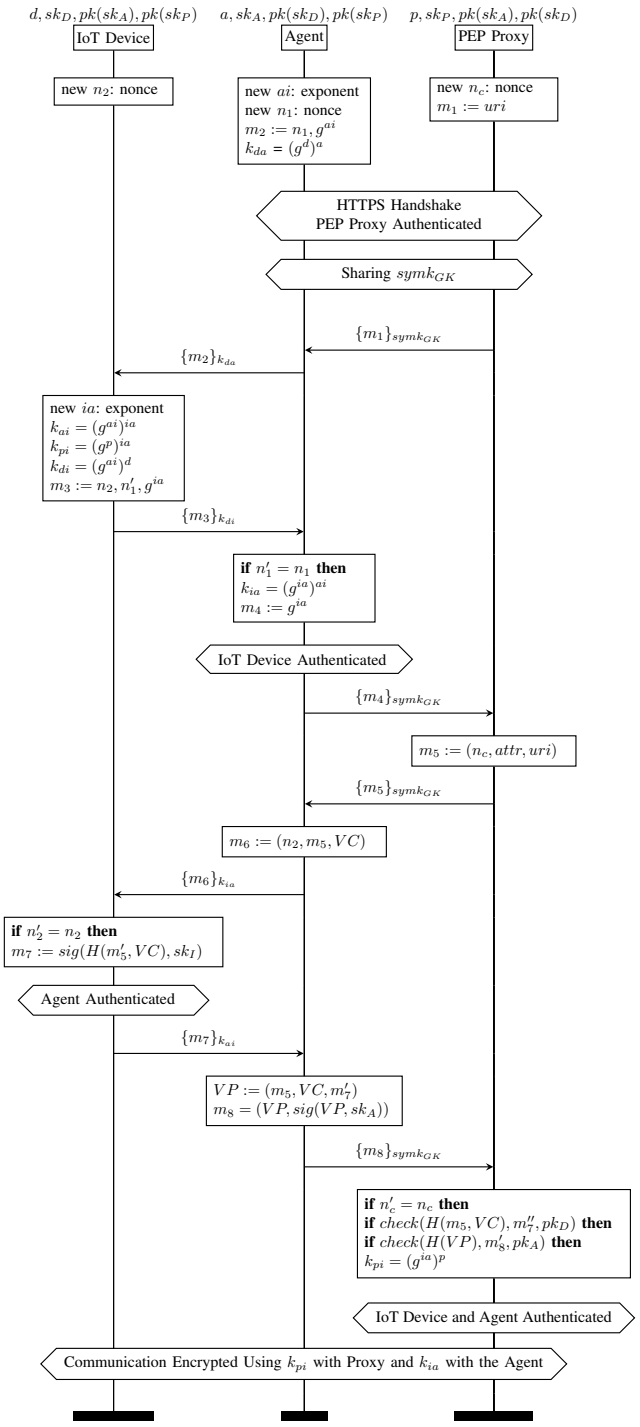

Fig. 2. Proposed authentication protocol, describing the interactions between the IoT device, Aries Multi-Tenant Agent (AMA), and PEP Proxy.

a TLS handshake. Consequently, this channel is vulnerable, allowing any party to potentially send messages. In contrast, the communication channel between the AMA and the PEP is secured through a TLS handshake using HTTPS, ensuring confidentiality and integrity for this segment of the message flow. We will demonstrate how our protocol achieves security by leveraging the session establishment at the very beginning of the protocol.

1) m_1: The PEP Proxy sends a request to access the data of the IoT Device identified by uri. Since the IoT Device cannont establish a direct connection, this request is handled by AMA.
2) m_2: AMA prepares to process the request by saving a record in its Secure Storage. This record contains the necessary transport information for communication with the IoT Device and with the PEP Proxy. AMA then sends a connection request, including a nonce n_1 and a session generator g^{ai}, encrypted using the shared secret derived from the IoT Device's public key pk_d and the AMA's private key sk_a.
3) m_3: The IoT Device generates a fresh exponent ia and derives two symmetric keys. The first key k_{ai} is generated by exponentiating the fresh exponent ia with g^{ai} for secure communication with AMA in m_6, while the second key is derived using g^p for communication with the PEP Proxy. A third key k_{di} is needed to encrypt the m_3 for the AMA. The device encrypts a message containing a new nonce n_2, the previously received nonce n_1, and the fresh generator g^{ia} with the novel shared secret k_{di}.
4) *IoT Device Authentication*: AMA verifies the received nonce n_1' against the previously generated value. If the nonces match, the IoT Device is authenticated. This is because if the IoT Device is able to access the nonce, it is assumed it is the owner of sk_D, considering the cyphering of m_2.
5) m_4: AMA generates the shared symmetric key k_{ia}. It then forwards the received generator g^{ai} to the PEP Proxy, encrypting the message with the shared secret $symk_{GK}$.
6) m_5: The PEP Proxy creates a Verifiable Presentation Request (VPR), which includes a nonce n_c, the required attribute $attr$, and the requested uri. This VPR is sent over the secure channel to AMA.
7) m_6: AMA prepares a signing request for the IoT Device. It encrypts a message with k_{ia} that includes the nonce n_2 (received earlier from the device), the VPR from m_5, and the VC retrieved from the wallet of DID_d.
8) *Agent Authentication*: The IoT Device authenticates AMA by verifying that the received nonce n_2' matches the previously sent n_2. This confirms that the IoT Device is communicating with the intended responder.
9) m_7: The IoT Device generates a signature over the VPR and VC. This message, considered a Verifiable Presentation (VP), is encrypted with the session shared secret k_{ai} and sent to AMA.
10) m_8: AMA wraps the VP by signing it with its private key sk_a and forwards the wrapped VP to the PEP

Proxy over the secure channel.

11) *IoT Device and AMA Authentication*: The PEP Proxy authenticates both AMA and the IoT Device. It verifies that the received nonce n_c' matches the previously sent n_c, and it confirms that the signatures on the VP and the wrapped VP match the public keys of their respective parties. After successful verification, the PEP Proxy generates a shared session secret k_{pi} with the IoT Device.
12) *Communication Encrypted Using k_{pi} with Proxy and k_{ia} with the Agent*: All subsequent communication between the PEP Proxy and the IoT Device occurs over the MQTT channel. However, the communication is encrypted with the session secret k_{pi}, ensuring security. Notably, this shared secret is not accessible to AMA, maintaining the separation of roles. Additionally, in the case AMA needs to exchange information with the IoT Device, it will leverage k_{ia} for the encryption and decryption, preserving the access from the PEP Proxy.

IV. Results

We evaluate the proposed architecture from both the security and performance perspectives. To this end, we first devise a formal security analysis, considering the most relevant properties: *secrecy*, *forward secrecy*, and *mutual authentication*. In the second part, we analyze the performance of the protocol, taking into account the time required for the generation of an asymmetric key pair, and the encryption time required for the protocol.

A. Experimental Setup

To evaluate the system, we used a Raspberry Pi 4 equipped with a Quad-core Cortex-A72 (ARM v8) 64-bit SoC running at 1.8GHz and 4GB of LPDDR4-3200 SDRAM, considered as the Aries Multi-Tenant Agent for the distributed protocol. The IoT device is an ESP32 microcontroller, featuring a dual-core 32-bit processor with a clock speed of 240 MHz and 520 KB of SRAM, specifically designed for low-power applications with Wi-Fi and Bluetooth connectivity. The role of the Proxy is an external server powered by an Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz - 1.80GHz with 8GB of RAM. Aries Multi-Tenant Agents were deployed on both ends leveraging the TypeScript version of Credo.ts: the Raspberry Pi integrates it in a server web which also publishes and subscribes over an MQTT Broker, managed by itself; and the other side, the PEP Proxy, integrates it in the existing library. The Raspberry Pi acted as the multi-tenant node responsible for managing decentralized protocols, while the ESP32 microcontroller served as the constrained device to demonstrate the feasibility of implementing secure communications even on low-resource platforms.

B. Security Analysis

Based on a model reflecting the trust assumption defined in Fig. 2, we now consider the security and privacy properties to verify that they hold. We leveraged ProVerif [19] for the automatic verification of the proposed protocol by dividing the analysis into (forward) secrecy and (mutual) authentication.

1) Secrecy: The secrecy guarantees that the message cannot be accessed from an attacker. In particular, the secrecy must be guaranteed on three messages of the considered protocol:

- Verifiable Credentials (m_6): This message is sent from the Agent to the IoT Device. This contains all the information of the identity of IoT device and must not be disclosed. The secrecy hold since this message is sent only by encrypting it using a shared secret k_{ia} only known to the IoT device, preventing the disclosure of the message.
- Verifiable Presentation (m_7): This message contains the signature applied over the VC from the IoT Device. The secrecy hold since the IoT Device send the message to the Agent by encrypting it using the shared secret k_{ai} known only to the IoT Device and Agent.
- Data Exchange (m_9, m_{10}): These messages contains the data exchanged between Proxy and IoT Device, and must be encrypted. The secrecy hold since the communication is encrypted using the k_{pi} known only by the Proxy and the IoT Device.

2) Forward Secrecy: Forward secrecy, also known as perfect forward secrecy, is a critical property of cryptographic protocols that ensures the compromise of long-term private keys does not affect the confidentiality of past communications. In our context, this means that even if an attacker later obtains the private key of an IoT Device, AMA, or PEP Proxy, they cannot use it to decrypt previously recorded messages. As previously described, the symmetric keys that guarantee secrecy are derived from asymmetric private keys. If these private keys are compromised, it is possible that some communications could be exposed. To mitigate this risk, forward secrecy employs session keys—ephemeral keys generated uniquely for each session—to encrypt data. These keys are independent of long-term credentials and are discarded after the session concludes, preventing retroactive decryption of past sessions even if long-term keys are compromised later.

In our protocol, during message m_3, the IoT Device selects a random ephemeral exponent, which is then used to establish shared secrets with both the Agent and the Proxy. These session-specific shared secrets are derived using the Ephemeral ECDH key exchange, ensuring that they are not dependent on any long-term keys.

However, while this property holds for the communication between the IoT Device and the Agent, it does not fully extend to the communication between the IoT Device and the PEP Proxy. Specifically, the analysis demonstrate possible attacks over the data message m_9. The motivation behind this attack come form the shared secret k_{pi}, which is generated from a combination of a fresh secret (ephemeral exponent) and the Proxy's long-term private key. Consequently, if an attacker compromises the Proxy's private key at any time in the future, they can decrypt past communications by accessing message m_4, which includes the exponent used in key derivation.

To enhance the protocol and ensure forward secrecy in the IoT Device–Proxy communication, it is necessary for the Proxy to also contribute a fresh ephemeral exponent. This exponent should be included in message m_5, along with a digital signature generated using the Proxy's long-term private key sk_P to ensure authenticity. The Agent can then forward this message to the IoT Device within message m_6. Upon receiving m_6, the IoT Device can compute a new session key based on the Proxy's ephemeral value g^{sp} and its own ephemeral exponent ia. This newly derived shared secret is then used exclusively for encryption and decryption, providing forward secrecy even in the event of a long-term key compromise.

TABLE I
SECRECY AND FORWARD SECRECY CHECKS OF THE PROTOCOL

Information	**Proposed**		**Improved**
	Secrecy	Fw. Secrecy	Fw. Secrecy
Ver. Credentials (m_6)	✓	✓	✓
Ver. Presentation (m_7, m_8)	✓	✓	✓
Data Exchange (m_9)	✓	✗	✓

3) Mutual Authentication: Authentication property consist in the ability of a party to assess the origin of a message, and so the authenticity of an action. In the case of decentralized authentication, this gain particular interest, since there is no party responsible for the authentication, which is processed at the communication level. We split our analysis by taking into account the individual session created between parties.

Proxy-Agent Authentication. The Agent authenticates the Proxy at the beginning of the protocol using the HTTPS Handshake.The Proxy is required to sign a nonce generated by the Agent, confirming its authenticity. On the other side, the Agent is authenticated only after the exchange of message m_8. m_8 contains a signature applied over the Verifiable Presentation (VP) sent from the IoT Device. This approach is also used for the authentication of the IoT Device by the Proxy.

Proxy-IoT Device Authentication. The Proxy authenticates the IoT Device through the signature applied over the VP. On the contrary, the IoT Device authenticates the Proxy only when a future request is made. The IoT Device encrypts the data exchange using the shared secret established in the previous interaction. No additional signature is required, as the asymmetric encryption applied over the fresh generator is sufficient for authentication.

Agent-IoT Device Authentication. The Agent authenticates the IoT Device at the beginning of the protocol through the exchange of messages m_1 and m_2. Authentication is achieved by encrypting the nonce n_1. The IoT Device authenticates the Agent through the encryption applied over the nonce n_2. The IoT Device generates messages only upon receiving the correct nonce, preventing attackers from falsifying authentication.

TABLE II
AUTHENTICATION VERIFICATION BETWEEN PARTIES

	Proxy	**Agent**	**IoT Device**
Proxy	–	✓(m_8)	✓(m_9)
Agent	✓(m_7)	–	✓(m_5)
IoT Device	✓(m_{10})	✓(m_6)	–

C. Performance Analysis

In this section, we aim to measure the overhead introduced by two distinct aspects. First, we will evaluate the impact of adding delegation to classical SSI Agents representing an innovation compared to traditional authentication and data exchange models. Second, the analysis seeks to quantify the additional overhead associated with integrating CredoTS into the FIWARE platform in terms of connection setup, verifiable presentation generation, and data sensing.

Table III presents the overhead introduced by the proposed solution, analyzing different cryptographic operations and their total execution time across an SSI agent and an IoT device.

Similarly to traditional approach, SSI agent performs most of the computationally intensive tasks, while the IoT device incurs minimal overhead. Public key generation requires 398.969 ms on the SSI agent, whereas the IoT device completes it in only 8.540 ms, resulting in a total time of 407.509 ms. Signature generation, the most time-consuming operation, accounts for 875.874 ms on the SSI agent and 12.118 ms on the IoT device, reaching a total of 887.992 ms. Encryption also demands a significant amount of time, with 390.832 ms attributed to the SSI agent and 36.935 ms to the IoT device, leading to a total encryption time of 427.767 ms. Similarly, decryption takes 387.641 ms on the SSI agent and 35.227 ms on the IoT device, summing up to 422.918 ms. To give a complete overview of the entire process we give insights about the connection setup, and the generation of verifiable presentation. Connection setup, which involves decryption, public key generation, and encryption, results in a total time of 1.258 seconds. Verifiable presentation generation, executed after the establishment of a secure connection incorporating both signature generation and encryption, requires 1.315 seconds. Finally, the data sensing process, considered as a complete loop of request and response, takes 850.685 ms.

The results, summarized in Table III, indicate that the implementation of SSI in a distributed setup introduces a marginal processing time. In particular, the introduction of a distributed approach only increases the distributed authentication time of a marginal time, with respect to the classical approach. Anyway, as a huge improvement, the proposed approach is able to guarantee authentication and confidentiality in 2.573 seconds. The results, also confirms a good time for the data exchange process which is a results comparable with similar approaches in the literature [12], where connection must be established every time with 115.8 ms and 8ms for the publishing; compared with our approach consisting of only 36.395ms for the encryption of the data,

averaged on 1KB of data, since connection is fixed once for the entire session and does not need to be established each time like in traditional MQTT/TLS approaches.

TABLE III
OVERHEAD INTRODUCED BY THE PROPOSED SOLUTION

	SSI Agent	IoT Device	Total
Public Key Gen.	398.969 ms	8.540 ms	407.509 ms
Signature Gen.	875.874 ms	12.118 ms	887.992 ms
Encryption	390.832 ms	36.935 ms	427.767 ms
Decryption	387.641 ms	35.227 ms	422.918 ms
Connection Setup (Decr. + Public Key Gen. + Encr.)			1.258 s
Verifiable Presentation Gen. (Sign. + Encr.)			1.315 s
Data Sensing (Decr. + Encr.)			850.685 ms

V. CONCLUSION

Authentication between the application and end device is a critical component in FIWARE, demanding the first to fully manage it. In this paper, we proposed a novel approach to solve this criticality while taking into account the typical resource challenges of IoT devices. The results presented in this work demonstrate the effectiveness of MQTT as a low-power protocol and a combination with DIDComm is able to strengthen the security of the authentication loop. The proposed approach not only overcomes existing technical challenges but also opens new opportunities for future device integration within the IoT ecosystem, improving a more secure and resilient environment. In the future we want to assess the quality of the solution in a high-density context, in order to evaluate the impact of the MQTT choice in the system design, also with respect to the choice of Quality of Service (QoS) adopted by the broker. Further analysis for scalability and heterogeneity across different IoT devices will be provided. Optimization of the handling of duplicate messages, as an alternative to traditional QoS mechanisms, could further enhance the system's overall efficiency and resource utilization.

DATA AVAILABILITY

All source code and implementation details related to this study are publicly available in the following GitHub repository: https://github.com/biagioboi/distribuited-credo-ts.

ACKNOWLEDGMENT

This work was partially supported by project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU.

REFERENCES

[1] Z. H. Toman, L. Hamel, S. H. Toman, M. Graiet, and D. C. G. Valadares, "Formal verification for security and attacks in IoT physical layer," *Journal of Reliable Intelligent Environments*, vol. 10, no. 1, pp. 73–91, 2024.

[2] S. Lakshminarayana, A. Praseed, and P. S. Thilagam, "Securing the IoT Application Layer from an MQTT Protocol Perspective: Challenges and Research Prospects," *IEEE Communications Surveys & Tutorials*, 2024.

[3] I. Singh and B. Singh, "Access management of IoT devices using access control mechanism and decentralized authentication: A review," *Measurement: Sensors*, vol. 25, p. 100591, 2023.

[4] FIWARE Foundation, "FIWARE: The Open Source Platform for Our Smart Digital Future." https://www.fiware.org, 2020. Accessed: 2025-04-01.

[5] P. Salhofer, "Evaluating the FIWARE platform," 2018.

[6] X. Fan, Q. Chai, L. Xu, and D. Guo, "DIAM-IoT: A Decentralized Identity and Access Management Framework for Internet of Things," in *Proceedings of the 2nd ACM international symposium on blockchain and secure critical infrastructure*, pp. 186–191, 2020.

[7] S. K. Gebresilassie, J. Rafferty, P. Morrow, L. Chen, M. Abu-Tair, and Z. Cui, "Distributed, secure, self-sovereign identity for IoT devices," in *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)*, pp. 1–6, IEEE, 2020.

[8] M. Abdelrazig Abubakar, Z. Jaroucheh, A. Al-Dubai, and X. Liu, "Blockchain-based identity and authentication scheme for MQTT protocol," in *Proceedings of the 2021 3rd International Conference on Blockchain Technology*, pp. 73–81, 2021.

[9] U. Khalid, M. Asim, T. Baker, P. C. Hung, M. A. Tariq, and L. Rafferty, "A decentralized lightweight blockchain-based authentication mechanism for IoT systems," *Cluster Computing*, vol. 23, no. 3, pp. 2067–2087, 2020.

[10] A. Dixit, M. Smith-Creasey, and M. Rajarajan, "A Decentralized IIoT Identity Framework based on Self-Sovereign Identity using Blockchain," pp. 335–338, IEEE, 2022.

[11] E. Fathalla and Y. Azab, "Towards a lightweight self-sovereign identity framework for iot network in a zero trust environment," in *2024 IEEE 15th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, pp. 335–341, IEEE, 2024.

[12] A. Philipp, A. Küpper, and P. Raschke, "DAXiot: A Decentralized Authentication and Authorization Scheme for Dynamic IoT Networks," in *2024 27th Conference on Innovation in Clouds, Internet and Networks (ICIN)*, pp. 01–07, IEEE, 2024.

[13] S. M. Hosseini, J. Ferreira, and P. C. Bartolomeu, "Blockchain-Based Decentralized Identification in IoT: An Overview of Existing Frameworks and Their Limitations," *Electronics*, vol. 12, no. 6, p. 1283, 2023.

[14] S. Loss, H. P. Singh, N. Cacho, and F. Lopes, "Using FIWARE and blockchain in smart cities solutions," *Cluster Computing*, vol. 26, no. 4, pp. 2115–2128, 2023.

[15] G. V. R. Lakshmi, R. Deeptha, and K. V. Sharma, "IoT Node Authentication Using Du-KAuth with Strong Access Control Model in Smart City Application," in *International Conference on Artificial Intelligence and Smart Energy*, pp. 447–457, Springer, 2024.

[16] D. J. Bernstein, "Curve25519: new diffie-hellman speed records," in *Public Key Cryptography-PKC 2006: 9th International Conference on Theory and Practice in Public-Key Cryptography, New York, NY, USA, April 24-26, 2006. Proceedings 9*, pp. 207–228, Springer, 2006.

[17] J. Brendel, C. Cremers, D. Jackson, and M. Zhao, "The provable security of ed25519: theory and practice," in *2021 IEEE Symposium on Security and Privacy (SP)*, pp. 1659–1676, IEEE, 2021.

[18] F. De Santis, A. Schauer, and G. Sigl, "ChaCha20-Poly1305 authenticated encryption for high-speed embedded IoT applications," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, pp. 692–697, IEEE, 2017.

[19] B. Blanchet *et al.*, "Modeling and verifying security protocols with the applied pi calculus and ProVerif," *Foundations and Trends® in Privacy and Security*, vol. 1, no. 1-2, pp. 1–135, 2016.