Cryptography

Vorlesung Modul E-Government

Leon Aaron Kaplan

E-Mail: aaron@lo-res.org

Nov., Dez. 2004

Inhaltsverzeichnis

Abbildung 1. Voynich Illustrationen

Intro

Kryptologie = Kryptographie + Kryptoanalyse
Kryptographie = Komplexitätstheorie + Zahlentheorie + Informationstheorie + Informatik
Kryptoanalyse = Statistik + Linguistik + Social engineering + noch mehr Informatik + black magic
Kryptographie $\neq$ Steganographie

Unterscheidung:
Symmetrische Verfahren vs. Assymetrische (public-key) Verfahren.

Symmetrische Verfahren: Substitution vs. Transposition, Block Cipher vs. Stream Cipher

Bezeichnungen: Cipher, Code, verschlüsselte Nachricht, Geheimtext, Kryptogram, ...

Der Wert der Information muss immer kleiner sein als der Aufwand, den Code zu knacken.

Problem bei allen symmetrischen Verfahren: Verteilung der Schlüssel.

Idee hinter der Verschlüsselung:

$$C = E_k(P)$$
$$P = D_k(C)$$

Subsitutionsverfahren

In der Praxis am häufigsten verwendet

Monoalphabetisch

Jedes Zeichen wird genau durch ein und nur ein Zeichen im Chiphertext ersetzt.
Maximale Komplexität: $26 \cdot 25 \cdot 24 \ldots 2 \cdot 1 = 26! = 403291461126605635584000000 = 4.03\ 10^{26}$.
Caesar Shift Algorithmus hat nur 25 Möglichkeiten.

Extrem leicht mit Frequency Analysis zu knaken. Falls einfache relative Häufigkeit der Einzelbuchstaben nicht ausreicht, um die Lösung abzulesen, kann man auf Digraphen zurückgreifen.

It will stop your little sister but not even a determinded solver of crossword puzzles.

Polyalphabetisch

Mehrere einfach monoalphabetischen Substitutionen (vgl. Vigenere). Idealerweise für jedes Zeichen in der Nachricht eine eigene monoalphabet. Subsitution.

Transposition

Umordnung des Originaltexts. z.B.: Vertauschung der Spalten. Wiederum mit Frequency Analysis zu knacken.

Geschichte

- Griechenland: Stab + Band
- Caesar: Buchstabe + k
- Kama-Sutra: Vatsyayana - "The Kama-sutra recommends that women should study 64 arts, including cooking, dressing, massage and the preparation of perfumes. The list also includes some less obvious arts, including conjuring, chess, bookbinding and carpentry. Number 45 on the list is mlecchita-vikalpa, the art of secret writing, advocated in order to help women conceal the details of their liaisons. One of the recommended techniques involves randomly pairing letters of the alphabet, and then substituting each letter in the original message with its partner." [Sin99] -
- 13. Jhd. englischer Mönch Roger Bacon: "Concerning the Marvelous Power of Art and of Nature and Concerning the Nullity of Magic". Listet 7 Chiffrierungen auf. "A man is crazy who writes a secret in any other way than one which will conceal it from the vulgar". Es wird vermutet, daß Roger Bacon Autor der berühmten ungelösten Voynich Chiffrierungen (vgl. Abb. 1, [dRB00]) ist.

- Um 1460: Leon Batista Alberti (Florenz) - Erfindung der polyalphabetischen Subsitution.
- 1586: Blaise de Vigenère 600 Seiten ‘‘Traicté des chiffres‘‘. Aufzählung vieler Systeme u.a. "Vigenère tableau" Methode (vgl. Abbildung 2). Vorläufer der Vernam Verschlüsselung.
- 1587: Mary, Queen of Scotts enthauptet. Ihr Code wurde von Thomas Phelippes, Sekretär von Queen Elizabeth I geknackt. War noch nicht einmal polyalphabetisch. Monoalphabetisches Substitutionsverfahren.
- Auguste Kerckhoffs von Niewenhof (1835-1903): "Kerckhoffs' Principle": The security of a cipher should not depend on an enemy's ignorance of the enciphering algorithm, only the enemy's ignorance of the key
- 1917: Vernam: theoretisch untermauerte sichere Verschlüsselung. ‘‘One-time-pad‘‘. Der Schlüssel muss 1) komplett zufällig 2) genauso lang wie die Nachricht sein. Anschliessend wird modulo 26 addiert. Sei n die Länge der Nachricht, dann kann man - ohne Wissen um den Schlüssel - jede beliebige Nachricht aus dem Chiffriertext erzeugen. Darin beruht die Sicherheit. Achtung: key darf nie ein 2tes Mal ver-

wendet werden! Allerdings schwierig in der Handhabung. GRU und russ. Botschaft haben oft Vernam verwendet. Allerdings wurden Schlüssel wiederverwendet und dadurch geknackt.

- 1928-1945: Enigma (Vgl. Abb 3). Theoretisch an die 10^{134} Möglichkeiten zu verschlüsseln. Zuerst von den Polen vor dem Krieg geknackt. Kurz vor Besetzung an die Franzosen und Engländer weitergegeben. In England in Bletchley Park 50km nördl. von London unter Mitarbeit von Alan Turing im Projekt ULTRA und Colossus regelmäßig geknackt. 1942 wurde Enigma verbessert und ein vierter Rotor kam hinzu (Naval-Enigma). Einige Zeit versenkte die deutsche U-Boot Flotte sehr erfolgreich.

> ‘‘The captured codebooks gave the British a significant advantage in the Battle of the Atlantic until 1 February 1942, when the Germans changed the cipher scheme again, introducing a new Enigma that featured a special fourth rotor. The fourth rotor could be set so that the new Enigma acted as a three-rotor machine, allowing U-boats to receive new Enigma machines sequentially, not going to four-rotor operation until the entire

> fleet was up to standard. However, in December a U-boat radio operator had made a blunder, sending a message using four-rotor operation. When he realized his mistake, he then retransmitted the same message in three-rotor mode. The British caught it and figured out the wiring of the fourth rotor.‘‘

US bauen 'Bombe' - Auftragnehmer National Cash Register (NCR) - um die Enigma Ciphers zu knacken. Massive Automatisierung und Parallelisierung (120 'Bomben' waren im Einsatz, Kostenpunkt je Stück: 6 mill US Dollar, vgl. Abb 4)

- Seit WWII: Immer Ausgefallenere symmetr. Verschlüsselungen. Momentan gilt z.B.: 128bit AES als sicher. 1976: FIPS DES (data encryption standard). Lange wurde vermutet, dass ein die NSA mitlesen konnte, da sie einer der S-BOXEN designed hat. Mittlerweile - nach langer Analyse - scheint DES doch gut designed zu sein. NSA verwendet - Vermutung! - einen Hardware DES knacker.
- 1974: Fred Winterbotham, senior field officer, British SIS, publiziert seine Memoiren ‘‘THE ULTRA SECRET‘‘. Noch lebende deutsche führende Entwickler der Enigma und Kryptographen sind fassungslos.

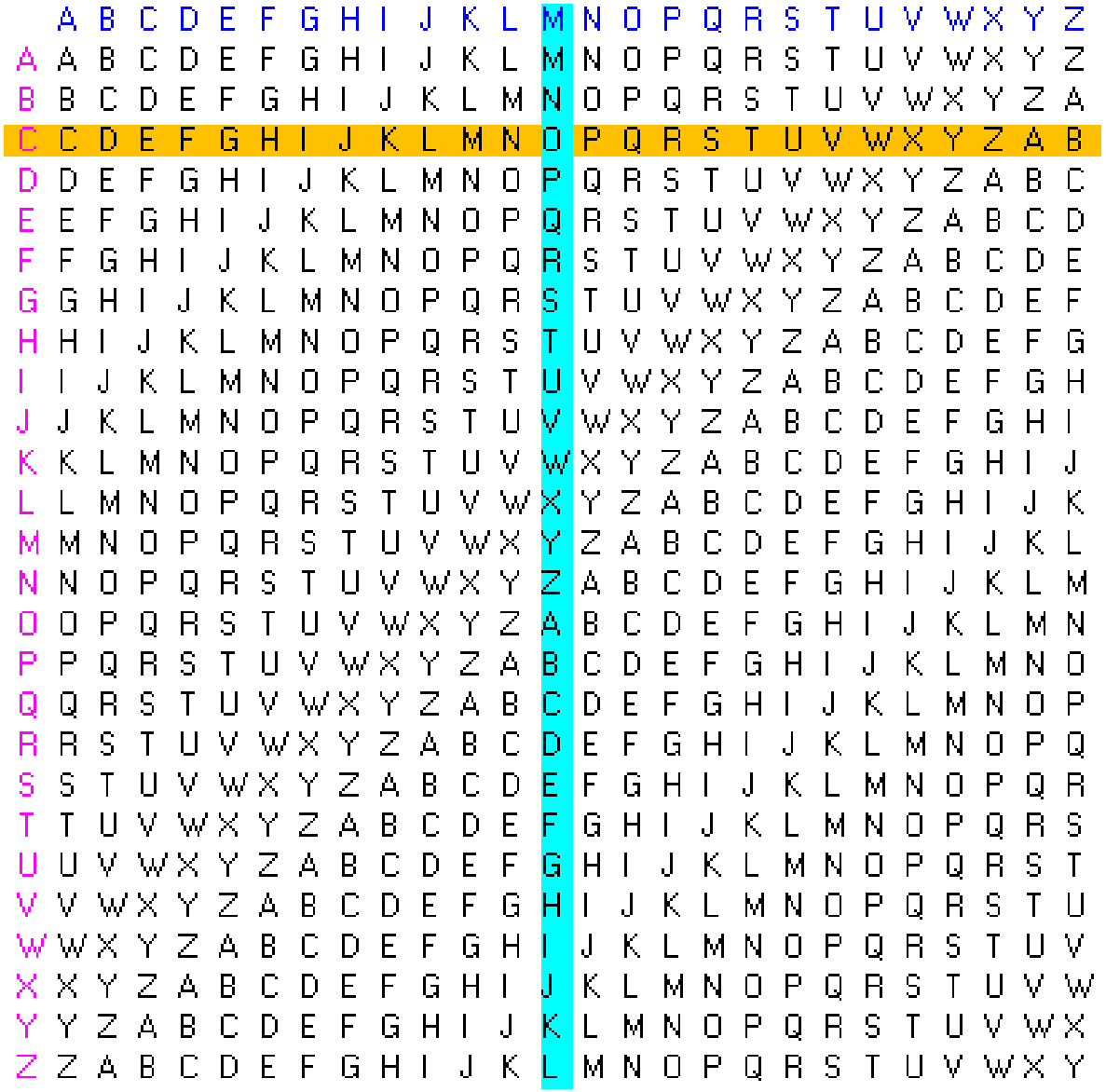
> 'Karl Doenitz was thunderstruck to find out that the British had been reading the "unbreakable" Enigma cipher through most of the war.'

Die Russen wussten schon lange von ULTRA aufgrund von Spionage.

- Seit den 1970ern: Dank des Durchbruchs der public-key crypto ergeben sich massig neue Einsatzgebiete der Crypto wie z.B.: electronic voting, e-cash, digitale Signaturen, faire Protokolle, etc. Anzahl der Publikationen explodiert. GCHQ und NSA stellen 1990 ''Anspruch'' auf die Entdeckung der public key crypto. Allerdings haben sie ihre Erkenntnisse nie publiziert.

- 1980er: Charles Bennett (IBM), Gilles Brassard greifft eine Idee von Stephen Weissner auf: Quantencryptography wird geboren.

- 1991: Phil Zimmerman veröffentlicht PGP - pretty good privacy - frei und gratis im Internet. Investigationen seitens des FBI etc. sind die Folge. Rechtstreit mit RSA Data Systems Inc. über Patentverletzung. Das RSA Patent ist 1997 ausgelaufen.



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

Abbildung 2. Vigenère Tableau

Abbildung 3. Enigma

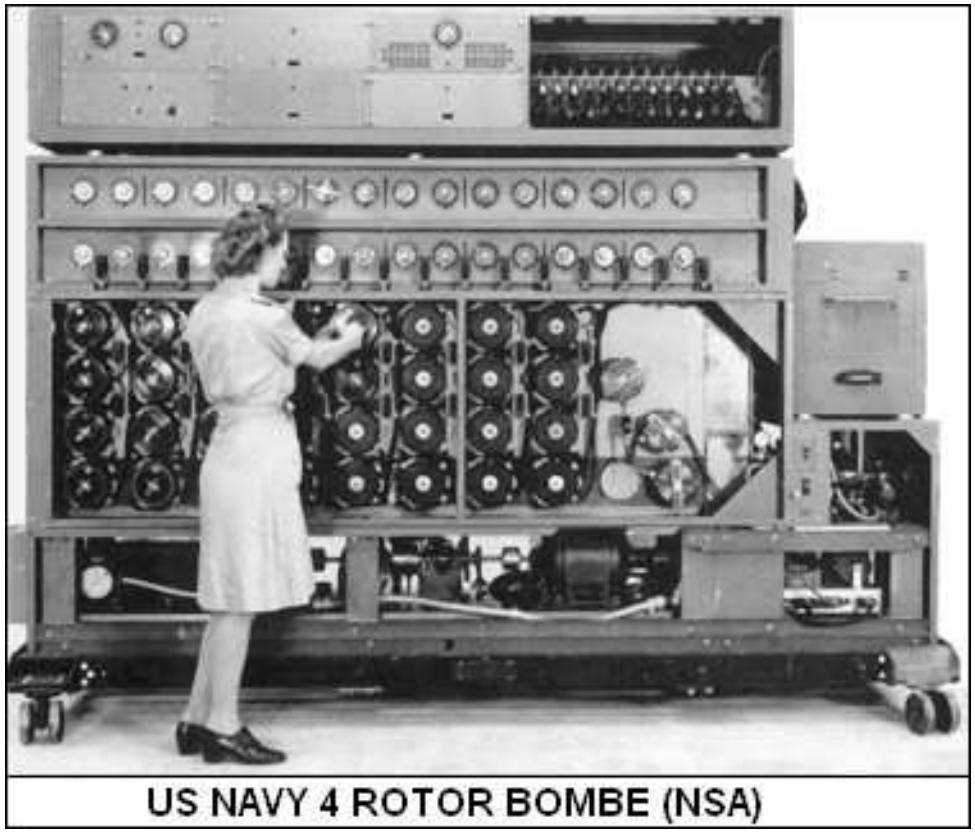


Abbildung 4. US ‘‘Bombe‘‘ – Enigma cracker

Komplexitätstheorie

"How fast can we do it if the problem size grows"

Unterscheidung: Time complexity (T) vs. Space complexity (S)
Komplexität lässt sich mittels einem Maß festlegen:

O-Notation von Landau-Bachman (1892)

Es seinen f und g Funktionen $\mathbb{N} \to \mathbb{N}$. Wir sagen
$f \in O(g) \Leftrightarrow \exists N, k \in \mathbb{N}$, sodass $\forall_{n>N} |f(n)| \leqslant |g(n)| \cdot k$

"f wächst **höchstens** so stark wie g"

Beispiel 1. Quicksort$(n) \in O(n \log(n))$

Als Berechnungsmodell dient immer die Turingmaschine, die nur Strings verarbeiten kann $\to$ Es zählt die Bitlänge genauso und nicht nur die Anzahl der Operationen.

Beispiel 2. Sieb des Erastosthenes. Der Aufwand (doppelte for schleife) schaut quadratisch aus. Aber in Wirklichkeit müssen wie die Bitlänge von MAX berücksichtigen. Auf der Turingmaschine haben wir keine WORDs sondern nur Strings!

```
unsigned char primes[MAX];
primes[0] = FALSE;
primes[1] = FALSE;
primes[2] = TRUE;
for (i=3; i < MAX; i++) primes[i] = TRUE;

for (i=2; i < MAX; i++) {           // optimierung: bis sqrt(MAX)
  for(j=i+1; j < MAX; j++) {
                                      // ``wegstreichen'' falls
    if (primes[j] && 0 == j%i)       // noch nicht weggestrichen
      primes[j] = FALSE;                // optimierung: bitfeld
  }
}
```

Aufwand: $O(n^2 \log(n))$ wobei $\log(n)$ die Bitlänge ist.

Rechnen in der O-Notation

Wenn f ein Polynom ist, so kann man alles außer die höchste Potenz ignorieren. Beweis anhand von

Beispiel 3. $f(x) = 6x^4 - 2x^3 + 5$
Wir behaupten, $f \in O(x^4)$.
Beweis: wir müssen ein N, $k \in \mathbb{N}$ finden, sodass $\forall_{x>N} \left|f(x)\right| \leqslant k \cdot \left|x^4\right|$. Wir nehmen an $N > 1$. Es gilt:

$$\left|6x^4 - 2x^3 + 5\right| \leqslant \left|6x^4 + 2x^3 + 5\right|$$
$$\left|6x^4 - 2x^3 + 5\right| \leqslant 6x^4 + 2x^4 + 5x^4 \qquad x^3 < x^4 \text{ und } 5 < 5x^4$$
$$\left|6x^4 - 2x^3 + 5\right| \leqslant 13x^4$$
$$\left|6x^4 - 2x^3 + 5\right| \leqslant 13\left|x^4\right| \qquad \text{Wobei } k := 13, N := 1 \quad \blacksquare$$

Es zählt also nur die höchste Potenz. Konstanten oder konstante multiplikative Faktoren sind egal.

Bemerkung 4. In der Praxis sind konstanten **nicht** zu vernachlässigen! Beispiel: $O(x^{17} + 10^{10^{1000}}x^{16} + \ldots) \in O(x^{17})$ obwohl die Konstante sehr groß ist.

Beispiel 5. Geburtstagsparty, jeder muss mit jedem anstoßen, wie oft klirrt es? Bei n Gästen muss jeder mit $n-1$ anderen anstoßen. Also $\frac{n(n-1)}{2} = \frac{n^2-n}{2}$ mal. Wir planen eine große Party mit 1000 Gästen. Um eine Abschätzung zu erhalten, rechnen wir $O(n^2)$ aus: $1000^2 = 10^6$.

Komplexitatsvergleiche bei n=10^6

Klasse	Komplexität	# Operationen	Zeit bei 10^6 Ops / sec
Konstant	$O(1)$	1	1 μsec
Linear	$O(n)$	10^6	1 sec
Quadratisch	$O(n^2)$	10^{12}	11,6 Tage
Kubisch	$O(n^3)$	10^{18}	32000 Jahre
Superpolynomiell	$O(c^{f(n)})$	bei $c=2$, $f(x)=\sqrt{x}$: $10.715 \cdot 10^{300}$	$3397.73 \cdot 10^{290}$ Jahre
Exponentiell	$O(c^n)$	für $c=2$: 10^{301030}	10^{301006} mal die Lebensdauer des uns bekannten Universums
Faktoriell	$O(n!)$	XX	XX

Komplexitätsklassen

P	(deterministisch) polynomiell
NP	nondeterministic polynomiell: die Maschine errät die richtige Lösung und verifiziert sie in polynomieller Zeit
NP-complete	NP-complete Probleme sind mindestens so hart wie alle anderen NP Probleme. Beweismethode: Reduktion [COOK]
PSPACE	kann mit polynom. Speicherplatz gelöst werden aber nicht unbedingt in P time
PSPACE-complete	s.o.
EXPTIME	definitiv nur in exponentieller Zeit zu lösen. Es ist bewiesen: $P \neq$ EXPTIME!

$\mathrm{NP} \neq^{?} P$.
Kann NP, PSPACE mit P oder NP zusammenfallen? Wir wissen es nicht! Falls gilt: $\mathrm{NP} = P$, dann wird ein Großteil der bekannten Crypto hinfällig! Im Moment gilt die Annahme, dass $\mathrm{NP} \neq P$, weil alles darauf hindeutet.

Beispiel: traveling salesman (TSP) ϵ NP, 3-SAT ϵ NP

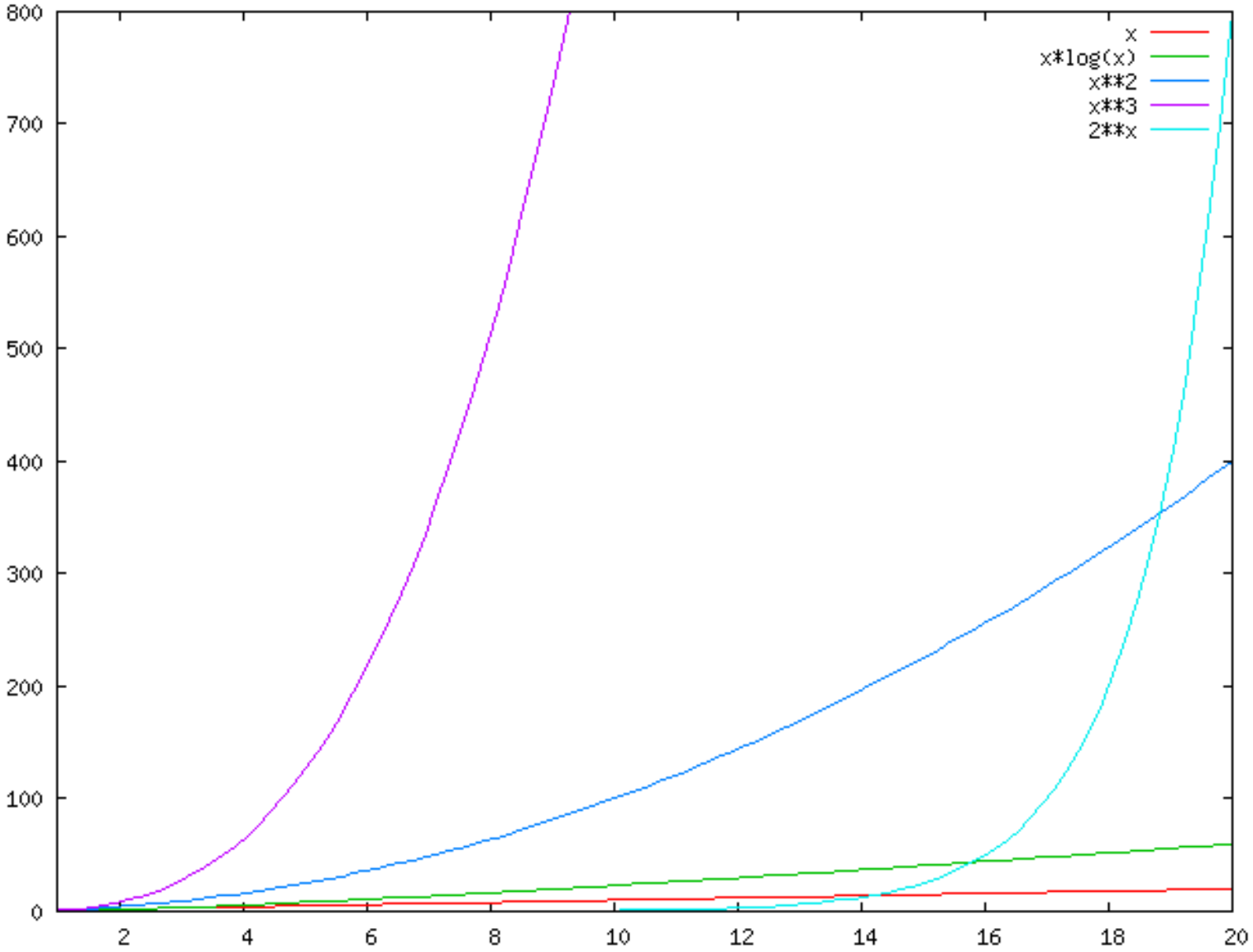


Abbildung 5. Anstieg der Komplexitätsklassen

Bedeutung für Crypto

Idealerweise wollen wir Verschlüsselung, die nur mit sehr hohem Aufwand zu knacken ist (‘‘es muss unrentabel werden‘‘). z.B.: Aufwand knacken exponentiell, Verschlüsseln $O(n)$. In der Praxis kann man lediglich maximal auf Aussagen der Form: ‘‘alle bekannten Knackverfahren sind zumindest superpolynomiell‘‘ kommen.
Quantencomputer sind NP knackende Maschinen.

Zahlentheorie

> Mathematics is the queen of sciences and number theory is the queen of mathematics. *Gauß*

Lange Zeit ‘‘nutzlos‘‘ in der Mathematik, aber dank Crypto mittlerweile sehr weit verbreitet.

Mit großen Zahlen rechnen

Wir sind gewohnt, im 10er System zu rechnen. Es hält und aber nichts davon ab, andere Zahlensysteme zu verwenden. z.B.: zur Basis $b = 2^{32}$:

$x = \sum_{i=0}^{N} a_i b^i$, wobei für jede Ziffer a_i gilt: $0 < a_i < b$.

Beispiel 6. $128_d = \mathbf{8} \cdot 16^1 + \mathbf{0} \cdot 16^0 = 80_{16}$

Beispiel 7. In der RSAREF library (RSA Systems) wird zur Basis $b = 2^{32}$ gerechnet. Die Multiplikation ist dabei - selber Algorithmus wie in der Schule - mit Aufwand $O(n^2)$ (wobei hier n die Anzahl der 32 Bit Blöcke bezeichnen soll). Fortgeschrittenere Algorithmen verwenden FFT zur Multiplikation (Aufwand: $O(n \log(n)^2)$). Vgl: `http://numbers.computation.free.fr/Constants/Algorithms/fft.html`

Grundlagen und Gruppentheorie

Definition 8. *Gruppe*
Sei G eine Menge und $\bullet$ eine binäre Operation, sodass $a \bullet b \in G$ wohldefiniert für alle $a, b \in G$. Dann heißt G eine Gruppe, wenn gilt:

$\forall a, b \in G$:	$a \bullet b \in G$	*für alle a,b in G (Abgeschlossenheit)*
$\forall a, b, c \in G$:	$(a \bullet b) \bullet c = a \bullet (b \bullet c)$	*Assoziativität*
$\exists! e \in G \, \forall a \in G$:	$a \bullet e = e \bullet a = a$	*Einselement e*
$\forall a \in G \; \exists! a^{-1} \in G$:	$a \bullet a^{-1} = a^{-1} \bullet a = e$	*Inverses Element*

Falls weiters gilt

$$\forall a, b \in G: \quad a \bullet b = b \bullet a$$

dann heißt die Gruppe abelsch (oder kommutativ).

Beispiel 9. $(\mathbb{R}, +)$ bildet eine Gruppe.

Beispiel 10. $(\mathbb{C} \to \mathbb{C}, \cdot)$, die Menge der Funktionen von $\mathbb{C}$ auf $\mathbb{C}$ bilden eine abelsche Gruppe, wobei unter $\cdot$ die punktweise Multiplikation zu verstehen ist.

Aufgabe 1. Beweise!

Aufgabe 2. Beweise, dass es nur ein Einselement geben kann!

Definition 11. *Ring*
Sei R eine abelsche Gruppe bezüglich $+$ und $\bullet$ eine binäre Operation, sodass $a \bullet b \in R$ für alle $a, b \in R$ wohldefiniert ist, dann heißt R Ring, wenn gilt:

1. $\forall a, b, c \in R$: $(a \bullet b) \bullet c = a \bullet (b \bullet c)$ — *Assoziativität bzgl $\bullet$*
2. $\forall a, b, c \in R$: $a \bullet (b + c) = a \bullet b + a \bullet c$ — *Distributivgesetz links*
3. $\forall a, b, c \in R$: $(a + b) \bullet c = a \bullet c + b \bullet c$ — *Distributivgesetz rechts*

Gibt es überdies ein Einselement 1, sodass gilt $\forall a \in R$: $1 \bullet a = a \bullet 1 = a$, dann spricht man von einem Ring mit Einselement.

Beispiel 12. $(\mathbb{Z}, +, \cdot)$ ist ein Ring.

Aufgabe 3. Beweise!

Definition 13. *Äquivalenzrelation*
Sei R eine binäre Relation auf eine Menge M, dann heißt R Äquivalenzrelation, wenn $\forall a, b, c \in M$ gilt:

1. aRa — *Reflexivität*

2. $aRb \Leftrightarrow bRa$ *Symmetrie*

3. $aRb \wedge bRc \Rightarrow aRc$ *Transitivität*

Aufgabe 4. Finde einige!

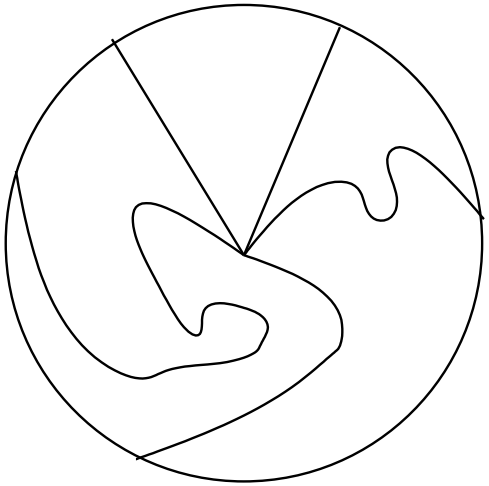


Abbildung 6. Aufteilung der Menge M in disjunkte Äquivalenzklassen

Man beachte, dass eine Äquivalenzrelation ein Aufteilen der Gesamtmenge in disjunkte Teilmengen bewirkt: jedes Element x einer Teilmenge ist äquivalent mit allen anderen Elementen in der selben Teilmenge (Vgl. Abbildung 6).

Primzahlen und Teilbarkeit

Definition 14. *Teilbarkeit*
Seien $a, b \in \mathbb{Z}$, wir sagen a teilt b (in Zeichen: $a|b$) genau dann wenn $\exists q \in \mathbb{Z}$, sodass $b = a\,q$.

Aufgabe 5. Welche Zahlen teilen lt. Definition die Zahl 0? Welche Zahl(en) sind durch 0 teilbar?

Aufgabe 6. Welche Zahlen sind durch 1 teilbar? Durch welche Zahlen ist 1 teilbar?
Ist $a \in \mathbb{Z}$ immer durch a und $-a$ teilbar?

Satz 15. *Seien $a, b, c \in \mathbb{Z}$ und $a|b+c$ und $a|b$, dann gilt: $a|c$*

Beweis. Laut Voraussetzungen $a|b \wedge a|b+c \Leftrightarrow \exists x, y \in \mathbb{Z}: a\,x = b \wedge a\,y = b+c$. Daraus ergibt sich $a\,y - b = c \Leftrightarrow a\,y - a\,x = c \Leftrightarrow a\,(y-x) = c \Leftrightarrow$ (weil $(y-x) \in \mathbb{Z}$) $a|c$. □

Definition 16. *gemeinsamer Teiler.*
Seien $a, b \in \mathbb{Z}$, jede Zahl $c \in \mathbb{Z}$, für die gilt: $c|a$ und $c|b$ heisst gemeinsamer Teiler von a und b.

Beispiel 17. $-1, 1$ teilen jede Zahl (inlusive 0), $2|6$ und $2|18462$

Definition 18. *ggT*
a, $b \epsilon \mathbb{Z}$, $a \neq 0$, $b \neq 0$. Mit ggT(a,b) (englisch: gcd(a,b)) bezeichnen wir den größten gemeinsamen Teiler von a und b. Wenn ggT(a,b) = 1 so heissen a und b relativ prim.

Beispiel 19. $a = 5, b = -20$. Teiler von a sind $1, -1, 5, -5$. Teiler von b sind $1, -1, 2, -2, 4, -4, 5, -5, 10, -10, 20, -20$. ggT(a,b) = 5.
Es gilt $\text{ggT}(a,b) = |a| \Leftrightarrow a|b. \Rightarrow \text{ggT}(a,0) = |a| \quad a \neq 0$

Satz 20. (Euklid) *(Beweisformulierung entnommen aus [HS90])*
Euklid *(365-300 v.Chr.) entwickelte ein Verfahren, um den ggT algorithmisch zu ermitteln. Das Verfahren kommt ohne Primfaktorzerlegung aus.*
Seien $a, b \epsilon \mathbb{N}, b \leqslant a$, dann gibt es ganze Zahlen $q_i, r_i \epsilon \mathbb{Z}, n \epsilon \mathbb{N}$, sodass $r_n = \text{ggT}(a, b)$:

$$a = q_0 r_0 + r_1, \quad 0 \leqslant r_1 < b = r_0 \tag{1}$$

$$r_0 = q_1 r_1 + r_2, \quad 0 \leqslant r_2 < r_1 \tag{2}$$

$$r_1 = q_2 r_2 + r_3, \quad 0 \leqslant r_3 < r_2 \tag{3}$$

$$\vdots$$

$$r_i = q_{i+1} r_{i+1} + r_{i+2}, \quad 0 \leqslant r_{i+2} < r_{i+1} \tag{4}$$

$$\vdots$$

$$r_{n-1} = q_n r_n + r_{n+1} \quad 0 \leqslant r_{n+1} < r_n \tag{5}$$

Es gibt ein kleinstes $n \geqslant 0$ *sodass* $r_{n+1} = 0$. *Es ist dann* $\mathrm{ggT}(a,b) = r_n$

Beweis. Wegen $0 \leqslant r_{i+1} < r_i$ gibt es ein solches n.
Wir zeigen zuerst, dass $r_n | a$ und $r_n | b$ durch Induktion: da $r_{n-1} = q_n r_n$, ist $r_n | r_{n-1}$. Angenommen (Induktionshypothese) wir haben schon gezeigt, dass $r_n | r_j$ für alle $i < j \leqslant n$ .Da wegen (4) $r_i = q_{i+1} r_{i+1} + r_{i+2}$ und weil $i < j$, gilt die Induktionshyptothese, also: $r_n | r_{i+1}$ und $r_n | r_{i+2} \Rightarrow r_j | r_i$. Folglich $r_n | r_o, r_1$ woraus folgt $r_n | a = q_0 r_0 + r_1$ und $r_n | b = r_0$.
Wir müssen noch zeigen, dass r_n der größte gemeinsame Teiler von a und b ist: es reicht also zu zeigen, dass jeder Teiler t von a, b kleiner gleich r_n ist.
Es sei t ein gemeinsamer Teiler von a und b. $t | a, b$. Dann ist $t | a - q_0 b = r_1$.
Wieder arbeiten wir mit Induktion. Es sei gezeigt, dass $t | r_j$ für $j \leqslant i$.
$t | r_{i+1} = r_{i-1} - q_i r_i$. Also teilt t auch $r_n \Rightarrow t \leqslant r_n$ □

Definition 21. *(lineares Kompositum)*

$a, b, g, x, y \epsilon \mathbb{Z}$. Wenn es für a, b ganze Zahlen x, y gibt, sodass gilt $g = a\,x + b\,y$, dann nennen wir g lineares Kompositum.

Satz 22. *Es seinen g_1, g_2 lineare Komposita von a und b, und es seien c_1, c_2 ganze Zahlen, dann ist auch $c_1 g_1 + c_2 g_2$ lineares Kompositum von a und b.*

Beweis. Übungsaufgabe □

Satz 23. *Sei a, $b \epsilon \mathbb{Z}$ und nicht beide gleich 0. Es sei $d = \mathrm{ggT}(a, b)$. Dann existieren ganze Zahlen x, y, sodass $a\,x + b\,y = d$.*

Beweis. oBdA können wir annehmen, dass $b \leqslant a$. Es sei zunächst $b > 0$. Wir verwenden den euklidischen Algorithmus. Es sei $a = q_0 b + r_1, 0 \leqslant r_1 < b = r_0$, $r_{i-1} = q_i r_i + r_{i+1}, 0 \leqslant r_{i+1} < r_i$ und $r_{n-1} = q_n r_n, r_n \neq 0, r_{n+1} = 0$. Dann ist nach dem euklid. Algorithmus $\mathrm{ggT}(a, b) = d = r_n$. Da $r_1 = a - q_0 b$, ist r_1 lineares Kompositum von a und b. Wir gehen weiter per Induktion vor: Es sei bereits bewiesen, dass für $j \leqslant i$, r_j lineares Kompositum von a und b sei. Nach Satz 22 ist dann aber auch $r_{i+1} = r_{i-1} - q_i r_i$ ein solches. Insbesondere ist $r_n = d$ ein lineares Kompositum von a und b.

Wenn $b = 0$, dann ist naturgemäß $\text{ggT}(a, 0) = a$ und trivialerweise $d = 1 \cdot a + 0 \cdot b$.

Wenn $b < 0$, dann gibt es ganze Zahlen x, y mit $d = a\,x + (-b)\,y = a\,x + b(-y)$. □

Bemerkung 24. Dieser Satz gibt erstens an, wie wir (mit dem euklid. Algorithmus) x und y finden können und zweitens wird sich später herausstellen, dass wir diese Eigenschaft für das Inverse im Restklassenring brauchen werden.

Satz 25. *Es gibt unendlich viele Primzahlen*

Beweis. (indirekt, Euklid) Es seinen p_1, ..., p_n alle (endlich vielen) Primzahlen. Wir nehmen nun eine Primzahl p heraus und betrachten, ob $p | p_1 \cdot \ldots \cdot p_n + 1$. Es muss ein i geben, sodass $p_i = p$ (weil wir ja alle Primzahlen ‘‘verwendet‘‘ haben). Dann teilt (laut Satz 15) $p | 1$. Widerspruch! Also kann es nicht endlich viele Primzahlen geben. □

Wir **wissen** somit, dass uns die Primzahlen für die Kryptographie nie ausgehen werden.

Ein paar weitere interessante Fakten über Primzahlen (aus [HS90]):

- Es ist nicht bekannt, ob es unendliche viele Primzahlen der Form p_n, $p_n + 2$ gibt (Primzahlzwillinge)
- Andererseits gibt beliebig große Abstände zwischen 2 Primzahlen:
  Sei N eine natürliche Zahl, wir behaupten, dass zwischen $(N+1)!$ $+2, \ldots (N+1)! + N + 1$ keine einzige Primzahl liegt.

  Beweis. Jede der Zahlen $(N+1)!+2, \ldots (N+1)!+N+1$ ist durch eine der Zahlen 2, 3, ..., $N+1$ teilbar. Sei also p_n die größte Primzahl $\leqslant (N+1)!+1$, so ist $p_{n+1} \geqslant (N+1)!+N+2 \Rightarrow p_{n+1} - p_n \geqslant N+1$ $\square$

- Es gibt keine brauchbare, schnelle(!) Formel für die Berechnung aller großen Primzahlen.
  Erstaunlich jedoch ist, dass es ein Polynom in 26 Variablen gibt, dessen Nullstellen $\cap \mathbb{N}$ alle Primzahlen ergeben. In irgendeiner Reihenfolge. Die Nullstellenberechnung ist nicht trivial! (vgl. XXX SATO, ...)
- Für die Berechnung aller Primzahlen $\leqslant 10^9$ ist ein ca. 2150 Jahre Algorithmus noch immer der schnellste: Sieb des Erastosthenes

Man streiche alle Vielfachen aller Primzahlen $p_n \leqslant \sqrt{N}$. Die Zahlen die übrig bleiben sind Primzahlen.

- Die Mersenne'schen Zahlen $2^n - 1$, $n \geqslant 2$ generieren uns die größten bekannten Primzahlen. Ob es unendlich viele p gibt, sodaß M_p prim ist, ist unbekannt. Wenn n nicht prim ist, so ist auch M_n nicht prim. Im Mai 2004 war die größte bekannte mersenne'sche Primzahl $2^{24036583} - 1$ (vgl. the primes page: `http://www.utm.edu/research/primes/`)
- $\pi(n)$ bezeichner die Anzahl der Primzahlen zwischen 2 und n. Es gilt $\lim_{n\to\infty} \pi(n) \frac{\log(x)}{x} = 1$ (Gauss). Mittlerweile ist bekannt, dass die $\mathrm{Li}(x)$ Funktion mit

$$\mathrm{Li}(x) = \int_2^x \frac{\partial u}{\log u}$$

  eine viel genauere Abshätzung ist (vgl. Abb 7).
- Agrawal, Kayal and Saxena haben 2002 herausgefunden und bewiesen, daß Primzahltests in P liegen. Wir können also immer mit einem deterministischen polynomiellen Algorithmus herausfinden, ob eine Zahl eine Primzahl ist. [AMN02]

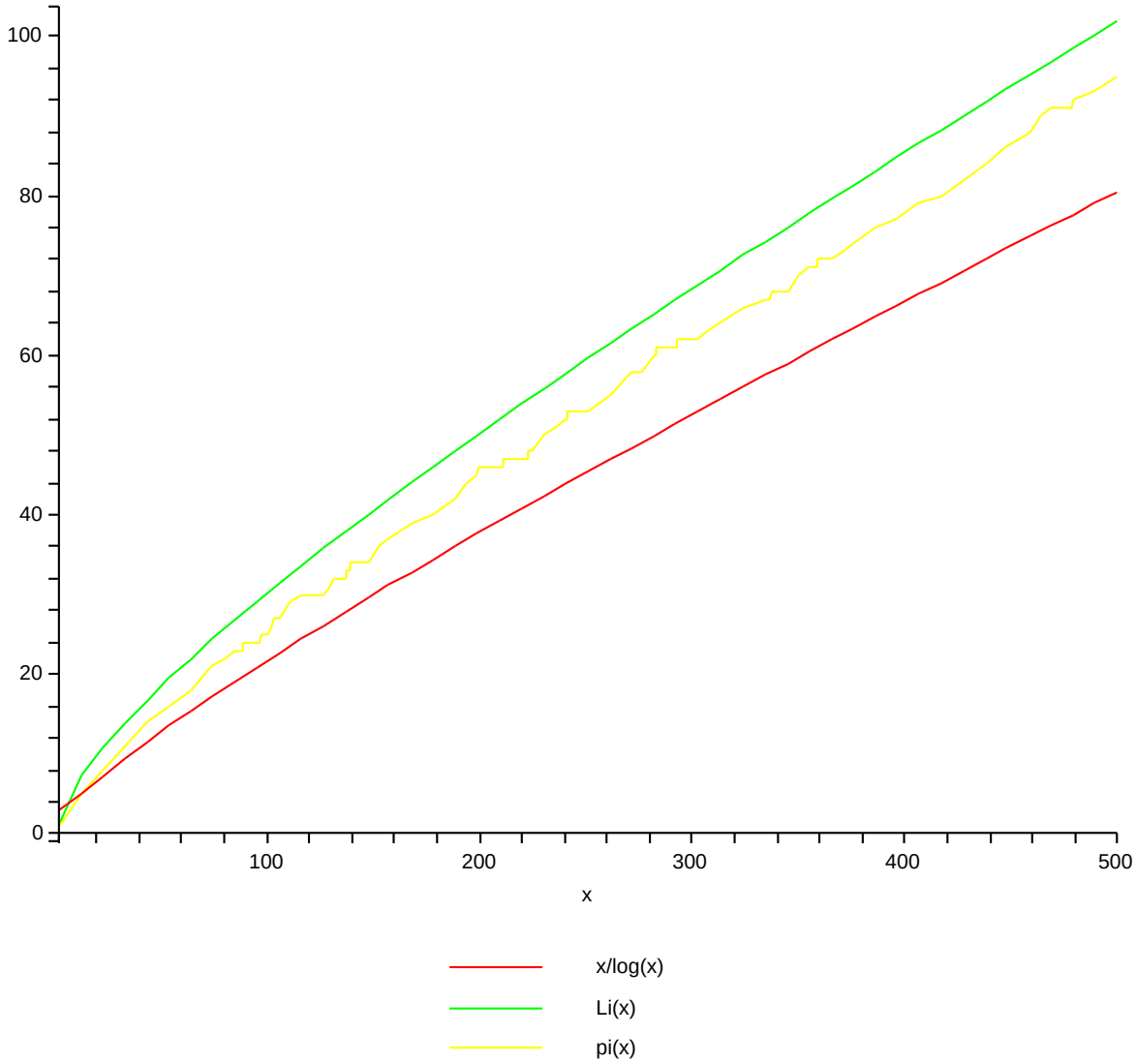


Abbildung 7. $\pi(x), \mathrm{Li}(x)$ und $x/\log(x)$ im Vergleich

Kongruenzen

Einfache Vorstellung: ‘‘der Rest bei der Division‘‘

Definition 26. *kongruent modulo m*
Sei $a, b \epsilon \mathbb{Z}, m \epsilon \mathbb{N}$. a *is kongruent* b *modulo* m *- in Zeichen:* $a \equiv b \pmod{m}$ *- genau dann, wenn* $m \mid a-b$.

Beispiel 27. $17 \equiv 15 \pmod{2}$, $33 \equiv 3 \pmod{10}$, $173 \equiv 28 \pmod{29}$

Satz 28. *Es seien* $a, b \epsilon \mathbb{Z}$ *und* $\exists q_1, q_2, r_1, r_2 \epsilon \mathbb{Z}$, *mit* $0 < r_1, r_2 < m$, *sodass* $a = m\,q_1 + r_1$ *und* $b = m\,q_2 + r_2$, *dann gilt* $a \equiv b \pmod{m} \Leftrightarrow r_1 = r_2$

Beweis. $a - b = m\,(q_1 - q_2) + r_1 - r_2$. $m \mid a-b \Leftrightarrow m \mid r_1 - r_2$. Wegen $|r_1 - r_2| < m \Rightarrow r_1 = r_2$ □

Damit wäre die Behauptung, dass modulo nichts anderes ist als der ‘‘Rest bei der Division‘‘, bewiesen.

Satz 29. *Die Relation* $\equiv \pmod{m}$ *ist eine Äquivalenzrelation*

Beweis.

1. Reflexivität: $a \equiv a \pmod{m}$, da $m \mid 0$ für alle m

2. Symmetrie: $a \equiv b \pmod{m} \Leftrightarrow m \mid a-b \Leftrightarrow \exists q \in \mathbb{Z}: m\,q = a-b$. Es sei nun $\bar{q} = (-1)\,q$. Dann ist $\bar{q}\,m = (-1)(a-b) = (b-a) \Leftrightarrow b \equiv a \pmod{m}$
3. Transitivität: $a \equiv b \pmod{m}$ und $b \equiv c \pmod{m} \Leftrightarrow m \mid a-b$ und $m \mid b-c \Leftrightarrow \exists q_1, q_2 \in \mathbb{Z}: m\,q_1 = a-b$ und $m\,q_2 = b-c \Rightarrow m\,(q_1+q_2) = a-c \Leftrightarrow m \mid a-c \Leftrightarrow a \equiv c \pmod{m}$

□

Rechenregeln bei Modulo

Satz 30. *Rechenregeln für* $\equiv \pmod{m}$

1. $a \equiv b \pmod{m} \wedge c \in \mathbb{Z} \Rightarrow a+c \equiv b+c \pmod{m}$
2. $a \equiv b \pmod{m} \wedge c \equiv d \pmod{m} \Rightarrow a+c \equiv b+d \pmod{m}$
3. $a \equiv b \pmod{m} \wedge c \equiv d \pmod{m} \Rightarrow a\,c \equiv b\,d \pmod{m}$

Beweis.

1. $a+c \equiv b+c \pmod{m} \Leftrightarrow m \mid (a+c)-(b+c) = a+c-b-c = a-b \Leftrightarrow m \mid a-b \Leftrightarrow a \equiv b \pmod{m}$
2. $m \mid a-b \wedge m \mid c-d \Rightarrow \exists q_1, q_2 \in \mathbb{Z}: m\,q_1 = a-b \wedge m\,q_2 = c-d$. $m(q_1+q_2) = a-b+(c-d) = (a+c)-(b+d) \Rightarrow m \mid (a+c)-(b+d) \Leftrightarrow a+c \equiv b+d \pmod{m}$

3. $m|a-b \wedge m|c-d \Rightarrow m|(a-b)\,c+b\,(c-d)=a\,c-b\,d$ □

Bemerkung 31. Man beachte, dass Satz 30 (2) eine Verallgemeinerung von Satz 30 (1) ist. Weiters aufgrund von Satz 30 (3) gilt auch im speziellen $a \equiv b \pmod{m} \Rightarrow a^k \equiv b^k \pmod{m}$.

$\mathbb{Z}_m$

Wir wollen nun zeigen, dass die Modulo Relation auf dem Zahlenbereich der ganzen Zahlen einen Ring bildet. Dazu fangen wir zuerst an zu zeigen, dass wir es mit einer Gruppe bezüglich $+$ zu tun haben:

1. $a, b \epsilon \mathbb{Z}: a + b (\mathrm{mod}\, m) \epsilon \mathbb{Z}_m$
2. $a, b, c \epsilon \mathbb{Z}: (a + b) + c = a + (b + c) (\mathrm{mod}\, m)$
3. $0 + a = a + 0 (\mathrm{mod}\, m)$
4. $a + (-a) = (-a) + a (\mathrm{mod}\, m)$
5. $a + b = b + a$

Bzgl. $\cdot$ gilt:

1. $a, b \epsilon \mathbb{Z}: a \cdot b (\mathrm{mod}\, m) \epsilon \mathbb{Z}_m$
2. $a \cdot (b \cdot c) = (a \cdot b) \cdot c (\mathrm{mod}\, m)$
3. $a \cdot 1 = 1 \cdot a = a$
4. i.A. gilt NICHT: $a \cdot a^{-1} = a^{-1} \cdot a = 1$!!! Gegenbeispiel: $\mathbb{Z}_4$: $2 \cdot 2 = 0 (\mathrm{mod}\, 4)$, $2 \cdot 1 = 1 (4)$, $2 \cdot 3 = 2 (4)$. Hier hat 2 kein inverses Element.

5. Kommutativität ist trivialerweise gegeben.

Definition 32. *Sei m eine natürliche Zahl, dann bezeichnen wir mit $\mathbb{Z}_m$ den Restklassenring modulo m, wobei wir jedes Element $a \epsilon \mathbb{Z}_m$ mit einem beliebigen Vertreter der Klasse aller $x \epsilon \mathbb{Z}$, für den gilt $a \equiv x \pmod{m}$, assoziieren können. Wir bezeichnen auch dieses a als $\bar{a}$.*

Bemerkung 33. Wir können somit mit den Vertretern der Restklassen wie mit normalen Zahlen Rechnen, ausser bei der Divison. Es stellt sich somit die Frage, ob und wann man in $\mathbb{Z}_m$ dividieren kann.

Welche $x \epsilon \mathbb{Z}_m$ erfüllen $a\,x \equiv x\,a \equiv b \pmod{m}$?

Satz 34. *Es seien a und b ganze Zahlen, m eine natürliche Zahl. $a\,x \equiv b \pmod{m}$ ist genau dann lösbar, wenn $\mathrm{ggT}(a,m) | b$. Es gibt dann genau $\mathrm{ggT}(a,m)$ inkongruente Lösungen modulo m.*

Beweis. $\mathrm{ax} \equiv b \pmod{m} \Leftrightarrow \mathrm{my} = \mathrm{ax} - b \Leftrightarrow b = \mathrm{ax} - \mathrm{my}$. Wenn es also so ein $y \epsilon \mathbb{Z}$ gibt, dann können wir die Gleichung lösen. Das ist aber nach Satz 23 genau dann der Fall, wenn $\mathrm{ggT}(a,m) | b$. □

Bemerkung 35. Wir haben somit ein wichtiges Resultat. Varianten von dem Satz sagen $a\,x \equiv 1 \pmod{m} \Leftrightarrow a$ und m sind relativ prim.

$\mathbb{Z}_p$

Nach der letzten Bemerkung wird klar, dass wenn $m = p$ eine Primzahl ist, dann hat jede Zahl $a \epsilon \mathbb{Z}_p$ ein multiplikatives Inverses. $\mathbb{Z}_p$ bildet somit einen Körper.
Wieviele a gibt es in $\mathbb{Z}_p$, die zu p relativ prim sind?

Definition 36. *Eulersche φ-Funktion*
Sei m eine natürliche Zahl, dann ist $\varphi(m)$ die Anzahl der Zahlen $< m$, die zu m relativ prim sind.

Satz 37. *Sei p eine Primzahl. Dann ist $\varphi(p) = p - 1$*

Beweis. Übungsaufgabe □

Satz 38. *(Ohne Beweis)*
Sei n und m teilerfremd. Dann ist $\varphi(m\,n) = \varphi(n)\,\varphi(m)$

Satz 39. *Kleiner Fermat (Ohne Beweis)*
Wenn $\mathrm{ggT}(a, n) = 1$, dann ist $a^{\varphi(n)} \equiv 1 \pmod{n}$

Bemerkung 40. Folglich ist $a^{-1} = a^{\varphi(n)-1} \pmod{n}$

Bemerkung 41. Wie berechnet man $a^x \pmod n$ wenn x sehr groß ist?
Mit folgendem Trick: $((((a^2 \bmod n)\, a) \bmod n)^2 \bmod n)^2 \bmod n$. So bleiben die Zahlen immer schön klein.

Public key crypto Systeme

Bahnbrechende Idee von Diffie, Hellman und Merkle: Aufteilen des Schlüssels in privaten Schlüssel (s) und public key (p).

Diffie Hellman (1976)

Streng genommen kein public-key encryption Algorithmus sondern ein public-key key exchange Algorithmus. Allerdings war DH der erste Algorithmus, der die public-key Idee aufzeigen konnte. Weiters ist er einer der einfachsten.

Seien g, p große Primzahlen, g Primitivwurzel modulo p (= es existiert ein k, sodass $g^k = 1 (\mathrm{mod}\, p)$, g erzeugt quasi die ganze Gruppe $\mathbb{Z}_p$). g und p können

durchaus öffentlich bekannt sein.

→ Alice schickt an Bob:

$X = g^x \pmod p$

→ Bob an Alice:

$Y = g^y \pmod p$

→ Alice berechnet jetzt

$k_1 = Y^x \pmod p$

→ …und Bob

$k_2 = X^y \pmod p$

Behauptung: $k_1 = k_2$. Beweis: $k_1 = Y^x = (g^y)^x = g^{yx} = g^{xy} = (g^x)^y = X^y = k_2$

Alice und Bob haben sich demnach auf einen gemeinsamen key k geeinigt,

ohne dass einer der beiden den geheimen key x (respektive y) des anderen wissen musste. Alleine mit diesem key können sie noch nicht verschlüsseln! Es braucht noch z.B. ein symmetrisches Verfahren, um geheime Daten auszutauschen.

Was muss man tun, um dieses Verfahren zu knacken?

1. Ein Angreifer muss entweder x oder y erraten oder berechnen. Angenommen der Angreifer hat X, Y, k_1, g, p. Dann kann er mittels $\log(k_1)/\log(Y) = x \pmod p$ oder mittels $\log(X)/\log(g) = x \pmod p$ berechnen. Allerdings ist der Logarithmus modulo p schwer zu berechnen. schnellste bekannte Algorithmus dazu benötigt $O(e^{(1.923+O(1))(\ln(p)^{1/3}(\ln(\ln(p)))^{2/3}})$ Schritte (number field sieve - vgl. [AL91],[Bru96, Seite 262] und Abbildung 8).

2. Einfacher: der Angreifer gibt vor, Bob zu sein (‘‘man in the middle attack‘‘). Alice und Bob können sich gegen diesen Angriff nur mit Signaturen wehren.

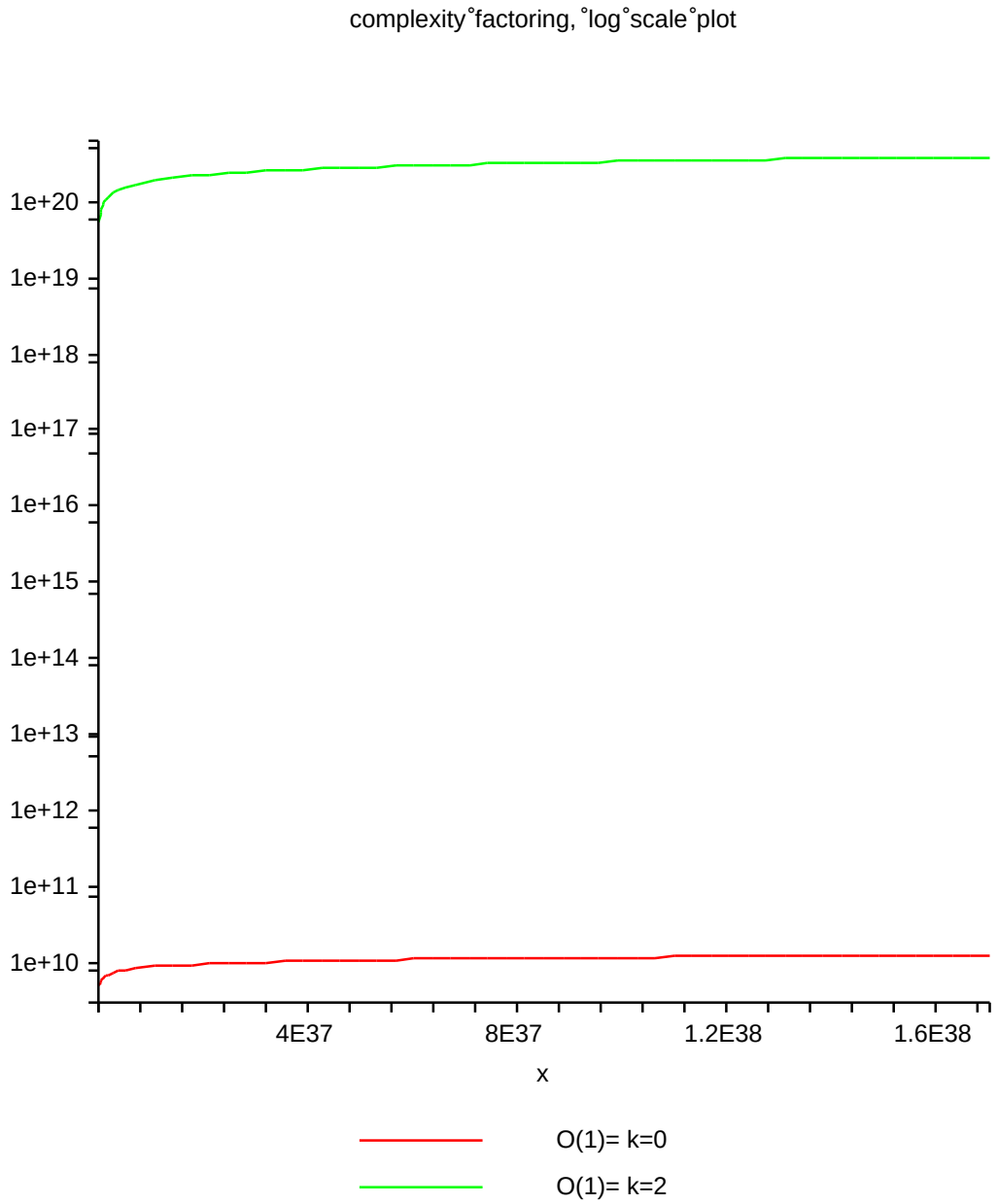


Abbildung 8. Komplexität der Faktorisierung in GF(p) mittels number field sieve. Achtung: logscale Darstellung!

RSA

Benannt nach den Erfindern Rivest, Shamir, Aldeman [RSA78]. Bis heute wohl der meist verwendete Algorithmus. Im Gegensatz zu DH kann man mit RSA direkt verschlüsseln und digital signieren. Der Rechenaufwand für RSA ist aber nicht zu unterschätzen. Deswegen wird meist eine hybride Variante verwendet, wobei ein session key (konventioneller symmetrischer key) mittels RSA verschlüsselt (und evtl. signiert) wird. Der Empfänger - und nur der Empfänger - kann den session key rekonstruieren und somit den eigentlichen Datenstrom, der mit dem session key verschlüsselt ist, entschlüsseln.

Funktionsweise

Wir wählen zuerst einen public key (n_i, e_i) und einen private key d_i für jeden Teilnehmer des Verfahrens:
Alice erzeugt: $n_{\text{Alice}} = p\ q$, wobei p, q große Primzahlen sind und ungefähr gleich lang sein sollen. Es gibt noch eine Reihe weiterer Tests, ob die gewählten Primzahlen ‘‘safe‘‘ sind. Dann wählt Alice zufällig(!) ein e_{Alice} sodass gilt $\text{ggT}(e_{\text{Alice}}, (p-1)\,(q-1)) = 1$ dadurch wird sichergestellt, dass e_{Alice} ein multiplikatives Inverses hat. Dies berechnen wir nun:

$$e_{\text{Alice}}\, d_{\text{Alice}} \equiv 1 \,(\text{mod}\,(p-1)\,(q-1))$$

Anschliessend werden p, q verworfen.

Jetzt verschlüsseln wir an Alice:
Bob schickt (nachdem er Alice's public key erhalten hat) die Nachricht m, indem er sie in mehrere Blöcke aufteilt: m_i. Dabei verwendet er Alice's public key, um zu verschlüsseln:

$$c_i = m_i^{e_{\text{Alice}}} \pmod{n}$$

Schauen, wir was passiert, wenn Alice die Nachricht c_i erhält:

$$m_i = c_i^{d_{\text{Alice}}} \pmod{n} = (m_i^{e_{\text{Alice}}})^{d_{\text{Alice}}} = m_i^{e_{\text{Alice}} d_{\text{Alice}}} = m_i^{k(p-1)(q-1)+1} = m_i \, m_i^{k(p-1)(q-1)} = m_i \cdot 1^k = m_i \cdot 1 \pmod{n}$$

Bemerkung 42. Wir haben hier implizit den kleinen Fermat verwendet. Wo hat er sich versteckt? Erinnerung und Hint: $\varphi(p\,q) = (p-1)\,(q-1)$, da p, q Primzahlen sind.

Analyse

Angenommen Mallory will den Code knacken, er hat n, e_{Alice}, c_i. Um m_i oder d_{Alice} zu berechnen, muss er folgende Gleichungen lösen können:

$$(c_i - x^{e_{\text{Alice}}}) = k\,(p\,q) \text{ oder: } n = p\,q \text{ und } d \equiv e^{-1} \pmod{(p-1)\,(q-1)}$$

In beiden Fällen muss er Faktorisieren können. Und das ist schwer. Im ersten Fall muss er sogar den Logarithmus im Restklassenring ausrechnen können. Das ist äquivalent schwer.
Es gibt aber eine weitere Attacke, die sich bei Signaturen auswirkt, wenn Alice blind eine unverständliche Nachricht mit Signatur quittiert (z.B: Auto-reply per mail mit Signatur). Schließlich funktioniert wie meistens der man-in-the-middle und stellt eine ernste Gefahr dar. Lässt sich mit Web-of-trust oder PKI lösen.

In der Praxis

Beispiel 43. GnuPG

Die OpenSource Implementierung von PGP.

Beispiel 44. OpenSSH Version 2 mit RSA

```
# ssh-keygen -t rsa
# Generating public/private rsa key pair.
Enter file in which to save the key (/Users/aaron/.ssh/id_rsa):rsatest
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in rsatest. Your public key has been saved in rsatest.pub.
The key fingerprint is: 9b:76:a1:2f:b3:f5:69:dd:0f:81:16:18:f5:05:3c:f1
# more rsatest
-----BEGIN RSA PRIVATE KEY-----
MIICWgIBAAKBgQDfrx4Z3TQR9DcAd8LhVhNd/C3iBteQs3ne2BP0Fo8biEZqFhwp
YU/tdKg3Y2ahpaUDdFwvpkWwMuhbtdSVAYamJqlKQrhfmvuuVnBjTknL5EIAe/1f
...
/obaPuUy6nIbE0cCQQC/MpRvOs3XXuHO2/jCXoxw/b2IHs4u9UMZS72J89KYSmmY
8HkyFBajQBP/D6bgqLPq0FPpPmRZyUtL8F0iL3N5
-----END RSA PRIVATE KEY-----
```

```
# more rsatest.pub ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAIEA368eGd00EfQ3AHfC4VYTXfwt4gbXkLN53tgT9BaPG4hG
ahYcKWFP7XSoN2NmoaWlA3RcL6ZFsDLoW7XUlQGGpiapSkK4X5r7rlZwY05Jy+RCAHv9X1G+RmubOtCT
FcLHglT1V8S12WJrQznTGF1yQYeks5IKdxOcSQi1FMkEfis= aaron@instabil.local
# ssh-keygen -l
Enter file in which the key is (/Users/aaron/.ssh/id_rsa): rsatest
1024 9b:76:a1:2f:b3:f5:69:dd:0f:81:16:18:f5:05:3c:f1 rsatest.pub
# ssh-keygen -B
Enter file in which the key is (/Users/aaron/.ssh/id_rsa): rsatest
1024 xilih-bycyf-tuhaf-duhyr-ryhaf-sogus-cucec-zelyv-nasan-bomyk-coxix rsatest.pub
```

Digitale Signaturen

Nachdem im RSA Verfahren gilt $x^{ed} = x^{de}$ und e public, aber d private ist, können wir folgenden Trick verwenden:

Alice schickt an Bob $x^{d_{\text{Alice}}e_{\text{Bob}}} \pmod{n_{\text{Bob}}}$. Damit kann Bob $x^{d_{\text{Alice}}}$ entschlüsseln. Wenn er nun $x^{d_{\text{Alice}}e_{\text{Alice}}} \pmod{n_{\text{Alice}}}$ berechnet, dann ergibt sich: $x^1 \pmod{n_{\text{Alice}}}$. Da nur Alice d_{Alice} kennt, kann die Nachricht nur von Alice gekommen sein.

Um nicht die ganze Nachricht mit RSA verschlüsseln zu müssen, wird oft folgendes Hybride Verfahren angewendet: Es wird ein (symmetrischer) session key erzeugt. Der wird mit RSA verschlüsselt und vor die (mit dem symmetr. session key verschlüsselte Nachricht) gehängt. Genauso gehen wir bei digitalen Signaturen vor: wir erzeugen einen fingerprint (MAC, message digest) der Nachricht und verschlüsseln diese mit dem private key. Dann wird dieser fingerprint zusammen mit der Nachricht (verschlüsselt oder nicht, egal) , verschickt. Bob kann somit genauso den fingerprint der Nachricht berechnen und somit verifizieren, dass Alice und nur Alice wirklich diese Nachricht geschickt hat.

Zeitstempel

Signatur über einen Zeitstempel. Signatur stammt von einer trusted third party (Notar, Trent). Problem dabei: der Notar kann noch die Nachricht lesen, wenn wir sie nicht extra verschlüsseln.
Nächste Version des Protokolls:

1. Alice $\rightarrow$ $h = \text{Hash}(M)$ an Trent
2. Trent generiert $h2 = \text{Hash(h,time(now()))}$
3. er signiert: $s = D_T(h2)$
4. Trent $\rightarrow s$ an Alice

blindfolded signatures

Trick: $(m_i \cdot B)^{d_{\text{Carla}}}$. Carla unterschreibt etwas blind, was mit der Zahl B versteckt wurde. Sie kennt B i.A. nicht. Das macht Sinn, wenn 1) Zeitstempel hinzugegeben werden und 2) Carla dafür bekannt ist, daß sie nie für den Inhalt einer Nachricht unterschreibt, sondern nur mit ihrer Unterschrift den Erhalt der Nachricht beglaubigt $\Rightarrow$ Notar-Funktion.

Aufgabe 7. Wie kann das für e-voting verwendet werden?

Zufallszahlen

Wir brauchen gute Zufallszahlen für RSA etc.
Gute Quellen:

1. natürliche Quellen: radioaktiver Zerfall, kosmisches Hintergrundrauschen, etc.
2. Disk seek time
3. Mausbewegung… eher schlecht
4. aktuelle Scanline des CRT
5. ankunftszeit von datenpaketen
6. /dev/audio ohne Micro

Diese so gewonnen Werten sollten noch mehrmals durch einen kryptographischen Hash geschickt werden.
Vgl. EGD in Linux / Unix

One way Hash functions

aka: Compression function, **message digest**, **fingerprint,** cryptographische Checksumme, ...

n-to-1 function. Mehrere Elemente der Grundmenge sollen auf ein Element im Wertebereich abgebildet werden, so dass man nicht mehr nachvollziehen kann, welches das Ursprungsbild war.
Wir fordern:

1. Kollisionsfreiheit: es soll schwierig sein, ein selbes Element in der Grundmenge zu finden, das den selben Hash hat.
2. Einweg Funktion: Output hängt in keiner erkennbaren Form vom Input ab . Eine Änderung von einem Bit im Input ändert im Durchschnitt 50% der Bits im Output.

Beispiele: MD5, MD4, SHA,
Eine spezielle Form von one way hash function ist der **message authentication code (MAC)**. $h = \mathrm{HASH}(M, k)$. k ist ein key.

Beispiel: jeder block cipher zB: public key variante: $H(M) = M^e \pmod p$, langsam aber so stark wie Log in GF(p) zu finden.

Codes brechen

Psychologie, menschliche Schwächen, man in the middle

Social Engineering attacks [MSW02]:

- → Tel anruf: ‘‘Ich bin dein Boss und mein Passwort geht nicht! Frechheit! Setz mir ein neues‘‘
- → Phishing
- → Dumpster Diving
- → Key logger (+Virus, Spyware)
- → Verräter (Geld kauft alles)
- → Hintergrundanalyse der Psychologie der Person, um Passwörter zu erraten (‘‘Cillie‘‘ die Freundin eines Enigma Bedieners)
- → Fehler im Protokoll durch den Menschen (2 mal senden etc)
- → Man in the middle: zB Vorbau vor den Bankomaten (vgl. Phishing), IMSI catcher, Kompromittieren des verwendeten Computersystems, des Netzwerkes, Anzapfen von Stellen, die unverschlüsselt sind: Supermarkt Bankomatbezahlung, transatlantische Unterseekabel, Satellitenkommunikation: Echelon, etc. etc. etc.
- → ...

Aufgabe 8. Erfinde neue Verfahren und Anwendungsgebiete und diskutiere sie!

Traditionelles brechen (symmetrischer) Chiffrierungen

Spannende Beschreibung der Ereignisse rund um Enigma: [Kah91]

- → Wahrscheinlichkeiten der natürlichen Sprache (vgl. Abb 9)
- → Redundanz der natürlichen Sprache
- → Wahrscheinlichkeitsraum des Schlüssels einengen
- → Kasiksi Test
- → Analyse der Chiffrierungsmaschine
- → Vorausberechnung aller IVs, nachsehen, ob ein IV wiederverwendet wurde [BGW01] - WEP 802.11 kann so geknackt werden, ähnlich GSM A5/1:

> ‘‘The attack can find the key in less than a second on a single PC with 128 MB RAM and two 73 GB hard disks, by analysing the output of the A5/1 algorithm in the first two minutes of the conversation. The attack requires a one time parallelizable data preparation stage whose complexity can be traded-off between 2^37 and 2^48 steps.‘‘ [Adi99]

- → Analyse des Zufallgenerators
- → Dictionary attack
- → Chosen Plaintext attack
- → Chosen Ciphertext attack
- → Brute force

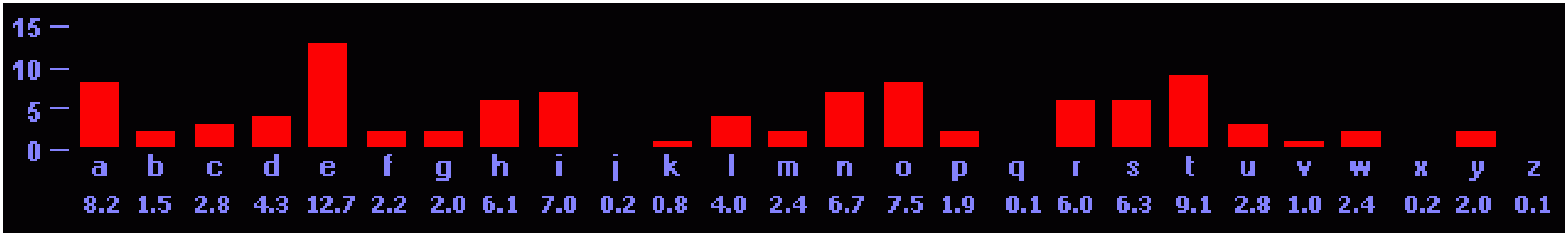


Abbildung 9. Buchstabenverteilung in Englisch

Bekannte Probleme von public key Systemen

- → $D_k(E_k(M)) = M$. Unterschieben von Unterschriften, wenn auto-reply
- → Meist wird der private key wiederum nur mit einer Passphrase geschützt $\Rightarrow$ genauso attackierbar wie symmetrische Verfahren
- → Faktorisierung ist nicht so stark exponentiell, wie wir es gerne hätten

→ Es ist unbekannt, ob große Geheimdienste RSA et.al nicht schon lange knacken können, wir nehmen es nicht an, aber niemand gibt uns die Gewissheit.

→ Wenn N=NP, dann sinnlos.

→ Wenn Quantencomputer existieren, dann sinnlos

→

Begleitende Verfahren

Traffic Analysis, Lokalisierung

Die Zukunft für Kryptoanalyse

- Quantencomputer?, DNA-Computer(?)
- Fortschritte in der Mathematik: Zahlentheorie, Komplexitätstheorie, Informationstheorie, Statistik, theoret. Informatik

Die Zukunft für Kryptographen

- Quantenkryptographie ?
- neue - härtere - Komplexitätsklassen ?
- Beweise der Unknackbarheit ?

Literaturverzeichnis

[**Adi99**] Shamir Adi. Real-time cryptanalysis of gsm's a5/1 on a pc. Personal mail to Mat Blaze url{http://cryptome.org/a51-crack.htm}, 5. Dec 1999.

[**AL91**] A.M.Odlyzko and B.A. LaMacchia. Computation of discreet logarithms in prime fields. In *Designs, Codes, and Cryptography*, pages 46--62, 1991.

[**AMN02**] Kayal N. Agrawal M. and Saxena N. Primes in p. Technical report, Department of Computer Science & Engineering, Indian Institute of Technology Kanpur, Kanpur-208016, INDIA, August 2002. based on Fermat's little theorem.

[**BGW01**] Nikita Borisov, Ian Goldberg, and David Wagner. Intercepting mobile communications: The insecurity of 802.11. url{http://www.isaac.cs.berkeley.edu/isaac/wep-faq.html}, 2001.

[**Bru96**] Schneier Bruce. *Applied Cryptography*. John Wiley & Sons, 1996. The bible for everything related to Cryptography.

[**dRB00**] (disputed) Roger Bacon. *The Voynich Book*. handwritten, 1500-1600.

[**HS90**] Edmund Hlawka and Johannes Schoißengeier. *Zahlentheorie*. Manz-Verlag Wien, 1990. sehr gute Einführung in die Zahlentheorie.

[**Kah91**] David Kahn. *Seizing the Enigma*. Boston, Houghton Mifflin, 1991.

[**MSW02**] Kevin D. Mitnick, William L. Simon, and Steve Wozniak. *The Art of Deception: Controlling the Human Element of Security*. John Wiley & Sons, 2002. Aufzählung diverser Social Engineering tricks, die Kevin Mitnick in den 80er und 90ern erfand.

[**RSA78**] R.L. Rivest, A. Shamir, and L.M. Aldeman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21, Feb 1978. MIT Laboratory for Computer Sciences, Technical Report.

[**Sin99**] Simon Singh. *The Code Book, the secret history of codes and code-breaking*. Doubleday New York, NY, USA, 1999.