Verschlüsselung, Kryptographie und Kryptoanalyse verstehen

Praktische Einführung
anhand von Beispielen mit
dem Lernprogramm CrypTool

Prof. Bernhard Esslinger
und CrypTool-Team

April 2016

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 wird schon lange nicht mehr nur von Agenten, Diplomaten und Militärs benutzt. 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, 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)

Übersicht

Inhalt der Präsentation

Was bietet CrypTool

Was ist CrypTool

Was kann CrypTool?

- Kostenloses, Open-Source Lern-Programm mit graphischer Oberfläche.
  Es gibt 3 Offline-Varianten (diese werden hier genauer vorgestellt) und eine Online-Variante.
- Kryptographische Verfahren anwenden *und* analysieren
- „Spielerischer“ Einstieg in moderne und klassische Kryptographie
- Sehr umfangreiche Online-Hilfe; ohne tieferes Kryptographie-Wissen verständlich
- Enthält fast alle State-of-the-art-Funktionen der Kryptographie
- Entwickelt von derzeit rund 70, meist ehrenamtlich arbeitenden Personen weltweit

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

Zielgruppe

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

Funktionsumfang (Auszug)

(1/3) [vergleiche https://www.cryptool.org/de/ctp-dokumentation/ctp-funktionsumfang]

Kryptographie

Verschlüsselungsklassiker

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

Optionen zum besseren Nachvollziehen von Literaturbeispielen

- Alphabet wählbar
- Behandlung von Sonderzeichen einstellbar

Kryptoanalyse

Angriffe auf klassische Verfahren

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

Unterstützende Analyseverfahren

- Entropie, gleitende Häufigkeit
- Histogramm, n-Gramm-Analyse
- Autokorrelation
- Perioden
- Zufallszahlenanalyse

Funktionsumfang (Auszug)
(2/3)

Kryptographie

Moderne symmetrische Verschlüsselung

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

Asymmetrische Verschlüsselung

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

Hybridverschlüsselung (RSA + AES)

- Visualisiert als interaktives Datenflussdiagramm

Kryptoanalyse

Brute-Force-Angriff auf symmetrische Algorithmen

- Für alle Algorithmen
- Annahmen:
  - Entropie des Klartextes klein,
  - Teilweise Kenntnis der Schlüssels, oder
  - Kenntnis des Klartextalphabets

Angriff auf RSA-Verschlüsselung

- Faktorisierung des RSA-Moduls*
- Gitterreduktions-basierte Angriffe

Angriff auf Hybridverschlüsselung

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

* Vergleiche die Untersuchung von Schulz und Witten: „Zeit-Experimente zur Faktorisierung – Ein Beitrag zur Didaktik der Kryptologie“, in: LOG IN, Heft 166/167, 2010, S. 107-114, http://bscw.schule.de/pub/bscw.cgi/d864899/Schulz_Witten_Zeit-Experimente.pdf

Funktionsumfang (Auszug)
(3/3)

Kryptographie

Digitale Signatur

- RSA mit X.509-Zertifikaten
  - Signatur zusätzlich visualisiert als interaktives Flussdiagramm
- 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 mod 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 (Geburtstagsparadoxon). (Bei 40 Bit finden moderne PCs eine Kollision für beliebige Hashfunktionen in weniger als 5 Minuten.)

Analyse von Zufallsdaten

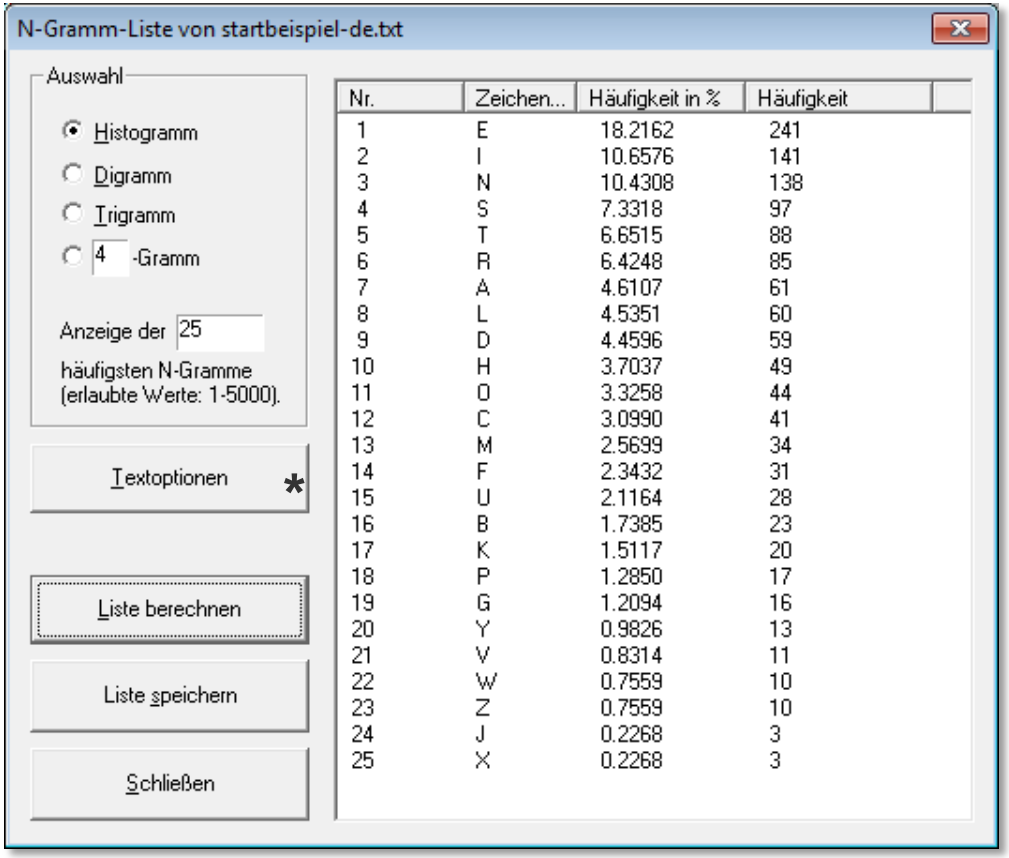
- FIPS-PUB-140-1 Test-Batterie
- Periode, Vitányi, Entropie
- Gleitende Häufigkeit, Histogramm
- n-Gramm-Analyse, Autokorrelation
- ZIP-Kompressionstest

Sprachstruktur analysieren

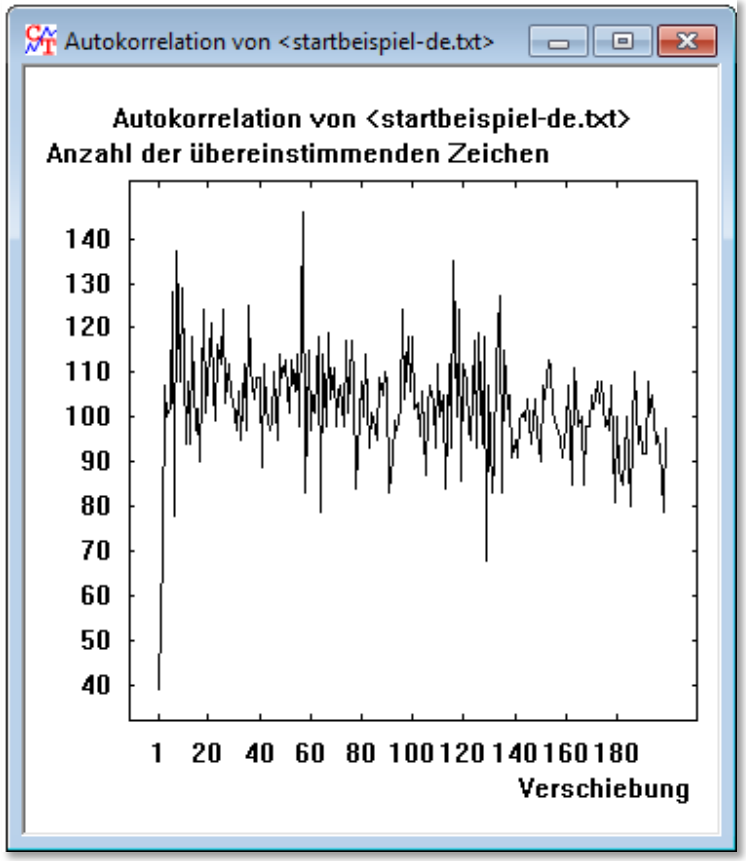
Hier ein Beispiel mit CrypTool 1 (CT1)

Anzahl Einzelzeichen, n-Gramme, Entropie

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



Nr.	Zeichen...	Häufigkeit in %	Häufigkeit
1	E	18.2162	241
2	I	10.6576	141
3	N	10.4308	138
4	S	7.3318	97
5	T	6.6515	88
6	R	6.4248	85
7	A	4.6107	61
8	L	4.5351	60
9	D	4.4596	59
10	H	3.7037	49
11	O	3.3258	44
12	C	3.0990	41
13	M	2.5699	34
14	F	2.3432	31
15	U	2.1164	28
16	B	1.7385	23
17	K	1.5117	20
18	P	1.2850	17
19	G	1.2094	16
20	Y	0.9826	13
21	V	0.8314	11
22	W	0.7559	10
23	Z	0.7559	10
24	J	0.2268	3
25	X	0.2268	3



* Hier waren die Einstellungen in den Textoptionen so, dass nur Buchstaben gewählt wurden und die Groß- und Kleinbuchstaben nicht unterschieden wurden.

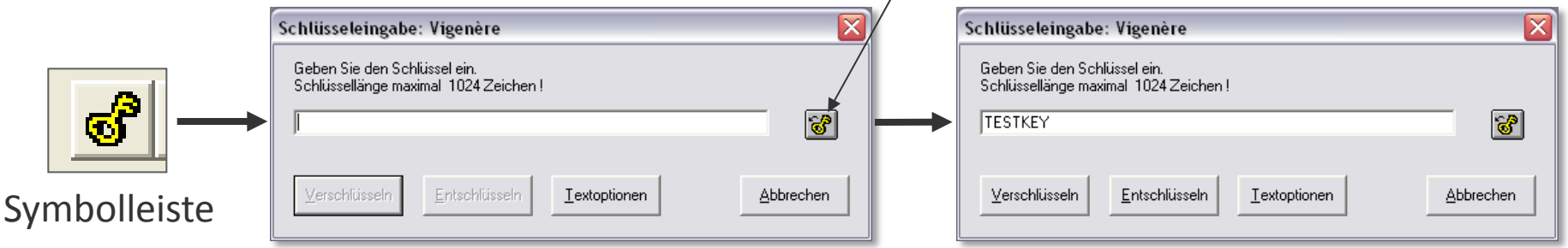
Konzepte zur Benutzerfreundlichkeit

Kontext-sensitive 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.

Einfügen von Schlüsseln in die Schlüsseleingabe-Maske

- Mit Strg-V (Paste) kann man immer einfügen, was in der Zwischenablage (Clipboard) enthalten ist
- Die zur Verschlüsselung benutzten Schlüssel können aus den Geheimtext-Fenstern per Icon in der Symbolleiste „entnommen“ und durch ein komplementäres Icon in der Schlüsseleingabemaske in das Schlüsselfeld eingefügt werden. Dazu wird ein CrypTool-interner Schlüssel-Speicher benutzt, der pro Verfahren zur Verfügung steht. Dies ist insbesondere bei „strukturierten“ Schlüsseln wie der homophonen Verschlüsselung hilfreich.



Schlüsseleingabe-Dialog **vor** dem Klick auf das Icon

Schlüsseleingabe-Dialog **nach** dem Klick auf das Icon

Herausforderungen für unsere Programmierer

Verschiedene Funktionen parallel laufen lassen

- Beispiel: Bei der Faktorisierung laufen die verschiedenen Algorithmen in Threads

Hohe Performance

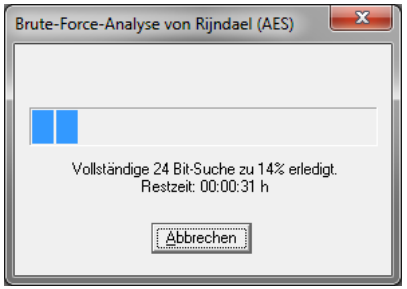
- Beim Finden von Hashkollisionen (Anwendung des Geburtstagsparadoxons) oder bei der Brute-Force-Analyse

Speicherbeschränkung beachten

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

Zeitmessung und -abschätzung

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



Wiederverwendung / Integration

- Masken zur Primzahlgenerierung
- RSA-Kryptosystem (schaltet nach erfolgreicher Attacke von der Ansicht des Public-Key-Anwenders zur Ansicht des Private-Key-Besitzers). Ein Angreifer kennt zu Beginn auch nur den Public-Key.

Automatisierung der Konsistenz der Ressourcen (Funktionen, GUI, Hilfe)

- Inklusive verschiedener Sprachen

Übersicht

Inhalt der Präsentation

Ausgewählte Beispiele mit CrypTool

Inhalt des Kapitels

Skytale von Sparta

Die Idee

Generelle Einordnung und historischer Hintergrund

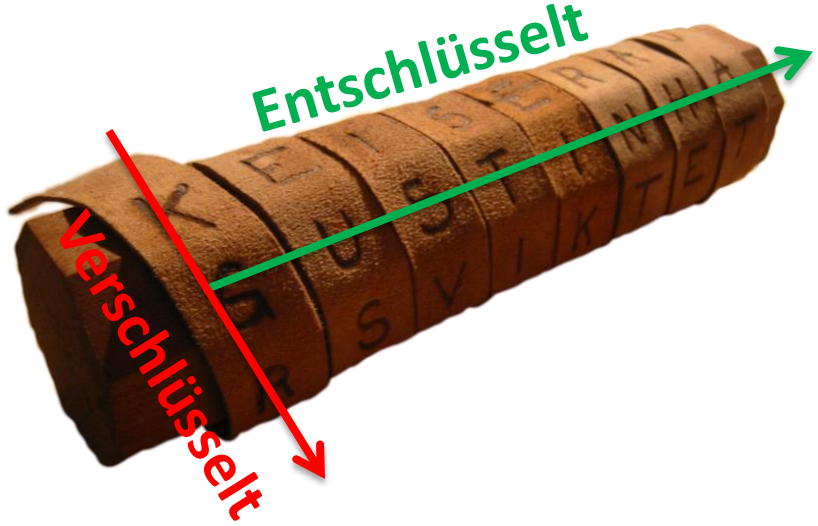
- Die Skytale von Sparta (etwa 500 v. Chr.) ist ein **Transpositions-Verfahren** – die Buchstaben des Klartextes bleiben erhalten, werden aber in ihrer Position vertauscht.
- Die Skytale ist das älteste militärisch genutzte Verschlüsselungsverfahren.
- Beschrieben vom griechischen Historiker/Schriftsteller Plutarch (45 - 125 n. Chr.)

Verschlüsselung

- Ein Band wird um einen eckigen Zylinderstab gewickelt und der Klartext wird längs der Zylinderseitenflächen auf das aufgewickelte Band geschrieben. Abgewickelt ergibt sich auf dem Band eine scheinbar willkürliche Zeichenfolge.

Entschlüsselung

- Wird das Band um einen Zylinder mit gleichem Durchmesser gewickelt, so lässt sich der Klartext wieder lesen.



Skytale von Sparta

Implementierung in CrypTool 1 (CT1)

Menü in CrypTool 1

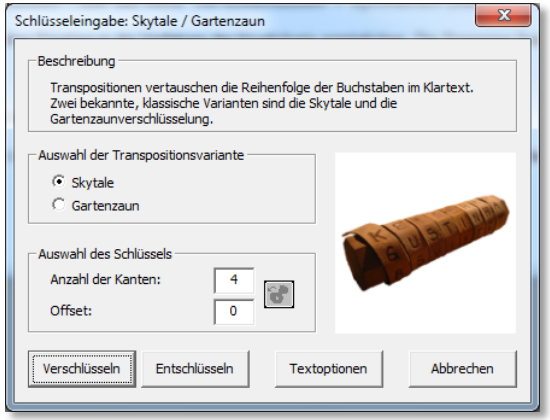
- „Ver-/Entschlüsseln“ \ „Symmetrisch (klassisch)“ \ „Skytale/Gartenzaun …“

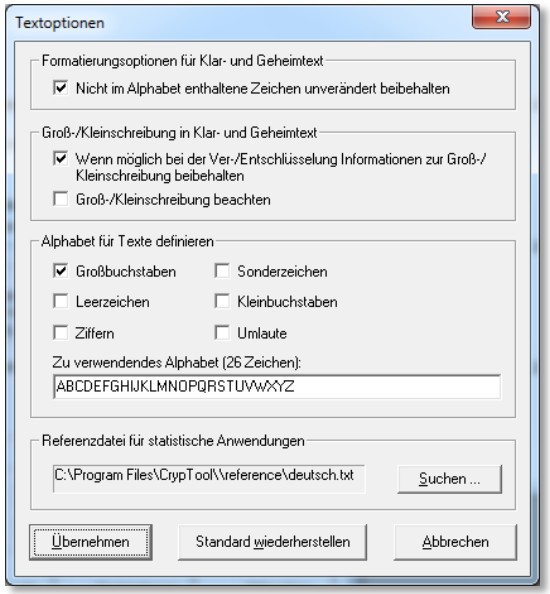
Auswahl des Schlüssels

- Anzahl der Kanten
- Offset

Textoptionen

- Erreichbar über das Menü: „Optionen“ \ „Textoptionen“ oder direkt bei der Schlüsseleingabe über den Button „Textoptionen“
- Einstellung der Art der Verarbeitung von Groß-/Kleinbuchstaben
- Definition des Alphabets zur Verarbeitung des Textes (z.B. welche Zeichen ver-/entschlüsselt werden)
- Auswahl der Datei mit statistischen Referenz-Kennzahlen (z.B. für die Häufigkeitsanalyse) zur Anpassung der statistischen Unterschiede verschiedener Sprachen



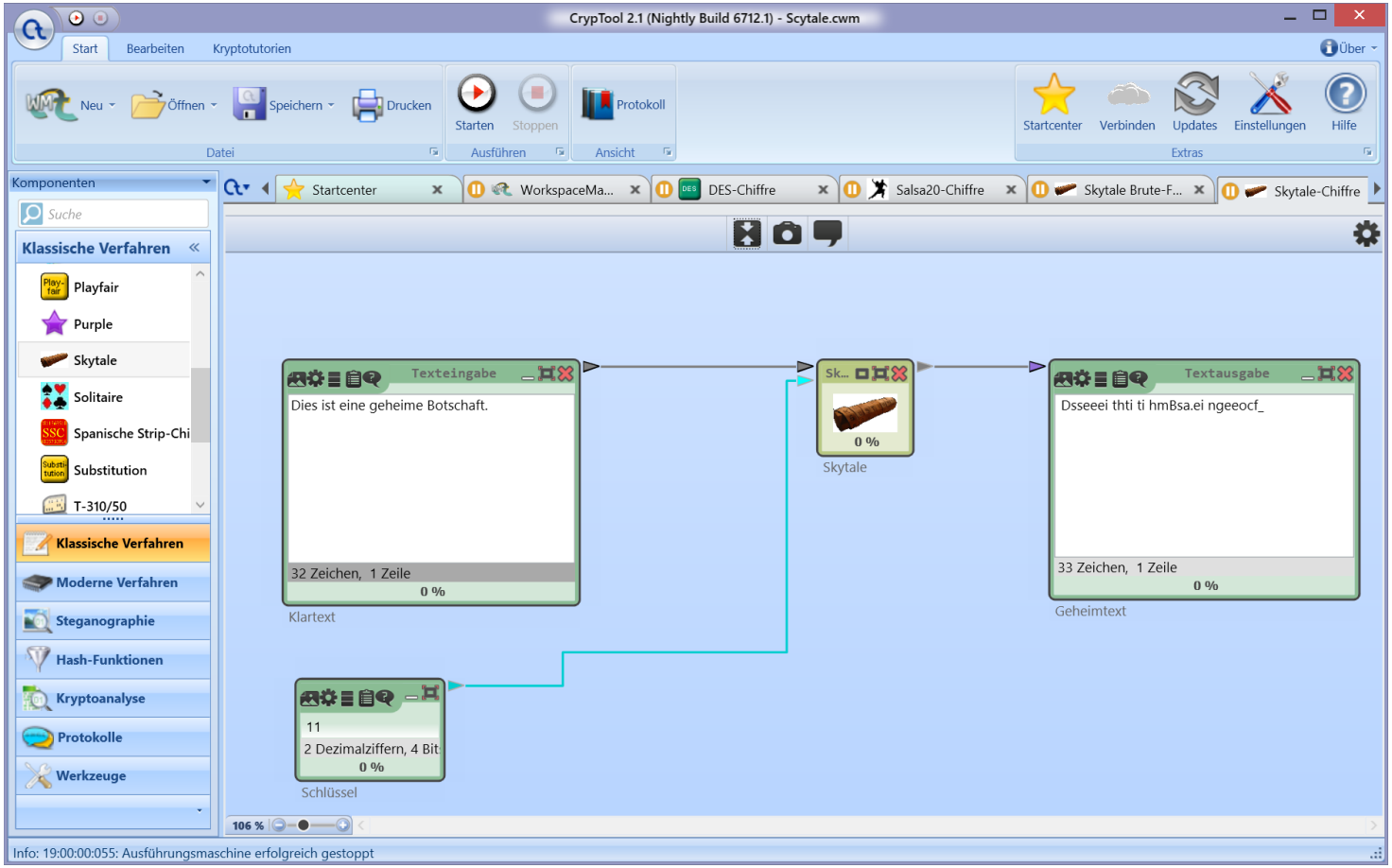


Skytale von Sparta

Implementierung in CrypTool 2 (CT2)

Vorlage in CrypTool 2

- Kryptographie → Klassisch → Skytale-Chiffre



Skytale von Sparta
Zusammenfassung

- Die Skytale ist ein Transpositions-Verfahren, die Reihenfolge der Zeichen im Klartext wird nur vertauscht, aber alle Zeichen bleiben unverändert erhalten (keine Substitution der Zeichen).

- Große Verbreitung im persischen Reich. Nach dem Untergang der klassischen Kulturen taucht das Verfahren erst wieder im Mittelalter auf.*

- Es werden zwei Zylinder (Holzstäbe) mit gleichem Durchmesser benötigt. Einmal zum Verschlüsseln und einmal zum Entschlüsseln der Nachricht.

- Mehrfaches Anwenden dieser Verschlüsselung kann auf eine einzelne Transposition (Zeichen des Klartextes werden umsortiert) zurückgeführt werden.

- Beispiel:

  Klartext: `STARTBE ISPIEL`

  Geheimtext: `SITSAPR ITEBLE`

  Schlüssel: 2

  Parameter: Alphabet soll nur Großbuchstaben enthalten, Nicht-Alphabetzeichen bleiben stehen.

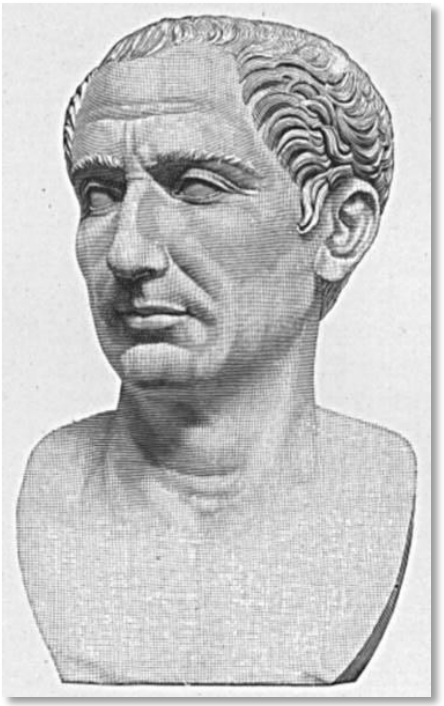
* Quelle Wikipedia: http://de.wikipedia.org/wiki/Skytale

Caesar-Verschlüsselung

Die Idee

Generelle Einordnung und historischer Hintergrund

- Die Caesar-Verschlüsselung (Julius Caesar, 100 - 44 v.Chr.) ist eine einfache **Substitutions-Verschlüsselung**.



Verschlüsselung

- Zur Verschlüsselung wird jeder Buchstabe aus dem Klartextalphabet auf den entsprechenden Buchstaben des Geheimtextalphabets abgebildet. Mit dieser Abbildung werden die Zeichen des Alphabets lediglich um einen konstanten, definierten Wert verschoben. Dieser numerische Wert ist der Schlüssel.

```
GALLIA  EST  OMNIS  DIVISA ...

Klartextalphabet:     ABCDEFGHIJKLMNOPQRSTUVWXYZ
Geheimtextalphabet:   DEFGHIJKLMNOPQRSTUVWXYZABC

JDOOLD  HVW  RPQLV  GLYLVD ...
```

Entschlüsselung

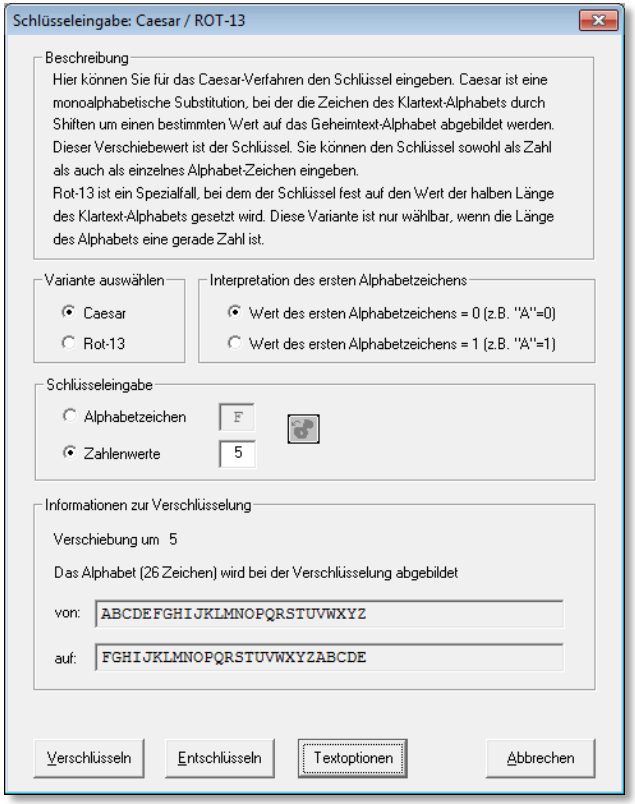
- Zur Entschlüsselung wird die Abbildung invers zur Verschlüsselung ausgeführt: War der Schlüssel zum Beispiel eine Verschiebung des Alphabets um +4, kann man mit einer Verschiebung des Alphabets von -4 die Verschlüsselung wieder aufheben.

Caesar-Verschlüsselung

Implementierung in CrypTool 1 (CT1)

Menü in CrypTool 1

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



Auswahl des Schlüssels

- Den Wert des ersten Buchstabens als 0 oder 1 definieren.
- Schlüssel als Zahl oder als Buchstaben eingeben
  (ergibt die Verschiebung)

Textoptionen

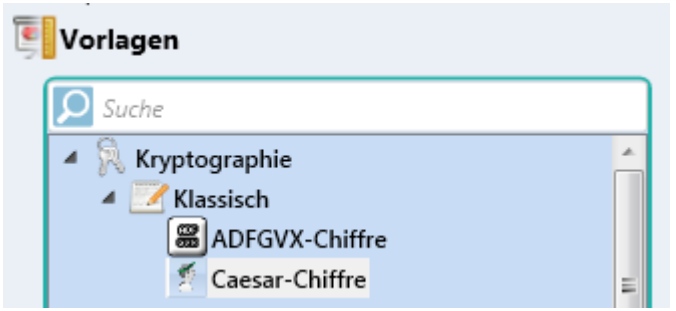
- Standardmäßig werden die 26 Großbuchstaben als Alphabet definiert (und Kleinbuchstaben vor der Verschlüsselung in Großbuchstaben konvertiert).
  Im Textoptionen-Dialog (siehe Seite 16 bei Skytale von Sparta) kann man aber auch andere Zeichen in das Alphabets aufnehmen oder Klein- und Großbuchstaben unterschiedlich behandeln.

Caesar-Verschlüsselung

Implementierung in CrypTool 2 (CT2)

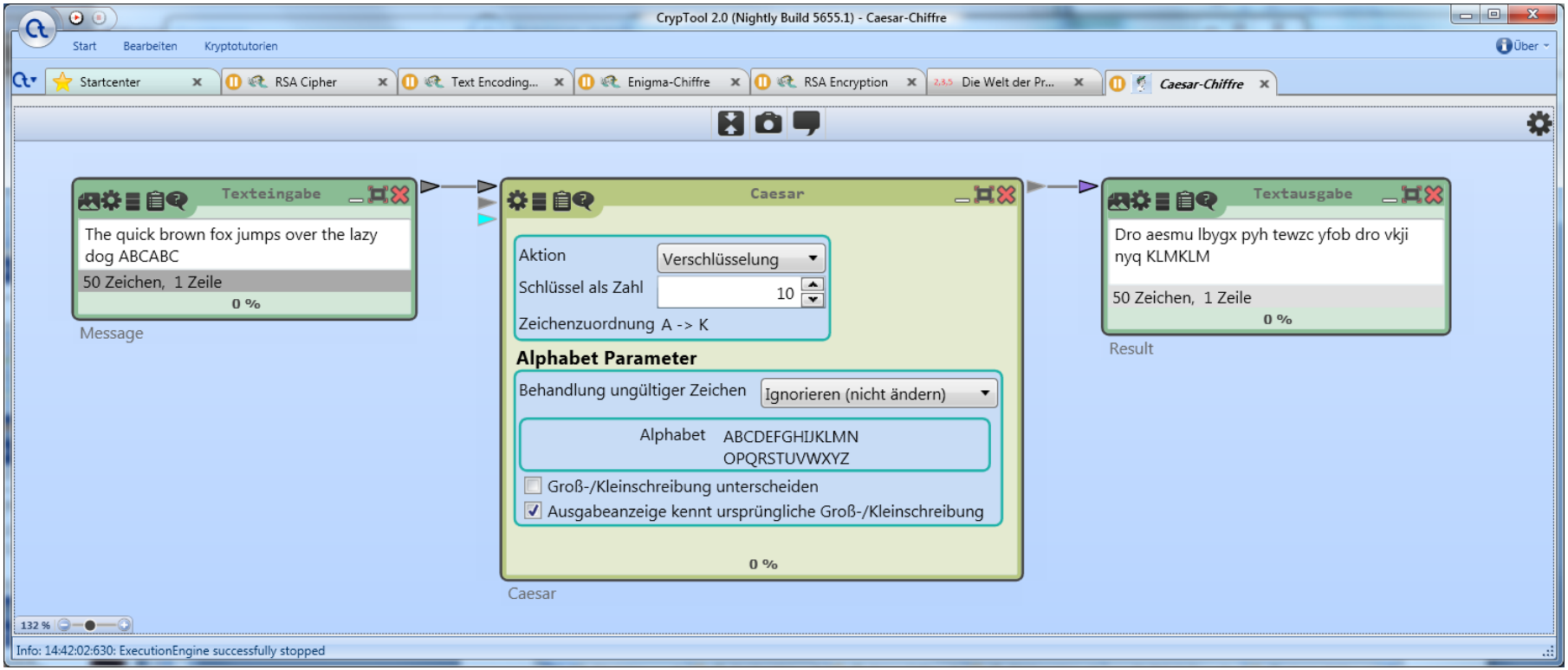
Vorlage in CrypTool 2

- **Kryptographie → Klassisch → Caesar-Chiffre**



Änderungen im Arbeitsbereich sofort sichtbar

- Nach jeder neuen Eingabe wird das Ergebnis sofort angezeigt.



Caesar-Verschlüsselung

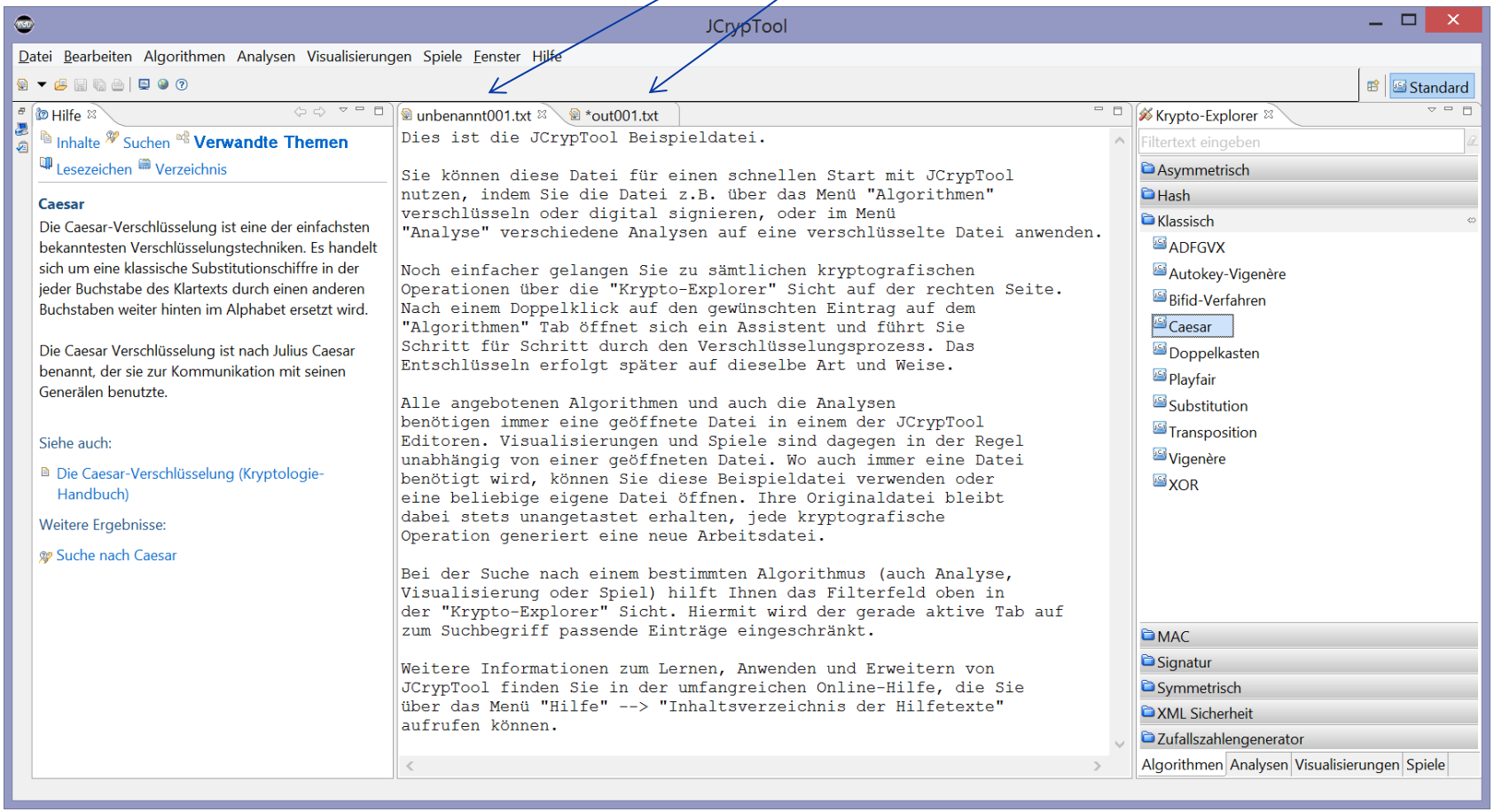
Implementierung in JCrypTool (1/2)

Menü in JCrypTool

- **Anwendung**
  „Algorithmen“ \ „Klassisch“ \ „Caesar“

Reiter mit dem Klartext

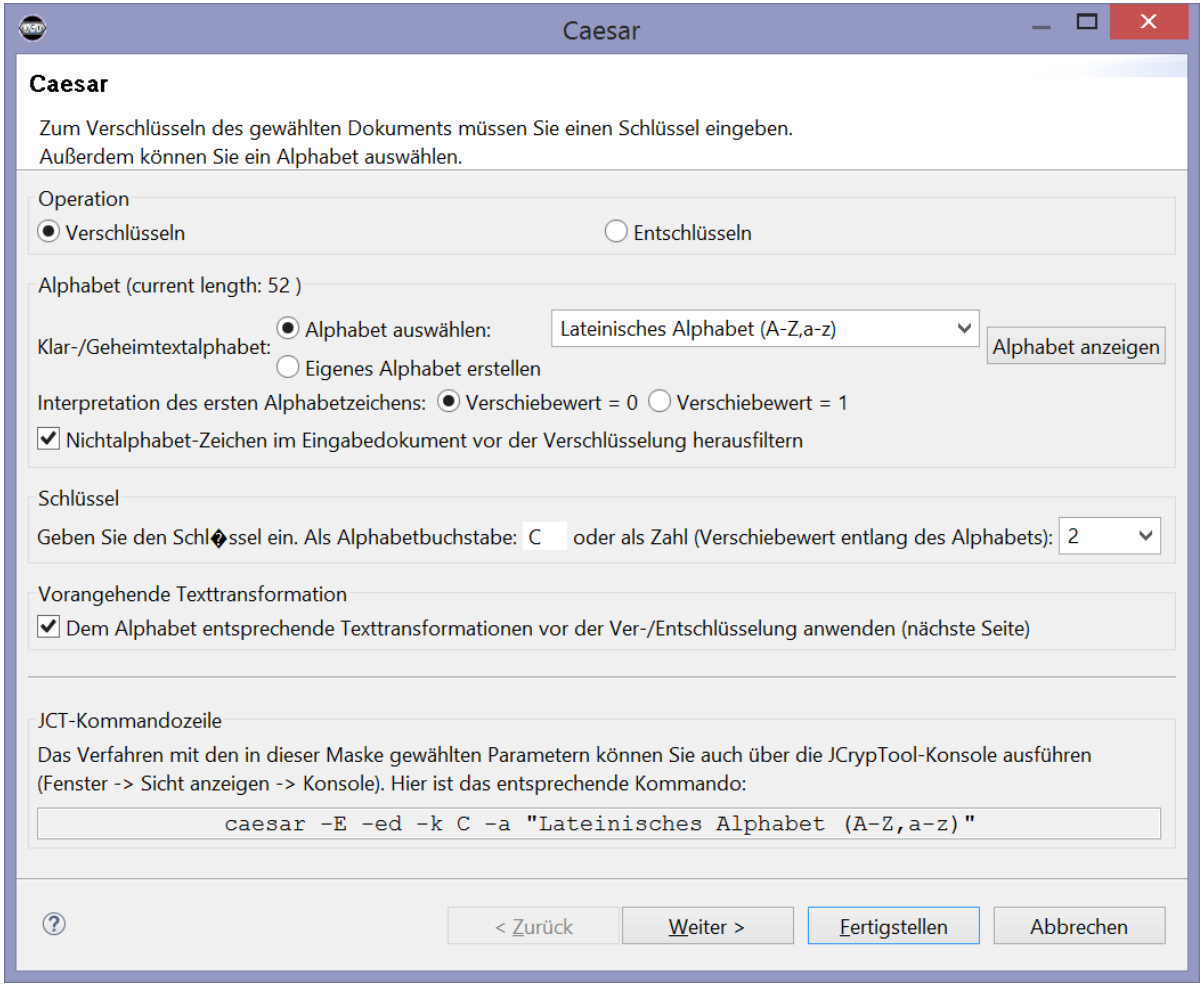
Reiter mit dem verschlüsselten Ergebnis



Caesar-Verschlüsselung

Implementierung in JCrypTool (2/2)

Standarddialog in JCT zur Eingabe der Parameter und des Schlüssels



Caesar-Verschlüsselung
Zusammenfassung

- Die Caesar-Verschlüsselung ist ein mono-alphabetisches Substitutions-Verfahren. Es wurde von dem römischen Schriftsteller Sueton (Gaius Suetonius Tranquillus, 70 - 122 n.Chr.) überliefert.
- Der Spezialfall ROT-13 ist ein involutorisches Verfahren.
- Diese Verschlüsselung kann mit einer Häufigkeitsanalyse angegriffen werden. Die Häufigkeitsanalyse beruht auf der ungleichen Verteilung von Buchstaben in natürlichen Sprachen, die von dieser Verschlüsselung nicht verborgen wird.
- Eine weitere Möglichkeit besteht darin, alle Schlüssel auszuprobieren. Bei einer Alphabetlänge von 26 sind zum Erraten des Schlüssels maximal 25 Versuche nötig.
- Die Sicherheit des Verfahrens basierte auf der Geheimhaltung des Verfahrens (die Schlüssel waren bei Caesar festgelegt auf den Wert 3), was dem Kerckhoffs'sche Prinzip* widerspricht.

* Das Kerckhoffs'sche Prinzip (auch als Kerckhoffs' Maxime bezeichnet) wurde 1883 von Auguste Kerckhoffs formuliert. Es ist ein Grundsatz der modernen Kryptographie, der besagt, dass die Sicherheit eines Verschlüsselungsverfahren auf der Geheimhaltung des Schlüssels und nicht des Verfahrens beruhen muss.

Enigma-Verschlüsselung

Die Idee (Rotormaschinen / Elektromechanische Verschlüsselungsmaschinen)

Enigma-Verschlüsselung

- Erfinder der Enigma war Arthur Scherbius (1878-1929)
- Über 200.000 Maschinen waren im 2. Weltkrieg im Einsatz.
- Der rotierende Walzensatz bewirkt, dass jedes Zeichen des Textes mit einem neuen Alphabet verschlüsselt wird (poly-alphabetische Verschlüsselung).
- Das polnische Cipher Bureau brach die Vorkriegs-Enigma schon 1932.
- Darauf aufbauend wurde die Kriegs-Enigma gebrochen, mit massivem Einsatz (etwas 7000 Personen in GB), unter Nutzung der ersten Entschlüsselungsmaschinen sowie erbeuteter Original-Maschinen, abgefangener täglicher Statusmeldungen (z.B. Wetternachrichten), etc.

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.“*

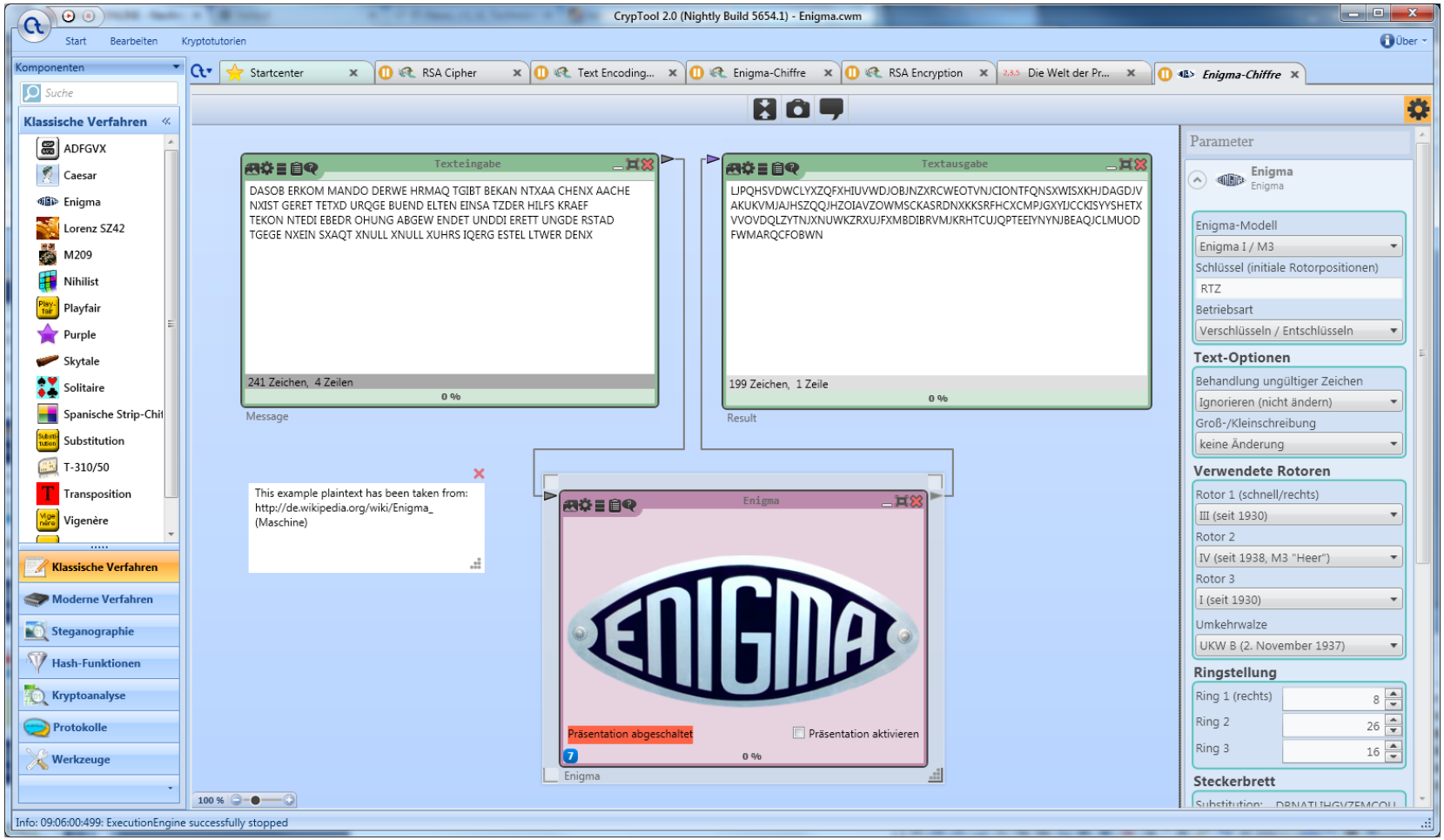
* vgl. auch Wikipedia: https://de.wikipedia.org/wiki/Enigma_(Maschine)

Enigma-Verschlüsselung

Implementierung in CrypTool 2 – Darstellung ohne Präsentation (Visualisierung)

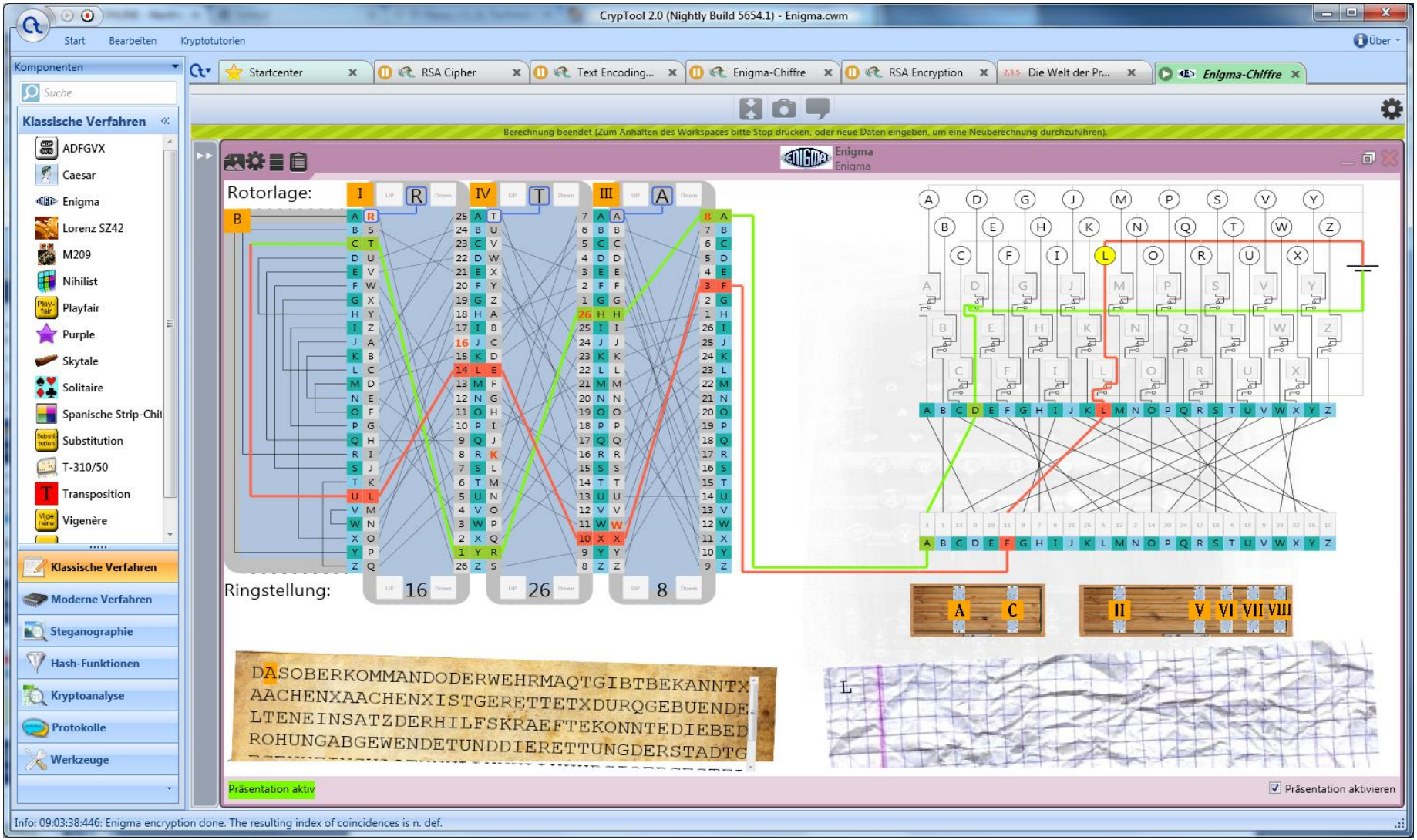
Vorlage in CrypTool 2

- Kryptographie → Klassisch → Enigma-Chiffre



Enigma-Verschlüsselung

Implementierung in CrypTool 2 – Darstellung mit Präsentation (Visualisierung): D → L



Visualisierungen von AES (Rijndael-Chiffre) (1 / 3)

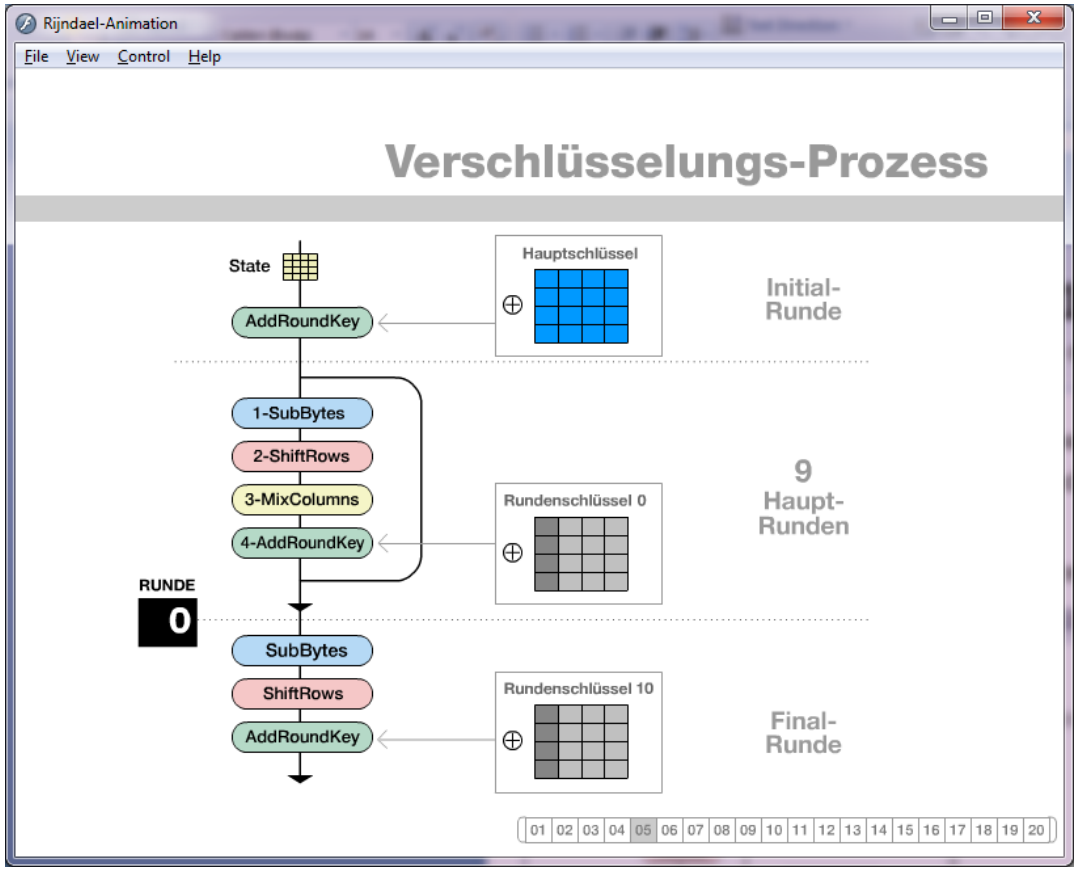
Rijndael-Animation

Rijndael-Animation

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

Menü in CrypTool 1

- „Einzelverfahren“ \ „Visualisierung von Algorithmen“ \ „AES“ \ „Rijndael-Animation“



Visualisierungen von AES (Rijndael-Chiffre) (2 / 3)

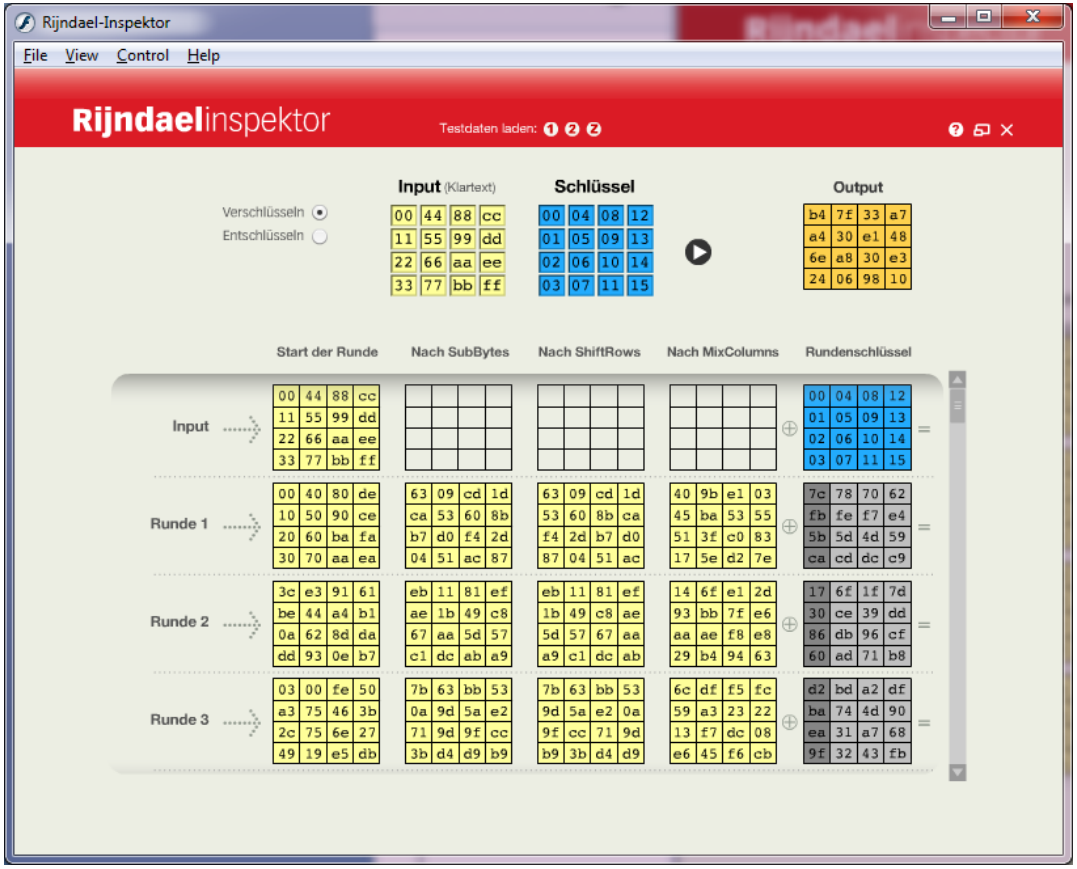
Rijndael-Inspektor

Rijndael-Inspektor

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

Menü in CrypTool 1

- „Einzelverfahren“ \ „Visualisierung von Algorithmen“ \ „AES“ \ „Rijndael-Inspector“



Visualisierungen von AES (Rijndael-Chiffre) (3 / 3)

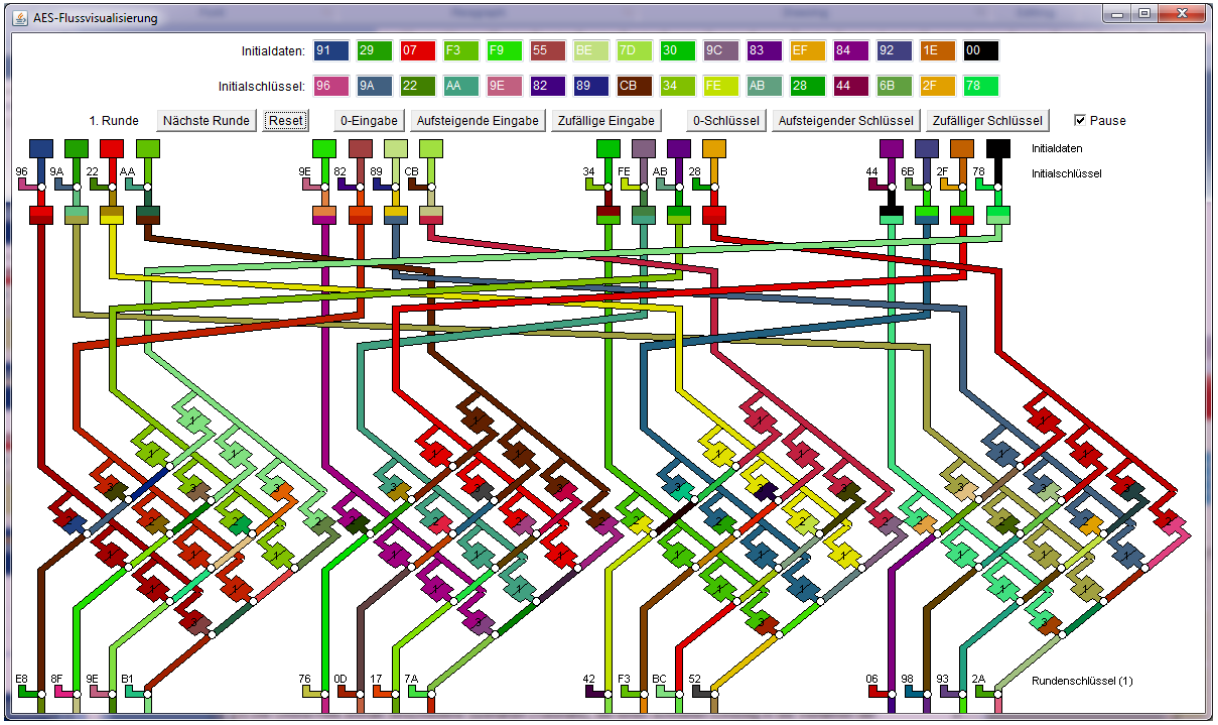
Rijndael-Flussvisualisierung

Rijndael-Flussvisualisierung

- Visualisierung der Datenveränderung je Runde durch Farbverläufe

Menü in CrypTool 1

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

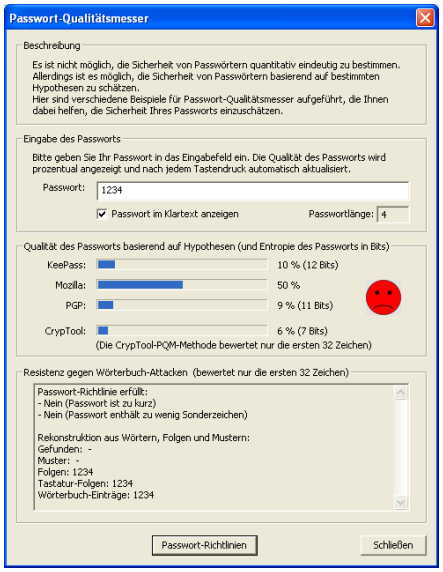


Passwort-Qualitätsmesser (PQM) und Passwort-Entropie

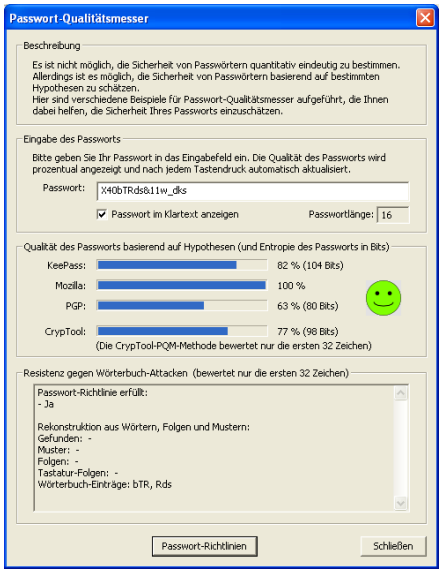
Funktionen

- 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

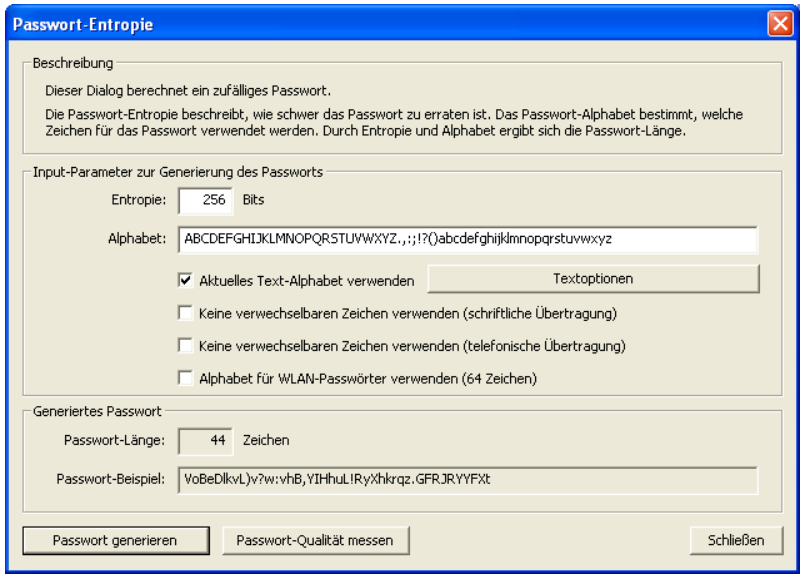
Passwort: ***1234***



Passwort: ***X40bTRds&11w_dks***



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



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

Passwort-Qualitätsmesser (PQM) und Passwort-Entropie

Erkenntnisse des Passwort-Qualitätsmessers

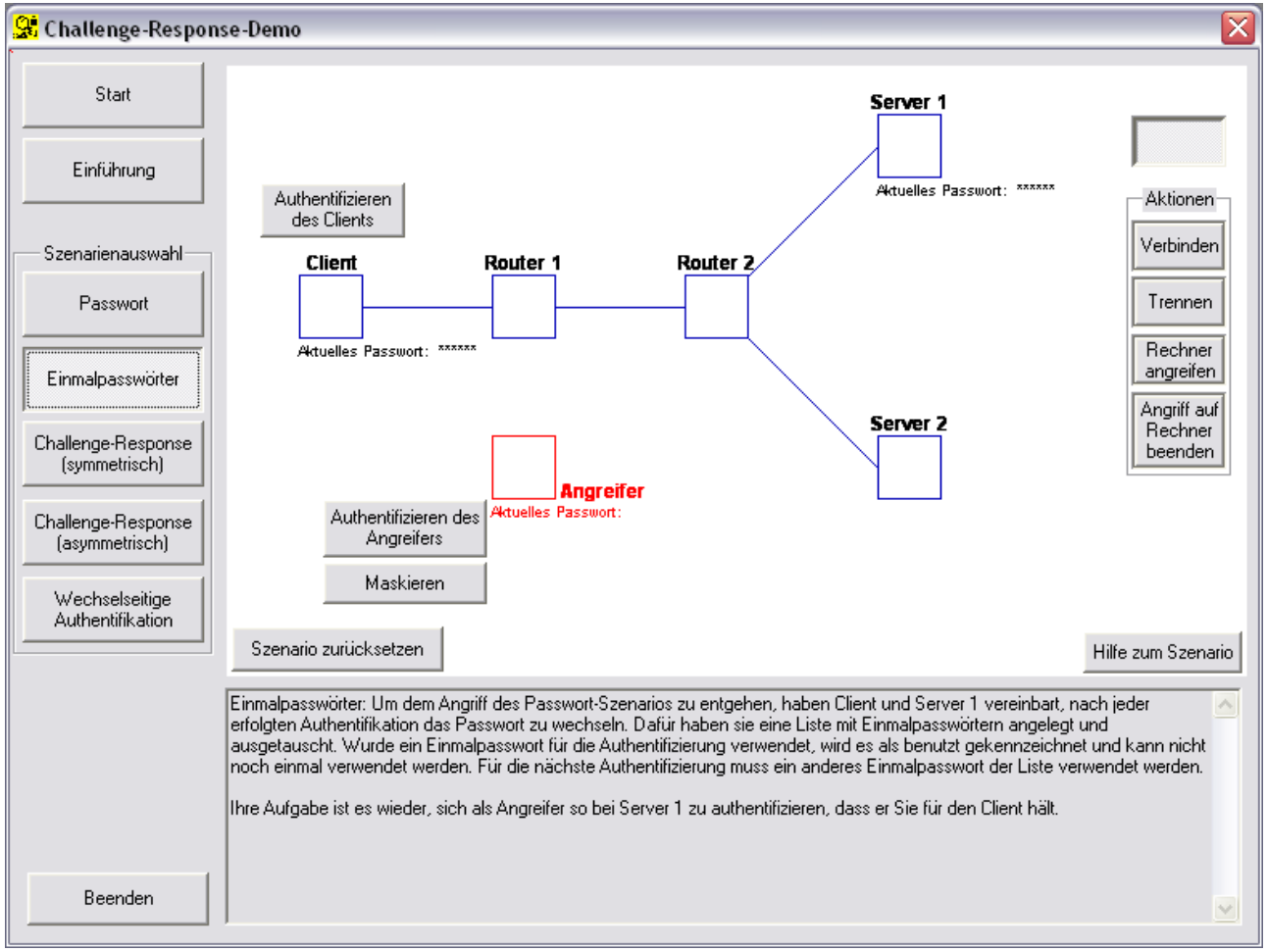
- Höhere Qualität des Passwortes durch die Verwendung von **verschiedenen Zeichenarten**: Groß-/Kleinschreibung, Zahlen und Sonderzeichen (**Passwortraum**)
- Passwortqualität hängt primär von der **Länge des Passwortes** ab!
- **Passwortentropie** als Maß der Zufälligkeit der Wahl von Zeichen aus dem Passwortraum (je zufälliger die Wahl, desto besser das Passwort)
- Passwörter sollten **nicht in einem Wörterbuch vorkommen**.

Qualität eines Passwortes aus der Angreifer-Perspektive

- Angriff auf ein Passwort (sofern beliebig viele Versuche zugelassen sind):
  1. Klassischer **Wörterbuchangriff**
  2. Wörterbuchangriff mit **weiteren Varianten** (z.B. Jahreszahlen anhängen: 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 9 Zeichen lang ist, und Zahlen sowie Sonderzeichen beinhaltet.

Authentifizierung in einer Client-Server-Umgebung



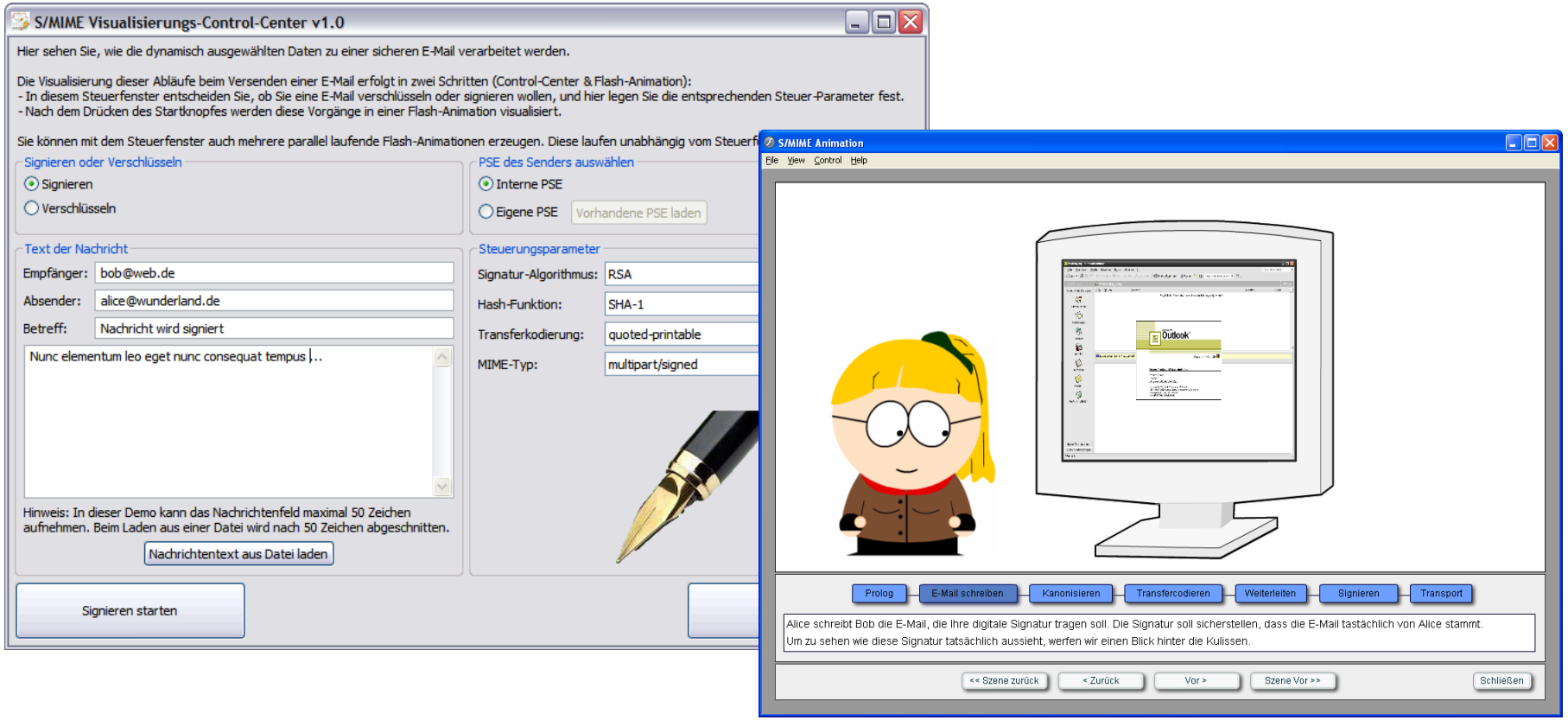
- Interaktive Demo für verschiedene Authentifizierungs-Verfahren
- Definierte Möglich-keiten des Angreifers
- Sie können in die Rolle eines Angreifers schlüpfen.
- **Lerneffekt**
  Nur die wechselseitige Authentifizierung ist sicher.

Menü in CT1: „Einzelverfahren“ \ „Protokolle“ \ „Authentisierungsverfahren im Netz“

Visualisierung von sicherer E-Mail per S/MIME

S/MIME-Visualisierung

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



Menü in CT1: „Einzelverfahren“ \ „Protokolle“ \ „Sichere E-Mail mit S/MIME…“

Signatur-Demo
Die Idee

Problematik

- Elektronische Dokumente können à priori nicht auf den Urheber überprüft werden.
  Dazu braucht man ein Verifizierungsmerkmal des Autors, dies kann z.B. eine Unterschrift sein.
- Ein rein elektronisches Dokument kann abgewandelt werden, ohne dass dies bemerkt wird.

Um diese Problematik zu umgehen, kann der Autor ein elektronisches Dokument digital signieren.

Funktionsweise des Signierens

- Der Urheber generiert aus dem Dokument einen Hashwert.
- Der Hashwert wird (bei Verwendung von RSA) mit dem privaten Schlüssel des Autors verschlüsselt.
- Den verschlüsselten Hashwert und die benutzte Hashfunktion stellt der Autor mit dem Dokument zur Verfügung.
- Ein Interessent, der die Integrität des Dokuments überprüfen möchte, kann nun mit dem öffentlichen Schlüssel den Hashwert des Dokuments aus dem verschlüsselten Hashwert zurückgewinnen.
- Den Hashwert kann er selbst gegenprüfen, indem er die vom Autor benutzte Hashfunktion erneut auf das Dokument anwendet. Sind die beiden Werte identisch, kann er sich sicher sein, dass das Dokument nicht verfälscht wurde.

Signatur-Demo
Die Implementierung in JCT

Im Menü unter:

„Visualisierungen“ \ „Signatur- Demo“

Der Algorithmus in der Anwendung

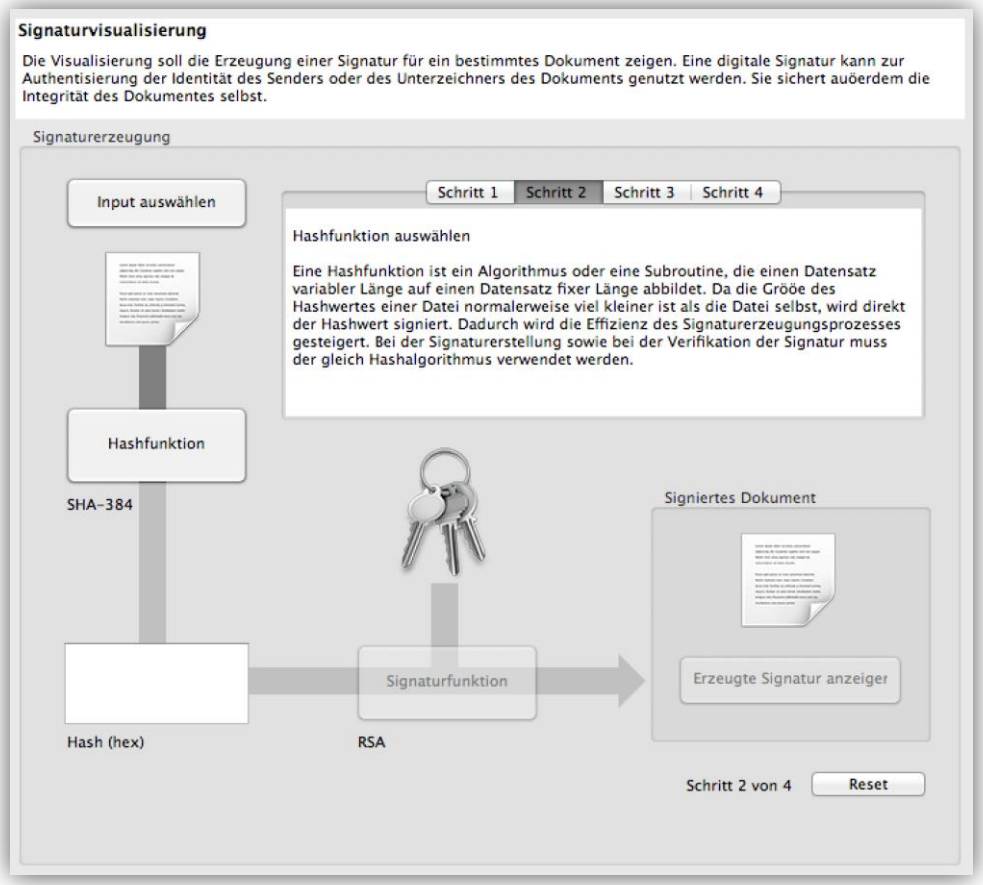
- In diesem Plugin gibt es die Möglichkeit, ein Dokument oder einen selbst eingegebenen Text zu signieren.
- Als Hashmethoden stehen die Funktionen MD5, SHA-1, SHA-256, SHA-384 und SHA-512 zur Verfügung.
- Anschließend kann, je nach gewählter Hashfunktion, als Signaturfunktion DSA, RSA, ECDSA oder RSA mit MFG1 benutzt werden.

Signatur-Demo
Ein Anwendungsbeispiel 1/2

Ein Dokument zu signieren ist nicht aufwändig und geht in zwei Schritten.

Erster Schritt

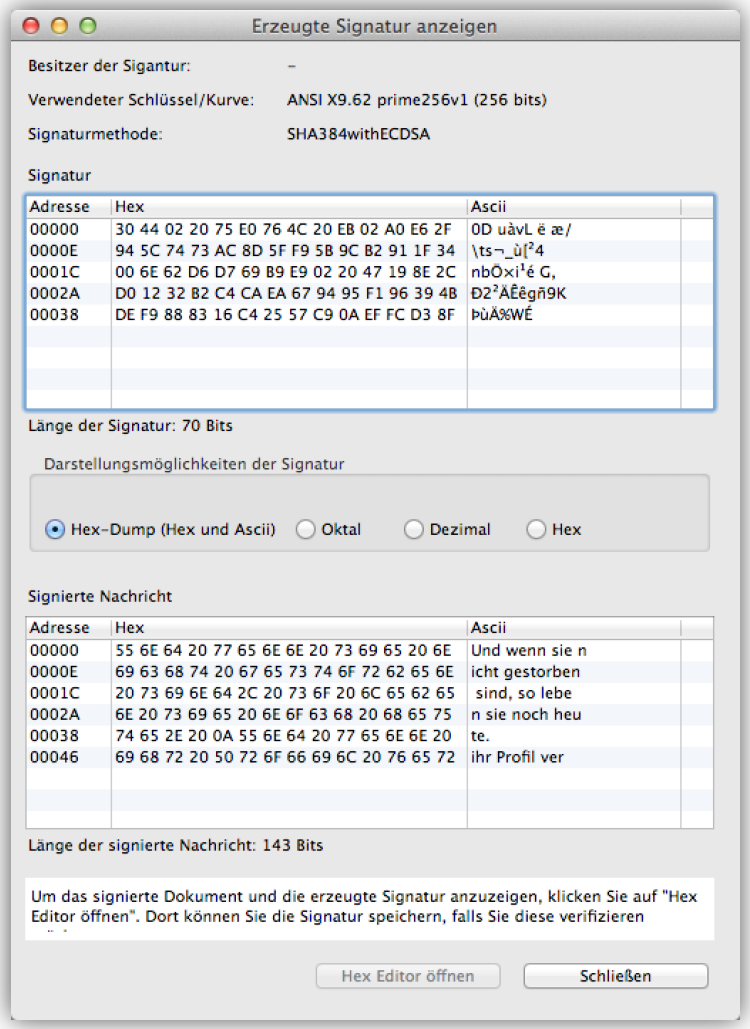
- Wähle das zu signierende Dokument über „Input auswählen“.
- Es erscheint ein Dialog mit „Datei öffnen“ oder „Direkte Eingabe“, um einen beliebigen Text direkt einzugeben.
- Anschließend muss eine „Hashfunktion“ ausgewählt werden.
- Der Hashwert wird dann erzeugt und unten angezeigt.



Signatur-Demo
Ein Anwendungsbeispiel 2/2

Zweiter Schritt

- Über „Signaturfunktion“ kann ein Verschlüsselungsalgorithmus ausgewählt werden, mit dem der Hashwert verschlüsselt wird.
- Wir wählen als Signaturmethode „ECDSA“ aus. Darunter muss dann noch aus dem JCT-Keystore ein Schlüssel für den Signierer (hier „Alice Whitehead“) ausgewählt werden.
- Durch Klick auf „Fertigstellen“ wird die Signatur erzeugt und kann anschließend über „Erzeugte Signatur anzeigen“ betrachtet und abgespeichert werden.



Fazit

- Die Integrität von elektronischen Dokumenten kann mit Hilfe einer Signatur überprüft werden.
- Krypto-Algorithmen helfen auch hier, um den Autor zu verifizieren.
- Wird ein Dokument verfälscht, so ändert sich damit auch dessen Hashwert.
- Da der Hashwert bei der Verifikation für jeden zugänglich sein muss, wird dieser verschlüsselt. Entschlüsseln kann man ihn nur mit dem „richtigen“ öffentlichen Schlüssel, also dem des angeblichen Signierers. Somit kann der Hashwert, obwohl er öffentlich verfügbar ist, nicht nachträglich geändert werden, ohne dass es bemerkt würde.

Brute-Force-Analyse

Implementierung in CrypTool 1

Brute-Force-Analyse

- Hier Brute-Force-Analyse unter der Annahme, dass ein Teil des Schlüssels bekannt ist.* Computer können allein entscheiden, ob die Entschlüsselung ein sinnvoller Klartext ist!

Beispiel: Analyse mit DES (ECB)

- Versuch, mittels Brute-Force den vollständigen Schlüssel zu finden, um damit dann den Geheimtext zu entschlüsseln (Annahme: Der Klartext ist ein Block aus 8 ASCII-Zeichen)

Schlüssel (Hex)	Geheimtext (Hex)
68ac78dd40bbefd*	66b9354452d29eb5
0123456789ab****	1f0dd05d8ed51583
98765432106*****	bcf9ebd1979ead6a
0000000000******	8cf42d40e004a1d4
000000000000****	0ed33fed7f46c585
abacadaba*******	d6d8641bc4fb2478
ddddddddddd******	a2e66d852e175f5c

* Wenn bei einem modernen symmetrischen Verfahren wie AES der gesamte Schlüssel zufällig und unbekannt ist, kann man den Geheimtext mit heute bekannten Mitteln nicht knacken. Laut den Dokumenten des Whistleblowers Edward Snowden speichert die NSA aber den gesamten Kommunikationsverkehr über vergangene Jahre, so dass sie es auch entschlüsseln kann, wenn sie später mal in den Besitz des Schlüssels kommt. Vergleiche die Notwendigkeit von Perfect Forward Secrecy bei SSL/TLS.

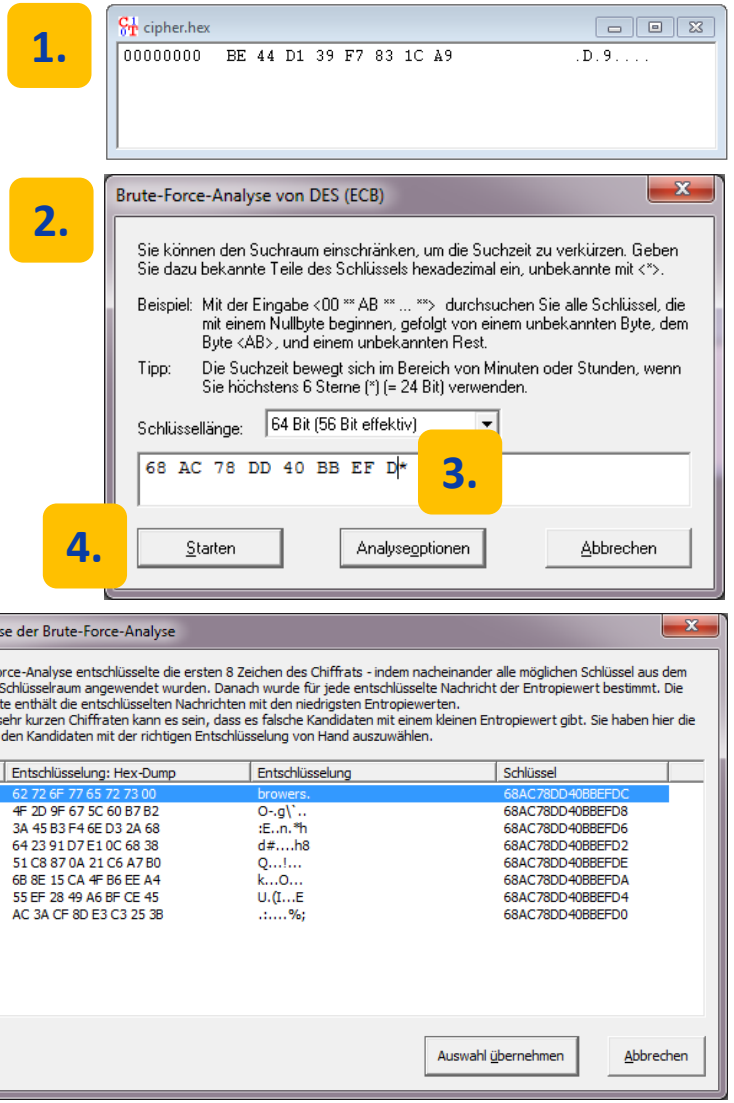
Brute-Force-Analyse

Implementierung in CrypTool 1

1. **Eingabe des verschlüsselten Textes**
   - Hex-codierte Nachrichten können in CT 1 über das Menü: „Ansicht“ \ „Als HexDump anzeigen“ eingegeben werden.
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 korrekte Entschlüsselung hin (im Beispiel hat der richtige Kandidat die kleinste Entropie, was aber nicht immer der Fall sein muss).**

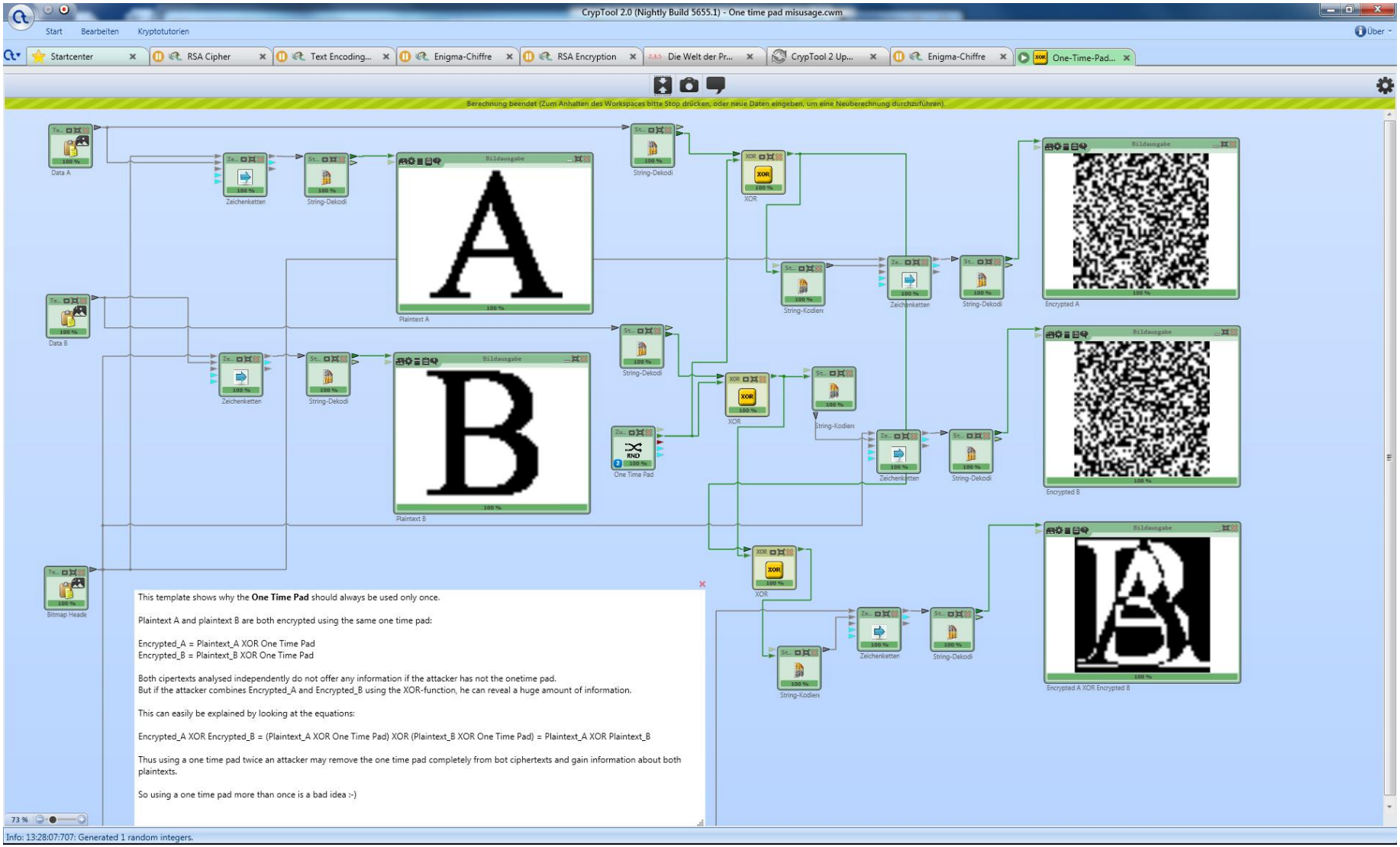
Menü in CrypTool 1

- „Analyse“ \ „Symmetrische Verfahren (modern)“ \ „DES (ECB)“



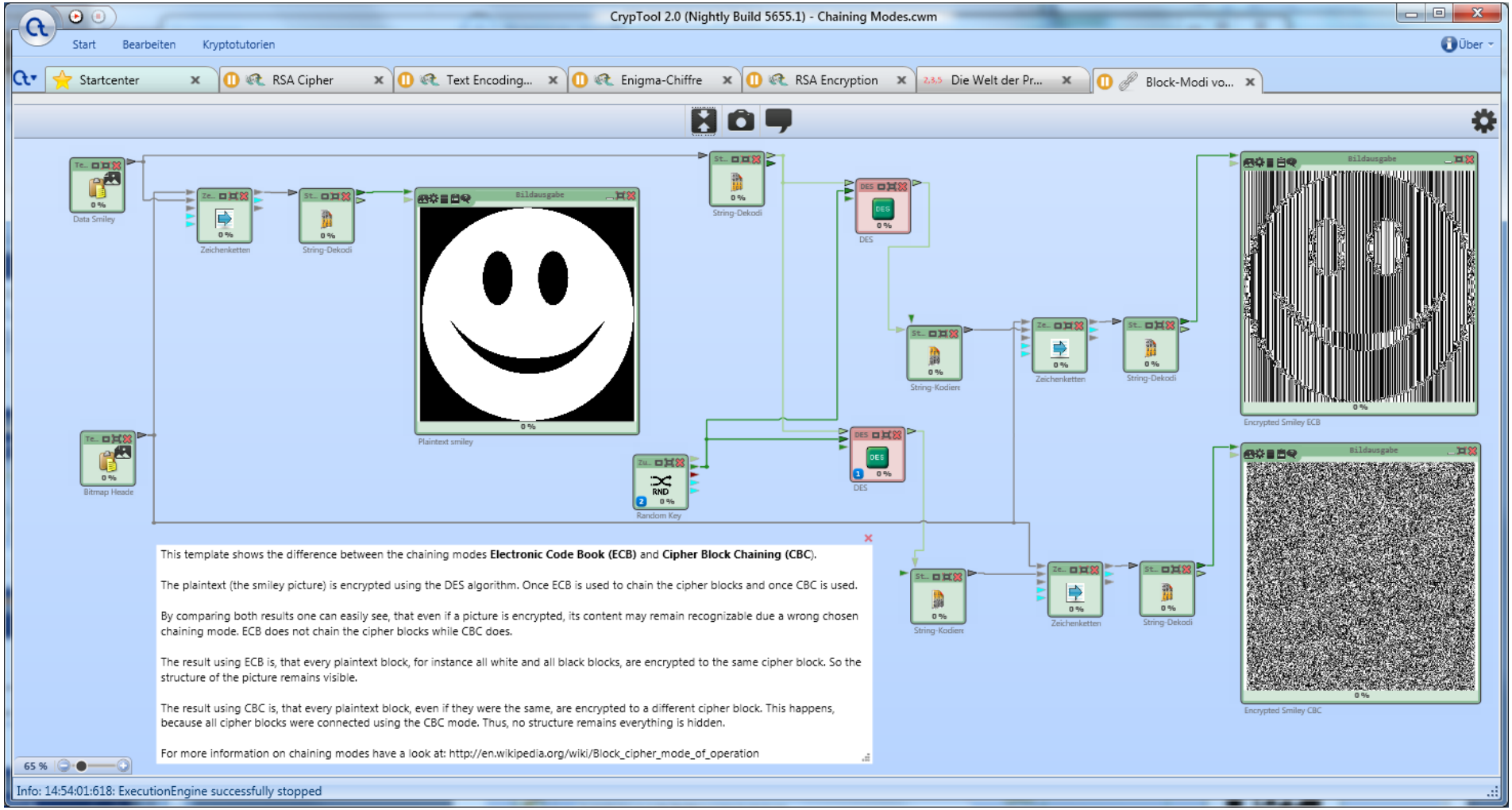
Ein One-Time-Pad bitte nur genau einmal benutzen!

Verschlüsselung nicht richtig parametrisiert (Demo in CrypTool 2)



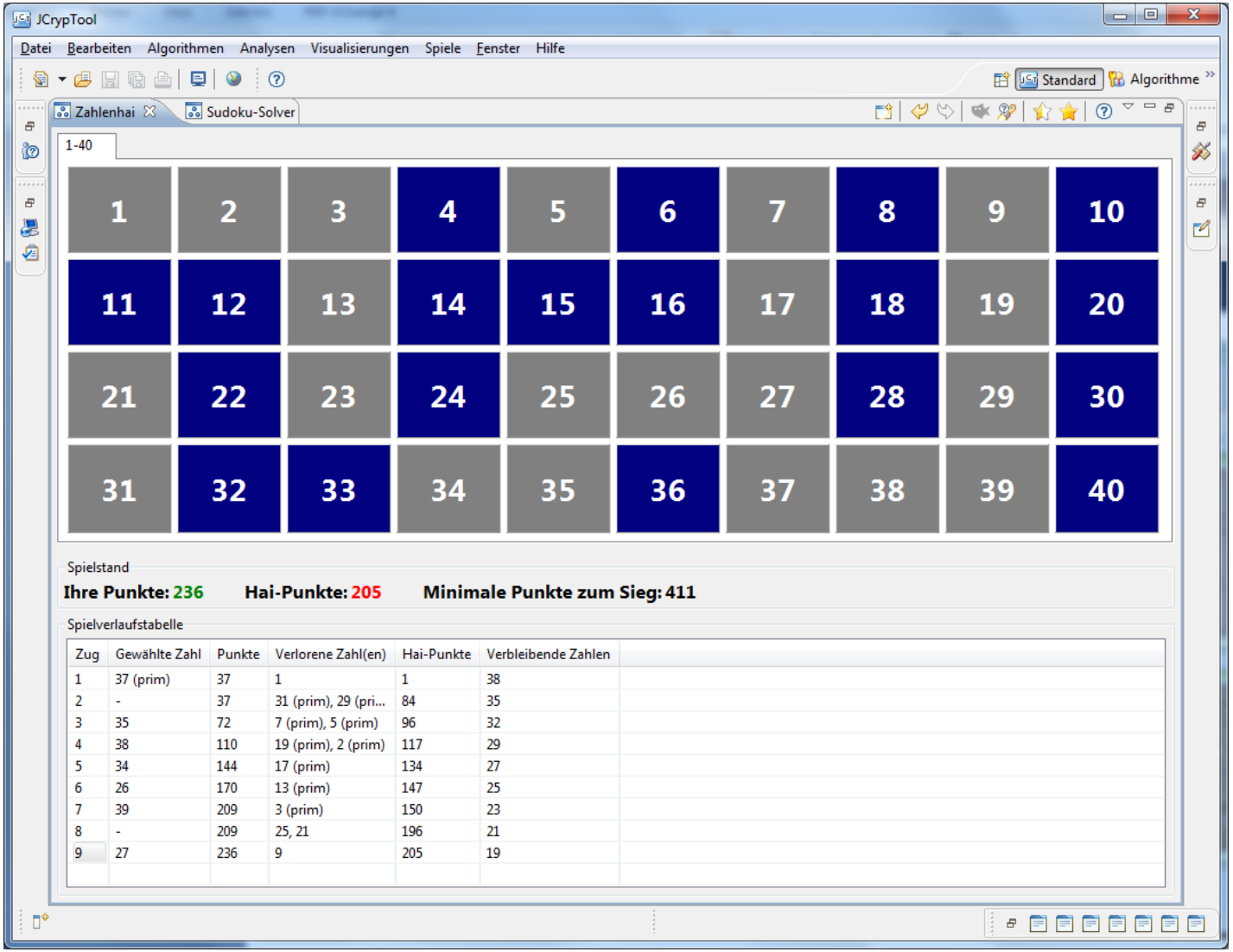
ECB- und CBC-Block-Modi von DES und AES als Bild

Verschlüsselung nicht richtig parametrisiert (Chaining) (Demo in CrypTool 2)



Lernspiele: Hier der Zahlenhai

Idee war ein didaktisches Mittel zu Primzahlen in Klasse 5 und 6 (Demo in JCT)



Zug	Gewählte Zahl	Punkte	Verlorene Zahl(en)	Hai-Punkte	Verbleibende Zahlen
1	37 (prim)	37	1	1	38
2	-	37	31 (prim), 29 (pri...	84	35
3	35	72	7 (prim), 5 (prim)	96	32
4	38	110	19 (prim), 2 (prim)	117	29
5	34	144	17 (prim)	134	27
6	26	170	13 (prim)	147	25
7	39	209	3 (prim)	150	23
8	-	209	25, 21	196	21
9	27	236	9	205	19

Übersicht

Inhalt der Präsentation

CrypTool – Bitte um Mitwirkung

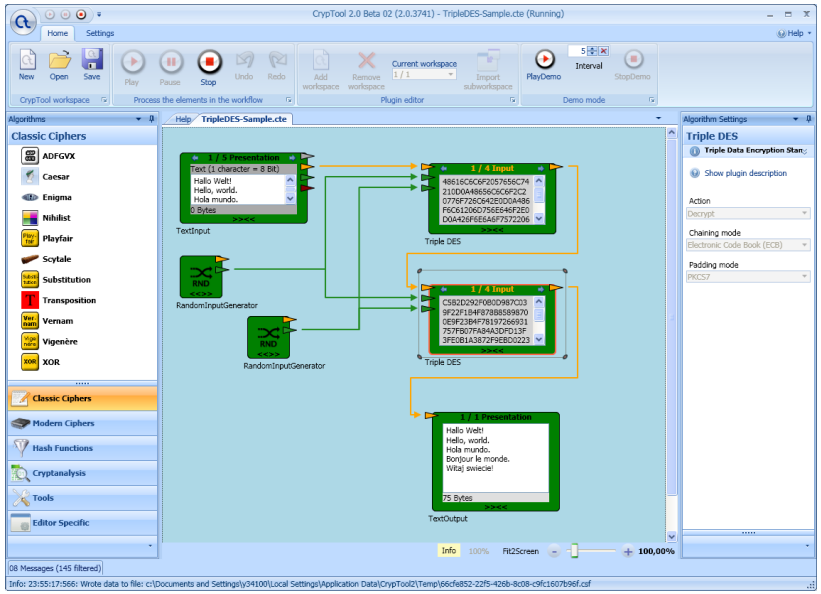
Wir freuen uns über jede weitere Mitarbeit

- 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 CrypTool 1 (CT1), und
  - in den **aktuellen Projekten** (bevorzugt):
    - C#-Projekt: „CrypTool 2“ = CT2 (http://www.cryptool.org/de/ct2-machmit-de)
    - Java-Projekt: „JCrypTool“ = JCT (http://www.cryptool.org/de/jct-machmit-de)
    - Browser-Projekt: „CrypTool-Online“ = CTO (http://www.cryptool-online.org)
- Insbesondere Lehrstühle, die CrypTool zur Ausbildung verwenden, sind herzlich eingeladen, zur Weiterentwicklung beizutragen.
- Beispiele offener Aufgaben finden sich auf den entsprechenden Entwickler-Seiten (nur englisch):
  - CT2: Siehe die Vorschlagsliste möglicher Aufgaben: https://www.cryptool.org/trac/CrypTool2/wiki/StudentTasksProposals
  - JCT: Siehe die Liste Projektideen: http://sourceforge.net/apps/mediawiki/jcryptool/index.php?title=CurrentDevelopment
- Bei signifikanten Beiträge können die Autoren gerne namentlich erwähnt werden (in der Hilfe, im Readme, im About-Dialog und auf der Webseite).
- Derzeit wird das CrypTool-Programmpaket knapp 10.000 mal pro Monat von der CrypTool-Webseite herunter geladen (davon macht die englische Version etwas mehr als 50 % aus).

Aktuelle Entwicklung (1)

Aktiv weiter entwickelt werden die zwei Versionen CT2 und JCT

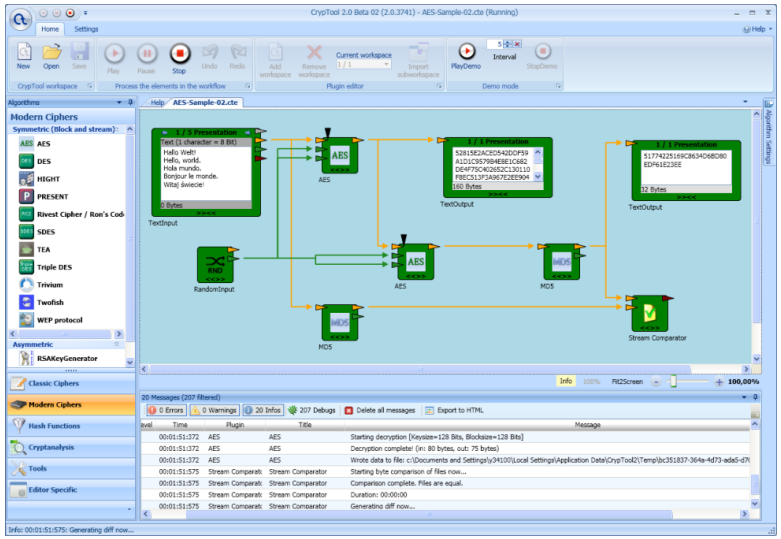
1. JCT: JCrypTool in Java
   - Plattform-unabhängiger Nachfolger von CT1: Läuft auf Windows, MacOS und Linux
   - Download von: http://www.cryptool.org/de/jct-downloads-de/jct-downloads-weekly-de
   - Release Candidate 7 ist verfügbar seit Dez. 2013 (seitdem gibt es wöchentlich einen neuen Weekly Build).
2. CT2: CrypTool mit C#
   - Direkter Nachfolger von CT1: Erlaubt visuelle Programmierung, …
   - Download von: http://www.cryptool.org/de/ct2-download-de
   - Release CT 2.0 ist verfügbar seit Aug. 2014 (Updates für CT 2.1 gibt es seitdem täglich als Nightly Builds).

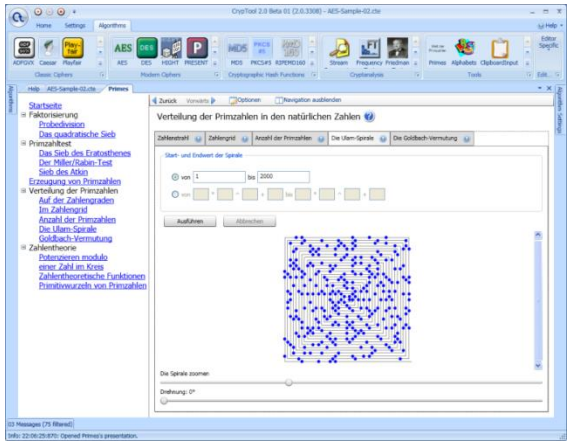


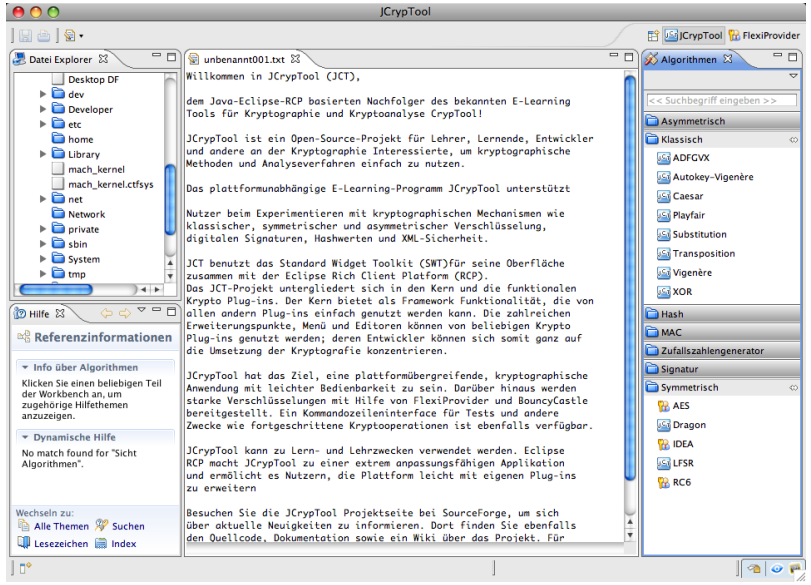
CrypTool 2 (CT2)

JCrypTool (JCT)

Aktuelle Entwicklung (2)







CrypTool 2 (CT2)

JCrypTool (JCT)

CrypTool als Framework für eigene Arbeiten

Angebot / Vorteile

- 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-, Bachelor-, Master- und Doktorarbeiten „verschwindet“ nicht, sondern wird weitergepflegt. Hilfe + Feedback beim Erstellen.
- Open-Source: Download aller Sourcen und Binaries der Release- und Beta-Versionen

Eingesetzte Entwicklungsumgebungen für:

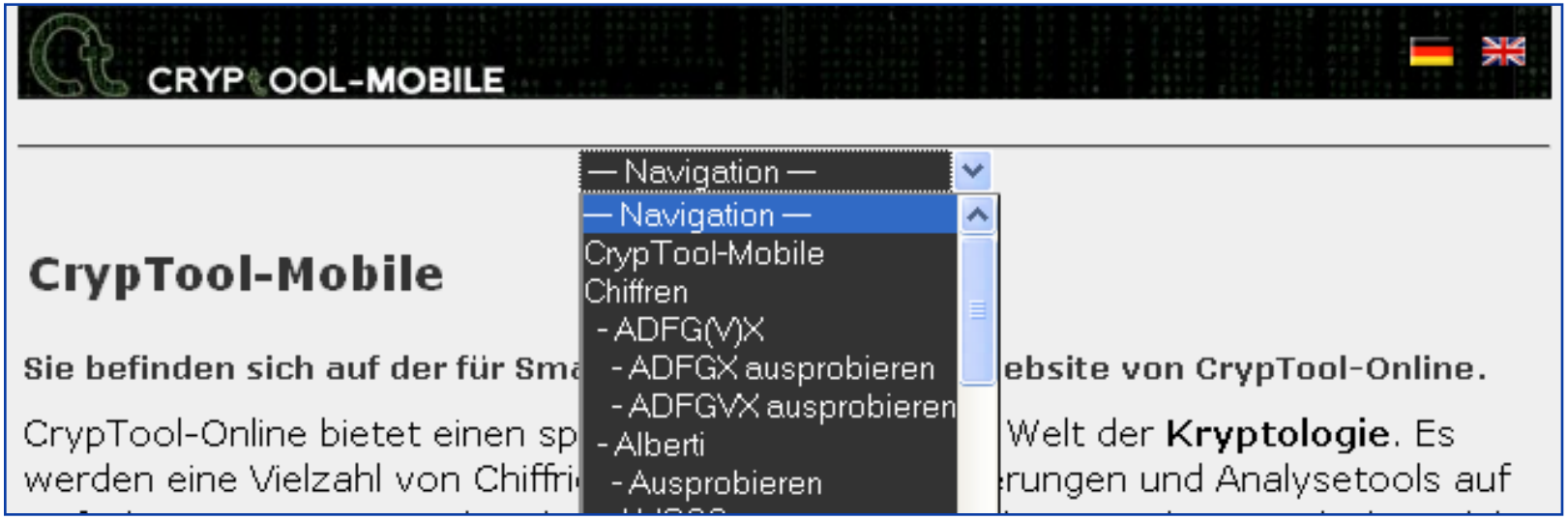
- **CrypTool 1.4.31 (CT1):**
  Visual C++ .NET in Microsoft Visual Studio 2015 Community Edition (kostenlos)
  C++, Perl, .NET >= 4.0, und das Subversion Source-Code-Management
- **CrypTool 2.0 (CT2):**
  Visual Studio 2015 Community Edition (kostenlos)
  C#, WPF, und das Subversion Source-Code-Management
- **JCrypTool (JCT):**
  Eclipse 4.4, RCP, SWT (kostenlos)
  Java 1.8, und das GitHub Source-Code-Management

www.cryptool-online.org

Mitglieder in der Familie der CrypTool-Webseiten:

- **CrypTool-Portal und CT1**
- **CT2-Entwicklerseite**
- **JCT-Entwicklerseite**
- **CrypTool-Online + CrypTool-Mobil** (Einstieg in Kryptologie und erstes Ausprobieren direkt im Browser, und mit dem Smartphone)
- **CryptoPortal** für Lehrer
- **MysteryTwister C3** (MTC3) ist ein internationaler Krypto-Wettbewerb.

m.cryptool.org (CrypTool-Online = CrypTool-Mobil)

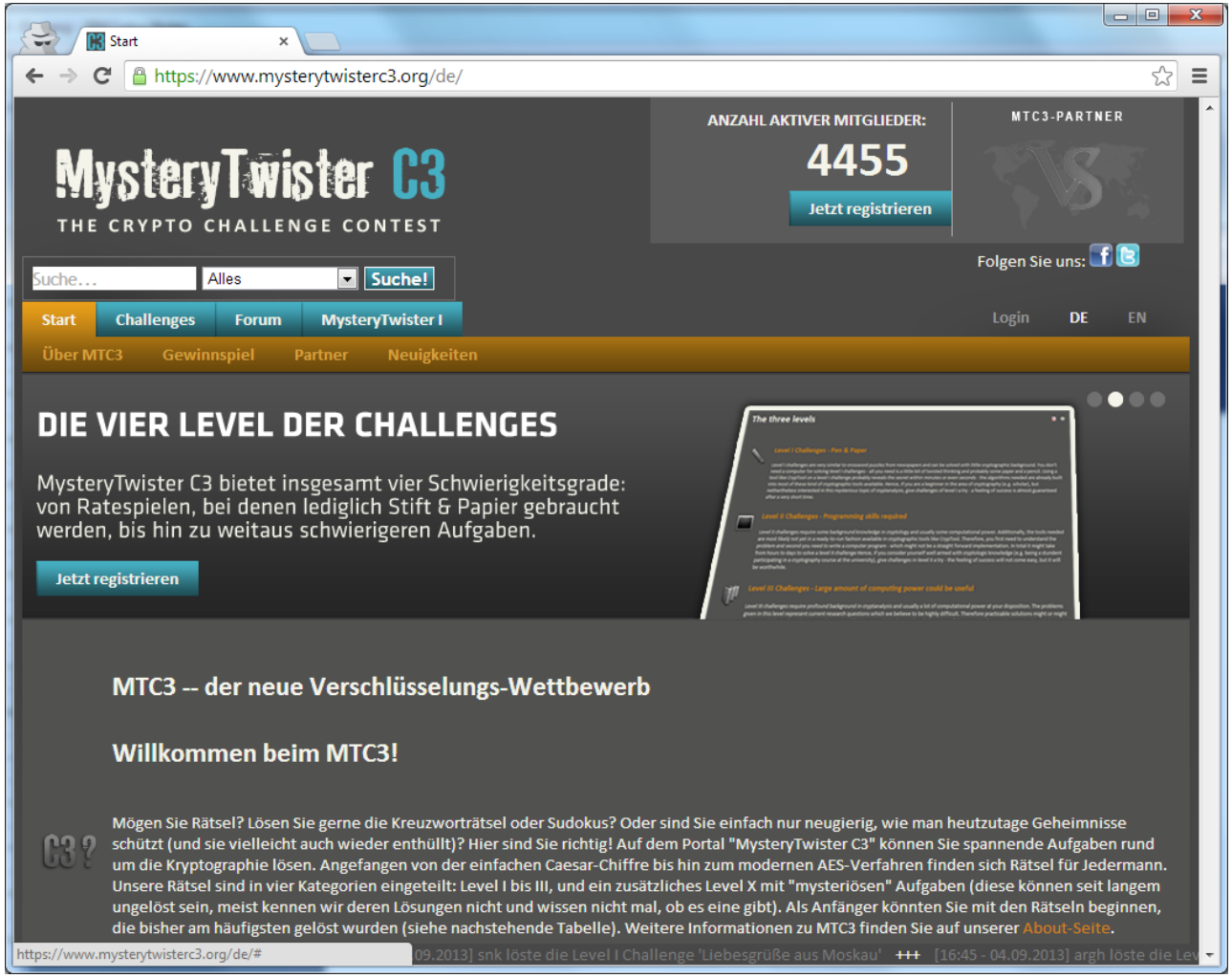


Einstieg in Kryptologie und erstes Ausprobieren mit dem Smartphone.

Basierend auf neuester Web-Technologie (AngularJS 2, Bootstrap, JavaScript, Joomla) entwickelt das Projekt gerade mit Studenten und Web-Experten die neue Version, die bis Ende 2016 life gehen soll.

www.cryptoportal.org (für Lehrer)

www.mysterytwisterc3.org (für Rätselfreunde)



- Ein Wettbewerb für Anfänger und Fortgeschrittene
- Über 7000 Mitglieder
- Über 200 Krypto-Rätsel
- Ein moderiertes Forum
- Eine Hall-of-Fame
- Ein enges Rennen an der Spitze
- Etwas für Dich?

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

CrypTool – Fazit

DAS E-Learning-Programm für Kryptologie

- Seit über 15 Jahren ein lebendiges Open-Source-Projekt
- Mehr als 1.000.000 Downloads
- Weltweiter Einsatz in Schulen und Universitäten sowie Firmen und Behörden
- Umfangreiche Online-Hilfe und Dokumentation
- Frei verfügbar, kostenlos, mehrsprachig

Weitere Aktivitäten im CrypTool-Projekt:

- Internationaler Krypto-Wettbewerb MTC3 mit Rätseln für alle
- Schülerkrypto-Kurse

Kontaktadresse

Prof. Bernhard Esslinger
(Leiter des CrypTool-Projekts, Universität Siegen)

bernhard.esslinger@uni-siegen.de

www.cryptool.org