Grundlagen der Steganographie

Inhaltsverzeichnis

1. Einführung

Steganographie (griech.: „verdeckt schreiben“) ist eine seit dem Altertum bekannte Methode zur vertraulichen Kommunikation: Die geheime Nachricht wird dazu innerhalb anderer, unauffälliger Daten — die i. allg. wesentlich umfangreicher sind als die eigentliche Nachricht — versteckt. Die Vertraulichkeit der Nachricht wird durch das Verbergen ihrer Existenz erreicht. Diese intuitive Art der Geheimhaltung wurde bereits in der Antike benutzt. Aus der

Geschichte sind viele Beispiele für ihre Anwendung und Realisierung bekannt, wie die geheimen Tinten, bestehend aus Milch, Fruchtsäften oder auch chemischen Stoffen, welche erst bei Erhitzung sichtbar werden.

Im Vergleich zur Kryptographie (griech.: „geheim schreiben“), mit welcher die Vertraulichkeit von Nachrichteninhalten geschützt werden kann, bietet die Steganographie eine weitere Stufe der Geheimhaltung. Die kryptographische Verschlüsselung erzeugt möglichst zufällig aussehende Texte, was einen Beobachter darauf hinweisen kann, dass etwas geheim gehalten werden soll. Wird dagegen Steganographie eingesetzt, werden „normal“ aussehende, für einen Angreifer plausibel wirkende Daten erzeugt. Dass diese Daten weitere, geheime Botschaften enthalten, ist für den Angreifer nicht ersichtlich. Steganographie bietet somit zusätzlich zum Schutz des Inhaltes auch die Verdecktheit der Kommunikation an sich.

Außerdem ist Steganographie ein wichtiges Argument gegen Kryptoregulierungen: Da die Anwendung sicherer Steganographie nicht erkennbar ist, kann sie auch nicht wirkungsvoll verboten werden. Somit ist Steganographie eine Alternative zur vertraulichen Kommunikation, sollte die Anwendung sicherer Kryptographie nicht möglich sein. Die verstärkten Diskussionen über Kryptoregulierungen vor reichlich 10 Jahren waren ein wesentlicher Anstoss für die Forschung auf dem Gebiet der Steganographie und verhalfen dieser alten Kunst zu einem neuen Aufschwung.

2. Funktionsweise steganographischer Systeme

Abbildung 1 zeigt die Funktionsweise eines steganographischen Systems (auch kurz als Stegosystem bezeichnet).

Ein steganographisches System beinhaltet zwei Operationen: *Einbetten (Embed)* und *Extrahieren (Extract)*. Der Sender verwendet die Operation *Einbetten*, um die geheime *Nachricht (emb)* in den *Coverdaten (cover,* auch als Trägerdaten oder Hülldaten bezeichnet) zu verstecken. Die daraus resultierenden *Stegodaten (stego)* werden zum Empfänger der Nachricht übertragen. Dieser liest die geheime Nachricht mit Hilfe der Operation *Extrahieren* wieder aus den Stegodaten aus. Die extrahierte *Nachricht* (emb*)* sollte der eingebetteten Nachricht entsprechen. Dagegen ist es meist nicht möglich (und für Steganographie auch nicht nötig), das ursprüngliche Cover wieder zu rekonstruieren, da Teile der Coverdaten durch das Einbetten verloren gehen.

Auch bei steganographischen Systemen darf die Sicherheit nicht auf der Geheimhaltung der Verfahren (*„security by obscurity“*) beruhen, sondern auf der Geheimhaltung von Schlüsseln.[1] Deshalb werden auch hier *Schlüssel* (*key*, auch als Stegoschlüssel bezeichnet) zur Parametrisierung der steganographischen Algorithmen eingesetzt. Die steganographischen Schlüssel zur Steuerung der Einbettungs- und Extraktionsalgorithmen sind dabei symmetrisch. Bei Steganographie mit öffentlichen Schlüsseln handelt es sich um hybride Systeme: Mittels Diffie-Hellmann-Schlüsselaustausch[2] (asymmetrisch) werden geheime (symmetrische) Sit-

[1] Kerkhoffs-Prinzip: Die Sicherheit eines Verschlüsselungssystems darf nicht auf der Geheimhaltung des Algorithmus beruhen, sondern nur auf Geheimhaltung des Schlüssels, denn Geheimhaltung kann das Brechen eines Systems nur verzögern, aber ein System nicht sicherer machen.

[2] Eine Beschreibung kann z. B. in [3] gefunden werden.

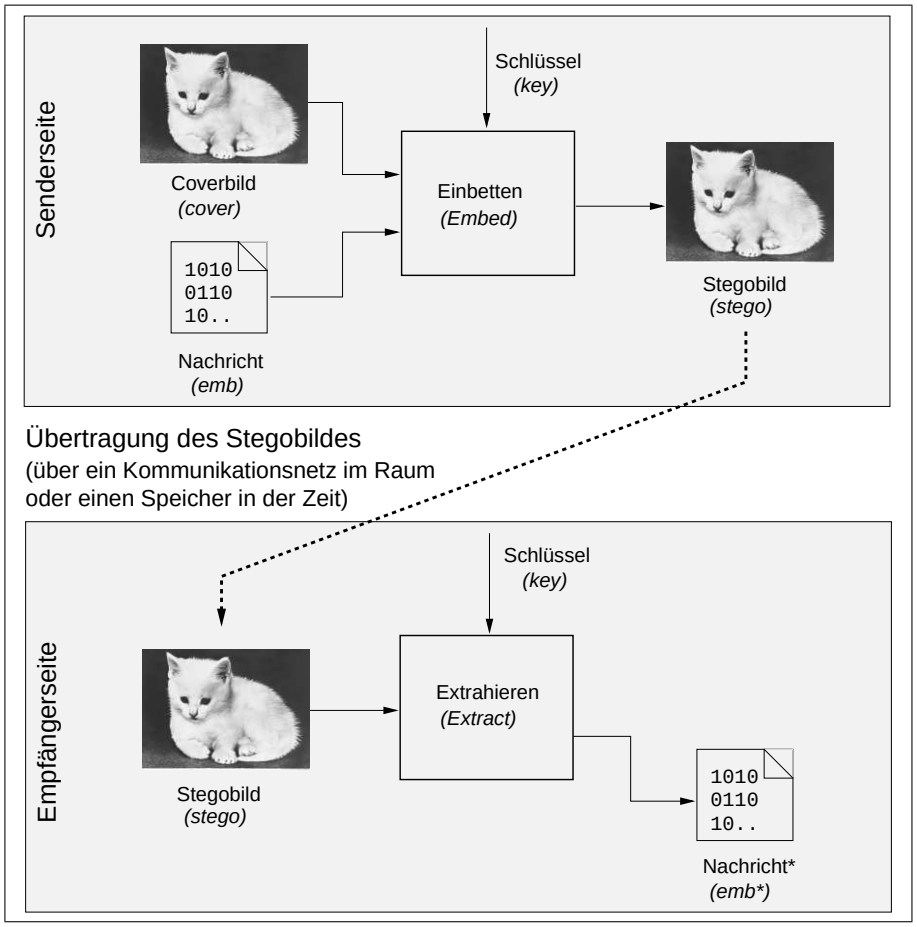

Abbildung 1: Allgemeines Modell eines Stegosystems

zungsschlüssel für die steganographischen Funktionen erzeugt.

Die Coverdaten müssen vor allem plausibel sein, es sollten also beispielsweise Bilder in einem üblichen Format verwendet werden. Es bieten sich außerdem eine Vielzahl weiterer Medien an, zum Beispiel Videos, Musikdaten oder Texte. Die durch das Einbetten entstandenen Stegodaten müssen ebenfalls plausibel sein, sie dürfen keinen Rückschluss auf die Anwendung von Steganographie zulassen.

3. Angriffe auf steganographische Systeme

3.1. Angriffsziel

Angriffe auf steganographische Systeme haben das Ziel, **steganographische Modifikationen aufzuspüren und damit die Anwendung von Steganographie zu entdecken**. Im Hinblick auf die Geheimhaltungseigenschaften wird dies in [6] folgendermaßen formuliert: Ste-

ganographische Systeme dienen ebenso wie kryptographische Systeme der Geheimhaltung von Nachrichten, wobei jedoch zusätzlich zur Konzelationseigenschaft „Vertraulichkeit der Nachricht“ noch die steganographische Eigenschaft „Vertraulichkeit der Existenz der Nachricht“ geboten wird. Ein steganographisches System gilt als unsicher, wenn die steganographische Eigenschaft verletzt, also die Existenz der Nachricht entdeckt wurde. Ob darüber hinaus auch der Inhalt der Nachricht entziffert werden kann (Brechen der Konzelationseigenschaft), ist dabei nicht relevant.

In Analogie zur Kryptographie gilt ein Stegosystem als theoretisch sicher, falls es einem Angreifer, der über unbeschränkte Mittel und unbegrenzte Zeit verfügt, nicht gelingt, seine Vermutung über die Anwendung von Steganographie zu bestätigen. Die Kenntnis des Algorithmus wird dabei entsprechend des Kerkhoffs-Prinzip vorausgesetzt. Da die Informationstheorie Basis für die Sicherheitsbetrachtungen ist, wird von *informationstheoretischer Sicherheit* gesprochen.

Beweisbar sichere Steganographie, die mit „natürlichen“ Daten arbeitet, ist bislang, im Gegensatz zur Kryptographie, praktisch nicht möglich. Wird in der Praxis von „sicheren Systemen“ gesprochen, bezieht sich das nicht auf die informationstheoretische Sicherheit. Vielmehr ist damit ein System gemeint, für das bislang keine erfolgreichen Angriffe gefunden wurden.

Der grundlegende Ansatz für die Angriffe liegt in der Feststellung der Veränderungen der Coverdaten, die durch das Einbetten hervorgerufen werden. Erfolgreichen Angriffen gelingt es, Unterschiede zwischen der Charakteristik der Cover- und der Stegodaten auszumachen. Beispielsweise stellen die in [5] beschriebenen *visuellen Angriffe* die Veränderung der Charakteristik der niederwertigsten Bits fest, die im gleichen Artikel beschriebenen *statistischen Angriffe* weisen typische Modifikationen der Verteilung der Grauwerte durch bestimmte Einbettungsoperationen nach.

Den wichtigsten Angriffspunkt auf steganographische Systeme beschreibt somit immer die Frage, wer über die bessere Modellierung der Coverdaten verfügt: der Designer des steganographischen Systems oder der Angreifer. Steganographie und Steganalyse sind eng miteinander verbunden. Beim Entwurf eines steganographischen Systems soll das Ziel erreicht werden, sicher gegen bekannte Angriffe einbetten zu können. Steganalyse ist somit ein Mittel zur Bewertung der Sicherheit der Stegosysteme.

3.2. Angreifermodell

Ein Angreifermodell beschreibt die Stärke eines Angreifers, gegen die ein System noch sicher ist. In Anlehnung an die Kryptoanalyse wird auch bei Steganographie beschrieben, welche Daten der Angreifer kennt, und über welche Möglichkeiten er verfügt. Ein passiver Angreifer kann nur beobachten. Ein aktiver Angreifer greift bereits vor dem eigentlichen Angriff in das System ein, indem er das System ganz gezielt Aktionen ausführen läßt oder auch Daten manipuliert.

Bezüglich der Daten geht man üblicherweise von der Annahme aus, daß der Angreifer einen Stego-Angriff durchführen kann. Die Stegodaten werden über einen unsicheren Kommunikationskanal übertragen, und auf dieser Strecke können sie vom Angreifer abgefangen und analysiert werden.

3.3. Strategie des Angreifers

Generell werden bei einem Angriff Merkmale der potentiellen Stegodaten ausgewertet. Als erstes muß sich der Angreifer für die Merkmale entscheiden, die er auswerten will. Die im folgenden beschriebenen Auswertungen beziehen sich auf digitale Bilder, welche in diesem Praktikumsversuch als Coverdaten verwendet werden.

Zunächst wird ein Angreifer sich ein Bild natürlich ansehen: Sieht es wie ein „normales“ Bild aus, oder sind auffällige Störungen zu erkennen? Es ist naheliegend, daß eine solche Beurteilung sehr schwierig ist. Darüber hinaus genügt die Wahrnehmbarkeit als Kriterium ohnehin nicht, da das Wahrnehmungsvermögen des Menschen nicht in der Lage ist, alle Informationen der Daten aufzunehmen. Dieser Umstand wird z. B. bei verlustbehafteter Kompression ausgenutzt, wo man zwischen relevanter und irrelevanter Information unterscheidet und die irrelevante Information entfernt wird, beispielsweise bei der JPEG-Kompression die hohen Frequenzen der zu komprimierenden Bilder. Der dadurch entstehende Qualitätsverlust ist i. allg. nicht wahrnehmbar.

Die Stegoprogramme der ersten Generation hatten die Nichtwahrnehmbarkeit der Modifikationen zum Ziel. Erfolgreiche Angriffe auf Systeme dieser Art benutzten weitere Auswertungen der Daten, um die Anwendung von Steganographie festzustellen. Für Steganalyse besonders interessant sind statistische Analysen, mit welchen nicht wahrnehmbare Unterschiede zwischen Bildern festgestellt werden können. Diese Analysen lassen sich nach Auswertung von statistischen Merkmalen erster Ordnung und statistischen Merkmalen höherer Ordnung unterteilen. Erstere beschreiben die Verteilung der Farb- bzw. Grauwerte des Bildes[3] letztere strukturelle Merkmale. Da bei Statistik erster Ordnung nur angegeben wird, mit welcher Häufigkeit die Grauwerte in einem Bild auftreten, können Bilder mit gleicher Grauwertverteilung völlig unterschiedlich aussehen. Die Auswertung statistischer Merkmale höherer Ordnung stellt demzufolge eine noch weitergehende Analyse des Bildes und damit einen stärkeren Angriff dar.

Welches Merkmal auch immer ausgewertet wird, sei es nun das Aussehen des Bildes, die Verteilung der Farbwerte oder Korrelationen zwischen Pixeln, es kann natürlich sowohl von Cover- als auch von Stegodaten ermittelt werden. Nach Berechnung der Merkmale muss festgelegt werden, welche Ausprägungen denn nun auf Steganographie hinweisen. Eine Möglichkeit besteht in der Modellierung der Auswirkungen steganographischer Operationen auf die Merkmale des Coverbildes. Weiterhin bietet die Untersuchung einer größeren Testmenge von Daten die Möglichkeit, auf empirischem Weg Schwellwerte zwischen Merkmalen von Coverdaten und Stegodaten zu ermitteln. Dieses Vorgehen kann mit Klassifizierungsalgorithmen der Bildverarbeitung [4] verglichen werden, bei denen der Analysealgorithmus mit Hilfe einer Trainingsmenge von Bildern auf Schwellwerte „trainiert“ wird. Eine dritte Möglichkeit ist die Untersuchung von Veränderungen der Merkmale durch die Ausführung von steganographischen und nicht-steganographischen Operationen. Um ein abgefangenes Bild auf diese Weise zu analysieren, werden diese Operationen mit ihm durchgeführt, die Veränderung der Merkmale gemessen und bewertet.

Es hängt von der gewählten Entscheidungsmethode ab, worauf sich die Aussage des Angriffs stützen kann:

[3] Bei Analysen im Frequenzraum werden natürlich Frequenzen ausgewertet.

- nur auf das subjektives Empfinden des Angreifers,
- auf den Vergleich mit empirisch ermittelten Schwellwerten oder
- auf den statistischen Test einer aufgestellten Hypothese.

Die letzten beiden Methoden können rechnergestützt realisiert werden. Gegenüber der subjektiven Entscheidung durch den Angreifer (Sieht das Bild auffällig aus? Könnte dieses Histogramm des Bildes auf die Anwendung von Steganographie hinweisen?) bringen sie den Vorteil einer automatisierten Entscheidung. Die Qualität hängt natürlich vom Aufwand zur Ermittlung des Schwellwertes bzw. von der aufgestellten Hypothese ab.

Innerhalb dieses Versuchs sollen die Ergebnisse subjektiv und mit einem statitischen Test bewertet werden.

4. Steganographie mit Bilddaten

4.1. Darstellung der Bilder

Bilder, welche natürliche Szenen darstellen, werden oft als Coverdaten für Steganographie benutzt. Durch die notwendige Digitalisierung — sei es nun durch einen Scanner, mit dem Fotos digitalisiert werden, oder durch eine Digitalkamera — sind diese Bilder mit einem Rauschen behaftet. Das bedeutet, dass die exakten Farbwerte einem Angreifer nicht bekannt sind, was für Steganographie ausgenutzt werden kann.[4]

Es gibt unkomprimierte und komprimierte Bildformate, wobei die letzteren in verlustbehaftet und verlustfrei komprimierte unterteilt werden können. Bei den komprimierten Bildern muss so eingebettet werden, dass die Nachricht trotz Kompression erhalten bleibt. Beispielsweise wird im Algorithmus JSTEG, welcher auf der verlustbehafteten JPEG-Kompression aufsetzt, erst nach dem verlustbehafteten Teil der Kompression eingebettet.

Ein Bild b wird als eine Matrix mit M Spalten und N Zeilen betrachtet (Abbildung 2).

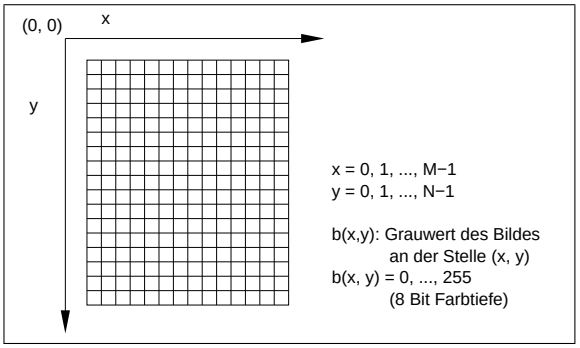

Abbildung 2: Darstellung von Bildern

[4] Im Gegensatz dazu ist die Verwendung von computergenerierten Zeichnungen mit tatsächlich einfarbigen Flächen weniger gut geeignet.

Ein Pixel (Bildpunkt, *„picture element“*) hat die Ortskoordinaten (x, y), dabei gibt x den Spalten- und y den Zeilenindex an. Der Bildursprung liegt in der linken oberen Ecke.

Im Versuch werden Grauwertbilder mit 8 Bit Farbtiefe betrachtet. Der Grauwert des Bildpunktes $b(x, y)$ ist in diesem Fall ein diskreter Wert aus dem Grauwertbereich $G = \{0, 1, ..., 255\}$, wobei 0 schwarz und 255 weiß entspricht. Es wird das unkomprimierte Format *„pgm“* verwendet. Dieses Format ist folgendermaßen aufgebaut: Zunächst werden in einem Header Informationen über das Bild, wie z. B. Höhe, Breite und Farbtiefe, angegeben. Anschließend werden die Grauwerte der Pixel fortlaufend spaltenweise von links nach rechts und zeilenweise von oben nach unten gespeichert.

Die Verteilung der Grauwerte eines Bildes wird mit Hilfe von Histogrammen beschrieben, welche die absoluten (Häufigkeitshistogramme) bzw. die relativen (Wahrscheinlichkeitshistogramme) Häufigkeiten der Grauwerte angeben.

4.2. Definition der Differenzbilder

Um die Auswirkungen steganographischer Operationen zu verdeutlichen, können Differenzen zwischen Cover- und Stegobild ausgewertet werden. Die Differenzen werden mit Hilfe von Differenzbildern visuell dargestellt. An zwei Testbildern (Abbildung 3) wird die Darstellung in diesem Kapitel erläutert.

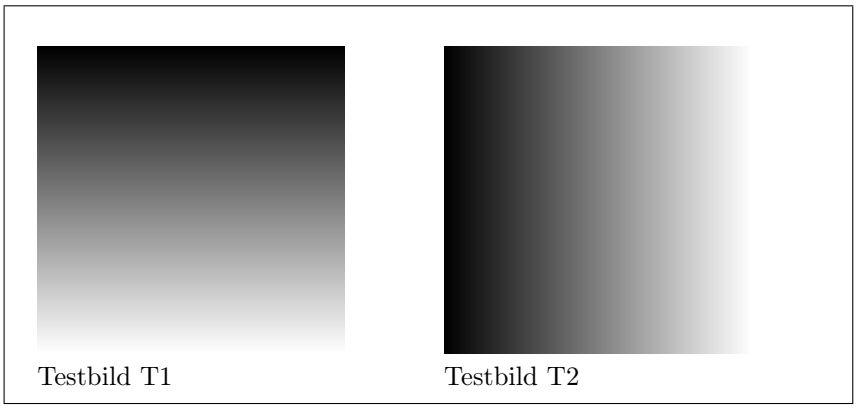

Abbildung 3: Testbilder zur Veranschaulichung der Differenzbilder

Beide Testbilder haben eine Größe von 256x256 Pixeln und enthalten alle 256 möglichen Graustufen. Das Testbild T1 enthält zeilenweise von oben nach unten die Grauwerte von 0 bis 255. Das Testbild T2 enthält die Grauwerte von 0 bis 255 spaltenweise von links nach rechts.

Die Differenzbilder werden ebenfalls als Grauwertbilder dargestellt, d. h. ihre Pixel können Werte von 0 bis 255 annehmen. Ein *absolutes Differenzbild* b_{diff_abs} zwischen zwei Bildern b_1 und b_2 ist folgendermaßen definiert:

$$b_{diff\_abs}(x, y) = |b_1(x, y) - b_2(x, y)|$$
$$x = 0, 1, ..., M - 1, \quad y = 0, 1, ..., N - 1.$$

Jedes Pixel des Differenzbildes entspricht der absoluten Differenz zwischen dem Pixel des ersten und dem Pixel des zweiten Bildes an dieser Position. Ein schwarzes Pixel (Grauwert 0) in einem Differenzbild bedeutet also, daß die Pixel der beiden Bilder an dieser Position übereinstimmen, während ein weißes Pixel mit dem Grauwert 255 für die maximale Differenz steht.

Die Differenzen sind meistens sehr gering. Da es nicht möglich ist, mit bloßem Auge geringe Unterschiede zwischen Grauwerten auszumachen, erscheint ein Differenzbild mit geringen Differenzen völlig schwarz. Um eine visuelle Auswertung zu ermöglichen, müssen die Differenzbilder aufbereitet werden.

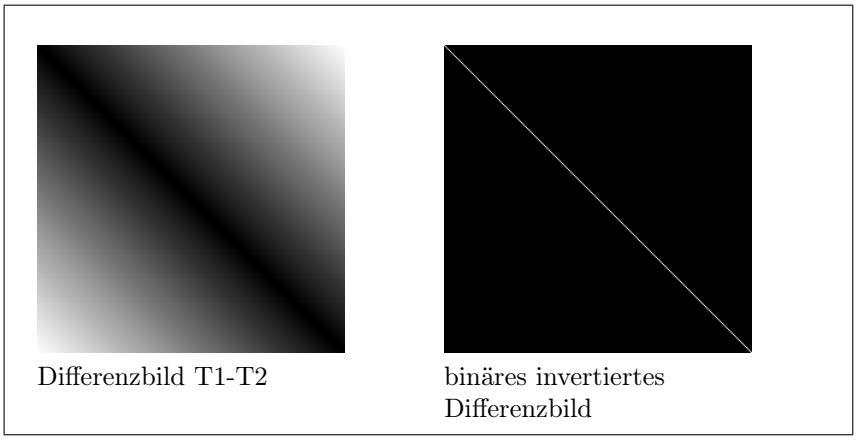

Abbildung 4: Differenzbild zwischen T1 und T2 und aufbereitete Darstellung

Dazu sollen hier *binäre invertierte Differenzbilder* b_{diff_b} verwendet werden. Diese Differenzbilder geben an, welche Pixel der beiden Bilder voneinander abweichen und liefern somit eine Aussage über das generelle Auftreten von Differenzen. Ein weißes Pixel bedeutet, daß die Pixel der verglichenen Bilder an dieser Position übereinstimmen, ein schwarzes Pixel bedeutet, daß irgendeine Differenz zwischen diesen Pixeln besteht:

$$b_{diff\_b}(x,y) = \begin{cases} 0 & : \quad b_1(x,y) \neq b_2(x,y) \\ 255 & : \quad \text{sonst} \end{cases}$$

$$x = 0, 1, ..., M-1, \quad y = 0, 1, ..., N-1$$

Abbildung 4 zeigt das Differenzbild zwischen den Testbildern T1 und T2 und die Darstellung als binäres invertiertes Differenzbild.

5. Einfache steganographische Algorithmen

5.1. LSB-Methode

Ein sehr einfacher und weit verbreiteter Ansatz für Steganographie ist das Überschreiben der niederwertigsten Bits („LSB-Methode“, *Least Significant Bit*). Beim Entwurf dieses Algorith-

mus' wurde die Zielstellung verfolgt, dass die Modifikationen der Coverdaten nicht wahrnehmbar sein sollen. Da die Grauwerte durch das beim Digitalisieren aufgebrachte Rauschen mit einem Meßfehler behaftet sind, gibt es keinen „exakten“ Grauwert. Vielmehr gibt es einen Toleranzbereich, in dem der Wert liegen kann. Daraus leitete man die Annahme ab, daß die Grauwerte innerhalb dieses Toleranzbereiches verändert werden können. Da das niederwertigste Bit den geringsten Anteil des Wertes kodiert (Abbildung 5), kann man annehmen, daß der exakte Wert dieses Bits aufgrund des Meßfehlers unbestimmt ist. Aufgrund dieser Annahme werden diese Bits bei der LSB-Methode einfach fortlaufend mit den Nachrichtenbits überschrieben.

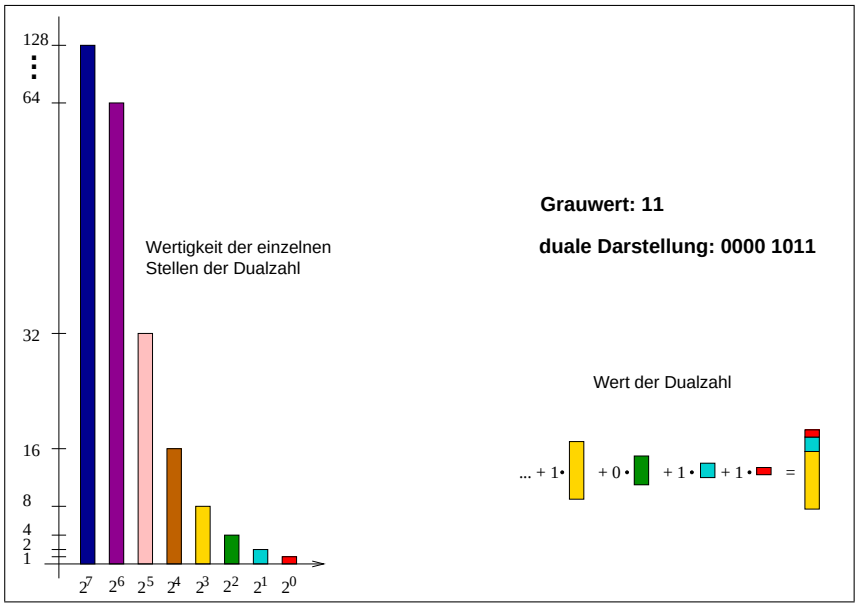

Abbildung 5: Anteil des niederwertigsten Bits am Gesamtwert

Das Überschreiben muss nicht immer zu einer Änderung des Grauwertes führen. Steht im niederwertigsten Bit des Grauwertes bereits das einzubettende Nachrichtenbit, so bleibt das Pixel unverändert. Das heißt, Einbetten einer Null in einen geraden Grauwert bzw. Einbetten einer Eins in einen ungeraden Grauwert führt zu keiner Änderung.

Die maximale Änderung des Grauwertes beträgt bei dieser Methode Eins: Beim Einbetten einer Null in einen ungeraden Grauwert wird dessen Betrag um Eins verringert, wogegen das Einbetten einer Eins in einen geraden Grauwert dessen Wert um Eins erhöht. Eine so geringe Veränderung des Grauwertes ist für das menschliche Auge nicht wahrnehmbar. Allerdings könnte ein Angreifer, der ein solches Stegobild abgefangen hat, einfach die niederwertigsten Bits der Pixel auslesen und testen, ob sich eine sinnvolle Nachricht ergibt. Falls diese nicht vorher verschlüsselt wurde, ist die Erkennung natürlich besonders einfach.

Diese Einbettungsmethode ist am Beispiel eines Minibildchens von 12 Pixeln in Abbildung 6 dargestellt.

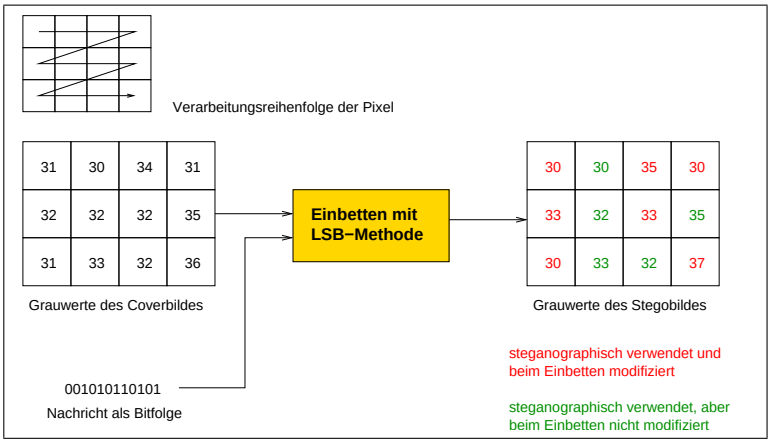

Abbildung 6: Einbetten mit der LSB-Methode

5.2. LSB-Methode mit schlüsselgesteuerter Pixelauswahl

Eine erste Verbesserung dieser Methode stellt die Benutzung von Schlüsseln zur Auswahl der zum Einbetten verwendeten Bildpunkte dar. Der Schlüssel wird dabei zur Steuerung der Abstände zwischen den Bildpunkten genutzt. Ein Angreifer kann nun nicht mehr einfach die niederwertigsten Bits auslesen, um die Nachricht zu erhalten. Der Schlüssel schützt die Information, welche Pixel tatsächlich steganographisch benutzt wurden. Auch bei dieser Methode beträgt die maximale Änderung Eins.

Abbildung 7 zeigt die Anwendung dieser Methode. Die Verarbeitungsreihenfolge entspricht der in Abbildung 6.

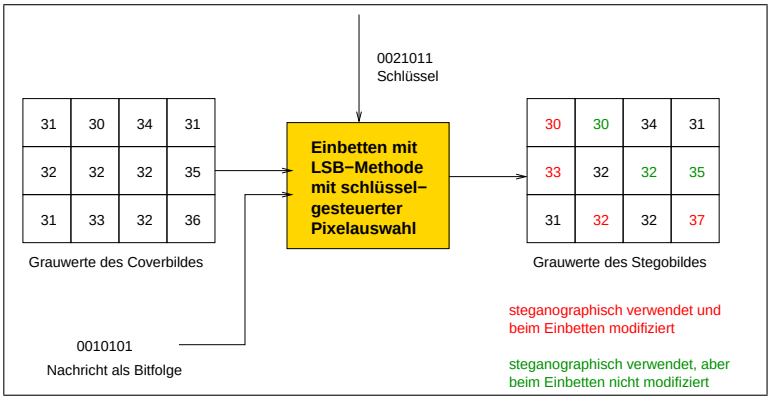

Abbildung 7: Einbetten mit der LSB-Methode mit schlüsselgesteuerter Pixelauswahl

5.3. Paritätskodierung

In den bisher vorgestellten Verfahren ist im Mittel die Änderung der Hälfte aller verwendeter Bildpunkte notwendig. Ross Anderson stellte in [1] ein verbessertes Verfahren vor, welches eine geringere Änderungsrate aufweist. Dabei werden n Pixel zusammengefasst. Das eingebettete Bit ergibt sich aus der Parität dieser Pixel bzw. ihrer niederwertigsten Bits. Entweder entspricht deren Parität schon dem einzubettenden Nachrichtenbit oder es muss maximal ein Pixel von insgesamt n verändert werden (Abbildung 8). Je größer die Länge n des Blocks, desto geringer ist die Änderungswahrscheinlichkeit. Zusätzlich zur geringeren Änderungsrate bietet das Verfahren den Vorteil, daß von den n Stellen die günstigste Stelle zum Ändern gewählt werden kann.

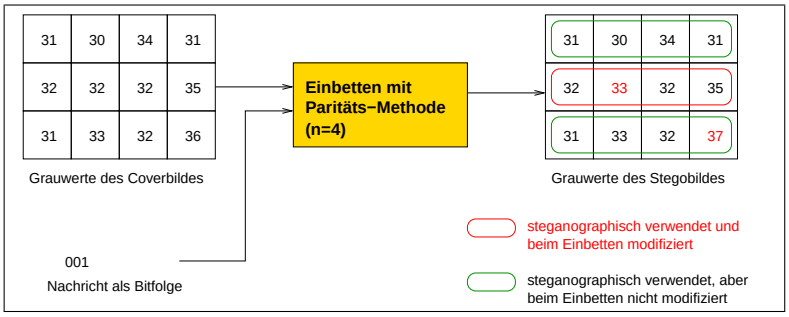

Abbildung 8: Einbetten mit der Paritäts-Methode

5.4. Inkrementieren

Das Einbetten mit der LSB-Methode ist zwar sehr einfach zu realisieren, in der Praxis aber unsicher. Durch das Ersetzen der niederwertigsten Bits kommt es zur Angleichung der Häufigkeiten von Grauwerten, die sich nur in diesem niederwertigsten Bit unterscheiden: Ein gerader Grauwert kann durch Einbetten einer Eins nur in den nächsthöheren ungeraden Grauwert übergehen, ein ungerader Grauwert durch das Einbetten einer Null nur in den nächstkleineren geraden. Wird eine gleichverteilte Nachricht eingebettet, kommt es deswegen zur Mittelung der Häufigkeiten dieser Grauwerte, es bilden sich Gruppen aus. Diese charakteristische Änderung wurde in [5] für den Entwurf eines statistischen Angriffs, des χ^2-Angriffs, genutzt.

Um diese auffälligen Veränderungen zu vermeiden, schlägt [5] vor, anstatt der LSB-Methode beispielsweise die Operation Inkrementieren zu verwenden. Stimmt das niederwertigste Bit nicht mit dem Nachrichtenbit überein, wird der Grauwert um Eins erhöht (Abbildung 9).

Aufgabe 1: Wie lang kann eine Nachricht maximal sein (ein Zeichen wird mit 8 Bit kodiert), wenn sie in das Testbild `dog.pgm` (Größe: 173×250 eingebettet werden soll bei

a) Verwendung der LSB-Methode und

b) Verwendung der Paritätskodierung für $n = 4$?

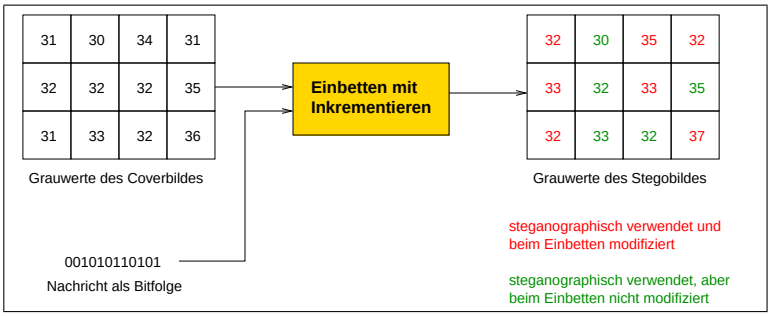

Abbildung 9: Einbetten mit Inkrementieren

Welches Problem ergibt sich bei einer kürzeren Nachricht für den Empfänger? Erarbeiten Sie Lösungsvorschläge!

Aufgabe 2: Welche Vorteile bringt der Einsatz steganographischer Schlüssel?

Aufgabe 3: Welche Probleme können beim Einbetten mittels Inkrementieren auftreten? Erarbeiten Sie Lösungsmöglichkeiten!

Aufgabe 4: Implementieren Sie die Funktionen Einbetten `embed` und Extrahieren `extract` für die folgenden Einbettungsoperationen.

Hinweis 1: Zum debuggen der Aufgabe kann es hilfreich sein vorher Aufgabe 6 zu lösen!

Hinweis 2: Der Parameter `percent` *gibt den Anteil des Bildes (in Prozent) an, der zum Einbetten benutzt werden soll. (Beispiel: Bei einer Bildgröße von n Pixeln und 25 Prozent Einbettung sollen die Pixel 1 bis* $\frac{n*25}{100}$ *verwendet werden.) Diese prozentuale Einbettung wird später für die Auswertung bzw. für Angriffe auf Stegobilder benötigt, insbesondere um Unterschiede zwischen Bildteilen mit und ohne Einbettung zu verdeutlichen.*

a) LSB-Methode (in der Klasse `EmbedLSB`),

b) LSB-Methode mit schlüsselgesteuerter Pixelauswahl (in der Klasse `EmbedKey`),

c) Paritätskodierung (in der Klasse `EmbedParity`),

d) Inkrementieren (in der Klasse `EmbedInc`).

Überprüfen Sie die Korrektheit Ihrer Algorithmen, indem Sie eingebettete und extrahierte Nachricht miteinander vergleichen.

Die Testumgebung ist im Anhang kurz beschrieben.

6. Bewertung der Einbettungsalgorithmen

6.1. Effizienz

Bei der Bewertung von Einbettungsalgorithmen können Effizienz und Sicherheit betrachtet werden. Folgende Parameter beschreiben die Effizienz eines Algorithmus':

- Die *Einbettungskapazität* ist hauptsächlich durch den Algorithmus vorgegeben. Sie gibt den Anteil der steganographisch *benutzbaren* Pixel des Coverbildes an. Beim Einbetten mit der LSB-Methode beträgt die Einbettungskapazität beispielsweise 100%, alle Pixel können steganographisch verwendet werden.

- Die *Einbettungsrate* gibt den Anteil der steganographisch *benutzten* Pixel des Coverbildes an. Sie kann maximal der Einbettungskapazität entsprechen.

- Bei manchen Einbettungsoperationen kann es zu Schwund kommen. Das ist immer dann der Fall, wenn der Empfänger nicht in der Lage ist, ein steganographisch benutztes Pixel als solches zu erkennen. Schwund kann demzufolge dann auftreten, wenn der Algorithmus eine Unterscheidung in steganographisch benutzbare und nicht benutzbare Pixel vornimmt. Wird ein benutzbares Pixel durch das Einbetten zu einem nicht benutzbaren, tritt Schwund auf. Die einfachste Lösung zum Ausgleich dieses Schwunds ist das erneute Einbetten. Die *Schwundrate* gibt den Anteil der steganographisch benutzten Pixel an, bei denen Schwund auftrat. (Bei den in diesem Versuch betrachteten Algorithmen kann Schwund nur beim Einbetten durch Inkrementieren auftreten.)

- Die *Änderungsrate* gibt den Anteil der Bildpixel an, die durch das Einbetten *verändert* wurden.

Aufgabe 5: Verändern Sie Ihre in Aufgabe 4 erstellten steganographischen Einbettungsalgorithmen so, dass die Effizienzparameter beim Einbetten berechnet und ausgegeben werden. (Die Extrahierungsalgorithmen aus Aufgabe 4 müssen dafür nicht verändert werden).

Benutzen Sie die Testumgebung, um mit diesen Algorithmen in verschiedene Testbilder jeweils 25% und 75% der möglichen Einbettungskapazität einzubetten.

Werten Sie die Effizienzparameter für die erstellten Stegobilder aus.

6.2. Sicherheit

Zur Bewertung der Sicherheit der Algorithmen werden mögliche Analysen der Daten betrachtet, die ein Angreifer durchführen könnte. Um die Auswirkungen der Einbettungsfunktionen, die innerhalb dieses Versuches erstellt werden sollen, zu verdeutlichen, werden sowohl Coverals auch Stegobilder ausgewertet.

Zur Analyse von Bildern ist oft Wissen über die Entstehung der Bilder sehr hilfreich. Betrachten wir darum die Entstehung der digitalen Bilddaten, die für das Einbetten benutzt wurden. Diese mußten zunächst digitalisiert werden, also beispielsweise eingescannt. (Eine andere mögliche Herkunft wäre eine Digitalkamera, doch innerhalb der Kamera findet natürlich wiederum eine Digitalisierung der „analogen Umwelt“ statt).

Jeder Digitalisierungsprozess ist mit einem Rauschen behaftet. Von dieser Tatsache ging man beim Entwurf der LSB-Methode aus. Wie im Abschnitt 5.1 beschrieben, kann man annehmen, daß die niederwertigsten Bits mit einem Rauschen überlagert sind. Daraus wurde die Annahme abgeleitet, dass diese Bits gleichverteilt sind. Darauf aufbauend ging man davon aus, daß man dieses weiße Rauschen in den niederwertigsten Bits der Grauwerte der Bildpunkte durch eine verschlüsselte Nachricht, die ja die stochastischen Eigenschaften von weißem Rauschen hat, problemlos ersetzen kann.

In der Praxis stellte sich aber heraus, dass es kein weißes Rauschen unabhängig vom Bildinhalt in den niederwertigsten Bits gibt. Vielmehr sind oft Strukturen erkennbar. Dieser Effekt läßt sich mit näheren Untersuchungen des Digitalisierungsprozesses erklären [2]. Diese Strukturen werden natürlich zerstört, wenn die niederwertigsten Bits einfach durch eine Nachricht in Form einer gleichverteilten Bitfolge ersetzt werden.

Aufgabe 6: Erstellen Sie Hilfsfunktionen zur Verdeutlichung der Auswirkungen des Einbettens:

a) Schreiben Sie ein Programm, mit dessen Hilfe Sie ein Bild erstellen können, welches die niederwertigste Bitebene eines Bildes repräsentiert. Implementieren sie hierfür die Methode
`showLSB(Image image)` in der Klasse `VisualAttack`.

b) Erstellen Sie ein Programm, welches Differenzbilder berechnet und ausgibt. Implementieren sie hierfür die Methode
`diffImg(Image image1, Image image2)` in der Klasse `VisualAttack`.

Aufgabe 7: Führen Sie verschiedene Analysen durch, um die Sicherheit der Einbettungsoperationen zu bewerten. Benutzen Sie für Ihre Analysen jeweils die Coverbilder und die Stegobilder der verschiedenen Algorithmen, bei denen 75% der Einbettungskapazität verwendet wurde:

a) Visualisieren Sie die LSB-Ebene der Cover- und Stegobilder und bewerten Sie die Ergebnisse.

b) Erstellen und bewerten Sie die Differenzbilder!

c) Führen Sie für Cover- und Stegobilder einen χ^2-Angriff durch und bewerten Sie die Ergebnisse. (Eine Beschreibung des χ^2-Angriff befindet sich im Anhang B.)

Literatur

[1] Ross J. Anderson: Stretching the Limits of Steganography. In Ross J. Anderson (Ed.): Information Hiding. First International Workshop, Cambridge, U.K., May/June 1996, Proceedings, Springer, LNCS 1174, 1996, 39-48.

[2] Elke Franz, Andreas Pfitzmann: Steganography Secure Against Cover-Stego-Attacks. In Andreas Pfitzmann (Ed.): Information Hiding. Third International Workshop, IH'99, Dresden, Germany, September/October 1999, Proceedings, Springer, LNCS 1768, 2000, 29-46.

[3] Andreas Pfitzmann: Sicherheit in Rechnernetzen: Mehrseitige Sicherheit in verteilten und durch verteilte Systeme. Vorlesungsskript, Institut für Systemarchitektur, Technische Universität Dresden, 2001.
http://dud.inf.tu-dresden.de/~pfitza/DSuKrypt.html

[4] Klaus Voss, Herbert Süße: Praktische Bildverarbeitung. Hanser-Verlag 1991.

[5] Andreas Westfeld, Andreas Pfitzmann: Attacks on Steganographic Systems. In Andreas Pfitzmann (Ed.): Information Hiding. Third International Workshop, IH'99, Dresden, Germany, September/October 1999, Proceedings, Springer, LNCS 1768, 2000, 61-76.

[6] Jan Zöllner, Hannes Federrath, Herbert Klimant, Andreas Pfitzmann, Rudi Piotraschke, Andreas Westfeld, Guntram Wicke, Gritta Wolf: Modeling the Security of Steganographic Systems. In: Information Hiding. Second International Workshop, IH'98, Portland, Oregon, USA, April 1998, Proceedings, Springer, LNCS 1525, 1998, 344-354.

A. Testdaten

Zur Durchführung des Versuchs werden Ihnen folgende Daten vorgegeben:

- Eine Datei `message.txt` mit Nachrichtentext.
- Eine Datei `key.txt` mit Schlüsseln (Integer-Werte).
- Mehrere Testbilder im Format `pgm`.
- Mehrere Java-Klassen, die als Programmierrahmen und Testumgebung für die zu implementierenden Algorithmen dienen. Die Klasse `tasks/Main.java` startet die Testumgebung. In der Testumgebung können in beide der gegebenen Bilder Nachrichten mit unterschiedlicher Kapazität (100%, 75%, 25%) eingebettet und extrahiert werden. Weiterhin können die visuellen Angriffe (Differenzbild, LSB-Ebene), und der χ^2-Angriff durchgeführt und deren Ergebnisse angezeigt werden. Es werden dabei jeweils die in den Aufgaben implementierten Algorithmen benutzt.

  (Achtung: Wenn die Algorithmen noch nicht implementiert sind, funktioniert die Testumgebung trotzdem. Sie zeigt dann bei den zu berechnenden Effizienzparametern jeweils -1 an, und die für die visuellen Angriffe erzeugten Bilder sind jeweils gleich dem Originalbild.)

Alle verwendeten Dateien müssen im dem Verzeichnis liegen, in dem das Programm ausgeführt wird (aktuelles Verzeichnis).

B. Der χ^2-Angriff (Chitest)

Nachdem in ein Bild eine Nachricht eingebettet wurde, kann darauf der χ^2-Angriff aus [5] ausgeführt werden (Taste "Chitest" in der Testumgebung). Als Ergebnis des Angriffs wird jeweils für den angegebenen Bildanteil die ermittelte Einbettungswahrscheinlichkeit ausgegeben.

Der χ^2-Angriff nutzt die Tatsache aus, dass sich die Häufigkeiten der Grauwerte, die sich nur im niederwertigsten Bit unterschieden, durch das Überschreiben mit der LSB-Methode an den Mittelwert dieser Häufigkeiten angleichen (Pärchenbildung). Damit kann eine Hypothese über die Verteilung der Grauwerte eines entsprechenden Stegobildes aufgestellt werden: Es wird angenommen, daß die Häufigkeit der Grauwerte eines solchen Pärchens jeweils dem Mittelwert der Häufigkeiten entspricht.

Zur Überprüfung der Hypothese verwendet der Angriff einen χ^2-Anpassungstest. Die Pixel des Bildes werden in Schritten von jeweils einem Prozent ausgewertet. Wird die Hypothese abgelehnt, wird vom Ende der eingebetteten Nachricht ausgegangen.