**Abstract**

Steganography is the science of hiding information obscurely in other files so that no one except the sender and the receiver are aware of the message. Cryptography, on the other hand, is the art of modifying the data in a way that only the person having a proper key can recover the original message. Although both of these sciences are used separately to provide security solutions, they can be implemented jointly to provide robust security to information which needs to be shared over a network. In this paper we present a method for encryption of information, combining AES (Advanced Encryption Standard) encryption and RSA (Rivest-Shamir-Adleman) encryption techniques, and its subsequent embedding in an image (24-bit bitmap/png) file using LSB (Least Significant Bit) replacement technique. Before embedding in the image, the information is first encrypted using 128-bit AES encryption. The AES encryption key and the related initialization vector are further encrypted using RSA encryption, and are embedded after the encrypted payload. To add further security, while embedding these details, no two consecutive pixels are chosen to embed consecutive bits, without reducing the payload capacity of the cover image.

**Introduction**

Since the rise of the Internet one of the most important factors of information technology and communication has been information security. Two major techniques used for securing information are steganography and cryptography. Steganography is the science of hiding information obscurely in other files so that no one except the sender and the receiver are aware of the message. Cryptography, on the other hand, is the art of modifying the data in a way that only the person having a proper key can recover the original message. Although both of these sciences are used separately to provide security solutions, they can be implemented jointly to provide robust security to information which needs to be shared over a network.

In this project we have implemented a method for encryption of information, combining AES (Advanced Encryption Standard) encryption and RSA (Rivest-Shamir-Adleman) encryption techniques, and its subsequent embedding in an image (24-bit bitmap/png) file using LSB (Least Significant Bit) replacement technique. Before embedding in the image, the information is first encrypted using 128-bit AES encryption. The AES encryption key and the related initialization vector are further encrypted using RSA encryption, and are embedded after the encrypted payload. To add further security, while embedding these details, no two consecutive pixels are chosen to embed consecutive bits, in a way without reducing the payload capacity of the cover image.

**System Requirements and Specifications**

**System Requirements:**

|  |  |
| --- | --- |
| ***Software Requirements :*** | Windows XP / Server 2003 / Vista / 7 /Server 2008. .NET Framework 3.5 (or higher versions, 4.0) |
| ***Hardware Requirements :*** | *Minimum:* 400 MHz CPU, 96 MB RAM, 800x600 256-color display *Recommended:*  1.0 GHz or higher CPU, 256 MB or more RAM, 1024x768 high-color 32-bit display, up to 500 MB of hard disk space may be required |

**Code Specifications:**

|  |  |
| --- | --- |
| ***Language :*** | C#.NET (using .NET Framework 3.5) |
| ***IDE :*** | Microsoft Visual Studio 2010 |

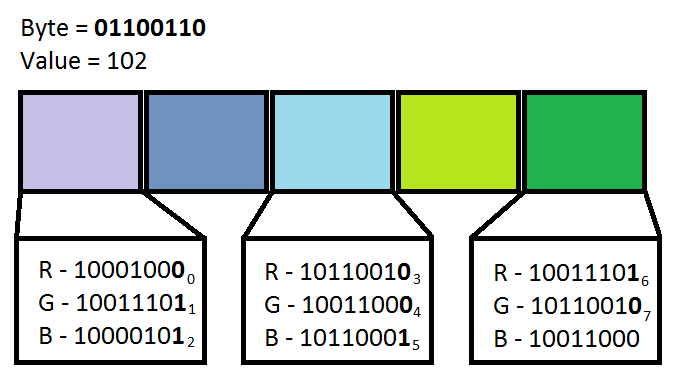
**Project Description**

## LSB (Least Significant Bit) Replacement Technique

LSB Replacement technique is a simple and straightforward approach to hide information bits in a cover image. In this method the Red(R), Green(G) and Blue(B) colours of some pixels are slightly altered to reflect the embedded bits, such that any difference between the original cover image and the generated stego image remains unnoticeable to human eyes.

Every pixel of a 24-bit bitmap layout of an image consists of Red(R), Green(G) and Blue(B) colour values, represented as three different bytes. In LSB Replacement technique, the least significant bits of these bytes are replaced by bits of the information message. Thus, each pixel can hold three bits of information. So, for embedding a complete byte (8 bits) of information, a total of three pixels are needed. As shown in Fig. 1 and Fig. 2, 8 bits of a byte can be sequentially embedded in first pixel’s R-G-B bytes, second pixel’s R-G-B bytes and third pixel’s R-G bytes (1R, 1G, 1B, 2R, 2G, 2B, 3R, 3G). Such an alteration remains undetected to the human eye. The pixels selected for embedding are determined by a stego key secretly shared between communicating parties.

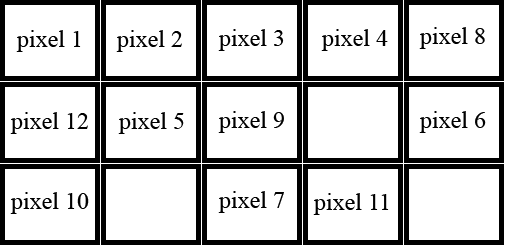
Being implemented on lossless image formats, LSB Replacement technique provides high levels of information invisibility and payload capacity. The use of .bmp files as cover images are generally discouraged because they are not widely used anymore and their large size can arouse suspicion. However, other lossless formats, like png, can be conveniently used as cover images to embed information and transfer over the internet obscurely.



*LSB* (*Least Significant Bit) Replacement Technique*

## Selection of Pixels Using Stego Key

As a security-through-obscurity measure, for embedding a consecutive set of bits, no two consecutive pixels are chosen. A gap of ‘n’ pixels is always maintained between two selected pixels. This value, ‘n’, referred to as message-key or stego-key, is essential to information extraction, and is embedded in consecutive pixels at the beginning of the image. Whatever information be embedded after it, always the (n+1)th pixel from the initial pixel is chosen for embedding.



S*equence of selected pixels for embedding 1 byte of key, 1 byte of message length and 2 bytes of message with value of key equal to 2 (1 byte embeds in 3 pixels).*

If a gap of n pixels is to be kept between two pixels used for embedding, the payload capacity of the cover image shall decrease drastically, depending upon the value of the key. To overcome this issue, once the image runs out of pixels for embedding information, embedding again starts from the first unused pixel from the top using the same stego-key. In this way, every pixel of the image can be used for embedding and maximum payload can be obtained. The total payload of the image can be given by (image height \* image width / 3).

**Encryption of Information and Keys**

Prior to embedding the information in the cover image, encryption of all the contents highly increases the security of the information. This can be implemented using the following procedures.

### Encryption of Message Key using RSA Encryption Algorithm:

### The Message Key, i.e., the ‘n’ pixels of gap we choose to leave between selected pixels while embedding the information length and information bits, is encrypted using RSA encryption algorithm. The length of the RSA public key is 1024 bits (128 bytes). So the generated cipher text is of 128 bytes in length.

### Encryption of Information Length using RSA Encryption Algorithm:

### The length of the message, in bytes, is again encrypted using an RSA cipher. The length of the RSA public key is 1024 bits (128 bytes) again, and so a cipher text of 128 bytes is again generated.

### Encryption of Information Bytes using AES Encryption Algorithm:

### The information bytes are now encrypted using AES encryption cipher with a symmetric key of 128 bytes in length. As such, the cipher text generated in this case is of the same length as that of the original message.

### Encryption of AES Encryption Key using RSA Encryption Algorithm:

### The AES encryption key used to encrypt the information bytes is encrypted using RSA encryption algorithm. A cipher text of 128 bytes is again generated in this case.

### Encryption of AES Initialization Vector using RSA Encryption Algorithm:

### Finally, the initialization vector used in the AES encryption of the information bytes is encrypted using RSA encryption algorithm. Again a cipher text of 128 bytes is generated here.

The RSA encryption keys exist as pairs of public and private keys. The public keys are used for encryption and the private keys are used for decryption. The public keys have to be distributed beforehand to enable users to encrypt the desired information before embedding, and the private keys have to be kept at the receiving end for decrypting the received encrypted information.

## Embedding the Cipher Message

For proper retrieval of desired information from the received stego image, the encrypted bytes have to be embedded systematically in the cover image. The following sequence of embedding the cipher texts is appropriate for this purpose.

* RSA encrypted stego key.
* RSA encrypted information length.
* AES-128 encrypted information bytes.
* RSA encrypted AES encryption key.
* RSA encrypted AES initialization vector.

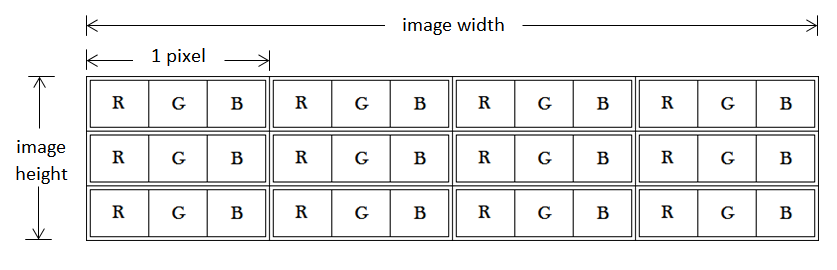
## Extracting the Embedded Information

Once the stego image has been received, the information hidden in it can be extracted in the following sequential manner.

* Extraction of the encrypted stego key and its subsequent decryption using the necessary RSA private key.
* Extraction of the encrypted message length and its subsequent decryption using the necessary RSA private key.
* Extraction of AES encrypted information bytes.
* Extraction of AES encryption key and AES initialization vector and their subsequent decryption using the necessary RSA private key.
* Decryption of the encrypted information bytes using the decrypted AES encryption key and initialization vector.

**Data Preparation**

The preparation of a stego image using the process described above requires a cover image file, a message which can be a string of characters or a file, and a set of encryption keys. The image formats have been selected as bmp and png file format, both being lossless image formats, as LSB Replacement method works only with lossless image formats.

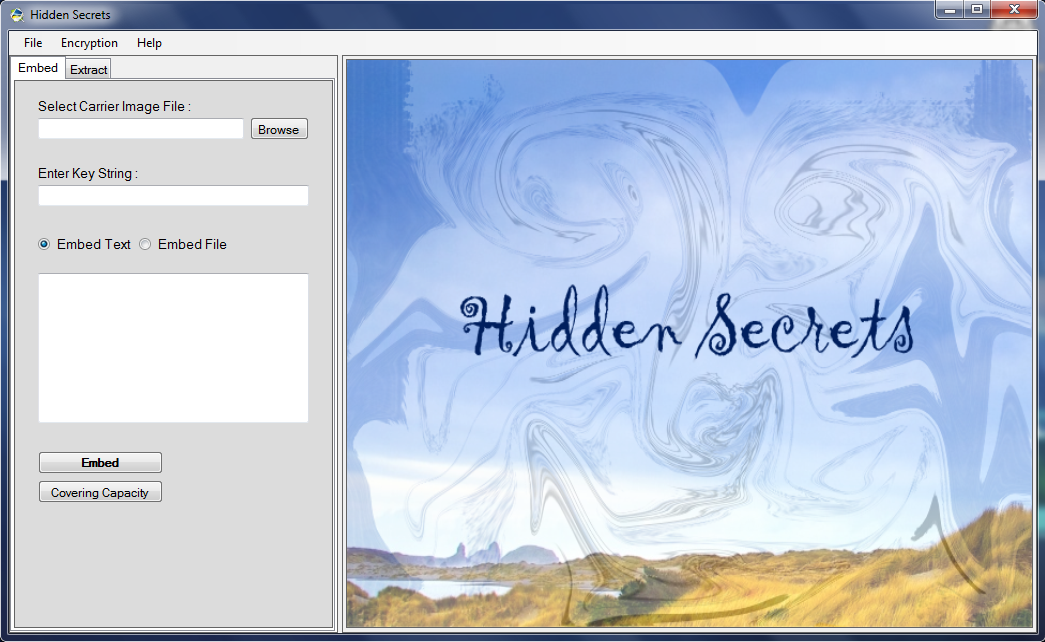


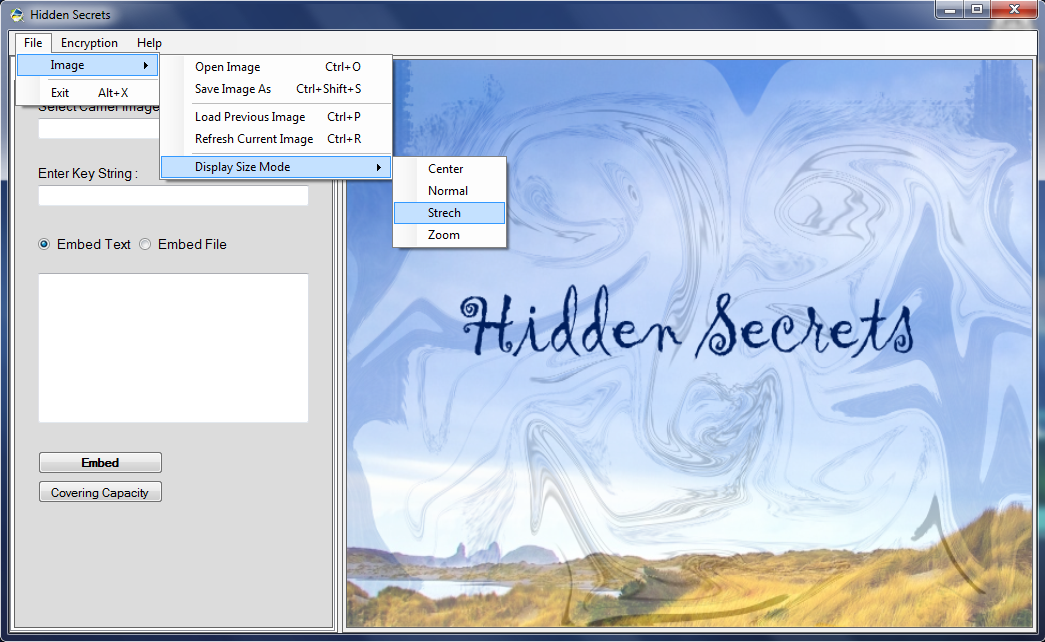
*Representation of pixels in a 4X3 24-bitmap image*

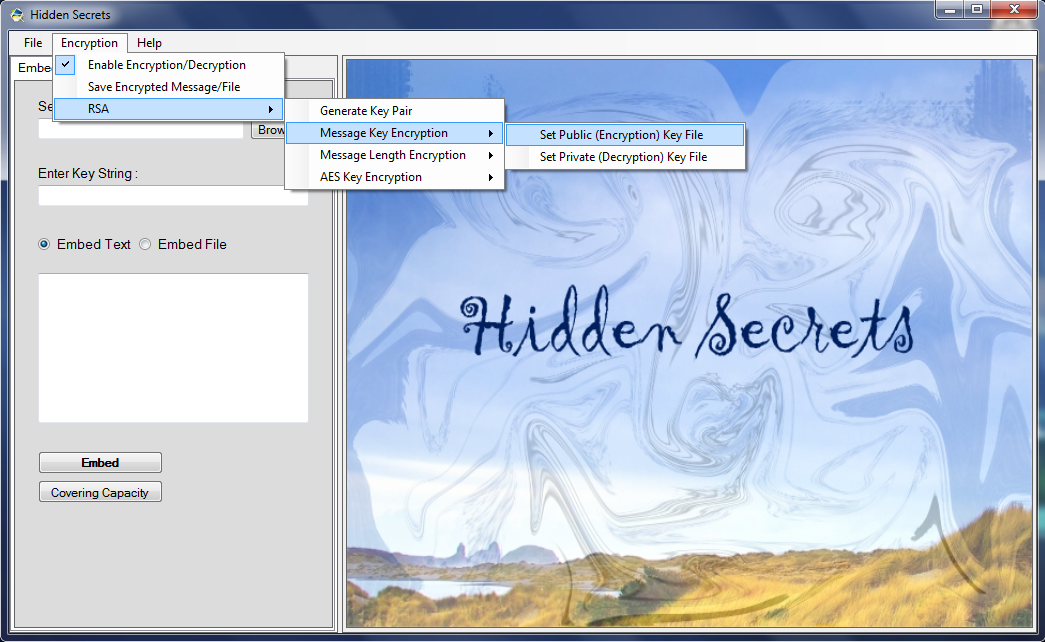
The message that a user wishes to embed can be any sequence of bytes. It can be a string of characters, which has to be converted into an array of bytes, and then embedded into the cover image. It can also be a file, from which bytes can be extracted into an array, and then embedded into the cover image.

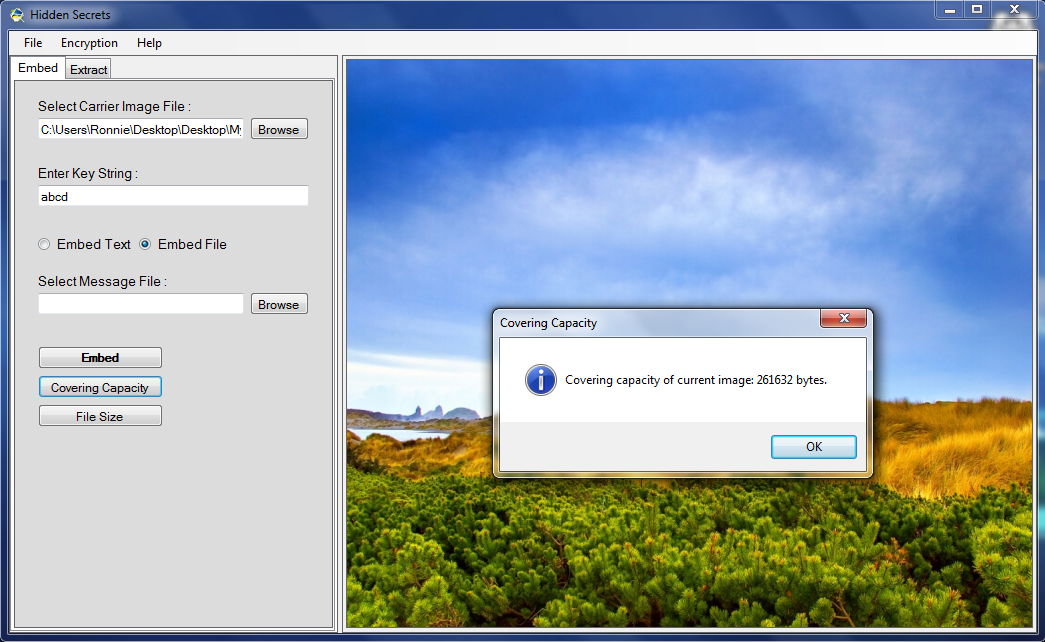
Finally, a set of encryption keys will are needed for encrypting the message before embedding. The AES encryption keys will be randomly generated by the application every time the user attempts to embed a message into the cover image. The RSA Encryption-Decryption keys are supplied from files previously stored in the computer where the application is running. The user can select these files according to his choice or need. Three pairs of default RSA keys are provided to the user.

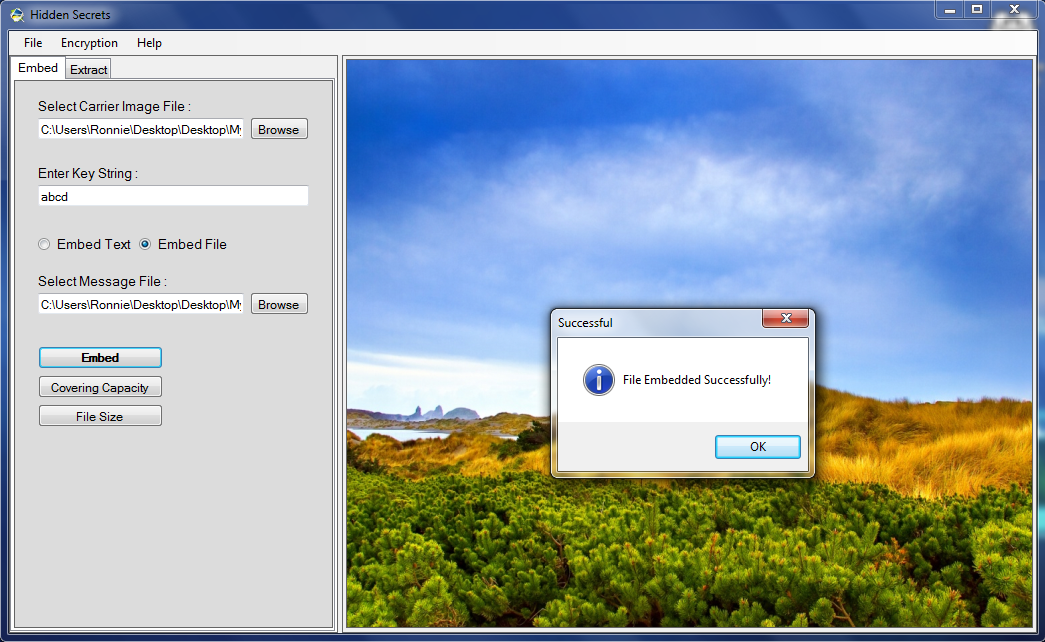
**Screenshots**

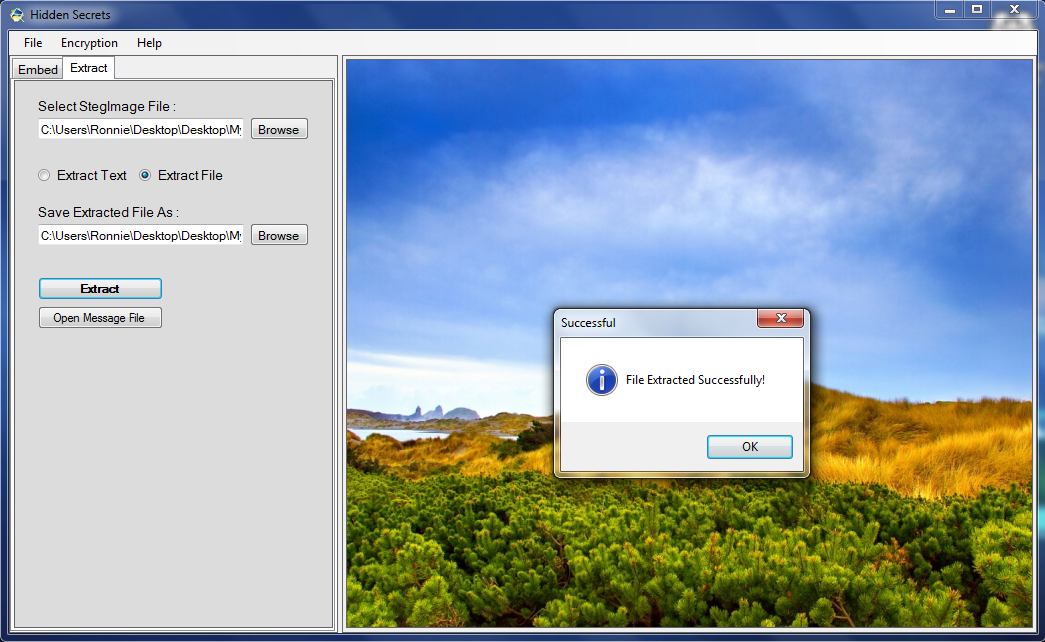


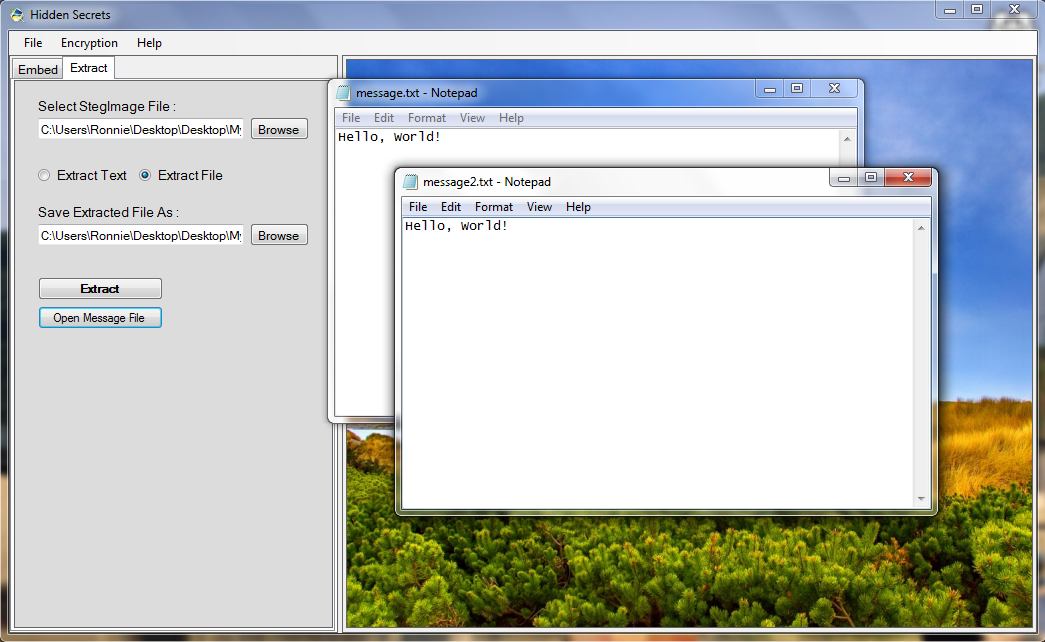












**Limitations**

1. A limitation is imposed on the message size by the size of the carrier media file.
2. If the size of the carrier image file is low, then the size of the stego-image file may become very large as compared to the original carrier file. This may arouse suspicion, thus not maintaining complete obscurity.
3. The application is being developed in C#.NET language using .NET Framework 3.5. This Framework has been completely implemented only in Windows systems till now. So, the final application cannot be completely platform independent.

**Further Scope**

1. Audio/Video file formats can be additionally implemented as carrier files. This will also help in increasing the limitation on the message length.
2. Inclusion of more encryption algorithms for customized encryption.
3. This project can be implemented in GNU Linux platform using Mono Project. This will drastically increase the platform independency of this project.
4. This project can also be implemented in a web based (ASP.NET) application. This is the best possible way to make the project accessible to millions of people worldwide.

**Conclusion**

This project can have a vast field of applications. Though it has a limitation of having minor anomalies in image sizes after embedding, it is extremely robust in terms of security. Before embedding, the data is made highly secured by being processed through three layers of RSA encryption and one of AES encryption, which may a large amount of time to crack, given the current technologies. Furthermore, the kind of embedding technique that has been used here makes it extremely tough to extract the encrypted data.

**Bibliography**

1. Mona M. El-Ghoneimy, “Comparison Between Two Watermarking Algorithms using DCT Coefficient, and LSB Replacement”, Journal of Theoretical and Applied Information Technology.
2. Rita Rana and Er. Dheerendra Singh, “Steganography-Concealing Messages in Images using LSB Replacement with Pre-Determined Random Pixel and Segmentation of Image”, International Journal of Computer Science and Communication, vol. 1, no. 2, July-December 2010, pp. 113-116.
3. T. Morkel, J.H.P. Eloff and M.S. Olivier, “AN OVERVIEW OF IMAGE STEGANOGRAPHY”, Information and Computer Security Architecture (ICSA) Research Group, Department of Computer Science, University of Pretoria.
4. B.H. Forouzan, “Cryptography and Network Security”, Tata McGraw Hill Publications.
5. Moreland, T., “Steganography and Steganalysis”, *Leiden Institute of Advanced Computing Science*.
6. “ADVANCED ENCRYPTION STANDARD”, Federal Information Processing Standards Publication 197, November 26, 2001.