Article

Not peer-reviewed version

Enhancing Resilience in Digital Twins: ASCON-Based Security Solutions for Industry 4.0

Mohammed El-Hajj * and Teklit H. GEBREMARIAM

Posted Date: 11 July 2024

doi: 10.20944/preprints202407.0963.v1

Keywords: Digital Twins; DT; LightWeight; ASCON; IIoT; ESP32

Article

Enhancing Resilience in Digital Twins: ASCON-Based Security Solutions for Industry 4.0

Mohammed El-Hajj †,* **and T.H. GEBREMARIAM** †

Department of Semantics, Cybersecurity & Services, University of Twente; m.elhajj@utwente.nl

* Correspondence: m.elhajj@utwente.nl

† These authors contributed equally to this work.

Abstract: The persistent security challenges in Industry 4.0 due to resource-constrained IoT devices necessitate innovative solutions. Addressing this, a study introduces the ASCON algorithm for lightweight Authenticated Encryption with Associated Data, enhancing confidentiality, integrity, and authenticity within IoT limitations. By integrating Digital Twins, the framework emphasizes the need for robust security in Industry 4.0, with ASCON ensuring secure data transmission, bolstering system resilience against cyber threats. Practical validation using MQTT protocol confirms ASCON's efficacy over AES-GCM, highlighting its potential for enhanced security in Industry 4.0. Future research should focus on optimizing ASCON for microprocessors and developing secure remote access tailored to resource-constrained devices, ensuring adaptability in the digital era.

Keywords: Digital Twins; DT; lightweight; ASCON; IIoT; ESP32

1. Introduction

In the ever-evolving landscape of the Internet of Things (IoT) and its expanding realms, the integration of Digital Twins (DTs) has emerged as a transformative concept, promising enhanced monitoring, analysis, and control of physical assets [1]. However, as the proliferation of IoT devices continues relentlessly, a critical concern surfaces: ensuring the security of communication channels between DTs and resource-constrained IoT endpoints [2].

Despite significant advancements, the current body of research on DT security reveals notable limitations. Many IoT devices, such as sensors, actuators, and RFID tags, operate under significant resource constraints, including limited computational power, memory, and energy reserves [3]. Conventional encryption methods and security mechanisms, often relied upon in existing studies, are too computationally demanding and impractical for these devices. Consequently, there is a compelling need for a lightweight, efficient, and robust cryptographic algorithm to secure the communication channels between DTs and their resource-constrained counterparts.

A detailed review of existing literature highlights several research gaps. Notably, there is a lack of focus on ensuring the authenticity and integrity of sensor data fed into the Digital Twin [4]. While some papers discuss securing data transmission channels, their reliance on traditional encryption and authentication mechanisms such as AES, SHA-256, and RSA is concerning. These methods are not feasible for deployment on resource-constrained devices, underscoring the urgent need for lightweight alternatives.

In response to these challenges, this paper introduces a novel cryptographic solution tailored to the unique demands of the Digital Twin ecosystem. Our objective is to provide a lightweight, secure communication channel that seamlessly operates between DTs and the often resource-constrained IoT devices. We aim to achieve a nuanced equilibrium between robust security and efficient resource utilization, enabling the unhindered flow of critical data while safeguarding against cyber threats.

Our proposed solution, grounded in the Authenticated Encryption with Associated Data (AEAD) algorithm ASCON ([5]), offers a promising approach to bridge the security gap in this dynamic and evolving domain. This paper outlines our journey in addressing this crucial issue, exploring the existing security landscape in DT applications, identifying limitations, and proposing an innovative cryptographic algorithm that enhances security without consuming additional resources on resource-constrained IoT devices.

 doi:10.20944/preprints202407.0963.v1

The structure of the remainder of this paper is as follows: Section 2 undertakes a comprehensive literature study, shedding light on the existing state of security in Digital Twin communication and noting challenges and gaps in current solutions. In Section 3 , we present our novel, lightweight cryptographic method for ensuring an effective secure communication channel. Sections 4 and 6 delve into comprehensive performance and security evaluations, demonstrating the practicality and resilience of our proposed solution.In Section 7, we discuss the performance and security analysis results, the scalability of ASCON in large-scale IoT applications, and the limitations of the ASCON algorithm. The paper concludes with Section 8, summarizing major findings, highlighting the significance of our solution, and suggesting possible directions for further investigation in this important area.

2. Literature Review

This section will comprehensively review the existing literature, encompassing academic research, industry reports, and practical implementations about secure communication between DTs and resource-constrained IoT devices.In our previous SLR work[6], We will analyze a wide array of studies and publications to understand the current state of knowledge in this domain comprehensively. A critical aspect of our literature review will involve assessing existing solutions and technologies utilized in practice. We will evaluate these solutions' effectiveness, limitations, and real-world applicability in securing the communication channels between DTs and resource-constrained IoT devices. Based on our thorough literature review, we will clearly articulate the specific research objectives and questions that our paper aims to address. These objectives will guide our exploration of novel approaches and the development of a lightweight cryptographic solution tailored to the needs of this emerging field. It is essential to have secure communication between the physical and digital components to ensure the reliability and security of DT-based systems. Those physical IoT components' computational, power, and storage limitations must be considered.

In this regard, we analyzed 14 papers that discuss data confidentiality, integrity, and privacy in the Digital Twin ecosystem. Table 1 provides a summary of the security mechanisms employed in the literature.

Table 1. Security Mechanisms for Securing Data in DT and Industrial Internet of Things Communication.

Ref	Security Mechanism(s)	Goal(s)
[7]	Central access control system based on OAuth and XACML	Secure access control
[8]	Anonymous communication based on secret-handshake scheme and group signature	Unforgeability and conditional traceability (privacy)
[9]	Differential privacy techniques	Privacy and confidentiality of data
[10]	Blockchain and Smart contract-based Proof-of-Authentication (PoA)	Validate the legitimacy and integrity of data collected from Industrial Internet of Things nodes.
[11]	Blockchain, Smart contract, and Deep learning	Integrity of data, detect botnet behavior
[12]	Quantum communication technologies	Improve overall security of communication between DT and IIoT
[13]	Trusted Execution Environment and Unclonable Functions (PUFs)	Security and Trustworthiness of communication
[14]	Attribute-based Access Control	Secure data storage
[15]	Blockchain	Authenticate data generated from clusters before they are used in DT
[16]	Authorization Blockchain and Storage Blockchain	Secure data sharing through authorization
[17]	Blockchain and SHA-256 hash for chained checksum	Increase the security and trustworthiness of sensor readings for Digital Twin applications.
[18]	Blockchain, access control secure transmission protocol	Improve the communication security of Internet of Vehicles (IoV).
[19]	Framework based on verifiable data register (VDR) and credentials	Secure and protect the privacy of data exchange in the Digital Twin ecosystem.
[20]	Attribute-Based Encryption (ABE) and Symmetric encryption scheme	Ensure the secure communication of Digital Twin and IoT.

 doi:10.20944/preprints202407.0963.v1

To address security challenges like communication trust and privacy protection, the authors in [8] introduced a robust framework for secure vehicular digital twin communication. This framework leveraged anonymous authentication as a key component. The authors introduced a specific authentication protocol that relied on a secret handshake scheme and group signature to accomplish this. This approach effectively addressed issues related to the authenticity of messages and conditional traceability. The proposed framework established secure communication channels not only between iTwins (Digital Twins) and their physical counterparts but also among iTwins themselves, ensuring the confidentiality and integrity of the transmitted data.

To address the challenges of data volume, unreliable communication channels, and security threats in Industrial IoT (IIoT), authors in [10] proposed an integrated framework that combines blockchain and Deep Learning (DL). This framework introduces a new DT model for simulating security-critical IIoT processes and employs blockchain-based data transmission with smart contracts to ensure data integrity. The DL scheme focuses on applying an Intrusion Detection System (IDS) to valid data retrieved from the blockchain. The practical implementation demonstrates substantial improvements in communication security and data privacy.

In [11], authors focused on enhancing the security of communication between IoT devices and DTs. They achieved this by employing a private blockchain, smart contracts, and deep learning techniques for network traffic monitoring. The utilization of a private blockchain and smart contracts played a crucial role in guaranteeing the security and integrity of data flow between physical devices and DTs, rendering it tamper-proof. Additionally, the deep learning model proved invaluable in the early detection of botnet behavior, triggering alerts for security intervention to isolate infected devices. This comprehensive approach effectively maintained the security of communication and data integrity.

In the study carried out by [12], the primary objective was to enhance the communication security between IIoT devices and DTs through the application of quantum communication technologies. The authors introduced a channel encryption scheme rooted in quantum communication principles, utilizing entanglement states and quantum teleportation. Additionally, they put forward the concept of an Adaptive Key Residue algorithm, built upon a quantum key distribution mechanism. This innovative approach was designed to significantly enhance the security of communication between IIoT devices and DTs. In their paper [14], the authors proposed a robust and privacy-preserving scheme tailored for digital twin-based traffic control. The scheme operates in two pivotal phases. Firstly, in the data uploading phase, they employed a group signature technique with time-bound keys, ensuring data source authentication, efficient member revocation, and privacy protection to secure data storage on cloud service providers once synchronization with the digital twin occurs. In the subsequent data-sharing stage, a secure and efficient attribute-based access control technique is utilized, allowing for flexible and efficient data sharing. Notably, this approach stores specific sub-policy parameters during the initial decryption, reducing computational complexity for subsequent data access with the same sub-policy. Additionally, the paper provides a theoretical analysis, validating the scheme's security and efficiency.

In [13], authors addressed the security and trustworthiness of the communication between the digital twin and physical device through various technologies and HW and SW solutions such as Trusted Execution Environment platforms and Physically Unclonable Functions (PUF)s for device authentication. In addition, blockchain technology, which provided secure, immutable, and auditable data storage for the exchanged critical data, was investigated by the authors. The authors in [15] proposed a secure smart manufacturing framework through the integration of Digital Twin (DT) and Blockchain technologies. The framework aimed to facilitate efficient and secure multi-party collaborative information processing in heterogeneous IIoT environments. Notably, the paper demonstrated that the proposed authentication mode outperformed the standard protocol in terms of time efficiency. Although the paper did not provide detailed information on other methods employed in the framework, it highlighted simulation results. In conclusion, the authors suggested the future inclusion of

 doi:10.20944/preprints202407.0963.v1

quantum computing technology to further enhance the overall efficiency and security of the proposed framework.

In [16] authors proposed a data security sharing architecture based on a dual Blockchain network to solve the security problems of the Internet of Things. The first blockchain called the authorization Blockchain, was used for permission control and consensus, and the other, called the storage Blockchain, was used for the storage of data bodies. The proposed architecture was applied to the Internet of Things system based on Digital Twin to address the data security transmission between the physical system, digital twin system, and IoT application system. However, the authors in this study provided only data authentication. They assumed the data from IoT devices was encrypted on transmission.

In [17], a novel use of the lightweight SHA-256 hash algorithm was proposed to create a blockchain of sensor readings, ensuring trustworthy communication between the control center and remote sensors. By chaining the checksums of current and previous readings, the implementation established trust based on the unbroken linked list length. The authors in this paper claimed that this approach strengthened the security and trustworthiness of sensor data in digital twin applications, particularly in high-value domains such as the power grid.

Authors in [18] proposed BC-Based IoV Secure Communication Framework, presenting an architecture designed to enhance secure communication in the context of the Internet of Vehicles (IoV). By leveraging blockchain technology, the authors claimed the framework securely stored vital data such as public keys and communication history. It consisted of five key modules: BC network, access control, secure transmission protocol, vehicle Ad Hoc, and a Sybil attack detection mechanism. To combat the rising prevalence of Sybil attacks in IoV scenarios, the framework utilized regular location certificates issued by base stations, which served to validate vehicle location accuracy. This proposed framework offered a viable solution to enhance communication security in IoV environments.

In [19] the authors introduced a framework called SIGNED, which aimed to enable a secure and verifiable exchange of digital twin data in a smart city context. The framework focused on data ownership, selective disclosure, and verifiability principles using Verifiable Credentials. It consisted of five functional components: Cyber & Physical Layer, Workflow Designer, Analysis Layer, Traceability Layer, and Digital Wallet. The Traceability Layer, integrated with a blockchain-based Verifiable Data Registry, maintained the public credentials and tracked registered assets. The authors presented a proof of concept using a smart water management use case to demonstrate the effectiveness of SIGNED in ensuring trusted and verifiable data exchange, with minimal performance impact. Overall, the framework provided enhanced security and privacy when sharing data between different functional units in a smart city. A contribution by [20] presented work to enhance IoT communication security in digital twin networking. They proposed an interference source location scheme with a mobile tracker to reduce attacks, improve resistance, and enhance Attribute-Based Encryption (ABE). They use access control policy and symmetric encryption to secure key exchange. To address observation noise through an unscented Kalman filter, the paper modifies interference source location. The authors in this work concluded that utilizing Jamming Signal Strength (JSS) information with the untracked Kalman filter algorithm can effectively estimate the interference source location and other related state information.

2.1. Security Mechanisms Analysis From Literature

Securing the communication channel between Digital Twin and Industrial Internet of Things deployment is a critical concept that should not be neglected especially in the critical infrastructure of Industry 4.0. To address this, a few research efforts on various security mechanisms were presented in the literature. In this subsection, we present a comparative analysis of security mechanisms for secure data communication in terms of practicality resource efficiency, and deployment.

The most widely used approach to provide privacy and security in the Digital Twin ecosystem in the existing literature is Blockchain (Smart Contract) technology. In [10,11,15–18] Blockchain-based data transmission scheme for data integrity are proposed. Due to the inherent nature of Blockchain, the proposed solution based on this technology has a limitation in providing data confidentiality.

 doi:10.20944/preprints202407.0963.v1

Blockchain-based security approaches offer data integrity in a distributed environment, but they may have computationally demanding underlying technology, impacting their suitability for resource-constrained IoT devices.

Three studies [8,9,14,19] focused on providing privacy using techniques such as secret-handshake scheme, group signature and differential privacy techniques. While enhancing privacy, these two approaches may require significant computational resources for cryptographic operations on resource-constrained Industrial Internet of Things devices.

Furthermore, we encountered security mechanisms that focus on access control and trust [7,13,14]. The first paper suggests a centralized access control system using XACML policies and tokens like SAML and OAuth to regulate access and ensure communication security. In another paper, the authors proposed a scheme for secure data sharing using attribute-based access control. In the third paper, the authors propose secure data exchange between Digital Twin and Industrial Internet of Things using technologies namely PUF (Physical Unclonable Functions) and TPM (Trusted Platform Module). Though these solutions are resource efficient, it might not be economically practical to use them as the special hardware setup and complex key management involved.

Lastly, we explored a unique study [12] based on quantum communication and quantum entanglement. It is a theoretical proposed solution that may not even be possible shortly as this technology has not yet developed. Therefore, quantum communication might provide very efficient and strong security but it might also require sophisticated hardware, which makes its practical implementation challenging.

2.2. Research Gap

In our review of selected papers, it became evident that most of the papers placed little emphasis on ensuring the authenticity and integrity of the sensor data that is fed into the Digital Twin. Even though a handful of papers discussed securing the data transmission channel, their recommendations relied on traditional encryption and authentication mechanisms such as AES, SHA-256, and RSA. This research gap and these proposals are concerning because in most use cases, the field sensors are power constraints where it is not feasible to deploy traditional encryption algorithms to secure them. Hence, it is important in future research to focus on lightweight algorithms to protect data confidentiality, integrity, and authenticity of data used in Digital Twin-based solutions.

The application of Digital Twins for security in Industry 4.0 is at its early stage. While researchers have made significant contributions to its development, there are remaining research gaps that still require exploration and improvement. In this section, we identify and discuss three potential research areas.

- Efficient lightweight encryption algorithms: As the development of Digital Twin technology progresses, it is expected that it will become accurate in replicating physical objects and processes. To achieve this level of accuracy, a large number of tiny, resource-constrained IoT sensors will need to be deployed on a massive scale to measure every aspect of the physical status being replicated. This presents future research directions for designing and implementing efficient encryption algorithms that can be deployed on resource-constrained devices.
- Remote access control for DT: One area of research that we have identified as a gap in the literature is the secure remote access control to the virtual counterpart of an ICS component for vendors to perform troubleshooting and testing. In the traditional real-world industry setup, vendors of ICS components have remote access control to the physical object of the industry for various reasons. However, it is not clear how this is going to be handled on the DT yet. One potential direction for research is to explore and investigate how secure remote access can be achieved to one or more components of the DT.
- Human-computer interaction: Finally, future research could explore the human-computer interaction (HCI) aspect of DT technology. This could involve examining how users interact with DT models and exploring new and innovative ways to improve the user experience. By improving

 doi:10.20944/preprints202407.0963.v1

the HCI aspect of DT technology, it may be possible to enhance the accuracy and reliability of the models by ensuring that human error is minimized.

3. Proposed Solution

The low-power and resource-constrained IIoT devices are incapable of running traditional cryptographic schemes such as AES, SHA-256, and RSA. Regardless, these devices are widely used for various Industry 4.0 applications across a range of industry sectors, such as manufacturing, transportation, health, and power grids. Moreover, with the emergence of DT in Industry 4.0, Industrial Internet of Things sensors are an integral part of Digital Twin technology, in which they are used to collect and send data over wired or wireless channels. Hence, it is crucial to secure the communication between the DT and Industrial Internet of Things, taking into consideration the limited resources they have. In light of the challenges and the gap mentioned in section 2.2, our proposed solution delves into the heart of this matter, recognizing the urgency to establish a secure communication channel that bridges the physical devices in IIoT applications with their mapped Digital Twins. This section presents innovative approaches, focusing on lightweight algorithms tailored to preserve data confidentiality, integrity, and authenticity. By addressing this critical need, we aim to not only enhance the security posture of IIoT applications but also unlock the full potential of Digital Twin technology. In this work, we proposed a resource-efficient communication scheme based on lightweight cryptographic authenticated encryption to enhance the security of the communication channel between the Digital Twin, Eclipse Ditto [21], and its physical components over the Message Queuing Telemetry Transport (MQTT) protocol using a technique called payload encryption.

In the following subsections, we will first describe the design considerations and the requirements (section 3.1) we considered, the general description of the implementation approach (section 3.2), and the experimentation setup steps we took, starting from setting up Eclipse Ditto (section 3.3), Mosquitto MQTT broker (section 3.4), implementing and integrating the selected AEAD algorithms (sections 3.5 and 3.6) and ending to demonstrating the functioning experimentation setup (section 3.7).

End-To-End Payload Encryption and Authentication: Our communication scheme is based on payload authenticated encryption using one of the AEAD (authenticated encryption with associated data) algorithms over the MQTT protocol.

3.1. Design Considerations and Requirements

This section highlights the design factors and requirements for developing our proposed communication solution, ensuring secure communication between Digital Twins and IIoT devices. These considerations are tailored specifically to address the resource limitations of IIoT devices. This section will discuss these considerations and define in- and out-of-scope requirements.

In our proposed solution, we have carefully considered the limitations of IIoT devices. To address these constraints, we have made specific design choices. Firstly, the scheme incorporates a lightweight application protocol tailored to the restricted resources of these devices. Secondly, the underlying cryptographic algorithm is selected based on a NIST-standardized lightweight encryption algorithm. These choices ensure our solution is optimized for the efficient operation of IIoT devices, enhancing their overall performance and security. Our proposed solution is based on the following design choices that take into account the limited resource Industrial Internet of Things devices have.

- The application protocol should be lightweight.
- The underlying cryptographic algorithm should be based on a lightweight, NIST-standardised encryption algorithm.

3.1.1. In Scope Requirements

Our in-scope requirements included one performance requirement and several requirements related to security. From the performance perspective, the underlying cryptographic algorithm of our proposed solution should be more efficient than the traditional algorithms in terms of power

 doi:10.20944/preprints202407.0963.v1

consumption, speed, and storage complexity. Furthermore, from the security perspective, the proposed solution should provide an adequate security level for typical data communication in an Industry 4.0 environment, including ensuring message/data confidentiality, integrity, and authenticity. In this regard, an encryption algorithm that provides a minimum 80-bit security level (size of key) should be used. The proposed solution encompasses two key requirements: performance and security. From a performance perspective, the solution must outperform traditional algorithms in terms of power consumption, speed, and storage complexity. In the realm of security, it should provide an adequate level of protection for data communication within an Industry 4.0 environment, necessitating the use of an encryption algorithm with a minimum 128-bit security level (key size). Moreover, the security requirement extends to specific services, including authentication, confidentiality, and data integrity.

Performance Requirement: The proposed solution should be based on a cryptographic algorithm performing better than traditional algorithms in terms of power consumption, speed, and storage complexity.

Security Requirements: the proposed solution should provide an adequate security level for typical data communication in an Industry 4.0 environment, including ensuring message/data confidentiality, integrity, and authenticity. In this regard, an encryption algorithm that provides a minimum 80-bit security level (size of key) should be used. In addition to the above general security requirement, the proposed solution should provide the following security services.

- *Message authentication*: The solution should enable the message receiver to check the authenticity of the message
- *Message Confidentiality*: The solution should provide message confidentiality by encrypting the message.
- *Data Integrity*: The solution should ensure the integrity of data transmission using checksums or other methods to detect message corruption.
- *Resilience*: The solution should be capable of detecting man-in-the-middle attacks that involve message modification and data injection.

3.1.2. Out of Scope Requirements

Even though proper key management is essential, it is an out-of-scope requirement for us. We assume that symmetric keys are pre-shared before communication. Additionally, we do not incorporate encryption technologies like SSL/TLS on the application level to avoid computation overhead on resource-constrained IoT devices.

- The proposed solution does not cover a secure key exchange mechanism between the communicating parties. Instead, symmetric keys are assumed to be pre-shared before communication.
- The communication protocol (MQTT) at the application level is not encrypted using technologies like SSL/TLS to avoid computation overhead on the constrained device.

3.2. Implementation Approach

To validate the applicability and effectiveness of our proposed solution, which relies on lightweight cryptographic algorithms, we conducted an experiment employing an ESP32 as a resource-constrained IoT device and Ditto as an open-source framework for constructing Digital Twins. We chose the ESP32 boards for the experiment because they are known for their cost-effectiveness and low-power characteristics in the market ([22]). Furthermore, we selected Ditto based on its being easily customized through Java-based plugins and its extensive usage within the open-source community. Figure 1 illustrates the research experiment setup. The measurement tool in the research experiment setup in Figure 1 was used for performance analysis.

Payload encryption is a technique for ensuring message confidentiality at the application level. In other words, this approach can be used to establish an end-to-end secure channel between the sender and receiver at the application level to provide confidentiality over transmitted data. However, in this paper, we extend the scope beyond message confidentiality and introduce the use of the ASCON

 doi:10.20944/preprints202407.0963.v1

([5]), a family of authenticated encryption with associated data (AEAD) algorithms, to provide both message authenticity and confidentiality[5].

We chose to utilize the MQTT protocol for three reasons. Firstly, MQTT is a commonly used communication protocol in IoT applications. Secondly, it is a lightweight messaging protocol, that can be configured and programmed to support payload encryption using any encryption algorithm, like the algorithms we chose for comparison, namely ASCON and AES-GCM (family of AEAD based on AES). Thirdly, Eclipse Ditto [21] supports MQTT protocol, i.e. through Eclipse Mosquitto [23], as a connection.

To assess the practicality and efficiency of our proposed solution, centered around lightweight cryptographic algorithms, an experiment was conducted using an ESP32 device, a cost-effective and low-power IoT board [22]. We employed Ditto, an open-source framework for Digital Twins, recognized for its customization capabilities through Java-based plugins and widespread usage in the open-source community. The experimental setup is depicted in Figure 1.

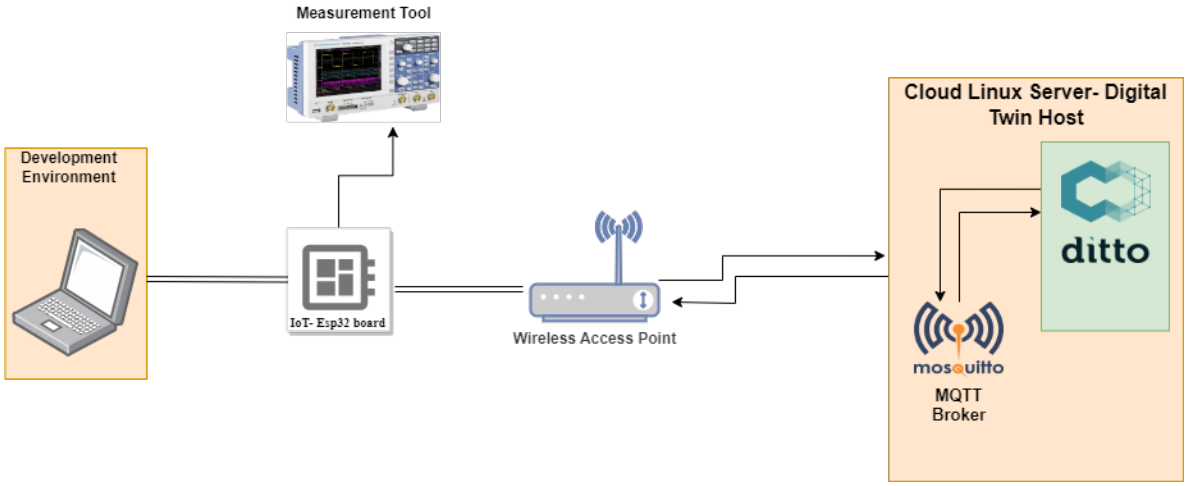


Figure 1. Research Experiment Setup.

Traditionally, payload encryption is utilized to ensure message confidentiality, establishing a secure channel between sender and receiver at the application level. In this study, we expand this concept by introducing ASCON, a family of authenticated encryption with associated data (AEAD) algorithms [5]. ASCON not only ensures message confidentiality but also guarantees message authenticity, enhancing security measures. The choice of MQTT protocol was deliberate for several reasons. Firstly, MQTT is widely used in IoT applications. Secondly, it is a lightweight messaging protocol that can be configured to support payload encryption using various algorithms, including ASCON and AES-GCM (an AEAD algorithm based on AES), which we utilized for comparison. Lastly, Eclipse Ditto, supported by Eclipse Mosquitto [23], integrates seamlessly with the MQTT protocol, enabling a robust connection for secure data transmission[24]. This strategic selection of components allowed us to thoroughly evaluate the effectiveness and practicality of our proposed approach.

3.3. Eclipse Ditto - Digital Twin Setup

Eclipse Ditto, an open-source framework, serves as a platform for managing IoT devices to create Digital Twins. Integrated with MQTT brokers, it enables Digital Twins to connect with various backend systems using protocols such as AMQP, Apache Kafka, HTTP, and MQTT. Ditto can be deployed either on-premises or in the cloud. For this research, we deployed Ditto in the cloud, utilizing Docker as it requires fewer resources than the alternative Kubernetes cluster setup. Within a Docker container, we have a collection of five microservices that constitute Ditto, as well as two additional components. These microservices encompass Policies for defining access rights, Things for

 doi:10.20944/preprints202407.0963.v1

handling physical components, Things-Search for searching within the datastore, Gateway for HTTP and WebSocket API, and Connectivity for connectivity management. Additionally, an nginx instance serves as a reverse proxy, offering basic authentication functionality. Furthermore, we utilize MongoDB as the underlying datastore for Ditto. Running Ditto in a docker container has seven microservices operating in parallel, each fulfilling distinct functions. These microservices include *Nginx* as the web server, *Ditto Connectivity* for managing the device-to-Ditto connectivity, *Ditto Thing* for managing things (counterpart of physical devices), *Thing Search* for facilitating efficient search using MongoDB, *Swagger-UI* for providing a web-based user interface, and *Ditto Policies* for maintaining controlled access over things. Since we aimed to test and compare the performance of different encryption algorithms, we utilized one of the Ditto example projects[1] [2] with minimal changes for setting up our experimentation system consisting of Ditto and a single ESP32 board. These changes included, for example, giving access rights in policies, defining ESP32 as a thing with humidity and temperature as its features, and defining customized payload mapper and published/subscribed MQTT broker topics in the connection configuration file.

3.4. MQTT Broker

We chose to utilise the MQTT protocol for three reasons. Firstly, MQTT is a commonly used communication protocol in IoT applications. Secondly, it is a lightweight messaging protocol, that can be configured and programmed to support payload encryption using any encryption algorithm, like the algorithms we chose for comparison, namely ASCON and AES-GCM (family of AEAD based on AES). Thirdly, Eclipse Ditto [21] supports MQTT protocol, i.e. through Eclipse Mosquitto [23], as a connection.

The MQTT broker, designed for IoT communication, is crucial in our project. We opted for Eclipse Ditto's MQTT implementation, i.e. Mosquitto ([23]). Initially, we aimed to customize and control the MQTT implementation by building it from the source on our Linux server. This approach allowed us to directly incorporate a lightweight encryption algorithm into the code. However, we chose to implement these algorithms by extending the Ditto source code itself, using the provided connectivity extension. We utilized the connectivity extensions provided by Ditto, which we further extended to utilise our AEAD encryption algorithms. However, an external MQTT broker can also be used by installing it on the same machine as Ditto (Digital Twin) or on a separate, remotely accessible machine. Our scheme ensures secure IoT device communication with the cloud-hosted Ditto service.

The MQTT broker plays a pivotal role in our IoT communication project. We chose Eclipse Ditto's MQTT implementation, specifically Mosquitto. Leveraging Ditto's connectivity extensions, we extended them to incorporate our AEAD encryption algorithms. Alternatively, an external MQTT broker can be utilized on the same machine as Ditto (Digital Twin) or a separate remote machine. Our approach guarantees secure communication between IoT devices and the cloud-hosted Ditto service. Despite MQTT not inherently being a secure protocol, our MQTT broker functions as a proxy, forwarding encrypted payload messages without access to IoT devices' encryption keys. Consequently, only the involved parties can decrypt and authenticate the payload. This entire process is visualized in Figure 2. The MQTT broker's limited access means it can only interact with the encrypted payload, preventing any malicious broker from compromising communication security. Our lightweight authentication and encryption algorithm, integrated into the solution, guarantees the confidentiality and integrity of the exchanged data.

Even though MQTT is not a secure protocol, in our case, the MQTT broker acts merely as a proxy forwarding encrypted payload messages and does not have the shared encryption keys of IoT devices.

---

[1] https://github.com/eclipse-ditto/ditto-examples/tree/master/mqtt-quick-introduction
[2] https://github.com/eclipse-ditto/ditto-examples/tree/master/mqtt-bidirectional

 doi:10.20944/preprints202407.0963.v1

Hence, only the communication parties can decrypt and authenticate the payload. The whole process is illustrated in Figure 2.

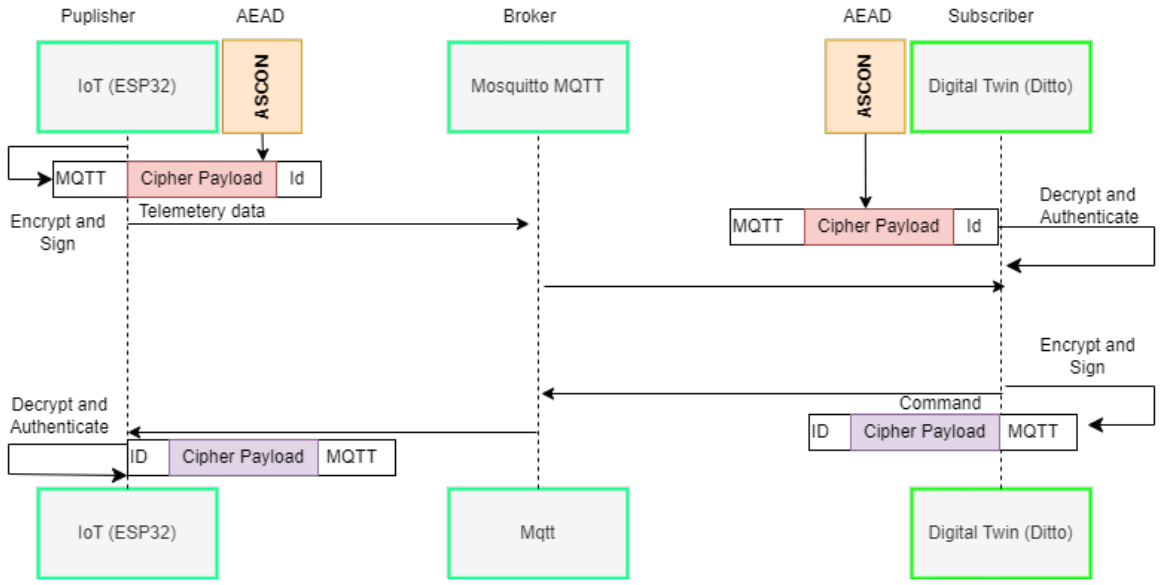


Figure 2. Scheme of Payload Encryption With Authentication Over MQTT Protocol.

The communication process between devices and Ditto involves several key steps. Firstly, each device connected to Ditto is assigned a unique device ID, crucial for ASCON's associative input. Secondly, a shared symmetric key exists between the device and its corresponding Digital Twin. When a message needs to be sent, the sender encrypts the payload and publishes it using MQTT under a topic that includes the device's unique ID. Similarly, the device uses its ID to publish messages for Ditto, creating a connection between the sender and receiver. Upon reception, the receiver decrypts and authenticates the MQTT message payload using the pre-shared symmetric key, ensuring the message's integrity and origin. In cases with multiple IoT devices, the receiver retrieves the appropriate shared key using the device ID, thereby ensuring secure and authenticated communication.

3.5. Payload Encryption Algorithms

Payload encryption is a technique for ensuring message confidentiality at the application level. In other words, this approach can be used to establish an end-to-end secure channel between the sender and receiver at the application level to provide confidentiality over transmitted data. However, in this paper, we extend the scope beyond message confidentiality and introduce the use of the ASCON ([5]), a family of authenticated encryption with associated data (AEAD) algorithms, to provide both message authenticity and confidentiality.

We used both C implementations of both encryption algorithms, ASCON[3] and AES-GCM[4] and Java implementation of ASCON for Ditto. The C language was opted for ESP32, since C and C++ are more suitable for low-level programming in such resource-constrained devices. The ESP32's main application was developed in C++, whereas the encryption algorithms, ASCON and AES-GCM were implemented in C. Since Ditto enables extending connectivity modules using jar packages, we chose to

[3] https://github.com/ascon/ascon_collection
[4] https://github.com/usnistgov/Lightweight-Cryptography-Benchmarking/tree/main/implementations/_reference_/crypto_aead/aes-gcm/mbedtls

 doi:10.20944/preprints202407.0963.v1

use the Java implementation of the ASCON. For performance analysis purposes, applying encryption algorithms to Ditto was not necessary, but to demonstrate a fully functional system, we incorporated ASCON on the Ditto side.

We selected and deployed C implementations of ASCON [5] and AES-GCM [6] algorithms on the hardware IoT device was carried out using C and C++ programming languages within the Arduino for esp-idf embedded development framework. [7] We chose C and C++ as implementation languages due to their suitability for low-level programming in resource-constrained devices. We implemented the main IoT device application in C++, and chose ASCON[8] and AES-GCM[9] algorithms implemented in C. We then integrated these AEAD algorithms into our C++ main application using the "extern" macro. In the Ditto, only ASCON implementation in Java[10] was used to verify that the connection and encryption/decryption between Ditto and ESP32 works as intended. Since our focus was on measuring performance on the resource-constrained devices, it was not necessary to implement both encryption/decryption algorithms on the Ditto side. Ditto's connectivity module is extendable with Java modules, so we chose the Java implementation of ASCON for decrypting the payloads of the IoT device messages. Importantly, we used ESP32 device-optimized reference implementation of ASCON. We did not alter the algorithms' implementations or introduce any further optimizations. However, to enhance security, we incorporated a nonce generation function, vital for countering security attacks like replay attacks involving the repeated use of encrypted information.

on the hardware IoT device was carried out using C and C++ programming languages within Arduino for esp-idf embedded development framework. We opted for C and C++ because those two choices are more suitable for low-level programming, such as embedded resource constraint devices. The main application for the IoT device was developed in C++, while the algorithm for ASCON and AES-GCM was implemented in C and incorporated through the use of the "extern" macro in the C++ main application.

The counterpart of the algorithms in the digital twin were implemented in Java. This is because the connectivity microservice of Ditto is implemented in Java. This allowed us to extend the connectivity module using Java to incorporate an extension for encryption and decryption of the payload that comes from IoT devices. We did not introduce optimizations in the design of these algorithms. Still, for ASCON we selected the optimised reference C implementation[11]. We also implemented a nonce generation function to provide nonces as inputs for ASCON, which is crucial in addressing security attacks, such as replay attacks, which involve the repeated use of encrypted information. For both algorithms (ASCON [12] and AES-GCM[13]), we selected the optimized reference implementations based on key length of 128 bit tailored for the ESP32 device chip. However, to enhance the security of our

---

5 https://github.com/ascon/ascon_collection
6 https://github.com/usnistgov/Lightweight-Cryptography-Benchmarking/tree/main/implementations/_reference_/crypto_aead/aes-gcm/mbedtls
7 https://github.com/ascon/ascon_collection
8 https://github.com/ascon/ascon-c/tree/29ef7a20a7372bd47fe7f4c92861e58e49cdce94/crypto_aead/ascon128av12/esp32
9 https://github.com/usnistgov/Lightweight-Cryptography-Benchmarking/tree/main/implementations/_reference_/crypto_aead/aes-gcm/mbedtls
10 https://github.com/ascon/javaascon/tree/ccd7622190791f1ba05dc61a9f7b5d286a0a9cb8/src/iaik
11 https://github.com/ascon/ascon-c/tree/29ef7a20a7372bd47fe7f4c92861e58e49cdce94/crypto_aead/ascon128av12/esp32
12 https://github.com/ascon/ascon_collection
13 https://github.com/usnistgov/Lightweight-Cryptography-Benchmarking/tree/main/implementations/_reference_/crypto_aead/aes-gcm/mbedtls

 doi:10.20944/preprints202407.0963.v1

implementation, we incorporated a function to generate a nonce. This aspect is crucial in addressing security attacks, such as replay attacks, which involve the repeated use of encrypted information.

3.6. Ditto Java Base Payload Mapping

In the context of Eclipse Ditto [21], data storage and transfer are facilitated through a format known as the Ditto protocol. This protocol utilizes a JSON structure, employing key-value pairs to represent and transmit information. To seamlessly integrate with Ditto's capabilities, the connectivity microservice bundled with Ditto offers an extension specifically designed for intercepting incoming data. This extension allows for the mapping of data from its original form to a format that Ditto can understand and store in its underlying MongoDB database. Using the Ditto payload mapping feature, we can decrypt incoming encrypted payload messages and convert them into a format that Ditto can process and store. With the payload mapping feature in Ditto's connectivity microservice, we can do the following: receive encrypted data from the IoT device, decrypt and authenticate it, and convert it into Ditto protocol messages. This helps ensure that the data sent between the IoT device and Ditto is secure and authentic. To implement our custom mapping functionality to encrypt and decrypt, we performed the following steps and illustrated in Figure 3:

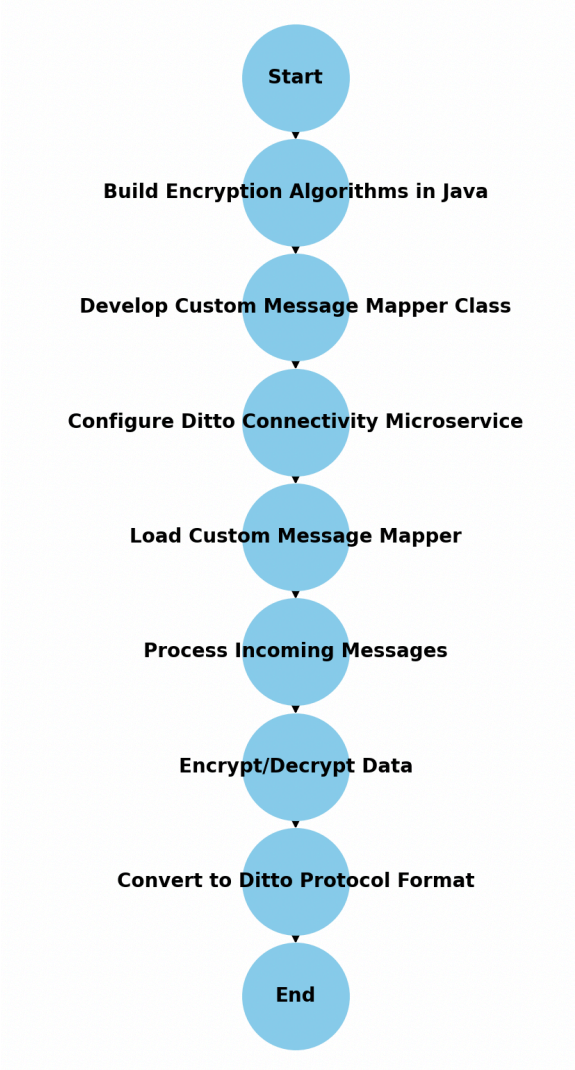


Figure 3. Flow diagram for custom data encryption and decryption in Ditto.

 doi:10.20944/preprints202407.0963.v1

- **Build Encryption Algorithms in Java:**We created a JAR package from the encryption algorithms implemented in Java. This class provides the ASCON or AES-GCM encryption and decryption operations needed for secure communication and data handling.
- **Develop Custom Message Mapper Class:** A custom message mapper class was developed to handle the conversion of incoming device messages to the appropriate Ditto protocol format. This class integrates with the encryption and decryption functionality to ensure data integrity and security during the mapping process.
- **Configure Ditto Connectivity Microservice:**The Ditto connectivity microservice was configured to recognize and load our custom message mapper. This configuration step ensures that incoming messages are routed to our custom mapper for processing, enabling seamless integration of our specific data transformation requirements within the Ditto framework.
- **Load Custom Message Mapper:** The custom message mapper is loaded by the Ditto connectivity microservice, ready to process incoming messages.
- **Process Incoming Messages:** Incoming messages are processed by the custom message mapper.
- **Encrypt/Decrypt Data:** Data within the incoming messages is encrypted or decrypted using the specified encryption algorithms (ASCON or AES-GCM).
- **Convert to Ditto Protocol Format:** The processed and secured data is converted to the Ditto protocol format.

To integrate our encryption jar packages with Ditto, we added the following lines to connection configurations "source": caption=Addition to "sources" in the connection configuration file

```
"payloadMapping": ["AsconPayload"],
"replyTarget": {
  "headerMapping": {},
  "expectedResponseTypes": [
    "response",
    "error"
  ],
  "enabled":false
}
```

And the following lines to the "targets" of the same file:

```
"headerMapping": {},
"payloadMapping": ["AsconPayload"]
```

In these lines, the "AsconPayload" was a reference to our encryption/decryption jar package. Within the Eclipse Ditto framework, data storage and transfer occur using the Ditto protocol, employing JSON format and key-value pairs for information representation. To integrate seamlessly with Ditto, its connectivity microservice offers an extension designed to intercept incoming data. This extension maps data to a format Ditto comprehends, storing it in MongoDB. Utilizing Ditto's payload mapping, encrypted payloads are decrypted and converted, ensuring secure and authentic communication between IoT devices and Ditto. To implement custom mapping for encryption and decryption, specific steps are followed. Firstly, a Java class providing ASCON or AES-GCM encryption/decryption operations is built. Then, a custom message mapper class is developed to convert device messages to the Ditto protocol format. This class integrates encryption/decryption functionality, ensuring data integrity and security during mapping. Finally, Ditto's connectivity microservice is configured to recognize and load the custom mapper, allowing seamless integration of specific data transformation requirements within the Ditto framework.

3.7. Sending Authenticated Encrypted Payload To Ditto

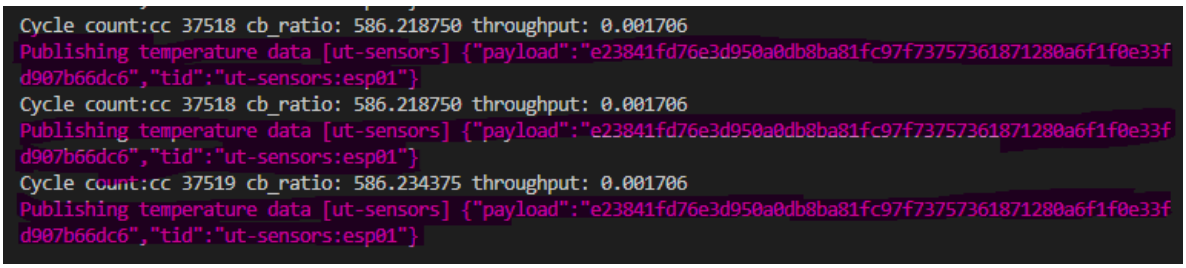
This section demonstrates the proof of concept securing the communication between the IoT device and the Digital Twin (Ditto) using the proposed solution.

Figure 4a depicts a snapshot captured from the serial monitor output of the board (device) utilizing PlatformIO (embedded development framework). The image showcases the device transmitting an

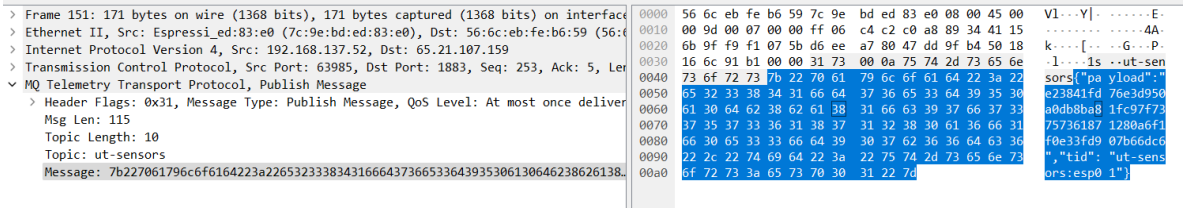
Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 July 2024 doi:10.20944/preprints202407.0963.v1

encrypted payload to the 'ut-sensors' topic while including additional data labeled as 'tid' to uniquely identify the device.

In Figure 4b, a captured packet during the communication is displayed. Upon observation, it becomes evident that the topic being utilized is 'ut-sensors', and the message section of the MQTT protocol header contains the device identifier along with the encrypted payload.



(a) Log Output of ESP32 Device Using Serial Monitor.



(b) Wireshark Captured MQTT Communication From IoT to Ditto.

Figure 4. Serial Monitor of ESP32 Board and Wireshark Capturing Communication Between The Device and Ditto(DT).

MQTT broker, hosted on the same server as Ditto, acts as a proxy, facilitating the transmission of authenticated and encrypted payloads through a publish-subscribe model. Once the MQTT broker receives a payload, it notifies Ditto of the new message it has subscribed to. Ditto then retrieves the payload, decrypts it, and maps it into a Ditto protocol message, which is subsequently stored in a database. The communication process between devices and Ditto involves several key steps. Firstly, each device connected to Ditto is assigned a unique device ID, crucial for ASCON's associative input. Secondly, a shared symmetric key exists between the device and its corresponding Digital Twin. When a message needs to be sent, the sender encrypts the payload and publishes it using MQTT under a topic that includes the device's unique ID. Similarly, the device uses its ID to publish messages for Ditto, creating a connection between the sender and receiver. Upon reception, the receiver decrypts and authenticates the MQTT message payload using the pre-shared symmetric key, ensuring the message's integrity and origin. In cases with multiple IoT devices, the receiver retrieves the appropriate shared key using the device ID, thereby ensuring secure and authenticated communication.

Encryption key management strategies for ASCON are crucial to ensure secure key distribution and storage, thereby enhancing the overall security posture of the proposed solution. ASCON employs symmetric encryption, where both encryption and decryption operations use the same secret key K.

3.8. Key Distribution Strategies in ASCON

3.8.1. Pre-Shared Key (PSK)

In scenarios where devices have prior knowledge of each other, a pre-shared key K is securely distributed among them. Mathematically, this can be represented as:

$$K = K_{\text{PSK}}$$

 doi:10.20944/preprints202407.0963.v1

Here, K_{PSK} denotes the pre-shared key.

3.8.2. Key Exchange Protocols

For dynamic key distribution, ASCON can integrate with key exchange protocols such as Diffie-Hellman (DH) or Elliptic Curve Diffie-Hellman (ECDH). These protocols allow devices to establish a shared secret over an insecure channel, which can then be used as the encryption key K.

3.8.3. Diffie-Hellman (DH)

Devices compute a shared secret $K = g^{ab} \mod p$, where g is a generator, a and b are private keys of the communicating parties, and p is a large prime number.

3.8.4. Elliptic Curve Diffie-Hellman (ECDH)

Similar to DH but operates over elliptic curve groups, providing stronger security with smaller key sizes.

3.9. Key Storage Practices

3.9.1. Secure Storage

Once established, keys K are stored securely using cryptographic key storage mechanisms. This ensures that unauthorized access to keys is prevented. Mathematically, secure storage can be denoted as:

$$\text{Secure Storage}(K)$$

This may involve hardware security modules (HSMs), secure enclaves, or trusted execution environments (TEEs) depending on the deployment environment.

3.9.2. Key Rotation

Periodic key rotation is essential to mitigate risks associated with long-term key exposure. New keys are generated and securely distributed, ensuring data confidentiality and integrity over time. Mathematically, key rotation can be represented as:

$$K_{\text{new}} \leftarrow \text{GenerateKey}()$$

In summary, ASCON's encryption key management strategies focus on secure distribution (PSK, key exchange protocols), robust storage practices, and periodic key rotation. These practices collectively contribute to maintaining a high level of security in IoT and Industry 4.0 deployments.

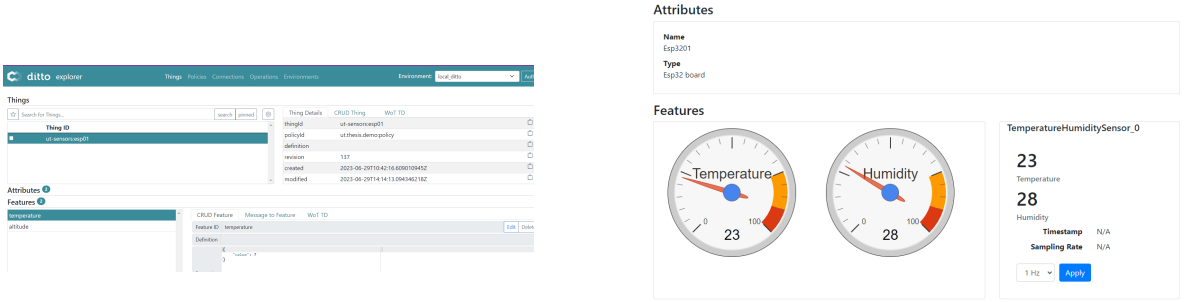
To simulate the life cycle of a Digital Twin, we have developed a web application that models the temperature and humidity features of an ESP32 sensor. The application utilizes JavaScript to retrieve these values through a stream of data using server-side events (SSE). Moreover, to send commands or messages to the server, we employ the HTTP POST API of Ditto. By subscribing to the command event associated with a specific topic, any device can consume the message and execute the corresponding action. This activity effectively simulates the communication between the digital twin and the actuators. Conversely, the communication from the IIoT device to the Digital Twin serves the purpose of collecting telemetry data from the operational environment.

Figure 5a illustrates the WebUI of Ditto, which is included by default in the code base. This web portal serves as a portal offering device, policy, and connection management functionalities. Additionally, Figure 5b provides an overview of an application layer built on the Digital Twin concept. The attributes displayed on the upper part of the image represent the name and type of the simulated device. The gauges visually represent the received device features, while the bottom right section presents textual information associated with the device.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 July 2024 doi:10.20944/preprints202407.0963.v1

This section discussed the relevant background information, the proposed solution's design requirement, and the implementation details. By implementing the proposed solution on both the Digital Twin (Ditto) and IIoT device (ESP32) we showed how to secure the communication channel using a resource-efficient authenticated encryption algorithm.

In the next section, we provided performance analysis results of our proposed solution regarding speed, memory usage, and power consumption. For the analysis, we measured the performance of our proposed solution based on three implementations without an encryption algorithm, with ASCON lightweight algorithm, and AES-GCM heavyweight algorithm based on AES.



(a) A Data Log Viewed from Ditto Platform.

(b) Web Application For Modeling Temperature and Humidity of ESP32

Figure 5. Ditto and Webapp Toward Simulating Digital Twin.

4. Performance Analysis

Performance analysis is a systematic study and measurement of various performance metrics to determine the effectiveness and efficiency of a system, process, or application. This includes factors such as processing speed (latency), memory utilization, and power consumption. In this section, we will focus on the quantitative assessment of performance metrics, specifically processing speed, memory requirements, and power consumption. The objective is to compare two distinct algorithms: one based on a lightweight algorithm and the other based on a traditional heavyweight encryption algorithm. This comparative analysis is conducted both at the functional level and within the broader application context of the proposed solution. Before starting the analysis, we present a case scenario that serves as the foundation for our measurements. Subsequently, this section reveals the outcomes of our performance measurements, shedding light on the quantitative results derived from the analysis. Last, we delve into a security analysis of the proposed communication scheme to validate the robustness and reliability of the proposed communication scheme in real-world scenarios.

Performance analysis involves a systematic study and measurement of various performance metrics to determine the effectiveness and efficiency of a system, process, or application. This includes factors such as processing speed (latency), memory utilization, and power consumption. In this section, we will focus on the quantitative assessment of performance metrics, specifically processing speed, memory requirements, and power consumption. The objective is to compare two distinct algorithms: one based on a lightweight algorithm and the other based on a traditional encryption algorithm. This comparative analysis is conducted both at the functional level and within the broader application context of the proposed solution. Before commencing the analysis, we present a case scenario that serves as the foundation for our measurements. Subsequently, this section reveals the outcomes of our performance measurements, shedding light on the quantitative results derived from the analysis.

4.1. Measurement Case Scenarios

To assess the performance of our proposed solution, we designed three case scenarios. For each scenario, we conducted analyses at both the algorithmic (function call) level and examined the broader application overhead.

 doi:10.20944/preprints202407.0963.v1

Case 1: AES-GCM: In this case, we assessed the performance of the AES-GCM encryption algorithm in terms of power consumption, memory usage, and processing speed. We analyze both its standalone performance as a function and its implementation within the application

Case 2: ASCON: In this case, we measured the performance of the ASCON encryption algorithm both at the functional level and its impact on the implementation of the proposed solution

Case 3: No encryption: This case scenario served as a baseline reference for the other two cases. In this case, we called a function that had a similar signature to the encryption function calls used for ASCON and AES-GCM. However, the underlying function did not perform any operations other than copying values from one memory location to another.

The implementation code for this case can be seen in the following code list.

```
int crypto_aead_encrypt(
      unsigned char *c,unsigned long long *clen,
      const unsigned char *m,unsigned long long mlen,
      const unsigned char *ad,unsigned long long adlen,
      const unsigned char *nsec,
      const unsigned char *npub,
      const unsigned char *k
      )
{
  *clen = mlen + CRYPTO_ABYTES;
  memcpy(c, m, mlen);
  memset(c + mlen, 0, CRYPTO_ABYTES);

  return 0;
}

int crypto_aead_decrypt(
  unsigned char *m, unsigned long long *mlen,
  unsigned char *nsec,
  const unsigned char *c, unsigned long long clen,
  const unsigned char *ad, unsigned long long adlen,
  const unsigned char *npub,
  const unsigned char *k
)
{
  unsigned long long len = *mlen = clen - CRYPTO_ABYTES;
  memcpy(m, c, len);

  return 0;
}
```

Listing 1. C Implementation Of No-Encryption - Base Line Reference of Measurement

4.2. Performance Measurement - Speed, Memory, and Power

The performance of a particular category of encryption algorithms is influenced by the algorithm's complexity, defined by factors such as the number of processing rounds and required CPU instructions. It's essential to recognize that there exists a trade-off between security and performance. More intricate algorithms generally offer higher security levels but demand greater computational resources and execution time. Therefore, designers and practitioners must meticulously assess the acceptable performance and the minimum required security level for a specific application before selecting encryption algorithms. This section aims to provide insights into the performance of our proposed solution, which relies on two algorithms: AES and ASCON, both utilizing a 128-bit key size. Our primary focus centers on measuring their processing speed, memory consumption, and power utilization. Additionally, we introduce a non-encrypted implementation to serve as a benchmark for comparative analysis.

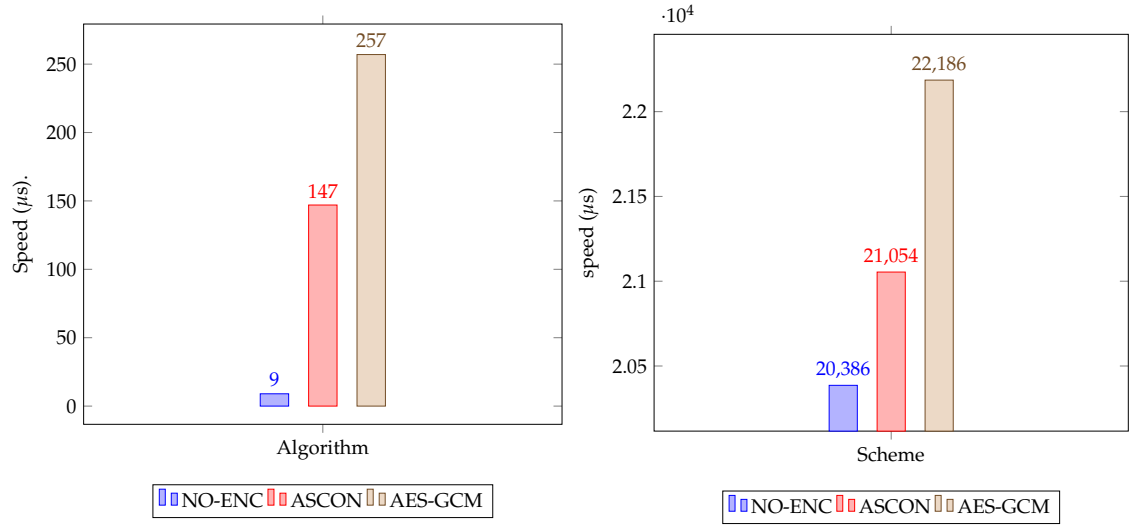
 doi:10.20944/preprints202407.0963.v1

4.2.1. Speed - Running Time

To measure the execution time of the algorithms, we used built-in function `esp_timer_get_time()` from the ESP-IDF framework. By capturing the start time before executing the function and the end time after the function completes, we can calculate the execution time by subtracting the start time from the end time. Our experimental setup involves three distinct case scenarios. Firstly, we measure the execution time when the application runs without any encryption algorithm. Then, we measure the execution time when the messages are encrypted using ASCON and AES algorithms. Note that we measure the performance of the proposed solution from two perspectives; 1) From the time required to run the encryption algorithm within the application program and 2) from the perspective of measuring the total time required to process the message and send it to the digital twin (Cloud). Both case scenarios are shown in figure Figure 6b and Figure 6a. To obtain accurate measurements, we allow our programs to run 1000 function calls sending data after encryption for each case scenario. We then calculate the average execution time using a Python script that processes the dump file we collected from the device. As it can be seen from Figure 6a ASCON's execution time is lower than AES-GCM. Similarly, Figure 6b shows that our proposed solution performs well when ASCON is used. These results suggest that ASCON is a promising candidate for use in our proposed solution. To measure the execution time of the algorithms, we utilized the built-in function `esp_timer_get_time()` from the ESP-IDF framework. By capturing the start time before executing the function and the end time after the function completes, we calculated the execution time by subtracting the start time from the end time. Our experimental setup comprised three distinct case scenarios. Firstly, we measured the execution time when the application ran without any encryption algorithm. Next, we measured the execution time when messages were encrypted using ASCON and AES algorithms. It's important to note that we evaluated the performance of the proposed solution from two perspectives:

1. The *time* required to run the encryption algorithm within the application program
2. The *total time* needed to process the message and send it to the digital twin (Cloud)

As Figure 6a illustrates, ASCON's execution time was lower than that of AES-GCM. Similarly, Figure 6b demonstrates that our proposed solution performed well when ASCON was used. These results strongly suggest that ASCON is a promising candidate for integration into our proposed solution.



(a) Speed From The Algorithm Execution Perspective (μs).

(b) Speed From The Scheme (Application) Perspective (μs).

Figure 6. Performance Of Three Case Scenarios From Algorithm Execution Speed and Application Running Time.

 doi:10.20944/preprints202407.0963.v1

Throughput and Cycle Byte Ratio

Throughput is a performance indicator of a system that measures the number of bytes processed per unit of time (typically seconds). It is a measure of how efficiently a system can process data. The more bytes that are processed per unit of time, the better the system performs. Throughput is typically measured in bytes per second (Bps) or kilobytes per second (Kbps). The Cycle Byte Ratio is another performance metric that measures the number of CPU cycles needed to process a single byte. Table 2 presents the cycle counts for different scenarios: 1) without employing any encryption algorithm, 2) utilizing the ASCON encryption algorithm, and 3) employing the AES-GCM encryption algorithm. Throughput serves as a crucial performance indicator for a system, quantifying the number of bytes processed within a specific time frame, typically measured in seconds. It gauges a system's efficiency in handling data; the higher the throughput, the more efficiently the system processes information. Additionally, the Cycle Byte Ratio, another vital performance metric, calculates the number of CPU cycles required to process a single byte of data. Table 2 displays cycle counts for the three different scenarios mentioned in 4.1:

Table 2. Cycle Count For 3 Cases: No-Encryption, ASCON, AES-GCM.

	No-Encryption	**ASCON**	**AES-GCM**
Cycle Count	2009	36811	49956
Byte Processed	32	32	32
CPU Freq MHz	240	240	240
Cycle per Byte	64.28	1094.68	31517.90
Time Elapsed μs	8.57	145.95	202.38
Throughput B/μs	3.73	0.219	0.158

$$\text{Throughput} = \frac{\text{Byte processed}}{\text{Total Time}} \tag{1}$$

$$\text{Cycle Byte} = \frac{\text{Cycle Count}}{\text{Byte processed}} \tag{2}$$

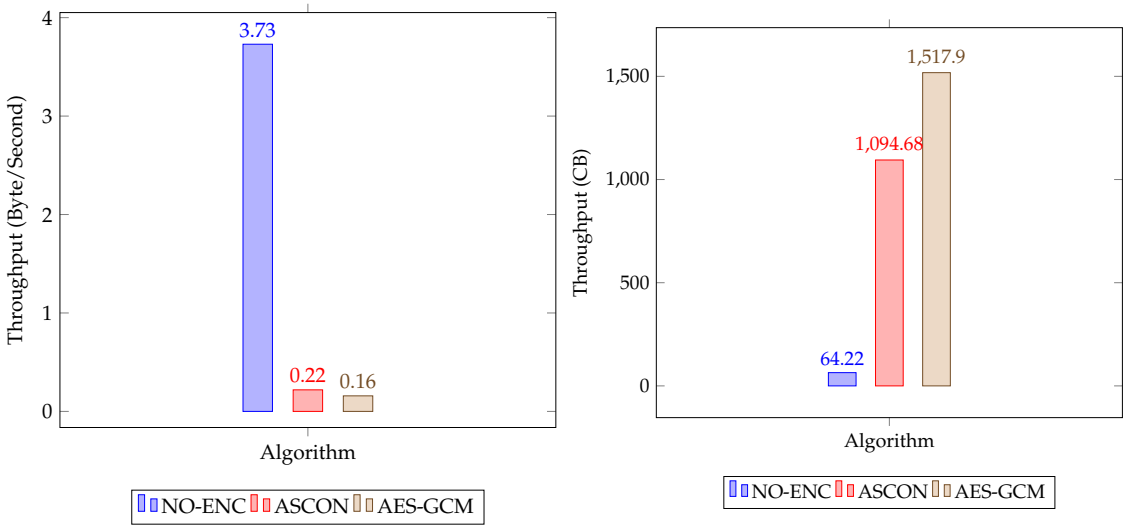
The total time taken by the CPU to perform an operation can be derived using cycle count and the CPU clock frequency using the following formula.

$$\text{Total Time} = \frac{\text{Cycle Count}}{\text{CPU frequency in Mhz/Khz}} \tag{3}$$

In this experiment, we use $xthal\_get\_ccount()$ from esp32/ck.h library to get the cycle count at a given time, and $getCpuFrequencyMhz()$ function to get the current set CPU frequency of the device from esp32-hal-cpu.c source file.

We calculated the throughput and cycle byte ratio of each algorithm in the proposed solution processing 16 bytes of message and 16 bytes of associated data, using equation 1 and 2. The results are shown in Figures 7a and 7b.

 doi:10.20944/preprints202407.0963.v1



(a) Throughput of the algorithms (Bytes/Seconds).

(b) Throughput In Cycle per Byte ratio.

Figure 7. Throughput and cycle per byte ratio of each algorithms.

4.3. Static and Dynamic Memory Footprint

Examining and evaluating the memory utilization of embedded programs is crucial, particularly in devices with limited memory capacity. In this section, we will delve into the analysis of both static and dynamic memory usage within our proposed solution. During our assessment, we explored three different implementation scenarios as mentioned in 4.1

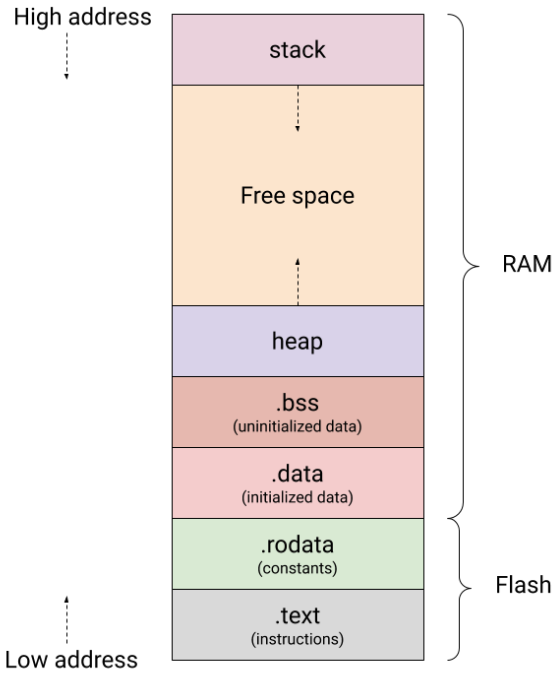


Figure 8. Memory Map of Embedded Programming.

A memory map of embedded programming is divided into sections. The flash part of the memory includes the .rodata and .text section contains the code resulting from compiling and building the source program. This section contains static codes and requires a fixed memory size. Whereas, the

 doi:10.20944/preprints202407.0963.v1

RAM section of the memory is responsible for containing a few sections for statically generated codes and the majority one for handling dynamic memory management such as stack and heap allocation.

The memory map in embedded programming is divided into sections. The flash memory part includes the *.rodata* which contains the data variables and *.text* section which contains the code resulting from compiling and building the source program [25]. This section contains static codes and requires a fixed memory size. In contrast, the RAM section of the memory is responsible for containing a few sections for statically generated codes and the majority of it is used for dynamic memory management, such as stack and heap allocation.

In our analysis of the dynamic memory usage, we focused only on the RAM section. However, when assessing the static memory usage (code size), we examined both the RAM and Flash sections. This approach was necessary as both sections contribute to the overall code size.

Static Memory Usage - Code Size

The static code size measurement was conducted to compare the code size requirements of three different implementation scenarios: No-Encryption, ASCON, and AES-GCM. The goal was to assess the impact of these implementations on resource-constrained devices in terms of memory usage.

Table 3 provides a summary of the code size measurements for each scenario. In the No-Encryption scenario, no additional code was required beyond the base program, resulting in minimal memory usage. Our implementation based on ASCON introduced a slight increase in code size. Approximately 1 KB of additional memory was needed in both RAM and Flash compared to the No-Encryption scenario. On the other hand, AES-GCM demonstrated slightly higher code size requirements. The implementation demanded approximately 8 KB more RAM and 5.6 KB more Flash memory compared to the No-Encryption case.

The static code size measurement was conducted to compare the code size requirements of three different implementation scenarios. The objective was to assess the impact of these implementations on resource-constrained devices in terms of memory usage. Table 3 provides a summary of the code size measurements for each scenario. In the No-Encryption scenario, no additional code was required beyond the base program, resulting in minimal memory usage. Our implementation based on ASCON introduced a slight increase in code size. Approximately 1 KB of additional memory was needed in both RAM and Flash compared to the No-Encryption scenario. On the other hand, AES-GCM demonstrated slightly higher code size requirements. The implementation demanded approximately 8 KB more RAM and 5.6 KB more Flash memory compared to the No-Encryption case

Table 3. Code Size (KB): No-Encryption, ASCON, AES-GCM.

	No-Encryption	**ASCON**	**AES-GCM**
Ram KB	59.3	59.3	68
Flash KB	766.5	767.7	772.1
Total KB	825.8	827	840.1
Cycle per Byte	64.28	1094.68	31517.90
Time Elapsed μs	8.57	145.95	202.38
Throughput B/μs	3.73	0.219	0.158

Table 4. Cycle Count For 3 Cases: No-Encryption, ASCON, AES-GCM.

	No-Encryption	**ASCON**	**AES-GCM**
Cycle Count	2009	36811	49956
Byte Processed	32	32	32
CPU Freq MHz	240	240	240
Cycle per Byte	64.28	1094.68	31517.90
Time Elapsed μs	8.57	145.95	202.38
Throughput B/μs	3.73	0.219	0.158

 doi:10.20944/preprints202407.0963.v1

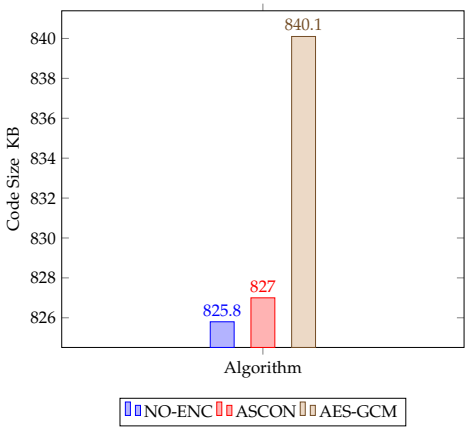


Figure 9. Static Code Size of the Scheme for Three Scenarios (No-Encryption, ASCON, AES-GCM).

Dynamic RAM Usage

Measuring the dynamic RAM usage of each algorithm is challenging compared to measuring static memory usage. This challenge arises from the need to monitor the dynamic memory allocations and deallocations on both the heap and stack, occurring throughout function calls and returns.

While an effective technique suggested by NIST involves overwriting memory with known values before program execution and subsequently tracking the memory cells affected by the program's operations, we have opted for alternative methods to approximate the dynamic memory usage of each algorithm described as follows.

In our device (ESP32), we discovered that the initially allocated memory for handling heap and stack allocation is 327,680 Bytes. With this information, we track the minimum heap size ever available, utilising the system call *esp_get_minimum_free_heap_size*. Our program was executed for 1000 iterations calling the encryption function for each implementation, allowing us to obtain a snapshot of the free memory available between the stack and heap 1000 times. By calculating the difference between the total allocated dynamic memory and the minimum heap size recorded, we were able to approximate the memory footprint of each implementation.

Furthermore, we employed the baseline implementation, which does not incorporate an encryption/decryption algorithm, as a benchmark to estimate the dynamic memory usage overhead added by ASCON and AES-GCM algorithms(see Figure 10b). This comparison with the baseline provided insight into the dynamic memory usage of each algorithm at the running time.

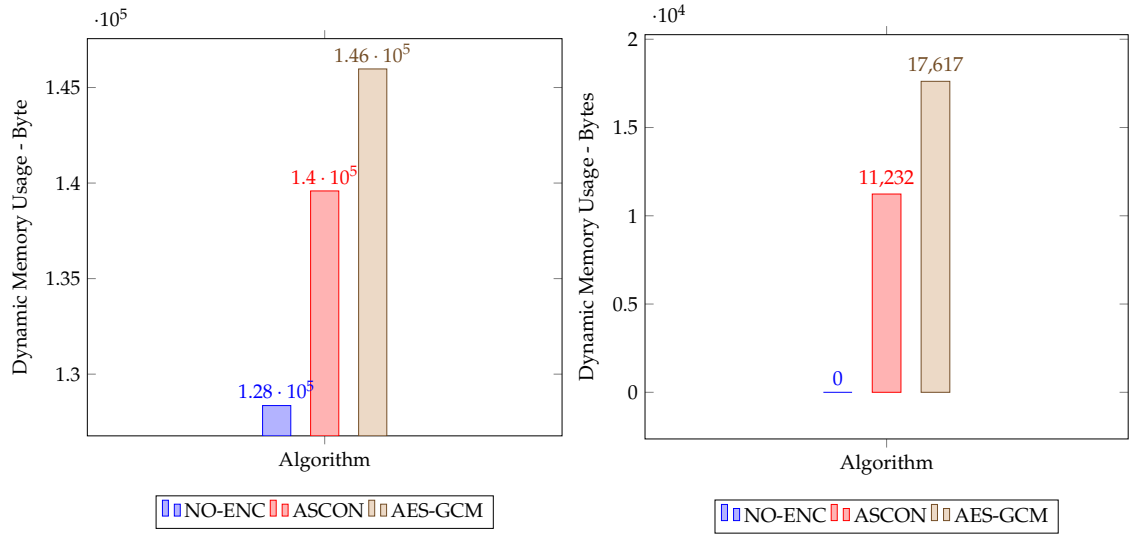
Measuring the dynamic RAM usage of each algorithm presents a challenge compared to measuring static memory usage. This difficulty stems from the need to monitor dynamic memory allocations and deallocations on both the heap and stack, occurring during function calls and returns. While an effective technique suggested by NIST involves overwriting memory with known values before program execution and subsequently tracking the memory cells affected by the program's operations, we have chosen alternative methods to approximate the dynamic memory usage of each algorithm, as described below.

In the ESP32 device, we found that the initially allocated memory for handling heap and stack allocation is 327,680 bytes. Armed with this information, we monitored the minimum heap size available using the system call *esp_get_minimum_free_heap_size*. The program was executed for 1000 iterations, calling the encryption function for each implementation. This allowed us to capture a snapshot of the free memory available between the stack and heap 1000 times. By calculating the difference between the total allocated dynamic memory and the minimum heap size recorded, we could approximate the memory footprint of each implementation.

Additionally, we utilized the baseline implementation, which lacks an encryption/decryption algorithm, as a benchmark to estimate the dynamic memory usage overhead introduced by the ASCON

 doi:10.20944/preprints202407.0963.v1

and AES-GCM algorithms (refer to Figure 10b). This comparison with the baseline offered valuable insights into the dynamic memory usage of each algorithm during runtime.



(a) Heap and Stack Memory Usage Each Implementation.

(b) The Additional Memory Overhead Introduced by the Algorithm.

Figure 10. Dynamic Memory Usage Comparison of Our Scheme Implementation and Algorithms.

4.4. Power Consumption Measurement

Power consumption is a critical factor for low-power Internet of Things (IoT) devices, as it can affect their battery life. There are a number of methods for measuring the power consumption of IoT devices, including using a USB power meter and an oscilloscope.

USB power meters are relatively inexpensive and easy to use, but they can be inaccurate, as they do not account for the power consumption of individual components in the device. Oscilloscopes are more accurate, as they can be used with a current probe to measure the current draw of the CPU, memory, and other components individually. However, oscilloscopes are more expensive and require more technical expertise to use.

In this section, we describe the methods and results of power measurements of our proposed solution implemented on the ESP32 device using Otti-arch tool.

Power consumption holds paramount importance, especially for resource-constrained Internet of Things (IoT) devices, as it directly impacts their battery life. Several techniques are available for measuring the power consumption of IoT devices, such as USB power meters and oscilloscopes.

USB power meters offer a cost-effective and user-friendly option; however, they may lack accuracy by not considering the power consumption of individual device components. On the other hand, while oscilloscopes offer high precision, they might lack the granularity needed to measure very low-power states accurately. In resource-constrained devices, power consumption often needs to be measured in microamps or even nanoamps, which can be challenging for standard oscilloscopes.

In this section, we outline the methods and outcomes of power measurements conducted on our proposed solution, implemented on the ESP32 device using the *Otii-arch tool* that calculates power and energy consumption and integrates with software output [26]

Method of Power Measurement

Measuring the power consumption of an algorithm or running application is a very challenging task due to the power noises stemming from other board components, such as Wi-Fi and Bluetooth. Getting a precise power consumption of running machine code on a chip requires a carefully set up lab environment, where the chip is isolated from the other components. This procedure involves

 doi:10.20944/preprints202407.0963.v1

connecting a fine-grained voltage source, typically an expensive oscilloscope, and capturing and measuring the power intake to the chip. Measuring the power consumption of an algorithm or a running application presents significant challenges, primarily due to power noise generated by other board components like Wi-Fi and Bluetooth. Obtaining an accurate measurement of the power consumption of machine code running on a chip necessitates a meticulously controlled laboratory environment, where the chip is isolated from other components. This process entails connecting a precise voltage source, often an expensive oscilloscope, and then capturing and measuring the power intake of the chip

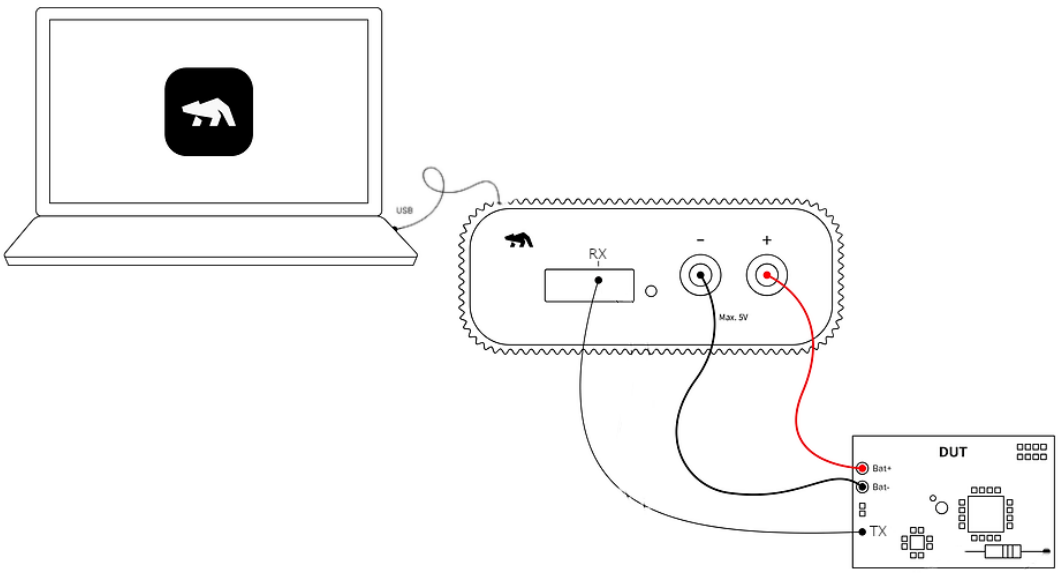


Figure 11. Power Measurement Setup Using Otii-arch.

However, in this project, we approximate the power consumption of our proposed solution using Otti arch from Quitech AB through techniques of setting up a UART recording. This technique allows us to zoom in on the power consumption of each function in the running application. The setup to measure the power consumption using UART recording is detailed as follows. Nevertheless, in this project, we estimate the power consumption of our proposed solution by employing Otii arch from Quitech AB. This is achieved through the setup of UART recording techniques [27], enabling us to focus specifically on the power consumption of each function within the running application. The procedure for measuring power consumption using UART recording is outlined as follows.

- The measuring device (Otti-arc) is connected to a power source of 3.7 Volt.
- The application of the proposed solution with UART logging code is built and deployed to the ESP32 device.
- A jumper is used to connect GPIO 17 TX from ESP32 to the expansion port RX on the Otti-arch.
- In the Otii application, the appropriate Digital voltage level for our device is selected and the UART channel with the correct Baud rate is configured.
- The impact of each called function on power consumption is analyzed by marking the UART message from the UART log window to get the corresponding power graph from the main window.

Table 5 shows the power analysis of three variants of our proposed solution in terms of current intake and energy consumption. The variant with no encryption exhibits lower average current intake and energy consumption compared to the other two variants. The AES-GCM variant consumes more

 doi:10.20944/preprints202407.0963.v1

current and energy, indicating its higher resource demands. On the other hand, ASCON maintains in between with average current consumption (116mA) and energy (530nWh).

Table 5. Power Consumption of LOLIN32 Lite ESP-32 Device With Three Variant Implementations of The Proposed Solution.

Algorithm	Min	Avg	Max	Energy
No encryption is applied	41.7mA	57.1mA	117mA	262nWh
ASCON	116mA	116mA	116mA	530nWh
AES-GCM	136mA	192mA	227mA	878nWh

The power analysis shows that ASCON requires less power while providing an equivalent level of security compared to AES-GCM. This is also evident from the cycle count of the algorithms, as shown in Table 2. AES-GCM has a higher cycle count than ASCON and No-Encryption, leading to higher current intake and energy consumption. On the other hand, the cycle count of ASCON falls in between No-Encryption and AES-GCM, resulting in average current and energy consumption levels higher than no-encryption but lower than AES-GCM.

Figure 12 shows the power consumption results of three implementations (no-encryption, ASCON, and AES-GCM). At around 9.56 seconds, the power consumption of the three implementations is approximately equal. However, as time progresses, the power consumption of AES-GCM increases significantly, while the power consumption of ASCON remains relatively constant at around 116 mA. This suggests that the AES-GCM algorithm requires more CPU cycles than ASCON, which results in a higher power spike.

Figure 12. Power Analysis Read of LOLIN32 Lite ESP32 Using Otii-arch Device.

5. Hypothesis Testing and Confidence Intervals

In this section we aimed to compare the mean processing speeds and memory requirements between AES-GCM and ASCON. By employing statistical tools such as **hypothesis testing** and **confidence intervals**, we ensured a robust evaluation of the performance metrics, providing insights into the effectiveness and efficiency of ASCON.

 doi:10.20944/preprints202407.0963.v1

5.1. Hypothesis Testing

5.1.1. Objective

- **Comparison**: The aim was to compare the mean processing speeds and memory requirements of two encryption algorithms, AES-GCM and ASCON, both using a 128-bit key size.
- **Null Hypothesis (H_0)**: There is no significant difference in the mean processing speeds and memory requirements between AES-GCM and ASCON.
- **Alternative Hypothesis (H_a)**: There is a significant difference in the mean processing speeds and memory requirements between AES-GCM and ASCON.

5.1.2. Methodology

- **Data Collection**: Performance metrics (processing speeds and memory requirements) were measured for both AES-GCM and ASCON in various operational scenarios.
- **Statistical Test**: Since the data likely follows a normal distribution (assuming large sample sizes or normality was checked), a two-sample t-test was used for hypothesis testing.
  - The two-sample t-test compares the means of two independent groups (AES-GCM and ASCON) to determine if they are statistically different from each other.

5.1.3. Interpretation

- **P-Value**: The computed p-value from the t-test indicates the probability of observing the data if the null hypothesis (H_0) is true. A lower p-value suggests stronger evidence against H_0.
- **Significance Level (α)**: Typically set at 0.05, the significance level determines the threshold for rejecting H_0. If the p-value is less than α, H_0 is rejected in favor of H_a, indicating a significant difference.

5.1.4. Outcome

- **Conclusion**: Based on the results of the hypothesis test, conclusions were drawn regarding whether there was a statistically significant difference in processing speeds and memory requirements between AES-GCM and ASCON.

5.2. Confidence Intervals

5.2.1. Objective

- **Assessment**: Confidence intervals were calculated to estimate the range of values within which the true mean processing speeds and memory requirements for AES-GCM and ASCON were likely to fall.
- **Reliability**: They provide a measure of the variability and reliability of the performance metrics, aiding in the interpretation of the quantitative results.

5.2.2. Methodology

- **Calculation**: For each performance metric (processing speeds and memory requirements), a confidence interval was computed.
  - A common choice is the 95% confidence level, which means there is a 95% probability that the true mean lies within the calculated interval.

5.2.3. Outcome

- **Conclusion**: The interpretation of the confidence intervals helps validate the consistency and reliability of the performance metrics observed for AES-GCM and ASCON, providing additional evidence to support the conclusions drawn from hypothesis testing.

 doi:10.20944/preprints202407.0963.v1

6. Security Analysis

The security analysis aims to evaluate the robustness of the proposed communication scheme for securing IoT device messages sent via the MQTT protocol. This analysis addresses the key security properties of the encryption scheme, including data confidentiality and message integrity. Although key management is also another critical aspect of security analysis, it is not considered within the scope of this analysis. The security analysis focuses on assessing the strength of the proposed communication scheme, safeguarding messages from IoT devices transmitted via the MQTT protocol. This evaluation delves into crucial security aspects of the encryption scheme, such as data confidentiality and message integrity. While key management holds significant importance in security analysis, it falls outside the scope of this examination.

6.0.1. Encryption Scheme Overview & Preliminary Assumption

Our proposed communication scheme leverages the ASCON algorithm which is computationally efficient and suitable for resource and power constraints for IoT devices with a 128-bit level of security. ASCON has been evaluated by a number of security experts and has been found to be secure [5].

As long as the security of ASCON is not broken, the proposed scheme with safe implementation is secure. The only information leaked to the adversary is the associated data, namely the thing ID (id). The thing ID is the additional data (or associated data) used in our *authenticated encryption with associated data* implementation to retrieve the right key upon receiving the message. The thing ID does not provide any advantage to an attacker even if it can be accessed in clear text. In addition, we assume that the private (symmetric) key is deployed before the communication starts through some sort of secure communication channel. Our proposed communication scheme relies on the ASCON algorithm, known for its computational efficiency and suitability for IoT devices operating under resource and power constraints, providing a 128-bit level of security. ASCON's security has been rigorously evaluated and confirmed by security experts [5]. As long as ASCON remains secure, our proposed scheme, when implemented safely, ensures security. The only information accessible to adversaries is the associated data, specifically the *thing ID*. This ID serves as additional data in our *authenticated encryption with associated data* implementation, facilitating the retrieval of the correct key upon message reception. Even if accessed in plain text, the thing ID does not grant any advantage to an attacker. Furthermore, we presume the private (symmetric) key is securely deployed before communication commences, accomplished through a secure communication channel.

6.0.2. Security Aspect of MQTT Protocol

It is important to note that the application protocol used, specifically the MQTT protocol, is not encapsulated or secured using TLS or any other security protocol. As a result, all the metadata associated with the MQTT protocol is openly available to the public including the topic names, the message payload sizes, and the timestamps of the messages.

MQTT protocol is not designed for secure communication. It is designed for lightweight, efficient communication between constrained devices. Hence, in the proposed scheme, only the message at the application level is secured through the AEAD algorithm and payload encryption. However, one can set up a gateway between the IIoT device and Digital Twin to provide security over the MQTT protocol.

Scheme Security Attacks

In our communication scheme, we rely on the security provided by ASCON as it is the encryption algorithm used to protect the payload. However, we also include the device ID in clear text alongside the encrypted payload for the reason explained in section 6.0.1 that could introduce weakness in our scheme. Following, we examine potential security attack scenarios that could exploit the availability of the device ID.

 doi:10.20944/preprints202407.0963.v1

Identity Spoofing Attack: An attacker might attempt to create a crafted encrypted payload and use a valid sensor's unique ID to impersonate a legitimate device. However, this message will be detected and discarded by the receiver (Digital Twin) as it will never result in a valid authentication tag. Therefore, As long as the private key of the device is secure and not accessible to potential attackers, the receiver can ensure the integrity and authenticity of the communication process.

Replay Attack: An attacker, as a man-in-the-middle, might intercept the communication with the intent of replaying the messages later to perform malicious intent. Yet, this attack is not feasible since the two communication parties are engaged with fresh nonce for each encryption. Hence, due to the nonce used in ASCON encryption, the proposed scheme is secure against replay attacks [5].

6.0.3. Security Attacks on ASCON

ASCON has been thoroughly evaluated by various security experts during the competition of CAESAR and no practical weaknesses have been found [5]. The algorithm's employed rounds (linearity and differentially) provide security against known attacks including linear, differential attacks and cube-like attacks [28]

Resistance Against Side-Channel Attack: The bit-sliced implementation of the S-boxes in ASCON provides defence against cache-time attacks [5]. This is because bit-sliced S-boxes are implemented in a way that does not require memory access or a lookup table. Instead, they are implemented using bit-level operations, which are difficult to attack.

In addition, the low algebraic degree of the S-boxes in ASCON allows an implementation to be resistant to extracting information from the power consumption or execution time of the algorithm [5]. Using masking [29] and shared-based [30] counter-measure techniques, ASCON can be implemented to provide resistance against power side-channel attacks.

6.1. Threat Model

1. **Adversary Capabilities**:

- *Eavesdropping*: The adversary $\mathcal{A}$ can intercept messages m between IoT devices and Digital Twins.
- *Active Attacks*: The adversary $\mathcal{A}$ can modify, inject, or replay messages m such that $m' \neq m$.
- *Cryptographic Attacks*: The adversary $\mathcal{A}$ can perform brute force, differential, linear, and algebraic attacks on the encryption algorithm.
- *Side-Channel Attacks*: The adversary $\mathcal{A}$ can exploit physical leakages such as timing t, power consumption P, and electromagnetic emissions E to gain information about the encryption keys k.

2. **Assets**:

- *Confidentiality*: Ensure that only authorized parties can read the data m. Formally, for an encryption algorithm $E_k(m)$, an adversary $\mathcal{A}$ should not be able to derive m from $E_k(m)$.
- *Integrity*: Ensure that data has not been altered by unauthorized parties. For a message m and its corresponding MAC $\text{MAC}_k(m)$, any modification $m' \neq m$ should result in verification failure.
- *Authenticity*: Verify the origin of the data. Formally, for a message m and its signature σ, $\text{Verify}(m, \sigma, pk) = \text{true}$ if and only if σ was generated by the private key corresponding to pk.
- *Availability*: Ensure that services remain available to authorized users.

3. **Security Goals**:

- *Data Encryption*: Protect data in transit between IoT devices and Digital Twins.
- *Authentication*: Verify the identity of communicating parties.
- *Message Integrity*: Detect and prevent tampering with messages.

6.2. Security Analysis of ASCON

ASCON is designed as a lightweight cryptographic algorithm, providing authenticated encryption with associated data (AEAD). Here's a detailed analysis of its security features:

1. **Resistance to Differential Cryptanalysis**:

 doi:10.20944/preprints202407.0963.v1

- *Assumption*: Differential probability $DP(f, \Delta x \rightarrow \Delta y)$ for an S-box f is low.
- *Proof*: For ASCON, the differential uniformity of the S-box is designed to be low:

$$\Pr[f(x) \oplus f(x \oplus \Delta x) = \Delta y] \leq 2^{-d}$$

where d is a small constant, ensuring that differential characteristics are hard to exploit.

2. **Resistance to Linear Cryptanalysis**:

- *Assumption*: Linear probability $LP(f, \alpha \rightarrow \beta)$ for an S-box f is low.
- *Proof*: For ASCON, the linear approximation of the S-box is designed to be sparse:

$$\Pr[\alpha \cdot x = \beta \cdot f(x)] \leq 2^{-l}$$

where l is a small constant, making linear cryptanalysis infeasible.

3. **Resistance to Algebraic Attacks**:

- *Assumption*: ASCON's internal state transformations P are highly non-linear.
- *Proof*: The algebraic complexity of ASCON's state transformations ensures that solving the system of equations derived from the cipher is computationally infeasible.

4. **Brute Force Attacks**:

- *Key Length*: ASCON uses a 128-bit key k.
- *Proof*: The number of possible keys is 2^{128}, making exhaustive search impractical:

$$\text{Time complexity} = O(2^{128})$$

5. **Side-Channel Attack Resistance**:

- *Power Analysis*:
  - *Assumption*: Power consumption $P(t)$ does not leak significant information about the key k.
  - *Proof*: ASCON operations can be implemented to ensure that power consumption is constant-time:

$$P(t_1) \approx P(t_2) \text{ for all } t_1, t_2$$

- *Timing Attacks*:
  - *Assumption*: Execution time t is independent of the key k.
  - *Proof*: Constant-time implementations ensure that:

$$t(f(x)) \approx \text{constant for all } x$$

6. **Message Authentication and Integrity**:

- *AEAD Mode*:
  - *Assumption*: ASCON provides both encryption and authentication.
  - *Proof*: Given a message m, a key k, and associated data a:

$$(c, \text{MAC}) = E_k(m, a)$$

where c is the ciphertext and MAC is the message authentication code. Verification ensures integrity:

$$\text{Verify}(c, \text{MAC}, k, a) = \text{true if and only if } (c, \text{MAC}) \text{ was generated by } k$$

 doi:10.20944/preprints202407.0963.v1

6.3. Comparative Analysis with TinyJAMBU

1. **Cryptographic Strength**:

- *ASCON*: Designed to provide robust security with resistance against various attack vectors.
  - *Proof*: ASCON's S-box and permutation functions are mathematically proven to resist differential and linear attacks.
- *TinyJAMBU*: Also designed for lightweight applications, focusing on minimalistic hardware implementation.
  - *Proof*: TinyJAMBU relies on a lightweight permutation that is secure against known attacks but may not be as versatile in software implementations.

2. **Efficiency**:

- *ASCON*: Optimized for both hardware and software, balancing security and performance.
  - *Proof*: ASCON's design ensures minimal overhead in resource-constrained environments.
- *TinyJAMBU*: Primarily optimized for hardware, potentially limiting its software performance.
  - *Proof*: TinyJAMBU's permutation is efficient in hardware but may require more cycles in software.

3. **Flexibility**:

- *ASCON*: Supports a wide range of applications with AEAD capabilities.
  - *Proof*: ASCON can be used for both encryption and authentication, making it versatile.
- *TinyJAMBU*: Focuses on specific use cases.
  - *Proof*: TinyJAMBU's design is tailored for minimal hardware footprint, which may limit broader applications.

4. **Resource Utilization**:

- *ASCON*: Demonstrates lower power consumption and memory usage compared to AES-GCM.
  - *Proof*: Experimental results show ASCON's efficiency in terms of power and memory s shown in Section 4.

ASCON's design, backed by attentive mathematical proofs, demonstrates its robustness and efficiency as a lightweight cryptographic algorithm. Its resistance to differential, linear, and algebraic attacks, combined with its low power consumption and memory usage, make it a suitable choice for securing IoT communications. The comparative analysis further highlights ASCON's advantages over other lightweight algorithms like TinyJAMBU, ensuring that industries can adopt secure and efficient encryption mechanisms without compromising performance. In addition to its cryptographic strength, ASCON is well-suited for securing communication channels between Digital Twins (DTs) and their associated physical devices in IoT ecosystems. By providing authenticated encryption with associated data (AEAD) capabilities, ASCON ensures that data integrity, confidentiality, and authenticity are maintained during transmission. This capability is crucial in IoT environments where resource-constrained devices, require efficient and robust security mechanisms to protect sensitive data and enable reliable communication with DTs. By offering both security guarantees and efficiency in resource utilization, ASCON supports the seamless integration of DT technologies in Industry 4.0 and beyond. Industries can confidently deploy ASCON to safeguard critical IoT communications while meeting rigid performance requirements.

 doi:10.20944/preprints202407.0963.v1

7. Discussion

Compared to the existing security mechanisms discussed in the literature review, our proposed solution stands out as it is based on lightweight authenticated encryption with associated data (AEAD), coupled with a payload encryption technique. Unlike all of the reviewed approaches that demand high computational resources, our lightweight authenticated encryption ensures data integrity and authenticity without imposing an excessive processing burden on resource-constrained IoT devices.

While the payload encryption technique employed in our solution avoids the necessity of SSL/TLS protocol handshake which is a highly resource-demanding process for resource-constrained devices, the AEAD techniques provide data integrity within the encrypted message without adding additional process steps for message authentication.

This combination of lightweight authenticated encryption and payload encryption, makes our solution particularly well-suited for real-world Industrial Internet of Things deployments where energy efficiency and low computational costs are critical considerations. In this way, our proposed approach offers an effective alternative to secure communication in Industry 4.0 based on the application of Digital Twin and Industrial Internet of Things technology, addressing the challenges posed by resource limitations while maintaining the required level of security.

7.1. Scalability of ASCON in Large-Scale IoT Applications

ASCON's scalability in large-scale IoT applications can be attributed to several key specifications and characteristics of its algorithm:

1. **Sponge Construction:** ASCON employs a sponge construction, well-suited for flexible data processing and variable input lengths. This design allows ASCON to handle varying amounts of data efficiently, making it adaptable to different message sizes encountered in IoT environments [31].
2. **Efficient Memory Utilization:** ASCON minimizes memory accesses during encryption and decryption processes, benefiting IoT devices with limited memory resources. By reducing memory operations, ASCON mitigates potential bottlenecks as the number of connected devices increases [32].
3. **Low Computational Overhead:** ASCON operates with low computational complexity, supporting high throughput across diverse IoT environments. Its streamlined operations ensure cryptographic tasks are performed swiftly and consistently [31].
4. **Adaptability and Configurability:** ASCON can be configured in various modes to optimize performance based on security needs, network constraints, and device capabilities. This flexibility allows ASCON to adapt to different operational contexts without compromising security or efficiency [32].
5. **Resistance to Side-Channel Attacks:** ASCON includes defenses against side-channel attacks, ensuring the security of cryptographic operations in IoT devices. This resilience enhances ASCON's suitability for large-scale IoT deployments [33].

These specifications highlight ASCON's capability to scale effectively in large-scale IoT applications, addressing challenges posed by increasing device connectivity and data volumes while ensuring efficient and secure cryptographic operations.

7.2. Potential Limitations of ASCON Algorithm

The ASCON algorithm, distinguished for its efficiency and security in resource-constrained environments, may encounter challenges in certain operational contexts:

1. **Performance in High-Dynamic Environments:** ASCON's performance may be influenced by the dynamic nature of IoT environments where data rates fluctuate rapidly. Sudden spikes in data volume or frequency of communication could impact ASCON's computational overhead and real-time responsiveness.

 doi:10.20944/preprints202407.0963.v1

2. **Resilience Against Advanced Attacks:** While ASCON exhibits robust resistance against common cryptographic attacks, its susceptibility to sophisticated attacks such as side-channel attacks or fault injections warrants careful consideration. These attacks exploit vulnerabilities in ASCON's implementation or its sensitivity to timing variations.
3. **Compatibility and Interoperability:** Integrating ASCON with existing systems and platforms may present compatibility challenges. Ensuring seamless interoperability across diverse IoT devices and communication protocols without compromising security or performance is crucial.
4. **Scalability in Large-Scale Deployments:** ASCON's scalability in deployments involving a large number of IoT devices transmitting data concurrently needs evaluation. Key management, synchronization overhead, and network congestion could affect ASCON's operational efficiency.
5. **Energy Efficiency on Low-Power Devices:** While ASCON is designed to be lightweight, its energy consumption on ultra-low-power devices (e.g., battery-powered sensors) with strict energy constraints may be a concern. Optimizing ASCON's cryptographic operations to minimize energy consumption without compromising security is challenging.
6. **Adaptability to Future Security Threats:** With evolving cybersecurity threats, ASCON's adaptability and resilience to future cryptographic attacks and vulnerabilities require continuous monitoring and updates. Algorithm agility and periodic security assessments are essential to maintain ASCON's effectiveness over time.

We have elaborated on these potential limitations to provide a balanced view of ASCON's applicability in real-world scenarios. By discussing these challenges, we aim to guide practitioners and researchers in making informed decisions regarding the adoption and implementation of ASCON in diverse IoT and cybersecurity applications.

7.3. Implication of Lightweight Solution

While traditional encryption methods can be highly secure, there are situations that require lightweight encryption solutions, especially in applications where real-time data processing and low latency are critical requirements. For instance, in the power automation system, the minimum tolerable time between fault detection and sending control to the power station should be as low as 4ms [34]. Hence, our proposed solution with less than 1ms for encryption and decryption can be used to meet the requirements of a such system.

Another implication of this work is related to power consumption. In a scenario where a number of sensors are deployed in a remote area with battery power, the low power consumption of the proposed solution can significantly increase the life span of the battery hence reducing frequent battery charging and replacement.

7.4. Limitations

While this study demonstrates the feasibility and practicality of employing lightweight encryption algorithms to secure the communication channel between resource-constrained devices and Digital Twin through the proposed communication scheme, it has also limitations like any other research effort.

Scope of Implementation: According to [35] there are more than 80 stream and block cipher algorithms candidates for lightweight cryptographic algorithms 2nd-round NIST competition. However, in this work, we only focus on the implementation of ASCON which is also a lightweight encryption algorithm. Investigating the implementation of various algorithm on various resource-constrained hardware micro-process cloud provide more comprehensive insight.

Performance Measurement: In this study, two performance metrics, memory, and power, are approximated. Due to time constraints and electronic laboratory resource limitations, a 100% accurate measurement is not provided. For the memory usage measurement, employing techniques like overwriting the memory with a known value and analysing the changed bit after running the program could provide more accurate than the techniques used in this work. Similarly, for the power

 doi:10.20944/preprints202407.0963.v1

consumption, an accurate measurement could have been achieved by running the algorithms on an isolated development board instead measuring the power consumption at the module level.

Despite these limitations, this research explores the practicability of lightweight encryption algorithms for enhancing security in communication between Digital Twin and Industrial Internet of Things. Having in mind these limitations, in the next section, we present future directions for further improvement.

7.5. Future Directions

Based on the insights gained from this work, the following future directions are proposed for further exploration:

Optimizing ASCON for ESP32 Microprocessor Chip: Different microprocessors have their own instruction architecture. Hence, future research work can be done on the optimization of lightweight encryption (ASCON) algorithms tailored for specific hardware devices, such as ESP32. This might involve identifying instructions that require fewer CPU cycles and replacing those instructions in the reference implementation that require more for the same operation.

Secure Remote Access for Resource-constrained Devices: In addition to securing the communication channel of resource-constrained devices, it is essential to develop secure remote access control solutions tailored specifically for those devices in Industry 4.0 settings. As a future work, authenticated encryption with associated data (AEAD) lightweight algorithms can be further explored to provide secure remote access to sensors and actuators deployed in industrial zones.

Exploring Trade-offs Between Security and Performance: Further exploration is needed to analyze the trade-offs between security and performance when implementing ASCON in resource-constrained IoT devices. This analysis will involve examining how different configurations impact security levels and operational efficiency.

User Study on Usability and User Experience: Conducting a user study or survey to gather feedback on the usability and user experience of implementing ASCON in real-world Industry 4.0 applications should be considered. Incorporating user perspectives will provide insights into the practical implications and usability challenges of ASCON implementations.

8. Conclusion

In conclusion, our research addresses a critical void in the realm of security mechanisms designed for Digital Twin applications within Industry 4.0. The existing literature lacks in-depth discussions and recommendations for security solutions that can effectively balance the demands for both speed and security within the constraints of Industry 4.0's Digital Twin ecosystem, especially considering the limitations of IoT devices. To bridge this gap, our study has introduced an innovative communication scheme utilizing the lightweight Authenticated Encryption with Associated Data (AEAD) algorithm, ASCON. This solution not only fulfills but surpasses crucial requirements for confidentiality, integrity, and authenticity while operating within the confines of resource limitations. A proof of concept involving the MQTT protocol validates the practicality of our approach, demonstrating its applicability in real-world scenarios. Moreover, our performance analysis, comparing ASCON to conventional AES (AES-GCM), unequivocally demonstrates the efficiency gains achieved in terms of speed, memory utilization, and power consumption, all while maintaining robust 128-bit security levels. This research makes a significant contribution by effectively addressing security challenges in Digital Twin applications for Industry 4.0, harmonizing with the unique constraints of sensor devices. Moving forward, future research should focus on the optimization and comprehensive validation of the proposed solution across various real-world scenarios, ensuring its enduring applicability and resilience in safeguarding critical infrastructures and beyond. Specifically, attention could be directed towards exploring the customization of lightweight encryption algorithms, such as ASCON, tailored for specific microprocessors like ESP32. This entails identifying CPU instructions requiring fewer cycles and replacing those in the reference implementation that demand more for the same operation.

 doi:10.20944/preprints202407.0963.v1

Additionally, there is a need to develop secure remote access control solutions specifically tailored for resource-constrained devices in Industry 4.0. A promising avenue for future work involves exploring the authenticated encryption with associated data (AEAD) family of lightweight algorithms, offering potential solutions for secure remote access to sensors and actuators deployed in industrial zones. By delving into these directions, researchers can further enhance the security landscape of Industry 4.0, ensuring the protection of critical infrastructures while adapting to the evolving demands of the digital era.

Abbreviations

The following abbreviations are used in this manuscript:

DT	Digital Twin
Industrial Internet of Things)	(Industrial) Internet of Things
AES	Advanced Encryption of Standard
AES-GCM	AES-Galois Counter Mode
RSA	Rivest–Shamir–Adleman
SLR	Systematic Literature Review
RQ	Research Question
NIST	National Institute of Standards and Technology
CASEAR	Competition for Authenticated Encryption: Security, Applicability, and Robustness
OT	Operational Technology
IT	Information Technology
PICOC	Population, Intervention, Comparison, Output and Context
CPS	Cyber-Physical Systems
ACM	Association for Computing Machinery
SCADA	Supervisory Control and Data Acquisition
AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
NFV	Network Functions Virtualization
SDN	Software Defined Network
ICS	Industrial Control System
SG	Smart Grid
SAML	Security Assertion Markup Language
MQTT	Message Queuing Telemetry Transport
RFID	Radio Frequency Identification
GPIO	General Purpose Input/Output
UART	Universal Asynchronous Receiver-Transmitter
ESP-IDF	Espressive IoT Development Framework
AEAD	Authenticated Encryption with Associated Data
SSE	Server Side Event
RAM	Random Access Memory

References

1. El-hajj, M.; Hahn, F. Security Aspects of Digital Twins in IoT. In Proceedings of the Proceedings of 9th International Conference on Information Systems Security and Privacy, ICISSP 2023, 2023, pp. 560–567. 9th International Conference on Information Systems Security and Privacy, ICISSP 2023, ICISSP 2023 ; Conference date: 21-02-2023 Through 24-03-2023, https://doi.org/10.5220/0011714500003405.
2. van der Wal, E.W.; El-Hajj, M. Securing Networks of IoT Devices With Digital Twins and Automated Adversary Emulation. In Proceedings of the 2022 26th International Computer Science and Engineering Conference (ICSEC), 2022, pp. 241–246. https://doi.org/10.1109/ICSEC56337.2022.10049355.

3. Elhajj, M.; Jradi, H.; Chamoun, M.; Fadlallah, A. Lasii: Lightweight authentication scheme using iota in iot platforms. In Proceedings of the 2022 20th Mediterranean Communication and Computer Networking Conference (MedComNet). IEEE, 2022, pp. 74–83.
4. van der Wal, E.W.; El-Hajj, M. Securing networks of iot devices with digital twins and automated adversary emulation. In Proceedings of the 2022 26th International Computer Science and Engineering Conference (ICSEC). IEEE, 2022, pp. 241–246.
5. Dobraunig, C.; Eichlseder, M.; Mendel, F.; Schläffer, M. Ascon v1.2: Lightweight Authenticated Encryption and Hashing. *Journal of Cryptology* **2021**, *34*, 33. https://doi.org/10.1007/s00145-021-09398-9.
6. El-Hajj, M.; Itäpelto, T.; Gebremariam, T. Systematic literature review: Digital twins' role in enhancing security for Industry 4.0 applications. *Security and Privacy*, p. e396.
7. Gehrmann, C.; Gunnarsson, M. A Digital Twin Based Industrial Automation and Control System Security Architecture. *IEEE Transactions on Industrial Informatics* **2020**, *16*, 669–680. https://doi.org/10.1109/TII.2019.2938885.
8. Xu, J.; He, C.; Luan, T.H. Efficient Authentication for Vehicular Digital Twin Communications. In Proceedings of the 2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall), 2021, pp. 1–5. https://doi.org/10.1109/VTC2021-Fall52928.2021.9625518.
9. Wu, J.; Guo, J.; Lv, Z. Deep Learning Driven Security in Digital Twins of Drone Network **2022**. pp. 1–6. https://doi.org/10.1109/ICC45855.2022.9838734.
10. Kumar, P.; Kumar, R.; Kumar, A.; Franklin, A.A.; Garg, S.; Singh, S. Blockchain and Deep Learning for Secure Communication in Digital Twin Empowered Industrial IoT Network. *IEEE Transactions on Network Science and Engineering* **2023**, *10*, 2802–2813. https://doi.org/10.1109/TNSE.2022.3191601.
11. Salim, M.M.; Comivi, A.K.; Nurbek, T.; Park, H.; Park, J.H. A Blockchain-Enabled Secure Digital Twin Framework for Early Botnet Detection in IIoT Environment. *Sensors* **2022**, *22*. https://doi.org/10.3390/s22166133.
12. Lv, Z.; Cheng, C.; Song, H. Digital Twins Based on Quantum Networking. *IEEE Network* **2022**, *36*, 88–93. https://doi.org/10.1109/MNET.001.2200131.
13. De Benedictis, A.; Esposito, C.; Somma, A. Toward the Adoption of Secure Cyber Digital Twins to Enhance Cyber-Physical Systems Security. In Proceedings of the Quality of Information and Communications Technology; Vallecillo, A.; Visser, J.; Pérez-Castillo, R., Eds., Cham, 2022; pp. 307–321.
14. Lai, C.; Wang, M.; Zheng, D. SPDT: Secure and Privacy-Preserving Scheme for Digital Twin-based Traffic Control **2022**. pp. 144–149. https://doi.org/10.1109/ICCC55456.2022.9880784.
15. Chen, H.; Jeremiah, S.R.; Lee, C.; Park, J.H. A Digital Twin-Based Heuristic Multi-Cooperation Scheduling Framework for Smart Manufacturing in IIoT Environment. *Applied Sciences* **2023**, *13*. https://doi.org/10.3390/app13031440.
16. Zheng, Q.; Wang, J.; Shen, Y.; Ding, P.; Cheriet, M. Blockchain Based Trustworthy Digital Twin in the Internet of Things. In Proceedings of the 2022 International Conference on Information Processing and Network Provisioning (ICIPNP), 2022, pp. 152–155. https://doi.org/10.1109/ICIPNP57450.2022.00040.
17. Danilczyk, W.; Sun, Y.L.; He, H. Blockchain Checksum for Establishing Secure Communications for Digital Twin Technology. In Proceedings of the 2021 North American Power Symposium (NAPS), 2021, pp. 1–6. https://doi.org/10.1109/NAPS52732.2021.9654790.
18. Liu, J.; Zhang, L.; Li, C.; Bai, J.; Lv, H.; Lv, Z. Blockchain-Based Secure Communication of Intelligent Transportation Digital Twins System. *IEEE Transactions on Intelligent Transportation Systems* **2022**, *23*, 22630–22640. https://doi.org/10.1109/TITS.2022.3183379.
19. Pervez, Z.; Khan, Z.; Ghafoor, A.; Soomro, K. SIGNED: Smart cIty diGital twiN vErifiable Data Framework. *IEEE Access* **2023**, *11*, 29430–29446. https://doi.org/10.1109/ACCESS.2023.3260621.
20. Feng, H.; Chen, D.; Lv, H. Sensible and secure IoT communication for digital twins, cyber twins, web twins. *Internet of Things and Cyber-Physical Systems* **2021**, *1*, 34–44. https://doi.org/https://doi.org/10.1016/j.iotcps.2021.12.003.
21. Eclipse Foundation. ditto...where IoT devices and their digital twins get together. https://eclipse.dev/ditto/index.html, 2023.
22. Maier, A.; Sharp, A.; Vagapov, Y. Comparative analysis and practical implementation of the ESP32 microcontroller module for the internet of things. In Proceedings of the 2017 Internet Technologies and Applications (ITA), 2017, pp. 143–148. https://doi.org/10.1109/ITECHA.2017.8101926.

 doi:10.20944/preprints202407.0963.v1

23. Eclipse Foundation. Eclipse Mosquitto. https://github.com/eclipse/mosquitto, 2023.
24. Fneish, Z.A.A.M.; El-Hajj, M.; Samrouth, K. Survey on iot multi-factor authentication protocols: A systematic literature review. In Proceedings of the 2023 11th International Symposium on Digital Forensics and Security (ISDFS). IEEE, 2023, pp. 1–7.
25. Randhawa, R.H.; Hameed, A.; Mian, A.N. Energy efficient cross-layer approach for object security of CoAP for IoT devices. *Ad Hoc Networks* **2019**, *92*, 101761. Special Issue on Security of IoT-enabled Infrastructures in Smart Cities, https://doi.org/https://doi.org/10.1016/j.adhoc.2018.09.006.
26. Qoitech. Otii Arc Pro. https://www.qoitech.com/otii-arc-pro/, 2023.
27. Pötsch, A.; Berger, A.; Springer, A. Efficient analysis of power consumption behaviour of embedded wireless IoT systems. In Proceedings of the 2017 IEEE International Instrumentation and Measurement Technology Conference (I2MTC), 2017, pp. 1–6. https://doi.org/10.1109/I2MTC.2017.7969658.
28. Dobraunig, C.; Eichlseder, M.; Mendel, F.; Schläffer, M. Cryptanalysis of Ascon. In Proceedings of the Topics in Cryptology — CT-RSA 2015; Nyberg, K., Ed., Cham, 2015; pp. 371–387.
29. Gross, H.; Mangard, S. Reconciling d+1 Masking in Hardware and Software. In Proceedings of the Cryptographic Hardware and Embedded Systems – CHES 2017; Fischer, W.; Homma, N., Eds., Cham, 2017; pp. 115–136.
30. Lambin, B.; Derbez, P.; Fouque, P.A. Linearly Equivalent S-Boxes and the Division Property. *Des. Codes Cryptography* **2020**, *88*, 2207–2231. https://doi.org/10.1007/s10623-020-00773-4.
31. Lefevre, C.; Mennink, B. Tight preimage resistance of the sponge construction. In Proceedings of the Annual International Cryptology Conference. Springer, 2022, pp. 185–204.
32. El-Hadedy, M.; Hua, R.; Yoshii, K.; Hwu, W.M. Optimizing ASCON Permutation in Multi-Clock Domains with Chisel: Resource Efficiency and Critical Path Reduction. In Proceedings of the 2024 IEEE 17th Dallas Circuits and Systems Conference (DCAS). IEEE, 2024, pp. 1–6.
33. Rezaeezade, A.; Basurto-Becerra, A.; Weissbart, L.; Perin, G. One for All, All for Ascon: Ensemble-based Deep Learning Side-channel Analysis. *Cryptology ePrint Archive* **2023**.
34. Rajkumar, V.S.; Tealane, M.; Ştefanov, A.; Presekal, A.; Palensky, P. Cyber attacks on power system automation and protection and impact analysis. In Proceedings of the 2020 IEEE PES Innovative Smart Grid Technologies Europe (ISGT-Europe). IEEE, 2020, pp. 247–254.
35. El-Hajj, M.; Fadlallah, A. Analysis of Lightweight Cryptographic Algorithms on IoT Hardware Platforms. In Proceedings of the 2022 32nd International Telecommunication Networks and Applications Conference (ITNAC), 2022, pp. 121–126. https://doi.org/10.1109/ITNAC55475.2022.9998413.