



A New Approach to Compression of
Text Data in Tabular Form
Using One-To-Many Dictionary-Based
Preprocessing Algorithm

By
Wuthikrai Tharachatr

Submitted in Partial Fulfillment of the
Requirements for the Degree of
Doctor of Philosophy
in Information Technology
Assumption University

November, 2012

A New Approach to Compression of Text Data in Tabular Form Using One-To-Many Dictionary-Based Preprocessing Algorithm

by

Wuthikrai Tharachatr



**Submitted in Partial Fulfillment of the
Requirements for the Degree of
Doctor of Philosophy
in Information Technology
Assumption University**

November 2012

The Faculty of Science and Technology

Dissertation Approval

Dissertation Title A New Approach to compression and confidentiality of Text Data
in Tabular Form Using One-T-Many Dictionary-Based
Preprocessing Algorithm: A Case Study in Cloud Supply Chain


By Mr. Wuthikrai Tharachatr

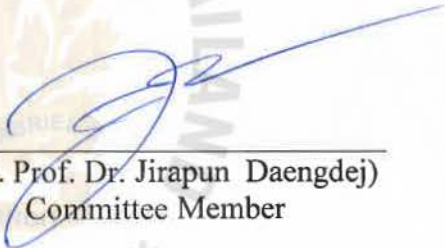
Dissertation Advisor Professor Dr. Graham Winley


Academic Year 2/2012


The Department of Information Technology, Faculty of Science and Technology of Assumption University has approved dissertation final report of the **thirty six** credits course. **IT9000 Dissertation**, submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Information Technology.

Approval Committee:



(Asst. Prof. Dr Thotsapon Sortrakul)
Advisor

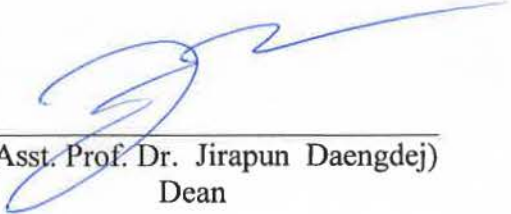

(Asst. Prof. Dr. Jirapun Daengdej)
Committee Member


(Professor. Dr. Graham Winley)
Committee Member


(Assoc. Prof. Dr. Surapong Auwatanamongkol)
Commission of Higher Education
University Affairs

Faculty Approval:


(Professor. Dr. Graham Winley)
Program Director


(Asst. Prof. Dr. Jirapun Daengdej)
Dean

November / 2012

The Faculty of Science and Technology

Declaration

This is to certify that the work presented in this thesis was carried out by the author in the Department of Information Technology at Assumption University, Thailand and is the result of original research conducted by the author, except where formally acknowledged and/or referenced, and has not been submitted for a degree to any other university or institution.

(Wuthikrai Tharachatr)



ACKNOWLEDGEMENT

I express my appreciation and sincere thanks to those who have contributed to the completion of this dissertation. First and foremost, I extend my deep gratitude to my advisor, Asst. Prof. Dr. Thotsapon Sortrakul, for his best valuable time, effort, assistance, suggestions, and encouragements, which were valuable in all steps of writing this dissertation.

I also acknowledge my committee members, Prof. Dr. Graham Kenneth Winley, Asst. Prof. Dr. Jirapun Daengdej, and Assoc. Prof. Dr. Surapong Auwatanamongkol who provided much valuable advices and recommendations during my dissertation defense.

Special acknowledgement goes to Asst. Prof. Dr. Supavadee Nontakao for her inspiration and guidance during my study.

Last but not least, to my beloved parents, I owe a debt of gratitude for their supports and encouragements along the path of my academic pursuits. I am also thankful to God for all the wonderful things He has given me.

ABSTRACT

Due to dramatic growth of data storage and transfer for most enterprises nowadays, it is necessary to have larger memories to store numbers of binary generated from everyday communication and documentary. Especially, for large or multinational enterprises where various parts of digital communication and documentary are duplicated and redundant, many data compression techniques have been used to reduce the storage requirements by compressing the data binaries.

In this research, a simple preprocessing technique for improving compression performance of text data in tabular form is introduced. Data sharing through cloud computing in the field of supply chain is selected as the illustration. Sets of duplicated data are preprocessed based on analyses of static-data ratio and number of receivers in order to generate a single compressed file, while different decoding overheads are encoded and separately distributed to particular receivers in parallel.

The results show that compression performance can be improved with some trade-offs for decoding overheads. Static-data ratio and size of duplicated static data are found to be directly proportional to the compression performance with some benefits to data confidentiality, while number of receivers is related to the efficient use of encoding tokens.

CONTENTS

	Page
ACKNOWLEDGEMENT	I
ABSTRACT	II
CONTENTS	III
LIST OF FIGURES	V
LIST OF TABLES	VII
CHAPTER 1 INTRODUCTION	
1.1 Overview	1
1.2 Literature Survey	4
1.3 Research Problems	9
1.4 Objectives of the Study	10
1.5 Scope of the Study	11
1.6 Definitions and Equations	12
1.7 Dissertation Organization	13
CHAPTER 2 BACKGROUND KNOWLEDGE	
2.1 Lossless Data Compression Algorithms	14
2.2 Data Preprocessing Techniques	18
2.3 Archiving Software	22
2.4 Cloud Supply Chain	23
2.5 Data Confidentiality	29
CHAPTER 3 PROPOSED TECHNIQUE	
3.1 Overview	37

	Page
3.2 Single Encoding Token to Multiple Decoding Overheads	38
3.3 Data Isolation	39
3.4 Integrated Solution	40
CHAPTER 4 DATA PREPARATION AND TESTING METHOD	
4.1 Data Preparation	42
4.2 Simulation Method	43
4.3 Selected Archiving Software	44
4.4 Selected Existing Preprocessing Technique	45
CHAPTER 5 EXPERIMENTAL RESULTS AND ANALYSES	
5.1 Results and Comparisons of the Compression Performance in Relation to Ratios of Static Data and Duplicated Static Data	46
5.2 Results and Comparisons of the Efficient Use of Encoding Tokens in Relation to Number of Receivers	68
CHAPTER 6 CONCLUSION AND FUTURE WORKS	
6.1 Interpretation of Findings and Conclusion	70
6.2 Discussion: The Most Suitable Approach	71
6.3 Future Research	74
BIBLIOGRAPHY	75

LIST OF FIGURES

	Page
Figure 1.1: Cloud security survey	3
Figure 1.2: Top-ten technology priority	3
Figure 2.1: Cloud Supply Chain	29
Figure 2.2: Three core parts of information security	34
Figure 2.3: Degrees of security	35
Figure 2.4: Risk-Openness-Value Paradigm	36
Figure 3.1: The proposed technique paradigm	38
Figure 3.2: Encoding module	40
Figure 3.3: Decoding module	41
Figure 5.1: Comparison of compressed file sizes (25% of static data and 25% of duplicated static data)	48
Figure 5.2: Comparison of compressed file sizes (50% of static data and 25% of duplicated static data)	49
Figure 5.3: Comparison of compressed file sizes (75% of static data and 25% of duplicated static data)	51
Figure 5.4: Comparison of compressed file sizes (25% of static data and 50% of duplicated static data)	52
Figure 5.5: Comparison of compressed file sizes (50% of static data and 50% of duplicated static data)	54

	Page
Figure 5.6: Comparison of compressed file sizes (75% of static data and 50% of duplicated static data)	55
Figure 5.7: Comparison of compressed file sizes (25% of static data and 75% of duplicated static data)	57
Figure 5.8: Comparison of compressed file sizes (50% of static data and 75% of duplicated static data)	58
Figure 5.9: Comparison of compressed file sizes (75% of static data and 75% of duplicated static data)	60
Figure 5.10: Comparison of compressed file size based on the use of 7-Zip as the selected archiving software	61
Figure 5.11: Comparison of compressed file size based on the use of PeaZip as the selected archiving software	62
Figure 5.12: Comparison of compressed file size based on the use of WinAce as the selected archiving software	62
Figure 5.13: Comparison of compressed file size based on the use of WinRAR as the selected archiving software	63
Figure 5.14: Comparison of compressed file size based on the use of WinZip as the selected archiving software	64
Figure 5.15: Degrees of compression performance of the proposed preprocessing technique integrated with archiving software in average based on ratios of static data and duplicated static data	67
Figure 5.16: Number of tokens used in relation to number of receivers	68

LIST OF TABLES

	Page
Table 2.1: Selected Archiving Software	23
Table 2.2: Summary of Security Mechanisms by Major Cloud Service Providers	31
Table 4.1: Data sets used for testing compression performance in relation to the ratios of static data and duplicated static data	43
Table 4.2: Functions of Microsoft Excel 2003 that were used in this research	44
Table 4.3: Selected archiving software	45
Table 5.1: Details of data sets and compression algorithms used for the analyses	47
Table 5.2: Comparison of compression ratios (25% of static data and 25% of duplicated static data)	48
Table 5.3: Comparison of space saving percentage (25% of static data and 25% of duplicated static data)	48
Table 5.4: Comparison of compression ratios (50% of static data and 25% of duplicated static data)	50
Table 5.5: Comparison of space saving percentage (50% of static data and 25% of duplicated static data)	50
Table 5.6: Comparison of compression ratios (75% of static data and 25% of duplicated static data)	51
Table 5.7: Comparison of space saving percentage (75% of static data and 25% of duplicated static data)	52

	Page
Table 5.8: Comparison of compression ratios	
(25% of static data and 50% of duplicated static data)	53
Table 5.9: Comparison of space saving percentage	
(25% of static data and 50% of duplicated static data)	53
Table 5.10: Comparison of compression ratios	
(50% of static data and 50% of duplicated static data)	54
Table 5.11: Comparison of space saving percentage	
(50% of static data and 50% of duplicated static data)	54
Table 5.12: Comparison of compression ratios	
(75% of static data and 50% of duplicated static data)	56
Table 5.13: Comparison of space saving percentage	
(75% of static data and 50% of duplicated static data)	56
Table 5.14: Comparison of compression ratios	
(25% of static data and 75% of duplicated static data)	57
Table 5.15: Comparison of space saving percentage	
(25% of static data and 75% of duplicated static data)	57
Table 5.16: Comparison of compression ratios	
(50% of static data and 75% of duplicated static data)	59
Table 5.17: Comparison of space saving percentage	
(50% of static data and 75% of duplicated static data)	59
Table 5.18: Comparison of compression ratios	
(75% of static data and 75% of duplicated static data)	60

	Page
Table 5.19: Comparison of space saving percentage (75% of static data and 75% of duplicated static data)	60
Table 5.20: Compression performances of the archiving software when they were integrated with the proposed preprocessing technique	66
Table 6.1: Comparisons of WRT with the proposed preprocessing technique	71



CHAPTER 1

INTRODUCTION

1.1 Overview

Nowadays, everyday communication and documentation among the supply chain parties usually create redundancy of duplicated data, which cause unnecessary requirements for larger cloud data storage. From the past, data compression algorithms have existed for almost 40 years but there is no compression algorithm that is able to effectively compress all data structures and types. In addition, complex internal structure file which simultaneously stores different types of data is commonly used, therefore, it makes a single lossless data compression technique a difficult problem to compress data effectively and get a high compression ratio as a result. Currently, there are many data compression and preprocessing techniques designed to reduce size of data that are shared and transmitted in the cloud. For data compression, it can be categorized into two main techniques consisting of Lossless Data Compression and Lossy Data Compression.

Lossless Data Compression can decode the data exactly what it encoded. This compression technique allows the exact original data to be reconstructed from the compressed data. It is normally used for all kinds of text, scientific and statistical databases to ensure that the decoded output will be exactly identical to the original data. For Lossy Data Compression, the compressed data will be slightly distorted in which it is mostly used for digital sound or image compression where the distortion is acceptable. It aims to minimize the amount of data that needs to be kept or transmitted. Starting from an entropy coder, it is a method that assigns every symbol from the alphabet to a code depended on the occurrence probability of that particular

symbol. The symbols, which are occurred more frequently, will get shorter codes than the less probable symbols. The codes are assigned to the symbols by focusing on the expected length of the compressed sequence would be minimal.

Recently, cloud computing is one of the latest IT developments which provides various benefits and competitive advantages through flexibility, configurability, scalability, reliability, and anywhere accessibility. It provides many options ranged from everyday computer users to large and small businesses. However, there are still many challenges with the opportunities being offered by the cloud. Some of the challenges include confidentiality, security, and lack of control. Enterprises have to be aware of the security risks of storing important data and information on the cloud. International Data Corporation (IDC) conducted surveys during 2008 to 2012 in which some of the sections were related to the growth of cloud and security aspects in IT industry as illustrated in figure 1.1 and 1.2.

Figure 1.1 shows the survey on cloud security presenting that security was ranked as the first thing which IT executives concerned (Chang et al., 2010). Figure 1.2 displays top ten technology priorities conducted by IDC in 2010 in which the cloud computing was the first priority by organizations in the field of technology (Shuai et al., 2010).

In term of cloud supply chain; the uses of cloud service are mainly in the areas of purchasing/procurement, inventory management, order filling and processing, production scheduling, transportation/distribution, customer service, and vendor relations. It can involve many parties including raw material suppliers, packaging suppliers, manufacturers of finished goods, sourcing, logistics, outsourced third-party logistics, consolidators, carriers, freight forwarders, cargo agents, financial and information services, and consumers. In general, sensitive information processed

outside an enterprise brings some level of risk and they are typically managed within shared environments along with data of other partners, customers, and competitors.

Figure 1.1: Cloud security survey

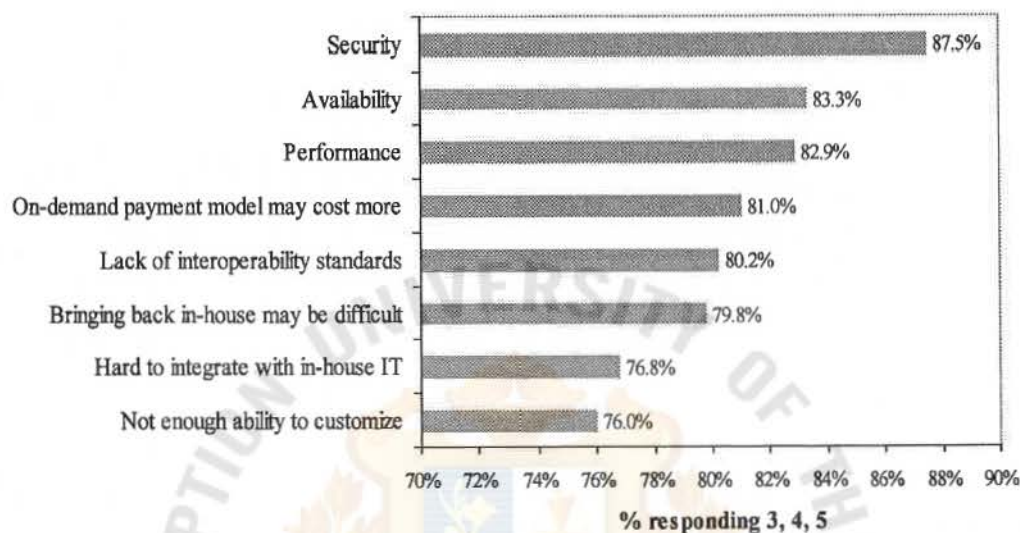
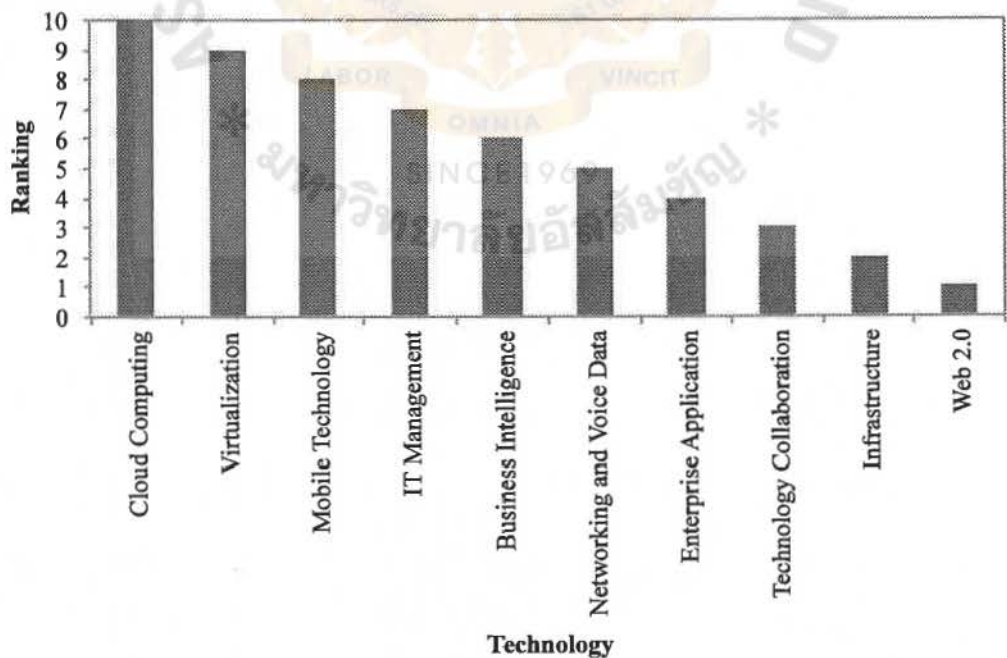


Figure 1.2: Top-ten technology priority



1.2 Literature Survey

The most well-known entropy coders are Huffman coder (Huffman, 1952) and an arithmetic coder. The Huffman coder is optimal in the way that it assigns codes of integer length, while the arithmetic coder is free from this limitation, therefore, arithmetic coding usually creates shorter expected code length. Sharma (2010) mentioned that Huffman coding is in wide use because of its simplicity, high speed and lack of encumbrance by patents while Arithmetic coding can be viewed as a generalization of Huffman coding; indeed, in practice arithmetic coding is often preceded by Huffman coding, as it is easier to find an arithmetic code for a binary input than for a non-binary input. Suri & Goel (2010) presented a memory efficient array data structure to decode the binary codeword based on Huffman techniques in which they developed two algorithms. In first algorithm they used Huffman ternary tree with height h with binary codeword, which resulted out the corresponding symbol of the given codeword in very short time and required less memory. In their second algorithm, they used array data structure to decode the binary codeword. In this case, both algorithms presented totally new formulas and require less effort.

In term of universal lossless data compression algorithms where they work well on the output data from general source class, currently they are popular and widely used. Historically, Ziv and Lempel introduced the first one in 1977 (Ziv & Lempel, 1977). The authors proposed the method to compress identical parts and replace the repetitions with the information where the identical subsequences appeared before. Ziv and Lempel (1978) proposed two main variants of their methods, which are LZ77 that encodes the information of repetitions directly, and LZ78 that maintains a supporting dictionary of subsequences previously appeared, and stores the indexes from this dictionary in the output sequence. The main

advantages of these two methods are high speed and ease of implementation. However, compression ratio of these two algorithms might be worse than the current ratios available nowadays. Mohd et al. (2009) developed the enhanced LZW technique (LZW++) in data compression in which the basic framework of LZW++ technique is based on the existing LZW technique. The LZW++ technique read three characters in time during data compression while existing LZW technique read character one by one. Comparison was made between LZW++ technique and existing LZW technique in terms of time performance and size of file after compression. The authors presented that LZW++ technique is more efficient in text compression than the existing LZW techniques.

In 1984, a prediction by partial matching (PPM) algorithm was introduced by Cleary and Witten (1984). This algorithm works differently compared to the previous algorithms proposed by Ziv and Lempel in which it statistically calculates occurrence of symbols previously appeared. After that it uses them to assign codes to the symbols from the alphabet that can occur at the next position in which the expected length of the sequence is minimized. In this case, the symbols that are more likely to occur will have shorter codes compared to less probably symbol. The statistics of symbol occurrences would be stored for separate contexts, therefore, after processing one symbol, the codes which are assigned for symbols usually completely differ because the changing of the context. Anyway, an arithmetic coder is used in order to assign codes for symbols. The disadvantages of the PPM algorithms are slow speed and large memory is required in order to store the statistics of symbol occurrences, which it might not be practical for some cases. However, these methods obtain the best compression ratios in the group of universal lossless data compression algorithms.

Later, in 1987, Cormack and Horspool introduced another statistical lossless data compression algorithm called as dynamic Markov coder (DMC). Their algorithm assumes that the data which being compressed is an output of some Markov source class. It tries to discover this source during the compression by better and better estimating the probability of occurrence of the next symbol by using the probability in which the codes for symbols from the alphabet are assigned by using an arithmetic coder. This algorithm has the same disadvantages as PPM because a large storage memory is required to store the statistics of symbol occurrences and also slow running. Anyway, it could be an alternative for the PPM methods because it generates the comparable compression ratios by using the similar running speed.

In 1994, another compression method called a Burrows–Wheeler compression algorithm (BWCA) is introduced. It is a block-sorting compression algorithm in which the technique is to build a matrix where rows store all the one-character cyclic shifts of the compressed sequence, to lexicographically sort the rows, and to use the last column of the matrix for further processing. This process is named as the Burrows–Wheeler transform (BWT). The output of the transform is then handled by a move-to-front transform (Bentley et al., 1986), and would be compressed by an entropy coder in the last stage, which either a Huffman coder or an arithmetic coder is used. As the result of this algorithm, a sequence is obtained in which all symbols appeared in the similar contexts would be grouped together. The advantages of the BWT algorithm are high speed of execution and reasonable compression ratios which are better than the LZ methods. Anyway, the result is slightly worse than the best existing PPM algorithms (Deorowicz, 2003) in which the PPM algorithms generates the higher compression ratios but execute a bit slower than the BWT method.

Willems et al. (1995) developed another compression method named context tree weighting (CTW) algorithm. This algorithm assumes that the data are produced by some source of that class and then the probability of symbol occurrence is estimated. This algorithm is similar to the PPM and the DMC algorithms because an arithmetic coder is applied to assign codes to the symbols during the compression. The main advantage of this algorithm is its higher compression ratio compared to the DMC technique but slightly worse than those obtained by the PPM algorithms (Deorowicz, 2003). By the way, the main disadvantage of this algorithm is its low running speed.

In 2004, Govindan and Shajeemohan presented an intelligent text data encryption and compression for high speed and secure data transmission over Internet called Intelligent Dictionary Based Encoding (IDBE) in which a preprocessing of the text prior to conventional compression will improve the compression efficiency and provide the required security.

In 2007, Robert and Nadarajan also proposed a method, which transforms a text file into intermediate file with minimum possible byte values called Dictionary-based transformation (DBT) and Dynamic reversible transformation (DRT) in order to reduce the number of possible bytes that appear after every byte in the source file which increases backend algorithm's compression performance.

Kattan (2010) purposed the applications of Genetic Programming (GP) in the lossless data compression domain. In particular, the author proposed a series of intelligent universal compression systems called the GP-zip family. The author presented four members of this family, namely, GP-zip, GP-zip*, GP-zip2 and GP-zip3. Each new version addressed the limitations of previous systems and improves upon them. Moreover, a new learning technique is introduced which it is specialized

on analyzing continued stream of data, detecting different patterns within them and associating these patterns with different classes according to the user's need.

Carus & Mesut (2010) developed a fast text compression method based on multiple static dictionaries and named this algorithm as STECA (Static Text Compression Algorithm). This algorithm is language dependent because of its static structure. To evaluate encoding and decoding performance of STECA with different languages, the author selected English and Turkish that have different grammatical structures then they compared the compression and decompression times and compression ratio results with the results of LZW, LZRW1, LZP1, LZOP, WRT, DEFLATE (Gzip), BWCA (Bzip2) and PPMd algorithms. In their research, the result presented that if speed is the primary consideration, STECA is an efficient algorithm for compressing natural language texts.

In addition to the lossless text compression, there are also various preprocessing techniques available nowadays which bring better compression performance when they are properly used together with lossless data compression algorithms. Preprocessing techniques bring some compression at the initial preprocessing stage itself as well as retain enough context and redundancy for existing compression algorithms to bring out better results (Rexline and Robert, 2011). The famous preprocessing techniques are Burrows Wheeler Transformation (BWT), Star Encoding, Length Index Preserving Transformation (LIPT), StarNT, and Word Replacement Transformation (WRT). According to these preprocessing methods, the dictionary is made in time as static one and distributed by both encoding units and decoding units. The usefulness of dictionary-based text preprocessing skills are the less memory utilization, simple concepts, higher speed in process, and better results in the compression rates (Rexline and Robert, 2011).

Ullah et al. (2010) presented a novel use of steganography for both confidentiality and compression in which message is encoded by using a grayscale bitmap image. The image acts as a steganographic carrier for the text and the carrier is never transmitted across the untrusted channel while only the compressed index array that contains the indices for hidden data in the image is transmitted. The image also acts as a shared key between sender and receiver, which is used for confidentiality and also to extract the desired text from the image. They described that encoding text into the image is not only to make it secure but good amount of compression is also achieved and found that the text length is directly proportional to compression performance.

1.3 Research Problems

From the past, data compression algorithms have existed for almost 40 years but there is no compression algorithm that is able to effectively compress all data structures and types. Nowadays, complex internal structure file which simultaneously stores different types of data is commonly used, therefore, it makes a single lossless data compression technique a difficult problem to compress data effectively and get a high compression ratio as a result. In addition, there are still many challenges including confidentiality, security, and lack of control for data sharing on the cloud, therefore, some of enterprises have to be aware of the security risks of storing important data and information on the cloud. In order to improve data compression with some benefits to data confidentiality, the researcher describes current problems and issues as follows.

- 1) Each particular algorithm has its own strengths and weaknesses based on data type that it was designed to work best for. Therefore, only one data

compression technique cannot be used to get the minimum compressed complex internal structure data. Therefore, data preprocessing technique might be another solution to improve data compression performance.

- 2) Large or multinational enterprises, especially in supply chain related field, have hugely amount of complex internal structure data, but most of the data are mainly and frequently duplicated in particular formats.
- 3) Since data confidentiality is still one of the most challenging problems in cloud supply chain, therefore, instead of using controls and encryptions, data isolation is applied as a solution to support enterprises about confidentiality of their valuable data and information.

To overcome these issues, the researcher proposes a simple approach that can improve compression performance with some benefits to data confidentiality of shared text data structured in tabular form by testing with real supply chain data that is shared on cloud computing.

1.4 Objectives of the Study

This section describes objectives of this dissertation in which it focuses on creating of a solution for improving data compression performance of text data structured in tabular form by integrating the proposed preprocessing technique with existing archiving software. Since each particular existing data compression algorithm has its own strengths and weaknesses based on which data type it was designed to work best for, therefore, only a single technique might not be employed in order to get the minimum compressed complex internal structure data together with data confidentiality. In this study, an efficient text preprocessing algorithm for data

compression that can encode specific multiple data into a single encoding token is introduced. The followings are the objectives of this research:

- 1) Introduces a simple preprocessing technique that can enhance compression performance of text data that is structured in tabular form when it is integrated with archiving software.
- 2) Increases utilization of encoding tokens, which are generated for a particular set of data that is structured in tabular form by relating it with number of assigned receivers.

1.5 Scope of the Study

The scope of this study is to introduce a simple text preprocessing approach that can improve compression performance of shared text data that is structured in tabular form for supporting cloud supply chain. An efficient text-preprocessing algorithm that can encode specific multiple data into a single encoding token is proposed. However, the proposed technique cannot be applied for other text formats that are not structured in spreadsheet form.

1.6 Definitions and Equations

Ratio of static data: Percentage of static data available in the original file.

$$= \frac{\text{Size of static data}}{(\text{Size of static data} + \text{Size of dynamic data})}$$

Ratio of duplicated static data: Percentage of static data that is duplicated in the original file.

$$= \frac{\text{Size of duplicated static data}}{\text{Size of static data}}$$

Total file size: = Size of static data + Size of dynamic data

Compression ratio: The ratio between the size of the compressed file and the size of the source file.

$$= \frac{\text{Size after compression}}{\text{Size before compression}}$$

Space saving percentage: Percentage of shrinkage of the source file.

$$= \frac{\text{Size before compression} - \text{Size after compression}}{\text{Size before compression}} \%$$

1.7 Dissertation Organization

This dissertation is divided into 6 chapters, which are discussed one by one as follow.

The backgrounds of lossless data compression, data preprocessing, well-known archiving software, cloud computing, and data confidentiality are elaborated in Chapter 2.

Chapter 3 introduces a simple approach to compression of text data in tabular form by using one-to-many dictionary-based preprocessing algorithm for supporting data sharing in cloud supply chain including encoding and decoding modules.

Chapter 4 describes about data preparation for testing the proposed text-preprocessing algorithm and discusses about the simulation in details.

Chapter 5 presents outcomes of integrating the proposed preprocessing algorithm with selected archiving software along with its experimental analyses with the use of selected archiving software individually as well as the use of existing well-known preprocessing technique called Word Replacement Transformation (WRT) as an integration tool.

Chapter 6 interprets the research findings, discussion about the most suitable approach for applying the proposed preprocessing technique, and some recommendations for future researches.

CHAPTER 2

BACKGROUND KNOWLEDGE

Prior to the proposed technique, this section provides background knowledge in relation to lossless data compression algorithms, data preprocessing techniques, some of well-known archiving software, cloud supply chain, and data confidentiality.

2.1 Lossless Data Compression Algorithms

For lossless data compression techniques, they can be categorized into five popular techniques which are 1) Shannon-Fano coding and Huffman coding, 2) Arithmetic coding, 3) Dictionary-based coding, 4) Prediction by Partial Matching (PPM), and 5) Burrows-Wheeler Transform (BWT).

- 1) **Shannon-Fano Coding and Huffman Coding:** Shannon-Fano coding, introduced by Claude Elwood Shannon (1948) and Robert Fano (1949), is a data compression technique that constructs a prefix code based on a set of symbols and their estimated or measured probabilities. The symbols are arranged in order from most probable to least probable, after that they will be divided into two sets in which total probabilities will be as close as possible to being equal. Then, all symbols have the first digits of their codes assigned; symbols in the first set will be assigned as "0" and symbols in the second set will be assigned as "1". If there is any set with more than one member remains, the same process will be repeated for those sets in order to determine successive digits of their codes. When a set has been reduced to one symbol, it means that the symbol's code is completed and will not assign the prefix of any other symbol's code.

Huffman coding was introduced in 1952 by David A. Huffman, which represents a different technique compared to Shannon-Fano algorithm. While the Shannon-Fano tree is created from the root to the leaves, the Huffman algorithm does in the opposite direction by starting from leaves to the root. Huffman coding can be divided into two types, which are static Huffman coding and adaptive Huffman coding. For adaptive Huffman coding (Dynamic Huffman coding), it is an adaptive coding technique based on Huffman coding in which it dynamically calculates the probabilities based on recent actual frequencies in the sequence of source symbols, and then changes the coding tree structure in order to match with the updated probability estimates.

- 2) Arithmetic Coding: Arithmetic coding takes a stream of input symbols and then replaces them with a single floating point output number in which the longer the message will require more bits for output number. Arithmetic coding works on infinite-precision numbers in which this single number can be only in the range of zero to one. The output numbers are generated by probabilities assigned to the symbols that are being encoded. It means that the arithmetic coding differs from other forms of entropy encoding especially Huffman coding because it does not separating the input into component symbols and replacing each with a code but the arithmetic coding encodes the entire message into a single number. By comparing compression ability between Huffman coding and Arithmetic coding, the Arithmetic coding performs better than Huffman in many parts because Arithmetic is an adaptive model and does not need to translate each symbol into an integral number of bits. However, Arithmetic coding

requires more computation on multiplication and division, which causes slower coding time and more complication.

- 3) Dictionary-based Coding: Liao et al. (1995) and Lefurgy et al. (1997) explored this algorithm by taking advantage of commonly occurring instruction sequences with the use of a dictionary in which the repeating occurrence will be replaced by a codeword that points to the index of the dictionary that contains a particular pattern. Although the dictionary-based technique represents the good compression ratio but, practically, it might not be effective because each data file might contain different characteristics. Therefore, list of words in the dictionary to be used for compression has to be different belongs to the different characteristics of data. In general, sender can create a list that suitable for each data file to be compressed but receiver does not have that particular list by itself, therefore, sender has to send the dictionary-based list to receiver as overhead. When the overhead is taken into consideration, it reduces compression ratio and effectiveness of this algorithm. The main issue of the dictionary-based compression technique is about the list of words to be used between sender and receiver because the list will have a high significant impact to its effectiveness. If the list is not properly set, it will absolutely affect to the compression ratio. In order to solve this issue, there are two main methods called as “static dictionary” and “dynamic dictionary”. For “static dictionary”, the complete set of strings is determined prior to the starting of coding and it will not be changed during the coding process. It is frequently used when set of messages being encoded is fixed and large. The list will be set up to cover all the types of

files to be coded which will be updated to both sender and receiver sides in advance, therefore, there will be no overhead sending from sender to receiver for decoding purpose later on. Anyway, this static dictionary list has to be large enough to cover as many as possible of the types of data to be compressed. A static dictionary that is created for particular types of data might not be effectively applied to other types of data because the compressed data might be larger than the original data instead as the dictionary might not perfectly matched with those kinds of data. On the other hands, there are numbers of methods where the dictionary-based algorithms start by using some predetermined state while the contents can be changed during the encoding process based on the data that has already been encoded, which are called as “dynamic dictionary”, for examples, LZ77 and LZ78 algorithms.

- 4) Prediction by Partial Matching (PPM): The prediction by partial matching (PPM) data compression technique was developed by Cleary and Witten (1984). It is an adaptive statistical data compression technique based on context modeling and prediction. PPM algorithm uses a set of previous symbols in the uncompressed symbol stream to predict probability of occurrence of the next symbol in the stream. This probability is then used for encoding the symbol by the use of arithmetic coding algorithm. The idea of this algorithm is that if a large context is used to determine the probability of the symbol being encoded, it will require estimation and storage of an extremely large number of conditional probabilities in which it is unfeasible. Instead of estimating these probabilities in advance, the burden can be reduced by estimating the probabilities as the coding

proceeds. In this case, it is only required to store these contexts which have been occurred in the sequence being encoded, therefore, it is much smaller number compared to number of all possible contexts.

- 5) Burrows–Wheeler Transform (BWT): In 1994, Michael Burrows and David Wheeler introduced a data compression algorithm based on the Burrows–Wheeler Transform (BWT). The Burrows–Wheeler transform is also called as block-sorting compression such as bzip2. The compression ratios were comparable with the ones obtained using known best methods. When a character string is transformed by the BWT, the value of the characters will not be changed. The transformation arranges the order of the characters. If the original string had several substrings that frequently occurred, then the transformed string will have several places where a single character is repeated multiple times in a row. This is useful for compression, since it is easy to compress a string that has runs of repeated characters by using move-to-front transform and run-length encoding.

2.2 Data Preprocessing Techniques

The usefulness of dictionary-based text preprocessing techniques are the less memory utilization, simple concepts, higher speed in process and better compression performance because they bring some compression at the initial preprocessing stage and retain enough context and redundancy for existing compression algorithms to bring out better results of compression performance (Rexline and Robert, 2011).

There are about eight standard and famous preprocessing algorithms currently available presented as follows:

- 1) Burrows-Wheeler Transformation (BWT): Burrow and Wheeler (1994) presented a block-sorting lossless data compression algorithm which operates by reversibly permutes a block of source data and then rearranges it by using a sorting technique, after that pipes through a Move-To-Front (MTF) stage follows by using of Run Length Encoder and an entropy encoder (Huffman coding or Arithmetic coding). The output generated by the BWT will be lexicographically arranged as series of blocks. If the file size is larger, the BWT technique will separate the data into numbers of independent blocks of a predetermined size prior to data compression. Because each block is independent, therefore, it should be possible to run the BWT algorithm simultaneously on multiple blocks of data in order to achieve higher speed. After that the separated blocks are concatenated back together in order to form a final compressed file. In order to achieve good compression performance, a block size of sufficient value must be selected; at least 2 kilobytes because increasing of the block size will also increase the effectiveness of the algorithm (Rexline and Robert, 2011).
- 2) Incremental Frequency Count (IFC) – A post BWT-stage for the Burrows-Wheeler Transformation: Abel et al. (2007) introduced the Incremental Frequency Count stage by combining it with a Run Length Encoding (RLE) stage along with the BWT and entropy coding stage instead of using Move-To-Front (MTF) algorithm. IFC algorithm replaces the MTF algorithm by assigning a counter for each character and places in descending order in which whenever a character comes for process from the input stream, the position of the corresponding counter is yield and incremented which will be recalculated and added into the counter of the

processed character, therefore, only one counter needs to be rearranged inside the list in which it will make the faster process compared to MTF. In this case, the counters will be frequently rescaled for the purpose of preventing overruns and in order to indicate closer symbols, and it will give high throughput in the same way as MTF stage along with the comparable good compression performance (Rexline and Robert, 2011).

- 3) The Star Transformation: this algorithm was introduced by Kruse and Mukherjee in which the transformation is designed to make it easier to compress the source file by set up a very big dictionary in advance that contains commonly used words which are expected in the input files and the dictionary must be known by the sender and receiver. Every word in the dictionary as a star encoded equivalent in which as many letters as possible are replaced by the “*” character and the dictionary is sectioned into numbers of streams of sub-dictionaries that consist of words with length $1 < n < 22$ because the more length of an English word is 22 letters. Practically, the most used words will gain the highest percentage of “*” characters in their encoding and if it is carried out properly in which the transformed file has a big number of “*” characters, it will make the transformed file more compressible than the original text file (Rexline and Robert, 2011).
- 4) Length Index Preserving Transformation (LIPT): This algorithm was introduced by Awan and Mukherjee in 2001 by developing a dictionary-based reversible lossless text transformation that can be applied to a source text in order to improve compression performance. In this case, the length of the input word and the offset of the words in the dictionary are denoted

with alphabets and the encoding process will make use of recurrence of same length of words in the English language to create context in the transformed text that the entropy coders can exploit (Awan and Mukherjee, 2001).

- 5) StarNT Transformation: Sun et al. (2003) introduced this algorithm in which it works with a ternary search tree that provides a very fast change of encoding speed at a low capacity overhead and they introduced capital conversion technique by placing the escape symbol and flag director at the end of the codewords. The concept comes up to the background dependencies and results better compression ratio over the skills (Rexline and Robert, 2011).
- 6) Intelligent Dictionary Based Encoding (IDBE) Transformation: The algorithm was introduced by Govindan and Mohan in 2006 in which it is designed to preprocess the text and transform it into some intermediate form that can be compressed with better compression result by exploiting the natural redundancy of the language in making the transformation (Govindan and Mohan, 2006). The dictionary is produced with multiple sources of files as input and codewords are formed by the use of ASCII characters (Rexline and Robert, 2011).
- 7) Two Level Dictionary Based Text Transformation: Zia et al. (2008) introduced this preprocessing technique in which the original words in a text file are transformed into codewords having length 2 and 3 by using a dictionary comprising 73,680 frequently used words in English language in which most frequently used words use 2 length codewords while the reset words use 3 length codewords in order to generate good compression

result. The codewords are selected by considering the spaces between words in the original text file that can be removed altogether recovering a substantial amount of space, and another feature of this scheme is the recovery of unused bit of ASCII character representation from each character to save one byte per 8 ASCII characters (Zia et al., 2008).

- 8) Word Replacement Transformation (WRT): Skibinski introduced this algorithm in 2004. Grabowski et al. (2005) applied the algorithm by using only ASCII character 128-225 in order to represent the codewords and reduce the effect caused by the End-Of-Line (EOL) symbols, which hamper the context because words are usually separated by spaces (Rexline and Robert, 2011). This technique substitutes EOL symbols by spaces and encodes their former positions. Words will be replaced with references to where the word is located in the dictionary. In addition, the technique also suggests ngram replacement, space stuffing between words, capital conversion by placing the flag at the beginning of the codeword instead of placing it at the end of the codeword like StarNT (Rexline and Robert, 2011).

2.3 Archiving Software

Currently, there are various archiving tools for compressing data which are available either free or proprietary software. Each of archiving software is generally designed based on specific algorithms and its output performance might be varied when it is compared with other archiving tools because it depends on type and size of the data that it is well-designed to work for. Table 2.2 presents five well known

archiving programs selected based on algorithms they implement, supported operating system, and supported format requirements.

Table 2.1: Selected Archiving Software

Software	Algorithms	Support Windows OS	ZIP Archive support (reading and writing)
7-Zip	Program used filters, LZ77, LZMA, PPM, BWT	Yes	Yes
PeaZip	LZ77, Huffman, LZMA, PPM	Yes	Yes
WinAce	Program used filters, LZ77, Huffman	Yes	Yes
WinRAR	Program used filters, LZ77, PPM, Huffman	Yes	Yes
WinZip	Shannon-Fano, PPM, Huffman, LZH, LZW	Yes	Yes

2.4 Cloud Supply Chain

Currently, Cloud Computing technology is one of the rapidly growing fields of Information Technology (IT) and has supported organizations to cope with dynamic changes in markets and financial situations (Hugos and Hulitzky, 2010). Cloud services bring flexibility, configurability, cost effectiveness, low implementation cost to IT and by extension, Supply Chain Management (SCM) (Durowoju et al., 2011). The key driver of Cloud Computing is the Internet in which, at an organizational level, Lancioni et al. (2003) found that the Internet has been applied in the areas of purchasing/procurement; inventory management; transportation; order processing; customer service; production scheduling; and vendor relations, with procurement, transportation, and customer service having the most Internet usage. Chan and Chan (2010) stated that exchange of information drives business efficiency and effectiveness. Zhou et al. (2012) described the application of

Cloud Computing in four areas that has taken place in relation to SCM perspective which are: (a) *Product trace systems*: which are based on establishing of technologies such as RFID (Radio-Frequency Identification) in which products can be tracked and traced by using of intelligent network systems; (b) *Visualized intelligent distribution network management systems*: by using of GPS (Global Positioning System) satellite navigation and positioning, vehicle distribution networks can be scheduled and managed in real-time; (c) *Automated logistics and distribution centers*: based on the uses of advanced technologies; for examples, sound, light, color, or mechanical or electrical characteristics for logistics operations to operate automatically under the intelligent control; and (d) *Supply chain public information platforms*: for examples, platform facilities coordination, collaboration, and integration of the parties across a supply chain.

The conceptual model of Cloud Supply Chain can be classified into three levels, which are Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). The components are related to a generic supply chain including raw material management; coordination among manufacturers, distributors, wholesalers, retailers, and end-customers through uses of SCM software solution for examples planning and forecasting; CRM (Customer Relationship Management), e-procurement, etc. (Zhou et al., 2012). The followings are three levels of the model.

- 1) Software as a Service (SaaS): Customers can use the applications on cloud computing instead of using the application on a computer or in a local data center, but they cannot control the operating system, hardware or network infrastructure on Cloud Computing (National Institute of Standards and Technology (NIST), 2012).

- 2) Platform as a Service (PaaS) is typically an application framework in which the customers can control the applications run in the hosting environment, or over the hosting environment, but again they cannot control the operating system, hardware or network infrastructure on which they are running (NIST, 2012).
- 3) Infrastructure as a Service (IaaS) provides a service that the customers can use all fundamental computing resources on the Cloud Computing, for examples, processing power, storage, networking components or middleware. This is different from SaaS and PaaS because the customers can also control the operating system, storage, deployed applications and possibly networking (NIST, 2012).

There are four types of Cloud Computing currently available, which are Private Cloud, Public Cloud, Community Cloud, and Hybrid Cloud.

- 1) Private Cloud is established for a specific groups or organizations in which it limits access and operates solely for that particular group only. The resources are not shared by other entities (Kok, 2010). The organization or a third party may manage it. This type of Cloud service provides minimum risk but may not support scalability and agility of Public Cloud service. Private cloud users are considered as trusted by the organization because they are either employees, or have contractual agreements with the organization (Kok, 2010).

- 2) Public Cloud is the service that allows an access by any subscriber with an Internet connection to the cloud space. It is available to the general public or a large industry group. Public cloud users are not considered to be trusted because they are not tied to the organization as employees and the user has no contracture agreement with the provider (Kok, 2010).
- 3) Community Cloud is shared among two or more organizations, which have similar cloud requirements and share their missions or interests. It is the same as Private Cloud but data may be stored with the data of competitors. Community users are also considered as trusted by the organizations that are part of the community (Kok, 2010).
- 4) Hybrid Cloud is a combination of at least two clouds with the mixture of private, public, or community that remains unique entities with standardized or proprietary technology. It leverages the capabilities of each cloud deployment model where each part of a hybrid cloud is connected to the other by a gateway, controlling the applications and data that flow from each part to the other (Kok, 2010). The users of hybrid clouds can be considered as trusted and untrusted where the untrusted users are prevented to access the resources of the private and community parts of the hybrid cloud (Kok, 2010).

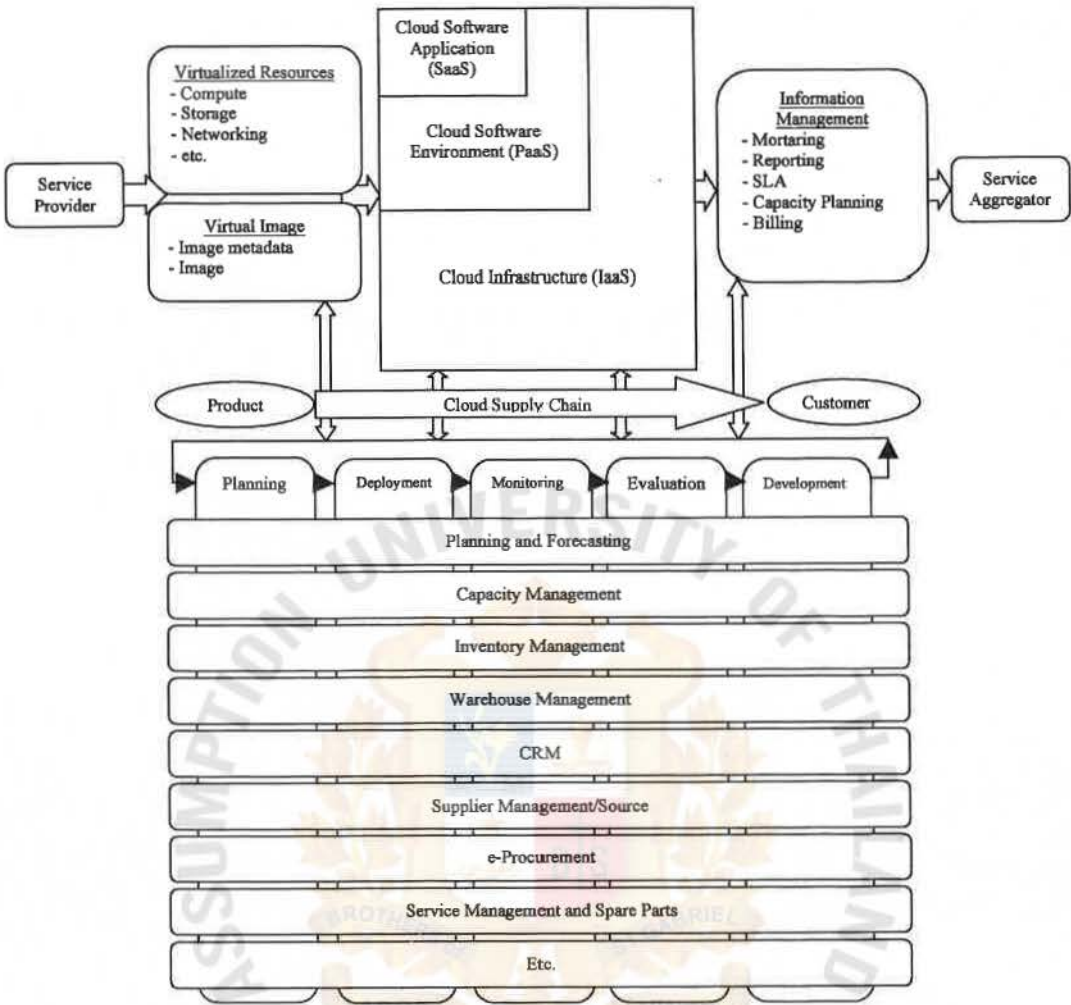
Although Cloud Computing is one of the latest IT developments that provides various benefits and competitive advantages through flexibility, capital investment saving, energy saving, increased efficiency, configurability, scalability, reliability, and anywhere accessibility. However, there are still many challenges with the opportunities being offered by the cloud as mentioned below:

- 1) Security, privacy, and trust: The cloud vendors typically provide an access control mechanism for their users (NIST, 2012). However, there are still some risks that Cloud Computing cannot provide some safety mechanisms, which can monitor and track their servers. For the private cloud, the user access and the networks used are restricted and designated, therefore, the data and processes are managed within the organization without restrictions of network bandwidth, security exposures and legal requirements that using Public Cloud services might entail (NIST, 2012). However, there are problems for the Public Cloud that it cannot always ensure about data confidentiality for information of all users, therefore, the users cannot avoid risks of receiving dangerous or malicious information along with service-hijacking, phishing, fraud, and exploitations. Sun et al. (2011) mentioned that some organizations can consider about providing of different security levels according to degrees of trust in Cloud Computing and they will manage trust degree change with interaction time and context together with monitor, adjust, and reflect the trust relationship dynamic with time.
- 2) Availability: The Cloud Computing user frequently wants constant access to their remote computing resources anytime and anywhere (Barnatt, 2010). However, the users may worry that when they need their data and applications, they may not be able to have access to the cloud at the time they need. In addition, although Cloud Computing is established basically on existing IT infrastructure, the newer infrastructure of Cloud Computing may meet some issues of hardware and software compatibility (Zhou et al., 2012).

- 3) Reliability: Individuals and organizations may concern if the data stored on a vendor's infrastructure will be safe enough. Although large Cloud Computing vendors who invest huge amount of money in establishing of firewalls and other securities, they are still cases that the Cloud Computing sites can be hacked (Zhou et al., 2012).
- 4) Controllability: Currently there are many vendors of Cloud Computing available where competitor suppliers may limit access to the Cloud Computing, or supply older hardware and software to customers, or provide limited capability for interoperability, therefore, it is difficult for users who would like to move their data or services from an existing Cloud Computing supplier to another vendor (Zhou et al., 2012). Sometimes, even the users would like to frequently upload or download the data; they may find some data transfer problems, for example, Internet flow limitation (Zhou et al., 2012).

Figure 2.1 presents the conceptual model of Cloud Computing in relation to supply chain (Zhou et al., 2012).

Figure 2.1: Cloud Supply Chain



2.5 Data Confidentiality

Confidentiality is one of the fundamental requirements for secure communication on an untrusted channel (Ullah et al., 2010). No matter how careful the users manage their personal data, by subscribing to the cloud, the users will be giving up some control to an external source because this distance between the users and the physical location of their data creates a barrier and it may also create more space for a third party to access the information (Huth and Cebula, 2011).

The followings are key security challenges of using Cloud Computing:

- 1) Data location: Generally, cloud users are not aware of the exact location of the datacenter and also they do not have any control over the physical access mechanisms to the particular data, but in some examples, applications and data may be stored in some countries where they have judiciary concerns and service providers will be subjected to the security requirements and legal obligations of that particular country (Sangroya et al., 2010).
- 2) Investigation: Data for multiple customers may be co-located and may also be spread across multiple datacenters; therefore, it is difficult for cloud service to be investigated and may be impossible. Users may not have much knowledge regarding the network topology and service provider may also impose restrictions on the network security of the users (Sangroya et al., 2010).
- 3) Data Segregation: Since the data in the cloud is stored in a shared environment together with the data from other users, therefore, encryption may not be assumed as the single solution for data segregation problems (Sangroya et al., 2010). In addition, some customers may not want to encrypt their data because there may be possible that the encryption may accidentally destroy their data (Sangroya et al., 2010).
- 4) Long-term viability: There is also a challenge for service providers to ensure the data safety in changing business situations; for examples, mergers and acquisitions; where the users must ensure that their data are availability without any issue during these situations (Sangroya et al., 2010).

- 5) **Compromised servers:** Since users do not have a choice of using physical acquisition toolkit in a cloud computing environment, in a situation where a server is compromised, they have to shut their servers down until they get a previous backup of the data (Sangroya et al., 2010).
- 6) **Recovery:** Cloud service providers must ensure that the users' data is securely stored. In general, the data is replicated across multiple sites but there may be some risks caused by unwanted events where the service provider must ensure about quick restoration of the data (Sangroya et al., 2010).

In table 2.2, Sangroya et al. (2010) presented results of a survey of security mechanisms available online at the official websites of major cloud service providers which are Amazon EC2, Amazon S3, GoGrid, Google App Engine, Microsoft Azure Services, Amazon Elastic Map Reduce, Salesforce, and Google Docs.

Table 2.2: Summary of Security Mechanisms by Major Cloud Service Providers

Security Issue *	Results *
Password Recovery	90% were using standard methods like other common services, while 10% were using sophisticated techniques.
Encryption Mechanism	40% were using standard SSL encryption, while 20% were using encryption mechanism but at an extra cost, 40% were using advance methods like HTTPS access also.
Data Location	70% had their datacenters located in more than one country, while 10% were located at a single location, and another 20% were not disclose about this information.
Availability of History	40% reported downtime along with results in data loss, while 60% of the cases were available in good condition.
Proprietary/Open	Only 10% of the service providers had open mechanism.
Monitoring Services	70% were providing extra monitoring services, while 10% were using automatic technique, and the remaining 20% were not open about this issue.

Currently, there are many methods employed to improve data security and confidentiality. Mainly, there are four kinds of safeguards; which are access controls, flow controls, inference controls, and cryptographic controls.

- 1) Access controls: These controls prevent accidental or malicious disclosure, modification, or destruction of records, data sets, and programs segments in which many access control systems incorporate a concept of ownership where a user may dispense and revoke privileges for objects he/she owns, for example, the patient does not own his record in a medical information system (Denning et al., 1979). The effectiveness of access controls are based on three assumption which are 1) *proper user identification* in which no one should be able to fool the system into giving him/her the capabilities of another, 2) *unanticipated observers do not gain access*, and 3) *privilege-information is heavily protected* in which all the information that specifies the access, each program has to objects in the system and the privilege-information is accessible only to authorized programs of the supervisor (Denning and Denning, 1979).
- 2) Flow controls: These controls regulate the dissemination or copying of information by prohibiting derived data from having lower confidentiality than the original in which they control the flow that occurs from object X to object Y, for example, a simple flow of copying a file from file X into file Y (Denning and Denning, 1979). Most flow controls employ some concept of security class; the transfer of information from a sender to a receiver is allowed only if the receiver's security class is at least as privileged as the sender's (Denning, 1976). Flow controls can prevent a service program from leaking a customer's confidential data, however,

such controls are often complex and hard to be efficiently used (Denning and Denning, 1979).

- 3) Inference controls: These controls prevent leakage through programs that produce summaries of groups of confidential records because the summaries may contain vestiges of the original information in which snooper can reconstruct the information by processing these enough summaries, which is called as deduction of confidential information by inference in order to make the cost of obtaining confidential information unacceptably high (Denning and Denning, 1979).
- 4) Cryptographic controls: Encryption is a common safeguard for data stored in, or in transit through, media whose security cannot be guaranteed by these other controls (Denning and Denning, 1979). In this case, with the help of a secret key, the sensitive plaintext is scrambled into unintelligible cipher text before being put in the insecure medium in which generally the practical controls are based on shorter keys than the original messages because the intruder may know the enciphering and deciphering algorithms and the security of these controls depend on the secrecy of the keys and the computational difficulty of inverting the enciphering algorithms (Denning and Denning, 1979). The Data Encryption Standard (DES) is the first encryption standard based on the IBM proposed algorithm called Lucifer. It is an efficient and economical algorithm, which became a standard in 1974. 3DES (Triple Data Encryption Standard) is an enhancement of DES in which the encryption method is similar to the one in original DES but it is applied three times to increase the encryption level but the encryption time is slower. AES (Advanced

Encryption Standard) is a newer encryption standard recommended by NIST to replace DES. Another algorithm called “blowfish” is one of the most common public domain encryption technique introduced by Bruce Schneier in 1993, which is a variable length key (64-bit block cipher).

In order to plan an information security, the security plan should have three essential parts as 1) protection of the information itself at the core, 2) hardening of the resources (systems and networks), and 3) authentication of those accessing the information as presented in figure 2.2 (Calabrese, 2004). Figure 2.4 presents the assessed level of risk decreases commensurate with the value of the information in which the relationship is said to be proportional that when the value increases, the risk will be increased (Calabrese, 2004).

Figure 2.2: Three core parts of information security

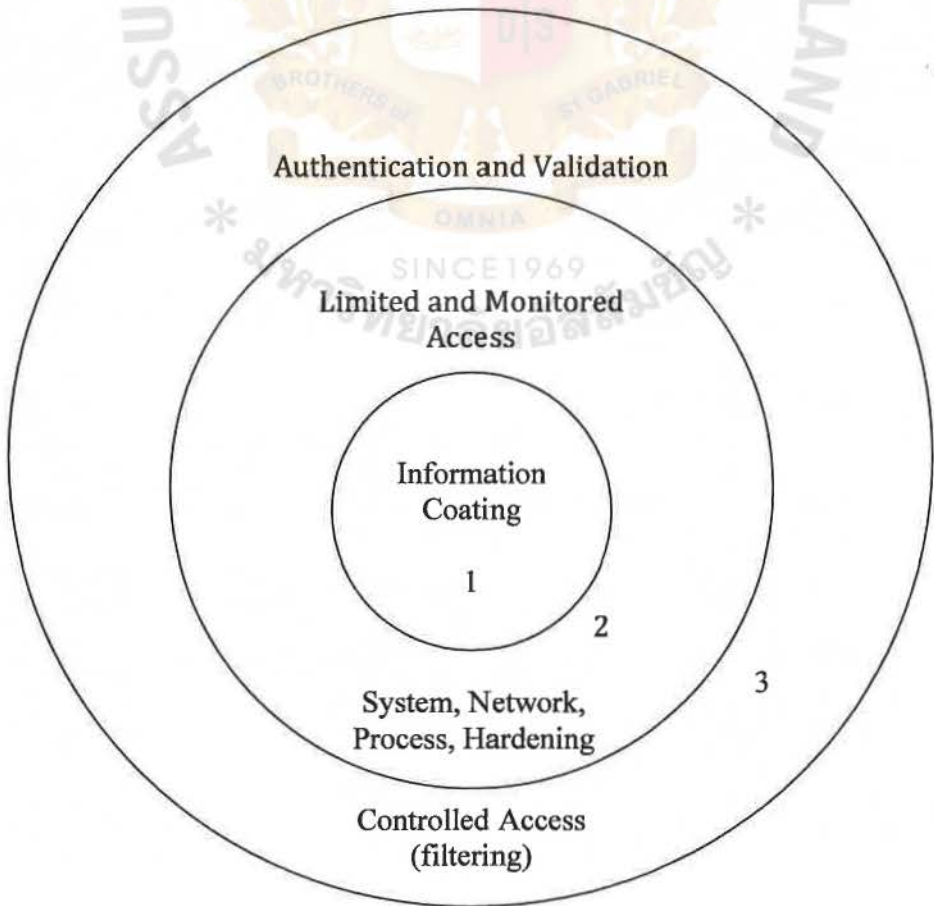


Figure 2.3 presents degrees of security in which an organization must address to meet the requirements of the organization that ranged from ultra-restrictive to extremely permissive and the degree of security that the organization wants will directly influence the degree to which any layer in the model is implemented (Calabrese, 2004).

Figure 2.3: Degrees of security

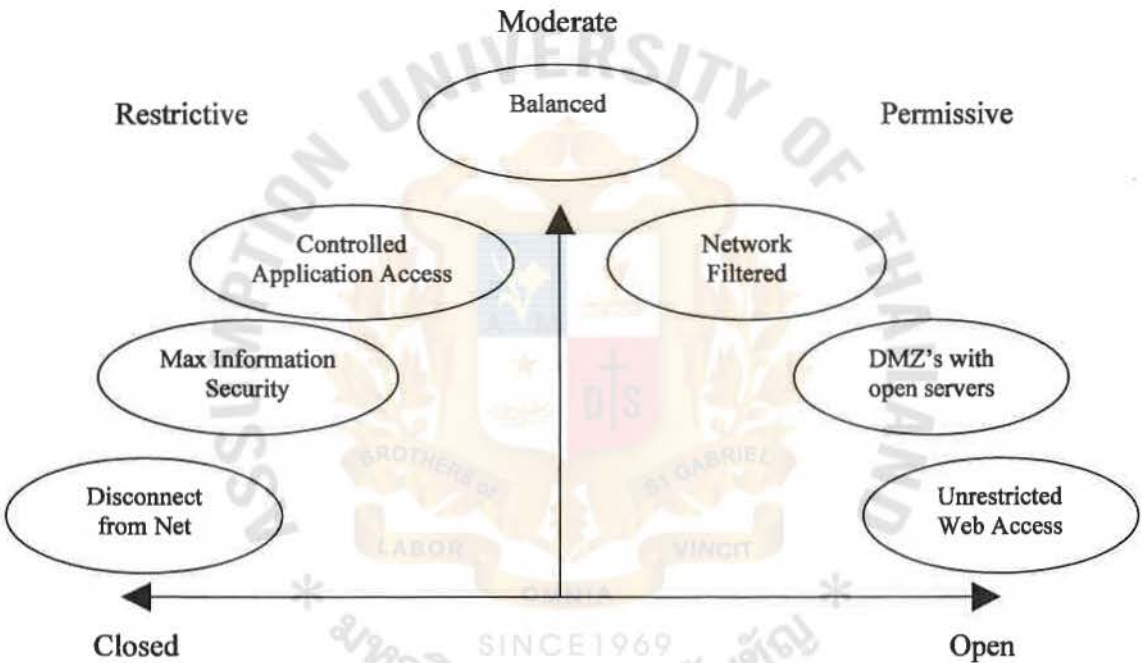
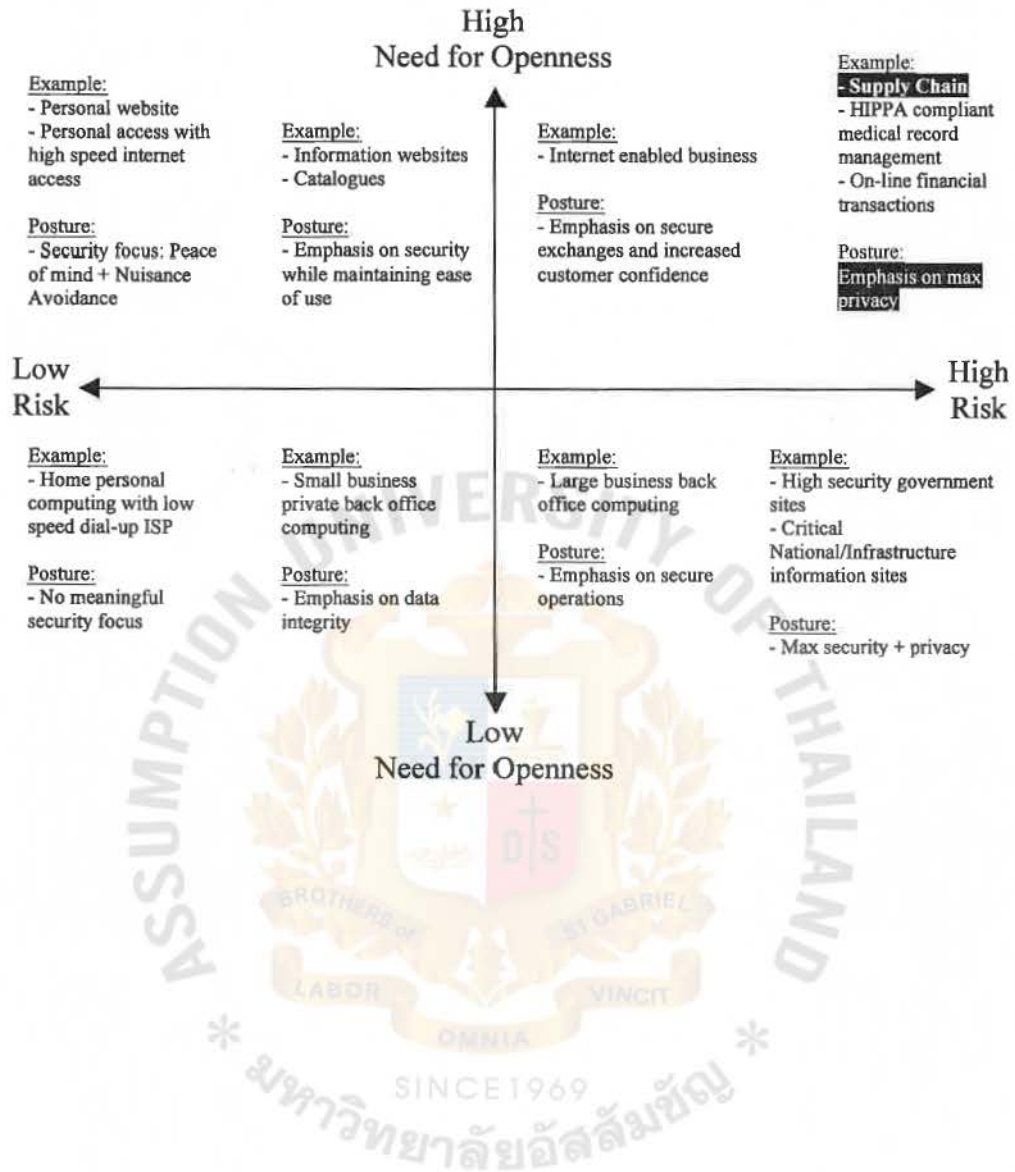


Figure 2.4: Risk-Openness-Value Paradigm



CHAPTER 3

PROPOSED TECHNIQUE

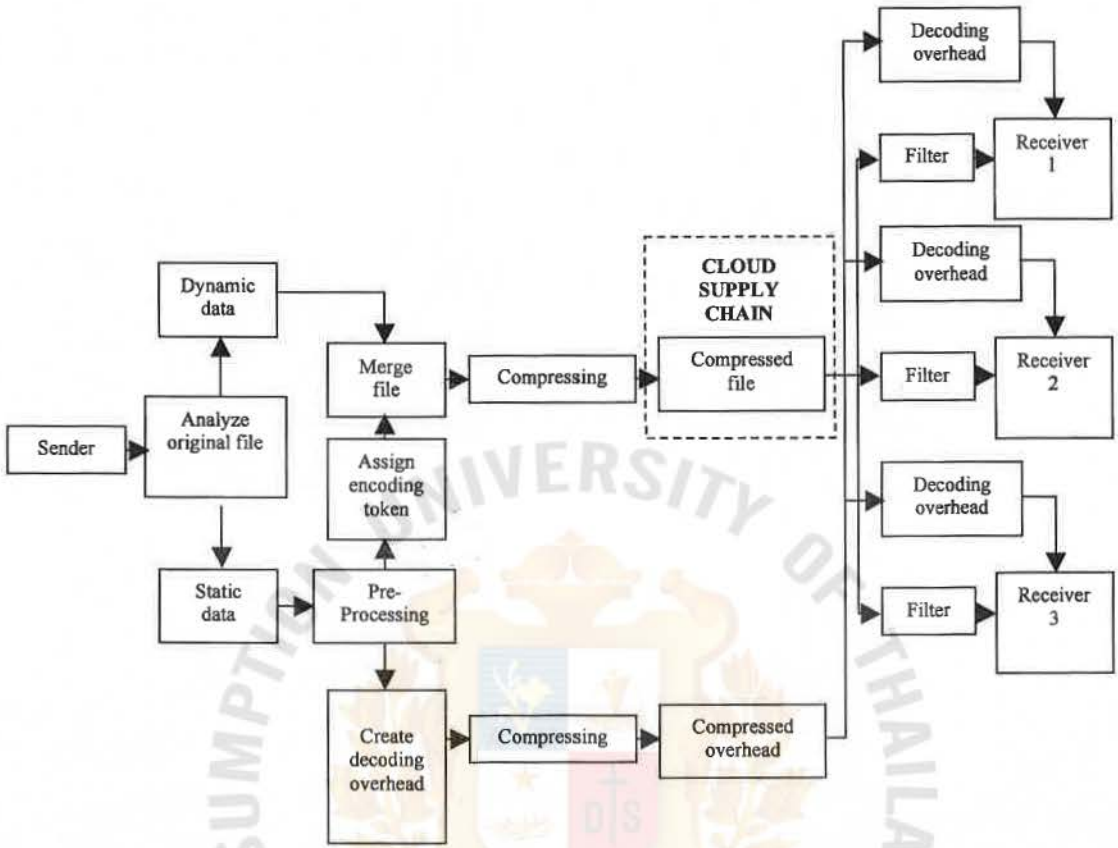
3.1 Overview

The prime concern of this study is to implement a simple approach that can improve compression performance of shared text data in structured tabular form in which cloud supply chain is selected for illustrating the proposed technique. An efficient text preprocessing algorithm for improving data compression that can encode specific multiple data into a single encoding token is introduced.

The proposed algorithm reduces number of encoding tokens used at the initial preprocessing stage itself for the static data by utilizing a single token for multiple-receiver data as many as possible. A single encoding token can support multiple receivers with different meanings and each receiver only sees the decoded data that are associated with that particular assigned receiver. The data visible in the cloud supply chain includes both static data that is preprocessed by the proposed algorithm and dynamic data that is filtered based on assigned receivers particularly.

By using the proposed preprocessing algorithm, the encoding tokens can be more efficiently utilized and decoding overheads are separately provided to particular assigned receivers. Therefore, the proposed algorithm can respectively increase compression performance of the shared text data in tabular form. There are three sub-techniques applied in this algorithm, which are 1) the use of single encoding token to many decoding overheads (one-to-many dictionary-based preprocessing), 2) data isolation, and 3) integrated solution. Figure 3.1 presents an overview paradigm of the proposed algorithm as a whole, while the encoding and decoding modules are elaborated in details presented in figure 3.2 and 3.3.

Figure 3.1: The proposed technique paradigm



3.2 Single Encoding Token to Multiple Decoding Overheads

In order to make the dictionary-based data compression technique more effective, the proposed study introduces another method to reduce the number of encoding tokens used for a particular set of original data. In this circumstance, static data in the original file is analyzed and unique in order to find distinctiveness of the static data contained in the file for the purpose of token utilization. Subsequently, a single encoding token is used based on a particular group of original data and multi-receivers in which it can be decoded into number of meanings differently based on sets of isolated decoding overheads that are transmitted to each assigned receiver differently and separately.

In general, one encoding token is used for a single meaning of an original data in which it is just only a one- to-one relationship. On the other hands, by using the proposed technique, the relationship of data coding is a one-to-many in which multiple meanings of decoding overheads can be referred just to a single encoding token.

3.3 Data Isolation

As cloud security is one of the important issues that enterprises have to be aware of and the main issue of dictionary-based compression technique is about an effective list of words used between sender and receiver known as overhead that may impact to compression performance, this research proposes a simple technique by utilizing the data isolation method to gain opportunities from isolated decoding overhead that directly improves the compression performance of the encoded file stored in the cloud since the decoding overhead is not taken into consideration because the overhead is only transmitted to each assigned receiver separately.

Another benefit of using isolated decoding overhead is to improve data confidentiality shared in the cloud, since the encoding tokens are assigned as one-to-many relationship with decoding overheads, each receiver only sees the decoded data that are only associated with that particular assigned receiver. Other receivers, hackers, or even eavesdroppers will not know that particular decoding overheads and are not able to interpret that encoded file into meaningful data.

3.4 Integrated Solution

In order to make the proposed preprocessing technique practical for data sharing in the cloud supply chain, the proposed technique is used together with selected existing archiving software as an integrated solution to improve data compression performance.

The proposed technique can bring some compression and data confidentiality at the initial preprocessing stage itself as aforementioned, and the compression performance is then improved again by using the selected archiving software to ensure that this integrated solution can effectively and efficiently provide good results in both compression performance and data confidentiality for cloud supply chain users.

Figure 3.2: Encoding module

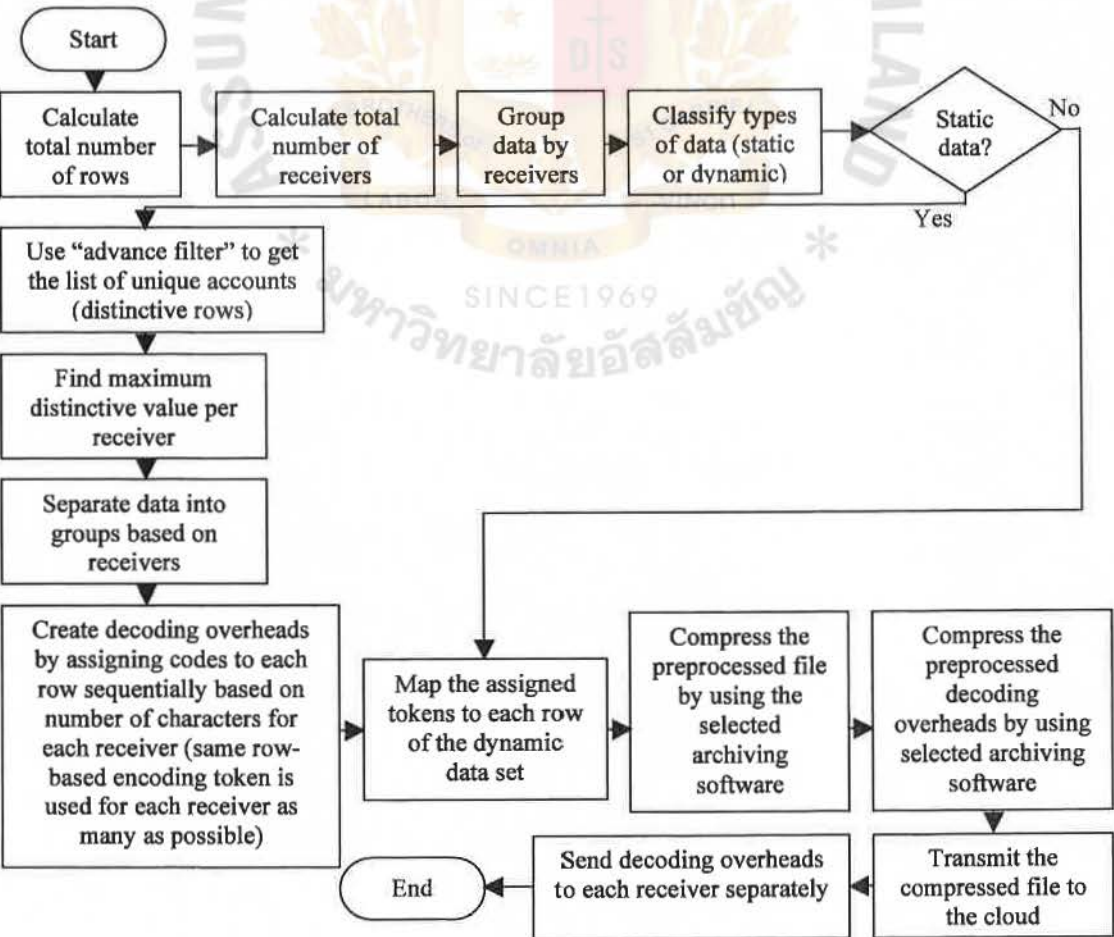
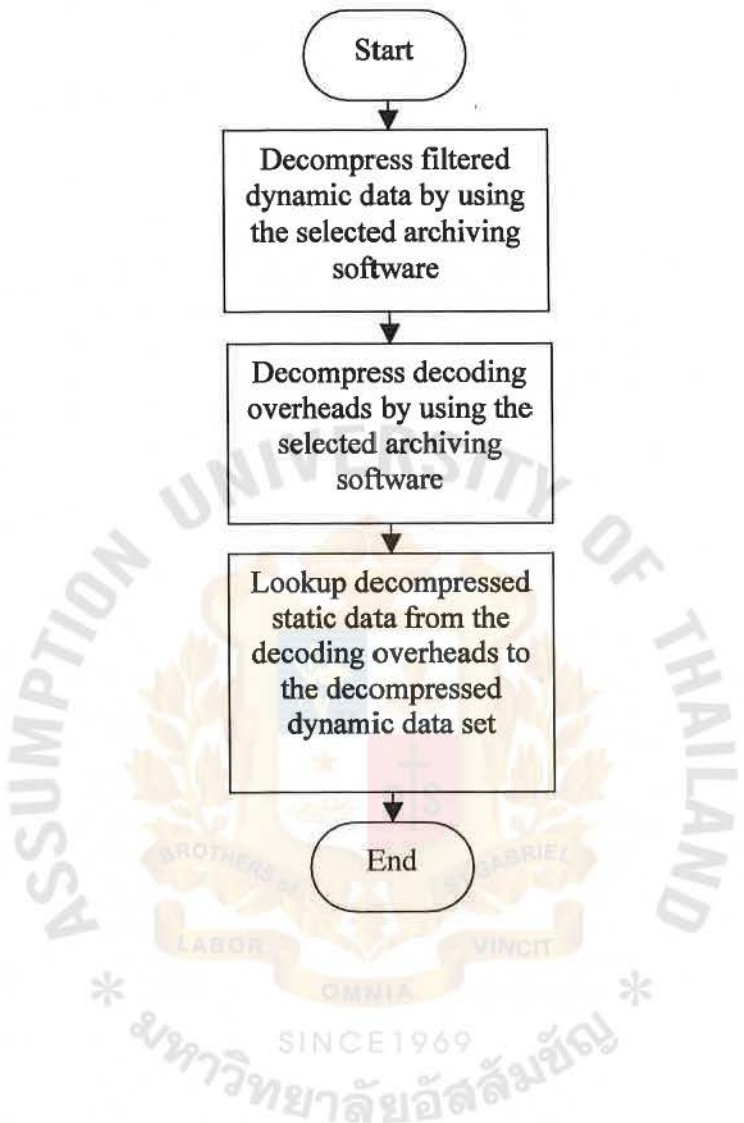


Figure 3.3: Decoding module



CHAPTER 4

DATA PREPARATION AND TESTING METHOD

4.1 Data Preparation

In general, there are two methods of choosing the test files for testing the compression performance. The first method is to use a well-known standard data. For examples, Calgary corpus, Canterbury corpus, or Silesia corpus. Another method is to prepare a new data set for testing. Actually, it is more convenient to use the standard sets of data because they are easier to compare new results with previous techniques. However, in order to test the compression performance of the proposed preprocessing algorithm, new data sets were prepared because the algorithm is designed specifically for text data in tabular form.

In this research, real data sets in spreadsheet format used in supply chain for communicating between manufacturers and raw material suppliers were selected. In order to test the compression performance in relation to number of receivers, static-data ratio (percentage of static data available in the original file), and size of duplicated static data (the static data that are duplicated in the original file); the data were modified and separated into 10 sets. The first 9 sets of the data were prepared as presented in table 4.1 in which they were used for measuring compression performance of the proposed technique in relation to static-data ratio and duplicated static-data ratio. The last set of the data contained 100 data rows in which it was used for testing minimum and maximum use of encoding tokens in relation to variation of the number of receivers.

Table 4.1: Data sets used for testing compression performance in relation to the ratios of static data and duplicated static data

Data set	Ratio of static data	Ratio of duplicated static data
1	25%	25%
2	25%	50%
3	25%	75%
4	50%	25%
5	50%	50%
6	50%	75%
7	75%	25%
8	75%	50%
9	75%	75%

4.2 Simulation Method

In this research, Microsoft Excel 2003 macros and functions were used for simulation purposes to simulate this one-to-many dictionary-based preprocessing algorithm in which encoding was preprocessed at the sender by using the encoding module at initial stage prior to the next data compression integrated with the selected archiving software. The simulation was also used for decoding module in which the decoding overheads prepared for each assigned receiver are used in retrieving the original messages from the shared file.

Table 4.2 presents the functions of Microsoft Excel 2003 that were used in this research.

Table 4.2: Functions of Microsoft Excel 2003 that were used in this research

Function	Description
Macro	It is an action or a set of actions you can use to automate tasks. Macros are recorded in the Visual Basic for Applications programming language. In this research, this function was used to automatically calculate duplication of text data for assigning encoding tokens and decoding overheads.
Pivot Table	A PivotTable report is an interactive table that quickly combines and compares large amounts of data. In this research, the pivot table is applied for the purpose of counting total number of rows, number of receivers, grouping the data by receivers, and finding maximum distinctive value per receiver.
Advanced filter	In this research, the advanced filter is applied in order to find unique records of the distinctive rows.
LEN	It is a function to returns the number of characters in a text string. In this research, this function is used in order to calculate number of characters of static data per row for assigning encoding tokens efficiently.
Sort	Numbers are sorted from the smallest negative number to the largest positive number. In this research, this function is used in order to sort number of characters of static data per row for assigning encoding tokens efficiently. The higher number of characters, the shorter of the encoding token will be assigned.
VLOOKUP	It is applied as a searching tool for a value in the specified column of a table, and then returns a value in the same row from a column that is specified in the table. In this research, it is used for mapping the assigned tokens to each row of the dynamic data set as well as mapping decompressed static data from the decoding overheads back to the decompressed dynamic data set.

4.3 Selected Archiving Software

In this research, the tested software products were selected based on their algorithms to support text data compression, supported operating system (Windows), and supported archive reading and writing (ZIP), which are 7-Zip, PeaZip, WinAce, WinRAR, and WinZip. Table 4.3 presents selected software products and data compression algorithms that are implemented by these software products (Konecki et al, 2011). This research focuses on how well these software products can support the

purposed one-to-many dictionary-based preprocessing technique as the integration tool by seeing the differences in compression efficiency among the integrated solutions. Integrations of each software product with the proposed algorithm were tested. The results of compression ratio of all integrations are presented and compared in the next chapter with elaborations.

Table 4.3: Selected archiving software

Software	Algorithms
7-Zip	Program used filters, LZ77, LZMA, PPM, BWT
PeaZip	LZ77, Huffman, LZMA, PPM
WinAce	Program used filters, LZ77, Huffman
WinRAR	Program used filters, LZ77, PPM, Huffman
WinZip	Shannon-Fano, PPM, Huffman, LZH, LZW

4.4 Selected Existing Preprocessing Technique

Word Replacement Transformation (WRT) was selected as an existing preprocessing technique for the purpose of compression performance comparison. According to a previous research, this transformation technique was the recent algorithm in which it was compared with another two well-known algorithms, LIPT and StarNT. The results presented that WRT provided better compression performance over LIPT and StarNT. WRT also operated faster than StarNT because WRT works based on the hashing technique, which can speed up the encoding and decoding process while StarNT uses a ternary search tree for maintaining the dictionary. Moreover, WRT carried out well with larger text file compared to StarNT and LIPT by providing a better compression ratio on larger corpora (Rexline and Robert, 2011).

CHAPTER 5

EXPERIMENTAL RESULTS AND ANALYSES

5.1 Results and Comparisons of the Compression Performance in Relation to Ratios of Static Data and Duplicated Static Data

In this research, the testing results are classified into 2 sections as 1) compression performance in relation to ratios of static data and duplicated static data and 2) minimum and maximum encoding tokens required in relation to number of receivers.

As the data were prepared by using real data in spreadsheet format used for communicating between manufacturers and raw material suppliers instead of selecting a well-known standard data set and the researcher would like to study the relationships of static data ratio and duplicated static data ratio toward data compression performance of the proposed preprocessing algorithm, therefore, the analyses of compression performance were separated based on the following 9 data sets and tested with selected archiving software, selected archiving software with WRT technique, and selected archiving software with the proposed preprocessing technique as presented in table 5.1

The last set of the data contained 100 data rows in which it was used for testing minimum and maximum use of encoding tokens in relation to variation of the number of receivers. The result of this analysis is illustrated in figure 5.17.

Table 5.1: Details of data sets and compression algorithms used for the analyses

Data set	Compression algorithm
25% of static data and 25% of duplicated static data	archiving software, archiving software with WRT, archiving software with proposed technique
50% of static data and 25% of duplicated static data	archiving software, archiving software with WRT, archiving software with proposed technique
75% of static data and 25% of duplicated static data	archiving software, archiving software with WRT, archiving software with proposed technique
25% of static data and 50% of duplicated static data	archiving software, archiving software with WRT, archiving software with proposed technique
50% of static data and 50% of duplicated static data	archiving software, archiving software with WRT, archiving software with proposed technique
75% of static data and 50% of duplicated static data	archiving software, archiving software with WRT, archiving software with proposed technique
25% of static data and 75% of duplicated static data	archiving software, archiving software with WRT, archiving software with proposed technique
50% of static data and 75% of duplicated static data	archiving software, archiving software with WRT, archiving software with proposed technique
75% of static data and 75% of duplicated static data	archiving software, archiving software with WRT, archiving software with proposed technique

Figure 5.1, table 5.2, and table 5.3 represent compression performances of the first data set that contains 25% of static data and 25% of duplicated static data, the results show that only some of integrations of selected archiving software with the proposed preprocessing technique generated better compression results. In this condition, only the combinations of the proposed technique with 7-Zip and WinRAR could generate better compression ratio and space saving when they were compared with WRT algorithm. Integration of WRT with PeaZip, WinAce, and WinZip could generate better compression performance. However, integrations of the proposed preprocessing technique with all of the selected archiving software could provide better compression performance when they were compared with using of only selected archiving software individually, ranged from 1% to 14.9% of improvement in compression outputs.

Figure 5.1: Comparison of compressed file sizes

(25% of static data and 25% of duplicated static data)

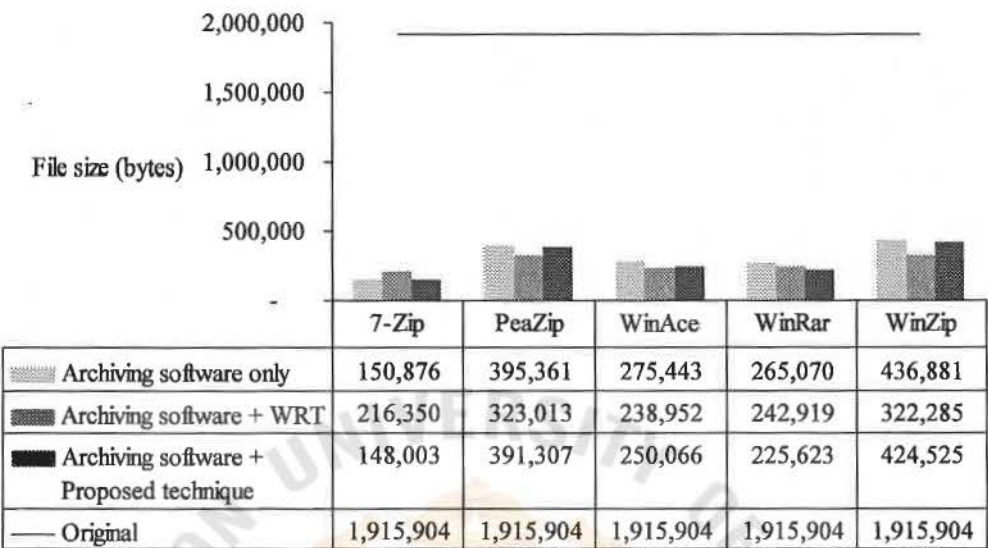


Table 5.2: Comparison of compression ratios

(25% of static data and 25% of duplicated static data)

	Archiving software only	Archiving software + WRT	Archiving software + Proposed technique
7-Zip	0.079	0.113	0.077
PeaZip	0.206	0.169	0.204
WinAce	0.144	0.125	0.131
WinRAR	0.138	0.127	0.118
WinZip	0.228	0.168	0.222

Table 5.3: Comparison of space saving percentage

(25% of static data and 25% of duplicated static data)

	Archiving software only	Archiving software + WRT	Archiving software + Proposed technique
7-Zip	92.13%	88.71%	92.28%
PeaZip	79.36%	83.14%	79.58%
WinAce	85.62%	87.53%	86.95%
WinRAR	86.16%	87.32%	88.22%
WinZip	77.20%	83.18%	77.84%

Figure 5.2, table 5.4, and table 5.5 represent compression performances of the second data set that contains 50% of static data and 25% of duplicated static data, the results show similar results as the first data set in which only some of integrations of selected archiving software with the proposed preprocessing technique generated better compression results. In this case, the combinations of the proposed technique with 7-Zip, WinAce, and WinRAR could generate better compression ratio and space saving when they were compared with WRT algorithm. Integration of WRT with PeaZip and WinZip could generate better compression performance. However, integrations of the proposed preprocessing technique with all of the selected archiving software could still provide better compression performance when they were compared with using of only selected archiving software individually, ranged from 6.9% to 30.49% of improvement in compression outputs.

Figure 5.2: Comparison of compressed file sizes
(50% of static data and 25% of duplicated static data)

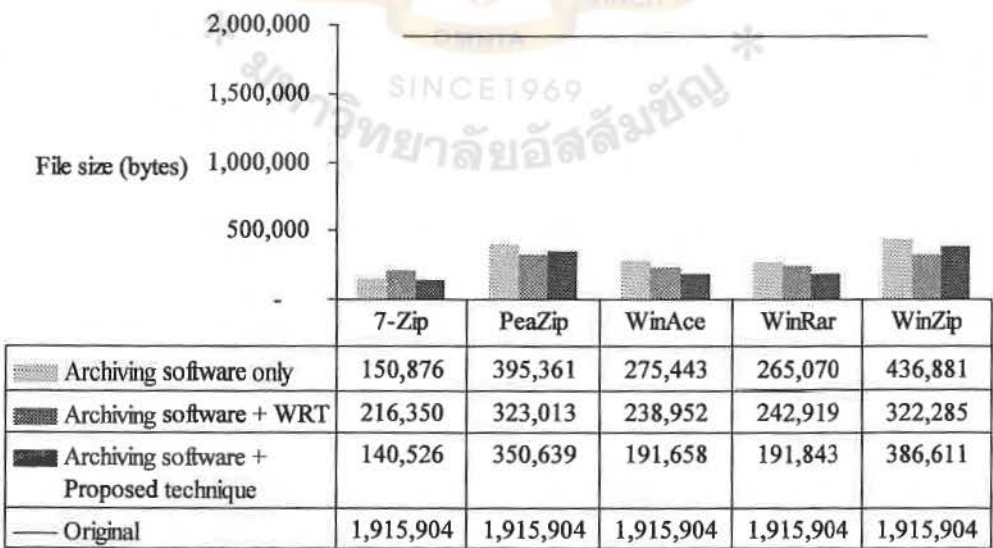


Table 5.4: Comparison of compression ratios

(50% of static data and 25% of duplicated static data)

	Archiving software only	Archiving software + WRT	Archiving software + Proposed technique
7-Zip	0.079	0.113	0.073
PeaZip	0.206	0.169	0.183
WinAce	0.144	0.125	0.100
WinRAR	0.138	0.127	0.100
WinZip	0.228	0.168	0.202

Table 5.5: Comparison of space saving percentage

(50% of static data and 25% of duplicated static data)

	Archiving software only	Archiving software + WRT	Archiving software + Proposed technique
7-Zip	92.13%	88.71%	92.67%
PeaZip	79.36%	83.14%	81.70%
WinAce	85.62%	87.53%	90.00%
WinRAR	86.16%	87.32%	89.99%
WinZip	77.20%	83.18%	79.82%

Figure 5.3, table 5.6, and table 5.7 represent compression performances of the third data set that contains 75% of static data and 25% of duplicated static data, the results show similar results as the first and the second data set in which only some of integrations of selected archiving software with the proposed preprocessing technique generated better compression results which were 7-Zip, WinAce, and WinRAR that could generate better compression ratio and space saving when they were compared with WRT algorithm. Integration of WRT with PeaZip and WinZip could still generate better compression performance. However, integrations of the proposed preprocessing technique with all of the selected archiving software could still provide better compression performance when they were compared with using of only selected archiving software individually, ranged from 17.2% to 46.6% of improvement in compression outputs. The compression outputs were improved from

the first, the second, and the third data set respectively in relation to the increases of static data ratio.

Figure 5.3: Comparison of compressed file sizes

(75% of static data and 25% of duplicated static data)

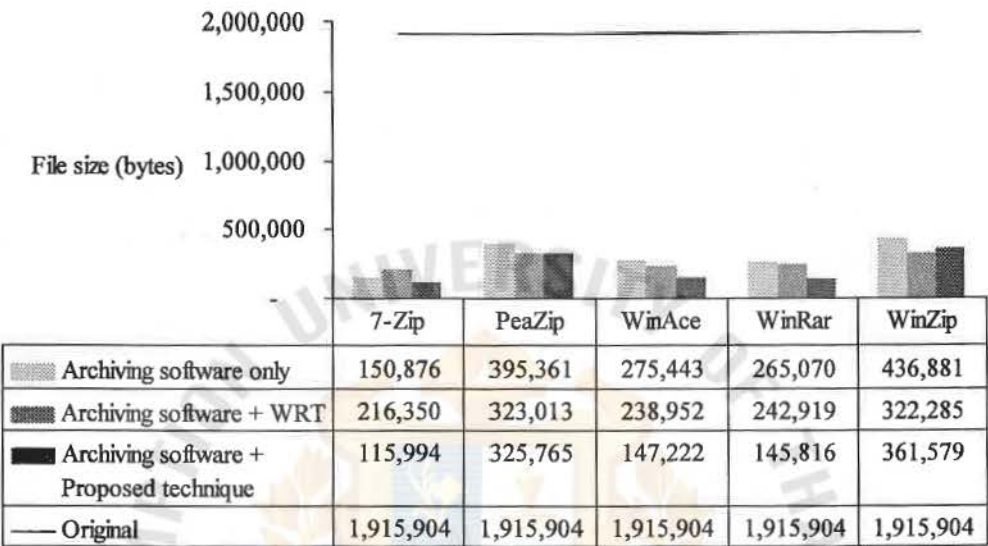


Table 5.6: Comparison of compression ratios

(75% of static data and 25% of duplicated static data)

	Archiving software only	Archiving software + WRT	Archiving software + Proposed technique
7-Zip	0.079	0.113	0.061
PeaZip	0.206	0.169	0.170
WinAce	0.144	0.125	0.077
WinRAR	0.138	0.127	0.076
WinZip	0.228	0.168	0.189

Table 5.7: Comparison of space saving percentage

(75% of static data and 25% of duplicated static data)

	Archiving software only	Archiving software + WRT	Archiving software + Proposed technique
7-Zip	92.13%	88.71%	93.95%
PeaZip	79.36%	83.14%	83.00%
WinAce	85.62%	87.53%	92.32%
WinRAR	86.16%	87.32%	92.39%
WinZip	77.20%	83.18%	81.13%

Figure 5.4, table 5.8, and table 5.9 represent compression performances of the fourth data set that contains 25% of static data and 50% of duplicated static data, the results show similar results as the first three data sets aforementioned in which only some of integrations of selected archiving software with the proposed preprocessing technique generated better compression results which were 7-Zip and WinRAR that could generate better compression ratio and space saving when they were compared with WRT algorithm. Integration of WRT with PeaZip, WinAce, and WinZip could still generate better compression performance.

Figure 5.4: Comparison of compressed file sizes

(25% of static data and 50% of duplicated static data)

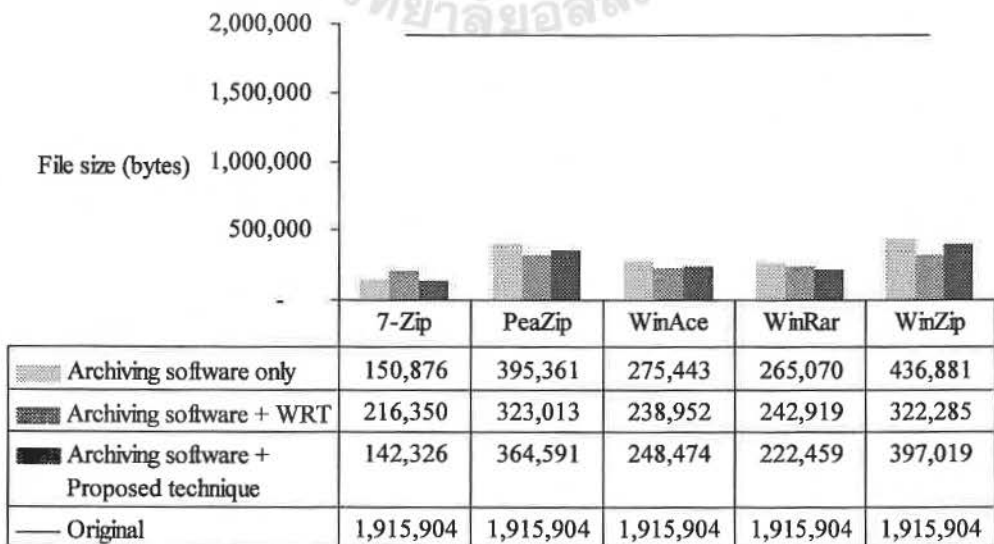


Table 5.8: Comparison of compression ratios

(25% of static data and 50% of duplicated static data)

	Archiving software only	Archiving software + WRT	Archiving software + Proposed technique
7-Zip	0.079	0.113	0.074
PeaZip	0.206	0.169	0.190
WinAce	0.144	0.125	0.130
WinRAR	0.138	0.127	0.116
WinZip	0.228	0.168	0.207

Table 5.9: Comparison of space saving percentage

(25% of static data and 50% of duplicated static data)

	Archiving software only	Archiving software + WRT	Archiving software + Proposed technique
7-Zip	92.13%	88.71%	92.57%
PeaZip	79.36%	83.14%	80.97%
WinAce	85.62%	87.53%	87.03%
WinRAR	86.16%	87.32%	88.39%
WinZip	77.20%	83.18%	79.28%

Figure 5.5, table 5.10, and table 5.11 represent compression performances of the fifth data set that contains 50% of static data and 50% of duplicated static data, the results show better results as compared to the first four data sets aforementioned in which four out of five integrations of selected archiving software with the proposed preprocessing technique generated better compression results which were 7-Zip, PeaZip, WinAce, and WinRAR that could generate better compression ratio and space saving when they were compared with WRT algorithm. Except only the integration of WRT with WinZip that could still generate better compression performance.

Figure 5.5: Comparison of compressed file sizes

(50% of static data and 50% of duplicated static data)

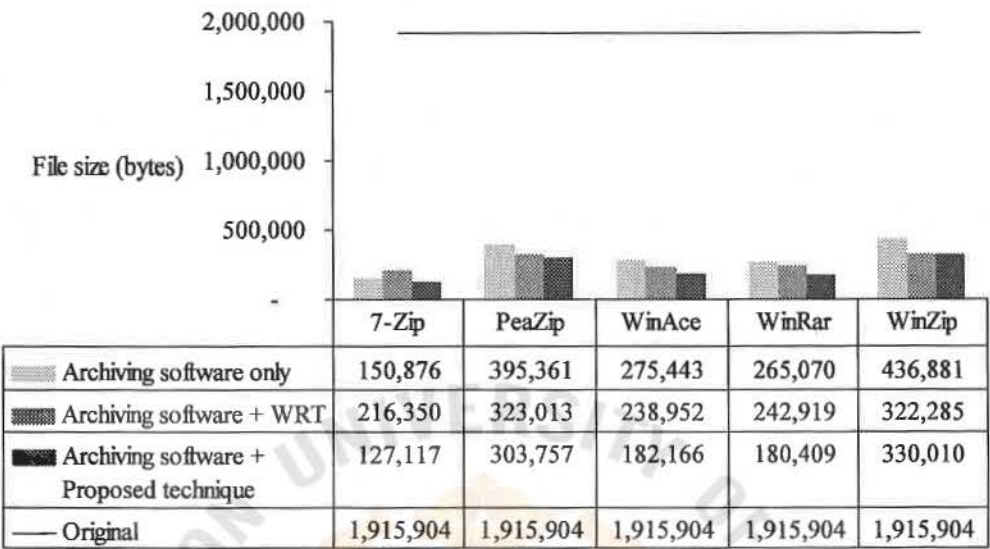


Table 5.10: Comparison of compression ratios

(50% of static data and 50% of duplicated static data)

	Archiving software only	Archiving software + WRT	Archiving software + Proposed technique
7-Zip	0.079	0.113	0.066
PeaZip	0.206	0.169	0.159
WinAce	0.144	0.125	0.095
WinRAR	0.138	0.127	0.094
WinZip	0.228	0.168	0.172

Table 5.11: Comparison of space saving percentage

(50% of static data and 50% of duplicated static data)

	Archiving software only	Archiving software + WRT	Archiving software + Proposed technique
7-Zip	92.13%	88.71%	93.37%
PeaZip	79.36%	83.14%	84.15%
WinAce	85.62%	87.53%	90.49%
WinRAR	86.16%	87.32%	90.58%
WinZip	77.20%	83.18%	82.78%

Figure 5.6, table 5.12, and table 5.13 represent compression performances of the sixth data set that contains 75% of static data and 50% of duplicated static data, the results show better results as compared to the first five data sets mentioned previously in which all of the integrations of selected archiving software with the proposed preprocessing technique generated better compression results which were 7-Zip, PeaZip, WinAce, WinRAR, and WinZip could generate better compression ratio and space saving when they were compared with WRT algorithm. It could be concluded that increases of the isolated static data and duplicated static data are directly proportional to compression performance of the proposed technique.

Figure 5.6: Comparison of compressed file sizes
(75% of static data and 50% of duplicated static data)

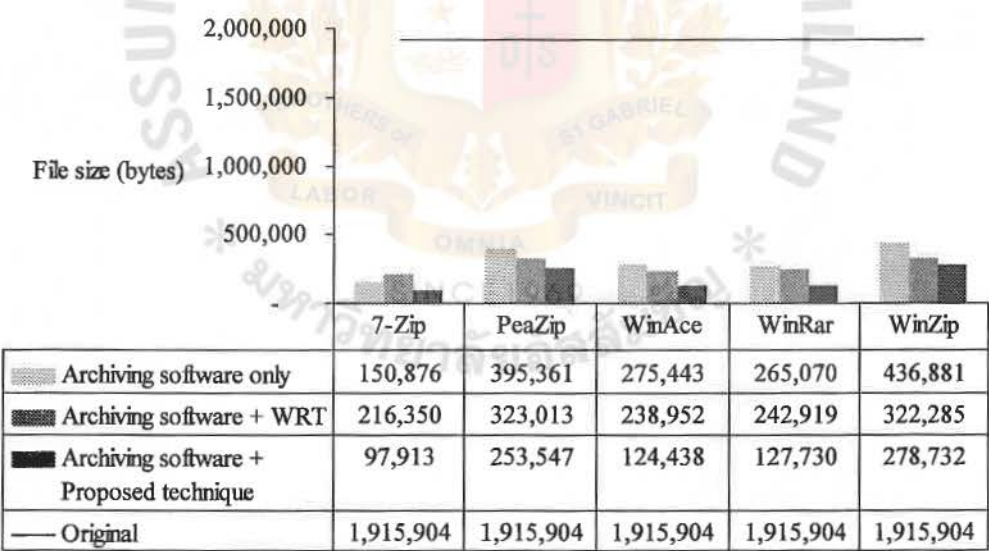


Table 5.12: Comparison of compression ratios

(75% of static data and 50% of duplicated static data)

	Archiving software only	Archiving software + WRT	Archiving software + Proposed technique
7-Zip	0.079	0.113	0.051
PeaZip	0.206	0.169	0.132
WinAce	0.144	0.125	0.065
WinRAR	0.138	0.127	0.067
WinZip	0.228	0.168	0.145

Table 5.13: Comparison of space saving percentage

(75% of static data and 50% of duplicated static data)

	Archiving software only	Archiving software + WRT	Archiving software + Proposed technique
7-Zip	92.13%	88.71%	94.89%
PeaZip	79.36%	83.14%	86.77%
WinAce	85.62%	87.53%	93.50%
WinRAR	86.16%	87.32%	93.33%
WinZip	77.20%	83.18%	85.45%

Figure 5.7, table 5.14, and table 5.15 represent compression performances of the seventh data set that contains 25% of static data and 75% of duplicated static data, the compression performance was lower as compared to the fifth data set because ratio of static data was reduced from 75% to 25%, therefore, only 7-Zip, and WinRAR could generate better compression results when they were compared with WRT algorithm. It was founded that decrease of the isolated static data ratio is directly proportional to the lower of compression performance of the proposed technique.

Figure 5.7: Comparison of compressed file sizes

(25% of static data and 75% of duplicated static data)

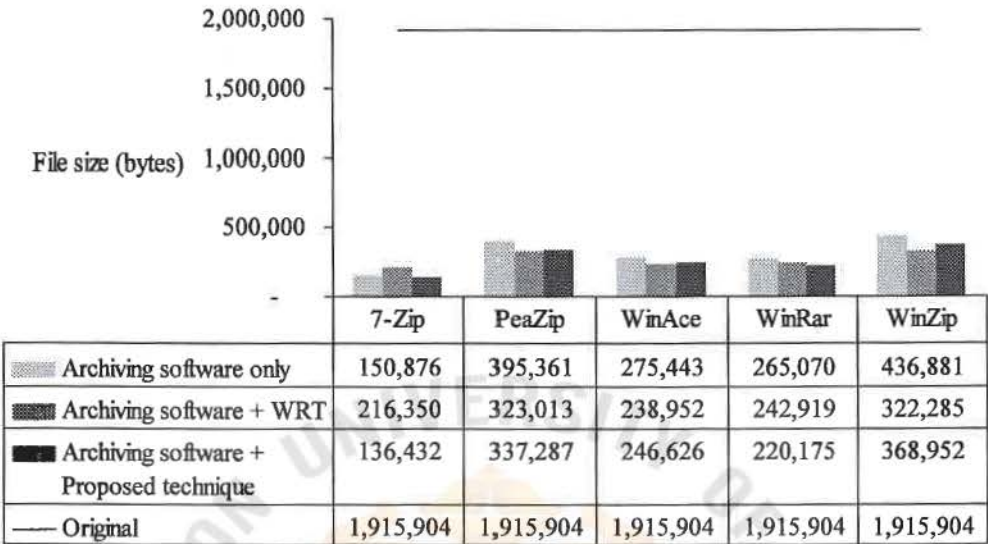


Table 5.14: Comparison of compression ratios

(25% of static data and 75% of duplicated static data)

	Archiving software only	Archiving software + WRT	Archiving software + Proposed technique
7-Zip	0.079	0.113	0.071
PeaZip	0.206	0.169	0.176
WinAce	0.144	0.125	0.129
WinRAR	0.138	0.127	0.115
WinZip	0.228	0.168	0.193

Table 5.15: Comparison of space saving percentage

(25% of static data and 75% of duplicated static data)

	Archiving software only	Archiving software + WRT	Archiving software + Proposed technique
7-Zip	92.13%	88.71%	92.88%
PeaZip	79.36%	83.14%	82.40%
WinAce	85.62%	87.53%	87.13%
WinRAR	86.16%	87.32%	88.51%
WinZip	77.20%	83.18%	80.74%

Figure 5.8, table 5.16, and table 5.17 represent compression performances of the eighth data set that contains 50% of static data and 75% of duplicated static data, the results were improved in which all of the integrations of selected archiving software with the proposed preprocessing technique generated better compression results which were 7-Zip, PeaZip, WinAce, WinRAR, and WinZip could generate better compression ratio and space saving when they were compared with WRT algorithm. It was confirmed that that increases of the isolated static data and duplicated static data are directly proportional to compression performance of the proposed technique when the results were compared with all of six data sets previously mentioned.

Figure 5.8: Comparison of compressed file sizes
(50% of static data and 75% of duplicated static data)

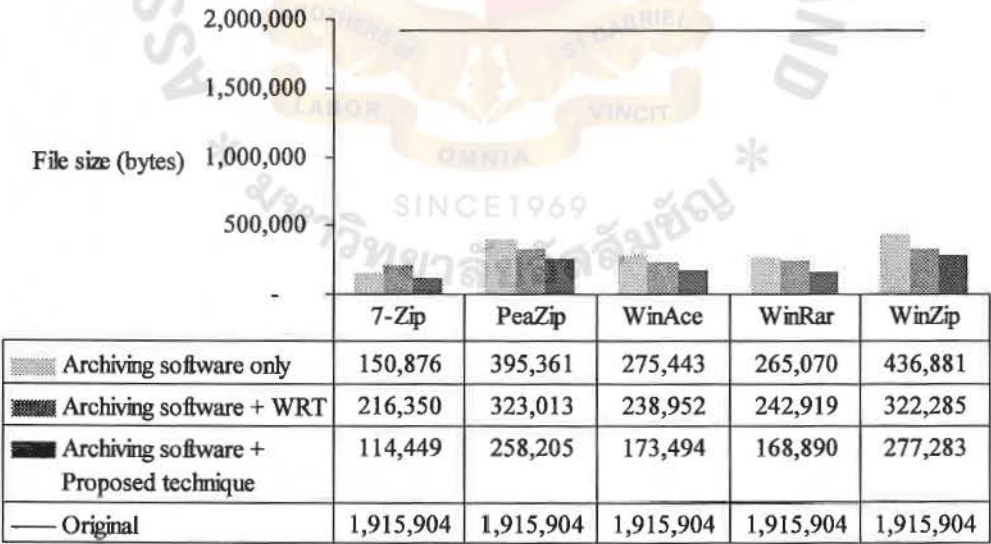


Table 5.16: Comparison of compression ratios

(50% of static data and 75% of duplicated static data)

	Archiving software only	Archiving software + WRT	Archiving software + Proposed technique
7-Zip	0.079	0.113	0.060
PeaZip	0.206	0.169	0.135
WinAce	0.144	0.125	0.091
WinRAR	0.138	0.127	0.088
WinZip	0.228	0.168	0.145

Table 5.17: Comparison of space saving percentage

(50% of static data and 75% of duplicated static data)

	Archiving software only	Archiving software + WRT	Archiving software + Proposed technique
7-Zip	92.13%	88.71%	94.03%
PeaZip	79.36%	83.14%	86.52%
WinAce	85.62%	87.53%	90.94%
WinRAR	86.16%	87.32%	91.18%
WinZip	77.20%	83.18%	85.53%

Figure 5.9, table 5.18, and table 5.19 represent compression performances of the ninth data set that contains 75% of static data and 75% of duplicated static data. In this condition, the proposed preprocessing technique could generate the best result in which the results of all integrations of selected archiving software with the proposed preprocessing technique could bring the best compression results in all perspectives. All over again, it was confirmed that that increases of the isolated static data and duplicated static data are directly proportional to compression performance of the proposed technique when the results were compared with all of seven data sets aforementioned. By comparing percentages of improvement between the use of selected archiving software individually and the use of the proposed preprocessing technique as an integration tool, the results were improved by ranging from 44.0% to 60.5% when the proposed technique was employed.

Figure 5.9: Comparison of compressed file sizes

(75% of static data and 75% of duplicated static data)

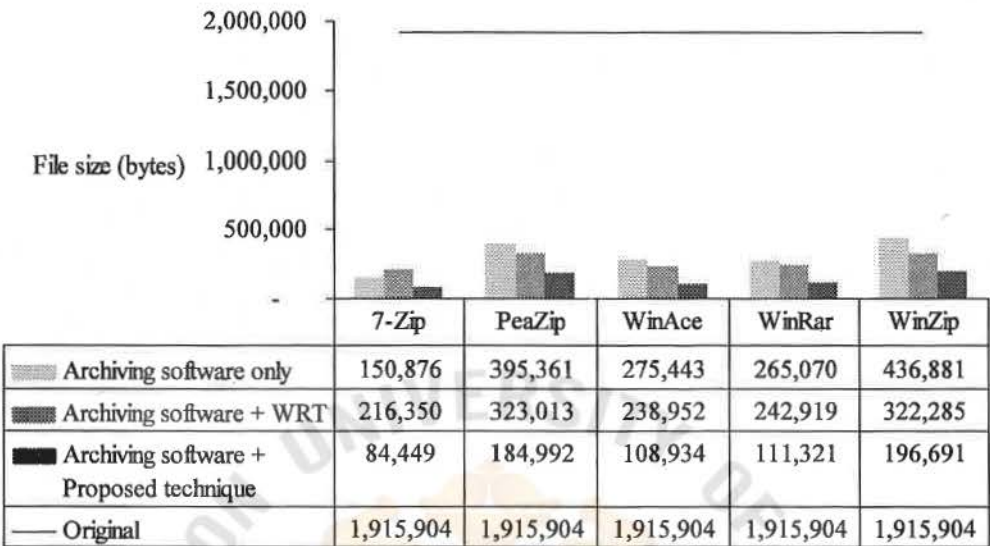


Table 5.18: Comparison of compression ratios

(75% of static data and 75% of duplicated static data)

	Archiving software only	Archiving software + WRT	Archiving software + Proposed technique
7-Zip	0.079	0.113	0.044
PeaZip	0.206	0.169	0.097
WinAce	0.144	0.125	0.057
WinRAR	0.138	0.127	0.058
WinZip	0.228	0.168	0.103

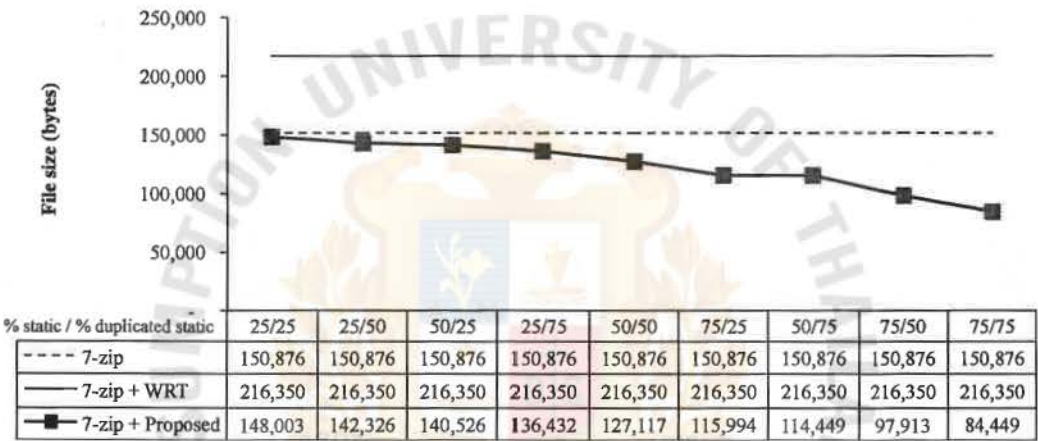
Table 5.19: Comparison of space saving percentage

(75% of static data and 75% of duplicated static data)

	Archiving software only	Archiving software + WRT	Archiving software + Proposed technique
7-Zip	92.13%	88.71%	95.59%
PeaZip	79.36%	83.14%	90.34%
WinAce	85.62%	87.53%	94.31%
WinRAR	86.16%	87.32%	94.19%
WinZip	77.20%	83.18%	89.73%

In order to propose the most suitable archiving software that works best with the proposed preprocessing algorithm, figure 5.10 to figure 5.14 represent the analytical results of particular archiving software that was used individually, used with WRT, and used with the proposed technique.

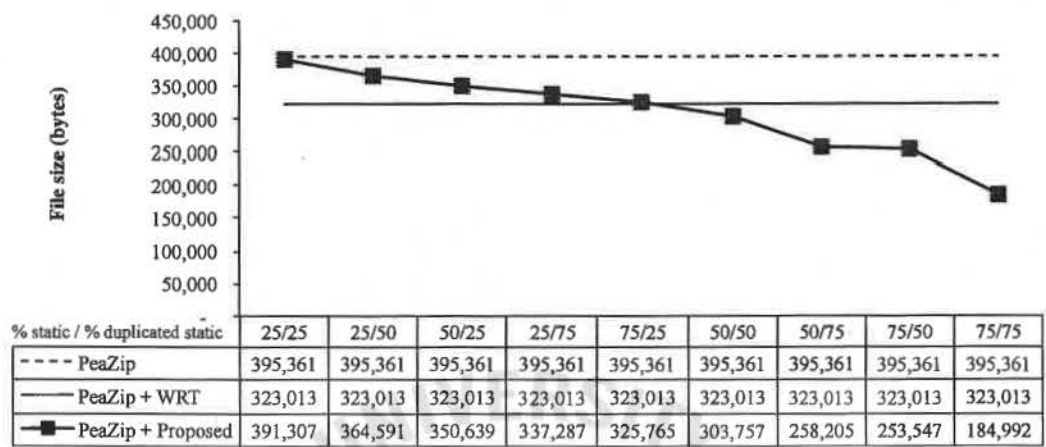
Figure 5.10: Comparison of compressed file size based on the use of 7-Zip as the selected archiving software



As presented in figure 5.10, the integration of 7-Zip with the proposed preprocessing technique generated better results for all data sets when the outputs were compared with the use of 7-Zip individually as well as the use of WRT as the preprocessing tool, and significantly outperformed when the ratio of static data was at 75%. However, there were only slightly improvements when the static data ratio was at 25%.

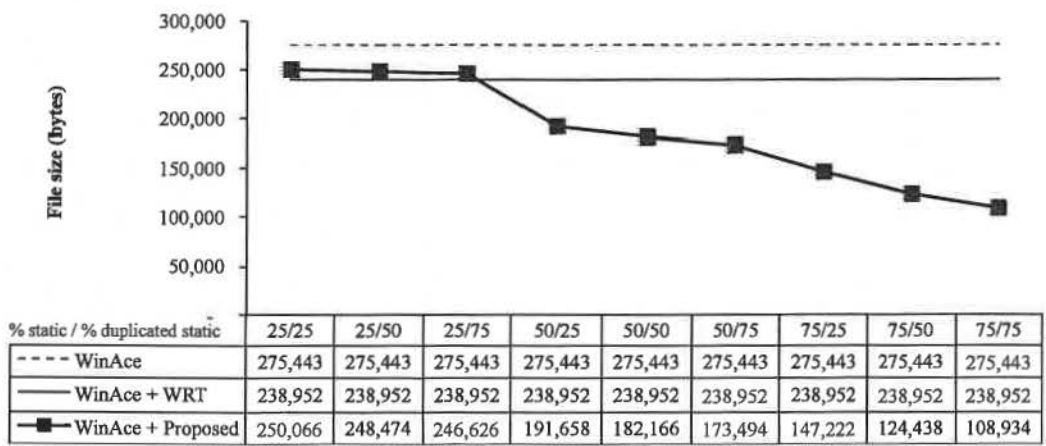
By using 7-Zip, it was found that the integration of WRT would generate the lower compression performance when it was compared with the use of 7-Zip individually.

Figure 5.11: Comparison of compressed file size based on the use of PeaZip as the selected archiving software



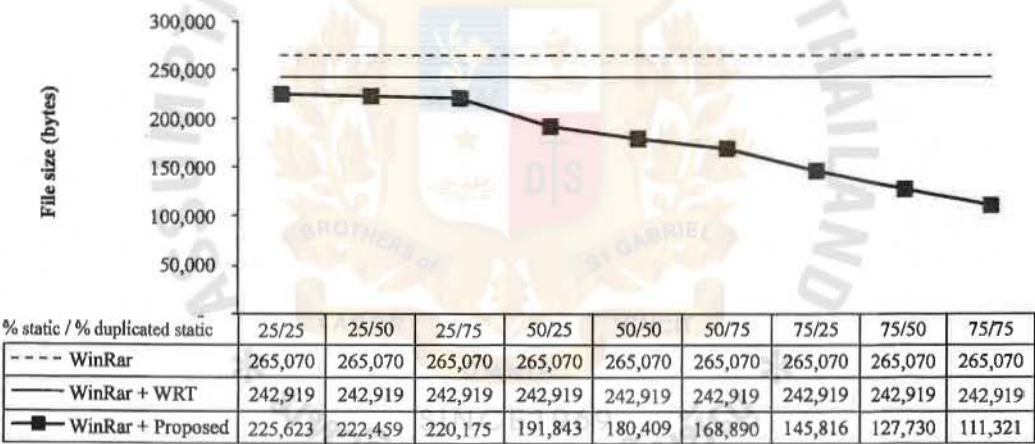
As presented in figure 5.11, the integration of PeaZip with the proposed preprocessing technique generated better results for only four out of nine of the data sets when the outputs were compared with the use of PeaZip individually as well as the use of WRT as the preprocessing tool in which the results of the proposed technique would be better than WRT when the ratios of both static data and duplicated static data were greater than 50%.

Figure 5.12: Comparison of compressed file size based on the use of WinAce as the selected archiving software



As presented in figure 5.12, the integration of WinAce with the proposed preprocessing technique generated better results when it was compared with PeaZip but the performance was still lower than the integration with 7-Zip. There were only six out of nine of the data sets in which compression performance of the proposed technique were better than WRT. According to the results, data with 25% of static data ratio was not recommended for the use of the proposed technique with WinAce as the compression tool.

Figure 5.13: Comparison of compressed file size based on the use of WinRAR as the selected archiving software

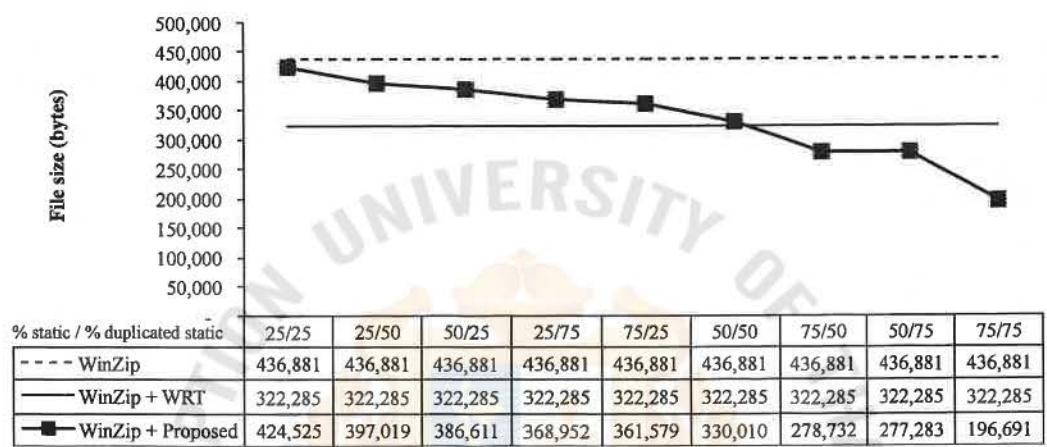


As presented in figure 5.13, the integration of WinRAR with the proposed preprocessing technique generated better results for all data sets when the outputs were compared with the use of WinRAR individually as well as the use of WRT as the preprocessing tool, and significantly outperformed when the ratios of static data and duplicated static data were at 50% and 75%, and there were only slightly improvements when the static data ratio was at 25%.

However, by comparing it with 7-Zip, the results of 7-Zip still outperformed in term of data compression performance in all conditions consisting of the use of

archiving software individually, the WRT integration, and also the integration with the proposed preprocessing technique.

Figure 5.14: Comparison of compressed file size based on the use of WinZip as the selected archiving software



As presented in figure 5.14, the integration of WinZip with the proposed preprocessing technique generated better results for only three out of nine data sets when the outputs were compared with the use of WinZip individually as well as the use of WRT as the preprocessing tool in which the results of the proposed technique would be better than WRT only when the ratios of static data was at 75%, or at 50% if the ratio of duplicated static data was at 75%.

By comparing WinZip with the aforementioned four archiving software in all conditions consisting of the use of archiving software individually, the WRT integration, and also the integration with the proposed preprocessing technique; WinZip provided the lowest compression performance.

The results of low compression performance in some conditions might be caused by unsuitable integration of the prepared data text files with some of the selected archiving software.

According to the experimental results, 7-Zip is the best archiving software that works best with the prepared text files. The integration of 7-Zip with the proposed preprocessing technique generated highest compression performance for all data sets when it was compared with the use of 7-Zip individually and the use of 7-Zip with WRT. It was the same as WinRAR in which the integration of WinRAR with the proposed preprocessing technique could generate highest compression performance for all data sets when it was compared with the use of WinRAR individually and the use of WinRAR with WRT but in the lower compression performance when it was compared with 7-Zip.

7-Zip archiving software produced the best compression performance for the selected data sets compared to other archiving software in which it might be because of the use of various algorithms to compress the files consisting of its own filters that might be suitable for the prepared data sets, LZ77, LZMA, PPM, and BWT. By comparing 7-Zip with other selected archiving software; the use of additional BWT algorithm that is a block-sorting lossless data compression algorithm which operates by reversibly permutes a block of source data and then rearranges it by using a sorting technique, after that pipes through a Move-To-Front (MTF) stage follows by using of Run Length Encoder and an entropy encoder (Huffman coding or Arithmetic coding) as well as its efficient program-used filter; the compression performance of the selected data sets were improved.

According to the experimental results, 7-Zip is recommended for this kind of supply chain data that is created in spreadsheet format. Table 5.20 represents compression performances of the archiving software when they were used with the proposed preprocessing technique.

Table 5.20: Compression performances of the archiving software when they were integrated with the proposed preprocessing technique

Software	Algorithms used	Compression performance	Range of space saving percentage
7-Zip	Program used filters, LZ77, LZMA, PPM, BWT	High compression performance	92.28% - 95.59%
WinAce	Program used filters, LZ77, Huffman	Moderate compression performance	86.95% - 94.31%
WinRAR	Program used filters, LZ77, PPM, Huffman	Moderate compression performance	88.22% - 94.19%
PeaZip	LZ77, Huffman, LZMA, PPM	Low compression performance	79.58% - 90.34%
WinZip	Shannon-Fano, PPM, Huffman, LZH, LZW	Low compression performance	77.84% - 89.73%

The higher ratio of static data provided more interesting outcomes compared to the higher duplicated static data ratio in which it delivered slightly less significant improvements. However, by increasing of both static data and duplicated static-data ratios, the integration of proposed algorithm with the archiving software could provide better results that benefit to both sender and receiver in term of data compression performance. Figure 5.15 represents the pattern of data that the proposed preprocessing technique can generate results. It is separated into 9 sets of data based on ratios of static data and duplicated static data, and classified into 5 degrees of compression performance improvement when the proposed technique is integrated with archiving software in average by ranging from lowest compression performance improvement, low compression performance improvement, moderate compression performance improvement, high compression performance improvement, and highest compression performance improvement respectively. According to the results, the

proposed preprocessing technique is recommended for the data that contain at least 50% of static data and duplicated static data because it is able to generate significant improvement for the compression performance, while the use of only archiving software individually might be satisfactory for compressing the data that contains less than 50% of static data.

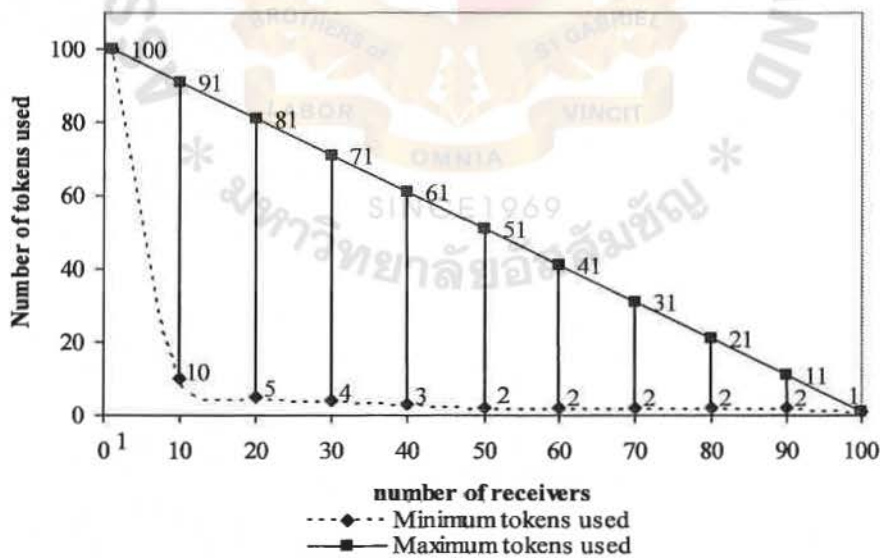
Figure 5.15: Degrees of compression performance improvement in average of the proposed preprocessing technique integrated with archiving software based on ratios of static data and duplicated static data

		% Duplicated static data		
		25	50	75
% Static data	25	Lowest improvement	Low improvement	Low improvement
	50	Low improvement	Moderate improvement	High improvement
	75	Moderate improvement	High improvement	Highest improvement

5.2 Results and Comparisons of the Efficient Use of Encoding Tokens in Relation to Number of Receivers

For the utilization of the encoding tokens, as the relationship of data coding of the proposed technique is a one-to-many in which multiple meanings of decoding overheads can be referred just to a single encoding token, the encoding tokens can be efficiently utilized based on particular groups of original data in relation to distinctiveness of the static data and number of receivers. Figure 5.16 presents minimum and maximum numbers of encoding tokens required for a particular set of data in relation to number of receivers. In this study, 100 data rows were selected in order to find out the ranges of tokens that could be generated for the specific ranges of receivers.

Figure 5.16: Number of tokens used in relation to number of receivers



The results show that encoding tokens can be more effectively utilized if the particular data set contains data for a larger number of receivers. Minimum numbers of required tokens can be reached if the distinctive data rows are balanced and equally

assigned to each receiver, while maximum numbers of tokens represent maximum required tokens that are possibly occurred if the distinctive data rows assigned to each receiver are imbalanced. In the range of 1 to 10 percent of receivers toward distinctive data rows (1 to 10 receivers per 100 distinctive data rows are tested), the numbers of required tokens are significantly dropped if the data set contains equivalent number of distinctive data rows per receiver. After 10 percent of the proportion, the minimum numbers of required token are slightly decreased.



CHAPTER 6

CONCLUSION AND FUTURE WORKS

6.1 Interpretation of Findings and Conclusion

Since text is still one of the most important data that is normally transmitted on day-to-day communication network nowadays, therefore, redundancy of data that frequently transferred and real-time updated many times per day can be problems for enterprises when sharing the data among stakeholders and related parties in which data compression is served as a common tool for supporting file size reduction.

In this research, a simple method for improving data compression performance by using the one-to-many data-sharing concept is introduced used in a novel way. By integrating the purposed preprocessing technique with existing data archiving software, users both sender and receivers can get benefits from the better data compression performance with some additional benefits to data confidentiality.

This research compares the proposed preprocessing technique with another popular preprocessing technique that also performs well for text data called as “Word Replacement Transformation (WRT)” and with the selected archiving software itself in order to discover usefulness of the proposed algorithm and benchmark its compression performances with existing tools. Table 6.1 presents comparisons of WRT and the proposed preprocessing technique in terms of compression performance, data confidentiality, and ease of use.

Table 6.1: Comparisons of WRT with the proposed preprocessing technique

Criterion	WRT	Proposed technique
Supported text format	All text formats are supported.	Only text format in tabular form (spreadsheet) is supported.
Compression performance	Generated better results for some conditions especially when ratios of static data and duplicated static data were low.	Generated better results under some conditions especially when ratios of static data and duplicated static data were high. However, it was depended on the selected archiving tool in which the proposed technique generated better results than WRT for all conditions of prepared data when it was used with 7-Zip and WinRAR.
Confidentiality	Use of ASCII characters but decoding overheads would not be isolated from the processed file.	Use of numerical codes as decoding overheads but the overheads would not be shared in the Cloud. The overheads are separately sent to each assigned receiver with different decoding data for each receiver for confidentiality purpose. Additional encryption can be an option to increase its confidentiality level.
Ease of use	Run EXE file in DOS, Microsoft Windows	Utilize available macro and functions in spreadsheet program itself.

6.2 Discussion: The Most Suitable Approach

This study presents benefits of using the proposed approach and suggests the appropriate archiving software that works best with the proposed technique in order to practically support text file sharing.

Based on the simulation results, it can be concluded that the purposed algorithm is very useful when the text data spreadsheet form has high static data and

uplicated static-data ratios that are directly proportional to the data compression performance.

However, the proposed preprocessing technique might not be able to generate better results when using it with some archiving software under the condition that ratios of static data and duplicated static data are quite low (less than 50% each).

The researcher recommends the use of 7-Zip and WinRAR for integrating with the proposed preprocessing technique because they are able to generate better compression performance compared to WRT in all conditions of the prepared text data even the ratios of static data and duplicated static data are low. However, compression performances of the selected archiving software might be changed from time to time based on continuous development and improvement of each of the archiving software in applying more efficient compression algorithms into the software. Ranking of the compression performance might be changed or there might be another archiving software that can generate better compression ratio in the near future.

The larger number of receivers per shared file can utilize the use of encoding tokens more efficiently. The larger number of receiver can enhance the utilization of the number of assigned encoding tokens used for that particular file because a single encoding token can be used for multiple receivers instead of encoding for just only one receiver in which the minimum numbers of required tokens can be reached when the distinctive data rows are balanced and equally assigned to each receiver.

In addition; instead of using access control, flow control, inference control, and cryptographic control; this proposed preprocessing technique can add some benefits to data confidentiality through the isolation of decoding overheads in which each receiver only sees the decoded data that are only associated with that particular

assigned receiver. Other receivers, hackers, or even eavesdroppers will not know that segregated decoding overheads and are not able to interpret that encoded file into meaningful data without the mapping of shared encoding tokens with the segregated decoding overheads particularly. The higher number of receivers and the higher ratios of static and duplicated static data can lower the chance of data disclosure in the cloud environment because the assigned encoding tokens will be more utilized and lesser dynamic data will be shared in the cloud. In this case, the more valuable static data can be segregated and separately transmitted to each assigned receiver outside the cloud or the sender can increase the confidentiality of data sharing by disconnecting the sending of decoding overheads from the Net and manually send it outside untrusted networks.

Cloud supply chain is selected in this research as an illustration of the proposed preprocessing technique because supply chain is one of the main business functions that have emerged as a critical and integral part of how enterprises operate and compete nowadays. In general, the supply chain data and information are valuable and have to be share to stakeholders while they are also the targets for hackers and other eavesdroppers to attack. It becomes challenges for many organizations to secure their valuable data and information when they have to share them to other parties via untrusted channels or networks. Especially, it is difficult to secure private and confidential data in the cloud since location and access in the cloud are controlled by cloud service providers in which it is different from traditional on-site storage or networks in which the organizations can controls where the data is located and exactly manage who can access it.

The proposed preprocessing technique is actually not limited to cloud supply chain but it can be applied to other data sharing environments as well as other

business functions where the improvement of data compression performance is needed and data segregation is applicable in which ratios of static data and duplicated static data along with number of receivers contained in a tabular file are main factors toward the efficiency of proposed preprocessing technique.

6.3 Future Research

This research focuses only an improvement in compression performance for text data in tabular form created for the purpose of data sharing in which the cloud supply chain is selected as the illustration. The future research may consider about other file formats that can similarly employ this simple technique with some modifications. Other business functions may differently use other file formats to record their valuable data and information in while other algorithms may be more efficiently compress and secure them. In addition, there are also various possible integrated solutions, for examples, by including data encryption or steganography with future techniques in order to get other interesting results.

BIBLIOGRAPHY

- Awan, F. and Mukherjee, A. (2001), LIPT: A Lossless Text Transform to Improve Compression, Proceedings of International Conference on Information and Theory, April 2001.
- Barnatt, C. (2010), A Brief Guide to Cloud Computing, Constable & Robinson Ltd., London, UK.
- Bentley, J. L.; Sleator, D. D.; Tarjan, R. E. and Wei, V. K. (1986), A locally adaptive data compression scheme, Communications of ACM, Vol. 29(4), April, pp. 320–330.
- Burrows, M. and Wheeler, D. J. (1994), A block-sorting lossless data compression algorithm, SRC Research Report 124, Digital Equipment Corporation, California.
- Calabrese, T. (2004), Information Security Intelligence: Cryptographic Principles & Applications, first edition, Thomson Delmar Learning.
- Carus, A. and Mesut A. (2010), Fast Text Compression Using Multiple Static Dictionaries, Information Technology Journal, Vol. 9 No. 5, pp. 1013-1021.
- Chan, H. K.; and Chan, F. T. S. (2010), Comparative Study of Adaptability and Flexibility in Distributed Manufacturing Supply Chains, Decision Support System, Vol. 48, No. 2, pp. 331-341.
- Chang, L. Ti; Chin, L.; Chang, A.Y.; Chun, J. C. (2010), Information Security issue of enterprises adopting the application of cloud computing, IEEE 2010 Sixth International Conference on Networked Computing and Advanced Information Management (NCM), pp. 645.

- Cleary, J. and Witten, I. (1984), Data compression using adaptive coding and partial string matching, IEEE Transactions on Communications, COM-32, pp. 396–402.
- Cormack, G. V. and Horspool, R. N. (1987), Data compression using dynamic Markov modeling, The Computer Journal, Vol. 30(6), December, pp. 541–550.
- Denning, D. E. (1976), A Lattice Model of Secure Information Flow, Communications of ACM, Vol. 19, No. 5 (May 1976), pp. 236-243.
- Denning, D. E.; Denning, P. J.; Schwartz, M. D. (1979), The Tracker: A Threat to Statistical Database Security, ACM Trans. Database Syst., Vol. 4, No. 1 (March 1979), pp. 76-96.
- Deorowicz, S. (2003), Universal lossless data compression algorithms, Doctoral Dissertation, Silesian University of Technology.
- Durowoju, O. A.; Chan, H. K.; Wang, X. (2011), The Impact of Security and Scalability of Cloud Service on Supply Chain Performance, Journal of Electronic Commerce Research, VOL 12, NO 4, pp. 243-256.
- Fano, R.M. (1949), The transmission of information, Technical Report, No. 65 (Cambridge (Mass.), USA: Research Laboratory of Electronics at MIT).
- Hugos, M. and Hulitzky, D. (2010), Business in the Cloud, John Wiley & Sons. Inc., Hoboken, New Jersey, USA.
- Huffman, D. A. (1952), A method for the construction of minimum-redundancy codes, In Proceedings of the Institute of Radio Engineers, September, pp. 1098–1101.
- Huth, A. and Cebula, J. (2011), The Basics of Cloud Computing, Carnegie Mellon University, produced for US-CERT.
- Kattan, A. (2010), Evolutionary Synthesis of Lossless Compression Algorithms: the GPzip Family, Doctoral Dissertation, University of Essex.

- Kok, G. (2010), Cloud Computing & Confidentiality, Master of Science graduation thesis, Computer Science, University of Twente.
- Konecki, M.; Kudelic, R.; Lovrencic, A. (2011): Efficiency of Lossless Data Compression, MIPRO 2011, May 23-27.
- Kruse, H. and Mukherjee, A. (1998), Preprocessing Text to Improve Compression Ratios, In: Storer JA, Proceedings of the 1998 IEEE Data Compression Conference, Los Alamitos, California.
- Lancioni, R.; Schau, H. J.; Smith, M. F. (2003), Internet Impacts on Supply Chain Management, Industrial Marketing Management, Vol. 32, pp. 173-175.
- Lefurgy, P. Bird, I. Chen and T. Mudge (1997), Improving code density using compression techniques, In Proceedings of International Symposium on Microarchitectures (MICRO), pages 194-203, 1997.
- Liao, S.; Devadas, S.; Keutzer, K. (1995), Code density optimization for embedded DSP processors using data compression techniques, In Proceedings of Advanced Research in VLSI, pages 393-399, 1995.
- Mohd Kamir, Y.; Mohd Sufian, M.; Ahmad Faisal Amri, A. and Elissa Nadia, M. (2009), Achieving Capability in Data Compression Using LZW++ Technique, IJCSNS International Journal of Computer Science and Network Security, Vol. 9, No. 8, pp. 327-334.
- National Institute of Standards and Technology (2012), NIST Cloud Computing Program, Available at: [http:// www.nist.gov/index.html](http://www.nist.gov/index.html) accessed on 06/01/2012.
- Rexline, S. J.; Robert, L. (2011): Dictionary Based Preprocessing Methods in Text Compression – A Survey, International Journal of Wisdom Based Computing, Vol. 1(2), pp. 13-18, August.

- Robert, L.; Nadarajan, R. (2009), Simple Lossless Preprocessing Algorithms for Text Compression, IET Software, 2009, Vol. 3, Issue 1, pp. 37-45
- Sangroya, A.; Kumar, S.; Dhok, J.; Varma, V. (2010), Towards Analyzing Data Security Risks in Cloud Computing Environments, ICISTM, CCIS 54, pp. 255-265.
- Shajeemohan, B. S. and Govindan, V. K. (2006), IDBE – An Intelligent Dictionary Based Encoding Algorithm for Text Data Compression for High Speed Data Transmission Over Internet, CoRR abs/cs/0601077.
- Shannon, C.E. (July 1948), A Mathematical Theory of Communication, Bell System Technical Journal, Vol. 27: pp. 379–423.
- Sharma, M. (2010), Compression Using Huffman Coding, IJCSNS International Journal of Computer Science and Network Security, Vol. 10 No. 5, pp. 133-141.
- Shuai, Z.; Shufen, Z.; Xuebin, C.; Xiuzhen, H. (2010), Cloud Computing Research and Development Trend, 2nd International Conference on Future Networks, 2010. ICFN'10. pp. 23.
- Skibinski, P. (2004), Two-Level Dictionary Based Compression, Przesmyckiego, Vol. 20, pp. 51-151 Wroclaw, 30 November 2004.
- Skibinski, P.; Grabowski, Sz.; Deorowicz, S. (2005), Revisiting dictionary-based compression, Software-Practice and Experiences, VOL. 35, No. 15, pp. 1455-1476.
- Sun, D.; Chang, G.; Sun, L.; Wang, X. (2011), Surveying and Analyzing Security, Privacy and Trust Issues in Cloud Computing Environments, Elsevier Ltd., China.

- Suri, P. and Goel, M. (2010), Ternary Tree & A New Huffman Decoding Technique, IJCSNS International Journal of Computer Science and Network Security, Vol 10 No.3, pp. 165-172.
- Ullah, F.; Naveed, M.; Inayatullah Babar, M.; Iqbal, F. (2010): Novel use of Steganography for both Confidentiality and Compression, International Journal of Engineering and Technology, vol. 2 no. 4, pp. 361-366, IACSIT Press Singapore.
- V.K.Govindan and B.S. Shajeemohan (2004), IDBE-An Intelligent Text Data Compression for High Speed data Transmission Over Internet, Proceedings of the International conference on Intelligent Signal Processing and Robotics, ISPR 2004 IIT Allahabad, 20-23, Feb. 2004.
- Willems, F. M. J.; Shtarkov, Yu. M. and Tjalkens, T. J. (1995), The context tree weighting method: Basic properties. IEEE Transactions on Information Theory, Vol. 41, May, pp. 653–664.
- Zhou, L.; Zhu, Y.; Lin, Y.; Bentley, Y. (2012), Cloud Supply Chain: A Conceptual Model, The Mediterranean Institute of Advanced Studies.
- Zia, M. Z. K.; Dewan, Md.; Rahman, F.; Rahman, C. M. (2008), Two-Level Dictionary-Based Text Compression Scheme, Proceedings of 11th International Conference on Computer and Information Technology (ICCIT 2008).
- Ziv, J. and Lempel, A. (1977), A universal algorithm for sequential data compression, IEEE Transactions on Information Theory, IT-23, pp. 337–343.
- Ziv, J. and Lempel, A. (1978), Compression of individual sequences via variable-rate coding, IEEE Transactions on Information Theory, IT-24, pp. 530–536.

