Master Degree Project

UNIVERSITY OF SKÖVDE

Needles in an infinite haystack: cover source, payload rate, and detectability in diffusion-generated steganography

Master Degree Project in Informatics with a specialization in Privacy, Information and Cyber Security

Second Cycle 30 ECTS

Spring term 2026

Student: Erik Mattias van den Dool Enebjörk

Supervisor: Oskar MacGregor

Examiner: Ali Padyab

ABSTRACT

With increased surveillance of online communication, there is a need for sending secret messages in a secure and inconspicuous way. One approach is steganography, which means hiding data in other data. This study examines whether images generated by diffusion models have favorable characteristics for embedding payloads. AI-generated images are now ubiquitous and could provide an endless source of cheap and disposable cover images for steganography. The study also assesses the feasibility of generating these images locally on consumer hardware along with an integrated and educational end-to-end image generation, steganography, and steganalysis tool. Lastly, it tests if provenance watermarks in images are damaged by steganographic embedding.

To answer these questions, the research tool Stegosaurus was built. Using this tool and consumer hardware, a corpus of 2,500 diffusion images was generated. Payloads were then embedded into 20,000 image pairs from six different image sources. The image pairs were analyzed using six statistical tests, one a sanity check and the others each corresponding to a potential adversary with increasing knowledge or access to the images or pipeline.

The most important finding is that the noise profile of the diffusion images had the least random low-bit plane among all tested sources, while minimally processed RAW photos were close to randomness. RAW was the only source in which a trained classifier could not detect the payload, owing to its unstructured and channel-independent noise.

Moreover, steganographic embedding did not destroy or prevent detection of the three types of AI-provenance watermarks tested. Google DeepMind's SynthID remained detectable and correctly attributed even with an embedding at 90% of capacity, which altered nearly half of the image's color values. The integrated tool shows how steganography could be implemented and used in practice.

Abstract i
1. Introduction 1
2. Background 3
2.1 Image steganography 3
2.2 Surveillance capitalism 6
2.3 Image diffusion 8
2.4 Steganalysis 9
2.5 Watermarking 10
2.6 Related-work gap 11
3. Methodology & Implementation 12
3.1 Adversary model 12
3.2 Corpus and protocol 13
3.3 Pipeline and statistical tools 15
3.4 Watermarks 17
3.5 Corpus generation 18
3.6 Stegosaurus 19
3.7 Reproducibility 19
4. Results 20
4.1 Local generation in practice 20
4.2 Cover source detectability 20
4.3 Generated noise profile 22
4.4 Payload rate as a better predictor 24
4.5 The impact of the coding layer 25
4.6 Watermarks are conserved 26
4.7 Stegosaurus graphical tool 28
5. Discussion 30
5.1 Camera sensor noise 30
5.2 Mass generation of cover images 31
5.3 Embedding rate vs cover source 32
5.4 Watermark survival 33
5.5 Adversary model limitations 34
5.6 PNG and image domains 34
5.7 Ethical and societal aspects 35
6. Conclusion 36
7. Future work 38
References 39

1. INTRODUCTION

There are situations in which an individual needs to share sensitive information with others. As the daily communication of billions of people is nowadays digital, the likely medium for this information exchange would be a popular social chat or messaging application, such as Gmail, iMessage, WeChat, or WhatsApp. These applications promise a degree of privacy (Alatawi & Saxena, 2023), but as the encryption and infrastructure are controlled by the service providers, the applications are open to abuse, snooping, and surveillance by different parties (Srivastava, 2019). They can hence be regarded as insecure channels for sending sensitive information.

Multiple algorithms and tools have been invented to make sensitive information unreadable to anyone except the intended recipient, should someone manage to intercept the communication. For example, encryption makes the information unreadable, but advertises that something is being hidden. To conceal that something secret is being communicated, steganography, which hides information within other information, has been developed. To make the information unreadable as well as conceal its existence, encryption can be combined with steganography. Information can be concealed in any type of data, such as audio, images, text, and video. For images, three general approaches are common: Convolutional Neural Network methods, Generative Adversarial Network methods, and Least Significant Bit substitutions (Subramanian et al., 2021).

Although there is quite some literature on this topic, the studies mostly focus on lossy JPEG images (Tao et al., 2019) or a different scope of applications (Qiao et al., 2023). The rapid rise of AI in general, and AI image generation in particular has changed what kind of images are sent and shared over the internet. The image format PNG, which is ideal for computationally and conceptually simple least significant bit steganography (Aslam, 2022), has had a resurgence thanks to online image generators. What was until recently a fringe format online is now more common. This unlocks interesting questions on what can be done with AI images and the PNG format in the space of steganography for ordinary individuals who want to communicate messages secretly without appearing suspicious to an observer that is monitoring the channel.

Someone who generates cover images locally on their own computer with model weights, parameters, and prompts that they control themselves, can change the covers as much and as often as they want in exactly the way they want. Since detectors are trained on pools of specific images, they perform worse and are likely ineffective when applied to a different pool of images. This makes it hard to deploy steganalysis in real-world scenarios and is called the cover-source mismatch problem (Mallet et al., 2024).

AI-generated images also increasingly have watermarks added to them during creation to be able to show provenance. Both the public and the tech companies themselves want to know if an image is AI-generated, and if so, by whom. Google's SynthID has been applied to over ten billion images and video frames (Gowal et al., 2025) and OpenAI recently decided to start using SynthID as well. Adobe promotes TrustMark (Bui et al., 2025). If hiding a message damages the watermark, the damage is itself a detection signal, and one that requires no steganalysis at all, only a verification query. This possibility had not previously been measured. The watermarking literature tests robustness against

compression, cropping and regeneration, none of which is steganography. Likewise, the steganography literature asks about detectability, not what happens to a watermark that was already embedded in the cover image before steganographic embedding.

In response, this study aims to examine AI image generation applied to steganography and steganalysis. It is structured around the following questions: (R1) Is diffusion-based local image generation on consumer hardware a viable option for mass-producing disposable cover images for steganography? (R2) Does the noise profile of diffusion-generated covers provide a more random or less random canvas for least significant bit steganography compared to natural images? (R3) Does least significant bit steganography interfere with embedded watermarks that prove provenance of generated images?

To answer those questions, a hybrid command-line and graphical tool was created that integrated image generation, steganographic embedding and extraction, steganalysis, watermark verification, visualization, as well as automated batch testing and analysis.

The next section provides an overview of the existing literature on this topic which is followed by the methodology. The second part of the paper presents the results and discusses the findings in light of the existing research. The conclusion provides ideas for applications and future research.

2. BACKGROUND

Censorship is becoming increasingly automated. Heuristics, different types of machine learning, and large language model AIs are now used to scan user content stored in cloud services or sent through online platforms. While censorship and surveillance of private correspondence is traditionally seen as something practiced mainly by authoritarian regimes, and to some extent by intelligence agencies, digital censorship and surveillance is now practiced by both authoritarian and democratic countries. The rise of internet giants with a virtual monopoly on certain online services, such as Cloudflare, Google (Alphabet), Meta, and Tencent, has produced corporate censorship and surveillance in a way that has never previously been possible. Citizens, companies, government agencies, hospitals, journalists, activists, and whistleblowers typically all have to use a handful of services that are controlled by a small number of tech giants in the USA or China.

In response to comparable situations, a number of solutions have been developed to overcome the challenge of communicating secrets over channels that are controlled or monitored by others than the sender and recipient. These include encryption, which entails altering the secret information in a way that makes it unreadable but can be restored by anyone who has the correct key. This technique is the basis of much of the modern internet and enables someone to for example check their bank account without everyone else being able to listen in.

There are however downsides to encryption. In order for the information exchange to be guaranteed private, the key or keys must be held only by the sender and recipient. Furthermore, the encryption must be done on the devices themselves. This is called end-to-end encryption. In the case of using this kind of encryption in social messaging applications, the encrypted data has to be sent as text messages or files. As the encrypted text will look nothing like normal text, automatic surveillance can detect the encrypted data and flag it. There is no plausible deniability for the sender or recipient.

To lessen the risk of detection and to provide a degree of plausible deniability, a different approach is steganography, which will be discussed in depth in the following section before explaining how image steganography will be applied in this project.

2.1 IMAGE STEGANOGRAPHY

Steganography is the practice of hiding data inside other data. The medium or type of data of the secret can be the same as or a different one than the one that the data is concealed in. Steganography as a method has probably been around as long as writing itself as a means of transmitting secret messages without divulging the message to someone other than the intended recipient.

Importantly, steganography is different from ciphers/encryption, where the goal is to render the message illegible. An encrypted text or image appears as random symbols to any observer, which makes it clearly recognizable as encrypted. Steganography on the other hand does not scramble the message by default. Instead, the message is clearly readable if the data is correctly extracted.

A suitable example would be a text that has been crafted so that it reads as a normal text when read in full, but if every seventh character is extracted, it holds a secret message. That is steganography. The message is hidden in the larger text, but not scrambled using some algorithm.

An example of encryption would be a messenger in the ancient world bringing a letter where every character has been replaced by another character using a key table. The resulting text does not look like normal text and, depending on how complex the key and algorithm are, cannot be read without the key.

The above examples illustrate an important difference between the two concepts, namely that an encrypted message is clearly detectable as encrypted. If the messenger from the above example were caught, the captor would know that the letter is encrypted since it is not natural text. Questioning the messenger may not reveal any secrets since the messenger is unlikely to know the undeciphered content. However, it marks both the sender and recipient as clearly suspicious. At this point, the adversary could choose to go after either of them or simply capture every encrypted letter sent between them.

If the sender instead used steganography to hide the message, the letter would appear completely normal to any readers. It is impossible to confidently say that a hidden message exists within a written text. Depending on the algorithm used for extracting the message, a large number of plausible messages can be found in any larger text. It is likely possible to extract a secret message from the classic book Moby Dick, claiming that the pope was behind the murder of prince Franz Ferdinand of Austro-Hungaria. That does not mean that such a message was ever consciously encoded in the text.

This property of steganography, that it provides plausible deniability, makes it useful for purposes that involve safely and unnoticeably communicating in an environment where an adversary is transporting and/or reading all messages that are sent between individuals or groups of people.

Secret messages do not necessarily have to be written using characters. The message might not even be natural language. It could just as well be a drawing, photograph, video, or audio recording. All of these could be hidden in exactly the same type of media, i.e., an audio recording hidden as audio inside another audio recording.

Alternatively, the message can be transformed in some way and hence represented by something else. For example, a sentence of text could be painted as a series of dots on a tree trunk in a landscape painting or transformed to certain frequencies or amplitudes in an audio recording. This would not be encryption, but simply a way to represent one type of media on a different type of canvas.

Contrary to analog mediums like those mentioned above, digital computers can only store two discrete states: charge (1) or no charge (0). This means that every type of data, whether it is audio, images, text, video or something else, must be transformed to a representation made up of ones and zeroes, also called binary.

Transforming characters to ones and zeros is conceptually the same as transforming them to dots in a painting. The difference is the scale, speed, and precision with which digital conversions can be made.

For steganography, digital representation of data provides an opportunity. A one or zero in a text is no different from a one or zero in a video. The distinction between the two file types comes down to software interpreting the ones and

zeros and how they are structured. Therefore, hiding one type of media in another and extracting it again is in theory a fairly simple task in the digital domain since it can be done algorithmically and automatically on binary data.

On the other hand, the same digital advantages that allow automated and accurate steganography also allow automated and precise scanning of data to try to determine if it has been altered using steganography. This is called steganalysis.

Considering the earlier claim that ones and zeros of text and video are no different from each other, and that the former can therefore be hidden in the latter, the notion of steganalysis may seem surprising. However, as mentioned above, representing a data type depends on both its structure and interpretation. The software that interprets the data expects a given data type to have a particular structure, i.e., to look a certain way. For example, an audio recording might have a barely noticeable cyclic hum in the background. If the recording is altered by putting a few extra ones or zeros in it, the hum might no longer be cyclic or the frequencies might change in unnatural ways. This unnaturalness is what steganalysis can detect.

Thus, the goal of steganography becomes to hide data while altering the cover data in a way that is indistinguishable from natural data. Conversely, the goal of steganalysis is to find even the tiniest deviations from what is natural, without flagging normal variations or outliers. It is a game of cat and mouse, with increasingly sophisticated algorithms hiding data and detecting hidden data.

Cachin (2004) gives a formal definition of steganographic security, which says that a steganographic system is secure when an object that contains hidden data statistically look the same as an object that is unaltered. Perfect security is defined as the embedding causing no statistical change at all and thus being completely identical even to an ideal observer.

For natural photos we cannot know the full statistical distribution of all possible images. This means that real-world steganography with natural images has to rely on approximations to make changes caused by embedding harder to detect, but cannot guarantee security.

Diffusion-generated images come from a fixed model and a controlled process. Because the distribution is more controlled and reproduceable compared to the unpredictability of natural photos, diffusion models could potentially be useful as a foundation for studying security. This is true regardless of whether the noise profile of diffusion-generated images is advantageous or not.

To make least-significant bit PNG steganography more secure as per the formal definition above, content-adaptive steganography can be used. This is based on the idea that different parts of an image can be assigned different cost values depending on how visible a change in that part would be, combined with a coding method which decides which locations must be changed to embed to payload. Changes to pixels in smooth areas are expensive, while highly textured regions are assigned a lower cost.

There are a number of different methods to estimate this risk and decide the costs. UNIWARD (Holub et al., 2014) uses something called wavelet filters to find locations where changes would be inconspicuous across several directions and scales. HILL (Li et al., 2014) instead identifies edges and texture using high-pass filtering and prefers to place changes within those regions. MiPOD (Sedighi et al., 2016) builds a statistical model of each pixel and calculates how noticeable

a change at that position would be to an ideal observer. All three methods use different algorithms to calculate the costs of embedding in difference regions, but they all produce a map that shows with the locations where the payload should preferably be embedded.

Complimentary to the cost models is the coding layer. Syndrome-trellis codes (Filler et al., 2011) try to minimize the total cost of the required changes. Instead of naïvely assigning each payload bit to a particular pixel, the code lets the encoder choose among a number of possible sets of changes and selects the solution with the lowest cost. The embedding process uses the cost map to find the lowest cost pixels, but takes the actual bit to flip into consideration to produce a stego image with as few changes as possible in expensive areas.

2.2 SURVEILLANCE CAPITALISM

Digital devices have not only changed how data is stored and handled. They have also changed how and where we communicate. Physical letters, telephone calls, and even in-person meetings have mostly been replaced by a handful of chat, messaging, and video call apps. Many of these are owned by an even smaller number of companies. This gives these companies, and to varying degrees the governments of the countries in which the companies are headquartered, a unique opportunity to monitor and analyze the communication passing through the apps.

Of course, monitoring and analyzing communication is nothing new. Both benevolent and malicious actors have scanned letters, tapped phone lines, and snooped on private conversations for a long time. What is new is the capability for a handful of actors to continuously monitor the communication of a majority of the world's population and to fully automate the collection and analysis of the data.

Given such a wide-ranging surveillance capability, companies and governments may be tempted to blur or overstep ethical and legal lines and engage in unlawful surveillance of individuals and organizations. The surveillance can take many shapes. In the major contemporary economic systems, it can be highly profitable to monitor individuals and use their data to sell advertisements, make products addictive, or sell the information to data brokers who in turn sell it to other parties. This phenomenon is often referred to as "surveillance capitalism".

Importantly, surveillance capitalism is not only practiced by first-party tech companies, i.e., the ones who make the application or service that the individual publishes their data on. Many data brokers collect the data from other services using a technique called scraping, which automatically collects large amounts of data from a site or service. A well-known example is the company Clearview AI that has collected photos of millions of individuals from public profiles on the photo sharing service Instagram in order to extract the biometric data, link it to real persons, and sell it to US law enforcement among others.

Not everyone is opposed to using personal data for marketing or collecting publicly published data. However, there is also a real risk of the technology being used for surveillance and prosecution of law-abiding individuals and organizations. Corrupt politicians could use it to surveil an opposition party. A tech company could use their chat app to surveil and intimidate a journalist who is looking into a less savory part of their business dealings.

Furthermore, it may be tempting for states, both authoritarian and democratic, to use to technology to surveil the communications of all its citizens with the pretense that everyone's communication has to be continuously monitored and analyzed to catch some specific category of criminal activity that is considered so heinous that it can be claimed to justify ignoring all rights to personal privacy for everyone. Any suspicious activity, for example an image that the machine learning algorithm mis-classifies as child pornography, is then automatically reported and sent to some instance, even when the photo in question happens to be the individual's two-year-old child building a sand castle on a beach while nude. There is likely to be many orders of magnitude more innocent photos of people's loved ones stored in all the world's mobile phones and computers, than there is child porn. Even if only a fraction of these photos is mis-classified and incorrectly reported, so-called false positives, it would entail millions of photos being reviewed by complete strangers without any prior suspicion of a crime or a search warrant.

This is how the criticized EU law proposal "chat control" works, part of which is already in place and ratified, scanning popular services such as iCloud Mail, Gmail, and Facebook/Instagram messenger. The implemented version of the law asks companies to voluntarily scan unencrypted services. The proposed version makes scanning of all communication mandatory. It applies even to end-to-end encrypted communication, which means that only the participants communicating have the keys and no outside party can read it.

In such an environment, where potentially all communication is scrutinized and the means of communication are controlled by a few companies, being able to communicate secrets is important to a wide array of people. For the purpose of this text, secrets are defined as any sensitive information that could potentially be used in ways perceived as negative by the owner of the data, if someone else gets access to it. This includes for example medical and financial records, social security numbers, company product design documents, communication between journalists and anonymous sources, as well as personal messages between LGBTQ-individuals in certain jurisdictions.

Depending on who the individual is, where they are located, what they are communicating, and who is potentially listening, the need for secure communication can be very different. The evaluation of what dangers are plausible, what kind of scenarios are realistic, and which countermeasures are proportional and practical is called a threat model.

If the envisioned threat is surveillance capitalism in a full democracy and the goal is to prevent a service provider from monetizing medical records sent through a chat app, sending the records in an encrypted format would be a reasonable countermeasure. However, there is a possibility that the use of encrypted files, which is easily detected when scanning the communication, will by itself be a data point to monetize. In some jurisdictions, the act of using encryption could even be seen as suspicious behavior and lead to more surveillance and scrutinizing of the individual.

Hence, wanting to be as inconspicuous as possible is perfectly reasonable for any law-abiding citizen in an environment where existing laws and practices do not protect the individual's private correspondence from prospective abuse by companies or authorities with potentially disastrous personal consequences.

This is where steganography fits in a practical, privacy toolkit. As mentioned earlier, steganography can provide both inconspicuousness and security as the

adversary scanning the communication will only see ordinary data passing through. It is certainly feasible to scan the stream of data for steganography, depending on how it is implemented, but this requires substantial resources with potentially little reward as hypothetically billions of files would have to be analyzed and result in millions of false positives and false negatives, i.e., files incorrectly flagged as containing steganography when they are clean or as being clean when they in fact contain hidden data.

2.3 IMAGE DIFFUSION

It is now possible to generate an endless number of images that are indistinguishable from photos or illustrations created by humans. All that is needed is a prompt and an AI image generator. For maximum privacy, the image generator can be run locally, which requires a modern computer, image generation software, and an image model. A commercial image generator can also be used, often for free, and all that is required then is an internet connection and a device to access the service.

AI images are commonly generated using a modeling technique called latent diffusion. Latent means that the process takes place in a compressed representation of the image called latent space, for the sake of efficiency. Diffusion refers to the idea that a neural network can be trained by converting an image into random noise little by little, and then learn to reverse the process. Consequently, generating an image begins with random noise that is removed step by step until a recognizable image has been produced. The number of steps is arbitrary and can be tweaked to find a balance between image quality and noise levels.

That the images are generated from noise, and that various degrees of noise remain in the final image, is a property that can potentially be taken advantage of when hiding data in an image. Concealing data in the color values of each pixel creates a pattern of noise. A natural image has non-random features such as long stretches of the same color as well as similar color values next to each other within a square area. When steganography embeds a piece of data in one of the pixels, the pattern becomes more random, i.e., less natural. Against a uniformly colored and plain background, such a change would be detectable. The more random noise that exists in an image, the more likely the hidden data is to blend in. This is also the reason why steganography algorithms have a preference for busy, highly patterned areas of natural images.

While noise is good for steganography, noisy images do not automatically make alterations of pixels invisible. Nevertheless, the amount and type of noise in generated images would potentially flag them as modified by a steganalysis algorithm since the random noise makes the pixels seem unnatural. Consequently, using generated images as cover images might decrease the risk of detection since the number of false positives flagged by a steganalysis system is possibly higher than what is acceptable.

Another method that can assist with the detection of hidden data in an image is to compare the suspected image to a known original of the same image using different statistical tools. However, if a generated image is used as cover image for the hidden data, then no original image exists that the adversary can use to perform the analysis. The image is generated to be used only one time and only exists on the computer for a short time until the embedding is finished and then

discarded. In the case of the research tool used for this study, the original generated image is never even written to disk. It only exists in memory. The finished stego image, which already contains the concealed data, is thus the only copy of the image that will ever be stored and distributed.

There are also other advantages with generated images. Using these, there is no longer any need to take a photo that can give away more information about the sender's private life, location, or hardware or to search for a suitable image online which would leave behind an original to compare the stego image to. Instead, the sender can generate appropriate images for each occasion and tailor them to the natural flow of messages and memes.

2.4 STEGANALYSIS

While steganography is the act of embedding secrets into data, steganalysis is the opposite, namely finding out if an ordinary-looking file contains hidden data. This can be done in different ways depending on what kind of data is hidden, in what, and how. Theoretically, it could be as simple as looking at the metadata of a file and finding something that does not belong. More relevant for this study are structural detectors, since they specifically look for image structures that have been damaged by manipulating the pixels.

Structural detectors use algebraic properties that least-significant-bit replacement creates. RS analysis (Fridrich et al., 2001) sorts groups of pixels into regular and singular groups by a smoothness test, and reads the embedding rate from how those counts differ when certain flipping operations are performed. Sample Pairs Analysis (Dumitrescu et al., 2003) groups adjacent pixel pairs into subsets with exact relations that steganographic embedding breaks in a way that can be quantified. The chi-square attack detects the way in which the pixel alterations equalize histogram bins in an unnatural way.

These detectors work very well against naïve least significant bit replacement but have a much harder time detecting LSB matching, i.e., adding or subtracting 1, because matching does not create as much asymmetry. The distinction is important in this study since it compares both methods.

Feature-based detectors use machine learning instead. SPAM (Pevný et al., 2010) models the differences between neighboring pixels as a Markov chain, which means that the next value depends only on the current one, and uses the change probabilities as features. The Spatial Rich Model (Fridrich & Kodovský, 2012) generalizes this to tens of thousands of features and classifies them using an ensemble.

Deep-learning detectors learn their own features end-to-end instead of having someone design them by hand. SRNet (Boroumand et al., 2019) established the architecture that is most often used as a reference point. Their approach has since been developed into multiple other systems, such as Siamese networks that compare an image against a reference rather than classifying it alone (You et al., 2021), efficient spatial architectures that cut the computational cost (Zhang et al., 2020), and adaptations of general-purpose image networks to the JPEG domain (Yousfi et al., 2021). De La Croix et al. (2024) survey the field as a whole.

These methods usually outperform feature-based detectors. Because of the chosen adversary model, and the focus on image generation and watermarking, they were not evaluated in this study.

2.5 WATERMARKING

Images generated online using the most common providers of AI services, such as Google's Gemini and OpenAI's ChatGPT, contain watermarks that can be read to show that the image was created by AI. These watermarks are not virtual stamps visually overlaid in the image which used to be common on the internet. Instead, algorithms similar to steganography are used to embed information directly into the pixels. As with steganography, the goal is to be able to embed the information in an invisible manner, while simultaneously being able to correctly extract the information again. Considering that watermarking should prove provenance of an image, it is important that the embedding survives the image being cropped, resaved, rotated, and shared multiple times in different file formats. This is where it diverges from steganography.

Steganography by definition needs to both be undetectable and to be able to embed payloads that are large enough to contain meaningful amounts of text, audio, images, etc. This leaves little room for error correction redundancy and may exclude certain embedding domains. Watermarks on the other hand, only have to be invisible to the eye, so any domain can be used, and since they contain so little information, error correction redundancy can be high. Furthermore, the data is likely to be unencrypted, which makes loss of data even less critical for reading the watermark.

There are a number of different watermark technologies, and they are implemented differently. C2PA Content Credentials (Rosenthol, 2022) are cryptographically signed metadata manifests with the purpose of showing that an image has not been altered at all. This is important to for example photographers and it therefore used in some professional cameras. The image bytes are hashed and the result is then signed with the issuer's private key. Any pixel change destroys the signature as the hash changes. As a result, C2PA cannot be preserved by any steganographic method.

SynthID (Gowal et al., 2025) uses a special encoder and decoder pair that is applied to images. The vendor reports 99.98% detection under random transformations with bit accuracy falling to 72% in worst case processing. The weights are not published, so no independent extractor and decoder can be built, and the available technical and robustness information is described only in a vendor technical report. The peer-reviewed description of the SynthID family covers the text variant (Dathathri et al., 2024). This study therefore treats SynthID as a deployed system to be observed and measured from the outside rather than as a tool of which its inner workings can be examined.

TrustMark (Bui et al., 2025) is a neural watermark from Adobe with a 100-bit payload, error correction, and open weights. It is the closest publicly testable relative of SynthID.

The DWT watermark family is used by Stable Diffusion and is fully open. It comes in multiple versions of which some are hardened for extra robustness. The dwtDctSvd variant works by converting an image to a color space that separates brightness and color and then applies a wavelet transform. Watermark bits are embedded by making tiny changes to image blocks so that a specific value falls on one side or the other of a decision threshold. Each bit is repeated across roughly 67 blocks, and the final bit value is decided by majority vote. It is this repetition that makes the watermark so resistant to image modifications in comparison with steganographic embedding.

As mentioned earlier, online image generators are an option for creating cover images. The downside is that the image is generated in the cloud and that the original might be stored by the provider. There is also the question of what happens to the watermarks applied. Since watermarking and steganography are similar in many ways, it could mean that embedding data in a watermarked image might damage or destroy the watermark. A damaged watermark could then be used as a metric for detecting steganography, since it can be detected that the image has been altered in some way. It does not have to be the result of steganography, but it warrants a deeper analysis of the image.

Having a clear marker that shows whether the image has been modified defeats the reason to use generated images. Therefore, it is important to know if steganography and watermarks are compatible.

2.6 RELATED-WORK GAP

Prior work has already established that adaptive embedding with Syndrome-Trellis codes minimizes distortion of an image during embedding, that cover-source mismatch severely weakens steganalysis and that synthetic cover images can possibly take advantage of that, as well as that AI-generated images on the internet increasingly contain provenance watermarks that can be attacked and destroyed. However, three gaps remained.

The coding layer's contribution has not been isolated. Méreur et al. (2024), the closest predecessor to this study, embedded messages with three cost models at a single rate into 194,000 grayscale images from fifteen generators. They evaluated both the mismatch matrix and the matched condition. Their matched experiment is conceptually similar to the matched evaluation in this study, so no claim is made to be the first to test adaptive embedding in generated covers against a matched adversary. What their design cannot separate is the contribution of the coding layer, because every one of their arms is adaptive. This study keeps the corpus, payload, features, and evaluation procedure fixed while varying the coding layer, allowing that contribution to be examined directly. Furthermore, this study used full-color images instead of greyscale images.

The interaction between steganography and provenance watermarks has not been measured at all. The watermarking literature has tested robustness against compression, cropping, filtering, and regeneration, while steganography studies focus on whether hidden messages can be detected. Neither literature addresses what happens to watermarks when steganographic embedding is applied to an image that already contains a provenance watermark. This question is increasingly relevant because images generated by the dominant AI service providers now have such watermarks embedded by default, and steganographic payloads and watermarks can now exist together in the same image.

Existing research tooling has limited transparency. Research steganography tools are typically scripts created to process specific datasets. Many are outdated, non-functional, or opaque, which is also the case for deployment tools. Neither generates its own covers, and neither exposes the internal state such as cost maps, selected embedding locations, and steganalysis reports that would let a user understand and verify how the algorithms work instead of merely operating the tool.

3. METHODOLOGY & IMPLEMENTATION

3.1 ADVERSARY MODEL

Everything in this study depends on what the adversary is assumed to be capable of and to possess. Ker et al. (2013) argue that the gap between laboratory steganalysis and real-world systems is mainly tacit assumptions. The results of the study support this, since the assumed capability of the adversary changes the verdict more than anything else.

The adversary assumed in this study is a passive system monitoring a channel with a very large amount of mixed traffic. Images, which will be one type of data passing through, come in a multitude of formats and from a wide variety of origins. As AI is ubiquitous, a substantial share of the images passing through the channel are presumed to be AI-generated.

The hypothetical adversarial system is performing image analysis on intercepted data for the sake of content moderation, personal data extraction, and perhaps censorship. This could include comparing hashes, reading metadata, extracting text, and basic steganalysis to determine if any image contains a hidden payload. It is also assumed that the adversary has access to enough storage capacity and compute power to collect and correlate any image that is sent multiple times, and consequently flag reused images that contain steganographic patterns.

Because of the large amount of traffic over the channel, threats come in many different forms, and it cannot be assumed that the adversary has matched training data or known embedding parameters for every steganographic tool or algorithm in the wild.

Likewise, the assumed sender generates all cover images locally on their computer and destroys the original image instantly after embedding. Only the stego image is ever sent over the channel, which makes any paired comparison analysis that requires the original, such as histogram subtraction and pixel differencing, impossible.

The adversary capabilities are evaluated across six tests, of which only B and D are realistically achievable by the assumed adversary. Tests A and C are feasible if the adversary is specifically targeting one particular sender, knows precisely which tool(s) they are using, and have access to the same tool(s). Test F, on the other hand, is only ever available to the sender and not to the adversary. Test E compares two random halves of the cover pool to make sure that the corpus is homogenous and is thus a sanity-check for statistical correctness, not an adversarial tool.

Table 1

Steganalysis Tests Based on Assumed Adversary Capabilities

Test	Capability required	Realistic?
A. per-feature KS	Pool of covers and pool of stego images, separated into groups	Partly
B. joint MMD	Pool of images from one sender	Yes
C. supervised classifier	Labeled cover and stego training corpus from the same generator and stego tool	Only if the tool is known
D. blind thresholds	Only the intercepted stego image	Yes
E. sanity check	(not an adversarial test)	
F. paired fingerprint	The original cover image of each suspected stego image	No, destroyed instantly

3.2 CORPUS AND PROTOCOL

The study used six different full-scale image corpora of 2,500 images each.

1) *Diffusion corpus:* 2,500 diffusion-based PNG images generated locally using FLUX.2 klein 4B model weights. Image dimensions were 768x768 pixels. Used in experiments S3, S4, S5.
2) *Processed diffusion corpus:* The above set of 2,500 generated photos, but converted to JPEG and then converted back to PNG again in order to test the effect of JPEG block processing on the noise profile. Used in S9.
3) *Sensor-matched diffusion corpus:* The 2,500 diffusion-based FLUX.2 klein images, but with added noise to emulate a camera sensor. Used in S10.
4) *Real photos corpus:* 2,500 photographs full size JPEG photos converted to PNG and center cropped to 768x768 pixels. The photos were randomly selected from a pool of 200,000 photos created using mobile phones, DSLR, and mirrorless cameras from different brands and covered a large variety of subjects and conditions. Used in S8.
5) *Resized corpus:* The same 2,500 photographs as above, but instead of center cropping, the photos were resized to 768x768 pixels. Used in S6.
6) *Raw photos corpus:* 2,500 photographs converted to PNG from full size Nikon NEF RAW format files and center cropped to 768x768 pixels. The photos were randomly selected from a pool of 20,000 low-ISO (100-400) photos created using Nikon D300 and D700 cameras. The camera models and RAW format were selected in order to preserve as much raw sensor noise as possible. Used in S7.

The payloads were divided into two pools, named *small* and *large*, which contained famous historical letters of different lengths. The small pool contained 8 texts of between 81 and 500 words in length, while the large pool held 10 texts varying in size up to 20,339 words. Because of this, the large pool could more accurately be described as a mixed pool, spanning the whole payload range to simulate an individual sending mixed traffic as would be expected in the real world.

The random selection and subsequent compression of the texts in both pools resulted in the mean fills found in the study runs table below. In the one study run that used the large pool, the mean payload fill was 4.437%, but the median was 1.012%, with 49.7% of images below 0.5% fill and 30.4% above 5%. For the

other study runs, the mean fill was around 0.4%. This low fill ratio was deliberately chosen, as a text of 500 words was hypothesized to be sufficiently large for submitting a secret message, while still being small enough to avoid detection by not altering too many pixels.

To determine the effects of the coding layer, cover source, and payload, eight full-scale runs and two smaller intervention runs were executed. For each image in the corpora, a payload was randomly selected from one of the text pools, and then embedded in the image using the *Stegosaurus* tool. This produced 2,500 pairs of cover and stego images per study run, which were then run through the steganalysis detectors in a randomized order without any labels so that the detectors did not know which image in a pair was the cover image and which was the stego image.

Table 2

The Cover Image Source Experiments Conducted During the Study

Run	Pairs	Coding	Payload	Cover source	Mean fill
S3	2,500	LSB	Small	FLUX	0.430%
S4	2,500	STC	Small	FLUX	0.427%
S5	2,500	STC	Large (mixed)	FLUX	4.44%
S6	2,500	STC	Small	Photographs, resized	0.426%
S7	2,500	STC	Small	Camera RAW	0.426%
S8	2,500	STC	Small	Photographs, cropped	0.426%
S9	2,500	STC	Small	FLUX, JPEG round trip	0.427%
S10	2,500	STC	Small	FLUX + added sensor-like noise	0.427%
S11	162	STC	Small	OpenAI gpt-image-1	0.46%
S12	162	STC	Small	FLUX at 16 sampling steps	0.46%

Study runs S3 vs S4 focused on the coding layer, comparing naïve LSB embedding, which spreads the payload throughout the image, to STC embedding, which uses cost models to place the payload. S4 vs S5 compared the effect of payload size. S4, S6, S7, S8, and S9 isolated the cover source to compare the impact of both cover type and processing on the noise profile and suitability for least significant bit steganography. S10, S11, S12 tested whether camera-sensor-like noise could be manufactured or purchased. Other factors were held constant.

Note that there are no S1 or S2 study runs. The runs were named in the order of execution and, before the real runs began, two test runs named S1 and S2 were executed to ensure end-to-end correctness of the calibration → steganography → steganalysis study run pipeline, and the results were discarded once completed.

Both of the test runs used 2,500 generated covers, 999 permutations, and small payloads, but differed in how the images were chosen. While S1 used 2,500 generated photos, like all the runs included in this study, S2 used 2,500 covers that were randomly selected from a larger pool of generated photos on the condition that they passed the steganalysis detection after embedding. In other words, generated covers that were deemed unsuitable to hide the payload, were filtered out and discarded. Consequently, the stego images produced in S2 should, on average, be considerably less likely to trigger the steganalysis detectors because the most anomalous images have been discarded during the embedding process.

The effect of filtered vs unfiltered covers would be an interesting area to explore in the future, especially since one of the biggest advantages of generated images is that they can be cheaply and endlessly regenerated until one is found that passes the detection. However, for this first study of the feasibility of generated images as cover images, not excluding any of the generated images was deemed a more appropriate option.

3.3 PIPELINE AND STATISTICAL TOOLS

The tool, Stegosaurus, developed for the study had two parts with a shared core. It comprised of a command-line tool used for the studies and a native macOS application intended for interactive and educational use. Go was chosen for the core because its native concurrency benefited encoding and decoding and because it can produce a single static binary with no runtime dependencies. The GUI was built with Swift.

Stegosaurus integrates the complete pipeline from image generation, to embedding, to extraction, and steganalysis. Both the steganography pipeline and the steganalysis tests can be seen illustrated in figure 1 below.

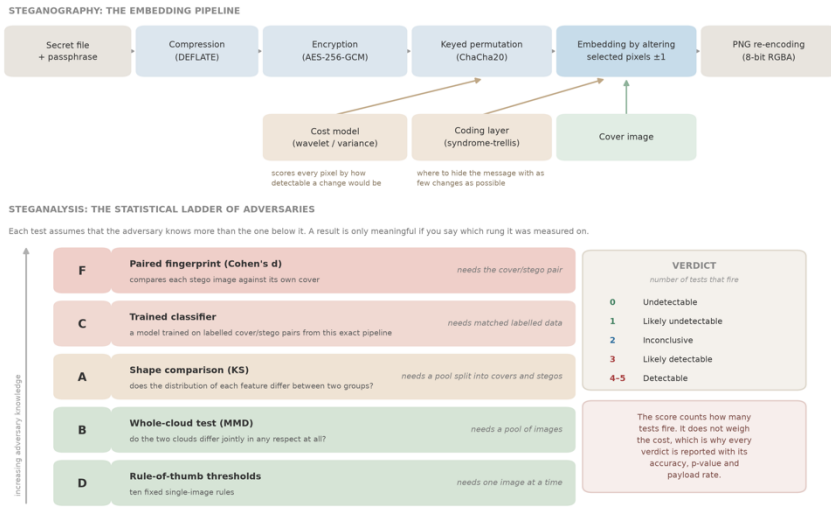

Figure 1. Stegosaurus: from a secret message to a verdict.

The embedding pipeline (top) and the statistical tests (bottom) are the tools behind every result in the study. There are six analyses in total, of which five (A, B, C, D, F) contribute to the verdict and E is a sanity check and therefore not shown in figure 1. The five tests are not five attempts at the same question. Instead, they form a ladder of adversaries, each assuming more knowledge than the one below it, from an adversary that has intercepted one suspicious image to a paired comparison that needs the original cover and is therefore only available to the researcher. A result is only meaningful if the step it was measured on is named.

The images were analyzed and reduced to a feature vector with 17 raw dimensions spanning nine detector families. Two of the dimensions, “LSB complexity” and “LSB adjacency match”, are inversions of each other and

“Overall score” is a deterministic function of the others. There are thus 15 non-redundant dimensions. All of these boil down to the six tests that are applied to the stego images, each test (except the sanity test) corresponding to a simulated adversary with specific capabilities. Any test which fires contributes one point to a verdict score.

Test A, the shape-comparison test (Kolmogorov–Smirnov). For each of the features, this test asks whether the distribution of values in the covers differs from the distribution in the stego images. The KS statistic D is the point of maximum disagreement about what fraction of images score below a given value. Because seventeen raw features are tested at once, some of them will look significant by luck alone. The Benjamini–Hochberg procedure (Benjamini & Hochberg, 1995) tries to control this by fixing the expected number of false findings among the detections made.

Test B, the whole-cloud test (Maximum Mean Discrepancy). Rather than examining the features one at a time, MMD (Gretton et al., 2012) asks whether the two clouds of points differ jointly, in any respect at all. It maps every image into a space of smooth functions and measures the distance between the average positions of the two clouds. Its significance is judged by a permutation test, in which the labels are shuffled at random hundreds of times, the statistic is recomputed each time, and the real value is compared against the resulting null distribution. If shuffled labels reproduce the observed separation as often as not, there is nothing there. This is the strongest test that a realistic adversary as per the definitions in this study can mount, since it needs only a pool of images collected from the sender.

Test C, the trained-informant test (supervised classifier). A logistic regression, i.e., a model that learns a weighted combination of the fifteen features and outputs a suspicion score, is trained on labeled examples. It has a penalty on large weights, so that it does not fit accidental quirks of individual images. It is evaluated by leave-one-out cross-validation, in which each image in turn is taken out, the model is trained on all the others, and the taken-out image is then classified. This maximizes the training data available at each step, and it makes sure that none of the images is ever scored by a model that has seen it before. Performance is reported three ways, as accuracy, as the balanced error P_E, and as AUC, which is the probability that a randomly chosen stego image is scored more suspicious than a randomly chosen cover (0.5 is chance and 1.0 is perfect). A permutation null establishes what accuracy is achievable by luck at this sample size, and a point is scored only if the observation beats both the 95th percentile of that null and a permutation p-value threshold. This test is extra relevant for the study with regard to the effect of noise profiles, which is central to the comparisons of different cover sources in the study.

Test D, the rule-of-thumb test (blind thresholds). Ten fixed single-image rules are applied with no training and no population reference. For each of them the true-positive rate, the false-positive rate, and their difference, which is called the lift, are computed. A detector that flags stego images and covers equally often has zero lift and is therefore useless. This is the most practically important test, since it demands nothing of the adversary. However, it is questionable whether such a system would be deployable for monitoring a large stream of traffic, since it is such a blunt tool.

Test E, the sanity check. The MMD test is run on two halves of the cover population, which should not differ. If it fires, the corpus is inhomogeneous and

no other result from that arm can be trusted. Every corpus in the study must pass this test.

Test F, the paired-fingerprint test (researcher-only). For each pair, the difference between the feature values of the stego image and those of its own cover is computed and tested for a consistent shift. This is summarized by Cohen's d, i.e., the mean shift divided by the standard deviation of the shifts. In other words, it says how large the change is relative to how much the change itself varies. If the adversary has access to both the original cover image and a suspected stego image, this test can check if a fingerprint exists and whether the image is different from the original. This does not automatically mean that one of the images contains steganography. It could have been processed for other perfectly normal reasons. As per the premises of this study, this test can never be exploited in deployment, because the cover image is destroyed directly after embedding the payload and creating the stego image. If Stegosaurus is used for image generation, the whole process stays in memory and only the stego image is ever written to the local hard drive.

The number of tests that fire determines to the verdict scoring on a scale from 0 to 5: undetectable (0), likely undetectable (1), inconclusive (2), likely detectable (3), and detectable (4-5). If test F is the only detector to fire, the verdict is likely undetectable, since the fingerprint is real but unexploitable as the cover image has been destroyed.

Accuracy, P_E, and AUC are reported together, because they answer different questions and can disagree. It should also be noted that a balanced error rate is not the operating point that a real adversary uses. Forensic steganalysis is evaluated at very low false-positive rates, because someone monitoring a large channel cannot accept the number of false alarms that a balanced threshold could result in (Cogranne et al., 2019). Hence, the verdicts used here are conservative.

3.4 WATERMARKS

To evaluate the resilience of provenance watermarks against steganographic embedding, three families were tested with different payload sizes.

DWT family. A reader was implemented and cross-validated bit-for-bit against the reference Python implementation over 8,832 decoded bits: dwtDctSvd matched exactly, dwtDct diverged on 0.18% of bits because of irreducible floating-point ties, but this changes no verdicts. The full matrix was 6 images × 2 methods × 3 payload ratios × 2 modes = 72 files. Each of the files was decoded by the reference decoder instead of this project's own decoder.

TrustMark. 6 images × 4 payload ratios × 2 modes = 48 files. The raw 100-bit vector was measured before error correction, to make sure that the underlying signal was not degraded while the pass or fail measure looked perfect. As controls, watermark round-trips were performed on all six clean marked images, with zero false positives on unmarked covers.

SynthID. To investigate whether the SynthID watermark can coexist alongside the steganographic payload if a file is embedded in an image with a pre-existing SynthID, 162 different images were generated using OpenAI's API and confirmed to contain SynthID using OpenAI's own verifier service. For each of the images, payloads of five different sizes were then prepared at 5%, 25%, 50%, 75%, and 90% of the total embedding capacity. The payloads were embedded

using Stegosaurus and each of the 162 cover images × 5 payload ratios = 810 stego images was fed to OpenAI's provenance verifier endpoint to determine whether they still contained a valid SynthID watermark. OpenAI's verifier was chosen, as Google's verifier is only available to select researchers.

Both C2PA and SynthID were reported as separate data points. C2PA is intended to break on any trivial alteration or processing that a photo goes through in transit, and it therefore irrelevant to this study. However, the loss of C2PA during processing was confirmed.

For all families, random data was used as payloads. The embedder compresses before encrypting, so the data needed to be incompressible to accurately reflect payload fill rate.

3.5 CORPUS GENERATION

A key constraint of the experiment was that the hardware and software had to be readily available, relatively cheap, and reasonably easy to obtain and use. For the experimental proof of concept, the end user was expected to have some computer literacy, such as a basic understanding of file systems, how to install programs, as well as rudimentary command line skills. However, in a future polished version of the software, the command line skills would no longer be needed.

The steganography software that was created for the experiment needs few resources and runs on almost any computer made during the last two decades. However, due to the resource intensive nature of diffusion-based AI, image generation necessitated much higher system requirements. For this purpose, and to satisfy the requirement for cheap and readily available hardware, two different setups were selected: a portable Apple MacBook based setup as well as an Intel desktop PC setup running Linux. Linux was selected instead of Microsoft Windows because it is free and comes in a variety of versions that can be downloaded almost anywhere in the World through local mirrors, even in places where the internet is heavily restricted.

The portable rig consisted of a six-year-old Apple MacBook Air M1 equipped with 16 GB memory. This model was exceedingly popular and is abundant and low cost on the second-hand market. At the time of the experiment, the local second-hand price was approximately CNY 2500.

The desktop computer was built from second hand server parts and an older, but sufficient, graphics card. Intel server CPUs and motherboards from the mid-2010's are available in such quantities on the second-hand market that the prices start as low as CNY 250 for both CPU and motherboard. For the build used in this experiment, an Intel Xeon 2696v4 CPU with a suitable motherboard and fan were bought in the Chinese app Taobao for CNY 800. An NVIDIA RTX 2060 12 GB graphics card was acquired from the same app for CNY 1100. The rest of the parts, which included a case, display, fans, keyboard, mouse, PSU, RAM, and SSD, were all salvaged from a nearly two-decade-old computer. If bought second hand, all of these parts, could be acquired for a price that would place the total cost of the build at a similar price as the Apple MacBook Air.

To generate images locally, free software and free models were used. The criteria for choosing the model were that it must be: (a) able to run locally; (b) lightweight enough to run on older hardware; (c) open-weight and free of charge; (d) easy to use without any complicated or paid software; (e) as modern as possible; (f) capable of producing images of at least 768x768 pixels on the chosen

hardware; (g) without watermarking, or with a way to disable it; (h) available for download from local repositories in multiple larger regions including China, EU, and the USA.

Black Forest Labs' FLUX.2 Klein 4B model fit the criteria well. Furthermore, just in time for the experiment, the lab released a distilled VAE decoder, called FLUX.2-small-decoder, which reduced memory requirements even more than the standard decoder and made the model match the criteria even better.

The open-source C++ library "stable-diffusion.cpp" for image generation was incorporated in Stegosaurus and modified to work well with the tested models on all the test computers.

3.6 STEGOSAURUS

The main tool used throughout the study for cover image generation, steganographic embedding, and steganalysis was Stegosaurus. This tool was specifically designed for the study and was implemented as a cross-platform command-line tool written in Go as well as a graphical interface application for macOS written in Swift. The diffusion feature used a subset of the C/C++ library stable-diffusion.cpp with Go bindings added. Go was chosen because of performance, concurrency, cross-platform features, and the ability to build single binaries without dependencies.

For the GUI application, macOS was selected as Apple-Silicon-equipped MacBooks were the cheapest and most abundant second-hand computers capable of doing local image generation at the time the study was conducted. The GUI was implemented using Swift on top of the Go Stegosaurus core.

3.7 REPRODUCIBILITY

All randomness is seeded. Intermediate results are checkpointed and permutations are built incrementally to survive interruption. Feature extraction is cached so that statistical parameters can be varied without re-computation. This was implemented partly because of the need to pause and resume long runs, and partly for reproducibility. Every experiment that was executed retains its full set of images, analyses, configuration files, and logs so that they can be checked for correctness or re-run at a later date.

4. RESULTS

4.1 LOCAL GENERATION IN PRACTICE

Local diffusion image generation was tested on three different computers, three open-weight image models, four model sizes, and five model quantization levels. A stated goal was to use affordable consumer hardware that was powerful enough to handle image generation of smaller images of decent quality. The quality did not have to match commercial offerings such as Google's Gemini or OpenAI's ChatGPT. Dimensions must be 768x768 pixels or more.

As per the requirements that the hardware must be relatively cheap and abundant and still be able to run the diffusion models, all of the models were tested on the previously selected Apple MacBook Air 2020 (16 GB RAM) and Intel Xeon-based desktop computer (32 GB RAM and 12 GB VRAM) built out of cheap, obsolete server parts from Chinese marketplace Taobao (淘宝). Additionally, a third computer, an Apple Mac Studio 2022 (32 GB RAM) was also tested. All three were able to run the models and produce images according to the specifications above within a reasonable time, ranging from 30 seconds to 5 minutes per image depending on model, resolution, and sampling steps. The cost in time scales with sampling steps, not model size, for the small models tested.

The 2,500 images FLUX.2 klein corpus used for the experiments was generated on the Xeon desktop computer using Stegosaurus command-line tool on Linux with the official FLUX.2 klein 4B safetensors set to 4 sampling steps.

Two model sizes at identical settings differed on none of six measurements across eight paired prompts. However, sampling step count made a big difference and every statistic moved when generating samples from 1 to 16 steps.

Considering the many factors involved in the diffusion generation pipeline, the measured bit-plane profile characterizes the pipeline and its settings, not the model. This is clear from comparing the 2,500 images study corpus to images generated with other FLUX.2 klein model sizes, weights, quantization, and settings. No generalizations can be made with the current sample sizes.

The full list of models that were successfully tested is: Stable Diffusion 1.5 (GGUF), Stable Diffusion XL (GGUF), FLUX.2 klein 9B (GGUF, Q4), and FLUX.2 klein 4B (GGUF, Q4).

4.2 COVER SOURCE DETECTABILITY

Eight full-scale experiments were conducted to establish the impact of the cover source on detectability. Each corpus consisted of 2,500 covers plus 2,500 corresponding stego images, i.e., 2,500 image pairs. Four genuinely different sources were used: FLUX.2 klein generated images (S3-S5), minimally processed RAW photos (S7), center-cropped JPEG photos (S8), and downsized JPEG photos (S6). Another two sources consisted of the FLUX.2 klein images processed by converting them to JPEG (S9) and by adding noise to mimic the RAW photos' noise profile (S10).

The first three experiments (S3, S4, S5), shared the same cover-source corpus, but varied in coding layer and payload. To answer the question of whether diffusion-generated images are a viable cover option, it had to be determined

how much of the detection results were attributable to the cover source itself, and how much to other factors.

All experiments passed the sanity check, confirming corpus homogeneity. Which tests fired, the verdict, and classifier accuracy can be found in the table below.

Table 3

Stego Image Detectability Based on Cover Source

Run	Coder	Cover	Fill %	A	B	C	D	F	Total	Verdict	Accu.	p
S3	LSB	FLUX	0.43	0	1	1	0	1	3/5	Likely detectable	0.530	0.025
S4	STC	FLUX	0.43	0	0	1	0	1	2/5	Inconclusive	0.523	0.040
S5	STC	FLUX	4.44	0	1	1	0	1	3/5	Likely detectable	0.604	0.005
S6	STC	photos, resized	0.43	0	0	1	0	1	2/5	Inconclusive	0.632	0.005
S7	STC	camera RAW	0.43	0	0	0	0	1	1/5	Likely undetectable	0.504	0.730
S8	STC	photos, cropped	0.43	0	0	1	0	1	2/5	Inconclusive	0.597	0.005
S9	STC	FLUX, JPEG round trip	0.43	0	0	1	0	1	2/5	Inconclusive	0.530	0.020
S10	STC	FLUX + read noise	0.43	0	0	1	0	1	2/5	Inconclusive	0.743	0.005

Note that the classifier accuracy is sign-corrected, as some raw accuracies fell below 0.5, meaning that the classifier was predicting labels backwards. An adversary observing this would not discard the classifier result, but simply invert the output. Consequently, the permutation null must be folded identically to raise the threshold appropriately. Without this correction, an almost perfect discrimination was reported as no evidence at all.

The total score and verdict should be seen as an indicator instead of true detectability. The score counts how many independent tests fire, without accounting for how complicated each test would be for an adversary to accomplish. Only taken together with the accuracy and p-value do the score and verdict paint a truthful picture. For example, the four arms scoring 2 of 5 at the same fill rate, have a wide spread of accuracies from 0.523 to 0.632.

To further illustrate this, see figure 2 below. As panel A shows, for an adversary holding only a pool of images, none of the available tests fire. Every cover source scores zero on the per-feature test, joint distribution test, and blind detectors. With such a small payload, the cover source is irrelevant to that specific adversary. In panel B, which shows an advanced adversary with matched and labeled training data from this exact pipeline, the cover source does matter, with camera RAW being the only source that remains equal to pure chance. The difference between the panels, shows that the detectability verdict is a statement about the assumed adversary, not a property of steganography itself.

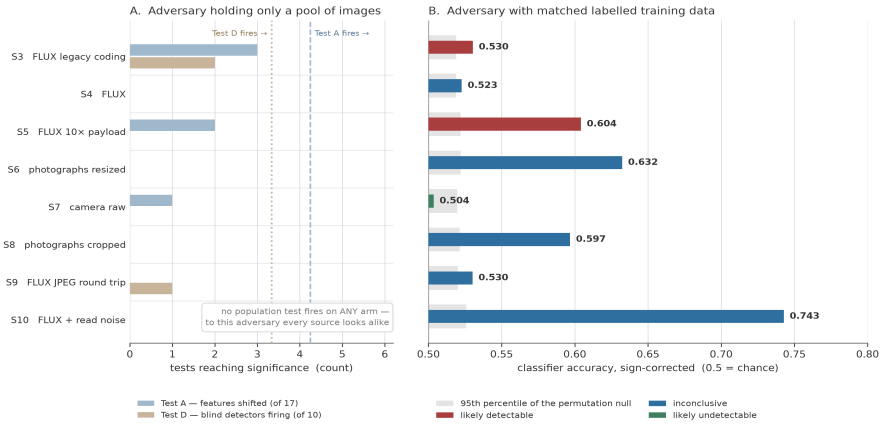

Figure 2. Different assumed adversaries lead to different detection outcomes.

Some caution is warranted when interpreting panel B, because the accuracy does not always agree with the paired feature shift. An experiment with little real signal is also the one that inflates most when subsampled, because a classifier with nothing real to fit, will fit noise instead. An accuracy figure should therefore be quoted alongside the amount of evidence standing behind it. This is the case with S10 which has the highest classifier accuracy in the study combined with the smallest paired feature shift, while S3 with a low classifier accuracy has the largest number of features shifted.

Six experiments have essentially identical fill and identical coding layer, differing only in the source of the images. FLUX.2 klein generated images, photographs resized, photographs cropped, minimally processed RAW photos, FLUX.2 klein images converted to JPEG, and FLUX.2 klein images with added noise. The population-level tests agree across all of these sources. None produces a significant per-feature shift or joint-distribution shift, no matter whether the covers are diffusion-generated, JPEG photographs, minimally processed camera RAW, or re-encoded diffusion images. As per figure 2 above, at this fill rate, the cover source is irrelevant for an adversary with only a pool of images.

The RAW photos corpus is the only cover source in this study on which the trained classifier fails, and it fails completely with an accuracy of 0.504 at p = 0.730, AUC of 0.5009. This is very close to pure chance.

4.3 GENERATED NOISE PROFILE

The intuitive premise that led to the second research question, whether latent diffusion-generated images have more random noise and are therefore better suited for least significant bit steganography than real photos, was quickly answered. It turned out to be the other way around. The FLUX.2 klein cover image corpus used in the study had a particularly structured, non-random noise profile.

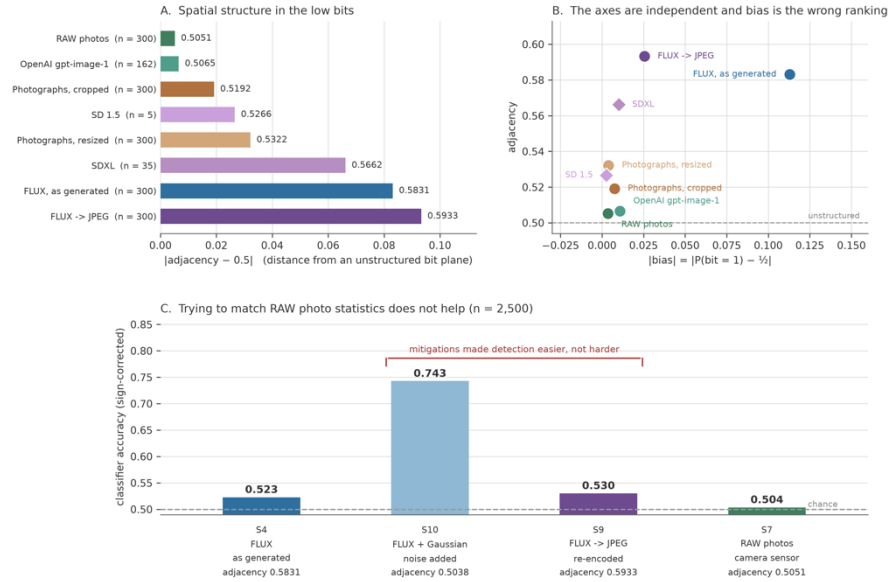

Figure 3. Diffusion covers are less random than real photos and randomness does not predict detection.

FLUX.2 klein sits furthest from an unstructured bit plane (panel A), yet bias and adjacency are independent: Stable Diffusion XL carries an order of magnitude less bias than FLUX.2 klein on almost the same spatial structure (panel B), so any claim about randomness must say which axis it means, and the axis that matters is adjacency, since that is what displacement tracks.

It is important to note that while the bias can identify a specific image generator, it cannot tell whether an image contains an embedded payload. The bias is a fingerprint of the model's decoder and identifies the image as being generated. Corvi et al. (2023a, 2023b) has shown that diffusion decoders produce artefacts in both the frequency domain and the remaining noise that can identify an image a synthetic and say which model was used to generate it. The measured bias in figure 3 above is a low-bit plane representation of this.

Panel C is the finding that matters most for practice. Four interventions tried to mimic the RAW photos' statistics, adding read noise, swapping to a generator that already produces them, sampling four times longer, and a JPEG round trip, and none produced a safer cover. Counterintuitively, adding noise made S10 reach the unstructured floor and become the most detectable full-scale run in the study. Only the minimally processed RAW photos, which reach these statistics through sensor physics rather than processing, are undetectable.

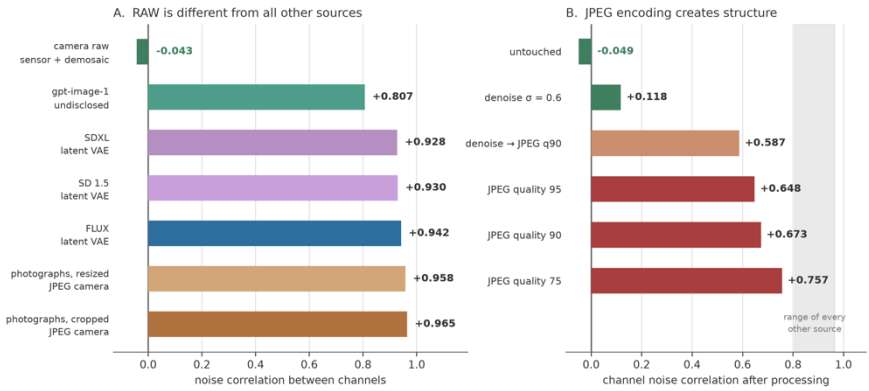

Figure 4. RAW photos have the least structure, and that cannot be mimicked

Only unprocessed sensor output has negative inter-channel noise correlation. Every other source, generated or captured, is strongly positive. It cannot be the physical camera sensor that is the reason for the RAW photo's noise profile. Both JPEG photograph corpora, from the same sensors and the same Bayer physics as the RAW corpus, are more correlated than every latent-diffusion model. The property must therefore be unprocessed versus processed, not captured versus generated. This is supported by re-encoding the 2,500 RAW photos to visually lossless JPEGs, which moved the camera RAW photos into the range of the others. This has interesting implications, since it means that a minimally processed RAW photo is only ideal as long as it stays a PNG.

4.4 PAYLOAD RATE AS A BETTER PREDICTOR

The single most reliable metric in this study comes from stratifying the large payload experiment, S5, by how much data each image actually carries. Its payload pool spans a wide range from 0.4 KB to 105 KB. The mean is 4.44% while the median is only 1.01%. 50% of the images are below 0.5% fill rate and 30% of the images above 5%. This is analytically valuable, since one study spans the whole payload range with every other variable held constant (the same covers, coding layer, features, and classifier training pool). The fill rates have been stratified into four bands, and each band is based on at least 236 pairs.

Table 4

Payload Size as a Predictor of Stego Image Detectability

Fill	*n*	Mean fill	Shape-shift *D*	Mean |Cohen's *d*|	Classifier AUC
< 0.5%	1,243	0.27%	0.032	0.074	0.509 (chance)
0.5–1.5%	236	1.01%	0.131	0.511	0.563
> 5%	759	13.11%	0.779	0.547	0.939
1.5–5%	262	2.18%	0.214	0.847	0.642

As seen previously, low fill detectability is near chance. However, at a high fill rate, the detectability is nearly perfect. The curve is continuous and detectability rises steeply across this range. The meaningful discrimination lies between 0.5% and 1.5% fill. The standardized effect size is overlaid on the same axis and shows

that mean |Cohen's d| peaks at around 2% and then falls. Counterintuitively, this makes it look as if detectability peaks near 2% of capacity and then declines. This is an artifact because the raw feature shift grows almost eight-fold while its standard deviation grows 14-fold, because the top band spans 5% to 41.7% fill and the shift varies greatly across it. Cohen's *d* divides by exactly that variability. AUC is normalized and shows that the detectability keeps increasing with the payload fill rate.

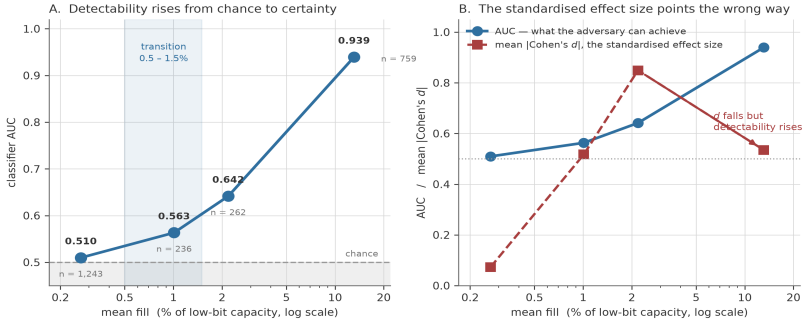

Figure 5. The size of the payload decides whether the payload is detected

The same data also demonstrate why pooling payloads of widely different sizes can result in understated or overstated detectability. Pooled, the AUC is 0.658, while the high fill images have an AUC of 0.939 on their own. The low fill images dilute the signal. For this reason, only the large payload experiment used a mixed-sized payload, and did so specifically for the stratification analysis. The dilution is a property of the evaluation design, not of the steganography, and it is invisible unless the payload distribution is reported alongside the verdict.

4.5 THE IMPACT OF THE CODING LAYER

Because of the hypothesis that diffusion-generated images would have an almost random lowest-bit plane, a coding layer experiment was added to the study. The idea was that if the plane is truly random, the difference in detectability between naïve least significant bit replacement and embedding that takes cost into account, would shrink. The premise turned out to be wrong, but the coding layer impact results are still interesting on their own.

Two experiments, S3 and S4, embedded the same payload pool into the same 2,500 FLUX.2 klein cover images at the same mean fill. Of these, 303 received an identical payload. They differed only in the coding layer. As per figure 6 below, the naïve LSB engine embedded the payloads almost perfectly in the average texture of the images, i.e., a random distribution that did not take the underlying images into account. The STC engine, on the other hand, embedded in pixels far above the mean texture, i.e., in highly detailed areas of the images.

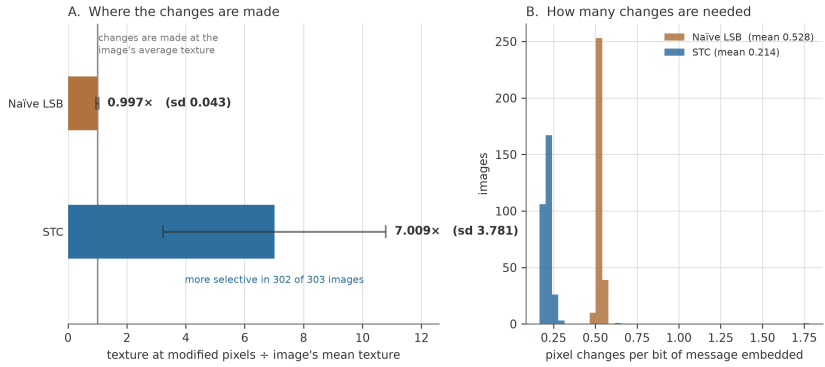

Figure 6. The STC coding layer hides the payload in the busiest parts of the image

What difference did embedding in random versus detailed areas make to detectability? The table below paints a stark picture. Most importantly, the STC engine made the embedding invisible to the blind detectors and kept the structure intact enough to not change any of the measured features significantly. Practically, this means that an adversary having collected only a pool of images, is unable to detect anything. Only an adversary who can train a classifier on labeled examples from this exact pipeline would still be able to detect the embedded payload. Hence, this computationally cheap coding layer improves security against the blind/population-level adversaries tested here at this payload rate.

Table 5

Naïve LSB embedding vs STC cost model and coding layer embedding

Quantity	Naïve LSB (S3)	STC (S4)	Change
Verdict	Likely detectable (3/5)	Inconclusive (2/5)	One band better
Features significant after correction	3 of 17	0 of 17	Eliminated
Joint-distribution test p	0.005	1.000	Fires vs. cannot
Classifier accuracy	0.530 (p = 0.025)	0.523 (p = 0.040)	Marginally lower
Blind detectors firing	2 of 10	0 of 10	Eliminated
Mean paired |Cohen's d|	0.494	0.068	7.2× smaller
Modifications per image	3,826	1,583	≈2.5× fewer
Texture at modified pixels ÷ image mean	0.997	7.01	7× more selective

An interesting detail is that one blind detector fires backwards in the naïve LSB experiment. The reason is that FLUX.2 klein generated images are created with an imbalance that the Sample Pairs Analysis finds suspicious. It is not the stego images that are flagged, but the cover image. The naïve LSB embedding restores the balance of the plane to what the SPA expects a clean image to look like. The effect disappears when the STC engine is used, because ±1 pixel matching does not overwrite the bit plane.

4.6 WATERMARKS ARE CONSERVED

Even though the study focuses on locally generated images, there are situations in which online image generation services could be an option, such as higher

quality images and no need for specialized hardware or software. These images come with digital watermarks that prove their origin as AI-generated. If steganography destroys such watermarks, it would be a clear sign that the image structure has been altered somehow and raise suspicion. However, if watermarks are preserved, an adversary might think that an intact watermark proves that the image has not been altered.

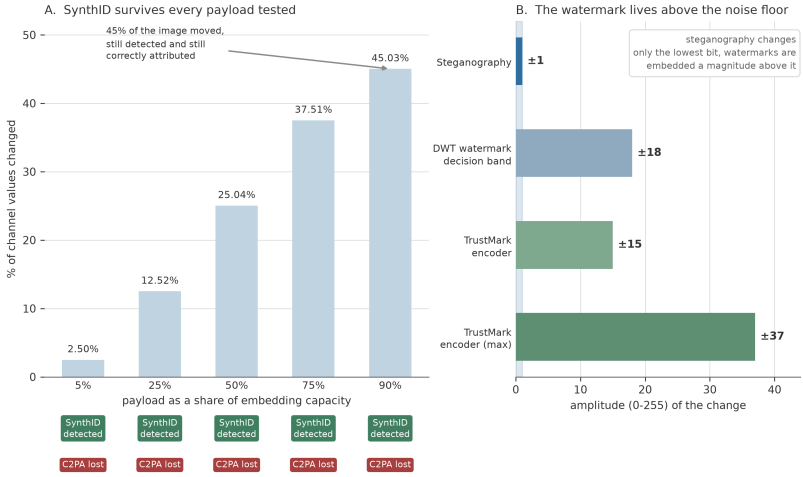

Figure 7. Steganography does not destroy watermarks

The experiments show that embedding a payload in a watermarked image does not erase the provenance watermark. SynthID and TrustMark were correctly detected and attributed at every payload tested up to 90% capacity. The Open DWT family fared the worst, with damage to the structure starting at around 30% of capacity, which is still far above any reasonable embedding rate. C2PA could not be preserved, as it signs a hash of the image and is destroyed by any alteration or processing.

Open DWT has multiple modes, which influence its robustness. The robustness is structural and built on a bit majority vote. A fifth of all blocks are individually vulnerable to damage, but only if a majority of 67 bits change simultaneously and in the same direction will a bit flip. At 18% of capacity, not a single bit was lost in any of the 72 test images.

Hardening eliminates the stronger variant's (dwtDctSvd) damage entirely and holds total damage for dwtDct flat at one bit regardless of payload, while the unprotected mode degrades from 0 to 4 to 15 bits as payload rises from 5% to 50%. However, hardening is not free and at 5% payload it costs one bit that the unprotected mode did not lose. For payloads below approximately 25% capacity it is unnecessary to use hardening to preserve the watermark.

Across the full 72-file matrix decoded by the reference implementation:

Table 6

Open DWT Watermark Bit-Level Robustness to Steganographic Embedding

Mode	Method	Files	Total bits changed	Worst file
off	dwtDct	18	5	3
off	dwtDctSvd	18	14	12
harden	dwtDct	18	3	1
harden	dwtDctSvd	18	0	0

All 810 SynthID detections succeeded, all 810 were correctly attributed to OpenAI, all 810 payloads were recovered, and all 810 C2PA manifests were lost. For clarity, only a single image with five different payload sizes is shown in figure 6 and the table below.

At 90% of capacity approximately 2.1 million, or 45%, of the channel values moved, and the watermark was still read correctly and attributed. The maximum change of one level and the near-zero mean explain why, and the amplitude argument established for TrustMark transfers directly.

Table 7

SynthID Robustness to Steganographic Embedding of Different Payload Sizes

Ratio	Payload	Channel values changed	SynthID	C2PA
5%	29,489 B	2.50%	detected	lost
25%	147,448 B	12.52%	detected	lost
50%	294,896 B	25.04%	detected	lost
75%	442,344 B	37.51%	detected	lost
90%	530,812 B	45.03%	detected	lost

The concern that adaptive embedding would place its changes precisely in the textured regions a neural decoder reads most heavily did not happen for either watermark. C2PA loss is not a failure, but a direct consequence of how it is calculated and attached.

4.7 STEGOSAURUS GRAPHICAL TOOL

The graphical application, as well as its command line interface companion, is feature-complete. In fact, all image generation, steganographic embedding and extraction, steganalysis, watermark testing (except SynthID), and experiment batch runs in this study were performed using Stegosaurus. While no end-users have tested the software yet, it has been thoroughly vetted and pushed during several months. The educational visualization feature of the graphical interface, meant to give a hands-on understanding of how embedding happens, helped in finding a bug in the coding layer implementation, since it clearly visualized where the payload had been embedded and that it did not agree with the expected pattern of the STC engine.

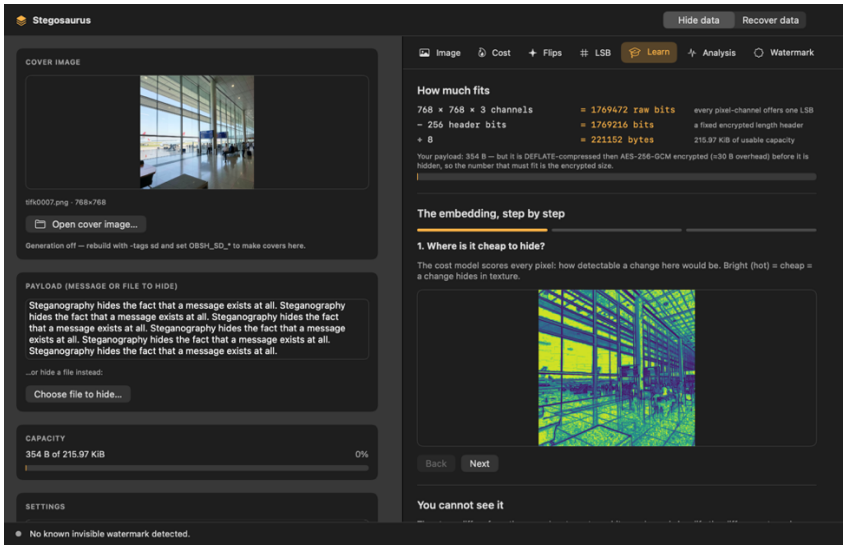

Figure 8. Stegosaurus graphical interface

5. DISCUSSION

5.1 CAMERA SENSOR NOISE

Diffusion image generation is computationally and memory intensive. Even small, open-weight models have hardware requirements that exclude much of consumer hardware. This severely limits the options available to generate cover images locally without leaving any traces online. However, the computer hardware constraint may not be the true limitation.

Before the study was conducted, it was theorized that randomness in the least significant bit plane was the key to making embedded payloads less detectable, and that generated images, being created out of random noise, would be the best suited for this. The first assumption turned out to be true, while the second was not, at least for the family of diffusion generators used in this study.

The large image corpus created locally with FLUX.2 klein 4B had the least random least significant bit plane of all the sources that were tested in the two measures that mattered: bias and spatial adjacency. On the opposite side of the spectrum, at the randomness ceiling, sat the minimally processed RAW file from an older Nikon DSLR camera. The RAW photos corpus proved ideal, with the matched classifier failing completely, and was therefore used as the benchmark in the study.

While the FLUX.2 klein, Stable Diffusion 1.5, and Stable Diffusion XL fared poorly in this metric, OpenAI's image generator produced images with what looked like an almost perfectly random noise profile, getting close to the minimally processed RAW photos.

Initially, this seemed both promising, because the OpenAI images could potentially hide payloads well, and discouraging, because the image generation happens online and is a paid service. However, when a small batch of OpenAI images was tested, they scored poorly as well. The steganalysis resulted in a near-perfect (AUC 0.997) separation of cover and stego images. Note that this result should be taken with a grain of salt because of the small sample size (162 images). Tests performed on equally small sample sizes excerpted from the other, larger corpora, suggest that the OpenAI images would perform similarly relative to the other generated cover sources. For example, subsampling the larger corpus S4 to that size gives an AUC of 0.86 for a true 0.52, while a subsampled S10 gives 0.98 for a true 0.74.

A number of attempts were made to alter the noise profiles of the different generated images to match that of the RAW photos. This included adding Gaussian noise and modifying the generation pipeline in stable-diffusion.cpp. These mitigations got the noise randomness, adjacency, and bias close to the RAW photos' values with minimal visual degradation of the images. Nevertheless, they did not make detection harder as predicted, but easier.

Only one factor could be found that clearly differed between the RAW photos and all other images in the study, both generated and processed photos. The difference was in the structure, or rather lack of structure. The randomness of the RAW photos was channel-independent and hence the noise was independently unstructured in a way that was neither naturally present in any

other source nor possible to achieve with mitigations without completely destroying the image.

It turns out that a random, but structured, plane is bad. The steganalysis used in the study detects where structure has been damaged. Conversely, this means that if there is little or no structure to destroy, the detectors have nothing to measure.

The exact reason why the minimally processed RAW photos are random and unstructured while the other sources are not, is outside the scope of the study, but a potential explanation worth exploring further is that it derives from the physics of the camera sensor. In the sensor, each color is captured separately using a color filter that excludes other colors. Therefore, each of the three colors that make up each pixel potentially pick up different noise from the messy real-world environment. When minimally processed, some of this channel individuality survives. If processed in any other way, even processing that is visually lossless such as saving the photo as a high-quality JPEG, the channel-independent noise collapses into strongly correlated joint values.

Does this mean that RAW-photo-like, channel-independent noise cannot be produced using latent diffusion image generation? Not by altering the available pipeline of the studied open-weight models or by applying post-generation mitigations at least. Also, likely not by the model quantization process. This does not mean that the joint channel output is inherent to latent diffusion, and whether it can be achieved is an interesting area to explore in future research.

This result connects to the previously mentioned problem of cover-source mismatch. Giboulot et al. (2020) show that a steganalyser trained on images from one cover source performs much worse on images from a different source. Méreur et al. (2024) ask specifically whether generated images could be used as such a mismatched source and thereby make steganalysis less effective.

This study suggests a more nuanced answer. A cover-source mismatch is only useful when the gap between the steganalyser's training data and the intercepted stego images are found in features that the detector actually relies on to identify a stego image. However, the image statistics that can be controlled during image-generation do not seem to be the ones that the steganalyser depends on. The only property that clearly differentiates the sources in this study is something that the diffusion generator is unable to reproduce and that cannot be added afterwards. Cover images generated may thus be different from the adversary's training data in a number of ways, but not necessarily in any way that matters in order to fool the detector.

5.2 MASS GENERATION OF COVER IMAGES

The local image-generation tests on three different computers with different diffusion models, image resolutions, and step counts show that it is possible to generate local cover images that are usable for steganography. While no benchmarks were recorded, it is clear that the cost was dominated by diffusion step count rather than model capacity and that available memory put a limit to which models can be used and which resolutions can be generated. However, nothing in this study suggests that the cost is too high to make it feasible for an individual to produce an arbitrary number of cover images as needed.

It can be argued that mass production of cover images can create a different security problem. Ker (2007) shows that the amount of data that can be securely hidden does not grow in proportion to the number of images produced, but

instead grows with the square root of the number of cover images. This means that doubling the number of images sent only increases the safe payload by approximately 1.4 times. An adversary who collects many images from one sender therefore becomes more effective. Dworetzky and Fridrich (2023) analyse why this pooling effect arises and what the consequences are in practice.

The ability to easily generate cover images can therefore be a disadvantage as well as an advantage. If generating covers is cheap, a user may be tempted to send more of them. At the same time, sending more images gives an adversary more images to analyze and while not providing a proportional increase in safe capacity. The main security risk of inexpensive generation could hence be the potential volume of communication, rather than any weakness in the diffusion-generated cover images themselves.

5.3 EMBEDDING RATE VS COVER SOURCE

What does the finding about unstructured randomness mean for local image generation as a source of steganography cover images? Is a minimally processed photo, from a camera that stores uncompressed RAW files, the only feasible option for least significant bit steganography?

The study suggests that embedding rate, and what tools the adversary is using, are more important than the cover source. A real-world adversary has to deal with real-world constraints. Someone using the detectors in this study on a pool of collected images, without labels or matched training sets, will not find the stego images. An adversary passively monitoring a busy channel for a range of threats is unlikely to use advanced detection systems for fringe technologies or to have a plethora of trained convolutional neural networks scanning every file. While advances in machine learning have made sophisticated, active scanning feasible in theory, a real-world adversary has to deal with network throughput, costs, and the burden of analyzing flagged files, whether they are true or false positives.

At low fill rates, the trained classifier is fooled. This is partly because adaptive encoding causes multiple signals to disappear, when compared to naïve LSB replacement, and the two blind detectors are therefore silenced. At embedding rates of below 0.5%, the trained classifier is not much better than random chance, but above 5% its accuracy rises dramatically. Hence, fill rate is a crucial aspect if the stego image is to go undetected. This is uncontroversial, as a small payload means fewer altered pixels than a larger payload, and Ker's square-root law (2007) tells us that security is expected at low rates. A more interesting question is where the transition lies. The internal stratification of the large payload test, indicates that this is between 0.5 and 1.5% embedding rate for the pipeline used in this study.

Except for the one test mentioned above, the embedding rate was kept low (mean 0.43%) in all the tests. At this fill rate, the payload should not be evident, and most of the detector spread between sources was not a detection signal. In the six study runs that tested different cover sources at the same embedding ratio and algorithm, one had one out of seventeen raw features with a real shift for cover and stego image pairs, while the other five had zero. If no feature has shifted, and the test still reports some ability to detect pairs, it suggests that it is merely a classifier with seventeen raw dimensions that is fitting label-correlated noise in 2,500 points.

For the pool-based population tests at ~0.43% fill there is no gradient of better and worse sources. Instead, there is a clear distinction between the minimally processed RAW photos with zero detection on one side of the tests, and all other cover sources, generated or natural, at near zero, on the other side.

5.4 WATERMARK SURVIVAL

The steganographic embedding did not destroy any of the tested watermarks. The main reason is that the embedding algorithms make much smaller changes in the images than the watermarking algorithms.

All three watermark families survived the embedding. The DWT watermark showed no bit errors at typical payloads and TrustMark showed no raw bit errors before error correction at up to 90% of its capacity. SynthID remained detectable and correctly attributed at all five tested payload ratios, even when embedding had altered 45% of the image's color values.

As stated above, the important difference between steganography and watermarking is the size of the changes made to an image. The steganographic embedder changes selected values by only ±1. TrustMark makes changes that are up to roughly 37 times larger, while DWT also uses a much larger effective amplitude plus repeated embedding to ensure robustness. This means that the watermark signal sits far above the small changes made by steganography and explains why a large payload can change many pixels without destroying any of the watermarks.

This result is consistent with previous research. Zhao et al. (2024) showed that invisible watermarks can be destroyed if an attack makes large enough changes to the image to reach the amplitude of the watermark and then uses a generative model to reconstruct the damaged image. The experiments in this study approached the problem from the opposite direction. As long as the steganographic changes made to the image remain low, the watermark remains unbroken.

This has three practical implications. First, it is encouraging for provenance because all of the tested watermarks survived this hidden modification of the images. Second, it means that destruction of or damages to watermarks are unlikely be useful in steganalysis, since the embedding of payloads many times larger than what is reasonable to use in real-world steganography did not be any measurable damage to the watermarks. Third, a different signal may potentially be somewhat more useful for steganalysis. Every stego image lost its C2PA manifest, while SynthID survived to identify the image as generated by OpenAI. An image that keeps one provenance signal but loses another could be seen as anomalous and a passive observer could use the official provenance checks to detect this inconsistency. However, it is very unlikely to be of any use, since it signals recompression, cropping, or any modification in general, not steganography specifically.

The SynthID watermark results cannot be used as evidence that there was no damage at all to the watermark. The verification API reports only detection and attribution, not whether there is some bit-level damage to the watermark. Furthermore, the steganographic embedding is made to change as little as possible and can therefore not be seen as an attack on the watermark. It is feasible to mount a real attack using different methods, and prior work has shown that this can be done (Jiang et al., 2023; Kassis and Hengartner, 2025).

What is relevant for this study, is to show that the watermarks tolerate small, non-targeted changes produced by least-significant bit steganography using cost models and coding layers.

It is also important to note that the embedded payload and the watermarks are opposites in robustness. This particular steganographic method depends on every pixel being intact to be able to extract the payload. There is no redundancy or error correction. Any re-encoding, resizing, or processing will destroy the payload. Since the payload is encrypted, no partial recovery is possible. The watermarks on the other hand are designed to survive as much processing as possible and still be able to prove provenance.

5.5 ADVERSARY MODEL LIMITATIONS

The main purpose of the study was to test the feasibility of generating local cover images for steganography on consumer hardware. As such, the assumed steganography user profile excludes high-profile targets. Likewise, the assumed adversary is passively monitoring a high-traffic channel that a wide variety of data passes through. The adversary does not know who or what they are looking for or if there is anything at all to detect. Therefore, advanced steganography and steganalysis based on deep-learning and convolutional neural networks were excluded. For targeted surveillance against a high-value individual or organization, it can be assumed that both sides have access to more sophisticated tools and know more specifically what they are trying to circumvent or detect. This significantly changes the stakes and issues to study. It is likely that a trained convolutional neural network could detect the payloads in this study at embedding rates the steganalysis tool used cannot.

5.6 PNG AND IMAGE DOMAINS

While in this study, all experiments used steganographic embedding in the spatial domain of lossless PNG images, this is not the only domain that can be used. The JPEG domain has its own algorithms and techniques for steganography and steganalysis, some of which are more powerful than those available for PNG images. For steganalysis, the reverse JPEG compatibility attack recovers evidence of embedding from the rounding behavior of the decompressed image (Butora & Fridrich, 2020), and incompatible-block analysis identifies blocks that no legitimate compression could have produced (Levecque et al., 2024). Both are more sensitive than any of the tests used in this study, but neither can be used with PNG carriers. An individual using steganography to embed into JPEG thus faces a stronger adversary given the same adversary model.

It is worth reflecting on this with regards to the S9 experiment, where the full FLUX.2 klein image corpus was converted to JPEG and then back again to assess the effect of JPEG processing on the noise profile. Here, the JPEG round trip happened before embedding and the carriers were delivered as a PNG images, which means that the compression artefacts were part of the covers, not of the carriers, so these types of attacks remain inapplicable.

As part of the development of the steganography and steganalysis tool Stegosaurus used in the study, JPEG embedding and analysis were added. JPEG is the de facto standard used on the internet, and nearly every image is converted

to a JPEG before being sent across a channel. If the image is already a JPEG, it is still likely to be recompressed and possibly resized to save bandwidth and storage space. This poses a challenge to steganography, as the payload must survive at least recompression. In Stegosaurus, two different JPEG embedding algorithms are used. One that survives heavy recompression, but where the payload is damaged by resizing, and another that survives both heavy recompression and resizing to a fraction of the original size. The former supports large payloads while the latter is limited to short messages.

As the JPEG side of the tool is fully functional and JPEG is ubiquitous on the internet, it could seem counterintuitive to focus only on PNG images. The primary reason is that the study theorized that diffusion-generated images, because of their origin as random noise, would have a noise profile that would lack structure in the lowest bits and therefore be well-suited to least significant bit PNG steganography where an unpredictable plane would be advantageous for embedding and disadvantageous for an adversary trying to detect payloads in images. As shown by the experiments involving the conversion of generated PNGs and RAW photos to JPEG, the block processing of JPEG images alters the structure. This would destroy the theorized PNG advantage of diffusion-based images.

5.7 ETHICAL AND SOCIETAL ASPECTS

With any research that deals with concealing data or detecting concealed data, which are both dual-use technologies, there are ethical concerns regarding potential usage. The technologies are opposites of each other, with the strengths and weaknesses of steganography being informative for steganalysis and vice versa. With this in mind, the study reported on both steganography and steganalysis with an equal emphasis on both evading detection as well as detecting embedded data in images.

In this study, no third-party networks or communications were monitored or had any traffic intercepted. No third-party detection systems were probed or attacked. All the steganographic and steganalysis experiments used locally generated covers with payloads consisting of random data or public domain texts that were embedded locally.

For the SynthID watermark resilience test, 162 images were created using OpenAI's API. After locally embedding random payloads of different sizes, the images were sent to OpenAI's public provenance verification API, which is intended for testing whether an image contains SynthID. The results of the watermark experiments, which showed that SynthID watermarks survives steganographic embedding, strengthens rather than undermines the SynthID provenance ecosystem that has now been widely adopted by Google, OpenAI, and others.

6. CONCLUSION

Local, diffusion-based generation of cover images on consumer hardware proved to be viable for the stated use case of creating disposable images. On an Apple MacBook Air 2020, an abundant and popular second-hand computer, a small open-weight diffusion model can generate an image of decent quality in a couple of minutes. While this is impractical for corpus-scale production on one machine, it is good enough for creating cover images. Patience may be needed, and depending on memory size, CPU, and GPU, the selection of available diffusion models could be limited. Conceivably the biggest constraint is that common local image generation tools require non-trivial technical knowledge to set up and run. This is one of the issues that the Stegosaurus tool developed in this study aims to address.

Disposability is where local generation shines. By destroying the cover, and being guaranteed that the cover has never been used or stored anywhere previously, the most sensitive statistical test available in this study is eliminated.

Perhaps the greatest surprise was that diffusion-generated covers give a less random canvas to work with, not a more random one as expected. Of the six sources profiled, FLUX.2 klein images had the least random low-bit plane. Minimally processed RAW photos had the most randomness and were the only source on which the strongest adversary failed. The mechanism is that the detectors measure the structure that steganographic embedding destroys, so an image having structure is a liability rather than extra headroom for embedding. It is important to note that the large bias could be specific to this pipeline rather than to latent diffusion generally. Moreover, the practical size of the cover source effect is only half of the effect of payload fill rate, and most of it seems to be noise-fitting.

The most useful finding is negative. The three low-bit summary statistics do not predict detectability in either direction, as four separate attempts to alter the cover noise profile demonstrate. Adding sensor-like noise, choosing a generator with camera-like output, and sampling for more diffusion steps all improved the statistics but at the same time made detection easier (or unchanged). A JPEG conversion round trip moved two of the statistics in opposite directions but did not change the outcome.

What actually distinguishes the best source, RAW photos, is that the noise is independent per color channel. This is likely a consequence of per-photosite physical color filtering in the sensor, which is destroyed by processing and can neither be restored afterwards nor added to an image with joint channel noise.

As alluded to earlier, detectability is decided mostly by embedding rate and by how advanced the adversary is assumed to be, far more so than by the source of the cover image. In the mixed-size payload experiment, the classifier moved from random chance below 0.5% fill to almost certainty above 5%, with the turning point between 0.5% and 1.5%. At a sufficiently low rate, i.e., less than 0.5%, an adversary that has collected a pool of images detects nothing on any source, while an adversary with a matched and labeled training data set detects something on every source except the RAW photos. It is important to make this split explicit. Detectability claims must state the adversary's assumed capabilities and the effect size, not just the verdict. Information security is relevant for everyone from

hobbyists to nation states, and the capabilities, resources, threats, and consequences are immensely different between the two.

A less surprising find was that steganographic embedding does not destroy AI-provenance watermarks. The open DWT watermark family remained decodable and usable despite some bit errors at higher payloads, with variant- and hardening-dependent damage. TrustMark showed zero pre-correction bit errors all the way up to 90% of capacity. SynthID was detected and correctly attributed at every ratio tested, including one where 45% of the channel values moved.

The reason is amplitude separation. Watermarks are written an order of magnitude above the ±1 channel-value/intensity-level perturbation that is used by least significant bit steganography. Watermarks are designed to be resilient, not undetectable.

The one exception was C2PA manifests, which are destroyed by any alteration of the image. This includes simple processing such as cropping and resizing, and a destroyed C2PA does not tell an adversary whether an image contains steganography or not, merely that it has gone through some kind of processing since it was originally watermarked.

As a part of the study, an end-to-end graphical tool integrating image generation, steganographic embedding and extraction, steganalysis, as well as watermark verification was built. Its pedagogical contribution is that it makes what's invisible and technical, such as embedding visualizations, cost heatmaps, flip maps, bit planes, and capacity calculations, visible and more accessible for anyone who is interested. It also lowers the threshold for experimentation with both steganography and steganalysis. Rather than being a prototype, the software is fully functional and ready for users to run and evaluate.

7. FUTURE WORK

The cover source statistics were measured across diverse pipelines that included both latent diffusion and real camera photos. However, FLUX.2 klein was the only generated cover source with a full-size corpus of 2,500 images. Generating multiple full-sized corpora was not feasible with the hardware, budget, and time constraints on the project. Adding more image generators, generating full-sized corpora with all of them, and repeating the study, would establish whether the results generalize or if there are cover source differences that only show up in a large enough corpus.

The pedagogical claims for Stegosaurus are based on the assumption that visualizations translate to insights and that graphical interfaces lower the technical threshold for users. Whether they actually improve understanding for students and non-technical users is an empirical question that can only be answered by a controlled study where comprehension is compared against a conventional command-line tool or suite of disparate tools that the users have to navigate.

Considering how effective the independent channel noise profile of the RAW photos was in thwarting the detectors, there are two ways in which it would be interesting to investigate it further. First of all, is the unstructured low-bit plane effective against a more advanced adversary, such as convolutional neural networks? Secondly, is it possible to create a camera-sensor-like output from a latent diffusion engine by changing something in the training process or generation pipeline? These are two very different non-trivial projects, but the outcomes of such studies could potentially unlock more resilient cover image generation.

REFERENCES

Alatawi, M., & Saxena, N. (2023). SoK: An analysis of end-to-end encryption and authentication ceremonies in secure messaging systems. In I. Boureanu, S. Schneider, B. Reaves, & N. O. Tippenhauer (Eds.), *Proceedings of the 16th ACM Conference on Security and Privacy in Wireless and Mobile Networks* (pp. 187–201). ACM. https://doi.org/10.1145/3558482.3581773

Aslam, M. A., Rashid, M., Azam, F., Abbas, M., Rasheed, Y., Alotaibi, S. S., & Anwar, N. W. (2022). Image steganography using least significant bit (LSB) - a systematic literature review. *Proceedings of the 2022 2nd International Conference on Computing and Information Technology (ICCIT)*, 32–38. https://doi.org/10.1109/iccit52419.2022.9711628

Benjamini, Y., & Hochberg, Y. (1995). Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society Series B*, 57(1), 289–300. https://doi.org/10.1111/j.2517-6161.1995.tb02031.x

Boroumand, M., Chen, M., & Fridrich, J. (2019). Deep residual network for steganalysis of digital images. *IEEE Transactions on Information Forensics and Security*, 14(5), 1181–1193. https://doi.org/10.1109/TIFS.2018.2871749

Bui, T., Agarwal, S., & Collomosse, J. (2025). Trustmark: Robust watermarking and watermark removal for arbitrary resolution images. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 18629–18639. https://doi.org/10.1109/ICCV51701.2025.01731

Butora, J., & Fridrich, J. (2020). Reverse JPEG compatibility attack. *IEEE Transactions on Information Forensics and Security*, 15, 1444–1454. https://doi.org/10.1109/TIFS.2019.2940904

Cachin, C. (2004). An information-theoretic model for steganography. *Information and Computation*, 192(1), 41–56. https://doi.org/10.1016/j.ic.2004.02.003

Cogranne, R., Giboulot, Q., & Bas, P. (2019). The ALASKA steganalysis challenge. *Proceedings of the ACM Workshop on Information Hiding and Multimedia Security*, 125–137. https://doi.org/10.1145/3335203.3335726

Corvi, R., Cozzolino, D., Poggi, G., Nagano, K., & Verdoliva, L. (2023a). Intriguing properties of synthetic images: From generative adversarial networks to diffusion models. *IEEE/CVF CVPR Workshops*, 973–982. https://doi.org/10.1109/CVPRW59228.2023.00104

Corvi, R., Cozzolino, D., Zingarini, G., Poggi, G., Nagano, K., & Verdoliva, L. (2023b). On the detection of synthetic images generated by diffusion models. *ICASSP 2023*, 1–5. https://doi.org/10.1109/ICASSP49357.2023.10095167

Dathathri, S., See, A., Ghaisas, S., Huang, P.-S., McAdam, R., Welbl, J., Bachani, V., Kaskasoli, A., Stanforth, R., Matejovicova, T., Hayes, J., Vyas, N., Al Merey, M., Brown-Cohen, J., Brunel, R., Balle, B., Cemgil, T., Ahmed, Z., Stacpoole, K., … Kohli, P. (2024). Scalable watermarking for identifying large language model outputs. *Nature*, 634, 818–823. https://doi.org/10.1038/s41586-024-08025-4

De La Croix, N. J., Ahmad, T., & Han, F. (2024). Comprehensive survey on image steganalysis using deep learning. *Array*, 22, 100353. https://doi.org/10.1016/j.array.2024.100353

Dumitrescu, S., Wu, X., & Wang, Z. (2003). Detection of LSB steganography via sample pair analysis. *IEEE Transactions on Signal Processing*, 51(7), 1995–2007. https://doi.org/10.1109/TSP.2003.812753

Dworetzky, E., & Fridrich, J. (2023). Explaining the bag gain in batch steganography. *IEEE Transactions on Information Forensics and Security*, 18, 3031–3043. https://doi.org/10.1109/tifs.2023.3267667

Filler, T., Judas, J., & Fridrich, J. (2011). Minimizing additive distortion in steganography using syndrome-trellis codes. *IEEE Transactions on Information Forensics and Security*, 6(3), 920–935. https://doi.org/10.1109/TIFS.2011.2134094

Fridrich, J., & Kodovsky, J. (2012). Rich models for steganalysis of digital images. *IEEE Transactions on Information Forensics and Security*, 7(3), 868–882. https://doi.org/10.1109/TIFS.2012.2190402

Fridrich, J., Goljan, M., & Du, R. (2001). Reliable detection of LSB steganography in color and grayscale images. *Proceedings of the 2001 Workshop on Multimedia and Security*, 27. https://doi.org/10.1145/1232454.1232466

Giboulot, Q., Cogranne, R., Borghys, D., & Bas, P. (2020). Effects and solutions of cover-source mismatch in image steganalysis. *Signal Processing: Image Communication*, 86, 115888. https://doi.org/10.1016/j.image.2020.115888

Gowal, S., Bunel, R., Stimberg, F., Stutz, D., Ortiz-Jimenez, G., Kouridi, C., Vecerik, M., Hayes, J., Rebuffi, S., Bernard, P., Gamble, C., Horváth, M. Z., Kaczmarczyck, F., Kaskasoli, A., Petrov, A., Shumailov, I., Thotakuri, M., Wiles, O., Yung, J., … Kohli, P. (2025). SynthID-image: Image watermarking at internet scale. *arXiv preprint*. https://doi.org/10.48550/arXiv.2510.09263

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., & Smola, A. (2012). A kernel two-sample test. *Journal of Machine Learning Research*, 13, 723–773. https://jmlr.org/papers/v13/gretton12a.html

Holub, V., Fridrich, J., & Denemark, T. (2014). Universal distortion function for steganography in an arbitrary domain. *EURASIP Journal on Information Security*, 2014(1). https://doi.org/10.1186/1687-417X-2014-1

Jiang, Z., Zhang, J., & Gong, N. Z. (2023). Evading watermark based detection of AI-generated content. *ACM SIGSAC CCS 2023*, 1168–1181. https://doi.org/10.1145/3576915.3623189

Kassis, A., & Hengartner, U. (2025). UnMarker: A universal attack on defensive image watermarking. *IEEE Symposium on Security and Privacy (SP)*, 2602–2620. https://doi.org/10.1109/SP61157.2025.00005

Ker, A. D. (2007). Batch steganography and pooled steganalysis. In J. L. Camenisch, C. S. Collberg, N. F. Johnson, P. Sallee (Eds), *Information Hiding: 8th International Workshop (IH 2006)* (pp. 265–281). https://doi.org/10.1007/978-3-540-74124-4_18

Ker, A. D., Bas, P., Böhme, R., Cogranne, R., Craver, S., Filler, T., Fridrich, J., & Pevný, T. (2013). Moving steganography and steganalysis from the laboratory into the real world. *Proceedings of the First ACM*

Workshop on Information Hiding and Multimedia Security, 45–58. https://doi.org/10.1145/2482513.2482965
Levecque, E., Butora, J., & Bas, P. (2024). Finding incompatible blocks for reliable JPEG steganalysis. *IEEE Transactions on Information Forensics and Security*, 19, 9467–9479. https://doi.org/10.1109/tifs.2024.3470650
Li, B., Wang, M., Huang, J., & Li, X. (2014). A new cost function for spatial image steganography. *IEEE International Conference on Image Processing (ICIP)*, 4206–4210. https://doi.org/10.1109/ICIP.2014.7025854
Mallet, A., Beneš, M., & Cogranne, R. (2024). Cover-source mismatch in steganalysis: Systematic review. *EURASIP Journal on Information Security*, 2024(1). https://doi.org/10.1186/s13635-024-00171-6
Méreur, A., Mallet, A., & Cogranne, R. (2024). Are deepfakes a game-changer in digital images steganography leveraging the cover-source-mismatch? *Proceedings of the 19th International Conference on Availability, Reliability and Security*, 1–9. https://doi.org/10.1145/3664476.3670893
Pevný, T., Bas, P., & Fridrich, J. (2010). Steganalysis by subtractive pixel adjacency matrix. *IEEE Transactions on Information Forensics and Security*, 5(2), 215–224. https://doi.org/10.1109/TIFS.2010.2045842
Qiao, T., Xu, S., Wang, S., Wu, X., Liu, B., Zheng, N., Xu, M., & Pan, B. (2023). Robust steganography in practical communication: A comparative study. *EURASIP Journal on Image and Video Processing*, 2023(1). https://doi.org/10.1186/s13640-023-00615-y
Rosenthol, L. (2022). C2PA: The world's first industry standard for content provenance. *Applications of Digital Image Processing XLV*, 12226, 122260P. https://doi.org/10.1117/12.2632021
Sedighi, V., Cogranne, R., & Fridrich, J. (2016). Content-adaptive steganography by minimizing statistical detectability. *IEEE Transactions on Information Forensics and Security*, 11(2), 221–234. https://doi.org/10.1109/TIFS.2015.2486744
Srivastava, M. (2019, May 13). *WhatsApp voice calls used to inject Israeli spyware on phones*. Financial Times. https://www.ft.com/content/4da1117e-756c-11e9-be7d-6d846537acab
Subramanian, N., Elharrouss, O., Al-Maadeed, S., & Bouridane, A. (2021). Image steganography: A review of the recent advances. IEEE Access, 9, 23409–23423. https://doi.org/10.1109/access.2021.3053998
Tao, J., Li, S., Zhang, X., & Wang, Z. (2019). Towards robust image steganography. *IEEE Transactions on Circuits and Systems for Video Technology,* 29(2), 594–600. https://doi.org/10.1109/tcsvt.2018.2881118
You, W., Zhang, H., & Zhao, X. (2021). A siamese CNN for image steganalysis. *IEEE Transactions on Information Forensics and Security*, 16, 291–306. https://doi.org/10.1109/TIFS.2020.3013204
Yousfi, Y., Butora, J., Fridrich, J., & Fuji Tsang, C. (2021). Improving EfficientNet for JPEG steganalysis. *Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security*, 149–157. https://doi.org/10.1145/3437880.3460397
Zhang, R., Zhu, F., Liu, J., & Liu, G. (2020). Depth-wise separable convolutions and multi-level pooling for an efficient spatial CNN-based steganalysis.

IEEE Transactions on Information Forensics and Security, 15, 1138–1150. https://doi.org/10.1109/TIFS.2019.2936913

Zhao, X., Zhang, K., Su, Z., Vasan, S., Grishchenko, I., Kruegel, C., Vigna, G., Wang, Y. X., & Li, L. (2024). Invisible image watermarks are provably removable using generative AI. *Advances in Neural Information Processing Systems 37*, 8643–8672. https://doi.org/10.52202/079017-0276