Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

КАФЕДРА ИНФОРМАТИКИ

Лабораторная работа № 8

**Стеганография**

Выполнил студент гр. 953501

Трофимук Г.А

Проверил

Протько М.И.

Минск, 2022

# 1. Введение

1)        Изучить теоретические сведения.

2)        Реализовать программное средство, сокрытия (извлечения) текстового сообщения в (из) JPEG изображение(я) на основе метода сокрытия в частотной области изображения.

**2. Теоретические сведения**

**Назначение стеганографической защиты**

В отличие от криптографической защиты информации, предназначенной для сокрытия содержания информации, стеганографическая защита предназначена для сокрытия факта наличия (передачи) информации.

Методы и средства, с помощью которых можно скрыть факт наличия информации, изучает **стеганография** (от греч. – тайнопись).

Методы и способы внедрения скрытой информации в электронные объекты относятся к **компьютерной стеганографии** .

**Основные стеганографические понятия**

Основными стеганографическими понятиями являются **сообщение** и **контейнер.**

***Сообщением****m ∈ M* , называют секретную информацию, наличие которой необходимо скрыть, где *M* - множество всех сообщений, обычно *M = Z2n* для *n ∈ Z.*

***Контейнером****b ∈ B* называют несекретную информацию, которую используют для сокрытия сообщений, где *B* - множество всех контейнеров, обычно *B=Z 2q* , при этом *q>>n* .

***Пустой контейнер*** (**контейнер-оригинал** )**-** это контейнер *b* , не содержащий сообщения, **заполненный контейнер** (**контейнер-результат** )*bm* - это контейнер *b* , содержащий сообщение *m* .

***Стеганографическим преобразованием*** принято называть зависимости

*F: M×B×K → B,   F -1 : B×K → M,*

которые сопоставляют тройке (сообщение, пустой контейнер, ключ) контейнер-результат, и паре (заполненный контейнер, ключ) исходное сообщение, т.е.

*F(m,b,k) = bm,k , F-1 (b m,k ) = m, где m ∈ M, b, bm ∈ B, k∈K*

***Стеганографической системой*** называют (*F, F-1 , M, B, K* ) совокупность сообщений, контейнеров и связывающих их преобразований [Мотуз].

**Методы компьютерной стеганографии**

Отметим, что, несмотря на то, что методы тайнописи известны с древних времен, компьютерная стеганография является относительно новой областью науки. В настоящее время компьютерная стеганография находится на стадии развития.

Теоретическая база и методы стеганографии только формируются, нет общепризнанной классификации методов, не существуют критерии оценки надежности методов и механизмов стеганографических систем, производятся первые попытки проводить сравнительные характеристики методов, например, в [Барсуков, 54].

Но уже сегодня специалисты признают, что «... на базе компьютерной стеганографии, являющейся одной из технологий информационной безопасности XXI века, возможна разработка новых, более эффективных нетрадиционных методов обеспечения информационной безопасности» [Барсуков, 54, с. 71].

Анализ применяемых на практике методов компьютерной стеганографии позволяет выделить следующие ***основные классы*** .

1. Методы, основанные на наличии свободных участков в представлении/хранении данных.

2. Методы, основанные на избыточности представления/хранения данных.

3.Методы, основанные на применении специально разработанных форматов представления/хранения данных.

Подчеркнем, что методы внедрения скрытой информации в объекты зависят, прежде всего, от назначения и типа объекта, а также от формата, в котором представлены данные. То есть, для любого формата представления компьютерных данных могут быть предложены собственные стеганографические методы.

Остановимся на стеганографических методах, которые часто применяются на практике.

Широко известен ***метод внедрения скрытой информации в младшие биты данных, представленных в цифровом виде*** . Метод основывается на том факте, что модификация младших, наименее значимых битов данных, представленных в цифровом виде, с точки зрения органов чувств человека не приводит к изменению функциональности и даже качества изображения или звука. Отметим, что информация, скрытая в последних битах цифрового контента не является помехоустойчивой, то есть при искажениях или сжатии с потерей данных она теряется.

На практике используются также ***широкополосные сигналы и элементы теории шума*** . Информация скрывается путем фазовой модуляции информационного сигнала (несущей) с псевдослучайной последовательностью чисел. Используется и другой алгоритм: имеющийся диапазон частот делится на несколько каналов, и передача производится между этими каналами.

Достаточно развиты ***методы, применяемые для тайнописи в текстовых файлах.***

· Скрытые гарнитуры шрифтов. Данный метод основан на внесении малозаметных искажений, несущих смысловую нагрузку, в очертания букв.

· Цветовые эффекты. Например, для символов скрываемого сообщения применяют белый цвет на белом фоне.

· «Нулевой шифр». Этот метод основан на выборе определенных позиций символов (иногда используются известные смещения слов\\предложений\\ абзацев).

· Обобщение акростиха. Метод заключается в том, что по определенному закону генерируется осмысленный текст, скрывающий некоторое сообщение.

· Невидимые коды. Символы скрываемого сообщения кодируются определенным количеством дополнительных пробелов между словами или числом пустых строк.

Разработаны ***методы внедрения скрытой информации и для файлов в форматеHTML:***

· в конец каждой строки добавляют определенное число пробелов, кодирующее скрываемую информацию;

· скрываемое сообщение размещают в специальном файле, у которого удаляют заголовок, причем такой заголовок хранится у получателя (скрываемое сообщение обычно дополнительно шифруется);

· присоединяют дополнительные страницы, на которых и размещают скрываемую информацию;

· записывают скрываемую информацию в мета-тэги (эти команды предназначены для сообщения информации о html-документе поисковым серверам и не видны при отображении страницы на экране);

· записывают скрываемую информацию в тэги с неизвестными программам-браузерам идентификаторами;

· применяют цветовые эффекты.

Особое внимание обратим на ***методы, применяемые для внедрения скрытой информации в исполняемые файлы*** .

Большинство из применяемых методов основано на наличии свободных участков в исполняемых файлах: полностью или частично свободные секторы (блоки) файла; структуры заголовков файлов в форматах EXE, NE-executable и PE-executable содержат зарезервированные поля; существуют пустоты между сегментами исполняемого кода и другие. Заметим, что именно такие методы компьютерной стеганографии традиционно используют авторы компьютерных вирусов для внедрения тел вирусов в исполняемые файлы. Обратим внимание, что для удаления скрытой таким образом информации нарушителю достаточно просто «обнулить» все имеющиеся свободные участки.

**Технологии защиты авторских прав на мультимедийную продукцию**

Среди методов и технологий, использующих стеганографическую защиту информации, наиболее развитыми представляются технологии защиты авторских прав на мультимедийную продукцию.

Предлагаемые на рынке программного обеспечения технологии и системы защиты авторских прав используют методы цифровой стеганографии *.* Системы защиты авторских прав сопровождают идентифицирующей информацией объекты, представляющие собой цифровое содержание: графические файлы, аудио- и видео файлы.

Самой известной технологией в области защиты прав автора на графическую информацию является технология Digital Water Marc (цифровой водяной знак) компании Digimarc Corporation (www.digimarc.com). Специальный программный продукт PictureMarc (ключевая часть технологии) позволяет внедрять в изображение цифровой идентификатор (метку) создателя. Для получения собственного идентификатора пользователь обязан зарегистрироваться в сервисном центре компании Digimarc (MarcCentre). Цифровая метка при внедрении в изображение кодируется величиной яркости пикселей, что определяет стойкость метки при различных трансформациях графического файла (редактирование, уменьшение/увеличение изображения, преобразование в другой формат, сжатие). Более того, цифровая метка, внедренная таким способом, не теряется даже после печати и последующего сканирования. Однако, цифровая метка не может быть ни изменена, ни удалена из маркированного изображения. Считывается цифровая метка с помощью программы ReadMarc. Специальный программный продукт MarcSpider просматривает изображения, доступные через Internet, и сообщает о незаконном использовании.

На рынке программного обеспечения в настоящее время предлагается множество систем и технологий, работающих по принципу, аналогичному цифровому водяному знаку. Все они преобразовывают идентификационный код производителя мультимедиа в невидимую цифровую метку и встраивают ее в объект защиты. Обычно такие системы называют системами цифровых водяных знаков. На рынке представлены технологии PixelTag (производства MIT Media Lab); EIKONAMARK (производства Alpha Tec Ltd.); TigerMark (компании NEC) и многие другие.

Некоторые технологии вместо термина «водяной знак» используют термин «отпечаток пальца». На рынке представлена технология FBI (Fingerprinting Binary Images) производства Signum Technologies (www.generation.net/~pitas/sign.html). Сервисные программы, использующие данную технологию, также позволяют встраивать, определять и читать «отпечаток пальца» из цифровых данных.

Внимания также заслуживают возможности комплексной системы управления электронным копирайтом Cryptolope (компании IBM), основанные на технологии Java.

Применяется на практике и специальный протокол защиты мультимедиа MMP (Multimedia Protection Protocol), разработанный для защиты от пиратства при продаже оцифрованных данных через Internet или другие каналы.

Однако необходимо заметить, что существуют и программы, удаляющие цифровые метки из файлов, содержащих изображения. Наиболее известны две из них: UnZign и StirMark, которые анонсированы в качестве средств тестирования стойкости меток, встраиваемых системами цифровых водяных знаков. Использование этих программ показывает, что на сегодняшний день «водяные знаки всех производителей уничтожаются без заметного ухудшения качества изображения» [Николенко, 56].

В настоящее время приобретают широкое распространение стеганографические продукты, позволяющие маскировать целые файлы в других файлах - файлах-контейнерах. Файлами-контейнерами обычно служат графические или звуковые файлы, иногда используются и текстовые файлы (в формате TXT и HTML). К такому классу программ относятся широко известные программы S-Tools, Steganos, Contraband, Hide4PGP и другие.

Широко известны стеганографические (недокументированные) вставки Easter Eggs (www.eeggs.com) в компьютерных программах. Разработчики программного обеспечения внедряют в свои программы самостоятельные модули, вызываемые определенной (часто довольно сложной) комбинацией клавиш или последовательностью действий. Такие программы, называемые секретами, после активизации демонстрируют различного рода шутки, развлекательную анимацию. Часто программа–секрет демонстрирует список разработчиков программного продукта, а иногда даже их фотографии. Поэтому в некоторых публикациях технологию Easter Eggs относят к технологиям защиты авторских прав на компьютерные программы.

Стеганографические методы защиты авторских прав на компьютерные программы сегодня недостаточно развиты. Так, например, среди 146 инструментальных стеганографических средств, представленных в специальном обзоре

[Steganography and Digital Watermarking Tool Table // www.jjtc.com/Steganography/toolmatrix.htm] средств, основанных на стеганографических методах и технологиях цифровых водяных знаков, анонсировано только одно средство – S–Mail производства Security Software Development (SSD) Ltd., которое встраивает скрытую информацию в EXE- и DLL-файлы.

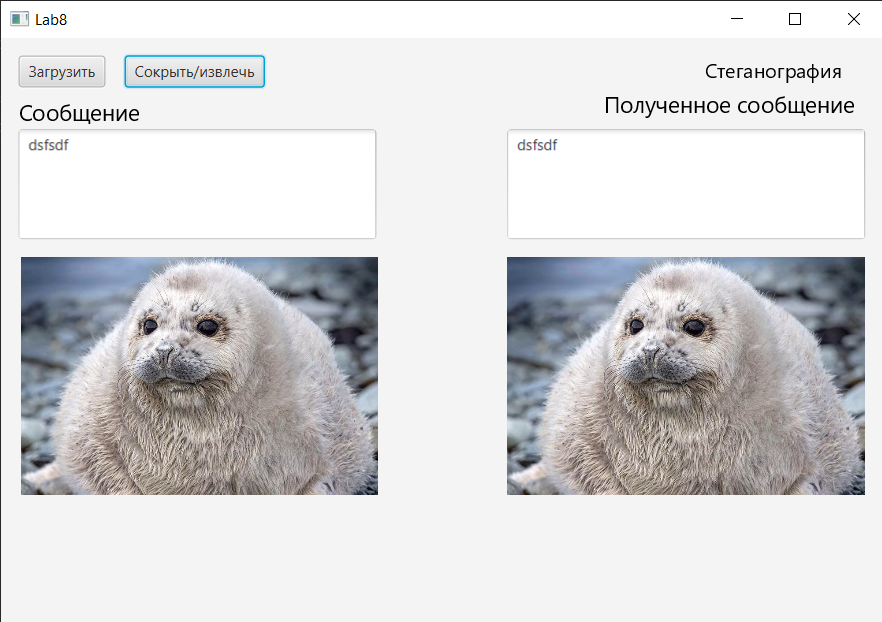
**Резюме**

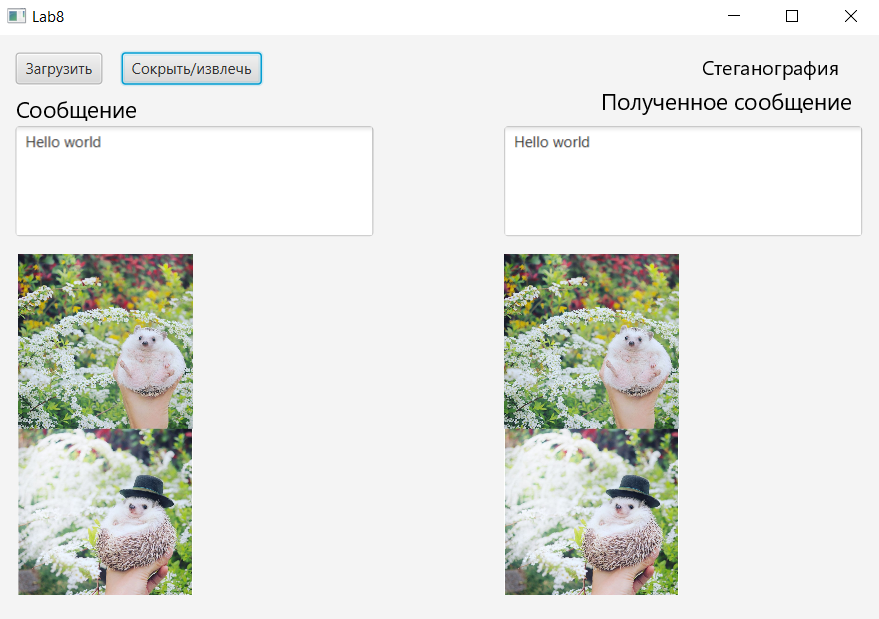
Анализ тенденций развития технологий, использующихся для обеспечения безопасности информации вообще и, в частности, для защиты авторских прав в области программного обеспечения, показывает, что *применение компьютерной стеганографии наряду с методами, традиционно применяемыми для защиты программных продуктов, увеличивает мощность механизмов защиты* .

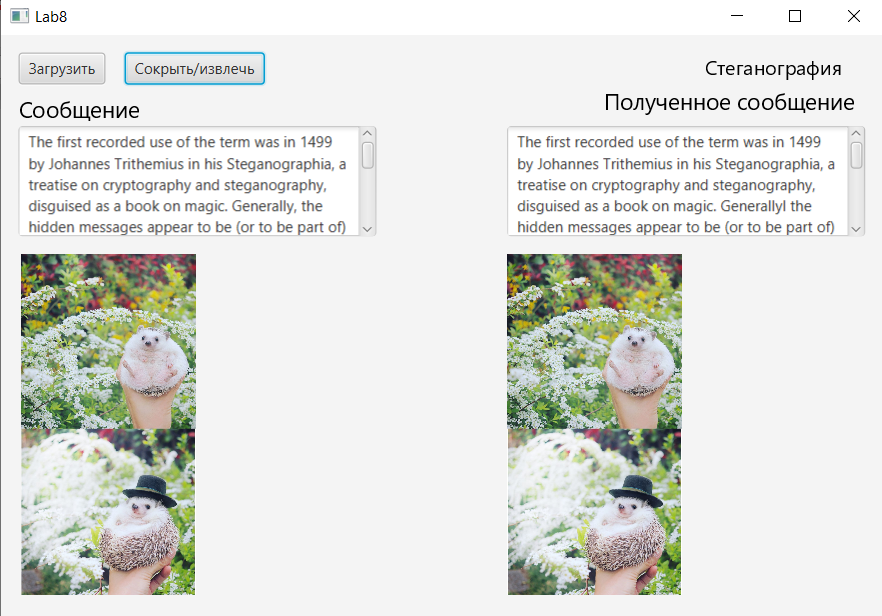
Анализ стеганографических методов защиты информации, технологий и стеганографических средств защиты интеллектуальной собственности, представленных на рынке программного обеспечения, а также проблем, связанных с применением данных методов, позволяет сделать следующие выводы.

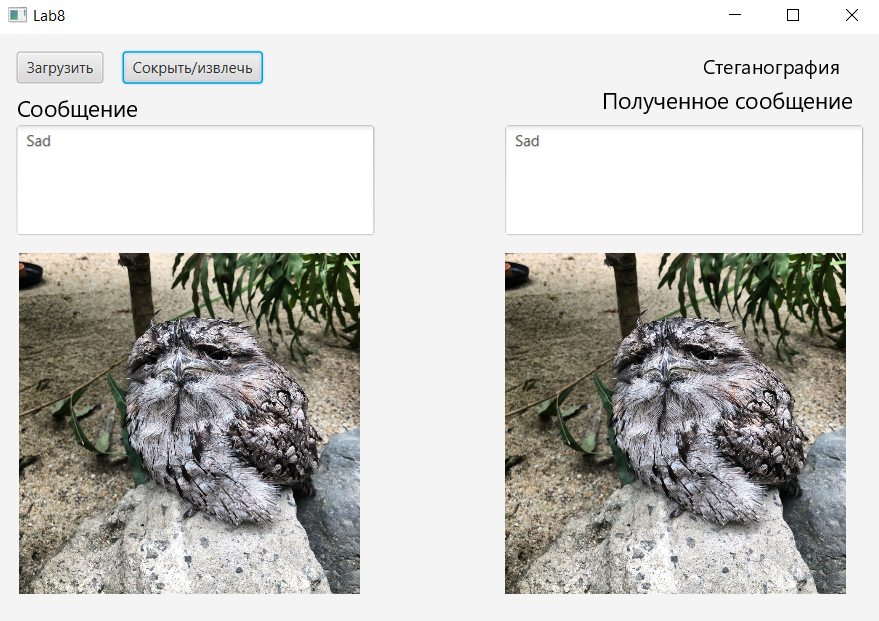
1. В настоящее время рынок программных средств защиты интеллектуальной собственности только складывается.
2. Несмотря на низкую стойкость цифровых меток, цифровые стеганографические технологии и системы успешно применяются на практике для защиты авторских прав создателей мультимедийной продукции при распространении их продуктов в компьютерных сетях и на цифровых носителях: компакт-дисках, цифровых музыкальных дорожках и видео.
3. В настоящее время среди производителей программного обеспечения только разработчики и издатели мультимедиа обладают средствами, обеспечивающими на том или ином уровне защиту прав авторов.
4. Не все существующие методы компьютерной стеганографии могут быть использованы для защиты авторских прав на компьютерные программы.
5. Наиболее развитые методы и алгоритмы компьютерной стеганографии, относящиеся к цифровой стеганографии, не могут применяться для внедрения скрытой информации, идентифицирующей автора, в исполняемые файлы программ.
6. Стеганографические методы защиты авторских прав на компьютерные программы сегодня недостаточно развиты.

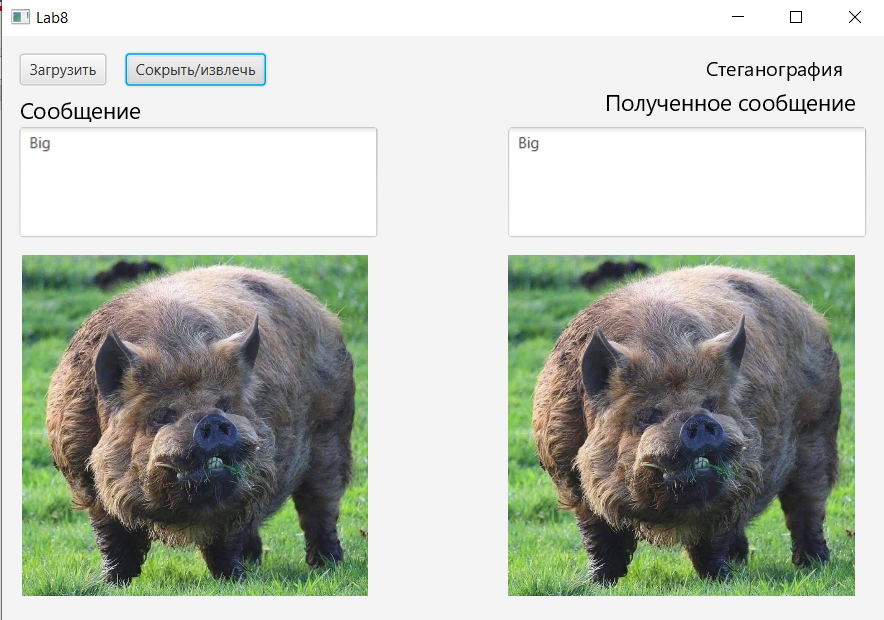
# 3. Результаты выполнения программы











# Выводы

Стеганография — способ передачи или хранения информации с учётом сохранения в тайне самого факта такой передачи (хранения).

В отличие от [криптографии](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F), которая скрывает содержимое тайного сообщения, стеганография скрывает сам факт его существования. Как правило, сообщение будет выглядеть как что-либо иное, например, как изображение, статья, список покупок, письмо или [судоку](https://ru.wikipedia.org/wiki/%D0%A1%D1%83%D0%B4%D0%BE%D0%BA%D1%83). Стеганографию обычно используют совместно с методами криптографии, таким образом, дополняя её.

Преимущество стеганографии над чистой криптографией состоит в том, что сообщения не привлекают к себе внимания. Сообщения, факт шифрования которых не скрыт, вызывают подозрение и могут быть сами по себе уличающими в тех странах, в которых запрещена криптография. Таким образом, криптография защищает содержание сообщения, а стеганография — сам факт наличия каких-либо скрытых посланий от обличения.

# КОД ПРОГРАММЫ

**Main.java:**

package com.example.lab8si;  
  
import javafx.application.Application;  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Scene;  
import javafx.stage.Stage;  
  
import java.io.IOException;  
  
public class Main extends Application {  
 @Override  
 public void start(Stage stage) throws IOException {  
 FXMLLoader fxmlLoader = new FXMLLoader(Main.class.getResource("view.fxml"));  
 Scene scene = new Scene(fxmlLoader.load());  
 stage.setTitle("Lab8");  
 stage.setScene(scene);  
 stage.show();  
 Controller controller = fxmlLoader.getController();  
 controller.setStage(stage);  
 }  
  
 public static void main(String[] args) {  
 *launch*();  
 }  
}

**Controller.java:**

package com.example.lab8si;  
  
import javafx.event.ActionEvent;  
import javafx.fxml.FXML;  
import javafx.scene.control.TextArea;  
import javafx.scene.image.Image;  
import javafx.scene.image.ImageView;  
import javafx.scene.text.Text;  
import javafx.stage.FileChooser;  
import javafx.stage.Stage;  
  
import javax.imageio.ImageIO;  
import java.awt.image.BufferedImage;  
import java.io.File;  
import java.io.IOException;  
import java.util.Objects;  
import javafx.embed.swing.SwingFXUtils;  
  
  
public class Controller {  
 private Stage stage;  
 @FXML  
 private TextArea sourceTextArea;  
 @FXML  
 private TextArea signedTextArea;  
 @FXML  
 private Text messageText;  
 @FXML  
 private ImageView sourceImageView;  
 @FXML  
 private ImageView encryptedImageView;  
 BufferedImage sourceImage;  
 String fileName;  
 public void setStage(Stage stage) {  
 this.stage = stage;  
 }  
 @FXML  
 void Load(ActionEvent event) throws IOException {  
 final FileChooser fileChooser = new FileChooser();  
 fileChooser.setInitialDirectory(new File("C:\\Users\\User\\Desktop\\Tests\\Jpegs"));  
 File file = fileChooser.showOpenDialog(stage);  
 if (file != null) {  
 fileName = file.getName();  
 sourceImage = ImageIO.*read*(file);  
 sourceImageView.setImage(SwingFXUtils.*toFXImage*(sourceImage, null));  
 }  
 }  
 @FXML  
 void EncryptDecrypt(ActionEvent event) throws Exception {  
 String message = sourceTextArea.getText();  
 messageText.setText("");  
 if (sourceImage != null && !message.equals("")){  
 int imageSize = sourceImage.getWidth() \* sourceImage.getHeight();  
 if (message.length() \* 8 + 32 > imageSize){  
 messageText.setText("Message too long for this image");  
 return;  
 }  
 embedInteger(sourceImage, message.length(), 0, 0);  
 byte[] messageBytes = message.getBytes();  
 for(int i=0; i<messageBytes.length; i++){  
 embedByte(sourceImage, messageBytes[i], i\*8+32, 0);  
 }  
  
 BufferedImage encryptedImage = sourceImage;  
  
 encryptedImageView.setImage(SwingFXUtils.*toFXImage*(encryptedImage, null));  
 saveImage(encryptedImage, fileName);  
  
 int len = extractInteger(encryptedImage, 0, 0);  
 byte[] encryptedByte = new byte[len];  
 for(int i=0; i<len; i++){  
 encryptedByte[i] = extractByte(encryptedImage, i\*8+32, 0);  
 }  
 signedTextArea.setText(new String(encryptedByte));  
 }  
 else {  
 if (sourceImage == null){  
 messageText.setText("No image");  
 }  
 if (message.equals("")){  
 messageText.setText("No message");  
 }  
 }  
 }  
 private void saveImage(BufferedImage sourceImage, String fileName) throws IOException {  
 File outfile = new File(fileName);  
 ImageIO.*write*(sourceImage, "jpeg", outfile);  
 }  
 private BufferedImage loadEncryptedImage(String fileName) throws IOException {  
 File file = new File(fileName);  
 return ImageIO.*read*(file);  
 }  
 private void embedInteger(BufferedImage img, int n, int start, int storageBit) {  
 int maxX = img.getWidth();  
 int maxY = img.getHeight();  
 int startX = start/maxY;  
 int startY = start - startX\*maxY;  
 int count=0;  
 for(int i=startX; i<maxX && count<32; i++) {  
 for(int j=startY; j<maxY && count<32; j++) {  
 int rgb = img.getRGB(i, j), bit = getBitValue(n, count);  
 rgb = setBitValue(rgb, storageBit, bit);  
 img.setRGB(i, j, rgb);  
 count++;  
 }  
 }  
 }  
 private void embedByte(BufferedImage img, byte b, int start, int storageBit) {  
 int maxX = img.getWidth();  
 int maxY = img.getHeight();  
 int startX = start/maxY;  
 int startY = start - startX\*maxY;  
 int count=0;  
 for(int i=startX; i<maxX && count<8; i++) {  
 for(int j=startY; j<maxY && count<8; j++) {  
 int rgb = img.getRGB(i, j), bit = getBitValue(b, count);  
 rgb = setBitValue(rgb, storageBit, bit);  
 img.setRGB(i, j, rgb);  
 count++;  
 }  
 }  
 }  
 private int extractInteger(BufferedImage img, int start, int storageBit) {  
 int maxX = img.getWidth();  
 int maxY = img.getHeight();  
 int startX = start/maxY;  
 int startY = start - startX\*maxY;  
 int count=0;  
 int length = 0;  
 for(int i=startX; i<maxX && count<32; i++) {  
 for(int j=startY; j<maxY && count<32; j++) {  
 int rgb = img.getRGB(i, j), bit = getBitValue(rgb, storageBit);  
 length = setBitValue(length, count, bit);  
 count++;  
 }  
 }  
 return length;  
 }  
  
 private byte extractByte(BufferedImage img, int start, int storageBit) {  
 int maxX = img.getWidth();  
 int maxY = img.getHeight();  
 int startX = start/maxY;  
 int startY = start - startX\*maxY;  
 int count=0;  
 byte b = 0;  
 for(int i=startX; i<maxX && count<8; i++) {  
 for(int j=startY; j<maxY && count<8; j++) {  
 int rgb = img.getRGB(i, j), bit = getBitValue(rgb, storageBit);  
 b = (byte)setBitValue(b, count, bit);  
 count++;  
 }  
 }  
 return b;  
 }  
 private int getBitValue(int n, int location) {  
 int v = n & (int) Math.*round*(Math.*pow*(2, location));  
 return v==0?0:1;  
 }  
 private int setBitValue(int n, int location, int bit) {  
 int toggle = (int) Math.*pow*(2, location), bv = getBitValue(n, location);  
 if(bv == bit)  
 return n;  
 if(bv == 0 && bit == 1)  
 n |= toggle;  
 else if(bv == 1 && bit == 0)  
 n ^= toggle;  
 return n;  
 }  
}