A STUDY ON
DATA EMBEDDING IN UNIVERSAL DOMAIN

MUSTAFA S. ABDUL KARIM

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2015

A STUDY ON
DATA EMBEDDING IN UNIVERSAL DOMAIN

MUSTAFA S. ABDUL KARIM

THESIS SUBMITTED IN FULFILMENT
OF THE REQUIREMENTS
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

FACULTY OF COMPUTER SCIENCE
AND INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2015

UNIVERSITI MALAYA

ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: **MUSTAFA S. ABDUL KARIM**

Registration/Matrix No.: **WHA100034**

Name of Degree: **Doctor of Philosophy**

Title of Project Paper/Research Report/Dissertation/Thesis ("this Work"):

A STUDY ON DATA EMBEDDING IN UNIVERSAL DOMAIN

Field of Study: **Information Hiding**

I do solemnly and sincerely declare that:

(1) I am the sole author/writer of this Work;
(2) This work is original;
(3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work;
(4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
(5) I hereby assign all and every rights in the copyright to this Work to the University of Malaya ("UM"), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
(6) I am fully aware that if in the course of making this Work I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidate's Signature Date

Subscribed and solemnly declared before,

Witness's Signature Date

Name:
Designation:

ABSTRACT

Conventionally, data embedding is a feature dependent process, where a feature of the host is modified to insert a payload while satisfying some properties. This dependency limits the interchangeability among data embedding methods. In other words, the applicability of a conventional data embedding method is restricted to certain types of signal. This restriction is observed in the most of the surveyed methods. Hence, a universal data embedding method applicable to any digital signal is nonexistent, albeit such method can potentially be applied in applications where feature extraction is technically challenging. For example, a cloud storage receives various multimedia contents. In addition, some contents are encrypted or compressed, which complicates the feature extraction process for data embedding purposes.

In this study, the conventional data embedding methods are surveyed and evaluated in terms of interchangeability (Chapter 2). The problem of limited interchangeability is overcome by the proposed concept of universal data embedding, which is realized by a novel parser referred to as universal parser (Chapter 3). This parser segments the host signal into partitions of unified length referred to as IC's (Imaginary Codewords). Theoretically, it is shown that the entropy (and hence redundancy) of IC's changes based on the length utilized in the segmentation process. Thus, the defined redundancy is replaced by payload using four proposed methods.

The first method, uREADS (Chapter 4), is based on mapping IC's to GRC (Golomb-Rice Codewords). Then, GRC's are modified to accommodate external information. However, uREADS has inconsistent and low carrier capacity. These problems are overcame by the second method, urDEED (Chapter 5), which applies a similar mapping to GRC's, but with different way to handle the side information. urDEED offers consistent average carrier capacity of 0.169 bpb (bit per bit), and it is universally applicable to

any encrypted signal. However, the mapping of IC's to GRC's involves the sophisticated processes in handling the side information. Also, this method is not applicable to high entropy signal. To overcome this problem, DeRand (Chapter 6) is proposed, which is based on histogram mapping. DeRand achieves a carrier capacity up to 0.4 bpb while being able to control the distortion in high entropy hosts such as random signals.

The conventional concept of data embedding is further generalized to the novel concept of data fusion in (Chapter 7). Here, unlike conventional data embedding, which implies the processing of two signals only, namely, the host and the payload, the proposed data fusion can conceptually fuse two or more signals. A novel DSC (Dual Semantic Code) is proposed as a mean to realize data fusion, where each DSC codeword can accommodate two independent data simultaneously. The proposed data fusion achieves scalable carrier capacity, which can be further traded-off with file-size.

All in all, the discussion in (Chapter 8) shows that the proposed data embedding methods are universal and superior to the conventional methods in terms of interchangeability. In addition, the proposed methods preserve file-size and they are reversible. Also, data fusion and DeRand offer scalable distortion and scalable carrier capacity.

Abstrak

Secara umumnya, data embedding adalah process yang bergantung kepada ciri, di mana ciri-ciri yang dipilih dari hos diubah-suai untuk menyimpan payload sementara memuaskan sifat-sifat yang ditetapkan. Pergantungan ini mengehadkan kesalingbolehtukaran antara kaedah-kaedah data embedding. Dengan lain perkataan, kebolehan kaedah konvensional data embedding dibatas oleh signal yang digunakan. Batasan ini didapati dalam kebanyakan kaedah data embedding yang dikaji. Dalam kesusasteraan, tidak ada kaedah universal data embedding yang boleh digunakan dalam sebarang signal digital. Kaedah universal sedemikian menpunyai potensi untuk digunakan dalam aplikasi yang sukar dalam pengekstrakan ciri, misalnya aplikasi cloud storage yang menyimpan pelbagai jenis kandungan multimedia yang berbeza. Sering kali, kandungan multimedia akan disulitkan atau dimampatkan, menjadikan process pengekstrakan ciri lebih sukar untuk mancapai tujuan data embedding.

Dalam kajian ini, kaedah konvensional data embedding ditinjau and dinilai dari segi kesalingbolehtukaran (Bab 2). Konsep universal data embedding direalisasikan dengan menggunakan novel parser yang turut dirujuk sebagai universal parser (Bab 3), untuk mengatasi masalah batasan kesalingbolehtukaran ini. Parser ini membahagikan signal hos kepada segmen-segmen yang sama panjang dan bahagian - bahagian ini dirujuk sebagai IC's (Imaginary Codewords). Secara teori, entropi (dan redundansi) bagi IC's berubah mengikut kepanjangan segmen. Dengan itu, redundansi yang ditakrifkan boleh diganti dengan payload, dengan menggunakan empat kaedah yang dicadangkan di dalam tesis ini.

Kaedah pertama bernama uReads (Bab 4), dicadang berdasarkan pemetaan IC's ke GRC (Golumb-Rice Codewords). Kemudiannya, GRC's akan diubah-suai untuk menampung external information. Namun demikian, prestasi uReads adalah tidak konsisten dan kapasitinya adalah rendah. Masalah-masalah yang diperhatikan di atas telah diatasi

oleh kaedah kedua, iaitu urDEED (Bab 5). Khususnya, urDEED mempunyai pemetaan yang serupa dengan GRC's, tetapi menggunakan cara pengendalian side information yang berbeza. urDEED menawarkan kapasiti yang konsisten, iaitu 0.169 bpb (bit per bit), dan urDEED boleh digunakan dalam sebarang signal yang telah disulitkan. Walau demikian, pemetaan IC's ke GRC's melibatkan process yang kompleks untuk mengendali side information. Di samping itu, kaedah ini tidah boleh digunakan dalam entropi signal yang tinggi. Untuk mengatasi masalah ini, DeRand (Bab 6), yang menggunakan histogram mapping telah dicadangkan. DeRand mencapai kapasiti sebanyak 0.4 bpb dan berkebolehan untuk mengawal distorsi di dalam entropi signal yang tinggi, contohnya signal-signal yang rawak.

Konsep konvensional data embedding ini telah digeneralisasikan kepada konsep novel data fusion (Bab 7). Berbeza dengan kaedah konvensional data embedding yang hanya mengimplikasikan process di antara dua signal sahaja, iaitu signal hos dan signal payload; konsep data fusion mempunyai kebolehan untuk bergabung lebih daripada dua signal. Justeru itu, kaedah novel DSC (Dual Semantic Code) telah dicadangkan untuk merealisasikan data fusion, di mana setiap DSC codeword boleh menampung dua data pada masa yang sama. Kaedah data fusion yang dicadangkan dapat mencapai kapasiti yang berskala, menggunakan trade-off dengan saiz fail.

Pada keseluruhannya, perbincangan di (Bab 8) menunjukkan cadangan-cadangan kaedah data embedding di tesis ini adalah universal dan mempunyai prestasi yang lebih baik berbanding dengan kaedah-kaedah konvensional, dari segi kesalingbolehtukaran. Tambahan pula, kaedah-kaedah yang dicadangkan adalah reversible dan dapat mengekalkan saiz fail. Di samping itu, data fusion dan DeRand juga menawarkan kebolehan untuk mencapai distorsi dan kapasiti yang berskala.

ACKNOWLEDGEMENTS

I would like to express may sincere gratitude to my supervisor, Dr. KokSheik Wong, for the invaluable support he provided to me from the first moment I started this research work. I highly appreciate his valuable and thorough comments and guidance, and in many ways, I have learnt much from him. I enjoyed the constructive discussions with him throughout the preparation of this study, which enriched in me the spirit of patience, discipline and hard-working.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

The digital revolution comprehensively invades our daily lifestyle thanks to advanced yet low cost capturing, processing and storing devices and technologies. The revolution is also induced by the advances in the communication networks, which allow the exchange of multimedia contents reliably and economically. These advances trigger the generation of massive multimedia contents that play core roles in our daily life, where optimum utilization of these contents requires efficient digital data management. To this end, the concept and applications of data embedding offer a digital right management framework to achieve data annotation, classification, indexing, ownership authentication and integrity preservation[1]. The concept of data embedding is detailed in the following section.

1.1 Overview on Data Embedding

Conceptually, data embedding is based on modifying a host in order to embed a payload (Figure 1.1) (Cover & Thomas, 1991). This process is independent from the content or features of the payload. For that, data embedding can be utilized in various applications, in which the payload has different functionality. However, the features of the host play significant role in data embedding. In particular, the modification of the features of the host should be achieved solely in certain domain or medium. Hence, data embedding is a feature dependent process (Karim & Wong, 2014). Traditionally, this feature dependency aims at reducing the detectability of the existence of the embedded payload. For that, data embedding is referred by the term "data hiding" or "information hiding" in the literature (Petitcolas, Anderson, & Kuhn, 1999).

[1] Various examples on the application of data embedding are detailed in (Cover & Thomas, 1991).

On the other hand, the feature dependency of data embedding process limits the interchangeability among data embedding methods. In other words, the applicability of a data embedding method is restricted by the domain or medium in which it operates. This is a common observation among most conventional data embedding methods as detailed in Chapter 2.

For example, methods that embed data in the compressed domain cannot be applied (i.e., without re-designing them) to embed data in the encrypted domain. Similarly, methods that embed data in the pixels of a raw image cannot be applied to embed data in coefficients of compressed audio, and so on. While a non-interchangeable data embedding method is viable in its designated domain and medium, such method is infeasible when applied to other domains and media. This infeasibility of the conventional feature-dependent data embedding methods is a significant problem because multimedia content is massively generated nowadays.

For example, a user went for a picnic with her children on a sunny beautiful day. The golden lights of the sun were wonderfully penetrating the white clouds in the blue sky, so she took some pictures of this wonderful view using a smart phone. Later on, the children had their lunch and started playing around, so she recorded this joyful moments by capturing a video using the smart phone. In the midst of such happy atmosphere, she recorded a voice message and share it online with other relatives and close friends. At the end of this wonderful day, she needed to annotate her private data captured at the picnic. Unfortunately, annotating these multimedia data in a one step using a single data embedding application is impossible based on the current technology, because each class of data should be processed by its designated data embedding method. Thus, the user needed to embed data in the photos by the software implementation of method A, and had to buy another software, which is based on method B, to embed data in the captured videos. However, the user could not afford the cost to buy another software,

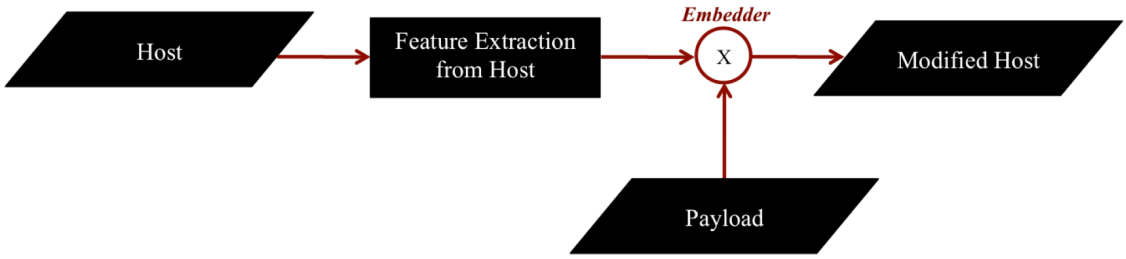


Figure 1.1: The conventional concept of data embedding

which embeds data in audio based on method C. Later on, when the user compressed all the captured multimedia contents of this particular outing for archiving purposes, none of method A, B, or C can be deployed to embed data into the compressed file. In this scenario, the reason of disappointment of the user is the absence of interchangeability among the majority of the existing data embedding methods. While this scenario is about leisure and entertainment, the absence of interchangeability has significant impact on other aspects, such as in the commercialized cloud storage services.

In cloud storage, users rent/buy online storage space, followed by uploading their private contents. These contents are mixture of data in various media, formats and coding structures. Hence, generally, it is almost impossible to practically decode and extract features (for data embedding purposes) of each file uploaded to the cloud. In particular, it is more challenging to extract features from the encrypted or compressed form, as well as data in uncommon format. Thus, the cloud administrator (i.e., a third party which has no access to the original format of the data) cannot annotate, index or classify the files stored online. For that, in this scenario, the application of the conventional non-interchangeable data embedding methods is not feasible and requires the user to reveal some features of his/her uploaded file, which may consist of some private information, to facilitate the selection of method applicable to the file in question.

1.2 Problem Statements

The problem of the limited interchangeability among most conventional data embedding methods is studied. The cause of this problem is the high[2] dependency on the features of the considered hosts in the conventional data embedding methods. For that, the applicability of most the data embedding methods are restricted by a certain domain, medium and coding format structure. Consequently, some potential applications using data embedding cannot be achieved as detailed in Section 2.8. In general, there is no universal data embedding method that is reversibly and interchangeably applicable to any digital signal.

1.3 Objectives of Study

This study aims at generalizing the conventional concept of data embedding towards universal data embedding. To achieve this goal, efforts are channeled to the following:

1. Studying the novel concept and theory of the universal data embedding
2. Putting forward a general framework that establishes the foundation for practical implementation of universal data embedding
3. Designing, implementing and evaluating a set of universal data embedding methods, which are based on the proposed concept and framework of universal data embedding
4. Achieving data embedding with un-conventional hosts that are not considered (or utilized) for data embedding in the existing literatures, such as random signals.

[2]The word "high" indicates that feature-dependency is an inevitable phenomenon in data embedding but decreasing such dependency is expected to increase interchangeability.

1.4 Scope and Limitations of Study

This study is concerned about reversible data embedding. The targeted hosts involve theoretically all media, domains and coding structures. Features such as the carrier capacity and file-size preservation are considered. It is assumed that there are no attacks or transmission error, which the modified host can be exposed to. Thus, the robustness against attacks or transmission error is not considered. In addition, the degree of distortion in the host due to data insertion is ignored because the proposed methods map the hosts into some significantly different forms[3].

1.5 Contributions of Study

This study is the first that considers the interchangeability aspect of data embedding in the existing literatures. In particular, the limited interchangeability among data embedding methods is analyzed (Chapter 2). The novel concepts of universal domain and universal data embedding are put forward to overcome the restricted interchangeability in the conventional methods (Chapter 3). Here, a novel framework to achieve universal data embedding is proposed. The theoretical and implementation aspects of the proposed universal parser are detailed, which establishes the foundation to achieve four proposed universal data embedding methods. In particular, the structure of the GRC codewords are modified (Chapters 4 and 5) to embed payload. In addition, the universal parser enables a random signal to be exploited for data embedding purposes (Chapter 6). Furthermore, the concept of data fusion is proposed to generalize the concept of data embedding and a new set of novel VLC codewords called DSC (Chapter 7) is proposed to realize data fusion.

[3]Except in Chapter 6, distortion is measured merely to visualize the scalability property of the proposed method.

1.6 Organization of Thesis

This thesis is organized as follows. In Chapter 2, conventional data embedding methods in the current literatures are surveyed. Then, an analysis on the interchangeability of the surveyed methods is presented. In Chapter 3, the methodology to overcome the limitation of interchangeability among existing data embedding methods is presented. The general framework for universal data embedding purposes, which decreases the dependency on the host features, is presented. The criteria to evaluate the performance of universal data embedding methods are also presented. Chapter 4 proposes a universal data embedding method based on GRC (Golomb-Rice Code). The performance of this method is further improved in Chapter 5, where the proposed method is applied to the encrypted signals. In Chapter 6, a universal data embedding method in random signal is put forward. A generalization of the concept of the conventional data embedding towards data fusion is proposed in Chapter 7. In Chapter 8, the discussions on the proposed methods are presented. Chapter 9 presents the conclusions and future work.

CHAPTER 2

LITERATURE SURVEY

2.1 Overview

In this chapter, various well-known conventional data embedding methods are surveyed. Particularly, the survey includes difference expansion, histogram shifting, transform based, bitstream mapping, LSB substitution, hybrid and generalized data embedding methods. Then, the surveyed methods are analyzed in terms of interchangeability. Throughout this study, the term "interchangeability" refers to the ability to apply the data embedding method in question interchangeably to signals in various domains, media and formats. In addition, the terms "interchangeability" and "applicability" are used exchangeably.

2.2 Difference Expansion Based Methods

In DE (Difference Expansion) based methods, pixels are manipulated directly in the spatial domain for data embedding purposes (Tian, 2003; Kim, Sachnev, Shi, Nam, & Choo, 2008). In particular, the pixels are grouped into pairs. Then, the difference and average value of each pair are computed. Finally, the difference is expanded to accommodate one bit from the payload. The application of this class of methods is restricted to the spatial domain of an image. The subsequent enhancements on DE do not consider improving their interchangeability, but focuses on output image quality and carrier capacity. For example, Alattar (2004) proposes for applying DE to a group of three or four adjacent pixels in order to increase the embedding capacity. Thodi and Rodríguez (2007) propose to achieve DE in the prediction errors of the pixels for better utilization of the correlation among pixels, where DE is combined in this method with histogram shifting. This

method is further improved by Hu, Lee, and Li (2009) by defining the locations of manipulated pixels (for data embedding purposes) based on the values of the payload. Li, Yang, and Zeng (2011) propose an adaptive DE method which further improves the embedding capacity to embed more bits in the smooth areas of the image. Zhou and Au (2012) propose a strategy to define the number of embeddable bits in an image using DE to ensure certain quality. Li, Li, Yang, and Zeng (2013) propose generalized algorithms for data embedding using DE. While the embedding capacity is improved, the interchangeability of all aforementioned methods is still restricted to an image represented in the spatial domain. Although Chen, Xiang, and Luo (2013) propose to apply DE to embed data in the coefficients of integer discrete cosine transform of audio signals, their method is only applicable in the frequency domain. Hence, the interchangeability of this method is restricted to the audio signal represented in the frequency domain.

All in all, despite DE has received much attention in the data embedding community, improving the interchangeability of DE methods has not been considered in the existing literature due to the dependency of DE on certain histogram features. Therefore, DE methods cannot be applied directly to, e.g., bitstream, compressed, encrypted domains.

2.3 Histogram Shifting Based Methods

In the conventional HS (Histogram Shifting) based methods, the histogram of an image is manipulated to vacate bins. The empty bins (i.e., with zero occurrence in the histogram) are obtained by shifting some intensity levels by means of increasing or decreasing the adjacent bins by some constant. The resulting empty bins are utilized for data embedding. Conventionally, the intensity level that is of the highest occurrence in the histogram is chosen for data embedding. Then, the process of data embedding commences by scanning the pixels and verify their intensity levels. If the intensity level of a pixel is of the highest occurrence in the histogram, this pixel is utilized for data embedding. In par-

ticular, to embed "1", the pixel is mapped to the intensity level (i.e., adjacent bin) that has zero occurrence (i.e., of empty bin) in the histogram. However, to embed "0", no mapping is performed (Ni, Shi, Ansari, & Su, 2006). Here, the applicability of the conventional HS methods is restricted to the spatial representation of an image (i.e., as an array of raw pixel values). The same applicability restriction is observed in the subsequent HS-based data embedding methods. For example, Lee, Suh, and Ho (2006) propose to achieve HS by generating a difference image (i.e., from the original image), followed by applying HS to the difference image. This method is superior to that proposed by Li et al. (2013) in terms of embedding capacity and lowers distortion in the modified image due to better exploitation of the correlation among pixels. Hong, Chen, and Shiu (2009) propose to embed data in the prediction errors of the image, which results in higher carrier capacity than the aforementioned HS methods. Wu and Huang (2012) propose a method to select the intensity levels for histogram shifting purposes such that the distortion in the modified image is suppressed. Obviously, the applicability of these methods is restricted, similar to the conventional HS. This observation includes other HS-based data embedding methods, such as (Xuan et al., 2008; Fujiyoshi, Sato, Jin, & Kiya, 2007; Fallahpour, 2008; Kim, Lee, Lee, & Lee, 2009; Tsai, Hu, & Yeh, 2009; Tai, Yeh, & Chang, 2009; Luo, Chen, Chen, Zeng, & Xiong, 2010; Hong, Chen, Chang, & Shiu, 2010; Gao, An, Yuan, Tao, & Li, 2011).

On the other hand, although HS-based data embedding methods are applicable to other domains and media, the applicability of these methods are still restricted. For example, HS is applied to the coefficients in an audio signal by the method proposed by Li, Lei, Liu, and Yan (2006) and hence its applicability is restricted to the temporal representation of an audio. As another example, HS is applied to embed data in the frequency domain, particularly in quantized transformed coefficients of a video by the method proposed by Shahid and Puech (2013) Also, Chen, Zhang, and Yu (2013) pro-

pose to achieve HS in the entropy coding and quantization stages in JPEG. Hence, the applicability of the these two methods are restricted to the frequency representation of video and image, respectively.

Similar to the development in DE, the performance of HS has been improving over years in terms of embedding capacity and output image quality. However, its interchangeability remains restricted.

2.4 Transformed Based Methods

In TB (Transformed Based) data embedding methods, the host is first transformed into another domain, or in other words, content represented in the transformed form is considered. The features in the new domain are utilized to achieve data embedding. For example, Coltuc and Chassery (2007), Wang, Li, Yang, and Guo (2010), Peng, Li, and Yang (2012), and Coltuc (2012) propose a method to reversibly transform a set of integers (image pixel values in the spatial domain) to another set of integers using the proposed transformation. In the new set of integers, each integer is utilized to accommodate one bit from the payload. In terms of interchangeability, the applicability of these methods is restricted to the spatial domain in the case of image. The restricted applicability is also observed in the methods proposed by (Ogihara, Nakamura, & Yokoya, 1996; Huang, Shi, & Shi, 2000; Chen, Zhang, Ma, & Yu, 2013) and (Xuan et al., 2002; Bhat, Sengupta, & Das, 2010; Wassermann, 2013), which are solely applicable to coefficients of DCT (Discrete Cosine Transform) and DWT (Discrete Wavelet Transform), respectively.

2.5 Bitstream Mapping Based Methods

In BS (BitStream mapping) based methods, a pre-defined look-up table of codewords is utilized to parse the bitstream of a particular host. The parsing process defines two subsets from the look-up table, namely, the set of codewords that actually occurs (i.e., used) in the bitstream, and the set of remaining (if any) codewords that are absent. Then,

data embedding is achieved by mapping the used codewords to un-used ones. Mobasseri, Berger, Marcinak, and NaikRaikar (2010) and Qian and Zhang (2012) proposed to apply BS data embedding to JPEG compressed image. In their approaches, the VLC (Variable Length Codewords) in the bitstream of an image is parsed and classified into used and un-used (i.e., absent) VLC's, followed by embedding data using the aforementioned mapping. The applicability in these methods is restricted to image compressed by the JPEG standard. Xu and Wang (2011) apply the similar approach to embed data in the bitstream of H.264/AVC, where Exp-Golomb codewords are utilized in the same manner. Thus, the applicability of this method is restricted to video compressed using the H.264/AVC standard.

2.6 LSB Substitution Based Methods

In LSB (Least Significant Bit) substitution based data embedding, the LSB of an integer value is substituted by one bit from the payload. This process is preceded by compressing the original LSB bitplane for lossless restoration (Fridrich, Goljan, & Du, 2001; Dittmann, Steinebach, & Ferri, 2002; Celik, Sharma, Tekalp, & Saber, 2005). Although any digital data can appear in binary form and hence LSB substitution can be achieved, LSB substitution is not completely universal. This is because, LSB substitution methods depend on the definition of the position of LSB. In other words, the coding structure of the binary data must be known before hand in order to achieve LSB substitution. Therefore, the applicability of this method is restricted to the defined LSB bitplane of a signal.

2.7 Hybrid and Generalized Methods

In this section, it is shown that some methods combine two (or more) data embedding methods under a single operational framework in order to improve interchangeability. Here, each method operates in a specific context (i.e., domain/medium). The analysis of the context and the decision to apply the appropriate method to each context increase

the overall complexity of the data embedding process. However, in general, the combination of multiple methods increases interchangeability. For example, Shih and Wu (2003) propose a hybrid data embedding method that operates simultaneously in the spatial and frequency domains. They split an image into spatial and frequency domains, and embed data into each domain independently. Particularly, in the spatial domain, some pixels are selected based on their spatial activity features, followed by replacing bits in the selected bitplanes with the payload. On the other hand, in the frequency domain, some DCT coefficients are selected and manipulated to accommodate another part of the payload. The coefficients are selected such that the distortion caused by the manipulation is imperceptible. The applicability of this method is restricted by the spatial and frequency representations of the image. Thus, the applicability of this method is better than those of DE, HS, BS and LSB substitution.

Stankovic, Orovic, and Zaric (2010) propose a method that manipulates features in the spatial and frequency domains using a unified scheme. Hence, unlike the method proposed by Shih and Wu (2003), no combination of different methods is required to process data in the two aforementioned domains. Therefore, this method is a generalized scheme for data embedding in the spatial and frequency domains. Here, both features in the spatial and frequency domains are considered to select the positions where the payload is embedded. Then, the coefficients (obtained by applying short-time Fourier transform to the selected pixels) are modified to embed data. This method is applicable to the spatial/temporal and frequency domains of image, audio and video signals. Obviously, the applicability of this method is the highest among all surveyed data embedding methods. However, this method is still infeasible in other domains such as the compressed and encrypted domains. Hence, this method is not universal.

2.8 Problem Analysis

Generally, the applicability of the surveyed data embedding methods are found to be restricted. In other words, each of the surveyed methods is solely applicable to its designated domain and cannot be interchangeably applied to signals defined in a different domain.

This limited interchangeability is due to the feature-dependency of these methods. In particular, a conventional data embedding method is designed to modify certain features in certain domain(s). The requirement to define features in a certain domain restricts the operation of a data embedding method in different domain.

In the current literature, universal data embedding is of nonexistent due to the aforementioned feature-dependency in the conventional methods. Since the features of data in some applications are not or cannot be defined, devising a universal data embedding method is vital. For example, in a cloud storage, users upload various contents of different formats, domains and media. Practically, it is almost impossible to define features of every data stored in the cloud. Furthermore, if the data are compressed or encrypted, the feature extraction process becomes more challenging. In this case, the cloud administrator, who has no access to the original format (or features) of the data, cannot utilize the conventional data embedding methods to annotate, index, archive, embed metadata or other relevant information in the data (or files stored) in the cloud.

As another scenario, technically, some hosts such as streams of data sent over the communication networks are still not exploited for data embedding purposes due to the limited interchangeability in the conventional data embedding methods. Also, data embedding in texts, for example, is still format/language dependent (Yee, Wong, & Chee, 2012). For that, there are many languages in which their texts remain unexplored for data embedding purposes. In these two cases, a universal data embedding can be deployed.

Finally, the low interchangeability among the existing data embedding methods prevents the realization of a universal data embedding method that can by utilized to embed data into any stream of data stored in any file system. Embedding data in host stored in a particular file system requires a library of various combined data embedding methods, where each of these methods is applied to a particular type of data. This requires the definition of features for each type of data, which is infeasible when multimedia data is considered. The aforementioned applications emphasize the need of a universal data embedding.

2.9 Summary

The well-known classes of data embedding methods, including difference expansion, histogram shifting, transformed based, bitstream mapping, LSB substitution and hybrid methods, are surveyed and analyzed in terms of interchangeability. The analysis shows that conventional data embeding methods are of limited interchangeability. This low interchangeability is an intrinsic feature in these methods because they are tailored for the applications in certain hosts with specific features. Hence, increasing the interchangeability requires the re-designing of these methods. On the other hand, there are many applications that need a universal data embedding method to realize the intended use. For that, the consideration of the concept and applications of the universal data embedding are significant and justify the need of this study.

CHAPTER 3

UNIVERSAL DOMAIN AND UNIVERSAL PARSER

3.1 Overview

In this chapter, a general framework to realize universal data embedding is proposed. This framework is based on the proposed concepts of universal domain and universal parser. The definition of the universal domain and the theoretical features of the universal parser are studied in detail, where the universal parser is utilized later to achieve four different universal data embedding methods. These methods are presented in the subsequent chapters. This chapter ends with the presentation of the criteria to evaluate the performance of the proposed universal data embedding methods.

3.2 Introduction

In general, formats and features of digital signals (or signals hereinafter) are under continuous consideration and development. For example, coding structures, compression standards and encryption schemes vary according to such development. Hence, in order for a universal data embedding method to operate independently from such unpredictable changes, there should be a common domain in which any signal (regardless its underlying features) can be defined, where data embedding can be achieved solely in such a common domain. This domain is named as the *universal domain*.

In this study, the universal domain is devised such that any signal can be defined in that domain. In other words, given any set of features in the signal, these features can be defined as a subset of the universal domain. Consequently, any data embedding method that operates solely in the universal domain is a universal data embedding method. The universal data embedding is achieved by the proposed framework, which is detailed in the

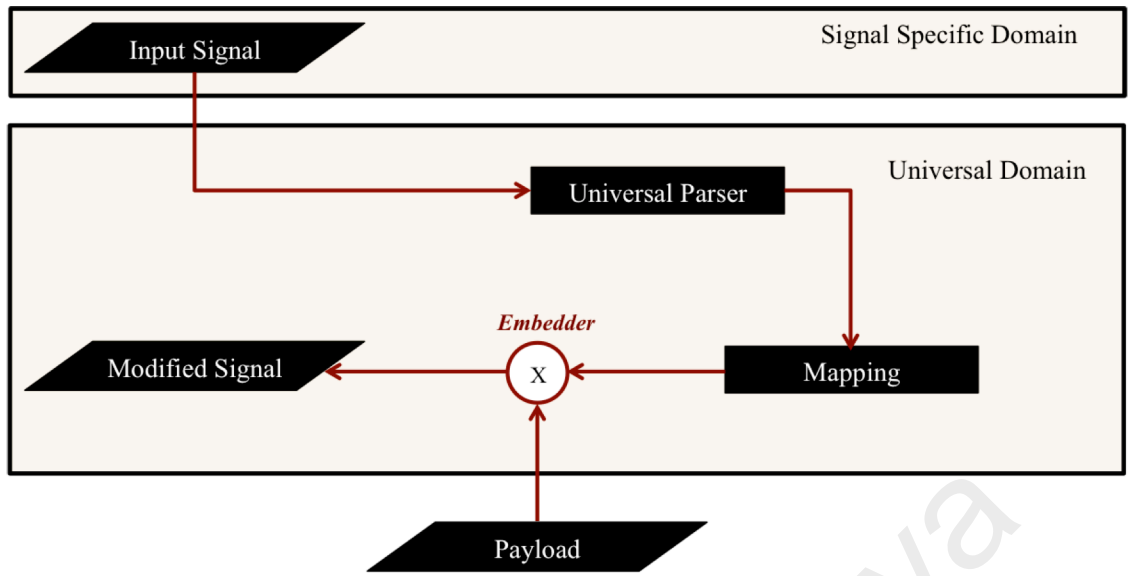


Figure 3.1: The proposed framework of universal data embedding, where the actual mapping and embedding processes are realized in four proposed methods, namely, uREADS (Chapter 4), urDEED (Chapter 5), DeRand (Chapter 6) and Data fusion (Chapter 7).

following section.

3.3 Proposed Framework of Universal Data Embedding

In this study, a general framework (Figure 3.1) that governs any universal data embedding method is proposed. Initially, the input signal (i.e., host) contains some unintelligible features in its specific domain (Section 3.3.1). Hence, this signal is defined in the universal domain (Section 3.3.2). Next, the signal is parsed by the proposed parser referred to as the universal parser (Section 3.3.3). This parser imposes some features in the signal to achieve data embedding. After that, based on the imposed features, the signal is mapped and modified to accommodate payload by four proposed methods, namely, uREADS (Chapters 4), urDEED (Chapters 5), DeRand (Chapters 6) and data fusion (Chapters 7). In the following sections, the features of the input signal, universal domain and universal parser are detailed.

3.3.1 Input Signal

Throughout this study, the input signal (i.e., the host) is denoted by X, which is defined as follows.

Definition 1. *Suppose $\mathbb{A}$ is a set of alphabets, the set $X = \{x_1 x_2 \cdots x_g \cdots x_N\}$ is a finite, stationary and statistically independent sequence of symbols derived from $\mathbb{A}$, where N is the number of symbols in X, g is the unique position of the symbol x_g in X, $x_g \in \mathbb{A}$ and $N \geq 2$* $\square$

Generally, the methods proposed in Chapters 4-7 are theoretically applicable to any set of alphabets considered. However, the scope of this study is restricted by the digital domain hence $\mathbb{A} = \{0, 1\}$.

3.3.2 Universal Domain

Conventionally, a signal X is defined in a certain domain, media and/or coding format. For example, a video camera that captures a scene generates raw sequence of frames defined in the temporal and spatial domains. Such video signal lives in domains that differ from that of, e.g., a text signal. Hence, a data embedding method applicable to raw video cannot be applied to another type of signal (such as raw texts) unless both intersect into a common domain. Based on this observation, it is assumed that a universal data embedding should operate in the universal domain in which any signal X lives.

The universal domain is denoted by $\mathbb{U}_{\mathbb{A}}$ and $X \subset \mathbb{U}_{\mathbb{A}}$. As such, the domain of a universal data embedding function is defined over $\mathbb{U}_{\mathbb{A}}$. Formally, $\mathbb{U}_{\mathbb{A}}$ is a universal set, which is defined as follows.

Definition 2. *Given a non-empty set of alphabets $\mathbb{A}$, the universal set $\mathbb{U}_{\mathbb{A}}$ is a collection*

of all possible sequences of alphabets in $\mathbb{A}$ with length $L \geq 1$, namely:

$$\mathbb{U}_{\mathbb{A}} = \bigcup_{L=1}^{\infty} \mathbb{A}_L \tag{3.1}$$

where $\mathbb{A}_L$ is the set of all sequences of length L generated by performing the following Cartesian product:

$$\mathbb{A}_L = \bigcup_{L=2}^{|\mathbb{A}|^L} \mathbb{A} \times \mathbb{A}_{L-1} = \{X_j^L\}_{j=1}^{|\mathbb{A}|^L}, \tag{3.2}$$

such that $\mathbb{A}_1 = \mathbb{A}$, and X_j^L is sequence of L ordered symbols (i.e., L-tuple), which is defined as follows:

$$X_j^L = \{x_{1,j}^L x_{2,j}^L \cdots x_{g,j}^L \cdots x_{G,j}^L\} \quad \square \tag{3.3}$$

Here, the g-th singleton in X_j^L is notated as $x_{g,j}^L$, where $1 \leq g \leq G = L$. For example, if the set of alphabets $\mathbb{A} = \{0,1\}$, then $\mathbb{A}_2 \in \mathbb{U}_{\{0,1\}} = \{X_1^2, X_2^2, X_3^2, X_4^2\}$, where $X_1^2 = \{0_{1,1}^2 0_{2,1}^2\}$, $X_2^2 = \{0_{1,2}^2 1_{2,2}^2\}$, $X_3^2 = \{1_{1,3}^2 0_{2,3}^2\}$ and $X_4^2 = \{1_{1,4}^2 1_{2,4}^2\}$. The subscript j and superscript L are omitted when there is no confusion. Definition 3.2 leads to the following corollary.

Corollary 1. *If the set of alphabets $\mathbb{A}$ is empty, i.e., $\mathbb{A} = \phi$, then the universal set $\mathbb{U}_\phi$ is also empty, i.e., $\mathbb{U}_\phi = \phi$.*

Proof. If $\mathbb{A} = \phi$, then $\forall \mathbb{A}_L$, $\mathbb{A}_L = \phi \times \mathbb{A}_{L-1} = \phi$, where $L > 1$. Hence, $\mathbb{U}_\phi = \bigcup_{L=1}^{\infty} \phi^L = \phi$ $\square$

Now, given any sequence of symbols X (i.e., signal), the following conditions must be satisfied in order to define X in the universal domain.

Definition 3. *Given a sequence X derived from the set of alphabets $\mathbb{A}_L$, if the following are satisfied for $\forall\, x_g \in X$:*

(1) The position/index of each symbol x_g within X is uniquely defined;

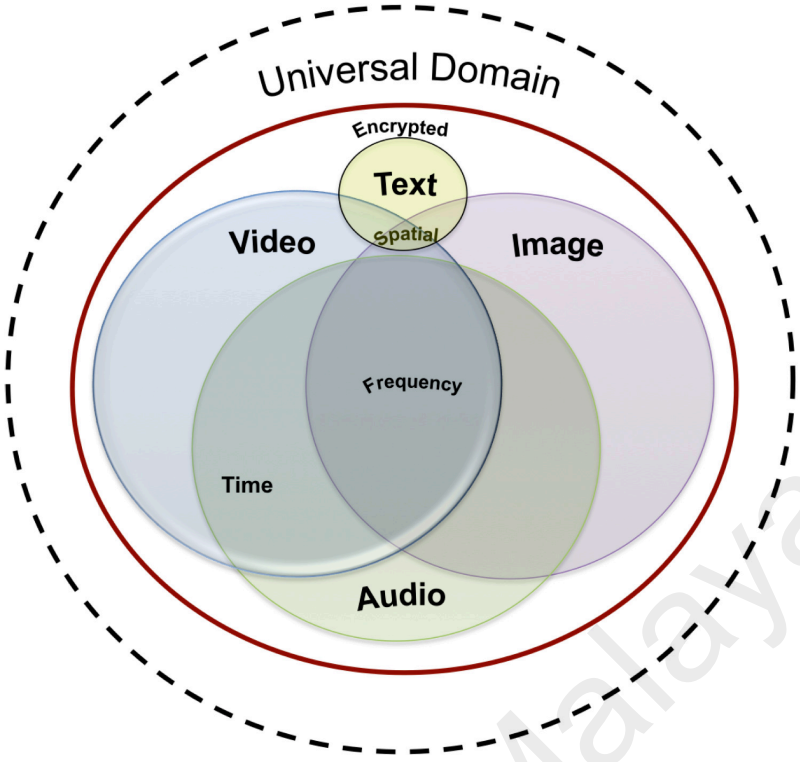


Figure 3.2: The universal domain $\mathbb{U}_A$

(2) The conditional probability $P(\{x_g\}|\{x_{g-1}\}|\{x_{g-2}\}|\cdots|\{x_1\}) = 0$,

then X is defined in the universal domain $\mathbb{U}_A$ □

Informally, each element in the universal domain is a sequence of independent symbols originating from $\mathbb{A}$ where their positions are uniquely defined. Generally, satisfying the conditions (1) and (2) of Definition 3 is sufficient to define any finite signal X in $\mathbb{U}_\mathbb{A}$. However, other features, which include but not limited to the coding scheme of the signal (e.g., type of file or file format), media (e.g., audio, image, text) and domain of the signal (e.g., spatial, frequency, temporal) are not required to define the signal X in $\mathbb{U}_A$. For that, the universal domain intersects with all media and domains as illustrated in Figure 3.2. Thus, any signal can be defined in the universal domain.

3.3.3 Universal Parser

Given X in the universal domain, the only extractable features from X are those defined in conditions (1) and (2) in Definition 3. These features include the definition of the set of stationery symbols in X and their unique positions. Otherwise, no features

(such as domain,media and coding format) are available. Hence, X should be modeled by imposing features on X. Then, X is manipulated based on the imposed features for data embedding purposes. Noteworthy, the modeling of X does not change its original features captured by Definition 3 because this modeling is reversible.

The modeling is achieved by a parser referred to as *universal parser*. By applying the universal parser, a set of imaginary codewords (IC's) is obtained by partitioning the bitstream of X into fixed length non-overlapping segments (each of length L bits) where each L-bits segment eventually becomes the IC. The total number of IC's is denoted by λ_L and it is computed as follows:

$$\lambda_L = \left\lceil \frac{N}{L} \right\rceil, \tag{3.4}$$

where $1 \leq L \leq N$ and $\lceil Y \rceil$ rounds Y to the nearest integer towards positive infinity. The last IC in the bitstream is called the *end-of-signal* IC, which may not be of length L. The sequence number of end-of-signal is recorded as side information and treated as part of the payload. Formally, the universal parser is defined as an ordering function as follows:

Definition 4. *Given X, the universal parser is an ordering function $\mathbb{D}(X,L)$ that partitions X into non-overlapping ordered tuples where each tuple consists of L symbols from X such that:*

$$\mathbb{D}(X,L) = X_L = \{T_1^L, T_2^L, \cdots, T_{\lambda_L}^L\}. \tag{3.5}$$

Informally, T_t^L is an imaginary codeword (IC) of length L. Formally, T_t^L is t-th ordered tuple that consists of L elements from X and is defined as follows:

$$T_t^L = (x_t^1, x_t^2, \cdots, x_t^L). \tag{3.6}$$

Here, the following equation relating the cardinalities holds true:

$$|\{\{x_1^1\},\{x_1^2,\cdots,x_1^L\}\}|+|\{\{x_2^1\},\{x_2^2,\cdots,x_2^L\}\}|+\cdots+ |\{\{x_{\lambda_L}^1\},\{x_{\lambda_L}^2,\cdots,x_{\lambda_L}^L\}\}|=|X|, \tag{3.7}$$

where x_t^l is at position l in the tuple T_t^L, $l \leq L$, $\{\{x_t^1\},\{x_t^2,\cdots,x_t^L\}\}$ is the Kuratowski's representation of the tuple T_t^L and $|U|$ is the cardinality of the set U. Here, the following relation also holds true:

$$\bigcap\{T_t^L\}_{v=1}^{\lambda_L} = \bigcap\{\{x_t^1\},\{x_t^2,\cdots,x_t^L\}\}_{t=1}^{\lambda_L} = \phi, \tag{3.8}$$

where each element x_t^l is unique with respect to its position l □

In the following theorems, it is proven that the entropy of X_L depends on L.

Theorem 1. *Suppose X is a discrete uniformly distributed set of elements such that the probabilities $p(x_1) = p(x_2) = \cdots = p(x_N) = \frac{1}{N}$. If a universal parser orders X such that*

$$X_L = \mathbb{D}(X, L) \tag{3.9}$$

and

$$X_{L+1} = \mathbb{D}(X, L+1), \tag{3.10}$$

then

$$I(T_t^L) > I(T_j^{L+1}) \tag{3.11}$$

holds true $\forall T_t^L \in X_L$ and $\forall T_j^{L+1} \in X_{L+1}$ where $I(T_t^L)$ and $I(T_j^{L+1})$ are the self-information (Shannon, 2001) of the tuples T_t^L and T_j^{L+1}, respectively.

Proof. Since X is discrete and uniformly distributed, $\forall T_t^L \in X_L$,

$$P(T_t^L) = \frac{1}{\lambda_L}, \tag{3.12}$$

and $\forall T_j^{L+1} \in X_{L+1}$,

$$P(T_j^{L+1}) = \frac{1}{\lambda_{L+1}} \tag{3.13}$$

hold true. Increasing L to $L+1$ results in

$$\lambda_L > \lambda_{L+1}. \tag{3.14}$$

Hence,

$$\frac{1}{\lambda_L} < \frac{1}{\lambda_{L+1}}. \tag{3.15}$$

By taking the logarithm of both sides of Inequality (3.15), the following is derived:

$$\log_2\left[\frac{1}{\lambda_L}\right] < \log_2\left[\frac{1}{\lambda_{L+1}}\right] \tag{3.16}$$

$$\Rightarrow -\log_2\left[\frac{1}{\lambda_L}\right] > -\log_2\left[\frac{1}{\lambda_{L+1}}\right] \tag{3.17}$$

$$\Rightarrow -\log_2(P(T_t^L)) > -\log_2(P(T_j^{L+1})) \tag{3.18}$$

and hence $I(T_t^L) > I(T_j^{L+1})$

□

Inequality (3.18) indicates that the amount of self-information associated with the outcome of the elements ordered at length L is higher than that of $L+1$. In other words,

X_L has higher entropy than X_{L+1} as shown in the next theorem.

Theorem 2. *If a universal parser is utilized to order X into X_L and X_{L+1} as defined in Eq. (3.9) and Eq. (3.10), respectively, then*

$$H(X_L) > H(X_{L+1}) \tag{3.19}$$

where X is a set of discrete and uniformly distributed symbols, $H(X_L)$ and $H(X_{L+1})$ are the entropies of the set of tuples X_L and X_{L+1}, respectively, for $1 \leq L \leq N$.

Proof. Initially, $H(X_L)$ is defined as (Vaseghi, 2006; Rabbani & Jones, 1991):

$$H(X_L) = \frac{1}{L} \sum_{t=1}^{\lambda_L} P(T_t^L) \times \theta(T_t^L) \tag{3.20}$$

where $p(T_t^L)$ is the probability of the tuple T_t^L in X_L and $\theta(T_t^L)$ is the length of the code-word required to encode the tuple T_t^L. Since X is discrete and uniformly distributed, $P(T_t^L)$ and $\theta(T_t^L)$ are constants $\forall T_t^L$. That is,

$$P(T_t^L) = \frac{1}{\lambda_L} \tag{3.21}$$

and

$$\theta(T_t^L) = \log_2(\lambda_L). \tag{3.22}$$

Hence, Eq. (3.20) becomes:

$$H(X_L) = \frac{\lambda_L \times \log_2(\lambda_L)}{L \times \lambda_L} \tag{3.23}$$

$$\Rightarrow H(X_L) = \frac{\log_2(\lambda_L)}{L} \tag{3.24}$$

Next, Inequality (3.19) is proven based on Inequality (3.14):

$$\log_2(\lambda_L) > \log_2(\lambda_{L+1}) \tag{3.25}$$

$$\Rightarrow \frac{\log_2(\lambda_L)}{L} > \frac{\log_2(\lambda_{L+1})}{L} \tag{3.26}$$

However:

$$\frac{\log_2(\lambda_{L+1})}{L} > \frac{\log_2(\lambda_{L+1})}{L+1}. \tag{3.27}$$

Hence, the following holds true by transitivity:

$$\frac{\log_2(\lambda_L)}{L} > \frac{\log_2(\lambda_{L+1})}{L+1}. \tag{3.28}$$

Therefore, Inequality (3.28) can be rephrased according to Eq. (3.24) as:

$$H(X_L) > H(X_{L+1}) \tag{3.29}$$

□

Corollary 2. *The following inequality holds true:*

$$H(X_1) > H(X_2) > \cdots > H(X_N) \tag{3.30}$$

Proof. Inequality (3.30) follows directly from Inequality (3.29)

□

Corollary 3. *The upper and lower bounds of Inequality (3.30) are given as follows:*

$$\log_2(\lambda_1) > H(X_2) > \cdots > H(X_{N-1}) > 0 \tag{3.31}$$

Proof. The upper and lower bounds of Inequality (3.30) are defined by taking the limits of Eq. (3.24) to 1 and ∞, respectively, as follows:

$$\lim_{L\to 1} \frac{\log_2(\lambda_1)}{L} = \log_2(\lambda_1) \tag{3.32}$$

$$\lim_{L\to \infty} \frac{\log_2(\lambda_\infty)}{L} = 0 \tag{3.33}$$

□

Example 1. *An example on the change of the entropy $H(X_L)$ (as L changes) is presented. Given an ordered tuple $X = (a_1, a_2, a_3, a_4)$. X can be processed by the universal parser as follows:*

At $L = 1$, $X_1 = \mathbb{D}(X, 1)$ has 4 tuples, namely, $T_1^1 =$ "a_1", $T_2^1 =$ "a_2", $T_3^1 =$ "a_3" and $T_4^1 =$ "a_4". Hence, $\lambda_1 = 4$ and $H(X_1) = \frac{\log_2(4)}{1} = 2$ bits/tuple.

At $L = 2$, X_2 has 2 tuples, namely, $T_1^2 =$ "a_1a_2", $T_2^2 =$ "a_3a_4". Hence, $\lambda_2 = 2$ and $H(X_2) = \frac{\log_2(2)}{2} = 0.5$ bits/tuple.

At $L = 3$, X_3 has 2 tuples, namely, $T_1^3 =$ "$a_1a_2a_3$", $T_2^3 =$ "$a_4\Delta\Delta$" where Δ denotes an arbitrary symbol that can be ignored. Hence, $\lambda_3 = 2$ and $H(X_3) = \frac{\log_2(2)}{3} = 0.3$ bits/tuple.

At $L = 4$, X_4 has 1 tuple, namely, $T_1^4 =$ "$a_1a_2a_3a_4$". Hence, $\lambda_4 = 1$ and $H(X_4) = \frac{\log_2(1)}{4} = 0$ bits/tuple.

Corollary 4. *As a consequence of Inequality (3.31), the following inequality holds true:*

$$R_1 < R_2 < \cdots < R_N \tag{3.34}$$

where R_L is the amount of redundancy in a set of tuples X_L and it is defined as follows:

$$R_L = \Theta_{avg}(X_L) - H(X_L) \tag{3.35}$$

where $\Theta_{avg}(X_L)$ is the average length of the codewords encoding the tuples of X_L and $H(X_L)$ is defined in Eq. (3.24) □

Corollary 5. *Since X can be modeled into X_L by Eq. (3.5) for $1 \leq L \leq N$, then the following holds true:*

$$\mu \geq N + 1 > 2, \tag{3.36}$$

where μ is the number of possible models[1] *to which X can be transformed.*

Proof. By applying the universal parser, X can be transformed into N models by using Eq. (3.5) where in each model, the symbols of X are ordered into tuples of unified lengths L. On the other hand, there is at least one parser $\hat{\mathbb{D}}(X,\sigma)$ that differs from the universal parser, i.e., $\hat{\mathbb{D}}(X,\sigma) \neq \mathbb{D}(X,L)$ where σ is a parsing coefficient (say a seed to generate a sequence of random lengths each less than N). By applying $\hat{\mathbb{D}}(X,\sigma)$, the symbols of X are ordered into tuples of non-unified lengths. Hence, by considering the two classes of parsers together, i.e., $\hat{\mathbb{D}}(X,\sigma)$ and $\mathbb{D}(X,L)$, there are at least $N+1$ models to which X can be transformed. On the other hand, by Definition (4), $N \geq 2$ and hence $N+1 > 2$, which proves Inequality (3.36)

□

For example, suppose a JPEG image is represented by a bitstream of length N bits. Using Eq. (3.5), the image can be modeled into N different ways. On the other hand, one addi-

[1] A model refers to a pattern by which X can be represented by ordering (i.e., grouping) the symbols of X.

tional model can be obtained by applying the standard (i.e., format-compliance) parser to the image.

Generally, Inequality (3.36) indicates that for any set of symbols X of length N bits, X can be transferred into at least $N+1$ different models. When the universal parser is applied, the maximum number of models is limited to N models, where each model is obtained by setting L in Eq. (3.5) to a certain value. Changing the length L leads to a change in the entropy $H(X_L)$ of a set of tuples as shown in Theorem 2. In this context, the length L that defines the largest redundancy R_L is referred to as the *optimum L*. Throughout this study, the optimum L is defined empirically. Here, the removed redundancy generates venues, which are utilized for universal data embedding. Following the proposed framework of universal data embedding in Figure 3.1, the mapping and embedding of payload are achieved in four different ways, which are detailed in Chapters (4$\sim$7).

3.4 Evaluation Criteria

Generally, the performance of a universal data embedding method is evaluated in terms of carrier capacity, reversibility, interchangeability and file-size persevering.

The carrier capacity is defined as the ratio of the size of the embedded payload to the size of the host and presented in the unit of bpb (bits per bit).

The ability to perfectly extract the embedded payload and reconstruct the original host is referred to as reversibility. This feature is significant in applications where data loss is prohibited such as in medical and military imaging applications (Li et al., 2013). Reversibility is achieved with the aid of side information, which are either embedded along with the payload or appended as an extra data to the signal.

The ability to achieve data embedding to any signal regardless of its underlying coding structure/domain/media is referred to as interchangeability (or universal applicability). This is the core objective of this study. In universal data embedding, universal

applicability is verified empirically by executing the implementation code of the proposed method to its completion for any input signal X such that bpb > 0 is obtained.

The ability to generate a modified signal of the exact same size as the original one is referred to as file-size preserving. This feature is significant in applications of limited bandwidth or storage capacity such as communication networks, DVD-RW and flash memories where file-size preservation is crucial. For example, after increasing the file-size due to data embedding, video contents originally fitted in a single DVD may no longer fit in a disc of the same capacity, hence requiring at least 2 DVDs to store the same contents. On the other hand, bitstream size (i.e., packet length) expansion caused by embedding data in network packets may significantly affect the network traffic, where the situation gets more complicated when fragmentation occurs at the routers.

3.5 Summary

The problem of nonexistent of universal data embedding method can be overcame by defining generic operations of data embedding which are commonly applicable in the proposed universal domain. The universal domain is a common domain that intersects with any other domain. For that, any digital signal lives (i.e., can be defined) in the universal domain. Signals in the universal domain are modeled in a way such that new features are defined using the universal parser. In particular, the universal parser changes the entropy (and consequently the redundancy) of the signal based on the partitioning length L. This change in redundancy is exploited to embed external data (i.e., a payload) by proposing four methods in the following chapters.

CHAPTER 4

DATA EMBEDDING USING GOLOMB-RICE CODE

4.1 Overview

In this section, the first proposed universal data embedding method is proposed and named as uREADS (universal Reversible data Embedding method for Any Digital Signal). In this method, the imaginary codewords (obtained by the universal parser) are mapped to GRC's (Golomb-Rice codewords). Then, GRC's are manipulated to accommodate the payload. In particular, each IC is mapped to an mGRC (modified Golomb-Rice Code), which is formed by grouping two GRC's that differ only in their LSB's. The LSB is utilized to embed external payload by means of modulation. Despite inserting external information into it, uREADS preserves the size of the original signal by treating excessive data (due to GRC mapping) as part of the payload. The payload and the original signal can be reconstructed independently from the modified signal. Experimental results suggest that uREADS is applicable to any digital signal and it offers up to 0.074 bpb of carrier capacity.

4.2 Introduction

Data are widely generated nowadays in various formats thanks to the availability of sophisticated capturing devices at affordable prices. However, most, if not all, existing data embedding methods are features dependent and hence they are not readily interchangeable. The applicability of these methods is restricted by either the domain or media of the signal in question. For example, methods designed to operate in the spatial domain (Hartung & Girod, 1998) are not applicable to the frequency domain (Cox, Kilian, Leighton, & Shamoon, 1996), and vice versa. In addition, the methods applicable to a

specific frequency domain for image (Cox et al., 1996) are not applicable to the frequency domain for audio (Bassia, Pitas, & Nikolaidis, 2001). Hence, changing the domain or media requires a re-design of the data embedding method. In addition, the increasing number of standards and formats for signal in various domains emphasize the need of a universally interchangeable data embedding method that is applicable to any digital signal. Stankovic et al. (2010) propose a unified watermarking method applicable to audio (in time domain) and an image (in frequency domain). This method is still not completely interchangeable because it is not applicable to compressed audio/image. Shih and Wu (2003) propose a hybrid method to embed data into spatial and frequency domains of an image, but yet independent and different method is considered for each domain. Since any digital signal can be represented in the bitstream domain, embedding data directly in the bitstream domain will overcome the aforementioned interchangeability problem. The bitstream domain is recognized as an important sub-area of data hiding (Mobasseri et al., 2010; Xu, Wang, & Shi, 2014). Although the data embedding method proposed by Wong, Au, and Wong (2001) operates in the bitstream domain, they need to decode (partially) the bitstream to access the quantized coefficients in the frequency domain for data embedding purposes. Hence, they are not completely applicable in the bitstream domain. To overcome this drawback, Mobasseri et al. (2010) and Xu et al. (2014) embed data directly in the bitstream domain without the need of partial decoding (i.e., de-compression). Features of the entropy coder (i.e., Huffman code) are utilized to achieve the embedding in the JPEG bitstream. However, this method is not applicable to bitstream generated by a different entropy coder (e.g., Arithmetic coding), as well as bitstream of an audio, text or any other content. In this chapter, a universal Reversible data Embedding method for Any Digital Signal (uREADS) is proposed. uREADS defines the input signal in the universal domain and applies the universal parser to generate IC's (imaginary codewords). These codewords are statistically mapped into modified version of Golomb-Rice codes

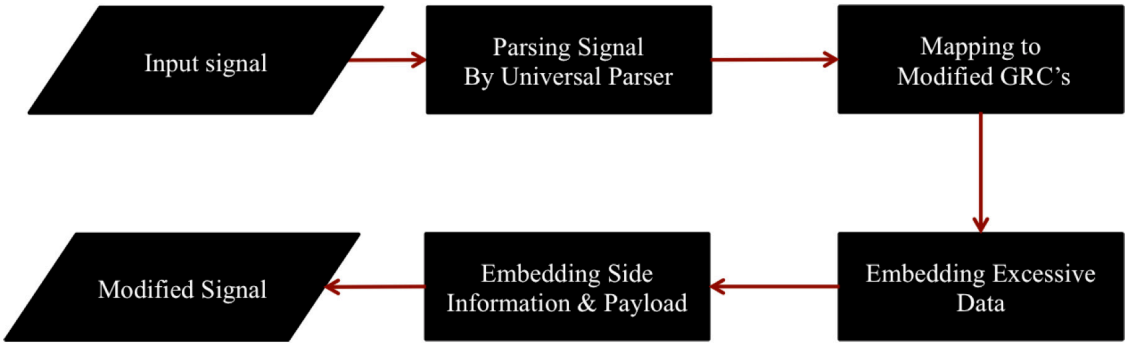


Figure 4.1: The proposed framework of uREADS, where the input signal is defined in the universal domain

(mGRC). Here, an mGRC is formed by grouping two GRC's that differ only in their LSB's. The LSB is utilized to accommodate one bit of the payload. uREADS is universally applicable to any digital signal and it is reversible in which case both the host and payload can be perfectly restored. The extraction of the payload is achieved without the need to restore the original signal, and vice versa. Despite inserting additional external information into it, uREADS preserves the size of the original signal. The performance of uREADS is verified empirically.

4.3 The Proposed uREADS

Figure 4.1 summarizes the procedures in uREADS and each procedure is discussed briefly as follows: 1) The universal parser is unaware about the source coding applied to generate the input signal X. It merely parses X into virtually correlated IC's. Here, the bits in IC's are correlated differently and independently from the actual correlation obtained by the applied source coding. However, reversibility is ensured by maintaining the relative order of each bit. 2) The IC's are then mapped into a set of modified GRC's according to their probability of occurrences. 3) To preserve the original file-size, any excessive data caused by the mapping are embedded into the IC's. 4) Side information and other header components are embedded for reversibility purposes. Finally, the payload is embedded into the remaining (available) IC's. The procedures of uREADS are further detailed in following sections.

4.3.1 Universal Parser

The universal parser supposes that the bitstream is uncorrelated. Thus, each bit is read sequentially while keeping its order unchanged for reversibility purposes. The assumption made (i.e., bits are uncorrelated) implies that X is virtually free of codewords. Hence, the universal parser imposes a virtual correlation among the bits, forming a set of IC's, as detailed in Section 3.3.3. Here, the Probability of Occurrence (PoO) of each IC is computed. IC's are then sorted in descending order according to their PoO's. The order is embedded as side information **S** along with the payload. The size of **S** is proportional to the number of bins in the histogram of the IC's that have counts larger than zero, and hence **S** increases when L increases. To achieve reversibility, L, **S**, and the reminder length r in GRC (defined in Section 4.3.2) are stored as side information.

4.3.2 Mapping IC's to modified GRC's

In this stage, the IC's are entropy coded by a modified version of GRC's. Here, it should be noted that this entropy coding does not aim at compressing X. In other words, the entropy coding is utilized for data embedding purposes. The entropy coding process starts with the construction of the histogram of the set of IC's. Then, IC's of higher PoO's are coded by GRC's of shorter length, and vice versa. For reversibility purposes, the sorted IC's (based on their PoO's) are stored as side information along with the augmented payload.

By definition, each GRC consists of three pre-defined parts, namely; (1) quotient part (q) which consists of a sequence of zeros of length j; (2) termination bit of constant value "1", and; (3) remainder part (b) which consists of an arrangement of bits of constant length r (Golomb, 1966; Weinberger, Seroussi, & Sapiro, 2000). Hence, any GRC is of the following form:

$$(q1b) = (0_1 0_2 \ldots 0_j 1 b_1 b_2 \ldots b_r). \tag{4.1}$$

Conventionally, $j \geq 0$ and hence q may or may not exist in a GRC codeword. This conventional form of GRC's is modified in two aspects to form mGRC (modified GRC). The first modification is imposing the condition $j > 0$, implying that the quotient part must exist in any GRC. Hence, the minimum length of $|GRC|$ is $r+2$ bits. For example, if $r = 2$, then the set of mGRC's consists of $(0100), (0101), (0110), (0111), (00100)$ and so forth. The second modification is grouping a pair of GRC's that differ only in the LSB for data embedding. Here, use the terms LSB of GRC and polarity of GRC are used interchangeably. For example 0100 and 0101 are combined to form one mGRC = 010$\mathfrak{p}$, where $\mathfrak{p}$ is the polarity and it is utilized for data embedding as detailed in Section 4.3.3. In order to preserve the file-size of the original signal after mapping, $|mGRC| = r+2 = L$ must hold true.

4.3.3 Data Embedding

Practically, mapping IC's into modified GRC's mostly increases the size of the bitstream, i.e., the mapping results in $|mGRC| > L$. In order to preserve the file-size, the excessive data is first embedded into modified GRC's, and the remaining modified GRC's are utilized for embedding the header and payload as detailed in the following subsections.

4.3.3 (a) Embedding Excessive Data

Generally, by grouping each two GRC's to form a single mGRC, the length of resulting codewords increases by 1 bit per $2^{(r-1)}$ codewords. For example, for $r = 3$, the codeword mGRC=0111$\mathfrak{p}$ is followed by the codeword mGRC=00100$\mathfrak{p}$. Here, the size of the later codeword increases by 1 bit in comparison to the first codeword, i.e., mGRC=00100$\mathfrak{p}$. Such increment causes some codewords to be of length $|mGRC| > r+2$, and mapping any IC's to such lengthy GRC's causes file-size increment. Therefore, $|mGRC| = r+2 = L$ is imposed for each codeword to preserve the file-size of the bitstream

(after the mapping). This is achieved by removing excessive zero(s) in the quotient part of codeword of $|GRC| > r+2$ and embedding them in the codeword(s) that follows. The number of removed zeros is denoted by R_0^i and it is computed as $R_0^i = |mGRC_i| - (r+2)$, where $mGRC_i$ refers to the i-th mGRC. When there is no confusion, the subscript i may be dropped. The process of removing the excessive zeros R_0^i and embedding them into the codewords that follows are performed by Algorithm 1. Here, R_{GRC} denotes the number of mGRC's such that $\forall mGRC, |mGRC| > L$. As an example, assume two mGRC's (i.e., $mGRC_1$ and $mGRC_2$) as shown in Figure 4.2. Here, $r = 2$ bits hence $mGRC_1$ has 2 excessive zeros, i.e., $R_0^1 = 00$. Thus, the size of $mGRC_1$ should be reduced from 6 to 4 bits in order to preserve the original file-size. The reduction is achieved by removing the first two 0's from the quotient part of $mGRC_1$. The removed 0's are embedded as payload in $mGRC_1$ and $mGRC_2$, respectively, as detailed in Figure 4.2. After embedding all excessive zeros, the bitstream of the host now consists of fixed-length codewords each of length $|GRC| = r+2$. In a rare case, a codeword GRC (or more) with excessive data may come at the end of the bitstream, where no subsequent codewords can be utilized for embedding R_0^i. When this occurs, the count of such codeword(s) is recorded as R_{mGRC} in the header, and R_0^i is embedded in the header as well.

4.3.3 (b) Embedding Header and Payload

After embedding the excessive data R_0^i, the header and the payload are embedded in the remaining mGRC's. The available mGRC's are recognized by inspecting their MSB's. If MSB=1 in a mGRC, it signifies that such mGRC has been utilized to accommodate a removed (excessive) zero, and hence, it is not eligible for data embedding. If MSB=0, then such mGRC is eligible for data embedding. The embedding is carried out by modulating the polarity of such mGRC using the payload (1 bit per mGRC). For simplicity, $\mathfrak{p}$ is set to 1 to embed "1" and 0 to embed "0" from the payload. For example, if

Algorithm 1 Embedding of excessive data

Let $i = 1$, $R_0^i = 0$, $\mathfrak{p} = 0$
Compute R_0^i
if R_0^i>0 **then**
    Flip $\mathfrak{p}$
    Flip the termination bit
    Remove R_0^i from q_i in the codeword $mGRC_i$
    Flip the MSB of $mGRC_i$ and MSB's of R_0^i codewords that follow
    Set the polarity of $mGRC_i$ and R_0^i codewords that follow to the current $\mathfrak{p}$
else
    Increase $i = i + 1$
    if $i = EOF$ **then**
        Stop
    else
        Go to Step (2)
    end if
end if

Algorithm 2 Restoration of excessive data

Let $i = 1$, $R_0^i = 0$, $\mathfrak{p} = 0$
Read the termination bit of $mGRC_i$
if termination bit=0 **then**
    Read $\mathfrak{p}$ in $mGRC_i$
    Set $R_0^i = R_0^i + 1$
    Read $\mathfrak{p}$ of the next codeword $mGRC_{i+1}$
    if $\mathfrak{p}$ of $mGRC_i$= $\mathfrak{p}$ of $mGRC_{i+1}$ **then**
        Set $R_0^i = R_0^i + 1$
        Go to Step (6)
    else
        Flip MSB of $mGRC_i$ and the R_0^i codewords that follow
        Append R_0^i zero to q_i in $mGRC_i$
        Increase $i = i + 1$
        if $i = EOF$ **then**
            Stop
        else
            Go to Step (2)
        end if
    end if
end if

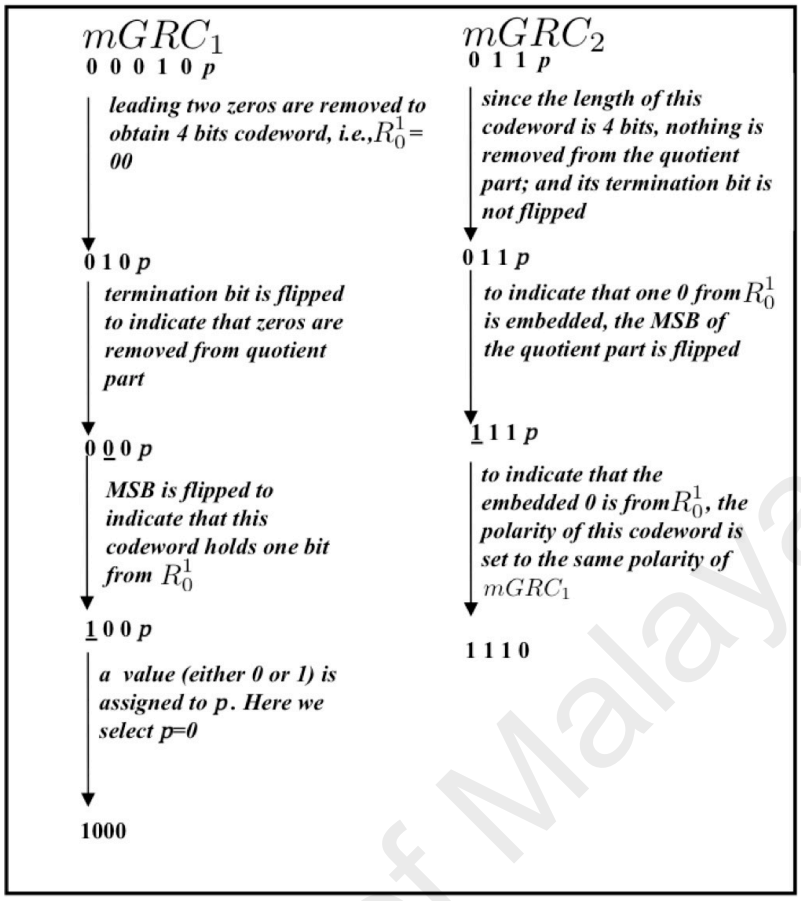


Figure 4.2: An example on embedding excessive data in the proposed uREADS

mGRC= 0010$\mathfrak{p}$, to embed "0", mGRC= 00100, and vice versa. Since each mGRC can hold 1 bit of information, the raw carrier capacity $\mathbf{C}_{raw} = |\lambda_L|$, where λ_L is defined in Eq. (3.4).

4.3.4 Restoration of Original Bitstream & Extraction of Payload

The restoration process includes retrieving two types of data. The first type is header/payload data, and the second type is the original bitstream. Here, retrieval can be achieved independently in any order. In particular, it is not required that the original bitstream is restored first in order to restore the header/payload, and vice versa. To restore the header/payload, the MSB of each codeword (of fixed length $r+2$) is checked. If MSB=1, then such codeword does not accommodate any header/payload data. Hence, it is skipped to the next codeword. If MSB=0, then one bit of the header/payload is restored by reading the polarity of this codeword. The restoration process continues until it

reaches IC_{λ_L}. The restoration of the original bitstream is achieved in two stages: (**Stage I**) Restoring the removed (excessive) zeros from the quotient part for codewords of original lengths $|GRC| > r+2$. This restoration is achieved by Algorithm 2; (**Stage II**) Restoring the original IC's. The restoration starts by reconstructing the header data as detailed at the beginning of this section. Then the table of sorted IC's (according to their PoO's) and their corresponding GRC's are rebuilt. Then the bitstream is parsed from the beginning, and each mGRC is mapped to its original IC until IC_{λ_L} is reached.

4.4 Experimental Results

Experiments are carried out to verify the performance of uREADS by implementing it in C programming language. Three types of data are tested, namely, text, image and audio, each encoded/compressed by different source coding schemes. In particular, the experiment on text involves Microsoft Word and Latex. The effective carrier capacities (in bpb) result by applying uREADS to aforementioned test signals are recorded in Table 4.1. Here, the effective carrier capacity $\mathbf{C}_e$ is defined as:

$$\mathbf{C}_e = \frac{\mathbf{C}_{raw} - (\text{Header data} + R'_0)}{\text{size of signal (bits)}} \tag{4.2}$$

where $R'_0 = \sum_{i=1}^{\lambda_L} R_0^i$ and $\mathbf{C}_{raw}$ is the raw capacity. It is observed the considered text bitstreams can successfully accommodate payload at (L,r)=(4,2). However, the carrier capacity varies, where MS-Word signal achieves $\mathbf{C}_e = 0.031$ and Latex signal achieves $\mathbf{C}_e = 0.051$ bpb.

The experiments on image are carried out using the gray-scale Lenna image of size 512×512 pixels with 8 bits pixel depth. The experiment involves the bitstreams of lossy compressed image (by standard JPEG and JPEG2000 at quality factor = 90 and 50, respectively), losslessly compressed by (JPEG-LS) and un-compressed Bitmap. As shown in Table 4.1, $\mathbf{C}_e > 0$ is attained for most of these bitstreams at (L,r)=(6,4), except that of

JPEG2000 due to the variation in the distribution of IC's in the histograms of the original bitstreams. Such a variation (among the bitstreams of Lenna) affects the obtained $\mathbf{C}_e$, which ranges from 0.018 to 0.023 bpb as shown in Table 4.1.

Last but not least, experiments on audio are carried out on an audio bitstream of length 2 seconds at sampling rate of 44.1KHz with 16 bits per sample. The experiment involves the bitstream of lossy compression by MP3 (MPEG-1 layer III) at 117 Kbit (VBR), the bitstream of losslessly compressed by FLAC (Free Lossless Audio Codec) and un-compressed stream coded by WAVE (Wave Audio File format). It is shown in Table 4.1 that $\mathbf{C}_e > 0$ is attained for all audio bitstreams but at different (L, r) settings for the same reason discussed earlier. It is observed that $\mathbf{C}_e$ varies between 0.526×10^{-4} to 0.042 bpb for all the audio formats considered. The high $\mathbf{C}_e$ in WAVE format is due to the PoO of IC's in its bitstream. In particular, 96% of the IC's are mapped to the first GRC of mGRC, which is of length $r+2$. Therefore, more $\mathbf{C}_e$ is obtained as a consequence of reducing R_0^i.

In terms of file-size preserving, for all test signals considered, uREADS does not cause any increment in file-size when compared to its original counterpart. This suggests that, by using uREADS, more information can be carried within the input signal without increasing its size. The experiments also verified that uREADS is a universal data embedding method because it can handle any type of signal stored can be defined in the universal domain. However, uREADS results achieves inconsistent carrier capacity. In addition, L and R are manually tuned to suite the statistical features of the considered host. These problems are addressed in the next chapter.

4.5 Summary

A novel data embedding method universally applicable to any digital signal was proposed. Features of the source coder were ignored and virtual correlation was imposed

Table 4.1: The performance of uREADS on different bitstreams

Stream	L (bits)	r (bits)	C_e (bpb)
MS-Word	4	2	0.031
Latex	4	2	0.051
JPEG-baseline	6	4	0.018
JPEG-LS	6	4	0.018
JPEG-2000	5	3	0.023
BMP	6	4	0.018
MP3	7	5	0.526×10^{-4}
FLAC	3	1	0.074
WAVE	3	1	0.042

on the sequence of bits by grouping them into IC's using the universal parser. Then, the IC's were mapped probabilistically to the modified Golomb-Rice codewords, which in turn utilized for data embedding. The proposed method is able to preserve bitstream size of the original signal while holding external information. It is also completely reversible where the original signal and embedded payload can be obtained independently. In the best case scenario, $\mathbf{C}_e = 0.074$ bpb was achieved. However, the performance of this method in terms of carrier capacity is low and inconsistent. In addition, the parameter L and R vary among the signals and require manual tuning.

CHAPTER 5

DATA EMBEDDING IN ENCRYPTED DOMAIN

5.1 Overview

In the previous chapter, the proposed universal data embedding method suffers from inconsistent and low effective carrier capacity. In addition, the segmentation length L (in the universal parser) and the remainder part must be manually tuned. To overcome these problems, it is proposed in this chapter a universal data embedding method referred to as urDEED (universal reversible Data Embedding in Encrypted Domain). Here, although urDEED follows the proposed framework of universal data embedding hence urDEED is theoretically applicable to any signal X, data embedding in the encrypted domain is considered because the conventional methods require partial access to the features of the signal prior to encryption which may lead to unauthorized viewing/accessing to the original (i.e., un-encrypted) signal. Thanks to the definition of X in the universal domain, urDEED operates completely in the encrypted domain, i.e., it requires no features of the signal prior to the encryption process. This motivates the application of urDEED to the encrypted signals in which case universal data embedding is required. In particular, urDEED exploits the coding redundancy of the encrypted signal by partitioning it into IC's. Then, IC's are entropy encoded by using *Golomb-Rice codewords* (GRC's). Finally, each GRC is modified to accommodate two bits from the augmented payload. urDEED is designed to preserve the same file-size as that of the original input (encrypted) signal by embedding the quotient part of the GRC's as side information. Moreover, urDEED is consistently reversible and universally applicable to any digital signal encrypted by any encryption method. Also, this method offers higher and stable carrier capacity than

the pervious one proposed in Chapter 4, i.e., uREADS. In addition, the data embedding algorithm assumes consistent setting of L and R. Experimental results show that urDEED achieves an average carrier capacity of ~ 0.169 bits per every bit of the encrypted (host) signal.

5.2 Introduction

Multimedia contents are increasingly produced and communicated in recent years thanks to the availability of efficient capturing devices at low cost and the existence of ubiquitous network environment. To this end, *encryption* is applied as the main conceptual and technical method to preserve privacy, integrity and access control of multimedia contents. Practically, encrypted data are unintelligible. Hence, it is difficult for a third party to extract features, if any available, from an encrypted signal without the legitimate decryption key. However, data management system for encrypted signal by a third party is highly demanded nowadays. For example, data are increasingly stored remotely in on-line storage servers (such as cloud storage), which are managed by a third party and the data are often encrypted to avoid unauthorized viewing. Here, *data embedding* can be adopted directly for managing encrypted signal because it allows the insertion of *extra information*, such as the particulars of the owner, copyright information and other relevant metadata, directly into an encrypted signal.

Date embedding is based generally on modifying some *features* of the host in order to embed the payload (Lei, Soon, & Li, 2011; Cheddad, Condell, Curran, & Kevitt, 2010; Xu, Wang, & Wang, 2011; Lusson, Bailey, Leeney, & Curran, 2013; Wong, Qi, & Tanaka, 2007). However, the extraction of features, if any available, is technically challenging in the encrypted domain because the encrypted signal is unintelligible. For that, most of the existing data embedding methods in the encrypted domain require partial access to other domains, which may reveal the pre-encryption features of the signal and

hence may possibly lead to security breach. In other words, the existing methods rely significantly on the underlying coding structure of the content, the type of media, or the properties of the domain in which the signal is defined prior to encryption. For example in the method proposed by Zhao, Kou, Li, Dang, and Zhang (2010), features of an image in the frequency domain (i.e., DCT domain) are exploited to achieve irreversible data embedding in the encrypted domain. The encryption is achieved by manipulating selected DCT coefficients, which is guided by the public-key structure without determining a specific standard or method. On the other hand, Lian, Liu, Ren, and Wang (2007) proposed a data embedding method for encrypted H.264/AVC compressed videos. In their method, motion and residual information are partially encrypted, followed by the modification of the prediction modes in an INTRA frame and flipping sign of the coefficients. Then, data embedding is achieved by utilizing the non-zero coefficients in the frequency domain. Similarly, Cancellaro et al. (2011) achieve the same commutative property in images stored by using coefficients of the tree structured Haar transform. In the method proposed by Zhang (2012), Hong, Chen, and Wu (2012) and Zhang (2011), an image is encrypted by applying XOR operation on its pixels using a pseudo-randomly generated bit sequence determined by a key. Then, payload is embedded by modifying group of pixels in a pre-defined manner. The reversibility property in these methods is achieved by exploiting the spatial correlation among pixel values.

Strictly speaking, a practical data embedding method in the encrypted domain should operates solely in the encrypted domain, where features of the signal prior to the encryption stage should be kept unknown to the data embedder. However, such requirement is not considered in most existing methods because they are designed to exploit selected features of the original signal in its pre-encryption stage to realize data embedding. For that, the existing methods are not operating solely in the encrypted domain, but instead they involve two or more domains. In addition, reversibility is not consistently achieved by

the methods proposed by Zhang (2012), Hong et al. (2012) and Zhang (2011). Hence, the aforementioned methods are not viable in applications where any loss of data is prohibited such as medical, military, and forensic. More importantly, they are not interchangeable and hence the proposed algorithms are not applicable when considering standard encryption methods such as AES (Kim, 2012), Triple-DES (Merkle & Hellman, 1981) or Blowfish (Mousa, 2005).

In this chapter, a novel Universal Reversible Data Embedding method in the Encrypted Domain (urDEED) is proposed. Our method operates solely in the universal domain, which intersects with the encrypted domain as shown in Figure 3.2. Hence, it is not required to exploit features of the signal prior the encryption in order to achieve data embedding. Here, the coding redundancy in the encrypted signal is exploited by entropy coding the signal and the resulting codewords are modified for data embedding purposes. Performance of the proposed urDEED is verified empirically. urDEED achieves the following features: (1) complete interchangeability, and hence, it is applicable to any signal encrypted by any encryption scheme; (2) consistent reversibility, in which both the original encrypted signal and the embedded data can be losslessly restored, whereas the reversibility in the methods proposed by Zhang (2012), Hong et al. (2012) and Zhang (2011) is conditional, and; (3) complete file-size preservation to that of the original encrypted (input) signal despite external data is embedded into it. In general, urDEED is the first data embedding method applicable to any encrypted signal, in which no pre-encryption feature is needed for reversible data embedding.

5.3 Applications of Data Embedding in Encrypted Domain

Ideally, data embedding in encrypted domain should be a process of manipulating an encrypted signal directly to accommodate payload without decrypting the signal or requiring any knowledge about its features prior to the encryption stage. Here, the data

embedder can be a third party who is not authorized to access to the original content of the signal (i.e., plaintext) and hence has no access to its features in the pre-encryption stage. For example, in medical imaging, an image is encrypted to protect privacy of the patient. However, a third-party who manages these image (e.g., system administrator, clerk, nurse) should be able to embed external relevant metadata directly into the encrypted image. Here, the data embedder should neither need to decrypt the image nor exploit any known feature(s) of the original signal in order to achieve data embedding. As another application, in the cloud computing scenario, users of the facility may encrypt their data to protect privacy. Hence, embedding the genre/classification information, ownership information, data retrieval information or any other relevant information into the encrypted data by cloud-computing service provider should be achieved practically without the need to decrypt the data. Another possible application is to preserve the integrity of encrypted signal by computing the hash value of the encrypted signal and embedding the hash directly into the encrypted signal. Similarly, CRC (Cyclic Redundancy Check) or other error-correction information can be embedded in the encrypted signal to serve the same purpose. The aforementioned applications justify the needs of data embedding method in encrypted domain.

5.4 Framework of urDEED

In general, an encrypted signal appears random and it should be unintelligible to any party. As such, it is difficult to parse the bitstream of an encrypted signal into a meaningful form. Based on this observation, a framework (Figure 5.1), which hypothetically defines some features in the encrypted signal in order to process it, is proposed. In particular, the universal parser models the signal into imaginary codewords, which are in turn entropy coded to exploit coding redundancy (Gonzalez & Woods, 2002; Vaseghi, 2006; Rabbani & Jones, 1991). In this method, GRC is considered for entropy coding purposes.

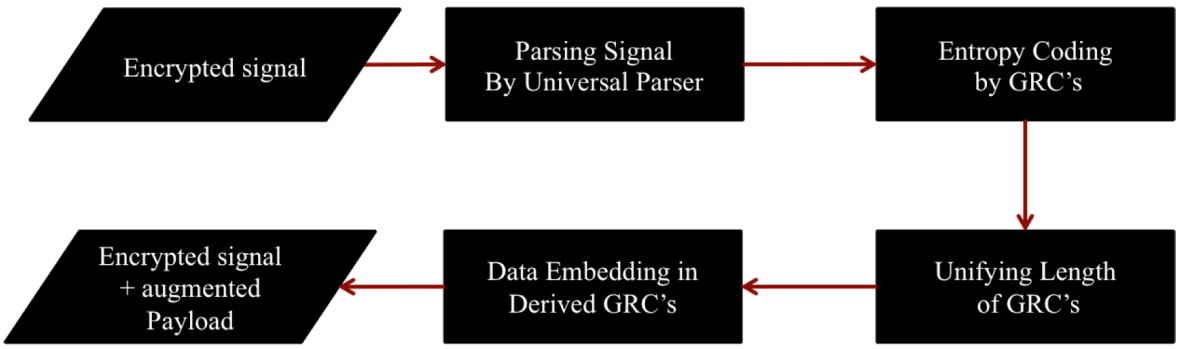


Figure 5.1: Flow of operations in urDEED

Next, the variable-length GRC's are converted into fixed-length GRC's in order to preserve file-size to that of the original (encrypted) signal. Finally, payload is embedded into the fixed-length GRC's. Here, the processing of side information and the venue of data embedding in GRC's differ from those applied in uREAD. The detail of each aforementioned operation is described in the following sub-sections.

5.4.1 Parsing Signal By Universal Parser

Since the extraction of features from unintelligible encrypted signal is technically challenging, new feature(s) are assigned to the encrypted signal and achieve data embedding by modifying these features. The assignment of new features is achieved by the universal parser (Section 3.3.3), which converts the encrypted signal into a set of imaginary codewords (IC's) by partitioning the bitstream of the encrypted signal into fixed length non-overlapping segments (each of length L bits). The features of IC's, i.e., tuples, are presented in Section 3.3.3

5.4.2 Entropy Coding

In urDEED, the set of imaginary codewords (IC's) are entropy coded for three purposes: (A) to represent the IC's in a format that can be modified in order to embed the augmented payload; (B) to preserve the original file-size of the encrypted signal, and; (C) to remove some redundant data in order to make room for accommodating the augmented payload. Hence, it should be noted that entropy coding in urDEED does not aim at com-

pressing the encrypted signal. In this chapter, Golomb-Rice codewords (GRC's) (Eq. 4.1) are utilized for entropy coding purposes. The entropy coding process starts with the construction of the histogram of the set of IC's. Then, IC's of higher occurrences in the bitstream are coded by GRC's of shorter length, and vice versa. For reversibility purposes, the sorted IC's (based on their frequencies) are stored as side information along with the augmented payload. Here, $j > 0$ is imposed for the set of GRC's, as described in Section 4.3.3 (a). However, no other modification is made to the set of GRC's.

5.4.3 Unifying the Length of GRC's

As described in the Section 4.3.3 (a), GRC's are variable-length codewords. Hence, some GRC's are of length $> L$ and they cause file-size increment. In order to preserve the file-size of the original encrypted signal, it is imposed that all GRC's must be of constant length $L = r + 2$ bits. Hence, when the length $L > r + 2$ bits, the quotient part q of each GRC is trimmed and appended to the augmented payload. In this chapter, the trimmed q's are processed differently from the method uREAD presented in Section 4.3.3 (a), as detailed in the following sub-sections.

5.4.3 (a) Trimming q's

Trimming is achieved by removing j bits in the order from left to right, (i.e., the entire quotient part q) of each GRC. The trimmed q is replaced with a dummy value of 0. The trimmed q's may be flipped to mark the end of a codeword and the beginning of another codeword as shown in the next sub-section. Thus, after the trimming process, all GRC's are in the general form of $(01b) = (01b_1b_2 \ldots b_r)$. For example, if $L = 4$ and given GRC = (000100), then such GRC is trimmed to (0100), where the trimmed 000, i.e., the entire quotient part, is appended to the augmented payload.

5.4.3 (b) Appending Trimmed q's to Augmented Payload

Given $GRC_i = (q_i 1 b_i)$ such that $\forall GRC_i, |GRC_i| > r+2$ for $i = 1, 2, \cdots, \mathfrak{n}$, the quotient part of these GRC's are appended to the augmented payload as $q_i \overline{q_{i+1}} q_{i+2} \overline{q_{i+3}} \cdots \overline{q_{\mathfrak{n}}}$ (assuming $\mathfrak{n}$ is an even number), where $\overline{q_i}$ is the logic inverse of q_{i-1}. The logical inverse operation plays a crucial role in signifying the end of the quotient part for a GRC_i. In particular, the length of zero-run (or one-run) determines the number of zeros (i.e., length j) in the original quotient part (which was earlier trimmed from GRC_i), and the change from '0' to '1' (and vice versa) marks the beginning of the next quotient part, i.e., GRC_{i+1}. This appending operation is further illustrated in the following examples.

Example 2. *Constant length scenario*

Given $\{(\boldsymbol{0}100), (\boldsymbol{0}101), (\boldsymbol{0}110)\}$ *as the set of GRC's. The corresponding q's are* ***0****,* ***1****, and* ***0****, respectively. Note that the original q of the second GRC (i.e., 0101) is 0, but it is logically inversed (i.e., flipped) to* $q = 1$.

Example 3. *Variable length scenario*

*Suppose {(****00000*** *1 00), (****00*** *1 01), (****000*** *1 10), (****000*** *1 00)} is a set of GRC's. Then, q of these GRC's are appended to the augmented payload as* ***00000****,* ***11****,* ***000*** *and* ***111****, respectively.*

5.4.4 Data Embedding in Derived GRC

The augmented payload now consists of the trimmed quotient parts, side information (i.e., sorted IC's and λ_L), and the actual payload as illustrated in Figure 5.2. This augmented payload is then embedded in the set of fixed-length GRC's. Each fixed length GRC = $(01b) = (01b_1 b_2 \ldots b_r)$ can accommodate two bits from the augmented payload. In particular, the dummy "0" and the termination bit, i.e., "1" that follows, in the trimmed

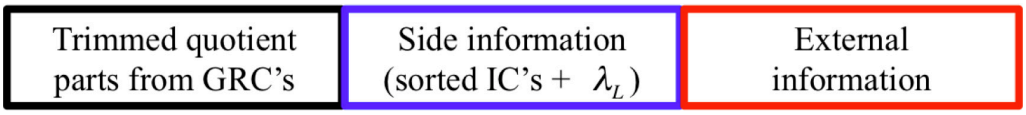


Figure 5.2: Layout of the augmented payload

GRC are replaced by the information (i.e., two bits) to be embedded. In other words, a modified GRC becomes $(u_1u_2b_1b_2\cdots b_r)$ where u_1 and u_2 are the two bits obtained from the augmented payload.

The first component of the augmented payload (i.e., the trimmed quotient parts) is embedded, followed by the side information. Here, the length of those trimmed q's is at least λ_L bits (since $j > 0$ is enforced), and the length of the side information is $L \times M$ bits where M is the number of the imaginary codewords that actually occur (i.e., frequency of occurrence greater than zero) and $1 \leq M \leq 2^L$. Finally, the payload is embedded. The length of the embeddable external payload is referred to as the effective carrier capacity $\mathbf{C}_e$ and it is computed as follows:

$$\mathbf{C}_e = (\lambda_L \times 2) - [\kappa + (L \times M)] \quad \text{bits}, \tag{5.1}$$

where κ is the size (in bits) of trimmed quotient parts in the augmented payload and $\kappa \geq \lambda_L$. Note that achieving $\mathbf{C}_e > 0$ implies that the following inequality must hold true:

$$\lambda_L \times 2 > \kappa + (L \times M). \tag{5.2}$$

Here, Inequality (5.2) depends on two factors: (a) λ_L, which depends on the chosen length for each imaginary codeword (i.e., L), and; (b) κ, which depends on the distribution of the imaginary codewords. If the distribution of the imaginary codewords is concentrated around a few arrangements (among the total of 2^L arrangements), then κ is small, and vice versa. Hence, choosing L that satisfies Inequality (5.2) plays the major role in controlling

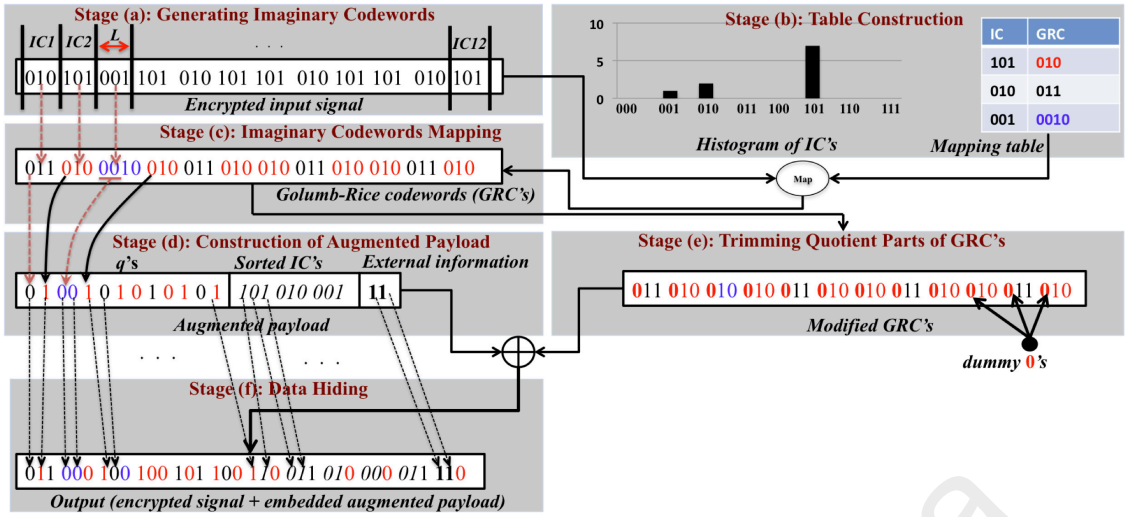


Figure 5.3: Example on data embedding in urDEED

the effective carrier capacity $\mathbf{C}_e$.

5.5 Example on Data Embedding/Extracting in urDEED

Figure 5.3 shows an example to walkthrough the proposed data embedding method. This example is based on a segment of random bitstream X shown in **Stage (a)** but the description is general enough to handle the entire encrypted content (i.e., sequence of bits). Here, the set of imaginary codewords (IC's) is generated by applying the universal parser, which partitions X in segments, each of 3 bits (i.e., $L = 3$), and $X_3 = \mathbb{D}(X, 3)$. In this example, $N = 36$ and hence $\lambda_3 = \frac{36}{3} = 12$ IC's.

In **Stage (b)**, the IC's are sorted according to their frequency of occurrences, starting with IC of the highest occurrence. Then, the mapping table (from IC's to GRC's) is constructed and this table is stored as part of the augmented payload in the actual implementation (as shown in **Stage (d)**). In this example, three GRC's are generated, each of length $n = 1$ for its remainder part. Since IC $= 101$ occurs 7 times (i.e., highest occurrence), it is assigned to the first GRC $= 010$. In similar fashion, IC$= 010$ occurs 4 times and hence it is assigned to GRC$= 011$. The longest GRC$= 0010$ is assigned to IC$= 001$ because it occurs only once.

In **Stage (c)**, the set of imaginary codewords are mapped into GRC's as stipulated

by the mapping table. In **Stage (d)**, the various components of the augmented payload are concatenated, which include the set of trimmed q's (generated in **Stage (e)**), the set of sorted IC's in the table and payload of two bits "11". Note that the length of GRC $= 0010$ is 4 bits, which is longer than $L = 3$ bits. Hence in **Stage (e)**, the quotient part of each GRC is trimmed and added to the augmented payload in order to preserve the original size of the input encrypted signal. Thus, $GRC_1 = 011$ and $q_1 = 0$, and $GRC_2 = 010$ and $q_2 = 1$, which is the logical inverse of q_1. For $GRC_3 = 0010$, $q_3 = 00$ are derived, which is the logical inverse of q_2, and so on. The derived q's are concatenated to form the first component of the augmented payload as shown in **Stage (d)**. Next, a dummy value of "0" is assigned to each modified (or fixed-length) GRC. Finally, in **Stage (f)**, data embedding is achieved by embedding the augmented payload in the set of fixed-length GRC's derived in **Stage (e)**. Here, each GRC accommodates two bits of the payload. These two bits are embedded by replacing the dummy "0" and the constant "1" in each modified GRC.

Note that the effective carrier capacity in this example is computed as follows: $\lambda_3 = 12$, $\kappa = 13$, $L = 3$ and $M = 3$. Hence, $\mathbf{C}_e = (12 \times 2) - [13 + (3 \times 3)] = 2$ bits. Here, $\mathbf{C}_e$ is of small amount because there are only a few IC's in the bitstream considered. For simplicity, data embedding is achieved in this example without coding L and M at the beginning of the augmented payload. Empirically, it is observed in the experimental results (Section 5.6) that maximum $\mathbf{C}_e$ is consistently achieved at $L = 3$ and hence the length of the side information is at most $L \times M = 3 \times 2^L = 24$ bits for this particular setting.

The extraction of the payload and restoration of the original encrypted signal in urDEED are almost the opposite of the embedding process. The same example is used to briefly walkthrough the decoding process:

(1) Encoded bitstream in **Stage (f)**, i.e., Output, is stored in the memory for reconstruction purposes.

(1.a) The original GRC's are reconstructed by examining the dummy 0's and b's of the set of GRC's obtained in **Stage (f)**. A change from 0 to 1 (or vice versa) marks the beginning of the quotient part for the next GRC where the length of zero-run (or one-run) indicates the original number of zeros in the quotient part of the current GRC. This is equivalent to performing the reversed steps of **Stage (f), (e+d)** and **(c)**. The process is repeated until $\mathfrak{n}$ zeros (from trimmed quotient parts) are restored.

(1.b) Next, the first two bits (i.e., originally the dummy '0' and termination bit 'b') from each of the remaining fixed-length GRC's are examined to extract the side information. After retrieving 24 bits (in the case of $L = 3$), the table in **Stage (b)** is re-generated by utilizing the extracted side information.

(1.c) The processes are repeated for the remaining GRC's to retrieve the actual payload.

(2) Based on the re-generated table, GRC's are losslessly re-mapped into IC's using the reverse steps from **Stage (c)** to **(a)**.

5.6 Experimental Results

As a proof of concept, the proposed method is implemented using C programming language. The performance of urDEED is verified from the following aspects: (1) interchangeability; (2) reversibility; (3) effective carrier capacity, and; (4) ability to preserve file-size. Since urDEED is applied to encrypted signal, the distortion of the signal due to data embedding (i.e., by applying urDEED) is irrelevant and hence not measured. However, it is verified that the embedded augmented payload can be perfectly extracted and the original input (encrypted) signal can be losslessly reconstructed. In addition, the reconstructed (encrypted) signal can be decrypted, which verified that the proposed urDEED is completely reversible. As mentioned earlier, there is no prior work operating in the encrypted domain that offers the same features. Hence, the performance of the proposed urDEED is compared to that of Zhang (2012) because it is the closest match in the current

literature. The following sub-sections detail the performances of urDEED.

5.6.1 Interchangeability

Interchangeability means that urDEED is applicable to any encrypted signal independently from the features of: (A) the original media of the signal, and; (B) the applied encryption scheme. To verify (A), urDEED is applied to encrypted signals which are originally three different media, namely, image, audio, and text. The image is Tiffany, which is a gray-scale raw image of 512×512 pixels, the audio is of Waveform Audio File Format (wave) of 2 seconds in length with bit rate of 1411Kbps, and the texts are two Microsoft Word© documents of size 12.7 and 292 KBytes.

To verify (B), each of the aforementioned media is encrypted by various encryption algorithms. The considered encryption schemes are Blowfish (Mousa, 2005), Advanced Encryption Standard (AES) (Kim, 2012), Rivest's cipher (RC) 2,4,6 (Gogniat et al., 2008), Data Encryption Standard (DES) (McLoone & McCanny, 2003), Triple-DES (Merkle & Hellman, 1981), Serpent (Najafi, Sadeghian, Saheb Zamani, & Valizadeh, 2004), Twofish (Su, Wuu, & Jhang, 2007), RSA (Sun, Wu, Ting, & Hinek, 2007), CAST (Adams, Heys, Tavares, & Wiener, 1999), Information Concealment Engine (ICE) (Rompay, Knudsen, & Rijmen, 1998), MARS (Mohamed, El-Etriby, & Abdulkader, 2012), MISTY (Wen & Yin, 2010) and Tinny Encryption Algorithm (TEA) (Moon, Hwang, Lee, Lee, & Lim, 2002). In addition to these methods, hybrid cascade encryption algorithms, such as AES-TwoFish-Serpent, are also applied. The output ciphertext of each encryption scheme is manipulated by urDEED to embed the augmented payload.

Table 5.2 records the experimental results using urDEED to embed the augmented payload into the encrypted image, audio and texts. Results show that, generally, urDEED is interchangeably applicable to a signal encrypted by various encryption methods because the program is able to execute to its completion and the effective carrier capacity

of value $\mathbf{C}_e > 0$ is achieved. This suggests that the limitation of interchangeability in the existing methods of Zhang (2012), Chen, Zhang, Ma, and Yu (2013), Alattar (2004) is addressed by the proposed urDEED.

For fair comparison purposes, external information is embedded into two images (i.e., Lenna and Man of dimensions 512×512 pixels) encrypted using Zhang's XOR operations described in (Zhang, 2012) and the results are recorded in the second column of Table 5.3. In terms of interchangeability, Zhang's method is restricted to one encryption scheme (i.e., XOR-based encryption proposed in (Zhang, 2012)) while urDEED is universally applicable to any encryption scheme. Also, Zhang's method (Zhang, 2012) is only applicable to image stored in the spatial domain while the proposed method can be applied to any signal coded in any domain.

5.6.2 Reversibility

Reversibility is the ability to perfectly reconstruct the original encrypted signal. The reversibility in urDEED is due to the utilization of the lossless GRC for data embedding, and this reversible functionality is verified by using various media and various encryption algorithms. On the other hand, Zhang's method (Zhang, 2012) is not able to guarantee a perfect reconstruction of the original image due to the dependency of correlation among neighboring pixels.

5.6.3 Effective carrier capacity

The effective carrier capacity $\mathbf{C}_e$ depends on the length L as shown in Eq. (5.1). To find the optimum L, which achieves the maximum $\mathbf{C}_e$, urDEED is applied to Lenna image (512 × 512 grayscale, raw) encrypted by Blowfish (Hu et al., 2009) using different L values as the representative case. Table 5.1 shows the resulting $\mathbf{C}_e$ at each L. It is observed that $\mathbf{C}_e$ at L=4 is 91323 bits, which is lower than its counterpart at L=3. $\mathbf{C}_e$ decreases to 3224 bits at L=5. It is observed that $\mathbf{C}_e < 0$ for $L \in \{6,7,8\}$, which indicates

Table 5.1: Effective carrier capacity of urDEED for various L in encrypted Lenna image

L	3	4	5	6	7	8
$\mathbf{C}_e$	352452	91323	3224	-30866	-44394	-49105

that there is no venue for embedding extra information in the encrypted image. This is due to the expansion of side information including the increment in the length of the set of trimmed q's and expansion of the table. Hence, all available data embedding venues are utilized to accommodate the side information. This poor performance at $L > 3$ is generally observed for all test signals considered. Thus, based on the results, it is concluded that the optimum length for L is 3 bits where the maximum $\mathbf{C}_e$ is achieved, in general, for all types of signal. However, it should be noted that this conclusion is limited by applying GRC's.

Table 5.2 presents the effective carrier capacity ($\mathbf{C}_e$) obtained by applying urDEED to various encrypted signals. Here, the results are collected by using $L = 3$. Generally, in the case of image, $\mathbf{C}_e$ is in the range of $[334709, 401431]$ bits. Such variation in $\mathbf{C}_e$ is due to the differences in statistical features resulting from the application of various encryption algorithms to the image. Table 5.2 also shows the effective embedding rate in terms of bpb, which is defined as the ratio of $\mathbf{C}_e$ to the original size of the encrypted signal. It is observed that the effective embedding rate is between 0.167 and 0.169 bpb. In the case of audio, $\mathbf{C}_e$ is in the range of $[472252, 518858]$ bits or in other words, between 0.167 and 0.184 bpb. The results are consistently close to that of applying urDEED to image.

Similar trend is also observed in the case of text, in which two files of different sizes are considered. The consideration of different sizes is to verify the effect of the original size on the performance of urDEED. It is observed that when the smaller size is considered, $\mathbf{C}_e$ is in the range of $[17630, 19706]$ bits. For the larger size, $\mathbf{C}_e$ is in the range

Table 5.2: Effective carrier capacity of urDEED when applied to different media encrypted by using various encryption schemes

	Image		Audio		Text	
Encryption Method	$\mathbf{C}_e$	bpb	$\mathbf{C}_e$	bpb	$\mathbf{C}_e$	bpb
Blowfish-448	335809	0.168	474112	0.168	17703	0.170
AES-128	335971	0.168	473672	0.168	17650	0.170
AES-192	335216	0.168	474659	0.168	17956	0.173
AES-256	336221	0.168	472252	0.167	17878	0.172
RC2-1024	336190	0.168	473750	0.168	17891	0.172
RC4-2048	335574	0.168	474963	0.168	17929	0.172
RC6-2048	336445	0.168	476807	0.169	17864	0.172
DES-56	335700	0.168	473870	0.168	17709	0.170
TDES-256	336216	0.168	473393	0.167	17743	0.171
Serpent-256	400422	0.167	472349	0.167	400316	0.167
Twofish-256	400621	0.167	473286	0.167	17630	0.169
Blowfish-AES-256	401167	0.168	474274	0.168	401370	0.168
Serpent-Flowfish-AES-256	401137	0.168	473432	0.168	400755	0.168
AES-Serpent-256	401431	0.168	474554	0.168	400309	0.167
AES-TwoFish- Serpent-256	400916	0.168	473057	0.167	400971	0.168
Serpent-TwoFish-256	399502	0.167	473573	0.168	400531	0.167
RSA-1024	367767	0.168	518858	0.184	19706	0.189
CAST-256	334709	0.167	473877	0.168	17994	0.173
ICE-64	336440	0.168	475146	0.168	17771	0.171
MARS-1248	336901	0.169	475569	0.168	17977	0.173
MISTY-128	335999	0.168	473432	0.168	18060	0.174
TEA-128	336019	0.168	474186	0.168	18084	0.174

of $[400309, 401370]$ bits. Hence, $\mathbf{C}_e$ is proportional to the size of the input signal because $\mathbf{C}_e$ depends on λ_L, which depends on the total number of bits N as shown in Eq. (3.4). Nevertheless, the effective embedding rate is approximately the same in both cases. In the case of the smaller file-size document, the effective embedding rate ranges from 0.169 to 0.189 bpb, while this range is between 0.167 and 0.168 bpb in the case of the larger file-size document. This suggests that urDEED offers a consistent effective embedding rate regardless of the type of media and the encryption algorithm in use. These results also suggest that urDEED is able to embed payload into an encrypted signal without causing file-size increment or data loss.

The second column of Table 5.3 compares the average effective embedding rate between Zhang's method (Zhang, 2012) and the proposed urDEED for two images (i.e.,

Lenna and Man). Here, the average effective embedding rate of (Zhang, 2012) is 0.006 bpb, which is based on the best reported results in (Zhang, 2012) at the expense of sacrificing the reversibility property (i.e., perfect reconstruction is not possible at this embedding rate). However, urDEED achieves, on average, 0.173 bpb, which outperforms the carrier capacity of Zhang's by 28.8% and urDEED is completely reversible.

5.6.4 File-Size Preserving

It is verified that the input signal (ciphertext) and the processed signal (chiphertext with extra payload embedded) are of the exact same file-size. This is an expected outcome because urDEED trims the quotient part of each GRC and embeds them along with the payload as detailed in Section 5.4.3. Although Zhang's method (Zhang, 2012, 2011) and its improvement (Hong et al., 2012) also preserve the file-size to that of the input encrypted image, it is clear that the proposed urDEED offers more features. These features include higher effective carrier capacity, applicability across different media, applicability across different encryption algorithms, and complete reversible functionality. Table 5.3 summarizes the functional comparison between Zhang's method and the proposed urDEED.

Table 5.3: Functional comparison between Zhang 's method (Zhang, 2012) and the proposed urDEED

Method	Average bpb	Applicable to any media?	Applicable to any encryption scheme?	File-Size Preserving?	Reversible?
Zhang (Zhang, 2012)	0.006[1]	No (image only)	No	Yes	Conditional
urDEED	0.173	Yes	Yes	Yes	Always

[1] This is the highest carrier capacity reported in (Zhang, 2012) while sacrificing reversibility. For the case of perfect reversibility, the highest carrier capacity reported is 0.004 bpb.

5.7 Summary

The inconsistent carrier capacity of uREADS was overcame by urDEED by handling the excessive data in GRC's efficiently. Also, urDEED overcame the problem of manual tuning the parameters of L and r. In particular, empirically, it is observed that the consistent setting of $(L,r) = (3,1)$ results in an average effective carrier capacity of ~ 0.169 bpb. In addition, urDEED achieved the following: (1) universally applicable to any encrypted signal and operational solely in the encrypted domain; (2) completely reversible, and; (3) file-size preserving while hosting external information in the encrypted signal.

CHAPTER 6

DATA EMBEDDING IN RANDOM DOMAIN

6.1 Overview

In previous chapter, it is shown that urDEED is interchangeably applicable to any signal with consistent high carrier capacity. However, this method depends on the statistical features of the signal to define redundancy. Hence, when redundancy is low, urDEED is infeasible because the size of side information (generated for reversibility) occupies all venues for data embedding hence the embedding of a payload is impossible. In this chapter, a universal data embedding method based on histogram mapping called DeRand (Data embedding in Random domain) is proposed. DeRand theoretically defines redundancy in any digital signal by applying the universal parser (Section 3.3.3) so that high entropy random signals can certainly be utilized for data embedding. First, DeRand recursively parses a random signal into a set of tuples each of certain length until there exist some tuples of zero occurrences, i.e., count$=0$. Then, tuple association is performed where a tuple of count>0 is associated with a tuple of count$=0$. In particular, a tuple of count>0 is mapped to a pre-assigned tuple of count$=0$ to embed "1", while the tuple is left unmodified to embed "0". DeRand is universal, reversible, applicable to any random signal and scalable in terms of carrier capacity and signal quality. Experimental results show that DeRand achieves carrier capacity up to 4909 bits in a random signal of the size 256 Kbytes, in other words, 0.0023 bpb. In addition, the quality of the processed signal ranges from 0.0075 to 395.67 in terms of MSE.

6.2 Introduction

Multimedia data is massively generated nowadays thanks to the advanced yet low cost capturing and storage technologies. In addition to the conventional network traffic such as web accesses, email communication, and transmission of e-commerce data, these multimedia data are communicated and shared among various users across continents. The advent of innovative social network services at no cost and the deployment of online content store further multiplied the utilization of network. For example, it is reported that $240,000$ photos and 100 hours of videos are uploaded each minute to Facebook (IACP, 2014) and Youtube (Statistics, 2014), respectively. Thus, the communication channels, including Internet and cellular phone networks, convey a mixed streams of information (appearing in various formats and coding structures) sent from multiple sources. These streams appear as random data when being transmitted over the communication channels. Generally, any unintelligible data can be considered as a random data (or random signal hereinafter) from the perspective of a third party who has no access to its original (intelligible) form. These random data include encrypted signal, records in database, and data uploaded to cloud-storage, which need to be managed for efficient utilization of the communication bandwidth and storage space. Here, for digital data management purposes, data embedding technologies provide conveniences to achieve annotation, authentication, watermarking, etc. However, fundamentally, data embedding is a feature-dependent process, where features of a host are modified in certain domain and coding structure in order to embed data (Lei et al., 2011; Cheddad et al., 2010; Xu et al., 2011; Lusson et al., 2013; Wong et al., 2007). For example: (a) the methods proposed by Luo et al. (2011), Chang and Kieu (2010) and Yang, Chung, and Liao (2012) manipulate the features of the host image in the spatial domain; (b) the methods proposed by Chang, Lin, Tseng, and Tai (2007), Chang, Chen, and Chung (2002) and Lin and Lin (2009) process the

host image in the frequency domain; (c) the method proposed by Wong, Tanaka, Takagi, and Nakajima (2009) is restricted to compressed video; (d) the method proposed by Malik, Ansari, and Khokhar (2007) is applicable only to audio; (e) the method proposed by Borges, Mayer, and Izquierdo (2008) embeds data in text, and; (f) the methods proposed by Zhao et al. (2010), Zhang (2012), Hong et al. (2012) and Zhang (2011) are restricted to encrypted signals. Therefore, the interchangeability among most data embedding methods is generally restricted by its domain or media with certain features. In other words, the definition of feature is necessary to achieve data embedding by these methods. On the other hand, it is technically challenging to extract features from a random signal, which is unintelligible. Generally, data embedding in the random domain has not been considered in the current literature.

Ong, Wong, and Tanaka (2014) put forward a reversible data embedding method that offers scalability in terms of carrier capacity and perceptual quality in image. In this method, the ability to control the perceptual quality (in the host image) is utilized to achieve perceptual encryption (viz., image scrambling), where the semantic of the image is intentionally masked by the designed substitution operation. Here, the payload is embedded into the image, along with the side information required to restore the original image. The substitution operation is achieved by modifying the histogram of the image. In particular, the probability $P(x)$ of each intensity level x in the image is analyzed. Basically, x is classified into two sets based on its probability, namely, $G_1 = \{x : P(x) > 0\}$ and $G_2 = \{x : P(x) = 0\}$. Hence, to embed "1", $x \in G_1$ is mapped to $y \in G_2$, and to embed "0", the value $x \in G_1$ is left unmodified, i.e., no mapping is performed. This method successfully overcomes the underflow and overflow problems in the conventional histogram shifting methods (Ni et al., 2006; Jung, Ha, & Ko, 2011). However, it fails to embed data when the entire range of intensity levels is occupied, i.e., $P(x) > 0$ for all $x \in [0, 2^L - 1]$, where it is assumed that the parsing length, i.e., L, is the pixel bit depth. In such case,

Ong et al. (2014) partition the image into non-overlapping blocks and handles each block individually. Furthermore, this method is verified only with images, and it depends on the statistical features of the image in the spatial domain. Hence, it is not feasible to be applied for handling a random signal, which consists of high entropy data.

In this chapter, a universal, reversible and scalable data embedding method that is applicable to any random signal based on histogram mapping is proposed. Unlike the traditional histogram mapping methods, the proposed method can *certainly* define redundancy in any given signal by applying the universal parser (Section 3.3.3), even when all bins in the histogram are occupied, i.e., $P(x) > 0$ for all $x \in [0, 2^L - 1]$. In particular, the universal parser recursively partitions the signal into segments using increasing length. Theoretically, it is proven that as the length of the segments increases, the probability of defining redundancy (i.e., $\exists x' | P(x') = 0$) increases. In other words, the amount of redundancy changes as the length of the segments changes. This change in redundancy is exploited to embed a payload into the random signal by histogram mapping. The proposed data embedding method achieves the following properties: (1) applicable to any random signal; (2) reversible, and; (3) scalablel in terms of carrier capacity and progressive quality degradation. Generally, the proposed method is the first of its kind to achieve reversible data embedding in random signal and the first universal data embedding method that offers scalability in carrier capacity and quality degradation.

6.3 Applications of Data Embedding in Random Domain

It is assumed that any unintelligible signal which losses its semantic due to intentional encryption or un-intentional loss of access to the original format is called a random signal. This includes signals generated by a random number generator. Hence, random signals occupy a significant percentage among all digital signals exchanged through the communication channels and those stored in the storage systems. Unfortunately, the ran-

dom signals are, by and large, still left unexplored in the applications of information processing. For that, data embedding in the random domain contributes in filling up this deficiency through some applications, such as data hiding for covert communication (i.e., steganography) in random signals. On the other hand, reversible data embedding enables the insertion of external information while preserving the file size of the original signal. This feature can be exploited to reduce the bandwidth required to communicate/store the random signal by embedding (or hosting) one segment of a random signal into another random signal. Hence, compression is gained and the total size of the random signals is reduced. The reduction in total signal size is a significant contribution in bandwidth utilization, especially for massive data transmission and storage purposes. As another application, a third party (e.g., a cloud administrator who has no access to the original format of the data) needs to embed data in random signals for annotation and management purposes, which include the insertion of a hash value in its corresponding random signal for integrity checking. The aforementioned potential applications justify the need to consider data embedding in the random domain.

6.4 Theoretical Study on Histogram Mapping

In this section, data embedding by histogram mapping is studied theoretically. Recall that the universal parser (Section 3.3.3) parses the set X into X_L by the ordering function $\mathbb{D}(X,L)$, which partitions X into tuples each of length L as follows:

$$\mathbb{D}(X,L) = X_L = \{T_1^L, T_2^L, \cdots, T_{t=\frac{N}{L}}^L\}, \tag{6.1}$$

where T_t^L denotes the t-th L-tuple that appears in X. The set X_L satisfies $X_L \subseteq \mathbb{A}_L$ where $\mathbb{A}_L$ is the set of all possible arrangements of alphabets with length L, which has the cardi-

nality:

$$|\mathbb{A}_L| = \underbrace{|\mathbb{A}| \times |\mathbb{A}| \times \cdots |\mathbb{A}|}_{L} = |\mathbb{A}|^L \tag{6.2}$$

Definition 5. *The complement set $\dot{X}_L$ of X_L is defined as follows:*

$$\dot{X}_L = \mathbb{A}_L \backslash X_L, \tag{6.3}$$

where the cardinality $|\dot{X}_L| = |\mathbb{A}_L| - |X_L|$ holds true □

Based on Definition 5, $\mathbb{A}_L = X_L \cup \dot{X}_L$.

Corollary 6. *Given the set $X_L \subseteq \mathbb{A}^L$, the probability of occurrences of the elements in X_L is computed as:*

$$P(X_L) = \sum_{t=1}^{|X_L|} P(T_t^L \in X_L) = \frac{|X_L|}{|\mathbb{A}^L|}. \tag{6.4}$$

Proof. Since the elements of X are uniformly distributed, the tuples in X_L are also uniformly distributed. As $X_L \subseteq \mathbb{A}_L$, each tuple assumes the constant probability of $P(T_t^L) = \frac{1}{|\mathbb{A}_L|}$. Hence, $\sum_{t=1}^{|X_L|} \frac{1}{|\mathbb{A}^L|} = \frac{|X_L|}{|\mathbb{A}^L|}$ □

Corollary 7. *Following the previous corollary,*

$$P(\dot{X}_L) = 1 - \frac{|X_L|}{|\mathbb{A}^L|} \quad \square \tag{6.5}$$

The histogram of $\mathbb{A}_L$ is denoted by the symbol $\mathbb{H}(\mathbb{A}_L)$ and defined as:

$$\mathbb{H}(\mathbb{A}_L) = \{h_{T_t^L}\}_{t=1}^{|\mathbb{A}^L|}, \tag{6.6}$$

where $h_{T_t^L}$ is the count of the tuple T_t^L in $\mathbb{A}_L$. To facilitate the discussion without lost of

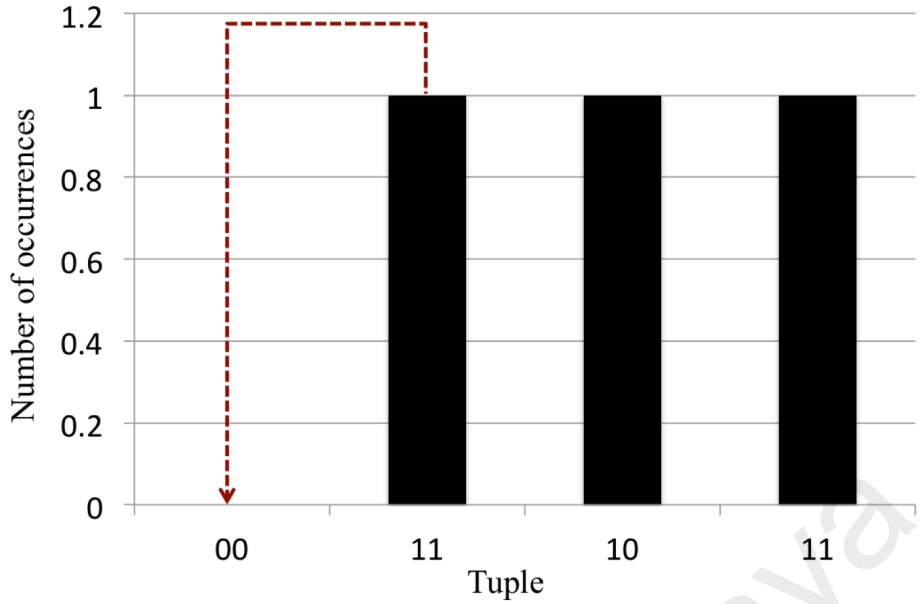


Figure 6.1: Since "00" does not exist, it can be utilized for data embedding using histogram mapping. For example, "11" is mapped to "00" to embed "1", and the value remains intact (i.e., no mapping is performed) to embed "0".

generality, it is assumed that $\mathbb{A}_L$ consists of uniformly distributed elements where $h_{T_t^L} = 1 \; \forall T_t^L \in \mathbb{A}_L$. That is, each tuple occurs exactly once in $\mathbb{A}_L$.

Let $\mathbb{A} = \{0,1\}$ be the set of alphabets, $d \in \{0,1\}$ be the 1-bit payload to be embedded in a tuple in X_L, and $\dot{X}_L \neq \phi$. The process of data embedding using histogram mapping is achieved by selecting a tuple $T_t^L \in X_L$ and a tuple $\dot{T}_t^L \in \dot{X}_L$, followed by the mapping:

$$T_t^L \leftarrow \begin{cases} \dot{T}_t^L, & \text{if } d = \text{"1"}, \\ T_t^L, & \text{otherwise.} \end{cases} \tag{6.7}$$

Figure 6.1 presents an example of data embedding using the proposed histogram mapping. Here, $X_2 = \{01,10,11\}$, $\dot{X}_2 = \{00\}$ and the frequency of occurrences $\mathbb{H}(\mathbb{A}_2) = \{h_{00} = 0, h_{01} = 1, h_{10} = 1, h_{11} = 1\}$. As an illustration, the tuples $T_1^2 = 01$ and $\dot{T}_1^2 = 00$ are selected from the sets X_2 and $\dot{X}_2$, respectively, for data embedding using histogram mapping. The tuple $T_1^2 = 01$ is mapped to $\dot{T}_1^2 = 00$ to embed "1", but left unmodified to embed "0". In other words, X_2 and $\dot{X}_2$ encodes "1" and "0", respectively. Note that $T_1^2 = 01$ and $\dot{T}_1^2 = 00$ should be saved as side information for perfect restoration of the

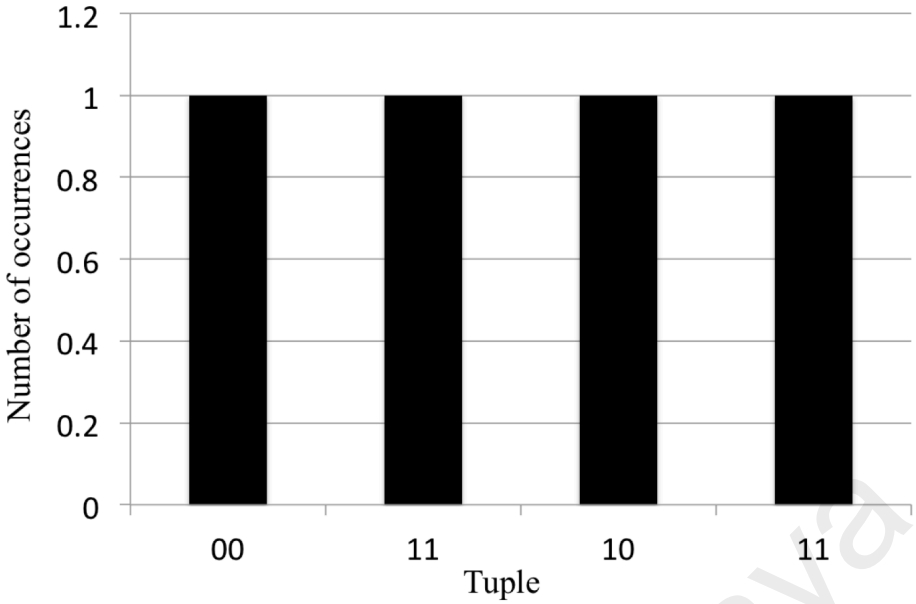


Figure 6.2: All $2^2 = 4$ bins (possible arrangements) are occupied in the histogram. Hence, data embedding using histogram mapping cannot be achieved directly with this particular histogram.

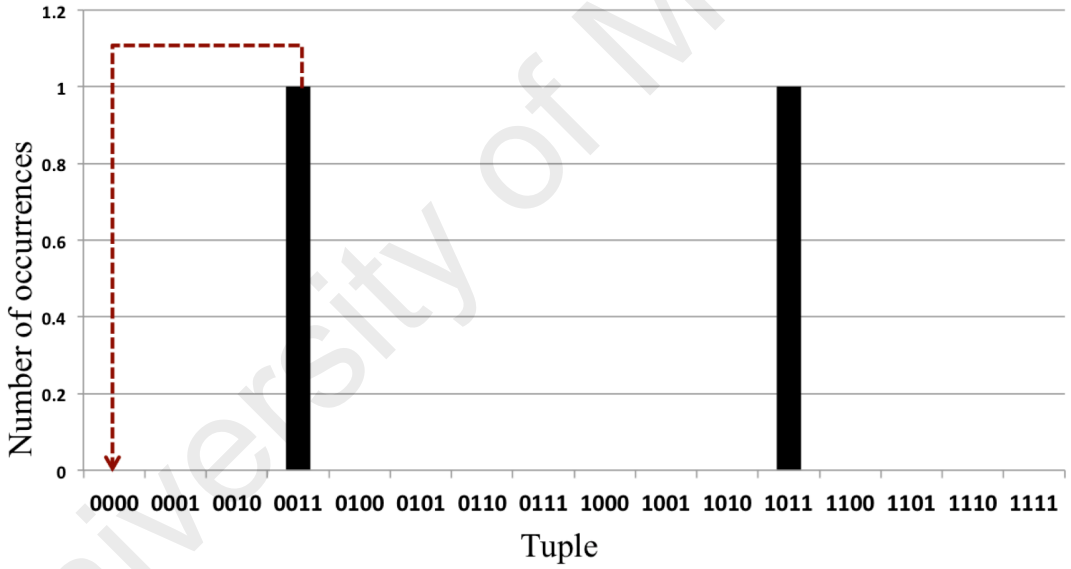


Figure 6.3: Using the universal parser, the elements in Figure 6.2 are parsed at $L = 4$. Data embedding using histogram mapping can be achieved similar to Figure 6.1.

original X_2.

As another example, Figure 6.2 shows the histogram $\mathbb{H}(\mathbb{A}_2) = \{h_{00} = 1, h_{01} = 1, h_{10} = 1, h_{11} = 1\}$, where the probability of all tuples are greater than 0. Thus, in this example, the cardinality of $\dot{X}_2$ is zero. However, there must be at least one tuple $\dot{T}_t^L \in \dot{X}_L$ (i.e., $|\dot{X}_L| \geq 1$) in order to achieve data embedding by histogram mapping. In other words, data embedding by histogram mapping fails when $|\dot{X}_L| = 0$, which is the case in the example

shown in Figure 6.2. To overcome this problem, X_L is redefined to vacate room in the histogram for mapping. The following theorem shows that increasing L in the universal parser (Section 3.3.3) increases the probability to achieve $|\dot{X}_L| > 0$.

Theorem 3. *Given two sets $\dot{X}_L$ and $\dot{X}_{L+K}$, the following inequality holds true for $K, L > 0$:*

$$P(\dot{X}_L) < P(\dot{X}_{L+K}). \tag{6.8}$$

Proof. Since $\mathbb{A}_L$ and $\mathbb{A}_{L+K}$ are finite sets, the following holds true:

$$\frac{1}{|\mathbb{A}^L|} > \frac{1}{|\mathbb{A}^{L+K}|}. \tag{6.9}$$

On the other hand, since X is also a finite set, there are more segments in X_L when compared to X_{L+K} and hence the following inequality holds true:

$$|X_L| > |X_{L+K}|. \tag{6.10}$$

Therefore,

$$|X_L| \times \frac{1}{|\mathbb{A}^L|} > |X_{L+K}| \times \frac{1}{|\mathbb{A}^{L+K}|}. \tag{6.11}$$

Hence, based on Corollary 6:

$$P(X_L) > P(X_{L+K}) \tag{6.12}$$

Now,

$$-P(X_L) < -P(X_{L+K}) \tag{6.13}$$

and,

$$1 - P(X_L) < 1 - P(X_{L+K}) \tag{6.14}$$

Based on Corollary 7, Eq. (6.14) can be re-expressed as follows:

$$P(\dot{X}_L) < P(\dot{X}_{L+K}) \tag{6.15}$$

□

Informally, Theorem 3 states that the probability of the elements in the complement set $\dot{X}_L$ increases when the original signal X is parsed at a larger L.

Based on Theorem 3, the histogram mapping in the example shown in Figure 6.2 is achievable by increasing the value of L from 2 to 4. Hence, the newly obtained histogram (viz., frequency of occurrences) based on $L = 4$ (see Figure 6.3) is $\mathbb{H}(\mathbb{A}_4) = \{h_{0000} = 0, h_{0001} = 0, h_{0010} = 0, h_{0011} = 1, h_{0100} = 0, h_{0101} = 0, h_{0110} = 0, h_{0111} = 0, h_{1000} = 0, h_{1001} = 0, h_{1010} = 0, h_{1011} = 1, h_{1100} = 0, h_{1101} = 0, h_{1110} = 0, h_{1111} = 0\}$. As expected, many tuples from $\mathbb{A}_4$ are of zero count in the histogram of X_4 as shown in Figure 6.3. Thus, the cardinality of the set $\dot{X}_4$ is 14. Hence, for the set $X = \{00, 01, 10, 11\}$, histogram mapping is feasible at X_4 but impossible at X_2. The next theorem shows that data embedding by histogram mapping can be *certainly* achieved in X when it is parsed at the maximum length $L = N$, i.e., the limiting case of $X_{L=N}$.

Theorem 4. *A finite set X with N elements induces the set X_N using Eq. (6.1) (i.e., $\mathbb{D}(X,N)$) and the inequality $|\dot{X}_N| \geq 1$ holds true.*

Proof. Since the cardinality of the set of alphabets $|\mathbb{A}| \geq 2$, the cardinality below holds

true:

$$|\mathbb{A}_N| = |\mathbb{A}|^N \geq 2^N. \tag{6.16}$$

Since, $|\dot{X}_N| = |\mathbb{A}_N| - |X_N|$ and the cardinality $|X_N| = 1$ (based on Eq. (6.1)), the following holds true:

$$|\dot{X}_N| = |\mathbb{A}_N| - 1 \geq 2^N - 1 \geq 1 \tag{6.17}$$

for the fact that N is an integer with $N > 0$. Therefore, $|\dot{X}_N| \geq 1$ □

Theorem 4 ensures that histogram mapping is always viable in any set X if the set is parsed at length $L = N$ to X_N using Eq. (6.1). Informally, this theorem shows that the histogram mapping can be universally applied to any discrete signal X regardless of its media, domain, underlying coding structure and statistical features.

Theorem 5. *Given two sets X_L and X_{L+K}, the following inequality holds true for $L, K > 0$:*

$$|\dot{X}_L| < |\dot{X}_{L+K}| \tag{6.18}$$

Proof. Since

$$|\mathbb{A}_L| = \mathbb{A}^L < \mathbb{A}^{L+K} = |\mathbb{A}_{L+K}|, \tag{6.19}$$

then,

$$|\mathbb{A}_L| - |X_L| < |\mathbb{A}_{L+K}| - |X_L| \tag{6.20}$$

Based on Eq. (6.10),

$$|\mathbb{A}_L| - |X_L| < |\mathbb{A}_{L+K}| - |X_L| < |\mathbb{A}_{L+K}| - |X_{L+K}| \tag{6.21}$$

Thus, by transitivity:

$$|\mathbb{A}_L| - |X_L| < |\mathbb{A}_{L+K}| - |X_{L+K}| \tag{6.22}$$

Therefore, by Definition 5:

$$|\dot{X}_L| < |\dot{X}_{L+K}| \tag{6.23}$$

□

Informally, Theorem 5 indicates that increasing L to $L+K$ results in $|\dot{X}_{L+K}| > |\dot{X}_L|$. In other words, there are more tuples in $\dot{X}_{L+K}$ than $\dot{X}_L$ that can be utilized for histogram mapping.

6.5 The Proposed DeRanD

In this section, the proposed DeRanD (Data embedding in Random Domain) method is presented. Figure 6.4 presents the schematic diagram of the proposed DeRand. In Stage 1, the input signal X is parsed by the universal parser (Section 3.3.3) to generate X_L. In Stage 2, the tuple association is performed, which includes the construction of histograms for X_L and $\dot{X}_L$. In Stage 3, histogram mapping is applied based on the input payload D. The following subsections further detail the operations involved in each stage.

6.5.1 Parsing by using Universal Parser

The input signal X is parsed by using the universal parser (Section 3.3.3) defined in Eq. (6.1). Here, the user is prompted to select an initial value of L. However, if the selected L results in $|\dot{X}_L| = 0$, the value of L is increased to $L \leftarrow L+1$ (based on Theorem 3) and the parsing process is performed again. In other words, L is increased repeatedly until $|\dot{X}_L| > 0$ is achieved. This stage is captured by steps 1 to 10 in Algorithm 3. Ideally, the initial value of L is unity. However, it is less probable to obtain $|\dot{X}_L| > 0$ with a small value

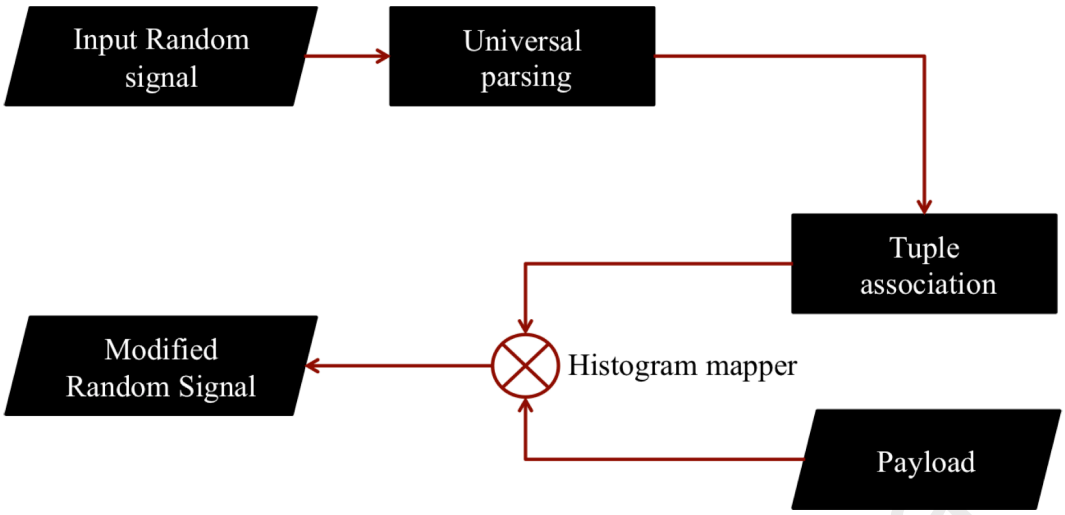


Figure 6.4: The schematic diagram of the proposed DeRand.

of L. Hence, it is recommended to initiate the algorithm at $L = 7$ based on the empirical findings presented in Section 6.6.

6.5.2 Tuples Association

Once the signal X is parsed at a length L such that $|\dot{X}_L| > 0$, the process of tuple association takes part. This includes the construction of two sets, namely, A_L and B_L, where $1 \leq |A_L|, |B_L| \leq \min\{|X_L|, |\dot{X}_L|\}$. The set $A_L = \{T^L_{P(i)}\}^{\rho}_{i=1} \subseteq X_L$ consists of ρ tuples that are of the highest occurrences in $\mathbb{H}$, where ρ is a positive integer such at $\rho \leq \hat{\rho}$ for $\hat{\rho} = \min\{|X_L|, |\dot{X}_L|\}$. Here, $P(i)$ is the index sorted accordingly to the probability of occurrence of $T^L_{P(i)}$ in $\mathbb{H}$ such that $T^L_{P(1)}$ has the highest occurrence among the tuples in X_L, followed by $T^L_{P(2)}$, and so forth. On the other hand, the set $B_L = \{\dot{T}^L_i\}^{\rho}_{i=1} \subseteq \dot{X}_L$ consists of any ρ tuples from the set $\dot{X}_L$. Next, a tuple $T^L_{P(i)} \in A_L$ is associated with a unique tuple $\dot{T}^L_i \in B_L$. For example, if $X_2 = \{00, 11\}$, then these two tuples are associated with the two tuples in $\dot{X}_2 = \{01, 10\}$. A possible association is “00" with “01", and “11" with “10". The process of tuple association is achieved by invoking steps (11) to (12) in Algorithm 3. This association is utilized for data embedding as detailed in the next sub-section.

For scalability purposes, $\rho \in [1, \hat{\rho}]$ is selected by the user. Here, as ρ increases, more tuples from X_L are utilized for data embedding and hence the carrier capacity increases.

However, increasing ρ causes more distortion (e.g., sum of bit-wise absolute difference) in the modified signal. Therefore, there is a trade-off between the carrier capacity and distortion, similar to the conventional data embedding method.

6.5.3 Data Embedding

The process of data embedding commences by reading a tuple T_t^L. Then, it is verified if $T_t^L = T_{P(i)}^L \in A_L$ holds true for $1 \leq t \leq \frac{N}{L}$. Next, to embed $d =$ "1" $\in D$, the tuple T_t^L is mapped to $\dot{T}_i^L$, where D is the payload in binary representation. In order to embed $d =$ "0" $\in D$, no mapping is performed. This process continues until all the $\frac{N}{L}$ tuples of X_L are mapped or when all bits in D are embedded. The process of tuple association is achieved by invoking steps (13) to (25) in Algorithm 3. Note that N, L, ρ, A_L and B_L should be stored as the side information for perfect reconstruction of the original signal X_L. For the current implementation, the signal X is concatenated to the side information $\mathbf{S}$, i.e., $X \leftarrow \mathbf{S} \oplus X$. For file-size preserving purposes, $|\mathbf{S}| = N + L + \rho + |A_L| + |A_L|$ bits of the original input signal, i.e., the number of bits occupied by $\mathbf{S}$, is removed and embedded as part of the payload.

6.5.4 Extraction of Payload and Restoration of X

During decoding, payload extraction and reconstruction of the original signal X are performed by first reading N, L, ρ, A_L and B_L, which are the stored side information as discussed in Section 6.5.3. Next, $\rho \times 2$ associated tuples are defined by reading each tuple T_t^L and its corresponding (viz., associated) tuple $\dot{T}_t^L$ from A_L and B_L, respectively. The modified signal X' is then segmented into tuples each of L bits, and each tuple τ is analyzed. In particular, if $\tau = T_t^L \in A_L$, then "0" is extracted. On the other hand, if $\tau = \dot{T}_t^L \in B_L$, then "1" is extracted and this tuple is restored to its associated T_t^L. When $\tau \notin A_L$ and $\tau \notin B_L$, it implies that no information is embedded. Regardless of the membership of τ, the decoder continues to read the next tuple that follows and repeat the aforementioned

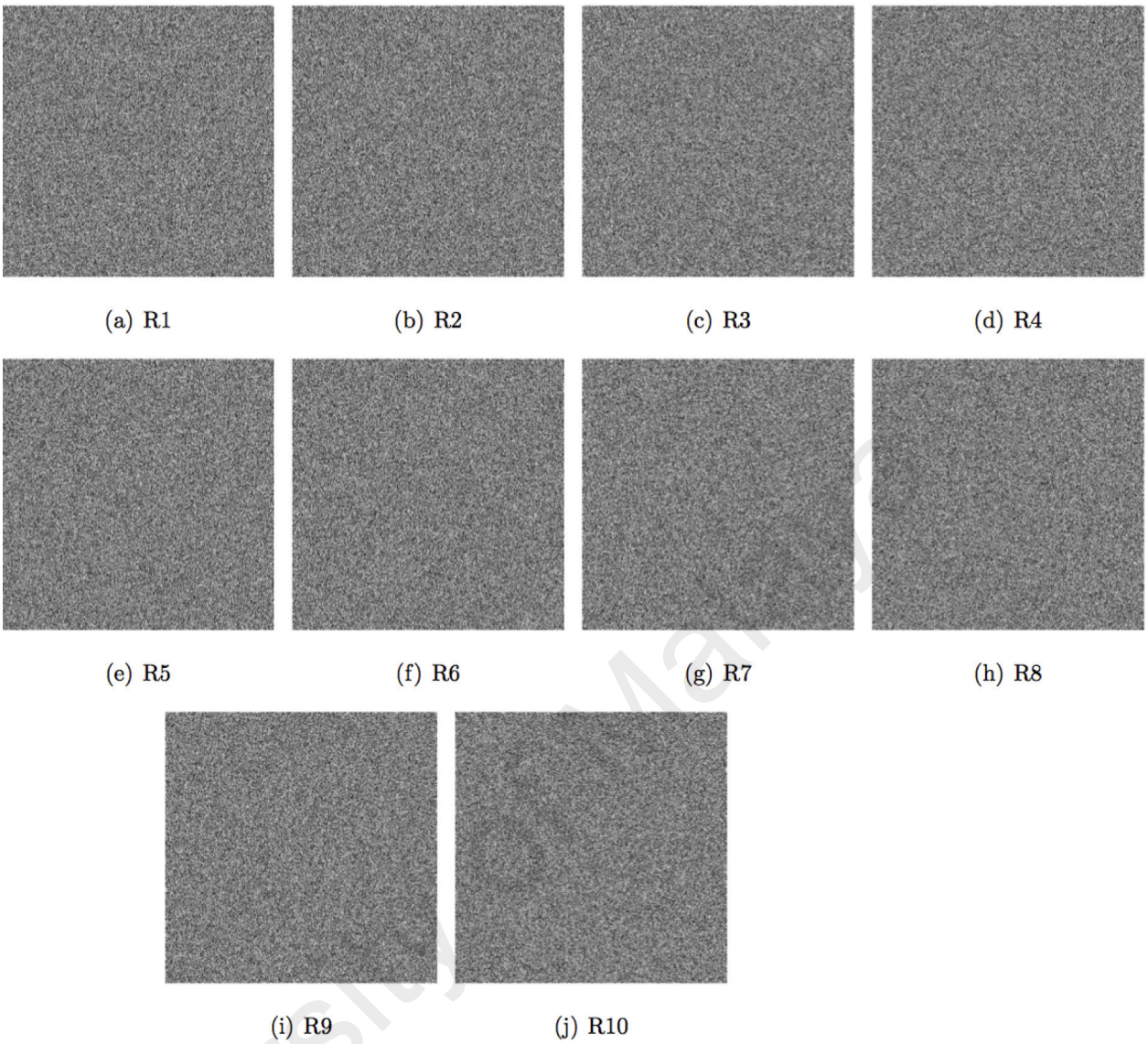
Algorithm 3 The Embedding Process of DeRand

Let the user pick $L \in [1, N]$
Set $i \leftarrow 1$ and $t \leftarrow 1$
Generate $X_L = \mathbb{D}(X, L)$
Construct $|\dot{X}_L|$
if $|\dot{X}_L|$=0 **then**
    $L \leftarrow L + 1$
    Go to Step 2
else
    Go to Step 11
end if
Let the user pick $\rho \in [1, \hat{\rho}]$
Construct the sets $A_L = \{T^L_{P(i)}\}^{\rho}_{i=1} \subseteq X_L$ and $B_L = \{\dot{T}^L_i\}^{\rho}_{i=1} \in \dot{X}_L$
Read one tuple T^L_t from X_L
if $T^L_t = T^L_{P(i)} \in A_L$ **then**
    Read one bit d from payload D
    if $d = 1$ **then**
        $T^L_t \leftarrow \dot{T}^L_i$
    end if
end if
$t \leftarrow t + 1$
if $t > \frac{N}{L}$ **then**
    Halt
else
    Go to Step 13
end if

analysis until all $\frac{N}{L}$ tuples are considered.

6.6 Experimental Results and Discussion

In this section, the performance of DeRand is studied empirically. Ten random sequences of integers in the range of $[0, 255]$ are generated as the random signals for experiment purposes. For the rest of the discussion, the phrase *random test signal* refers to the randomly generated test signal. Each random test signal is visualized as an image in Figure 6.5 using the resolution of 512×512 pixels. For additional comparison purposes, DeRand is applied to the standard test image Lenna, which act as the representative non-random test signal. The experiments are carried out using Matlab (version 7.12.0.635-R2011a) operating on OS X 10.9.2 (13C1021) platform and the performance of DeRand is evaluated based on three criteria, namely, carrier capacity, signal quality,



(a) R1 (b) R2 (c) R3 (d) R4

(e) R5 (f) R6 (g) R7 (h) R8

(i) R9 (j) R10

Figure 6.5: The set of 10 random signals each visualized as an 8-bit image. These random signals are generated by using Matlab function "randn", which is based on the ziggurat method for random variables generation (Marsaglia & Tsang, 2000).

and file-size increment.

6.6.1 Carrier capacity

Hereinafter, carrier capacity $\mathbf{C}_{raw}$ refers to the number of bits from the set D embeddable into the host signal X. The amount of $\mathbf{C}_{raw}$ indicates the efficiency of DeRand in defining redundancy (or venue) in X to accommodate D. Here, $\mathbf{C}_{raw}$ depends on three factors, namely, the statistical features of X, L and ρ.

First, the results when setting $\rho = 1$, viz., only one tuple (i.e., T_1^L) is allowed to be mapped to $\dot{T}_1^L$, are considered. When using Lenna as the representative non-random

test signal, $\mathbf{C}_{raw} = 1201,1532,419$ and 67 bits are achieved for $L = 9$, 10, 13 and 22, respectively. Here, $\mathbf{C}_{raw}$ is low (with respect to the size of the signal $|X|$) because only one tuple is allowed for mapping. In addition, the side information for $\rho = 1$ consumes $|\mathbf{S}| = |T_1^L| + |\dot{T}_1^L| + L + N + \rho$ bits of the carrier capacity. To facilitate the discussion, the effective carrier capacity $\mathbf{C}_e$ is defined as follows:

$$\mathbf{C}_e = \mathbf{C}_{raw} - (|A_L| + |B_L| + L + N + \rho). \tag{6.24}$$

Results reveal that $\mathbf{C}_e$ is 1174, 1502, 380 and 1 bits when $L = 9$, 10, 13 and 22, respectively, for the case of Lenna. Nonetheless, it is observed that the low effective carrier capacity (e.g., $\mathbf{C}_e = 1$ bit at $L = 22$) is a general trend in all test signals when they are parsed at large L's. In particular, the number of occurrences of T_1^L in the finite set X_L reduces when L increases. Hence, the number of mapped tuples and $\mathbf{C}_{raw}$ decrease when L increases.

On the other hand, the effective carrier capacity $\mathbf{C}_e$ increases when ρ increases. Table 6.1 records the value of $\mathbf{C}_e$ attained by setting ρ to different values when $L = 8$ and 13. For example, $\mathbf{C}_e = 380$ at $\rho = 1$ as reported earlier. However, the effective carrier capacity is increased to $\mathbf{C}_e = 13217$ bits by setting $\rho = 50$. In general, $\mathbf{C}_e$ consistently increases when ρ increase as suggested by Table 6.1. Similar trend is observed when ρ increased for different L. In order to study the trend of $\mathbf{C}_e$ due to increasing ρ, Figure 6.6 plots $\Delta_{\mathbf{C}_e}(\rho) = \mathbf{C}_e(\rho+1) - \mathbf{C}_e(\rho)$, where $\Delta_{\mathbf{C}_e}(\rho)$ refers to the effective carrier capacity when using the parameter ρ. Here, $\Delta_{\mathbf{C}_e}(\rho)$ generally decreases when ρ increases. This is because, as ρ increases, the newly utilized tuples for histogram mapping are of small number of occurrences in $\mathbb{H}$. At the same time, the side information increases as suggested by Eq. (6.24). For that, $\mathbf{C}_e$ is small for large ρ. This suggests that the complete utilization of the range $\rho \in [1, \hat{\rho}]$ (e.g., by setting $\rho = \hat{\rho}$) is not required to obtain high $\mathbf{C}_e$

Table 6.1: The effective carrier capacity $\mathbf{C}_e$ and SSIM of the test image Lenna achieved by using various values of ρ with $L = 8$ and $L = 13$.

L	ρ	$\mathbf{C}_e$ (bits)	SSIM
8	10	35114	0.5604
	20	60187	0.2923
	30	83933	0.2471
	40	106216	0.2439
13	50	13217	0.4038
	100	31986	0.2578
	150	38557	0.2245
	200	43776	0.2062
	250	47894	0.1917
	300	46594	0.1917
	350	49955	0.1792
	400	52929	0.1706
	450	59208	0.1536
	500	61331	0.1459
	550	63192	0.1368
	600	64877	0.1313
	650	66402	0.1266
	700	67760	0.1217
	750	69012	0.1186
	800	70163	0.1145
	850	71217	0.1099
	900	72177	0.1075
	950	73036	0.1038
	1000	73814	0.0999
	1050	74526	0.0964

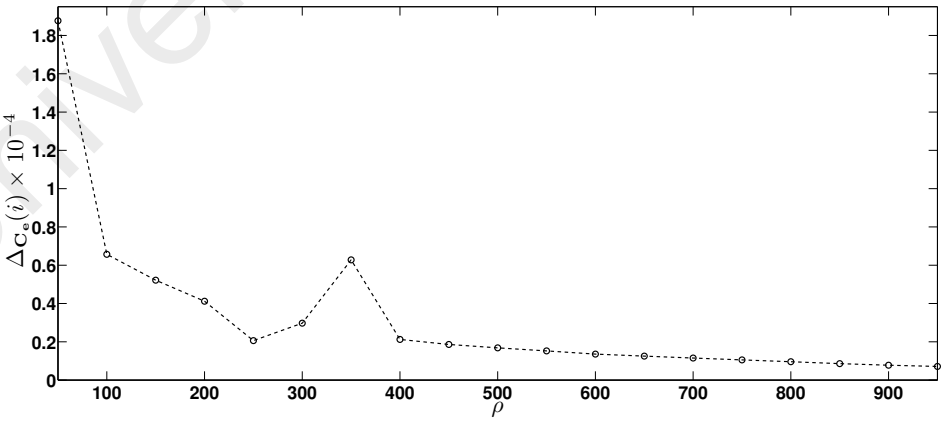


Figure 6.6: Change in effective carrier capacity $\Delta_{\mathbf{C}_e}(\rho)$ for the test image Lenna parsed at $L = 13$.

because as ρ approaches to $\hat{\rho}$, $\mathbf{C}_e$ increases insignificantly.

Among the parameters considered, the highest effective carrier capacity of $\mathbf{C}_e =$

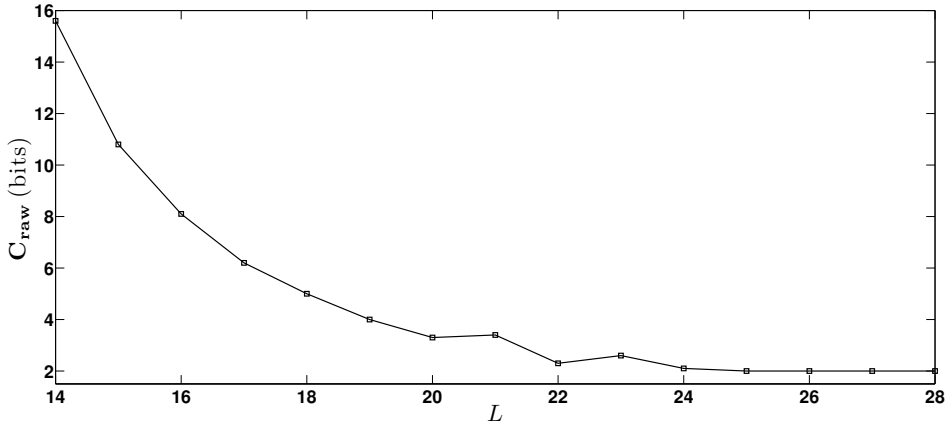


Figure 6.7: The average capacity of 10 test (random) signals for $\rho = 1$ versus L

106216 bits is achieved with (L,ρ)=(8,40) as reported in Table 6.1. Note that $\mathbf{C}_e(8,40) > E(13,1050)$. This is because, the number of mapped tuples at $L = 8$ is significantly larger than that of $L = 13$. The results in Table 6.1 also suggest that $\mathbf{C}_e$ is scalable, depending on the parameter values of (L,ρ).

Next, for the random test signals, the effect of the high entropy nature of these signals (i.e., low redundancy) on $\mathbf{C}_{raw}$ and $\mathbf{C}_e$ are apparent. For example, it is observed in Figure 6.7 that as L increases, the average value of $\mathbf{C}_{raw}$ decreases generally, where the horizontal axis presents the average $\mathbf{C}_{raw}$ for each L considered. Here, maximum of the averages of $\mathbf{C}_{raw}$ (i.e., 15 bits) is obtained when (L,ρ)=(14,1). Note that $|\dot{X}_L| = 0$ for $L < 14$ due to the high entropy nature of the random signals. Similar to the Lenna image, $\mathbf{C}_{raw}$ of the random signals also increase when ρ increases. Table 6.2 records $\mathbf{C}_{raw}$ for various values of ρ using R5 as the representative test random signal parsed at $L = 15$. Results suggest that, by increasing ρ from 1 to 50, $\mathbf{C}_{raw}$ increases from 10 to 417 bits. Table 6.2 shows that $\mathbf{C}_{raw}$ increases up to 3369 bits when $\rho = 500$. However, this increment in $\mathbf{C}_{raw}$ results in file size increment. This is because, in general, $\mathbf{C}_e < 0$ (see Eq. (6.24)) in the random signals. In other words, the size of the side information is always larger than $\mathbf{C}_{raw}$. Thus, $\mathbf{C}_{raw}$ is insufficient to embed the side information along with the payload D. To realize data embedding in such case, the side information is appended to the

Table 6.2: carrier capacity $\mathbf{C}_{raw}$, MSE and the percentage of file size increment for the representative random test signal R5 achieved by using various values of ρ with $L = 15$ and $L = 16$.

L	ρ	$\mathbf{C}_{raw}$ (bits)	MSE	File-size increment %
15	50	417	50.81	0.08
	100	783	90.07	0.15
	150	1133	112.66	0.23
	200	1483	149.02	0.30
	250	1833	202.53	0.37
	300	2169	265.10	0.44
	350	2469	273.85	0.51
	400	2769	291.76	0.58
	450	3069	305.14	0.65
	500	3369	324.26	0.73
16	50	623	56.64	0.09
	100	1173	94.19	0.16
	150	1710	134.18	0.24
	200	2210	169.64	0.32
	250	2710	209.00	0.39
	300	3210	247.30	0.47
	350	3710	281.54	0.55
	400	4210	315.15	0.62
	450	4666	364.92	0.70
	500	4909	395.67	0.78

modified signal, therefore causing file-size increment as detailed in Section 6.6.3.

On other hand, it is observed that $\mathbf{C}_{raw}(16,\rho) \geq C(15,\rho)$ when considering the same values of ρ. For example, Table 6.2 shows that $\mathbf{C}_{raw}(16,500) = 4909 \geq C(15,500) = 3369$. This trend holds true for all values of ρ at $L = 16$ and 15. This is because the random signals are generated in the range $[0,255]$, i.e., 8-bit integer. Hence, the probability of having similar tuples at $L = 8\alpha$ (i.e., multiple of 8 such as $16,24,32,\cdots$) is high. On the other hand, it is observed that the amount of increment in $\mathbf{C}_{raw}$ decreases as ρ increases. Figure 6.8 shows the plot of $\Delta_{\mathbf{C}_{raw}}(\rho) = \mathbf{C}_{raw}(\rho+1) - \mathbf{C}_{raw}(\rho)$ using R5 as the representative random test signal parsed at $L = 16$, where $\mathbf{C}_{raw}(\rho)$ refers to the carrier capacity when using the parameter ρ. Generally, as ρ increases, the increment in $\mathbf{C}_{raw}$ decreases. For example, at $L = 16$, as ρ increases from 50 to 100, $\mathbf{C}_{raw}$ increases from 623 to 1173 bits. However, as ρ increases from 300 to 350, $\mathbf{C}_{raw}$ merely increases from

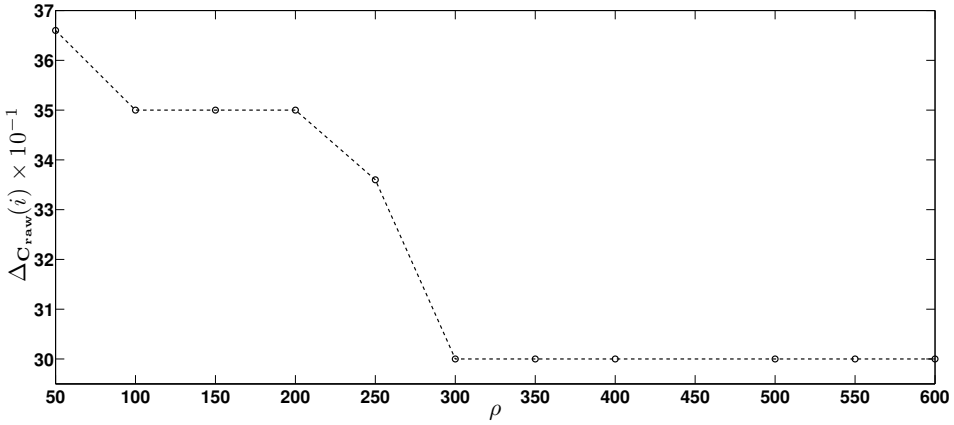


Figure 6.8: Change in capacity $\Delta_{\mathbf{C}_{raw}}(\rho)$ for the representative random test signal R5 parsed at $L = 16$.

3210 to 3710 bits. This is because, as ρ increases, the newly utilized tuples for histogram mapping have low number of occurrences in the histogram of X_L. Similar trend is also observed for the other nine random test signals, and the discussion is omitted here.

6.6.2 Signal Quality

In this section, the distortion in the modified signal due to the application of DeRand is evaluated. First, Lenna is considered as the representative non-random test signal. Here, it should be emphasized that SSIM is considered merely as a mean to visualise the scalability feature of DeRand in terms of output signal quality. It is observed that the distortion is generally subtle (i.e., high SSIM value) when $\rho = 1$. This is expected because only one tuple in X_L is utilized for histogram mapping while the remaining tuples remain unchanged. For example, at $L = 9, 10, 13$ and 22, the corresponding SSIM values are $0.8775, 0.8299, 0.9087$ and 0.9744, respectively. Here, the change in quality depends on L. In particular, when L is small, the number of tuples in X_L (and consequently the number of modified tuples) is high. For that, high distortion occurs in the modified image, such as the aforementioned cases of $L = 9$ and 10. However, as L increases, the number of tuples in X_L decreases and hence less tuples are modified. For that, at a relatively large L (e.g., $L = 13$ and 22), SSIM is high, indicating that the distortion is less severe for larger

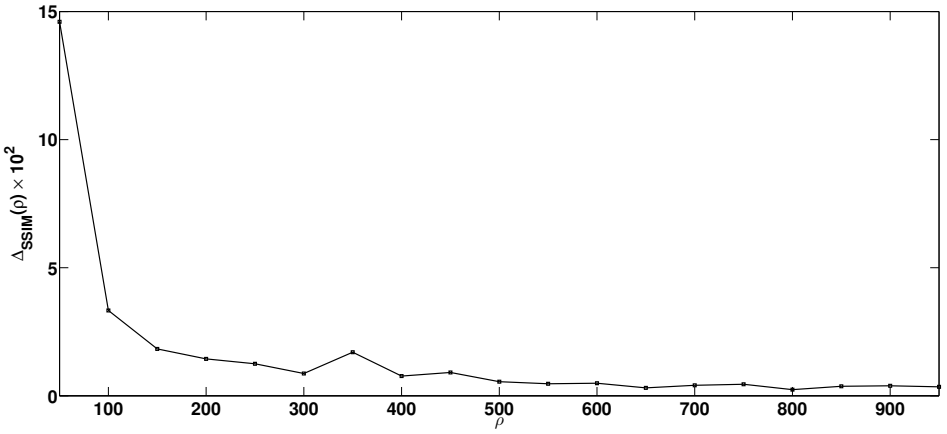


Figure 6.9: Change in visual quality $\Delta_{\text{SSIM}}(\rho)$ for the test image Lenna parsed at $L = 13$.

L.

On the other hand, the distortion is more severe as ρ increases. The last column of Table 6.1 records the SSIM attained by different ρ values for Lenna parsed at $L = 13$. Here, by increasing ρ from 1 to 50, SSIM decreases from 0.9087 to 0.4038. Here, SSIM decreases because more tuples are mapped (i.e., modified) when ρ increases. Generally, as ρ increases, the degree of distortion increases, where the most severe distortion occurs when (L,ρ)=(13,1050) with SSIM= 0.0964 as recorded in Table 6.1. Here, the value L defines the maximum possible distortion. For example, in Table 6.1, at $L = 8$, the lowest SSIM (viz., maximum distortion) is 0.2439 when $\rho = 40$. This is because, at $L = 8$, it is impossible to increase ρ to more than 40 due to the fact that $|\dot{X}_8|= 40$. Figure 6.9 shows the plot of $\Delta_{\text{SSIM}}(\rho) = SSIM(\rho) - SSIM(\rho+1)$, where $\Delta_{\text{SSIM}}(\rho)$ refers to SSIM when using the parameter ρ. Here, similar to the effective carrier capacity $\mathbf{C}_e$, $\Delta_{\text{SSIM}}(\rho)$ decreases as L increases for the same reason mentioned in the previous section. Figure 6.10 shows the output Lenna image produced by DeRand. Here, the distortion is controlled, i.e., scalable, by using the parameters L and ρ. In particular, DeRand offers a wide range of distortion by setting the aforementioned parameters. Specifically, the SSIM value ranges from 0.038 using (L,ρ)=(16,5000) to 0.975 using (L,ρ)=(22,1). It should

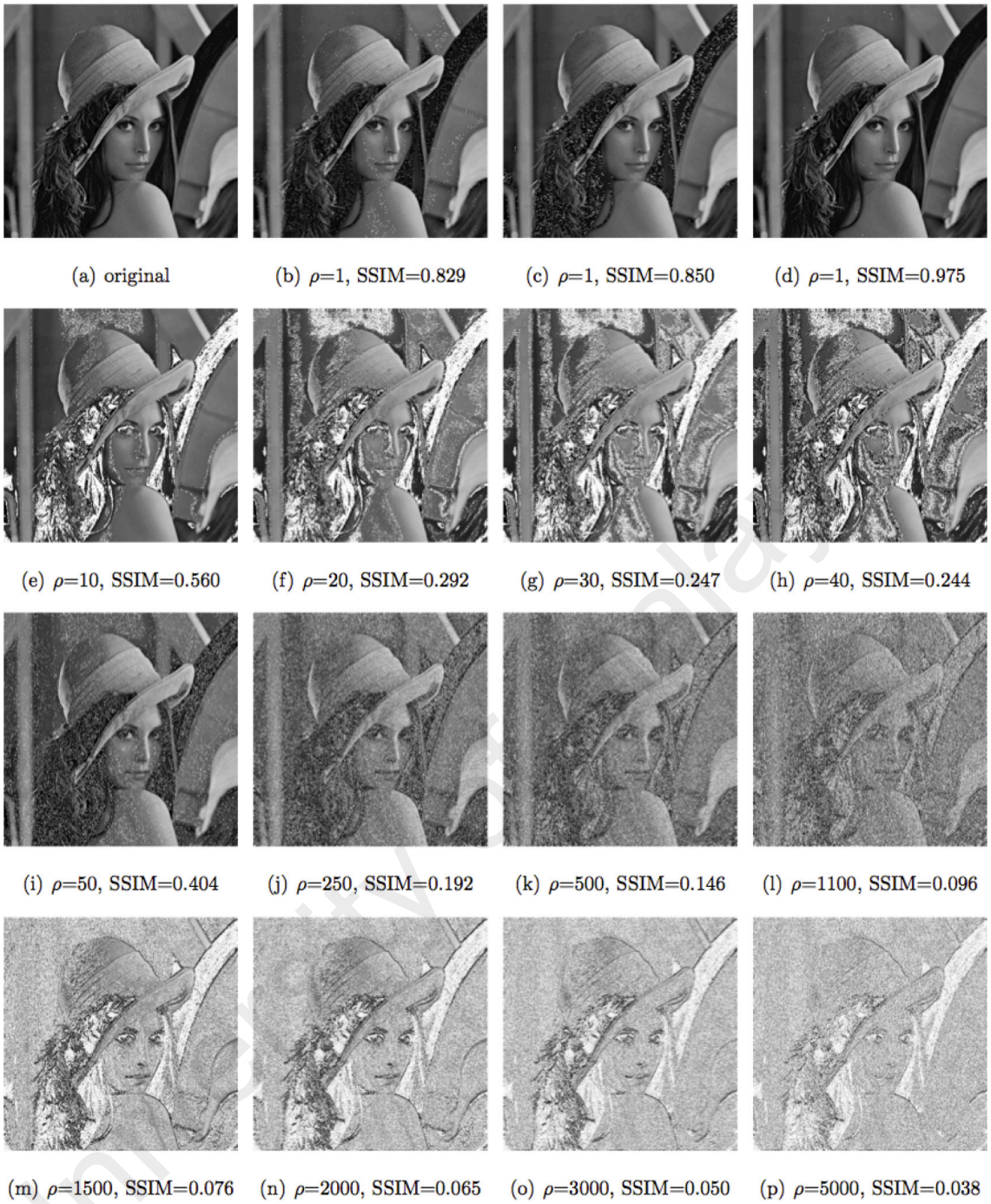


Figure 6.10: Output of the test image Lenna processed by DeRand using various combinations of L and ρ. The value L for 6.10(b), 6.10(c) and 6.10(d) are $10, 12$ and 22, respectively. For the second, third and forth rows, $L = 8, 13$ and 16, respectively.

be noted that this range of progressive degradation in image quality is wider than that attained by Ong et al. (2014), which spans the interval of $[0.040, 0.771]$.

Secondly, the random test signals are considered for data embedding. Instead of SSIM, the signal quality is measured by MSE (mean square error). Here, the random test signals behave similar to the Lenna test image when $\rho = 1$, where the average MSE of all

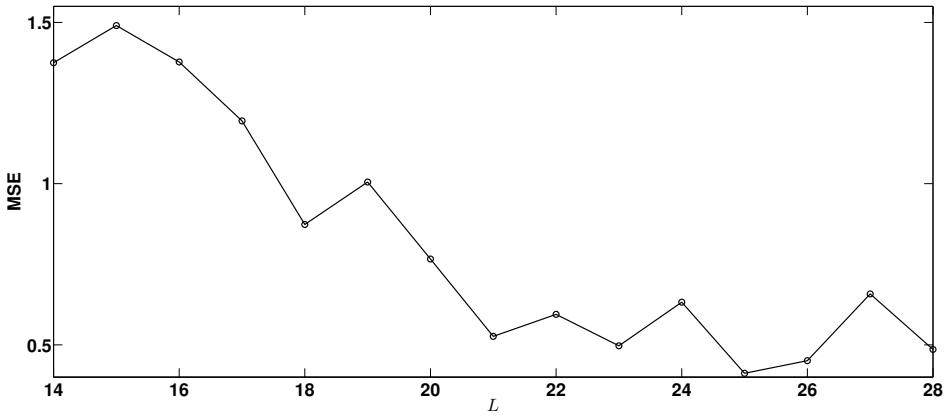


Figure 6.11: The average MSE of 10 test (random) signal for $\rho = 1$ versus L

random test signals decreases as L increases as shown in Figure 6.11. Here, the maximum of the averages of MSE is 1.5 and it is obtained when (L,ρ)=(14,1). This suggests that the modified image resembles its original counterpart. Among the 10 randomly generated test signals, it is observed that the minimum MSE is 0.0075, which is obtained with R6 at $(L,\rho) = (23,1)$. This is because, in general, only small number of tuples in the random test signal are mapped (regardless the value of L), which leads to limited distortion.

On the other hand, as shown in Table 6.2, MSE increases from 1.5 to 50.81 in the random test signal R5 (parsed at $L = 15$) when ρ increases from 1 to 50. This increment in MSE is directly proportional to the increment of $\mathbf{C}_{raw}$ due to the increment of ρ, and the maximum distortion attained is MSE= 395.67 at (L,ρ)=(16,500). Figure 6.12 shows the graph of $\Delta_{\text{MSE}}(\rho) = MSE(\rho+1) - MSE(\rho)$, where $\Delta_{\text{MSE}}(\rho)$ refers to MSE when using the parameter ρ. It is observed that $\Delta_{\text{MSE}}(\rho)$ increases when $\rho \leq 250$ because many tuples are utilized for histogram mapping. However, $\Delta_{\text{MSE}}(\rho)$ shows no significant change for $\rho \geq 300$. This is because, the tuples are of low number of occurrences for $\rho \geq 300$. Hence, the number of tuples utilized for histogram mapping is low. For that, the variation in distortion, i.e., $\Delta_{\text{MSE}}(\rho)$, is smaller than those at $\rho \leq 250$. This suggests that the increase in distortion is higher for smaller ρ, and vice versa. Similar trends are observed in other random test signals.

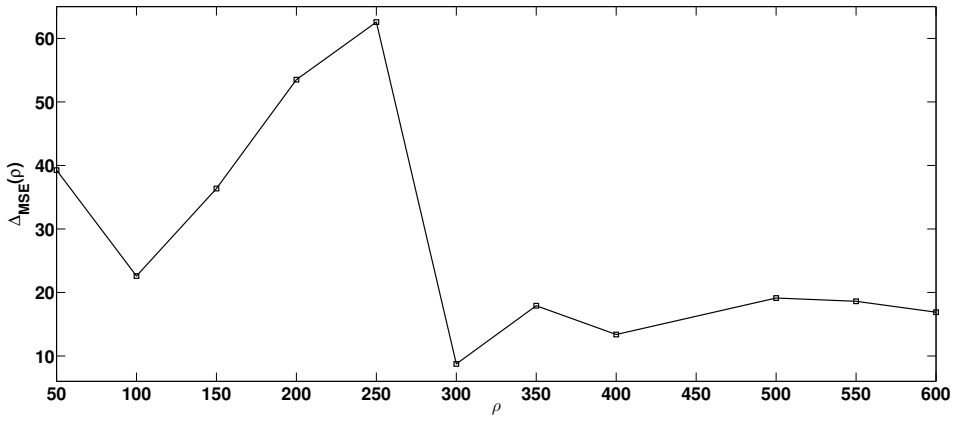


Figure 6.12: Change in visual quality $\Delta_{\mathrm{MSE}}(\rho)$ for the representative random test signal R5 parsed at $L = 16$.

6.6.3 File-Size Preserving

File-size preserving refers to the scenario where the size of the modified signal is similar to that of its original counterpart. This feature is important when the bandwidth of the communication channel is limited. In DeRand, this feature is conditional. Particularly, for the test image Lenna (i.e., non-random signal), DeRand completely preserves the file-size because all the side information can be embedded along with the payload. For random signal, DeRand increases the file-size because there is no venue to embed all the side information. In other words, only concatenation $X \leftarrow \mathbf{S} \oplus X$ is performed without removing $|\mathbf{S}|$ bits from the original input signal X. However, the file-size increment is insignificant. Table 6.2 records the percentage of file-size increment for R5 as the representative random test signal. Here, for (L, ρ)=(15,50), the file-size is increased merely by 0.08%. This percentage increases gradually as ρ increases and it reaches $\sim 0.73\%$ for $\rho = 500$. The increment in file-size for $L = 16$ is slightly higher than that of $L = 15$. For example, when $\rho = 500$, the file-size is increased by $\sim 0.78\%$ for $L = 16$, which is 0.05% higher than that of $L = 15$ using the same ρ. This slight increment is due to the side information, which is proportional to L as expressed by Eq. (6.24). Generally, the average of the file-size increment in Table 6.2 is only 0.42%.

Table 6.3: Functional comparison between the proposed DeRand and and (Ong et al., 2014)'s method.

	DeRand	(Ong et al., 2014)
Reversibility	Yes	Yes
Universal	Yes	No
Scalable carrier capacity?	Yes	Yes
File-Size Preserving (Non-Random Signals)	Yes	Yes
File-Size Preserving (Random Signals)	No ($< 0.42\%$)	Failed to embed
Progressive quality degradation?	Yes	Yes
Distortion Range (SSIM) (Non-Random Signals)	[0.038,0.975]	[0.040,0.771]
Distortion Range (MSE) (Random Signals)	[0.0075,395.67]	Failed to embed
Applicable to Random Signals	Yes	No

6.6.4 Functional Comparison with Existing Methods

For completion of discussion, Table 6.3 shows a functional comparison between the proposed DeRand and the method proposed by Ong et al. (2014). Here, both methods are reversible. However, DeRand is universal but the method proposed by Ong et al. (2014) is only applicable to image represented by an array of pixel values. DeRand and (Ong et al., 2014)'s method offer scalability in terms of carrier capacity and quality degradation. On the other hand, (Ong et al., 2014)'s method fails to embed any data in random signal. However, by design, DeRand achieves its objective of embedding data in random signals. In addition, DeRand is file-size preserving for non-random signals and only increases the file-size insignificantly (in average 0.42% only) for random signals as detailed in Section 6.6.3. Furthermore, when handling image (i.e., non-random signal), DeRand offers a range of distortion with SSIM spanning the interval of $[0.038, 0.975]$, which is wider than the interval of $[0.040, 0.771]$ attained by (Ong et al., 2014)'s method. This suggests that DeRand offers more scalability in terms of quality degradation. On the

other hand, DeRand can control the distortion of the modified random test signals and the observed range is $[0.0075, 395.67]$ in terms of MSE. However, the range of the quality degradation in the method proposed by Ong et al. (2014) could not be studied when it is applied to the random signals because this method fails to embed data. It should be noted that more distortion (than those reported in Table 6.1) could be attained theoretically by applying DeRand. However, a lower signal quality (i.e., more severe distortion) could not be achieved empirically due to the limitation in memory. In particular, the histogram of a signal parsed at $L > 16$ could not be handled since there are more than 2^{16} bins in the histogram to consider. Nonetheless, it should be recalled that SSIM is merely used as a mean to illustrate the scalable capability of DeRand, and the targeted signals of this chapter are the random signals. All in all, this functional comparison suggests that the proposed DeRand is superior to that proposed by Ong et al. (2014).

6.7 Summary

A novel universal histogram mapping method that defines redundancy in any digital signal for data embedding purposes was proposed. Theoretically, Theorem 3 establishes a mean to define redundancy based on the universal parser, and Theorem 4 shows that finding such redundancy is always possible for any digital signal. As such, even for high entropy signal such as random signal, the proposed histogram mapping can still be applied. Practically, these theorems are utilized to design the proposed data embedding method called DeRand, which is universally applicable to any digital signal including random signal. DeRand is the first universal reversible data embedding method applicable to the random signal. Experimental results show that DeRand achieves scalable carrier capacity (up to 4909 bits when processing random bit sequence of length 256 Kbytes) and scalable distortion with MSE spanning the range of $[0.0075, 395.67]$ with insignificant average file-size increment of 0.42%.

CHAPTER 7

DATA FUSION USING DUAL SEMANTIC CODE

7.1 Overview

In the previous chapters, the proposed universal data embedding methods are based on the conventional concept of data embedding. In particular, the concept considers the processing of two signals, namely, a payload and a host, which are associated by data embedding defined in Section 1.1. For that, the feasibility of this concept is limited. In particular, this concept cannot be generalized to the applications of data association such as metadata. In this chapter, the novel concept of data fusion is put forward, where any two or more signals are associated under the proposed framework. Data fusion aims at addressing the limitation of interchangeability among the conventional data association methods such as data embedding and metadata. First, the signals are mapped to the universal domain. Then, the signals are parsed by the universal parser and encoded losslessly by the proposed DSC (Dual Semantic Code), where each codeword of DSC can accommodate two pieces of data simultaneously. Hence, data fusion is achieved by mapping a segment of each signal to different parts of a DSC codeword using two proposed coding schemes, namely, the basic and partial coding schemes. In the basic coding scheme, all segments of the largest signal (in terms of size for storage) are mapped to the DSC codewords. However, in the partial coding scheme, only segments of high probability of occurrences are mapped to the DSC codewords. The proposed coding schemes are universally applicable to any signal, such as image, video, audio and text. Experimental results suggest that the basic coding scheme achieves, on average, a fusion bit-rate of 0.640 bpb (bits per bit). On the other hand, the partial coding scheme achieves, on av-

erage, a fusion bit-rate of 0.060 bpb in the file-size preserving mode. The fusion bit-rate can also be traded for compression purposes.

7.2 Introduction

Data association is a common phenomenon that occurs in many information systems. This phenomenon is of high significance as associated data are massively generated nowadays for various purposes. For example, applications such as data archival, retrieval, protection and integrity checking are based on the processing of auxiliary information, which are in turn associated with the data in question in some manner (Pereira, Vetro, & Sikora, 2008). Emerging applications such as semantic web and cloud computing also utilize various types of metadata that are closely associated to the data being processed (Leuf, 2004). In addition, the application of watermarking (Kamran, Khan, & Malik, 2014), steganography (Yuan, 2014), and fingerprinting (Wu & Satoh, 2013) can be perceived as the process of associating two sets of data to serve the purposes of copyright protection, covert communication, and illegal duplication detection, respectively.

Conventionally, the problem of processing associated data is application-oriented. For example, metadata management is not categorized under the same framework of watermarking, although they consider the problem of processing associated data. Hence, the applicability of metadata is restricted by the intended applications, and similar argument holds true for watermarking, steganography, and other applications of data association.

Thanks to the advancement of low-cost data capturing devices and storage technologies, data and their associated data are frequently generated nowadays. Various metadata management and data embedding methods are applied to handle these associated data. The unification of these methods under a single framework is significant because it results in a universal data association method that can be deployed to conveniently associate two or more sets of data of any format stored in any file system. As another example of

application, metadata can be associated with its corresponding data by the cloud administrator without suffering from increment in data size (e.g., caused by appending metadata to the header of the data in question). In addition, the universally applicable property in the proposed method eliminates the need to extract features from the contents stored in the cloud, where feature extraction is one of the technical challenges faced by the conventional feature-dependent data embedding methods. In general, a universal method that processes associated data without relying on the features of the data has not been considered in the current literature.

In this chapter, a novel concept referred to as *data fusion* is proposed. The concept of data fusion generalizes the concept of data embedding. In particular, the conventional concepts of data embedding, including difference expansion (Tian, 2003), histogram shifting (Li et al., 2013), LSB substitution (Celik et al., 2005), and codeword mapping (Mobasseri et al., 2010), are limitedly applicable to signals of certain features. In other words, they are of restricted interchangeability due to feature-dependency. However, data fusion generalizes the concept of data embedding to achieve any kind of data association. Hence, data fusion is a universal process. In addition, the aforementioned data embedding methods are limited to associate two data only, i.e., payload and host. However, data fusion can be applied to associate two or more data. Furthermore, the proposed data fusion has no restriction on the intended application of data association. For that, the applications of data fusion can be tailored for metadata, watermarking, fingerprinting and any application that associates data. The proposed data fusion operates in a domain referred to as the universal domain, in which any signal can be represented. As such, data fusion can be interchangeably applied to any associated data or signals. The process of data fusion is realized by applying the universal parser (Section 3.3.3) to any two associated (or to be associated) signals defined in the universal domain. Then, the parsed signals are encoded by a proposed coding method referred to as DSC (Dual Se-

mantic Code). Here, each codeword of DSC can simultaneously accommodate symbols from two different signals. The mapping to DSC is achieved by two proposed coding methods, namely, the basic coding scheme and partial coding scheme.

7.3 Data Association

Fundamentally, the amount of information (i.e., entropy) is zero if a single symbol is sent alone (i.e., without an association with at least another different symbol) over a communication channel. Thus, data association is an inherent feature that serves various purposes. In Section 7.3.1, the representative applications that associate or based on assoiated data are surveyed. Then, the notion of associated data is formally defined in Section 7.3.2.

7.3.1 Survey on Associated Data

Data association can occur naturally among two or more sets of data for essential functions. For example, any set of data should be associated with a name (or index/-pointer), in addition to the end-of-file information for proper storage and retrieval purposes. Traditionally, auxiliary information is associated with its corresponding data, such as the type of file (i.e., the underlying coding structure of the signal), time of creation and modification, accessibility permissions, and so on (Gal & Toledo, 2005; Chen, Zhang, & Yu, 2013). The purposes of data association are to facilitate data indexing, retrieval, archiving, etc. as detailed in (Alattar, 2004; Hu et al., 2009). Conventionally, associated or auxiliary information is commonly referred to as *metadata* (Alattar, 2004; Hu et al., 2009; Li et al., 2011). Table 7.1 shows examples of metadata for various classes of information, such as image (Stvilia, Jörgensen, & Wu, 2012; Li et al., 2013), text (Blake & Knudson, 2002), video (e.g., YouTube, Daily Motion) (Kinsella, Passant, & Breslin, 2011), audio (Smith & Schirling, 2006; Fallahpour, 2008) and web pages, where metadata plays the vital role of facilitating the search processes in search engines (Wu & Huang,

Table 7.1: Examples of metadata in various classes of information

Data	Metadata
Image	dimension of image, pixel depth
Video	tags (or keywords), description of content
Audio	artist, album, year, track number, copyright information
Text	name of the author, publisher information, summary (or abstract)
Web pages	Description and keywords meta tags

2012; Xuan et al., 2008). Due to its significance in various application, some standards for metadata were put forward, including IIM (Information Interchange Model) (Sandhaus & Boll, 2011), XMP (Extensible Metadata Platform) (Jeszenszky, 2007), Exif (Exchangeable image file format) (Fan, Cao, & Kot, 2013) and Dublin Core (Tseng, 2005).

In audio, image and video signals, metadata is embedded (or accommodated) in the header of the file ((NISO-U.S.), 2004). However, the header is generally vulnerable to unauthorized access, intentional tampering and un-intentional alteration. For that, the conventional metadata management is infeasible in applications where the associated data needs protection. For example, in watermarking, a watermark is associated with its corresponding data for copyright protection. Here, the embedded watermark should be robust against intentional/un-intentional removal or modification. Also, the watermark should be hidden in its corresponding data. Obviously, the conventional metadata management does not satisfy these requirements. To this end, date embedding is adopted as an alternative solution to associate data for applications such as watermarking, steganography, media broadcast monitoring, tamper detection, ownership authentication, device control, metadata storage and others as detailed in Cover and Thomas (1991).

In data embedding, two (correlated or un-correlated) sets of data, i.e., a host and a payload, are associated. The process of data embedding is achieved by modifying features in the host to accommodate the payload. Therefore, data embedding is a feature dependent process, where the interchangeability among the conventional data embedding

methods is often restricted, as discussed in Chapter 2.

7.3.2 Definition of Associated Data

Definition 6. *Given a set of alphabets* $\mathbb{A}$ *and two sources of information, namely,* $\mathfrak{D}_1 = \{a\}$ *and* $\mathfrak{D}_2 = \{b\}$ *such that a and b are singletons and* $a, b \in \mathbb{A}$, $\mathfrak{D}_1$ *and* $\mathfrak{D}_2$ *are* associated *in an ordered set* $\mathfrak{D}_{12} = \{a, b\}$ *if there exists a function* $S(X)$ *such that:*

$$S(\mathfrak{D}_{12}) = \begin{cases} True, & if\ (\mathfrak{D}_{12} \backslash \mathfrak{D}_1 = \mathfrak{D}_2)\ and\ (\mathfrak{D}_{12} \backslash \mathfrak{D}_2 = \mathfrak{D}_1); \\ False, & otherwise, \end{cases} \tag{7.1}$$

where the symbol "\" refers to the set difference, the cardinalities of $\mathfrak{D}_1$, $\mathfrak{D}_2$ *and* $\mathfrak{D}_{12}$ *are* $|\mathfrak{D}_1| = 1$, $|\mathfrak{D}_2| = 1$ *and* $|\mathfrak{D}_{12}| = 2$, *respectively* □

Here, the function $S(X)$ indicates the validity of the association among the elements of the ordered set X based on some rules. For example, if the English dictionary is considered to define the rules, the set $X = \{B, E, E\}$ has an association among its symbols, which is the English word "BEE". Thus, based on the English dictionary, $S(\{B, E, E\}) = True$. However, based on the French dictionary, the symbols in X has no association. Hence, based on the French dictionary, $S(\{B, E, E\}) = False$. Defining the rules that govern the function S is beyond of the scope of this study. However, generally, data association is a relative feature that depends on the rules of the function $S(X)$.

Definition 7. *The order of data association in a set* $\mathfrak{D}_{1,2,\cdots,B}$ *is the number of sources for the information associated in* $\mathfrak{D}_{1,2,\cdots,B}$, *and it is measured by the function O as follows:*

$$O(\mathfrak{D}_{1,2,\cdots,B}) = |\mathfrak{D}_{1,2,\cdots,B}| = |\mathfrak{D}_1| + |\mathfrak{D}_2| + \cdots + |\mathfrak{D}_B|, \tag{7.2}$$

where $|Z|$ *is the cardinality of the set* Z □

For the previous example, $O(\mathfrak{D}_{12}) = 2$, i.e., the set $\mathfrak{D}_{12}$ is of *second order association*. In this chapter, our study is limited to apply data fusion to second order associated data due to the limitation of the proposed dual semantic coding, which is detailed in Section 7.4.1. However, in theory, the concept of data fusion is generally applicable to any order higher than unity.

7.4 Data Fusion

In this section, the proposed concept and methods of data fusion are presented. Conceptually, data fusion is applied to data defined in the universal domain $\mathbb{U}_{\mathbb{A}}$. In this chapter, the second order data fusion is considered and applied to digital signals, i.e., $\mathbb{A} = \{0, 1\}$. However, the proposed method is generally applicable to any set of alphabets. Practically, data fusion is achieved in two major stages. The first stage is modeling the signals where the modeling is *strictly optional* in order to preserve (or reduce) the size of the resulting fused signals to that of the larger (i.e., between the two to be fused) signal in terms of storage size. The second stage is fusing the (modeled, if applicable) signals.

Given two signals X and $\bar{X}$ to be fused, the modeling stage aims at locating redundancy in one of these two signals. Here, the signal of larger size (say X without lost generality) is chosen for modeling. The modeling is achieved by the universal parser (Section 3.3.3). Initially, any signal defined in $\mathbb{U}_{\mathbb{A}}$ has zero redundancy because the entropy of such signal is assumed to be $\frac{1}{|\mathbb{A}|}$, as shown in Section 3.3.2. Using the universal parser, redundancy is located by partitioning the symbols of X into non-overlapping segments, where each segment is of fixed-length L for $1 \leq L \leq N$. The modeling is achieved by the ordering function as detailed in Section 3.3.3. Such modelling changes the entropy of the partitioned sequences such that $H(X_L) > H(X_{L+1})$ (Section 3.3.3). Hence, the redundancy R_L satisfies the condition of $R_L < R_{L+1}$. This change in redundancy can be optionally utilized for data fusion.

In the second stage, sequences in the set X_L are fused with $\bar{X}$ using the order-2 data fusion function Ξ^2 as follows:

$$\Xi^2 : X_L \times \bar{X} \rightarrow C, \tag{7.3}$$

or, in other words,

$$\Xi^2 : \{T_1^L, T_2^L, \cdots, T_{\lambda_L}^L\} \times \{\bar{x}_1, \bar{x}_2, \cdots, \bar{x}_{\lambda_L}\} \rightarrow \{c_1, c_2, \cdots, c_{\lambda_L}\}, \tag{7.4}$$

where C is a set of DSC (Dual Semantic Code) detailed in the next section.

7.4.1 Dual Semantic Code

In this section, an entropy coding method dedicated for data fusion referred to as DSC (Dual Semantic Code) is proposed. DSC is a set of codewords in which case each codeword can accommodate two symbols. This set is generated in two steps: The first step generates a set of general expressions; the second step derives a set of codewords from each general expression. These two steps are detailed in the following sub-sections.

7.4.1 (a) Generating General Expressions (GE)

A general expression (GE) is a set of ordered terms, namely, d, z and y, arranged based on the rules **R1** and **R2** as illustrated in Figure 7.1. Each general expression starts and ends based on the rules in Figure 7.1 for unique decodability purposes as detailed in Al-Wahaib and Wong (2010). Here, the terms d, z and y are referred to as the starting bits, information bits, and ending bits, respectively. Generally, a GE starts with d, which could be single or multiple (consecutive or non-consecutive) terms in the expression, followed by a single z, which could be immediately after a single d or in between multiple d's, and a GE always ends with a single y.

R1 A general expression always starts with d, followed by z and ends with y.

R2 In any codeword derived from a general expression, d can be an individual arrangement, or multiple similar arrangements repeated in consecutive or non-consecutive order while z and y are always single arrangements.

R3 d is a particular element in the set of all possible arrangements $\mathbb{P}$. In other words, $\{d\} \subset \mathbb{P}$ such that $|\{d\}| = 1$ where $|I|$ denotes the cardinality of the set I.

R4 z can assume any element in the set $Z := \mathbb{P} \backslash \{d\}$.

R5 y can assume any element in the set $Y := \mathbb{P} \backslash \{d\}$.

Figure 7.1: Rules of generating general expression

For example, the first five GE's are *dzy*, *ddzy*, *dzdy*, *dddzy*, *dzddy*, *ddzdy*, and the possible arrangements of GE's can be generated endlessly as illustrated in Figure 7.2. Each GE derives a sub-set of codewords by assigning an arrangement of bits to d, z and y. Generally, two different sets of codewords can be generated, namely; (1) EBI (Ending Bits Independent) codewords, and; (2) EBD (Ending Bits Dependent) codewords (Al-Wahaib & Wong, 2010). In this study, only EBI codewords are considered and they are detailed in the next sub-section.

7.4.1 (b) Deriving EBI Codewords from GE

An EBI codeword is generated by deterministically assigning arrangement of bits to d and z only, while dummy bits are assigned to y, which are updated later during the actual implementation of data fusion. Such assignment results in the set of codewords C.

Generally, a codeword $c_q \in C$ is generated by assigning a combination of n bits from a set $\mathbb{P}$ to each term in a general expression based on the rules **R3**, **R4** and **R5** in Figure 7.1. Here, $\mathbb{P}$ is the set of all possible binary sequences of length n, with the cardinality $|\mathbb{P}| = 2^n$ for $n \geq 2$. In this study, $n = 2$ is selected to achieve the minimum possible length of the codewords, and hence $\mathbb{P} = \{00, 01, 10, 11\}$. Based on **R2** in Figure 7.1, d can take any of element (i.e., combination of two bits) in $\mathbb{P}$. However, z and y can only

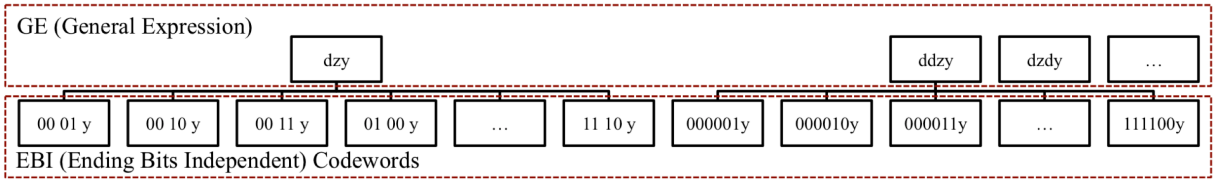


Figure 7.2: The generation of DSC, which is achieved in two stages: (1) generating general expression, and; (2) assigning arrangement of bits to general expressions to produce EBI codewords

assume elements from the set $\mathbb{P}\backslash d$ based on **R4** and **R5** in Figure 7.1. For example, if $d = 00$, then z,$y \in \{01, 10, 11\}$. Hence, for GE of dzy, the first EBI codeword is $c_0 = 00, 01, y$, the second codeword is $c_1 = 00, 10, y$, and so on, where the commas (i.e., ",") are included here to illustrate the arrangement of bits assigned to each individual term in c_q. However, hereinafter, these commas are removed. Thus, the set of EBI codewords is $C = \{0001y, 0010y, 0011y, 0100y, 0110y, 0111y, 1000y, \cdots\}$. Figure 7.2 shows some EBI codewords of GE=dzy and GE=$ddzy$.

Before achieving data fusion, y is either un-defined temporarily or given any dummy arrangement from $\mathbb{P}\backslash d$ based on **R5**. Here, although y is un-defined in c_q, this codeword is still uniquely decodable because d and z are unique in each codeword $c_q \in C$ (Al-Wahaib & Wong, 2010). However, y is required to exist in c_q (regardless the element of $\mathbb{P}$ assigned to y) to indicate that the end of a codeword is reached based on **R1**. During data fusion, y is changed based on the data to be fused.

The length of an EBI codeword is denoted by ζ and it depends on the number of terms in GE utilized in the derivation of that particular codeword. Specifically, if ω is the number of occurrences of the term d in a GE, then ζ is defined as:

$$\zeta = (\omega + 2) \times n \quad \text{bits}, \tag{7.5}$$

where $\omega \geq 1$. Hence, if $n = 2$ and $\omega = 1$, then $\zeta = 6$ bits is the minimum length of any DCS codeword, where such codeword is derived from GE=dzy. Note that in DSC, it is

possible for two (or more) different GE's to generate codewords of similar length ζ but of different arrangements. For such GE's, the value of ω is the same. For example, ζ of all codewords generated by the GE's *dddzy*, *ddzdy* and *dzddy* are 10 bits.

Hence, given a set of GE's all assuming the same ω, the total number of codewords of similar length that can be generated (based on ω) is denoted by ψ_ω and defined as follows:

$$\psi_\omega = \omega \times |d| \times |\mathbb{P} \backslash d| \quad \text{codewords}, \tag{7.6}$$

where all these codewords are of identical length ζ. In particular, if $n = 2$, then:

$$\psi_\omega = \omega \times 12 \quad \text{codewords}. \tag{7.7}$$

For example, the total number of codewords of $\zeta = 10$ is $\psi_3 = 3 \times 12 = 36$. These codewords are derived from the GE's *dddzy*, *ddzdy* and *dzddy*.

In order to reduce the length of a signal encoded by DSC, it is desirable to encode most tuples of the signal by short codewords. This can be achieved by encoding the tuples based on their PoO (probability of occurrences), such that tuples of high PoO are encoded by short codewords, and vice versa.

7.4.2 Coding Schemes for Data Fusion

In this section, DSC (Dual Semantic Code) generated in Section 7.4.1 is utilized for data fusion. The purpose of applying DSC is because each DSC codeword is capable in accommodating two different symbols simultaneously as detailed in Section 7.4.2 (a). DSC is the first coding method that has such capability. However, generally, any codewords that has similar capability as that of DSC can also be utilized for data fusion purposes.

The proposed flow of operations in data fusion using DSC is summerized in Figure 7.3. Here, assume that two sequences X and $\bar{X}$ are to be fused, where $|X| > |\bar{X}|$ is assumed.

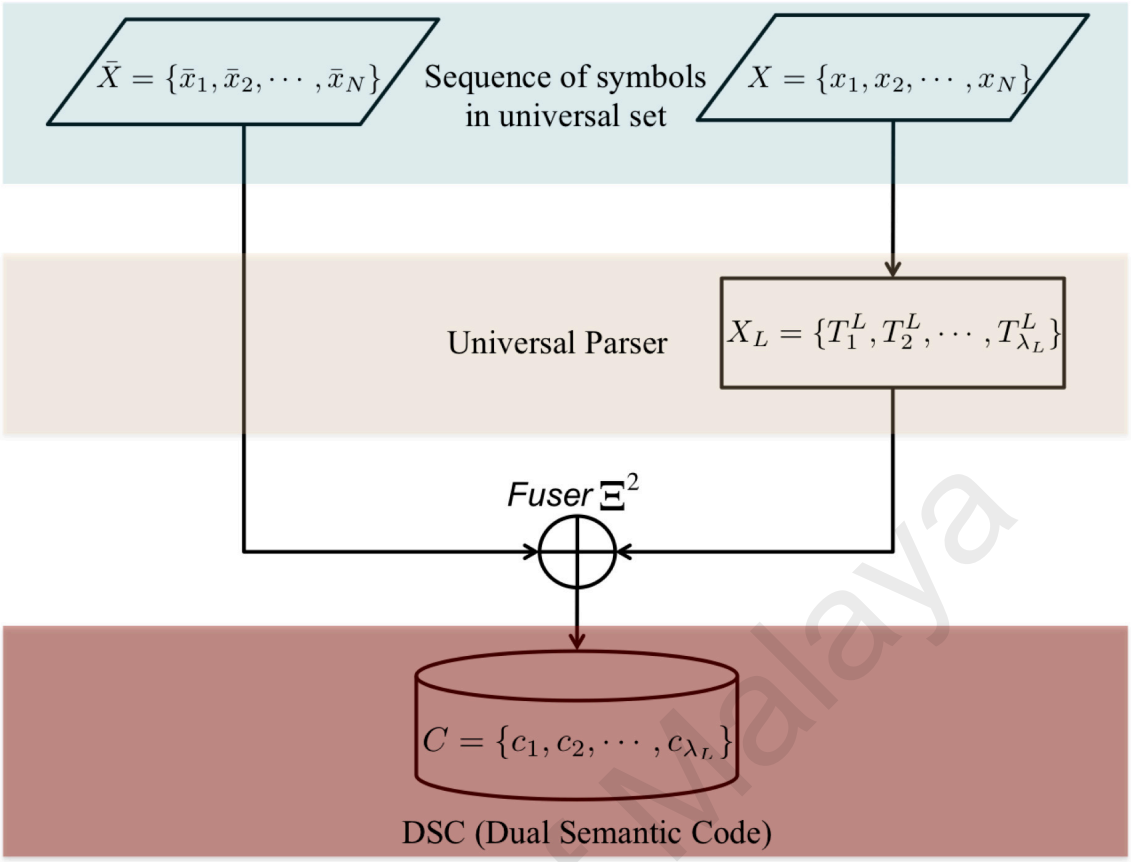


Figure 7.3: The flow of data fusion operations using dual semantic code

Then, the universal parser (Section 3.3.3) is applied to X for deriving X_L. Next, the set X_L is scanned and PoO's of the tuples are computed. Finally, the tuples in X_L are fused with bits from $\bar{X}$ by coding them using DSC based on PoO of the segments of X_L. Particularly, data fusion is achieved by utilizing the EBI codewords as detailed in the next sub-sections.

7.4.2 (a) Data Fusion using EBI

An EBI codeword c_q can accommodate two different symbols simultaneously by mapping one symbol to the dz part of c_q and the other symbol to the y part of the same c_q, where this y is un-defined or assumes a dummy value before achieving data fusion.

Given an EBI codeword $c_q = d\ z\ y$, this codeword is utilized to fuse $T_t^L \in X_L$ and $\{\bar{x}_t, \bar{x}_{t+1}\}$ as follows:

$$\Xi^2 : T_t^L \times \{\bar{x}_t, \bar{x}_{t+1}\} \rightarrow c_q. \tag{7.8}$$

Informally, T_t^L is mapped to the starting and information bits (i.e., dz terms) and $\{\bar{x}_t, \bar{x}_{t+1}\}$

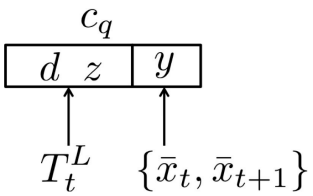


Figure 7.4: Data fusion of a tuple T_t^L from X and two bits $\bar{x}_t, \bar{x}_{t+1} \in \bar{X}$ in EBI codeword c_q

is mapped to the ending bits (i.e., y term) of c_q as illustrated in Figure 7.4. Here, the mapping of $\{\bar{x}_t, \bar{x}_{t+1}\}$ to y is based on the three-state property of y, which in turn based on **R5** in Figure 7.1. In other words, y can assume any element (arrangement of bits) in the set $\mathbb{P}\backslash d$, where cardinality of $|\mathbb{P}\backslash d| = 3$. For example, if $d = \{00\}$, then $y \in \{01, 10, 11\}$. However, if $d = \{10\}$, then $y \in \{00, 01, 11\}$, and so on. These states are referred to as $\{\mathfrak{S}_1, \mathfrak{S}_2, \mathfrak{S}_3\}$, where $\mathfrak{S}_1 < \mathfrak{S}_2 < \mathfrak{S}_3$ holds true when considering these arrangements as binary numbers. For example, if $d = \{10\}$, then $\{\mathfrak{S}_1 = 00, \mathfrak{S}_2 = 01, \mathfrak{S}_3 = 11\}$, and so on. The next two sub-sections propose two coding schemes based on data fusion using EBI.

7.4.2 (b) Basic Coding Scheme

In this coding scheme, all segments of X_L are mapped to EBI codewords. If modeling is applied (to reduce the length of the resultant codewords), then the sorted segments of X_L (in descending order) are stored as side information. Next, the EBI codewords are assigned to the segments of X_L based on their PoO's in the histogram of X_L as detailed in Section 7.4.1 (b). Finally, data fusion is achieved by utilizing the codewords as detailed in Section 7.4.2 (a).

For example, if the tuple $T_t^L = \{1, 1, 1, 1, 1, 1, 1, 1\}$ has the highest PoO in X_L, then EBI codeword $c_1 = 0001y$ is assigned to it. Hence, to fuse this tuple with $\{0, 0\}$, the following mapping occurs:

$$\Xi^2 : (T_t^L, \{0,0\}) : \{1,1,1,1,1,1,1,1\} \rightarrow 0001, \tag{7.9}$$

and $\{0,0\}$ is converted to the binary number 00_2 so that

$$\mod(00_2, 3) = 0. \tag{7.10}$$

Thus, the 0-th element in $\mathbb{P} \setminus d$, i.e., 01, is assigned to y, and the resultant codeword $c_1 = 000101$. Data fusion using this coding scheme may results in file-size increment, i.e., $\mathscr{T}(\Xi^2) > \mathscr{T}(X)$ where $\mathscr{T}(\Xi^2)$ is the size (in bits) of the assemble of codewords that accommodates the fused signals, and $\mathscr{T}(X)$ is the size of the original X. Such file-size increment is observed in our empirical study (Section 7.5). To avoid file-size increment, the partial coding scheme is proposed and presented in the next sub-section.

7.4.2 (c) Partial Coding Scheme

This coding scheme has two aims, namely: (1) achieving data fusion with a gain in data compression or achieving data fusion while preserving the file-size of X, and; (2) realizing scalability. In order to achieve these objectives, only some segments in X_L with high PoO's are selected and mapped to short EBI codewords. For example, assume a segment $T_t^L \in X_L$ is mapped to an EBI codeword c_q such that $\mathscr{T}_{c_q} < L$. In this case, compression is gained while data fusion is achieved in c_q as shown in Section 7.4.2 (a). The amount of removed bits due to the mapping of T_t^L to c_q is denoted by Π_{c_q} and defined as:

$$\Pi_{c_q} = L - T_{c_q} \quad \text{bits}. \tag{7.11}$$

The partial coding first defines three sets, namely, $\mathbb{A}_L$, $\mathscr{B}$ and $\mathscr{C}$. The set $\mathbb{A}_L$ consists of all possible arrangements (tuples) of L bits and hence the cardinality $|\mathbb{A}_L| = 2^L$. For example, if $L = 3$, then $\mathbb{A}_3 = \{000, 001, 010 \cdots, 111\}$ and $|\mathbb{A}_3| = 2^3 = 8$. The set $\mathscr{B}$ consists of all tuples that actually occur in X_L, i.e., the tuples with PoO > 0 sorted in descending order based on their PoO. Formally, $\mathscr{B} = \{b_i : b_i \text{ is a tuple in } X_L \text{ with } \text{PoO}(b_i) > 0\}$. The set $\mathscr{C}$

is the set that consists of all tuples that do not occur in X_L, i.e., $\mathscr{C} = \mathbb{A}_L \backslash \mathscr{B}$. For example, if the set $X_L = \{000, 000, 001\}$, then $\mathscr{B} = \{000, 001\}$ and $\mathscr{C} = \{010, 011, 100, \cdots, 111\}$. Then, data fusion is achieved by Algorithm 4, where t is the sorted PoO of T_t^L in the histogram of X_L and the set $|Q_C|$ is defined later in this section. Here, Step (18), which is optional, increases the number of fused bits but sacrifices compression. However, without invoking Step (18), data compression is achieved. Note that, in Algorithm 4, c is a flag indicating that the immediate next codeword is an EBI codeword. In the decoding stage, the decoder searches for this flag to correctly decode the EBI codewords.

In order to guarantee that tuples are mapped to codewords in which they satisfy $\mathscr{T}_{c_q} < L$, the set Q_C is defined such that the mapping is achieved solely with codewords in the set Q_C. Hence,

$$\Xi^2 : X_L \times \bar{X} \rightarrow Q_C \subset C. \tag{7.12}$$

The set Q_C consists of all EBI codewords that satisfies $\mathscr{T}(c_q) < L$ where $|Q_C| \leq |\mathscr{B}|$. The set of codewords in Q_C is defined using Algorithm 5.

For example, if $L = 10$ is considered, then all DSC of the length $\zeta \leq 10$ bits are utilized for data fusion. This includes the union of all codewords of 6, 8 and 10 bits in length. Based on Eq. (7.7), the number of codewords with length 6, 8 and 10 bits are 12, 24 and 36, respectively. These codewords are $Q_C = \{0001y, 0010y, \cdots, 1110y, 000001y, 000010y, \cdots, 111100y, 00000001y, 00000010y, \cdots, 11111100y\}$, which account to $|Q_C| = 12 + 24 + 36 = 72$ codewords. In this case, only 72 patterns of the highest PoO's are mapped to codewords in Q_C, or in other words, codewords of length $\zeta > 10$ are not utilized.

An example of data fusion using the partial coding scheme is presented in Figure 7.5. Here, the set $X_8 = \{\{00000000\}, \{00000000\}, \{11111111\}\}$, where $L = 8$ bits and the set $\bar{X} = \{01\}$. Note that $\mathscr{B} = \{00000000, 11111111\}$. Hence, any tuple (from the remaining 2^8 tuples) that does not occur in X_8 can be selected as the flag c. In this example,

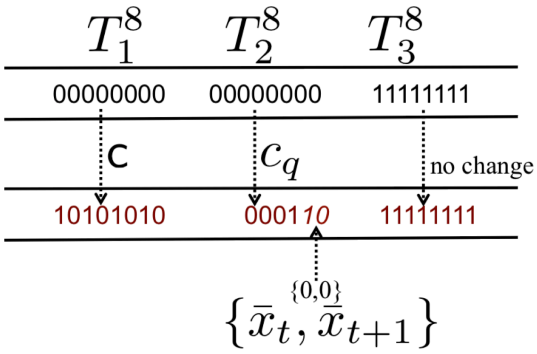


Figure 7.5: Example of partial coding using EBI codeword

$\mathsf{c} = \{10101010\}$ is selected. Since the first two groups in X_8 are $T_1^8 = T_2^8 = \{00000000\}$, i.e., they are adjacent and identical, the following mappings are achieved: T_1^8 is mapped to $\mathsf{c} = \{10101010\}$ but T_2^8, is mapped to $c_q = 0001y$, where $Q_C = \{0001y\}$ in this example. Next, T_2^8 is fused with $\bar{X} = \{01\}$ as detailed in Section 7.4.2 (a). Thus, data fusion results in $\Xi^2 = \{000110, 11111111\}$. Note that the last tuple $T_3^8 = \{11111111\}$ is not mapped to any EBI codeword because it is a single tuple, i.e., there is no neighbouring tuple of identical pattern. Thus, the actual output of the fused data is $\Xi^2 = \{10101010, 000110, 11111111\}$, as shown in Figure 7.5. In this example, the size $\mathscr{T}(X_8) = 16$ bits before data fusion and reduces to $\mathscr{T}(\Xi^2) = 14$ bits (i.e., compressed) after data fusion using the partial coding scheme. In this example, Step (18) in the encoding algorithm is skipped, and for that, data compression is gained. Nevertheless, L and c are stored as side information.

In the decoding stage, the bitstream is partitioned into segments of $L = 8$ bits. Here, the length L and flag c are derived from the side information. If the encountered segment is c, this implies that the following segment is an EBI codeword c_q. Thus, the inverse mapping is applied to c_q as follows:

$$dz \rightarrow T_2^8, \tag{7.13}$$

and

$$y \rightarrow \bar{X} \tag{7.14}$$

Algorithm 4 Data Fusion in Partial Coding Scheme

1: Generate X and X_L based on L
2: Generate $\mathscr{C}$
3: **if** $|\mathscr{C}| > 0$ **then**
4: Select any element $\mathsf{c} \in \mathscr{C}$
5: Merge L and c with $\bar{X}$
6: Go to step 12
7: **else**
8: $L \leftarrow L+1$
9: Generate the new set X based on L
10: Go to step 3
11: **end if**
12: Generate set Q_C based on L (**Algorithm 5**)
13: Scan two adjacent segments $T_t^L, T_{t+1}^L \in X_L$
14: **if** $(T_t^L = T_{t+1}^L)$ and $(i \leq |Q_C|)$ **then**
15: Map $T_t^L \rightarrow \mathsf{c}$
16: Map $T_{t+1}^L \rightarrow c_i \in C_Q$
17: Fuse T_{t+1}^L and $\{\bar{x}_t, \bar{x}_{t+1}\}$ using c_q as detailed in Section (7.4.2 (a))
18: Concatenate Π_{c_q} bits from $\bar{X}$, starting from $\bar{x}_{r+2}$, to c_q
19: **else if** All non-overlapping pairs in X_L are scanned **then**
20: Halt
21: **else**
22: Go to step 13
23: **end if**

Algorithm 5 Generating the Set Q_C

1: $\omega \leftarrow 1$ and $\zeta \leftarrow 6$
2: Suppose Q_C contains all codewords of length $\zeta = 6$, i.e., $|Q_C| = \psi_1 = 12$ (Eq. 7.7)
3: **if** $L - \zeta > 1$ **then**
4: $\omega \leftarrow \omega + 1$
5: Compute ψ_ω in Eq. (7.7)
6: Go to step 10
7: **else**
8: Halt
9: **end if**
10: Add all codewords of length $\zeta + 2$ to Q_C
11: $|Q_C| \leftarrow |Q_C| + \psi_\omega$
12: Go to step 3

while

$$\mathsf{c} \rightarrow T_2^8. \tag{7.15}$$

On the other hand, the partial coding scheme is scalable. Here, scalability is defined as the ability to control the size of the fused data, i.e., $\mathscr{T}(\Xi^2)$. Such flexibility enables three modes of operation, namely, compression, file-size preserving, and partial compression. As the names imply, each mode changes the size of the fused data.

Compression is gained by skipping Step (18) in the encoding algorithm. Here, data fusion is achieved solely by using EBI codewords where the venues generated by removing Π_{c_q} bits (due to mapping some segments in X_L to short EBI codewords) are not utilized for data fusion.

On the other hand, file-size preserving is achieved by utilizing the removed bits (i.e., Π_{c_q} bits in total) for data fusion. Here, Step (18) is invoked and hence *all* removed bits are replaced by segments of the smaller signal. In other words, the size of the fused data is larger than that achieved in the "compression" mode by Π_{c_q} bits, and no compression is gained.

Last but not least, partial compression is achieved when only *selected* removed bits are replaced by the fused data. This amount is denoted by π_{c_q} where $\pi < \Pi_{c_q}$, and it is pre-defined by the data fuser. Hence, Step (18) is invoked where Π_{c_q} is reduced to π_{c_q}. Here, the size of the fused data is increased, and the compression ratio is decreased but still greater than unity. All these three modes are selectively achievable during the actual data fusing process. However, there is a trade-off between the size of the fused data and the size $\mathscr{T}(\Xi^2)$.

7.4.3 Discussion

Data fusion is achieved by mapping two different symbols, namely, T_t^L and $\{\bar{x}_t, \bar{x}_{t+1}\}$, into a single codeword $c_q \in C$. However, the codeword c_q is completely different from

the two symbols that it accommodates. That is, $c_q \neq T_t^L$ and $c_q \neq \{\bar{x}_t, \bar{x}_{t+1}\}$. Here, data fusion does not aim at hiding data into a host as in data embedding. For that, quality assessment is not applicable to the codewords resulting from data fusion. This is one aspect supporting the claim that data fusion is different from data embedding. Other differences between data embedding and data fusion are summarized in Table 7.2. In data fusion, the terms “host" and “payload" are not used because data fusion is not based on modifying features of a signal in order to accommodate the other. Instead, the signals (to be fused) are mapped to a new entity (i.e., c_q) where such mapping depends solely on the basic features of the signals after defining them in the universal domain (Section 3.3.2). On the other hand, conceptually, the data embedding process is limited to two signals only, namely, the host and the payload. In the proposed concept of data fusion, the number of signals to be fused can be more than 2. Since any signal is defined in the universal domain, data fusion is a universally interchangeable process. However, data embedding depends on the features of the host in a certain domain, medium and coding format to embed the payload. For that, the interchangeability of data embedding is restricted. The comparison in Table 7.2 also suggests that data fusion is a generalization of data embedding. Such flexibility can be exploited to fuse any associated data together, for example, metadata or watermark with the relevant data. Thus, the problems of data embedding and metadata storage can be unified under the umbrella of data fusion.

The process of data fusion is completely reversible in which both T_t^L and $\{\bar{x}_t, \bar{x}_{t+1}\}$ can be perfectly extracted from c_q. The reversibility property is achieved by the inverse mapping of Eq. (7.8). Also, the integrity of all components of DSC codeword must be ensured for correct decodability. In other words, if either the first part of a codeword (i.e., dz which accommodates T_t^L) or the second part (i.e., y which accommodates $\{\bar{x}_t, \bar{x}_{t+1}\}$) is lost, then c_q cannot be correctly decoded. Other coding methods such as GRC (Golomb-Rice code) (Rice & Plaunt, 1971) do not have integrity for the data they are fusing . For example,

Table 7.2: Comparison between data embedding and the proposed data fusion

	Data Embedding	Data Fusion
Number of Signals	2 only	Any number of associated signals with order $O \geq 2$
Components	Host and Payload	Any two signals say (a and b)
Concept	Host is modified to embed Payload	a and b are fused in c
Feature Dependency	Features of Host	Basic features induced in the Universal domain
Distortion	Imperceptible	c is similar to neither a nor b
Interchangeability	Restricted	Universal

assume a GRC = "0101" is to be fused with "00". Data fusion can be simply achieved by concatenating "00" to such GRC, which results in "010100". However, if "00" is lost (e.g., due to communication error), this GRC (i.e., "0101") is still decodable where the error is undetected. On the other hand, assume DSC = "0011y" is to be fused with "00". Here, "00" is mapped to "01" as discussed in Section 7.4.2 (a). Hence, the resultant codeword of the fused data is "001101". Now, assume that 1 bit is lost from the fused data, i.e., the codeword becomes "00110". Such codeword cannot be decoded correctly because it is incomplete. The same is true when the last 2 bits, i.e., "01", are lost. It is concluded that, the capability to provide integrity to the fused data for correct decoding is a unique feature of DSC.

Table 7.3 presents a functional comparison between the proposed data fusion and conventional data embedding methods. The comparison in Table 7.3 includes the following aspects: reversibility, universal applicability, compressibility, file-size preservation, scalability in handling data embedding/fusion (depending on the method), and number of passes required to achieve data embedding/fusion. In general, the conventional data embedding methods considered and the proposed data fusion are reversible and file-size

preserving. The proposed data fusion is superior to the aforementioned data embedding methods in terms of universal applicability, compressibility and scalability. However, all the aforementioned data embedding methods require lower number of passes to achieve data embedding. Hence, it is expected that the execution time for these methods are shorter than that of the proposed data fusion.

Table 7.3: Functional comparison between the proposed data fusion and conventional data embedding methods

Method	Reversible?	Universal?	Compressibility?	File-Size Preserving?	Scalability?	Number of Passes
LSB(Celik et al., 2005)	✓	×	×	✓	×	1
HS(Li et al., 2013)	✓	×	×	✓	×	1
DE(Tian, 2003)	✓	×	×	✓	×	1
CM(Mobasseri et al., 2010)	✓	×	×	✓	×	2
Data Fusion	✓	✓	✓[1]	✓	✓	2

[1] In the proposed data fusion, the compression and file-size preserving are two different operating modes that can be selectively chosen in the partial coding scheme.

Table 7.4: Information of test signals considered for empirical study

	Raw	Compressed	Encrypted
Image	3 (BMP)	3 (JPEG)	3 (AES-256)
Video	3 (AVI)	3 (H.264)	3 (AES-256)
Audio	3 (WAV)	3 (MP3)	3 (AES-256)
Text	3 (Doc,Latex)	3 (ZIP)	3 (AES-256)

7.5 Experimental Results

In this section, data fusion is studied empirically by applying the proposed basic and partial coding schemes to a set of 36 test signals. These signals are of different media and coding schemes to verify the viability of the proposed coding schemes to various formats. Table 7.4 records the detailed information of each class of signals, along with the number of considered signals from each class. Particularly, the classes of signals are image, video, audio and text. Each class consists of a set of signals coded in different ways, namely, raw, compressed and encrypted.

The set of images include 3 standard test images, namely, Lenna, Baboon and Man, from (USC-SIPI, 2014). This set is utilized to derive two additional sets, each consisting of 3 images. The first set consists of the raw (un-compressed) images in BMP (BitMap) format. The second set consists of JPEG compressed images obtained by setting the quality factor to 90. The third set consists of encrypted images using AES (Advanced Encryption Standard) with key length of 256 bits.

The set of videos consists of three standard test videos, namely, Foreman (of dimension 176 $\times$ 144, 29.97 fps), Carphone (of dimension 176 $\times$ 144, 29.97 fps) and Suzie (of dimension 176 $\times$ 144, 29.97 fps). The raw uncompressed videos are in AVI (Audio Video Interleave), the compressed videos are in the H.264/MPEG-4 AVC format of the targeted bit-rate of 192 kbps, and AES-256 is deployed to encrypt videos.

The set of audios consists of segment of three different classical musics, namely, Audio1 (length of 2 seconds), Audio2 (length of 5 seconds) and Audio3 (length of 3 seconds).

Each raw uncompressed audio is in the WAV (Waveform Audio file) format with a sampling rate of 44,100 samples per second. The compressed audios are in the MP3 format with bit-rate of 128Kbps, and AES with key length of 256 bits is considered to encrypt the audio files.

The set of texts consists of three texts, namely, Text1 (54 KB) in MS-Word format, Text2 (6 KB) in Latex format, and Text3 (13 KB) in MS-Word format. The set of compressed texts is generated by using Zip (version 12.0 Pro (8252) for Windows operating system), while the set of encrypted texts were generated by using AES-256 with a key of length 256 bits. All the aforementioned signals are fused with a randomly generated signal.

The experiments are carried out using C programming language. The performance of the two proposed coding schemes are evaluated based on the following criteria: fusibility, fusion bit-rate, and compression gain. It is verified that the proposed basic and partial coding schemes are reversible, and the fused signals can be perfectly restored from the DSC codewords. The following sub-sections detail the performance of the proposed coding schemes.

7.5.1 Fusibility

Fusibility is the ability to define the data fusion function Ξ^2 over any two sequences X and $\bar{X}$ such that Eq. (7.4) holds true. Practically, the fusibility is verified by running the implemented data fusion program to its completion for any two inputs X and $\bar{X}$. Figures 7.10, 7.11 7.13 and 7.12 show that data fusion can be applied to all classes of signal considered in the raw, compressed and encrypted forms, where the average achieved FBR is higher than zero for both the basic and partial coding schemes. This suggests that data fusion is a universal process. In other words, by applying the proposed coding schemes, the fusibility between any two signals is not restricted in general.

7.5.2 Fusion Bit-Rate

Given two signals X and $\bar{X}$ to be fused by Ξ^2 such that $|X| > |\bar{X}|$, FBR (Fusion Bit-Rate) is the bpb ratio of the size of $\bar{X}$ to the size of X. Here, high FBR indicates that more data are fused. Hence, the higher FBR, the better data fusion performance. This ratio is measured based on the applied coding scheme as detailed in the next sub-sections.

7.5.2 (a) Fusion Bit-Rate in Basic Coding Scheme

In basic coding scheme, FBR is defined as

$$FBR = \frac{|X_L|}{|X|}, \tag{7.16}$$

Generally, since $|X_L|$ depends on the length L, changing L will affect FBR. Specifically, as L increases, the number of segments in $|X_L|$ decreases, and hence FBR decreases. On the other hand, as L increases, the side information $\mathscr{T}(\mathbf{S})$ increases as follows:

$$\mathscr{T}(\mathbf{S}) = 2^L \times L, \tag{7.17}$$

Figures 7.6 and 7.7 show the effect of changing L on FBR and $\mathscr{T}(\mathbf{S})$ in the encrypted Lenna test image. In Figure 7.6, it is shown that as L increases, FBR decreases. This is because the increment in L reduces the number of segments that are mapped to the EBI codewords. Hence, the venues for data fusion is reduced. Figure 7.7 shows that as L increases, the side information $\mathscr{T}(\mathbf{S})$ also increases as defined in Eq. (7.17). The side information increment causes file-size increment. Similar effect on FBR and $\mathscr{T}(\mathbf{S})$ are observed for other test signals when L changes. Based on Figures 7.6 and 7.7, the optimum performance in this coding scheme is achieved at a relatively small L. In particular, the optimum performance is achieved when $L = 6$ bits for all types of signal, except for raw images that achieve the best results when $L = 8$ bits. Table 7.5 records FBR achieved

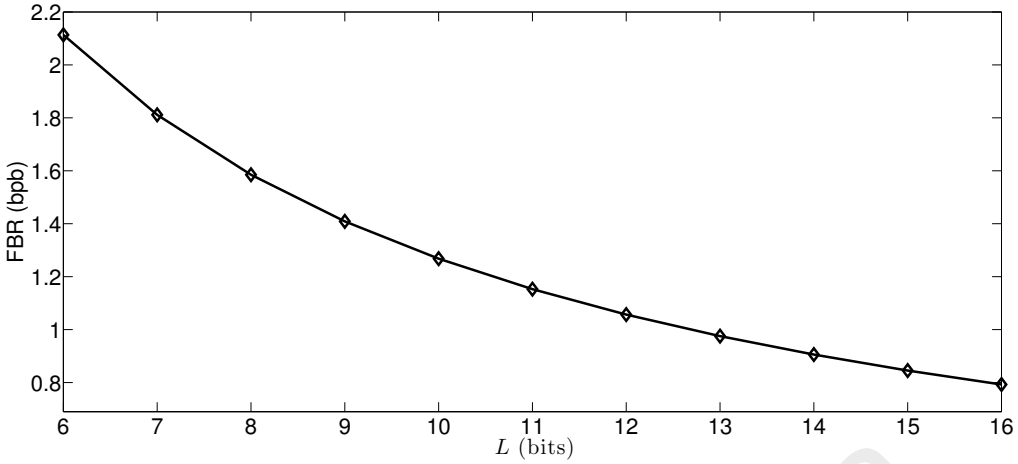


Figure 7.6: The effect of L on FBR in basic coding scheme for the encrypted Lenna test image

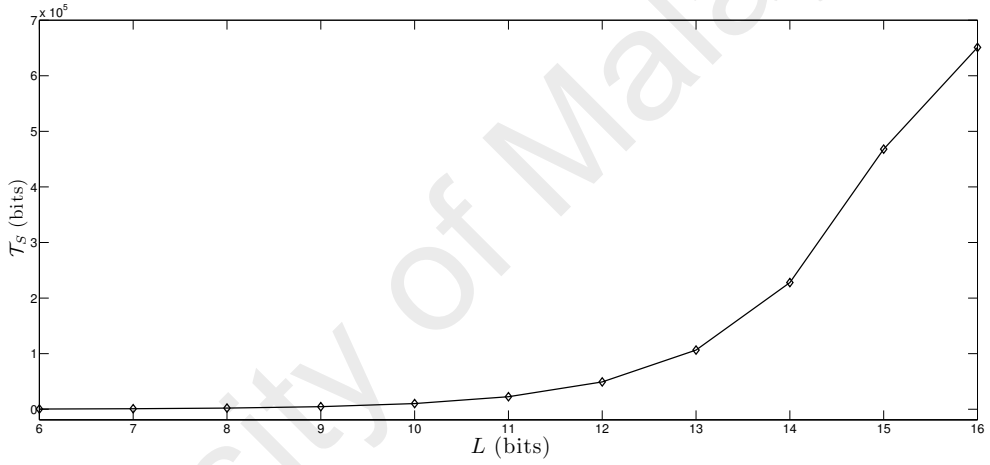


Figure 7.7: The effect of L on the size of side information in basic coding scheme for the encrypted Lenna test image

Table 7.5: FBR (bpb) for the three test images achieved by the proposed two coding schemes

Coding Method	Signal	Raw	Compressed	Encrypted
Basic scheme	Lenna	1.58500	2.11335	1.58500
	Baboon	3.17001	2.11333	1.58500
	Man	1.54532	2.11334	1.15899
Partial scheme	Lenna	0.00036	0.00046	0.03710
	Baboon	0.01905	0.00038	0.03340
	Man	1.58500	0.00023	0.02575

by applying the proposed basic coding scheme to the test images.

For raw images, the basic coding scheme achieves FBR that ranges from 1.58500 to 3.17000 bpb. FBR ranges from 2.11333 to 2.11335 bpb for compressed images, and FBR

ranges from 1.58500 to 1.15899 bpb for the encrypted images. It is worth mentioning that such trend of variation in FBR is also observed in all other test signals. It is observed that this variation in FBR is based on the size of each specific signal (i.e., $|X|$) and the value of L. However, the class and type of the considered signal has no effect on FBR.

7.5.2 (b) Fusion Bit-Rate in Partial Coding Scheme

In partial coding scheme, FBR is defined as

$$FBR = \frac{(|Q_C| \times log_2(3) + \Lambda) - \mathscr{T}(\mathbf{S})}{|X|}, \tag{7.18}$$

where $\mathscr{T}(\mathbf{S})$ is the size of the size information defined as

$$\mathscr{T}(\mathbf{S}) = (|Q_C| \times L) + L, \tag{7.19}$$

and Λ is defined as

$$\Lambda = \sum_{q=1}^{|X_L|} \Pi_{c_q}. \tag{7.20}$$

Here, the side information is concatenated to the smaller signal $\bar{X}$, and they are fused with the larger signal X. On the other hand, FBR in the partial coding scheme depends on the value of L in the universal parser. Figure 7.8 shows the effect of changing the length L on FBR for the Man test image encoded by the partial coding scheme. Here, as L increases, the cardinality $|X_L| = \lambda_L$ decreases, and hence the number of codewords decreases. Thus, FRB decreases (Eq. (7.18)) as L increases.

On the other hand, FBR depends on $|Q_C|$ as suggested by Figure 7.9. Here, when $|Q_C|$ increases, the side information $\mathscr{T}(\mathbf{S})$ increases, and hence FBR decreases (Eq. (7.18)). Similar effects of L and Q_C on FBR are observed for all test signals. Nevertheless, only the highest FBR of each test signal (results of parsing the signal at various L and setting various $|Q_C|$ values) is considered and presented hereinafter. All FBR results presented

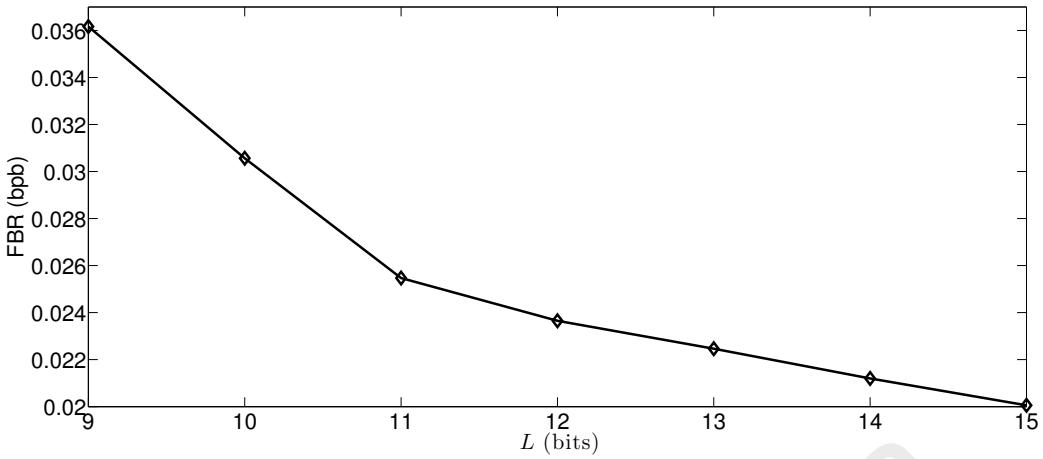


Figure 7.8: The effect of L on FBR in the partial coding scheme for the Man test image

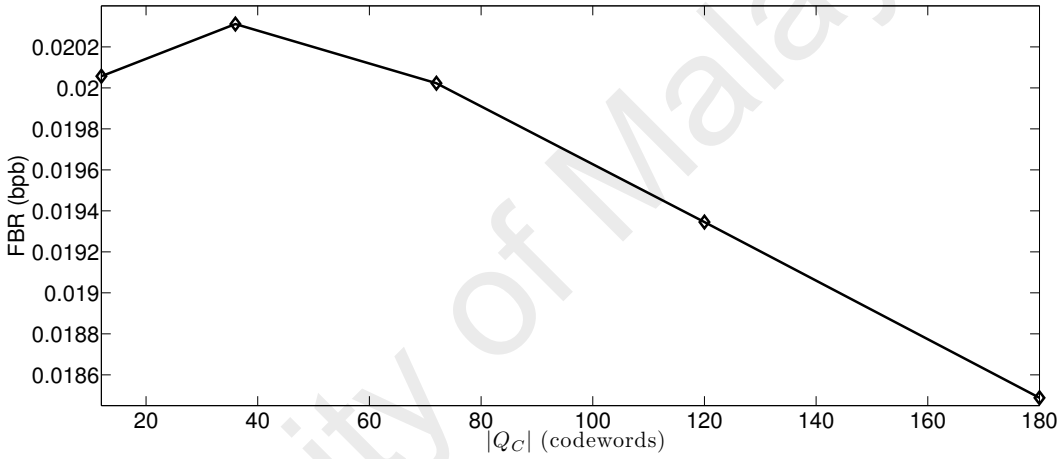


Figure 7.9: The effect of $|Q_C|$ on FBR in the partial coding scheme for the Man test image

in this sub-section are obtained when the proposed data fusion operates in the file-size preserving mode.

Table 7.5 records the FBR achieved by applying the proposed partial coding scheme to the test images. Here, for raw images, FBR ranges from 0.00036 to 1.58500 bpb. For compressed images, FRB ranges from 0.00023 to 0.00046 bpb, and for encrypted images, FBR ranges from 0.02575 to 0.03710 bpb. Such variation in FBR is also observed in other test signals in various formats and the range depends on $|Q_C|$ and Λ in Eq. (7.18). However, in the partial coding scheme, the effect of $\mathscr{T}(\mathbf{S})$ on FBR is insignificant because the size of the generated side information in this coding scheme is relatively small in comparison to that of the basic coding scheme as defined by Eq. (7.19).

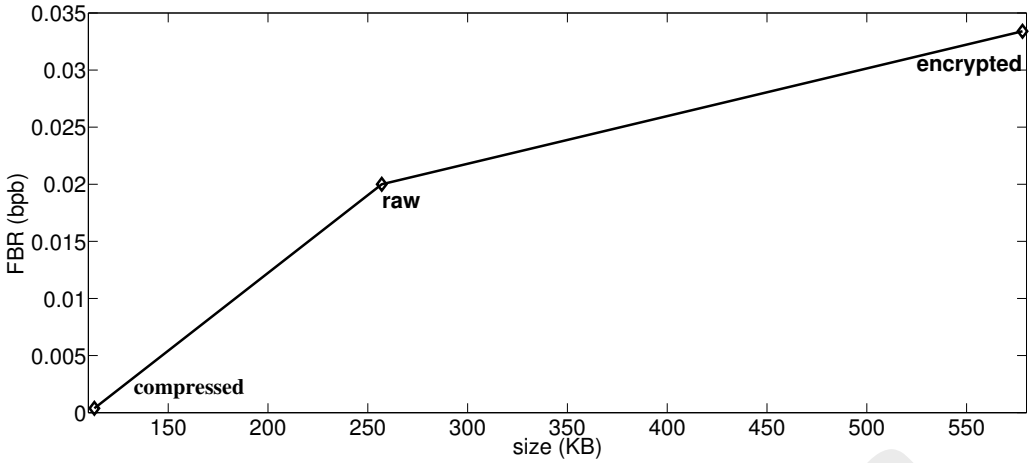


Figure 7.10: The graphs of FBR versus size of Baboon test signal

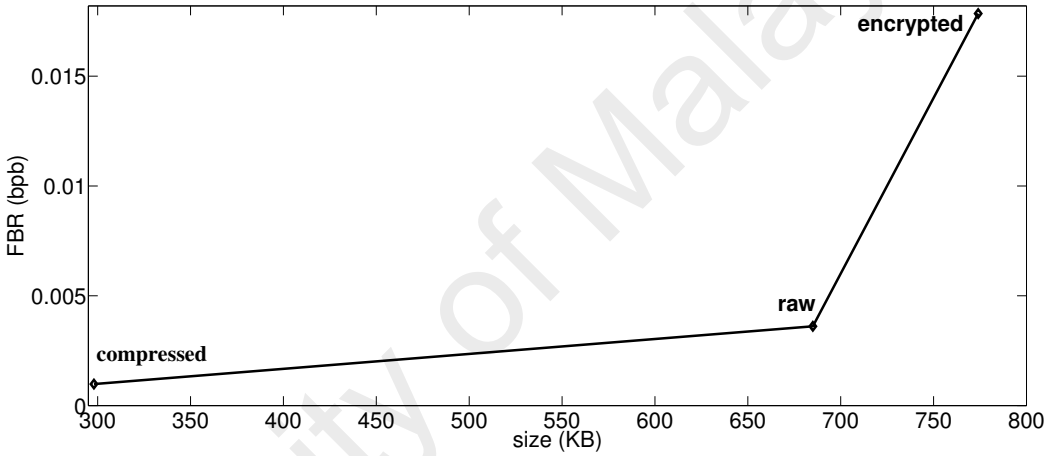


Figure 7.11: The graphs of FBR versus size of Foreman test signal

Nonetheless, for compressed texts, it is observed that its FBR is negative, i.e., FBR$<$0. In particular, in these signals, the FBR ranges from -0.00041 to 0.00023 bpb. This is an expected result because these signals are small in size. Hence, the probability of encountering two adjacent segments with identical pattern in $|X_L|$ is low (Section 7.4.2 (c)). Thus, only a few number of segments are mapped to EBI codewords, which results in a relatively small number of venues (i.e., in EBI codewords) for data fusion purposes. For that, the size of the side information is larger than the venues attained for data fusion, which lead to FBR$<$ 0 as shown in Eq. (7.18). Hence, data fusion using the partial coding scheme is not applicable to signals that are small in size. Instead, the basic coding scheme should be applied to achieve FBR$>$0.

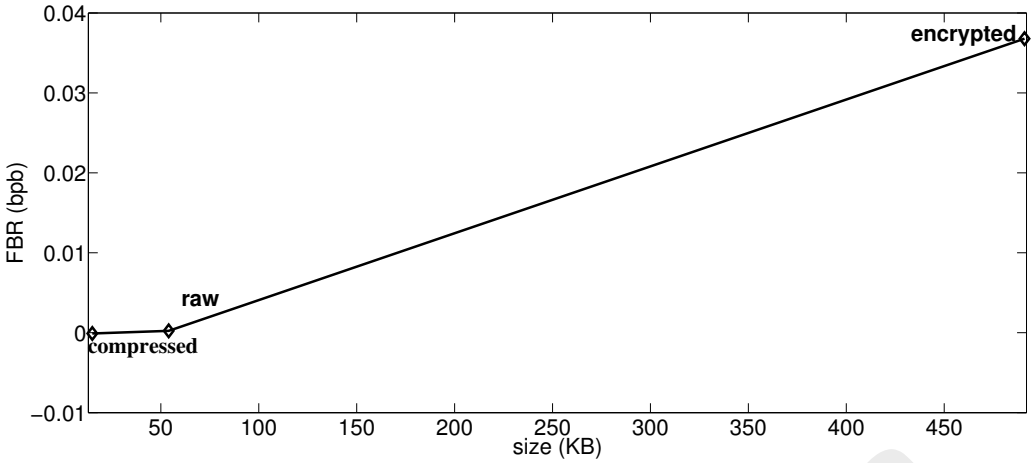


Figure 7.12: The graphs of FBR versus size of text1 test signal

In addition, in the partial coding scheme, it is observed that FBR is directly proportional to the size of the test signal. Such property is observed for all test signals, except for audio. On the other hand, generally, as the size of the signal increases, FBR increases accordingly. Figure 7.10 shows the graph of the size versus FBR in the Baboon image. Here, FBR is 0.00038 bpb when it is compressed, i.e., 113 KB in size. However, FBR increases to 0.02000 bpb in the raw image format, which is of size 257 KB. As such, the maximum FBR among the images is 0.03340 bpb, which is attained by the encrypted image that has the largest size (i.e., 578 KB). Similar trend is observed in all other signals (except audios). Figure 7.11 shows the graph of the size versus FBR in the encrypted foreman video, which is of the largest size among the raw, compressed and encrypted formats (i.e., 774 KB), and it achieves the maximum FBR of 0.01785 bpb. Figure 7.12 shows the graph of the size versus FBR in the encrypted text (i.e., Text1), which has the largest size, i.e., 491 KB, and it achieves the maximum FBR of 0.03678 bpb. Generally, FBR is directly proportional to the size because $|X_L|$ in Eq. (7.18) depends on the number of segments, i.e., λ_L.

In the case of audio signal (Figure 7.13), it is observed that FBR achieved by the compressed audio is larger than that of the raw audio, although the size of the compressed audio is smaller than that of the raw audio. This is because in the raw audio (i.e., wav for-

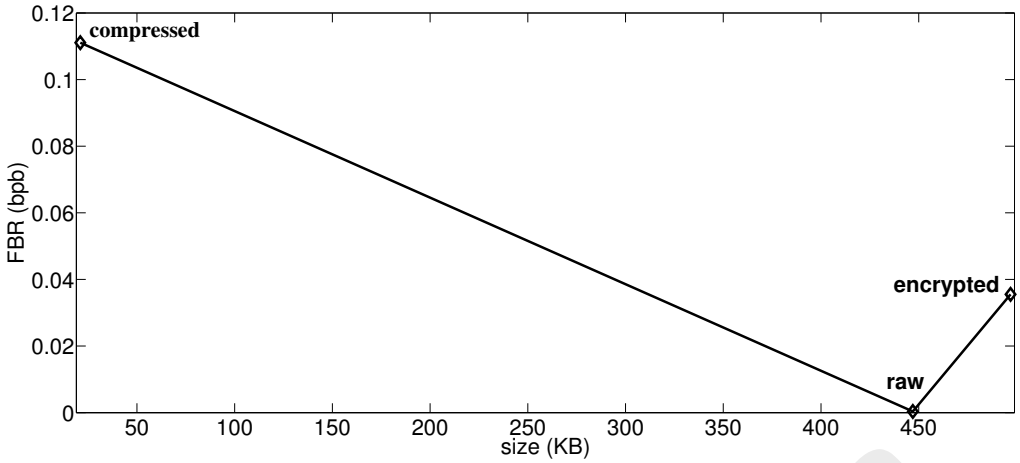


Figure 7.13: The graphs of FBR versus size of Audio1 test signal

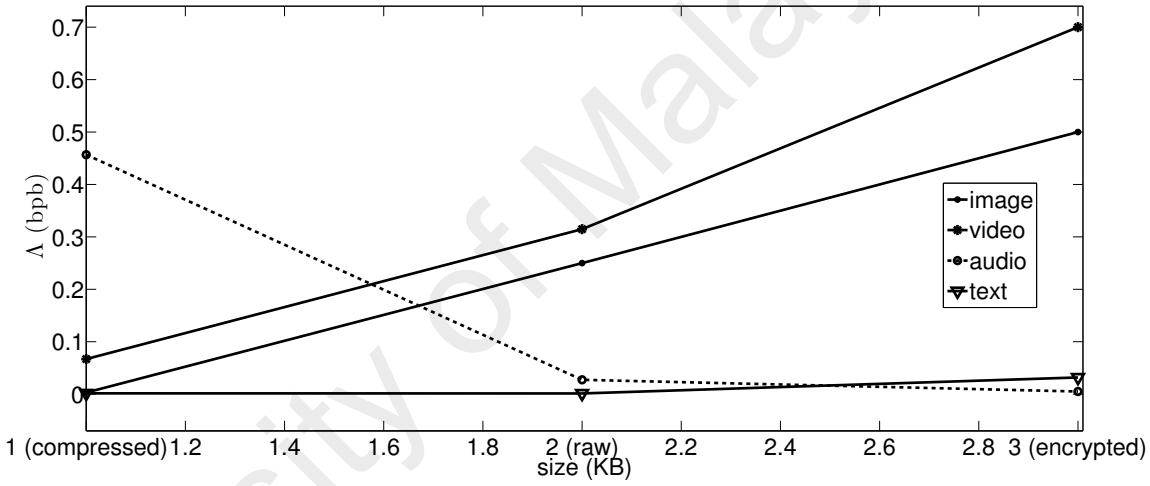


Figure 7.14: Graph of Λ versus the size of Baboon, Foreman, Audio1 and Text1 test signals

mat), the number of removed bits by compression Λ (Eq. 7.20) is relatively small due to its statistical features. Hence, in the case of raw audio, Λ contributes in a smaller amount to FBR when compared to that of the compressed audio. Figure 7.14 shows the amount of contribution of Λ to FBR (in bpb) for the same test signals in Figures 7.10, 7.11, 7.13 and 7.12. Here, Λ of the compressed audio is 0.45670 bpb. However, the raw audio has a lower rate of 0.02740 bpb. This is because the tuples of the compressed audio in X_L are concentrated around few patterns in the histogram of X_L, which increases Λ (i.e., more compression is gained) when these tuples are encoded by DSC. For that, FBR of the compressed audio in Figure 7.13 is higher than that of the raw audio.

For other signals (i.e., except audio), Λ increases as the size of the signal increases. For example, the compressed image and video have lower Λ (0.00180 and 0.00300 bpb, respectively) than that of the raw image and video (0.25000 and 0.31470 bpb, respectively) as suggested by Figure 7.14. Therefore, for the images and videos considered, the compressed versions offer lower FBR when compared to their raw counterparts.

7.5.2 (c) Comparison of FBR between The Basic and Partial Coding Schemes

Figure 7.15 shows the graph of the average FBR of the test signals achieved by the basic and partial coding schemes. Generally, it is observed that FBR of the basic coding scheme is higher than that of the partial coding scheme. In particular, it is concluded that, on average, the basic and partial coding schemes achieve FBR of 0.640 and 0.060 bpb, respectively. Here, the basic coding scheme achieves higher FBR than the partial coding scheme because all tuples of the set X_L are mapped to EBI codewords in the basic coding scheme where each codeword accommodates data from $\bar{X}$. On the other hand, in the partial coding scheme, less tuples from X_L are mapped to EBI codewords. Hence, the basic coding scheme is superior to the partial coding scheme in terms of FBR performance. This suggests that, for applications where the amount of data to be fused is high, it is recommended to deploy the basic coding scheme.

7.5.3 Compressibility

Essentially, data fusion does not aim at achieving data compression. However, gaining compression while fusing data is considered as an extra advantage. The partial coding scheme can be switched from the file-size preserving mode to the compression mode by setting $\Lambda = 0$ in Eq. (7.18). Table 7.6 records the average compression ratio of all test signals manipulated by the partial coding scheme, where all compression ratios are greater than 1. These results suggest that gaining compression while fusing data can *always* be achieved by setting $\Lambda = 0$. However, in this coding scheme, FBR reduces because the

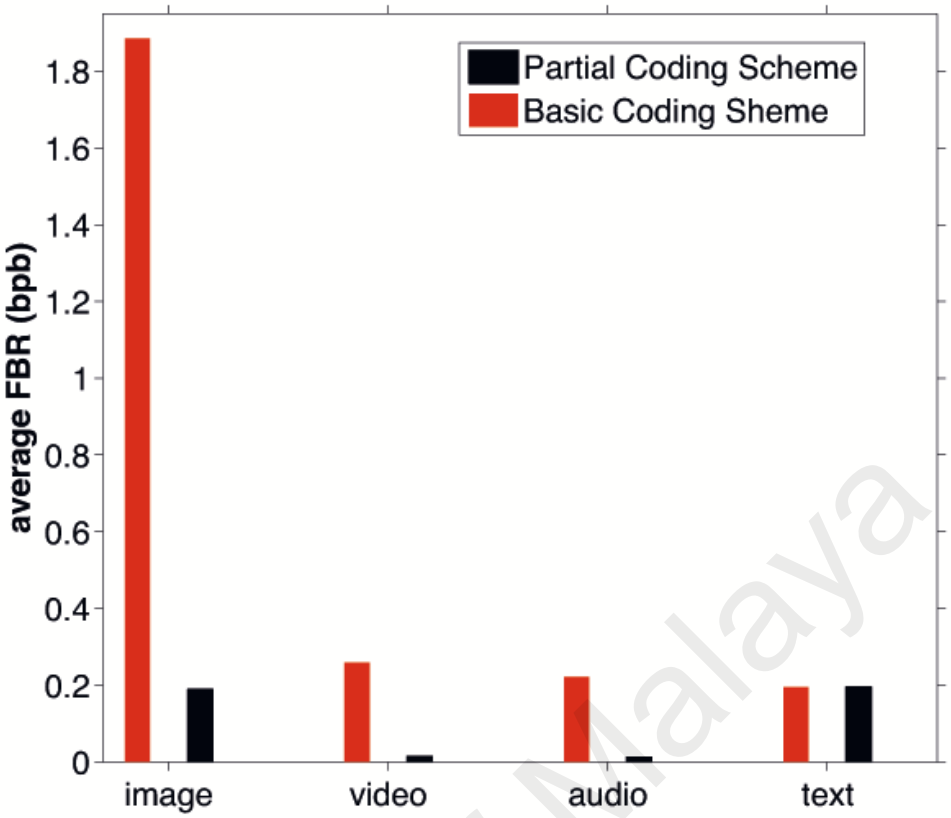


Figure 7.15: Comparison between the basic coding scheme and the partial coding scheme in terms of average FBR

removed bits are not utilized for data fusion. For example, in the raw Lenna image, at $L = 10$ and $|Q_C| = 12$, FBR=0.00019 bpb while achieving compression. However, at the same L and $|Q_C|$, this image achieves FBR=0.00036 bpb in the file-size preserving mode. This indicates that FBR drops by $\sim 47.2\%$ due to compression. The drop of FBR depends on the statistical features of each signal considered. Generally, it is observed in Table 7.6 that, on average, FRB drops for image, video, audio and text signals by $57.9\%, 41.7\%$, 70.4% and 53.5%, respectively. Hence, the trade-off is between FBR and compression ratio.

7.6 Summary

In this chapter, the conventional concept of data embedding was generalized by the proposed concept of data fusion. In particular, the proposed data fusion establishes a universal framework for merging associated data into a single entity. Specifically, the universal

Table 7.6: Average compression ratios, FBR (in bpb) and the drop in FBR (%) for all test signals when considering the compression mode in the partial coding scheme.

	Compression Ratio	FBR	Drop in FBR (%)†
Lenna	1.01049	0.08823	51.0
Baboon	1.01257	0.09621	45.9
Man	1.03513	0.01590	76.9
Foreman	1.00490	0.00247	33.3
Carphone	1.00803	0.00664	46.1
Suzie	1.01042	0.00808	45.6
Audio1	1.00528	0.00443	57.8
Audio2	1.01158	0.01075	75.5
Audio3	1.01246	0.01365	77.9
Text1	1.01929	0.00980	65.2
Text2	1.02495	0.01905	55.4
Text3	1.02454	0.01877	55.4

†The drop is relative to the results obtained when considering the file size preserving mode.

interchangeability property is attained by defining the associated data and carrying out data fusion in the universal domain. Then, the universal parser (Section 3.3.3) models the associated data, followed by encoding them using dual semantic code. Two coding schemes were proposed, namely, the basic and partial coding schemes. Experimental results verified that signals of various classes can be fused by using both the proposed basic and partial coding schemes. Therefore, data fusion is achievable for any two sets of digital signal. The basic coding scheme achieves an average data fusion bit rate of 0.640 bpb. The partial coding scheme achieves an average FBR of 0.060 bpb with a trade-off between data fusion bit rate and file size. In particular, the partial coding scheme offers three operating modes, namely: (1) file-size-preserving; (2) compression, and; (3) partial compression. Notably, a compression ratio of up to 1.03513 is observed while achieving a FBR of 0.01590 bpb.

CHAPTER 8

DISCUSSIONS

8.1 Overview

In this section, the performance of the proposed methods in Chapters 4∼7 are discussed. The discussion here focuses mainly on verifying the achievement of the objectives stated in Chapter 1. In particular, it is verified that the proposed methods are universal, reversible and file-size preserving. In addition, the achieved carrier capacities of the proposed methods are compared.

8.2 Interchangeability

The theoretical and empirical verifications throughout this study show that the proposed uREADS, urDEED, DeRand and data fusion methods are universal data embedding methods. Theoretically, these methods exploits only two features, which are common among all signals, to achieve data embedding. These features are: (1) the size of the considered signal, and; (2) the position of its alphabets. Defining these two features for any signal X is sufficient for parsing by the universal parser. In other words, except these two features, the proposed methods operate independently from all features defined in certain domain, media or coding structure. Empirically, the experiments performed on various classes of signal verify the applicability of the universal parser and the proposed methods to any digital signal.

8.3 Carrier Capacity

The experimental results presented in Sections 4.4, 5.6.3, 6.6.1 and 7.5.2 show that all proposed methods are able to achieve some positive effective carrier capacity for any input signal X. This suggests that the proposed methods successfully define venues for

Table 8.1: Comparison of the maximum carrier capacity obtained by the proposed methods

Method	Carrier Capacity (bpb)
DeRand (Chapter 6)	0.405
urDEED (Chapter 5)	0.174
uREADS (Chapter 4)	0.074
Data Fusion (Chapter 7)	0.060

payload insertion regardless of the features of the considered signal. However, the carrier capacity varies among the proposed methods as shown in Table 8.1. Here, the maximum bpb value in the file-preserving mode of each corresponding method is recorded for direct comparison purposes. In general, DeRand records the maximum bpb value because this method is based on histogram mapping where no look-up table is required for perfect restoration of the original host. Thus, the amount of side information (required for reversibility) is relatively low in comparison to other methods, such as uREADS, urDEED and data fusion. Among methods that depend on look-up table, urDEED achieves the best bpb value of 0.174. This is because urDEED processes the side information of GRC's in a more efficient manner in comparison to the most relevant method, i.e., uReads. Finally, data fusion records the minimum bpb value of 0.060. This low bpb value is due to the dependency of urDEED on DSC's, which consists of lengthy codewords that cannot be utilized for data fusion in file-size preservation mode, i.e., using the partial coding scheme (Section 7.4.2 (c)). Consequently, as the number of utilized codewords decreases, the resultant FBR (in bpb) is decreased. These results suggest that, DeRand is superior to other methods in terms of carrier capacity.

8.4 Reversibility

It was verified that all the aforementioned proposed methods are revisable, where the original host can be perfectly restored from its modified counterpart. This reversibility is achieved at the cost of storing side information. For example, uREADS, urDEED and data fusion generate look-up tables, which are embedded along with the payload (or the

signal of the smallest size in data fusion), as detailed in Sections 4.3.3 (a), 5.4.2 and 7.4.2 (c), respectively. DeRand generates side information, which consists of the set of mapped bins in the histogram, as shown in Section 6.5.3. The embedding of side information along with the payload reduces the effective carrier capacity.

8.5 File-Size Preserving

Conventionally, reversible data embedding methods aim at minimizing file-size expansion. However, the file-size is likely to be expanded due to reversible data embedding, particularly when a compressed content is utilized as the host (Mobasseri et al., 2010; Wong et al., 2009; Xuan et al., 2007). Practically, in the scenario where the storage medium is limited in capacity (e.g., 4.7GB for a single-layer DVD, a USB flash memory) or cost (e.g., expensive cellular network), an expansion in bitstream size caused by data embedding may require additional storage medium or higher network traffic.

In the proposed methods, it was verified that the size of the modified host (or signal of the largest size in data fusion) is exactly of the same size as its original counterpart, except DeRand which causes an average file-size increment by 0.41%. Hence, the file-size preservation property is successfully achieved while achieving universal data embedding. However, in some proposed methods such as uREADS and urDEED (Sections 4.3 and 5.4), file-size preservation is an achieved at the expense of longer execution time for data embedding. In particular, uREADS and urDEED need to parse the host three times in order to embed data, where the final pass is to process and embed excessive data (if any) to preserve the file-size. In data fusion, file-size preservation is achieved at the expense of lower FBR because some DSC's cannot be utilized as they cause file-size increment (Section 7.4.2 (c)).

8.6 Summary

In this study, a universal domain was first established so that any digital signal can be represented in it. Next, the concept and implementation of universal parser was proposed as a mean to define redundancy in any signal when represented in the universal domain. This redundancy is exploited to embed data universally in any signal based on the proposed framework for universal data embedding. This framework established the foundation of four proposed universal data embedding methods, namely, uREADS, urDEED, DeRand, and the proposed data fusion method. Generally, the main objectives of this study were attained. In particular, the proposed methods are interchangeable and universally applicable to any digital signal. Furthermore, the proposed methods are reversible hence inserting/-fusing data causes no data loss. In terms of carrier capacity, all the proposed methods offer some amount of positive carrier capacities. Hence, defining venues for data embedding/fusion is achievable by the proposed methods. The carrier capacities vary among the proposed methods, where DeRand achieves the highest carrier capacity. Finally, the proposed methods are able to modify the input signal X to accommodate/fuse additional data without increasing its file-size.

CHAPTER 9

CONCLUSIONS AND FUTURE WORK

9.1 Conclusions

Data embedding is the process of modifying features of a host in order to embed a payload. Conventionally, data embedding is a feature-dependent process and hence each data embedding method is specifically designed to modify a host of certain features. The feature-dependency of the conventional data embedding methods restricts their interchangeability. Hence, each method operates solely in a certain domain, medium or coding format. The restricted interchangeability of the conventional data embedding methods limits the applicability of these methods, hence multiple data embedding methods are required to embed data in different multimedia contents. As digital multimedia contents are massively being generated in various formats, the interchangeability of data embedding should be considered. In this study, the problem of the limited interchangeability among most of the conventional data embedding methods was considered. Generally, in the current literature, a universal data embedding is nonexistent.

This study aimed at studying the novel concept and implementation of universal data embedding, which can be interchangeably applied to any digital signal. Generally, universal data embedding methods were designed to operate in a domain referred to as the universal domain in which all digital signals live. The general framework of universal data embedding is proposed, and it consists of four stages, namely, defining the signal in the universal domain, universal parsing, mapping and data embedding. Generally, any signal is defined in the universal domain by extracting two features, namely, the size of the signal and the unique position of the symbols of the signal. The universal parser segments the signal

into partitions of unified length. Theoretically, it has been proven that this partitioning process changes the entropy and hence redundancy of the signal (based on the length L), where this redundancy can be replaced by the payload. The mapping and data embedding were achieved by four proposed methods, namely, uREADS, urDEED, DeRand and data fusion

The proposed uREADS (Chapter 4) is based on mapping the universally parsed signal to a set of modified GRC's, which are processed to accommodate a payload. uREADS embeds external information and excessive zeros (if any) in the quotient part of GRC using Algorithm 1 in order to preserve the size of the modified signal. This method is completely reversible and universally applicable to any digital signal. In addition, uREADS achieves on average carrier capacity of 0.030 bpb. However, in uREADS, some parameters, such as L and R are manually tuned.

The proposed urDEED (Chapter 5) is based on mapping the universally parsed signal to a set of modified GRC's. Unlike uREADS, urDEED proposes a novel scheme to embed the excessive zeros in the quotient part along with the payload. This scheme allows the definition of constant optimum values for L and r, which are 3 and 1, respectively. urDEED can be effectively applied to the encrypted signals, viz., urDEED operates solely in the encrypted domain. In contrast, the conventional data embedding methods in the encrypted domain, such as the method proposed by (Zhang, 2012), requires partial decryption or knowledge of the features of the signal prior to the encryption. Hence these methods are not solely operational in the encrypted domain. In addition, unlike Zhang's method (Zhang, 2012), urDEED can operate independently from the encryption scheme applied to the signal. In other words, urDEED can operate in any signal encrypted by any encryption scheme. Also, urDEED is superior to Zhang's method (Zhang, 2012) in terms reversibility, universal applicability and carrier capacity, where urDEED achieves an average carrier capacity of ~ 0.169 bpb while ensuring perfect reconstruction of the

original signal. However, urDEED is not applicable to signals of high entropy such as random signals because when urDEED is applied to these signals, all venues for data embedding is occupied by the side information.

The proposed DeRand (Chapter 6) is based on histogram mapping. DeRand overcomes the inability of urDEED to embed data in high entropy signals. Practically, DeRand is applicable to any signal, including those that are random in natural or of undefined feature, such as signals exchanged over the communication networks. DeRand applies the universal parser recursively to define redundancy in a signal regardless of its original statistical features. Theoretically, it is shown that the universal parser can certainly define empty bins in the histogram of any signal. The defined empty bins are utilized for data embedding, where non-vacant bins are mapped to vacant bins. It was verified that the proposed DeRand can successfully embed data in high entropy signals such as randomly generated signals. DeRand is reversible and universally applicable to any signal X. In addition, its performance in terms of carrier capacity and visual quality is scalable. The maximum carrier capacity obtained at file-size preserving mode is 0.405 bpb at SSIM=0.262.

Data fusion (Chapter 7) is proposed as a novel concept that generalizes the concept of data embedding. In particular, data fusion realizes the concept of associating any two (or more) signals into a single entity regardless their features. Hence, data fusion is not limited by the concept of data embedding, which is embedding a payload into a host by means of modifying the host's features. In addition, the number of signals to fuse can exceed two signals. The applications of data fusion can be tailored for metadata management, watermarking, fingerprinting and any application that processes associated data. Data fusion is achieved in the universal domain, where the universal parser is applied to define redundancy in the signal of the largest size (between the two involved in data fusion). The tuples obtained by the universal parser are mapped to DSC, which is a novel entropy coding in which case each codeword can accommodate two independent

data simultaneously. The mapping to DSC is achieved by two proposed coding schemes, namely, basic and partial coding schemes. The basic coding scheme is based on mapping all tuples to DSC's. This may cause a file-size increment. However, the average FBR for this coding scheme is 0.640 bpb. The partial coding scheme is based on the selection of some tuples for mapping DSC's such that no file-size increment occurs. Here, three modes are achievable by using this coding scheme, namely, file-size preserving, compression and partial compression. In the file-size preserving mode, the average FBR is 0.060 bpb. In the compression mode, it is observed that the compression ratio is up to 1.03513 while achieving a FBR of 0.01590 bpb.

Generally, all the proposed methods are reversible and file-size preserving. The performance of the proposed methods varied in terms of carrier capacity. Last but not least, all the proposed methods are universal and interchangeably applicable to any signal.

9.2 Future Work

Generally, the contributions made in this study can be exploited to extend the applications of data embedding. Particularly, the proposed universal data embedding methods can be deployed to achieve the applications of data association. In addition, the practical utilization of data fusion in metadata management shall be considered. Also, DSC can be potentially applied in the security related areas, such as steganography. Furthermore, DSC can be invoked as an entropy coding scheme for data compression. Technically, the generated side information by the proposed methods should be reduced in order to increase the effective carrier capacity. Last but not least, the complexity of the universal data embedding methods should be decreased for the deployment on smart tablet, smart phone, and battery-powered devices.

Appendices

APPENDIX A

LIST OF ACRONYMS AND ANNOTATIONS

Table A.1: LIST OF ACRONYMS AND ANNOTATIONS

Symbol	Meaning
X	The input signal
x_g	A symbol of position g in X
$\mathbb{F}$	The universal set of features
F	A subset of features
$\mathbb{A}$	A set of alphabets
$\mathbb{A}_L$	A set of all possible arrangements of alphabets (derived from $\mathbb{A}$ of length L)
$\mathbb{U}_{\mathbb{A}}$	The universal domain of alphabets derived from $\mathbb{A}$
N	The size of the input signal X
IC	Imaginary Codeword
λ_L	The total number of tuples of length L
L	The partitioning length in the universal parser
T_t^L	A tuple of L symbols of position t in X_L
Θ_{avg}	The average length of codewords that encode X_L
$\mathbb{D}$	The ordering function in the universal parser
$P(X)$	The probability of X
$H(X)$	The entropy of X
$I(X)$	The self-information of X
$\theta(T_v^L)$	The length of a codeword required to encode T_v^L

Continued on Next Page...

Table A.1 – Continued

Symbol	Meaning
ρ_L	The amount of redundancy in X_L
GRC	Golomb-Rice Code
DSC	Dual Semantic Code
MRI	Measurement of Relative Interchangeability
$\mathbf{C}_{raw}$	The raw carrier capacity
$\mathbf{C}_e$	The effective carrier capacity
μ	The number of possible models to which X can be transformed to
ζ	A non-ideal coding scheme
σ	A parsing coefficient
$\mathbf{S}$	The side information
j	The length of quotient part in GRC
κ	The length of trimmed quotient parts in GRC's (urDEED)
M	The number of IC's of count "0" in the histogram (urDEED)
ρ	A scalability parameter in DeRand
A_L	A set of tuples in X_L of length ρ in DeRand
B_L	A set of tuples in $\dot{X}_L$ of length ρ in DeRand
GE	General Expression
ζ	The size of EBI codeword
ω	The number of occurrences of term d in GE
μ	The number of possible models to which X can be transformed to
O	The order of data association
$\mathbb{P}$	The set of all possible binary sequences of length n

Continued on Next Page...

Table A.1 – Continued

Symbol	Meaning
ψ_ω	The total number of codewords generated by a set of GE
Ξ	The fusion function
$\mathfrak{S}$	The state of y in EBI
$\mathscr{T}(X)$	The size (in bits) of X
Π_{c_q}	The amount of removed bits due to mapping T_t^L to c_q
C	A set of DSC codewords
c_q	A single codewords of position q in C
Q_C	A set consists of all EBI of $\mathscr{T}(c_q) < L$
Λ	The total number of removed bits in the partial coding scheme
R	The amount of redundancy
d	Starting bits in DSC
z	Information bits in DSC
y	Ending bits in DSC

APPENDIX B

LIST OF PUBLICATIONS

Journals:

[1] M. S. Abdul Karim, K.Wong, Universal data embedding in encrypted domain, Signal Processing 94 (2014)174−182.

[2] M. S. Abdul Karim, K.Wong, Data fusion in universal domain using dual semantic code, Information Sciences 283 (0) (2014) 123−141.

[3] M. S. Abdul Karim, K.Wong, Universal data embedding in encrypted domain, Signal Processing 94 (2014)174−182.

International Peer-Reviewed Conferences:

[1] M. S. Abdul Karim, K.Wong, A novel paradigm of lossless image compression by reversible data embedding, in: Image Electronics and Visual Computing Workshop (IEVC2014), 2014.

[2] M. S. Abdul Karim, K.Wong, Re-conceptualization of applications achieved by data hiding, in: International Symposium on Intelligent Signal Processing and Communications Systems (ISPACS), IEEE, 2012, pp.447−451.

[3] M. S. Abdul Karim, K. S. Wong, K. Tanaka, Reversible data embedding for any digital signal, in: 5th International Symposium on Communications Control and Signal Processing (ISCCSP), IEEE, 2012, pp. 1−4.

[4] M. S. Abdul Karim, K. Wong, K. Tanaka, Reversible data embedding in Golomb-Rice code, in: IEEE International Conference on Signal and Image Processing Applications (ICSIPA), IEEE, 2011, pp. 515−519.

[5] K. Wong, M. S. Abdul Karim, K. Tanaka, Improvements of data embedding in DCT

compressed domain, in: IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PacRim), IEEE, 2011, pp. 268−273.

[6] M. S. Abdul Karim, K.Wong, Improvements of duplication free run-length coding for textured images, in: Image Electronics and Visual Computing Workshop (IEVC2012), 2012.

[7] M. S. Abdul Karim, K.Wong, A lossless image compression algorithm using duplication free run-length coding, in: Second International Conference on Network Applications Protocols and Services (NETAPPS), IEEE, 2010, pp. 245−250.

REFERENCES

Adams, C., Heys, H., Tavares, S., & Wiener, M. (1999). An analysis of the CAST-256 cipher. In *IEEE canadian conference on electrical and computer engineering* (Vol. 1, p. 361 -366 vol.1).

Alattar, A. M. (2004). Reversible watermark using the difference expansion of a generalized integer transform. *IEEE Transactions on Image Processing*, *13*(8), 1147–1156.

Al-Wahaib, M., & Wong, K. (2010). A lossless image compression algorithm using duplication free run-length coding. In *Second international conference on network applications protocols and services (netapps),* (p. 245-250).

Bassia, P., Pitas, I., & Nikolaidis, N. (2001). Robust audio watermarking in the time domain. *IEEE Transactions on Multimedia*, *3*(2), 232–241.

Bhat, V., Sengupta, I., & Das, A. (2010). An adaptive audio watermarking based on the singular value decomposition in the wavelet domain. *Digital Signal Processing*, *20*(6), 1547–1558.

Blake, M. E., & Knudson, F. L. (2002). Metadata and reference linking. *Library Collections, Acquisitions, and Technical Services*, *26*(3), 219 - 230.

Borges, P., Mayer, J., & Izquierdo, E. (2008). Robust and transparent color modulation for text data hiding. *IEEE Transactions on Multimedia*, *10*(8), 1479-1489.

Cancellaro, M., Battisti, F., Carli, M., Boato, G., Natale, F. D., & Neri, A. (2011). A commutative digital image watermarking and encryption method in the tree structured haar transform domain. *Signal Processing: Image Communication*, *26*(1), 1 - 12.

Celik, M. U., Sharma, G., Tekalp, A. M., & Saber, E. (2005). Lossless generalized-LSB data embedding. *IEEE Transactions on Image Processing*, *14*(2), 253–266.

Chang, C.-C., Chen, T.-S., & Chung, L.-Z. (2002). A steganographic method based upon JPEG and quantization table modification. *Information Sciences*, *141*(1–2), 123 - 138.

Chang, C.-C., & Kieu, T. D. (2010). A reversible data hiding scheme using complementary embedding strategy. *Information Sciences*, *180*(16), 3045 - 3058.

Chang, C.-C., Lin, C.-C., Tseng, C.-S., & Tai, W.-L. (2007). Reversible hiding in DCT-based compressed images. *Information Sciences*, *177*(13), 2768 - 2786.

Cheddad, A., Condell, J., Curran, K., & Kevitt, P. M. (2010). Digital image steganography: Survey and analysis of current methods. *Signal Processing*, *90*(3), 727 - 752.

Chen, B., Zhang, W., Ma, K., & Yu, N. (2013). Recursive code construction for reversible data hiding in DCT domain. *Multimedia Tools and Applications*, 1–25.

Chen, B., Zhang, W., & Yu, N. (2013). Reversible watermarking in JPEG images based on modified RZL codes and histogram shift. *Wuhan University Journal of Natural Sciences*, *18*(2), 126-132.

Chen, Q., Xiang, S., & Luo, X. (2013). Reversible watermarking for audio authentication based on integer DCT and expansion embedding. In *Proceedings of the 11th international conference on digital forensics and watermaking* (pp. 395–409). Berlin, Heidelberg: Springer-Verlag.

Coltuc, D. (2012). Low distortion transform for reversible watermarking. *IEEE Transactions on Image Processing*, *21*(1), 412–417.

Coltuc, D., & Chassery, J.-M. (2007). Very fast watermarking by reversible contrast mapping. *IEEE Signal Processing Letters*, *14*(4), 255–258.

Cover, T. M., & Thomas, J. A. (1991). *Elements of information theory*. New York, NY, USA: Wiley-Interscience.

Cox, I. J., Kilian, J., Leighton, T., & Shamoon, T. (1996). A secure, imperceptible yet perceptually salient, spread spectrum watermark for multimedia. In *Southcon/96. conference record* (pp. 192–197).

Dittmann, J., Steinebach, M., & Ferri, L. (2002). Watermarking protocols for authentication and ownership protection based on timestamps and holograms. In *Proc. spie* (Vol. 4675, pp. 240–251).

Fallahpour, M. (2008). Reversible image data hiding based on gradient adjusted prediction. *IEICE Electronics Express*, *5*(20), 870–876.

Fan, J., Cao, H., & Kot, A. (2013). Estimating exif parameters based on noise features for image manipulation detection. *IEEE Transactions on Information Forensics and Security*, *8*(4), 608-618.

Fridrich, J., Goljan, M., & Du, R. (2001). Invertible authentication. In *Photonics west 2001-electronic imaging* (pp. 197–208).

Fujiyoshi, M., Sato, S., Jin, H. L., & Kiya, H. (2007). A location-map free reversible data hiding method using block-based single parameter. In *IEEE international conference on image processing, 2007. icip 2007.* (Vol. 3, pp. III–257).

Gal, E., & Toledo, S. (2005). Algorithms and data structures for flash memories. *ACM Comput. Surv.*, *37*(2), 138–163.

Gao, X., An, L., Yuan, Y., Tao, D., & Li, X. (2011). Lossless data embedding using generalized statistical quantity histogram. *IEEE Transactions on Circuits and Systems for Video Technology*, *21*(8), 1061–1070.

Gogniat, G., Wolf, T., Burleson, W., Diguet, J.-P., Bossuet, L., & Vaslin, R. (2008). Reconfigurable hardware for high-security/ high-performance embedded systems: The safes perspective. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, *16*(2), 144 -155.

Golomb, S. (1966). Run-length encodings. *IEEE Transactions on Information Theory*, *12*(3), 399-401.

Gonzalez, R. C., & Woods, R. E. (2002). *Digital image processing.* Prentice hall Upper Saddle River, NJ:.

Hartung, F., & Girod, B. (1998). Watermarking of uncompressed and compressed video. *Signal processing*, *66*(3), 283–301.

Hong, W., Chen, T.-S., Chang, Y.-P., & Shiu, C.-W. (2010). A high capacity reversible data hiding scheme using orthogonal projection and prediction error modification. *Signal Processing*, *90*(11), 2911–2922.

Hong, W., Chen, T.-S., & Shiu, C.-W. (2009). Reversible data hiding for high quality images using modification of prediction errors. *Journal of Systems and Software*, *82*(11), 1833–1842.

Hong, W., Chen, T.-S., & Wu, H.-Y. (2012). An improved reversible data hiding in encrypted images using side match. *Signal Processing Letters, IEEE*, *19*(4), 199 -202.

Hu, Y., Lee, H.-K., & Li, J. (2009). De-based reversible data hiding with improved overflow location map. *IEEE Transactions on Circuits and Systems for Video Technology*, *19*(2), 250–260.

Huang, J., Shi, Y. Q., & Shi, Y. (2000). Embedding image watermarks in dc components. *IEEE Transactions on Circuits and Systems for Video Technology*, *10*(6), 974–979.

IACP. (2014). Center for social media [Computer software manual]. (`http://www.iacpsocialmedia.org/Resources/FunFacts.aspx`)

Jeszenszky, P. (2007). Adding XMP support to Firefox. *Acta Cybern.*, *18*(2), 257–274.

Jung, S.-W., Ha, L. T., & Ko, S.-J. (2011). A new histogram modification based reversible data hiding algorithm considering the human visual system. *Signal Processing Letters, IEEE*, *18*(2), 95-98.

Kamran, Khan, A., & Malik, S. A. (2014). A high capacity reversible watermarking approach for authenticating images: Exploiting down-sampling, histogram processing, and block selection. *Information Sciences*, *256*(0), 162 - 183.

Karim, M. S. A., & Wong, K. (2014). Universal data embedding in encrypted domain. *Signal Processing*, *94*(0), 174 - 182.

Kim, C. H. (2012). Improved differential fault analysis on AES key schedule. *IEEE Transactions on Information Forensics and Security*, *7*(1), 41 -50.

Kim, H. J., Sachnev, V., Shi, Y. Q., Nam, J., & Choo, H.-G. (2008). A novel difference expansion transform for reversible data embedding. *IEEE Transactions on Information Forensics and Security*, *3*(3), 456–465.

Kim, K.-S., Lee, M.-J., Lee, H.-Y., & Lee, H.-K. (2009). Reversible data hiding exploiting spatial correlation between sub-sampled images. *Pattern Recognition*, *42*(11), 3083–3096.

Kinsella, S., Passant, A., & Breslin, J. (2011). Topic classification in social media using metadata from hyperlinked objects. In P. Clough et al. (Eds.), *Advances in information retrieval* (Vol. 6611, p. 201-206). Springer Berlin Heidelberg.

Lee, S.-K., Suh, Y.-H., & Ho, Y.-S. (2006). Reversible image authentication based on watermarking. In *IEEE international conference on multimedia and expo* (pp. 1321–1324).

Lei, B. Y., Soon, I. Y., & Li, Z. (2011). Blind and robust audio watermarking scheme based on SVD-DCT. *Signal Processing*, *91*(8), 1973 - 1984.

Leuf, B. (2004). *Semantic web*. Addison-Wesley Longman, Limited.

Li, M., Lei, Y., Liu, J., & Yan, Y. (2006). A novel audio watermarking in wavelet domain. In *International conference on intelligent information hiding and multimedia signal processing, 2006. IIH-MSP'06.* (pp. 27–32).

Li, X., Li, B., Yang, B., & Zeng, T. (2013). A general framework to histogram-shifting-based reversible data hiding.

Li, X., Yang, B., & Zeng, T. (2011). Efficient reversible watermarking based on adaptive prediction-error expansion and pixel selection. *IEEE Transactions on Image Processing*, *20*(12), 3524–3533.

Lian, S., Liu, Z., Ren, Z., & Wang, H. (2007). Commutative encryption and watermarking in video compression. *IEEE Transactions on Circuits and Systems for Video Technology*, *17*(6), 774 -778.

Lin, T.-C., & Lin, C.-M. (2009). Wavelet-based copyright-protection scheme for digital images based on local features. *Information Sciences*, *179*(19), 3349 - 3358.

Luo, H., Yu, F.-X., Chen, H., Huang, Z.-L., Li, H., & Wang, P.-H. (2011). Reversible data hiding based on block median preservation. *Information Sciences*, *181*(2), 308 - 328.

Luo, L., Chen, Z., Chen, M., Zeng, X., & Xiong, Z. (2010). Reversible image watermarking using interpolation technique. *IEEE Transactions on Information Forensics and Security*, *5*(1), 187–193.

Lusson, F., Bailey, K., Leeney, M., & Curran, K. (2013). A novel approach to digital watermarking, exploiting colour spaces. *Signal Processing*, *93*(5), 1268 - 1294.

Malik, H., Ansari, R., & Khokhar, A. (2007). Robust data hiding in audio using allpass filters. *IEEE Transactions on Audio, Speech, and Language Processing*, *15*(4), 1296-1304.

Marsaglia, G., & Tsang, W. W. (2000). The ziggurat method for generating random variables. *Journal of*

Statistical Software, *5*(8), 1–7.

McLoone, M., & McCanny, J. (2003). High-performance FPGA implementation of DES using a novel method for implementing the key schedule. *IEE Proceedings - Circuits, Devices and Systems*, *150*(5), 373-8.

Merkle, R. C., & Hellman, M. E. (1981). On the security of multiple encryption. *Commun. ACM*, *24*(7), 465–467.

Mobasseri, B. G., Berger, R. J., Marcinak, M. P., & NaikRaikar, Y. J. (2010). Data embedding in JPEG bitstream by code mapping. *IEEE Transactions on Image Processing*, *19*(4), 958–966.

Mohamed, E. M., El-Etriby, S., & Abdul-kader, H. S. (2012). Randomness testing of modern encryption techniques in cloud environment. In *8th international conference on informatics and systems (infos)* (p. CC-1 -CC-6).

Moon, D., Hwang, K., Lee, W., Lee, S., & Lim, J. (2002). Impossible differential cryptanalysis of reduced round XTEA and TEA. In *Revised papers from the 9th international workshop on fast software encryption* (pp. 49–60). London, UK, UK: Springer-Verlag.

Mousa, A. (2005). Data encryption performance based on blowfish. In *47th international symposium elmar* (p. 131-134).

Najafi, B., Sadeghian, B., Saheb Zamani, M., & Valizadeh, A. (2004). High speed implementation of serpent algorithm. In *The 16th international conference on microelectronics* (p. 718 - 721).

Ni, Z., Shi, Y.-Q., Ansari, N., & Su, W. (2006). Reversible data hiding. *IEEE Transactions on Circuits and Systems for Video Technology*, *16*(3), 354–362.

(NISO-U.S.), N. I. S. O. (2004). *Understanding metadata*. NISO Press.

Ogihara, T., Nakamura, D., & Yokoya, N. (1996). Data embedding into pictorial images with less distortion using discrete cosine transform. In *Proceedings of the 13th international conference on pattern recognition* (Vol. 2, pp. 675–679).

Ong, S., Wong, K., & Tanaka, K. (2014). A scalable reversible data embedding method with progressive quality degradation functionality. *Signal Processing: Image Communication*, *29*(1), 135 - 149.

Peng, F., Li, X., & Yang, B. (2012). Adaptive reversible data hiding scheme based on integer transform. *Signal Processing*, *92*(1), 54–62.

Pereira, F., Vetro, A., & Sikora, T. (2008). Multimedia retrieval and delivery: Essential metadata challenges and standards. *Proceedings of the IEEE*, *96*(4), 721-744.

Petitcolas, F. A. P., Anderson, R., & Kuhn, M. (1999). Information hiding-a survey. *Proceedings of the IEEE*, *87*(7), 1062-1078.

Qian, Z., & Zhang, X. (2012). Lossless data hiding in JPEG bitstream. *Journal of Systems and Software*, *85*(2), 309–313.

Rabbani, M., & Jones, P. W. (1991). *Digital image compression techniques* (1st ed.). Bellingham, WA, USA: Society of Photo-Optical Instrumentation Engineers (SPIE).

Rice, R., & Plaunt, J. (1971). Adaptive variable-length coding for efficient compression of spacecraft television data. *IEEE Transactions on Communication Technology*, *19*(6), 889-897.

Rompay, B. V., Knudsen, L. R., & Rijmen, V. (1998). Differential cryptanalysis of the ICE encryption algorithm. In *Proceedings of the 5th international workshop on fast software encryption* (pp. 270–283). London, UK, UK: Springer-Verlag.

Sandhaus, P., & Boll, S. (2011). Semantic analysis and retrieval in personal and social photo collections.

Multimedia Tools Appl., *51*(1), 5–33.

Shahid, Z., & Puech, W. (2013). A histogram shifting based RDH scheme for H. 264/AVC with controllable drift. In *IS&T/SPIE electronic imaging* (pp. 86650S–86650S).

Shannon, C. E. (2001). A mathematical theory of communication. *SIGMOBILE Mob. Comput. Commun. Rev.*, *5*(1), 3–55.

Shih, F. Y., & Wu, S. Y. (2003). Combinational image watermarking in the spatial and frequency domains. *Pattern Recognition*, *36*(4), 969–975.

Smith, J., & Schirling, P. (2006). Metadata standards roundup. *IEEE MultiMedia*, *13*(2), 84-88.

Stankovic, S., Orovic, I., & Zaric, N. (2010). An application of multidimensional time-frequency analysis as a base for the unified watermarking approach. *IEEE Transactions on Image Processing*, *19*(3), 736–745.

Statistics. (2014). Youtube [Computer software manual]. (`http://www.youtube.com/yt/press/statistics.html`)

Stvilia, B., Jörgensen, C., & Wu, S. (2012). Establishing the value of socially-created metadata to image indexing. *Library and Information Science Research*, *34*(2), 99 - 109.

Su, S.-L., Wuu, L.-C., & Jhang, J.-W. (2007). A new 256-bits block cipher Twofish256. In *International conference on computer engineering systems* (p. 166 -171).

Sun, H.-M., Wu, M.-E., Ting, W.-C., & Hinek, M. (2007). Dual RSA and its security analysis. *IEEE Transactions on Information Theory*, *53*(8), 2922 -2933.

Tai, W.-L., Yeh, C.-M., & Chang, C.-C. (2009). Reversible data hiding based on histogram modification of pixel differences. *IEEE Transactions on Circuits and Systems for Video Technology*, *19*(6), 906–910.

Thodi, D. M., & Rodríguez, J. J. (2007). Expansion embedding techniques for reversible watermarking. *IEEE Transactions on Image Processing*, *16*(3), 721–730.

Tian, J. (2003). Reversible data embedding using a difference expansion. *IEEE Trans. Circuits Syst. Video Techn.*, *13*(8), 890–896.

Tsai, P., Hu, Y.-C., & Yeh, H.-L. (2009). Reversible image hiding scheme using predictive coding and histogram shifting. *Signal Processing*, *89*(6), 1129–1143.

Tseng, F. S. (2005). Design of a multi-dimensional query expression for document warehouses. *Information Sciences*, *174*(1–2), 55 - 79.

USC-SIPI. (2014). *USC-SIPI image database. on-line: http://sipi.usc.edu/database/.*

Vaseghi, S. V. (2006). *Advanced digital signal processing and noise reduction*. John Wiley & Sons.

Wang, X., Li, X., Yang, B., & Guo, Z. (2010). Efficient generalized integer transform for reversible watermarking. *IEEE Signal Processing Letters*, *17*(6), 567–570.

Wassermann, J. (2013). New robust video watermarking techniques based on DWT transform and spread spectrum of basis images of 2D hadamard transform. In *Multimedia communications, services and security* (pp. 298–308). Springer.

Weinberger, M., Seroussi, G., & Sapiro, G. (2000). The LOCO-I lossless image compression algorithm: principles and standardization into JPEG-LS. *IEEE Transactions on Image Processing*, *9*(8), 1309 -1324.

Wen, F., & Yin, L. (2010). Optimal tweakable blockcipher based on dual MISTY-type structure. In *Second international conference on future networks* (p. 39 -41).

Wong, K., Qi, X., & Tanaka, K. (2007). A DCT-based Mod4 steganographic method. *Signal Processing*, *87*(6), 1251 - 1263.

Wong, K., Tanaka, K., Takagi, K., & Nakajima, Y. (2009). Complete video quality-preserving data hiding. *IEEE Transactions on Circuits and Systems for Video Technology*, *19*(10), 1499-1512.

Wong, P. H., Au, O. C., & Wong, J. W. (2001). Data hiding technique in JPEG compressed domain. In *Photonics west 2001-electronic imaging* (pp. 309–320).

Wu, H.-T., & Huang, J. (2012). Reversible image watermarking on prediction errors by efficient histogram modification. *Signal Processing*, *92*(12), 3000–3009.

Wu, X., & Satoh, S. (2013). Ultrahigh-Speed TV commercial detection, extraction, and matching. *IEEE Transactions on Circuits and Systems for Video Technology*, *23*(6), 1054-1069.

Xu, D., & Wang, R. (2011). Watermarking in H. 264/AVC compressed domain using Exp-Golomb code words mapping. *Optical Engineering*, *50*(9), 097402–097402.

Xu, D., Wang, R., & Shi, Y. (2014). Data hiding in encrypted h.264/avc video streams by codeword substitution. *IEEE Transactions on Information Forensics and Security*, *9*(4), 596-606.

Xu, D., Wang, R., & Wang, J. (2011). A novel watermarking scheme for H.264/AVC video authentication. *Signal Processing: Image Communication*, *26*(6), 267 - 279.

Xuan, G., Shi, Y. Q., Chai, P., Cui, X., Ni, Z., & Tong, X. (2008). Optimum histogram pair based image lossless data embedding. In *Digital watermarking* (pp. 264–278). Springer.

Xuan, G., Shi, Y. Q., Ni, Z., Chai, P., Cui, X., & Tong, X. (2007). Reversible data hiding for JPEG images based on histogram pairs. In *Image analysis and recognition* (pp. 715–727). Springer.

Xuan, G., Zhu, J., Chen, J., Shi, Y. Q., Ni, Z., & Su, W. (2002). Distortionless data hiding based on integer wavelet transform. *Electronics Letters*, *38*(25), 1646–1648.

Yang, W.-J., Chung, K.-L., & Liao, H.-Y. M. (2012). Efficient reversible data hiding for color filter array images. *Information Sciences*, *190*(0), 208 - 226.

Yee, L., Wong, K., & Chee, K. O. (2012). Unispach: A text-based data hiding method using unicode space characters. *Journal of Systems and Software*, *85*(5), 1075–1082.

Yuan, H.-D. (2014). Secret sharing with multi-cover adaptive steganography. *Information Sciences*, *254*(0), 197 - 212.

Zhang, X. (2011). Reversible data hiding in encrypted image. *IEEE Signal Processing Letters*, *18*(4), 255 -258.

Zhang, X. (2012). Separable reversible data hiding in encrypted image. *IEEE Transactions on Information Forensics and Security*, *7*(2), 826 -832.

Zhao, B., Kou, W., Li, H., Dang, L., & Zhang, J. (2010). Effective watermarking scheme in the encrypted domain for buyer–seller watermarking protocol. *Information Sciences*, *180*(23), 4672 - 4684.

Zhou, J., & Au, O. C. (2012). Determining the capacity parameters in PEE-based reversible image watermarking. *IEEE Signal Processing Letters*, *19*(5), 287–290.