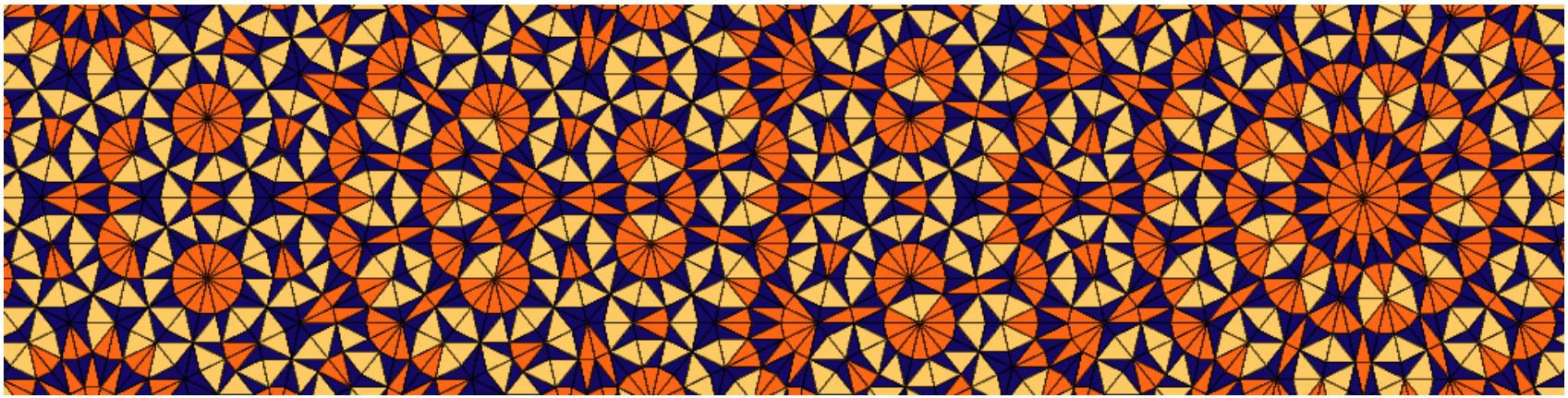
Panorama der Mathematik und Informatik

17: Kryptographie II + III

Dirk Frettlöh
Technische Fakultät / richtig einsteigen



Mechanische Kryptografie

Bis ins 20. Jhdt. waren Codes also entweder sehr aufwendig: lange Ver- und Entschlüsselungsdauer, Problem der sicheren Schlüsselverteilung (Grand Chiffre); oder sie waren zu knacken (Vigenère).

Es gab kuriose Ideen, die dennoch klappten: Im 2. Weltkrieg setzten die USA z.B. Navajo-Indianer als Funker ein: ihre Sprache ist mit keiner anderen verwandt, es gab genügend Sprecher und (wichtig!) kein deutscher Anthropologe hatte bislang die Navajo besucht.

Um diese Zeit wurde ein komplett sicheres Verfahren entwickelt: das **one-time pad**.

One time pad

"Einmalschlüssel". Unter folgenden Bedingungen bietet das Verfahren (math. beweisbar) perfekte Sicherheit.

- Schlüssel ist völlig zufällig (gleichverteilt)
- Schlüssel ist (mindestens) so lang wie die Nachricht
- Schlüssel ist geheim
- Schlüssel wird nur einmal verwendet

Klartext + Schlüssel = Geheimtext $\mod 26$.

Klartext:	`angriffimmorgengrauen`
Schlüssel:	`WZSLXWMFQUDMPJLYQOXXB`
Geheimtext:	`XNZDGCSODHSEWOZFIPSCP`

a+W=X, n+Z=N, ... n+B=P ist also $1 + 23 = 24$, $14 + 26 = 14$, ... $14 + 2 = 16 \mod 26$.

Zum Entschlüsseln einfach Geheimtext - Schlüssel = Klartext.

Perfekte Sicherheit, was will man mehr? Nachteile:

- Länge Schlüssel = Länge Nachricht
- Transport
  - Aufwand: wie? *Alle* Schlüssel an an *alle* militärischen Einheiten?
  - Sicherheit: Schlüssel muss geheim bleiben
- Aufwand: Jeden Schlüssel nur einmal benutzen

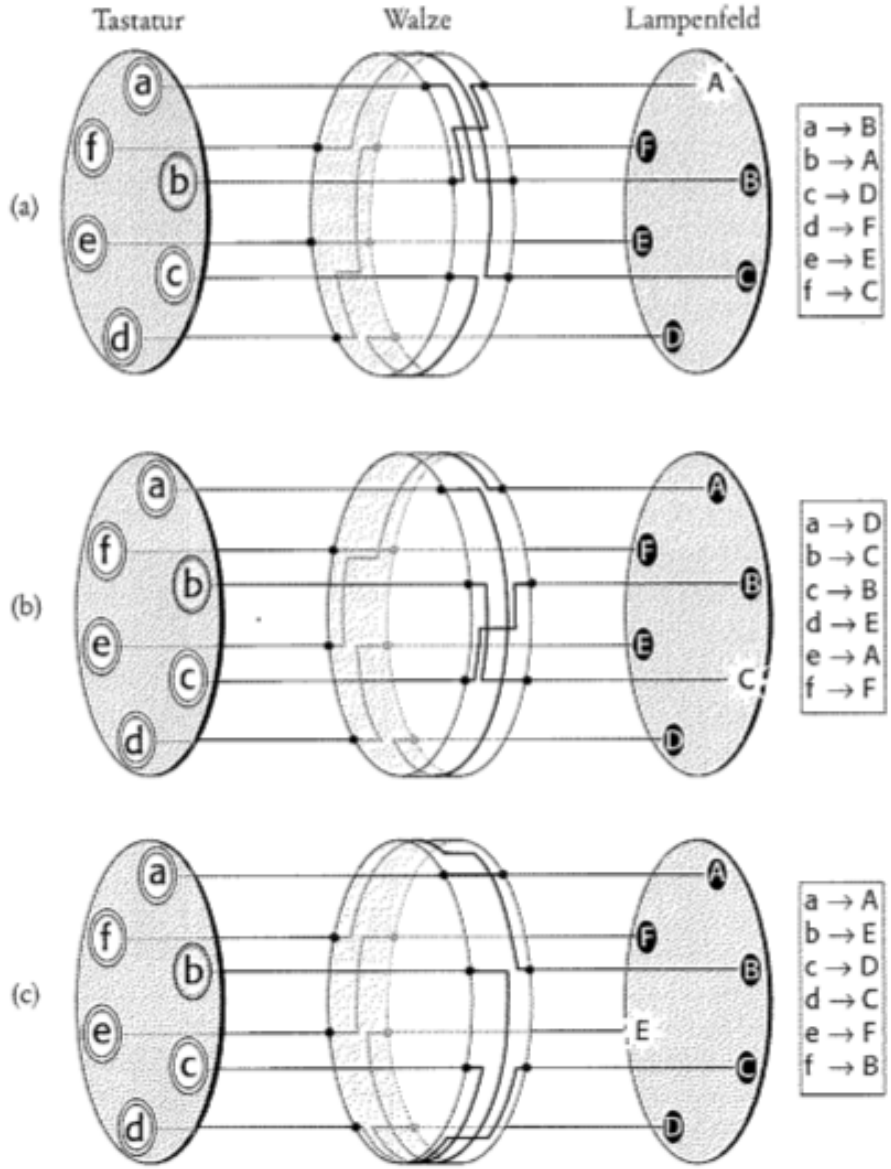
Praktisch nur in bestimmten Fällen nützlich.

Mechanisierung bot eine praktikablere Lösung.

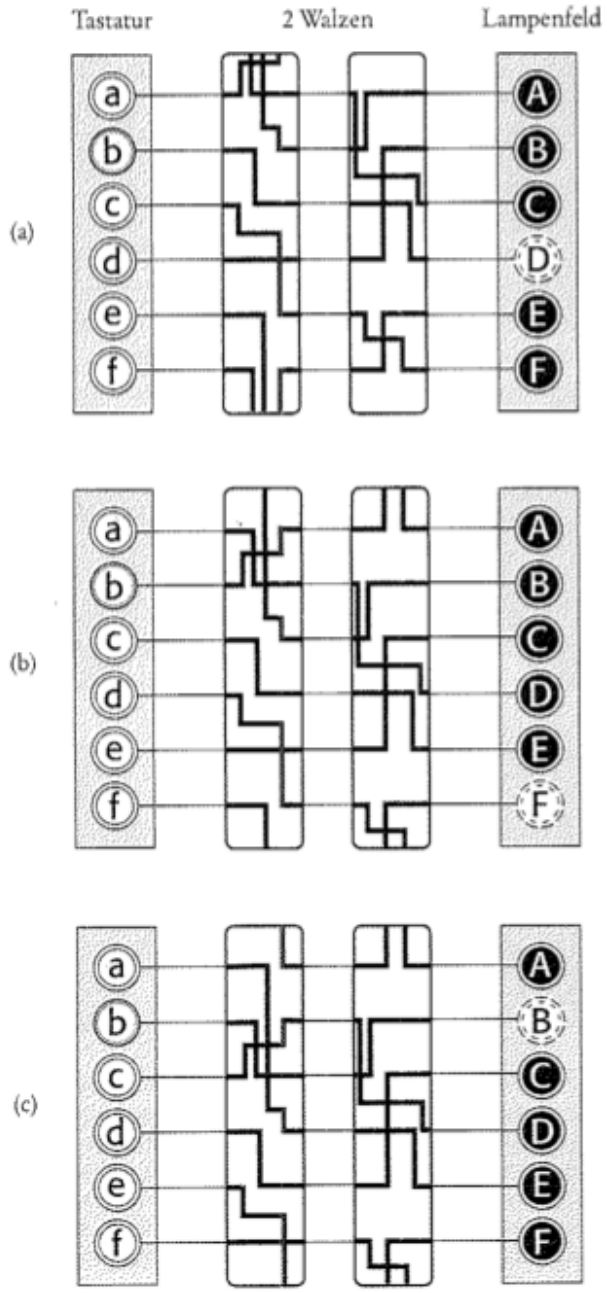
Etliche ähnliche Maschinen wurden nach 1915 entwickelt. Die **Enigma** war die einzige, die sich in den 20ern weit verbreitete (in Deutschland). Da Deutschland im 1. Wk ein paar Pannen mit schlechter Verschlüsselung erlebte, setzte das Militär sie hier schon früh ein (googele "Zimmermann-Depesche")

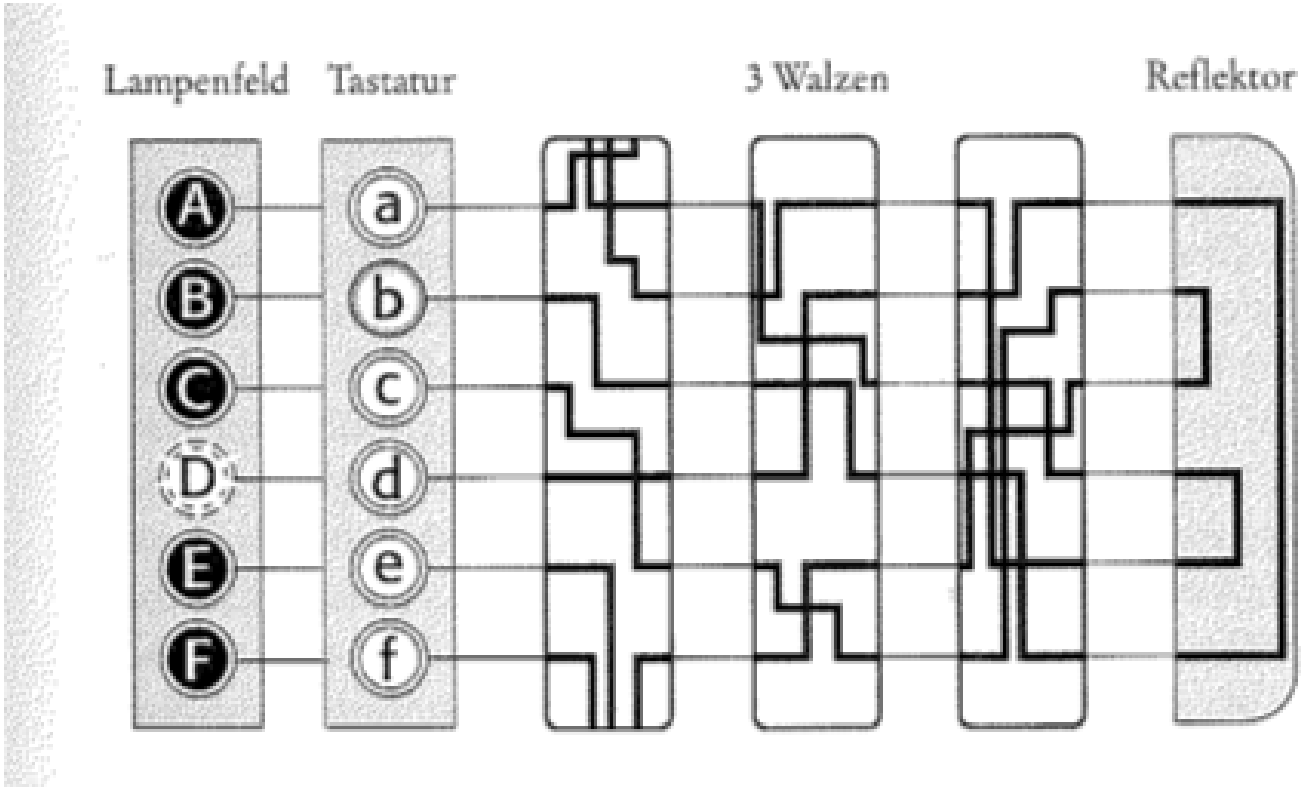
Funktionsweise der Enigma: Drei Walzen, jede realisiert einen Substitutionscode.

Erste Walze dreht sich immer einen Schritt weiter (das Bild zeigt drei Schritte *einer* Walze, hier nur für 6 Buchstaben, Bilder alle aus S.Singh: The Code Book):



Hat sich die erste Walze einmal komplett gedreht, dreht die zweite Walze einen Schritt weiter. (Bild: (a) kurz vorher; (b) Walze 2 dreht; (c) kurz nachher). Jeder Buchstabe wird also mit einem anderen Substitutionscode verschlüsselt.





Es gibt außerdem einen Reflektor: Signal läuft hin und zurück. In jedem Schritt ein anderer Substitutionscode.

Es sind immer zwei Buchstabenpaare verbunden: Hier A$\leftrightarrow$C, B$\leftrightarrow$D, E$\leftrightarrow$F. (Beachte! Niemals: A$\leftrightarrow$A, B$\leftrightarrow$B, ...)

Weil $A \to C$ und $C \to A$ usw. ist gilt: Bei gleichen Anfangseinstellungen ist Ver- und Entschlüsseln symmetrisch:

Verschlüsseln: Tippe Klartext ein, lies Geheimtext ab.
Entschlüsseln: Tippe Geheimtext ein, lies Klartext ab.

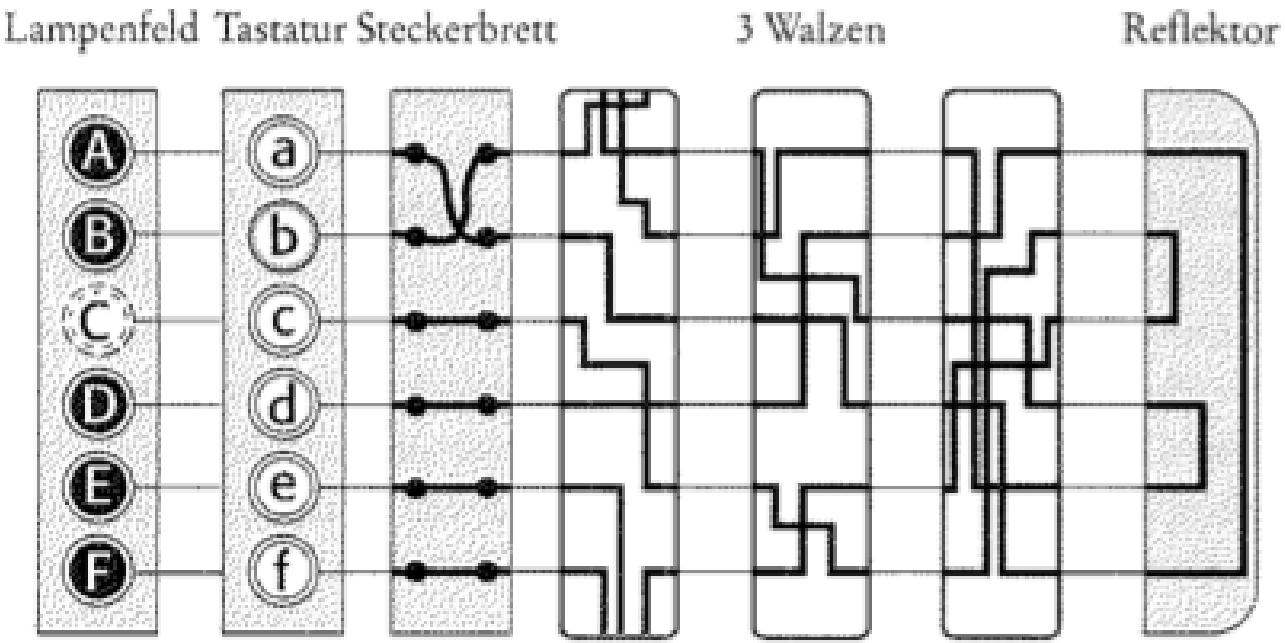
Zahl der Schlüssel:

Es gibt drei Walzen, die können getauscht werden: 6 Mögl.

Es gibt 26 Ausgangsstellungen pro Walze: $26^3 = 17576$ Mögl.

Also insgesamt nur $6 \cdot 17576 = 105456$ Anfangsstellungen. Zu wenig! Daher

Schalte ein Steckerbrett dazwischen:



Hier wird nur ein Paar Buchstaben vertauscht. Real: fünf Paare.

Damit gibt es ca. 10^{16} Anfangsstellungen = mögl. Schlüssel.

Es gibt noch ein weiteres Element ("Ring"), das ist nicht so entscheidend für das Prinzip.

Entschlüsselung der Enigma: Dazu mussten viele Faktoren zusammenkommen. Kurz:

- Spionage (Thilo Schmidt lieferte Franzosen Fakten zur Enigma)
- Unoptimales Vorgehen der Verschlüssler
- Angst der Polen vor (Nazi-)Deutschland
- Mathematiker (Rejewski, Turing,...)

Die Franzosen arbeiteten gar nicht selbst an der Entschlüsselung der Enigma, sie hielten die für unmöglich. Die Polen aber taten das; sie bekamen von den Franzosen Informationen dazu und bauten die Enigma nach.

"Der polnische Erfolg beim entschlüsseln der Enigma beruhte auf drei Faktoren: Angst, Mathematik und Spionage" (S.Singh)

Vorgehen der Deutschen: Es gab *Tageschlüssel* (festgelegt in Codebüchern, die z.B. jeden Monat neu erstellt und verteilt wurden). Ein Tagesschlüssel sah so aus:

- Walzenreihenfolge: 3,1,2
- Walzen einstellen auf: Q, G, U
- Steckerbrett: A/L - P/R - T/D - W/Q - O/Z

Der besseren Sicherheit wegen wurde mit dem Tagesschlüssel der *Spruchschlüssel* gesendet: Sende mit den Einstellungen oben z.B. GKAGKA. Stelle dann die Walzen auf G, K, A und verschlüssele damit die eigtl. Nachricht.

Das zweifache Senden des Tagesschlüssels diente der Reduzierung von Irrtümern. Lieferte aber einen Ansatz zur Entschlüsselung.

Marian Rejewksi nutzte die Wiederholung: ein Spruchschlüssel wird z.B. in einer Nachricht verschlüsselt als LOKRGM; der in einer anderen Nachricht als MVXTZE; in einer dritten als JKTMPE; in einer vierten als DVYPZX.

Er weiß: L und R stehen für den selben Buchstaben (3 Zeitschritte versetzt), genauso M und T, J und M, D und P. Erstelle eine Tabelle:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
   P      M RT
```

Gibt es genug Funksprüche, so kann die Tabelle vervollständigt werden.

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
FQHPLWOGBMVRXUYCZITNJEASDK
```

Die sagt noch nichts über den Schlüssel, erst recht nicht über das Steckerbrett. Aber das liefert einen "Fingerabdruck" der Walzenstellung.

Rejewski erkannte, dass die Zykelstruktur nur von den Walzen abhängt, nicht vom Steckerbrett.

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
FQHPLWOGBMVRXUYCZITNJEASDK
```

Hier:

$A \to F \to W \to A$	Länge 3
$B \to Q \to Z \to K \to V \to E \to L \to R \to I \to B$	Länge 9
$C \to H \to G \to O \to Y \to D \to P \to C$	Länge 7
$J \to M \to X \to S \to T \to N \to U \to J$	Länge 7

Dasselbe kann man nun mit den zweiten und dritten Buchstaben des Spruchschlüssels machen.

Andere Walzenstellungen haben andere Zykelstrukturen: Statt $(3, 7, 7, 9)$ wie oben z.B. $(4, 4, 5, 6, 7)$ usw.

Mit dem Nachbau der Enigma konnten die Polen eine Tabelle der $6 \cdot 26^3$ Walzenstellungen erstellen.

Für jedes komplette Zykelmuster ("Fingerabdruck", z.B. (3,7,7,9)-(4,6,8,8)-(3,4,6,6,7)) bleiben nur eine (oder wenige) Möglichkeiten der Walzenstellungen. Die kann man nun einstellen.

Der daraus resultierende Text ist nur noch mit dem Steckerbrett verschlüsselt: einfacher Substitutionscode. Z.B. Häufigkeitsanalyse; oder noch einfacher: 16 von 26 Buchstaben sind ja korrekt.

"alkulxtilbernil" heißt also sicher: Ankunft in Berlin. Also L $\leftrightarrow$ R, X $\leftrightarrow$ F usw.

Kurz vor Ausbruch des 2. Weltkriegs erhöhte Deutschland die Zahl der Walzen auf 5. Statt $3 \cdot 2 \cdot 1 = 6$ nun $5 \cdot 4 \cdot 3 = 60$ Mögl.

Mittlerweile hatten die Polen das "Fingerabdruck"-suchen automatisiert: "Bomba" hießen die Maschinen.

Wegen der größeren Probleme beschlossen sie, den Franzosen und Engländern ihr Wissen zur Verfügung zu stellen.

(Bundesarchiv Bild 101I-769-0229-10A)

Im englischen Bletchley Park arbeiteten im 2. Wk. Hunderte Wissenschaftler (u.a. Turing) und Tausende weitere an der Entschlüsselung der Enigma (u.a.: Italien, Japan... andere deutsche Chiffriermaschine: "Lorenz") Details in vielen Büchern und Filmen. (Stichworte "Enigma", "Codename Ultra". Meine Empfehlung: N.Stephenson: "Cryptonomicon")

Kurz: Premierminister Churchill erkannte die Wichtigkeit dieser Arbeit (im Ggs. zu anderen hohen Tieren). Genügend Mittel $\Rightarrow$ Erfolg beim Entschlüsseln $\Rightarrow$ Vorteil im Krieg.

Aus den automatisierten Entschlüsselungsmaschinen Entwicklung eines der ersten Computer: Colossus. Wegen Geheimhaltung blieb das alles bis in die 70er unbekannt!

Aus Bletchley Park wurde nach dem 2. Weltkrieg das GCHQ (Government Communications Headquarters), das britische Gegenstück zur NSA (s. Nachrichten, Edward Snowden).

Der Computer macht's möglich

Mit dem Einsatz von Computern und Mathematik gehen natürlich noch ganz andere Sachen. Im Folgenden:

- Sicherer Schlüsselaustausch: Diffie-Hellman
- Sichere Verschlüsselung: RSA Verschlüsselung
- Fälschungssichere Unterschrift: RSA Signatur

Alle drei Lösungen wurden 1976/77 gefunden.

Recall: One-time-pads sind theoretisch sicher. Man muss nur einen Weg finden, die Schlüssel sicher zu transportieren.

Alice und Bob

Generelle Situation ab jetzt:

Alice (gut) verschlüsselt eine Nachricht, schickt sie an **Bob** (gut), der entschlüsselt sie. **Eve** (böse) fängt Nachricht ab und liest/ändert sie, oder Eve gibt sich als Alice aus.

Gesucht: Sichere Schlüsselübertragung, und/oder unentschlüsselbare Nachricht, und/oder fälschungssichere "Unterschrift" (signature)

[Schloss- und Kisten-Geschichte]

Wichtig: **Einwegfunktionen** (trapdoor functions). Idee: eine Richtung einfach zu berechnen, andere schwer bis unmöglich.

Bsp.: $1007 = a \cdot b$, $(a, b \in \mathbb{N} \setminus \{1\})$. $a =?$ $b =?$
Dagegen: $19 \cdot 53 = ?$ 1007.
Multiplizieren: einfach. Faktorisieren: schwierig.

Diffie-Hellman Schlüsseltausch

Einwegfunktion hier: $3^? = 1 \mod 7$ vs. $3^6 =? \mod 7$.
(**Diskreter Logarithmus** vs diskretes Potenzieren)

Wichtig im Folgenden: $(3^a)^b = 3^{ab} = (3^b)^a$.

Prinzip:

- Alice und Bob einigen sich auf zwei Zahlen Y, P (mit $Y < P$) die sie austauschen (per Telefon, Postkarte, egal, Y und P sind nicht geheim).
- Alice wählt geheime Zahl A, Bob wählt geheime Zahl B.
- Alice schickt $Y^A \mod P$ an Bob, Bob schickt $Y^B \mod P$ an Alice.
- Alice berechnet $s = (Y^B)^A \mod P$, Bob $s = (Y^A)^B \mod P$.
- s ist der Schlüssel (z.B. als one-time-pad).

Bsp.:

- Alice und Bob vereinbaren $Y = 7$ und $P = 11$.
- Alice wählt $A = 3$, Bob $B = 6$.
- Alice schickt $7^3 = 343 = 2 \mod 11$, also 2, an Bob, Bob schickt $7^6 = 343 \cdot 343 = 2 \cdot 2 = 4 \mod 11$, also 4, an Alice.
- Alice erhält $4^A = 4^3 = 64 = 9 \mod 11$, Bob $2^B = 2^6 = 64 = 9 \mod 11$.
- Also Schlüssel $s = 9$.

Natürlich sind echte Schlüssel viel höhere Zahlen. Ihre Binärdarstellung dient dann als one-time-pad (für eine Binärbotschaft).

Eve kennt $Y = 7$, $P = 11$, $7^A = 2 \mod 11$ und $7^B = 4 \mod 11$. Sie braucht $(7^A)^B = (7^B)^A = 7^{AB}$. Dazu braucht sie A oder B, muss also $7^? = 2 \mod 11$ oder $7^? = 4 \mod 11$ lösen.

Falls es keinen cleveren Algorithmus zum Lösen von $a^? = b \mod p$ gibt, können Y, P, A, B so hoch gewählt werden, dass ein brute-force-Ansatz Jahrtausende braucht. (Bis neulich typische Größe 1024 bit, also 2^{1024})

Nun (1976/77) da die Idee mit Einwegfunktionen in der Welt ist: Gesucht: "asymmetrisches" Verfahren. D.h.: Verschlüsseln einfach, Entschlüsseln extrem aufwendig, selbst wenn das Verfahren bekannt ist (!) (aber der Schlüssel nicht).

RSA

Nach Rivest, Shamir, Adleman: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, *Communications of the ACM* 21 (1978) 120-126.

Einwegfunktion: Faktoriseren vs. Multiplizieren.

- Faktorisiere 175 828 273.
- Multipliziere 17 159 und 10 247.

In einem Artikel über RSA setzt M. Gardner 1977 einen Preis von 100 $ aus für die Faktorisierung von "RSA-129" (129 bit)

114381625757888867669235779976146612010218296721242362562561842935706935245733897830597123563958705058989075147599290026879543541

Gelang 1994 mit 1600 per Internet verbundenen Computern:

$$3490529510847650949147849619903898133417764638493387843990820577 \cdot 32769132993266709549961988190834461413177642967992942539798288533$$

Prinzip RSA:

- Alice wählt zwei Primzahlen p und q (geheim).
- Alice berechnet $N = p \cdot q$ und $F = (p-1)(q-1)$.
- Alice wählt e mit $\mathrm{ggT}(e, F) = 1$ und d mit $e \cdot d = 1 \mod F$. (e: *encrypt*, d: *decrypt*)
- Alice gibt N und e öffentlich bekannt. F und d sind geheim.
- Bob kann nun eine Botschaft in eine Zahl m (für *message*) verwandeln (z.B. Text $\to$ ASCII $\to$ Binärzahl $\to$ Dezimalzahl) und m verschlüsseln als $m^e \mod N$.
- Alice berechnet $(m^e)^d = m^{ed} = m \mod N$.

Warum klappt $m^{ed} = m \mod N$?

Ein **Satz von Euler**: Ist $\mathrm{ggT}(m, N) = 1$, dann ist $m^{\varphi(N)} = 1 \mod N$.

Ein **Satz von Euler**: Ist $\mathrm{ggT}(m, N) = 1$ dann ist $m^{\varphi(N)} = 1 \mod N$.

Dabei ist $\varphi(N)$ die eulersche Phi-Funktion:

$$\varphi(n) := \left|\{a \in \mathbb{N} \mid 1 \leq a \leq n \text{ und } \mathrm{ggT}(a, n) = 1\}\right|$$

$\varphi(n)$ zählt die zu n teilerfremden Zahlen in $1 \ldots n$.

Bsp.:

- $\varphi(6) = 2$ (denn ggT(1,6)=1, ggT(2,6)=2, ggT(3,6)=3, ggT(4,6)=2, ggT(5,6)=1, ggT(6,6)=6)
- $\varphi(12) = 4$: 1,5,7,11.
- $\varphi(7) = 6$: 1,2,3,4,5,6.
- $\varphi(11) = 10$: 1,2,3,4,5,6,7,8,9,10.
- $\varphi(p) = p - 1$, falls p Primzahl.

Und $\varphi(pq) = (p-1)(q-1)$, falls p und q Primzahlen.

Ein **Satz von Euler**: Ist $\mathrm{ggT}(m, N) = 1$ dann ist $m^{\varphi(N)} = 1 \mod N$.

Damit: $N = pq$, p, q Primzahlen, also $\varphi(N) = (p-1)(q-1)$. Da $ed = 1 \mod \varphi(N)$ ist $ed = k\varphi(N) + 1$ für ein $k \in \mathbb{N}$.

$$(m^e)^d = m^{ed} = m^{1+k\varphi(N)} = m \cdot (m^{\varphi(N)})^k = m \cdot 1^k = m \mod N.$$

Obacht: Beweis klappt nur für $\mathrm{ggT}(m, N) = 1$. Ist aber auch in den (wenigen!) anderen Fällen wahr. In Lehrbüchern Beweis daher oft mittels des kleinen Satzes von Fermat: $m^{p-1} = 1 \mod p$ für p Primzahl, und dem chinesischen Restsatz.

Beispiel:

- Alice wählt $p = 17$ und $q = 11$ (geheim).
- Alice berechnet $N = 17 \cdot 11 = 187$ und $F = (p-1)(q-1) = 160$.
- Alice wählt $e = 7$ ($\text{ggT}(e, F) = 1$, also OK), und berechnet d mit $7 \cdot d = 1 \mod 160$, also $d = 23$ (! s.u.)
- Alice gibt $N = 187$ und $e = 7$ öffentlich bekannt.
- Bob kann nun eine Botschaft, z.B. X, in eine Zahl m verwandeln (z.B. In ASCII: $X = 88$) und verschlüsselt $88^7 \mod 187$.
  - $88^7 = 88 \cdot 88^2 \cdot 88^4$
  - $88^2 = 7744 = 77 \mod 187$
  - $88^4 = 77^2 = 5929 = 132 \mod 187$
  - Also $88^7 = 88 \cdot 77 \cdot 132 = 894432 = 11 \mod 187$
- Alice empfängt 11 und berechnet $11^{23} = 11 \cdot 11^2 \cdot 11^4 \cdot 11^{16} = 11 \cdot 121 \cdot 55 \cdot 154 = 88 \mod 187$.

$7 \cdot d = 1 \mod 160$: mit dem erweiterten euklidischen Algorithmus.

Erweiterter Euklidischer Algorithmus: Berechnet zu $a, b \in \mathbb{N}$ den $\mathrm{ggT}(a, b)$ sowie $k, m \in \mathbb{Z}$ mit $ka + mb = \mathrm{ggT}(a, b)$.

Für RSA brauchen wir den eukl. Alg. zweimal:

- $F = (p-1)(q-1)$ ist bereits gewählt. Gesucht e mit $\mathrm{ggT}(e, F) = 1$: Wähle e zufällig, checke mit einfachem euklid. Alg. ob $\mathrm{ggT}(e, F) = 1$. Wenn nicht, wähle anderes e, solange bis es klappt.
- F, e nun bekannt. Gesucht d mit $d \cdot e = 1 \mod F$: erweiterter euklidischer Algorithmus liefert $k, \ell \in \mathbb{Z}$ mit: $ke + \ell F = \mathrm{ggT}(e, F) = 1$, also $ke = 1 + (-\ell)F$, also $ke = 1 \mod F$. Also $d := k$.

Erweiterter Euklidischer Algorithmus:

Seien $e, F \in \mathbb{N}$ gegeben ($e < F$).

1. $a_1 := F,\ a_2 := e,\ n := 2.$
   $c_1 := 1,\ c_2 := 0,\ d_1 := 0,\ d_2 := 1.$
2. $q_n := \max\{r \in \mathbb{N} \mid a_{n-1} - r a_n \geq 0\};\ a_{n+1} := a_{n-1} - q_n a_n.$
   $c_{n+1} := c_{n-1} - q_n c_n,\ d_{n+1} := d_{n-1} - q_n d_n.$
3. Falls $a_{n+1} = 0$ STOP, Ausgabe ggT$=a_n$, sowie c_n, d_n.

Dann ist $c_n F + d_n e = \text{ggT}(e, F)$.

Bsp.: (wie oben $e = 7, F = 160$) Gesucht k mit $k7 + \ell 160 = 1$.

n	a_n	q_n	c_n	d_n
1	160	\	1	0
2	7	22	0	1
3	6	1	1	-22
4	1	6	-1	23
5	0			

Also $23 \cdot 7 + (-1) \cdot 160 = 1$. Also $d = 23$ effizient berechenbar.

One more thing.... Wie kann Alice zwei große Zahlen wählen und sie als Primzahlen erkennen? Ohne sie zu faktorisieren?

Dazu nur kurz: Es gibt **probabilistische Primzahltests** (Solovay-Strassen, Miller-Rabin). Diese erkennen eine Nicht-Primzahl n mit einer Wahrscheinlichkeit von mehr als $\frac{1}{2}$ pro Durchlauf. Jedem Durchlauf liegt eine andere Zufallszahl (unabhängig, gleichverteilt) zwischen 1 und n zu Grunde.

Ist n nach einem Durchlauf als Nicht-Primzahl erkannt: Ausgabe "n keine Primzahl", STOP. (Wahrsch. $> \frac{1}{2}$)
Falls nicht: (also "n ist vielleicht Primzahl") weiter.

Nach k Durchläufen:

- Ist n keine Primzahl, dann wird das mit Wahrsch. $> 1 - (\frac{1}{2})^k$ erkannt.
- Ist n *nicht* als Nicht-Primzahl erkannt, dann ist die Wahrsch., dass n *doch* Nicht-Primzahl ist, kleiner als $(\frac{1}{2})^k$.

Dieser Algorithmus:

1. Erzeuge Zufallszahl n
2. Teste k mal mit probabil. Primzahltest, on n Primzahl ist
3. Falls "Ja": Ausgabe n
4. Falls "Nein" weiter bei 1.

liefert für $k = 30$ eine Zahl, die mit Wahrscheinlichkeit 0,999999999 eine Primzahl ist. Mit $k = 100$ ist das für unsere Zwecke ausreichend.

Es gibt auch *deterministische* Primzahltests (ohne Zufall), sogar effiziente:

Agrawal-Kayal-Saxena: PRIME is in P, *Annals of Mathematics* (2002).

Soviel zu RSA-Verschlüsselung. Nun zu Problem 3: fälschungssichere Unterschrift.

RSA Signature

Grobe Idee: Alice verschlüsselt ihre Nachricht mit ihrem privaten Schlüssel d.

Bob entschlüsselt mit Alices öffentlichem Schlüssel e. Falls Text sinnvoll: Nur Alice kann die Nachricht verfasst haben.

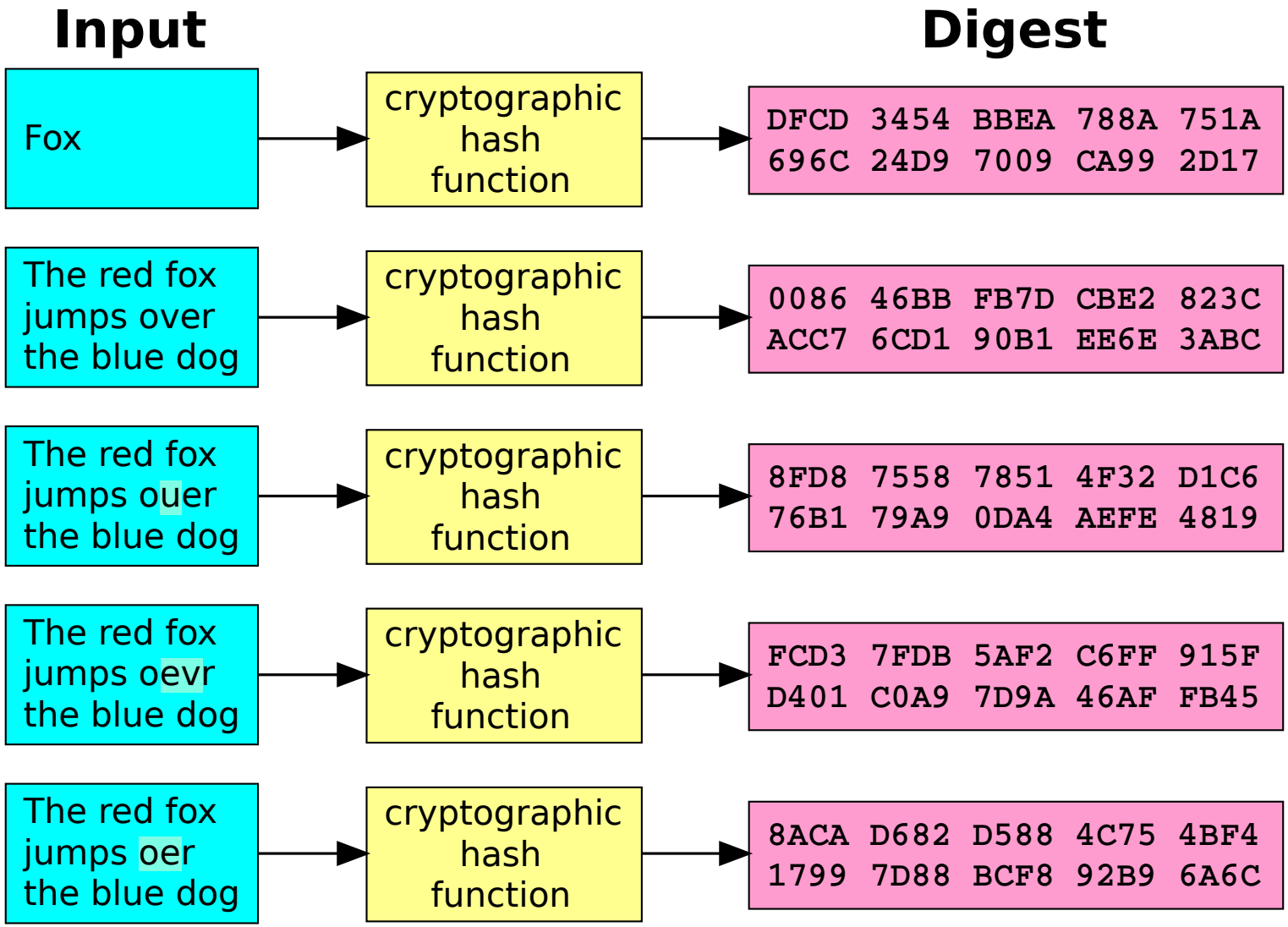
Offensichtlicher **Nachteil**: Jeder kann Alices Nachricht an Bob entschlüsseln. Also Zwischenschritt:

Kryptographische **Hash-Funktion**. Eigenschaften:

- Aus jeder Eingabe ist der Hash-Wert einfach zu berechnen.
- Aus Hash-Wert Nachricht rekonstruieren: Nicht machbar.
- Nachricht ändern, ohne Hash-Wert zu ändern: Nicht machbar.
- Zwei Nachrichten finden, die denselben Hash-Wert haben: Nicht machbar.

Die Hashfunktion h ist allgemein bekannt.

Kurz: sowas gibt's. (google SHA-1)



Also insgesamt so (Verschlüsseln und signieren):

- Alice schreibt Nachricht m. Berechnet Hashwert $h(m)$
- Alice sendet verschlüsselte Nachricht $m^{\overline{e}}$ (mit Bobs public key $\overline{e}$), sowie verschlüsselten Hashwert $(h(m)^d)$ (mit ihrem private key d)
- Bob entschlüsselt: $(m^{\overline{e}})^{\overline{d}} = m$ (mit seinem private key $\overline{d}$), sowie $(h(m)^d)^e$ (mit Alices public key e).
- Bob berechnet auch $h(m)$. Stimmen beide überein, dann kann nur Alice $h(m)^d$ gesendet haben.

Nebenbei: Das gleiche Verfahren (RSA Verschlüsselung und -Signatur) wurde schon 1973 (also 4 Jahre früher) vom englischen Geheimdienst NCHQ entwickelt. Das wurde erst 1997 bekannt.

PGP

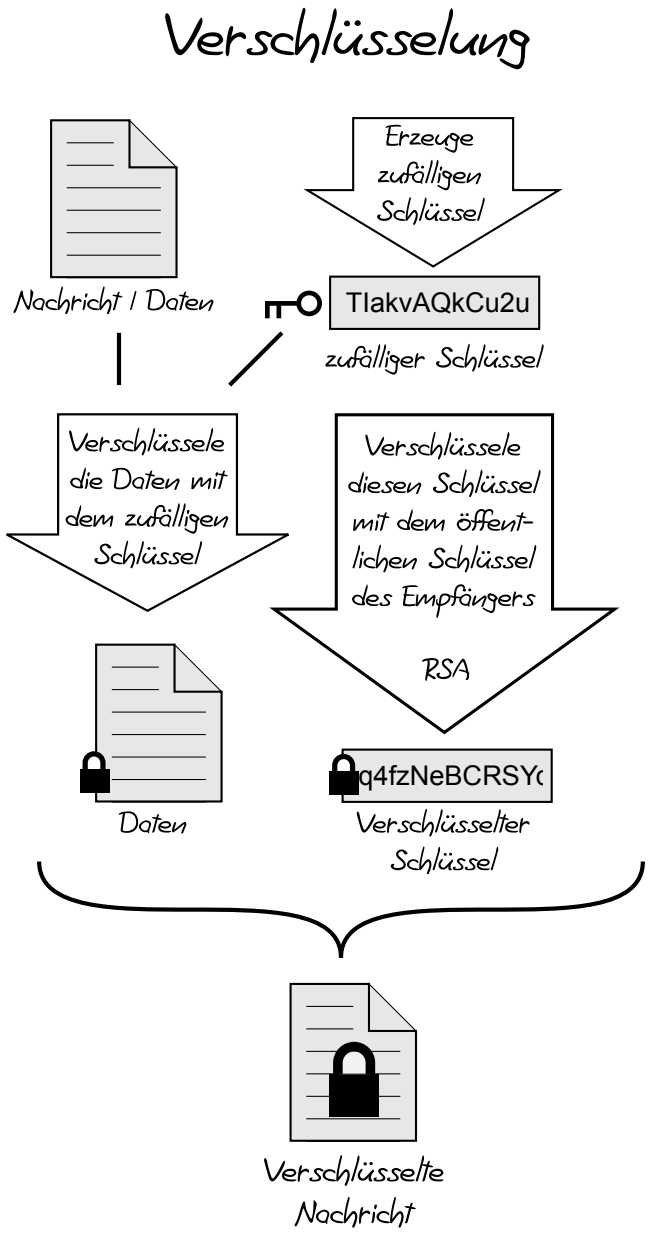
Wie funktioniert RSA in der echten Welt: Z.B. **PGP** (Pretty Good Privacy)

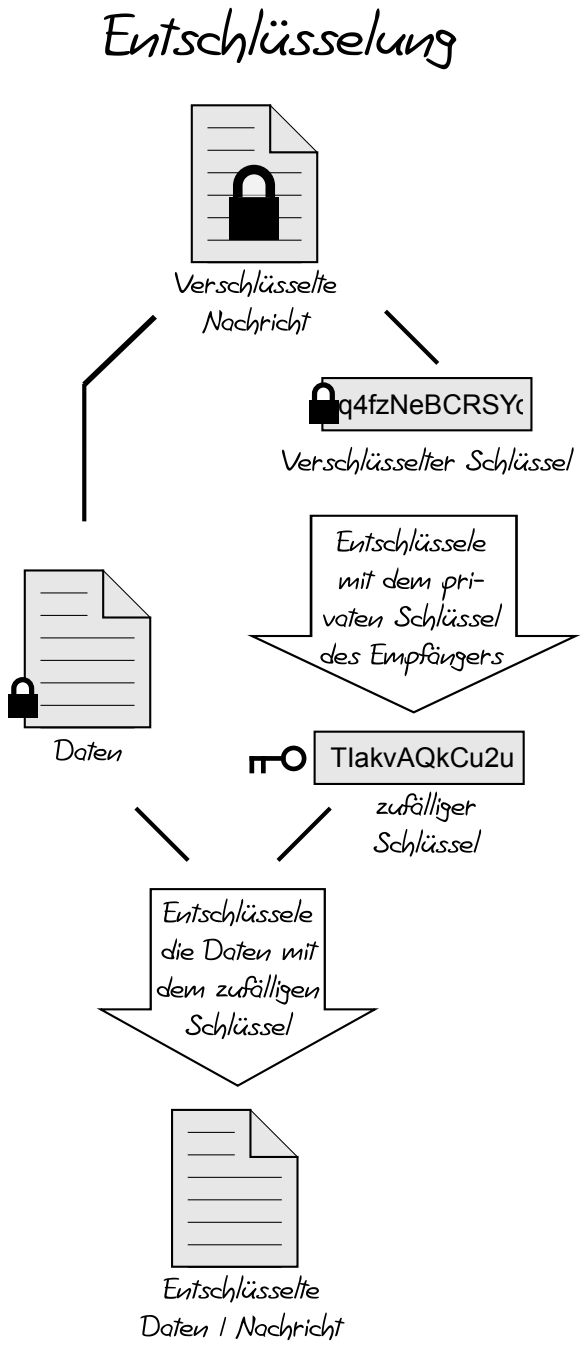
Das ist in der Praxis weit verbreitet (aber siehe auch TLS bzw SSL: verschlüsselte Internetverbindungen, nutzt auch Public-Key-Verfahren)

Eine Implementierung der verschiedenen Algorithmen, um das ganze jedem nutzbar zu machen.

Kombiniert Verschlüsselung, Signatur, Schlüsselverwaltung (!)

(Eine) **Idee**: Für jede Nachricht wird ein eigener Schlüssel erzeugt. (OK wäre ein one-time-pad; reales Verfahren: IDEA). Nur dieser wird mit RSA verschlüsselt.


Verschlüsselung
Nachricht / Daten
Erzeuge zufälligen Schlüssel
TIakvAQkCu2u
zufälliger Schlüssel
Verschlüssele die Daten mit dem zufälligen Schlüssel
Verschlüssele diesen Schlüssel mit dem öffentlichen Schlüssel des Empfängers
RSA
Daten
q4fzNeBCRSYc
Verschlüsselter Schlüssel
Verschlüsselte Nachricht


Entschlüsselung
Verschlüsselte Nachricht
q4fzNeBCRSYc
Verschlüsselter Schlüssel
Entschlüssele mit dem privaten Schlüssel des Empfängers
Daten
TIakvAQkCu2u
zufälliger Schlüssel
Entschlüssele die Daten mit dem zufälligen Schlüssel
Entschlüsselte Daten / Nachricht

1991 wurde PGP von Phil Zimmermann herausgebracht.
Verbreitete sich im Netz (das damals winzig war) sehr schnell.

Er bekam rechtliche Probleme:

- Weil RSA mittlerweile unter einem Patent stand,
- Weil es in den USA ein Gesetz gab, das Kryptographie-Software als Rüstungsgut klassifiziert: Er hatte sich des illegalen Rüstungsexports schuldig gemacht.

Ermittlungen, FBI, ... es kam aber nicht zum Prozess:
Zimmermann veröffentlichte den Quellcode von PGP in einem Buch (P. Zimmermann: *PGP Source Code and Internals*, MIT Press 1995).

Bücher galten nicht als Rüstungsgüter.

[Demo GPG]