

KRYPTOLOGIE MIT CRYPT**OOL** v1.4.21

Einführung in Kryptographie und Kryptoanalyse

Umfang, Technik und Zukunft von CrypTool

Prof. Bernhard Esslinger und CrypTool-Team, 2008

www.cryptool.com

www.cryptool.de

www.cryptool.org

www.cryptool.pl

www.iec.csic.es/cryptool



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. 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
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) (...)



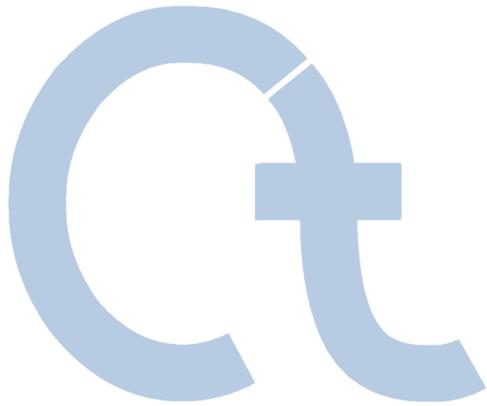
Übersicht (II)

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. Visualisierung von AES
16. Visualisierung der Enigma-Verschlüsselung
17. Erzeugung eines Message Authentication Code (MAC)
18. Hash-Demo
19. Lernprogramm zur Zahlentheorie und zur asymmetrischer Verschlüsselung
20. Punktaddition auf elliptischen Kurven
21. Passwort-Qualitätsmesser und Passwort-Entropie
22. Brute-Force-Analyse
23. CrypTool Online-Hilfe

IV. Projekt / Ausblick / Kontakt





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

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 Chiffretext 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 Fernsehsignale 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>

Bedeutung der Kryptographie

Einsatzbeispiele für Kryptographie

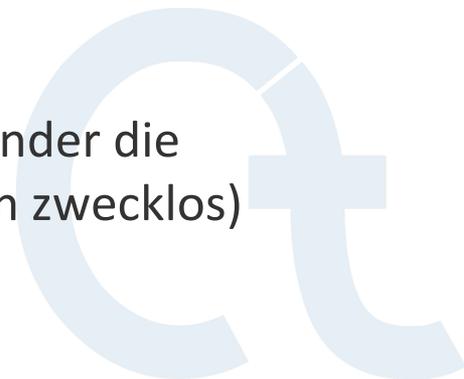
- Telefonkarten, Handys, Fernbedienungen
 - Geldautomaten, Geldverkehr zwischen Banken
 - Electronic cash, Online-Banking, Sichere E-Mail
 - Satellitenfernsehen, PayTV
 - Wegfahrsperre im Auto
 - Digital Rights Management (DRM)
-
- Kryptographie ist schon lange nicht mehr nur auf Agenten, Diplomaten und Militärs begrenzt. Kryptographie ist eine moderne, mathematisch geprägte Wissenschaft.
 - Der Durchbruch für den breiten Einsatz kam mit dem Internet.
 - Für Firmen und Staaten ist es wichtig, dass sowohl die Anwendungen sicher sind, als auch, dass ...

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



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**
 - 1998 – **Projektstart** – Aufwand bisher mehr als 30 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 (Fr. Prof. Eckert)
 - 2007 – CrypTool in deutsch, englisch, polnisch und spanisch
 - 2008 – .NET-Version und Java-Version – Hosting durch die Uni Duisburg (Hr. Prof. Weis)

- **Auszeichnungen**

- 2004 TeleTrust (TTT Förderpreis) 
- 2004 NRW (IT-Sicherheitspreis NRW) 
- 2004 RSA Europe (Finalist beim European Information Security Award) 
- 2008 "Ausgewählter Ort" bei der Standortinitiative "Deutschland – Land der Ideen" 

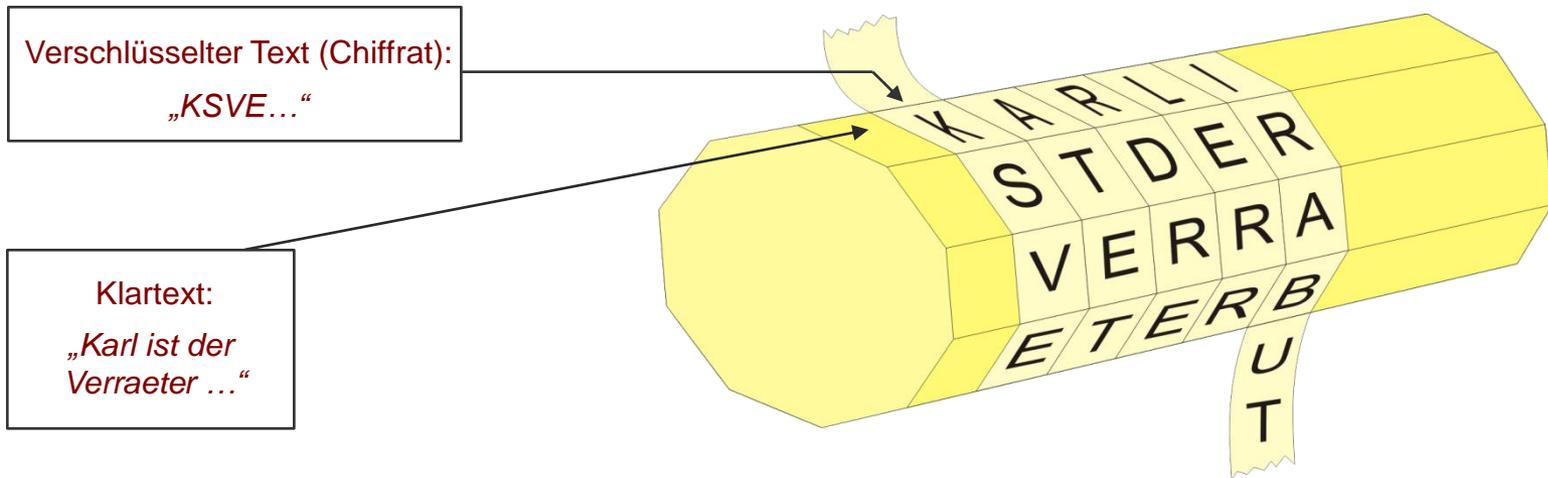
- **Entwickler**

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

Beispiele aus der klassischen Kryptographie (1)

Älteste bekannte Verschlüsselungsverfahren

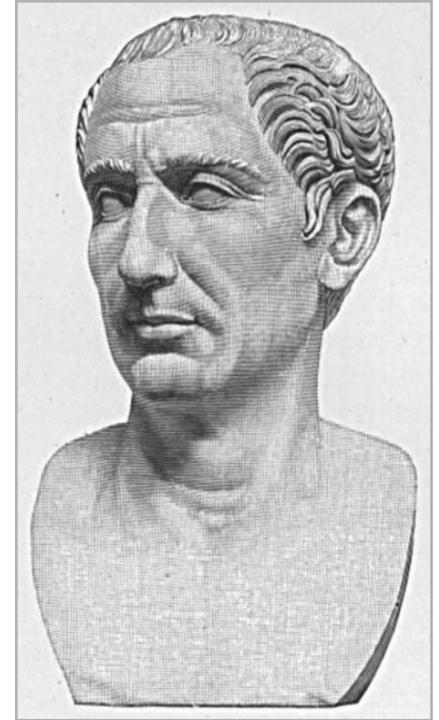
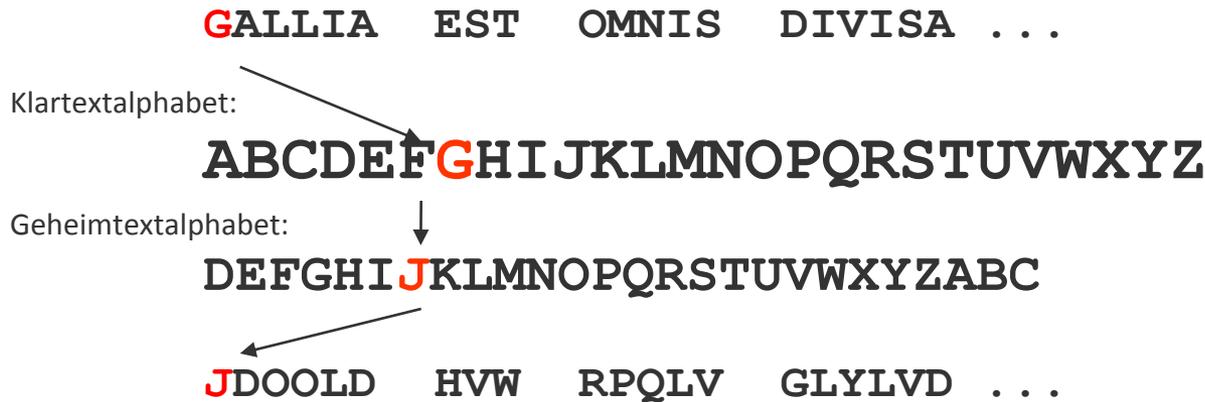
- **Tattoo auf kahlgeschorenen Kopf eines Sklaven** (verdeckt von nachgewachsenen Haaren)
- **Atbash** (um 600 v. Chr.)
 - Hebräische Geheimschrift, umgedrehtes Alphabet
- **Skytale von Sparta** (etwa 500 v. Chr.)
 - Beschrieben vom griechischen Historiker/Schriftsteller Plutarch (45 - 125 n. Chr.)
 - 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 (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 OTP*
- **Lochschablonen** (Fleißner)
- **Permutationsverschlüsselung**
 - "Doppelwürfel"
 - Reine Transposition / sehr effektiv

* OTP = One-Time-Pad

Screenshot of the "Schlüssel eingabe: Playfair" dialog box in CrypTool 1.4.21. The dialog contains the following elements:

- Optionen:**
 - Text vordatieren
 - Doppelte Zeichen im Schlüssel ignorieren
- Playfair-Schlüssel:**
 - Kurzform des Playfair-Schlüssels: CHARLES
 - 5x5 Matrix
 - 6x6 Matrix
- Schlüsselmatrix:**

C	H	A	R	L	
E	S	B	D	F	
G	I	K	M	N	
O	P	Q	T	U	
V	W	X	Y	Z	
- Buttons:** Verschlüsseln, Entschlüsseln, Abbrechen

Beispiel erste Hälfte 20. Jahrhundert

Elektromechanische Verschlüsselungsmaschinen (Rotormaschinen)

Enigma-Verschlüsselung (Arthur Scherbius, 1878-1929)

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

(http://de.wikipedia.org/wiki/Enigma_%28Maschine%29 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. „No security through obscurity“.
- **One-Time-Pad – Shannon / Vernam**
 - Nachweislich theoretisch sicher, jedoch praktisch kaum anwendbar (nur Rotes Telefon).
- **Shannons Konzepte: Konfusion und Diffusion**
 - Zusammenhang zwischen M, C und K möglichst komplex (M=Message, C=Cipher, K=Key)
 - Jedes Chiffrezeichen sollte von möglichst vielen Klartextzeichen und vom gesamten Schlüssel abhängen
 - „Avalanche effect“ (kleine Änderung, große Wirkung)
- **Trapdoor Function** (Falltür, Einweg-Funktion, ...)
 - in einer Richtung schnell, in die andere (ohne Geheim-Information) nicht
 - nur mit dem Geheimnis geht auch die andere Richtung (Zugang zur Falltür)



Beispiel für die Verletzung des Kerckhoffs-Prinzips

Geheimnis sollte nur im Schlüssel und nicht im Algorithmus liegen

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

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

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

Beispiel für eine One-Time-Pad-Adaption



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



Schlüsselverteilungsproblem

Schlüsselverteilung bei symmetrischer Verschlüsselung

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

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

Das sind bei

$n = 100$ Personen bereits

$S_{100} = 4.950$ Schlüssel; bei

$n = 1.000$ Personen sind es

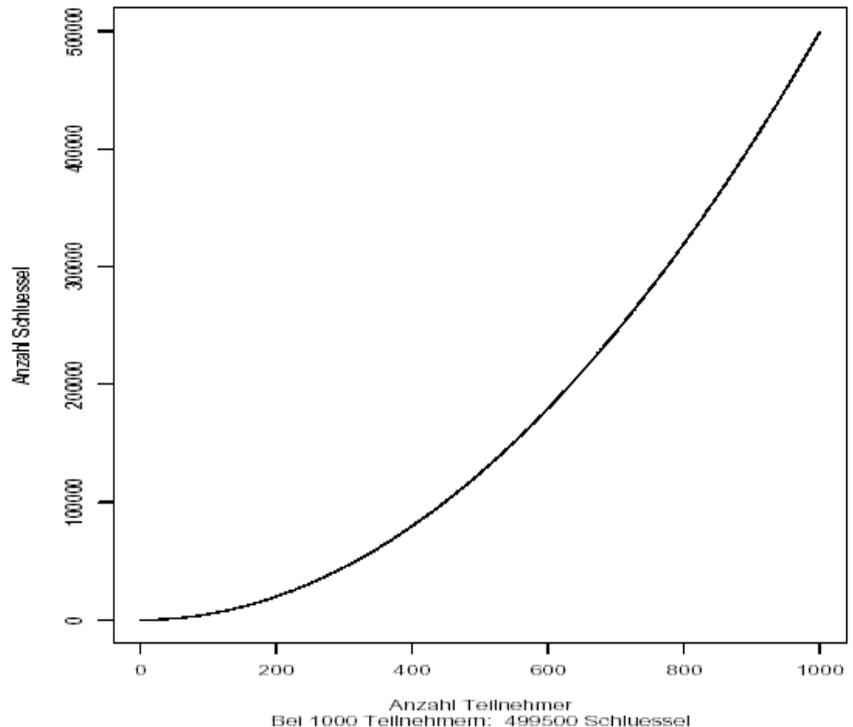
$S_{1000} = 499.500$ Schlüssel.

(Quadratischer Anstieg:

Faktor 10 mehr Personen,

Faktor 100 mehr Schlüssel)

Entwicklung der Schlüsselszahl



Kryptographie – Entscheidende Erkenntnisse (2)

Lösung des Schlüsselverteilungsproblems durch asymmetrische Kryptographie

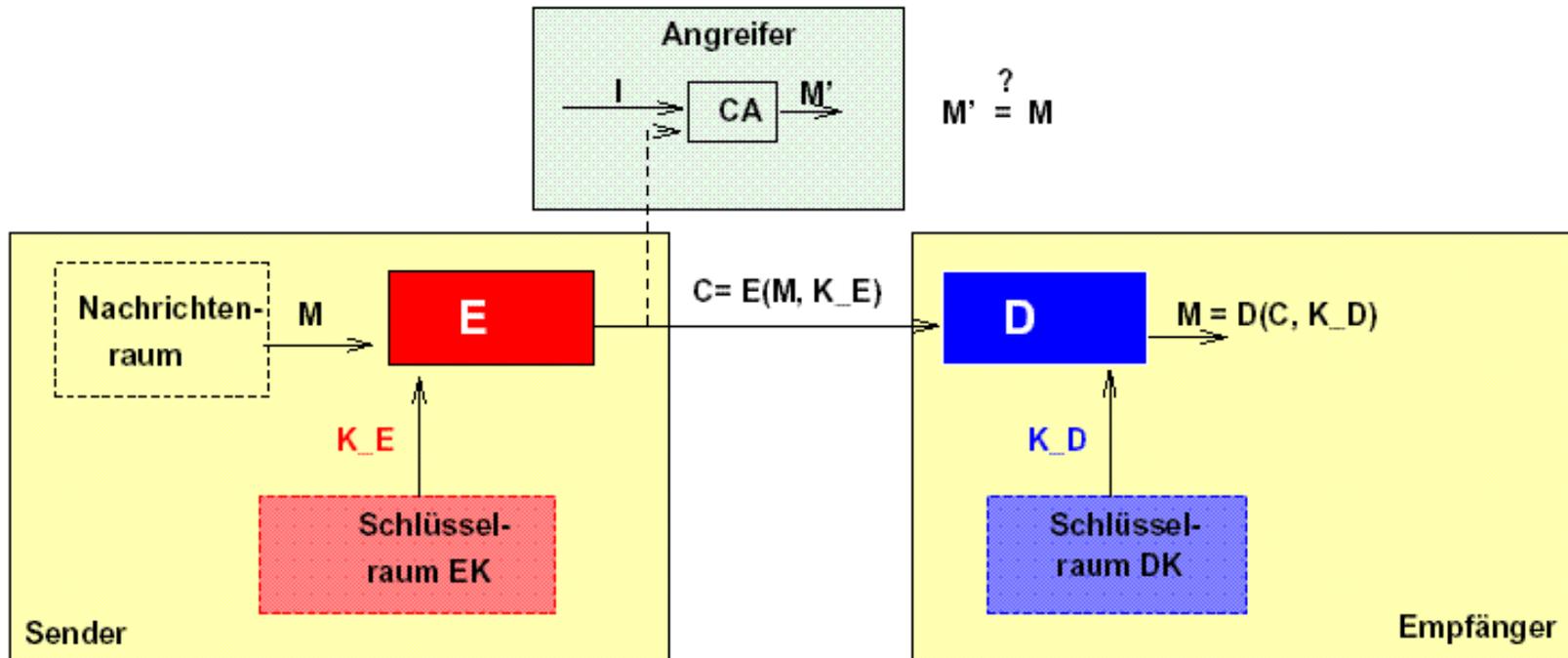
Asymmetrische Kryptographie

- 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 : $K_E = K_D$ geheim! z.B. AES

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

öffentlich geheim!

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-200 (CrypTool-Skript, Kap. 4.11.3)
 - Sehr hohe Intensität in der aktuellen Forschung:
Faktorisierung, nicht-parallelisierbare Algorithmen (wegen Quantencomputing), besseres Verständnis von Protokoll-Schwächen und Zufallszahlengeneratoren, ...).
- Entscheidender Irrtum: „*Echte Mathematik*“ hat keine Auswirkungen auf den Krieg. (G.H. Hardy, 1940)
- Hersteller entdecken **Sicherheit** als ein zentrales **Kaufkriterium**

Demo mit CrypTool

- **Statistische Analyse**

- **Zweimal nacheinander ist nicht immer besser:**

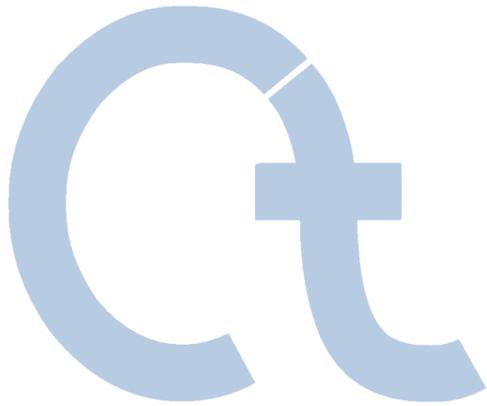
 - Caesar: $C + D = G$ ($3 + 4 = 7$)

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

 - "Hund" + "Katze" = "RUGCLENWGYXDATRNNMH")

- **Vernam (OTP)**

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



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

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



Deutsch, Englisch,
Polnisch und
Spanisch

CrypTool-Programm

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

AES-Tool

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

Lernbeispiel

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

Umfangreiche Online-Hilfe (HTML-Help)

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

Skript (.pdf-Datei) mit Hintergrundinformationen

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

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

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

Authorware-Lernprogramm zur Zahlentheorie

Funktionsumfang (1)

Kryptographie

Verschlüsselungsklassiker

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

Zum besseren Nachvollziehen von Literaturbeispielen ist

- Alphabet wählbar
- Behandlung von Leerzeichen etc. einstellbar

Kryptoanalyse

Angriffe auf klassische Verfahren

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

Unterstützende Analyseverfahren

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

Funktionsumfang (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 oder 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, Vitany, Entropie
- Gleitende Häufigkeit, Histogramm
- n-Gramm-Analyse, Autokorrelation
- ZIP-Kompressionstest

Funktionsumfang (4)

Visualisierungen / Demos

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



Funktionsumfang (5)

Weitere Funktionen

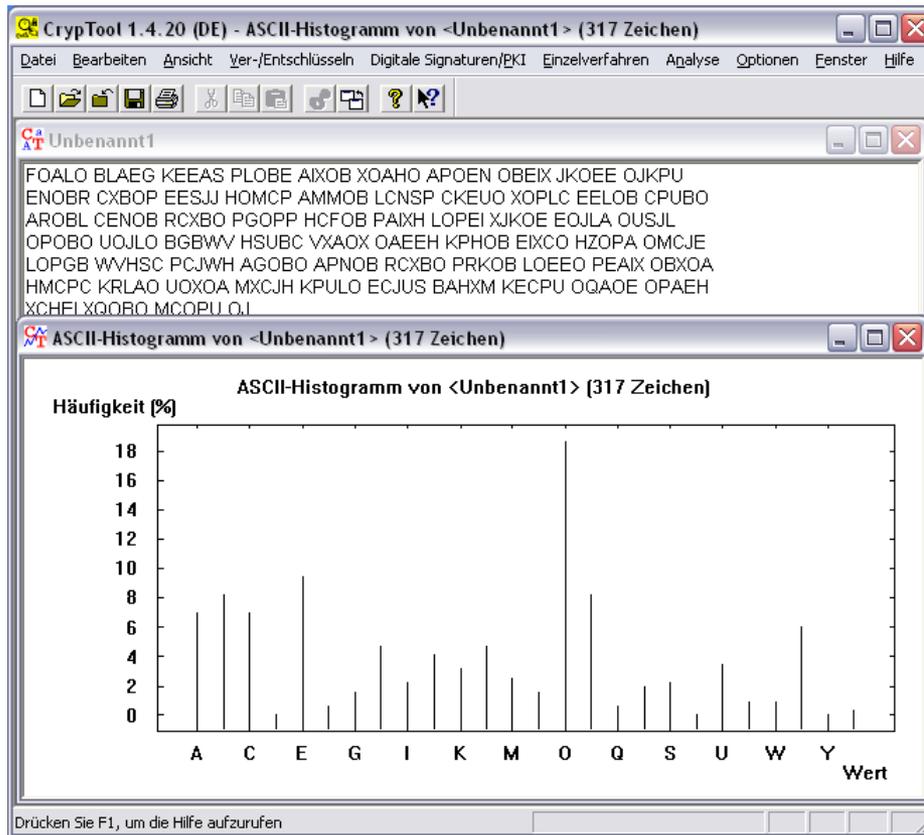
- Homophone und Permutationsverschlüsselung (Doppelwürfel)
- PKCS #12-Import/Export für PSEs (Personal Security Environment)
- Hashwerte großer Dateien berechnen, ohne sie zu laden
- Generische Brute-Force-Attacke auf beliebige moderne symmetrische Algorithmen
- ECC-Demo (als Java-Applikation)
- Passwort-Qualitätsmesser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



Sprachstruktur analysieren

Anzahl Einzelzeichen, n-Gramme, Entropie

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



Entropie <Unbenannt1>

Das analysierte Dokument enthält 23 der 26 Zeichen des eingestellten Alphabets.

**Seine Entropie beträgt 3,99
(maximal mögliche Entropie 4,70).**

OK

N-Gramm-Liste von Unbenannt1

Auswahl:

- Histogramm
- Digramm
- Trigramm
- 4-Gramm

Anzeige der 26 häufigsten N-Gramme (erlaubte Werte: 1-5000).

Liste berechnen

Nr.	Zeichen...	Häufigkeit in %	Häufigkeit
1	D	18.6120	59
2	E	9.4637	30
3	B	8.2019	26
4	P	8.2019	26
5	A	6.9401	22
6	C	6.9401	22
7	X	5.9937	19
8	H	4.7319	15
9	L	4.7319	15
10	J	4.1009	13
11	U	3.4700	11
12	K	3.1546	10
13	M	2.5237	8
14	I	2.2082	7
15	S	2.2082	7
16	R	1.8927	6
17	G	1.5773	5
18	N	1.5773	5
19	V	0.9464	3
20	W	0.9464	3
21	F	0.6309	2
22	Q	0.6309	2
23	Z	0.3155	1

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** ⇨ „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** ⇨ „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

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

Faktorisierungs-Beispiel 1:

316775895367314538931177095642205088158145887517

48-stellige Dezimalzahl

=

3 * 1129 * 6353 * 1159777 * 22383173213963 * 567102977853788110597

Faktorisierungs-Beispiel 2:

$2^{250} - 1$

75-stellige Dezimalzahl

=

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

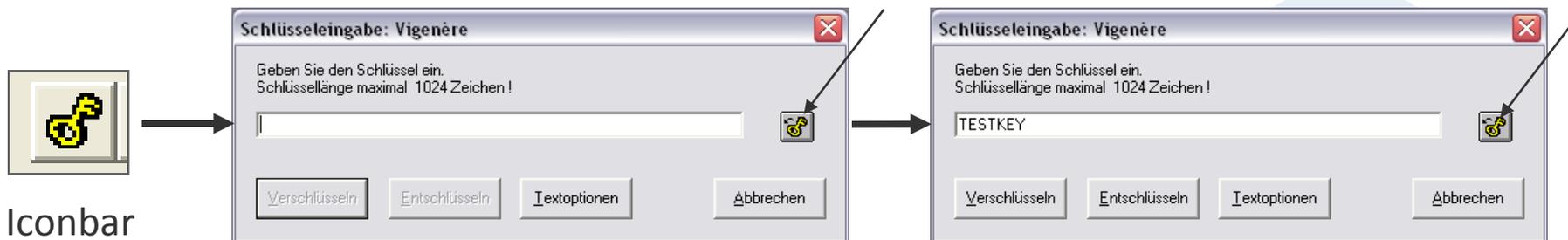
Konzepte zur Benutzerfreundlichkeit

1. Kontextsensitive Hilfe (F1)

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

2. Einfügen von Schlüsseln in die Schlüsseleingabe-Maske

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



Herausforderungen für den Programmierer

1. Viele Funktionen parallel laufen lassen

- Bei der Faktorisierung laufen die verschiedenen Algorithmen in Threads.

2. Hohe Performance

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

3. Speicherbeschränkung beachten

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

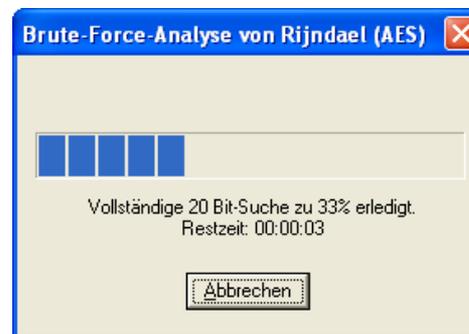
4. Zeitmessung und -abschätzung

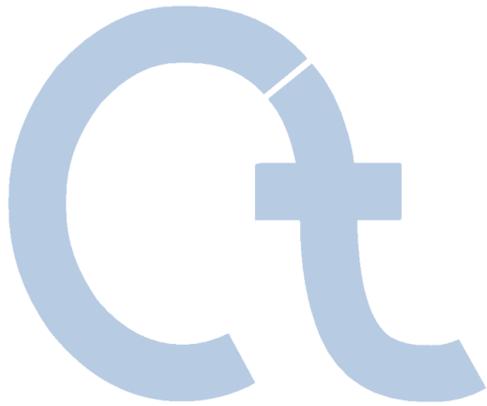
- Ausgabe der Ellapsed Time bei Brute-Force

5. Wiederverwendung / Integration

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

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





I. CrypTool und Kryptologie –
Überblick

II. Was bietet CrypTool?

III. Ausgewählte Beispiele

IV. Projekt / Ausblick / Kontakt

CrypTool-Anwendungsbeispiele

Übersicht der Beispiele

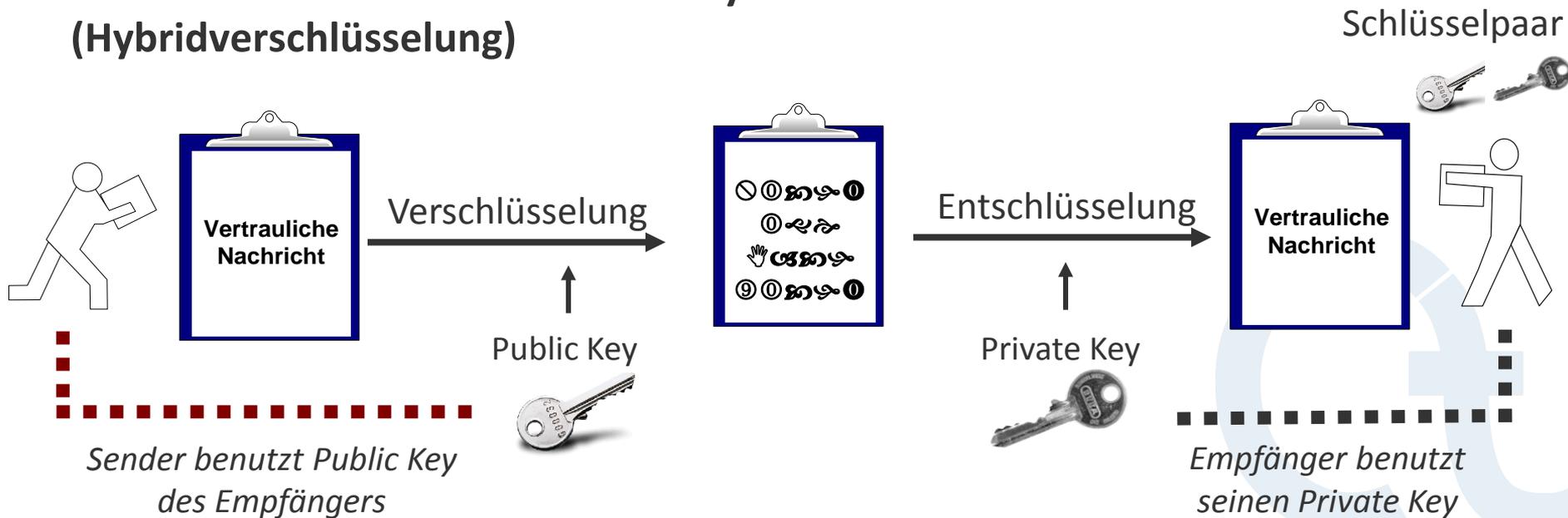
1. [Verschlüsselung mit RSA / Primzahltests / Hybridverschlüsselung und Digitale Zertifikate / SSL](#)
2. [Elektronische Signatur visualisiert](#)
3. [Angriff auf RSA-Verschlüsselung \(Modul N zu kurz\)](#)
4. [Analyse der Verschlüsselung im PSION 5](#)
5. [Schwache DES-Schlüssel](#)
6. [Auffinden von 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. [Visualisierung von AES](#)
16. [Visualisierung der Enigma-Verschlüsselung](#)
17. [Erzeugung eines Message Authentication Code \(MAC\)](#)
18. [Hash-Demo](#)
19. [Lernprogramm zur Zahlentheorie und zur asymmetrischen Verschlüsselung](#)
20. [Punktaddition auf elliptischen Kurven](#)
21. [Passwort-Qualitätsmesser und Passwort-Entropie](#)
22. [Brute-Force-Analyse](#)
23. [CrypTool Online-Hilfe](#)



Anwendungsbeispiele (1)

Verschlüsselung mit RSA

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



Anwendungsbeispiele (1)

Verschlüsselung mit RSA – Mathematischer Hintergrund / Verfahren

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

wobei:

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

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

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

- n ist der Modulus (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: $2^r \leq n$ ($2^{r-1} < n$)

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-Bibliothek (**G**NU **M**ultiple **P**recision Arithmetic Library) in CrypTool integriert.

Primzahltest

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

Algorithmen zum Primzahltest

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

Primzahltest

Zahl aus Datei laden

Zu testende Zahl

Ergebnis 

Zahl testen

Abbrechen

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

Anwendungsbeispiele (1)

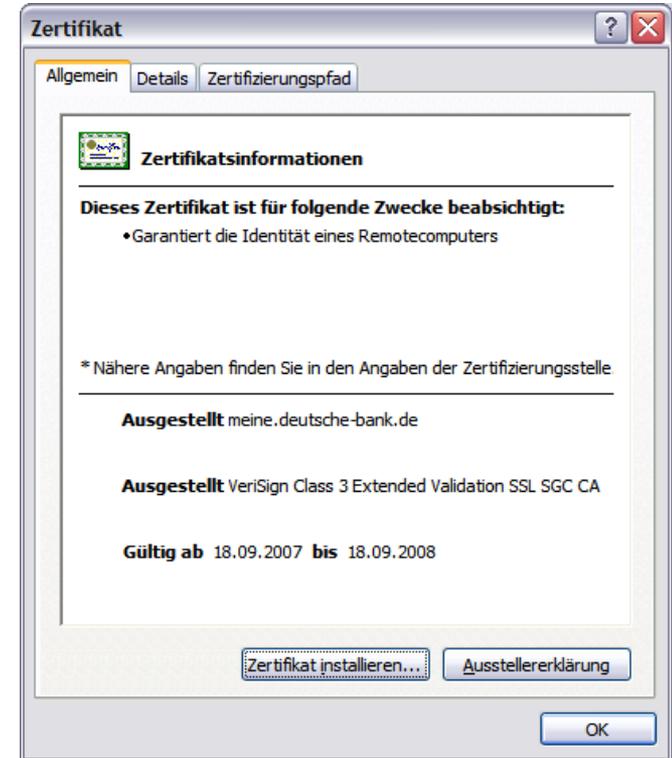
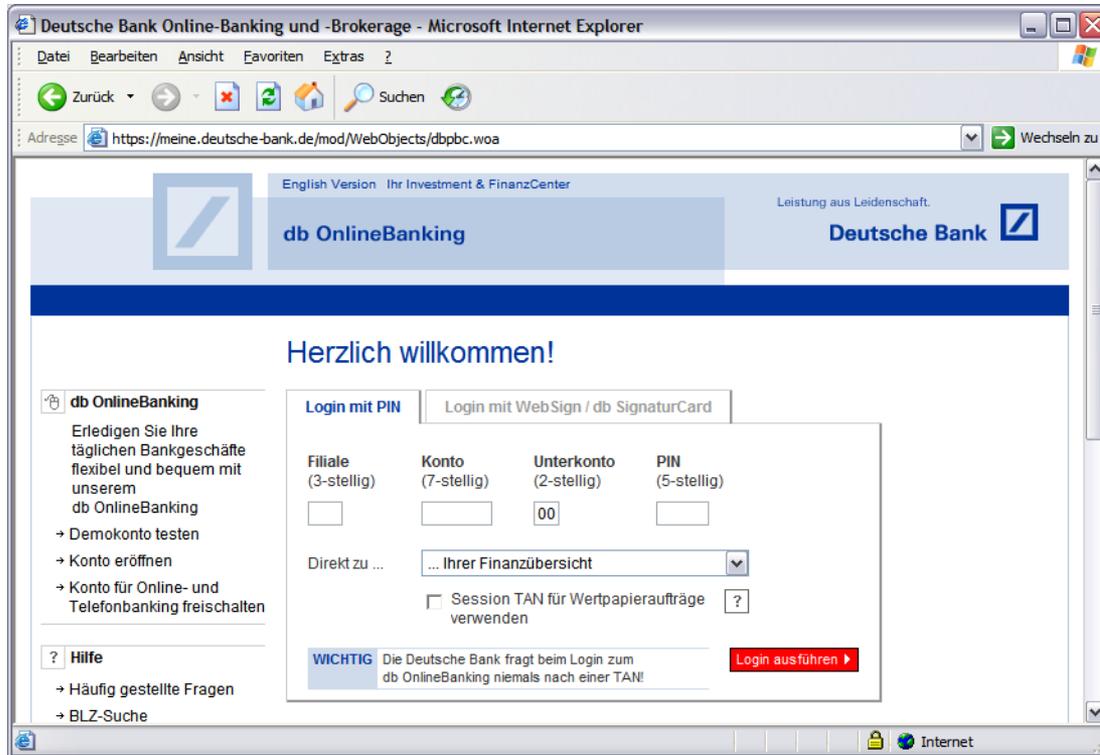
Hybridverschlüsselung und Digitale Zertifikate

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



Anwendungsbeispiele (1)

Gesicherte Online-Verbindung mit SSL und Zertifikaten

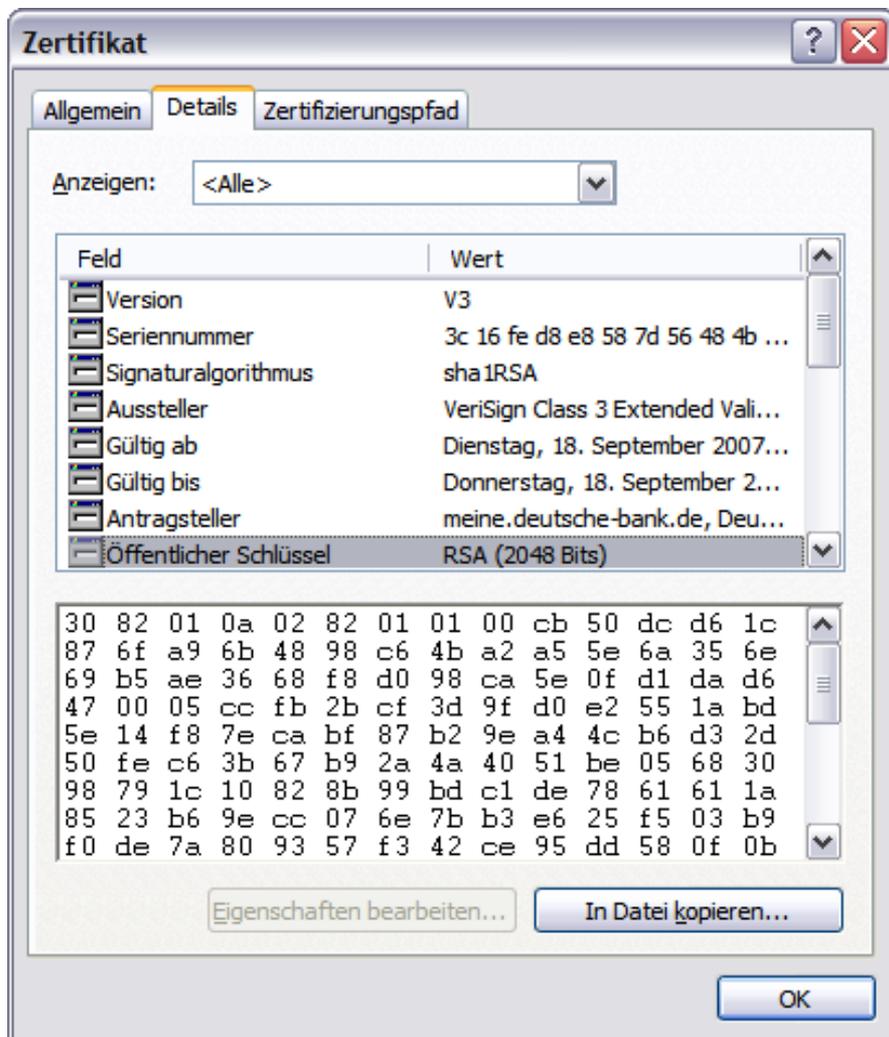


D.h. die Verbindung ist (zumindest einseitig) authentisiert und der übertragene Inhalt wird stark verschlüsselt.



Anwendungsbeispiele (1)

Attribute / Felder von Zertifikaten



Grundlegende Attribute / Felder

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

Öffentlicher Schlüssel



Anwendungsbeispiele (1)

Aufbau einer gesicherten SSL-Verbindung (Server Authentication)

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 7.
(Entschlüsselung durch privaten Schlüssel des Servers)



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

Anmerkungen zur SSL-Verbindung

- ad (1): SSL Verbindungsaufbau – hierbei wird u.a. ausgehandelt welche Eigenschaften der Session Key besitzen soll (z.B. Bit-Länge) und welcher Algorithmus für die symmetrische Verschlüsselung verwendet werden soll (z.B. 3DES, AES).
- ad (2): Sofern Zwischenzertifikate notwendig sind (bei mehrstufigen 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)

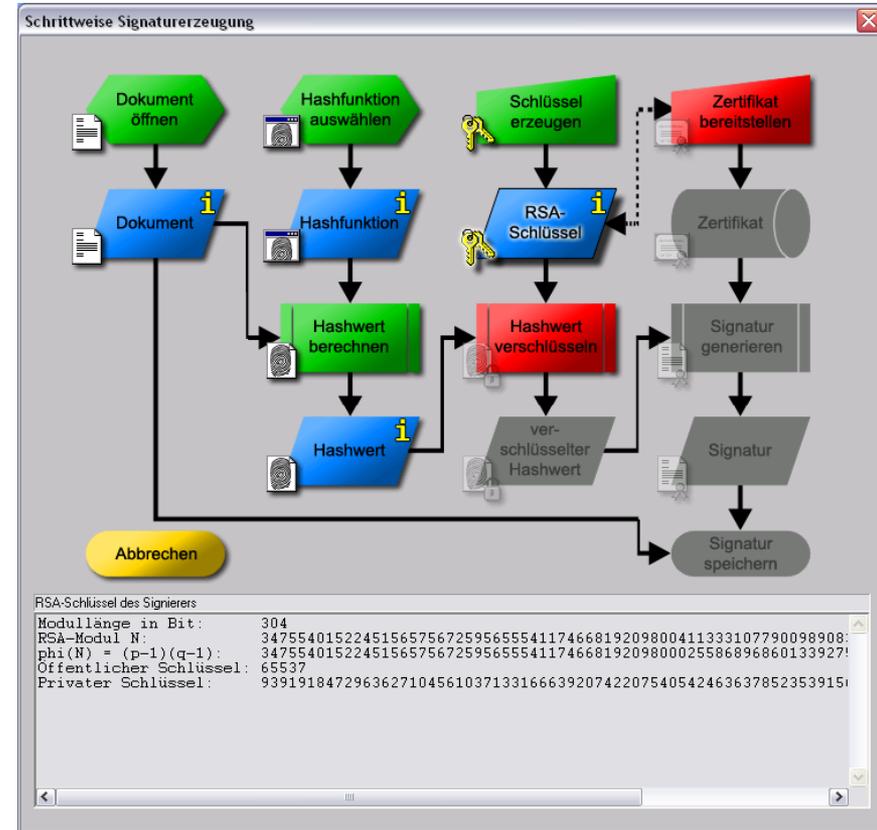
Elektronische Signatur visualisiert

Elektronische Signatur

- Wird immer wichtiger durch
 - Gleichstellung mit manueller Unterschrift (Signaturgesetz)
 - 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



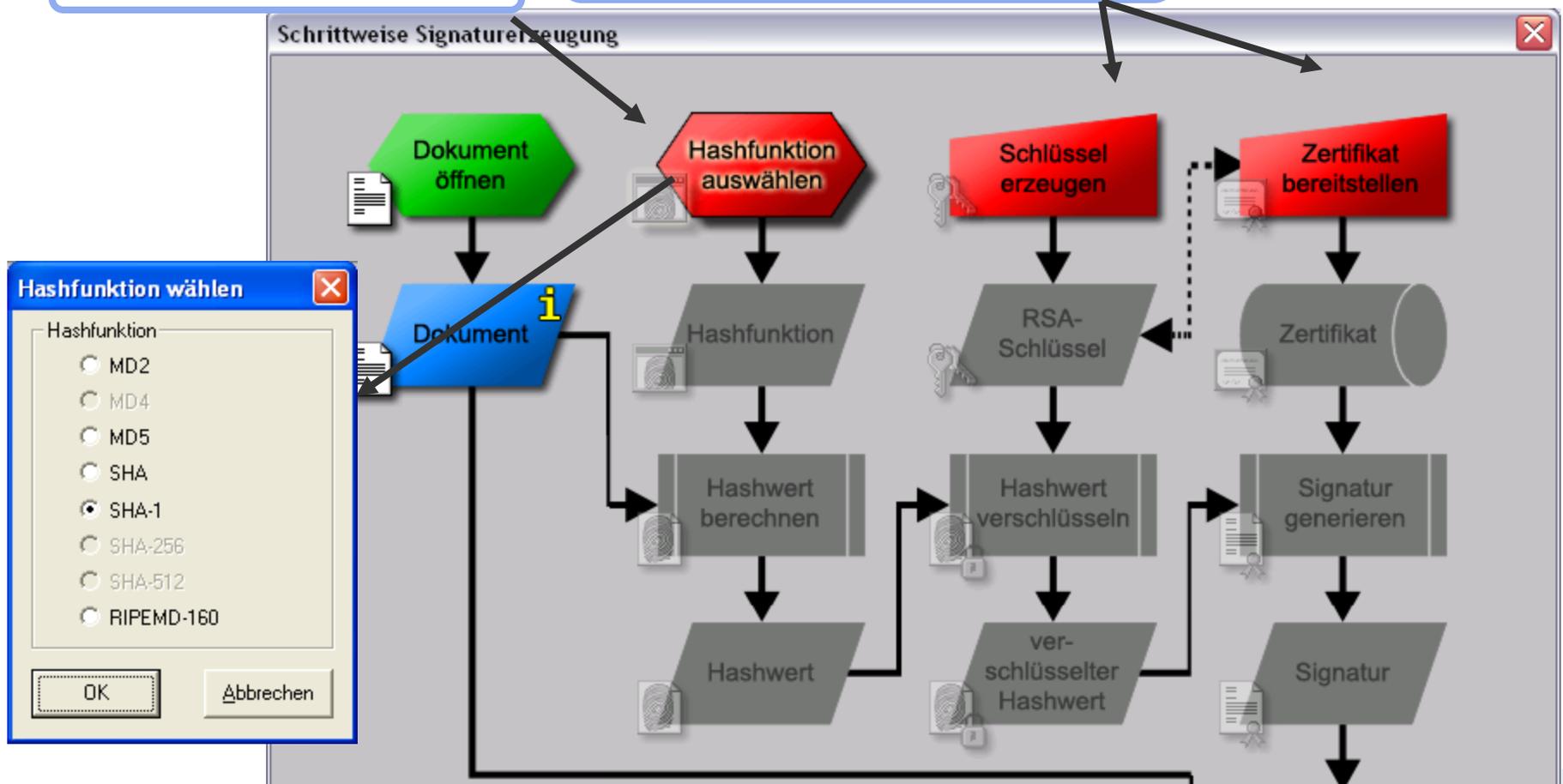
Menü: „Digitale Signaturen/PKI“ \
„Signaturdemo (Signaturerzeugung)“

Anwendungsbeispiele (2)

Elektronische Signatur visualisiert: a) Vorbereitung

1. Hashfunktion wählen

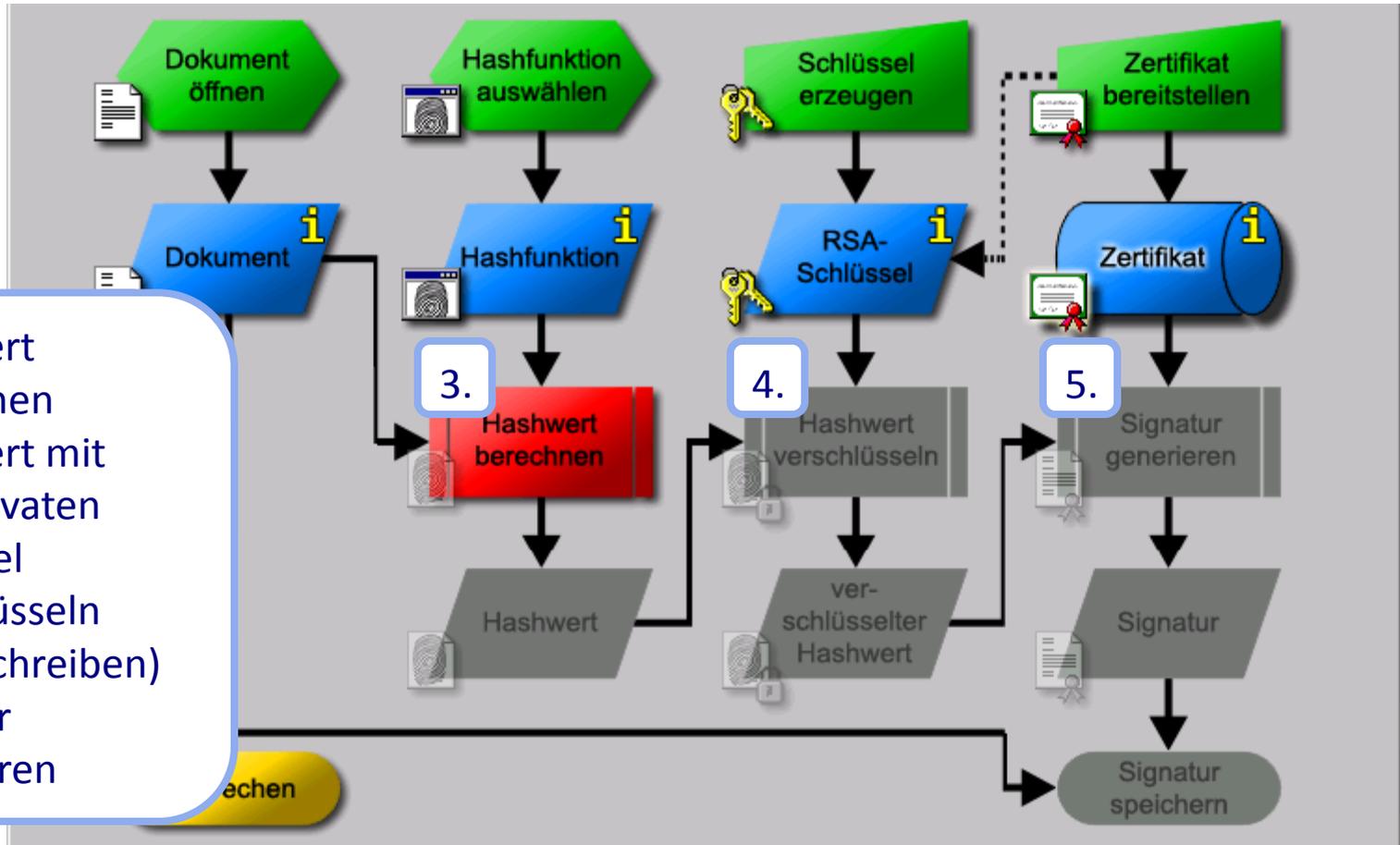
2. Schlüssel und Zertifikat bereitstellen (nicht gezeigt)



Anwendungsbeispiele (2)

Elektronische Signatur visualisiert: b) Kryptographie

- 3. Hashwert berechnen
- 4. Hashwert mit dem privaten Schlüssel verschlüsseln (unterschreiben)
- 5. Signatur generieren

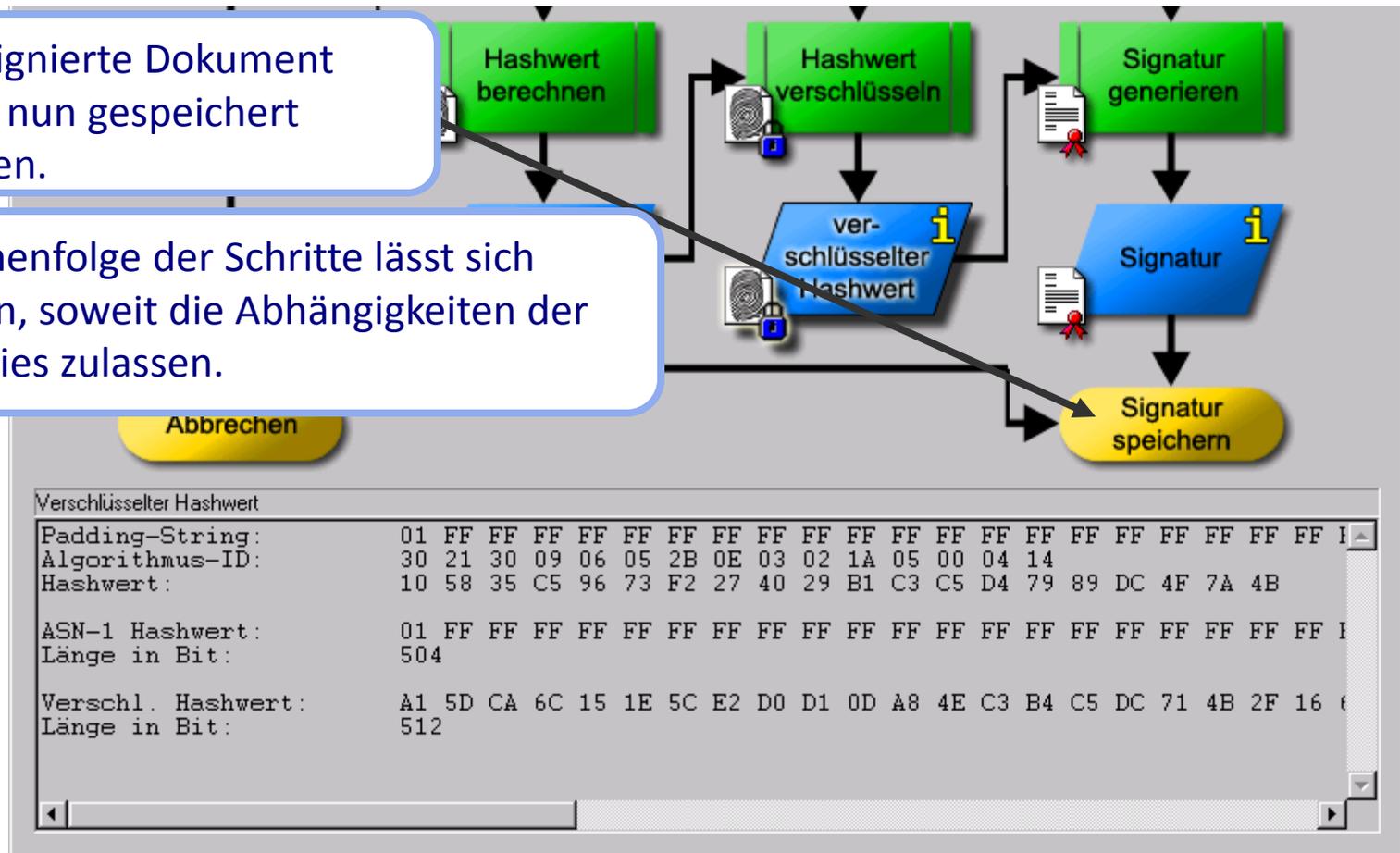


Anwendungsbeispiele (2)

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 zu kurzen 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 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...“

RSA-Demo

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

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

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

Faktorisierungsangriff

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

RSA-Modul faktorisieren...

RSA-Parameter

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

Parameter aktualisieren

RSA-Verschlüsselung mit e / Entschlüsselung mit d

1. Öffentliche RSA-Parameter N und e eingeben

2. Faktorisieren

Anwendungsbeispiele (3)

Kurzer RSA-Modul: RSA-Modul faktorisieren

Faktorisieren einer Zahl

Algorithmen zur Faktorisierung

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

Eingabe

Geben Sie die zu faktorisierende Zahl ein:

63978486879527143858831415041

Faktorisierungsergebnis

Die Faktorisierung wird in dem Format $\langle z1^{a1} * z2^{a2} * \dots * zn^{an} \rangle$ dargestellt. Zusammengesetzte Zahlen sind rot markiert.

Letzte Faktorisierung durch: Pollard

Insgesamt benötigte Zeit: 0,391 Sekunden.

Produktdarstellung der Faktorisierung:

145295143558111 * 440334654777631

Details

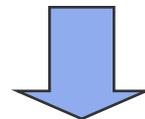
Schließen

CrypTool

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

OK

3. Faktorisierung ergibt p und q



Anwendungsbeispiele (3)

Kurzer RSA-Modul: Geheimen Schlüssel d bestimmen

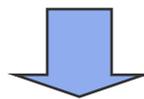
The screenshot shows the 'RSA-Demo' application window. It has a title bar with a close button. The main content area is divided into several sections:

- Options:** A radio button is selected for 'Wählen Sie 2 Primzahlen p und q. Die Zahl $N = pq$ ist der öffentliche RSA-Modul und $\phi(N) = (p-1)(q-1)$ ist die Eulersche Zahl. Der öffentliche Schlüssel e ist teilerfremd zu $\phi(N)$. Daraus wird der geheime Schlüssel $d = e^{-1} \pmod{\phi(N)}$ berechnet.' Another radio button is for '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:** Two text input fields for 'Primzahl p' (145295143558111) and 'Primzahl q' (440334654777631). A 'Primzahlen generieren...' button is to the right.
- RSA-Parameter:** Four text input fields: 'RSA-Modul N' (63978486879527143858831415041) labeled '(öffentlich)', 'phi(N) = (p-1)(q-1)' (63978486879526558229033079300) labeled '(geheim)', 'Öffentlicher Schlüssel e' (17579), and 'Geheimer Schlüssel d' (10663687727232084624328285019). A 'Parameter aktualisieren' button is to the right.
- RSA-Verschlüsselung mit e / Entschlüsselung mit d:** Radio buttons for 'Text' and 'Zahlen'. A button for 'Optionen für Alphabet und Zahlensystem...' is to the right.

Wechsel in die Ansicht des Besitzers des geheimen Schlüssels.

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

5. Optionen einstellen



Anwendungsbeispiele (3)

Kurzer RSA-Modul: Optionen einstellen

Optionen für die RSA-Demo

Alphabetoptionen

Alle 256 Zeichen Anzahl Zeichen: 27

Alphabet vorgeben

ABCDEFGHIJKLMNOPQRSTUVWXYZ

RSA-Variante

Normal Dialog der Schwestern

Methode, wie ein Block als Zahl codiert wird

b-adisch Basissystem

Blocklänge

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

Blocklänge in Zeichen: (Maximale Blocklänge 14 Zeichen)

Zahlensystem

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

Dezimal Binär Oktal Hexadezimal

OK Abbrechen

6. Alphabet wählen

7. Kodierung wählen

8. Blocklänge wählen



Anwendungsbeispiele (3)

Kurzer RSA-Modul: Text entschlüsseln

RSA-Parameter

RSA-Modul N	<input type="text" value="63978486879527143858831415041"/>	(öffentlich)
$\phi(N) = (p-1)(q-1)$	<input type="text" value="63978486879526558229033079300"/>	(geheim)
Öffentlicher Schlüssel e	<input type="text" value="17579"/>	
Geheimer Schlüssel d	<input type="text" value="10663687727232084624328285019"/>	

RSA-Verschlüsselung mit e / Entschlüsselung mit d

Eingabe als Text Zahlen

Chiffretext in Zahlendarstellung zur Basis 10 .

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

Ausgabertext aus der Entschlüsselung (in Blöcken der Länge 11; das Symbol '#' dient nur als Trennzeichen).

Klartext

9. Ciphertext eingeben

10. Entschlüsseln

Anwendungsbeispiele (4)

Analyse der Verschlüsselung im PSION 5

Praktische Durchführung der Kryptoanalyse:

Angriff auf die Verschlüsselungsoption der Textverarbeitungsapplikation im PSION 5 PDA



Gegeben: eine auf dem PSION verschlüsselte Datei

Voraussetzung

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

Vorgehen

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

*wahrscheinlich klassische
Verschlüsselung*

Anwendungsbeispiele (4)

PSION-PDA: Entropie bestimmen, Kompressionstest

CrypTool - Gleitende Häufigkeit von <psion-enc.hex>

Datei Bearbeiten Ansicht Schlüsselverwaltung Einzelverfahren Optionen Fenster Hilfe

psion-enc.hex

00000	77	F8	C2	99	87	FD	FB	A6	59	66	98	C7	87	DD	CD	wøÅ. .ýù Yf. Ç. Ý
0000F	5C	76	29	E8	83	91	04	DF	29	25	C5	75	5A	90	6F	\vèè. BvZ
0001E	FC	FE	86	FA	61	87	88	FD	FC	91	AB					Uz
0002D	04	17	CA	D1	91	07	31	93	FE	DE	D7					Lé.
0003C	A2	74	F5	08	74	EB	CD	99	7D	FB	EF					2?ác
0004B	A5	41	9C	CC	87	99	43	E7	76	90	R0					I\$wá

Gleitende Häufigkeit von <psion-enc.hex>

Verschiedene Zeichen pro 64 Byte Block

60
55
50
45
40

1 5000 10000 15000 20000

CrypTool

Kompressionsgrad: 21 %

OK

Entropie <psion-enc.hex>

Das analysierte Dokument enthält alle 256 möglichen Bytes.

Seine Entropie beträgt 7.56 (maximal mögliche Entropie 8.00).

OK

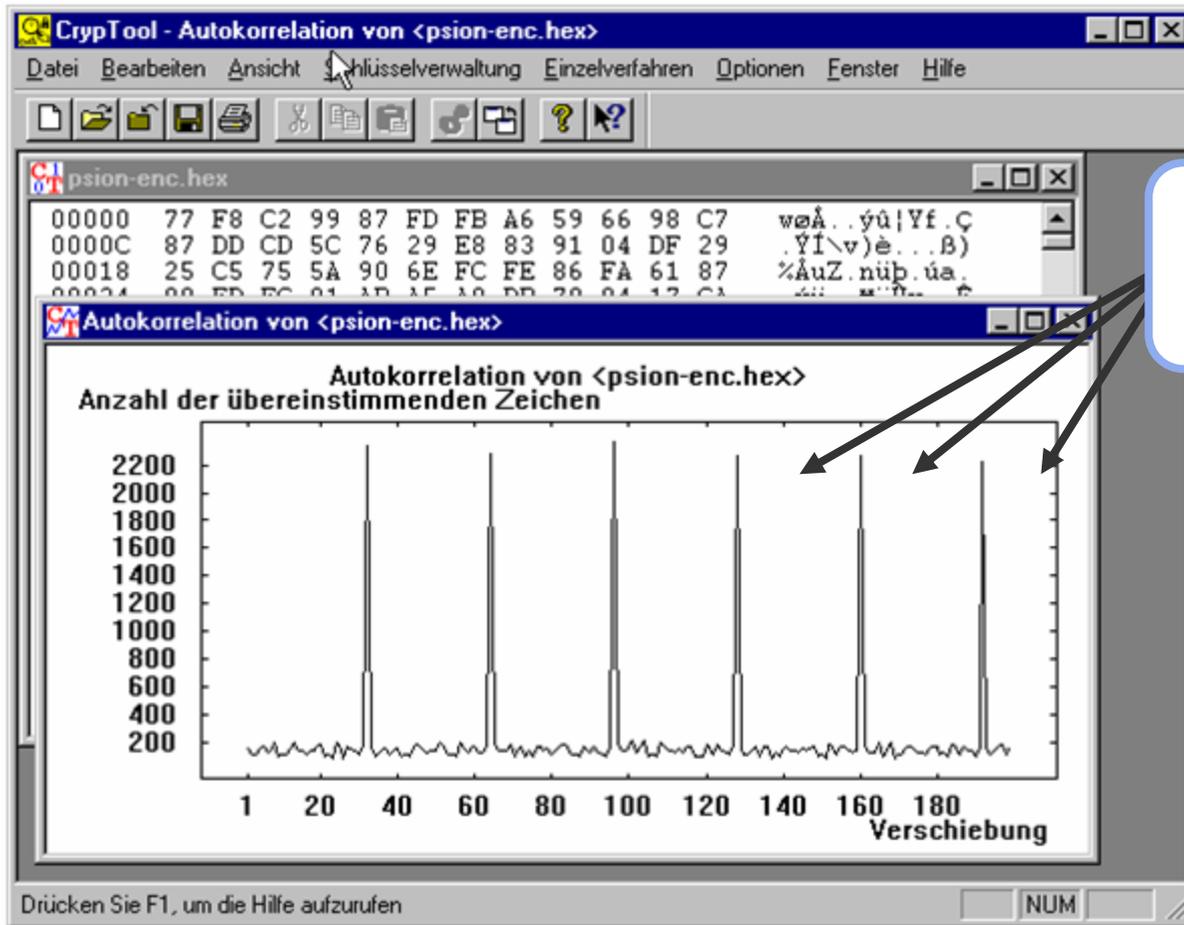
Drücken Sie F1, um die Hilfe aufzurufen

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

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

Anwendungsbeispiele (4)

PSION-PDA: Autokorrelation bestimmen



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

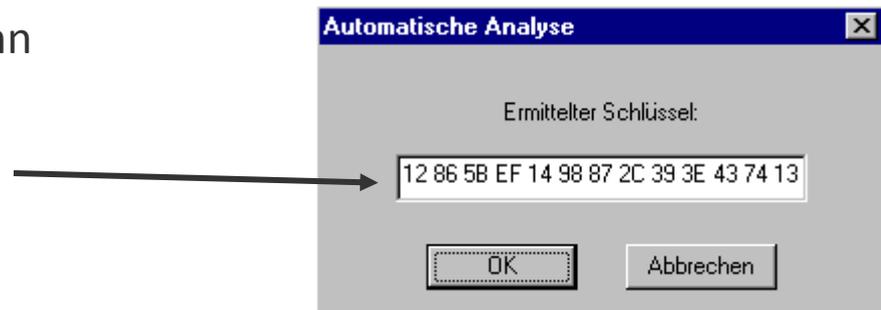
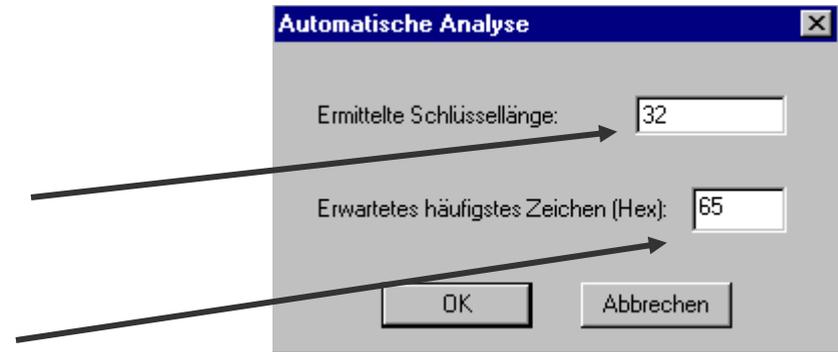
* Diese verschlüsselte Datei wird mit CrypTool ausgeliefert (siehe CrypTool\examples\psion-enc.hex)

Anwendungsbeispiele (4)

PSION-PDA: Automatische Analyse

Automatische Analyse 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



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. ←

```
Automatische ADD-Analyse von <psion-enc.hex>, Schlüssel: <12 86 5B EF 14 98 87 2C 39 3E 43...
00000 65 72 67 AA 73 65 74 7A 20 28 55 53 74 47 29 06  erg³setz (UStG).
00010 06 06 8A 72 73 74 65 72 65 41 62 B8 A8 68 6E AE  ...rstereAb,`hn@
00020 74 74 06 98 74 65 75 65 72 67 65 67 65 6E 73 74  tt..teuergegenst
00030 61 6E A9 20 75 6E 64 20 8C 65 6C B9 BA 6E 67 B8  an@ und .el¹ng.
00040 62 65 72 AA 69 63 68 06 06 A7 20 31 2E 06 28 31  ber³ich..$ 1..(1
00050 29 20 89 65 72 20 55 6D B8 61 74 BF B8 74 65 BA  ) .er Um,atì,te²
00060 65 72 20 BA 6E 74 65 72 6C 69 65 67 65 6E 20 64  er ²nterliegen d
00070 69 65 65 66 6F 6C 67 65 B3 64 65 B3 65 55 6D B8  ieefolge³de³eUm,
00080 E4 74 7A AA 3A 06 31 2E 20 64 69 65 20 4C 69 65  ätz³:.1. die Lie
00090 66 65 B7 75 6E 67 65 6E 65 75 6E A9 65 73 6F B3  fe·ungeneun@eso³
000A0 73 74 69 AC 65 6E 20 4C 65 69 73 74 75 6E 67 65  sti·en Leistunge
000B0 6E 2C 65 64 69 65 20 65 AE 6E 20 9A B3 74 65 B7  n,edie e@n .³te·
000C0 6E 65 68 B2 65 72 20 69 6D 20 49 6E 6C 61 6E 64  neh²er im Inland
000D0 20 67 AA 67 65 6E 20 45 B3 74 67 AA B1 74 20 AE  g³gen E³tg³tt @
000E0 6D 20 52 A6 68 6D 65 6E 20 73 65 69 6E 65 73 20  m R|hmen seines
000F0 55 6E B9 65 72 6E 65 68 B2 65 6E B8 65 61 75 B8  Un¹erneh²en,eau,
```

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

Anwendungsbeispiele (4)

PSION-PDA: Bestimmung der restlichen Schlüsselbytes

Schlüssel während der automatischen Analyse in die Zwischenablage kopieren

Im Hexdump der automatischen Analyse

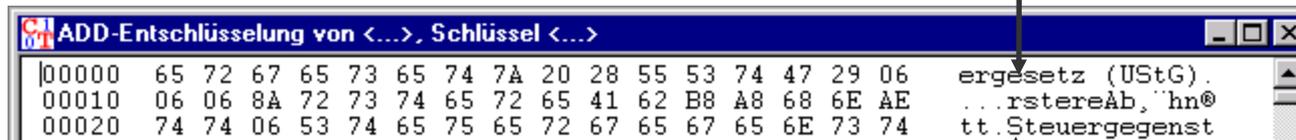
- Falsche Bytepositionen bestimmen, z.B. 0xAA an Position 3
- Korrespondierende korrekte Bytes erraten und notieren: „e“ = 0x65

Im Hexdump der verschlüsselten Ausgangsdatei

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

Schlüssel aus der Zwischenablage

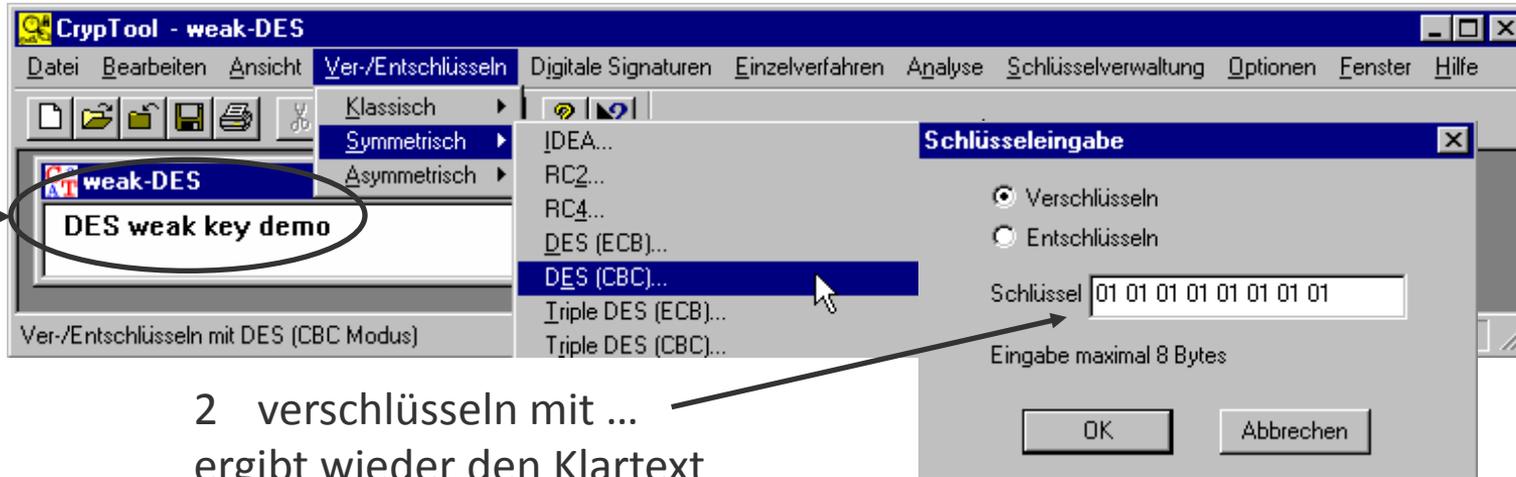
- Korrigieren 12865B**34**1498872C393E43741396A45670235E111E907AB7C0841...
- Verschlüsseltes Ausgangsdokument mittels binärer Addition entschlüsseln
- Nun sind die Bytepositionen 3, 3+32, 3+2*32, ... ok



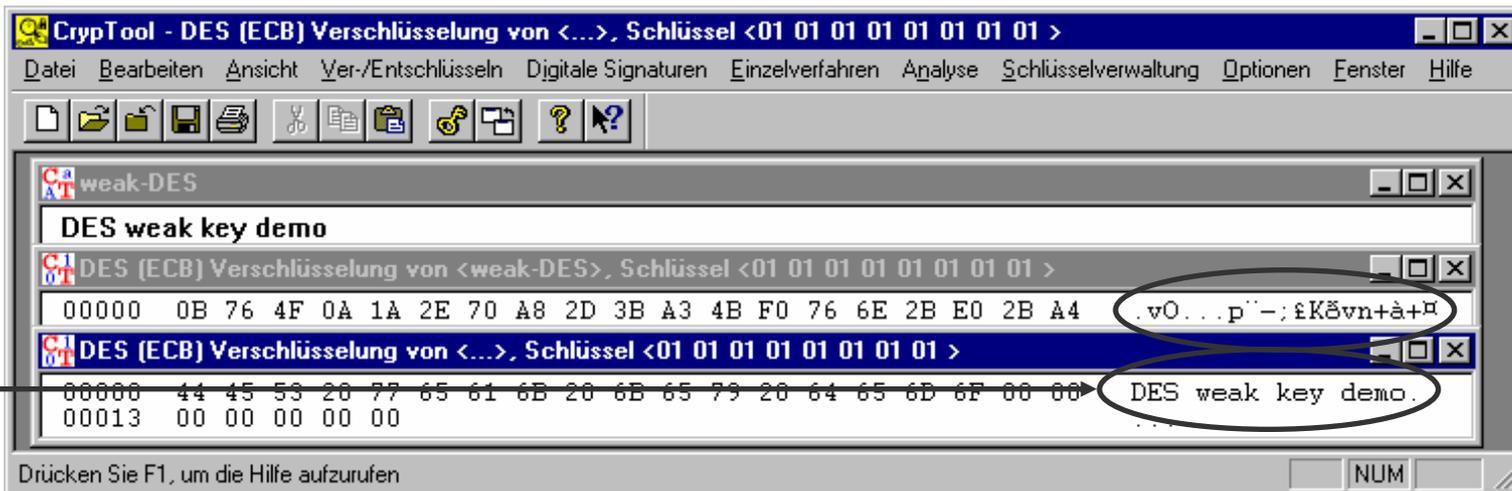
```
ADD-Entschlüsselung von <...>, Schlüssel <...>
|00000 65 72 67 65 73 65 74 7A 20 28 55 53 74 47 29 06  ergesetz (UStG).
|00010 06 06 8A 72 73 74 65 72 65 41 62 B8 A8 68 6E AE  . . rstereAb,`hn@
|00020 74 74 06 53 74 65 75 65 72 67 65 67 65 6E 73 74  tt.Steuergegenst
```

Anwendungsbeispiele (5)

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



2 verschlüsseln mit ...
ergibt wieder den Klartext



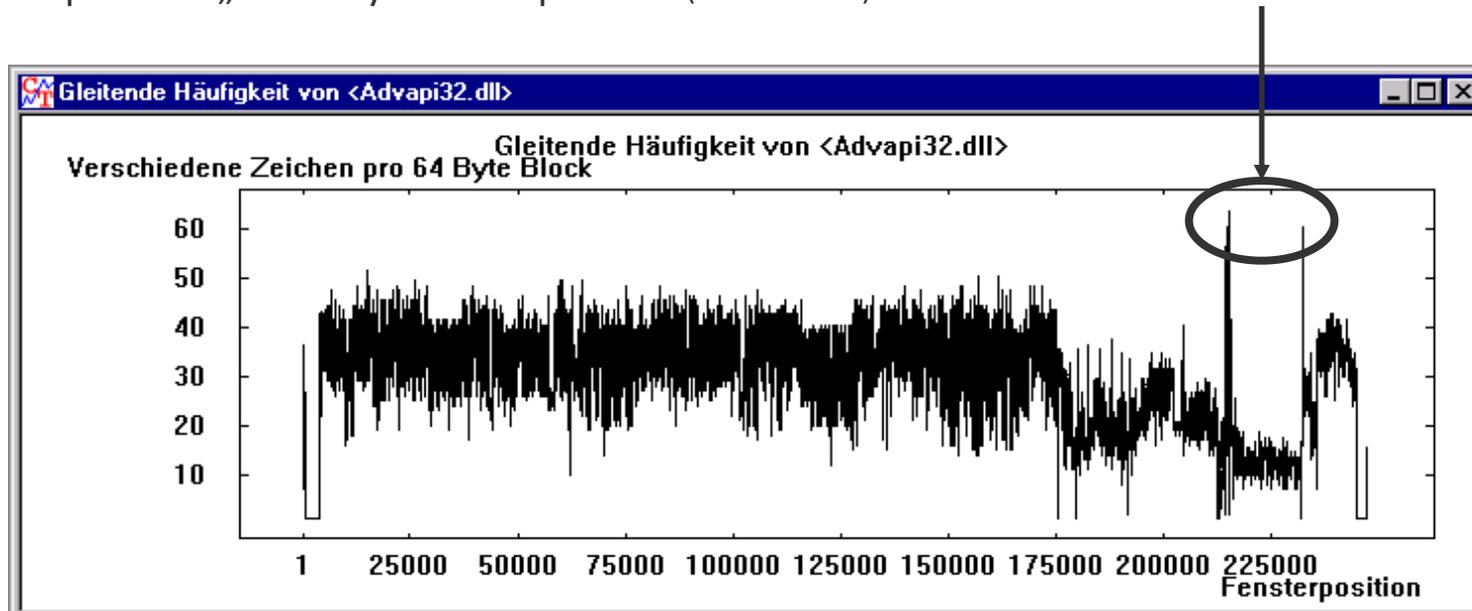
Anwendungsbeispiele (6)

Auffinden von Schlüsselmaterial

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

Hintergrund:

- Diese Daten sind „zufälliger“ als Text oder Programmcode.
- Sie sind als Peak in der „gleitenden Häufigkeit“ zu erkennen.
- Beispiel: der „NSA-Key“ in advapi32.dll (Windows NT)



Anwendungsbeispiele (6)

Vergleich der gleitenden Häufigkeit anderer Dateien

CrypTool - Gleitende Häufigkeit von <AA_Cry-Rijndael-startbeispiel-de.hex>

Datei Bearbeiten Ansicht Schlüsselverwaltung Einzelverfahren Optionen Fenster Hilfe

AA_Cry-startbeispiel.txt

CrypTool

Dies ist eine Textdatei, mit der Sie Ihre ersten Schritte mit CrypTool machen können.

- 1) Sie können diese Datei z.B. über das Menü "Ver/Entschlüsseln \ Klassisch" mit dem Caesar-Verfahren verschlüsseln.
- 2) Den besten Überblick über die Möglichkeiten von CrypTool bietet die Startseite in der Windows Online-Hilfe zu CrypTool. Von der Startseite aus können Sie alle wesentlichen Funktionen über Links erreichen. Die Startseite erreichen Sie über das Menü "Hilfe \ Startseite" oder indem Sie in der Online-Hilfe im Index den Begriff "Startseite" eingeben.
- 3) Insbesondere mit den Szenarien (Tutorials) in der Online-Hilfe finden Sie einen schnellen Einstieg. Auch die Szenarien können Sie direkt über das Menü "Hilfe" erreichen.

Gleitende Häufigkeit von <AA_Cry-startbeispiel.txt>

Verschiedene Zeichen pro 64 Byte Block

AA_Cry-ZIP-AA_Cry-startbeispiel.hex

```
00000 43 54 5A 6D 52 41 6E DB 40 0C BC 07 C8 1F 08 9D 62 CTzMRaNU@.M.E..b
00011 A0 56 91 F6 05 8D 9D A2 41 13 E4 60 23 BD E4 B2 92 V.ö...ca.a.#Mä.
00022 28 8B D0 9A 5B 2C A9 24 F5 5B FB 8E 9C 7C 28 77 A5 (.D[.0s[u..|(v#
00033 BA 4A 91 DB 82 9C 19 0E 87 BB 8A BF 7E 6E 43 F0 E7 3.J.U...>^mCSp
00044 67 E7 67 8B 42 01 12 05 24 4A D8 E2 8A 3E 91 3D qcgkB...$F0a.6N.>
00055 C0 9E 14 1A 8C B0 21 84 9B 2E 22 60 14 45 86 4D DD A...E.MY
00066 45 52 C5 0C 58 4D 4A B0 77 75 67 CD FE 95 19 B9 4C ERÄ.XMJ^wugIp..l
00077 C2 97 8B 4C 1D 2B D0 D8 18 84 75 92 86 43 79 55 C2 Ä.L.+D0..u.CyUA
00088 B1 32 ED CE 09 DC 21 1F A1 78 0C B8 FC 78 CD 2A 75 t2iE.U.l.xä,uxIwu
00099 E7 8F 22 E8 19 1E E1 BB 77 22 64 A5 62 72 B3 87 95 c."e..äw"d#br^..
000AA 43 71 71 69 F0 D6 99 29 86 27 B3 75 E2 E4 C1 9F 16 Cqql80.)^uää..
000BB E0 B6 46 85 D9 EE 6F 9B 53 79 AA FB 69 A2 59 BA 7B *MF.Uio.SyÄioY9(
000CC DD 59 A1 EB 91 12 E0 29 F0 BF 35 2A 42 45 CD A8 8D Yfie..ä)S$#BEI
000DD BA 88 92 30 40 9C 93 F8 41 DC 84 67 81 7B F6 96 D5 8^0e..eAU.g.(o.Ö
000EE F2 1B F9 16 E1 30 9C D8 25 3C 84 11 39 E3 BA 41 FE ö.ü.ä0.0%<.98Ap
000FF 66 90 E2 70 DE 23 3C 5B 16 AC C9 83 55 BF 0E DC 2B f..äpb#*[-.E.U.Ü.+
00110 85 04 C8 0E 6F 89 7B B1 B0 23 E6 7E 99 4F 34 57 3C .E.o.{t^#~.04W<
00121 B5 E2 E0 7F 39 8E A6 1E 67 F8 02 20 24 4F B6 45 0A µ^ä.9.|.gö.S0ME.
00132 30 71 A6 7D DE EC 41 7B B8 31 C0 8B 75 18 AE 70 17 0q}}piÄ{.lÄ.u,@p.
```

Gleitende Häufigkeit von <AA_Cry-ZIP-AA_Cry-startbeispiel.hex>

Verschiedene Zeichen pro 64 Byte Block

AA_Cry-Rijndael-startbeispiel-de.hex

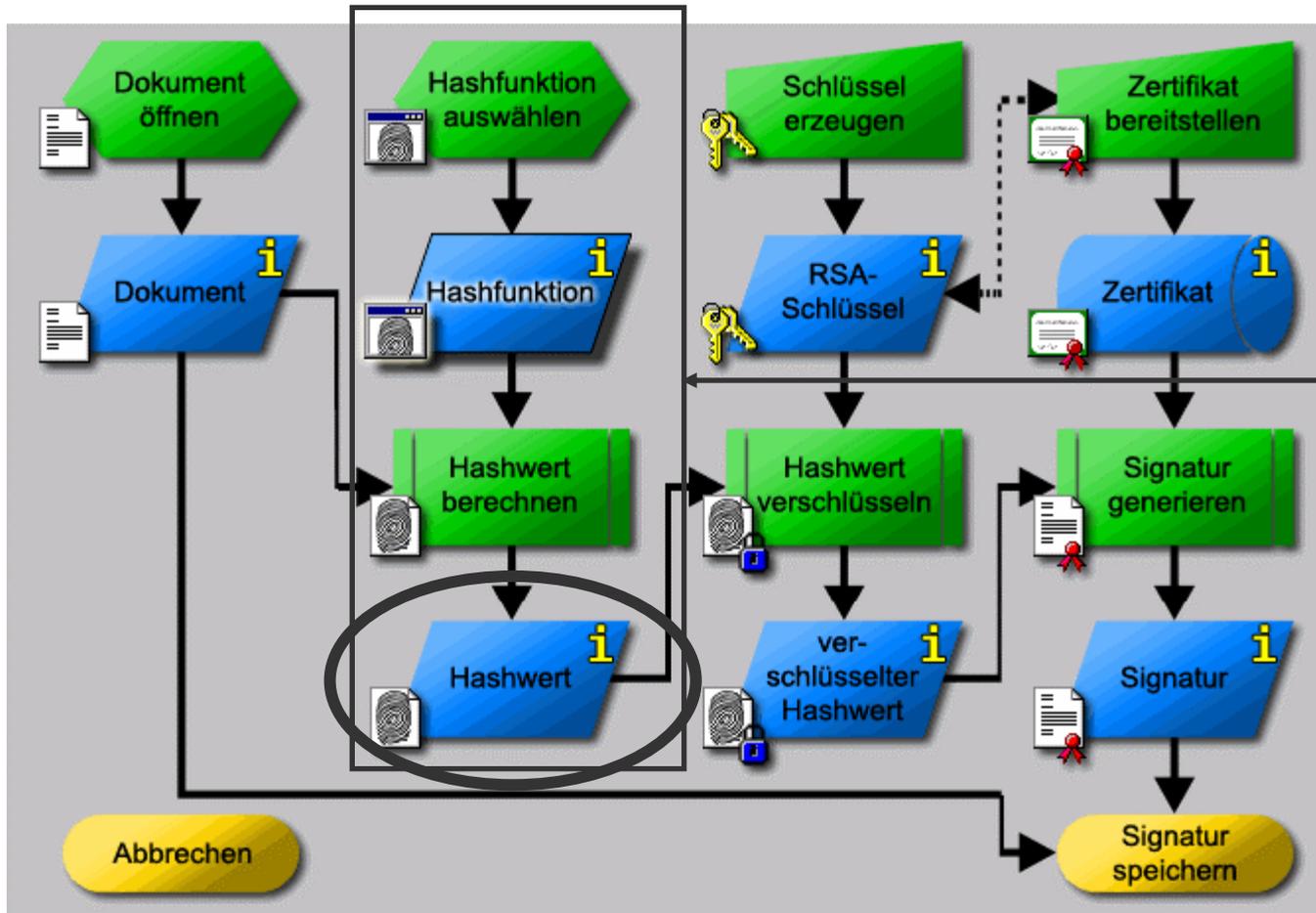
```
00099 3D 00 ED AB 51 FE 18 E3 FC 1A D7 E5 99 0B B3 C8 83 =.i<Qp.äü.xä..ä.E
000AA 6A 30 E7 8D 6A 1C 57 B6 BB 27 24 08 02 CF AB 3B 6D j0c.j.WB>$.~.n
000BB AA 1C C0 D2 91 6F 5D DF 4C 1C F3 B9 5D 4D ED DC 85 3.AO.o]BL.ö|MiU.
000CC 10 C3 E1 A2 B1 50 AC B6 89 AD 63 93 72 79 6A 86 2E .Äc+P^#i.c.rv].
000DD 88 C6 A5 AA 1A 71 79 15 EA F9 67 AF C6 F5 0B 5C EE .Ä#3.qy.ëug.Äö.Ni
000EE F6 10 67 C8 F1 8E 5A 43 07 26 93 6B A3 10 D5 35 D0 ö.gEByZ.&.E.050
000FF D8 1E AD EB B7 E0 EB 07 96 26 8B 47 CA 00 8E 90 08w.äe...&.GE.
00110 CE C0 F3 BD 65 88 31 33 F9 94 89 D7 A6 C4 CB 9B 48 IÄche.13ü.X|ÄE.H
00121 52 52 8D FC E6 F3 17 59 A2 D6 6F 18 66 16 06 12 0F RR.uwö.YeOo.f..
00132 9D 04 C8 F8 2B C4 7B A3 2A 21 95 9F 0B C8 9D 2D 0E .Ee+A{e*!..E.D-
00143 F2 66 90 6C 62 7B DE CB 6C 2F 8E BE 83 E6 A7 2E 02 öf.lb{E1|k.ä.w.
00154 9C 9A 60 EB 66 B2 A6 CA 78 5E 94 89 00 17 E1 41 45 .eif^|Ex...äAE
00165 87 D8 D5 B6 65 4A 42 EC EB BE 64 B6 A5 08 77 36 90 .00eJBlëdW#w6
00176 64 F5 C8 EB F8 68 AF 46 CE AC CD 65 47 CA 00 8E 90 dS&äh.FI-InoeF1>
00187 74 98 0E C2 BB 2F 9B EF 52 9A 99 CA F8 46 B5 0E 22 t..Äw^iR..EeF0.
00198 1E 0B 73 B5 88 73 D5 3C 98 F2 6C 4E 50 FA 4E 34 0A ..su.sÖ.ö1NPaN4
001A9 90 6A 28 24 9E F5 8C 4E C8 C4 63 23 7F 80 E3 E5 C0 j{$.ö.Leäc#.ääÄ
001BA 60 E2 8E 21 AF 5C 2A FA DF D8 3B 62 91 58 9A DB E7 .än|*wäBö;b.X.Üc
001CB 08 6E E2 EB FD 73 08 39 89 7A 6F B3 C9 49 33 B8 20 .näeys.9.zo^E13.
001DC 06 FE EF 9C 8A 12 B3 FD 64 0D 34 EE 9D 4B 0E 17 84 np...äyd.4i.K...
001ED 51 29 67 D1 6C AA 04 B1 68 79 67 AB 65 1F 83 B9 13 öv.Niä.+hru.e
```

Gleitende Häufigkeit von <AA_Cry-Rijndael-startbeispiel-de.hex>

Verschiedene Zeichen pro 64 Byte Block

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
- Modifiziere 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)

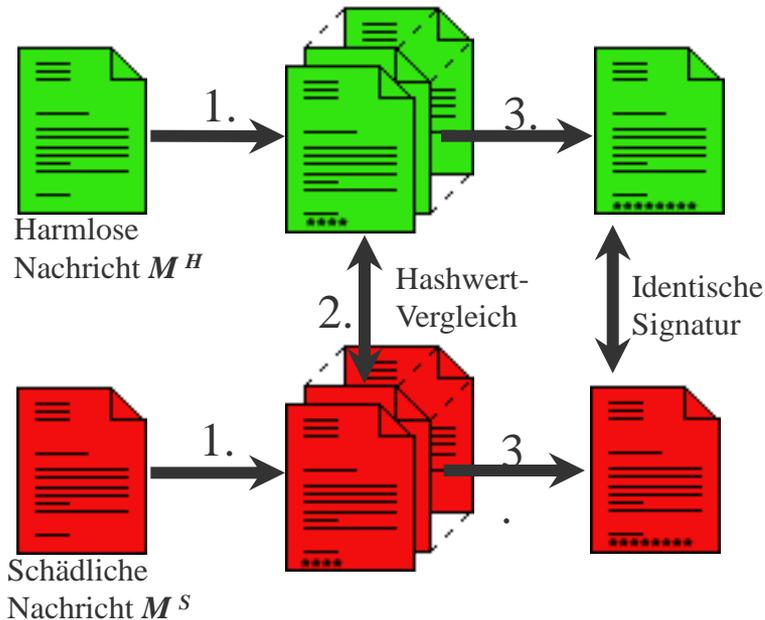
Konzepte:

- 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 :

$$N \approx 2^n$$

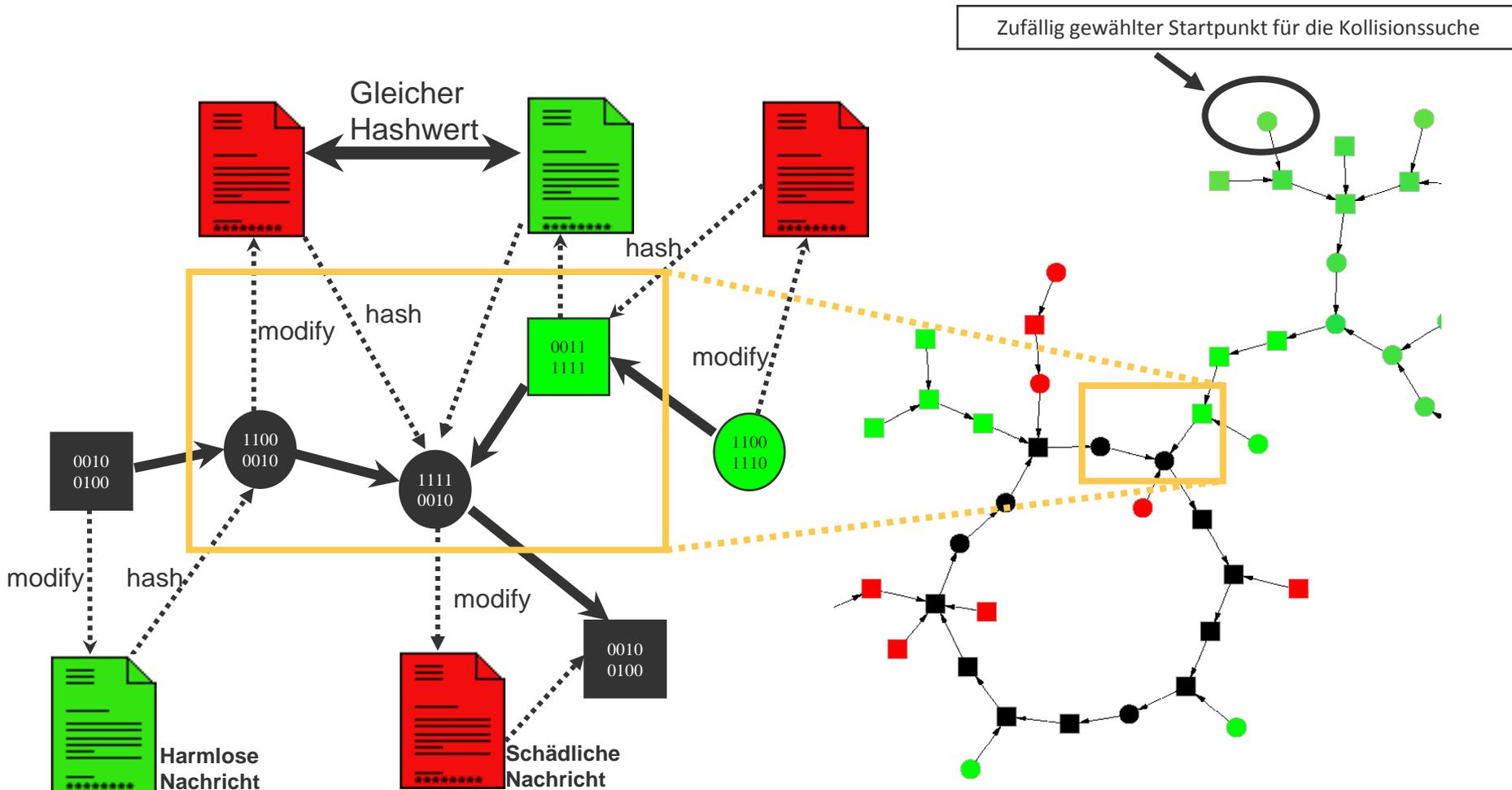
$$N \approx 2^{n/2}$$



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

Hashkollisionssuche (1)

Mapping durch Textmodifikation

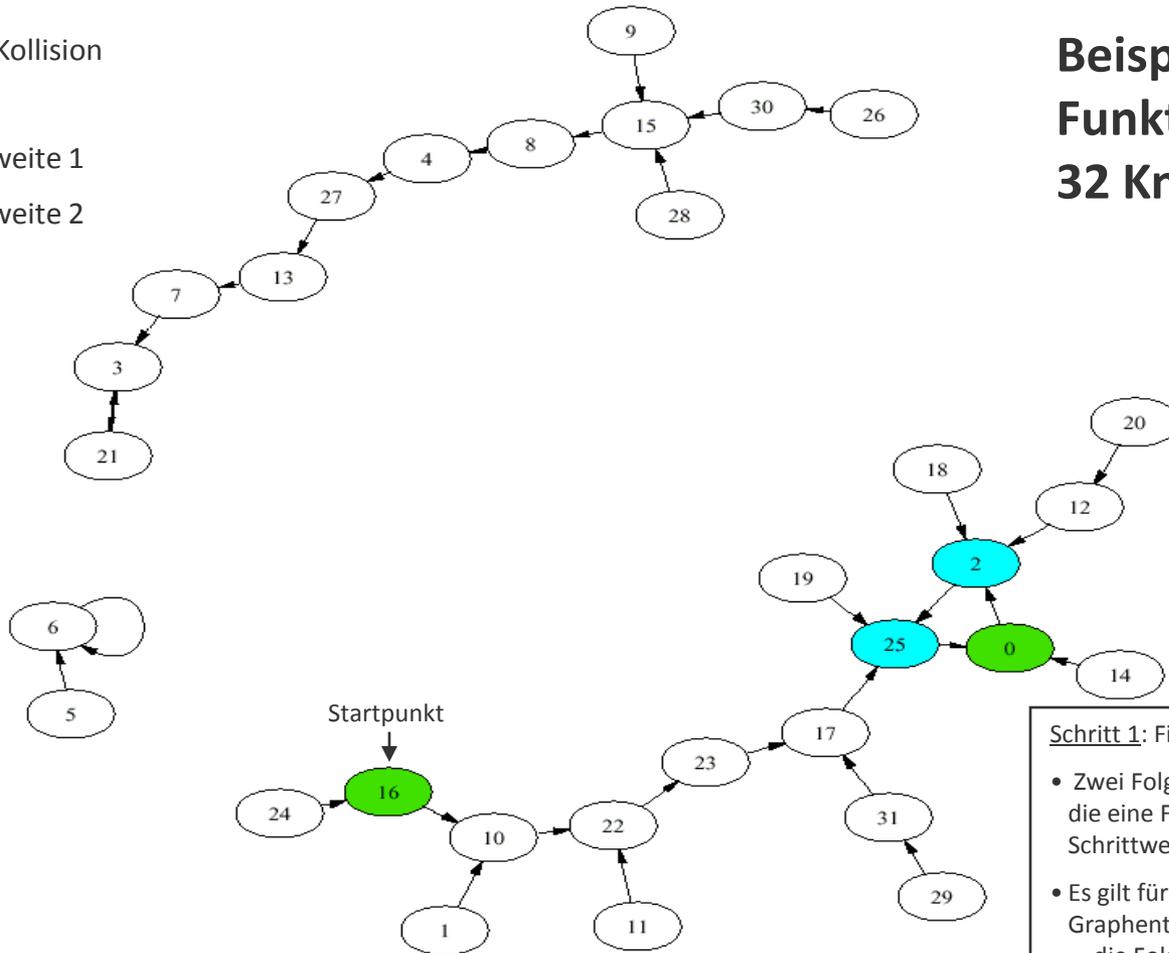


- grün / rot: Pfade aus einem Baum in den Zyklus, die zu einer für den Angreifer nützlichen (grün) / nicht nutzbaren Kollision (rot) führen.
- quadratisch / rund: Hashwert hat gerade / ungerade Parität
- schwarz: alle Knoten im Zyklus

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:

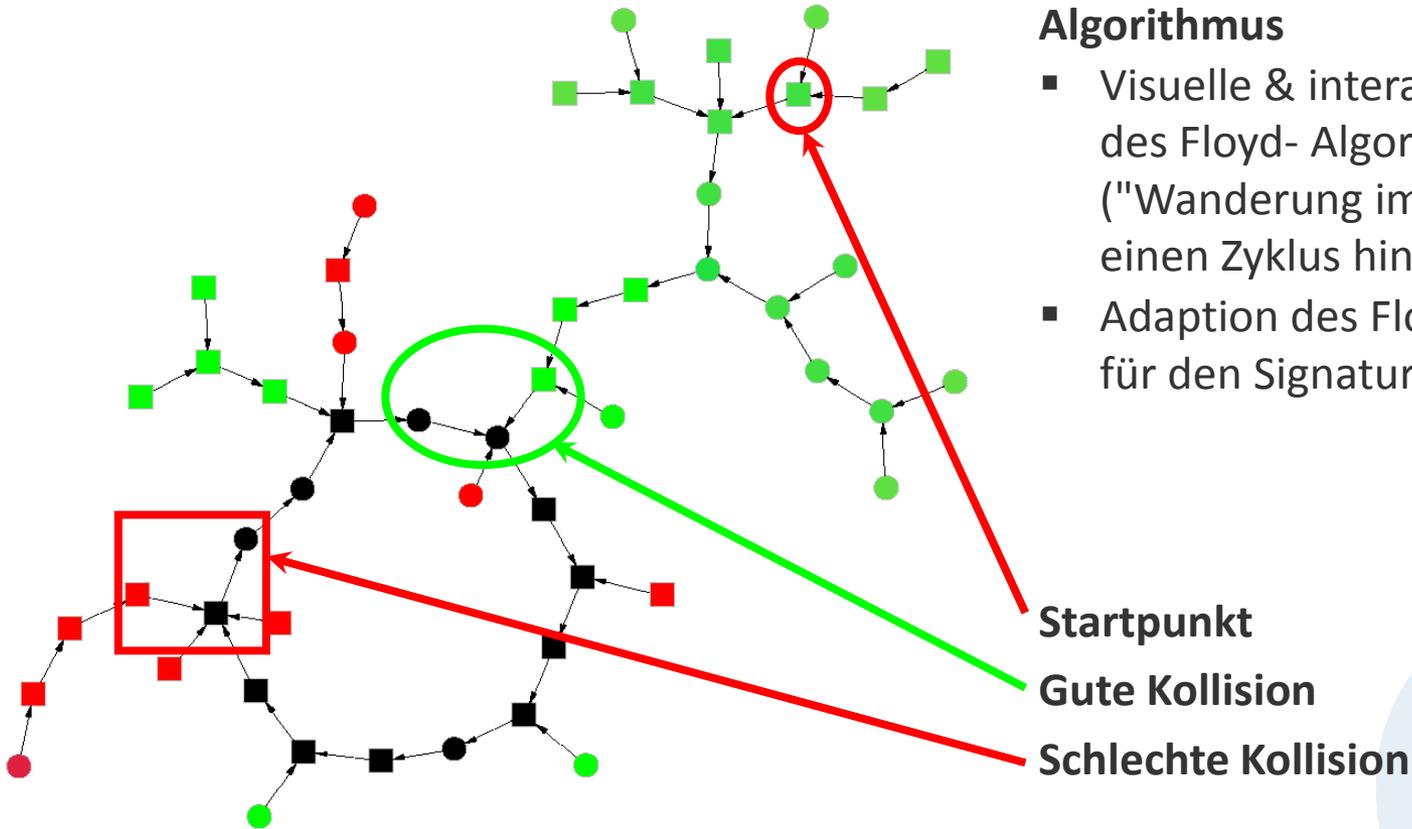
- Zwei Folgen mit gleichem Start [16]:
die eine Folge hat Schrittweite 1, die andere Schrittweite 2.
- Es gilt für alle Zykluslängen (aufgrund der Graphentheorie):
 - die Folgen enden immer in einem Zyklus.
 - beide Folgen treffen sich in einem Knoten im Zyklus (hier 0).

Hashkollisionssuche (4)

Geburtstagsangriff auf die digitale Signatur

Auseinandersetzung mit dem Floyd-Algorithmus

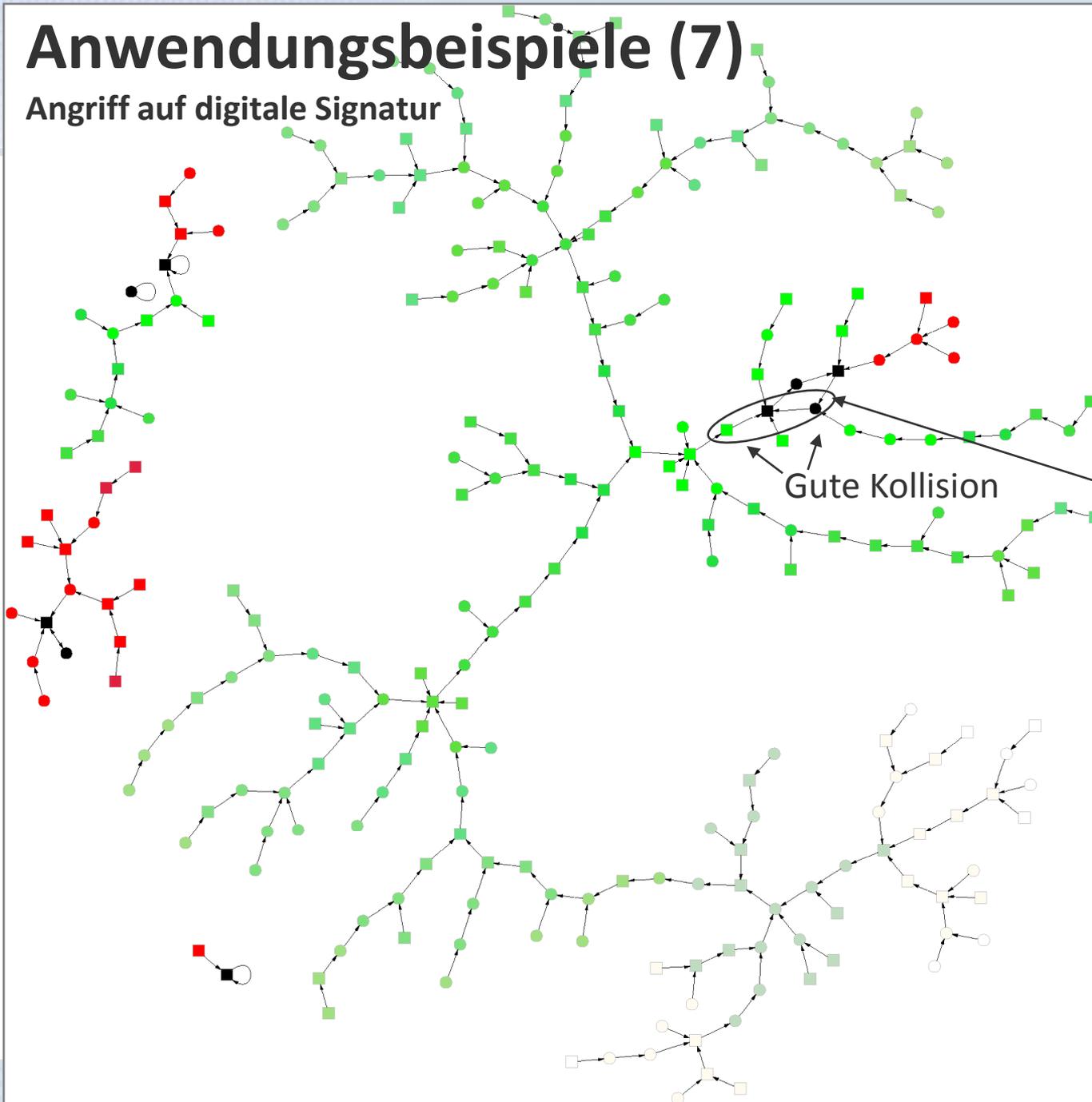
- Visuelle & interaktive Darstellung des Floyd- Algorithmus ("Wanderung im Mapping" in einen Zyklus hinein).*
- Adaption des Floyd- Algorithmus für den Signaturangriff.



*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

1. "Harmlose" Datei auswählen
Die "harmlose" Datei ist eine Nachricht, von der der Angreifer vermutet, dass der Unterzeichner sie digital signieren wird.
C:\program files\CrypTool-pre1-de\examples\Original.txt

2. "Gefährliche" Datei auswählen
Die "gefährliche" Datei ist eine Nachricht, von der der Angreifer nach erfolgreichem Angriff behaupten wird: "Diese Nachricht wurde vom Unterzeichner digital signiert."
C:\program files\CrypTool-pre1-de\examples\Faelschung.txt

4. Nachrichtenpaar suchen

3. Optionen festlegen

Optionen für den Angriff auf den Hashwert der digitalen ...

Hashfunktion
Wählen Sie eines der sechs Hashverfahren sowie die Anzahl der Bits, die für den Vergleich der Hashwerte herangezogen werden sollen.

MD2 MD4 MD5
 SHA SHA-1 RIPEMD-160

Signifikante Bitlänge: 40 (Wertebereich: 1 - 128)

Optionen für die Nachrichtenmodifikation
Entscheiden Sie, nach welchem Verfahren die Nachrichten modifiziert werden sollen.

Leerzeichen einfügen Verschieben
 Zeichen anhängen Löschen
 Nicht modifizieren Nicht anhängen

Ein Nachrichtenpaar wird gesucht ...

Run 1
Zyklussuche (40 Bit)
Fortschritt: 10% Restzeit: 00:00:06

Ein Nachrichtenpaar wird gesucht ...

Run 1
Kollisionssuche (40 Bit)
Fortschritt: 18% Restzeit: 00:00:22

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

Anwendungsbeispiele (7)

Angriff auf digitale Signatur: Ergebnisse

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

Sehr geehrter Herr Einkaufsgern,
bitte bestellen Sie eine Schreibmaschine
MfG.
Peter Gutermann
AADBADCBACBACDADCB

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

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

Sehr geehrter Herr Einkaufsgern,
bitte bestellen Sie für Herrn Dieter Dieb ein Porsche und eine Tankkarte.
MfG.
Peter Gutermann
AAAABDDDDDBBAAABBC

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

Die ersten 32 Bit des Hashwertes sind gleich.

Praktische Resultate

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

Zusätzlich zur interaktiven Bedienung:

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

Anwendungsbeispiele (8)

Authentifizierung in einer Client-Server-Umgebung

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

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

Anwendungsbeispiele (9)

Demonstration eines Seitenkanalangriffes (auf ein Hybridverschlüsselungsprotokoll)

Seitenkanalangriff auf das Hybridverschlüsselungsprotokoll (Textbook-RSA)

Angriff Schritt für Schritt

- Einleitung in das Szenario
- Vorbereitungen durchführen
- Nachricht übertragen
- Nachricht entschlüsseln
- Nachricht abfangen
- Angriffszyklus starten
- Zusammenfassung erstellen

Beenden

Informationsdialoge anzeigen

Alice [Client]

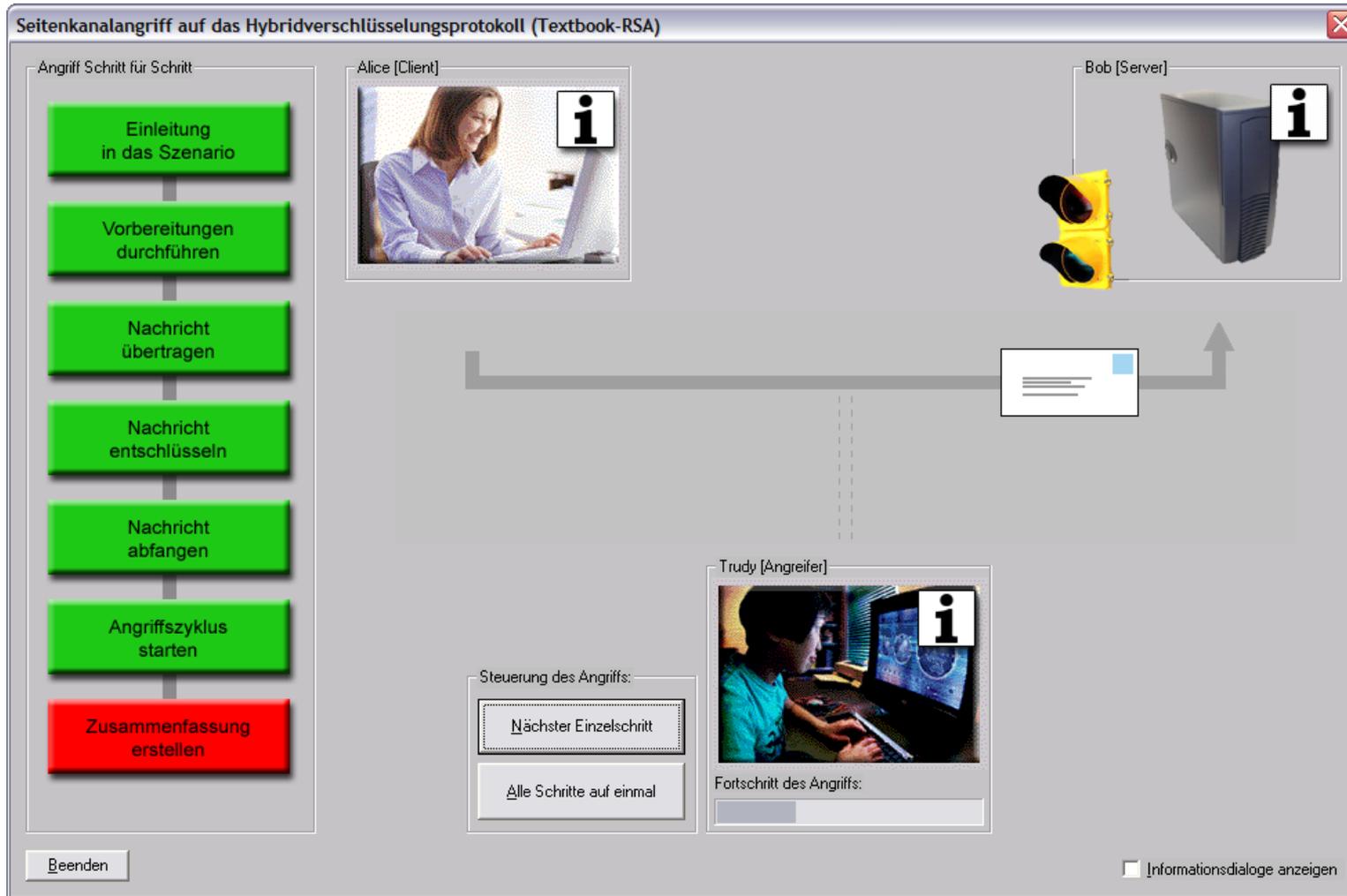
Bob [Server]

Trudy [Angreifer]

Steuerung des Angriffs:

- Nächster Einzelschritt
- Alle Schritte auf einmal

Fortschritt des Angriffs:



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

Anwendungsbeispiele (9)

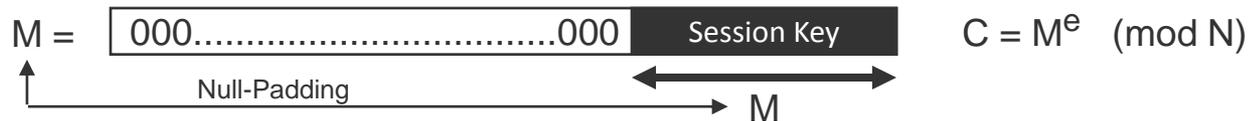
Idee zu diesem Seitenkanalangriff

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

Voraussetzungen:

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

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



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 δ ein. Alternativ können Sie d direkt eingeben, woraus δ berechnet wird.

Um sich ein Beispiel erzeugen zu lassen, geben Sie δ 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 : δ :

N :

e :

d :

Schritt 2: Angriffsparameter für das Gitterreduktionsverfahren eingeben

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

t : Wird abhängig von m optimal bestimmt.

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

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

Schritt 3: Angriff starten

Erzeuge Gitter:

Reduziere Gitter: Reduktionen:

Bilde Resultante: Resultanten:

Gesamtzeit:

Gefundene Faktorisierung:

p : q :

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

- **Drei Varianten**, die nicht standhalten:

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

- Diese Annahmen sind realistisch.

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

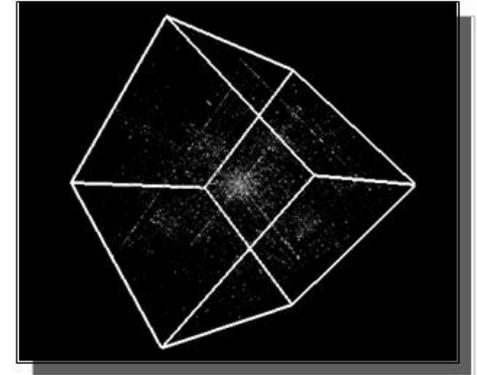
Anwendungsbeispiele (11)

Zufallsanalyse mit 3-D-Visualisierung

3-D Visualisierung zur Analyse von Zufallszahlen

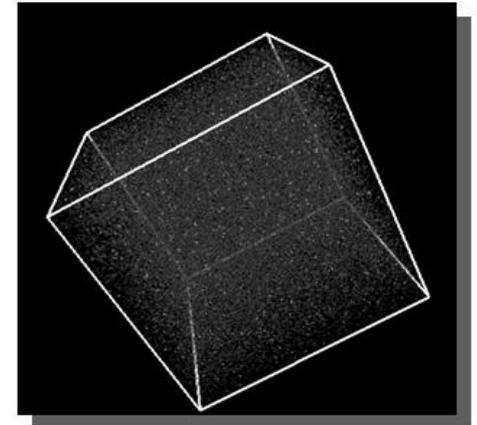
Beispiel 1

- 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 über das Menü: „Analyse“ \ „Zufallsanalyse“ \ „3-D-Visualisierung“
- Ergebnis: **Strukturen sind offensichtlich erkennbar**



Beispiel 2

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

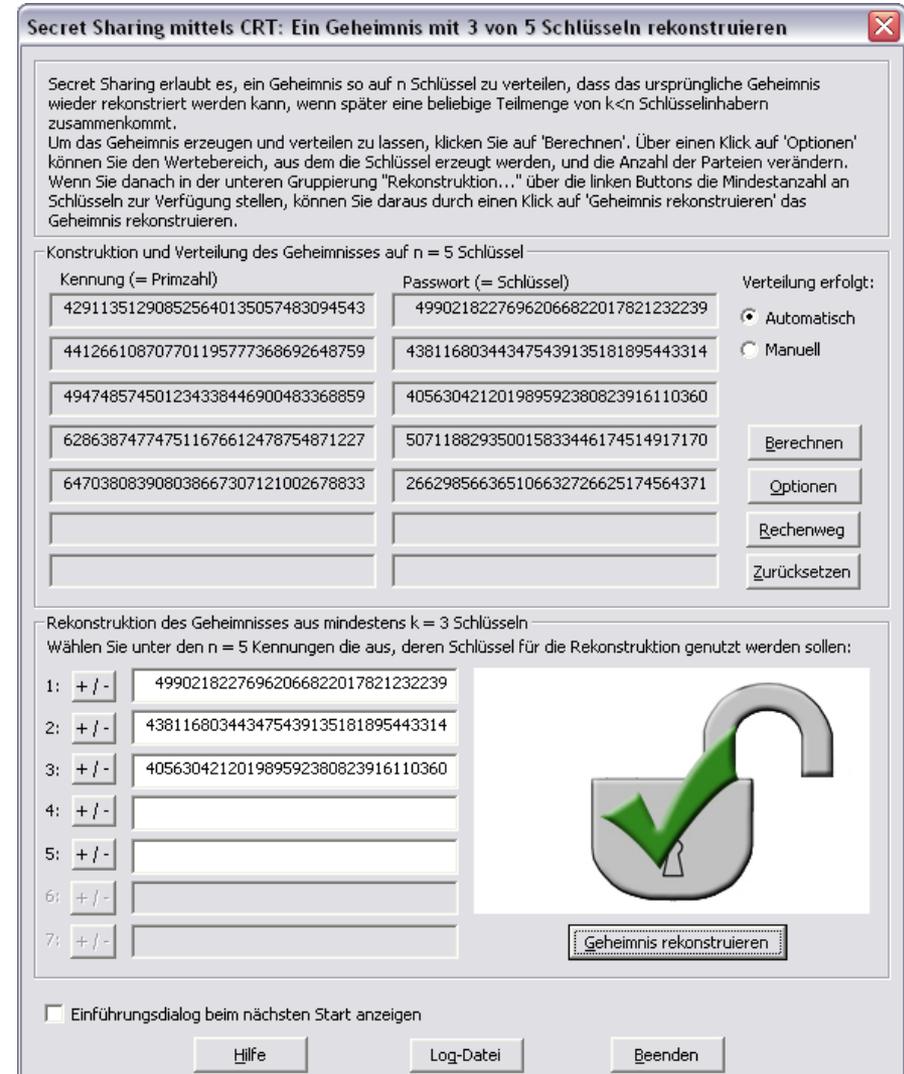
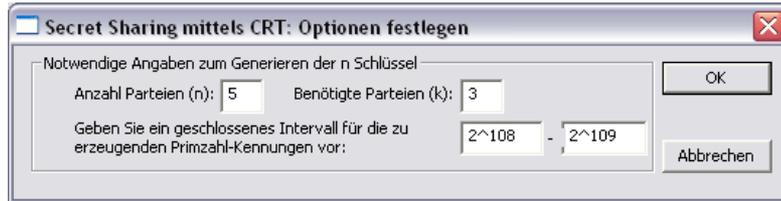


Anwendungsbeispiele (12)

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
- **Menü:** „Einzelverfahren“ \ „Anwendungen des Chinesischen Restsatzverfahrens“ \ „Secret Sharing mittels CRT“
- **„Optionen“** ermöglicht weitere Details des Verfahrens einzustellen.
- **„Rechenweg“** zeigt die Schritte zur Generierung der Schlüssel.



Anwendungsbeispiele (12)

Secret Sharing mittels Schwellenwertschema von Shamir

Secret Sharing Beispiel (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

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

1. Angabe des Geheimnisses K , sowie Anzahl der Teilnehmer n und Schwellenwert t
2. Polynom generieren
3. Parameter übernehmen

- #### ■ Mittels „Rekonstruktion“ kann das eingegebene Geheimnis wiederhergestellt werden

Secret Sharing: Aufsetzen eines Schwellenwertschemas

Mit dem (t, n) -Schwellenwertschema nach Shamir ist es möglich, ein Geheimnis 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 $S \geq 0$

Anzahl der Teilnehmer n mit $n > 0$

Schwellenwert (Mindestanzahl) t mit $t > 0$

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

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

Polynom $f(x)$

Primzahl p

Aus den Parametern berechnete Werte der Teilnehmer:

	Teilnehmer	Öffentlicher Wert x	Share (geheimer Wert $f(x)$)
<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 Checkboxes ankreuzen.

Informationsdialog zu Beginn anzeigen

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 $x = a[i] \bmod m[i]$ ($i=1, \dots, 9$) eingeben und anschließend lösen. Solche Gleichungssysteme kann man z.B. nutzen, um herauszufinden, in wie viel Tagen bestimmte Planeten aufgereiht wie auf einer Perlschnur 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	

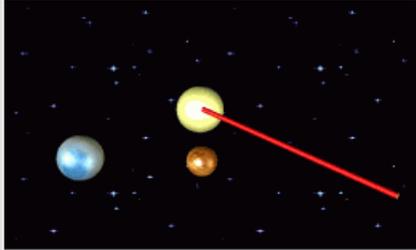
Lösung

126.228.390.655

Lösen Beenden

Löschen aller Parameter Zurücksetzen auf den Initialzustand

Anwendungsbeispiel aus der Astronomie / Film-Visualisierung ist fix



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

15 und 100 Tage.

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

Planetenauswahl

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

In welchen Zeitabständen (Tagen) wiederholt sich das Ereignis?

8.359.702.902.760

Anwendungsbeispiele (14)

Visualisierung von symmetrischen Verschlüsselungsverfahren mit ANIMAL (1)

Animierte Darstellung verschiedener symmetrischer Verfahren

- Caesar
- Vigenère
- Nihilist
- DES

CrypTool

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

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

Animationsgeschwindigkeit Skalierung der Darstellung

Animal Animation: Caesar-Verschlüsselung

Speed 0 200 400 600 800 1000

Zoom 0 100 200 300 400 500

Caesar-Verschlüsselung

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

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

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

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

J D O Geheimtext

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

Navigation Kioskmodus Manuelle Schrittkontrolle

21 / 88 0 20 40 60 80 100

Direkte Anwahl eines Animationsschrittes

Anwendungsbeispiele (14)

Visualisierung der symmetrischen Verschlüsselungsverfahren mit ANIMAL (2)

Visualisierung der DES-Verschlüsselung

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

Zoom: 0 100 200 300 400 500

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

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

PC2: $\begin{bmatrix} 14 & 17 & 11 & 24 & 1 & 5 \\ 3 & 28 & 15 & 6 & 21 & 10 \\ 23 & 19 & 12 & 4 & 26 & 8 \\ 16 & 7 & 27 & 20 & 13 & 2 \\ 41 & 52 & 31 & 37 & 47 & 55 \\ 30 & 40 & 51 & 45 & 33 & 48 \\ 44 & 49 & 39 & 56 & 34 & 53 \\ 46 & 42 & 50 & 36 & 29 & 32 \end{bmatrix}$

K[1]: $\begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$

Schema:

```

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

16 Teilschlüssel

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)

Zoom: 0 100 200 300 400 500

Die Funktion f :

$110110 \ 001010 \ 110110 \ 010100 \ 000100 \ 100110 \ 101001 \ 010011$

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

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

S-Box 1:

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	9	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

Schema:

```

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

16 Teilschlüssel

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)

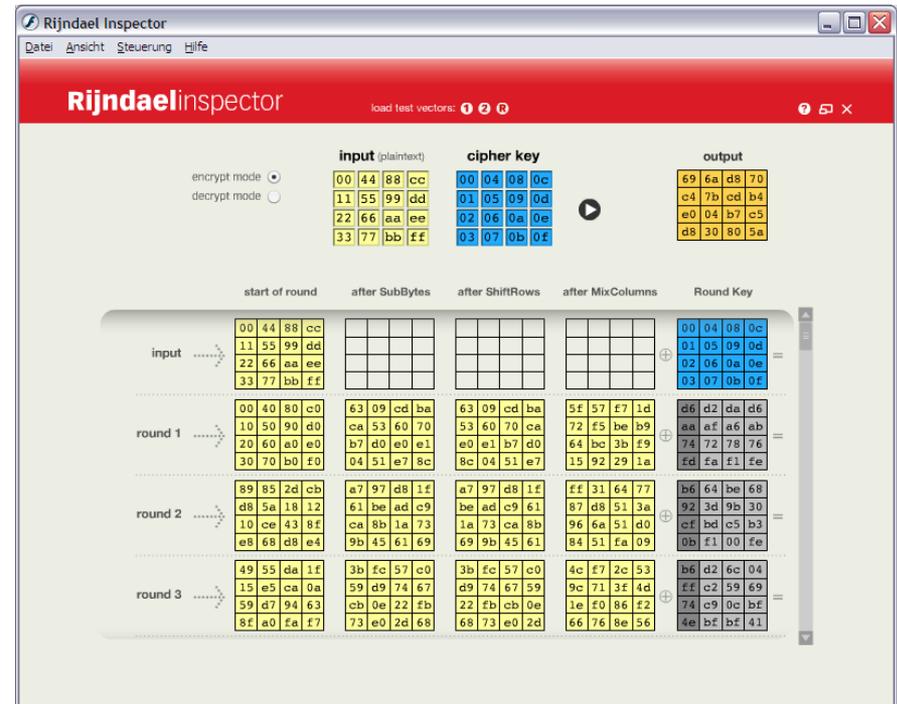
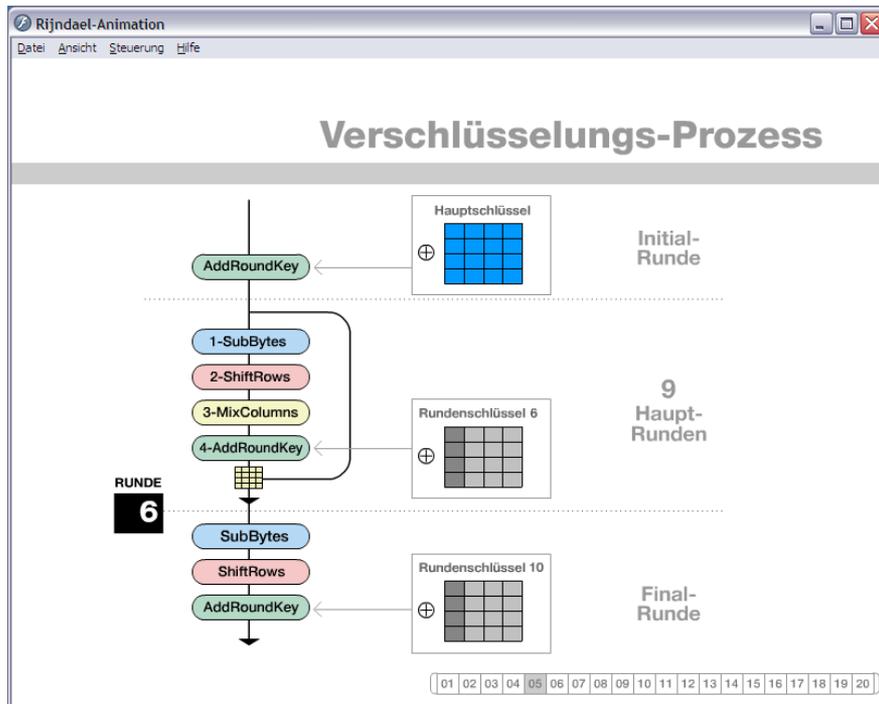
Visualisierung von AES (Rijndael-Chiffre)

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

- Visualisierung durch Animation des rundenbasierten Verschlüsselungsprozesses

Rijndael-Inspector

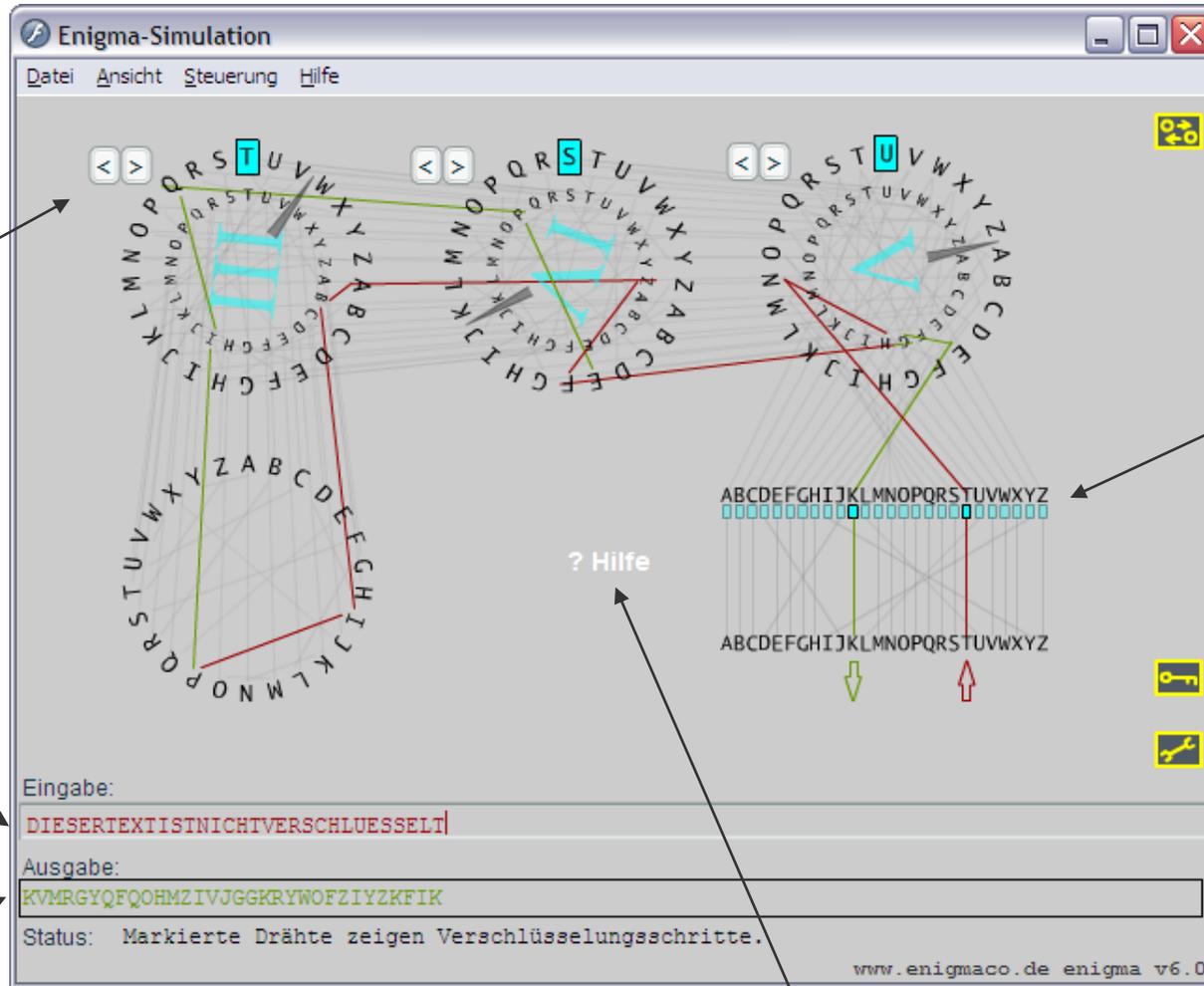
- Verschlüsselungsprozess zum Ausprobieren (mit selbst wählbaren Daten)



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

Anwendungsbeispiele (16)

Visualisierung der Enigma-Verschlüsselung



Rotorstellung
verändern

Eingabe des
zu Klartext

Ausgabe des
verschlüsselten
Textes

Auswahl der
Rotoren

Setzen der
Stecker

Anzeige der
Einstellungen

Zufälliges
Setzen und
Zurücksetzen
der Enigma

? Hilfe

Klicken für weitere HTML-Hilfe

Anwendungsbeispiele (17)

Erzeugung eines Message Authentication Code (MAC)

Message Authentication Code (MAC)

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

Berechnung eines MAC in CrypTool

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

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

1. Auswahl der Hashfunktion

2. Auswahl der MAC-Variante

3. Angabe eines Schlüssels

4. Erzeugung des MAC

Message Authentication Code

Beschreibung

Mit Hilfe eines MACs kann der Empfänger einer Nachricht ihre Integrität und die Authentizität ihres Ursprungs (Senders) überprüfen. Hierzu verwenden beide Parteien ein gemeinsames Geheimnis (symmetrischer Schlüssel).

Um den MAC einer Nachricht zu erzeugen, wird eine kryptographische Hashfunktion auf eine Verknüpfung aus der Nachricht m und dem geheimen Schlüssel k angewendet. Je nach MAC-Variante können auch zwei geheime Schlüssel k und k' verwendet werden.

Nachricht

CrypTool ist ein Programm, mit dessen Hilfe Sie kryptografische Verfahren anwenden und an...

Hashfunktion wählen

- MD2
- MD4
- MD5
- SHA
- SHA-1
- SHA-256
- SHA-512
- RIPEMD-160

MAC-Variante (Position der Schlüssel; Verschachtelung)

- $H(k, m)$: vor der Nachricht
- $H(m, k)$: hinter der Nachricht
- $H(k, m, k)$: vor und hinter der Nachricht
- $H(k, H(k, m))$: doppeltes Hashing
- $H(k, m, k')$: zwei verschiedene Schlüssel

Geben Sie den Schlüssel (k) ein:

Chiffre

Geben Sie den zweiten Schlüssel (k') ein:

Eingabe für äußere Hashfunktion (abhängig von der oben gewählten MAC-Variante):

ChiffreCrypTool ist ein Programm, mit dessen Hilfe Sie kryptografische Verfahren anwenden und analysieren können. So können Sie neue Dokumente erstellen und bestehende Dokumente öffnen und weiter bearbeiten.

Hashwert der Nachricht m:

FB A8 6E 32 B3 1F B9 EE EF A1 D7 86 AD 8B 2C 45

Erzeugter MAC:

EE 10 62 2A 6C A4 A1 2F 50 1A B1 37 F9 73 96 EF

Schließen

Anwendungsbeispiele (18)

Hash-Demo

Sensitivität von Hashfunktionen bei Änderungen des Originaltextes

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

Beispiel:

Die Eingabe eines zusätzlichen Leerzeichens hinter „CrypTool“ in der Originaldatei bewirkt eine 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“

Hash-Demo: SHA-1 (160 Bit)-Hash für Startbeispiel-de.txt

Beschreibung

- Wählen Sie ein Hashverfahren aus und editieren Sie dann unten die Kopie der Originaldatei (Textfeld "Aktuelles Dokument").
- Ganz unten sehen Sie, wie viele Bits sich im Hashwert ändern, wenn Sie das Dokument editieren.

Auswahl der Hashfunktion: SHA-1 (160 Bit)

Darstellung der Hashwerte: hexadezimal dezimal binär

Aktuelles Dokument (Lesen Sie die Kopie der Originaldatei können Sie hier ändern)

CrypTool 1.4.20

CrypTool ist ein umfangreiches Lernprogramm zu den Themen Kryptographie und Kryptoanalyse.

Dies ist eine Textdatei, mit der Sie Ihre ersten Schritte mit CrypTool machen können.

1) Sie können diese Datei z.B. über das Menü "Ver-/Entschlüsseln \ Symmetrisch (klassisch)" mit dem Caesar-Verfahren verschlüsseln.

2) Den besten Überblick über die Möglichkeiten von

Hashwert der Originaldatei
37 2D 92 D1 E4 2F 84 10 16 3E 79 06 FB 38 EE 8A 3E 42

Hashwert der aktuellen Datei
E9 3C CA 60 84 8A 89 4F 9C BF E1 EC 71 A6 EB A7 C9 5A

Unterschied zwischen dem Hashwert der Original- und dem Hashwert der aktuellen Datei

```
11011110#00010001#01011000#10110001#01100000#10100101#
00001101#01011111#10001010#10000001#10011000#11101010#
10001010#10011110#00000101#00101101#11110111#00011000#
11010100#00110100#
45.6% der Bits unterscheiden sich (73 von 160).
Längste unveränderte Bitfolge: Offset 73, Länge 6.
```

Dialog beenden

Anwendungsbeispiele (19)

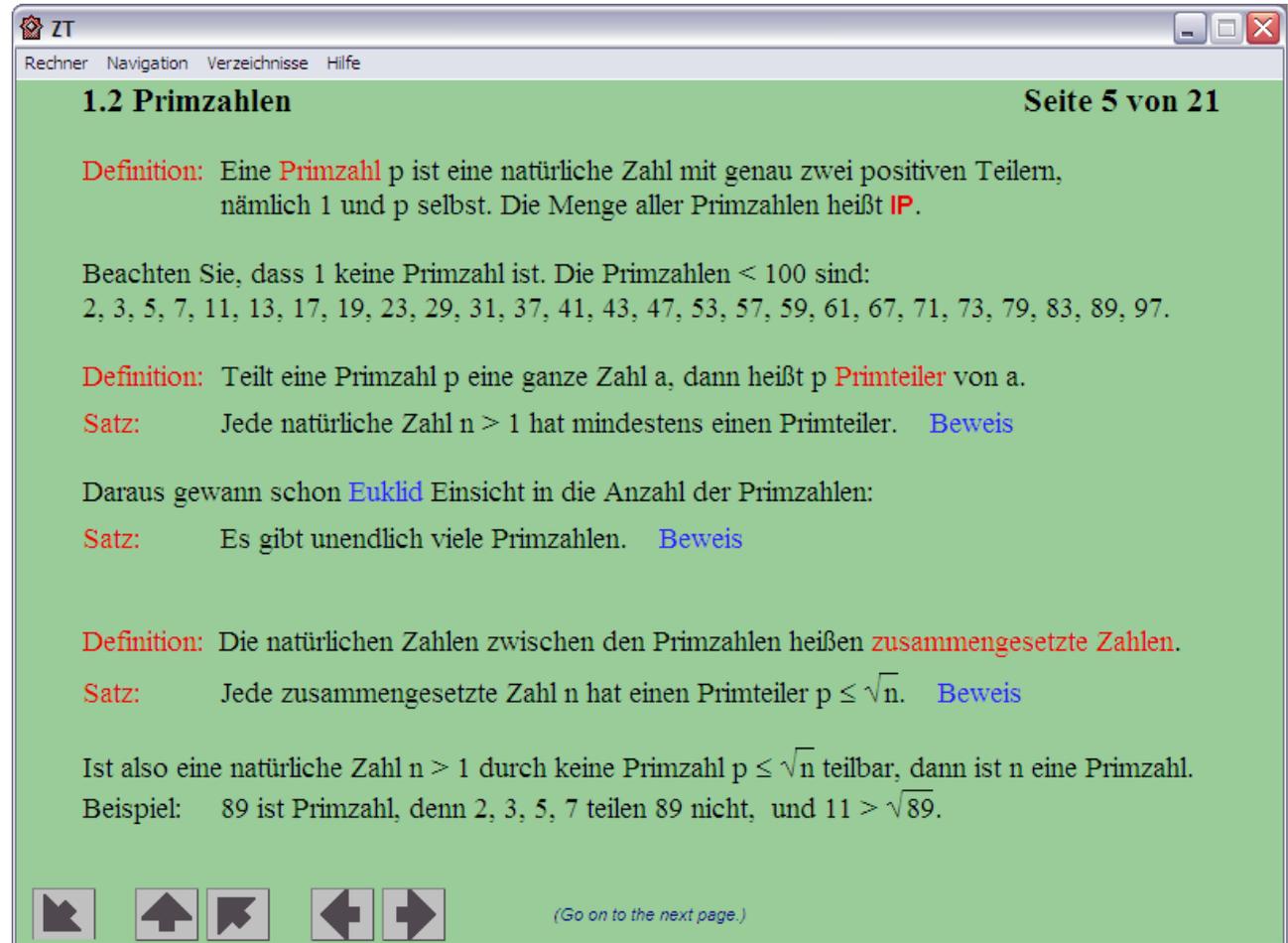
Lernprogramm zur Zahlentheorie und zur asymmetrischen Verschlüsselung

■ 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



Rechner Navigation Verzeichnisse Hilfe

1.2 Primzahlen Seite 5 von 21

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

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

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

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

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

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

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

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

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

 (Go on to the next page.)

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

Anwendungsbeispiele (20)

Punktaddition auf elliptischen Kurven

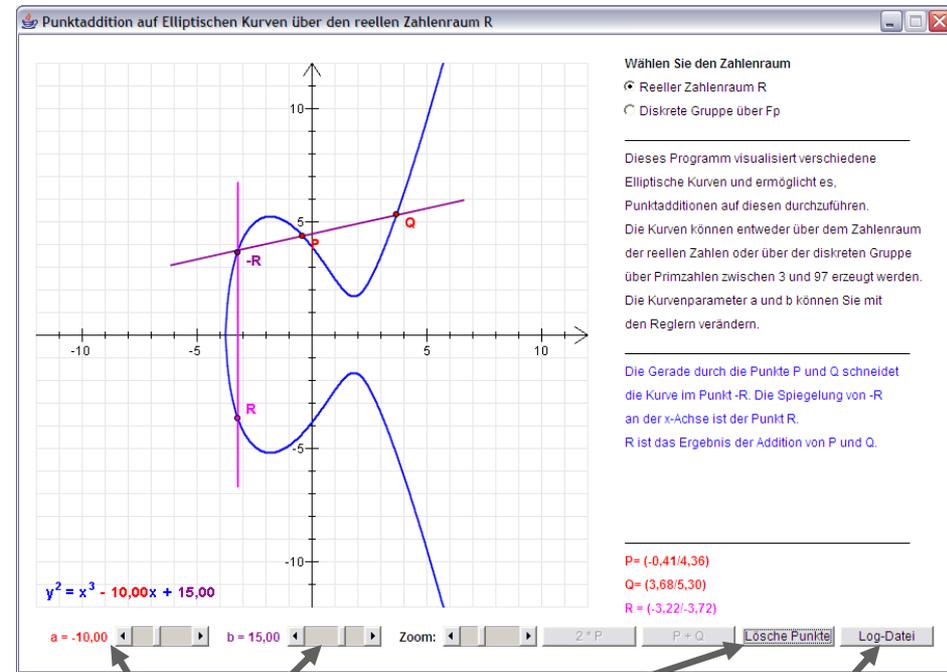
- Visualisierung der Punktaddition auf elliptischen Kurven
- Grundlage der Elliptischen Kurven Kryptographie (ECC)

Beispiel 1

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

Beispiel 2

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



Kurven-Parameter einstellen

Lösche Punkte

Log-Datei für die Berechnungen

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

Anwendungsbeispiele (21)

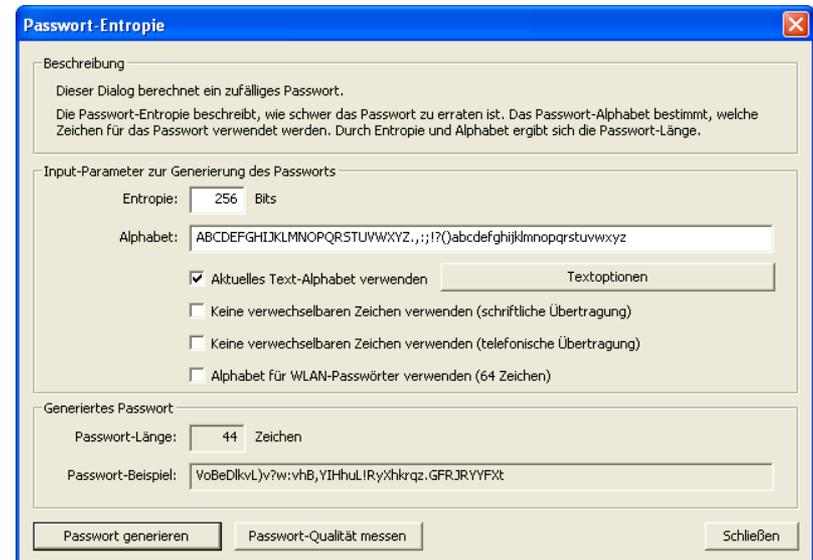
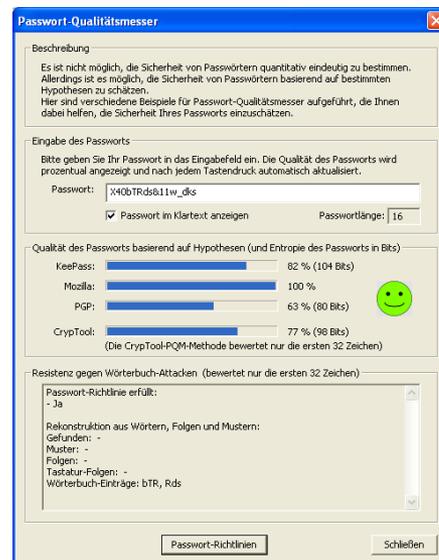
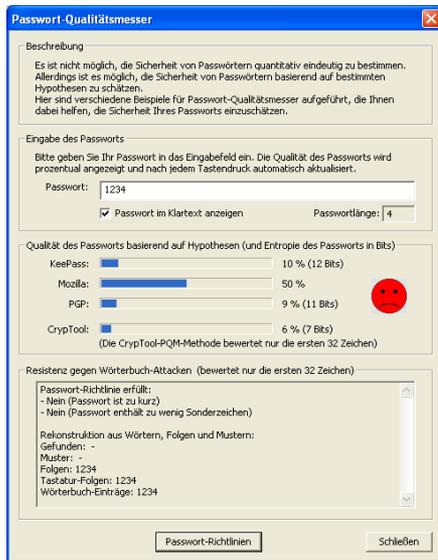
Passwort-Qualitätsmesser (PQM) und Passwort-Entropie (1)

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 (21)

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**.

Qualität eines Passwortes aus Angreiferperspektive

- Angriff auf ein Passwort (sofern beliebig viele Versuche zugelassen sind):
 1. Klassischer **Wörterbuchangriff**
 2. Wörterbuchangriff mit **weiteren Varianten** (z.B. 4-stellige Zahlen: Sommer2007)
 3. **Brute-Force-Angriff** durch Test aller Kombinationen (ggf. mit Einschränkungen auf Zeichenarten)
- ⇒ Ein gutes Passwort sollte so gewählt werden, dass es den Angriffen 1. und 2. standhält, im Hinblick auf 3. zumindest 8 Zeichen lang ist und Zahlen sowie Sonderzeichen beinhaltet.

Anwendungsbeispiele (22)

Brute-Force-Analyse (1)

Brute-Force-Analyse

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

Beispiel: Analyse mit DES (ECB)

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

Schlüssel (Hex)

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

Verschlüsselter Text (Hex)

```
66b9354452d29eb5  
1f0dd05d8ed51583  
bcf9ebd1979ead6a  
8cf42d40e004a1d4  
0ed33fed7f46c585  
d6d8641bc4fb2478  
a2e66d852e175f5c
```

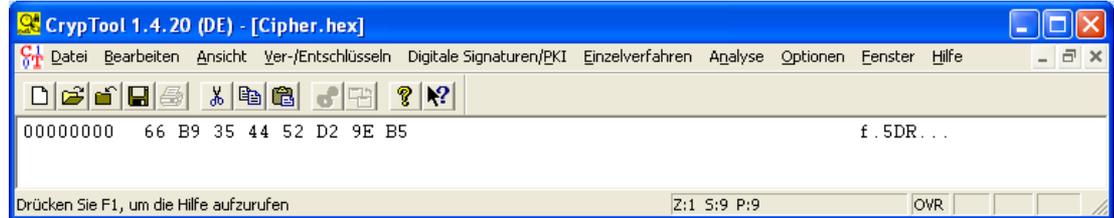


Anwendungsbeispiele (22)

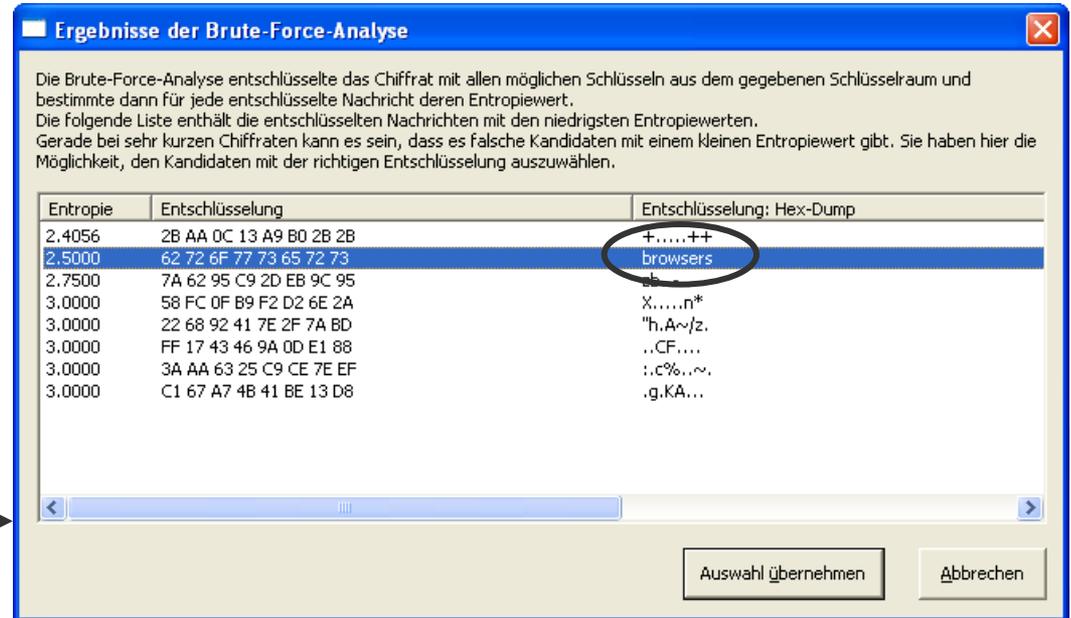
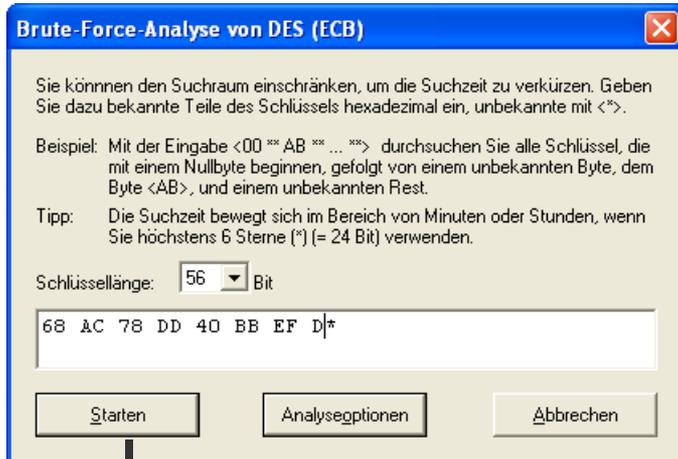
Brute-Force-Analyse (2)

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“

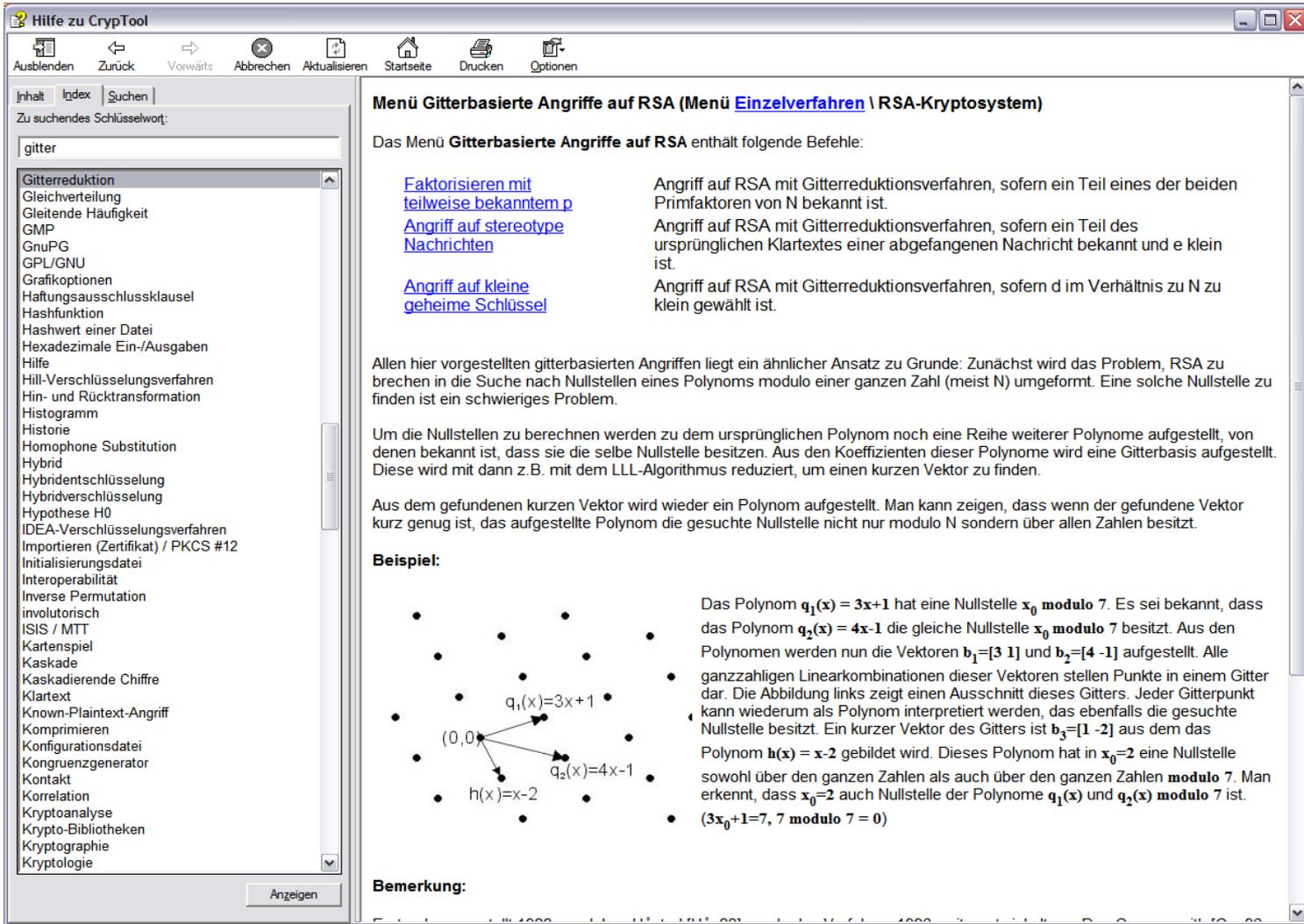


Menü: „Analyse“ \ „Symmetrische Verfahren (modern)“ \ „DES (ECB)“



Anwendungsbeispiele (23)

CrypTool Online-Hilfe (1)



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
- Hybridentschlüsselung
- Hybridverschlüsselung
- Hypothese H0
- IDEA-Verschlüsselungsverfahren
- Importieren (Zertifikat) / PKCS #12
- Initialisierungsdatei
- Interoperabilität
- Inverse Permutation
- involutorisch
- ISIS / MIT
- Kartenspiel
- Kaskade
- Kaskadierende Chiffre
- Klartext
- Known-Plaintext-Angriff
- Komprimieren
- Konfigurationsdatei
- Kongruenzgenerator
- Kontakt
- Korrelation
- Kryptoanalyse
- Krypto-Bibliotheken
- Kryptographie
- Kryptologie

Anzeigen

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

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

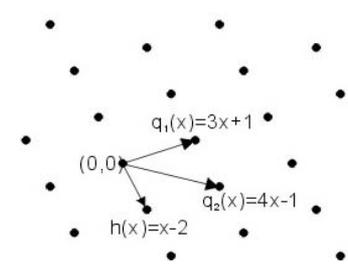
Faktorisieren mit teilweise bekanntem p	Angriff auf RSA mit Gitterreduktionsverfahren, sofern ein Teil eines der beiden Primfaktoren von N bekannt ist.
Angriff auf stereotype Nachrichten	Angriff auf RSA mit Gitterreduktionsverfahren, sofern ein Teil des ursprünglichen Klartextes einer abgefangenen Nachricht bekannt und e klein ist.
Angriff auf kleine geheime Schlüssel	Angriff auf RSA mit Gitterreduktionsverfahren, sofern d im Verhältnis zu N zu klein gewählt ist.

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

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

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

Beispiel:



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

- $(3x_0+1=7, 7 \text{ modulo } 7 = 0)$

Bemerkung:

Menü: „Hilfe“ \ „Startseite“

Anwendungsbeispiele (23)

CrypTool Online-Hilfe (2)

Hilfe zu CrypTool 1.4.20 (DE)

Vergleich von Base64- und UU-Codierung

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

Base64 **UUEncode**

Aufteilen von 3 x 8 Bit in 4 x 6 Bit

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

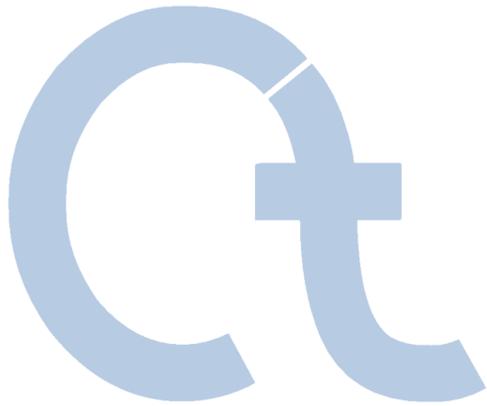
2. Schritt: Darstellung der 6-bit Werte - unterschiedliche Methoden.

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

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

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

Vorteile	Nachteile
<ul style="list-style-type: none">• 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). <u>Nur UUencode:</u>• Keine EBCDIC Unterstützung.• Keine definierten Standards.



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

Weiterentwicklung (1)

CT = CrypTool
CT2 = CrypTool 2.0
JCT = JCrypTool

Geplant nach 1.4.21 (vgl. Readme-Datei)

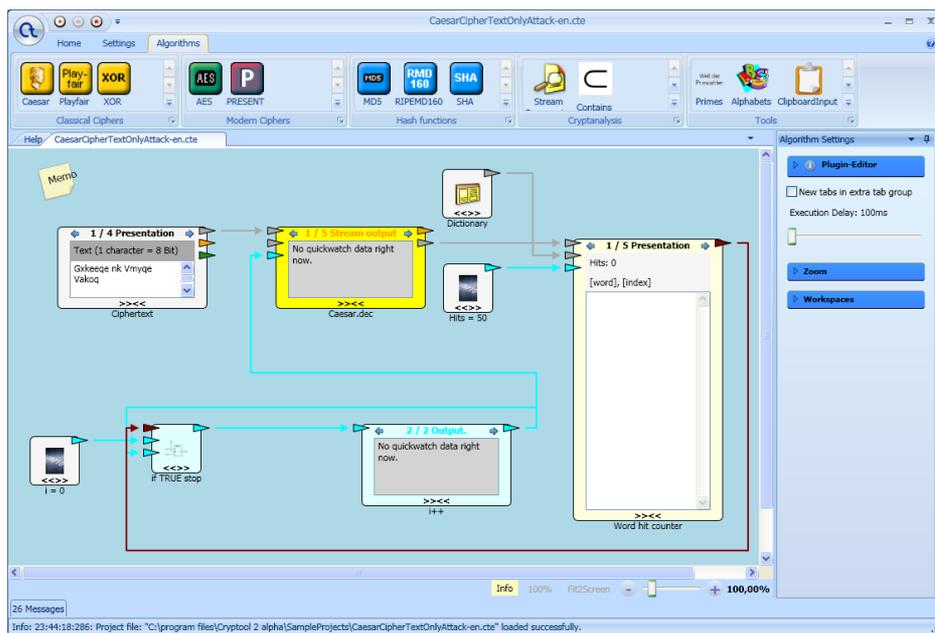
- CT 1.x Massenmustersuche
- JCT Visualisierung der Interoperabilität von S/MIME- und OpenPGP-Formaten
- JCT Tri-partite Schlüsselvereinbarung
- JCT Entropie-Untersuchungen
- JCT Statistische Untersuchung von Blockchiffreverfahren
- CT2 Umfangreiche Visualisierungen zum Thema Primzahlen
- CT2 Demo von Bleichenbachers Angriff auf RSA-Signaturen
- CT2 Demo virtueller Kreditkartennummern als Ansatz gegen Kreditkartenmissbrauch
- CT2 WEP-Verschlüsselung und WEP-Analyse
- CT2 Grafik-Design-orientierter Modus für Einsteiger plus Expertenmodus
- 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
- CT2/JCT Visualisierung des SSL-Protokolls
- CT2/JCT Demo zur Visuellen Kryptographie
- CT2/JCT Einbindung der Krypto-Bibliothek „Crypto++“ von Wei Dai



Weiterentwicklung (2)

In Arbeit (vgl. Readme-Datei)

1. JCT: Portierung und Neudesign von CrypTool in Java / SWT / Eclipse 3.4 / RPC
 - siehe: <http://jcryptool.sourceforge.net>
 - Meilenstein 2 für Benutzer und Entwickler verfügbar ab August 2008
2. CT2: Portierung und Neudesign von CrypTool mit C# / WPF / VS2008 / .NET 3.5
 - Direkter Nachfolger des aktuellen Releases: erlaubt visuelle Programmierung, ...
 - Beta 1 für Benutzer und Entwickler verfügbar ab Juli 2008
3. C2L: Direkte Portierung der bisherigen C++-Version nach Linux mit Qt4
 - siehe: <http://www.cryptoolinux.net>



CrypTool 2 (CT2)

CrypTool 1.4.21



JCryptTool (JCT)

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 weitergepflegt.

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

- Bis CrypTool 1.3.05: nur Visual C++ 6.0 (gab es als Buchbeilage kostenlos)
- Bis CrypTool 1.4.21: Visual C++ .net (= VC++ 7.1)(= Visual Studio 2003)
- Beschreibung für Entwickler: siehe readme-source.txt
- Download: Sourcen und Binaries der Release-Versionen
Interessierte und Entwickler erhalten auch die Sourcen der aktuellen Betas.

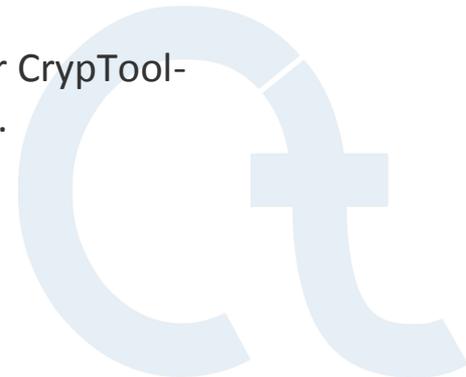
Zukünftige Entwicklungsumgebungen

- Für Versionen nach 1.4.2x:
 - CT2 – C#-Version: .NET mit Visual Studio 2008 Express Edition (kostenlos), WPF und Perl
 - JCT – Java-Version: mit Eclipse 3.4, SWT, RCP (kostenlos)
 - C2L – C++-Version für Linux mit Qt 4.x, GCC 4.x und Perl

CrypTool – Bitte um Mitwirkung

Wir freuen uns über jede weitere Mitarbeit

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



CrypTool – Fazit

- *DAS* E-Learning-Programm für Kryptologie
- Seit 10 Jahren ein erfolgreiches Open-Source-Projekt
- Mehr als 150.000 Downloads
- Weltweiter Einsatz in Schulen und Universitäten sowie Firmen und Behörden
- Umfangreiche Online-Hilfe und Dokumentation
- Frei verfügbar und mehrsprachig



Kontaktadresse

Prof. Bernhard Esslinger

Universität Siegen
Fachbereich 5, Wirtschaftsinformatik

Deutsche Bank AG
Direktor, IT-Security Manager

esslinger@fb5.uni-siegen.de

www.cryptool.com

www.cryptool.de

www.cryptool.org

www.cryptool.pl

www.iec.csic.es/cryptool

Weitere Kontaktadressen: siehe Readme im CrypTool-Programmpaket



Weitere Lektüre

als Einstieg in die Kryptologie

- Simon Singh: „*Geheime Botschaften*“, 2000, Hanser
- Klaus Schmeh: „*Codeknacker gegen Codemacher. Die faszinierende Geschichte der Verschlüsselung*“, 2. Auflage, 2007, W3L
- Udo Ulfkotte: „*Wirtschaftsspionage*“, 2001, Goldmann
- Johannes Buchmann: „*Einführung in die Kryptographie*“, 3. Auflage, 2004, Springer
- Claudia Eckert: „*IT-Sicherheit*“, 5. Auflage, 2008, Oldenbourg
- A. Beutelspacher / J. Schwenk / K.-D. Wolfenstetter: „*Moderne Verfahren der Kryptographie*“, 5. Auflage, 2004, Vieweg
- [HAC] Menezes, van Oorschot, Vanstone: „*Handbook of Applied Cryptography*“, 1996, CRC Press
- van Oorschot, Wiener: „*Parallel Collision Search with Application to Hash Functions and Discrete Logarithms*“, 1994, ACM
- Vielfältige Krypto-Literatur – siehe Links auf der CrypTool-Webseite sowie Quellenangaben in der Online-Hilfe von CrypTool (z.B. Bücher von Wätjen, Salomaa, Brands, Schneier, Shoup, Stamp/Low, ...)
- Bedeutung der Kryptographie in dem breiteren Rahmen von IT-Sicherheit, Risiko-Management und organisatorischen Kontrollen
 - Siehe z.B. Kenneth C. Laudon / Jane P. Laudon / Detlef Schoder: „*Wirtschaftsinformatik*“, 2005, Pearson, Kapitel 14
 - Siehe Wikipedia (<http://de.wikipedia.org/wiki/Risikomanagement>)



The screenshot shows the homepage of the CryptTool website. At the top left is the CryptTool logo, which consists of a stylized 't' inside a circle followed by the text 'CRYPTtOOL'. Below the logo is a navigation menu with the following items: 'Über', 'Funktionen', 'Screenshots', 'Dokumentation', 'Download', and 'Cryptoportal für Lehrer'. To the right of the menu are flags for Germany, UK, Spain, and Poland. Below the navigation menu is a yellow banner with the text 'Aktuelle stabile Version: 1.4.21' and a 'Download' link. The main content area is divided into two columns. The left column contains a sidebar with a 'Über' section and a list of links: 'Was ist CryptTool?', 'CryptTool in der Lehre', 'CryptTool für Awareness', 'Präsenz in Printmedien', 'Auszeichnungen', 'Mitwirkende', 'Partnerprojekte', and 'Kontakt'. Below this list is a badge that says 'Ausgewählter Ort 2008 in: Deutschland Land der Ideen' with a row of colorful dots. At the bottom of the sidebar is a quote: 'CryptTool has been produced in an extremely professional and innovative way. It is helpful as an educational tool for the novice and provides'. The right column features a section titled 'Was ist CryptTool?' with a print icon. Below the title is a paragraph: 'Das Programm CryptTool ist ein freies E-Learning-Programm für Windows, mit dem kryptographische Verfahren angewendet und analysiert werden können. Diese Software wird weltweit eingesetzt. Dabei unterstützt sie eine moderne Lehre an Schulen und Hochschulen sowie die Sensibilisierung von Firmenangehörigen.' This is followed by a sub-section 'Die aktuelle Version bietet unter anderem:' and a bulleted list of features: 'Zahlreiche klassische und moderne kryptographische Algorithmen (Ver- und Entschlüsselung, Schlüsselerzeugung, sichere Passworte, Authentisieren, sichere Protokolle, ...)', 'Visualisierung einiger Verfahren (z.B. Caesar, Enigma, RSA, Diffie-Hellman, Digitale Signaturen, AES)', 'Kryptoanalyse gegen ausgewählte Algorithmen (z.B. Vigenère, RSA, AES)', 'Kryptoanalytische Messverfahren (z.B. Entropie, N-Gramme, Autokorrelation)', 'Unterstützende Verfahren (z.B. Primzahltest, Faktorisierung, Base64-Kodierung)', 'Lernprogramm zur Zahlentheorie', 'Umfangreiche Online-Hilfe', and 'Begleitendes Skript mit weiterführenden Informationen über Kryptologie'. Below the list is a paragraph: 'Ursprünglich für IT-Sicherheits-Schulungen im Unternehmen entwickelt, hat sich CryptTool inzwischen zu einem bedeutenden Open-Source-Projekt im Bereich Kryptologie entwickelt.' This is followed by another paragraph: 'Seit dem Frühjahr 2008 wird durch das CryptTool-Projekt auch das Cryptoportal für Lehrer betrieben. Dieses Portal wurde auf Anregung mehrerer Lehrer ins Leben gerufen. Hier soll insbesondere Schul-Lehrern eine Plattform geboten werden, auf der sie Unterrichtsmaterialien und Links rund um das Thema Kryptologie zur Verfügung stellen können.' At the bottom of the right column is a paragraph: 'Derzeit arbeitet das CryptTool-Team an zwei Zukunftsprojekten für die aktuelle, in C++ geschriebene CryptTool-Version 1.4.x. Beide Nachfolgeprojekte nutzen modernste Standards der Software-Entwicklung, sind aber noch im'.



CRYPTOPORTAL für Lehrer

- Über
- Unterrichtsmaterial
- Linksammlung
- Registrierung
- Cryptool
- Einloggen

Filterkriterien

Land:

Schultyp:

Autor:

Material enthält folgenden Text:

Unterrichtsmaterial

[1] **Die Stromchiffre A5**

Autor: PS
Land: Deutschland - alle Bundesländer
Schultyp: Gymnasien

In dieser Ausarbeitung zum Seminar IT-Sicherheit wird der auf der Verschaltung von linear rückgekoppelten Schieberegistern (LFSR) basierende Algorithmus A5 und die bisher gefundenen [...]

 [a5_thesis.pdf](#) 8 mal heruntergeladen

[2] **Die wichtigsten Verfahren der Kryptologie**

Autor: HW
Land: Deutschland - Berlin
Schultyp: alle Schultypen

Die Präsentation besteht aus zwei Folien. In der ersten wird die Entwicklung der klassischen Kryptographie (von Caesar bis zum one-time-pad) dargestellt. In der zweiten wird ein Überblick zur [...]

 [Krypto-Entwicklung.ppt](#) 15 mal heruntergeladen

[3] **Kryptografie für Jedermann**

Autor: Consultant
Land: Deutschland - alle Bundesländer
Schultyp: alle Schultypen

Einführung in die Kryptografie, Erläuterungen zu populären kryptografischen Primitiven und Protokolle [...]

 [Originalpraesentation.pdf](#) 14 mal heruntergeladen