

Table of Contents

1. Group Members	3
2. Project Title	3
3. Abstract	3
4. Introduction.....	3
5. Model and Implementation	4
5.1 User Interface.....	4
5.2. Methods	5
5.2.1. UI Methods.....	5
5.2.2. Steganography Methods	6
5.2.3. Helper Methods.....	7
5.2.4. Caesar Cipher Methods	7
5.2.4. Key Cryptography Methods.....	8
5.3. Steganography Algorithm Implementation.....	9
5.4. Caesar Cipher Algorithm Implementation	10
5.5. Key Cryptography Algorithm Implementation.....	10
6. Results and Analysis	11
6.1. Analysis.....	17
6.1.1. McAfee Online Tool for Steganography Detection	17
6.1.2. Noise Analysis.....	18
6.1.3. Principal Component Analysis	19
6.1.4. Own Analysis Tool (Python).....	21
6.2. Results	22
7. Summary and Conclusions	23
8. References & Resources.....	24

List of Shapes and Images

Images	Page
Image 5.1.1 User Interface	

1. Group Members

Oğulcan Topsakal

2. Project Title

Application of hiding crypted text messages into image (steganography).

3. Abstract

In this developing digital world in order to keep confidential information confidential, there are too many peoples work. They are using different techniques. In this project, I will explain one of these techniques which is steganography and I will develop an application on it. The application will embed the message that is requested to be hidden in the any picture. Main purpose of this application is hiding the message to be transmitted in an image.

4. Introduction

Cryptography is the study of secure communications techniques that allow only the sender and intended recipient of a message to view its contents. The term is derived from the Greek word kryptos, which means hidden. It is closely associated to encryption, which is the act of scrambling ordinary text into what's known as ciphertext and then back again upon arrival. In addition, cryptography also covers the obfuscation of information in images using techniques such as microdots or merging. Ancient Egyptians were known to use these methods in complex hieroglyphics, and Roman Emperor Julius Caesar is credited with using one of the first modern ciphers.

On the other hand, steganography is the technique of hiding secret data within an ordinary, non-secret, file or message in order to avoid detection; the secret data is then extracted at its destination. The use of steganography can be combined with encryption as an extra step for hiding or protecting data. The word steganography is derived from the Greek words steganos (meaning hidden or covered) and the Greek root graph (meaning to write).

Steganography is distinct from cryptography but using both together can help improve the security of the protected information and prevent detection of the secret communication. If steganographically-hidden data is also encrypted, the data may still be safe from detection -- though the channel will no longer be safe from detection. There are advantages to using steganography combined with encryption over encryption-only communication.

This project is built on these two terms. We take the text that the user wants to hide and hide it in the image as a result of encryption. In the next title of the report, detailed information about how these procedures are carried out will be given.

5. Model and Implementation

We will examine the project in two different ways, frontend and backend. Frontend can name as user interface and it includes buttons, picture boxes, drop downs and texts. Which they all are what user sees on screen. All the magic happens in backend part. Backend includes all steganography and cryptography algorithms. Which only can see by developer.

5.1 User Interface

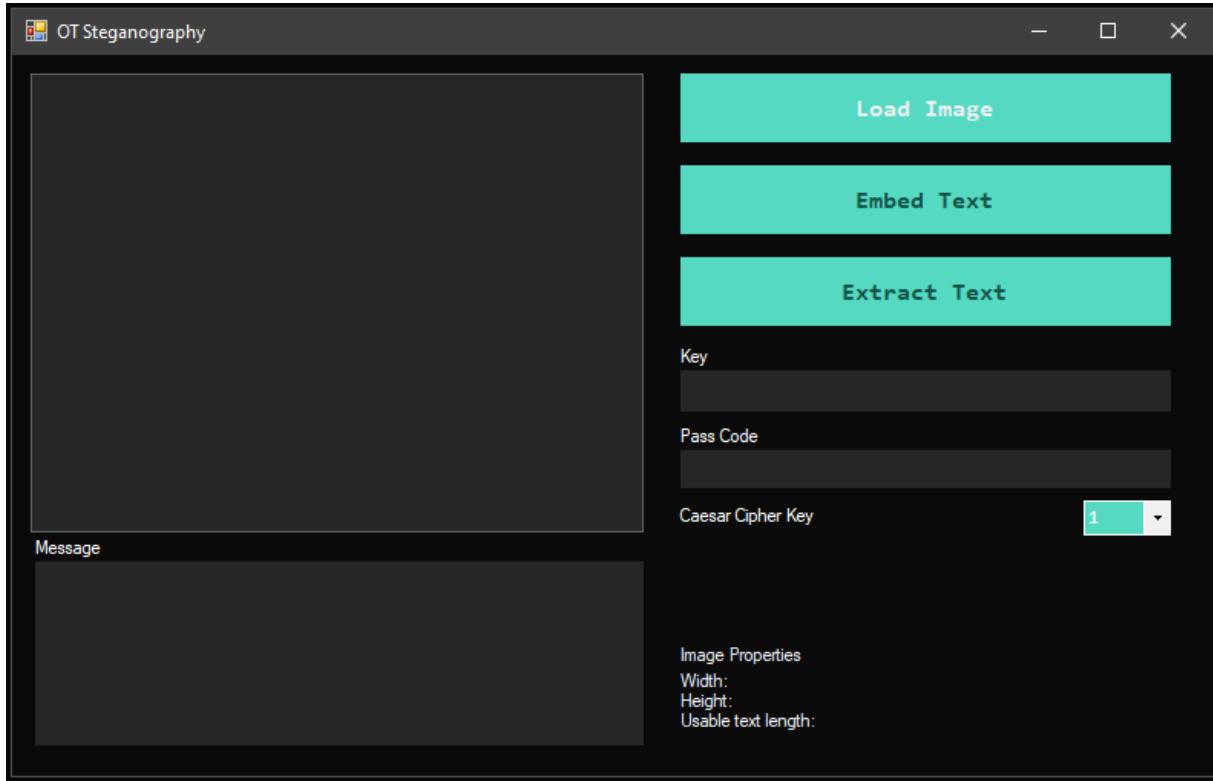


Image 5.1.1 User Interface

On Image 5.1.1. you can see certain elements on programs user interface. They are:

- Picture Box
- Load Image Button
- Embed Text Button
- Extract Text Button
- Key Text Input Field
- Pass Code Text Input Field
- Caesar Cipher Key Dropdown
- Image Properties Text Area

Picture Box: It is used for to show user loaded image.

Load Image Button: It brings File Dialog to user choose image to work on.

Embed Text Button: It does all embedding job. Taking data from larger text area and embedding data.

Extract Data: It brings hidden message to user interface with decryption key.

Key Text Input Field: To message extraction, generated key should be put here.

Pass Code Text Input Field: Both embedding and extracting text this field requires text which arranged by user.

Caesar Cipher Key Dropdown: Selector of Caesar cipher key.

Image Properties Text Area: It shows the certain specs to user work on.

5.2. Methods

Sample examination of methods:

Method Name	“Example Name”
Return Value and Function Type	“String/private”
Description	“Example Description”

5.2.1. UI Methods

Method Name	Form1_Load(object sender, EventArgs e)
Return Value and Function Type	-/ private void
Description	Calls at start frame of project. Initializing dropdown values under this method.

Method Name	textBox1_TextChanged(object sender, EventArgs e)
Return Value and Function Type	-/ private void
Description	Calls every time ‘Message Text Area’ text changes. Purpose of this method is controlling embed button’s state.

Method Name	textBox3_TextChanged(object sender, EventArgs e)
Return Value and Function Type	-/ private void
Description	Calls every time 'Pass Code' text changes. Purpose of this method is controlling embed button's state.

Method Name	buttonExtractText_Click(object sender, EventArgs e)
Return Value and Function Type	-/private void
Description	Calls when 'Extract Text' button clicked.

Method Name	buttonEmbeddText_Click(object sender, EventArgs e)
Return Value and Function Type	-/ private void
Description	Calls when 'Embed Text' button clicked.

Method Name	private void ButtonLoadImage_Click(object sender, EventArgs e)
Return Value and Function Type	-/ private void
Description	Calls when 'Load Image' button clicked.

5.2.2. Steganography Methods

Method Name	ExtractMessage()
Return Value and Function Type	-/ private void
Description	Main algorithm of extracting hidden message.

Method Name	EmbeddMessage(string msg, Bitmap bmap)
Return Value and Function Type	-/ private void
Description	Main algorithm of embedding hidden message.

Method Name	LRWriter(int totalPixel, Bitmap bmap, string bin)
Return Value and Function Type	-/ private void
Description	Algorithm type of embedding message. This provides writing message left to write into pixels.

5.2.3. Helper Methods

Method Name	BinaryToString(string data)
Return Value and Function Type	String / public static string
Description	Converts string data to binary as text.

Method Name	PickStartPos(Bitmap bmap, int totalSpace)
Return Value and Function Type	Int[] / private int[]
Description	Generating random starting position from image for embedding message.

5.2.4. Caesar Cipher Methods

Method Name	Cipher(char ch, int key)
Return Value and Function Type	Char / public char
Description	It calculates and changes the original character to ciphered character by given key from user.

Method Name	Encipher(string input, int key)
Return Value and Function Type	string / public string
Description	Enciphers the given message from user.

Method Name	Decipher(string input, int key)
Return Value and Function Type	string / public string
Description	Deciphers the extracted message.

5.2.4. Key Cryptography Methods

Method Name	Encrypt(string plainText, string passPhrase)
Return Value and Function Type	string / public string
Description	It generates random key by using start position, end position and pass code information.

Method Name	Decrypt(string cipherText, string passPhrase)
Return Value and Function Type	string / public string
Description	It decrypt given key by using given pass code.

Method Name	Generate256BitsOfRandomEntropy()
Return Value and Function Type	Byte[] / private byte[]
Description	Generating true random 256 bits.

5.3. Steganography Algorithm Implementation

Embedding Algorithm pseudocode:

1. Take text(message) input from user.
2. Take image from user and convert this image to a bitmap and calculate boundaries of this bitmap to understand limitations of maximum message can be hidden.
3. Cipher text input using Caesar Cipher Algorithm with selected key from user.
4. Convert text input string to binary string (“Example”) -> (“101010”).
5. Generate random start point with using the bounds which calculated at second step.
6. Visit pixels and put pixel’s lsb to our encrypted binary data one by one until reach the length of binary string.
7. Create new image with embedded message bitmap. Use png extension to prevent data loss.
8. Generate text file which includes key, pass code and Caesar cipher key.
9. End.

Extracting Algorithm pseudocode:

1. Take text decryption key from user.
2. Take pass code from user.
3. Take Caesar cipher key from user to use it last step.
4. Take image which includes hidden message from user.
5. Check if these values is true. If not show dumb message.
6. If values are correct start decryption.
7. Extract starting and end point from decrypted key.
8. Convert given image to bitmap.
9. Go to start point pixel of bitmap with using decrypted points.
10. Read least significant bits of pixels until reaching end point.
11. Store all bits inside string.
12. Convert the binary string by reading 8 bits by 8 bits.
13. Decipher string with using Caesar cipher key from we take at step 3.
14. Show last product to user.

5.4. Caesar Cipher Algorithm Implementation

Caesar Algorithm Encipher pseudocode:

1. Take command line arguments for string to be encoded and an integer as a cipher key.
2. Loop through each character in input string, change value by the value of the cipher.
3. Return out encrypted string.

Caesar Algorithm Decipher pseudocode:

1. Take command line arguments for string to be decoded and an integer as a cipher key.
2. Loop through each character in input string, change value by the value of the cipher.
3. Return out decrypted string.

5.5. Key Cryptography Algorithm Implementation

Text Cryptography Algorithm Encrypt pseudocode:

1. Generate random 256bit, store this value in salt.
2. Generate random 256bit, store this value in iv.
3. Encode user input with UTF8.
4. Generate password with using Rfc2898DeriveBytes.
5. Return crypted string.

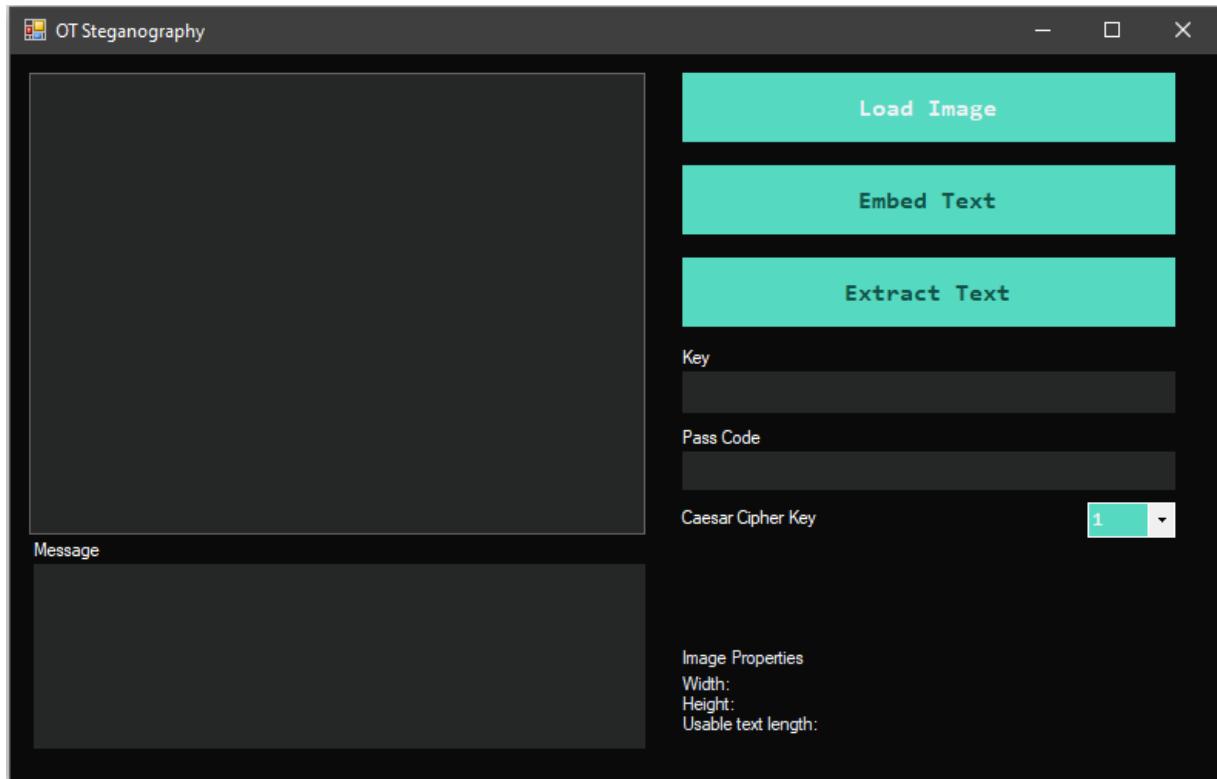
Text Cryptography Algorithm Decrypt pseudocode:

1. Convert cipher text from base 64 string.
2. Split converted cipher text as salt, iv and cipher text bytes.
3. Generate string with using Rfc2898DeriveBytes.
4. Return decrypted string.

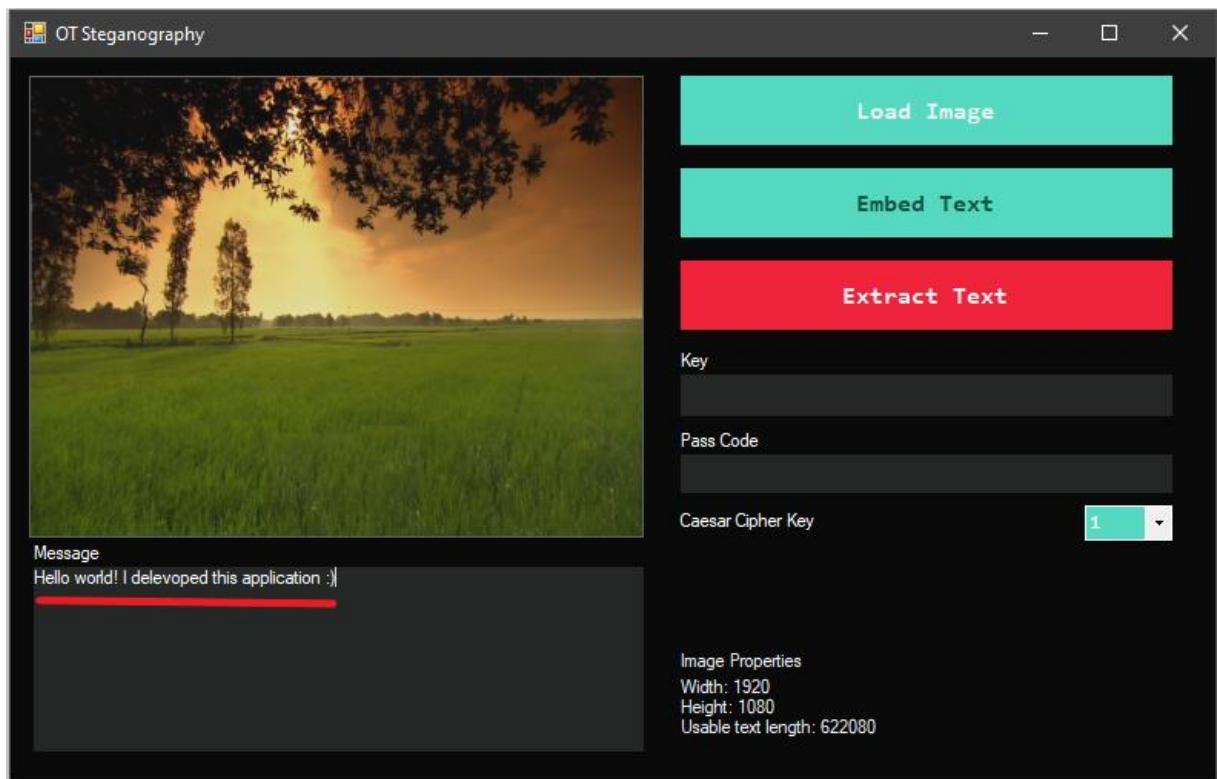
6. Results and Analysis

The operation of the program is explained in detail with pictures.

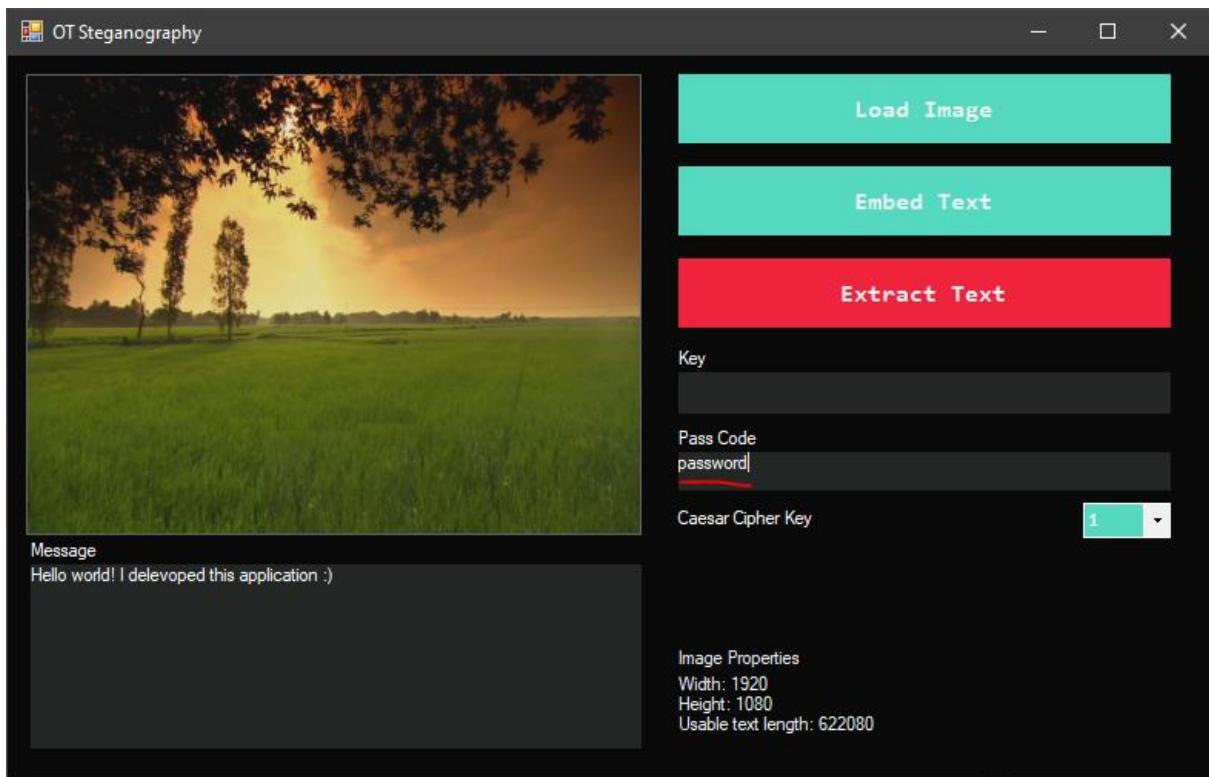
1. First click load image button to select sketch picture.



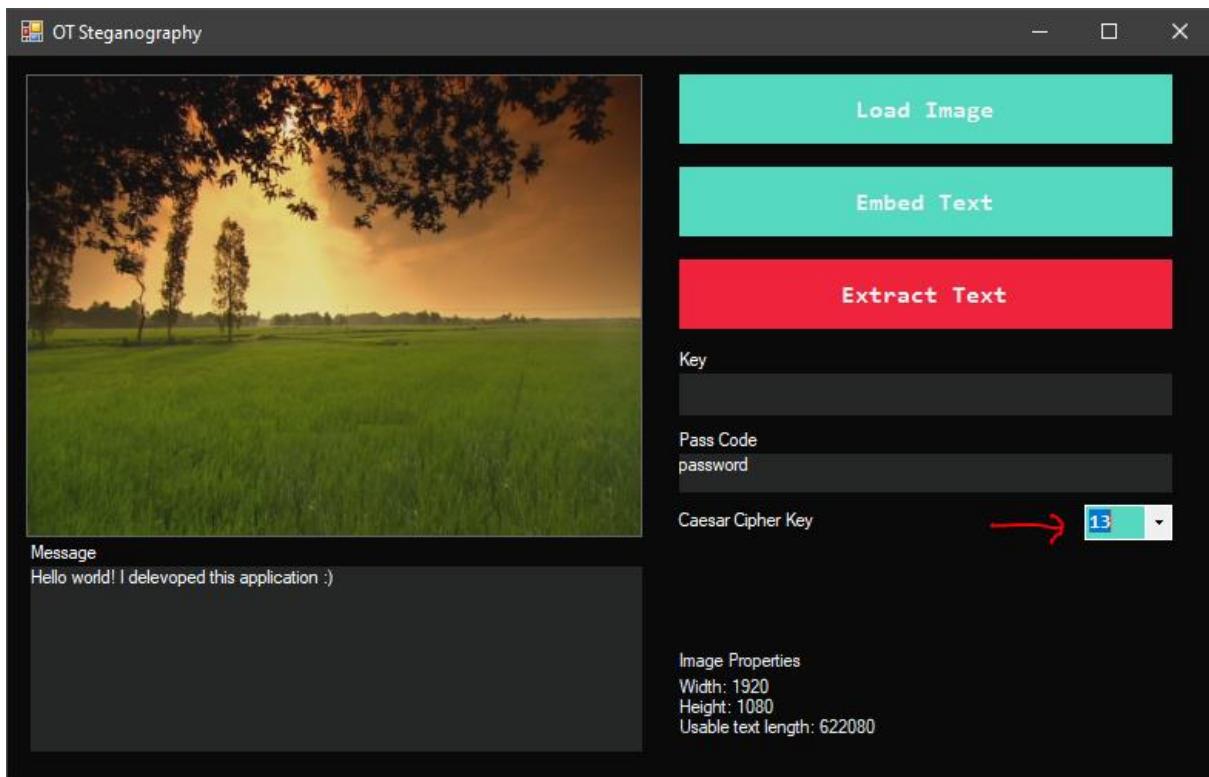
2. Write the message you want to be hidden in the Message field.



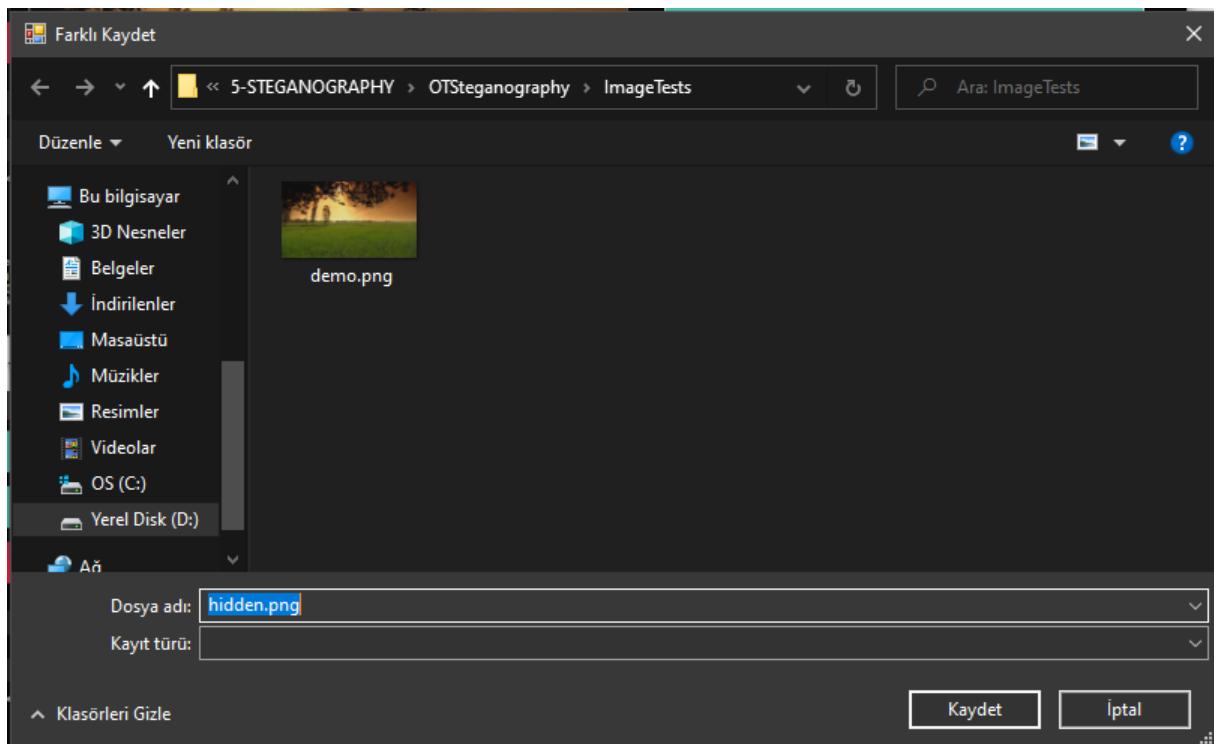
3. Enter a password to create the key personally.



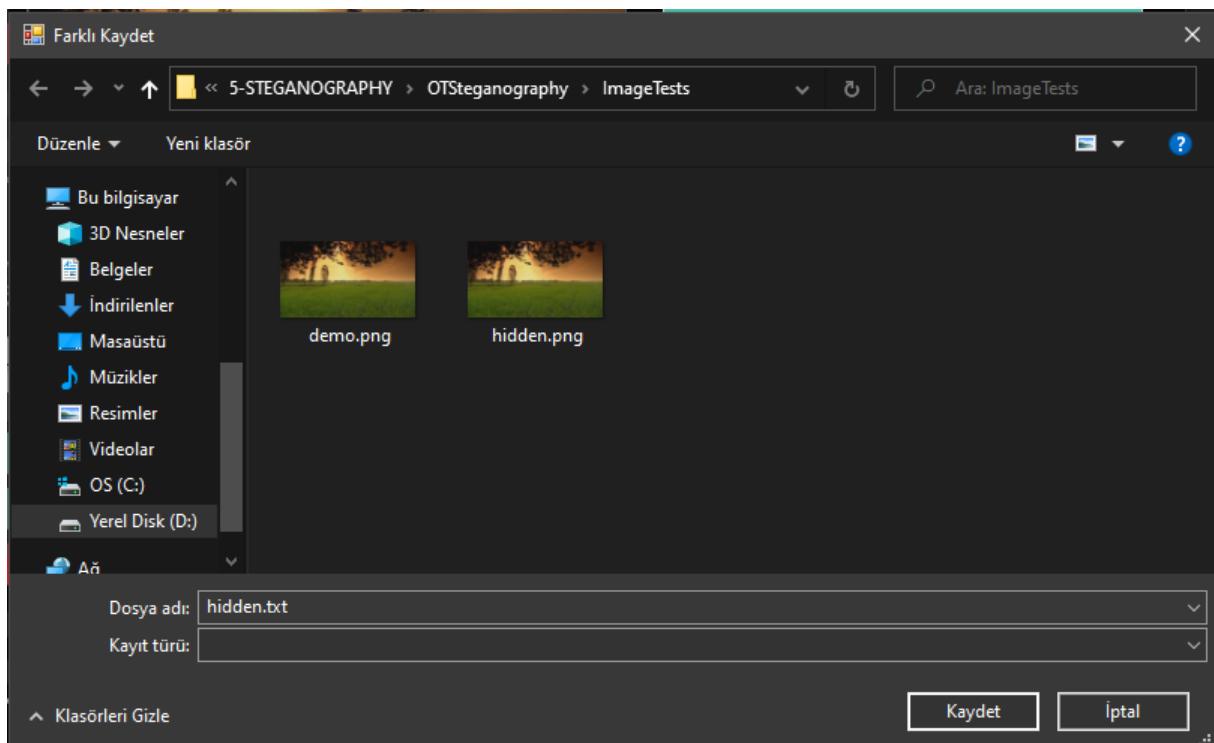
4. Select Caesar Cipher Key to cipher your text message before embedding into picture.



5. Click Embed Text button and give a name to new image will be created.



6. Give a name to a text product. Text product includes these; Key, Pass Code and Caesar Cipher Key.



7. This is text product created by program.

hidden.txt - Not Defteri

Dosya Düzen Biçim Görünüm Yardım

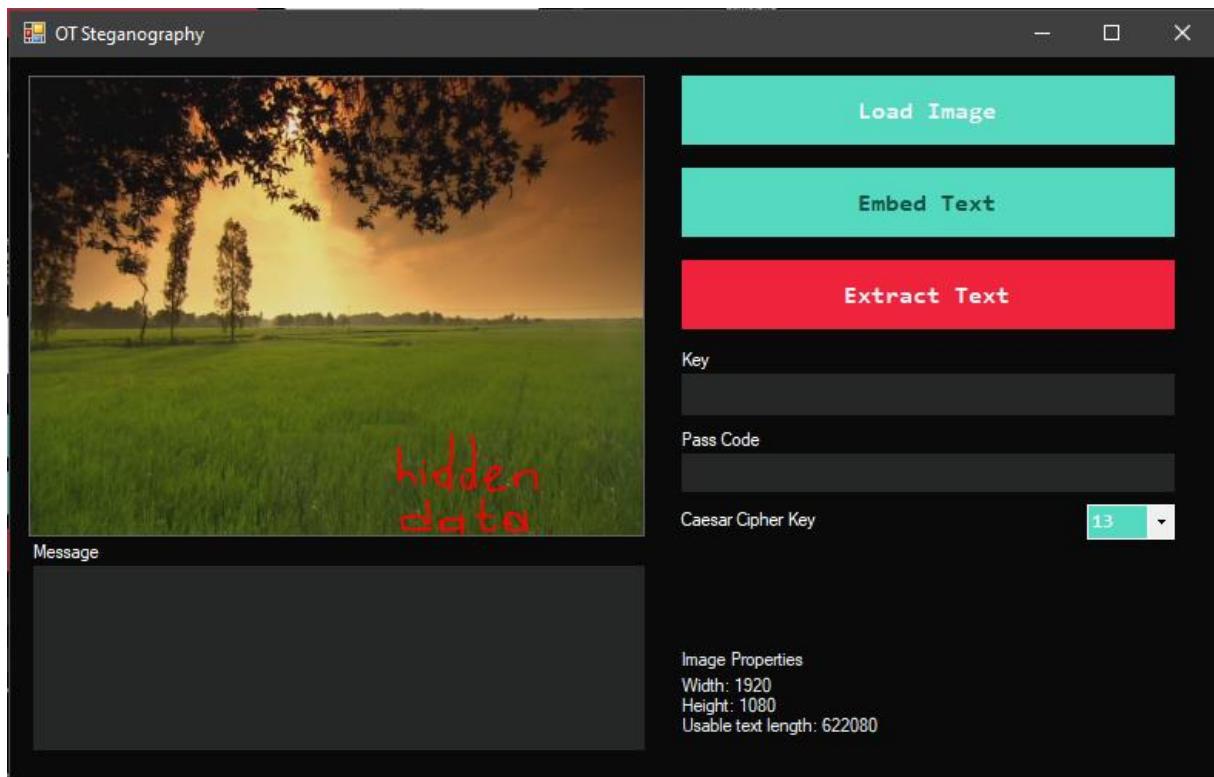
Key: wGs87L9zK1I1ARDfR1NQWHUeeYmJZxfVuWX0pAqpnXJLnqMMyLnM4m/sYKOnYDIno9Fq9Ka0a3nGHD0mlWGSwd/ZG408RkIgu3bhC7

Pass Code: password

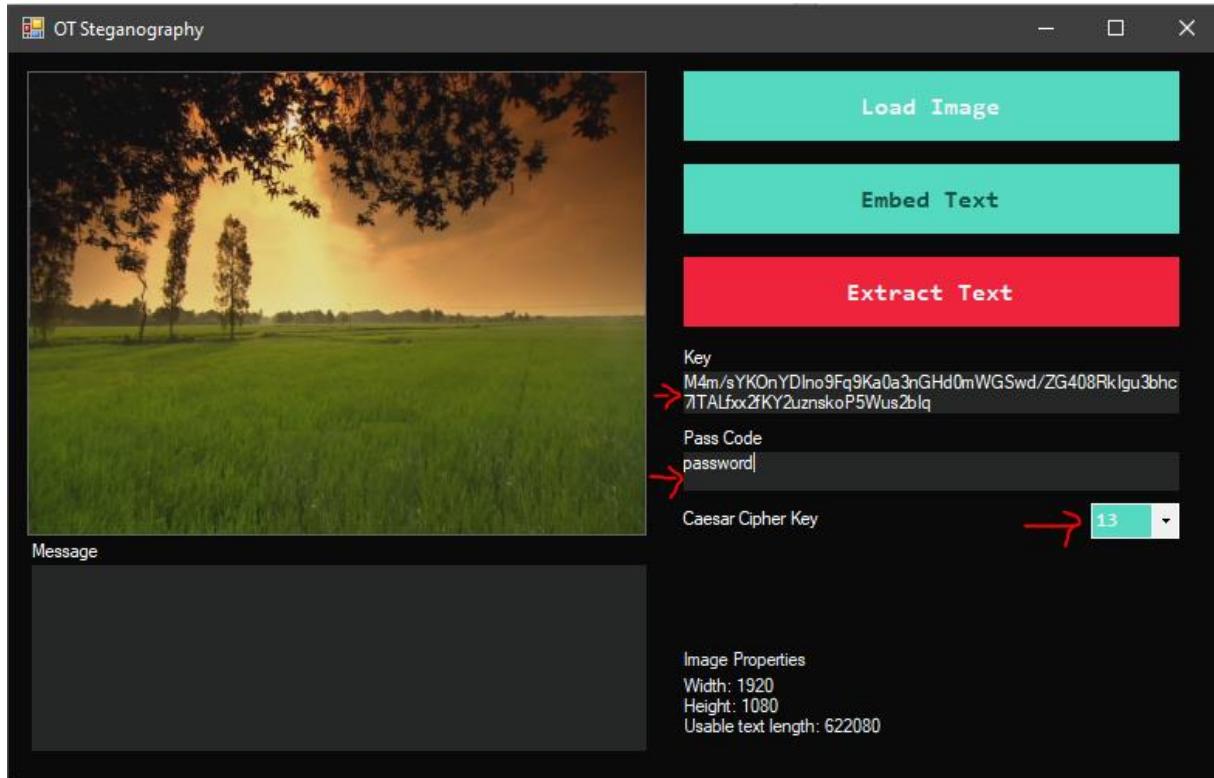
Caesar Cipher Key: 13

St 1, Stn 1 100% Unix (LF) UTF-8

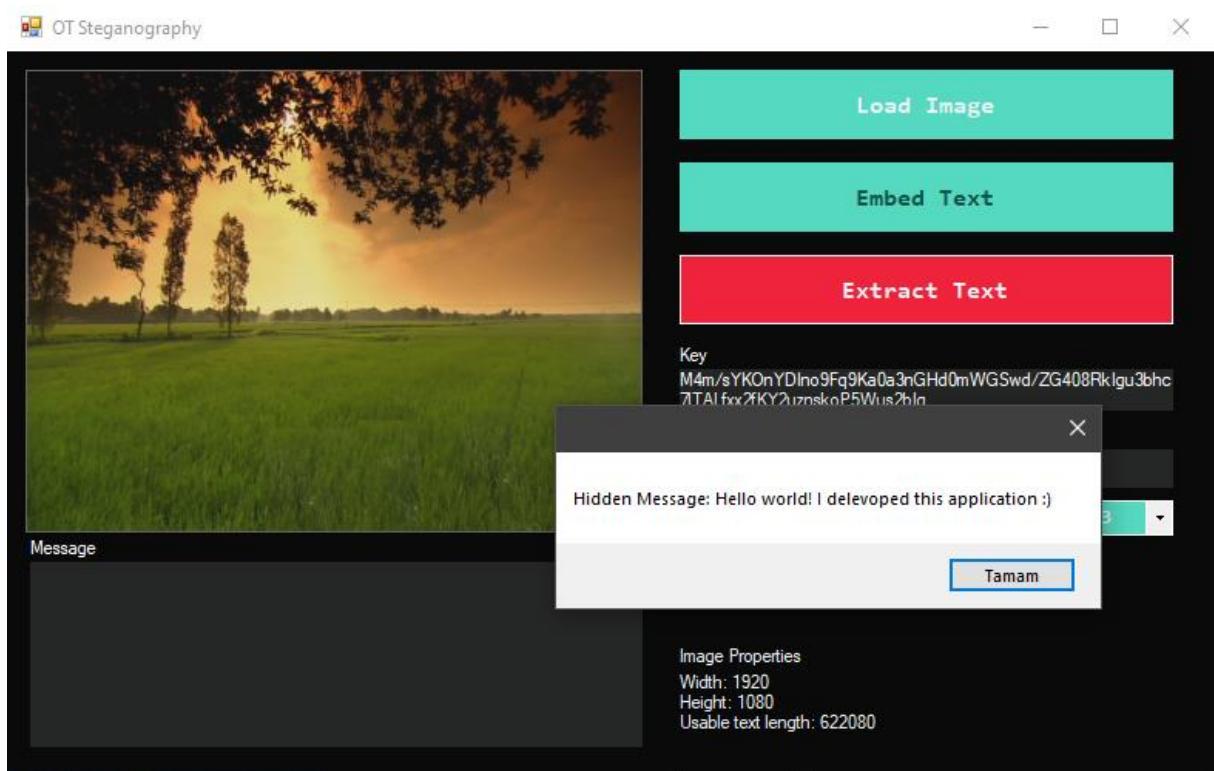
8. Click Load Image button to load image which has encrypted data.



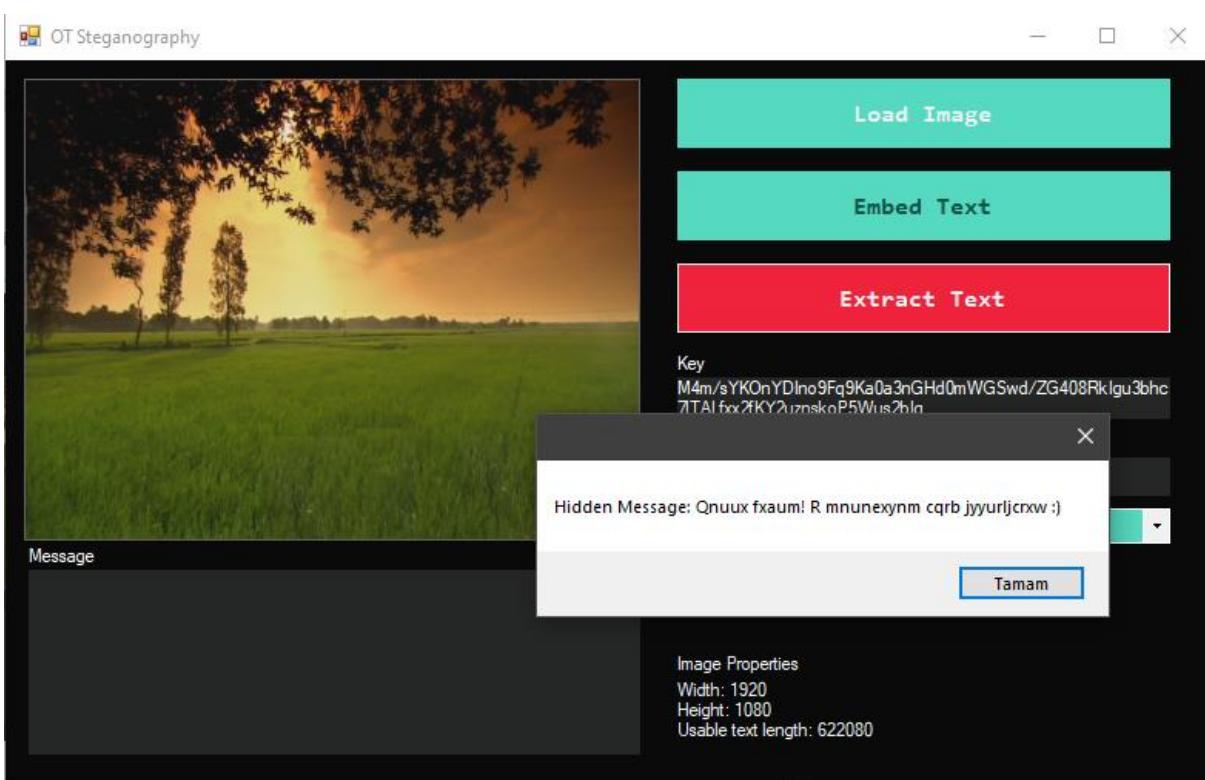
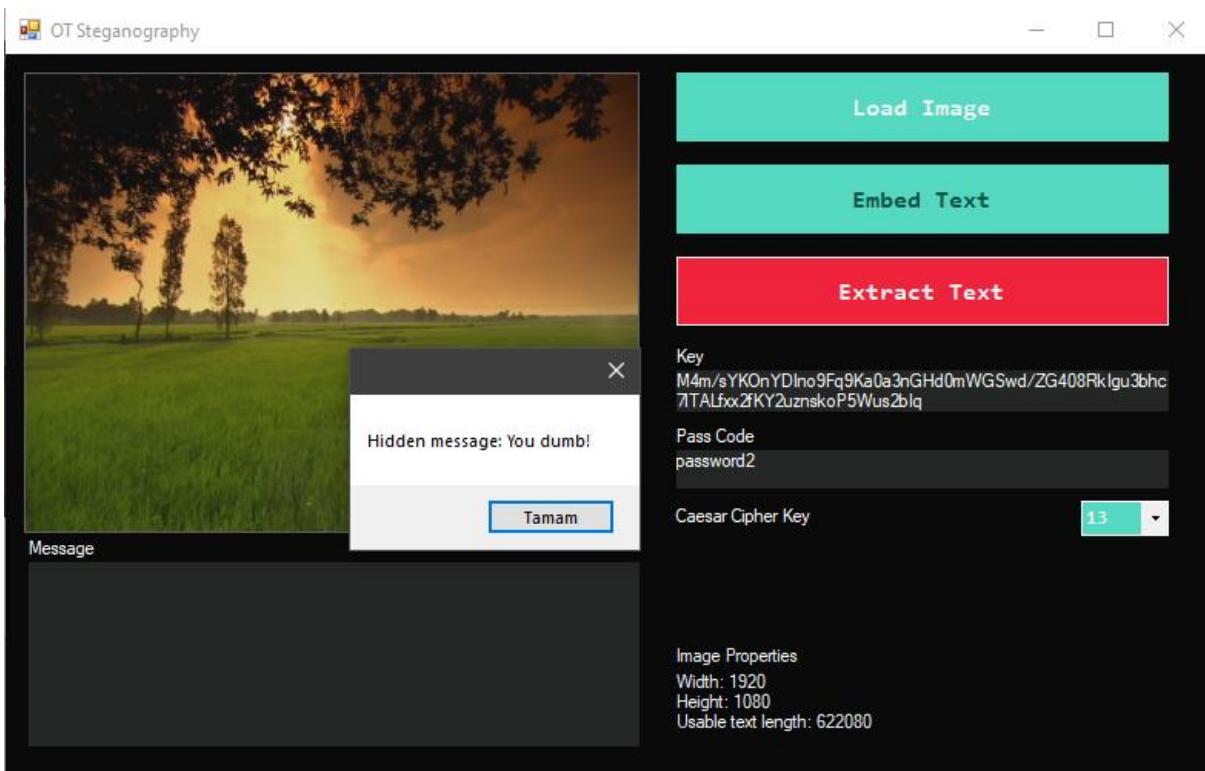
9. Enter your key, pass code and Caesar cipher key correctly on UI before clicking extract text.



10. Click Extract Text button.



11. What if any of these field entered wrong.

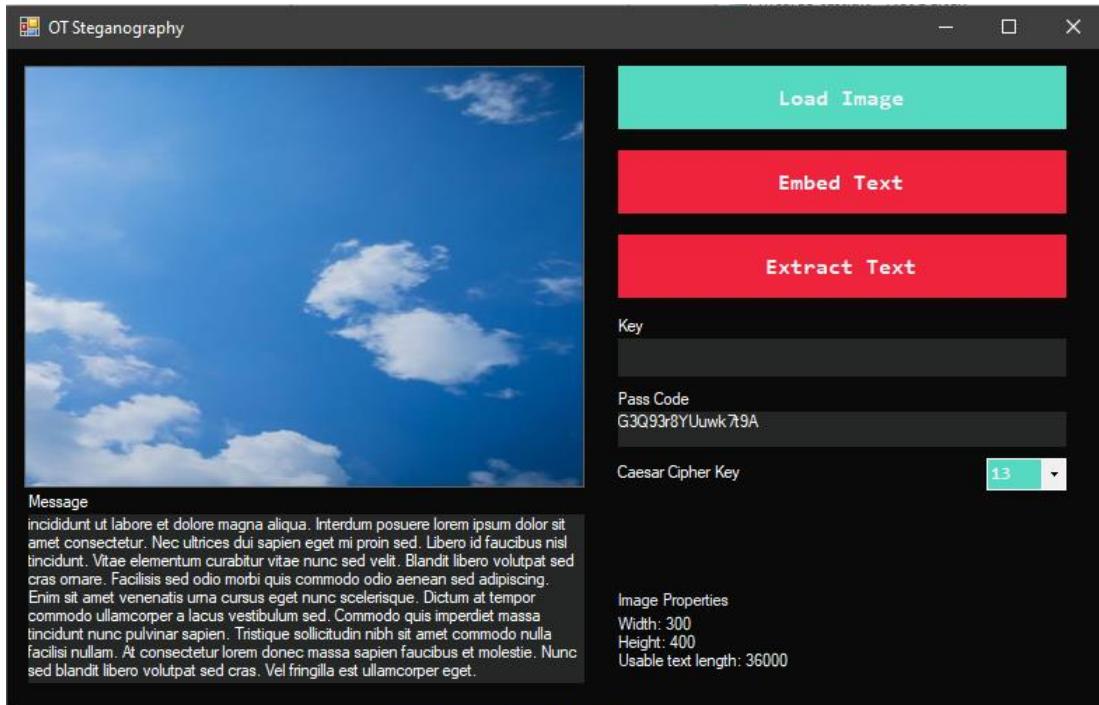


6.1. Analysis

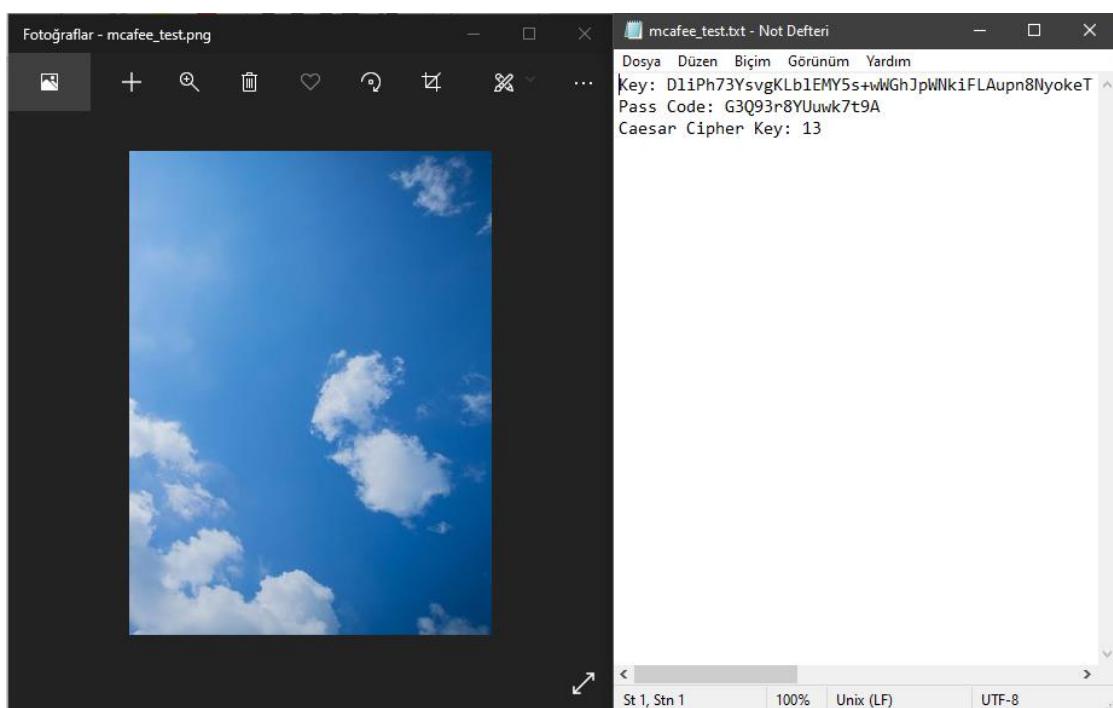
There are several ways to test how undetectable this program. The first of these is "STEGANOGRAPHY DEFENSE INITIATIVE", which the mcafee antivirus program offers online.

6.1.1. McAfee Online Tool for Steganography Detection

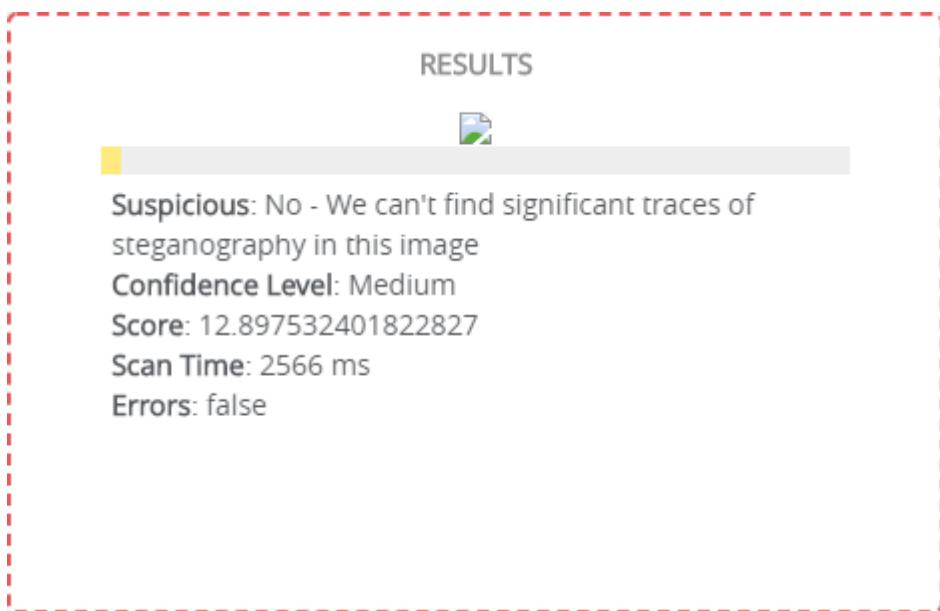
Test inputs:



Final products:



Mcafee result:



6.1.2. Noise Analysis

Next analysis is examine noise on original result and last product. Current state of this application does not support adding random noise on product image. As you can see on final product noise analysis image, where the data is hidden is clearly visible.

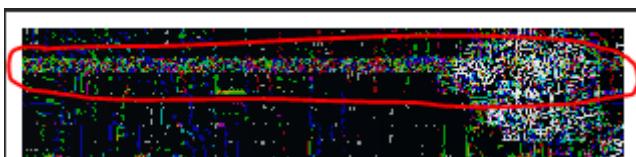
Original Image:



Final Product:



Hidden data:

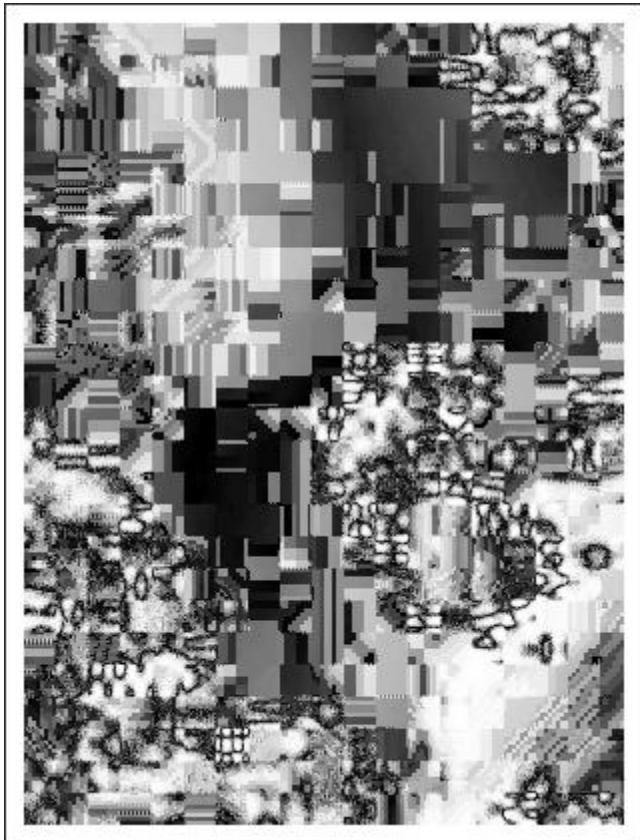


For future upgrading this project adding random noise on final product is must.

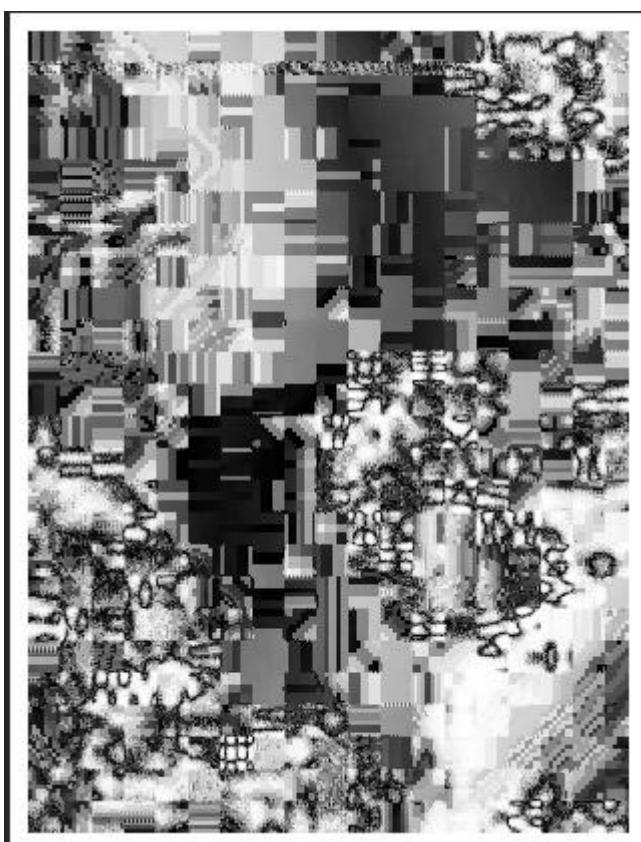
6.1.3. Principal Component Analysis

Next analysis is examine principal component on original result and last product. As the result of noise on product image, this analysis gives the same result as noise analysis. Although not as obvious as in the previous analysis, hidden data can still be seen when viewed with a careful eye. But with the future development of the program – adding random noise support- with this analysis, it can be made impossible to see the hidden data.

Original Image:



Final Product:



6.1.4. Own Analysis Tool (Python)

I developed small analysis tool to maybe I detect hidden data as well. But on the first step I need original image to compare/find hidden data. This analysis does not work at real world for sure.

First, I calculate mean squared error and structural similarity on these images.

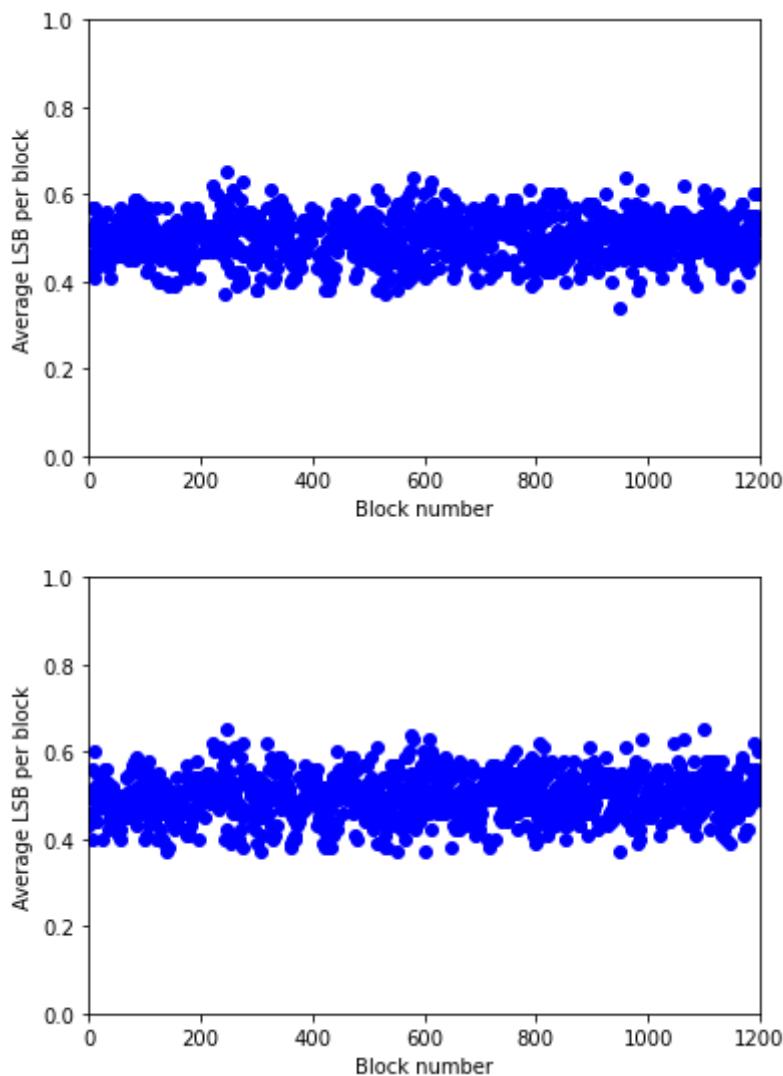
Original vs. Original MSE: 0.00, SSIM: 1.00



Original vs. Final Product MSE: 0.02, SSIM: 1.00



We can see 0.02 which is very small mean squared error. After this I checked avarage least significant bits per blocks. Block size is 100 pixel.



After this process still no significant difference can seen. It may caused by message size. I used small one paragraph data to hide.

6.2. Results

Steganography is still a reliable way to hide your data today. Especially using different encryption algorithms with steganography would be a logical choice for more difficult decryption of data. It will be almost impossible to break the data in the final product produced by a fully developed project.

We know that it is possible to locate the data hidden in this project with various analyzes. Since the data is encrypted even if it is removed from that area, an additional cracking process will be required to crack hidden data. However, as the project continues to be developed, the process of accessing confidential data will also become difficult.

7. Summary and Conclusions

Information hiding techniques received very much less attention from the research community and from industry than cryptography. Steganography has its place in security. It is not intended to replace cryptography but supplement it. It can be clearly observed in this report that these two terms, when used together, reinforce each other. Steganography software is used to perform a variety of functions in order to hide data, including encoding the data in order to prepare it to be hidden inside another file, keeping track of which bits of the cover text file contain hidden data, encrypting the data to be hidden and extracting hidden data by its intended recipient. Steganography is still a reliable way to hide your data today. Especially using different encryption algorithms with steganography would be a logical choice for more difficult decryption of data.

8. References & Resources

1. Mohammad Shirali-Shahreza , “A new method for real time steganography”, 2006
2. Jonathan Cummins, Patrick Diskin, Samuel Lau and Robert Parlett, “Steganography and digital watermarking” 2011
3. Peter Thorsteinson, G. Gnana Arun Ganesh, “NET Security and Cryptography”
4. Domenico Bloisi and Luca Iocchi, ”IMAGE BASED STEGANOGRAPHY AND CRYPTOGRAPHY”
5. Hassanain Raheem Kareem, “Hiding encrypted text in image steganography”
6. Sagar S.Pawar , Prof. Vinit Kakde, “REVIEW ON STEGANOGRAPHY FOR HIDING DATA”
7. Steganography Analysis Tool by McAfee, “<https://www.mcafee.com/enterprise/en-us/downloads/free-tools/steganography.html>”
8. Steganography Analysis Tool by 29a,”<https://29a.ch/photo-forensics/#forensic-magnifier>”
9. Steganography Analysis Tool by livz,” <https://github.com/livz/cloacked-pixel>”