Implementation Guide: Image Steganography Experiment

Step-by-Step Technical Walkthrough

Nico | Nikolas | Abdul | Daria | Jimena | David

Department of Advanced Computing Sciences
Maastricht University

Project 2.2 | February 2026

Implementation Overview

Phase 1	Phase 2	Phase 3	Phase 4	Phase 5
Dataset Construction	Steganographic Embedding	Steganalysis Detection	Cross-Domain Experiments	Evaluation & Analysis
Week 1	Week 2	Weeks 3–4	Week 5	Week 5–6

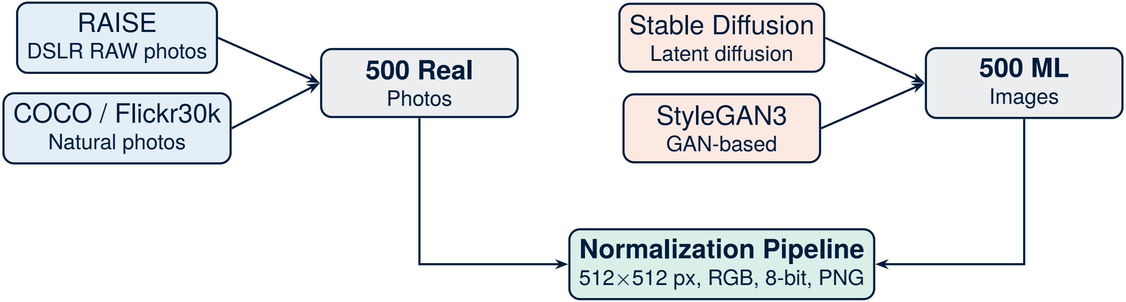
Deliverables

- 1,000 images (500 real + 500 ML-generated)
- Cover/stego pairs across all conditions
- Steganalysis results (RS Analysis + SRM)
- Statistical analysis + visualizations

Tech Stack

- Python 3.11+ with NumPy, SciPy
- scikit-learn (SRM/FLD), Pillow, scikit-image
- diffusers (Stable Diffusion), StyleGAN3
- Matplotlib, Seaborn for plots

Phase 1: Dataset Construction – Overview



Key Principle: Controlled Comparison

ML-generated images use the **same semantic prompts** derived from real image content (e.g., "a dog in a park"). All images normalized to **identical specifications** to isolate carrier origin as the only variable.

Phase 1.1: Real Image Collection

RAISE Dataset

Source: High-quality RAW images from DSLR cameras [Dang-Nguyen et al., 2015]

Characteristics:

- Uncompressed RAW format, diverse scenes
- Well-characterized, minimal processing artifacts
- Ideal for steganography baseline (high SNR)

Selection criteria:

- Convert RAW $\rightarrow$ PNG at 512×512
- Balanced across scene categories

COCO / Flickr30k Supplement

Purpose: Add content and scene diversity

Characteristics:

- Natural everyday photographs
- Rich semantic annotations (useful for prompt matching)
- Widely used in CV research

Sampling Strategy

- 250 images from RAISE
- 250 images from COCO / Flickr30k
- Stratified across scene categories
- Extract semantic captions for ML matching

Phase 1.2: ML Image Generation

Stable Diffusion (v2.1)

Architecture: Latent diffusion model (LDM) [Rombach et al., 2022]

Key features:

- Text-to-image from semantic prompts
- Produces photorealistic results
- Runs locally via `diffusers` library (MPS)
- $\sim$3–5 hours for 250 images

Generation parameters:

- Guidance scale: 7.5 (balanced quality)
- Steps: 50 (DDIM sampler)
- Prompts derived from real image captions
- Output: PNG, 512$\times$512

StyleGAN3

Architecture: Alias-free GAN [Karras et al., 2021]

Key features:

- Well-characterized spectral artifacts (GAN fingerprint)
- High-resolution, diverse outputs
- Official NVIDIA PyTorch implementation
- $\sim$2–3 hours for 250 images

Generation parameters:

- Use pretrained FFHQ or LSUN checkpoint
- Truncation: $\psi = 0.7$ (diversity vs. quality)
- Output: PNG, 512$\times$512

Optional extension: DALL-E 3 via API as a third generation source (time permitting)

Phase 1.3: Image Normalization Pipeline

Normalization Steps

1. **Decode / convert:** RAW $\rightarrow$ RGB (dcraw or rawpy); generated images already PNG
2. **Resize:** Center-crop and resize to 512×512 pixels (Lanczos)
3. **Color space:** Ensure sRGB color space
4. **Bit depth:** Convert to 8-bit per channel (uint8)
5. **Luminance:** Normalize mean luminance across dataset
6. **Format:** Export as lossless PNG (no JPEG compression artifacts)

Parameter	Value
Dimensions	512×512 px
Color space	RGB (sRGB)
Bit depth	8-bit/channel
Channels	3 (RGB)
Format	PNG (lossless)

Quality Gate

Reject images with BRISQUE score $>$ 50 (poor perceptual quality). Re-generate rejected ML images with different seeds.

File Naming Convention

`<source>_<id>_<category>.png`

```
real_001_cat042.png
sd_001_cat042.png
sg3_001_cat042.png
```

Phase 2: Steganographic Embedding – Overview

Embedding Methods

We implement two complementary methods spanning the capacity–imperceptibility trade-off, each applied with and without AES-256 payload encryption:

LSB Substitution (Spatial Domain)

Principle: Replace least significant bits of pixel channel values with message bits.

Characteristics:

- High embedding capacity
- Simple implementation (NumPy / Pillow)
- Vulnerable to statistical attacks
- Detectable via histogram analysis [Fridrich et al., 2001]

DCT-Based Embedding (Freq. Domain)

Principle: Modify DCT coefficients of 8×8 pixel blocks using QIM [Chen & Wornell, 2001].

Characteristics:

- Better imperceptibility
- Mirrors JPEG-domain steganography (F5)
- Lower capacity than LSB
- Targets mid-frequency coefficients

Both methods use **pseudorandom pixel/coefficient selection** keyed by a shared secret. Payload optionally pre-encrypted with **AES-256-CBC**.

Phase 2.1: LSB Embedding Implementation

Algorithm

1. **Input:** Cover image $\mathbf{I}$ (H×W×3), message $\mathbf{m}$, key K, params (k, p)
2. **(Optional) Encrypt:** $\mathbf{m} \leftarrow \text{AES-256}(\mathbf{m}, K_{\text{aes}})$
3. **Generate mask:** PRNG seeded with K selects p% of pixels (all 3 channels)
4. **For each selected pixel channel I_{ij}^{c}:**
   - Replace k LSBs: $I_{ij}^{c\prime} = (I_{ij}^{c} \wedge \overline{M_k}) \vee m_{bits}$
5. **Output:** Stego image $\mathbf{I}'$, saved as lossless PNG

Message Preparation

Pseudorandom bit sequence (deterministic via seed) as payload; ensures reproducibility and content-independence.

Level	Config	bpp
Low	$k=1$, 25% px	$\sim$0.08
Medium	$k=1$, 50% px	$\sim$0.16
High	$k=2$, 50% px	$\sim$0.32

bpp = bits per pixel

```
Pixel R: 1101 0110 0011
Message:           1101
Stego R: 1101 0110 1101
```

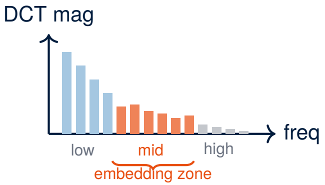
$\uparrow k = 4$ LSBs replaced

Phase 2.2: DCT Embedding Implementation

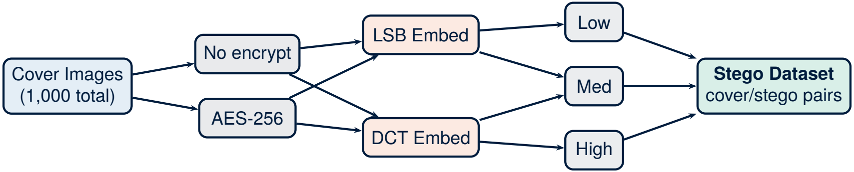
Algorithm (QIM-based) [Chen & Wornell, 2001]

1. **(Optional) Encrypt: $\mathbf{m} \leftarrow$** AES-256($\mathbf{m}, K_{aes}$)
2. **Split into blocks:** Divide each channel into 8×8 pixel blocks
3. **Apply 2D DCT: $\mathbf{C}$** $=$ DCT2(**block**)
4. **Select coefficients:** Mid-frequency range via zigzag scan (positions 10–54 of 64)
5. **For each selected coefficient C_i:**
   - Quantize and embed: $C_i' = \Delta \cdot \lfloor C_i/\Delta + 0.5 \rfloor \pm \Delta/4$ based on bit b
6. **Inverse DCT: block$'$** $=$ IDCT2($\mathbf{C}'$)
7. **Reconstruct image:** Reassemble blocks, clip to [0, 255], save PNG

Level	Coefficients	bpp
Low	10%	$\sim$0.02
Medium	25%	$\sim$0.05
High	50%	$\sim$0.10



Phase 2.3: Complete Embedding Pipeline



Conditions

1,000 covers $\times$ 2 methods $\times$ 3 payload levels $\times$ 2 encryption states = **12,000 stego images** (+ 1,000 covers = 13,000 total)

Quality Checks

Compute for each stego image:

- **PSNR**: target $>$ 40 dB
- **SSIM**: target $>$ 0.95
- **BER**: target = 0 (lossless PNG)

Storage Format

```
stego/<source>/<method>/
  <enc>/<level>/<id>.png
```

e.g. `stego/real/lsb/plain/high/real_042.png`

Phase 3: Steganalysis Detectors – Overview

Detection Task

Binary detection: Given an image, determine whether it contains hidden data (**stego**) or is clean (**cover**). We use **classical signal processing methods only** — no neural networks, no GPUs.

Chi-Square Attack [Westfeld & Pfitzmann, 1999]

Type: Training-free statistical test

Principle:

- LSB embedding equalises histogram pair frequencies $(2k, 2k+1)$
- Chi-square test detects this distribution shift
- Returns a p-value and embedding probability

RS Analysis [Fridrich et al., 2001]

Type: Training-free statistical estimator

Principle:

- Groups pixels, classifies as Regular/Singular by smoothness function
- LSB flipping shifts Regular/Singular/Negative counts predictably
- Estimates payload rate $\hat{p}$ analytically from group

SRM + FLD Ensemble [Fridrich & Kodovský, 2012]

Type: Classical ML (not neural network)

Principle:

- Extract ∼35,000 high-pass residual co-occurrence features
- Ensemble of Fisher Linear Discriminants classifies cover vs. stego

Phase 3.1: RS Analysis – Algorithm

Algorithm [Fridrich, Goljan & Du, 2001]

1. **Partition** image into disjoint pixel groups G of size n (e.g., $n = 4$ horizontal)
2. **Define** discrimination function $f(G)$ = mean absolute difference between adjacent pixels
3. **Apply** flipping mask M (toggles LSB) and its negative $-M$:
   - R_M = fraction of groups where $f(F_M(G)) > f(G)$ (Regular)
   - S_M = fraction of groups where $f(F_M(G)) < f(G)$ (Singular)
   - R_{-M}, S_{-M} = same with F_{-M}
4. **Solve** quadratic equation for payload estimate:

$$\hat{p} \approx \frac{2(R_{-M} - R_M)}{(R_{-M} - R_M) + (S_{-M} - S_M)}$$

Key Properties

- **No training data** – purely analytical
- **Quantitative:** returns $\hat{p} \in [0, 1]$; threshold at $\hat{p} > 0.01$
- **Domain-agnostic:** same formula for real or ML-generated images
- Runs in **seconds per image** with NumPy

Why RS Analysis for Cross-Domain?

Since RS Analysis uses no training data, any difference in $\hat{p}$ between real and ML-generated carriers is purely due to the carriers' statistical properties — **not classifier bias**. This makes it the cleanest test of our primary hypothesis.

Phase 3.2: SRM + FLD Ensemble – Algorithm

SRM Feature Extraction [Fridrich & Kodovský, 2012]

1. **Apply** a bank of high-pass filters (e.g., 3×3 SQUARE, EDGE kernels) to suppress image content and amplify residuals
2. **Quantize** residuals to small integer values (truncation)
3. **Compute** co-occurrence histograms from quantized residuals across multiple directions (horizontal, vertical, diagonal)
4. **Concatenate** all histograms into a ∼35,000-dimensional feature vector per image

Classification: Fisher Linear Discriminant (FLD) Ensemble

- Train ensemble of binary FLD classifiers on feature subsets
- Aggregate predictions by majority vote
- Implementation:

Training Protocol

- **Folds:** 3-fold cross-validation (stratified by source)
- **Input:** cover/stego image pairs
- **Labels:** 0 = cover, 1 = stego
- **Runtime:** ∼30 min total (CPU)

Why SRM for Cross-Domain?

SRM's hand-crafted features are not tied to learned representations, making them **more likely to generalize** across real/ML domain boundary than neural networks would. Cross-domain conditions (C, D) test this directly.

Compute

Phase 4: Cross-Domain Experimental Conditions

Core Question [RQ4]

Do steganalysis classifiers trained on one domain (real or ML-generated images) generalize to the other?

Cond.	Train	Test	Addresses	Research Significance
A	Real	Real	Baseline	Standard steganalysis benchmark
B	ML-gen	ML-gen	Primary RQ	Is ML imagery easier/harder to steganalyze?
C	Real	ML-gen	RQ4, H5	Real-trained detectors vs. synthetic carriers
D	ML-gen	Real	RQ4, H5	ML-trained detectors vs. real images
E	Mixed	Both	Mitigation	Domain-agnostic training

Security Implication

If **C** shows poor performance, adversaries could evade real-world detectors simply by using ML-generated images as carriers.

Expected Outcome [H4]

10–25% AUC degradation in cross-domain conditions (C, D) vs. within-domain (A, B), paralleling findings in image deepfake detection [Wang et al., 2020].

Phase 4.1: Full Experiment Matrix

Detector	Training?	Method	Payload	Encryption	Condition	# Runs
RS Analysis	None	LSB	Low, Med, High	Plain, AES	A–E (all)	$1 \times 3 \times 2 \times 5 = 30$
Chi-square	None	LSB	Low, Med, High	Plain, AES	A–E (all)	30
SRM + FLD	3-fold CV	LSB	Low, Med, High	Plain, AES	A, B, C, D, E	30
SRM + FLD	3-fold CV	DCT	Low, Med, High	Plain, AES	A, B, C, D, E	30
					Total unique configurations:	**120**
					SRM training runs ($\times$3-fold):	**$\sim$54**
					RS / chi-square: no training	**0**

Compute Estimate

- RS Analysis & chi-square: **seconds per image**, 0 training
- SRM: $\sim$30 sec/run $\times$ 54 runs $\approx$ **27 min total**
- Image generation (Wk 1): $\sim$4 h (M4 Pro, MPS)
- **Total detection compute: $<$1 hour**

Experiment Tracking

- CSV/JSON logs per configuration
- Columns: detector, method, payload, encryption, condition, AUC, EER, accuracy
- SRM: save FLD weights per fold for reproducibility
- No experiment tracking service needed (no GPU training)

Phase 5: Evaluation Metrics

Detection Performance (Primary)

ROC-AUC (primary)

- Area under ROC curve
- Threshold-independent; 0.5 = random, 1.0 = perfect

Accuracy @ optimal threshold

- Percentage correct; threshold via Youden's J

EER (Equal Error Rate)

- Point where FPR = FNR; lower is better

Image Quality (Secondary)

PSNR

- Peak Signal-to-Noise Ratio (dB); higher is better
- Target: $>$ 40 dB for imperceptible embedding

SSIM

- Structural Similarity Index; scale 0–1
- Captures luminance, contrast, structural changes

FSIM

- Feature Similarity; based on phase congruency

Payload Integrity

BER = 0 expected for lossless PNG; non-zero indicates implementation error.

Phase 5.1: Statistical Analysis

Two-Way ANOVA

Factors:

- Carrier source (real vs. ML-gen)
- Embedding method (LSB vs. DCT)

Covariates: Payload rate, encryption

Tests:

- Main effect of carrier source (Primary RQ)
- Main effect of embedding method (RQ2)
- Interaction effect (source $\times$ method)

Significance: $\alpha = 0.05$ with Bonferroni correction.

Effect Size Analysis

Cohen's *d*

- Standardized mean difference
- $|d| < 0.2$: negligible; $|d| \approx 0.5$: medium; $|d| > 0.8$: large

Why it matters:
Statistical significance $\neq$ practical significance. With multiple conditions, even small AUC differences may be significant.

Hypothesis Testing

H4 predicts 10–25% AUC drop in cross-domain. Test: H_0: $\text{AUC}_{\text{cross}} = \text{AUC}_{\text{within}}$ vs. H_1: $\text{AUC}_{\text{cross}} < \text{AUC}_{\text{within}}$

Phase 5.2: Key Visualizations

1. ROC Curves by Condition

- Overlay A, B, C, D, E on same plot
- Separate subplot per method (LSB/DCT)
- Show AUC in legend; highlight cross-domain gap

2. Heatmaps

- AUC matrix: Carrier $\times$ Payload $\times$ Method
- Confusion matrices per condition
- Cross-domain performance drop visualization

3. Payload Sensitivity Curves

- X: Payload rate (Low $\rightarrow$ High); Y: AUC
- Separate lines: Real vs. ML-gen
- Separate panels: LSB vs. DCT
- Shows divergence (H2) and method interaction (H3)

4. Image Quality Profiles

- PSNR and SSIM vs. payload rate
- Separate lines: Real vs. ML-gen carriers
- Visual examples: cover / stego side-by-side

Tools: Matplotlib, Seaborn, scikit-image, piq

Implementation Summary

Phase	Deliverable	Key Steps	Success Criteria
1	1,000 images (500 real + 500 ML)	Collect RAISE/COCO; Generate SD/StyleGAN3; normalize to 512$\times$512 PNG	BRISQUE $<$ 50; all same format
2	12,000 stego images	LSB + DCT at 3 levels $\times$ plain/AES	PSNR $>$ 40 dB, BER = 0
3	Detection results	RS Analysis + chi-square (no training); SRM 3-fold CV (CPU)	AUC $>$ 0.7 within-domain
4	120 configs	Run A–E conditions; log all metrics	All configs evaluated
5	Analysis + paper	ANOVA; visualizations; effect sizes	Publication-ready

Key References

- Chi-square: Westfeld & Pfitzmann (1999)
- RS Analysis: Fridrich et al. (2001)
- SRM: Fridrich & Kodovský (2012)
- Stable Diffusion: Rombach et al. (2022)
- StyleGAN3: Karras et al. (2021)

Critical Path

1. Week 1: Dataset collection + generation
2. Week 2: Embedding pipeline (LSB + DCT + AES)
3. Weeks 3–4: RS Analysis + SRM detection (CPU, fast)
4. Week 5: Cross-domain experiments + analysis
5. Weeks 6–7: Statistics, visualizations + writing

Implementation Guide Complete

Ready to Begin Phase 1

Questions?