Aufgabe: Berechne das Unizitätsmaß einer affinen Chiffrierung (312 Schlüssel) bzw. einer allgemeinen monoalphabetischen Chiffrierung (26! Schlüssel).

Antwort: affin: $n = \frac{\log_2(312)}{3,2} = 2,6$ (dem Nenner 3,2 liegt die Schätzung 1,5 bit pro Buchstabe zugrunde. Bei $n = 2,6$ stimmt dies nicht. Man sollte eher vielleicht von $n = 4$ als U-Maß ausgehen).

allgemein: $n = \frac{\log_2(26!)}{3,2} = 27,6$

Aufgabe: Eine 1000 Bit lange Nachricht soll unter Verwendung eines modernen bzw. hybriden Kryptoverfahrens übertragen werden. Es steht ein Rechner mit 10^{-9} Rechenoperationen/sec zur Verfügung. Die Schlüssellänge sei 100 Bit.

~~historische Verfahren~~ 1. Methode: Man überträgt den gesamten Nachrichtentext auf „moderne Art" (Laufzeit: m^3, m = Bitlänge). 2. Methode: Man überträgt den Schlüssel zunächst auf moderne Art,

danach überträgt man den eigentlichen Nachrichtentext auf klassische Art mit Hilfe des nun beiden Partnern zur Verfügung stehenden Schlüssels (Rechenzeit der klassischen Methode: m). Vergleiche die Rechenzeiten bei Methode 1 bzw. 2!

Antwort: Methode 1: Rechenzeit =

$(1000)^3 \cdot 10^{-9}$ sec = 1 sec

Methode 2: Rechenzeit $= ((100)^3 + 1000) \cdot 10^{-9}$ sec

$= 0{,}001$ sec

Realistischer als in der Aufgabenstellung angegeben ist folgendes Szenario:
Der Text der Länge 1000 wird in Blöcke à 100 Bits zerlegt. Jeder der Blöcke wird auf moderne Art verschlüsselt, was jeweils

$(100)^3 \cdot 10^{-9}$ sec Rechenzeit

erfordert. Auch diese Vorgehensweise ist langsamer als das hybride Verfahren.

Aufgabe: Hier ist ein anderer Versuch, die Sicherheit eines klassischen Kryptosystems mit einer Maßzahl zu versehen:

Der Kryptoanalytiker darf 2 verschiedene Klartexte x, y einer bestimmten Länge beliebig auswählen. Diese werden sodann verschlüsselt zu x', y' und (in beliebiger Reihenfolge) dem Kryptoanalytiker mitgeteilt. Die kürzeste Klartextlänge für x, y, so dass der Kryptoanalytiker eindeutig in der Lage ist, die richtige Zuordnung (x zu x', y zu y') zu finden, ist ein Maß für die Sicherheit des Kryptosystems.

Welche Maßzahl hat eine monoalphabetische Chiffrierung? eine homophone Chiffrierung? die Playfair-Chiffre? Eine Vigenère-Verschlüsselung mit ~~der~~ einem Schlüsselwort der Länge l?

Antwort: monoalphabetisch: Wähle $x = AA$, $y = AB$, also $n = 2$. homophon: Wähle QQ und AB, $n = 2$.
Playfair: Wähle ABAB und ABCD, $n = 4$.
Vigenère: Wähle $\underbrace{AA\ldots A}_{l+1}$ und $\underbrace{A\ldots AB}_{l}$, also $n = l+1$.

Aufgabe: Dem Kryptoanalytiker sollte zum „Knacken" eines Krypto-Systems kein polynomieller Algorithmus zur Verfügung stehen. Reicht es zu sagen, dass der „Knack"-Algorithmus nicht in $\mathbb{P}$ liegen soll?

Antwort: Dass die Laufzeit $f(n)$ eines Algorithmus nicht polynomial ist, also $f(n) \notin O(n^k)$ (für alle k) bedeutet, nach Definition:

$$\forall k \, \forall c \, \underbrace{\forall n_0 \, \exists n \geq n_0}_{\overset{\infty}{\exists} n} : \quad f(n) > c \cdot n^k$$

Dies bedeutet nur, dass es für unendlich viele n Eingaben der Länge n gibt, bei denen der Algorithmus „sehr lange" ($> cn^k$) rechnen muss.

Erstens: Zwischen diesen unendlich vielen n, für die es Eingaben der Länge n mit langer Laufzeit gibt, können beliebig lange Intervalle liegen mit viel schnellerer Laufzeit.

Zweitens: Innerhalb der Eingaben mit Länge n kann es sein, dass viele davon effizient gelöst werden können.

Also: Die Aussage „$\notin \mathbb{P}$" reicht für kryptographische Zwecke nicht!

Probabilistische Algorithmen kann man in 2 Klassen unterteilen:

- Monte Carlo-Algorithmen: Es können (mit kleiner Wahrscheinlichkeit) falsche Ergebnisse ausgegeben werden:

(einseitiger Fehler)

$x \in A \xrightarrow{\varepsilon}$ ja
$x \notin A \xrightarrow{1-\varepsilon}$ nein

oder

$x \in A \rightarrow$ ja
$x \notin A \rightarrow$ nein

(bei polyn. Laufzeit entspricht dies der Klasse RP)

oder: (zweiseitiger Fehler)

$x \in A \rightarrow$ ja
$x \notin A \rightarrow$ nein

(bei polynom. Laufzeit entspricht dies der Klasse BPP)

$P \subseteq RP \subseteq NP$
$\uparrow \subseteq BPP$ bounded error polynomial-time
random polynomial-time

- Las Vegas-Algorithmen: Ein falsches Ergebnis kann nicht ausgegeben werden, aber evtl: „keine Lösung"

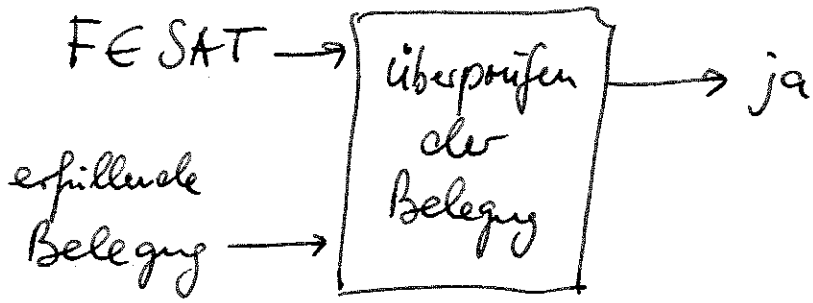
$x \in A \rightarrow$ ja
$\rightarrow$? (keine Lösung)
$x \notin A \rightarrow$ nein

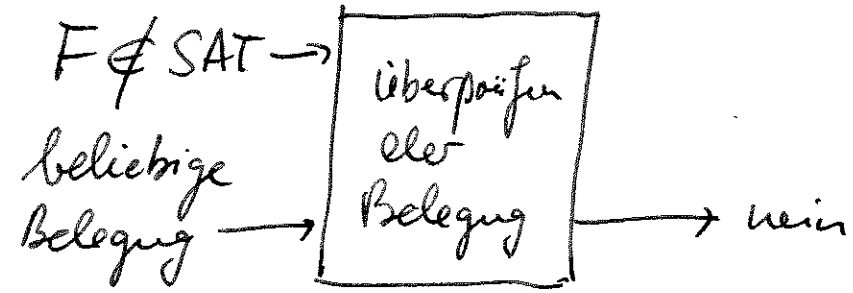
$RP \cap co\text{-}RP$

(bei polyn. Laufzeit ist dies ZPP)

Beispiel für ein NP-Problem: SAT

$$SAT = \{ F \mid F \text{ ist Boolesche Formel und es gibt eine Belegung der Booleschen Variablen } A_1, \ldots, A_n \text{ in } F, \text{ die } F \text{ erfüllt (wahr macht)} \}$$





Die direkte, naive Art ein NP-Problem mit einem deterministischen Algorithmus zu lösen: Bei Eingabe x überprüfe systematisch alle y der maximalen Länge $p(|x|)$, ob M bei Eingabe (x,y) 1 ausgibt. Laufzeit: $2^{p(n)}$ also exponentiell.
In vielen Fällen (insbes. bei den NP-vollst. Problemen) sind keine wesentlich schnelleren Algorithmen bekannt.

Bisher gibt es keinen Beweis, dass für irgendein NP-Problem kein polynomiales Algorithmus existiert, obwohl die meisten Forscher $P \neq NP$ vermuten.

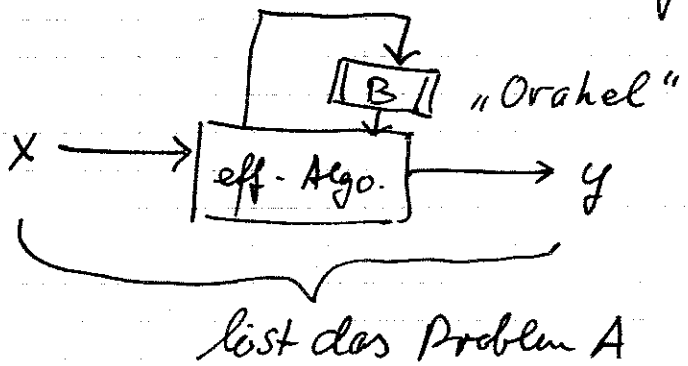
Das P-NP-Problem ist die Frage, ob

$$P = NP \quad \text{oder} \quad P \neq NP$$

Beweise von absoluten unteren Schranken für bestimmte NP-Probleme (welche man dann für kryptographische Zwecke verwenden könnte) erscheinen sehr schwierig zu sein. Was man stattdessen macht, sind relative Vergleiche von Problemen. Hierzu dient folgende Definition:

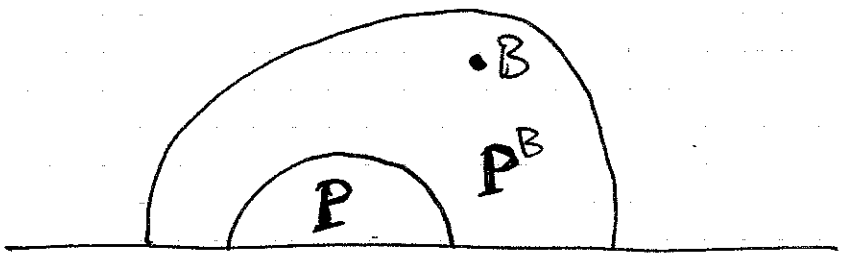
Das algorithmische Problem A lässt sich auf das Problem B effizient reduzieren (Notation: $A \leq_{eff} B$), wenn es einen polynomialen Algorithmus gibt, der das Problem

A löst und dabei ein Unterprogramm verwendet, das das Problem B berechnet. Die Laufzeit zur Berechnung von B wird hierbei nicht berücksichtigt.



Statt $A \leq_{eff} B$ andere Notation: $A \in \mathbf{P}^B$

Skizze:



ansteigende Komplexität

Wichtige Beobachtung:

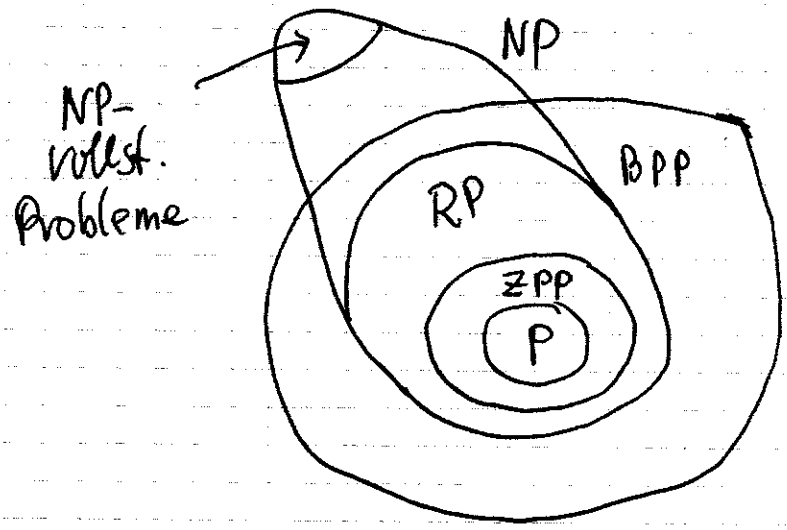
Angenommen,
Es wurde $A \leq_{eff} B$ gezeigt. Ferner sei $B \in \mathbf{P}$

Dann folgt: $A \in \mathbf{P}$.

Gleichwertig, die Kontraposition:

Es gelte $A \leq_{eff} B$. Wenn gezeigt wurde, dass $A \notin \mathbf{P}$, dann folgt auch $B \notin \mathbf{P}$.

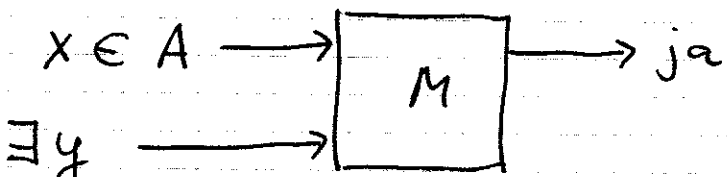
Komplexitätsklassen:



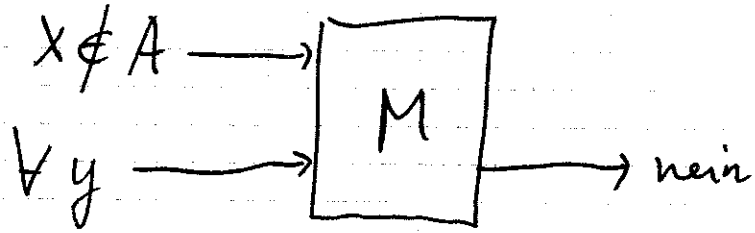
- Algorithmen der Art BPP kann man noch als „effizient" ansehen. Solche Algorithmen sollten dem Kryptoanalytiker nicht zur Verfügung stehen.

Definition der Klasse NP:

Entscheidungsproblem A liegt in NP, falls es einen polynomialen Algorithmus M mit 2 Inputs x, y gibt, so dass Folgendes gilt:



Hierbei gilt:
$|y| \leq poly(|x|)$



Man sagt auch, (das richtige) y ist der Zeuge (witness) dafür, dass x in A liegt.

Zusatz-Notation: Sollte der effiziente Algorithmus für A (in der Def. von $A \leq_{eff} B$) ein probabilistischer sein, so schreiben wir: $A \leq_{prob.eff} B$.

Def: Eine Problemstellung A heißt NP-vollständig, falls gilt:

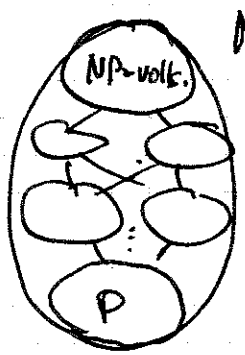
(1) $A \in NP$

(2) für alle Probleme $L \in NP$ gilt:

$$L \leq_{eff} A .$$

Wir schreiben $A \equiv_{eff} B$, falls sowohl $A \leq_{eff} B$ und $B \leq_{eff} A$ gelten.

Die Klasse P bildet eine Äquivalenzklasse von $\equiv_{eff}$, ebenso die NP-vollständigen Probleme (wir wissen nur nicht, ob diese identisch sind):



NP-intermediate-Bereich. Vermutliche Position von GRAPH ISOMORPHIE

Vermutete Situation: $P \neq NP$.

Bekannte NP-vollständige Probleme:

SAT ... Erfüllbarkeitsproblem der Aussagenlogik

HAM ... Hamiltonkreisproblem für Graphen

CLIQUE ... Cliquen-Problem für Graphen

COLOR ... Färbungsproblem für Graphen

TSP ... Travelling Salesman Problem

Der Status eines einzigen NP-vollständigen Problems entscheidet die P-NP-Frage:

> Sei A NP-vollständig. Es gilt
>
> $P = NP$, falls $A \in P$
>
> $P \neq NP$, falls $A \notin P$.

Als „Referenzproblem" für das Knacken eines kryptographischen Systems dient oft:

<u>Faktorisieren</u>: Gegeben n (keine Primzahl)

Finde einen Primfaktor von n.

Das Diskrete Logarithmus - Problem:

Gegeben: n (Primzahl), $a < n$ (Erzeuger der Gruppe $\mathbb{Z}_n^*$ (das heißt $\{1,2,\dots,n-1\} = \{a^1, a^2, \dots, a^{n-1}\} \pmod n$)) und $y < n$

Finde x so dass $y = a^x \pmod n$.

Diese Probleme wurden schon lange untersucht, und die Möglichkeiten für Algorithmen (allerdings nur exponentielle) sind weitgehend erforscht. Daher versucht man zu zeigen:

Das „Knacken" des Krypto-Systems XYZ $\equiv_{\text{eff}}$ Faktorisieren (oder Diskret. Log.)

Das gibt zwar keine absolute Sicherheit, dass das Knacken unmöglich ist, aber eine relative...