

**UNIVERSITY OF BUCHAREST**

**FACULTY OF**

**MATHEMATICS AND INFORMATICS**

**MASTER OF SECURITY AND APPLIED LOGIC**

**Dissertation**

**Video Steganography**

**Graduate**

**DLarisa**

**Supervisor**

Conf. Dr. Boriga Radu

**Bucharest**

**July 2023**

**Abstract**

We live in a world dominated by the use of technology. The 21st century came with this great opportunity of having access to all sorts of information just with a touch or a swipe of our fingers on a tiny screen. Companies realised this was the future in terms of consumer requests and capitalised on it, thus requesting sensitive information from the users in order to perform business transactions. Unfortunately, this puts all of our data at risk, considering that the internet is not a very secure place; not to mention that, in recent years, the number of cyberattacks has multiplied at an alarming rate. Security specialists are working diligently towards the goal of a safe digital space, where all the user’s private needs are met and cared for, but, until then, we, as individuals, have an obligation to try to keep our data as protected as possible, for example, through the use of complex passwords (encryption).

Another great idea for the purpose of warding off potential malicious parties is, instead of leaving the information that we try to keep safe out in the open (even if it is encrypted), to seek to hide it in another channel, so that its existence is not even noticeable. That is the objective of steganography and recent studies have shown promise for such an approach when it comes to this particular area of cybersecurity.

This paper’s goal is to analyse how steganography techniques can be used on a multimedia format of type audio-video, in hopes of obfuscating the presence of data hidden within. Methods for embedding information both in the visual and audio channels will be analysed and the results will be compared. Also, this research will propose a new algorithm that will take advantage of both of these cover channels and it will make use of Hamming code (4, 7) and AES encryption scheme, for additional security purposes. Taking into account the findings of this new suggested method, further research can be performed on it, such as assessing its robustness against compression or evaluating how well it holds up against steganalysis attacks.

**Abstract**

Trăim într-o lume digitalizată. Secolul XXI a oferit această mare oportunitate de a avea acces la tot felul de informații doar cu o atingere sau o glisare a degetelor noastre pe un ecran minuscul. Companiile și-au dat seama că acesta este viitorul în ceea ce privește cererile consumatorilor și au profitat de moment, solicitând astfel informații sensibile de la utilizatori pentru a efectua tranzacții comerciale. Din păcate, acest lucru pune în pericol toate datele noastre, având în vedere că internetul nu este un loc foarte sigur; ca să nu mai vorbim de faptul că, în ultimii ani, numărul atacurilor cibernetice s-a înmulțit într-un ritm alarmant. Specialiștii în securitate lucrează cu sârguință cu scopul de a crea un mediu digital cât mai sigur, în care toate nevoile private ale utilizatorului sunt satisfăcute și respectate, dar, până atunci, noi, ca indivizi, avem responsabilitatea de a ne proteja datele și a le oferi un plus de securitate prin utilizarea de parole complexe, spre exemplu.

O altă idee interesantă în ceea ce privește protejarea informațiilor delicate față de părțile malițioase ar fi să încercăm să le ascundem pe un alt canal, astfel încât existența lor nici să nu fie observată, în loc să le păstrăm „la vedere”, chit că ar fi criptate. Acesta este obiectivul steganografiei, iar studiile recente au arătat că este o arie promițătoare de cercetare, în ceea ce privește o astfel de abordare.

Scopul acestei lucrări este de a analiza modul în care tehnicile de steganografie pot fi utilizate pe un format multimedia de tip audio-video, în speranța de a ascunde prezența datelor încodate în acestea. Vor fi analizate metode de încorporare a informațiilor atât în canalul vizual, cât și în cel audio și vor fi comparate rezultatele. De asemenea, această lucrare va propune un nou algoritm care va profita de ambele canale, menționate anterior, și va folosi codul Hamming (4, 7) și schema de criptare AES, pentru o creștere în cadrul nivelului de securitate. Luând în considerare observațiile acestei noi metode sugerate, pot fi efectuate cercetări suplimentare asupra ei, cum ar fi evaluarea robusteței sale împotriva compresiei sau evaluarea rezistenței la atacurile de steganaliză.

Contents

[**1. Introduction** 5](#_Toc137667046)

[1.1. Motivation 5](#_Toc137667047)

[1.2. Objectives 6](#_Toc137667048)

[1.3. Personal contribution 6](#_Toc137667049)

[1.4. Chapters’ overview 6](#_Toc137667050)

[**2. Theoretical Framework** 8](#_Toc137667051)

[2.1. About steganography 8](#_Toc137667052)

[2.2. Image steganography 11](#_Toc137667053)

[*2.2.1. Digital images* 11](#_Toc137667054)

[*2.2.2. Color models for digital images* 12](#_Toc137667055)

[*2.2.3. Literature overview* 14](#_Toc137667056)

[2.3. Audio steganography 18](#_Toc137667057)

[*2.3.1. Digital audio* 18](#_Toc137667058)

[*2.3.2. Literature overview* 20](#_Toc137667059)

[2.4. Hamming code (4, 7) 23](#_Toc137667060)

[2.5. AES 25](#_Toc137667061)

[2.6. Mersenne Twister 27](#_Toc137667062)

[**3. Proposed algorithm** 28](#_Toc137667063)

[3.1. The sub-algorithm for the image channel 28](#_Toc137667064)

[3.2. The sub-algorithm for the audio channel 31](#_Toc137667065)

[3.3. Assumptions taken 33](#_Toc137667066)

[3.4. Implementation 34](#_Toc137667067)

[3.5. Performance analysis 34](#_Toc137667068)

[**4. Conclusions** 39](#_Toc137667069)

[**5. Bibliography** 40](#_Toc137667070)

# **1. Introduction**

## 1.1. Motivation

Life in the modern day is characterized by easy access to and use of the online environment, which both enhances and significantly eases users’ lives through obvious benefits. However, with the digitization of a substantial number of areas of our everyday lives, there is a huge increase in the risk, in terms of security, to which we are exposed in the virtual sphere. It is therefore no wonder that there is an ever-increasing demand for new and improved means of protection.

The presented paper has as its main subject the idea of confidentiality. An impressive amount of data is transmitted daily on the Internet. A huge chunk of it can be considered sensitive, whether we refer to mundane family photos posted on social networks, which have an emotional bond to us, to bank accounts and passwords, on which the well-being of our lives depends.

Of course, in order to hide important information, one can make use of standardized encryption methods, as well as secure communication protocols or certificates, which come to the user’s aid, preventing unauthorized access by malicious third parties, which could intervene and compromise them. Cryptography is the area of interest for these functionalities, but it has an obvious disadvantage in the sense that encrypted data is easily identifiable by simple observation, thus drawing attention to its potentially valuable character and making it vulnerable to attackers who follow it. With these ideas in mind, we can deduce that a good starting point in securing sensitive information would be to mask it so that it passes unnoticed, as harmless and mundane material, through the data traffic and does not become the target of attackers.

Steganography aims at the idea presented above, being a safer alternative for the transmission of personal data in the online environment and thus becomes a topic of interest for solving the problem faced by contemporary society, through the use of what it is called as „security by obscurity”. This area of research is not new by any means and it can be found in more well-known applications such as invisible watermarking or even in famous works of art, such as „*Inferno*” by author Dan Brown.

## 1.2. Objectives

Through this paper, the author aimed to present the defining concepts, techniques and some well-known steganographic designs that will supplement typical digital security measures and give an additional layer of protection against malicious third parties.

Additionally, after a review of the specialized literature, the author discovered that video steganography algorithms tend to only deal with either the visual channel (video frames - images), or the audio channel (audio frames), but scarcely with both, at the same time. And so, another objective of this research is to implement a secure approach that will take advantage of both cover channels, with the ability to transmit a confidential message in a modified form, imperceptible to the human eye, which does not raise possible suspicions and preserves the original content in an exact manner.

## 1.3. Personal contribution

Despite the fact that the paper solely tackles video steganography as its main subject, substantial study into imagine and audio steganography (implicitly proper steganographic methods for both) was conducted in order to implement a well-rounded algorithm, emphasizing the transformation of the message into a harmless channel, undetectable by the human eye or ear. Improvements are additionally provided to some of the most well-known algorithms in the domain of steganography, which are integrated with elements from the field of cryptography, in order to boost the degree of security and fulfill the goal proposed by this work.

Understanding the concepts made available and being able to formulate pertinent conclusions based on the academic research was a great challenge. Due to limited knowledge in this field, ample documentation was required to comprehend, interpret, and apply the theoretical notions examined. The ultimate target for this paper was to submit a meaningful contribution to the field of video steganography.

## 1.4. Chapters’ overview

The paper comprises four chapters, the first being the current one, which focuses on presenting the motivation for researching this topic and outlines the goals pursued in this endeavour.

The second chapter will discuss in detail the theoretical framework needed in order to better understand the concepts utilised inside the proposed algorithm of video steganography. Both image steganography and audio steganography will be analysed, focusing on the methods for embedding the secret data and their advantages and disadvantages. The Hamming code (7, 4), AES encryption standard and the Mersenne Twister algorithm will also be brought up.

The third one is comprised of the proposed video steganographic method and its requirements in terms of implementation. The algorithm is split into two smaller ones, one for the image channel and one for the audio signal – these sub-algorithms present different methods for embedding the data in order to increase security and make the overall technique more difficult to crack. The topic of performance will also be tackled here.

Finally, the last chapter is the conclusion of this paper.

# **2. Theoretical Framework**

## 2.1. About steganography

First and foremost, before venturing deeper, it is critical to have a clear understanding of what steganography is and what it does. The concept of steganography can be seen as both art and science and involves the skillful use of techniques designed at hiding confidential data within seemingly harmless messages. The end goal is to make sure that that only the expected recipient is informed of the existence and content of the message, while persevering secrecy from all other parties. From an etymological point of view, the term comes from joining the Greek words *steganos* (original Greek: *στεγανός*), which means *hidden/concealed*, and *graphia* (original Greek: *γραφία*), which means *writing* and has been used since the end of the XVI century [1].

Steganography allows Alice to secretly send secret messages to Bob without anyone knowing. The message is embedded in another entity, known as ***cover object*** [[1]](#footnote-1). The resulting output, commonly referred to as a ***stego-gram***, is intended to closely resemble the original cover file, but also incorporate the hidden information [2]. This can be easily performed using digital data. In the unfortunate case of interception by a malicious party, it is the obligation of steganography to guarantee that the communication is perceived as naive by the adversary.

Historically speaking, steganography has appeared in rudimentary forms since ancient times. The earliest recorded case of steganography dates back to 500 BC, when the historian Herodotus writes the story of Histaiaeus. Him, being a prisoner, wanted to send a message to Aristagoras of Miletus to revolt against the Persian king. To prevent the message from being intercepted he shaved the head of a willing slave, tattooed the message on his scalp and waited until the hair grew back in order to dispatch him as messenger [3]. In the modern period, during the Second World War, the Germans invented the microdots, complete documents, images and plans reduced in size to that of a dot and attached as a punctuation mark, described by J. Edgar Hoover, FBI Director at the time, as „*the enemy’s masterpiece of espionage*” [4]. And the examples can go on.

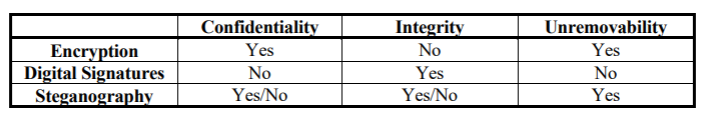
Steganography has developed a lot, especially in recent years, due to the rapid and large-scale evolution of technology. In today’s world, digital steganography is based on encoding bits from a harmless medium, such as text, pictures, audio, or video files, with secret data, so that the probability of malicious third parties detecting the existence of the secret message using techniques of existing steganalysis, to reach as close as possible to zero. Resistance to distinct procedures aimed at altering files without loss of information (example: compression or rescaling), but also improving the problem of the storage capacity of secret messages in cover objects, especially pictures and text files, which have limitations due to their low size compared to other existing multimedia formats, are other important objectives of steganography, just to name a few.

One of the most well-known examples of modern steganography is the concept of ***watermarking*** [[2]](#footnote-2). Steganographic methods are typically not robust or show limited robustness against data tampering. On the other hand, offers additional features that resist attempts to remove hidden data. Thus, watermarking, to the detriment of steganography, is used whenever cover files (such as banknotes) are widely distributed, potentially reaching parties who know the existence of the secured information and wish to alter it [5]. Among the applications of watermarking we list copyright protection, protection against duplication (of interest is the subject of digital piracy or banknote counterfeiting) and content authentication.



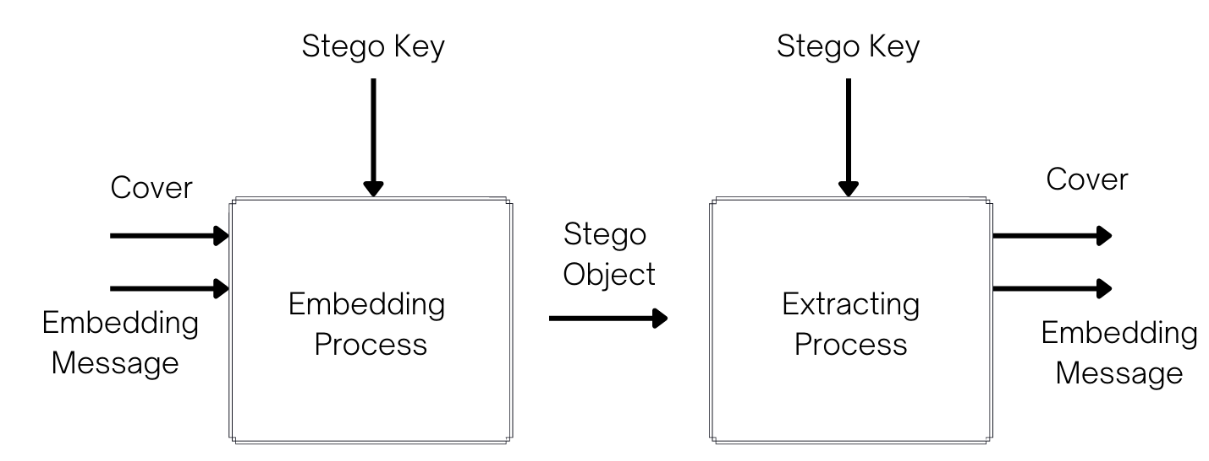
Figură 2.1.1 - Link between Cryptography, Steganography and Watermarking [6]

As we know, cryptography (from Greek: *kryptos* – *κρυπτός* – *secret/hidden* and *graphia* – *γραφία* – *writing* [7]) represents the process of concealing or transforming information in such a manner that its confidentiality is upheld and only access by authorized individuals, being uninteligeble otherwise [8]. But why do we prefer steganography over cryptography? Essentially, both have the same goal of being used to ensure data privacy, but they employ totally different mechanisms in order to achieve it. The primary benefit of steganography is derived from its capacity of conceling messages in such a way that they remain undetected during transmission, thereby escaping the surveillance of potential eavesdroppers. Encrypted messages, easily identifiable to the naked eye, impenetrable as they may be, arouse interest and possess the potential for self-incrimination in countries where encryption is illegal. In other words, steganography can be perceived as a more discreet approach rather than its conterpart, when we want to deal with sending confidential data. On the other hand, messages hidden by steganography are easier to extract, if their presence is unveiled. Consequently, for significant enhancements in terms of security measures, a logical first step would involve merging the 2 areas. To better understand the difference between the two domains, the following picture was attached:



Figură 2.1.2 - Encryption vs Steganography [9]

The following diagram shows a basic schematic of a steganographic model. As we can observe, both the cover object and the secret message are introduced inside the steganographic encoder. The output is the stegogram, which is very similar in appearance to the original file. This stego object is presented as input for the decoder, from which the secret message will be extracted:



Figură 2.1.3 - Steganographic System [10]

Depending on the nature of the cover object, steganography can be divided into several types, of which we mention (the most common):

1. Text steganography
2. Image steganography
3. Audio steganography
4. Video steganography

In subsequent discussion, our attention will be directed towards the latter form of steganography. It is straightforward to notice the fact that a video file is composed of both a visual channel (i.e. images/pictures) and an auditory one (i.e. audio). By exploring both of these avenues, we increase the size of the cover object, allowing for a larger amount of secret information to be conceled. Video steganography is therefore of interest for this area of ​​digital security.

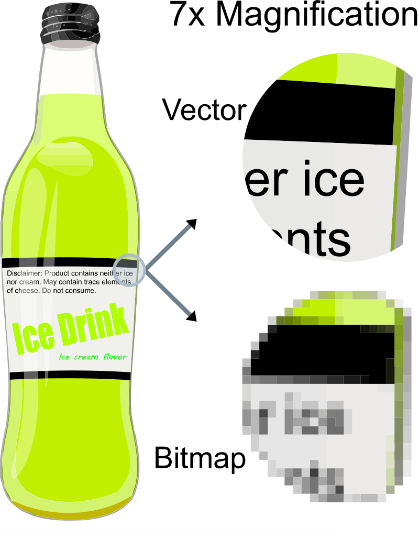
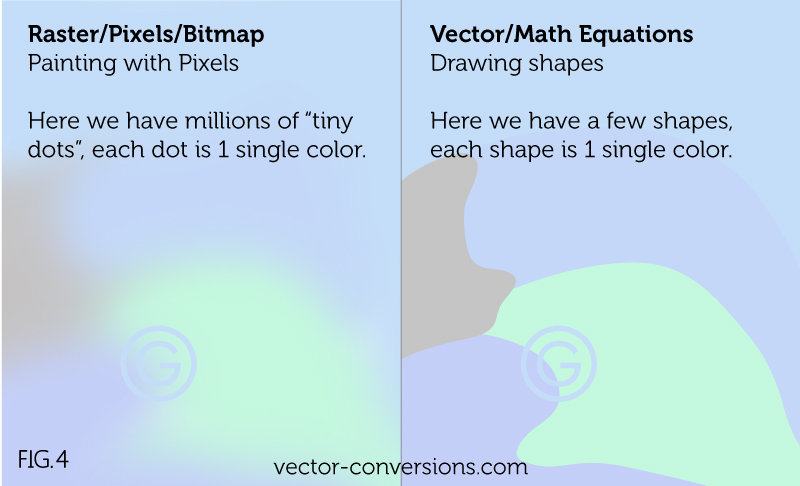
## 2.2. Image steganography

The human visual system (HVS) cannot detect subtle changes that occur in images or in the visual channel of a video file [11]. That’s why pictures are a feasible choice when it comes to choosing them as a cover object.

### *2.2.1. Digital images*

Having gained a comprehensive understanding of the basics of steganograpy, it is time to discuss some concepts related to image processing, in order to recognise their role and figure out how to properly use them in the field of steganography.

An image can be described as a two-dimensional function *f(x, y)*, where *x* and *y* represent the spatial coordinates of the plane, and the value of the function *f*, at each mark identified by the parameters *(x, y)*, is given by the intensity (or *gray level*) of the image at that point. When *x, y,* and *f(x, y)* are all finite, discrete quantities, they define a digital image [[3]](#footnote-3). Any image is composed of a finite sequence of elements, each with a unique value and spatial location. These elements are called pixels [12]. So, we can think of an image as an matrix of pixels, having a fixed number of columns and rows. By itself, the term *digital image* usually refers to *bitmap* images (.bmp format) or *raster* images (.png, .gif, .jpg format etc...) and not to *vector* pictures (.svg format , .pdf, .eps etc...).

Figură 2.2.1.1 - Difference between vector image and bitmap one [13] [14]

Briefly, from a conceptual standpoint, the three categories of image, presented above, can be defined in the following manner:

1. *Vector images* – are characterised by Cartesian coordinates, which are connected by lines and curves to form polygons and other shapes. The points determine the direction of the vector, each of which can have different properties such as color, shape, curvature, thickness, or fill. This kind of images leverages mathematical equations in order to generate the resulting output. Their origin can be found in the field of vectorial geometry [15].
2. *Raster images* – are composed of a dot matrix, that embodies a rectangular grid of pixels. Thus, this particular type of image is characterised by its width and height in pixels, as well as the bit depth, which indicates the number of bits allocated per pixel [14].
3. *Bitmap* – it is also called a bit matrix.

In the following chapters, we will be working with raster and bitmap pictures. Another concept that needs to be brought into disscussion is the color model (or format) of digital images.

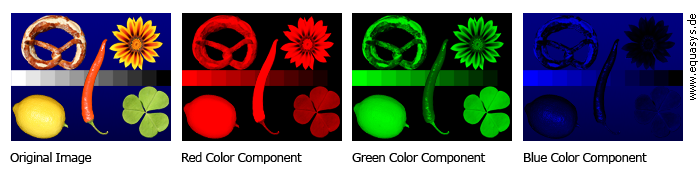
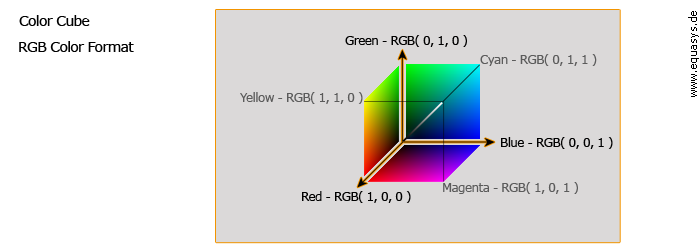
### *2.2.2. Color models for digital images*

A few well known color models are utilised in the domain of picture or video processing.

#### 2.2.2.1. RGB

The abbreviation comes from the three main colours that describe the format: ***R*** – red; ***G*** – greeen; ***B*** – blue. Since these three are additive primaries, each color can be represented as a sum of a red, green, and blue component, thus explaining why this model is the most commonly encountered and used in image processing (LCD, plasma, OLED, etc... make use of this technology). Each pixel in the image is composed of 3 values (red, green, blue) which are 8-bit (or 1-byte) representations of numbers in the range of 0 – 255, adding up to aprox. 16 million combinations. The three central colours can be computed as three separate vectors, describing a cube [16].

Figură 2.2.2.1.1 - RGB Cube [16]



Figură 2.2.2.1.2 - RGB split [16]

#### 2.2.2.2. YUV

This format is taking advantage of the HVS in the sense that the human eye is more sensitive to the light, so to the luminance component, rather then the colour component, so, the latter, can be reduced without concern for losing quality (this process is known as *chroma-subsampling*). ***Y*** stands for the brightness information, while ***U*** and ***V*** represent the chrominance channels. This model was developed for analog multimedia (historically, was first used for colour TVs) [16].



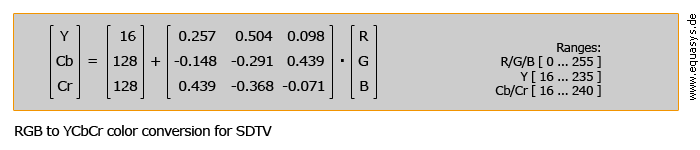
Figură 2.2.2.2.1 - YUV split [16]

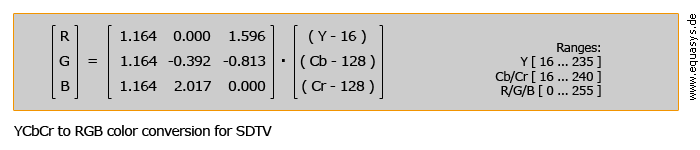
#### 2.2.2.3. YCbCr

This model is very similar to the YUV format, as it also has a luminance channel (***Y***) and 2 chromatic components (***Cb*** and ***Cr***). Because of this, often times it gets confused with the YUV model. In comparison, YCbCr model was designed to support digital image / video processing (expecially, JPEG and MP4 – MPEG encoding) [16].

Knowing these facts, it is plain obvious that, for our research, we will be focusing on the ***RGB*** and ***YCbCr*** colour formats. In order to be able to switch between them, the following formulas of conversion can be used:

Figură 2.2.2.1 - RGB to YCbCr [17]





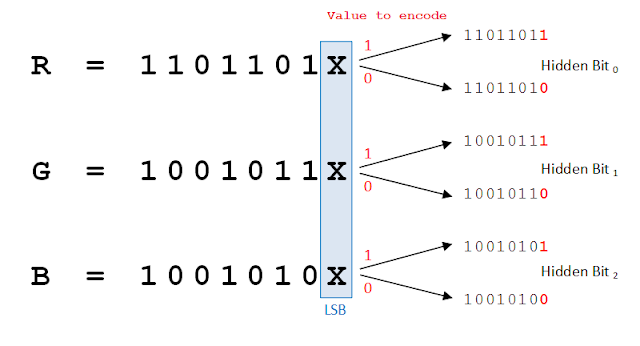
Figură 2.2.2.2 - YCbCr to RGB [17]

### *2.2.3. Literature overview*

In the field of image steganography, various methods have been proposed in recent years, most of them based on the theory of substitution systems (in the spatial domain). Such a method attempts to replace redundant parts of the cover object with the secret message. The main disadvantage for such an approach is the relative weakness to changes brought to the cover environment (such as resizing).

Below, we will review a classification of the most common image steganographic techniques, depending on the changes made to the cover file, as part of the message embedding process, as presented also in [18]:

1. **Spatial domain methods**: where the message is directly embedded into the pixel:
   1. ***LSB*** (Least Significant Bit)***:*** is the easiest to implement, the most efficient and the most common technique to be used, because the human eye cannot easily distinguish between insignificant differences in colour. It comprises of embedding bits from the secret message into the LSB-s of the pixel. This technique can be extended to 2, 4 or even 8 bits, but this can cause distortion and noise, resulting in a quality-lossy picture. Another downside of this method is the fact that the cover object cannot be compressed, cropped, resized, enhanced, or have any other procedure applied to it that would change the pixels, so this algorithm works with either .bmp or .png format, but not .jpeg, for example.



Figură 2.2.3.1 - LSB technique [18]

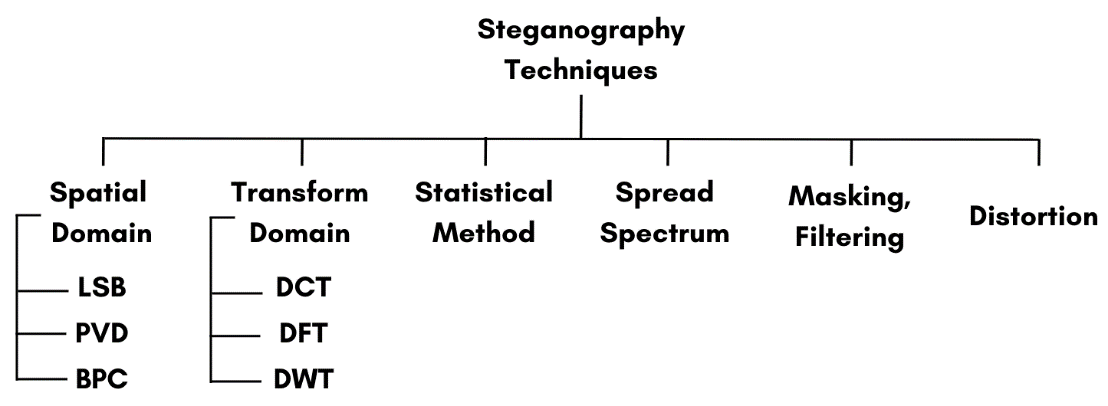
As discussed previously, each pixel has 3 color channels, so it can hide 3 bits of secret information. Thus, in a 255x255 resolution image (eg: *Lenna.png*; approximately 65,000 pixels ~ 200 kb) we can incorporate, with approximation, 25 kB of sensitive data, which is the equivalent of 12.5 pages of a book. In addition, LSB also has the benefit that a significant number of bits in the secret message already corresponded to those in the cover image. Statistically speaking, only half of the bits in an image will need to be changed, on average, to hide the desired data [19]. This approach will be used for our modified algorithm.

As I stated above, the original technique does not withstand attacks, either of statistical type (analysis of the introduced noise), or structural type (robustness, compression, etc...). Therefore, improvements must be made to achieve better performance. Gupta et al. suggested to hide the information only in the blue channel of a pixel, as it is the least visible to the human eye [20]. M. Juneja and P.S. Sandhu suggested to first convert the secret data using S-DES and then detect the smooth edges of the cover image through Canny filter and Hough transform, after which the pixels from it will be randomly selected using a pseudo-random number generator. 1 LSB of red, 3 LSB of green, and 4 LSB of blue are replaced by cipher message bits in the selected pixel [21]. H. Farhan and Zena Alwan used 2 XOR operations in their algorithm. The first XOR takes place between the LSB from the red channel of the pixel and first bit from the message. The second XOR is executed on the outcome of the initial XOR operation, along with the least significant bit of the green channel. The computed output will take the place of the LSB from the blue channel [22]. In 2015, a hash based 2-3-3 LSB algorithm was suggested by Manjula and Danti, where each pixel stores a total of 8 bits. Specifically, two bits are allocated to red, while three bits each are allocated to green and blue channels. Incorporating a hash function for embedding secret information is a common practice in information security. Specifically, in this study, a hash function denoted as *m = k % l* was employed (*m* = bit position within pixel; *k* = bit position; *l* = number of LSBs taken into account for the embedding process) [23].

In the context of video steganography, Eltahir et al. introduced a technique that uses LSB, tranforming the video into individual frames, treating them as static images and embedding the secret message as follows: 3 bits in red, 3 in green and 2 in blue. Only about a third of the video size is used for the actual embedding process [24]. M. Thaneshawri et al. suggest making use of all the frames’ video in order for the secret data to be successfully recovered. This method can be compared to secret key cryptography, as all *N* static frames are used in a similar way to a key in order to reconstruct the message; proces achieved using XOR operations [25].

* 1. ***PVD*** (Pixel Value Differencing)***:*** Two individual pixels (adjacent) are selected from the cover object and the algorithm determines whether they are situated within the smooth regions or the margins. PVD is calculated and the bit from the secret message is concelead in accordance with the calculated difference [26].
  2. ***BPCS*** (Bit-Plane Complexity Segmentation)***:*** This method involves the conversion of the secret message into binary form, which is subsequently utilized to substitute the noisy regions present within the cover medium. The HVS possesses a characteristic where it is incapable of recognizing the distinctions within a very complex visual pattern (for example, every patch of sand at the beach looks the same, even thou there are differences) [27].

1. **Transform domain methods**: embeds the data in a tranform domain (such as the frequency one). This sphere is characterized by a high level of abstraction, since the values of the pixel’s intensities and the digital signals are processed. It is an area of interest (especially for picture processing), because it offers the possibility of much more efficient optimization of an image, from the stand-point of quality and/or occupied memory space. The algorithms from this category are more resistent to changes brought upon digital images (such as compression or cropping) [18]:
   1. ***DCT*** (Discrete Cosine Transform)
   2. ***DFT*** (Discrete Fourier Transform)
   3. ***DWT*** (Discrete Wavelet Transform)
2. **Statistical methods**: information encoding is done by modifying several statistical properties of the cover object (usually is split into blocks and if the message bit is 1, then the block is modified, otherwise is left unchanged) and makes use of hypothesis testing for the extraction process [18].
3. **Spread spectrum methods**: most commonly used in military. The secret information is spread over a wide frequency bandwidth [18].
4. **Distortion methods**: embeds the classified data using signal distortion and subsequently ascertains the deviation from the original cover file during the decoding phase [18].
5. **Masking/Filtering methods**: used in watermarking [18].



Figură 2.2.3.2 - Image steganography methods

## 2.3. Audio steganography

In comparison with HVS, the Human Auditory System (HAS) is far more sensitive to minor distortions brought upon the audio signal. Thus, it is a far more complex task to hide a secret message in an audio file. Despite that, it has a great strength when it comes to embedding capacity: according to some research, all of Shakespeare's writings could be contained within one 8-minute song [28]. This is one of reasons why audio signals can be chose as good cover files.

### *2.3.1. Digital audio*

Just like in the previous chapter, there are both analog and digital audios, the meanings of which are identical to the image scenario presented above. Analog audio data can be defined as a continuous depiction of sound waves in the form of electrical voltage, being the direct replica of the original waveform, with changes in the amplitude and frequency of the signal corresponding to changes in the sound wave. Analog audio can be processed using appropriate equipment and has the advantage of preserving the subtle nuances of the sound, but it is more susceptible to noise and degradation during transmission or recording processes [29].

On the other hand, digital audio is the numerical representation of the sound signal, which will inevitably be converted to binary values. Digital audio offers advantages such as noise immunity, precise reproduction, and the ability to store and process audio data in various digital formats [30].

In terms of security advantages over image steganography, we can enumerate [31]:

1. Embedding secret data in audio signals is a safer option due to the existence of fewer steganalysis methods for attacking audio signals (attacks like spatial scaling can be utilised only in image steganography and have no equivalent in the audio realm).
2. Due to the increased complexity of steganography algorithms, statistical analysis is made very difficult, thus increasing robustness.
3. The audio channel has a greater capacity for embedding secret information.

When talking about disadvantages, we need to mention [31]:

1. The steganographic methods are far more complex and difficult to implement on the audio signal.
2. Due to the sensitivity of the HAS, embedding even small amounts of data can prove futile as it can corrupt the audio file.
3. The secret data can be easily corrupted if the audio file is tampered with.

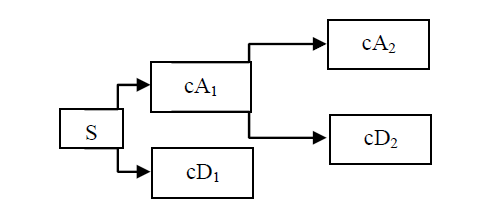
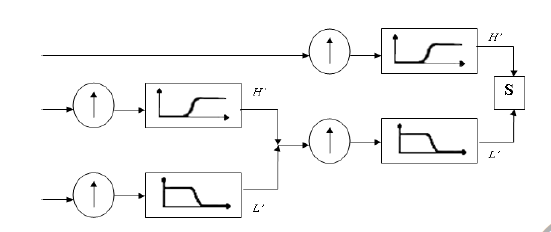
There are three main categories of audio files [32]:

1. *Uncompressed audio files*: as the name suggests, they store the audio data without any compression or loss of quality, thus making the file larger in size. They are managed in professional environments, with the purpose of archiving original recordings. On Windows, the ***.wav*** format is being used, while on macOS, the equivalent is ***.aiff***. Most commonly, a *.wav* file has a metadata header of 44 bytes, where it stores details about the audio, such as *frame rate, number of channels, sample width*, etc... [33]. This will be the format used in our algorithm.
2. *Lossless compressed audio files*: they store audio data without losing any quality, but in a smaller file size (the compression rate is usually 2:1 – i.e. their size is usually half of the original uncompressed audio file). Interestingly, the uncompressed audio signal can be retrieved from such an audio file (example of such format: ***.flac***, ***.alac***).
3. *Lossy compressed audio files*: are the most popular audio format, used widely on the internet, such as ***.mp3***. They perform compression on the audio file, losing some data that might be considered as unimportant to the HAS, in order to reduce the file size.
4. *Compressed audio formats for specific applications*: such as MIDI (Musical Instrument Digital Interface – representing actual music notes and commands, rather than actual waveforms), these files are specifically designed for certain platforms [34].

### *2.3.2. Literature overview*

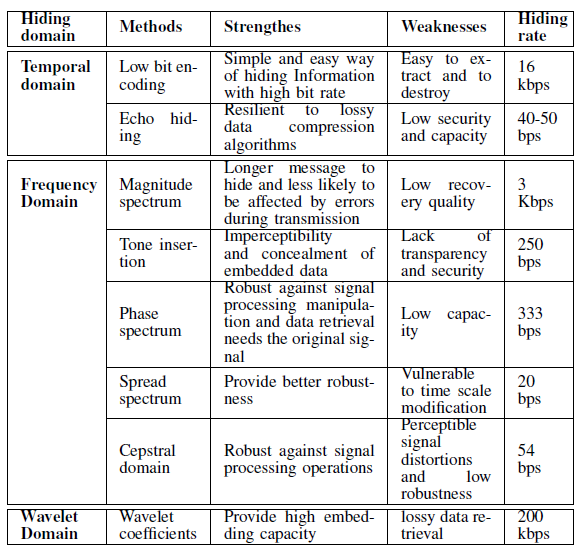
As mentioned above, audio steganography is far more complex to implement than image steganography, so one can notice less research done in this area. Below, we will review a classification of the most common steganographic techniques performed over audio files, as they are discussed in [35] [31] [36]:

1. **Temporal domain methods**: the message is embedded directly into the binary form of the audio signal:
   1. ***LSB*** (Least Significant Bit): similar to image steganography, LSB is the easiest algorithm to implement, but it is incredibly vulnerable to attacks and susceptible to data loss. A lot of papers focus on searching for ways to improve this technique, using, for example, a genetic approach, that will find bits in „deeper” layers, which will minimally affect the sound if changed.
   2. ***Using metadata:*** this technique suggests embedding the secret information inside the header metadata.
   3. ***Echo hiding:*** the idea is to introduce echo (resonance) to the host signal that will carry the secret message. However, on audio signals with a big enough silent interval, this approach becomes less successful because the echo becomes more noticeable.
2. **Tranform domain methods**: take advantage of the HAS’ weakness to detect weaker frequencies near stronger resonant ones. They wield better results in terms of robustness and resistance against attacks:
   1. ***Tone insertion***: these strategies rely on the insertion of lower tones into the audio, which are inaudible in the presence of much higher ones.
   2. ***Phase coding***: A phase is the current state of a sound’s wave. A phase shift in an audio signal is not easily detected by the HAS, so the message is encoded as them. Essentially, data is multiplied by an M-sequence code known to both sender and receiver, so, in case of corruption, there are still copies of the secret message hidden throughout the audio file.
   3. ***DCT*** (Discrete Cosine Transform): is able to present the audio signal as the result of cosine functions at different frequencies. The embedding process involves applying DCT to regions of the cover object and hiding the message inside the obtained coefficients. This approach is a very secure one, as it is somewhat resistant to lossy compression, cannot be easily detected by attacks and the resulted audio does not contain noticeable artifacts (if heavy compression is not being used). The formula to calculate the DCT coefficients of a 1D sequence of *N* length is as follows (we refer to ***DCT-II***, which is the most used one, noted as *DCTcoef* below; ***DCT-III*** also named ***IDCT***, i.e. Inverse Discrete Cosine Transform, is the inverse function of DCT-II and, below, is named *f(x)*):
   4. ***DWT*** (Discrete Wavelet Transform): with it, the audio signal can be seen as a pair of (frequency, location in time information). It is far superior to DCT, as it creates less noise, is very robust and cannot be easily detected, being resistant to common steganalysis techniques, although it can cause lossy data retrieval. DWT uses the Haar wavelet transform. It breaks down an audio signal into 2 groups of coefficients, based upon 2 filters: low and high. cA1 (low frequency; approximation coefficient) is obtained after passing the signal through the low filter, while cD1 (high frequency; details coefficient) is obtained after passing the signal through the high filter. The filtered signals are downsampled by discarding alternate samples, reducing the resolution of the signal. Then, using Divide et Impera, we apply the same process first on cA1 and then on cD1 and so on. The inverse process, for reconstruction of the original data, just follows the same steps, but in reverse order. The distribution of embedded data across numerous frequency levels makes it more difficult for steganalysis methods to detect concealed information.

Figură 2.3.2.1 - 2-levels DWT and reconstruction (IDWCT) [37]

* 1. ***Spread spectrum:*** The secret information is spread over a wide frequency bandwidth.



Figură 2.3.2.2 - Pro and cons for audio steganography methods [35]

Based on the information provided above, the best course of action for audio steganography is to use DCT or DWT. The proposed algorithm will make use of both of these techniques, in a manner similar to the one implemented in [37].

## 2.4. Hamming code (4, 7)

Due to its characteristics, Hamming(7, 4) is a type of code that is often utilised in digital communication policies. The Hamming code (4, 7) can also be represented as [38]:

According to the mathematical formula, we call *C* a block-code over *F* (code’s alphabet), where *F* is a finite set, if:

A code has code-words. *n* is the length of code *C*; in our case 7, i.e. the code-words will consist of 7 „*letters*”. The value *q* is equal to the number of different characters inside the alphabet *F* (*q* = |*F*|, i.e. cardinality of the code); because we have *q* = 2, that means that Hamming code is binary (consists of only 0 and 1 – letters ↔ *F* = = {0, 1}). The size of the code is equal to , which in our case is 24 = 16 (so 16 code-words). The Hamming distance is defined as:

The minimal distance for the code *C* is:

If we know the length of the code *C* (*n*), the minimal distance (*d*) and the number of elements (*M* = |*C*|, i.e. the number of possible code-words), then we can define *C* as a (*n, M, d*)-code over *F* (Hamming code (7, 4) is equivalent to writing (7, 24, 3)). From above, we know that Hamming code has *d* = 3, so:

Meaning that it can detect up to 2 errors that can occur inside the codeword, locate their positions and correct up to one error. This set of properties is very useful when transmitting and receiving a message.

It is also a linear code, which denotes from the fact any linear combination of code-words is also a code-word, noted as:

Also, . That is the reason why we can write only Hamming (7, 4). So, to sum it up, a Hamming code (7, 4) consists of 7 bits, where 4 bits are the message that needs to be encoded (*k* = 4) using 3 parity bits (*d* = 3). Because of the linearity property, this code has both a generating matrix and a parity-check one:

The matrices were changed slightly in order to get the original 4-bit message as the last 4-bit part inside the Hamming code.

Let’s see an example to better understand how this code works. We assume that we have the message *m* = (1, 1, 0, 1). In order to get the Hamming code, *H*, we multiply *m* with *G*, and the result will have modulo 2 applied to it:

We get *H* = (0, 0, 0, 1, 1, 0, 1). We note that the last four bits are the original message that we wanted transmitted (which will be helpful in our proposed algorithm). To check if the message is correct, we multiply using *P* and the result has modulo 2 applied to it. We must obtain the syndrome vector *Z* = (0, 0, 0) if it is an error-free message:

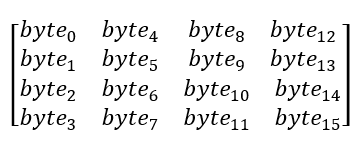
For our *H*, we get Z = (0, 0, 0). Let’s assume that during the transmission, due to a noisy channel, one of the bits has flipped and we get *H* = (0, 0, 0, 1, 1, 1, 1). If we multiply *H* with *P*, the syndrome vector *Z* becomes (1, 1, 1), which is equal to the 6th row of the parity-check matrix. This implies that the 6th bit from *H* has changed, so, the correct code is (0, 0, 0, 1, 1, **0**, 1), and the original message is (1, 1, 0, 1) – the last 4 bits.

## 2.5. AES

AES (Advanced Encryption Standard) is a subset of the family of Rijndael algorithms, block-type algorithms [39]. It has the following characteristics:

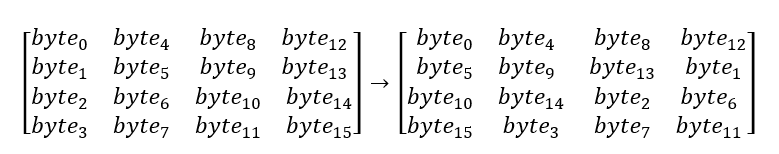
* The block is 128 bits long;
* The encryption key is of variable length and can take values from the set {128, 192, 256}.
* It is a symmetric encryption algorithm.

AES was selected by the US government to protect classified information and today is used around the world to encrypt sensitive data, becoming the standard for cybersecurity. The National Institute of Standards and Technology (NIST) started experimenting with AES in 1997, when, following investigation and tests, it was noticed that DES (Data Encryption Standard) began to become vulnerable to brute force attacks (the following year, DES will be cracked in only three days). The strength of AES is its resistance to various cryptographic attacks, such as brute-force, or linear and differential cryptanalysis. For optimal security, we will use the 256-bit key (the largest offered by the standard). AES works on bytes rather than bits. The algorithm has a block length of 128 bits, i.e. 16 bytes, which are stored in a 4∙4 matrix, just like this:



The size of the key used specifies the number of rounds of successive transformations that convert the plaintext into the encrypted message. For a key of size 256, like the one chosen, 14 rounds are required. Each round consists of four operations (*SubBytes, ShiftRows, MixColumns, AddRoundKey*), and the algorithm is presented as described in [40]:

1. *KeyExpansion*: the round keys are derived from the original encryption key, each depending on the previously created one, using the "AES Key Schedule" algorithm.
2. *AddRoundKey:* between the initial state (the plaintext) and the first round key, the XOR operation is performed.
3. **13 rounds of 4 operations each:**
   1. *SubBytes:* each byte is replaced by another, according to the S-box substitution table;
   2. *ShiftRows:* is a transposition step in which the last three rows of the matrix are shifted by a certain number of steps (if we start counting the rows from zero, then the values of the first row are shifted to the left by one column; on the second row with two columns and the third with three, and row 0 remains unchanged):



* 1. *MixColumns:* each column of the state matrix is multiplied by , obtaining a new matrix;
  2. *AddRoundKey:* XOR between the corresponding round key and the matrix.

1. **The final (14th) round:**
   1. *SubBytes*
   2. *ShiftRows*
   3. *AddRoundKey*

Block ciphers have two key properties that make them immune to attackers, namely confusion and diffusion. Confusion means that the input and output have no connection with each other. Diffusion is the property that makes it so small changes in the input have a huge impact on the output [41]. In AES, confusion is added at the *SubBytes* step, and diffusion occurs at the *ShiftRows* and *MixColumns* steps. AES is also computationally efficient because encryption and decryption do not use many resources.

Overall, AES is a fundamental cryptographic algorithm that is critical in current data security, providing a reliable and efficient means of encrypting and protecting sensitive data.

## 2.6. Mersenne Twister

In order to obtain a certain degree of randomness, in such a way that can be replicate it if the same key is being used (in a 2-way communication channel, for example), we can employ the help of the Mersenne Twister algorithm. It is a widely recognised pseudorandom number generator (PRNG) algorithm that whitsteand a test of time and has good statistical features. It was created in 1997 by Makoto Matsumoto and Takuji Nishimura and is named after the Mersenne prime numbers, which play an important role in its design [42].

In math, a Mersenne prime is a prime number that is one less than a power of two, so it has the following formula: , where *n* is an integer.

One of the highlights of this PRNG is its exceptionally long period, of 219937-1, which implies it can create a sequence of nearly 106000 random numbers before repeating. This long period guarantees that the generated sequence is highly unlikely to repeat within practical computation limits. The Mersenne Twister employs a big state vector of 624 words, each comprising of 32 bits, to store its inner state. It makes use of a combination of matrix operations, bitwise shifts, and bitwise XORs to create the pseudorandom numbers.

This algorithm is additionally known for its strong statistical properties, as it demonstrates great equidistribution, meaning that the generated numbers are evenly dispersed over the full range of possible values. Also, it has great dimensionality qualities, making it suitable for various applications requiring high-quality random numbers.

The Mersenne Twister has become the default PRNG in many programming languages and libraries, including Python's ***random*** module and the ***NumPy*** library. Its combination of long period, good statistical properties, and efficient implementation makes it a reliable choice for a wide range of scientific simulations, cryptographic protocols, and statistical analyses.

# **3. Proposed algorithm**

As mentioned previously, there are little to none proposed algorithms for video steganography that focus on embedding data on both visual and audio channels, and even rarer implementations of them can be found. In [43], the authors mention that mainly, the research into video steganography has been maintained to a theoretical stand-point, with little to none moving past the implementation stage. A few software tools can be found online (such as *OmniHide Pro* or *StegoStick*) that only focus on EOF injection, metadata embedding, or DCT, but all of them have been proven weak by the authors and incapable of keeping users’ data secure.

In order for a steganographic algorithm to be considered successful, it needs to follow these criteria, as defined in [44]:

1. It needs to have a high embedding *capacity*, meaning that a reasonable amount of data needs to be hidden inside the cover object, without visible distortion being perceived.
2. It also needs to provide *imperceptibility*, so that the secret information is invisible to the human senses.
3. Lastly, it needs to have a decent amount of *robustness*, meaning that the cover object needs to withstand a reasonable amount of manipulation without drawing the attention of malicious parties.

The proposed algorithm will try to interlink these properties, in order to achieve a tool as secure as possible, capable of protecting user’s private data.

## 3.1. The sub-algorithm for the image channel

As mentioned previously, we will be using a modified LSB technique for our embedding process. The algorithm will try to hide a binary image[[4]](#footnote-4) inside the video cover object (the same process can be replicated using text, instead of an image). The reason for choosing this kind of data to be hidden is because of its versatility. In practise, a photo can be taken of a piece of document – in real life, it is highly unlikely that a user will try to transmit a secret photo of his cat, so colours are not a necessity, as the binary photo can provide enough detail to be retrieved.

**The embedding phase** will respect the following steps:

**Input**: **O** = video (cover object), **P** = photo (data to be hidden), **Key1** = key used to shuffle the video frames, **Key2** = key used to shuffle the photo

**Step 1**: Take the video **O** and turn it into still frames **F**.

**Step 2**: Take each frame from the array **F** and split it into three separate photos, corresponding to the Y, Cr and Cb channels (for comfort, we will be referring to them as **Y**, **U, V** arrays – we transform from RGB to YCbCr).

**Step 3**: Using **Key1**, shuffle all the components of the **Y, U, V** arrays, using the *Mersenne Twister*.

**Step 4**: Take **P** and convert it to a binary photo. Use **Key2** to shuffle it, using the *Mersenne Twister* and turn it into a 1D array **P\_array**.

**Step 5**: Obtain the number of pixels that need to be changed inside a frame by calculating (len(**P\_array**) // 4) // len(**F**).

**Step 6**: Obtain the *Hamming* code, **H**, taking a sequence of 4 bits at a time from **P\_array**.

**Step 7**: Embed the 7 bits from **H** as follows: 3 bits in **Y** array and 2 each in the **U**, **V** arrays. If (**Key1** ^ **Key2**) % 2 = 0, then the first **H** will be embedded at the beginning of **Y, U, V**; the second **H** will be embedded at the end and repeat. Otherwise, the first **H** will be embedded at the end and the second at the beginning and so on.

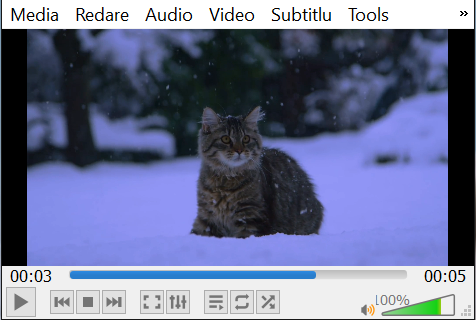
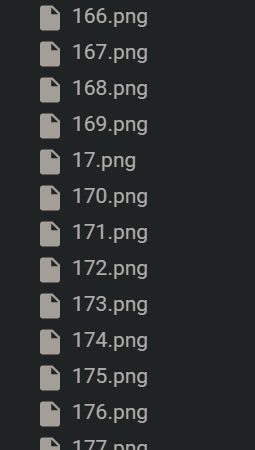
**Step 8**: Reposition the pixels inside **Y, U, V** in the correct order.

**Step 9**: From the 3 arrays, rebuild the colour frames and then the video stream back to the original version.

Example of some steps:

**Step 1**:

Figură 3.1.1 - Original video and frames created

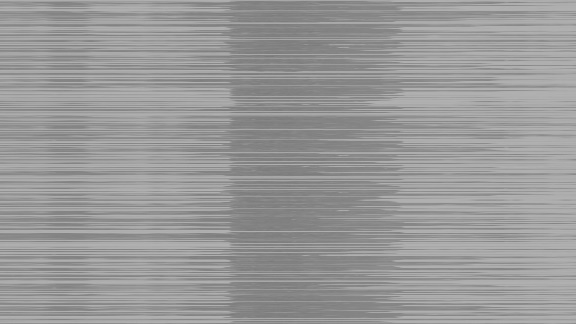
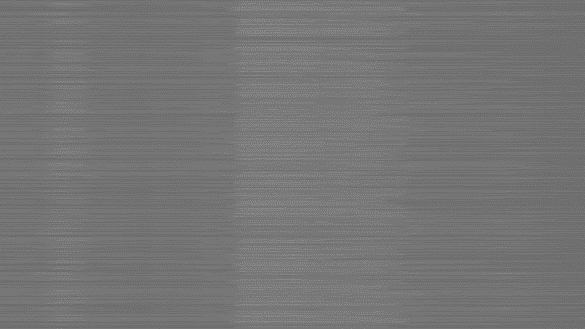
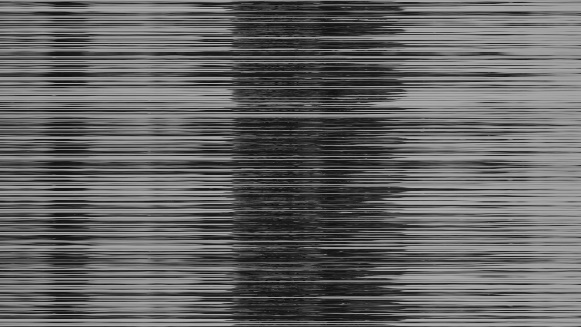
 

**Step 2**:

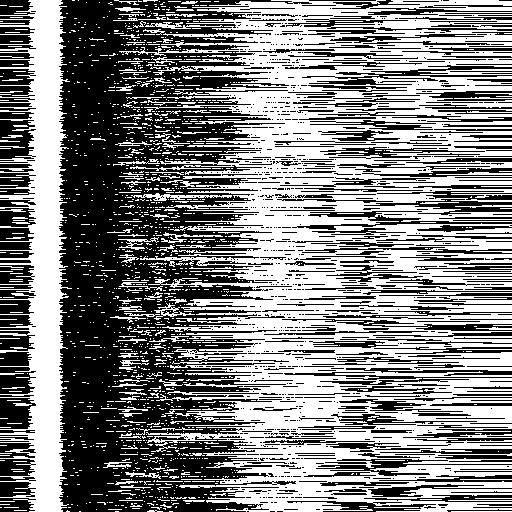
Figură 3.1.2 - In order, U, V, Y photos

**Step 3**:

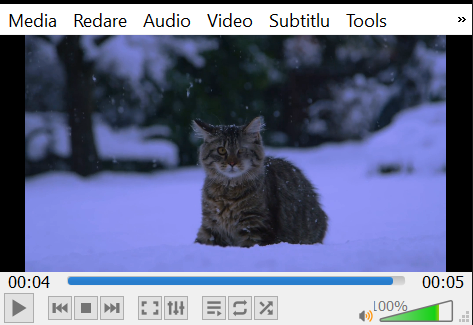
Figură 3.1.3 - U, V, Y shuffled

**Step 4**:

Figură 3.1.4 - Lena.png original, binary, shuffled

**Step 9**:



Figură 3.1.5 - Reconstructed video with hidded data inside it

**The decoding phase** will respect the following steps:

**Input**: **S** = stego-video, **Key1** = key used to unshuffle the video frames, **Key2** = key used to unshuffle the photo, **Width** and **Heigth** of the hidden photo

**Step 1**: Take the video **S** and turn it into still frames **F**.

**Step 2**: Take each frame from the array **F** and split it into three separate photos, corresponding to the Y, Cr and Cb channels (for comfort, we will be referring to them as **Y**, **U, V** arrays – we convert RGB to YCbCr).

**Step 3**: Using **Key1**, shuffle all the components of the **Y, U, V** arrays, using the *Mersenne Twister*.

**Step 4**: Obtain the number of pixels that need to be changed inside a frame by calculating (len(**Width** \* **Heigth**) // 4) // len(**F**).

**Step 5**: If (**Key1** ^ **Key2**) % 2 = 0, then the first **H** will be embedded at the beginning of **Y, U, V**; the second **H** will be embedded at the end and repeat. Otherwise, the first **H** will be embedded at the end and the second at the beginning and so on.

**Step 6**: Apply error-correction on **H**, then the last 4 bits are the secret data. Place the 4 bits in an array **recovered\_img**.

**Step 7**: Reconstruct the hidden image using **Width** and **Heigth**; then apply **Key2** to unshuffle.

## 3.2. The sub-algorithm for the audio channel

We notice that, in order to decode the hidden photo from the video, we need to transmit some information: **Key1** (for shuffling the frames), **Key2** (for unshuffling the hidden photo) and the dimensions of the hidden photo (**width**, **height**). As these inputs are very important for the algorithm, we decided to hide them inside the audio channel. As mentioned in the previous channel, the audio signal is more secure as the algorithm that would be employed will use a safer embedding technique and statistical analysis is also very difficult on it.

**The embedding phase** will respect the following steps, from [37] (very similar, not identical):

**Input**: Write the hidden message into a .txt file, **F** (will contain Key1, Key2, and the dimensions for the hidden photo), **Password** (for the *AES* encryption).

**Step 1**: Encrypt the .txt file using *AES* (**Password**) and turn in into a binary array **F\_array**.

**Step 2**: Extract the audio from the original video and split it into len(**F\_array**) segments.

For each segment of the audio signal, repeat *steps 3 – 11*:

**Step 3**: Apply 1-level *DWT* with „*db1*” as parameter and obtain the coefficients **cA1** (low frequency) and **cD1** (high frequency).

**Step 4**: Obtain vector **V** after applying *DCT* on **cA1**.

**Step 5**: From **V**, obtain sub-vectors **V1** and **V2**, following the equations:

**Step 6**: Obtain **nrmV1** and **nrmV2** after applying the norm of **V1** and **V2**:

**Step 7**: Hide the bit like this ( is an arbitrary value):

If the hidden bit = 1, then:

Else (hidden bit = 0):

**Step 8**: Reconstruct **V1** and **V2**, using the formulas:

**Step 9**: Produce the modified vector **V** from **V1** and **V2**:

**The decoding phase** will respect the following steps:

**Input**: The extracted audio file from the stego-video (**A**), **Password** (for *AES*) and the length (number of bytes - **X**) of the hidden message (the length is obtained after the .txt file is encrypted and turned into a binary array).

**Step 1**: Split **A** into **X** segments.

**Step 2**: For each segment of the audio signal, repeat *steps 3 – 6* from the embedding process.

**Step 3**: If **nrmV1** > **nrmV2**, then **reconstructed\_message** += 1, otherwise **reconstructed\_message** += 0.

**Step 4**: Turn the binary array into text.

**Step 5**: Decrypt the text using *AES* and the **Password**.

**Step 10**: Apply *IDCT* on **V**, in order to get the new **cA1**.

**Step 11**: Apply *IDWT* on coefficients **cA1** (new one) and **cD1** in order to reconstruct the frame.

**Step 12**: Concatenate the frames and obtain the new stego audio.

**Step 13**: Attach it to the stego video and render the new final video.

## 3.3. Assumptions taken

The presented algorithm comes with a list of assumptions:

1. In this paper, the author assumes that both of the parties that try to communicate with each other are in possession of the AES key and the length of the encrypted binary file.
2. Because the algorithm uses LSB for the image channel, lossy picture formats (implicitly video formats) cannot be used. The frames of the video and the hidden image must be of type *.png*. When remaking the video from the frames, the stegano-object will use the FFV1 codec and will be of type *.avi*, as it is a lossless video format. This can be considered a limitation, as it increases the size of the file. If we attempt to use a lossy video format (such as .mp3 that does compression), the hidden data will be lost.
3. Although DCT and DWT can be used on lossy audio formats, after experiments were done, it was noticed that the hidden message was partially lost or retrieved corrupted. This prompted the use of *.wav* as the audio format, because it is a lossless one. This can also be viewed as a limitation.
4. The delta coefficient used for the audio steganography, should be at least 5, in order for the algorithm to work at its best.

## 3.4. Implementation

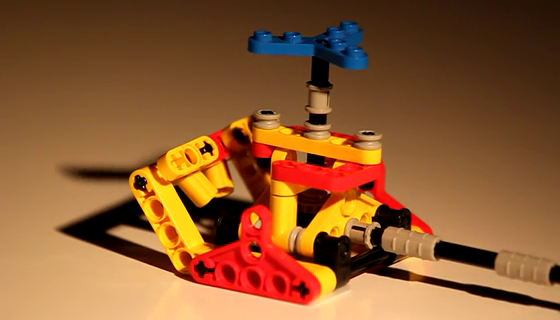
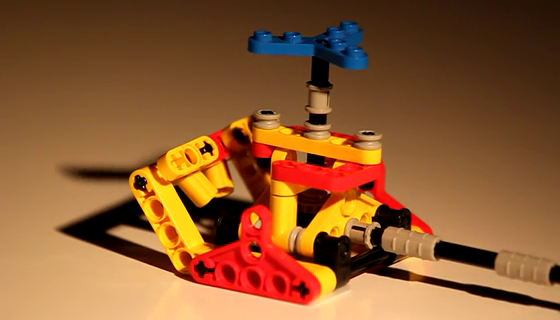
The algorithm was implemented using the Python3 [45] programming language and the Google Colab [46] platform was used as the development environment.

Python was chosen because it makes it easy to work with files, be they text, images, audio, or video. Some of the most important libraries that were used are: *CV2* (for transforming a video into frames, for working with pictures, for converting RGB to YCbCr and vice versa, etc...), *moviepy.editor* (for creating a new video, extracting audio, attaching audio to video, etc...), *wave* (for handling audio files of type .wav), *scipy.fftpack* (for DCT and IDCT), *pywt* (for DWT and IDWT), *pycryptodome* (for AES encryption) or *bitstring*.

Google Colab was used as a development environment because it is an open-source tool, that provides access to Cloud space, with free and powerful GPU resources, but also a disc memory of about 70 GB, which is quite large.

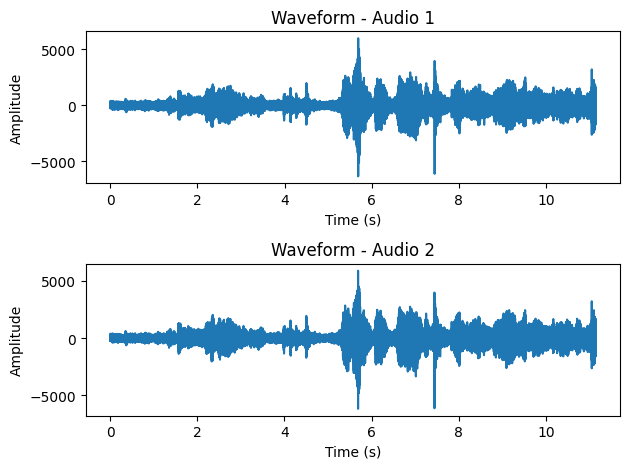
## 3.5. Performance analysis

In terms of imperceptibility, the proposed algorithm is performing well. Below, the first photo is from the original video and the second is from the steganographic one:

Figură 3.5.1 - Comparison of original video frame vs steganographic video frame

Also, both the original audio file and the steganographic one were compared and no differences could be observed (or heared). Below are the waveforms of the 2 audio files (the first is the original one, while the second contains the hidden information):



Figură 3.5.2 - Comparison of the original and steganographic audio files

Let’s test it with specialised measurements, in order to actually see how well it performs. For the image channel we can use PSNR, MSE and SSIM. For the audio channel, we can use PSNR and MSE and also PESQ (without SSIM, which is specifically tailored for image evaluation).

**PSNR** (Peak Signal to Noise Ratio) [44] is the primary metric used to measure how perceptible the resulting noise caused by the algorithm was in the collected results. PSNR is an error metric and is considered to be the ratio of the maximum strength of a digital signal to the amount of unwanted noise affecting it. A higher PSNR value implies less distortion. The calculation formula is (where MAX represents the maximum possible value of the signal):

**MSE** (Mean Square Error) [44] measures the mean square of the „error”; the error being the amount by which the estimator differs from the quantity to be estimated. The smaller the MES is, the smaller the error as well. The MSE is calculated by taking the average of the squared differences between corresponding samples of the original signal and the reconstructed/compressed signal. It provides a measure of the overall magnitude of the error or distortion present in the reconstructed signal. The first formula is for the image channel, while the second can be applied on the audio side:

... where *I(x, y)* denotes the intensity of the original image at position *(x, y)*, and *J(x, y)* represents the corresponding intensity of the steganographic image. *M* and *N* are the width and height.

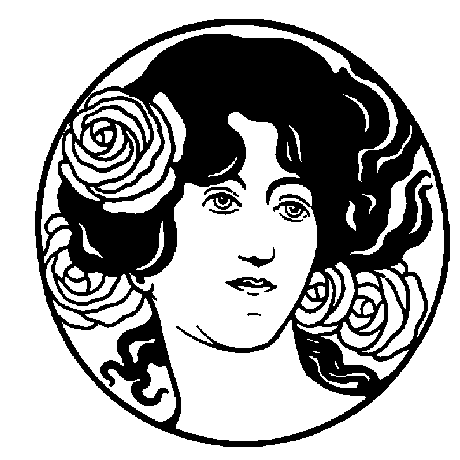
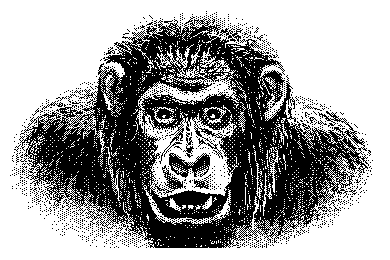
... where *A(i)* represents the amplitude of the original audio at sample index *i*, and *J(i)* represents the corresponding amplitude of the reconstructed audio. *N* represents the total number of audio samples.

**SSIM** (Structural Similarity Index Measure) [44] is used to measure the similarity between two signals, from the point of view of perception. The difference between this method and the previously mentioned techniques is that PSNR and MSE calculate absolute errors, instead, SSIM is a perception-based model that considers image degradation as a perceived change in structural information, while incorporating important perceptual phenomena, among which we list both luminance masking and contrast masking. Structural information is the idea that pixels have strong interdependencies, especially when they are spatially close. These dependencies carry important information about the structure of objects in the visual scene. SSIM can take values between -1 and 1: a score of 1 means that the two photos are very similar and -1 is the opposite.

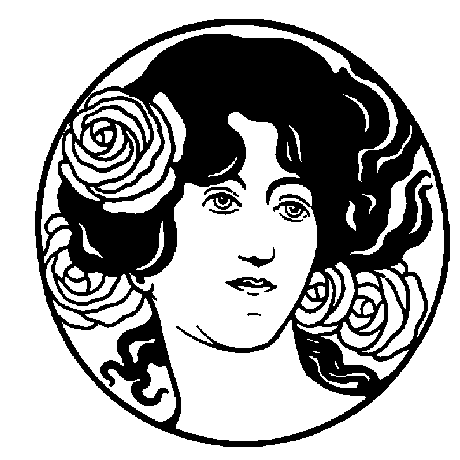
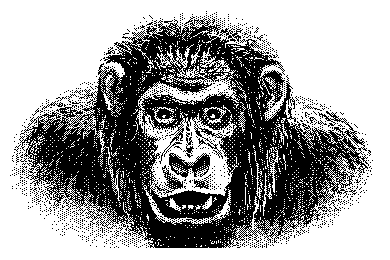
**PESQ** (Perceptual Evaluation of Speech Quality) is a metric used for assessing the quality of the audio signal, especially in terms of speech. It takes values between -0.5 and 4.5, with higher values indicating better perceived speech quality [47].

The secret photos that were embedded:

Figură 3.5.3 - Lenna.png [48] Figură 3.5.4 - Woman.png [49] Figură 3.5.5 - Monkey.png [49]

The photos that were extracted:

Figură 3.5.6. - Extracted hidden photos (Lena, Woman, Monkey)

In the table below, we display the average results of the performance metrics for the image channel (a small video of 5 seconds, 30 fps, 560x320 was used; aprox. 300 frames were extracted – Figure 3.5.1 from above):

|  |  |  |  |
| --- | --- | --- | --- |
|  | *Lenna.png* | *Woman.png* | *Moneky.png* |
| **PSNR** | 51.901 | 53.4754 | 52.227 |
| **MSE** | 0.4756 | 0.3277 | 0.3121 |
| **SSIM** | 0.9889 | 0.9977 | 0.9991 |

Figură 3.5.7 - Table performance image channel - I

As we can notice, we obtain good results. We also need to consider the fact that if we increase the size of the video, the values will get even better, but the algorithm will take longer to perform. For example, for a video of 10 seconds, 24fps, 2560x1440 (aprox. 240 frames were extracted – Figure 3.5.9), we will obtain:

|  |  |  |  |
| --- | --- | --- | --- |
|  | *Lenna.png* | *Woman.png* | *Moneky.png* |
| **PSNR** | 52.978 | 54.9407 | 54.150 |
| **MSE** | 0.3540 | 0.2624 | 0.2701 |
| **SSIM** | 0.9989 | 0.9899 | 0.9974 |

Figură 3.5.8 - Table performance image channel - II

  Figură 3.5.9 – Original frame vs stego-frame of the second video

For the audio sub-algorithm, we obtain again good results, in accordance with the findings of [37]: PSNR was, on average, around 57.769, MSE 0.457 and PESQ was 3.886.

Capacity-wise, the algorithm has a high embedding ability. Let’s try to estimate a minimum limit. The first thing we need to consider is that one second of a video contains anywhere from 24, up to 60, 120, or even 240 frames [50]. Considering that most videos have a minimum of 10 seconds [51] (this is dependent on the social media platform used), we can already work with, at least, 240 frames. We also consider that we have a 480p video (lower qualities are not as common nowadays), meaning that we are working with 480 x 640 pixels in just one frame, a total of 307.200 pixels. One pixel is capable of holding 4 bits of secret data information, so almost 1.2 million bits can be hidden in only one frame of the video. We multiply by 240 frames and we get aprox. 295 million bits, so ~36.7 MB (of course, it is very likely that this will corrupt the video and the malicious parties will know it has been tampered with). For the audio channel, we don’t need to store as much data, as it only contains some text information, but even there we have a reasonable embedding capacity.

# **4. Conclusions**

This paper proposed a new method for hiding secret data inside a video. The idea was to conceal binary photos inside the frames of the video using Hamming code (7, 4) and the Mersenne Twister (of course, the same thing can be achieved using text instead), in a modified LSB technique. The necessary keys and information for accessing the visual channel were hidden inside the audio signal, after encryption with AES and using DCT and DWT algorithms. The performance of the proposed steganographic algorithm is really promising, passing the tests to which it was subjected with flying colours, thus turning it into a starting point for future research. Both the visual and audio channels were used for embedding purposes, making the algorithm different from other methods proposed in the literature. Along with this, a large embedding capacity was achieved, while also maintaing good PSNR, MSE, SSIM and PESQ scores.

There were some limitations discovered, such as:

1. The algorithm takes some time to compile, especially if big videos are being used (either big when it comes to time, or big as in high resolution).
2. In order for the algorithm to work, we need to use with *.png* images, *.avi* and *.wav* files, that are lossless. This is not very memory efficient, but we have to make this trade-off for added security.
3. This means that the algorithm is not very robust, as the secret information will be lost if compression or resizing is applied to the video.
4. The algorithm cannot hide colour images, just binary ones (or text).

This being said, in the future, the algorithm should be made to support colour pictures as well as lossy formats, such as *.mp3* and *.mp4*. For this, Huffman codes could represent a starting point. Another interesting avenue would be to compare the algorithm with others on the internet and see their performances.

In conclusion, the present work represents a starting point in the study of techniques for masking confidential data in a multimedia environment, in this case the digital video files, and detailed information and instructions are provided on the procedures necessary for its implementation.

# **5. Bibliography**

|  |  |
| --- | --- |
| [1] | Merriam-Webster Dictionary, “steganography,” [Online]. Available: https://www.merriam-webster.com/dictionary/steganography. [Accessed 07 June 2023]. |
| [2] | A. Tiwary, A. Gupta and R. K. Tiwary, “A Comparable Analysis of Steganography, Cryptography and Watermarking,” *Journal of Emerging Technologies and Innovative Research,* vol. 6, no. 6, pp. 88 - 92, 2019. |
| [3] | F. A. Peticolas, R. J. Anderson and M. G. Kuhn, “Information Hiding - A Survey,” in *Proceedings of the IEEE, vol. 87, no. 7, pp. 1062–1078*, doi: https://doi.org/10.1109/5.771065, 1999. |
| [4] | Wikipedia Contributors, “Microdot,” Wikipedia, 16 March 2023. [Online]. Available: https://en.wikipedia.org/wiki/Microdot. [Accessed 07 June 2023]. |
| [5] | S. Katzenbeisser and F. A. Peticolas, Information Hiding Techniques for Steganography and Digital Watermarking, USA: Artech House INC, 2000. |
| [6] | S. Soderi, “Acoustic-Based Security: A Key Enabling Technology for Wireless Sensor Networks,” *International Journal of Wireless Information Networks,* vol. 20, no. 1, pp. 45 - 59, 2020. |
| [7] | Merriam-Webster, “cryptography,” Merriam-Webster Dictionary, 23 May 2023. [Online]. Available: https://www.merriam-webster.com/dictionary/cryptography. [Accessed 07 June 2023]. |
| [8] | F. Piper and S. Murphy, Cryptography - A very Short Introduction, Hampshire: Ashford Colour Press Ltd, 2002. |
| [9] | P. Richard, “An Analysis of Steganographic Techniques,” [Online]. Available: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.88.9413&rep=rep1&type=pdf. [Accessed 07 June 2023]. |
| [10] | A. Awad, „A Survey of Spatial Domain Techniques in Image Steganography,” *Journal of Education College Wasit University,* vol. 1, nr. 26, pp. 497 - 510, 2018. |
| [11] | Y.-M. Huang and P.-W. Jhan, “Two improved data hiding schemes,” *4th International Congress on Image and Signal Processing,* vol. 4, pp. 1784 - 1787, 2011. |
| [12] | R. Gonzalez and R. Woods, Digital Image Processing - Third Edition, New Jersey: Prentice Hall, 2007. |
| [13] | Wikibooks, open books for an open world, “Fundamentals of Data Representation: Images,” Wikibooks.org, 2020. [Online]. Available: https://en.wikibooks.org/wiki/ A-level\_Computing/AQA/Paper\_1/ Fundamentals\_of\_data\_representation/ Bit\_patterns,\_images,\_sound\_and\_other\_data#/ media/File:VectorBitmapExample.svg. [Accessed 07 June 2023]. |
| [14] | Vector-conversions.com, „Raster (Bitmap) vs Vector,” vector-conversions.com, 2023. [Interactiv]. Available: https://vector-conversions.com/vectorizing/raster\_vs\_vector.html. [Accesat 07 June 2023]. |
| [15] | N. Chapman and J. Chapman, Digital Multimedia, West Sussex: John Wiley & Sons Ltd., 2000. |
| [16] | Internet Archive - WayBackMachine, “Color Formats,” Archieve.org, 16 April 2018. [Online]. Available: https://web.archive.org/web/20180416084136/http://www.equasys.de/colorformat.html. [Accessed 07 June 2023]. |
| [17] | Internet Archive - WayBackMachine, “Color Conversion,” Archive.org, 23 April 2018. [Online]. Available: https://web.archive.org/web/20180423091842/http://www.equasys.de/colorconversion.html. [Accessed 07 June 2023]. |
| [18] | H. Kaur and J. Rani, “A Survey on different techniques of steganography,” *MATEC Web of Conferences,* vol. 57, no. 02003, doi: https://doi.org/10.1051/matecconf/20165702003, 2016. |
| [19] | J. Krenn, “Steganography and Steganalysis,” January 2004. [Online]. Available: https://www.krenn.nl/univ/cry/steg/article.pdf. [Accessed 07 June 2023]. |
| [20] | S. Gupta, G. Gujral and N. Aggarwal, “Enhanced Least Significant Bit algorithm For Image Steganography,” *IJCEM International Journal of Computational Engineering & Management,* vol. 15, no. 4, pp. 40 - 42, 2012. |
| [21] | M. Juneja and P. S. Sandhu, “An Improved LSB Based Steganography Technique for RGB Color Images,” *International Journal of Computer and Communication Engineering,* vol. 2, no. 4, pp. 513 - 517, 2023. |
| [22] | H. M. Farhan and Z. A. Alwan, “Improved method using a two Exclusive-OR to binary image in RGB color image steganography,” *International Journal of Engineering & Technology,* vol. 7, no. 4, pp. 4295 - 4299, 2018. |
| [23] | G. Manjula and A. Danti, “A Novel Hash Based Least Significant Bit (2-3-3) Image Steganography In Spatial Domain,” *International Journal of Security, Privacy and Trust Management,* vol. 4, no. 1, pp. 11 - 20, 2015. |
| [24] | M. Eltahir, L. Kiah și B. Zaidan, „High Rate Video Streaming Steganography,” *Information Management and Engineering,* pp. 550 - 553, 2009. |
| [25] | M. Thaneshwari and N. Arumugan, “Implementation of Steganography Secret Sharing approach (N,1) for Color Digital Images,” *International Journal of Computer Applications,* vol. 2, pp. 7 - 11, 2012. |
| [26] | K. R. S. a. G. S. A. Pradhan, “Adaptive PVD Steganography Using Horizontal, Vertical, and Diagonal Edges in Six-Pixel Blocks,” *Security and Communication Networks,* vol. 2017, no. doi: https://doi.org/10.1155/2017/1924618, pp. 1 - 13, 2017. |
| [27] | E. Kawaguchi and R. Eason, “Principle and applications of BPCS-steganography,” *Multimedia Systems and Applications,* vol. 3528, 1998. |
| [28] | P. Xue, H. Liu, J. Hu and R. Hu, “A multi-layer steganographic method based on audio time domain segmented and network steganography,” *AIP Conference Proceedings,* vol. 1967, no. 1, doi: https://doi.org/10.1063/1.5039018, p. 020046, 2018. |
| [29] | J. Watkinson, An Introduction to Digital Audio, Oxford: Focal Press, 2002. |
| [30] | J. Watkinson, The Art of Digital Audio, Oxford: Focal Press, 2001. |
| [31] | T. Shay, A. Seady and L. Zaid, “Github: MP3-Steganography-Project,” 23 June 2022. [Online]. Available: https://github.com/tomershay100/MP3-Steganography-Project. [Accessed 11 June 2023]. |
| [32] | Wikipedia Contributors, “Audio file format,” Wikipedia, 07 June 2023. [Online]. Available: https://en.wikipedia.org/wiki/Audio\_file\_format. [Accessed 11 June 2023]. |
| [33] | IBM Corporation and Microsoft, “Multimedia Programming Interface and Data Specifications 1.0,” 1991. [Online]. Available: https://www.mmsp.ece.mcgill.ca/Documents/AudioFormats/WAVE/Docs/riffmci.pdf. [Accessed 11 June 2023]. |
| [34] | Wikipedia Contributors, “MIDI,” Wikipedia, 04 June 2023. [Online]. Available: https://en.wikipedia.org/wiki/MIDI. [Accessed 11 June 2023]. |
| [35] | F. Djebbar, B. Ayad, H. Hamam and K. Abed-Meraim, “A view on latest audio steganography techniques,” *International Conference on Innovations in Information Technology (IIT),* no. doi:10.1109/innovations.2011.5893859, pp. 409 - 414, 2011. |
| [36] | A. AlSabhany, A. H. Ali, F. Ridzuan, A. Azni and M. Mokhtar, “Digital audio steganography: Systematic review, classification, and analysis of the current state of the art,” *Computer Science Review,* vol. 38, no. 100316, 2020. |
| [37] | S. Saadi, A. Merrad and A. Benziane, “Novel Secured Scheme for Blind Audio/Speech Norm-Space Watermarking by Arnold Algorithm,” *Signal Processing,* vol. 154, no. doi: https://doi.org/10.1016/j.sigpro.2018.08.011, pp. 74 - 86, 2018. |
| [38] | M. Prunescu, “Coding Theory,” in *Special Topics in Security & Applied Logic II*, Univeristy of Bucharest, 2023. |
| [39] | D. Selent, “ADVANCED ENCRYPTION STANDARD,” *InSight: RIVIER ACADEMIC JOURNAL,* vol. 6, no. 2, 2010. |
| [40] | Federal Information Processing Standards Publication (FIPS 197), “Advanced Encryption Standard (AES),” Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg, doi: https://doi.org/10.6028/NIST.FIPS.197-upd1, 2001. |
| [41] | P. Hellekalek and S. Wegenkittl, “Empirical evidence concerning AES,” *ACM Transactions on Modeling and Computer Simulation,* vol. 13, no. 4, doi: https://doi.org/10.1145/945511.945515, pp. 322 - 333, 2003. |
| [42] | M. Matsumoto and T. Nishimura, “Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator,” *ACM Transactions on Modeling and Computer Simulation,* vol. 8, no. 1, pp. 3 - 30, 1998. |
| [43] | T. Sloan and J. Hernandez-Castro, “Forensic analysis of video steganography tools,” *PeerJ Computer Science,* vol. 1:e7, no. doi: https://doi.org/10.7717/peerj-cs.7, 2015. |
| [44] | M. Dalal and M. Juneja, “A survey on information hiding using video steganography,” *Artificial Intelligence Review volume ,* vol. 54, p. 5831 – 5895, 2021. |
| [45] | Python.org, “Download Python,” Python, 2022. [Online]. Available: https://www.python.org/downloads/. [Accessed 11 June 2023]. |
| [46] | Google, “Google Colaboratory,” Google.com, 2019. [Online]. Available: https://colab.research.google.com/?utm\_source=scs-index. [Accessed 11 June 2023]. |
| [47] | A. W. Rix, J. G. Beerends, M. P. Hollier și A. P. Hekstra, „Perceptual evaluation of speech quality (PESQ)-a new method for speech quality assessment of telephone networks and codecs,” în *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Salt Lake City, UT, USA, 2001. |
| [48] | Wikipedia Contributors, „File:Lenna (test image).png,” Wikipedia, 15 November 2021. [Interactiv]. Available: https://en.wikipedia.org/wiki/File:Lenna\_(test\_image).png. [Accesat 11 June 2023]. |
| [49] | Johnloomis.org, „Introduction to Digital Images,” Johnloomis, [Interactiv]. Available: https://johnloomis.org/ece563/notes/basics/depth/color\_depth.html. [Accesat 11 June 2023]. |
| [50] | D. Brunner, “Frame Rate: A Beginner’s Guide,” The TechSmith Blog, 14 January 2021. [Online]. Available: https://www.techsmith.com/blog/frame-rate-beginners-guide/. [Accessed 11 June 2023]. |
| [51] | J. Jughes, „How Long Should a Social Media Video Be? Tips for Every Network,” Social Media Marketing & Management Dashboard, 08 March 2022. [Interactiv]. Available: https://blog.hootsuite.com/how-long-should-a-social-media-video-be/. [Accesat 11 June 2023]. |
| [52] | J. L. Mohr, “Analog versus Digital Imaging,” Dan Koboldt Blog, 06 September 2018. [Online]. Available: http://dankoboldt.com/analog-versus-digital-imaging/. [Accessed 07 June 2023]. |

1. We name cover object/file any type of multimedia record that acts as a „*screen*” in steganography and in which the secret message will be integrated, thus resulting in the final steganographic object. [↑](#footnote-ref-1)
2. Despite having significant links to steganography, the topic of watermarking is not the focus of this research and is thus summarized briefly. A more thorough investigation of this branch is left to the reader. [↑](#footnote-ref-2)
3. At the other end of the spectrum, we have analog pictures (such as photographs, paintings, etc...). Analog media is something that doesn’t ought to be converted before you can actually interact with it (digital images are a combination of 0-s and 1-s, a code that needs to be processed by a computer in order for us to be able to see it). Analog can be viewed as more of the traditional means for pictures [52]. [↑](#footnote-ref-3)
4. A binary image is a type of digital image that consists of only two possible values or colours, typically black and white. It is a representation of an image where each pixel is assigned a binary value, typically 0 for black and 1 for white. The term "binary" refers to the use of two levels or states to represent information. [↑](#footnote-ref-4)