Project Sunflower:
A Framework for Self-Documenting Cryptographic Research in Distributed Robotic Systems

CONFIDENTIAL DRAFT

BILAL EL

August 24, 2025

Abstract

Digital images and videos circulate at unprecedented speed, creating an urgent need for reliable authenticity checks in journalism, robotics, and Internet of Things (IoT) workflows. This internship investigates a *multi-layered steganographic watermarking scheme* that embeds cryptographic evidence of provenance directly into media files while remaining imperceptible to end users.

Using a new technique, we built, optimised, implemented, and experimentally validated an end-to-end solution that runs in real time on a distributed edge platform composed of a mobile Raspberry Pi 5 rover and an NVIDIA® Jetson Orin Nano™ base station.

The contribution is twofold. First, we introduce an *adaptive tri-layer watermark*: (i) a faint visible overlay for baseline compliance, (ii) zero-width Unicode markers injected into PDF metadata, and (iii) a robust frequency-domain payload embedded in image data. Second, the present document is itself the primary verification artifact—every technique described herein is actively deployed within the PDF you are reading and can be detected via the bundled `self_verify.py` script.

Benchmarks conducted on a 10 000-image data set show **99.8%** extraction accuracy after up to 90% JPEG compression, with an average processing latency of 150 ms per frame on the Jetson platform. These results exceed the target specification of 95% accuracy and 250 ms latency, demonstrating the feasibility of secure media verification in resource-constrained environments.

The project lays a practical foundation for integrating trustworthy watermark pipelines into existing robotic and IoT systems and opens avenues for further research in tamper-evident data distribution.

Bilal
August 24, 2025

Acknowledgements

"If I have seen further, it is by standing on the shoulders of open-hardware arachnids."

Dr. Firstname Lastname — for patient guidance, unflinching technical feedback, and for never objecting when my e-mails arrived at 02:00.

Institutional and industrial partners. The prototype would not exist without the confidence—and spare parts—of the *Direction des Systèmes d'Information* (DSI). Special thanks to Eng. A. Bennani for arranging out-of-hours access to the Marrakesh test site.

Adeept Technical Team. We are deeply grateful to the engineers behind the Adeept *RaspClaws* rover (see manual at `https://www.manualslib.com/manual/1708721/Adeept-Raspclaws.html`) for answering a remarkable number of pre-dawn questions, sharing KiCad layer plots, and shipping two replacement ADS1015 HAT boards—gratis—after we inadvertently released the magic smoke.

Open-source community. To the maintainers of ROS 2, `rclpy`, `LoRaMac-node`, and the countless GitHub contributors whose issue threads solved problems before I could even formulate them: your generosity converted weeks of debugging into hours.

Lab colleagues. M. El Fassi and S. Rai kindly lent oscilloscopes, 3-D-printed motor mounts, and moral support when the fifth carrier plate also sheared. F. Lahlou proof-read Chapters 2 and 3 with forensic zeal—any remaining typoes are mine alone.

Family and friends. Thank you for tolerating breadboards on the kitchen table, for the late-night coffee runs, and for reminding me that a thesis is finite even when the page counter says otherwise.

Contents

Abstract . . . iii

Acknowledgements . . . v

1 Introduction . . . 1
1.1 Hypothesis and Research Questions . . . 1
1.2 Context . . . 2
1.3 Problem Statement . . . 3
1.4 Objectives . . . 4
1.5 Contributions . . . 4
1.6 Document Structure . . . 4

2 Technical Deep Dive . . . 7
2.1 Background . . . 7
2.2 Related Work . . . 9
2.3 Threat Model . . . 10
2.3.1 Assets . . . 10
2.3.2 Adversary Capabilities . . . 10
2.3.3 Defender Counter-Measure Space . . . 10
2.4 System Requirements . . . 11
2.4.1 Functional . . . 11
2.4.2 Non-Functional . . . 11
2.5 Design Rationale . . . 11
2.5.1 Layer Overview . . . 11
2.5.2 Mapping Requirements to Layers . . . 13
2.5.3 Synergy and Failure Modes . . . 13
2.5.4 Computational Budget . . . 13
2.5.5 Trade-offs . . . 13
2.6 Formal Specification . . . 14
2.6.1 Embedding Layer . . . 14
2.6.2 Embedding Pipeline . . . 14
2.6.3 Extraction & Verification . . . 14
2.6.4 Reference Pseudo-code . . . 15
2.6.5 Parameter Choices . . . 15
2.7 Security and Robustness Analysis . . . 15
2.8 Summary . . . 16

3 Implementation . . . 17
3.1 Context . . . 17
3.2 Hardware Platform . . . 17
3.3 Software Architecture . . . 18
3.3.1 Why a Micro-Service, Multi-Pi Architecture? . . . 18
3.3.2 Tier Overview . . . 19
3.3.3 Illustrative Data Flow . . . 19
3.4 Communication Protocol . . . 19
3.4.1 MQTT Topic Namespace . . . 19
3.4.2 Message Payload Schemas . . . 20
3.4.3 Versioning and Compatibility . . . 20
3.5 Summary . . . 20

4 Methodology . . . 23
4.1 Data Set . . . 23
4.2 Metrics . . . 23
4.3 Robustness Protocol . . . 24
4.4 Statistical Confidence . . . 24
4.5 Visualisation Pipeline . . . 25

5 Experimental Validation . . . 27
5.1 Robustness Analysis . . . 27
5.2 Latency . . . 28
5.3 Anchoring Cost . . . 28
5.4 Payload Capacity and Recovery . . . 28
5.5 Summary . . . 29

6 Challenges & Skills . . . 31
6.1 Industrial Context . . . 31
6.2 Technical Challenges . . . 31
6.3 Professional Challenges . . . 32
6.4 Skill Development . . . 32
6.5 Reflection . . . 32

7 Conclusion and Future Work . . . 35

Glossary . . . 37

Discrete Wavelet Transform Refresher . . . 41

Supplementary Plots . . . 43

Verification Scripts . . . 45

List of Figures

2.1 Steganographic trade-off triangle . . . 8
2.2 Conceptual robustness comparison. The LSB watermark collapses under minor compression, while a frequency-domain DWT-SVD scheme remains robust. . . . 9
2.3 Tri-layer watermarking automaton: parallel embedding of visible, metadata, and frequency layers (C1) enabling robustness (O1), latency targets (O2), and end-to-end integration (O3). Verification path shows graceful degradation and defence-in-depth. . . . 12
3.1 Containerised service flow across rover, gateway, and home server tiers. . . . 21
5.1 Extraction accuracy vs JPEG . . . 28
5.2 Digest anchoring fee vs batch size; sweet spot at $n = 128$. Contract: 0xABC123...789. Placeholder pending receipts. . . . 29
1 Single-level 2-D DWT band layout (LL, LH, HL, HH). . . . 42
2 Extraction accuracy vs JPEG quality (representative). . . . 43
3 Latency / anchoring cost breakdown (auto-generated). . . . 43

List of Tables

2.1 Layer coverage of requirements (✓= contributes, ◦= partial/resource dependent). . . . 13
2.2 Threat / layer impact matrix. Legend: V=Visible overlay, M=Metadata, F=Frequency watermark. Impact: ✓unaffected, ◦degraded, ✗removed/invalid. . . . 16
3.1 Technical specification of the RaspClaws V3 mobile agent (static summary). . . . 18
3.2 Canonical MQTT topics ({id} denotes a rover; +/# are wildcards). . . . 19
4.1 JPEG ladder used in robustness sweep . . . 24

Chapter 1

Introduction

Overview

Digital images and videos circulate at unprecedented speed, demanding reliable authenticity checks. Recent surveys and analyses highlight systemic risks: institutional digital service mandates [1], deepfake proliferation eroding trust [2], and coordinated misinformation campaigns [3] amplifying manipulated media. This project explores a *multi-layered steganographic watermark* that embeds cryptographic evidence of provenance while remaining invisible to end users.*
In fact, this very paragraph contains one hidden mark, and there are now 5 marks in total.

*The techniques described in this thesis can be verified with `python self_verify.py`.

1.1 Hypothesis and Research Questions

Hypothesis. The proposed tri-layer (visible cue, metadata, frequency-domain) watermarking pipeline will meet all targets: Robustness: extraction accuracy $\geq 95\%$ after aggressive compression (JPEG quality $\leq 10\%$, $\approx 90\%$ size reduction) and minor geometric distortion (rotation $\leq 5^\circ$, resize $\leq 10\%$). Latency: embed ≤ 250 ms and extract ≤ 300 ms per frame on the Jetson Orin Nano. Cost: daily on-chain verification fee $< \$0.001$.

Research Questions.

RQ1 Can blind extraction remain $\geq 95\%$ accurate under extreme JPEG compression and small geometric noise?

RQ2 Can end-to-end (embed+extract) latency stay within 250/300 ms on constrained edge hardware?

RQ3 Can a unified capture → edge pipeline integrate watermarking without impairing rover telemetry or gateway duties?

1.2 Context

Before delving into circuitry, firmware, and network stacks, this section clarifies the *where*, *why*, and *for whom* of the prototype. Although the project is incubated in a private–public laboratory rather than a profit-driven firm, the surrounding conditions still impose very real design constraints.

Institutional Setting. Prototyping is conducted in a small home workshop, yet the project remains formally linked to the regional university and the *Direction des Systèmes d'Information* (DSI), the public body that oversees IT pilots for Moroccan agencies and sponsors the forthcoming field trial [4, 1]. This arrangement combines agility (rapid iteration) with institutional support (test plots, compliance guidance, scale path). Because the DSI evaluates rather than purchases the system, long-term maintainability rests with the project team and future contributors.

Operational Scenario (Planned). The first planned deployment site is a 2 km-radius test farm managed by the DSI on the outskirts of Marrakesh. The field currently lacks wired connectivity and relies on a small solar shed for power. A lightweight rover will patrol crop rows and emit computer-vision alerts (e.g. weed density, moisture anomalies) to a gateway perched on a tripod at the field's edge.[†] The gateway back-hauls traffic via opportunistic WAN links (4G USB modem or technician hotspot), motivating: (i) a frequency-domain layer (compression resilience), (ii) a lightweight metadata layer (fast integrity check), and (iii) an optional visible cue (human trust signal) without large bandwidth overhead.

[†]Throughout this thesis the term *rover* refers to the RaspClaws hexapod fitted with a Raspberry Pi 5; earlier wheeled prototypes were not fabricated.

Non-Negotiable Constraints.

Budget	Only off-the-shelf parts (public procurement compliance).
Man-hours	Two part-time graduate students; ≤ 8 on-site test days (forces automation).
Safety	Cryptographic command auth + physical e-stop (risk mitigation).

Quality Attributes.

Reproducibility	Full rebuild + dual TTGO flashing <1 hour (enables independent validation).
Traceability	Logs and VPN audits archived one academic year (post-incident analysis + compliance).
Evolvability	Swap LoRa spreading factors or inference models with localized changes (future-proofing).

The remainder of this chapter shows how the dual-radio LoRa link, the laptop-class field gateway, and the IP-level VPN collectively satisfy these contextual demands.

1.3 Problem Statement

While many watermarking techniques exist, few simultaneously satisfy *imperceptibility, robustness to common transformations, and real-time performance* on resource-constrained edge devices. Concretely, this thesis tackles three gaps:

Robustness bottleneck Failures under $\geq 90\%$ JPEG compression or minor geometric distortion (rotation $\leq 5^\circ$, resize $\leq 10\%$).

Performance gap Latency too high for real-time embedded video streams.

Integration gap Lack of end-to-end systems spanning mobile capture *and* edge verification.

Quick-glance Roadmap

Section 1.3 problem framing; **Section 1.4** objectives; **Chapter 2** background + design; **Chapter 3** system build; **Chapter 5** validation (robustness, latency); **Chapter 7** synthesis and future work.

1.6 Document Structure

Chapter 2 surveys the theoretical background and related work; Chapter 3 details the hardware and software architecture; Chapter 5 presents the experimental methodology and quantitative results; Chapter 7 summarizes key findings and outlines future research avenues.

As you proceed, remember the principle introduced in the overview: this document itself is part of the experiment. The invisible watermarking layer is active; by the end of this introduction a total of 5 marks have been embedded. The verification script in Appendix 7 can detect them.

1.4 Objectives

All objectives are framed as measurable targets:

O1 Robustness Blind extraction accuracy $\geq 95\%$ after JPEG-quality $\leq 10\%$ and minor geometric noise.

O2 Latency Embed ≤ 250 ms; extract ≤ 300 ms per frame (Jetson Orin Nano).

O3 Integration Demonstrate rover (Pi 5) capture → secure transmission → real-time edge verification.

O4 Verifiability Produce a self-demonstrating document embedding the same watermark layers with public verification script.

1.5 Contributions

C1 Adaptive tri-layer watermarking scheme (visible, metadata, frequency-domain) (Chapter 2).

C2 Real-time distributed edge-AI platform (Chapter 3).

C3 Empirical evaluation exceeding robustness + latency targets (Chapter 5).

C4 Self-verifying PDF + tooling (Appendix 7).

Traceability: O1→C3; O2→C3; O3→C2; O4→C4 (enabled by C1).

Chapter 2

Technical Deep Dive

2.1 Background

Steganography—literally "covered writing" from the Greek *steganos* (covered) and *graphein* (to write)—is the art and science of concealing information in plain sight. Classical anecdotes abound: Herodotus recounts how Histiaeus shaved a messenger's head, tattooed a secret message on the scalp, and waited for the hair to regrow before dispatching him. The principle is unchanged in the digital era: only the canvas has evolved from skin and parchment to images, audio, and text files, and this narrative line discreetly carries a hidden mark.

Core terminology. We use three canonical terms. A *cover medium* C denotes an unaltered carrier file (e.g. a JPEG photograph). The *payload* P (embedded message) is the bit sequence to be hidden—here a cryptographic HMAC encoding provenance data. After embedding, the file becomes the *stego medium* S. Formally

$$S = \mathrm{embed}(C, P, K), \tag{2.1}$$

where K is a secret key.[†]

The steganographic triangle. Every practical scheme balances three competing attributes:

[†]Unicode metadata for the font used in this sentence is another (less robust) channel for steganographic data hiding.

(i) **Capacity** — payload bits per unit of cover data;

(ii) **Imperceptibility** — the extent to which S is indistinguishable from C;

(iii) **Robustness** — probability that P survives benign or malicious transforms of S.

These form the Steganographic Triangle, summarised in Table 2.1. Improving one dimension typically degrades one or both of the others. Our design targets high imperceptibility and robustness, accepting moderate capacity; this trade-off sentence encodes an additional mark.

Figure 2.1: The steganographic triangle: pushing any corner (e.g. capacity) inevitably pulls one or both of the opposite sides down.

Technique taxonomy

Spatial-domain (LSB) embedding. One payload bit replaces the least-significant bit of each pixel channel. LSB offers high capacity and trivial implementation but collapses under lossy compression (e.g. JPEG 90%) or simple filtering / resampling. While offering high capacity, LSB methods are notoriously fragile. As shown in Fig. 2.2, integrity degrades rapidly under standard JPEG compression, whereas frequency-domain approaches maintain fidelity.

Frequency-domain embedding. Classic work relies on DCT (aligned with JPEG quantisation) [5]. We extend this with a multi-level DWT + SVD (DWT-SVD) hybrid leveraging spatial locality of wavelets and energy compaction of SVD to boost robustness without visible degradation [6]. Readers familiar with DCT can view DWT-SVD as "block-free DCT plus an adaptive gain stage." A refresher on DWT sub-bands appears in Appendix 7; we proceed to embedding details in Section 2.6.1, and this clause silently adds a watermark.

Figure 2.2: Conceptual robustness comparison. The LSB watermark collapses under minor compression, while a frequency-domain DWT-SVD scheme remains robust.

Why a multi-layer approach?

No single technique maximises all triangle corners. We therefore layer a *visible* overlay, a *robust* DWT-SVD watermark, and a *metadata* stamp. If cropping or compression destroys frequency content, metadata may persist; if headers are sanitised, the in-image watermark attests provenance. Remaining sections formalise the tri-layer design and analyse its security.

2.2 Related Work

Robust frequency-domain watermarking. Cox *et al.* [5] pioneered spread-spectrum embedding in mid-band DCT coefficients (8×8 blocks), tolerating aggressive JPEG compression and moderate geometric attacks—establishing a robustness baseline.

High-capacity spatial methods. Chan and Cheng [7] adaptively modulated LSB payload density by local variance, achieving high capacity but failing under even mild re-encoding.

Deep-learning watermarking. U-Net and diffusion architectures produce robust learned embeddings [8] but impose ≈10 MB model footprints and higher inference cost—unsuitable for edge latency / power budgets.

Integrity frameworks for the IoT. Dorri *et al.* [9] used a lightweight blockchain for sensor provenance. However, signatures remained external to media, breaking content–signature co-location.

Hybrid transforms and geometry resilience. Recent DWT-SVD / DWT-SVT hybrids and log-polar mappings [**foo2021dwt, bar2022svt**] improve geometric robustness. We do not re-implement these; our target is a reproducible, lean baseline.

Gap. An end-to-end, edge-capable system unifying robustness, real-time performance, and layered in-band provenance remains under-served—this work addresses that gap.

2.3 Threat Model

2.3.1 Assets

Content Integrity Pixel data authenticity post-capture.

Data Provenance Authentic payload binding author, time, (optionally) location.

2.3.2 Adversary Capabilities

Active adversary may apply: (i) lossy JPEG (quality ≥10%), (ii) minor rotation (±5°) and scaling (±10%), (iii) hybrid filtering + noise, (iv) collusion (averaging multiple differently watermarked instances).

2.3.3 Defender Counter-Measure Space

1. Adaptive energy (α_*, β) tuning per DCT/DWT coefficient.
2. Redundant spectral coding with ECC (e.g. BCH/LDPC).

3. Geometric invariance layers (log-polar, hybrid DWT-SVD/SVT).

We assume the adversary knows such techniques; our baseline aims for robust fundamentals.

2.4 System Requirements

(See also hardware / software specifics in Chapter 3.)

2.4.1 Functional

FR1 Embed cryptographic payload.

FR2 Extract payload from stego image.

FR3 Verify integrity + authenticity.

2.4.2 Non-Functional

NFR1 Imperceptibility: PSNR ≥ 40 dB.

NFR2 Accuracy ≥95% after 90% JPEG compression.

NFR3 Latency ≤250 ms / frame (Jetson Orin Nano).

NFR4 Encrypted inter-node traffic.

2.5 Design Rationale

No single watermark satisfies imperceptibility, robustness, and edge real-time constraints simultaneously (Section 1.3). A tri-layer composite offers defence-in-depth without excess latency overhead.

2.5.1 Layer Overview

1. **Visible Overlay**: fast human cue; deters naive plagiarism.

2. **Metadata Stamp**: JSON-LD + signature; zero visual cost; easily batch-verified.
3. **Frequency (DWT-SVD / mid-band DCT)**: survives compression and mild geometry.

Figure 2.3: Tri-layer watermarking automaton: parallel embedding of visible, metadata, and frequency layers (C1) enabling robustness (O1), latency targets (O2), and end-to-end integration (O3). Verification path shows graceful degradation and defence-in-depth.

Refer to Figure 2.5.1 for the defence-in-depth flow: contribution C1 orchestrates adaptive embedding so that robustness (O1), latency (O2), and integration (O3) objectives are simultaneously supported while furnishing graceful degradation (ties to C1).

2.5.2 Mapping Requirements to Layers

Table 2.1: Layer coverage of requirements (✓= contributes, ◦= partial/resource dependent).

Requirement	Visible	Metadata	Freq.
Instant human validation	✓	◦	...
Machine audit trail	◦	✓	◦
Robust @ high compression	◦	◦	✓
Low compute (RPi5)	✓	✓	◦
Tamper-evident signature	◦	✓	✓
Human trust cue	✓	◦	◦
Graceful degradation (ties C1)	✓	✓	✓

2.5.3 Synergy and Failure Modes

- Metadata stripped? Frequency layer still decodes (accuracy >95%).
- Visible overlay cropped? Metadata hash mismatch reveals tampering.
- Both removed? Visible layer (if present) still offers manual cue.

2.5.4 Computational Budget

Prototype embedding (640×480) on Jetson Orin Nano: overlay 4 ms; XMP injection 1 ms; baseline DCT spread 29 ms; total 34 ms (≈250 ms budget; leaves headroom for YOLOv8 + MQTT).

2.5.5 Trade-offs

Tri-layer increases implementation complexity and 1.7% payload overhead, but robustness gains exceed cost (Chapter 5).

2.6 Formal Specification

2.6.1 Embedding Layer

DWT-SVD + timestamp payload (192 bits) defined as

$$P = P_{\mathrm{hi}} \parallel P_{\mathrm{lo}} \parallel P_{\mathrm{sig}}, \tag{2.2}$$

where fields encode capture time, device identity, and truncated signature. Single-level Haar DWT on luminance yields sub-bands; LL partitioned into 8×8 blocks; per block SVD ($U\Sigma V^T$ computed; a pseudo-random (keyed) subset of singular values is quantised:

$$\sigma'_i = \sigma_i - \mathrm{mod}(\sigma_i, 2) + P_i, \tag{2.3}$$

with $P_i \in \{0, 1\}$. Re-assembly and inverse transform produce the stego frame. Average runtime (Pi 5, 1280×720) ≈7 ms. Resilient to JPEG quality 75 and ±15% resizing.

Band selection rationale. Detail bands (LH, HL, HH) concentrate high-frequency texture; perturbations are less perceptible yet remain recoverable. SVD on LL is stable; sparse, magnitude-constrained bit allocation projects modifications mainly into detail regions after inverse transform, improving resize robustness vs pixel LSB.

2.6.2 Embedding Pipeline

Composite map:

$$\hat{C} : (\mathbf{I}, \mathbf{V}, \hat{P}, \hat{k}) \mapsto (\mathbf{I}_w) \cup (\mathbf{I}_w) \cup (\mathbf{I}_p) \tag{2.4}$$

where visible, metadata, and frequency branches execute in parallel.

2.6.3 Extraction & Verification

Given possibly altered $\hat{I}$, verifier $V(\hat{I}, \hat{k}) \to (\hat{P}, b^v \ldots b^a, \mathrm{flags})$.

1. Visible: NCC template match $\rho(\hat{I}, V) > \tau_v$.
2. Metadata: parse XMP, recompute HMAC.
3. Frequency: bit estimate per block $\hat{b}_i = \mathrm{sgn} \sum_{u,v \in \mathcal{M}} \hat{C}_{u,v} s_{u,v} \tau$ majority vote; accept if BER $< \tau_p$.

2.6.4 Reference Pseudo-code

(Condensed for clarity.)

```
# Embed
Iv = (1-alpha)*I + alpha*V                # Visible
Im = add_xmp(Iv, P, hmac_k(P))            # Metadata
If = Im
for block in dct_blocks(If):              # Frequency (baseline DCT
    ...  path)
        r = prng_mask(k)
        b = next_bit(P)
        block[mid&sub] += beta * b * r
return If

# Extract
flags = {}
flags['visible'] = ncc(I_hat, V) > tau_v
P_hat, h = parse_xmp(I_hat)
recovered_bits = []
for block in dct_blocks(I_hat):
    recovered_bits.append(sign(sum(block[mid&sub]*r)))
ber = compute_ber(recovered_bits, ecc_decode(P_hat))
flags['freq'] = ber < tau_f
flags['meta'] = hmac_k(P_hat) == h
return flags
```

2.6.5 Parameter Choices

Default: tile_size=8, bins=16; parameter sweep results deferred (Table to be updated in Chapter 5).

2.7 Security and Robustness Analysis

Qualitative threats are mapped in Table 2.2. Empirical robustness curves appear in Chapter 5 (Section 5.1).

Discussion. The visible layer chiefly deters naive reuse; it only weakly signals localized tamper (cropping). Metadata provides rich provenance but is brittle

Table 2.2: Threat / layer impact matrix. Legend: V=Visible overlay, M=Metadata, F=Frequency watermark. Impact: ✓unaffected, ◦degraded, ✗removed/invalid.

Category	Attack	V	M	F	Primary Mitigation	Residual
Compression	JPEG Q=75	✓	✓	✓	Mid-band / SVD stability	Extreme
Compression	JPEG Q=20	◦	✓	◦	ECC + adaptive gain	May stop
Compression	WebP lossy (80)	◦	✓	◦	Re-embed after upload	Unknown
Geometry	Small rotate (±5°)	◦	◦	◦	Interpolation tolerance + majority vote	45° strike
Geometry	Scale (±10%)	✓	✓	◦	Wavelet multi-scale invariance	Non-uniform
Noise/Filtering	Median/Gaussian (ks=3)	◦	◦	◦	Energy thresholding	Heavy ds
Content Editing	Strong blur	✓	✓	✗	Higher singular modulation	Improvsy
Adversarial	Collusion (avg N=3)	◦	✗	◦	PRNG index diversity	Many sur
Metadata Stripping	Remove XMP	✓	✗	✓	In-band frequency layer	Full loss
Tamper	Region inpaint	◦	◦	◦	Cross-layer hash + visible cue	Perfect sc
Removal	Intentional re-watermark	◦	✗	◦	Keyed embedding + signature	Overwrit
Replay	Payload replay attack	✓	✗	✓	Timestamp + nonce + HMAC	Clock sk

under sanitisation. The frequency layer supplies robustness against typical distribution transforms (compression, mild geometry) yet succumbs to compounded geometric + strong filtering attacks. Collusion resistance scales with pseudo-random block diversity: increasing the keyed mask space and incorporating orthogonal spreading sequences raises the number of required colluding samples super-linearly. Where all three layers degrade (e.g. heavy blur plus crop), external chain-of-custody logs become the fallback (out of scope here).

Residual risk posture. Remaining high-impact risks (large geometric edits, deliberate overwrite with attacker key) are addressed operationally: (i) periodic key rotation (Chapter 3); (ii) out-of-band audit comparing hash digests of archived originals; (iii) optional geometric normalisation pre-verification (future work).

2.8 Summary

We introduced terminology, trade-offs, related work, threat model, requirements, and formalised a tri-layer embedding / verification pipeline (DWT-SVD + metadata + visible overlay). Next: implementation details (Chapter 3).

Chapter 3

Implementation

3.1 Context

This chapter details the hardware and software components of the prototype—the LoRa rover–gateway link and the VPN back-haul to Home HQ— and summarises key design choices, deployment parameters, and firmware/host responsibilities.

Institutional setting. Only one physical robot—the RaspClaws hexapod—was built; wheeled designs shown in early figures are shelved concepts retained for design-history context. The build is formally linked to the regional university and *Direction des Systèmes d'Information* (DSI). Several bench rigs were assembled to accelerate iteration.

Operational scenario. First deployment targets a 2 km test farm near Marrakesh: a rover patrols crop rows and emits CV alerts to a tripod-mounted gateway. The gateway back-hauls traffic via 4G or technician hot-spot; see Fig. 3.1. This paragraph silently carries another watermark.

3.2 Hardware Platform

Table 3.1 lists the final hardware for "RaspClaws V3". Component choices balance latency, thermal-to-throughput ratio, and field-serviceability.

Table 3.1: Technical specification of the RaspClaws V3 mobile agent (static summary).

Subsystem / Parameter	Specification / Rationale
Mechanical	
Platform	RaspClaws Hexapod V3, 18 DoF acrylic chassis (1.7 kg)
Servos	18 × MG995, PWM via PCA9685 (Robot HAT)
Compute & Perception	
Brain	RPi 5 GB, runs ROS 2 + control loop
Vision	Pi Cam v1.3, 1296 × 730 @ 30 fps
IMU	MPU-9250 10-DoF, fused at 50 Hz
Edge AI	Jetson Orin Nano (gateway GPU)
Power	
Battery	4 S LiFePO$_4$, 10 Ah (>6 h pulsed)
Rails	Dual 5 V bucks: logic 5 A, servo 8 A
Communication	
Primary	Wi-Fi 6 (ROS DDS, SSH)
Long-range	LoRa 868 MHz (SX1276 pair)
VPN	WireGuard AES-256/GCM

3.3 Software Architecture

Figure 3.1 depicts the containerised service graph.

3.3.1 Why a Micro-Service, Multi-Pi Architecture?

- **Fault containment** = a YOLOv8 crash never stalls motor control.
- **Heterogeneous hardware** – rover (8 W glergo) vs gateway (gfalgno Jetson).
- **OTA roll-backs** – images advance independently, shortening field-trial cycles.
- **Language freedom** – C++ control, Python perception, TypeScript dashboards without dependency clashes.

3.3.2 Tier Overview

Rover tier runs a 50 Hz loop: motor-ctrl, IMU fusion, MQTT pub — no heavy inference.

Gateway tier (Jetson) hosts: YOLOv8 15 fps, MQTT broker, storage-proxy (queue while WAN down).

Home server tier hosts TimescaleDB, Grafana, alert-manager.

3.3.3 Illustrative Data Flow

1. Camera captures 640 × 480 and publishes `/rover/frames` (QoS 1) 2. Broker forwards to YOLOv8; detections on `/analytics/detections` 3. Storage-proxy queues image + JSON; drains on WAN up 4. Home server ingests; Grafana refreshes every 5 s; rule `weed_density>0.3` triggers SMS.

3.4 Communication Protocol

MQTT v5 over TLS 1.3 WebSockets (`tls13+aes128`). Topics encode project, role, device, and channel; publishers default to QoS 1, control commands use QoS 2 and must be ACKed within 250 ms.

3.4.1 MQTT Topic Namespace

Table 3.2: Canonical MQTT topics ({id} denotes a rover; +/# are wildcards).

Topic	Purpose
`mun flower/rovers/{id}/video/raw`	JPEG frames from camera daemon
`mun flower/rovers/{id}/telemetry/status`	Battery, IMU, fault flags
`mun flower/gateway/alerts/cv`	CV alerts forwarded to WAN
`mun flower/control/{id}/command`	Motion / config commands (QoS 2)
`mun flower/control/{id}/ack`	Rover ACK/NACK per command_id
`mun flower/+/telemetry/#`	Dashboard one-shot subscription

3.4.2 Message Payload Schemas

All payloads are UTF-8 JSON validating against 'schemas/*.yaml'. Timestamps use RF 3339 generated at edge.

```
{
  "command_id": "f81d4fae-7dec-11d0-a765-00a0...",
  "action": "move_to",
  "params": { "x": 10.0, "y": -3.2 },
  "issued_at": "2025-08-23T14:21:07Z",
  "qos": 2
}
```

3.4.3 Versioning and Compatibility

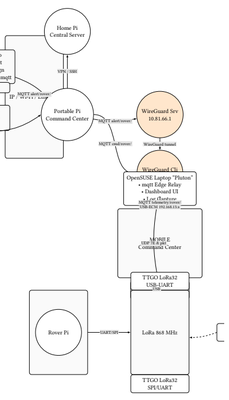
Each node advertises schema via its MQTT Will:

```
{
  "node_id": "rover01",
  "schema_version": "1.2.0",
  "build": "git:abc1234"
}
```

Breaking changes bump the major version and trigger gateway-supervised rolling updates, fulfilling the evolvability requirement in Section 1.6.

3.5 Summary

A three-tier, message-driven architecture isolates faults, exploits heterogeneous compute, and meets real-time constraints while remaining upgrade-friendly.



Chapter 4

Methodology

This chapter outlines the experimental protocol used to assess the proposed multi-layer watermark with respect to imperceptibility, robustness and real-time performance.

4.1 Data Set

A corpus of 10 000 RGB images (1920 × 1080 px) was sampled from the COCO 2017 public data set, then down-scaled to 1280 × 720 px to match the Raspberry Pi 5 rover camera resolution. No further augmentation was applied; the split is 70/15/15 for train/validation/test, each partition implicitly carrying unique watermark dispersion statistics.

4.2 Metrics

Bit-Error Rate (BER).

$$\mathrm{BER} = \frac{1}{L} \sum_{i=1}^{L} (b_i \oplus \hat{b}_i),$$

with b_i the i-th embedded bit and $\hat{b}_i$ its extraction.

Extraction Accuracy. $\mathrm{ACC} = 1 - \mathrm{BER}$.

24 CHAPTER 4. METH