

# Kryptologie mit CrypTool

**Version 1.4.10**

**Einführung in Kryptographie und Kryptoanalyse  
Umfang, Technik und Zukunft von CrypTool**



[www.cryptool.com](http://www.cryptool.com)  
[www.cryptool.de](http://www.cryptool.de)  
[www.cryptool.org](http://www.cryptool.org)  
[www.cryptool.pl](http://www.cryptool.pl)

# Übersicht (I)

## I. CrypTool und Kryptographie – Überblick

1. [Das CrypTool-Projekt](#)
2. [Bedeutung der Kryptographie und Beispiele klassischer Verfahren](#)
3. [Erkenntnisse aus der Entwicklung der Kryptographie](#)

## II. Was bietet CrypTool?

1. [Überblick](#)
2. [Beispiele zur Interaktion](#)
3. [Herausforderungen für Entwickler](#)

## III. Ausgewählte Beispiele

1. [RSA-Verschlüsselung / SSL](#)
2. [Elektronische Signatur visualisiert](#)
3. [Angriff auf RSA-Verschlüsselung](#)
4. [Psion-Analyse](#)
5. [Schwache DES-Schlüssel](#)
6. [Auffinden von NSA-Keys](#)
7. [Finden von Hashkollisionen](#)
8. [Authentifizierungsarten](#)
9. [Seitenkanalangriffs-Demo](#) (...)

# Übersicht (II)

## III. Ausgewählte Beispiele

- 10. [Angriffe auf RSA per Gitterreduktion](#)
- 11. [Zufallsanalyse mit 3-D Visualisierung](#)
- 12. [Secret Sharing als Anwendung des Chinesischen Restsatzverfahrens und nach Shamir](#)
- 13. [Anwendung des CRT in der Astronomie](#)
- 14. [Visualisierung von Verfahren mit ANIMAL](#)
- 15. [Visualisierung von AES](#)
- 16. [Visualisierung der Enigma-Verschlüsselung](#)
- 17. [Erzeugung eines MAC](#)
- 18. [Hash-Demo](#)
- 19. [Lernprogramm zu Zahlentheorie und asymmetrischer Verschlüsselung](#)
- 20. [Punktaddition auf elliptischen Kurven](#)
- 21. [Passwort-Qualitätsmesser](#)
- 22. [Brute-Force-Analyse](#)
- 23. [CrypTool Online-Hilfe](#)

## IV. Projekt / Ausblick / Kontakt



# Inhalt

I. **CrypTool und Kryptographie – Überblick**

II. Was bietet CrypTool?

III. Ausgewählte Beispiele

IV. Projekt / Ausblick / Kontakt

# Das CryptTool-Projekt

- Ursprung im Awareness-Programm einer Großbank (betriebliche Ausbildung)  
→ **Sensibilisierung der Mitarbeiter**
- Entwickelt in Kooperation mit Hochschulen (Verbesserung der Lehre)  
→ **Mediendidaktischer Anspruch**

1998 – **Projektstart** – Aufwand bisher mehr als 17 Mannjahre

2000 – CryptTool als **Freeware** verfügbar

2002 – CryptTool auf der **Bürger-CD des BSI** „Ins Internet – mit Sicherheit“

2003 – CryptTool wird **Open-Source** – Hosting durch die Uni Darmstadt (Fr. Prof. Eckert)

2004 – Auszeichnungen:

TeleTrust (TTT Förderpreis 2004)



NRW (IT-Sicherheitspreis NRW)



Ministerium für Innovation,  
Wissenschaft, Forschung  
und Technologie des Landes  
Nordrhein-Westfalen

**NRW.**

RSA Europe (Finalist beim European Information Security Award 2004)



2007 – CryptTool in deutsch, englisch und polnisch

## ■ **Entwickler**

- Entwickelt von Mitarbeitern verschiedener Firmen und Universitäten
- Weitere Projekt-Mitarbeiter oder verwertbare vorhandene Sourcen sind immer herzlich willkommen (z.Zt. arbeiten ca. 30 Leute weltweit mit).

# Bedeutung der Kryptographie

## Typischer Einsatz von Kryptographie im Alltag

### Einsatzbeispiele für Kryptographie

- Telefonkarten, Handys, Fernbedienungen
- Geldautomaten, Geldverkehr zwischen Banken
- Electronic cash, Online-Banking, Sichere E-Mail
- Satellitenfernsehen, PayTV
- Wegfahrsperre im Auto
- Digital Rights Management (DRM)



- Kryptographie ist schon lange nicht mehr nur auf Agenten, Diplomaten und Militärs begrenzt. Kryptographie ist eine moderne, mathematisch geprägte Wissenschaft.
- Der Durchbruch für den breiten Einsatz kam mit dem Internet.
- Für Firmen und Staaten ist es wichtig, dass sowohl die Anwendungen sicher sind, als auch, dass ...

*... die Nutzer (Kunden, Mitarbeiter) ein Mindestverständnis und Bewusstsein (Awareness) für IT-Sicherheit besitzen !*

# Kryptographie – Was will man damit erreichen?

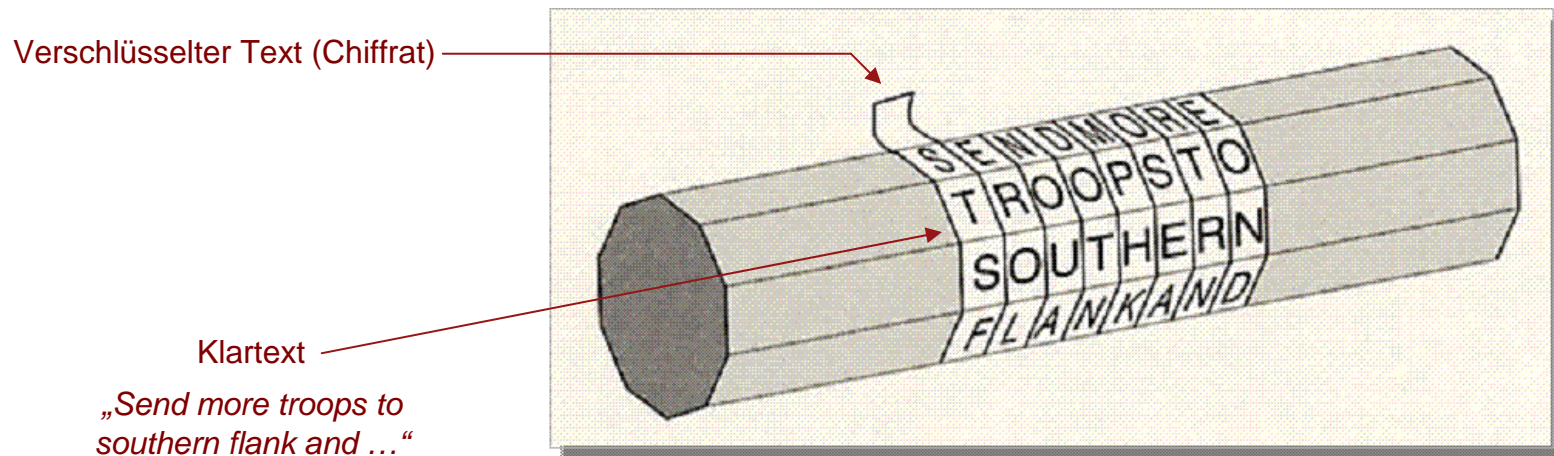
## Sicherheitsziele im Kontext von Kryptographie

- **Vertraulichkeit** (*Confidentiality*)
  - Lesen des eigentlichen Inhalts für Unbefugte „praktisch“ unmöglich machen
- **Authentifizierung** (*Authentication*)
  - Identitätsbeweis des Senders einer Nachricht gegenüber dem Empfänger
- **Integrität** (*Integrity*)
  - Eigenschaft, dass die Nachricht nicht verändert wurde
- **Verbindlichkeit** (*Non-Repudiation*)
  - Der Empfänger kann den Nachweis erbringen, dass der Sender die Nachricht mit identischem Inhalt abgeschickt hat (Leugnen zwecklos)

# Beispiele aus der klassischen Kryptographie (I)

## Älteste bekannte Verschlüsselungsverfahren

- **Kahlgeschorener Sklave**
- **Atbash** (um 600 v. Chr.)
  - Hebräische Geheimschrift, umgedrehtes Alphabet
- **Skytale von Sparta** (etwa 500 v. Chr.)
  - Beschrieben vom griechischen Historiker/Schriftsteller Plutarch (45 - 125 n. Chr.)
  - Zwei Zylinder (Holzstäbe) mit gleichem Durchmesser
  - Transposition (Zeichen des Klartextes werden umsortiert)





# Beispiele aus der klassischen Kryptographie (II)

## Caesar-Verschlüsselung (mono-alphabetische Substitution)

- **Caesar-Verschlüsselung** (Julius Cäsar, 100 - 44 v.Chr.)
- Einfache Substitutionschiffre

**G**ALLIA EST OMNIS DIVISA ...

Klartextalphabet:    **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

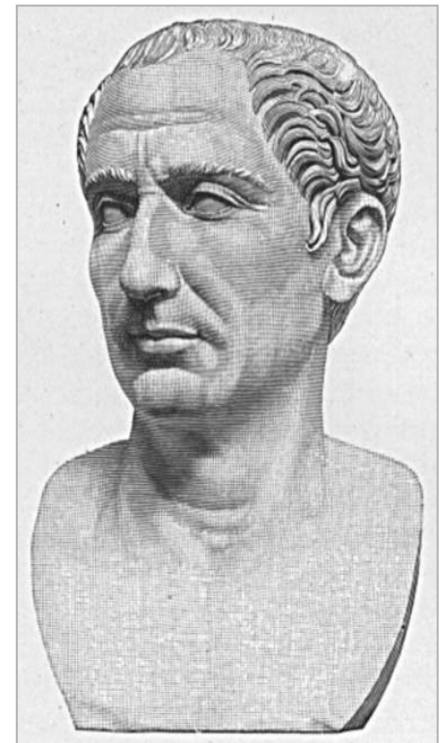
Geheimtextalphabet: **D** E F G H I **J** K L M N O P Q R S T U V W X Y Z A B C

**J** D O O L D    H V W    R P Q L V    G L Y L V D    ...

- **Angriff:** Häufigkeitsanalyse (typische Verteilung von Zeichen)

Vorführung mit CrypTool über folgende Menüs:

- **Als Animation:** „Einzelverfahren“ \ „Visualisierung von Algorithmen“ \ „Caesar ...“
- **Als Anwendung:** „Ver-/Entschlüsseln“ \ „Symmetrisch (Klassisch)“ \ „Caesar / Rot-13“



# Beispiele aus der klassischen Kryptographie (II)

## Vigenère-Verschlüsselung (poly-alphabetische Substitution)

- **Vigenère-Verschlüsselung** (Blaise de Vigenère, 1523-1596)
- Verschlüsselung mit einem Schlüsselwort unter Nutzung einer Schlüsseltabelle
- Schlüsselwort: **CHIFFRE**
- Verschlüsselung: **VIGENERE** wird zu **XPOJSVVG**
- Das Klartextzeichen wird ersetzt durch das Zeichen in der Zeile des Klartextes (bsp. V) und in der Spalte des Schlüsselwortzeichens (bsp. C). Das nächste Zeichen (bsp. I) wird in der Spalte des zweiten Zeichens des Schlüsselwortes (bsp. H) abgelesen, usw.
- Sobald man beim letzten Zeichen des Schlüsselwortes angekommen ist, beginnt man wieder mit dem ersten Zeichen des Schlüsselwortes.
- **Angriff** (u. a. durch Kasiski-Test): Es können gleiche Klartextzeichenkombinationen mit jeweils der gleichen Geheimtextzeichenkombination auftreten. Der Abstand dieser Muster kann nun genutzt werden, um die Schlüsselwortlänge zu bestimmen. Eine anschließende Häufigkeitsanalyse kann dann den Schlüssel bestimmen.

Schlüsselwort

	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

Tableau carré, dit « Carré de Vigenère »

Klartextzeichen

Verschlüsseltes  
Zeichen

# Beispiele aus der klassischen Kryptographie (IV)

## Weitere Verfahren der klassischen Kryptographie

- **Homophone Substitution**
- **Playfair** (1854 von Sir Charles Wheatstone, 1802-1875)
  - veröffentlicht von Baron Lyon Playfair
  - Substitution eines Buchstabenpaares durch ein anderes anhand einer quadratischen Alphabetsanordnung
- **Übermittlung von Buchseiten**
  - Adaption des OTP\*
- **Lochschablonen** (Fleißner)
- **Permutationsverschlüsselung**
  - "Doppelwürfel"
  - Reine Transposition / sehr effektiv

**Schlüssel eingabe Playfair**

Optionen

- ☒ Text vorformatieren
- ☒ Doppelte Zeichen im Schlüssel ignorieren

Playfair-Schlüssel

Kurzform des Playfair-Schlüssels:

CHARLES

Schlüsselmatrix

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

☒ 5x5 Matrix  
☐ 6x6 Matrix

Verschlüsseln Entschlüsseln Abbrechen

\* OTP = One-time pad

# Kryptographie in der Neuzeit

## Entwicklung der Kryptographie in den letzten 100 Jahren bis 1970

### Klassische Verfahren

- ... werden teilweise heute noch eingesetzt.  
(nicht alles geht per Computer...)
- ... und deren Prinzipien **Transposition** und **Substitution** fanden Eingang beim Design moderner Algorithmen:  
Kombination der einfacheren Operationen (eine Art der Mehrfach-Verschlüsselung, cascades of ciphers), auf Bit-Ebene, Blockbildung, Runden.

### Verschlüsselungsverfahren werden

- ... weiter **verfeinert**,
- **mechanisiert** bzw. **computerisiert** und
- bleiben zunächst **symmetrisch**.





# Beispiel erste Hälfte 20. Jahrhundert

## Elektromechanische Verschlüsselungsmaschinen (Rotormaschinen)

- **Enigma Verschlüsselung** (Arthur Scherbius, 1878-1929)
- Mehr als 200.000 Maschinen kamen im 2. Weltkrieg zum Einsatz
- Der rotierende Walzensatz bewirkt, dass jedes Zeichen des Textes mit einer neuen Permutation verschlüsselt wird.
- Gebrochen durch massiven Einsatz von Kryptographie-Experten (etwa 7.000 Personen in UK) mit ersten Entschlüsselungsmaschinen sowie erbeuteten Original-Maschinen und dem Abfangen von täglichen Statusmeldungen (z.B. Wetternachrichten).
- **Konsequenzen der erfolgreichen Kryptoanalyse:**  
*„Allgemein wird die Kompromittierung des ENIGMA-Codes als einer der strategischen Vorteile angesehen, der maßgeblich zum Gewinn des Krieges durch die Alliierten geführt hat. Es gibt Historiker, die vermuten, dass der Bruch der ENIGMA den Krieg um etliche Monate, vielleicht sogar um ein volles Jahr, verkürzt hat.“*

([http://de.wikipedia.org/wiki/Enigma\\_%28Maschine%29](http://de.wikipedia.org/wiki/Enigma_%28Maschine%29) vom 06.03.2006)



# Kryptographie – Entscheidende Erkenntnisse (I)

- **Kerckhoffs-Prinzip** (formuliert 1883)

- Trennung von Algorithmus (Verfahren) und Schlüssel

**Algorithmus:** “Verschiebe Alphabet um eine best. Anzahl Positionen zyklisch nach links”

**Schlüssel:** die “bestimmte Anzahl Positionen” (bei Caesar: 3)

- Kerckhoffs-Prinzip: Geheimnis liegt im Schlüssel und nicht im Algorithmus bzw. „No security through obscurity“

- **One Time Pad – Shannon / Vernam**

- Nachweislich theoretisch sicher, jedoch praktisch kaum anwendbar (nur Rotes Telefon).

- **Shannons Konzepte: Konfusion und Diffusion**

- Zusammenhang zwischen M, C und K möglichst komplex (M=Message, C=Cipher, K=Key)
- Jedes Chiffrezeichen sollte von möglichst vielen Klartextzeichen und vom gesamten Schlüssel abhängen
- „Avalanche effect“ (kleine Änderung, große Wirkung)

- **Trapdoor Function** (Falltür, Einweg-Funktion, ...)

- in einer Richtung schnell, in die andere (ohne Wissen) langsam
- nur mit dem Geheimnis hat man Zugang zur Falltür



## Beispiel für die Verletzung des Kerckhoffs-Prinzips

Geheimnis sollte nur im Schlüssel und nicht im Algorithmus liegen

- **Handy-Verschlüsselung angeblich geknackt** (07.12.1999)

„Die beiden israelischen Kryptologen Alex Biryukov und Adi Shamir haben Medienberichten zufolge den Verschlüsselungsalgorithmus geknackt, der GSM-Handy-Telefonate auf der Funkstrecke zur Mobiltelefon-Basisstation schützt. Das Verfahren soll mit einem handelsüblichen PC auskommen, der mit 128 MByte RAM und zwei 73 GByte Festplatten ausgestattet ist. Auf diesem soll das Programm der Forscher durch eine Analyse der ersten zwei Gesprächsminuten in weniger als einer Sekunde den verwendeten Schlüssel errechnen können. Umstritten ist, ob und mit welchem Aufwand es möglich ist, die Gespräche überhaupt abzufangen, um sie anschließend zu dechiffrieren. Eines zeigen die Vorfälle um die GSM-Verschlüsselungsalgorithmen A5/1 und A5/2 aber schon jetzt deutlich: **Der Versuch, Krypto-Verfahren geheim zu halten, dient nicht der Sicherheit.** Das hat anscheinend auch die GSM-Association gelernt: Ihr Sicherheitsdirektor James Moran äußerte dem Online-Magazin Wired gegenüber, dass man künftige Algorithmen von vornherein offen legen will, um der Fachwelt eine Prüfung zu ermöglichen.“ [<http://www.heise.de/newsticker/meldung/7183>]

- **Weiteres Beispiel: Netscape Navigator** legte 1999 die Passworte für den Zugriff auf E-Mail-Server noch proprietär schwach verschlüsselt ab.

# Beispiel für eine One-Time-Pad-Adaption



Kleiderbügel einer Stasi-Spionin  
mit verstecktem One-Time-Pad  
(Aus: *Spiegel Spezial* 1/1990)



# Schlüsselverteilungsproblem

## Schlüsselverteilung bei symmetrischer Verschlüsselung

Wenn **2 Personen** miteinander mit einer symmetrischen Verschlüsselung kommunizieren, brauchen sie **einen gemeinsamen und geheimen Schlüssel**.

Wenn bei  $n$  Personen jeder mit jedem geheim kommunizieren möchte, dann braucht man  $S_n = n(n-1) / 2$  Schlüssel.

Das sind bei

$n = 100$  Personen bereits

$S_{100} = 4.950$  Schlüssel, bei

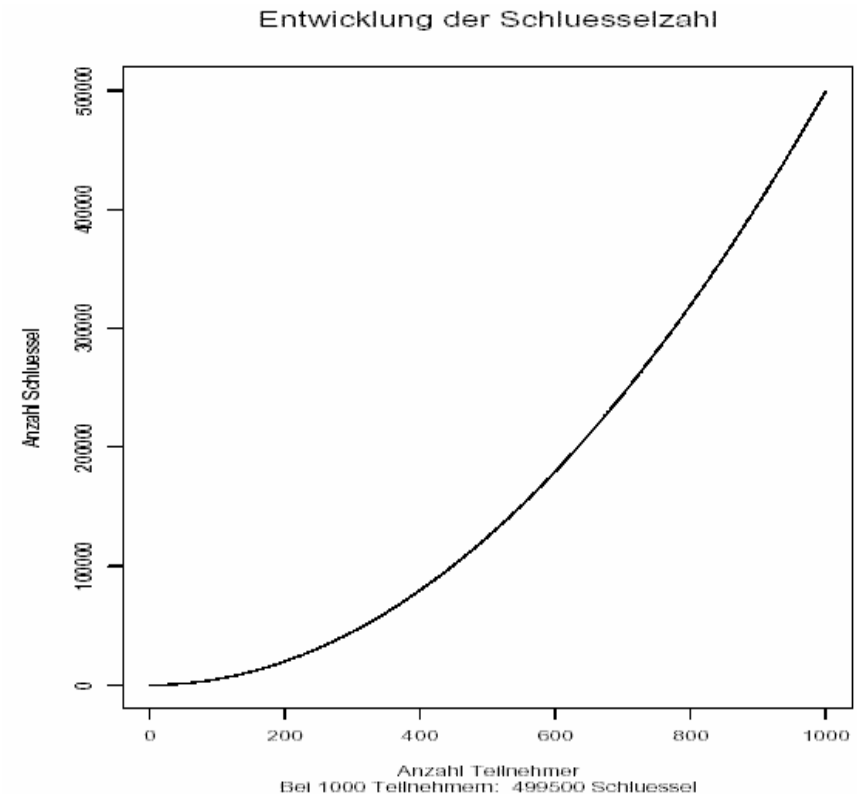
$n = 1.000$  Personen sind es

$S_{1000} = 499.500$  Schlüssel.

(Quadratischer Anstieg:

Faktor 10 mehr Personen,

Faktor 100 mehr Schlüssel)



# Kryptographie – Entscheidende Erkenntnisse (II)

## Lösung des Schlüsselverteilungsproblems durch asymmetrische Kryptographie

### ■ Asymmetrische Kryptographie

- **Jahrhunderte lang glaubte man:** Sender und Empfänger brauchen dasselbe Geheimnis.
- **Neu:** Jeder Teilnehmer hat ein Schlüsselpaar („Lösung“ des Schlüsselverteilungsproblems)

### ■ Asymmetrische Verschlüsselung

- „Jeder kann ein Vorhängeschloss einschnappen lassen oder einen Brief in einen Kasten werfen“
- MIT, 1977: Leonard Adleman, Ron Rivest, Adi Shamir (bekannt durch RSA)
- GCHQ Cheltenham, 1973: James Ellis, Clifford Cocks (am 18.12.1997 öffentlich zugegeben)

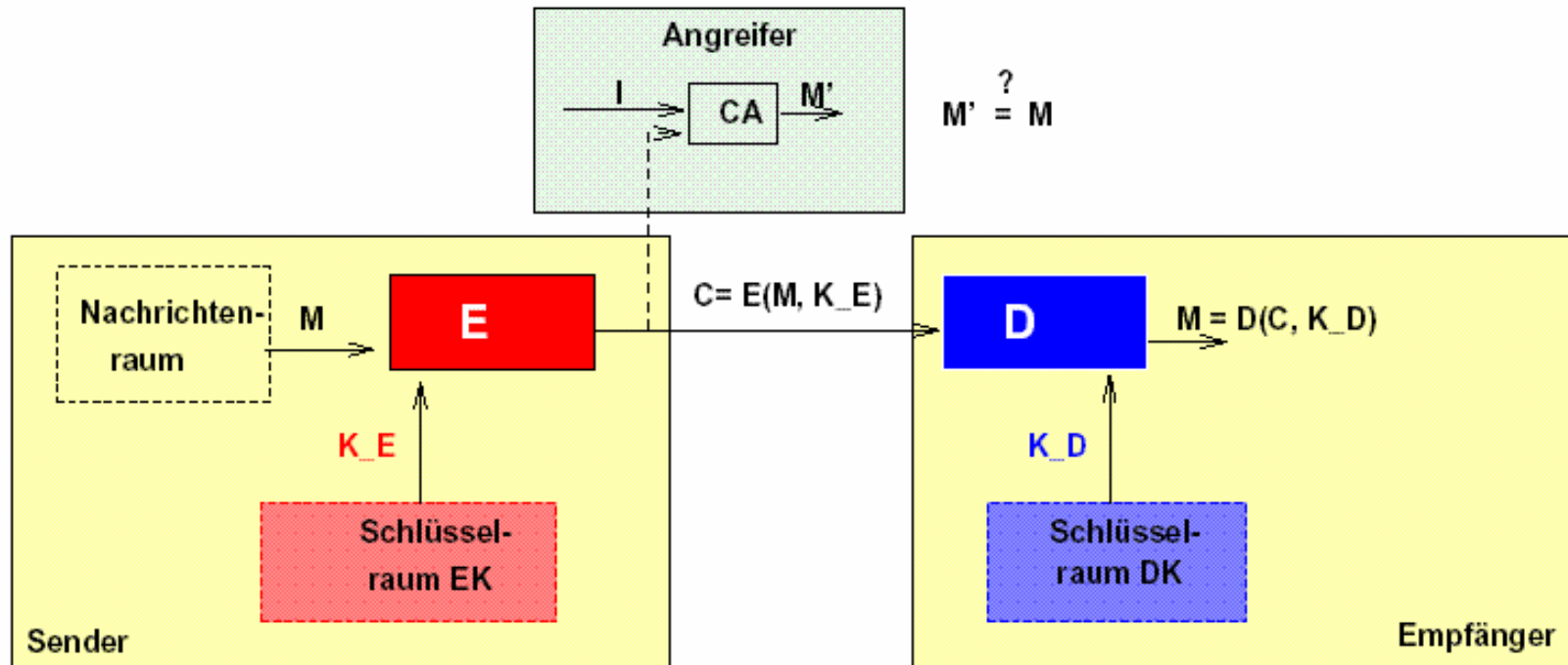
### ■ Schlüsselverteilung

- Stanford, 1976: Whitfield Diffie, Martin Hellman, Ralph Merkle (Diffie-Hellman Key Exchange)
- GCHQ Cheltenham, 1975: Malcolm Williamson

*Sicherheit in offenen Netzen (wie dem Internet) wäre  
ohne asymmetrische Kryptographie extrem teuer und komplex !*

# Durchführung von Ver- und Entschlüsselung

## Symmetrische und asymmetrische Verschlüsselung



(a) Symmetrische :  $K_E = K_D$  geheim! z.B. DES

(b) Asymmetrische :  $K_E \neq K_D$  z.B. RSA

öffentlich      geheim!

# Kryptographie – Entscheidende Erkenntnisse (III)

## Steigende Bedeutung der Mathematik und der Informationstechnologie

- **Moderne Kryptographie** basiert stärker auf **Mathematik**
  - Trotzdem gibt es weiter symmetrische Verfahren wie den AES (bessere Performance und kürzere Schlüssellängen als die auf rein mathematischen Problemstellungen beruhenden asymmetrischen Verfahren).
- Die Sicherheit praktisch eingesetzter Verfahren hängt entscheidend vom Stand der **Mathematik** und der **Informationstechnologie** (IT) ab.
  - Berechnungskomplexität (d.h. Rechenaufwand in Abhängigkeit von der Schlüssellänge, Speicherplatzbedarf, Datenkomplexität)
    - > siehe aktuell RSA: Bernstein, TWIRL-Device, RSA-200 (CrypTool-Skript, Kap. 4.11.3)
  - Sehr hohe Intensität in der aktuellen Forschung:
    - Faktorisierung, nicht-parallelisierbare Algorithmen (wegen Quantencomputing), besseres Verständnis von Protokoll-Schwächen und Zufallszahlengeneratoren, ...).
- Entscheidender Irrtum: „*Echte Mathematik*“ hat keine Auswirkungen auf den Krieg. (G.H. Hardy, 1940)
- Hersteller entdecken **Sicherheit** als ein zentrales **Kaufkriterium**

# Demo mit CrypTool

- *Statistische Analyse*

- *Zweimal nacheinander ist nicht immer besser:*

*Caesar:*  $C + D = G$  ( $3 + 4 = 7$ )

*Vigenère:* -  $CAT + DOG = FOZ$   $[(2,0,19)+(3,14,6)=(5,14,25)]$

-  $"Hund" + "Katze" = "RUGCLENWGYXDATRHNHNMH"$

- *Vernam (OTP)*

- *AES (Ausgabe-Key, Brute-Force-Analyse)*



# Inhalt

I. CrypTool und Kryptographie – Überblick

II. **Was bietet CrypTool?**

III. Ausgewählte Beispiele

IV. Projekt / Ausblick / Kontakt

# Was bietet CrypTool?

E-Learning

## 1. Was ist CrypTool?

- Freeware-Programm mit graphischer Oberfläche
- Kryptographische Verfahren anwenden *und* analysieren
- Sehr umfangreiche Online-Hilfe, ohne tieferes Kryptographiewissen verständlich
- Enthält fast alle State-of-the-art-Kryptographiefunktionen
- „Spielerischer“ Einstieg in moderne und klassische Kryptographie
- Kein „Hackertool“

## 2. Warum CrypTool?

- Ursprung im End-User Awareness-Programm einer Großbank
- Entwickelt in Kooperation mit Hochschulen → mediendidaktischer Anspruch
- Verbesserung der Lehre an Hochschulen und der betrieblichen Ausbildung

## 3. Zielgruppe

- Kernzielgruppe: Studierende der Informatik, Wirtschaftsinformatik, Mathematik
- Aber auch: Computernutzer und Anwendungsentwickler, Mitarbeiter, Schüler
- Voraussetzung: PC-Kenntnisse
- Wünschenswert: Interesse an Mathematik und Programmierung

# Inhalt des Programmpakets

Deutsch, Englisch  
und Polnisch

## CrypTool-Programm

- alle Funktionen integriert in *einem* Programm mit einheitlicher graphischer Oberfläche
- läuft unter Win32
- Kryptographie aus den Bibliotheken von Secude, cryptovision und OpenSSL
- Langzahlarithmetik per Miracl und GMP, Gitterbasenreduktion per NTL (V. Shoup)

## AES-Tool

- Standalone-Programm zur AES-Verschlüsselung (selbst extrahierend)

## Lernbeispiel

- „Der Zahlenhai“ fördert das Verständnis für Teiler und Primzahlen.

## Umfangreiche Online-Hilfe (HTML-Help)

- kontextsensitive Hilfe mit F1 für *alle* Programmfunktionen (auch auf Menüs)
- ausführliche Benutzungs-Szenarien (Tutorials) für viele Programmfunktionen

## Skript (.pdf-Datei) mit Hintergrundinformationen

- Verschlüsselungsverfahren • Primzahlen/Faktorisierung • Digitale Signatur
- Elliptische Kurven • Public Key-Zertifizierung • Elementare Zahlentheorie • Krypto 2020

## Zwei Kurzgeschichten mit Bezug zur Kryptographie von Dr. C. Elsner

- „Der Dialog der Schwestern“ (eine RSA-Variante als Schlüsselement)
- „Das chinesische Labyrinth“ (zahlentheoretische Aufgaben für Marco Polo)

## Authorware-Lernprogramm zur Zahlentheorie



# Funktionsumfang (I)

## Kryptographie

### Verschlüsselungsklassiker

- Caesar
- Vigenère
- Hill
- Homophone Substitution
- Playfair
- ADFGVX
- Addition
- XOR
- Vernam
- Permutation
- Solitaire

### Zum besseren Nachvollziehen von Literaturbeispielen ist

- Alphabet wählbar
- Behandlung von Leerzeichen etc. einstellbar

## Kryptoanalyse

### Angriffe auf klassische Verfahren

- Ciphertext-only
  - Caesar
  - Vigenère
  - Addition
  - XOR
  - Substitution
  - Playfair
- Known-plaintext
  - Hill
- Manuell (unterstützt)
  - Monoalphabetische Substitution
  - Playfair
  - ADFGVX
  - Solitaire

### Unterstützende Analyseverfahren

- Entropie, gleitende Häufigkeit
- Histogramm, n-Gramm-Analyse
- Autokorrelation
- Perioden
- Zufallszahlenanalyse
- Base64 / UU-Encode

# Funktionsumfang (II)

## Kryptographie

### Moderne symmetrische Verschlüsselung

- IDEA, RC2, RC4, RC6, DES, 3DES, DESX, DESL
- AES-Kandidaten der letzten Auswahlrunde (Serpent, Twofish, ...)
- AES (=Rijndael)
- DESL, DESXL

### Asymmetrische Verschlüsselung

- RSA mit X.509-Zertifikaten
- RSA-Demonstration
  - zum Nachvollziehen von Literaturbeispielen
  - Alphabet und Blocklänge einstellbar

### Hybridverschlüsselung (RSA + AES)

- visualisiert als interaktives Datenflussdiagramm

## Kryptoanalyse

### Brute-force-Angriff auf symmetrische Algorithmen

- für alle Algorithmen
- Annahme:
  - Entropie des Klartextes klein oder teilweise Kenntnis der Schlüssels oder Kenntniss des Klartextalphabets.

### Angriff auf RSA-Verschlüsselung

- Faktorisierung des RSA-Moduls
- Gitterreduktions-basierte Angriffe

### Angriff auf Hybridverschlüsselung

- Angriff auf RSA oder
- Angriff auf AES (Seitenkanalangriff)

# Funktionsumfang (III)

## Kryptographie

### Digitale Signatur

- RSA mit X.509-Zertifikaten
  - Signatur zusätzlich visualisiert als interaktives Datenflussdiagramm
- DSA mit X.509-Zertifikaten
- Elliptic Curve DSA, Nyberg-Rueppel

### Hashfunktionen

- MD2, MD4, MD5
- SHA, SHA-1, RIPEMD-160

### Zufallsgeneratoren

- Secude
- $x^2 \bmod n$
- Linearer Kongruenzgenerator (LCG)
- Inverser Kongruenzgenerator (ICG)

## Kryptoanalyse

### Angriff auf RSA-Signatur

- Faktorisierung des RSA-Moduls
- praktikabel bis ca. 250 bits bzw. 75 Dezimalstellen (auf Einzelplatz-PC)

### Angriff auf Hashfunktion / digitale Signatur

- Generieren von Hash-Kollisionen für ASCII-Texte (Geburtstagsparadox) (bis 40 bit in etwa 5 min)

### Analyse von Zufallsdaten

- FIPS-PUB-140-1 Test-Batterie
- Periode, Vitany, Entropie
- Gleitende Häufigkeit, Histogramm
- n-Gramm-Analyse, Autokorrelation
- ZIP-Kompressionstest

# Funktionsumfang (IV)

## Visualisierungen / Demos

- Caesar, Vigenère, Nihilist, DES mit Animal
- Enigma (Flash)
- Rijdael/AES (Flash)
- Hybride Ver- und Entschlüsselung (AES-RSA und AES-ECC)
- Erzeugung und Verifikation von Signaturen
- Diffie-Hellman-Schlüsselaustausch
- Secret Sharing (mit CRT oder mit dem Schwellenwertschema nach Shamir)
- Challenge-Response-Verfahren (Authentisierung im Netz)
- Seitenkanalangriff
- Grafische 3-D Darstellung von (Zufalls-)datenströmen
- Sensibilität von Hashfunktionen bzgl. Änderungen der Daten
- Zahlentheorie und RSA-Kryptosystem (Authorware)



# Funktionsumfang (V)

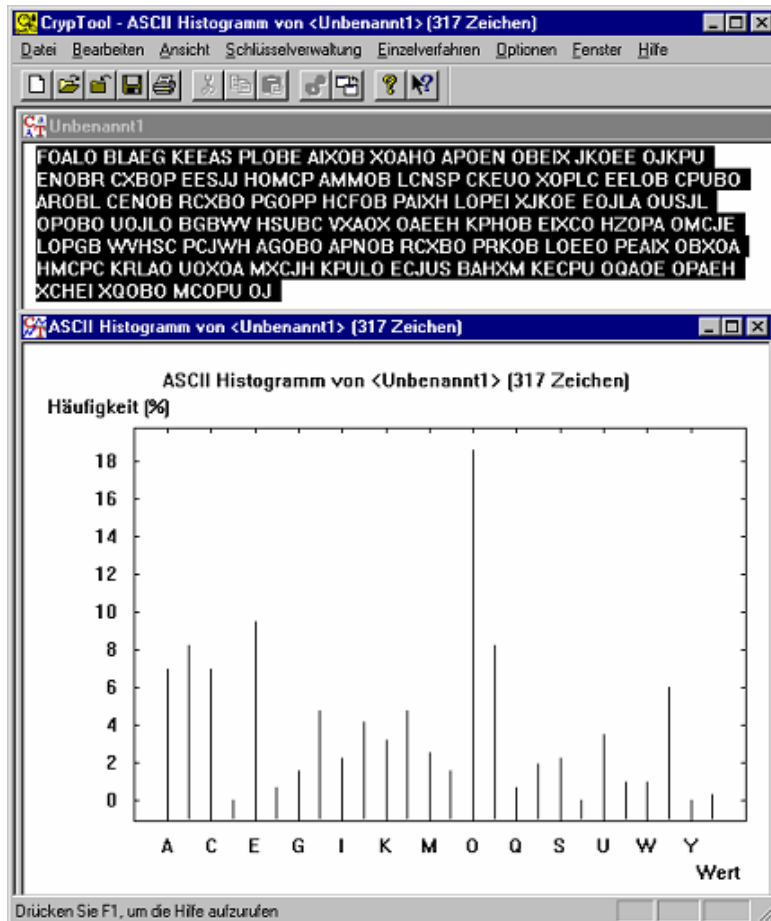
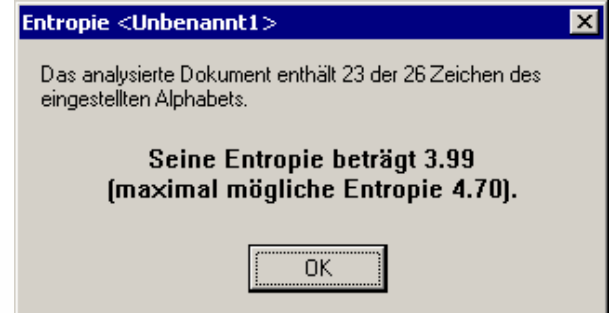
## Weitere Funktionen

- Homophone und Permutationsverschlüsselung (Doppelwürfel)
- PKCS #12-Import/Export für PSEs (Personal Security Environment)
- Hashwerte großer Dateien berechnen, ohne sie zu laden
- Generische Brute-Force-Attacke auf beliebige moderne symmetrische Algorithmen
- ECC-Demo (als Java-Applikation)
- und vieles mehr ...

# Sprachstruktur analysieren

## Anzahl Einzelzeichen, n-Gramme, Entropie

- z.B. mit CrypTool per Analyse / Allgemein / ...



N-Gramm Liste

Auswahl

☒ Histogramm  
☐ Digramm  
☐ Trigramm  
☐ 4 -Gramm

Anzeige der 26 häufigsten N-Gramme

Liste bestimmen

Liste speichern

Schließen

Nr.	Zeichen...	Häufigkeit in %
1	O	18.612
2	E	9.464
3	B	8.202
4	P	8.202
5	A	6.940
6	C	6.940
7	X	5.994
8	H	4.732
9	L	4.732
10	J	4.101
11	U	3.470
12	K	3.155
13	M	2.524
14	I	2.208
15	S	2.208
16	R	1.893
17	G	1.577
18	N	1.577
19	V	0.946
20	W	0.946
21	F	0.631
22	Q	0.631
23	Z	0.315

# Demonstration der Interaktivität (I)

## Vigenère-Analyse

*Demo per CrypTool*

**Das Ergebnis der Vigenère-Analyse kann manuell nachbearbeitet werden (gefundene Schlüssellänge ändern):**

1. Datei „Startbeispiel-de.txt“ mit **TEST** verschlüsseln

- „Ver-/Entschlüsseln“ \ „Symmetrisch (klassisch)“ \ „Vigenère...“
- Eingabe TEST  $\Rightarrow$  „Verschlüsseln“



Analyse der Verschlüsselung

- „Analyse“ / „Symmetrische Verschlüsselung (klassisch)“ \ „Ciphertext only“ \ „Vigenère“
- Schlüssellänge 4, Ermittelter Schlüssel TEST 

2. Datei „Startbeispiel-de.txt“ mit **TESTETE** verschlüsseln

- „Ver-/Entschlüsseln“ \ „Symmetrisch (klassisch)“ \ „Vigenère...“
- Eingabe TESTETE  $\Rightarrow$  „Verschlüsseln“

Analyse der Verschlüsselung

- „Analyse“ \ „Symmetrische Verschlüsselung (klassisch)“ \ „Ciphertext only“ \ „Vigenère“
- Schlüssellänge 14 – nicht korrekt 
- Schlüssellänge wird angepasst (automatisch – könnte aber auch manuell angepasst werden)
- Ermittelter Schlüssel TESTETE 

# Demonstration der Interaktivität (II)

## Automatisierte Primzahlzerlegung

*Demo per CrypTool*

### Primzahlzerlegung mit Hilfe von Faktorisierungsverfahren

- Menü „Einzelverfahren“ \ „RSA-Kryptosystem“ \ „Faktorisieren einer Zahl“
- Verschiedene Verfahren werden in mehreren Threads parallel ausgeführt
- Alle Verfahren haben bestimmte Vor- und Nachteile  
(z.B. erkennen bestimmte Verfahren nur kleine Faktoren)

### Faktorisierungs-Beispiel 1:

316775895367314538931177095642205088158145887517

48-stellige Dezimalzahl

=

3 \* 1129 \* 6353 \* 1159777 \* 22383173213963 \* 567102977853788110597

### Faktorisierungs-Beispiel 2:

$2^{250} - 1$

75-stellige Dezimalzahl

=

3 \* 11 \* 31 \* 251 \* 601 \* 1801 \* 4051 \* 229668251 \* 269089806001 \*  
4710883168879506001 \* 5519485418336288303251



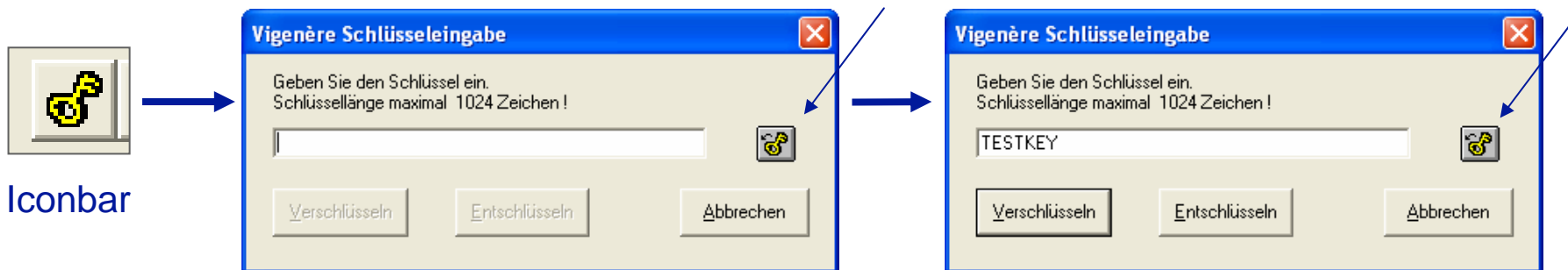
# Konzepte zur Benutzerfreundlichkeit

## 1. Kontextsensitive Hilfe (F1)

- F1 bei einem gewählten Menüeintrag zeigt Informationen zum Verfahren.
- F1 in einer Dialogbox erläutert die Bedienung des Dialogs.
- Diese Hilfen und die Inhalte des übergeordneten Menüs sind in der Online-Hilfe immer gegenseitig verlinkt.

## 2. Einfügen von Schlüsseln in die Schlüsseleingabe-Maske

- Mit Strg-V (Paste) kann man immer einfügen, was im Clipboard steht.
- Schon benutzte Schlüssel können aus Ciphertext-Fenstern per Icon in der Symbolleiste entnommen und durch ein komplementäres Icon in der Schlüsseleingabemaske in das Schlüsselfeld eingefügt werden. Dazu wird ein CrypTool-interner Speicher benutzt, der pro Verfahren zur Verfügung steht (nützlich bei „besonderen“ Schlüsseln wie der homophonen Verschlüsselung).



# Herausforderungen für den Programmierer (Beispiele)

## 1. Viele Funktionen parallel laufen lassen

- Bei der Faktorisierung laufen die verschiedenen Algorithmen in Threads

## 2. Hohe Performance

- Bei der Anwendung des Geburtstagsparadoxons zum Finden von Hashkollisionen oder bei der Brute-Force-Analyse

## 3. Speicherbeschränkung beachten

- Beim Floyd-Algorithmus (Mappings für das Finden von Hashkollisionen) oder beim Quadratic Sieve.

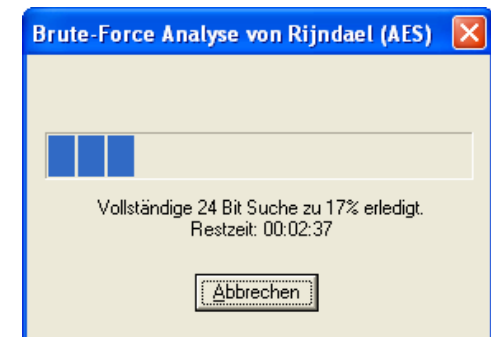
## 4. Zeitmessung und -abschätzung

- Ausgabe der Elapsed Time bei Brute-Force

## 5. Wiederverwendung / Integration

- Masken zur Primzahlgenerierung
- RSA-Kryptosystem (schaltet nach erfolgreicher Attacke von der Ansicht des Public-Key-Anwenders zur Ansicht des Private-Key-Besitzers)

## 6. Konsistenz der Funktionen, des GUI und der Online-Hilfe (inklusive verschiedener Sprachen)





# Inhalt

- I. CrypTool und Kryptographie – Überblick
- II. Was bietet CrypTool?
- III. **Ausgewählte Beispiele**
- IV. Projekt / Ausblick / Kontakt

# CrypTool-Anwendungsbeispiele

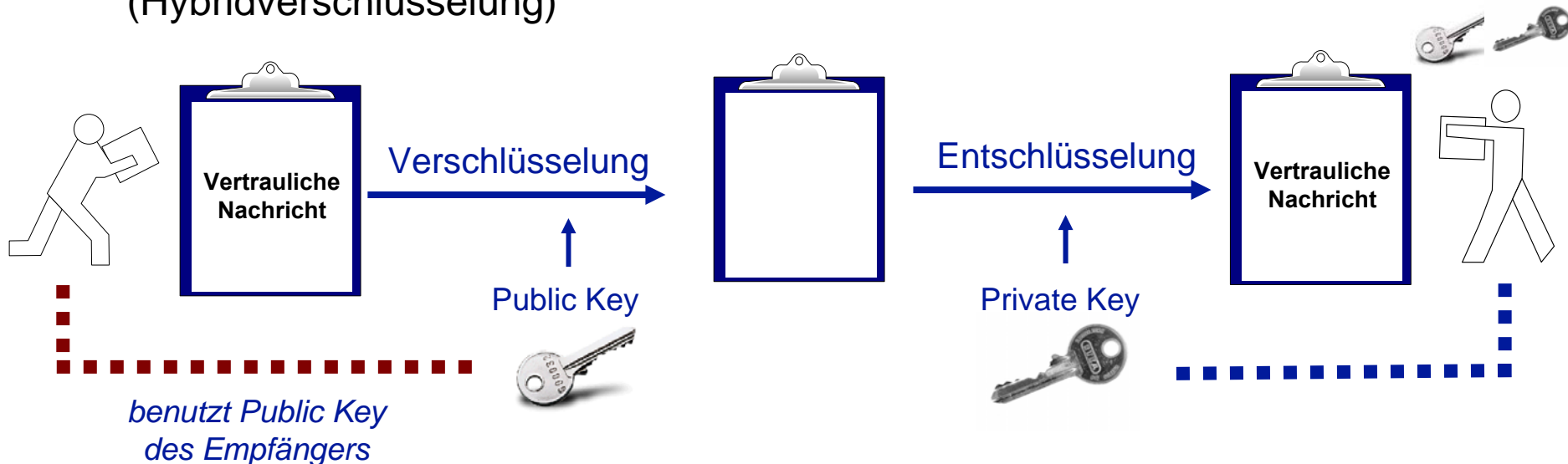
## Übersicht der Beispiele

1. [Verschlüsselung mit RSA / Primzahltest / Hybridverschlüsselung und Digitale Zertifikate / SSL](#)
2. [Elektronische Signatur visualisiert](#)
3. [Angriff auf RSA-Verschlüsselung \(Modul N zu kurz\)](#)
4. [Analyse der Verschlüsselung im PSION 5](#)
5. [Schwache DES-Schlüssel](#)
6. [Auffinden von Schlüsselmaterial \(„NSA-Key“\)](#)
7. [Angriff auf Digitale Signatur durch Finden von Hashkollisionen](#)
8. [Authentisierung in einer Client-Server-Umgebung](#)
9. [Demonstration eines Seitenkanalangriffs \(auf Hybridverschlüsselungsprotokoll\)](#)
10. [Angriffe auf RSA mittels Gitterreduktion](#)
11. [Zufallsanalyse mit 3-D Visualisierung](#)
12. [Secret Sharing als Anwendung des Chinesischen Restsatzverfahrens \(CRT\) und nach Shamir](#)
13. [Anwendung des CRT in der Astronomie](#)
14. [Visualisierung von symmetrischen Verschlüsselungsverfahren mit ANIMAL](#)
15. [Visualisierung von AES](#)
16. [Visualisierung der Enigma-Verschlüsselung](#)
17. [Erzeugung eines Message Authentication Code \(MAC\)](#)
18. [Hash-Demo](#)
19. [Lernprogramm zu Zahlentheorie und asymmetrischer Verschlüsselung](#)
20. [Punktaddition auf elliptischen Kurven](#)
21. [Passwort-Qualitätsmesser](#)
22. [Brute-Force-Analyse](#)
23. [CrypTool Online-Hilfe](#)

# Anwendungsbeispiele (1)

## Verschlüsselung mit RSA

- **Grundlage** für z.B. SSL-Protokoll (Zugriff auf gesicherte Web Seiten)
- **Asymmetrische Verschlüsselung mit RSA**
  - Jeder Benutzer hat ein Schlüsselpaar – einen öffentlichen und einen privaten.
  - Sender verschlüsselt mit dem öffentlichen Schlüssel (*public key*) des Empfängers.
  - Empfänger entschlüsselt mit seinem privaten Schlüssel (*private key*).
- Einsatz i. d. R. in Kombination mit symmetrischen Verfahren (Hybridverschlüsselung)



# Anwendungsbeispiele (1)

## Verschlüsselung mit RSA – Mathematischer Hintergrund / Verfahren

- Öffentlicher Schlüssel (public key):  $(n, e)$
- Privater Schlüssel (private key):  $(d)$

**wobei:**

$p, q$  große zufällig gewählte Primzahlen mit  $n = p \cdot q$ ;

$d$  wird unter den NB  $\text{ggT}[\varphi(n), e] = 1$ ;  $e \cdot d \equiv 1 \pmod{\varphi(n)}$ ; bestimmt.

Ver- und Entschlüsselungs-Operation:  $(m^e)^d \equiv m \pmod{n}$

- $n$  ist der Modulus, dessen Schlüssellänge beim RSA-Verfahren angegeben wird.
- $\text{ggT}$  = größter gemeinsamer Teiler.
- $\varphi(n)$  ist die Eulersche Phi-Funktion.

**Vorgehen:**

- Transformation von Nachrichten in binäre Repräsentation
- Nachricht  $m = m_1, \dots, m_k$  blockweise verschlüsseln, wobei für alle  $m_j$  gilt:  
 $0 \leq m_j < n$ ; also maximale Blockgröße  $r$  so, dass gilt:  $2^r \leq n$  ( $2^{r-1} < n$ )

# Anwendungsbeispiele (1)

Primzahltests – Für RSA werden große Primzahlen benötigt.

- Schnelle probabilistische Tests
- Deterministische Tests

Die bekannten Primzahltest-Verfahren können für große Zahlen viel schneller testen, ob die Zahl prim ist, als die bekannten Faktorisierungsverfahren eine Zahl ähnlicher Größenordnung in ihre Primfaktoren zerlegen können.

Für die AKS-Methode wurde die GMP-Bibliothek (**GNU Multiple Precision Arithmetic Library**) in Cryptool integriert.

**Primzahltest**

Um zu testen, ob eine Zahl eine Primzahl ist, gibt es verschiedene Verfahren (Mathematiker sagen auch, man testet, ob die Zahl prim ist).  
Am häufigsten werden die probabilistischen Verfahren verwendet: Sie sind sehr schnell, geben aber nur mit einer bestimmten (einstellbar kleinen) Wahrscheinlichkeit an, ob die Zahl prim ist.  
Daneben gibt es noch deterministische Verfahren: Wenn diese ein Ergebnis liefern, ist es mit 100% Wahrscheinlichkeit korrekt (aus mathematischer Sicht).


Algorithmen zum Primzahltest

- ☒ Miller-Rabin-Test
- ☐ Fermat-Test
- ☐ Solovay-Strassen-Test
- ☐ AKS-Test (deterministisches Verfahren)

Primzahltest

Zahl aus Datei laden

Zu testende Zahl

Ergebnis  Die Zahl ist keine Primzahl: 578960446186580977117854925043439539266349

Zahl testen

Abbrechen

# Anwendungsbeispiele (1)

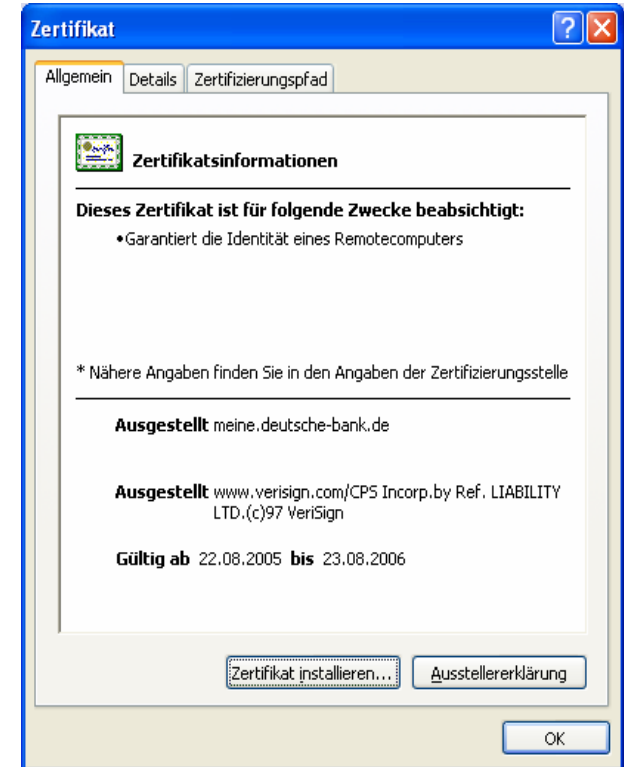
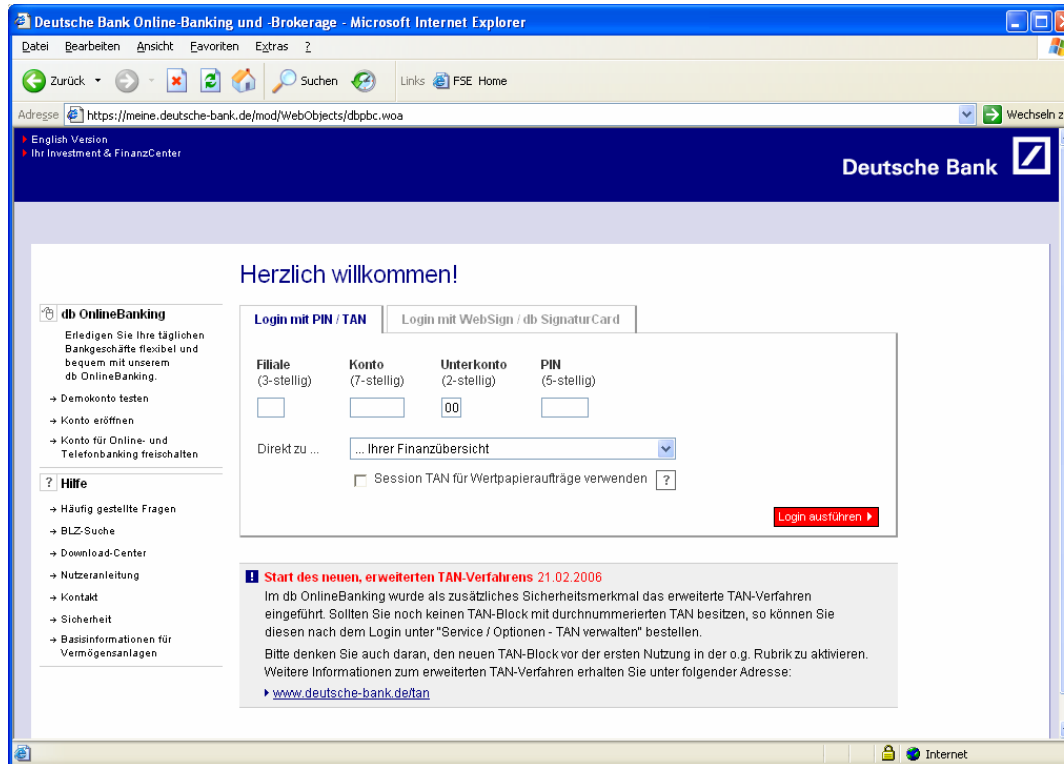
## Hybridverschlüsselung und Digitale Zertifikate

- **Hybridverschlüsselung** – Kombination aus asymmetrischer und symmetrischer Verschlüsselung
  1. Generierung eines zufälligen symmetrischen Sitzungs-Schlüssels (Session Key)
  2. Der Session Key wird – geschützt mit dem asymmetrischen Schlüssel – übertragen
  3. Die Nachricht wird – geschützt mit dem Session Key – übertragen
- **Problem:** Man-in-the-middle Angriffe: Gehört der öffentliche Schlüssel (Public Key) des Empfängers auch wirklich dem Empfänger?
- **Lösung: Digitale Zertifikate** – Eine zentrale Instanz (z.B. Telesec, VeriSign, Deutsche Bank PKI), der alle Benutzer trauen, garantiert die Authentizität des Zertifikates und des darin enthaltenen öffentlichen Schlüssels (analog zu einem vom Staat ausgestellten Personalausweis).
- **Hybridverschlüsselung auf Basis von digitalen Zertifikaten** ist die Grundlage für sichere elektronische Kommunikation:
  - Internet Shopping und Online Banking
  - Sichere E-Mail



# Anwendungsbeispiele (1)

## Gesicherte Online-Verbindung mit SSL und Zertifikaten

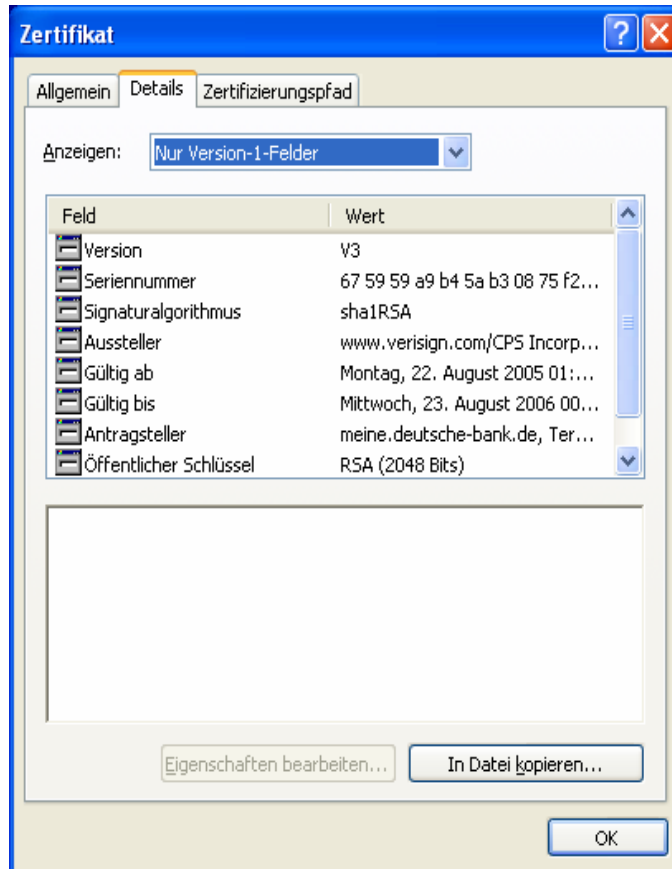


D.h. die Verbindung ist (zumindest einseitig) authentisiert und der übertragene Inhalt wird stark verschlüsselt.



# Anwendungsbeispiele (1)

## Attribute / Felder von Zertifikaten



### Grundlegende Attribute / Felder

- Aussteller (z.B. VeriSign)
- Antragsteller
- Gültigkeitszeitraum
- Seriennummer
- Zertifikatsart / Version (X.509v3)
- Signaturalgorithmus
- Öffentlicher Schlüssel (und Verfahren)

Öffentlicher Schlüssel



# Anwendungsbeispiele (1)

## Aufbau einer gesicherten SSL-Verbindung (Server Authentication)



# Anwendungsbeispiele (1)

## Aufbau einer gesicherten SSL-Verbindung (Server Authentication)

### ■ Allgemein

- Das Beispiel skizziert den typischen Aufbau einer SSL-Verbindung zur Übertragung von sensiblen Informationen (z.B. Internet-Shopping).
- Beim Aufbau der SSL-Verbindung authentisiert sich lediglich der Server durch ein digitales Zertifikat (die Authentisierung des Benutzer erfolgt in der Regel durch die Eingabe von Benutzername und Passwort nach dem Aufbau der SSL-Verbindung).
- SSL bietet auch die Möglichkeit einer zweiseitigen Authentisierung auf Basis digitaler Zertifikate.

### ■ Anmerkungen zur SSL-Verbindung

- ad (1): SSL Verbindungsaufbau – hierbei wird u.a. ausgehandelt welche Eigenschaften der Session Key besitzen soll (z.B. Bit-Länge) und welcher Algorithmus für die symmetrische Verschlüsselung verwendet werden soll (z.B. 3DES, AES).
- ad (2): Sofern Zwischenzertifikate notwendig sind (bei mehrstufigen Zertifikathierarchien), werden diese ebenfalls übertragen.
- ad (3): In diesem Schritt werden die im Browser installierten Root-Zertifikate verwendet, um das empfangene Server-Zertifikat zu validieren.
- ad (5): Der Session Key basiert auf den unter (1) ausgehandelten Eigenschaften.

# Anwendungsbeispiele (2)

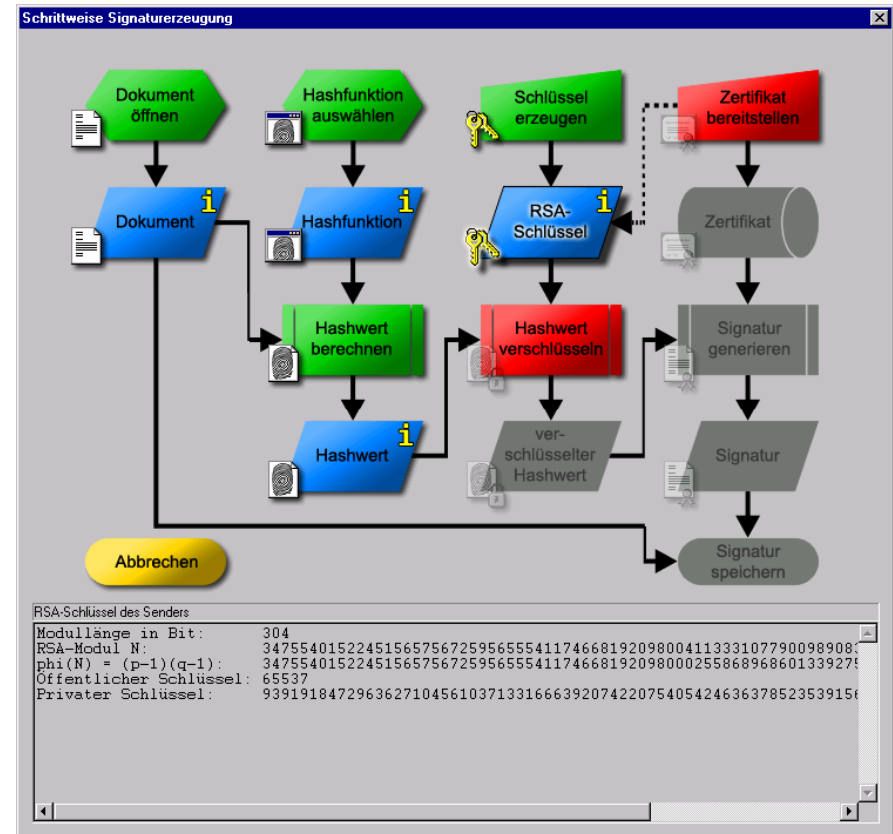
## Elektronische Signatur visualisiert

### Elektronische Signatur

- Wird immer wichtiger durch
  - Gleichstellung mit manueller Unterschrift (Signaturgesetz)
  - Zunehmenden Einsatz in Wirtschaft, durch den Staat und privat
- Wer weiß, wie sie funktioniert?

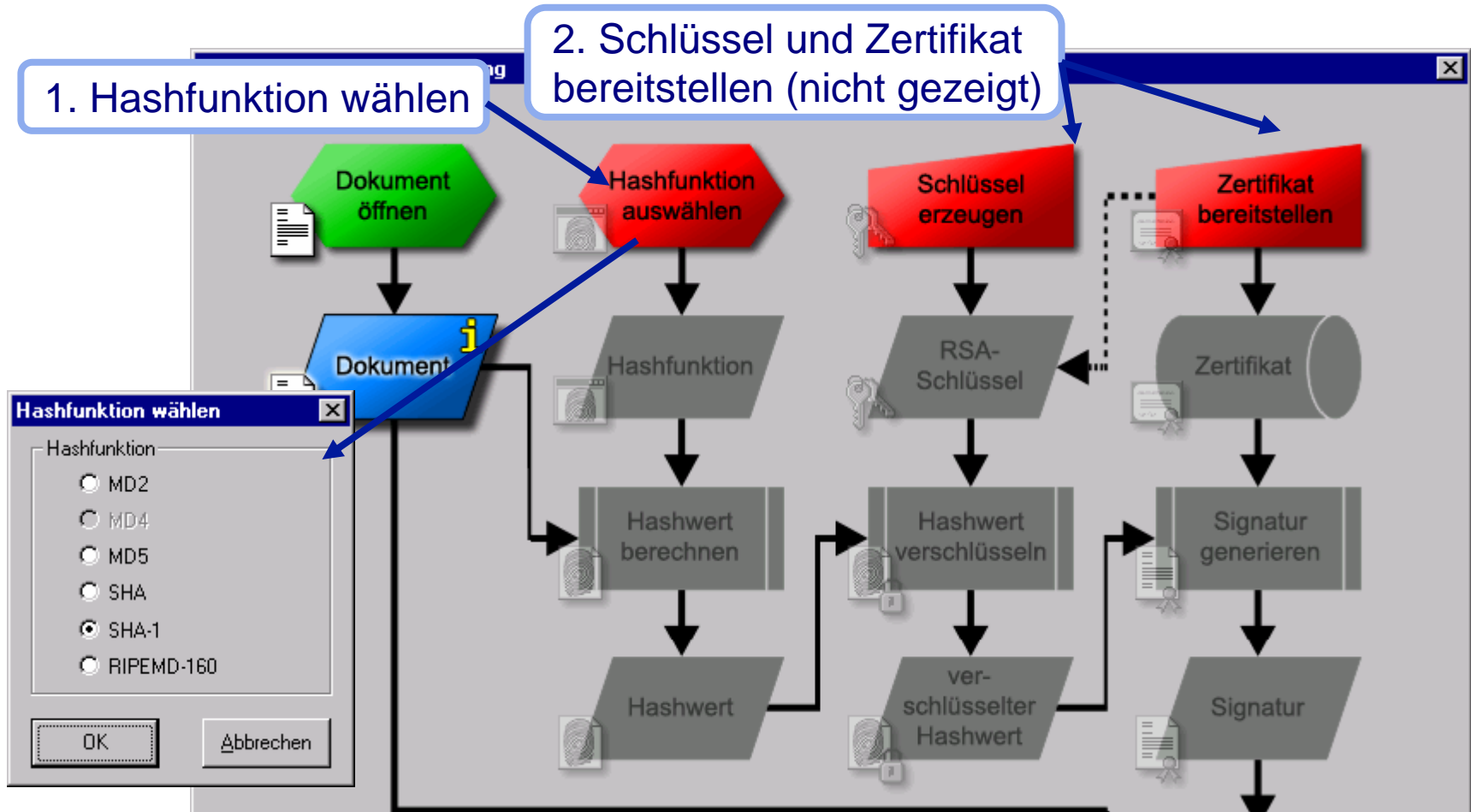
### Visualisierung in CrypTool

- Siehe Menü „Digitale Signaturen/PKI“ \ „Signaturdemo (Signaturerzeugung)“
- Interaktives Datenflussdiagramm
- Ähnlich wie die Visualisierung der Hybridverschlüsselung



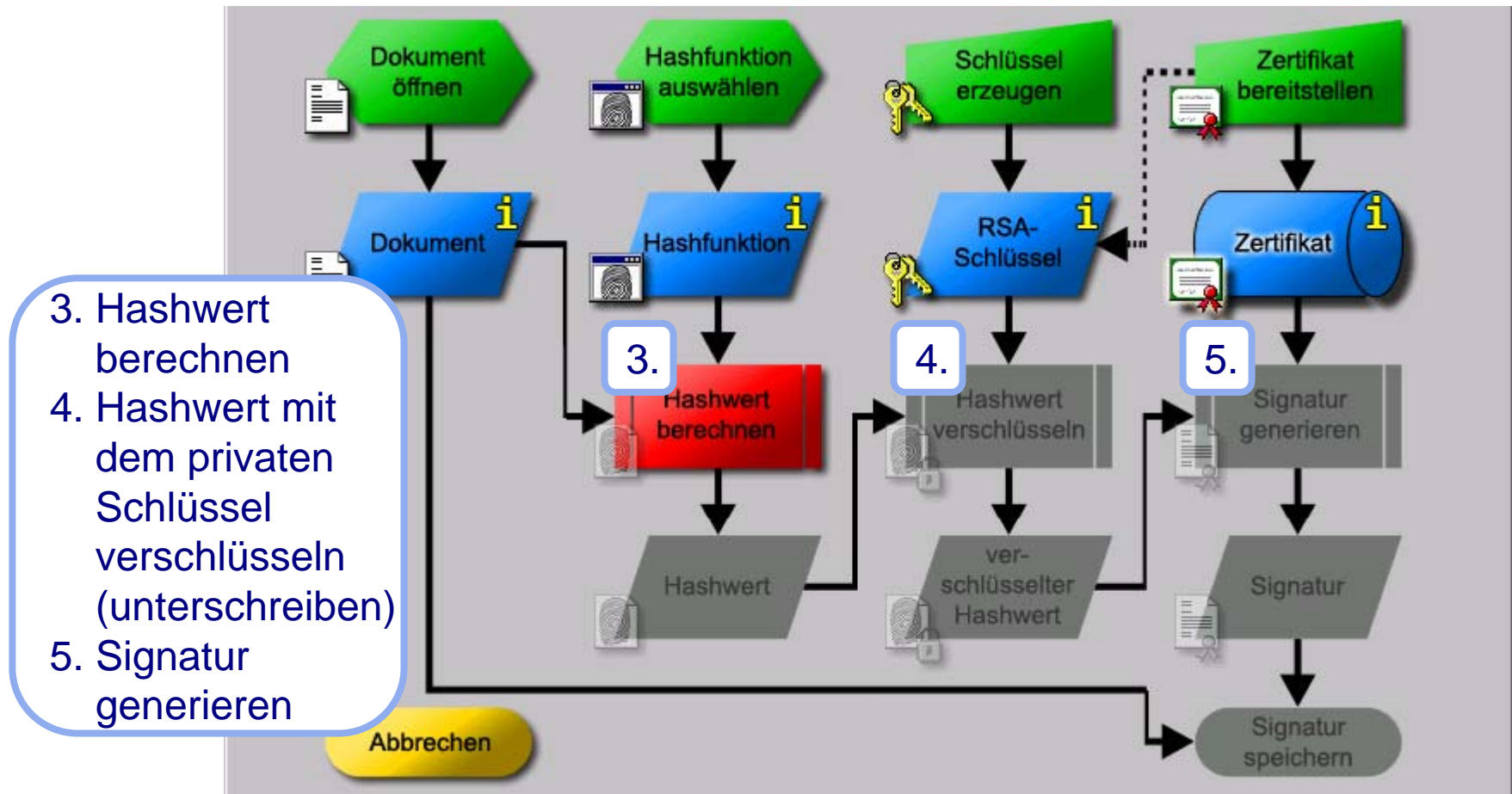
# Anwendungsbeispiele (2)

Elektronische Signatur visualisiert: a) Vorbereitung



# Anwendungsbeispiele (2)

## Elektronische Signatur visualisiert: b) Kryptographie



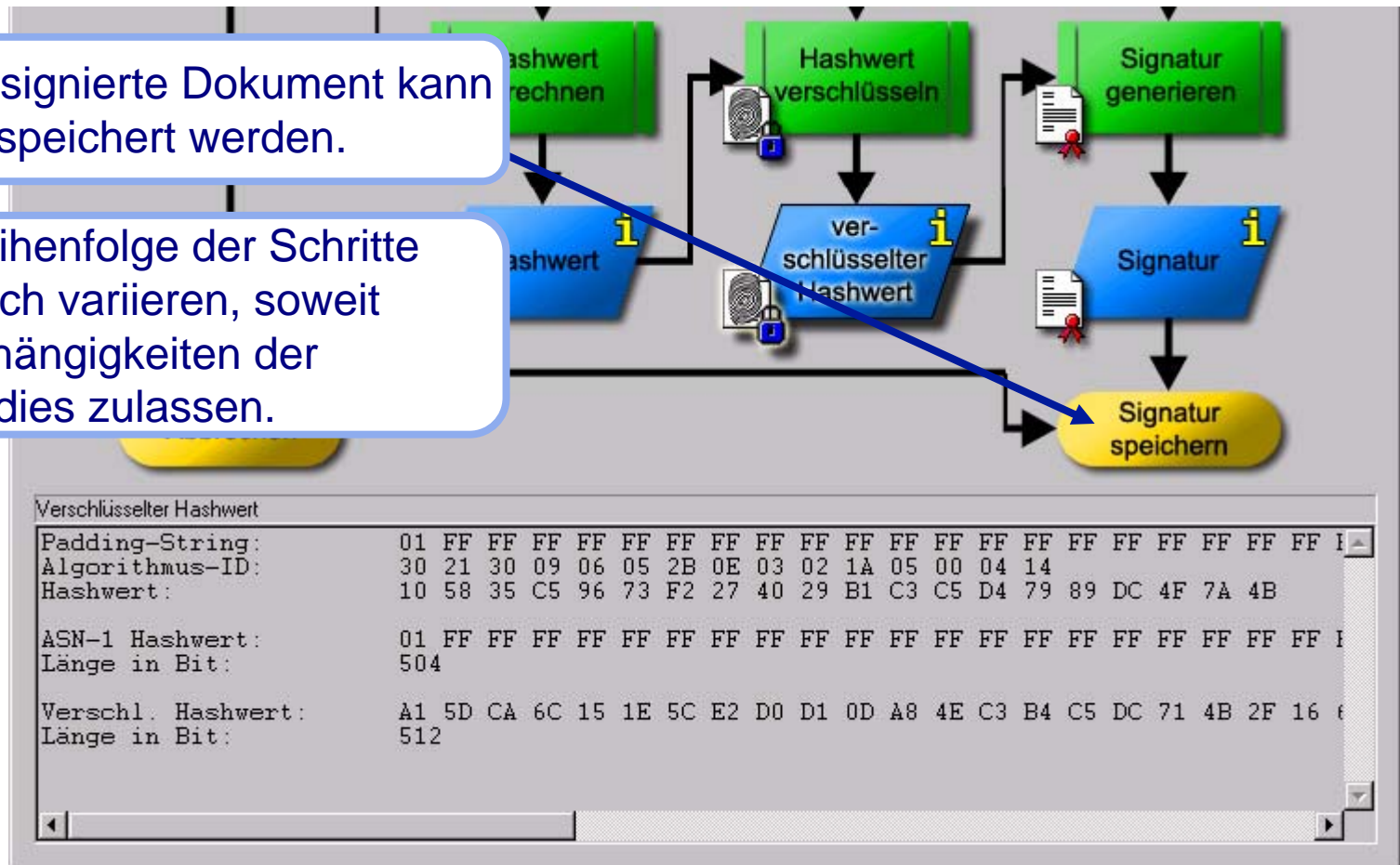


## Anwendungsbeispiele (2)

Elektronische Signatur visualisiert: c) Ergebnis

6. Das signierte Dokument kann nun gespeichert werden.

Die Reihenfolge der Schritte lässt sich variieren, soweit die Abhängigkeiten der Daten dies zulassen.





# Anwendungsbeispiele (3)

## 3. Angriff auf zu kurzen RSA-Modul N

### Aufgabe aus *Song Y. Yan, Number Theory for Computing, Springer, 2000*

- Öffentlicher Schlüssel
  - RSA-Modul  $N = 63978486879527143858831415041$  (95bit, 29 Dezimalstellen)
  - Öffentlicher Exponent  $e = 17579$
- Verschlüsselter Text (Blocklänge = 8):
  - $C_1 = 45411667895024938209259253423$ ,
  - $C_2 = 16597091621432020076311552201$ ,
  - $C_3 = 46468979279750354732637631044$ ,
  - $C_4 = 32870167545903741339819671379$
- Der Text soll entschlüsselt werden.

Für die eigentliche  
Kryptoanalyse (das Finden des  
privaten Schlüssels) ist der  
Ciphertext nicht notwendig !

### Lösung mit **CrypTool** (ausführlich in den Szenarien der Online-Hilfe beschrieben)

- Öffentliche Parameter in RSA-Kryptosystem (Menü Einzelverfahren) eintragen
- Funktion „RSA-Modul faktorisieren“ liefert die Primfaktoren  $p$  und  $q$  mit  $pq = N$
- Daraus wird der geheime Schlüssel  $d = e^{-1} \bmod (p-1)(q-1)$  abgeleitet
- Entschlüsseln des Textes mit Hilfe von  $d$ :  $M_i = C_i^d \bmod N$

**Angriff mit CrypTool ist für RSA-Module bis ca. 250 bit praktikabel**

**Dann könnte man für jemand anderen elektronisch unterschreiben !!!**

# Anwendungsbeispiele (3)

## Kurzer RSA-Modul: Öffentliche Parameter eingeben

**Das RSA-Kryptosystem**

☐ RSA mit privatem und öffentlichem Schlüssel -- oder nur mit öffentlichem Schlüssel

☐ Wählen Sie 2 Primzahlen p und q. Die Zahl  $N = pq$  ist der öffentliche RSA-Modul und  $\phi(N) = (p-1)(q-1)$  ist die Eulersche Zahl. Der öffentliche Schlüssel e ist teilerfremd zu  $\phi(N)$ . Daraus wird der geheime Schlüssel  $d = e^{-1} \pmod{\phi(N)}$  berechnet.

☒ Zur Verschlüsselung von Daten oder zur Verifikation einer Signatur genügt es, dass Sie die veröffentlichten RSA-Parameter angeben: den RSA-Modul N und den öffentlichen Schlüssel e.

**Faktorisierungsangriff**

Sie können mit Hilfe der Faktorisierung versuchen, den öffentlichen RSA-Modul N in seine Primfaktoren p und q zu faktorisieren.

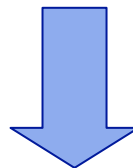
**RSA-Parameter**

RSA-Modul N	<input type="text" value="63978486879527143858831415041"/>	(öffentlich)
$\phi(N) = (p-1)(q-1)$	<input type="text"/>	(geheim)
öffentlicher Schlüssel e	<input type="text" value="17579"/>	
geheimer Schlüssel d	<input type="text"/>	

☐ RSA-Verschlüsselung mit e / Entschlüsselung mit d

1. Öffentliche RSA-Parameter N und e eingeben\*

2. Faktorisieren



# Anwendungsbeispiele (3)

## Kurzer RSA-Modul: RSA-Modul faktorisieren

The screenshot shows the 'Faktorisieren einer Zahl' (Factorizing a Number) window in CryptTool. The 'Algorithmen zur Faktorisierung' (Factoring Algorithms) section has several checked options: Brute-force Methode, Algorithmus nach Brent, Pollard Methode, Williams Methode, Algorithmus nach Lenstra, and Quadratische Sieb Methode. The 'Eingabe' (Input) section contains the number 63978486879527143858831415041. The 'Faktorisierungsergebnis' (Factorization Result) section shows the last factorization was done using Pollard's method, took 01,191 seconds, and displays the product representation of the factorization: 145295143558111 \* 440334654777631. A blue circle highlights this product, with an arrow pointing to a text box that says '3. Faktorisierung ergibt p und q' (3. Factorization gives p and q). Another arrow points from this text box to the 'OK' button in a smaller 'CryptTool' dialog box that appears on the right. This dialog box contains an information icon and the text: 'Der RSA-Modul N wurde erfolgreich in die Primzahlen p und q faktorisiert! Sie können die RSA-Operation auch mit dem geheimen Schlüssel d durchführen: benutzen Sie hierfür den Knopf Entschlüsseln.' (The RSA modulus N was successfully factorized into the prime numbers p and q! You can also perform the RSA operation with the secret key d: use the button Decrypt for this purpose.)

**Faktorisieren einer Zahl**

Algorithmen zur Faktorisierung

- ☒ Brute-force Methode
- ☒ Algorithmus nach Brent
- ☒ Pollard Methode
- ☒ Williams Methode
- ☒ Algorithmus nach Lenstra
- ☒ Quadratische Sieb Methode

Eingabe

Geben Sie die zu faktorisierende Zahl ein:

63978486879527143858831415041

Faktorisierungsergebnis

Die Faktorisierung wird in dem Format  $\langle z_1^{a_1} * z_2^{a_2} * \dots * z_n^{a_n} \rangle$  dargestellt. Zusammengesetzte Zahlen sind rot markiert.

Letzte Faktorisierung durch: Pollard

Benötigte Zeit: 01,191 Sekunden.

Produkt Darstellung der Faktorisierung:

145295143558111 \* 440334654777631

3. Faktorisierung ergibt p und q

**CryptTool**

Der RSA-Modul N wurde erfolgreich in die Primzahlen p und q faktorisiert! Sie können die RSA-Operation auch mit dem geheimen Schlüssel d durchführen: benutzen Sie hierfür den Knopf Entschlüsseln.

OK

Schließen

# Anwendungsbeispiele (3)

## Kurzer RSA-Modul: geheimen Schlüssel d bestimmen

**Das RSA-Kryptosystem**

☐ RSA mit privatem und öffentlichem Schlüssel -- oder nur mit öffentlichem Schlüssel

☒ Wählen Sie 2 Primzahlen p und q. Die Zahl  $N = pq$  ist der öffentliche RSA-Modul und  $\phi(N) = (p-1)(q-1)$  ist die Eulersche Zahl. Der öffentliche Schlüssel e ist teilerfremd zu  $\phi(N)$ . Daraus wird der geheime Schlüssel  $d = e^{-1} \pmod{\phi(N)}$  berechnet.

☐ Zur Verschlüsselung von Daten oder zur Verifikation einer Signatur genügt es, dass Sie die veröffentlichten RSA-Parameter angeben: den RSA-Modul N und den öffentlichen Schlüssel e.

Primzahleingabe

Primzahl p: 145295143558111

Primzahl q: 440334654777631

RSA-Parameter

RSA-Modul N: 63978486879527143858831415041 (öffentlich)

$\phi(N) = (p-1)(q-1)$ : 63978486879526558229033079300 (geheim)

öffentlicher Schlüssel e: 17579

geheimer Schlüssel d: 10663687727232084624328285019

RSA-Verschlüsselung mit e / Entschlüsselung mit d

Eingabe als ☐ Text ☒ Zahlen

Eingabe der Nachricht als Zahlen im Format: Zahl(1) # Zahl(2) # ... # Zahl(n) (Zahlen zur Basis 10).

Wechsel in die Ansicht des Besitzers des geheimen Schlüssels.

4. p und q wurden automatisch eingetragen und der geheime Schlüssel d berechnet.

5. Optionen einstellen

# Anwendungsbeispiele (3)

## Kurzer RSA-Modul: Optionen einstellen

**Optionen für die RSA-Verschlüsselung**

Textoptionen

☐ Alle 256 Zeichen      Anzahl Zeichen: 27

☒ Alphabet vorgeben:

Verfahren bei der Kodierung der Nachricht in Zahlen

Methode: ☐ b-adisch    ☒ Basissystem

Blocklänge

Die Anzahl der Zeichen, die pro RSA-Operation verschlüsselt werden.  
Die maximale Anzahl ist abhängig von der Bitlänge des RSA-Moduls N, der Anzahl der Zeichen im Alphabet und der Kodierungsmethode der Nachricht.

Blocklänge in Zeichen:  (maximale Blocklänge 14 Zeichen)

Zahlensystem

Die Zahlen der RSA-Ver-/Entschlüsselung werden in dem folgenden Zahlensystem dargestellt.

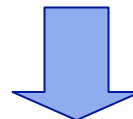
☒ Dezimal    ☐ Binär    ☐ Oktal    ☐ Hexadezimal

OK      Abbrechen

6. Alphabet wählen

7. Kodierung wählen

8. Blocklänge wählen



# Anwendungsbeispiele (3)

## Kurzer RSA-Modul: Text entschlüsseln

RSA-Parameter

RSA-Modul N	63978486879527143858831415041	(öffentlich)
$\phi(N) = (p-1)(q-1)$	63978486879526558229033079300	(geheim)
öffentlicher Schlüssel e	17579	
geheimer Schlüssel d	10663687727232084624328285019	

Parameter aktualisieren

RSA-Verschlüsselung mit e / Entschlüsselung mit d

Eingabe als ☐ Text ☒ Zahlen Optionen für Alphabet und Zahlensystem...

Chiffretext in Zahlendarstellung zur Basis 10 .

7091621432020076311552201 # 46468979279750354732637631044 # 32870167545903741339819671279

Entschlüsselung in den Klartext  $m[i] = c[i]^d \pmod{N}$

00000000000001401202118011200 # 00000000000001421130205181900 # 000000000000011805001301

Ausgabertext aus der Entschlüsselung (in Blöcken der Länge 8; das Symbol '#' dient nur als Trennzeichen).

NATURAL # NUMBERS # ARE MADE # BY GOD

Klartext

NATURAL NUMBERS ARE MADE BY GOD

Verschlüsseln Entschlüsseln Schließen

9. Ciphertext eingeben

10. Entschlüsseln

# Anwendungsbeispiele (4)

## Analyse der Verschlüsselung im PSION 5

### Praktische Durchführung der Kryptoanalyse: Angriff auf die Verschlüsselungsoption der Textverarbeitungsapplikation im PSION 5 PDA



**Gegeben: eine auf dem PSION verschlüsselte Datei**

#### Voraussetzung

- verschlüsselter deutscher oder englischer Text
- je nach Verfahren und Schlüssellänge 100 Byte bis einige kB Text

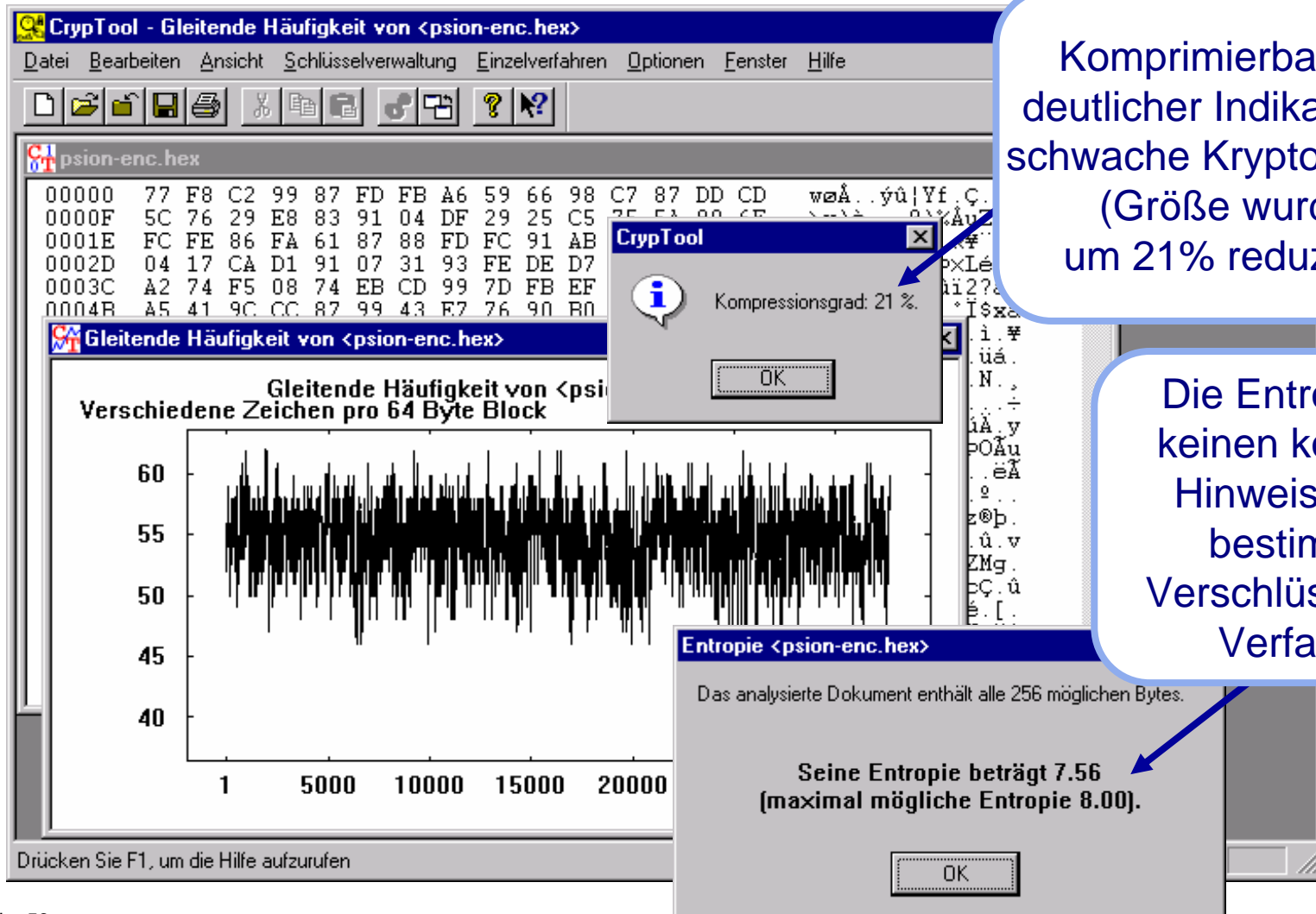
#### Vorgehen

- Voranalyse
  - Entropie
  - gleitende Häufigkeit
  - Kompressionstest
- Autokorrelation
- automatische Analyse mit verschiedenen klassischen Verfahren durchprobieren

} wahrscheinlich klassische  
Verschlüsselung

# Anwendungsbeispiele (4)

## PSION-PDA: Entropie bestimmen, Kompressionstest



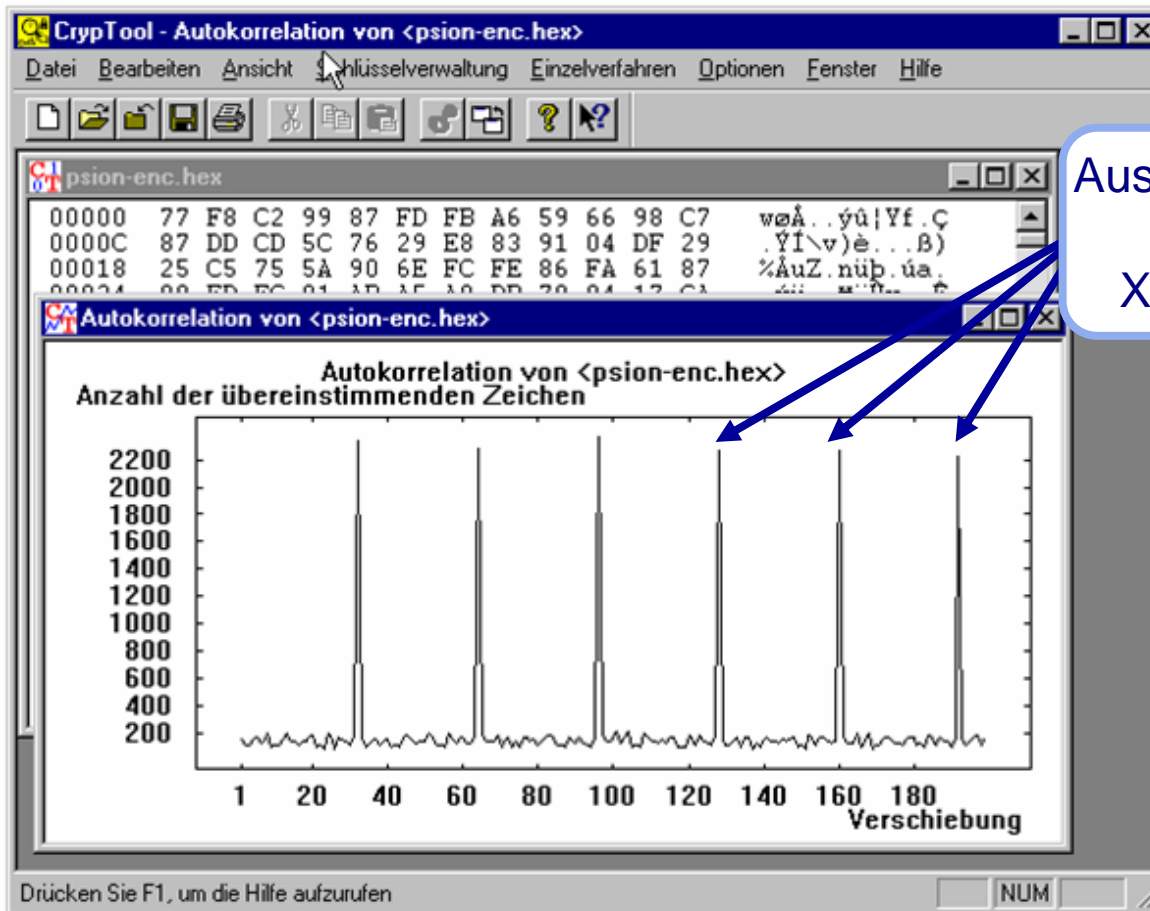
Komprimierbarkeit:  
deutlicher Indikator für  
schwache Kryptographie  
(Größe wurde  
um 21% reduziert)

Die Entropie gibt  
keinen konkreten  
Hinweis auf ein  
bestimmtes  
Verschlüsselungs-  
Verfahren.



# Anwendungsbeispiele (4)

## PSION-PDA: Autokorrelation bestimmen



Ausgeprägtes Kamm-Muster:  
typisch für Vigenère,  
XOR und binäre Addition

\* Diese verschlüsselte Datei wird mit CrypTool ausgeliefert (siehe CrypTool/examples/psion-enc.hex)

# Anwendungsbeispiele (4)

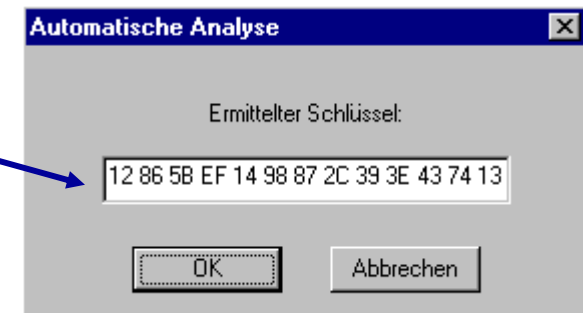
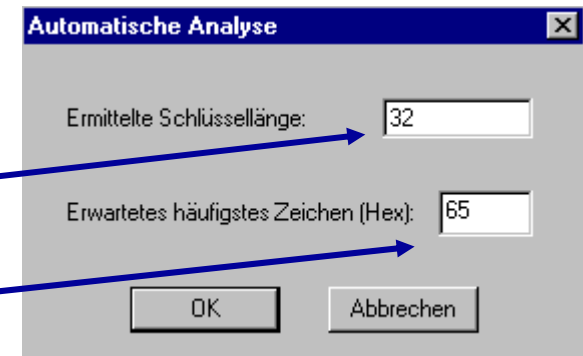
## PSION-PDA: Automatische Analyse

**Automatische Analyse Vigenère: kein Erfolg**

**Automatische Analyse XOR: kein Erfolg**

**Automatische Analyse binäre Addition:**

- CrypTool ermittelt die Schlüssellänge mittels Autokorrelation: 32 Byte
- Das erwartete häufigste Zeichen kann der Benutzer wählen: „e“ = 0x65 (ASCII-Code)
- Analyse ermittelt den (unter der Verteilungsannahme) wahrscheinlichsten Schlüssel



## Anwendungsbeispiele (4)

### PSION-PDA: Ergebnis der automatischen Analyse

#### Ergebnis der automatischen Analyse unter der Annahme „binäre Addition“:

- Ergebnis gut, aber nicht perfekt: 24 von 32 Schlüsselbytes richtig.
- Die Schlüssellänge 32 wurde korrekt bestimmt. ←

The screenshot shows a window titled "Automatische ADD-Analyse von <psion-enc.hex>, Schlüssel: <12 86 5B EF 14 98 87 2C 33 3E 43...". The window is divided into two main sections. The left section displays a hex dump with addresses from 00000 to 000F0 and corresponding hexadecimal values. The right section displays the decrypted text, which appears to be a German document snippet. A blue arrow points from the second bullet point of the text above to the key length "32" in the window title.

Address	Hex Data	Decrypted Text
00000	65 72 67 AA 73 65 74 7A 20 28 55 53 74 47 29 06	erg³setz (UStG).
00010	06 06 8A 72 73 74 65 72 65 41 62 B8 A8 68 6E AE	...rstereAb,"hn@
00020	74 74 06 98 74 65 75 65 72 67 65 67 65 6E 73 74	tt...teuergegenst
00030	61 6E A9 20 75 6E 64 20 8C 65 6C B9 BA 6E 67 B8	an@ und .el¹ng,
00040	62 65 72 AA 69 63 68 06 06 A7 20 31 2E 06 28 31	ber³ich...\$ 1..(1
00050	29 20 89 65 72 20 55 6D B8 61 74 BF B8 74 65 BA	) .er Um,at¿,te²
00060	65 72 20 BA 6E 74 65 72 6C 69 65 67 65 6E 20 64	er ²nterliegen d
00070	69 65 65 66 6F 6C 67 65 B3 64 65 B3 65 55 6D B8	ieefolge³de³eUm,
00080	E4 74 7A AA 3A 06 31 2E 20 64 69 65 20 4C 69 65	ätz³:.1. die Lie
00090	66 65 B7 75 6E 67 65 6E 65 75 6E A9 65 73 6F B3	fe·ungeneun@eso³
000A0	73 74 69 AC 65 6E 20 4C 65 69 73 74 75 6E 67 65	stiren Leistunge
000B0	6E 2C 65 64 69 65 20 65 AE 6E 20 9A B3 74 65 B7	n,edie e@n .³te·
000C0	6E 65 68 B2 65 72 20 69 6D 20 49 6E 6C 61 6E 64	neh²er im Inland
000D0	20 67 AA 67 65 6E 20 45 B3 74 67 AA B1 74 20 AE	g³gen E³tg³tt @
000E0	6D 20 52 A6 68 6D 65 6E 20 73 65 69 6E 65 73 20	m R hmen seines
000F0	55 6E B9 65 72 6E 65 68 B2 65 6E B8 65 61 75 B8	Un¹erneh²en,eau,

- Das eingegebene Passwort war nicht 32 Byte lang.  
⇒ PSION Word leitet aus dem Passwort den eigentlichen Schlüssel ab.
- Nacharbeiten von Hand liefert den entschlüsselten Text

# Anwendungsbeispiele (4)

## PSION-PDA: Bestimmung der restlichen Schlüsselbytes

### Schlüssel während der automatischen Analyse in die Zwischenablage kopieren

#### Im Hexdump der automatischen Analyse

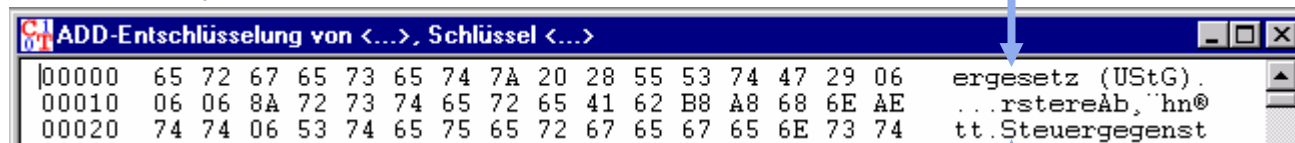
- Falsche Bytepositionen bestimmen, z.B. 0xAA an Position 3
- Korrespondierende korrekte Bytes erraten und notieren: „e“ = 0x65

#### Im Hexdump der verschlüsselten Ausgangsdatei

- Ausgangsbytes an der ermittelten Bytepositionen bestimmen: 0x99
- Mit CALC.EXE korrekte Schlüsselbytes errechnen:  $0x99 - 0x65 = 0x34$

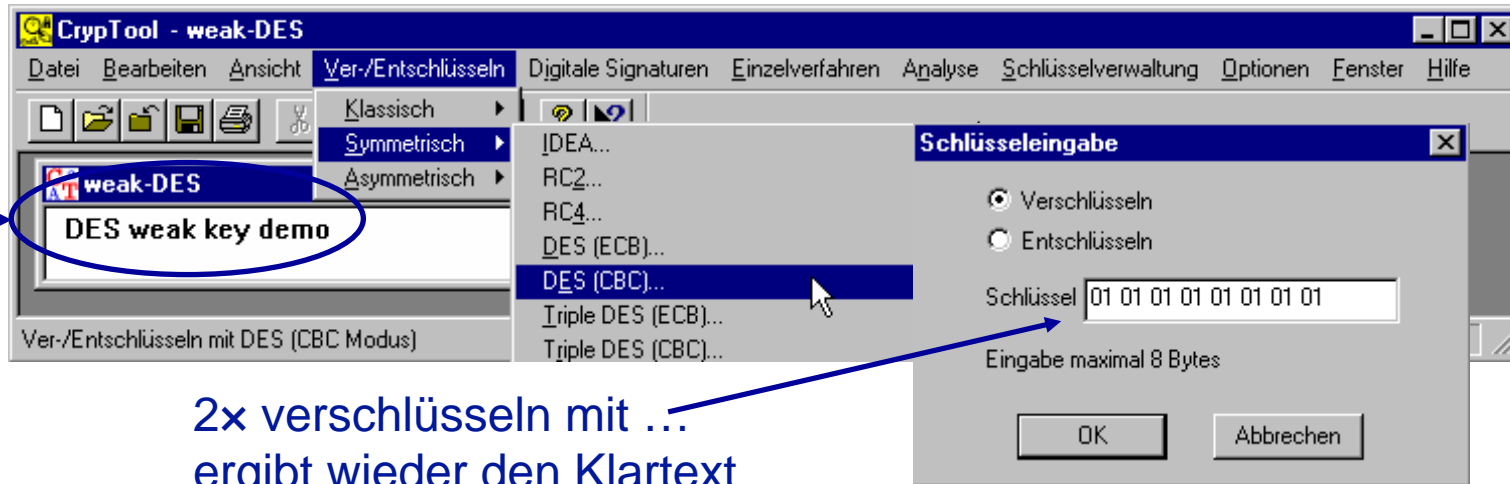
### Schlüssel aus der Zwischenablage

- korrigieren  
12865B341498872C393E43741396A45670235E111E907AB7C0841...
- verschlüsseltes Ausgangsdokument mittels binärer Addition entschlüsseln
- Nun sind die Bytepositionen 3, 3+32, 3+2\*32, ... ok

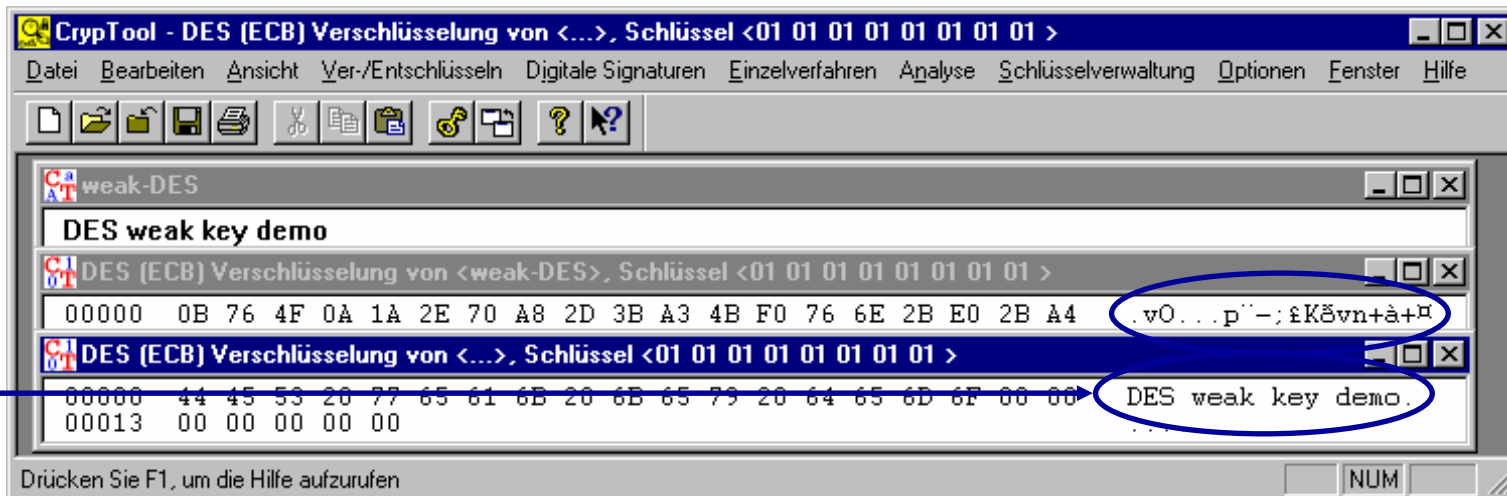


# Anwendungsbeispiele (5)

„Schwache“ DES-Schlüssel – Implementierung bestätigt die Angaben der Literatur [vgl. HAC]



2x verschlüsseln mit ...  
ergibt wieder den Klartext



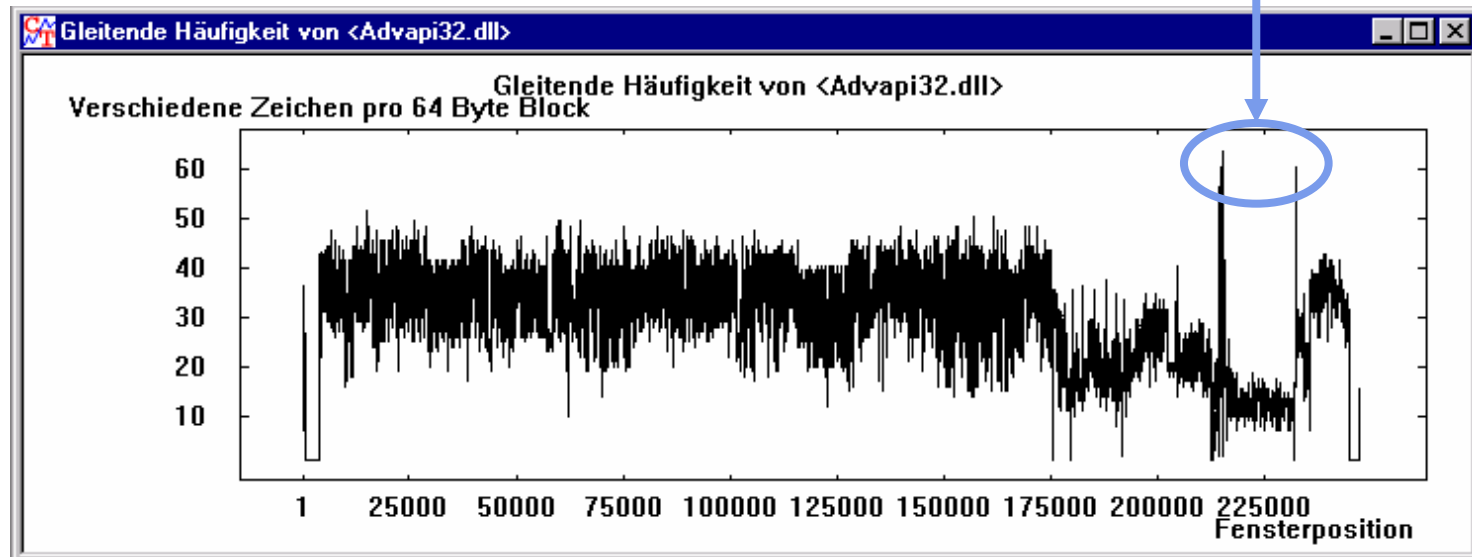
# Anwendungsbeispiele (6)

## Auffinden von Schlüsselmaterial

**Die Funktion „Gleitende Häufigkeit“ eignet sich zum Auffinden von Schlüsselmaterial und verschlüsselten Bereichen in Dateien.**

### Hintergrund:

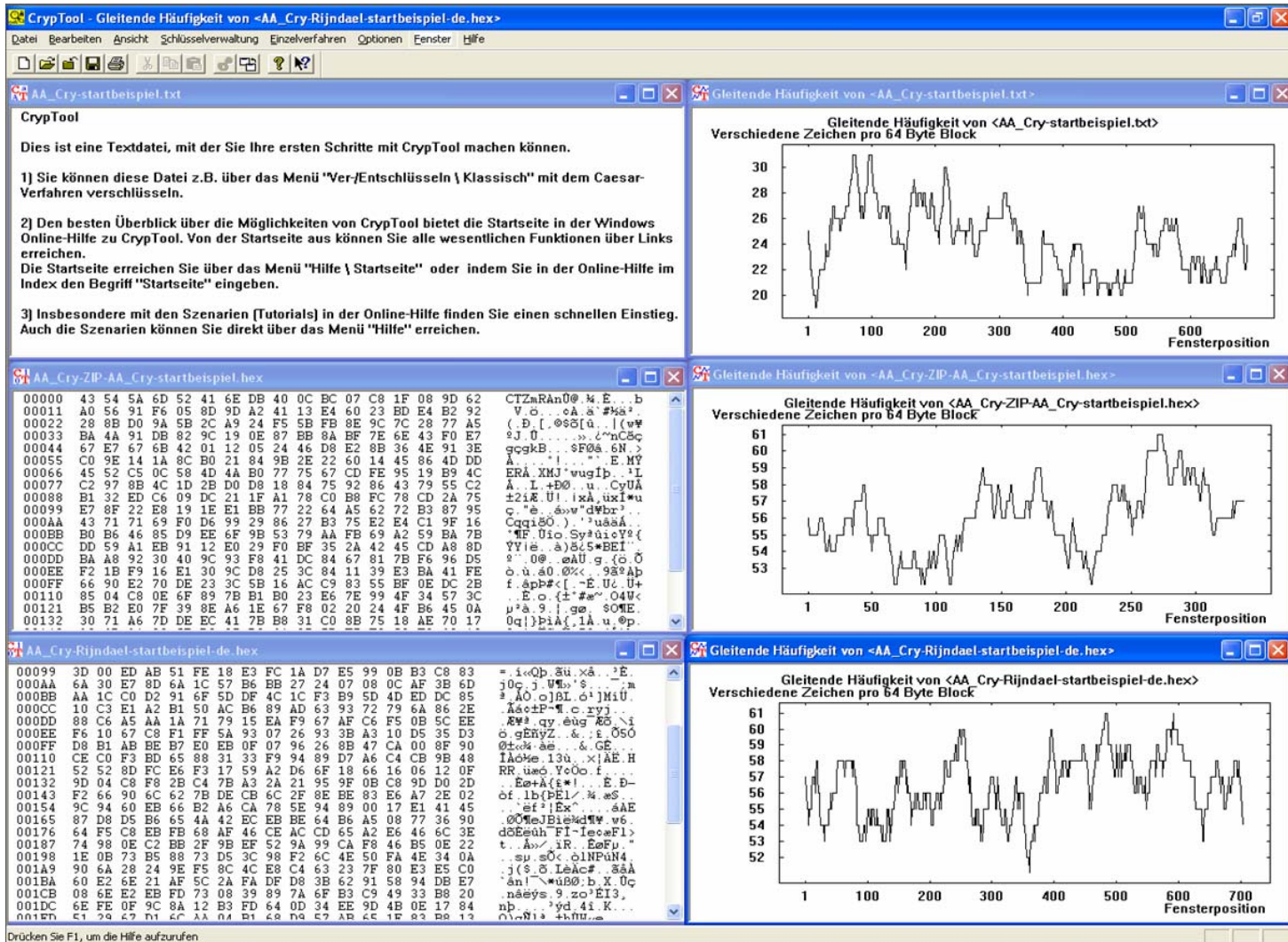
- diese Daten sind „zufälliger“ als Text oder Programmcode
- sie sind als Peak in der „gleitenden Häufigkeit“ zu erkennen
- Beispiel: der „NSAKEY“ in advapi32.dll





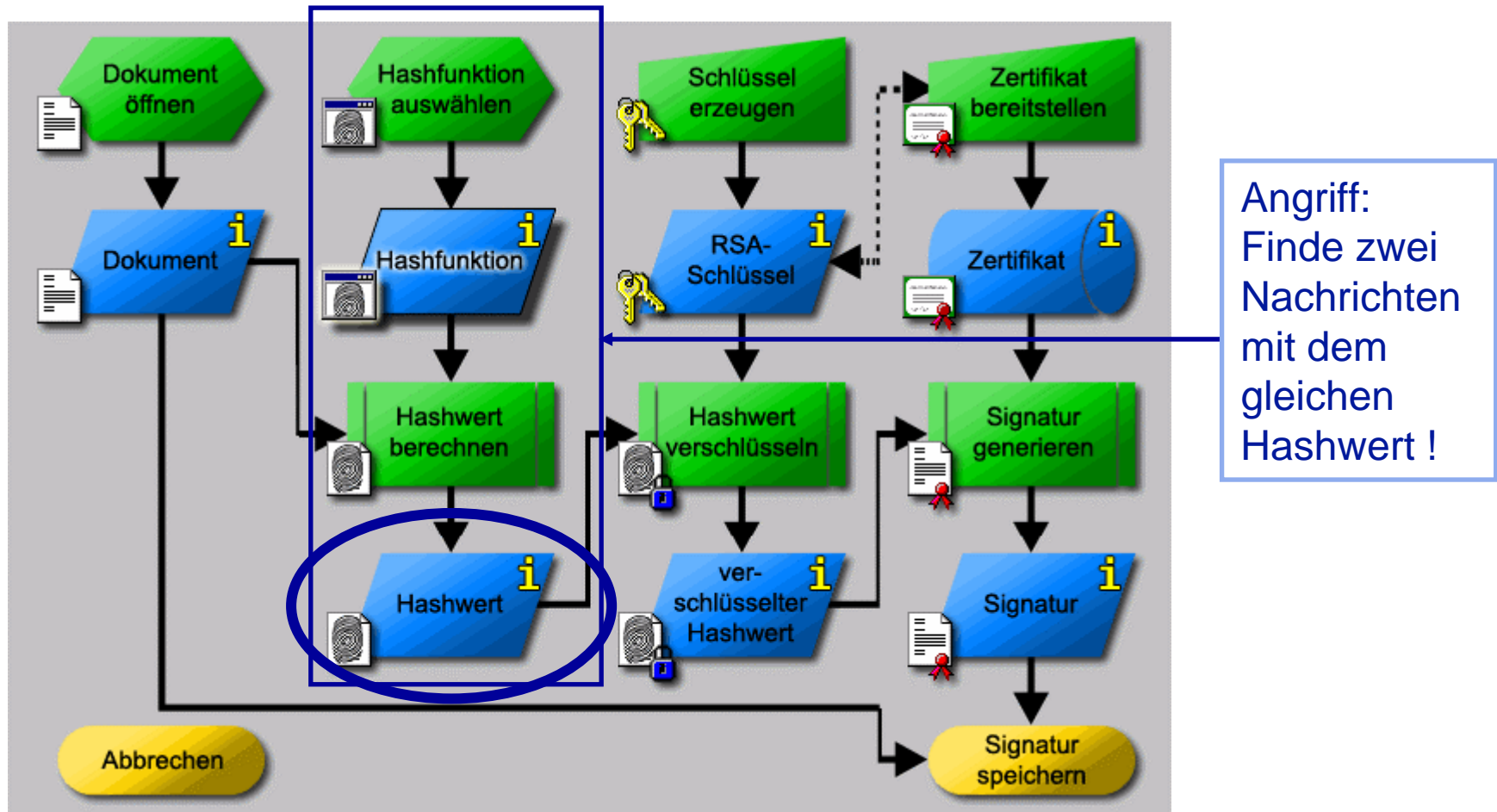
# Anwendungsbeispiele (6)

## Vergleich der gleitenden Häufigkeit anderer Dateien



# Anwendungsbeispiele (7)

## Angriffsziel digitale Signatur





# Anwendungsbeispiele (7)

## Angriff auf digitale Signatur: Idee (1)

### Angriff auf die digitale Signatur eines ASCII-Textes durch Suche nach Hashkollisionen

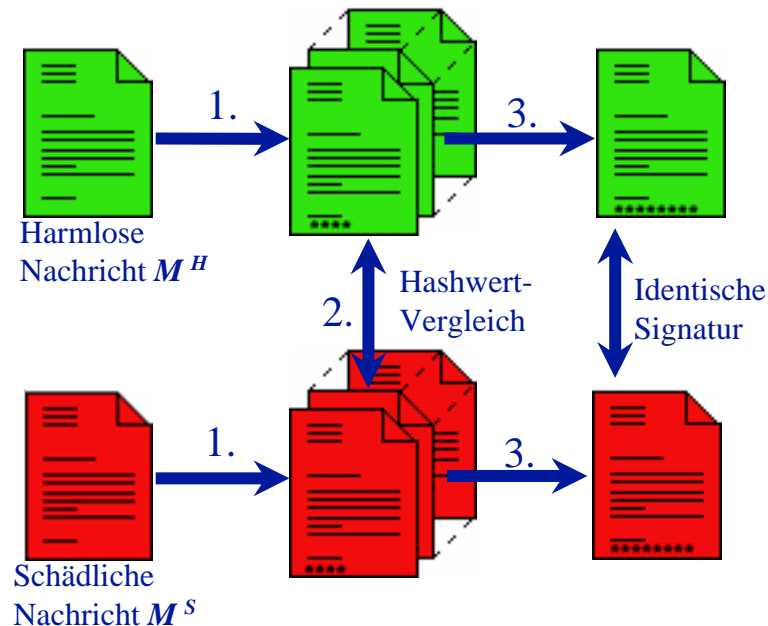
#### Idee:

- ASCII-Text kann mittels **nicht-druckbarer** Zeichen modifiziert werden, ohne den lesbaren Inhalt zu verändern
- Modifiziere parallel zwei Texte, bis eine Hashkollision erreicht wird
- Ausnutzung des Geburtstagsparadoxons (Geburtstagsangriff)
- Generischer Angriff auf beliebige Hashfunktion
- Angriff ist gut parallelisierbar (nicht implementiert)
- In Cryptool implementiert im Rahmen der Bachelor-Arbeit „*Methoden und Werkzeuge für Angriffe auf die digitale Signatur*“, 2003.

Konzepte: Mappings, modifizierter Floyd-Algorithmus (konstanter Speicherbedarf) !

# Anwendungsbeispiele (7)

## Angriff auf digitale Signatur: Idee (2)



- 1. Modifikation:** Ausgehend von der Nachricht  $M$  werden  $N$  verschiedene Nachrichten  $M_1, \dots, M_N$  – „inhaltlich“ gleich mit der Ausgangsnachricht – erzeugt.
- 2. Suche:** Gesucht werden *modifizierte* Nachrichten  $M_i^H$  und  $M_j^S$  mit gleichem Hashwert.
- 3. Angriff:** Die Signaturen zweier solcher Dokumente  $M_i^H$  und  $M_j^S$  sind identisch.

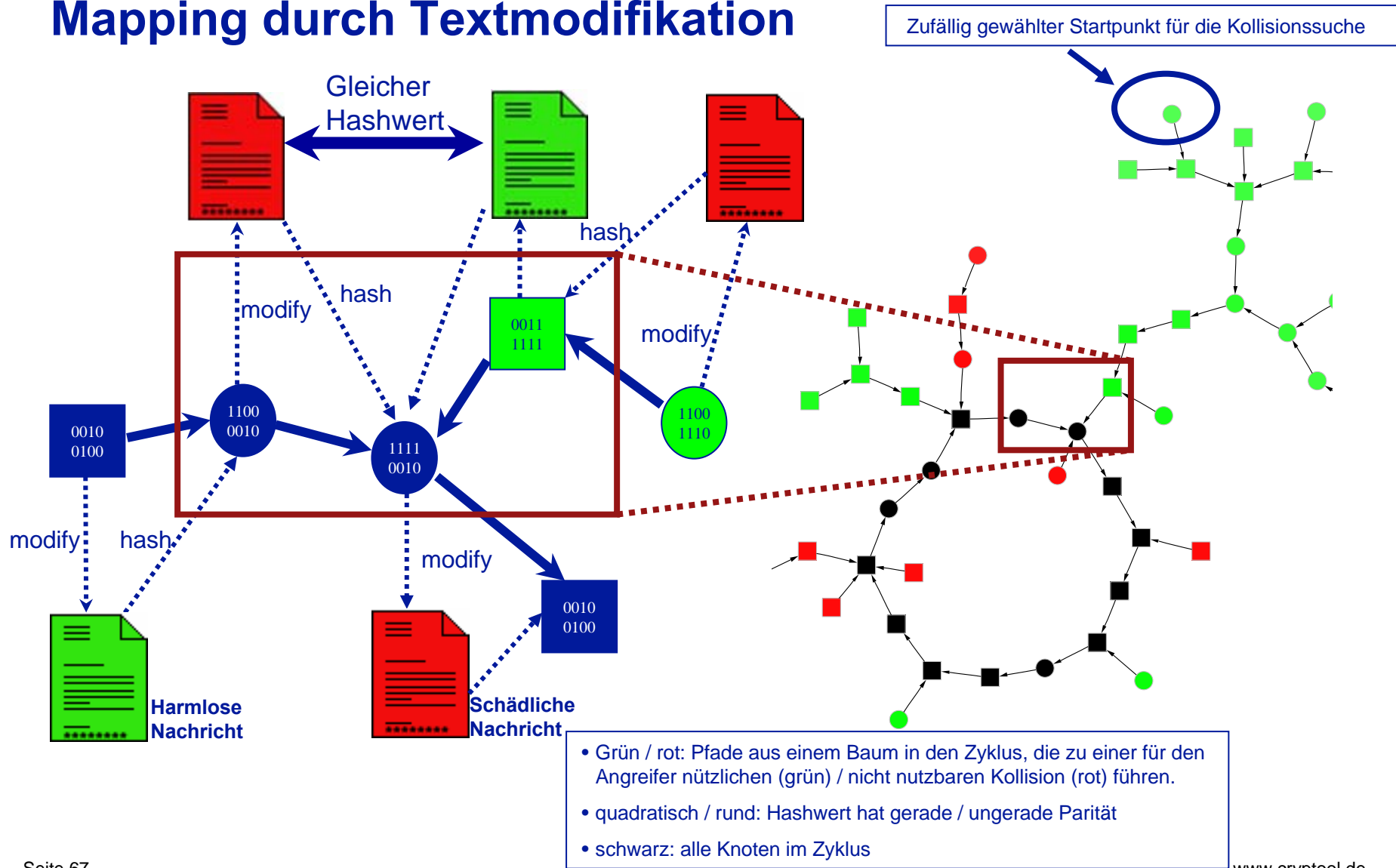
### Für Hashwerte der Bitlänge $n$ sagt das Geburtstagsparadoxon:

- Kollisionssuche zwischen  $M^H$  und  $M_1^S, \dots, M_N^S$ :  $N \approx 2^n$
- Kollisionssuche zwischen  $M_1^H, \dots, M_N^H$  und  $M_1^S, \dots, M_N^S$ :  $N \approx 2^{n/2}$

↑  
Erwartete Anzahl der zu erzeugenden Nachrichten, um eine Kollision zu erhalten.

# Hashkollisionssuche

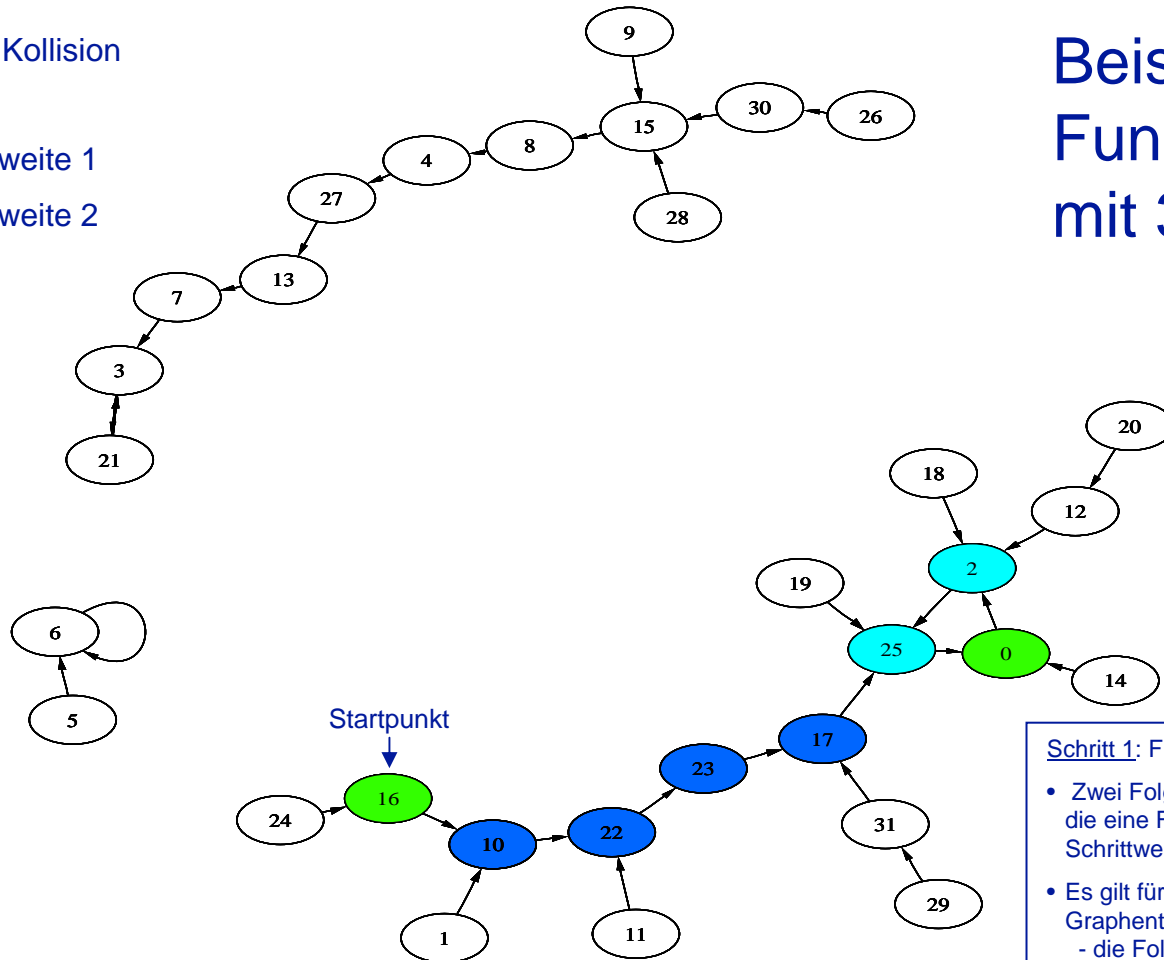
## Mapping durch Textmodifikation



# Hashkollisionssuche

## Floyd-Algorithmus: Treffen im Zyklus

-  Start / Kollision
-  Zyklus
-  Schrittweite 1
-  Schrittweite 2



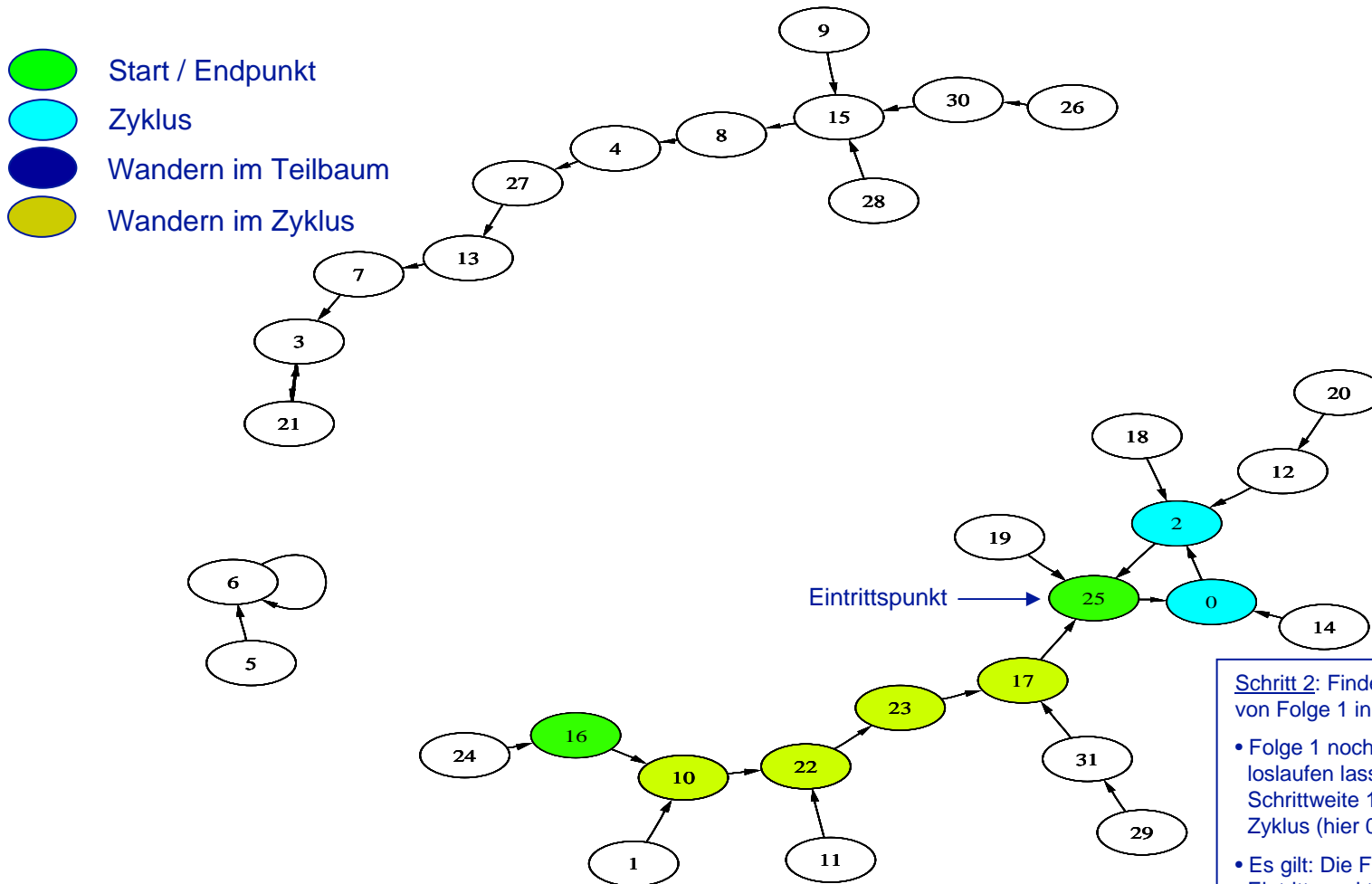
Beispiel:  
Funktionsgraph  
mit 32 Knoten

Schritt 1: Finden des Treffpunktes im Zyklus:

- Zwei Folgen mit gleichem Start [16]:  
die eine Folge hat Schrittweite 1, die andere Schrittweite 2.
- Es gilt für alle Zykluslängen (aufgrund der Graphentheorie):
  - die Folgen enden immer in einem Zyklus.
  - beide Folgen treffen sich in einem Knoten im Zyklus (hier 0).

# Hashkollisionssuche

## Eintritt in den Zyklus (Erweiterung von Floyd): Finde Eintrittspunkt



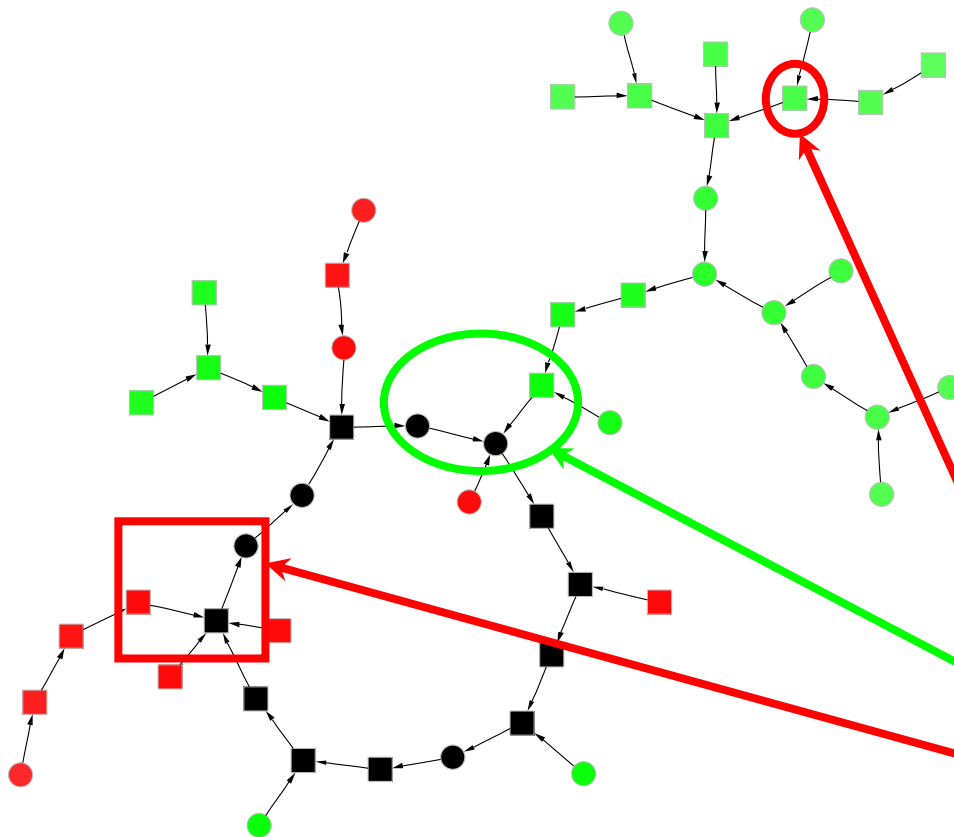
Schritt 2: Finden des Eintrittspunktes von Folge 1 in den Zyklus [25]:

- Folge 1 noch mal vom Startwert loslaufen lassen; dritte Folge mit Schrittweite 1 ab Treffpunkt im Zyklus (hier 0) loslaufen lassen.
- Es gilt: Die Folgen treffen sich im Eintrittspunkt (hier 25) der Folge 1.
- Die Vorgänger (hier 17 und 2) liefern die Hashkollision.

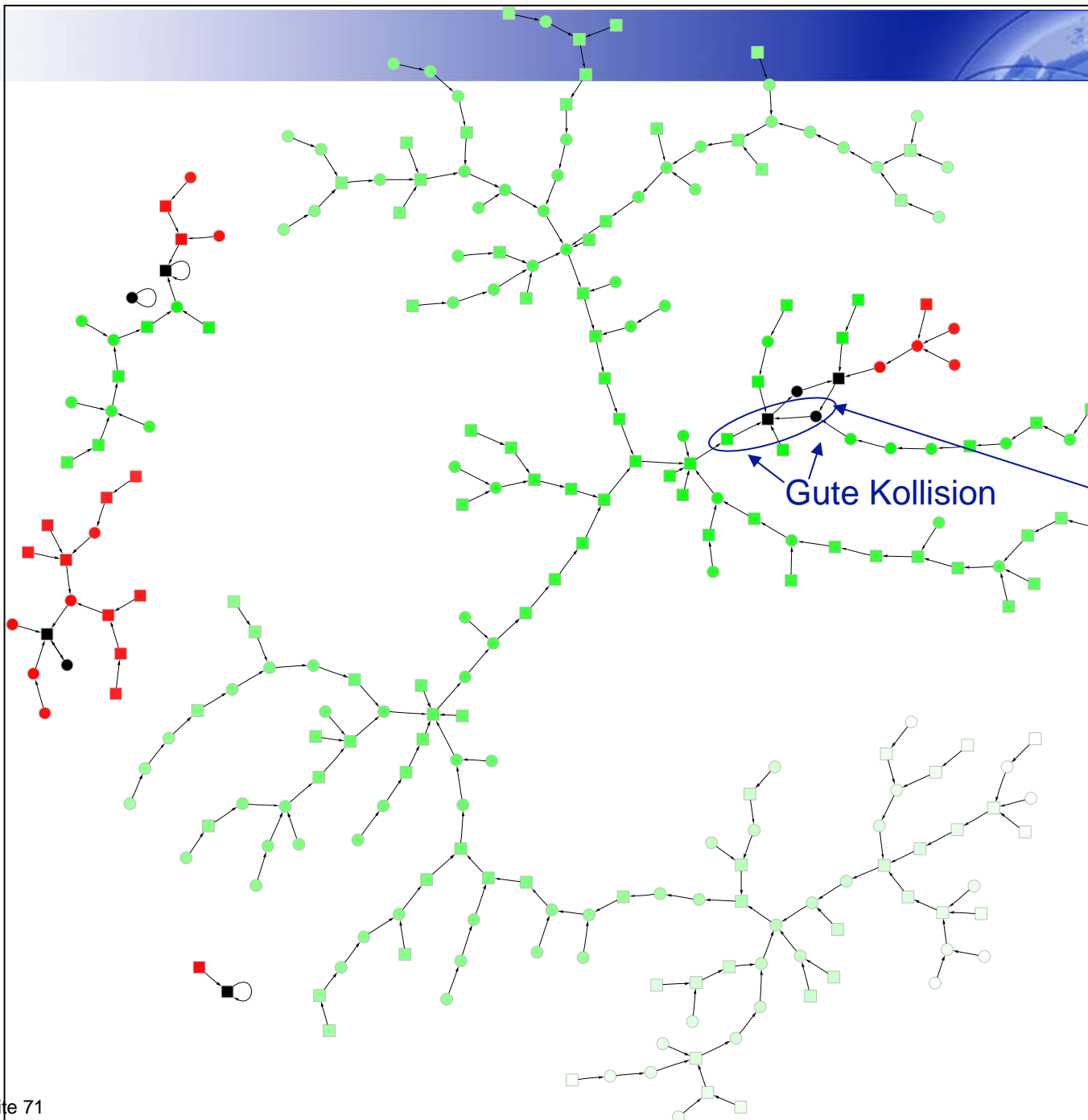
# Geburtsstagsangriff auf die digitale Signatur

## Auseinandersetzung mit dem Floyd-Algorithmus

- Visuelle & interaktive Darstellung des Floyd-Algorithmus ("Wanderung im Mapping" in einen Zyklus hinein).\*
- Adaption des Floyd-Algorithmus für den Signaturangriff.



\* Der Floyd-Algorithmus ist implementiert. Die Visualisierung von Floyd ist noch nicht in CrypTool integriert.



Ein Beispiel für ein **“gutartiges” Mapping** (fast alle Knoten darin sind grün gefärbt). In diesem Graphen gehören die meisten Knoten zu einem großen Baum, der in den Zyklus mit einem geraden Hashwert gelangt und wo der Eintrittspunkt-Vorgänger im Zyklus ungerade ist. D.h. der Angreifer findet für fast jeden zufälligen Startpunkt eine brauchbare Kollision.

# Anwendungsbeispiele (7)

## Angriff auf digitale Signatur: Angriff

The main window 'Angriff auf den Hashwert der digitalen Signatur' contains the following elements:

- 1. "Harmlose" Datei auswählen:** A text box with the path 'C:/Programme/CrypTool-medida/Original.txt' and a 'Suchen ...' button.
- 2. "Gefährliche" Datei auswählen:** A text box with the path 'C:/Programme/CrypTool-medida/Faelschung.txt' and a 'Suchen ...' button.
- Suche starten / Optionen festlegen:** A section with explanatory text and buttons for 'Nachrichtenpaar suchen', 'Optionen ...', and 'Finden'.

An arrow points from the 'Optionen ...' button to a secondary window titled 'Optionen für den Angriff auf den Hashwert der digitalen Signatur'.

This options window includes:

- Hashfunktion:** Radio buttons for MD2, MD4, **MD5** (selected), SHA, SHA-1, and RIPEMD-160.
- Signifikante Bitlänge:** A numeric input field set to 32.
- Optionen für die Nachrichtenmodifikation:** Checkboxes for 'Leerzeichen einfügen' and 'Zeichen anhängen'.
- Buttons for 'Übernehmen' and 'Standard wiederherstellen'.

Two progress windows titled 'Ein Nachrichtenpaar wird gesucht ...' are shown in the foreground:

- The top window shows 'Run 1 Zyklussuche' with 'Fortschritt: 23%' and 'Restzeit: 00:00:35'.
- The bottom window shows 'Run 1 Kollisionssuche' with 'Fortschritt: 27%' and 'Restzeit: 00:01:08', and an 'Abbrechen' button.



# Anwendungsbeispiele (7)

## Angriff auf digitale Signatur: Ergebnisse

Harmlose Nachricht: MD5, <4F 47 DF 1F>

Sehr geehrter Herr Einkaufsfern,

bitte bestellen Sie eine Schreibmaschine

MfG.  
Peter Gutermann  
AADBADCBACBACDADCB

MD5: 4F 47 DF 1F  
D2 DE CC BE 4B 52  
86 29 F7 A8 1A 9A

Gefährliche Nachricht: MD5, <4F 47 DF 1F>

Sehr geehrter Herr Einkaufsfern,

bitte bestellen Sie für Herrn Dieter Dieb ein Porsche und eine Tankkarte.

MfG.  
Peter Gutermann  
AAAABBDDBBAAABBC

MD5: 4F 47 DF 1F  
30 38 BB 6C AB 31  
B7 52 91 DC D2 70

Die ersten 32 Bit des Hashwertes sind gleich.

## Praktische Resultate

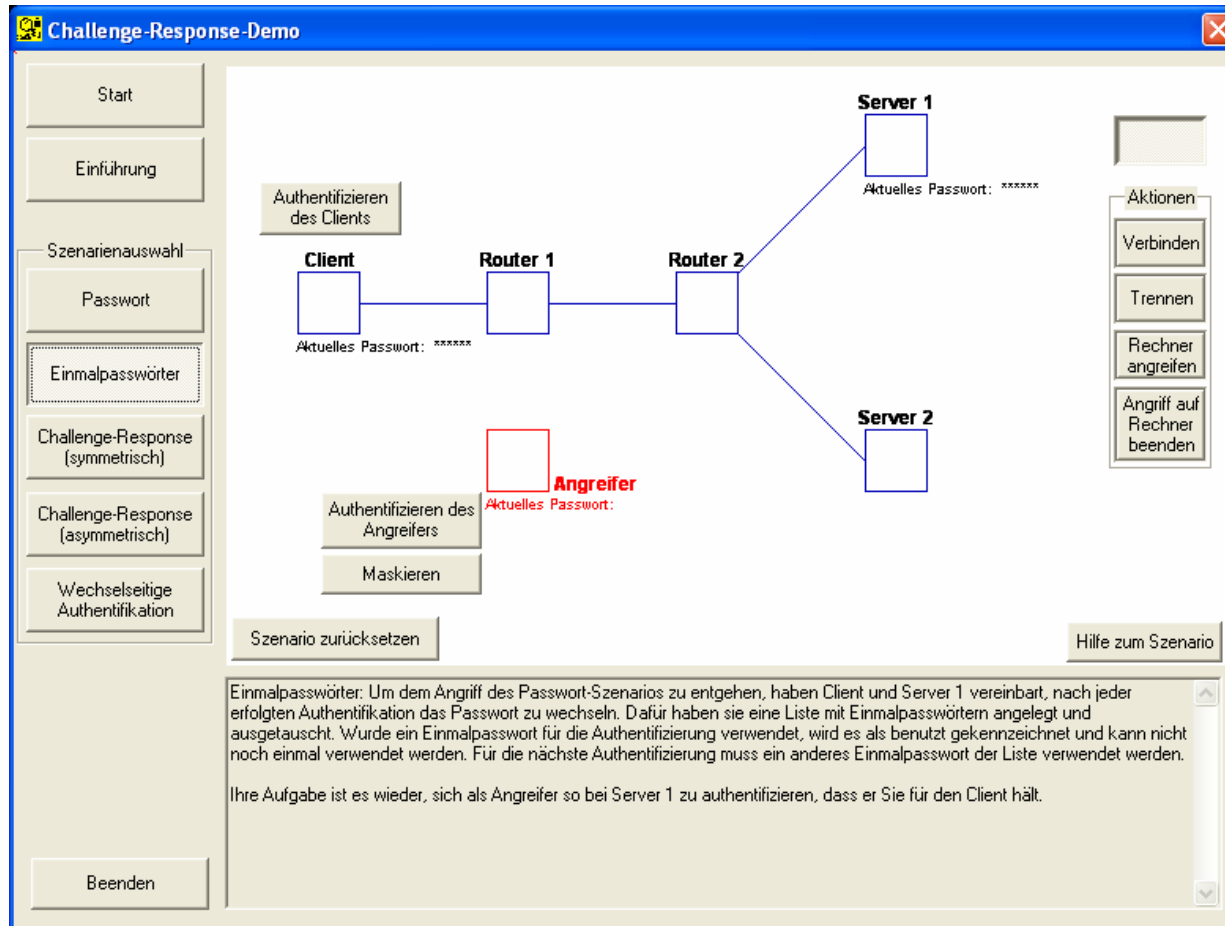
- 72 Bit *Teilkollisionen* (Übereinstimmung der ersten 72 Bit-Stellen der Hashwerte) konnten im Zeitraum von wenigen Tagen auf einem einzigen PC gefunden werden.
- Signaturverfahren mit Hashverfahren bis zu 128 Bit Länge sind heute gegenüber massiv parallelen Verfahren angreifbar!
- Es sollten Hashwerte mit mindestens 160 Bit verwendet werden.

## Zusätzlich zur interaktiven Bedienung:

Automatisierte Offline-Funktion in CrypTool: Durchspielen und Loggen der Ergebnisse für ganze Sets von Parameterkonfigurationen. Möglich durch entsprechenden Aufruf von CrypTool über die Eingabeaufforderung.

# Anwendungsbeispiele (8)

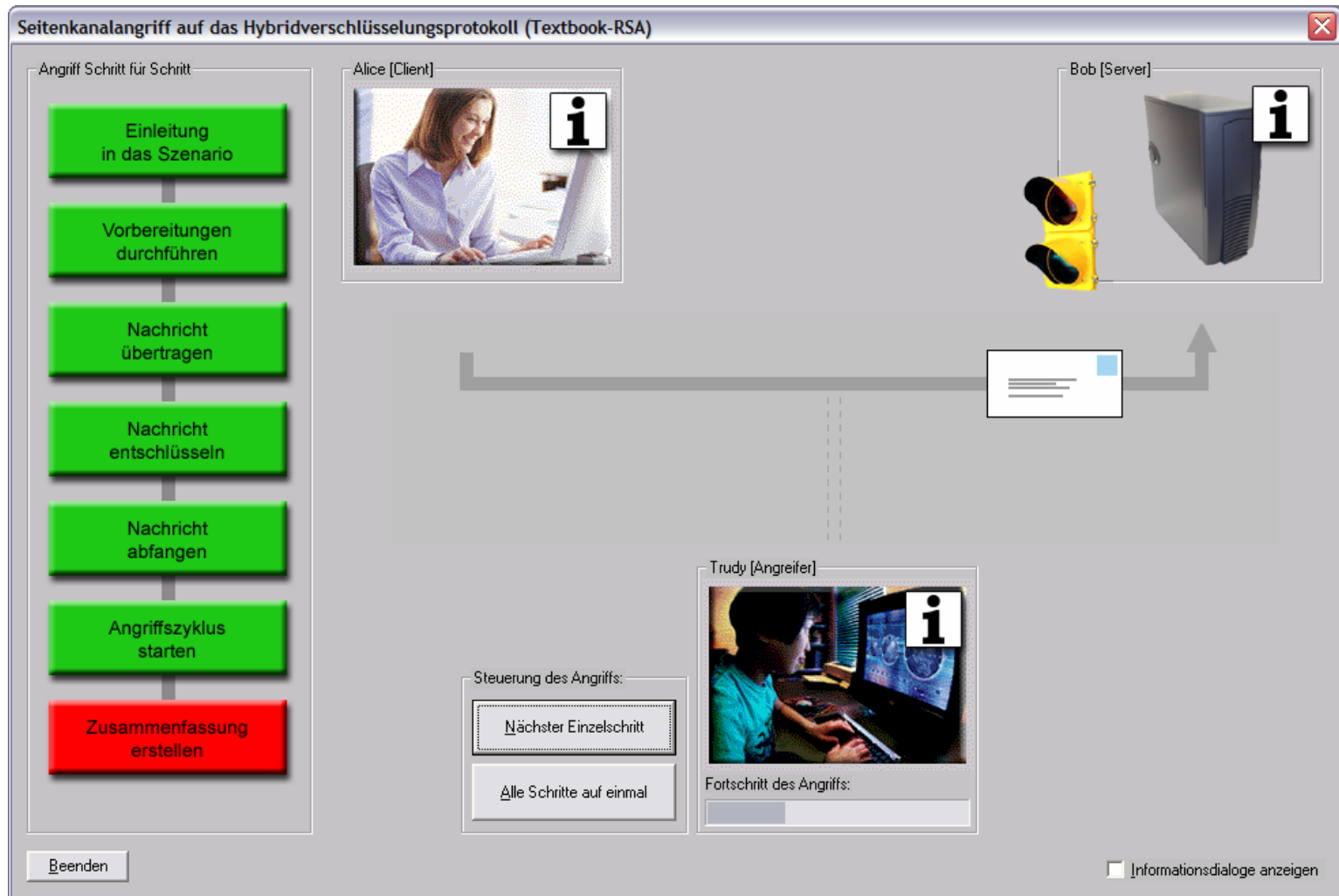
## Authentifizierung in einer Client-Server-Umgebung



- Interaktive Demo für verschiedene Authentifizierungs-Verfahren.
- Definierte Möglichkeiten des Angreifers.
- Sie können in die Rolle eines Angreifers schlüpfen.
- **Lerneffekt:** Nur die wechselseitige Authentifizierung ist sicher.

# Anwendungsbeispiele (9)

## Demo eines Seitenkanalangriffes (auf Hybridverschlüsselungsprotokoll)



# Anwendungsbeispiele (9)

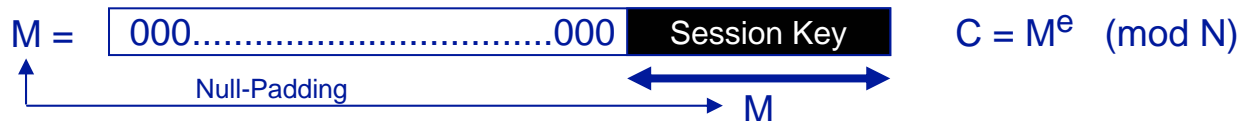
## Idee zu diesem Seitenkanalangriff

- Ulrich Kühn, *Side-channel attacks on textbook RSA and ElGamal encryption* (2003)

### Voraussetzungen:

- RSA-Verschlüsselung:  $C = M^e \pmod{N}$  und Entschlüsselung:  $M = C^d \pmod{N}$ .
- 128-Bit Sessionkeys (in M) werden „Textbuch-verschlüsselt“ (Null-Padding).
- Der Server kennt den geheimen Schlüssel d und
  - benutzt nach der Entschlüsselung nur die 128 niederwertigsten Bits (keine Überprüfung der Null-Padding Bits) (d.h. er erkennt nicht, wenn dort was anderes als Nullen stehen).
  - liefert eine Fehlermeldung, wenn bei der Entschlüsselung ein „falscher“ Session Key bestimmt wird (entschlüsselter Text kann nicht vom Server interpretiert werden). Im anderen Fall kommt keine Meldung.

Angriffs-idee: Approximation von Z auf 129 Bitstellen aus der Gleichung  $N = M * Z$  per  $M = \lfloor N/Z \rfloor$



Für Z werden die Bitstellen sukzessive ermittelt: Pro Schritt erhält man 1 Bit mehr. Der Angreifer modifiziert C nach C' (siehe unten). Abhängig davon, ob es beim Server (Empfänger) zu einem Bit-Überlauf bei der Berechnung von M' kommt, schickt er eine Fehlermeldung oder nicht. Basierend auf dieser Information erhält der Angreifer ein Bit für Z.



# Anwendungsbeispiele (10)

## Mathematik: Angriffe auf RSA per Gitterreduktion

**Angriff auf kleine geheime Exponenten (nach Blömer / May)**

**Beschreibung**  
Mit diesem Angriff ist es möglich, den RSA-Modul  $N$  zu faktorisieren, wenn der geheime Schlüssel  $d$  im Vergleich zu  $N$  zu klein gewählt wurde. Die Zahl  $\delta = \log(d)/\log(N)$  bezeichnet man als "Größe von  $d$ ". Der Angriff funktioniert für  $\delta < 0,290$ .

☐ Um Beispiele aus der Literatur auszuprobieren, geben Sie zunächst den öffentlichen Schlüssel  $(N, e)$  ein. Danach geben Sie einen geschätzten Wert für  $\delta$  ein. Alternativ können Sie  $d$  direkt eingeben, woraus  $\delta$  berechnet wird.

☒ Um sich ein Beispiel erzeugen zu lassen, geben Sie  $\delta$  und die Bitlänge von  $N$  an. Durch Klicken auf "Beispielschlüssel erzeugen" werden die Schlüssel erzeugt.

Danach klicken Sie auf "Starten".

**Schritt 1: Schlüsselparameter und Schlüssel eingeben**

Bitlänge von  $N$ :   $\delta$ :

$N$ :

$e$ :

$d$ :

**Schritt 2: Angriffsparameter für das Gitterreduktionsverfahren eingeben**

$m$ :  Bestimmt die Größe des zu reduzierenden Gitters und die maximale Größe von  $\delta$ . Sollte mindestens den Wert 4 haben.

$t$ :  Wird abhängig von  $m$  optimal bestimmt.

Gitterdimension:  Größe des zu reduzierenden Gitters. Bestimmt maßgeblich die Laufzeit.

Maximales  $\delta$ :  Maximale Größe von  $\delta$  für große  $N$  ( $N > 1000$  Bit).

**Schritt 3: Angriff starten**

Erzeuge Gitter:

Reduziere Gitter:  Reduktionen:

Bilde Resultante:  Resultanten:

Gesamtzeit:

Gefundene Faktorisierung:

$p$ :   $q$ :

- Veranschaulicht, wie die Parameter des RSA-Verfahrens beschaffen sein müssen, damit sie den aktuellen, auf Gitterreduktion beruhenden Angriffen aus der Literatur standhalten.

### ▪ Drei Varianten

1. Der geheime Exponent  $d$  ist im Verhältnis zu  $N$  zu klein.
  2. Einer der Faktoren von  $N$  ist teilweise bekannt.
  3. Ein Teil des Klartextes ist bekannt.
- Diese Annahmen sind realistisch.

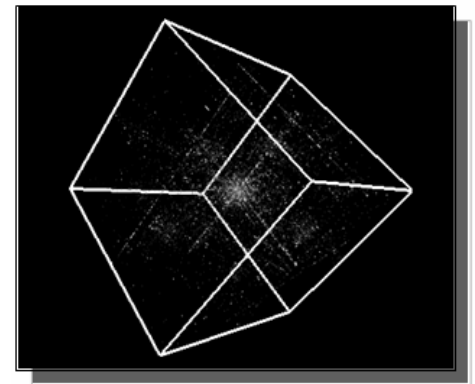
# Anwendungsbeispiele (11)

## Zufallsanalyse mit 3-D Visualisierung

### 3-D Visualisierung zur Analyse von Zufallszahlen

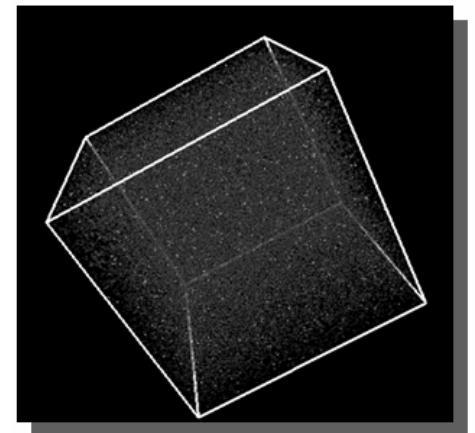
#### Beispiel 1

- Öffnen einer beliebigen Datei (z.B. Bericht in Word oder PowerPoint-Präsentation)
- Es empfiehlt sich eine zumindest 100 KB große Datei zu wählen
- 3-D Analyse über das Menü „Analyse“ \ „Zufallsanalyse“ \ „3-D Visualisierung...“
- Ergebnis: **Strukturen sind offensichtlich erkennbar**



#### Beispiel 2

- Generierung von Zufallszahlen („Einzelverfahren“ \ „Tools“ \ „Zufallsdaten erzeugen ...“)
- Hierbei sollte man zumindest 100.000 Bytes an Zufallsdaten erzeugen
- 3-D Analyse mit „Analyse“ \ „Zufallsanalyse“ \ „3-D Visualisierung...“
- Ergebnis: **Gleichverteilung (keine Strukturen erkennbar)**

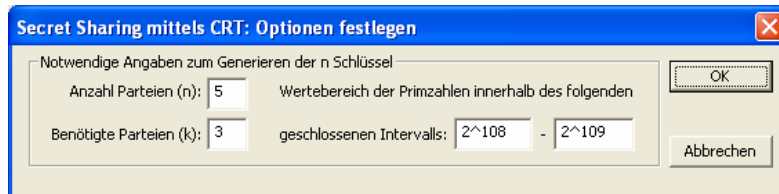


# Anwendungsbeispiele (12)

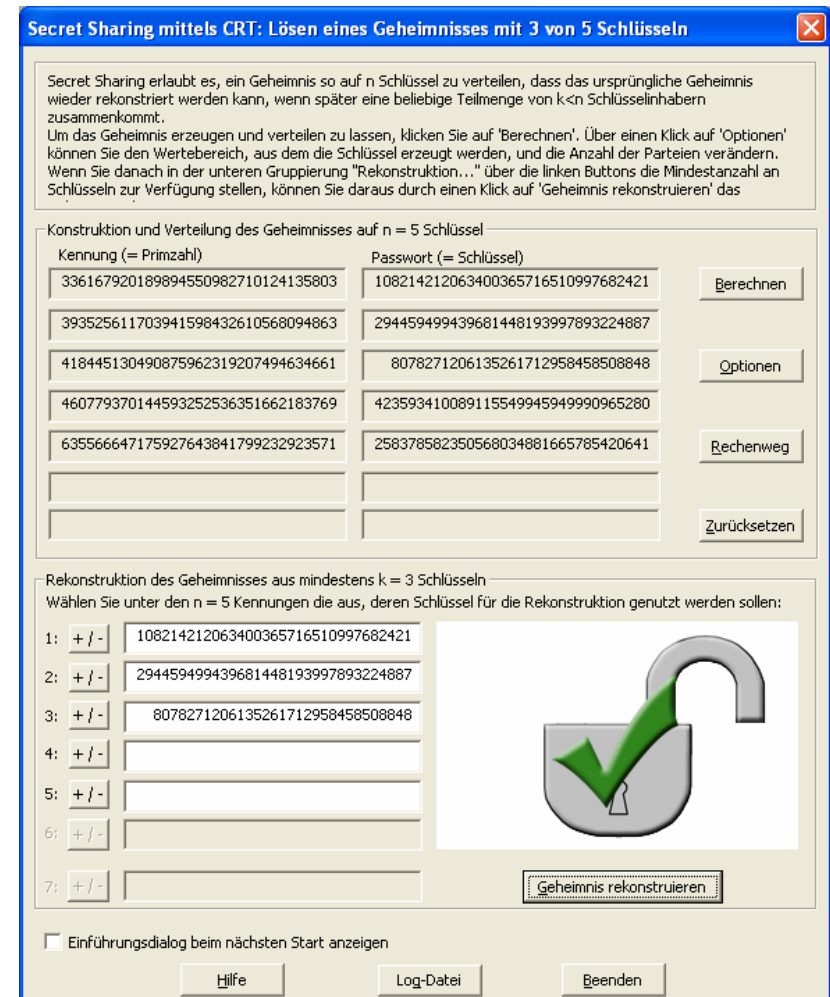
## Secret Sharing mittels CRT – Implementierung des Chinesischen Restsatzverfahrens

### Secret Sharing Beispiel (I):

- Problemstellung:
  - 5 Personen erhalten jeweils einen Schlüssel
  - Um Zugriff zu erlangen, müssen mindestens 3 der 5 Personen anwesend sein
- **CrypTool**: Menu „Einzelverfahren“ \ „Anwendungen des Chinesischen Restsatzverfahrens“ \ „Secret Sharing mittels CRT“
- „Optionen“ ermöglicht weitere Details des Verfahrens einzustellen.



- „Rechenweg“ zeigt die Schritte zur Generierung der Schlüssel.





# Anwendungsbeispiele (12)

## Secret Sharing mittels Schwellenwertschema von Shamir

### Secret Sharing Beispiel (II)

- Problemstellung
  - Ein geheimer Wert soll unter  $n$  Personen aufgeteilt werden.
  - $t$  von  $n$  Personen sind notwendig um den geheimen Wert wiederherzustellen.
  - $(t, n)$  Schwellenwertschema
- **CrypTool**: Menu „Einzelverfahren“ \ „Secret Sharing Demo (nach Shamir)...“
  1. Angabe des Geheimnisses  $K$ , sowie Anzahl der Teilnehmer  $n$  und Schwellenwert  $t$
  2. Polynom generieren
  3. Parameter übernehmen
- Mittels „**Rekonstruktion**“ kann das eingegebene Geheimnis wiederhergestellt werden

**Secret Sharing: Aufsetzen eines Schwellenwertschemas**

Mit dem  $(t, n)$ -Schwellenwertschema nach Shamir ist es möglich, ein Geheimnis  $K$  auf  $n$  Personen aufzuteilen. Danach sind  $t$  Personen ( $t \leq n$ ) in der Lage, mit ihren Teilgeheimnissen (Shares) das ursprüngliche Geheimnis wiederherzustellen. Dazu werden ein Polynom  $f(x)$  vom Grad  $t-1$  (also mit  $t-1$  zufälligen Koeffizienten  $a[i]$ ) und eine zufällige Primzahl  $p$  erzeugt. Jedem Teilnehmer werden dann ein zufällig gewählter, öffentlicher Wert  $x$  und ein geheimer Wert  $y=f(x)$  [sein Share] zugewiesen. Weitere Details erhalten Sie in der Online-Hilfe, indem Sie F1 drücken.

Geheimnis und Steuer-Parameter wählen

Eingabe des Geheimnisses  $K$  als Dezimalzahl:

Anzahl der Teilnehmer  $n$ :

Schwellenwert  $t$ :

Parameter des Polynoms  $f(x)$  vom Grad  $t-1$

Alle Operationen finden im diskreten Raum  $GF(p)$  statt.

Polynom  $f(x)$ :

Primzahl  $p$ :

Aus den Parametern berechnete Werte der Teilnehmer:

Teilnehmer	Öffentlicher Wert $x$	Share (geheimer Wert $f(x)$ )
Teilnehmer 1	4112	1828
Teilnehmer 2	407	2078
Teilnehmer 3	3486	4028
Teilnehmer 4	3325	1387
Teilnehmer 5	2282	3208
Teilnehmer 6	718	606
Teilnehmer 7	543	681
Teilnehmer 8	406	1668

Bitte wählen Sie die Teilnehmer aus der Liste aus, die das Geheimnis wiederherstellen sollen. Zur Mehrfachauswahl halten Sie einfach die Strg-Taste gedrückt und klicken den Eintrag mit Hilfe der Maus an.

☐ Informationsdialog zu Beginn anzeigen



# Anwendungsbeispiele (13)

## Anwendung des CRT in der Astronomie zur Lösung linearer Kongruenzsysteme

### Problemstellung aus der Astronomie

- Wie lange dauert es, bis sich eine gegebene Anzahl Planeten (mit unterschiedlichen Umlaufgeschwindigkeiten) auf einem Bahnradiusvektor  $s$  treffen.
- Ergebnis ist ein System simultaner Kongruenzen, das sich mit Hilfe des Chinesischen Restsatzes (CRT) lösen lässt.
- In dieser Demo können bis zu 9 Kongruenzen aufgestellt und mittels CRT gelöst werden.

**Anwendungsbeispiel des Chinesischen Restsatzes aus der Astronomie: Planetenumlaufbahn**

Mit dem Chinesischen Restsatz (CRT) kann man lineare modulare Gleichungssysteme lösen. Unten können Sie 9 Gleichungen der Form  $x = a[i] \bmod m[i]$  ( $i=1, \dots, 9$ ) eingeben und anschließend lösen. Solche Gleichungssysteme kann man z.B. nutzen, um herauszufinden, in wie viel Tagen bestimmte Planeten aufgereiht wie auf einer Perlenschnur hintereinander in einer Linie (Strahl) stehen.

Simultane Kongruenzen/ Modulare lineare Gleichungssysteme

$x \equiv$	15	mod	88
$x \equiv$		mod	
$x \equiv$	100	mod	365
$x \equiv$		mod	
$x \equiv$	0	mod	4327
$x \equiv$		mod	
$x \equiv$		mod	
$x \equiv$	0	mod	60149
$x \equiv$		mod	

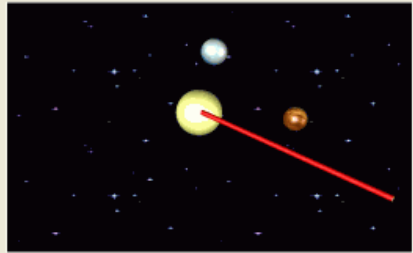
Lösung

126.228.390.655

Lösen Beenden

Löschen aller Parameter Zurücksetzen auf den Initialzustand

Anwendungsbeispiel aus der Astronomie / Film-Visualisierung ist fix



Die Umlaufzeiten der Planeten Merkur und Erde um die Sonne betragen 88 und 365 Tage. Bis zum Erreichen eines bestimmten Bahnradiusvektors  $s$  (rot) vergehen

15 und 100 Tage.

Kann es vorkommen, dass sich Merkur und Erde irgendwann einmal auf dem Strahl  $s$  befinden?

Planetenauswahl

<input checked="" type="checkbox"/> Merkur	<input type="checkbox"/> Mars	<input type="checkbox"/> Uranus
<input type="checkbox"/> Venus	<input checked="" type="checkbox"/> Jupiter	<input checked="" type="checkbox"/> Neptun
<input checked="" type="checkbox"/> Erde	<input type="checkbox"/> Saturn	<input type="checkbox"/> Pluto

In welchen Zeitabständen (Tagen) wiederholt sich das Ereignis?

8.359.702.902.760

# Anwendungsbeispiele (14)

## Visualisierung von symmetrischen Verschlüsselungsverfahren mit ANIMAL (1)

### Animierte Darstellung verschiedener symmetrischer Verfahren

- Caesar
- Vigenère
- Nihilist
- DES

### CrypTool

- „Einzelverfahren“ \ „Visualisierung von Algorithmen“ \ ...
- Steuerung der Animation über integrierte Steuerelemente

Animationsgeschwindigkeit      Skalierung der Darstellung

Caesar-Verschlüsselung

Bei der Caesar-Chiffre handelt es sich um ein klassisches Verschlüsselungsverfahren mit einem festen Rechtsschift um 3 auf dem normalen geordneten 26-Zeichen-Alphabet. Um wie viele Buchstaben man im Alphabet beim Rechtsschift voranschreitet, ist der Schlüssel. Dies war bei Caesar immer fest die Zahl 3.

G A L L I A E S T O M N I S D I V I S A ... Klartext

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 Klartextalphabet

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 Geheimtextalphabet

J D O ... Geheimtext

Das Geheimtextalphabet, dass Voraussetzung für die Verschlüsselung ist, erhält man auf folgende Weise:  
man schreibt unter jeden Buchstaben im Klartextalphabet den Buchstaben, der 3 Plätze weiter rechts davon steht.  
Bei der Verschlüsselung wird jeder einzelne Buchstabe der Klartext-Nachricht durch entsprechenden Buchstaben im Geheimtextalphabet ersetzt.

Steuerung der Animationsschritte (Vor, Zurück, Pause, etc.)

Schritt: 21

Direkte Anwahl eines Animationsschrittes

# Anwendungsbeispiele (14)

## Visualisierung von symmetrischen Verschlüsselungsverfahren mit ANIMAL (2)

### ■ Visualisierung der DES-Verschlüsselung

Animal Animation: unbenannt

0 200 400 600 800 1000 Speed 100%

[64bit] Eingabeblock X [64bit] Schlüssel K

Permutierte Eingabe

1	0	1	0	0	1	0	0
1	0	1	1	1	0	1	1
0	0	0	1	1	0	1	1
1	0	1	0	0	0	1	0
0	1	0	1	1	0	1	0
1	0	0	1	0	0	1	0
1	1	0	0	0	0	1	0
1	1	1	0	0	0	0	1

K'

0	1	1	1	1	1	1	1
0	0	1	1	0	0	0	0
1	1	1	1	0	0	0	1
1	1	1	0	0	0	1	1
0	1	1	1	1	0	0	0
0	1	0	0	0	0	0	1
0	1	0	0	0	0	1	1
0	0	1	1	0	0	0	1

PC2

14	17	11	24	1	5		
3	28	15	6	21	10		
23	19	12	4	26	8		
16	7	27	20	13	2		
41	52	31	37	47	55		
30	40	51	45	33	48		
44	49	39	56	34	53		
46	42	50	36	29	32		

K[1]

0	1	1	1	0	1
1	1	1	1	1	1
1	0	0	1	0	

Schritt: 169

Nach der Permutation des Eingabeblocks mit Hilfe des Initialisierungsvektors IV wird der Schlüssel K mit Hilfe von PC1 und PC2 permutiert.

Animal Animation: unbenannt

0 200 400 600 800 1000 Speed 100%

Die Funktion  $f$  :

110110 001010 110110 010100 000101 100110 101001 010011

B[1] B[2] B[3] B[4] B[5] B[6] B[7] B[8]

$10 = 1 \times 2^1 + 0 \times 2^0 = 2$   $1011 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 11$

S-Box 1:

Reihe \ Spalte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	3

S-Box 8:

Reihe \ Spalte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

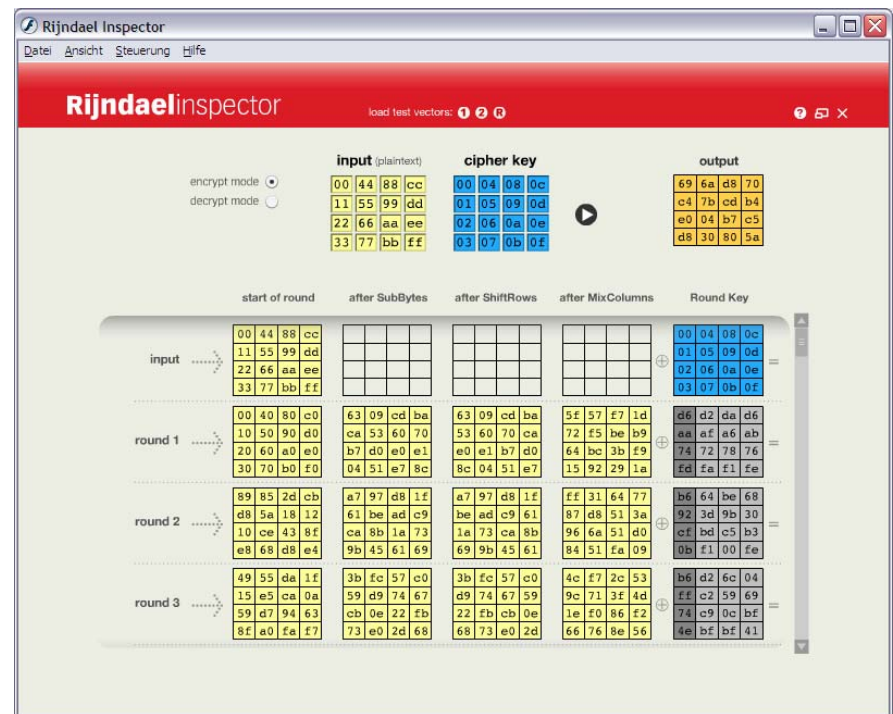
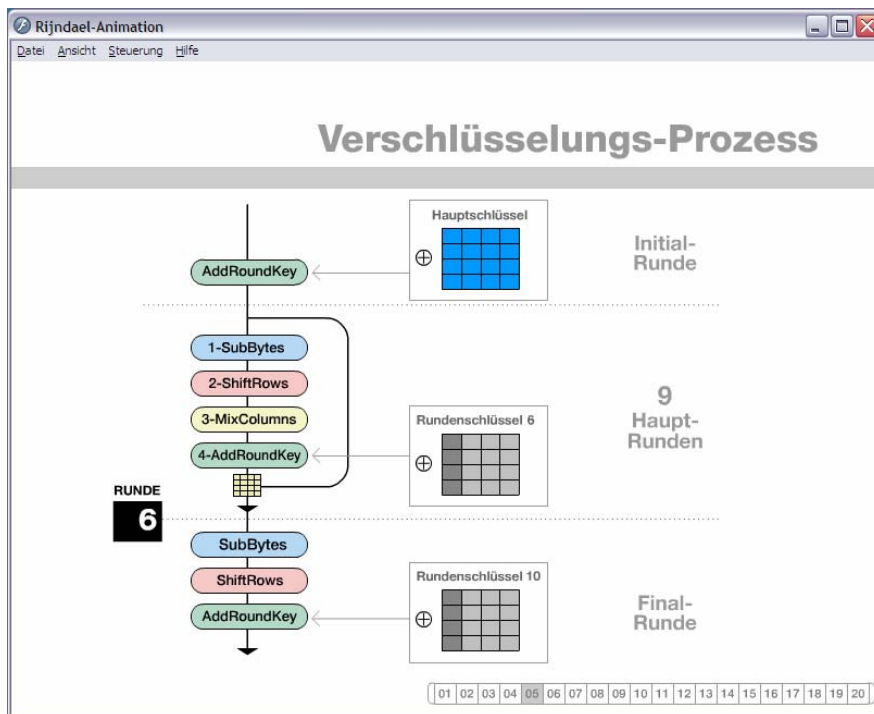
Schritt: 294

Die Kernfunktion  $f$  des DES, die die rechte Blockhälfte  $R_{i-1}$  mit dem Teilschlüssel  $K_i$  verknüpft.

# Anwendungsbeispiele (15)

## Visualisierung von AES (Rijndael-Cipher)

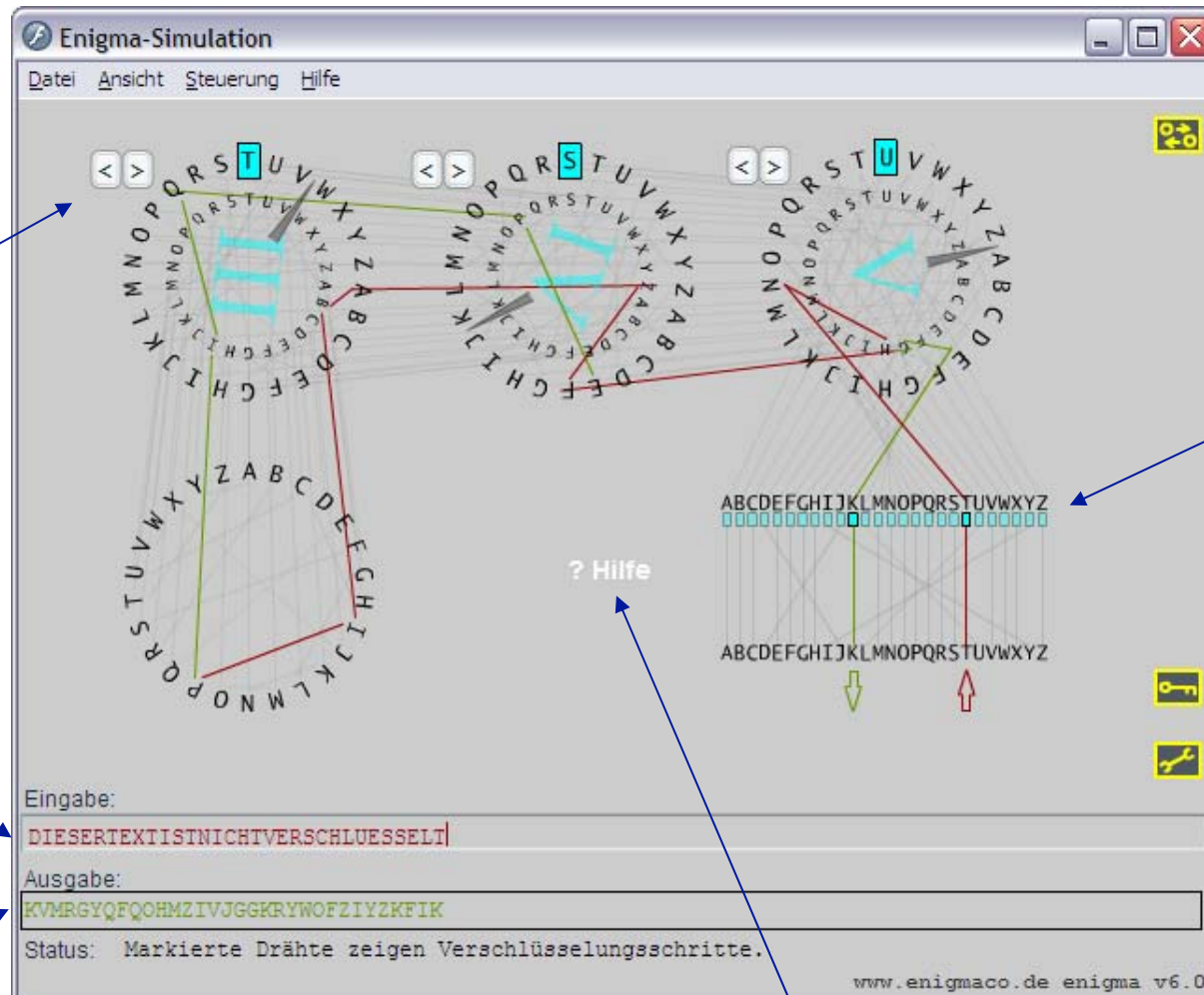
- **Rijndael-Animation** (die Rijndael-Chiffre war Gewinner der AES-Ausschreibung)
  - Visualisierung durch Animation des rundenbasierten Verschlüsselungsprozesses
- **Rijndael-Inspector**
  - Verschlüsselungsprozess zum Ausprobieren (mit selbst wählbaren Daten)



“Einzelverfahren” / “Visualisierung von Algorithmen” / “AES” / “Rijndael-Animation ...” bzw. “Rijndael-Inspector ...”

# Anwendungsbeispiele (16)

## Visualisierung der Enigma-Verschlüsselung



Rotorstellung  
verändern

Eingabe des  
zu Klartext

Ausgabe des  
verschlüsselten  
Textes

Auswahl  
der Rotoren

Setzen der  
Stecker

Anzeige der  
Einstellungen

Zufälliges  
Setzen und  
Zurücksetzen  
der Enigma

Klicken für weitere HTML-Hilfe

# Anwendungsbeispiele (17)

## Erzeugung eines Message Authentication Code

### Message Authentication Code (MAC)

- Gewährleistet:
  - Integritätsschutz der Nachricht
  - Authentizität der Nachricht
- Basis: Ein gemeinsamer Schlüssel für Sender und Empfänger
- Alternativ: Digitale Signatur

### Berechnung eines MAC in CrypTool

1. Auswahl der Hashfunktion
2. Auswahl der MAC-Variante
3. Angabe eines Schlüssels (je nach MAC-Variante auch zwei Schlüssel)
4. Erzeugung des MAC (automatisch)

The screenshot shows the 'Message Authentication Code' dialog box in CrypTool. It contains the following elements:

- Beschreibung:** A text area explaining that MACs help verify message integrity and authenticity using a shared key and a cryptographic hash function.
- Nachricht:** A text input field containing the message: 'CrypTool|||||CrypTool ist ein umfangreiches Lernprogramm zu den Themen Kryptographie und Kryptoanalyse.'
- Hashfunktion wählen:** A group box with radio buttons for MD2, MD4, MD5, SHA, SHA-1, and RIPEMD-160. **Annotation 1** points to this group.
- MAC-Variante (Position der Schlüssel; Verschachtelung):** A group box with radio buttons for H(k, m): vor der Nachricht, H(m, k): hinter der Nachricht, H(k, m, k): vor und hinter der Nachricht, H(k, H(k, m)): doppeltes Hashing, and H(k, m, k'): zwei verschiedene Schlüssel. **Annotation 2** points to this group.
- Geben Sie den Schlüssel (k) ein:** A text input field containing 'Chiffre'. **Annotation 3** points to this field.
- Geben Sie den zweiten Schlüssel (k') ein:** An empty text input field.
- Eingabe für äußere Hashfunktion (abhängig von der oben gewählten MAC-Variante):** A text input field containing 'ChiffreCrypTool'.
- Durch äußere Hashfunktion erzeugter Hashwert:** A text input field containing the hexadecimal string '9C E2 E9 8A 8F 13 F3 68 36 98 ED 6C 24 EA 90 6B'.
- Erzeugter MAC:** A text input field containing the hexadecimal string '36 1A 5D 05 F4 A5 BF F0 70 26 B8 F2 85 5A 86 23'. **Annotation 4** points to this field.
- Schließen:** A button at the bottom right.



# Anwendungsbeispiele (18)

## Hash-Demo

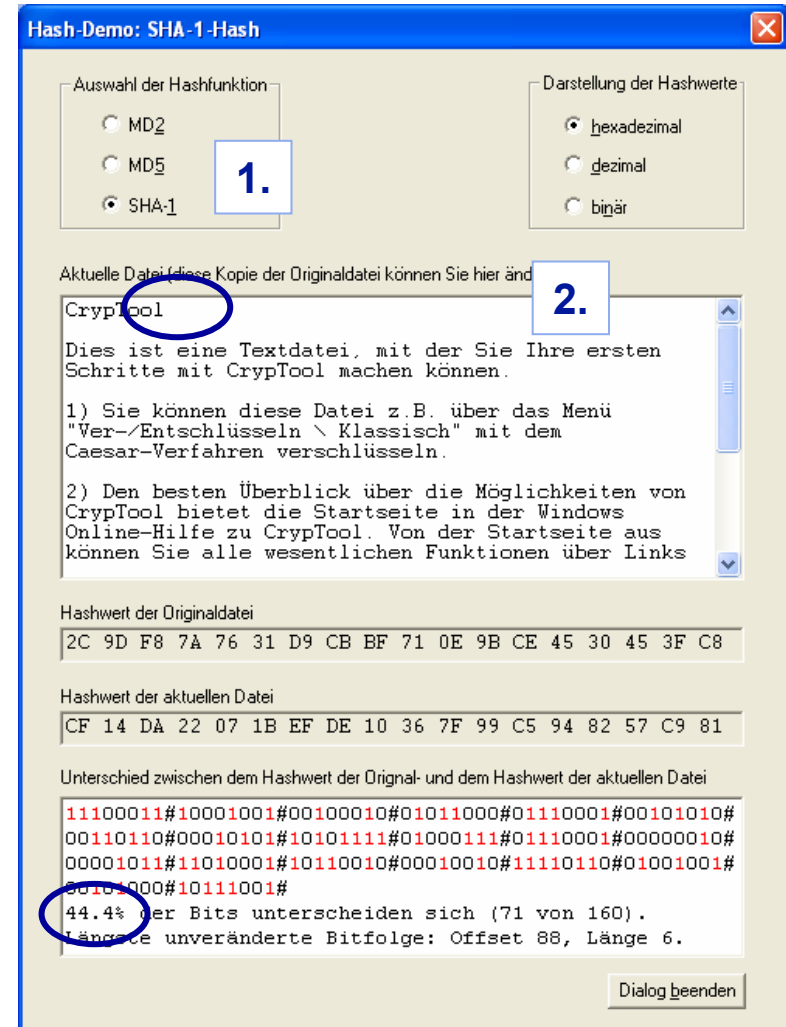
### Sensitivität von Hashfunktionen bei Änderungen des Originaltextes

1. Auswahl der Hashfunktion
2. Zusätzliches Einfügen von Zeichen im Text

#### Beispiel:

Die Eingabe eines zusätzlichen Leerzeichens hinter „CrypTool“ in der Originaldatei bewirkt eine 44,4%-ige Änderung der Bits des resultierenden Hashwertes.

Eine gute Hashfunktion sollte auf jede noch so kleine Änderung der Originaldatei möglichst sensitiv reagieren – „*Avalanche effect*“ (kleine Änderung, große Wirkung).



# Anwendungsbeispiele (19)

## Lernprogramm zur Zahlentheorie und zur asymmetrischen Verschlüsselung

- **Zahlentheorie**  
unterstützt durch mit  
graphischen  
Elementen und Tools  
zum Ausprobieren
- Themen:
  1. Ganze Zahlen
  2. Restklassen
  3. Primzahlerzeugung
  4. Asymmetrische Verschlüsselung
  5. Faktorisierung
  6. Diskrete Logarithmen

The screenshot shows a web-based learning program window titled 'ZT'. The window has a menu bar with 'Rechner', 'Navigation', 'Verzeichnisse', and 'Hilfe'. The main content area is green and displays the section '1.2 Primzahlen' (Page 5 of 21). The text includes definitions, theorems, and examples related to prime numbers.

**1.2 Primzahlen** Seite 5 von 21

**Definition:** Eine **Primzahl**  $p$  ist eine natürliche Zahl mit genau zwei positiven Teilern, nämlich 1 und  $p$  selbst. Die Menge aller Primzahlen heißt **IP**.

Beachten Sie, dass 1 keine Primzahl ist. Die Primzahlen  $< 100$  sind:  
2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 57, 59, 61, 67, 71, 73, 79, 83, 89, 97.

**Definition:** Teilt eine Primzahl  $p$  eine ganze Zahl  $a$ , dann heißt  $p$  **Primteiler** von  $a$ .

**Satz:** Jede natürliche Zahl  $n > 1$  hat mindestens einen Primteiler. [Beweis](#)

Daraus gewann schon [Euklid](#) Einsicht in die Anzahl der Primzahlen:

**Satz:** Es gibt unendlich viele Primzahlen. [Beweis](#)

**Definition:** Die natürlichen Zahlen zwischen den Primzahlen heißen **zusammengesetzte Zahlen**.

**Satz:** Jede zusammengesetzte Zahl  $n$  hat einen Primteiler  $p \leq \sqrt{n}$ . [Beweis](#)

Ist also eine natürliche Zahl  $n > 1$  durch keine Primzahl  $p \leq \sqrt{n}$  teilbar, dann ist  $n$  eine Primzahl.  
Beispiel: 89 ist Primzahl, denn 2, 3, 5, 7 teilen 89 nicht, und  $11 > \sqrt{89}$ .

Navigation icons: back, forward, search, and a link to the next page: (Go on to the next page.)



# Anwendungsbeispiele (20)

## Punktaddition auf elliptischen Kurven

- Visualisierung der Punktaddition auf elliptischen Kurven
- Grundlage der Elliptischen Kurven Kryptographie (ECC)

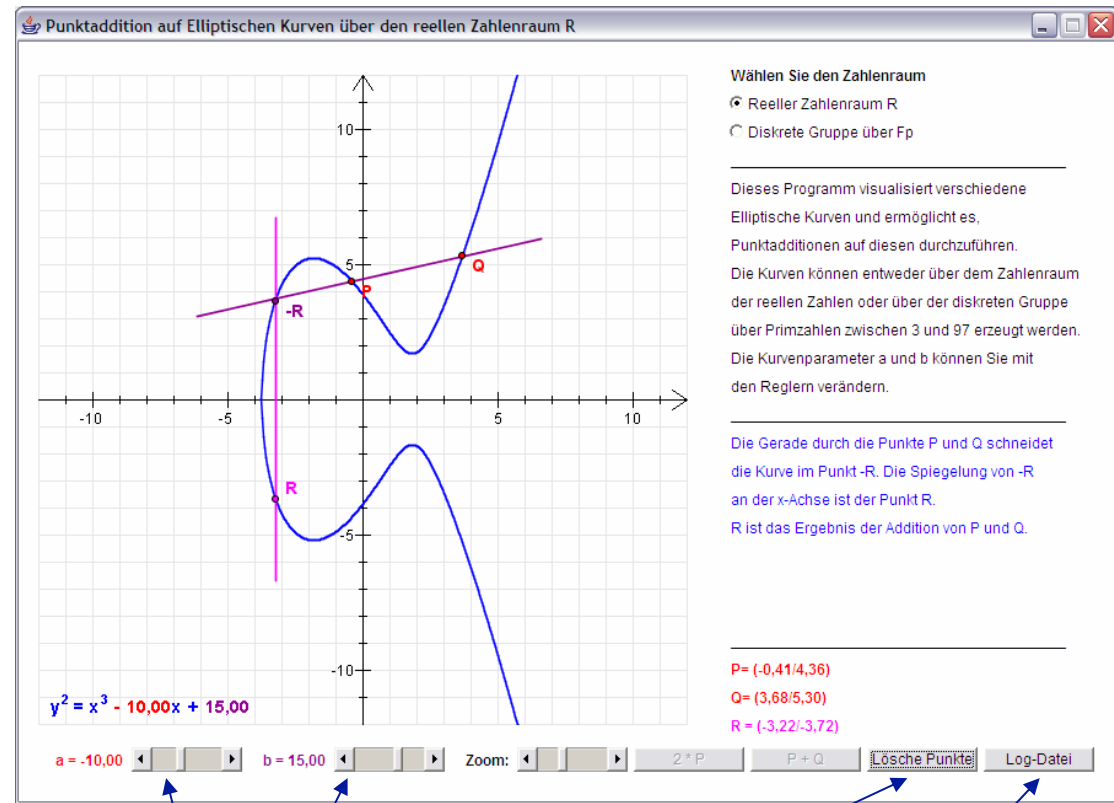
„Einzelverfahren“ /  
„Zahlentheorie - Interaktiv“ /  
„Punktaddition auf elliptischen Kurven...“

### Beispiel 1

- Punkt P auf der Kurve markieren
- Punkt Q auf der Kurve markieren
- Schaltfläche „P+Q“: Die Gerade durch P und Q schneidet die Kurve im Punkt -R / Spiegelung an der X-Achse R

### Beispiel 2

- Punkt P auf der Kurve markieren
- Schaltfläche „2\*P“: Die Tangente an P schneidet die Kurve in -R / Spiegelung an der X-Achse ergibt R



Änderung der elliptischen Kurve

Löschen der Punkte

Log der Berechnungen

# Anwendungsbeispiele (21)

## Passwort-Qualitätsmesser (PQM) I

- Messung der Qualität von Passwörtern
- Vergleich mit PQMs aus anderen Applikationen: KeePass, Mozilla und PGP
- Experimentelle Bewertung durch Cryptool-Algorithmus
- Beispiel: Eingabe eines Passwortes im Klartext

Passwort: 1234

The screenshot shows the 'Passwort-Qualitätsmesser' window. The 'Beschreibung' section explains that the tool estimates password security based on certain hypotheses. The 'Eingabe des Passworts' section shows the password '1234' entered in the text field, with a red sad face icon indicating low quality. The 'Qualität des Passworts' section shows a comparison with other tools: KeePass (10%), Mozilla (50%), PGP (9%), and Experimentell (14%). The 'OK' and 'Abbrechen' buttons are at the bottom.

Tool	Quality (%)
KeePass	10 %
Mozilla	50 %
PGP	9 %
Experimentell	14 %

Passwort: X40bTRds&11w\_dks

The screenshot shows the 'Passwort-Qualitätsmesser' window. The 'Beschreibung' section explains that the tool estimates password security based on certain hypotheses. The 'Eingabe des Passworts' section shows the password 'X40bTRds&11w\_dks' entered in the text field, with a yellow happy face icon indicating high quality. The 'Qualität des Passworts' section shows a comparison with other tools: KeePass (82%), Mozilla (100%), PGP (63%), and Experimentell (58%). The 'OK' and 'Abbrechen' buttons are at the bottom.

Tool	Quality (%)
KeePass	82 %
Mozilla	100 %
PGP	63 %
Experimentell	58 %

„Einzelverfahren“ / „Tools“ / „Passwort-Qualitätsmesser“

# Anwendungsbeispiele (21)

## Password-Qualitätsmesser (PQM) II

- Erkenntnisse des Passwort-Qualitätsmessers
  - Passwortqualität hängt primär von der **Länge des Passwortes** ab
  - Höhere Qualität des Passwortes durch die Verwendung von **verschiedenen Zeichenarten**: Groß-/Kleinschreibung, Zahlen und Sonderzeichen (**Passwortraum**)
  - **Passwortentropie** als Maß der Zufälligkeit der Wahl von Zeichen aus dem Passwortraum (je zufälliger die Wahl, desto besser das Passwort)
  - Passwörter sollten **nicht in einem Wörterbuch vorkommen** (Hinweis: Eine Wörterbuch-Überprüfung ist derzeit noch nicht in CrypTool implementiert).
- Qualität eines Passwortes aus Angreiferperspektive
  - Angriff auf ein Passwort (sofern beliebig viele Versuche zugelassen sind):
    1. Klassischer **Wörterbuchangriff**
    2. Wörterbuchangriff mit **weiteren Varianten** (z.B. 4-stellige Zahlen: [Sommer2007](#))
    3. **Brute-Force-Angriff** durch Test aller Kombinationen (ggf. mit Einschränkungen auf Zeichenarten)
  - ⇒ Ein gutes Passwort sollte so gewählt werden, dass es den Angriffen 1. und 2. standhält, im Hinblick auf 3. zumindest 8 Zeichen lang ist und Zahlen sowie Sonderzeichen beinhaltet.

# Anwendungsbeispiele (22)

## Brute-Force Analyse I

### Brute-Force Analyse

Optimierte Brute-Force Analyse unter der Annahme, dass ein Teil des Schlüssels bekannt ist.

### Beispiel – Analyse mit DES (ECB)

Versuch über Brute-Force den vollständigen Schlüssel zu finden, um den verschlüsselten Text zu entschlüsseln (Annahme: der Klartext ist ein Block aus 8 ASCII-Zeichen)

#### Schlüssel (Hex)

68ac78dd40bbefd\*  
0123456789ab\*\*\*\*  
98765432106\*\*\*\*\*  
0000000000\*\*\*\*\*  
000000000000\*\*\*\*  
abacadaba\*\*\*\*\*  
dddddddddd\*\*\*\*\*

#### Verschlüsselter Text (Hex)

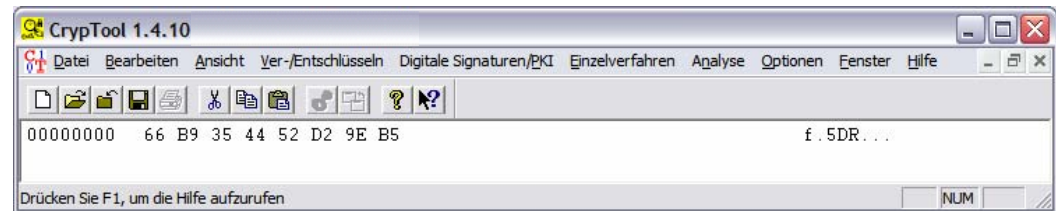
66b9354452d29eb5  
1f0dd05d8ed51583  
bcf9ebd1979ead6a  
8cf42d40e004a1d4  
0ed33fed7f46c585  
d6d8641bc4fb2478  
a2e66d852e175f5c

# Anwendungsbeispiele (22)

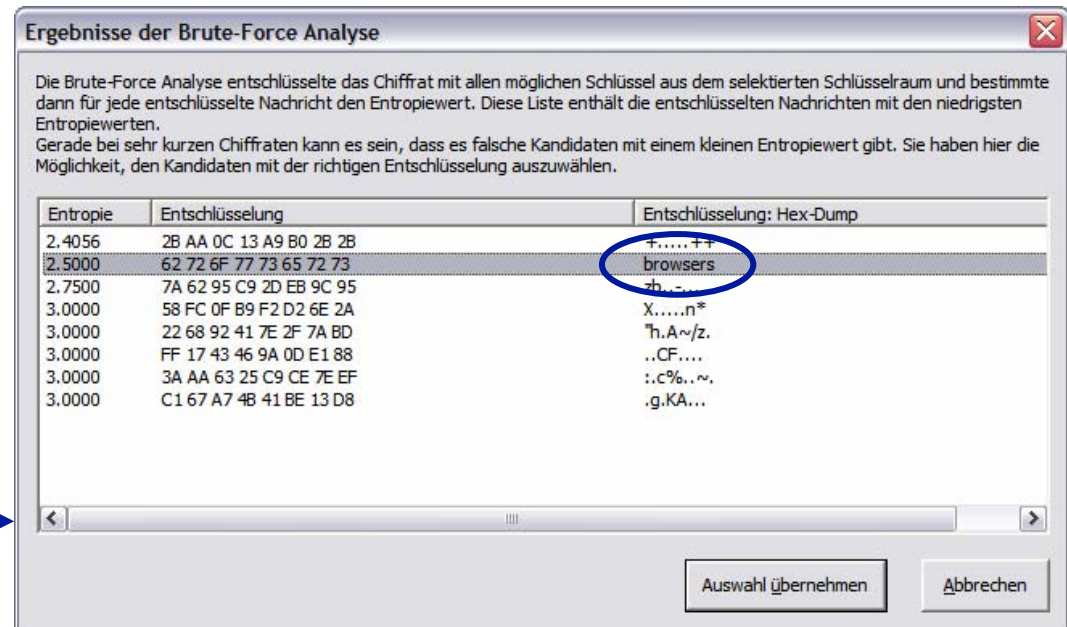
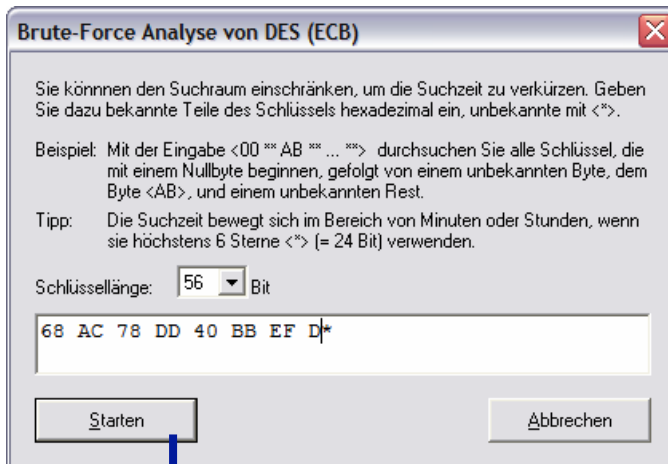
## Brute-Force Analyse II

1. Eingabe des verschlüsselten Textes
2. Verwendung der Brute-Force Analyse
3. Eingabe des teilweise bekannten Schlüssels
4. Start der Brute-Force Analyse
5. Analyse der Ergebnisse: Kleine Entropie deutet auf eine mögliche Entschlüsselung. Allerdings hat bei diesem Beispiel aufgrund des kurzen Textes der richtige Kandidat nicht die kleinste Entropie.

“Ansicht” \ “Als HexDump anzeigen”



“Analyse” \ “Symmetrische Verfahren (modern)” \ “DES (ECB)...”





# Anwendungsbeispiele (23)

## CrypTool Online-Hilfe I

Hilfe zu CrypTool 1.4.10 beta 4

Ausblenden Zurück Vorwärts Abbrechen Aktualisieren Startseite Drucken Optionen

Inhalt Index Suchen

Zu suchendes Schlüsselwort:

gitter

Gitterreduktion

Gleichverteilung

Gleitende Häufigkeit

GMP

GnuPG

GPL/GNU

Grafikoptionen

Haftungsausschlussklausel

Hashfunktion

Hashwert einer Datei

Hexadezimale Ein-/Ausgaben

Hilfe

Hill-Verschlüsselungsverfahren

Hin- und Rücktransformation

Histogramm

Historie

Homophone Substitution

Hybrid

Hybridentschlüsselung

Hybridverschlüsselung

Hypothese H0

IDEA-Verschlüsselungsverfahren

Importieren (Zertifikat) / PKCS #12

Initialisierungsdatei

Interoperabilität

Inverse Permutation

invertorisch

ISIS / MTT

Kartenspiel

Kaskade

Kaskadierende Chiffre

Klartext

Known-Plaintext-Angriff

Komprimieren

Konfigurationsdatei

Kongruenzgenerator

Kontakt

Korrelation

Kryptoanalyse

Krypto-Bibliotheken

Kryptographie

Kryptologie

Anzeigen

Menü Gitterbasierte Angriffe auf RSA (Menü [Einzelverfahren](#) \ RSA-Kryptosystem)

Das Menü **Gitterbasierte Angriffe auf RSA** enthält folgende Befehle:

[Faktorisieren mit teilweise bekanntem p](#)

Angriff auf RSA mit Gitterreduktionsverfahren, sofern ein Teil eines der beiden Primfaktoren von N bekannt ist.

[Angriff auf stereotype Nachrichten](#)

Angriff auf RSA mit Gitterreduktionsverfahren, sofern ein Teil des ursprünglichen Klartextes einer abgefangenen Nachricht bekannt und e klein ist.

[Angriff auf kleine geheime Schlüssel](#)

Angriff auf RSA mit Gitterreduktionsverfahren, sofern d im Verhältnis zu N zu klein gewählt ist.

Allen hier vorgestellten gitterbasierten Angriffen liegt ein ähnlicher Ansatz zu Grunde: Zunächst wird das Problem, RSA zu brechen in die Suche nach Nullstellen eines Polynoms modulo einer ganzen Zahl (meist N) umgeformt. Eine solche Nullstelle zu finden ist ein schwieriges Problem.

Um die Nullstellen zu berechnen werden zu dem ursprünglichen Polynom noch eine Reihe weiterer Polynome aufgestellt, von denen bekannt ist, dass sie die selbe Nullstelle besitzen. Aus den Koeffizienten dieser Polynome wird eine Gitterbasis aufgestellt. Diese wird mit dann z.B. mit dem LLL-Algorithmus reduziert, um einen kurzen Vektor zu finden.

Aus dem gefundenen kurzen Vektor wird wieder ein Polynom aufgestellt. Man kann zeigen, dass wenn der gefundene Vektor kurz genug ist, das aufgestellte Polynom die gesuchte Nullstelle nicht nur modulo N sondern über allen Zahlen besitzt.

**Beispiel:**

Das Polynom  $q_1(x) = 3x+1$  hat eine Nullstelle  $x_0$  modulo 7. Es sei bekannt, dass das Polynom  $q_2(x) = 4x-1$  die gleiche Nullstelle  $x_0$  modulo 7 besitzt. Aus den Polynomen werden nun die Vektoren  $b_1 = [3 \ 1]$  und  $b_2 = [4 \ -1]$  aufgestellt. Alle ganzzahligen Linearkombinationen dieser Vektoren stellen Punkte in einem Gitter dar. Die Abbildung links zeigt einen Ausschnitt dieses Gitters. Jeder Gitterpunkt kann wiederum als Polynom interpretiert werden, das ebenfalls die gesuchte Nullstelle besitzt. Ein kurzer Vektor des Gitters ist  $b_3 = [1 \ -2]$  aus dem das Polynom  $h(x) = x-2$  gebildet wird. Dieses Polynom hat in  $x_0=2$  eine Nullstelle sowohl über den ganzen Zahlen als auch über den ganzen Zahlen modulo 7. Man erkennt, dass  $x_0=2$  auch Nullstelle der Polynome  $q_1(x)$  und  $q_2(x)$  modulo 7 ist. ( $3x_0+1=7$ , 7 modulo 7 = 0)

**Bemerkung:**

# Anwendungsbeispiele (23)

## CrypTool Online-Hilfe II

Hilfe zu CrypTool 1.4.10 beta 4

Ausblenden Zurück Vorwärts Abbrechen Aktualisieren Startseite Drucken Optionen

Inhalt Index Suchen

Zu suchendes Schlüsselwort:

base

- Authentisierungsverfahren im Netz
- Authentizität
- Autokorrelation
- Autoren
- Autorisierung
- Balkendiagramm
- Base64-Codierung
- BC
- Bearbeiten
- Beenden
- Beispiele/Szenarien
- Binäres exklusives Oder
- Bitlänge
- Blöcke
- Brute-Force-Angriff
- Bücher
- Byteweise Addition
- Caesar-Verschlüsselungsverfahren
- CBC-Modus
- Challenge
- Challenge-Response-Demo
- CHI\*2-Verteilung
- Chiffre
- Chiffretext
- Chinesischer Restsatz (CRT)
- Chosen-Plaintext-Angriff
- Ciphertext
- Ciphertext-Only-Angriff
- Codierungen
- Copyright
- CrypTool
- cv cryptovision
- Datei
- Default-Settings
- Demonstrationen/Szenarien
- DES-Verschlüsselungsverfahren
- DES-Verschlüsselungsverfahren
- Dialog (Übersicht aller Dialoge)
- Dialog der Schwestern
- Diffie-Hellman-Schlüsselaustauschverfahren
- Digitale Signaturen/PKI
- Digitales Zertifikat

Anzeigen

### Vergleich von Base64- und UU-Codierung

Die Codierung bei [Base64](#) und [UUencode](#) ist sehr ähnlich, was nachstehendes Schaubild zeigt:

1. Schritt: Aufteilung des Datenstromes - bei beiden Verfahren gleich.

2. Schritt: Darstellung der 6-bit Werte - unterschiedliche Methoden.

Base64

UUencode

Aufteilen von 3 x 8 Bit in 4 x 6 Bit

Byte 1						Byte 2						Byte 3											
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Zeichen 1						Zeichen 2						Zeichen 3						Zeichen 4					

Wert aus Base64-Codetabelle holen. (per Standard definiert)

Wert, vermehrt um dezimal 32, aus ASCII-Zeichentabelle auslesen.

Durch die ähnliche Art der Codierung weisen die Codierungen auch gemeinsame Vor- und Nachteile auf:

Vorteile	Nachteile
<ul style="list-style-type: none"><li>• Beliebige Binärdaten können mit einem 6-bit Zeichensatz dargestellt werden:<ul style="list-style-type: none"><li>◦ Keine Probleme mit 7-bit Zeichensatzbeschränkungen.</li><li>◦ Keine Probleme mit Zeilenlängenbeschränkungen oder</li></ul></li></ul>	<ul style="list-style-type: none"><li>• Keine Unterstützung für die Aufteilung von großen Dateien.</li><li>• Vergrößerung der Dateien um ca. 33 % (im Vergleich zur Originaldatei). Nur UUencode:</li></ul>



# Inhalt

- I. CrypTool und Kryptographie – Überblick
- II. Was bietet CrypTool?
- III. Ausgewählte Beispiele
- IV. **Projekt / Ausblick / Kontakt**



# Weiterentwicklung

## Geplant nach 1.4.10 (vgl. Readme-Datei)

- Massenmustersuche
- Visualisierung der Interoperabilität von S/MIME- und OpenPGP-Formaten
- Visualisierung des SSL-Protokolls
- Demo zur Visuellen Kryptographie
- Einbindung der Krypto-Bibliothek „Crypto++“ von Wei Dai
- Demo von Bleichenbachers Angriff auf RSA-Signaturen
- Demo virtueller Kreditkartennummern als Ansatz gegen Kreditkartenmissbrauch

## In Arbeit (vgl. Readme-Datei)

- Direkte Portierung der bisherigen C++-Version nach Linux mit Qt4 (siehe: <http://www.cryptoolinux.net/>)
- Portierung und Neudesign von CrypTool in Java / SWT / Eclipse / RPC (siehe: <http://jcryptool.sourceforge.net/>)
- Portierung und Neudesign der C++-Version mit C# / WPF/Vista / VS2005 / .NET (direkter Nachfolger des aktuellen Releases: Treeview statt Menüs, ...)

## Angedacht (vgl. Readme-Datei)

- Visualisierung von Protokollabläufen (z.B. Kerberos)
- Visualisierung von Angriffen auf diese Protokollabläufe
- Erstellung einer Kommandozeilenversion für Batch-Steuerung
- Weitere Parametrisierung / Flexibilisierung der vorhandenen Verfahren
- Grafik-Design-orientierter Modus für Einsteiger plus Expertenmodus

# CrypTool als Framework für eigene Arbeiten nutzen

## Angebot

- Man kann auf einem umfassenden Set aus Algorithmen, inkludierten Bibliotheken und Oberflächenelementen aufsetzen (Re-Use)
- Kostenlose Schulung in Frankfurt, wie man in die CrypTool-Programmierung einsteigt
- Vorteil: Der eigene Code aus Seminar-, Diplom- und Doktorarbeiten „verschwindet“ nicht, sondern wird weitergepflegt.

## Aktuelle Entwicklungsumgebung: **Microsoft Visual Studio C++ , Perl, Subversion Source-Code Management**

- Bis CrypTool 1.3.05: nur Visual C++ 6.0 (gab es als Buchbeilage kostenlos)
- Bis CrypTool 1.4.10: Visual C++ .net (= VC++ 7.1)(= Visual Studio 2003)
- Beschreibung für Entwickler: siehe readme-source.txt
- Download: Sourcen und Binaries der Release-Versionen  
Interessierte und Entwickler erhalten auch die Sourcen der aktuellen Betas.

## Zukünftige Entwicklungsumgebungen

- Für Versionen nach 1.4.10:
  - C# Version: .NET mit Visual Studio 2005 Express Edition (kostenlos), WPF (ohne MFC) und Perl
  - Java Version: mit Eclipse 3.2, SWT, RCP (kostenlos)
  - C++ Version für Linux mit Qt 4.x, GCC 4.0 und Perl

# CrypTool – Bitte um Mitwirkung

- **Wir freuen uns über jede weitere Mitarbeit**
  - Feedback, Kritik, Anregungen und Ideen
  - Einbau weiterer Algorithmen, Protokolle, Analysen (Konsistenz und Vollständigkeit)
  - Mithilfe bei der Entwicklung (Programmierung, Layout, Übersetzung, Test, Webseiten-Erweiterung)
    - Sowohl im bisherigen C/C++ Projekt als auch in den neuen Projekten für „CrypTool 2.0“: Java-Projekt und C#-Projekt !
    - Insbesondere Lehrstühle, die CrypTool zur Ausbildung verwenden, sind herzlich eingeladen, zur Weiterentwicklung beizutragen.
  - Signifikante Beiträge können namentlich erwähnt werden (in der Hilfe, Readme, About-Dialog und auf der Webseite).
  - Derzeit wird das gesamte Programmpaket etwa 3.000 mal pro Monat herunter geladen (davon etwas mehr als 1/3 die englische Version).

# Kontaktadresse

**Bernhard Esslinger**

**Universität Siegen  
Dozent, Fachbereich 5 Wirtschaftswissenschaften, Wirtschaftsinformatik**

**Deutsche Bank AG  
Direktor, IT-Security Manager**

**esslinger@fb5.uni-siegen.de**

**[www.cryptool.com](http://www.cryptool.com)**

**[www.cryptool.de](http://www.cryptool.de)**

**[www.cryptool.org](http://www.cryptool.org)**

**[www.cryptool.pl](http://www.cryptool.pl)**

**Weitere Kontaktadressen: siehe Readme im CrypTool-Programmpaket  
Mailing list: [cryptool-list@sec.informatik.tu-darmstadt.de](mailto:cryptool-list@sec.informatik.tu-darmstadt.de)**

# Weitere Lektüre (auch als Einstieg in die Kryptologie)

- Simon Singh, *“Geheime Botschaften”*, 2000, Hanser [in Deutsch]
- Simon Singh, *“The Codebook”*, 1999, Doubleday [Englisches Original]
- Udo Ulfkotte, *“Wirtschaftsspionage”*, 2001, Goldmann
- Claudia Eckert, *“IT-Sicherheit”*, 3. Auflage, 2004, Oldenbourg
- A. Beutelspacher / J. Schwenk / K.-D. Wolfenstetter, *“Moderne Verfahren der Kryptographie”*, 5. Auflage, 2004, Vieweg
- [HAC] Menezes, van Oorschot, Vanstone, *“Handbook of Applied Cryptography”*, 1996, CRC Press
- van Oorschot, Wiener, *“Parallel Collision Search with Application to Hash Functions and Discrete Logarithms”*, 1994
- **Vielfältige Krypto-Literatur** – siehe Links auf der CrypTool-Webseite sowie Quellenangaben in der Online-Hilfe von CrypTool (z.B. Bücher von Wätjen, Buchmann, Salomaa, Brands, Schneier, Shoup, Stamp/Low, ...)
- **Bedeutung der Kryptographie in dem breiteren Rahmen von IT-Sicherheit, Risikomanagement und organisatorischen Kontrollen**
  - Siehe z.B. Kenneth C. Laudon / Jane P. Laudon / Detlef Schoder, *“Wirtschaftsinformatik”*, 2005, Pearson, Kapitel 14
  - Siehe Wikipedia (<http://de.wikipedia.org/wiki/Risikomanagement>)