Einführung in die Kryptologie

3. Mathematische Basis

Lorenz Müller
BFH / HTI
Quellgasse 21
2505 Biel

Block- und Stromchiffren > Lernthemen:

Blockchiffren

- **Schema**
- **Beispiel**
- **Betriebsmoden**

Stromchiffren

- **Schema**
- **Schieberegister**
- **Beispiel**

Theoretische Grundlagen

- **Elemente der Zahlentheorie**
- **Informationstheorie**
- **Eindeutigkeitsdistanz**
- **Komplexitätstheorie**
- **Einwegfunktionen**
- **Sicherheit von Kryptosystemen**

Restklassen

- **Kongruenz mod n**

$$a \equiv_n b \quad \therefore \quad a = b \bmod n$$

- **Restklassen**

$$Z_n = \{0,1,2,3 \ldots n-1\}$$

- **Restklassenring**

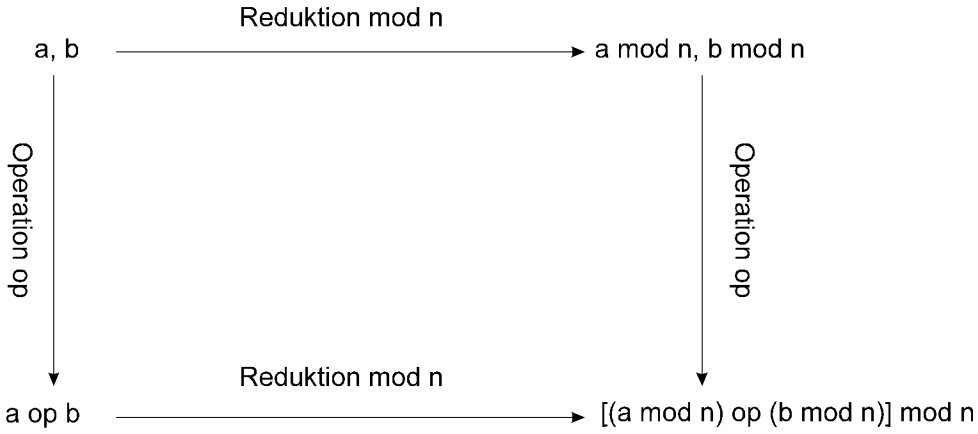
$$< Z_n, +, \cdot >$$

Modulare Arithmetik

Es gilt $(a + b) \bmod n = (a \bmod n + b \bmod n) \bmod n$

$(a.b) \bmod n = [(a \bmod n) \cdot (b \bmod n)] \bmod n$

$(a^b) \bmod n = (a \bmod n)^b \bmod n$



Beispiel. Neuner Probe

$n \in N$	$n \bmod 9$
124657	7
356821	7
33321	3
514799	8

Verkettung

$25\,binär: 11001\,dh.\, 25 = 2^4 + 2^3 + 2^0$

$a^{25} \bmod n = (a * a^8 * a^{16}) \bmod n = (a * ((a^2)^2)^2 * (((a^2)^2)^2)^2 \bmod n = [((a^2)^2)^2 \bmod n * (((a^2)^2)^2 \bmod n)^2 \bmod n * a] \bmod n$

Multiplikativ Inverses

- **Inverses in Z_n, diophantische Gleichung**

$$a \cdot x \bmod n = 1$$
$$ax - yn = 1$$

- **Relativ prim**

$$ggt(a,n) = 1 \quad \Leftrightarrow \quad \exists\, a^{-1} \quad \text{so dass} \quad a \cdot a^{-1} \bmod n = 1$$

- **Euklidischer Algorithmus**

$$n > a: \quad ggt(a,n) = ggt(n \bmod a, a)$$

Multiplikativ Inverses

Existenz des multiplikativ Inversen

Falls es Zahlen (x,y) so gibt, dass $ax - yn = 1$ existiert ein in Z_n zu a multiplikativ Inverses x.

Die Bedingung der diophantischen Gleichung ist gleichbedeutend wie $ggt(a,n) = 1$

Euklidischer Algorithmus zur Bestimmung des ggt und des mult. Inversen (falls ggt=1)

Euclid(d, f) $d < f$

1. $X \leftarrow f$; $Y \leftarrow d$
2. If $Y = 0$ Then $X = \gcd(d, f)$
3. $R = X \bmod Y$
4. $X \leftarrow Y$
5. $Y \leftarrow R$
6. Goto 2

Extended Euclid(d, f) $d < f$

1. $(X_1, X_2, X_3) \leftarrow (1,0, f)$; $(Y_1, Y_2, Y_3) \leftarrow (0,1, d)$
2. If $Y_3 = 0$ Then $X_3 = \gcd(d, f)$; kein Inverses
3. If $Y_3 = 1$ Then $Y_3 = \gcd(d, f)$; $Y_2 = d^{-1} \bmod f$
4. $Q = \left[\frac{X_3}{Y_3} \right]$
5. $(T_1, T_2, T_3) \leftarrow (X_1 - Q \cdot Y_1, X_2 - Q \cdot Y_2, X_3 - Q \cdot Y_3)$
6. $(X_1, X_2, X_3) \leftarrow (Y_1, Y_2, Y_3)$
7. $(Y_1, Y_2, Y_3) \leftarrow (T_1, T_2, T_3)$
8. Goto 2

Beispiel: Multiplikativ Inverses zu 53 mod 120

	Extended Euklid Algorithmus									
Step	Q	X1	X2	X3	Y1	Y2	Y3	T1	T2	T3
0	-	1	0	120	0	1	53	1	-2	14
1	2	0	1	53	1	-2	14	-3	7	11
2	3	1	-2	14	-3	7	11	4	-9	3
3	1	-3	7	11	4	-9	3	-15	34	2
4	3	4	-9	3	-15	34	2	19	-43	1
5	1	-15	34	2	19	-43	1			
					X3(0)*Y1	Y3(0)*Y2				
Test	53*-43 mod 120		-2279	1	120*19	53*-43				
oder	53*77 mod 120		4081	1	2280	-2279				

Reduzierte Restklassenmenge

$$Z_n^* = \{a \in Z_n | ggt(a,n) = 1\}$$

- **Euler-Phi Funktion gibt Anzahl Restklassen mit Inversem**

p Primzahl

$\phi(p) = (p-1)$

$n = p \cdot q \quad p,q$ Primzahlen

$\phi(n) = \phi(pq) = \phi(p)\phi(q) = (p-1)(q-1)$

- **Euler-Fermat Satz**

$$a^{\phi(n)} \bmod n = 1$$

$$ax \bmod n = 1 \quad \Leftrightarrow \quad x = a^{\phi(n)-1} \bmod n$$

Reduzierte Restklassenmenge

Multiplikationstabelle von Z^*_{11}

*	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	1	3	5	7	9
3	3	6	9	1	4	7	10	2	5	8
4	4	8	1	5	9	2	6	10	3	7
5	5	10	4	9	3	8	2	7	1	6
6	6	1	7	2	8	3	9	4	10	5
7	7	3	10	6	2	9	5	1	8	4
8	8	5	2	10	7	4	1	9	6	3
9	9	7	5	3	1	10	8	6	4	2
10	10	9	8	7	6	5	4	3	2	1

Euler-Phi Funktion: Anzahl Elemente in der mult. Gruppe Z^*_n

Nutzung der Euler-Phi Funktion zur Bestimmung des multiplikativ Inversen und der Wurzel

$$a^{\phi(n)} \bmod n = 1$$

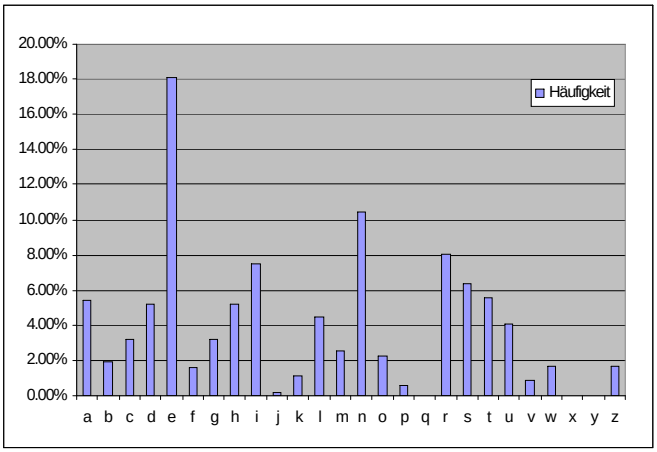
$$ax \bmod n = 1 \quad \Leftrightarrow \quad x = a^{\phi(n)-1} \bmod n$$

$$(a^e)^x \bmod n = a = a^{1+k*\phi(n)} \bmod n$$

$$ex = 1 \bmod \phi(n); \quad ggt(e,\phi(n)) = 1; x = e^{-1} \bmod \phi(n)$$

Zeichenhäufigkeit in Sprache

Redundanz deutsche Sprache



Beispiel:
Redundanz der alphabetischen Codierung der natürlichen Sprache

Der folgende Text mit rund 45 % weniger Zeichen als mit der normalen Syntax und Ortographie illustriert die Redundanz der geschriebenen natürlichen Sprache:

Etropi ein Nchritrms M brchnt as Whrschktn n-Gram im zgöhrign Alpabt, wo n imr grösr wrdn läst, bis im Przip lngst Nchrit ercht ist. In Prxs wrd skzsiv Brchng Etropi n-Gram bstn Lnge n bstit. Für engl Sprc ergbt Rdundz vn 3.2 bits/Bchstb.

Rdundnz Sprch

Rdundnz Sprch ist Dfrnz zwschn maxml möglch und realstn Infrmtnsghlt Sprch

272 Zeichen

Derselbe Text in normaler Schreibweise enthält keine zusätzliche Information

Die Entropie eines Nachrichtenraumes M berechnet sich aus den Wahrscheinlichkeiten der n-Gramme im zugehörigen Alphabet, wobei man n immer grösser werden lässt, bis die im Prinzip längste Nachricht erreicht ist. In der Praxis wird dies durch sukzessive Berechnung der Entropie von n-Grammen einer bestimmten Länge n bestimmt. Für die englische Sprache ergibt dies eine Redundanz von 3.2 bits/Buchstabe.

Redundanz einer Sprache

Die Redundanz einer Sprache ist die Differenz zwischen dem maximal möglichen und dem realisierten Informationsgehalt einer Sprache

480 Zeichen

Informationstheorie: Möglichkeit eines Angriffs

- **Informationsgehalt einer Nachricht mit p(*m*)**

$$I(m) := \log_2 \frac{1}{p(m)} = -\log_2 p(m)$$

==> I(*m*): Länge eines optimalen Suchbaumes um *m* in *M* zu identifizieren

- **Entropie einer Nachrichtenmenge**

$$H(M) := \sum_{m \in M} p(m) \log_2 \frac{1}{p(m)} = -\sum_{m \in M} p(m) \log_2 p(m)$$

==> H(*m*): Mittlere Länge aller opt. Suchbäume um eine bel. Nachricht in *M* zu identifizieren

Anwendung der Informationstheorie

Grundbegriff Information

Der Begriff ***Information*** nach Shannon bezeichnet die minimale Anzahl von Fragen, die mit ja/nein beantwortet werden, um ein bestimmtes Element m in einer (geordneten) Menge zu identifizieren. ***I(m)*** entspricht der Länge des idealen binären Suchbaumes für ***m***. Es folgt daraus, dass

$$I(m) = \log_2 \frac{1}{p(m)}$$

Beispiel: Information durch Münzenwurf m=Kopf oder m=Zahl

$$I(Kopf) = \log_2 \frac{1}{p(Kopf)} = \log_2 2 = 1$$

Einheit der Information

Einheit der Information ist das *bit.* 1 bit Information entspricht der Kenntnis einer ja/nein Antwort über zwei gleichwahrscheinliche Ereignisse.

Entropie

Entropie bezeichnet die mittlere Information, die beim Auftreten einer Nachricht (Element) ***m*** aus ***M*** mit zugeordneter Auftretenswahrschweinlichkeit ***p(m)*** anfällt:

$$H(M) := \sum_{m \in M} p(m) \log_2 \frac{1}{p(m)} = -\sum_{m \in M} p(m) \log_2 p(m)$$

Oder umgekehrt kann die Entropie H(M) als mittlere ***Unsicherheit*** über die vom Sender ausgewählte und übertragene Nachricht aus ***M*** interpretiert werden.

Entropie und Redundanz eines Alphabets

- **Entropie eines Huffmann-Codes mit 26 Zeichen**

$$H(A_{ideal}) = \sum_{i=1}^{26} \frac{1}{26} \log_2 26 = 4.7 \quad bits = R$$

- **Entropie eine Nachrichtenraumes *M* (nat. Sprache)**

$$H(M) = \lim_{n \to \infty} \frac{H(n-Gramme)}{n} = 1.2 - 1.7 \quad bits = r$$

- **Redundanz eines Nachrichtenraumes *M***

$$D := H(M_{ideal}) - H(M) = \log_2 |M| - r$$
$$\cong 3.0 - 3.2 \; bits / Zeichen$$

Informationsgehalt eines Codes

Ideal codiertes Alphabet

Bei idealer Codierung in einem Alphabet mit n Zeichen, bezeichnet jedes Zeichen eine Nachricht(ensequenz), die mit gleicher Wahrscheinlichkeit wie alle anderen auftritt

$$p(m) = 1/n$$

Jedes Zeichen hat damit gleichen Informationsgehalt und die Entropie eines solchen Huffman-Codes ist dann

$$H(A_{ideal}) = \sum_{i=1}^{n} \frac{1}{n} \log_2 \frac{1}{1/n} = \log_2 n \quad bits = R$$

Entropie eines Nachrichtenraumes

Die Entropie eines Nachrichtenraumes M berechnet sich aus den Wahrscheinlichkeiten der n-Gramme im zugehörigen Alphabet, wobei man n immer grösser werden lässt, bis die im Prinzip längste Nachricht erreicht ist. In der Praxis wird durch sukzessive Berechnung von bestimmt. Für die englische Sprache ergibt dies

$$H(1-gramme) \cong 4.15 \, bits / Zeichen$$
$$H(2-gramme)/2 \cong 3.62 \, bits / Zeichen$$
$$H(3-gramme)/3 \cong 3.22 \, bits / Zeichen$$
$$H(n-gramme)/n \cong 1.5 \, bits / Zeichen$$

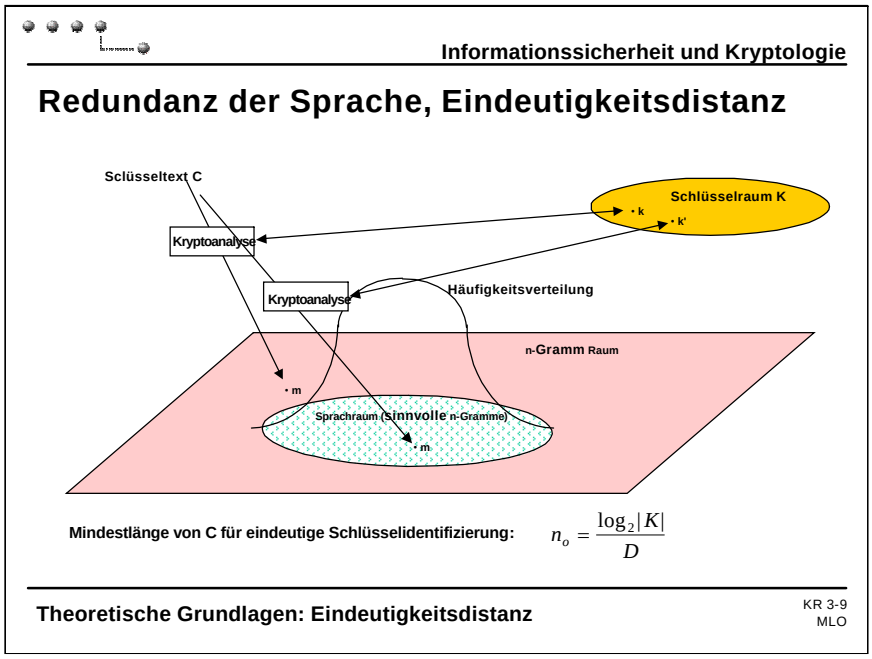
und führt zu den Grenzwerten

$$H(M_{engl}) \cong 1.5 \, bits / Zeichen = r \qquad H_n / n$$
$$H(M_{dtl}) \cong 1.7 \; bits / Zeichen = r$$

Redundanz einer Sprache

Die Redundanz einer Sprache ist die Differenz zwischen dem maximal möglichen und dem realisierten Informationsgehalt einer Sprache

$$D := H(M_{ideal}) - H(M) = \log_2 |A| - H(M) = R - r \cong 3.0 - 3.2 \, bits / Zeichen$$



Voraussetzungen und Möglichkeiten der Kryptoanalyse

Vorkenntnisse und Redundanz einer Sprache

Natürliche, formale oder technische (zB. Geldabruf in ATM mit Pincode) Sprachen bilden eine (kleine) Teilmenge aller möglichen n-Gramme in einem Alphabet

$$Sprache \subset \{n-gramme | n = 1,2,3...\}$$

Der Kryptoanalytiker muss über **apriori Wissen** verfügen, das heisst

- den Kontext (Algorithmus, Anwendungsumfeld) des Kryptogramms und
- die Sprache des zugrundeliegenden Klartextes kennen sowie
- über verlässliche n-Gramm Statistiken für die betreffende Sprache verfügen.

Beim Angriff wird die redundante Information einer Sprache ausgenutzt, die durch die Verschlüsselung des Klartextes in den Schlüsseltext einfliesst. In der Praxis werden dabei die n-Gramm (n<6) Häufigkeiten analysiert, um den Klartext oder den Schlüssel zu finden. Die Diffusion hat zum Ziel n-Gramme aufzubrechen und damit den rein statistischen Angriff abzuwehren

Eindeutigkeitsdistanz

Für die Durchführung von statistischen Analysen muss genügend Schlüsseltext C vorhanden sein. Die minimal notwendige Länge n_0<|C| um einen Schlüssel eindeutig zu identifizieren, bezeichnet man als **Eindeutigkeitsdistanz**. Sie hängt vom Kryptosystem (Grösse des Schlüsselraumes) und von der verwendeten Sprache (Redundanz) ab.

Eine mathematische Analyse zeigt, dass die Eindeutigkeitsdistanz durch

$$n_o = \frac{\log_2 |K|}{D}$$

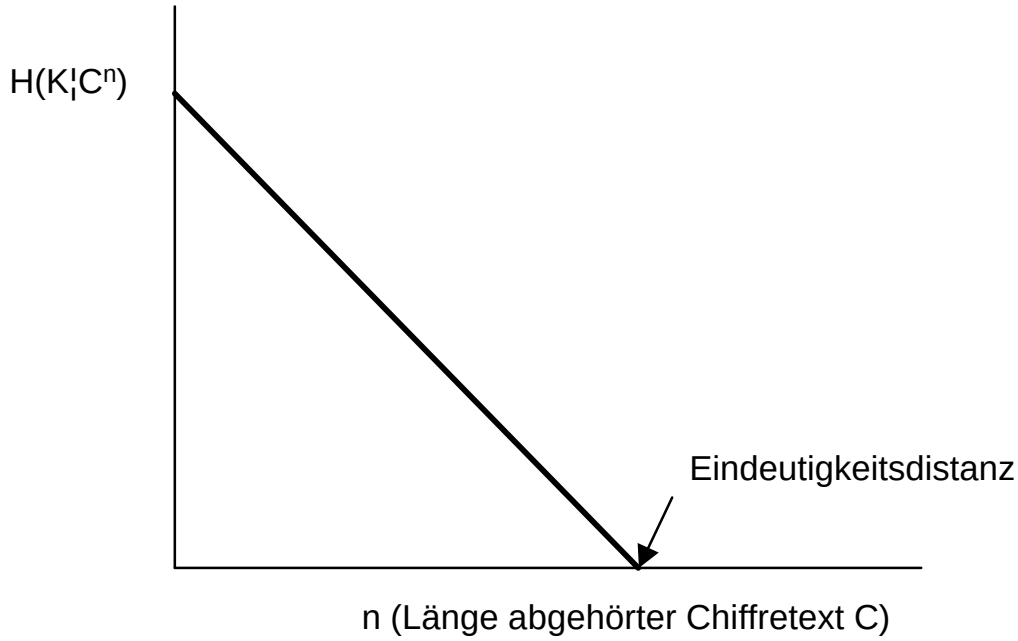
approximiert werden kann. K ist der Schlüsselraum, D die mittlere Redundanz pro Zeichen in der entsprechenden Sprache (D=3.0 - 3.2 für Deutsch und Englisch).

Bedeutung der Eindeutigkeitsdistanz

Chiffre	Schlüsselraum |K|	H(K) [bits]	n_0
Add. Substitution	26	4.7	1.5
affine Substitution	312	8.27	2.6
allg. Substitution	$26! = 4.03 \times 10^{26}$	88.38	27.6
Vigenere	26^d	4.7 d	1.5 d
ENIGMA	$26!^{16900}$	1.49×10^6	0.466×10^6
DES	$2^{56} = 7.2 \times 10^{16}$	56	17.5
RSA	$2^{512} = 1.3 \times 10^{154}$	512	160

Interpretation der Eindeutigkeitsdistanz

Die Eindeutigkeitsdistanz gibt nur an, wieviele Schlüsseltextzeichen mindestens bekannt sein müssen, um die verbleibende Entropie des Schlüsselraumes auf Null zu reduzieren.



Ueber den zur Identifikation notwendigen Aufwand sagt die Eindeutigkeitsdistanz nichts aus.

Man unterscheidet deshalb zwei Zugänge zur Sicherheit eines Chiffre

Informationstheoretische Sicherheit: ***Eindeutigkeitsdistanz zu hoch für Angriff***

Komplexitätstheoretische Sicherheit: ***Berechnungsaufwand zu hoch für Angriff***

Komplexität: Aufwand eines Angriffs

Angriffsformen

- **analytischer Angriff (Klartext)**
- **statistischer Angriff (Schlüsseltext)**
- **Angriff mit extensiver Schlüsselsuche**

Aufwand

- **Lösen von mindestens Rn_0 Gleichungen mit H(K) Variabeln**
- **Komplexe Abschätzungen mit Häufigkeitsanalysen**
- **|K| Ver- oder Entschlüsselungstransformationen**

Aufwand ist von Lösungsalgorithmus und Kryptosystem abhängig

Maximaler Aufwand bei extensiver Schlüsselsuche

Chiffre	Schlüsselraum |K|	Zeit [sec]	
Add. Substitution	26	2.6×10^{-5}	
affine Substitution	312	3.2×10^{-4}	
allg. Substitution	4.03×10^{26}	10^{20}	
ENIGMA	$26!^{16900}$	10^{676000}	(keine Kenntnisse)
DES	$2^{56} = 7.2 \times 10^{16}$	10^{11}	
RSA	$2^{512} = 3.3 \times 10^{154}$	10^{152}	

Für die meisten Kryptosysteme nicht machbar (Ausnahme DES, Teraflopmaschine)

Grundbegriffe der Komplexitätsanalyse

- **Problem** $f: X \to Y$ **Instanzen des Problems** $f: X_n \to Y_n$
- **Aufwand (Komplexität) des Algorithmus *ALG[f(n)]***
  - **Anzahl Berechnungsschritte in Funktion der Instanz f(n)**

$$|ALG[f: X_n \to Y_n]|$$

- **Komplexität der Probleminstanz *f(n)***
  - **Berechnungsschritte für kürzesten Algorithmus**

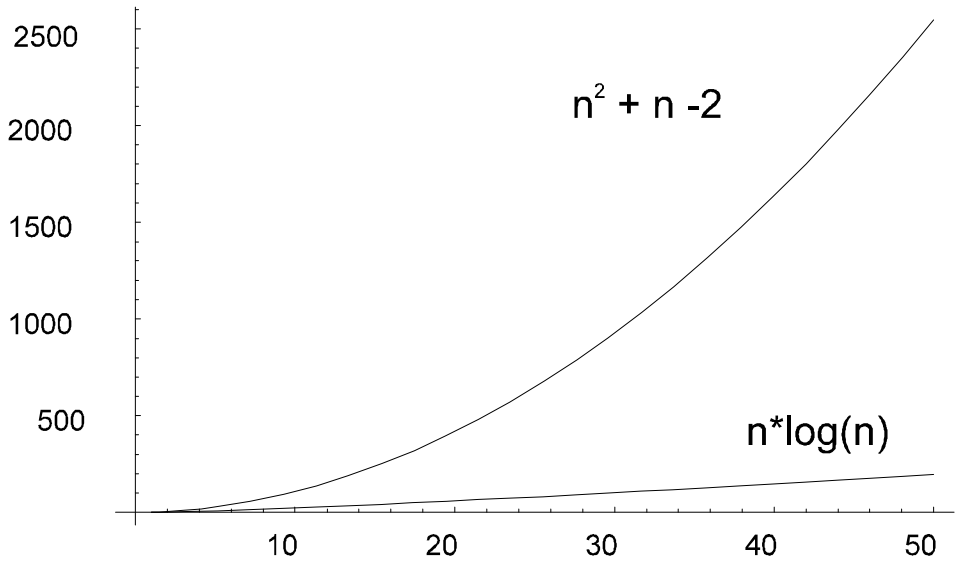
$$\min|ALG[f: X_n \to Y_n]|$$

- **Komplexität des Problems f**
  Skalierung in Funktion von n

Komplexitätsanalyse

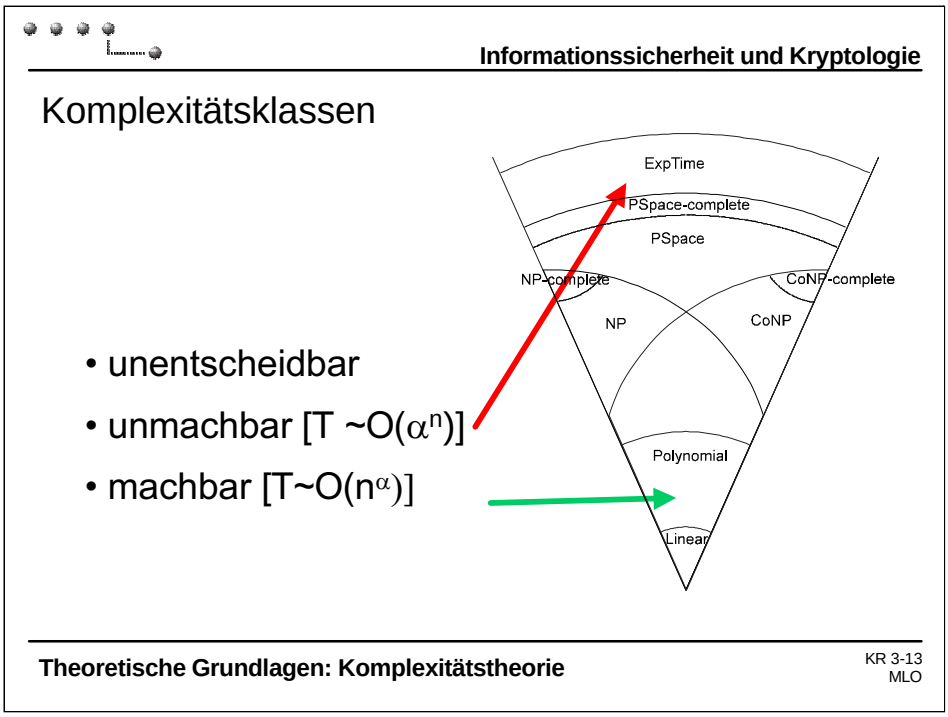
Beispiel: ***Probleminstanz: Ordnen einer Menge mit n-Elementen X(n)***

Einfügealgorithmus vs. Fusionsalgorithmus bzw.Quicksort



Komplexität des Algorithmus: $n^2+n-2 \propto O(n^2)$ $O(n \log n)$

Komplexität des Problems: $O(n \log n)$



Komplexitätsklassen

Der Aufwand für die Lösung eines Problems hängt wesentlich von der Komplexitätsklasse ab.

Man unterscheidet zwischen folgenden Problemkategorien:

Polynomial	***Aufwand***	$T \propto O(n^{\alpha})$
Nichtdeterministisch polynomial (NP)	***Aufwand***	$T_{finden} \propto O(\alpha^{n}); T_{prüfen} \propto O(n^{\alpha})$
NP-vollständig	***Aufwand***	$T_{finden} \propto O(\alpha^{n}); T_{prüfen} \propto O(n^{\alpha})$
Exponentiell	***Aufwand***	$T \propto O(\alpha^{n})$

In der Praxis sind nur Probleme für grössere Instanzen lösbar, die höchstens polynomiale Komplexität haben.

Harte Probleme

Probleme, die nicht zu dieser Klasse gehören, bezeichnet man als **hart**.

Auswirkung der Komplexität

Komplexität klasse	Zeit für Instanz n=10 [sec]	Zeit für Instanz n=100 [sec]	Zeit für Instanz n=1000 [sec]
TSP (brute force) NP-complet	0.02	2.7×10^{37}	2.0×10^{428}
Matrixinversion $n^2*\log(n)$	0.002	0.05	6.9
Sortieren N*log(n)	0.00002	0.0005	0.012

Chiffren auf der Basis von harten Problemen

- **Einwegfunktion**

$$f: X \rightarrow Y$$

- **Berechnung von f(x) hat polynomiale Zeitkomplexität**

$$f \in P \quad \forall x \in X$$

- **Berechnung der Umkehrfunktion ist hart**

$$f^{-1} \in NP \quad \forall y \in Y$$

Einwegfunktionen in der Kryptologie

1976 publizierten W. Diffie und M.E. Hellman die Arbeit *New directions in cryptography*, in der sie die Grundlagen für die public-key Systeme entwickelten.

Grundidee der public key Systeme

Die entscheidende Idee der public-key Systeme ist es, die kryptographische Sicherheit eines Kryptocodes auf der rechnerischen Komplexität von bekannten mathematischen Problemen aufzubauen. Geeignet sind NP (vollständige) Probleme, da das Nachprüfen einer gegebenen Lösung in polynomialer Zeit machbar ist, das Finden einer Lösung jedoch im allgemeinen exponentielle Zeit braucht.

Einwegfunktionen

Verwirklicht wird diese Idee durch sogenannte Einwegfunktionen, deren Vorwärtsberech-nung einfach (höchstens P), deren Umkehrberechnung aber hart (NP) ist.

Beispiele

Diskrete Exponentierung (Potenzierung) ist einfach

$$y = a^x \bmod n \quad \rightarrow \quad T \propto O(\log_2 n)$$

Diskrete Logarithmierung (Wurzelziehen) ist schwierig

$$x = \log_a y \bmod n \quad \rightarrow \quad T \propto O(e^{\sqrt{\ln n \cdot \ln \ln n}})$$

Summieren von ausgewählten Elementen eines Vektors **a**=(a_1, a_2,...a_n) ist einfach

$$y = \mathbf{x} \cdot \mathbf{a} = \sum_{i=1}^{n} x_i a_i \quad \rightarrow \quad T \propto O(n)$$

$$\mathbf{x} \in Z_2^n \text{ so dass } \mathbf{x} \cdot \mathbf{a} = y \quad \rightarrow \quad T \propto O(2^{n/2})$$

Zerlegen einer Summe in Elemente eines vorgegebenen Vektors **a**=(a_1, a_2,...a_n) ist schwierig

Zusatzinformation

- **Einwegfunktion mit Falltüre k (parametrisierte Schar)**

$$f_k : X \to Y$$

- **Berechnung von $f_k(x)$ hat polynomiale Zeitkomplexität**

$$f_k \in P \quad \forall x \in X$$

- **Berechnung der Umkehrfunktion ist hart,**

$$f^{-1} \in NP \quad \text{fast } \forall y \in Y$$

- **aussert man kennt die Zusatzinformation k**

$$f_k^{-1} \in P \quad \forall y \in Y$$

Einwegfunktionen mit Falltüre

Die Zusatzinformation k erlaubt die einfache Berechnung der Umkehrfunktion von *f*. k hat die Funktion des Schlüssels.

$$C = E_k(m) = f_k(m) \quad \therefore \quad D_k(C) = f_k^{-1}(C)$$

Beispiele

Diskrete Potenzierung mit $n=p\,q$ ist einfach

$$y = x^a \bmod n \qquad \to \qquad T \propto O(\log_2 n)$$

Diskretes Wurzelziehen ist schwierig

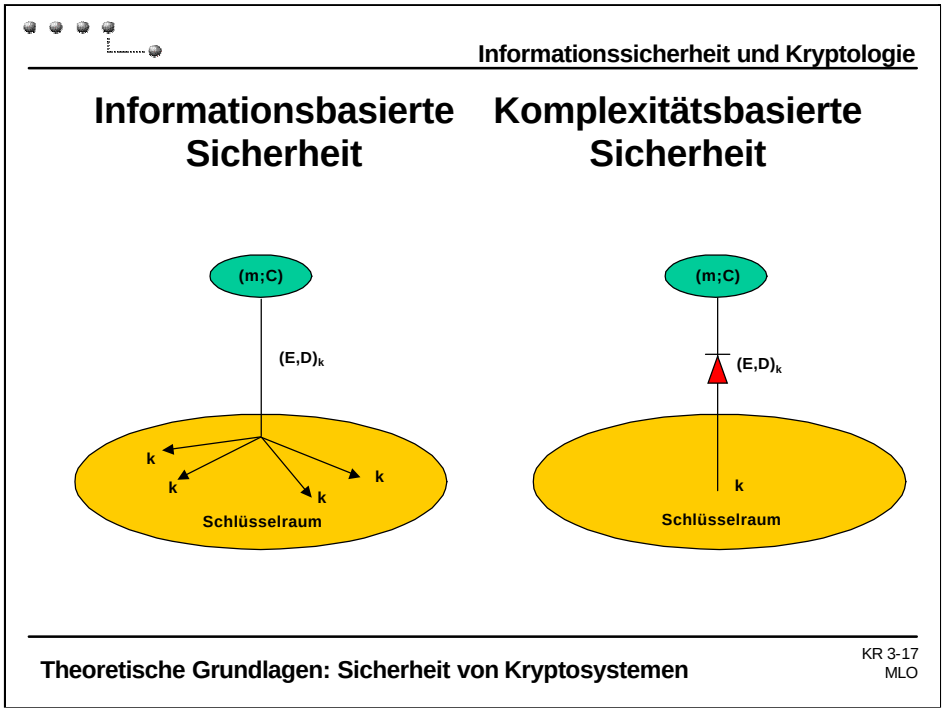
$$x = \sqrt[a]{y} \bmod n \qquad \to \qquad T \propto O(e^{\sqrt{\ln n \cdot \ln\ln n}})$$

aussert man kennt die Faktorisierung von $n=p\,q$

$$x = y^{a^{(p-1)(q-1)-1} \bmod (p-1)(q-1)} \bmod n = x^{a \cdot a^{-1} \bmod (p-1)(q-1)} = x^1$$

$$\to \qquad T \propto O(\log_2 n)$$

(Anwendung des Satzes von Fermat)



Sicherheitsgrad

Die Sicherheit eines Kryptosystems beruht auf:

- **Informationstheoretischer (kombinatorischer)** Sicherheit

  wenn der Angreifer über zuwenig Information verfügt, um das System zu knacken (zu jedem Paar (m,C) existieren noch viele mögliche Schlüssel)

  Diese Sicherheit ist absolut und unabhängig von irgendwelchen neuen Angriffstechniken

  Realisiert wird diese Sicherheit durch einen Schlüsselraum K mit

  $|K| >> |M|$

  und/oder durch eine redundanzfreie Sprache (alle Klar- und Schlüsseltextpaare) gleich wahrscheinlich

- **Kompexitätstheoretischer (mathematischer)** Sicherheit

  wenn der Angreifer über zuwenig Speicher oder Zeit verfügt, um das System zu knacken (Berechnung von k aus (m,C) ist zu aufwendig)

  Diese Sicherheit ist relativ und abhängig von möglichen neuen Angriffstechniken oder besseren Rechenhilfsmitteln

  Realisiert wird diese Sicherheit durch die Konstruktion eines mathematisch schwierig zu beschreibenden Algorithmus (DES, IDEA) oder durch den Einbezug eines bekannten schwierig zu lösenden Problems (NP vollständig)

- Eine Mischform ist die systemtheoretische Sicherheit. Der Angreifer verfügt entweder über zuwenig Information und/oder Rechenkapazität (Kombination DES-Public key)

Umkehrfunktion ohne Schlüssel

- zuwenig Information
  Informationstheoretischer Ansatz
- zu komplexer Algorithmus
  Komplexitätstheoretischer Ansatz
- zu aufwendig und zu teuer
  Systemtheoretischer Ansatz

Sicherung der nicht autorisierten Rücktransformation

Im Prinzip können drei Ansätze unterschieden werden, die je nach Sicherheitsbedarf zur Anwendung kommen:

Höchste militärische und staatliche Sicherheit:	Informationstheorie One time pad
Hohe militärische oder kommerzielle Sicherheit:	Komplexitätstheorie Math. Problem, Schlüssellänge
Geschäftliche Sicherheit:	Systemtheorie Chiffre über Risikoschwelle zB. DES für kleines Risiko