Hochschule Wismar
Fakultät für Ingenieurwissenschaften
Bereich Elektrotechnik und Informatik

Bachelor-Thesis

Analyse von Werkzeugen zur steganografischen Untersuchung

Eine Bewertung von Steganografie zur Möglichkeit der verborgenen Übertragung von Informationen

Eingereicht am: 10. August 2023
Eingereicht von: Konsuela Bednarek

Studiengang IT-Forensik
Betreuer: Prof. Dr.-Ing. habil. Andreas Ahrens
Weitere Gutachter: Prof. Dr.-Ing. Antje Raab-Düsterhöft

Aufgabenstellung

Die Aufgabe dieser Bachelor-Thesis besteht darin eine Analyse von Werkzeugen zur steganografischen Untersuchung durchzuführen. Dabei wird eine Bewertung von Steganografie zur Möglichkeit der verborgenen Übertragung von Informationen vorgenommen.
Dabei wird unter anderem untersucht

- welche Werkzeuge genutzt werden können,
- ob Interoperabilität zwischen verschiedenen Werkzeugen bestehen
- ob verborgene Botschaften erkannt werden.

Mithilfe eines Python-Codes wird an einem exemplarischen Beispiel die Least significant bit Methode angewendet und anschließend überprüft, ob Werkzeuge die verborgene Übertragung erkennen können.
Das Ziel ist es, eine Bewertung von Steganografie zur Möglichkeit der verborgenen Übertragung von Information vorzunehmen und verschiedene Werkzeuge zu analysieren.

Kurzreferat

Die vorliegende Bachelor-Thesis befasst sich mit der Analyse von Werkzeugen zur steganografischen Untersuchung und der Bewertung von Steganografie zur Möglichkeit der verborgenen Übertragung von Informationen. Die Motivation für diese Arbeit liegt darin, die Techniken der geheimen Kommunikation zu untersuchen und die steganografischen Werkzeuge zu vergleichen. Die Arbeit ordnet sich in das Gebiet der Informationssicherheit ein und umfasst die Steganografie im Umfeld der Kryptografie, die sich mit der Ver- sowie Entschlüsselung von Informationen befasst. Steganografie wird in der heutigen Zeit in Ländern verwendet, wo eine Zensur besteht, wie zum Beispiel (z. B.) in China. Zudem werden steganografische Techniken verwendet, um Malware auf einem Zielsystem zu verbreiten, was erhebliche finanzielle Schäden verursachen kann. Die Abgrenzung der Arbeit liegt in darin, den Fokus auf die Untersuchung der ausgewählten steganografischen Werkzeuge zu legen. Zu den wesentlichen Schwerpunkten der Arbeit gehören die Erläuterung der „Least significant bit“-Methode (LSB-Methode), die Vorstellung sowie Untersuchung ausgewählter Werkzeuge auf verschiedenen Betriebssystemen und die Bewertung der Werkzeuge. Ein besonderer Fokus wird auf die LSB-Methode gelegt, in dem die LSB-Methode mithilfe eines Python-Codes vorgestellt und angewendet wird. Im Zuge dieser Untersuchung wird überprüft, ob die ausgewählten Werkzeuge diese LSB-Methode erkennen können. Zu den Ergebnissen der Bachelor-Thesis zählen der Vergleich der Werkzeuge anhand objektiver Gütekriterien wie Mean Squared Error (MSE), Peak Signal-to-Noise Ratio (PSNR) und Structural Similarity Index (SSIM), um die Leistungsfähigkeit bei der Einbettung und Extrahierung von versteckten Botschaften zu bewerten. Mit dieser Arbeit kann ein besseres Verständnis der Steganografie vermittelt und ein Überblick über die aktuellen verfügbaren Werkzeuge zur geheimen Kommunikation gegeben werden. Abschließend wird eine Bewertung dieser Werkzeuge vorgenommen.

Abstract

The bachelor thesis deals with the analysis of tools for steganographic investigation and evaluation of steganography for the possibility of covert transmission of information. The motivation for this thesis is to investigate the techniques of secret communication and compare the steganographic tools. The work places itself in the field of information security and includes steganography in the environment of cryptography, which deals with the encryption as well as decryption of information. Steganography is used in modern times in countries where censorship exists, such as (for example) in China. In addition, steganographic techniques are used to spread malware on a computer, which can cause significant financial damage. The delimitation of the work is in to focus on the study of the selected steganographic tools. The main focus of the work includes the explanation of the least significant bit (LSB) method, the presentation as well as investigation of selected tools on different operating systems and the evaluation of the tools. A special focus is on the LSB method, because the LSB method is presented and applied with the help of a Python code. In the course of this investigation, it is verified whether the selected tools can detect this LSB method. The results of the bachelor thesis include the comparison of the tools based on objective quality criteria such as Mean Squared Error (MSE), Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM) to evaluate the performance in embedding and extracting hidden messages. This work can provide a better understanding of steganography, an overview of the current tools available for secret communication, and an evaluation of them.

Inhaltsverzeichnis

1 EINLEITUNG 7

2 GRUNDLEGENDE INFORMATIONEN ZUR KRYPTOGRAFIE 10

2.1 DEFINITION 10
2.2 GESCHICHTLICHER HINTERGRUND 11
2.3 KRYPTOGRAFISCHE VERFAHREN 12

3 STEGANOGRAFIE 15

3.1 DEFINITION 15
3.2 ABGRENZUNG ZUR KRYPTOGRAFIE 15
3.3 GESCHICHTLICHER HINTERGRUND 16
3.4 METHODEN 18
3.4.1 Audio-Steganografie 18
3.4.2 Video-Steganografie 18
3.4.3 Text-Steganografie 19
3.4.4 Bild-Steganografie 19

4 EINFÜHRUNG IN DIE GRUNDLAGEN DER BILDVERARBEITUNG 21

4.1 PIXEL 21
4.2 BILDTYPEN 22
4.2.1 Binärbilder 22
4.2.2 Grauwertbilder 22
4.2.3 Farbbilder 22
4.3 DATEIFORMATE FÜR BILDER 22
4.3.1 Joint Photographic Experts Group – JPEG 23
4.3.2 Graphics Interchange Format (GIF) 23
4.3.3 Portable Network Graphics (PNG) 23
4.3.4 Tagged Image File Format (TIFF) 24

5 THEORETISCHE GRUNDLAGE DER LSB-METHODE 25

5.1 EINFÜHRUNG IN DIE STEGANOGRAFISCHE METHODEN 25
5.2 LSB-METHODE 25
5.2.1 Erläuterung der LSB-basierten Steganografie 26
5.2.2 Erläuterung der LSB-Methode anhand eines Beispiels 27
5.3 THEORETISCHE ERLÄUTERUNG DES PYTHON-CODES 32

6 ANALYSE DER STEGANOGRAFISCHEN WERKZEUGE 35

6.1 VERWENDETE BETRIEBSSYSTEME 35
6.2 WERKZEUGE FÜR DIE STEGANOGRAFISCHE UNTERSUCHUNG 35
6.3 KOSTENLOSE WERKZEUGE 36

6.3.1 Steghide ... *36*
6.3.2 Stegosuite ... *39*
6.3.3 OutGuess ... *41*
6.3.4 Zsteg ... *44*
6.3.5 Tater ... *47*
6.3.6 PicStealth ... *50*
6.3.7 f2d ... *54*
6.3.8 SSuite Picsel Security ... *56*
6.3.9 Stegano ... *58*
6.3.10 Stegify ... *61*
6.4 KOSTENPFLICHTIGE WERKZEUGE ... 63
6.4.1 The Hider ... *63*
6.4.2 East-tec InvisibleSecrets ... *67*
6.5 VERGLEICH ... 72
6.5.1 Objektive Gütekriterien ... *72*
6.5.2 Erklärung der Formeln MSE, PSNR und SSIM ... *73*
6.5.3 Objektive Gütekriterien (Tabelle) ... *76*
6.5.4 Bewertung der objektiven Gütekriterien ... *77*
6.5.5 Vergleich Werkzeug mit großer Datei bzw. eines Schadcodes ... *82*
6.6 AUSWERTUNG DER WERKZEUGE ... 86

7 PRAKTISCHER TEST DER LSB-METHODE UND ANWENDUNG DER WERKZEUGE 88

7.1 BILDVERSCHLÜSSELUNG MIT PYTHON ... 88
7.2 ENTSCHLÜSSELUNG MITHILFE STEGANOGRAFISCHER WERKZEUGE ... 95
7.3 ÜBERPRÜFUNG UND VERGLEICH DER ERGEBNISSE ... 101

8 FAZIT UND AUSBLICK ... 103

9 LITERATURVERZEICHNIS ... 106

10 ABBILDUNGSVERZEICHNIS ... 114

11 TABELLENVERZEICHNIS ... 118

12 ANHANG ... 121

13 SELBSTSTÄNDIGKEITSERKLÄRUNG ... 127

1 Einleitung

Motivation

Bereits in beiden Weltkriegen war die Steganografie von großer Bedeutung, denn mittels Steganografie konnten geheime Informationen z. B. über Anzahl der Kriegsschiffe, Standort der Gegner oder Besatzungsgröße übertragen werden. Aber auch heutzutage wird Steganografie noch verwendet, um z. B. Malware verdeckt zu verbreiten. Ende August 2022 wurde über einen Malware-Angriff berichtet, der in Verbindung mit einem Bild auftrat, welches über das James Webb Telescope aufgenommen wurde. Das Bild zeigte den Galaxiehaufen SMACS 0723 und wurde im Juli 2022 von der National Aeronautics and Space Administration (NASA) veröffentlicht. Verbreitet wurde die Malware über eine Phishing-Mail mit einem schädlichen Dokument „Geo-Rates.docx“, welches das Bild des Galaxiehaufens integriert hatte. Nach der Öffnung des Dokuments wurde die Malware automatisch heruntergeladen. [1] Zudem häufen sich Berichte über gefälschte Quick-Response-Codes (QR-Codes). Im Mai 2023 wurde berichtet, dass eine Frau in Singapur um 20.000 $ betrogen wurde. Die Frau wurde an einem Bubble-Tea-Laden auf einen QR-Code aufmerksam, der am Ladenfenster geklebt hatte. Dieser Sticker verleitet die Frau an einer Umfrage teilzunehmen, um einen kostenlosen Tee zu erhalten. Nach dem Scannen des QR-Codes wurde jedoch eine Drittanbieter-App auf ihr Android-Gerät heruntergeladen, welche in der Nacht von ihrem Bankkonto den Betrag von 20.000 $ abgebucht hatte. [2] Durch die Covid-Pandemie wurde das Scannen von QR-Codes z. B. für das Besuchen von Restaurants zwingend erforderlich. Dabei fehlt eine Sensibilisierung der Menschen, dass das Scannen von QR-Codes zum Herunterladen eines Schadcodes missbraucht werden kann. Durch den Einsatz von steganografischer Werkzeuge können Rückschlüssen auf versteckte Botschaften bzw. auf versteckte Inhalte wie Schadcode gezogen werden. Die vorliegende Bachelor-Thesis zur Steganografie kann einen Forschungsbeitrag leisten hinsichtlich des besseren Verständnisses für die Analyse von potenziell manipulierten Bildern und dem Verstecken von Inhalten. Die Vielzahl der Werkzeuge, welche im Rahmen der Arbeit untersucht werden konnten zeigt, dass

das Thema weiterhin von Bedeutung ist. Die oben genannten Beispiele zeigen zudem, das Hacker von diesen Techniken weiterhin Gebrauch machen, indem sie die technischen Möglichkeiten dazu nutzen, Schadcode zu übertragen.

Ausgangssituation und Problemstellung

Die Ausgangssituation dieser Bachelor-Thesis liegt in der wachsenden Bedeutung der Informationssicherheit sowie dem Schutz von vertraulichen Daten in der digitalen Welt. Den roten Faden und Forschungsrahmen der Bachelor-Thesis bildet folgende Forschungsfrage: „*Wie lässt sich Steganografie zur Möglichkeit der verborgenen Übertragung von Informationen einsetzen und vergleichend bewerten?*“. Dabei wird die Steganografie als eine Methode der versteckten Kommunikation betrachtet. Die Problemstellung der Bachelor-Thesis besteht darin, steganografische Werkzeuge zu untersuchen und zu bewerten, welche auf verschiedenen Betriebssystemen analysiert und verglichen werden. Die Herausforderung ist die Bedeutung der Steganografie für die moderne Kommunikation und Datensicherheit zu vermitteln und dabei den ambivalenten Charakter dieser Methoden verdeutlichen, da steganografische Verfahren eine Medaille mit zwei Seiten darstellt. Zum einen sind sie eine effektive Möglichkeit, Informationen von Unbefugten zu schützen, indem geheime Botschaften in z. B. Bildern versteckt werden können. Zum anderen sind sie auch geeignet um Schadcode versteckt zu übertragen und in Zielsysteme einzuschleusen.

Zielsetzung und Grenzen

Im Rahmen der Bachelor-Thesis werden ausgewählte Werkzeuge auf den Betriebssystemen Windows, Linux sowie MacOS getestet und miteinander verglichen. Dabei soll technisch untersucht werden, wie geheime Botschaften und Informationen versteckt werden können und ob sich diese ebenfalls mit den verschiedenen Werkzeugen erkennen und extrahieren lassen. Zudem soll überprüft werden, ob kostenpflichtige Werkzeuge gegenüber den kostenlosen Werkzeugen einen Vorteil im Erkennen und Extrahieren von geheimen Botschaften haben. Dabei soll unterschieden werden, ob Werkzeuge zusätzlich Passwörter für das Verstecken von Botschaften einsetzen. Im Bereich Steganografie wird die LSB-Methode besonders häufig genutzt. Daher wird die

LSB-Methode mithilfe eines Python-Codes für das Einbetten einer Geheimnachricht verwendet und anschließend das Extrahieren der versteckten Nachricht mit den ausgewählten Werkzeugen ausprobiert. Das Hauptziel ist es, einen umfassenden Einblick in die Steganografie zu vermitteln, die ausgewählten steganografischen Werkzeuge vorzustellen, ihre Anwendungsmöglichkeiten sowie eine Vergleichbarkeit der steganografischen Werkzeuge darzustellen. Die Bewertung wird mithilfe der objektiven Gütekriterien durchgeführt. Zudem wird eine Überprüfung der Interoperabilität der Werkzeuge durchgeführt. Die Bachelor-Thesis beschränkt sich auf die Steganografie und grenzt sich somit von anderen Sicherheitstechniken ab. Zudem wird eine Analyse mit ausgewählten steganografischen Werkzeugen durchgeführt. Es gibt eine Vielzahl von anderen steganografischen Werkzeugen, jedoch wird aufgrund der beschränkten Zeit nur eine Auswahl der steganografischen Werkzeuge getroffen, diese vorgestellt, analysiert und verglichen.

2 Grundlegende Informationen zur Kryptografie

In diesem Kapitel erfolgt eine Einführung in die Kryptografie. Dabei wird der Begriff der Kryptografie erläutert und kryptografische Verfahren beschrieben.

2.1 Definition

Der Begriff Kryptografie stammt aus dem Griechischen und lässt sich auf die zwei Wörter „kryptós“ und „gráphein“ zurückführen. „Kryptós“ steht für „verborgen“ und „gráphein“ für „schreiben“. [3] Mit der Kryptografie wird die Gesamtheit mathematischer Methoden und ihrer Anwendungen betrachtet, welche zum Schutz von digitaler Kommunikation sowie von digital vorliegenden Informationen eingesetzt wird. Dabei steht die Sicherstellung der Vertraulichkeit, der Integrität, der Authentizität und die Nicht-Abstreitbarkeit im Vordergrund. [4] Die Vertraulichkeit beziehungsweise (bzw.) Geheimhaltung soll sicherstellen, dass Informationen und Daten vor Unberechtigten verborgen bleiben. Die Integrität bzw. Unverändertheit stellt sicher, dass unbefugte Manipulation von Informationen oder Daten entdeckt werden können. Die Authentizität bzw. Echtheit soll beweisen können, dass die Identität und die Urheberschaft sowie die Unverändertheit der Nachricht eines Kommunikationspartner übereinstimmen. Bei der Nichtabstreitbarkeit bzw. Verbindlichkeit soll ein Kommunikationspartner nicht abstreiten können, dass eine Nachricht versandt und eine Nachricht empfangen wurde. [5] Weitere wichtige Begriffe im Zusammenhang mit der Kryptografie sind Kryptoanalysis, Kryptosysteme, Kryptoanalyse und Kryptologie. Die Kryptoanalysis beschäftigt sich mit Methoden der unbefugten Entschlüsselung von Daten, welche den Zweck der Rückführung der ursprünglichen Informationen hat. Die Kryptosysteme, welche aus einem kryptografischen Verfahren und einem dazu notwendigen Schlüssel bestehen, sind für die Geheimhaltung von gespeicherten oder übertragenen Informationen gegenüber Dritten zuständig. Mit der Bewertung der kryptografischen Stärke beschäftigt sich die Kryptoanalyse. Die Wissenschaft der Geheimhaltung von Informationen durch Transformation von Daten wird mit der Kryptologie

beschrieben. Zur Kryptologie wird die Kryptanalysis sowie Kryptografie gezählt. [6] Ferner wird die Steganografie zur Kryptologie gerechnet. [7]

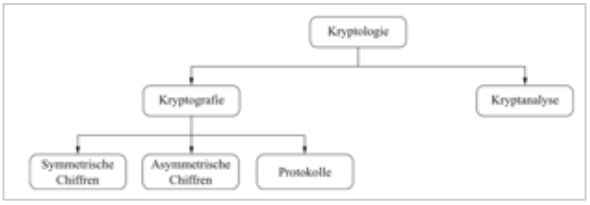

Abbildung 1 - Kryptologie und ihre Untergebiete [8]

2.2 Geschichtlicher Hintergrund

Mit Kryptografie werden moderne Begriffe wie z. B. E-Mail-Verschlüsselung oder Kryptowährungen verbunden. Jedoch ist die Kryptografie nicht nur der modernen Zeit einzuordnen, denn früherer Formen sind seit etwa 2000 vor Christus (v. Chr) in Ägypten neben den Standard-Hieroglyphen als geheime Varianten verwendet worden. In den meisten Kulturen wurden in den letzten 4000 Jahren Kryptografie mit Schrift eingesetzt, wie z. B. die Cäsar-Verschlüsselung. [8] Außerdem gehörte Kryptografie zur Domäne des Militärs, der Politiker und Diplomaten, denn das Ziel war die Entwicklung von Verfahren, welche von Gegnern nicht gebrochen werden konnten. Erst mit der Entwicklung der ersten Computer sowie Computernetze und insbesondere des Internets hat Kryptografie an Bedeutung gewonnen. Gegen Ende des 2. Weltkriegs wurden die ersten Computer entwickelt, um Geheimcodes von abgefangenen Geheimtexten brechen zu können. Kryptografie wurde revolutioniert durch die Entdeckung sowie Entwicklung vieler neuer Verfahren in den 70er und 80er Jahre des 20. Jahrhunderts. Zu den Verfahren zählen unter anderem (u. a.) Data Encryption Standard (DES), Advanced Encryption Standard (AES), Rivest-Shamir-Adleman-Algorithmus (RSA) oder ElGamal. [9] Heutzutage prägt Kryptografie durch die Nutzung von Online-Banking, der verschlüsselten Speicherung von Daten auf Information Technology Systemen (IT-Systemen) den Remote-Zugriff auf Firmennetze das Alltagsleben. Ohne den Einsatz von Techniken wie z. B. Internet Protocol Security (IPSec) und Transport Layer Security (TLS)/ Secure Sockets Layer (SSL) sind diese Tätigkeiten praktisch nicht durchführbar. Außerdem werden kryptografische Verfahren beispielsweise eingesetzt bei der Blockchain,

dem neuen Personalausweis oder bei den elektronischen Wegfahrsperren in Autos eingesetzt. [6]

2.3 Kryptografische Verfahren

Das Hauptziel der Kryptografie ist auf der einen Seite die Verschlüsselung und auf der anderen Seite die Entschlüsselung von Informationen. Das Grundprinzip wird wie folgt beschrieben: Ein Klartext wird durch das Verschlüsseln, auch als Chiffrieren bezeichnet, in einen Chiffretext bzw. Chiffrat umgewandelt, mit dem Ziel, dass ein Angreifer keinen Rückschluss auf die ursprüngliche Information ziehen kann. Mit einem Schlüssel kann der Chiffretext wieder entschlüsselt bzw. dechiffriert werden und somit in den ursprünglichen Klartext umgewandelt werden. [10]

Abbildung 2 - Einfachste mögliche Vorgehensweise bei einer Verschlüsselung [6]

Die Ziele der Kryptografie lassen sich mit geeigneten Verschlüsselungsverfahren erreichen. Dazu zählen die symmetrischen, asymmetrischen oder hybriden Verschlüsselungsverfahren. Bei den symmetrischen Verschlüsselungsverfahren wird ein und derselbe Schlüssel zur Ver- und Entschlüsselung des Klartextes verwendet. Dies setzt voraus, dass die Kommunikationspartner den Schlüssel zuvor auf einem sicheren Kanal ausgetauscht haben. Zu bekannten symmetrischen Verfahren gehören DES und AES, welche zur Gruppe der Blockchiffre gehören, da bei jedem Verschlüsselungstakt ein ganzer Block von Bits verschlüsselt wird. Geeignet ist dieses Verfahren für Online-Anwendungen, da diese bei verhältnismäßig kurzer Schlüssellänge eine hohe Sicherheit bietet. Nachteilig ist die Verwaltung des Schlüssels, da jeder Kommunikationsteilnehmer für jeden weiteren Kommunikationspartner einen eigenen Schlüssel benötigt. [5]

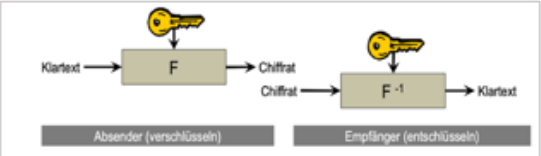

Abbildung 3 - Symmetrische Verschlüsselung [4]

Im Gegensatz zu der symmetrischen Verschlüsselung werden bei der asymmetrischen Verschlüsselung zwei verschiedene, jedoch zusammengehörende Schlüssel verwendet. Dieses Verfahren wird ebenfalls als Public-Key-Verfahren bezeichnet. In diesem Verfahren gibt es einen öffentlichen Schlüssel, den Public Key, welcher für die Verschlüsselung des Klartextes zuständig ist. Zusätzlich gibt es einen weiteren Schlüssel, den sogenannten privaten Schlüssel, der für die Entschlüsselung des Chiffrats benutzt wird. Der öffentliche Schlüssel wird bekannt gemacht und der private Schlüssel muss geheim gehalten werden. Das RSA-Verfahren, welches nach Rivest, Shamir und Adleman benannt wurde, zählt zu einem bekannten asymmetrischen Verschlüsselungsverfahren. Das asymmetrische Verschlüsselungsverfahren eignet sich besonders gut für digitale Signaturen (elektronische Unterschriften) sowie für Verbindlichkeitsnachweise. Im Vergleich zu den symmetrischen Verfahren braucht dieses Verfahren relativ lange Schlüssel. [5]

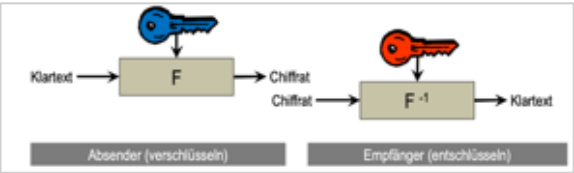

Abbildung 4 - Asymmetrische Verschlüsselung [4]

Das Ziel der hybriden Verfahren soll die Vereinigung der Vorteile der symmetrischen sowie asymmetrischen Verschlüsselung sein. Beim hybriden Verfahren übermittelt der Kommunikationspartner den Sitzungsschlüssel für eine symmetrische Verschlüsselung, die in einer asymmetrischen Verschlüsselung vorliegt. [5] Folglich wird die symmetrische Verschlüsselung aufgrund der hohen Effizienz und Geschwindigkeit für die Verschlüsselung der Daten verwendet. Die asymmetrische Verschlüsselung wird hinsichtlich der sicheren Übertragung des dafür verwendeten Schlüssels genutzt. [4]

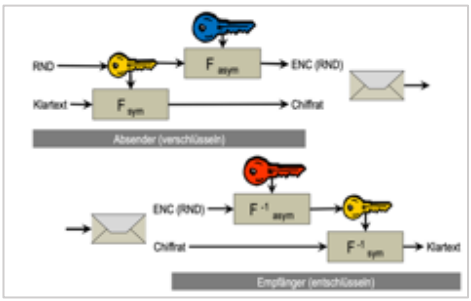

Abbildung 5 - Hybride Verschlüsselung

Des Weiteren werden digitale Signaturen als kryptische Anwendung zum Schutz der Verbindlichkeit verwendet. Sofern eine Übertragung von Nachrichten über ein offenes Kommunikationsnetz erfolgt, reicht es nicht aus, wenn nur die Vertraulichkeit der übertragenen Information geschützt wird, denn der Empfänger der elektronischen Nachricht kann lediglich auf deren Inhalt vertrauen, wenn der Absender der Nachricht zuverlässig identifiziert wird (Gewährleistung von der Verbindlichkeit) und wenn festgestellt wird, dass für den Zeitraum vom Absenden und Empfangen eine Unversehrtheit der Nachricht besteht (Schutz der Integrität). Demnach können die Verbindlichkeit sowie Überprüfung der Integrität mithilfe von digitalen Signaturen erfolgen, denn diese dienen als eine Art Siegel für digitale Daten bzw. Informationen. Vergleichbar ist eine digitale Signatur mit einer handgeschriebenen Unterschrift. Als spezielle Anwendung von asymmetrischen Verschlüsselungsverfahren sind die digitalen Signaturen einzukategorisieren. [11]

Abbildung 6 - Digitale Signatur [11]

3 Steganografie

Nach der Einführung in die grundlegenden Informationen der Kryptografie wird in diesem folgenden Kapitel die Steganografie betrachtet. Dabei wird ebenfalls die Definition sowie der geschichtliche Hintergrund erläutert. Anschließend wird die Abgrenzung zur Kryptografie erklärt und daraufhin werden Arten und Beispiele der Kryptografie aufgezeigt.

3.1 Definition

Der Begriff „Steganografie“ lässt sich auf die zwei griechischen Wörter „steganós“ und „gráphein“ zurückführen. „Stéganos“ steht für Geheimes oder Verstecktes und „gráphein“ steht für Schreiben, daher bedeutet Steganografie geheimes oder verstecktes Schreiben. [12] Das Ziel des Einsatzes von Steganografie ist die Vertraulichkeit sowie die Geheimhaltung und die Steganografie lässt sich auf zwei Weisen einordnen. Sie kann entweder als Unterkapitel der Kryptografie oder als eigenständiger Bereich betrachtet werden. Für den eigenständigen Bereich spricht, dass die Ziele der Kryptografie, nämlich die Geheimhaltung, nicht mit den Zielen der Steganografie, die vertrauliche Geheimhaltung durch das Verbergen und Verstecken der Geheimhaltung übereinstimmen. Daher wird die Kryptografie und Steganografie häufig miteinander kombiniert. [13] Beispiele der klassischen Steganografie sind z. B. unsichtbare Tinte, das Verstecken von Informationen in Bildern oder verdeckte Kommunikationskanäle. Beispiele der modernen Steganografie sind z. B. das Verstecken von Bits in Audio-, Bild- oder Textdateien. [6]

3.2 Abgrenzung zur Kryptografie

Das Besondere bei der Steganografie ist, dass das Medium, welches eine versteckte Botschaft in sich trägt, nicht für einen Dritten beim Betrachten dessen offensichtlich ist. Bei der Kryptografie ist ein Dritter in Kenntnis, dass eine Botschaft verschlüsselt ist. Im Gegensatz dazu erkennt der Dritte bei Steganografie nicht, dass eine Botschaft im Medium integriert bzw. versteckt ist.

So zielen die Techniken und Methoden der Steganografie darauf ab, geheime Nachrichten in Bildern, Texten aber auch Videos zu verstecken. [14]

3.3 Geschichtlicher Hintergrund

Die frühesten bekanntesten Aufzeichnungen über Steganografie stammen aus dem alten Griechenland. Der griechische Geschichtsschreiber Herodot erwähnte in seinen Schriften „Historien“ im 5. Jahrhundert v. Chr. den Einsatz von Tätowierung als Methode der Steganografie. Dabei wurde eine geheime Nachricht auf den Kopf von Sklaven, den Überbringern der Nachricht, tätowiert und sobald das Haar nachgewachsen war, wurde der Sklave losgeschickt, um die versteckte Botschaft zu übermitteln. Angekommen beim Empfänger wurde dem Sklaven wiederum das Haar abrasiert, um die Botschaft zu lesen. [15] Eine Blütezeit erreichte die Steganografie während des Mittelalters und der Renaissance. Johannes Trithmius, ein in Deutschland geborener Abt, veröffentlichte das erste gedruckte Buch über Kryptografie sowie über Steganografie, mit dem Titel „Steganographia“ im 16. Jahrhundert. In „Steganographia“ beschrieb er Methoden, mit denen es möglich ist, geheime Texte in einem offenen Text zu übermitteln. Ebenso beschreibt er eine steganografische Technik in Form einer Null-Chiffre. [16] Des Weiteren spielte die Steganografie in beiden Weltkriegen eine große Rolle. Im ersten Weltkrieg wurde die Methode des Codierens mit Hilfe eines Codebuches oft eingesetzt. In einem Codebuch sind Wörter, Silben oder einzelne Buchstaben, die oft verwendet werden, in geeigneter Reihenfolge notiert. Hinter diesen Wörtern stehen Nummern, in der Regel zwei- bzw. vierstellig. Der Klartext kann mit Hilfe dieses Codebuches codiert werden. [17] Das bekannteste Beispiel ist das Zimmermann-Telegramm, welches mit einem Codebuch verschlüsselt und versandt wurde. Dieses Telegramm wurde von britischen Chiffrierexperten entschlüsselt und an die USA übermittelt. Daher trug das Zimmermann-Telegramm wesentlich dazu bei, dass die USA im Jahr 1917 gegen Deutschland in den Ersten Weltkrieg eintraten. [18]

Abbildung 7 - Zimmermann-Telegramm [19]

Spezielle Maschinen, die zur Verschlüsselung verwendet wurde, wurden ab etwa 1920 eingesetzt. Die bekannteste Maschine ist das Enigma, welche die Deutschen im Zweiten Weltkrieg verwendet haben. [18] Darüber hinaus wurde im Zweiten Weltkrieg die Verwendung von unscheinbaren Buchstaben in Briefen genutzt, um geheime Nachrichten zu versenden. Diese Technik wird als Nullchiffre bezeichnet. Ein Beispiel hierfür ist das Schreiben des US-Soldaten Frank G. Jonelis, welcher im Zweiten Weltkrieg in japanische Kriegsgefangenschaft geriet. [20] Im Jahre 1943 schrieb er folgenden Brief an das Federal Bureau of Investigation (FBI):

„…*After surrender, health improved*
Fifty percent. Better food, etc.
Americans lost confidence
In Philippines. Am comfortable
In Nippon. Mother: invest
30%, salary, in business. Love
Frank G. Jonelis.” [21]

Um den Klartext zu entschlüsseln, werden die ersten beiden Wörter einer Zeile gelesen: „*After surrender Fifty percent Americans lost In Philippines In Nippon 30%.“* Mit dieser Botschaft konnte Jonelis dem FBI Informationen über das Kriegsgebiet geben [20]. Seit dem 20. Jahrhundert werden neue Methoden der Steganografie entwickelt, um geheime Botschaften in Medien wie z.B. Video-, Audio- oder Bilddateien einzubetten. [22]

3.4 Methoden

Wie im letzten Kapitel bereits erwähnt, werden seit dem 20. Jahrhundert neue Methoden der Steganografie entwickelt. Dabei können geheime Botschaften in Audio-, Text-, Bild- oder Videodateien eingebettet werden.

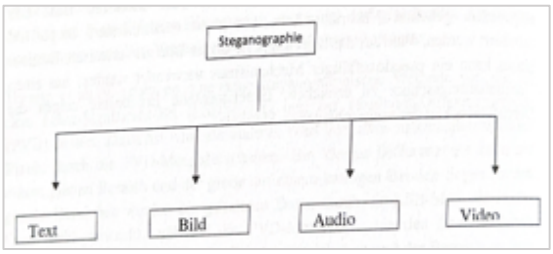

Abbildung 8 - Kategorien der für Steganografie verwendete Dateiformate

3.4.1 Audio-Steganografie

Es ist möglich Steganografie auf Audiodateien anzuwenden. Dabei wird eine Modifizierung von Audiodateien vorgenommen, sodass diese geheime Informationen erhalten. Die Schwierigkeit besteht dabei, die geheime Information zu entfernen, ohne eine Zerstörung des Originalsignals vorzunehmen. Zunutze macht sich die Audio-Steganografie das psychoakustische Maskierungsphänomen des menschlichen Gehörs (Human Auditory System - HAS), denn das HAS kann ein additives Zufallsrauschen wahrnehmen und ebenfalls Störungen in Tondateien erkennen. Ein Vorteil des HAS ist, dass diese einen großen Dynamikbereich hat, jedoch hat das HAS den Nachteil, dass diese lediglich einen kleinen Differenzbereich hat, was dazu führt, dass laute Töne dazu neigen, leise Töne zu überdecken. Techniken für die Audio-Steganografie sind Least Significant Bit-Kodierung (LSB-Kodierung), Phasen-Kodierung, Ausblendung des Echos und Spreizung des Spektrums. [23]

3.4.2 Video-Steganografie

Da Videostreams einen hohen Grad an zeitlicher sowie räumlicher Redundanz in der Darstellung aufweisen und eine allgegenwärtige Anwendung im täglichen Leben haben, eignen sich diese für das Verstecken von geheimen Informationen.

Die Video-Steganografie ist eine Erweiterung der Bild-Steganografie, denn eine Videodatei kann betrachtet werden als eine Sequenz von Bildern. Dennoch gibt es Unterschiede zwischen der Bild- und Video-Steganografie. Ferner ist ein Videoinhalt dynamisch, daher besteht eine geringe Wahrscheinlichkeit, dass verborgene Dateien darin entdeckt werden im Gegensatz zu statischen Bildern. Besonders von Vorteil ist, dass bei Videos eine große Menge an Daten im Inneren versteckt werden können und dass es sich dabei um einen bewegten Strom von Tönen und Bildern handelt. Dadurch bleiben kleine, kaum wahrnehmbare Verzerrungen im Video aufgrund des stetigen Informationsflusses von Menschen unentdeckt. Bei der Video-Steganografie kann eine Einordnung von Einbettungen in räumliche Bereiche, Frequenzbereiche oder format-spezifische Bereiche vorgenommen werden. [23]

3.4.3 Text-Steganografie

Wie dem Namen zu entnehmen ist, geht es bei der Text-Steganografie darum, einen Text in einen anderen Text so einzubetten, dass ein unbeteiligter Dritter keinen Rückschluss auf eine versteckte Botschaft ziehen kann. Dabei können verschiedene Techniken verwendet werden, wie z. B. Verwendung kontextfreier Grammatik zu einer Erzeugung von lesbaren Texten, Änderung der Formatierung eines bereits bestehenden Textes, Generierung von zufälligen Zeichenfolgen oder Änderung von Wörtern innerhalt eines Textes. Da jedoch die Text-Steganografie auf einem Mangel an redundanten Informationen beruht, gilt diese Technik als die Schwierigste, da im Gegensatz dazu diese Informationen in einer Audio-, Bild- oder Videodatei existent ist. Es können an Videos oder Bildern unerklärliche Änderungen vorgenommen werden, sobald bei Textdateien ein zusätzlicher Buchstabe oder eine zusätzliche Interpunktion vorgenommen wird, ist diese bereits für den unbeteiligten Dritten wahrnehmbar. Die Text-Steganografie kann klassifiziert werden in die formatbasierte Methode, linguistische Methode sowie die Zufalls- und Statistikgenerierungsmethode. [23]

3.4.4 Bild-Steganografie

Digitale Bilder haben verschiedene Bilddateiformate und für diese verschiedenen Formate gibt es unterschiedliche steganografische Algorithmen. Ziel der Bild-

Steganografie ist es die begrenzten Möglichkeiten des menschlichen visuellen Systems (HVS) auszunutzen. Grundsätzlich kann die Bild-Steganografie in zwei Kategorien unterteilt werden. Die eine Kategorie bezieht sich auf die Steganografie im Frequenzbereich und die andere Kategorie bezieht sich auf den Bild-Raum-Bereich. Im Frequenzbereich wird eine Transformierung vorgenommen. Dabei können Cover-Bilder mit einem sogenannten frequenzorientierten Mechanismus wie der diskreten Wavelet-Transformation (DWT) oder der diskreten Kosinustransformation (DCT) durchgeführt werden. Bei der raumbereichsbasierten Methode wird eine Modifikation der geheimen Daten im räumlichen Bereich des Deckblattes vorgenommen. Dies wird ebenfalls als Least Significant Bit (LSB) Methode bezeichnet. [23]

4 Einführung in die Grundlagen der Bildverarbeitung

In diesem Kapitel wird eine kurze Einführung in die Bildverarbeitung gegeben. Dabei werden die Begriffe Pixel definiert und die diversen Bildtypen erläutert. Daraufhin gibt es eine kurze Zusammenfassung über die verschiedenen Dateiformate.

4.1 Pixel

Grundsätzlich stellen Bilder eine flächenhafte Verteilung der Bestrahlungsstärke in einer Ebene dar. Computer können digitale Zahlenfelder verarbeiten, jedoch keine kontinuierlichen Bilder, daher müssen Bilder als zweidimensionale Punktfelder abgespeichert werden. Dieser Punkt wird als Pixel bezeichnet und repräsentiert die Bestrahlungsstärke an der zugehörigen Position des Gitters. Ergänzend dazu repräsentiert ein Pixel nicht nur einen Bildpunkt, sondern eine rechteckige Region, die sogenannte Elementarzelle des Gitters. [24] Um eine Bildposition für ein Bildelement zu bestimmen, wird ein Koordinatensystem benötigt. Die Werte eines Pixels sind immer binäre Wörter der Länge *k*. Daher kann ein Pixel grundsätzlich 2^k verschiedene Werte annehmen, wobei *k* oft als Tiefe beziehungsweise (bzw.) Bit-Tiefe beschrieben wird. Die Kodierung einzelner Pixelwerte in Bitmuster ist abhängig vom Bildtyp, wie z. B. RGB-Farbbild, Binärbild oder Grauwertbild. [25]

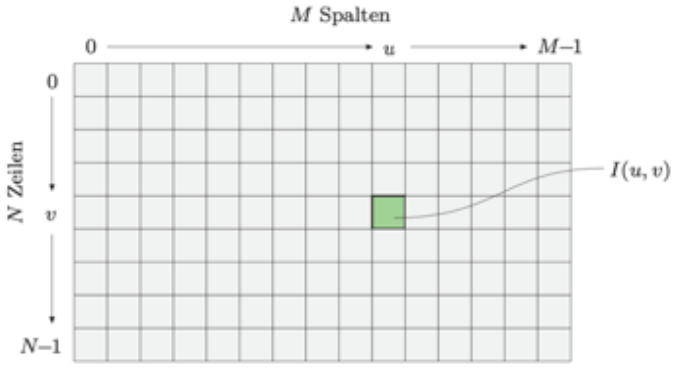

Abbildung 9 – Bildkoordinaten [25]

4.2 Bildtypen

In diesem Abschnitt wird eine kurze Einführung in die Bildtypen gegeben. Zu den verschiedenen Bildtypen gehören Binärbilder, Grauwertbilder sowie Farbbilder.

4.2.1 Binärbilder

Wie sich aus dem Namen ableiten kann, sehen Binärbilder nur zwei Pixelwerte vor, nämlich die Farben schwarz und weiß. Diese werden mit einem einzigen Bit (0/1) pro Pixel kodiert. Verwendet werden Binärbilder zur Archivierung von Dokumenten, für den Druck, für die Kodierung von Fax-Dokumenten und zur Darstellung von Strichgrafiken. [25]

4.2.2 Grauwertbilder

Grauwertbilder bestehen aus lediglich einem Kanal. Dieser beschreibt die Helligkeit, Dichte des Bildes oder die Intensität des Bildes und ein typisches Grauwertbild verwendet in der Regel k = 8 Bits (1 Byte) pro Pixel und kann somit die Intensitätswerte 0 – 255 abdecken. Dabei entspricht der Wert 0 die minimale Helligkeit (schwarz) und 255 der maximalen Helligkeit (weiß). [25]

4.2.3 Farbbilder

Viele Farbbilder bestehen aus je einer Komponente aus den Primärfarben rot, grün und blau (RGB). Pro Komponente sind diese Farbbilder mit 8 Bits kodiert und daher besteht jeder Pixel eines solchen Farbbilds aus 3 x 8 = 24 Bits. Der Wertebereich von jeder Farbkomponente ist 0 – 255. [25]

4.3 Dateiformate für Bilder

Bilder sind meist in Dateien gespeichert, darum sind Dateien eine essenzielle Grundlage für den Austausch, für die Archivierung und für die Speicherung von Bilddaten. Ebenso ist die Wahl des richtigen Dateiformats eine wichtige Entscheidung. In der heutigen Zeit stehen eine Reihe von standardisierten sowie passenden Dateiformaten zur Verfügung, welches z. B. den Austausch von Bilddaten erleichtert. Für die Auswahl des richtigen Dateiformats gibt es folgende

Kriterien: die Art der Bilder, der Speicherbedarf und der Kompression, der Anwendungsbereich sowie die Kompatibilität. Zur Art der Bilder zählen z. B. Schwarzweißbilder, Grauwertbilder, Farbfotos, farbige Spezialbilder oder Grafiken und Scans von Dokumenten. Beim Speicherbedarf und bei der Kompression ist zu überprüfen, ob die Dateigröße ein Problem und ob eine verlustbehaftete Kompression der Bilddaten zulässig ist. Der Anwendungsbereich soll bestimmen, für welche Zwecke die Bilddaten verwendet werden wie z. B. im Web, in der Computergrafik oder im Film. Zuletzt ist mit der Kompatibilität zu überprüfen, ob eine langfristige Lesbarkeit bzw. Archivierung der Bilddaten wichtig ist. [25]

4.3.1 Joint Photographic Experts Group – JPEG

Das Bildformat JPEG bestimmt ein Verfahren zur Kompression von Farb- und Grauwertbildern. Das Ziel der Entwicklung des JPEG-Standards war eine durchschnittliche Datenreduktion um den Faktor 1:16 herbeizuführen. Heutzutage ist JPEG das meistverwendete Darstellungsformat für Bilder und erlaubt je nach Anwendung eine Kompression von 24-Bit-Farbbildern bei einer akzeptablen Bildqualität im Bereich von 1 Bit pro Pixel (Kompressionsfaktor ist ungefähr 1:25). [25]

4.3.2 Graphics Interchange Format (GIF)

CompuServe hat im Jahre 1986 das Format GIF entwickelt, welches hauptsächlich für Internet-Anwendung galt. Bis heute sind GIFs sehr weit verbreitet. Konzipiert sind GIFs nur für Indexbilder, also Grauwert- und Farbbilder mit maximal 8-Bit-Indizes, und somit kein Vollfarbenformat und es werden Farbtabellen mit 256 Einträgen unterstützt. Allerdings sollte bei neueren Entwicklungen Portable Network Graphics (PNG) als moderneres Format bevorzugt werden. [25]

4.3.3 Portable Network Graphics (PNG)

Um das GIF-Format zu ersetzen und ein universelles Bildformat für verschiedene Internetanwendungen zu schaffen wurde PNG entwickelt. PNG unterstützt folgende drei Arten von Bildern: Indexbilder mit bis zu 256 Farben, Grauwertbilder

mit bis zu 16 Bits/Pixel sowie Vollfarbbilder mit bis zu 3 x 16 Bits/Pixel. Da keine verlustbehaftete Kompression bei PNG vorliegt, eignet sich dieses PNG-Format nicht als Ersatz für JPEG. Im Gegensatz dazu kann das PNG-Format das GIF-Format, außer bei Animationen, ersetzen. [25]

4.3.4 Tagged Image File Format (TIFF)

Konzipiert von Aldus, weiterentwickelt von Microsoft sowie von Adobe, ist TIFF ein flexibles und universelles Dateiformat, welches professionelle Ansprüche in vielen verschiedenen Anwendungsbereichen gerecht wird. Vollfarbbilder, Indexbilder und Grauwertbilder werden vom TIF-Format unterstützt. Zudem spezifiziert das TIF-Format diverse Farbräume sowie Kompressionsverfahren, somit ist es möglich, mehrere Formen und Varianten von Bildern in diversen Darstellungsformen und Größen in einer TIFF-Datei abzulegen. Daher kann dieses Format genutzt werden zur Archivierung von Dokumenten, in der digitalen Video- und Filmproduktion, in der Digitalfotografie, als universelles Austauschformat und in wissenschaftlichen Anwendungen. [25]

5 Theoretische Grundlage der LSB-Methode

Im nachfolgenden Kapitel erfolgt eine Einführung in die steganografische Methoden. Dabei wird erklärt, wieso die LSB-Methode für diese Bachelor-Thesis ausgewählt wurde und wie diese im Detail funktioniert. Dieser theoretische Abschnitt bildet ein Fundament für das Verständnis der anwendeten Methode. Im späteren praktischen Teil der Bachelor-Thesis wird eine Verschlüsselung mithilfe der LSB-Methode durchgeführt. Hierfür wird ein Code verwendet, der in Python geschrieben ist.

5.1 Einführung in die steganografische Methoden

Im Kapitel 3.4 wurden die steganografischen Methoden vorgestellt. Diese Methoden sind in der folgenden Tabelle abgebildet.

Methoden	Klassifizierung
Text-Steganografie	Formatbasierte Methode, Linguistische Methode
Bild-Steganografie	Raumbereichsbasiert (Least significant bit-Methode), Frequenzbereichsbasiert
Audio-Steganografie	LSB-Kodierung, Phasenkodierung, Ausblenden des Echos, Spreizung des Spektrums
Video-Steganografie	Räumlicher Bereich, Frequenzbereich, Format-Spezifisch

Tabelle 1 - Steganografische Methoden und ihre Klassifizierungen [23]

Die LSB-Methode gehört zu der raumbereichs-basierten Bild-Steganografie. Sie gehört zu den am häufigsten angewendeten Methoden der Steganografie und ist somit die meist genutzte steganografische Technik, deswegen wird in dieser Bachelor-Thesis die LSB-Methode erklärt und angewendet. [26] [27]

5.2 LSB-Methode

In diesem folgenden Kapitel wird das sogenannte LSB-Verfahren beschrieben.

Daraufhin wird ein Code beschrieben, der in der Programmiersprache Python programmiert wurde, welcher im späteren Abschnitt verwendet wird, um Bilder zu verschlüsseln.

5.2.1 Erläuterung der LSB-basierten Steganografie

Für die Untersuchung der steganografischen Werkzeuge wird als digitales Medium das Bild verwendet, da ein Bild, das am häufigsten verwendete Medium ist. Zudem ist die LSB-Methode die gebräuchlichste und beliebteste zur Einbettung von Schemata, denn die Informationen werden in den kleinsten Teilen des Bildes versteckt. Des Weiteren ist das LSB-Verfahren die einfachste sowie klassischste steganografische Technik, in welcher eine Geheiminformation in eine Teilmenge der LSB-Ebene in das Bild eingebettet, also verschlüsselt, wird. Diese Methode, nämlich eine Geheiminformation in einem Bild zu verstecken, ist überraschend effektiv, weil dabei die wenigsten signifikantesten Bits jedes Pixels in einem Bild genutzt werden, um die signifikantesten Bits eines anderen Bildes zu verstecken. [23] Ergänzend dazu können die Begriffe „Big-Endian“ und „Little-Endian“ eine deutlichere Erklärung liefern. Das übliche Modell einer Datei in der Computertechnik besteht aus einer einfachen Folge von Bytes, also 8 Bits. Dabei ist ein Byte die kleinste Einheit, welche aus einer Datei lesbar ist oder in welche geschrieben werden kann. Im Gegensatz dazu sind die den Bildelementen entsprechenden Datenobjekten in den Speichern oft größer als ein Byte, z. B. eine 32 Bits große int-Zahl mit 4 Bytes für ein RGB-Farbpixel. Die Anordnung der entsprechenden 4 Bytes in der zugehörigen Bilddatei können jedoch an verschiedenen Stellen erfolgen, nämlich am Anfang oder am Ende in einer einfachen Folge. Werden die 4 Bytes am Anfang angeordnet, so wird von dem Most Significant Bit (MSB) gesprochen. Wenn die 4 Bytes am Ende angeordnet werden, wird dies als LSB bezeichnet. Werden die einzelnen Bytes innerhalb einer Datei in der Reihenfolge von MSB nach LSB gespeichert, dann wird die Anordnung als „Big-Endian“ bezeichnet. Im umgekehrten Fall wird die Anordnung als „Little-Endian“ definiert. Bei der Anordnung mit dem „Big-Endian“ ist jedoch mit einer offensichtlichen Erkennung der Manipulation des Bildes zu rechnen, weil die ersten veränderten Bytes eine große Auswirkung auf das Bild haben. Stattdessen ist keine offensichtliche Wahrnehmung bei der Manipulation des

Bildes zu rechnen, da in der Regel nur das letzte bzw. vorletzte Bit geändert wird. [25] Im Wesentlichen basiert die LSB-Methode darauf, dass die am wenigsten signifikanten Bits jedes Pixels eines im Bild verwendet als zufälliges Rauschen zugeordnet werden kann und aufgrund dessen wird mit dieser Technik versucht redundante Teile eines Signals durch eine Geheiminformation zu ersetzen. Da diese LSB-Methode öfters mit Rasterbildern arbeitet, welche in einem kompressionslosen Format vorliegen, wie z. B. .bmp oder .gif, werden diese bevorzugt, da sie in einer verlustfreien Komprimierung vorliegen. Da die Bits der versteckten Nachricht direkt in das LSB des Bildpixels eingebettet wird, ist dies ein großer Vorteil der LSB-Methode und für das Verstecken der Nachricht. Ein weiterer Vorteil ist, dass die LSB-Modifikation nicht zu einer Bildverzerrung führt und daher das modifizierte steganografische Bild identisch mit dem Cover-Bild ausschaut. Zudem wird beim Überschreiben des LSB der nummerische Wert des Bytes nur sehr wenig verändert und kann daher von der menschlichen Wahrnehmung am geringsten erkannt werden. Jede Änderung des LSB eines Pixels führt zu einer Veränderung in der Intensität der Farbe, weil es 256 mögliche Intensitäten von jeder Primärfarbe gibt. [23]

5.2.2 Erläuterung der LSB-Methode anhand eines Beispiels

Um die LSB-Methode besser zu verstehen, wird diese Methode anhand eines Beispiels erläutert. Vorliegend haben wir ein Quadrat mit vier Farben: rot, schwarz, gelb und blau. Das Bild wird Quadrat1 bezeichnet.

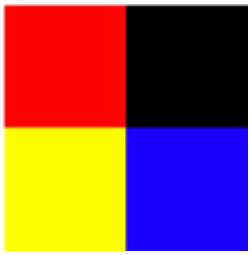

Abbildung 10 - Quadrat1

Anschließend werden die RGB-Werte sowie der Schwarz-Wert herausgeschrieben, dafür eignet sich folgender Link.

Abbildung 11 - Rot-Wert (255, 0, 0)

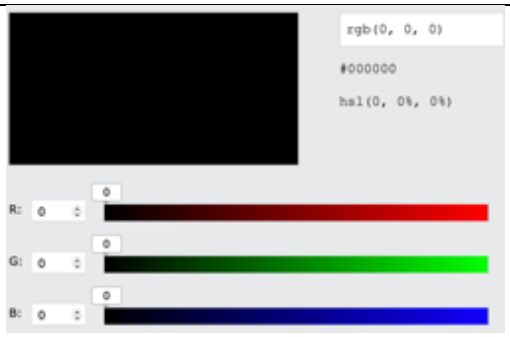

Abbildung 12 - Schwarz-Wert (0, 0, 0)

Abbildung 13 - Gelb-Wert (255, 255, 0)

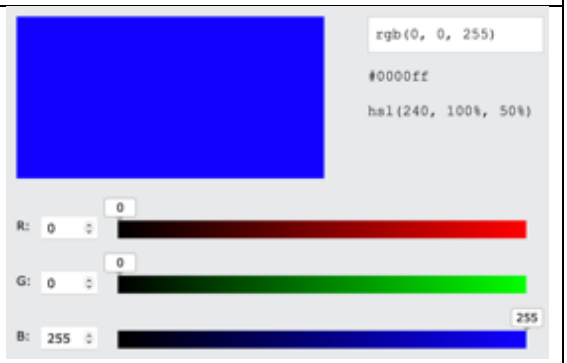

Abbildung 14 - Blau-Wert (0, 0, 255)

Tabelle 2 - Übersicht Farbwerte

In dem nächsten Bild werden die jeweiligen Farb-Werte untereinander aufgeschrieben.

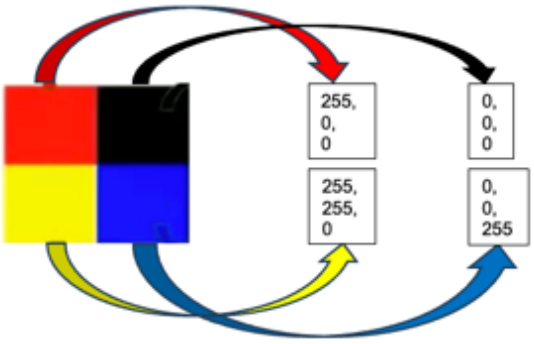

Abbildung 15 - ursprüngliche Farbwerte "Quadrat1"

Daraufhin werden die Farb-Werte in Binärzahlen umgewandelt. Dafür kann die American Standard Code for Information Interchange (ASCII) Tabelle verwendet werden, welche ein 7-Bit-Zeichensatz ist, der die Zeichen 0 – 127 beinhaltet. Da

in unserem Beispiel, z.B. 255, verwendet wird, wird der American National Standards Institute (ANSI) verwendet, da dieser für 8-Bit-Zeichensätze verwendet wird und zusätzlich zum unveränderten ASCII-Code weitere Zeichen von 128 – 255 nutzt. Daher folgt hier 255 = 11111111 [28].

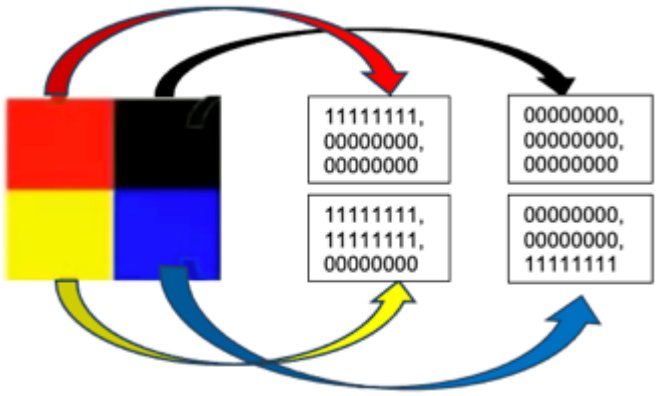

Abbildung 16 - ursprüngliche Farbwerte "Quadrat1" mit Binärzahlen

Im nächsten Schritt wird nun eine Geheimbotschaft festgelegt und diese mittels LSB-Methode integriert. Als Geheimbotschaft wird der Begriff „dog“ ausgewählt. Erneut wird mittels der ASCII-Tabelle der Binärwert für den Begriff „dog“ herausgeschrieben:

d = 01100100
o = 01101111
g = 01100111

Zusammengesetzt ergibt dies folgenden Wert: 011001000110111101100111. Anschließend werden die letzten zwei Zahlen (LSB-Prinzip) nacheinander mit den Werten des Geheimbotschaft ersetzt, d.h. rote Werte nacheinander, danach die schwarzen Werte et cetera (etc.). In einer Tabelle veranschaulicht, sind werden die Farb-Werte wie folgt verändert:

Farbe	**Ohne Geheimbotschaft**	**Mit Geheimbotschaft „dog“**
Rot	11111111, 00000000, 00000000	11111101, 00000010, 00000001
Schwarz	00000000, 00000000, 00000000	00000000, 00000001, 00000010
Gelb	11111111, 11111111, 00000000	11111111, 11111111, 00000001
Blau	00000000, 00000000, 11111111	00000010, 00000001, 11111111

Tabelle 3 - Übersicht Binärzahlen ohne und mit Geheimbotschaft

Bereits erkennbar ist, dass durch die Integration der Geheimbotschaft an die letzten beiden Binärstellen, eine kleine Veränderung in der Binärzahl erfolgt ist. Daraufhin werden die neuen Binärzahlen in die ursprünglichen Dezimalzahlen umgewandelt.

Rot:

11111101 = 253
00000010 = 02
00000001 = 01

Schwarz:

00000000 = 00
00000001 = 01
00000010 = 02

Gelb:

11111111 = 255
11111111 = 255
00000001 = 01

Blau:

00000010 = 02
00000001 = 01
11111111 = 255

Im nächsten Schritt werden die neuen RGB und Schwarz-Werte auf der Webseite eingetragen, um zu vergleichen, ob und wie sich die Farben verändert haben.

Abbildung 17 - Rot-Wert (253, 2, 1)

Abbildung 18 - Schwarz-Wert (0, 1, 2)

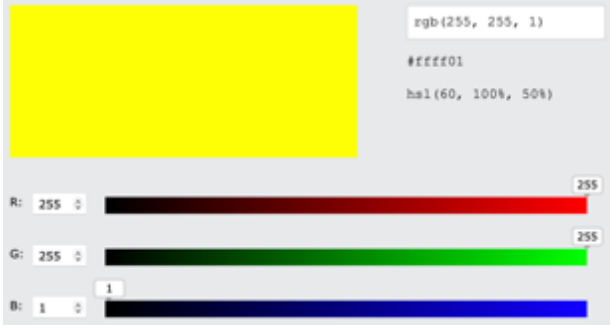

Abbildung 19 - Gelb-Wert (255, 255, 1)

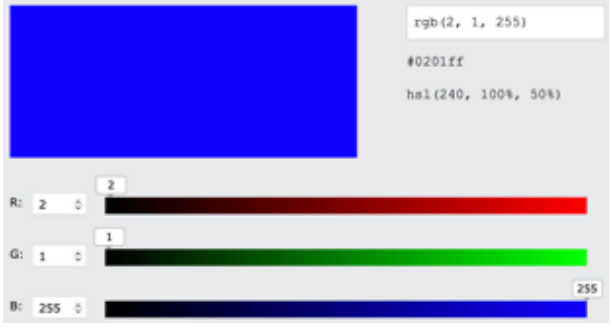

Abbildung 20 - Blau-Wert (2, 1, 255)

Tabelle 4 - Übersicht Farbwerte mit eingebetteter Geheimnachricht

Zusammengesetz sieht das ursprüngliche Quadrat1-Bild nun wie folgt aus:

Abbildung 21 - Quadrat1 mit eingebetteter Geheimbotschaft

Die beiden Bilder werden nebeneinandergelegt, um zu überprüfen, ob eine Farbunterscheidung wahrnehmbar ist.

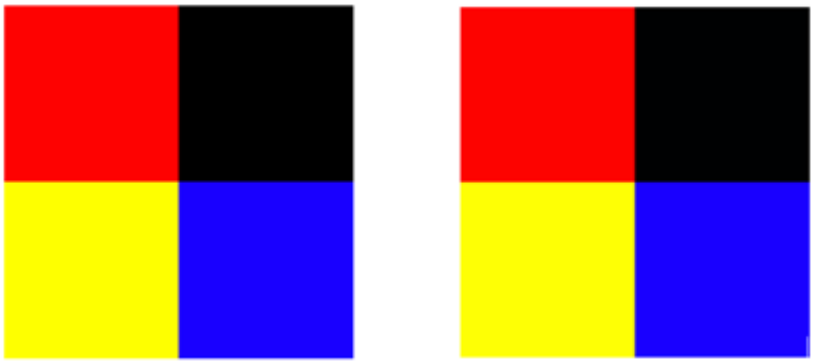

Abbildung 22 - Quadrat1 ohne Geheimbotschaft (links) und mit Geheimbotschaft (rechts)

Beim Betrachten der beiden Bilder ist offensichtlich kein Unterschied in den Farben zu erkennen. Das Ergebnis zeigt, dass das menschliche Auge die Veränderung an den letzten beiden Bitwerten nicht erkennen kann. Folglich eignet sich die LSB-Methode für das Verschlüsseln von Geheimbotschaften. [29]

5.3 Theoretische Erläuterung des Python-Codes

In diesem Abschnitt wird eine theoretische Erläuterung des Python-Codes präsentiert, der die praktische Anwendung der LSB-Methode ermöglicht.

```python
import numpy as np
from PIL import Image

message = ""

b_message = ''.join(["{:08b}".format(ord(x)) for x in message ])
b_message = [int(x) for x in b_message]

b_message_lenght = len(b_message)

with Image.open("") as img:
    width, height = img.size
    data = np.array(img)

data = np.reshape(data, width*height*3)

data[:b_message_lenght] = data[:b_message_lenght] & ~1 | b_message

data = np.reshape(data, (height, width, 3))

new_img = Image.fromarray(data)
new_img.save("")
new_img.show()
```

Abbildung 23 - Code für LSB-Verschlüsselung [30]

1. import numpy as np
Beginnend wird die NumPy-Bibliothek importiert, welche für die Manipulation von Arrays genutzt wird.

2. from PIL import Image
Anschließend wird die Image-Klasse aus der Python Imaging Library (PIL) Bibliothek importiert, welche für das Öffnen und das Manipulieren von Bildern genutzt wird.

3. message = „“
Bei message wird die zu verschlüsselnde Nachricht definiert.

4. b_message = ''.join(["{:08b}".format(ord(x)) for x in message])
Die Nachricht wird in eine binäre Darstellung konvertiert und dabei wird jeder Buchstabe in seine entsprechende 8-Bit-Binärzahll umgewandelt.

5. b_message = [int(x) for x in b_message]
Der Binärstring wird in eine Liste von Gesamtzahlen umgewandelt, in welcher jede einzelne Zahl ein Bit der verschlüsselten Botschaft repräsentiert.

6. b_message_lenght = len(b_message)
Hier wird die Länge der verschlüsselten Botschaft gespeichert.

7. with Image.open("") as img:
width, height = img.size
data = np.array(img)
Das Bild, welches verschlüsselt werden soll, wird geöffnet und die Breite sowie Höhe des geöffneten Bildes wird gespeichert. Zudem wird das Bild in ein NumPy-Array, welches die Pixelwerte des Bildes beinhaltet, umgewandelt.

8. data = np.reshape(data, width*height*3)
Das Pixelarray wird in eine eindimensionale Form abgeflacht und die Größe des Arrays wird auf width * height + 3 gesetzt, da jeder Pixel drei Farbkanäle besitzt,

rot, grün und blau (RGB).

9. data[:b_message_lenght] = data[:b_message_lenght] & ~1 | b_message
Die Least Significant Bits der Pixelwerte werden mit den Bits der zu verschlüsselnden Botschaft überschrieben.

10. data = np.reshape(data, (height, width, 3))
Daraufhin wird das Pixelarray wieder zurück in die ursprüngliche Form des Bildes zurückgesetzt.

11. new_img = Image.fromarray(data)
Aus dem manipulierten Pixelarray wird ein neues Bild erstellt.

12. new_img.save("")
Das neue Bild wird als neues Bildobjekt an einen gewünschten Dateiablageort gespeichert.

13. new_img.show()
Das neu erstellte Bild wird geöffnet und gezeigt.

Das Kapitel 5 widmet sich der Einführung in die steganografische Methoden und erklärt den Grund für die Auswahl der LSB-Methode für diese Bachelor-Thesis. Die LSB-Methode wurde anhand eines Beispiels vorgeführt sowie erläutert. Anschließend erfolgte eine theoretische Erläuterung des Python-Codes, mit dem die LSB-Methode durchgeführt wird. Dieser beschriebene Code wird im Kapitel 7 verwendet, um ein ausgewähltes Bild mit einer Geheimnachricht zu verschlüsseln. Daraufhin wird versucht, die Geheimbotschaft mit den vorgestellten steganografischen Werkzeugen zu entschlüsseln.

6 Analyse der steganografischen Werkzeuge

In diesem Kapitel werden steganografische Werkzeuge vorgestellt, die verwendet werden, um eine Nachricht zu verschlüsseln und anschließend zu entschlüsseln. Da eine Vielzahl steganografischer Werkzeuge im Umlauf sind, wird nur eine Auswahl an bestimmten Werkzeugen getroffen, da der Umfang dieser Arbeit begrenzt ist. Zudem wird betrachtet, ob diese Werkzeuge kostenlos oder kostenpflichtig sind und auf welchen Betriebssystemen die steganografischen Werkzeuge funktionieren. Des Weiteren wird die Installation der Werkzeuge erklärt sowie die Funktionen der Ver- und Entschlüsselung betrachtet. Nach der Vorstellung der steganografischen Werkzeuge wird ein Vergleich der Werkzeuge vorgenommen, indem die objektiven Gütekriterien vorgestellt und in einer Tabelle angewendet werden.

6.1 Verwendete Betriebssysteme

Für die Analyse der verschiedenen Werkzeuge für Steganografie werden drei unterschiedliche Betriebssysteme verwendet.

Für das Betriebssystem Linux wird eine virtuelle Maschine mit einer Kali Distribution installiert (Release 2023.1, Linux 6.1.0-kali7-amd64). Das verwendete Betriebssystem für Windows ist „Windows 10 Home“ und das genutzte Betriebssystem für MacOS ist „Ventura 13.4.1 (c)“.

6.2 Werkzeuge für die steganografische Untersuchung

Eine große Rolle bei der Entdeckung und Analyse von versteckten Informationen in digitalen Medien spielen steganografische Werkzeuge. Mit diesen Werkzeugen können Sicherheitsexperten, Forschern oder Analysten verdächtige Dateien überprüfen und potenziell verdeckte Inhalte aufspüren. Für die steganografische Untersuchung gibt es eine Vielzahl an Werkzeugen, die dafür genutzt werden können. Sowohl kostenpflichtige als auch kostenfreie Werkzeuge sind verfügbar. In den folgenden Abschnitten werden daher zuerst die

kostenlosen Werkzeuge und daraufhin die kostenpflichtigen Werkzeuge vorgestellt, die Ver- sowie Entschlüsselung angewendet und abschließend miteinander verglichen.

6.3 Kostenlose Werkzeuge

Dieser Abschnitt stellt einen Überblick über die Auswahl kostenfreier steganografischer Werkzeuge dar, die im Rahmen der Untersuchung verwendet werden. Zu den kostenfreien Werkzeugen gehören Steghide, Stegosuite, OutGuess, Zsteg, Tater, PicStealth, f2d, SSuite Picsel Encryption, Stegano und Stegify.

6.3.1 Steghide

Steghide ist ein Programm, mit dem sich verschiedene Bilder und Audiodateien verschlüsseln sowie entschlüsseln lassen. Unterstützt werden die Formate JPEG, Bitmap (BMP), Waveform Audio File Format (WAV) und Audio (AU) Dateien. Das Verfahren von Steghide gliedert sich in folgenden Schritten: Im ersten Schritt wird eine Datei ausgewählt, die eine Geheimbotschaft eingebettet bekommt. Zusätzlich wird eine zu verschlüsselnde Geheimbotschaft benötigt. Im nächsten Schritt wird ein Passwort für die Verschlüsselung der zu versteckenden Datei ausgesucht. Anschließend wird die Geheimnachricht in das zu verschlüsselnde Bild eingebettet. Steghide verwendet das Passwort, um die eingebettete Geheimnachricht zu verschlüsseln. Zuletzt wird für die Extrahierung der Geheimnachricht das Passwort verwendet. [31] Die Analyse für Steghide wurde auf dem Linux-Betriebssystem mit der virtuellen Maschine mit der Kali Distribution durchgeführt. Mit dem Befehl *apt install steghide* wird Steghide installiert. Über den Befehl *steghide --help* wird die aktuelle Version angezeigt und es werden alle Befehle gezeigt, die für das Einbetten als auch Extrahieren der Geheimbotschaft genutzt werden.

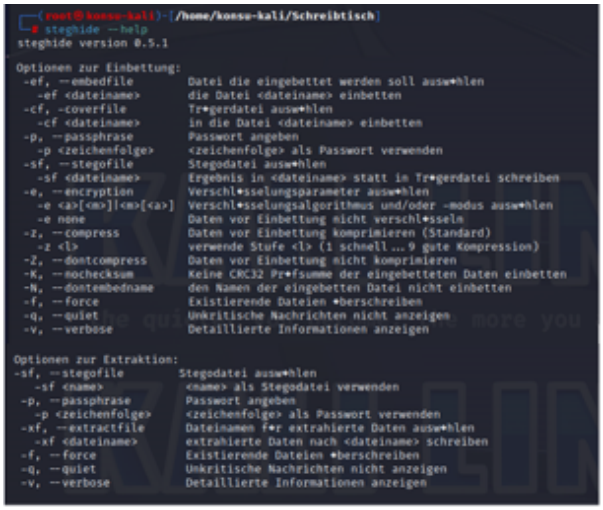

Abbildung 24 - Steghide - Übersicht der Befehle

Im nächsten Schritt wird probiert, eine Nachricht zu verschlüsseln und anschließend zu entschlüsseln. Als zu verschlüsselnde Nachricht wird „Pigs on bridge“ genommen. Das Foto, in welches die Nachricht eingebettet wird, ist emu.jpg.

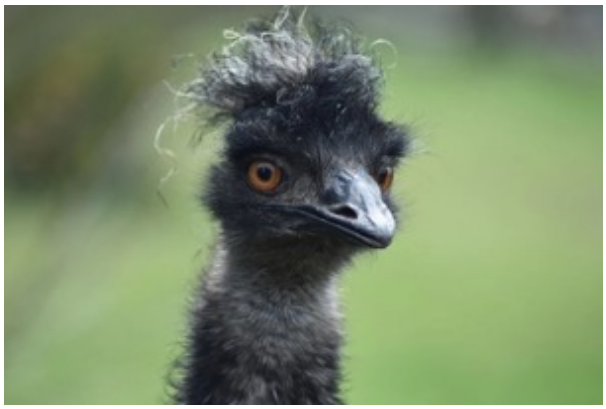

Abbildung 25 - emu.jpg [32]

Zuerst wird eine .txt-Datei mit der zu verschlüsselnden Nachricht eingebettet und daraufhin wird der Befehl *steghide embed -cf emu.jpg -ef secretinfo.txt -sf emunew.jpg* ausgeführt. Daraufhin muss zwei Mal ein Passwort eingegeben werden, dieses lautet: emchen. Mit der Auflistung *-ls* ist zu erkennen, dass emunew.jpg erstellt wurde.

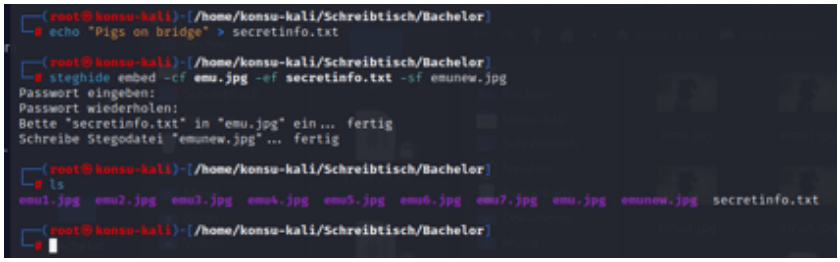

Abbildung 26 - Erstellung "emunew.jpg" mit Geheimbotschaft

Abbildung 27 - emu.jpg ohne Geheimnachricht (links) und emunew.jpg mit Geheimnachricht

Beim Betrachten beider Bilder ist keine Veränderung erkennbar. Mit dem Befehl *steghide info emunew.jpg* kann überprüft werden, ob eine Geheimbotschaft integriert wurde. Als Ergebnis wird bestätigt, dass die Datei „secretinfo.txt“ integriert und die Datei verschlüsselt sowie komprimiert ist.

Abbildung 28 - Überprüfung emunew.jpg auf Komprimierung

Im nächsten Schritt wird aus emunew.jpg die Geheimbotschaft extrahiert. Dazu wird zuerst die Datei „secretinfo.txt“ gelöscht und dann der Befehl *extract -sf emunew.jpg* ausgeführt. Es wird eine „secretinfo.txt“ neu geschrieben und die extrahierte Geheimbotschaft angezeigt, welche lautet „Pigs on bridge“.

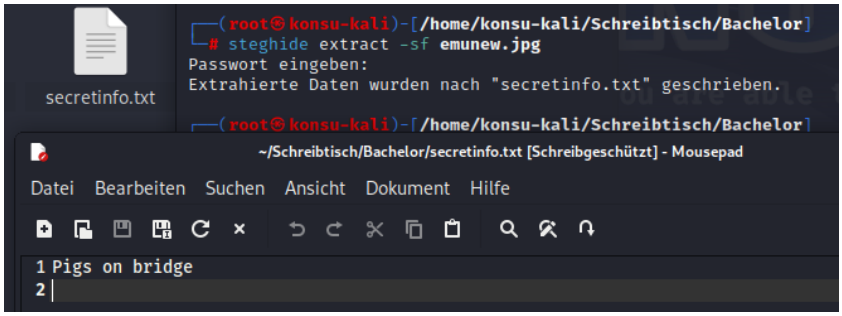

Abbildung 29 - Steghide - extract-Befehl

Mit dem Werkzeug Steghide lassen sich dementsprechend Bilder mit Geheimnachrichten mit einem zusätzlichen Passwort ver- sowie entschlüsseln.

6.3.2 Stegosuite

Stegosuite ist ein weiteres steganografisches Werkzeug, um Geheimbotschaften in Bilddateien zu verstecken. Im Gegensatz zu Steghide nutzt Stegosuite eine grafische Oberfläche (Graphical User Interface - GUI). Zusätzlich besteht die Möglichkeit mit AES zu verschlüsseln. Stegosuite unterstützt die folgenden Dateiformate GIF, JPG, BMP sowie PNG. Das Verfahren gliedert sich in folgende Schritte auf: zum Anfang wird in der grafischen Benutzeroberfläche ein Bild ausgewählt, welches die Geheimbotschaft integriert bekommen soll. Daraufhin wird die Geheimbotschaft eingetragen und optional kann ein Passwort ausgewählt werden, um die Sicherheit zu erhöhen. Das neue Bild wird erstellt und über Stegosuite kann abschließend die Geheimnachricht extrahiert werden. [33] Die Analyse für Stegosuite wurde auf dem Linux-Betriebssystem mit der virtuellen Maschine mit der Kali Distribution durchgeführt.

Zuerst wird Stegosuite mit dem Befehl *apt install stegosuite* installiert. Über *stegosuite -h* werden die benötigten Befehle angezeigt.

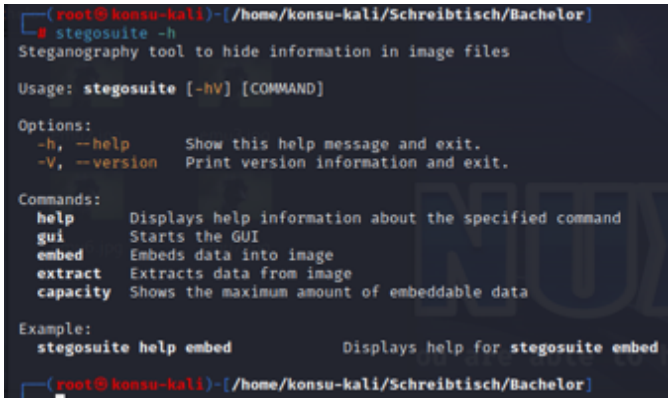

Abbildung 30 - Stegosuite - Übersicht der Befehle

Um die GUI zu nutzen, wird *stegosuite gui* aufgerufen und das Bild emu1.jpg ausgewählt.

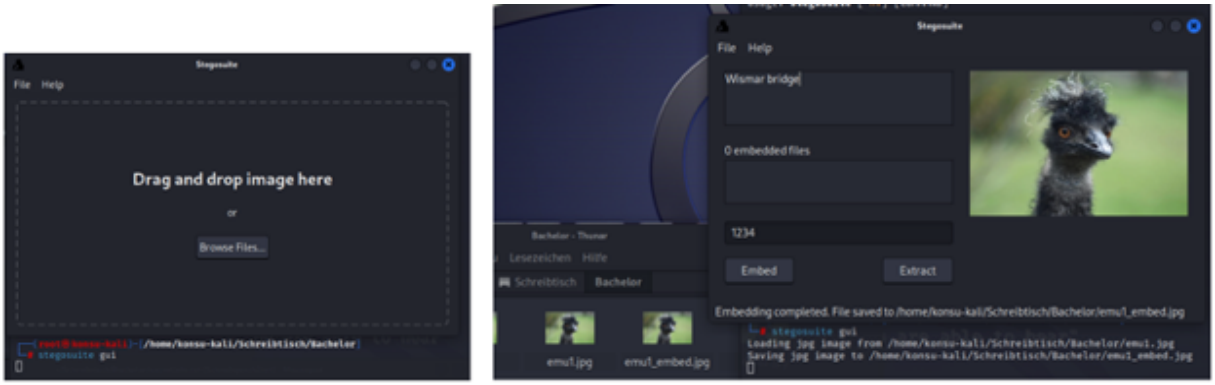

Abbildung 31 - Stegosuite GUI (links) und Einbettung Geheimbotschaft mit Passwort (rechts)

Wie bereits für Steghide muss ebenfalls ein Passwort ausgewählt werden für das Verschlüsseln der Botschaft. Die Geheimbotschaft lautet „Wismar bridge“ und das Password „1234“. Es wird das neue Bild „emu1_embed.jpg“ erstellt.

Abbildung 32 - emu1.jpg ohne Geheimbotschaft (links) und emu1_embed.jpg mit Geheimbotschaft (rechts)

Wie bereits im Beispiel bei Steghide sind die beiden Bilder nicht oberflächlich gesehen anders. Um die Geheimbotschaft zu extrahieren, wird im Terminal der Befehl *stegosuite extract -k 1234 emu1_embed.jpg* ausgeführt. Die extrahierte Botschaft lautet „Wismar bridge“ und ist die korrekte verschlüsselte Botschaft.

```
┌──(root㉿konsu-kali)-[/home/konsu-kali/Schreibtisch/Bachelor]
└─# stegosuite extract -k 1234 emu1_embed.jpg
Loading jpg image from /home/konsu-kali/Schreibtisch/Bachelor/emu1_embed.jpg
Extracting data...
Extracting completed
Extracted message: Wismar bridge

┌──(root㉿konsu-kali)-[/home/konsu-kali/Schreibtisch/Bachelor]
```

Abbildung 33 - Extraktion der Geheimbotschaft

6.3.3 OutGuess

OutGuess ist ein Werkzeug für Steganografie und unterstützt die Formate JPEG, Portable Pixel Map (PPM) sowie Portable Any Map (PNM). Das Verfahren beinhaltet die Schritte der Trägerauswahl, der Kapazität, der Verschlüsselung, der Wahl eines Passworts und der Extraktion der Daten. Bei der Trägerauswahl wird das Medium ausgewählt, welches mit der Geheimbotschaft verschlüsselt wird. OutGuess bietet eine hohe Einbettungskapazität an, mit welcher eine große Datenmenge in einem Bild versteckt wird. Outguess biete im Gegensatz zu anderen Werkzeugen keine Verschlüsselung für die eingebetteten Daten an, sondern versteckt die Daten lediglich durch die Einbettungstechnik. Beim Verwenden eines Passworts wird die eingebettete Datei verschlüsselt. Zum

Schluss kann OutGuess die versteckte Botschaft, ggf. mit einem Passwort, extrahieren. [34] [35] Die Analyse für OutGuess wurde auf dem Linux-Betriebssystem mit der virtuellen Maschine mit der Kali Distribution durchgeführt. Installiert wird OutGuess mit dem Befehl *sudo apt install outguess*. Über *outguess -h* wird in die Übersicht der Befehle angezeigt.

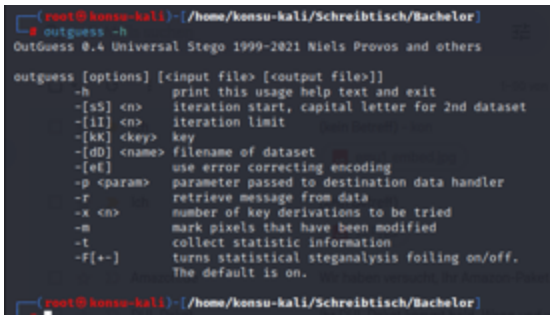

Abbildung 34 - OutGuess - Übersicht der Befehle

Im ersten Schritt wird eine neue Geheimbotschaft „Pigs in Wismar“ erstellt mit dem Befehl *echo „Pigs in Wismar“ > information.txt*. Die Datei „information.txt“ wird in das Bild „emu2.jpg“ integriert mit dem Befehl *outguess -d information.txt emu2.jpg emu2_new.jpg*. Das Bild emu2_new.jpg wird erfolgreich erstellt.

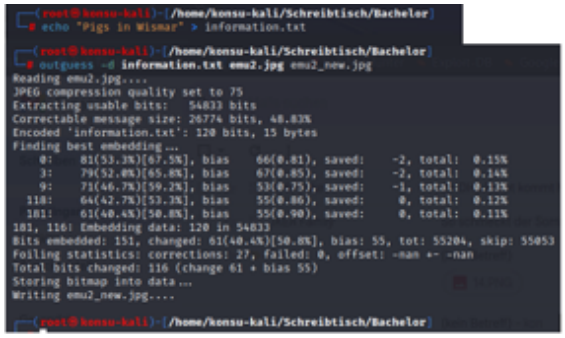

Abbildung 35 - Einbettung der Geheimnachricht "information.txt"

Daraufhin werden die Bilder „emu2.jpg“ und „emu2_new.jpg“ miteinander vergleichen. Wie in den beiden oberen Beispielen ist keine Auffälligkeit erkennbar.

Abbildung 36 - emu2.jpg ohne Geheimbotschaft (links) und emu2_new.jpg mit Geheimbotschaft (rechts)

Um die Nachricht zu extrahieren, wird der Befehl *outguess -r emu2_new.jpg whatsinthetext.txt* ausgeführt. Mit dem Befehl *cat whatsinthetext.txt* erhalten wir die richtig entschlüsselte Nachricht „Pigs in Wismar“.

```
┌──(root㉿konsu-kali)-[/home/konsu-kali/Schreibtisch/Bachelor]
└─# outguess -r emu2_new.jpg whatsinthetext.txt
Reading emu2_new.jpg....
Extracting usable bits:   54833 bits
Steg retrieve: seed: 181, len: 15

┌──(root㉿konsu-kali)-[/home/konsu-kali/Schreibtisch/Bachelor]
└─# cat whatsinthetext.txt
Pigs in Wismar

┌──(root㉿konsu-kali)-[/home/konsu-kali/Schreibtisch/Bachelor]
└─# cat information.txt
Pigs in Wismar

┌──(root㉿konsu-kali)-[/home/konsu-kali/Schreibtisch/Bachelor]
```

Abbildung 37 - Extraktion der Geheimnachricht

Bei OutGuess besteht die Möglichkeit ein Bild ebenfalls mit einem Passwort, in diesem Beispiel wird das Passwort „1234“ genommen, zu versehen. Der Befehl dafür lautet *outguess -k 1234 -d emu2.jp emu2_new_pw.jpg*. Der Befehl für das Extrahieren der Geheimbotschaft lautet *outguess -r -k 1234 emu2_new_pw.jpg news.txt.*

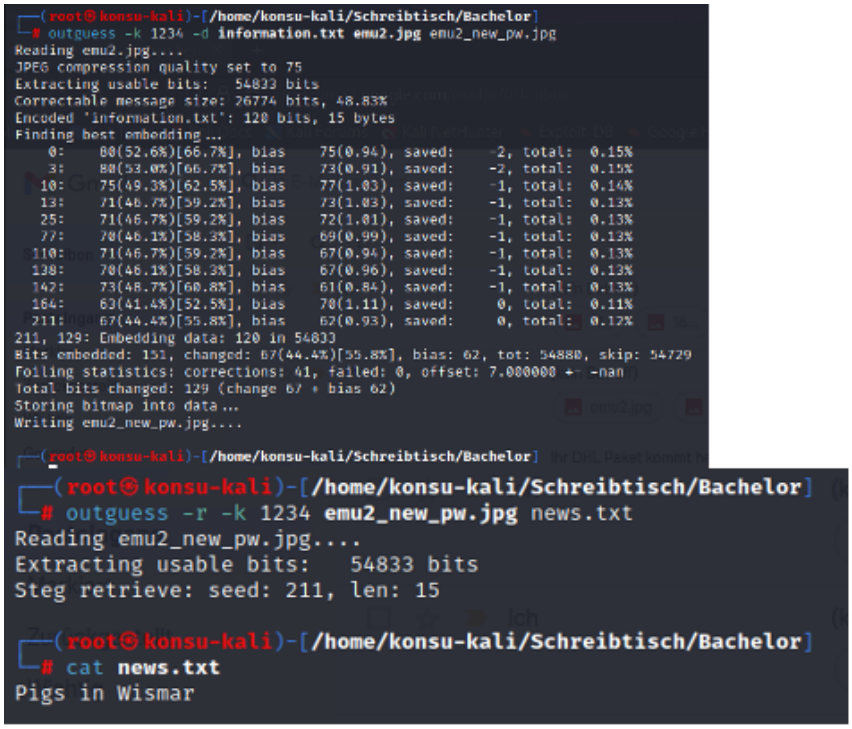

Abbildung 38 - Einbettung der Geheimbotschaft mit einem Passwort

6.3.4 Zsteg

Als nächstes Werkzeug wird Zsteg angeschaut. Zsteg ermöglicht es eine LSB-Verschlüsselung zu erkennen, wenn das Bildformat ein PNG oder BMP ist. Das Verfahren von Zsteg kann wie folgt beschrieben werden. Es wird zuerst ein Bild ausgewählt, danach wird eine Bildanalyse durchgeführt und überprüft, ob eine Datenmanipulation stattgefunden hat. Zsteg kann verschiedene Einbettungstechniken wie z. B. die LSB-Methode und die Palettenbasierte-Technik identifizieren. Sobald Zsteg eine versteckte Datei erkennt, wird diese extrahiert. [36] [37] Die Analyse für Zsteg wurde auf dem Linux-Betriebssystem mit der virtuellen Maschine mit der Kali Distribution durchgeführt.

Die Installation wird über die Befehle *apt install ruby* sowie *gem install zsteg* durchgeführt. Der Befehl *zsteg* wird ausgeführt, um einen Überblick über das Werkzeug zu erhalten.

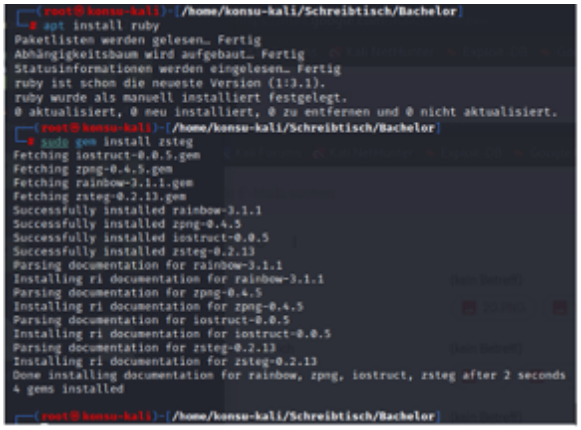

Abbildung 39 - Zsteg - Installation

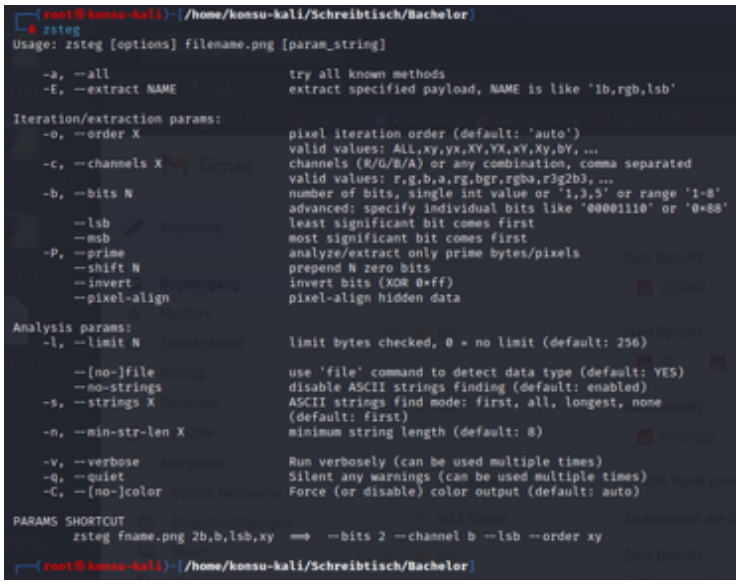

Abbildung 40 - Übersicht Befehle

Wie bereits in der Beschreibung zu entnehmen, lässt sich mit Zsteg keine Nachricht verschlüsseln, sondern nur eine Nachricht entschlüsseln, die mittels einer LSB-Methode verschlüsselt wurde. Folgende werden zwei Bilder analysiert. Zum einen das Bild pig.png und zum anderen das Bild wismar2_encoded.png. Die Einbettung der Geheimnachricht erfolgte über den Code der auf der Seite www.thepythoncode.com beschrieben ist. [38] Die verschlüsselte Nachricht lautet „Hello world“.

```
(StegoSession) (base) konsuelabednarek@Konsuelas-MBP opencv % python stegano.py -e wismar2.png -t "Hello world"
[*] Maximum bytes to encode: 9144576
[*] Data size: 11
[*] Encoding data...
[+] Saved encoded image.
(StegoSession) (base) konsuelabednarek@Konsuelas-MBP opencv %
```

Abbildung 41 - Verschlüsselung "wismar2.png"

Daraufhin wird im Terminal der Befehl *zsteg pig.png* sowie *zsteg wismar2_encoded.png* aufgerufen.

Abbildung 42 - pig.png ohne Geheimbotschaft (links) und wismar2_encoded.png mit Geheimbotschaft (rechts)

```
┌──(root㉿konsu-kali)-[/home/konsu-kali/Schreibtisch/Bachelor]
└─# zsteg pig.png
meta XML:com.adobe.xmp .. text: "<x:xmpmeta xmlns:x=\"adobe:ns:meta/\" x:xmptk=\"XMP Core 6.0.0\">\n
p://www.w3.org/1999/02/22-rdf-syntax-ns#\">\n      <rdf:Description rdf:about=\"\"\n            xmlns
om/exif/1.0/\">\n         <exif:PixelYDimension>858</exif:PixelYDimension>\n         <exif:PixelXDime
mension>\n         <exif:UserComment>Screenshot</exif:UserComment>\n      </rdf:Description>\n   </rd
imagedata       .. file: little endian ispell 3.0 hash file, and 255 string characters
b1,g,msb,xy     .. text: "*GbH{~ R"
b2,rgb,msb,xy   .. file: VISX image file
b2,bgr,msb,xy   .. file: VISX image file
b3,abgr,msb,xy  .. text: "q?u.oxgGz_'yT?"
b4,r,lsb,xy     .. text: "ffeUffgcVt8"
b4,g,lsb,xy     .. text: "foYqj(tWeTEUDDEV1%"
b4,b,lsb,xy     .. text: "XfhuTEvvTQO"
b4,b,msb,xy     .. text: "{{wwwww{w;"
b4,rgb,lsb,xy   .. text: "gVefHdvUeUTTEEdvFdvVeWdcS"
b4,bgr,lsb,xy   .. text: "WehFteVUdUDUEtfFtfVUdgSa"
b4,rgba,lsb,xy  .. text: "e_e_UOTOT_d"

┌──(konsu-kali㉿konsu-kali)-[~/Downloads]
└─$ zsteg wismar2_encoded.png
imagedata       .. file: MIPSEB ECOFF executable not stripped - version 1.2
b1,bgr,lsb,xy   .. text: "Hello world"
b4,r,lsb,xy     .. text: "2#C#3333#DC3EDDC\"VTFUCETE14323$eTC5S44B4C"
b4,g,lsb,xy     .. file: PGP Secret Sub-key -
b4,bgr,msb,xy   .. file: PGP Secret Sub-key -

┌──(konsu-kali㉿konsu-kali)-[~/Downloads]
```

Abbildung 43 - Vergleich beider Bilder auf Geheimbotschaft

Um eine detaillierte Ausführung zu erhalten kann der Befehl *zsteg -v pig.png* und *zsteg -v wismar-secret.png* verwendet werden. Dabei wird der Verbose-Modus aktiviert, welche mehr Informationen über die versteckte Datei anzeigt.

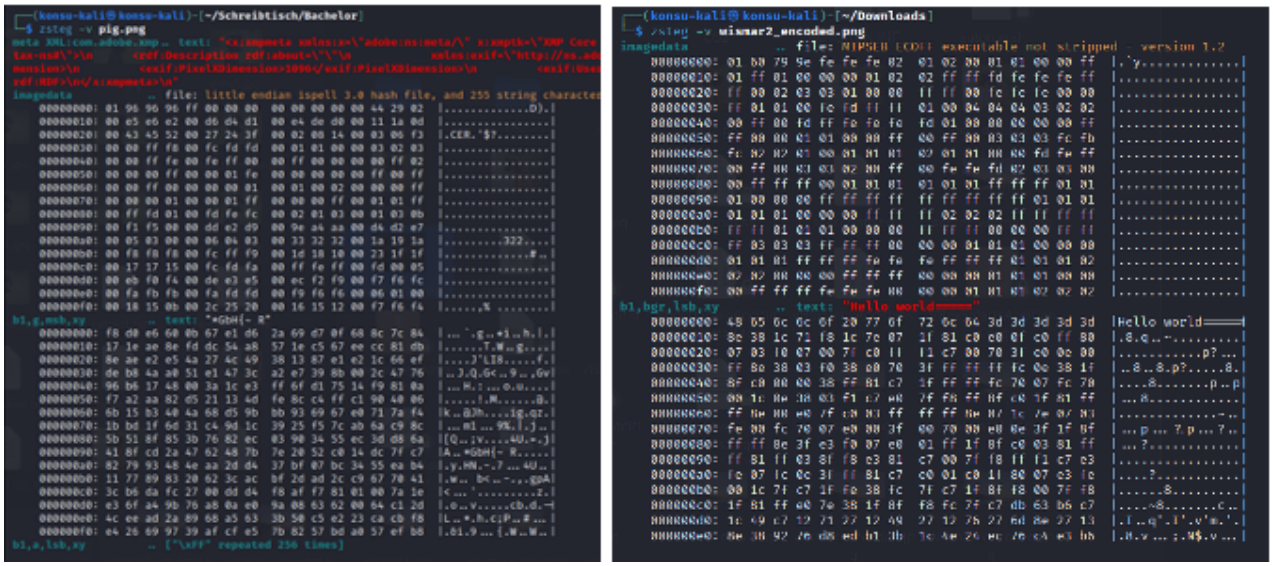

Abbildung 44 - Verbose-Modus

Das Ergebnis bestätigt, dass sich hinter dem Bild wismar-secret.png die Geheimbotschaft „Hello World!“ befindet. Wichtig ist zu erwähnen, dass bei Zsteg keine Möglichkeit besteht, die Bilder selbst zu verschlüsseln. Bei allen anderen Werkzeugen werden Bilder mit dem jeweiligen Werkzeug ver- und entschlüsselt.

6.3.5 Tater

Tater ist eine App die im Windows-Store heruntergeladen werden kann. Tater ermöglicht versteckte Botschaften in Bildern einzubetten und zu extrahieren. Welche Bildformate unterstützt werden, lässt sich aus der Beschreibung nicht herausfinden. Das Verfahren funktioniert wie folgt: Die Anwendung wird aufgerufen und ein Bild ausgewählt, welches verschlüsselt wird. Bereits beim Auswählen eines Bildes untersucht Tater, ob bereits ein verschlüsseltes Bild vorliegt. Wenn Tater erkennt, dass das Bild nicht manipuliert ist, wird eine geheime Botschaft eingetragen und als ein neues Bild gespeichert. Anschließend kann das verschlüsselte Bild extrahiert werden. [39] Die Analyse von Tater wurde auf dem Betriebssystem Windows 10 Home durchgeführt.

Abbildung 45 - Tater - Windows Store-App

Beim Aufrufen der Tater-App öffnet sich direkt eine grafische Oberfläche, in welcher das Bild hochgeladen werden kann. Das Bild kann über die integrierte Suche oder per Drag-and-Drop geöffnet werden.

Abbildung 46 - Übersicht der Anwendung

Das Bild „emu.jpg“ wird ausgewählt und hinzugefügt. Sobald das Bild ausgewählt wurde, überprüft das Werkzeug direkt, ob eine Geheimbotschaft integriert wurde.

Abbildung 47 - Direkte Überprüfung eines Bildes auf Geheimbotschaft

Da in „emu.jpg“ keine Geheimnachricht vorhanden ist, wird das Bild aufgerufen. Im Textfeld wird eine Nachricht „Hello Wismar!“ geschrieben und eine neue Datei mit dem Namen „emu-secret.jpg“ als PNG-Datei gespeichert.

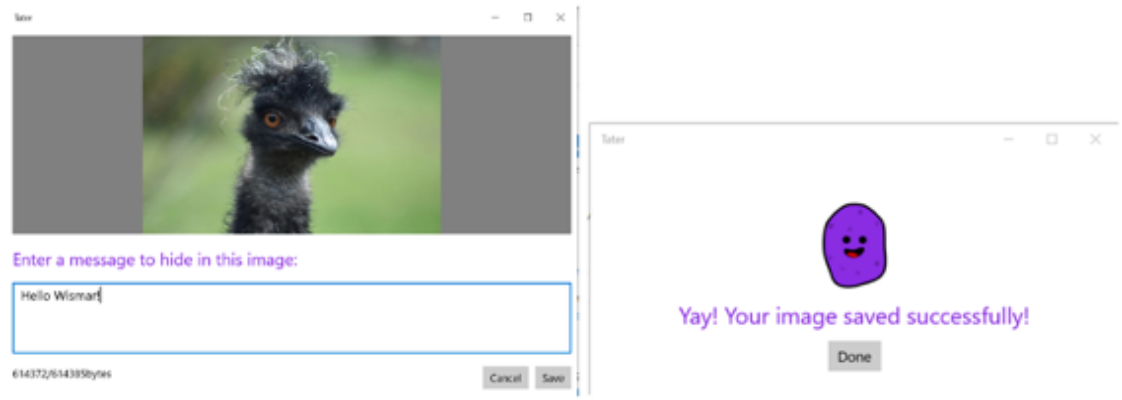

Abbildung 48 - Einbettung Geheimbotschaft

Nach der Erstellung des neuen Bildes wird das Bild geöffnet, um nachzuprüfen, ob Tater erkennt, dass sich in dem Bild eine Geheimbotschaft befindet. Das Ergebnis zeigt, dass die Geheimbotschaft erfolgreich herausgelesen wird.

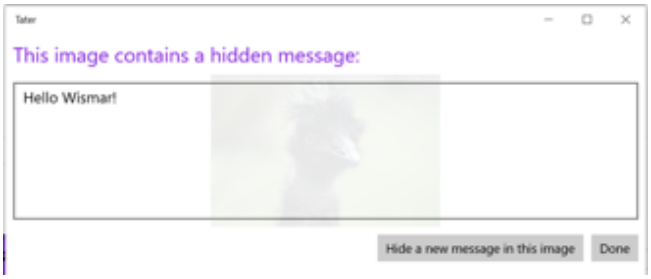

Abbildung 49 - Extraktion der Geheimbotschaft (JPG)

Im nächsten Schritt wird das Bild „welpe.png“ ausgewählt, um zu testen, ob das Format PNG ebenfalls akzeptiert. Sowohl ein neues Bild „welpe-secret.png“ und die anschließende Extraktion der Geheimbotschaft ist möglich. Daher unterstützt diese App sowohl Bilder im JPG als auch PNG-Format.

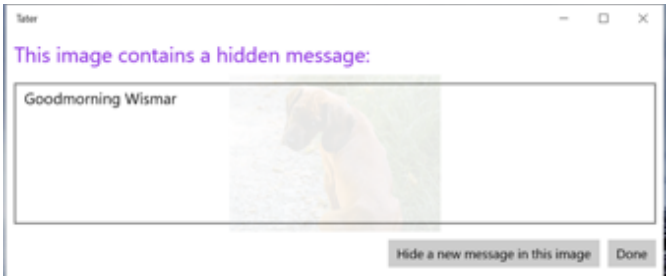

Abbildung 50 - Extraktion der Geheimbotschaft (PNG)

6.3.6 PicStealth

Das nächste Werkzeug heißt PicStealth und wird wie Tater über den Windows App Store angeboten. PicStealth bietet die Möglichkeit an, Bilder zu ver- und entschlüsseln. Die Bilder werden beim Verschlüsseln als PNG-Format ausgegeben. Das Verfahren von PicStealth funktioniert im ersten Schritt mit dem Hochladen eines Bildes in die Anwendung. Daraufhin wird eine zu verschlüsselnde Nachricht eingebettet und ein neues Bild erstellt. Des Weiteren wird in der Anwendung eine Nachricht aus einem verschlüsselten Bild extrahiert. [40] Die Analyse von Tater wurde auf dem Betriebssystem Windows 10 Home durchgeführt.

Abbildung 51 - PicStealth - Windows Store-App

Beim Aufrufen der App sollte der Vollbildmodus aktiviert werden, damit die gesamte App angezeigt werden kann. Nützlich ist es mit der rechten Maustaste auf die Oberfläche der App zu klicken, damit die Werkzeugleiste geöffnet wird, worüber die Bilder hochgeladen sowie gespeichert werden können.

Abbildung 52 - PicStealth - Übersicht

Als zu verschlüsselndes Bild wird „emu.jpg“ genommen und mit der Geheimbotschaft „Meet me in Wismar“ verschlüsselt.

Abbildung 53 - Verschlüsselung der Geheimbotschaft

Das neu gespeicherte Bild heißt „Emu_info“ und befindet sich im PNG-Format. Diese Bild wird nun hochgeladen und „Extract text from picture“ angewendet. Im Textfeld erscheint die Nachricht „Meet me in Wismar“.

Abbildung 54 - Entschlüsselung der Geheimbotschaft (JPG)

Dieser Test wird mit dem Bild „welpe.png“ ausprobiert, um zu bestätigen, dass dieses Werkzeug für PNG-Formate angewendet werden kann. Wir bereits beim JPG-Format wird das PNG-Format unterstützt und die korrekt verschlüsselte Geheimnachricht entschlüsselt.

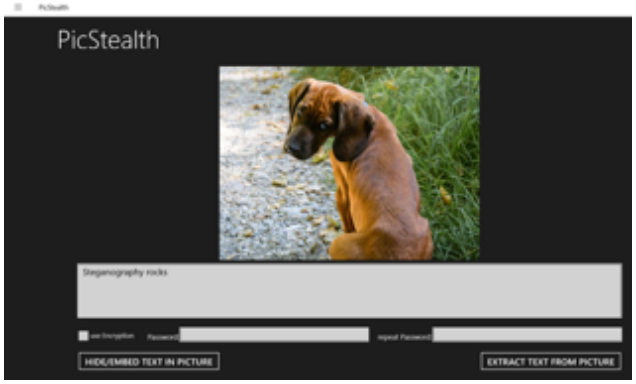

Abbildung 55 - Entschlüsselung der Geheimbotschaft (PNG)

Eine weitere Möglichkeit bei PicStealth ist die Möglichkeit das Bild mit einem Passwort zu verschlüsseln. Zum Verschlüsseln wird das Bild „welpe.png“ ausgewählt und das Passwort auf „1234“ festgelegt (der Haken wird dabei auf „use encrytion“ gesetzt).

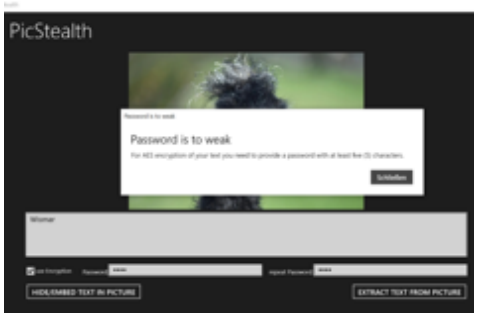

Abbildung 56 - Hinweis Passwortlänge

Ein Hinweis erscheint, dass das Passwort zu kurz ist und mindestens 5 Zeichen benötigt, da eine AES-Verschlüsselung beinhaltet. Daher wird das Passwort auf „123456“ erhöht. Das neue Bild „emu_password.png“ wird erstellt. Anschließend wird dieses Bild hochgeladen und versucht das Bild zu entschlüsseln, ohne das Passwort zu setzen. Das Ergebnis liefert eine Zeichenkette im Textfeld, welche keinen Rückschluss auf die Geheimnachricht gibt. Daraufhin wird das korrekte Passwort gesetzt und die Geheimbotschaft „Wismar“ wird extrahiert.

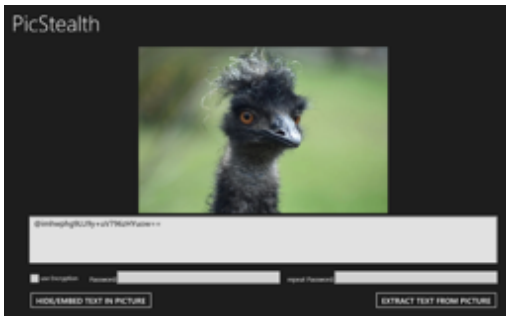

Abbildung 57 - Versuchte Entschlüsselung ohne Eingabe des Passworts

Abbildung 58 - Entschlüsselung mit Passworteingabe

6.3.7 f2d

Daraufhin wird das Werkzeug „f2d“ vorgestellt. F2d kann im Store für Windows-App kostenlos heruntergeladen werden. Dieses Werkzeug ermöglicht Texte in Bilder zu ver- sowie entschlüsseln. Der Text wird mit einem AES-256 Algorithmus verschlüsselt und die neu gespeicherten Bilder werden als PNG-Format gespeichert. Das Verfahren von f2d funktioniert, indem ein Bild ausgesucht, eine geheime Nachricht eingetragen und ein Passwort eingetragen wird. Danach wird ein neues Bild erstellt, welches die geheime Botschaft eingebettet hat. Zusätzlich kann das Passwort extrahiert werden im Anschluss. [41] Die Analyse von Tater wurde auf dem Betriebssystem Windows 10 Home durchgeführt.

Abbildung 59 - f2d - Windows Store-App

Das Werkzeug f2d wird aufgerufen und es öffnet sich eine Oberfläche, auf welcher eine Alternative angeboten wird. Zum einen kann ein neues Bild verschlüsselt werden, „New cipher image“, und zum anderen kann ein verschlüsseltes Bild geöffnet und entschlüsselt werden, „Open cipher image“.

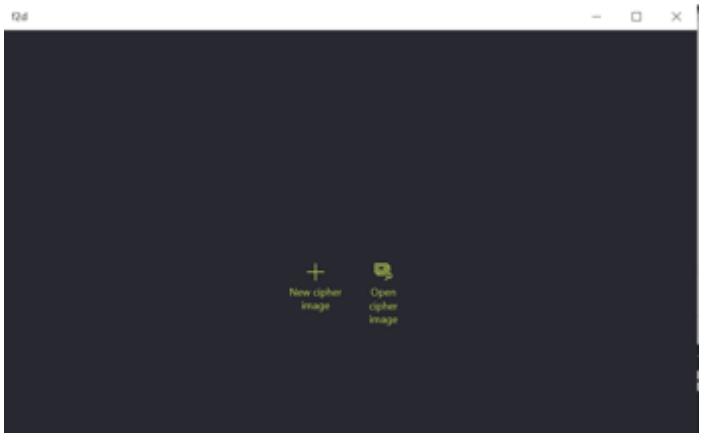

Abbildung 60 - Übersicht der Funktionen von f2d

Zuerst wird versucht ein Bild mit einer Geheimbotschaft zu verschlüsseln. Zu

erkennen ist, dass sowohl ein Passwort als auch eine Botschaft erforderlich sind.

Abbildung 61 - Verschlüsselung mit Passwort und Text

Das Foto, welches verschlüsselt werden soll, heißt „puppy.png“. Bilder im JPG-Format können nicht geöffnet werden, daher werden nur PNG-Formate unterstützt. Nach Eingabe der Textnachricht „Pigs in Wismar“ und ohne Eingabe eines Passworts wird direkt ein Hinweis ausgespielt, dass ein Passwort verwendet werden muss. Daher wird das Passwort auf „123456“ festgelegt.

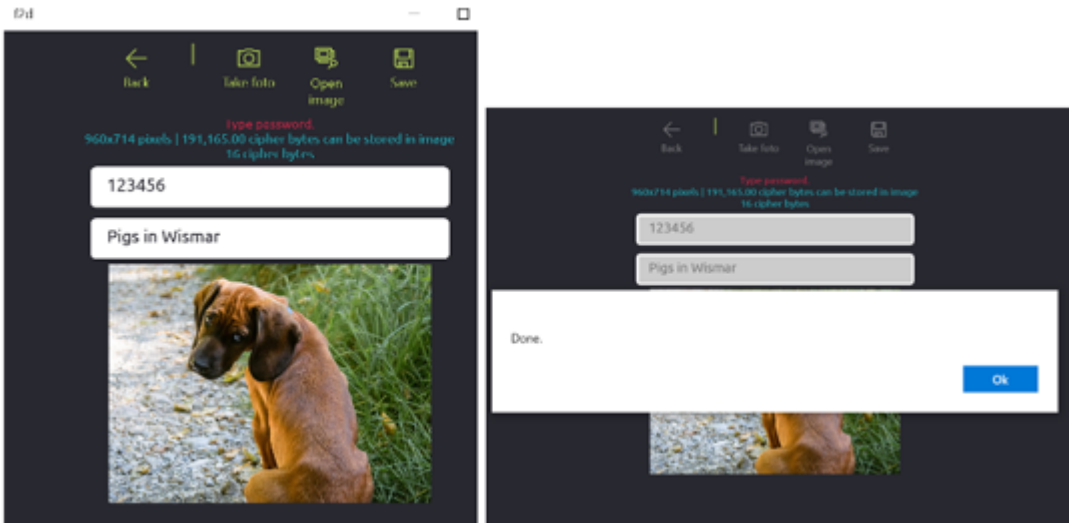

Abbildung 62 - Verschlüsselung einer Geheimbotschaft

Das neue Bild „puppy_pw.png“ wird, wie bereits erwähnt, in einem PNG-Format abgespeichert. Anschließend wird versucht, das verschlüsselte Bild zu entschlüsseln. Daher wird der Bereich „Open cipher image“ betreten und das Bild „puppy_pw.png“ hochgeladen. Nach Drücken der Funktionstaste „decrypt“ sowie der Eingabe des Passworts wird der Text „Pigs in Wismar“ korrekt angezeigt.

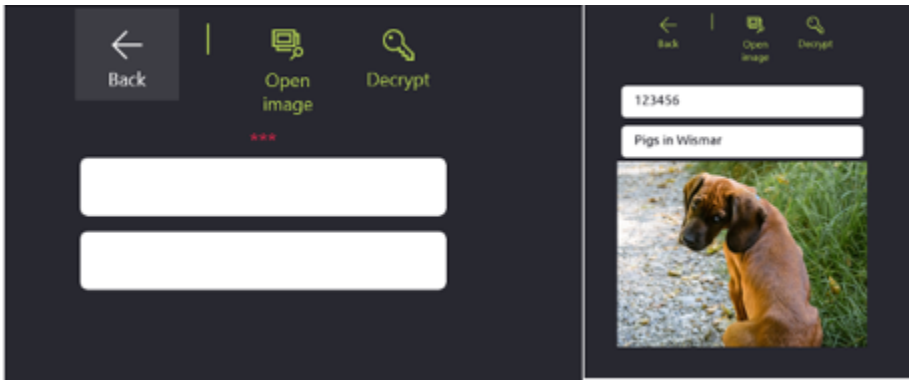

Abbildung 63 - Entschlüsselung der Geheimbotschaft

6.3.8 SSuite Picsel Security

Mit dem Werkzeug SSuite Picsel Security können Textnachrichten in Bildern mit BMP, JPG, PNG oder Windows-Metadatei-Format (WMF) ver- sowie entschlüsselt werden. Dabei können Bilder mit und ohne Passwörter ver- und entschlüsselt werden. Das Verfahren ist in folgender Reihenfolge zu erklären: Ein Bild wird ausgewählt, wo eine Nachricht eingebettet wird. Eine Nachricht wird hochgeladen und ein neues Bild mit der eingebetteten Nachricht wird erstellt. Für die Extrahierung eines Bildes wird zuerst das Ursprungsbild und anschließend das neu erstellte Bild hochgeladen. Daraufhin wird die versteckte Botschaft angezeigt. [42] Die Analyse von SSuite Picsel Security wurde auf dem Betriebssystem Windows 10 Home durchgeführt.

SSuite Picsel Security kann auf der Webseite heruntergeladen werden. Nach der erfolgreichen Installation wird die Anwendung geöffnet. In SSuite Picsel Encryption sind grundsätzlich drei Bereiche erkennbar. Zuerst gibt es die Möglichkeit ein Bild zu ver- oder entschlüsseln. Diese Bild ist das originale, also ursprüngliche Bild. Daraufhin gibt es ein „Delta Image“. In den weißen Balken steht der Pfad der Originaldatei und der Pfad zur Geheimnachricht. Im „Target Image“ wird das fertig erstellte verschlüsselte Bild präsentiert. Über „Message“ wird eine Text-Datei geöffnet und integriert.

Abbildung 64 - SSuite Picsel Security - Übersicht der Funktionen

Im nächsten Schritt wird das Bild „turtle.jpg“ hochgeladen sowie deine erstellte Text-Nachricht. Danach wird „Encrypt Image“ ausgeführt und das Bild als „turtle_pw.png“ gespeichert.

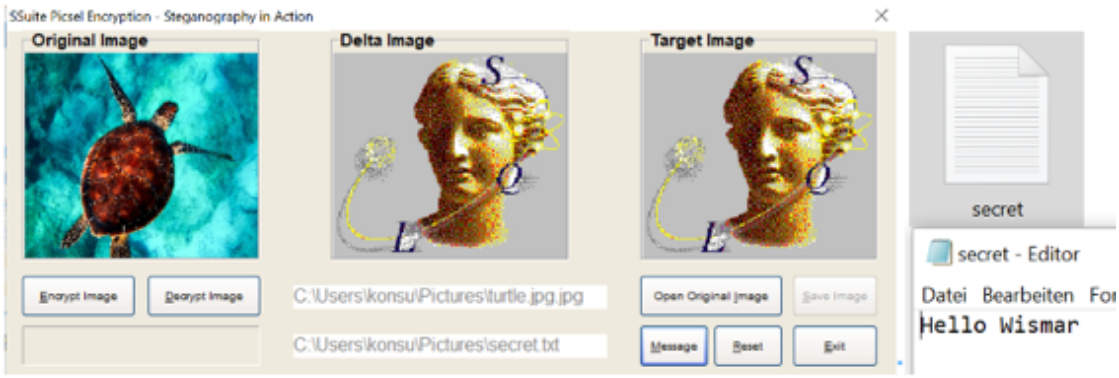

Abbildung 65 - Einbettung einer Geheimbotschaft

SSuite Picsel Securit zeigt in der Übersicht, wie das erstellte verschlüsselte Bild aussieht.

Abbildung 66 - Verschlüsseltes Bild mit Geheimnachricht

Des Weiteren wird die Entschlüsselung ausgeführt. Wichtig ist, dass zuerst das originale Bild „turtle.jpg“ ausgesucht und anschließend das verschlüsselte Bild „turtle_pw.png“ ausgewählt werden muss, da sonst das Passwort nicht korrekt angezeigt wird. Das Ergebnis zeigt die korrekte Geheimbotschaft „Hello Wismar“ an.

Abbildung 67 - Entschlüsselte Geheimnachricht

6.3.9 Stegano

Stegano ermöglicht das Verstecken von Geheimbotschaften in Bildern im PNG-Format sowie Audio-Dateien (WAV). Stegano ist eine Java-basierte Anwendung. Die allgemeine Beschreibung für das Verfahren ist: eine Datei wird ausgewählt und eine Datei mit einer geheimen Nachricht eingebettet. Im Anschluss daran

wird die verschlüsselte Nachricht extrahiert aus dem manipulierten Bild. [43] [44] Die Analyse von Stegano wurde auf dem MacOS-Betriebssystem Ventura 14.4.1 (c) durchgeführt.

Für die Installation des Werkzeugs Stegano wird zuerst die rustup toolchain mit dem Befehl *curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh* und anschließend der stegano-cli mit dem Befehl *cargo install --force stegano-cli* installiert.

```
(base) konsuelabednarek@Konsuelas-MBP ~ % cargo install --force stegano-cli
    Updating crates.io index
  Downloaded stegano-cli v0.5.3
  Downloaded 1 crate (26.5 KB) in 1.27s
  Installing stegano-cli v0.5.3
    Updating crates.io index
   Compiling stegano-core v0.5.3
   Compiling stegano-cli v0.5.3
    Finished release [optimized] target(s) in 1m 05s
  Installing /Users/konsuelabednarek/.cargo/bin/stegano
   Installed package `stegano-cli v0.5.3` (executable `stegano`)
(base) konsuelabednarek@Konsuelas-MBP ~ %
```

Abbildung 68 - Stegano - Installation

Über den Befehl *$HOME/.cargo/bin/stegano* oder *stegano --help* können die Befehle des Werkzeugs eingesehen werden.

```
(base) konsuelabednarek@Konsuelas-MBP ~ % $HOME/.cargo/bin/stegano
Hiding secret data with steganography in PNG images and WAV audio files

Usage: stegano [OPTIONS] [COMMAND]

Commands:
  hide        Hides data in PNG images and WAV audio files
  unveil      Unveils data from PNG images
  unveil-raw  Unveils raw data in PNG images
  help        Print this message or the help of the given subcommand(s)

Options:
      --x-color-step-increment <color channel step increment>
          Experimental: image color channel step increment [default: 1]
  -h, --help
          Print help
  -V, --version
          Print version
(base) konsuelabednarek@Konsuelas-MBP ~ %
```

Abbildung 69 - Übersicht der Befehle

Daraufhin wird der Befehl *stegano hide --help* aufgerufen für die Hilfe zum Verstecken der Geheimbotschaft und der Befehl *stegano unveil --help* zum Entschlüsseln der Geheimbotschaft ausgeführt.

Abbildung 70 - Hilfsbefehl für das Ver- und Entschlüsseln einer Geheimbotschaft

Zusätzlich wird eine Datei im TXT-Format geschrieben mit der Geheimbotschaft „pigs“.

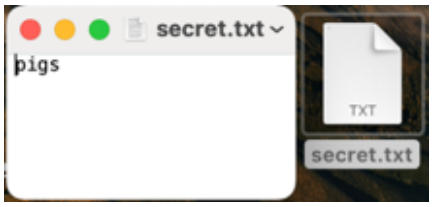

Abbildung 71 - Geheimbotschaft

Das Bild, welches die Geheimbotschaft tragen soll, ist „pig.png“. Anschließend wird der Befehl *stegano hide --data secret.txt --in pig.png --out pignews.png* ausgeführt, damit die Textdatei in das Bild pig.png eingebettet wird. Die neue Bilddatei pignews.png wird erstellt.

```
(base) konsuelabednarek@Konsuelas-MBP Desktop % stegano hide --data secret.txt --in pig.png --out pignews.png
(base) konsuelabednarek@Konsuelas-MBP Desktop %
```

Abbildung 72 - Einbettung der Geheimbotschaft

Abbildung 73 – pig.png ohne Geheimbotschaft (links) und pignews.png mit Geheimbotschaft (rechts)

Daraufhin wird versucht die Geheimbotschaft mit dem Befehl *stegano unveil --in pignews.png --out Bachelor-Thesis* zu extrahieren.

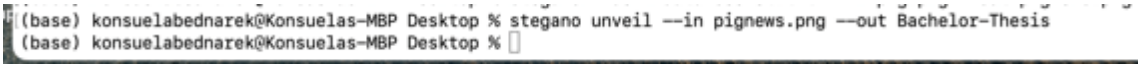

Abbildung 74 - Extrahierung der Geheimbotschaft

Im Ordner Bachelor-Thesis wurde die Datei secret.txt neu erstellt und nach dem Öffnen wird die korrekte Botschaft „pigs“ angezeigt.

Abbildung 75 - Extrahierte Geheimbotschaft

6.3.10 Stegify

Nachfolgend wird das Werkzeug Stegify analysiert. Stegify ist ein kommandobasiertes Werkzeug, mit welchem sich Dateien in Bildern verstecken lassen. Die Technik verwendet die LSB-Methode. Bilder im JPG-Format werden

nach der Einbettung von Geheimnachrichten als PNG-Dateien gespeichert. Das Verfahren funktioniert, indem ein Bild ausgewählt und eine Datei mit einer geheimen Nachricht auswählt wird. Daraufhin wird die Nachricht in das ausgewählte Bild eingebettet. Bei der Extraktion der Nachricht wird eine neue Datei erstellt, die die geheime Nachricht enthält. [45] [46] Die Analyse von Stegify wurde auf dem MacOS-Betriebssystem Ventura 14.4.1 (c) durchgeführt.
Um Stegify zu nutzen wird das Werkzeug mit den Befehlen *brew tap DimitarPetrov/stegify* und anschließend *brew install stegify* ausgeführt. Mit dem Befehl *stegify --help* wird überprüft, welche Befehle das Werkzeug nutzt.

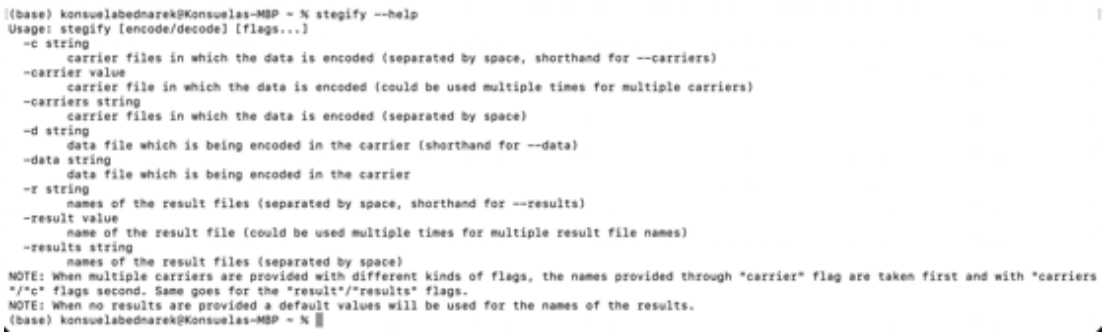

Abbildung 76 - Stegify Befehlsübersicht

Das Bild „pig.png“ wird genommen und die bereits vorhandene secret.txt mit dem Geheimwort „pigs“ versucht einzubetten. Der Befehl *stegify encode --carrier pig.png --data secret.txt --result pig-news.png.*

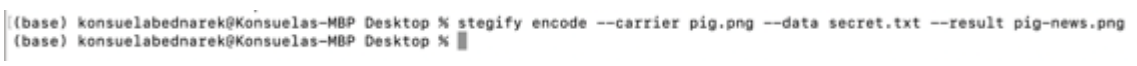

Abbildung 77 - Einbettung Geheimnachricht

Als Ausgabe wird das Bild pig-news.png erstellt.

Abbildung 78 - pig.png ohne Geheimnachricht (links) und pig-news.png mit Geheimnachricht (rechts)

Ein Unterschied zwischen den beiden Bildern ist nicht erkennbar. Daraufhin wird beim erstellten Bild pig-news.png der Befehl *stegify decode --carrier pig-news.png --result privat.txt* ausgeführt und die Datei privat.txt erstellt. Die neu erstellte Datei privat.txt wird geöffnet und die versteckte Botschaft „pigs“ ist erfolgreich extrahiert worden.

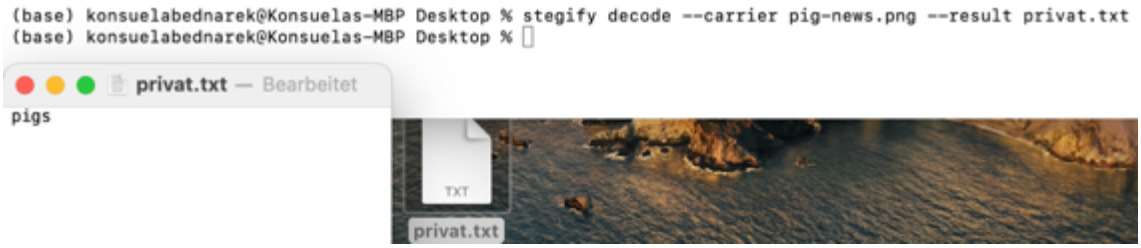

Abbildung 79 - Extrahierung der Geheimbotschaft

6.4 Kostenpflichtige Werkzeuge

In diesem Abschnitt wird ein Überblick über die kostenpflichtigen steganografischen Werkzeuge gegeben. Für die steganografische Analyse werden folgende kostenpflichtige Werkzeuge untersucht: The Hider und East-tec.

6.4.1 The Hider

„The Hider“ ist eine App und wird im App Store für 22,99 € angeboten. Mit „The Hider“ können ebenfalls Bilder im PNG-Format ver- sowie entschlüsselt werden. Die Entschlüsselung funktioniert lediglich mit „The Hider“. Das Verfahren läuft wie

folgt ab: Ein Foto im Format JPEG, PNG oder BMP und eine Datei, die verschlüsselt werden soll, werden hochgeladen. The Hider versteckt die geheime Botschaft in das Bild, kreiert ein neues Bild, welches im PNG-Format erstellt wird. Anschließend kann das verschlüsselte Bild mit The Hider selbst wieder entschlüsselt werden. [47] [48] Die Analyse von The Hider wurde auf dem MacOS-Betriebssystem Ventura 14.4.1 (c) durchgeführt.

Abbildung 80 - Übersicht im App Store "The Hider"

The Hider wird gestartet undin der Übersicht ist zu erkenne, dass eine „Hide“ und eine „Reveal“-Funktion eingebaut ist.

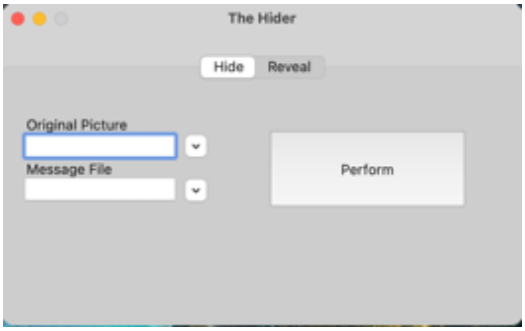

Abbildung 81 - The Hider - Oberfläche der App

Für die Analyse wird das Bild „pig.png“ ausgesucht. Die zu verschlüsselnde Nachricht ist „secret.txt“ mit der Geheimbotschaft „pigs“.

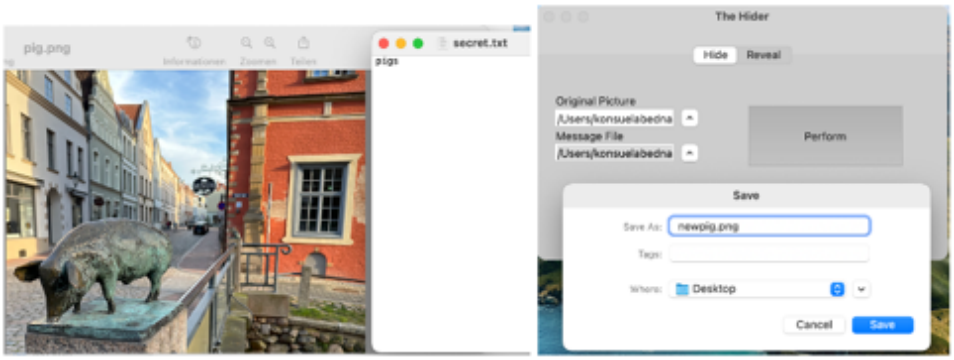

Abbildung 82 - Zu verschlüsselndes Bild "pig.png" und die Geheimbotschaft "secret.txt"

Daraufhin wird „Perform“ gedrückt und es wird ein neues Bild „newpig.png“ erstellt, welches direkt geöffnet wird. Das neu erstellte Bild wird direkt verglichen.

Abbildung 83 - pig.png ohne Geheimbotschaft (links) und newpig.png mit Geheimbotschaft (rechts)

Wie bereits bei den anderen Werkzeugen ist keine optische Veränderung erkennbar. Anschließend wird das neu erstellte Bild „newpig.png“ in die Funktion „Reveal“ hochgeladen und als „information“ gespeichert. Die Datei „information“ wird erstellt und direkt geöffnet. Der richtige verschlüsselte Text „pigs“ wird korrekterweise angezeigt.

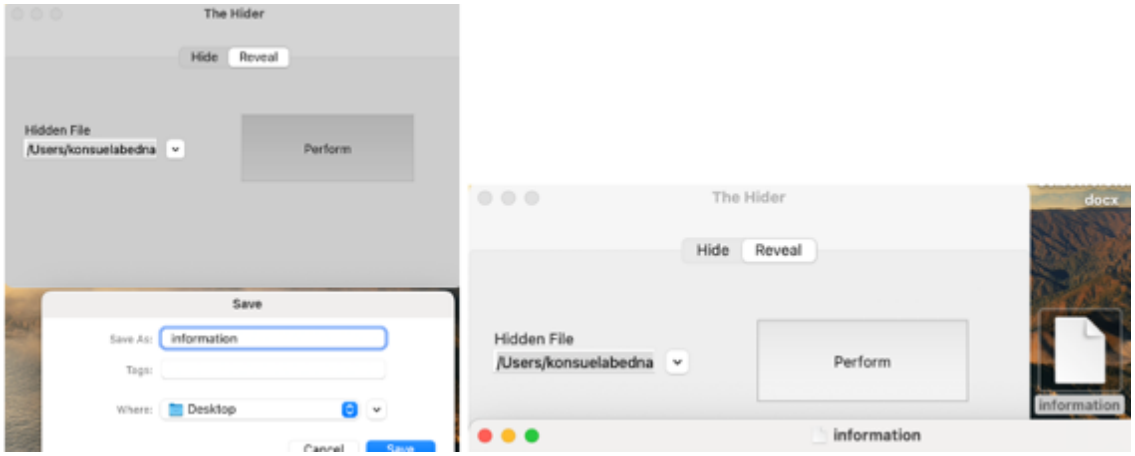

Abbildung 84 - Entschlüsselung des Geheimtextes

Anschließend wird ein JPEG Bild mit der Datei „secret.txt“ zu verschlüsseln. Dafür wird das Bild „emu.jpg“ verwendet. Das Bild „newsemu.png“ wird erstellt und ein visueller Vergleich zeigt keine Auffälligkeiten beim Betrachten beider Bilder.

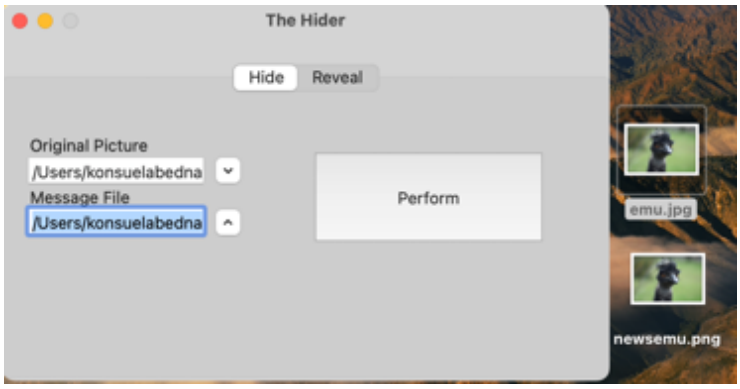

Abbildung 85 - Verschlüsselung "emu.jpg"

Abbildung 86 - emu.jpg ohne Geheimbotschaft (links) und newsemu.png mit Geheimbotschaft (rechts)

Die Entschlüsselung mit JPG-Formaten funktioniert ebenfalls, die neue Datei „emunews“ zeigt die Geheimbotschaft „pigs“ an.

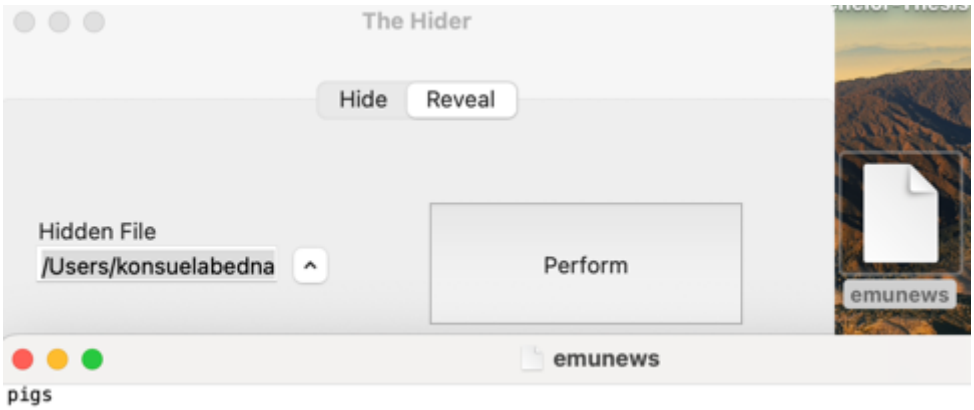

Abbildung 87 - Entschlüsselung "newsemu.png"

6.4.2 East-tec InvisibleSecrets

East-tec InvisibleSecrets kann auf der Webseite kostenpflichtig erworben werden. Die Software für Steganografie kostet 29.95$. East-tec InvisibleSecrets kann Dateien sowie sensible Dateien in andere Dateien verstecken, Dateien mit oder ohne Passwort verschlüsseln, bietet einen Passwortmanager und einen Passwortgenerator an und dient zusätzlich als eine Software, die E-Mails verschlüsselt. Dieses Werkzeug bietet mehr als nur das reine Ver- und Entschlüsseln. Angeboten werden drei Pakete „InvisibleSecrets Plan“ für jährlich 29.95$, „Total Security Plan“ für jährlich 49.95$ und „Privacy Protection Plan“ für

jährlich 39.95$. Ebenfalls wird ein 15-tägiger gratis Zugang angeboten, der genutzt wird. Für die Analyse wird daher nur das Ver- und Entschlüsseln betrachtet. East-tec InvisibleSecrets kann geheime Informationen in die Formate JPEG, BMP, WAV, Hypertext Markup Language (HTML) und PNG einbetten. Das Verfahren verläuft wie folgt ab: zuerst wird eine Datei ausgesucht, die in eine andere Datei eingebettet wird. Danach wird die Datei ausgesucht, in welche die zu versteckende Datei integriert werden soll. Daraufhin wird ein Passwort ausgesucht und kann anschließend das neue Bild abspeichern. [49]

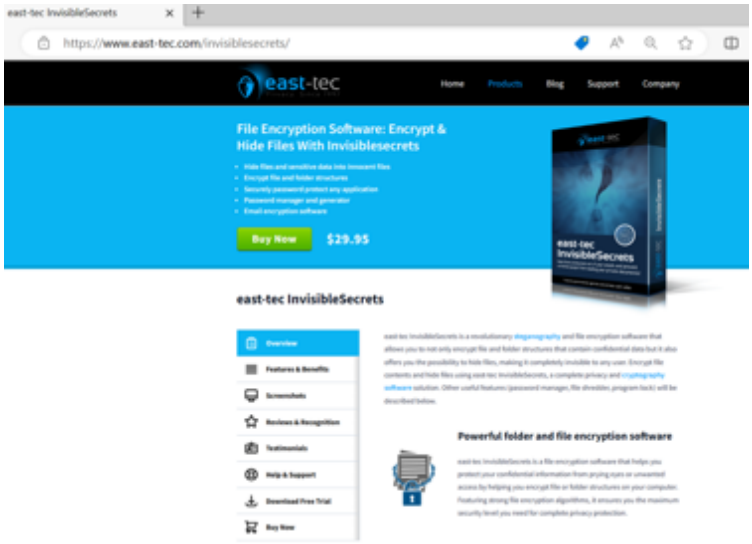

Abbildung 88 - Webseite "east-tec InvisibleSecrets"

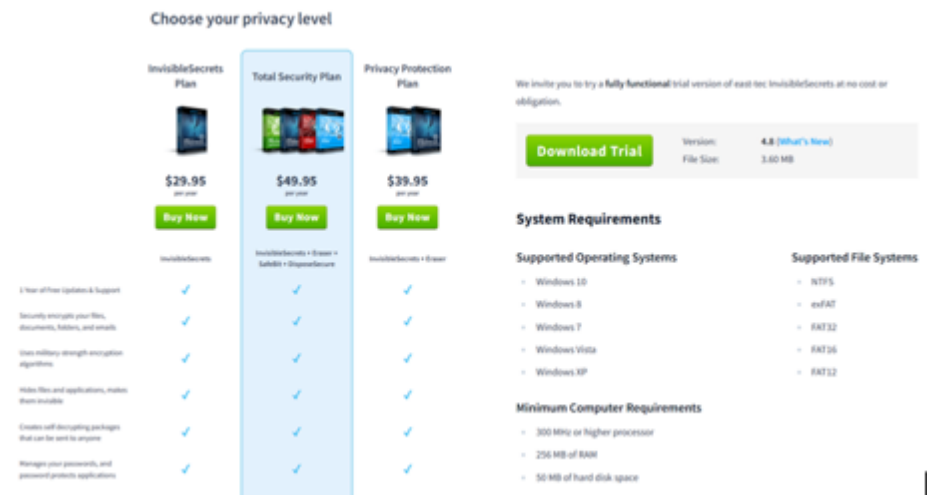

Abbildung 89 - Übersicht der Produktversionen und Gratiszugang

Nach der Installation wird der 15-tägige gratis Zugang genutzt.

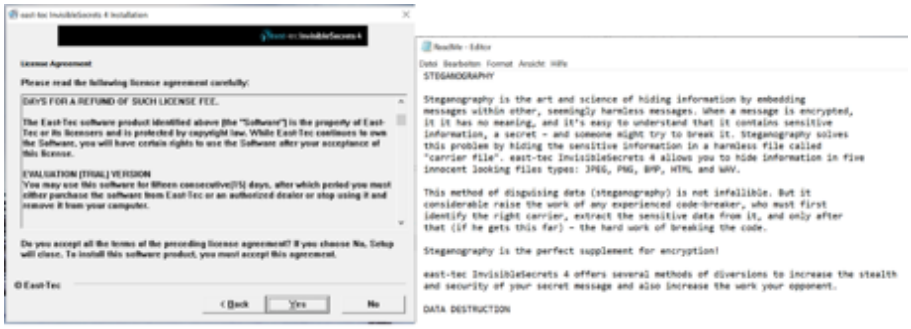

Abbildung 90 - Installation und Auszug von „ReadMe“ von East-tec

East-tec hat eine strukturierte Übersicht und das Programm „Hide Files“ wird geöffnet.

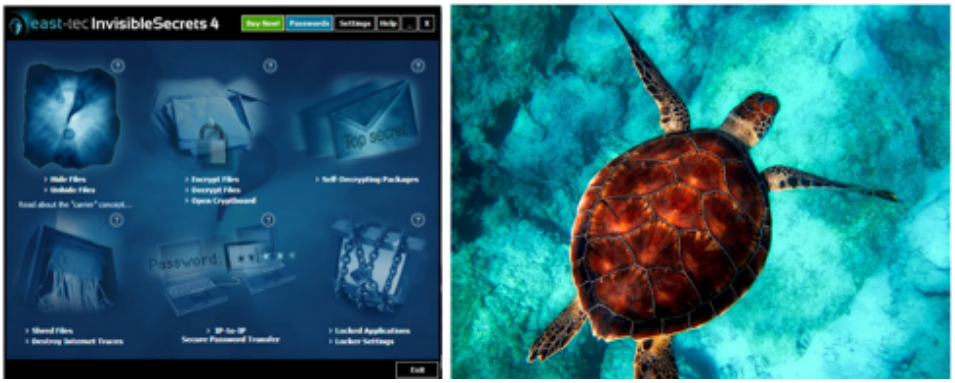

Abbildung 91 - Übersicht Anwendungen East-tec und zu verschlüsselndes Bild turtle.png

Zuerst wird die secret.txt Datei, welche den Textinhalt „Hello Wismar“ beinhaltet, hochgeladen, da beim ersten Schritt die zu versteckende Datei ausgesucht wird. Im nächsten Schritt wird das Bild „turtle.png“ (Bild-Quelle) ausgesucht, das Bild, in welche der Geheimtext eingebettet werden soll.

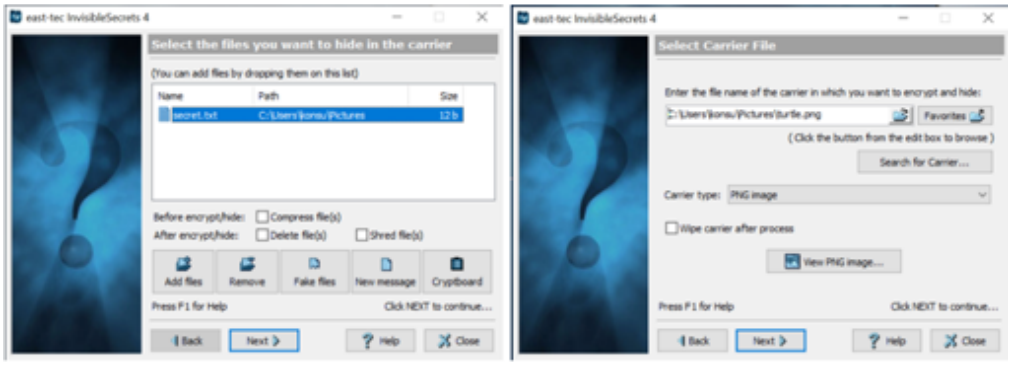

Abbildung 92 - Auswahl Geheimbotschaft und Bild "turtle.png"

Im darauffolgenden Schritt wird ein Passwort, in diesem Fall die 1234, ausgesucht, eine extra Verschlüsselung abgewählt, ein neuer Bildname „newsturtle.png“ festgelegt und das Bild mit „Hide“ versteckt. Die Anwendung zeigt, dass die Einbettung funktioniert hat, ein neues Bild „newsturtle.png“ erstellt wurde und das Bild geöffnet werden kann.

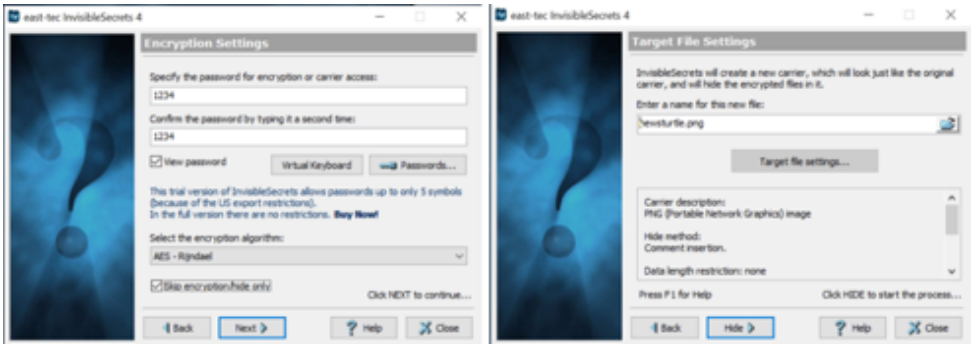

Abbildung 93 - Festlegung Passwort und Bildname "newsemu.png"

Abbildung 94 - Versteckens, Möglichkeiten des Anzeigens und neues Bild „newsturtle.ong“

Beide Bilder werden betrachtet und keine Veränderung festgestellt.

Abbildung 95 - "turtle.png" ohne (links) und "newsturtle.png" mit Geheimbotschaft (rechts)

Die Einbettung funktioniert, daher wird direkt die Anwendung „Unhide Files“ untersucht. Zuerst wird das Bild „newsturtle.png“ ausgesucht, dann wird das Passwort 1234 eingegeben.

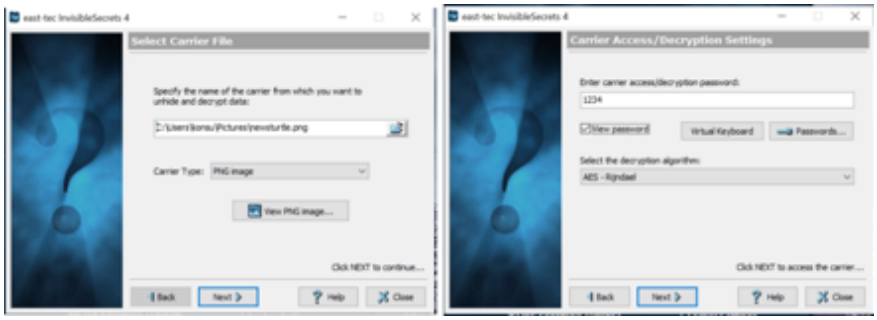

Abbildung 96 - Extraktion und Eingabe des Passworts

Daraufhin wird der neue Speicherort ausgewählt, die Extraktion durchgeführt und eine neue Datei „secret“ erstellt.

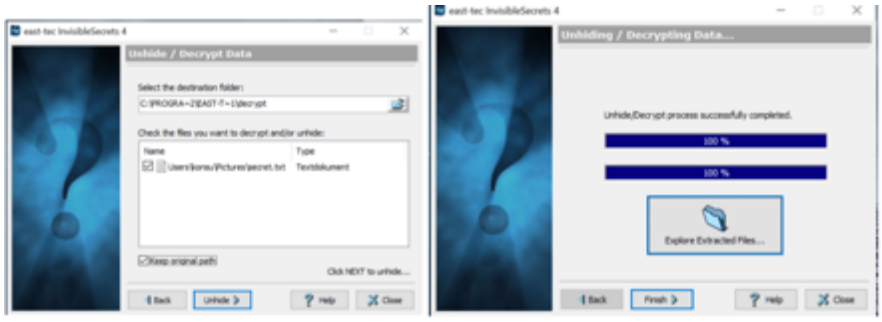

Abbildung 97 - Erfolgreiche Extraktion

Die neu erstellte Datei „secret“ wird geöffnet und der Text „Hello Wismar“ wird korrekterweise angezeigt.

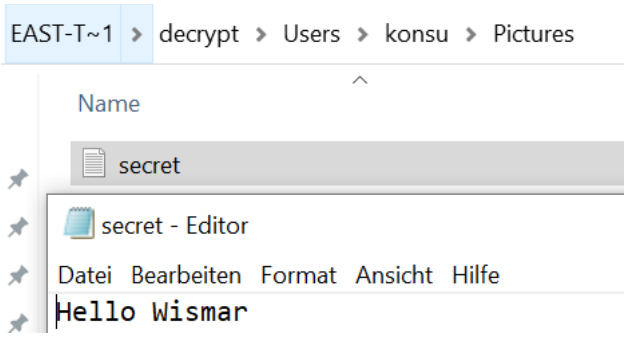

Abbildung 98 - Extrahierte Geheimbotschaft

6.5 Vergleich

Nachdem die verschiedenen steganografischen Werkzeuge vorgestellt, beschrieben und mit Beispielbildern angewendet wurden, wird in diesem Kapitel ein Vergleich der Werkzeuge durchgeführt. Bei der Untersuchung wurden zwölf steganografische Werkzeuge vorgestellt. Eine Bewertung kann anhand von subjektiven und von objektiven Gütekriterien erfolgen. Im Rahmen dieser Bachelor-Thesis werden die subjektiven Bewertungskriterien nicht betrachtet, sondern nur die objektiven Bewertungskriterien untersucht.

6.5.1 Objektive Gütekriterien

Zu den objektiven Bewertungskriterien gehören folgende Punkte: Betriebssysteme, Formate, Dokumentationen, Benutzeroberfläche (User Interface (UI)) versus (vs.) Befehlszeile (command-line interface (CLI)), Dateigröße, Mean Squared Error (MSE), Peak Signal-to-Noise Ratio (PSNR) und Structural Similarity Index (SSIM). Das Kriterium Betriebssystem überprüft, ob die steganografisches Werkzeug auf den Betriebssystemen Linux, Windows oder MacOS funktionieren. Anschließend wird angekreuzt, ob das Werkzeug kostenlos oder kostenpflichtig ist. Das nächste Kriterium sind Formate, wo aufgelistet wird, welches Werkzeug welche Bildformate unterstützt. Daraufhin wird bei der Dokumentation überprüft, ob eine aktuelle Dokumentation und Tutorials verfügbar sind (Kriterium: Ja, Wenig oder Nein). Beim nächsten Kriterium wird betrachtet, welche Benutzeroberfläche von den steganografischen Werkzeugen genutzt wird (Kriterium: UI oder CLI). Beim Kriterium Dateigröße

wird analysiert, ob die Dateigröße beeinflusst wird nach einer Manipulation (Kriterium: Ja oder Nein). Daraufhin wird das Kriterium MSE aufgeführt, mit welchem ebenfalls die Qualität beider Bilder verglichen wird. Der MSE-Wert berechnet einen durchschnittlichen quadratischen Fehler zwischen den Pixelwerten beider Bilder und kann eine quantitative Bewertung der Ähnlichkeit der Bilder geben. Je niedriger der Wert von MSE ist, desto geringer ist der Fehler (Kriterium: MSE-Wert). [23] Anschließend wird das Kriterium PSNR betrachtet, welches die Qualität der Bilder misst und angibt, wie stark die Qualität des Bildes durch eine Kompression oder Veränderung beeinflusst wurde. Ein höherer PSNR-Wert zeigt eine höhere Bildqualität an und zeigt weniger Unterschiede zwischen dem originalen Bild und dem manipulierten auf (Kriterium: PSNR-Wert). [23] [50] Beim SSIM-Kriterium kann die Strukturähnlichkeit zwischen zwei Bildern bestimmt werden. Der SSIM-Wert liegt zwischen -1 und 1. Eine perfekte Ähnlichkeit wird mit dem Wert 1 angezeigt. Ein SSIM-Wert bei -1 zeigt eine geringe Ähnlichkeit sowie eine schlechte Bildqualität an. Im Gegensatz dazu zeigt ein Wert nahe 1 eine hohe Ähnlichkeit sowie eine bessere Bildqualität an (Kriterium: SSIM-Wert). [51] Bei der Bewertung der MSE-, PSNR- sowie SSIM-Werte sind die jeweiligen folgenden Zielgrößen wie folgt zu klassifizieren: der MSE-Wert soll möglichst bei 0 liegen und daher möglichst klein sein, denn wenn der MSE-Wert nahe 0 liegt und somit niedrig ist, dann deutet dies auf eine geringere Fehlerquote hin. Der PSNR-Wert soll im Gegensatz zum MSE-Wert möglichst hoch sein, denn je größer ein PSNR-Wert ist, desto schwieriger ist für das menschliche Auge zu erkennen, dass eine Manipulation erfolgt ist. [52] Im Allgemeinen sollte der PSNR-Wert über 30 dB liegen, um eine gute Qualität des Stegobildes aufzuweisen. [53] Für die Bewertung der SSIM-Werte wird als Zielgröße der Wert 1 festgelegt. Ein SSIM-Wert nahe 1 sagt aus, dass das erzeugte Bild nahezu identisch ist mit dem Originalbild. [52] Sowohl die MSE, PSNR als auch SSIM-Wert sind gerundete Werte. Die MSE-Werte sind bis auf 6 Nachkommastellen, die PSNR sind auf 2 Nachkommastelle und die SSIM-Werte sind bis auf 4 Nachkommastellen gerundet.

6.5.2 Erklärung der Formeln MSE, PSNR und SSIM

In den objektiven Gütekriterien werden die MSE, PSNR und SSIM-Werte

vergleichen. Alle drei Faktoren sind für die steganografische Analyse von Bedeutung, da eine rein optische Betrachtung der Bilder vor und nach der Einbettung von Geheimbotschaften keine Anzeichen auf eine Manipulation schließen lassen. Mit diesen Werten kann die Bildqualität beider Bilder verglichen werden und die Werte zeigen, ob eine mögliche Manipulation stattgefunden und wie gut diese funktioniert hat. [53] Die Bildqualität von steganografischen Bilder wird gemessen mithilfe von Image Quality Measurement (IQM). Eine Verzerrung der steganografischen Bilder muss so gering wie möglich sein, damit die Unterschiede nicht erkennbar sind. Zur Messung der IQM können die Faktoren MSE, PSNR sowie SSIM genutzt werden. [54]
Der Faktor MSE ist eine wichtige Metrik zur Bewertung der Bildqualität. Der MSE berechnet den Durchschnitt der Quadrate der Fehler bzw. Abweichungen zwischen dem Originalbild und dem manipulierten Bild. Die Formel dazu lautet:

$$MSE = \frac{1}{m\,x\,n} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} [f(i,j) - f'(i,j)]^2$$

M ist dabei die Anzahl der Zeilen des Deckbildes, *N* ist die Anzahl der Spalten des Deckbildes, *f(i,j)* ist das Pixel des Coverbildes (Originalbild) und *f '(i,j)* ist der Pixelwert des rekonstruierten bzw. komprimierten Bildes an der Position *(i,j)*. [55] Wie bereits beschrieben bedeutet das, dass je niedriger der Wert von MSE ist, desto geringer der Fehler ist und je höher der MSE-Wert ist, desto größer sind die Unterschiede zwischen dem Originalbild und dem manipulierten Bild. [55] In dieser Bachelor-Thesis wird für die Berechnung der MSE-Werte die MATLAB-Bibliothek verwendet. [56]

In der Steganografie ist die PSNR-Messung die populärste Messung zur Bewertung der Qualität eines Stego-Bildes verwendet wird. PSNR wird zur Bewertung der Verzerrung zwischen den beiden Bildern verwendet und wird durch die Verwendung des mittleren quadratischen Fehlers (MSE) definiert. [23] Das Ergebnis bzw. die Einheit des PSNR-Werts wird in Dezibel (dB) ausgegeben. [57] Die Formel für die PSNR-Berechnung lautet:

$$\mathrm{PSNR} = 10 \log 10 \left(\frac{max^2}{MSE} \right)$$

MAX steht für die maximalen möglichen Pixelwerte im Bild (z. B. 255 für 8-Bit-Graufstufenbilder oder 16777215 für 24-Bit-Farbbilder). *MSE* steht für den mittleren quadratischen Fehler und wird berechnet, indem der quadratische Unterschied zwischen den Pixelwerten jedes Paares von entsprechenden Pixeln im Stego-Bild und Cover-Bild ermittelt und über das gesamte Bild ermittelt wird. Das bedeutet, dass je höher der PSNR-Wert ist, desto geringer ist der Unterschied zwischen den Pixelwerten des Stego-Bildes sowie des Cover-Bildes ist und weist damit auf eine bessere Qualität der versteckten Daten hin. PSNR ist ein logarithmisches Maß ist und somit steigt die subjektive empfundene Bildqualität nicht linear mit einem Anstieg des PSNR-Wertes, denn eine Erhöhung des PSNR-Wertes um 10 entspricht nicht einer ungefähren Verdoppelung der subjektiven empfundenen Qualität. Zum Beispiel führt eine Steigerung von 30 auf 40 PSNR zu einer Verbesserung der Bildqualität, welche nicht als doppelt so gut empfunden wird, jedoch als verbessert. [53] Für die Berechnung der PSNR-Werte wird die Webseite MATLAB genutzt. [58] Zudem gibt es eine CLI-Anwendung mit dem Namen *Imagemagick*, die zusätzlich genutzt wird, um den PSNR zwischen den beiden Bildern zu überprüfen. [59]

Der nächste Faktor ist SSIM. Die Formel dafür ist:

$$SSIM\ (x, y) = \frac{(2\mu_x\mu_y + c1)(2\sigma_{xy} + c2)}{(\mu_x^2 + \mu_y^2 + c1)(\sigma_x^2 + \sigma_y^2 + c2)}$$

X und y stellen die Fenstergrößen der zu vergleichender Bilder dar, μ ist der Durchschnitt der Bilder x und y, σ^2 ist die Varianz von x und y, σ ist die Kovarianz der beiden Bilder und c1 sowie c2 sind zwei Variablen, die der Stabilisierung der Division mit dem schwachen Nenner dienen. [53] [52] Der SSIM-Wert zeigt, wie ähnlich die Struktur der Bilder ist. Wie bereits beschrieben, definiert der Wert 1 eine perfekte Ähnlichkeit. [54] Für die Berechnung der SSIM-Werte wurde die MATLAB-Bibliothek verwendet. [60]

6.5.3 Objektive Gütekriterien (Tabelle)

In der Tabelle, die die objektiven Gütekriterien auflistet, sind die untersuchten steganografischen Werkzeuge aufgelistet. Mit jedem Werkzeug wurde jeweils ein Bild ver- und entschlüsselt. Das Original- und das Coverbild wurden verglichen und die Werte in die Tabelle eingetragen. Die Berechnung der PSNR-, SSIM- und MSE-Werte sind im Anhang unter dem Punkt *Anhang 1* zu entnehmen.

Stegano-grafisches Werkzeug	Betriebs-systeme	Kostenlos	Kostenpflichtig	Formate	Dokumentation	Benutzer-oberfläche	Dateigröße	MSE	PSNR	SSIM
Steghide	Windows, Linux	X		JPEG, BMP, WAV, AU	Ja	CLI	Ja	0,031822	63,10	0,9999
Stegosuite	Linux	X		BMP, GIF, JPEG, PNG	Ja	UI	Ja	3,230642	43,04	0,9937
OutGuess	Linux	X		JPEG, PPM, PNM	Ja	CLI	Ja	8,158686	39,01	0,9862
Zsteg	Linux	X		PNG, BMP	Ja	CLI	Ja	0,000002	105,57	1,0000
Tater	Windows	X		JPEG, PNG	Wenig	UI	Ja	0,052941	60,89	0,9999
PicStealth	Windows	X		JPEG, PNG	Wenig	UI	Ja	0,052941	60,89	0,9999
						UI	Ja	0,039550	62,16	0,9999
F2d	Windows	X		JPEG, PNG	Wenig	UI	Ja	0,056004	60,65	0,9999
SSuite Picsel Security	Windows	X		BMP, JPEG, PNG, WMF	Ja	UI	Ja	0,196491	55,20	0,9999
Stegano	MacOS, Windows, Linux	X		PNG, WAV	Ja	CLI	Ja	0,000084	88,91	1,0000
Stegify	MacOS, Linux	X		JPEG, PNG	Ja	CLI	Ja	0,000028	93,66	1,0000
The Hider	MacOS		X	JPEG, PNG, BMP	Wenig	UI	Ja	0,000486	81,26	1,0000
East-sec InvisibleSecrets	Windows		X	JPEG, BMP, PNG, HTML	Ja	UI	Nein	0,000000	-	1,0000

Tabelle 5 - Objektive Gütekriterien

6.5.4 Bewertung der objektiven Gütekriterien

Nach der Auflistung der objektiven Gütekriterien in der Tabelle kann eine Bewertung der steganografischen Werkzeuge durchgeführt werden. Für die Bewertung werden die Kriterien der IQM genommen, also die PSNR, SSIM und MSE-Werte.

Steghide: Der PSNR-Wert von 63,10 ist relativ hoch, was hinweist, dass das versteckte Bild eine hohe Ähnlichkeit mit dem Originalbild hat. Der SSIM-Wert von 0,9999 ist nahezu 1 und bedeutet, dass eine hohe strukturelle Ähnlichkeit zwischen dem versteckten Bild und dem Originalbild und somit eine sehr gute Bildqualität hat. Der MSE-Wert von 0,031822 ist sehr niedrig und zeigt, dass nur eine geringfügige durchschnittliche Abweichung zwischen den Pixelwerten des versteckten Bildes sowie dem Originalbild besteht.

Stegosuite: Der PSNR-Wert von 43,04 ist relativ niedrig im Vergleich zu Steghide und bedeutet, dass das versteckte Bild eine geringere Ähnlichkeit mit dem Originalbild aufweist und die Abweichungen der Pixelwerte im Vergleich zum Originalbild etwas größer sind. Der SSIM-Wert von 0,9937 ist ebenfalls nahezu 1, jedoch etwas geringer als bei Steghide und bedeutet, dass eine hohe strukturelle Ähnlichkeit zwischen dem versteckten Bild und dem Originalbild und somit eine sehr gute Bildqualität hat. Der MSE-Wert von 3,230642 weist hin, dass etwas größere Abweichungen zwischen den Pixelwerten des versteckten Bildes sowie dem Originalbild bestehen.

OutGuess: Der PSNR-Wert von 39,01 ist niedriger als bei Steghide und Stegosuite und bedeutet, dass das versteckte Bild eine noch geringere Ähnlichkeit mit dem Originalbild aufweist. Der SSIM-Wert von 0,9862 ist ebenfalls relativ hoch, aber etwas geringer als bei den Werkzeugen davor und der Abstand zu 1 ist größer. Daher gibt es bei OutGuess eine etwas geringere strukturelle Ähnlichkeit des versteckten Bildes und des Originalbildes. Der MSE-Wert von 8,158686 ist höher als bei den beiden Werkzeugen davor und bedeutet, dass eine größere Abweichung zwischen den Pixelwerten des versteckten Bildes und des Originalbildes existiert. Das versteckte Bild kann daher größere

Abweichungen zum Originalbild aufweisen und die Bildqualität kann insgesamt geringer sein.

Zsteg: Der PSNR-Wert von 105,57 ist sehr hoch und deutet darauf hin, dass das versteckte Bild eine nahezu perfekte Übereinstimmung mit dem Originalbild hat und die Abweichungen der Pixelwerte zwischen dem versteckten Bild sowie dem Originalbild sehr gering sind. Der SSIM-Wert von 1,0000 ist eine sehr hohe Bewertung und sagt aus, dass eine vollständige strukturelle Ähnlichkeit zwischen dem versteckten Bild und dem Originalbild vorhanden ist und deutet auf eine sehr gute Bildqualität des zu versteckten Bildes hin. Der MSE-Wert von 0,000002 zeigt, dass fast keine quadratische Abweichung zwischen den Pixelwerten des versteckten Bildes und des Originalbildes vorhanden ist. Dies bedeutet, dass das versteckte Bild fast exakt mit dem Originalbild übereinstimmt und es nur sehr wenige Unterschiede zwischen den Bildern gibt. Hier wird jedoch eine Abgrenzung zu den anderen Bildern geführt. Da Zsteg keine Möglichkeit hat, eigene Bilder zu verschlüsseln, können dieses objektive Gütekriterien mit den anderen Werkzeugen verglichen werden. Bei allen anderen Werkzeugen werden Bilder mit dem jeweiligen Werkzeug ver- und entschlüsselt. Da bei Zsteg auf einen externen Code eine Verschlüsselung zurückzuführen ist, ist eine Bewertung der Werte zwischen Ver- und Entschlüsselung nicht korrekt.

Tater: Der PSNR-Wert von 60,89 ist relativ hoch. Das sagt aus, dass das versteckte Bild eine hohe Ähnlichkeit mit dem Originalbild hat. Der SSIM-Wert von 0,9999 ist nahezu 1 und bedeutet, dass eine hohe strukturelle Ähnlichkeit zwischen dem versteckten Bild und dem Originalbild und somit eine sehr gute Bildqualität hat. Der MSE-Wert von 0,052941 ist sehr niedrig und zeigt, dass nur eine geringfügige durchschnittliche Abweichung zwischen den Pixelwerten des versteckten Bildes sowie dem Originalbild besteht und eine hohe Bildqualität vorhanden ist.

PicStealth: Bei PicStealth wurden zwei Bilder analysiert, davon wurde ein Bild mit einem Passwort und ein Bild ohne Passwort verschlüsselt. Beide Bilder weisen ähnliche Werte auf. Die PSNR-Werte von 60,89 und 62,16 sind relativ hoch und

zeigen eine hohe Ähnlichkeit mit dem Originalbild auf. Die SSIM-Werte von 0,9999 deuten auf eine ausgezeichnete strukturelle Ähnlichkeit zwischen dem versteckten Bild und dem Originalbild hin. Die MSE-Werte von 0,052941 und 0,039550 sind relativ niedrig und bedeuten, dass die versteckten Bilder den Pixelwerten der Originalbilder ziemlich ähneln und nur wenig Abweichung bestehen.

F2d: Der PSNR-Wert von 60,65 ist vergleichsweise hoch und zeigt, dass eine hohe Ähnlichkeit zum Originalbild besteht. Der SSIM-Wert von 0,9999 ist nahezu 1 und bedeutet, dass eine hohe strukturelle Ähnlichkeit zwischen dem versteckten Bild und dem Originalbild und somit eine sehr gute Bildqualität hat. Der MSE-Wert von 0,056004 ist sehr niedrig und zeigt, dass nur eine geringfügige durchschnittliche Abweichung zwischen den Pixelwerten des versteckten Bildes sowie dem Originalbild besteht und eine hohe Bildqualität vorhanden ist.

SSuite Picsel Security: Der PSNR-Wert von 55,20 ist relative hoch, geringer als bei den meisten anderen untersuchten steganografischen Werkzeugen und zeigt damit auch eine hohe Ähnlichkeit zum Originalbild auf. Der SSIM-Wert von 0,9999 ist nahezu 1 und bedeutet, dass eine hohe strukturelle Ähnlichkeit zwischen dem versteckten Bild und dem Originalbild und somit eine sehr gute Bildqualität hat. Der MSE-Wert von 0,196491 ist relativ niedrig und sagt aus, dass eine geringe durchschnittliche quadratische Abweichung zwischen den Pixelwerten des versteckten Bildes und des Originalbildes vorhanden ist.

Stegano: Der PSNR-Wert von 88,91 ist außergewöhnlich hoch und bedeutet, dass das versteckte Bild eine nahezu perfekte Übereinstimmung mit dem Originalbild hat und die Abweichungen der Pixelwerte zwischen dem versteckten Bild und dem Originalbild sehr gering sind. Der SSIM-Wert von 1,0000 ist eine sehr gute Bewertung und sagt aus, dass eine vollständige strukturelle Ähnlichkeit zwischen dem versteckten Bild und dem Originalbild vorhanden ist und deutet auf eine sehr gute Bildqualität des zu versteckten Bildes hin. Der MSE-Wert von 0,000084 zeigt, dass fast keine quadratische Abweichung zwischen den

Pixelwerten des versteckten Bildes und des Originalbildes vorhanden ist. Dies bedeutet, dass das versteckte Bild fast exakt mit dem Originalbild übereinstimmt und es sehr wenige Unterschiede zwischen den Bildern gibt.

Stegify: Der PSNR-Wert von 93,66 ist äußert hoch und bedeutet, dass das versteckte Bild eine nahezu perfekte Übereinstimmung mit dem Originalbild hat und die Abweichungen der Pixelwerte zwischen dem versteckten Bild und dem Originalbild sehr gering sind. Der SSIM-Wert von 1,0000 und der MSE-Wert von 0,000028 zeigen, dass die Struktur und Textur zwischen den beiden Bildern nahezu gleich sind und es kaum quadratische Abweichung zwischen den Pixelwerten zwischen den beiden Bildern gibt.

The Hider: Der PSNR-Wert von 81,91 ist außergewöhnlich hoch und bedeutet, dass das versteckte Bild eine nahezu perfekte Übereinstimmung mit dem Originalbild hat und die Abweichungen der Pixelwerte zwischen dem versteckten Bild und dem Originalbild sehr gering sind. Der SSIM-Wert von 1,0000 und der MSE-Wert von 0,000486 zeigen, dass die Struktur und Textur zwischen den beiden Bildern ähnlich sind und es wenig quadratische Abweichung zwischen den Pixelwerten beider Bilder gibt.

East-tec InvisibleSecrets: Der PSNR-Wert wird in diesem Fall nicht geliefert. Sowohl bei MATLAB als auch über die CLI-Anwendung kam kein Ergebnis. Die SSIM- und MSE-Werte liefern jedoch gute Werte. Der SSIM-Wert von 1,0000 ist eine sehr hohe Bewertung und der MSE-Wert von 0,000000 ist nahezu null, da der Wert auf 6 Nachkommastellen gerundet ist. Besonders interessant ist die Tatsache, dass sich die Dateigröße nicht geändert hat. Bei allen anderen steganografischen Werkzeugen wurden die Bilder entweder kleiner oder größer, sobald die Bilder mit einer Geheimnachricht eingebettet wurden. Bei dem Werkzeug gibt es keine Änderung in der Dateigröße. Da die SSIM- und MSE-Werte für eine sehr gute Qualität des Versteckens stehen, lässt sich vermuten, dass der PSNR-Wert nahezu perfekt ist. Das bedeutet, dass nach der Einbettung des Bildes fast keine subjektive und objektive Änderung feststellbar ist.

Zusammenfassung und Bewertung:

Da die Berechnung der PSNR-Werte die populärste Methode ist zur Messung der Bildqualität nach der Verarbeitung eines Bildes, kann eine Bewertung der steganografischen Werkzeuge primär nach Betrachtung der PSNR-Werte erfolgen. Der PSNR-Wert bewertet die Bildqualität nach der Einbettung einer Nachricht in das Originalbild, je höher der PSNR-Wert, desto besser ist die Bildqualität. Somit kann der PSNR-Wert einen sehr guten Eindruck vermitteln, welches Werkzeug sich am besten für eine Übertragung von geheimen Botschaften eignet, denn entscheidend ist, dass das erstellte Medium so unauffällig wie möglich ist, damit kein Dritter erkennt, dass eine Manipulation erfolgt ist. Je besser die Qualität, desto geringer ist die Auffälligkeit der manipulierten Bilder. Anschließend kann der MSE-Wert zur Bewertung herangezogen werden. Niedrige MSE-Werte zeigen an, dass das versteckte Bild den Originalpixelwerten sehr nahekommt. Zudem impliziert ein niedriger MSE-Wert grundsätzlich einen hohen PSNR-Wert, weil beide Werte miteinander zusammenhängen. Dies ist in der Tabelle erkennbar. Zuletzt kann der SSIM-Wert zur Bewertung herangezogen werden, die eine Bewertung der strukturellen Ähnlichkeit zwischen dem versteckten und Originalbild geben. Der Wert 1,0000 bedeutet eine fast perfekte Ähnlichkeit zwischen den beiden Bildern. Aufgrund dessen kann eine Werkzeugbewertung basierend auf den PSNR-Werten erfolgen, weil dieser am aussagekräftigsten ist, da diese die visuelle Qualität der mit den steganografischen Werkzeugen erzeugten Ergebnissen angibt. Schließlich lässt sich zusammenfassend sagen, dass folgende Werkzeuge sich besonders gut für die Nutzung von Steganografie eignen: Stegify, Stegano und The Hider (hohe PSNR-Werte, seht gute SSIM- und niedrige MSE-Werte), Stegify, Stegano, The Hider und zu den weniger guten Werkzeugen gehören Stegosuite und OutGuess (niedrige PSNR- und SSIM-Werte und höhere MSE-Werte). Da bei East-tec InvisibleSecrets kein PSNR-Wert bestimmt werden kann, ist eine Bewertung nicht möglich. Dies gilt ebenfalls für Zsteg, da dieses Werkzeug keine eigene Verschlüsselung durchführen kann und somit nur durch eine externe Quelle verschlüsselt wird, welches dadurch ein Ergebnis verfälschen kann. In der nachfolgenden Tabelle werden die steganografischen Werkzeuge nach den PSNR-, MSE- und SSIM-Werten absteigend sortiert.

Werkzeug	PSNR	MSE	SSIM
Stegify	93,66 dB	0,000028	1,0000
Stegano	88,91 dB	0,000084	1,0000
The Hider	81,26 dB	0,000486	1,0000
Steghide	63,10 dB	0,031822	0,9999
Tater	60,89 dB	0,052941	0,9999
PicStealth	60,89 dB	0,052941	0,9999
F2d	60,65 dB	0,056004	0,9999
SSuite Picsel Security	55,20 dB	0,196491	0,9999
Stegosuite	43,04 dB	3,230642	0,9937
OutGuess	39,01 dB	8,158686	0,9862
Nicht bewertbare Werkzeuge	PSNR	MSE	SSIM
Zsteg	105,57 dB	0,000002	1,0000
East-tec InvisibleSecrets	-	0,000000	1,0000

Tabelle 6 - Bewertung steganografischer Werkzeuge

6.5.5 Vergleich Werkzeug mit großer Datei bzw. eines Schadcodes

Per Steganografie lassen sich nicht nur Dateien in anderen Bildern einbetten, sondern auch Schadcode, der mittlerweile in Bilder versteckt werden kann. Ein Schadcode kann vollkommen unbemerkt in ein Bild integriert werden und wenn dieses manipulierte Bild auf einen Computer gelangt, aus der Datei ausgelesen wird, dann kann der Schadcode ausgeführt werden. [61] In diesem Abschnitt wird daher überprüft, wie sich die Werte verändern, wenn eine Datei, die einem Schadcode ähnelt, mit viel größeren Datenvolumen eingebettet wird. Dabei werden die zwei Werkzeuge ausgewählt, bei denen die PSNR-Werte hoch waren. Zu den zwei Werkzeugen gehören daher Stegify und Stegano. Bei Stegify

wurde zuvor die Datei secret.txt in das Bild pig.png eingebettet. Die Datei secret.txt hat eine Größe von 4 Byte und ist daher relativ klein. Der PSNR-Wert liegt bei 93,66 dB und ist relativ hoch.

Das Bild pig.png wird erneut verwendet, um eine größere Datei einzubetten. Für die Einbettung von größeren Dateien werden zuerst verschiedene Textdateien mit verschiedenen Größen erstellt. Zu den erstellten Dateigrößen gehören Gigabyte (GB), Megabyte (MB) sowie Kilobyte (KB). Folgende Dateien werden erstellt: news.txt (1 GB), news1.txt (500 MB), news2.txt (250 MB), news3.txt (100 MB), news4.txt (50 MB), news5.txt (25 MB), news6.txt (10 MB), news7.txt (1 MB), news8.txt (500 KB), news9.txt (800 KB) sowie news10.txt (700 KB). Daraufhin wird nacheinander versucht das Bild pig.png mit den Dateien mit den Befehl *stegify encode --carrier pig.png --news.txt --result secret_pigs.png* zu verschlüsseln. Das Ergebnis zeigt, dass eine Verschlüsselung oft nicht möglich ist, da die zu verschlüsselnde Textnachricht zu groß ist („*data file too large for this carrier.*“). Lediglich werden zwei neue Bilder erstellt. Das neue Bild secret_pigs.png wurde mit der Datei news8.txt (500 KB) und das neue Bild secret_pigs2.png wurde mit der Datei news10.txt (700 KB) verschlüsselt. Größere Datei können daher nicht verschlüsselt werden, da eine Begrenzung der Datengröße für die Verschlüsselung bei Stegify vorliegt.

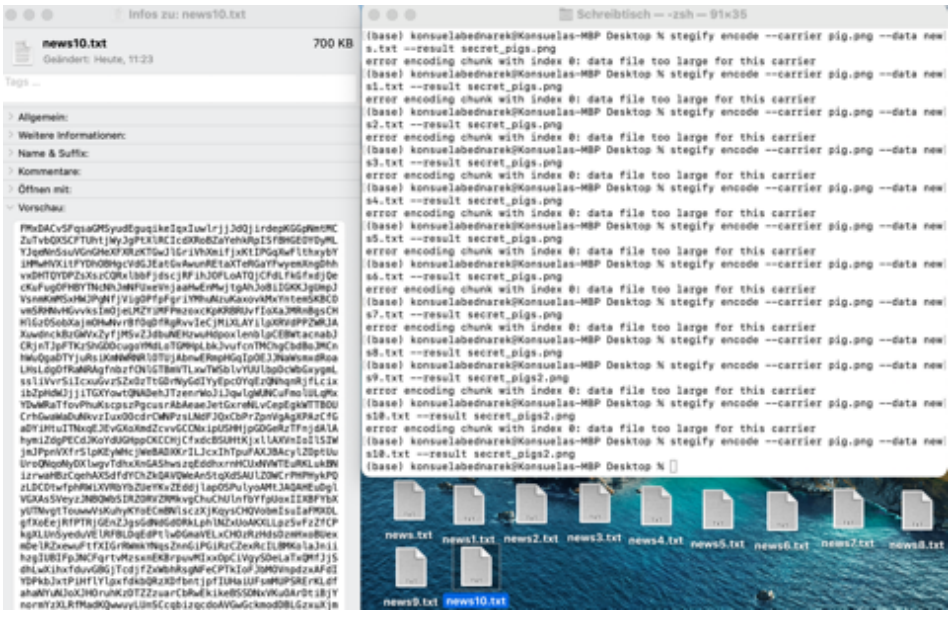

Abbildung 99 – Stegify - Verschlüsselung größerer Dateien

Anschließend werden die PSNR-, MSE und SSIM-Werte berechnet.

```
bild1 = imread('pig.png');
bild2 = imread('secret_pigs.png');

psnr_value = psnr(bild1, bild2);
fprintf('PSNR-Wert: %0.6f dB\n', psnr_value);
PSNR-Wert: 46.289152 dB
>>
bild1 = imread('pig.png');
bild2 = imread('secret_pigs2.png');

psnr_value = psnr(bild1, bild2);
fprintf('PSNR-Wert: %0.6f dB\n', psnr_value);
PSNR-Wert: 44.754883 dB
>>
bild1 = imread('pig.png');
bild2 = imread('secret_pigs.png');

mse_value = immse(bild1, bild2);
fprintf('MSE-Wert: %0.6f\n', mse_value);
MSE-Wert: 1.556558
>>
bild1 = imread('pig.png');
bild2 = imread('secret_pigs2.png');

mse_value = immse(bild1, bild2);
fprintf('MSE-Wert: %0.6f\n', mse_value);
MSE-Wert: 2.175705
>>

bild1 = imread('pig.png');
bild2 = imread('secret_pigs.png');

ssim_value = ssim(bild1, bild2);
fprintf('SSIM-Wert: %0.6f\n', ssim_value);
SSIM-Wert: 0.996126
>>
bild1 = imread('pig.png');
bild2 = imread('secret_pigs2.png');

ssim_value = ssim(bild1, bild2);
fprintf('SSIM-Wert: %0.6f\n', ssim_value);
SSIM-Wert: 0.995313
>>
```

Abbildung 100 – Stegify - Berechnung PSRN-, MSE- und SSIM-Werte

Daraufhin wird das Werkzeug Stegano verwendet, um die größeren Textdateien in das Bild pig.png einzubetten. Dazu wird der Befehl *stegano hide --data news.txt --in pig.png --out pigs_secret.png* ausgeführt. Da alle erstellten Dateien zu einem Fehler führen, wurden weitere kleinere Dateien erzeugt und angewendet. Die größte Datei news14.txt, die eingebettet werden konnte, hat eine Größe von 400 KB. Eine weitere kleinere Datei news11.txt mit 100 KB wurde ebenfalls eingebettet. Das neu erstellte Bild mit der Datei news11.txt heißt pigs_secret.png und das neu erstellte Bild mit der Datei news14.txt lautet pigs_secret2.png.

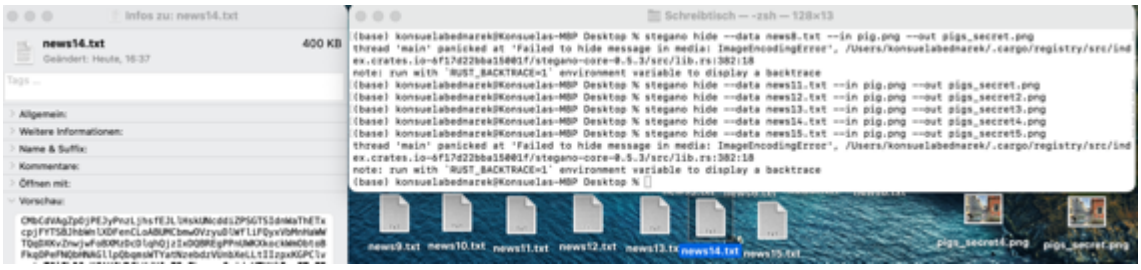

Abbildung 101 Stegano - Verschlüsselung größerer Dateien

Bei beiden Bildern werden die PSNR-, MSE- sowie SSIM-Werte berechnet.

```
bild1 = imread('pig.png');
bild2 = imread('pigs_secret.png');

psnr_value = psnr(bild1, bild2);
fprintf('PSNR-Wert: %0.6f dB\n', psnr_value);
PSNR-Wert: 57.966037 dB
>>
bild1 = imread('pig.png');
bild2 = imread('pigs_secret4.png');

psnr_value = psnr(bild1, bild2);
fprintf('PSNR-Wert: %0.6f dB\n', psnr_value);
PSNR-Wert: 51.956352 dB
>>
bild1 = imread('pig.png');
bild2 = imread('pigs_secret.png');

mse_value = immse(bild1, bild2);
fprintf('MSE-Wert: %0.6f\n', mse_value);
MSE-Wert: 0.103867
>>
bild1 = imread('pig.png');
bild2 = imread('pigs_secret4.png');

mse_value = immse(bild1, bild2);
fprintf('MSE-Wert: %0.6f\n', mse_value);
MSE-Wert: 0.414424
>>
bild1 = imread('pig.png');
bild2 = imread('pigs_secret.png');

ssim_value = ssim(bild1, bild2);
fprintf('SSIM-Wert: %0.6f\n', ssim_value);
SSIM-Wert: 0.999638
>>
bild1 = imread('pig.png');
bild2 = imread('pigs_secret4.png');

ssim_value = ssim(bild1, bild2);
fprintf('SSIM-Wert: %0.6f\n', ssim_value);
SSIM-Wert: 0.999026
>>
```

Abbildung 102 - Stegano - Berechnung PSRN-, MSE- und SSIM-Werte

Die Ergebnisse der PSNR-Werte zeigen, dass beide manipulierten Bilder kleinere PSNR-Werte haben. Je größer die zu verschlüsselte Datei war, desto geringer ist der PSNR-Wert. In der folgenden Tabelle werden die Werte abgebildet.

Werkzeuge	Bilder	eingebettete Dateigröße	PSNR	MSE	SSIM
Stegify	secret_pigs.png	500 KB	46,21 dB	1,556558	0,996126
	secret_pigs2.png	700 KB	44,75 dB	2,175705	0,995313
Stegano	pigs_secret.png	100 KB	57,97 dB	0,103867	0,999638
	pigs_secret4.png	400 KB	51,96 dB	0,414424	0,999026

Tabelle 7 - Übersicht PSNR-, MSE- und SSIM-Werte

Ein linearer Zusammenhang lässt sich nicht erkennen. Da die Berechnung der PSNR-Werte anhand einer logarithmischen Funktion berechnet wird, kann

bestätigt werden, dass größere eingebettete Dateien, den PSNR-Wert reduzieren. Der Logarithmus ist eine nicht-lineare Funktion und führt dazu, dass kleine Änderungen in den MSE-Werten bei hohen PSNR-Werten größere Auswirkungen als bei niedrigeren PSNR-Werten haben. Eine Verringerung des MSE-Werts führt zu einem größeren Anstieg des PSNR-Werts. Eine Verringerung des PSNR-Werts (z. B. um 10 dB) stellt eine Verschlechterung der Bildqualität dar. Da ein Anhang, integriert mit einem maliziösen Code, zusätzliche Informationen in ein Bild einfügt, kann dies zu einem niedrigeren PSNR-Wert und somit zu einer Verringerung der Bildqualität führen. Zusammenfassend kann bestätigt werden, dass Bilder, die mit größeren Dateien eingebettet werden, zu einer größeren Verzerrung der Bilder, damit zu einer Verschlechterung der Bildqualität und der Reduzierung der PSNR-Werte führt.

6.6 Auswertung der Werkzeuge

In Kapitel 6 wurden die ausgewählten steganografischen Werkzeuge Steghide, Stegosuite, OutGuess, Zsteg, Tater, PicStealth, f2d, SSuite Picsel Security, Stegano, Stegify, The Hider sowie East-tec InvisibleSecrets vorgestellt, analysiert und miteinander verglichen. Auf dem Betriebssystem Linux können die Werkzeuge Steghide, Stegosuite, OutGuess sowie Zsteg genutzt werden. Steghide ist ebenfalls für Windows verwendbar. Tater, PicStealth, f2d, SSuite Picsel und East-tec InvisibleSecrets sind nur auf dem Windows-System ausführbar. Stegano lässt sich im Gegensatz dazu auf dem MacOS, Windows sowie Linux-Betriebssystem nutzen und auch Stegify kann auf dem MacOS und auf dem Linux verwendet werden. The Hider kann nur auf dem MacOS-Betriebssystem genutzt werden. Sowohl East-tec InvisibleSecrets und The Hider sind kostenpflichtige Werkzeuge, alle weiteren vorgestellten Werkzeuge sind kostenlos. Da bei East-tec InvisibleSecrets keine PSNR-Werte vorliegen, jedoch niedrige MSE sowie sehr gute SSIM-Werte und bei The Hider liegen hohe PSNR-Werte, ebenfalls niedrige MSE sowie sehr gute SSIM-Werte vor, kann grundsätzlich bestätigt werden, dass die kostenpflichtigen Werkzeuge zu guten steganografischen Werkzeugen gezählt werden. Im Gegensatz dazu ist das Werkzeug mit den schwächsten Werten der objektiven Gütekriterien OutGuess, welches kostenlos angeboten wird. Die meisten Werkzeuge bieten

Dokumentationen an, die den Benutzern eine gute Hilfestellung geben. Wenige Dokumentation lagen nur bei Tater, PicStealth, f2d und The Hider vor. Die Werkzeuge Stegosuite, Tater, PicStealth, f2d, SSuite Picsel Security, The Hider und East-tec InvisibleSecrets nutzen eine grafische Benutzeroberfläche. Der Rest der Werkzeuge nutzt das Command-Line-Interface. Aus diesem Merkmal könnte grundsätzlich bestätigt werden, dass Werkzeuge, welche über das Commad-Line-Interface genutzt werden, bessere Werte erzielen, denn zu den besseren Werkzeugen gehören Stegify sowie Stegano. Zum Kriterium der Dateigröße ist aufgefallen, dass nur bei East-tec InvisibleSecrets keine Änderung in der Dateigröße vorgenommen wurde, während bei allen anderen Werkzeugen die Dateigröße entweder verkleinert oder vergrößert wurde. Zuletzt ist zu erwähnen, dass die MSE-, PSNR- und SSIM-Werte eine gute Vergleichbarkeit der Werkzeuge beschreiben. Da die Werkzeuge Zsteg und East-tec InvisibleSecrets nicht gewertet werden können, kann hervorgehoben werden, dass Stegify und Stegano zu den besseren steganografischen Werkzeugen gehören. Auch nach der Einbettung von größeren Dateien, die einen möglichen Hinweis geben können auf einen möglichen Schadcode, können die Werte der Werkezeuge die größere Manipulation erkennen, da die Bildqualität verschlechtert wird.

7 Praktischer Test der LSB-Methode und Anwendung der Werkzeuge

In Kapitel 5 wurde die theoretische Grundlage der LSB-Methode vorgestellt und im Anschluss daran wurden die benötigten steganografischen Werkzeuge untersucht. Daher wird in den folgenden Kapiteln ein ausgewähltes Bild mit einer Geheimnachricht mit der LSB-Methode verschlüsselt und mit Hilfe der steganografischen Werkzeuge versucht zu entschlüsseln. Dieser praktische Test vermittelt ein tieferes Verständnis für die Wirksamkeit der LSB-Methode sowie der verwendeten Werkzeuge.

7.1 Bildverschlüsselung mit Python

In diesem Abschnitt wird ein ausgewähltes Bild mittels des bereits beschriebenen Python-Codes mit einer geheimen Nachricht verschlüsselt. Das ausgewählte Bild heißt „dog“ und sieht wie folgt aus:

Abbildung 103 - Bild "dog" ohne Geheimnachricht [62]

Bei dem Bild handelt es sich um ein PNG-Format. Für die LSB-Methode eignen sich Bilder mit verlustfreier Kompression, also PNG oder GIF. Komprimierte Formate wie z. B. JPEG sollen nicht verwendet werden, da die LSB-Bits während der Komprimierung verändert werden können. [30] [63] Die Botschaft, die

verschlüsselt wird, lautet: „Wismar“.

```python
import numpy as np
from PIL import Image

message = "Wismar"

b_message = ''.join(["{:08b}".format(ord(x)) for x in message ])
b_message = [int(x) for x in b_message]

b_message_lenght = len(b_message)

with Image.open("dog.png") as img:
    width, height = img.size
    data = np.array(img)

data = np.reshape(data, width*height*3)

data[:b_message_lenght] = data[:b_message_lenght] & ~1 | b_message

data = np.reshape(data, (height, width, 3))

new_img = Image.fromarray(data)
new_img.save("dog-new.png")
new_img.show()
```

Abbildung 104 - Anwendung LSB-Code auf das Bild "dog.png"

Das Bild „dog.png“ wird zu „dog-new.png“, welches die Geheimnachricht „Wismar“ in sich trägt. Das Bild „dog-new.png“ sieht wie folgt aus:

Abbildung 105 - Bild "dog-new.png" mit Geheimnachricht "Wismar"

Die Bilder werden parallel betrachtet und auf Unterschiede überprüft:

Abbildung 106 - linkes Bild "dog.png" und rechtes Bild "dog-new.png"

Beim Betrachten beider Bilder lassen sich keine Unterschiede feststellen. Um sicherzugehen, dass der Python-Code funktioniert, werden drei Tests gemacht, die beweisen sollen, dass die LSB-Methode funktioniert hat. Der erste Test erfolgt mittels der Webanwendung „redketchup“, die auf folgender Webseite erreicht wird. Diese Seite ermöglicht punktuelle Farbwerte auszuwählen und deren Farbe zu bestimmen.

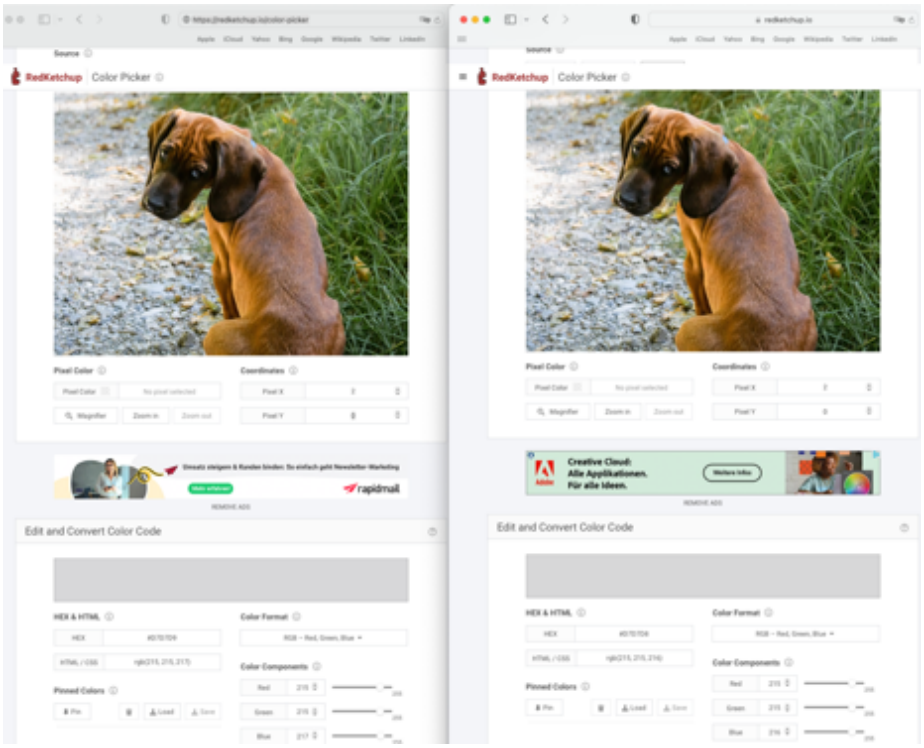

Abbildung 107 - Vergleich Pixelwerte (2,0) von "dog.png" (links) und "dog-new.png" (rechts) [64]

Bei den Bildern „dog.png“ und „dog-new.png“ wird der Pixelwert auf der Koordinate (2,0) zufällig ausgewählt. Bei „dog.png“ lautet der RGB-Wert (215, 215, 217) und bei „dog-new.png“ lautet der RGB-Wert (215, 215, 216). Damit wäre bereits eine Veränderung der RGB-Werte feststellbar. Folglich wird ein zweiter Test gemacht. In diesem Fall wird ein Python-Code geschrieben, der die RGB-Werte manuell bestimmt. Wie in dem folgenden Test werden die Pixelwerte (2,0) ausgewählt und die RGB-Werte ausgegeben. [65]

Abbildung 108 - RGB-Werte (2,0) von "dog.png"

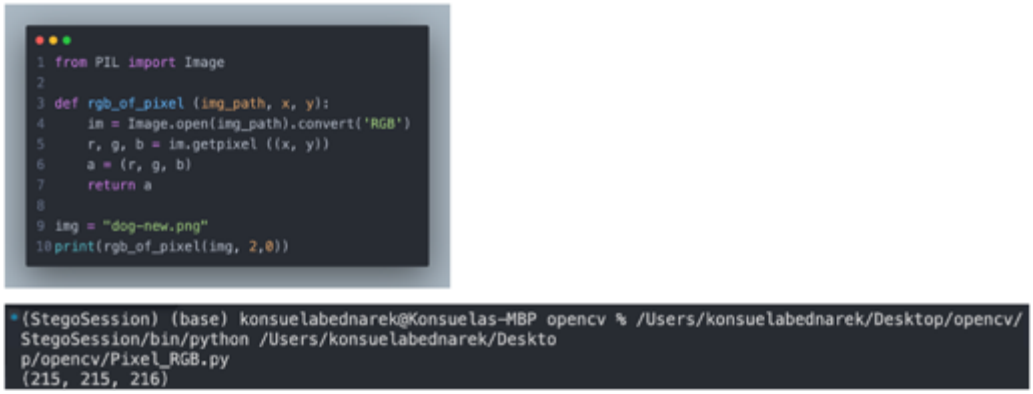

Abbildung 109 - RGB-Werte (2,0) von "dog-new.png"

Als Ergebnis wird für die Koordinaten (2,0) „dog.png“ die RGB-Werte (215, 215, 217) und für „dog-new.png“ die RGB-Werte (215, 215, 216) aufgelistet. Dies beweist, dass wie beim ersten Test, eine Veränderung der RGB-Werte bei „dog-new.png“ erfolgte. Eine manuelle Auswahl von zufälligen Pixeln erfordert jedoch viel Zeit. Daher wird in dem dritten und letzten Test ein Python-Code geschrieben, der beide Bilder vergleicht und überprüft, ob Pixelwerte verändert

wurden. Sollten Pixelwerte verändert worden sein, werden diese aufgelistet:

```python
from PIL import Image

def compare_images(image1_path, image2_path):
    image1 = Image.open(image1_path)
    image2 = Image.open(image2_path)

    pixels1 = list(image1.getdata())
    pixels2 = list(image2.getdata())

    width, height = image1.size
    changed_pixels = []

    for i in range(len(pixels1)):
        pixel1 = pixels1[i]
        pixel2 = pixels2[i]

        rgb1 = pixel1[:3]
        rgb2 = pixel2[:3]

        if rgb1 != rgb2:
            x = i % width
            y = i // width
            changed_pixels.append(((x, y), rgb1, rgb2))

    return changed_pixels

image1_path = "dog.png"
image2_path = "dog-new.png"

changed_pixels = compare_images(image1_path, image2_path)
changed_pixel_count = len(changed_pixels)

if changed_pixel_count > 0:
    print("Es wurden", changed_pixel_count, "Pixelwerte geändert:")

    for coordinates, rgb1, rgb2 in changed_pixels:
        x, y = coordinates

        print("Koordinaten:", (x, y))
        print("Bild 1: RGB =", rgb1)
        print("Bild 2: RGB =", rgb2)
        print()

else:
    print("Es wurden keine veränderten Pixelwerte gefunden.")
```

Abbildung 110 - Vergleich beider Bilder auf veränderte Pixelwerte [66] [67] [68]

Als Ergebnis wird bestätigt, dass 16 Pixelwerte geändert wurden. Zudem wird aufgelistet, welche Pixelwerte betroffen sind, wie die RGB-Werte zuvor waren und wie sie nach der Integration der Geheimnachricht aussehen.

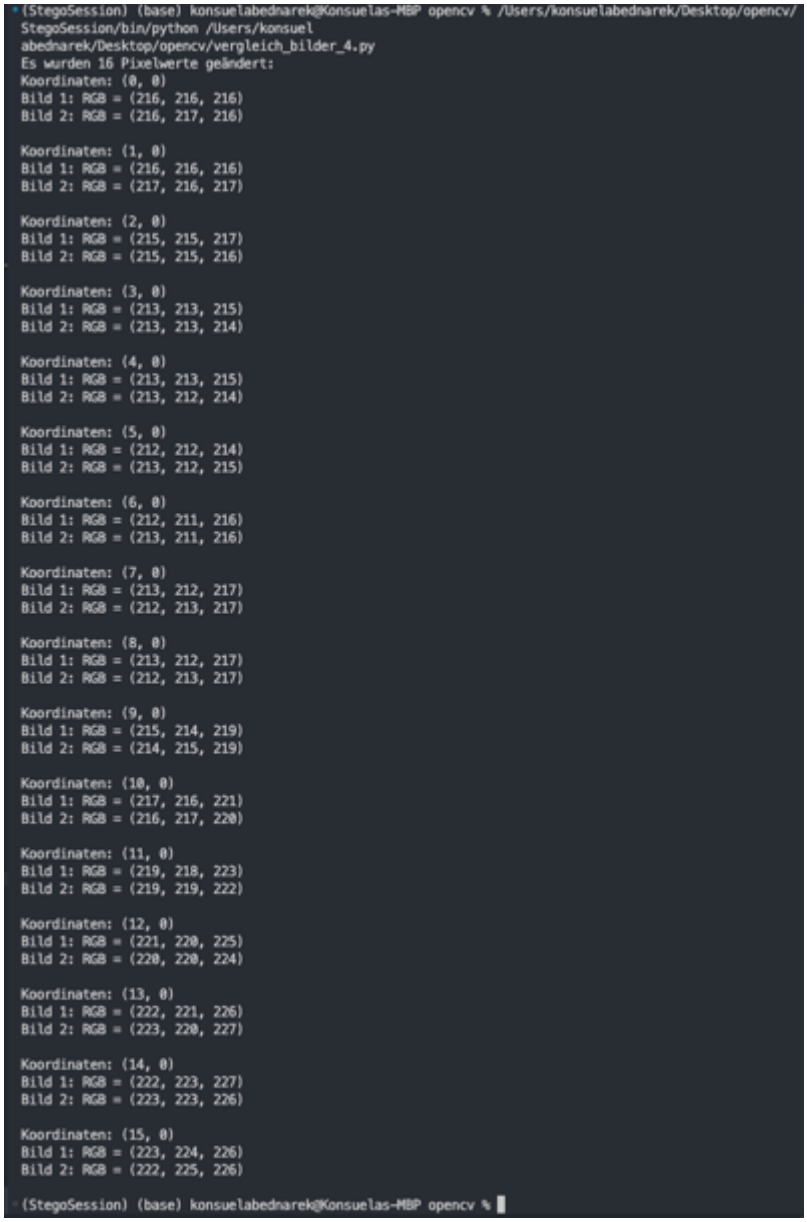

Abbildung 111 - Aufzählung der veränderten Pixelwerte

Der Code liefert das Ergebnis, dass 16 Pixelwerte geändert wurden. Aufgrund dessen wird eine weitere Untersuchung gemacht, die beweisen soll, dass „Wismar“ als Geheimnachricht verschlüsselt und im Bild „dog-new.png“ eingebettet wurde. „Wismar“ wird im Binärcode aufgeschrieben: 01010111011010010111001101101101011000010111001O. [69] Anschließend wird eine Tabelle aufgelistet, in welcher die RGB-Werte von „dog.png“ ohne Geheimnachricht und die RGB-Werte von „dog-new.png“ mit Geheimnachricht „Wismar“ aufgeschrieben sind.

Koordinaten	Bild „dog.png“ ohne Geheimbotschaft	Bild „dog-new.png“ mit Geheimbotschaft „Wismar“
Koordinate (0,0)	RGB (216, 216, 216) 216 11011000 216 11011000 216 11011000	RGB (216, 217, 216) 216 11011000 217 11011001 216 11011000
Koordinate (1,0)	RGB (216, 216, 216) 216 11011000 216 11011000 216 11011000	RGB (217, 216, 217) 217 11011001 216 11011000 217 11011001
Koordinate (2,0)	RGB (215, 215, 217) 215 11010111 215 11010111 217 11011001	RGB (215, 215, 216) 215 11010111 215 11010111 216 11011000
Koordinate (3,0)	RGB (213, 213, 215) 213 11010101 213 11010101 215 11010111	RGB (213, 213, 214) 213 11010101 213 11010101 214 11010110
Koordinate (4,0)	RGB (213, 213, 215) 213 11010101 213 11010101 215 11010111	RGB (213, 212, 214) 213 11010101 212 11010100 214 11010110
Koordinate (5,0)	RGB (212, 212, 214) 212 11010100 212 11010100 214 11010110	RGB (213, 212, 215) 213 11010101 212 11010100 215 11010111
Koordinate (6,0)	RGB (212, 211, 216) 212 11010100 211 11010011 216 11011000	RGB (213, 211, 216) 213 11010101 211 11010011 216 11011000
Koordinate (7,0)	RGB (213, 212, 217) 213 11010101 212 11010100 217 11011001	RGB (212, 213, 217) 212 11010100 213 11010101 217 11011001
Koordinate (8,0)	RGB (213, 212, 217) 213 11010101 212 11010100 217 11011001	RGB (212, 213, 217) 212 11010100 213 11010101 217 11011001
Koordinate (9,0)	RGB (215, 214, 219) 215 11010111 214 11010110 219 11011011	RGB (214, 215, 219) 214 11010110 215 11010111 219 11011011
Koordinate (10,0)	RGB (217, 216, 221) 217 11011001 216 11011000 221 11011101	RGB (216, 217, 220) 216 11011000 217 11011001 220 11011100
Koordinate (11,0)	RGB (219, 218, 223) 219 11011011 218 11011010 223 11011111	RGB (219, 219, 222) 219 11011011 219 11011011 222 11011110
Koordinate (12,0)	RGB (221, 220, 225) 221 11011101 220 11011100 225 11100001	RGB (220, 220, 224) 220 11011100 220 11011100 224 11100000
Koordinate (13,0)	RGB (222, 221, 226) 222 11011110 221 11011101 226 11100010	RGB (223, 220, 227) 223 11011111 220 11011100 227 11100011
Koordinate (14,0)	RGB (222, 223, 227) 222 11011110 223 11011111 227 11100011	RGB (223, 223, 226) 223 11011111 223 11011111 226 11100010
Koordinate (15,0)	RGB (223, 224, 226) 223 11011111 224 11100000 226 11100010	RGB (222, 225, 226) 222 11011110 225 11100001 226 11100010

Tabelle 8 - Übersicht der veränderten Pixelwerte [69]

Daraufhin werden farblich die letzten Bits von „dog.new.png“ hervorgehoben. Zu erkennen ist, dass diese tatsächlich für den Binäre-Code „Wismar“ stehen:
01010111011010010111001101101101011000010111001 0

Farbliche Hervorhebung:
01010111 (W)
01101001 (i)
01110011 (s)
01101101 (m)
01100001 (a)
01110010 (r)

Damit sind alle Tests erfolgreich verlaufen. Anhand dieser kann bestätigt werden, dass im „dog-new.png“ die Geheimnachricht „Wismar“ integriert ist.

7.2 Entschlüsselung mithilfe steganografischer Werkzeuge

In Kapitel 6 wurden die steganografischen Werkzeuge Steghide, Stegosuite, OutGuess, Zsteg, Tater, PicStealth, f2d, SSuite Picsel Encryption, Stegano, Stegify, The Hider und East-tec InvisibleSecrets vorgestellt. Da bereits Bewertungskriterien vorgenommen wurden, ist das Kriterium der Interoperabilität ebenfalls interessant. Kann ein steganografisches Werkzeug erkennen, dass ein Bild manipuliert ist, wenn es mit der vorgestellten LSB-Methode verschlüsselt wurde? Das Bild dog-new.png wurde mit dem Pythons-Codes mittels der LSB-Methode verschlüsselt. Nachfolgend wird jedes vorgestellte steganografische Werkzeug aufgerufen, das manipulierte Bild hochgeladen und überprüft, ob das jeweilige Werkzeug, die Geheimbotschaft „Wismar“ herauslesen kann oder ob das Werkzeug nur eigene Verschlüsselungen zulässt.
Zuerst wird Steghide angeschaut. Steghide unterstützt nur die Formate JPEG, BMP, WAV sowie AU. Da das verschlüsselte Bild ein PNG-Format ist und es kein Passwort für das Bild existiert, ist eine Entschlüsselung mit diesem Werkzeug nicht möglich. Dies kann mit dem Befehl *steghide info dog-new.png* überprüft werden. Das Ergebnis bestätigt, dass das Format nicht unterstützt wird. Steghide ist daher nicht interoperabel mit diesem LSB-Code.

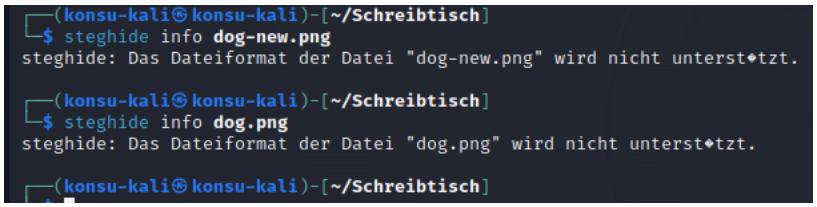

Abbildung 112 - Steghide - Fehlerhinweis

Danach wird das Werkzeug Stegosuite mit dem Befehl *stegosuite gui* aufgerufen. Die Datei dog-new.png wird hochgeladen und versucht zu extrahiert. Auf der Kommandozeile wird der Fehler angezeigt, dass eine Extraktion nicht möglich ist. Dementsprechend kann eine Interoperabilität bei Stegosuite mit diesem LSB-Code nicht festgestellt werden.

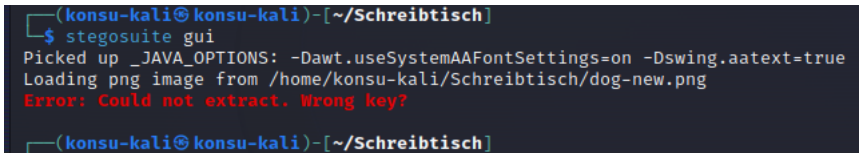

Abbildung 113 - Stegosuite – Fehlerhinweis

Daraufhin wird versucht, mit dem Werkzeug OutGuess das Bild zu entschlüsseln. In Kapitel 6.2.6 wurde bereits beschrieben, dass OutGuess lediglich JPEG, PPM und PNM als Formate unterstützt. Der Befehl *outguess -r dog-new.png text.txt* bestätigt, dass das Format PNG nicht unterstützt wird. Eine Operabilität mit diesem LSB-Code ist daher bei OutGuess ebenfalls nicht zu bestätigen.

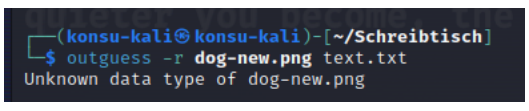

Abbildung 114 - OutGuess - Fehlerhinweis

Anschließend wird das Werkzeug Zsteg aufgerufen und versucht die Botschaft zu extrahieren. Mit dem Befehl *zsteg dog-new.png* wird zuerst überprüft, ob das Werkzeug eine Geheimbotschaft entdeckt. Das Ergebnis zeigt keine versteckte Botschaft an. Zudem zeigt der Verbose-Modus (*zsteg -v dog-news.png*) keine

Geheimnachricht im Bild an. Daher besteht bei Zsteg, wie bei den anderen Werkzeugen, keine Interoperabilität bei der angewendeten LSB-Methode.

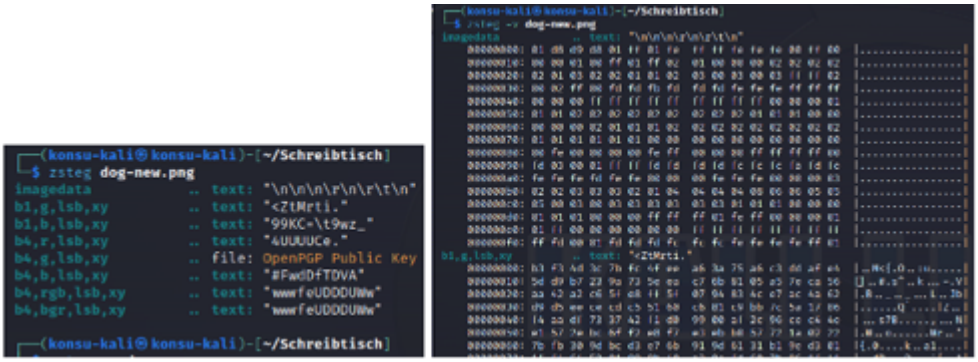

Abbildung 115 - Zsteg - kein Hinweis auf eine versteckte Nachricht

Des Weiteren erfolgt eine weitere Analyse mit dem Werkzeug Tater. Tater wird aufgerufen und das Bild dog-new.png hochgeladen, sodass Tater direkt überprüft, ob eine Geheimnachricht im Bild versteckt ist. Tater erkennt jedoch keine versteckte Nachricht und ist somit ebenfalls nicht interoperabel mit dem LSB-Code.

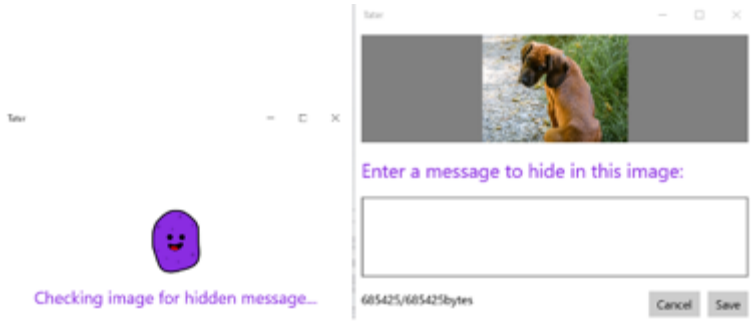

Abbildung 116 - Tater - kein Hinweis auf versteckte Botschaft

Darauffolgend wird das Werkzeug PicStealth aufgerufen und kontrolliert, ob dieses Werkzeug die versteckte Nachricht entschlüsseln kann. Das Bild dog-new.png wird hochgeladen und danach wird versucht die Nachricht zu extrahieren. In dem Textfeld erscheint eine Abfolge von Sonderzeichen, sodass die versteckte Nachricht nicht erkennbar ist. Damit ist festgestellt, dass PicStealth keine Interoperabilität bei der Verschlüsselung mit dem durchgeführten LSB-Code unterstützt.

Abbildung 117 - PicStealth - Fehlertext bzw. Sonderzeichen

Als nächstes Werkzeug wird f2d untersucht. Das Bild dog-new.png wird hochgeladen und ebenfalls versucht zu dechiffrieren. Das Werkzeug fordert direkt auf, ein Passwort einzutragen. Da kein Passwort verwendet wurde, wird ein simples 1234-Passwort eingetragen. Das Ergebnis zeigt, dass das Bild nicht gelesen werden kann, da das Bild nicht von f2d verschlüsselt wurde. F2d ist daher nicht interoperabel mit diesem LSB-Code.

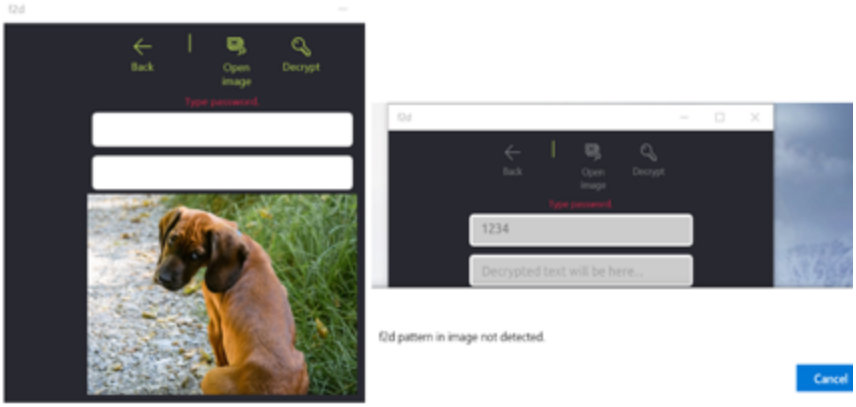

Abbildung 118 - f2d - Fehlerhinweis

Das nächste steganografische Werkzeug SSuite Picsel Encryption wird aufgerufen. Wie in Kapitel 6.2.17 beschrieben, wird zuerst das originale Bild dog.png ausgewählt, „Decrypt Image“ angeklickt und das manipulierte Bild dog-new.png ausgewählt, um dies zu entschlüsseln. Im Textfeld erscheinen

Sonderzeichen und nicht die versteckte Botschaft. Ssuite Picsel Encryption ist somit ebenfalls nicht interoperabel mit der durchgeführten LSB-Verschlüsselung.

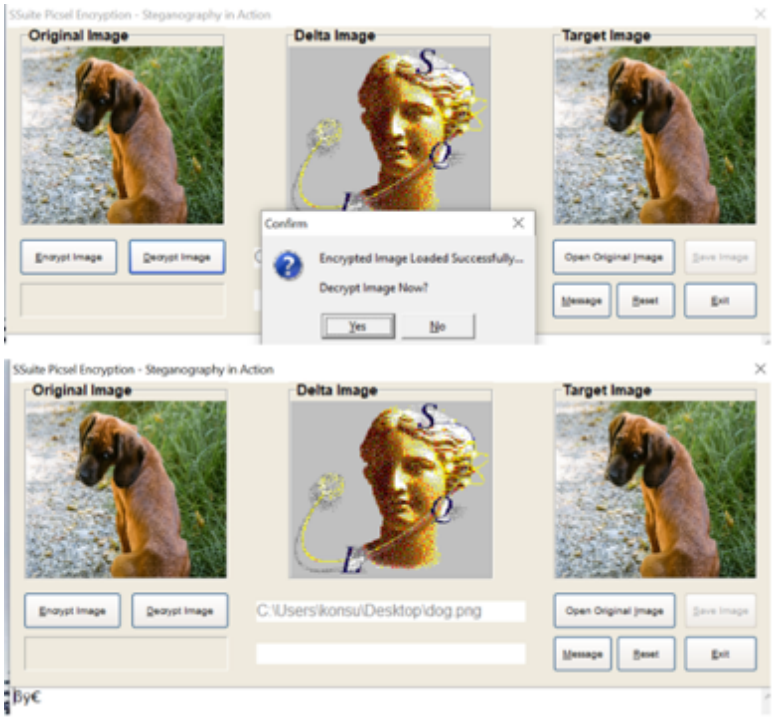

Abbildung 119 - SSuite Picsel Security - Fehlertext

Daraufhin wird das Werkzeug Stegano verwendet, um die versteckte Botschaft aus dow-new.png zu extrahieren. Der Befehl *stegano unveil --in dog-new.png --out Bachelor-Thesis* wird ausgeführt. Als Ergebnis liefert Stegano eine Fehlermeldung. Das manipulierte Bild kann nicht extrahiert werden, jedoch gibt es einen Hinweis mit dem Text: „*Seems like you've got an invalid stegano* file“, dass eine ungültige steganografische Datei vorliegt. Dies bestätigt, dass Stegano nur seine eigenen manipulierten Bilder extrahieren kann und somit keine Interoperabilität bei der LSB-Verschlüsselung mittels des LSB-Codes besteht.

```
(base) konsuelabednarek@Konsuelas-MBP Desktop % stegano unveil --in dog-new.png --out Bachelor-Thesis
thread 'main' panicked at 'Seems like you've got an invalid stegano file', /Users/konsuelabednarek/.cargo/
registry/src/index.crates.io-6f17d22bba15001f/stegano-core-0.5.3/src/message.rs:52:17
note: run with `RUST_BACKTRACE=1` environment variable to display a backtrace
(base) konsuelabednarek@Konsuelas-MBP Desktop %
```

Abbildung 120 - Stegano - Fehlermeldung bei der Extraktion

Anschließend wird das Werkzeug Stegify genutzt, um zu überprüfen, ob das Bild dog-new.png eine versteckte Botschaft beinhaltet. Der Befehl *stegify decode --carrier dog-new.png --result dognews.txt* wird ausgeführt und eine Datei dognews.txt wird erstellt. Beim Öffnen dieser Datei ist offensichtlich, dass Stegify das manipulierte Bild mittels Python-Code nicht erkennt.

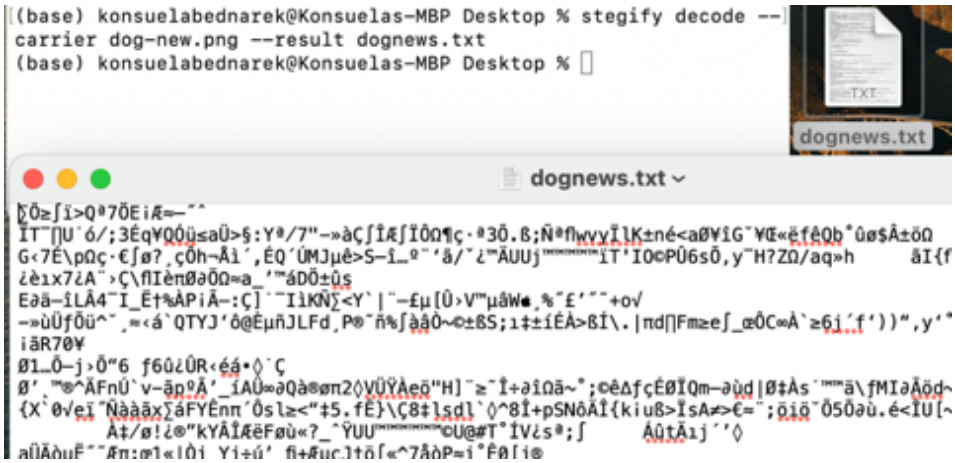

Abbildung 121 - Stegify - Extraktion und fehlerhafte txt-Datei

Als nächstes Werkzeug wird The Hider verwendet. Das Bild dog.png wird geöffnet und direkt nach dem Versuch das Bild zu entschlüsseln erscheint der Hinweis, dass dog.png kein vom The Hider-Programm verschlüsseltes Bild ist. Wie bereits bei den anderen Werkzeugen, ist eine Interoperabilität bei der Verwendung des LSB-Codes nicht möglich.

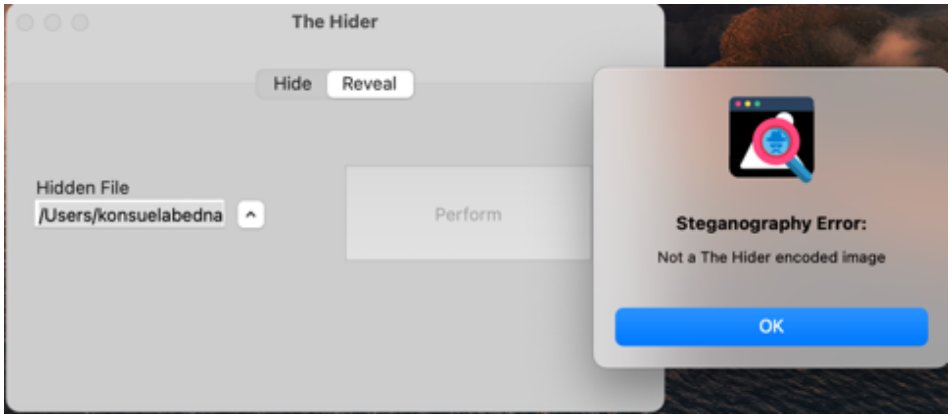

Abbildung 122 - The Hider - Fehlermeldung

Das letzte steganografische Werkzeug ist East-tec InvisibleSecrets. Mit diesem wird ein letzter Versuch gestartet, das Bild dog-new.png zu entschlüsseln. Das Bild dog-new.png wird geöffnet. Daraufhin muss ein Passwort für das Entschlüsseln gesetzt werden, da kein Passwort existiert, wird ein simples 1234-Passwort eingetragen und versucht die nächste Maske aufzurufen. East-tec erkennt direkt, dass das Passwort nicht richtig ist und verhindert den weiteren Ablauf. In der Beschreibung von East-tec wird hingewiesen, dass nur Bilder entschlüsselt werden können, die mit East-tec verschlüsselt wurden. Damit ist eine Interoperabilität bei East-tec mittels der LSB-Codierung nicht möglich.

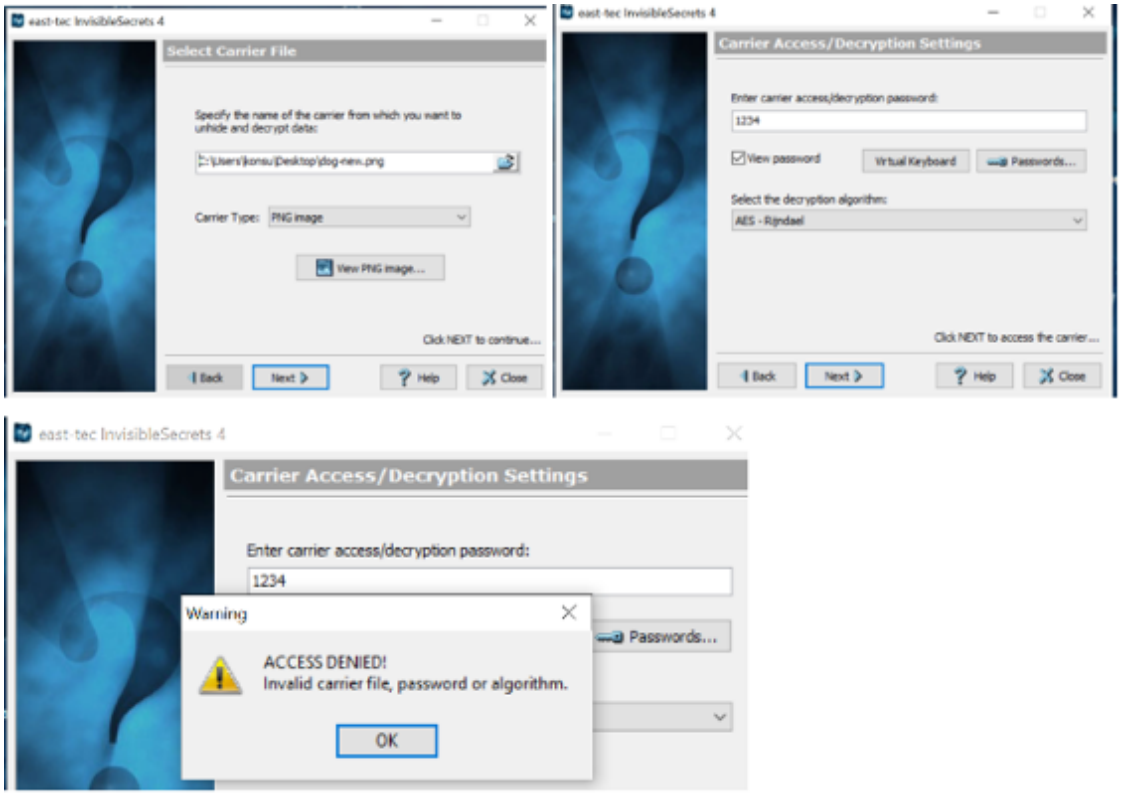

Abbildung 123 - East-tec – Fehlerhinweis

7.3 Überprüfung und Vergleich der Ergebnisse

In dem vorgestellten praktischen Test wurde versucht, mit den zuvor vorgestellten steganografischen Werkzeug, das verschlüsselten Bild dog-new.png zu entschlüsseln. Die Ergebnisse der Tests der jeweiligen Werkzeuge waren gleich. Keins der vorgestellten steganografischen Werkzeuge hat einen Hinweis auf die versteckte Botschaft in dem Bild dog-new.png geliefert. Das Bild

dog.png welches mittels Python-Code und der LSB-Methode verschlüsselt wurde, kann daher nicht mit einem anderen steganografischen Werkzeug erkannt werden. Zum einen liegt die fehlende Interoperabilität an den Dateiformaten, da das manipulierte Bild im PNG-Format vorlag und Werkzeuge wie z. B. Steghide und OutGuess kein PNG-Format unterstützen. Zum anderen liegt es an Werkzeugen, die eine Verschlüsselung mittels eines Passworts durchführen und somit für das Entschlüsseln des versteckten Bildes daher zum Entschlüsseln ein Passwort benötigt haben, wie z. B. bei den Werkzeugen f2d und East-tec. Als abschließende Analyse wird bewertet, dass keine Interoperabilität bei den vorgestellten steganografischen Werkzeugen mit dem vorgestellten LSB-Code besteht. Die Ver- sowie Entschlüsselung von Bildern mit den jeweiligen Werkzeugen funktionieren daher mit den jeweiligen Werkzeugen selbst (außer Zsteg, da keine Verschlüsselung möglich ist). Dieses Ergebnis unterstreicht die Effektivität der gewählten LSB-Methode bei der verborgenen Übertragung von Informationen.

8 Fazit und Ausblick

Die vorliegende Bachelor-Thesis beschäftigt sich mit der Analyse von Werkzeugen zur steganografischen Untersuchung und der Bewertung von Steganografie zur Möglichkeit der verborgenen Übertragung von Informationen. Um die Abgrenzung zur Steganografie zu erläutern, wurden die Grundlagen der Kryptografie vorgestellt. Daraufhin erfolgte eine Einführung in die Grundlagen der Bildverarbeitung und danach wurden die steganografischen Methoden beschrieben. Insbesondere wurde die LSB-Methode erklärt und anhand eines Python-Codes vorgestellt sowie erläutert. Im Hauptteil der Bachelor-Thesis wurden ausgewählte Werkzeuge vorgestellt. Dabei wurden Bilder mit den jeweiligen Werkzeugen ver- sowie entschlüsselt. Des Weiteren wurden die Werkzeuge mithilfe der objektiven Gütekriterien verglichen. Dabei wurde ebenfalls der Fall betrachtet, dass eine größere Datei z. B. als Hinweis für einen Schadcode, mit zwei Werkzeugen verschlüsselt wurde, welche hohe PSNR-Werte hatten. Anschließend konnte analysiert werden, ob sich die PSNR-Werte verändert hatten, wodurch der Nachweis erbracht wurde, dass niedrige PSNR-Werte einen Rückschlüss auf versteckte Dateien zulassen können. Die Ergebnisse zeigen, dass die jeweiligen Werkzeuge für sich betrachtet sowohl Bilder ver- und entschlüsseln konnten. Anhand der objektiven Gütekriterien kann eine gute Übersicht gegeben werden, welche steganografischen Werkzeuge sich besser für eine verborgenen Übertragung von Informationen eignen. Diese Bewertung konnte anhand der PSNR-, MSE- sowie SSIM-Werte vorgenommen werden. Durch diese Bewertung konnten Stegify und Stegano als sehr gute Werkzeuge für die Übermittlung sowie für das Erkennen von der geheimen Übertragung von Informationen präsentiert werden, da sie eine sehr hohe Dateiqualität aufweisen konnten. Nach einer Einbettung von größeren Dateien bei Stegano und Stegify konnte ebenfalls erläutert werden, dass die PSNR-, MSE- sowie SSIM-Wert zu einer schlechteren Bewertung führt, was einen Rückschluss darauf gibt, dass die Bildqualität verringert wird. Schlussfolgernd kann festgehalten werden, dass eine Übermittlung von Schadcode über die Werte erkannt werden kann. Nachfolgend wurde ein Bild mit der vorgestellten LSB-Methode mittels des Python-Codes verschlüsselt und mit den ausgewählten

Werkzeugen versucht zu entschlüsseln. Die Ergebnisse waren einheitlich, da keins der ausgewählten Werkzeuge einen Hinweis auf die versteckte Information liefern konnte, welches zuvor mit der LSB-Methode verschlüsselt wurde. Zwei konkrete Gründe konnten einen Rückschluss geben, wieso eine Entschlüsselung nicht funktioniert hatte. Zum einen liegt es an den Dateiformaten, da das erstellte Bild im PNG-Format vorlag und Werkzeuge wie Steghide und OutGuess kein PNG-Format unterstützen und zum anderen setzen Werkzeuge wie f2d und East-tec InvisibleSecrets zwingend Passwörter für eine Entschlüsselung vor, daher können keine Bilder entschlüsselt werden, wenn sie zuvor ohne ein Passwort verschlüsselt wurde. Ein weiterer Grund ist die fehlende Interoperabilität der Werkzeuge zu der vorgestellten LSB-Methode. Insgesamt kann als abschließende Analyse festgehalten werden, dass eine Interoperabilität zwischen den ausgewählten Werkzeugen und der vorgestellten LSB-Methode mittels Python-Code nicht vorhanden ist. Das Ergebnis zeigt, dass eine Ver- und Entschlüsselung von Bildern daher nur innerhalb der ausgewählten Werkezuge selbst funktioniert. Eine Ausnahme davon ist Zsteg, da dieses Werkzeug nur Entschlüsselungen zulässt. Im Umkehrschluss bedeutet dies, dass wenn eine verborgene Information mit der vorgestellten LSB-Methode verschlüsselt wird und ein Dritter mithilfe eines vorgestellten Werkzeugs versucht, die verborgene Information zu entschlüsseln und abzugreifen, keinen Rückschluss auf die verborgene Information erhält. Dieses Ergebnis zeigt, dass Steganografie für die Möglichkeit der verborgenen Übertragung von geheimen Informationen genutzt werden kann. Zwei Kommunikationspartner, die sich untereinander geheime Informationen zuspielen wollen, können sich auf ein bestimmtes Werkzeug einigen, um das erhaltene Bild zu entschlüsseln. Dieses Fazit zeigt, dass die Auswahl des richtigen steganografischen Werkzeugs von großer Bedeutung ist, um eine erfolgreiche Ver- sowie Entschlüsselung von Bildern zu gewährleisten. Zudem ist ratsam die Kompatibilität der Werkzeuge mit den verwendeten Dateiformaten und Verschlüsselungsmethoden zu prüfen, um ein bestmögliches Ergebnis zu erhalten.

Die vorliegende Bachelor-Thesis liefert eine Grundlage für weitere Forschung und Untersuchungen im Bereich der Steganografie. Zukünftige Arbeiten könnten anderer steganografische Techniken untersuchen und vergleichen, denn diese

Bachelor-Thesis hat sich auf die LSB-Methode konzentriert. Zudem können weitere Werkzeuge analysiert werden, die aufgrund der eingeschränkten Zeit nicht durchgeführt worden. Zukünftige Arbeiten könnten ebenfalls den Einsatz von Kombinationen verschiedener steganografischer Techniken erforschen, um die Robustheit und Sicherheit der versteckten Informationen zu erhöhen. Des Weiteren kann ein zukünftiger Ausblick im Rahmen der digitalen Technologie erfolgen, denn aus dem ständigen Fortschritt in der Technologie können sich neue Möglichkeiten in der Steganografie entwickeln. Durch Technologien wie maschinelles Lernen oder künstliche Intelligenz kann Steganografie weiter analysiert werden im Hinblick auf Automatisierung von Ver- und Entschlüsselungen. Abschließend kann zusammenfassend gesagt werden, dass die Steganografie ein vielseitiges, faszinierendes und spannende Forschungsgebiet ist, welches Potenzial für zukünftige Entwicklung bietet. Besonders bei der Möglichkeit Malware mithilfe der Steganografie zu übermitteln oder das Scannen von QR-Codes, kann diese Bachelor-Thesis eine erste Sensibilisierung hervorrufen. Damit kann mit dieser Thesis eine Grundlage geschafft werden für weiterführende Studien, um die Sicherheit und Zuverlässigkeit von versteckten Informationen zu verbessern. Die zugrunde liegende Forschungsfrage: „*Wie lässt sich Steganografie zur Möglichkeit der verborgenen Übertragung von Informationen einsetzen und vergleichend bewerten?“* wurde im Rahmen der Bachelor-Thesis ausführlich untersucht. Die Ergebnisse zeigen, dass die verschiedenen steganografischen Werkzeuge eine Möglichkeit der verborgenen Übertragung von Informationen zulassen und ebenfalls anhand von objektiven Gütekriterien vergleichend bewertet werden können. Eine weitere Erkenntnis ist, dass es einfacher ist, Dateien mit einem bestimmten Werkzeug zu verschlüsseln und danach mit demselben Werkzeug zu entschlüsseln. Der Wunsch nach verborgener Kommunikation kann im Gegensatz dazu mit der vorgestellten LSB-Methode verschlüsselt werden. Insgesamt bietet diese Bachelor-Thesis einen wertvollen Ausgangspunt für vertiefende Untersuchungen im Bereich der Steganografie.

9 Literaturverzeichnis

[1] „Bleepingcomputer,“ [Online]. Available: https://www.bleepingcomputer.com/news/security/hackers-hide-malware-in-james-webb-telescope-images/ [Zugriff am 05.08.2023].

[2] „Bleepingcomputer,“ [Online]. Available: https://www.bleepingcomputer.com/news/security/fbi-warns-of-malicious-qr-codes-used-to-steal-your-money/ [Zugriff am 05.08.2023].

[3] S. WENDZEL, „IT-Sicherheit für TCP/IP - und IoT_Netzwerke. Grundlagen, Konzepte, Protokolle, Härtung,“ Wiesbaden, Springer Vieweg, 2018, S. 107.

[4] E. VON FABER, „IT und IT-Sicherheit in Begriffen und Zusammenhängen. Thematisch sortiertes Lexikon mit alphabetischem Register zum Nachschlagen,“ Wiesbaden, Springer Vieweg, 2021, S. 217.

[5] K.-R. MÜLLER, „Handbuch Unternehmenssicherheit. Umfassendes Sicherheits-, Kontinuitäts- und Risikomanagement mit System,“ Wiesbaden, Springer Vieweg, 2022, S. 449.

[6] N. POHLMANN, „Cyber-Sicherheit. Das Lehbruch für Konzepte, Prinzipien, Mechanismen, Architekturen und Eigenschaften von Cyber-Sicherheitssystemen in der Digitalisierung,“ WIesbaden, Springer Vieweg, 2022, S. 61.

[7] H. ERNST, J. SCHMIDT und G. BENEKEN, „Grundkurs Informatik. Grundlagen und Konzepte für die erfolgreiche IT-Praxis - Eine umfassende, praxisorientierte Einführung.,“ Wiesbaden, Springer Vieweg, 2020, S. 137.

[8] C. PAAR und J. PELZL, „Kryptografie verständlich. Ein Lehrbuch für Studierende und Anwender,“ Heidelberg, Springer-Verl. Berlin, 2016, S. 1.

[9] A. BEUTELSPACHER, H. B. NEUMANN und T. SCHWARZPAUL, „Kryptografie in Theorie und Praxis. Mathematische Grundlagen für Internetsicherheit, Mobilfunk und elektronisches Geld," Wiesbaden, Vieweg+Teubner, 2010, S. 5-6.

[10] C. MEINEL und H. SACK, „Sicherheit und Vertrauen im Internet. Eine technische Perspektive," Wiesbaden, Springer Vieweg, 2014, S. 9.

[11] P. WEBER, R. GABRIEL, T. LUX und K. MENKE, „Basiswissen Wirtschaftsinformatik," Wiesbaden, Springer Vieweg, 2022, S. 321, 322.

[12] J. GALLENBACHER, „Abenteuer Informatik. IT zum Anfassen für alle von 9 bis 99 - vom Navi bis Social Media," Heidelberg, Springer Verlag, 2020, S. 320.

[13] C. POSTHOFF, „Computer und Künstliche Intelligenz. Vergangenheit - Gegenwart - Zukunft," Wiesbaden, Springer Vieweg, 2022, S. 153.

[14] „ComputerWeekly.de," [Online]. Available: https://www.computerweekly.com/de/definition/Steganographie [Zugriff am 08.06.2023].

[15] S. PINCOCK und M. FRARY, „Geheime Codes. Die brühmtesten Verschlüsselungstechniken und ihre Geschichten.," Bergisch Gladbach, Lübbe GmbH & Co. KG, 2007, S. 14.

[16] K. SCHMEH, „Versteckte Botschaften. Die faszinierende Geschichte der Steganografie.," Hannover, Heise Gruppe HmbH & Co. KG, 2017, S. 122.

[17] „Hamfu," [Online]. Available: https://www.hamfu.ch/de/pdf/publikationen/pdf-GMS-REDOK-Korr-2019.pdf [Zugriff am 10.06.2023].

[18] „Stiftunglesen," [Online]. Available: https://www.stiftunglesen.de/fileadmin/Schulportal/06_Lehrmaterial/01_Materialien_fuer_den_Unterricht/115_Unterrichtsideen_Kryptologie/WW_Unterrichtsmaterial_FINAL.pdf [Zugriff am 10.06.2023].

[19] „Scienceblogs," [Online]. Available: https://scienceblogs.de/klausis-krypto-kolumne/2015/12/05/das-verschluesselte-telegramm-das-weltgeschichte-schrieb/ [Zugriff am 10.06.2023].

[20] „Focus," [Online]. Available: https://www.focus.de/wissen/experts/schmeh/versteckte-geheimbotschaften-wie-ein-kriegsgefangener-den-zensor-ueberlistete_id_3529736.html [Zugriff am 06.10.2023].

[21] „Micsymposium," [Online]. Available: https://micsymposium.org/mics_2006/papers/TosoAndChung.pdf [Zugriff am 08.06.2023].

[22] J. FRIDRICH, „Steganography in Digital Media: Principles, Algorithms, and Applications.," Cambridge, Cambridge University Press, 2010.

[23] S. RACHNA und R. KUMAR, „Bild-Steganographie. Verwendung der Least Significant Bit-Methodik.," Beau Bassin, Mauritius, AV Akademikerverlag, 2020, S. 27.

[24] B. JÄHNE, „Digitale Bildverarbeitung und Bildgewinnung.," Heidelberg, Springer Vieweg, 2012, S. 111.

[25] W. BURGER und M. J. BURGE, „Digitale Bildverarbeitung. Eine algorithmische Einführung mit Java.," Heidelberg, Springer Vieweg, 2015, S. 14.

[26] A. Siper, R. Farley und C. Lombardo, „The Rise of Steganography," *Proceedings of Student/Faculty Research Day, CSIS, Pace University,* Bd. 05, Nr. 06, S. 2, 2005.

[27] „Kaspersky," [Online]. Available: https://www.kaspersky.com/resource-center/definitions/what-is-steganography [Zugriff am 18.07.2023].

[28] „Gaijin," [Online]. Available: https://www.gaijin.at/de/infos/ascii-ansi-zeichentabelle [Zugriff am 15.06.2023].

[29] „Youtube," [Online]. Available: https://www.youtube.com/watch?v=XmZNxabIcRQ&t=1446s [Zugriff am 17.06.2023].

[30] „Medium," [Online]. Available: https://medium.com/@stephanie.werli/image-steganography-with-python-83381475da57 [Zugriff am 01.07.2023].

[31] „Steghide Sourceforge," [Online]. Available: https://steghide.sourceforge.net [Zugriff am 06.07.2023].

[32] „Pixabay Emu," [Online]. Available: https://pixabay.com/de/photos/vogel-emu-flugunfähige-schnabel-7917458/ [Zugriff am 23.06.2023].

[33] „Github," [Online]. Available: https://github.com/osde8info/stegosuite# [Zugriff am 06.07.2023].

[34] „kali," [Online]. Available: https://www.kali.org/tools/outguess/ [Zugriff am 07.07.2023].

[35] „Github Outguess," [Online]. Available: https://github.com/resurrecting-open-source-projects/outguess [Zugriff am 19.07.2023].

[36] „Wiki bi0s," [Online]. Available: https://wiki.bi0s.in/steganography/zsteg/ [Zugriff am 08.07.2023].

[37] „Github Zsteg,“ [Online]. Available: https://github.com/zed-0xff/zsteg [Zugriff am 08.07.2023].

[38] „Thepythoncode,“ [Online]. Available: https://www.thepythoncode.com/code/hide-secret-data-in-images-using-steganography-python [Zugriff am 31.07.2023].

[39] „Apps Microsoft Tater,“ [Online]. Available: https://apps.microsoft.com/store/detail/tater/9N2MDX397443 [Zugriff am 09.07.2023].

[40] „Windows Apps Store PicStealth,“ [Online]. Available: https://apps.microsoft.com/store/detail/picstealth/9NBLGGGZ5FJG?hl=en-us&gl=us [Zugriff am 09.07.2023].

[41] „Windows-Apps Store f2d,“ [Online]. Available: https://apps.microsoft.com/store/detail/f2d/9NBLGGH6CJZC [Zugriff am 09.07.2023].

[42] „SSuite Picsel Security,“ [Online]. Available: https://ssuite-picsel-security.de.uptodown.com/windows [Zugriff am 09.07.2023].

[43] „Stegano,“ [Online]. Available: https://www.stegano.org [Zugriff am 09.07.2023].

[44] „Github Sreganogram,“ [Online]. Available: https://github.com/steganogram/stegano-rs [Zugriff am 23.07.2023].

[45] „Github Stegify,“ [Online]. Available: https://github.com/DimitarPetrov/stegify [Zugriff am 13.07.2023].

[46] „Pentesttools,“ [Online]. Available: https://pentesttools.net/stegify-capable-of-hiding-any-file-within-an-image/ [Zugriff am 13.07.2023].

[47] „Lucasmw,“ [Online]. Available: https://lucasmw.itch.io/the-hider?ref=steemhunt [Zugriff am 20.07.2023].

[48] „Mac App Store Vorschau,“ [Online]. Available: https://apps.apple.com/de/app/the-hider/id1355396309?mt=12 [Zugriff am 20.07.2023].

[49] „East-tec,“ [Online]. Available: https://www.east-tec.com/invisiblesecrets/ [Zugriff am 20.07.2023].

[50] „Mathworks,“ [Online]. Available: https://de.mathworks.com/help/images/ref/psnr.html [Zugriff am 21.07.2023].

[51] „Wikipedia,“ [Online]. Available: https://de.wikipedia.org/wiki/Strukturelle_Ähnlichkeit#:~:text=SSIM%20wird%20zur%20Messung%20der,oder%20störungsfreien%20Ursprungsbild%20als%20Bezug. [Zugriff am 21.07.2023].

[52] L. Almehmadi , A. Basuhail , D. Alghazzawi und O. Rabie, „Framework for Malware Triggering Using Steganography,“ *applied sciences,* 16.08.2022.

[53] S. K. Ghosal, A. Chatterjee und R. Sarkar, „Image steganography based on Kirsch edge detection,“ *Multimedia Systems (2021) 27:73–87,* 12 10 2020.

[54] D. R. I. M. Setiadi, „PSNR vs SSIM: imperceptibility quality assessment for image steganography,“ Springer Science+Business Media, LLC, part of Springer Nature 2020, 2020.

[55] K. Sharanpreet, S. Surender, K. Manjit und L. Heung-No, „A Systematic Review of Computational Image Steganography Approaches,“ *Archives of Computational Methods in Engineering (2022) 29:4775–4797,* 14 06 2022.

[56] M. A. M. El-Bendary, „FEC merged with double security approach based on encrypted image steganography for different purpose in the presence of noise and different attacks," *Multimed Tools Appl (2017) 76:26463–26501,* 07.01.2017.

[57] „Mathworks MSE," [Online]. Available: https://de.mathworks.com/help/images/ref/immse.html [Zugriff am 23.07.2023].

[58] „Unipub," [Online]. Available: https://unipub.uni-graz.at/obvugrhs/download/pdf/6499176?originalFilename=true [Zugriff am 30.07.2023].

[59] „MATLAB," [Online]. Available: https://matlab.mathworks.com [Zugriff am 22.07.2023].

[60] „Imagemagick," [Online]. Available: https://imagemagick.org/script/compare.php [Zugriff am 22.07.2023].

[61] „Mathworks SSIM," [Online]. Available: https://de.mathworks.com/help/images/ref/ssim.html [Zugriff am 23.07.2023].

[62] F. Bauer, „Felix-Bauer," [Online]. Available: https://www.felix-bauer-it.de/blog/steganografie-schadcode-in-bildern/ [Zugriff am 03.08.2023].

[63] „Pixabay," [Online]. Available: https://pixabay.com/de/photos/hund-welpe-klein-blick-hundeblick-4500444/ [Zugriff am 01.07.2023].

[64] „Medium," [Online]. Available: https://medium.com/swlh/lsb-image-steganography-using-python-2bbbee2c69a2 [Zugriff am 02.07.2023].

[65] „Redketchup," [Online]. Available: https://redketchup.io/color-picker [Zugriff am 02.07.2023].

[66] „Youtube,“ [Online]. Available: https://www.youtube.com/watch?v=6Hk5KyiEktI [Zugriff am 02.07.2023].

[67] „Codecentric,“ [Online]. Available: https://www.codecentric.de/wissens-hub/blog/einfuehrung-in-computer-vision-mit-opencv-und-python [Zugriff am 02.07.2023].

[68] „Pillow,“ [Online]. Available: https://pillow.readthedocs.io/en/stable/ [Zugriff am 02.07.2023].

[69] „Geekyhuman,“ [Online]. Available: https://geekyhumans.com/de/vergleichen-sie-zwei-bilder-und-markieren-sie-unterschiede-mit-python/ [Zugriff am 02.07.2023].

[70] „w-hs,“ [Online]. Available: http://www3.w-hs.de/fb01/02-Informatik/BGI.SS2019/ASCII_Tabelle_DevCPP.pdf [Zugriff am 29.06.2023].

[71] „Wiki bi0s,“ [Online]. Available: https://wiki.bi0s.in/steganography/stegoveritas/ [Zugriff am 08.07.2023].

[72] „Github Stegoveritas,“ [Online]. Available: https://github.com/bannsec/stegoVeritas [Zugriff am 08.07.2023].

[73] „Softpedia QuickStego,“ [Online]. Available: https://www.softpedia.com/get/Security/Encrypting/QuickStego.shtml [Zugriff am 09.07.2023].

10 Abbildungsverzeichnis

Abbildung 1 - Kryptologie und ihre Untergebiete [8] 11
Abbildung 2 - Einfachste mögliche Vorgehensweise bei einer Verschlüsselung [6] 12
Abbildung 3 - Symmetrische Verschlüsselung [4] 13
Abbildung 4 - Asymmetrische Verschlüsselung [4] 13
Abbildung 5 - Hybride Verschlüsselung 14
Abbildung 6 - Digitale Signatur [11] 14
Abbildung 7 - Zimmermann-Telegramm [19] 17
Abbildung 8 - Kategorien der für Steganografie verwendete Dateiformate 18
Abbildung 9 – Bildkoordinaten [25] 21
Abbildung 10 - Quadrat1 27
Abbildung 11 - Rot-Wert (255, 0, 0) 28
Abbildung 12 - Schwarz-Wert (0, 0, 0) 28
Abbildung 13 - Gelb-Wert (255, 255, 0) 28
Abbildung 14 - Blau-Wert (0, 0, 255) 28
Abbildung 15 - ursprüngliche Farbwerte "Quadrat1" 28
Abbildung 16 - ursprüngliche Farbwerte "Quadrat1" mit Binärzahlen 29
Abbildung 17 - Rot-Wert (253, 2, 1) 31
Abbildung 18 - Schwarz-Wert (0, 1, 2) 31
Abbildung 19 - Gelb-Wert (255, 255, 1) 31
Abbildung 20 - Blau-Wert (2, 1, 255) 31
Abbildung 21 - Quadrat1 mit eingebetteter Geheimbotschaft 31
Abbildung 22 - Quadrat1 ohne Geheimbotschaft (links) und mit Geheimbotschaft (rechts) 32
Abbildung 23 - Code für LSB-Verschlüsselung [30] 32
Abbildung 24 - Steghide - Übersicht der Befehle 37
Abbildung 25 - emu.jpg [32] 37
Abbildung 26 - Erstellung "emunew.jpg" mit Geheimbotschaft 38
Abbildung 27 - emu.jpg ohne Geheimnachricht (links) und emunew.jpg mit Geheimnachricht 38
Abbildung 28 - Überprüfung emunew.jpg auf Komprimierung 38
Abbildung 29 - Steghide - extract-Befehl 39
Abbildung 30 - Stegosuite - Übersicht der Befehle 40
Abbildung 31 - Stegosuite GUI (links) und Einbettung Geheimbotschaft mit Passwort (rechts) 40
Abbildung 32 - emu1.jpg ohne Geheimbotschaft (links) und emu1_embed.jpg mit Geheimbotschaft (rechts) . 41
Abbildung 33 - Extraktion der Geheimbotschaft 41
Abbildung 34 - OutGuess - Übersicht der Befehle 42
Abbildung 35 - Einbettung der Geheimnachricht "information.txt" 42

Abbildung 36 - emu2.jpg ohne Geheimbotschaft (links) und emu2_new.jpg mit Geheimbotschaft (rechts) 43
Abbildung 37 - Extraktion der Geheimnachricht 43
Abbildung 38 - Einbettung der Geheimbotschaft mit einem Passwort 44
Abbildung 39 - Zsteg - Installation 45
Abbildung 40 - Übersicht Befehle 45
Abbildung 41 - Verschlüsselung "wismar2.png" 46
Abbildung 42 - pig.png ohne Geheimbotschaft (links) und wismar2_encoded.png mit Geheimbotschaft (rechts) 46
Abbildung 43 - Vergleich beider Bilder auf Geheimbotschaft 46
Abbildung 44 - Verbose-Modus 47
Abbildung 45 - Tater - Windows Store-App 48
Abbildung 46 - Übersicht der Anwendung 48
Abbildung 47 - Direkte Überprüfung eines Bildes auf Geheimbotschaft 48
Abbildung 48 - Einbettung Geheimbotschaft 49
Abbildung 49 - Extraktion der Geheimbotschaft (JPG) 49
Abbildung 50 - Extraktion der Geheimbotschaft (PNG) 50
Abbildung 51 - PicStealth - Windows Store-App 50
Abbildung 52 - PicStealth - Übersicht 51
Abbildung 53 - Verschlüsselung der Geheimbotschaft 51
Abbildung 54 - Entschlüsselung der Geheimbotschaft (JPG) 52
Abbildung 55 - Entschlüsselung der Geheimbotschaft (PNG) 52
Abbildung 56 - Hinweis Passwortlänge 53
Abbildung 57 - Versuchte Entschlüsselung ohne Eingabe des Passworts 53
Abbildung 58 - Entschlüsselung mit Passworteingabe 53
Abbildung 59 - f2d - Windows Store-App 54
Abbildung 60 - Übersicht der Funktionen von f2d 54
Abbildung 61 - Verschlüsselung mit Passwort und Text 55
Abbildung 62 - Verschlüsselung einer Geheimbotschaft 55
Abbildung 63 - Entschlüsselung der Geheimbotschaft 56
Abbildung 64 - SSuite Picsel Security - Übersicht der Funktionen 57
Abbildung 65 - Einbettung einer Geheimbotschaft 57
Abbildung 66 - Verschlüsseltes Bild mit Geheimnachricht 58
Abbildung 67 - Entschlüsselte Geheimnachricht 58
Abbildung 68 - Stegano - Installation 59
Abbildung 69 - Übersicht der Befehle 59
Abbildung 70 - Hilfsbefehl für das Ver- und Entschlüsseln einer Geheimbotschaft 60
Abbildung 71 - Geheimbotschaft 60
Abbildung 72 - Einbettung der Geheimbotschaft 60

Abbildung 73 – pig.png ohne Geheimbotschaft (links) und pignews.png mit Geheimbotschaft (rechts) 61
Abbildung 74 - Extrahierung der Geheimbotschaft 61
Abbildung 75 - Extrahierte Geheimbotschaft 61
Abbildung 76 - Stegify Befehlsübersicht 62
Abbildung 77 - Einbettung Geheimnachricht 62
Abbildung 78 - pig.png ohne Geheimnachricht (links) und pig-news.png mit Geheimnachricht (rechts) 63
Abbildung 79 - Extrahierung der Geheimbotschaft 63
Abbildung 80 - Übersicht im App Store "The Hider" 64
Abbildung 81 - The Hider - Oberfläche der App 64
Abbildung 82 - Zu verschlüsselndes Bild "pig.png" und die Geheimbotschaft "secret.txt" 65
Abbildung 83 - pig.png ohne Geheimbotschaft (links) und newpig.png mit Geheimbotschaft (rechts) 65
Abbildung 84 - Entschlüsselung des Geheimtextes 66
Abbildung 85 - Verschlüsselung "emu.jpg" 66
Abbildung 86 - emu.jpg ohne Geheimbotschaft (links) und newsemu.png mit Geheimbotschaft (rechts) 67
Abbildung 87 - Entschlüsselung "newsemu.png" 67
Abbildung 88 - Webseite "east-tec InvisibleSecrets" 68
Abbildung 89 - Übersicht der Produktversionen und Gratiszugang 68
Abbildung 90 - Installation und Auszug von „ReadMe“ von East-tec 69
Abbildung 91 - Übersicht Anwendungen East-tec und zu verschlüsselndes Bild turtle.png 69
Abbildung 92 - Auswahl Geheimbotschaft und Bild "turtle.png" 70
Abbildung 93 - Festlegung Passwort und Bildname "newsemu.png" 70
Abbildung 94 - Versteckens, Möglichkeiten des Anzeigens und neues Bild „newsturtle.ong“ 70
Abbildung 95 - "turtle.png" ohne (links) und "newsturtle.png" mit Geheimbotschaft (rechts) 71
Abbildung 96 - Extraktion und Eingabe des Passworts 71
Abbildung 97 - Erfolgreiche Extraktion 71
Abbildung 98 - Extrahierte Geheimbotschaft 72
Abbildung 99 – Stegify - Verschlüsselung größerer Dateien 83
Abbildung 100 – Stegify - Berechnung PSRN-, MSE- und SSIM-Werte 84
Abbildung 101 Stegano - Verschlüsselung größerer Dateien 84
Abbildung 102 - Stegano - Berechnung PSRN-, MSE- und SSIM-Werte 85
Abbildung 103 - Bild "dog" ohne Geheimnachricht [62] 88
Abbildung 104 - Anwendung LSB-Code auf das Bild "dog.png" 89
Abbildung 105 - Bild "dog-new.png" mit Geheimnachricht "Wismar" 89
Abbildung 106 - linkes Bild "dog.png" und rechtes Bild "dog-new.png" 90
Abbildung 107 - Vergleich Pixelwerte (2,0) von "dog.png" (links) und "dog-new.png" (rechts) [64] 90
Abbildung 108 - RGB-Werte (2,0) von "dog.png" 91
Abbildung 109 - RGB-Werte (2,0) von "dog-new.png" 91
Abbildung 110 - Vergleich beider Bilder auf veränderte Pixelwerte [66] [67] [68] 92

Abbildung 111 - Aufzählung der veränderten Pixelwerte 93
Abbildung 112 - Steghide - Fehlerhinweis 96
Abbildung 113 - Stegosuite – Fehlerhinweis 96
Abbildung 114 - OutGuess - Fehlerhinweis 96
Abbildung 115 - Zsteg - kein Hinweis auf eine versteckte Nachricht 97
Abbildung 116 - Tater - kein Hinweis auf versteckte Botschaft 97
Abbildung 117 - PicStealth - Fehlertext bzw. Sonderzeichen 98
Abbildung 118 - f2d - Fehlerhinweis 98
Abbildung 119 - SSuite Picsel Security - Fehlertext 99
Abbildung 120 - Stegano - Fehlermeldung bei der Extraktion 99
Abbildung 121 - Stegify - Extraktion und fehlerhafte txt-Datei 100
Abbildung 122 - The Hider - Fehlermeldung 100
Abbildung 123 - East-tec – Fehlerhinweis 101

11 Tabellenverzeichnis

TABELLE 1 - STEGANOGRAFISCHE METHODEN UND IHRE KLASSIFIZIERUNGEN [23] 25
TABELLE 2 - ÜBERSICHT FARBWERTE 28
TABELLE 3 - ÜBERSICHT BINÄRZAHLEN OHNE UND MIT GEHEIMBOTSCHAFT 30
TABELLE 4 - ÜBERSICHT FARBWERTE MIT EINGEBETTETER GEHEIMNACHRICHT 31
TABELLE 5 - OBJEKTIVE GÜTEKRITERIEN 76
TABELLE 6 - BEWERTUNG STEGANOGRAFISCHER WERKZEUGE 82
TABELLE 7 - ÜBERSICHT PSNR-, MSE- UND SSIM-WERTE 85
TABELLE 8 - ÜBERSICHT DER VERÄNDERTEN PIXELWERTE [69] 94

Verzeichnis der Abkürzungen

AES	Advanced Encryption Standard
ANSI	American National Standards Institute
AU	Audio
ASCII	American Standard Code for Information Interchange
BMP	Bitmap
bzw.	beziehungsweise
CLI	command-line-interface
dB	Dezibel
DCT	diskreten Kosinustransformation
DES	Data Encryption Standard
DWT	diskrete Wavelet-Transformation
etc.	et cetera
FBI	Federal Bureau of Investigation
GB	Gigabyte
GIF	Graphics Interchange Format
GUI	Graphical User Interface
HAS	Human Auditory System
HTML	Hypertext Markup Language
HVS	Human Visual System
Int	Integer
IQM	Image Quality Measurement
IPSec	Internet Protocol Security
IT-System	Information Technology System
JPEG, JPG	Joint Photographic Experts Group
KB	Kilobyte
LSB	Least significant bit
MB	Megabyte
MSB	Most significant bit
MSE	Mean Squared Error
NASA	National Aeronautics and Space Administration
PIL	Python Imaging Library

PNG	Portable Network Graphics
PNM	Portable Any Map
PPM	Portable Pixel Map
PSNR	Peak Signal-to-Noise Ratio
QR-Code	Quick-Response-Code
RGB	Rot-Grün-Blau
RSA-Algorithmus	Rivest-Shamir-Adleman-Algorithmus
SSIM	Structural Similarity Index
SSL	Secure Sockets Layer
TIFF	Tagged Image File Format
TLS	Transport Layer Security
u. a.	unter anderem
UI	User Interface
v. Chr.	vor Christus
vs.	versus
WAV	Waveform Audio File Format
WMF	Windows-Metadatei-Format
z. B.	zum Beispiel

12 Anhang

Anhang 1:

Anzeige der Dateigröße sowie Berechnung der MSE-, PSNR-, und SSIM-Werte mit *MATMAB* [58] und der CLI-Anwendung *magick* [59].

Steghide	Dateigröße	MSE	PSNR	SSIM
emu.jpg	120 KB	0.031822	63.10 dB	0.9999
emunew.jpg	130 KB			

```
>> ref=imread('emu.jpg');
A=imread('emunew.jpg');
err = immse(A, ref);
fprintf('\n The mean-squared error is %0.4f\n', err);

 The mean-squared error is 0.0318
>>
```

```
>> bild1 = imread('emu.jpg');
bild2 = imread('emunew.jpg');

% Berechne den PSNR-Wert zwischen den Bildern
psnrValue = psnr(bild1, bild2);

fprintf('PSNR-Wert: %.2f dB\n', psnrValue);
PSNR-Wert: 63.10 dB
>>
```

```
>> bild1 = imread('emu.jpg');
bild2 = imread('emunew.jpg');

% Berechne den SSIM-Wert zwischen den Bildern
ssimValue = ssim(bild1, bild2);

fprintf('SSIM-Wert: %.4f\n', ssimValue);
SSIM-Wert: 0.9999
>>
```

```
fprintf('Mean Squared Error (MSE): %0.6f\n', mse);
Mean Squared Error (MSE): 0.031822
>>
```

```
(base) konsuelabednarek@Konsuelas-MBP Neuer Ordner % magick compare -metric PSNR emu.jpg emunew.jpg null:
63.1035 (0.631035)
```

Stegosuite	Dateigröße	MSE	PSNR	SSIM
emu1.jpg	120 KB	3.230642	43.04 dB	0.9937
emu1_embed.jpg	123 KB			

```
>> ref=imread('emu1.jpg');
A=imread('emu1_embed.jpg');
err = immse(A, ref);
fprintf('\n The mean-squared error is %0.4f\n', err);

 The mean-squared error is 3.2306
>>
```

```
>> bild1 = imread('emu1.jpg');
bild2 = imread('emu1_embed.jpg');

% Berechne den PSNR-Wert zwischen den Bildern
psnrValue = psnr(bild1, bild2);

fprintf('PSNR-Wert: %.2f dB\n', psnrValue);
PSNR-Wert: 43.04 dB
>> bild1 = imread('emu1.jpg');
bild2 = imread('emu1_embed.jpg');

% Berechne den SSIM-Wert zwischen den Bildern
ssimValue = ssim(bild1, bild2);

fprintf('SSIM-Wert: %.4f\n', ssimValue);
SSIM-Wert: 0.9937
>> bild1 = imread('emu1.jpg');
bild2 = imread('emu1_embed.jpg');
```

```
bild1 = imread('emu1.jpg');
bild2 = imread('emu1_embed.jpg');
mse = immse(bild1,bild2);

fprintf('Mean Squared Error (MSE): %0.6f\n', mse);
Mean Squared Error (MSE): 3.230642
```

```
(base) konsuelabednarek@Konsuelas-MBP Neuer Ordner % magick compare -metric PSNR emu1.jpg emu1_embed.jpg null:
43.0379 (0.430379)
```

OutGuess	Dateigröße	MSE	PSNR	SSIM
emu2.jpg	120 KB	8.158686	39.01 dB	0.9862
emu2_new.jpg	117 KB			

```
>> ref=imread('emu2.jpg');
A=imread('emu2_new.jpg');
err = immse(A, ref);
fprintf('\n The mean-squared error is %0.4f\n', err);

 The mean-squared error is 8.1587
>>
```

```
>> bild1 = imread('emu2.jpg');
bild2 = imread('emu2_new.jpg');

% Berechne den PSNR-Wert zwischen den Bildern
psnrValue = psnr(bild1, bild2);

fprintf('PSNR-Wert: %.2f dB\n', psnrValue);
PSNR-Wert: 39.01 dB
>> bild1 = imread('emu2.jpg');
bild2 = imread('emu2_new.jpg');

% Berechne den SSIM-Wert zwischen den Bildern
ssimValue = ssim(bild1, bild2);

fprintf('SSIM-Wert: %.4f\n', ssimValue);
SSIM-Wert: 0.9862
>> bild1 = imread('emu2.jpg');
bild2 = imread('emu2_new.jpg');
```

```
bild1 = imread('emu2.jpg');
bild2 = imread('emu2_new.jpg');
mse = immse(bild1,bild2);

fprintf('Mean Squared Error (MSE): %0.6f\n', mse);
Mean Squared Error (MSE): 8.158686
>>
```

```
(base) konsuelabednarek@Konsuelas-MBP Neuer Ordner % magick compare -metric PSNR emu2.jpg emu2_new.jpg null:
39.0146 (0.390146)
```

Zsteg	Dateigröße	MSE	PSNR	SSIM
wismar.png	12,3 MB	0.000002	105.57 dB	1.0000
wismar-secret.png	11,2 MB			

```
>>
bild1 = imread('wismar2.png');
bild2 = imread('wismar2_encoded.png');

mse_value = immse(bild1, bild2);
fprintf('MSE-Wert: %.2f\n', mse_value);
MSE-Wert: 0.00
>>
bild1 = imread('wismar2.png');
bild2 = imread('wismar2_encoded.png');

mse_value = immse(bild1, bild2);
fprintf('MSE-Wert: %.6f\n', mse_value);
MSE-Wert: 0.000002
>>
bild1 = imread('wismar2.png');
bild2 = imread('wismar2_encoded.png');

psnr_value = psnr(bild1, bild2);
fprintf('PSNR-Wert: %0.6f dB\n', psnr_value);
PSNR-Wert: 105.567600 dB
>>
bild1 = imread('wismar2.png');
bild2 = imread('wismar2_encoded.png');

ssim_value = ssim(bild1, bild2);
fprintf('SSIM-Wert: %0.6f\n', ssim_value);
SSIM-Wert: 1.000000
>>
```

```
(base) konsuelabednarek@Konsuelas-MBP opencv % magick compare -metric PSNR
 wismar2.png wismar2_encoded.png null:
105.568 (1.05568)
(base) konsuelabednarek@Konsuelas-MBP opencv %
```

Tater	Dateigröße	MSE	PSNR	SSIM
emu.jpg	120 KB	0.052941	60.89 dB	0.9999
emu-secret.png	1,2 MB			

```
>> bild1 = imread('emu.jpg');
bild2 = imread('emu-secret.png');
mse = immse(bild1,bild2);

fprintf('Mean Squared Error (MSE): %0.4f\n', mse);
Mean Squared Error (MSE): 0.0529
>>
```

```
>> bild1 = imread('emu.jpg');
bild2 = imread('emu-secret.png');

% Berechne den PSNR-Wert zwischen den Bildern
psnrValue = psnr(bild1, bild2);

fprintf('PSNR-Wert: %.2f dB\n', psnrValue);
PSNR-Wert: 60.89 dB
>> bild1 = imread('emu.jpg');
bild2 = imread('emu-secret.png');

% Berechne den SSIM-Wert zwischen den Bildern
ssimValue = ssim(bild1, bild2);

fprintf('SSIM-Wert: %.4f\n', ssimValue);
SSIM-Wert: 0.9999
>> bild1 = imread('emu.jpg');
bild2 = imread('emu-secret.png');
```

```
bild1 = imread('emu.jpg');
bild2 = imread('emu-secret.png');
mse = immse(bild1,bild2);

fprintf('Mean Squared Error (MSE): %0.6f\n', mse);
Mean Squared Error (MSE): 0.052941
>>
```

```
(base) konsuelabednarek@Konsuelas-MBP Neuer Ordner % magick compare -metric PSNR emu.jpg emu-secret.png null:
60.8933 (0.608933)
```

PicStealth	Dateigröße	MSE	PSNR	SSIM
emu.jpg	120 KB	0.052941	60.89 dB	0.9999
Emu_info.png	1,1 MB			
emu.jpg	120 KB	0.039950	62.16 dB	0.9999
emu_password.png	1 MB			

```
>> bild1 = imread('emu.jpg');
bild2 = imread('Emu_info.png');
mse = immse(bild1,bild2);

fprintf('Mean Squared Error (MSE): %0.4f\n', mse);
Mean Squared Error (MSE): 0.0529
>>
```

```
bild1 = imread('emu.jpg');
bild2 = imread('emu_password.png');
mse = immse(bild1,bild2);

fprintf('Mean Squared Error (MSE): %0.4f\n', mse);
Mean Squared Error (MSE): 0.0396
>>
```

```
>> bild1 = imread('emu.jpg');
bild2 = imread('Emu_info.png');

% Berechne den PSNR-Wert zwischen den Bildern
psnrValue = psnr(bild1, bild2);

fprintf('PSNR-Wert: %.2f dB\n', psnrValue);
PSNR-Wert: 60.89 dB
>> bild1 = imread('emu.jpg');
bild2 = imread('Emu_info.png');

% Berechne den SSIM-Wert zwischen den Bildern
ssimValue = ssim(bild1, bild2);

fprintf('SSIM-Wert: %.4f\n', ssimValue);
SSIM-Wert: 0.9999
>> bild1 = imread('emu.jpg');
bild2 = imread('Emu_info.png');
```

```
>> bild1 = imread('emu.jpg');
bild2 = imread('emu_password.png');

% Berechne den PSNR-Wert zwischen den Bildern
psnrValue = psnr(bild1, bild2);

fprintf('PSNR-Wert: %.2f dB\n', psnrValue);
PSNR-Wert: 62.16 dB
>> bild1 = imread('emu.jpg');
bild2 = imread('emu_password.png');

% Berechne den SSIM-Wert zwischen den Bildern
ssimValue = ssim(bild1, bild2);

fprintf('SSIM-Wert: %.4f\n', ssimValue);
SSIM-Wert: 0.9999
>> bild1 = imread('emu.jpg');
bild2 = imread('emu_password.png');
```

```
bild1 = imread('emu.jpg');
bild2 = imread('Emu_info.png');
mse = immse(bild1,bild2);

fprintf('Mean Squared Error (MSE): %0.6f\n', mse);
Mean Squared Error (MSE): 0.052941
>>
bild1 = imread('emu.jpg');
bild2 = imread('emu_password.png');
mse = immse(bild1,bild2);

fprintf('Mean Squared Error (MSE): %0.6f\n', mse);
Mean Squared Error (MSE): 0.039550
>>
```

```
(base) konsuelabednarek@Konsuelas-MBP Neuer Ordner % magick compare -metric PSNR emu.jpg Emu_info.png null:
60.8933 (0.608933)
```

```
(base) konsuelabednarek@Konsuelas-MBP Neuer Ordner % magick compare -metric PSNR emu.jpg emu_password.png null:
62.1593 (0.621593)
```

f2d	Dateigröße	MSE	PSNR	SSIM
puppy.png	262 KB	0.056004	60.65 dB	0.9999
puppy_pw.png	1,3 MB			

```
bild1 = imread('puppy.png');
bild2 = imread('puppy_pw.png');
mse = immse(bild1,bild2);

fprintf('Mean Squared Error (MSE): %0.4f\n', mse);
Mean Squared Error (MSE): 0.0560
>>
```

```
>> bild1 = imread('puppy.png');
bild2 = imread('puppy_pw.png');

% Berechne den PSNR-Wert zwischen den Bildern
psnrValue = psnr(bild1, bild2);

fprintf('PSNR-Wert: %.2f dB\n', psnrValue);
PSNR-Wert: 60.65 dB
>> bild1 = imread('puppy.png');
bild2 = imread('puppy_pw.png');

% Berechne den SSIM-Wert zwischen den Bildern
ssimValue = ssim(bild1, bild2);

fprintf('SSIM-Wert: %.4f\n', ssimValue);
SSIM-Wert: 0.9999
>> bild1 = imread('puppy.png');
bild2 = imread('puppy_pw.png');
```

```
bild1 = imread('puppy.png');
bild2 = imread('puppy_pw.png');
mse = immse(bild1,bild2);

fprintf('Mean Squared Error (MSE): %0.6f\n', mse);
Mean Squared Error (MSE): 0.056004
>>
```

```
(base) konsuelabednarek@Konsuelas-MBP Neuer Ordner % magick compare -metric PSNR puppy.png puppy_pw.png null:
60.6486 (0.606486)
```

SSuite Picsel	Dateigröße	MSE	PSNR	SSIM
turtle.jpg	486 KB	0.196491	55.20 dB	0.9999
turtle_pw.png	2,3 MB			

```
bild1 = imread('turtle.jpg');
bild2 = imread('turtle_pw.png');
mse = immse(bild1,bild2);

fprintf('Mean Squared Error (MSE): %0.4f\n', mse);
Mean Squared Error (MSE): 0.1965
>>
```

```
% Berechne den PSNR-Wert zwischen den Bildern
psnrValue = psnr(bild1, bild2);

fprintf('PSNR-Wert: %.2f dB\n', psnrValue);
PSNR-Wert: 55.20 dB
>> bild1 = imread('turtle.jpg');
bild2 = imread('turtle_pw.png');

% Berechne den SSIM-Wert zwischen den Bildern
ssimValue = ssim(bild1, bild2);

fprintf('SSIM-Wert: %.4f\n', ssimValue);
SSIM-Wert: 0.9999
>> bild1 = imread('turtle.jpg');
bild2 = imread('turtle_pw.png');
```

```
bild1 = imread('turtle.jpg');
bild2 = imread('turtle_pw.png');
mse = immse(bild1,bild2);

fprintf('Mean Squared Error (MSE): %0.6f\n', mse);
Mean Squared Error (MSE): 0.196491
>>
```

```
(base) konsuelabednarek@Konsuelas-MBP Neuer Ordner % magick compare -metric PSNR turtle.png turtle_pw.png null:
56.5476 (0.565476)
```

Stegano	Dateigröße	MSE	PSNR	SSIM
pig.png	2 MB	0.000084	88.91 dB	1.0000
pignews.png	1,9 MB			

```
bild1 = imread('pig.png');
bild2 = imread('pignews.png');
mse = immse(bild1,bild2);

fprintf('Mean Squared Error (MSE): %0.4f\n', mse);
Mean Squared Error (MSE): 0.0001
>>
```

```
% Berechne den PSNR-Wert zwischen den Bildern
psnrValue = psnr(bild1, bild2);

fprintf('PSNR-Wert: %.2f dB\n', psnrValue);
PSNR-Wert: 88.91 dB
>> bild1 = imread('pig.png');
bild2 = imread('pignews.png');

% Berechne den SSIM-Wert zwischen den Bildern
ssimValue = ssim(bild1, bild2);

fprintf('SSIM-Wert: %.4f\n', ssimValue);
SSIM-Wert: 1.0000
>> bild1 = imread('pig.png');
bild2 = imread('pignews.png');
```

```
fprintf('Mean Squared Error (MSE): %0.6f\n', mse);
Mean Squared Error (MSE): 0.000084
>>
```

```
(base) konsuelabednarek@Konsuelas-MBP Neuer Ordner % magick compare -metric PSNR pig.png pignews.png null:
90.1553 (0.901553)
```

Stegify	Dateigröße	MSE	PSNR	SSIM
pig.png	2 MB	0.000028	93.66 dB	1.0000
pig-news.png	1,6 KB			

```
>>  bild1 = imread('pig.png');
bild2 = imread('pig-news.png');
mse = immse(bild1,bild2);

fprintf('Mean Squared Error (MSE): %0.6f\n', mse);
Mean Squared Error (MSE): 0.000028
>>
```

```
% Berechne den PSNR-Wert zwischen den Bildern
psnrValue = psnr(bild1, bild2);

fprintf('PSNR-Wert: %.2f dB\n', psnrValue);
PSNR-Wert: 93.66 dB
>> bild1 = imread('pig.png');
bild2 = imread('pig-news.png');

% Berechne den SSIM-Wert zwischen den Bildern
ssimValue = ssim(bild1, bild2);

fprintf('SSIM-Wert: %.4f\n', ssimValue);
SSIM-Wert: 1.0000
>> bild1 = imread('pig.png');
bild2 = imread('pig-news.png');
```

```
(base) konsuelabednarek@Konsuelas-MBP Neuer Ordner % magick compare -metric PSNR pig.png pig-news.png null:
94.9081 (0.949081)
```

The Hider	Dateigröße	MSE	PSNR	SSIM
pig.png	2 MB	0.000486	81.26 dB	1.0000
newpig.png	1,9 MB			

```
bild1 = imread('pig.png');
bild2 = imread('newpig.png');
mse = immse(bild1,bild2);

fprintf('Mean Squared Error (MSE): %0.6f\n', mse);
Mean Squared Error (MSE): 0.000486
>>
```

```
% Berechne den PSNR-Wert zwischen den Bildern
psnrValue = psnr(bild1, bild2);

fprintf('PSNR-Wert: %.2f dB\n', psnrValue);
PSNR-Wert: 81.26 dB
>> bild1 = imread('pig.png');
bild2 = imread('newpig.png');

% Berechne den SSIM-Wert zwischen den Bildern
ssimValue = ssim(bild1, bild2);

fprintf('SSIM-Wert: %.4f\n', ssimValue);
SSIM-Wert: 1.0000
>> bild1 = imread('pig.png');
bild2 = imread('newpig.png');
```

```
(base) konsuelabednarek@Konsuelas-MBP Neuer Ordner % magick compare -metric PSNR pig.png newpig.png null:
82.5108 (0.825108)
```

East-tec	Dateigröße	PSNR	SSIM	MSE
turtle.png	2,7 MB	Inf dB	1.0000	0.000000
newsturtle.png	2,7 MB			

```
>> bild1 = imread('turtle.png');
bild2 = imread('newsturtle.png');
mse = immse(bild1,bild2);

fprintf('Mean Squared Error (MSE): %0.10f\n', mse);
Mean Squared Error (MSE): 0.0000000000
```

```
% Berechne den PSNR-Wert zwischen den Bildern
psnrValue = psnr(bild1, bild2);

fprintf('PSNR-Wert: %.2f dB\n', psnrValue);
PSNR-Wert: Inf dB
>> bild1 = imread('turtle.png');
bild2 = imread('newsturtle.png');

% Berechne den SSIM-Wert zwischen den Bildern
ssimValue = ssim(bild1, bild2);

fprintf('SSIM-Wert: %.4f\n', ssimValue);
SSIM-Wert: 1.0000
>> bild1 = imread('turtle.png');
bild2 = imread('newsturtle.png');
```

```
(base) konsuelabednarek@Konsuelas-MBP Neuer Ordner % magick compare -metric PSNR turtle.png newsturtle.png null:
0 (0)
(base) konsuelabednarek@Konsuelas-MBP Neuer Ordner %
```

13 Selbstständigkeitserklärung

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe.

Die Stellen der Arbeit, die anderen Quellen im Wortlaut oder dem Sinn nach entnommen wurden, sind durch Angaben der Herkunft kenntlich gemacht. Dies gilt auch für Zeichnungen, Skizzen, bildliche Darstellungen sowie für Quellen aus dem Internet.

Ich erkläre ferner, dass ich die vorliegende Arbeit in keinem anderen Prüfungsverfahren als Prüfungsarbeit eingereicht habe oder einreichen werde.

Die eingereichte schriftliche Fassung entspricht der auf dem Medium gespeicherten Fassung.

Potsdam, 10.08.23 Konstanze Bauer

Ort, Datum (Unterschrift)