NT219.P21.ANTT - Post-Quantum Lattice-Based Lightweight Anonymous Authentication for IoT Devices

Do Quang Trung, Nguyen Dinh Khang, Hoang Bao Phuoc

May 2025

Group Information

STT	Full Name	Student ID	Email
1	Do Quang Trung	23521673	23521673@gm.uit.edu.vn
2	Nguyen Dinh Khang	23520694	23520694@gm.uit.edu.vn
3	Hoang Bao Phuoc	23521231	23521231@gm.uit.edu.vn

1 Assets, Stakeholders, and Roles

The primary assets in this project include:

- **Data**: Sensor data (e.g., temperature, pressure), authentication messages, and blinded identifiers (e.g., ID'_T, Ψ'_T, φ') exchanged between IoT devices and the server.
- **Cryptographic Keys/Secrets**: Lattice-based credentials (ID_T, Ψ_T, φ), secret values (X_0, s_1), and session keys (λ, λ') used for authentication and secure communication.
- **Systems and Devices**: IoT client devices (emulated ESP32) and backend server (emulated Raspberry Pi). Network infrastructure, including MQTT communication channels, is also an asset.
- **Intangible Assets**: Device anonymity, untraceability, and trust in the authentication system, critical for protecting IoT deployments in scenarios like smart factories or agriculture.

Stakeholders and their roles are:

- **End-Users**: IoT device operators (e.g., factory workers) who rely on secure, anonymous authentication to access or control devices.
- **Developers**: The project team (us) responsible for designing, implementing, and testing the LBAA protocol.
- **System Administrators**: Manage the backend server (Raspberry Pi), handling key generation, storage, and MQTT broker configuration.
- **Security Engineers**: Conduct threat modeling and security testing to ensure quantum resistance and anonymity.

- **Business Managers**: Oversee IoT deployments (e.g., in smart factories) and define security requirements.
- **Auditors/Regulators**: Verify compliance with security standards and evaluate the system's resistance to quantum and classical attacks.
- **Attackers (Threat Model)**: External adversaries or honest-but-curious servers attempting to compromise anonymity, trace devices, or forge credentials.

2 Security Risk Analysis

A structured risk analysis was conducted using the STRIDE model to identify threats to the assets. The system was decomposed into components (client: ESP32, server: Raspberry Pi, MQTT communication) with trust boundaries (client-server, public-private networks). Key entry points include MQTT topics, client credentials, and server verification processes.

- **Tampering**: Modifying authentication messages or sensor data. *Vulnerability*: Unencrypted MQTT channels. *Likelihood*: Medium. *Impact*: High (data integrity loss). *Mitigation*: Session keys and hash-based verification.
- **Repudiation**: A device or server denies sending/receiving messages. *Vulnerability*: Lack of logging. *Likelihood*: Low. *Impact*: Medium. *Mitigation*: Non-repudiable logging with timestamps and signatures.
- **Information Disclosure**: Leaking device identities or secret values. *Vulnerability*: Poor anonymity mechanisms. *Likelihood*: High. *Impact*: High (loss of anonymity). *Mitigation*: Blinded identifiers and random vectors (r_2, r_3).
- **Denial-of-Service (DoS)**: Overloading the server to disrupt authentication. *Vulnerability*: Unoptimized server operations. *Likelihood*: Medium. *Impact*: High (service disruption). *Mitigation*: Lightweight operations and scalable architecture.

A risk matrix was used to prioritize threats. High-likelihood, high-impact threats (e.g., Information Disclosure, Spoofing) are addressed through the LBAA protocol's anonymity and mutual authentication features. Regular reviews will update the risk assessment as the system evolves.

3 Project Goals

The project aligns with the CIA triad and extended security principles:

- **Confidentiality**: Protect authentication messages and credentials from unauthorized access using lattice-based cryptography (ISIS problem).
- **Integrity**: Ensure authentication data and sensor readings are untampered via hash-based verification and session keys.
- **Availability**: Provide reliable authentication for resource-constrained IoT devices, optimized for low computational overhead.
- **Authenticity**: Verify the identity of both client (ESP32) and server (Raspberry Pi) through mutual authentication.
- **Non-repudiation**: Ensure actions (e.g., authentication requests) cannot be denied using signed messages and logs.

- **Anonymity and Untraceability**: Prevent tracking of IoT devices using blinded identifiers and random vectors.
- **Quantum Resistance**: Ensure security against quantum attacks by relying on the ISIS problem, resistant to Shor's algorithm.

4 Solution Architecture

The solution uses the lattice-based lightweight anonymous authentication (LBAA) protocol based on the Inhomogeneous Small Integer Solution (ISIS) problem, proven secure against quantum algorithms [1]. Key cryptographic components include:

- **Hash Function**: SHA-256 for computing blinded identifiers and verification values (γ, λ).
- **Matrix/Vector Operations**: Small integer matrices (A, $m \times n$) and vectors (r_1, r_2, r_3) over modulus q for lightweight computations.
- **Libraries**: MicroPython-compatible libraries for matrix operations and SHA-256, avoiding custom implementations.

The system is modular, with the following components:

- **Authentication Module**: Handles mutual authentication using a challenge-response mechanism. The server generates challenges (r_1), and the client computes blinded responses (ID'_T, Ψ'_T, φ').
- **Key Management Service**: Manages generation, storage, and distribution of lattice-based credentials (ID_T, Ψ_T, φ, X_0, s_1). Credentials are stored securely in emulated storage (ESP32) or server memory.
- **Communication Module**: Uses Mosquitto MQTT broker for secure client-server communication over TCP/IP, with MQTT topics for challenges and responses.
- **Logging/Auditing**: Logs authentication attempts with timestamps and signatures for non-repudiation.

The system applies defense-in-depth and least-privilege principles:

- Encrypt authentication messages using session keys (λ, λ').
- Restrict server and client privileges to minimal required operations.
- Use MQTT over TLS for secure communication.
- Validate all inputs (e.g., random vectors, hash values) to prevent injection attacks.

5 Demonstration, Deployment, and Testing

The system will be developed and tested in a physical environment:

- **Server**: Raspberry Pi 3 (1.2GHz quad-core ARM Cortex-A53, 1GB RAM) running Raspbian with Mosquitto MQTT broker.
- **Client**: ESP32 development board running MicroPython.
- **Network**: Virtual network with TCP/IP for MQTT communication.
- **Version Control**: Git for code management, ensuring reproducible builds.

Unit and integration tests will verify:

- Correct computation of blinded identifiers (ID'_T, Ψ'_T, φ') and hash values (γ, λ).
- Successful mutual authentication over MQTT.
- Error handling for invalid credentials or malformed messages.
- Compliance with lattice-based parameters (e.g., modulus q, matrix A).

A demo script will showcase authentication in a smart factory scenario, with ESP32 clients authenticating to a Raspberry Pi server.

Security tests include:

- **Anonymity Testing**: Verify that multiple authentication sessions from the same client produce different messages (ID'_T, Ψ'_T) due to random vectors (r_2).
- **Attack Simulation**: Attempt replay, impersonation, and MITM attacks to confirm protocol resistance.

Performance benchmarks will measure:

- Client-side operation time (e.g., matrix-vector multiplication, hashing) on ESP32 development board.
- Server-side verification time on Raspberry Pi 3.
- Total authentication round-trip time.
- Memory usage on ESP32 (target: lower 512KB RAM).

Results will be compared to theoretical costs in [1] and other protocols.

The deployment plan includes: Demonstrate encrypted authentication in a smart factory scenario, with ESP32 sensors sending data to a Raspberry Pi 3 server.

6 Assessment Guidelines

The project will be evaluated across the following dimensions:

- **Research & Background**: Cites recent sources [1, 2] and explains lattice-based cryptography and IoT security challenges.
- **Goals & Threat Analysis**: Clearly defines CIA, authenticity, anonymity, and quantum resistance goals. Uses STRIDE for comprehensive threat modeling.
- **Solution Design & Effectiveness**: Employs LBAA protocol with ISIS problem, secure key management, and defense-in-depth principles.
- **Technical Implementation**: Implements modular MicroPython code for ESP32 and Raspberry Pi, with error handling and no security flaws.
- **Testing & Validation**: Includes unit, integration, and security tests, with performance benchmarks and attack simulations.
- **Documentation & Reporting**: Provides a clear report with design decisions, setup instructions, data flow diagrams, and professional formatting.

7 References

References

[1] Fathenojavan, S., Moeini, A., & Javadi, H. H. S. (2025). A post-quantum lattice-based lightweight anonymous authentication scheme for IoT. *Journal of Cybersecurity*, 11(1), tyafo04.

[2] Tandel, P., & Nasriwala, J. (2024). Secure authentication framework for IoT applications using a hash-based post-quantum signature scheme. *Service Oriented Computing and Applications*, 1–12.