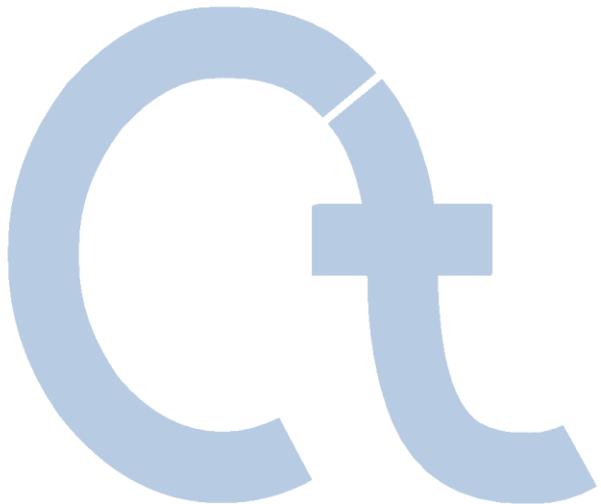


KRYPTOLOGIE MIT CRYPTOTool 1



Praktische Einführung in
Kryptographie und Kryptoanalyse

Umfang, Technik und Zukunft von CrypTool 1.4.xx

Prof. Bernhard Esslinger und CrypTool-Team
(Update: 18. September 2017 mit Release CT 1.4.40)

www.cryptool.org

Übersicht (I)

I. CrypTool und Kryptologie – Überblick

1. Definition und Bedeutung der Kryptologie
2. Das CrypTool-Projekt
3. Beispiele klassischer Verschlüsselungsverfahren
4. Erkenntnisse aus der Entwicklung der Kryptographie

II. Was bietet CrypTool 1?

1. Überblick
2. Beispiele zur Interaktion
3. Herausforderungen für Entwickler

III. Ausgewählte Beispiele

1. RSA-Verschlüsselung / Primzahltests / 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 Suche nach Hashkollisionen
8. Authentisierung in einer Client-Server-Umgebung
9. Demonstration eines Seitenkanalangriffs (auf ein Hybridverschlüsselungsprotokoll)

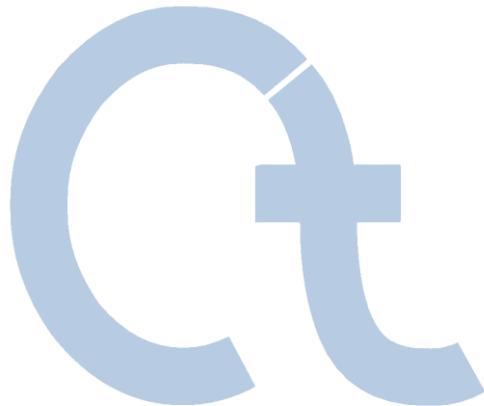
(...)

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 \(CRT\) und nach Shamir](#)
13. [Anwendung des CRT in der Astronomie \(Lösung linearer Kongruenzsysteme\)](#)
14. [Visualisierung von symmetrischen Verschlüsselungsverfahren mit ANIMAL](#)
15. [Visualisierungen von AES](#)
16. [Visualisierung der Enigma-Verschlüsselung](#)
17. [Visualisierung von sicherer E-Mail mit S/MIME](#)
18. [Erzeugung eines Message Authentication Code \(HMAC\)](#)
19. [Hash-Demo](#)
20. [Lernprogramm zur Zahlentheorie und zur asymmetrischer Verschlüsselung](#)
21. [Punktaddition auf elliptischen Kurven](#)
22. [Passwort-Qualitätsmesser und Passwort-Entropie](#)
23. [Brute-Force-Analyse](#)
24. [Skytale / Gartenzaun](#)
25. [Hill-Verschlüsselung / Hill-Analyse](#)
26. [CrypTool Online-Hilfe / Menübaum zum Programm](#)

IV. Projekt / Ausblick / Kontakt



- I. CrypTool und Kryptologie – Überblick
 - II. Was bietet CrypTool 1?
 - III. Ausgewählte Beispiele
 - IV. Projekt / Ausblick / Kontakt
- Anhang

Bedeutung der Kryptographie

Einsatzbeispiele für Kryptographie

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

- 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!

Definition Kryptologie und Kryptographie

Kryptologie (vom Griechischen *kryptós*, "versteckt," und *lógos*, "Wort") ist die Wissenschaft von sicherer (allgemein geheimer) Kommunikation. Diese Sicherheit bedingt, dass die berechtigten Teilnehmer in der Lage sind, eine Nachricht mit Hilfe eines Schlüssels in einen Geheimtext zu transferieren und zurück. Obwohl der Geheimtext für jemand ohne den geheimen Schlüssel unlesbar und unfälschbar ist, kann der berechtigte Empfänger entweder das Chiffrat entschlüsseln, um die den verborgenen Klartext wieder zu erhalten, oder verifizieren, dass die Nachricht aller Wahrscheinlichkeit nach von jemand geschickt wurde, der den richtigen Schlüssel besaß.

Kryptographie beschäftigte sich ursprünglich damit, für Vertraulichkeit von geschriebenen Nachrichten zu sorgen. Die kryptographischen Prinzipien werden jedoch genauso angewandt, um den Informationsfluss zwischen Computern oder Fernsehsignalen zu verschlüsseln.

...Heutzutage liefert die moderne (mathematische) Wissenschaft der Kryptologie nicht nur Verfahren zur Verschlüsselung, sondern auch zur Integrität, für elektronische Signaturen, für Zufallszahlen, sicheren Schlüsselaustausch, sichere Container, elektronische Wahlen und elektronisches Geld. Damit kommen diese Verfahren in einer breiten Palette von Anwendungen des modernen Lebens zum Einsatz.

Quelle: Britannica (www.britannica.com)

Ähnliche Definitionen finden sich auch auf Wikipedia:

- <http://de.wikipedia.org/wiki/Kryptologie>
- <http://de.wikipedia.org/wiki/Kryptografie>

Sicherheitsziele der Kryptographie

- **Vertraulichkeit (*Confidentiality*)**

Lesen des eigentlichen Inhalts für Unbefugte „praktisch“ unmöglich machen

- **Authentifizierung (*Authentication*)**

Identitätsbeweis des Senders gegenüber dem Empfänger einer Nachricht

- **Integrität (*Integrity*)**

Eigenschaft, die bedeutet, 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)

CrypTool-Projekt

- Ursprung im Awareness-Programm einer Großbank (betriebliche Ausbildung)
→ **Sensibilisierung der Mitarbeiter**
- Entwickelt in Kooperation mit Hochschulen (Verbesserung der Lehre)
→ **Mediendidaktischer Anspruch**
- Einige Meilensteine
 - 1998 **Projektstart** – Aufwand für CT1 bisher mehr als 50 Mannjahre
 - 2000 CrypTool als **Freeware** verfügbar für Windows
 - 2002 CrypTool auf der **Bürger-CD des BSI** „Ins Internet – mit Sicherheit“
 - 2003 CrypTool wird **Open-Source** – Hosting durch die Uni Darmstadt
 - 2007 CrypTool in deutsch, englisch, polnisch und spanisch
 - 2008 .NET- und Java-Versionen begonnen – Hosting durch die Uni Duisburg und SourceForge
 - 2010 CT1 auch in Serbisch und Griechisch
 - 2010 CrypTool-Online (CTO) und MysteryTwister C3 (MTC3) veröffentlicht
 - 2011 .NET-Version (CT2) und Java-Versionen (JCT) als erste Betas veröffentlicht
 - 2012 Neues einheitliches Web-Portal für alle 5 CT-Projekte, genannt CrypTool-Portal (CTP)
 - 2014 CT 2.0 Release (August 2014) – Hosting durch die Uni Kassel
 - 2017 CT1 auch in französisch (Release 1.4.40); Relaunch des CrypTool-Portals und des CTO
Siehe auch <https://de.wikipedia.org/wiki/CrypTool>

CrypTool-Projekt

■ Einige Auszeichnungen

2004 TeleTrusT (TTT Förderpreis) 

2004 NRW (IT-Sicherheitspreis NRW)  **NRW.**

2004 RSA Europe (Finalist beim European Information Security Award)



2008 "Ausgewählter Ort" bei der Standortinitiative "Deutschland – Land der Ideen"



■ Entwickler

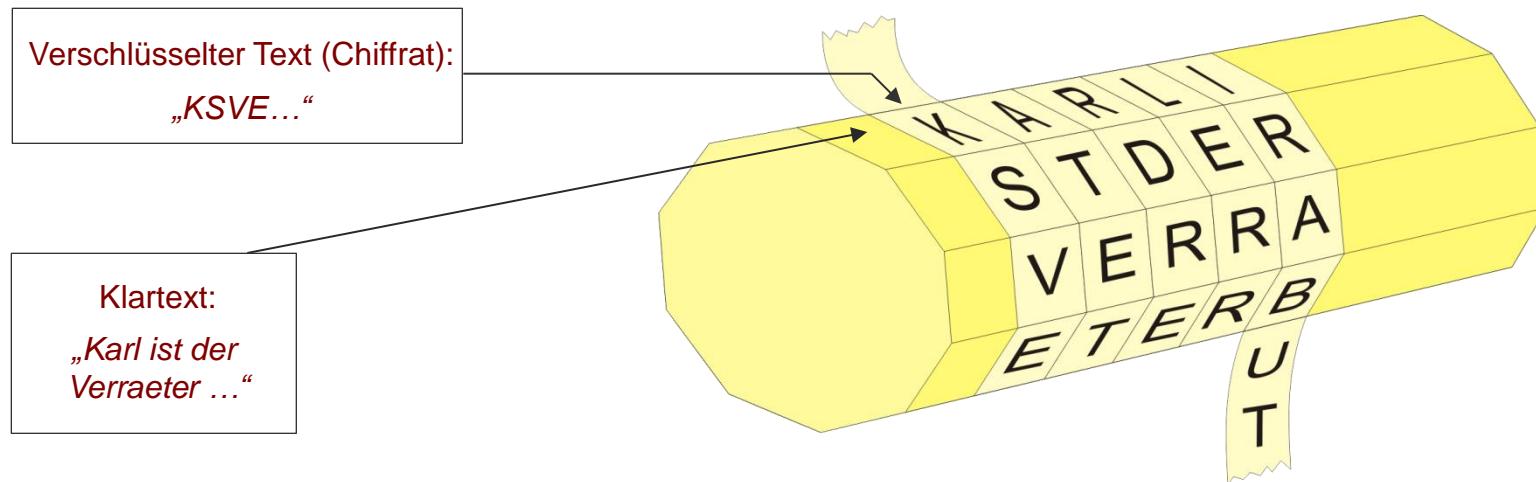
- Entwickelt von Mitarbeitern verschiedener Firmen und Universitäten, Schülern + Studenten
- Weitere Projekt-Mitarbeiter oder verwertbare vorhandene Sourcen sind immer herzlich willkommen (z.Zt. arbeiten ca. 100 Leute weltweit mit).



Beispiele aus der klassischen Kryptographie (1)

Älteste bekannte Verschlüsselungsverfahren

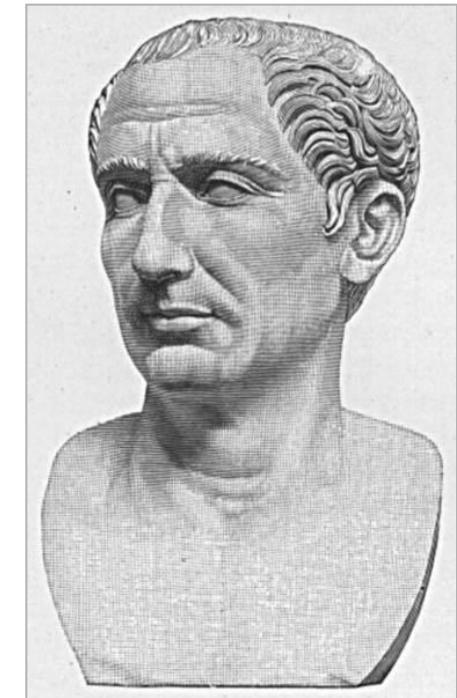
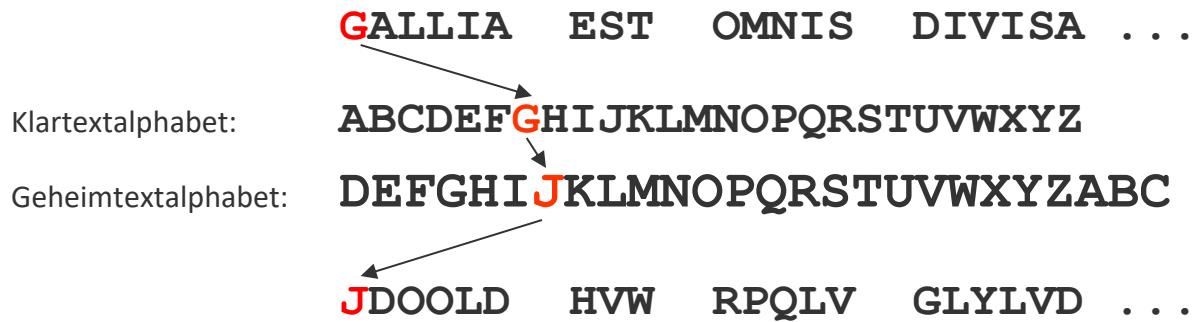
- **Tattoo auf kahlgeschorenen Kopf eines Sklaven** (verdeckt von nachgewachsenen Haaren)
- **Atbash** (um 600 v. Chr.)
 - Hebräische Geheimschrift, umgedrehtes Alphabet
- **Skytale von Sparta** (etwa 500 v. Chr.)
 - Beschrieben vom griechischen Historiker/Schriftsteller Plutarch (45 - 125 n. Chr.)
 - Zwei Zylinder (Holzstäbe) mit gleichem Durchmesser
 - Transposition (Zeichen des Klartextes werden umsortiert)



Beispiele aus der klassischen Kryptographie (2)

Caesar-Verschlüsselung (mono-alphabetische Substitution)

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



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

Vorführung mit CrypTool über folgende Menüs:

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

Beispiele aus der klassischen Kryptographie (3)

Vigenère-Verschlüsselung (poly-alphabetische Substitution)

- **Vigenère-Verschlüsselung** (Blaise de Vigenère, 1523-1596)
- Verschlüsselung mit einem Schlüsselwort unter Nutzung einer Schlüsseltabelle
- Beispiel
Schlüsselwort: **CHIFFRE**
Verschlüsselung: **VIGENERE** wird zu **XPOJSVVG**
- Das Klartextzeichen wird ersetzt durch das Zeichen in der Zeile des Klartextes (bspw. V) und in der Spalte des Schlüsselwortzeichens (bspw. c). Das nächste Zeichen (bspw. I) wird in der Spalte des zweiten Zeichens des Schlüsselwortes (bspw. 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.

The diagram illustrates the Vigenère square, also known as the Tableau Carré or 'Carre de Vigenère'. It consists of a grid of letters where each row and column is shifted by one position relative to the previous one. The columns are labeled with the letters of the alphabet (a through z), and the rows are labeled with the letters of the alphabet (a through z).

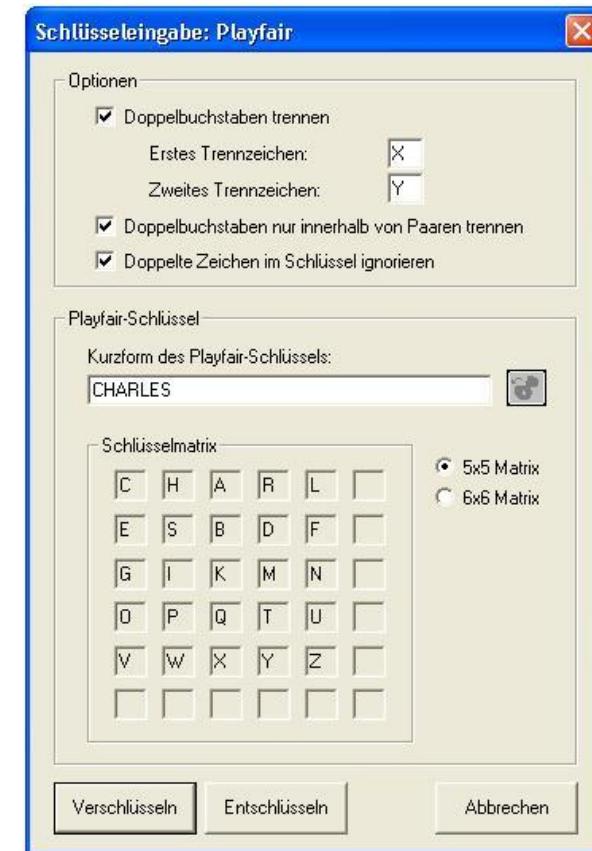
A red arrow points from the text "Schlüsselwort" to the first letter of the key word "CHIFFRE" in the top-left corner of the grid. A red circle highlights the letter 'c' in the first column. Another red circle highlights the letter 'V' in the first row. A red arrow points from the text "Klartextzeichen" to the letter 'V' in the first row. A red arrow points from the text "Verschlüsseltes Zeichen" to the letter 'X' in the second column of the first row. The bottom right corner of the grid contains the text "Tableau carré, dit « Carre de Vigenère »".

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

Beispiele aus der klassischen Kryptographie (4)

Weitere Verfahren der klassischen Kryptographie

- **Homophone Substitution**
- **Playfair** (erfunden 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 One-Time-Pads (OTP)
- **Lochschablonen** (Fleißner)
- **Permutationsverschlüsselung**
 - „Doppelwürfel“
(Reine Transposition / sehr effektiv)



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 symmetrischer Algorithmen:
Kombination der einfacheren Operationen (eine Art der Mehrfach-Verschlüsselung, cascades of ciphers) auf Bit-Ebene, Blockbildung, Runden.

Verschlüsselungsverfahren wurden

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

Beispiel erste Hälfte 20. Jahrhundert

Elektromechanische Verschlüsselungsmaschinen (Rotormaschinen)

Enigma-Verschlüsselung (Arthur Scherbius, 1878-1929)

- Über 200.000 Maschinen waren im 2. Weltkrieg im Einsatz.
- Der rotierende Walzensatz bewirkt, dass jedes Zeichen des Textes mit einem neuen Alphabet verschlüsselt wird.
- Das polnische Cipher Bureau brach die Vorkriegs-Enigma schon 1932.
- Darauf aufbauend wurde die Enigma gebrochen mit massivem Einsatz [etwas 7000 Personen in UK, mit den ersten Entschlüsselungsmaschinen sowie erbeuteten Original-Maschinen, Nutzen abgefangener täglicher 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_Maschine vom 06.03.2006)



Kryptographie – Entscheidende Erkenntnisse (1)

Kerckhoffs-Prinzip (formuliert 1883)

- Trennung von Algorithmus (Verfahren) und Schlüssel
z.B. bei Caesar:
 - Algorithmus: „Verschiebe Alphabet um eine bestimmte Anzahl Positionen zyklisch nach links“
 - Schlüssel: Diese „bestimmte Anzahl Positionen“ (bei Caesar: 3)
- Kerckhoffs-Prinzip:
Das Geheimnis liegt im Schlüssel und nicht im Algorithmus bzw. keine „security through obscurity“.

One-Time-Pad – Shannon / Vernam

- Beweisbar sicher, jedoch praktisch kaum anwendbar (benutzt beim Roten 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 Geheim-Information) nicht
- nur mit dem Geheimnis geht auch die andere Richtung (Zugang zur Falltür)



* Siehe http://de.wikipedia.org/wiki/Hei%C3%9Fer_Draht

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 vorneherein offen legen will, um der Fachwelt eine Prüfung zu ermöglichen.“*

[<http://www.heise.de/newsticker/meldung/7183>]

■ Netscape Navigator (1999)

Er legte die Passworte für den Zugriff auf E-Mail-Server 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*)

Menü:
„Ver-/Entschlüsseln“ \
„Symmetrisch (klassisch)“ \
„Vernam“

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

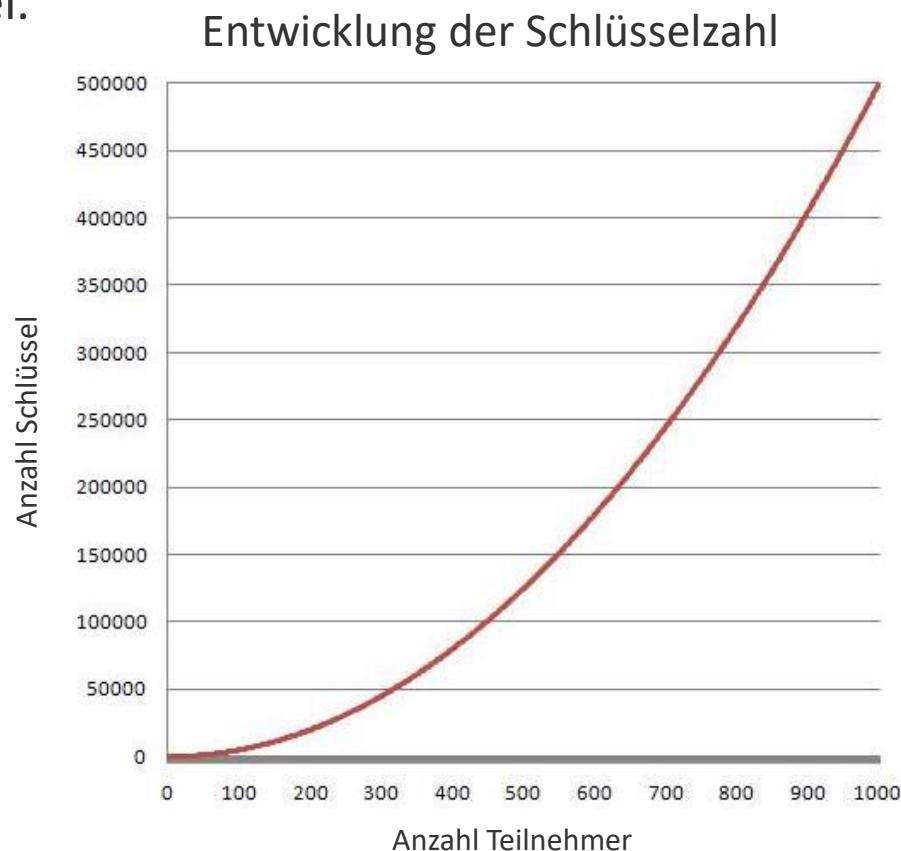
$n = 1.000$ Personen sind es

$S_{1000} = 499.500$ Schlüssel.

⇒ Quadratischer Anstieg:

Faktor 10 mehr Personen,

Faktor 100 mehr Schlüssel



Kryptographie – Entscheidende Erkenntnisse (2)

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)

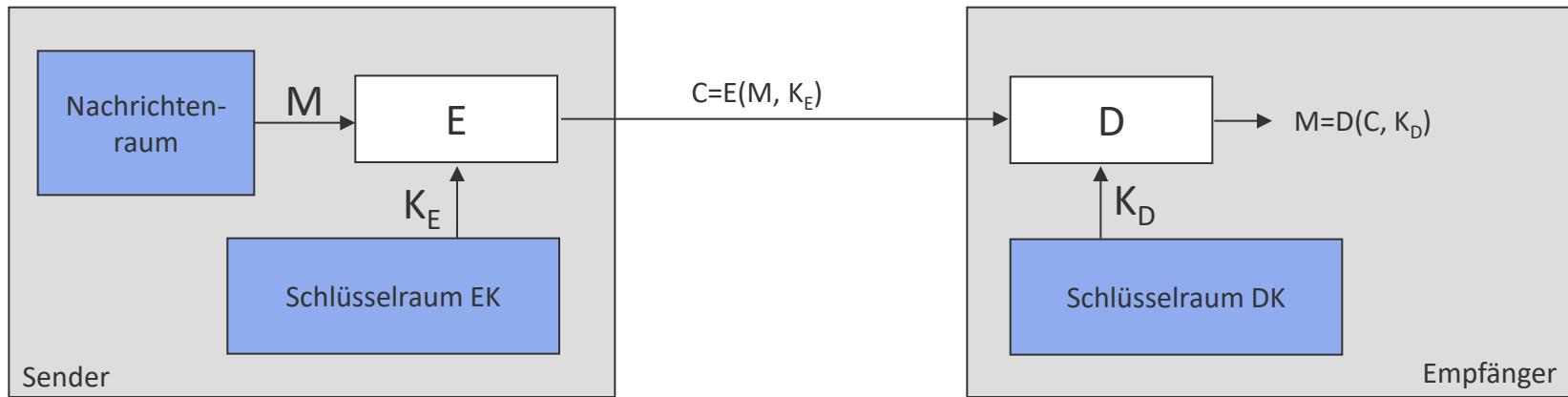
Schlüsselverteilung

- 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 Verschlüsselung: $K_E = K_D$ (z.B. AES)

b) Asymmetrische Verschlüsselung: $K_E \neq K_D$

$$K_E \neq K_D$$

geheim
↑
↓
öffentliche Schlüssel privater Schlüssel

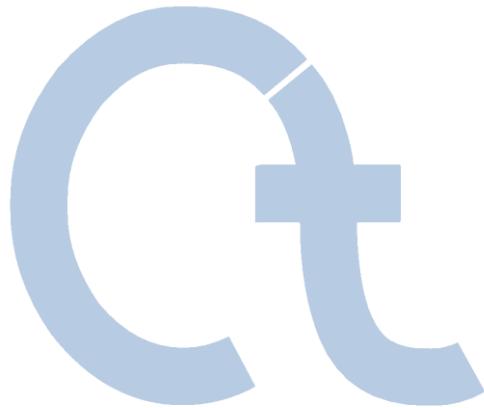
Kryptographie – Entscheidende Erkenntnisse (3)

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-768 (CrypTool-Buch, Kap. 4.11.3)
 - Sehr hohe Intensität in der aktuellen Forschung
Faktorisierung, nicht-parallelisierbare Algorithmen (gegen Quantencomputer), besseres Verständnis von Protokoll-Schwächen und Zufallszahlengeneratoren, usw.
- Entscheidender Irrtum: „*Echte Mathematik*“ hat keine Auswirkungen auf den Krieg. (G.H. Hardy, 1940)
- Hersteller entdecken **Sicherheit** als ein zentrales **Kaufkriterium**.
- Falsche Glaubenssätze:
Verschlüsselung / Datenschutz und Aufklärung / Innovation sind Gegensätze.

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" = "RUGCLENWGYXDATRNHNMH"
- **Vernam (OTP)**
- **AES (Ausgabe-Key, Brute-Force-Analyse)**



- I. CrypTool und Kryptologie – Überblick
 - II. Was bietet CrypTool 1?
 - III. Ausgewählte Beispiele
 - IV. Projekt / Ausblick / Kontakt
- Anhang

1. Was ist CrypTool?

- Kostenloses 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



CrypTool-Programm

- Alle Funktionen integriert in *einem* Programm mit einheitlicher graphischer Oberfläche
- Läuft unter Win32
- Nutzt Kryptographiefunktionen aus den Bibliotheken von Secude, cryptovision und OpenSSL
- Langzahlarithmetik per Miracl, APFLOAT und GMP/MPIR, 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* Programmfunctionen (auch auf Menüs)
- Ausführliche Benutzungs-Szenarien (Tutorials) für viele Programmfunctionen

Buch (.pdf-Datei) mit Hintergrundinformationen

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

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 (1)

Kryptographie

Verschlüsselungsklassiker

- Caesar (und ROT-13)
- Monoalphabetische Substitution (und Atbash)
- Vigenère
- Hill
- Homophone Substitution
- Playfair
- ADFGVX
- Byteweise Addition
- XOR
- Vernam
- Permutation (Gartenzaun, Skytale, Doppelwürfel, ...)
- Solitaire

Optionen zum besseren Nachvollziehen von Literaturbeispielen

- Alphabet wählbar
- Behandlung von Leerzeichen etc. einstellbar

Kryptoanalyse

Angriffe auf klassische Verfahren

- Ciphertext-Only
 - Caesar
 - Vigenère (nach Friedman + Schrödel)
 - Addition
 - XOR
 - Substitution
 - Playfair
- Known-Plaintext
 - Hill
 - Einstufige Permutation/Transposition
- Manuell (unterstützt)
 - Monoalphabetische Substitution
 - Playfair, ADFGVX, Solitaire

Unterstützende Analyseverfahren

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

Funktionsumfang (2)

Kryptographie

Moderne symmetrische Verschlüsselung

- IDEA, RC2, RC4, RC6, DES, 3DES, DESX
- 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
- Annahmen:
 - Entropie des Klartextes klein,
 - Teilweise Kenntnis der Schlüssels, oder
 - Kenntnis 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 (3)

Kryptographie

Digitale Signatur

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

Hashfunktionen

- MD2, MD4, MD5
- SHA, SHA-1, SHA-2, 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 Bit 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, Vitányi, Entropie
- Gleitende Häufigkeit, Histogramm
- n-Gramm-Analyse, Autokorrelation
- ZIP-Kompressionstest

Funktionsumfang (4)

Visualisierungen / Demos

- Caesar, Vigenère, Nihilist, DES (mit ANIMAL)
- Enigma (Flash)
- Rijndael/AES (zweimal mit Flash, einmal mit Java)
- 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
- Sichere E-Mail mit dem S/MIME-Protokoll (mit Java und Flash)
- Grafische 3-D-Darstellung von (Zufalls-)Datenströmen
- Sensibilität von Hashfunktionen bezüglich Änderungen an den Daten
- Zahlentheorie und RSA-Kryptosystem (mit Authorware)



Funktionsumfang (5)

Weitere Funktionen

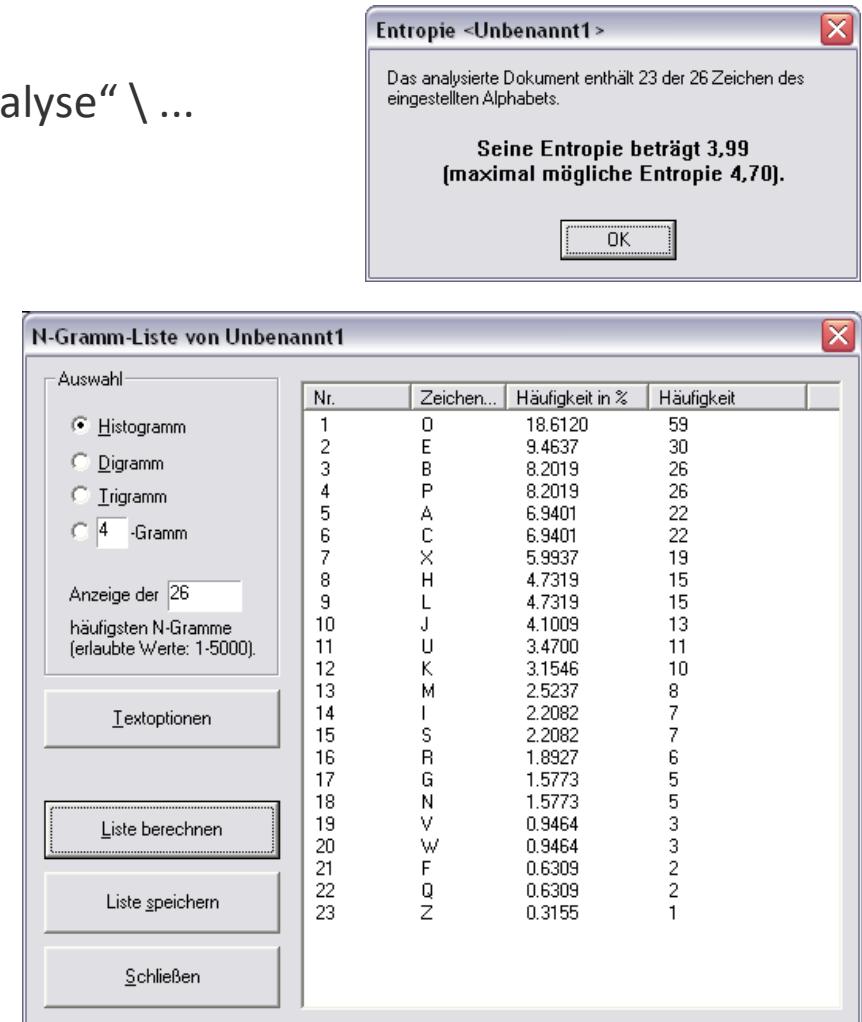
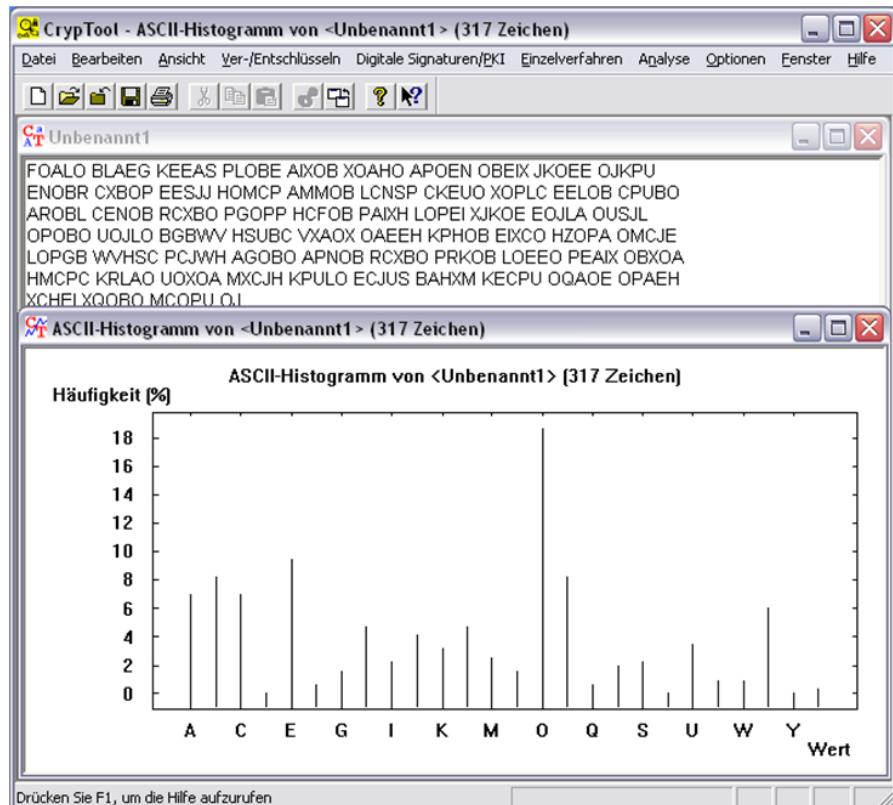
- Verschiedene Funktionen zu RSA und Primzahlen
- Homophone und Permutationsverschlüsselung (Doppelwürfel)
- PKCS #12-Import/Export für PSEs (Personal Security Environment)
- Hashwerte großer Dateien berechnen, ohne sie zu laden
- Generischer Brute-Force-Angriff auf beliebige moderne symmetrische Algorithmen
- ECC-Demo (als Java-Applikation)
- Passwort-Qualitätsmesser (PQM) und Passwort-Entropie
- Vielfältige Textoptionen für die klassischen Verfahren (siehe Anwendungsbeispiel S. 99)
- Und vieles mehr ...



Sprachstruktur analysieren

Anzahl Einzelzeichen, n-Gramme, Entropie

- z.B. im Menü: „Analyse“ \ „Werkzeuge zur Analyse“ \ ...



Demonstration der Interaktivität (1)

Demo per CrypTool

Vigenère-Analyse

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 (2)

Demo per CrypTool

Automatisierte Primzahlzerlegung

Primzahlzerlegung mit Hilfe von Faktorisierungsverfahren

- 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

Menü: „Einzelverfahren“ \ „RSA-Kryptosystem“ \ „Faktorisieren einer Zahl“

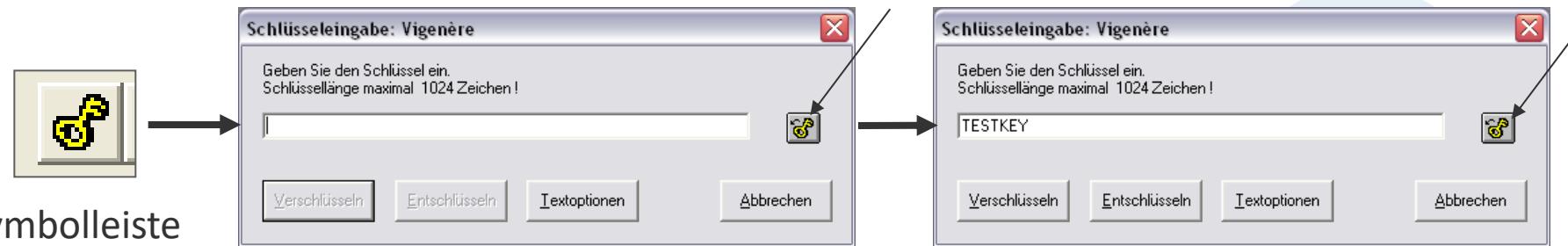
Konzepte zur Benutzerfreundlichkeit

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 Geheimtext-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 Schlüssel-Speicher** benutzt, der pro Verfahren zur Verfügung steht (nützlich insbesondere bei „strukturierten“ Schlüsseln wie der homophonen Verschlüsselung).



Herausforderungen für den Programmierer

1. Verschiedene Funktionen parallel laufen lassen

- Bei der Faktorisierung laufen die verschiedenen Algorithmen in Threads.

2. Hohe Performance

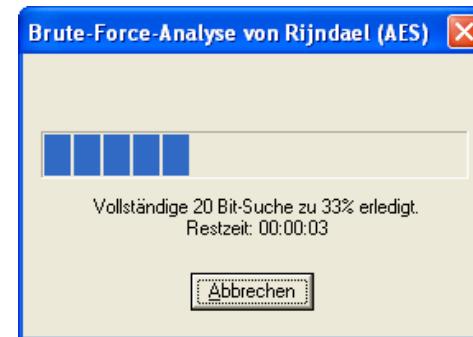
- Bei der Anwendung des Geburtstagsparadoxons zum Finden von Hashkollisionen oder bei der Brute-Force-Analyse.

3. Speicherbeschränkung beachten

- Beim Floyd-Algorithmus (Mappings für das Finden von Hashkollisionen) oder beim Quadratischen Sieb.

4. Zeitmessung und -abschätzung

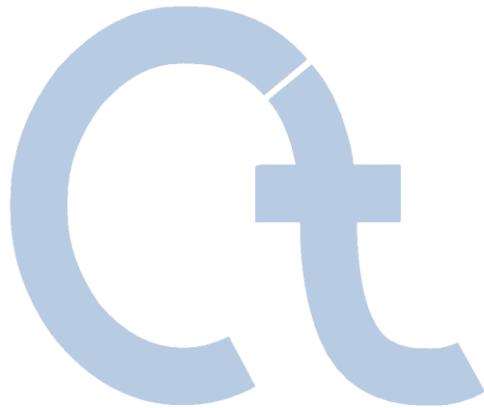
- Ausgabe der Restzeit (z.B. bei Brute-Force-Analyse)



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. Automatisierung der Konsistenz der Funktionen, der GUI und der Online-Hilfe (inklusive verschiedener Sprachen und den unterstützten Windows-Betriebssystemen)



I. CrypTool und Kryptologie –
Überblick

II. Was bietet CrypTool 1?

III. Ausgewählte Beispiele

IV. Projekt / Ausblick / Kontakt

Anhang

CrypTool-Anwendungsbeispiele

Übersicht der Beispiele

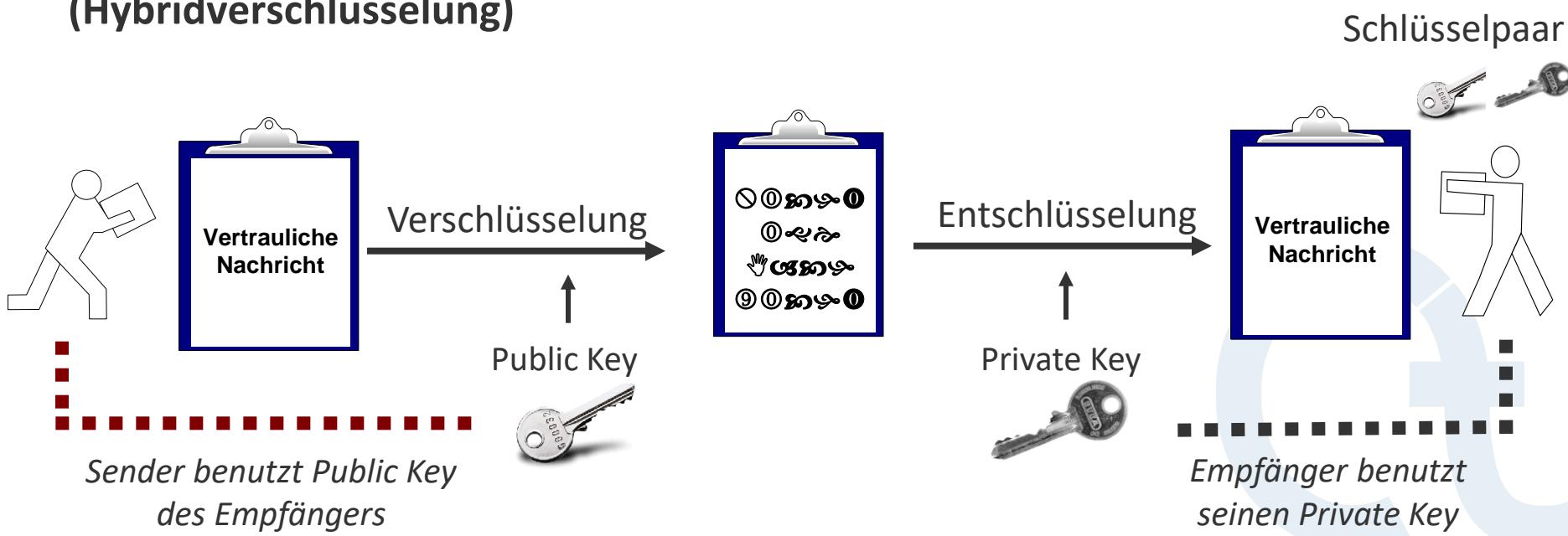
1. [Verschlüsselung mit RSA / Primzahltests / Hybridverschlüsselung und Digitale Zertifikate / SSL](#)
2. [Elektronische Signatur visualisiert](#)
3. [Angriff auf RSA-Verschlüsselung \(Modul N zu kurz\)](#)
4. [Analyse der Verschlüsselung im PSION 5](#)
5. [Schwache DES-Schlüssel](#)
6. [Auffinden von Schlüsselmaterial \(„NSA-Key“\)](#)
7. [Angriff auf Digitale Signatur durch Suche nach Hashkollisionen](#)
8. [Authentisierung in einer Client-Server-Umgebung](#)
9. [Demonstration eines Seitenkanalangriffs \(auf ein Hybridverschlüsselungsprotokoll\)](#)
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 \(Lösung linearer Kongruenzsysteme\)](#)
14. [Visualisierung von symmetrischen Verschlüsselungsverfahren mit ANIMAL](#)
15. [Visualisierungen von AES](#)
16. [Visualisierung der Enigma-Verschlüsselung](#)
17. [Visualisierung von sicherer E-Mail mit S/MIME](#)
18. [Erzeugung eines Message Authentication Code \(HMAC\)](#)
19. [Hash-Demo](#)
20. [Lernprogramm zur Zahlentheorie und zur asymmetrischen Verschlüsselung](#)
21. [Punktaddition auf elliptischen Kurven](#)
22. [Passwort-Qualitätsmesser und Passwort-Entropie](#)
23. [Brute-Force-Analyse](#)
24. [Skytale / Gartenzaun](#)
25. [Hill-Verschlüsselung / Hill-Analyse](#)
26. [CrypTool Online-Hilfe / Menübaum zum Programm](#)



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) [oft wird der Modulus n auch groß N geschrieben]
- Privater Schlüssel (private key): (d)

wobei

p, q sind große zufällig gewählte Primzahlen mit $n = p * q$;

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

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

- n ist der Modulus (seine Länge ist die „Schlüssellänge“ beim RSA-Verfahren).
- 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: $2^r \leq n$ ($2^r - 1 < n$)

Siehe auch: Aufwändige, interaktive Flash-Animation zu den Grundlagen von RSA:

<https://www.cryptoool.org/images/ct1/presentations/RSA/RSA-Flash-de/player.html>

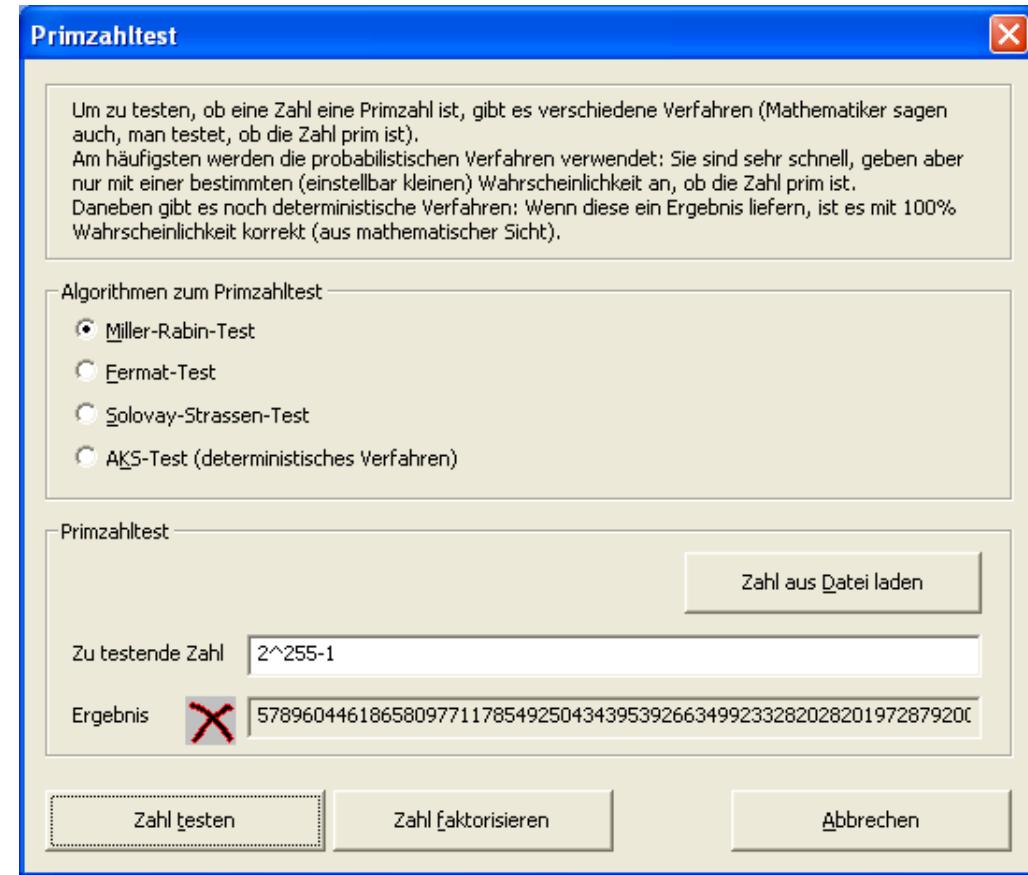
Anwendungsbeispiele (1)

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 zu untersuchende Zahl prim ist, als die bekannten Faktorisierungsverfahren eine Zahl ähnlicher Größenordnung in ihre Primfaktoren zerlegen können.

Für den AKS-Test wurde die GMP/MPIR-Bibliothek (**GNU Multiple Precision Arithmetic Library ; Multiple Precision Integers and Rationals**) in CrypTool integriert.



Menü: „Einzelverfahren“ \ „RSA-Kryptosystem“ \ „Primzahltest“

Bemerkung: $2^{255} - 1 = 7 * 31 * 103 * 151 * 2143 * 11119 * 106591 * 131071 * 949111 * 9520972806333758431 * 5702451577639775545838643151$

Anwendungsbeispiele (1)

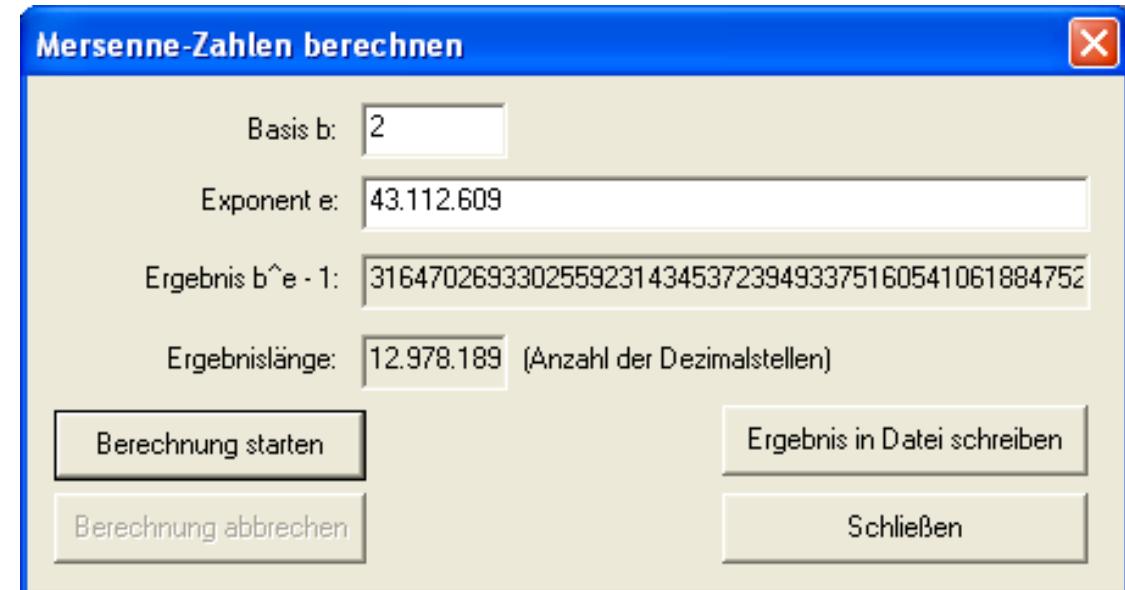
Ausgabe aktueller Primzahl-Rekorde – Mersenne-Primzahlen

Die größten bekannten Primzahlen sind sogenannte Mersenne-Primzahlen.

Die aktuell viert-größte hat 12.978.189 Dezimalstellen und wurde 2008 innerhalb des GIMPS-Projektes gefunden.

Der nebenstehende Dialog kann jede Stelle solcher Zahlen sehr schnell berechnen und ausgeben.

Dazu wurde die APFLOAT-Bibliothek in CrypTool integriert. Über das Kontextmenü jedes Ein- und Ausgabefeldes dieses Dialogs kann man die Tausender-Trenner an- und ausschalten.



Bemerkung: $2^{43.112.609} - 1 = 316.470.269 \dots 697.152.511$

Große Zahlen sollte man wegen der Performance der GUI nicht im Ergebnisfeld markieren und abgreifen, sondern mit dem Button „Ergebnis in Datei speichern“ in ihrer ganzen Länge im CrypTool-Hauptfenster anzeigen.

Menü: „Einzelverfahren“ \ „Zahlentheorie interaktiv“ \ „Mersenne-Zahlen berechnen“

Anwendungsbeispiele (1)

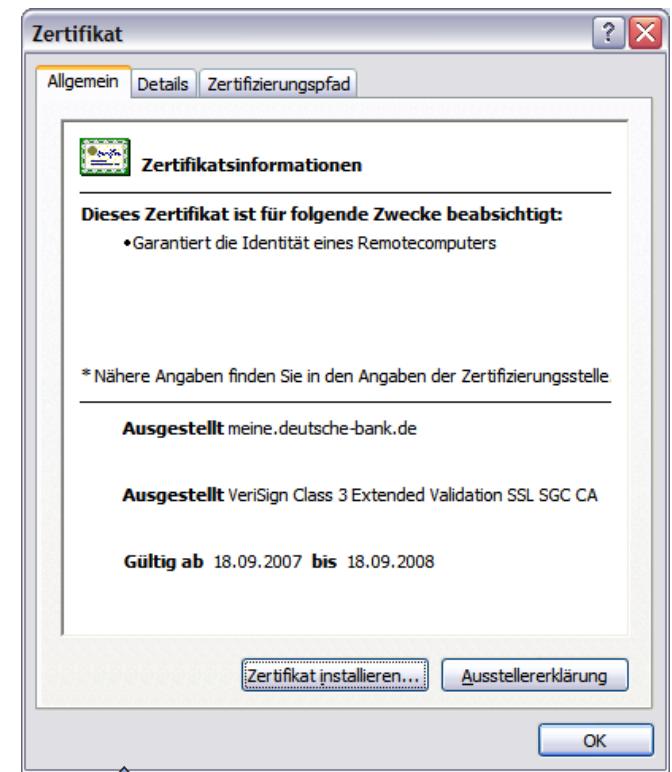
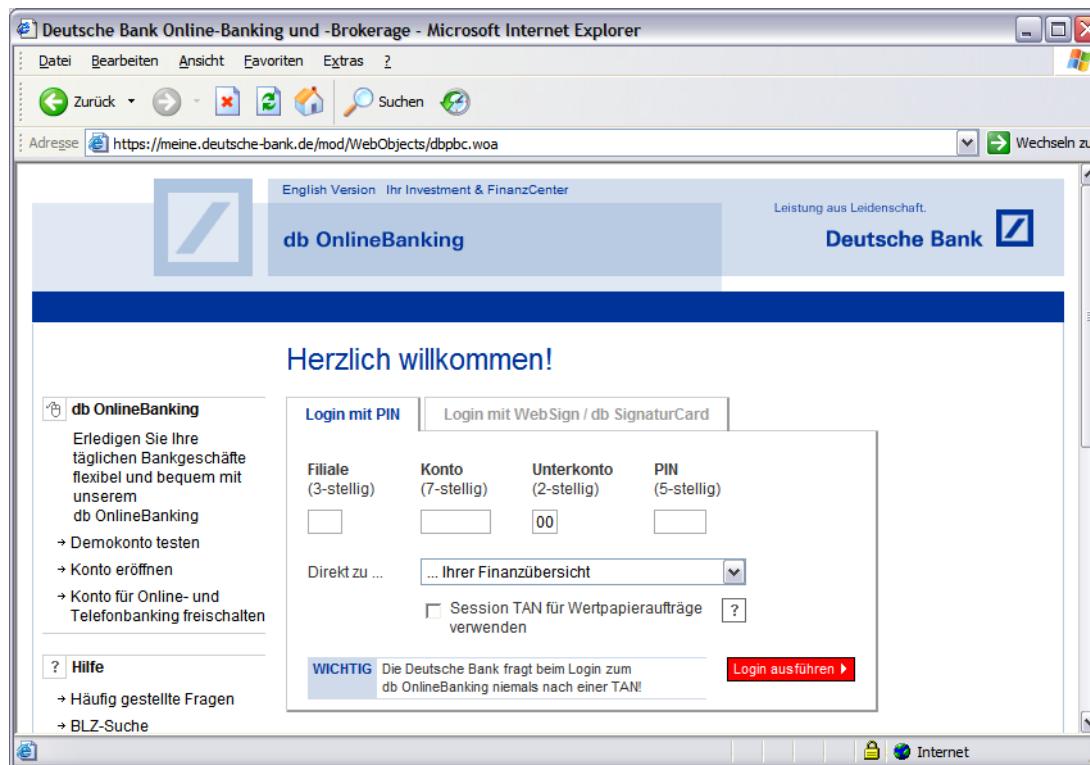
Hybridverschlüsselung und Digitale Zertifikate

- 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 Empfänger auch wirklich dem Empfänger?**
- Lösung: Digitale Zertifikate – **Eine zentrale Instanz (z.B. GlobalSign, 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

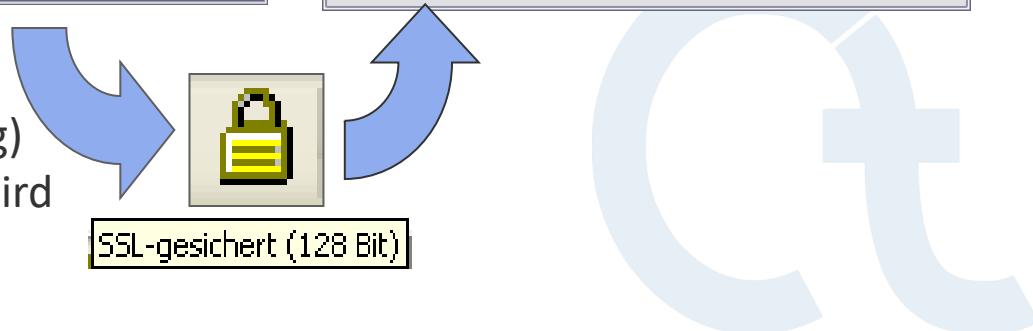


Anwendungsbeispiele (1)

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

Zertifikat

Allgemein Details Zertifizierungspfad

Anzeigen: <Alle>

| Feld | Wert |
|------------------------|-----------------------------------|
| Version | V3 |
| Seriennummer | 3c 16 fe d8 e8 58 7d 56 48 4b ... |
| Signaturalgorithmus | sha1RSA |
| Aussteller | VeriSign Class 3 Extended Vali... |
| Gültig ab | Dienstag, 18. September 2007... |
| Gültig bis | Donnerstag, 18. September 2... |
| Antragsteller | meine.deutsche-bank.de, Deu... |
| Öffentlicher Schlüssel | RSA (2048 Bits) |

```
30 82 01 0a 02 82 01 01 00 cb 50 dc d6 1c  
87 6f a9 6b 48 98 c6 4b a2 a5 5e 6a 35 6e  
69 b5 ae 36 68 f8 d0 98 ca 5e 0f d1 da d6  
47 00 05 cc fb 2b cf 3d 9f d0 e2 55 1a bd  
5e 14 f8 7e ca bf 87 b2 9e a4 4c b6 d3 2d  
50 fe c6 3b 67 b9 2a 4a 40 51 be 05 68 30  
98 79 1c 10 82 8b 99 bd c1 de 78 61 61 1a  
85 23 b6 9e cc 07 6e 7b b3 e6 25 f5 03 b9  
f0 de 7a 80 93 57 f3 42 ce 95 dd 58 0f 0b
```

Eigenschaften bearbeiten... In Datei kopieren... OK

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)

Client



1. SSL-Verbindungsaufbau

Server



Sende Server-Zertifikat

- 2.

3. Überprüfung des Server-Zertifikats (mit Hilfe der gespeicherten Root-Zertifikate)

4. Ermittle öffentlichen Schlüssel des Server-Zertifikat

5. Generiere einen zufälligen symmetrischen Schlüssel (Session Key)

6. Sende Session Key

(verschlüsselt mit öffentlichem Schlüssel des Servers)

Empfange Session Key

(Entschlüsselung durch privaten Schlüssel des Servers)

- 7.



SSL-gesichert (128 Bit)

**Verschlüsselte Kommunikation auf Basis
des vereinbarten Session Keys**

Anwendungsbeispiele (1)

Aufbau einer gesicherten SSL-Verbindung (Server Authentication)

Allgemein

- Das Beispiel skizziert den typischen Aufbau einer SSL-Verbindung zur Übertragung von sensiven 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 Benutzernamen 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 Verbindungsauftbau – 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 Zertifikatshierarchien), 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)

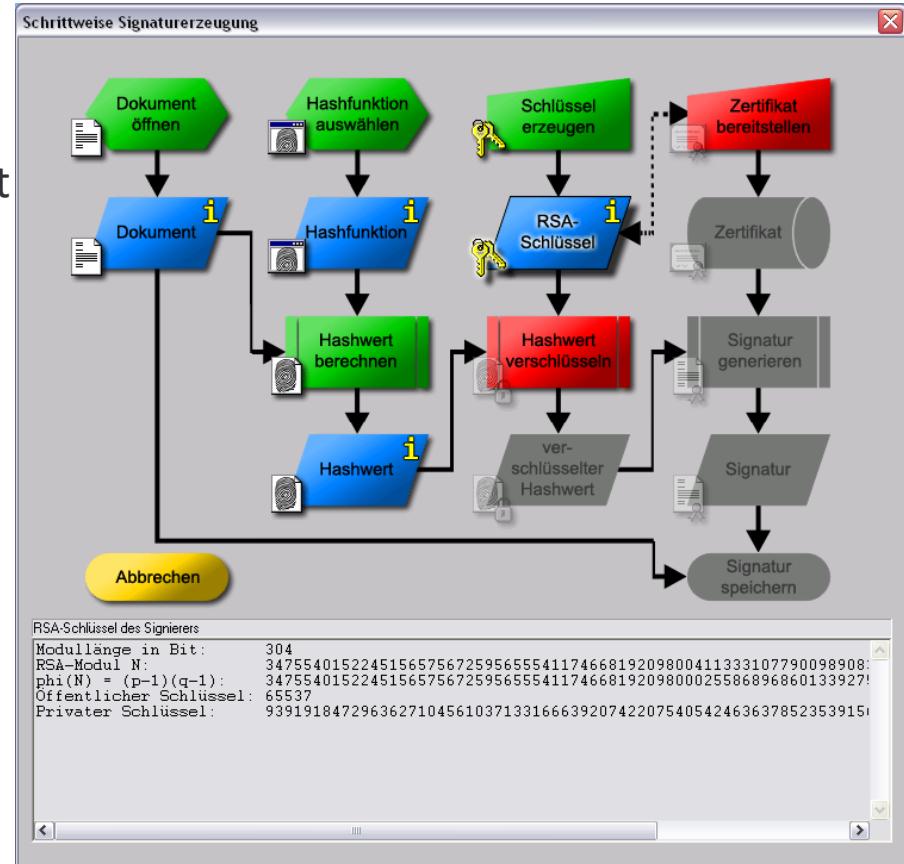
Elektronische Signatur visualisiert

Elektronische Signatur

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

Visualisierung in CrypTool

- Interaktives Datenflussdiagramm
- Ähnlich wie die Visualisierung der Hybridverschlüsselung



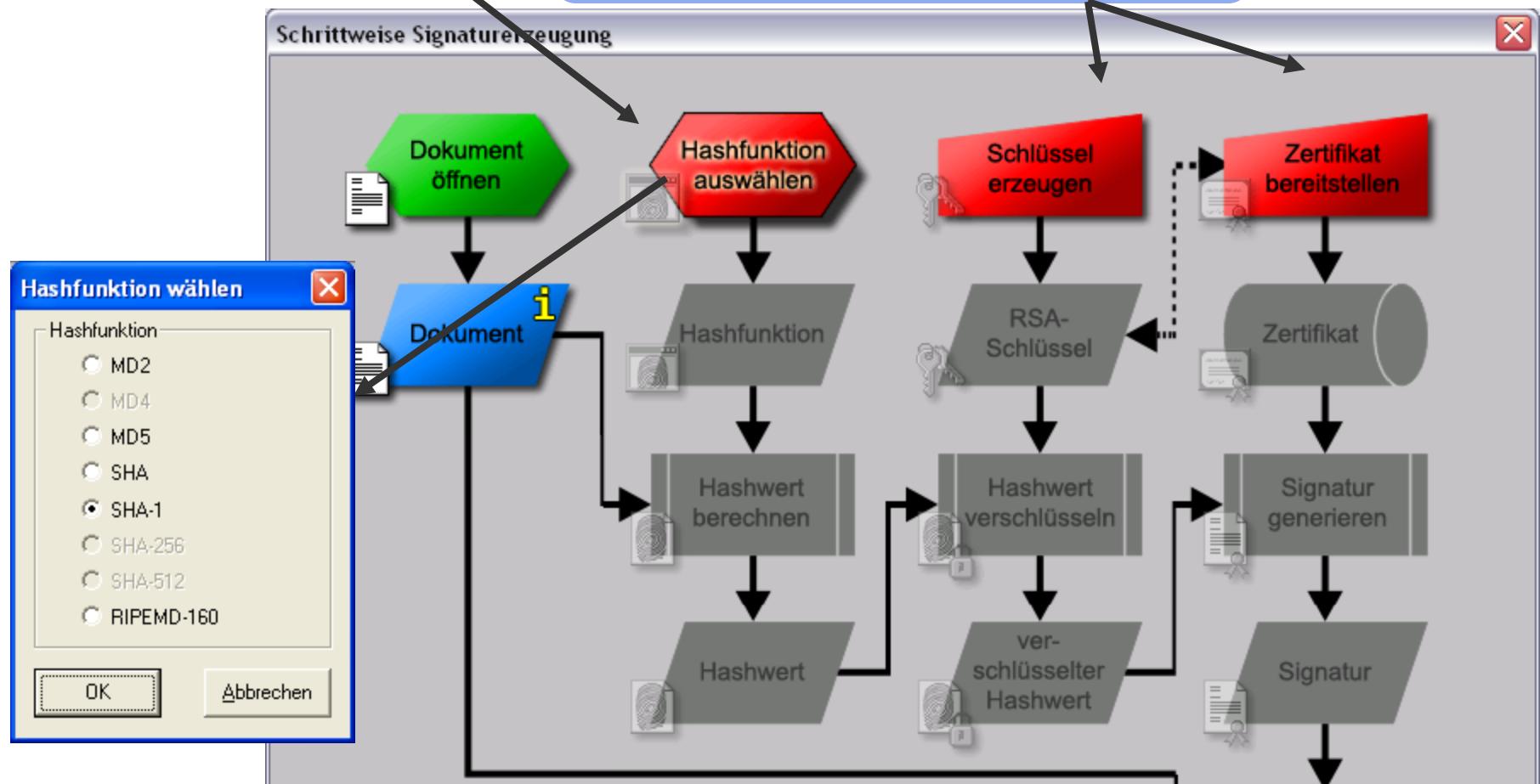
Menü: „Digitale Signaturen/PKI“ \
„Signaturdemo (Signaturerzeugung)“

Anwendungsbeispiele (2)

Elektronische Signatur visualisiert: a) Vorbereitung

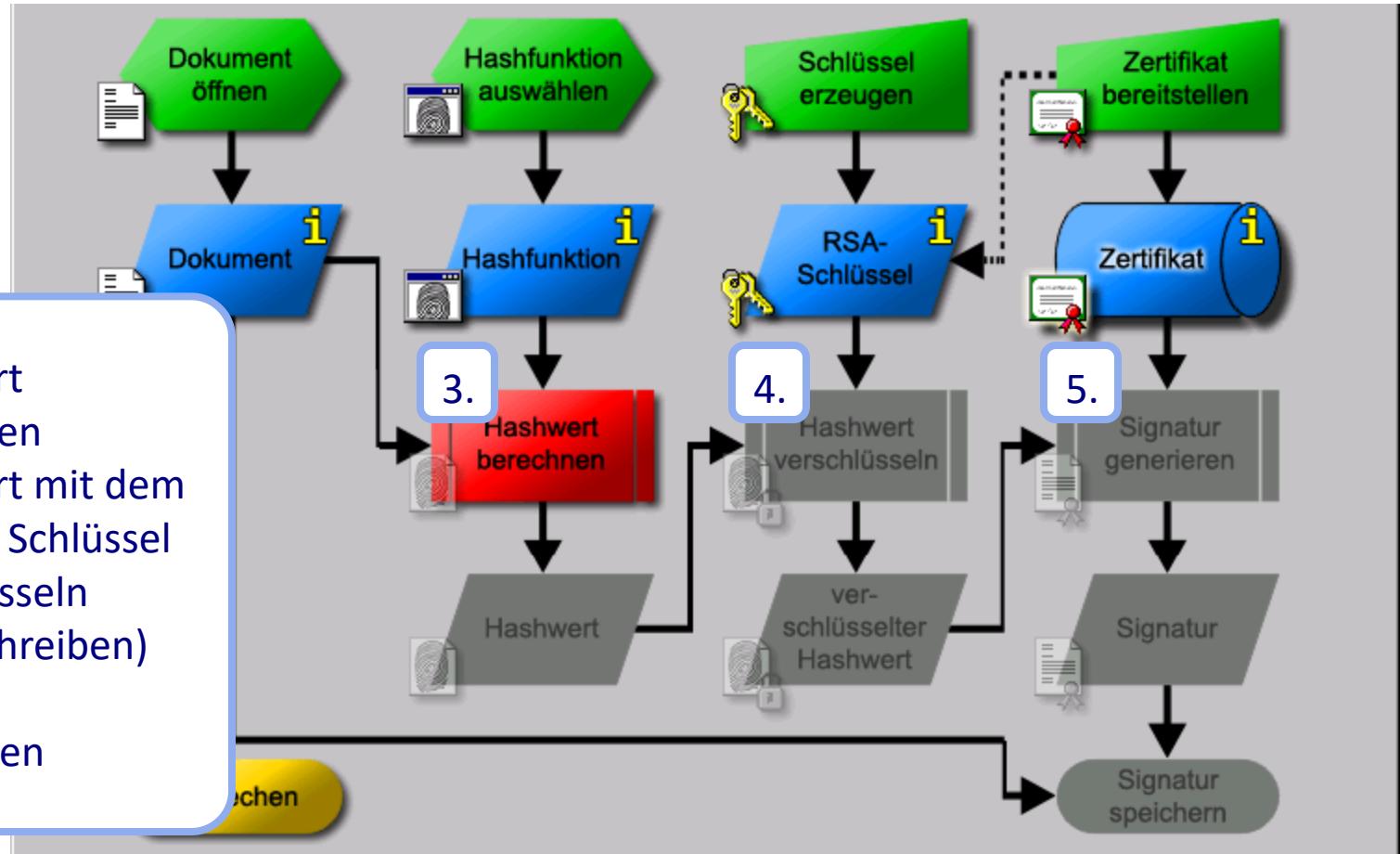
1. Hashfunktion wählen

2. Schlüssel und Zertifikat bereitstellen
(hier nicht abgebildet)



Anwendungsbeispiele (2)

Elektronische Signatur visualisiert: b) Kryptographie

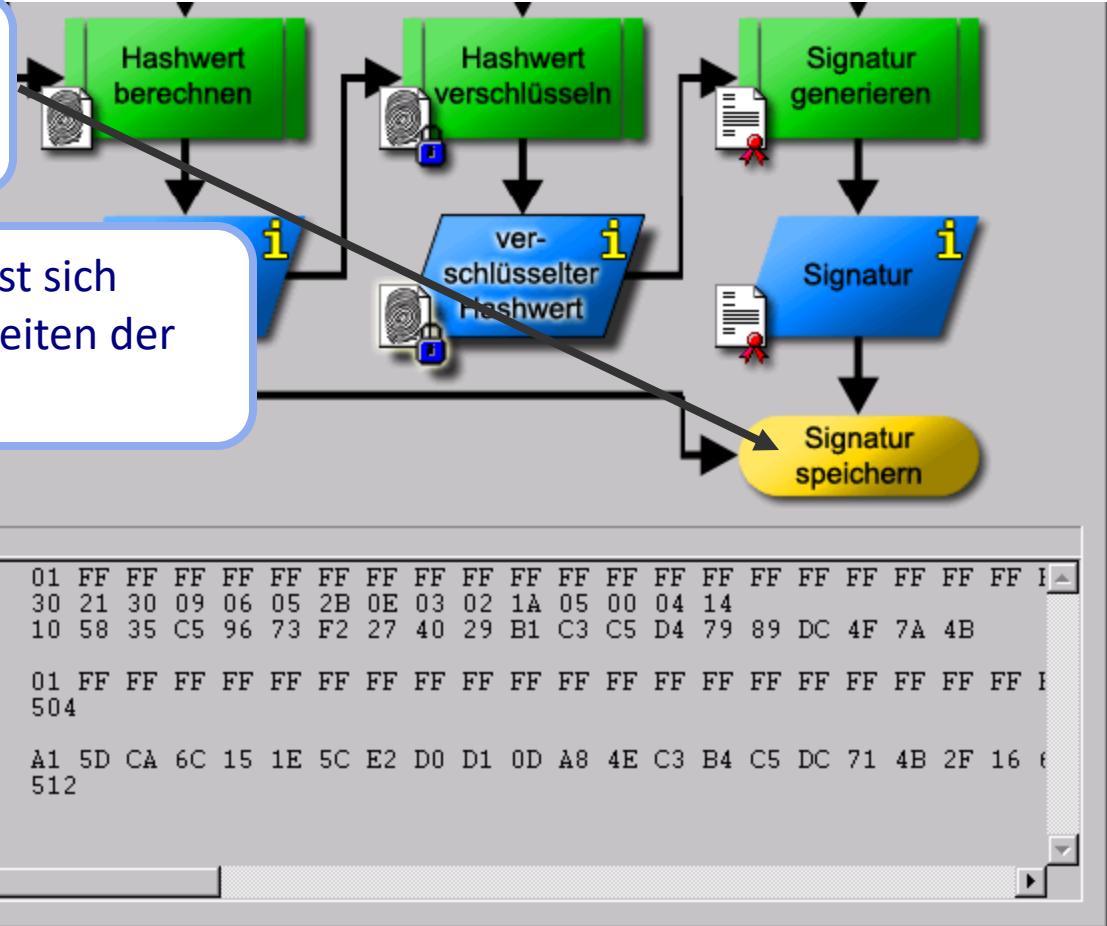


Anwendungsbeispiele (2)

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)

Angriff auf RSA-Verschlüsselung bei zu kurzem RSA-Modul N

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

- Öffentlicher Schlüssel
- RSA-Modul **N = 63978486879527143858831415041** (95 Bit, 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 Geheimtext 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: $M_i = C_i^d \bmod N$

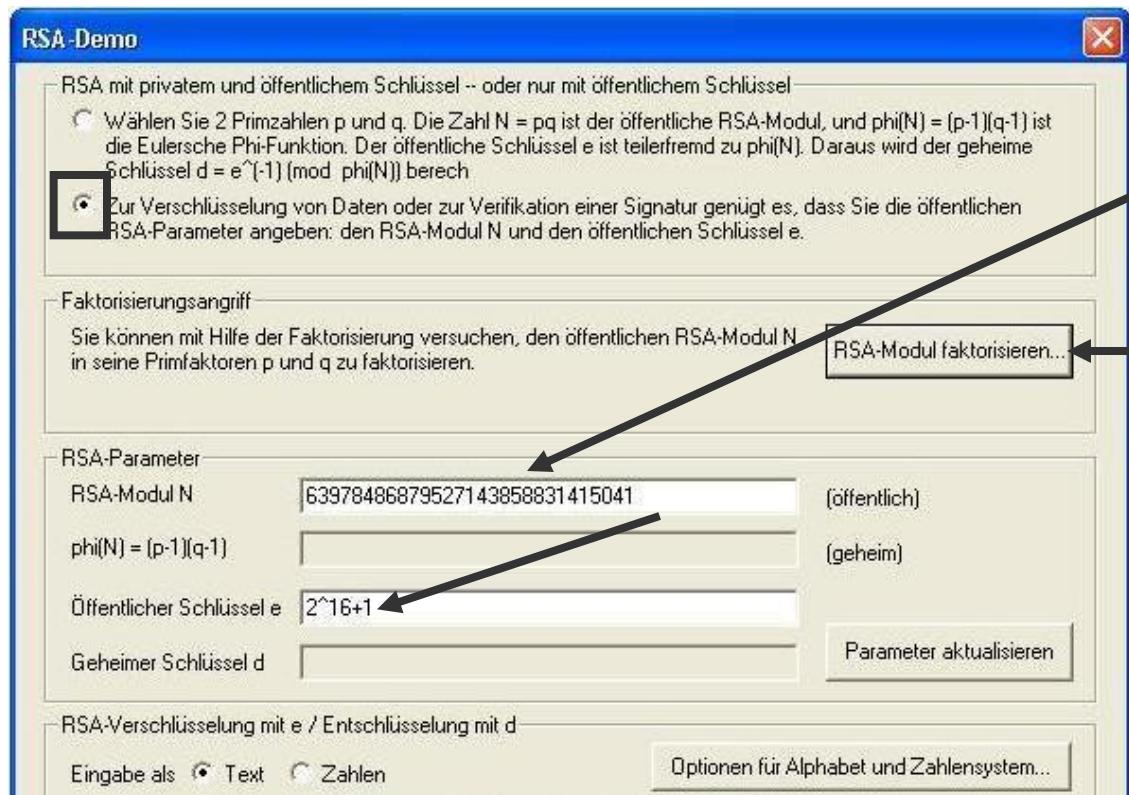
Angriff mit CrypTool 1 ist für RSA-Module bis ca. 250 Bit praktikabel

Danach könnte man für jemand anderen elektronisch unterschreiben!

Anwendungsbeispiele (3)

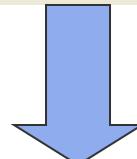
Kurzer RSA-Modul: Öffentliche Parameter eingeben

Menü: „Einzelverfahren“ \ „RSA-Kryptosystem“ \ „RSA-Demo...“



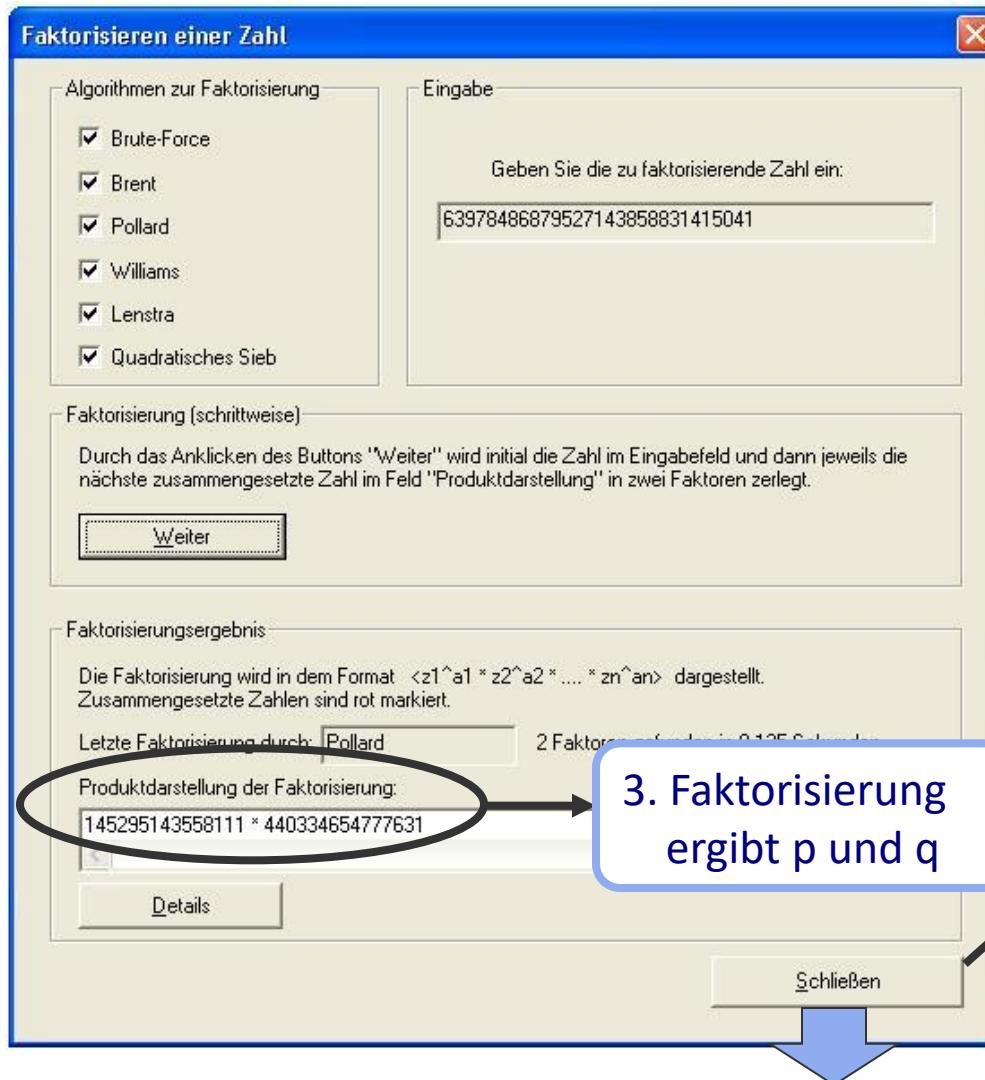
1. Öffentliche RSA-Parameter N und e eingeben

2. Faktorisieren



Anwendungsbeispiele (3)

Kurzer RSA-Modul: RSA-Modul faktorisieren



Anwendungsbeispiele (3)

Kurzer RSA-Modul: Geheimen Schlüssel d bestimmen

RSA-Demo

RSA mit privatem und öffentlichen 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 Phi-Funktion. 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 öffentlichen RSA-Parameter angeben: den RSA-Modul N und den öffentlichen Schlüssel e.

Primzahleingabe

Primzahl p: 145295143558111
Primzahl q: 440334654777631

Primzahlen generieren...

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...

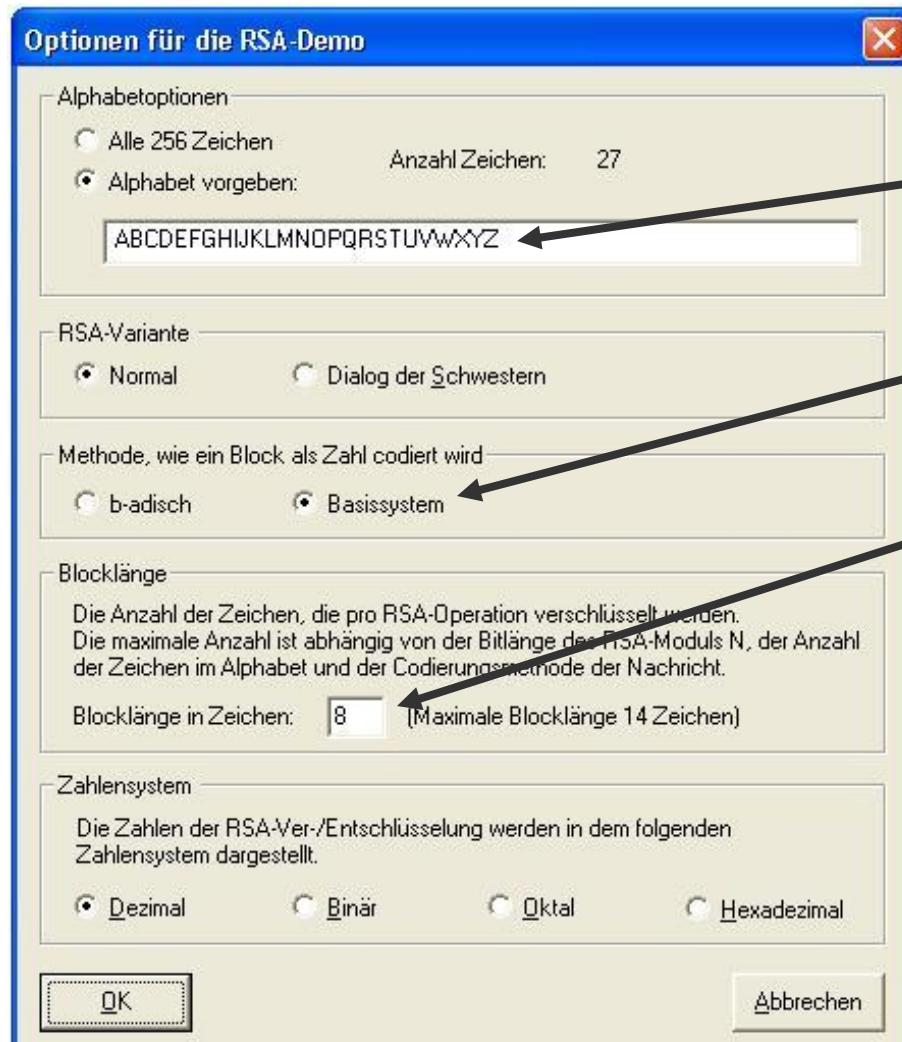
Wechsel in die Ansicht des Besitzers des geheimen Schlüssels

4. p und q wurden automatisch eingetragen, und der private Schlüssel d wurde berechnet

5. Optionen einstellen

Anwendungsbeispiele (3)

Kurzer RSA-Modul: Optionen einstellen



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 .

6764533946879682306085915 # 40145547044051327868820472463 # 30821430257177284710956359014

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

00000000000000920190020080500 # 000000000000000514040015060020 # 00000000000000080500231518

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

ITS THE # END OF T # HE WORLD # AS WE K # NOW IT A # ND I FEE # L FINE

Klartext

ITS THE END OF THE WORLD AS WE KNOW IT AND I FEEL FINE

[Verschlüsseln](#) [Entschlüsseln](#) [Schließen](#)

9. Geheimtext 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

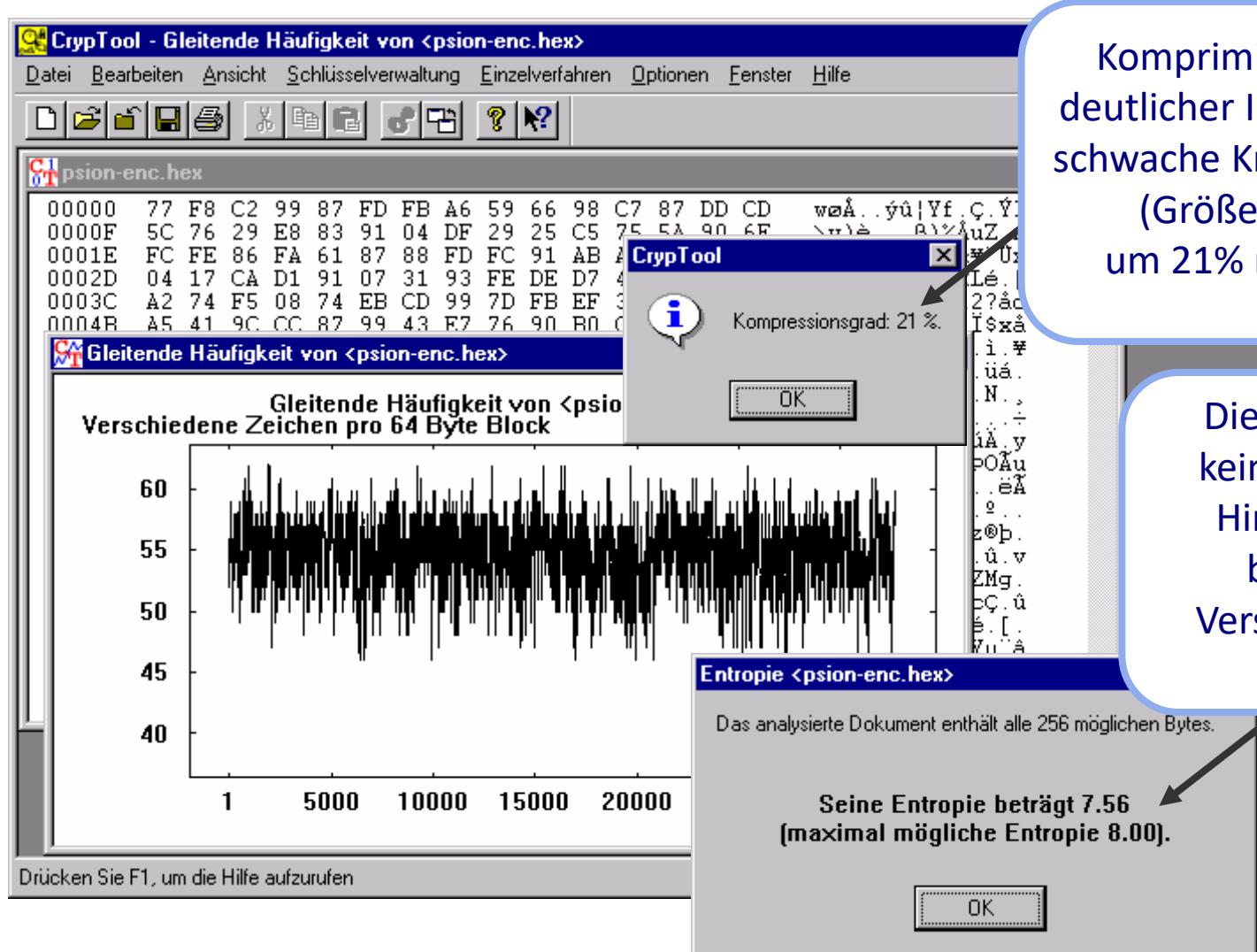
- Voranalyse
 - Entropie
 - gleitende Häufigkeit
 - Kompressionstest
- Autokorrelation
- automatische Analyse mit verschiedenen klassischen Verfahren durchprobieren

} *wahrscheinlich klassische
Verschlüsselung*



Anwendungsbeispiele (4)

PSION-PDA: Entropie bestimmen, Kompressionstest

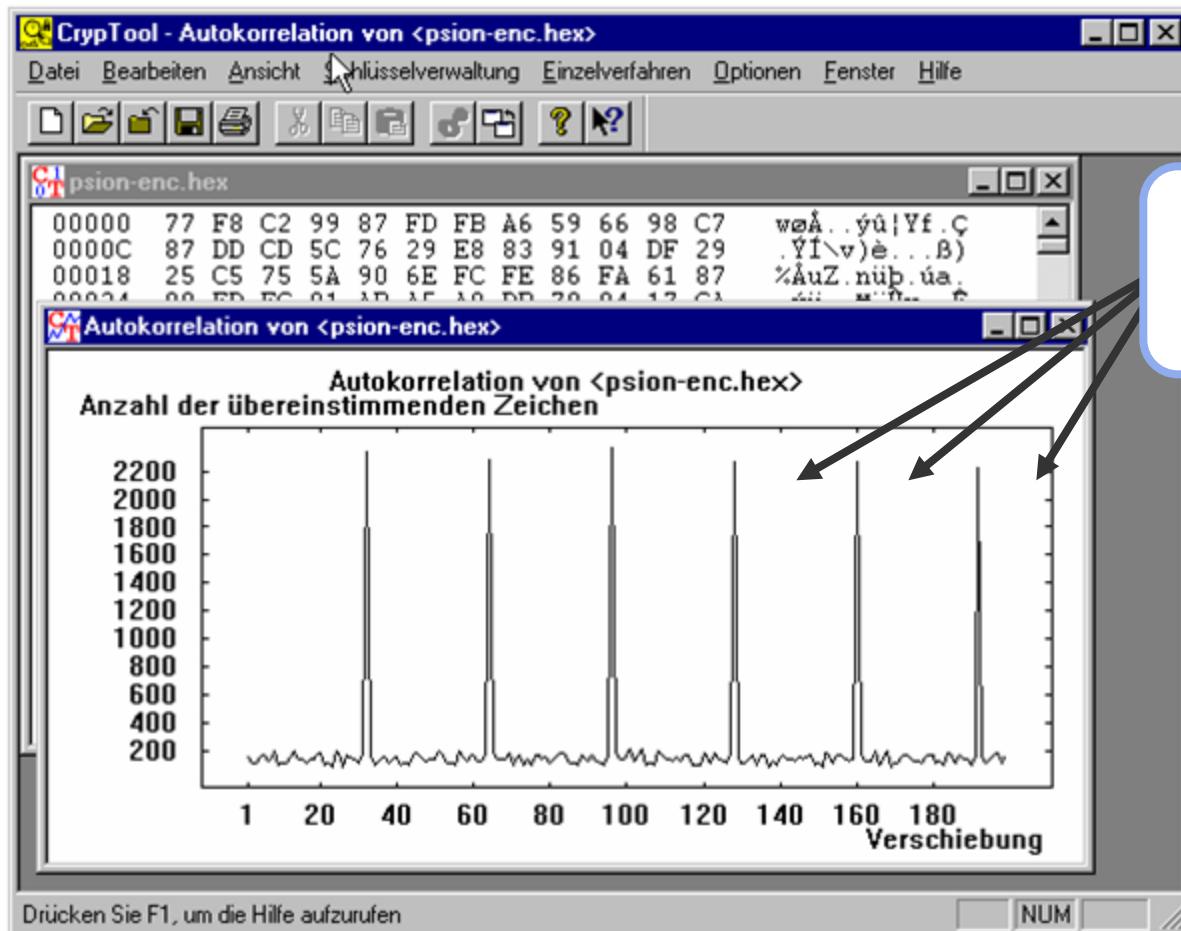


Komprimierbarkeit:
deutlicher Indikator für
schwache Kryptographie
(Größe wurde
um 21% reduziert)

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

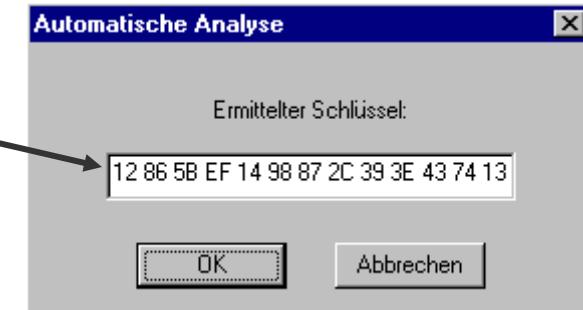
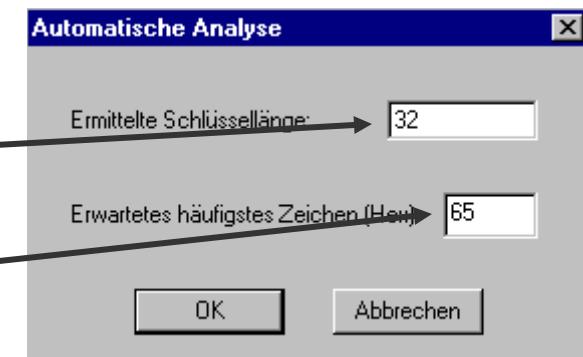
* Diese verschlüsselte Datei wird mit CrypTool ausgeliefert (siehe CrypTool\examples\psion-de-enc.hex)

Anwendungsbeispiele (4)

PSION-PDA: Automatische Analyse

Automatische Analyse mit

- Vigenère: kein Erfolg
- XOR: kein Erfolg
- Binärer 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
 - Ergebnis: gut, aber nicht perfekt

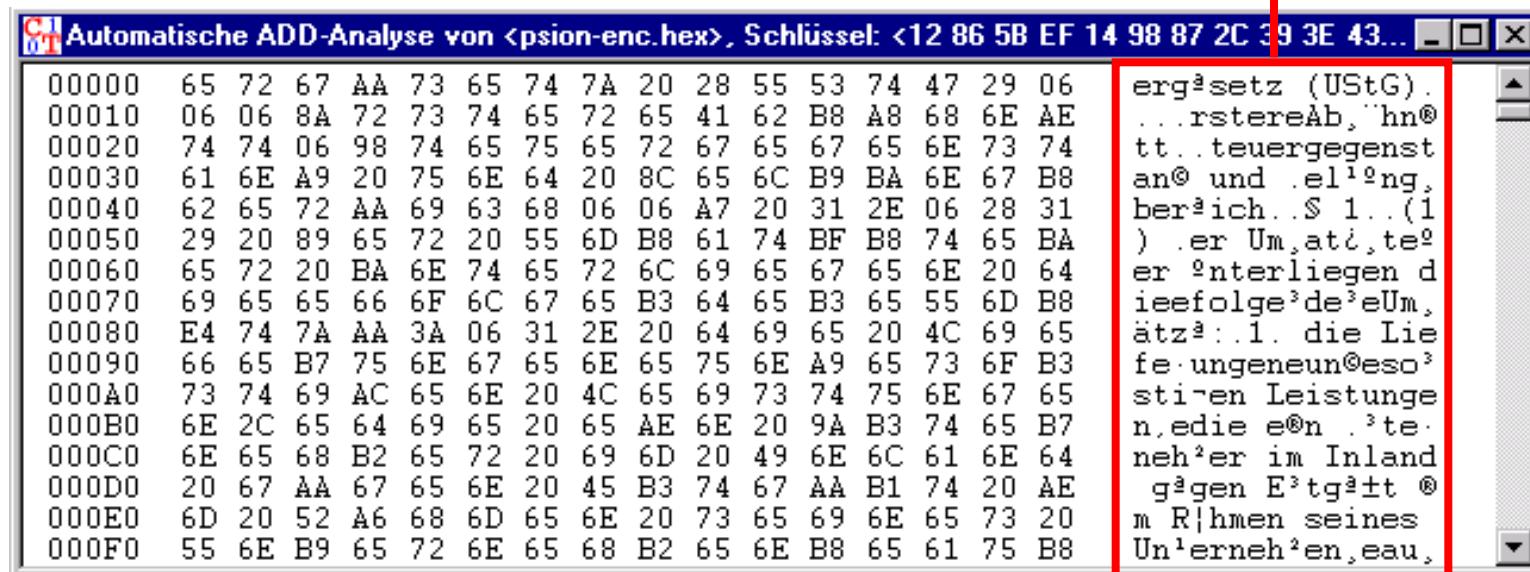


Anwendungsbeispiele (4)

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. 



| Adresse | Hex | ASCII |
|---------|---|-------|
| 000000 | 65 72 67 AA 73 65 74 7A 20 28 55 53 74 47 29 06 | |
| 000010 | 06 06 8A 72 73 74 65 72 65 41 62 B8 A8 68 6E AE | |
| 000020 | 74 74 06 98 74 65 75 65 72 67 65 67 65 6E 73 74 | |
| 000030 | 61 6E A9 20 75 6E 64 20 8C 65 6C B9 BA 6E 67 B8 | |
| 000040 | 62 65 72 AA 69 63 68 06 06 A7 20 31 2E 06 28 31 | |
| 000050 | 29 20 89 65 72 20 55 6D B8 61 74 BF B8 74 65 BA | |
| 000060 | 65 72 20 BA 6E 74 65 72 6C 69 65 67 65 6E 20 64 | |
| 000070 | 69 65 65 66 6F 6C 67 65 B3 64 65 B3 65 55 6D B8 | |
| 000080 | E4 74 7A AA 3A 06 31 2E 20 64 69 65 20 4C 69 65 | |
| 000090 | 66 65 B7 75 6E 67 65 6E 65 75 6E A9 65 73 6F B3 | |
| 0000A0 | 73 74 69 AC 65 6E 20 4C 65 69 73 74 75 6E 67 65 | |
| 0000B0 | 6E 2C 65 64 69 65 20 65 AE 6E 20 9A B3 74 65 B7 | |
| 0000C0 | 6E 65 68 B2 65 72 20 69 6D 20 49 6E 6C 61 6E 64 | |
| 0000D0 | 20 67 AA 67 65 6E 20 45 B3 74 67 AA B1 74 20 AE | |
| 0000E0 | 6D 20 52 A6 68 6D 65 6E 20 73 65 69 6E 65 73 20 | |
| 0000F0 | 55 6E B9 65 72 6E 65 68 B2 65 6E B8 65 61 75 B8 | |

- 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 (nicht abgebildet).

Anwendungsbeispiele (4)

PSION-PDA: Bestimmung der restlichen Schlüsselbytes

Schlüssel während der automatischen Analyse in die Zwischenablage kopieren

Im Hexdump der automatischen Analyse:

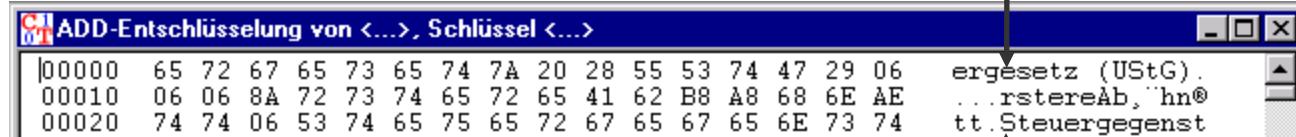
- Falsche Bytepositionen bestimmen, z.B. 0xAA an Position 3
- Korrespondierende korrekte Bytes erraten und notieren: „e“ = 0x65

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 12865B**34**1498872C393E43741396A45670235E111E907AB7C0841A2E8068E50
- Verschlüsseltes Ausgangsdokument mittels binärer Addition entschlüsseln
- Nun sind die Bytepositionen 3, 3+32, 3+2*32, ... ok



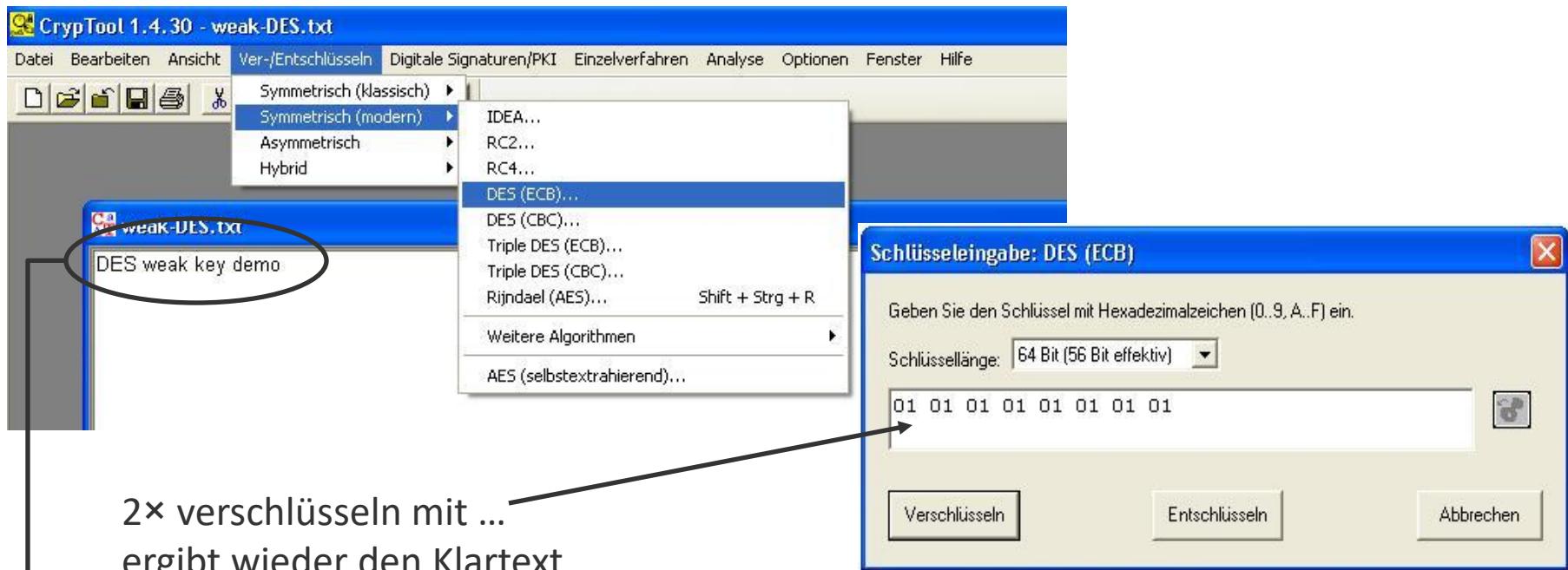
The screenshot shows a window titled "ADD-Entschlüsselung von <...>, Schlüssel <...>". It displays a hex dump of data from addresses 00000 to 00020. An arrow points from the text "ergesetz (UStG)." in the dump to the corresponding text "ergesetz (UStG)." in the decrypted text area. Another arrow points from the text "...rstereAb, hn®" to "...rstereAb, hn®". A third arrow points from the text "tt.Steuergegenst" to the same text in the decrypted text area.

| | 00000 | 65 | 72 | 67 | 65 | 73 | 65 | 74 | 7A | 20 | 28 | 55 | 53 | 74 | 47 | 29 | 06 | |
|-------|-------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|--|
| 00010 | 06 | 06 | 8A | 72 | 73 | 74 | 65 | 72 | 65 | 41 | 62 | B8 | A8 | 68 | 6E | AE | | |
| 00020 | 74 | 74 | 06 | 53 | 74 | 65 | 75 | 65 | 72 | 67 | 65 | 67 | 65 | 6E | 73 | 74 | | |

ergesetz (UStG).
...rstereAb, hn®
tt.Steuergegenst

Anwendungsbeispiele (5)

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



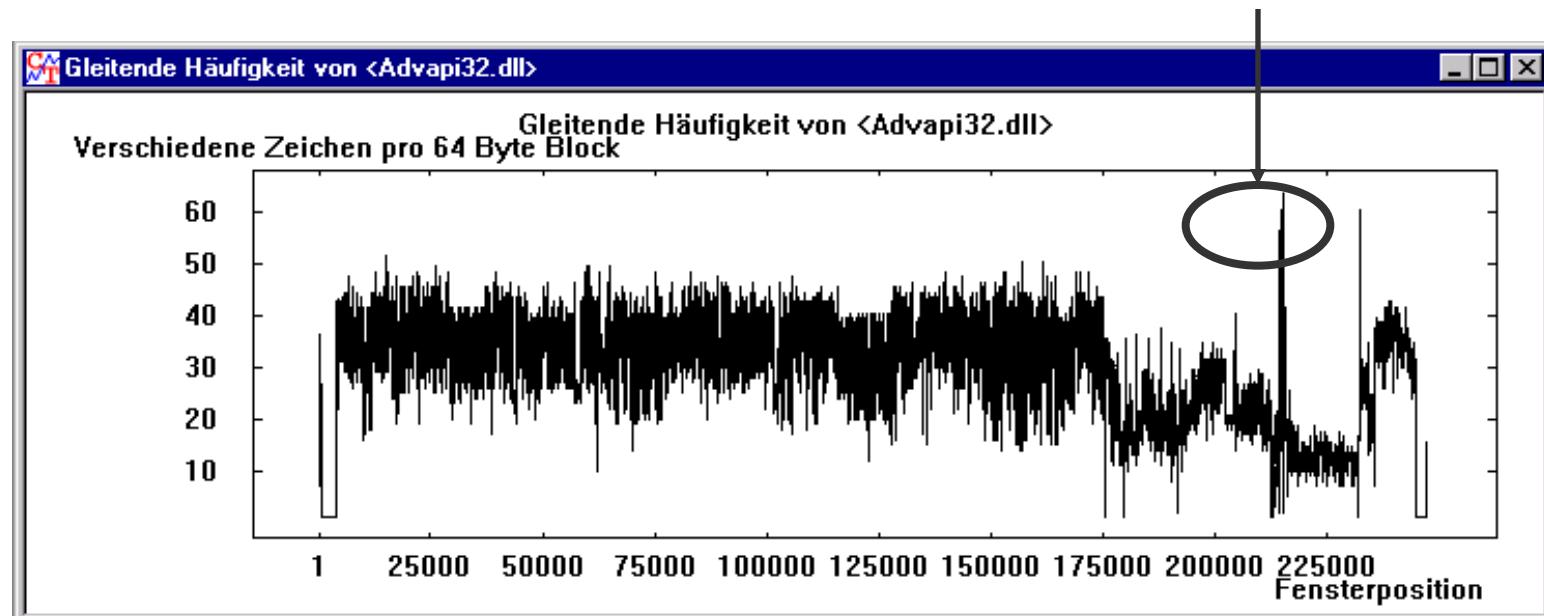
Anwendungsbeispiele (6)

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 „NSA-Key“ in advapi32.dll (Windows NT)



Anwendungsbeispiele (6)

Vergleich der gleitenden Häufigkeit anderer Dateien

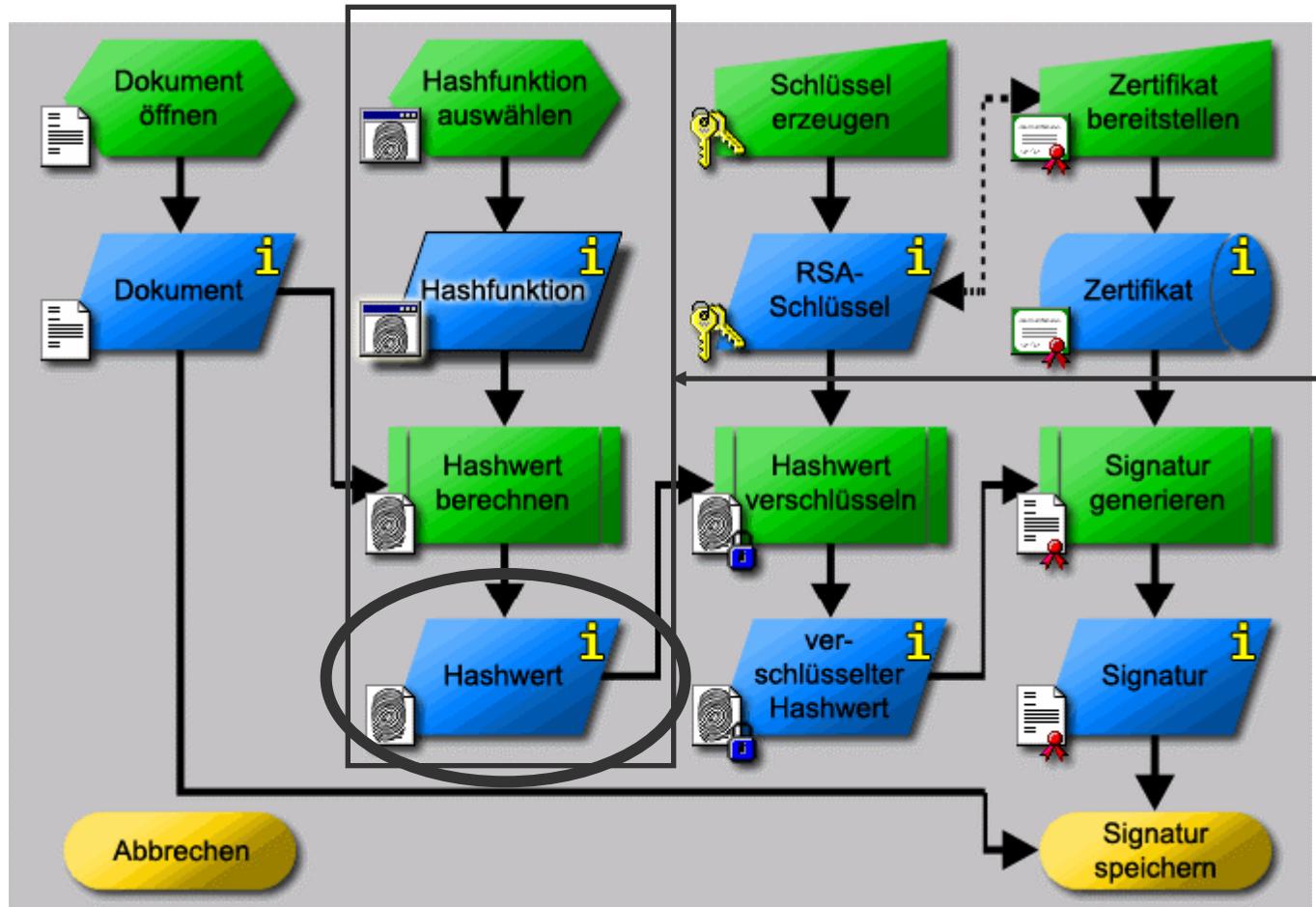
Screenshot of the CrypTool application showing four windows comparing the sliding frequency of different files:

- CryptTool - Gleitende Häufigkeit von <AA_Cry-Rijndael-startbeispiel.de.hex>**: Hex dump of the file AA_Cry-Rijndael-startbeispiel.de.
- Gleitende Häufigkeit von <AA_Cry-startbeispiel.txt>**: Sliding frequency plot for AA_Cry-startbeispiel.txt. The Y-axis represents character frequency from 20 to 30, and the X-axis represents window position from 1 to 600. The plot shows a relatively flat distribution with some minor fluctuations.
- Gleitende Häufigkeit von <AA_Cry-ZIP-AA_Cry-startbeispiel.hex>**: Sliding frequency plot for AA_Cry-ZIP-AA_Cry-startbeispiel.hex. The Y-axis represents character frequency from 53 to 61, and the X-axis represents window position from 1 to 300. The plot shows a more distinct pattern with higher frequencies (around 58-61) appearing at various intervals.
- Gleitende Häufigkeit von <AA_Cry-Rijndael-startbeispiel-de.hex>**: Sliding frequency plot for AA_Cry-Rijndael-startbeispiel-de. The Y-axis represents character frequency from 52 to 61, and the X-axis represents window position from 1 to 700. This plot shows a highly fluctuating and non-uniform distribution compared to the others.

Drücken Sie F1, um die Hilfe aufzurufen

Anwendungsbeispiele (7)

Angriff auf digitale Signatur



Angriff:
Finde zwei
Nachrichten
mit dem
gleichen
Hashwert !

Menü: „Analyse“ \ „Hashverfahren“ \ „Angriff auf den Hashwert einer digitalen Signatur“

Anwendungsbeispiele (7)

Angriff auf digitale Signatur: Idee (1)

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

Idee

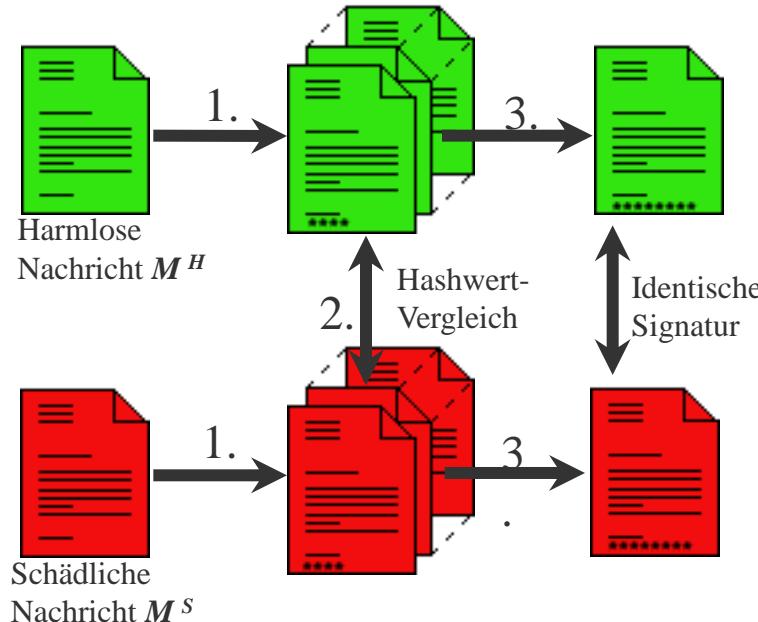
- ASCII-Text kann mittels **nicht-druckbarer** Zeichen modifiziert werden, ohne den lesbaren Inhalt zu verändern
- Modifizierte parallel zwei Texte, bis eine Hashkollision erreicht wird
- Ausnutzung des Geburtstagsparadoxons (Geburtstagsangriff)
- Generischer Angriff auf beliebige Hashfunktion
- In CrypTool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

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 :
- Kollisionssuche zwischen M_1^H, \dots, M_N^H und M_1^S, \dots, M_N^S :

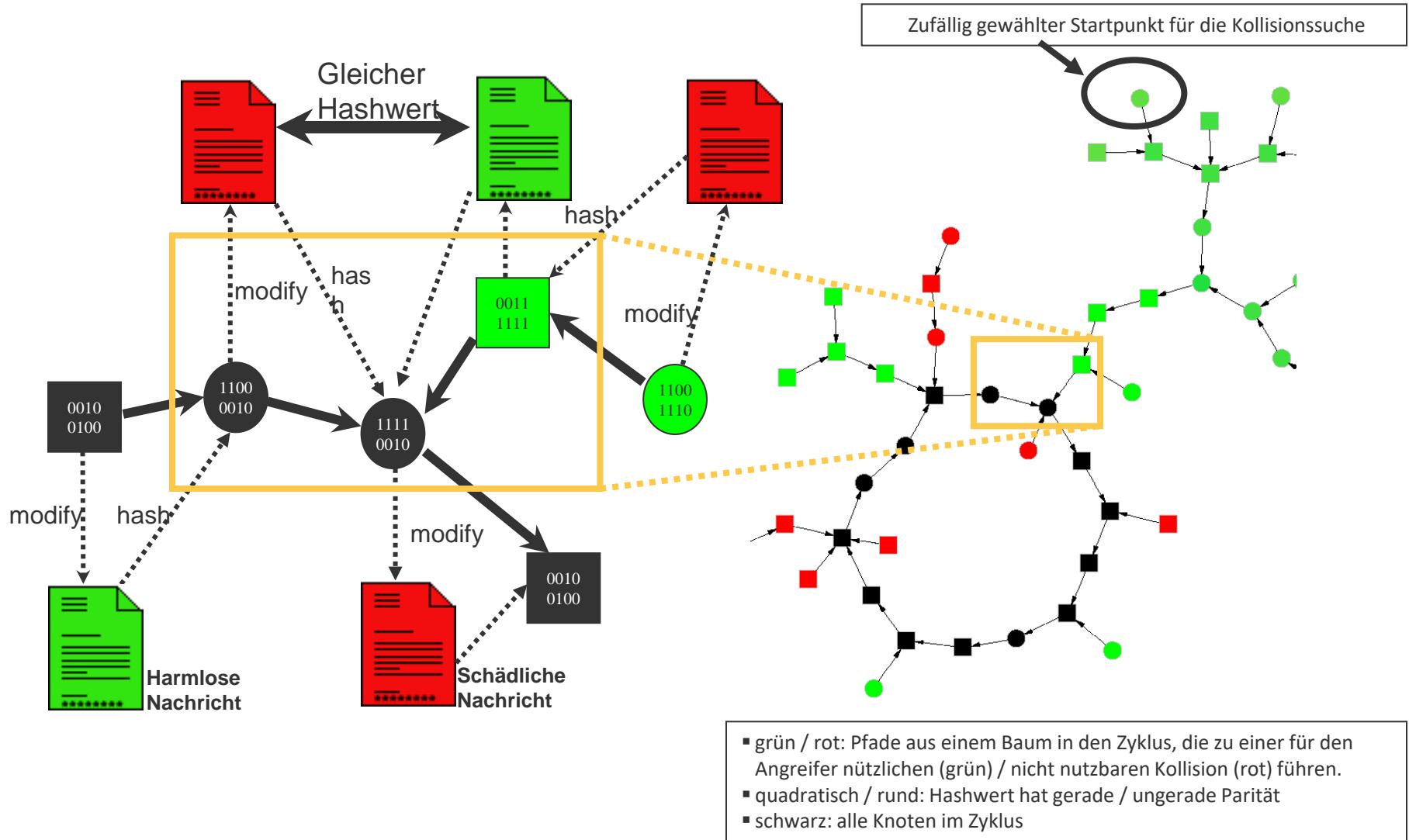
$$N \approx 2^n$$

$$N \approx 2^{n/2}$$

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

Hashkollisionssuche (1)

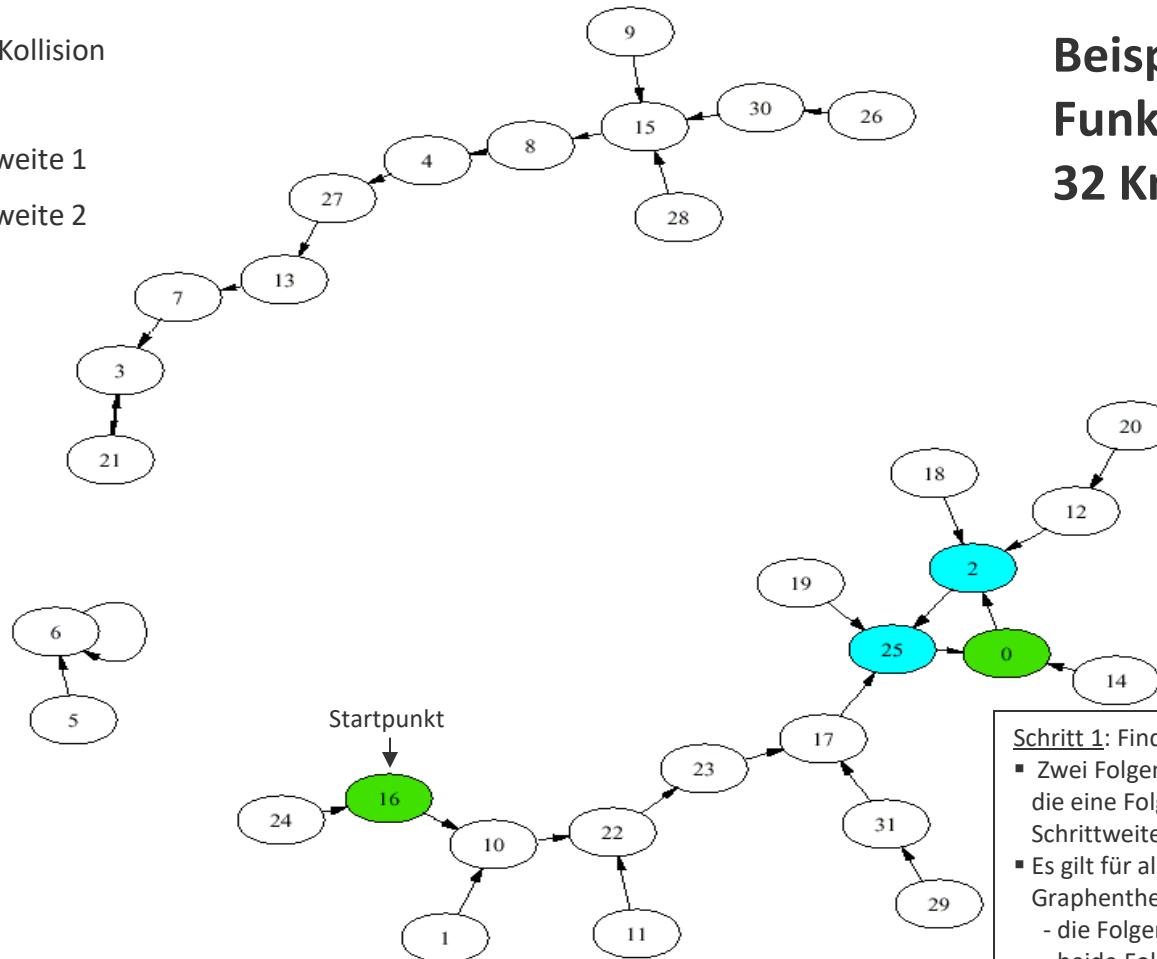
Mapping durch Textmodifikation



Hashkollisionssuche (2)

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 (3)

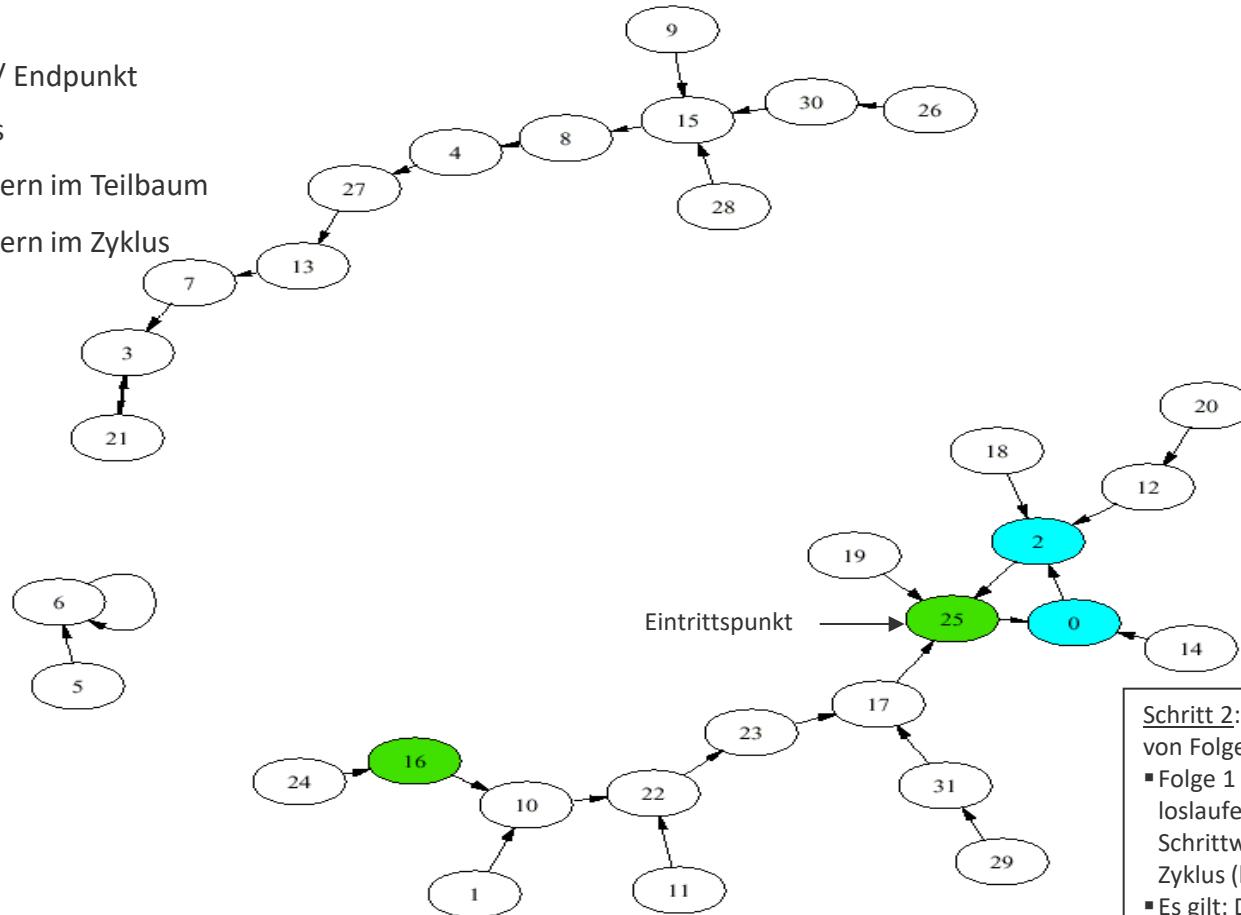
Eintritt in den Zyklus (Erweiterung von Floyd): Finde Eintrittspunkt

 Start / Endpunkt

 Zyklus

 Wandern im Teilbaum

 Wandern im Zyklus

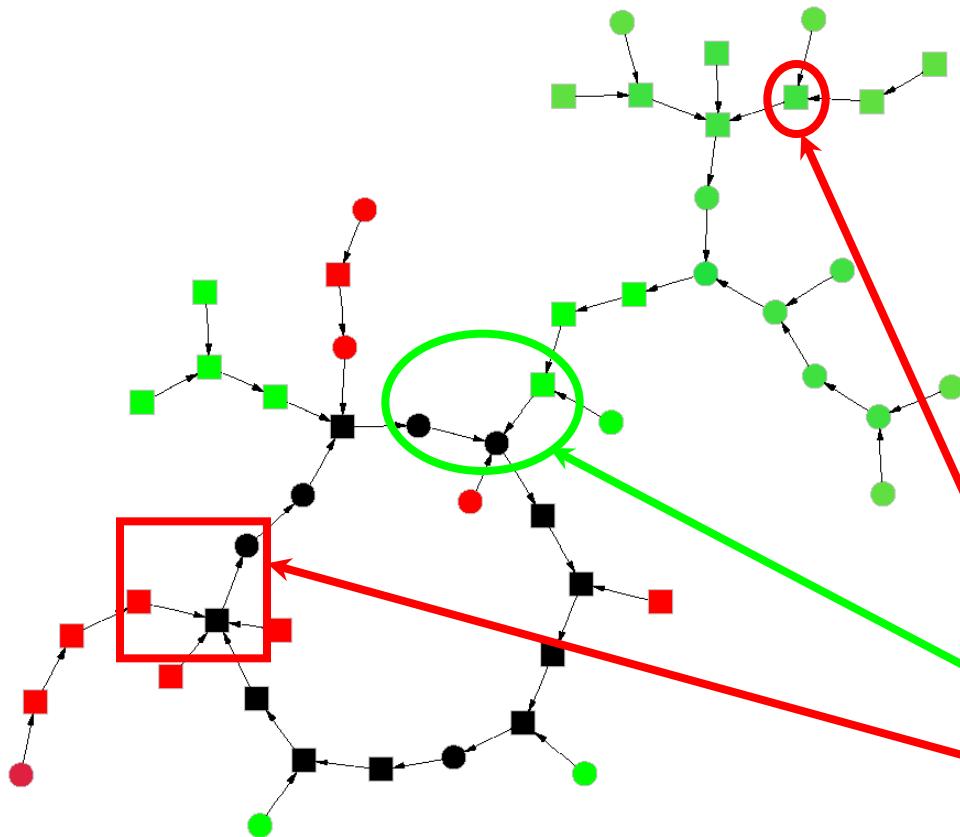


Eintrittspunkt

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
■ Die Vorgänger (hier 17 und 2) liefern die Hashkollision.

Hashkollisionssuche (4)

Geburtstagsangriff auf die digitale Signatur



Auseinandersetzung mit dem Floyd-Algorithmus

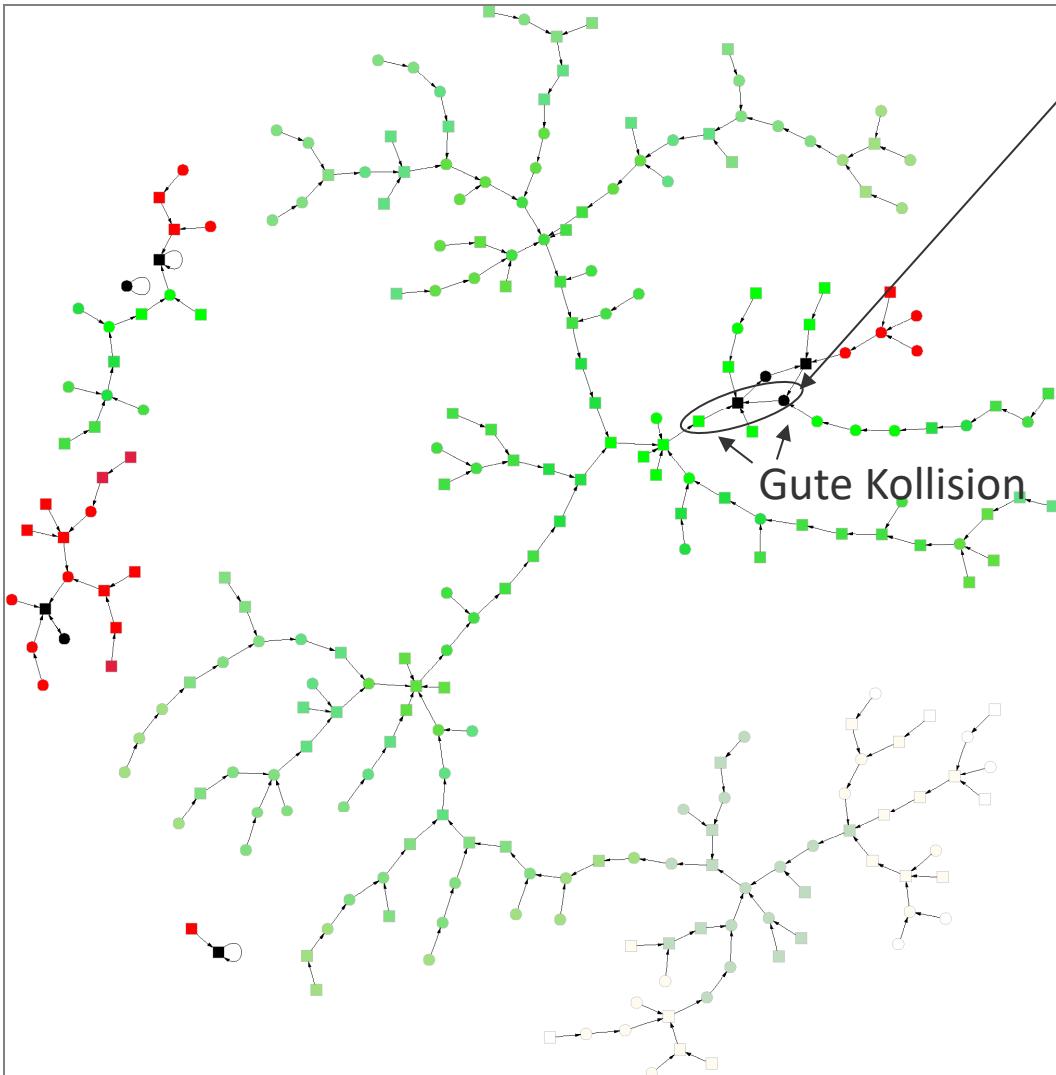
- Visuelle & interaktive Darstellung des Floyd- Algorithmus („Wanderung im Mapping“ in einen Zyklus hinein).*
- Adaption des Floyd- Algorithmus für den Signaturangriff.

Startpunkt
Gute Kollision
Schlechte Kollision

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

Anwendungsbeispiele (7)

Angriff auf digitale Signatur

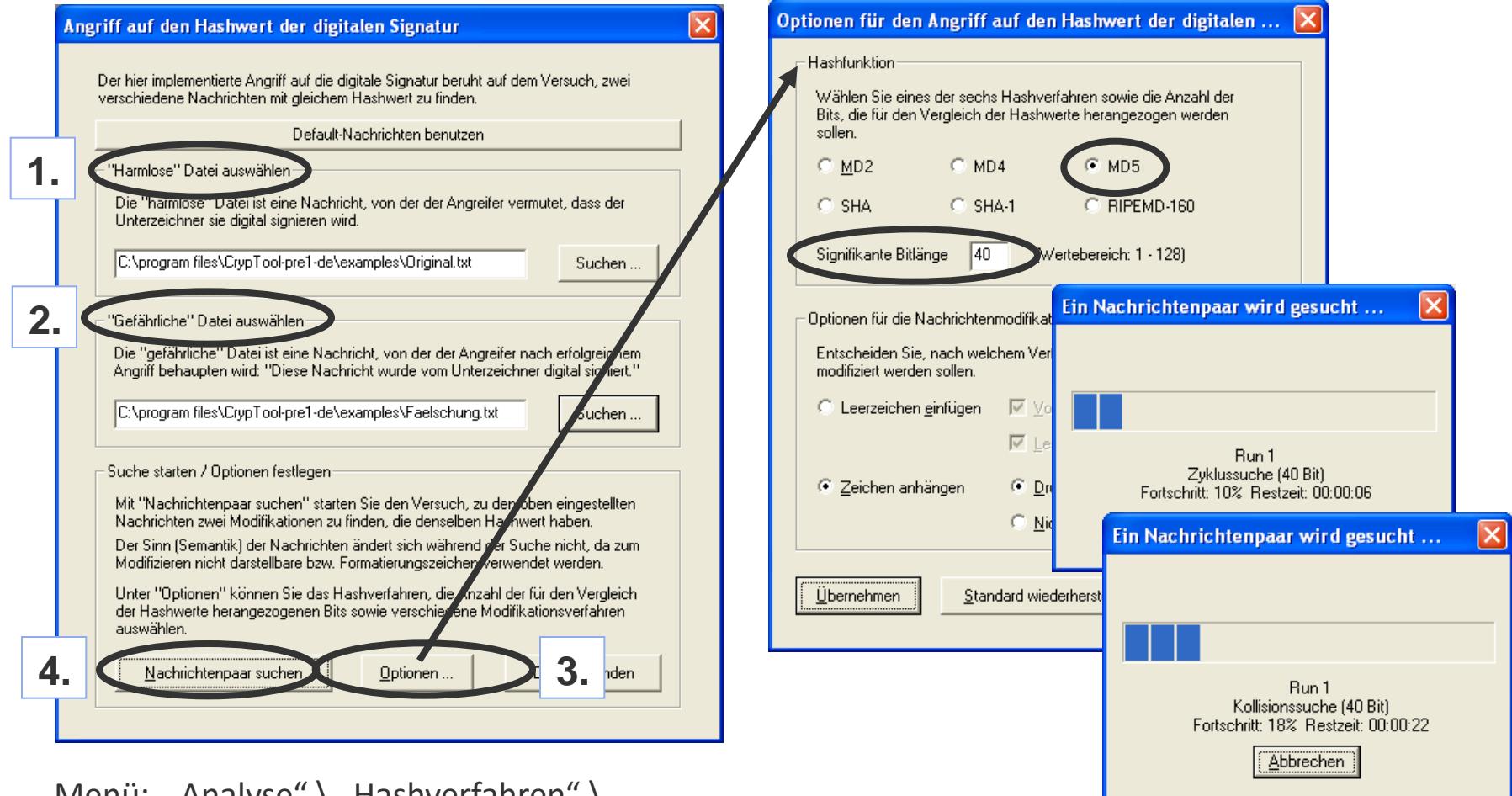


Ein Beispiel für ein „gutartiges“ Mapping (fast alle Knoten darin sind grün gefärbt). In diesem Graphen gehören die meisten Knoten zu einem großen Baum, der in den Zyklus mit einem geraden Hashwert gelangt und wo der Eintrittspunkt-Vorgänger im Zyklus ungerade ist. D.h. der Angreifer findet für fast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



Anwendungsbeispiele (7)

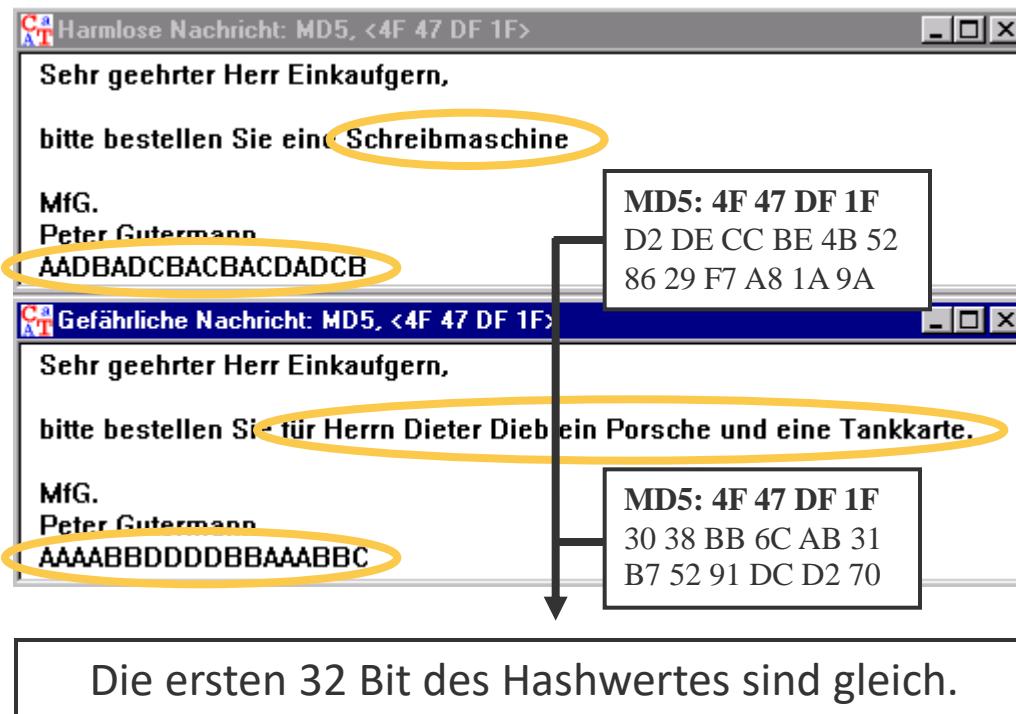
Angriff auf digitale Signatur: Durchführung



Menü: „Analyse“ \ „Hashverfahren“ \ „Angriff auf den Hashwert einer digitalen Signatur“

Anwendungsbeispiele (7)

Angriff auf digitale Signatur: Ergebnisse



Zusätzlich zur interaktiven Bedienung

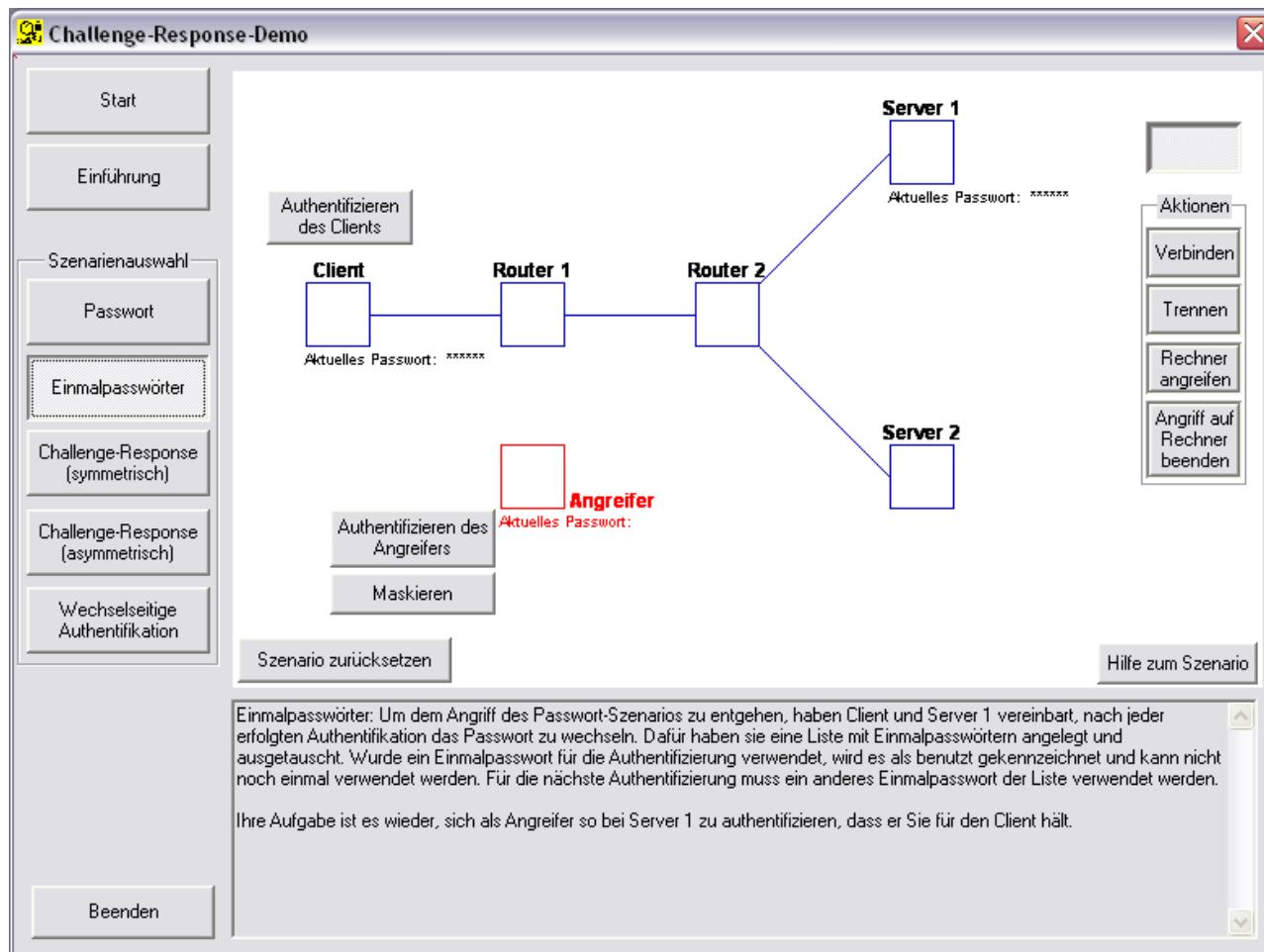
Automatisierte Offline-Funktion in CrypTool: Durchspielen und Loggen der Ergebnisse für ganze Sets von Parameterkonfigurationen. Möglich durch entsprechenden Aufruf von CrypTool über die Eingabeaufforderung.

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 mit massiv parallelen Verfahren angreifbar!
- Es sollten Hashwerte mit mindestens 160 Bit verwendet werden.

Anwendungsbeispiele (8)

Authentifizierung in einer Client-Server-Umgebung

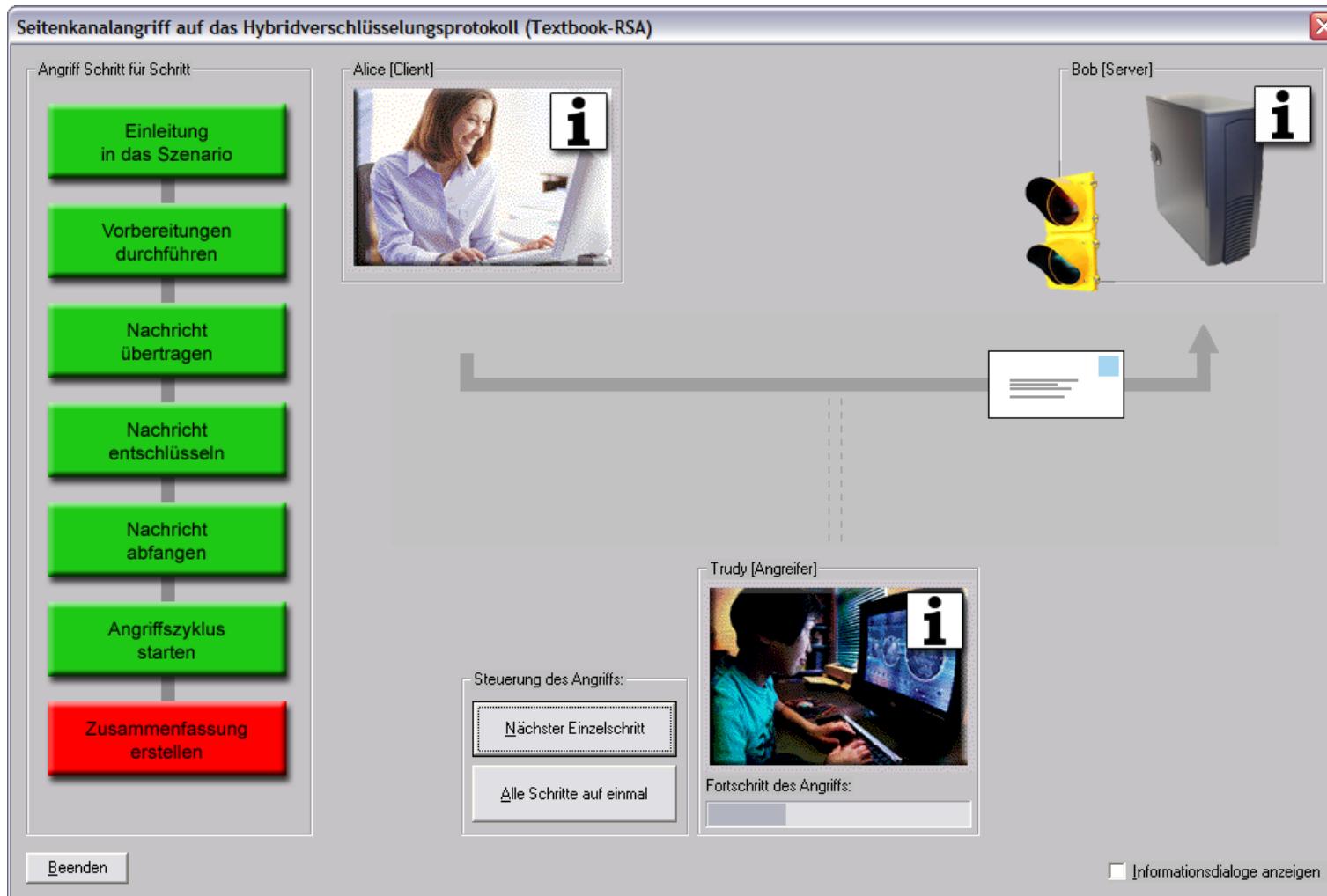


- Interaktive Demo für verschiedene Authentifizierungsverfahren.
- Definierte Möglichkeiten des Angreifers.
- Sie können in die Rolle eines Angreifers schlüpfen.
- **Lerneffekt**
Nur die wechselseitige Authentifizierung ist sicher.

Menü: „Einzelverfahren“ \ „Protokolle“ \ „Authentisierungsverfahren im Netz“

Anwendungsbeispiele (9)

Demonstration eines Seitenkanalangriffes (auf ein Hybridverschlüsselungsprotokoll)



Menü: „Analyse“ \ „Asymmetrische Verfahren“ \ „Seitenkanalangriff auf Textbook-RSA“

Anwendungsbeispiele (9)

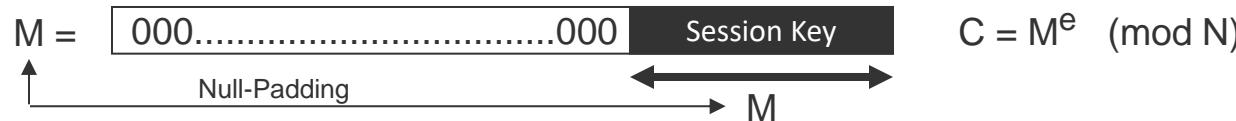
Idee zu diesem Seitenkanalangriff

Ulrich Kühn: „Side-channel attacks on textbook RSA and ElGamal encryption“, 2003

Voraussetzungen [CCA-Angriff (Chosen-ciphertext attack) auf Dechiffrier-Orakel]

- RSA-Verschlüsselung: $C = M^e \pmod{N}$ und Entschlüsselung: $M = C^d \pmod{N}$.
- 128-Bit Sessionkeys (in M) werden „Textbuch-RSA“ verschlüsselt (Null-Padding).
- Der Server kennt den geheimen Schlüssel d und
 - benutzt nach der Entschlüsselung nur die 128 niederwertigsten Bit (keine Überprüfung der Null-Padding-Bit) (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.

Angriffsseite: 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: 300 delta: 0,2600 Standard-Schlüsselparameter setzen

N: 857073798346215184382161530544361373050019640299447938271693996822747113271714535706
e: 218305208015393063472478474434063284822795207914943862442679226243449153095110094950
d: 241315232572409146883257

Zufälligen RSA-Schlüssel erzeugen

Schritt 2: Angriffsparameter für das Gitterreduktionsverfahren eingeben

m: 4 Bestimmt die Größe des zu reduzierenden Gitters und die maximale Größe von delta. Sollte mindestens den Wert 4 haben.
t: 2 Wird abhängig von m optimal bestimmt.
Gitterdimension: 15 Größe des zu reduzierenden Gitters. Bestimmt maßgeblich die Laufzeit.
Maximales delta: 0,2653 Maximale Größe von delta für große N (N>1000 Bit).

Schritt 3: Angriff starten

Erzeuge Gitter: 0h 0m 0s Reduktionen: 6237 Starten
Reduziere Gitter: 0h 0m 1s Resultante: 1 Abbrechen
Bilde Resultante: 0h 0m 2s
Gesamtzeit: 0h 0m 5s

Gefundene Faktorisierung:
p: 1037689133569485779480551961279977316026994707 q: 825944659744115722584120500232353574453988321

Logdatei ausgeben Dialog schließen

- Veranschaulicht, wie die Parameter des RSA-Verfahrens beschaffen sein müssen, damit sie den aktuellen, auf Gitterreduktion beruhenden Angriffen aus der Literatur standhalten.
- Drei Varianten**, die *nicht* standhalten
 - Der geheime Exponent d ist im Verhältnis zu N zu klein.
 - Einer der Faktoren von N ist teilweise bekannt.
 - Ein Teil des Klartextes ist bekannt.
- Diese Annahmen sind realistisch.

Menü: „Analyse“ \ „Asymmetrische Verfahren“ \ „Gitterbasierte Angriffe auf RSA“ \ ...

Anwendungsbeispiele (11)

Zufallsanalyse mit 3-D-Visualisierung

3-D Visualisierung zur Analyse von Zufallszahlen

Beispiel 1

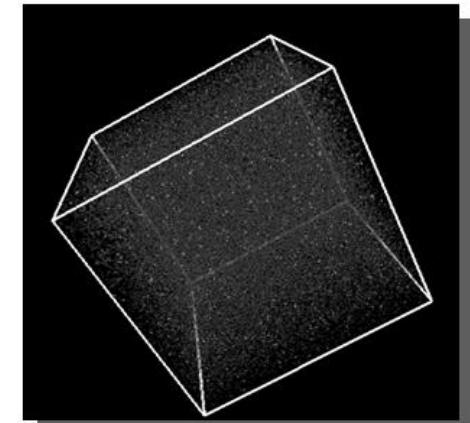
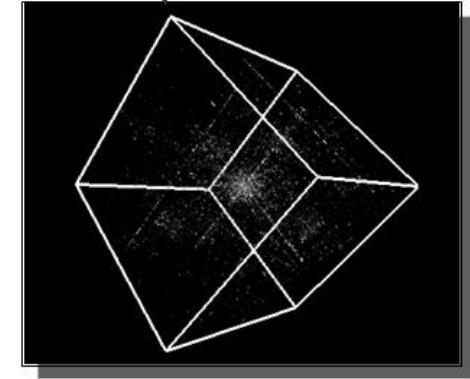
- Öffnen einer beliebigen Datei (z.B. Bericht in Word oder PowerPoint-Präsentation)
- Es empfiehlt sich eine zumindest 100 KB große Datei zu wählen
- 3-D-Analyse
- Ergebnis: **Strukturen sind offensichtlich erkennbar**

Beispiel 2

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

Menü: „Analyse“ \ „Zufallsanalyse“ \ „3-D-Visualisierung“

Sie können den Würfel mit der Maus so drehen, wie Sie wünschen.



Anwendungsbeispiele (12)

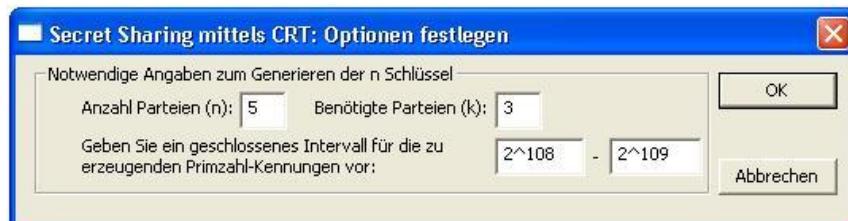
Secret Sharing mittels CRT – Implementierung des Chinesischen Restsatzverfahrens

Secret Sharing Beispiel (1)

■ Problemstellung

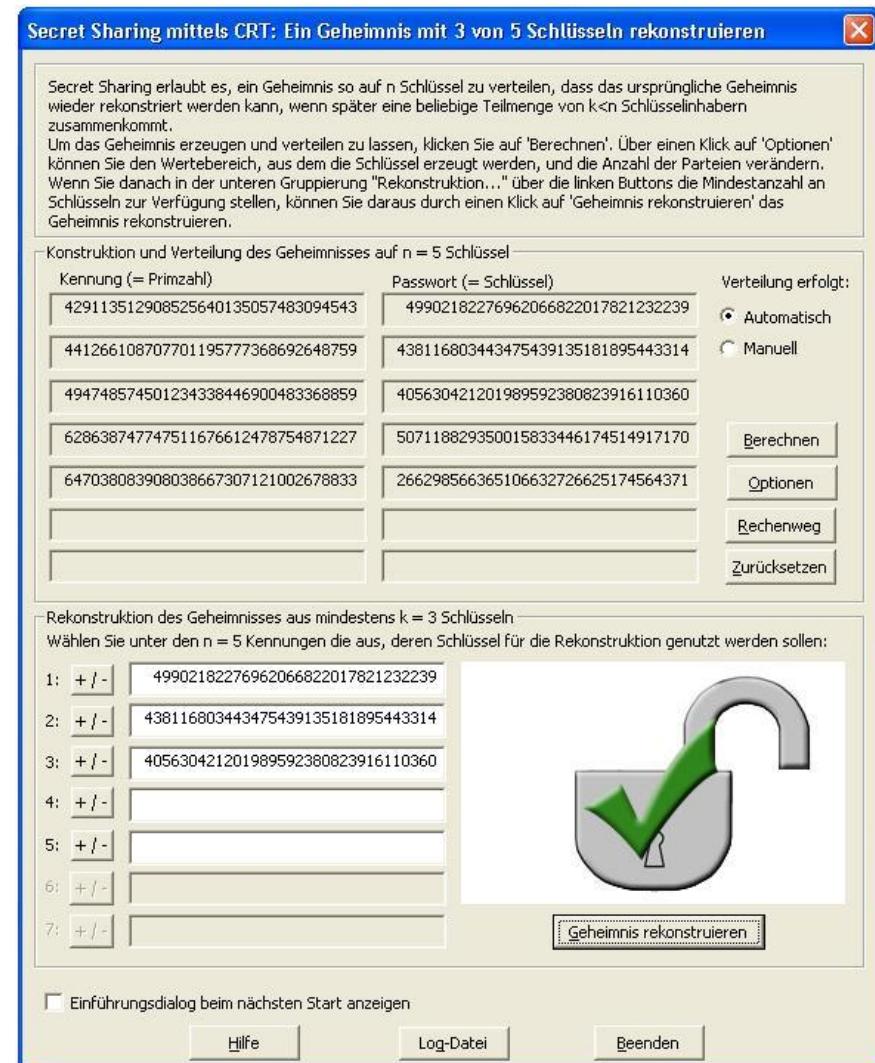
- 5 Personen erhalten jeweils einen Schlüssel
- Um Zugriff zu erlangen, müssen mindestens 3 der 5 Personen anwesend sein

■ „Optionen“ ermöglicht weitere Details des Verfahrens einzustellen.



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

Menü: „Einzelverfahren“ \ „Anwendungen des Chinesischen Restsatzverfahrens“ \ „Secret Sharing mittels CRT“



Anwendungsbeispiele (12)

Secret Sharing mittels Schwellenwertschema von Shamir

Secret Sharing Beispiel (2)

■ Problemstellung

- Ein geheimer Wert soll unter n Personen aufgeteilt werden.
- t von n Personen sind notwendig um den geheimen Wert wiederherzustellen.
- (t, n) Schwellenwertschema

■ Durchführen im Dialog:

1. Angabe des Geheimnisses K, sowie Anzahl der Teilnehmer n und Schwellenwert t
2. Polynom generieren
3. Parameter übernehmen
4. Mit dem Button „Rekonstruktion“ wird das eingegebene Geheimnis wiederhergestellt.

Menü: „Einzelverfahren“ \
 „Secret-Sharing-Demo (nach Shamir)“

Secret Sharing: Aufsetzen eines Schwellenwertschemas

Mit dem (t, n)-Schwellenwertschema nach Shamir ist es möglich, ein Geheimnis S 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 Steuerparameter wählen (ganze Zahlen)

| | | |
|---|------|--------------------------|
| Geheimnis S mit $S \geq 0$ | 1244 | Beispielparameter setzen |
| Anzahl der Teilnehmer n mit $n > 0$ | 8 | Optionen |
| Schwellenwert (Mindestanzahl) t mit $t > 0$ | 3 | |

Parameter des Polynoms $f(x)$ vom Grad t-1
Alle Operationen finden im diskreten Raum $GF(p)$ statt.

Polynom $f(x)$

Primzahl p Parameter übernehmen

Aus den Parametern berechnete Werte der Teilnehmer:

| Teilnehmer | Öffentlicher Wert x | Share (geheimer Wert f(x)) |
|--|---------------------|----------------------------|
| <input checked="" type="checkbox"/> Teilnehmer 1 | 642 | 132 |
| <input type="checkbox"/> Teilnehmer 2 | 2207 | 2241 |
| <input checked="" type="checkbox"/> Teilnehmer 3 | 2307 | 2446 |
| <input type="checkbox"/> Teilnehmer 4 | 2201 | 865 |
| <input checked="" type="checkbox"/> Teilnehmer 5 | 1275 | 1533 |
| <input type="checkbox"/> Teilnehmer 6 | 1067 | 54 |
| <input type="checkbox"/> Teilnehmer 7 | 1456 | 2515 |
| <input type="checkbox"/> Teilnehmer 8 | 2863 | 121 |

Bitte wählen Sie die Teilnehmer, die das Geheimnis wiederherstellen sollen, aus der Liste aus, indem Sie die entsprechenden Checkboxen ankreuzen.

Informationsdialog zu Beginn anzeigen

Abbrechen Rekonstruktion

Anwendungsbeispiele (13)

Anwendung des CRT in der Astronomie (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

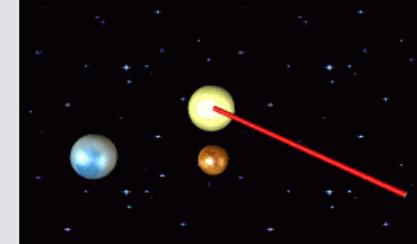
Mit dem Chinesischen Restsatz (CRT) kann man lineare modulare Gleichungssysteme lösen. Unten können Sie 9 Gleichungen der Form $x = a[i] \text{ mod } m[i]$ ($i=1, \dots, 9$) eingeben und anschließend lösen. Solche Gleichungssysteme kann man z.B. nutzen, um herauszufinden, in wie viel Tagen bestimmte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

Simultane Kongruenzen/ Modulare lineare Gleichungssysteme

| | |
|-----------|-----------|
| $x = 15$ | mod 88 |
| $x = $ | mod |
| $x = 100$ | mod 365 |
| $x = $ | mod |
| $x = 0$ | mod 4327 |
| $x = $ | mod |
| $x = $ | mod |
| $x = 0$ | mod 60149 |
| $x = $ | mod |

Lösung: 126.228.390.655

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

Menü: „Einzelverfahren“ \ „Anwendungen des Chinesischen Restsatzverfahrens“ \ „Astronomie und Planetenbewegung“

Anwendungsbeispiele (14)

Visualisierung von symmetrischen Verschlüsselungsverfahren mit ANIMAL (1)

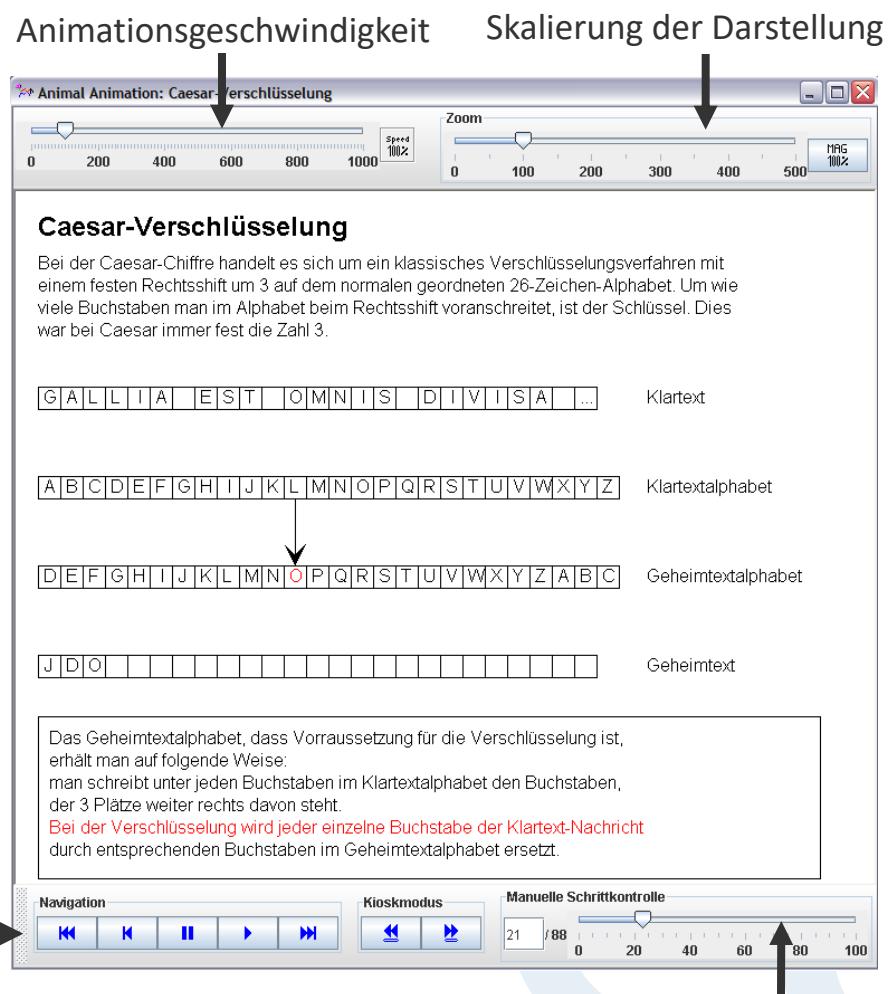
Animierte Darstellung verschiedener symmetrischer Verfahren

- Caesar
- Vigenère
- Nihilist
- DES

CrypTool

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

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



Anwendungsbeispiele (14)

Visualisierung von symmetrischen Verschlüsselungsverfahren mit ANIMAL (2)

Visualisierung der DES-Verschlüsselung

Animal Animation: DES Data Encryption Standard (ECB-Modus)

Eingabeblock X: $\begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ \vdots & \vdots \end{bmatrix}$

Schlüssel K: $\begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \vdots & \vdots \end{bmatrix}$

K' und PC2:

| | | |
|-------------------|-----------------|-------------------|
| 1 0 1 1 1 0 0 0 | 0 0 1 1 0 0 0 0 | 14 17 11 24 1 5 |
| 1 0 1 1 1 0 0 1 | 0 0 1 1 0 0 0 1 | 3 28 15 6 21 10 |
| 0 0 0 1 1 0 1 1 | 1 1 1 1 1 0 0 1 | 23 19 12 4 26 8 |
| 1 0 1 0 0 0 1 0 | 1 1 1 1 0 0 0 1 | 16 7 27 20 13 2 |
| 0 0 1 1 1 0 1 0 | 0 1 1 1 1 0 0 1 | 41 52 31 37 47 55 |
| 1 0 0 1 0 0 0 1 0 | 0 1 0 0 0 0 0 1 | 30 40 51 45 33 48 |
| 1 1 0 0 0 0 0 1 0 | 0 1 0 0 0 0 1 1 | 44 49 39 56 34 53 |
| 1 1 1 0 0 0 0 1 1 | 0 0 1 1 0 0 0 1 | 46 42 50 36 29 32 |

Schema:

```
graph TD
    A[Eingabeblock X] --> B[IP]
    B --> C[Permutierte Eingabe]
    D[Schlüssel K] --> E[PC1]
    E --> F[PC2(K)]
    C --> G[16 DES-Runden]
    F --> G
    G --> H[Pre-Ausgabe]
    H --> I[IP^-1]
    I --> J[Ausgabeblock Y]
```

Nun werden aus dem 56-Bit Schlüssel K' 16 48-Bit Teilschlüssel K[1] bis K[16] abgeleitet.
Dazu wird dieser zunächst in zwei 28-Bit-Sätze L und R unterteilt.
L und R werden in 16 Runden gemäß folgender Tabelle linksrotiert:

| Runde Nr. | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|------------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|
| Bits rotieren um | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 1 |

L und R werden wieder zusammengefügt und gemäß PC2 in bekannter Weise permutiert.

Navigation: ⏪ ⏴ ⏵ ⏶ ⏷ ⏸ ⏹ ⏻ ⏺ ⏻ ⏸ ⏷ ⏵ ⏴ ⏶ ⏪
Kioskmodus: ⏪ ⏴ ⏵ ⏶ ⏷ ⏸ ⏹ ⏻ ⏸ ⏷ ⏵ ⏴ ⏶ ⏪
Manuelle Schrittkontrolle: 169 / 424

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

Animal Animation: DES Data Encryption Standard (ECB-Modus)

Die Funktion f :

$B[1] = 1 \cdot 2^1 + 0 \cdot 2^0 = 2$

$B[8] = 1 \cdot 2^8 + 0 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 = 11$

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

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

Nun wird jeweils eine Zahl von jeder der 8 S-Boxen auf folgende Weise extrahiert:
Für alle $S[n]$ repräsentieren das erste und das letzte Bit von $B[n]$ den Binärwert für die Spalte und die mittleren 4 Bits den Wert für die Reihe der jeweiligen Box.
Es folgt das Beispiel für $S[1], B[1]$ und $S[8], B[8]$. Alle anderen $B[n]$ werden analog substituiert.
Umrechnung des Binärwerts in Dezimal.

Navigation: ⏪ ⏴ ⏵ ⏶ ⏷ ⏸ ⏹ ⏻ ⏸ ⏷ ⏵ ⏴ ⏶ ⏪
Kioskmodus: ⏪ ⏴ ⏵ ⏶ ⏷ ⏸ ⏹ ⏻ ⏸ ⏷ ⏵ ⏴ ⏶ ⏪
Manuelle Schrittkontrolle: 294 / 424

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

Anwendungsbeispiele (15)

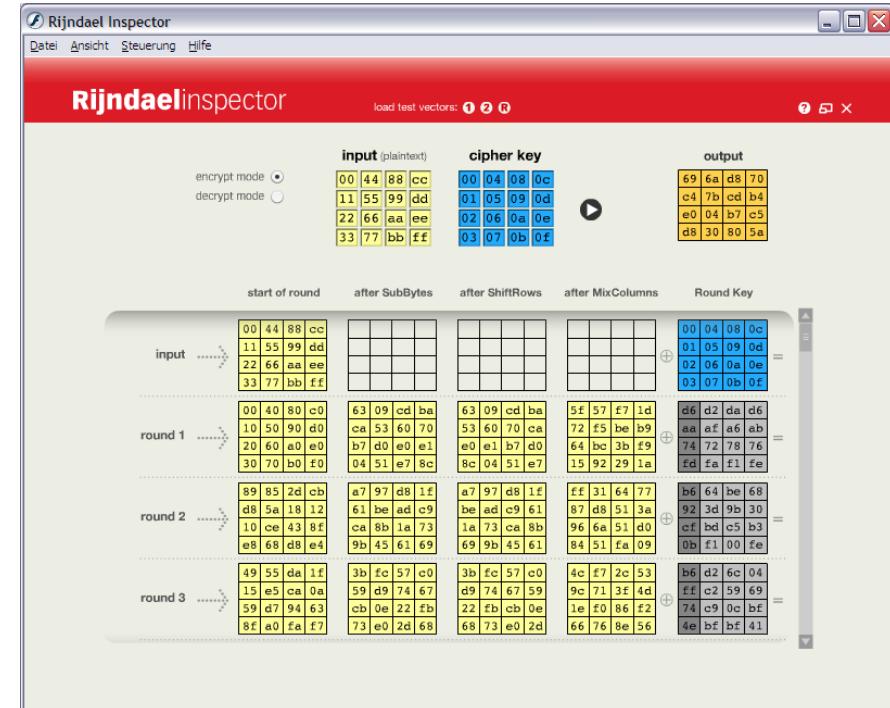
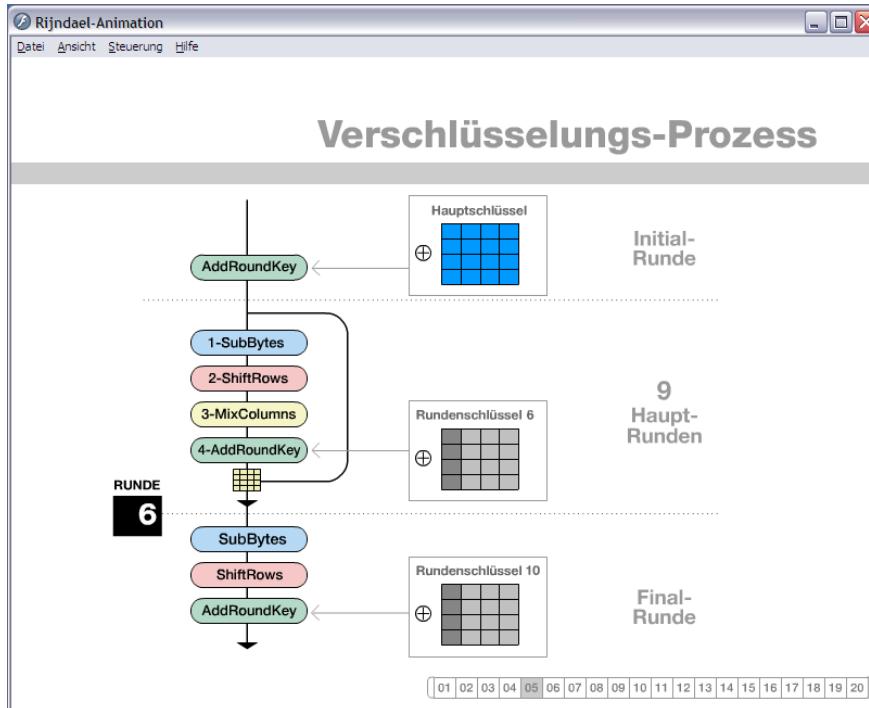
Visualisierungen von AES (Rijndael-Chiffre) – mit Flash

Rijndael-Animation (die Rijndael-Chiffre war Gewinner der AES-Ausschreibung)

- Zeigt die Verschlüsselungsprozesse in jeder Runde (mit fixen Ausgangsdaten)

Rijndael-Inspector

- Zum Ausprobieren mit eigenen Daten (zeigt den Inhalt der Datenmatrix in jeder Runde)



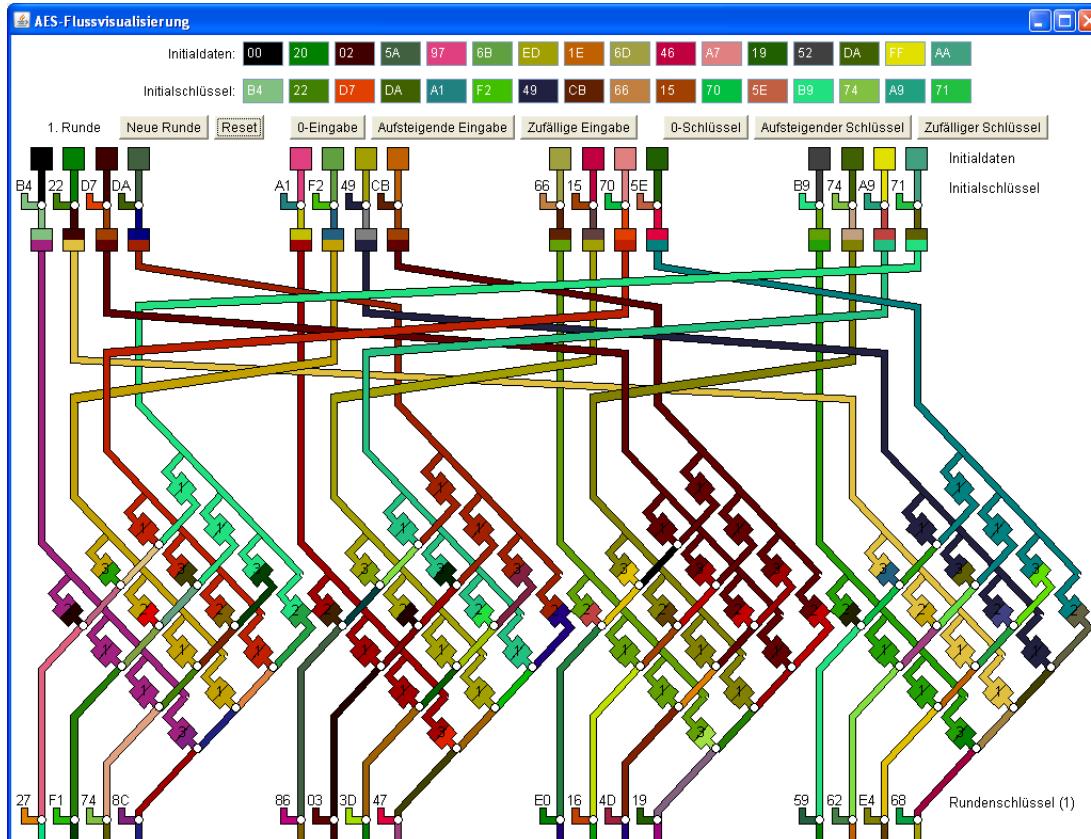
Menü: „Einzelverfahren“ \ „Visualisierung von Algorithmen“ \ „AES“ \ „Rijndael-Animation“ bzw. „Rijndael-Inspector“

Anwendungsbeispiele (15)

Flussvisualisierungen von AES (Rijndael-Chiffre) – mit Java

Rijndael-Flussvisualisierung

- Visualisierung der Datenveränderung je Runde durch Farbverläufe

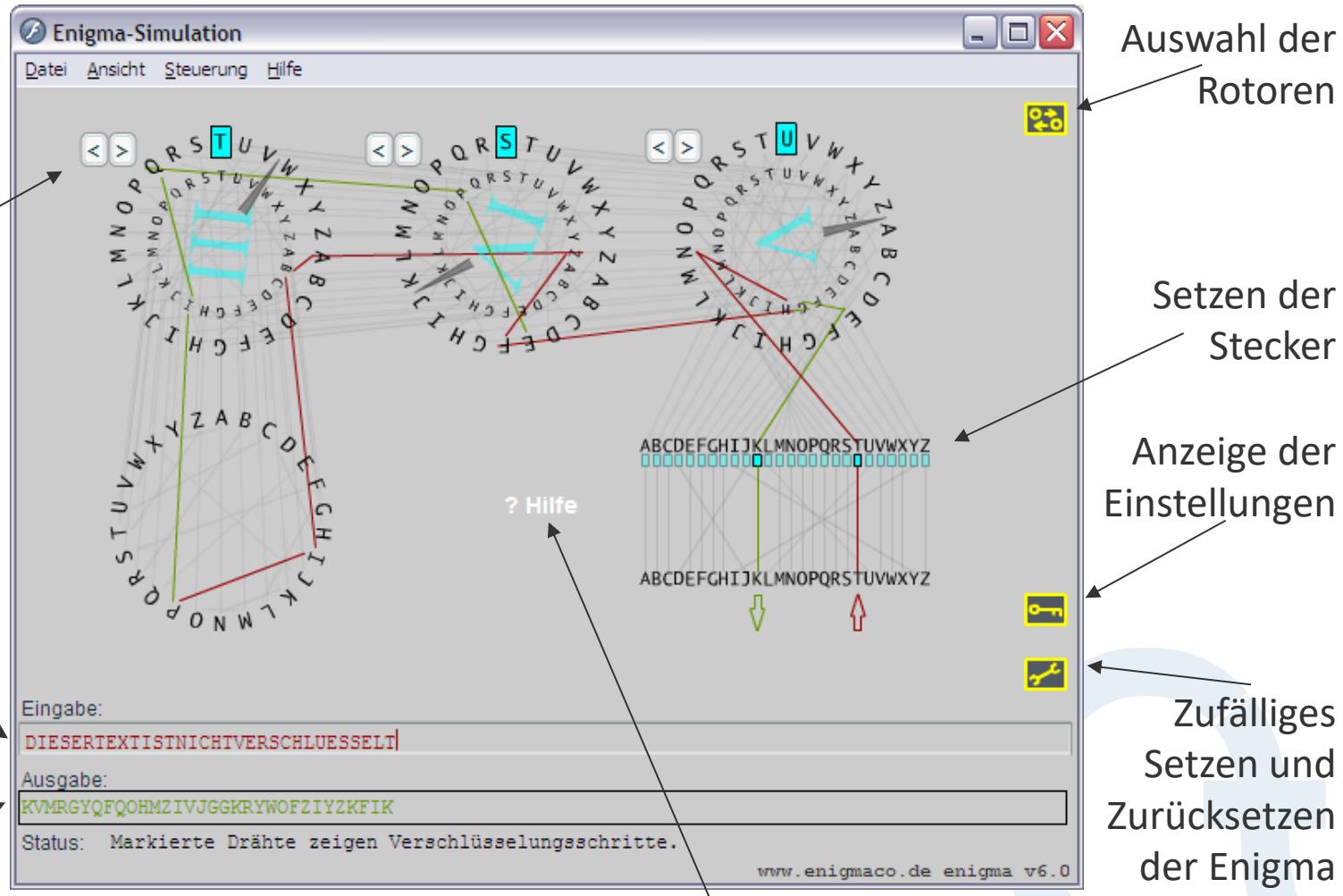


Menü: „Einzelverfahren“ \ „Visualisierung von Algorithmen“ \ „AES“ \ „Rijndael-Flussvisualisierung“

Anwendungsbeispiele (16)

Visualisierung der Enigma-Verschlüsselung – mit Flash

Rotorstellung verändern

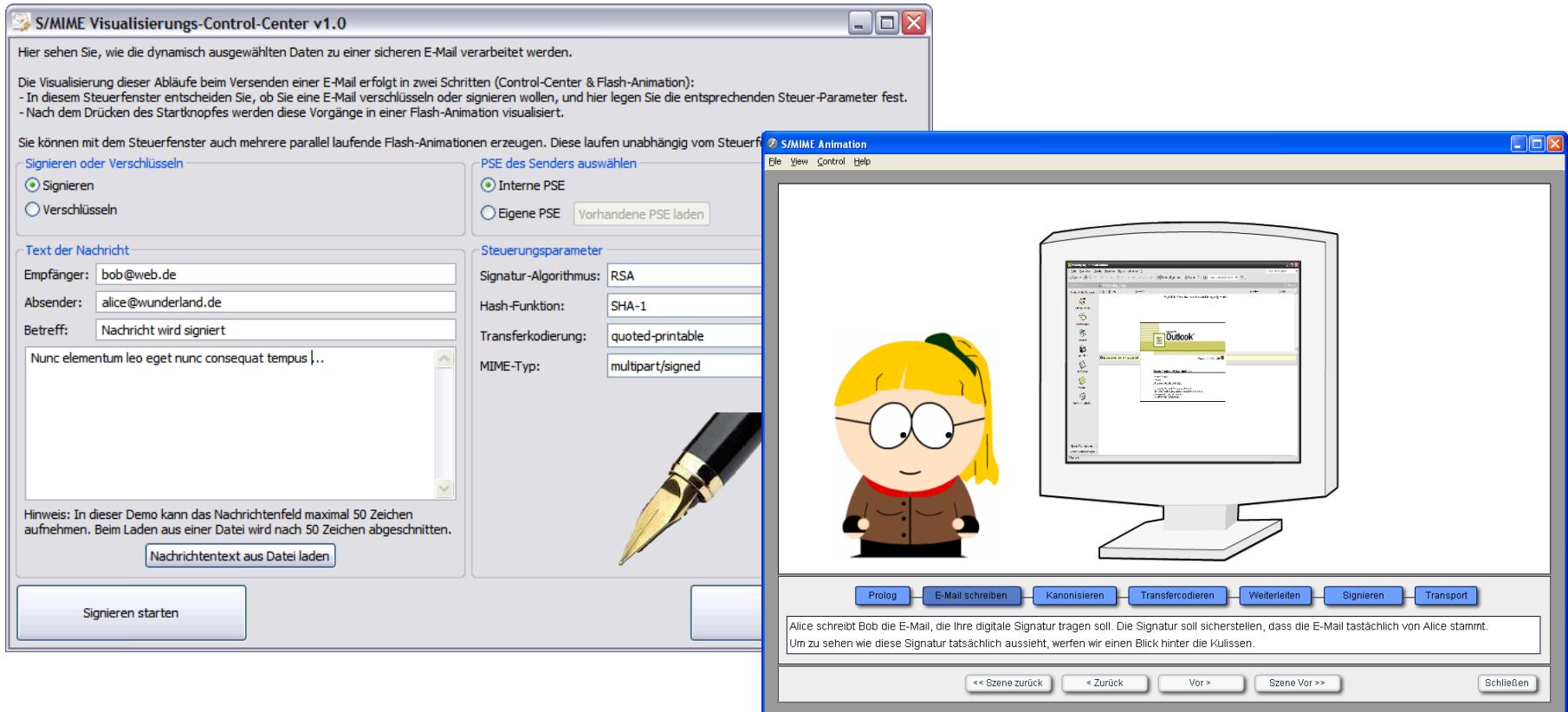


Anwendungsbeispiele (17)

Visualisierung von sicherer E-Mail per S/MIME

S/MIME-Visualisierung

- Control-Center: Signieren/Verschlüsseln von Nachrichten mit verschiedenen Parametern
- Animation: Von der Erstellung beim Sender bis zum Lesen beim Empfänger



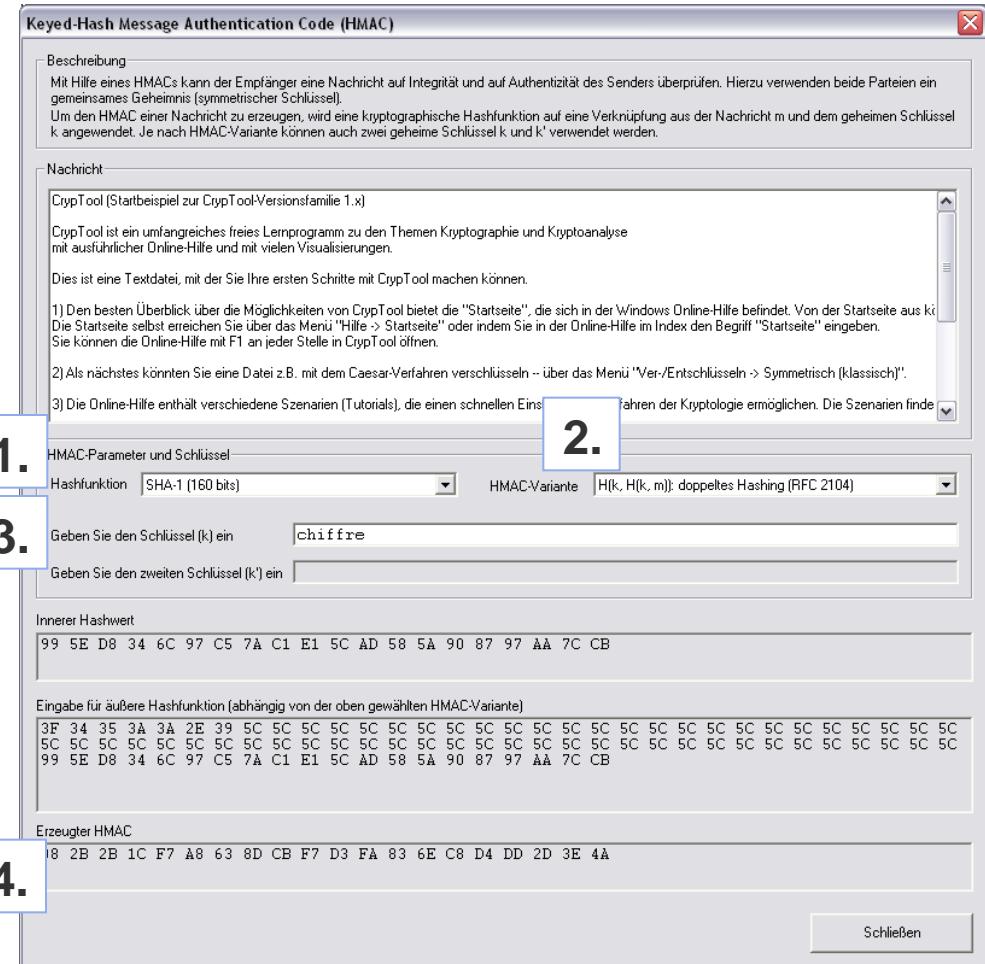
Menü: „Einzelverfahren“ \ „Protokolle“ \ „Sichere E-Mail mit S/MIME...“

Anwendungsbeispiele (18)

Erzeugung eines Keyed-Hash Message Authentication Code (HMAC)

Keyed-Hash Message Authentication Code (HMAC)

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



Menü: „Einzelverfahren“ \ „Hashverfahren“ \ „Generieren von HMACs“

Anwendungsbeispiele (19)

Hash-Demo

Sensitivität von Hashfunktionen bei Änderungen des Originaltextes

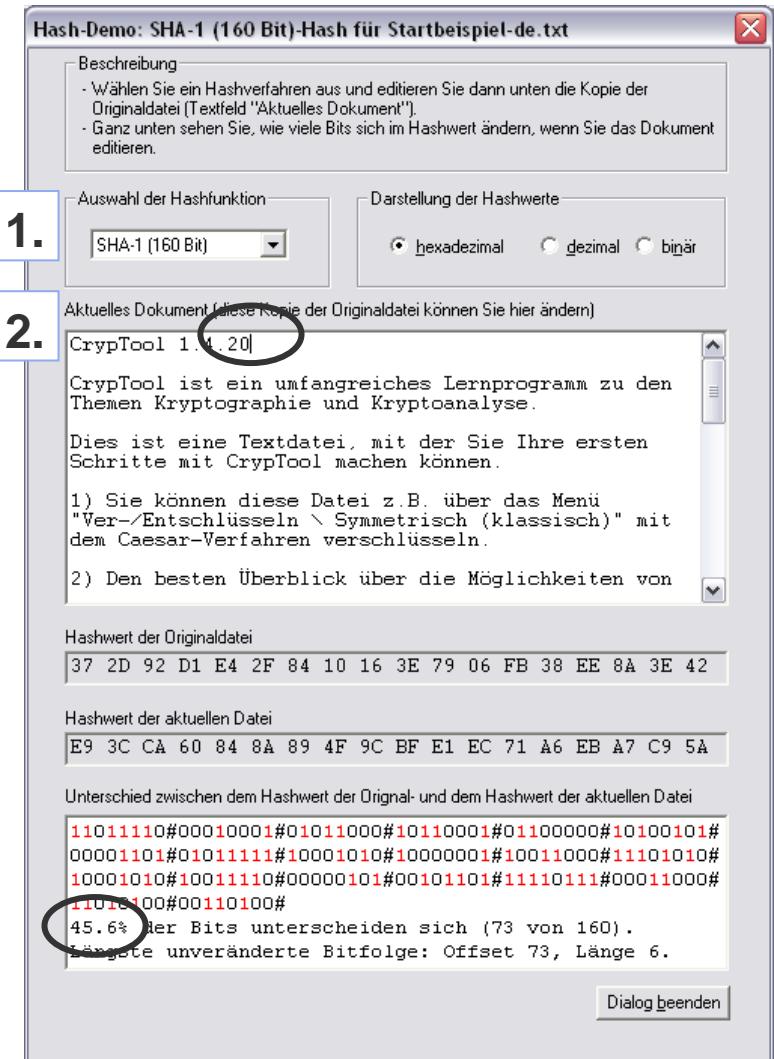
1. Auswahl der Hashfunktion
2. Zusätzliches Einfügen von Zeichen im Text

Beispiel

Die Eingabe eines zusätzlichen Leerzeichens hinter „CrypTool“ in der Originaldatei bewirkt eine 45,6%-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).

Menü: „Einzelverfahren“ \ „Hashverfahren“ \ „Hash-Demo“



Anwendungsbeispiele (20)

Lernprogramm zur Zahlentheorie und zur asymmetrischen Verschlüsselung

■ Zahlentheorie

Tutorial plus
graphische Elemente
und Tools zum
Ausprobieren

■ Themen

1. Ganze Zahlen
2. Restklassen
3. Primzahlerzeugung
4. Asymmetrische
Verschlüsselung
5. Faktorisierung
6. Diskrete
Logarithmen

The screenshot shows a software window titled "ZT" with a menu bar containing "Rechner", "Navigation", "Verzeichnisse", and "Hilfe". The main content area is titled "1.2 Primzahlen" and displays the following text:

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 $11 > \sqrt{89}$.

At the bottom, there are navigation icons: back, forward, search, and help, followed by the text "(Go on to the next page.)".

Menü: „Einzelverfahren“ \ „Zahlentheorie interaktiv“ \ „Lernprogramm zur Zahlentheorie“

Anwendungsbeispiele (21)

Punktaddition auf elliptischen Kurven

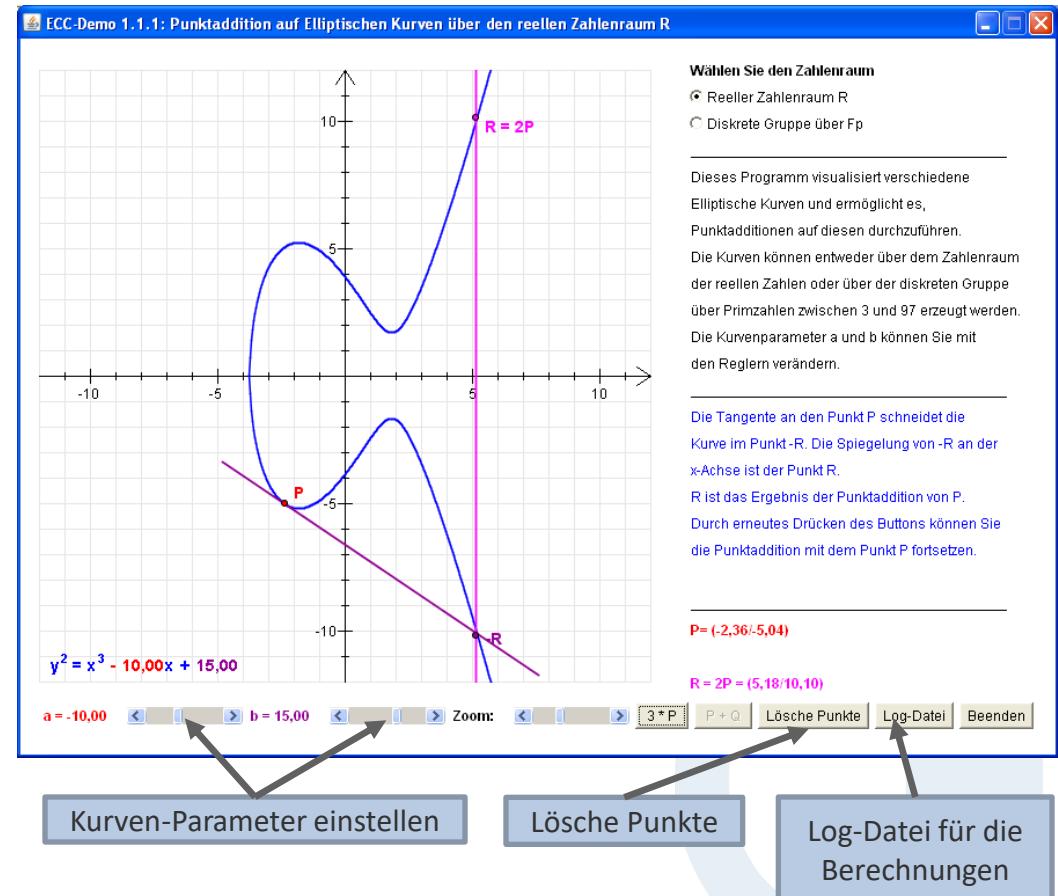
- Visualisierung der Punktaddition auf elliptischen Kurven (sowohl reell wie diskret)
- Grundlage der Elliptischen Kurven Kryptographie (ECC)

Beispiel 1: Versch. Punkte addieren

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

Beispiel 2: Einen Punkt vervielfachen

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



Menü: „Einzelverfahren“ \ „Zahlentheorie interaktiv“ \ „Punktaddition auf Elliptischen Kurven“

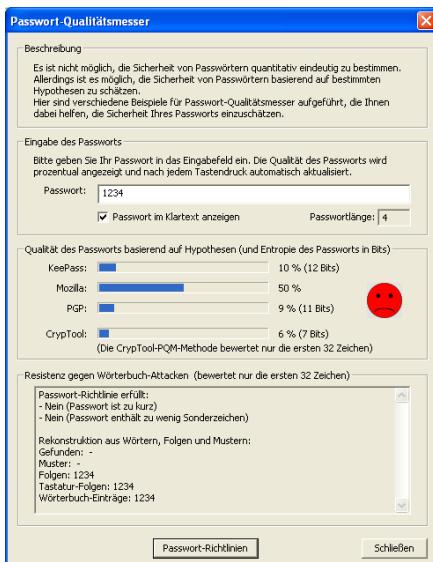
Anwendungsbeispiele (22)

Passwort-Qualitätsmesser (PQM) und Passwort-Entropie (1)

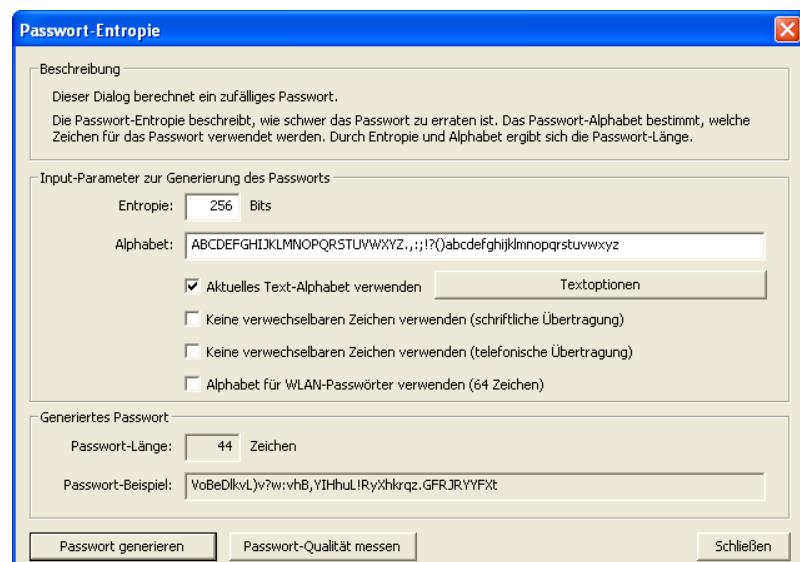
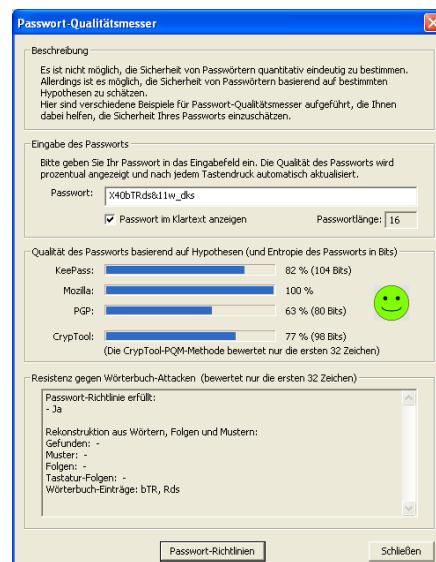
Funktionen

- 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**



Passwort: **X40bTRds&11w_dks**



Menü: „Einzelverfahren“ \ „Tools“ \ „Passwort-Qualitätsmesser“

Menü: „Einzelverfahren“ \ „Tools“ \ „Passwort-Entropie“

Anwendungsbeispiele (22)

Passwort-Qualitätsmesser (PQM) und Passwort-Entropie (2)

Erkenntnisse des Passwort-Qualitätsmessers

- Höhere Qualität des Passwortes durch die Verwendung von **verschiedenen Zeichenarten**: Groß-/Kleinschreibung, Zahlen und Sonderzeichen (**Passwortraum**)
- Passwortqualität hängt primär von der **Länge des Passwortes** ab!
- **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** (Anmerkung: Hier ist bisher in CT1 kein Wörterbuch-Abgleich 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 (23)

Brute-Force-Analyse (1)

Brute-Force-Analyse

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

Beispiel: Analyse mit DES (ECB)

Versuch, mittels 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***** |
| ddddddddd***** |

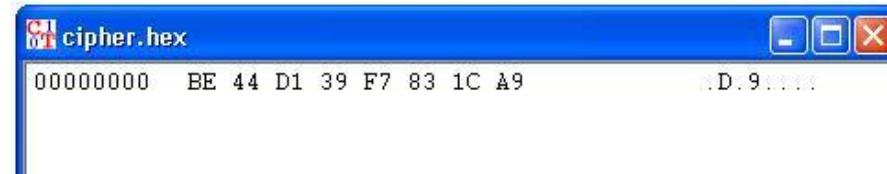
| Verschlüsselter Text (Hex) |
|----------------------------|
| 66b9354452d29eb5 |
| 1f0dd05d8ed51583 |
| bcf9ebd1979ead6a |
| 8cf42d40e004a1d4 |
| 0ed33fed7f46c585 |
| d6d8641bc4fb2478 |
| a2e66d852e175f5c |

Anwendungsbeispiele (23)

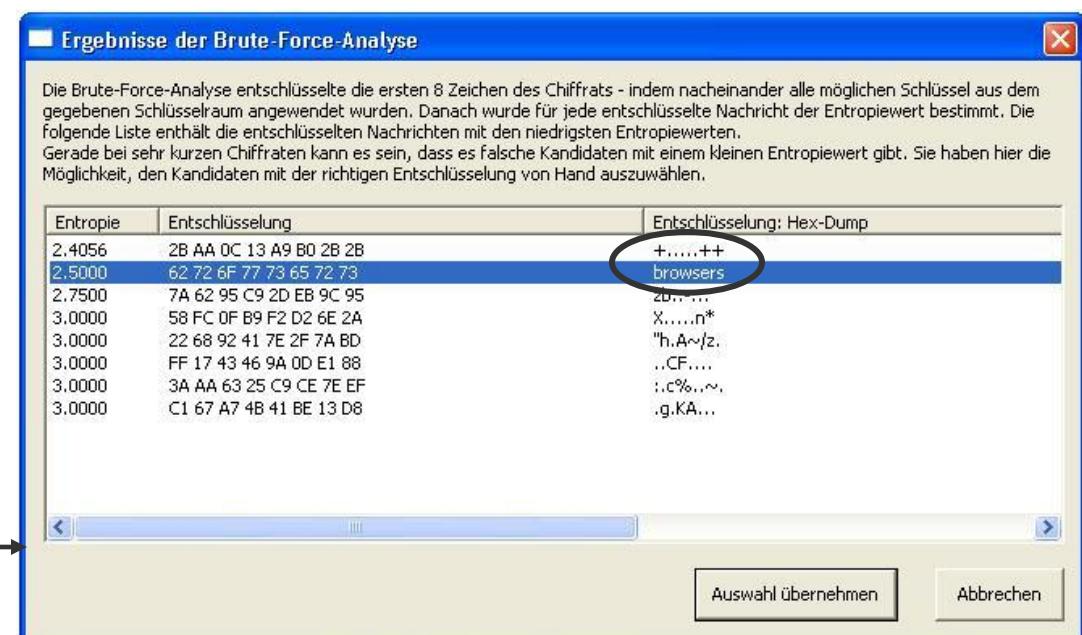
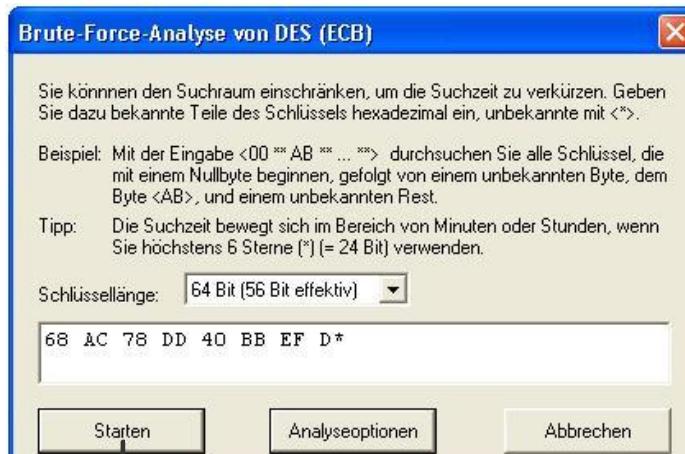
Brute-Force-Analyse (2)

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.

Menü: „Ansicht“ \ „Als HexDump anzeigen“



Menü: „Analyse“ \ „Symmetrische Verfahren (modern)“ \ „DES (ECB)“



Anwendungsbeispiele (24)

Skytale / Gartenzaun

Skytale und Gartenzaun

- Transpositionsverfahren
(Vertauschen der Reihenfolge der Zeichen im Klartext)
- **Auswahl des Schlüssels**
 - Anzahl der Kanten (Skytale)
 - Anzahl der Zeilen (Gartenzaun)
 - Offset



Menü: „Ver-/Entschlüsseln“ \ „Symmetrisch (klassisch)“ \ „Skytale/Gartenzaun ...“

Textoptionen

- Übergeordnete Optionen (Menü: „Optionen“ \ „Textoptionen“)
- Formatierungsoptionen für Klartext und Geheimtext
- Verarbeitung von Groß-/Kleinbuchstaben
- Alphabet für die Verarbeitung des Textes
(z.B. welche Zeichen sollen ver-/entschlüsselt werden)
- Einfache Rückkehr zu den Standard-Einstellungen
- Dynamische Erstellung der statistischen Referenz-Kennzahlen



Anwendungsbeispiele (25)

Hill-Verschlüsselung / Hill-Analyse (1)

Hill-Verschlüsselung

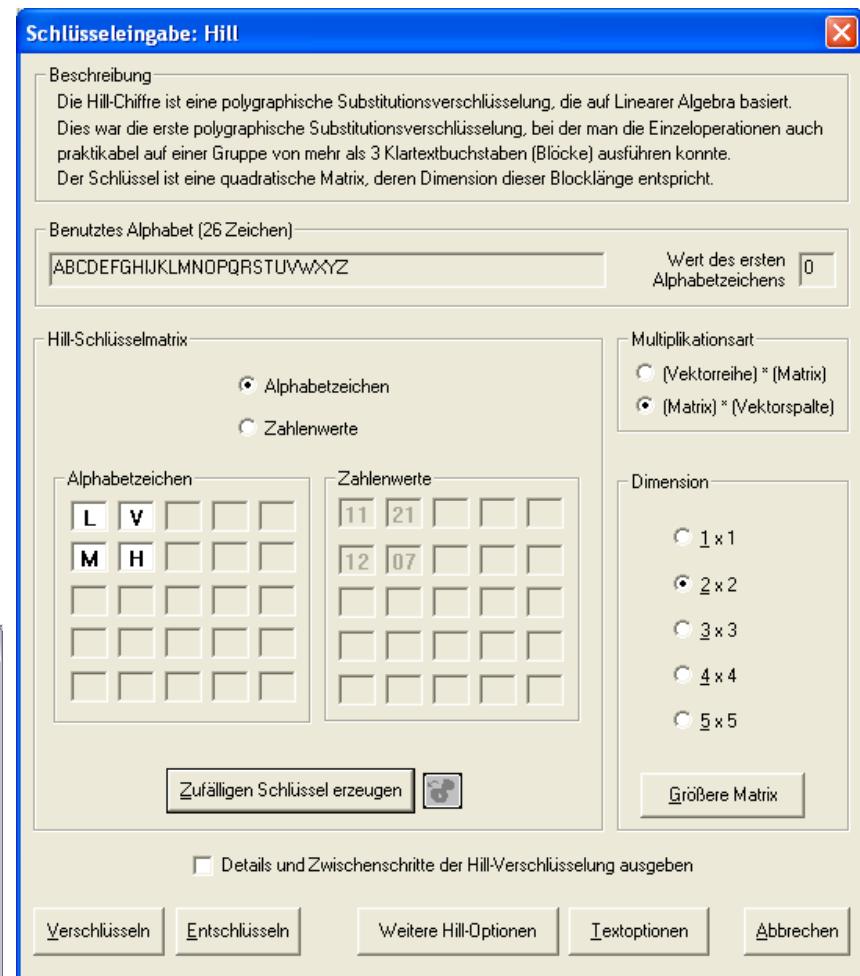
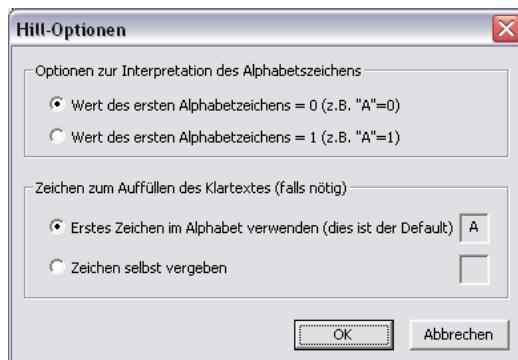
- Polygraphische Substitutionsverschlüsselung
- Basiert auf linearer Algebra

Schlüssel

- Alphabetzeichen (siehe Textoptionen) oder Zahlenwerte
- Eingabe oder zufällige Schlüssel erzeugen
- Wahl der Multiplikationsart
- Dimension der Matrix
- Hill-Optionen

Menü:

„Ver-/Entschlüsseln“ \
 „Symmetrisch (klassisch)“ \
 „Hill ...“



Anwendungsbeispiele (25)

Hill-Verschlüsselung / Hill-Analyse (2)

Hill-Verschlüsselung

- Beispiel-Text mit dem Schlüssel LVMH

Hill-Analyse (Known Plaintext)

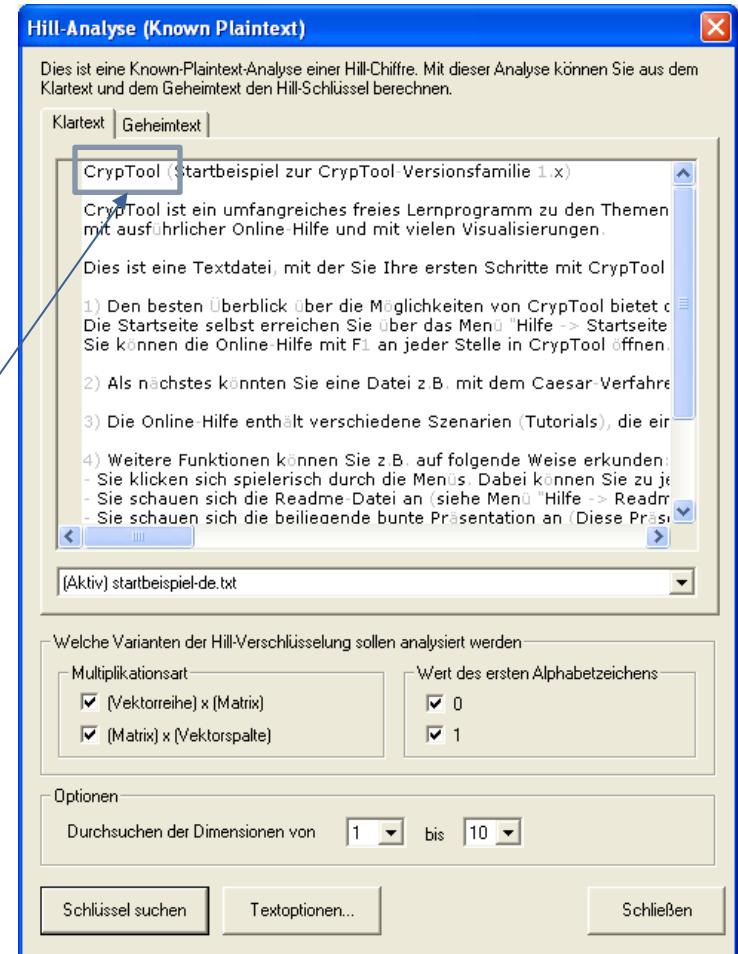
1. Langer Klar-/Geheimtext

- Auswahl des Klartext (startbeispiel-de.txt)
- Auswahl des Geheimtexts (Hill-Verschlüsselung von <startbeispiel-de.txt>)
- „Schlüssel suchen“

2. Verkürzter Klar-/Geheimtext

- Alles löschen im Klartext bis auf den Anfang („CrypTool“)
- Reduzieren des Geheimtexts auf „PnhdJovl“
- „Schlüssel suchen“ findet korrekten Schlüssel!

Welche Textlänge im Klar-/Geheimtext-Paar ist notwendig, um den richtigen Schlüssel zu finden?



Menü: „Analyse“ \ „Symmetrische Verschlüsselung (klassisch)“ \ „Known Plaintext“ \ „Hill...“

Anwendungsbeispiele (26)

CrypTool Online-Hilfe (1)



Menü: „Hilfe“ \ „Startseite“

The screenshot shows the CrypTool 1.4.30 help window with the title 'Hilfe zu CrypTool 1.4.30'. The menu bar is identical to the first window. The main content area is titled 'CBC-Modus' and contains text about Cipher Block Chaining and its steps.

CBC-Modus

CBC steht für **Cipher Block Chaining**.

Bei diesem Modus fließt das Ergebnis der Verschlüsselung früherer Blöcke in die Verschlüsselung des aktuellen Blocks mit ein. Jeder Block des verschlüsselten Textes hängt also nicht nur vom zugehörigen Klartextblock, sondern auch von allen vorherigen Klartextblöcken ab. Für den ersten Klartextblock wird zusätzlich ein Initialisierungsblock benötigt.

Die Verschlüsselung läuft wie folgt:

1. Der erste Klartextblock wird verschlüsselt.
2. Der resultierende Block verschlüsselten Textes wird mit dem nächsten Klartextblock mittels exklusivem Oder verknüpft und erst anschließend verschlüsselt.
3. Punkt 2 wird so lange fortgesetzt, bis das Ende des Klartextes erreicht ist.

Die Entschlüsselung läuft analog ab:

1. Der erste Block des verschlüsselten Textes wird entschlüsselt.
2. Der nächste Block des verschlüsselten Textes wird zunächst entschlüsselt anschließend mit dem vorhergehenden Block des verschlüsselten Textes mittels exklusivem Oder verknüpft.
3. Punkt 2 wird so lange fortgesetzt, bis das Ende des verschlüsselten Textes erreicht ist.

Dieses Verfahren hat allerdings immer noch den Nachteil, dass zwei Nachrichten bis zum ersten Unterschied den gleichen verschlüsselten Text ergeben. Insbesondere liefern also identische Nachrichten den gleichen verschlüsselten Text. Um das zu verhindern, verwendet man sogenannte Initialisierungsvektoren. Der Initialisierungsvektor ist ein Zufallswert, der unverschlüsselt mit der Nachricht übertragen werden kann. Vor der Verschlüsselung (bzw. Entschlüsselung) des ersten Blockes wird dieser Block mit dem Initialisierungsvektor mittels exklusivem Oder verknüpft. Anschließend läuft das Verfahren so ab wie oben beschrieben.

In ECB-Modus wird jeder Block unabhängig von den anderen Blöcken verschlüsselt.

Anwendungsbeispiele (26)

CrypTool Online-Hilfe (2)

Hilfe zu CrypTool

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 Hybridschlüsselung Hybrideverschlüsselung Hypothese H0 IDEA-Verschlüsselungsverfahren Importieren (Zertifikat) / PKCS #12 Initialisierungsdatei Interoperabilität Inverse Permutation involutorisch 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 stereotype Nachrichten
- Angriff auf kleine geheime Schlüssel

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

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

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:

The diagram shows a 2D grid of points representing a lattice. Two specific points are highlighted: $q_1(x) = 3x + 1$ and $q_2(x) = 4x - 1$. Both points pass through the origin $(0,0)$, which is also a point on the line $h(x) = x - 2$.

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.

Bemerkung:

Anwendungsbeispiele (26)

CrypTool Online-Hilfe (3)

Hilfe zu CrypTool 1.4.20 (DE)

Ausblenden Zurück Vorwärts Abbrechen Aktualisieren Startseite Drucken Optionen

Inhalt Index Suchen Zu suchendes Schlüsselwort:

base

Base64-Codierung

- BC
- Bearbeiten
- Beenden
- Beispiele/Szenarien
- Binäres exklusives Oder
- Bitlänge
- Blöcke
- Brute-Force-Angriff
- Bucher
- Byteweise Addition
- Caesar-Verschlüsselungsverfahren
- CBC-Modus
- Challenge
- Challenge-Response-Demo
- CHM2-Verteilung
- Chiffrat
- Chiffertext
- Chinesischer Restsatz (CRT)
- Chosen-Plaintext-Angriff
- Ciphertext
- Ciphertext-Only-Angriff
- Codierungen
- Copyright
- CryptTool
- cv cryptovision
- Datei
- Default-Settings
- Demonstrationen/Szenarien
- DES-L-Verschlüsselungsverfahren
- DES-Verschlüsselungsverfahren
- Dialog (Übersicht aller Dialoge)
- Dialog der Schwestern
- Difffie-Hellman-Schlüsselaustausch
- Digitale Signaturen/PKI
- Digitales Zertifikat
- Digramm
- DIN
- Dokument
- Dokument-Eigenschaften
- Drucken
- DSA
- ECB-Modus
- ECCP-DSA-Signatur
- ECCP-NR-Signatur
- Einordnung/Überblick/Übersicht

Anzeigen

Vergleich von Base64- und UU-Codierung

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

Base64 UUEncode

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

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

Aufteilen von 3 x 8 Bit in 4 x 6 Bit

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.

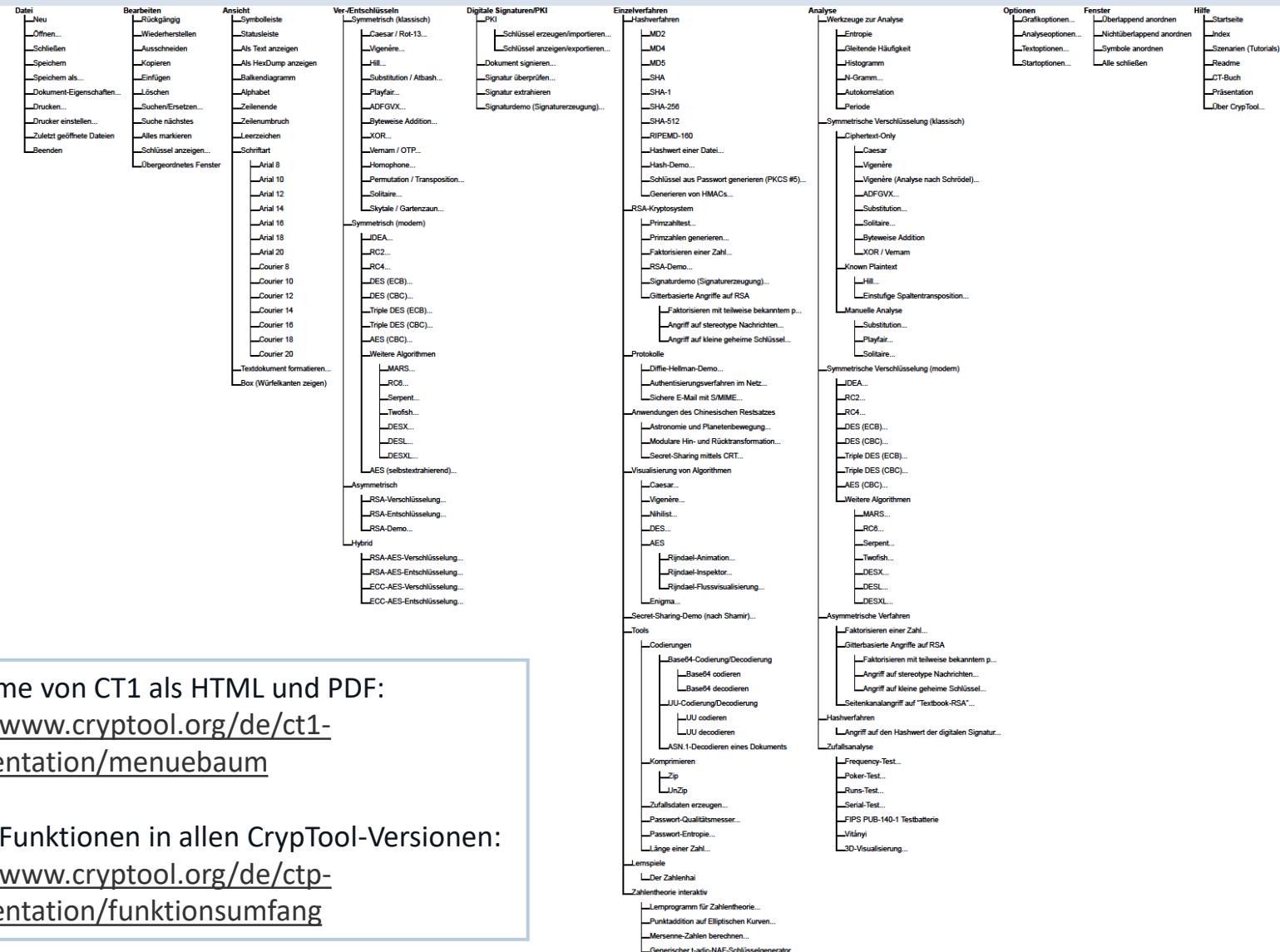
Base64UUencodevgl3.gif

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

| Vorteile | Nachteile |
|---|--|
| <ul style="list-style-type: none">Beliebige Binärdaten können mit einem 6-bit Zeichensatz dargestellt werden:<ul style="list-style-type: none">Keine Probleme mit 7-bit Zeichensatzbeschränkungen.Keine Probleme mit Zeilenlängenbeschränkungen oder besonderen Steuerzeichen.Vergrößerung nur um ca. 33 % (anstelle von z.B. Kodierung in Hexadezimalwerte = Vergrößerung um 100 %). | <ul style="list-style-type: none">Keine Unterstützung für die Aufteilung von großen Dateien.Vergrößerung der Dateien um ca. 33 % (im Vergleich zur Originaldatei). Nur UUencode.Keine EBCDIC Unterstützung.Keine definierten Standards. |

Anwendungsbeispiele (26)

Menübaum zum Programm CryptTool 1.4.40

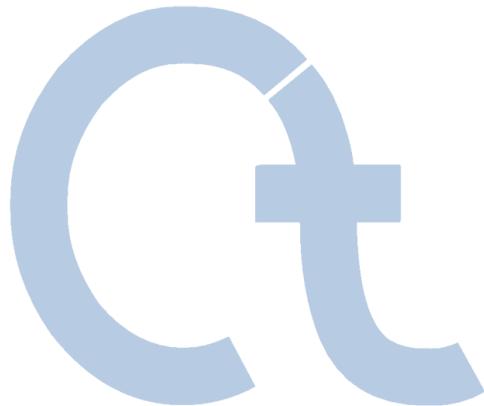


Menübäume von CT1 als HTML und PDF:

<https://www.cryptool.org/de/ct1-dokumentation/menuebaum>

Liste aller Funktionen in allen CryptTool-Versionen:

<https://www.cryptool.org/de/ctp-dokumentation/funktionsumfang>



- I. CrypTool und Kryptologie – Überblick
- II. Was bietet CrypTool 1?
- III. Ausgewählte Beispiele
- IV. Projekt / Ausblick / Kontakt**

Anhang

Weiterentwicklung (1)

Beispiele, was nach CrypTool Release 1.4.40 kommt (vgl. Readme-Datei)

| | |
|---------|---|
| CT1 | FIPS-Test , der Pakete von Längen ungleich 2500 Bytes untersucht; ... |
| JCT | Tri-partite Schlüsselvereinbarung |
| JCT | Visualisierung der Interoperabilität von S/MIME- und OpenPGP-Formaten |
| JCT | Entropie-Untersuchungen |
| JCT | Fleißner-Schablone, Autokey-Vigenère, interaktive Kryptoanalyse klassischer Verfahren |
| JCT | Analyse von Transpositionschiffen mit dem ACO-Algorithmus |
| JCT | Visualisierung von Zero-Knowledge-Beweisen |
| JCT+CT2 | Visualisierung eines Quantum-Schlüsselaustausch-Protokolls (BB84) |
| JCT | Visualisierung des SETUP-Angriffes auf die RSA-Schlüsselerzeugung (Kleptographie) |
| JCT | Action-History mit der Möglichkeit, Produktchiffren (Kaskaden) zu bauen und abzuspielen |
| CT2 | Umfangreiche Visualisierungen zum Thema Primzahlen |
| CT2 | GNFS (General number field sieve) |
| CT2 | Verschlüsselung und automatisierte Kryptoanalyse der Enigma, und evtl. der M-138 und der Sigaba |
| CT2 | Cube-Attack (I. Dinur und A. Shamir: „Cube Attacks on Tweakable Black Box Polynomials“, 2008) |
| CT2 | Demo von Bleichenbachers und Kühns Angriff auf RSA-Signaturen |
| CT2 | evtl.: Demo virtueller Kreditkartennummern als Ansatz gegen Kreditkartenmissbrauch |
| CT2 | WEP-Verschlüsselung und WEP-Analyse |
| CT2 | Massenmustersuche |
| CT2 | Framework zur verteilten Kryptoanalyse → CrypCloud |
| CT2 | evtl.: Demo SOA-Security (SOAP-Nachrichten nach WS-Security zwischen den Beteiligten) |
| CT2 | Framework zum Bau und zur Analyse von LFSR-Stromchiffren |
| CT2/JCT | Erstellung einer Kommandozeilenversion für Batch-Steuerung |
| CT2/JCT | Moderne Pure-Plugin-Architektur mit Nachladen von Plugins |
| Alle | Weitere Parametrisierung / Flexibilisierung der vorhandenen Verfahren |
| Ideen | Visualisierung des SSL-Protokolls // Demo zur Visuellen Kryptographie // Post-Quantum-Computing // Kryptographie als Webapplikation // Privacy-Preserving-Verfahren |

CT1 = CrypTool 1.x

Neue Versionen von CT:

CT2 = CrypTool 2

JCT = JCrypTool

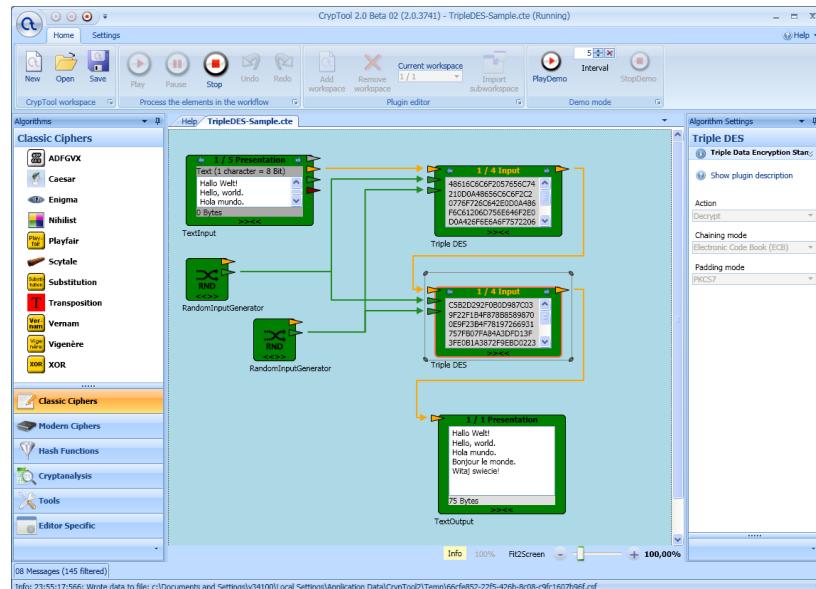
(beide auf den nächsten Folien eingeführt)



Weiterentwicklung (2)

Zwei Nachfolger-Versionen von CT v1 (vgl. Readme-Datei)

1. JCT: Portierung und Neudesign der C++-Version mit Java / SWT / Eclipse / RCP
 - siehe: <https://github.com/jcryptool/core/wiki>
 - Release Candidate 8 ist verfügbar seit Oktober 2016 (seit 2010 gibt es wöchentlich einen Weekly Build).
2. CT2: Portierung und Neudesign der C++-Version mit C# / WPF / Visual Studio / .NET
 - Erlaubt visuelle Programmierung und verteilte Berechnungen (CrypCloud)
 - siehe: <https://www.cryptool.org/de/ct2-dokumentation>
 - Release CT 2.0 ist verfügbar seit August 2014 (seit Juli 2008 werden täglich Nightly Builds erstellt).

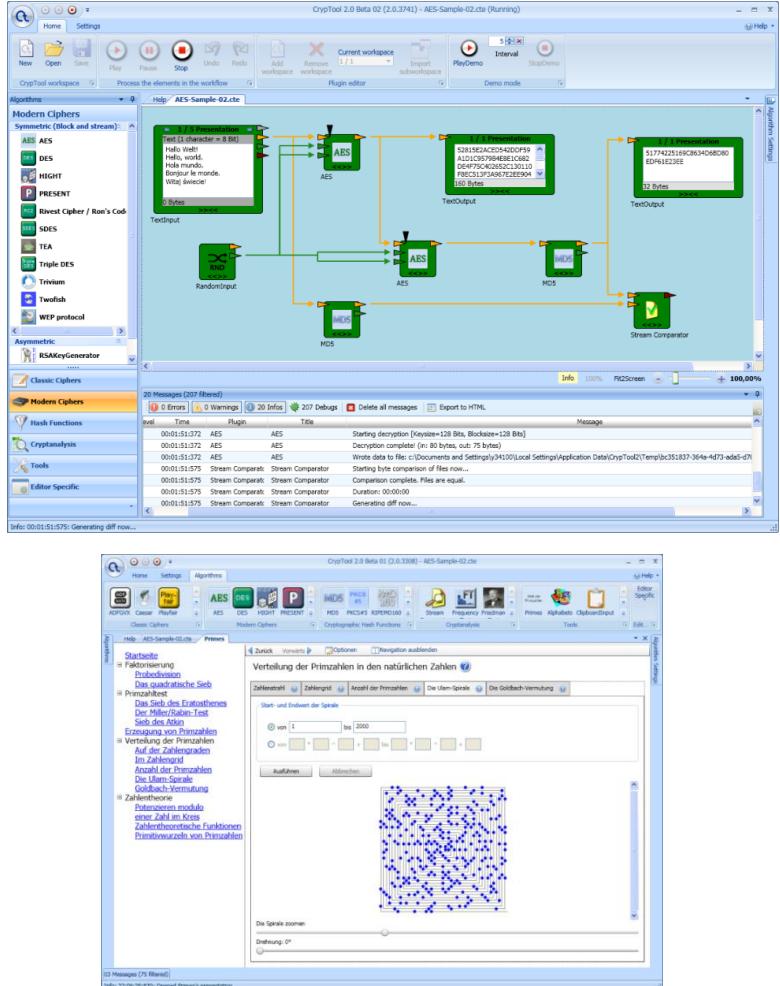


CrypTool 2 (CT2) (Bild Stand 2010)

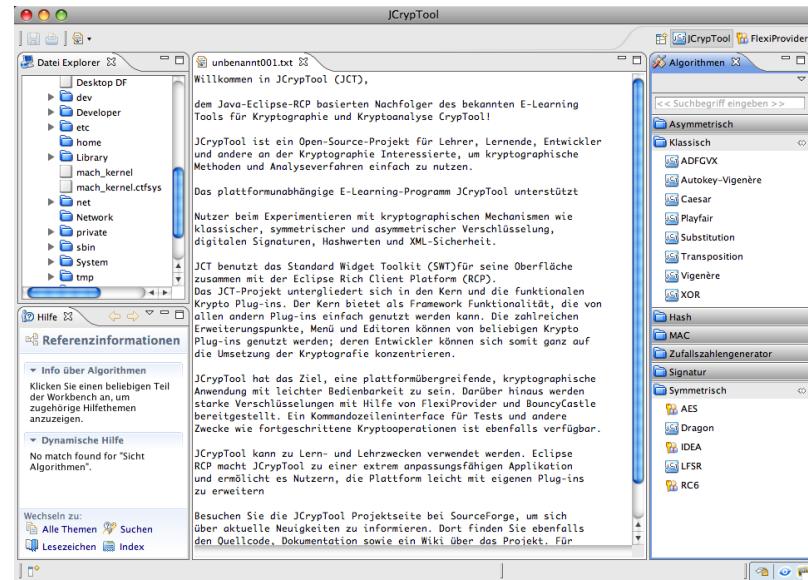


JCrypTool (JCT) (Bild Stand 2010)

Weiterentwicklung (3)



CrypTool 2 (CT2) (Bilder Stand 2010)



JCrypTool (JCT) (Bilder Stand 2010)

CrypTool als Framework für eigene Arbeiten

Angebot

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

Aktuelle Entwicklungsumgebung für CT1: Microsoft Visual Studio C++ , Perl, Subversion Source-Code-Management

- CrypTool 1.4.40: Visual C++ .NET (= VC++ 9.0)(= Visual Studio 2008 Standard)
- Beschreibung für Entwickler: siehe CrypToolDeveloperReadme.pdf im Repository
- Download: Sources und Binaries der Release-Versionen
Jeder Interessierte und Entwickler hat Lesezugriff auf die Sourcen.

Entwicklungsumgebungen von CT2 und JCT

- CT2 – C#-Version: .NET 4.0 mit Visual Studio 2015 Express Edition (kostenlos) und WPF
- JCT – Java-Version: mit Eclipse 4.6, RCP, SWT (kostenlos)



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)
- CT1: für das bisherige C/C++ Projekt, und
- In den neuen Projekten (bevorzugt):
 - C#-Projekt: „CrypTool 2“ = CT2
 - Java-Projekt: „JCrypTool“ = JCT
- Insbesondere Lehrstühle, die CrypTool zur Ausbildung verwenden, sind herzlich eingeladen, zur Weiterentwicklung beizutragen.
- Beispiele offener Aufgaben finden sich auf den entsprechenden Entwickler-Seiten (nur englisch):
 - CT2: Siehe die Liste <https://www.cryptool.org/trac/CrypTool2/wiki/WikiStart>
 - JCT: Siehe das Wiki <https://github.com/jcryptool/core/wiki/Project-Ideas>
- Signifikante Beiträge können namentlich erwähnt werden
(in der Hilfe, Readme, About-Dialog und auf der Webseite).
- Derzeit wird das gesamte CT1-Programmpaket etwa 6.000 mal pro Monat von der CrypTool-Webseite herunter geladen (davon etwas mehr als 50 % die englische Version).
Die zwei Nachfolger-Versionen werden schon über 2.000 mal pro Monat heruntergeladen.

CrypTool – Fazit

DAS E-Learning-Programm für Kryptologie

- Seit über 15 Jahren ein erfolgreiches Open-Source-Projekt
- Mehr als 600.000 Downloads
- Weltweiter Einsatz in Schulen und Universitäten sowie Firmen und Behörden
- Umfangreiche Online-Hilfe und Dokumentation
- Frei verfügbar
- Mehrsprachig

CT: Das weltweit meistverbreitete Lernprogramm für Kryptographie und Kryptoanalyse.



Kontaktadresse

Prof. Bernhard Esslinger

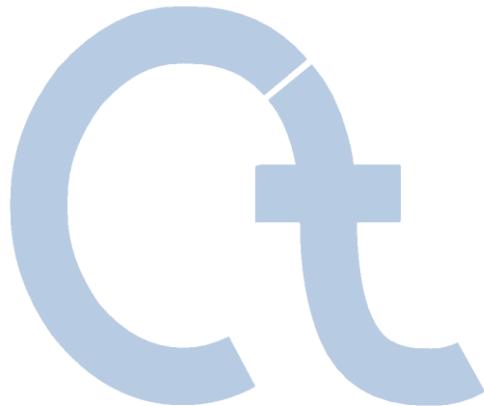
Universität Siegen
Institut für Wirtschaftsinformatik

bernhard.esslinger@uni-siegen.de

www.cryptool.org

Weitere Kontaktadressen: siehe Readme im CrypTool-1-Programmpaket





- I. CrypTool und Kryptologie – Überblick
- II. Was bietet CrypTool 1?
- III. Ausgewählte Beispiele
- IV. Projekt / Ausblick / Kontakt

Anhang

(Literatur, CrypTool-bezogene Webseiten, Download)

Weitere Lektüre

Als Einstieg in die Kryptologie und mehr

- Klaus Schmeh: „*Codeknacker gegen Codemacher. Die faszinierende Geschichte der Verschlüsselung*“, 2. Auflage, 2007, W3L
- Simon Singh: „*Geheime Botschaften*“, 2000, Hanser
- Johannes Buchmann: „*Einführung in die Kryptographie*“, 4. Auflage, 2008, Springer
- Paar / Pelzl: „*Understanding Cryptography – A Textbook for Students and Practitioners*“, 1. Auflage, 2009, Springer
- Klaus Schmeh: „*Kryptografie – Verfahren, Protokolle, Infrastrukturen*“, 6. Auflage, 2016, Dpunkt
- Swoboda / Pramateftakis / Spitz: „*Kryptographie und IT-Sicherheit -- Grundlagen und Anwendungen*“, 2008, Vieweg + Teubner
- Beutelspacher / Schwenk / 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, ACM [Englisch]
- Antoine Joux, „*Algorithmic Cryptanalysis*“, 2009, Chapman & Hall/CRC Cryptography and Network Security Series
- Vielfältige Krypto-Literatur – siehe Links auf der CrypTool-Webseite sowie Quellenangaben in der Online-Hilfe von CrypTool (z.B. von Wätjen, Salomaa, Brands, Schneier, Shoup, Stamp/Low, Oppliger, Martin, ...)
- Bedeutung der Kryptographie in dem breiteren Rahmen von IT-Sicherheit und Risiko-Management
 - Siehe z.B. Kenneth C. Laudon / Jane P. Laudon / Detlef Schoder: „*Wirtschaftsinformatik*“, 3. Auflage 2016, Pearson, Kapitel 15 „IT-Sicherheit“
 - Wikipedia: <http://de.wikipedia.org/wiki/Risikomanagement>
 - CrypTool-Seite: <https://www.cryptool.org/de/ctp-lehre/awareness>

Das CrypTool-Portal: www.cryptool.org

Was ist CrypTool 1?

CrypTool 1 (CT1) ist ein Open-Source-Programm für Kryptographie und Kryptoanalyse. Es ist die verbreiteste E-Learning-Software ihrer Art. CT1 läuft unter Win 7, Vista und XP.

Über CrypTool 1 Dokumentation Downloads Screenshots

DRITTE ÖFFENTLICHE BETA VON CRYPTOTool 1.4.31

Die dritte öffentliche Beta von CrypTool 1.4.31 ist am 4.10.2013 erschienen. Diese Version steht nun in den 6 Sprachen Deutsch, Englisch, Spanisch, Polnisch, Serbisch und Griechisch zur Verfügung. Wir würden uns freuen, wenn Sie diese Beta ausführlich testen

Über CrypTool 1

Das Programm **CrypTool 1 (CT1)** ist ein kostenloses, Open-Source Windows-Programm für Kryptographie und Kryptoanalyse. Diese E-Learning-Software gibt es in 5 Sprachen und sie ist die weltweit verbreiteste ihrer Art. CrypTool 1 unterstützt eine moderne Lehre an Schulen und Hochschulen sowie die Sensibilisierung von Firmen- und Behördenangehörigen. Das Programm kann [hier](#) heruntergeladen werden. Ursprünglich entwickelt für IT-Sicherheitstrainings in einer Firma hat sich CrypTool 1 mittlerweile in ein wichtiges Open-Source-Projekt im Kryptographie- und Awareness-

Über

- Was ist CrypTool?
- CrypTool in der Lehre
- CrypTool für Awareness
- Präsenz in Printmedien
- Auszeichnungen
- Mitwirkende
- Partnerprojekte
- Kontakt

Funktionen

- CrypTool-Funktionen
- Roadmap

Medien

- Screenshots
- Screencast

Dokumentation

- Präsentationen
- Skript
- Krypto-Historie
- Links / Bücher

www.cryptool-online.org

Vom Smartphone aus in die Kryptologie einsteigen

The screenshot shows the homepage of the CryptTool-Online website. At the top, there's a navigation bar with tabs for "Awareness - CrypTool Portal", "Padding oracle attack - Wiki", and "Über CrypTool-Online - CTO". Below the navigation is a search bar and language selection. The main header "CRYPTOOL-ONLINE" is displayed with the tagline "Cryptography for everybody". A sidebar on the left lists several tools: "CHIFFREN", "KODIERUNGEN", and "KRYPTOANALYSE", each accompanied by a small thumbnail image. The main content area features a banner with handwritten text and a barcode, with the text "SCHON GEWUSST... dass Sie mit CrypTool-Online Barcodes generieren können?". To the right, a large section titled "Über CrypTool-Online (CTO)" provides an overview of the service, mentioning its focus on encryption, coding, and cryptanalysis, and how it allows users to experiment directly in their browser.

Weitere Produkte aus dem CT-Projekt

Mitglieder in der Familie der CrypTool-Webseiten:

- **CrypTool 1** - Seite (CT1)
- **CT2** (Download- und Entwicklerseite)
- **JCT** (Download- und Entwicklerseite)
- **CrypTool-Online**
(Einstieg in Kryptologie und erstes Ausprobieren direkt im Browser, auf dem PC oder mit dem Smartphone)
- **CryptoPortal** für Lehrer (Teilen von Unterrichtsmaterialien)
- **Schülerkrypto** (Tagesveranstaltungen für Schüler & Lehrer)
- **MysteryTwister C3** (MTC3) ist ein internationaler Krypto-Wettbewerb.



[Impressum](#) | [Kontaktieren Sie uns](#) | [Datenschutz](#)

The screenshot shows the CRYPTOPORTAL website interface. At the top, there is a banner featuring a boy writing on a chalkboard with various mathematical and cryptographic symbols. The main navigation bar includes links for "Über", "Unterrichtsmaterial", "Linksammlung", "Registrierung", "CrypTool", and "Einloggen".

Filterkriterien

Land: alle Länder

Schultyp: alle Schultypen

Autor: alle Autoren

Material enthält folgenden Text:

Filtern Zurücksetzen

Unterrichtsmaterial

Passend zu den Filterkriterien: 13 Unterrichtsmaterialien
Letzte Aktivität: 18.01.2010

[1] 3 praktische Unterrichtsbeispiele zur Verschlüsselung mit Excel und CrypTool 0 Kommentare

Autor: Anonym
Land: alle deutschsprachigen Länder
Schultyp: Gymnasien

Materialien zum Workshop "Kryptographie im Unterricht" von Rainer Gürth. Die 3 Beispiele (ROT13, Vigenère und RSA) werden in einem PDF-Arbeitsblatt erläutert und als Excel-Programm angeboten, um sie [...]

[2] Asymmetrische Kryptologie: Schritt-für-Schritt-Anleitung mit CrypTool (01/2010) 0 Kommentare

Autor: B.E
Land: alle deutschsprachigen Länder
Schultyp: alle Schultypen

Diese Anleitung mit über 60 Folien und vielen Screenshots aus CrypTool-1 beschreibt, wie man asymmetrische Krypto-Verfahren zum Verschlüsseln und Signieren verwendet. Die Anleitung kann als komplette [...]

Asymmetrische Kryptologie am Beispiel RSA entdecken v1.1.pdf 395 mal heruntergeladen

MysteryTwister C3

THE CRYPTO CHALLENGE CONTEST

ANZAHL AKTIVER MITGLIEDER: 7886

Jetzt registrieren

MTC3-PARTNER

Folgen Sie uns:

Login DE EN

Start Challenges Forum MysteryTwister I

Über MTC3 Partner Neuigkeiten

TAUSCHEN SIE SICH AUS

Das Forum bietet eine Diskussionsplattform, um sich mit anderen MTC3-Usern auszutauschen. Eventuell hat ja jemand anderes eine Idee, die Sie weiterbringt.

Jetzt registrieren

Who is online

In total there are 21 user online :: 21 registered, 0 hidden
Most users ever online was 25 on Wed May 26, 2010 3:30 PM

Registered users:

Willkommen beim MTC3 — dem Verschlüsselungs-Wettbewerb

Mögen Sie Rätsel? Lösen Sie gerne die Kreuzworträtsel oder Sudokus? Oder sind Sie einfach nur neugierig, wie man heutzutage Geheimnisse schützt (und sie vielleicht auch wieder enthüllt)? Hier sind Sie richtig! Auf dem Portal "MysteryTwister C3" können Sie spannende Aufgaben rund um die Kryptographie lösen. Angefangen von der einfachen Caesar-Chiffre bis hin zum modernen AES-Verfahren finden sich Rätsel für Jedermann. Unsere Rätsel sind in vier Kategorien eingeteilt: Level I bis III, und ein zusätzliches Level X mit "mysteriösen" Aufgaben (diese können seit langem ungelöst sein, meist kennen wir deren Lösungen nicht und wissen nicht mal, ob es eine gibt). Als Anfänger könnten Sie mit den Rätseln beginnen, die bisher am häufigsten gelöst wurden (siehe nachstehende Tabelle). Weitere Informationen zu MTC3 finden Sie auf unserer About-Seite.

9:59 - 01.03.2017] Zylius löste die Level II Challenge 'Brechen SHA1-gehashter Passwörte' +++ [15:27 - 01.03.2017] capiaghi löste die Level I Challenge 'Zahlenfolge' +++

MysteryTwister C3 (MTC3) ist ein internationaler Krypto-Wettbewerb.

Das CrypTool-Buch (PDF ist kostenlos)

The screenshot shows a PDF viewer interface with the following details:

- Top Bar:** Includes icons for cloud, print, email, search, and navigation (up, down, first, last, search). A page number box shows "1 (1 von 568)".
- Toolbar:** Includes icons for zoom (44.3%), orientation (Portrait/Landscape), and other document functions.
- Left Sidebar:** Titled "Lesezeichen" (Bookmarks). It lists the following chapters:
 - Überblick über den Inhalt des CrypTool-Buchs
 - Kurzinhaltverzeichnis
 - Inhaltsverzeichnis
 - Vorwort zur 12. Auflage des CrypTool-Buchs
 - Einführung – Zusammenspiel von Buch und Programmen
 - 1 Sicherheits-Definitionen und Verschlüsselungsverfahren
 - 2 Papier- und Bleistift-Verschlüsselungsverfahren
 - 3 Primzahlen
 - 4 Einführung in die elementare Zahlentheorie mit Beispielen
 - 5 Die mathematischen Ideen hinter der modernen Kryptographie
 - 6 Hashfunktionen und Digitale Signaturen
 - 7 Elliptische Kurven
- Right Content Area:** Shows the cover of the book "Das CrypTool-Buch: Kryptographie lernen und anwenden mit CrypTool und SageMath". The cover features a whiteboard with mathematical notes about Mersenne primes ($2^p - 1$) and their factorizations. The title is overlaid on a blue banner at the bottom. Below the title, it says "Prof. Bernhard Esslinger und das Entwickler-Team des Open-Source-Projektes CrypTool". At the bottom right, it says "12. Auflage (2017)".