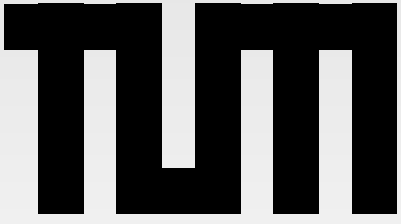
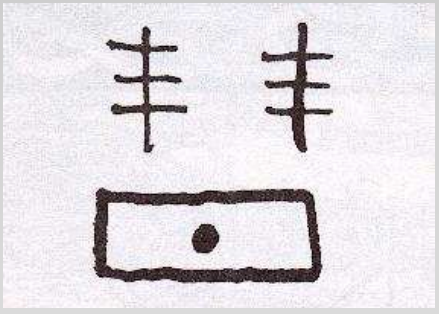
Einführung in die Kryptographie

Stefan Katzenbeisser

Institut für Informatik
Technische Universität München
skatzenbeisser@acm.org



Vom Zeichen zum Code

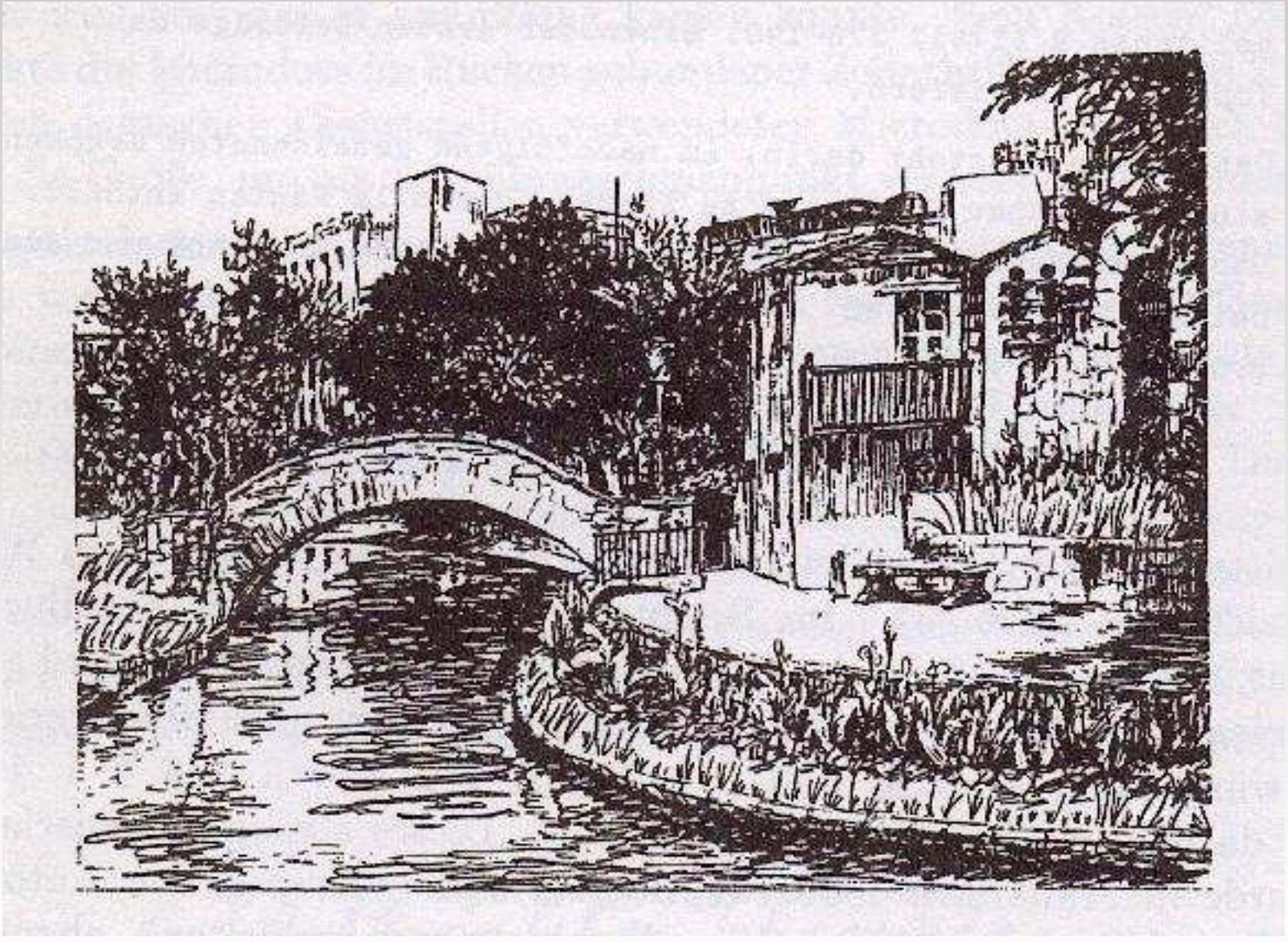


- Älteste Form: Codes repräsentieren Nachrichten
- *Beispiel: Gaunerzinke*

Unterscheidung in

- Steganographie: verdeckte Übermittlung von Nachrichten
- Kryptographie: Codierung von geheimen Texten

Steganographie (1)



Steganographie (2)

Worthie Sir John: — Hope, that is ye beste comfort of ye afflicted, cannot much, I fear me, help you now. That I would saye to you, is this only: if ever I may be able to requite that I do owe you, stand not upon asking me. 'Tis not much that I can do: but what I can do, bee ye verie sure I wille. I knowe that, if dethe comes, if ordinary men fear it, it frights not you, accounting it for a high honour, to have such a rewarde of your loyalty. Pray yet that you may be spared this soe bitter, cup. I fear not that you will grudge any sufferings; only if bie submission you can turn them away, 'tis the part of a wise man. Tell me, an if you can, to do for you anythinge that you wolde have done. The general goes back on Wednesday. Restinge your servant to command. — R. T.

Dritter Buchstabe nach Satzzeichen!

Substitution (1)

- einfache Ersetzung
- Beispiel:

$$\begin{pmatrix} a & b & c & \dots & z \\ Z & T & B & \dots & A \end{pmatrix}$$

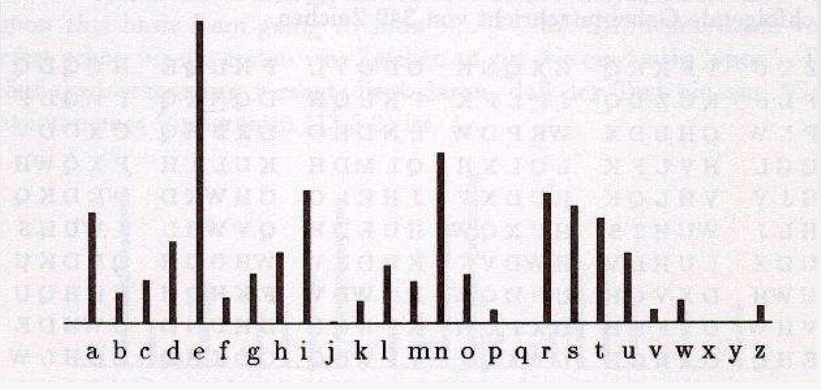
- Permutation!
- *Cäsar-Chiffre:*

$$\begin{pmatrix} a & b & c & \dots & z \\ D & E & F & \dots & C \end{pmatrix}$$

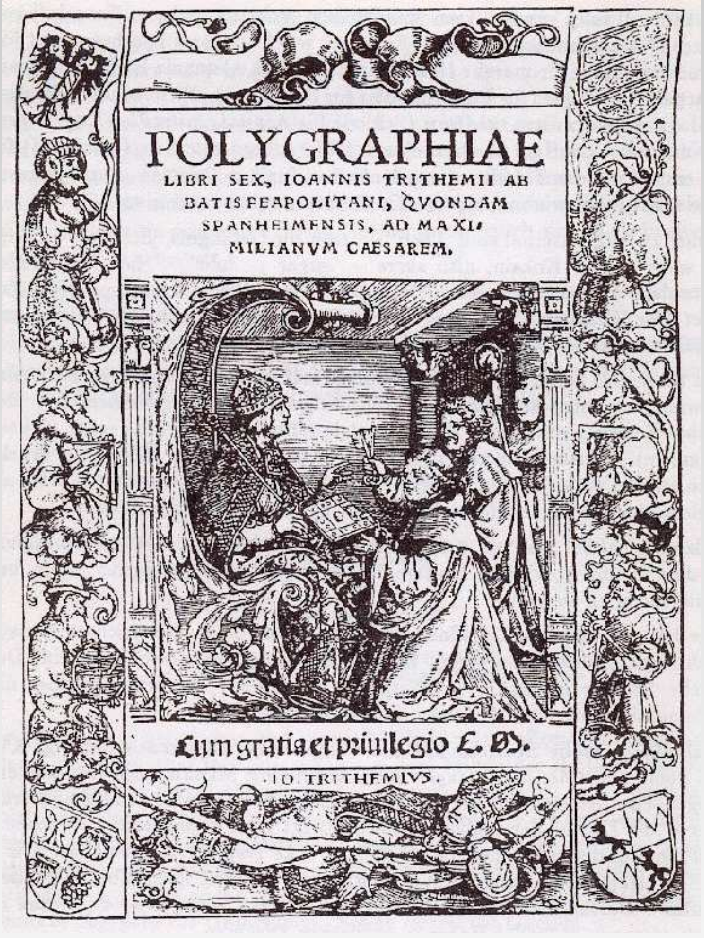
Substitution (2)

Abu Ja'far Muhammad ibn Musa Al-Kwarizmi
(ca. 790-850 n. Chr.)

- Häufigkeiten der Buchstaben unterschiedlich
- Angriff durch statistische Analyse

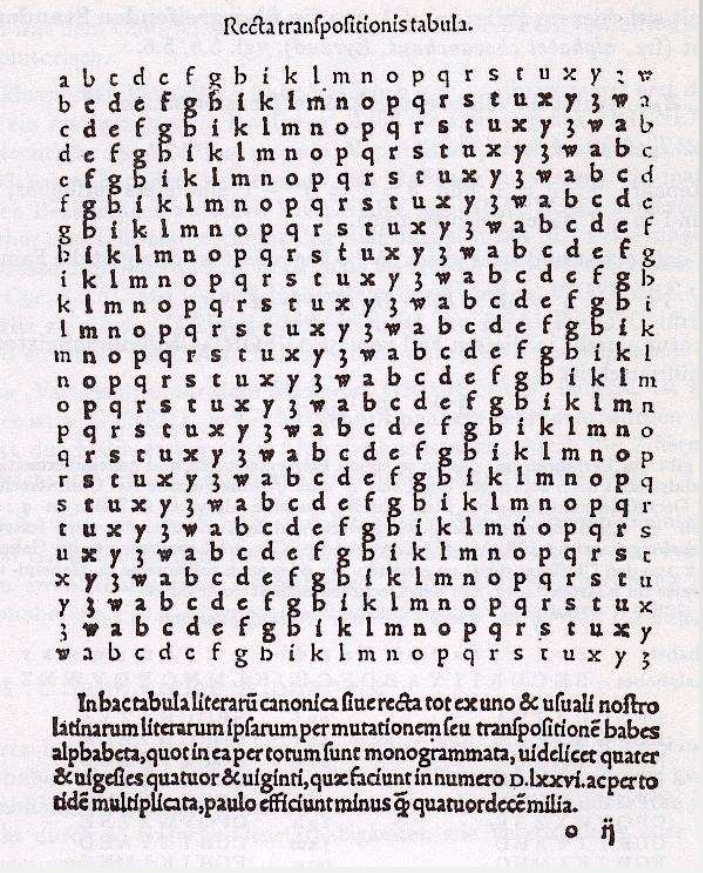


Polyalphabetische Chiffre (1)



- Mehrere monoalphabetische Chiffren verwenden
- Ab ca. 1500 eingesetzt

Polyalphabetische Chiffre (2)



Recta tranſpoſitionis tabula.

a b c d e f g h i k l m n o p q r s t u x y z w
b c d e f g h i k l m n o p q r s t u x y z w a
c d e f g h i k l m n o p q r s t u x y z w a b
d e f g h i k l m n o p q r s t u x y z w a b c
e f g h i k l m n o p q r s t u x y z w a b c d
f g h i k l m n o p q r s t u x y z w a b c d e
g h i k l m n o p q r s t u x y z w a b c d e f
h i k l m n o p q r s t u x y z w a b c d e f g
i k l m n o p q r s t u x y z w a b c d e f g h
k l m n o p q r s t u x y z w a b c d e f g h i
l m n o p q r s t u x y z w a b c d e f g h i k
m n o p q r s t u x y z w a b c d e f g h i k l
n o p q r s t u x y z w a b c d e f g h i k l m
o p q r s t u x y z w a b c d e f g h i k l m n
p q r s t u x y z w a b c d e f g h i k l m n o
q r s t u x y z w a b c d e f g h i k l m n o p
r s t u x y z w a b c d e f g h i k l m n o p q
s t u x y z w a b c d e f g h i k l m n o p q r
t u x y z w a b c d e f g h i k l m n o p q r s
u x y z w a b c d e f g h i k l m n o p q r s t
x y z w a b c d e f g h i k l m n o p q r s t u
y z w a b c d e f g h i k l m n o p q r s t u x
z w a b c d e f g h i k l m n o p q r s t u x y
w a b c d e f g h i k l m n o p q r s t u x y z

In hac tabula literarū canonica ſiue recta tot ex uno & uſuali noſtro latinarum literarum ipſarum per mutationem ſeu tranſpoſitionē habes alphabeta, quot in ea per totum ſunt monogrammata, uidelicet quater & uigeſies quatuor & uiginti, quæ faciunt in numero D.lxxvi. ac per totidē multiplicata, paulo efficiunt minus q̄ quatuordecē milia.

o ij

Ioannis Trithemius
(1462-1516)

- i-tes Alphabet für i-ten Buchstaben verwenden
- Permutation der Alphabete als Schlüssel
- Sicherheit hängt von der Periodenlänge ab!
- *Friedrich Kasiski (1863)*

Kerckhoffs' Axiome

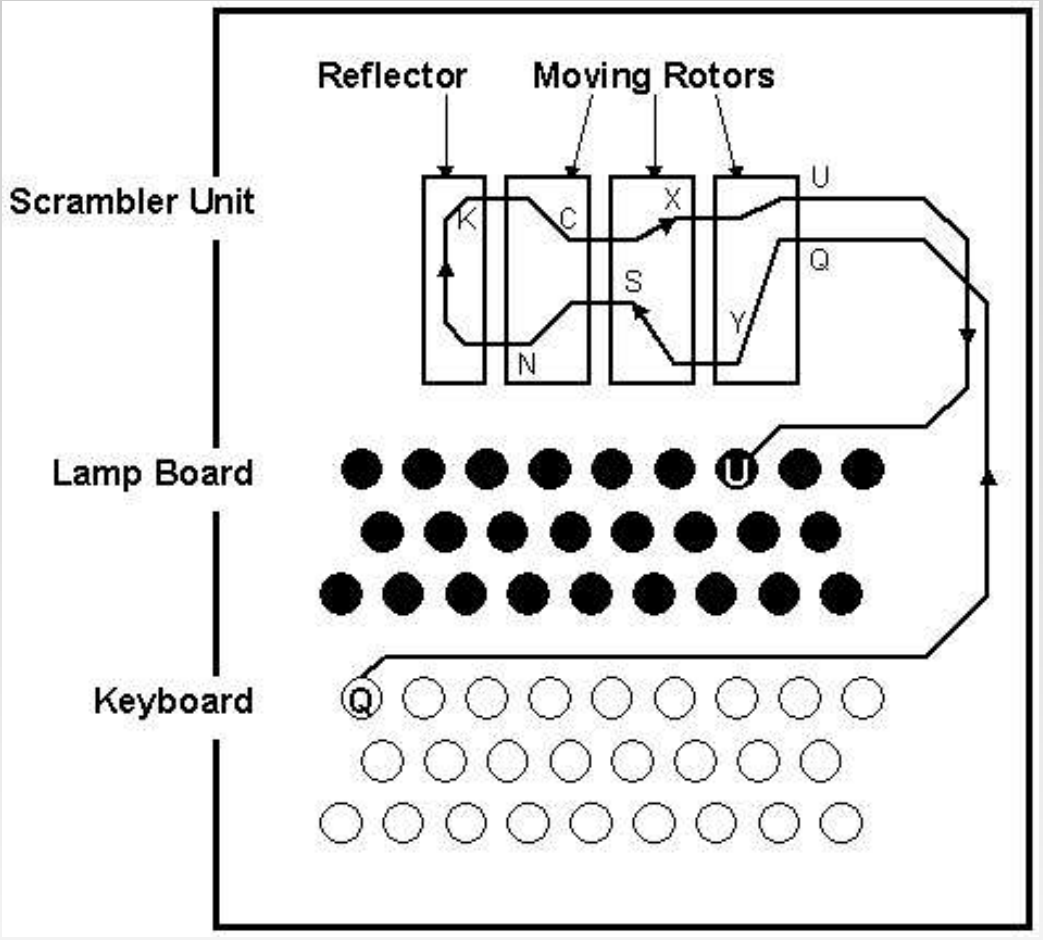
- Man soll den Gegner nicht unterschätzen
- Nur der Kryptanalytiker kann die Sicherheit eines Chiffrierverfahrens beurteilen
- "Der Feind kennt das benutzte System"
- Äußerliche Komplikationen können illusionär sein
- Verstöße gegen die Chiffrierdisziplin sind bei der Analyse einzuplanen

Auguste Kerckhoffs (1835-1903)

Enigma (1)

- Polyalphabetische Chiffre mit langer Periode (max. 26^4)
- Rotorprinzip
- Erfinder: *Edward Hebern*, *Arthur Scherbius*, *Hugo Koch*
- Mehrere Varianten der Enigma in Betrieb

Enigma (2)


Reflector
Moving Rotors
Scrambler Unit
Lamp Board
Keyboard
K
C
X
U
Q
S
Y
N
U
Q

Enigma (3)

Marian Rejewski
(1905-1980)

Alan Turing
(1912-1954)

Symmetrische Chiffre

- *Alice* und *Bob* wollen über einen sicheren Kanal kommunizieren
- *Eve* hört mit

Problem des Schlüsseltausches!

Was ist Sicherheit?

Claude Shannon
1916-2001

- Chiffre ist sicher, wenn

$$\mathbf{P}[M|C] = \mathbf{P}[M]$$

- Informell: Kenntnis von C ändert nichts an der Ungewißheit über M.
- Keine Aussage über Rechenleistung!

Vernam-Chiffre

Klartextmenge = Chiffretextmenge = $\mathbb{F}_2^n$

Verschlüsselung und Entschlüsselung durch:

$\otimes$	0	1
0	0	1
1	1	0

$$E_K(M) = M \otimes K$$

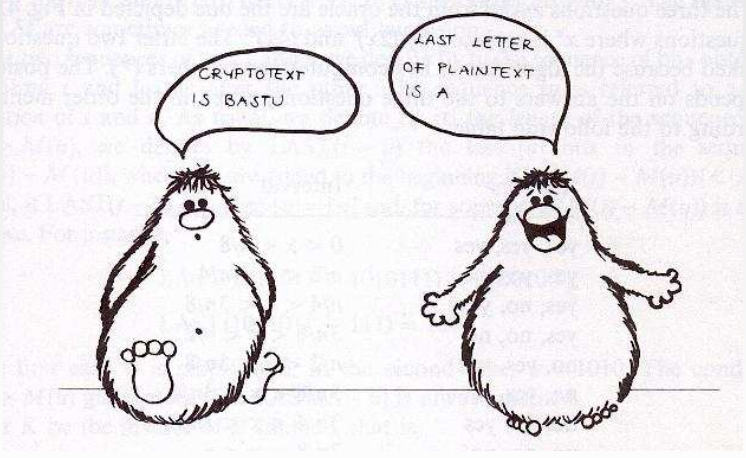
$$D_K(C) = C \otimes K$$

Ein neuer Schlüssel pro Nachricht notwendig!

Satz 1 *Die Vernam-Chiffre ist perfekt sicher.*

Computationale Sicherheit

- Perfekte Chiffriersysteme benötigen lange Schlüssel: $|K| \geq |M|$
- Grund: keine Einschränkung der Rechenleistung des Angreifers
- Computationale Sicherheit: *kein probabilistischer polynomieller Algorithmus kann Eigenschaften des Klartextes "erraten"*



Probabilistische Algorithmen

- . . . können zufällige Entscheidungen treffen
- Ergebnis ist nur mit bestimmter Wahrscheinlichkeit korrekt

Konzept $\rightarrow$ *Tafel!*

Probabilistische TM (1)

Eine probabilistische polynomielle Turing-Maschine ist eine nichtdeterministische TM mit:

- jede Berechnung endet nach exakt n Schritten,
- $n = \mathrm{poly}(|x|)$,
- bei jedem Schritt existieren zwei nichtdeterministische Nachfolgezustände.

Erfolgswahrscheinlichkeit:

$$\mathbf{P}[TM(x) \text{ liefert korrektes Resultat}]$$

Probabilistische TM (2)

Eine polynomielle Monte-Carlo TM für eine Sprache L ist eine probabilistische polynomielle TM mit:

- für $x \in L$, mindestens die Hälfte aller Endzustände liefern TRUE:

$$\mathbf{P}[TM(x) = \text{TRUE} \,|\, x \in L] \geq 1/2$$

- für $x \notin L$ enden alle Berechnungen mit FALSE:

$$\mathbf{P}[TM(x) = \text{TRUE} \,|\, x \notin L] = 0$$

Probabilistische TM (2)

- Die Klasse **RP** enthält alle Probleme, für die eine polynomielle Monte-Carlo TM existiert.
- **co-RP**: Klasse aller Probleme mit Monte-Carlo TM, die keine falsch negative Antwort gibt.
- **ZPP** $=$ **RP** $\cap$ **co-RP** (*Las Vegas Algorithmen*)

Satz 2 $\mathbf{P} \subseteq \mathbf{ZPP} \subseteq \mathbf{RP} \subseteq \mathbf{NP}$.

Probabilistische TM (3)

Eine Sprache L wird durch eine probabilistische polynomielle TM akzeptiert, wenn:

- für $x \in L$, mindestens $2/3$ aller Endzustände liefern TRUE:

$$\mathbf{P}[TM(x) = \text{TRUE} \,|\, x \in L] \geq 2/3$$

- für $x \notin L$ akzeptieren ebenfalls mindestens $2/3$ aller Endzustände:

$$\mathbf{P}[TM(x) = \text{FALSE} \,|\, x \notin L] \geq 2/3$$

Probabilistische TM (4)

Die Klasse aller Sprachen, die durch probabilistische polynomielle TM akzeptiert werden, bezeichnet man **BPP**.

Satz 3 $\mathrm{P} \subseteq \mathrm{BPP} \subseteq \mathrm{NP}$.

Primzahltest

PRIMES

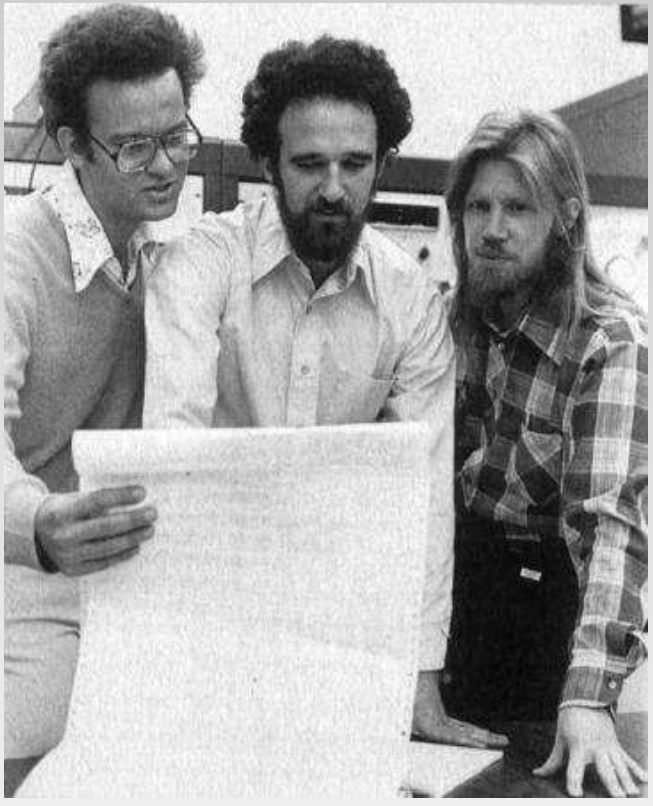
Input: $x \in \mathbb{N}$
Output: TRUE falls x Primzahl, sonst FALSE

Satz 4 (Pratt) PRIMES $\in \mathbf{NP} \cap \mathbf{co-NP}$

Rabin-Miller Test liefert probabilistischen Primzahltest mit Fehlerwahrscheinlichkeit $< 1/4$

Inzwischen kennt man einen polynomiellen Algorithmus: PRIMES $\in \mathbf{P}$

Public-Key Kryptographie (1)



Martin Hellman (Mitte), Whitfield Diffie (rechts)
(zusammen mit Ralph Merkle)

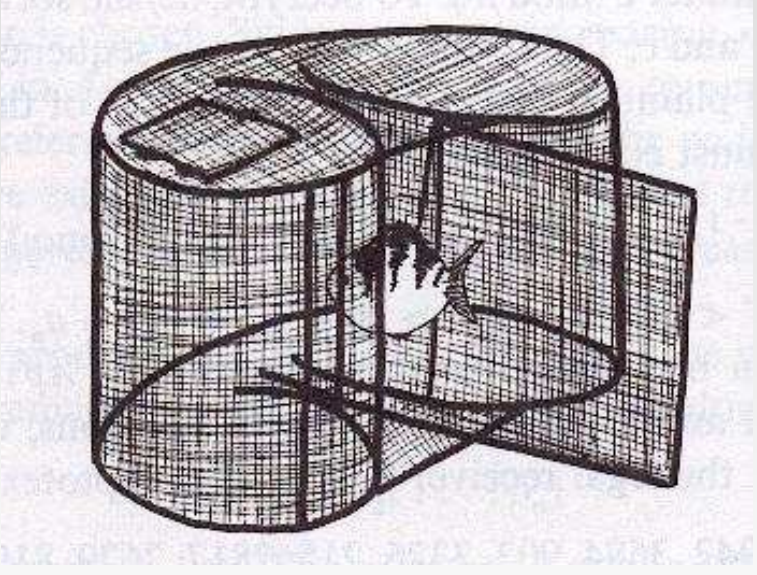
Public-Key Kryptographie (2)

- Trennung der Schlüssel in. . .
- *privaten* Schlüssel und. . .
- und *öffentlichen* Schlüssel

Inverse Funktionen (1)

Grundlage der Kryptographie sind Funktionen, die

- einfach zu berechnen, . . .
- aber schwer zu invertieren sind



Inverse Funktionen (2)

Kann auch die inverse Funktion jeder polynomiell berechenbaren Funktion polynomiell berechnet werden?

Betrachte nur $f(x) = y$ mit $|y| = poly(|x|)$.

Satz 5 *Sei f eine polynomiell berechenbare Funktion. Dann gibt es eine nichtdeterministische polynomielle Maschine M, die das Urbild von f berechnet.*

Beweis $\rightarrow$ Tafel!

Kann man auch eine polynomielle Maschine angeben?

Inverse Funktionen (3)

Satz 6 *Es gilt* $\mathbf{P} = \mathbf{NP}$ *genau dann wenn jede polynomiell berechenbare Funktion eine Inverse besitzt, die ebenfalls polynomiell berechenbar ist.*

Beweis $\rightarrow$ Tafel

Die Kryptographie erfordert jedoch eine probabilistische Sichtweise!

One-Way Functions (1)

Eine Funktion f heißt “one-way function”, falls

- ein polynomieller Algorithmus A zur Berechnung von f existiert und
- für alle probabilistischen polynomiellen Algorithmen A' die Erfolgswahrscheinlichkeit zur Invertierung von f kleiner ist als der Quotient jedes Polynoms.

One-Way Functions (2)

Formal:

Für alle probabilistischen polynomiellen Algorithmen A' und alle Polynome $p(\cdot)$ gibt es einen Index k_0 sodaß

$$\mathbf{P}[f(z) = y :: x \in \{0,1\}^k, y = f(x), z = A'(y)] \leq 1/p(k)$$

für alle $k \geq k_0$ gilt.

One-Way Functions (3)

Satz 7 *Die Existenz von one-way functions impliziert* $\mathbf{P} \neq \mathbf{NP}$.

[Die Public-Key Kryptographie] kann also eine unterhaltsame Diskussion über die leere Menge sein
Prof. Friedrich Bauer, 1993

Trapdoor One-Way Function (1)

Eine one-way function f heißt *trapdoor one-way function*, falls

- ein Algorithmus I existiert,
- der $f(\cdot)$ mithilfe eines Strings $t_k \in \{0,1\}^*$ invertieren kann,
- wobei die Größe von t_k maximal polynomiell wächst: $|t_k| < p(k)$.

Trapdoor One-Way Function (2)

Ein Public-Key Cryptosystem kann aus einer trapdoor one-way function gebildet werden:

- Die Verschlüsselung enthält die Auswertung von f an einer oder mehreren Stellen
- Die Entschlüsselung verwendet das trapdoor, um f zu intertieren; das trapdoor wird zum privaten Schlüssel
- Ein Angreifer muß die one-way function invertieren, da er das trapdoor nicht kennt

Mathematische Grundlagen

- $\mathbb{Z}_n = \{0, 1, \ldots, n-1\}$
- $\langle \mathbb{Z}_n, +, \cdot \rangle$ ist ein Ring
- Alle Elemente $x \in \mathbb{Z}_n$ mit $\mathrm{ggT}(x, n) = 1$ haben ein Inverses
- $\mathbb{Z}_n^* = \{x \in \mathbb{Z}_n \mid \mathrm{ggT}(x, n) = 1\}$
- $\langle \mathbb{Z}_n^*, \cdot \rangle$ ist eine Gruppe mit $\varphi(n)$ Elementen
- Für alle $x \in \mathbb{Z}_n^*$: $x^{\varphi(n)} \equiv 1 \pmod{n}$.
  Beweis $\rightarrow$ *Tafel*

RSA (1)

Ron Rivest, Adi Shamir, Leonard Adleman (1978)

Seien p, q zwei große Primzahlen, $n = pq$ und $1 \leq e \leq \varphi(n)$ mit $\mathrm{ggT}(e, \varphi(n)) = 1$.

RSA-Funktion: $f(x) = x^e \bmod n, \qquad x \in \mathbb{Z}_n^*$

RSA (2)

Die zugehörige inverse Funktion ist gegeben durch

$$f^{-1}(x) = x^d \bmod n$$

wobei d so gewählt wird, daß $ed \equiv 1 \pmod{\varphi(n)}$.

Beweis $\rightarrow$ *Tafel*

- $\langle e, n \rangle$ ist öffentlicher Schlüssel
- d ist privater Schlüssel (trapdoor)

RSA (3)

Satz 8 *Die Berechnung von d aus $\langle e, n \rangle$ ist polynomiell äquivalent zur Berechnung der Primfaktorzerlegung von n*

Satz 9 FACTORIZE $\in \mathbf{NP} \cap \mathbf{co} - \mathbf{NP}$.

Es ist kein polynomieller Algorithmus zur Berechnung der Primfaktorzerlegung bekannt.

RSA ist ein Kandidat für eine one-way function!

Die exakte Relation zwischen RSA und FACTORIZE *ist aber unbekannt!*

Diskreter Logarithmus (1)

Sei G eine endliche zyklische Gruppe:

$$G = \{g^{e_1}, \ldots, g^{e_n}\}$$

In der Kryptographie verwendet man:

- $\mathbb{Z}_p^*$ mit p Primzahl,
- Elliptische Kurven,
- Multiplikative Gruppen endlicher Körper,
- …

Diskreter Logarithmus (2)

DISCRETELOG über $\mathbb{Z}_p^*$

Input: p, g, x
Output: e sodaß $g^e = x \bmod p$

Dieses Problem scheint härter zu sein als FACTORIZE*!*

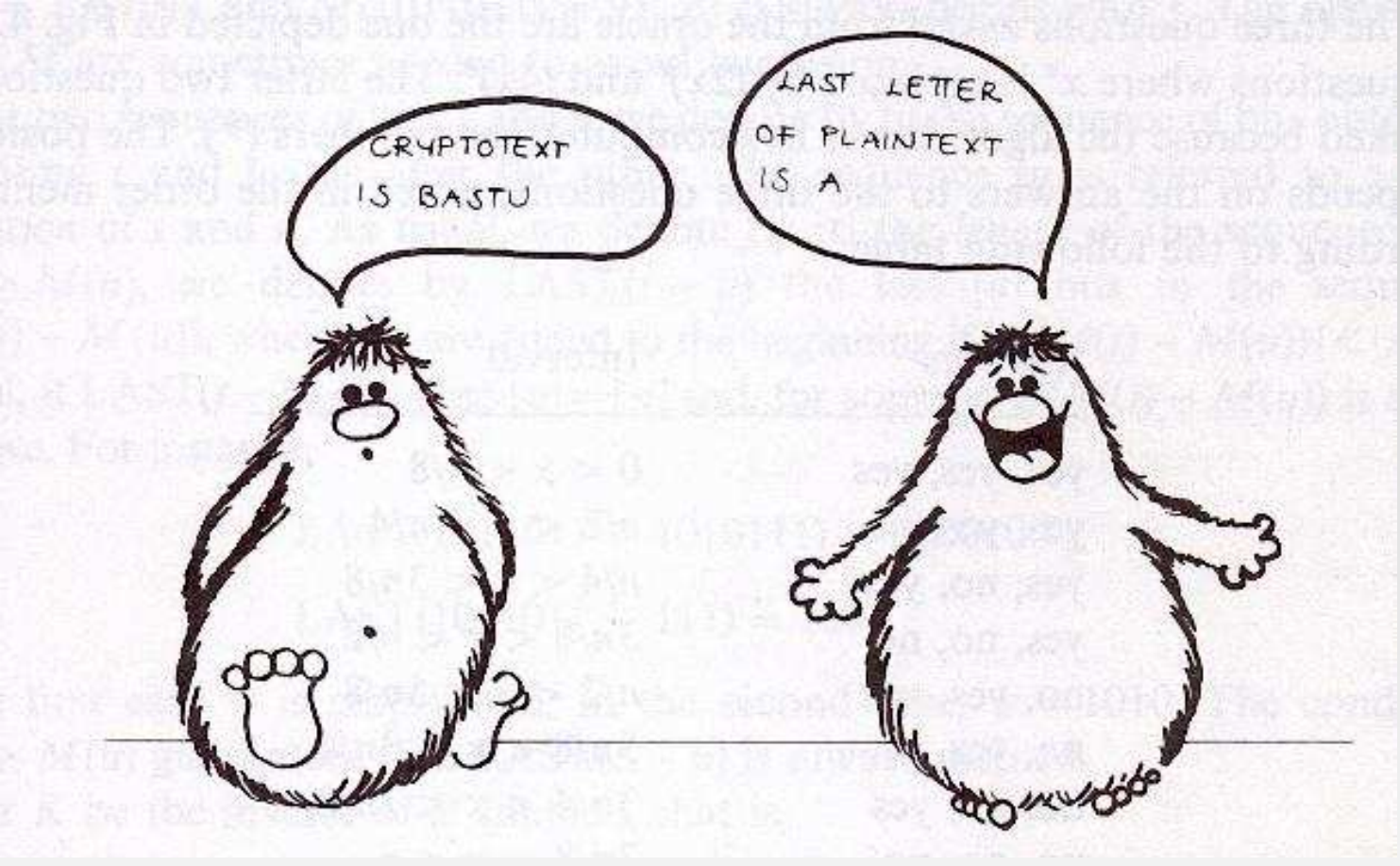
Basis des ElGamal Public-Key Kryptosystems

Diskreter Logarithmus (3)

ElGamal:

- Sei g ein Generator von $\mathbb{Z}_p^*$ und $h \equiv g^a \bmod p$, für eine Zufallszahl a
- Öffentlicher Schlüssel: $\langle p, g, h \rangle$
- Privater Schlüssel: a
- Verschlüsselung wählt Zufallszahl k und …
- produziert Tupel
  $\langle y_1, y_2 \rangle := \left\langle g^k \bmod p, x h^k \bmod p \right\rangle$
- Entschlüsselung: $y_2 (y_1^a)^{-1} \bmod p$

Computationale Sicherheit (1)



Computationale Sicherheit (2)

Forderungen:

- Der private Schlüssel soll nicht aus dem öffentlichen Schlüssel berechenbar sein
- Nachrichten sollten nicht aus dem Chiffrat ableitbar sein
- Aus dem Chiffrat alleine soll keine Information über den Klartext effizient berechenbar sein

Kryptanalyse

Satz 10 *Sei $h : M \rightarrow \{0,1\}^*$ ein polynomiell berechenbares Prädikat über der Menge aller Klartextnachrichten M. Für jedes deterministische Public-Key Kryptosystem gilt: das Problem der Berechnung von $h(m)$ aus dem Chiffrat c liegt in* $\mathbf{NP} \cap \mathbf{co} - \mathbf{NP}$.

Beweis $\rightarrow$ Tafel

Semantische Sicherheit (1)

- Ausgangspunkt: sei $h : M \rightarrow \{0,1\}^*$ eine Funktion über der Menge aller Klartextnachrichten M.
- Ziel: ein Chiffriersystem heißt semantisch sicher, wenn *alle Funktionen* h "schwer" zu berechnen sind, falls nur verschlüsselte Nachrichten vorliegen.

Formal wird diese Eigenschaft über zwei Spiele formuliert.

Semantische Sicherheit (2)

A... Angreifer, B... Richter

Spiel 1.

B wählt $m \in M$.
A errät $h(m)$.

Spiel 2.

B wählt $m \in M$.
B sendet $c = E(m)$ an A.
A berechnet $h(m)$ aus c.

Ein System ist semantisch sicher, falls *für alle* h die Wahrscheinlichkeit für A das Spiel 2 zu gewinnen *nicht viel größer* ist als die Wahrscheinlichkeit Spiel 1 zu gewinnen.
Ein Spiel ist ein probabilistischer Algorithmus!

Ununterscheidbarkeit (1)

Ein Chiffriersystem ist “polynomiell ununterscheidbar”, wenn ein Angreifer keinen probabilistischen polynomiellen Algorithmus verwenden kann kann, um zwei Nachrichten zu generieren, deren Chiffrate er in polynomieller Zeit unterscheiden kann.

Spiel 3.

- A wählt $m_1, m_2 \in M$ und sendet diese an B.
- B wählt zufällig m_1 oder m_2,
- …und sendet das Chiffrat c an A
- A entscheidet, ob er das Chiffrat von m_0 oder m_1 erhalten hat.

Ununterscheidbarkeit (2)

Formal heißt ein Chiffriersystem polynomiell ununterscheidbar falls (für alle Polynome p) die Wahrscheinlichkeit für A, Spiel 3 zu gewinnen, maximal $1/2 + 1/p(n)$ ist.

Satz 11 *Ein Chiffriersystem ist semantisch sicher genau dann wenn es polynomiell ununterscheidbar ist.*

Ist RSA semantisch sicher?

Nein; es gibt einen Algorithmus, der RSA-Chiffrate unterscheidet:

- A generiert m_0 und m_1.
- B wählt m_0 oder m_1 aus; nenne diese Nachricht m.
- B sendet $c = m^e \bmod n$ an A.
- A testet, ob $c = m_0^e \bmod n$ oder $c = m_1^e \bmod n$

Satz 12 *Jedes deterministische Public-Key Kryptosystem ist nicht semantisch sicher.*

RSA-OAEP

- randomisiertes RSA
- $G : \{0,1\}^{k_0} \longrightarrow \{0,1\}^{n+k_1}$ pseudorandom generator
- $H : \{0,1\}^{n+k_1} \longrightarrow \{0,1\}^{k_0}$ hash function
- r ist Zufallszahl der Länge k_0
- Verschlüsselung durch $E(x) = RSA(s\|t)$, wobei

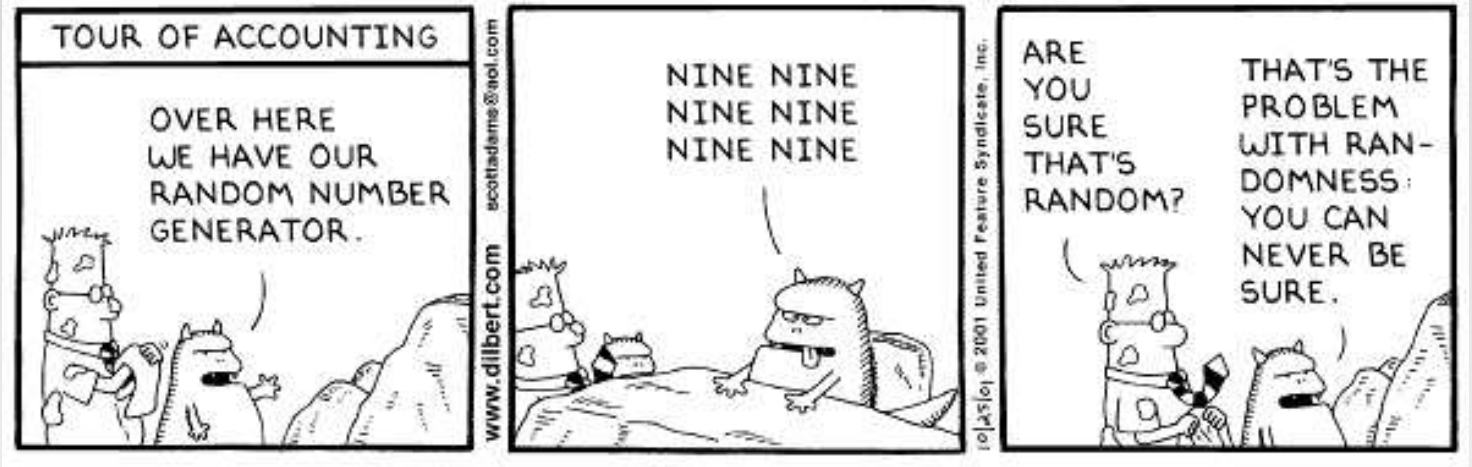
$$s = (x\|O^{k_1}) \otimes G(r)$$

$$t = r \otimes H(s)$$

- RSA-OAEP ist semantisch sicher

Zufallszahlen (1)

Wie kann man sicher Zufallszahlen erzeugen?



Copyright © 2001 United Feature Syndicate, Inc.

Zufallszahlen (2)

Wichtigste Sicherheitseigenschaft: produzierte Bits sollen nicht “vorhersagbar” sein

Next Bit Test:

- Test erhält bisherige Pseudozufallszahlen: $t_1, \ldots, t_n$
- …und soll nächste Zahl t_{n+1} vorhersagen.
- Ein Zufallszahlengenerator ist sicher, falls *jeder* Test nur “raten” kann.

Hard Core Predicates (1)

Ein Prädikat B heißt “hard-core” für eine one-way function f, falls

- $B(x)$ in polynomieller Zeit berechenbar ist, aber
- jeder probabilistische polynomielle Algorithmus mit Input $f(x)$ den Wert $B(x)$ nur “erraten” kann.

$$\mathbf{P}[z = B(x) :: x \in \{0,1\}^k, z = A'(f(x))] \leq \tfrac{1}{2} + 1/p(k)$$

Hard Core Predicates (2)

Beispiel:

- LSB: least significant bit
- Die Berechnung des LSB des Klartextes aus dem Chiffretext ist polynomiell äquivalent zu einem Angriff gegen RSA.
- Falls RSA sicher ist, kann auch der Wert des LSB nicht effizient berechnet werden.
- LSB ist dann ein Hard Core Predicate.

Hard Core Predicates (3)

Jede one-way function f mit Hard Core Predicate B liefert einen sicheren Zufallszahlengenerator:

$$\begin{aligned} x &\rightarrow f(x)\|B(x) \\ f(x) &\rightarrow f(f(x))\|B(f(x)) \\ f^2(x) &\rightarrow f^3(x)\|B(f^2(x)) \\ \vdots &\quad\ \vdots \end{aligned}$$

Nehme $B(x)\|B(f(x))\|B(f^2(x))\|\ldots$ als Zufallszahl bei seed x.