The First International Conference on Cross-Domain Security
in Distributed, Intelligent and Critical Systems (CROSS-SEC 2026)

Secure-by-Design Prototyping of an IoT Access-Control System

Oliver Vainikko[1], **Ulrich Norbisrath**[1], **Ruben Jubeh**[2]

[1] Department of Computer Science, University of Tartu, Estonia

[2] Faculty of Computer Science and Mathematics, OTH Regensburg, Germany

Presenter: Prof. Dr. Ruben Jubeh · ruben.jubeh@oth-regensburg.de

About the Presenter

Prof. Dr. Ruben Jubeh
Faculty of Computer Science and Mathematics, OTH Regensburg, Germany

Research focus: Internet of Things (IoT), Software Engineering, Edge AI, Embedded Machine Learning, and IoT Security

Active collaboration with the **University of Tartu** (Estonia) on the IoTempower framework (U. Norbisrath)

Teaching Focus: IoT, Cyber Physical Systems, Prototype System Development

1 The Prototyping–Security Tension

Rapid IoT prototyping in universities and industry training delivers **functional systems fast** — but routinely ships with insecure defaults that persist into final demonstrations.

Frameworks like **IoTempower**, Tasmota, and ESPHome abstract away networking and messaging — and, with it, security configuration.

> Security is *deferred* until "later" — and "later" never comes before the deadline.

Rapid Iteration

Declarative config, OTA updates, MQTT messaging — prototypes run in hours, e.g. the access control system (demo)

Security Debt

Plaintext messaging, shared/default passwords, no device identity — left as-is

2 Research Questions

RQ1 How can a **secure-by-design architecture** be integrated into prototyping workflows?

RQ2 Which security mechanisms can be introduced **in phases** to reduce developer friction?

RQ3 How can **collaborative threat modeling** support security awareness in mixed-experience student teams?

RQ4 Which **framework-level extensions** meaningfully reduce recurring misconfigurations?

> Goal: Make secure operation the **expected end state** of educational prototypes — without a last-minute hardening sprint.

3 IoTempower: Educational IoT Framework

Open-source framework for **ESP8266/ESP32** rapid prototyping, used at several European universities.

Typical Setup

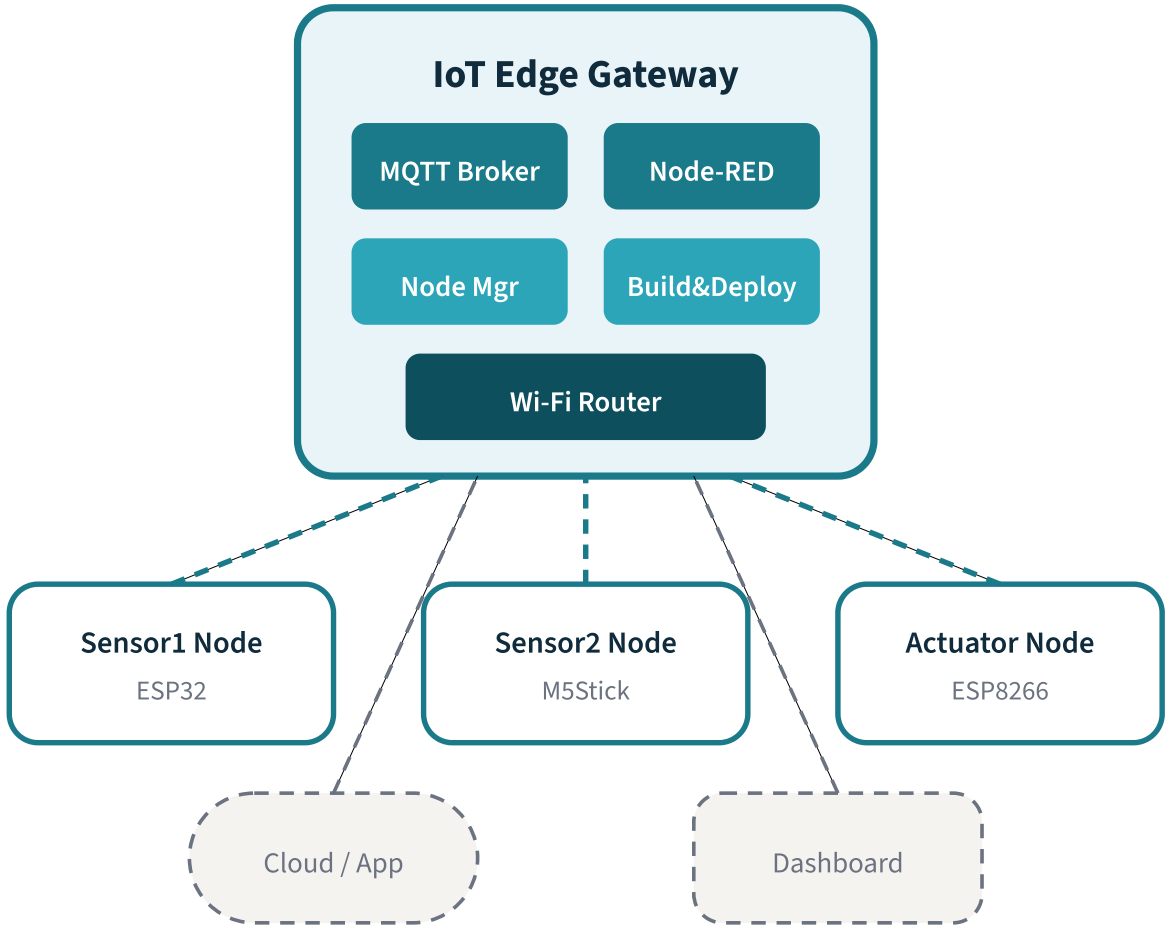
Gateway (Raspberry Pi / laptop) + dedicated Wi-Fi router + MQTT broker + Node-RED integration

Developer Experience

Declarative node configs → IoTempower handles networking, MQTT pub/sub, OTA flashing

Key Features

Device Management, flexible integration, works offline



4 Access-Control Use Case

Edge-local RFID/NFC door access control — suitable for a lab or office environment.

Trust boundary: Gateway = trusted infrastructure. Nodes = exposed (attacker may physically access them). MQTT broker = shared message bus — if you can reach it, you can observe and inject.

5 Insecure Prototype Baseline

Three recurring properties observed in typical classroom IoTempower deployments:

Plaintext MQTT

Port 1883 with weak or shared authentication — all messages visible on the network

Shared Identities

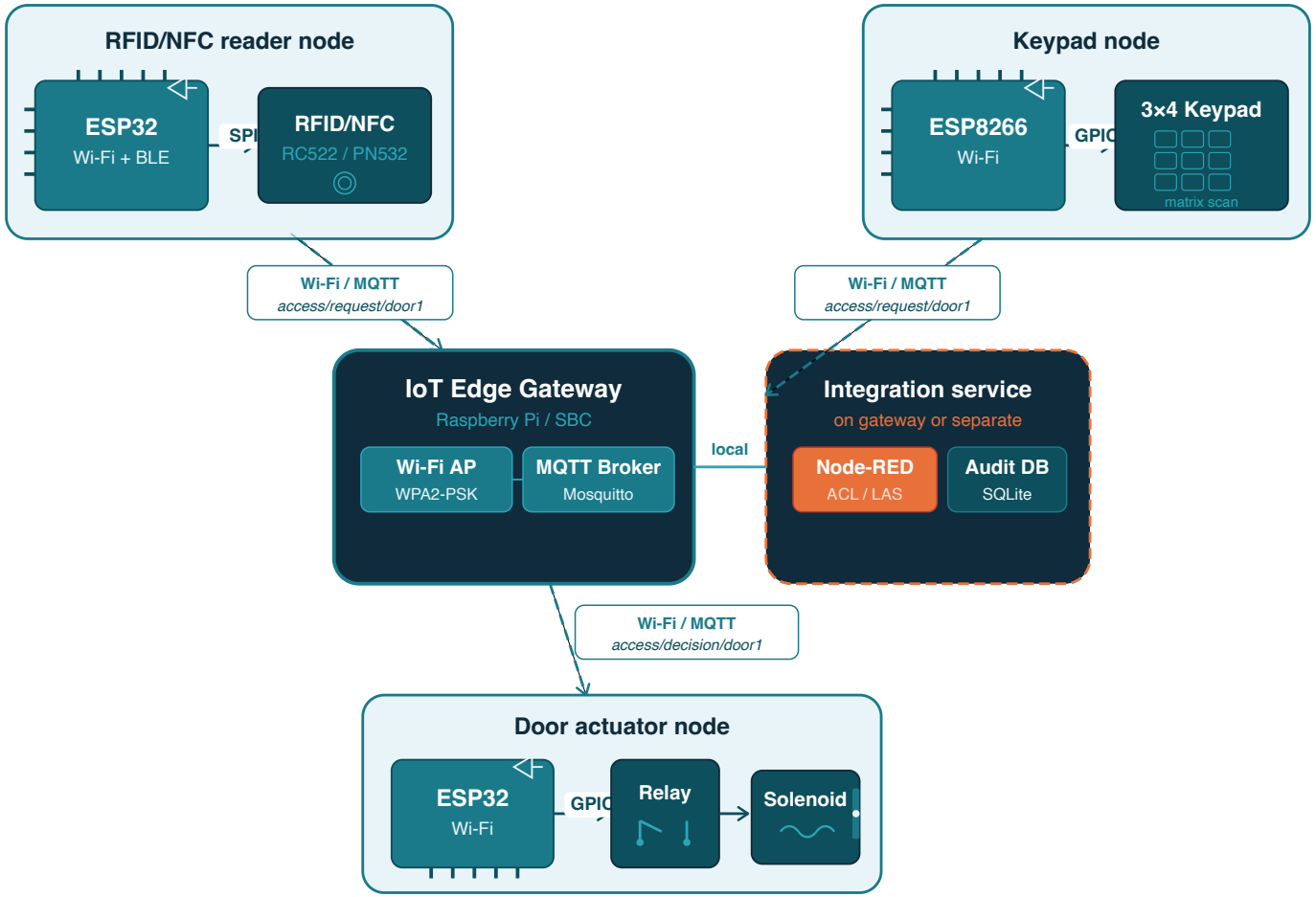
All nodes share the same Wi-Fi PSK, MQTT credentials, and OTA password

Weak OTA Protection

Updates protected only by a password hash; sensitive config embedded in firmware

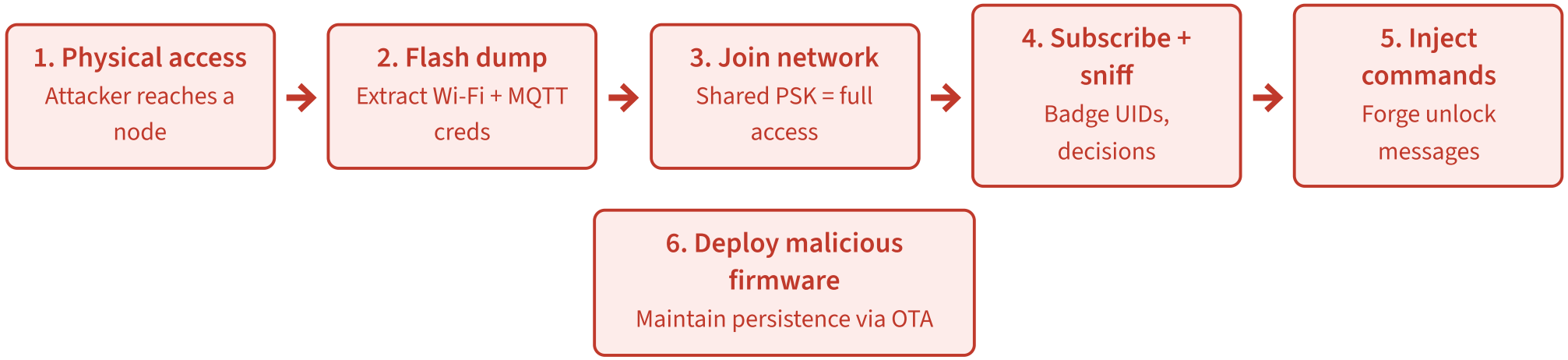
> IoTempower is **secure-capable** but not **secure-by-default** — TLS and ACLs are supported but rarely configured in time.

6 Access Control Reference Architecture



7 Attack Scenario: Single-Device Compromise

Shared secrets + plaintext messaging → one compromise cascades system-wide.



Root cause: One device breach = total breach when all nodes share the same credentials.

Secure-by-Design Reference Architecture

Addressing baseline weaknesses with three core principles

8 Architecture: Four Pillars (1/2)

Unique Device Identity

Each node gets a unique credential (X.509 client certificate + key). Broker authenticates per-device → enables individual revocation. Breaks "one breach = total breach."

TLS-Only MQTT

Broker accepts connections only over TLS; plaintext rejected. Mutual authentication prevents unauthorized clients even with Wi-Fi access.

Least-Privilege Authorization

Broker-side ACLs restrict each identity to specific topics and operations. Compromised reader can't publish unlock; compromised actuator can't subscribe to other doors.

OTA & Audit Trail

Per-device update secrets (not fleet-wide). Firmware signing where possible. Gateway logs auth failures and denied access for auditing.

9 Architecture (2/2)

Identity & TLS

Each ESP node provisioned with its own **X.509 client cert + key**. Broker rejects plaintext (port 1883 disabled); mutual TLS enforced.

Compromise of one node yields no credentials for others — individual revocation without fleet disruption.

ACLs & OTA

```
rfid_reader_door1 → PUB access/request/door1
actuator_door1   → SUB access/decision/door1
las_service      → SUB access/request/# & PUB
access/decision/#
```

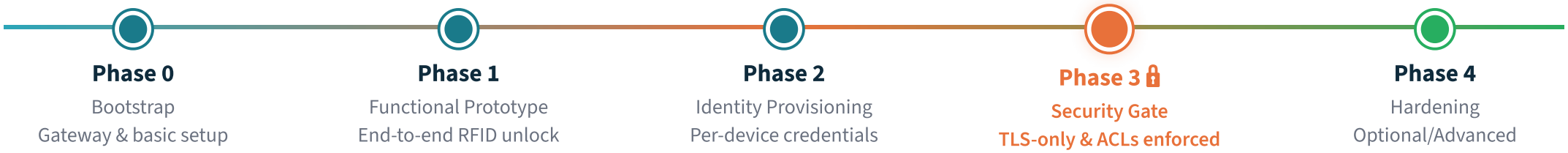
OTA secrets are **per-device**; firmware signing prevents untrusted updates. Gateway logs auth failures and denied access.

Phased Security Model

Aligning security upgrades with functional milestones

10 Phased Security Model: Overview / Timeline

Security controls are layered incrementally, with a **mandatory security gate** at Phase 3.



> Phase 3 is a **precondition** for final demonstrations or grading — a system that "works" must also "work securely."

11 Phased Security Model

Security controls layered incrementally — **Phase 3 is a mandatory gate** before grading or demo.

Phase 0 — Bootstrap

Gateway, broker, initial nodes. Temporary dev secrets acceptable.

- Basic network segmentation
- Consistent topic naming from the start

Phase 1 — Functional Prototype

End-to-end workflow: badge → LAS decision → unlock.

- Separate request/decision topics
- Enable access event logging
- Remove wildcard subscriptions

Phase 2 — Identity Provisioning

Prepare for stronger guarantees without forcing TLS yet.

- Per-device X.509 certs generated
- Broker accounts pre-created
- ACL entries defined per identity

Phase 3 — Security Gate

Insecure fallbacks removed. Precondition for grading/demo.

- MQTT **TLS-only**; port 1883 disabled
- ACLs enforced; shared credentials removed
- **→ Collaborative Threat Modeling**

12 Collaborative Threat Mapping

Review Bundle

Each team prepares:

- Architecture diagram
- Structured list of MQTT topics & roles
- Selected config excerpts (no private keys)
- Description of security controls implemented

Outcome

Vulnerability report with concrete findings + proposed mitigations.
Original team must address issues or provide justified explanation before final assessment.

Process

1. Team A completes Phase 3 baseline
2. Prepares review bundle
3. Team B applies STRIDE-lite checklist
4. Submits vulnerability report
5. Team A addresses findings

13 Discussion & Limitations

Threat Coverage

Focuses on MQTT-related threats (spoofing, tampering, lateral movement). Does not fully mitigate DoS (broker overload, Wi-Fi jamming), supply-chain compromises, or sophisticated physical attacks. Intentional for pedagogical scope.

Peer Review Variability

Quality of threat mapping depends on team experience levels. STRIDE-lite checklists standardize the process, but uneven depth is possible. Requires instructor calibration and iterative checklist refinement.

Isolation ≠ Security

Even local-only deployments have exposed surfaces: Node-RED UI, MQTT, SSH, OTA port. Teams must document and verify their secure setup through measurements (traffic analysis, flash extraction tests).

Usability–Enforcement Balance

The phased model defers resource-intensive tasks until the system is stable, while ensuring insecure fallbacks cannot persist into final demonstrations.

14 Conclusion & Future Work

Key Takeaway

Secure operation can become the **expected end state** of educational IoT prototypes — without sacrificing development speed. The integration of secure defaults, a phased security gate, and collaborative threat modeling makes this practical for novice teams.

Contributions

- Secure-by-design reference architecture for edge-local systems
- Phased security model with mandatory gate before deployment
- STRIDE-lite collaborative threat-mapping exercise for student teams
- Lightweight IoTempower extensions encoding secure defaults

Future Work

Empirical evaluation at scale across multiple course iterations, qualitative/quantitative

Tighter per-device credential provisioning and revocation support, signed firmware

Automated Assessment? Post Quantum Cryptography?

Thank You

Questions & Discussion

Oliver Vainikko · Ulrich Norbisrath · Ruben Jubeh

{oliver.vainikko | ulrich.norbisrath}@ut.ee · ruben.jubeh@oth-regensburg.de

University of Tartu · OTH Regensburg

github.com/iotempire/iotempower

Gap Analysis: Standards vs. Practice

Mapping baseline prototype against **ETSI EN 303 645** and **NISTIR 8259A** expectations:

Default Credentials

Standard: No universal default passwords.

Gap: All nodes share the same Wi-Fi PSK and MQTT creds.

Transport Security

Standard: Secure communications required.

Gap: MQTT on plaintext port 1883 is the norm.

Device Identity

Standard: Unique device identification.

Gap: Non-unique identities across the fleet.

Secure Updates

Standard: Protected update mechanisms.

Gap: OTA protected only by shared password hash.

Not aiming for formal compliance — identifying **highest-leverage gaps** to close first.

15 STRIDE-Lite Checklist for IoT Projects (Case: Access Ctrl)

S Spoofing

Can a rogue client impersonate a device or the LAS to publish unlock commands?

T Tampering

Can MQTT messages or allowlists be modified in transit or on the gateway?

R Repudiation

Are door events logged so actions can be attributed to identities?

I Info Disclosure

Do badge IDs, credentials, or configs leak over the network or in storage?

D Denial of Service

What happens if Wi-Fi is jammed or the broker is flooded — does the door fail-secure?

E Elev. of Privilege

Can a low-privilege node publish admin topics or bypass LAS decisions?

> Adapted from Microsoft's STRIDE — tailored to devices, topics, and trust boundaries rather than enterprise IT.

16 IoTempower Extensions for Secure Defaults

Making the **secure path the default** — reducing the effort of "doing the right thing."

Secure Project Scaffolding

Default broker config enables TLS, disables anonymous access, includes example ACL file reflecting role separation. Plaintext MQTT is no longer the starting point.

Credential Generation Utility

Single command creates a local CA, broker/server certs, and per-device client credentials with device-ID mapping. Eliminates the temptation of shared secrets.

Embedded Security Checklist

Structured checklist + threat-mapping worksheet aligned with development phases. Milestone reminders prompt teams at phase transitions.

Classroom Mode

Instructors can selectively enable temporary insecure settings for demonstrating attacks, while the default remains aligned with the secure baseline.

17 Preliminary Observations & Evaluation Plan

Observations to Date

Across prior course iterations, recurring issues consistently appeared: plaintext MQTT, shared credentials, over-permissive topic access. These motivated introducing the **Phase 3 security gate**.

Planned Evaluation

Upcoming course iteration: ~20 students across 5 teams.

Quantitative

Number and severity of vulnerabilities found in peer review; quality of implemented fixes — collected across multiple courses.

Qualitative

Surveys and interviews on perceived friction, security awareness, and usefulness of framework extensions.