Secure Raspberry Pi-Based IoT System for Sensor Data Acquisition and Integrity Assurance

Shine Vattuparampil Sam
Dept. of ECE
SUNY Binghamton University
Binghamton, USA
svattuparamp@Binghamton.edu

***Abstract*—This paper presents the design and implementation of a secure IoT sensor node leveraging a Raspberry Pi for real-time environmental and motion monitoring. The system integrates multiple sensors, including a PIR motion detector, an AHT20 temperature and humidity sensor, and a USB camera to capture short video segments upon motion detection. Communication is handled using the MQTT protocol secured with TLS 1.3 and mutual certificate-based authentication. Sensor data is transmitted in JSON format to a central laptop broker and logged in real time. To ensure the integrity of motion-triggered video files, the system computes a SHA-256 hash of each video and includes it in the MQTT payload. This enables post-transfer verification of video authenticity. The architecture is designed to be scalable to a multi-node network with distributed sensing and supports future integration with neuromorphic computing models for anomaly detection. The proposed system emphasizes secure communication, data integrity, and modular expansion for smart grid and IoT security applications.**

***Index Terms*—Internet of Things (IoT), MQTT, TLS, Raspberry Pi, SHA-256, Sensor Integration, Real-Time Monitoring, Video Integrity, Data Security, Smart Grid Security**

I. Introduction

The integration of advanced sensing and communication technologies has rapidly transformed monitoring and security strategies in diverse fields such as smart grids, industrial IoT, environmental monitoring, and security surveillance. The ability to continuously monitor environmental parameters and reliably detect events in real-time can significantly enhance operational efficiency, improve decision-making, and strengthen security protocols. Modern IoT-based monitoring systems leverage various sensors for capturing data related to temperature, humidity, motion, and visual events. However, ensuring data integrity and secure communication across IoT devices remains a primary concern, given the increased cyber threats targeting interconnected devices and networks. Thus, robust, secure, and reliable solutions are imperative to meet these evolving security requirements and challenges.

This paper describes the implementation of a secure multi-modal IoT sensor node utilizing a Raspberry Pi platform. The system integrates diverse sensing technologies, including a Passive Infrared (PIR) sensor for motion detection, an AHT20 sensor to monitor temperature and humidity via the I2C communication interface, and a USB-connected camera to record short video clips in response to detected motion events. At the core of this system is the Message Queuing Telemetry Transport (MQTT) protocol, a lightweight publish-subscribe messaging framework ideal for resource-constrained IoT environments. To further enhance system security, MQTT communications are secured with Transport Layer Security (TLS) version 1.3, providing robust end-to-end encryption. The implementation also employs mutual certificate-based authentication, significantly strengthening access control and preventing unauthorized interactions within the system.

An essential aspect of this implementation is maintaining data integrity, especially for video recordings triggered by critical events such as motion detection. To achieve this, the system employs the Secure Hash Algorithm 256 (SHA-256) to generate a unique digital fingerprint of each recorded video file. The SHA-256 hash is transmitted along with the sensor data to the central MQTT broker, enabling the receiving end to verify video authenticity upon receipt. This approach protects the video files from tampering or corruption during transmission or storage, ensuring reliable evidence collection and analysis. Furthermore, the system is designed with scalability in mind, enabling future expansion into distributed multi-node configurations. Such expansions facilitate seamless coordination among multiple Raspberry Pi sensor nodes, potentially incorporating advanced analytics methods, such as neuromorphic computing and machine learning algorithms, for anomaly detection and predictive analytics. Overall, this integrated, secure, and scalable IoT sensor platform provides a comprehensive solution for real-time environmental and security monitoring applications.

The main contributions of this paper are as follows: (1) Integration of a secure, multi-modal sensor node using a Raspberry Pi that combines environmental and motion sensing technologies with real-time video capture. (2) Implementation of secure MQTT-based communication over TLS 1.3, featuring mutual certificate authentication for enhanced security. (3) Development of a robust data integrity validation mechanism using SHA-256 hashing to ensure the authenticity of recorded video files. (4) Demonstration of a scalable system architecture capable of expanding into a distributed, multi-node network suitable for integration with advanced anomaly detection techniques, such as neuromorphic computing models, for smart grid and IoT security applications.

II. Reservoir Architecture Design

The core of this IoT monitoring system is centered around a Raspberry Pi, chosen for its versatility, ease of integration, and robust community support. The Raspberry Pi serves as a central computing and communication platform, interfacing seamlessly with various sensors and actuators through multiple built-in communication protocols and general-purpose input/output (GPIO) pins. Specifically, the sensor integration includes three primary sensors: a Passive Infrared (PIR) motion sensor, an AHT20 temperature and humidity sensor, and a USB-connected camera. Each sensor fulfills a distinct function and utilizes specific interfaces of the Raspberry Pi. The PIR sensor connects via GPIO to monitor motion events continuously and efficiently. The AHT20 sensor, interfaced through the Raspberry Pi's Inter-Integrated Circuit (I2C) communication protocol, captures precise environmental data such as temperature and humidity. Lastly, the USB webcam enables visual data capture, recording short video segments whenever motion is detected by the PIR sensor.

To support reliable, real-time data transmission and centralized management, the hardware components are interconnected within a secure local area network (LAN). This network configuration comprises a dedicated Wi-Fi router and an Ethernet switch, providing stable connectivity and secure communications between the Raspberry Pi sensor node and the central laptop server. The laptop hosts an MQTT broker (Mosquitto) secured using TLS 1.3, ensuring that all data exchanges between the sensor node and the central server are encrypted and authenticated. The MQTT protocol was specifically selected due to its lightweight nature, minimal network overhead, and suitability for environments with limited computational resources. Additionally, the system is further strengthened by implementing MAC address filtering at the router, firewall rules to restrict unnecessary network traffic, and Fail2Ban to prevent unauthorized access attempts. These combined hardware and network security measures significantly reduce potential vulnerabilities and protect the system against common network-level attacks.

A significant component of the system design involves the secure handling of motion-triggered video recordings. After recording a 10-second video segment upon detecting motion, the system calculates a SHA-256 cryptographic hash of the recorded video file. This SHA-256 hash serves as a digital fingerprint, uniquely representing the video data. Both the video file and its SHA-256 hash are securely transmitted to the laptop via SCP (Secure Copy Protocol) and MQTT, respectively. Upon receiving the video, the laptop recalculates the hash to ensure integrity, verifying that the video has not been tampered with or corrupted during transmission or storage. This combination of hardware integration, secure network architecture, and robust data integrity measures ensures that the overall design not only efficiently meets real-time monitoring requirements but also adheres strictly to best practices in cybersecurity and IoT security. Furthermore, the modular and scalable design facilitates easy expansion, allowing additional sensor nodes or computing resources to be integrated into the system seamlessly for advanced analysis such as anomaly detection through neuromorphic computing methods.

A. Software Flow

The software developed for this IoT sensor node is primarily written in Python, leveraging a modular and structured approach to handle various sensor interactions, video recording, data hashing, and secure communication. The Python script first initializes necessary GPIO pins, I2C interface, and USB camera resources. The GPIO interface is employed to interact with the PIR motion sensor, continually monitoring the digital input signal to detect movement. Simultaneously, the I2C interface is initialized to communicate with the AHT20 sensor, facilitating the periodic acquisition of temperature and humidity data. For visual monitoring, the USB camera is configured using the FFmpeg command-line tool, enabling efficient real-time video recording triggered by detected motion events.

```
GPIO.setmode(GPIO.BCM)
GPIO.setup(PIR_PIN, GPIO.IN, pull_up_down=GPIO.
    PUD_DOWN)

i2c = busio.I2C(board.SCL, board.SDA)
aht20 = adafruit_ahtx0.AHTx0(i2c)
```

Listing 1. GPIO and Sensor Initialization

Upon detecting a high-level signal from the PIR sensor (indicating motion), the script initiates a sequence of events aimed at capturing and securing relevant data. First, a timestamped, ten-second video segment is recorded and saved locally on the Raspberry Pi. Immediately following the video capture, the script computes a cryptographic SHA-256 hash of the recorded video file. This hash acts as a unique digital fingerprint for the video, ensuring that any alteration or corruption during transfer or storage can be reliably detected.

```
if motion:
    timestamp = datetime.now().strftime("%Y%m%d_%H%M
    %S")
    video_filename = f"motion_{timestamp}.mp4"
    subprocess.run([
        "ffmpeg", "-f", "v4l2", "-input_format", "
    mjpeg",
        "-video_size", "1024x768", "-i", "/dev/
    video0",
        "-t", "10", "-vcodec", "libx264", "-preset",
     "ultrafast",
        "-y", video_path
    ])

video_hash = compute_sha256(video_path)
```

Listing 2. Video Capture and SHA-256 Hashing

The calculated hash is then incorporated into a structured JSON payload, along with sensor data such as temperature, humidity, motion status, and the associated timestamp. For secure real-time communication, the Python script utilizes the Paho MQTT client library, connecting to the MQTT broker hosted on the central laptop. The MQTT connection is encrypted and authenticated using Transport Layer Security (TLS) version 1.3, relying on certificate-based mutual authentication to verify both the server and client identities. After

successful connection and authentication, the prepared JSON payload containing all collected sensor data and the SHA-256 hash is published securely to a predefined MQTT topic.

```
payload = {
    "timestamp": datetime.now().isoformat(),
    "temperature_C": temperature,
    "humidity_percent": humidity,
    "motion_detected": motion,
    "video_filename": video_filename,
    "video_hash": video_hash
}

client.publish(topic, json.dumps(payload))
```

Listing 3. MQTT Payload Construction and Publishing

Additionally, the script employs a sleep-time mechanism, introducing deliberate delays after each data publishing cycle. Specifically, the system waits five seconds after detecting motion and sending a video event to prevent message flooding and reduce network and CPU load. In cases where no motion is detected, a shorter two-second delay is used, ensuring an optimized balance between responsiveness and resource utilization.

```
time.sleep(5 if motion else 2)
```

Listing 4. Adaptive Sleep Time Based on Motion Event

This structured and secure software flow not only ensures data confidentiality, authenticity, and integrity but also facilitates seamless scalability and modularity for future multi-node expansions and more advanced computational analyses.

III. Result and Future Work

The implemented system was successfully tested in a local environment where a Raspberry Pi sensor node monitored motion, temperature, and humidity, and responded to motion events by capturing and storing video. The sensor data and SHA-256 hash of the recorded video were published securely to a Mosquitto MQTT broker running on a laptop over TLS 1.3. The laptop subscriber script received all JSON payloads and logged the readings in real-time into a CSV file.

While the current implementation successfully integrates multiple sensors and secure communication into a single-node system, the architecture is intentionally designed with scalability in mind. A key direction for future development is the expansion into a multi-node configuration, where multiple Raspberry Pis can operate as distributed sensing units. Each node may handle specific tasks—for example, one dedicated to motion detection, another to temperature and humidity monitoring, and another for capturing video. These nodes can communicate securely with one another using MQTT, coordinated through a central broker, enabling event-driven responses (e.g., motion detected by one Pi triggers video capture on another). This distributed model would support larger deployments across physical spaces, increasing both coverage and flexibility.

Additionally, future iterations of the system can include real-time data analytics using neuromorphic computing models or TinyML techniques. Data collected from all nodes can be forwarded to a centralized server capable of running anomaly detection algorithms to identify abnormal behavior patterns in environmental data or sensor activity. The inclusion of digital signatures for payload authentication and data encryption at rest for video files would further strengthen security. The final vision is a highly secure, intelligent, and scalable monitoring system that not only senses and records events but also reasons about them—paving the way for smart grid integration, building automation, and critical infrastructure security.

References

[1] A. Banks and R. Gupta, "MQTT Version 3.1.1," OASIS Standard, 2014. [Online]. Available: https://docs.oasis-open.org/mqtt/mqtt/v3.1.1/

[2] Eclipse Foundation, "Mosquitto MQTT Broker Documentation." [Online]. Available: https://mosquitto.org/documentation/

[3] Python Software Foundation, "paho-mqtt: MQTT client library for Python." [Online]. Available: https://pypi.org/project/paho-mqtt/

[4] Adafruit Industries, "AHT20 Temperature & Humidity Sensor Python Library." [Online]. Available: https://github.com/adafruit/Adafruit_CircuitPython_AHTx0

[5] Raspberry Pi Foundation, "Raspberry Pi GPIO Pinout and Usage." [Online]. Available: https://www.raspberrypi.com/documentation/computers/os.html

[6] OpenSSL Project, "OpenSSL: Cryptography and SSL/TLS Toolkit." [Online]. Available: https://www.openssl.org/

[7] National Institute of Standards and Technology (NIST), "FIPS PUB 180-4: Secure Hash Standard (SHS)," Mar. 2012. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf

[8] K. Zhao and L. Ge, "A Survey on the Internet of Things Security," in *Proc. 9th Int. Conf. Computational Intelligence and Security*, 2013, pp. 663–667.

[9] H. Ning and H. Liu, "Cyber-Physical-Social Based Security Architecture for Future Internet of Things," *IEEE Internet of Things Journal*, vol. 1, no. 4, pp. 401–407, Aug. 2014.