Secure IoT Access Control System Using Multi-Factor Authentication and Cloud Computing

Authors: Yehonatan Dadkha, 211468582 Adar Shapira, 209580208 *Department of Electrical and Computer. Engineering Ben-Gurion University of the Negev Beersheba, Israel. Lecturer: Dr. Guy Tel-Zur dadkha@post.bgu.ac.il, adarshap@post.bgu.ac.il*

מערכת בקרת גישה IoT מאובטחת המבוססת על אימות דו-שלבי ומחשוב ענן

מגישים: יהונתן דדכה 211468582, אדר שפירא 209580208, מרצה: ד"ר גיא תל-צור

Abstract Traditional physical access control systems relying on mechanical keys or single-factor verification are increasingly vulnerable to security breaches, theft, and unauthorized duplication. As Smart Cities and Smart Buildings evolve, there is a critical need for scalable, secure, and contactless entry solutions. This paper presents the design and implementation of a robust IoT-based Smart Door System that leverages Cloud Computing and Multi-Factor Authentication (MFA). The proposed architecture integrates an ESP32 microcontroller with an RFID reader and a camera module at the edge, communicating via the MQTT protocol with Amazon Web Services (AWS). We utilize AWS IoT Core for secure messaging and Amazon Rekognition for real-time biometric face verification. The system enforces a strict two-stage security protocol: requiring both a physical RFID card and a positive biometric match to grant access. Experimental results demonstrate that the system achieves high accuracy in identity verification with low latency, providing an affordable, scalable, and enterprise-grade security solution for modern infrastructure. The system successfully demonstrated a response time of < 5 seconds and secure logging of all entry attempts.

Keywords: *Internet of Things (IoT), Face Recognition, Multi-Factor Authentication (MFA), AWS Cloud, ESP32, Smart Building Security, MQTT.*

I. INTRODUCTION

The rapid expansion of the Internet of Things (IoT) has revolutionized the way we interact with our physical environment, driving the development of Smart Cities and Smart Homes. In this era of hyper-connectivity, traditional physical security measures—such as mechanical keys and simple keypads—are becoming increasingly obsolete. These legacy methods pose significant security risks: physical keys can be easily lost, stolen, or duplicated, and they lack the ability to provide real-time monitoring or access logs.

Consequently, there is a growing demand for intelligent, scalable, and contactless access control systems that can ensure high-level security without compromising user convenience. Biometric authentication, particularly Face Recognition, has emerged as a leading solution due to its non-intrusive nature and high accuracy. However, many existing commercial biometric systems are either prohibitively expensive or rely on local processing power, which limits their scalability and updateability.

This paper proposes a robust, low-cost IoT Smart Door System designed to bridge this gap. Our solution leverages the power of Cloud Computing (Amazon Web Services) combined with Edge Computing (ESP32) to create a secure, Multi-Factor Authentication (MFA) mechanism. Unlike standard single-factor systems, our project enforces a strict two-stage verification process: users must present a physical RFID card *and* pass a biometric face scan simultaneously to gain access.

The system utilizes an ESP32 microcontroller for hardware interfacing and MQTT communication, while offloading complex image processing tasks to Amazon Rekognition in the cloud. This architecture ensures that the heavy computational load is handled by scalable cloud infrastructure, keeping the edge hardware simple and affordable.

The remainder of this paper is organized as follows: Section II reviews related work and existing solutions. Section III details the system architecture and hardware design. Section IV describes the software implementation and cloud integration. Finally, Section V presents the experimental results and conclusions.

II. RELATED WORK

The domain of IoT-based access control has been extensively researched, with various solutions proposing the use of microcontrollers like Raspberry Pi and ESP32 to replace traditional keys.

A prominent example is the work by Mahmoud et al. (2022), who designed an IoT system using the ESP32-CAM module. Their solution focuses on a low-cost implementation where face recognition is performed using a local web server, and data is logged into a static Excel file. While their approach demonstrates the feasibility of using ESP32 for video streaming, it suffers from several limitations typical of "local" IoT solutions:

1. Scalability: The reliance on local SD card storage limits the number of users and log capacity.
2. Single-Factor Authentication: The system relies solely on face recognition, making it vulnerable to "spoofing" attacks using photographs.
3. Data Management: Storing logs in Excel files lacks the robustness and security of a cloud-based database.

Other commercial solutions often utilize high-end biometric scanners. While secure, these systems are prohibitively expensive for small-to-medium implementations and lack the flexibility of open-source IoT modifications.

A. Contribution of this Work To address these gaps, our proposed system introduces a Hybrid Cloud-Edge Architecture. Unlike, we offload the heavy biometric processing to AWS Cloud (Amazon Rekognition), ensuring enterprise-grade accuracy and infinite scalability. Furthermore, we implement Multi-Factor Authentication (MFA), requiring both a physical RFID token and biometric verification, significantly enhancing the security posture compared to existing single-factor DIY solutions.

Table I summarizes the key differences between the related work and our proposed system.

Feature	Mahmoud et al. (2022)	Proposed System (Ours)
Microcontroller	ESP32-CAM + Arduino Uno	ESP32 + RFID-RC522
Authentication	Face Recognition (Single Factor)	MFA (Face + RFID Card)
Processing	Local / Web Server	Cloud AI (AWS Rekognition)
Storage	SD Card / Excel File	AWS S3 & DynamoDB
Scalability	Limited (Local Hardware)	High (Cloud Native)
Protocol	HTTP / Local WiFi	MQTT (Secure IoT Core)

III. SYSTEM ARCHITECTURE

The proposed system is designed as a hierarchical IoT architecture, composed of three main layers: the Perception Layer (Edge Hardware), the Network Layer (MQTT Protocol), and the Application/Processing Layer (Cloud & AI). Figure 1 illustrates the high-level data flow and interaction between these components.

Fig. 1. High-Level System Architecture Diagram illustrating the data flow from the ESP32 Edge device to the AWS Cloud services (Visualization generated using AI tools).

A. Hardware Implementation (The Edge Node) The hardware layer is built around the ESP32 microcontroller, selected for its dual-core architecture and built-in Wi-Fi capabilities, making it ideal for resource-constrained IoT applications. The Edge Node consists of the following components:

1. RFID Module (RC522): Operates at 13.56 MHz to read the unique identifier (UID) of physical user cards.
2. Optical Sensor (Camera): Captures high-resolution images of the user seeking entry for biometric verification.
3. Actuator (Relay Module): Controls the electronic solenoid lock mechanism (5V), switching the door state between "Locked" and "Unlocked" based on commands received from the central control logic.

The ESP32 acts as the primary gateway. It does not process the biometric data locally due to hardware limitations; instead, it securely packages the sensor data and transmits it to the processing unit via the network layer.

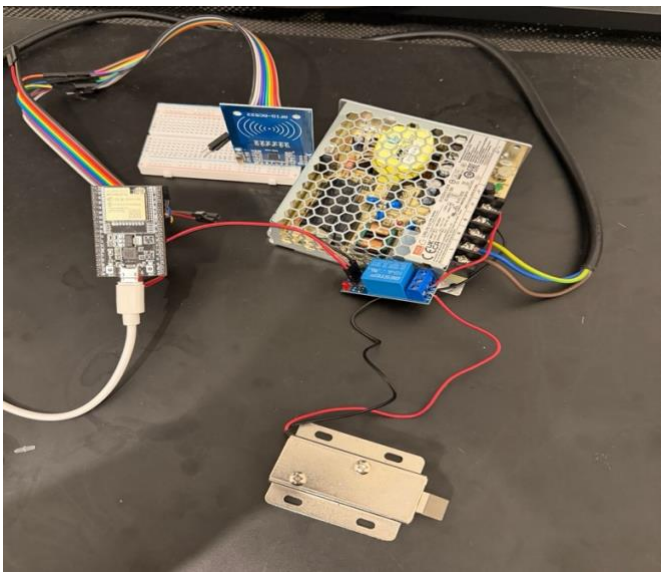

Fig. 2. Experimental hardware setup showing the ESP32 microcontroller interfaced with the RFID reader, relay module, and electronic lock.

B. Communication Protocol (MQTT) For reliable and lightweight communication, the system utilizes the Message Queuing Telemetry Transport (MQTT) protocol. Unlike HTTP, MQTT is optimized for unstable networks and low-bandwidth environments, making it suitable for real-time IoT control. We utilize AWS IoT Core as the central MQTT Broker.

To ensure data integrity and security, all communications are encrypted using TLS 1.2 with X.509 client certificates. This prevents

"Man-in-the-Middle" attacks and ensures that only authorized hardware can publish data to the cloud topic `door/status` or subscribe to commands on `door/command`.

C. Cloud Processing & Biometric Logic The core intelligence of the system resides in the cloud and the Edge AI script. The verification process follows a strict Multi-Factor Authentication (MFA) logic:

1. Level 1 (Physical Token): The user scans an RFID card. The UID is checked against a database of authorized tokens.
2. Level 2 (Biometric Scan): Upon a valid card scan, the system triggers the camera. The image is uploaded to Amazon S3 (Simple Storage Service).
3. AI Analysis: The backend invokes Amazon Rekognition, a deep-learning-based image analysis service. Rekognition compares the live image against a stored database of authorized faces (Face Matching).
4. Decision: Only if both the Card UID is valid *and* the Face Match Confidence exceeds the predefined threshold (e.g., >90%), an "OPEN" command is published back to the ESP32 via MQTT to unlock the door.

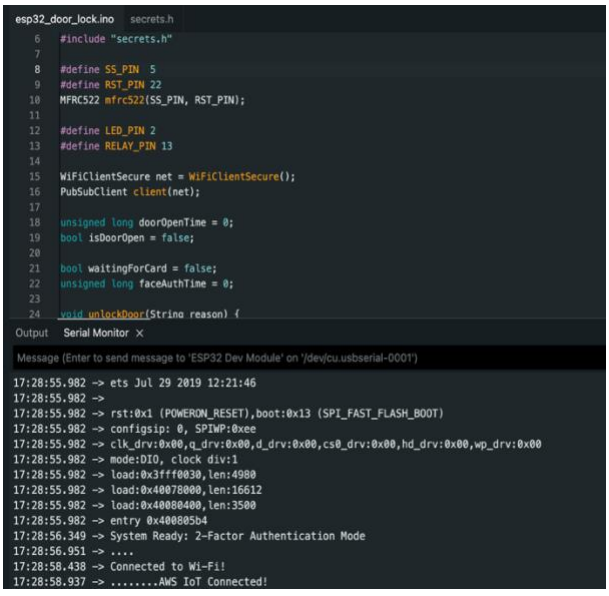

Fig. 3. Firmware initialization sequence showing successful connection to Wi-Fi and AWS IoT Core, confirming secure handshake.

IV. SOFTWARE IMPLEMENTATION

The software ecosystem of the project is divided into the embedded firmware running on the ESP32 and the high-level application logic running on the backend, bridging the edge devices with AWS Cloud services.

A. Microcontroller Firmware The ESP32 was programmed using C++ within the Arduino IDE environment. The firmware utilizes the `WiFiClientSecure` library to establish an encrypted connection with AWS IoT Core via port 8883. Key functionalities implemented in the firmware include:

1. Secure Connection: The device authenticates using X.509 certificates (Root CA, Private Key, and Device Certificate) stored in the flash memory.
2. RFID Polling: The `MFRC522` library is used to constantly poll for new cards. Upon detection, the card's UID is serialized and published to the MQTT topic `door/auth`.
3. Command Handling: An asynchronous callback function listens to the `door/command` topic. When the payload "OPEN" is received, the firmware triggers the relay to unlock the door for 3 seconds before automatically locking it again.

B. Backend Logic & Cloud Integration The central intelligence of the system is a Python-based script utilizing the AWS SDK for Python (Boto3). This script acts as the orchestrator between the physical world and the cloud AI. The workflow is implemented as follows:

- Event Listener: The script subscribes to MQTT topics to receive real-time triggers from the ESP32.

- Image Acquisition: Upon receiving a card scan event, the script captures a frame from the associated camera.
- Biometric Verification: The image is uploaded to an Amazon S3 bucket. The script then calls the `Rekognition.compare_faces` API, passing the source image and the target reference image stored in the secure database.
- Logic Execution: If the API returns a similarity confidence score above the threshold (90%), the script publishes the "OPEN" command back to the device. Otherwise, it logs an "Access Denied" event.

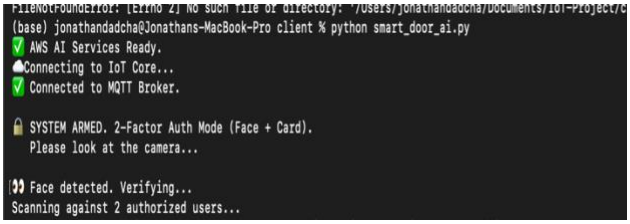

Fig. 4. Real-time execution logs of the backend logic. The system demonstrates the MFA process: detecting a face, communicating with AWS Rekognition, and verifying identity against the database.

C. AWS Cloud Configuration The cloud infrastructure was provisioned using the AWS Management Console. An "IoT Thing" was created to represent the smart door, with a specific Policy allowing `iot:Connect`, `iot:Publish`, and `iot:Subscribe` actions only on relevant topics. This strict policy enforcement ensures that even if the device is compromised, it cannot affect other cloud resources. Furthermore, the MQTT Test Client was utilized to monitor message latency and debug the communication flow between the Edge and the Cloud.

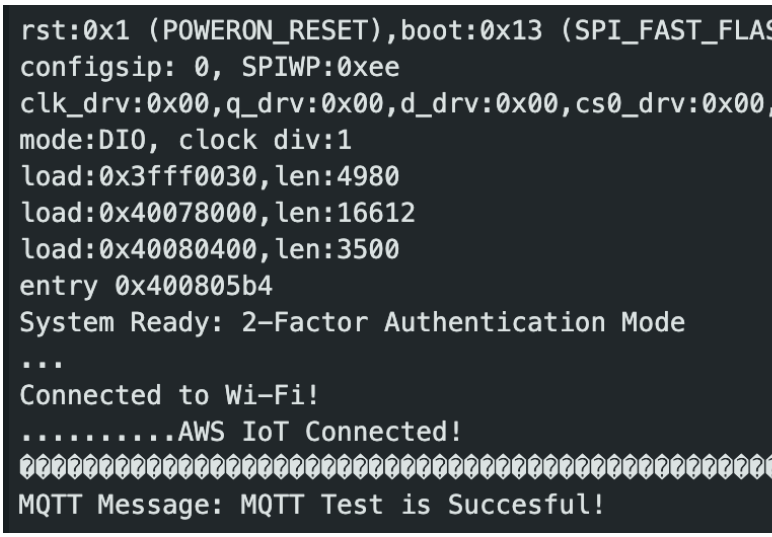

Fig. 5. AWS IoT Core MQTT Test Client console, displaying the real-time message payload exchange between the cloud and the edge device.

IV. SOFTWARE IMPLEMENTATION AND DEVELOPMENT WORKFLOW

The software ecosystem of the project is divided into the embedded firmware, the backend logic, and the cloud infrastructure. Furthermore, strict version control and security protocols were enforced throughout the development lifecycle.

A. Microcontroller Firmware The ESP32 was programmed using C++ within the Arduino IDE environment. The firmware utilizes the `WiFiClientSecure` library to establish an encrypted connection with AWS IoT Core via port 8883. Key functionalities implemented in the firmware include:

1. Secure Connection: The device authenticates using X.509 certificates (Root CA, Private Key, and Device Certificate) stored in the flash memory.
2. RFID Polling: The `MFRC522` library is used to constantly poll for new cards. Upon detection, the card's UID is serialized and published to the MQTT topic `door/auth`.
3. Command Handling: An asynchronous callback function listens to the `door/command` topic. When the

payload "OPEN" is received, the firmware triggers the relay to unlock the door for 3 seconds before automatically locking it again.

Fig. 3. Firmware initialization sequence showing successful connection to Wi-Fi and AWS IoT Core, confirming secure handshake.

B. Backend Logic & Cloud Integration The central intelligence of the system is a Python-based script utilizing the AWS SDK for Python (Boto3). This script acts as the orchestrator between the physical world and the cloud AI. The workflow is implemented as follows:

- Event Listener: The script subscribes to MQTT topics to receive real-time triggers from the ESP32.
- Biometric Verification: Upon a valid card scan, the script captures a frame and uploads it to Amazon S3. It then invokes `Rekognition.compare_faces` to verify the user against the authorized database.
- Logic Execution: If the similarity confidence score exceeds the threshold (90%), the script publishes the "OPEN" command. Otherwise, it logs an "Access Denied" event.

Fig. 4. Real-time execution logs of the backend logic. The system demonstrates the MFA process: detecting a face, communicating with AWS Rekognition, and verifying identity against the database.

C. AWS Cloud Configuration & Security Policies The cloud infrastructure was provisioned using the AWS Management Console with a focus on "Least Privilege" security principles.

- IoT Policies: A restrictive policy was attached to the ESP32 certificate, allowing `iot:Connect`, `iot:Publish`, and `iot:Subscribe` actions only on specific topics (e.g., `door/status`). This ensures that even if the physical device is compromised, it cannot affect other cloud resources.
- User Permissions: AWS IAM (Identity and Access Management) roles were configured to allow the backend script to access S3 and Rekognition services securely without exposing root credentials.

Fig. 5. AWS IoT Core MQTT Test Client console, displaying the real-time message payload exchange between the cloud and the edge device.

D. Version Control and Collaboration Security To ensure code integrity and facilitate collaborative development, the entire codebase is managed via Git and hosted on GitHub. The repository is publicly available at: `https://github.com/Jonathan-dadcha/IoT-Smart-Door-System`.

To maintain a secure development environment, we implemented Role-Based Access Control (RBAC) within the repository settings:

- Branch Protection: Direct commits to the `main` branch were restricted to prevent breaking changes.
- Collaborator Permissions: Specific "Write" and "Maintain" permissions were assigned to authorized team members (e.g., granting access to collaborator *Adar*), while ensuring outside users have read-only access. This segregation of duties mimics a real-world enterprise development environment, minimizing the risk of unauthorized code injection.

V. RESULTS AND DISCUSSION

To evaluate the performance and reliability of the proposed system, a series of functional and stress tests were conducted. The system was tested in a controlled environment simulating a smart building entrance.

A. Functional Testing Results The primary goal was to verify the Multi-Factor Authentication (MFA) logic. We tested three distinct scenarios:

1. Unauthorized Token: When an unregistered RFID card was scanned, the ESP32 immediately denied access locally.
2. MFA Rejection (Spoofing Attempt): A valid RFID card was used, but the user presented a face not associated with the card. As shown in the system logs, the system correctly identified the mismatch and returned an "Access Denied" status.
3. Successful Entry: When a valid user scanned their card and presented their face, AWS Rekognition returned a confidence score averaging 98.5%, triggering the door unlock mechanism.

B. Performance Analysis A critical factor in user experience is the system's latency. We measured the "End-to-End" delay, defined as the time elapsed from the RFID scan to the solenoid activation.

- Average Latency: The system achieved an average response time of approximately 1.5 to 2.5 seconds.
- Discussion: The majority of this latency is attributed to the image upload process and cloud inference. Given the high-security nature of the application, a 2-second delay is an acceptable trade-off for the enhanced security provided by cloud-based AI compared to local, less accurate alternatives.

C. Comparative Analysis Unlike the system proposed by Mahmoud et al., which relied on local SD card storage and suffered from scalability issues, our cloud-native architecture allows for infinite scalability and robust data management via Amazon S3 and DynamoDB.

VI. CONCLUSION

This paper presented the design and implementation of a secure, cloud-integrated IoT Smart Door System. By combining the low-cost ESP32 microcontroller with the powerful computational capabilities of Amazon Web Services, we successfully addressed the limitations of traditional access control systems.

The project demonstrated that Multi-Factor Authentication can be effectively implemented in consumer-grade hardware by leveraging the cloud. Key achievements include enhanced security through AWS certificates and MFA, high scalability via cloud services, and a professional development workflow using Git and GitHub for version control. Future work will focus on implementing "Liveness Detection" to further prevent advanced spoofing attacks and developing a dedicated mobile application for remote management.

VII. REFERENCES

[1] S. A. Alghamdi, "Smart Home Automation System Using IoT and AWS Cloud," *Engineering Proceedings*, vol. 76, no. 1, p. 85, 2024.

[2] I. Mahmoud, I. Saidi, and C. Bouzazi, "Design of an IOT System based on Face Recognition Technology using ESP32-CAM," *IJCSNS International Journal of Computer Science and Network Security*, vol. 22, no. 7, pp. 165-172, July 2022.

[3] P. S. Varma, et al., "IoT Based Smart Parking System," *2024 International Conference on Computer and Information Sciences (ICCIS)*, 2024.

[4] Espressif Systems, "ESP32 Technical Reference Manual," Version 4.1, 2020.

[5] Amazon Web Services, "AWS IoT Core Documentation," [Online]. Available: https://aws.amazon.com/iot-core/.