information

Article

Transmission of Digital Data in the 5G Era: Compression and Privacy

Bruno Carpentieri * and Francesco Palmieri

Dipartimento di Informatica, Università di Salerno, 84084 Fisciano (SA), Italy
* Correspondence: bcarpentieri@unisa.it

Abstract: The vast majority of compressed digital data that flows nowadays on modern high-speed networks is directly related to human activity. It describes what we do, what we see and photograph, where we go, whom we meet, and specifically every moment of our lives. This brings up issues and concerns regarding the necessity to safeguard user privacy as well as to protect the digital multimedia contents that are delivered to offer new experiences. In this paper, we explore a unified approach to compression and privacy by considering different types of digital data (text, images, sound, and hyperspectral images).

Keywords: compression; privacy; social networks; region of interest (ROI)

Citation: Carpentieri, B.; Palmieri, F. Transmission of Digital Data in the 5G Era: Compression and Privacy. *Information* **2023**, *14*, 135. https://doi.org/10.3390/info14020135

Academic Editor: Lorenzo Mucchi

Received: 16 January 2023
Revised: 14 February 2023
Accepted: 15 February 2023
Published: 18 February 2023

1. Introduction

We live in a digital age and every day a huge amount of digital data is transmitted and received in compressed form. Today most digital data flowing on modern high-speed networks is directly related to human activity and often this data describes what we do, what we see or photograph, where we go, whom we meet and, in particular, every moment of our lives.

With the arrival of new smartphones and new social networks and with the speed of 5G, the network traffic of multimedia documents has significantly increased. The daily rhythms of our lives are becoming increasingly frenetic and fast, generally directly corresponding to the speed with which we can communicate and interact with others, even using the internet and new technologies. However, for everything to be fast and immediately usable, it is necessary to implement a data transmission that is immediate, complete, but at the same time secure.

For obvious reasons, both feasibility and economics, this huge amount of digital information is stored or transmitted in a compressed form. On the other hand, however, this information must often be able to travel or be stored in a secure form, so there is a strong need to combine both compression and security in a single coding technique.

Every digital communication should be based on a data transmission layout that includes the coupling of two heterogeneous operations: compression and security. Digital data security is a challenge that has been going on for years in both asset protection and authentication. The assets to be protected are often of different types: photographs, documents, files, videos, etc.

The identification of users through data circulating on social networks is a hot new problem and malicious or even government agencies could often use these channels to spy on individuals.

It is therefore necessary to protect the privacy of users when digital images or videos are used, for example in current social networks and, even more so, in those that will come and will be based on virtual and augmented reality.

Perhaps the greatest perceived danger of these new social media based on Virtual Reality (VR) or Augmented Reality (AR) is precisely the user's privacy which is considered

at risk because the new virtual and augmented reality technologies can show even clearer what the user is doing, where he is and who he is with. Furthermore, the software of these new realities can easily collect information on the user's identity and behavior, to a much greater extent than, for example, the old social networks or other forms of current digital technology.

In this paper, we explore a unified and effective approach to data compression and privacy maintenance by taking into consideration various digital data types. The next section discusses text protection and compression, and it presents privacy in interactive data compression as a case study.

Section 3 is dedicated to two-dimensional data (images), and it presents a scrambling technique to maintain privacy in a region of interest (ROI) of a digital image. Section 4 considers safety and privacy in the lossless, compressed, transmission of hyperspectral images. Section 5 deals with maintaining privacy in digital audio with appropriate digital watermarking techniques and finally Section 6 outlines our conclusion and describes future research directions.

2. Privacy and Compression of One-Dimensional Data

Lossless compression algorithms are generally used for the compression of one-dimensional data: texts, programs, object codes, and for all data in which the correlation is intended as unidimensional: i.e., sample x is closely correlated with the previous sample and the next one.

Of course, even one-dimensional data must be transmitted safely and securely.

For example, when using mobile devices such as modern cell phones, we want to ensure the confidentiality of the messages and data transmitted (see [1,2]).

If the compression algorithm is dictionary-based, privacy and security can be obtained by reordering, with techniques that maintain secrecy, the indexes pointing to the dictionary elements.

Examples of a unified algorithm that incorporates data compression and privacy have been presented for textual substitution methods, such as the LZW algorithm, and for algorithms based on the Burrows–Wheeler transform such as bzip.

Kelley and Tamassia in [3] present a formal framework to address the security problem when data compression is combined with encryption. In their work, they discuss the safety of the LZW lossless compression algorithm when the management of dictionary indexes is randomized. Ref. [4] describes an encryption algorithm that provides security and compression using compression algorithms of the Bzip family.

Bzip and Bzip2 produce small compressed files with a better compression ratio than other lossless compression algorithms, even if from the point of view of computational efficiency, they are slightly slower than other approaches such as gzip.

The authors of [4] use the Burrows-Wheeler transform and the approach presented in [4] alters the Burrows–Wheeler transform (BWT) using a permutation of the input symbols that are randomly selected in order to make the encoded message secure and undetectable by third parties.

Security of Interactive Compression

If we compress and send messages from a data source that is well known to both Sender and Receiver, then lossless, dictionary-based, data compression algorithms can use the same (static) dictionary for both Sender and Receiver: at the beginning of the communication the Sender will send its dictionary to the Receiver.

For example, in Vector Quantization compression algorithms the compressor constructs its dictionary on a training set of images. This dictionary will successively be used to compress new data. Of course, the compressor must send this dictionary to the decompressor to make communication possible. This is true also when similar techniques are applied to digital images (see [5]).

A large part of the compression methods used in practice that are based on dictionaries are not static dictionary methods but dynamic dictionary methods: they grow up the Sender and the Receiver dictionary at run time. They start with dictionaries that are empty and then they grow the dictionaries by considering data in the already compressed part of the data stream.

The compression performance could be improved if, in a dynamic, dictionary-based, compression method, Sender and Receiver could start with a common, full, dictionary.

El Gamal and Orlistky in [6] study this problem: "Given two random variables X and Y with entropies H(X) and H(Y) and joint entropy H(X,Y), and two persons PX and PY, such that PX knows X and PY knows Y, suppose that the two persons wish to communicate over a noiseless two-way channel so that at the end of the communication process they both know X and Y. How many bits on the average must they exchange and what are the optimal codes?"

They discover that not less than $H(X \mid Y) + H(Y \mid X)$ bits must be transmitted on the average and that H(X,Y) + 2 bits are sufficient and that if the joint probability p(x,y) is uniform then the average number of bits needed is close to $H(X \mid Y) + H(Y \mid X)$.

Moreover, they present randomized protocols that limit the amount of data exchanged in the communication if we can accept a limited possibility of communication errors.

Carpentieri in [7,8] and Carpentieri and Palmieri in [9] present an effective method to solve this problem in the framework of compressed communication. When a compressor and a decompressor have already a good experience of a given source, because in the past the compressor has compressed many messages from that source and the decompressor has itself de-compressed other many messages (not necessarily all the same messages that the compressor has compressed) and if there is an interaction between the communication factions, then it is possible to improve the data compression operations when compression is used for data transmission.

The cost paid is an extremely low chance of decoding errors. If we can accept this, then we can design interactive protocols that permit a Sender and a Receiver to benefit from the information they already have of the data source that produces the messages to exploit their interaction so to minimize the cost of communication.

As an example of everyday usage of this method consider a user that downloads regularly a file (an upgrade, a report, a newsletter, etc.) from a given data source, or a system manager who often downloads an update or a patch for an application or an antivirus, or a mirroring internet site that has to be systematically updated, etc.

Ref. [7] describes a communication protocol that allows a "learned" Sender and a "learned" Receiver to be in communication by using a dynamic dictionary method. The Sender and Receiver's dictionaries are built, starting with prior (and maybe conflicting) instances of source messages that the Sender or Receiver have at hand, so they can differ in several instances.

This protocol improves compression and pays a small, almost zero, chance of communication errors.

When this protocol is used in practice, Sender and Receiver possess many messages from a given information source (not necessarily the same messages and, independently, build up their dictionaries, SenderDictionary and ReceiverDictionary, including in the dictionaries what each of them supposes are the m most common words of the information source.

The sender will transmit a compressed message to the Receiver using the SenderDictionary and Receiver shall decode the message using the ReceiverDictionary. The Receiver can interact with the Sender by sending acknowledgment messages.

The first time a new word is transmitted Sender and Receiver use a protocol based on Karp and Rabin's fingerprinting for the transmission, the subsequent times the word is indexed univocally in both dictionaries, and that index is sent from the Sender to the Receiver, and it is univocally decoded.

To maintain communication privacy when using interactive data compression, we can add to the dictionary management of the encoder and decoder a RandomSwap operation such as the one introduced in [3] in the context of LZW compression.

In particular, when Sender finds a word for which there is already a common index in the dictionaries it sends the index to the Receiver and after that, independently but consistently, they both do a RandomSwap on the dictionaries (using both Sender and Receiver the same random number generator and the same seed) by randomly swapping the transmitted word with another in the common dictionary, so that next time the word will be used a different index will be sent.

In this way, it is possible to prove the security of Interactive Compression by using the same theoretical framework used in [3].

We have also implemented and tested this new secure interactive compression algorithm by compressing some of the books in the test data set used in [8].

Transmitting books in a certain language, which can be sent from a remote source to a client destination, is one possible application of our algorithm.

As in [8] we can improve the transmission/compression process by taking into account the common rules of the language (i.e., the set of most used words of the natural language, the syntax, etc.) which can be considered as shared knowledge between the Sender and the Receiver.

For example, when the Receiver receives the book, it can use a standard online dictionary of the language in which the book is written to decode the Sender's messages.

The Sender sends a book to the Recipient one sentence or word at a time. If the phrase or word has already been encountered the Receiver an index to the dictionary built at run time which contains already shared phrases.

If instead the word has not already been encountered and is therefore not in the shared dictionary, the Sender sends a token that includes the initial character of that word, the length of the word, and a hash value (for example a Karp and Rabin hash) for that word. The Receiver will receive the token and will try to decode the word using its local dictionary. If the token sent by the Sender cannot be decrypted due to the hash value and there is a collision in the receiver's dictionary, then the Receiver asks the Sender to send a new token for the same word but with a different letter (may be the middle letter of the word) and a different hash.

If the Receiver finds a unique word that matches the token in the dictionary, it decodes the word and eventually recognizes it and puts it, as well as the Sender, into the shared dictionary.

If the Receiver does not find any word matching the token in the dictionary, it sends a request for the word to be sent directly as raw data to the sender.

The difference with [8] is therefore in the management of the dictionary which includes the sentences already recognized by both sender and receiver: this is the dictionary which, as in [3], is manipulated to ensure privacy.

Table 1 shows the results obtained. We have compared the original Interactive Compression Algorithm (12 bits hash) and this new Secure Interactive Compression Algorithm (12 bits hash): the compression performances are almost identical to the original approach.

Table 1. Secure Interactive Compression.

Book Title	Original Dimensions	Interactive Protocol	Secure Interactive Protocol
		Compressed Size	Compressed Size
20,000 Leagues Under The Sea	875.5 KB	306.2 KB	306.3 KB
The Wealth Of Nations	2273.1 KB	602.9 KB	602.9 KB
Catcher in the Rye	384.3 KB	122.4 KB	122.4 KB
For Whom The Bell Tolls	937.4 KB	288.2 KB	288.2 KB
The Grapes Of Wrath	935.3 KB	310.8 KB	310.8 KB

The very small differences in compressing the book "20,000 Leagues Under The Sea" depend on the fact that the cost of sending a single pointer is now randomly changed (but on average it is almost the same as before).

3. Privacy and Compression of Bi-Dimensional Data

Nowadays storing and sharing photos online is a common practice adopted by many users. This practice is favored by the significant increase in the performance of the new smartphones capable of taking increasingly detailed photographs and the huge diffusion of social networks and data storage sites.

Every day an ever-increasing number of photographs are transmitted and received through social networks or photo-hosting services, and this raises serious privacy issues.

Social networks typically offer privacy protection solutions that are often rudimentary, not easy to use, and not always fully efficient in terms of maintaining user privacy: the photographs themselves can potentially reveal a great deal of sensitive information about the people depicted.

In many cases, users want to share their photos online but would like to protect specific areas of the images by applying, for example, masking, blurring, or encoding of sensitive areas that can only be seen in their entirety by other duly authorized users.

3.1. Scrambling the Region of Interest (ROI) of an Image

In [9] we consider the problem of obtaining privacy through data manipulation. Our algorithm does not delete the pixels to be hidden but it modifies an area of the photograph in a reversible way that will allow complete decoding only by whoever owns the associated secret key.

The encoding strategy we present is simple and does not significantly affect the file size: it will not be necessary to add many other types of data to allow the complete decoding of the image to the entitled users (the original pixels do not have to be kept).

The strategy is based on scrambling a part of the image.

Scrambling involves the use of two main actors: scramblers and descramblers that share the same key.

Before sending the data, the scrambler manipulates the data stream using the secret key (shared with the descrambler) and then transmits this data stream to the recipient. Once the data has arrived at its destination, the descrambler uses the secret key to perform the reverse de-scrambling operation and thus obtains the original image to show to the receiver. Anyone who does not have the key has no way of decrypting the covered part of the image.

We will apply scrambling to images encoded with the JPEG coding algorithm (Joint Photographic Experts Group) using an approach called JPEG transcoding: that is, scrambling will be performed starting from the compressed JPEG image and in particular it will be applied after calculating the DCT coefficients (Discrete Cosine Transform).

The symmetric encryption algorithm we used in our experiments was an implementation of the Advanced Encryption Standard (AES) with a 256-bit key. Of course, keys of different sizes (128 or 192 bits) or even different algorithms can be used without modifying the actual scrambling procedure.

On the other hand, descrambling will be performed after decoding the JPEG image but before calculating the DCT coefficients.

The basic idea of our algorithm is to select a region of interest (ROI) within the JPEG image and apply the scrambling operation to it.

For example, the ROI could coincide with the faces of the people portrayed in the photo and therefore only those in possession of the key will be able to recognize those people, all the others will see the photograph with their faces blurred by scrambling.

The de-scrambling process will perform the same operations but in reverse order.

The main steps of our algorithm are the following:

1. In the first phase, the ROI is identified and selected, for example through the Viola-Jones algorithm.
2. Scrambling is applied, block by block, by modifying the values of the DCT coefficients already quantized during the JPEG compression process.
3. The modification of the DCT coefficients depends on the selected scrambling level (the value of the scrambling level can vary from 0: no coefficient is modified, to 64: all the coefficients are involved in the modification).
4. A seed is generated to obtain a pseudorandom binary sequence and the signs of the DCT coefficients are inverted for each value corresponding to 1 in the pseudorandom string.
5. The seed for the generation of the pseudorandom sequence is encrypted before being inserted together with the information on the ROI within the JPEG metadata. Due to this procedure, the image size will increase by a few bytes (about 0.37 Kb in the case of a single ROI).
6. At this point the JPEG encoding process continues normally.

3.2. Experimental Results

Figure 1 shows our algorithm applied to the image "Lena" (225 × 400 pixels, 24 bits for pixel), by using in step 3 a value of the scrambling level of 8.

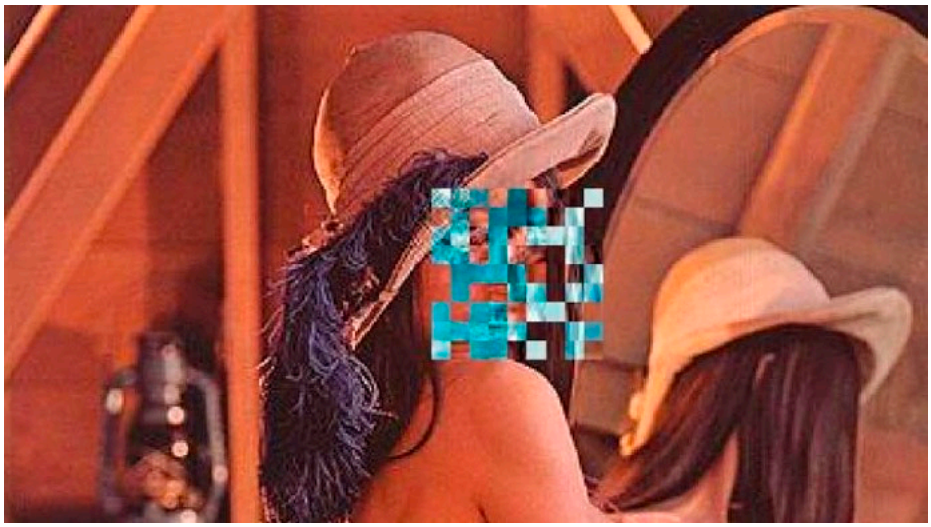

Figure 1. Scrambling *Lena*.

Our approach can also be applied to images that have more than one ROI.

Figure 2 shows the example of an image with 21 ROIs and it shows the result of applying our algorithm with a scrambling level of 50.

The goal here was to protect, as ROIs, human faces.

In our experiments, we have validated the effectiveness of our algorithm on the Group4a [10] and BaoDataset [11] datasets. Our purpose was again to protect, as ROIs, the human faces.

In these experiments, we have tested our approach (with scrambling level values: 8, 16, 32, and 64). After the execution of our scrambling technique, we have then run the same face detection algorithm we have previously used, but this time on the scrambled dataset as input, counting again the number of faces that have been identified before and after scrambling.

The two results in Tables 2 and 3 show the effectiveness of our approach in masking the faces: the higher the scrambling level value the more difficult is to identify the ROI as a face (and of course it shall be harder or practically impossible, to recognize the face).

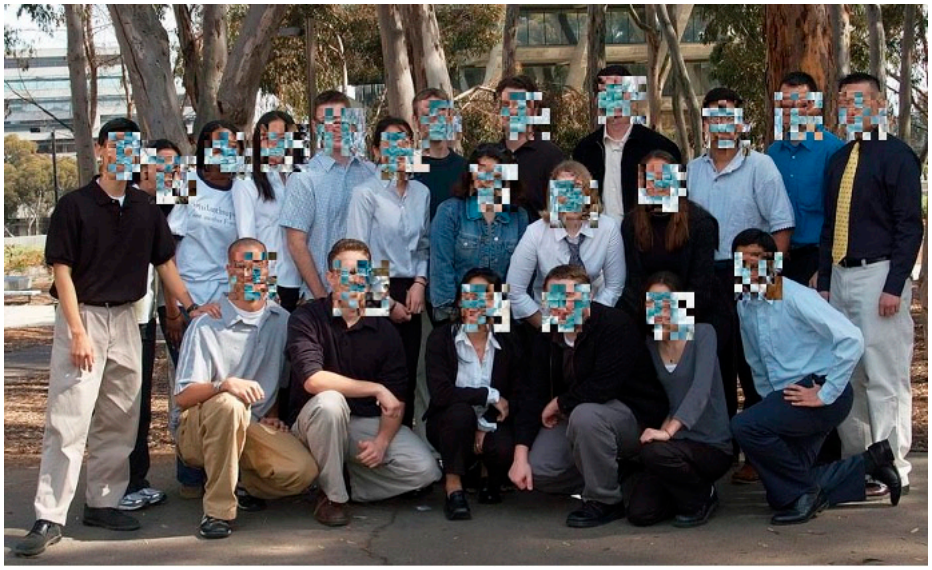

Figure 2. Scrambling multiple ROIs.

Table 2. Experiments on Group 4A.

Scrambling Level	No. of Human Faces	No. of Human Faces after Scrambling
8	4334	3502
16	4334	3001
32	4334	1780
64	4334	170

Table 3. Experiments on *BaoDataset*.

Scrambling Level	No. of Human Faces	No. of Human Faces after Scrambling
8	1357	1024
16	1357	870
32	1357	581
64	1357	71

We can compare scrambling algorithms for digital images in the case of face detection by running the face detection algorithm after scrambling and counting the number of faces that are detected. We have experimentally compared our approach to the approach presented in [12] by implementing it with the Viola-Jones face detection algorithm (the same we use in our approach). We have also implemented it by using the OpenCV face detection tool.

The experiments use a standard dataset (Group4a) and they are targeted to see how many faces are detected after scrambling. The results are shown in Table 4.

Table 4. Comparison of scrambling results (Group4A).

[12] and (Viola-Jones)	[12] and (OpenCV)	Our Algorithm
4334/234	4116/252	4334/170

The first column in Table 4 refers to the approach presented in [12] with face detection by the Viola-Jones algorithm. The second column refers to the approach presented in [12] with face detection obtained by using the OpenCV tool. The third column shows the results

obtained with our approach with scrambling level *64* and this has the best performance. We have also experimented on *BaoDataset* and we have obtained similar performances.

3.3. Better Scrambling

The visual result of scrambling, i.e., deletion of ROI information, can be improved using a secret key and a pseudorandom number generator.

We propose below two different and simple approaches to ROI scrambling that have proved to be very efficient experimentally.

In a first attempt, we scramble all the DCT coefficients by applying a bit-by-bit XOR between all the DCT coefficients and a pseudorandom binary number always chosen on the basis of a seed.

We show examples here in Figures 3 and 4.

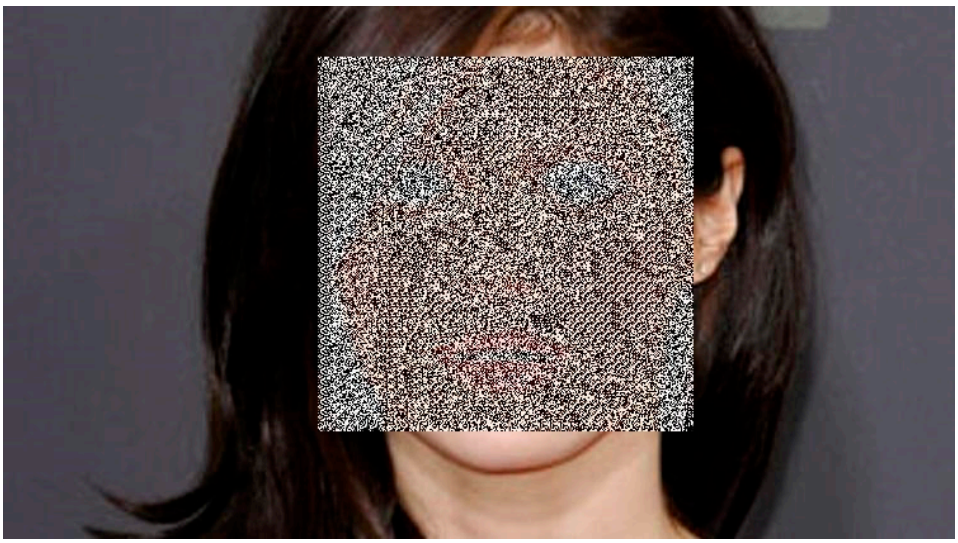

Figure 3. Better scrambling a single ROI: first approach.

Figure 4. Better scrambling multiple ROIs: first approach.

The second approach involves the complete scrambling of all the DCT coefficients but this time instead of the XOR, first inverting all the coefficients and then adding a pseudo-random value chosen based on the seed. We show an example here in Figure 5.

Figure 5. Better scrambling multiple ROIs: second approach.

4. Privacy and Compression of Hyperspectral Images

Remote sensing uses the physical principle that the energy level of each photon of electromagnetic radiation determines its wavelength. This means that any electromagnetic radiation can be described in terms of wavelengths.

In particular, this is also true for reflectance: the percentage of light reflected by an object.

Each object has its own reflectance which depends on its molecular structure, and this makes it possible to identify materials and objects through the spectral analysis of the reflected light.

There are two categories of remote sensors: multispectral spectrometers and hyperspectral sensors.

Both measure multiple wavelength bands.

The typical number of bands for multispectral images varies between 4 and a few tens. Hyperspectral images, on the other hand, have a number of bands ranging from a few dozen to a couple of hundred.

In hyperspectral images, each image element (pixel) is not constituted by a simple monochromatic value (panchromatic or grayscale images), or by a set of values (RGB color images), but by a set of values belonging to the electromagnetic spectrum.

A hyperspectral image is a multidimensional image in which all the information from the entire electromagnetic spectrum of the observed object is collected and processed. Observation of the reflected light spectrum of objects has shown that each of them leaves a unique imprint on the entire electromagnetic spectrum.

These imprints are known as spectral signatures and allow for the identification of the different types of materials that make up the object under observation. For example: With the spectral signature of oil, you can help mineralogists find new oil wells.

The electromagnetic spectrum is the range of all possible frequencies of radiation (electromagnetic waves). With our sight, we can perceive wavelengths between 380 and 760 nanometers (nm) to which we give the name of visible light.

The bands in hyperspectral images represent discrete intervals of the electromagnetic spectrum and produce a continuous spectrum for each pixel represented in the scene.

A hyperspectral image can therefore be considered as a data cube, made up of as many planes as the number of bands that make up the remotely sensed spectrum, and with a width and height equal to the size of the captured area. Ideally, it is possible to consider the two spatial dimensions as if they were resting on the retracted surface and the third (spectral) dimension perpendicular to them.

If you move along the spectral direction, you obtain different values and colors, and if you take one of the remote sensing bands, you obtain a monochrome picture. By taking the information relating to a single point of the image, the continuous spectrum of that specific pixel is obtained.

The knowledge obtained through spectral remote sensing can be applied to various purposes and applications: environmental risk, surveillance, monitoring, historical research, etc.

Hyperspectral remote sensing is therefore present in many real-world applications today. For example, it is used in military, mining applications, geology, ecology, surveillance, archeology, historical research, etc.

However, this technology is increasingly available to the public, so it begins to be used in a wide variety of new ways, making applications in every field of science.

Hyperspectral images are typically acquired via an aircraft or satellite or drone equipped with sensors that fly over the ground whose hyperspectral image is to be acquired. The acquired data is generally communicated in real-time to a base where it is examined and stored.

Given the many possible applications of hyperspectral images, these images are often exchanged or sent via the network and since they are images that sometimes have a high acquisition cost and considerable importance (just think of military applications) they are therefore sensitive to privacy and security, and it is necessary to make special efforts to protect them.

AVIRIS (Airborne Visible/Infrared Imaging Spectrometer) hyperspectral images are those in which at a proper spatial resolution it covers an area of 20 $\times$ 20 m per pixel, the reflected light is divided into 224 contiguous bands, each of them 10 nm wide and the spectrum range goes from 400 to 2500 nm.

The spectral components are acquired with a 12-bit Analog-to-Digital Converter (ADC). The elements of the spectrum, after having performed the calibration and the necessary geometric corrections, are represented with a precision of 16 bits.

Generally, spectral correlation is harder than spatial correlation and the dynamic range and noise levels of AVIRIS data are much higher than those in photographic images: these are the main reasons why the median predictor of JPEG fails on hyperspectral images and ad hoc compression algorithms, sometimes based on linear prediction as the median predictor presented in [13], are generally used to losslessly compress hyperspectral images.

We have considered the protection of the integrity and privacy of the hyperspectral images and our algorithm (see [14]) uses a part of the image as a watermark, so it is possible to detect tampering on the image by examining this watermark.

We are interested in the extraction of a Region of Interest (ROI), intended as a portion (sub-cube) of the original image.

These sub-cubes must be consistent in format and configurations with the original file, and consequently, in the case of hyperspectral images, viewable by the main hyperspectral image viewers.

The idea is to use this ROI as a watermark for the original hyperspectral image and our algorithm, i.e., the process of cutting and injection of a region of interest in the hyperspectral image given in input, can be described as follows:

1. Cut a part of the (hyperspectral) image given in input to obtain a new smaller image: the region of interest (ROI).
2. The ROI (we call it Crop) will be input to a compressor configured with a two-dimensional median predictor. This phase outputs a file containing the prediction errors (prediction file).
3. The prediction file is arithmetically encoded.
4. The file obtained from the previous arithmetic encoding is then injected into the original image and compressed.

The data compression and injection scheme is shown in Figure 6, where I stands for the original image, *C* stands for Compressed crop, *AC* stands for Arithmetic Coding, and finally, *EC* stands for Entropy Coding.

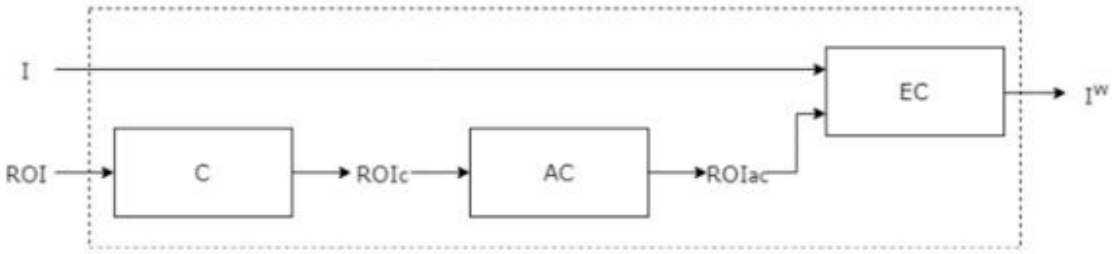

Figure 6. Stages of Compression and Injection of the ROI.

To experimentally test our approach, we have selected the image shown in Figure 7 representing an urban center.

This image will be cut into a smaller region of interest (ROI), which will be compressed to then act as a watermark for the original image. In Figure 8 we show the ROI that we have used with respect to the image depicted in Figure 7.

After the ROI is extracted, it is given in input to the compression method of the framework that will build the prediction file. In our experimental test, the ROI file in this step is compressed from 9.844 KB to 5.034 KB.

This file is further compressed by using arithmetic coding to obtain in our test an even smaller file (2.835) which will finally be injected into the non-ROI part of the image, generating the image with a watermark and possibly a file with additional information to identify the ROI.

In our example, the injection capacity of the original image is about 8.5 kb. Then it is possible to inject ROI of even larger sizes. After performing the stages of extraction and decompression we confirm that the test was successfully performed. The injection of data into images is therefore a complex procedure, especially when the amount of data that may be hidden in the image is not known in advance.

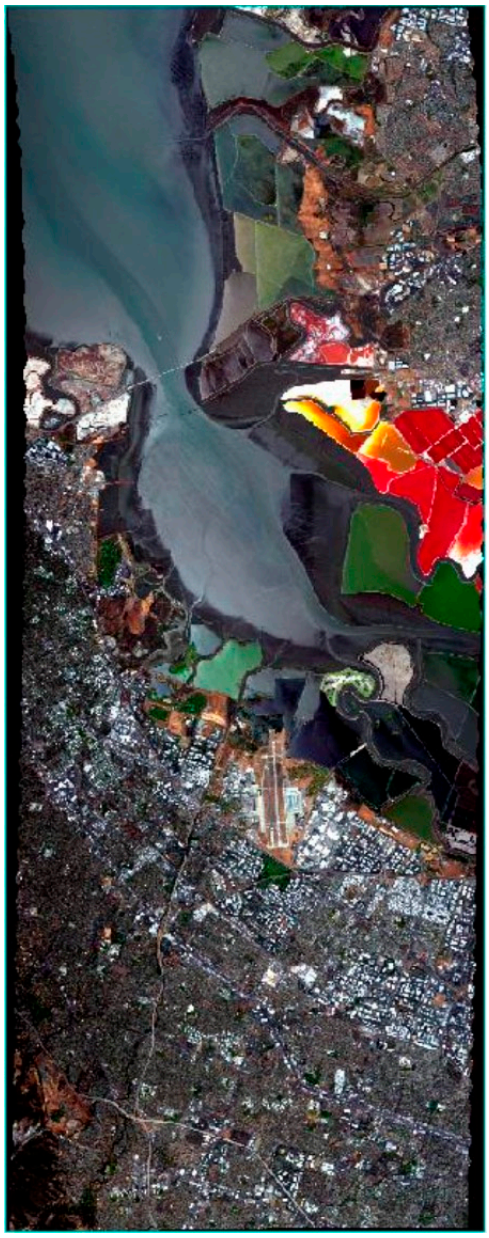

Figure 7. Test image: an urban center.

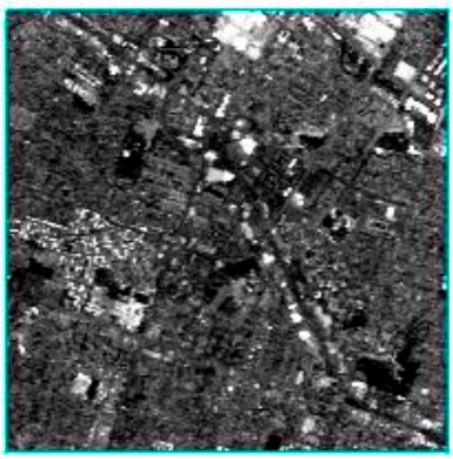

Figure 8. ROI.

5. Privacy and Compression of Digital Audio

Steganography encompasses a set of techniques that allow us to hide messages or information within any type of data, including audio tracks.

We can therefore speak of audio steganography.

Steganography in audio communications has been used since the time of the world wars due to the need for secure communications.

These techniques have come down to the present day, since the main communication channels transmit signals followed by some type of noise that is well suited to hide steganographic messages.

The audio signal is by its nature an analog signal, i.e., a signal that varies continuously over time.

Today, with the advent of new communication technologies and the speed of data transmission of 5G, the use of steganography and data hiding techniques in audio communications has on the one hand, reasons for the search for privacy and freedom of communication and from other gives the possibility of sending unencrypted messages to third parties so that they cannot be intercepted.

Modern data-hiding techniques are heavily dependent on the format in which we represent digital audio.

A popular representation of digital audio is the Waveform Audio File Format (WAV): a standard format, developed by IBM and Microsoft, for coding audio bitstreams.

It is the primary format used in Microsoft Windows systems for audio.

While a WAV file can also include audio in compressed form, WAV audio is commonly not compressed audio in the LPCM (Linear Pulse-Code Modulation) form. For this reason, WAV is often regarded as a lossless audio format that can maintain high sound quality. Data is saved in chunks.

In this paper, we will only consider uncompressed WAV files.

WAV sound files in Windows are stored using 8 or 16 bits. A mono wav file, sampled at 44,100 Hz at 16 bits, for example, indicates a file that has generated a string of 16 bits every 1/44100th of a second. In the case of a stereo wav, the obtained 16-bit strings are two. LPCM is also the standard audio coding format for audio CDs, storing two-channel LPCM audio sampled at 44,100 Hz with 16 bits per sample.

Since the LPCM format is uncompressed and preserves all samples of an audio track, professional users or audio experts can use the WAV format with LPCM audio to achieve the highest audio quality. WAV files can also be edited and manipulated with relative easiness via software. Often audio files are stored in compressed form and the most used form in this is MP3 encoding.

Audio steganography on MP3 files is challenging due to the compression involved: the secret information must be preserved in a way that it is not damaged by the compression process.

An MP3 file consists of several MP3 frames which in turn consist of a header (header frame) and a data block (data frame). Each frame contains a 4-byte header.

When audio frames are used for data hiding it is necessary to select the appropriate frames to hide the secret information so that secret information is not lost during compression.

Each frame has a header frame that includes information such as audio version ID, protection bits, padding bits, channel mode, emphasis, etc. There are some bits in the header frame that are rarely used, for example, the Private bit (pos. 23), the Copyright bit (pos. 28), the Original bit (pos. 29).

5.1. Least Significant Bit (LSB) Modification

LSB is an audio data hiding technique in which the last least significant bit of each sample of the audio file is replaced with a data bit and it is generally used to detect possible modifications on a transmitted audio, this is because the LSB watermarking algorithm is fragile.

This method is efficient, but at the same time it is vulnerable to steganalysis, and it is weak against data compression.

It is possible to improve the implementation of the LSB technique using ameliorative methods, as in [15].

The first method (*LSB bit selection*) is based on randomizing the selection of the bit to be replaced with the data bit in the cover file. In this type of bit selection algorithm, to confuse the intruder, the same bit of a sample is never used.

Randomness is produced by selecting a different bit of each sample to hide the secret message.

The first two most significant bits will decide which bit of the same sample will contain the secret message bit. Thus, the secret message bit must always be embedded in the first three least significant bits of a sample.

Furthermore, before injecting the secret text into the cover file, it is encrypted with the AES-256 algorithm in such a way as to increase the level of security compared to those who perform active steganalysis on the cover file.

The second method (*LSB sample selection*) is based on randomizing the choice of the sample to be used to inject the data bit: to confuse the intruder, more randomness is added in incorporating the secret information by selecting a random number of samples.

This means that not all audio samples will contain information, only some.

These samples are chosen by the value of the top three most significant bits (MSB) of the sample.

In this case, regardless of the value of the first three most significant bits, the last least significant bit of the chosen sample will always be replaced.

The main steps of the algorithm are the following:

1. Encrypt the secret message with the AES-256 algorithm.
2. Select the bits in the cover text that must be replaced by the bits of the secret message. Do this by using LSB bit selection and LSB sample selection.
3. Replace the selected bits in the cover text with the bits of the secret message.

We have experimentally tested the LSB approach.

As regards the bit selection technique, to find the right audibility threshold (so as not to have a substantial difference in perception between the original audio and the modified audio), various tests were performed.

At each test, the position of the bit to be replaced within the sample was fixed, to find the right compromise in choosing the least significant bits to be replaced.

In the end, analyzing the results, it was seen that by replacing other bits instead of the three least significant bits, there are unpleasant outputs as the difference with respect to the original audio file is perceptibly audible.

Thus, the maximum position of the least significant bit that has come to be replaced in the third.

As regards the sample selection technique, the results obtained were excellent, as for the bit selection technique, at the expense of having a reduction in the number of possible bits to be injected into the file cover.

The following three figures (Figures 9–11) picture the original audio and the results obtained by LSB bit selection and LSB sample selection: they are almost identical.

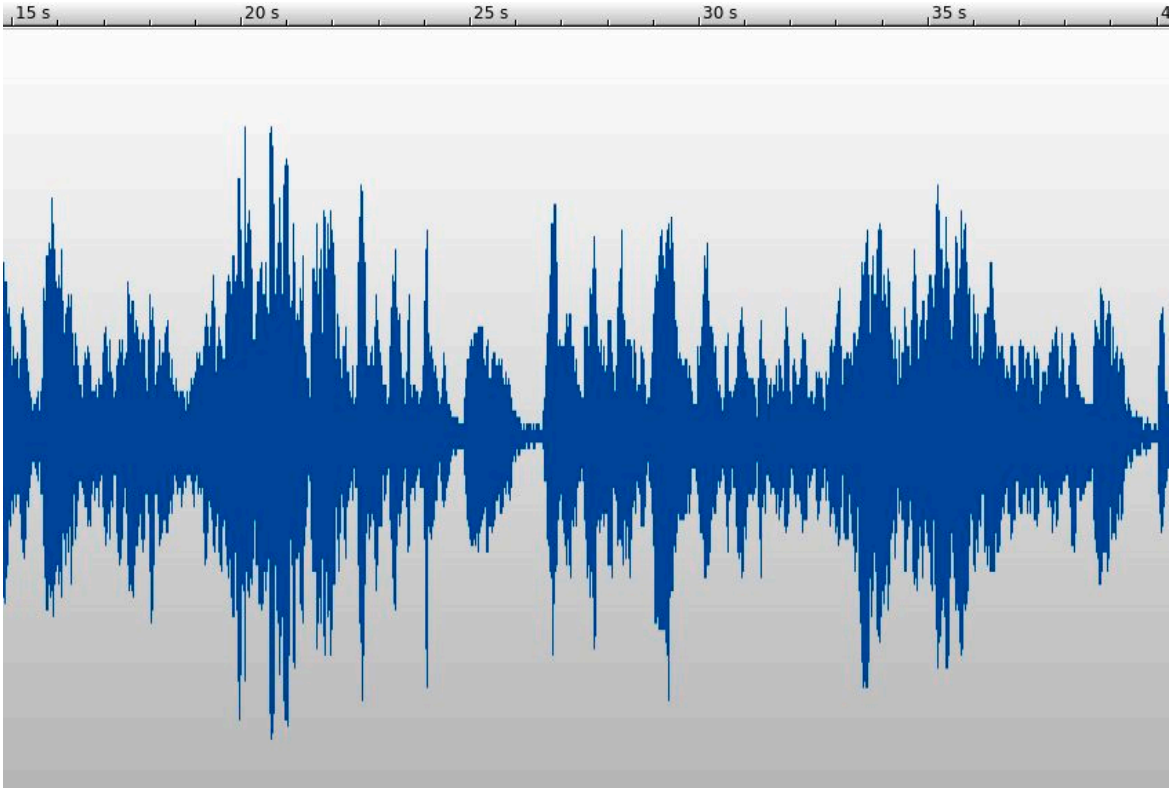

Figure 9. Original audio, interval between 15 and 40 s.

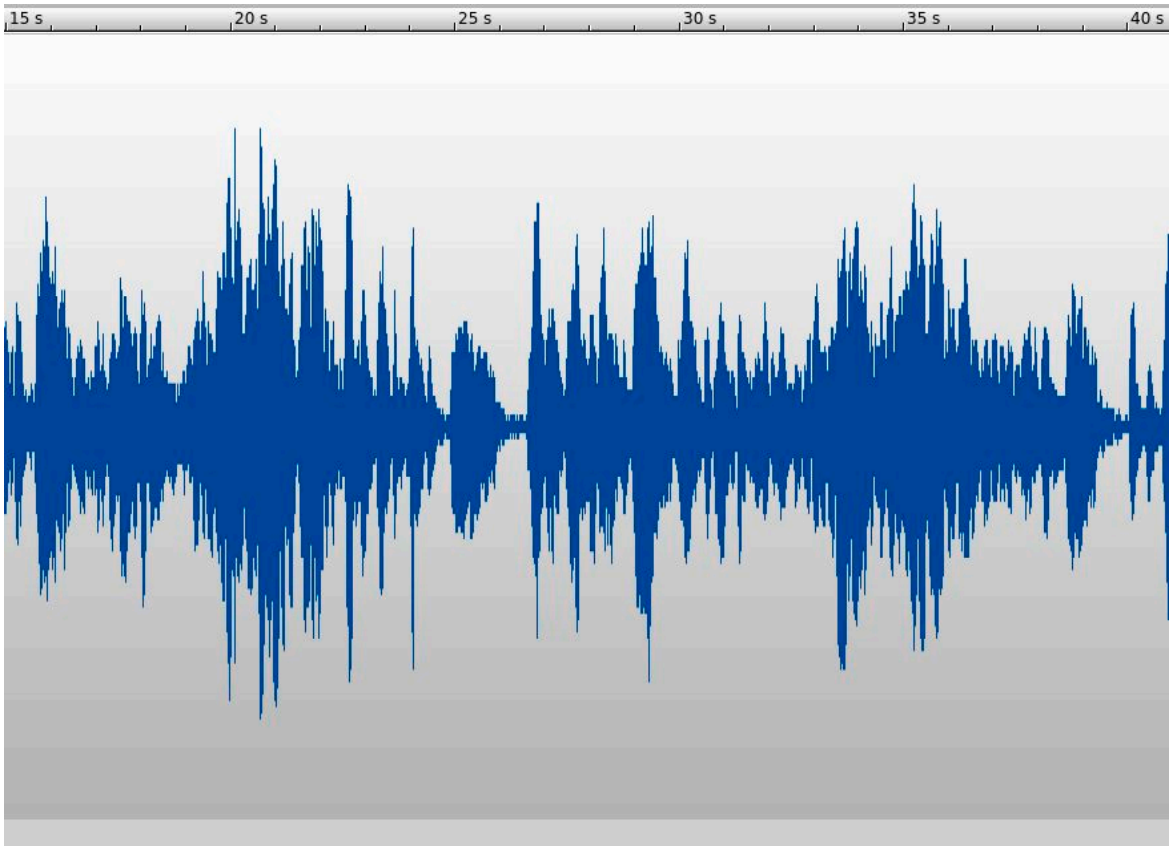

Figure 10. LSB bit selection, interval between 15 and 40 s.

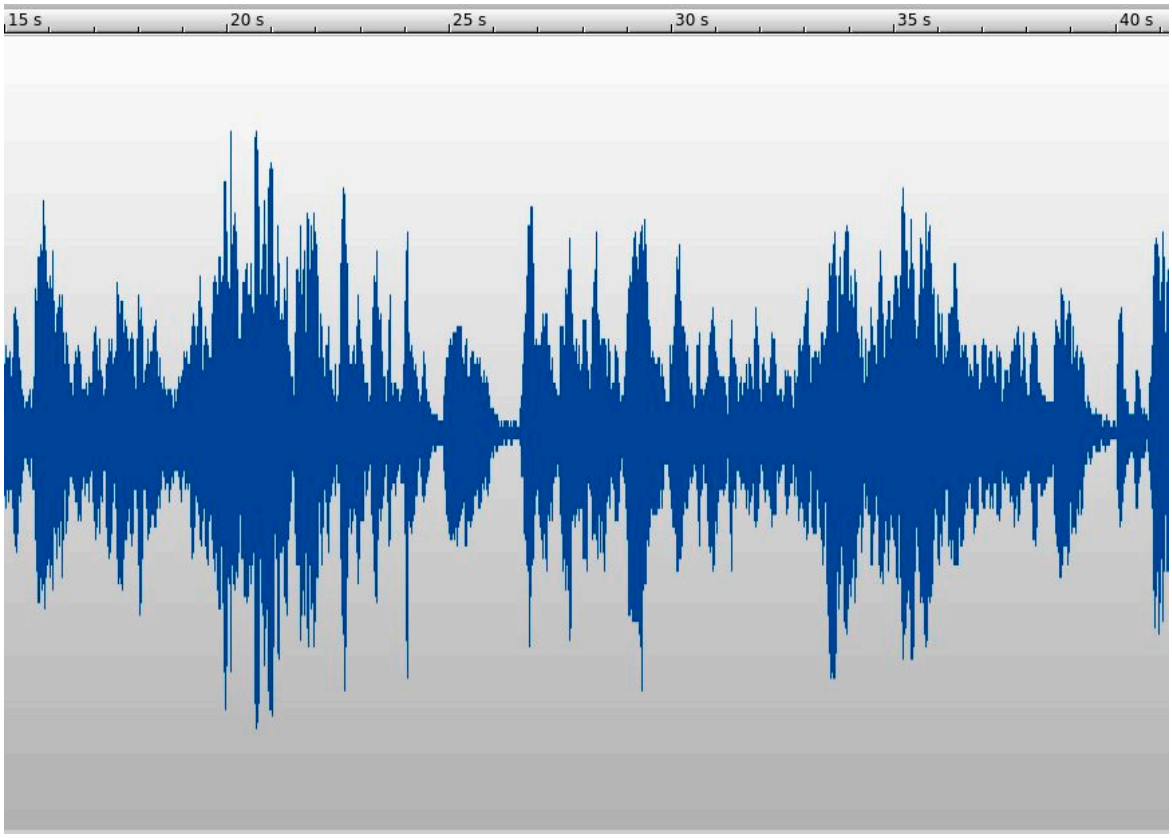

Figure 11. LSB sample selection, interval between 15 and 40 s.

5.2. Header Bit in MP3

The LSB technique is not robust against changes in the sample values caused by data compression on the audio file and therefore cannot be applied to compressed audio formats such as MP3.

The technique described below overcomes this problem and it is inspired by [16].

Basically, the implementation of the algorithm is based on the concept of incorporating the secret message after the compression phase.

In particular, the various bits of the secret text are incorporated in one of three specific fields of the mp3 header frame, namely:

- Private bit (pos. 23)
- Copyright bit (pos. 28)
- Original bit (pos. 29)

(Each header frame is 32 bits long.)

The choice of which field to use (among the above-mentioned) for each frame is made based on the value of the two most significant bits of the last byte (sample) of the data contained in that frame.

Before injecting the bits of the secret message into the frames, the secret text is first encrypted and then each bit is inverted.

The main steps of the algorithm are the following:

1. Encrypt the secret message with the AES-256 algorithm.
2. For all the bits in the secret message:
   - 2.1. Select the specific field of the next mp3 header frame (Private bit, Copyright bit, Original bit) in which to inject the next secret message bit.
   - 2.2. Invert the next secret message bit and inject it in the right position.

The decision to use only one bit of the three available in the header frame for each frame (and to apply the inverse bit) was made to introduce confusion and diffusion on the part of those who perform active steganalysis. This, of course, involves a reduction in the number of available bits that can be used for the injection of the secret text. The decoding algorithm can correctly reconstruct the initial message from the cover audio file.

Since the value of the data frames (and therefore the audio samples) is not altered, but only some information fields of the header frame, therefore no alteration of the sound is detected during listening.

6. Conclusions

In this paper, we have explored a unified approach to compression and security.

We have discussed, implemented, and experimentally tested new ideas connected to the compression and privacy of one-dimensional data (security of interactive data compression), two-dimensional data (JPEG image scrambling), three-dimensional data (hyperspectral images), and digital sound.

Future research includes extensive experimentation of the ideas presented here both for the secure lossless compression of images (see, for example, [17,18]) and new approaches to the combination of lossy compression and security for holograms, and other three-dimensional data, i.e., digital video or three-dimensional medical images (see [17,18]) and for multidimensional data in general.

Author Contributions: All authors have contributed equally to: conceptualization; methodology; software; validation; formal analysis; investigation; resources; data curation; writing—original draft preparation; writing—review and editing; visualization; supervision; project administration; funding acquisition. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially supported by project SERICS (PE00000014) under the NRRP MUR program funded by the EU—NGEU.

Acknowledgments: The first author would like to thank his students: F. Rizzo, G. delle Donne, L. Desiato, G. Fiorentino, E. Massaro, C. De Blasio, M. Nilo, P. Marrone, P. Vigorito, for conducting preliminary experiments on some of the material covered in this paper.

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

References

1. Pizzolante, R.; Carpentieri, B. Copyright protection for images on mobile devices. In Proceedings of the 6th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), Palermo, Italy, 4–6 July 2012; IEEE Computer Society Press: Washington, DC, USA, 2012; pp. 585–590.
2. Carpentieri, B. Efficient Compression and encryption for digital data transmission. *Secur. Commun. Netw.* **2018**, *2018*, 9591768. [CrossRef]
3. Kelley, J.; Tamassia, R. Secure Compression: Theory & Practice. Available online: https://eprint.iacr.org/2014/113.pdf (accessed on 1 December 2022).
4. Oğuzhan Külekci, M. On scrambling the Burrows-Wheeler transform to provide privacy in lossless compression. *Comput. Secur.* **2012**, *31*, 26–32. [CrossRef]
5. Rizzo, F.; Storer, J.A.; Carpentieri, B. Overlap and channel errors in Adaptive Vector Quantization for image coding. *Inf. Sci.* **2005**, *171*, 125–143. [CrossRef]
6. El Gamal, A.; Orlitsky, A. Interactive data compression. In Proceedings of the 25th Ann. Symp. Foundations Computer Science (FOCS), Singer Island, FL, USA, 24–26 October 1984; pp. 100–108.
7. Carpentieri, B. Sending compressed messages to a learned receiver on a bidirectional line. *Inf. Process. Lett.* **2002**, *83*, 63–70. [CrossRef]
8. Carpentieri, B. Interactive Compression of Digital Data. *Algorithms* **2010**, *3*, 63–75. [CrossRef]
9. Carpentieri, B.; Palmieri, F. Efficient and secure transmission of digital data in the 5G era. In Proceedings of the 1st International Conference on eXtended Reality (XR SALENTO), Lecce, Italy, 6–8 July 2022.
10. The Images of Groups Dataset. Available online: http://chenlab.ece.cornell.edu/people/Andy/ImagesOfGroups.html (accessed on 1 December 2022).
11. Face Detection & Recognition. Available online: https://facedetection.com/datasets/ (accessed on 1 December 2022).
12. Korshunov, P.; Ebrahimi, T. Scrambling-based tool for secure protection of JPEG images. In Proceedings of the IEEE International Conference on Image Processing (ICIP), Paris, France, 27–30 October 2014.

13. Pizzolante, R.; Carpentieri, B. Lossless, low-complexity, compression of three-dimensional volumetric medical images via linear prediction. In Proceedings of the 18th International Conference on Digital Signal Processing (DSP), Fira, Greece, 1–3 July 2013; pp. 1–6.
14. Carpentieri, B. Compression and Privacy of Hyperspectral images. In Proceedings of the 2022 International Conference on Electrical, Computer, Communication and Mechatronics Engineering (ICECCME 2022), Maldives, Maldives, 16–18 November 2022.
15. Pizzolante, R.; Carpentieri, B. Visualization, band ordering and compression of hyperspectral images. *Algorithms* **2012**, *5*, 76–97. [CrossRef]
16. Asad, M.; Gilani, J.; Khalid, A. An enhanced least significant bit modification technique for audio steganography. In Proceedings of the International Conference on Computer Networks and Information Technology, Abbottabad, Pakistan, 11–13 July 2011; pp. 143–147. [CrossRef]
17. Rizzo, F.; Storer, J.A.; Carpentieri, B. LZ-based image compression. *Inf. Sci.* **2001**, *135*, 107–122. [CrossRef]
18. Motta, G.; Storer, J.A.; Carpentieri, B. Adaptive linear prediction lossless image coding. In Proceedings of the 1999 Data Compression Conference (DCC), Snowbird, UT, USA, 29–31 March 1999; pp. 491–500.