

Steganographie

Ingo Blechschmidt,
Michael Hartmann

LUGA

4. Oktober 2006

Inhalt

1 Grundlagen

- Definition der Steganographie
- Beispiele für Steganographie
- Historische Anwendungen der Steganographie
- Abgrenzung zur Verschlüsselung
- Mathematischer Hintergrund

2 Anwendungen

- Verzeichnisse
- Text
- Bilder
- Variation des TTL-Wertes
- Domain Name Service (DNS)
- Tastaturverzögerung

3 Quellen

- Literatur

Was ist Steganographie?

Definition

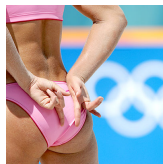
Steganographie ist die Wissenschaft der verborgenen Übermittlung von Informationen.

Einsatzgebiete:

- Sprache
- Digitale Medien
- Sport
- Kopierschutz

Beispiele für Steganographie

- Handzeichen beim Volleyball und Fußball
- Einsagen bei Ausfragen
- Versteckte Daten in Bildern, Audio- und Videodateien
- Umgehen von Servereinschränkungen (versteckte MP3 in Bildern)
- Passiver Kopierschutz → Wasserzeichen
- Verfassen von Briefen mit „Geheimtinte“



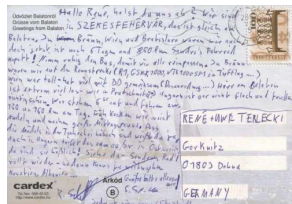
Historische Anwendungen

- Demeratus' Warnung an Sparta vor Angriff der Xerer
- Tätowierung von kahlrasierten Köpfen
- microdots: winzige Punkte, denen man keine Beachtung schenkt (1941 in Deutschland erfunden)
- Verbot von Postsendungen mit u.a. Schachaufgaben, Kreuzworträtseln, Zeitungsausschnitten, Strickmuster in den USA und England während des Zweiten Weltkriegs



Abgrenzung zur Verschlüsselung

- Erregung von Verdacht durch verschlüsselte Nachrichten
- Verheimlichung von Informationsübermittlung durch Steganographie
- Steganographie kaum erkennbar
- Einfache bis sehr komplexe steganographische Algorithmen



Mathematischer Hintergrund

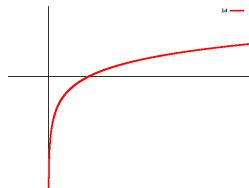
Informationsgehalt eines Zeichens

$$I(x) := -\log_2 P(x); \quad [1 \text{ bit}]$$

Entropie: Zu erwartender Informationsgehalt

$$E(I) := \sum_{x \in \Sigma} P(x) \cdot I(x);$$

- x : ein bestimmtes Zeichen
 Σ : Menge aller vorkommenden Zeichen
 $P(x)$: Wahrscheinlichkeit des Auftretens von x



Beispiel

Definitionen

$$I(x) := -\log_2 P(x);$$

$$E(I) := \sum_{x \in \Sigma} P(x) \cdot I(x);$$

$$M = (a, l, l, e, s);$$

$$\Sigma = \{a, e, l, s\};$$

$$I(a) = I(e) = I(s) = -\log_2 \frac{1}{5} \approx 2,32 \text{ bit};$$

$$I(l) = -\log_2 \frac{2}{5} \approx 1,32 \text{ bit};$$

$$E(I) \approx 3 \cdot \frac{1}{5} \cdot 2,32 \text{ bit} + \frac{1}{5} \cdot 1,32 \text{ bit} \approx 1,67 \text{ bit};$$

Eigenschaften der Entropie

- Je seltener ein Zeichen, desto höherer Informationsgehalt
- Spezialfall bei gleicher Häufigkeit aller Zeichen:

$$P(x_1) = P(x_2) = \dots = P(x_n) =: p$$

$$I(x_1) = I(x_2) = \dots = I(x_n) = -\log_2 p =: i$$

$$E(I) = i$$

Beispiel:

$$M = (1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4);$$

$$p = \frac{1}{4};$$

$$E(I) = i = -\log_2 \frac{1}{4} = 2 \text{ bit};$$

- Maximale Entropie bei $\Sigma = \{0, 1\}^8$:

$$E(I) = 8 \text{ bit};$$

Praktische Entropiebestimmung

Live-Demo

Praktische Entropiebestimmung

Entropie

Eingabe
Bohmsche Mechanik|
aus Wikipedia, der freien Enzyklopädie

Die Bohmsche Mechanik ist - je nach Definition der Begriffe - eine alternative Interpretation bzw. Modifikation der Quantenmechanik. Sie reproduziert alle Vorhersagen der (nicht-relativistischen) Quantenmechanik, hat aber ein radikal abweichendes Wirklichkeitsverständnis zur Grundlage. Die Bohmsche Mechanik ist eine deterministische Theorie und erlaubt eine einfache Lösung des Messproblems der Quantenmechanik, d.h. der Akt der Messung bzw. Beobachtung spielt keine

Informationsgehalt pro Zeichen

Zeichen	Unicode	Abs. Häufigkeit	Rel. Häufigkeit	Informationsgehalt in bit
!	U+0033	1	0,00004	14,50420
"	U+8220	1	0,00004	14,50420
..	U+8222	1	0,00004	14,50420
'	U+0039	2	0,00009	13,50420
?	U+0063	2	0,00009	13,50420
ρ	U+0961	2	0,00009	13,50420
+	U+0043	4	0,00017	12,50420
	U+0124	8	0,00034	11,50420
ψ	U+0968	8	0,00034	11,50420

Gesamtstatistik

Textlänge: 23238 B

Entropie: 4,59324 bit

Komprimiert mit gzip, Länge: 8627 B

Komprimiert mit gzip, auf Textlänge: 0,37125

Komprimiert mit gzip, Entropie: 7,97649 bit

Optionen

Normalisierung

Praktische Entropiebestimmung

- Beispielimplementierung `entropy.pl` zur Bestimmung von Informationsgehalt und Entropie

- Aufruf mit:

```
$ perl entropy.pl oder:
```

```
$ perl entropy.pl corpus.txt
```

- Benötigte nicht-Standard-Perl-Module:
Compress::Zlib, Gtk2

- Installation über:

```
# apt-get install libcompress-zlib-perl libgtk2-perl oder:
```

```
# perl -MCPAN -eshell
```

```
> install Compress::Zlib
```

```
> install Gtk2
```

Verzeichnisse als Trägermedium

```
$ ls -l
-rw-rw-rw- 1 user group 32372 01-Mathematik.txt
-rw-rw-rw- 1 user group 63814 02-Physik.txt
-rw-rw-rw- 1 user group  4831 03-Chemie.txt
-rw-rw-rw- 1 user group  1129 04-Biologie.txt
-rw-rw-rw- 1 user group 10037 05-Musik.txt
-rw-rw-rw- 1 user group 57001 06-Kunst.txt
```

Verzeichnisse als Trägermedium

```
$ ls -l
-rw-rw-rw- 1 user group 32372 01-Mathematik.txt
-rw-rw-rw- 1 user group 63814 02-Physik.txt
-rw-rw-rw- 1 user group  4831 03-Chemie.txt
-rw-rw-rw- 1 user group  1129 04-Biologie.txt
-rw-rw-rw- 1 user group 10037 05-Musik.txt
-rw-rw-rw- 1 user group 57001 06-Kunst.txt
```

Text als Trägermedium

Linux ist ein freies und plattformübergreifendes Betriebssystem, das UNIX ähnlich ist. Schon bald nach Beginn der Entwicklung liefen viele GNU-Tools, beispielsweise `cat`, `echo`, `bash` uvm. Nach kurzer Zeit wurde auch ein Maskottchen für das noch junge System gefunden: ein Pinguin.

Text als Trägermedium

Linux ist ein freies und plattformübergreifendes Betriebssystem, das UNIX ähnlich ist. Schon bald nach Beginn der Entwicklung liefen viele GNU-Tools, beispielsweise cat, echo, bash uvm. Nach kurzer Zeit wurde auch ein Maskottchen für das noch junge System gefunden: ein Pinguin.

- Relativ einfach zum Lesen und Erstellen
- Viele Algorithmen denkbar
- Kaum nachvollziehbar

Bilder als Trägermedium

- Verbergen von Daten durch Veränderung von Farben oder Verzerrung
- Farbänderung:
Nutzung der Least Significant Bits (LSB) als Träger → minimale (kaum wahrnehmbare) Veränderung des Bildes

$$10001010_2 (= 1 \cdot 2^7 + 1 \cdot 2^3 + 1 \cdot 2^1 = 138) \rightarrow$$
$$10001011_2 (= 1 \cdot 2^7 + 1 \cdot 2^3 + 1 \cdot 2^1 + 1 \cdot 2^0 = 139)$$

- Veränderung der LSB prinzipiell auch bei Audiodateien möglich
- Verzerren:
Geringfügige, systematische Verzerrung abhängig von den zu versteckenden Daten

stega.pl – Funktionsweise

- Beispielimplementierung stega.pl von <http://xrl.us/lugastega>
- Einbettungsalgorithmus:
Veränderung der drei LSB der Rot-, Grün- und Blaufarbwerte jedes Pixels (von links nach rechts und oben nach unten)
- Extraktionsalgorithmus:
Auslesen der drei LSB des Rot-, Grün- und Blaufarbwerte jedes Pixels (von links nach rechts und oben nach unten)
- Kapazität:

$$\frac{\text{Höhe} \cdot \text{Breite} \cdot 3 \text{ bit}}{8 \frac{\text{bit}}{\text{B}}}$$

(Höhe, Breite in Pixeln; z.B.: 50x50px: 0,9 KiB)

stega.pl – Nutzung

■ Daten in Bild verstecken:

```
$ ./stega.pl \  
    --mode=embed \  
    --input=originalbild.png \  
    --output=verändertes-bild.png
```

■ Daten aus Bild extrahieren:

```
$ ./stega.pl \  
    --mode=extract \  
    --input=verändertes-bild.png
```



unverändertes Bild



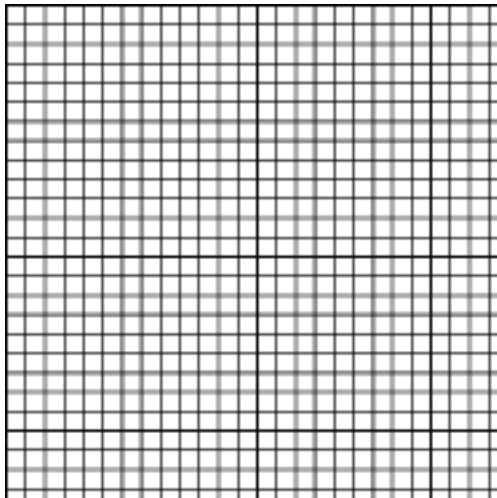
steganographisch verändertes Bild



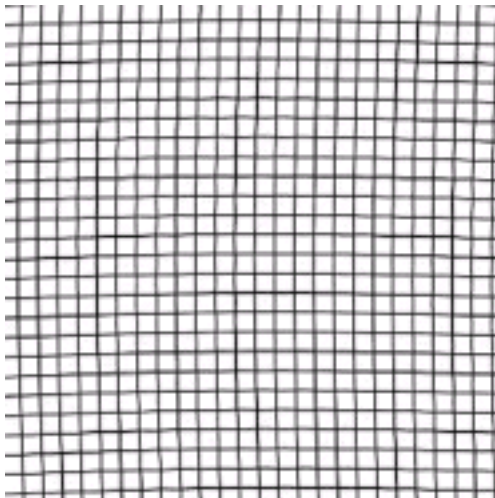
Lena, Original



Lena, steganographisch verändert



Raster, Original



Raster, steganographisch verändert

Beispielimplementierung

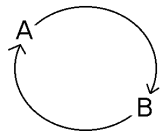
- steghide von Stefan Hetzl
(<http://steghide.sourceforge.net/>)
- Einbettung in Bilder (JPEG und BMP) und Audiodateien (WAV, AU)

```
1 steghide embed -cf  träger.jpeg -ef geheim.txt
```

```
2 steghide extract -sf träger.jpeg
```

Variation des TTL-Wertes von IP Paketen

- Time-to-Live: Zahl im Header von IP Paketen, die nach jedem passiertem Rechner dekrementiert wird
- Verhindern von „Endlosrouting“ durch TTL (A schickt Paket an B, B zurück zu A)
- Übermittlung von Daten durch unterschiedliche TTL-Werte in IP Paketen



Beispiel: Variation des TTL-Wertes

Variation des TTL-Wertes nach einem (hier sehr einfachen) Algorithmus:

- Zuordnung:

TTL des empfangenen Pakets \rightarrow Buchstabe im Alphabet

1 \mapsto a,

2 \mapsto b,

...

- Beispiel-Route: Sender \rightarrow Router \rightarrow Empfänger:

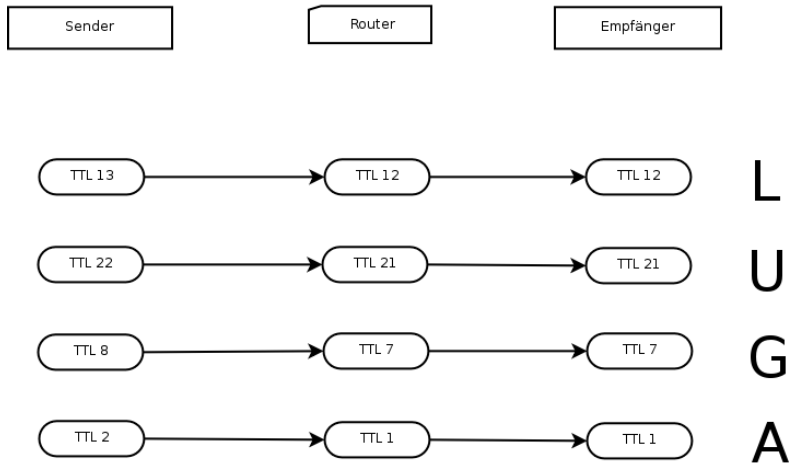
- 1 Sender schickt Paket mit TTL 13

- 2 Router dekrementiert TTL auf 12

- 3 Empfänger empfängt Paket mit TTL 12

- 4 Interpretiert Paket als „L“

Beispiel: Variation des TTL-Wertes



Domain Name Service (DNS)

1. Nicht-rekursiver	} Lookup auf nicht existente Domain	→	{ NOERROR := 0 NXDOMAIN := 1 NXDOMAIN := 1
2. Rekursiver			
3. Nicht-rekursiver			

- Setzen eines Bits:

```
$ dig @ns bit-addr +recursive
```

- Abfragen eines Bits:

```
$ dig @ns bit-addr +norecursive →
```

```
NOERROR? → 0
```

```
NXDOMAIN? → 1
```

- ... mit bit-addr beispielsweise

```
1234.pugs-rules.de für das 1234. Bit
```

Beispielimplementierung

Live-Demo

Beispielimplementierung

- Beispielimplementierung dnsx von `http://xrl.us/lugadnsx`
- Benötigtes nicht-Standard-Perl-Modul:
`Net::DNS`
- Installation über:

```
# apt-get install libnet-dns-perl oder:  
# perl -MCPAN -eshell  
> install Net::DNS
```


Speichern im DNS

```
$ ./dnsx \  
  --nameserver=Nameserveradresse \  
  --domain=Unbenutzte Domain \  
  --mode=store  
(dnsx liest von der Standardeingabe.)
```

Holen aus dem DNS

```
$ ./dnsx \  
  --nameserver=Nameserveradresse \  
  --domain=Unbenutzte Domain \  
  --mode=receive \  
  --length=Länge der Daten  
(dnsx schreibt auf die Standardeingabe.)
```

Anmerkungen zu dnsx

- Wahl einer unbenutzten Domain als `--domain` zwingend
- Keine „prinzipielle Korrektheit“ der Domain erforderlich;
beispielsweise Möglichkeit von `--domain=weltklasse.9live`
- Unterdrücken der Debuggingmeldungen durch Umleitung der Standardfehlerausgabe

Tastaturverzögerung mit JitterBugs

- Verbergen von Information durch systematische Verzögerung von Tastenanschlägen durch ein externes Gerät zwischen Tastatur und Computer
- Mathematisches Prinzip:

Δt : Zeitdauer zwischen zwei Anschlägen

T : Zeitfenster, bspw. 20 ms

Übertragung von 0?

→ Verzögerung so, dass Δt durch T teilbar

Übertragung von 1?

→ Verzögerung so, dass $\Delta t + \frac{T}{2}$ durch T teilbar

Beispiel

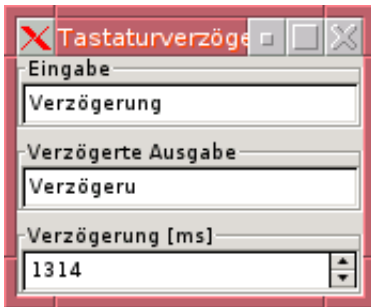
- Zeitdauer zwischen Anschlägen des Benutzers:
42 ms, 57 ms, 19 ms, 86 ms
- Zeitfenster: $T = 20$ ms
- Zu übertragende Information: 1011_2
- Zeitdauer zwischen den Anschlägen nach künstlicher Verzögerung:

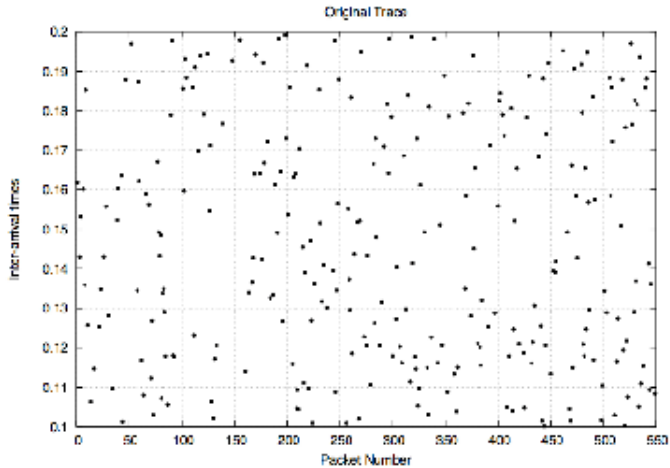
$\underbrace{60 \text{ ms}}_1, \underbrace{70 \text{ ms}}_0, \underbrace{20 \text{ ms}}_1, \underbrace{100 \text{ ms}}_1$

Trotz verschiedenen Quellen von Rauschen. . .

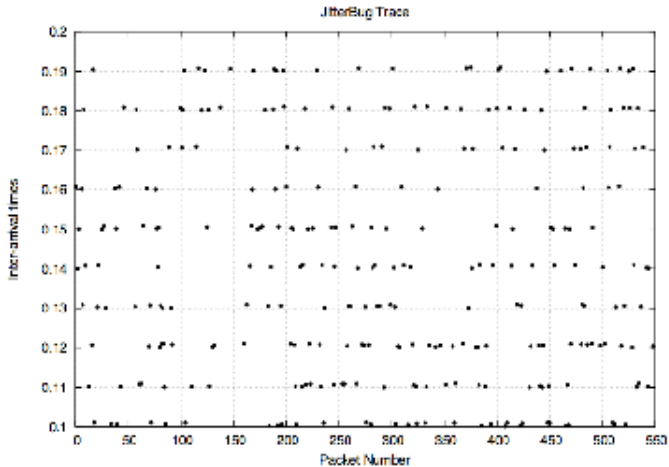
. . . Detektierbarkeit solcher Verzögerungen übers Netzwerk! (!!)

Live-Demo





Ohne JitterBugs



Mit JitterBugs

Fragen?

Literatur

- <http://arxiv.org/pdf/math.HO/0404335>
- <http://m19s28.vlinux.de/iblech/klasse11/Sonstiges/1337/LUGA/DNS.pdf.pdf>
- https://db.usenix.org/events/sec06/tech/shah/shah_html/index.html
- http://www.expmath.org/expmath/volumes/11/11.3/Calude361_370.pdf
- <http://www.petitcolas.net/fabien/publications/ieee99-infohiding.pdf>

Bildquellen

- <http://perl.plover.com/yak/presentation/samples/present.gif>
- https://db.usenix.org/events/sec06/tech/shah/shah_html/index.html
- <http://www.alte-geschichte.uni-hd.de/grafik/tempell.jpg>
- <http://www.htwm.de/~sstiller/kroatien/Postkarte%20Balaton%202.jpg>
- <http://www.meganandjack.com/mt/archives/signals4.jpg>
- <http://www.petitcolas.net/fabien/publications/ieee99- infohiding.pdf>

Bonus-Slides

- 4 Algorithmische Informationstheorie (AIT)
 - Algorithmische Information
 - Vergleich zur Entropie



Algorithmische Informationstheorie (AIT)

- Problem am Entropiekonzept:
Entropien von
00001111 und
01101000 gleich!
- Reihenfolge spielt beim Entropiekonzept keine Rolle!
- Lösung: Algorithmische Information –
Länge des kürzesten Programms einer
bestimmten Programmiersprache, die die
jeweilige Zeichenkette ausgibt

$$H(x) := \min \{|p| \mid M(p) = x\};$$

$M(p)$: Ausgabe des Programms m

Vergleich zur Entropie

- Beachtung der Reihenfolge durch die algorithmische Information
- Bessere Erfassung des anschaulichen Konzepts *Ordnung*:

00001111: 4 Nuller gefolgt von 4 Einsern

01101000: Eine Null, zwei Einsen, eine Null,
eine Eins, drei Nuller

- Allerdings: Praktische Bestimmung von $H(x)$ mitunter schwierig, da Ausprobieren sehr vieler Programme notwendig
- *Weitreichende, hier nicht näher genannte Konsequenzen für die Mathematik!*

Stichworte: G. Chaitin, CHAITINSche

Konstante/Haltewahrscheinlichkeit Ω , ...