

Detection Of Malicious Payloads Hidden in Digital Images Using Steganalysis Techniques

Rudrarajsinh Ghelot¹, Dr. Swapnesh Taterh²

¹*Student, Master of Science (M.Sc.) in Cyber Security, Savitribai Phule Pune University (SPPU), Pune, Maharashtra, India*

²*Associate Professor, Department of Design Analytics & Cyber Security, MIT Arts, Commerce and Science College, Alandi (D), Pune, Maharashtra, India*

doi.org/10.64643/ijirt.209056-459

Abstract—Digital images are now used everywhere—on social media, in messaging apps, on websites, and on file-sharing services—and this widespread use provides attackers with a convenient way of hiding information. The technique known as steganography, which involves hiding information within media in such a way that its presence is not noticed, has many legitimate applications, but it can just as readily be used to conceal malicious payloads, scripts, or commands within what appears to be a normal picture. This paper looks at the existing literature on image steganalysis with a special emphasis on its application as a security measure to detect malicious payloads, and suggests a security-oriented image analysis system that is intended to identify exactly this type of threat. When images are uploaded, they will be analysed using steganalysis techniques together with other general image features, searching for anomalies that indicate the presence of hidden data; any data that is extracted will then be subject to static analysis to check for harmful content, and each image will be classified as safe, suspicious, or possibly malicious, with the appropriate warning given to the user. The review includes a range of image formats and steganographic methods, with a particular focus on Least Significant Bit (LSB) manipulation, and brings together the statistical, structural, and machine-learning-based detection methods reported in the primary literature examined, including quantitative data on detection rates and accuracy that have been obtained directly from published experiments. The results reported indicate that traditional statistical detectors (RS analysis, chi-square attack, sample pair analysis) are still effective against non-adaptive LSB embedding but perform worse when faced with content-adaptive schemes, whereas convolutional neural network detectors such as Zhu-Net achieve a detection accuracy of about 88% against WOW embedding at 0.4 bits per pixel, representing an improvement of nearly 12 percentage points over the classical SRM-plus-ensemble-classifier baseline. A

survey of eighteen documented real-world stegomalware cases also reveals that PNG and JPEG are still the most commonly used carrier formats. The aim is to demonstrate how steganalysis can be used to uncover covert threats that are hidden in images, thus providing a practical security measure before users open or download suspicious files, while also highlighting the areas that still need to be addressed in future work.

Index Terms—steganography; steganalysis; least significant bit (LSB); stegomalware; digital image forensics; convolutional neural networks; malware detection; image security

I. INTRODUCTION

Digital images are one of the most commonly shared types of content on the internet. They are constantly exchanged through social media, messaging applications, emails, cloud storage, and websites, making it practically impossible to inspect every image manually. Unlike executable files or scripts, images are generally not viewed as a major security threat. This makes them an attractive way for attackers to hide information and bypass traditional security controls. Steganography is the technique of hiding information inside another medium in such a way that the presence of the hidden information is not noticeable. It has several legitimate uses, including digital watermarking, covert communication, and digital rights management.

However, the same ability to conceal information can also be misused. Attackers have used image steganography to hide executable code, command-and-control (C2) instructions, configuration information, and stolen data inside image files that

appear completely normal to users and may not be detected by traditional signature-based security tools [1].

The use of image steganography in attacks is not only theoretical. Researchers have identified its use in different malware families and attack campaigns over the past decade. Examples include early cases such as Duqu and more recent campaigns involving remote access trojans, information stealers, and loaders associated with Worok, Witchetty, LokiBot, IcedID, and XWorm [2]. The MITRE ATT&CK framework also recognises steganography as sub-technique T1027.003 under adversary obfuscation. This shows that steganography is a recognised technique used in real-world cyberattacks rather than simply being a theoretical concept [1]. Since hidden payloads can be stored within pixel data or file structures, traditional antivirus software and content-filtering systems may fail to identify them, allowing stegomalware to pass through security controls that would normally detect suspicious executable files or scripts.

Steganalysis is the process of detecting hidden information within a carrier file and, when possible, identifying or extracting that information. Research in this field has developed significantly over the years. Early approaches mainly focused on statistical and structural analysis techniques that detected the changes introduced by simple embedding methods [3]–[6].

These approaches were later followed by machine-learning methods that used manually designed statistical features to classify images [7]. More recently, deep-learning techniques, particularly deep convolutional neural networks, have been used to automatically learn useful features directly from image data [8], [9]. At the same time, steganographic techniques have also become more advanced. Modern content-adaptive methods are designed to reduce the statistical changes caused by embedding, making hidden information more difficult to detect [10], [11]. This has resulted in an ongoing cycle in which improvements in hiding techniques are followed by improvements in detection techniques.

This paper makes three main contributions. First, it reviews research on steganalysis, covering statistical, machine-learning, and deep-learning approaches, and compares the quantitative detection results reported in primary research studies. Second, it examines eighteen documented real-world stegomalware cases to

understand how attackers have used image steganography, including the image formats most commonly used as carriers. Third, it presents a conceptual layered architecture for a security-focused image analysis system. The proposed architecture combines steganalysis with static analysis of extracted content to classify images into three categories: safe, suspicious, or malicious.

The remainder of this paper is organised as follows. Section II reviews existing research covering the fundamentals of steganography, statistical and structural steganalysis, machine-learning and deep-learning approaches, JPEG-domain steganalysis, and documented stegomalware cases. Section III explains the proposed methodology and dataset design. Section IV presents the implementation architecture. Section V discusses the data analysis and charts based on the reviewed literature. Section VI provides a comparative discussion of the approaches. Section VII discusses the major challenges and limitations, while Section VIII presents possible directions for future research. Finally, Section IX concludes the paper.

II. LITERATURE REVIEW

I have organised the literature review into six themes: (A) steganography fundamentals and the LSB technique, (B) steganalysis (C) feature based machine learning steganalysis, (D) deep learning and CNN based steganalysis (E) JPEG and frequency domain steganalysis and (F) documented real world stegomalware cases.

A. *Steganography fundamentals and the LSB technique*

Steganography is not the same as cryptography because it has a different goal. Cryptography makes a message unreadable unless you have the key. Steganography hides the fact that a message exists by embedding it inside something that looks ordinary. Image steganography comes in two forms. The first is called spatial-domain techniques. These hide data by altering the numbers that describe pixel colours. The best studied spatial-domain method is Least Bit (LSB) substitution. LSB substitution works because changing the significant bit of a pixel's colour hardly changes how the picture looks [3]. The second form is frequency-domain techniques. These modify the numbers used in image compression, such, as the

Discrete Cosine Transform (DCT) coefficients found in JPEG files [12].

I notice that simple methods such as LSB replacement and basic frequency-domain embedding can be discovered because they leave clues. This fact has led to methods that modify images in a way that is harder to detect. These new methods focus on parts of the image that have a lot of detail. They use a system that tries to keep the changes small as possible. Some well-known methods in the domain are called WOW [10]. There are also methods that work in the domain and the JPEG domain. All of these methods share the idea of keeping changes small, as possible. These adaptive methods make it much harder for people to detect the message. They are made to stop the ways of detecting hidden messages.

B. Statistical and structural steganalysis

The first methods used to find messages in images focused on the specific changes in numbers that happen when you use the least significant bit method. The square attack made by Westfeld and Pfitzmann looks at pairs of numbers that change when you add hidden data. Because hiding data makes these pairs more similar if the actual numbers match what you expect it probably means there is hidden data [6]. RS analysis, created by Fridrich, Goljan and Du splits an image into groups of pixels and checks how smooth they are. This helps find data even when the data is spread out randomly [7]. Sample Pair Analysis, made by Dumitrescu, Wu and Wang looks at how pairs of numbers change when hidden data is added. It uses a model to predict how much hidden data there is with high accuracy [8].

A new study done by Joshi and Bhand looked at how these three old methods work when hiding data using the least significant bit method. They tested 1,000 images with amounts of hidden data from 0.05 to 0.40 bits per pixel [11]. Their results, shown in Section V (Fig. 1) Show that the square attack is almost always able to find hidden data no matter how much data is there or how deep it is hidden. RS analysis can find data between 80 and 90 percent of the time. SPA is not as good. Can only find between 34 and 42 percent of hidden messages. This shows that these three old methods are not equally good at finding hidden data. The same study found that even an advanced version of LSB embedding that takes into account edges and how much the image changes does not fully hide the

data from RS analysis. It also does not stop the square attack at all. The square attack works because it looks at pairs of numbers and these pairs are still there even when you replace bits anywhere, in the image [11].

C. Feature-based machine learning steganalysis

As steganography started to use content- embedding steganalysis had to change too. It moved toward using general-purpose machine learning classifiers. These classifiers are trained on sets of handcrafted statistical features. The Spatial Rich Model (SRM) which was proposed by Fridrich and Kodovsky works by calculating a bank of statistical features based on noise residuals. These features capture correlations between neighboring pixels. Then the system feeds these features into a classifier. This ensemble classifier is usually made of Fisher Linear Discriminant base learners. The goal of the classifier is to tell the difference between cover images and stego images [7]. For ten years this SRM+EC combination was the best way to do blind spatial-domain steganalysis. Today the SRM+EC combination is still used as a standard baseline to test new methods, which I show with numbers, in Section V (Fig. 2).

D. Deep-learning and CNN-based steganalysis

Convolutional neural networks have achieved accuracy improvements compared with handcrafted feature methods, especially against modern adaptive embedding schemes because Convolutional neural networks combine feature extraction and classification into one trainable model. Convolutional neural networks steganalyzers such as Xu-Net and Ye-Net used fixed or trainable high-pass preprocessing filter banks derived from Rich Model theory to reduce image content and highlight the weak steganographic noise signal. Yedroudj-Net, proposed by Yedroudj, Comby and Chaumont merged SRM-based filter-bank preprocessing with five layers batch normalisation and Truncated Linear Unit activations and employed a BOSSBase-plus-BOWS2 training strategy; Yedroudj-Net was reported to beat earlier Convolutional neural networks and Rich-Model baselines [8]. Zhu-Net, proposed by Zhang, Zhu, Liu and Liu improved the SRM-based preprocessing filters during training added separable-convolution blocks to increase the signal-to-noise ratio of the signal and applied spatial pyramid pooling to manage images of different sizes; Zhu-Net reduces detection

error by 2.3–8.2 percentage points compared with Yedroudj-Net and by 8.1–13.7 percentage points compared with SRM+EC on the same BOSSBase evaluation protocol [9]. Later Convolutional neural networks have kept this trend adding connections, depthwise-separable convolutions and other efficiency-oriented refinements that narrow the gap with and in some narrow benchmark settings surpass the accuracy of earlier Convolutional neural networks detectors. Exact comparative figures, for five detectors. SRM+EC, Xu-Net, Ye-Net, Yedroudj-Net and Zhu-Net. Versus WOW embedding at two payload levels are shown quantitatively in Section V (Fig. 2) Taken directly from Table VI of Zhang et al. [9].

Reported accuracies in this research should be looked at closely. Numbers that are near or higher than 99 percent are usually found in situations. These situations involve using an embedding method, a set payload and testing on safe academic data sets, like BOSSBase and BOWS2. These tests are not done on payloads or images taken from real network traffic. Also, many studies show that this accuracy drops when images are compressed changed in size or altered in ways after they are embedded [13].

E. JPEG and frequency-domain steganalysis

Since JPEG is the common image format people use every day a lot of research focuses on hiding data inside the frequency domain. JPEG steganography algorithms like F5 J-UNIWARD, UED and UERD work by changing the DCT coefficients. Usually, these algorithms use an additive distortion function to guide the process [12]. Some people even use learning for this job. These deep learning methods combine phase-convolutional feature extraction with special classification stages. Some researchers even apply architectures like DenseNet directly to JPEG images [12]. Because DCT-domain embedding works, inside the JPEG compression pipeline JPEG steganalysis is very sensitive. JPEG steganalysis can be easily broken by changing the quality factor or by re-compression. This happens all the time when people share images on platforms that automatically re-encode your uploads.

F. Documented real-world stegomalware cases

I see that a growing catalogue of documented incidents shows that image-based steganography is an

established used technique in the malware landscape. The MITRE ATT&CK framework lists steganography as sub-technique T1027.003 under Obfuscated Files or Information. Mentions campaigns such as Duqu, PowerDuke and malware linked to Operation Spalax and the SolarWinds-related Raindrop tool [1]. Community-maintained catalogues and vendor threat research list a longer set of cases [2]. Table I summarises eighteen cases; the spread of carrier formats, in these cases is analysed quantitatively in Section V (Fig. 3).

Malware / Campaign	Carrier Format	Malicious Use of Steganography
PowerDuke	PNG	Backdoor components hidden inside PNG files, delivered via spear-phishing [1].
LokiBot (variants)	BMP / JPG	Encrypted binary/DLL data appended to or embedded in BMP and JPG images [2].
IcedID	Image files	Steganographic payload embedded in delivered images as part of the infection chain [2].
Worok	PNG	LSB steganography conceals a PowerShell script and a .NET data-stealing tool [2].
Witchetty (Stegmap)	BMP	Backdoor downloads a BMP concealing a payload encrypted with an XOR key [2].
MyKings Botnet	JPG	Malicious SQL brute-forcing payload hidden inside JPG images [2].

I noticed an interesting thing from this pattern. First, as you can see in Fig. 3 PNG and JPEG formats are the common carriers. I think PNG and JPEG formats are used much because everyone uses them and PNG and JPEG formats have plenty of extra space to hide small things like scripts loader stubs or configuration strings. Second, I have seen that attackers often use steganography along with a way to deliver files. This means the image itself is one small part of a much longer attack chain. Third most cases that people have written about use simple ways to hide data like LSB substitution. They do not use the content-adaptive

schemes that researchers talk about in books. To me this shows that classical statistical steganalysis (Section II-B, Fig. 1) is still very useful for defense, against real-world stegomalware. Even if smarter attackers start using to-detect adaptive schemes classical statistical steganalysis can still help us fight back.

III. PROPOSED METHODOLOGY

In line, with the research abstract that motivates this review this study aims to test steganalysis techniques. The study uses steganalysis techniques as a security control to stop image payloads. It does not view steganalysis techniques as a tool to find hidden messages.

A. Dataset design

Testing is planned using a dataset that has images. These images are not changed. They come from photography sources. The dataset also includes two types of modified images. The first type has safe payloads. These payloads are text or random bits. They are of lengths. This follows what is used in studies, like BOSSBase and BOWS2.

The second type has payloads that look like ones. These are not malware. They are executable code or script parts. They are made to look like attacks but are not dangerous. This setup matches what the abstract said about testing. It lets the same process be used as an image detector and as a tool to find harmful content.

B. Detection features

The pipeline follows what was discussed in Section II. It uses three types of features. First there are structural features that come from RS analysis the chi-square attack and sample pair analysis [4]– [6]. There are handcrafted statistical features similar to those used in the Rich Model and these are used with an ensemble classifier [7]. Third there are features that a CNN learns from an architecture that starts with an SRM filter bank, like the Yedroudj-Net or the Zhu-Net [8] [9].

C. Classifiers

I decided to compare two types of classifier families, which is how most people do it in research papers. The first one is a linear/ baseline that uses a Fisher Linear Discriminant ensemble over Rich Model features just

like the SRM+EC [7] method. The second one is a neural network classifier that works directly on pixel data from start, to finish. By using classifiers from modelling families, I can see if the results come from the way the features are built or if they come from how much the model can learn.

D. Evaluation protocol

Four evaluation conditions are proposed, adapted from steganalysis evaluation practice [8] [9]: (i) -format evaluation, in which a classifier is trained and evaluated on a single image format (e.g. PNG only); (ii) cross-format evaluation in which a classifier trained on one format is evaluated on another to test generalisation; (iii) payload-scaling evaluation, in which detection performance is measured across a range of embedding rates (0.05–0.40 bpp consistent with [11]); and (iv) malicious-payload evaluation, in which the systems final safe/suspicious/malicious classification is assessed specifically against the controlled malicious-like samples described in Section III-A rather than only, against the binary stego/cover label.

IV. IMPLEMENTATION ARCHITECTURE

Using the research from Section II and the methods from Section III this part explains a step-by-step design for a security-focused image analysis system. Like the data set and testing plan mentioned earlier the design shown here is a suggestion and has not been built or tried from start to finish in this paper. Its purpose is to show how the clues from language and pixel steganalysis, in Section II could be put together into a real security control.

A. Core modules

1. Ingestion and normalisation: the uploaded image is accepted across multiple common formats (JPEG, PNG, BMP, GIF, WebP, SVG) and normalised, with headers, metadata, and format-specific structures parsed so later stages can apply the correct format-specific analysis.
2. Lightweight statistical screening: fast classical detectors (RS analysis, chi-square attack, sample pair analysis) are applied first as an inexpensive first pass, since Section II-F shows that many real-world stegomalware cases still rely on comparatively

simple, non-adaptive embedding that these detectors catch efficiently.

3. Deep-learning classification: images that pass the first screen are passed through domain-appropriate CNN classifiers a spatial-domain model in the style of Zhu-Net or SRNet for uncompressed formats, and a JPEG-domain phase-aware or DCT-based CNN for JPEG images to catch adaptive embedding the first pass would miss.

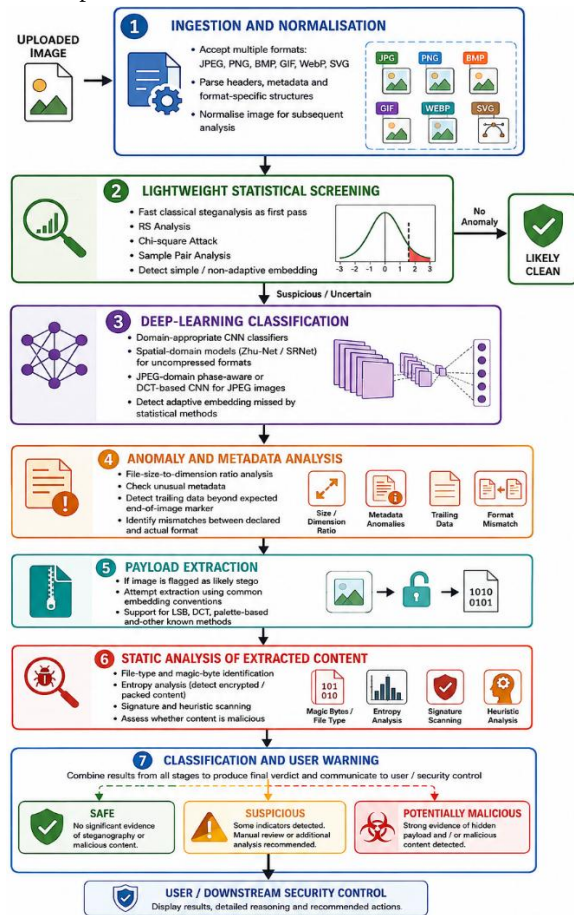


Figure 1: Flowchart

4. Anomaly and metadata analysis: file-size-to-dimension ratios, unusual metadata or trailing data after the expected end-of-image marker, and mismatches between declared and actual format are checked, reflecting common artefacts observed in the case studies in Table I.
5. Payload extraction: where a flagged image is found likely to be a stego image, extraction is attempted using knowledge of common embedding conventions observed across documented campaigns.

6. Static analysis of extracted content: any successfully extracted payload is analysed using file-type and magic-byte identification, entropy analysis to flag encrypted or packed content, and signature/heuristic scanning, to assess whether the hidden content is itself malicious.

7. Classification and user warning: the system combines the outputs of the preceding stages into a final classification of safe, suspicious, or potentially malicious, and communicates this classification and the underlying reasoning to the user or downstream security control.

B. Design rationale

This design has two ideas from the review of the literature. The first idea is that no one method works best for all situations. Some older methods still work well for hiding techniques that are often used in real stegomalware cases. For more advanced methods that change based on the content newer methods like CNN classifiers are needed. Using simple checks first and then more detailed and expensive deep learning analysis helps cover more cases without using too much computing power. The second idea is that just finding data is not enough, for real security work. So the process includes steps after finding hidden data. It also takes the hidden data out. Looks at it closely instead of stopping at just knowing if something is hidden or not.

V. DATA ANALYSIS AND RESULTS

This section presents quantitative data compiled from the primary sources reviewed in Section II, illustrated with charts, to support the comparative discussion in Section VI.

A. Detection rate of classical statistical steganalysis

Fig. 1 Shows the detection rate of RS analysis the chi-square attack and Sample Pair Analysis when used against fixed-depth LSB-2 embedding at five levels. The data come from the experiments reported by Joshi and Bhand [11]. The chi-square attack keeps its detection rate at 100 % for every payload that was tested. RS analysis shows a detection rate between 81 % and 83 %. Sample Pair Analysis has a detection rate of about 34 % to 35 %. These results confirm that the detection rates of these three detectors are not the

same. A fused or multi-detector strategy gives a detection rate, then any single classical test alone.

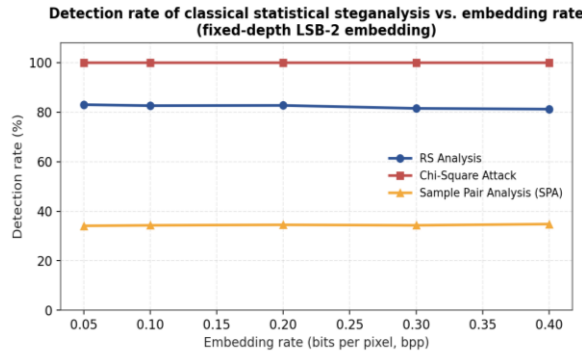


Fig. 1. Detection rate of classical statistical steganalysis techniques against fixed-depth LSB-2 embedding across payload levels. Data source: Joshi & Bhand, arXiv:2603.18105, Table 3 [11].

B. CNN steganalysis accuracy comparison

Fig. 2 Shows how the error rates reported by Zhang, Zhu, Liu and Liu in Table VI of the paper are turned into detection accuracy for five steganalyzers. The steganalyzers are SRM+EC Xu-Net, Ye-Net, Yedroudj-Net and Zhu-Net. These steganalyzers were tested against WOW embedding at 0.2 and 0.4 bits per pixel on the BOSSBase dataset. Accuracy improves steadily from the SRM+EC baseline. The SRM+EC baseline gives 63.5 percent accuracy at 0.2 bits per pixel and 74.5 percent accuracy at 0.4 bits per pixel. Accuracy climbs to 76.7 percent at 0.2 bits per pixel for Zhu-Net and 88.2 percent at 0.4 bits per pixel for Zhu-Net. This is an improvement of thirteen percent points at 0.2 bits per pixel and fourteen percent points at 0.4 bits per pixel.

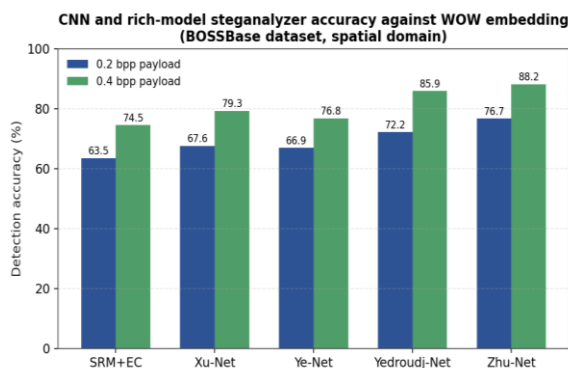


Fig. 2. Detection accuracy of five steganalyzers against WOW embedding at 0.2 and 0.4 bpp, derived from reported error rates. Data source: Zhang, Zhu, Liu & Liu, arXiv:1807.11428, Table VI [9].

Accuracy gains are consistent with the pattern described in Section II-D. Section II-D explains that each new CNN architecture refines preprocessing filters uses channel correlations and adds pooling. These changes give measurable accuracy gains, over both earlier CNNs and the handcrafted Rich Model baseline.

C. Carrier format distribution in real-world stegomware

Fig. 3 summarises the carrier image format used across the eighteen documented stegomware cases compiled in Table I. PNG is the most common carrier (used in six of eighteen cases), followed closely by JPEG/JPG (five cases) and BMP (three cases), with SVG and other/mixed formats accounting for the remainder. This supports the design decision in Section IV-A to prioritise PNG- and JPEG-specific detection paths, while still maintaining coverage for less common but still-exploited formats such as BMP and SVG.

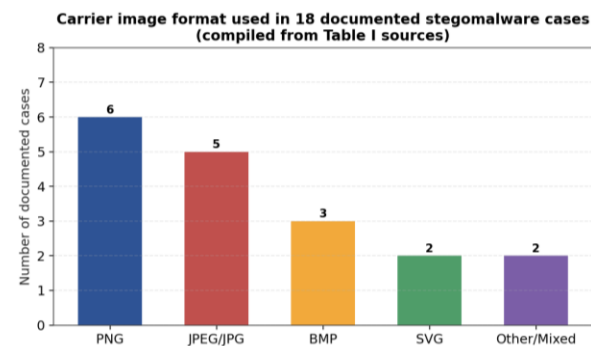


Fig. 3. Distribution of carrier image formats across 18 documented stegomware cases. Compiled by the authors from sources cited in Table I [1], [2].

D. Fidelity-detectability trade-off of adaptive embedding

Fig. 4 plots the PSNR (peak signal-to-noise ratio) of fixed-depth LSB-1, fixed-depth LSB-2, and entropy/edge-aware adaptive embedding across the same five payload levels, using data reported by Joshi and Bhand [11]. The adaptive scheme consistently yields higher PSNR than both fixed-depth baselines approximately 2.8–3.0 dB higher than LSB-1 and 6.8–7.0 dB higher than LSB-2 indicating better visual fidelity at every tested payload. However, as shown in Fig. 1's counterpart data for the adaptive scheme (not plotted separately here for clarity), this fidelity improvement only partially reduces RS detectability

and does not defeat the chi-square attack, illustrating that visual imperceptibility and statistical undetectability are related but distinct properties of a steganographic scheme.

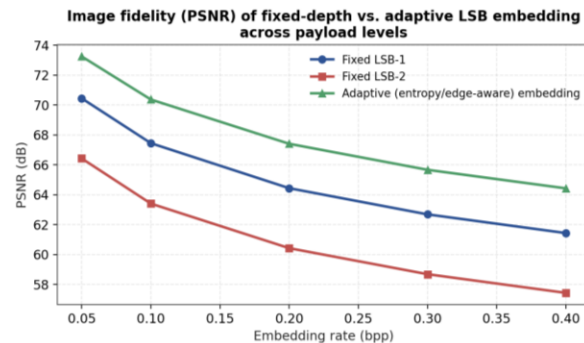


Fig. 4. Peak signal-to-noise ratio (PSNR) of fixed-depth vs. adaptive LSB embedding across payload levels. Data source: Joshi & Bhand, arXiv:2603.18105, Table 1 [11].

VI. COMPARATIVE DISCUSSION

Table II synthesises the technique families reviewed in Section II and quantified in Section V, comparing their detection approach, representative reported performance, and principal limitation.

Technique Family	Representative Reported Performance	Principal Limitation
Statistical/structural (RS, chi-square, SPA)	Chi-square ~100% detection on fixed LSB; RS ~81–92%; SPA ~34–42% [11]	Effectiveness drops sharply against content-adaptive embedding designed to evade these specific statistics
Feature-based ML (SRM + Ensemble Classifier)	63.5%–74.5% accuracy vs. WOW at 0.2–0.4 bpp [9]	Accuracy plateaus below modern CNNs; requires domain-specific handcrafted features
CNN-based (Xu-Net, Ye-Net, Yedroudj-Net, Zhu-Net, SRNet)	76.7%–88.2% accuracy vs. WOW at 0.2–0.4 bpp for Zhu-Net [9]	Compute-intensive; accuracy sensitive to compression/resizing; benchmark accuracy may not transfer to real-world images

JPEG/frequency-domain (DCTR, phase-aware CNN, DenseNet)	Matches the dominant real-world image format; detects F5/J-UNIWARD/UE/ED/UEIRD-style embedding [12]	Sensitive to JPEG quality factor and re-compression; largely separate models needed per domain
Integrated pipelines (steganalysis + static analysis)	Directly targets malicious intent rather than presence of hidden data alone (Section IV)	Relatively little published academic work; extraction not always possible without knowledge of embedding key/algorithm

Two implications come from Table II. The data in Section V. First the accuracy gap between classical statistical detectors and modern CNN classifiers. About 13 to 25 percentage points at similar payloads as shown in Fig. 2.

Explains why the layered architecture described in Section IV is useful. In that architecture lightweight statistical screening acts as a first pass while CNN classification is saved for content that needs higher-confidence analysis. Second chi-square detection stays at 100% against LSB replacement even when an adaptive-depth variant is used [11].

The real-world case survey, in Section II-F and Fig. 3 Also shows that many attacks still use non-adaptive embedding. Because of that a well-tuned classical detector layer alone would probably catch a share of current real-world stegomalware even before the more expensive CNN layer is called on.

Second chi-square detection remains at 100 % against LSB replacement even when an adaptive-depth variant is used, as shown in reference [11]. The real-world case survey, in Section II-F and Fig. 3 Further shows that many documented attacks rely on non-adaptive embedding.

Therefore, a well-tuned classical detector layer alone could capture a share of current real-world stegomalware before the more expensive CNN layer is invoked.

VII. CHALLENGES AND OPEN RESEARCH ISSUES

A. Adaptive and adversarial steganography

Content- embedding algorithms are created to reduce the statistical distortion that steganalysis can find. This competition keeps going. It includes embedding costs that are improved using machine learning to stop steganalyzers.

This is a changing area of research. Published detection accuracy against a fixed known embedding algorithm, like the ones in Fig. 2 Might not show the real-world performance, against carefully adjusted embedding methods.

B. Generalisation and robustness to transformations

When people share images on the internet, they often resize them crop them or change the file types. They also compress the images to make them smaller. These common changes mess up the patterns that steganalysis looks for.

These changes can even break the hidden data, inside an image. Some studies mentioned in Section II-D show that CNN steganalyzer tools do not work well after these changes happen. Even though the CNN steganalyzer tools work great on data the CNN steganalyzer tools fail when the image is not perfect [13].

C. Scarcity of malware-labelled datasets

Academic steganalysis research depends a lot on datasets like BOSSBase [14] and BOWS2 [15] which take photographs and add random-bit messages that are created on purpose instead of real harmful code. There is not data that shows real harmful messages with images as carriers. This limits the ability to properly test systems, like the one described in Section IV against dangers.

D. Multi-format coverage and computational cost

World steganographic malware, as shown in Figure 3 uses PNG, JPEG, BMP and SVG files as carriers. Each of these files needs ways to find hidden information. Deeper neural network models like SRNet are harder to use because they take computing power. This creates a problem, between covering all ways to detect the malware and being fast enough to check a lot of images quickly.

VIII. FUTURE RESEARCH DIRECTIONS

- We need to focus on benchmarking that actually tests steganalysis systems under real-world changes like recompression, resizing or format conversion. It is not enough to test them on clean benchmark images.
- We should. Share malware-aware datasets. These datasets should pair image carriers with realistic safely contained like payloads so we can do better security-specific evaluation.
- It is important to extend -domain architectures like SRNet. We want to create detectors that can cover more formats like SVG and WebP.
- I think we need more research into hybrid statistical-plus-deep-learning pipelines. These pipelines, like the ones mentioned in Section IV can help balance cost with detection coverage at scale.
- We must add training into steganalysis models. This will help make steganalysis models less vulnerable, to GAN-optimised embedding.
- We should develop detection systems. These systems need to work with static and dynamic malware-analysis tooling to help make analyst workflows easier.
- Research is needed into model compression, pruning and quantisation for steganalysis CNNs. This will help steganalysis CNNs run in time on email gateways or messaging platforms.

IX. CONCLUSION

Image steganography is a tool that works two ways in security. It has uses but people also use image steganography to hide bad files control commands and stolen data to sneak past security tools. The books and papers I looked at along with the data in Section V show how technology has grown. We have moved from statistical tools to machine learning and then to deep convolutional neural networks. Each new step makes finding hidden data easier. For example, old tools like SRM+EC only hit 63–75% accuracy. Zhu-Net reaches 77–88% accuracy when looking for the same WOW embedding [9]. At the time Table I and Fig. 3 Show eighteen real cases of stegomalware. Many of these attacks still use methods and PNG/JPEG files. This tells me that old techniques are

still very useful when you use them as part of a defense system.

Finding data is only one part of the security problem. A real security system must do more than just find data. A good system should try to pull out the hidden data and check it to see if it is actually a virus before it tells the user. In this paper I have suggested a plan in Section IV. This plan uses statistical checks, CNN classification, metadata checks and malware analysis to handle everything from start to finish. There are still problems to solve. It is hard to stop people from making changes to hide data and it is hard to make tools that work on every type of image. Also, it is hard to find examples of malware to train the systems. Solving these problems is an important goal for future research, in steganalysis and cybersecurity.

ACKNOWLEDGMENT

The author thanks [Swapnesh Taterh], [Associate Professor, Department of Design Analytics & Cyber Security,] for guidance during this review, and [MIT Arts, Commerce and Science College, Alandi (D)] for institutional support.

REFERENCES

- [1] MITRE ATT&CK, "Obfuscated Files or Information: Steganography, Sub-technique T1027.003," MITRE Corporation. [Online]. Available: <https://attack.mitre.org/techniques/T1027/003/>
- [2] "steg-in-the-wild," GitHub repository, "A list of attacks or malware using steganography or information hiding." [Online]. Available: <https://github.com/lucacav/steg-in-the-wild>
- [3] M. Hussain, A. W. A. Wahab, Y. I. B. Idris, A. T. S. Ho, and K.-H. Jung, "Image steganography in spatial domain: A survey," *Signal Processing: Image Communication*, vol. 65, pp. 46–66, 2018.
- [4] A. Westfeld and A. Pfitzmann, "Attacks on steganographic systems," in *Information Hiding*, LNCS, vol. 1768, Springer, 1999, pp. 61–76.
- [5] J. Fridrich, M. Goljan, and R. Du, "Reliable detection of LSB steganography in colour and grayscale images," in *Proc. ACM Workshop on Multimedia and Security*, 2001, pp. 27–30.
- [6] S. Dumitrescu, X. Wu, and Z. Wang, "Detection of LSB steganography via sample pair analysis," *IEEE Trans. Signal Processing*, vol. 51, no. 7, pp. 1995–2007, 2003.
- [7] J. Fridrich and J. Kodovsky, "Rich models for steganalysis of digital images," *IEEE Trans. Information Forensics and Security*, vol. 7, no. 3, pp. 868–882, 2012.
- [8] M. Yedroudj, F. Comby, and M. Chaumont, "Yedroudj-Net: An efficient CNN for spatial steganalysis," in *Proc. IEEE ICASSP*, 2018, pp. 2092–2096.
- [9] R. Zhang, F. Zhu, J. Liu, and G. Liu, "Efficient feature learning and multi-size image steganalysis based on CNN," *arXiv:1807.11428*, 2018.
- [10] V. Holub and J. Fridrich, "Designing steganographic distortion using directional filters," in *Proc. IEEE WIFS*, 2012, pp. 234–239.
- [11] A. Joshi and K. Bhand, "Adaptive fuzzy logic-based steganographic encryption framework: A comprehensive experimental evaluation," *arXiv:2603.18105*, 2026.
- [12] J. Fridrich, "Feature-based steganalysis for JPEG images and its implications for future design of steganographic schemes," in *Proc. 6th Int. Workshop on Information Hiding*, LNCS, vol. 3200, 2004, pp. 67–81.
- [13] *Frontiers in Artificial Intelligence*, "Deep learning for steganalysis: Evaluating model robustness against image transformations," *Frontiers in Artificial Intelligence*, vol. 8, Art. no. 1532895, 2025.
- [14] P. Bas, T. Filler, and T. Pevny, "Break our steganographic system: The ins and outs of organizing BOSS," in *Information Hiding*, Springer, 2011, pp. 59–70.
- [15] P. Bas and T. Furon, "BOWS-2 Contest," 2007. [Online]. Available: <http://bows2.ec-lille.fr>
- [16] V. Holub, J. Fridrich, and T. Denemark, "Universal distortion function for steganography in an arbitrary domain," *EURASIP Journal on Information Security*, vol. 2014, no. 1, 2014.
- [17] A. Joshi and K. Bhand, "Adaptive fuzzy logic-based steganographic encryption framework: A comprehensive experimental evaluation," *arXiv:2603.18105*, 2026.
- [18] J. Fridrich, "Feature-based steganalysis for JPEG images and its implications for future design of steganographic schemes," in *Proc. 6th Int. Workshop on Information Hiding*, LNCS, vol. 3200, 2004, pp. 67–81.

- [19] B. Li, M. Wang, J. Huang, and X. Li, "A new cost function for spatial image steganography," in Proc. IEEE ICIP, 2014.
- [20] V. Sedighi, R. Cogranne, and J. Fridrich, "Content-adaptive steganography by minimizing statistical detectability," IEEE Trans. Information Forensics and Security, vol. 11, no. 2, pp. 221–234, 2016.
- [21] A. Westfeld, "F5—A steganographic algorithm," in Information Hiding, LNCS, vol. 2137, Springer, 2001, pp. 289–302.
- [22] J. Kodovsky, J. Fridrich, and V. Holub, "Ensemble classifiers for steganalysis of digital media," IEEE Trans. Information Forensics and Security, vol. 7, no. 2, pp. 432–444, 2012.
- [23] Issues in Information Systems, "Comparative evaluation of machine learning models for image steganalysis," Issues in Information Systems, vol. 25, no. 3, pp. 119–132, 2024.
- [24] M. Boroumand, M. Chen, and J. Fridrich, "Deep residual network for steganalysis of digital images," IEEE Trans. Information Forensics and Security, vol. 14, no. 5, pp. 1181–1193, 2019.
- [25] T. S. Reinel et al., "GBRAS-Net: A convolutional neural network architecture for spatial image steganalysis," IEEE Access, vol. 9, pp. 14340–14350, 2021.
- [26] N. J. D. L. Croix, T. Ahmad, and F. Han, "Comprehensive survey on image steganalysis using deep learning," Array, vol. 22, Art. no. 100353, 2024.
- [27] Frontiers in Artificial Intelligence, "Deep learning for steganalysis: Evaluating model robustness against image transformations," Frontiers in Artificial Intelligence, vol. 8, Art. no. 1532895, 2025.
- [28] J. Zeng, S. Tan, B. Li, and J. Huang, "Large-scale JPEG image steganalysis using hybrid deep-learning framework," IEEE Trans. Information Forensics and Security, vol. 13, no. 5, pp. 1200–1214, 2018.
- [29] J. Zeng, S. Tan, B. Li, and J. Huang, "Large-scale JPEG image steganalysis using hybrid deep-learning framework," arXiv:1611.03233, 2016.
- [30] "JPEG steganalysis based on DenseNet," arXiv:1711.09335, 2017.
- [31] PDI Technologies, "Steganography in cybersecurity: A growing attack vector," PDI Security Blog, 2025. [Online]. Available: <https://security.pditechnologies.com/blog/steganography-in-cybersecurity-a-growing-attack-vector/>