

AI-Driven Quantum-Safe IoT Security Framework for Real-Time Threat Detection

Mrs. M. Gomathi¹, S. Sowndharya²

¹Assistant Professor, Department of Computer Science and Engineering, Vivekanandha College of Engineering for Women, Tamilnadu, India

²PG Scholar, Department of Computer Science and Engineering, Vivekanandha College of Engineering for Women, Tamilnadu, India

¹Email: gomathim82@gmail.com

²Email: ssowndharya0220@gmail.com

Abstract—Internet of Things (IoT) devices now run in factories, hospitals, buildings and other critical infrastructure, so the security of their communication matters more every year. Most deployments protect keys and identities with RSA or elliptic-curve cryptography (ECC), which a large fault-tolerant quantum computer could break. At the same time, IoT networks are exposed to everyday attacks such as denial of service, port scanning and spoofing. In practice, cryptographic protection and intrusion detection are built as two unrelated pieces. This paper proposes a framework that studies them together. It pairs the NIST post-quantum standards ML-KEM (key establishment) and ML-DSA (digital signatures) with a Random Forest classifier and a Long Short-Term Memory (LSTM) network for traffic analysis. An ESP32-class edge device, MQTT messaging, a signed event log and a monitoring dashboard complete the design. We describe the problem, the research gap, the architecture and the evaluation plan. The prototype is still to be built, so no measured results are reported.

Keywords—Internet of Things, post-quantum cryptography, ML-KEM, ML-DSA, ESP32, MQTT, intrusion detection, Random Forest, LSTM.

I. INTRODUCTION

The Internet of Things (IoT) links sensors, actuators and cloud services across smart homes, factories, power grids, transport systems and hospitals. Many of these devices have little processing power, memory or battery life, and they are rarely maintained by hand. Any security measure for them has to protect data and device identity while staying light enough to deploy. Security here means confidentiality, integrity, authentication, availability and accountability. Encryption keeps data private, signatures prove where a message came from and that it was not altered, and intrusion detection spots traffic that behaves abnormally. None of these can stand in for another. Strong encryption, for example, says nothing about whether a legitimate device has started sending malicious traffic.

Today's public-key systems, RSA and ECC, are secure because factoring and discrete logarithms are hard for classical machines. Shor's algorithm [4] removes that hardness once a sufficiently large and reliable quantum computer exists. Nobody can say exactly when that will happen, but data and equipment with long lifetimes need a migration plan now, because an attacker can store encrypted traffic today and decrypt it later. In 2024 NIST released three post-quantum standards: ML-KEM [1], ML-DSA [2] and SLH-DSA [3] [10]. They give researchers a concrete starting point for quantum-resistant key exchange and signatures in IoT.

Machine learning adds a different kind of protection. A Random Forest [12] is an interpretable baseline for tabular flow features, and an LSTM [11] can pick up patterns that unfold over time. We use both as one layer of a wider design; neither replaces cryptographic checks.

A. Integration Challenges

Putting post-quantum cryptography (PQC) and machine-learning detection on the same IoT platform raises several difficulties. PQC keys, signatures and working buffers are larger than their classical counterparts, so memory use and message size grow, and some operations may not fit on a microcontroller at all [6]. Learned models depend on good training data and can overfit or leak test information, and they will raise some false alarms and miss some attacks. There is also a question of where PQC sits in the protocol stack: encapsulating keys inside the application is not the same thing as running a standardized PQC-enabled TLS stack [13], and a prototype should say which one it uses.

B. Objectives and Contributions

Our goal is to design and evaluate a framework that combines post-quantum protection with AI-based intrusion detection for IoT communication, while keeping an eye on responsiveness and what an edge device can afford. Specifically, we aim to:

- review IoT security threats and the current NIST PQC standards;
- design a message flow that uses ML-KEM for shared secrets and ML-DSA for signatures;
- build a traffic-monitoring pipeline around a Random Forest baseline and an LSTM sequence model;
- connect secure MQTT messaging and a monitoring dashboard in one prototype; and
- measure detection quality, cryptographic timing, memory use and communication overhead.

The main contribution is an architecture and an evaluation method that look directly at how PQC and AI detection interact and what each costs, instead of assuming the combination will meet a chosen performance target. Three research questions guide the work. How can PQC and AI-based detection be coordinated in one IoT architecture? What trade-offs arise among detection quality, cryptographic overhead and device resources? Which metrics show real usefulness beyond a single accuracy figure?

C. Scope

We target small IoT deployments and industrial-style telemetry. An ESP32-class board handles sensing and publishing. Cryptographic steps that exceed what the microcontroller can realistically do may move to a host or gateway; the exact split will be settled after profiling the libraries and memory. This paper covers design and literature only. Implementation results are not yet available, and the quantum resistance of any final system will depend on correct algorithm use, parameter choice, implementation quality and key management.

II. RELATED WORK

A. Post-Quantum Standards

FIPS 203 defines ML-KEM, a key-encapsulation mechanism built on the module learning-with-errors problem and derived from CRYSTALS-Kyber [1]. FIPS 204 defines ML-DSA, derived from CRYSTALS-Dilithium [2], and FIPS 205 defines SLH-DSA, a stateless hash-based signature scheme [3]. These documents specify algorithms, not finished IoT products. A real deployment still needs secure key storage [5], protocol integration, validation and lifecycle management.

B. Cryptography on Constrained Devices

A microcontroller has far less RAM, flash and processing speed than a server. Large PQC keys and signatures, temporary buffers and polynomial arithmetic all affect memory and latency. Projects such as PQM4 on ARM Cortex-M4 [6] benchmark individual primitives. A fuller study should time key generation, encapsulation and decapsulation, signing and verification, record stack usage and, where possible, energy, and should keep the cost of a primitive separate from the cost of the whole protocol.

C. Machine-Learning Intrusion Detection

An intrusion detection system (IDS) labels observed traffic as benign or suspicious. A Random Forest averages many decision trees, handles non-linear relationships and reports which features matter most, although it can use a lot of memory and struggles with imbalanced classes. LSTM networks read ordered sequences, but how well they work depends on how the sequences are built, how features are scaled and how good the data is.

D. Datasets and Evaluation Pitfalls

CICIDS2017 [7] holds labelled benign and attack traffic, TON_IoT offers telemetry and network records from IoT and IIoT settings, and UNSW-NB15 [8] is another popular benchmark. These sets make experiments repeatable, but they must be used carefully. Features can be specific to one dataset, and duplicate records, class imbalance, leakage between training and test data, and random splits that ignore time can all

make results look better than they are. When generalization is the question, a time-aware or device-aware split is the better choice.

E. Secure MQTT and Event Logging

MQTT [9] is a common choice for telemetry because its publish/subscribe model is light. A secure setup still has to authenticate clients, restrict access by topic, protect the transport, rotate credentials and validate messages. Post-quantum key establishment can be added at the transport or the application layer, and the prototype should state which. For tamper evidence, an append-only log of hashes and signatures is often enough; a distributed ledger is not automatically needed.

F. Problem Analysis and Research Gap

The building blocks above are mature, but drawing them together in a diagram does not prove that they work together. Table I compares the main approaches and Table II lists the gaps this work targets.

TABLE I. COMPARISON OF SECURITY APPROACHES

Approach	Strength	Limitation	Role
RSA/ECC	Mature, widely supported	Vulnerable to future quantum computers	Baseline
PQC only	Quantum-resistant keys and signatures	Cannot spot malicious traffic from valid endpoints	Crypto layer
AI IDS only	Finds known and anomalous traffic	Key exchange not quantum resistant	Detection layer
PQC + AI IDS	Protection and monitoring together	Integration cost still unmeasured	Target design
+ signed log	Traceable security events	Extra storage and complexity	Optional

TABLE II. RESEARCH GAP ANALYSIS

Gap	Why it matters	Planned response
PQC and IDS studied separately	A secure channel does not reveal abnormal behaviour	Combine the crypto workflow with traffic monitoring
Little edge-resource evidence	PQC and AI compete for memory and compute	Profile latency, memory and offloading
Heavy reliance on accuracy	Class imbalance hides false alarms	Report precision, recall, F1, confusion matrix and FPR
Unclear protocol boundary	Application-level crypto is not PQC-TLS	State the protocol layer and implementation status
Weak reproducibility	Data leakage inflates results	Record dataset version, preprocessing and split

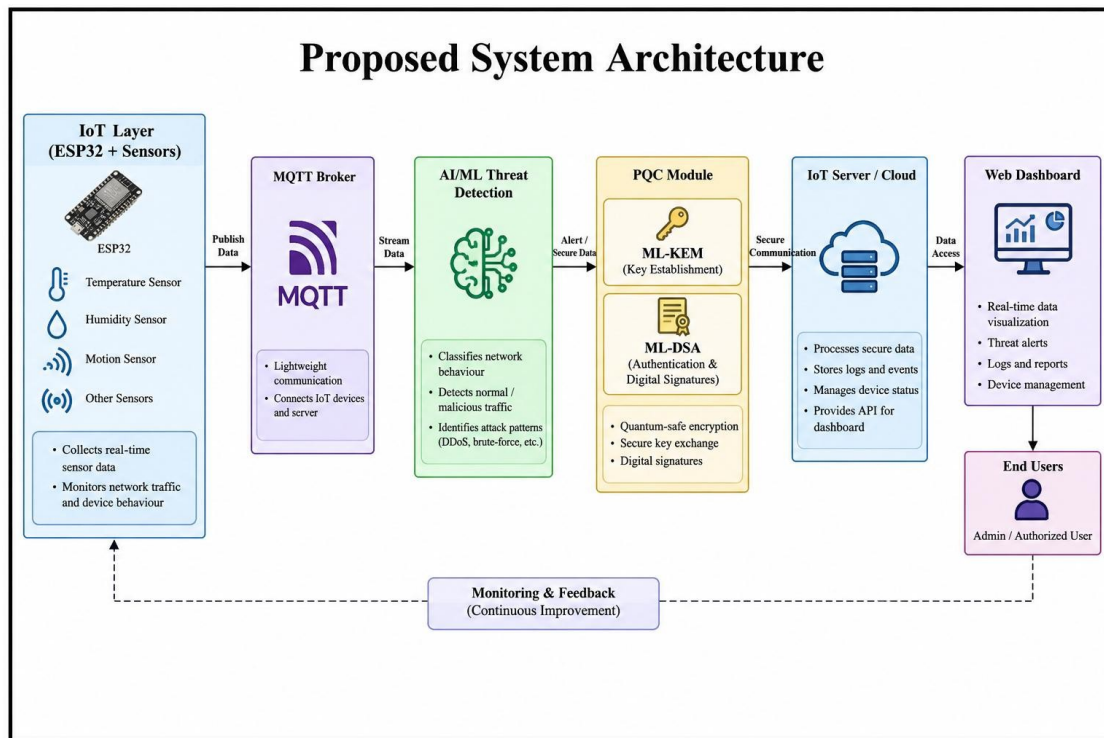


Fig. 1. Proposed system architecture: IoT layer (ESP32 and sensors), MQTT broker, AI/ML threat detection, PQC module (ML-KEM, ML-DSA), IoT server/cloud and web dashboard.

III. PROPOSED SYSTEM

A. Overview

The framework has five main parts: a sensing endpoint, a cryptographic protection module, an MQTT communication path, a traffic-analysis service and a monitoring interface. An optional integrity-protected log records selected alerts and audit events. We keep the design modular so that the cryptography and the AI can be tested separately before they are joined end to end. Fig. 1 shows the architecture and Table III lists the layers. The figure shows functional modules only; it does not mean they are already implemented. Where the PQC operations run will be decided once the library and the ESP32 limits are known.

TABLE III. LAYERS AND COMPONENTS

Layer	Component	Responsibility
Sensing	Sensor / ESP32	Collect telemetry and form messages
Cryptography	ML-KEM + ML-DSA library	Establish secrets; sign and verify
Messaging	MQTT client and broker	Route telemetry with authenticated topic access
Monitoring	Feature extractor	Prepare features for the IDS
AI detection	Random Forest + LSTM	Classify flows; detect sequence anomalies
Audit/UI	Signed log, dashboard	Show alerts; support investigation

B. Design Principles

- Use standardized algorithms and record library versions.

- Keep identity, key establishment, message integrity and anomaly detection separate.
- Keep the edge device light and profile operations before offloading them.
- Use reproducible splits with a held-out test set.
- Make alerts easy to interpret.
- Treat performance targets as hypotheses, not as results.

C. Threat Model

We consider passive traffic collection, message tampering, unauthorized MQTT publishing or subscribing, replay attempts, denial-of-service traffic, and compromised or misconfigured devices. PQC strengthens key establishment and signatures, but it cannot stop every denial-of-service attack or make up for a weak endpoint. Rate limiting, topic authorization, credential rotation, secure firmware practices and continuous monitoring are still required.

IV. METHODOLOGY

A. Post-Quantum Cryptography Module

ML-KEM (FIPS 203) establishes the shared secret, and ML-DSA (FIPS 204) signs and verifies messages or identity objects. At session setup, the endpoint or gateway runs encapsulation and decapsulation to obtain the secret, following the chosen protocol design. Each operation will be tested on its own, with the library version, parameters, timing and memory use recorded. Keys must never be written to ordinary logs.

B. MQTT Messaging and Event Logging

The sensor publishes structured telemetry that carries its device identity and a timestamp, over an authenticated MQTT connection with topic-level authorization. Security events go

into a signed, append-only log that stores the timestamp, alert type, model score and integrity metadata, and avoids keeping sensitive payloads that are not needed. Ledger-style or IPFS-like storage is treated as an optional extension, and its benefit will be weighed against its cost.

C. Data Preparation

We plan to use CICIDS2017 and/or TON_IoT, subject to feature mapping and licence terms. Preparation removes invalid records, encodes labels, fills or drops missing values and documents the class distribution. Normalization parameters are fitted on training data only, a time-aware or device-aware split is preferred, and the test set is held out and left untouched.

D. Random Forest Baseline

The forest is trained on many decision trees, each fitted to a bootstrap sample and a random subset of features, and the trees vote on the final class [12]. A small set of hyperparameters is tuned on validation data, and results are reported on the separate test set. Feature importance scores keep the baseline transparent.

E. LSTM Sequence Detector

The LSTM [11] reads sequences of observations ordered by time or flow context, which lets it learn temporal patterns that flow-level classifiers can miss. Sequence length and stride are chosen using validation data. Depending on the labels available, the output is either a class label or an anomaly score. Training cost and explainability will need attention.

F. Integrating PQC and AI Detection

The two layers complement each other. PQC protects key establishment and message authenticity, while the Random Forest and LSTM look for abnormal traffic from valid or compromised endpoints. The AI module does not replace cryptographic validation. A message travels through the system as follows: (1) the sensor forms a telemetry message with its identity and a timestamp; (2) ML-KEM establishes a session secret; (3) ML-DSA signs or verifies the identity or message object; (4) the message is published through the authenticated MQTT broker; (5) features are derived from permitted network metadata and broker events; (6) the Random Forest and LSTM score the traffic and forward suspicious events to the dashboard; and (7) an audit record with integrity metadata is stored. Table IV summarizes each module.

TABLE IV. MODULE DESCRIPTION

Module	Input → Output	Processing
Sensor acquisition	Readings → telemetry record	Read, validate and timestamp
PQC key establishment	Public key / encapsulation → shared secret	ML-KEM encapsulation and decapsulation
Digital signature	Message or identity → signature, status	ML-DSA sign and verify

Module	Input → Output	Processing
MQTT messaging	Payload → delivered message	Publish/subscribe with client and topic controls
Feature extraction	Flow metadata → feature vectors	Clean, transform and window
Random Forest IDS	Feature vectors → class, probability	Train and classify flows
LSTM IDS	Sequences → anomaly score	Learn temporal patterns
Dashboard/audit	Alerts → operator events	Visualize, filter and keep integrity evidence

V. PLANNED EVALUATION

Implementation will proceed in stages: requirements and threat modelling; environment setup (ESP32, sensor, ESP-IDF or Arduino, PQC library, MQTT broker); cryptographic unit tests; dataset preparation; model training; integration of MQTT, dashboard and logging; security testing; and performance evaluation. Security tests will check that invalid signatures are rejected, unauthorized topic access is denied, malformed messages are handled and alerts are generated.

Detection quality will be reported with precision, recall, F1-score, a confusion matrix and the false-positive rate, not with accuracy alone. System cost will be reported as cryptographic latency (key generation, encapsulation and decapsulation, signing and verification), end-to-end delay, RAM, flash and stack usage, and the communication overhead of larger keys and signatures. The software stack is Python with scikit-learn and TensorFlow/Keras, plus version control and experiment logs for reproducibility. The hardware is an ESP32 board with a DHT11 or simulated sensor stream and a laptop or desktop gateway for training, the broker and the dashboard. Early experiments may use a simulator or a host-based test harness.

VI. CONCLUSION AND FUTURE WORK

This paper proposed a coordinated IoT security design that addresses both the future risk to public-key cryptography and the network threats of today. It combines the NIST standards ML-KEM and ML-DSA, MQTT communication and Random Forest and LSTM intrusion detection. The literature review showed that integration, resource profiling, reproducible evaluation and clear protocol boundaries remain open issues. The modular architecture allows staged development and independent testing of each component. All performance values in this paper are planned targets for evaluation, not experimental findings.

Future work will select and validate a PQC implementation that suits the device and gateway split, finish the threat model and decide which messages or identities are signed, and prepare leakage-resistant datasets. We will then train and compare the two detectors on several metrics, integrate MQTT,

cryptography, detection and a dashboard, and measure latency, memory, communication overhead and false-positive rate in repeatable tests.

ACKNOWLEDGMENT

The authors thank the Department of Computer Science and Engineering, Vivekanandha College of Engineering for Women, for its support.

REFERENCES

- [1] National Institute of Standards and Technology, “Module-Lattice-Based Key-Encapsulation Mechanism Standard,” FIPS 203, Aug. 2024, doi: 10.6028/NIST.FIPS.203.
- [2] National Institute of Standards and Technology, “Module-Lattice-Based Digital Signature Standard,” FIPS 204, Aug. 2024, doi: 10.6028/NIST.FIPS.204.
- [3] National Institute of Standards and Technology, “Stateless Hash-Based Digital Signature Standard,” FIPS 205, Aug. 2024, doi: 10.6028/NIST.FIPS.205.
- [4] P. W. Shor, “Algorithms for quantum computation: Discrete logarithms and factoring,” in Proc. 35th Annu. Symp. Found. Comput. Sci. (FOCS), 1994, pp. 124–134, doi: 10.1109/SFCS.1994.365700.
- [5] E. Barker, “Recommendation for key management: Part 1 – General,” NIST Spec. Publ. 800-57 Part 1 Rev. 5, May 2020, doi: 10.6028/NIST.SP.800-57pt1r5.
- [6] M. J. Kannwischer, J. Rijneveld, P. Schwabe, and K. Stoffelen, “PQM4: Testing and benchmarking NIST PQC on ARM Cortex-M4,” in Proc. 8th ACM ASIA Public-Key Cryptography Workshop, 2021.
- [7] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, “Toward generating a new intrusion detection dataset and intrusion traffic characterization,” in Proc. ICISSP, 2018, pp. 108–116, doi: 10.5220/0006639801080116.
- [8] N. Moustafa and J. Slay, “UNSW-NB15: A comprehensive data set for network intrusion detection systems,” in Proc. Mil. Commun. Inf. Syst. Conf. (MilCIS), 2015, pp. 1–6, doi: 10.1109/MilCIS.2015.7348942.
- [9] OASIS, “MQTT Version 5.0,” OASIS Standard, Mar. 2019.
- [10] National Institute of Standards and Technology, “Post-Quantum Cryptography,” NIST Computer Security Resource Center. [Online]. Available: <https://csrc.nist.gov/projects/post-quantum-cryptography>
- [11] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural Comput., vol. 9, no. 8, pp. 1735–1780, 1997, doi: 10.1162/neco.1997.9.8.1735.
- [12] L. Breiman, “Random forests,” Mach. Learn., vol. 45, no. 1, pp. 5–32, 2001, doi: 10.1023/A:1010933404324.
- [13] NIST, “Guidelines for the selection, configuration, and use of Transport Layer Security (TLS) implementations,” NIST Spec. Publ. 800-52 Rev. 2, Aug. 2019, doi: 10.6028/NIST.SP.800-52r2.