applied sciences

MDPI

Article

An Efficient Steganographic Protocol for WebP Files

Katarzyna Koptyra and Marek R. Ogiela *

Cryptography and Cognitive Informatics Laboratory, AGH University of Krakow, 30-059 Krakow, Poland; kkoptyra@agh.edu.pl
* Correspondence: mogiela@agh.edu.pl

Featured Application: The methods of digital steganography in image files presented in this research may find application in numerous areas and industries, because they offer great capacity and may be combined with cryptography. ① Hidden communication (in public or monitored channels, when encryption is not possible or difficult to use, contacts between whistleblowers and journalists or informants and police, to send dangerous contents, privacy protection, data security); ② military and intelligence (conceal strategic information during war and occupation operations); ③ business (protection against industrial/corporate espionage, hidden communication between business partners or company branches); ④ property protection (watermarking of images to embed information about the author, additional comments or some metadata, fraud detection); ⑤ other areas (bypassing censorship in countries violating human rights, sending viruses, tracking users). As can be seen, potential applications lie in both white hat and black hat fields of operation.

Abstract: In this paper, several ideas of data hiding in WebP images are presented. WebP is a long-known, but not very poplar file format that provides lossy or lossless compression of data, in the form of a still image or an animation. A great number of WebP features are optional, so the structure of the image offers great opportunities for data hiding. The article describes distinct approaches to steganography divided into two categories: format-based and data-based. Among format-based methods, we name simple injection, multi-secret steganography that uses thumbnails, hiding a message in metadata or in a specific data chunk. Data-based methods achieve secret concealment with the use of a transparent, WebP-specific algorithm that embeds bits by choosing proper prediction modes and alteration of the color indexing transform. The capacity of presented techniques varies. It may be unlimited for injection, up to a few hundred megabytes for other format-based algorithms, or be content-dependent in data-based techniques. These methods fit into the container modification branch of steganography. We also present a container selection technique which benefits from available WebP compression parameters. Images generated with the described methods were tested with three applications, including the Firefox web browser, GNU Image Manipulation Program, and ImageMagick. Some of the presented techniques can be combined in order to conceal more than one message in a single carrier.

Keywords: steganography; WebP; image; data hiding

check for updates

Citation: Koptyra, K.; Ogiela, M.R. An Efficient Steganographic Protocol for WebP Files. *Appl. Sci.* **2023**, *13*, 12404. https://doi.org/10.3390/app132212404

Academic Editors: Wenfeng Zheng, Mingzhe Liu, Kenan Li and Xuan Liu

Received: 12 October 2023
Revised: 6 November 2023
Accepted: 9 November 2023
Published: 16 November 2023

1. Introduction

Information security plays a crucial role in contemporary times, as data became a vital part of businesses and the everyday life of people in modern society. Security breaches and data leaks pose a threat to individuals, companies, and even whole countries. Among numerous attacks, we may mention malware, vulnerabilities, distributed denial of service, scams, credential stuffing, brute-force, frauds, spam, backdoors, disinformation, etc. In the last year, the most threatened targets were industries, healthcare, public services, scientific institutes, individuals, education establishments, finance or insurance organizations, and many more [1]. Data protection is therefore an important challenge everyone faces.

Sometimes, standard security measures, like encryption of the secret message, are not sufficient. In such strategic applications, third parties should not even be aware that communication is taking place. One of the possible solutions of this problem is steganography, a branch of security focused on information hiding. In steganography, the secret message is embedded in an insignificant container which is used as a carrier for the payload. The container is provided to the receiver, who extracts the message out of it. The main goal of this scheme is to conceal from external observers that secret data are hidden inside the cover medium.

Over the years, steganographic methods evolved from physical messages on paper to modern digital algorithms. Due to increasing computing power, new complex solutions were established (for example, the combination of steganography and cryptography [2]) and also a lot of potential carriers were used, such as texts [3,4], images [5–8], videos [9,10], binary executables [11], network packets [12–15], documents [16–18], or even DNA [19]. On the other hand, a number of counter-techniques were developed, called steganalysis. Their objective is to detect the possible presence of secret data in a container. The perpetual struggle between two sides results in the development of more and more creative algorithms.

Of all carriers, the most popular are digital images, for two reasons. First, they are omnipresent (not suspicious), and second, they offer great capacity (a large amount of data may be hidden inside). Image steganography algorithms from the literature may be divided into a few categories, depending on the adopted methodology.

Popular methods with low computational cost include spatial domain methods, which conceal secret data directly into pixel values. The best known is the least significant bit (LSB) technique, belonging to the container modification branch of steganography. In this method, bits of the secret message are placed as least significant bits of subsequent bytes of the container [20]. The invention of LSB replacement was a big achievement and researchers have created multiple variants of this algorithm. For instance, to increase capacity, it is possible to hide more than one bit per pixel [21–24] in various planes. The other modifications use a key to scatter secret data evenly throughout various regions of the carrier [25,26]. There is also the idea of choosing the embedding area without a key, but by considering only the most frequent pixel values, ignoring their least significant bit [27]. Then the capacity and payload placement are dependent on the contents of the image. Overall, the LSB technique has good sensory undetectability even when more than one bit per pixel is used [28,29]. However, with specialized software, the adversary is able to detect statistical anomalies introduced by the embedding process. They are manifested by a similar number of even and odd values in the histogram [30]. Some papers try to combat it and invent mechanisms of preserving the histogram during data hiding [5]. Another approach to spatial steganography is based on the fact that edges have better hiding potential than smooth areas. In this technique, called pixel value differentiation (PVD), embedding regions are selected by analyzing differences between consecutive pixels [31]. PVD is characterized by good stego image quality and has its own modifications [32–34].

Spatial methods of image steganography, while useful, also have disadvantages. They are limited to lossless formats and vulnerable to filtration or compression attacks. For this reason, another category of algorithm has emerged, namely transform domain. Such techniques convert picture data into the frequency domain and then conduct the hiding process. There are various possible transforms, including discrete cosine transform (DCT), discrete (DFT) or fast Fourier transform (FFT), and discrete wavelet transform (DWT). Usually they find application in JPEG files to compress data. DCT divides the image into frequency bands and hides the secret message in quantized coefficients [35]. We may find different variants of this idea, using coefficient randomization [36], a specific frequency band [37], or matrix encoding [6]. DFT is rarely used compared to other transforms, but the literature mentions existing methods [20]. DWT is much more popular, probably because of better compression and less artifacts. This kind of transform decomposes the image into a low-frequency signal and high-frequency details in horizontal, vertical, and diagonal directions. There are methods for both grayscale [38] and color images [39,40].

The frequency approach is useful in steganography, as the secret data are scattered among pixels and therefore difficult to wipe without degrading the quality of the container. We may also mention other domains useful for data hiding, like eigenvectors [41]. The main goal of these algorithms is to achieve high robustness, as it is difficult to remove the payload without the destruction of important areas of the photograph. There are also attacks aimed at frequency-domain methods, based on coefficient analysis [42] or classification with neural networks [43].

Thanks to development of neural networks and deep learning, a new category has found its place in image steganography. Deep neural networks have been used to select the embedding area for the LSB method and to scatter data between all of the available bits, while another network served the purpose of secret extracting [44]. Other types of architectures also found application in steganography. The authors of [45] proposed two networks (U-Net-based encoder and decoder) for hiding and recovering secret data, both working on color photographs. An interesting approach uses the style modification technique [46]. This algorithm takes three input pictures: cover, style, and secret. The resulting image resembles the carrier, but contains concealed data and is also drawn in the requested style.

Other examples of steganography include combinations with cryptography [47,48], watermarking [27,49] or secret sharing [50–52], multi-secret steganography used for concealing more than one message in a single container [53], techniques that do not require a predefined medium but generate the carrier from scratch [54,55], format-specific methods destined for particular files [36,56,57], the creation of subliminal channels in existing schemes to achieve better undetectability [58,59], systems that use biometric authentication for data hiding [60,61], combining various techniques to obtain secure data transmission in telemedicine applications [62,63], and many more.

Among container modification techniques, spatial-domain algorithms focus on uncompressed or lossless compressed images, like BMP or PNG. On the other hand, the majority of transform-domain methods operate on JPEG images. There are also some proposals of data hiding in GIF images. The WebP format had not aroused much interest in the information hiding community. Although it was introduced years ago, its popularity was not very high compared to other, well-established image files. As can be seen in related works, there is a shortage of papers concerning WebP steganography. This is the reason for this study, to present techniques appropriate for these images. In recent years, the presence of WebP on the internet became visible, so the time has come for this versatile format to take its place among other carriers suitable for steganography.

2. Materials and Methods

This sections presents proposed approaches to data hiding in WebP images. Besides the background, it is divided into two parts. The first one concerns methods which are based on format specification, but leave the pixels untouched. The second one is about data-based methods, which change image contents in an unnoticeable way, but have a negligible impact on file size.

2.1. Background Information about WebP Format

WebP is an image format developed by Google in order to save web resources. It supports both lossy and lossless compression of pictures. WebP files are designed to store still images or animations. Optionally, they may contain alpha channel, color profile, and metadata. The maximum dimensions of such images are 16,383 $\times$ 16,383 px. One of the biggest strengths of WebP is its predictive coding drawn from the VP8 video format. This algorithm uses nearby values to predict pixel value in the block. Only the difference is saved; in this way, a high level of compression can be achieved. According to claims by Google, WebP images can be 26% smaller than PNG and 25–34% smaller than JPEG.

The structure of WebP files consists of RIFF (Resource Interchange File Format) chunks. They are built of three sections: four bytes for the identifier, four bytes for the size, and

the remaining bytes for data. The size concerns only the data part, which means that the whole chunk length is the data length + 8 bytes. The first chunk of a WebP file is a RIFF chunk that contains the 'WEBP' identifier and one of the possible chunks: 'VP8' for simple lossy format (the last character is 0x20—space), 'VP8L' for lossless compression without advanced functions, or 'VP8X' for extended format. The first 16 bytes of a lossless WebP file may look similar to Figure 1. As can be seen, the chunk size is 0x3dc6 (in little endian), so the file size is 15,814 + 8 = 15,822 bytes.

```
52 49 46 46 c6 3d 00 00  57 45 42 50 56 50 38 4c  |RIFF.=..WEBPVP8L|
```

Figure 1. The first bytes of lossless WebP image; on the right is ASCII representation.

'VP8' and 'VP8L' chunks contain VP8 bitstream data [64] and WebP Lossless Bitstream [65], respectively. More complex are 'VP8X' chunks, which consist of the following:

- information about features used in the file;
- optional ICCP chunk with color profile;
- optional ANIM chunk with animation data;
- image data;
- optional EXIF chunk with Exif metadata;
- optional XMP chunk with XMP metadata;
- optional list of unknown chunks.

These parts are present in the order indicated above. Optional unknown chunks are intended to be used in the future as the standard evolves and should be ignored by readers to achieve backwards compatibility.

There are also some disadvantages of the WebP format. Although it is more than 10 years old, it still suffers from compatibility problems. In fact, these images are rarely used outside the web and the majority of users are clueless as to how to open downloaded pictures. The study conducted by the mozjpeg project regarding lossy encoding techniques revealed that WebP turned out to be better in some scoring algorithms, but comparative or worse according to others [66]. A serious threat was revealed in September 2023, when a buffer overflow was discovered [67] in both libwebp (CVE-2023-4863) and imageio (CVE-2023-41064). This vulnerability was acknowledged as critical and received the maximum severity score.

2.2. Framework

The framework of proposed methods consists of two complementary algorithms: embedding and extracting. The embedding algorithm takes the WebP carrier, secret message, and optional key as input. The message is hidden in the image, and the resulting file is transmitted to the receiver. Then, the container with hidden data (optionally with the key) is given to the extracting algorithm that retrieves the secret message out of it. The schema of the whole process is presented in Figure 2.

As can be seen in the schema, the WebP carrier consists of a chunk containing metadata and other file-related information, and an image chunk containing compressed image data. Presented in this paper are format-based methods relying on the chunk part, whereas data-based techniques are focused on modifications of image data.

2.3. Format-Based Methods

2.3.1. Simple Injection

This is a very simple technique of data hiding not limited to the WebP format. It works by appending another file at the end of the carrier. It is based on optimization made in applications to ignore any excess data. The WebP standard says that "The file SHOULD NOT contain any data after the data specified by File Size. Readers MAY parse such files, ignoring the trailing data". This means that the majority of software opening the stego image should result in displaying the carrier, but ignoring the secret image. The quality of

the cover image is not affected; however, the file size increases and is equal to container size + secret size. This method is not secure, the as hidden file may be detected with specialized software or inspection.

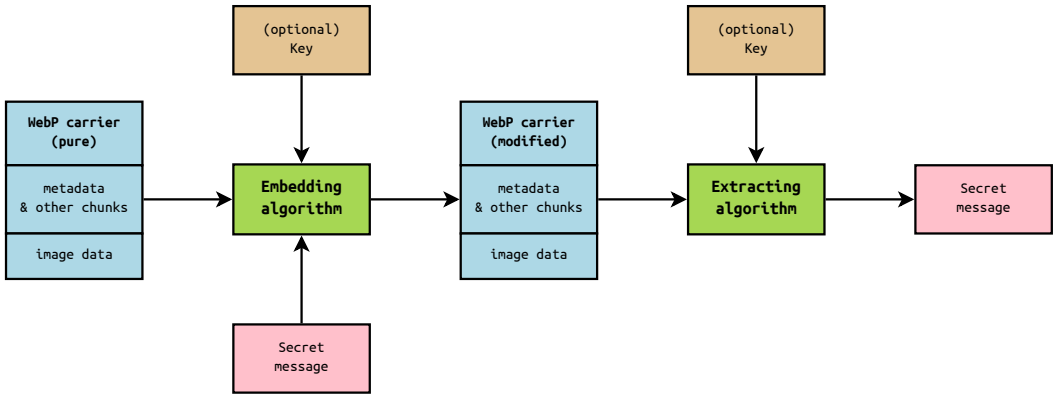

Figure 2. Framework of presented methods.

2.3.2. Thumbnail

Thanks to its ability to store metadata, WebP images may also contain a thumbnail (reduced-size version of the picture). Thumbnails are defined in the Exif Specification with the use of two connected tags: JPEGInterchangeFormat (the offset in the image directory to the JPEG) and JPEGInterchangeLength (the length of the thumbnail in bytes). As can be seen, thumbnails are required to be in the JPEG format. The Image Metadata and Exiv2 Architecture book [68] says that the thumbnail cannot have embedded metadata. However, in our tests, we succeeded in embedding a JPEG file with metadata.

The proposed solution starts by creating a thumbnail in the JPEG format. Then, a secret message should be hidden in this thumbnail with the selected embedding algorithm. For steganographic purposes, JPEG files are great carriers. They offer a lot of possible methods and creditable capacity. Finally, the carrier with the hidden message should be inserted in a WebP image. In this way, recovering requires two steps: extracting the thumbnail and decoding the secret. It is then an example of multi-level steganography [51].

2.3.3. Metadata

Hiding data in WebP metadata may be realized with Exif [69] or XMP [70]. They are both defined for storing metadata created by cameras, smartphones, scanners, and other devices designed for recording images. Exif has tags for Maker, Model, Aperture, and many other settings. XMP is an extendable format that offers XML-like or array-like structures, called Bag, Seq, or Struct.

According to the specification, metadata may be present in the extended version of WebP images (those with the VP8X chunk). If so, there should be at most one chunk of each type. If there are more such chunks, readers may ignore all except the first one. This opens the possibility for two approaches to data hiding. We may add more than one chunk with the embedded secret in the WebP file, as it will probably be ignored by software. Alternatively, we may conceal a secret in the existing metadata chunk (or insert a new one with our message). In both cases, it is advisable to encrypt concealed information, as metadata may be recovered and seen by anyone, even accidentally while checking camera parameters, etc.

In Exif metadata, there are multiple tags suitable for data hiding. The list of standard Exif tags is available at https://exiv2.org/tags.html (accessed on 10 August 2023). Depending on the secret format, we may need Ascii type for Latin text, Long for numbers, or Undefined for binary data. Moreover, the standard defines a MakerNote tag, destined for camera manufacturers, in order to place any custom metadata in the file. Usually they describe camera settings, for example, serial number, focusing more, etc. The Exif standard

does not define the structure of the MakerNote. Some manufacturers encrypt portions of the information, which means that such areas may be used for steganographic purposes and even combined with cryptography.

XMP, on the other hand, is an extensible format. Besides overwriting existing metadata, we may create our own structures and place secret data inside. The available types are Bag, Seq, or Struct, described as follows [68]:

> An "XmpBag" is a set of key/value pairs which are represented by XML attributes. An "XmpSeq" is an array of metadata similar to a JavaScript or Python array. It is represented by an XML list and can be accessed by index. An XmpStruct is a set of keys to trees of metadata rather like a JavaScript or Python object.

2.3.4. Chunk Injection

The chunk injection method exploits the expandability of the WebP format. To warrant openness to future releases, the standard allows WebP files to contain unknown chunks. These chunks should be ignored to achieve backwards compatibility. In this way, one may append new chunks with hidden data to the file.

Embedding a secret with this technique is divided into a few steps. Firstly, we need to create a chunk with secret data. The name of this chunk should be different than specified in the standard, and even better, with low probability of future use. Therefore, a new chunk consists of an identifier (four bytes), a size (four bytes encoded in little endian), and contents (of length equal to the aforementioned size). All these parts are concatenated without any separator. Secondly, the prepared chunk is embedded as the last one in the WebP file. Thirdly, we must modify the size of the RIFF chunk, as adding new data increases the length of the carrier file. The new size is computed as the old size + secret size + 8.

To recover the hidden secret, the receiver needs to find the offset and length of the embedded data. Both pieces of information may be acquired with exiv2 software (version 0.27.3). It is recommended to encrypt the secret, as chunks are stored in plain text, so additional protection is a good idea.

2.4. Data-Based Methods

A lossless version of a WebP image may serve as a container for existing methods, like least significant bit embedding. However, there are other possible approaches to steganography in this format.

2.4.1. Transparency

The WebP format supports transparency, in both lossy and lossless modes. The interesting option used in this method is "exact", described as follows.

-exact preserves RGB values in the transparent area; default = off

Normally, the WebP encoder skips bytes in transparent areas of the carrier to save space. However, with this option enabled, it is possible to use these locations to conceal some data. The secret may be embedded in each transparent pixel. Then we are able to achieve the capacity of 3 bytes per pixel. In such a case, the "lossless" option is also required to avoid the destruction of hidden data. Another approach is to conceal graphic data by pasting the secret image into the transparent area, preserving transparency.

The secret is not visible on the preview, but the method does not provide much security (the more data are hidden, the bigger the file becomes). On the other hand, it has satisfactory capacity and does not visually change the container. Additionally, the presented technique is also applicable to PNG images. In PNG images, the container modification method of embedding in transparent pixels works similarly as in WebP files. The embedding area consists of all transparent pixels. Secret data are translated to bytes, and for each color component with alpha channel equal to zero, one byte is concealed. This means that a single pixel can hold three bytes of secret data. The secret message length should be less than or equal to the number of transparent pixels to be able to conceal it. If the capacity is greater than the secret message length, the data may be spread over available space. The recovering

process is carried out in reverse. First, secret bytes are read from transparent areas, and then they are shaped into the message. When the stego image is open in graphical software, the hidden pixels are not visible. But this technique causes the file size to grow, which gives a clue that something may be hidden inside. This method can also be used to conceal a digital image instead of text or binary data. Then, to reveal the hidden image, the receiver needs to remove the alpha channel and to save the resulting image.

2.4.2. Prediction Modes

This method uses block prediction modes in the WebP format (derived from VP8) to store secret data. During encoding, the frame is divided into smaller segments called macroblocks. Then, based on previously processed blocks, the encoder is able to predict unknown pixel values in the macroblock. Redundant data may be subtracted from the block, which gives smaller residues that are efficiently compressed. It is called predictive coding.

Different prediction modes used in WebP compression are presented in Figure 3. Each of them uses fairly simple arithmetic to extrapolate from already-calculated values. For example, horizontal prediction fills unknown parts of the block with a copy of the left-hand-side column. The mode with the best match is used for encoding the current block.

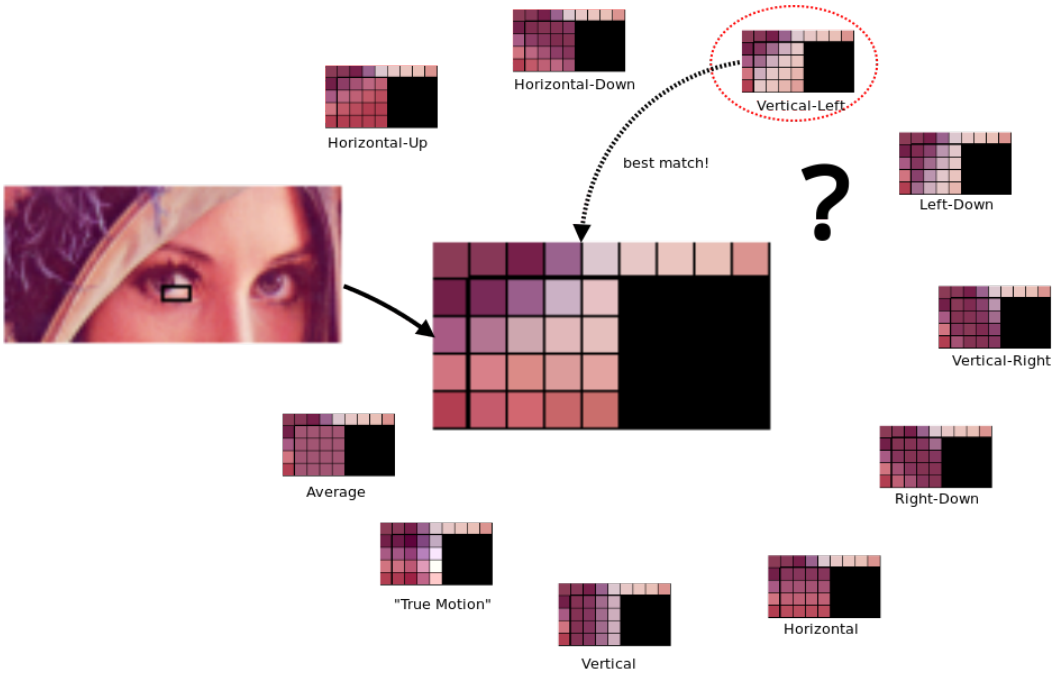

Figure 3. Prediction modes used in WebP lossy compression (https://developers.google.com/speed/webp/docs/compression?hl=en (accessed on 2 August 2023); ①).

In reference implementation, each of the prediction modes is represented as a number. To use them in steganography, we divided available modes into two groups, depending on their parity. In other words, even modes are in a separate group from odd modes. To hide a message, the encoder needs to select modes according to message bits. Normally, the chosen prediction mode is the one with the best match. This time we impose a limit to choose the best match only from the corresponding group. In this way, the stego file may be slightly bigger than the optimal image produced by the encoder, but the difference would be very small and inconspicuous. Also, there may occur little discrepancies between the optimal image and the stego image introduced by compression, which should not be visible. The capacity is one bit per block.

2.4.3. Color Indexing Transform

The color indexing transform technique is based on a WebP property which says that "If there are not many unique pixel values, it may be more efficient to create a color index array and replace the pixel values by the array's indices". It works as follows:

> The color indexing transform checks for the number of unique ARGB values in the image. If that number is below a threshold (256), it creates an array of those ARGB values, which is then used to replace the pixel values with the corresponding index; the green channel of the pixels is replaced with the index, all alpha values are set to 255, and all red and blue values are set to 0.

This feature may be used in steganography. To hide a message, we need a container with up to 128 colors. Additionally, the difference between every two colors should be greater than 1. The difference may be computed based on the following formula [71]:

$$d = \sqrt{(R_1 - R_2)^2 + (G_1 - G_2)^2 + (B_1 - B_2)^2},$$

where R_x, G_x, B_x are the red, green, and blue components of a color.

New colors are created by modifying a randomly selected component by 1. For each pixel, when a message bit is equal to 0, we use the original version of the color. When the bit is equal to 1, we choose the modified version. In this way, the carrier with the embedded message contains up to 256 colors, meeting the limit imposed by the format. Both versions of colors are slightly different, but very similar and hard to distinguish with the naked eye. The maximum capacity of this method is equal to the total number of pixels. A similar strategy may also be applied to GIF images.

2.4.4. Container Selection

This technique requires the preparation of an image set of size N in advance. They will be used indirectly as carriers. Each image is compressed 100 times with various levels of parameters. The cwebp compressor offers numerous options to choose from. The selected one is the -q option, which denotes compression level and may be set between 0 and 100. In this way, a single image encodes a number which may be translated to a character by adding 32 to it and casting it to the char type. It is then possible to encode uppercase and lowercase letters, numbers, spaces, punctuation, brackets, and some special characters.

To transmit a secret message, the sender chooses images that correspond to subsequent characters of the secret message and sends them to the receiver. To decode the secret, the receiver compares obtained images with his set. Then, he decodes the message by adding 32 and translating to ASCII characters. The images themselves contain no data, and the secret is recovered with the use of a predefined set. The modifications of this method are able to use a shared key to change the order of images or halve the image set and transmit one bit per carrier.

The capacity of this method depends on the shared image set size. The maximum length of the transmitted message is N characters. The bigger the set, the longer the message that may be transmitted. However, it is advisable to use this technique sporadically for short messages, to avoid sending the same images repeatedly. The disadvantage of this method is the disc space needed for the database and the used bandwidth, as we need to send much more data than the actual length of the secret message. Although a similar approach may be used with other image formats, here we exploit the quality parameter, which gives a broad data range and is not present in every format.

3. Results

This section provides a description of the conducted experiments of data hiding and their results. It contains the used commands and interesting dumps of involved files, with some additional conclusions.

3.1. Format-Based Methods

3.1.1. Simple Injection

The purpose of this experiment was to test the simple injection method. The tests were able to create the stego file (the carrier with embedded message), checking it in a few applications and extracting the secret from the container. Creating a stego file with an attached secret image is super easy; we just need to glue two input images together and save it as a new file.

```
cat img1.webp img2.webp > img3.webp
```

The newly created image is correctly recognized as RIFF (little endian) data, a Web/P image. Its size is the sum of the original files.

The interesting part of this research was checking whether the new image would open in various programs and if it would be displayed or not. The testing suite included GIMP, Firefox, and ImageMagick. This limitation is caused by the fact that the WebP format is not implemented in every piece of software. Regardless of the hidden data, some applications just do not (yet?) support the format. During tests, ImageMagick and Firefox handled the modified file easily and displayed the cover (insignificant image denoted as img1.webp in this experiment), which was expected behavior. On the other hand, GIMP returned an error because its plugin could not read the file in a proper way.

As we knew the offset and the length of the hidden file, we might easily recover the secret with dd:

```
dd if=img3.webp count=$((179632-163802)) bs=1 skip=$((163802+8))>img4.webp
```

There are multiple ways of finding the offset and the length: with the exiv2 program, from original files sizes, using any programming language, etc.

Finally, the restored image was compared to the original img2.webp and they turned out to be identical. The experiment proved that with this technique, we may obtain lossless recovery if the stego image is not modified.

Obviously, the secret image may be found if someone conducts an analysis of the stego file. This method is not safe, but does not require special software apart from the one available by default in the operating system. Additionally, it may be used without effort, especially on websites.

3.1.2. Thumbnail

The purpose of this experiment was to test the thumbnail method. During the tests, the thumbnail with the hidden secret message was created. Later, it was embedded in the carrier. Further tests were able to check the stego image in applications, recovering the thumbnail and comparing it with the original one. Hiding a secret in a thumbnail has been realized in a few steps. At the beginning, the original image was scaled down and converted to JPEG.

```
convert img.webp -scale 5% img-thumb-pure.jpg
```

Multiple steganography methods are possible for the JPEG format; in this case, we used the F5 algorithm [6]. The secret text "Steganography among other rare disciplines is honored to be described as both an art and Science field." was concealed and the new file saved as img-thumbnail.jpg. To embed it into the original image, the following command was executed (insert thumbnail with exiv2 program):

```
exiv2 -it in img.webp
```

The resulting image contains the thumbnail with the hidden message. In this experiment, the main carrier contained no metadata to avoid discrepancy between existing meta records and the added thumbnail.

As a result of embedding, the file size increased, but the image contents remained unchanged. The presence of the thumbnail may be detected with binwalk:

```
DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
4626978       0x469A22        TIFF image data, little-endian offset (...)
4627808       0x469D60        TIFF image data, little-endian offset (...)
4631404       0x46AB6C        JPEG image data, JFIF standard 1.01
```

However, it is not considered as a disadvantage, because thumbnails are normal phenomena often seen in photos, and therefore not suspicious.

Later, the stego file was recovered from the carrier and compared to the pure version to check whether embedding was lossless. The diff program reported no changes, so it was possible to correctly decode the secret message.

3.1.3. Metadata

The purpose of this experiment was to test the metadata method, including two metadata types: Exif and XMP. These tests were able to choose the appropriate tags or structures, embedding secret messages, analyzing created files in applications, and reading the secrets from carriers. Metadata modification may be achieved with applicable software; in the following experiments, the exiv2 program was used. The first technique uses Exif metadata, specifically Exif.Photo.ImageUniqueID of type Ascii. According to the tag description, it indicates an identifier assigned uniquely to each image. It is presented as a hexadecimal string of 128-bit length.

Before embedding, this tag looked as shown below.

```
Exif.Photo.ImageUniqueID    Ascii    33  090caaf2c085f3e102513b24750041aa
```

As a proof of concept, a memorable value `deadbeef1337c0defeedf00d2137face` has been hidden with the following command.

```
exiv2 -M'set Exif.Photo.ImageUniqueID
deadbeef1337c0defeedf00d2137face' img.webp
```

After embedding, the listed metadata contain the modified unique ID of the image.

```
Exif.Photo.ImageUniqueID    Ascii    33  deadbeef1337c0defeedf00d2137face
```

They are also visible in other applications. Figure 4 shows how we can read our hexstring in a different way. To do this, we should choose Image → Metadata → Show metadata in GIMP.

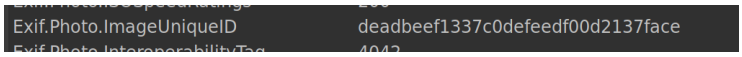

Figure 4. Embedded metadata visible im GIMP.

Both the file size and the image contents remained unchanged. In normal applications, this method may be used to embed a small encrypted file with a fixed size of 128 bytes. Bigger files may be divided between a few images, but the receiver must know the order of files to properly recover the secret.

To conceal a secret in XMP metadata, we used the LocationCreated property from the IPTC Extension schema, designed for the location in which the photo was taken. The secret message has been hidden in newly created TextBag with selected cities. First, provided places have been encoded in the following manner.

```
exiv2 -M'set Xmp.iptc.LocationCreated XmpText type=Bag' img.webp
exiv2 -M'set Xmp.iptc.LocationCreated[1]/iptcExt:City Helsinki' img.webp
exiv2 -M'set Xmp.iptc.LocationCreated[2]/iptcExt:City Edinburgh' img.webp
exiv2 -M'set Xmp.iptc.LocationCreated[3]/iptcExt:City Larissa' img.webp
exiv2 -M'set Xmp.iptc.LocationCreated[4]/iptcExt:City Las Vegas' img.webp
exiv2 -M'set Xmp.iptc.LocationCreated[5]/iptcExt:City Oslo' img.webp
```

After executing these commands, let us present the resulting XMP metadata.

```
<?xml version="1.0"?>
<?xpacket begin="" id="W5M0MpCehiHzreSzNTczkc9d"?>
<x:xmpmeta xmlns:x="adobe:ns:meta/" x:xmptk="XMP Core 4.4.0-Exiv2">
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<rdf:Description
xmlns:Iptc4xmpCore="http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/"
xmlns:iptcExt="http://iptc.org/std/Iptc4xmpExt/2008-02-29/"
rdf:about="">
<Iptc4xmpCore:LocationCreated>
<rdf:Bag>
<rdf:li iptcExt:City="Helsinki"/>
<rdf:li iptcExt:City="Edinburgh"/>
<rdf:li iptcExt:City="Larissa"/>
<rdf:li iptcExt:City="Las Vegas"/>
<rdf:li iptcExt:City="Oslo"/>
</rdf:Bag>
</Iptc4xmpCore:LocationCreated>
</rdf:Description>
</rdf:RDF>
</x:xmpmeta>
<?xpacket end="w"?>
```

As can be seen, the embedded cities are visible in the listed metadata (they are formatted for clarity; normally, there are no indentations or white spaces). The secret message may be recovered by reading the first letters: HELLO.

3.1.4. Chunk Injection

The purpose of this experiment was to test the chunk injection method. These tests included the generation of a chunk with the selected name containing secret data, embedding it into a carrier, checking the file in applications, and extracting the message out of the stego medium. During this experiment, two approaches were tested: with updating the carrier main chunk size and without it. Chunk injection tests started by choosing a secret message and a chunk identifier. We have decided to embed 38-byte shellcode into a chunk named XYZW. Such a name is not part of the standard and the probability of using it in the future is low because of its meaninglessness. The final shape of the chunk is presented as follows; as can be seen, the four bytes after the identifier denote the chunk size.

```
58 59 5a 57 26 00 00 00  48 c7 c0 3b 00 00 00 48  |XYZW&...H..;...H|
8d 3d 10 00 00 00 48 c7  c6 00 00 00 00 48 c7 c2  |.=....H......H..|
00 00 00 00 0f 05 2f 62  69 6e 2f 73 68 00        |...../bin/sh.|
```

Later, we attached it to a pure container as a last chunk. As expected, the file size increased by 46 bytes (header + payload). We also updated the RIFF chunk size to correspond to the current file length. After these operations, exiv2 correctly detects the embedded chunk.

```
STRUCTURE OF WEBP FILE: stego-file1.webp
Chunk |   Length |   Offset | Payload
RIFF |  4642280 |        0 | WEBP
VP8X |       10 |       12 | ....o....
VP8  |  4624168 |       30 | .....*p...>)..B..!$(.Jx...gk.qd.
EXIF |    15264 |  4624206 | II*.........................
XMP |     2755 |  4639478 | <?xpacket begin="..." id="W5M0Mp
XYZW |       38 |  4642242 | H..;...H.=....H......H......../b
```

Also, the file has been tested in Firefox, GIMP, and ImageMagick. All these programs displayed the image without problems.

The alternative approach is to inject the chunk without changing the RIFF chunk size. Then it is not detected with exif2 software.

```
STRUCTURE OF WEBP FILE: stego-file2.webp
Chunk |   Length |   Offset | Payload
RIFF |  4642234 |        0 | WEBP
VP8X |       10 |       12 | ....o....
VP8  |  4624168 |       30 | .....*p...>)..B..!$(.Jx...gk.qd.
EXIF |    15264 |  4624206 | II*............................
XMP |     2755 |  4639478 | <?xpacket begin="..." id="W5M0Mp
```

At first glance, it seems to be a good idea, as hidden data are more difficult to spot. However, in such a case, some applications have problems with opening the file. Even though it is displayed in the browser (Firefox) and by ImageMagick, GIMP refuses to open the image, as the WebP plugin returns an error.

To make this experiment more complete, we also recovered hidden shellcode from the container. It was realized with dd software, which is useful for copying parts of the file.

```
dd if=stego-file1.webp count=$((38)) bs=1 skip=$((4642242+8)) > recovered
```

Retrieving was possible as we knew payload length (38) and offset (4,642,242 + 8 bytes of header). We compared the recovered shellcode with the original and the files were identical. This means that the presented method is suitable for carrying even binary data which cannot be corrupted.

3.2. Data-Based Methods

3.2.1. Transparency

The purpose of this experiment was to test the transparency method. Tests included the embedding of two secrets; the first one was in the form of bytes, while the second one was an image. In the subsequent part, the secrets were extracted. Additional testing served for comparing the results of WebP and PNG embedding in transparent pixels. To test this method, we used a WebP image of size 588 × 97 with 18,622 transparent and 38,414 non-transparent pixels. This gives about 32.64% of the container available for data hiding. We conducted two experiments. In the first one, red, green, and blue components of each transparent pixel were replaced with secret data. In the second experiment, we concealed a secret image in one of the transparent regions.

The resulting images were of a lossless type of WebP, so their header was marked with 'VP8L' chunk. In the first experiment, we used all possible capacity, which means that we were able to embed 55,866 bytes of data. As a result of this, the image size increased from 1.8 kB to 61.3 kB. For comparison, we also conducted the same test for the PNG image. The sizes before and after embedding were the same, namely 67.7 kB.

In the second experiment, we prepared a secret image containing text in the transparent region of the container. In this case, only part of the available capacity was used, so the size of the stego file increased only to 2.2 kB. The stego image and recovered secret are presented in Figure 5.

Figure 5. Secret image hidden in transparent region. (**Left**): the carrier, (**right**): recovered secret.

3.2.2. Prediction Modes

The purpose of this experiment was to test the prediction mode method. The tests included embedding a secret into a carrier, checking parity of selected modes, and comparing

the stego file with the pure container. In this experiment, we modified the original implementation of the library to choose only modes not divisible by two. It may be perceived as hiding a message of maximum possible length consisting of 1 s. The regular encoder produced the image of size 1.9 MB, while a steganographically enhanced version created a WebP image of size 2.1 MB. This difference is tiny considering the amount of hidden data and the carrier size (6000 × 4000 px). We were also checking selected modes during embedding. The following dump fragment proves that all modes were from the odd group:

```
Selected mode is 1.
Selected mode is 9.
Selected mode is 5.
Selected mode is 7.
Selected mode is 5.
Selected mode is 3.
Selected mode is 1.
Selected mode is 5.
Selected mode is 5.
Selected mode is 1.
Selected mode is 3.
```

Finally, we compared our laden carrier with the pure carrier produced with the unmodified library. The differences were marked with the violet color, as visible in Figure 6. All these changes are very slight and not visible with the naked eye. On the close-up, we may observe characteristic shapes of modified fragments, which are bigger in plain areas and smaller in complex regions. Additionally, the alterations are focused near lines and noisy parts that require various optimal modes. With this method, we were able to conceal 1,157,252 bits, which is almost 145 kB in total.

Figure 6. (**Left**): differences between pure and laden carrier; (**right**): close-up details.

3.2.3. Color Indexing Transform

The purpose of this experiment was to test the color indexing transform method. Tests included concealing a secret message in a container and checking the properties of the created file and pure container. It included comparing their sizes and color palettes. In this experiment, we used an image with a reduced number of colors—24 and size 64 × 64 px. We hid a message consisting of alternating zeros and ones. During secret embedding, new colors were created by changing the blue component by 1. The difference between the original and resulting images is presented in Figure 7. With the naked eye it is impossible to spot the color changes. In the stego image, there are 48 colors in total. Its size increased from 6.7 kB to 6.8 kB.

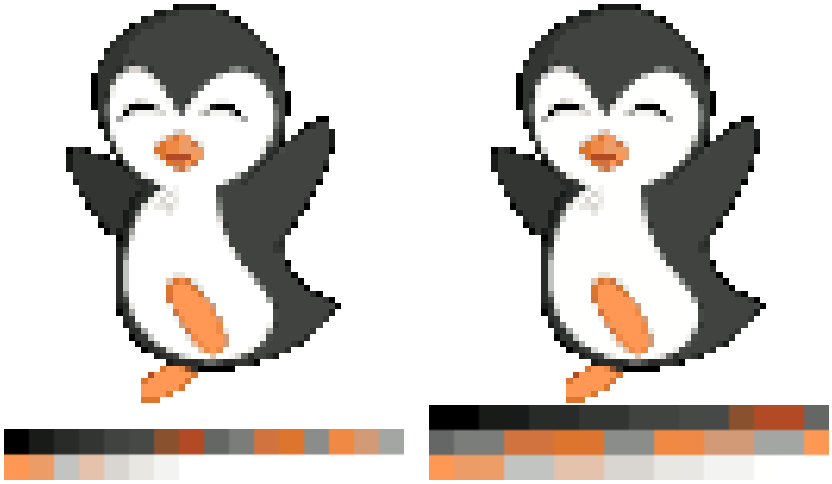

Figure 7. (**Left**): pure image with its colors; (**right**): cover image with extended colors.

4. Discussion

The methods presented in this paper are divided into two categories: the manipulation of file structure or the data. The main difference between these two approaches is that format-based algorithms do not change image contents, only the file structure. It may be seen during pixel-by-pixel comparison of images that data-based algorithms introduce some discrepancies, whereas format-based methods not. On the other hand, data manipulation does not significantly increase the file size. From all described techniques, all except one are focused on container modification, whereas the remaining one uses container selection.

The original image is modified in all techniques except container selection (in which different compression levels indicate secret messages). In format-based techniques, modifications affect the file, but not the image content itself. This means that some data may be added to the container or replaced, but pixels remains identical. The second category of methods, data-based methods, introduces alterations to pixels. In color indexing, one of the components may be modified by 1. In prediction modes, alterations concern blocks. The size of the block may vary, but usually it is a square group of pixels, the values of which are changed during the embedding process. These modifications are very slight.

As the WebP format is not widely discussed in the literature, we may show similarities and differences to methods working on other file types. Format-based techniques are dependent both on the standard and the software. It is best visible in simple injection. A similar technique may be applied to BMP images. Then, depending on the software used, another image may be displayed [72]. In WebP files, such behavior has not been observed. Thumbnail and metadata methods are based on the same structures (Exif and XMP) that are found in other file types. The difference is that in the thumbnail, we may choose from available algorithms of JPEG steganography, whereas in metadata, participants should invent more creative strategies. Chunk injection exploits the extendability of the standard, so other RIFF-based files may potentially use this method. Embedding data in transparent pixels is also possible in PNG images, as confirmed by the conducted tests. On the other hand, indexed images such as GIF have a limited color palette and may benefit from the color indexing method. Information hiding in prediction modes works on formats that use a specialized encoder which predicts unknown pixel values. From popular image types, only WebP implements such an algorithm. The container selection technique may be used in almost any type of file. However, WebP images are a good choice, as a single image may be compressed with various parameters, creating multiple similar versions of the same file. In this way, the container database grows significantly and users are able to transmit longer messages with fewer data sent.

There are some similarities and differences between WebP and JPEG steganography. Format-based methods may be applicable for both image formats, because JPEG and WebP files allow for metadata and thumbnails. The details may vary, for example, the encoding

method or embedding algorithm for the thumbnail image. On the other hand, data-based techniques are different for WebP and JPEG. While JPEG steganography usually focuses on discrete cosine transform to conceal secret information, the WebP format has more possibilities because of its frame prediction. This mechanism, derived from VP8, gives the possibility to hide a message during predictive coding. Color indexing is not available for JPEG images because this format is lossy; however, WebP supports lossless pictures as well. The same is true for transparency. The container selection approach is not necessarily connected to file format, but WebP images have more parameters that users may adjust, so their potential in steganography is greater. In this way, a single image may contain more than one value, which gives more effective transfer than in JPEG images. Therefore, we may conclude that the WebP format offers more potential techniques of data hiding than JPEG.

Considering the amount of data possible to conceal, some methods offer fixed capacity, while others depend on some parameters or traits of the carrier. With simple injection, we are able to append data of any size to the cover image in exchange for increasing file size. This is because the data are added at the end of the file where they are not constrained by any marker or rules imposed by standard. Metadata and chunk injection techniques are limited by RIFF format specification, as chunk size is stored on four bytes. The maximum possible chunk size is then 4,294,967,295 bits, which is about 536.87 MB. However, the real capacity is a bit less, as image data and headers also take up space. For high-resolution images, a few megabytes may be used and the available space should be reduced by this amount. The capacity of thumbnail information hiding depends on the selected embedding algorithm. For F5, it is about 13% of the image size [6]. For other methods, the capacity may be equal to a number of non-zero coefficients, so it may vary for plain and detailed images. Data-based techniques are normally very sensitive to carrier contents. For example, the maximum possible length of a message hidden behind transparent pixels depends on their number. Similarly, in the prediction mode algorithm, the more blocks, the more bits we are able to embed. Luma blocks are 16×16 px and chroma blocks 8×8 px, which are later divided into subblocks of size 4×4. Therefore, the number of blocks is connected to image size, which has its own restriction of 16,383 pixels per side. In color indexing transform, each pixel may encode a single bit; therefore, the maximum capacity is equal to the number of pixels. The last approach uses a database of potential carriers that are stored in multiple variants (or recomputed every time). Because each container may encode one character, in a single conversation, we are able to transmit the number of characters equal to the image set size at maximum. With different encoding, this value may vary. For example, the parties may decide to diminish the amount of transferred data to one bit per image in order to arouse less suspicion. The summary is presented in Table 1.

Table 1. Comparison of capacities of presented methods.

Category	Method	Capacity
Format-based	Simple injection	unlimited
	Thumbnail	algorithm-dependent
	Metadata	up to a few hundred MB
	Chunk injection	up to a few hundred MB
Data-based	Transparency	3× transparent pixel number bytes
	Prediction modes	number of block bits
	Color indexing transform	number of pixel bits
	Container selection	image set size characters

For format-based methods, execution times are almost invulnerable to image contents. This is because messages are hidden in specific fields or after the image data. In simple injection, embedding time results simply from disc operations; on an SSD drive, such times are counted in microseconds. Recovering is about two times larger, as it needs to read the stego file, finding the end of the original WebP image and storing the remaining

part on the disc. Thumbnail and metadata hiding are similar in terms of time after the thumbnail is ready. Then, both methods place the secret in a specific chunk and add it to the carrier. Both embedding and extracting are counted in microseconds. In the thumbnail technique, additional time is needed for thumbnail creation, which is dependent on the steganographic algorithm. For data-based methods, times depend strongly on the carrier. When hiding secret data in transparent pixels, times grow about 30% compared to image conversion to WebP. Extraction times are comparable. In the prediction mode algorithm, it turned out that focusing only on modes with appropriate identifiers shortened execution times by about 24%. This is because, in this step, the reconstruction of the image part and its score are computed. Avoiding this time-consuming code allowed for shorter compression. Recovering the message does not increase the time considerably. The container selection approach is characterized by the largest times of database generation. The reason for this is that every image needs to be prepared in multiple versions. After that, secret extraction is very fast. It only requires finding the corresponding file in its own group, which may be accomplished by comparing precomputed hashes.

Considering steganalysis, an adversary may apply a few approaches. To detect format-based techniques, the adversary should analyze the carrier in search of anomalies. In our opinion, the easiest to detect is simple injection, because image size is stored in the RIFF container as a segment size. Then we may compare this value to the real image size and find discrepancies. The most difficult to detect is hiding a message in a photo's unique ID, as this field contains a random hexadecimal value which is impossible to distinguish from encrypted data. Other metadata and chunks are visible in plain text, so the adversary may read them; message security may vary depending on the selected embedding algorithm, applied encryption, resemblance of metadata to other common images etc. Similarly, thumbnail steganography detection depends purely on the algorithm used. The specific JPEG steganalysis method should be applied to check whether a thumbnail contains a secret message or not. In the container selection approach, the adversary needs to intercept a longer conversation between parties, as a single image contains no data inside. Only the analysis of multiple messages may lead to the discovery of a hidden channel based on a collection of pictures. Considering data-based methods, the adversary may try to modify the carrier or to analyze its contents. To reveal data hidden under transparent pixels, he may remove the alpha channel. The detection of messages hidden via color indexing is harder, because some images contain similar colors, and the adversary needs to decide whether such similarities are artificial or are natural phenomena present in a specific image. The interesting problem is also the detection of prediction mode steganography. It requires decoding all used modes and deciding if they contain any secret information.

Format-based methods are robust to webp compression, as they are independent of pixels of the image. The one exception is simple injection, in which hidden data are ignored during re-compression and the new image does not contain a secret message. In other techniques, secret data are copied into a newly compressed image to retain metadata and all chunks. Considering data-based methods, some of them may be affected by compression. In the prediction mode technique, applying compression changes final pixel values, therefore leading to destruction of hidden data. Color indexing is only used on images with lossless compression, which means that these pictures are robust. The transparency method described in this paper was also used on lossless images. Finally, the container selection approach is based on the compression, which means that when the adversary applies compression to images in the transmission, he would be able to modify the transmitted message.

When we compare the results of the study in the context of the used software, we may draw some conclusions about the utility of the presented methods. Firefox and ImageMagick were able to open all generated files and display them as expected. However, GIMP sometimes had problems with images that deviated from the standard. Of course, WebP support is implemented in GIMP even if these pictures are rarely used outside the web, but the program sticks most closely to the format. On the other hand, web browsers

have to be more flexible considering pictures, because files on the web are of various versions and may be generated in a number of ways. Competition between browsers is fierce and lack of support or problems with displaying images may incline users to switch to another application. The ImageMagick software suite is designed especially for editing and manipulating digital images. It supports a wide variety of formats and did not have problems even with slightly malformed files. Also, it turned out to be useful in conducting experiments.

Deep learning techniques may be useful in data-based methods. They operate on pixels and may be helpful in both better concealment and detection strategies. In this matter, the WebP format is no different that other image formats, except for the fact that it supports both lossy and lossless compression. On the other hand, it does not concern format-based approaches.

The presented methods are adequate for use on the web. This is because the WebP format is almost exclusively used on the Internet. However, this limitation still gives a lot of possible applications. WebP images with hidden data may be sent by email, uploaded on a server, pasted in a web chat, placed on a website, or shared on social media. The specific embedding and extracting algorithms may vary depending on the medium. The main purpose of using the presented techniques is secret communication, but other personal or business applications are also possible. The fact that the WebP format became widespread in recent years is an advantage from steganographic points of view. This is because uncommon files or any deviations from normal activity may arouse suspicions. On the other hand, popular formats are considered as typical and expected, so their presence is not questionable.

5. Conclusions

Undoubtedly, WebP is a modern format with a lot of features, like transparency, metadata, or animations. It is also versatile, as the user may choose the compression level to find a compromise between file size and image quality, and decide whether to use lossy or lossless mode. The conducted experiments showed that the WebP format is suitable for steganographic purposes and offers numerous embedding areas. Data may be placed in specific fields defined in standard, outside the file data, in user-defined chunks, or in data directly. Concealing secrets in data gives a few possibilities as well. In lossless pictures with an alpha channel, secret information may be hidden in transparent pixels, in indexed images, and neighboring colors may store a payload. The general technique of data hiding during the selection of prediction modes produces images not very different from pure carriers. Moreover, WebP may also be used for container selection, because a lot of possible parameters available for encoding allow for the creation of big image databases. The obtained results prove the potential of the WebP format for steganography, indicating especially high capacity of the described methods. Further studies may explore more features of the WebP format, for example, using animations and special types of frames. Additionally, there is a potential to combine WebP steganography with machine learning techniques. Time will tell the future of the WebP format, but its increasing presence on the web is noteworthy and its possible applications in steganography should definitely have their place among other methods.

Author Contributions: Conceptualization, K.K.; methodology, K.K. and M.R.O.; software, K.K.; validation, K.K. and M.R.O.; investigation, K.K.; resources, K.K.; data curation, K.K.; writing—original draft preparation, K.K.; writing—review and editing, K.K.; visualization, K.K.; supervision, M.R.O.; project administration, K.K. and M.R.O. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ARGB	alpha, red, green, blue
ASCII	American Standard Code for Information Interchange
BMP	bitmap
CVE	Common Vulnerabilities and Exposures
DCT	Discrete cosine transform
DFT	Discrete Fourier transform
WFT	Discrete wavelet transform
EXIF	EXchangeable Image Format
FFT	Fast Fourier transform
GIF	Graphics Interchange Format
ICC	International Color Consortium
IPTC	International Press Telecommunications Council
JPEG	Joint Photographic Experts Group
LSB	Least significant bit
PNG	Portable Network Graphics
PVD	Pixel value differentiation
RGB	red, green, blue
RIFF	Resource Interchange File Format
XML	Extensible Markup Language
XMP	Extensible Metadata Platform

References

1. Passeri, P. Q1 2023 Cyber Attacks Statistics. 2023. Available online: https://www.hackmageddon.com/2023/04/21/q1-2023-cyber-attacks-statistics/ (accessed on 1 August 2023).
2. Yadnya, M.S.; Kanata, B.; Anwar, M.K. Using Phase Coding Method for Audio Steganography with the Stream Cipher Encrypt Technique. In I*nformatics and Computer Science, Proceedings of the First Mandalika International Multi-Conference on Science and Engineering 2022, MIMSE 2022, Mataram, Indonesia, 14–15 September 2022* ; Wijaya, I.G.P.S., Hwang, J., Widodo, A.M., Irawan, B., Eds.; Springer: Dordrecht, The Netherlands, 2022; pp. 66–75. [CrossRef]
3. Liu, M.; Guo, Y.; Zhou, L. Text steganography based on online chat. In Proceedings of the Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing, Kyoto, Japan, 12–14 September 2009; pp. 807–810.
4. Wang, Z.H.; Chang, C.C.; Kieu, T.D.; Li, M.C. Emoticon-based text steganography in chat. In Proceedings of the Asia-Pacific Conference on Computational Intelligence and Industrial Applications, Wuhan, China, 28–29 November 2009; Volume 2, pp. 457–460.
5. Qazanfari, K.; Safabakhsh, R. A new steganography method which preserves histogram: Generalization of LSB++. *Inf. Sci.* **2014**, *277*, 90–101. [CrossRef]
6. Westfeld, A. F5—A steganographic algorithm: High capacity despite better steganalysis. In Proceedings of the 4th International Workshop on Information Hiding, Pittsburgh, PA, USA, 25–27 April 2001; pp. 289–302.
7. Provos, N. Defending against statistical steganalysis. In Proceedings of the 10th Conference on USENIX Security Symposium, Berkeley, CA, USA, 13–17 August 2001; Volume 10, p. 24.
8. Saha, A.; Halder, S.; Kollya, S. Image steganography using 24-bit bitmap images. In Proceedings of the 14th International Conference on Computer and Information Technology, Dhaka, Bangladesh, 22–24 December 2011; pp. 56–60.
9. Furuta, T.; Noda, H.; Niimi, M.; Kawaguchi, E. Bit-plane decomposition steganography using wavelet compressed video. In Proceedings of the Fourth International Conference on Information, Communications and Signal Processing and the Fourth Pacific Rim Conference on Multimedia, Singapore, 15–18 December 2003; Volume 2, pp. 970–974.
10. Bin, H.; Li-Yi, Z.; Wei-Dong, Z. A novel steganography algorithm based on motion vector and matrix encoding. In Proceedings of the IEEE 3rd International Conference on Communication Software and Networks, Xi'an, China, 27–29 May 2011; pp. 406–409.
11. Kipper, G. *Investigator's Guide to Steganography*; CRC Press LLC: Boca Raton, FL, USA, 2004.
12. Nair, A.; Kumar, A.; Sur, A.; Nandi, S. Length based network steganography using UDP protocol. In Proceedings of the IEEE 3rd International Conference on Communication Software and Networks, Xi'an, China, 27–29 May 2011; pp. 726–730.
13. Savateev, E. Design of the steganography system based on the version 4 Internet protocol. In Proceedings of the Siberian Conference on Control and Communications, Tomsk, Russia, 21–22 October 2005; pp. 38–51.

14. Murdoch, S.; Lewis, S. Embedding Covert Channels into TCP/IP. In Proceedings of the IH'05: Proceedings of the 7th international conference on Information Hiding, Barcelona, Spain, 6–8 June 2005; pp. 247–261. [CrossRef]
15. Li, Z.; Sun, X.; Wang, B.; Wang, X. A steganography scheme in P2P network. In Proceedings of the International Conference on Intelligent Information Hiding and Multimedia Signal Processing, Harbin, China, 15–17 August 2008; pp. 20–24.
16. Castiglione, A.; De Santis, A.; Soriente, C. Taking advantages of a disadvantage: Digital forensics and steganography using document metadata. *J. Syst. Softw.* **2007**, *80*, 750–764. [CrossRef]
17. Castiglione, A.; D'Alessio, B.; De Santis, A.; Palmieri, F. New steganographic techniques for the OOXML file format. In Proceedings of the IFIP WG 8.4/8.9 International Cross Domain Conference on Availability, Reliability and Security for Business, Enterprise and Health Information Systems, Vienna, Austria, 22–26 August 2011; Springer: Berlin/Heidelberg, Germany, 2011; pp. 344–358.
18. Shirali-Shahreza, M.; Shirali-Shahreza, S. Steganography in TEXdocuments. In Proceedings of the 3rd International Conference on Intelligent System and Knowledge Engineering, Xiamen, China, 17–19 November 2008; Volume 1, pp. 1363–1366.
19. Shiu, H.; Ng, K.L.; Fang, J.F.; Lee, R.; Huang, C.H. Data hiding methods based upon DNA sequences. *Inf. Sci.* **2010**, *180*, 2196–2208. [CrossRef]
20. Cheddad, A.; Condell, J.; Curran, K.; Mc Kevitt, P. Digital image steganography: Survey and analysis of current methods. *Signal Process.* **2010**, *90*, 727–752. [CrossRef]
21. Singh, A.; Singh, H. An improved LSB based image steganography technique for RGB images. In Proceedings of the 2015 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT), Coimbatore, India, 5–7 March 2015; pp. 1–4. [CrossRef]
22. Elharrouss, O.; Almaadeed, N.; Al-ma'adeed, S. An image steganography approach based on k-least significant bits (k-LSB). In Proceedings of the 2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT), Doha, Qatar, 2–5 February 2020; pp. 131–135. [CrossRef]
23. Zakaria, A.; Hussain, M.; Wahab, A.; Idris, M.; Abdullah, N.; Jung, K.H. High-Capacity Image Steganography with Minimum Modified Bits Based on Data Mapping and LSB Substitution. *Appl. Sci.* **2018**, *8*, 2199. [CrossRef]
24. Mohamed, M.; Mohamed, L. High Capacity Image Steganography Technique based on LSB Substitution Method. *Appl. Math. Inf. Sci* **2016**, *10*, 259–266. [CrossRef]
25. Patel, N.; Meena, S. LSB based image steganography using dynamic key cryptography. In Proceedings of the International Conference on Emerging Trends in Communication Technologies (ETCT), Dehradun, India, 18–19 November 2016; pp. 1–5.
26. Koptyra, K.; Ogiela, M.R. Key Generation for Multi-Secret Steganography. In Proceedings of the 2015 2nd International Conference on Information Science and Security (ICISS), Seoul, Republic of Korea, 14–16 December 2015; pp. 1–4. [CrossRef]
27. Cox, I.; Miller, M.; Bloom, J.; Fridrich, J.; Kalker, T. *Digital Watermarking and Steganography*; Morgan Kaufmann Publishers: Burlington, MA, USA, 2008.
28. Johnson, N.F.; Jajodia, S. Exploring Steganography: Seeing the Unseen. *Computer* **1998**, *31*, 26–34. [CrossRef]
29. Gupta, S.; Gujral, G.; Aggarwa, N. Enhanced Least Significant Bit algorithm for Image Steganography. *IJCEM Int. J. Comput. Eng. Manag.* **2012**, *15*, 40–42.
30. Zhang, T.; Li, W.; Zhang, Y.; Ping, X. Detection of LSB matching steganography based on distribution of pixel differences in natural images. In Proceedings of the 2010 International Conference on Image Analysis and Signal Processing (IASP), Zhejiang, China, 9–11 April 2010; pp. 548–552. [CrossRef]
31. A steganographic method for images by pixel-value differencing. *Pattern Recognit. Lett.* **2003**, *24*, 1613–1626. [CrossRef]
32. Swain, G. Very High Capacity Image Steganography Technique Using Quotient Value Differencing and LSB Substitution. *Arab. J. Sci. Eng.* **2019**, *44*, 2995–3004. [CrossRef]
33. Gulve, A.; Joshi, M. A High Capacity Secured Image Steganography Method with Five Pixel Pair Differencing and LSB Substitution. *Int. J. Image, Graph. Signal Process* **2015**, *7*, 66–74. [CrossRef]
34. Mansor, N.K.; Asraf, S.M.H.; Idrus, S.Z.S. Steganographic on Pixel Value Differencing in Iris Biometric. *J. Phys. Conf. Ser.* **2020**, *1529*, 032078. [CrossRef]
35. Rahman, S.A.E. A comparative analysis of image steganography based on DCT algorithm and steganography tool to hide nuclear reactors confidential information. *Comput. Electr. Eng.* **2018**, *70*, 380–399. [CrossRef]
36. Provos, N.; Honeyman, P. Hide and Seek: An Introduction to Steganography. *IEEE Secur. Priv.* **2003**, *11*, 32–44. [CrossRef]
37. Li, X.; Wang, J. A steganographic method based upon JPEG and particle swarm optimization algorithm. *Inf. Sci.* **2007**, *177*, 3099–3109. [CrossRef]
38. Banoci, V.; Bugar, G.; Levicky, D. A novel method of image steganography in DWT domain. In Proceedings of the 21st International Conference Radioelektronika 2011, Brno, Czech Republic, 19–20 April 2011; pp. 1–4. [CrossRef]
39. Chen, W.Y. Color image steganography scheme using set partitioning in hierarchical trees coding, digital Fourier transform and adaptive phase modulation. *Appl. Math. Comput.* **2007**, *185*, 432–448. [CrossRef]
40. Baby, D.; Thomas, J.; Augustine, G.; George, E.; Michael, N.R. A Novel DWT Based Image Securing Method Using Steganography. *Procedia Comput. Sci.* **2015**, *46*, 612–618. [CrossRef]
41. Hachaj, T.; Koptyra, K.; Ogiela, M.R. Eigenfaces-Based Steganography. *Entropy* **2021**, *23*, 273. [CrossRef]

42. Li, X.; Zhang, T.; Li, K.; Ping, X. A Blind Detection Method for Additive Noise Steganography in JPEG Decompressed Images. In Proceedings of the 2011 Third International Conference on Multimedia Information Networking and Security, Shanghai, China, 4–6 November 2011; pp. 489–493. [CrossRef]
43. Manikopoulos, C.; Shi, Y.Q.; Song, S.; Zhang, Z.; Ni, Z.; Zou, D. Detection of block DCT-based steganography in gray-scale images. In Proceedings of the 2002 IEEE Workshop on Multimedia Signal Processing, St. Thomas, VI, USA, 9–11 December 2002; pp. 355–358. [CrossRef]
44. Baluja, S. Hiding images in plain sight: Deep steganography. In Proceedings of the NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 2066–2076.
45. Xintao, D.; Jia, K.; Li, B.; Guo, D.; Zhang, E.; Qin, C. Reversible Image Steganography Scheme Based on a U-Net Structure. *IEEE Access* **2019**, *7*, 9314–9323. [CrossRef]
46. Bi, X.; Yang, X.; Wang, C.; Liu, J. High-Capacity Image Steganography Algorithm Based on Image Style Transfer. *Secur. Commun. Netw.* **2021**, *2021*, 1–14. [CrossRef]
47. Mare, S.; Vladutiu, M.; Prodan, L. Secret data communication system using steganography, AES and RSA. In Proceedings of the 2011 IEEE 17th International Symposium for Design and Technology in Electronic Packaging (SIITME), Timisoara, Romania, 20–23 October 2011; pp. 339–344. [CrossRef]
48. Saini, J.K.; Verma, H.K. A hybrid approach for image security by combining encryption and steganography. In Proceedings of the 2013 IEEE Second International Conference on Image Information Processing (ICIIP-2013), Shimla, India, 9–11 December 2013; pp. 607–611.
49. Katzenbeisser, S.; Petitcolas, F. *Information Hiding Techniques for Steganography and Digital Watermarking*, 1st ed.; Artech House Computer Security Series; Artech House: Norwood, MA, USA, 2000.
50. Abboud, G.; Marean, J.; Yampolskiy, R. Steganography and Visual Cryptography in Computer Forensics. In Proceedings of the 2010 Fifth IEEE International Workshop on Systematic Approaches to Digital Forensic Engineering (SADFE), Oakland, CA, USA, 20 May 2010; pp. 25–32. [CrossRef]
51. Yuan, H.D. Secret sharing with multi-cover adaptive steganography. *Inf. Sci.* **2014**, *254*, 197–212. [CrossRef]
52. Zhang, X.; Wang, S.; Zhang, W. Efficient steganography based on a data decomposition mechanism. In Proceedings of the Third International Conference on Communications and Networking in China (ChinaCom 2008), Hangzhou, China, 25–27 August 2008; pp. 1248–1252. [CrossRef]
53. Koptyra, K.; Ogiela, M.R. Multiply information coding and hiding using fuzzy vault. *Soft Comput.* **2019**, *23*, 4357–4366. [CrossRef]
54. Zhang, H.; Hu, J.; Wang, G.; Zhang, Y. A Steganography Scheme Based on Fractal Images. In Proceedings of the 2011 Second International Conference on Networking and Distributed Computing (ICNDC), Beijing, China, 21–24 September 2011; pp. 28–31. [CrossRef]
55. Zhang, Z.; Fu, G.; Liu, J.; Fu, W. *Generative Information Hiding Method Based on Adversarial Networks*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 261–270. [CrossRef]
56. Bailey, K.; Curran, K. An evaluation of image based steganography methods. *Multimed. Tools Appl.* **2006**, *30*, 55–88. [CrossRef]
57. Kim, C.; Yang, C.N.; Baek, J.; Leng, L. Survey on Data Hiding Based on Block Truncation Coding. *Appl. Sci.* **2021**, *11*, 9209. [CrossRef]
58. Simmons, G. The subliminal channel and digital signature. In *Theory and Application of Cryptographic Techniques, Lecture Notes in Computer Science, Proceedings of the Eurocrypt 84 Workshop on Advances in Cryptology, Paris, France, 9–11 April 1984*; Springer: Berlin/Heidelberg, Germany, Volume 209, pp. 364–378.
59. Koptyra, K.; Ogiela, M.R. Subliminal Channels in Visual Cryptography. *Cryptography* **2022**, *6*, 46. [CrossRef]
60. Koptyra, K.; Ogiela, M.R. Fuzzy Vault Schemes in Multi-secret Digital Steganography. In Proceedings of the 10th International Conference on Broadband and Wireless Computing, Communication and Applications, BWCCA 2015, Krakow, Poland, 4–6 November 2015; pp. 183–186. [CrossRef]
61. Koptyra, K.; Ogiela, M.R. Biometric Traits in Multi-secret Digital Steganography. In Proceedings of the Conference on New Trends in Image Analysis and Processing—ICIAP 2017, Catania, Italy, 11–15 September 2017; Battiato, S., Farinella, G.M., Leo, M., Gallo, G., Eds.; Springer: Cham, Switzerland, 2017; pp. 313–319.
62. Mansour, R.F.; Girgis, M.R. Steganography-based transmission of medical images over unsecure network for telemedicine applications. *Comput. Mater. Contin.* **2021**, *68*, 4069–4085. [CrossRef]
63. Mansour, R.F.; Parah, S.A. Reversible Data Hiding for Electronic Patient Information Security for Telemedicine Applications. *Arab. J. Sci. Eng.* **2021**, *46*, 9129–9144. [CrossRef]
64. Bankoski, J.; Koleszar, J.; Quillio, L.; Salonen, J.; Wilkins, P.; Xu, Y. V8 Data Format and Decoding Guide. 2011. Available online: https://datatracker.ietf.org/doc/html/rfc6386 (accessed on 2 August 2023).
65. Alakuijala, J. Specification for WebP Lossless Bitstream. 2023. Available online: https://developers.google.com/speed/webp/docs/webp_lossless_bitstream_specification (accessed on 3 August 2023).
66. Aas, J. Studying Lossy Image Compression Efficiency. 2013. Available online: https://blog.mozilla.org/research/2013/10/17/studying-lossy-image-compression-efficiency/ (accessed on 10 August 2023).
67. Goodin, D. Incomplete Disclosures by Apple and Google Create "Huge Blindspot" for 0-Day Hunters. 2023. Available online: https://arstechnica.com/security/2023/09/incomplete-disclosures-by-apple-and-google-create-huge-blindspot-for-0-day-hunters/ (accessed on 10 August 2023).

68. Mills, R. Image Metadata and Exiv2 Architecture. Available online: https://exiv2.org/book/index.html (accessed on 10 August 2023).
69. Camera & Imaging Products Association. Exchangeable Image File Format for Digital Still Cameras: Exif Version 2.32. 2019. Available online: https://fotomagazin.hu/wp-content/uploads/2020/05/CIPA_DC_008_EXIF_2019.pdf (accessed on 2 August 2023).
70. Adobe Developers Association. TIFF 6.0 Specification. 1992. Available online: https://developer.adobe.com/content/dam/udp/en/open/standards/tiff/TIFF6.pdf (accessed on 2 August 2023).
71. Kim, S.M.; Cheng, Z.; Yoo, K.Y. A New Steganography Scheme Based on an Index-Color Image. In Proceedings of the Sixth International Conference on Information Technology: New Generations, Las Vegas, NV, USA, 27–29 April 2009; pp. 376–381. [CrossRef]
72. Coldwind, G. Steganografia w BMP. 2023. Available online: https://www.youtube.com/watch?v=60D-_xH63fg (accessed on 3 August 2023).