Positive Identification of LSB Image Steganography Using Cover Image Comparisons

Michael Pelosi
Texas A&M University
Texarkana, Texas

Least Significant Bit (LSB) steganography is a technique which encodes data to the least significant bit of pixel color channel data in an image. This can include the red, green, blue, or alpha color channel data. The LSB is favorable for potential modification since changes here will result in the *least detectable visual artifacts* to the human viewer compared to the original image.

We introduce a new software concept specifically designed to allow the digital forensics professional to clearly identify and attribute instances of LSB image steganography by using the original cover image in side-by-side comparison with a suspected steganographic payload image.

The *"CounterSteg"* software allows detailed analysis and comparison of both the original cover image and any modified image, using sophisticated bit- and color-channel visual depiction graphics. In certain cases, the steganographic software used for message transmission can be identified by the forensic analysis of LSB and other changes in the payload image. We demonstrates usage and typical forensic analysis with nine commonly available steganographic programs.

LSB Steganography

- The simplest and popular image steganographic method is the least significant bit (LSB) substitution.
- It embeds messages into cover image by replacing the least significant bits directly.
- The hiding capacity can be increased by using up to 4 least significant bits (one each for Red, Green, Blue, and Alpha color channels, respectively) in each pixel.
- It has a common weak point, the LSB value changes alter most images statistically to some extent.

2

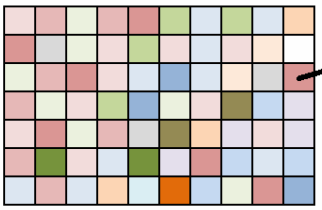
The simplest and popular image steganographic method is the least significant bit (LSB) substitution.

It embeds messages into cover image by replacing the least significant bits directly.

The hiding capacity can be increased by using up to 4 least significant bits (one each for Red, Green, Blue, and Alpha color channels, respectively) in each pixel.

It has a common weak point, the LSB value changes alter most images statistically to some extent. What this means is that certain advanced statistical techniques can be utilized to detect an image may have a message embedded. Our software take some steps to prevent this type of detection, and I will go over some of those shortly.

LSB Steganography



Value to encode

R = 1 1 0 1 1 0 1 X → 1 → 11011011 Hidden Bit 0
0 → 11011010

G = 1 0 0 1 0 1 1 X → 1 → 10010111 Hidden Bit 1
0 → 10010110

B = 1 0 0 1 0 1 0 X → 1 → 10010101 Hidden Bit 2
0 → 10010100

LSB

Rise in the Usage of Steganography for Malware

- Microcin, alias Six Little Monkeys
- NetTraveler
- Zberp
- Enfal, and version with new loader called Zero.T
- Shamoon
- KinS
- ZeusVM
- Triton, alias Fibbit

4

The first is that these methods aid malicious actors in concealing not only the data itself but the fact that data is being uploaded and downloaded. A second reason is that steganography can bypass deep packet inspection (DPI) systems. A third reason is that the use of steganography may help bypass security checks by anti-APT products.

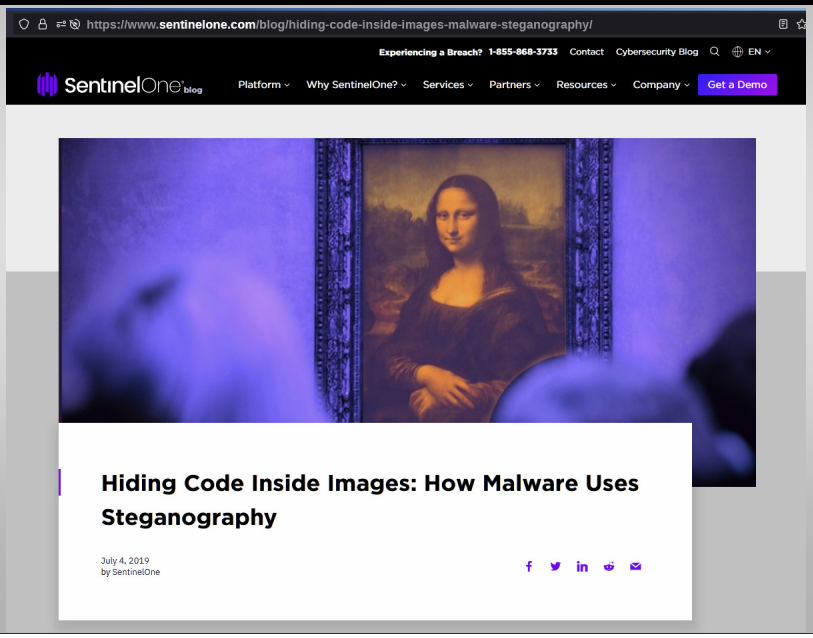
Various Malware Using Steganography

- AdGholas – this malware hides malicious JavaScript in image, text, and HTML files
- Cerber – embeds malicious code in image files
- DNSChanger – uses PNG LSBs to hide malware AES encryption key
- Stegano – PNG formatted banner ads containing malicious code
- Stegoloadr (aka 'Lurk') – this malware uses both steganography and cryptography to conceal an encrypted URL to deliver later stage payloads
- Sundown – white PNG files are used to conceal exploit code or exfiltrate user data
- SyncCrypt – ransomware that hides part of its core code in image files
- TeslaCrypt – HTML comment tags in an HTTP 404 error page contain C2 server commands
- Vawtrak (aka 'Neverquest') – hides a URL in the LSBs of favicons in order to download a malicious payload
- VeryMal – malware targets macOS users with malicious javascript embedded in white bar
- Zbot – appends data to the end of a JPEG file containing hidden data
- ZeroT – Chinese malware that uses steganography to hide malware in an image of Britney Spears

5

Attackers have been found to use steganography to conceal parts of ransomware attack code, deliver malicious javascript and even carry cryptominers.

Hiding Code Inside Images: How Malware Uses Steganography



6

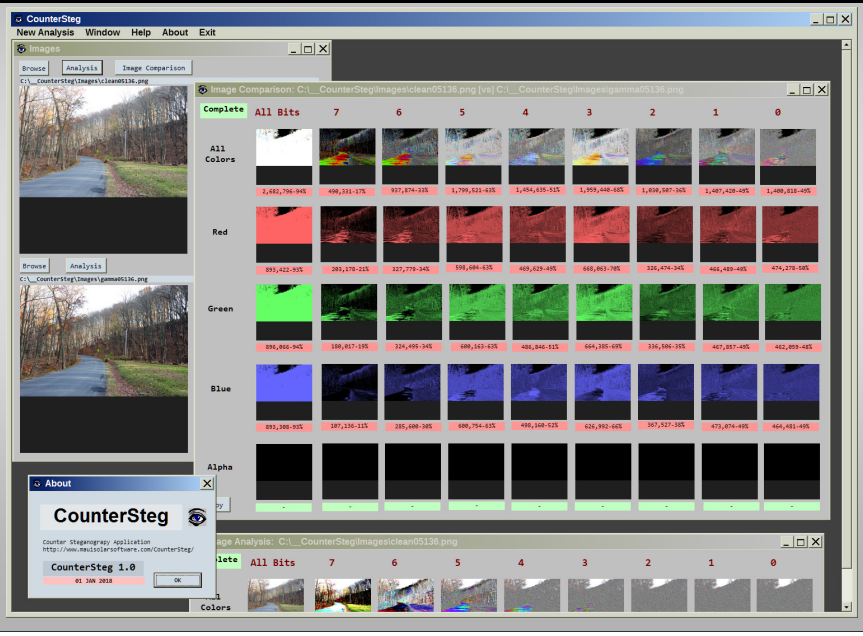
Conclusion

Hiding a file, picture, message or even a video within another file can be an effective way for malware authors to obscure either their own payload or to exfiltrate user data. Given the popularity of image sharing on social media sites and the prevalance of image-based advertisements, we expect the recent trend of using steganography in malware to continue.

Combined with how difficult it is for end users to spot a maliciously crafted image file, it's vital that enterprises are using behavioral AI software to detect the execution of malicious code, regardless of whether it originates from an image or other file, or even if it is fileless malware.

CounterSteg v1.0

http://199.175.52.196/CounterSteg/



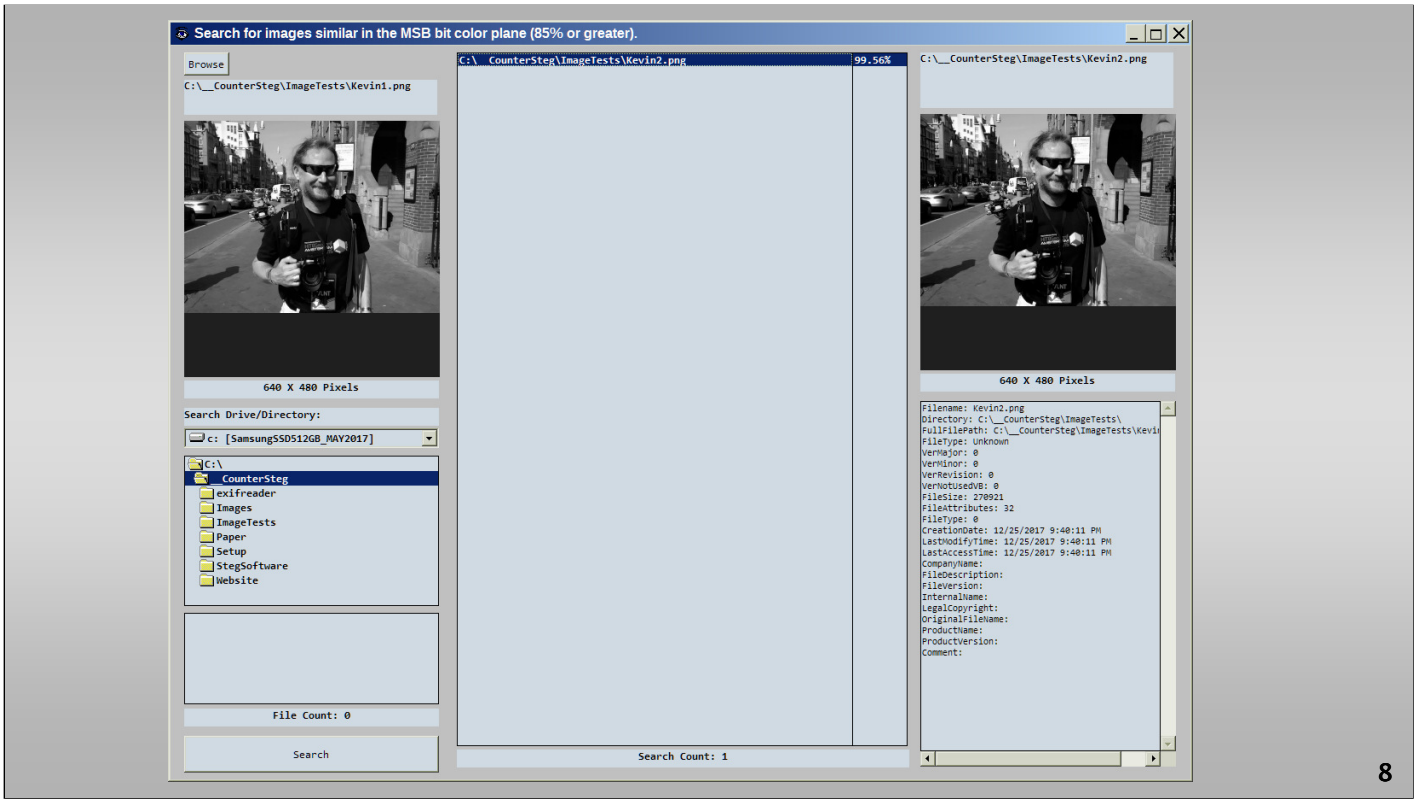
7

The software introduced with this paper, *CounterSteg*, is available free of charge from the following website: http://199.175.52.196/CounterSteg/. This Windows-based software allows the loading of two images and comparison of pixel color bits in the LSB plane. The software will also run under Linux with Wine installed.

Detailed analysis is performed visually at the moment, however we envision future algorithms which can automate the results conclusion positively. The software can be used to detect differences between original and payload image. Positive detection implies, in general, identical pixel dimensions and most pixels identical except for various LSB values. Human forensic analysis does confirm final analysis using additional informed investigation techniques described in this paper.

If a suspect image is detected, a search should be conducted for visually similar and pixel dimensionally identical images in the locations listed above, among others. Once potential matches are identified, the software should be used to look for differences in the LSB and perhaps nearby planes.

If such differences are detected, it can be considered positive identification for the use of steganography, although it is unlikely the original message or data can be recovered, except with the cooperation of the suspect, or acquisition of the original software and/or encryption key used to embed the data. However, with positive results in hand, this further investigation or warrant acquisition can be embarked upon with great confidence.



8

SIMILAR IMAGE SEARCH

To aid in locating images for further investigation, the *CounterSteg* software includes facilities for similar image searching on all machine drives and directories. The similar image search will identify images of identical pixel dimensions and most significant bit values. The value for search of most significant bit similarity percentage is software selectable. The search window design is shown below as Slide 9.

Places to find non-payload (also known as a "cover") image:

- Suspect hard drive filesystems
- Suspect removable USB drives
- Suspect cameras and mobile devices
- Suspect CDs and DVDs
- Local email inboxes/outboxes
- Cloud email inboxes/outboxes
- Recent web search and browser histories
- Google image searches
- Network attached storage devices
- Employment computers and networks
- Recycle bins
- Deleted files removed from recycle bins
- Online photo galleries
- Personal and business associates' files as listed above

9

By locating the original digital image file for comparison, the alteration of LSBs alone is quite the forensic "smoking gun" so to speak for reliable attribution for the use of steganography software to send messages or data files. This could be a starting point for further investigation for law enforcement or other investigative authorities.

In performing good steganographic procedure, a suspect will take care to data wipe any original cover file, to prevent such a comparison from taking place. However, in practice human error and technical limitations may prevent completely effective data erasure of the original cover image.

In that light, we recommend an active search for the original image if suspicion of steganographic usage exists. There are many possibilities for locating the original image that will allow later positive attribution for the use of steganography.

Results Using Various Steganographic Programs for Experimentation and Analysis

- **Steganography Online** – http://stylesuxx.github.io/steganography/
- **StegoShare** – http://stegoshare.sourceforge.net/
- **Geocaching Toolbox** – https://www.geocachingtoolbox.com/index.php?page=steganography
- **DevFarmSteganography** – https://devfarm.it/steganography/
- **f5stego.js** – http://desudesutalk.github.io/f5stegojs/
- **BitCrypt** – http://bitcrypt.moshe-szweizer.com/
- **OpenPuff** – http://embeddedsw.net/OpenPuff_Steganography_Home.html
- **OpenStego** – https://www.openstego.com/index.html
- **OTP-Steg** – http://www.199.175.52.196.com/OTP-Steg/

10

As examples of the type of forensic steganalysis that can be conducted with *CounterSteg*, we have embedded text data into a standard cover image using some of the above listed and easily available steganographic programs.

Each of these programs was accessed directly from a website or downloaded executable.

Each took less than 10-20 minutes to use to embed the standard text data into the standard cover image, which is shown on Slide 12.

JFK Inauguration Speech

1,366 Words – 7,566 bytes – January 20, 1961

The standard text embedded in the image was the President Kennedy inauguration speech, which is 1,366 words, and 7,512 characters.

The size of the text is 7.38 KB (7,566 bytes), file size on disk was 8.00 KB (8,192 bytes). Kennedy's inauguration speech was delivered on January 20, 1961.

7566 * 8 = 60,529 bits

60,529 * 0.5 = 30,264

30,264 * 0.1 = 3,264 bits.

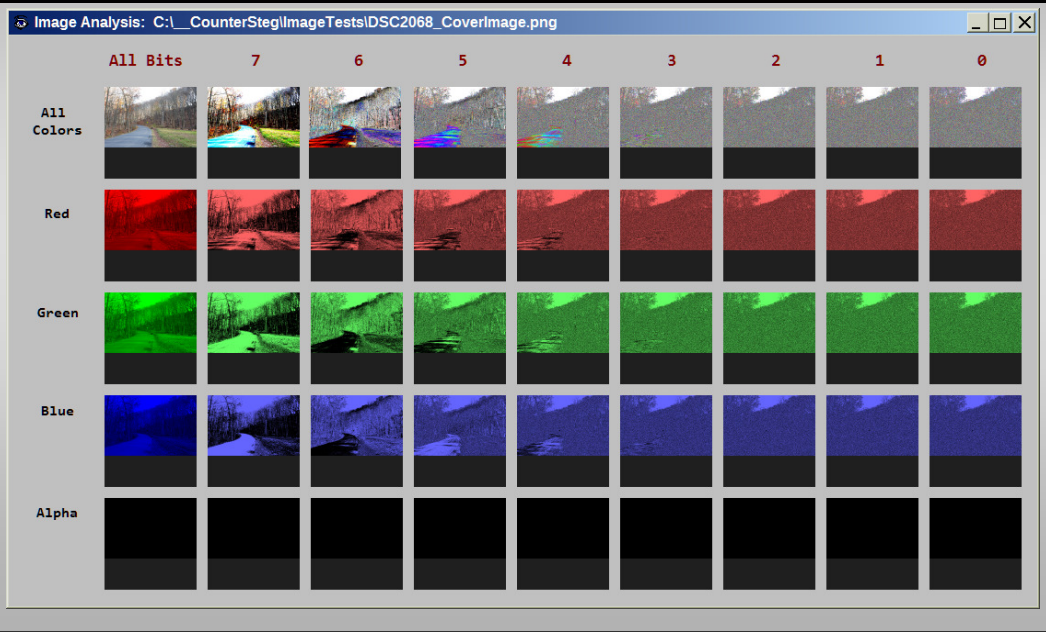
Cover image taken with Nikon D90

The standard cover image shown was taken by the authors in Keyser, West Virginia using a Nikon D90 digital camera. This image shows a fairly even distribution of red, green, and blue colors throughout the image, except for the center top open to the sky. In this specific area, the camera CMOS sensor saturated to white (RGB(255,255,255)), and each of the pixels here represents that single saturated white color. This is notable for LSB steganograpy in that steganographic programs that modify pixels in this area will be more easily statistically detected. Alterations to the LSB values in pixels in solid color, or saturated, portions of the image are a good indicator of nonstandard modifications (such as steganography). Good steganographic programs will attempt to avoid modifications to these specific areas, among others.

The *CounterSteg* program produces detailed visual analysis and comparison of digital images, specifically in each color and bit-plane. In addition to combinations of colors in a particular bit-plane. Slide 13 shows that the image analysis window, which calculates results for 45 various bit plane and color combinations. The Alpha channel is the transparency channel, and remains on unused in many images.

Surprisingly, however, some steganographic programs make spurious or data-carrying modifications in the alpha channel, so it is important to also keep an investigative eye on this color channel. In the analysis window shown below, each color channel is broken down by bits, with bit 0 corresponding to the LSB, and bit 7 corresponding to the MSB. The values in each bit plane are shown for red, green, and blue channels, as well as the alpha channel. For the graphic shown for "All Bits", the pixel color here will be non-black if any of the bits (0-7) is set to a nonzero. The color value is the relative intensity of that color (red, green, or blue) in the range of 0 to 255.

Analysis of cover image



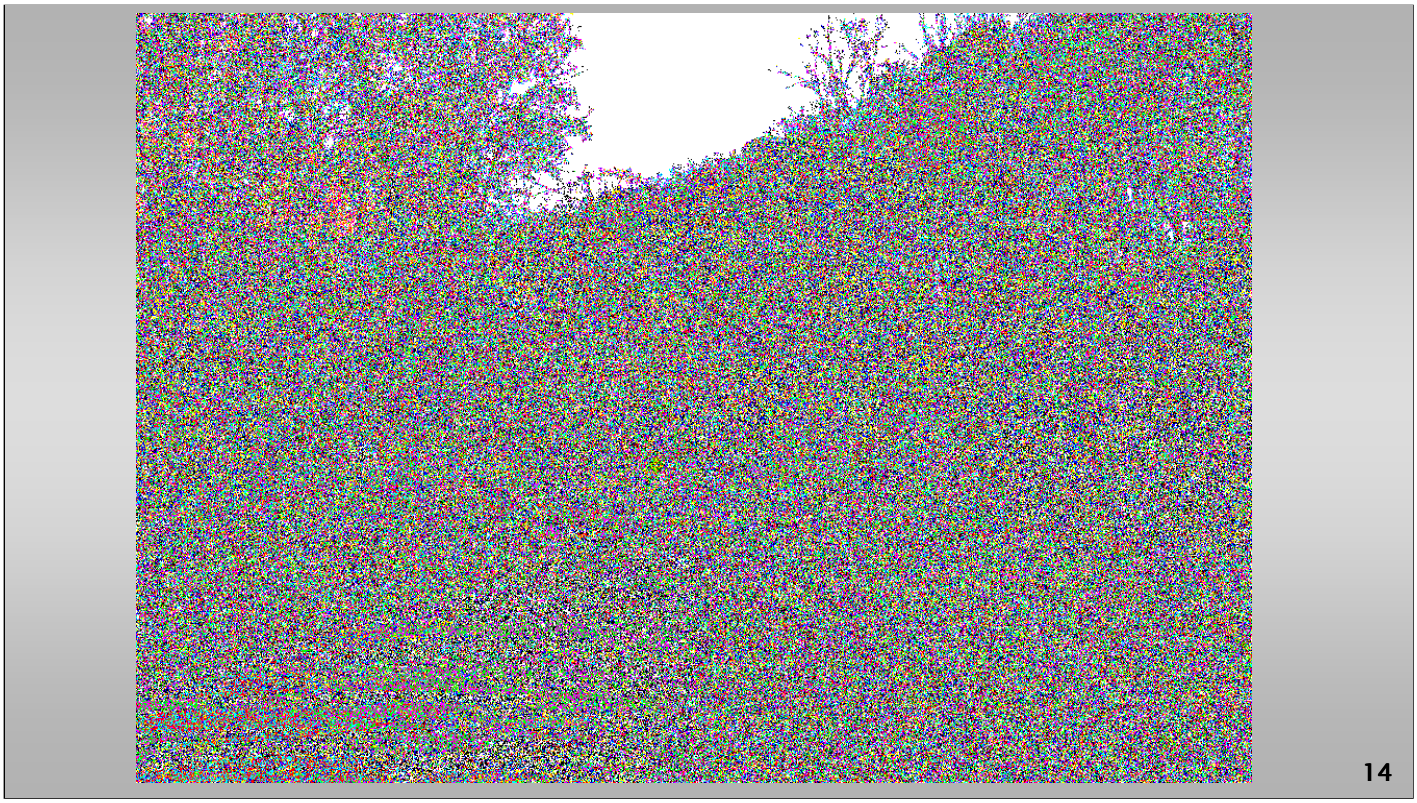
13

Finally the image shown in the grid in the upper-left for “All Bits” and “All Colors” is basically the original image, since it shows the combined color values in all channels and all bits. Any of the images shown in the grid can be clicked on to bring up a new window showing that image full-size. This can be copied and pasted into an image editing program for further analysis or the saving of the image.

The overall idea and philosophy of the *CounterSteg* steganalysis software is to allow the convenient analysis and comparison of before and after images to look for the telltale traces of steganographic software activity and modifications. In many cases these follow similar patterns, and the forensic analysis conducted can make informed conclusions based on typical similar patterns from the various categories of steganographic software currently available. The software available generally falls into several categories, which for the purposes of this talk we will categorize as: 3. the Low, 2. the Mid, and 1. the High.

We will start showing telltale traces from “low" steganographic software, typically this quality of software can be easily detected even without the original cover image for comparison. Even in the case of good steganographic software, having the original cover image on hand makes positive identification of the activity highly probable.

The cover image analysis window shown below clearly depicts the area of white saturation in the upper center of the image (the area open to the sky through the trees). Other color and blue channels show a reasonable distribution of intensities throughout most areas of the image. Ideally for steganograhic activities areas of solid colors, saturation, and low noise between colors and shades should be avoided. This is to circumvent statistical analysis of the steganographic payload carrying image that may indicate a high probability of data carrying modifications.

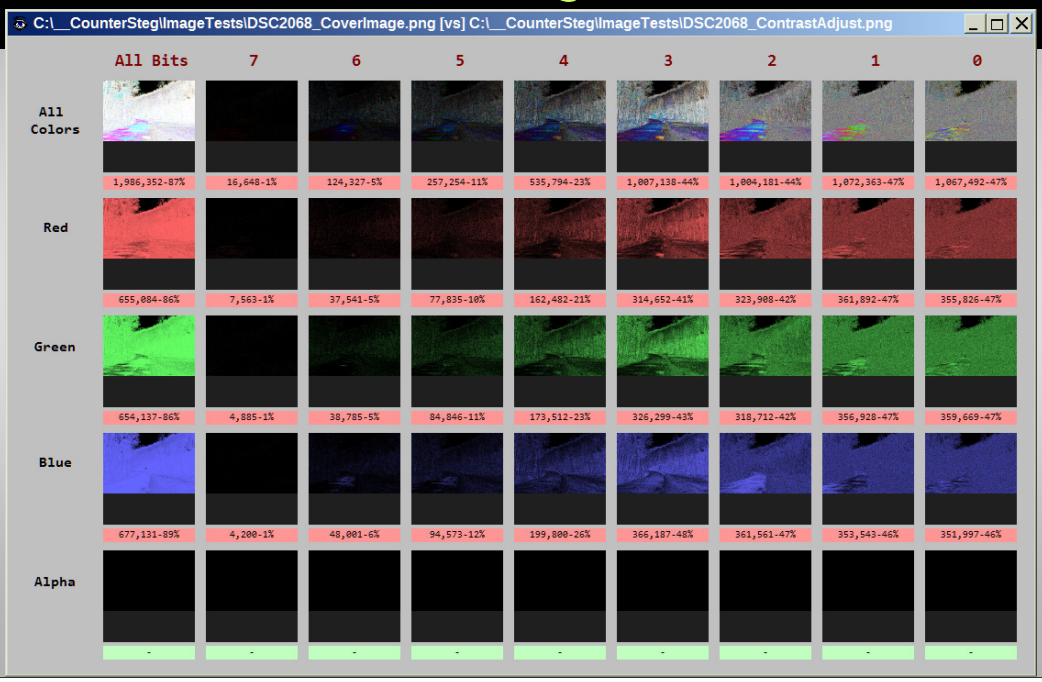


Clicking on the LSB (bit 0) image for all colors, the image below is brought up and is shown as Slide 14. This image shows the bits colored for whether red, green, or blue pixel LSBs are sent to 0 or 1. Various color shades are shown depending on multiple values, however if only one bit is set, such as red, the pixel will remain red as shown in the red bit 0 color image shown. Using the LSB image can give an idea of the relative distribution of LSB values in the image in various colors. In general, for many photographs the distribution will be largely random except for saturated or solid color areas of the image. This image is the starting point for our further analysis, as modifications to the LSB bit plane are particularly evident in certain qualities of steganographic software available.

In addition to *CounterSteg* providing bit by bit and color analysis for a single image, the software also allows image comparisons using a similar breakdown. The comparison of the original cover image to a contrast adjusted image is shown below. In this breakdown, only differences between pixels, colors, and bits are shown. Below each comparison image is shown the total number of bit, color, and/or pixel differences (depending on the analysis), as well as the percentage of changes relative to the total number of changes possible.

This analysis window allows quick and convenient comparison between an assumed original cover image, and the assumed steganographic payload image. The specific type, location, and scale of the differences can help to clearly identify steganographic activities that have been performed on the image, the likely image payload size, and perhaps even the likely specific steganographic software that has been used in certain cases. In the following narrative we will detail forensic profiles of various software packages and their results.

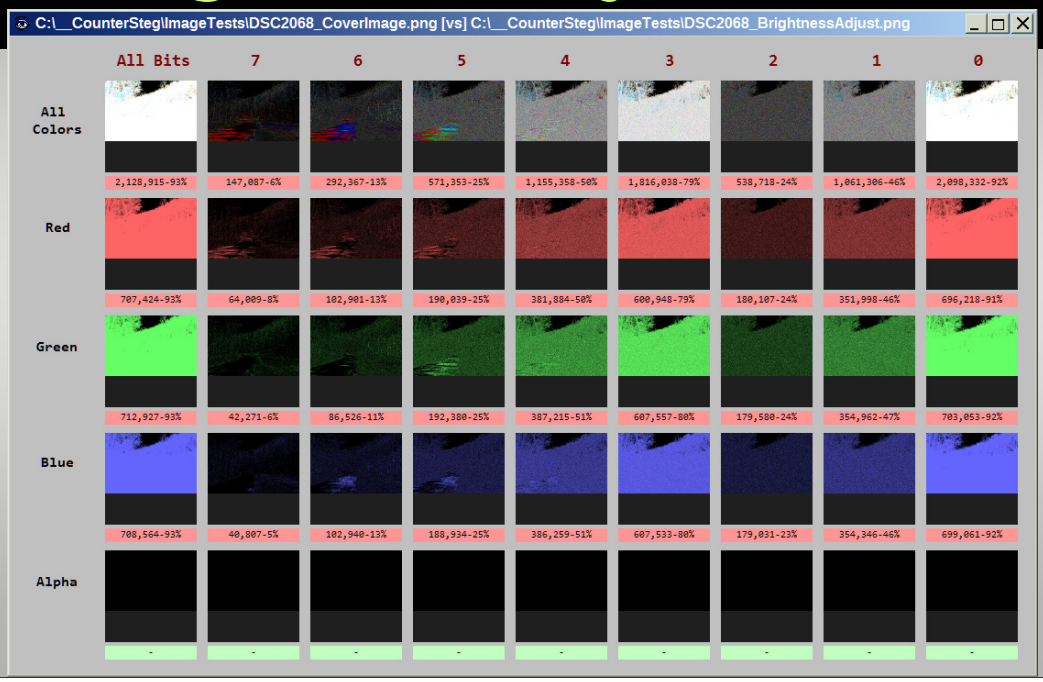
Contrast Adjustment



15

As an example of a standard image adjustment in comparison, below is the results of comparing the original cover image to a modified image with a small contrast adjustment. In addition to large modifications in the LSB plane we are also seeing large to small modifications in all other bit planes, and in all colors. In general, if two visually similar versions of an image exist — seeing changes like this in comparison would generally not indicate steganographic activities.

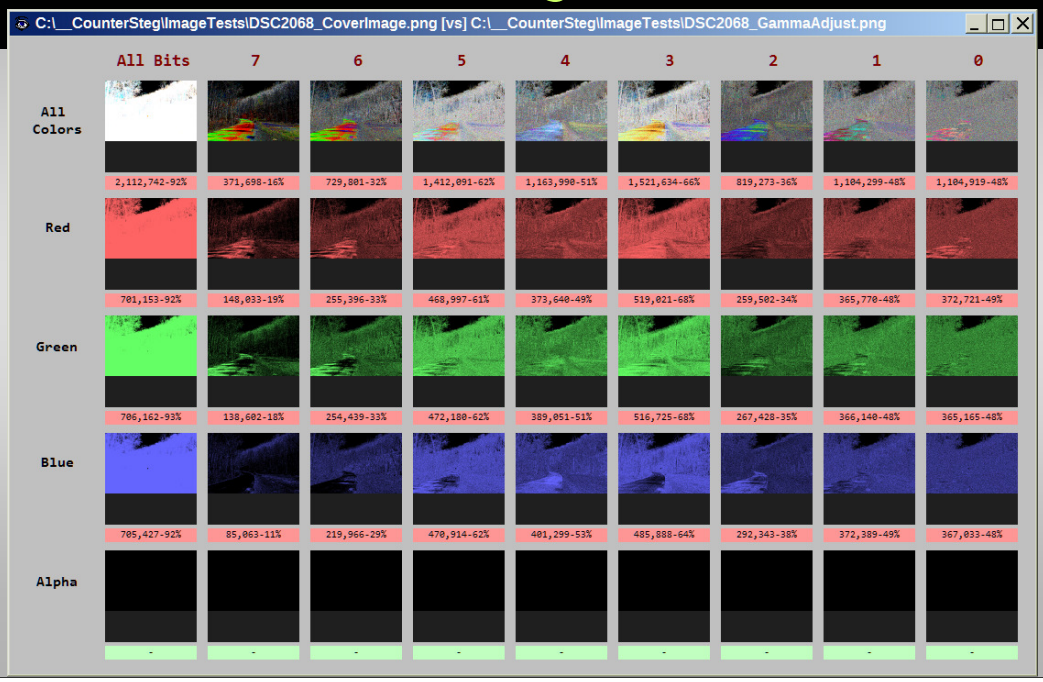
Brightness Adjustment



16

Similar to the small contrast adjustment, a brightness adjustment of the original image results in a comparison profile showing large modifications in all colors and in all bit planes.

Gamma Adjustment



17

The Gamma adjustment color filter compresses or stretches various colors and would result in a comparison profile similar to the one shown below. All colors and all bit planes are greatly modified. The differences between the two images will be visually apparent.

In conclusion — standard image filters, such as those found in Photoshop, or other image editing software, for contrast, brightness, and gamma adjustment, do not generally yield comparison results similar to what we will depict in the following narrative for steganogrphic related changes. This is with one exception — BitCrypt — which should still be detectable using other digital forensic clues and analysis.

The “Low”

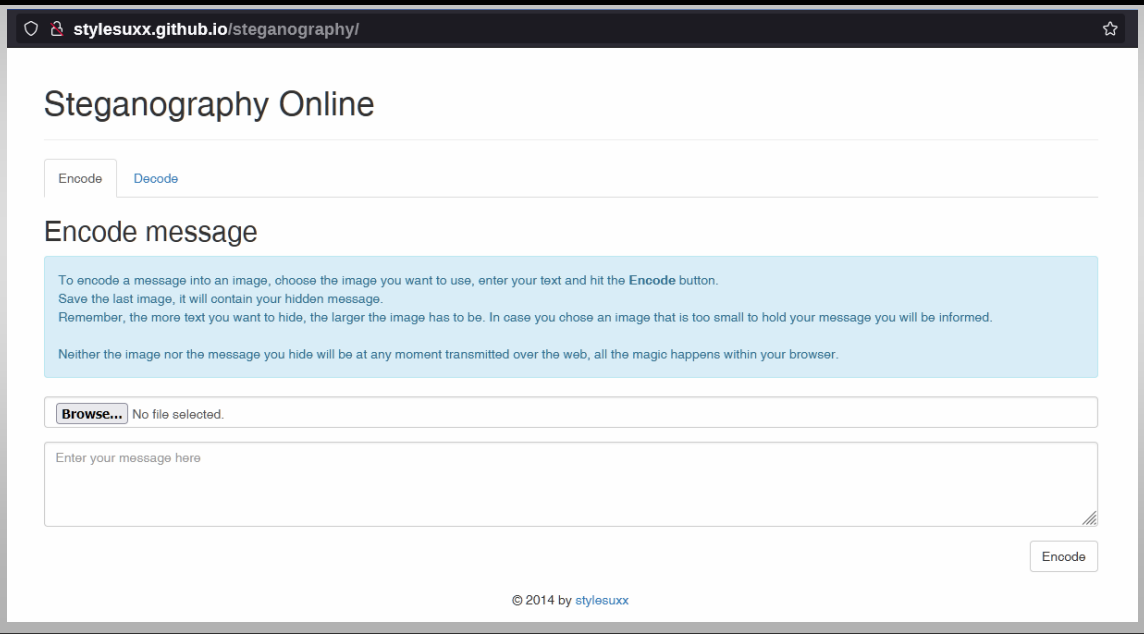
- Steganography Online – http://stylesuxx.github.io/steganography/
- StegoShare – http://stegoshare.sourceforge.net/
- Geocaching Toolbox – https://www.geocachingtoolbox.com/index.php?page=steganography
- DevFarmSteganography – https://devfarm.it/steganography/

18

THE LOW

Typically, ugly software takes a "let it rip" attitude and just shoves the data to be embedded into the LSB image color plane, without regard to how easily this would be possibly detected using a forensic analysis.

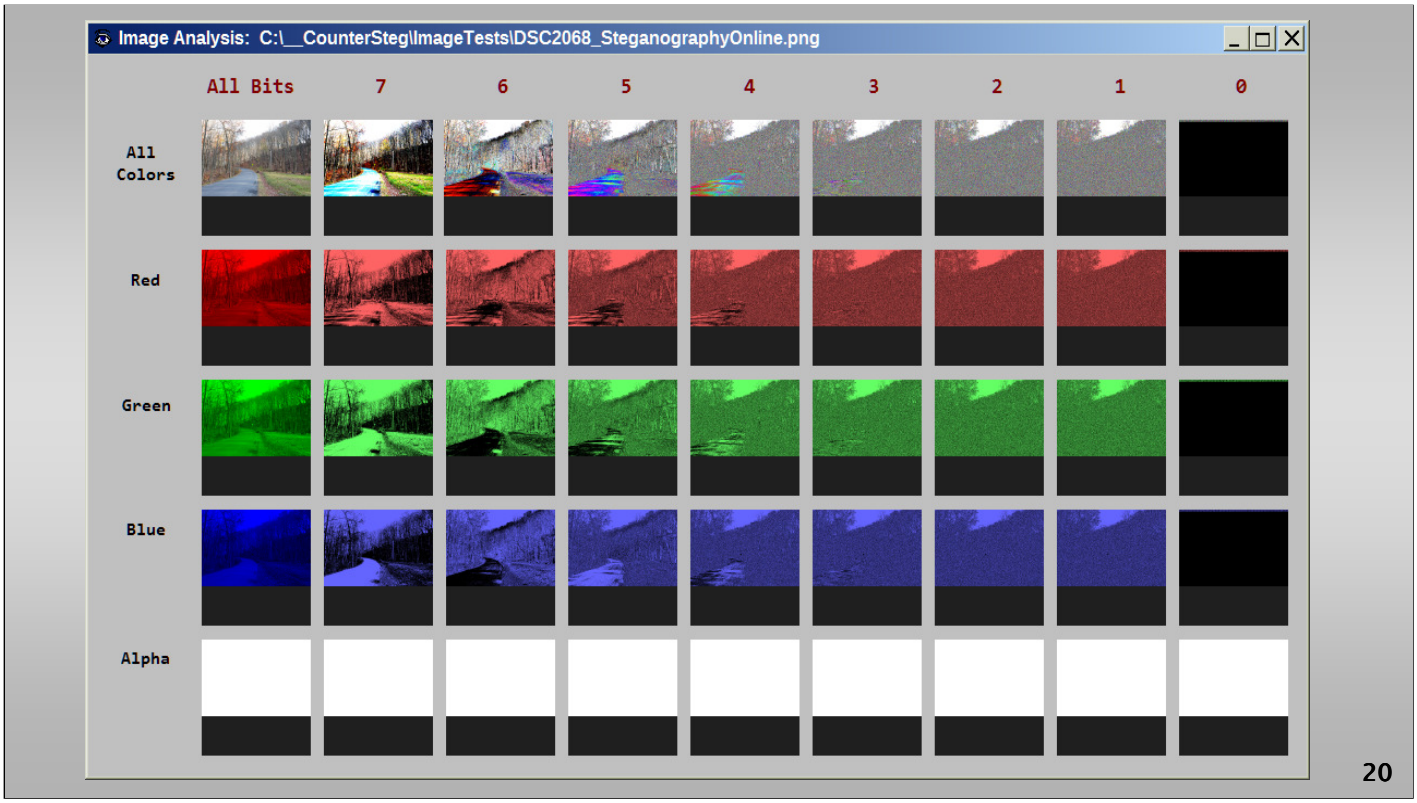
Steganography Online



19

Steganography Online

http://stylesuxx.github.io/steganography/



This particular software apparently first completely zeroes out the LSB bit plane in all colors, and then encodes the data into a narrow strip at the top of the image. This is visually evident in the analysis image shown below — where all LSB bits are simply set to zero, except for the data at the top. This is the least sophisticated and most naïve of all the steganographic software we will be analyzing — the ugly.

Figure. Analysis of image created by Steganography Online

In the comparison image shown below, you can see all bit planes are identical from bit plane 7 to 1. All modifications take place in bit plane 0 (LSB), which is in fact first set to zero, and then the data is encoded.

Figure. Comparison of cover image to image created by Steganography Online.

The image shown below is an expansion of all colors in bit plane 0, the data containing strip at the top is easily evident. In particular, the area of saturated white pixels at the top is completely overwritten. This software will be easily forensically detectable and identifiable in usage.

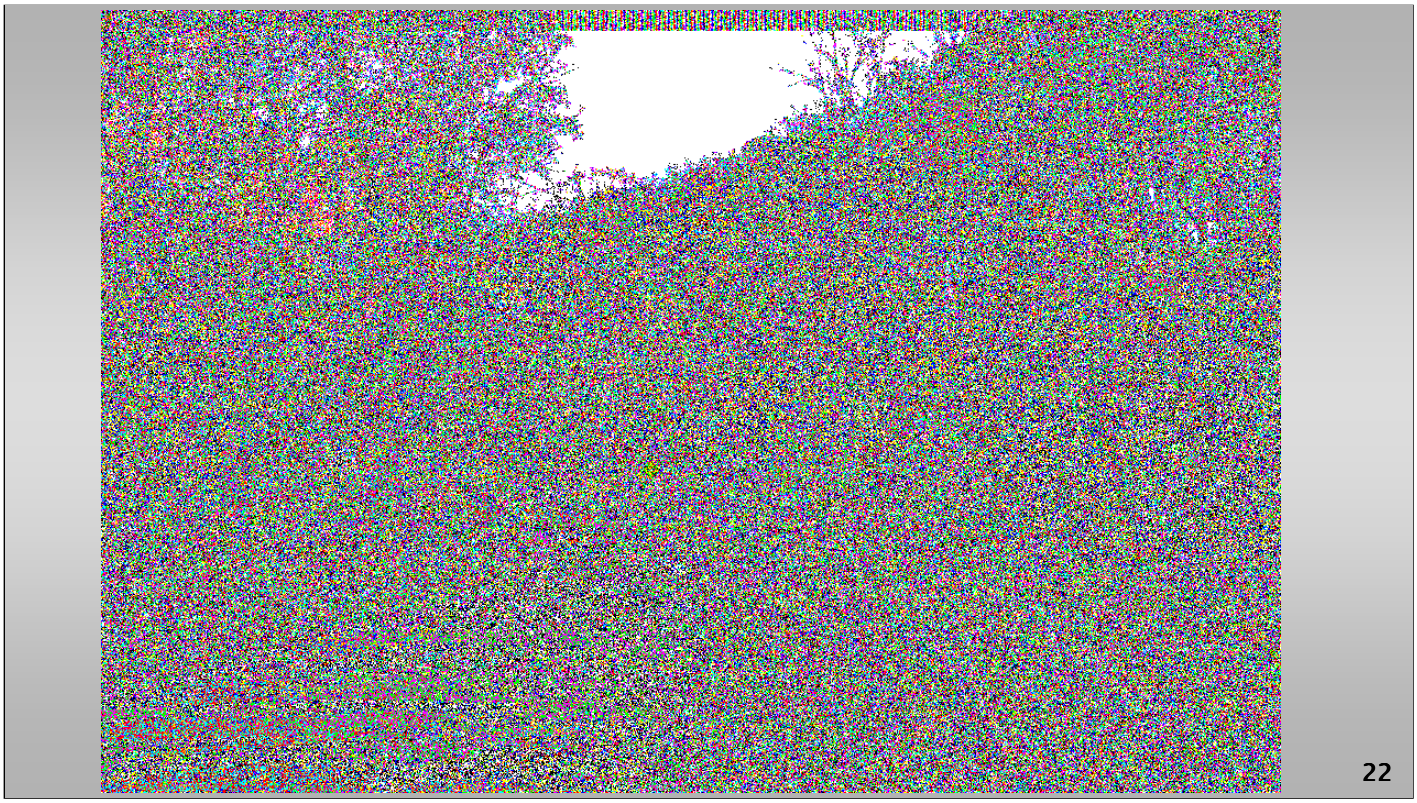


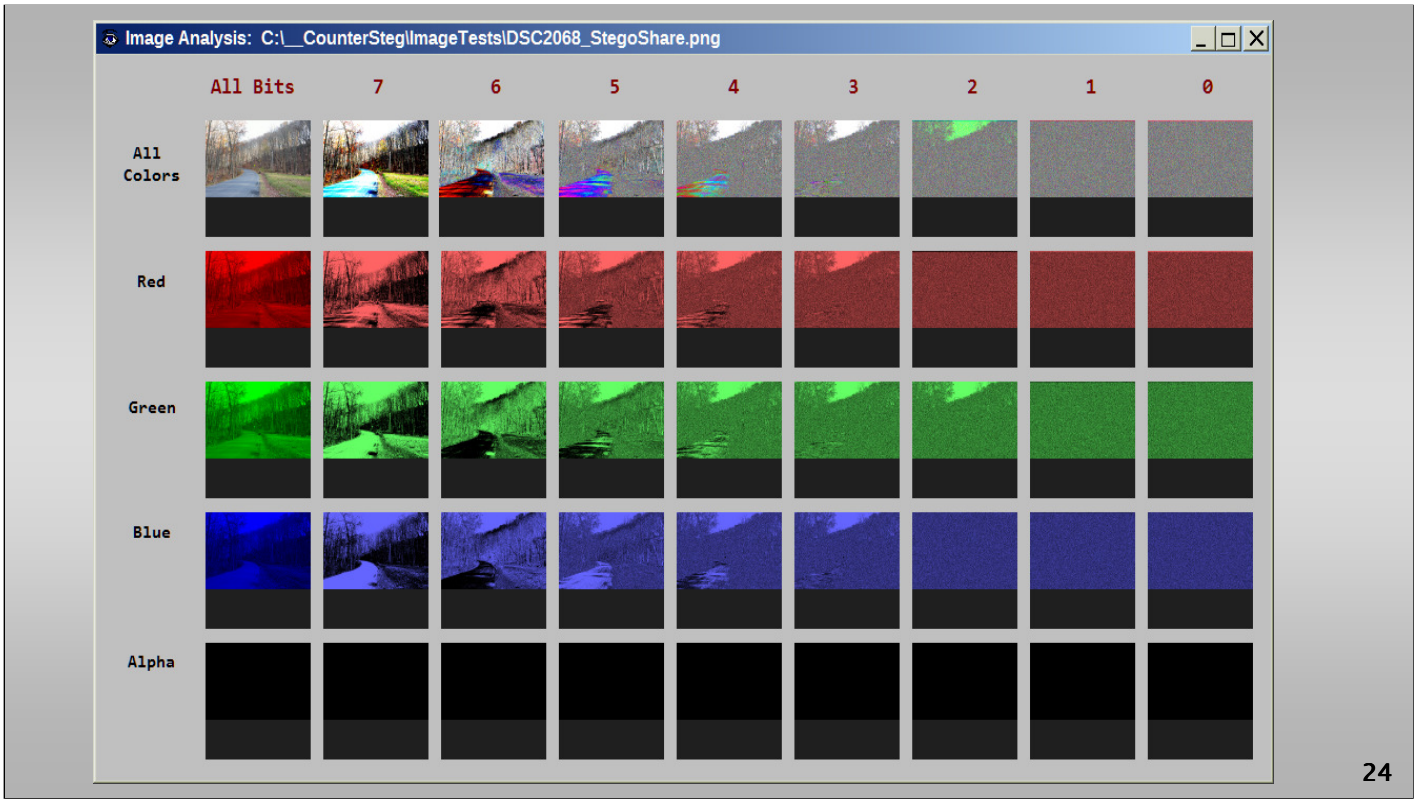
Figure. LSB changes with Steganography Online image

StegoShare

23

StegoShare

http://stegoshare.sourceforge.net/



The next ugliest software uses a similar approach, however, does not completely zero out the LSB plane. In addition, it makes modifications to bit plane 1 and 2, for reasons unknown. Other bit planes remain unchanged.

Further, some type of narrow strip of information is embedded in the top center of the image. Analyzing the image in bit planes 0, 1, and 2, as shown below, clearly depicts the modifications.

Figure. StegoShare image analysis.

In comparing the images, you can see the large percentage (33-50%) of modifications made to bit plane 0, 1, and 2, with the exception of green in bit plane 2.

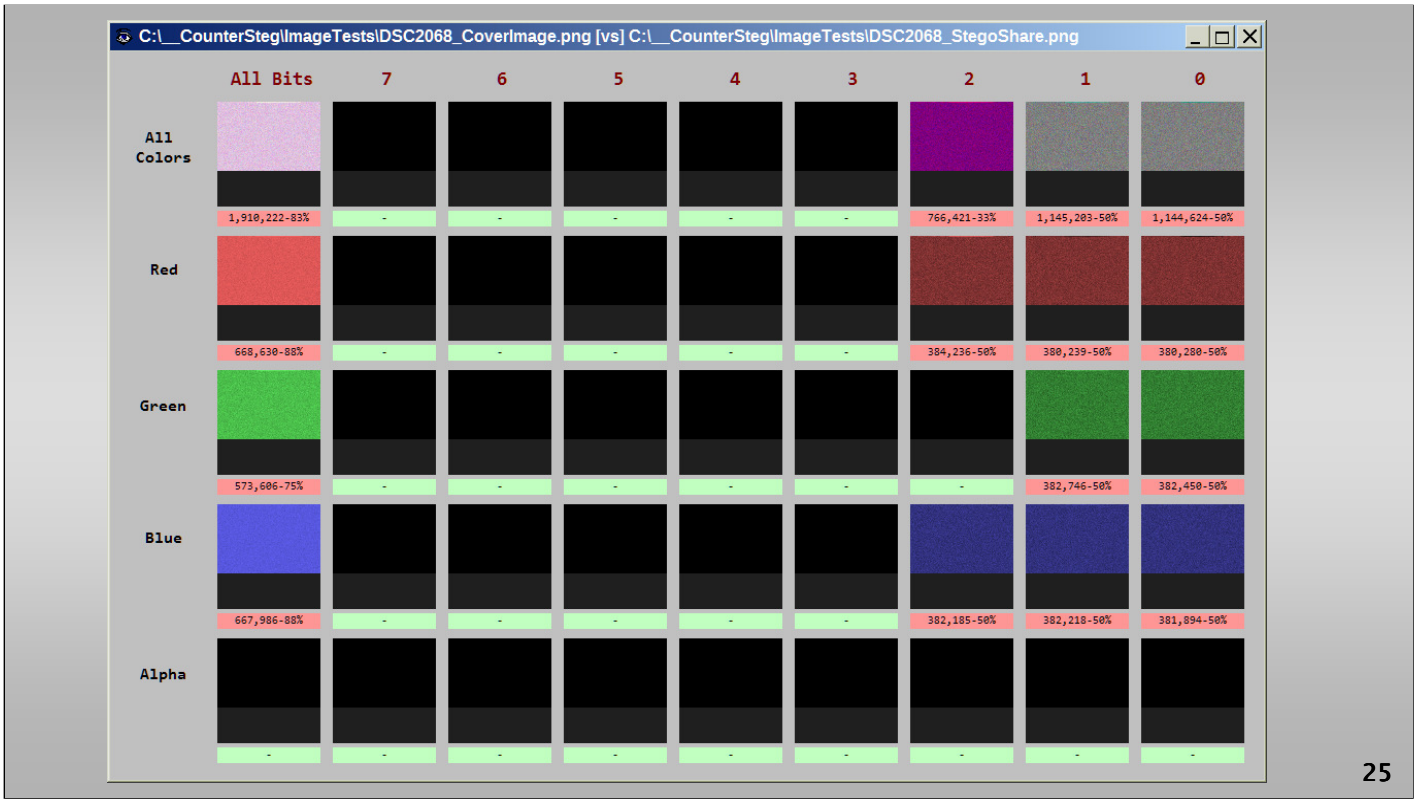


Figure. StegoShare image comparison.

The expansion of the changes to all colors in all bit planes image shown below shows the narrow strip of information also embedded into the top center of the image. This is shown in Slide 23 that follows.

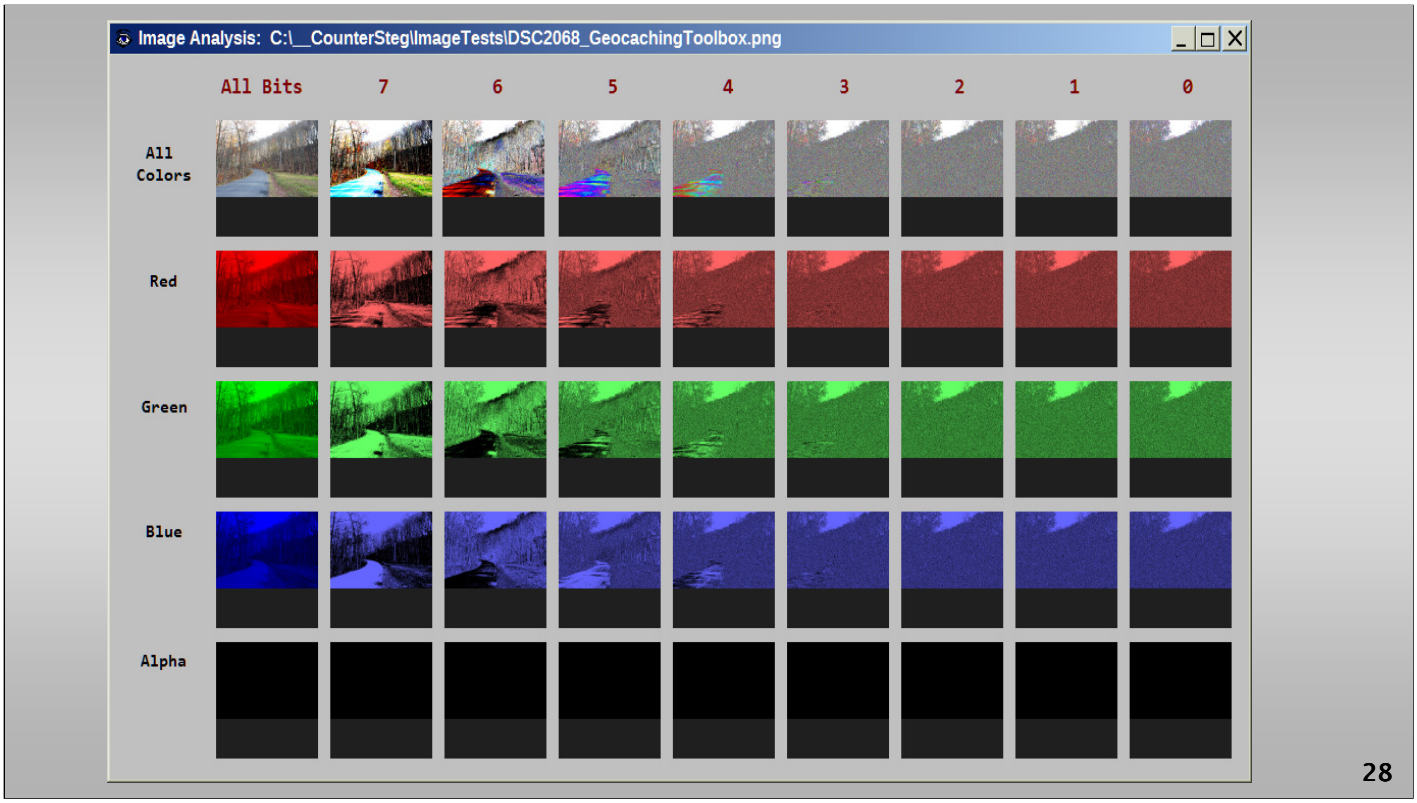
Figure. StegoShare LSB values.

Geocaching Toolbox

27

Geocaching Toolbox

https://www.geocachingtoolbox.com/index.php?page=steganography



This software only alters the LSB bit plane in all three colors, however it includes all the data into a narrow strip at the bottom of the image. This would be easily detectable using RS statistical analysis as changes to the bit patterns in only a small fraction of the image (the portion containing the data).

The overall analysis of the image, as shown below in Slide 25, does not indicate much forensically. In this case we also need the original cover image for comparison, which then makes the positive conclusion evident.

Figure. Geocaching Toolbox image analysis.

Below in Slide 26 is the comparison image — showing the data payload embedded to the narrow strip in the bottom of the image. However, due to the small amount of pixel changes (1%), it is likely at least the software compresses the data before embedding.

Compressing the text data before embedding typically can reduce the size of the necessary modifications to the image by 80 to 90%. As a result, higher quality steganographic software will always compress data before engaging in the image embedding process.

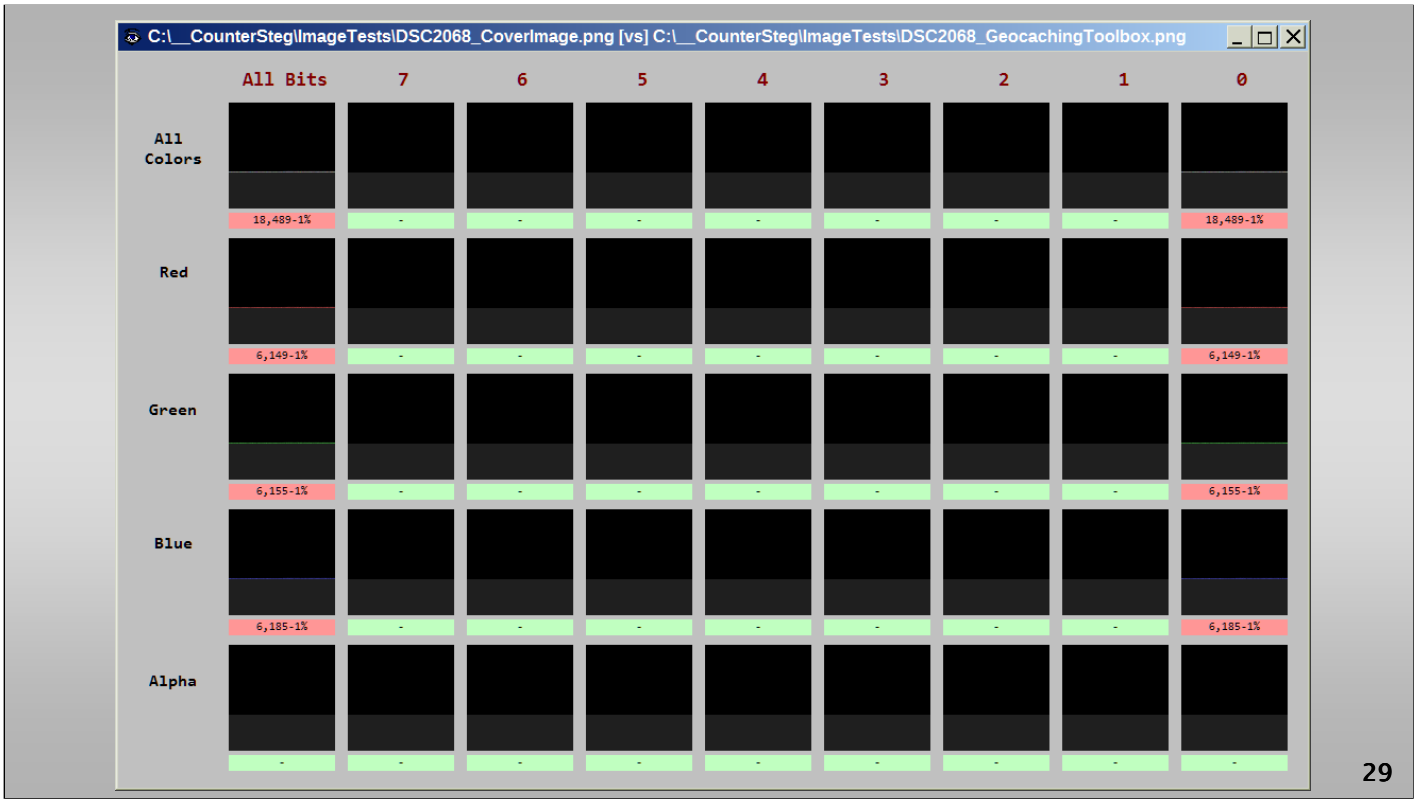
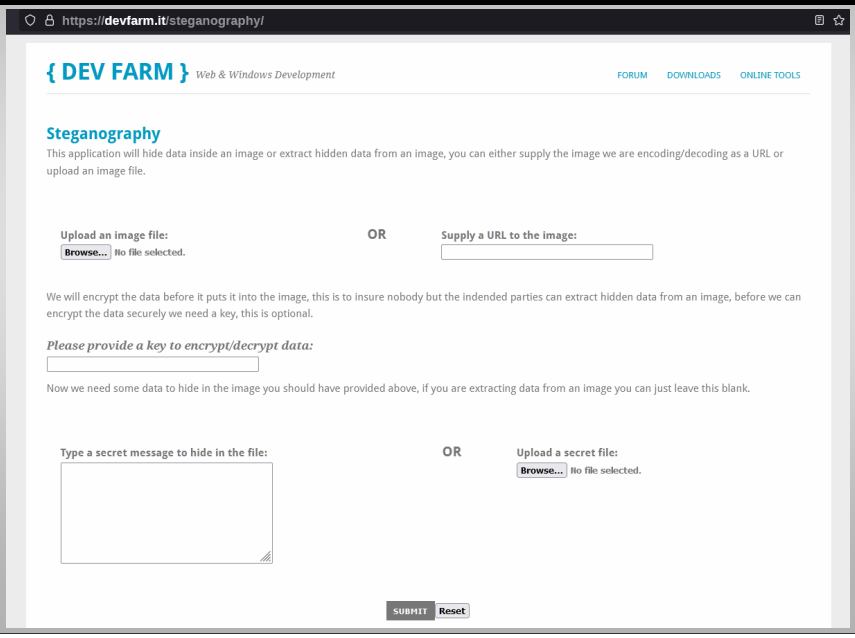


Figure. Geocaching Toolbox image comparison.

The narrow strip is clearly visible in Slide 27 at the bottom.

Figure. Geocaching Toolbox LSB changes.

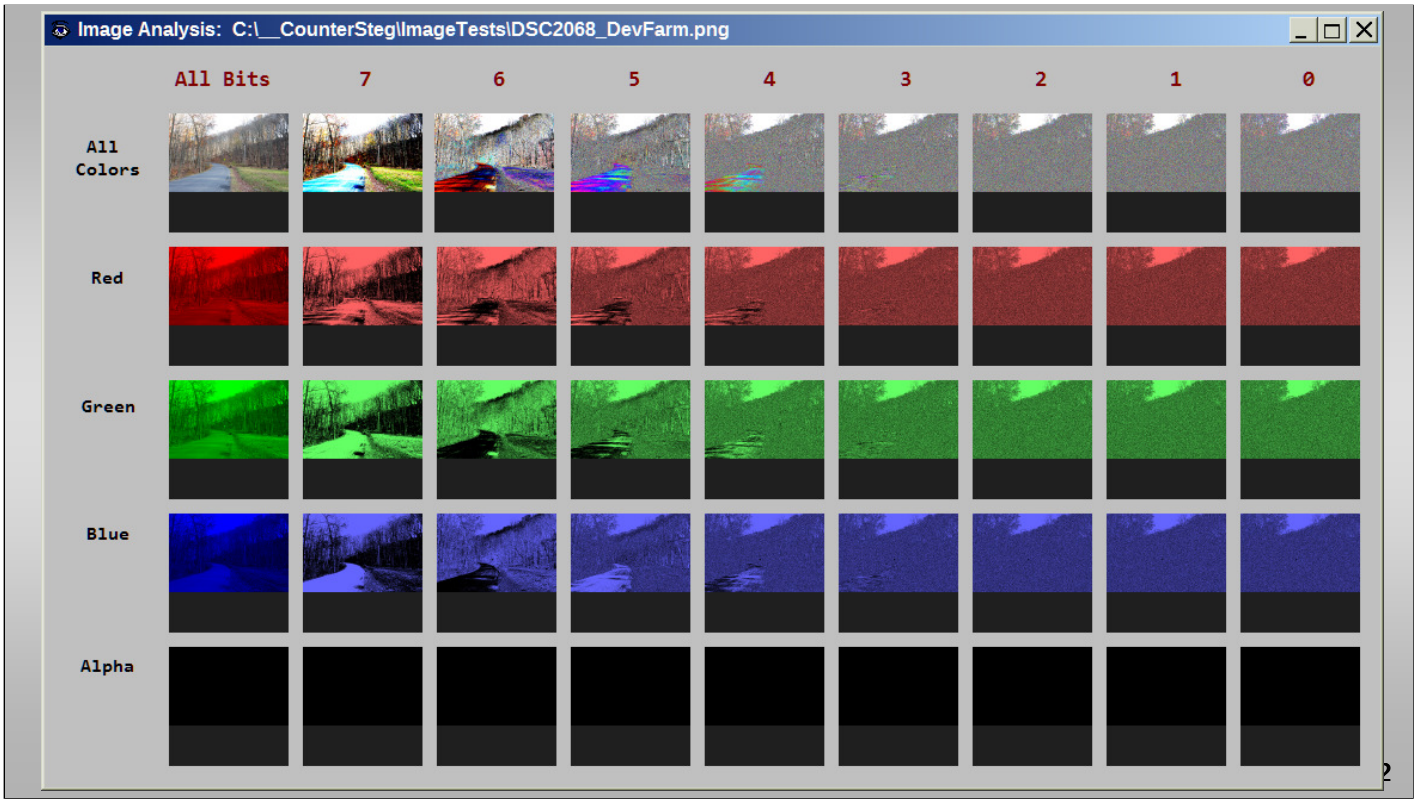
Dev Farm Steganography



31

DevFarmSteganography

https://devfarm.it/steganography/



This software is comparable to the previous software, Geocaching Toolbox, and may make use of the same steganographic embedding libraries.

The modifications to the image are very similar, with only slight differences in the amount of pixels modified.

Figure. DevFarm Steganography image analysis.

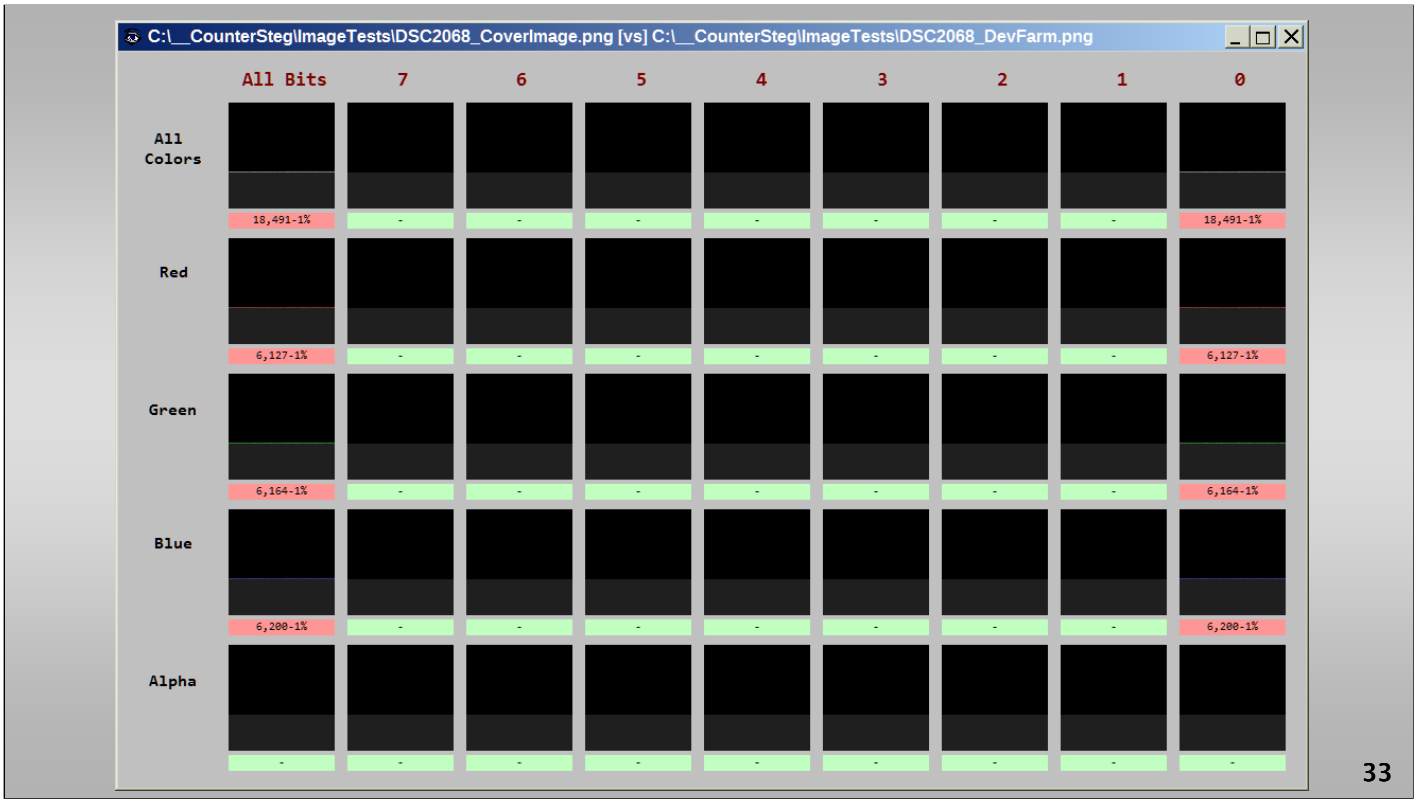


Figure. DevFarm Steganography image comparison.

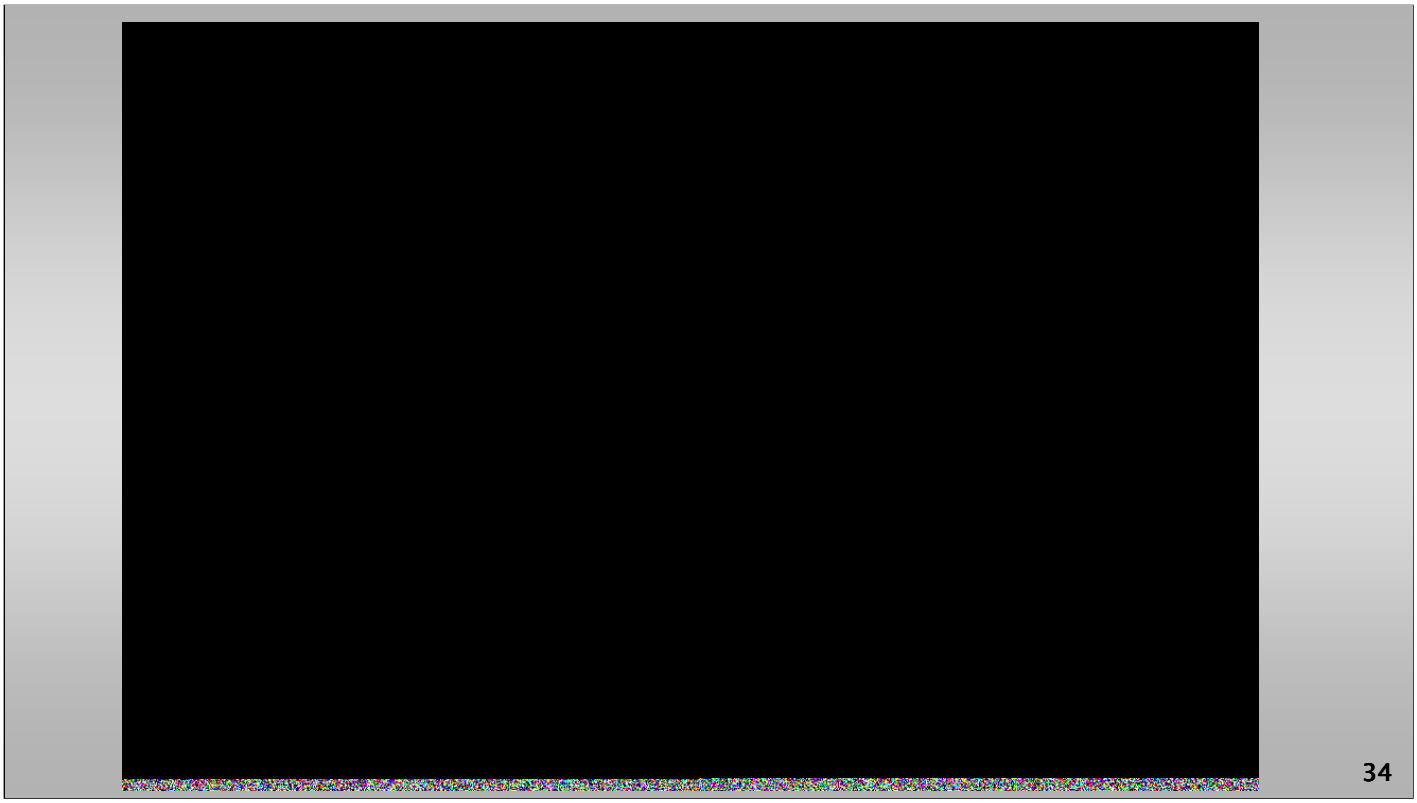


Figure. DevFarm Steganography LSB image changes.

The “Mid”

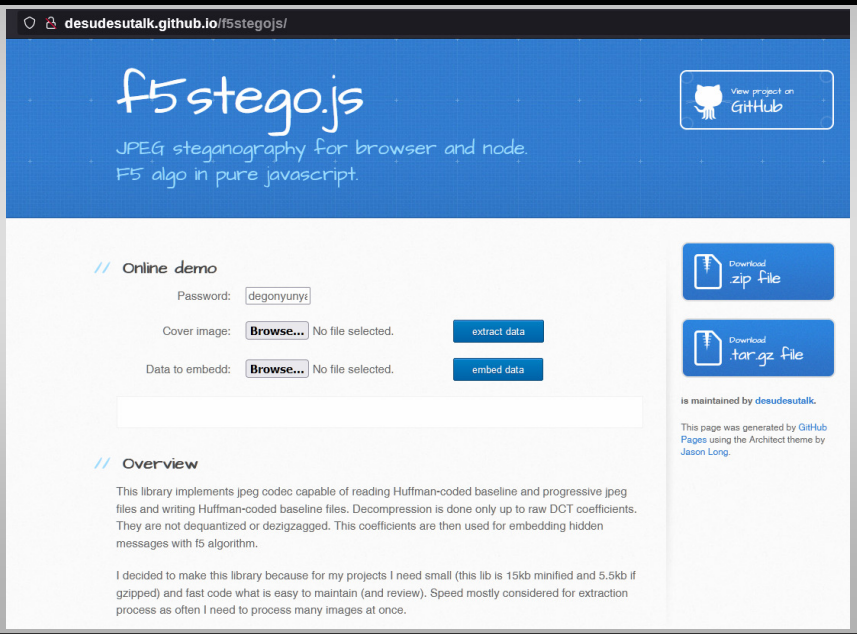
- f5stego.js – http://desudesutalk.github.io/f5stegojs/
- BitCrypt – http://bitcrypt.moshe-szweizer.com/
- OpenPuff – http://embeddedsw.net/OpenPuff_Steganography_Home.html

35

THE MID

The best software is an improvement over the ugly, at least narrow strips of pixels are not apparently encoded (and more easily detected), however security shortcomings are still evident.

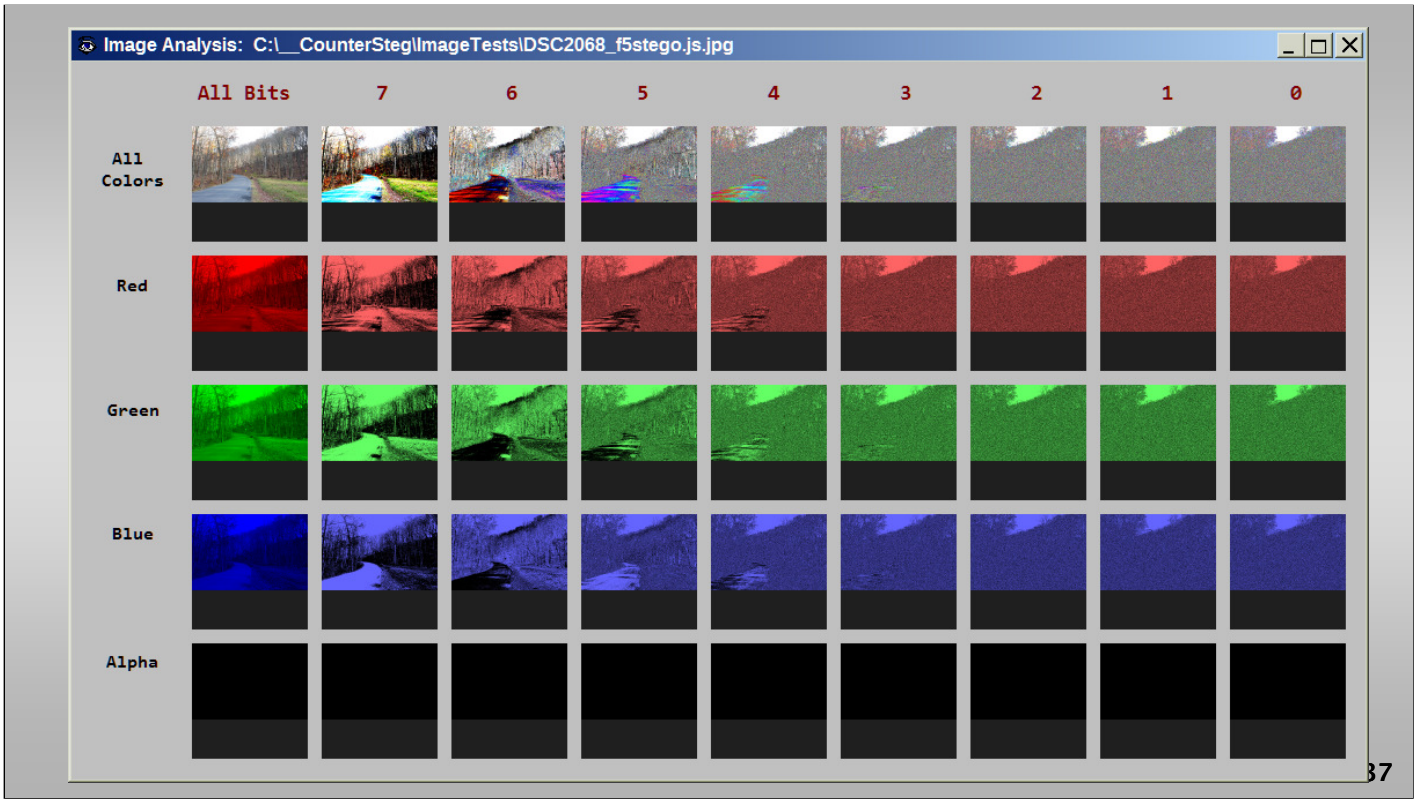
f5stego.js



36

f5stego.js

http://desudesutalk.github.io/f5stegojs/



This software seems to take the unique approach of ignoring the LSB and simply encoding data depending whether color values are odd or even. Even though many of the pixel colors are modified through multiple bit planes, some of the color values remain unchanged. This is evident in the comparison image of all bits in all colors shown as Slide 35.

An analysis of the image shown in Slide 34 does not show undue pixel modifications or strips. However, due to the large amount of color changes, visual differences will be evident between the original and modified image.

Figure. f5stegojs image analysis.

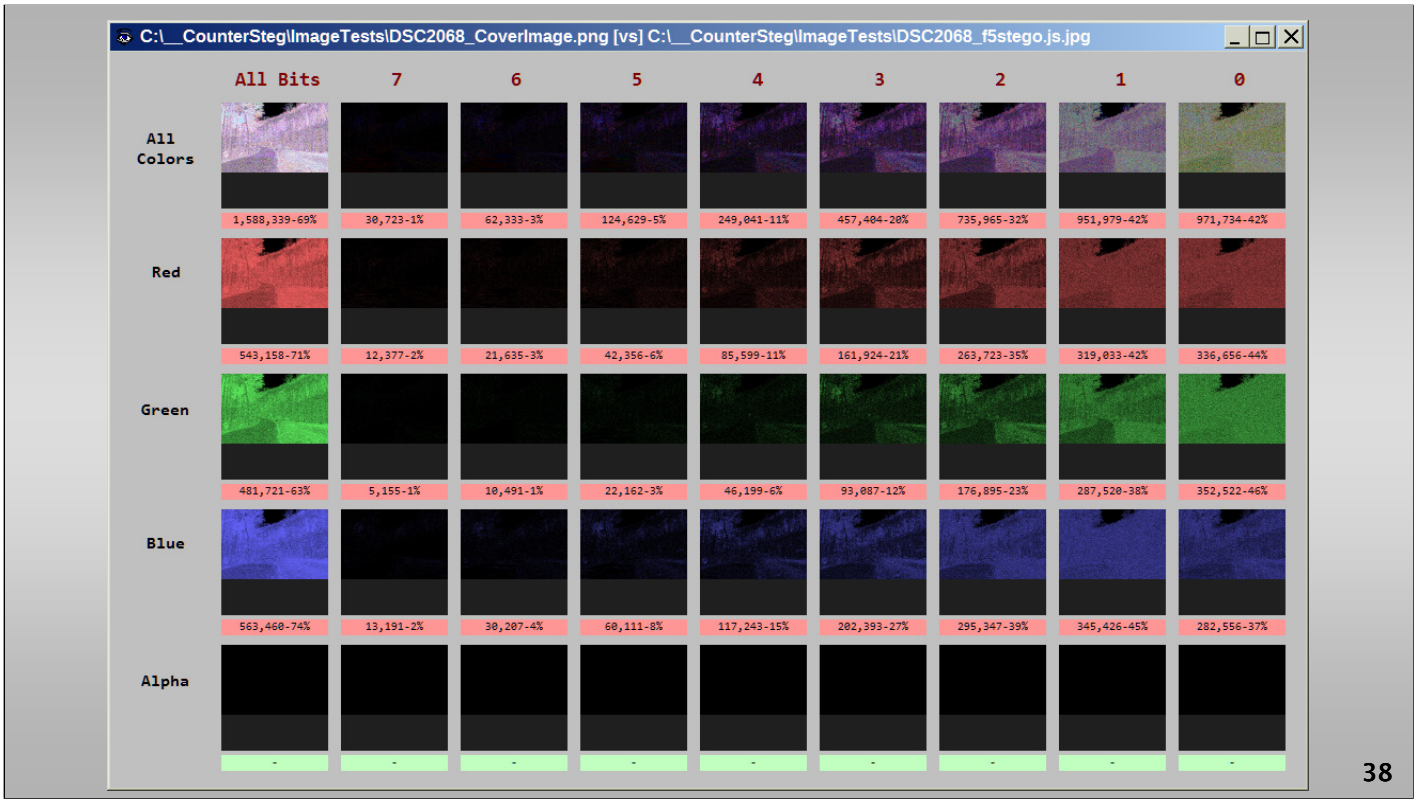


Figure. f5stegojs image changes.

Below in Slide 35 is the comparison analysis. Notice that all bit planes and all colors are modified. On first analysis, this would appear to be very similar to the contrast adjustment shown previously, however in that case virtually all pixels are modified, except for the saturated white area.

In this case, many pixels remain unchanged — hinting at the possibility of "all bit" encoding. In other words, data is encoded by overall color intensity value for the respective color channel, red, blue, or green (in the range 0 to 255).

Depending on whether the color intensity is odd or even, this indicates the value of the bit for that particular color channel. Three bits can be encoded for each pixel in this fashion. However, forensic analysis of the steganographic image easily identifies a payload because of the fact that only selective pixels are modified.

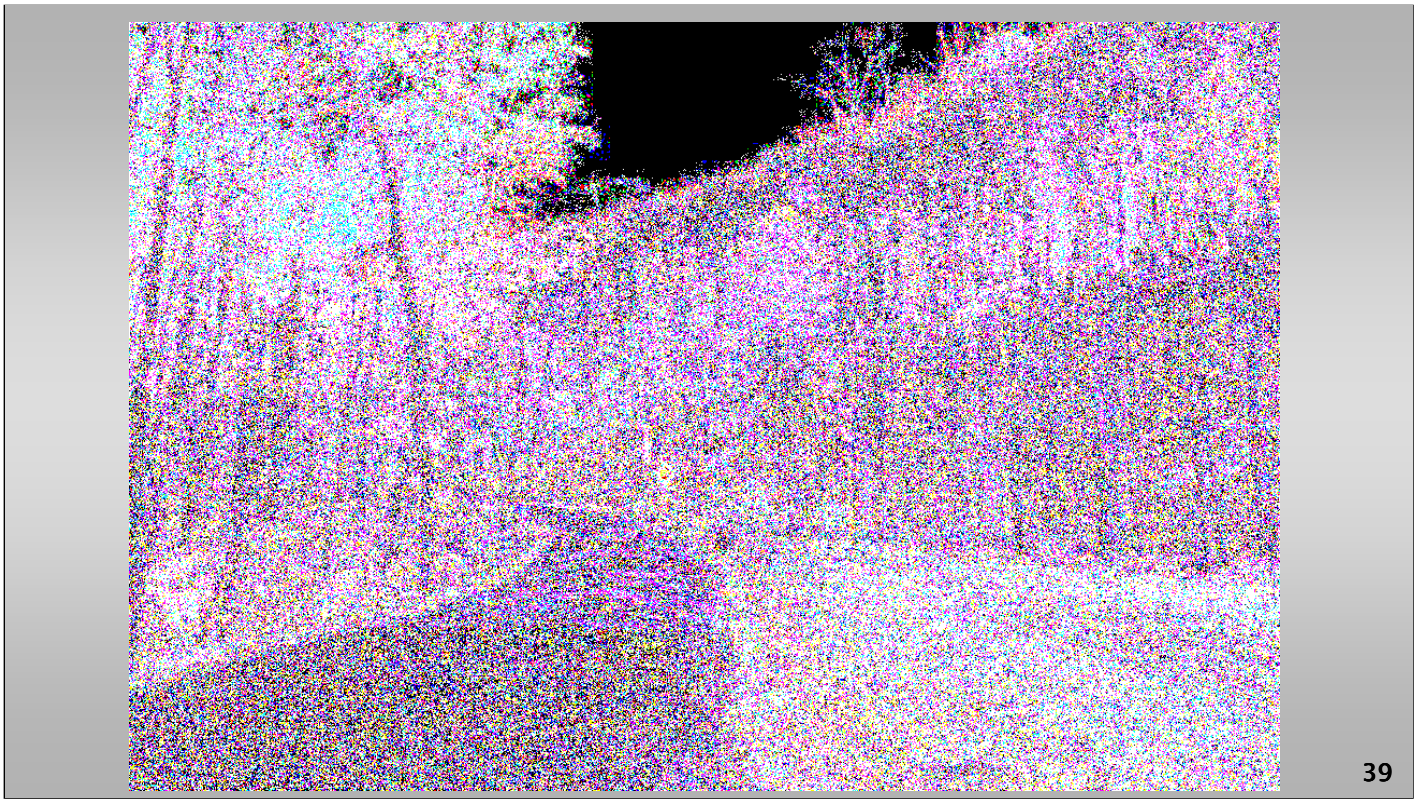


Figure. f5stegojs LSB image changes.

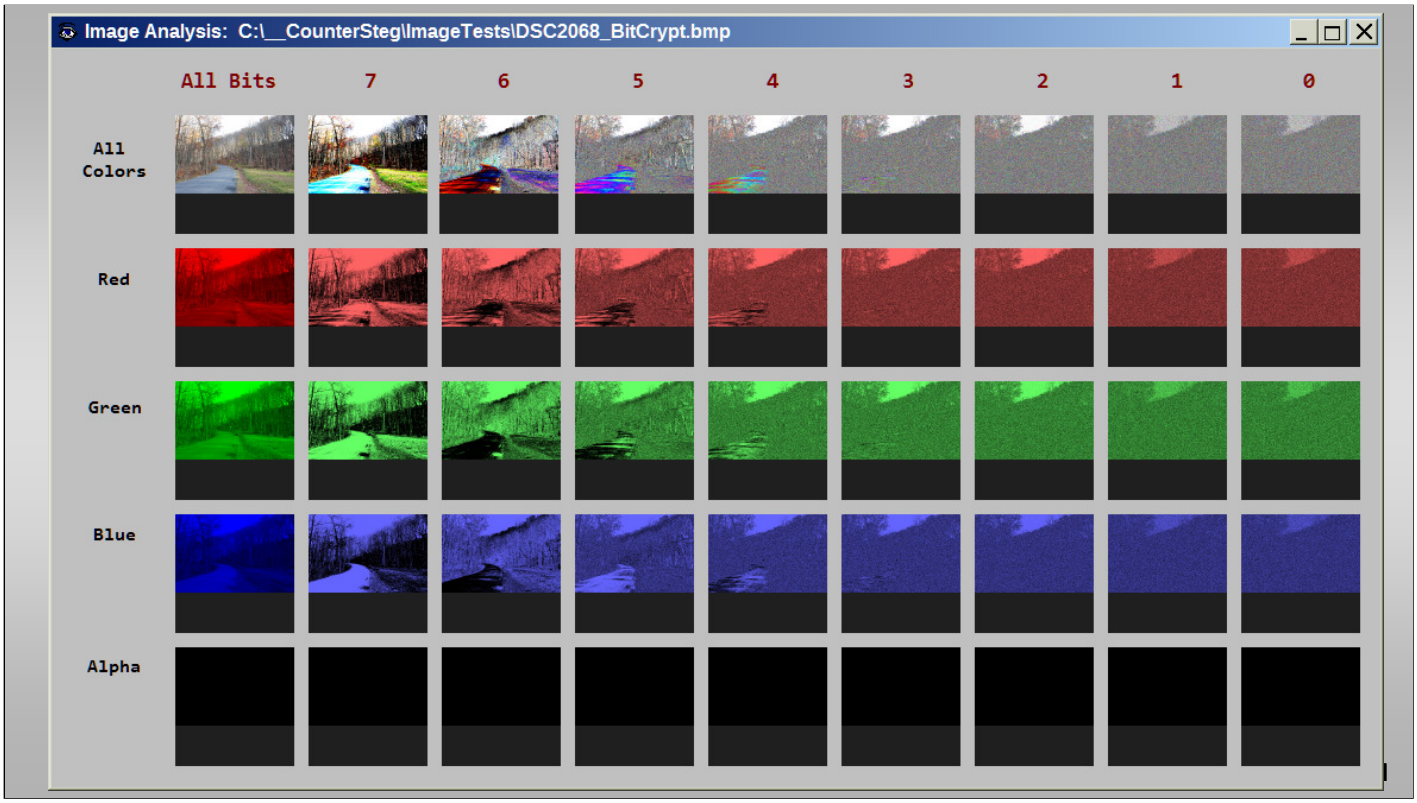
As shown below in Slide 36 in the expansion of the comparison image in all color LSB bits — many pixels remain unchanged in an apparent random fashion. It is likely saturated pixels are avoided (an aspect of quality in this software), and random noise is added in areas of the image not needed for further data encoding. Overall, 69% of the pixel color values in this image have been modified.

BitCrypt

40

BitCrypt

http://bitcrypt.moshe-szweizer.com/



BitCrypt fails in the area of modifying saturated white pixels, and makes this software more easily detectable. The sophisticated forensic analyst will be aware that camera CMOS sensors typically saturate to maximum values in bright areas, such as the sky, will not vary between colors pixel to pixel.

These block areas will be fully saturated and will remain so through the area of the similar object, such as the overcast sky. Modifications to pixels in these areas will be telltale signs for software image message or data carrying modifications.

An overall analysis of the image shows the bit 0 and 1 modifications in the area of saturation. Bits 2-7 appear to be largely unchanged in the area of saturation.

Figure. BitCrypt image analysis.

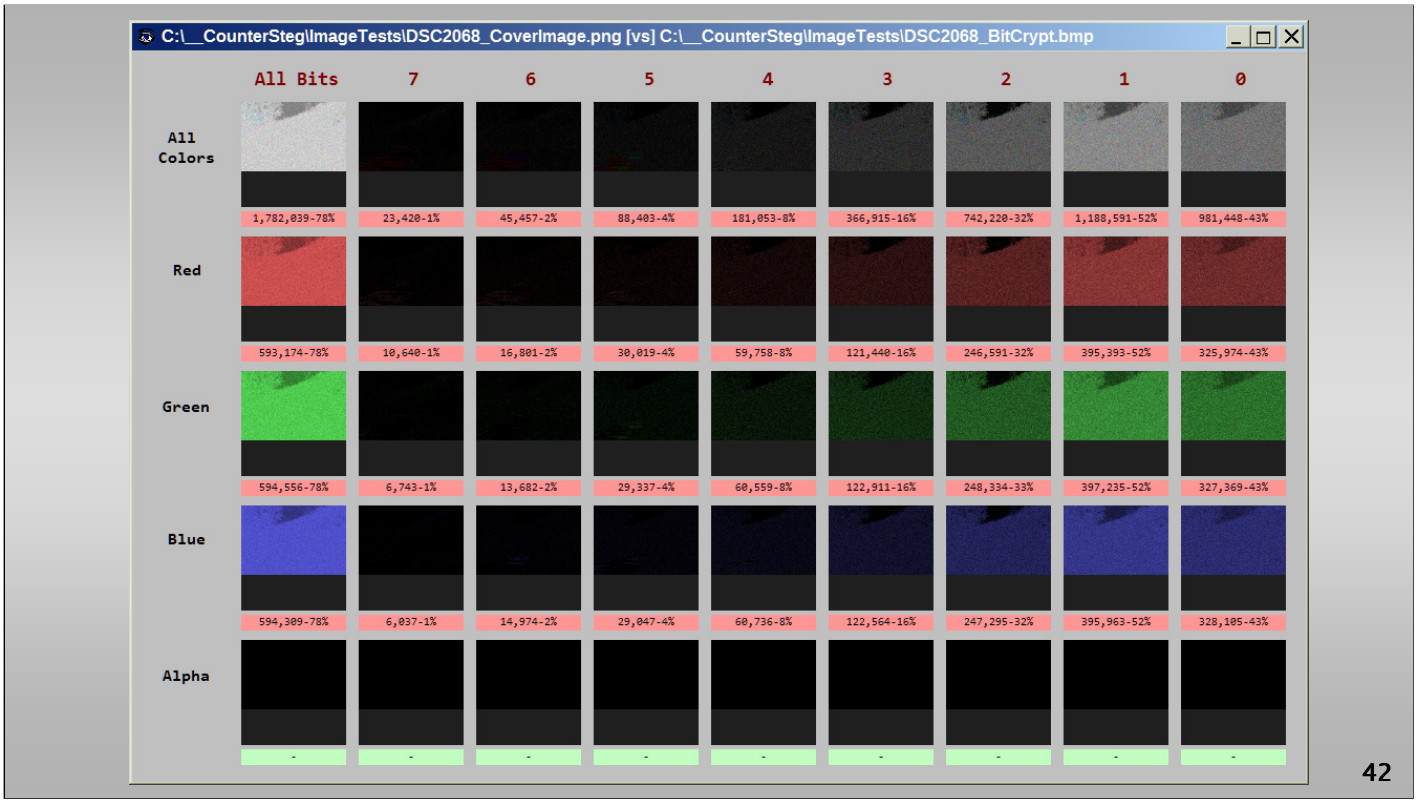


Figure. BitCrypt image comparison.

Comparison with the cover image shows the broad modification of pixel colors in bits 0 through 7, however the area of saturation is avoided above bit 1. This most likely is a software coding implementation to create a similarity with a standard image processing function, such as brightness, contrast, or gamma adjustment. However in those cases the changes in the saturated area will propagate up through bit 7, and including bit 7.

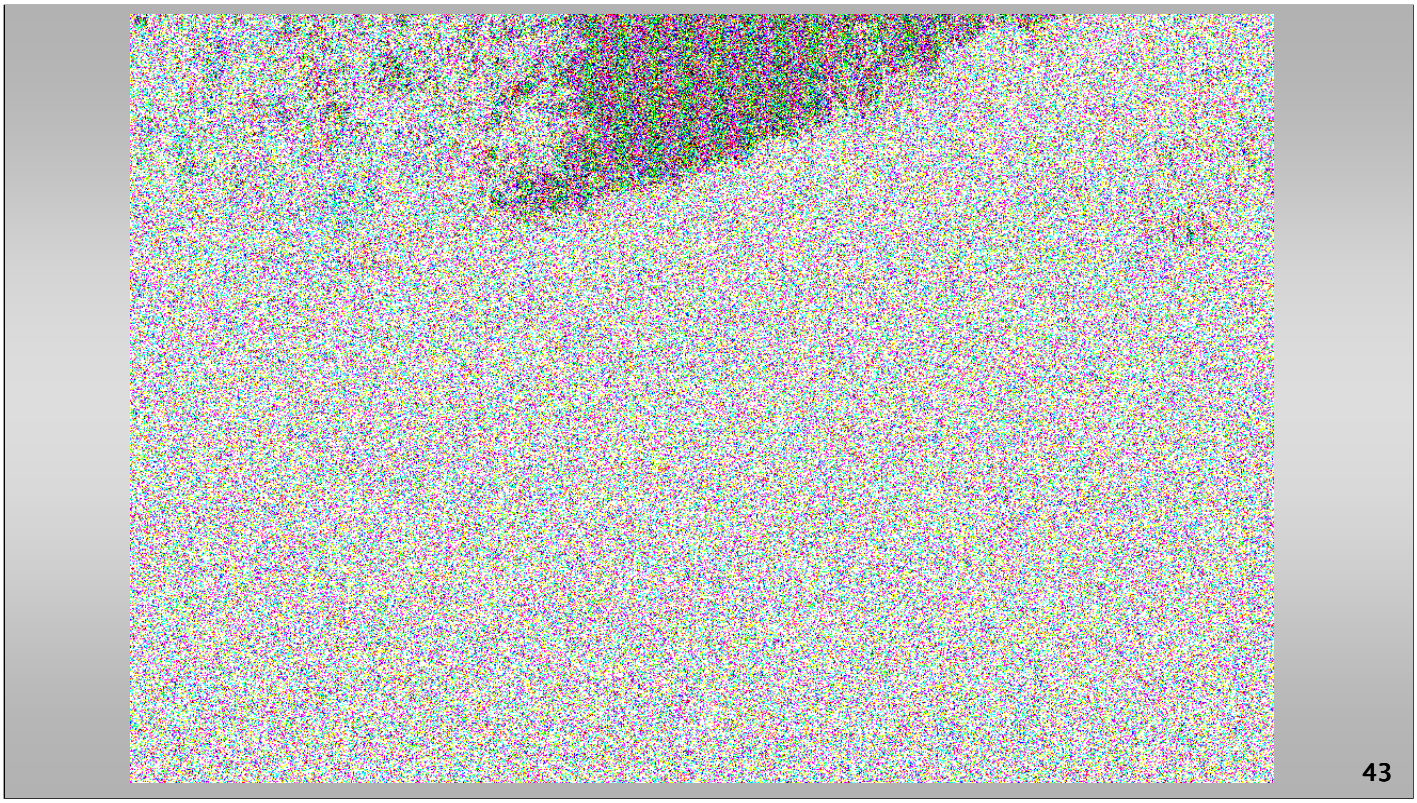


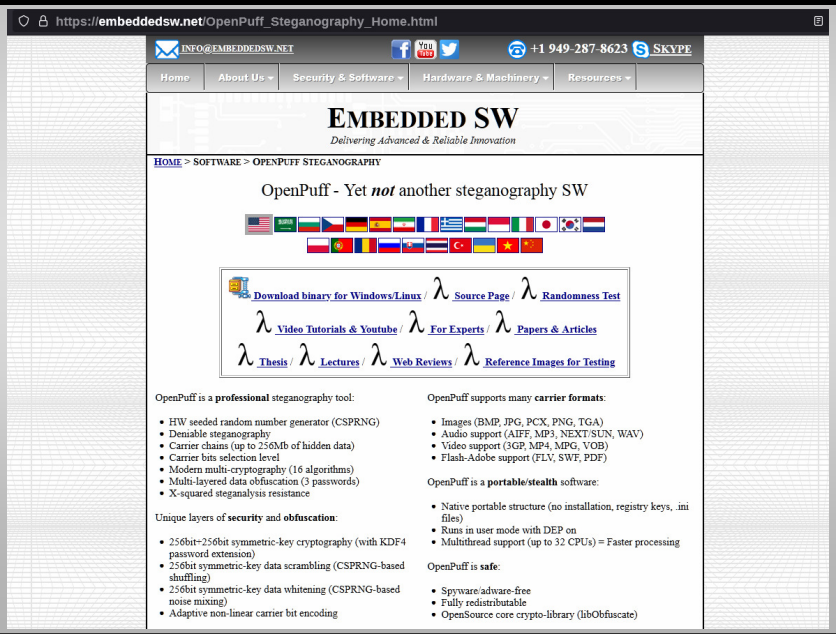
Figure. BitCrypt LSB image changes.

Further, the expansion of the bit 0 in all colors graphic analysis depicts the seemingly random dispersal of LSB bit changes, except for the relative lack of changes in the saturated area. This particular forensic pattern for steganographic activity is indeed unique.

While subtly different from standard image processing comparison results, the bit modification pattern should still be able to be classified and identifiable.

Further test images should be created and comparisons conducted in the future to analyze and identify with further clarity the modifications BitCrypt makes to images.

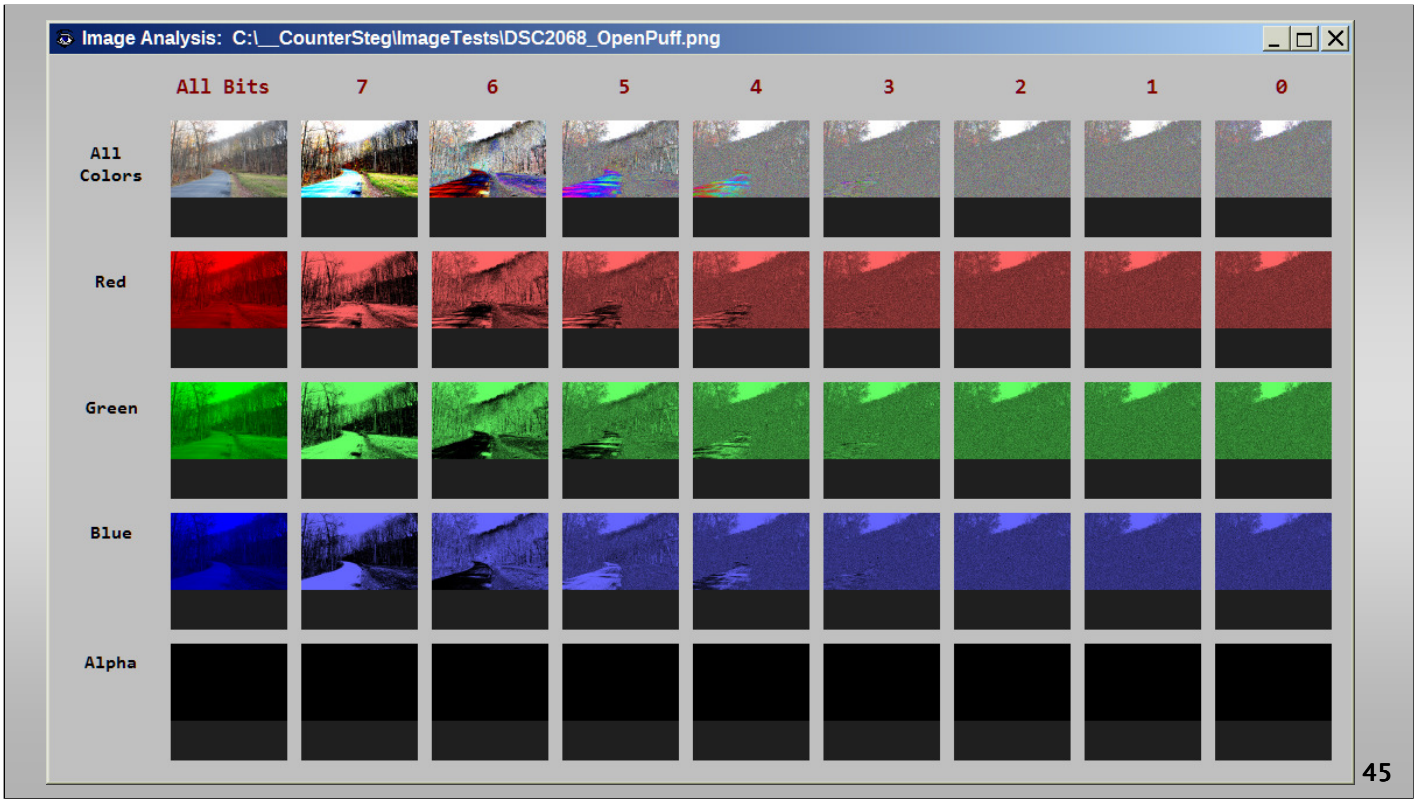
OpenPuff



44

OpenPuff

http://embeddedsw.net/OpenPuff_Steganography_Home.html



While completely avoiding the saturated pixel area, which is good, OpenPuff attempts to modify far too many pixels to not be detectable especially with a comparison image in hand. The software only modifies pixels in the LSB plane, making it virtually undetectable visually.

However, overall 12% of the pixels are modified in the image — compare this to 1% even with less sophisticated programs. Statistical analysis of the image will reveal changes to typical photograph LSB bit patterns in typical similar photos with similar CMOS sensors.

Overall analysis of the image, as shown below in Slide 42, does not reveal any particular fine points for analysis — making OpenPuff the best of the bad software for Steganography in this analysis.

Figure. OpenPuff image analysis.

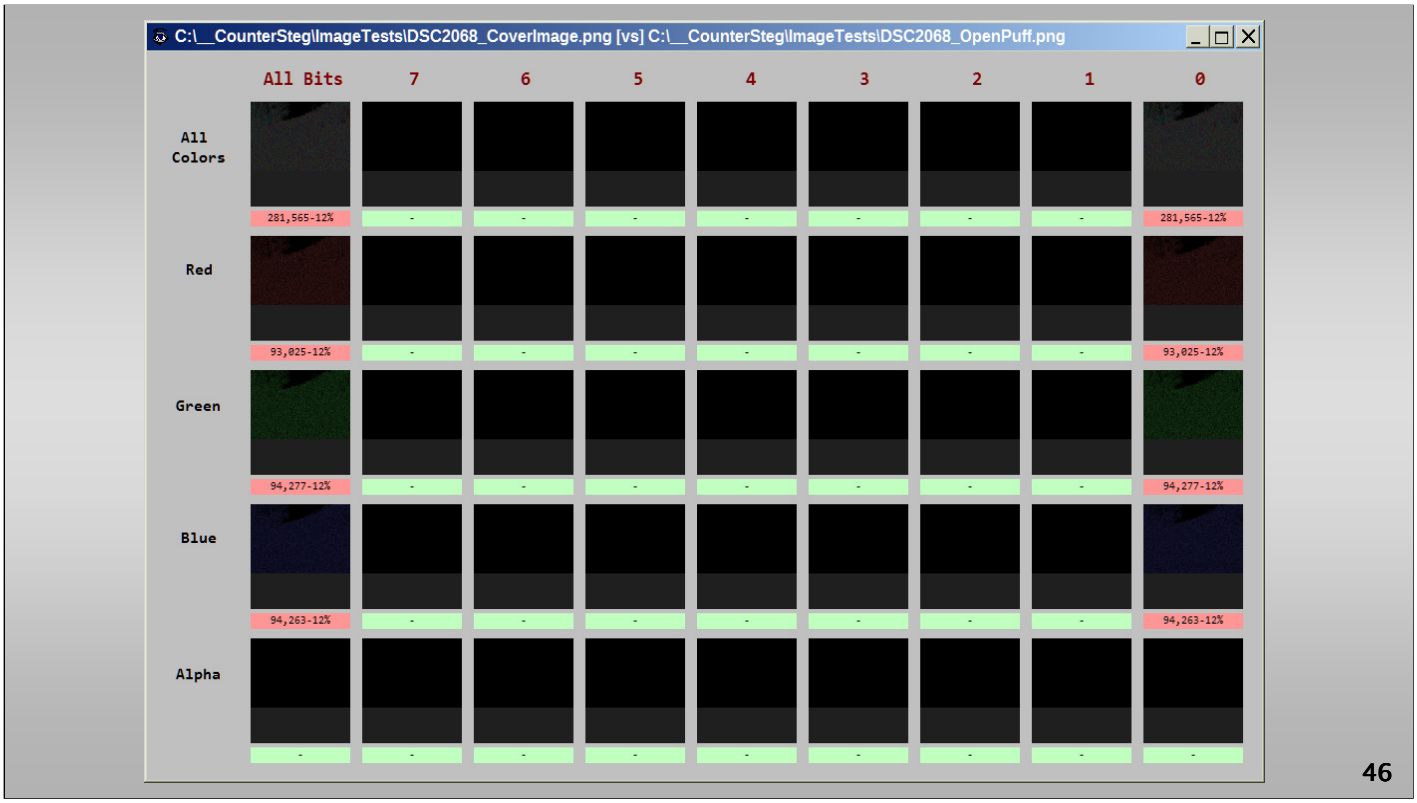


Figure. OpenPuff image comparison.

As shown below in Slide 43, OpenPuff only makes changes to the LSB values in the image, but does these to far too great an extent at 12%. This leaves the payload image vulnerable to statistical analysis techniques.

With a comparison image in hand, since only LSB values are modified, it is highly probable to conclude steganographic payload has been embedded into the image.

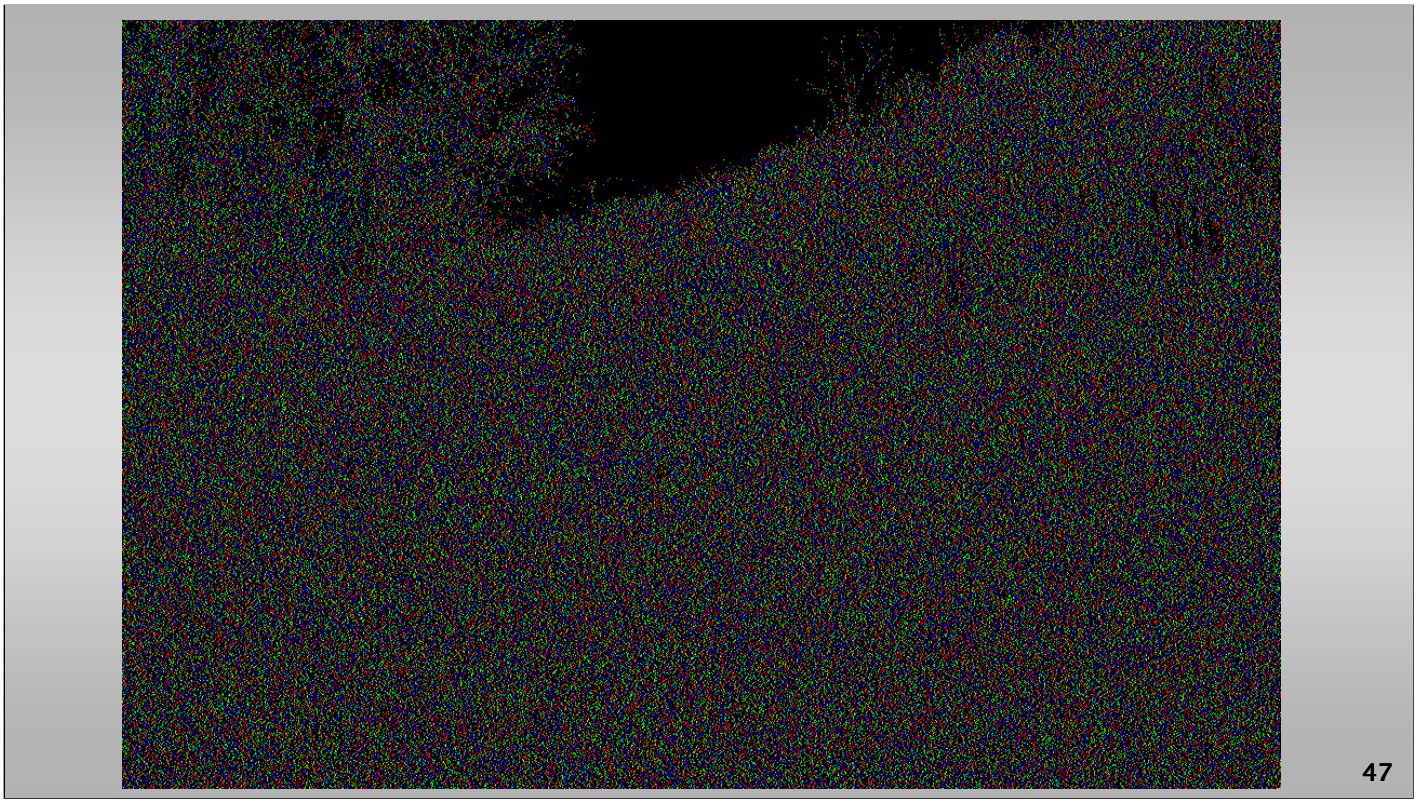


Figure. OpenPuff LSB image changes.

Further, as shown below in Slide 44, the software takes care to avoid saturated color pixel areas. This is commendable. However, without exception, all other areas of the image are highly modified — possibly with random data. This apparently random data will be more apparent to sophisticated image forensic analysis techniques.

The "High"

- OpenStego – https://www.openstego.com/index.html
- OTP-Steg – http://www.199.175.52.196.com/OTP-Steg/

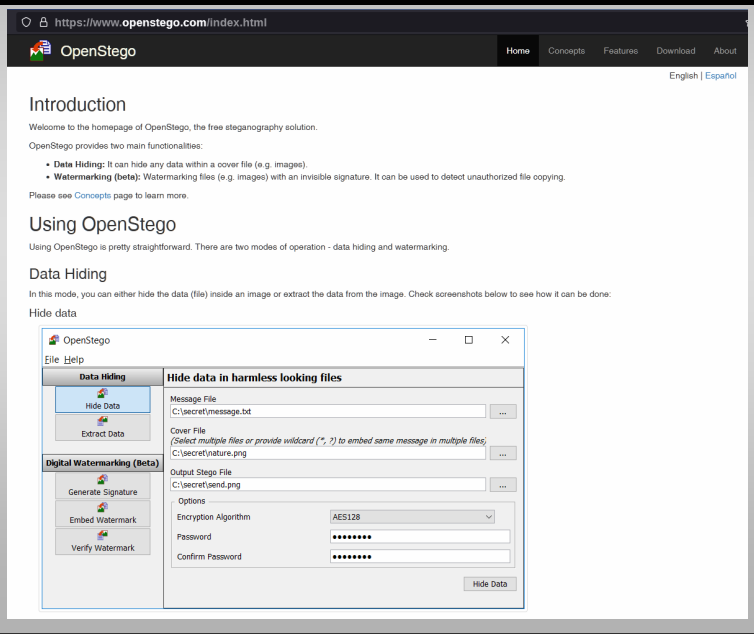
48

THE HIGH

The high steganographic software available in general attempts to do several things. It creates a low payload profile — in our examples below 1% of the pixel values only of the image are modified. This compares to the 12% to 83% of the pixels modified by the other software analyzed previously. Less modifications will correspond to and equal less detectability.

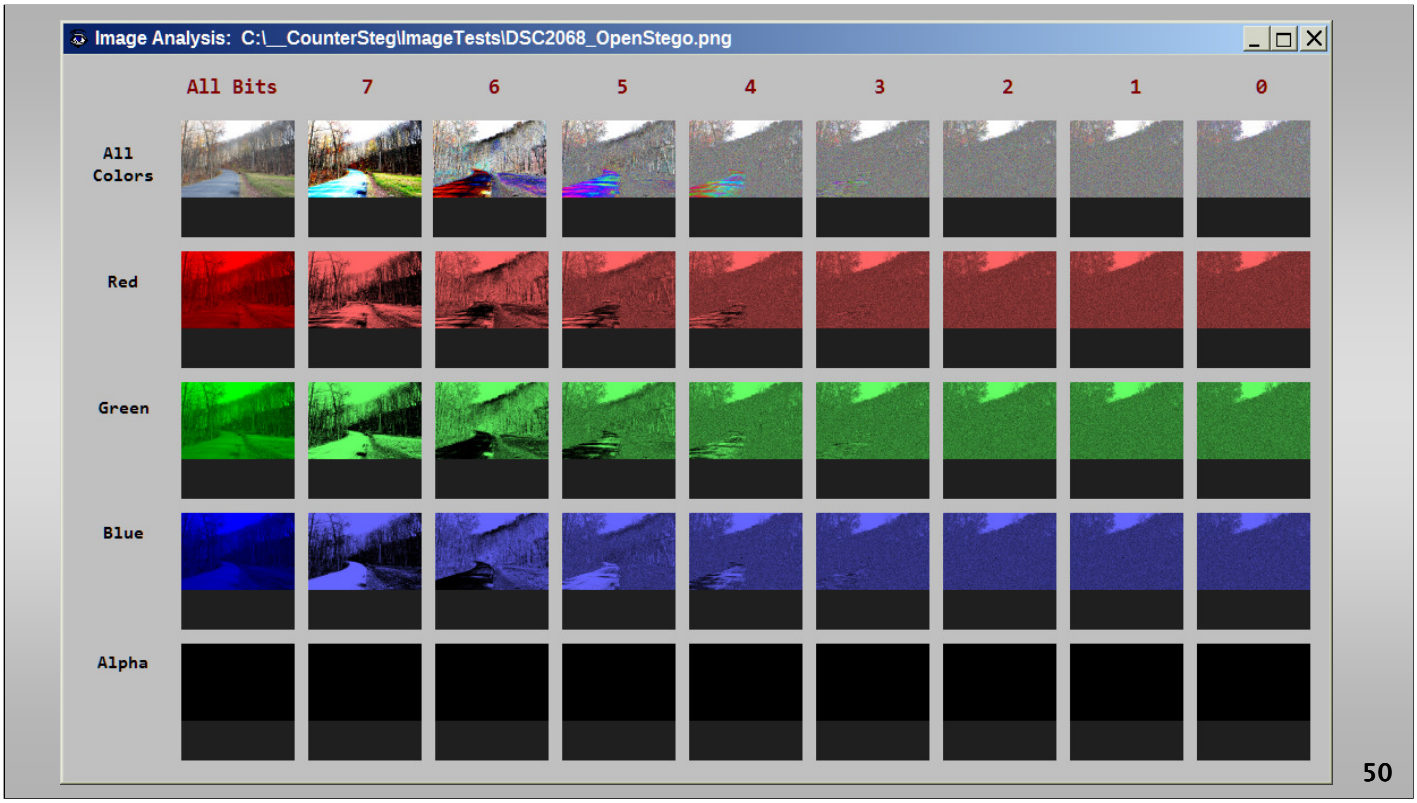
Also, it is important to disperse the changes into various areas of the image, specifically areas of higher noise and color variation. Areas of solid colors or pixel saturation should be actively and strongly avoided to lower the possibility and probability of forensic detectability.

OpenStego



OpenStego

https://www.openstego.com/index.html



OpenStego takes care to compress the data before embedding, reducing the overall payload size to about 1% of the image pixels. Also, it disperses the data encoding seemingly randomly throughout the image.

The overall image analysis shown below in Slide 47 provides little for further examination if only the payload image exists. It is likely any known statistical analysis technique will fall flat and fail when trying to determine if any data modifications have been performed on the existing image in hand.

Figure. OpenStego image analysis.

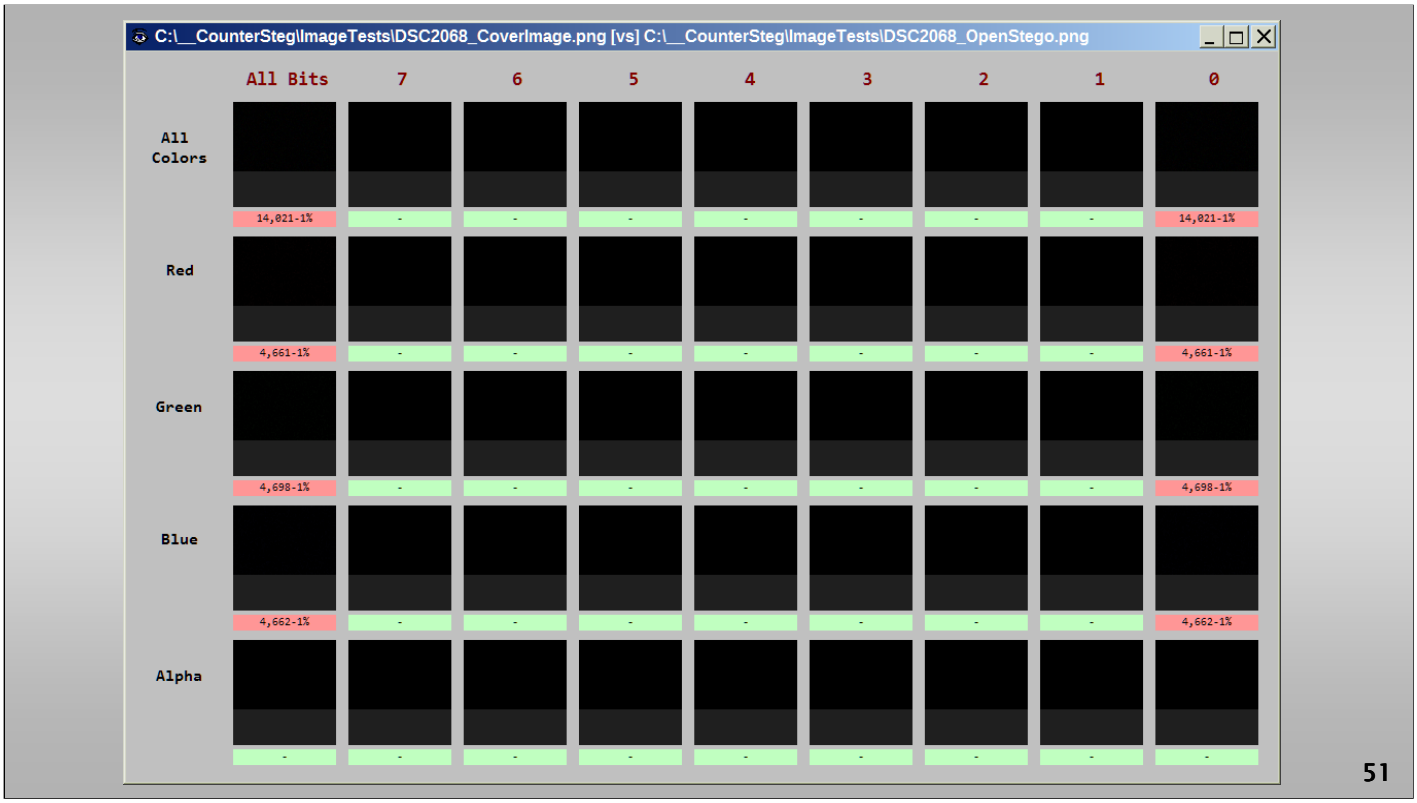


Figure. OpenStego image comparison.

The comparison analysis window shown in Slide 48 indicates only LSB values have been modified, in all three color channels. The color channels share equally with modifications, at about 1% each. OpenStego, therefore, is not taking into account relative individual color variations when deciding where to embed data values in various color channels. No modifications are made to the alpha (transparency) color channel.

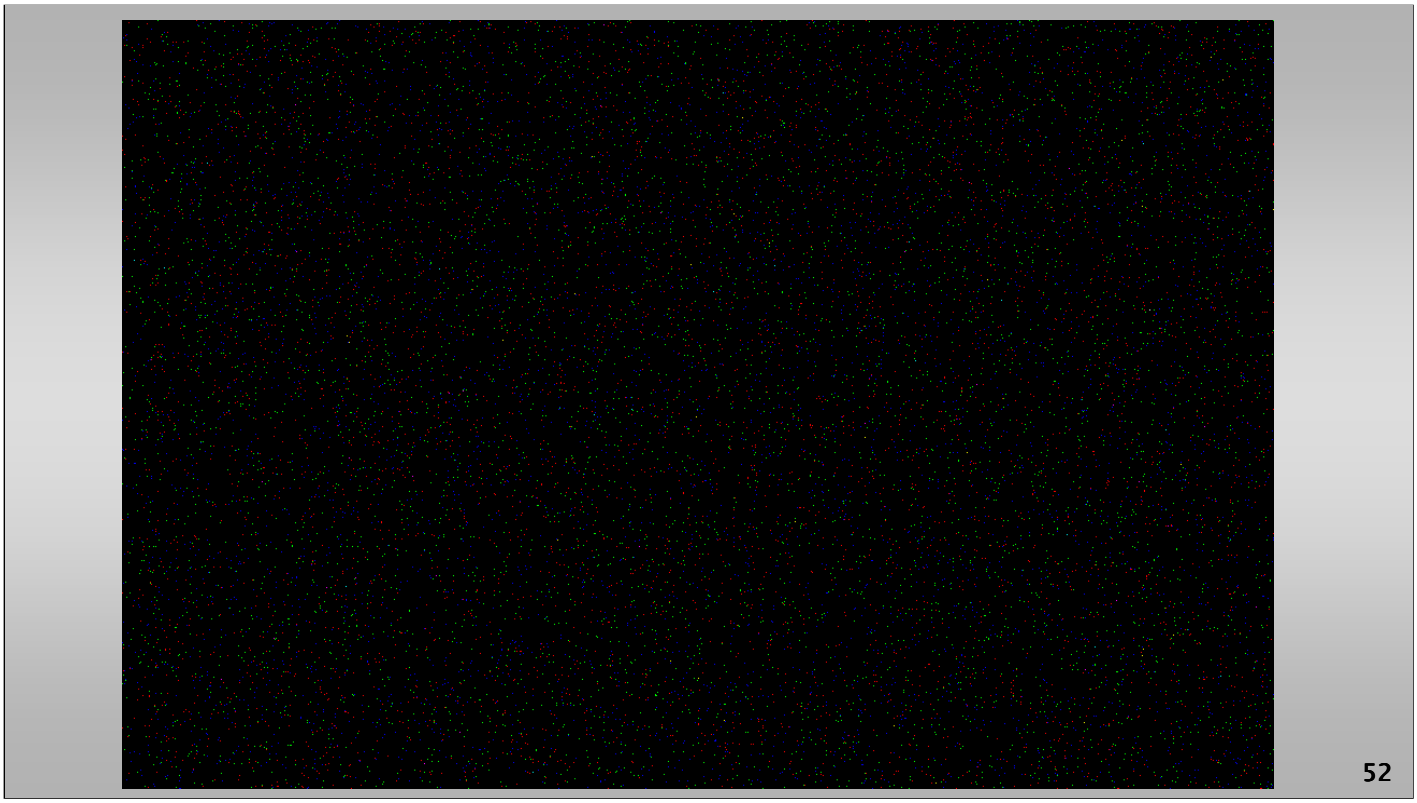
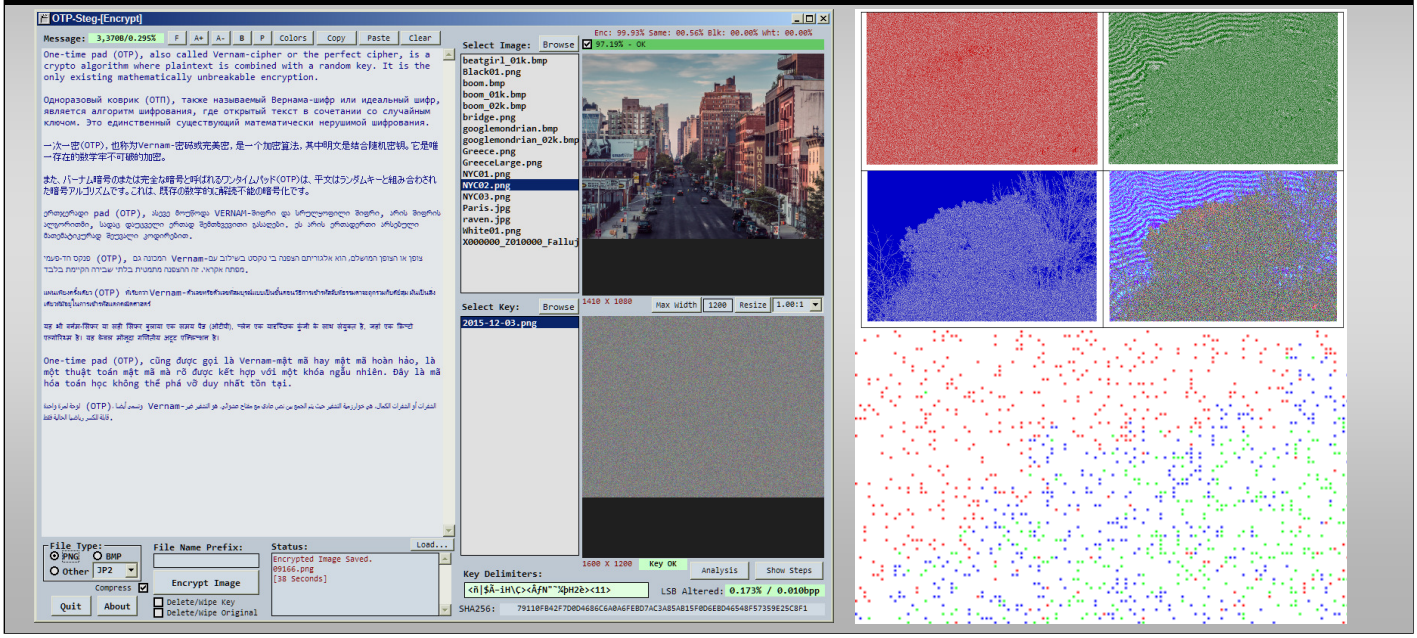


Figure. OpenStego LSB image changes.

Below in Slide 49 are shown the LSB modifications in each color channel. Apparently, this is a random distribution not taking into account color variations throughout the image as previously mentioned. Also, the software does not apparently take into consideration noise levels or saturation levels as well when encoding data.

OTP-Steg (One-Time-Pad Steg)

http://199.175.52.196/OTP-Steg/

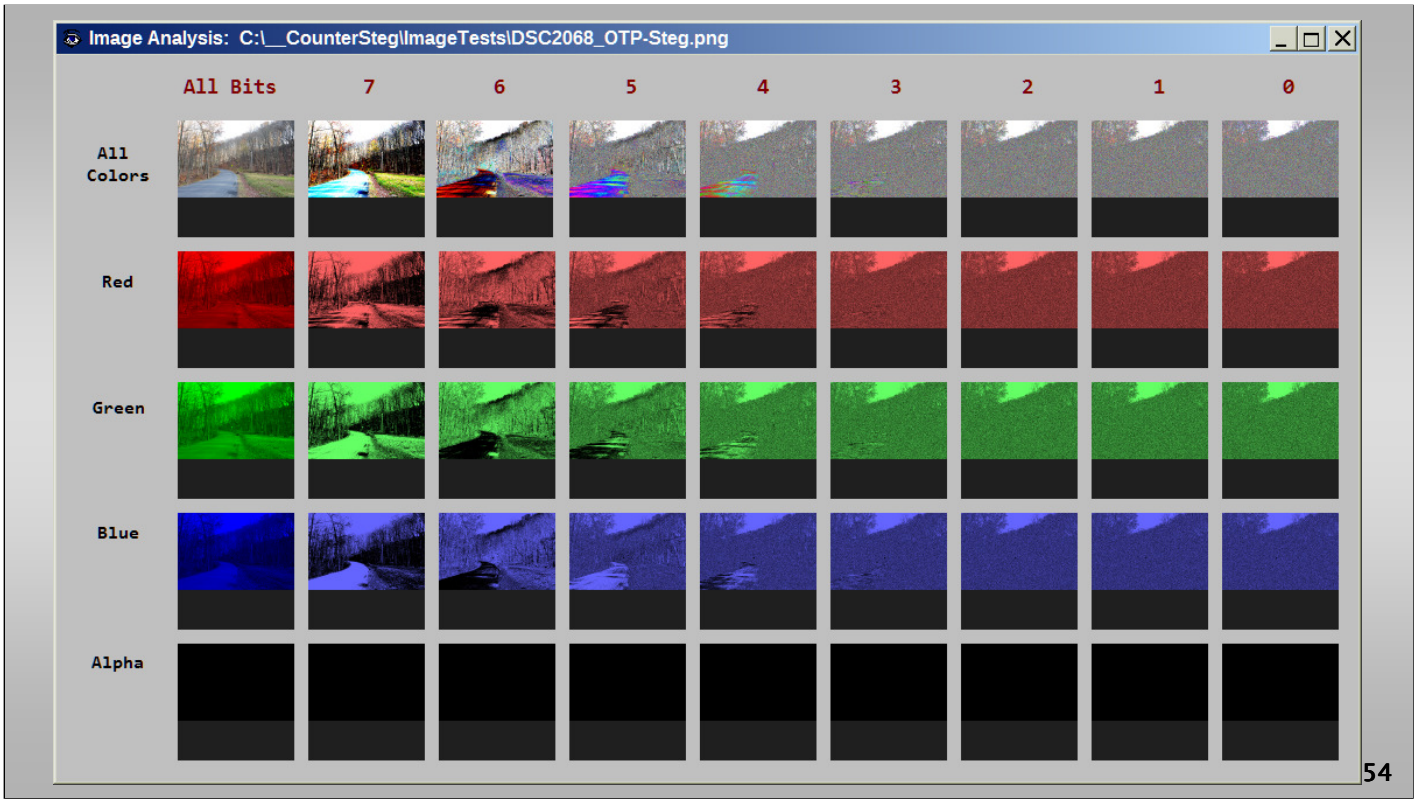


OTP-Steg

http://www.199.175.52.196/OTP-Steg/

OTP-Steg receives its name from one-time pad encryption, which is used for encrypting the data in this software before compression and encoding of the data. *OTP-Steg* uses the zlib library to compress all data heavily before encoding.

Further, the encoding process looks for, and avoids, saturated color or solid color areas of the image. It conducts a noise and variation color analysis of the entire image, to prioritize encoding of LSB data into less statistically detectable areas.



Overall payload image analysis shown below in Slide 51 shows no features different from a standard photograph in all bit planes. Also, the payload image will be visually indistinguishable from the cover image, as only LSB values are modified.

Figure. OTP-Steg image analysis.

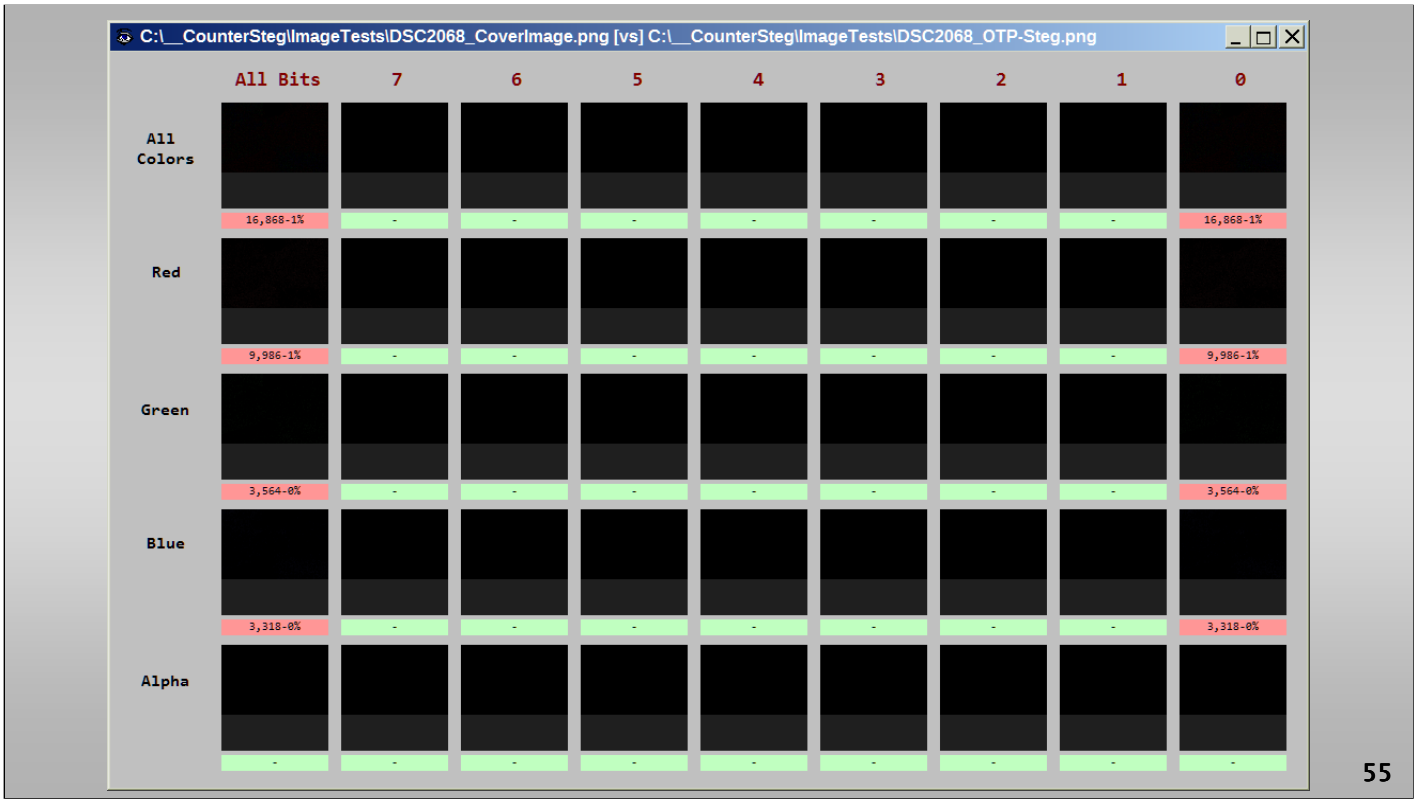


Figure. OTP-Steg image comparison.

The comparison to the cover image window shown below in Slide 52 indicates only 1% of the LSB values have been modified. Notice this also in fact varies significantly between color channel, with the red color channel carrying more than double the respective payloads of the green and blue channels.

This is because red color variation varies much more significantly through the image than the green and blue color variations. Thus, data carried in the red channel will be much harder to detect statistically.

Figure. OTP-Steg LSB image changes.

Shown below in Slide 53 is the image of the specific LSB values changed, in the three visible colors. Changes are dispersed randomly throughout the image, with respect to noise variation. The saturated sky area is completely avoided. Noticeably, red is much more heavily encoded into them blue and green.

This image has a much stronger red component than the other two colors. However, where blue or green are dominant, that is the color encoded into them that respective area. Only one color channel bit per pixel is allowed to be modified.

Conclusion

- Positively detecting the use of steganography in digital image files currently generally results in an unreliable and inconclusive effort.
- Instead of performing statistical analysis to overall produce subjective results, if steganography is suspected, an investigator would be more effective in simply looking for the original image file for comparison.
- If malware is suspected, compare with similar images and examine for steganographic artifacts.

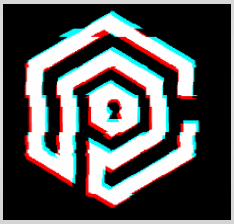
57

CONCLUSION

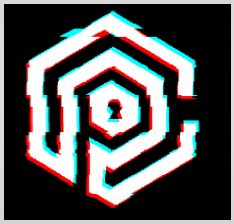
Positively detecting the use of steganography in digital image files is currently generally results in an unreliable and inconclusive effort. At least by attempting to actively recover the original image, before pixel bit alteration, it is estimated to make this procedure much more reliable for positive identification by providing investigative software to allow such a comparison efficiently.

Instead of performing statistical analysis to overall produce dubious results, if steganogrphy is suspected, we posit the investigator would be more effective in simply looking for the original image file for comparison. With comparison software in place, the investigator and assign virtually conclusive attribution. This can be immediately obtained as shown in the numerous examples previously presented in this paper, and useful as evidence in legal proceedings or requests for initial or additional warrants.

Thank You



Michael Pelosi
mpelosi@tamut.edu
903-334-6744
Texas A&M University, Texarkana, Texas



Thank you.