

ROBUST TEXT STEGANOGRAPHY ALGORITHMS FOR SECURE DATA COMMUNICATIONS

Ammar Odeh

Under the Supervision of: Prof. Khaled Elleithy

DISSERTATION

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE

AND ENGINEERING

THE SCHOOL OF ENGINEERING

UNIVERSITY OF BRIDGEPORT


CONNECTICUT

May, 2015

ROBUST TEXT STEGANOGRAPHY ALGORITHMS FOR SECURE DATA COMMUNICATIONS

Approvals

Committee Members

Name	Signature	Date
Dr. Khaled Elleithy		3/6/2015
Dr. Miad Faezipour		03/06/2015
Dr. Eman Abdelfattah		3/6/2015
Dr. Navarun Gupta		3/6/15
Dr. Elif Kongar		3/6/15
Dr. Akram Abuaisheh		3/6/15

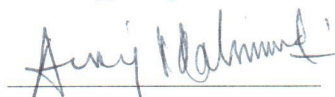
Ph.D. Program Coordinator

Dr. Khaled M. Elleithy

 3/10/2015

Chairman, Computer Science and Engineering Department

Dr. Ausif Mahmood

 3-10-2015

Dean, School of Engineering

Dr. Tarek M. Sobh

 3-10-15

ROBUST TEXT STEGANOGRAPHY ALGORITHMS FOR SECURE DATA COMMUNICATIONS

© Copyright by Ammar Odeh 2015

ROBUST TEXT STEGANOGRAPHY ALGORITHMS FOR SECURE DATA COMMUNICATIONS

ABSTRACT

In the era of information technology, a large amount of digital information is distributed over the Internet in the form of videos, images, text, and audio. The main purpose of widespread distribution enables users to share knowledge between one another. As a result, data transmission and information sharing are effected due to malicious activities. Different techniques exist to protect user authentication, data privacy, and copyrights. Two main techniques that improve security and data protection are Cryptography and Steganography. Cryptographic algorithms convert the secret message from plain text into cipher text. Then the message is sent over the communication channel. Steganography hides a secret message inside the carrier media.

This proposal will investigate different Steganography algorithms and present novel algorithms employing text file as a carrier file. The proposed model hides secret data in the text file by applying various properties into file font format by inserting special symbols in the text file. In addition, the suggested model can be applied in both Unicode and ASCII code languages, regardless of the text file format. The proposed system achieves a high degree of the main Steganography attributes like hidden ratio, robustness, and transparency. In addition, this proposal provides guidance for other researchers in text Steganography algorithms.

ACKNOWLEDGEMENTS

My thanks are wholly devoted to God who has helped me all the way to complete this work successfully. I owe a debt of gratitude to my family for understanding and encouragement.

I would like to express my deep gratitude to Professor Prof. Khaled Elleithy, my research supervisor, for his patient guidance, enthusiastic encouragement and useful critiques of this research work. My grateful thanks are also extended to Dr. Miad Fazipour as co-advisor for her valuable comments.

Furthermore, my acknowledgments are due to Professors Abdelfattah, Abuaishah, Gupta, and Kongar for their positive comments and encouragement. Last but not least, I would like to thank all the staff of the School of Engineering for their support that made my study at the University of Bridgeport a wonderful and excited experience.

TABLE OF CONTENTS

ABSTRACT	iv
ACKNOWLEDGEMENTS.....	v
TABLE OF CONTENTS.....	vi
LIST OF TABLES	viii
LIST OF FIGURES	ix
CHAPTER 1 : INTRODUCTION	1
1.1 CRYPTOGRAPHY	1
1.2 STEGANOGRAPHY	2
1.3 COMBINING CRYPTOGRAPHY WITH STEGANOGRAPHY	3
1.4 RESEARCH PROBLEM AND SCOPE	4
1.5 MOTIVATION BEHIND THE RESEARCH.....	4
1.6 POTENTIAL CONTRIBUTIONS OF THE PROPOSED RESEARCH	5
CHAPTER 2 : LITERATURE SURVEY	7
2.1 SHORT MESSAGE SERVICES (SMS)	7
2.2 TEXT STEGANOGRAPHY ON ONLINE CHAT	8
2.3 TEXT STEGANOGRAPHY BY CHANGING SPELLING OF WORDS	9
2.4 SYNONYM TEXT STEGANOGRAPHY BRITISH AND AMERICAN ENGLISH.....	10
2.5 LINGUISTIC STEGANOGRAPHY OVER CHINESE TEXT	11
2.6 LINGUISTIC STEGANOGRAPHY OVER MALAY TEXT	12
2.7 STEGANOGRAPHY IN TeX DOCUMENTS.....	13
2.8 EMOTICON-BASED TEXT STEGANOGRAPHY IN CHAT.....	13
2.9 AN EVOLUTION OF HINDI TEXT STEGANOGRAPHY	14
2.10 STEGANOGRAPHY USING MATRAYE.....	15
2.11 TELUGU PUNCTUATION MARKS	16
2.12 TELUGU INHERENT VOWEL SHIFTING TECHNIQUE	17
2.13 HTML TAG ATTRIBUTES TECHNIQUE	17
2.14 HTML COMMENT TECHNIQUE.....	18
2.15 HTML TAG LETTERS FORMAT TECHNIQUE	19
2.16 HTML OPEN AND CLOSE TAGS TECHNIQUE.....	19
2.17 HTML TAGS	20
2.18 HTML PROPERTIES	20
2.19 CHINESE LANGUAGE STEGANOGRAPHY USING ARABIC DIACRITICS	20
2.20 SHARP-EDGES METHOD IN ARABIC TEXT STEGANOGRAPHY	21
2.21 AN IMPROVED VERSION OF PERSIAN/ARABIC TEXT STEGANOGRAPHY USING" LA" WORD.....	22
2.22 PSEUDO-SPACE PERSIAN/ARABIC TEXT STEGANOGRAPHY (PSEUDO-SPACE CHARACTER)	22
2.23 PERSIAN/ARABIC UNICODE TEXT STEGANOGRAPHY (LETTER SHAPES).....	23
2.24 ARABIC DIACRITICS BASED STEGANOGRAPHY	24
2.25 HIGH CAPACITY DIACRITICS-BASED METHOD FOR INFORMATION HIDING IN ARABIC TEXT.....	24
2.26 IMPROVED METHOD OF ARABIC TEXT STEGANOGRAPHY USING THE EXTENSION "KASHIDA" CHARACTER	25
2.27 IMPROVING SECURITY AND CAPACITY HIDING OF ARABIC TEXT STEGANOGRAPHY USING 'KASHIDA' EXTENSIONS	25

2.28 EXPLOIT KASHIDA ADDING TO ARABIC E-TEXT FOR HIGH CAPACITY STEGANOGRAPHY	26
2.29 ARABIC/PERSIAN TEXT STEGANOGRAPHY UTILIZING SIMILAR LETTERS WITH DIFFERENT CODES...	26
2.30 TEXT STEGANOGRAPHY IN ARABIC LANGUAGE BY REVERSE FATHA.....	26
2.31 ARABIC TEXT STEGANOGRAPHY USING KASHIDA EXTENSION WITH HUFFMAN CODE	27
2.32 SECRET STEGANOGRAPHY CODE FOR EMBEDDING.....	27
2.33 EXTRA BLANK.....	28
2.34 LINE AND WORD SHIFTING ALGORITHMS.....	29
2.35 SUMMARY	29
CHAPTER 3 : SOFTWARE IMPLEMENTATION	32
3.1 MULTIPOINT ALGORITHM.....	32
3.2 ZERO WIDTH CHARACTER AND KASHIDA ALGORITHM (ZKA)	34
3.3 DIACRITICS ALGORITHM	35
3.4 KASHIDA VARIATION ALGORITHM (KVA).....	37
3.5 ZERO WIDTH CHARACTER ALGORITHM (ZWC).....	39
3.6 REMARKS ALGORITHM.....	40
3.7 STEGANOGRAPHY OVER HTML CODE	43
3.7.1 Experiments and Results.....	45
3.7.2 HTML Code Algorithm Analysis	46
3.8 STEGANOGRAPHY IN TEXT BY USING MS WORD SYMBOLS	47
3.9 PERFORMANCE EVALUATION	50
3.10 MERGE CAPABILITY	52
CHAPTER 4 : HARDWARE SIMULATION.....	58
4.1 MULTIPOINT ALGORITHM [52]	59
4.2 ZKA ALGORITHM.....	60
4.3 ZWC ALGORITHM[53]	61
4.4 DIACRITICS ALGORITHM	62
4.5 KVA ALGORITHM	63
4.6 REMARKS ALGORITHM[54]	65
4.7 ANALYSIS AND SYNTHESIS RESULTS	66
4.8 SUMMARY	68
CHAPTER 5 : CONCLUSIONS.....	69
CHAPTER 6 : REFERENCES	70

LIST OF TABLES

Table 2.1	Short Message Meaning	8
Table 2.2	Sample Word Dictionary between British and American	9
Table 2.3	Synonyms in American and British English	10
Table 2.4	Telugu Language Punctuation marks	16
Table 2.5	Arabic Diacritics	24
Table 2.6	SSCE Articles	26
Table 2.7	Relation between Word Length and Space to Hide Bits	27
Table 2.8	Classification of Hidden Data Inside Text File	30-31
Table 3.1	Relation of Shifting and Distance to Hide bits	32
Table 3.2	Experiment Results of Multipoint Algorithm	32
Table 3.3	Cover Object before and after Insert Secret Bits	33
Table 3.4	Cover Object Effect before and after Insert Secret Bits	34
Table 3.5	Vertical Shifting of Diacritics Algorithm	35
Table 3.6	Cover Object Changes after Apply KVA	37
Table 3.7	KVA Simulation Results	37
Table 3.8	Hidden Ratio for ZWC Algorithm	39
Table 3.9	Position of Secret Bits in Carrier Object	39
Table 3.10	Hidden Ratio be Using Remarks Algorithm	42
Table 3.11	Sample of Hidden Bits by Using Word Symbols	47
Table 3.12	Hidden Capacity of 3 Text Steganography Algorithms	48
Table 3.13	Comparison between Eight Different Algorithms	55
Table 3.14	Output of Merge Process between Proposed Algorithms	56
	merged Simulation Results of A1, A2, A4, A7	54
Table 4.1	Analysis and Synthesis Results of Comprehensive System	66
Table 4.2	Algorithm Throughputs	66

LIST OF FIGURES

Figure 1.1	General Form of Cryptography	2
Figure 1.2	General form of Steganography	3
Figure 1.3	Merge between Cryptography and Steganography	3
Figure 2.1	Exchange Order Interior Letters Order	9
Figure 2.2	Hindi Text Hidden Algorithm	15
Figure 2.3	Sharp-Edges Method in Arabic Text Steganography	21
Figure 2.4	Letter Position Shapes	23
Figure 3.1	Histogram for Multipoint and Single Point Algorithms	33
Figure 3.2	Carrier Files Size with and without Diacritics	36
Figure 3.3	KVA Blocks	37
Figure 3.4	Represent ZWC Blocks	38
Figure 3.5	Steganography Process	42
Figure 3.6	Encryption Gates	43
Figure 3.7	Letter Frequency	45
Figure 3.8	Data Hiding Algorithm	48
Figure 3.9	Hidden Bits Comparison between Three Different Algorithms	49
Figure 3.10	Hidden Ratio [Bit/Kbyte]	50
Figure 3.11	File Size Effect after Inserting Secret Data	51
Figure 3.12	Hidden Ratio Merged with Multipoint	53
Figure 3.13	File Size Change Merged with Multipoint	54
Figure 4.1	Finite “State” Machine Diagram for Multipoint	59
Figure 4.2	RTL View of the Hardware Engine for Multipoint	60
Figure 4.3	RTL View of the Hardware Engine ZKA Algorithm	60
Figure 4.4	Finite “State” Machine Diagram for Diacritics Algorithm [H= Hidden bit, D= diacritic]	62

Figure 4.5	RTL View of the Hardware Engine for Diacritics Algorithm	62
Figure 4.6	Finite “State” Machine Diagram for KVA [H= Hidden bit, C= Connected Letter, NC = NOT C]	63
Figure 4.7	RTL View of the Hardware Engine for KVA	64
Figure 4.8	Finite “State” Machine Diagram for Remarks Algorithm	65
Figure 4.9	RTL View of the Hardware Engine for Remarks Algorithm	65
Figure 4.10	Hiding Data Algorithm’s Timing Simulation Results and “State” Transitions	67

CHAPTER 1 : INTRODUCTION

The massive amount of information is transferred over public channels; some sensitive data and information pass through un-trusted communication channels. Attackers and hackers attempt to break security over the sensitive data, thus exposing the actual information. As a result, various security mechanisms have been applied to improve data integrity and privacy. Security techniques are divided into three categories:

1.1 Cryptography

Cryptography is the method of protecting data by encoding and transferring sensitive information. Cryptography refers to scrambling up sensitive messages so no one can read it. Only intended users can decrypt and read the message [1]. A major disadvantage of Cryptography exists when the sensitive messages are transferred from plain text into a cipher text. Hacker focus on the sensitive data in formation and, thus, may compromise secure for involved parties [2].

Early Cryptography techniques relied upon simple ideas to encrypt data. Today, modern Cryptography techniques use complicated mathematical formulation to encrypt secret messages, such as the public key encryption algorithm Ron Rivest, Adi Shamir and Leonard Adleman (RSA). As an example, Figure 1.1 shows general components of

Cryptography.

Modern science classifies Cryptography into two categories; symmetric key encryption and public key encryption. In symmetric key encryption, the sender and receiver use the same key to decrypt and encrypt data. In public key encryption, two different keys are used; one for encryption and the another for decryption [3]. Early Cryptography techniques depended upon simple ideas to encrypt data.

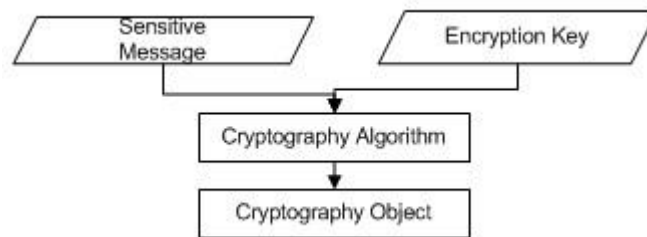


Figure 1.1 General Form of Cryptography

1.2 Steganography

The Steganography method hides sensitive messages inside another carrier file. The object file is sent over a public channel enabling all users to see the carrier file. However, only trusted users can extract the hidden message [4]. Methods that are more complicated were introduced on different digital files. Some Steganography techniques use images to hide data by using image properties, such as image resolution and pixel depth. Other strategies use the audio file frequency to hide data. In addition, the use of video hides data by using both image and audio characteristics. Text files were also, used as a carrier file [5]. Steganography components are defined in Figure 1.2, the output file called Stego Object, which represents the result of merging between the secret message and the Stegano key using specific algorithms to insert sensitive data inside the carrier file.

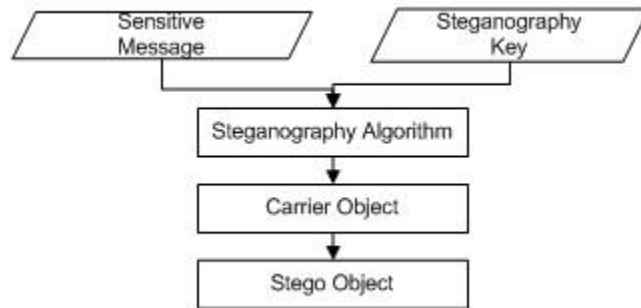


Figure 1.2 General Form of Steganography

1.3 Combining Cryptography with Steganography

By combining the benefits of Cryptographic methods and hiding techniques, novel techniques were introduced to transmit secure data by encrypting the secret message, and then hiding the encrypted secret message inside another carrier file. This combination represents the most secure and intricate system untrusted channels [5, 6].

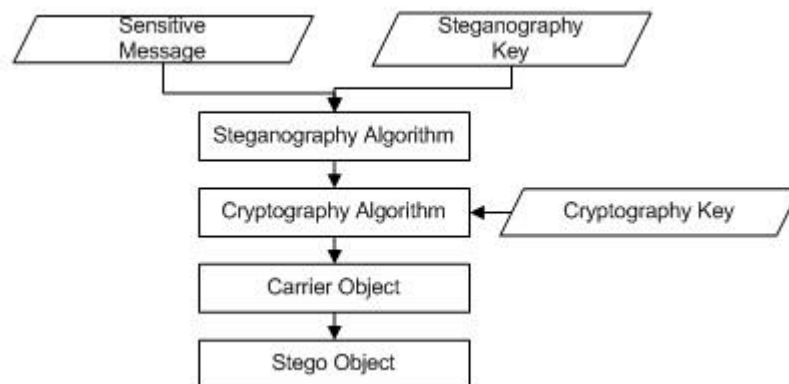


Figure 1.3. Merge Between Cryptography and Steganography

Figure 1.3 represents the main idea of merging between Steganography and Cryptography techniques. Once the secret message is detected, the attacker still needs to solve a Cryptographic problem.

1.4 Research Problem and Scope

Due to the rapid growth of emerging technologies and information broadcasting techniques are introduced to protect sensitive information from plagiarism. A common secure technique is data hiding. Data hiding is classified into three main categories:

- a. Digital watermarking is the process of inserting owner identification inside the carrier file to improve copyright protection,
- b. Fingerprinting is the process of tracing unauthorized copy,
- c. Steganography hosting the secret data inside a carrier object to pass sensitive information.

Various methods of Steganography involve the use of images, audio, and video as carrier files. Some scientific research indicates several methods exist that utilize text files as Steganography carrier media.

This proposal will introduce a new model utilizing text Steganography to improve the reliability of communications between network users. The end result is to hide the secret data in the text file without raising attacker suspicions. The proposed model demonstrates a new hardware text Steganography engine with a high hidden data rate.

1.5 Motivation behind the Research

Significant amounts of information are shared over public Internet channels to enable authorized customers to read and process their sensitive data. For example, medical reports, expression of political opinions and other records are remotely accessible. In the United States, this information is protected under data privacy acts such as The Health Insurance Portability and Accountability Act of 1996 (HIPAA).

This proposal can help government, companies, and organizations protect the privacy of their users from the malicious intent of hackers. In addition, the proposal can be used by authorized users to enhance authentication over the networks.

The novel algorithms presented in this proposal provide the necessary protection for communicating sensitive data. The variation and randomization processes, increase the degree of complexity against attackers. The suggested system prevents infringing copyright material, by hiding sensitive data and keeping it undetectable. Moreover, the implementation of these novel algorithms enables the broadcasting industries, multimedia and publishing to produce products without concern about the attackers' ability to read information.

The implementation of the software and the hardware of this proposal to provide a valuable and vital tool for people to establish safe communication, protect their privacy, securely share data, and protect copyrighted products.

1.6 Potential Contributions of the Proposed Research

Three potential contributions are recognized in this proposal. The first contribution focuses on safe communication. The proposed system enables users to send sensitive information through public channels. The suggested model uses text files as carrier media. Text files are less suspicious; and have the flexibility to change size, various file formats, and font format. These features enhance the hiding data capability. As a result, the possibility for attackers to access the transmitted data is minimized.

The second contribution of this proposal is that it introduces several text Steganography algorithms offering multiple options for users. Some algorithms can be

applied over Unicode language, while others can be used in both Unicode and ASCII code systems. In addition, the proposed algorithms can be applied in the different text file format such as doc and html files.

The third contribution of this proposal is a real time hardware system. This proposal is the first real-time hardware implementation presented in research for Text Steganography. Previous implementations provided efficient hardware implementations over other carriers such as image, video and audio files.

In conclusion, this proposal presents the mathematical and statistical analysis of the presented techniques in terms of transparency, robustness, and data hiding capacity. The final results demonstrate that the presented algorithms outperform the algorithms introduced in previous scientific research.

CHAPTER 2 : LITERATURE SURVEY

This section focuses on Text Steganography algorithms, and discusses 34 different techniques that use text files as the carrier to hide sensitive information. Text files represent the most complicated carrier media; due largely to the relative lack of redundant information, when compared to images and audio files. So, one can classify text Steganography in different categories depending on the hiding purpose:

Protection against detection (Data hiding)

Protection against removal (Document marking)

Watermarking (All objects have same mark)

Fingerprinting (Each object has special mark)

2.1 Short Message Services (SMS)

SMS is a global and common service in messaging systems; especially after the use of cell phones became popular. Via mobile phone, SMS allows users to write and exchange information among each other. [7] introduced one interesting hiding algorithm using Short Message Services. Usually mobile users create some abbreviations for their messages to be easier and simpler; Table 2.1 shows a number of such abbreviations commonly used in SMS. By creating an abbreviation dictionary, the sender and the receiver can send their abbreviations through mobile phones without any attention from

middle attackers. One of the advantages of SMS is popularity. This represents an issue to the Stegoanalyzer in following messages around the world, as well as its ability to update. In other words, the sender and the receiver may use their own abbreviations to hide data, which also disables attackers from performing statistical analysis on messages. Moreover, large amount of hidden data can be passed through the message, which improves the hiding capacity. In addition, this algorithm is language independent. On the other hand, the SMS-based algorithm creates heavy network traffic, where a high volume of messages transferred between a sender and a receiver may attract an intruder's attention. Also, there are some abbreviations that are very clear, for example UNIV which means University. So, anyone who reads it can discern the meaning of the hidden message.

Table 2.1 Short Message Meaning

Acronym	Meaning
ASAP	Immediately
ZZZZ	I'm tired
F2F	Face to face
URW	You are welcome
ILY	I love you
EOL	End of lecture
AYS	Are you serious?

2.2 Text Steganography on Online Chat

Authors presented a novel Steganography method [8]. It is considered an interesting technique to hide data inside text. Online chatting is a very popular application over the internet, especially after social networks and smart phone applications were introduced. All of these factors attracted the authors to employ online chatting as Steganography techniques. Changing the interior orders of adjacent letters was employed

to hide data inside the message. Figure 2.1 represents one of the message transfers between trusted users.



Figure 2.1. Exchange Order Interior Letters Order

Authors presented properties of online chat messages such as sending speed and spelling correction. In addition, the online chatting method can be applied in any language. The main drawback of this strategy is low hiding capacity; the amount of hidden data inside the message in Figure 2.1 is only 1 bit; while the number of characters is 149. In addition, most of the chat editors correct the spelling mistakes or highlight them. The third gap in the online chatting method is that it is completely dependent on one strategy to hide data by swapping the letters. So, if the attacker notes this exchanging of letters, it can easily reveal the hidden data and this breaks Steganography transparency.

2.3 Text Steganography by Changing Spelling of Words

Another method of Steganography was introduced in [9] and was applied in the English language only, between US and UK spellings of words. In this method, authors created a dictionary of corresponding words, as shown in Table 2.2.

Table 2.2. Sample Word Dictionary between British and American

British English	American English
Colour	Color
Flavour	Flavor
Harbour	Harbor
Honour	Honor
Advice	Advise
Licence	License

Switching between two words enables us to hide one bit per word. The main advantage of this algorithm is that it is easy to be applied regardless of which editing

program is used; chatting programs or word editors. Besides being widely used, word editors usually support British English and American English, and this reduces attacker's attention. However, the changing spelling algorithm faces a low hidden data ratio, since each word only passes one bit. Moreover, during the message transfer, if a middle system does not support both format (British and American English), it can lead to breaking system robustness by changing the whole British text to American text or vice versa. The produced text (Stego object) has only one format so the receiver may translate the whole hiding data as zeros and ones. Furthermore, if the attacker follows more than one message, it can note the repeated words which have a different spelling. Therefore, the intruder may suspect hidden data and this breaks the algorithm transparency.

2.4 Synonym Text Steganography British and American English

One interesting method was presented in [9]. The authors have suggested a new synonym scenario between British English and American English by creating a dictionary for some words as shown in Table 2.3. An agreement should be made between trusted parties. For example, Candy in American English represents one corresponding to Sweets in British English to represent zero.

Table 2.3. Synonyms in American and British English

American English	British English
Vacation	Holiday
Fall	Autumn
Movie	Film
Gas	Petrol
Line	Queue

Different advantages were provided by this algorithm such as simplicity, where most words have a synonym, and this supports the success of this algorithm. Moreover,

changing the word form will not motivate hackers to analyze the message. For example, “Call me as soon as you get there” to hide zero and “Ring me (phone me) as soon as you get there” to hide one. In addition, this algorithm is not limited only to electronic documents but also to printed ones and this prevents attackers to removing the hidden data. On the other hand, the synonym method provides a low hiding data rate, where each word passes one bit only. The average number of bits in each word is 63 bits. So, the hidden rate is $1/63$. Besides the hiding data ratio problem, any change in the text may remove the hidden data and this threatens the algorithm robustness.

2.5 Linguistic Steganography over Chinese Text

In [10], the authors utilize a synonym substitution technique in Chinese language. In this technique, the authors reduce the interaction between sounded words without affecting the sentence semantics. The first step of this algorithm is that it creates a substitution set and divides it into three categories: the first one is broad synonyms, the second one is variant synonyms, and the last one is interchangeable synonyms. The system creates a table for each group before it combines the three tables to produce a crude lexical substitution set. Afterwards, the algorithm removes unfamiliar words and keeps common words, which depend on some statistical information. There is the problem of how to do word change without changing the semantics of the sentence. This is possible by using ICTCLAS software, which is developed for that purpose, and also creates a dynamic library. The main advantage of linguistics is the file size, where the interchange process does not change the file size, therefore enhancing the cover file to be imperceptible. Moreover, this method does not depend on the file format, which enhances

robustness, so that data will not change during travel. On the other hand, this algorithm is language-dependent. In other words, it is applied to a Chinese text. In addition, the amount of data that can be hidden in the carrier object is limited, and it depends on the dictionary.

2.6 Linguistic Steganography over Malay Text

In [11], the authors have presented a new linguistic method for the Malay language. This method creates a dictionary containing synonym words. The Malay Linguistic algorithm consists of two main steps: the first one is converting the secret message into binary form. Each character is represented by seven bits depending on ASCII code. The next step is to find out how to substitute words with synonyms. For the substitution comparison purposes, the authors developed the My Stega Link software to hide data by choosing the best word without changing the semantics or syntax of the sentence. This software keeps the file size unchanged, which avoids any suspicion from unintended users. It also maintains the semantics and syntax of the sentence, which improves the system's efficiency. However, the system gets complicated and time consuming during the search for suitable words given that Stegoanalyzer requires time to read and identify hidden messages. Another drawback is the hidden capacity, where each word synonym can pass only one bit. Besides, message repetition with different synonyms may attract unwanted users to read transferred data between the sender and the receiver and analyze it, which leads to discerning the hidden information. In addition, the introduced algorithm only supports the Malay language, which is relatively less common compared to English.

2.7 Steganography in TeX Documents

In [9], the authors choose text format techniques to hide data without changing the meaning, like other methods which uses TeX to hide data. TeX is a type-setting system that merges with METAFONT to create books of high quality and keeps them robust now and in the future. So, to hide data inside a text; the authors first try to find what is called ligatures in the file, then decide which bit $\{0,1\}$ will be passed as secure information. Therefore, there are usually five common ligatures "fi","fl", ff","ffl", ffi". In Computer Modern Roman, we can use `{ }` to separate any joined letters by using the separator command `{ }` and can pass 1; otherwise, pass 0. The main advantage of this scenario is that it is not limited to electronic documents, and it can also be applied to other document types. In addition, the ligatures method will not affect the file size and thus, will avoid attacker suspicions. On the other hand, the low rate of hidden data can be embedded in the carrier file, where the secret data can only be inserted if there is a ligatures letter; otherwise, no data can be inserted. Moreover, the algorithm faces retyping problem, where reprinting text removes the whole hidden data, and this violates robustness. In addition, ligatures techniques are limited to TXT for editing books.

2.8 Emoticon-based Text Steganography in Chat

In [12], a novel text hiding algorithm is introduced by using special chat properties, as chatting rooms have become very popular. The main idea in this research employed emoticons to hide data and tremendous number of those symbols allows for hiding data inside the file. Nowadays, most of chat users use emoticons instead of words, so the first step of this algorithm classifies symbols by semantics and then controls the

symbols order by using a secret key. In this work, the authors have created four sets for emoticons to hide data. After ordering symbols, it will hide data by using the symbol order. For instance, if the symbol is inserted at the beginning of the sentence, it means users pass 0; otherwise, the secret bit is 1. Another procedure used in this work is the symbol order so that system can extract data from the symbol depending on the symbol order number in its set. Therefore, if it is set 4 and order 3, secret data is then 0011, and so on. Many advantages can be acquired from this method, which explains why chatting programs are popular, and most of chatting applications support emoticons. So a huge variety of software can be used to apply emoticon algorithms. Moreover, some chatting programs enable users to create their own customized symbols, in this way increasing the size of the sets, assisting users to hide more secure data and avoid attacker suspicions.

2.9 An Evolution of Hindi Text Steganography

In [13], the authors introduced a novel Hindi text Steganography by using Hindi letters, numbers, and diacritics. In this paper, two scenarios were introduced to hide data. The first one is by using letters and diacritics, as Hindi scripts contain letter and diacritics. If we want to hide data such as “UB,” which consist of two letters and 16 bits, we will represent it by zeroes and ones. The authors suggested adding the encryption algorithm like Data Encryption Standard (DES) to improve security over the proposed algorithm. Then, each one is represented by an Indian diacritic letter, and zero is represented by letter only without diacritics, as shown in Figure 2.2. The main advantage of this algorithm is simplicity. However, the hidden data rate is very low compared to other algorithms. In addition, the algorithm concentrates on the Indian Language, and is

difficult to be applied on the English language. Any further retyping of the message removes the whole hidden data, resulting in eliminating the algorithm robustness. The second Hindi algorithm introduced in [13] is a numerical code converting a secret message into a binary form, and choosing which numerical code fits with vowels and consonants in a four bits binary number. Numerical algorithms have the same properties as the first one, whereas the second one employs some numerical table to hide data. However, the same advantages and disadvantages exist in both methods.

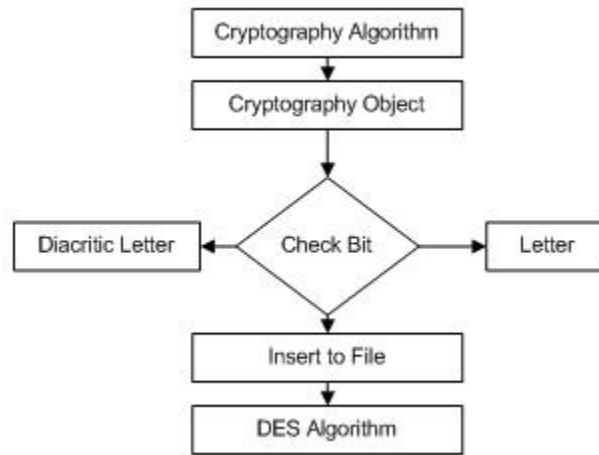


Figure 2.2. Hindi Text Hidden Algorithm

2.10 Steganography Using Matraye

In [14], three scenarios were applied in Hindi. The first method transfers a plain secret message into ASCII code. Then, the authors used an encoding table to convert each bit into top and lower Hindi modifiers. The second method employed an open header, bar, no bar, and special characters in order to hide data inside Hindi. The first step creates tables, classifying the position of the Matraye of Hindi letters, as Hindi letters are classified into top, core, and bottom. Optimal Characters Reorganization (OCR) can easily identify top and bottom letters. The second table is for the encoding scheme. Each

letter has 0 or 1, so, the message can be embedded. The last algorithm introduced in [14] was based on matrix characters to create vowels and numbers relation by matrix. Each letter is represented by a Hexadecimal code and the message is encoded inside it. The main advantage of these algorithms is knowledge. In other words, only those who know the properties of Hindi can classify it. Contra what has previously been stated that the message will be passed without any suspicion, these algorithms can be applied on Hindi, and can be extended to English or other languages, as each language has its own properties. However, the hidden rate ratio is very small, since each symbol has only one bit. Also, the algorithm does not support robustness, since retyping may remove the whole hidden message.

2.11 Telugu Punctuation Marks

[15] introduced two linguistic methods by using Telugu language (spoken language in the state of Andhra and other states). The first method uses one of Telugu characteristics by classifying characters into two groups. Then, it hides secret data in language symbols and passes it inside cover media. The first group to pass was 0 and the other group to hide was 1. The other method was applied by using Telugu language punctuation marks by distributing them into four groups as shown in Table 2.4. Each group was used to pass two bits.

Table 2.4.Telugu Language Punctuation Marks

Symbol	Hidden Bits
/ !	00
○○○ ○○ , ;	01
. :	10

The main advantage of Telugu algorithm is the hidden capacity where each symbol in the first algorithm can pass one bit, which depends on language letters. Moreover, the algorithm's simplicity makes it more applicable for different document formats. The punctuation algorithm enables the user to hide two bits in each symbol, but the symbols limitation reduces the hidden data rate inside it. Moreover, both introduced algorithms are language dependent, which will minimize its application range.

2.12 Telugu Inherent Vowel Shifting Technique

[16] introduced a method to hide data inside Telugu text by horizontally shifting inherent vowel signs. Unlike the English language, the Telugu alphabet is composed of syllables; each of them consists of two main components: the non-core component and the core component. The non-core component can be classified into two categories, the upper noncore, and the lower noncore. So, by horizontally shifting in very low percentage, a hidden bit can be inserted depending on left or right shifting. High hidden ratio represents the main advantage, as each letter may have two non-core symbols enabling it to hide two bits in each letter. On the other hand, retyping can remove the whole hidden data inside the carrier file. Moreover, this algorithm is suitable only for Telugu language and cannot be extended to English or any other language.

2.13 HTML Tag Attributes Technique

HTML or Hyper Text Markup Language is the basic programming language in a web page, and can be mixed with other languages like Macromedia Flash, Java Script for animation goals [17]. Moreover, HTML does not need any special software to program it. Most of new web programming languages are based on HTML concepts. Usually HTML

is used to create the static part of websites. HTML code consists of two parts. The first is a tag which is surrounded by angle parentheses (<>), and the second is the information between tags. Internet browser displays the content without tags, where tags control the appearance of web page content. So, the tag's order represents page organization and design, which is non-case sensitive. In other words, uppercase letters are like lower case letters. Therefore, HTML represents the source code of a webpage. Internet users are concerned only with page content and not the programming page. Based on these hypotheses, most of Steganography algorithms over the WebPages are interested in the coding of page and not the page information. Authors classify HTML code into two categories, primary attributes and secondary attributes. If secondary attributes are followed by primary attributes, then a bit 0 is detected, otherwise a 1 is detected. The authors suggested two steps to pass sensitive data. The first step is an encrypted process to scramble message content; the second step is applying HTML Steganography scenario in order to hide bits [18].

2.14 HTML Comment Technique

Usually HTML files support inclusion of comments by adding “<!--” at the beginning of the comments and “-->” at the end of the comments [19]. Moreover, comments do not appear in the internet browser. So, the Internet users will not be notified by any changes in website appearance. By applying this scenario, huge amount of data can be hidden inside webpage files without any notice. In addition, comments can be added in any location inside the file. The main advantage of this method is that huge amount of data can be inserted in the carrier file, and that the hidden data can be inserted

anywhere in the HTML document. The hidden data is usually readable, but average Internet users rarely explore the page code. Only programmers are concerned about comments to understand programming methods. Most of the comments algorithms provide the same advantages and disadvantages of primary and secondary tags [20].

2.15 HTML Tag Letters Format Technique

Another HTML Steganography algorithm was presented by using one of the characteristics of the HTML file. This has been possible by changing tag's letter case to hide data 1 for uppercase, and 0 for lowercase. There is no difference between upper and lower case letters in HTML coding. The advantages of this method are similar to the other scenarios where the hidden data does not appear in the web browser. Moreover, huge amount of data can be hidden inside HTML files, achieving a high percentage of hidden data. However, reprinting the carrier file will remove the whole hidden data, leading to the violation of robustness [21] [22].

2.16 HTML Open and Close Tags technique

Another HTML algorithm was proposed by using HTML tags, which employs some varying combinations or gaps to hide data [22].

Example

 to hide 0

 to hide 1

By using this method, each tag can pass one bit by adding “\” to the end of tag.

The main point of this method is non-suspicion, as “\” is usually used in most of the HTML tags without any change in the output file.

2.17 HTML Tags

Usually HTML files start with an `<html >` tag and end with an `</html>` tag [23]. The simplest method to hide data inside HTML is to insert the hidden data after the closed HTML tag `</html>`. The secret data will not appear in the browser output. But the whole hidden data can be read from the end of the HTML code file. The main problem with this method is the degree of simplicity. If someone explores the source code, the whole hidden data can be read. However, a huge amount of secret data can be hidden inside an HTML file.

2.18 HTML Properties

Authors use one of HTML properties In [23]; the ID attribute of each tag on an HTML page. Each ID tag consists of three parts: the object name, the title of HTML page, and the four coded characters. The authors discussed the main advantages and disadvantages of the ID attribute algorithm. The main advantage of the suggested algorithm is the massive number of HTML files over the Internet. Moreover, the ID attribute is a common way used to compress HTML files. Also, this method can be applied over similar web coded languages like XML, and ASP. However the fact that this technique can only be applied over HTML represents the main drawback.

2.19 Chinese Language Steganography using Arabic Diacritics

Another algorithm, introduced in [24], suggested to combine among three languages: Chinese, Arabic, and English. At the beginning, the authors created two tables; the first one was for Arabic diacritics and the second table was to store English

letters. The Chinese text was translated into English text, and then each English letter was converted to two Arabic diacritics. The system then creates Arabic text which contains selected diacritics. High data hiding rate was introduced in the three languages algorithm. Diacritics are used over each Arabic character to hide one bit in each letter. However, Arabic diacritics are rarely used today. Usage of these diacritics may attract unintended users to read and analyze the carrier file. Moreover, any reprinting process for the whole file without diacritics will remove the whole hidden secret data and this threatens the algorithm's robustness. The complexity of the suggested scenario represents an advantage to defeat against attackers. On the other hand, the algorithm may produce some ambiguity in translation from Chinese language to English language.

2.20 Sharp-Edges Method in Arabic Text Steganography

Author introduced a new method to hide bits inside an Arabic text by using one of Arabic characteristics [25]. That is done by using Sharpe edges and classify letters into four categories depends on edges.

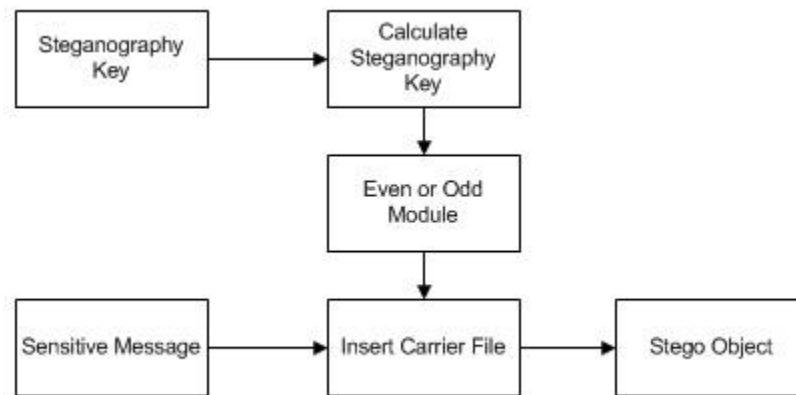


Figure 2.3. Sharp-Edges Method in Arabic Text Steganography

Figure 2.3 shows steps to insert the secret data by employing edges properties.

The authors reported that using this method, huge amount of data can be hidden inside an Arabic file. Moreover, a high degree of invisibility is supported by this scenario. In addition, the Arabic language supports a wide range of font formats. So, Stegoanalyzer cannot distinguish between the different files. However, re-editing of the Stego-object removes the hidden data.

2.21 An Improved Version of Persian/Arabic Text Steganography Using "La" Word

[26] introduced another method to hide data by using Arabic/Persian language properties, where the letters shape depends on the letters position in the word. "La" has two shapes “ﻻ” or ﻻﺀ” and they have the same meaning. The main advantage of this method is that there is no change on the file semantics, and everyone can read the file and process it. On the other hand, the “La” algorithm suffers from different problems: the first one is low hiding data rate, where the secret bit is inserted in the text if the text contains “La” character, otherwise, nothing is inserted. Moreover, reprinting the message deletes the whole secret data. Other problems that the “La” algorithm faces are that the file size will be increased and the letters may have abnormal shapes.

2.22 Pseudo-Space Persian/Arabic Text Steganography (pseudo-space character)

In [27], the authors used one of Persian/Arabic text characteristics, which is called pseudo-space character (200C). The suggested system can add this code to hide 1, otherwise 0. Pseudo-Space algorithm has many advantages. The suggested algorithm can

be applied in different file extensions like HTML and document files. Also, the copy and paste operation does not affect on the hidden bits. In addition, the carrier file size and format does not change. Moreover, the suggested algorithm can be used in different Steganography systems which provide Unicode system. Furthermore, the algorithm can be applied in other languages like Pashto (the official language of Afghanistan) and Urdu (the official language of Pakistan). However, retyping of text can remove the hidden data. It is important to mention that the repetition of a unique character like pseudo-space may enable the attacker to analyze the carrier file and read the hidden data.

2.23 Persian/Arabic Unicode Text Steganography (Letter Shapes)

[27] introduced a Steganography algorithm that use interesting features of Persian and Arabic languages. Most of the letters have four different shapes depending on their positions in the word, as can be seen in Figure 2.4 .So, each word in the text file can be saved in two ways: the first one is by saving representative letters and second is by saving the word by saving the code of the correct shape of each letter. The authors suggest hiding 0 in the first case and one in second cases. The main advantages of this method are that the hidden rate capacity does not change the size of the text and it can be applied on different file formats like HTML documents. The text format will also not be changed.



Figure 2.4. Letter Position Shapes

2.24 Arabic Diacritics based Steganography

In [28], the authors presented another text Steganography technique for the Arabic language, by using one of Arabic characteristics; diacritics (As shown in Table 5). Diacritics represent vowel sounds in Arabic language. Nowadays, diacritics are rarely used. On the other hand, any religious documents must have it. Arabic language has nine diacritics; the most frequently one is Fatha. Therefore, the proposed algorithm uses Fatha to represent one and zero can be represented by any of the remaining eight diacritics. If the first hidden bit is one, then diacritics system finds the first Fatha and removes any other diacritics before it. One of the advantages of this method is its reusability, as the same cover can be used for more than one hidden message. Moreover, this technique does not need complex software. However, currently diacritics are rarely used; the existence of diacritics in normal text may tempt the attacker to analyze the carrier file and compare it with the original one.

2.25 High Capacity Diacritics-based Method for Information

Hiding in Arabic Text

Another promising technique was introduced in [29], where it also used the diacritics; one of the optional characters in Arabic language. If the Steganography system wants to pass one, the system keeps the diacritic character, otherwise removes it to pass

zero. In contrast, diacritics existence represents a hidden bit, and this enables users to employ the same carrier text more than once. On the other hand, any comparison between two consecutive sending packets leads to the exclusion of hidden data.

Table 2.5. Arabic Diacritics

Diacritic	Diacritic Shape
Tanween Dammah	◌َ
Dammah	◌ِ
Tanween Fateh	◌ُ
Sokon	◌ْ
Fateh	◌̣

2.26 Improved Method of Arabic Text Steganography using the Extension “Kashida” Character

In Arabic, the Kashida character represents an extension letter which does not affect the meaning of the word. Usually, it is used to justify the text. In [30], the authors suggested to add one Kashida to hide zero and two consecutive Kashida when the hidden bit is one.

2.27 Improving Security and Capacity Hiding of Arabic Text

Steganography Using 'Kashida' Extensions

In [31], the authors introduced another text Steganography method to hide the sensitive data inside Arabic text. The main idea in the algorithm depends on one special merit of the Arabic language, the Kashida character. The authors discussed a maximum number of Kashida letters that can be added to the word. Moreover, three scenarios were tested to use Kashida character. The authors evaluated the number of hidden bits that can be embedded in the carrier file and compared the results with diacritics, and Kashida methods.

2.28 Exploit Kashida Adding to Arabic e-Text for High Capacity

Steganography

In [32], the authors introduced a text Steganography approach to maximize Kashida to hide data inside e-texts by using Maximizing Steganography Capacity Using Kashida in Arabic Text (MSCUKAT). The first part of this algorithm specified which Arabic letter can be extended by using Kashida either before or after that letter, but not at the beginning of the word. Furthermore, one must note that not all letters can be extended.

2.29 Arabic/Persian Text Steganography Utilizing Similar Letters with Different Codes

Another text Steganography method was introduced in [33] by merging Arabic and Persian languages using this characteristic that these languages include some letters having the same pronunciation but with a little different shape. In this algorithm, the authors are only concerned with two letters format to hide data. For example, to hide one the system can use Arabic letter, and Persian to hide zero.

2.30 Text Steganography in Arabic Language by Reverse Fatha

Authors in [34] presented a novel Steganography method was applied in Arabic language by using Fatha and its inverse. In other words, the Steganography system uses regular Fatha to pass one; otherwise, zero is passed. By installing new font properties that can accept inverse Fatha, this task can be facilitated. In this technique, reading hidden file bits to format file depends on hidden bits. Reverse Fatha performs a high hidden bit ratio,

and it is not a noticeable direction if it changes. However, the main problem is that if the carrier file is reused, the attackers may notice the change between them. Moreover, retyping can reduce the algorithm robustness.

2.31 Arabic Text Steganography Using Kashida Extension with Huffman Code

Authors employed the Huffman code by using Kashida Steganography to hide a huge amount of secret data inside text file [35]. Kashida inserted after connected letters is used to pass one as a secret bit, and its absence is used to pass zero. Authors suggested to compress Stego file by using Huffman code to minimize file size and avoid attacker suspicions. Moreover, the proposed algorithm can be applied to other languages like Urdu and Persian. Also, the algorithm can be simulated to different document formats such as: *.doc, *.txt, and *.html.

2.32 Secret Steganography Code for Embedding

In this technique, the authors suggested a special code called Secret Steganography Code for Embedding (SSCE) to embed the secret data inside text file. This is done by selecting the message and then converting it into ASCII code. The ASCII code is converted into SSCE code by using a conversion table. After that, the produced code is translated into characters. Then, the encrypted message can be embedded into the carrier file. This embedding process depends on Table 2.6 to insert the secret data inside the carrier file [36].

For example, if the encrypted hidden message is (10), it would find out the word (an) from the carrier file and the next character is a vowel. The same scenario can be

applied to the remaining document. Hiding bit ratio represents the main weak point of the SSCE algorithm, as one bit can be embedded if the carrier document contains articles, otherwise secret data cannot be passed. Moreover, the proposed system needs to use a text generator to hide the intended data.

Table 2.6. SSCE Articles

Articles	First letter of Word	Hidden data
A	Consonant	00
A	Vowel	10
An	Consonant	01
An	Vowel	11

2.33 Extra Blank

Another algorithm introduced was in [22], where the authors suggested hiding the secret data inside the carrier file by inserting an extra blank after words. The first step in null space algorithm is encrypting the secret message. The first bit from the encrypted secret message is ready to start the inserting process into the carrier file. If the hidden bits are 00, then the embedded function starts to search for even word lengths and adds two blank spaces after it. The procedure will be continuously repeated based on Table 2.7.

Table 2.7. Relation between Word Length and Space to Hide Bits

Word length	Number of space	Hidden bits
Even	1	01
Even	2	00
Odd	1	10
Odd	2	11

A huge amount of data can be inserted by using the null space algorithm. However, there are some text editors that highlight double space and convert it to single space. In addition, the algorithm can affect the transparency, since Stegoanalyzer can follow the number of blank spaces and extract data line and word shifting.

2.34 Line and Word Shifting Algorithms

Different papers discussed the ability of vertical shifting for lines or for words. Shifting by an amount of 1/300 inches is non-notable for attackers. Simplicity is the main advantage of shifting algorithms but the hidden data ratio is very low compared to other algorithms [37].

2.35 Summary

Table 2.8 represents a comprehensive comparisons and classification for various hidden data files on Text Steganography algorithms. Furthermore, Table 2.8 categorizes each algorithm and its applied technique. More importantly, Table 2.8 highlights the algorithm's satisfaction of Steganography criteria that includes robustness, hidden capacity and transparency.

Table 22.8. Classification of Hidden Data inside Text File

		General Categories			Text Methods			Criteria		
	Method	Substitution	Injection	Propagation	Format	Random and statistical generation	Linguistic	Hiding Capacity	Transparency	Robustness
1	Short Message services (SMS)	√					√	High	0.3-1.00	Low
2	Text Steganography on Online Chat	√					√	0.54%	0.2-1.00	Low
3	Text Steganography by Changing Words Spelling	√					√	Low	0.1-1.00	Low
4	Synonym Text Steganography over British and American English	√					√	Low	0.5-1.00	Low
5	Linguistic Steganography over Chinese text	√					√	3.59%	0.3-1.00	High
6	Linguistic Steganography over Malay text	√					√	Low	0.2-1.00	High
7	Steganography in TeX Documents		√		√			(0.5-1.51)%	0.1-1.00	Low
8	Emoticon-based Text Steganography in Chat	√				√		High	0.5-1.00	Low
9	An Evolution of Hindi Text Steganography	√			√			High	0.3-1.00	Low
10	Steganography Using Marty		√		√			High	0.2-1.00	Low
11	A New Approach to Telugu Text Steganography	√			√			High	0.1-1.00	Low
12	A new approach to Telugu text Steganography by shifting inherent vowel	√			√			Low	0.2-1.00	Low
13	A novel text Steganography presented by using html files		√			√		Low	0.1-1.00	Low
14	HTML comments		√					High	0.5-1.00	Low
15	of HTML characteristics by change tags letter case to hide data		√		√			High	0.6-1.00	Low
16	HTML Open and Close Tags technique		√		√			High	0.6-1.00	Low
17	Other HTML algorithm proposed by using HTML tags		√		√			High	0.6-1.00	Low
18	HTML page is the ID attribute	√					√	High	0.5-1.00	High
19	Chinese Language Steganography using the Arabic Diacritics as a Covered Media		√		√			High	0.5-1.00	Low
20	Sharp-Edges Method in Arabic Text Steganography	√			√			Low	0.1-1.00	Low
21	An Improved Version of Persian/Arabic Text Steganography Using "La" Word		√		√			Low	0.1-1.00	Low
22	Pseudo-Space Persian/Arabic Text Steganography		√		√			(3.7-4.66)%	0.1-1.00	Low
23	Persian/Arabic Unicode Text Steganography	√			√			(89-109)%	0.5-1.00	Low
24	Arabic diacritics based Steganography		√		√			0.0327%	0.5-1.00	Low
25	High Capacity Diacritics-based Method For Information Hiding in Arabic Text	√			√			6.79%	0.2-1.00	Low
26	Improved Method of Arabic Text Steganography is using the Extension		√		√			High	0.3-1.00	Low

	"Kashida" Character.									
27	Improving Security and Capacity for Arabic Text Steganography Using 'Kashida' Extensions.		√		√			1.22%	0.2-1.00	Low
28	Exploit Kashida Adding to Arabic e-Text for High Capacity Steganography		√		√			39.5%	0.1-1.00	Low
29	Arabic/Persian Text Steganography Utilizing similar Letters with Different Codes	√			√			(31-36)%	0.5-1.00	Low
30	A Novel Text Steganography Technique to Arabic Language Using Reverse Fatha	√			√			High	0.3-1.00	High
31	Arabic Text Steganography Using Kashida Extension With Huffman Code		√		√			3.01%	0.2-1.00	High
32	SSCE		√		√			Low	0.1-1.00	Low
33	Extra blank		√		√			One Bit per space	0.7-1.00	Low
34	Line and word shifting		√		√			One Bit per line/word	0.7-1.00	Low

CHAPTER 3 : SOFTWARE IMPLEMENTATION

This section, presents eight new text Steganography algorithms: (1) Multipoint, (2) ZKA, (3) Diacritics, (4) KVA, (5) ZWC, and (6) Remarks (7) HTML code (8) MS Word symbols. The proposed algorithms enhance the hidden capacity ratio including the system's transparency and robustness. The diversity of algorithms enables the users to pass their sensitive data in authenticated and secure manner. Each algorithm has a different strategy that allows the user to hide data inside the text file; some of these algorithms employ the text's font format to embed data such as Multipoint Algorithm. Other algorithm use symbol insertion technique to pass sensitive data over public channel. In addition, the presented algorithms can be classified into two categories (1) Unicode languages and (2) Multiple languages.

3.1 Multipoint Algorithm

Multipoint algorithm is presented and applied in Unicode language (Arabic, Persian). These languages have some letters called multipoint shaped letters, which contain more than one point, unlike the English shaped letters (i, j). Vertically or horizontally shifting techniques can be employed to hide two bits of information in each character. The proposed algorithm offers a highly hidden capacity ratio compared to other algorithms. Each multipoint letter can hide 2 bits, whereas, other algorithms can hide only one bit per letter. The proposed algorithm enhances robustness by changing Stego-object to an Image or a PDF file in order to avoid the

retype problem. In addition, the introduced algorithm can be applied into other languages such as Pashto and Urdu [38]. Table 3.1 illustrates relation between letter effect and hidden code.

Table 3.1. Relation of Shifting and Distance to hide Bits

<i>Point Shift</i>	<i>Distance between points</i>	<i>Hidden code</i>	<i>Letter Format effect</i>
0	0	00	No Change
0	1	01	Distance between points
1	0	10	Vertical shifting
1	1	11	Distance and shifting

Multipoint Algorithm Pseudo Code

1. Enter the text and hidden file and its size
2. Search for multipoint letters
3. Hide size of embedded data at the beginning
4. For I= start to EOF

IF hidden data ="00" then call No change ();

Else if hidden data= "01" then call distance ();

Else if hidden data ="10" then call shifting ();

Else call distance-shifting ();

End for

5. Convert file to image file and send to other side

6. End

Table 3.2 illustrates the experimental results of the Multipoint algorithm. As shown in Figure 3.1, the Multipoint algorithm merges with a Single point algorithm to enhance the hidden data ratio capacity.

Table 3.2. Experiment Results of Multipoint Algorithm

<i>#</i>	<i>Website</i>	<i>Page Size</i>	<i>The character # 2 point or more</i>	<i>Capacity Ratio (Bit/ Kilobyte)</i>
1	aljazeera.net	23.8 KB	1245	105
2	daralhayat.com	15.4 KB	968	126
3	salahws.com	10.3 KB	535	104
4	holyquran.net/tadabur	13.8 KB	516	75
5	khayma.com	21.8 KB	499	46

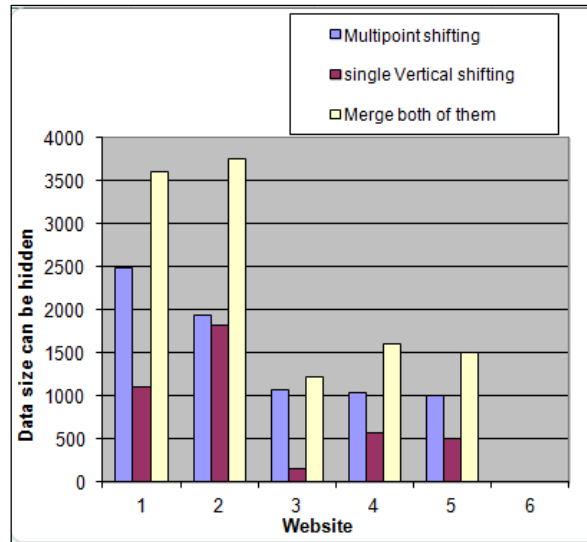


Figure 3.1. Histogram for Multipoint and Single Point Algorithms

3.2 Zero Width Character and Kashida Algorithm (ZKA)

Zero Width character and Kashida are introduced through the Unicode language (Arabic) by using special characters to justify sentences; Kashida and Zero Width character hide two bits between connected letters. In order to avoid any attackers' suspicion, a randomization algorithm utilizing ZKA has been applied. By using this process, each message applied different strategies to conceal secret messages. The proposed algorithm can be extended to other Unicode languages. Two weaknesses of ZKA are: (1) the retyping problem, and (2) the clear format problem. These deficiencies reduce the algorithm robustness [39]. Table 3.3 illustrates cover object before and after inserting secret bits.

Table 3.3. Cover object before and after insert secret bits

Cover Object	كان ساحل مصر الشمالي سلّة غذاء مصر والامبراطورية الرومانية التي كانت تحتل مصر قبل الإسلام. فقبل بناء السد العالي، اعتمد المصريون على مياه النيل في الزراعات الصيفية في الوادي والدلتا كما اعتمدوا على مياه الأمطار في زراعة القمح
Stego Object	كان ساحل مصر الشمالي سلّة غذاء مصر والامبراطورية الرومانية التي كانت تحتل مصر قبل الإسلام. فقبل بناء السد العالي، اعتمد المصريون على مياه النيل في الزراعات الصيفية في الوادي والدلتا كما اعتمدوا على مياه الأمطار في زراعة القمح
Secret Bits	100100100000111010110001011110110001111110011101111100111110100000100101110 001100000001011010001111111000000101001

3.3 Diacritics Algorithm

Diacritics Algorithm is represented in [40] to hide data inside Arabic text. This proposed algorithm uses Arabic diacritics which are language characteristics represented by small vowel letters. These diacritics are an optional property for any Arabic text, and are not popularly used. Most of the Arabic letters need diacritics to correct word pronunciation. The proposed algorithm employs diacritics to hide two secret bits from each diacritic, in order to offer a high hidden ratio by applying this algorithm compared to other algorithms. The main drawback of the diacritics algorithm is that the use of diacritics is uncommon. Thus, hackers might be more intrigued to analyze the file. Table 3.4 shows cover object effect after and before insert the secret data.

Table 3.4. Cover object effect before and after insert secret bits

Cover Object	كَانَ سَاحِلُ مِصْرَ الشَّامِلِي سَلَّةَ غِذَاءٍ مِصْرَ وَالْإِمْبَرَاطُورِيَّةِ الرُّومَانِيَّةِ الَّتِي كَانَتْ تَحْتَ مِصْرَ قَبْلَ الْإِسْلَامِ . فَقَبِلَ بِنَاءُ السِّدِّ الْعَالِي ، اعْتَمَدَ الْمِصْرِيُّونَ عَلَى مِيَاهِ النَّيْلِ فِي الزَّرَاعَاتِ الصَّيْفِيَّةِ فِي الْوَادِي وَالْإِلْتِنَا كَمَا اعْتَمَدُوا عَلَى مِيَاهِ الْأَمْطَارِ فِي زِرَاعَةِ الْقَمْحِ
Stego Object	كَانَ سَاحِلُ مِصْرَ الشَّامِلِي سَلَّةَ غِذَاءٍ مِصْرَ وَالْإِمْبَرَاطُورِيَّةِ الرُّومَانِيَّةِ الَّتِي كَانَتْ تَحْتَ مِصْرَ قَبْلَ الْإِسْلَامِ . فَقَبِلَ بِنَاءُ السِّدِّ الْعَالِي ، اعْتَمَدَ الْمِصْرِيُّونَ عَلَى مِيَاهِ النَّيْلِ فِي الزَّرَاعَاتِ الصَّيْفِيَّةِ فِي الْوَادِي وَالْإِلْتِنَا كَمَا اعْتَمَدُوا عَلَى مِيَاهِ الْأَمْطَارِ فِي زِرَاعَةِ الْقَمْحِ
Secret Bits	1001001000001110101100010111101100011111110011101111100111110100000100101110 001100000001011010001111111000000101001

Table 3.5. Vertical Shifting of Diacritics Algorithm

#	Website	Number of letter	Number of letter without Diacritic	Number of Diacritic	Size without Diacritic	Size with Diacritic	change in size	Size Ratio	Capacity Ratio (Bit/Kilobyte)
1	http://mubashermisr.aljazeera.net/	6138	3631	2507	16.6	17.19	0.59	3.55%	145.84
2	http://www.alfikralarabi.org/	6492	3854	2638	17.4	17.9	0.5	2.87%	147.37
3	http://mentouri.ibda3.org/	5651	3363	2288	16.6	17.6	1	6.02%	130.00
4	http://alamatonline.net/	7342	4352	2990	17.9	19.1	1.2	6.70%	156.54
5	http://www.sawaleif.com	2382	1479	903	14.8	15.3	0.5	3.38%	59.02
6	http://www.ahram.org.eg/	4272	2444	1828	15.7	16.5	0.8	5.10%	110.79
7	http://www.alquds.co.uk/	1661	1006	655	14.3	14.8	0.5	3.50%	44.26
8	http://www.alriyadh.com/	6512	3951	2561	17.5	18.4	0.9	5.14%	139.18

Table 3.5 demonstrates numerical values of the carrier file characteristic as applied through the Diacritics algorithm. Figure 3.2 as shown below offers a comparison of file size with and without diacritics.

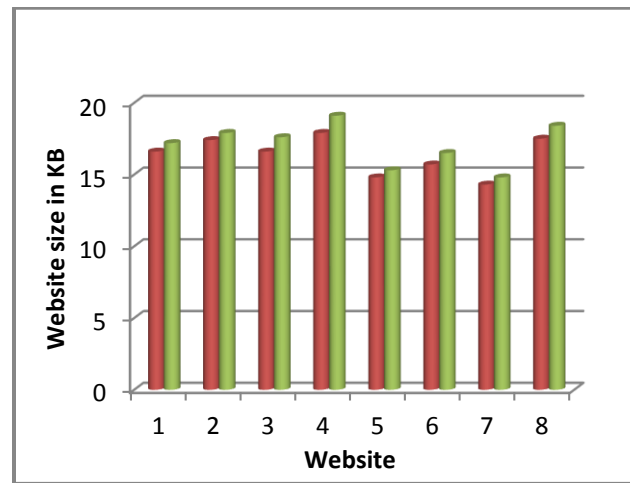


Figure 3.2. Carrier Files Size With and Without Diacritics

3.4 Kashida Variation Algorithm (KVA)

Kashida variation algorithm (KVA) is presented in Kashida according to the space position in order to hide two bits between words. KVA is a simple application that, typically, uses justified text over any word editors. The proposed KVA introduces four different Kashida information hidden scenarios. A specific scenario applies in each fragment as shown in Figure 3.3. Furthermore, an aggregation is applied over message blocks to reassemble the message that contains hidden information. The benefits of the variation process within KVA creates an extra complex dimension, enhance robustness, and transparency [41]. Table 3.6 illustrates the cover file effect after embedded secret data. According to the Table 3.6, the semantic of the carrier file did not change.

Table 3.6. Cover Object Changes After Apply KVA

Cover Object	كان ساحل مصر الشمالي سلة غذاء مصر والامبراطورية الرومانية التي كانت تحتل مصر قبل الإسلام. فقيل بناء السد العالي، اعتمد المصريون على مياه النيل في الزراعات الصيفية في الوادي والدلتا كما اعتمدوا على مياه الأمطار في زراعة القمح
Stego Object	كان ساحل مصر الشمالي سلة غذاء مصر والامبراطورية الرومانية التي كانت تحتل مصر قبل الإسلام. فقيل بناء السد العالي، اعتمد المصريون على مياه النيل في الزراعات الصيفية في الوادي والدلتا كما اعتمدوا على مياه الأمطار في زراعة القمح
Secret Bits	100100100000111010110001011110110001111111001110111110011111010000010010111000110000000101101000111111000000101001

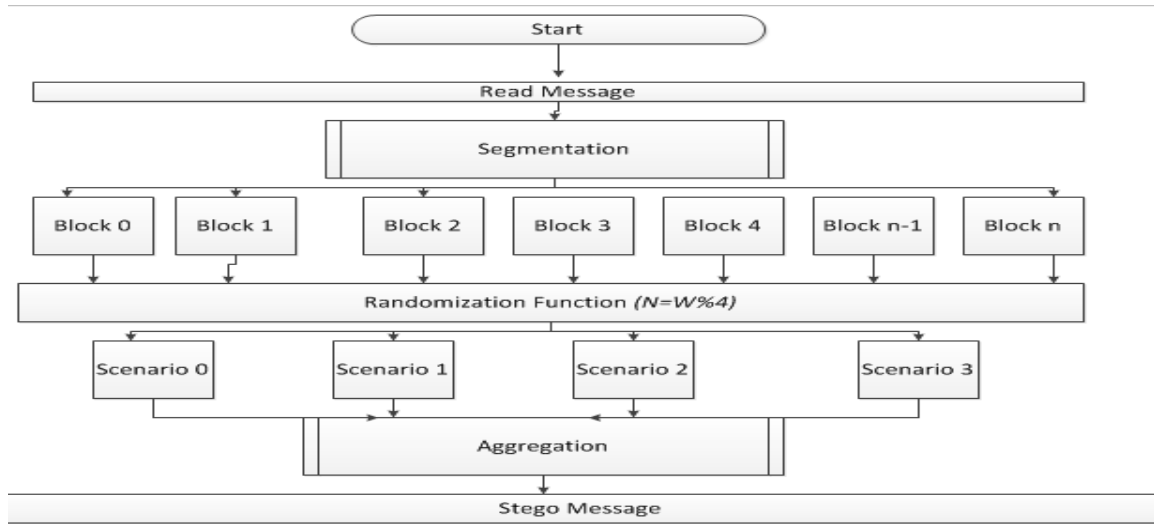


Figure 3.3. KVA Blocks

Table 3.7. KVA Simulation Results

#	Website	KVA	Line Shift	Word shift
1	www.aljazeera.net	34.5	0.20	2.87
2	www.bbc.co.uk	30.8	0.21	2.06
3	www.ahram.org.eg	35.5	0.23	2.96
4	www.addustour.com	41.2	0.32	2.75

Table 3.7 presents the simulation results using KVA, line shift, and word shift algorithms. Based on the simulation, KVA's hidden data ratio is comparatively higher than the line shift and word shift algorithms.

3.5 Zero Width Character Algorithm (ZWC)

Zero Width Character algorithm is presented in[42] by merging a space and Zero Width Character to hide two bits between words inside any document. The main idea of ZWC is variation between two letters in order to hide data as shown in Figure 3.4. By hiding the data in the file, the time complexity equation will be $= M + (N/2)$ where M represents the number of carrier files, and N represents the number of hidden bits.

An advantage of ZWC is its ability to be applied to any language. A disadvantage of ZWC is its transparency, which is created by adding an extra space that can be checked by most of word editors' programs. In addition, retyping can destroys robustness of the algorithm.

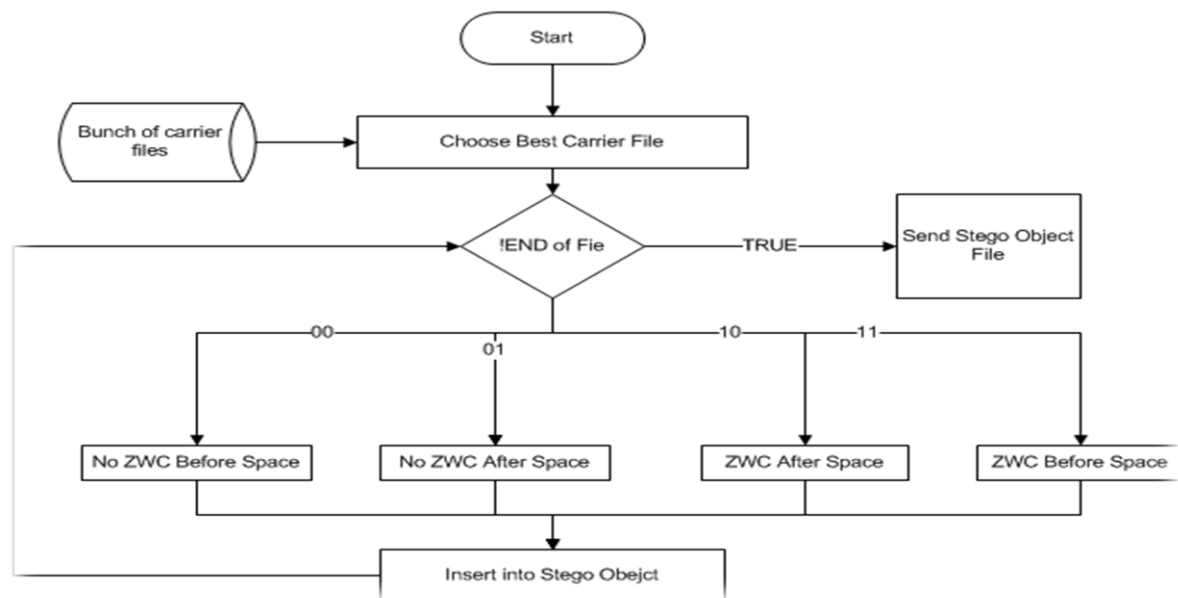


Figure 3.4. Represent ZWC Blocks

Table 3.8. Hidden Ratio for ZWC Algorithm

#	Website	Original size	Number of secret bits	Worst case	Average	Best Case	Space Ratio
1	www.bbc.co.uk	13,348	3798	17146	15247	13,348	28.45
2	http://www.cnn.com	16,722	2340	19062	17892	16,722	13.99
3	http://www.nytimes.com	14,954	2286	17240	16097	14,954	15.29
4	http://education.astate.edu	40,960	6138	47098	44029	40,960	14.99
5	http://www.post-gazette.com	15,766	2214	17980	16873	15,766	14.04

Table 3.8 identifies an additional increase of secret bits, when ZWC algorithm is applied to a carrier webpage. Table 3.9 shows carrier object after and before insert secret data, as Table 3.9 also shows hidden bits positions in carrier object.

Table 3.9 Position of Secret Bits in Carrier Object

Cover Object	كان ساحل مصر الشمالي سلّة غذاء مصر والامبراطورية الرومانية التي كانت تحتل مصر قبل الإسلام. فقبل بناء السد العالي، اعتمد المصريون على مياه النيل في الزراعات الصيفية في الوادي والدلتا كما اعتمدوا على مياه الأمطار في زراعة القمح
Stego Object	كان ساحل مصر الشمالي سلّة غذاء مصر والامبراطورية الرومانية التي كانت تحتل مصر قبل الإسلام. فقبل بناء السد العالي، اعتمد المصريون على مياه النيل في الزراعات الصيفية في الوادي والدلتا كما اعتمدوا على مياه الأمطار في زراعة القمح
Secret Bits Positions	كن ساحل مصر الشمالي سلّة غذاء مصر والامبراطورية الرومانية التي كانت تحتل مصر قبل الإسلام. فقبل بناء السد العالي، اعتمد المصريون على مياه النيل في الزراعات الصيفية في الوادي والدلتا كما اعتمدوا على مياه الأمطار في زراعة القمح ٩

3.6 Remarks Algorithm

Remarks Steganography Algorithm is published in ASEE 2013[41]. The algorithm uses a text file as a carrier to hide the data inside it. The main goal of this idea is to hide data inside a word file without any change in the file format. A Stegoanalyzer program analyzes the file content and formatting. If there is any change in the file format, Stegoanalyzer program can detect the hidden data. In Remarks Algorithm the Right-to-Left Remark (U200F) symbol “^٩” and the Left-to-Right Remark (U200E) symbol “^٩” are used to hide the

bits inside the carrier file. This method will not change the format of the file and can also be applied to different languages, regardless of the UNICODE or ASCII coding. This method is easily applied to the Microsoft Office Word application in order to hide data.

Algorithm I: Hiding Data

Input: Carrier file, hidden bits file

Output: Stego file (embedded U200E && U200F file)

Step 1: Choose any DOC file

Step 2: Repeat while! (EOF)// repeat until the end of the hidden file

Step 3: Embed hidden data in the selected file

Step 3(a): Start from the first letter of the carrier file

Step 3(b): Pack out the first two hidden bits

If 00, then no U200F nor U200E

Else if 01, then there is U200F

Else if 10, then there is U200E

Else add U200F and U200E.

Step 5: Go to step 2

Step 6: Save file as PDF then send it to other side.

Algorithm II: Data Extraction

Input:-Stego file

Output: - Secure data, original file

Step 1: Open PDF Message

Step 2: Repeat while! (EOF)// repeat until the end of Stego file

Step 3: Embed hidden data in selected file

Step 3(a): Separate each letter

Step 3(b):

If there is nothing then 00

Else if only U200F then 01

Else if U200E then it's a 10

Else, 11

Step 4: Go to step 2

Step 5: Read hidden data.

Due to the symbol insertion, The Remark Algorithm has an effect on the file size, which depends on the hidden data that may increase dramatically. In order to solve this file size issue, identified in the Remark Algorithm.

Prior to embedding data, statistical information was collected representing the percentage of ones and zeros, and then the following equation was applied.

$$f(x) = \begin{cases} \text{if zero's} > \text{one's} & \text{Apply the same algorithm} \\ \text{if one's} > \text{zero's} & \text{Switch between Scenario 1 and Scenario 4} \end{cases} \quad (1)$$

By applying the above-mentioned equation, optimal result would be evident when all hidden data are zeros or ones, and the file size remains unchanged. Alternatively, an unfavorable result would emerge if half of the hidden data consists of zeros, and the other half comprises of ones. Through Table 3.10, the Remark algorithm simulation results demonstrate the number of words that can be embedded in the hidden capacity ratio.

Table 3.10. Hidden ratio be using Remarks Algorithm

#	Website	Number of words that can be embedded	Text Size (Kilo Byte)	Capacity Ratio
1	www.nydailynews.com	826	8.8	674
2	www.aljazeera.com	1658	18.7	637
3	www.englisharticles.info	1351	15.9	610
4	www.latimes.com	1208	14.8	586

3.7 Steganography over HTML Code

In this paper[43], we employ Cryptography and Steganography techniques to pass secure information. WebPages are used as the carrier for secret data, and these WebPages are published over the Internet. Authenticated users can access the hidden data. The proposed algorithm consists of three main steps, as shown in Figure 3.5, where the first and third steps represent inverse operations.



Figure 3.5. Steganography Process

The Conceal operation consists of six stages:

1. The Statistical Stage consists of an array of 26 elements that count the characters' frequency. The frequency array can be increased or decreased based on a Webpage's language. Our experiments are based on the English language.
2. The Character Representation Stage assigns "bits" based in the frequent use of the character. For example, after the frequency array has been generated. The lowest two frequently found characters could be represented by one bit. If two characters have the same frequency number, the character order specifies which one is zero. For example, if letter 'X' appears 10 times and a letter 'Z' appears 6 times, then 'Z' is represented as '0', and 'X' as '1'. Moreover, if both letters have the same occurrence number, then 'X' is represented as '0' and 'Z' is represented as '1'. Similarly, the next four characters can be represented by two bits.
3. The Embedding Stage occurs when the secret bits are embedded after the character representation equals '8' bits. In other words, if the first character representation is '0' and the secret bits are "0111011", the code will be "00111011".

4. The Encryption Stage: consists of three simple binary operations. First, the binary representation is complemented. Then, “exclusive OR” (XOR) is performed with the binary key. In order to produce the output of XOR gate, shift left must be applied by one bit and reentered as XOR input. In addition, the binary key code creation depends on the Webpage index where each page has a rear index. This operation repeats twice as shown in Figure 3.6.

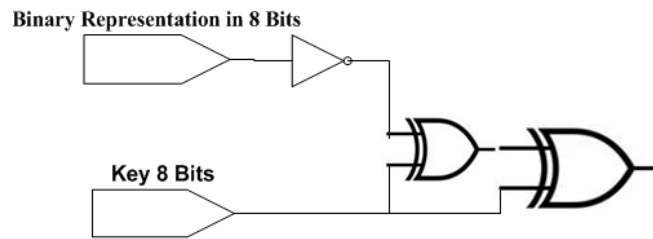


Figure 3.6. Encryption Gates

This numerical example explains that the input is (C) 01000011, and the key is 10001100:

Step 1:- Binary representation for C =>01000011

Step 2:- 1's complement of C => 10111100

Step 3:- 10001100 XOR 10111100=>00110000

Step 4:-Shift left 00110000 => 01100000

Step 5:- 10001100 XOR 01100000=>11101100

Step 6:- Shift left 11101100=>11011001

5. The Decoding Stage converts binary code to ASCII code. Here, the numerical form is changed into text. In the example, shown above “11011001” is decoded to (Ù).
6. The Insertion Stage occurs when inserting the text from the decoding stage is produced. Then, the text is placed into the Webpage code as comment. The comments do not appear in the Webpage's output view.

To reveal the secret message apply the following procedural stages:-

1. The Statistical Stage relies on 26 characters, similar to those found in the “Conceal Operation” statistical stage.
 2. The Reading Comments Stage allows the user to read comments from the Webpage.
 3. The Encoding Stage converts any comment appearing in the text into binary representation.
 4. The Decryption Stage follows in the same binary processes that exist in the encryption stage founded in the “Conceal Operation”.
 5. The Reveal Character Code Stage is performed by using the frequency array created in the statistical stage, and comparing it with the binary output of the decryption stage.
- The embedded information is obtained by removing the character representation.

3.7.1 Experiments and Results

This section explains the concepts of decryption of Hidden Data Ratio and the results formed after applying HTML Code Algorithm. In Figure 3.7, letter frequency is displayed by applying the HTML code Algorithm to different news Websites.

$$letter\ Freq = LEN(website) - LEN(SUBSTITUTE(Website, letter)) \quad (2)$$

When applying the HTML Code Algorithm, the max volume of secret hidden bit is 122 bits, regardless of the Website size. Moreover, the suggested algorithm utilized non-pure Steganography by employing encryption gates in order to improve the system’s transparency. In addition, the statistical equation improves the system’s robustness. By improve robustness; the sensitive message remains unchanged during the transformation process.

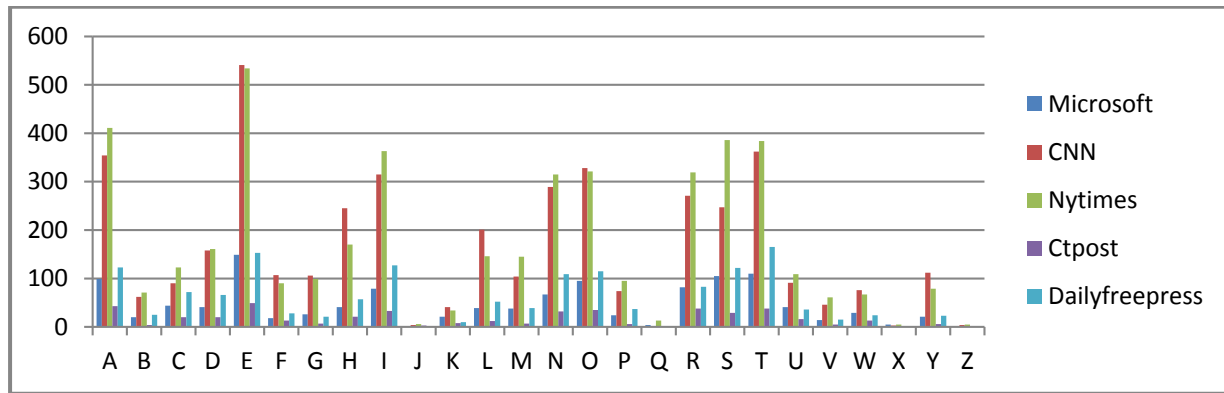


Figure 3.7. Letter Frequency

3.7.2 HTML Code Algorithm Analysis

HTML Code Algorithm has a number of advantages over other algorithms. This section explores some of it's benefits.

1. Language independency:

The HTML Code algorithm can be applied to multiple languages. This ability enables users to employ it regardless of the language used on the Webpage. Different languages have different frequency array sizes. For example, if the Webpage contains Arabic letters, then the array size is 28. If the Website contains English text, then the array size is 26.

2. Algorithm Transparency:

One of the most important criteria in measuring the performance of a Steganography algorithm is the ability to prevent hacker from reading hidden data. The HTML Code Algorithm improves the transparency feature by hiding encrypted data inside the code. In addition, the secret bits are inserted as comments, and the comments do not appear on the screen of the Webpage. The proposed technique eliminates the need to change file format in

order to reduce intruder suspicion.

3. Hidden Ratio Capacity:

The HTML Code algorithm hides different amount of bits inside each Webpage. For example, if we assume the Webpage is written in English text, then the maximum amount of hidden bits per page is 126.

4. Algorithm Reusability:

The HTML Code algorithm enables the user to create an individual Webpage and reuse the same Webpage to hide different secret messages.

5. Algorithm Robustness:

The HTML Code Algorithm prohibits any change to carrier Webpage code during the transmission process, since the hidden data is stored in the page code as comments.

3.8 Steganography in Text by Using MS Word Symbols

The MS Word Symbols Algorithm [44] hides data inside a Word file without altering the carrier file properties such as file size, content and format. The MS Word Symbols Algorithm employs some invisible symbols to hide four bits between letters. This process improves the hidden capacity ratio compared to other algorithms. Moreover, no modification in the Word format file or letter shapes would be made. Thus, the suggested algorithm avoids raising any hackers' suspicions.

Table 3.11 represents some of the hidden codes. For example, if we insert all four-table variation symbols after each letter, then the passing bits code is 0000. This insertion enables us to hide four bits of secret data. The four symbols applied are Right remark

(200E), Left remark (200F), Zero width joiner (200D), and Zero width non-joiner (200C).

In this technique, different variations can be used to represent hidden bits for a total of 16 different codes.

Table 3.11 Sample of Hidden Bits by Using Word Symbols

<i>Right Remark</i>	<i>Left Remark</i>	<i>ZWJ</i>	<i>ZWNJ</i>	<i>Hidden code</i>
X	X	X	X	0000
X	X	X		0001
X		X		0101
X	X			0011
X				0111
				1111
		X		1101

Figure 3.8 represents the data hiding steps when using three inputs: the Stego key, carrier file, and hidden data. The main purpose of Stego key is to change the symbols' bit representation. For example, '0' represents a bit's absence, and '1' represents a bit's presents. In the next step, a symbols' table is created depending on the outcome of the Stego key.

The capacity of the carrier file is computed as follows:

$$\text{Capacity of carrier file} = \text{Number of letters} \times 4 \quad (3)$$

The hidden capacity of our algorithm is computed as follows:

$$\text{Capacity Ratio} = (\text{Number of letters} \times 4) / \text{carrier file size} \quad (4)$$

The receiver can extract the hidden data by reading the carrier file, and then apply the Stego key to build a symbols' table. By reading the symbols after each letter and matching those with the symbols' table would enable the receiver to extract the hidden data.

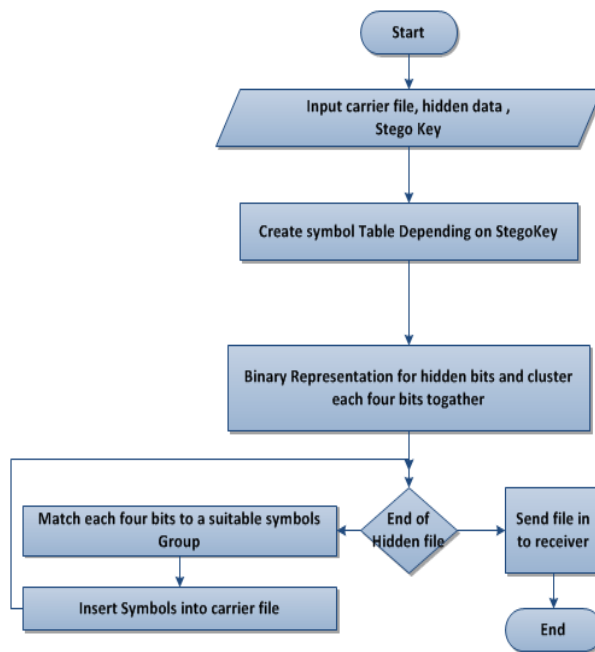


Figure 3.8. Data Hiding Algorithm

Table 3.12 presents the simulation's results using MS Word, Line shift, and Word shift algorithms. Based on the simulation, MS Word's hidden data ratio is comparatively higher than the Line shift and Word shift algorithms. Figure 3.9 shows a comparison histogram for the three algorithms.

Table 3.12. Hidden Capacity of 3 Text Steganography Algorithms

#	Website	Size (K.B)	Number of lines	Number of words	Number of letters	MS Word Symbols	Line shift	Word shift
1	www.cnn.com	19.8	74	763	4592	928	4	39
2	www.bbc.com	19.3	67	749	4065	842	3	39
3	www.nypost.com	19.8	48	634	3532	714	2	32
4	www.guardian.co.uk	21	64	935	5625	1071	3	45
5	www.ctpost.com	20.5	51	640	3652	713	2	31

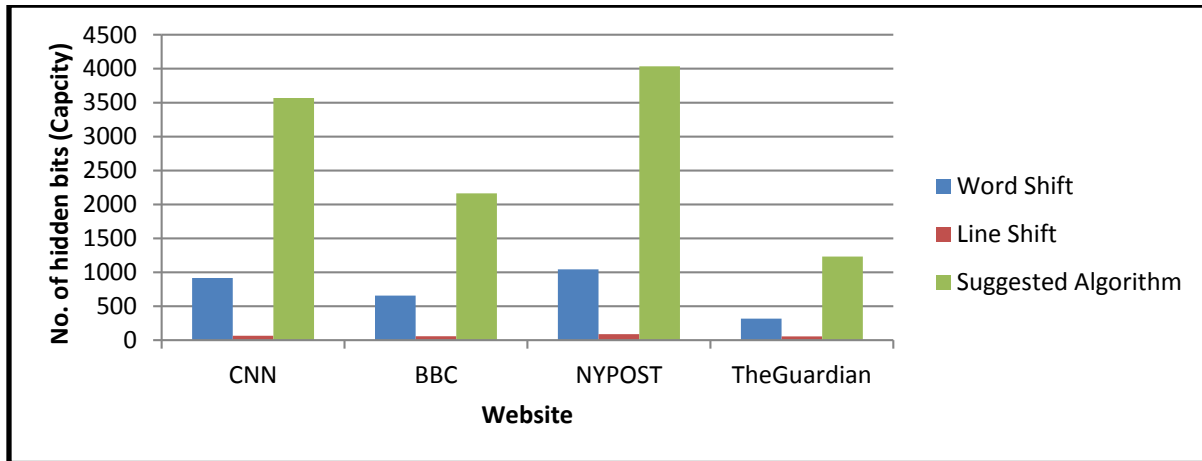


Figure 3.9. Hidden Bits Comparison between Three Different Algorithms

3.9 Performance Evaluation

Table 3.13 shows the proposed algorithms and their classifications. Multipoint, Diacritics, ZKA, and KVA algorithms are language dependent. These algorithms (A1, A2, A3, and A4) have the potential to hide secret data inside Unicode Languages such as Arabic, Persian, and Urdu. Multipoint and Diacritics algorithms hide data by substituting one letter with another modified shaped letter. This approach works well, since there is no standard position or distance between the letter and the points. Also, this approach has proven that there is no standard position between the letter and the Diacritics. The main advantage of the A1 and A2 algorithms is the file's size stability. Moreover, A1 and A2 have the ability to combine with A3, A4, A5, A6, A7, and A8 in order to improve the hidden ratio capacity. A3 and A4 hide the data inside a carrier object by inserting non-printed symbols. The main drawback of A3 and A4 is that the carrier file's size increases with the process of embedding the secret message.

A5, A6, and A8 hide the data by embedding non-printed symbols. In addition, A5, A6, and A8 provide a highly hidden ratio capacity. Moreover, these algorithms have the ability to be combined with A1 and A2 in order to improve the hidden ratio capacity. One disadvantage associated with A5, A6, and A8 is that the carrier file's size increases through the insertion of non-printed symbols. HTML Code A7 algorithm hides the encrypted secret data as comments inside the HTML Webpage. Limited hidden bits can be inserted based on the webpage's original language. The hidden capacity of HTML Code A7 algorithm can be enhanced by combining it with A1 and/or A2. Table 3.14 explains the results of combining one algorithm with another algorithm and the effect on the hidden ratio capacity and the file size. It is notable that combining one algorithm with another will increase the hidden ratio capacity at a risk of increasing the file size. The only case that the file size does not increase is when A1 and A2 are combined. Here, either A1 or A2 did not increase the carrier file size. These algorithms are only based on changing the points or the Diacritics positions.

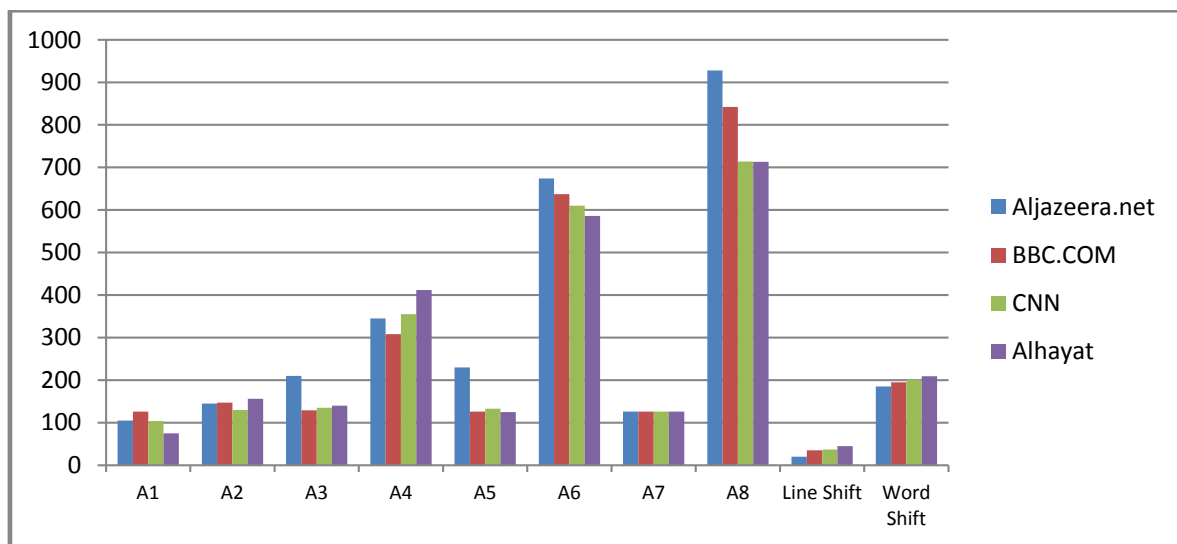


Figure 3.10. Hidden Ratio [Bit/Kbyte]

Figure 3.10 illustrates and compares the hidden ratio of eight proposed algorithms

and Line and Word Shift algorithms introduced in literature. Equation 5, stated below, represents the hidden ratio capacity (HR). Based on the simulation results, A8 achieves the highest hidden ratio range. A6 achieves the second highest hidden ratio range. A7 represents a constant hidden ratio range. The average range of the hidden ratio of A1 = 104 bits per Kbyte (KB); A2 = 144 b/KB; A3 = 153 b/KB; A4 = 355 b/KB; A5 = 153 b/KB; A6 = 626 b/KB; A7 = 127 b/KB; and A8 = 799 b/KB.

$$HR = \frac{\text{Number of bits embedded}}{\text{Carrier file Size [KByte]}} \quad (5)$$

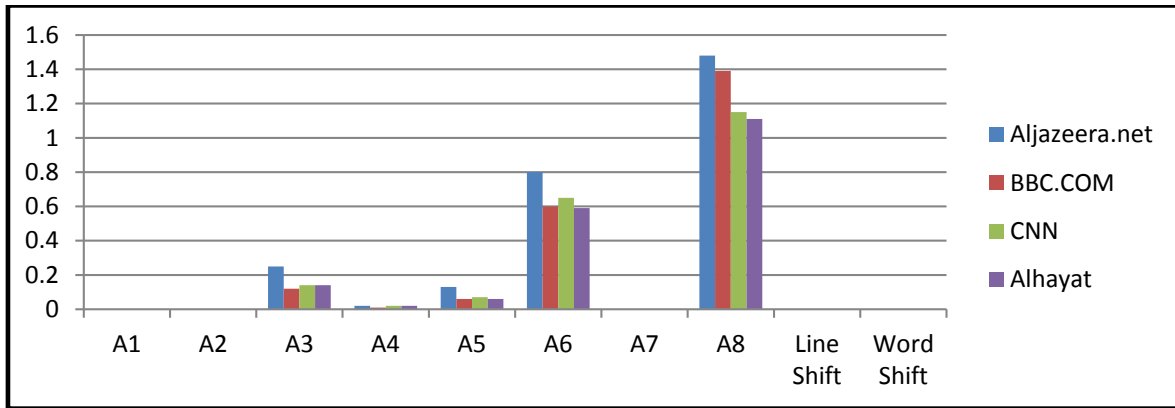


Figure 3.11. File Size Effect after Inserting Secret Data

Figure 3.11 represents and compares the file size of the Stego objects that increase the hidden ratio capacity of eight proposed algorithms with Line and Word Shift algorithms.

Based on the simulation results for the hidden ratio range, there is no change in the carrier file size when each A1, A2, and A7 is applied. On other hand, the highest file increase ratio appears in A8. A6 offers the second highest file size increase.

3.10 Merge Capability

Table 3.13 describes the proposed algorithms and their classification. Multipoint, Diacritics, ZKS, and KVA are language dependent. These algorithms (A1, A2, A3, and A4)

could be used to hide secret data inside Arabic, Persian, and Urdu. Multipoint and Diacritics algorithms hide data by substitute one letter by another modified shaped letter. Where there is no standard position or distance between letter and points or between letter and Diacritic. The main advantage of these algorithms (A1, A2) is the file size stability. Moreover, A1 and A2 have ability to merge with ((A3, A4, A5, A6, A7, and A8) to improve the hidden ratio capacity. A3, A4 hide data inside a carrier object by insert non-printed symbols. ZKS and KVA are language dependent. The main drawback of A3 and A4 is carrier file size increment.

A5, A6, and A8 hide data by embedded non-printed symbols. In addition, A5, A6, and A8 present highly hidden ratio. Moreover, they have ability to merge with A1 and A2 to improve hidden capacity ratio. On the other hand, A5, A6, and A8 have carrier file size increment problem. HTML Code algorithm suggested hiding encrypted secret data inside HTML Webpage as comments. Limited hidden bits can be inserted based on Webpage language. The hidden capacity of HTML Code algorithm can be enhanced by merge with A1, A2.

In this section, we discuss the possibility of merging more than one algorithm. The proposed algorithms can be merged collectively to further improve the hidden ratio range of the carrier file. For example, A1 can be combined with either A2, A3, A4, A5, A6, A7 or A8. Figure 3.12 illustrates the hidden ratio output when A1 is combined with other algorithms. Figure 3.13 represents the change in the carrier file size, once other algorithms have been merged with A1. The merging process will be done in sequence. For example, the user can select A1 algorithm. The carrier file will be updated based on the secret data, and then the user can get the output file of the A1 algorithm and select the second suggested algorithm.

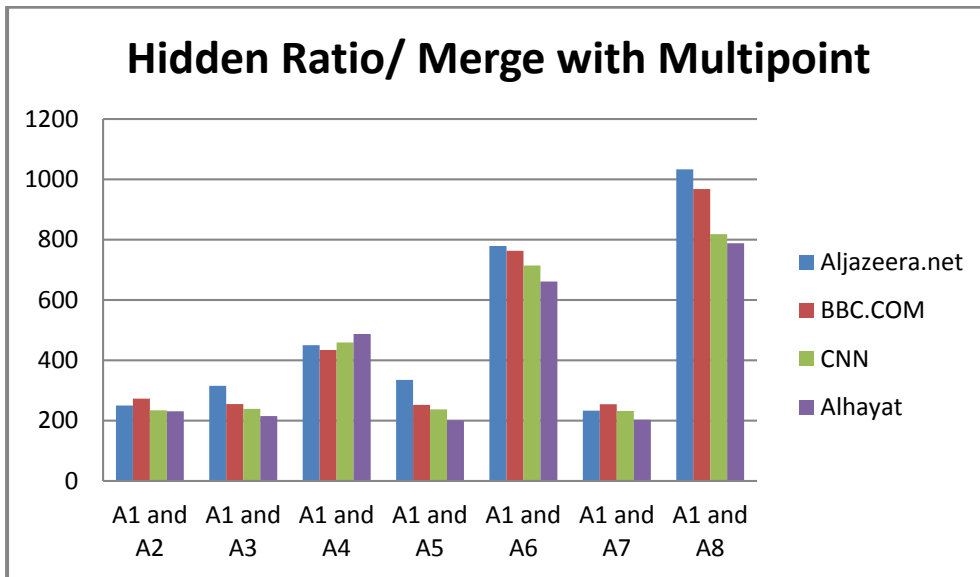


Figure 3.12. Hidden Ratio Merged with Multipoint

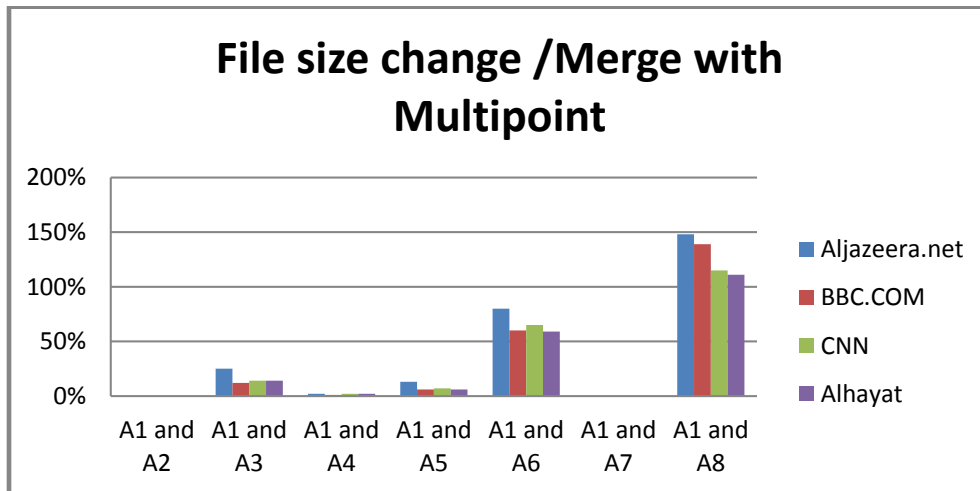


Figure 3.13. File Size Change Merged with Multipoint

Based on the simulation results, the best outcome will depend on the individual user requirements. If the file size is not allowed to change, then the user will select A1, A2, or A7. Otherwise, if the carrier file size is allowed to increase, and large amounts of secret data are being embedded in the carrier file, then A8 will be the best algorithm choice. One distinctive merging algorithm scenario is (A1, A2, A4, A7) Multipoint, Diacritics, KVA, and HTML Code. This scenario provides a highly hidden ratio while the size of the carrier file will

minimally increase, as shown in Table 3.13. The results show that the merging scenario is very consistent among different Websites in terms of hidden ratio capacity and the file size change.

Table 3.13. Merged Simulation Results of A1, A2, A4, A7

<i>Webpage</i>	<i>Hidden Ratio b/KB</i>	<i>File size Change</i>
Aljazeera	723	2%
CNN	709	1%
BBC	717	2%
Alhayat	771	2%

Table 3.14. Comparison between Eight Different Algorithms

<i>Algorithm</i>	<i>Applicable Language</i>	<i>General Categories</i>	<i>Technique</i>	<i>File size</i>	<i>Ability to merge with other Algorithm(s)</i>
A1: Multipoint	Unicode(Arabic, Persian, Urdu)	Substitution	Linguistic	Not Effect	Can be merge with all other algorithms (A2,A3,A4,A5,A6)
A2: Diacritics	Unicode(Arabic, Persian, Urdu)	Substitution	Linguistic	Not Effect	Can be merge with all other algorithms (A1,A3,A4,A5,A6)
A3: ZKS	Unicode(Arabic, Persian, Urdu)	Injection	Linguistic	Increase	Can merge with (A1,A2)
A4: KVA	Unicode(Arabic, Persian, Urdu)	Injection	Linguistic	Increase	Can merge with (A2)
A5: ZWC	Language Independent	Injection	Format	Increase	Can merge with (A1,A2)
A6: Remarks	Language Independent	Injection	Format	Increase	Can merge with (A1,A2)
A7: HTML Code	Language Independent	Injection	Format	Increase	Can merge with (A1,A2)
A8: MS Word	Language Independent	Injection	Format	Increase	Can merge with (A1,A2)

Table 3.15. Output of Merge Process between Proposed Algorithms

<i>Algorithm</i>	<i>Applicable Language</i>	<i>General Categories</i>	<i>Technique</i>	<i>Hidden Capacity</i>	<i>File Size</i>
A1 (A3,A4,A5,A6)	Unicode(Arabic, Persian, Urdu)	Substitution/Injection	Linguistic/format	Increase	Increase
A2 (A3,A4,A5,A6)	Unicode(Arabic, Persian, Urdu)	Substitution/Injection	Linguistic/format	Increase	Increase
(A1,A2)	Unicode(Arabic, Persian, Urdu)	Injection	Linguistic	Increase	No change
A3 (A1,A2)	Unicode(Arabic, Persian, Urdu)	Substitution/Injection	Linguistic/format	Increase	Increase
A4 (A2)	Unicode(Arabic, Persian, Urdu)	Substitution/Injection	Linguistic/format	Increase	Increase
A5(A1,A2)	Unicode(Arabic, Persian, Urdu)	Substitution/Injection	Linguistic/format	Increase	Increase
A6 (A1,A2)	Unicode(Arabic, Persian, Urdu)	Substitution/Injection	Linguistic/format	Increase	Increase
A7(A1,A2)	Unicode(Arabic, Persian, Urdu)	Substitution/Injection	Linguistic/format	Increase	Increase
A8(A1,A2)	Unicode(Arabic, Persian, Urdu)	Substitution/Injection	Linguistic/format	Increase	Increase

CHAPTER 4 : HARDWARE SIMULATION

In Chapter 4, we present the hardware simulation results of our proposed algorithms. In addition, sections (4.1, 4.2, 4.3, 4.4, 4.5, and 4.6) explain real-time Steganography techniques that hide data inside a text file using a hardware engine.

Most of the introduced techniques use software implementation to embed secret data inside the carrier file. Nevertheless, software implementations are not sufficiently fast for real-time applications. In this chapter, we present new real-time Steganography techniques that hide data inside a text file using a hardware engine.

In [45], a novel hardware design was proposed for image Steganography using the least significant bit (LSB) algorithm. The implementation was carried out using Cyclone II FPGA of the ALTERA family. The technique employed 2/3LSB design to produce a good image quality to avoid any attacker doubt. Meanwhile, it provided a high memory access performance to speed up the system performance. In [46], an FPGA hardware architecture was introduced to hide the secret information by exploiting the noise regions in an image. This strategy improved system transparency which made it hard to realize the hidden data. In [47], an implementation of audio or video Stenography using FPGAs was discussed. The proposed algorithm speeds up the secret data embedding rate at the hardware implementation for real-time Steganography. Another hardware architecture was introduced in [48] to simulate the ability to hide information

inside image and video carrier files. Two schemes were applied to speed up real-time video applications. The main drawback of this system is its need for a high speed memory buffer. In [49], the proposed algorithm employed image as carrier file by using multilayer embedding in parallel with three-stage pipeline on FPGA. Promising results showed high throughputs while maintaining the image quality. In [50] and [51], authors employed perturbed quantization to hide data inside JPEG image. The main feature of perturbed quantization is that it is undetectable with current Stego-analyst.

As can be seen, most of the presented algorithms were implemented in hardware focus on image, video, or audio as the carrier file for the secret message. This is while text Steganography has not been considered for implementation in hardware engines and/or digital signal processors.

In this Chapter, we build on top of our prior works and apply our algorithms over hardware concepts to speed up the system efficiency. In our software implementation, we try to achieve the highest Steganography performance “Magic Triangle Concepts” for Steganography; that is, the ability to achieve and maintain high transparency, robustness, and hiding capacity.

4.1 Multipoint Algorithm [52]

In order to simulate a Multipoint Algorithm into a hardware engine, a “State” transition diagram is constructed that reflects the Multipoint Algorithm’s procedure. Figure 4.1 illustrates the Finite “States” Machine Diagram of the Multipoint system. This system consists of five “States.” Each “State” depends on the input character in the text file and in the hidden data. “State” ‘A’ begins the process of the initial “State” of the

search. The hidden information represents input data that transfers from one “State” to another “State.” Figure 4.2 describes the main components of the hardware engine in RTL view. This Multipoint hardware system consists of four comparison units that scan hidden information in order to determine a suitable data path.

When applying “Quartus II” hardware compilation application,” to the Multipoint hardware simulation our findings indicate the critical path time equals $T_{\min\text{clk}} = 1.42\text{ns}$. In each cycle, we process 16 bits. The following equation is used to determine the maximum frequency:

$$f_{\max} = \frac{1}{T_{\min\text{CLK}}} = \frac{1}{1.42\text{ns}} = 704.22 \text{ MHz} .$$

The system, therefore, has an overall throughput of 11.27 Gbit/second.

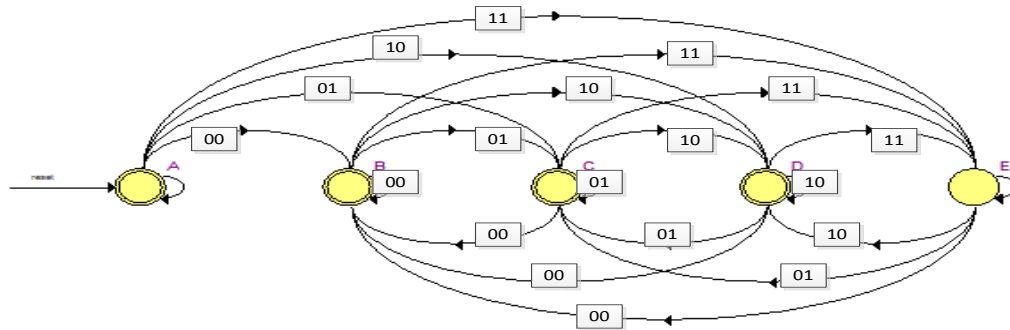


Figure 4.1. Finite “State” Machine Diagram for Multipoint

4.2 ZKA Algorithm

Our hardware engine is applied over text in a form of a carrier file through ZKA. This algorithm permits hiding data inside a Word file without any changes to the carrier file format. Figure 4.3 exemplifies the main components of the hardware engine in RTL view. The system consists of four comparison units that check the hidden information in order to choose a suitable data path.

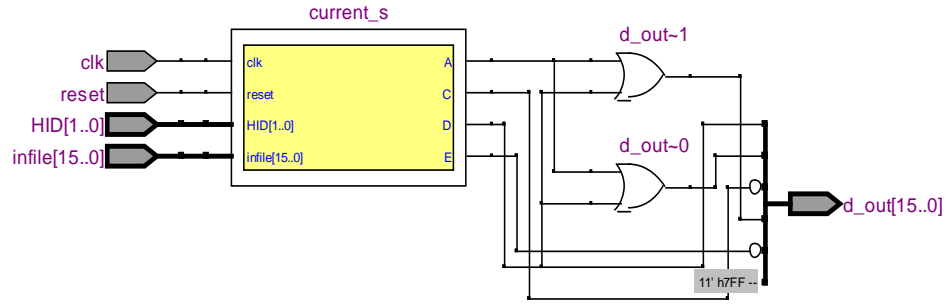


Figure 4.2. RTL View of the Hardware Engine for Multipoint

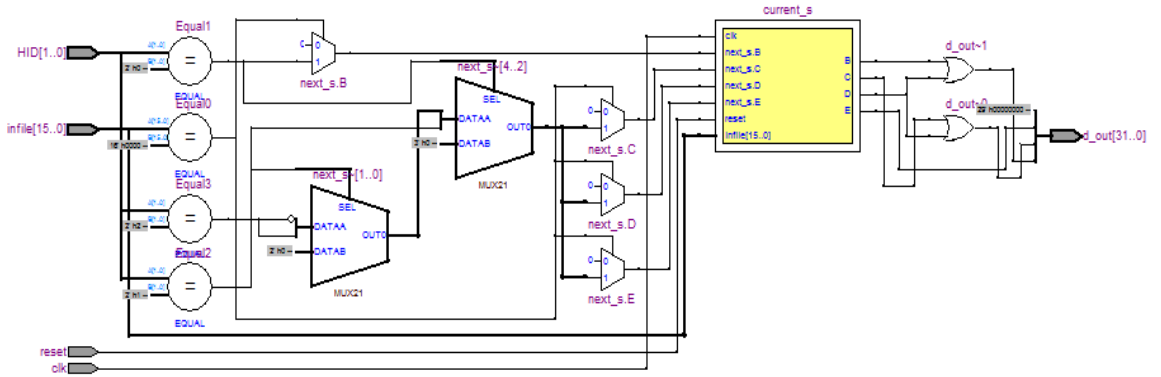


Figure 4.3. RTL View of the hardware engine ZKA Algorithm

The critical path time reported by the (Quartus II) is $T_{\min\text{clk}} = 2.494\text{ns}$.

Therefore, the maximum frequency is:

$$f_{\max} = \frac{1}{T_{\min\text{CLK}}} = \frac{1}{1.669\text{ns}} \approx 599 \text{ MHz} .$$

Hence 16 bits are processed in each clock cycle; the system has an overall throughput of 9.5 Gbit/second.

4.3 ZWC Algorithm[53]

For the ZWC Algorithm, we also configured the hardware engine. In our implementation, we process the hidden file two bits in each step to hide it and then make a transition from the current state to another state based on the conditions.

The critical path time reported by the (Quartus II) tool is $T_{\min\text{clk}} = 2.494\text{ns}$.

Therefore, the maximum frequency is:

$$f_{\max} = \frac{1}{T_{\min\text{CLK}}} = \frac{1}{2.494\text{ns}} \approx 401 \text{ MHz} .$$

Hence 16 bits are processed in each clock cycle; the system has an overall throughput of 6.415 Gbit/second.

4.4 Diacritics Algorithm

In order to build a Diacritics Algorithm into the hardware system, a “State” transition diagram is constructed that reflects the Diacritics Algorithm procedure. Figure 4.4 demonstrates the “State” diagram of the hardware system. This Diacritics hardware system contains three “States.” Each “State” depends on the input value character located in the text file and the hidden data. “State” A represents the initial “State” of the search. The hidden information transfers input data from one “State” to another “State.”

The hardware engine is configured based on this finite state machine in (Quartus II). Figure 4.5 illustrates the main components of the hardware engine in RTL view. The Diacritics hardware system contains four comparison units that check the hidden information to determine a suitable data path.

The critical path time reported by the application (Quartus II) was $T_{\min\text{clk}} = 1.824\text{ns}$. Therefore,

$$f_{\max} = \frac{1}{T_{\min\text{CLK}}} = \frac{1}{1.824\text{ns}} \approx 548 \text{ MHz} .$$

Hence 16 bits are processed in each clock cycle; the system has an overall throughput of 8.77 Gbit/second.

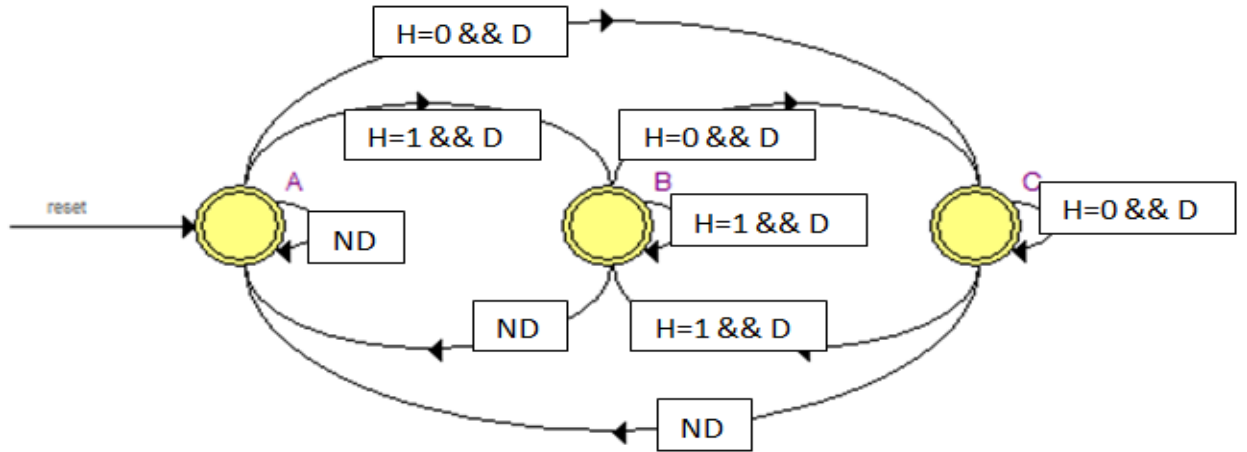


Figure 4.4 . Finite “State” Machine Diagram for Diacritics Algorithm [H= Hidden bit, D= diacritic]

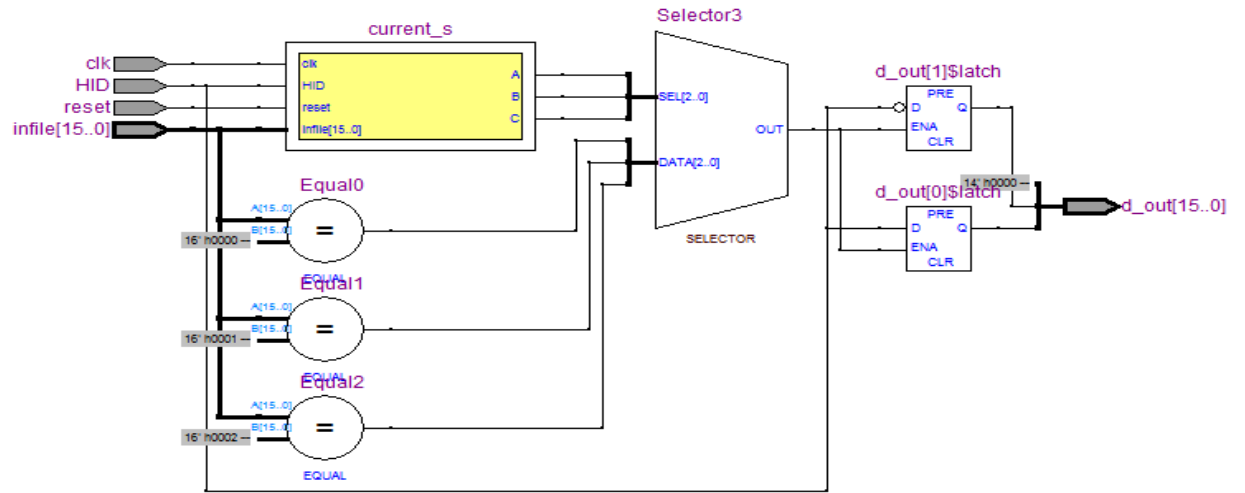


Figure 4.5. RTL View of the Hardware Engine for Diacritics Algorithm

4.5 KVA Algorithm

In order to configure the KVA in hardware, a “State” transition diagram is constructed that reflects KVA’s procedure. Figure 4.6 illustrates the Finite “State” Machine Diagram of the KVA system. The KVA system consists of three “States.” Each “State” depends on the input value character located in the text file and the hidden data. “State” A represents the initial “State” of the search. The hidden information transfers an input data from one “State” to another “State”.

Figure 4.7 illustrates the main components of the hardware engine in RTL view. The critical path time reported by the application (Quartus II) was $T_{\min\text{clk}} = 2.103\text{ns}$. Therefore,

$$f_{\max} = \frac{1}{T_{\min\text{CLK}}} = \frac{1}{2.103\text{ns}} \approx 475.5 \text{ MHz} .$$

Hence 16 bits are processed in each clock cycle; the system has an overall throughput of 7.608 Gbit/second

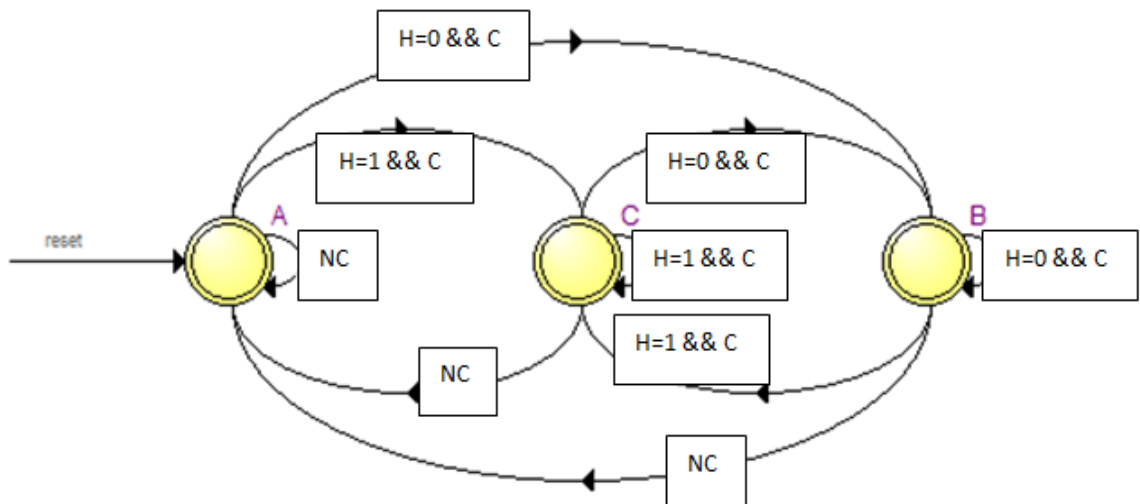


Figure 4.6. Finite “State” Machine Diagram for KVA [H= Hidden bit, C= Connected Letter, NC = NOT C]

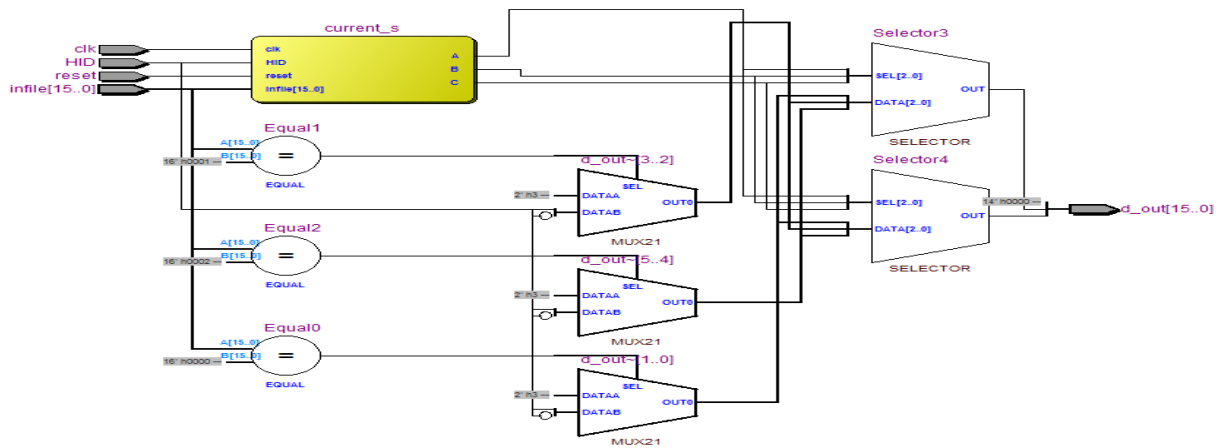


Figure 4.7. RTL View of the Hardware Engine for KVA

4.6 Remarks Algorithm[54]

As previously mentioned, in order to apply various algorithms into a hardware configuration, a “State” transition diagram is constructed that reflects each algorithm’s procedure. Figure 4.8 illustrates the Finite “State” Machine Diagram of the Remarks Algorithm. Figure 4.9 consists of five “States.” Each “State” depends on the input value character located in the text file and the hidden data. “State” A represents the initial “State” of the search. The hidden information transfers input data from one “State” to another “State.” As shown in Figure 28, the Remarks Algorithm detects the hidden bits and input data bytes at each “State”. If the hidden bits are “00”, the output is “Null”, and next “State” is ‘B’. If the hidden bits are “01”, the next “State” is ‘C’, and the symbol inserted in the output file is “U200F.”

Figure 4.9 represents the main components of the Remarks hardware engine in RTL view. The Remarks Hardware system consists of four comparison units that check the hidden information to determine a suitable data path.

The critical path time reported by the application (Quartus II) was $T_{\min \text{clk}} =$

1.669ns. Therefore,

$$f_{\max} = \frac{1}{T_{\min\text{CLK}}} = \frac{1}{1.669\text{ns}} = 599161174.36 \text{ Hz} .$$

Hence 16 bits are processed in each clock cycle; the system had an overall throughput of 9.6 Gbit/second.

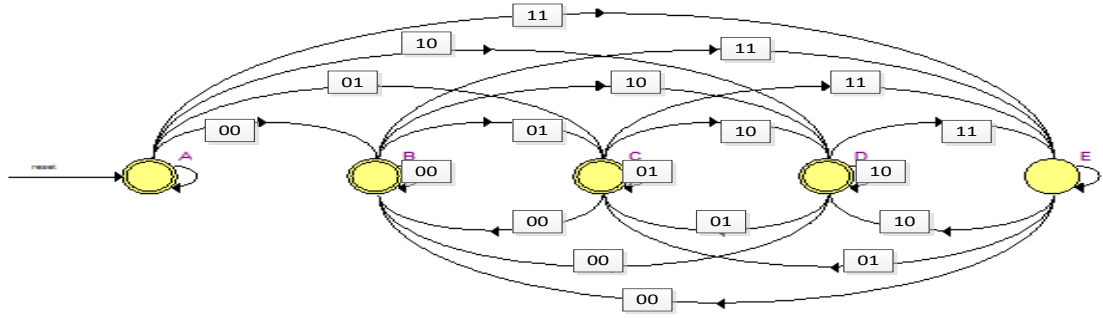


Figure 4.8. Finite “State” Machine Diagram for Remarks Algorithm

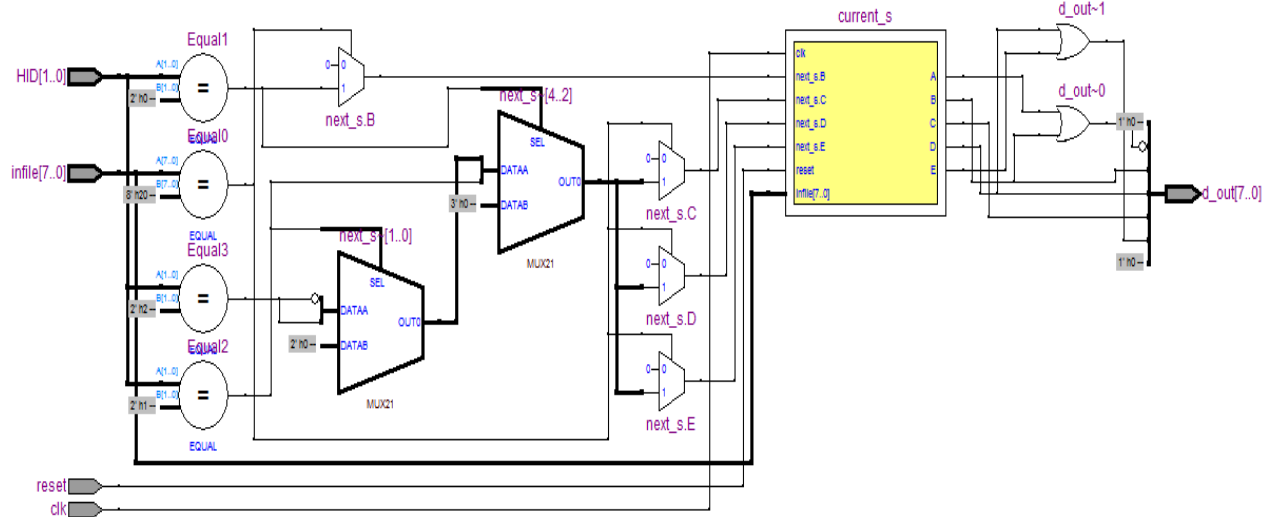


Figure 4.9. RTL View of the Hardware Engine for Remarks Algorithm

4.7 Analysis and Synthesis results

Table 4.1 shows the analysis and synthesis results of comprehensive text Steganography system. Our model enables users to choose the higher performance

algorithm according to the carrier file format. The suggested comprehensive model also supports the ability to change the applied algorithm after agreement between trusted parties. Table 4.2 summarizes the presented algorithms and throughputs.

Table 4.1. Analysis and Synthesis Results of Comprehensive System

ATTRIBUTE	VALUE
Top Level Entity Name	System
Family	Stratix II
Logic Utilization	<1%
Total registers	5
Total Pins	53/343 (15%)
Parallel compilation	Enable (4 Processors detected)
Hold time violation between Source pin and Destination pin	3.173ns
Largest skew	3.794ns
Shortest register to register delay	0.527ns
Longest register to register delay	10.170ns
Shortest clock path from clock to destination register	4.663ns
Longest clock path from clock to source register	7.561ns
Longest register to pin delay	4.370ns
Peak Virtual Memory	183 megabyte
Estimated ALUTs Used	60

Table 4.2. Algorithm Throughputs

Algorithm	Throughputs Gbit/second
Multipoint Algorithm	11.27
ZKA	9.5
ZWC Algorithm	6.415
Diacritics Algorithm	8.77.
KVA Algorithm	7.608
Remark Algorithm	9.6

Figure 4.10 shows the simulation waveforms of data inserted and the system states transformation using a timing wave graph. The system consists of three main inputs carrier file, hidden data, and selected algorithm. For example, if the user chooses the first option, it will select the ZWC algorithm. According to the hidden data the d_out will be changed and the next state will be updated.

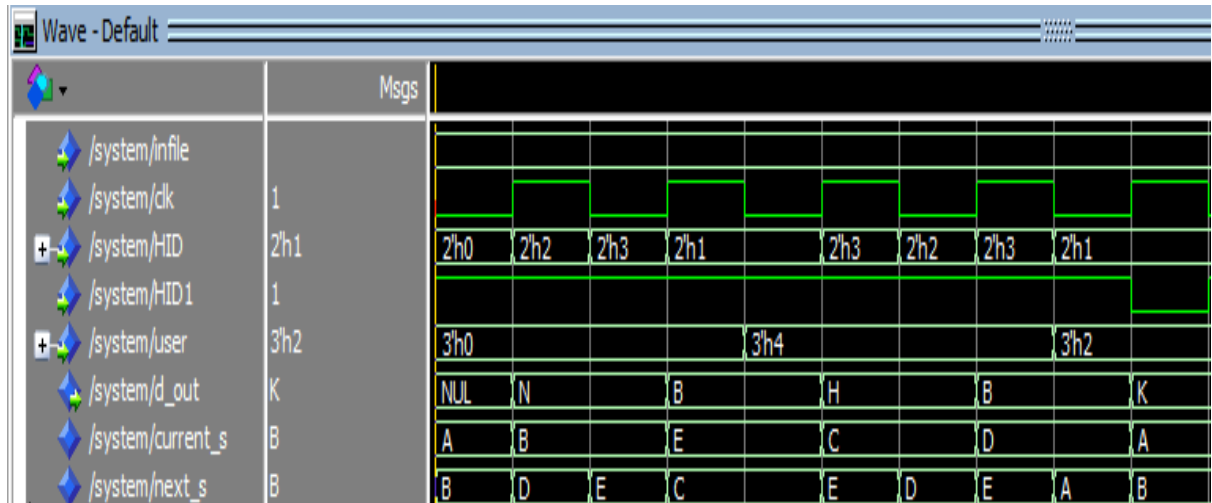


Figure 4.10. Hiding Data Algorithm's Timing Simulation Results and "State" Transitions

4.8 Summary

In this chapter, we tested fast and real-time hardware simulations to build secure and safe communication algorithms over cyber networks. We presented the hardware simulations of 6 original algorithms: (1) Multipoint; (2) ZKA; (3) Diacritics; (4) KVA; (5) ZWC; and (6) Remarks. The proposed hardware configurations represent extremely efficient processing speeds as applied to text Steganography in hardware applications. In addition, the proposed hardware systems are first of their kind to utilize text as a carrier file in Steganography. The suggested engines provides the flexibility to the users to select a specific algorithm or run multiple hardware engines simultaneously to improve the system speed and enhance hidden data ability. Furthermore, the proposed hardware system could be embedded in a router, or a gateway, or in a Network Interface Card to improve security over public communication channels.

CHAPTER 5 : CONCLUSIONS

The novel eight algorithms: Multipoint, ZKA, Diacritics, KVA, ZWC, Remarks, HTML code, and MS Word symbols presented in this dissertation provide additional and necessary protection for communicating sensitive data over the cyber world. In this research, we have outlined a list of existing text Steganography techniques, and presented eight original algorithms to increase the protection of data hiding inside text carrier files. The algorithm simulation results present a highly hidden data ratio that can be adopted in multiple languages.

Our proposal's results and research offer valuable and vital utilization in order to establish safe communication, privacy enhancement, secure data sharing, and additional protection of copyrighted products. For example, our algorithms can be applied to multimedia and publishing products. By using any of the eight algorithms presented in this proposal, users will have a more secured system in place to prevent and curtail hackers.

The proposed hardware systems represent extremely efficient processing speed as applied to text Steganography in hardware applications. In addition, the proposed hardware systems are first of their kind to utilize text as a carrier file in Steganography.

CHAPTER 6 : REFERENCES

- [1] B. Joonsang, N. Jan, S.-N. Reihaneh, and S. Willy, "A survey of identity-based cryptography," presented at the Australian Unix Users Group Annual Conference, 2004.
- [2] F. Piper and S. Murphy, *Cryptography: A Very Short Introduction*: Oxford University Press Publishing 2002.
- [3] K. Jonathan and L. Yehuda, *Introduction to modern cryptography: principles and protocols*: Chapman & Hall/CRC, 2007.
- [4] R. Kefa, "Steganography-the art of hiding data," *Information Technology Journal*, vol. 3, pp. 245-269, 2004.
- [5] P. Fabien, A. Ross, and K. Markus, "Information hiding-a survey," *IEEE, special issue on protection of multimedia content*, vol. 87, pp. 1062-1078, 1999.
- [6] R. Joseph and S. Viny, "Cryptography and Steganography-A Survey," *International Journal of Computer Technology and Applications*, vol. 2, p. 4, 2011.
- [7] H. Shirali and M. Shirali, "Text steganography in chat," presented at the 3rd IEEE/IFIP International Conference in Central Asia on Internet, 2007.

- [8] L. Minhao, G. Yunbiao, and Z. Linna, "Text Steganography Based on Online Chat," presented at the Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing, 2009.
- [9] H. Shahreza and M. Shahreza, "A New Synonym Text Steganography," presented at the International Conference on Intelligent Information Hiding and Multimedia Signal Processing., 2008.
- [10] Y. Liu, X. Sun, C. Gan, and H. Wang, "An efficient linguistic steganography for Chinese text," presented at the IEEE International Conference on Multimedia and Expo., 2007.
- [11] M. Hasanah, R. Sharifah, and S. Asma, "Synonym based malay linguistic text steganography," presented at the Innovative Technologies in Intelligent Systems and Industrial Applications, 2009.
- [12] W. Zhi-Hui, C. Chin-Chen, and L. Ming-Chu, "Emoticon-based text steganography in chat," presented at the Asia-Pacific Conference on Computational Intelligence and Industrial Applications, 2009.
- [13] A. Kalavathi and P. S. Rama, "An evolution of hindi text steganography," presented at the Sixth International Conference on Information Technology: New Generations, 2009.
- [14] A. Kalavathi and R. Prasad, "A New Approach to Hindi Text Steganography Using Matraye, Core Classification and HHK Scheme," presented at the Seventh International Conference on Information Technology: New Generations (ITNG), 2010.

- [15] P. RSR and K. Alla, "A new approach to Telugu text steganography," presented at the IEEE Symposium on Wireless Technology and Applications (ISWTA), 2011.
- [16] A. Sravani, P. Sake, and B. Ashok, "A New Approach to Telugu Text Steganography By Shifting Inherent Signs," *International Journal of Engineering Science and Technology*, vol. 2, pp. 7203-7214, 2010.
- [17] B. Lawson and R. Sharp, *Introducing html5*: New Riders, 2011.
- [18] M. Garg, "A Novel Text Steganography Technique Based on Html Documents," *International Journal of Advanced Science and Technology*, vol. 35, pp. 129-138, 2011.
- [19] G. Ian, *The HTML sourcebook*: John Wiley & Sons, Inc, 1995.
- [20] K. Matz, "Designing and evaluating an intention-based comment enforcement scheme for Java," Department of Computing, The Open University 2010.
- [21] B. Krista., "Linguistic steganography: Survey, analysis, and robustness concerns for hiding information in text," Purdue University, West Lafayette 2004.
- [22] P. Singh, R. Chaudhary, and A. Agarwal, "A Novel Approach of Text Steganography based on null spaces," *IOSR Journal of Computer Engineering*, vol. 3, pp. 11-17, July-Aug. 2012 2012.
- [23] M. Shahreza, "A New Method for Steganography in HTML Files," *Advances in Computer, Information, and Systems Sciences, and Engineering*, pp. 247-252, 2006.
- [24] S. Ahmed, X. Gu, and M. Jia, "Chinese Language Steganography using the Arabic Diacritics as a Covered Media," *International Journal of Computer Applications IJCA*, vol. 11, pp. 24-28, 2010.

- [25] R. Alifah, M. Ramlan, and U. Izura, "Sharp-Edges Steganography in Arabic Characters for Information Hiding," *Journal of Theoretical and Applied Information Technology*, vol. 33, pp. 32-41, 2012.
- [26] M. Shahreza, "Pseudo-space Persian/Arabic text steganography," presented at the IEEE Symposium on Computers and Communications, 2008.
- [27] M. Shahreza and S. Shahreza, "Persian/Arabic Unicode Text Steganography," presented at the Fourth International Conference on Information Assurance and Security, Naples, 2008.
- [28] A. Mohammed, A. S. M, E. A-RM, and G. Adnan, "Arabic diacritics based steganography," presented at the IEEE International Conference on Signal Processing and Communications, 2007.
- [29] B. Mohamed and Y. Mohamed, "High capacity diacritics-based method for information hiding in Arabic text," presented at the International Conference on Innovations in Information Technology (IIT), 2011.
- [30] G. Adnan, A.-A. Wael, and A. Bin, "Improved Method of Arabic Text Steganography Using the Extension 'Kashida'Character," *Bahria University Journal of Information & Communication Technology*, vol. 3, pp. 68-72, 2010.
- [31] A.-H. Fahd, G. Adnan, A.-K. Khalid, and H. Jameel, "Improving security and capacity for Arabic text steganography using 'Kashida'extensions," presented at the IEEE/ACS International Conference on Computer Systems and Applications, 2009.

- [32] A.-N. Ahmed and G. Adnan, "Exploit Kashida Adding to Arabic e-Text for High Capacity Steganography," presented at the Third International Conference on Network and System Security, Gold Coast, Queensland, Australia, 2009.
- [33] H. Shahreza and M. Shahreza, "Arabic/Persian Text Steganography Utilizing similar Letters With Different Codes," *Arabian Journal for Science and Engineering*, vol. 35, pp. 214-228, 2010.
- [34] M. Memon and A. Shah, "A Novel Text Steganography Technique to Arabic Language Using Reverse Fatha," *Pakistan Journal of Engineering Technology & Science (PJETS)*, vol. 1, pp. 106-113, 2011 2011.
- [35] A. Ali and F. Moayad, "Arabic text steganography using kashida extensions with huffman code," *Journal of Applied Sciences*, vol. 10, pp. 436-439, 2010.
- [36] B. Indradip, B. Souvik, and S. Gautam, "Novel text steganography through special code generation," presented at the International Conference on Systemics, Cybernetics and Informatics, 2011.
- [37] A. Mohiuddin, S. Shahriar, and M. Elias, "Cryptography and State-of-the-art Techniques," *IJCSI International Journal of Computer Science Issues*, vol. 9, pp. 583-586, 2012.
- [38] A. Odeh, A. Alzubi, Q. B. Hani, and K. Elleithy, "Steganography by multipoint Arabic letters," in *Systems, Applications and Technology Conference (LISAT), 2012 IEEE Long Island, Farmingdale State College - State University of New York*, 2012, pp. 1-7.

- [39] A. Odeh and K. Elleithy, "Steganography in Arabic Text Using Zero Width and Kashidha Letters," *International Journal of Computer Science & Information Technology (IJCSIT)*, vol. 4, pp. 12-21, 2012.
- [40] A. Odeh and K. Elleithy, "Steganography in Arabic Text Using Full Diacritics Text," presented at the 25th International Conference on Computers and Their Applications in Industry and Engineering (CAINE-2012), New Orleans, Louisiana, USA, 2012.
- [41] A. Odeh, K. Elleithy, and M. Faezipour, "Steganography in Arabic text using Kashida variation algorithm (KVA)," in *Systems, Applications and Technology Conference (LISAT), 2013 IEEE Long Island*, 2013, pp. 1-6.
- [42] A. Odeh and K. Elleithy, "Steganography in Text by Merge ZWC and Space Character," in *28th International Conference on Computers and Their Applications*, Honolulu, Hawaii, USA, 2013, pp. 1-7.
- [43] A. Odeh, K. Elleithy, M. Faezipour, and E. Abdelfattah, "Novel Steganography over HTML Code," in *New Trends in Networking, Computing, Informatics, Systems Sciences, and Engineering*, T. Sobh and E. Khaled, Eds., ed: Springer, 2014.
- [44] A. Odeh, K. Elleithy, and M. Faezipour, "Steganography in text by using MS word symbols," in *2014 Zone 1 Conference of the American Society for Engineering Education (ASEE Zone 1)*, CT, USA, 2014, pp. 1-5.
- [45] B. J. Mohd, S. Abed, T. Al-Hayajneh, and S. Alouneh, "FPGA hardware of the LSB steganography method," in *International Conference on Computer, Information and Telecommunication Systems (CITS)*, 2012, pp. 1-4.

- [46] E. Gómez-Hernández, C. Feregrino-Urbe, and R. Cumplido, "FPGA hardware architecture of the steganographic context technique," in *18th International Conference on Electronics, Communications and Computers*, 2008, pp. 123-128.
- [47] H. Farouk and M. Saeb, "Design and implementation of a secret key steganographic micro-architecture employing FPGA," in *Proceedings Design, Automation and Test in Europe Conference and Exhibition*, 2004, pp. 212-217.
- [48] H. Leung, L. Cheng, L. Cheng, and C.-K. Chan, "Hardware realization of steganographic techniques," in *Third International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, 2007, pp. 279-282.
- [49] A. F. Mahmood and N. A. Kanai, "An FPGA Implementation of Secured Steganography Communication System," *Tikrit Journal of Engineering Science*, vol. 19, pp. 14-23, 2012.
- [50] G. Gul and A. E. Dirik, "Steganalytic features for JPEG compression-based perturbed quantization," *IEEE Signal Processing Letters*, vol. 14, pp. 205-208, 2007.
- [51] J. Fridrich, M. Goljan, and D. Soukal, "Perturbed quantization steganography," *Multimedia Systems*, vol. 11, pp. 98-107, 2005.
- [52] A. Odeh, K. Elleithy, and M. Faezipour, "Fast Real-Time Hardware Engine for Multipoint Text Steganography," in *2012 IEEE Long Island Systems Applications and Technology Conference*, Farmingdale State College - State University of New York, 2014, pp. 1-6.

- [53] A. Odeh, K. Elleithy, and M. Faezipour, "Fast Real Hardware Engine for ZWC Text Steganography," in *ICAT'14 International Conference on Advanced Technology & Science*, Antalya, Turkey, 2014, pp. 1-5.
- [54] A. Odeh, K. Elleithy, and M. Faezipour, "A Reliable and Fast Real-Time Hardware Engine for Text Steganography," in *IEEE LISAT 2014 Long Island Systems, Applications and Technology*, Long Island, New York, 2014, pp. 1-6.