University of Zawia

Faculty of Science

Department of Computer Science

Text Steganography Method using Font Properties in Tifinagh Language

By

Naser Ashour Shlaky

Supervised by

Prof. **Abduelbaset Mustafa Goweder**

A Thesis Submitted to the Department of Computer Science in Conformity

with the Department Requirements as Partial

Fulfillment for the Degree of Master of Science

April 2025

Acknowledgment

First and foremost, I offer my deepest gratitude to God, the Almighty Allah, for His countless blessings throughout my life and the journey of my research.

I am profoundly grateful to my research supervisor, **Professor Abduelbaset Goweder**, whose patience, insightful feedback, and practical guidance have been invaluable to my thesis. I could not have asked for a more supportive mentor throughout my studies.

My heartfelt thanks also go to the University of Zawia for granting me the opportunity to pursue my postgraduate studies.

Lastly, I would like to extend my sincere appreciation to my parents, my wife, and friends, whose unwavering support and encouragement were instrumental in helping me complete this research within the given timeframe.

To my father and my mother may Allah forgive them

To my wife

To my children: Abdul-Muhaymin, Fatima, Roudina, Ibadah, Rnad, and Yazan

To my brothers and sisters

Table of Abbreviations

Discrete Cosine Transform ……..	DCT
Transmission Control Protocol ……	TCP
User Datagram Protocol ……..…	UDP
Internet Control Message Protocol ……..	ICMP
Internet Protocol ……	IP
Unicode Transformation Format – 8-bit ……..……	UTF-8
Unicode Transformation Format – 16-bit ……..……	UTF-16
Open Systems Interconnection ……	OSI
Similarity of English Font Types ……..……	SEFS
Natural language processing ……	NLP
Peak Signal to Noise Ratio ……	PSNR
Structured Similarity Indexing Method……...……	SSIM
Rivest Shamir Adleman…….……	RSA
Carrier Message ……	CM
Secret Message ……...……	SM
Rich text format …….	RTF

الملخص

في ظل النمو المتسارع لآليات الشبكات التي تتيح نقل كميات ضخمة من البيانات بين المستخدمين عبر مختلف الوسائط، برزت الحاجة الملحة إلى نظم أمنية للحفاظ على خصوصية البيانات بشكل متزايد. وقد تم تقديم تقنيات متنوعة لتشفير البيانات أثناء عملية النقل من أجل تفادي أي نوع من الهجمات المحتملة. تتمثل الهدف الرئيسي لهذه الأطروحة في تطوير منهجية لإخفاء المعلومات (Steganography) تستند إلى خصائص لغة تيفيناغ. ويمكن توظيف هذه المنهجية لتشفير الرسائل (سواء كانت قصيرة أو طويلة) بما يضمن مستوى عالٍ من الأمان عند إرسال البيانات الخاصة. وتنبثق المشكلة من استخدام اللغات الشائعة في تقنيات إخفاء المعلومات، نظراً لكون خصائص هذه اللغات معروفة على نطاق واسع؛ وعليه، فقد تم تطوير العديد من الخوارزميات بناءً على هذه اللغات الشائعة. تقدم هذه الدراسة منهجية مختلفة لإخفاء المعلومات تعتمد على لغات غير شائعة (نادراً ما تم استخدامها من قبل)، ومنها لغة تيفيناغ. تتسم هذه اللغة بوجود العديد من اللهجات المعقدة التي يصعب ترجمتها على غير المتخصصين. وفي هذا الإطار، تم استخدام ملف نصي مكتوب بالنص التيفيناغي كحامل (وسيط) لأي رسالة سرية (مكتوبة بأي لغة بشرية مثل الإنجليزية أو العربية وغيرها). ومن المتوقع ألا يثير الملف النصي الناتج بعد عملية التضمين أي انتباه، وأن يحمل نفس المعنى الذي يحمله الملف النصي الأصلي. تم تنفيذ المنهجيات المقترحة باستخدام لغة البرمجة .VB.net حيث نجحت طريقة الإخفاء المقترحة في تضمين الرسائل السرية المكتوبة باللغتين الإنجليزية والعربية بشكل فعال داخل الرسالة الحاملة المكتوبة بلغة تيفيناغ. كما تمكنت طريقة استخراج (فك الإخفاء) المقترحة من استخلاص الرسالة السرية من الرسالة الحاملة بكفاءة. تم إجراء سلسلة من التجارب لتقييم المنهجيات المقترحة. وقد تم تقييم خوارزمية الإخفاء المنفذة بناءً على مقاييس السعة، والأمان، والمتانة. كما تم إجراء تقييم لخوارزمية الإخفاء المقترحة مقارنةً بخوارزميتين موجودتين (خوارزمية SEFTوخوارزمية إخفاء النصوص العربية)، حيث تم فحص أداء الخوارزميات الثلاث باستخدام عدد كبير من الرسائل الحاملة (الأغلفة) العشوائية. أظهرت النتائج أن متوسط نسب السعة لخوارزمية الإخفاء المقترحة بلغ 30.7% و30.3% للرسائل السرية الإنجليزية والعربية على التوالي. وتفوقت هذه القيم على تلك التي تم الحصول عليها من خوارزمية SEFT (5.80%) وخوارزمية إخفاء النصوص العربية (5.07%). وبذلك، فإن نسبة السعة لخوارزمية الإخفاء المقترحة تتفوق على نظيراتها في الخوارزميتين الأخريين. وفيما يتعلق بالأمان، تفوقت خوارزمية الإخفاء المقترحة على أداء الخوارزميتين الأخريين. علاوة على ذلك، أظهرت خوارزمية الإخفاء المنفذة متانة أكبر مقارنة بالخوارزميتين الأخريين. وبشكل عام، فإن النتائج المستخلصة مشجعة وواعدة.

.

الكلمات المفتاحية – التخفي، التخفي اللغوي، إخفاء النصوص، تغطية النصوص، حماية البيانات، لغة تيفيناغ.

Abstract

With the rapid growth of network mechanisms, where large amounts of data can be transferred between users through various media, the need for secure systems to maintain data privacy has significantly increased. Various techniques have been introduced to encrypt data during transmission to avoid any type of attack. In this thesis, the main aim is to develop a steganography method based on the characteristics of Tifinagh language. This method can be used to encrypt messages (short or long) in a way that ensures high security for sending private data. The problem arose due to the use of common languages in steganography, as the properties of these languages are well-known. Thus, many algorithms were developed based on common languages. This research study proposes a different steganography method based on uncommon languages (rarely and previously used) such as Tifinagh. This language has many complex dialects that are difficult to translate for non-specialists. In this work, a text file written in Tifinagh script was used as a carrier for any secret message (written in any human language such as English, Arabic, etc.). It is expected that the resultant embedded text file would not attract any attention and will carry the same meaning as the original text file. The proposed methods are implemented using VB.net. The suggested hiding method effectively embedded a secret message written in English and Arabic within a carrier message written in Tifinagh. Also, the proposed un-hiding method effectively extracted the secret message out of the carrier message. A series of experiments were carried out to evaluate the proposed methods. The implemented hiding algorithm is assessed through capacity, security, and robustness metrics. An evaluation of the hiding algorithm against two existing algorithms (SEFT and Arabic text concealment) is conducted. We investigated the performance of the three algorithms for many random carrier (covers) messages. The results show that the average capacity ratios of our hiding

algorithm are 30.7% and 30.3% for English and Arabic secret messages respectively. These values are higher than those obtained for SEFT algorithm (5.80%) and Arabic text concealment algorithm (5.07%). The capacity ratio of our proposed hiding algorithm is superior to that of the other two. In the context of security, our hiding algorithm surpasses the performance of the other two algorithms. Furthermore, the hiding algorithm that has been implemented exhibits greater robustness than the other two algorithms. In general, the findings are encouraging and promising.

Keywords— Steganography, Linguistic Steganography, Text Hiding, Text Covering, Data Protection, Tifinagh Language.

Table of Contents

Title	Page
Acknowledgment ……..……	I
Table Of Abbreviations……..	II
Abstract …….....	V
Table Of Contents ……	VI
List of Tables…….	X
Table of Figures ……...……	XI
Chapter 1: Introduction ……	1
1.1 Overview…….....	1
1.2 Justifications for the Choice of Tifinagh Language ……..	3
1.3 Tifinagh Language Characteristics ……..	3
1.4 Problem Statement …….....	6
1.5 Thesis Objectives ……	6
1.6 Research Questions ……	7
1.7 Research Methodology ……...……	7
1.8 Evaluation ……	7
1.9 Thesis Organization ……	8
Chapter2: Background and Literature Review ……	9
2.1 Introduction…….....	9
2.2 Cryptography …….……....	10
2.3 Types of Cryptography …….	10
2.3.1 Symmetric-key cryptography ……	10

Title Page

2.3.2 Asymmetric-key Cryptography ……. 12

2.4 Steganography ……... 13

2.4.1 Text steganography ……... 14

2.4.1.1 Format Based Methods ……... 15

2.4.1.2 Random and Statistical Generation …….. 15

2.4.1.3 Linguistic Steganography ……... 16

2.4.2 Image Steganography …… 16

2.4.3 Audio Steganography ……. 17

2.4.4 Video Steganography …….. 17

2.4.5 Network or Protocol Steganography ……. 17

2.4.6 Water Marking ……. 17

2.4.6.1 Visible Watermarks ……. 17

2.4.6.2 Invisible Watermark…… 18

2.4.6.3 Digital Watermarks…… 18

2.5 Relevant Work ……... 18

Chapter 3: An Overview of Fonts Design and the RSA Algorithm ……. 29

3.1 Introduction …….. 29

3.2 The Font …… 30

3.2.1 Types of Fonts ……. 31

3.2.1.1 Serif Font Type …… 31

3.2.1.2 Sans Serif Font Type …….. 32

Title Page

3.2.1.3 Script Font Type ……. 33

3.2.1.4 Display Font Type ……... 33

3.3 The Influence of Custom Fonts on the Protection of Secret Messages…. 35

3.4 The Font Lab Application ……..... 36

3.4.1 Font Window ……. 36

3.4.2 Glyph Window …… 37

3.4.3 Editing Fonts and their Information …….. 38

3.4.4 The Export Profile…… 38

3.5 RSA Algorithm ……. 39

3.5.1 Justification for Adopting the RSA Algorithm……. 39

3.5.1 Creation of an RSA Public/Private Key Pair …….. 40

3.5.2 The Encryption Equation …….. 41

3.5.3 The Decryption Equation……... 41

3.6 Summary ……..... 42

Chapter 4: Design of the Proposed Methods and Discussion …… 43

4.1 Introduction …… 43

4.2 Research Methodology ……..…… 43

4.3 The Proposed Hiding Method ……. 44

4.4 Discussion of the Proposed Hiding Algorithm ……..... 51

4.5 The Proposed Uncovering (Extraction) Method ……..... 54

4.6 Discussion of the Extraction Algorithm ……. 57

4.7 Summary …….. 58

Title Page

Chapter5: An Implementation and Evaluation of the Proposed ……. 59

5.1 Introduction …… 59

5.2 Implementations of the Hiding and Uncovering Algorithms 59

5.3 An Evaluation of the Designed Algorithms ……. 62

5.4 Summary ……..... 72

Chapter6: Conclusions and Future Works……..... 73

6.1 Conclusions ……..... 73

6.2 The Main Contribution…… 73

6.3 The Distinctiveness of this Work…… 74

6.2 Future Works ……..... 75

6.3 References ……... 76

Appendix A: …… 76

Appendix B: …… 87

List of Tables

Table	Page
Table 1.1: A set of different font Sizes in Tifinagh Language ……..	3
Table 1.2: Fonts that are available in Tifinagh Language ……	3
Table 1.3: A selected set of similar attributes of fonts ……...	4
Table 1.4: The selected fonts of different sizes …….	4
Table 4.1:Sixteen (16) main possibilities for each Tifinagh character ……..	38
Table 4.2: Sixteen (16) sub-possibilities for each Tifinagh character ……...	39
Table 4.3: The binary codes and their associated character properties ……..	40
Table 4.4: The generated binary codes and their associated character properties .	46
Table 4.5: The generated binary codes of the SM associated with their properties .	51
Table 5.1: A comparison of the proposed hiding algorithm with alternative algorithms from the robustness perspective …….	58
Table 5.2: A comparison of our algorithm and alternative algorithms in terms of capacity and safety ratios ……..	61
Table5.3: Detailed comparisons of different algorithms based on Capacity ratio ...	62

Table of Figures

Figure	Page
Figure 1.1: A depiction of ancient writing monuments in Tifinagh language..	2
Figure 2.1: A block diagram of symmetric encryption…….	10
Figure 2.2: A block diagram of Asymmetric Encryption……..	11
Figure 2.3: A block diagram of hash function……..	12
Figure 2.4: A block diagram of text steganography broad categories……...	15
Figure 3.1: The standard character set of a PostScript Type…….	26
Figure 3.2: The main types of fonts…….	27
Figure 3.3: The Serif font…….	28
Figure 3.4: The sans serif font type……..	32
Figure 3.5: The Script fonts…….	29
Figure 3.6: The display fonts……..	29
Figure 3.7: The handwritten fonts……..	30
Figure 3.8: The monospace fonts……..	30
Figure 3.9: The font window…….	31
Figure 3.10: The glyph Window from the catalog……	37
Figure 3.11: The editing fonts and their information……	38
Figure 3.12: The export profile…….	38

Table of Figures

Figure	Page
Figure 3.13: A diagram of the RSA process……..	34
Figure 3.14: The creation of public/private keys using RSA Algorithm……..	35
Figure 3.15: An Implementation of RSA algorithm using VB.Net……...	35
Figure 3.16: The encryption equation for the RSA algorithm……...	36
Figure 3.17: The decryption equation for RSA algorithm…….	36
Figure 4.1: An illustration of 8-bit embedding……..	39
Figure 4.2 : message in Tifinagh……..	41
Figure 4.3: An English translation…….	41
Figure 4.4: The block diagram of the proposed hiding (concealing) method……	42
Figure 4.5: The key steps of the proposed hiding algorithm…….	43
Figure 4.6 : A flowchart of the hiding Algorithm…….	44
Figure 4.7: UTF-8 encoding of the secret message……...	45
Figure 4.8: The encrypted secret message…….	45
Figure 4.9: The carrier message before concealing the secret message…….	47
Figure 4.10 The CM after concealing the SM……...	47
Figure 4.11: A block diagram of the proposed extraction method……	48
Figure 4.12: The proposed algorithm outlined in pseudocode……...	49
Figure 4.13: Flowchart of the Decoding Algorithm……..	50

Table of Figures

Figure	Page
Figure 4.14: The encrypted SM in decimal numbers…….	52
Figure 4.15: The decrypted SM. ……...	52
Figure 4.16: The UTF-8 encoding of the SM……	52
Figure 5.1: A dialog window for the implemented hiding algorithm……	54
Figure 5.2: The process of hiding the SM within the CM……..	54
Figure 5.3: The process of loading the CM embedded with SM……...	55
Figure 5.4: The process of extracting the SM from the CM…….	55
Figure 5.4: Capacity ratios of three different algorithms……...	63
Figure 5.5: Average capacity ratio of three different algorithms……...	63

Chapter 1
Introduction

Chapter 1: Introduction

1.1 Overview

Information can be stored and transmitted over the Internet. The process could lead to hacking, or the information could be obtained by an unauthorized third party. Therefore, it requires encrypting and concealing important and confidential information. Internet users, especially employees in fields such as journalism, documenting violations, and crimes, need some form of concealing their secret messages and confidential reports (Trilling, 2017). Therefore, hiding or encrypting information is an essential value for maintaining users' digital security and protecting the confidentiality and privacy of their information. Text files represent the most challenging covert messages, as they do not contain redundant patterns like other carrier messages. Good embedding algorithms can avoid suspected attackers by applying certain mechanisms before sending the data. The problem arose due to the use of common languages in obfuscation, as the characteristics of these languages are well-known. Thus, many algorithms have been developed based on common languages. In this research proposal, a new method is suggested based on uncommon languages (rarely used in the past), such as Tifinagh. It has many complex dialects that are difficult to translate for non-specialists. In this work, a text file written in Tifinagh script will be used as a carrier for any secret message (written in any human language, such as English, Arabic, etc.). Etc.). The secret message (written in any human language, such as English or Arabic) is embedded in a text file written in Tifinagh, commonly known as the ancient Libyan script. The Tifinagh alphabet, which dates back to around three thousand years BC, is one of the oldest discoveries ever made, as evidenced by the writings and inscriptions found in the Tassili and Akakus mountains

(Blanco, 2018). Figure 1.1 represents the landmarks of ancient writing in the Amazigh language. Research in optical character recognition (OCR) for Amazigh scripts is still in its early stages, although it has significantly advanced for Arabic, Latin, and Chinese scripts (Erritali, 2020). The geometric structure and non-cursive nature of Tifinagh present challenges for generic (Erritali, 2020). The main goal of this work is to develop a hiding algorithm based on font properties in Tifinagh language. This algorithm could be used to encrypt (short or long) messages in a way that ensures high security for sending private data. In this research study, an experimental research methodology would be employed. Experimental and creative research is the foundation of science. An experimental study is primarily concerned with cause and effect. We will design and implement the proposed algorithm in this research using the VB.net programming language. Once the implemented algorithm produces the correct output, a series of experiments would be conducted to gather the obtained results, which would be analyzed and discussed, and any observations would be pointed out.



Figure [1.1]: A depiction of ancient writing monuments in Tifinagh language.

1.2 Justifications for the Choice of Tifinagh Language

This section clarifies the underlying reasons for the selection of Tifinagh Language to serve as a cover for secret information. These reasons are as follows:

1- Cyber Deception: It has been decided to rely on frameworks like MITER Engage, which studies how to use "deceptive assets" (such as rare languages) to distract the attackers and slow down their decision-making speed (López, 2024).

2- Encrypted linguistic analysis (Steganography): our insights were derived from research regarding “Information concealment within texts," particularly studies carried out at universities in Morocco such as University of Mohammed V on the use of "Tifinagh" as a cover for encrypted data due to its geometric structure (Sadki, 2014).

3- Cognitive Load Theory: This theory was adopted in our thesis in which it explains how analyzing an unfamiliar text consumes more mental energy from the attackers, leading them to make mistakes or abandon the goal (Hall, 2025).

4- Recent OCR research indicates that the processing of 'rare scripts' and archaic typefaces continues to present significant obstacles for the advancement of optical character recognition technologies (Erritali , Amrouch, 2020).

5- The analysis of hacking psychology is grounded in the study of cognitive hacker behavior, incorporating Peter Arntz’s (2018) research on security through obscurity and the dynamics of attacker interaction with unfamiliar data (Restor, 2025).

1.3 Tifinagh Language Characteristics

Tifinagh language characterizes the fact that changing the size of the font does not have any noticeable effect. Font sizes such as: (12, 12.25, 12.30, and 12.35) do not affect the appearance of the character. Table [1.1] lists a set of different font sizes in Tifinagh. As

a matter of fact, Tifinagh language, like any other language, contains a number of special fonts that can be designed, as indicated in Table [1.2]. By examining these fonts, it can be observed that some of them have similar attributes, such as: (Nasser_font1, Nasser_font2 .. etc). Unlimited types of fonts could be designed as shown Table[1.3]. For the designed set of fonts, the font style can be changed to produce an italic letter, thus another feature can be added to the previous properties. Different font sizes would be used as given in Table [1.4] based on the combination of font size and type. Two hundred and fifty six (256) possibilities for each Tifinagh character can be created.

Table 1.1: A set of different font Sizes in Tifinagh Language..

Code	Font size	Example
Fs1	12 p	ⵜⵉⴼⵉⵏⴰⵖ ⵙⵉ ⵏ ⵜⵙⴰⵡⴰⵍⵜ ⵙⵉ ⵍⵉⴱⵢⴰ ⵓⵙⵜⴰⵙ,
Fs2	12.25p	ⵜⵉⴼⵉⵏⴰⵖ ⵙⵉ ⵏ ⵜⵙⴰⵡⴰⵍⵜ ⵙⵉ ⵍⵉⴱⵢⴰ ⵓⵙⵜⴰⵙ,
Fs3	12.30 p	ⵜⵉⴼⵉⵏⴰⵖ ⵙⵉ ⵏ ⵜⵙⴰⵡⴰⵍⵜ ⵙⵉ ⵍⵉⴱⵢⴰ ⵓⵙⵜⴰⵙ,
Fs4	12.35 p	ⵜⵉⴼⵉⵏⴰⵖ ⵙⵉ ⵏ ⵜⵙⴰⵡⴰⵍⵜ ⵙⵉ ⵍⵉⴱⵢⴰ ⵓⵙⵜⴰⵙ,
Unlimited fonts		

Table 1.2: Fonts that are available in Tifinagh Language.

Font Name	example
TIFINAGHE AMAWAS	□□□□□□ □□ □ □□□□ □□□□□□ □□ □□□□□ □
TIFINAGHE AMNNAGH	□□□□□□ □□ □ □□□□ □□□□□□ □□ □□□□□ □
TIFINAGHE AMTROY	□□□□□□ □□ □ □□□□ □□□□□□ □□ □□□□□ □
TIFINAGHE AMURAR	□□□□□□ □□ □ □□□□ □□□□□□ □□ □□□□□ □
TIFINAGHE ANAFAW2	□□□□□□ □□ □ □□□□ □□□□□□ □□ □□□□□ □
TIFINAGHE ASFRGH	□□□□□□ □□ □ □□□□ □□□□□□ □□ □□□□□ □
Tamalout Standard UNICODE	□□□□□□ □□ □ □□□□ □□□□□□ □□ □□□□□ □
Tamzwart Standard UNICODE	□□□□□□ □□ □ □□□□ □□□□□□ □□ □□□□□ □
Tassafout Standard UNICODE	□□□□□□ □□ □ □□□□ □□□□□□ □□ □□□□□ □
Teddus Standard	□□□□□□ □□ □ □□□□ □□□□□□ □□ □□□□□ □

UNICODE	
Nasser_font1	ⵣⵙⵍⵄⵙⵔ ⵉⵔ ⵙ ⵛⵉⵜⵏ ⵍⵙⵛⵙⵜⴰ ⵉⵍ ⵍⵔⵜⵙⵔ ⵍⵉⵎⵜⵔ,
Nasser_font2	ⵣⵙⵍⵄⵙⵔ ⵉⵔ ⵙ ⵛⵉⵜⵏ ⵍⵙⵛⵙⵜⴰ ⵉⵍ ⵍⵔⵜⵙⵔ ⵍⵉⵎⵜⵔ,
Nasser_font3	ⵣⵙⵍⵄⵙⵔ ⵉⵔ ⵙ ⵛⵉⵜⵏ ⵍⵙⵛⵙⵜⴰ ⵉⵍ ⵍⵔⵜⵙⵔ ⵍⵉⵎⵜⵔ,
Ammouri Tifinagh	□□□□□□ □□ □ □□□□ □□□□□□ □□ □□□□□ □
Ammouri Tifinagh Outline	□□□□□□ □□ □ □□□□ □□□□□□ □□ □□□□□ □
Ammouri Tifinagh Shadow	□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Table 1.3: A selected set of similar attributes of fonts.

Code	Font Name	example
Ft1	Nasser_font	ⵣⵙⵍⵄⵙⵔ ⵉⵔ ⵙ ⵛⵉⵜⵏ ⵍⵙⵛⵙⵜⴰ ⵉⵍ ⵍⵔⵜⵙⵔ ⵍⵉⵎⵜⵔ,
Ft2	Nasser_font1	ⵣⵙⵍⵄⵙⵔ ⵉⵔ ⵙ ⵛⵉⵜⵏ ⵍⵙⵛⵙⵜⴰ ⵉⵍ ⵍⵔⵜⵙⵔ ⵍⵉⵎⵜⵔ,
Ft3	Nasser_font2	ⵣⵙⵍⵄⵙⵔ ⵉⵔ ⵙ ⵛⵉⵜⵏ ⵍⵙⵛⵙⵜⴰ ⵉⵍ ⵍⵔⵜⵙⵔ ⵍⵉⵎⵜⵔ,
Ft4	Nasser_font3	ⵣⵙⵍⵄⵙⵔ ⵉⵔ ⵙ ⵛⵉⵜⵏ ⵍⵙⵛⵙⵜⴰ ⵉⵍ ⵍⵔⵜⵙⵔ ⵍⵉⵎⵜⵔ,
. . . **Unlimited fonts**		

Table 1.4: The selected fonts of different sizes.

Code	Font size	Example	Font name
Fs1+Ft 1	12	ⵣⵙⵍⵄⵙⵔ ⵉⵔ ⵙ ⵛⵉⵜⵏ ⵍⵙⵛⵙⵜⴰ ⵉⵍ ⵍⵔⵜⵙⵔ ⵍⵉⵎⵜⵔ,	Nasser_font
Fs2+Ft 2	12.2 5	ⵣⵙⵍⵄⵙⵔ ⵉⵔ ⵙ ⵛⵉⵜⵏ ⵍⵙⵛⵙⵜⴰ ⵉⵍ ⵍⵔⵜⵙⵔ ⵍⵉⵎⵜⵔ,	Nasser_font1
Fs3+F t3	12.3 0	ⵣⵙⵍⵄⵙⵔ ⵉⵔ ⵙ ⵛⵉⵜⵏ ⵍⵙⵛⵙⵜⴰ ⵉⵍ ⵍⵔⵜⵙⵔ ⵍⵉⵎⵜⵔ,	Nasser_font2
Fs4+F t4	12.3 5	ⵣⵙⵍⵄⵙⵔ ⵉⵔ ⵙ ⵛⵉⵜⵏ ⵍⵙⵛⵙⵜⴰ ⵉⵍ ⵍⵔⵜⵙⵔ ⵍⵉⵎⵜⵔ,	Nasser_font3

1.4 Problem Statement

Computer networks allow the transfer of large amounts of data between users over different media. This transfer has significantly led to the need for secure systems to protect data privacy. Various techniques have been proposed to encrypt transferred data to avoid any kind of attack. One of these techniques is called steganography, in which data is hidden inside other data. Data is hidden in carrier files that anyone can see, but the hidden data cannot be easily discovered. Good steganography algorithms can avoid raising suspicions among attackers by applying certain mechanisms before sending data. The problem arises due to common language usage in steganography, where the characteristics of such languages are well known. Thus, many algorithms are developed based on common languages. In this research, a new method is proposed that is based on uncommon (previously and rarely used) languages such as Tifinagh. It has many complex dialects that are difficult to translate for non-specialists. In this work, a text file that is written in Tifinagh will be used as a carrier of any secret message (written in any human language, such as English, Arabic, etc.).

1.5 Thesis Objectives

The main goal of this work is to develop a hiding algorithm based on font properties in Tifinagh Language. This algorithm could be used to encrypt (short or long) messages in a way that ensures high security for sending private data.The stages of this thesis to accomplish the main objective can be summarized as follows:

1- Design and implement an algorithm to embed and extract a secret message.

2- Test the two proposed algorithms for correctness.

3- Evaluate the performance of the proposed algorithms in terms of efficiency and effectiveness.

1.6 Research Questions

This thesis attempts to answer the following three questions:

1- How can the proposed algorithm provide a safe specificity to transmit information?

2- What is the effect of this algorithm on information security?

3- How can the proposed algorithm improve the security by avoiding drawbacks of some existing algorithms?

1.7 Research Methodology

Research methodology is a collection of systematic research techniques. This simply implies a guide to research and how it is carried out. There are several approaches. In this research study, an experimental research methodology would be employed. Experimental and creative research is the foundation of science. Experimental study is primarily concerned with cause and effect. In this research, the proposed algorithm would be designed, implemented using VB.net programming language. Once, the implemented algorithm produces the correct output, a series of experiments would be conducted to gather the obtained results which would be analysed, discussed, and any observations would be pointed out.

1.8 Evaluation

The designed and implemented algorithms would be evaluated by conducting a set of experiments using different sizes of secret and carrier messages. The evaluation of the proposed algorithms could be performed in three ways, these are:

1- A comparison of the obtained results with manually generated results as well as a comparison of our results with other results produced by any similar existing algorithms for hiding data.

2- An assessment of the efficiency of the proposed algorithms using a mathematical analysis.

3- An assessment of the possibility of decrypting a set of secret messages by sending carrier messages to a group of computer researchers and hackers.

1.9 Thesis Organization

This thesis contains six chapters. The reminder of this thesis can be summarized as follows: Chapter two presents a background and literature review. Chapter three explains the font design application and the RSA encryption algorithm. Chapter four describes the implementation of the proposed algorithms and discusses the obtained results. Chapter five covers an evaluation of the proposed algorithms. Conclusions and future work recommendations are given in Chapter six.

Chapter 2
Background and Literature Review

Chapter 2: Background and Literature Review

2.1 Introduction

The world of information that exists now is digital. These days, data transmission across an insecure channel is a big cause for concern (Kumar, 2025). Additionally, hackers are becoming more widespread and active online at the same time. Therefore, some security precautions must be taken to prevent theft of the confidential data (Trilling, 2017). Different methods have been used to encrypt and decrypt the secret data in order to keep it secret. The two most well-known methods among these are cryptography and steganography (Kumar, 2025). However, the combined effect of these two approaches is greater than either one working alone. "Stegano" and "graphy" combine to form the Greek word "steganography." "Steganography" is the term for hidden writing, where "stegano" means hidden and "graphy" means writing (Kumar, 2025). Data communication can be concealed through the use of steganography. While cryptography can transform a secret communication into a format that is not accessible by humans, this method has the drawback that everyone can see the encrypted message (Drummond, 2024). Hackers may attempt to employ hit and trial (trial and error) approach over the Internet in order to obtain the secret message. Steganography addresses cryptography's drawback by disguising the existence of transmission. The secret message in steganography is concealed in formats other than the original text, image, video, and audio (Kumar, 2025). These two methods differ from one another, and each has a special meaning.

2.2 Cryptography

Cryptography is the study and practice of techniques for secure communication in the presence of third parties called adversaries. It deals with developing and analyzing protocols that prevent malicious third parties from retrieving information being shared between two entities thereby following the various aspects of information security. Secure Communication refers to the scenario where the message or data shared between two parties cannot be accessed by an adversary (Drummond, 2024). In Cryptography, an Adversary is a malicious entity, which aims to retrieve precious information or data thereby undermining the principles of information security (Drummond, 2024). Data Confidentiality, Data Integrity, Authentication and Non-repudiation are core principles of modern-day cryptography (Omar , Asaad , 2017). In general terms, encryption refers to the challenge of enabling two parties to communicate confidentially while an eavesdropper is present. (Boneh , Shoup, 2023).

2.3 Types of Cryptography

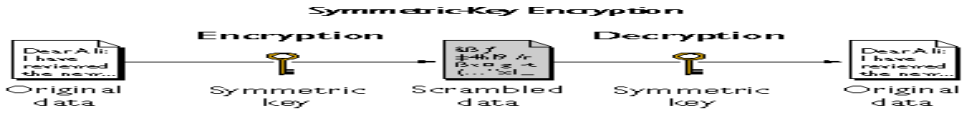
There are various forms of cryptography, and each has special characteristics and uses. Among the most popular varieties of cryptography are:

2.3.1 Symmetric-key cryptography

One of the techniques in cryptography is symmetric key algorithms (Drummond, 2024). This occurs when the shared secret keys used for encryption and decryption are identical (Drummond, 2024). The secret can be generated randomly, through a password, or through a Diffie-Hellman key exchange process. Prominent examples of symmetric algorithms include the Advanced Encryption Standard (AES), which is the current standard, and older algorithms such as the Data Encryption Standard (DES) and Triple

DES (3DES) (Boneh , Shoup, 2023). Symmetric key algorithms operate on computers faster than public key algorithms, making them extremely essential. With public key encryption (also known as asymmetric key encryption), anyone can send you encrypted messages, and the encryption key can be freely distributed to the entire public. Since the encryption key cannot be used to obtain the decryption key in practice, it is considered an "open" key. Although this is extremely useful, public key encryption techniques are limited to sending secret keys because they are very slow on computers. For everything else, symmetric encryption methods are used due to their faster nature. Figure 2.1 illustrates a diagram of symmetric encryption.

“In practice, one often uses a symmetric key to encrypt a single message. The key is never used again. For example, when sending encrypted email one often picks an ephemeral key and encrypts the email body under this ephemeral key. The ephemeral key is then encrypted and transmitted in the email header. A new ephemeral key is generated for every email” (Omar, 2017).

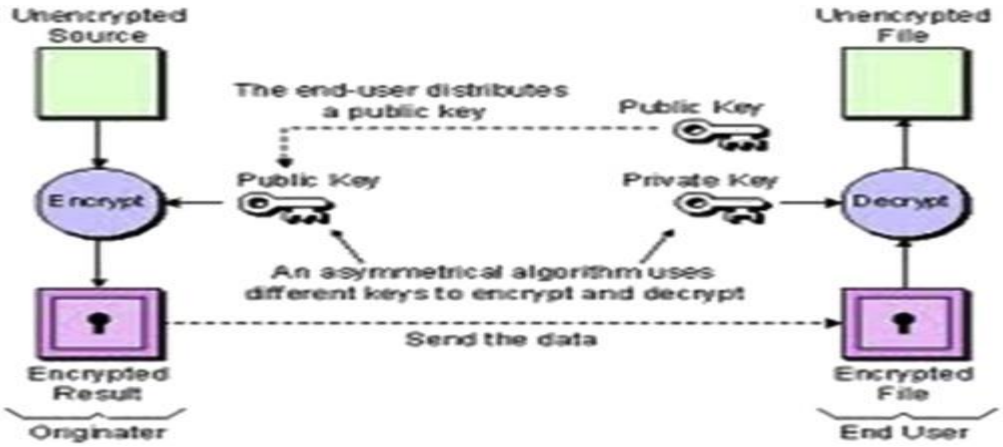


. Figure 2.1: A block diagram of **symmetric encryption.**

\

2.3.2 Asymmetric-key Cryptography

Asymmetric encryption is often used to exchange the secret key in preparation for using symmetric encryption to encrypt data. In the case of key exchange, one party creates the secret key and encrypts it using the recipient's public key (Omar, 2017). Then the recipient decrypts it using their private key. The rest of the communication will be conducted using the secret key as the encryption key. Some examples of public key encryption systems are Elgamal, RSA, Rabin, DSA (Digital Signature Algorithm), and Elliptic Curve Cryptography (ECC), which relies on the algebraic structure of elliptic curves over finite fields (Omar, 2017). Asymmetric encryption is used in key exchange, email security, web security, and other encryption systems that require key exchange over a public network. Figure 2.2 illustrates a block diagram of the asymmetric encryption algorithm that uses a different key for encryption and decryption. "Public-key encryption is sometimes called asymmetric encryption to highlight that the encryption process uses one key, the public key (pk), while the decryption process uses a different key, the secret key (sk)." (Boneh , Shoup, 2023)



. Figure 2.2: A block diagram of Asymmetric Encryption.

2.3.3 Hash functions

Hash functions are extremely useful and appear in almost all applications of information security. A hash function is a mathematical function that transforms a numerical input value into another compressed numerical value (Deena and Rajendra, 2021). The inputs of the hash function are of random length, but the outputs are always of fixed length. Cryptographic hash functions are designed to be secure and they are used in cryptography. Examples include MD5, SHA-1, and SHA-256 (Deena and Rajendra, 2021). The values returned by the hash function are called message digests or simply hash values. Figure 2.3 illustrates a block diagram of hash functions.

"Hash constructions are used in cryptographic algorithms from very long. Features of Hashes that gives the applications the confidence to use them in security methodologies." (Deena , Rajendra, 2021)

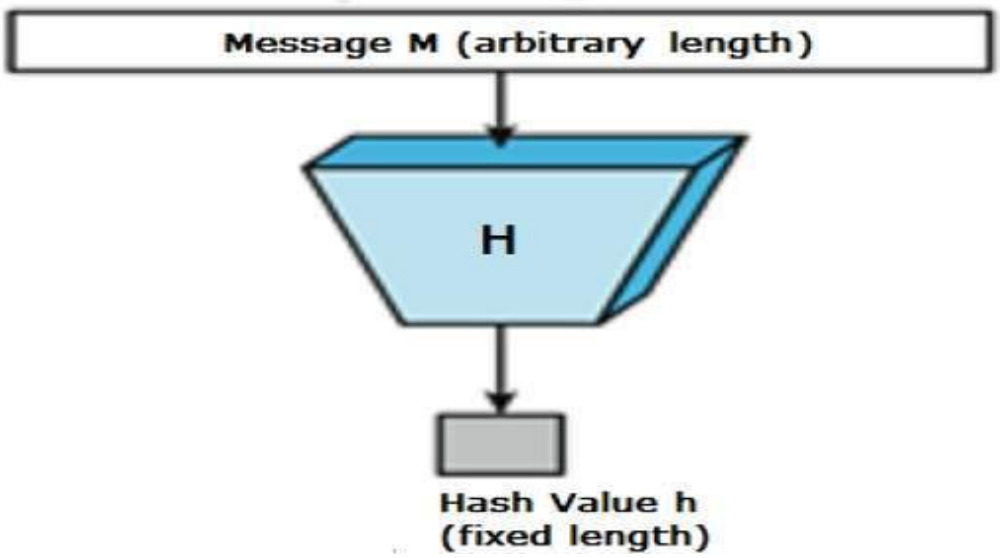


Figure 2.3: A block diagram of hash function.

2.4 Steganography

Steganography is the art of concealing not only information but also the element of secret information transmission. Steganography conceals the secret information in another file in such a way that only the recipient is aware of its existence (Majeed et al.,

2021). In the past, data was safeguarded by concealing it on the back of wax, scripting tables, rabbit stomachs, or slave scalps (Majeed et al., 2021). However, some people today use the medium to transmit information in the form of text, images, video, and audio. It is possible to securely transmit confidential information using multimedia objects such as audio, video, and images as cover sources to hide the information. (Majeed et al,2021). The following are the various types of steganography:

2.4.1 Text steganography

The technique of hiding a secret text message as a covering message or constructing a cover message linked to the original hidden message is known as text steganography (*Odeh, et al*, 2013). Text steganography can involve anything from altering a text's formatting to making random character sequences, modifying words within a text, or using context-free grammars to make messages readable (*Odeh, et al*, 2013). Text steganography is considered the most challenging due of the lack of duplicate data seen in image, audio, and video files. Text documents work in the same way as everything they can identify, but other sorts of documents, such as pictures, have a distinct structure from what they can identify (Majeed et al., 2021). As a result, it can conceal information in these kinds of documents by observing modifications to the document's structure without noticeably altering the output in question. Images and audio files may undergo unnoticeable alterations, but in text files, a simple addition of a letter or punctuation mark may catch the eye of an inexperienced reader. It is better than other stenographic techniques since it can save text files with less memory usage and has faster and easier communication. “Text steganography is considered exceedingly difficult due to the inadequate redundant data in textual files compared with other digital media, such as

audio, image, or video files" (Majeed et al., 2021). Three broad categories can be used to characterize text steganography: format-based, random and statistical generation, and linguistic methods (Majeed et al., 2021). These methods are explained as given in the following sub-sections. Figure 2.4 shows a block diagram of text steganography broad categories.

2.4.1.1 Format Based Methods

Format-based steganography modifies certain elements of a chosen cover text, such as spelling or punctuation, to conceal information.(Majeed et al., 2021) Commonly, non-printing characters and white space can hold more information. Additionally, alter the font's type and color to make it undetectable to outside observers. "In this form of steganography, the physical features of text symbols are used. The features are altered in such a manner that the human eye cannot sense them" (Majeed et al., 2021).

2.4.1.2 Random and Statistical Generation

Random and Statistical Generation: To avoid matching a known plaintext, steganography methods can generate their own cover texts in Random and Statistical Generation (Majeed et al., 2021). Data can be hidden using a random viewing sequence of characters as one technique. Another approach creates words that will occur to have similar statistical attributes as actual words in the given language by using the statistical features of word length and letter frequencies. "The statistical characteristics of a language are obtained and then employed to generate cover text." (Majeed et al., 2021).

2.4.1.3 Linguistic Steganography

The goal of linguistic steganography is to conceal data within text written in natural language. Synonym substitution is one of the main changes utilized in linguistic steganography. The usefulness of this strategy has, however, not received much attention in research to yet. “This technique uses linguistic steganography for hiding secret information inside text files.” (Majeed et al., 2021).

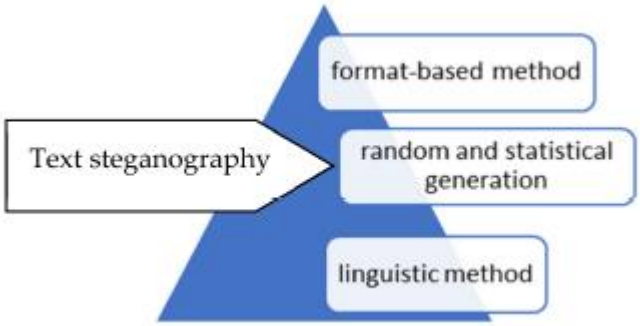


Figure 2.4: A block diagram of text steganography broad categories.

2.4.2 Image Steganography

Image steganography is the ability to conceal information by using a cover object as an image. Pixel intensities are used to conceal data in image steganography. Because an image contains multiple bits in its computer description, images are widely used as a cover source in digital steganography. Commonly used techniques and evaluation metrics include LSB (Least Significant Bit): A popular spatial domain method that replaces the least significant bits of image pixels with secret data bits (Kumar, 2018).

2.4.3 Audio Steganography

Audio Steganography is the technology of embedding information in an audio channel. It can be used to secure digital copyright. Watermarking is a method of concealing one piece of information (message) within another element of information (carrier). It is commonly used in applications such as audio clips.

2.4.4 Video Steganography

It is a method of concealing documents or information in computer video format. In this method, video (a series of images) can be used as a carrier to conceal information. Generally, discrete cosine transform (DCT) inserts values (such as 8.667 to 9) that can be used to conceal data in each image in the video that is unnoticeable to the human eye. Video steganography employs the H.264, Mp4, MPEG, and AVI formats.

2.4.5 Network or Protocol Steganography

It entails concealing information by using network protocols such as TCP, UDP, ICMP, and IP as cover objects. There are covert channels in the OSI layer network model that can be used with steganography.

2.4.6 Water Marking

A watermark serves as a method for incorporating identifying information (a watermark) into both physical and digital media to ensure its authenticity, identify its owner, or prevent unauthorized use (Sadeghi, 2019).

2.4.6.1 Visible Watermarks

Visible watermarks represent the predominant category of watermark and are frequently employed to discourage unauthorized utilization of content (Shelke et al., 2018).

2.4.6.2 Invisible Watermark

Invisible watermarks are frequently used to monitor the ownership of content and are more challenging to detect. They are usually hidden from view and incorporated into the text itself. The ownership and mobility of content on the Internet can be monitored using invisible watermarks. They are helpful for protecting copyright and utilizing specialized software to locate the source of unauthorized copies (Shelke et al., 2018).

2.4.6.3 Digital Watermarks

Digital watermarks, which are difficult to remove or change, are used to safeguard digital assets, including photos, movies, and music. They can be used to monitor ownership, safeguard content, and stop unauthorized usage. They are usually integrated within the content itself. Additional details, such the creation date and time, can also be included (Shelke et al., 2018).

2.5 Relevant Work

In this section, some recent research studies are presented and discussed.

1. Alshahrani (2017) has proposed an improved technique for Arabic text steganography. This technique hides an Arabic text inside another which is based on a hybrid approach. Both Kashida and Arabic diacritics are used in his research work. In this improved technique, the secret message is divided into two parts, the first part is to be hidden by the Kashida technique, and the second is to be hidden by the diacritics or Harakat technique.

- The results and evaluation:

It is not affected by printing, using optical character recognition methods, changing

fonts, or rewriting, as long as the medium can display the Arabic language. This method provides the highest capacity. And it is more secretive because it is uncommon nowadays to send a text with diacritics. It relies on ASCII code because it is more secure (Alshahrani, 2017).

- The critique concerning:

Arabic text that has been thoroughly diacritical must be used as the cover medium. If the cover language draws the reader's attention with religious or political texts, it is inappropriate. Additionally, utilizing the same fixed line is necessary. Furthermore, in comparison to our search technique, there are very few concealed characters.

2. Al-Anzi et al (2020) propose a unique method for using Unicode to hide hidden bits in the Arabic language. Their technique relies on hiding some important details using the contextual shape of the Arabic letter. To increase capacity while maintaining security, additional characters such as the zero-width non-joiner, the kashida, and the joiner are used. Based on their experimental results, the strategy performs better in terms of performance metrics related to capacity and security compared to most current solutions. In addition, their method can be widely used in any language that uses Arabic script, such as Urdu and Persian, as well as many other languages that benefit from Unicode, which is a character encoding standard that can be used for most writing systems in the world.

- The results and evaluation:

They discovered that the capacity measurement given in the utilized equation is the best choice and best suits their plan. It was discovered that the equation worked well for calculating their method's capacity estimation ratio and contrasting it with that of other,

related techniques. The security measure needs to be discussed since the goal of the security assessment is to guarantee that private information cannot be discovered by a third party, i.e., when secret messages are being sent over an open network. When security is assumed to be compromised, it is weak to the extent that it can be pushed to its limits. To assess the security of all approaches, the visible and unseen additional characters are measured after concealment (Al-Anzi et al, 2020).

- The critique concerning:

The choice of encoding format depends on the requirements in each scenario. For example, in a general context where all characters are of equal importance, UTF-8 should be used, as it is more flexible in terms of the number of bytes consumed per character . They used UTF-16 because they needed to convert a character from the general form of Unicode to a contextual form, whereas our study does not require conversion. The security level is lower if the extra characters in the secret text are more prominent than the characters in the original text. In our studies, there are no extra characters . And this confirms the aforementioned logic that high security is achieved by not introducing any extra characters to conceal sensitive information.

3. Odeh et al (2013) have proposed a research paper which uses remarks from right to left (U200F) and remarks from left to right (U200E) in hiding.

- The results and evaluation:

Different methods have been proposed to hide information within text documents. While some of these techniques are exclusive to certain languages, others are applicable to all languages. In his research, he proposed an approach that uses comments (right, left) to hide data within text files in any language. We pass two figs in each symbol using his

approach. Additionally, he presented improvement strategies that can be applied (Odeh et al, 2013)

- The critique concerning:

On the other hand, hidden data determines the file size and can grow significantly. To solve the file size issue, I recommend using alternative strategies. Moreover, while our technology can hide eight bits in a single character, it is possible to hide only two bits in each character.

4. Ali and Seddike (2013) have worked on an approach which employs a text file as a carrier for the secret message data in such a way that the resulting text file is undetectable and retains the same meaning as the original file. Where the size of the letters used to include the hidden message is altered.

- The results and evaluation:

Instead than employing extensions or the spaces between words or paragraphs, a novel method of text concealing has been proposed that makes use of the cover file's characters. Unlike gaps between words, which require seven words to conceal a character, the suggested solution just requires seven letters. Thus, in comparison to other approaches, it retains a lot more data while preserving the text's exact meaning and producing these amazing lines. Ultimately, we can state that this technique can be applied effectively and has attained its encryption objectives (Ali and Seddike, 2013).

- The critique concerning:

This study claimed that one letter may be concealed among seven English letters, which is a far lower minimum than keeping one letter for each English letter. Furthermore, this method is limited to Latin scripts. The Arabic and Persian scripts are not feasible.

5. El Rahman et al (2019) have published an article which discusses two steganography methods based on English font style similarity (SEFS). To incorporate the secret message after encrypting it, the primary document font style is substituted with a similar font style. This is accomplished using either: 1) the carrier document's upper-case letters and punctuation marks, or 2) the white space between words, as well as the start and end letters of any word with more than two letters in the carrier document. These methods are put to the test by being applied to a variety of document formats and font styles.

- The results and evaluation:

The suggested technique for text hiding was put to the test by concealing the same secret message in some of the cover papers made using different font kinds. To conceal one letter from the hidden message (one symbol written in three capital letters) we need three letters. In other words, we can conceal two characters in the cover file if it has six characters. When the covered and hidden document sizes were compared, it was found that the hidden document's average size had grown by around 0.766% from its original size after concealment (El Rahman et al, 2019).

- The critique concerning:

This study found that, as opposed to keeping one letter for each English letter, one letter may be buried among three English letters, which is a far lower minimum. Furthermore, only Latin characters may be used with this technique. Persian and Arabic scripts are not suitable. The size of the delivered message also increases as a result of the various font kinds.

6. Rahma et al (2013) have published a novel method which is proposed for data hiding in English scripts using Unicode of English alphabet in another languages. In this

method, 13 characters from English alphabet were chosen for hiding process which have appearance in another languages. can be suggested for data compression to overcome this drawback. Thus, the proposed method provides an effective way to use text obfuscation through Unicode, and an excellent method for achieving a secure and reliable transformation of information.

- The results and evaluation:

The proposed method has excellent perceptual transparency because the hidden text is similar to the original text using multilingual characters that resemble and fit the symbols of the original text. The concealment capability of the proposed method is very high and depends on the frequency of the selected characters as shown in Table 2. Additionally, this method is robust against digital copy-paste operations, meaning that copying and pasting text between computer programs preserves the hidden information. On the other hand, there is an increase in the size of the stego document, due to the use of Unicode which uses 2 bytes per character. Huffman coding (Rahma et al, 2013).

- The critique concerning:

Depending on the character count that is supplied, certain bits can be concealed. We can hide 400 bits in the cover file if it has 200 chosen characters, for example, because two bits might be hidden in a single character. It is evident that the stego document's average size (after concealing) has grown by around 11.1% when compared to its initial size. This is because the hiding method uses Unicode, which employs UTF-16 encodings for every character, rather than ASCII code. As opposed to the method utilized in this study, which uses UTF-8 encodings and ASCII for the concealing procedure, which may conceal about 8 bits per character.

7. Khekan et al (2021) have published a paper which describes a method of hiding data that uses dots, more than half of the Arabic characters contain dots. Several characters have upper dots and others have lower dots. Some have one dot, others have two dots. Few have even three dots. In this paper, they have used the dots of characters to embed an English secret message.

- The results and evaluation:

This method proved that it could successfully conceal a number of characters without the text having any errors or doubts. The total capacity to conceal and the possibility of achieving great performance while preserving the letter frequencies in accordance with what is there and preventing textual distortions that might result in detection by technological or visual means (Khekan et al, 2021).

- The critique concerning:

The method of compressing 8 bits to 5 bits means that some of the 256 combinations are not used. By creating a map of the patterns. On one hand, it only uses the concealment of English letters, and its method is somewhat dubious. It was mentioned that the concealment method involves counting the dots of the letters in a single word of the transmitting message. Knowing that the number of these dots is even means that a secret bit can be embedded in it. The amount of the secret bit to be concealed will be done by resorting to a synonym, where the word is replaced with another word that carries the same meaning. In case it is not possible to find a word close in meaning and containing the same number of dots in its letters to equal the value of the bit to be concealed, the word is neutralized.

8. Ahvanooey et al (2019) have presented an overview of state of the art of the text hiding area, and provided a comparative analysis of recent techniques, especially those focused on marking structural characteristics of digital text message/file to hide secret bits.

- The results and evaluation:

This case study compares several text concealing strategies currently in use, with a particular emphasis on those that alter the structural properties of digital text messages and files. To clarify the situation, various essential standards, applications, and attacks related to the field of text hiding were reviewed, as well as the current security challenges facing the cybersecurity sector. Additionally, three main types were outlined. Several text concealing methods that rely on handling cover text messages and files in order to implant the secret bits—that is, linguistic, random, structural, and statistical bits. Next, the constraints and features of every group were described to demonstrate their effectiveness in diverse applications. Additionally, the recently suggested methods regarding the essential criteria were assessed to emphasize their advantages and disadvantages. Lastly, a few guidelines were suggested (Ahvanooey et al, 2019).

- The critique concerning:

No criticisms about the study.

9. Ching-Yun et al (2013) have used linguistic steganography to hide information in natural language text. One of the major transformations used in linguistic steganography is synonym substitution.

- The results and evaluation:

Research in linguistic steganography requires experience and knowledge of both

information security and NLP. In addition, due to the complexity of language, there have been more studies on image or video steganography than linguistic steganography. Hence linguistic steganography is a relatively new research area, and further efforts are needed to develop more secure and efficient systems. The novel and original ideas provided in this article can benefit research in both computational linguistics and information security. It is hoped that our work can form the basis for more research devoted to linguistic steganography (Ching-Yun et al, 2013).

- The critique concerning:

No criticisms about the study.

10. Kumar (2018) has published a paper which illustrates various techniques used in steganography and implemented these techniques. A comparative analysis is performed between various stenographic tools using the sample test images and test data. The quality metrics such as PSNR and SSIM are calculated for the final output images which are used for rating the tools. This paper also discusses the Steganalysis which is known as the process of identifying the use of steganography.

- The results and evaluation:

Digital watermarks give content producers a means of identifying the owners of their works and safeguarding them against unauthorized dissemination. Laws limiting steganography may also be implemented, given that governments have already asserted that criminals utilize these methods for communication (Kumar, 2018).

- The critique concerning:

No criticisms about the study.

11. Curran (2018) have examined the leading methods of text based steganography. They evaluated a variety of stenographic techniques including open space encoding, synonym replacement, UK/US English translation algorithm and Wayner's Mimic Functions using five benchmarks which compare speed, capacity, complexity, compromisability, and size. They have found that the best methods to hide information should not use a single scheme, but a hybrid of many schemes. In order to further hide information, text should be compressed, encrypted and then hidden in a cover document.

- The results and evaluation:

Different algorithms related to text steganography were studied, and four of them were also implemented. This research initially included an investigation into various methods within the field of embedding as a whole, and then delved specifically into text-based embedding. The study concluded that steganography is not limited to embedding only, but it can also perform encryption and compression as part of a hybrid system. This will make steganography much more secure as the data is hidden, but if it is discovered, it remains encrypted (Curran, 2018).

- The critique concerning:

We support the same perspective as the findings of this study, and our research also made use of encryption.

12. Alabish, et al (2013) have presented English as a representative case of natural languages to investigate the application of synonym-based steganography within general texts. The primary goal is to develop a general framework for verbal steganography that supports various natural languages, aiming to improve encoding efficiency by reducing

the number of required bits and increasing information capacity. An evaluation of the proposed method was conducted, resulting in encouraging and promising outcomes.

- The results and evaluation

Empirical evaluation against standard ASCII demonstrated that the proposed 5-bit encoding scheme reduces bit-length by 37.5%, thereby significantly enhancing embedding capacity. Subjective assessments further confirmed the high quality and naturalness of the generated text, validating the method's effectiveness in maximizing information concealment while minimizing encoding costs (Alabish et al, 2013).

- The critique concerning:

The proposed method is limited by insufficient synonyms, forcing a cyclic reuse of words that creates ambiguity, as the same term can represent different letters based solely on sequence. This reliance on order makes the system fragile and susceptible to complete decoding failure upon any text modification. Furthermore, the claim of universality is questionable, particularly for complex languages like Arabic, where grammatical constraints hinder effective synonym substitution.

Chapter 3
An Overview of Fonts Design and the RSA Algorithm

Chapter 3: An Overview of Fonts Design and the RSA Algorithm

3.1 Introduction

This chapter explores two main components: a font design applications specifically developed for Tifinagh language characters, and an examination of the RSA encryption algorithm. Typography principles have endured across centuries, and despite technological evolution, their primary objective remains unchanged: delivering aesthetically pleasing text that enhances the reading experience. A font encompasses a collection of printable or displayable characters featuring consistent style and size characteristics. These font styles are applicable across both printed materials and digital platforms. Typographic font styles incorporate specific spacing and design elements that uniformly apply to all characters, including upper and lowercase letters and various punctuation symbols. The overall design concept for a font collection is known as a typeface, with its various iterations forming what's called a font family. The RSA encryption algorithm, representing a significant advancement in cryptography, functions as an asymmetric or public-key encryption system. This method safeguards confidential information through encryption and decryption processes utilizing paired private and public keys. The algorithm emerged in 1977, developed by three researchers at the Massachusetts Institute of Technology - Ron Rivest, Adi Shamir, and Leonard Adelman - whose surnames' initials form the acronym RSA. The system operates on a dual-key principle: a confidential private key retained exclusively by the key pair creator, and a publicly accessible key available to all users. The encryption process can employ either key, while the corresponding key serves for decryption purposes.

3.2 The Font

The term "Font Type" encompasses all elements related to visual text, although it initially described the physical wood or metal pieces used to form letters. Typography is the term employed by designers to denote the style and visual presentation of printed text. Designers may focus on typography, which involves the art of visually arranging and designing text. A typeface, or font family, consists of an alphabet designed with consistent characteristics across all letters and symbols. Examples of typefaces include Times, Arial, and Minion. The term "font" specifically refers to a particular style within a typeface. For example, Arial Black is a font within the Arial typeface, while Minion Pro Italic is a font from the Minion font family. The typeface contains all necessary information for positioning and visualizing the characters it represents. (Naskar,2019). Figure 3.1 displays the standard character set of a PostScript type.



Figure 3.1: The standard character set of a PostScript Type.

3.2.1 Types of Fonts

Fonts are omnipresent in our everyday experiences and play a crucial role in branding and communication. They are visible in numerous contexts, ranging from business cards to large-scale advertisements. Despite the vast array of fonts accessible on the internet, the fundamental classifications of fonts are confined to six primary categories. The main types of fonts are illustrated in Figure 3.2.(Sahni,2023).

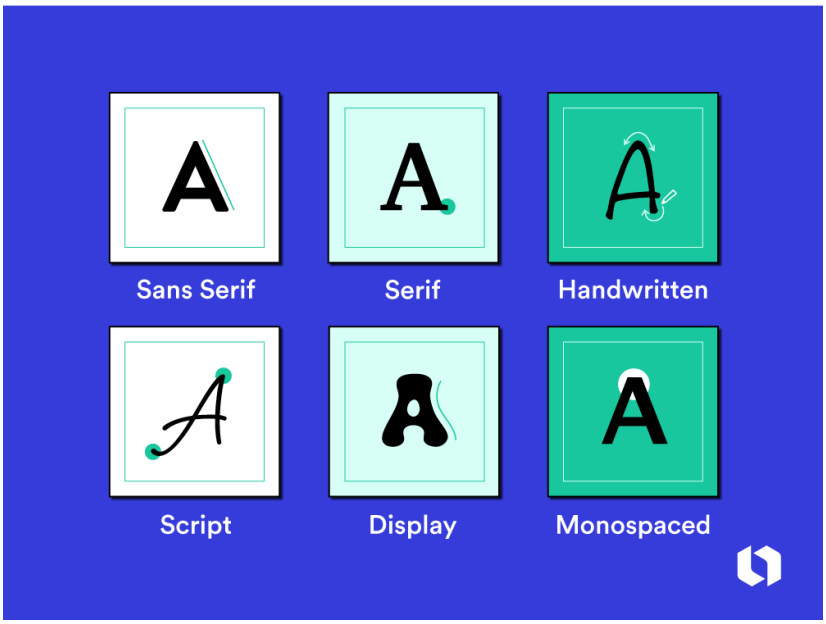


Figure 3.2: The main types of fonts.

3.2.1.1 Serif Font Type

Typefaces characterized by decorative strokes at the ends of their letters, referred to as serifs, are classified as serif fonts. There exists a wide variety of serif typefaces, each differing in the shape, thickness, and length of their serifs. Notable examples of serif fonts include Garamond and Times New Roman. The extensions at the ends of a

character are specifically termed serifs.(Sahni , 2023).Figure 3.3 illustrates the Serif font type.



Figure 3.3: The Serif font.

3.2.1.2 Sans Serif Font Type

The term "sans serif" refers to typefaces that do not feature serifs. Many brands favor sans serif fonts due to their broad availability and versatility across various applications. Illustrated in Figure 3.4 is an example of a sans serif font. These typefaces are characterized by a clean, modern aesthetic and the absence of serifs. Their enhanced readability on screens makes them a popular choice for digital media, including mobile applications and websites (Sahni,2023).



Figure 3.4: The sans serif font type.

3.2.1.3 Script Font Type

Cursive or script typefaces evoke the historical art of calligraphy. These typefaces are crafted to mimic individual handwriting, resulting in designs that are both artistic and distinctive. Script fonts replicate the fluidity of cursive writing, presenting a refined aesthetic, (Sahni,2023) as illustrated in Figure 3.5.



Figure 3.5: The Script fonts.

3.2.1.4 Display Font Type

The phrase "display fonts" denotes visually appealing typefaces that are commonly employed in larger formats or digital presentations. These fonts are typically utilized for titles and headings due to their design, which aims to capture the viewer's attention (Sahni,2023). Figure 3.6 illustrates examples of display fonts.



Figure 3.6: The display fonts.

- **Handwritten fonts**

Handwritten fonts, as indicated by their designation, replicate the visual characteristics of handwritten text. These fonts are derived from authentic hand lettering, imparting an

organic, imperfect, and individualized aesthetic. The handwritten fonts illustrated in Figure [3.7] serve to enhance the personal dimension of designs or brand logos, as they emulate the look of handwriting and bear a resemblance to handwritten calligraphy (Sahni,2023).



Figure [3.7] The handwritten fonts.

- **Monospaced fonts**

Monospaced fonts, often known as typewriter fonts, are characterized by uniform character spacing. This is illustrated in Figure [3.8]. Due to the fixed width of monospace typefaces, each character occupies an equal amount of space. (Sahni,2023).



Figure [3.8] The monospace fonts.

3.3 The Influence of Custom Fonts on the Protection of Secret Messages

The idea of designing custom fonts involves adding a physical and visual layer of protection that makes "automated reading" and "visual analysis" by hackers nearly impossible. Custom fonts are designed for the following reasons:

- Bypassing OCR Tools: The font is designed so that optical character recognition software cannot recognize it. If the hacker tries to convert the image of the message into text, random symbols or spaces will appear because the program's database does not contain this design (Martin, 2016).
- Glyph Mapping: The designer can technically swap the positions of the letters; for example, the user sees the letter "A" while the software reads it as the letter "S". This makes copy-pasting produce encrypted text that is incomprehensible to the hacker (Scheichelbauer et al., 2013).
- Steganographic Fonts: The font can be designed so that secret data is stored in very slight changes in the shape of the letters (such as a slight tilt or specific thickness at certain angles) that are not perceived by the naked eye, but are decoded by special software (Shirali-Shahreza ., 2008).
- Breaking the pattern of frequency analysis: Hackers rely on "frequency analysis" to crack codes. Using a custom font with irregular geometric properties prevents algorithms from identifying the repetitive patterns of the language (Shirali-Shahreza ., 2008).

- The psychological and temporal factor: Unfamiliar fonts increase the "cognitive load" on the attacker, causing them to spend a long time trying to understand the nature of the font instead of focusing on breaking the core encryption, which is known as the "slowing tactic." (Diemand-Yauman, et al, 2011).

3.4 The Font Lab Application

This section elucidates the procedures for utilizing the Font Lab application to alter fonts, duplicate characters, adjust encoding tables, choose characters for modification, and revise font information fields. The subsequent sub-sections will detail the techniques employed by the Font Lab application for font creation (Jońca,2014).

3.4.1 Font Window

The illustration presented in Figure 3.9 depicts a font window characterized by an extensive table of cells that correspond to various characters and glyphs. This window also includes a caption, as well as several buttons and options. Each cell contains glyph identification details, which may include the name, Unicode index, or additional relevant information.

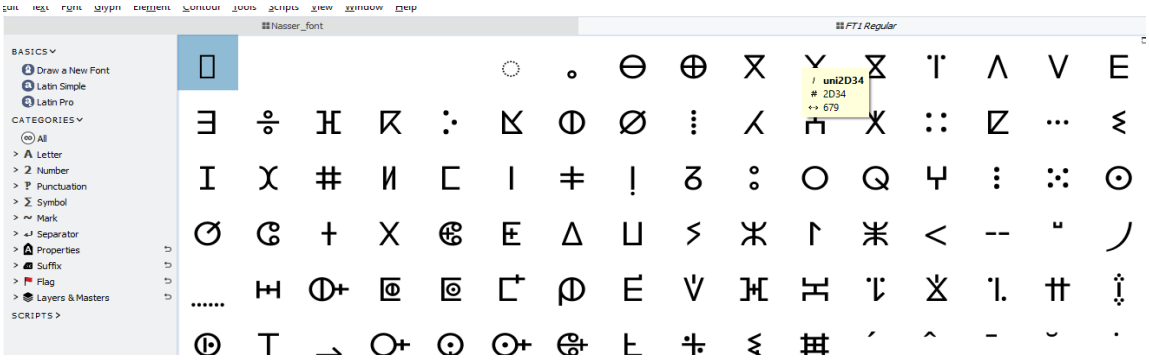


Figure 3.9: The font window.

3.4.2 Glyph Window

To modify the specific graphic characters required, access the Glyph window. Once the graphic cell has been established, we can proceed to open the avatar window. Figure 3.10 illustrates the process of adding a glyph window from the catalog (Jońca,2014).

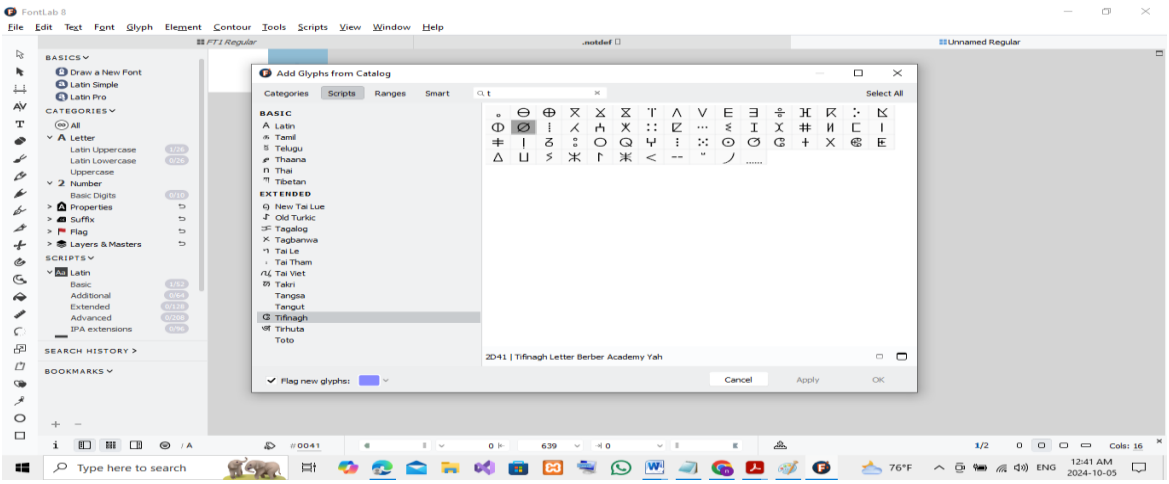


Figure 3.10: The glyph Window from the catalog.

3.4.3 Editing Fonts and their Information

In this subsection, the modifications related to font settings are discussed. The information encompasses character encoding details, font designations, copyright data, and additional specifications necessary for font utilization. This facilitates alterations in formatting attributes to align with search algorithms. Figure 3.11 illustrates the process of editing fonts along with their associated information (Jońca,2014).

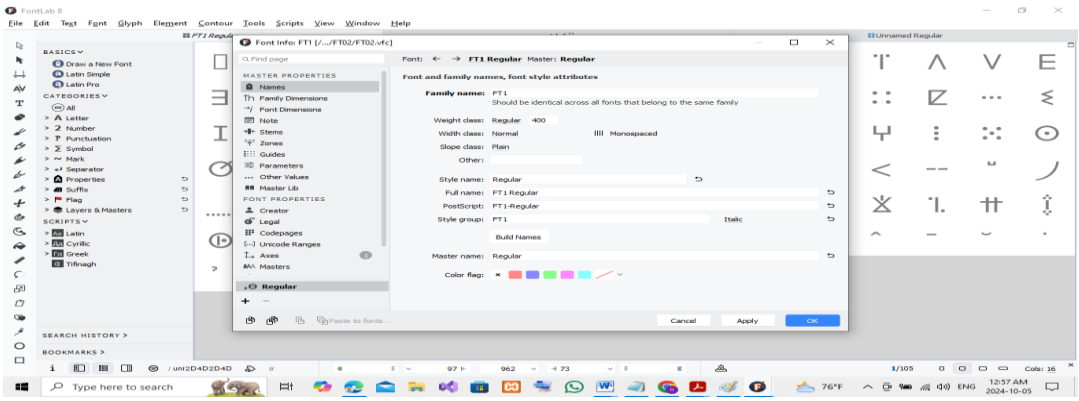


Figure 3.11: The editing fonts and their information.

3.4.4 The Export Profile

The profile constitutes a collection of preferences designated for the exportation of fonts in a particular format. Font Lab provides a series of pre-configured profiles for all standard output formats. Additionally, users have the capability to swiftly create new profiles. These attributes can be adjusted to align with the requirements of the research. Figure 3.12 illustrates the Export Profile.

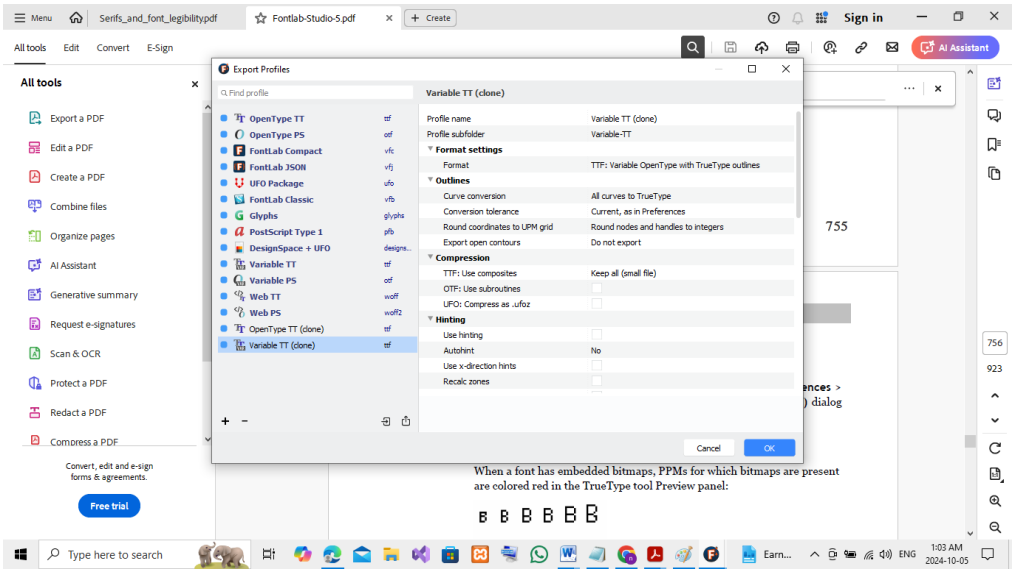


Figure 3.12: The export profile.

3.5 RSA Algorithm

Data can be secured through encryption and decryption utilizing the widely recognized and secure cryptographic method known as RSA. This technique provides a reliable means of transmitting confidential information across the Internet. Although it has several vulnerabilities, RSA continues to be employed for various applications, including digital signatures that authenticate the source of a message. The RSA algorithm is a fundamental component of public-key cryptosystems (Asjad,2018). Figure 3.13 illustrates the RSA process.



Figure 3.13: A diagram of the RSA process.

3.5.1 Justification for Adopting the RSA Algorithm

In this section, we highlight the primary reasons for choosing the RSA algorithm to be incorporated into our proposed method for steganography. These reasons are as follows:

- Although it requires a key length of at least 2048 bits for absolute security (Barker, 2020), the use of the RSA algorithm in our steganography method is not limited to encryption alone; it also serves as a transformation function that makes the data appear as random noise, which is a fundamental requirement in data masking to prevent statistical detection.

The integration of the RSA algorithm with Tifinagh language and FontLab creates a system that relies on data masking through obscurity. Even if the RSA encryption were theoretically compromised due to the small N (Barker, 2020), the biggest obstacle remains the "discovery" of the data within the visually oblique and programmatically fixed font properties. This system is very suitable for applications that require high confidentiality and denial of the existence of encryption in the first place.

3.5.2 Creation of an RSA Public/Private Key Pair

The initial phase of RSA encryption involves generating a pair of keys: one public and one private. The public key is made available to individuals who intend to send encrypted communications to the holder of the private key, whereas the private key is confidential and retained solely by its owner. As illustrated in Figure 3.14, adapted from Thomas H. C and colleagues (Jay Luo et al ,2023) , these are the steps involved in key creation.

1. Choose two large random prime numbers p and q privately.
2. Obtain n and $\phi(n)$ through $n = p \cdot q$ and $\phi(n) = (p-1)(q-1)$.
3. Choose a large number d that is relatively prime to $\phi(n)$.
4. Compute e such that $ed \equiv 1 \pmod{\phi(n)}$.
5. Release the key (e, n) to the public. [20].

Figure 3.14: The creation of public/private keys using RSA Algorithm.

Figure 3.15 shows an implementation of RSA algorithm using VB.Net Programming language.

```
Dim n As Integer

n = p * q
'   calculating totient, t
Dim t As Integer
t = (p - 1) * (q - 1)
'   selecting public key, e
Dim e As Integer
For i = 2 To t - 1
    If GCD(i, t) = 1 Then
        e = i
        Exit For
    End If
Next
'   selecting private key, d
Dim j As Object
j = 0
Dim mes As Integer

Dim ct As Integer
Dim dl As Integer
While True
    If (j * e) Mod t = 1 Then
        dl = j
        Exit While
    End If
    j += 1
End While
```

Figure 3.15: An Implementation of RSA algorithm using VB.Net.

3.5.3 The Encryption Equation

To secure a message through the RSA algorithm, the sender employs the public key of the recipient for the encryption process. The encryption of a message (M) is executed using the public key (e, n) according to the formula: (C = M^e mod n), which generates the cipher text (C) (Jay Luo and et al,2023). An illustration of the RSA message encryption process is presented in Figure 3.16.

```
'   performing encryption

ct = (message ^ e) Mod n
```

Figure 3.16: The encryption equation for the RSA algorithm.

3.5.4 The Decryption Equation.

To retrieve the original plaintext message from the encrypted text, recipients employ their private key. Subsequently, by utilizing the private key (d, n) along with the formula M = Cd mod n, the recipient is able to decrypt the encrypted message. Figure 3.17 illustrates the process of message decryption utilizing the RSA algorithm.

```
'  performing decryption
Dim buf As System.Numerics.BigInteger = BigInteger.Pow(message, d1)

' mes = (ct ^ d1) Mod n
mes = buf Mod n

System.Numerics.BigInteger.TryParse(mes, (ct ^ d1) Mod n)
Return mes
```

Figure 3.17: The decryption equation for RSA algorithm.

3.6 Summary

This chapter has presented a comprehensive examination of fonts, including their fundamental categories and the principles of their design. Additionally, it has offered an introduction to the RSA algorithm. These elements are deemed crucial for the development and execution of the proposed information-hiding algorithm. We elaborated on the creation of specialized fonts, which allow for the manipulation of various attributes, such as type and degree of slant, to facilitate the concealment of information. Furthermore, the RSA encryption algorithm can be effectively adapted to align with the requirements of the concealment method.

Chapter 4

Design of the Proposed Methods and Discussion of Illustrative Examples

Chapter 4: Design of the Proposed Methods and Discussion of Illustrative Examples

4.1- Introduction

This chapter encompasses a comprehensive review of various sections that elucidate the research methodology employed, which has evolved considerably since the initial conception of the idea. It also details the tools that have played a crucial role in elucidating the algorithms. The implementation steps for the proposed methods are examined, including the suggested hiding algorithm, accompanied by a detailed explanation of the corresponding diagrams. Furthermore, the proposed extraction (un hiding) algorithm is analyzed alongside its relevant diagrams. Conclusively, the chapter revisits earlier algorithms and presents the results that were achieved.

4.2- Research Methodology

Research methodology is a set of organized research methods, serving as a guide on how to conduct the research. The methodologies used in this context vary. In this study, the experimental method will be adopted. Experimental and creative research is considered a fundamental pillar in the sciences(Goundar,2012).

The experimental study primarily focuses on the cause-and-effect relationship. In this research, the idea begins with the use of certain tools in its development, among which are typography design applications that played an important role in the proposed idea, as they allowed us to modify the properties of the fonts to align with the concept. Additionally, the latest version of the Visual Basic programming language was used, which also contributed to achieving the results. An encryption algorithm (RSA) was also applied, which increased the reliability of the idea. Finally, the use of the common serial

standard, which supports all languages and scripts, cannot be overlooked. All these elements contributed in one way or another to shaping the idea and transforming it into an applicable and evaluable algorithms according to standards such as capacity, security, and reliability, in addition to presenting the expected results and comparing them with those of an existing algorithms similar to the proposed algorithms.

4.3- The Proposed Hiding Method

In this section, we will present the proposed method for concealing a secret message (written in any natural language) within a cover text (written in Tifinagh) file. Initially, the characters of the secret message will be encoded into binary form. Subsequently, the binary code will be converted into decimal numbers. Following this, the renowned RSA algorithm will be utilized to encrypt the resulting decimal numbers of the secret message. The encrypted secret message will then be transformed back into binary code. The final step involves correlating these binary codes with the properties of Tifinagh characters based on Tables 4.1 and 4.2, each containing 256 possibilities (16x16). Figure 4.1 illustrates each element in the primary table (Table 4.1) and its corresponding mapping as delineated in Table 4.2.

Table 4.1:Sixteen (16) main possibilities for each Tifinagh character.

Font Type	Binary Code			
	B1	B2	B3	B4
Ft002	0	0	0	0
Ft003	0	0	0	1
Ft004	0	0	1	0
Ft005	0	0	1	1
Ft006	0	1	0	0
Ft007	0	1	0	1
Ft008	0	1	1	0
Ft009	0	1	1	1
Ft010	1	0	0	0
Ft011	1	0	0	1
Ft012	1	0	1	0
Ft013	1	0	1	1
Ft014	1	1	0	0
Ft015	1	1	0	1
Ft016	1	1	1	0
Ft017	1	1	1	1

Table 4.2: Sixteen (16) sub-possibilities for each Tifinagh character.

Font Style	Binary Code			
	B1	B2	B3	B4
R+FS1	0	0	0	0
I+FS1	0	0	0	1
B+FS1	0	0	1	0
R+FS2	0	0	1	1
I+FS2	0	1	0	0
B+FS2	0	1	0	1
R+FS3	0	1	1	0
I+FS3	0	1	1	1
B+FS3	1	0	0	0
R+FS4	1	0	0	1
I+FS4	1	0	1	0
B+FS4	1	0	1	1
R+FS5	1	1	0	0
I+FS5	1	1	0	1
B+FS5	1	1	1	0
R+FS6	1	1	1	1

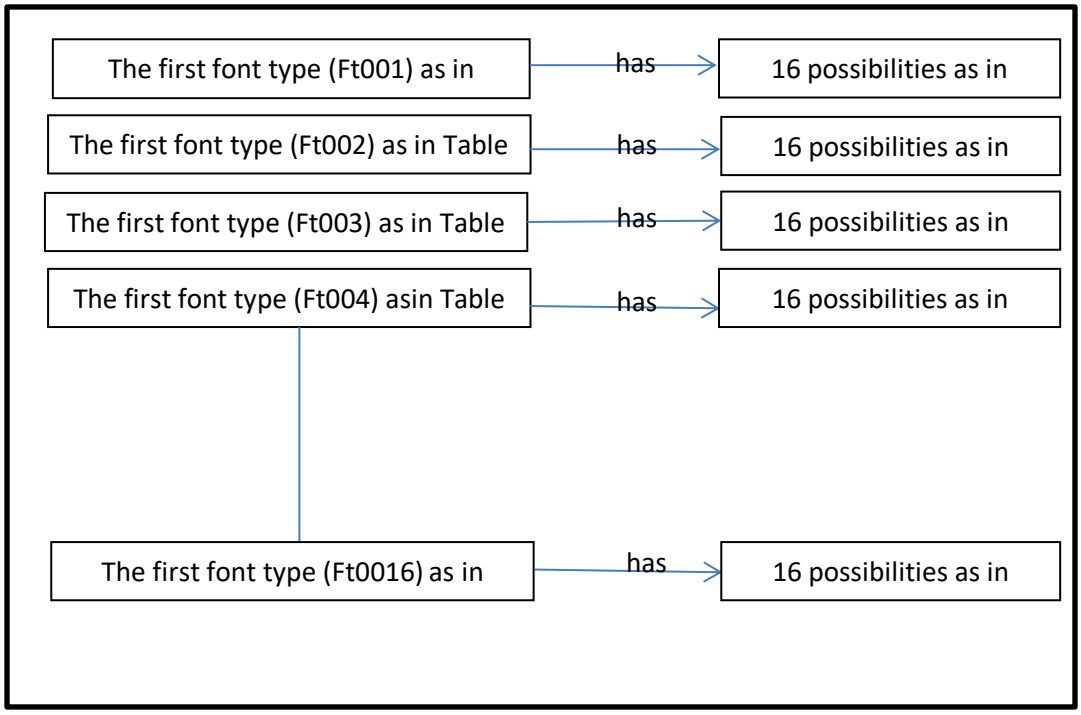


Figure (4.1): An illustration of 8-bit embedding.

Both tables 4.1 and 4.2 are combined to produce a new table which is referred to as Table 4.3. This new table contains 6 columns. The first column describes all possible binary codes which are 256 possibilities. The second column is referred to as x-bin which contains the first 4 bits of the binary codes of column 1. The third column describes the font type that can be used. The fourth column is referred to as y-bin which contains the second 4 bits of the binary codes of column 1. The fifth column describes the font style that can be used. The sixth column describes the font size that can be used. The value of the font size "OS+0.1" means that the original size of the carrier message (CM) plus a little value (0.1) that cannot influence the appearance of the CM.

Table 4.3: The binary codes and their associated character properties.

Possible Binary Codes	X-bin	Font Type	Y-bin	Font Style	Font Size
00000000	0000	FT002	0000	Regular	OS+0.1
00000001	0001	FT003	0001	Italic	OS+0.1
00000010	0010	FT004	0010	Bold	OS+0.1
00000011	0011	FT005	0011	Regular	OS+0.15
00000100	0100	FT006	0100	Italic	OS+0.15
00000101	0101	FT007	0101	Bold	OS+0.15
00000110	0110	FT008	0110	Regular	OS+0.2
00000111	1000	FT009	0111	Italic	OS+0.2
00001000	1001	FT010	1000	Bold	OS+0.2
00001001	1010	FT011	1001	Regular	OS+0.25
00001010	1000	FT012	1010	Italic	OS+0.25
-	-	-	-	-	-
-	-	-	-	-	-
11111111	1111	FT017	1111	Regular	OS+0.35

Most applications utilize points as the standard measurement for fonts, with 72 points in each inch. Despite being displayed at the same point size, certain fonts may appear larger or smaller. Free and shared software often exhibit these size discrepancies, especially in typefaces created by independent artists lacking the expertise of professionals working for major printing companies. Helvetica and Times fonts are generally considered reliable choices. However, variations in font sizes can occur due to different versions from various printing firms. In this proposed method, the font type and size will be leveraged to conceal a secret message. Figure 4.2 depicts a message written in Tifinagh, while a corresponding English translation is provided in Figure 4.3. The block diagram outlining the proposed method is illustrated as shown in Figure 4.4.

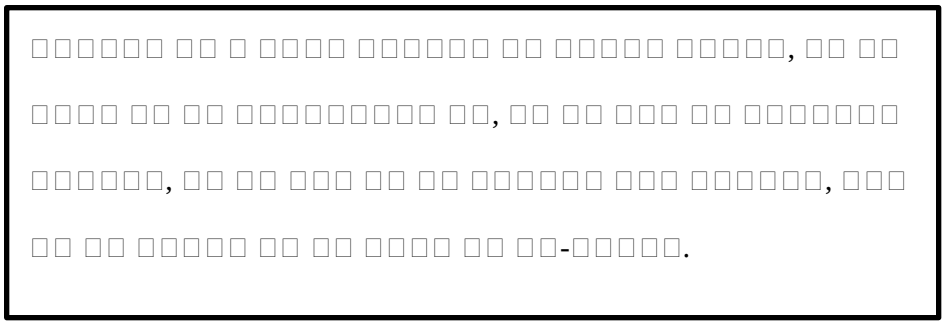


Figure 4.2 : message in Tifinagh.

Zuwara is a city located in western Libya, to the north of the Mediterranean Sea, to the east of the Malitha factory, to the west the village of Abi Kamash, and to the south the city of Al-Jamil.

Figure 4.3: An English translation.

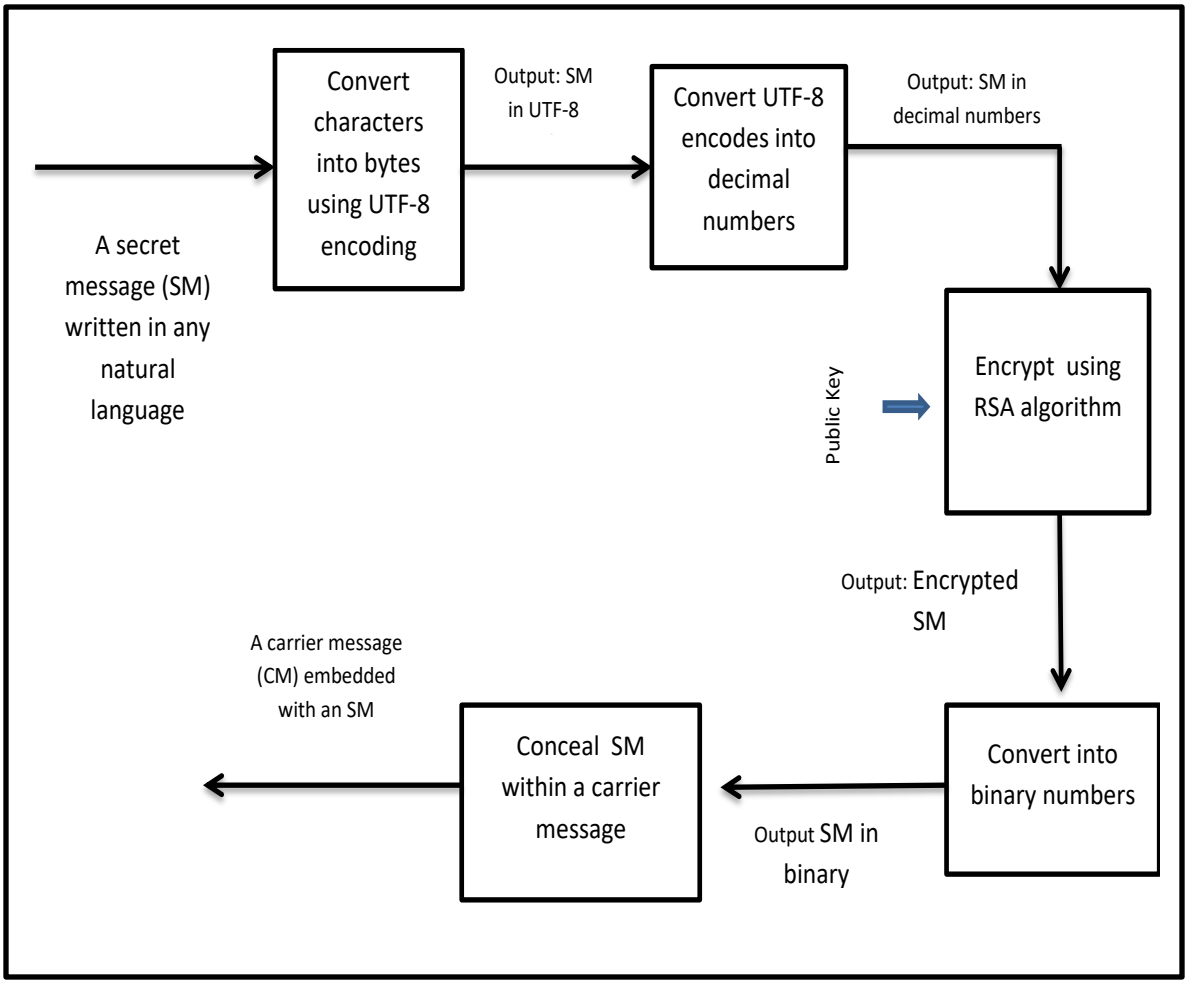


Figure (4.4): The block diagram of the proposed hiding (concealing) method.

Every logical computer program is built upon an algorithm, which should tackle a significant and clearly defined problem to be compelling. Algorithms can be expressed in plain text, formal flowcharts, or pseudocode when intended for computer implementation. Pseudocode is a structured form of English used to delineate algorithms, easily translatable into any programming language due to its general, code-like nature. Therefore, an algorithm is introduced for hiding (concealing) a plaintext message which is written in Tifinagh. This hiding algorithm is crafted based on the method of embedding text in the Tifinagh script. The key steps of the proposed hiding algorithm are outlined in the generic algorithm presented in Figure 4.5. The pseudocode

of the proposed hiding algorithm is listed as given in Appendix A. A generic flowchart of the concealing algorithm is illustrated in Figure 4.6. The detailed flowchart of the concealing algorithm is shown as given in Appendix A.[22]

The proposed method can be illustrated according to the following Generic Algorithm:

Algorithm: Steganography (Hiding) using Tifinagh Language

Input : Carrier message (CM) which is a text written in Tifinagh and a Secret Message (SM) which is a text written in any natural language.

Output : The carrier message (CM) embedded with a secret message (SM).

Start

1) Convert the secret message (SM) into UTF-8 encoding and store it in a one-dimensional array

2)If (The length of the CM is less than the length of the SM.) Then

Error Message:

The length of the CM must be at least twice or more the length of the SM.

Exit the Algorithm

End if

.

3) Convert the UTF-8 encoding into decimal and store it in a one-dimensional array.

4) Encode the generated decimal numbers using the well-known RSA algorithm and store the result in a one-dimensional array.

5) Convert the encoded decimal numbers into binary codes and store the result in a one-dimensional array.

6) Embed the SM with the CM: change the font properties of CM characters according to the generated binary codes.

Figure 4.5: The key steps of the proposed hiding algorithm.

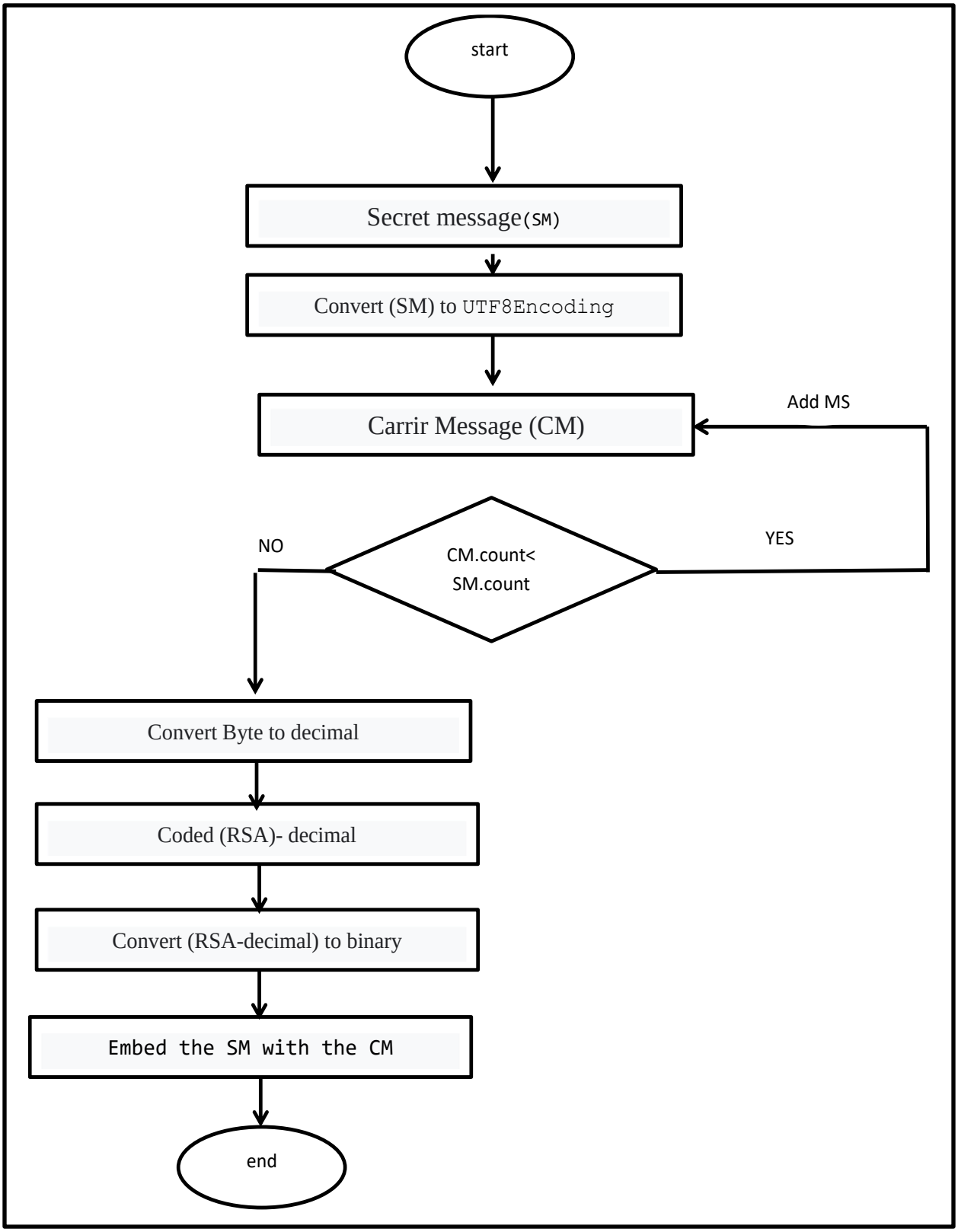


Figure 4.6 : A flowchart of the hiding Algorithm

4.4- Discussion of the Proposed Hiding Algorithm

The process of the proposed hiding algorithm can be effectively demonstrated through an example illustrating how the concealing algorithm functions. Let us consider the English message "welcome" which will be hidden. Furthermore, we will utilize the carrier message written in Tifinagh: "□□□□□□□ □□ □ □□□□ □□□□□□ □□ □□□□□ □□□□□,". It is chosen to be concise to facilitate a clear explanation and comprehension of the hiding algorithm steps.

As a matter of fact, the hiding algorithm operates based on the following general steps:

Step 1: involves retaining a one-dimensional array and populating it with the secret message converted to a UTF-8 encoding, as depicted in Figure 4.7. In this array, the first four locations are reserved for UTF-8 encoding which are fixed codes. Starting from the 5^{th} location, UTF-8 of the SM will be stored.

239	187	191	119	101	108	99	111	109	101	32

Figure 4.7: UTF-8 encoding of the secret message.

Step 2: involves generating an encrypted secret message utilizing the RSA algorithm, as demonstrated in Figure 4.8.

239	238	251	119	101	12	24	66	139	101	128

Figure 4.8: The encrypted secret message.

Step 3: concerns converting the encrypted secret message into an 8-bit binary codes, as outlined in column 1 of Table 4.4 which shows the generated binary codes and their associated character properties.

Step 4: the 8-bit binary codes as shown in column 1 of Table 4.4 will be divided into two separate columns (x-bin and y-bin), with each column containing 4 bits, as illustrated in column 2 and 4 of Table 4.4.

Step 5: presents the corresponding font types associated with x-bin as shown in column 3 of Table 4.4.

Step 6: presents the corresponding font style and font size associated with y-bin as shown in columns 5 and 6 of Table 4.4.

Step 7: assigns font properties, including: font type, font style, and font size, to variables involving searching for the first character of the carrier message and adjusting its properties. This process is then repeated for each character of the secret message until all characters properties are adjusted.

Table 4.4: The generated binary codes and their associated character properties.

Binary codes	X-bin	Font type	y-bin	Style Font	Font size
00001100	0000	FT007	1100	Bold	OS+0.15
00011000	0001	FT008	1000	Regular	OS+0.2
01000010	0100	FT009	0010	Italic	OS+0.2
10001011	1000	FT010	1011	Bold	OS+0.2
01100101	0110	FT006	0101	Regular	OS+0.25
10000000	1000	FT010	0000	Italic	OS+0.25

Once the secret message has been embedded, the carrier message will be as follows:

"□□□□□□□ □□ □ □□□□ □□□□□□ □□ □□□□□ □□□□□,"

Figure 4.9 displays the carrier message before concealing the secret message. The font type of the carrier message is “FT001”, its font style is “Italic” and its size is 12.

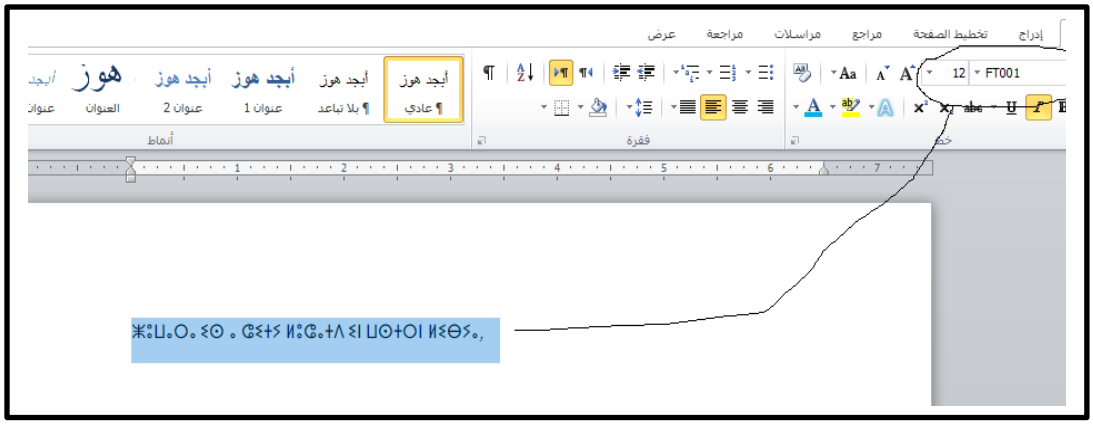


Figure 4.9: The carrier message before concealing the secret message.

Figure 4.10 displays the carrier message before and after the embedding process. The second message which is displayed in Figure 4.10 shows the font type of the carrier message is “FT004”, its style is “Bold”, and its size is 12.1. However, the MS-word application does not display this size, instead it displays the size 12. The font type, style, and size of the CM after the embedding process are clearly changed compared with the font type, style, and size of the CM before the embedding process. The appearance of the CM after the embedding process is almost exactly as the appearance of the CM before the embedding process. This appearance makes it difficult to notice in other applications (Baker,2014).

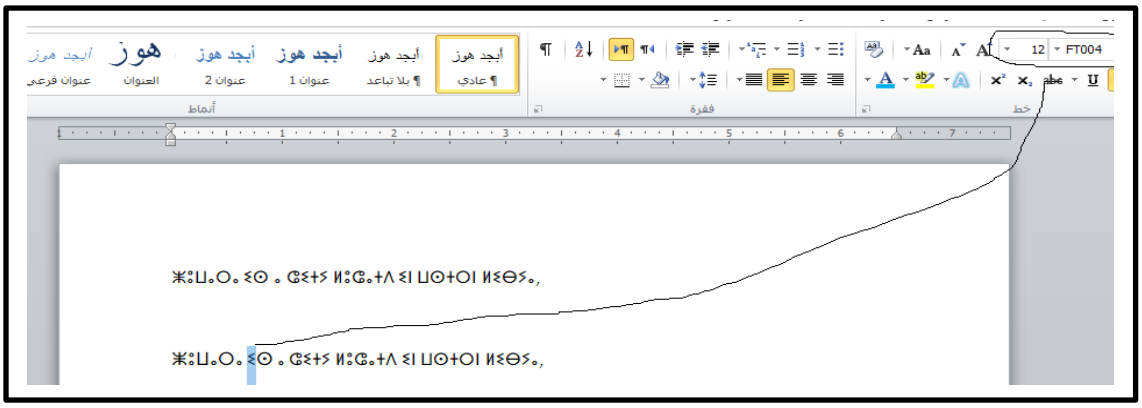


Figure 4.10 The CM after concealing the SM.

4.5- The Proposed Uncovering (Extraction) Method

In this section, the methodology used to extract a secret message embedded within a carrier message will be reviewed. Initially, the binary codes representing the characters of the carrier message will be obtained based on the characteristics of Tifinagh language. Subsequently, these binary codes will be converted into decimal numbers. Then, the well-known RSA algorithm will be applied to decrypt the extracted decimal numbers related to the secret message. After the decryption process, the secret message will be converted back into binary codes. The final step involves converting these binary codes back into characters according to the extraction algorithm. Figure 4.11 illustrates a block diagram of the proposed extraction method.

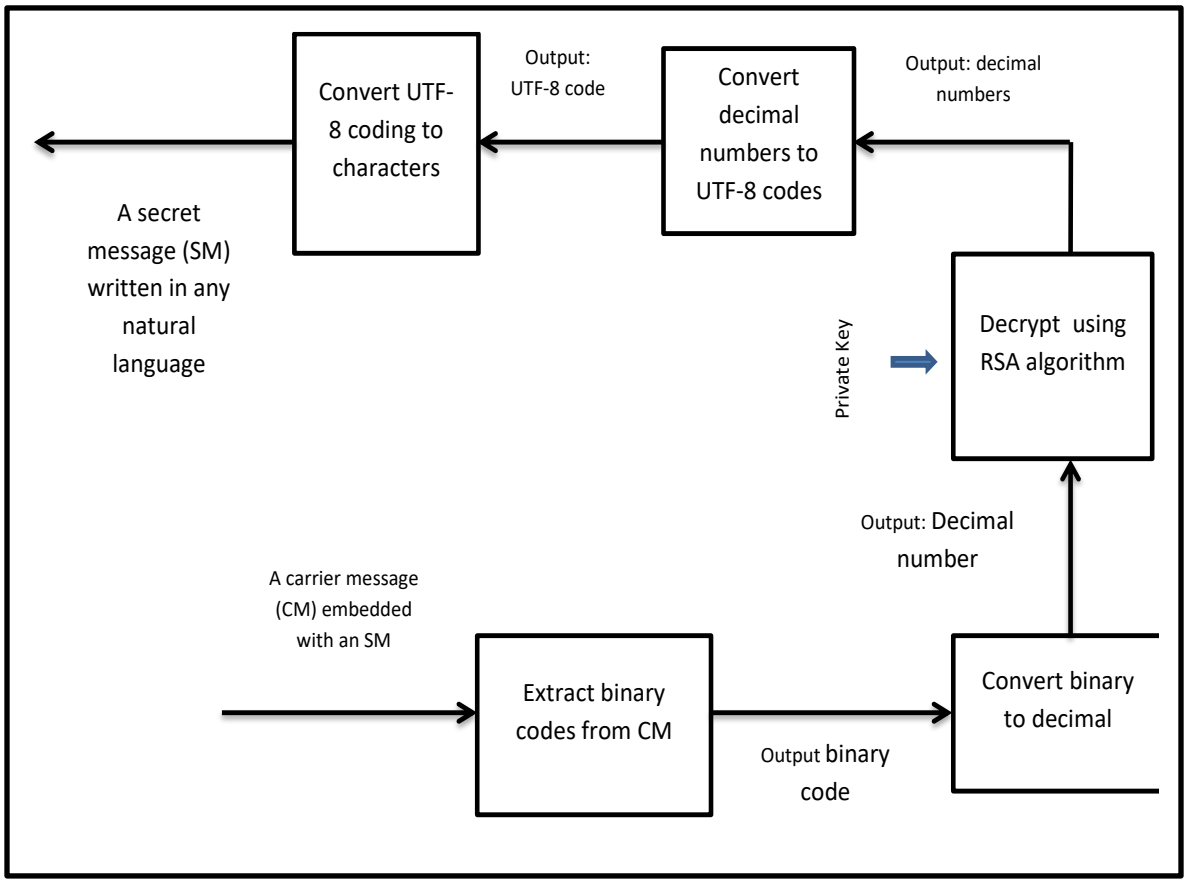


Figure 4.11: A block diagram of the proposed extraction method.

An algorithm, based on the extraction method, for extracting a hidden text message written in any natural language has been proposed. Figure 4.12 illustrates the generic algorithm outlining the key stages of the proposed extraction method. Figure 4.13 presents the generic flowchart of the extraction method. The pseudocode of the proposed extraction algorithm is listed as given in Appendix B. Also, the detailed flowchart of the extraction algorithm is shown as given in Appendix B.

The proposed extraction method can be illustrated according to the following Algorithm:

Algorithm: Extraction of Secret Message

Input: The carrier message (CM) embedded with a secret message (SM).

Output: The Secret Message (SM) written in any human language.

Start

1) Highlighting the features of the fonts used in the letters of the carrier message and replacing them with the binary numbers associated with those features.
2) The Jth (8 bits in length) of the binary code is converted into decimal number.
3) The decimal number is decoded with the well-known RSA algorithm.
4) Converting the decoded numbers UTF-8 encoding.
5) Converting the UTF-8 encoding to characters .
6) End. {End of the Algorithm}

Figure 4.12: The proposed algorithm outlined in pseudocode.

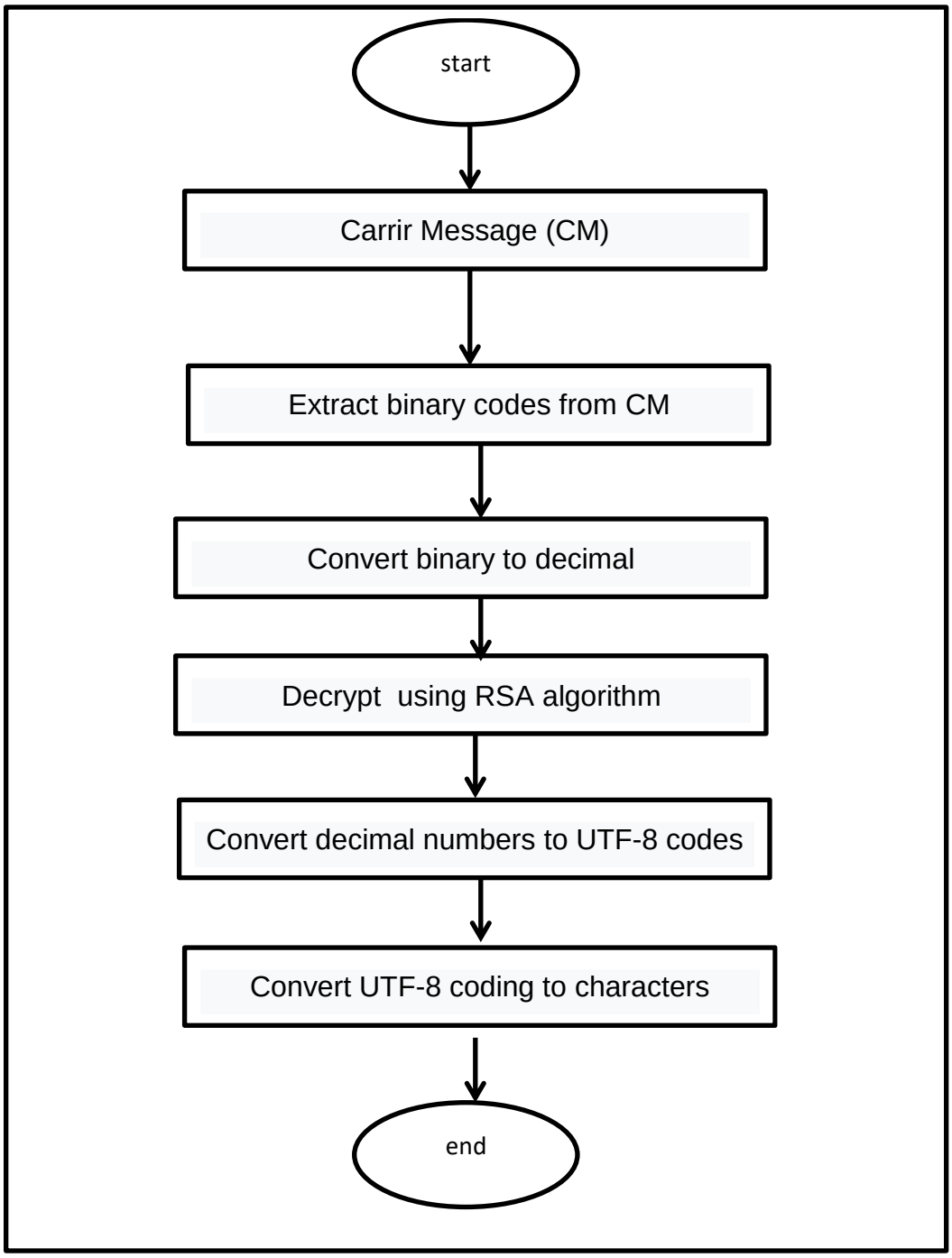


Figure 4.13: Flowchart of the Decoding Algorithm.

4.6- Discussion of the Extraction Algorithm

The CM embedded with an SM reads as follows: "□□□□□□□ □□ □ □□□□ □□□□□□ □□ □□□□□ □□□□□,". As a matter of fact, the extraction algorithm works based on the following general steps:

Step 1: Search characters of the CM until reaching the character that matches the font type "original font type" OR reaching end of the CM.

Step 2: Based on character properties of the CM, extract binary codes according to Table 4.3 (as shown above). For example, the properties of the CM's first character are font type "FT016", font style "Regular", and font size "12.35". Corresponding to Table 4.3 (above), the X-bin of the font type "FT016" is "1110". The Y-bin of the font style "Regular", and font size "12.35" is "1111".

Step 3: Combine both X-bin and Y-bin to obtain an 8-bit binary code. The generated binary codes of the SM associated with their properties are listed as given in Table 4.5.

Table 4.5: The generated binary codes of the SM associated with their properties.

11101111
11101110
11111011
01110111
01100101
00001100
00011000
01000010
10001011
01100101

10000000

Step 4: Convert 8-bit binary codes into decimal numbers, as illustrated in Figure 4.14

239	238	251	119	101	12	24	66	139	101	128

Figure 4.14: The encrypted SM in decimal numbers.

Step 5: Using the RSA algorithm, decrypt the encrypted SM in decimal numbers. The result is stored in a one-dimensional array as depicted in Figure 4.15.

239	187	191	119	101	108	99	111	109	101	32

Figure 4.15: The decrypted SM.

Step 6: Convert a one-dimensional array of the decrypted SM into UTF-8 encoding and store them in a one-dimensional array as illustrated in Figure 4.16.

239	187	191	119	101	108	99	111	109	101	32

Figure 4.16: The UTF-8 encoding of the SM.

Step 7: Convert the UTF-8 Encoding of SM into characters of the SM. The final output of extraction algorithm will be the SM "**welcome**".

4.7- Summary

In this chapter, the research methodology has been discussed. Also, the proposed stenography methods are explained accompanied with their algorithms. Eventually, the hiding and un-hiding algorithms are discussed provided by examples illustrating the embedding and extraction processes.

Chapter 5

An Implementation and Evaluation of the Proposed Methods

Chapter 5: An Implementation and Evaluation of the Proposed Methods

5.1- Introduction

In the previous chapters, we discussed the proposed methods, designed the associated algorithms, and provided a comprehensive explanation of their details, including flowcharts. Illustrative examples are provided to highlight how the designed algorithms work. In this chapter, the designed algorithms are implemented using the latest version of VB.Net programming language (Shapiro, 2017). An evaluation of the implemented algorithms is carried out in terms of security, capacity, and reliability. A comparison has been conducted between the implemented algorithms and similar existing algorithms operating in a similar environment.

5.2- Implementations of the Hiding and Uncovering Algorithms

This section presents an explanation of the implementations for the hiding and uncovering algorithms.

Figure 5.1 shows a screen of the implemented hiding algorithm which is at the sender side. This screen is a dialog window which includes a dedicated field for entering the secret message (SM), written in any natural language. Also, a button for uploading the carrier message (CM) in RTF file format. In addition, a “hiding” button which can be clicked to hide the SM.

Figure 5.2 illustrates the process of hiding the SM within the CM. As shown in Figure 5.2, the embedded SM with the CM will be displayed in its designated area. When the send button is clicked, the CM embedded with the SM will be sent to the receiver.

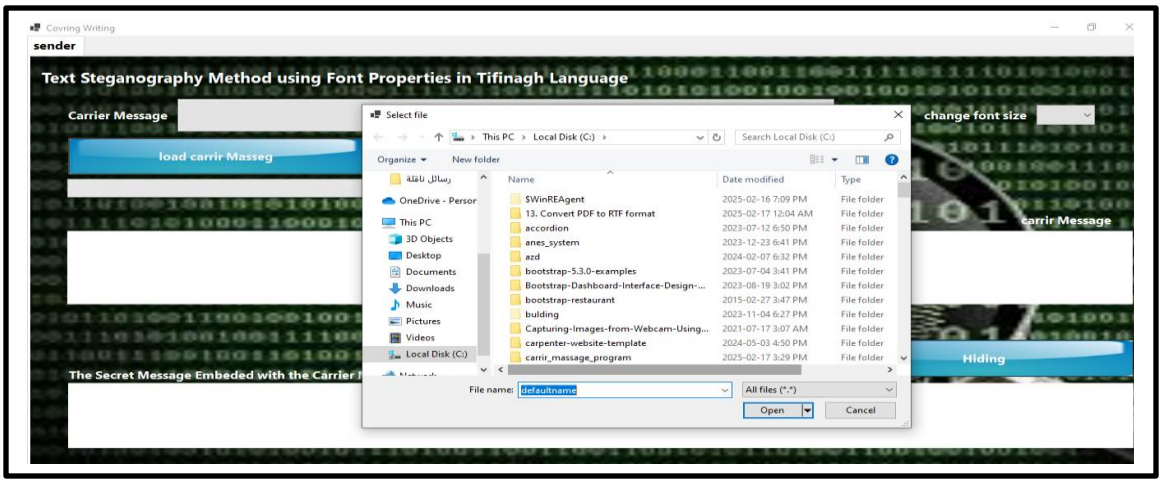


Figure 5.1: A dialog window for the implemented hiding algorithm.

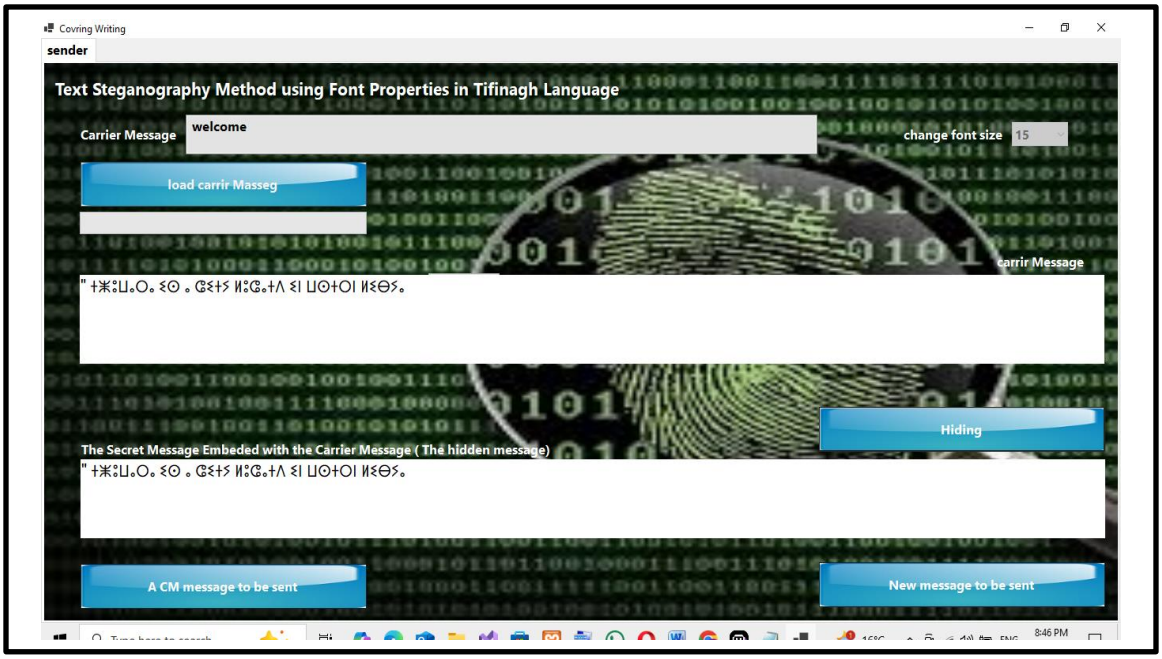


Figure 5.2: The process of hiding the SM within the CM.

Figure 5.3 shows a screen of the implemented uncovering algorithm which is at the receiver side. This screen is a dialog window which contains a button to download the CM, embedded with the SM, received from the sender side. Figure 5.3 illustrates the process of downloading the CM, embedded with the SM. As shown in Figure 5.4, the CM, embedded with the SM will appear in its designated area. When the “extraction”

button is clicked, the extraction process is executed automatically and the extracted SM will be displayed in its designated rectangle.

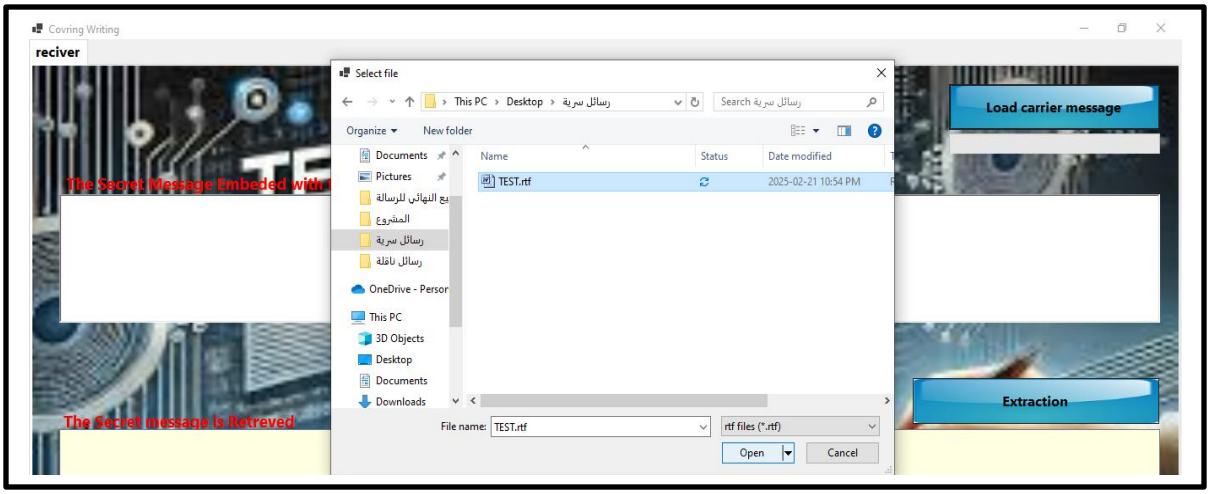


Figure 5.3: The process of loading the CM embedded with SM.

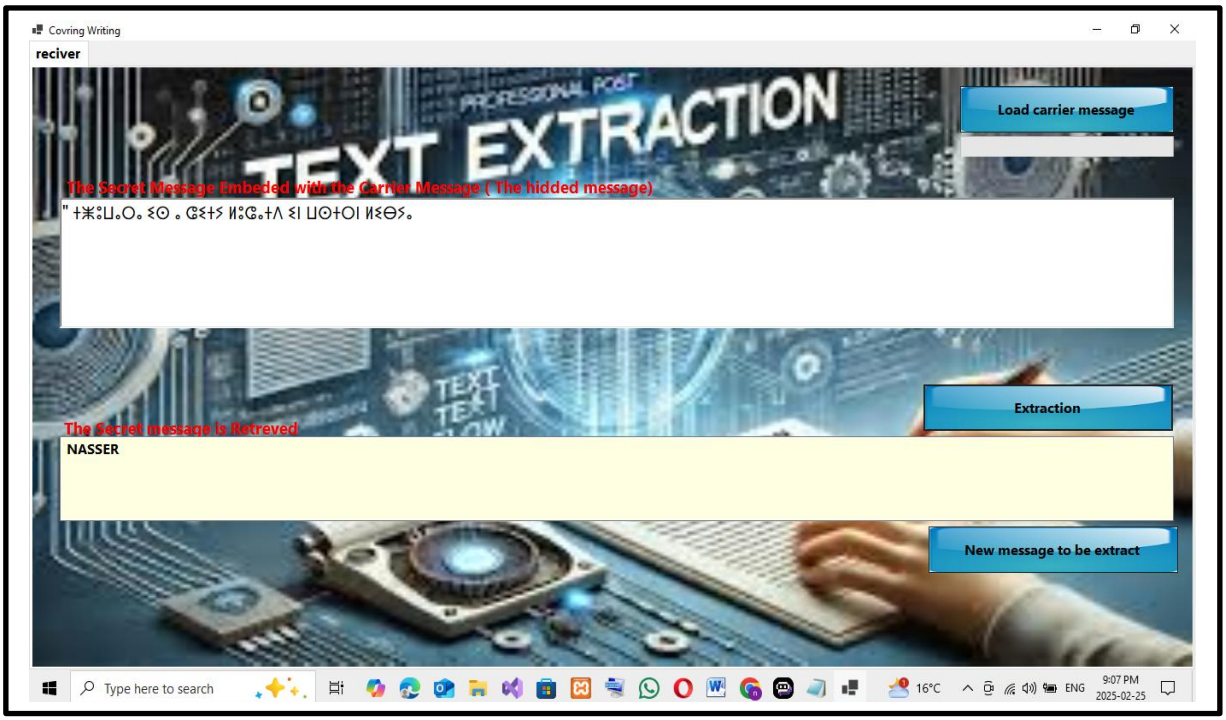


Figure 5.4: The process of extracting the SM from the CM.

5.3- An Evaluation of the Designed Algorithms

The designed algorithms can be assessed through capacity, security, and robustness metrics. Capacity denotes the amount of information bits that can be concealed within a cover medium, while security pertains to the difficulty of detecting the hidden data by unauthorized parties. Robustness reflects the algorithm's resilience against data corruption or tampering. In text steganography, achieving a high embedding capacity is challenging due to the limited redundancy in textual data compared to multimedia files. Maintaining text symmetry while embedding data is crucial to avoid detection by adversaries. The capacity and security evaluations in text steganography consider these factors and employ various measurement approaches. The capacity measurement equation, which is adopted from (Nofaie and et al,2021),. Provides a reliable estimation of the algorithm's capacity and enables comparison with similar algorithms. The capacity measurement is given as in equation 5.1.

$$\text{Capacity Ratio} = (\text{Number of hidden bits} \div \text{Size of carrier cover}) \times 100 \quad \text{Eq. (5.1)}$$

The security evaluation aims to ensure that the concealed data remains undetectable during transmission over open networks to prevent unauthorized access. If a third party detects or suspects the presence of hidden information within the cover media, the security of the system is compromised. The measurement of excess visible and invisible characters post-embedding is crucial in assessing the security of steganographic approaches. In text-based steganography, a lower security ratio indicates weaker security when the number of excess characters in the stego text surpasses those in the original cover text. The provided equations are well-suited for calculating the security evaluation

metric and enable comparisons with similar methodologies. These metrics quantify the increase in visible and invisible characters in the stego text compared to the original text. Security evaluations play a fundamental role in verifying the algorithm's success in ensuring data confidentiality, as demonstrated in equations 5.2, 5.3, and 5.4(Taha and etl,2020).

$$\text{Discount Value} = (\text{Excess characters} \div \text{Original Characters}) \times 100 \quad \text{Eq. (5.2)}$$

$$\text{Amount after Discount} = (\text{Original characters} - \text{Discount Value}) \quad \text{Eq. (5.3)}$$

$$\text{Security Ratio} = (\text{Amount after Discount} \div \text{Original Characters}) \times 100 \quad \text{Eq. (5.4)}$$

The designed hiding algorithm, as detailed in Chapter 2, Figure 2.8, exhibits linear characteristics with a complexity of O(n), where 'n' represents the window size utilized for hiding the secret message. Evaluating the algorithm involves comparing it with established algorithms like the "Similarity of English Font Types (SEFT)" and Hybrid Arabic Text Steganography. The SEFT algorithm leverages the similarity among English font types to conceal messages by transitioning between fonts. Each font type typically has numerous similar counterparts, forming the basis of the algorithm. By identifying the document font type and selecting similar alternatives, the SEFT algorithm encodes each symbol in the secret message using three different fonts, enabling the concealment of twenty seven characters (English alphabets with space) with

three fonts. This algorithm requires three characters to hide one character of the secret message, allowing for efficient data embedding. Conversely, the Arabic text concealment algorithm, based on Unicode encoding, conceals information within Arabic text by utilizing the contextual form of Arabic characters, hiding one bit within each character. This algorithm focuses on embedding crucial details within the structure of Arabic characters to ensure secure data concealment. Drawing upon the information presented, we will now conduct a comparative analysis from two distinct perspectives. First, from the robustness perspective, as presented in Table 5.1. Second, from the capacity and safety standpoints, as illustrated in Table 5.2.

First: from the robustness perspective, Table 5.1 depicts a comparative analysis of the proposed hiding algorithm and alternative algorithms. Table 5.1 is designed to be self-explanatory, conveying its message effectively (Michelucci, 1999).

Table 5.1: A comparison of the proposed hiding algorithm with alternative algorithms from the robustness perspective.

The point of comparison	The " SEFT" Algorithm	Arabic Text Concealment Algorithm	The Proposed Hiding Algorithm
Stealth technology	Replacing the font type or style.	Change in appearance (stretching, dots, wavy lines).	Manipulating metadata.
Robustness	Medium: Data is lost if the fonts are standardized in the document.	Very low: Data is lost when "copying text" or changing the format.	Very high: The data is hidden in the deep tables of the font file itself.
Imperceptibility	Moderate: A sudden change in the font style may raise	Weak: It is easy to notice the excessive kashida or strange dots.	Superior: Absolute visual stability; the eye and artificial

	suspicion.		intelligence see no change.
Efficiency	Good, but it is limited to the available font types.	It requires a very long text (high overhead).	Super: High storage capacity for each character (multi-features).

Second:

A demonstrative case highlighting the utilization of the capacity and safety equations is provided. An algorithm's capability is demonstrated for overall comparison based on fixed cover, and detailed comparisons based on random covers with the potential to embed a specified number of bits as documented in (Al-Anzi et al., 2020).

For Arabic text concealment algorithm as illustrated in (Al-Anzi et al,, 2020), the Capacity ratio, Discount value, Amount after discount, and Security ratio are computed as follows:

Capacity ratio = $(38 \div (43 \times 16)) \times 100 = 5.52$

Discount value= $((7) \div (43)) \times 100 = 16.27$

Amount after discount = $(43 - 16.27) = 26.73$

Security ratio = $(26.73 \div 43) \times 100 = 62$

1. Analysis of the Equations

i. Capacity Ratio

Equation:

Capacity ratio = (38 ÷ (43 × 16)) × 100 = 5.52

Analysis:

- The Numerator (38): Represents the size of the data actually hidden.

- The Denominator (43 × 16): Represents the Total Theoretical Capacity of the text. If we utilized all characters (43) and maximized the capacity of each character (16), we could theoretically hide this large number of units.

- The Result (5.52%): This indicates that only approximately 5.5% of the text's total available capacity was used. This ratio demonstrates the efficiency of capacity, suggesting that the text is capable of accommodating a significantly larger secret message.

ii.Discount Value

Equation:

Discount value= ((7) ÷ (43)) ×100 = 16.27

Analysis:

- This equation calculates the ratio of "loss" or inefficiency within the text. The percentage 16.27% represents the portion of the text (7 characters out of 43) that proved unusable for embedding. A lower value in this metric indicates a more efficient algorithm, as it maximizes the utility of the cover text's characters.

iii.Amount After Discount

Equation:

Amount after discount = (43 – 16.27) = 26.73

Analysis:

- This equation subtracts the Discount Value (as a numerical quantity) from the total character count.

- The result (26.73) serves as an Intermediate Value used by the authors to compute the security ratio. It effectively represents the "net effective value" of the text after accounting for the unused or insecure segments.

iv.Security Ratio

Equation:

Security ratio = (26.73 ÷ 43) ×100 = 62

Analysis:

- This equation measures the level of "security" or "robustness" based on the usable portions of the text.

- The result of approximately 62% suggests that the text retains a satisfactory balance between the characters used for embedding and the original characters. According to the security definition in this study, this ratio implies that the data-bearing text maintains a structure that is more difficult to detect, thereby preserving steganographic security.

For SEFT algorithm, the above evaluation measures are computed as follows:

Capacity ratio = (16 ÷ (43 × 8)) × 100 = 4.65

Discount value= ((6) ÷ (43)) × 100 = 13.95

Amount after discount = (43 – 13.95) = 29.05

Security ratio = (29.05÷ 43) × 100 = 67.55

For our proposed hiding algorithm, the above evaluation measures are computed as follows:

Capacity Ratio = (296÷ (43 × 24)) × 100 = 28.68

Discount value= ((0) ÷ (43)) × 100 =0

Amount after discount = (43 – 0) = 43

Security ratio = (43÷ 43) × 100 = 100%

An algorithm's capability is demonstrated for overall comparison based on fixed cover, and detailed comparisons based on random covers. In this section, we carry out an overall testing on three text steganography algorithms. Fixed cover text is used and secret bits applied similarly on our proposed hiding algorithm as well as the other two algorithms. The experiment results are listed in Table 5.2. It shows an overall comparison of three different algorithms. It can be noticed that the data-hiding capacity for our proposed algorithm is higher than all other algorithms. For the cover (**Carrier**) message (M1), the capacity ratio of our proposed algorithm 28.68%, which is better than the SEFT algorithm's capacity ratio (4.65%) for the cover (**Carrier**) message (M2) and also better than Arabic text concealment algorithm's capacity ratio (5.52%) for the cover (**Carrier**) message (M3). Furthermore, the SEFT algorithm's capacity ratio (4.65%) for the cover message (M2) is noticed very low that uses only one character. In fact, the algorithms which uses only one or two characters to hide secret bits influence the overall performance of such algorithms, as there are many overlooked characters to be explored. In terms of security and as shown in Table 5.2, our proposed hiding algorithm outperforms the other two algorithms. The study further investigated the possible reason of making our proposed hiding algorithm the best performing algorithm is that it changes Unicode of isolated characters to hide secret bits without extra characters. This can be further investigated in future as predictive to show some sort of negative effect on security. The study also investigated the performance of the three algorithms for many

random messages (covers) as shown in Table 5.3. The results show that the average capacity ratios of our proposed hiding algorithm are 30.7% and 30.3% for English and Arabic secret messages respectively. These values are higher than those obtained for SEFT algorithm (5.80%) and Arabic text concealment algorithm (5.07%). The capacity ratio of our proposed hiding algorithm is found to be very encouraging compared with those of the other two algorithms. For more convenient illustration, the obtained results are plotted as shown in Figures 5.5 and 5.6 respectively.

Table 5.2: A comparison of our algorithm and alternative algorithms in terms of capacity and safety ratios.

Input/output		No. of hidden bits	Increased characters	Capacity	Security	Method
Cover Message (M1)	□□□□□□□□ □□ □ □□□□ □□□□□□ □□ □□□□□ □□□□□,□□	296	0	28.68	100	The proposed algorithm
Cover Message (M2)	A Zawiya Graduate School Computer Science .	16	6	4.65	67.55	SEFT algorithm
Cover Message (M3)	المسلم من سلم المسلمون من لسانه ويده	38	7	5.52	62	Arabic text concealment algorithm

Table5.3: Detailed comparisons of different algorithms based on Capacity ratio.

No.	no. of bits of cover message	M1 (Our proposed Algorithm)				M2 (SEFT Algorithm)		M3 (Arabic text concealment Algorithm)	
		Bits (English)	Bits (Arabic)	Capacity ratio (English)%	Capacity ratio (Arabic)%	Bits	Capacity ratio (English)%	Bits	Capacity ratio (Arabic)%
1	688	208	208	30.23	30.23	32	4.65	40	5.81
2	336	96	96	28.57	28.57	16	4.76	20	5.95
3	816	248	240	30.39	29.41	48	5.88	46	5.64
4	1056	328	320	31.06	30.30	64	6.06	62	5.87
5	960	296	288	30.83	30	56	5.83	56	5.83
6	1728	552	544	31.94	31.48	104	6.01	98	5.67
7	992	304	304	30.64	30.64	64	6.45	52	5.24
8	1648	520	512	31.55	31.06	104	6.31	88	5.33
9	1152	360	352	31.25	30.55	72	6.25	66	5.72
10	832	256	256	30.76	30.76	48	5.76	46	5.40
AVERAGE				30.7	30.3		5.80		5.079

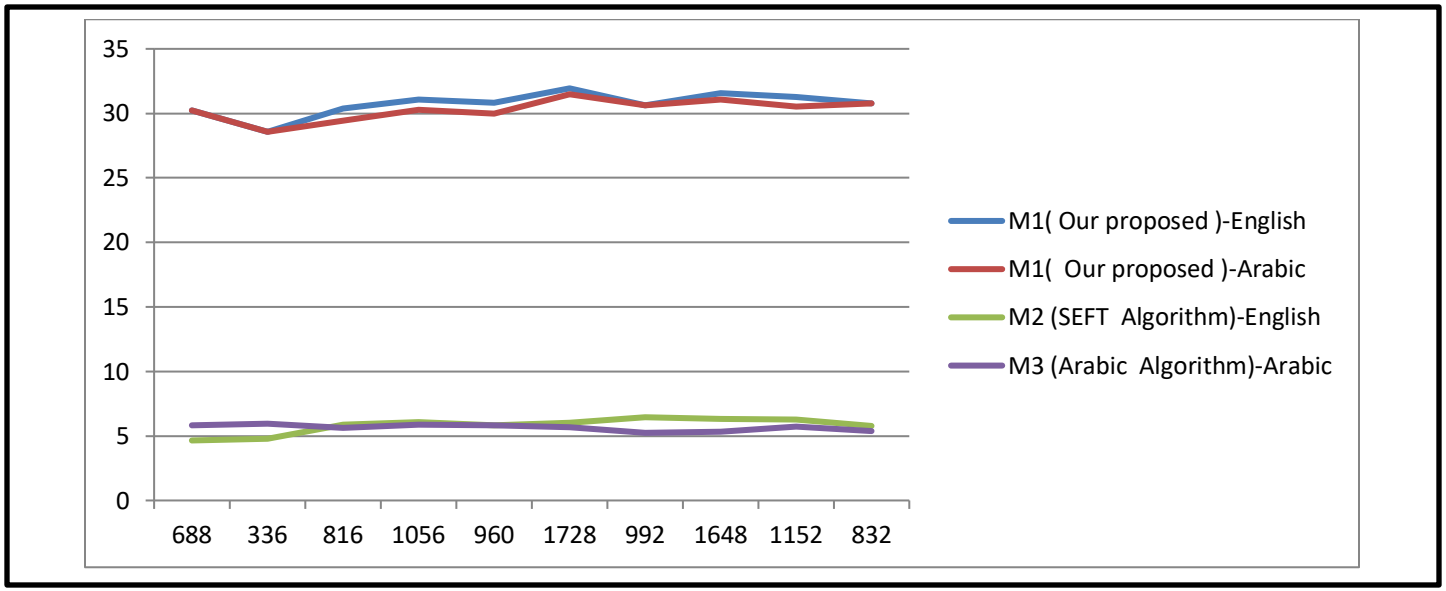


Figure 5.4: Capacity ratios of three different algorithms.

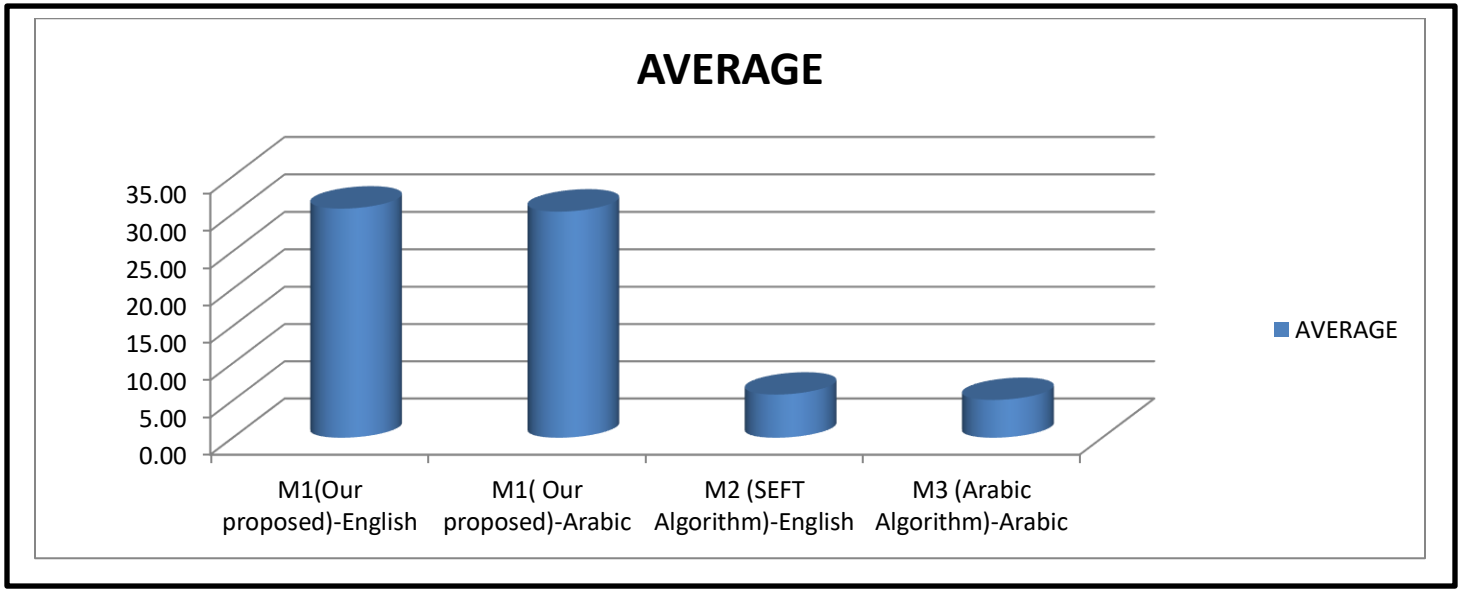


Figure 5.5: Average capacity ratio of three different algorithms.

5.4- Summary

The designed algorithms associated with the proposed hiding and un-hiding methods are implemented, presented, and discussed. The implemented algorithms work well, especially with short and multi-lingual messages. The proposed hiding algorithm has undergone evaluation in accordance with the predetermined criteria. A comparative evaluation of the proposed algorithm against existing and similar algorithms reveals that it excels in efficiency, security, and capacity when measured against the alternatives.

Chapter 6

Conclusions and Future Works

Chapter 6: Conclusions and Future Works.

6.1- Conclusions

Information security is considered one of the essential factors that contribute to the success of institutions and countries, as significant financial resources are allocated to ensure its protection. Documenting and transferring data securely requires taking measures to prevent any unauthorized entity from accessing or benefiting from it. The concepts of information security and its components were discussed, in addition to the threats that any information system may face and the protection mechanisms, as explained in Chapter 2. In Chapter 3, the fonts, font types, and design methods were presented, in addition to discussing encryption and its concepts. As presented in Chapter 4, new methods for hiding and un-hiding are proposed and their associated algorithms are designed. The new methods are also explained by providing some illustrative examples. Chapter 5 provides an implementation and evaluation of the associated algorithms. An evaluation of the implemented algorithms has been conducted through a series of experiments to test their accuracy and efficiency. The obtained results clearly demonstrate the correctness and completeness of the implemented algorithms. These results are encouraging and promising. However, the search for experts in hacking within Libya has proven unsuccessful, rendering us unable to procure a professional assessment of the implemented algorithms. We plan to disseminate it on hacking platforms as a challenge aimed at capturing their interest in resolving it.

6.2- The Main Contribution

The contribution lies in building a "hybrid system with asymmetric layers of protection" that integrates classical encryption (the RSA algorithm) and advanced information

hiding (Steganography) within the "invisible software properties" of fonts. In other words, the proposed method did not just encrypt the text; it also re-engineered the "Font File" to be the secret storage container itself, so that the presence of the data cannot be detected even if the text undergoes traditional visual or statistical analysis.

6.3- The Distinctiveness of this Work

The progress made in this research study consists of several technical points that have not been integrated together in this way before, these points are as follows:

- Exploiting the gap between "visual rendering" and "tabular values":

The new idea here is that the font is visually slanted in a fixed manner (Hardcoded), while its programming value (Italic Angle) changes in the Metadata tables to encode the data. Most previous research changed the tilt visually to hide the data, making it easy to detect by the eye or comparison programs. However, the proposed method makes the change "zero" at the visual level and "real" at the programming level.

- Using "Tifinagh Language" as a security cover:

Using a digitally rare language with a unique geometric nature is considered an innovation in choosing the "cover text." The Tifinagh letters, with their design based on circles and intersecting lines, provide a larger "hiding space" in terms of width and spacing without raising suspicion, compared to Latin or Arabic scripts that are saturated with studies.

- Transforming RSA from an encryption tool to a "probabilistic distribution function":

The originality of this proposed method is found in the use of RSA not as a terminal security lock (considering the small N), but as a device for "randomizing" the data before it is inserted into the font values. This ensures that the software values injected

into FontLab do not follow any repetitive pattern, making attempts at "steganalysis" fail to find any logical relationship between the characters.

- Designing Fonts as a Security Tool (FontLab Security):

Transforming the font design program from a creative tool into a security engineering tool introduces the concept of the "security font" which carries two identities: a visual identity (text in Tifinagh Language) and a digital identity (data encrypted with RSA hidden in the Glyphs properties).

6.4- Future Works

There are a set of ideas can be worked on in the future such as:

- A larger number of fonts can be created so that we can hide more than 8-bits in one character of the carrier message.
- Unused font style (e.g. font color, underlined style, etc.) can be added along with those that have been used.
- The proposed hiding and un-hiding methods can be modified so that other languages would be used as message carriers.
- Other types of encryption algorithms can be applied within the proposed methods instead of the RSA algorithm.

6.5- References

1- Trilling, D. (2017) 'Digital security tips for journalists: Protecting sources and yourself', *Journalist's Resource*. Available at: https://journalistsresource.org/economics/digital-security-tips-protecting-sources-journalist/ (Accessed: 20 May 2025).

2- Buchanan, B. (2024) 'Cryptography', *Asecuritysite.com*. Available at: https://asecuritysite.com/cryptography/ (Accessed: 04 January 2025).

3- Blanco, J. (2018) 'Tifinagh alphabet: An ancient survivor in a modern multi-script environment', *Palace of Typographic Masonry*. Available at: https://palaceoftypographicmasonry.nl/sign/the-labyrinth-of-scripts/tifinagh-alphabet-an-ancient-survivor-in-a-modern-multi-script-environment/ (Accessed: 20 January 2024).

4- Erritali, M. and Amrouch, M. (2020) 'Search-based classification for offline Tifinagh alphabets recognition', in E.S.M. El-Alfy, J. Hemanth and N. Dey (eds.), *Advancements in computer vision applications in intelligent systems and multimedia technologies*, Hershey, PA: IGI Global, pp. 255–267. Available at: https://doi.org/10.4018/978-1-7998-4444-0.ch013 (Accessed: 25 May 2024).

5- López, P. et al. (2022) 'Cyber deception: State of the art, trends, and open challenges', *Journal of Open Information Technologies*, 10(9), pp. 126–134. Available at: https://doi.org/10.24412/2304-6701-2022-9-126-134 (Accessed: 20 May 2024).

6- Sadki, H. et al. (2014) 'A novel approach of Amazigh text steganography based on elliptic curve', *International Journal of Security and Its Applications*, 8(5), pp. 385–394. Available at: https://doi.org/10.14257/ijsia.2014.8.5.36 (Accessed: 01 May 2024).

7- Hall, P. and Park, S. (2025) 'The pitfalls of security by obscurity and what they mean for transparent AI', *Proceedings of the AAAI Conference on Artificial Intelligence*. Available at: https://ojs.aaai.org/index.php/AAAI/article/view/35022/3717 (Accessed: 20 May 2025).

8- Ristord, B. (2025) 'Security Through Obscurity: Valid Layer or Dangerous Illusion?', *Medium*, Available at: https://medium.com/@9772_2061/security-

through-obscurity-valid-layer-or-dangerous-illusion-3e0f45119869 (Accessed: 20 May 2025).

9- Kumar, M. (2025) 'Difference between cryptography and steganography: Data security and concealment techniques', *UpGrad*. Available at: https://www.upgrad.com/blog/difference-between-cryptography-and-steganography/ (Accessed: 20 May 2025).

10- Boneh, D. and Shoup, V. (2023) *A graduate course in applied cryptography*. Available at: https://toc.cryptobook.us/book.pdf (Accessed: 20 January 2024).

11- Omar, R. and Asaad, W. (2017) 'Cryptography and data security: An introduction', in *2017 International Conference on Engineering and Technology (ICET)*, pp. 1–6. IEEE.

12- Deena, G. and Rajendra, K. (2021) 'Sponge based lightweight cryptographic hash functions for IoT applications', in *2021 International Conference on Intelligent Technologies (CONIT)*, pp. 3–6. IEEE. Available at: https://doi.org/10.1109/CONIT51499.2021.9558489 (Accessed: 20 May 2024).

13- Majeed, M. et al. (2021) 'Text steganography techniques', *Mathematics*, 9(21), 2829. Available at: https://doi.org/10.3390/math9212829 (Accessed: 20 May 2024).

14- Alshahrani, H. and Weir, G. (2017) 'Hybrid Arabic text steganography', *International Journal of Computer and Information Technology*, vol. 6, no. 6, pp. 250–257.

15- Al-Anzi, H. et al. (2020) 'Functionality improved Arabic text steganography based on Unicode features', *Arabian Journal for Science and Engineering*, vol. 45, no. 7, pp. 5419–5430. Available at: https://doi.org/10.1007/s13369-020-04645-4 (Accessed: 20 May 2024).

16- Odeh, A. A. et al. (2013) 'Text steganography using language remarks', in *Proceedings of the American Society of Engineering Education (ASEE 2013)*.

17- Kumar, K. (2018) *Steganography: A data hiding technique*. Master's thesis, St. Cloud State University. Available at: https://www.researchgate.net/publication/49587597_Steganography-_A_Data_Hiding_Technique (Accessed: 20 May 2024).

18- Sadeghi, A. (2019) 'Digital watermarking techniques: challenges and applications: a review', *International Journal of Image, Graphics and Signal Processing*, vol. 11, no. 6, pp. 35–45. Available at: https://doi.org/10.5815/ijigsp.2019.06.05 (Accessed: 20 May 2024).

19- Shelke, R. et al. (2018) 'Digital watermarking techniques', *International Research Journal of Engineering and Technology*, vol. 5, no. 3, pp. 415–419. Available at: https://www.researchgate.net/publication/324717743_Digital_Watermarking_Techniques (Accessed: 20 May 2024).

20- Odeh, A. (2013) 'Text steganography using language remarks', in *ASEE Annual Conference & Exposition*, Northfield, VT, USA. Available at: https://peer.asee.org/text-steganography-using-language-remarks (Accessed: 20 May 2024).

21- Ali, A. and Seddike, H. (2013) 'New text steganography technique by using mixed-case font', *International Journal of Computer Applications*, vol. 62, no. 3, pp. 34–38. Available at: https://doi.org/10.5120/10078-4924 (Accessed: 20 May 2024).

22- El Rahman, S. (2019) 'Text steganography approaches using similarity of English font styles', *International Journal of Software Innovation (IJSI)*, vol. 7, no. 3, pp. 22–43. Available at: https://doi.org/10.4018/IJSI.2019070102 (Accessed: 20 May 2024).

23- Rahma, A. and George, L. E. (2013) 'Text steganography based on unicode of characters in multilingual', *International Journal of Engineering Research and Applications (IJERA)*, vol. 3, no. 4, pp. 1153–1165. Available at: https://www.ijera.com/papers/Vol3_issue4/CJ34011531165.pdf (Accessed: 20 May 2024).

24- Khlefa, A. and Al-Bakri, N. M. F. (2021) 'New text steganography method using the Arabic letters dots', *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 21, no. 3, pp. 1784–1793. Available at: https://doi.org/10.11591/ijeecs.v21.i3.pp1784-1793 (Accessed: 20 May 2024).

25- Ahvanooey, M. et al. (2019) 'Modern text hiding, text steganalysis, and applications: a comparative analysis', *Entropy*, vol. 21, no. 4, 355. Available at: https://doi.org/10.3390/e21040355 (Accessed: 20 May 2024).

26- Chang, C. and Clark, S. (2013) 'Practical linguistic steganography using contextual synonym substitution and a novel vertex coding method', in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 724–729. Association for Computational Linguistics. Available at: https://aclanthology.org/P13-2130/ (Accessed: 20 May 2024).

27- Kumar, A. (2010) 'Steganography: A data hiding technique', *International Journal of Computer Applications*, vol. 9, no. 7, pp. 24–27. Available at: https://doi.org/10.5120/1450-1964 (Accessed: 20 May 2024).

28- Curran, K. (2017) 'Text based steganography', *International Journal of Information Privacy Security and Integrity*, vol. 3, no. 1, pp. 44–51.

29- Naskar, M. (2019) 'Uses of typography - different types of fonts' [presentation slides], *Academia.edu*. Available at: https://www.academia.edu/39929819/Uses_of_typography_Different_types_of_fonts (Accessed: 20 May 2024).

30- Sahni, H. (2023) 'Types of fonts, examples, and when to use them', *Piktochart*. Available at: https://piktochart.com/blog/types-of-fonts/ (Accessed: 20 May 2024).

31- Martin, J. (2016) 'Limitations of using OCR for file classification', *IDM*. Available at: https://idm.net.au/article/0011231-limitations-using-ocr-file-classification (Accessed: 20 May 2024).

32- Scheichelbauer, R. E. (2013) 'Font hacking', presentation at TYPO Berlin 2013: Shift, Berlin, Germany. Available at: https://www.slideshare.net/robertscri/font-hacking-24476200 (Accessed: 20 May 2024).

33- Shirali-Shahreza, M. (2008) 'Text steganography by changing the appearance of characters', in *2008 Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pp. 80–83. IEEE. Available at: https://doi.org/10.1109/IIH-MSP.2008.45 (Accessed: 20 May 2024).

34- Diemand-Yauman, M. et al. (2011) 'Fortune favors the bold (and the italicized): effects of disfluency on educational outcomes', *Cognition*, vol. 118, no. 1, pp. 111–115. Available at: https://doi.org/10.1016/j.cognition.2010.09.012 (Accessed: 20 May 2024).

35- Jońca, P. (2014) 'FontLab Studio', *FontLab*. Available at: https://www.fontlab.com/ar/font-editor/fontlab-studio-5/ (Accessed: 20 May 2024).

36- Asjad, S. (2018) 'The RSA algorithm', *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/computer-networks/rsa-algorithm-cryptography/ (Accessed: 20 May 2024).

37- Barker, E. B. and Roginsky, A. (2020) *Recommendation for key management: Part 1 – general* (NIST Special Publication 800-57 Part 1 Rev. 5). U.S. Department of Commerce, National Institute of Standards and Technology. Available at: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf (Accessed: 20 May 2024).

38- Luo, Z. J. and Sun, Y. (2023) 'Understanding the RSA algorithm', *Rider University Student Journal of Sciences and Humanities*, vol. 1, no. 1, art. 7. Available at: https://digitalcommons.rider.edu/rusj/vol1/iss1/7/ (Accessed: 20 May 2024).

39- Goundar, S. (2012) 'Research methodology and research method: methods commonly used by researchers', in *Research methodology for academic researchers*. International Scholarly and Scientific Research & Innovation, pp. 27–73.

40- Baker, S. (2014) 'In Microsoft Word, how is font size measured?', *Quora*. Available at: https://www.quora.com/In-Microsoft-Word-how-is-font-size-measured (Accessed: 20 May 2024).

41- Shapiro, J. (2017) *Visual Basic.NET*. McGraw-Hill/Osborne. Available at: https://www.amazon.com/Visual-Basic-NET-Joseph-Shapiro/dp/0072215439 (Accessed: 20 May 2024).

42- Al-Nofaie, S. (2021) 'Enhancing Arabic text steganography for personal usage utilizing pseudo-spaces', *Journal of King Saud University - Computer and Information Sciences*, vol. 33, no. 8, pp. 963–974. Available at: https://doi.org/10.1016/j.jksuci.2019.06.010 (Accessed: 20 May 2024).

43- Taha, A. et al. (2020) 'A high capacity algorithm for information hiding in Arabic text', *Journal of King Saud University - Computer and Information Sciences*, vol. 32, no. 6, pp. 658–665. Available at: https://doi.org/10.1016/j.jksuci.2019.01.015 (Accessed: 20 May 2024).

44- Michelucci, D. et al. (1999) 'Robustness issues', in *Discrete Geometry for Computer Imagery: 8th International Conference, DGCI 1999*, Springer, Berlin, Heidelberg, pp. 423–434. Available at: https://doi.org/10.1007/3-540-46784-X_46 (Accessed: 20 May 2024).

45- Alabish, A., et al. (2013) 'A universal lexical steganography technique', *International Journal of Computer and Communication Engineering*, 2(2), pp. 153–157. Available at: https://doi.org/10.7763/IJCCE.2013.V2.159.

Appendix A:

- **An in-depth algorithm proposed for the hiding process**

The proposed method can be illustrated according to the following Algorithm:

Algorithm: Tifinagh Steganography

Input : Carrier message (CM) which is a text written in Tifinagh and Secret Message (SM) which is a text written in any human language.

Output : The carrier message (CM) embedded with a secret message (SM).

Start

1) If (the SM is written in any human language) Then

The length of the CM must be at least twice or more the length of the SM.

End if

2) I = The first character of the secret message (SM).

3) Repeat (for all characters of the secret message).

4) The Ith character of the secret message (SM)is converted into UTF8Encoding.

5) The Ith byte of the UTF8Encoding is converted into decimal.

6) The generated decimal number is encoded with the well-known RSA algorithm.

7)) The encoded decimal number of the Jth portion is converted into binary code.

8) if(the UTF8.COUNT>0 go to number 5.

9)Divide the set of generated binary bits into 8-bit portions.

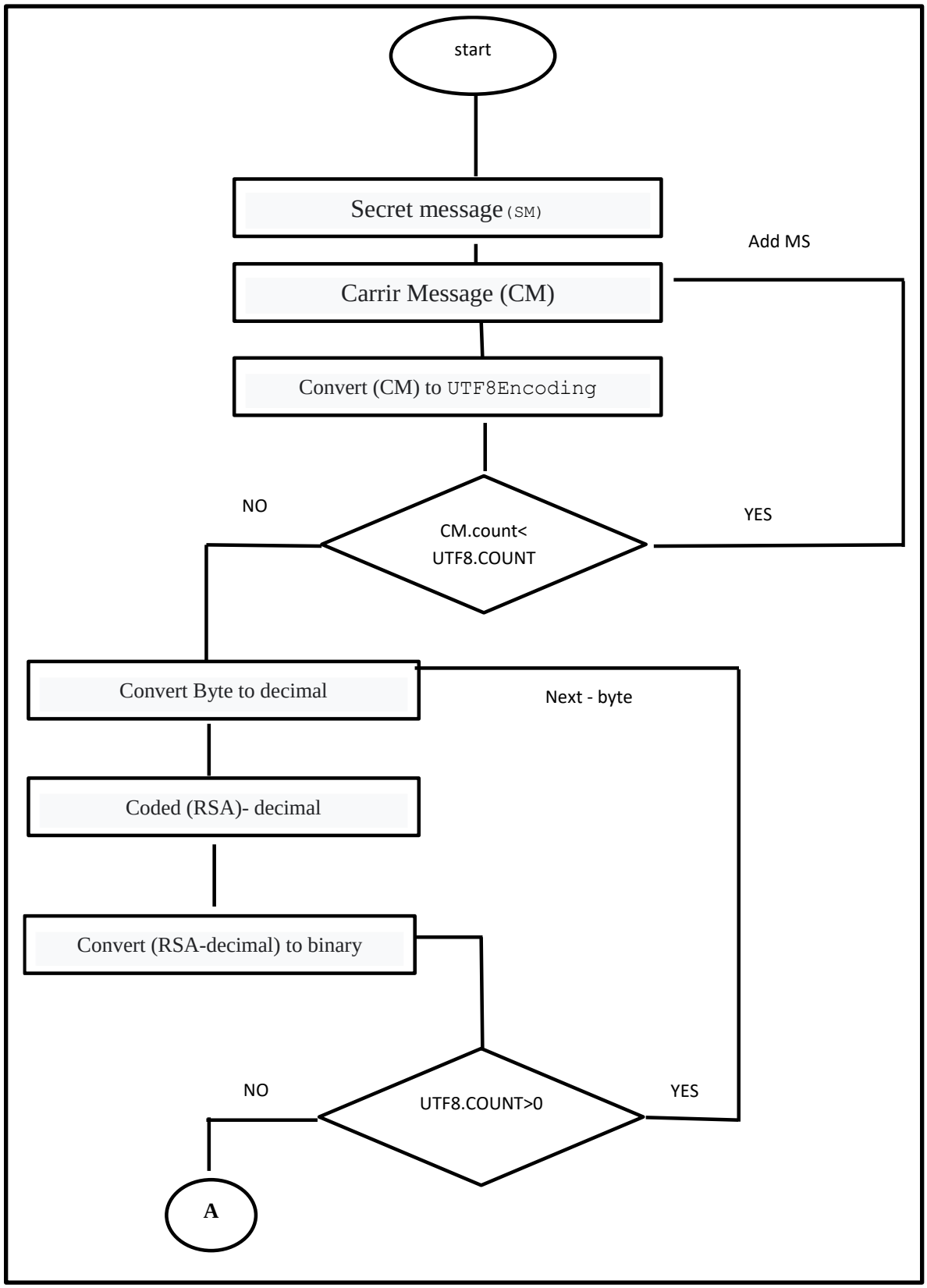
10) J = The first portion of binary bits (x-bin).

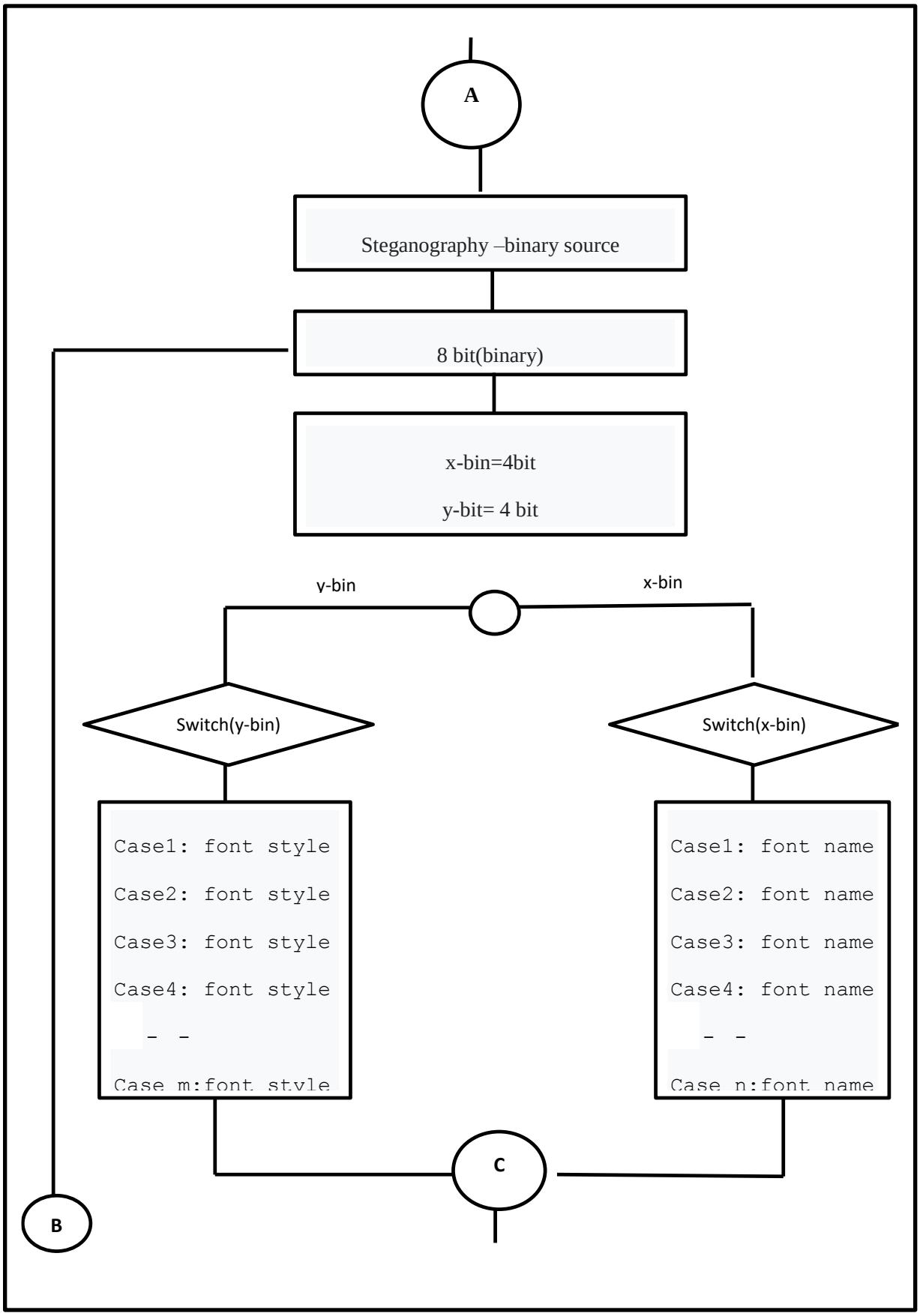
11) Repeat (for all portions of binary bits).

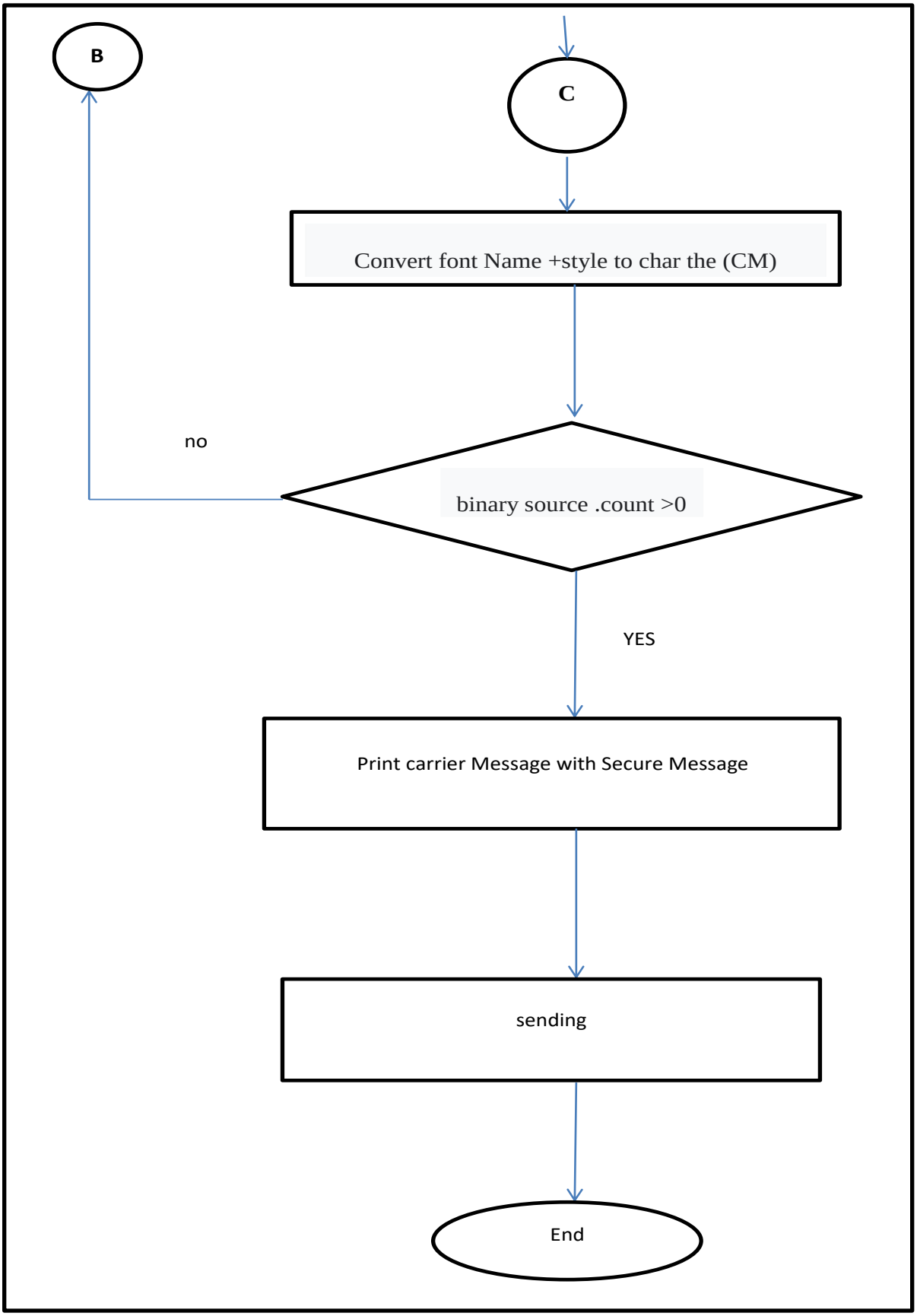
12) Select Pj from The Jth portion. where k=0 to the font Name array length
 subSwich(Next Pj)
{ Case 1 (1st 4-bits): Font_name(Ft01) ; Break;
{ Case 2 (2st 4-bits): Font_name (ft02); Break;
.
Case m (mth 4-bits): Size (Ftm) ; Break;
Fsm; Break;
Default: Break;
}
Break;
}
Until (The last portion of binary code)
13) J = The second portion of binary bits (y-bin).
14) Repeat (for all portions of binary bits).
15) Select Pj from The Jth portion. where k=0 to the font Style array length
16) Switch(Pj)
{ Case 1 (1st 4-bits): Font_Type (Ft1);
 subSwich(Next Pj)
{ Case 1 (1st 4-bits): Font_Size (Fs1) + R; Break;
{ Case 2 (2st 4-bits): Font_Size (Fs1) + L; Break;

Case m (mth 4-bits): Size (Fsm) +Rm and Lm; Break;
Fsm+ftm; Break;
Default: Break;
}
Break;
}
Until (The last portion of binary code)
17) Tokenize the CM into a set of characters
Until (The last character of the SM)
End. {End of the Algorithm}

- **An in-depth flowchart proposed for the concealment process**




A
Steganography –binary source
8 bit(binary)
x-bin=4bit
y-bit= 4 bit
y-bin
x-bin
Switch(y-bin)
Switch(x-bin)
Case1: font style
Case2: font style
Case3: font style
Case4: font style
Case m:font style
Case1: font name
Case2: font name
Case3: font name
Case4: font name
Case n:font name
C
B


B
C
Convert font Name +style to char the (CM)
no
binary source .count >0
YES
Print carrier Message with Secure Message
sending
End

Appendix B:

- **An in-depth algorithm proposed for the uncovering process**

The proposed extraction method can be illustrated according to the following Algorithm:

```
Algorithm: Extraction of Secret Message
Input: The carrier message (CM) embedded with a secret message (SM).
Output: The Secret Message (SM) written in any human language.
Start
I = The first character of the Carrier message (CM).
 Switch(I j )
{ Case 1 Font_Type (Ft1) :(1st 4-bits);
 subSwich(I j)
                          { Case 1  (Font_Size (Fs1) + R ):  1st 4-bits; Break;
                           { Case 2 (Font_Size (Fs1) + L) : 2st 4-bits; Break;
                             .
                               Case m (Size (Fsm) +Rm and Lm):  mth 4-bits  ; Break;
                               Fsm+ftm; Break;
                               Default:   Break;
                                 }
Break;
}
{ Case 2 Font_Type (Ft1) :(1st 4-bits);
 subSwich(I j)
                          { Case 1  (Font_Size (Fs1) + R ):  1st 4-bits; Break;
                           { Case 2 (Font_Size (Fs1) + L) : 2st 4-bits; Break;
                             .
                               Case m (Size (Fsm) +Rm and Lm):  mth 4-bits  ; Break;
                               Fsm+ftm; Break;
                               Default:   Break;
                                 }
Break;
}
```

```
{ Case n Font_Type (Ft1) :(1st 4-bits);
  subSwich(I j)
                              { Case 1  (Font_Size (Fs1) + R ):  1st 4-bits; Break;
                                { Case 2 (Font_Size (Fs1) + L) : 2st 4-bits; Break;
                                  .
                                    Case m (Size (Fsm) +Rm and Lm):  mth 4-bits  ; Break;
                                    Fsm+ftm; Break;
                                    Default:   Break;
                                      }
Break;
}
```

4) Repeat (for all characters of the Carrier message).

5) Extracted binary Code from The Ith character of the Carrier message (CM).

6) J = The first part of binary bits.

7) The Jth (8 bits in length) of the binary code is converted into decimal number.

8) The decimal number is decoded with the well-known RSA algorithm.

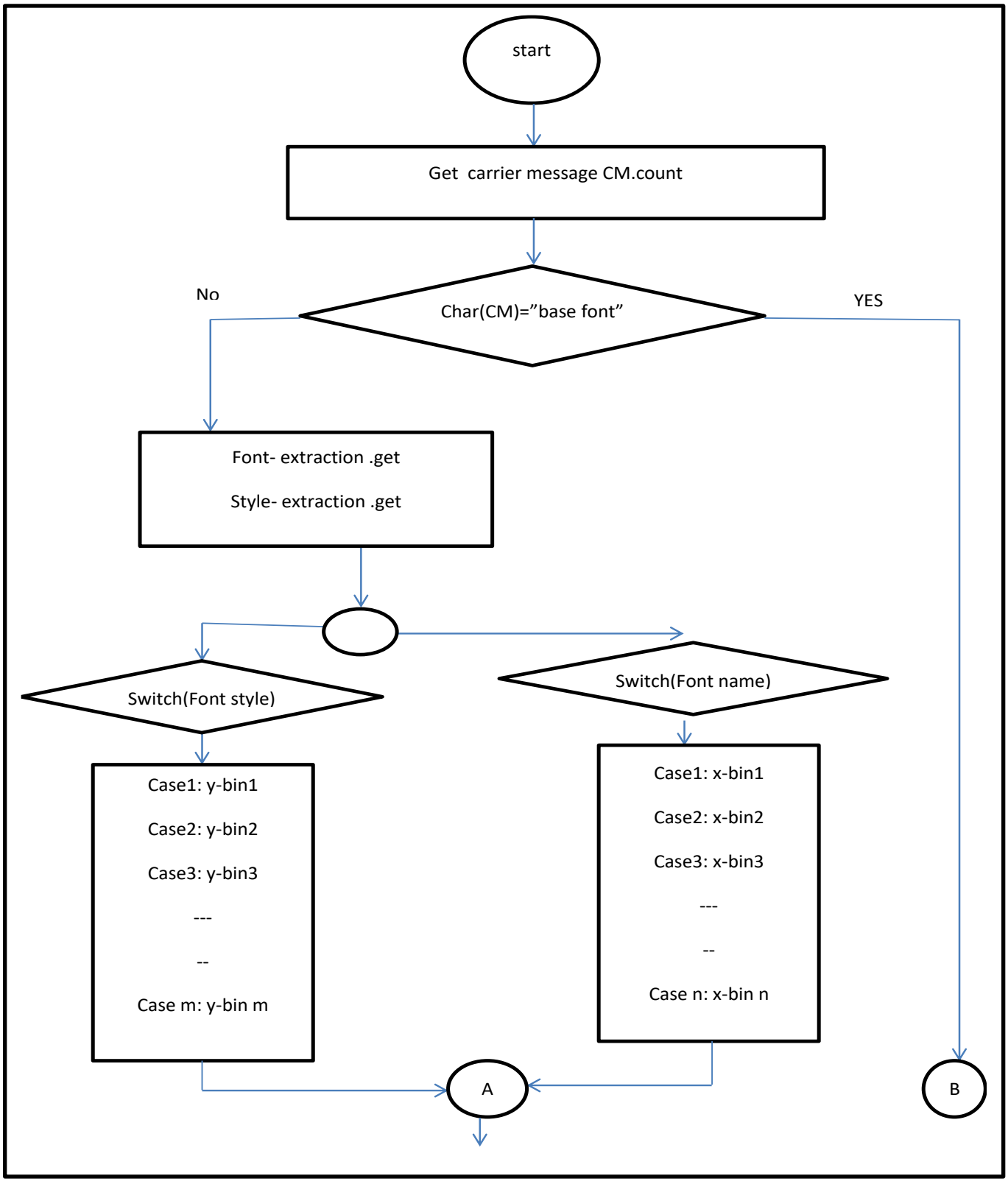
9) The decoded decimal number of the Jth is converted into binary code.

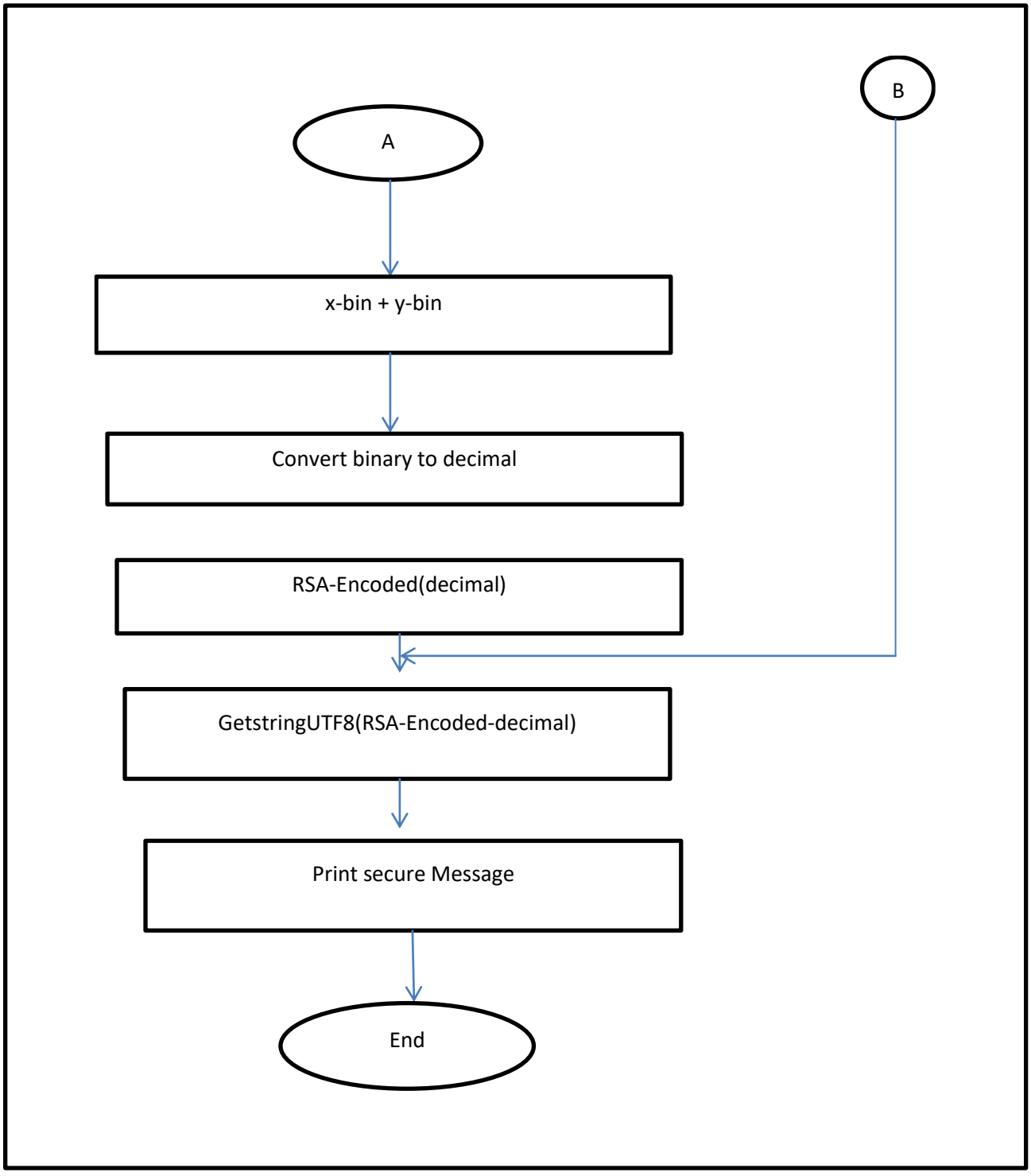
10) Tokenize the CM into a set of characters

Until (The last character of the CM)

End. {End of the Algorithm}

- **An in-depth flowchart proposed for the uncovering process**




B
A
x-bin + y-bin
Convert binary to decimal
RSA-Encoded(decimal)
GetstringUTF8(RSA-Encoded-decimal)
Print secure Message
End

- **The code written in VB.net for the algorithms of hiding and unhiding:**

```
Imports System.Numerics

Imports System.Runtime.InteropServices.JavaScript.JSType

Imports System.Windows.Forms.VisualStyles.VisualStyleElement

Imports Windows.Win32.System

Imports System.Reflection.Metadata.Document

Imports System.Text

Imports System.Text.Unicode

Imports System.IO

Imports System.Reflection.Emit

Imports Syncfusion.Pdf

Imports Syncfusion.Pdf.Graphics

Imports System.Security.Cryptography

Public Class Form1

    Public one_binary As String

    Public bin_sucer As New List(Of String)()

    Public bin_test As New List(Of String)()

    Public bin_desucer As New List(Of String)()

    Public dec_desucer As New List(Of Integer)()

    Public RSA_desucer As New List(Of Integer)()

    Public RCH_FILE As RichTextBox

    Public RSA_secure As New List(Of Integer)()

    Public decimal_sucer As New List(Of Integer)()

    Public x_bin, y_bin As String
```

```
        Public myString As String
        Public cl_bin As New binary_value
        Public font_clin As New casse_xBin
        Public style_clin As New case_Ybin
        Public font_extra As New extry_font
        Public extra_styl As New extry_style
        Public client As New coding_txt
        Public font_name As String
        Public countProc As Integer
        Dim ofd As OpenFileDialog = New OpenFileDialog
        Dim fileReader As String
    Public Shared Function RSA_code(ByVal p As Integer, ByVal q As Integer, ByVal
message     As Integer) As Integer
            ' calculating n
            Dim n As Integer
            n = p * q
            ' calculating totient, t
            Dim t As Integer
            t = (p - 1) * (q - 1)
            ' selecting public key, e
            Dim e As Integer
            For i = 2 To t - 1
                If GCD(i, t) = 1 Then
                    e = i
                    Exit For
```

```
    End If
Next
'  selecting private key, d
Dim j As Object
j = 0
Dim mes As Integer
Dim ct As Integer
Dim d1 As Integer
While True
    If (j * e) Mod t = 1 Then
        d1 = j
        Exit While
    End If
    j += 1
End While
'  performing encryption
ct = (message ^ e) Mod n
'System.Numerics.BigInteger.TryParse(ct, (message ^ e) Mod n)
Return ct
'  performing decryption
' Dim buf As System.Numerics.BigInteger = BigInteger.Pow(ct, d1)
' mes = (ct ^ d1) Mod n
' mes = buf Mod n
'System.Numerics.BigInteger.TryParse(mes, (ct ^ d1) Mod n)
' TextBox2.Text = mes
```

```
End Function
Public Shared Function GCD(a As Integer, b As Integer) As Integer
    If a = 0 Then
        Return b
    End If
    While b <> 0
        If a > b Then
            a -= b
        Else
            b -= a
        End If
    End While
    Return a
End Function
Public Function Dec2Bin(ByVal num As Long) As String
    Dim i As Long, sRetVal As String
    ' If num < 0 Or num > 255 Then
    'MsgBox("This function only accepts numbers from 0 to 255")
    'End If
    For i = 0 To 7
        sRetVal = IIf(num And (2 ^ i), "1", "0") & sRetVal
    Next
   Dec2Bin = sRetVal
End Function
Public Function Bin2Dec(ByVal sBin As String) As Long
```

```
        Dim i As Long
        Dim nRetVal As Long
        For i = 0 To Len(sBin) - 1
            If Mid$(sBin, Len(sBin) - i, 1) = "1" Then
                nRetVal = nRetVal + 2 ^ i
            End If
        Next
        Bin2Dec = nRetVal
    End Function
    'Binary To Decimal
    ' ================
    Private Sub Button4_Click(sender As Object, e As EventArgs) Handles
Button4.Click
        Dim s, phrase, ch As String
        Dim pos, i, jx As Integer
        Dim fontname, fontstyl As String
        Dim fontsize As Object
        pos = carrir_message.SelectionStart
        phrase = "□"
        s = carrir_message.Text
        carrir_secure.Text = carrir_message.Text
        RichTextBox1.Text = carrir_message.Text
        extra_secur()
        chang_font.Enabled = False
        ' extra_secur_arabic()
        For cu = 0 To bin_sucer.Count - 1
```

```
            steganograph(cu)
            x_bin = cl_bin.getx_bin
            y_bin = cl_bin.gety_bin
            font_clin.setx_bin(x_bin)
            fontname = font_clin.gety_bin()
            style_clin.sety_bin(y_bin)
            fontsize = style_clin.getfont_siz()
            fontstyl = style_clin.getstyle()
            carrir_secure.SelectionStart = cu
            RichTextBox1.SelectionStart = cu
            carrir_secure.SelectionLength = phrase.Length
            RichTextBox1.SelectionLength = phrase.Length
            If fontstyl = "Italic" Then
               carrir_secure.SelectionFont = New Font(fontname, fontsize,
FontStyle.Italic)
               'Label9.Text = carrir_secure.SelectionBackColor = Nothing
               'Label6.Text = carrir_secure.SelectionColor = Color.Black
               If Val(carrir_secure.SelectionFont.Size.ToString) = (v + 0.1) Then
                  RichTextBox1.SelectionFont = New Font(fontname, (v + 0.1) + 0.4,
FontStyle.Italic)
               ElseIf Val(carrir_secure.SelectionFont.Size.ToString) = (v + 0.15) Then
                  RichTextBox1.SelectionFont = New Font(fontname, (v + 0.15) + 0.85,
FontStyle.Italic)
               ElseIf Val(carrir_secure.SelectionFont.Size.ToString) = (v + 0.2) Then
                  RichTextBox1.SelectionFont = New Font(fontname, (v + 0.2) + 1.3,
FontStyle.Italic)
               ElseIf Val(carrir_secure.SelectionFont.Size.ToString) = (v + 0.25) Then
```

```
        RichTextBox1.SelectionFont = New Font(fontname, (v + 0.25) + 1.75,
FontStyle.Italic)

                ElseIf Val(carrir_secure.SelectionFont.Size.ToString) = (v + 0.3) Then

                    RichTextBox1.SelectionFont = New Font(fontname, (v + 0.3) + 2.2,
FontStyle.Italic)

                ElseIf Val(carrir_secure.SelectionFont.Size.ToString) = (v + 0.35) Then


                    RichTextBox1.SelectionFont = New Font(fontname, (v + 0.35) + 2.65,
FontStyle.Italic)

                End If

            End If

            If fontstyl = "Bold" Then

                carrir_secure.SelectionFont = New Font(fontname, fontsize, FontStyle.Bold)

                If Val(carrir_secure.SelectionFont.Size.ToString) = (v + 0.1) Then

                    RichTextBox1.SelectionFont = New Font(fontname, (v + 0.1) + 0.4,
FontStyle.Bold)

                ElseIf Val(carrir_secure.SelectionFont.Size.ToString) = (v + 0.15) Then

                    RichTextBox1.SelectionFont = New Font(fontname, (v + 0.15) + 0.85,
FontStyle.Bold)

                ElseIf Val(carrir_secure.SelectionFont.Size.ToString) = (v + 0.2) Then

                    RichTextBox1.SelectionFont = New Font(fontname, (v + 0.2) + 1.3,
FontStyle.Bold)

                ElseIf Val(carrir_secure.SelectionFont.Size.ToString) = (v + 0.25) Then

                    RichTextBox1.SelectionFont = New Font(fontname, (v + 0.25) + 1.75,
FontStyle.Bold)

                ElseIf Val(carrir_secure.SelectionFont.Size.ToString) = (v + 0.3) Then

                    RichTextBox1.SelectionFont = New Font(fontname, (v + 0.3) + 2.2,
FontStyle.Bold)
```

```
                ElseIf Val(carrir_secure.SelectionFont.Size.ToString) = (v + 0.35) Then

                    RichTextBox1.SelectionFont = New Font(fontname, (v + 0.35) + 2.65,
FontStyle.Bold)

                End If    End If

            If fontstyl = "Regular" Then

                carrir_secure.SelectionFont = New Font(fontname, fontsize,
FontStyle.Regular)

                If Val(carrir_secure.SelectionFont.Size.ToString) = (v + 0.1) Then

                    RichTextBox1.SelectionFont = New Font(fontname, (v + 0.1) + 0.4,
FontStyle.Regular)

                ElseIf Val(carrir_secure.SelectionFont.Size.ToString) = (v + 0.15) Then

                    RichTextBox1.SelectionFont = New Font(fontname, (v + 0.15) + 0.85,
FontStyle.Regular)

                ElseIf Val(carrir_secure.SelectionFont.Size.ToString) = (v + 0.2) Then

                    RichTextBox1.SelectionFont = New Font(fontname, (v + 0.2) + 1.3,
FontStyle.Regular)

                ElseIf Val(carrir_secure.SelectionFont.Size.ToString) = (v + 0.25) Then

                    RichTextBox1.SelectionFont = New Font(fontname, (v + 0.25) + 1.75,
FontStyle.Regular)

                ElseIf Val(carrir_secure.SelectionFont.Size.ToString) = (v + 0.3) Then

                    RichTextBox1.SelectionFont = New Font(fontname, (v + 0.3) + 2.2,
FontStyle.Regular)

                ElseIf Val(carrir_secure.SelectionFont.Size.ToString) = (v + 0.35) Then

                    RichTextBox1.SelectionFont = New Font(fontname, (v + 0.35) + 2.65,
FontStyle.Regular)

                End If

            End If

            'carrir_secure.AppendText(carrir_message.SelectedText)
```

```
            ' Label3.Text = carrir_secure.SelectionFont.Name
            ' Label5.Text = carrir_secure.SelectedText
        Next
    End Sub
    Private Sub Button5_Click(sender As Object, e As EventArgs) Handles
Button5.Click
    End Sub
    Public Sub steganograph(ByVal cu As Integer)
        Dim k, i As Integer
        one_binary = bin_sucer(cu)
        k = 0
        i = 0
        x_bin = Nothing
        y_bin = Nothing
        Do While (i < 4)
            x_bin = x_bin & one_binary(i)
            i += 1
        Loop
        Do While (i < 8)
            y_bin = y_bin & one_binary(i)
            i += 1
        Loop
        cl_bin.setx_bin(x_bin)
        cl_bin.sety_bin(y_bin)
    End Sub
    Public Sub extra_secur()
```

```
        Dim k As String
        Dim utf8 As UTF8Encoding = New UTF8Encoding(True, True)
        client.setchar(txtscure_message.Text)
        myString = client.getchar
        'Dim s As String = "nasser"
        ' Dim bytes As Byte() = New Byte(utf8.GetByteCount(s) +
utf8.GetPreamble().Length - 1) {}
        Dim bytes As Byte() = New Byte(utf8.GetByteCount(myString) +
utf8.GetPreamble().Length - 1) {}
        Array.Copy(utf8.GetPreamble(), bytes, utf8.GetPreamble().Length)
        utf8.GetBytes(myString, 0, myString.Length, bytes, utf8.GetPreamble().Length)
        k = carrir_message.Text
        If k.Count - 1 < bytes.Count - 1 Then
            MsgBox("الرسالة السرية أعلى من الرسالة الناقلة")
            Exit Sub
        End If
        For j = 0 To bytes.Count - 1
            decimal_sucer.Add(bytes(j))
            RSA_secure.Add(RSA_code(17, 15, decimal_sucer(j)))
            bin_sucer.Add(Dec2Bin(RSA_secure(j)))
        Next
        ' Dim s2 As String = utf8.GetString(bytes, 0, bytes.Length)
        '  For i = 0 To bytes.Count - 1
        '   bin_sucer.Add(bytes(i))
        '   Next
        'TextBox1.Text = s2
```

```
            'اعادة الى الاصل

            '   Dim ff As Byte() = New Byte(utf8.GetByteCount(myString) + 2) {}

            '  For i = 0 To bin_sucer.Count - 1

            'ff(i) = bin_sucer(i)

            ' Next

            ' Dim s3 As String = utf8.GetString(ff, 0, ff.Length)

        End Sub

        Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load

            ' carrir_message.Font = New Font("FT001", 12)

        End Sub

        Private Sub Button3_Click(sender As Object, e As EventArgs) Handles
Button3.Click

            SaveFile()

        End Sub

        Public Shared Function RSA_uncode(ByVal p As Integer, ByVal q As Integer,
ByVal message As Integer) As Integer

            '  calculating n

            Dim n As Integer

            n = p * q

            '  calculating totient, t

            Dim t As Integer

            t = (p - 1) * (q - 1)

            '  selecting public key, e

            Dim e As Integer

            For i = 2 To t - 1

                If GCD(i, t) = 1 Then
```

```
        e = i
        Exit For
    End If
Next
'  selecting private key, d
Dim j As Object
j = 0
Dim mes As Integer
Dim ct As Integer
Dim d1 As Integer
While True
    If (j * e) Mod t = 1 Then
        d1 = j
        Exit While
    End If
    j += 1
End While
'  performing encryption
' ct = (message ^ e) Mod n
'System.Numerics.BigInteger.TryParse(ct, (message ^ e) Mod n)
'Return ct
'  performing decryption
Dim buf As System.Numerics.BigInteger = BigInteger.Pow(message, d1)
' mes = (ct ^ d1) Mod n
mes = buf Mod n
```

```
            System.Numerics.BigInteger.TryParse(mes, (ct ^ d1) Mod n)
            Return mes
        End Function
        Public Sub extra_secur_arabic()
            Dim client As New coding_txt
            client.setchar(txtscure_message.Text)
            myString = client.getchar
            Dim vOut() As Byte =
System.Text.UTF8Encoding.Default.GetBytes(myString)
            For j = 0 To vOut.Count - 1
                decimal_sucer.Add(vOut(j))
                RSA_secure.Add(RSA_code(17, 15, decimal_sucer(j)))
                bin_sucer.Add(Dec2Bin(RSA_secure(j)))
            Next
        End Sub
        Private Sub chang_font_SelectedIndexChanged(sender As Object, e As
EventArgs) Handles chang_font.SelectedIndexChanged
            v = chang_font.Text
            carrir_message.Font = New Font("FT001", v)
            carrir_secure.Font = New Font("FT001", v)
            RichTextBox1.Font = New Font("FT001", v)
        End Sub
        Private Sub Button7_Click(sender As Object, e As EventArgs)
            txtscure_message.Text = Nothing
            carrir_secure.Text = Nothing
            'VARIABLE
```

```
            bin_sucer.Clear()
            decimal_sucer.Clear()
            RSA_secure.Clear()
            RSA_desucer.Clear()
            bin_desucer.Clear()
            dec_desucer.Clear()
            RSA_desucer.Clear()
            chang_font.Enabled = True
        End Sub
        Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
            'Dim fileReader As String
            'fileReader = My.Computer.FileSystem.ReadAllText("C:\test.txt")
            ProgressBar1.Maximum = 100
            countProc = 1
            'carrir_message.Text = fileReader
            ofd.DefaultExt = "txt"
            ofd.FileName = "defaultname"
            ofd.InitialDirectory = "c:\"
            ofd.Filter = "All files|*.*|Text files|*.txt"
            ofd.Title = "Select file"
            If ofd.ShowDialog() <> DialogResult.Cancel Then
                fileReader = My.Computer.FileSystem.ReadAllText(ofd.FileName,
              System.Text.Encoding.UTF8)
            Else
                Exit Sub
```

```
            End If
            Timer1.Enabled = True
        End Sub
        Private Sub Button7_Click_1(sender As Object, e As EventArgs) Handles
Button7.Click
            txtscure_message.Text = Nothing
            carrir_secure.Text = Nothing
            carrir_message.Text = Nothing
            'VARIABLE
            bin_sucer.Clear()
            decimal_sucer.Clear()
            RSA_secure.Clear()
            RSA_desucer.Clear()
            bin_desucer.Clear()
            dec_desucer.Clear()
            RSA_desucer.Clear()
            chang_font.Enabled = True
        End Sub
        Public Sub SaveFile()
            Dim myStream As Stream
            Dim saveFileDialog1 As New SaveFileDialog()
            saveFileDialog1.Filter = "rtf files (*.rtf)|*.rtf"
            saveFileDialog1.FilterIndex = 2
            saveFileDialog1.RestoreDirectory = True
            If saveFileDialog1.ShowDialog() = DialogResult.OK Then
```

```
            myStream = saveFileDialog1.OpenFile()
            If (myStream IsNot Nothing) Then
                RichTextBox1.SaveFile(myStream,
RichTextBoxStreamType.RichNoOleObjs)
                ' Code to write the stream goes here.
                myStream.Close()
            End If
        End If
    End Sub
    Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles
Timer1.Tick
        ProgressBar1.Value = ProgressBar1.Value + 1
        Label5.Text = "Process" & countProc & "%Complete"
        countProc += 1
        If countProc > 100 Then
            carrir_message.Text = fileReader
            ProgressBar1.Value = 0
            Label5.Text = ""
            Timer1.Enabled = False
        End If
    End Sub
End Class
```