Aufgabe: Eine Zufallsvariable nehme ihre 4 möglichen Werte mit den Wahrscheinlichkeiten $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8}$ an. Berechne die Shannon-Entropie, den Koinzidenzindex und die Renyi-Entropie.

Antwort: Shannon-Entropie: $-\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{4}\log_2\frac{1}{4} - \frac{1}{8}\log_2\frac{1}{8} - \frac{1}{8}\log_2\frac{1}{8}$

$$= \frac{1}{2} + \frac{2}{4} + \frac{3}{8} + \frac{3}{8} = \underline{\underline{1{,}75}}$$

Koinzidenzindex: $\left(\frac{1}{2}\right)^2 + \left(\frac{1}{4}\right)^2 + \left(\frac{1}{8}\right)^2 + \left(\frac{1}{8}\right)^2 = \frac{1}{4} + \frac{1}{16} + \frac{1}{64} + \frac{1}{64}$

$$= \frac{16+4+1+1}{64} = \frac{22}{64} = \frac{11}{32}$$

Renyi-Entropie: $-\log_2\left(\frac{11}{32}\right) = \log_2\left(\frac{32}{11}\right) = \ln\left(\frac{32}{11}\right)/\ln(2)$

$$= \underline{\underline{1{,}54}}$$

(Taschenrechner)

Aufgabe: Die Zufallsvariable X kann 2 verschiedene Werte annehmen; die Z.V. Y kann 3 Werte annehmen. Diese Werte(-kombinationen) treten mit folgenden W'heiten auf:

Y \ X		
	1/4	1/4
	1/4	0
	1/4	0
	3/4	1/4

1/2
1/4
1/4
1

← Randverteilungen

Man bestimme $H(X,Y)$, $H(X)$, $H(Y)$, $H(X|Y)$, $H(Y|X)$!

Antwort: $H(X,Y) = 4 \cdot \frac{1}{4} \log_2 4 = 1 \cdot 2 = \underline{2}$

$H(X) = \frac{3}{4} \cdot \log_2 \frac{4}{3} + \frac{1}{4} \log_2 4 = \underline{0{,}811}$

$H(Y) = \frac{1}{2} \log_2 2 + 2 \cdot \frac{1}{4} \log_2 4 = \frac{1}{2} + \frac{2}{4} \cdot 2 = \underline{1{,}5}$

$H(X|Y) = H(X,Y) - H(Y) = 2 - 1{,}5 = \underline{0{,}5}$

$H(Y|X) = H(X,Y) - H(X) = 2 - 0{,}811 = \underline{1{,}189}$

$I(X,Y) = H(X) + H(Y) - H(X,Y) = 0{,}811 + 1{,}5 - 2$

$= \underline{0{,}311}$

Frage: Man benötige für einen one-time pad Schlüssel 1000 echte Zufallsbits. Also $H(k) = 1000$ bit. Stattdessen wird ein Pseudozufallsbitgenerator benutzt, der mit einem „seed" von 100 zufälligen Bits initialisiert wird. Die notwendigen 1000 Bits werden daraus mittels eines deterministischen Algorithmus ermittelt. Welche Entropie hat diese Bitfolge von 1000 Bits?

Antwort: Leider nur 100 bit.

Frage: Sind die folgenden Zufallsvariablen X und Y unabhängig?

Y \ X	1	2
0	1/6	1/3
5	1/8	1/4
7	1/24	1/12

Aufwand: Zunächst die Randverteilungen bestimmen:

$$P(X=1) = \frac{1}{6} + \frac{1}{8} + \frac{1}{24} = \frac{1}{3}$$

$$P(X=2) = \frac{1}{3} + \frac{1}{4} + \frac{1}{12} = \frac{2}{3}$$

$$P(Y=0) = \frac{1}{6} + \frac{1}{3} = \frac{1}{2}$$

$$P(Y=5) = \frac{1}{8} + \frac{1}{4} = \frac{3}{8}$$

$$P(Y=7) = \frac{1}{24} + \frac{1}{12} = \frac{1}{8}$$

Dann überprüfen, dass die 6 Einträge in der Tabelle die Formel

$$P(X=x, Y=y) = P(X=x) \cdot P(Y=y)$$

erfüllen! Trifft zu.

Unizitätsmaß

Maßzahl für die Sicherheit eines Kryptosystems (basierend auf Shannon-Entropie).

Annahmen: Schlüssellänge fest, aber Klartextlänge und Chiffretextlängen können beliebig groß werden (skalierbar). n = Klartextlänge = Chiffretextlänge.

Notation: (K, M_n, C_n)

Das Unizitätsmaß eines Kryptosystems (K, M_n, C_n) ist das kleinste n, so dass bei Kenntnis von C_n der Schlüssel K (in informationstheoret. Hinsicht) eindeutig feststeht:

$$H(K \mid C_n) = 0.$$

Das heißt noch lange nicht, dass man den Schlüssel aus der Chiffre effizient berechnen kann.

Beispiel: Gegeben sei die Cäsar-Chiffre NRJ.
Es gibt nur 26 (bzw. 25) mögliche Schlüssel.
Probieren wir sie alle durch:

N	R	J
O	S	K
P	T	L
Q	U	M
R	V	N
S	W	O
T	X	P
U	Y	Q
V	Z	R
W	A	S
X	B	T
Y	C	U
Z	D	V
A	E	W
B	F	X
C	G	Y
D	H	Z
E	I	A
F	J	B
G	K	C
H	L	D
I	M	E
J	N	F
K	O	G
L	P	H
M	Q	I

nur 1 sinnvoller Klartext

Daher nur 1 möglicher Schlüssel.

Verschiedene Entropie-Berechnungen: Aus Experimenten und Schätzungen weiß man: $H(M_n) \approx 1{,}5 \cdot n$

Ferner: $H(K) = \log_2 |K|$ bei Gleichverteilung.

$$H(M,K,C) = H(M,K) + \underbrace{H(C|M,K)}_{=0} = H(M,K) = H(M) + H(K)$$

wegen Unabhäng.

$$H(M,K,C) = H(K,C) + \underbrace{H(M|K,C)}_{=0} = H(K,C)$$

Zusammengefasst/Eingesetzt:

$$H(K|C) = H(K,C) - H(C) = H(M,K,C) - H(C)$$

$$= H(M) + H(K) - H(C)$$

Obige Werte einsetzen:

$$H(K|C_n) = 1{,}5 \cdot n + \log|K| - \overbrace{H(C_n)}^{\text{idealisiert: } = n \cdot \log 26 = 4{,}7 \cdot n}$$

$$H(K|C_n) = \log_2|K| - 3{,}2 \cdot n$$

$\overset{!}{=} 0$

Für die Unizitätsdistanz ergibt sich also:

$$n = \frac{\log|K|}{3{,}2}$$

Bsp: Cäsar: $|K| = 26$ $\log 26 = 4{,}7$

$n = \frac{4{,}7}{3{,}2} = 1{,}5$ (n zu klein für die

Schätzung 1,5 bit/Buchstabe,
Idlke etwas höher liegen, etwa $n=3$)

Bsp: Vigenère-Schlüssel der Länge t:

$$|K| = 26^t \qquad \log(26^t) = 4{,}7 \cdot t$$

$$\Rightarrow n = \frac{4{,}7}{3{,}2} \cdot t = 1{,}5 \cdot t$$

Unizitätsmaß nur aus historischen Gründen zitiert.

Bzgl. Kryptoanalyse sehr unterschiedliche Kryptoverfahren können trotzdem sehr ähnliche Unizitätsmaße haben.

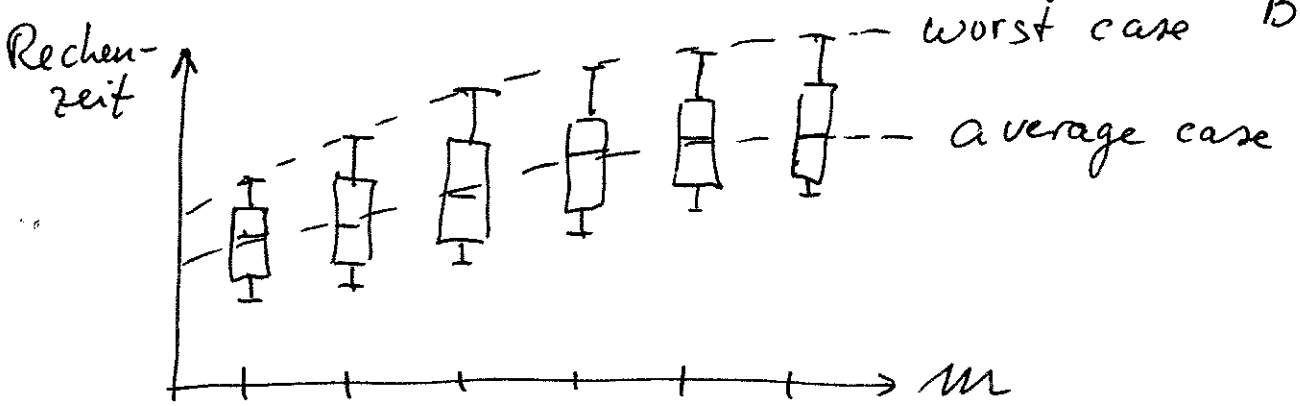
Grundsätzlich: $\frac{\log|K|}{3{,}2}$ wird umso größer

bei: - großer Anzahl Schlüssel

- Nenner wird kleiner, wenn weniger Entropie im Klartext: Dies ggf. ausgleichen durch Daten-Kompression vor der eigentlichen Verschlüsselung

Moderne Kryptographie bezieht Sicherheit nicht aus informationstheoretischen sondern aus komplexitätstheoretischen Überlegungen.

Komplexität eines Algorithmus' ≙ Anzahl Rechenschritte, als Funktion der Eingabelänge n (in Bits)



Bei vielen Algorithmen gibt es keinen Unterschied zwischen worst-case und average-case (z.B. schriftliche Addition / Multiplikation nach Schulmethode). „stereotype Algorithmen"
Bei anderen Algorithmen gibt es sehr große Unterschiede (z.B. Faktorisierungsalgorithmen)

Kryptographische Sicherheit sollte eher average-case als den worst-case betrachten (womöglich sogar den „best-case").

Oft werden Komplexitätsangaben in der O-Notation angegeben: z.B. $O(n^2)$

$O(f(n))$ bezeichnet die Klasse aller Funktionen $g(n)$ mit der Eigenschaft:

$$\exists c>0 \quad \underbrace{\exists n_0 \quad \forall n \geq n_0}_{\text{bedeutet:}} : \quad g(n) \leq c \cdot f(n)$$

bedeutet:
- für alle bis auf endlich viele n
- für fast alle n

$O(f(n))$ gibt eine asymptotische obere Schranke an unter Ignorieren von konstanten Faktoren und Termen kleinerer Ordnung: Beispiel:

$$5n^2 + 2n \cdot \log n + 7n + \log n \underset{\uparrow}{=} O(n^2)$$

eigentlich sollte man hier „$\in$" schreiben.

Verschlüsselungsverfahren der klassischen Kryptographie sind im Allgemeinen effizienter als in der modernen Krypto.

typisches
Beispiel:

$O(n)$ in der klassischen Krypto.

$O(n^3)$ in der modernen Krypto.

Für den berechtigten Krypto-Teilnehmer sollten die Algorithmen möglichst effizient sein.

Für den Unberechtigten (Kryptoanalytiker) sollte das Dechiffrieren möglichst ineffizient sein (z.B 2^n) – und das auch im average-case. Nicht einmal Teile der Nachricht / des Schlüssels sollten mit vertretbarem Aufwand berechenbar sein. (Die Standard-Komplexitätstheorie stellt für solches Anliegen keine entsprechenden Definitionen bereit.)

Recht grobe Einteilung der Komplexitätstheorie:

„effizient" wird gleichgesetzt mit polynomialer Rechenzeit, also $O(n^k)$, k Konstant.

Mit **P** fasst man alle algorithmischen Problemstellungen zusammen, für die polynomiale Algorithmen existieren.

Liste von effizient lösbaren Aufgabenstellungen:

(modulare) Addition, Multiplikation, Exponentiation

größten gemeinsamen Teiler berechnen,

multiplikative Inverse berechnen,

Chinesische Restsatz - Transformation

modulo einer Primzahl: Quadratwurzeln berechnen

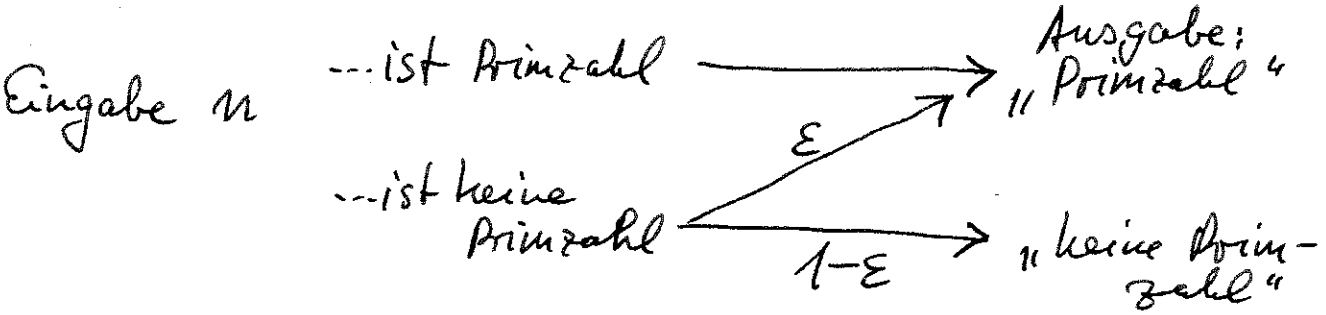
Primzahleigenschaft feststellen

Für folgende Aufgaben sind bisher keine effizienten Algorithmen bekannt (nur ineffiziente, z.B. Laufzeit c^n, $c>1$) — allerdings auch kein Beweis, dass keine effiziente Algorithmen existieren:

- Eine Zahl n $(=p \cdot q)$ faktorisieren
- den diskreten (modularen) Logarithmus berechnen.
- die NP-vollständigen Probleme
- Graphisomorphie
- Modulo n $(=p \cdot q)$ Quadratwurzel bestimmen

Probabilistische Algorithmen können oft effizientere Aufgaben lösen als dies deterministische können – auf Kosten einer kleinen Fehler / Misserfolgs-Quote.

Beispiel: Primzahl-Testen



Die Wahrscheinlichkeit ε kann (auf Kosten etwas größerer Rechenzeit) beliebig klein gemacht werden.

Algorithmus t-mal wiederholen. Wenn mindestens 1x die Antwort „keine Primzahl", dann ist n sicher keine Primzahl.

Ausgabe „Primzahl" ist dann korrekt mit Wahrscheinlichkeit $\geq 1 - \varepsilon^t$