Does the Source of Carrier Image Affect Steganographic Detectability?

Midway Project Proposal

Project 2.2, Department of Advanced Computing Sciences

Nico Nikolas Abdul Daria Jimena David

Maastricht University · February 2026

1. Motivation and Problem Statement

Image steganography (the practice of hiding secret information within digital images) is a fundamental topic in information security [1, 2]. The detectability of hidden content is not absolute: it depends critically on the *statistical properties of the carrier image*. Steganalysis systems exploit the subtle distributional changes that embedding introduces into pixel values, and a well-chosen carrier that already exhibits high local variability can mask these changes [3, 4]. The implicit assumption underlying nearly all published steganalysis work is that carriers are *real photographs*: images captured by digital cameras with well-characterized noise distributions, sensor patterns, and compression histories.

This assumption is increasingly untenable. Generative AI models including Stable Diffusion [5], StyleGAN3 [6], DALL-E 3, and Midjourney now produce images that are perceptually indistinguishable from real photographs, yet are synthesized through entirely different processes: latent diffusion over learned image manifolds, adversarial training against a discriminator, or transformer-based token prediction. Each process imposes a *distinct statistical fingerprint* on the output: diffusion models produce characteristic noise residual patterns, GANs leave spectral artifacts at high frequencies, and all generative models operate within learned data distributions that differ from the empirical distribution of camera photographs [12, 13].

The central problem this study addresses is: **do existing steganalysis methods, designed and validated on photographs, remain effective when the carrier image is ML-generated?** This question has three concrete implications:

- **Security:** If ML-generated images are harder to steganalyze, adversaries could trivially evade current detectors by switching to synthetic carriers, without changing the embedding algorithm at all.
- **Scientific:** The interaction between a generative model's learned data distribution and the distributional perturbation caused by steganographic embedding is theoretically unexplored. Understanding it requires controlled empirical study.
- **Practical:** As AI-generated images proliferate across social media and digital communications, the steganographic attack surface is silently expanding. Practitioners need evidence on whether existing tools require retraining or adaptation.

Prior work most closely related to ours, De et al. [7], showed that AI-generated images can achieve statistically undetectable steganographic embedding using minimum-entropy coupling. However, that study used a bespoke probabilistic embedding scheme and did not systematically compare standardized methods (LSB, DCT) across real vs. ML-generated carriers. No existing work provides a controlled comparison of carrier origin under identical embedding and steganalysis conditions.

We propose to fill this gap with a $2\times2\times3\times2$ factorial experiment: two carrier types (real photographs vs. ML-generated), two embedding methods (spatial LSB and frequency-domain DCT), three payload levels, and two steganalysis detectors (RS Analysis and SRM+FLD). The study uses 500 real and 500 ML-generated images and is designed to be completed within 7 weeks using only CPU-based open-source tools.

2. Research Questions

RQ1 (Carrier Origin Effect, Primary)
Within a 7-week study using 500 real and 500 ML-generated images embedded with identical LSB and DCT methods at three payload levels, does carrier origin produce a statistically significant difference ($\alpha = 0.05$, Bonferroni-corrected) in steganalysis AUC, as measured by RS Analysis and SRM+FLD?

RQ2 (Payload Sensitivity)
Across Low, Medium, and High payload rates (≈0.08–0.32 bpp), does the AUC gap between real and ML-generated carriers increase monotonically, and does this trend differ between LSB and DCT embedding (interaction effect in two-way ANOVA)?

RQ3 (Embedding Method Interaction)
Does the embedding method (spatial LSB vs. frequency-domain DCT) interact significantly with carrier origin in determining detectability, as quantified by a two-way ANOVA F-statistic with Bonferroni correction applied across six tested hypotheses?

RQ4 (Payload Encryption Effect)
When the embedded payload is pre-encrypted with AES-256-CBC before LSB or DCT embedding, does the steganalysis AUC change significantly compared to unencrypted embedding, and does this effect differ between carrier types (real vs. ML-generated)?

3. Chosen Approaches

3.1 Datasets

Real images (500 total). We draw from three established photographic datasets chosen for their diversity

and research accessibility: **RAISE** [14] contributes 250 RAW-demosaiced DSLR images spanning outdoor, indoor, portrait, and macro scenes; **COCO** [15] contributes 150 images from its validation split; and **Flickr30k** [16] contributes 100 images. All images are normalized to 512×512 px, RGB, 8-bit, lossless PNG. RAISE is preferred as the primary source because its RAW format preserves camera sensor noise structure, which is the natural image statistics that steganalysis exploits.

ML-generated images (500 total). We generate two matched sets of 250 images each using **Stable Diffusion v2.1** [5] (via the `diffusers` library on Apple MPS) and **StyleGAN3** [6] (official NVIDIA PyTorch implementation). Prompts for SD are derived directly from COCO/Flickr30k captions to achieve semantic alignment with real images. A BRISQUE ≤ 50 quality gate rejects perceptually degraded outputs.

These two generative paradigms (latent diffusion and GAN-based synthesis) represent the dominant architectures in open-source image generation and are expected to impose distinct statistical signatures on the output.

3.2 Embedding Methods

We implement two canonical steganographic methods spanning the two principal domains:

LSB substitution (spatial domain) replaces the k least significant bits of each pixel channel value with pseudorandom message bits, using a PRNG-keyed pixel selection mask. We test $k=1$ (Low, Medium payload) and $k=2$ (High payload). Payloads are optionally pre-encrypted with AES-256-CBC before embedding, addressing RQ1's encryption sub-question.

DCT-based embedding (frequency domain) partitions each image channel into non-overlapping 8×8 blocks, computes the 2D DCT, and embeds bits into mid-frequency coefficients (zigzag positions 10–54) via **Quantization Index Modulation** (QIM) [11]:

$$C_i' = \Delta \cdot \mathrm{round}\left(\frac{C_i}{\Delta}\right) \pm \frac{\Delta}{4},$$

where the sign encodes the message bit. DCT embedding is chosen alongside LSB to test whether frequency-domain methods are more sensitive to carrier origin (H3), since DCT coefficients reflect the generative model's learned spectral distribution directly.

3.3 Steganalysis Detectors

Our detector selection is deliberately scoped to *classical signal processing and statistics*, consistent with this project's cryptography/steganography focus:

Table 1: Steganalysis detector comparison.

Detector	Type	Training	Domain	Time
RS Analysis [8]	Statistical	None	Any	∼2 s/img
χ^2 attack [9]	Statistical	None	LSB	<1 s/img
SRM+FLD [4]	Classical ML	Labeled	LSB+DCT	<30 min

RS Analysis [8] partitions an image into pixel groups and classifies each as Regular or Singular by a smoothness function; LSB embedding shifts the R/S ratio predictably, yielding an analytical estimate $\hat{p}$ of embedding rate. Since it requires no training, any detection difference between real and ML-generated carriers is attributable solely to carrier statistics and not to classifier bias.

SRM + Fisher Linear Discriminant (FLD) ensemble [4] extracts ∼35,000-dimensional co-occurrence feature vectors from high-pass residuals, then classifies with an ensemble of FLD classifiers. It handles DCT embedding better than the training-free methods and its hand-crafted features are hypothesised to generalise better across the real/ML boundary than learned neural network representations. Implemented with `scikit-learn`'s `SGDClassifier`; 3-fold stratified CV.

The χ^2 attack [9] is applied as a supplementary check on LSB results.

3.4 Validation

Detection: ROC-AUC (primary, threshold-independent); accuracy at Youden's J; EER; FPR at 5% FNR. **Image quality:** PSNR (>40 dB target), SSIM (>0.95 target), FSIM. **Statistics:** Two-way ANOVA (carrier × method) on AUC with payload as covariate; Wilcoxon signed-rank for pairwise comparisons; Cohen's d effect sizes; Bonferroni correction ($\alpha_{\text{adj}} = 0.05/6 \approx 0.0083$) across six hypotheses.

4. Experiments

Each research question maps to exactly one experiment; every experiment links back to its RQ.

Exp. 1 (RQ1: Carrier Origin Effect). Apply RS Analysis and SRM+FLD to all 1,000 images embedded at all payload levels and methods. Compute AUC per carrier type (conditions A, B). Compare real vs. ML-generated AUC with Wilcoxon signed-rank test and Cohen's d; apply Bonferroni correction. A significant difference (Bonferroni-corrected $p<0.0083$) with $|d|>0.2$ confirms H1.

Exp. 2 (RQ2: Payload Sensitivity). From Exp. 1 results, plot AUC vs. payload level (Low/Medium/High) separately for real and ML-generated carriers, with separate curves per embedding method. Test whether the real–ML AUC gap increases monotonically using Spearman's ρ on the difference series; test the carrier × payload interaction in ANOVA.

Exp. 3 (RQ3: Method Interaction). Run a 2×2 two-way ANOVA with factors carrier origin (real/ML) and embedding method (LSB/DCT) on SRM AUC scores. A significant interaction (F-test, Bonferroni-corrected) indicates the method's detectability gap depends on carrier origin, confirming H3.

Exp. 4 (RQ4: Encryption Effect). For each carrier type and embedding method, compare AUC scores between the plain-payload and AES-256-CBC-encrypted-payload conditions. AES encryption randomises the bit pattern of the message before embedding; if detector AUC drops significantly (Wilcoxon signed-rank, Bonferroni-corrected), this indicates that message structure contributes to detection beyond purely carrier-level embedding distortion. An interaction with carrier type (real vs. ML) would suggest that generative model statistics moderate the encryption benefit.

5. Prototype

Vertical prototype (algorithm depth): Isolated implementation and verification of all four core algorithms: (1) LSB embedding and extraction, verified by BER $=0$ on a 25-image test set; (2) DCT-QIM embedding, verified by lossless payload recovery; (3) RS Analysis, validated against published estimates on known-cover images; (4) SRM feature extraction, verified by reference AUC >0.70 on a 25-cover/25-stego pair set.

Horizontal prototype (integration breadth): The four verified algorithms connected into an end-to-end pipeline on a 50-image subset (25 real + 25 ML) at medium LSB payload, validating inter-component interfaces before scaling to the full 1,000-image experiment.

6. Related Work

6.1 Generative Steganography

Hu et al. [19] and Liu et al. [20] use GANs and diffusion models *as the embedding mechanism itself*, synthesising images that inherently encode a message without any post-hoc modification. Duan et al. [21] developed coverless steganography that generates stego-images from scratch. Our work is fundamentally different: we use ML-generated images purely as *passive carriers* for standard LSB/DCT embedding, without modifying the generation process. The research question of how the carrier's statistical origin affects detectability is orthogonal to generative embedding.

6.2 AI-Generated Images as Carriers

De et al. [7] is the closest prior work, demonstrating steganographic secret sharing via AI-generated photorealistic images using minimum-entropy coupling. However, three key differences distinguish our study: (1) De et al. use a bespoke probabilistic embedding scheme rather than standard LSB/DCT; (2) they do not compare real vs. ML-generated carriers side-by-side under controlled conditions; and (3) they do not evaluate steganalysis detection rates. Our study directly fills these gaps with a factorial design that isolates carrier origin as the independent variable.

6.3 Cross-Domain Steganalysis

Recent work has studied cross-domain generalisation in steganalysis, specifically training on images from one camera model and testing on another [4]. Unsupervised domain adaptation and self-supervised approaches have been proposed to bridge this gap. However, none of this work examines the specific shift from natural photographs to ML-generated images, which is qualitatively different from inter-camera variation: it involves a shift in the entire generative process, not merely sensor noise characteristics.

6.4 Deepfake and Synthetic Image Detection

Wang et al. [12] showed that CNN-generated images are surprisingly detectable by simple linear classifiers, confirming that generative models impose statistical regularities absent from photographs. Corvi et al. [13] extended this to diffusion-model-generated images. We leverage these findings (ML-generated images do have different statistical properties from photographs) and apply the same insight to steganalysis rather than image forensics. The key difference: deepfake detection aims to distinguish real from fake; we aim to understand how this statistical difference affects the *detectability of embedded content*.

6.5 Classical vs. Deep Steganalysis

State-of-the-art steganalysis uses deep residual networks achieving near-perfect AUC on standard benchmarks [17]. We deliberately choose SRM+FLD (classical ML) over these neural approaches for three reasons: (1) it matches our course scope in cryptography/steganography; (2) its hand-crafted features are interpretable and less likely to overfit to carrier-specific artefacts, making it a fairer cross-domain test; (3) it runs on CPU in minutes, making the full 1,000-image study feasible within the project timeline.

7. Relation to Curriculum

This project applies core concepts from **Cryptography and Steganography** (LSB/DCT embedding, AES-256 encryption, information-theoretic detectability), **Research Methods** (factorial experimental design, ANOVA, effect sizes, hypothesis testing), **Machine Learning** (SRM+FLD feature-based classification), and **Algorithm Design and Data Structures** (DCT and QIM implementation in Python).

8. Planning

The project is divided into two phases aligned with the Semester 2 academic calendar. Detailed Gantt charts are provided in Appendix A.

Phase 2: Implementation (Period 5, 30 Mar–15 May 2026, 7 weeks) covers three parallel workstreams: dataset construction and ML image generation (Wk 1–2), steganography pipeline implementation including LSB, DCT, and AES-256 encryption (Wk 2–3), and detection, analysis, and writing (Wk 3–7).

Phase 3: Completion (Project Period, 25 May–12 Jun 2026, 3 weeks) covers completing any remaining implementation, verifying and rerunning experiments, and finalising all deliverables (presentation slides, poster, and paper).

9. Minimal Passing Requirements

Product: Functional LSB and DCT embedding pipelines (plain and AES-256 encrypted); RS Analysis and SRM+FLD detectors evaluated on all 1,000 images across both encryption conditions.

Validation: RQ1 and RQ4 answered with significance tests on AUC across carrier types and encryption conditions; null results characterised with 95% CIs.

10. References

References

[1] F. A. P. Petitcolas, R. J. Anderson, and M. G. Kuhn, "Information hiding: a survey," *Proc. IEEE*, vol. 87, no. 7, pp. 1062–1078, 1999.

[2] A. Cheddad, J. Condell, K. Curran, and P. McKevitt, "Digital image steganography: Survey and analysis of current methods," *Signal Process.*, vol. 90, no. 3, pp. 727–752, 2010.

[3] M. Hussain, A. W. A. Wahab, Y. I. B. Idris, A. T. S. Ho, and K. H. Jung, "Image steganography in spatial domain: A survey," *Signal Process.: Image Commun.*, vol. 65, pp. 46–66, 2018.

[4] J. Fridrich and J. Kodovský, "Rich models for steganalysis of digital images," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 3, pp. 868–882, 2012.

[5] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proc. IEEE CVPR*, pp. 10684–10695, 2022.

[6] T. Karras, M. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Alias-free generative adversarial networks," in *Proc. NeurIPS*, vol. 34, pp. 852–863, 2021.

[7] A. De, W. Kinzel, and I. Kanter, "Steganographic secret sharing via AI-generated photorealistic images," *EURASIP J. Wireless Commun. Netw.*, art. 108, 2022.

[8] J. Fridrich, M. Goljan, and R. Du, "Detecting LSB steganography in color and grayscale images," *IEEE Multimedia*, vol. 8, no. 4, pp. 22–28, 2001.

[9] A. Westfeld and A. Pfitzmann, "Attacks on steganographic systems," in *Proc. 3rd Int. Workshop Information Hiding*, LNCS 1768, pp. 61–76, 1999.

[10] N. Provos and P. Honeyman, "Hide and seek: An introduction to steganography," *IEEE Security Privacy*, vol. 1, no. 3, pp. 32–44, 2003.

[11] B. Chen and G. W. Wornell, "Quantization index modulation: A class of provably good methods for digital watermarking and information embedding," *IEEE Trans. Inf. Theory*, vol. 47, no. 4, pp. 1423–1443, 2001.

[12] S. Y. Wang, O. Wang, R. Zhang, A. Owens, and A. A. Efros, "CNN-generated images are surprisingly easy to spot... for now," in *Proc. IEEE CVPR*, pp. 8695–8704, 2020.

[13] R. Corvi, D. Cozzolino, G. Zingarini, G. Poggi, K. Nagano, and L. Verdoliva, "On the detection of synthetic images generated by diffusion models," in *Proc. IEEE ICASSP*, pp. 1–5, 2023.

[14] D.-T. Dang-Nguyen, C. Pasquini, V. Conotter, and G. Boato, "RAISE: A raw images dataset for digital image forensics," in *Proc. ACM MMSys*, pp. 219–224, 2015.

[15] T.-Y. Lin et al., "Microsoft COCO: Common objects in context," in *Proc. ECCV*, LNCS 8693, pp. 740–755, 2014.

[16] P. Young, A. Lai, M. Hodosh, and J. Hockenmaier, "From image descriptions to visual denotations," *Trans. Assoc. Comput. Linguist.*, vol. 2, pp. 67–78, 2014.

[17] Y. Luo et al., "Deep learning for steganalysis of diverse data types: A review," *Neurocomputing*, Elsevier, 2024.

[18] V. Holub, J. Fridrich, and T. Denemark, "Universal distortion function for steganography in an arbitrary domain," *EURASIP J. Inf. Security*, art. 1, 2014.

[19] P. Hu et al., "A coverless image steganography based on generative adversarial networks," *Electronics*, vol. 12, no. 5, art. 1253, 2023.

[20] X. Liu et al., "Message-driven generative steganography using GAN," *IEEE Trans. Dependable Secure Comput.*, 2024.

[21] X. Duan, D. Jia, B. Li, D. Guo, E. Zhang, and C. Qin, "Coverless steganography based on generative adversarial network," *EURASIP J. Image Video Process.*, art. 46, 2020.

A. Project Gantt Charts

Phase 2: Implementation

Period 5: 30 March – 15 May 2026 (7 weeks)

WBS	Task	Start	End	Days	Wk 1	Wk 2	Wk 3	Wk 4	Wk 5	Wk 6	Wk 7
1	**Data Construction**										
1.1	Dataset collection (RAISE/COCO/Flickr30k)	30.03	10.04	10							
1.2	ML image generation (SD v2.1 + StyleGAN3)	30.03	10.04	10							
1.3	BRISQUE quality gate & normalisation	07.04	17.04	8							
2	**Steganography Implementation**										
2.1	LSB embedding pipeline ($k=1,2$)	07.04	17.04	8							
2.2	DCT-QIM embedding pipeline	07.04	17.04	8							
2.3	AES-256-CBC payload encryption	14.04	17.04	4							
3	**Detection & Analysis**										
3.1	RS Analysis + χ^2 detector	14.04	24.04	8							
3.2	SRM + FLD training	14.04	01.05	14							
3.3	Image quality metrics (PSNR/SSIM/FSIM)	21.04	01.05	8							
3.4	Encryption-effect experiments (RQ4)	28.04	08.05	8							
3.5	Statistical analysis (ANOVA, Wilcoxon)	28.04	08.05	8							
4	**Writing & Revision**										
4.1	Visualisations + report writing	04.05	15.05	8							
4.2	Buffer / final revision	11.05	15.05	5							

Data Steganography Detection Evaluation Analysis Writing

Milestones: **M1** (end Wk 2): Dataset ready **M2** (end Wk 3): Embedding pipelines verified **M3** (end Wk 5): All experiments complete **M4** (end Wk 7): Report submitted

Phase 3: Completion

Project Period: 25 May – 12 June 2026 (3 weeks)

WBS	Task	Start	End	Days	Wk 1	Wk 2	Wk 3
1.1	Complete remaining implementation	25.05	05.06	10			
1.2	Verify results & rerun experiments	25.05	05.06	10			
1.3	Statistical verification & effect sizes	01.06	05.06	5			
2.1	Presentation slides	01.06	08.06	6			
2.2	Poster	05.06	10.06	4			
2.3	Final paper write-up	01.06	12.06	10			
2.4	Buffer / revision / submission	10.06	12.06	3			

Implementation Verification Analysis Slides Poster Paper

Milestones: **M5** (end Wk 1): Implementation finalised **M6** (end Wk 2): Results verified, slides ready **M7** (end Wk 3): All deliverables submitted