LeafSense Scouts

Security Design Document

Threat model, defence layers, and justification of design choices

6ENT1180 — Robot Communications Coursework

Tomiwa Adeyemi

University of Hertfordshire April 2026

---

1. Purpose and Scope

This document describes the security architecture of the LeafSense Scouts robot-to-robot communication system. It establishes the operating context and threat model, then maps threats to the four implemented defence layers, justifies each design choice against published standards, and acknowledges the limitations that fall outside the coursework scope. It is intended as a supporting artefact to the main coursework submission and provides the reasoning behind choices demonstrated in the video.

2. Operating Context

LeafSense Scouts is a two-node robot-to-robot (R2R) wireless network for agricultural crop disease surveillance. A Field Scout patrols the field detecting diseased plants and transmits compact event messages to a Base Scout. The Base maintains a persistent 256-zone field disease map in an on-board EEPROM and returns encrypted acknowledgements. The context grounds each security decision:

- **Commercially sensitive data.** Yield and disease information is proprietary farm data — competitors or insurers could exploit it.
- **Time-sensitive but not safety-critical.** A delayed frame has low consequence; a forged “no disease” report is moderate consequence.
- **Small, continuous traffic.** Four-byte payloads every two seconds over IEEE 802.15.4 at 2.4 GHz — bandwidth constraints favour lightweight primitives.
- **Resource-constrained endpoints.** STM32F411 Cortex-M4 MCUs with no hardware AES accelerator — algorithm choice must balance security with CPU cost.

3. Threat Model

The following threats were identified for the LeafSense operating context. Each is mapped to the defence layer or layers that mitigate it. Severity reflects the likely impact of a successful exploit on the integrity of the field disease map and the confidentiality of commercial data.

Threat	Description	Severity	Mitigation Layer
Passive eavesdropping	Competitor / curious party captures radio frames with their own XBee to read disease data	Medium	Layer 1 + Layer 2 (Zigbee AES + App AES)
Active injection	Attacker forges frames claiming fake events, corrupting the field disease map	High	Layer 3 + Layer 4 (UID allow-list + nonce)
Replay	Attacker captures a legitimate encrypted frame and rebroadcasts it later	Medium	Layer 4 (monotonic nonce counter)
Clone substitution	Attacker extracts firmware binary from JTAG and flashes it to their own Nucleo	High	Layer 3 (UID allow-list — factory ID is non-rewritable)
Cross-farm interference	Neighbouring farm with similar hardware causes accidental interference	Low	Unique PAN ID + unique encryption key

4. Defence-in-Depth Architecture

Four independent security layers were implemented, each addressing a different threat class. Each layer uses different key material so that the compromise of one layer does not cascade to the others.

#	Mechanism	Key Material	Design Justification
1	Zigbee AES-128 (XCTU)	Network key KY (128-bit): `BAEADEADBEEFCAFE12345678`. Pre-shared, stored in XBee chip non-volatile memory.	Transparent radio-layer defence. Any XBee without the key cannot decrypt captured frames or join the PAN (EO=2 enforces secure joining).
2	Application-layer AES-128 ECB	Application key APP_AES_KEY (128-bit): ASCII `LeafSenseAgri202`. Stored in firmware constant; separate from Zigbee key.	Defence in depth: encrypted independently of the Zigbee link. Even if an XBee is captured and its link key extracted, payload remains opaque.

#	Mechanism	Key Material	Design Justification
3	UID allow-list	STM32F411 96-bit factory UID (word 0 only): Base = `0x00200049`, Field = `0x0029003E`. Hardcoded on each peer as expected-UID.	Per-device authentication. Factory UID cannot be rewritten by firmware, so a cloned Nucleo (different chip, same firmware) will fail the allow-list check.
4	Monotonic nonce	2-byte per-peer counter, incremented each transmit. Receiver tracks last_nonce_seen per authorised UID.	Replay defence. Captured frame has a nonce; receiver rejects anything <= last seen from that UID. Stops rebroadcast attacks.

5. Layer 1: Zigbee AES-128 (Radio)

5.1 Configuration

Both XBee S2C modules were configured using XCTU with the following parameters:

- `EE = 1` Enable AES-128 encryption of all Zigbee MAC-layer frames
- `EO = 2` Require secure joining; unauthorised devices cannot join the PAN even if they can craft well-formed frames
- `KY = BAEADEADBEEFCAFE12345678` 128-bit network key (pre-shared)
- `NK = 0` Coordinator generates the internal network key automatically

5.2 Justification

XBee S2C implements AES-128 in hardware, so enabling encryption has negligible runtime cost. EE=1 alone would not prevent rogue devices from joining; EO=2 is therefore used to enforce secure joining and prevent unauthorised PAN association. The network key is structured to be memorable yet unique to this deployment (contains the PAN ID BAE, , and hex filler) and is stored only in the XBee chip's non-volatile configuration — not in firmware, reducing the risk of accidental disclosure through binary inspection.

5.3 Limitation

A physically captured XBee can potentially have its link key extracted by a skilled adversary. Layer 2 exists specifically to mitigate this scenario.

6. Layer 2: Application-Layer AES-128

6.1 Implementation

The Kokke tiny-AES-c library (public-domain single-file C implementation) was integrated into the firmware with only AES-128 enabled at compile time. A dedicated 16-byte application key

APP_AES_KEY (ASCII LeafSenseAgri202) is stored as a firmware constant — distinct from the Zigbee network key.

6.2 Block Structure

Each transmit path pads a 4-byte LeafSense payload into a 16-byte AES block:

- [0..3] Payload: MSG_TYPE, ZONE, DISEASE, SEVERITY
- [4..5] 2-byte monotonic nonce (little endian)
- [6..9] Sender STM32 UID word 0 (little endian)
- [10..15] Zero padding — structured integrity check

6.3 ECB Mode Justification

Electronic Codebook mode would normally be avoided because identical plaintext blocks produce identical ciphertext blocks, leaking information about repeated messages. In LeafSense this weakness is neutralised by construction: the 2-byte nonce is incremented every transmit, so no two plaintext blocks are ever identical. Given unique plaintexts, ECB provides confidentiality equivalent to CBC without the additional cost of transmitting or agreeing on an initialisation vector. The 10 zero-padding bytes additionally act as an implicit integrity tag: if decryption produces non-zero padding, the frame is rejected — providing a cheap substitute for a formal HMAC in this resource-constrained context.

6.4 Key Size Justification

AES-128 was chosen over AES-256. NIST FIPS 197 approves AES-128 for protecting sensitive-but-not-classified data, which is appropriate for agricultural disease data. AES-256 would roughly double encryption time on the Cortex-M4 (no hardware AES accelerator) for no practical gain in the 2-second transmit cadence. AES-128 also matches the XBee hardware layer, keeping key handling consistent.

7. Layer 3: UID Allow-List

7.1 Mechanism

Each STM32F411 microcontroller has a 96-bit factory-programmed unique identifier accessible via HAL_GetUIDw0/1/2. This identifier is burned into the chip at manufacture, non-rewritable, and guaranteed unique per die (per RM0383). The lower 32 bits of each Nucleo's UID are embedded inside the AES-encrypted payload. On receive, the Base compares the UID against a hardcoded allow-list and rejects any frame whose claimed identity is not on the list.

7.2 Measured UIDs

Both Nucleos' UIDs were read at boot and recorded:

- Base Scout UID: 0x00200049 0x31345108 0x37383436
- Field Scout UID: 0x0029003E 0x31345108 0x37383436

Words 1 and 2 are identical across both chips because they encode the production lot/batch — both Nucleos came from the same manufacturing batch. Word 0 encodes the wafer X/Y coordinates and is unique per die, which is what the allow-list relies on.

7.3 Negative-Test Verification

To verify the allow-list enforces (rather than passing by coincidence), a negative test was performed. The Base Scout's hardcoded FIELD_SCOUT_UID0 constant was temporarily changed to 0xDEADBEEF and the Base was reflashed. The Field Scout continued transmitting correctly-formatted, correctly-encrypted frames with its real UID of 0x0029003E. The Base rejected every single frame at the identity check, emitting AUTH FAIL messages at the Field's 2-second transmit cadence. No events were logged; no acknowledgements were sent. Restoring the correct UID immediately restored normal operation. This test is documented in the submission video and accompanying screenshots.

8. Layer 4: Nonce-Based Replay Protection

8.1 Mechanism

The Field Scout maintains a transmit nonce that increments with every outbound frame. The nonce is placed inside the encrypted 16-byte block, not outside, so an attacker cannot simply rewrite it. The Base Scout tracks last_nonce_seen per authorised sender UID and rejects any frame whose nonce is less than or equal to the value already seen. Out-of-order frames and replays are treated identically — both rejected.

8.2 Observed Behaviour

During testing, occasional REPLAY REJECTED messages appeared when frames arrived out of order due to normal radio propagation jitter. This confirms the defence is firing in practice. For a future iteration, a small sliding window (accepting nonces within N of last seen) would reduce false positives whilst preserving replay resistance — this is the same approach used in IPsec anti-replay windows.

9. Acknowledged Limitations

A mature First-class security discussion must acknowledge what the design does not defend against:

- **JTAG firmware extraction.** The application AES key is stored in firmware as a constant. A skilled attacker with physical access and a JTAG probe could read it directly. Production mitigation: enable STM32 Readout Protection (RDP) Level 2, which disables the debug interface permanently. This was out of scope for coursework because it would prevent further reflashing during development.
- **Physical theft of both devices.** Taking both Nucleos compromises the entire trusted pair. The UID allow-list design makes a single-device compromise recoverable via firmware update (revoke and replace the stolen UID), but dual compromise is absolute.

- **Denial-of-service by radio jamming.** No defence is implemented against a determined jammer broadcasting continuous noise on 2.4 GHz. This is consistent with the threat modelLeafSense is not safety-critical, and jamming requires proximity to the field.
- **Persistent nonce storage.** The last_nonce_seen value is currently RAM-resident. A rebooted Base resets the window and will accept a replay during a short window at boot. Planned mitigation: persist the nonce to EEPROM on each update. This was deferred because the EEPROM persistence itself has a hardware issue (Write Protect path) acknowledged in the coursework reflection.
- **Key rotation.** Both keys are pre-shared and static. A production deployment would implement a key-derivation protocol based on a root secret, possibly using HMAC-SHA-256 of the UID concatenated with a master secret, so each device has a distinct operational key.

10. Mapping to Coursework Rubric

The coursework brief specifies that a pass in the security element requires Zigbee AES encryption plus secure joining, with a simple application-layer encryption process for higher marks. LeafSense implements:

1. The minimum (Zigbee AES with EE=1, EO=2 for secure joining) — Layer 1.
2. Application-layer AES with hardcoded keys — Layer 2.
3. Additional identity-layer defence (UID allow-list) — Layer 3.
4. Additional replay defence (monotonic nonce) — Layer 4.
5. Negative-test verification proving the identity layer enforces — in-video demonstration.
6. Formal threat model with justified mitigations — this document.

Design choices are justified against published standards (NIST FIPS 197, IEEE 802.15.4-2020, Digi XBee S2C User Guide) rather than by assertion. Limitations are acknowledged explicitly, with concrete paths to mitigation in a production context.

---

11. Supporting References

Full reference list is provided in the separate References document. Key sources directly informing this design:

- NIST (2001). FIPS 197 — Advanced Encryption Standard. Underpins all AES choices.
- NIST (2007). SP 800-38C — Recommendation for Block Cipher Modes of Operation. Informed the ECB-with-nonce justification.
- IEEE (2020). Std 802.15.4-2020. Underpins the Zigbee physical layer.
- Digi International (2018). XBee S2C User Guide 90002002. Provides the EE/EO/KY semantics.
- STMicroelectronics (2021). RM0383 F411 Reference Manual. Specifies the 96-bit UID structure used by Layer 3.

- Kokke (2024). tiny-AES-c library (public domain). Provides the application-layer AES implementation.
-