International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 10 | Oct - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

Secure IOT Node with Hardware-Assisted Authentication

1st M. Sravanthi, 2nd A. Sai Shivani, 3rd Ch. Shirisha, 4th Ch. Manusha, 5th B. Raju

1, 2, 3, 4, 5 Department of Electronics and Communication Engineering, CMR Technical Campus, Hyderabad, India

msravanthi.ece@cmrtc.ac.in, saishivani4306@gmail.com, shirishachukka05@gmail.com, manushacherukupally@gmail.com, rajupatelbanda143@gmail.com

---

Abstract - Internet-connected embedded devices are increasingly used to monitor and control applications such as buildings, healthcare equipment, manufacturing processes, and environmental systems. Their limited resources, unattended deployment, and wireless interfaces can make it difficult to apply security controls commonly used in larger computing platforms. In particular, a gateway that accepts sensor messages without sufficiently checking their origin may be exposed to device impersonation and data injection. This paper describes a low-cost IoT monitoring prototype that uses an ESP32-based sensor node, a gateway, and a separate test node to examine packet authentication. The sensor node obtains temperature readings from a DHT11 sensor and sends them through ESP-NOW. The design uses the ESP32's device-specific eFuse identifier as an identifier input and SHA-256 as a hashing primitive, while the gateway checks incoming packets before displaying accepted readings on an OLED and web dashboard. A buzzer is used to indicate rejected traffic. The prototype provides a practical framework for demonstrating node identification and gateway-side validation with commonly available hardware. The security properties depend on how the identifier and digest are combined: a public hardware identifier concatenated with an ordinary hash is not, by itself, a secret authenticator and does not provide reliable replay protection. For deployment, the design should use a keyed message authentication code, protected key provisioning, and a nonce or monotonic sequence number. These considerations are discussed to distinguish the prototype's observable behaviour from the stronger protections required in a production system.

Key Words: Internet of Things, ESP32, device authentication, SHA-256, HMAC, ESP-NOW, gateway security, replay protection.

1. INTRODUCTION

Connected sensors and controllers now support a wide range of monitoring tasks, from room-level environmental measurement to industrial telemetry. Their value comes from collecting information close to the physical process and forwarding it to a display, gateway, or remote service. However, the same wireless connectivity that makes these systems convenient also gives an adversary an opportunity to transmit fabricated readings, imitate a legitimate node, or repeatedly send previously captured packets. The consequences can range from misleading dashboard values to incorrect decisions by an automated controller [2].

Security is particularly challenging for small embedded devices. Their memory, energy budget, and processing capacity may be limited, and they are often installed in locations where physical access cannot be completely controlled. A design that depends only on a device name or an unprotected identifier can be copied by another node. Similarly, a checksum can detect accidental transmission errors but cannot establish that a message came from an authorized sender. A secure design therefore needs to consider device identity, message integrity, key storage, and freshness as related but distinct requirements [5].

Hardware features can contribute to device identification. For example, ESP32 devices expose chip-specific information associated with eFuse-backed fields. Such information can help a gateway distinguish registered devices during a prototype demonstration. Nevertheless, an identifier should not automatically be treated as a cryptographic secret: if an attacker can learn or reproduce the identifier, a hash computed from that value and public message fields may also be reproducible. Strong authentication requires a secret key or a suitable hardware-backed cryptographic mechanism, along with a protocol that prevents reuse of old messages.

This paper presents a compact IoT node and gateway arrangement for exploring these issues using commonly available components. A DHT11 sensor supplies temperature measurements to an ESP32 node. The node forms a packet and transmits it over ESP-NOW. At the receiver, the gateway applies its configured validation procedure before forwarding accepted readings to an OLED display and a web dashboard. A separate ESP32 is used to generate test traffic, and a buzzer signals packets that fail the configured check.

The main contributions of the prototype are:

- A low-cost, three-node test arrangement consisting of a sensor node, a gateway, and a traffic-injection node.
- An implementation workflow that combines device-specific identification, packet hashing, wireless transfer, and gateway-side validation.
- A visible response to rejected packets through an alert output and dashboard/display behaviour.
- A discussion of the limitations of identifier-based hashing and the additional measures required for robust message authentication and replay resistance.

The remainder of the paper reviews related approaches, explains the proposed arrangement and its implementation, discusses the observations that can be made from the prototype, and identifies improvements needed before deployment in security-sensitive environments.

2. RELATED WORK

IoT authentication research seeks to confirm that a communicating endpoint is permitted to participate while keeping processing, memory, and energy demands manageable. Existing approaches differ in the source of identity they trust, the way cryptographic material is stored, and the protection they provide against active attacks.

One research direction uses physical unclonable functions (PUFs). A PUF derives device-specific responses from physical variations in a circuit and can be used as part of an authentication protocol. PUF-based designs aim to reduce reliance on conventional stored identifiers, although their security depends

on the PUF construction, the challenge-response protocol, and resistance to modelling or physical attacks [1], [3], [8], [9]. A PUF is not equivalent to a normal chip identifier, and results for a PUF-based scheme should not be assumed to apply directly to an ESP32 eFuse field.

Another direction applies conventional cryptography. Hash functions such as SHA-256 are useful for creating fixed-length digests, but an unkeyed hash only binds the digest to the input data; it does not prove who generated that data. Message authentication codes, including HMAC built from a secure hash function, use a shared secret to verify both message integrity and possession of the key. Public-key techniques, including elliptic-curve cryptography, can reduce the need for a shared secret but introduce their own implementation and key-management requirements [5].

Hardware security components provide a further option. Secure elements, hardware security modules, trusted execution features, and platform security facilities can help protect keys or constrain sensitive operations. These mechanisms can improve resistance to key extraction, but their availability, cost, integration effort, and provisioning process vary between platforms [6]. For small prototypes, adding dedicated components may not be feasible, which motivates careful use of the security features already present in the microcontroller.

Gateway-based security is also relevant because a gateway can apply a consistent admission policy before data reaches an application or dashboard. Centralized validation simplifies monitoring and policy updates, but the gateway must still authenticate the source correctly and reject stale packets. A gateway that compares only a digest derived from publicly observable fields may reject accidental mismatches yet remain vulnerable to an attacker who can calculate the same digest.

The approach described here is intended as an educational, low-cost implementation rather than a claim that an eFuse identifier alone constitutes a cryptographic root of trust. Its value is the integration of sensing, packet transmission, receiver-side checks, and visible alerting in one test platform. To provide production-grade authentication, the design should replace any unkeyed identifier-based digest with HMAC-SHA-256 or another reviewed authentication protocol, protect the key, and include freshness information in each message.

3. PROPOSED SYSTEM AND METHODOLOGY

3.1 System Overview

The prototype contains three logical nodes: an authorized sensor node, a gateway node, and a test node used to transmit unauthorized or malformed packets. The sensor node connects to a DHT11 temperature sensor. The gateway receives ESP-NOW messages and controls the user-facing outputs. The OLED provides a local view of accepted measurements, while a web dashboard can be used for remote monitoring. A buzzer indicates that a packet has failed the configured validation check.

Figure 1 is intended to show the data path and the gateway's decision point. The diagram should be prepared so that it clearly distinguishes the sensor node from the gateway and attacker/test node.

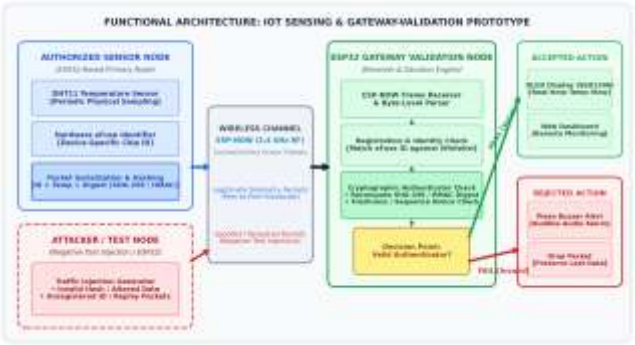


Fig -1: Functional architecture of the IoT sensing and gateway-validation prototype

3.2 Sensor Acquisition and Packet Formation

At each sampling interval, the sensor node reads the temperature reported by the DHT11. The reading is converted into a consistent representation before being inserted into a packet. A packet should include enough fields for the receiver to interpret the measurement, identify the sender, and determine whether the message is fresh. For example, a deployment-ready packet can contain a device identifier, temperature, sequence number, and authentication tag.

The prototype uses device-specific ESP32 information associated with eFuse-backed fields as an identifier. The identifier can be registered at the gateway during setup. It is important to distinguish identification from authentication: an identifier labels a device, whereas a cryptographic authenticator proves possession of a credential that an unauthorized sender should not have. The identifier must therefore not be treated as a secret key.

3.3 Digest Generation and Wireless Transfer

SHA-256 maps input data of arbitrary length to a 256-bit digest. In a basic prototype, the sensor data and device identifier may be combined and hashed to demonstrate repeatable digest generation. However, because an ordinary SHA-256 digest does not require a secret, this construction is not sufficient to resist an attacker who knows the input fields and algorithm.

For a stronger implementation, the sender and gateway should calculate an HMAC-SHA-256 tag over a canonical packet representation using a per-device secret key. The key must be provisioned securely and must not be transmitted in the packet. Both sides must encode fields in the same order and format to avoid verification errors. ESP-NOW provides a connectionless data transport mechanism; application-level authentication and replay checks still need to be designed explicitly.

3.4 Gateway Verification

When a packet arrives, the gateway first checks its format and the sender's registration status. It then verifies the authentication tag using the credential assigned to that device. If a sequence number or nonce is included, the gateway also checks that the value is new and acceptable under the protocol. A packet is accepted only when all required checks succeed.

For the demonstration workflow, a packet that passes the configured comparison is treated as accepted and its temperature is forwarded to the OLED and web dashboard. A packet that fails validation is discarded, and the gateway activates the buzzer or displays an intrusion warning. Rejected values should not overwrite the last accepted sensor reading. This separation ensures that an invalid packet cannot silently become the displayed measurement.

3.5 Test-Node Behaviour

A third ESP32 board acts as a test sender. It can transmit packets with incorrect digest values, altered measurements, unregistered identity fields, or repeated sequence numbers. The purpose is to check whether the gateway consistently rejects traffic that does not meet its admission rules. Testing with fabricated packets can demonstrate rejection of those particular cases, but it does not by itself prove resistance to every spoofing or replay strategy. Such claims require a defined threat model and tests against the final cryptographic protocol.

3.6 Algorithm

The receiver-side processing sequence is summarized below.

1) Initialize the sensor node, gateway, wireless parameters, display, and alert output.
2) Read a temperature sample from the DHT11 sensor.
3) Add the registered device identifier and the required packet fields.
4) Generate the packet authentication value using the configured method.
5) Send the packet over ESP-NOW.
6) At the gateway, check packet structure and confirm that the sender is registered.
7) Verify the authentication value using the gateway's trusted credential.
8) Check the nonce or sequence number when replay protection is enabled.
9) If all checks pass, update the OLED and web dashboard with the accepted reading.
10) If a check fails, discard the packet and issue an alert.
11) Repeat the process for subsequent samples and received packets.

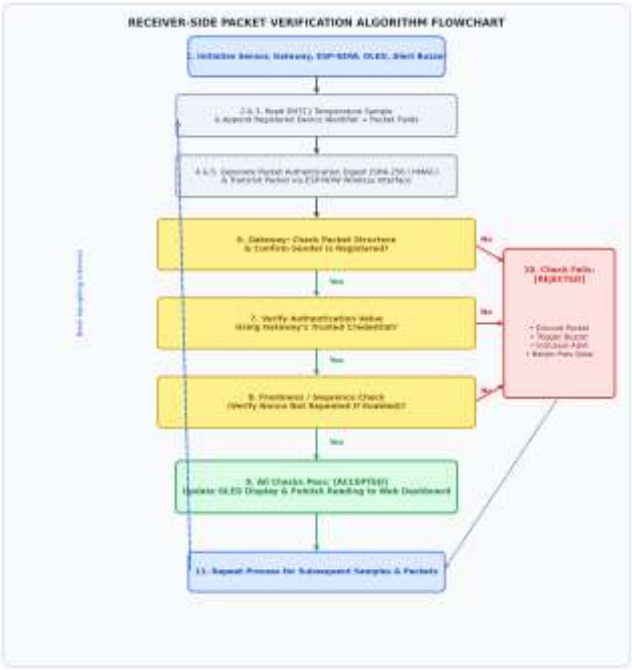


Fig -4: Receiver-side packet verification and alert algorithm flowchart

4. IMPLEMENTATION

The prototype is assembled from ESP32 development boards, a DHT11 sensor, an OLED module, and the wireless communication facility provided by ESP-NOW. The gateway also provides the buzzer alert and dashboard connection. The exact wiring, GPIO assignments, sampling interval, ESP-NOW configuration, and dashboard implementation should be documented in the final submission so that the setup can be reproduced.

On the sensor side, firmware initializes the DHT11 and wireless interface, obtains the device identifier used by the prototype, and periodically samples temperature. It then serializes the measurement and associated fields into a packet. A production implementation should use an explicitly defined byte-level format rather than relying on compiler-dependent structures, because padding and field representation can differ across builds.

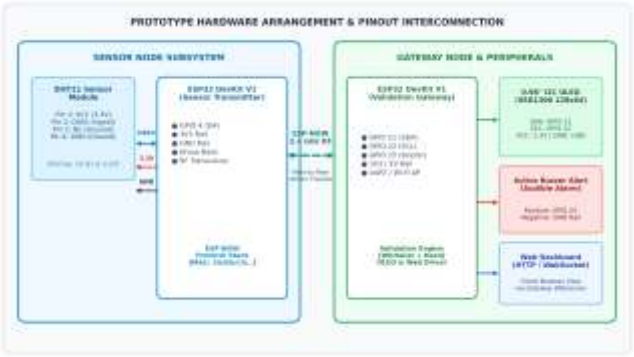


Fig -2: Prototype hardware arrangement and module interconnection schematic

The gateway firmware initializes the receiver, display, and alert output. For each incoming packet, it checks whether the data can be parsed safely and whether the sender is registered. The authentication check is performed before the measurement is passed to any output component. If verification succeeds, the gateway refreshes the displayed temperature and publishes the reading to the web dashboard. If it fails, the packet is discarded and an alert is raised.

The test node is used to generate negative test cases. Examples include an unregistered sender, a modified temperature field, a malformed packet, and a packet carrying an incorrect authentication value. If freshness protection is implemented, a previously accepted packet should also be resent to confirm that the gateway detects the repeated sequence number or nonce. The test report should record the number of packets sent, accepted, and rejected, along with the test conditions.

The use of ESP32 hardware identification can simplify device registration, but it should not be presented as proof that the sender possesses a secret. If the implementation currently uses only an unkeyed SHA-256 hash of the identifier and sensor data, the paper must describe the result as a basic consistency check, not as cryptographically strong authentication. To claim stronger protection, implement and test HMAC-SHA-256 with protected per-device keys and freshness validation.

5. RESULTS AND DISCUSSION

The prototype is designed to demonstrate the complete path from sensing to gateway-side packet validation and user notification. Under normal operation, the authorized node reads temperature from the DHT11 and sends a packet through ESP-NOW. When the packet passes the configured gateway checks, the receiver can display the accepted value on the OLED and publish it to the web dashboard. This verifies the integration of the sensor, wireless link, gateway firmware, and output devices.

Table -1: QUALITATIVE COMPARISON OF AUTHENTICATION APPROACHES

Aspect	Identifier or unkeyed hash	Keyed authentication design
Device identification	Uses a registered identifier	Uses a registered identity and credential
Message integrity	Detects changes only when digest is trusted	Authentication tag detects unauthorized modification
Impersonation resistance	Limited if fields are public	Depends on secrecy and protection of the key
Replay resistance	Not provided by a hash alone	Requires nonce/sequence validation
Implementation effort	Low	Moderate; requires provisioning and key handling
Deployment suitability	Demonstration or basic checking	More appropriate when properly implemented and tested

Negative tests use the separate ESP32 test node to transmit packets that do not satisfy the configured validation rules. A packet with an invalid digest, unknown device identifier, malformed fields, or a repeated sequence number should be discarded when the corresponding check is enabled. The buzzer and warning message provide a visible indication that the gateway has rejected traffic. These observations show the intended behaviour of the prototype for the test cases performed.

The current paper does not include measured latency, power consumption, packet-loss rate, false-acceptance rate, or a complete attack-test dataset. Such values should be measured and reported rather than estimated or invented. A useful evaluation should repeat each test over a defined number of packets and report the number of accepted and rejected messages. Authentication latency can be measured from packet reception to the gateway's decision; communication reliability can be assessed by comparing transmitted packets with received packets under stated distance and interference conditions.

A gateway timeout can also be used to identify a sensor that has stopped reporting. The timeout should be defined in the implementation and tested separately from authentication failure. A missing packet may be caused by radio interference, power loss, or a failed node, so it should be reported as a communication or availability event rather than automatically labelled as an attack.

Table I provides a qualitative comparison of broad design approaches. The entries are design-level observations, not experimentally measured scores. The actual security level depends on the cryptographic protocol, key management, freshness checks, implementation quality, and threat model. The results should be interpreted within these limits.

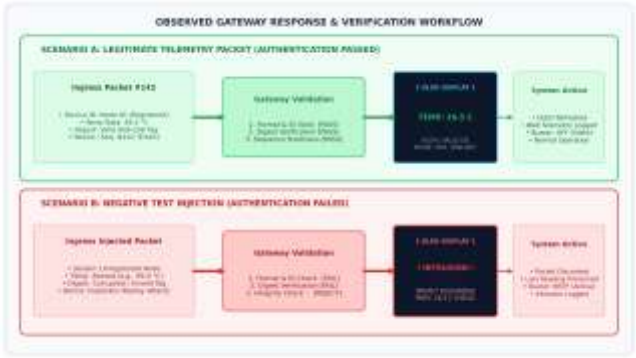


Fig -3: Observed gateway response and verification workflow for accepted vs. rejected packets

Rejecting the forged packets used in a demonstration is evidence that the implemented checks work for those packets; it is not sufficient to claim complete protection against spoofing, replay, or all forms of network attack. A stronger evaluation would document the packet format, authentication algorithm, key storage, replay policy, number of trials, and measured performance.

6. CONCLUSIONS

This paper described a low-cost ESP32-based IoT monitoring prototype that combines temperature sensing, ESP-NOW communication, gateway-side packet checks, and local or web-based presentation. A separate test node provides a way to exercise the receiver with unauthorized or malformed traffic. The arrangement is useful for demonstrating why a gateway should validate messages before updating an application-facing display.

A central design lesson is that hardware identification and cryptographic authentication serve different purposes. An eFuse-backed identifier can support device registration, but it should not be assumed to be confidential or unforgeable at the protocol level. Similarly, SHA-256 is a hash function rather than a digital-signature algorithm. A robust implementation should authenticate messages with HMAC-SHA-256 or another suitable cryptographic protocol, protect keys against disclosure, and use sequence numbers or nonces to prevent replay. Security claims should be based on tests of the implemented protocol rather than on the presence of a hardware identifier or hash function alone.

With these improvements and a documented performance evaluation, the prototype can provide a foundation for studying practical authentication in resource-constrained IoT systems.

7. FUTURE SCOPE

Several extensions could make the system more suitable for deployment. First, a per-device HMAC key can be provisioned securely and stored using platform-supported security features. Second, sequence numbers or nonces can be incorporated into the authenticated packet to identify replayed messages. Third, secure boot, signed firmware updates, and debug-interface restrictions can reduce the risk of firmware replacement or unauthorized access to the device.

The evaluation can be extended by measuring authentication time, energy use, packet delivery ratio, and behaviour under interference. Additional test cases should cover replay attempts, altered fields, unregistered senders, malformed packets, and key rotation. For larger deployments, the gateway may support controlled device enrolment, credential revocation, event logging, and secure transfer of verified readings to a cloud service. These enhancements could be explored in smart buildings, environmental monitoring, agriculture, and industrial telemetry,

subject to the security and reliability requirements of each application.

ACKNOWLEDGEMENT

The authors express their gratitude to the Department of Electronics and Communication Engineering at CMR Technical Campus, Hyderabad, India, for providing the laboratory facilities and academic guidance that supported this research work.

REFERENCES

[1] Y. Zhang et al., "Design and Implementation of Cost-Effective End-to-End Authentication Protocol for PUF-Enabled IoT Devices," IEEE Access, 2025. [Verify volume, issue, pages, DOI, and exact title.]

[2] S. Kim et al., "Low-Cost No-Touch IoT Security," IEEE Internet of Things Journal, 2021. [Verify exact title and publication details.]

[3] M. Singh et al., "PUF-Based Authentication in IoT Against Strong Physical Adversaries Using Zero-Knowledge Proofs," IEEE Access, 2023. [Verify exact title and publication details.]

[4] A. Kumar et al., "Two-Factor Authentication Using Hardware and Biometrics for IoT Devices," IEEE Access, 2023. [Verify exact title and publication details before citing.]

[5] R. Sharma et al., "Integration of Hardware Security Modules and Permissioned Blockchain in Industrial IoT Networks," IEEE Internet of Things Journal, 2022. [Verify exact title and publication details.]

[6] J. Lee et al., "Formalising an Architectural Model of a Trustworthy Edge IoT Security Gateway," IEEE, 2021. [Add venue, pages, and DOI after verification.]

[7] "PUF-Based Secure Lightweight Authentication and Key Exchange Protocol for IoT (PLAKE)," 2022. [Verify authors, venue, and DOI before use.]

[8] A. Verma et al., "A PUF-Based Modeling-Attack Resilient Authentication Protocol for IoT Devices," 2022. [Verify venue and publication details.]

[9] K. Patel et al., "On the Security of a PUF-Based Authentication and Key Exchange Protocol for IoT Devices," 2023. [Verify venue and publication details.]

[10] X. Jin et al., "A Lightweight Authentication Scheme for Power IoT Based on PUF and Chebyshev Chaotic Map," IEEE Access, 2024. [Verify exact bibliographic record.]

[11] Y. Cho et al., "A Secure and Anonymous User Authentication Scheme for IoT-Enabled Smart Home Environments Using PUF," IEEE Access, 2022. [Verify exact bibliographic record.]

[12] Y.-H. Chuang et al., "PUF Based Authenticated Key Exchange Protocol for IoT without Verifiers and Explicit CRPs," IEEE Access, 2021. [Verify exact bibliographic record.]

[13] S. Lee et al., "Joint Heterogeneous PUF-Based Security-Enhanced IoT Authentication," IEEE Internet of Things Journal, 2023. [Verify exact bibliographic record.]

[14] C. Gupta et al., "PEAK: PUF-based End-to-End Authentication and Key-Exchange for IoT," IEEE Embedded Systems Letters, 2026. [Verify publication status and full details.]

BIOGRAPHIES (Optional not mandatory)

Author biographies and photographs (optional per IJSREM template structure):

[1'st Author Photo]	M. Sravanthi is currently with the Department of Electronics and Communication Engineering, CMR Technical Campus, Hyderabad, India.
[2'nd Author Photo]	A. Sai Shivani is currently an undergraduate student in the Department of Electronics and Communication Engineering, CMR Technical Campus, Hyderabad, India.
[3'rd Author Photo]	Ch. Shirisha is currently an undergraduate student in the Department of Electronics and Communication Engineering, CMR Technical Campus, Hyderabad, India.
[4'th Author Photo]	Ch. Manusha is currently an undergraduate student in the Department of Electronics and Communication Engineering, CMR Technical Campus, Hyderabad, India.
[5'th Author Photo]	B. Raju is currently an undergraduate student in the Department of Electronics and Communication Engineering, CMR Technical Campus, Hyderabad, India.