Projektdokumentation

«Steganographie»

Renato Estermann Niki Hausammann

Inhalt

Einführung

Motivation

Für dieses Projekt wollten wir uns mit einem Thema aus der Kryptografie auseinandersetzen. Beim durchforschen des Fachgebietes sind wir mit dem Begriff «Steganographie» in Berührung gekommen. Bei der Steganographie handelt es sich im Wesentlichen um das Verstecken von Informationen in einem Trägermedium.

Der Gedanke, dass man Informationen versteckt versenden kann und der Empfänger die verborgene Information wieder herausfiltern kann, war für uns der Anstoss, dass wir uns im Rahmen dieses Projektes mit der Umsetzung einer solchen Lösung beschäftigen wollten.

Ziel

Die Steganographie hat Geheimhaltung und Vertraulichkeit zum Ziel: So wollen wir eine Applikation entwickeln, welche Bildern und Audiodateien als Trägermedium verwenden kann um andere Daten darin zu verstecken. Unser ist es wichtig, dass unabhängig von den verwendeten Algorithmen keine Einschränkungen bei den versteckten Daten haben. Diese soll uns ermöglichen beliebige Textdokumente, Bilder, Audiodateien oder sogar EXE-Dateien in den genannten Trägermedien zu verbergen.

Nebst dem Verstecken von Informationen ist natürlich das Herausfiltern oder Suchen ebenso wichtig und stellt den zweiten wichtigen Bestandteil der Applikation dar.

Mit unserer Applikation soll als Beispiel das folgende Szenario ermöglicht werden: Eine Person (Sender) soll Daten mit unserer Applikation in einer Trägerdatei verstecken könne. Anschliessend soll das veränderte Trägermedium über einen beliebigen Kanal an eine andere Person (Empfänger) übermittelt werden, welche wiederum mit unserem Programm die verborgenen Informationen herausfiltern kann, ohne jemals das originale unveränderte Trägermedium gesehen zu haben.

Abgrenzung

Um die gesteckten Ziele in der zur Verfügung stehenden Zeit erfüllen zu können, mussten wir einige Abgrenzungen bezüglich der Funktionalitäten und des Umfanges festlegen.

1. In unserer Applikation werden wir Algorithmen implementieren, welche es ermöglichen sollen, Dateien oder Informationen in Bild- oder Audio-Trägerdateien zu verstecken. Es werden keine weiteren Algorithmen zum Beispiel für Text- oder Video-Trägerdateien implementiert.

2. Da es bei Bild- und Audiodateien diverse Dateiformate gibt, die sich insbesondere in den Komprimierungsverfahren, als auch in der inneren Dateistruktur unterscheiden, haben wir uns auf die verlustfreien (Komprimierungs-)Verfahren beschränkt.Lediglich die Formate BMP und PNG bei Bilddateien und WAV bei den Audiodateien werden unterstützt.

Projektmanagement

Organisation
Scrum Master
Jens-Christian Fischer
Dozent an der ZHAW

Kunde
Lars Kruse
Dozent an der ZHAW

Projektmitglieder
Renato Estermann
Informatik-Student an der ZHAW, 4. Semester

Niki Hausammann
Informatik-Student an der ZHAW, 4. Semester

Iterationsplan
Der Iterationsplan besteht aus vier Iterationen, mit den jeweiligen User-Stories. Die User Stories wurden im Tool The Bug Genie erfasst, sowie den verwendeten Aufwand.
http://esternet.dyndns.org/thebuggenie-3.2rc8/thebuggenie/swp2stegano

1.Iteration

	User Story Titel	Beschreibung
1	Einlesen von Trägerdaten (Bild)	Der User kann ein Bild, welches als Trägermedium für die versteckten Informationen/Daten dient, in das System einlesen.
2	Einlesen von Trägerdaten (Bild) mit versteckten Informationen	Der Benutzer kann ein Bild mit den versteckten Informationen ins System laden.
3	Textinformationen ins System laden	Der User kann einen Text mit verstecktem Inhalt ins System laden.

2.Iteration

	User Story Titel	Beschreibung
4	Bildexport	Der Benutzer kann das modifizierte Bild aus der Applikation exportieren damit es weiter versendet oder verarbeitet werden kann.
5	«Steganographisieren»: Informationsdaten in Trägerdaten (Bild) verstecken	Der Benutzer kann die zuvor eingelesenen Informationsdaten in einer Trägerdatei hier ein Bild verstecken.
6	Gegenüberstellung des Original und des modifizierten Bildes	Die originale Trägerdatei und das neu modifizierte Bild werden dem Benutzer in der Applikation, zum Vergleich, angezeigt.

3.Iteration

	User Story Titel	Beschreibung
7	Verborgene Daten aus einer Bilddatei wiederherstellen	Der Benutzer kann das modifizierte Bild mit den verborgenen Informationen einlesen (siehe User Story #2) und die verborgenen Daten werden mit einem Algorithmus wieder herausgefiltert.
8	«Steganographisieren»: Informationsdaten in Trägerdaten (Audio) verstecken	Der Benutzer kann die zuvor eingelesenen Informationsdaten in eine Audiodatei, welche als Trägerdatei dient, verstecken.

4.Iteration

	User Story Titel	Beschreibung
9	Verborgene Daten aus einer Audiodatei wiederherstellen	Der Benutzer kann die modifizierte Audiodatei mit den verborgenen Informationen einlesen (siehe User Story #2) und die verborgenen Daten werden mit einem Algorithmus wieder herausgefiltert.
10	Veränderungsgrad der Trägerdatei	Der Benutzer, welcher die Information in der Trägerdatei versteckt, kann den Veränderungsgrad (Stärke der Veränderung od. Verunreinigung), der angewendet werden soll, angeben.

Theoretische Grundlagen

Die Steganographie ist die Kunst Informationen in einem Trägermedium zu speichern/übermitteln.Das Wort kommt aus dem griechischen und bedeutet übersetzt «verdeckt schreiben». Das modifizierte Medium wird als Steganogramm bezeichnet. Historisch ist diese Methode schon lange bekannt und wurde schon abseits der digitalen Welt erfolgreich eingesetzt. Zum Beispiel wurden Nachrichten auf die Kopfhaut von Sklaven tätowiert und der Empfänger hat diese dann geschoren um an die Nachricht zu kommen. Auch uns allen bekannte Hilfsmittel wie «Geheimtinte» fallen unter Steganographie.

Sicherheit

Gemäss dem allgemein bekannten Prinzip von Kerckhoff ist uns bekannt, dass die Sicherheit eines Systems nicht von der Geheimhaltung des Algorithmus abhängen darf, sondern einzig und allein von der Geheimhaltung eines Schlüssels. Wenn wir nun die Steganographie betrachten, fällt uns auf, dass das Verbergen einer Nachricht keine Datensicherheit garantieren kann.

Wir haben uns bei der Umsetzung auf ein einfaches Verstecken und Herausfinden beschränkt und daher ist diese Methode nicht sicher.

Technische Steganographie

Wir beschränken uns bei der Umsetzung auf die technische, computergestützte Steganographie. Möglichkeiten dazu bieten vor allem Dateiformate, welche nicht von einem Computer weiterverarbeitet sondern von einem Menschen interpretiert werden. wie zum Beispiel Bildformate aber auch Audio- oder Videodateien. Informationen werden also so verborgen, dass einer dritten Person neben den offensichtlichen Informationen des Trägermediums nichts auffällt. Damit ist zugleich erreicht, dass geheime Informationen nicht Dritten bekannt werden.

Praktische Umsetzung

In diesem Kapitel zeigen wir auf, wie wir Informationen in anderen Dateien verstecken und wieder herausholen.

Grundlage

Wir modifizieren Dateien auf Byte-Ebene. Das heisst, dass wir die Trägerdatei zuerst einmal Byte für Byte einlesen, die Information Bit für Bit «einpflanzen» und danach die modifizierte Datei wieder auf dem Dateisystem ablegen. Die meisten Audio-, Video- und Bildformate sind schon mehr oder weniger als Bytes organisiert. Wenn wir als Beispiel einen Bildpunkt eines BMP-Files anschauen sehen wir, dass dieser durch 3 Bytes beschrieben wird: je ein Byte für den Blau-, Rot- und Grünkanal. Ein Weisser Bildpunkt präsentiert sich als 1111 1111 1111 1111 1111 1111 in binär oder 255 255 255 in hex. Damit können 16'777'216 unterschiedliche Farben repräsentiert werden.

Man geht heute davon aus, dass das menschliche Auge etwa 400'000 Farbwerte voneinander unterscheiden kann1, was offensichtlich viel weniger ist, als wir mit einer BMP-Datei darstellen können. Wenn wir nun hingehen bei jedem Byte das «least significant byte» (LSB) modifizieren, haben wir immer noch 221 = 2'097'152 unterschiedliche Farbwerte, was nach wie vor deutlich mehr ist als der durchschnittliche Betrachter erkennen kann. Wir können also sagen, dass bei einem Verfahren, welches jeweils das LSB jedes Bytes modifiziert, optisch keine Manipulation der Bilddaten auffällt.

Das Ziel unserer Methode ist es, Daten, welche der Mensch mit seinen Sinnen interpretiert so zu verändern, dass die Veränderung nicht wahrgenommen wird.

Header – Aufbau der versteckten Daten in der Trägerdatei

Aufgrund der Anforderung beliebige Daten in der verwendeten Trägerdatei verstecken zu können, mussten wir zusätzliche Informationen in der Trägerdatei hinterlegen. Damit wird sichergestellt, dass beim Herausfiltern die verborgene Datei auch wieder komplett rekonstruiert werden kann. Wir verfolgten für die Lösung dieses Problems den Ansatz eines Protokolls, wie man es aus der Netzwerk- und Nachrichtentechnologie kennt. Unser Protokoll setzt sich aus den folgenden Feldern zusammen:

- **File-Extension (3 Bytes):**
  Dieses Feld repräsentiert die Dateierweiterung, wobei jedes der drei Byte einen Buchstaben (US-ASCII codiert) der Erweiterung beinhaltet. Die Bytes werden sequentiell gelesen.

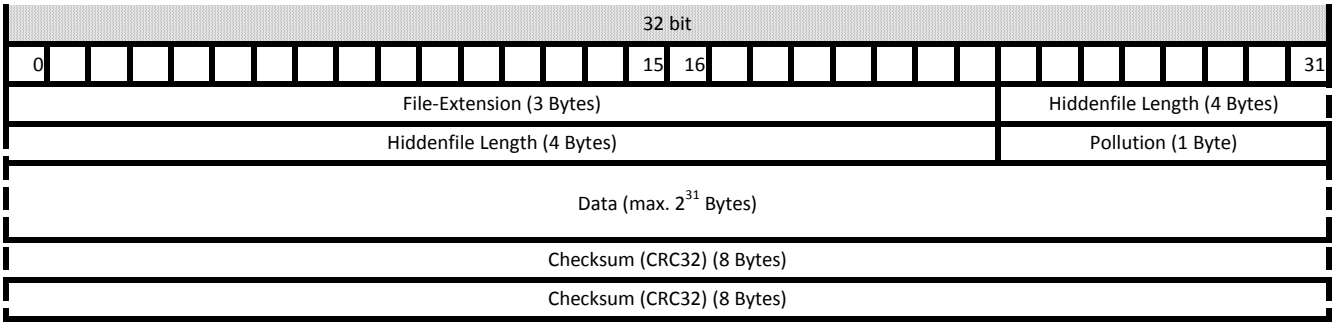
- **Hiddenfile Length (4 Bytes):**
  In diesem Feld wird die Länge der versteckten Datei (in Bytes) angegeben. Es handelt sich hierbei um einen «signed Integer Wert», der positiv sein muss. Dadurch ergibt sich auch die maximale Anzahl Bytes von 231 Bytes im Feld Data.

Praktische Umsetzung

- **Pollution (1 Byte):**
  Dieses Feld gibt an, wie gross die Verunreinigung des ausgelesenen Wertes aus der modifizierten Trägerdatei ist. Die Verunreinigung kann je nach Steganographie-Algorithmus unterschiedlich interpretiert werden. (zum Beispiel Anzahl verunreinigte Bits in einem Byte oder Anzahl verunreinigte Bytes in einem Wort)

- **Data (n Bytes):**
  In diesem Feld werden die eigentlichen Nutzdatenbzw. die eigentlichen Bytes, die die versteckende Datei repräsentieren, abgefüllt. Die maximale Anzahl an Bytes wurde bereits im Feld «Hiddenfile Lenght» thematisiert.

- **Checksum (CRC32) (8Bytes):**
  Abschliessend wird eine Prüfsumme (basierend auf java.util.zip.CRC32) über den kompletten Header und die Daten erzeugt und am Ende angefügt. Dadurch wird sichergestellt, dass nachträgliche (manuelle) Manipulationen in der modifizierten Trägerdatei erkannt werden.



Praktische Umsetzung

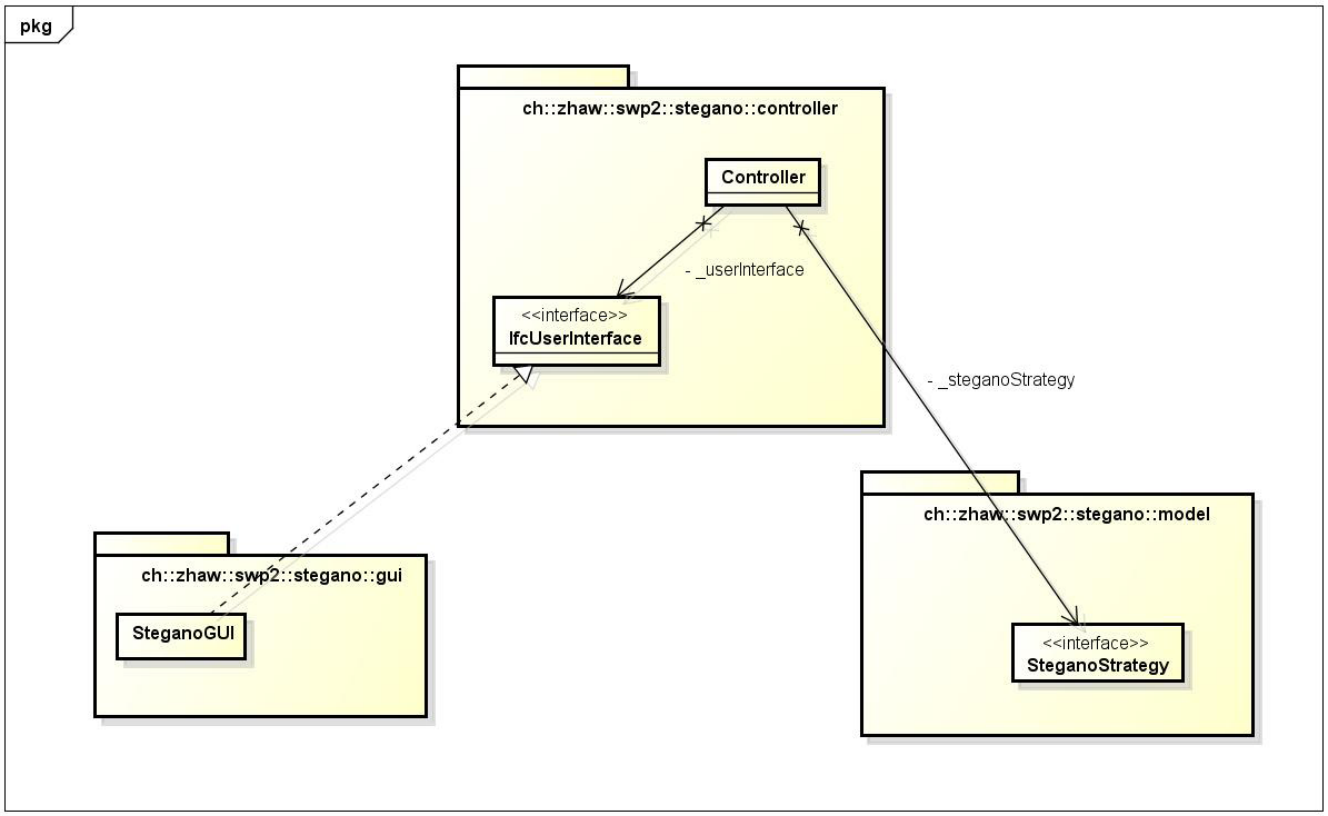
Angewandte Software Pattern

In den folgenden Unterkapiteln werden die angewendeten Pattern beschreiben.

Model-View-Controller Pattern (MVC)

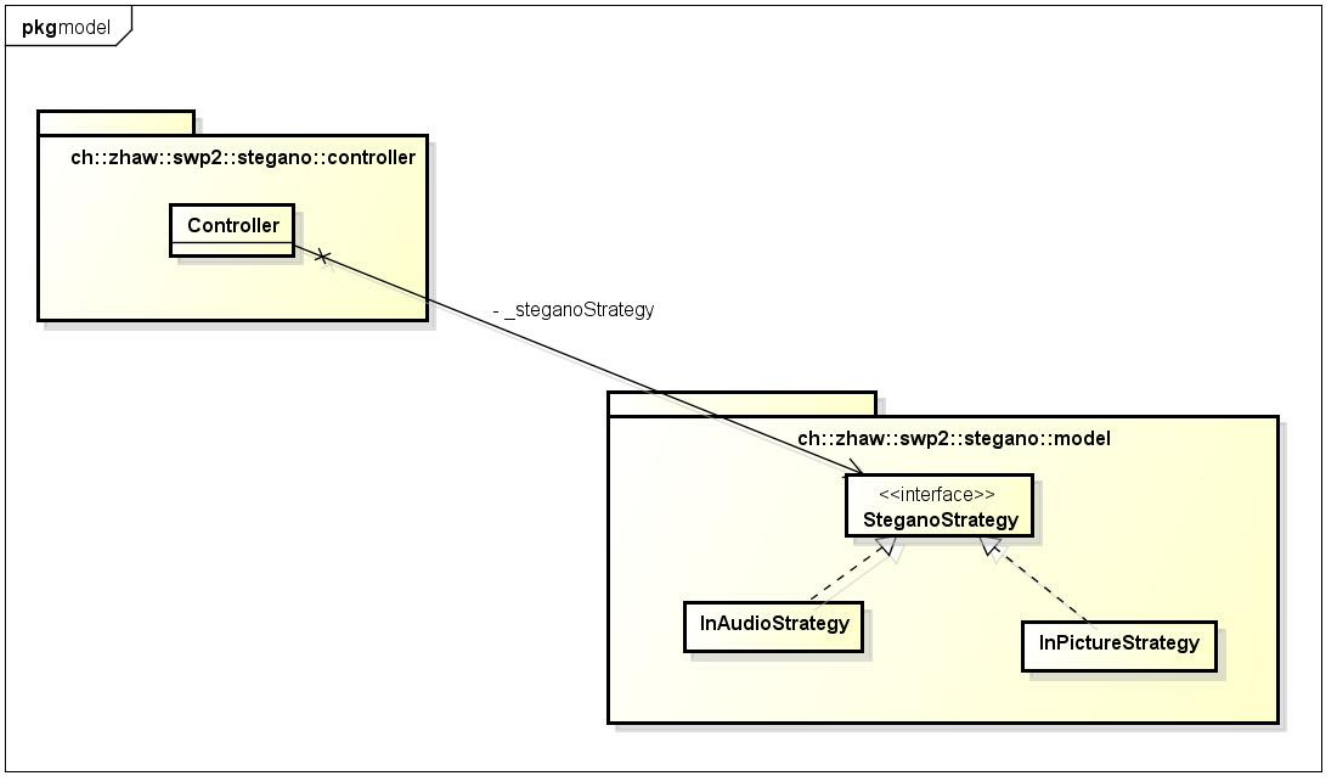
Da unser Programm zum einen Teil aus einer graphischen Benutzeroberfläche und zum anderen aus Algorithmen für den Steganographischen Teil besteht, haben wir uns für die Anwendung des Model-View-Controller (MVC) Pattern entschieden.

Durch die Aufteilung in drei Komponenten, konnte man im Model an den eigentlichen Algorithmen arbeiten, so dass diese keine Auswirkungen auf den Controller oder den GUI (View) ergab.



Praktische Umsetzung

Strategie-Pattern
Da wir in Bilddateien sowie Audiodateien Informationen verstecken wollen, haben wir uns an dieser Stelle für das Strategie-Pattern entschieden. Der Controller entscheidet aufgrund der Trägerdatei, die vom Benutzer angegeben wurde, welche Strategie respektive welchen Algorithmus, angewandt werden soll. In jedem Anwendungsfall entscheidet der Controller wieder von neuem wodurch die Austauschbarkeit der Algorithmen gewährleistet wird und übergibt der konkreten Strategie-Implementation die notwendigen Dateien und Informationen.

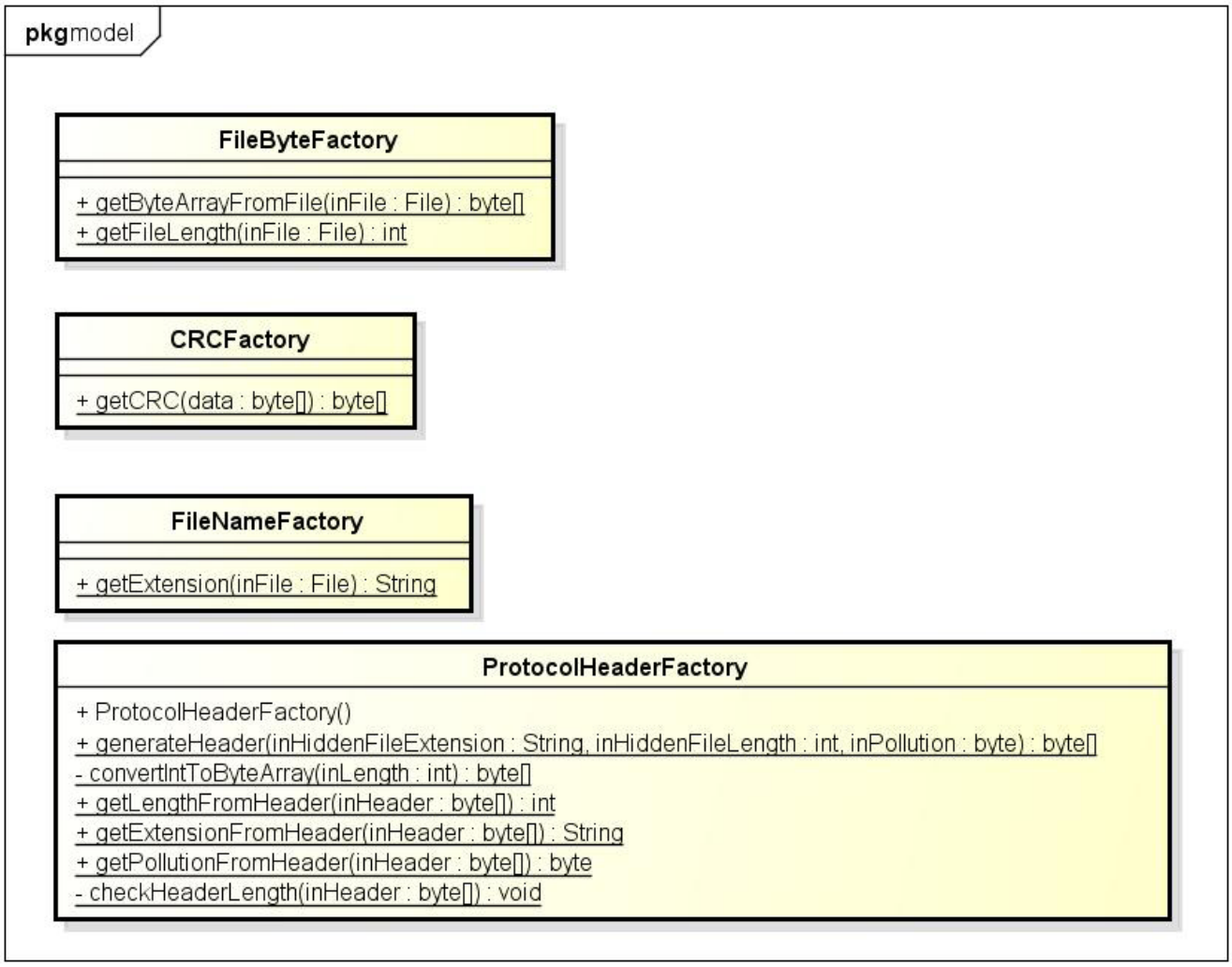


Praktische Umsetzung

Factory-Pattern
Bei den implementierten Steganographie-Algorithmen ergab sich, dass gewisse Elemente des Codes wiederverwendet werden können. Insbesondere aufgrund dessen, dass wir die Nutzdaten in das Protokoll mit dem Header, den Daten und der Prüfsumme abfüllen mussten. So ergab sich die Notwendigkeit von einigen Factories.

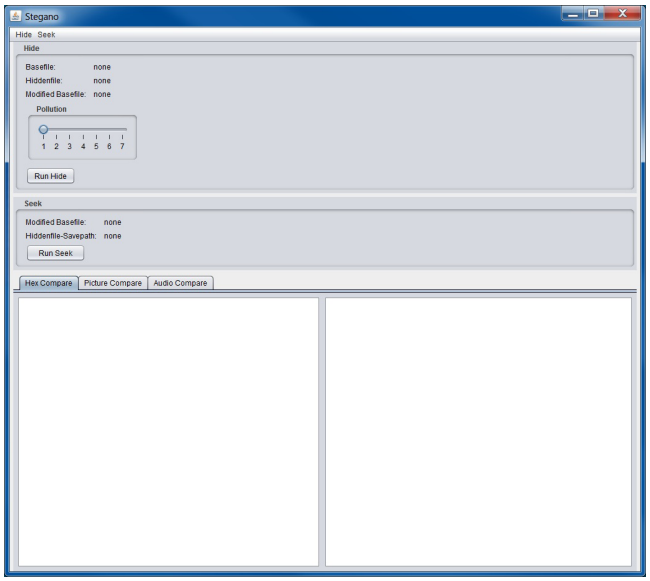
Die programmierten Factories erfüllen die folgenden Funktionalitäten:

- **FileByteFactory:**
  Diese Factory wird für Operationen im Zusammenhang mit Dateien und Byte-Arrays verwendet.
- **CRCFactory:**
  Diese Factory dient lediglich dazu die Prüfsumme (CRC32) zu errechnen.
- **FileNameFactory:**
  Die FileNameFactory dient für Operationen mit Dateinamen und Dateierweiterungen
- **ProtocolHeaderFactory:**
  Diese Factory dient zum Generieren des Headers und dem Auslesen einzelner Felder aus dem Header.



Benutzeranleitung

In diesem Kapitel wird unsere Applikation und die notwendigen Schritte erklärt: Wie wird eine Datei in einer Bild- oder in einer Audiodatei versteckt und wie wird sie wieder herausgefiltert.



Verstecken einer Datei

Um eine Datei in einer Trägerdatei zu verstecken, muss folgendermassen vorgegangen werden:

1. Über den Menüpunkt «Hide – Import Basefile» wird die Trägerdatei angeben. Erlaubte Formate sind BMP, PNG und WAV.

2. Danach muss die Datei ausgewählt werden, die in der Trägerdatei versteckt werden soll. Dies erfolgt über das Menü «Hide – Import Hiddenfile». Einschränkungen sind, dass die Dateiendung aus drei Zeichen bestehen (z.B.: txt, exe, u.s.w.) und in der Trägerdatei genügend Speicherplatz vorhanden sein muss. Die Applikation meldet allerdings zu geringer Speicherplatz.

3. Anschliessend muss der Speicherort/Dateipfad über den Menüpunkt «Save-Path: Modified Basefile» für die modifizierte Trägerdatei angegeben werden (inklusive Dateinamen und Erweiterung).

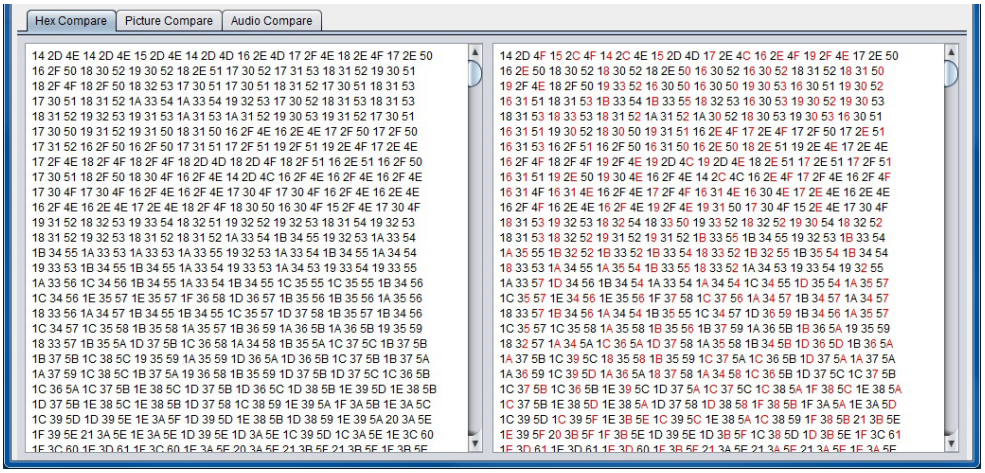
Benutzeranleitung

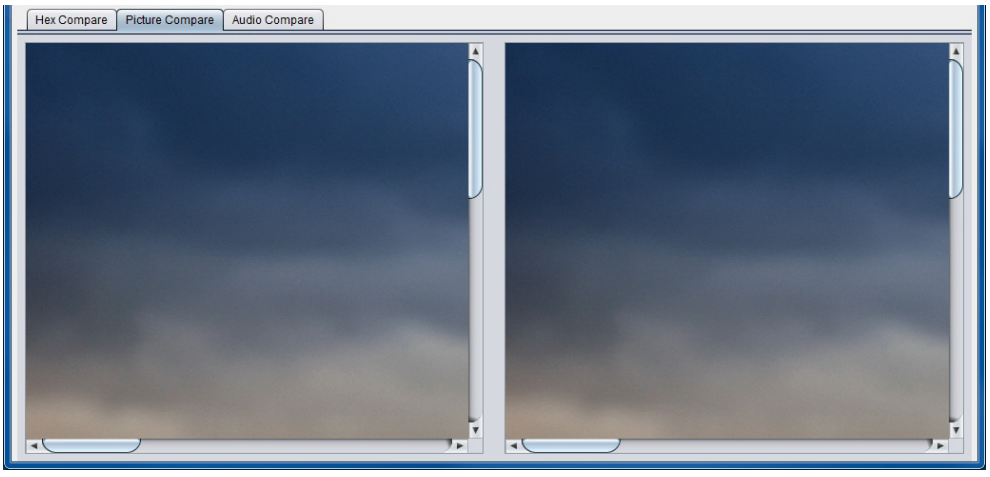
4. Nun kann man auf den Button «Run Hide» klicken und die Applikation erstellt die modifizierte Trägerdatei.

5. Anschliessend werden die Änderungen, die durch den entsprechenden Algorithmus bewirkt wurden, in einer Hexadezimal Vergleichsansicht (Hex Compare) dargestellt. Auf der linken Seite befinden sich jeweils die Originaldaten und Rechts die Veränderungen in rot eingefärbt.

a. Handelt es sich bei der Trägerdatei um ein Bild, so wird zusätzlich der Bildvergleich (Picture Compare) entsprechend aufbereitet.

b. Handelt es sich bei der Trägerdatei um eine Audiodatei, so kann man direkt über den Audiocompare das Trägermedium oder das Steganogramm mit einem systemüblichen Audioprogramm abspielen. Es sind keine Implementierung für das Abspielen oder weitere Vergleichsmöglichkeiten in der Applikation vorgenommen worden.





Benutzeranleitung

Herausfiltern einer verborgenen Datei

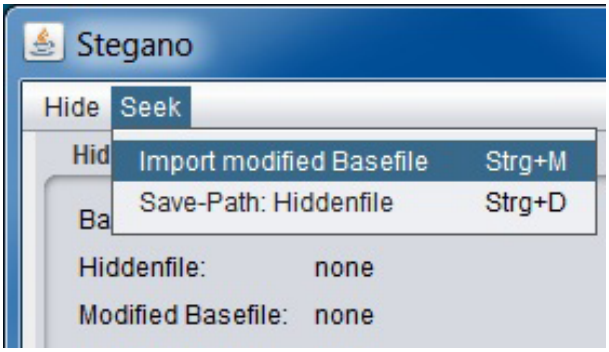
Um eine verbogene Datei aus einer modifizierten Trägerdatei wieder heraus zu filtern, muss folgendermassen vorgegangen werden:

1. Über den Menüpunkt «Seek – Import modified Basefile» wird die modifizierte Trägerdatei, ausgewählt. Erlaubte Formate sind BMP, PNG und WAV.

2. Danach muss der Speicherpfad über das Menü «Save-Path: Hiddenfile» angegeben werden, dort wo die herausgefilterte Datei gespeichert werden soll.

3. Nun kann man auf den Button «Run Seek» klicken, und Applikation filtert über den entsprechenden Algorithmus die versteckte Datei heraus.

4. Die «gefundene» Datei wird am zuvor angegebenen Ort abgespeichert und direkt mit einem entsprechenden Programm über das Betriebssystem geöffnet. (Sofern die Dateiendung im Betriebssystem mit einem passendem Editor/ Programm verknüpft ist. z.B.: .txt ➜ Datei wird in Notepad geöffnet)



Nicht erreichte Ziele

Stärke der Verunreinigung (Pollution)

Die Userstory Nr. 10 wurde ursprünglich als optionale Erweiterung vorgesehen. In Absprache mit dem Scrum Master wurde beim 4. Interationsmeeting entschieden, dass wir diese Aufgabe aufgrund Zeitmangels nicht mehr in den bereits vorhanden Algorhitmen implementieren werden.

Als Vorbereitung wurde diese Erweiterung bereits im GUI grafisch dargestellt (Pollution-Einstellskala) jedoch hat sie auf die steganographischen Algorithmen keine Einfluss.

Codierung mit Audiodateien

Unser Ziel Nachrichten auch in Audiodateien zu verstecken, haben wir leider nicht erreicht. Mit der Library (einzige welche wir gefunden haben) können zwar Audiodateien produziert werden, jedoch schleichen sich dabei Fehler ein. Eine grosse Schwäche ist, dass die Mitgelieferte Chunk-Grösse im Header nicht mir der Datei übereinstimmt. Eine eigene Implementation einer Audio-I/O-Klasse ist daran gescheitert, dass diese Dateien sehr komplex aufgebaut sind und mehrere Standards gültig sind. Es ist somit nicht möglich innerhalb eines Tasks dieses Softwareprojekt dies zu implementieren. Den Zeitaufwand für eine sauber funktionierende WAV-I/O-Klasse schätzen wir so hoch wie für die ganze restliche Arbeit.

Anhang

Verwendete Programme und Hilfsmittel

Die Applikation wurde in Java programmiert und es wurden die folgenden Programme und Hilfsmittel verwendet:

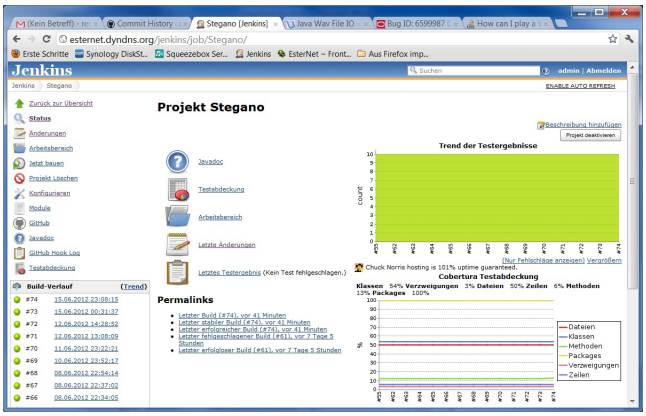
Es wurden die folgenden zwei Entwicklungsumgebungen verwendet:
Eclipse [http://eclipse.org/] und
NetBeans [http://netbeans.org/]

Für das Erstellen von automatisierten Tests wurde das folgende Framework verwendet:
JUnit [http://www.junit.org/]

Für die Verwaltung des Codes wurde das folgende Version Control System eingesetzt:
Git & Github [https://github.com]
Repository: https://github.com/esterren/stegano

Für die Planung des Projektes wurde das folgende Tool verwendet:
The Bug Genie [http://www.thebuggenie.com/]
http://esternet.dyndns.org/thebuggenie-3.2rc8/thebuggenie/swp2stegano

Für die Sicherstellung eines konsistenten Programm-Codes sowie die automatisierte Ausführung von JUnit-Test und Generierung von JavaDoc wurde das folgende Continuous Integration Tool eingesetzt:
Jenkins CI [http://jenkins-ci.org/]
http://esternet.dyndns.org/jenkins/view/SWP2%20Stegano/



Für die Manipulationen an den WAV Audiodateien wurden externe Java-Klassen verwendet, die von Herrn A. Greensted bereitgestellt wurden
http://www.labbookpages.co.uk/audio/javaWavFiles.html