Hindawi
Security and Communication Networks
Volume 2018, Article ID 9591768, 9 pages
https://doi.org/10.1155/2018/9591768

Research Article

Efficient Compression and Encryption for Digital Data Transmission

Bruno Carpentieri

Dipartimento di Informatica, Università di Salerno, 84084 Fisciano, Italy

Correspondence should be addressed to Bruno Carpentieri; bc@dia.unisa.it

Received 28 February 2018; Accepted 10 April 2018; Published 17 May 2018

Academic Editor: Kim-Kwang Raymond Choo

We live in a digital era in which communication is largely based on the exchange of digital information on data networks. Communication is often pictured as a sender that transmits a digital file to a receiver. This file travels from a source to a destination and, to have a quick and immediate communication, we need an encoding strategy that should be efficient and easy yet secure. This communication could be based on a layout articulated in two operations that are heterogeneous and in some case conflicting but that are needed to be applied to the original file to have efficiency and security. These two operations are data compression and encryption. The aim of this work is to study the combination of compression and encryption techniques in digital documents. In this paper we will test the combinations of some of the state-of-the-art compression and cryptography techniques in various kinds of digital data.

1. Introduction

We live in a digital era in which the way we communicate has dramatically changed. We can communicate digital files at any time, on any device, and with anyone on the planet.

Data compression has been one branch of computer science that made this digital revolution possible. By compressing the input data, for example, *n to 1* (the compressed file size is 1 *n*th of original file size) it is possible to send the same information n times faster or to send the file at once on a transmission channel that has $1/n$ capacity or even to send n files in parallel on a channel that has capacity n.

In this digital revolution communication has a price: accidentally we have to consent that the digital message we are sending will be potentially intercepted and read on its way to the destination.

Cryptography might be a solution to this issue: if the sender encrypts the message, assuming that only the destination has a way to decrypt it, then privacy will be insured.

Therefore digital communication should be based on a layout articulated in two operations that are heterogeneous and in some case conflicting but that are needed to be applied to the original file to have efficiency and security. These two operations are data compression and encryption.

The aim of this work is to study the combination of compression and encryption techniques on digital documents. In this paper we test the state-of-the-art compression and cryptography techniques on various kinds of digital data.

The next section shall present an introduction to the most commonly used methods in data compression and cryptography, together with a short survey of past work.

Sections 3, 4, and 5 will show the experimental results obtained on one-dimensional, two-dimensional, and three-dimensional data.

Section 6 contains our conclusions and future research directions.

2. Data Compression and Cryptography

Today the way we communicate has dramatically changed. We communicate digitally and we aim to have an efficient and secure communication. The research in this field is devoted to improving the way we communicate so as to have stronger requirements of efficiency and security, where efficiency is given by data compression and security by encryption.

Data compression is today essential for digital communication. Without data compression we would not have

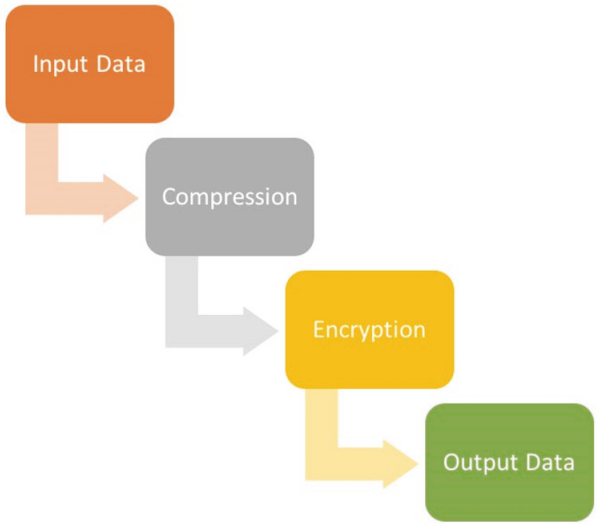

Figure 1: Encryption after compression.

digital televisions, smart-phones, satellite communications, Internet, etc.

Information theory tells us that there is a limit to the amount of compression we can gain from a digital file and that this limit is the Shannon entropy [1]: the highest the entropy is, the lowest the possibility of compressing data shall be.

Therefore the enemy of compression is randomness. But, on the other side, encryption needs to bring randomness into the digital data to bring security. This is why, when we have to perform both compression and encryption, we will always compress first and then encrypt, as shown in the workflow of Figure 1.

There are therefore two interesting questions to be posed: What is the cost of encryption, in terms of file size, after performing compression? How bad is performing first encryption and then compression?

Here we give, experimentally, an answer to both these questions.

2.1. Data Compression. Data Compression can be defined as the coding of data to minimize its representation. The compression process is called lossless if the original one can be exactly reconstructed from the compressed copy; otherwise it is called lossy.

The theoretical background of the data compression techniques is strong and well established.

It dates back to the seminal work of Shannon who, more than half a century ago, gave precise limits on the performance of any lossless compression algorithm: this limit is the entropy of the source we want to compress.

Data compression techniques are specifically dependent on the type of data that has to be compressed and on the desired performance.

In this paper we will experiment with largely used compression methods such as Run Length Encoding, Huffman Coding, Arithmetic Coding, LZW, JPEG, and JPEG2000.

Run Length Encoding (RLE) [1] is one of the simplest compression algorithms in which any repeated *run* of characters is coded by using only two elements: the first is used as a counter: it memorizes how long the string is; the second contains the repetitive element that constitutes the string. If a data item d occurs at n consecutive times in the input stream, RLE replaces the n occurrences with the single pair nd.

Huffman Coding has been introduced in 1952 by Huffman [2]. The output from Huffman's algorithm is a variable-length code where more common symbols are generally represented using fewer bits than less common symbols.

Arithmetic Coding [3] is a form of entropy encoding that encodes a message into a single number: an arbitrary-precision fraction q where $0.0 \leq q < 1.0$.

In Huffman Coding there is a limit: each source character has to be coded with at least one bit. This limitation does not apply to Arithmetic Coding.

Textual substitution methods, often called dictionary methods or Lempel-Ziv methods after the important papers by Ziv and Lempel [4, 5], maintain a constantly changing dictionary of strings to adaptively compress a stream of characters by replacing common substrings with indices into the dictionary. Lempel and Ziv proved that these schemes were practical as well as asymptotically optimal for a general source model.

In 1984 Welch published [6] in which he described a practical implementation of the method outlined in [5]. He called this work LZW and LZW is the compression method used in the UNIX COMPRESS program and in earlier versions of the MS-DOS ARC program.

Digital images can be defined as a set of two-dimensional arrays of integer data (the samples), represented with a given number of bits per component.

Because of the many applications of digital images there have been various processes of standardization involving image compression.

Perhaps the widest diffused standard is JPEG (that is an acronym for Joint Photographic Experts Group, the unofficial name of the Standardization Committee that promoted the standard—see [7]).

JPEG has four modes of operation. One of them is Lossless JPEG that is the lossless version of this standard. Today Lossless JPEG is far from the state of the art of lossless image compression but it is used in this paper to show the performance of a simple image compression method when coupled with encryption.

We generally refer to JPEG as the baseline lossy method of the compression standard.

Today JPEG compression is still very popular and widespread.

JPEG-2000 [8] is a recent standard for image compression developed by the Joint Photographic Experts Group for still image coding.

The standard is intended to offer unprecedented access into the image while still in compressed domain. Thus, images can be accessed, manipulated, edited, transmitted, and stored in a compressed form. The lossless mode of the standard is very close to the actual state of the art in lossless image compression.

Video compression, since the beginning of the 1980s, has been an attractive research area because a digital video may provide more information than a single image frame. The huge computational complexity and memory space required for video processing are now more attainable, due to the more advanced, achievable computational capability today we have available.

MPEG4 [9] is a standard for the encoding of digital audio and video developed by the ISO/IEC Moving Picture Experts Group (MPEG). MPEG-4 is used primarily for applications such as video telephony and digital television, for the transmission of movies via the web and for storage.

Current research in data compression focuses also on the compression of other three-dimensional data, i.e., hyperspectral images or medical images (see, for example, [10, 11]).

In this paper we have used the implementation of RLE, Huffman Coding, and Arithmetic Coding described in [12].

For JPEG and JPG2000 we used the open source libraries LibJpegTurbo [13] and OpenJpeg [14]. For MPEG4 we tested an on-line codec.

2.2. Encryption. Cryptography today is crucial in many aspects of today's world: from Internet banking and ecommerce to email and web-based business processes.

A common way to encrypt a message is in blocks. Naturally such systems are known as block ciphers. The most important and still most widely used block cipher is the Data Encryption Standard or DES.

Triple DES, often indicated as 3DES, was developed by W. Tuchman. It is a block cipher that builds on the success of DES, while increasing the key length so that brute force attacks become almost impossible. It uses DES three times.

In 1997 NIST started the search for a successor to DES called Advanced Encryption Standard (AES). This would be an unclassified, public encryption scheme. In 2000 the scheme *Rijndael*, named after its Belgian inventors Joan Daemen and Vincent Rijmen of the COSIC Laboratory, was selected to be the new AES.

For a complete, comparative analysis on the above symmetric encryption block cipher algorithms see [15].

RC4 (Rivest Cipher 4) is a stream cipher. In the past it was widespread for its simplicity and speed in software, but after that its multiple vulnerabilities have been discovered and it is not considered secure any more [16]. It is still largely used in noncritic applications.

The code we experimented for encryption is taken from the open source library OpenSSL [17].

2.3. Other Approaches to Compression and Security. The need for a secure and efficient data transmission has had an impact also on the research in other applications of compression and security.

The delicate balance between compression and cryptography might be framed in a slightly wider context, and it is possible to consider other points of view from which this deep and intriguing question has been studied; see, for example, [18, 19].

In this digital era information owners are reluctant to distribute their data in an insecure environment. In fact they fear that digital products such as music files, videos, and images could be copied by anyone who has access to the network.

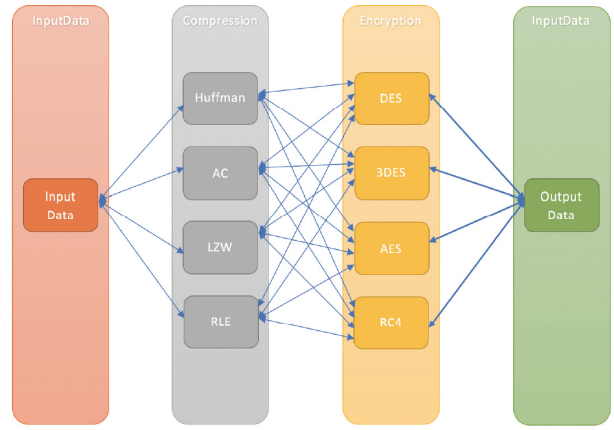

Figure 2: One-dimensional digital data: workflow.

This has led to the idea of protecting the products by using a watermark that could be visible or invisible.

Watermarking can be coupled to data compression to have a safer transmission of a digital object. This combination of compression and steganography has been tested in various domains; for example, see [20–22].

There are other papers in literature that study the combination of compression and encryption techniques, generally by considering only one-dimensional data or specific compression algorithms; see, for example, [23–26].

3. Text and One-Dimensional Digital Data

One-dimensional data, for example, textual data, programs, and object codes, are generally lossless coded because the end user, a human in the case of textual data or a computer in the case of object codes, would not accept a single bit error in the decoded message.

The test data set on which we have experimented includes 17 files from the well-known Calgary Corpus [27]. The Calgary Corpus is a collection of text and binary files created in 1987 by Ian Witten, Tim Bell, and John Clearly. It is named after the University of Calgary in which the three creators worked.

We have experimented on one-dimensional data with four standard compression algorithms, Huffman Coding, Arithmetic Coding, Lempel-Ziv-Welch Coding, and Run Length Encoding, and with four standard encryption algorithms, DES, 3DES, AES, and RC4.

Figure 2 shows the workflow of the tests.

In a first set of tests the input files have been first compressed and then encrypted; then in a second round of testing the same input files have been first encrypted and then compressed.

Tables 1, 2, 3, and 4 show the experimental results obtained in first compressing and then encrypting the files.

The first column shows the file name, the second column shows the file size in bytes, the third column shows the file

Table 1: Compression with Run Length Encoding and then encryption.

FILE	PLAIN	RL	RL + DES	RL + 3DES	RL + AES	RL + RC4
bib	111261	111261	111264	111264	111264	111261
book 1	768771	768645	768648	768648	768656	768645
book 2	610856	610561	610568	610568	610576	610561
geo	102400	100804	100808	100808	100816	100804
news	377109	366661	366664	366664	366672	366661
obj2	246814	246939	246944	246944	246944	246939
paper 1	53161	52992	53000	53000	53008	52992
paper 2	82199	82190	82192	82192	82192	82190
paper 3	46526	46526	46528	46528	46528	46526
paper 4	13286	13286	13288	13288	13296	13286
paper 5	11954	11950	11952	11952	11952	11950
paper 6	38105	38071	38072	38072	38080	38071
pic	513216	105881	105888	105888	105888	105881
progc	39611	38892	38896	38896	38896	38892
progl	71646	66612	66616	66616	66624	66612
progp	49379	45225	45232	45232	45232	45225
trans	93695	89739	89744	89744	89744	89739

Table 2: Compression with Huffman Coding and then encryption.

FILE	PLAIN	HUFF	HUFF + DES	HUFF + 3DES	HUFF + AES	HUFF + RC4
bib	111261	72933	72936	72936	72944	72933
book 1	768771	440041	440048	440048	440048	440041
book 2	610856	369216	369224	369224	369232	369216
geo	102400	73394	73400	73400	73408	73394
news	377109	246793	246800	246800	246800	246793
obj2	246814	195180	195184	195184	195184	195180
paper 1	53161	33491	33496	33496	33504	33491
paper 2	82199	47833	47840	47840	47840	47833
paper 3	46526	27415	27416	27416	27424	27415
paper 4	13286	7966	7968	7968	7968	7966
paper 5	11954	7549	7552	7552	7552	7549
paper 6	38105	24165	24168	24168	24176	24165
pic	513216	122039	122040	122040	122048	122039
progc	39611	26042	26048	26048	26048	26042
progl	71646	43217	43224	43224	43232	43217
progp	49379	30456	30464	30464	30464	30456
trans	93695	65414	65416	65416	65424	65414

size after compression, and the remaining columns show the file size in bytes after first compression and then encryption.

The experiments summarized in the four tables show that the cost of encryption after compression is negligible for this kind of data.

Table 5 sums up the results. Its first column is the compression method examined, its second column is the test set size in bytes, the third column is the size in bytes of compressing the test set with that particular compression method, and the columns from 4 to 7 include the size in bytes of compressing and then encrypting with a specific encryption method and also the average cost of encryption in terms of percentage of the original file.

The cost of encryption is almost zero. For example, the table shows that this cost, as expected, is zero for RC4 and between 0,0021% and 0,0053% of the original file size for all the other methods.

As shown in Figure 1 we have also tested the opposite approach in which we first encrypt the file and only after that we compress the encrypted file.

Table 6 is the analogue of Table 5 for this opposite approach. Its rows represent the encryption method used, and the columns from 4 to 7 represent the encryption plus compression process, showing for each compression method the size in bytes obtained after compressing the encrypted files.

Table 3: Compression with Arithmetic Coding and then encryption.

FILE	PLAIN	AC	AC + DES	AC + 3DES	AC + AES	AC + RC4
bib	111261	33562	33568	33568	33568	33562
book 1	768771	262417	262424	262424	262432	262417
book 2	610856	223451	223456	223456	223456	223451
geo	102400	117020	117024	117024	117024	117020
news	377109	147348	147352	147352	147360	147348
obj2	246814	179492	179496	179496	179504	179492
paper 1	53161	19857	19864	19864	19872	19857
paper 2	82199	29137	29144	29144	29152	29137
paper 3	46526	18356	18360	18360	18368	18356
paper 4	13286	6015	6016	6016	6016	6015
paper 5	11954	5957	5960	5960	5968	5957
paper 6	38105	14694	14696	14696	14704	14694
pic	513216	63337	63344	63344	63344	63337
progc	39611	15291	15296	15296	15296	15291
progl	71646	19105	19112	19112	19120	19105
progp	49379	13728	13736	13736	13744	13728
trans	93695	47142	47144	47144	47152	47142

Table 4: Compression with LZW and then encryption.

FILE	PLAIN	LZW	LZW + DES	LZW + 3DES	LZW + AES	LZW + RC4
bib	111261	53844	53848	53848	53856	53844
book 1	768771	390810	390816	390816	390816	390810
book 2	610856	346530	346536	346536	346544	346530
geo	102400	78755	78760	78760	78768	78755
news	377109	232811	232816	232816	232816	232811
obj2	246814	302543	302544	302544	302544	302543
paper 1	53161	31182	31184	31184	31184	31182
paper 2	82199	41666	41672	41672	41680	41666
paper 3	46526	23918	23920	23920	23920	23918
paper 4	13286	7442	7448	7448	7456	7442
paper 5	11954	7020	7024	7024	7024	7020
paper 6	38105	23303	23304	23304	23312	23303
pic	513216	70227	70232	70232	70240	70227
progc	39611	24467	24472	24472	24480	24467
progl	71646	34916	34920	34920	34928	34916
progp	49379	23288	23296	23296	23296	23288
trans	93695	50544	50552	50552	50560	50544

Table 5: Cost of encryption for one-dimensional data (compression + encryption).

Compression method	Test set size	Compr. Size	Compr. + DES	Compr. + 3DES	Compr. + AES	Compr. + RC4
RLE	3229989	2796235	2796304 (+0,0021%)	2796304 (+0,0021%)	2796368 (+0,0041%)	2796235 (+0%)
Huffman	3229989	1833144	1833224 (+0,0025%)	1833224 (+0,0025%)	1833296 (+0,0047%)	1833144 (+0%)
Arithmetic	3229989	1215909	1215992 (+0,0026%)	1215992 (+0,0026%)	1216080 (+0,0053%)	1215909 (+0%)
LZW	3229989	1743266	1743344 (+0,0024%)	1743344 (+0,0024%)	1743424 (+0,0049%)	1743266 (+0%)

Table 6: Cost of encryption for one-dimensional data (encryption + compression).

Encryption method	Test set size	Encrypt. Size	Encrypt. + Huffman	Encrypt. + AC	Encrypt. + LZW	Encrypt. + RLE
RC4	3229989	3229989	3235861 (+5872 b.)	6376898 (+3146909 b.)	4587226 (+1357237 b.)	3242600 (+12611 b.)
DES	3229989	3230064	3235939 (+5950 b.)	6370696 (+3140707 b.)	4587115 (+1357126 b.)	3242550 (+12561 b.)
3DES	3229989	3230064	3235932 (+5943 b.)	6373622 (+3143633 b.)	4588580 (+1358591 b.)	3242570 (+12581 b.)
AES	3229989	3230128	3235985 (+5996 b.)	6370180 (+3140191 b.)	4586952 (+1356963 b.)	3242619 (+12630 b.)

Table 7: Compression with JPEG and then encryption.

FILE	PLAIN	JPEG	JPEG + DES	JPEG + 3DES	JPEG + AES	JPEG + RC4
TsukubaR.ppm	331791	20028	20032	20032	20032	20028
Zlatuost.ppm	417645	30434	30440	30440	30448	30434
couple.ppm	1665645	56958	56960	56960	56960	56958
emir.ppm	1766859	99184	99192	99192	99200	99184
etud.ppm	2801710	203650	203656	203656	203664	203650
model.ppm	1724418	54992	55000	55000	55008	54992

In this case it is clear that compressing after an encryption does not help; instead it even increases both the encrypted and the original file sizes. We have reported for each compression method the amount, in bytes, of the increase in size with respect to the original file.

Compression after encryption does not work; for example, the randomness induced by the encryption algorithms brings Arithmetic Coding to almost double the original file size.

4. Images and Two-Dimensional Digital Data

Two-dimensional data, i.e., digital images, are generally lossy coded because the human end user is not capable of noticing small pixel errors.

Lossless image compression is required in applications where the pictures are subject to further processing, intensive editing, or repeated compression/decompression.

It is generally the choice also for images obtained at great cost or in applications where the desired quality of the rendered image is yet unknown. Thus, precious art works to be preserved, medical imaging, remotely sensed images, prepress industry, and image archival systems are generally losslessly compressed.

The test data set on which we have experimented includes 6 images from [28].

We have experimented on two-dimensional data with three standard compression algorithms, JPEG, Lossless JPEG, and JPEG 2000 in lossless mode, and with four standard encryption algorithms, DES, 3DES, AES, and RC4.

Figure 3 shows the workflow of the tests.

In a first set of tests the input files have been first compressed and then encrypted; then in a second round of testing the same input files have been first encrypted and then compressed.

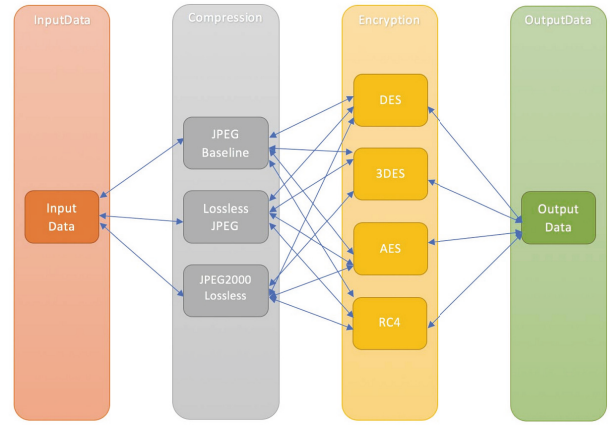

Figure 3: Two-dimensional digital data: workflow.

Tables 7, 8, and 9 show the experimental results obtained in first compressing and then encrypting the files. The first column shows the file name, the second column shows the file size in bytes, the third column shows the file size after compression, and the remaining columns show the file size in bytes after first compression and then encryption.

Lossless JPEG and JPEG 2000 in lossless mode are of course lossless compression algorithms. JPEG, being lossy, has been tested with an "average" quality setting.

Again the tables show that, even for image compression, the cost of encryption after compression is negligible for this kind of data.

Table 10 sums up the results in the case of encryption after compression for digital images.

Its first column is the compression method examined, its second column is the test set size in bytes, the third column is the size in bytes when compressing the test set with that

Table 8: Compression with Lossless JPEG and then encryption.

FILE	PLAIN	LOSSLESS JPEG	LOSSLESS JPEG + DES	LOSSLESS JPEG + 3DES	LOSSLESS JPEG + AES	LOSSLESS JPEG + RC4
TsukubaR.ppm	331791	222547	222552	222552	222560	222547
Zlatuost.ppm	417645	307076	307080	307080	307088	307076
couple.ppm	1665645	890998	891000	891000	891008	890998
emir.ppm	1766859	1209945	1209952	1209952	1209952	1209945
etud.ppm	2801710	2118361	2118368	2118368	2118368	2118361
model.ppm	1724418	906468	906472	906472	906480	906468

Table 9: Compression with JPEG2000 (lossless mode) and then encryption.

FILE	PLAIN	JPEG-2000	JPEG2000 + DES	JPEG2000 + 3DES	JPEG2000 + AES	JPEG2000 + RC4
TsukubaR.ppm	331791	150711	150712	150712	150720	150711
Zlatuost.ppm	417645	208344	208352	208352	208352	208344
couple.ppm	1665645	433583	433584	433584	433584	433583
emir.ppm	1766859	700606	700608	700608	700608	700606
etud.ppm	2801710	1229192	1229200	1229200	1229200	1229192
model.ppm	1724418	503768	503776	503776	503776	503768

particular compression method, and the columns from 4 to 7 include the size in bytes of compressing and then encrypting with a specific encryption method and also the average cost of encryption in terms of percentage of the original file.

For example, the table shows that this encryption cost, as expected, is zero for RC4 and between 0,0003% and 0,0008% of the original file size for all the other methods.

Table 11 is the analogue of Table 10 for this opposite approach. Its rows represent the encryption method used, and the columns from 4 to 7 represent the encryption plus compression process, showing for each compression method the size in bytes obtained after compressing the encrypted files.

Also in this case it is clear that compressing after an encryption does not help; instead it increases the file size with respect to both the encrypted and the original file sizes for lossless compression algorithms. We have reported for each compression method the amount, in bytes, of the increase in size with respect to the original file.

Do not be misguided by column 4: JPEG is a lossy algorithm and it still compresses the encrypted file, with an average compression ratio, for our test data set, of 4.6 for RC4, 4.4 for DES, 4.4 for 3DES, and 4.4 for AES. But this is not good with respect to the 18.7 compression ratio obtained on the test data set by JPEG, with the same settings, before encryption.

In order to obtain the best result in terms of privacy and size, for digital images, it is therefore necessary to sequentially operate compression techniques and, only after compression, encryption techniques.

In fact to operate these techniques in reverse order, although possible, does not guarantee the best results in terms of compression ratio, and it also implies the need to cope with problems relating to the encryption. The application of an encryption algorithm on a digital image implies the encryption of all the bytes of the image, including the bytes defining its header. Therefore a compression algorithm designed for images, such as JPEG, JPEG2000, or Lossless JPEG, would not recognize the data as a digital image and it would not have enough elements to understand how many rows and columns should have the image.

To avoid this problem in our tests we have implemented a simple algorithm that is able to reconstruct, starting from the encrypted file, the original image header that will be used by the compression algorithms to recognize the data as an image.

5. Video and Three-Dimensional Digital Data

Digital video is an example of three-dimensional data. It is generally lossy coded because the target end user is human and because the size of digital video files is such that lossy video compression is essential for transmission or storage purposes.

The test data set on which we have experimented includes 5 small synthetic videos in the.avi format.

We have experimented on these 5 files the effects of the MPEG4 compression standard, paired with DES and RC4.

In this set of tests the input files have been first compressed and then encrypted.

Table 12 shows the experimental results obtained in first compressing and then encrypting the files. The first column shows the file name, the second column shows the file size in bytes, the third column shows the file size after compression, and the remaining columns show the file size in bytes after first compression and then encryption.

Table 13 sums up the results in the case of encryption after MPEG4 compression for digital videos.

Its first column is the compression method examined, MPEG4, its second column shows the test set size in bytes, the third column is the size in bytes when compressing the test set with that particular compression method, and columns from 4 to 5 include the size in bytes of compressing

Table 10: Cost of encryption for two-dimensional data (compression + encryption).

Compression method	Test set size	Compr. Size	Compr. + DES	Compr. + 3DES	Compr. + AES	Compr. + RC4
JPEG	8708068	465246	465280 (+0,0004%)	465280 (+0,0004%)	465312 (+0,0008%)	465246 (+0%)
Lossless JPEG	8708068	5655395	5655424 (+0,0003%)	5655424 (+0,0003%)	5655456 (+0,0007%)	5655395 (+0%)
JPEG2000 (lossless mode)	8708068	3226204	3226232 (+0,0003%)	3226232 (+0,0003%)	3226240 (+0,0004%)	3226204 (+0%)

Table 11: Cost of encryption for two-dimensional data (encryption + compression).

Encryption method	Test set size	Encrypt. Size	Encrypt. + JPEG	Encrypt. + Lossless JPEG	Encrypt. + JPEG2000 (lossless mode)
RC4	8708068	8708068	1894281 (−6813787 b.)	12031804 (+3323736 b.)	9532748 (+824680 b.)
DES	8708068	8708088	1983720 (−6724348 b.)	12036327 (+3328259 b.)	9529660 (+821592 b.)
3DES	8708068	8708088	1983558 (−6724510 b.)	12033974 (+3325906 b.)	9529591 (+821523 b.)
AES	8708068	8708096	1978988 (−6729080 b.)	12034622 (+3326554 b.)	9528145 (+820077 b.)

Table 12: Compression with MPEG4 and then encryption.

File	PLAIN	MPEG4	MPEG4 + DES	MPEG4 + RC4
airhorse.avi	7375360	1393594	1393600	1393594
cookie.avi	5347840	1204348	1204352	1204348
coralastra2.avi	589312	169162	169168	169162
dfs.avi	1083904	376182	376184	376182
fractogene.avi	1743872	792581	792584	792581

Table 13: Cost of encryption for three-dimensional data (compression + encryption).

Compression method	Test set size	Compr. Size	MPEG4 + DES	MPEG4 + RC4
MPEG4	16140288	3935867	3935888 (+0,0001%)	3935867 (+0%)

and then encrypting with DES or RC4 and also the average cost of encryption in terms of percentage of the original file.

From Table 13 it is clear, one more time, that the cost of encryption is negligible and that this cost tends to be even more neglectable when the file sizes grow.

6. Conclusions and Future Research

We live in a digital era in which the way we communicate has dramatically changed. We can communicate digital files at any time, on any device, and with anyone on the planet.

We communicate digitally, and we aim to have an efficient and secure communication. We are always improving the way we communicate to have stronger requirements of efficiency and security, where efficiency means data compression and security means encryption.

The digital communication could be based on a layout articulated in two operations that are heterogeneous and in some case conflicting but that are needed to be applied to the original file to have efficiency and security. These two operations are data compression and encryption.

The enemy of compression is randomness, but on the other side encryption needs to bring randomness into the digital data to bring security. This is why, when we have to perform both compression and encryption, we will always compress first the data and then encrypt it.

In this paper we have given an experimental answer to two interesting questions: "What is the cost of encryption in terms of file size after performing compression?" And "how bad is performing first encryption and then compression?"

The answer to the first question is that, happily, the cost of security is negligible if we perform first compression and then encryption. This cost has been examined by testing some of the state-of-the-art compression and encryption algorithms on one-dimensional, two-dimensional, and three-dimensional data.

The answer to the second question is the answer the theory already gave us. It is not efficient at all to do first encryption and then compression. The file size will definitely grow and in some cases the resulting output will be far larger than the original input.

New research is now involving more testing and the development of ad hoc algorithms that combine and exploit data compression and security.

Data Availability

All the data and software we have used in our experiments is available online.

Conflicts of Interest

The author declares that they have no conflicts of interest.

Acknowledgments

The author would like to thank the students Alessandro Sacco, Luigi Amitrano, Paolo Anastasio, and Marco Castaldo for performing preliminary experiments related to this paper.

References

[1] J. A. Storer, *Data Compression: Methods and Theory*, Computer Science Press, Maryland, Md, USA, 1988.

[2] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.

[3] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *Communications of the ACM*, vol. 30, no. 6, pp. 520–540, 1987.

[4] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Transactions on Information Theory*, vol. 23, no. 3, pp. 337–343, 1977.

[5] J. Ziv and A. Lempel, "Compression of individual sequences via variable-rate coding," *Institute of Electrical and Electronics Engineers Transactions on Information Theory*, vol. 24, no. 5, pp. 530–536, 1978.

[6] T. A. Welch, "A Technique for high-performance data compression," *The Computer Journal*, vol. 17, no. 6, pp. 8–19, 1984.

[7] W. B. Pennebaker and J. L. Mitchell, *JPEG Still Image Data Compression Standard*, Van Nostrand Reinhold, 1993.

[8] https://jpeg.org/jpeg2000/.

[9] https://web.archive.org/web/20100428021525/http://mpeg.chiariglione.org:80/standards/mpeg-4/mpeg-4.htm.

[10] R. Pizzolante and B. Carpentieri, "Visualization, band ordering and compression of hyperspectral images," *Algorithms*, vol. 5, no. 1, pp. 76–97, 2012.

[11] R. Pizzolante and B. Carpentieri, "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 '13*, 2013.

[12] M. Nelson and J.-l. Gailly, *The Data Compression Book*, M and T Books, 1995.

[13] http://www.github.com/libjpeg-turbo/libjpeg-turbo.

[14] http://www.github.com/uclouvain/openjpeg.

[15] W. Tuchman, in *A Brief History of the Data Encryption Standard, Internet Besieged: Countering Cyberspace Scofflaws*, pp. 275–280, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1998.

[16] P. Prasithsangaree and P. Krishnamurthy, "Analysis of energy consumption of RC4 and AES algorithms in wireless LANs," in *Proceedings of the IEEE Global Telecommunications Conference GLOBECOM '03*, pp. 1445–1449, usa, December 2003.

[17] http://www.github.com/openssl/openssl.

[18] M. R. Ogiela and L. Ogiela, "On using cognitive models in cryptography," in *Proceedings of the 30th IEEE International Conference on Advanced Information Networking and Applications, AINA '16*, pp. 1055–1058, IEEE, Crans-Montana, Switzerland, March 2016.

[19] U. Fiore, "Selective redundancy removal: a framework for data hiding," *Future Internet*, vol. 2, no. 1, pp. 30–40, 2010.

[20] R. Pizzolante, B. Carpentieri, A. Castiglione, and G. De Maio, "The AVQ algorithm: Watermarking and compression performances," in *Proceedings of the 3rd IEEE International Conference on Intelligent Networking and Collaborative Systems, INCoS '11*, pp. 698–702, December 2011.

[21] P. Albano, A. Bruno, B. Carpentieri et al., "A secure distributed video surveillance system based on portable devices," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface*, vol. 7465, pp. 403–415, 2012.

[22] P. Albano, A. Bruno, B. Carpentieri et al., "Secure and distributed video surveillance via portable devices," *Journal of Ambient Intelligence and Humanized Computing*, vol. 5, no. 2, pp. 205–213, 2014.

[23] R. Sharma and S. Bollavarapu, "Data Security using Compression and Cryptography Techniques," *International Journal of Computer Applications*, vol. 117, no. 14, pp. 15–18, 2015.

[24] E. Setyaningsih and R. Wardoyo, "Review of image compression and encryption techniques," *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 2, 2017.

[25] Y. Zhang, B. Xu, and N. Zhou, "A novel image compression–encryption hybrid algorithm based on the analysis sparse representation," *Optics Communications*, vol. 392, pp. 223–233, 2017.

[26] C. P. Wu and C. C. J. Kuo, "Design of integrated multimedia compression and encryption systems," *IEEE Transactions on Multimedia*, vol. 7, no. 5, pp. 828–839, 2005.

[27] http://corpus.canterbury.ac.nz/descriptions/#calgary.

[28] http://www.csd.uwo.ca/courses/CS4487a/image_samples.html.