TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
Central Campus, Pulchowk
Lalitpur, Nepal

A Project Report
on

PHOTOCRYPT

Submitted by:

Manish Munikar, `070-BCT-520`
Dipesh Pandey, `070-BCT-514`
Brihat Ratna Bajracharya, `070-BCT-513`

Submitted to:

Er. Daya Sagar Baral
Department of
Electronics and Computer
Engineering

March 14, 2015

Acknowledgement

This project report is prepared for the project required for the partial fulfillment of the academic requirement of the 3rd semester (2nd year, 1st part) of Bachelors in Computer Engineering. Effort has been made to ensure that this report is accurate and professional as far as possible.

We are very grateful to the Department of Electronics and Computer Engineering (DoECE) of IOE Central Campus, Pulchowk for providing us with this great opportunity to develop a minor project. It has greatly enhanced our knowledge of object-oriented programming paradigm and has given us the valuable experience of research and teamwork, which will definitely be of great help for our career development.

We'd like to express our deepest gratitude to our project coordinator, *Er. Daya Sagar Baral*[1], for guiding us through the course of this project development by providing precious knowledge and suggestions.

We'd like to show our appreciation to all the senior brothers and sisters who helped us solve problems and manage project well by providing us with their valuable experience.

We'd also like to thank all our generous friends, especially *Raj Dangol*, for reviewing our project and providing us valuable suggestions, recommendations and criticism.

Team members:
Manish Munikar, `070-BCT-520`
Dipesh Pandey, `070-BCT-514`
Brihat Ratna Bajracharya, `070-BCT-513`

[1] He is also the author of our main course book[4] for C++.

Abstract

This is the technological era—the age of the computers. All the information that you ever need to know are stored in computers in terms of binary digits (0s and 1s). Most of the computers in the world are connected by a super-large wireless network called the *internet*. The whole world has transitioned from almost completely mechanical to almost completely digital technologies in a matter of couple of decades.

Due to this tremendous growth in information and communication technology, it is now a real challenge to maintain privacy and to send some confidential data over a network. People can easily copy your private information without making you slightest aware of it. People can interrupt the message between two parties and even alter the messages.

To help maintain digital privacy and security, various information security systems, such as digital cryptography and steganography, were developed. Cryptography makes data unreadable and steganography makes data invisible. The advantage of steganography over cryptography alone is that the intended secret information does not attract attention to itself as an object of scrutiny. Plainly visible encrypted messages—no matter how unbreakable—will arouse interest, and in themselves be incriminating in countries where encryption is illegal.

Therefore, in this project, we tried to build a software that implements the concept of steganography to secure confidential and secret data.

Table of Contents

List of Figures

1 Introduction

This project, entitled ***Photocrypt***, is a software project that implements the concept of *steganography*. The word, *Photocrypt*, is a hybrid mixture of the words *photograph* (image) and *encrypt* (to hide or conceal). Steganography, like cryptography, is a *not-so-popular* concept of information security and privacy. It is a way to transfer secret message from one party to another without making anyone aware of the transfer (i.e., by hiding the message in such a place where no one suspects).

So, *Photocrypt* is basically a simple software that can hide (encrypt) textual information in image files without changing the size and appearance of the image in a noticeable way. This concept is known as *Image Steganography*. It is made possible by the fact that, in computer technology, a file is nothing but a finite sequence of bits (0s and 1s), whether it be a text file or an image file. But in almost all cases, the size of a text file is much smaller than the size of an image file. Also, some of the bits of the image files can be altered without altering the appearance of the image in a noticeable way. But altering even a single bit of a text file corrupts it. By exploiting these facts, we can hide text files in images files by altering some bits of the image files to represent the bits of the text files in such a way that the text can be recovered back from the image with a deterministic algorithm. In this way, we can hide a text file in an image file and keep the text file secret and secured from unwanted parties.

This project is an aim to fulfill people's ever increasing need for digital information privacy and security.

2 Objectives

We can broadly categorize our objectives into two main categories. They are as follows:

2.1 Academic Objectives

- To fulfill the partial requirement of our course.
- To learn the object-oriented programming (OOP) programming paradigm using C++ programming language.
- To learn to develop software in a managed way.
- To gain research and team work experiences.

2.2 Software Objectives

- To implement the concept of steganography and build a software for digital privacy and security.
- To conceal textual information in images with negligible changes.
- To secure the hidden text in image with password protection.
- To build simple and easy interface (both graphical as well as command-line) for our program.
- To build a simple library for text-to-image steganography.

3 Application

Steganography is a very interesting concept which has a significant place in the future of information privacy and security.[6] Our project, which implements text-to-image steganography, can be applicable in following areas:

3.1 Personal Application

- We have many accounts created in various websites and keeping track of the usernames and passwords in each can get tedious very quickly. In such a case, Photocrypt can be used to secure the login information of all our accounts in a simple image and protect that information with a single master password. In this way, we can secure our passwords without using any password managers. The best thing is that other people won't have a slightest idea how you're securing your secret information.

- We know, *"a picture speaks a thousand words"*. Now we can store the precious memories related to a picture *inside* that picture. The photograph and the history of the photograph are united in a single file. We can then refresh our memories of that picture few years later with the picture itself (i.e., without needing any more information/files).

3.2 Business Application

- In business, Photocrypt can be used to transfer confidential data/information (such as secret letters, or algorithms) between two parties without making anyone else aware of the information transfer. This is especially useful because

there are many things in business sector that should remain a secret (as well as secure) and cryptography alone cannot achieve that.

3.3 General Application

- Photocrypt can also be used to slightly reduce disk usage. Since Photocrypt makes images carry more information than just the image and since a single image can hide a large amount of text, we can get rid of the text files in the disk and save some space.

4 Literature Survey

4.1 Introduction to Steganography

Steganography[7] is described as "the art and science of writing hidden messages in such a way that no one apart from the sender and the intended recipient even knows that a message has been sent". The word, *steganography*, is derived from Ancient Greek words *steganos* (στεγανός), meaning "covered, concealed, or protected", and *graphien* (γράφειν), meaning "writing".[2]

Digital steganography is the art or practice of concealing a file, message, image, or video within another file, message, image, or video in such a way that the carrier object remains almost identical to the original one. It is an art and science of invisible communication.[3] It is slightly different (better) than cryptography in the fact that cryptography is the practice of protecting the contents of an information alone, whereas steganography is concerned with concealing the fact that a secret information is being protected, as well as protecting the information.

4.2 Brief history of steganography

Steganography as a whole has existed in many forms throughout much of history. Some of their forms are given below:

- The ancient Chinese wrote notes on small pieces of silk that they then wadded into little balls and coated in wax, to be swallowed by a messenger and retrieved at the messenger's gastro-intestinal convenience.

- Herodotus (485 – 525 BC) recounts the story of Histaiaeus, who wanted to encourage Aristagoras of Miletus to revolt against the Persian king. In order to securely convey his plan, Histaiaeus shaved the head of his messenger, wrote the message on his scalp, and then waited for the hair to regrow. The messenger, apparently carrying nothing contentious, could travel freely. Arriving at his destination, he shaved his head and pointed it at the recipient.
- Giovanni Battista Porta described how to conceal a message within a hard-boiled egg by writing on the shell with a special ink made with an ounce of alum and a pint of vinegar. The solution penetrates the porous shell, leaving no visible trace, but the message stained on the surface of the hardened egg albumen, so it can be read when the shell is removed.
- Modern digital steganography entered the world in 1985 with the advent of personal computers being applied to classical steganography problems, and it has since taken off, going by the large number of steganography software available:
  - Concealing messages within the lowest bits of noisy image or sound files.
  - Changing the order of elements in a set.
  - Modifying the echo of a sound file (Echo Steganography) and so on.

4.3 Research on Steganography

There are a lots of researches going on from university level in the field of steganography. They are mostly based on new approaches of steganography & steganalysis, and new, better and more secure steganography algorithms.

5 Existing System

While steganography is a pretty old concept, there has not been very popular steganography softwares. Steganography still remains a relatively unexplored area in the field of information technology. It is, therefore, a field for students to work on, and that's exactly what is seen being done, i.e., most of the steganography softwares of today seem to be primarily for educational purposes rather than business purposes.

Some of the softwares similar to Photocrypt are OpenSteg, JSteg, MP3Stego, etc.

6 Methodology

6.1 Project Development

The project was developed in about 3 months time. Our methodology can be briefly summarized as follows:

6.1.1 Selection of idea

While researching about ideas to do a project on, we came across this concept of steganography. Then we researched about it if it was feasible and we decided it to be a very good and optimal idea to do a project on.

6.1.2 Research

After deciding the project idea, we all worked on collecting as much information about it as possible. We searched rigorously through the web for documents, articles and journals on the concept of steganography. During this period, we also learned about basic steganography algorithms and used it to derive our own algorithm.

6.1.3 Proposal submission

The proposal was submitted with our basic idea and objectives.

6.1.4 Design

Before actual coding, we all worked on having a proper design. We discussed about our interfaces and the classes required for object-oriented design. We also researched about the external third-party libraries that we needed to complete this project.

6.1.5 Coding

After the design had been fixed, we all worked on learning the necessary libraries to perform image-processing, basic cryptography and to build graphical user interface (GUI). Then we started the actual coding. We hosted our project source code in

GitHub and used Git to manage the project development. During this phase, we also performed intensive testing and debugging. We used comments in our source code to describe key concepts used while coding, for better maintenance.

6.1.6 Report

After the project was developed enough to fulfill our primary objectives, we finalized it and prepared this project report.

6.2 Tools and concepts used

We have used many tools, algorithms, libraries, or concepts to build this project. Some of the significant ones are as follows:

6.2.1 C++ programming language

C++ is a general-purpose multi-paradigm programming language created by Bjarne Stroustrup. It is an improvement of the C programming language. It has *imperative*, *object-oriented*, and *generic* programming features, while also providing the facilities for low-level memory manipulation.[4][8][5] It can be used to build almost any type of software, whether it be system software, application software, or even embedded software.

We used C++ as our programming language because of course requirements and also because we learned it most recently and it's our favorite programming language right now.

6.2.2 Vim text editor

Vim is a powerful free and open-source text editor. It is an acronym of "Vi IMproved". It is an improvement over the *vi* editor of UNIX. It provides many facilities beneficial for programmers and is amazingly extensible. The best thing about vim is that you don't have to take your fingers away from the keyboard while you're using it, which makes us faster and more productive.

Vim served as the default text editor doing the development of this project.

6.2.3 OpenCV library — image-processing

Our program needs to access every bit of every pixel of image of many formats. For that, we need a powerful image-processing library. And after researching for a while, OpenCV was the obvious choice. It is an actively developed open-source C++ computer vision and image-processing library that provides lots of facilities to handle and manipulate images.

We used OpenCV to access the bits of image and to support many image formats.

6.2.4 gtkmm library — GUI

gtkmm is the official C++ interface for the popular GUI (graphical user interface) library *GTK+*. It is a free and open-source library distributed under the GNU Lesser General Public License (LGPL). It allows the creation of user interfaces either in code or with graphical interface designers. Other features include type-safe callbacks, comprehensive set of graphical control elements (widgets), and the extensibility of widgets via inheritance.[9]

Main features of gtkmm are listed as follows:

- Use *inheritance* to derive custom widgets.
- Type-safe signal handlers, in standard C++.
- *Polymorphism*.
- Use of Standard C++ Library, including *strings*, *containers*, and *iterators*.
- Full internationalization with *UTF-8*.
- Complete C++ memory management:
  - Object composition.
  - Automatic deallocation of dynamically allocated widgets.
- Full use of C++ *namespaces*.
- No macros.
- Cross-platform: Linux, FreeBSD, Solaris, Win32, Mac OS X, others.

We used Gtkmm because other libraries make use of ugly macros and redefine features that are already available in the standard library.

6.2.5 Crypto library — cryptography

The crypto library is a library of basic cryptographic functions provided by the OpenSSL project. OpenSSL is the open-source implementation of the *Secure Sock-*

ets Layer (SSL) and *Transfer Layer Security* (TLS) protocols. It is written in C, a procedural language, but we decided to use it due to its simplicity.

We used this cryptographic library to encrypt the password using the SHA1 one-way hashing algorithm, making the password practically impossible to recover.

6.2.6 Modified LSB substitution algorithm

The concept of the classical LSB substitution algorithm is very simple. All it does is hide every *bit* of the information (to be hidden) in the *least* significant bits of the color values in the image pixels. Although this algorithm is very good at keeping the stego-image identical to the carrier image, it is very trivial and easy to crack. Anyone with a slight knowledge of programming can decrypt the information which is encrypted using the classical LSB substitution algorithm. Therefore, it is not secure and not acceptable for our project.

We wanted the hidden information to be secured by password, and not by algorithm. So we modified the LSB algorithm to use *less* significant bits (not always least) to store the information bits. We made the pattern of bits to use dependent on the password.

6.2.7 Git — source code management system

Git is a distributed revision control system with an emphasis on speed, data-integrity, and support for distributed, non-linear workflows. It is, by far, the most widely adopted version control system for software development.

Our project was developed as a git repository from the beginning. Git helped us manage our project development more smoothly. It also helped us collaborate on this project individually.

7 Implementation

Photocrypt has been implemented with a modified (better) LSB substitution steganography algorithm. It is not the most sophisticated and elegant solution but it is much better than the classical LSB substitution algorithm. We have created two user defined classes, `MatImage` and `TextFile`, to represent an image and a text file respec-

tively. These classes work together, along with standard C++ classes, to effectively hide the contents of the text file in the image.

7.1 Block Diagram

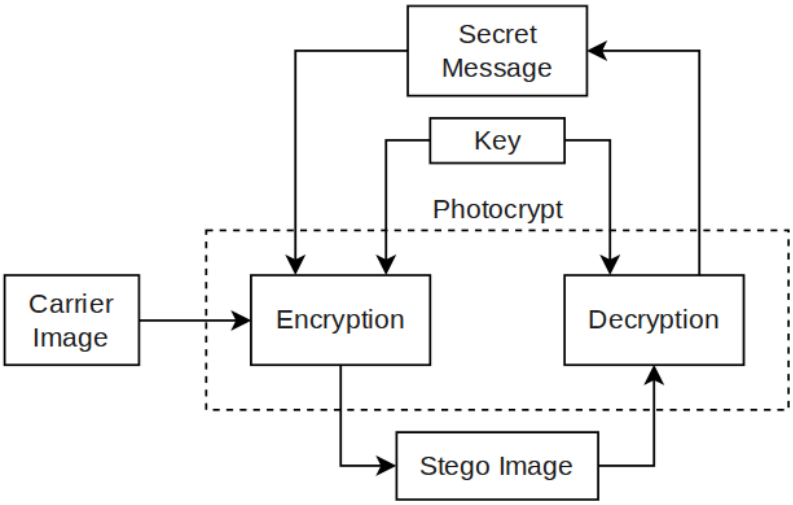

Figure 1: System block diagram

8 Results

We tested our program in various types of computers and various types of bitmap images. We noticed that except in some plain colored images, the stego-image produced by our program is practically indifferentiable with the original image.

Our program can hide, in an image, a text file of size at most $\frac{3n}{8}$, where $n =$ number of pixels in the image. For example, an image of size 500×500 can hide, at most, approximately 93 KiB of text, which is, relatively, a *very* large amount of text.

8.1 Screenshots

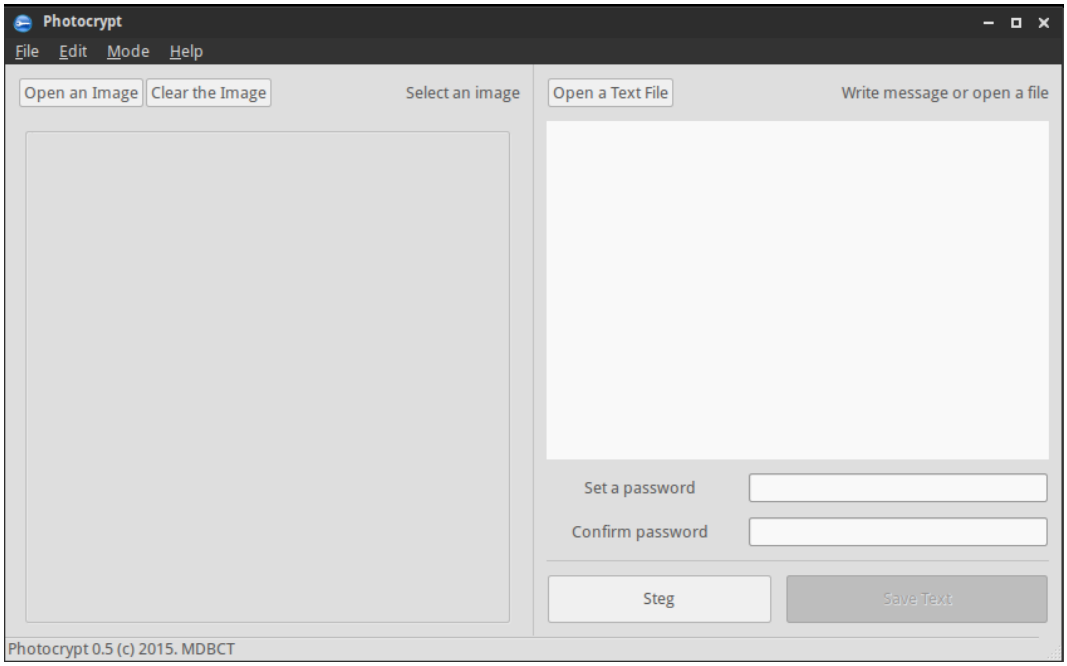

Figure 2: Default view of the program

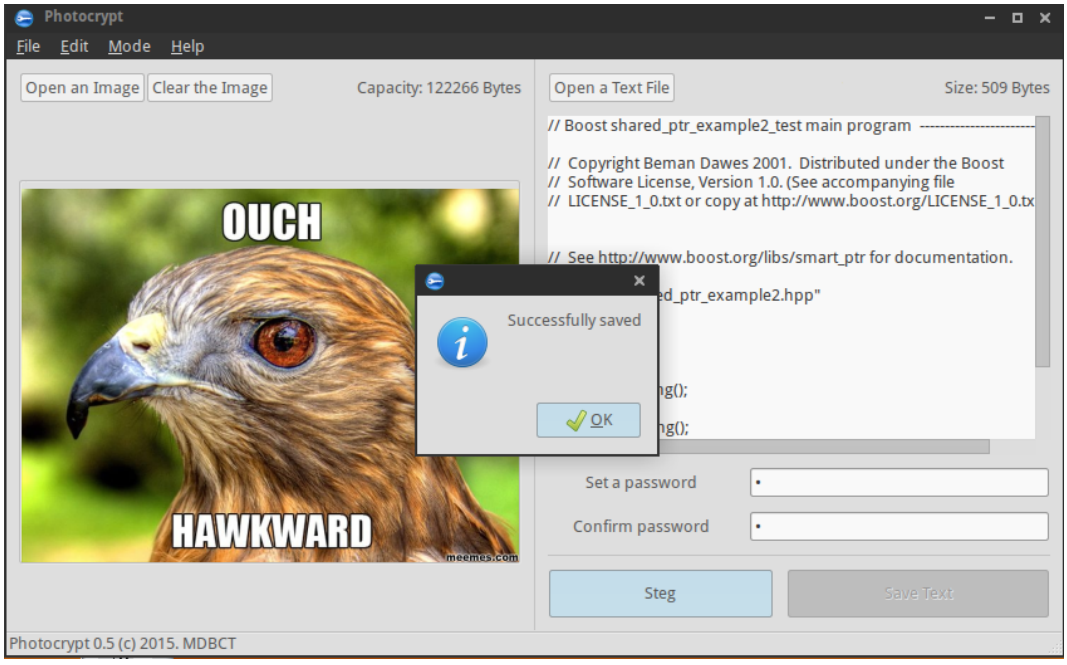

Figure 3: Stego-image is successfully written

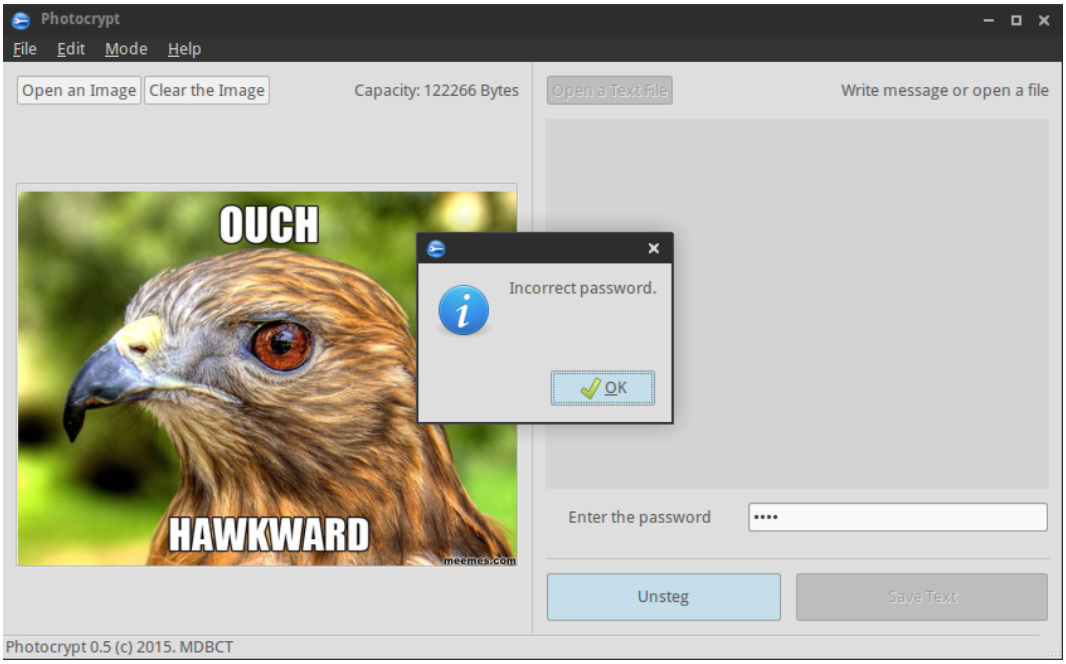

Figure 4: Incorrect password

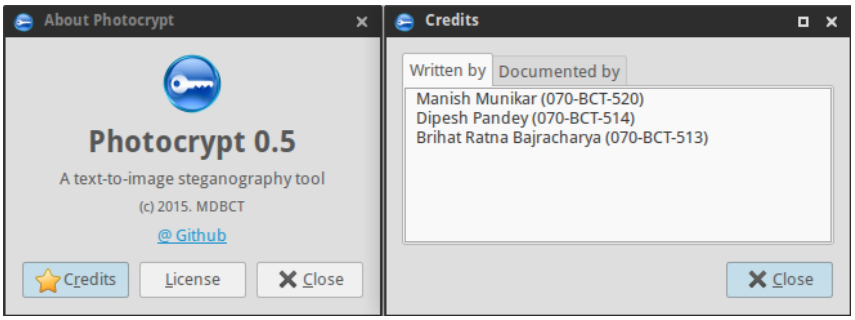

Figure 5: About dialogs

9 Problems Faced and Solutions

The problems faced[2] while developing this project are discussed below:

9.1 Image-processing

Our project requires us to be able to access images bit-by-bit. For this, we need to understand exactly how the various image formats work. This task would have been really tedious. Fortunately, there are some image-processing libraries that let you

[2]Most of our programming problems were solved, thanks to stackoverflow.com[1]

play with different image formats without having to know the details of that format. After all, an image is just a matrix of pixels. So, we needed something that let us use image of many formats as single data type. Fortunately, we found that there are many libraries that do this. We chose to use *OpenCV* library at last because of its popularity and power.

9.2 Steganography algorithm

Our project needs to store a text in image without changing the image in a noticeable way. After researching for an algorithm that can do this, we were directed, by most, to something called the *LSB Substitution Algorithm*. All it does is hide every *bit* of the text in the *least* significant bits (LSBs) of the image's pixel's color values. After coding up this algorithm, we realized two things:

1. The "hiding" part is done exceptionally well by this algorithm.
2. But this algorithm is very trivial and very easy to crack.

So we decided to tweak the algorithm a bit to make it non-trivial and more secure.

9.3 Information Security

Even if our project can successfully encrypt a text in an image, it's of no use it anyone can easily decrypt the hidden text from the stego-image. We need to make the hidden text practically unfeasible to be decrypted without the password (set during encryption). For that we need some cryptography schemes:

- We need a way to store password (in encrypted form) in the image to check for password validity.
- We need an algorithm, to hide data, that depends on the password so that the information is protected by password instead of algorithm.

To encrypt the password, we decide to use the *SHA1* one-way hashing function provided by *OpenSSL* project's *crypto* library. For the secure algorithm, we devised our own modification of the LSB Substitution algorithm by using *less* significant bits (not always *least*) which depends on the password.

9.4 Interface

The command-line interface is trivial and straight-forward. So we didn't have to waste time on that. But the graphical interface was a real problem for us. We had never made a graphical application before. So we had two problems here:

- We needed to design an interactive graphical interface.
- We needed to learn at least one external library to build that interface.

As this is our very first graphical application, we decided to make our interface very simple and uncomplicated. For the library, we had many choices like Qt, wxWidgets, Gtkmm, FLTK, SDL, etc. After researching on all of these for their pros and cons, we chose *Gtkmm* because we found it to be closer to the standard C++ (i.e., it doesn't use unnecessary macros) and also because it is the C++ port of GTK+, which is by far the most used graphical interface library in Linux, which is the platform we're mostly targeting.

9.5 Source code management

Source code management is an absolutely vital tool for managing large software projects. Without it, it's very difficult to keep track of the software development history and to manage bugs and add features. We used *Git* as the source code management system from the beginning because at the moment, it is no doubt the most widely adopted one and is easy to use.

10 Limitations and Future Enhancements

Although we have tried to make our project as complete and flawless as possible, as nothing is perfect, it still has some limitations with possibilities for future enhancements. They are discussed as follows:

10.1 JPEG image support

Our project cannot encrypt/decrypt text to/from images of JPEG formats. This is because the JPEG image format implements a *lossy* compression algorithm (which is beyond our scope) and alters the bits to decrease the file size of the image. Even if

we know the decompression algorithm, it is still very hard because there is not one, but many, versions of the JPEG compression algorithm.

But if one is very competent in image processing, which we are not at this moment, maybe he/she can somehow make our application to be used with JPEG images. It would be very practical because JPEG images have smaller sizes and are ideal to be transfered through internet.

10.2 Encryption of other file types

Right now, our application can only encrypt/decrypt text files. But sometimes we want to encrypt other file types as well, such as secret pictures, videos, or even executable files. We haven't been able to add this feature due to time and knowledge constraints.

But our application is developed with extensibility in mind. And for someone having enough knowledge, it must not be a *very* difficult task to extend the application to include these features.

10.3 Transfer of file within application

While our program does a good job of hiding information in image securely, it cannot be used directly to transfer the stego-image to the indented recipient. We would have liked to make this program able to transfer the stego-image using networks, but we couldn't do so due to time and knowledge constraints.

Again, we have developed this project with extensibility in mind and it can be extended to have this feature as well.

11 Conclusion

Hence, the project was completed in time and we were able to successfully fulfill our objectives.

In the present era, where information privacy and security is a vital as well as controversial issue, we have built a software that aims to help people maintain digital privacy and security.

References

[1] Stack overflow. `https://stackoverflow.com`.

[2] Wikipedia. `http://en.wikipedia.org`.

[3] U. Rizwan & H. Faheem Ahmed. A New Approach in Steganography using different Algorithms and Applying Randomization Concept. 2012.

[4] Daya Sagar Baral & Diwakar Baral. *The Secrets of Object Oriented Programming in C++*. Bhundipuran Publication, 1st edition, 2010.

[5] Marc Gregoire. *Professional C++*. John Wiley & Sons, Inc., 3rd edition, 2014.

[6] Eiji Kawaguchi. Applications of Steganography. 2015.

[7] Gary C. Kessler. Steganography: Hiding Data Within Data. 2001.

[8] Ira Pohl. *C++ by Dissection*. Addision Wesley, 2nd edition, 2002.

[9] The GNOME Project. Programming with gtkmm 2, 2011.