Projektarbeit 2016
TwitterStego-App

F.Zeller
E.Hubenschmidt

13. Mai 2016

Inhaltsverzeichnis

1 Funktionsweise und Beschreibung

1.1 Beschreibung

Die TwitterStego-App ist ein Programm für den PC welches Nachrichten in Bildern verstecken und wieder herausfiltern kann. Des weiteren kann man diese Bilder auf Twitter tweeten und auch nach Bildern auf Twitter suchen. Das Programm dient also zur verstecken Kommunikation über Twitter.

1.2 Funktionsweise

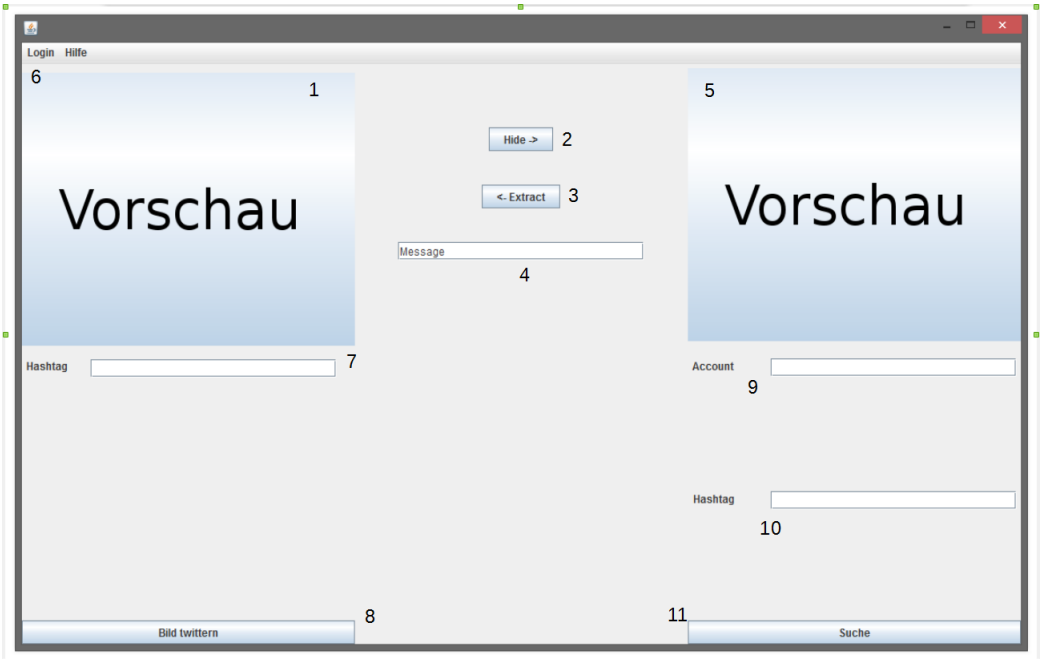

Abbildung 1: Die Oberfläche

1.2.1 Nachrichten in Bildern verstecken

Um eine Nachricht in einem Bild zu verstecken muss als erstes das Bild ausgewählt werden. Hierfür betätigt man die große Vorschau Fläche(1). Es öffnet sich ein Dateiexplorer und man kann ein Bild auswählen. Dieses Bild wird nun auf der Vorschaufläche angezeigt.
Als nächstes gibt man die zu versteckende Nachricht in das Message Feld(4) rechts neben der Vorschau ein. Um die Nachricht zu verstecken muss nur noch der Hide-Button(2) betätigt werden. Es öffnet sich wieder ein Dateiexplorer und man wählt den Speicherort für das Bild mit versteckter Nachricht aus.

1.2.2 Nachrichten aus Bildern filtern

Um eine Nachricht aus einem Bild zu filtern muss das Bild ausgewählt werden. Hierfür betätigt man die rechte Vorschau Fläche(5). Mithilfe des Dateiexplorers wählt man das gewünschte Bild aus. Es erscheint nun in der rechten Vorschau. Sobald man den Extract-Button(3) betätigt wird die Nachricht im Message Feld(4) angezeigt.

1.2.3 Der Login

Um auf Twitter zugreifen zu können muss man sich ersteinmal anmelden. Hierfür benötigt man den AccesToken, das AccesTokenSecret, den ConsumerKey sowie das ConsumerSecret. Diese vier Schlüsselbekommt man von Twitter. Man meldet sich hier: https://apps.twitter.com/ mit seinem Twitter-Account an und folgt den Instruktionen. Sobald man seine Schlüssel hat wählt man in dem Menu Login(6) den Unterpunkt Neuer Login aus. Hier gibt man seine eben erworbenen Schlüssel sowie einen Speichername für diesen Login ein. Ab sofort kann man auch die Twitterfunktionen der TwitterStego-App nutzen.

1.2.4 Gespeicherte Logins

Wenn man die vier Schlüssel eingibt muss man auch einen Namen eingeben. Mithilfe dieser Daten wird ein File erstellt welches nach schließen des Programms wieder aufgerufen werden kann. Dies wird mithilfe des Menüpunktes Login laden im Menü Login(6) bewerkstelligt. Somit muss man die Keys nicht wiederholt eingeben.

1.2.5 Ein Bild Twittern

Um ein Bild zu Twittern muss man nur einen beliebigen (optimal einen wenig oder nur eigen benutzten) Hashtag in das vorgesehene Feld(7) eintragen. Sobald der Button Bild-Twittern(8) betätigt wurde öffnet sich der Dateiexplorer und man kann das gewünschte Bild auswählen.

1.2.6 Ein Bild von Twitter suchen

Um ein Bild von Twitter herunterzuladen muss man den Accountname des Publishers sowie den zum Bild zugehörigen Hashtag eingeben. Sobald man den Button Suche betätigt(11) öffnet sich der Dateiexplorer und man wählt den Speicherort aus. Das Bild wird gespeichert und automatisch in der rechten Vorschau(5) angezeigt.

2 Steganographie

Die Steganographie ist ein Unterkapitel der Kryptographie. Ihr Ziel ist es Nachrichten in einem andern Medium zu speichern. Der begriff leitet sich aus dem griechischen ab und bedeutet "bedeckt schreiben´´. Erstmals wurde diese verfahren um 400 v.Chr. verwendet, indem man z.B. Nachrichten auf die Innenseite von Wachstafeln ritzte und dort drüber dann Wachs goss.
Heutzutage wird die Steganographie hauptsächlich in digitalen Median verwendet. Hierzu bieten sich besonders Bilder oder Audiodateien an. Ein digitales Bild, z.B. eine png-Datei, besteht aus einer Matrix von Werten. Jeder Wert stellt einen Pixel da, welcher sich wiederum aus 4 Werten zusammensetzt, nämlich dem rot, grün, blau und Alpha Wert. Diese sind jeweils 8 Bit lang, somit ergibt sich eine 32-Bit lange Zahl. Um nun eine Nachricht darin verstecken zu können, muss man diese Werte so manipulieren, dass ohne das Original, die Manipulation nicht auffällt. Dazu wird die Nachricht zuerst in eine binäre Darstellungsform (z.B. binärer ASCII-Code) gebracht. Anschließend werden die Last Significant Bit einer Farbe durch die Bits der Nachricht ersetzt. Der Alpha-Wert ist hierfür nicht geeignet, denn er gibt die Transparenz eines Pixels an. Mit dem bloßen Auge wäre zwar trotzdem kein Unterschied zu erkennen, allerdings ist der Alpha Wert normalerweise für jeden Pixel auf 255 (nicht Transparent), wodurch eine Manipulation auffallen würde.
In unserm Programm gibt eine Bitfolge von 8 Einsern in den ersten 8 Pixeln eines Bildes an, das sich in dem Bild eine Nachricht versteckt. Das Programm liest dann solange alle LSBs der Pixel, bis es auf zuerst 8 Nullen und dann auf 16 Einsen trifft. Dies bedeutet, dass die Nachricht zu ende ist. Die 8 Nullen sind dazu da, damit nicht die Einser, die zu einer Nachricht gehören als Einser dieser Schluss Bitfolge gezählt werden. Zum Beispiel ist ein Fragezeichen in binären ASCII Code eine 0011 1111. Wenn direkt darauf die 16 Einser folgen würden nur die zwei Nullen zur Nachricht zählen und den Rest als Schluss Signal.

3 Entwicklungshergang

3.1 Zeitliches Limit

Da das Schulprojekt auf nur 3 Wochen begrenzt war mussten wir uns bei der Gestaltung der TwitterStego-App beschränken. Bei der Steganographie wird eine Nachricht versteckt aber mithilfe der Kryptographie eigentlich noch zusätzlich verschlüsselt. Aufgrund der kurzen Zeitspanne haben wir beschlossen diese Funktion auszulassen. Auch wollten wir zu Anfang weitere Twitter-Funktionen einfügen wie mehrere Bilder auf einmal zu suchen und diese in einer Datenbank zu speichern. Darin hätte man Teilnachrichten verstecken und diese dann mithilfe der TwitterSego-App zusammenfügen können. Trotzdem sind wir mit dem Ergebnis zufrieden da die Grundfunktionen gewährleistet sind.

3.2 Aufteilung der Aufgaben

Da wir zu zweit mithilfe von GitHub an dem gleichen Projekt arbeiten konnten war die Teilung der Aufgaben sehr einfach. Anfangs beschäftigte sich Herr Hubenschmidt mit dem Verstecken von Nachrichten in Bildern und Herr Zeller mit der Verbindung zu Twitter. Als diese großen Teilpunkte erfüllt waren verband Herr Hubenschmidt diese und Herr Zeller gestaltete die Grafische Oberfläche.

3.3 Codeausschnitte

3.3.1 Steganographie Code

```
1  public BufferedImage hideText(String text, File img) throws IOException {
2
3      BufferedImage bufferedImage = ImageIO.read(img);
4      int w = bufferedImage.getWidth();
5      int h = bufferedImage.getHeight();
6
7
8      if (w*h -16 <= text.length() * 8) {
9        gui.error("Text ist zu lang");
10       return null;
11     }
12
13     int stelle = 0;
14
15     int[] textBits = new int[text.length() + 8];
16     textBits = bitsInText(text);
17
18     for (int i = 0; i < h; i++) {
19       for (int j = 0; j < w; j++) {
20         if (textBits.length <= stelle) {
21           break;
22         }
23
24         bufferedImage.setRGB(j, i, setLastBit(bufferedImage.getRGB(j, i), textBits[
       stelle]));
25         stelle++;
26
27       }
28     }
29
30     return bufferedImage;
31
32   }
```

Codeausschnitt

Beschreibung:
Diese Methode versteckt eine Nachricht in einem Bild.

3.3.2 Twitter Code

```
public String getTweetandMediafromHash(String hash, String user) {

    String returnvalue = null;

    try {
      ResponseList<Status> timeline = twitter.getUserTimeline(twitter.showUser(user
    ).getId());

      for (Status status : timeline) {
        for (HashtagEntity hashtags : status.getHashtagEntities()) {
          if (hashtags.getText().equals(hash)) {
            for (MediaEntity mediaEntity : status.getMediaEntities()) {
              returnvalue = mediaEntity.getMediaURL();
            }
          }
        }
      }
    } catch (IllegalStateException e) {

      e.printStackTrace();
    } catch (TwitterException e) {

      e.printStackTrace();
    }

    return returnvalue;

  }
```

Codeausschnitt

Beschreibung:
Diese Methode sucht auf Twitter nach einem Bild.

3.3.3 Gui Code

```
protected void btLoadLoginClicked() {

      String url = LoginData.class.getResource("").toString();
      url = url.substring(5, url.length());
      url = url.replace("Twitter/", "") + "LoginData/";
      File folder = new File(url);
      File[] listOfFiles = folder.listFiles();
      Vector<String> fileNames = new Vector<String>();
      for (int i = 0; i < listOfFiles.length; i++) {
        if (listOfFiles[i].isFile() && listOfFiles[i].getName().contains(".bin")) {
          fileNames.add(listOfFiles[i].getName().replace(".bin", ""));
        }
      }
      String[] fileNamesArray = fileNames.toArray(new String[0]);
      String fileName = (String) JOptionPane.showInputDialog(null, "Auswahl", "
    Auswahl",
          JOptionPane.QUESTION_MESSAGE, null, fileNamesArray, fileNamesArray[0]);
      url = url + fileName + ".bin";
      try {
        loginData = loginData.loadLoginData(url);
      } catch (FileNotFoundException e) {
        e.printStackTrace();

      } catch (ClassNotFoundException e) {
        e.printStackTrace();

      } catch (IOException e) {
        e.printStackTrace();

      }
      twitterLogin.setLoginData(loginData);
      twitterLogin.reConfiguration();
      loginDatenGesetzt = twitterLogin.checkLogin();

  }
```

Codeausschnitt

Beschreibung:
Diese Methode wird aufgerufen wenn ein bestimmter Button betätigt ist.

4 Reflexion

Durch schon vorherige zusammenarbeitet fiel uns diese Projekt nicht allzu schwer. Es war etwas Einarbeitungszeit in bestimmte Themen wie die Twitter4J Bibliothek oder die Steganographie nötig aber keine allzu großen Hürden. Mit dem Resultat sind wir beide zufrieden. Es gab nie oder keine Unstimmigkeiten und Probleme wurden schnell diskutiert und behoben. Alles in allem ein gelungenes 3 Wochen Projekt an dem wir viel Spaß hatten