

Kryptologie mit JCrypTool (JCT)

Praktische Einführung in
Kryptografie und
Kryptoanalyse

*Prof. Bernhard Esslinger
und das CrypTool-Team*

24. November 2020





Einführung in das JCrypTool-Programm

2

Anwendungsbeispiele

22

Möglichkeiten zur Mitwirkung

87

Einführung in das Programm JCrypTool (JCT)

Überblick



JCrypTool – Kryptografische E-Learning-Plattform	Seite 4
Was ist Kryptologie?	Seite 5
Die Standard-Perspektive von JCT	Seite 6
Typische Benutzung von JCT in der Standard-Perspektive	Seite 7
Die Algorithmen-Perspektive von JCT	Seite 9
Der Krypto-Explorer	Seite 10
Algorithmen in der Krypto-Explorer-View	Seite 11
Die Analysetools	Seite 13
Visualisierungen & Spiele	Seite 14
Allgemeine Bedienungshinweise	Seite 15
Benutzervorgaben	Seite 20
Kommandozeilen-Parameter	Seite 21

JCrypTool – Kryptografische E-Learning-Plattform

Das Projekt

Übersicht

- JCrypTool – im folgenden JCT abgekürzt – ist eine kostenlose E-Learning-Software für klassische und moderne Kryptologie.
- JCT ist plattformunabhängig, d.h. es läuft unter **Windows, MacOS und Linux**. JCT hat eine moderne Pure-Plugin-Architektur.
- JCT ist im Open-Source-Projekt CrypTool (www.cryptool.org) entstanden.
- Das CrypTool-Projekt hat sich zur Aufgabe gemacht, Kryptografie und Kryptoanalyse einfach, verständlich und trotzdem auf wissenschaftlichem Niveau zu erklären und zu visualisieren.
- Zielgruppe von JCT sind hauptsächlich:
 - Schüler und Studenten
 - Lehrer und Dozenten/Professoren
 - Mitarbeiter in Awareness-Kampagnen
 - Kryptologie-Begeisterte.
- Da JCT Open-Source-Software ist, kann jeder eigene Erweiterungen schreiben und bereits entwickelte Komponenten wieder benutzen.
- Zu JCT trugen über 100 Menschen aus verschiedenen Ländern bei.



Der JCT-Ladebildschirm (Splash-Screen)

Was ist Kryptologie?

Worum geht es in JCrypTool



Der Begriff Kryptologie

- Aus dem Griechischen von „kryptós“ („versteckt, verborgen, geheim“) und „lógos“ („Wort“, in diesem Kontext steht es für „Lehre“).
- Kryptologie beschäftigt sich im Allgemeinen mit Verfahren und Protokollen, die Informationen nur Befugten verfügbar machen. Kryptologie besteht aus 2 Teilbereichen.

Der Teilbereich Kryptografie

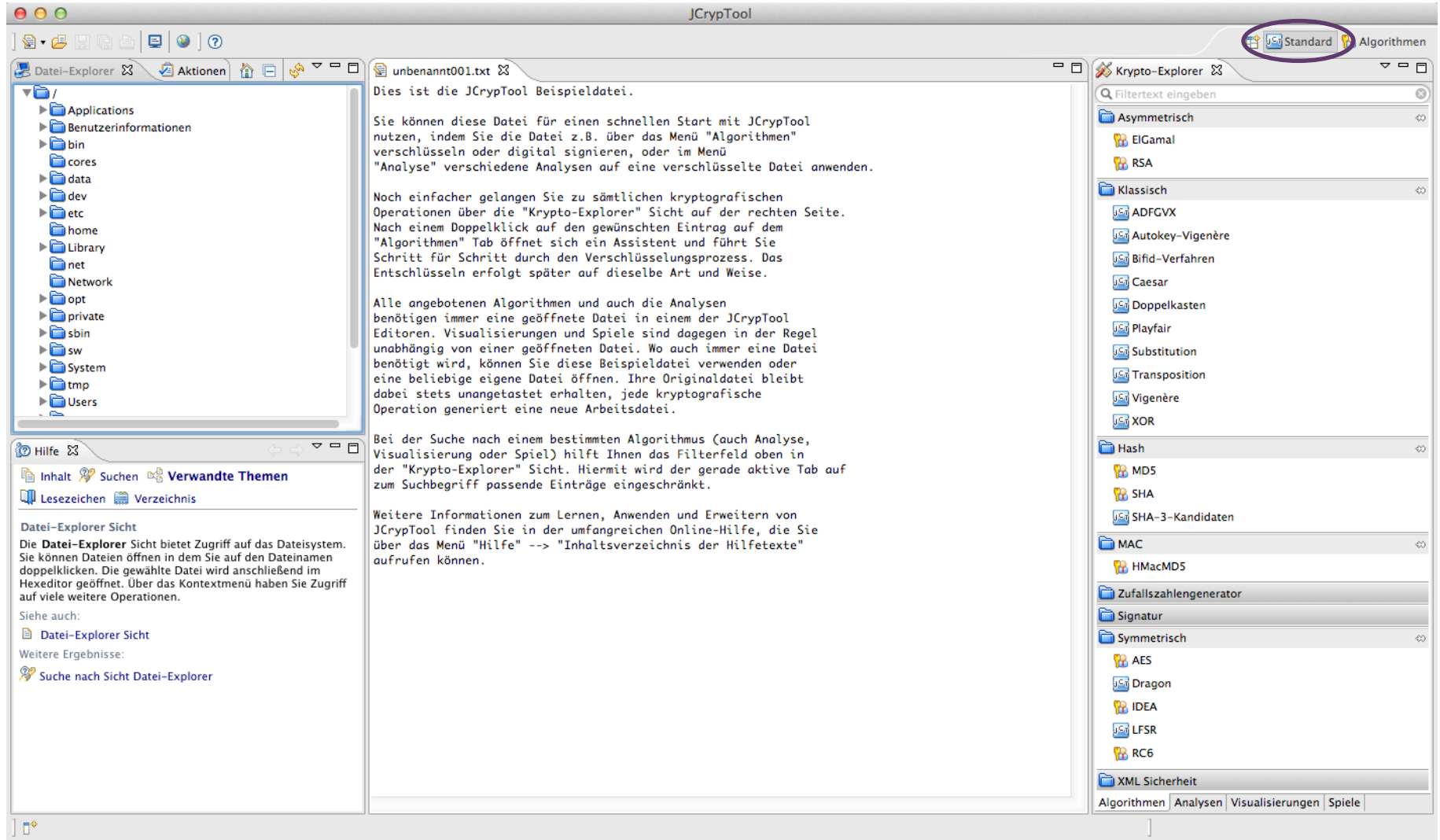
- Wissenschaft, die Verschlüsselungssysteme bereitstellt, um Sicherheit und Vertraulichkeit beim Speichern und beim Informationsaustausch (z.B. zwischen Computern) zu gewährleisten.
- Neben Verschlüsselung sind heutzutage auch sicherer Schlüsselaustausch und Integritätsprüfung wichtig, um z.B. Online-Banking, elektronische Wahlen oder elektronisches Geld zu ermöglichen.
- Die Sicherheit vieler Verfahren beruht auf (ungelösten/schwierigen) mathematischen Problemen.

Der Teilbereich Kryptoanalyse

- Kryptoanalyse ist das Gegenstück zur Kryptografie und liefert Theorie und Verfahren zum Testen und Brechen kryptografischer Verfahren.
- Es wird z.B. versucht, aus dem Geheimtext – dem Ergebnis einer Verschlüsselung – Informationen über den Klartext oder den benutzten Schlüssel zurück zu gewinnen.
- Dazu werden Mathematik und Informatik benutzt (z.B. statistische Tests, Entropie, Häufigkeits- und Strukturanalysen, Komplexitätsbetrachtungen, Brute-Force-Algorithmen und vieles mehr).

Die Standard-Perspektive von JCT

... Dokumenten-orientiert



Typische Benutzung von JCT in der Standard-Perspektive

... Auswahl eines Verfahrens im Hauptmenü Algorithmen

The screenshot displays the JCT application interface. The main menu bar includes 'Datei', 'Bearbeiten', 'Algorithmen', 'Analysen', 'Visualisierungen', 'Spiele', 'Fenster', and 'Hilfe'. The 'Algorithmen' menu is open, showing a list of categories: 'Asymmetrisch', 'Hash', 'Klassisch', 'MAC', 'Signatur', 'Symmetrisch', and 'Zufallszahlengenerator'. The 'Symmetrisch' category is selected, revealing a sub-menu with the following algorithms: AES, ARC4/Spritz, Dragon, IDEA, LFSR, and RC6. The 'Krypto-Explorer' sidebar on the right is also visible, showing a search filter and a list of categories: 'Asymmetrisch', 'Hash', 'Klassisch', 'MAC', 'Signatur', 'Symmetrisch', and 'Zufallszahlengenerator'. The 'Symmetrisch' category is expanded, showing the same list of algorithms as the main menu. The main window displays a text editor with a file named 'unbenannt001.txt' and a help window titled 'Krypto-Explorer Sicht'.

Alle angebotenen Algorithmen und auch die Analysen benötigen immer eine geöffnete Datei in einem der JCrypTool Editoren. Visualisierungen und Spiele sind dagegen in der Regel unabhängig von einer geöffneten Datei. Wo auch immer eine Datei benötigt wird, können Sie diese Beispieldatei verwenden oder eine beliebige eigene Datei öffnen. Ihre Originaldatei bleibt dabei stets unangetastet erhalten, jede kryptografische Operation generiert eine neue Arbeitsdatei.

Bei der Suche nach einem bestimmten Algorithmus (auch Analyse, Visualisierung oder Spiel) hilft Ihnen das Filterfeld oben in der "Krypto-Explorer" Sicht. Hiermit wird der gerade aktive Tab auf zum Suchbegriff passende Einträge eingeschränkt.

Weitere Informationen zum Lernen, Anwenden und Erweitern von JCrypTool finden Sie in der umfangreichen Online-Hilfe, die Sie über das Menü "Hilfe" --> "Inhaltsverzeichnis der Hilfetexte" aufrufen können.

Typische Benutzung von JCT in der Standard-Perspektive

... Auswahl eines Verfahrens im Hauptmenü Visualisierungen

The screenshot displays the JCT application interface. The main menu bar includes 'Datei', 'Bearbeiten', 'Algorithmen', 'Analysen', 'Visualisierungen', 'Spiele', 'Fenster', and 'Hilfe'. The 'Visualisierungen' menu is open, showing a list of cryptographic algorithms. The 'Kleptographie' option is highlighted. The 'Krypto-Explorer' window is open, showing a search bar and a list of visualization options. The 'Datei-Explorer' window shows the file system structure, and the 'Texteditor' window is also visible.

Visualisierungen

- Ameisenkolonie-Optimierung (ACO)
- Android-Mustersperre (AUP)
- ARC4 / Spritz
- Chinesischer Restsatz (CRT)
- Diffie-Hellman Schlüsselaustausch (EC)
- ElGamal-Kryptosystem
- Elliptische-Kurven-Berechnungen
- Erweiterter Euklid / Wechselwegnahme
- Erweitertes RSA-Kryptosystem
- Grille
- Hash-Sensitivität
- Homomorphe Verschlüsselung (HE)
- Huffman-Kodierung
- Innere Zustände im Data Encryption Standard (DES)
- Kleptographie**
- McEliece-Kryptosystem
- Mehrparteien-Schlüsselaustausch (BD II)
- Merkle-Hellman Rucksack-Verschlüsselung
- Merkle-Signaturen (XMSS^MT)
- Multivariate Kryptografie
- RSA-Kryptosystem
- Shamirs Secret Sharing
- Shanks Babystep-Giantstep
- Signatur-Demo
- Signatur-Verifikation
- Simple Power Analysis / Square and Multiply
- SPHINCS+-Signatur
- SPHINCS-Signatur
- SSL/TLS-Handshake
- Verifiable-Secret-Sharing
- Winternitz Einmal-Signatur (WOTS / WOTS+)

Krypto-Explorer

Filtertext eingeben

Visualisierungen

- Huffman-Kodierung
- Innere Zustände im Data Encryption Standard (DES)
- Kleptographie
- McEliece-Kryptosystem
- Mehrparteien-Schlüsselaustausch (BD II)
- Merkle-Hellman Rucksack-Verschlüsselung
- Merkle-Signaturen (XMSS^MT)
- Multivariate Kryptografie
- RSA-Kryptosystem
- SPHINCS+-Signatur
- SPHINCS-Signatur
- SSL/TLS-Handshake
- Shamirs Secret Sharing
- Shanks Babystep-Giantstep
- Signatur-Demo
- Signatur-Verifikation
- Simple Power Analysis / Square and Multiply
- Verifiable-Secret-Sharing
- Winternitz Einmal-Signatur (WOTS / WOTS+)
- Zero-Knowledge: Feige-Fiat-Shamir
- Zero-Knowledge: Fiat-Shamir
- Zero-Knowledge: Graphenisomorphie
- Zero-Knowledge: Magische Tür

Die Algorithmen-Perspektive von JCT

... Funktions-orientiert

The screenshot displays the JCryptTool application window. The title bar reads 'JCryptTool'. The interface is divided into several panes:

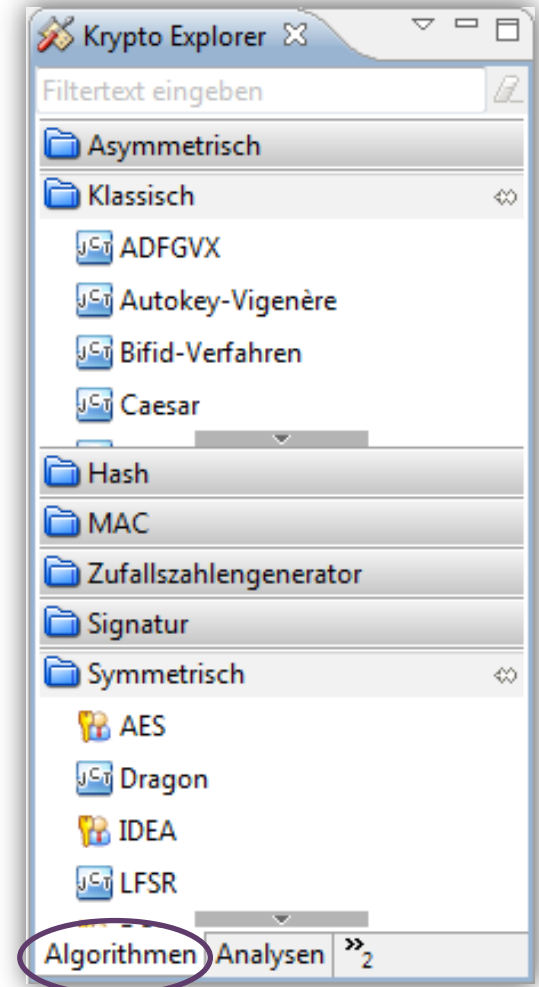
- Top Left:** 'Schlüsselspeicher' (Keystore) showing a tree structure under 'Alice Whitehat' with entries for keys and key pairs.
- Top Center:** A text editor window titled 'unbenannt001.txt' containing instructions in German about using the application and navigating to the 'Algorithmen' tab.
- Top Right:** A pane titled 'Algorithmen' (Algorithms) with a red circle around the title. It lists various cryptographic algorithms categorized into 'Asymmetrische Blockchiffren', 'Authentifizierungscodes', 'Blockchiffren', 'Hash-Funktionen', and 'Hybride Verschlüsselung'. 'MD5' is currently selected.
- Bottom Left:** A 'Hilfe' (Help) pane with sections for 'Hash-Funktionen' and 'Operationen'.
- Bottom Center:** An 'Operationen' (Operations) pane showing a list of actions, with '(AES) no name Created on: Thu Apr 11 00:06:16 CEST 2013' selected.

Der Krypto-Explorer

In der Standard-Perspektive von JCT

Funktionalität

- In der Standard-Perspektive von JCT befindet sich auf der rechten Seite der Reiter „Krypto-Explorer“. Im Krypto-Explorer werden die Funktionen von JCT dargestellt.
- Alle hier enthaltenen Funktionen lassen sich auch über die Menüs in der Menüleiste anwählen.
- Der Explorer ist – wie die Menüs – gegliedert in
 - Algorithmen
 - Analysen
 - Visualisierungen
 - Spiele
- Algorithmen und Analysen werden normalerweise auf das im Editor angezeigte Dokument angewendet; und der berechnete Output wird in einem neuen Editorfenster angezeigt.
- Visualisierungen und Spiele starten normalerweise unabhängig von dem im Editor angezeigten Dokument.



Algorithmen in der Krypto-Explorer-View

Aufteilung 1/2

Klassische Verfahren

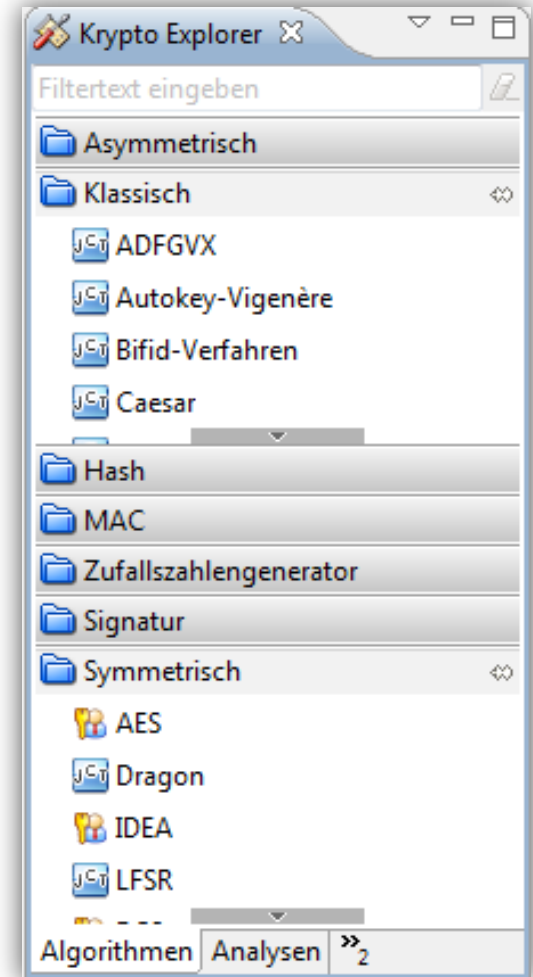
- In diese Kategorie sind Verfahren einsortiert, die etwa bis zum ersten Weltkrieg zur Verschlüsselung eingesetzt wurden. Viele dieser Verfahren sind durch Häufigkeitsanalysen zu knacken. Die allermeisten sind nicht mehr sicher.

Symmetrische Verfahren

- Dies sind moderne Verfahren, bei denen Sender und Empfänger den gleichen Schlüssel besitzen müssen.
- Problematik dieser Verfahren:
Der Schlüssel muss auf sicherem Wege unter den relevanten Kommunikationsteilnehmern verteilt werden.

Asymmetrische Verfahren

- Dies sind moderne Verfahren, bei denen jeder Teilnehmer ein Paar von Schlüsseln besitzt – jedes Paar besteht aus einem privaten und einem öffentlichen Schlüssel.
- Der Sender **verschlüsselt** dann mit dem öffentlichen Schlüssel des Empfängers; der Empfänger **entschlüsselt** mit seinem privaten Schlüssel.



Algorithmen in der Krypto-Explorer-View

Aufteilung 2/2

Hash & MAC

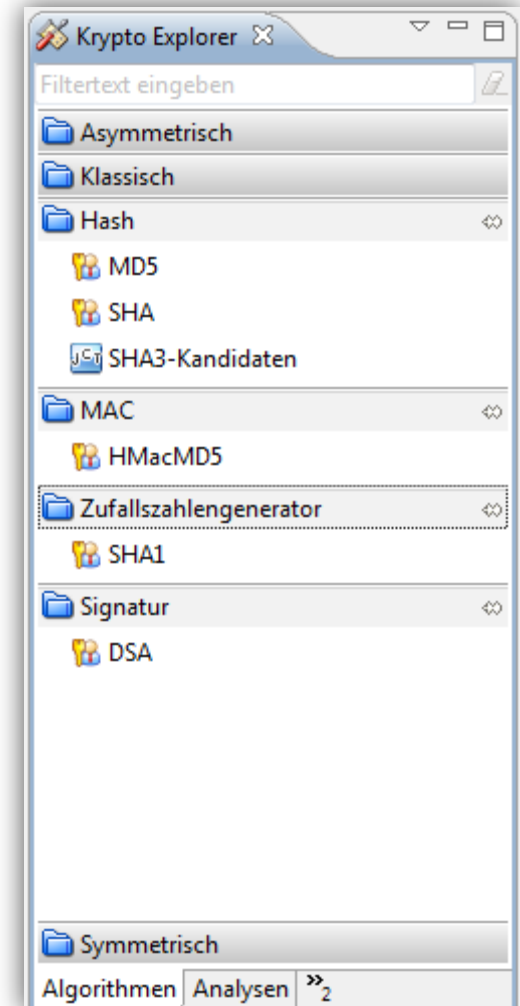
- Hashfunktionen bilden Daten beliebiger Länge auf einen Hashwert ab. Dieser Hashwert ist den Daten möglichst eindeutig zugewiesen. Er hat eine begrenzte feste Länge, die normalerweise viel kleiner ist als die Länge der Daten (analog einem Fingerabdruck).
- Hashwerte können benutzt werden, um Veränderungen an Dokumenten festzustellen (Integrität). Hashwerte werden auch in Datenbanken gespeichert, um Passwörter abzugleichen.

Signaturen

- Signier-Algorithmen dienen dazu, Nachrichten und Dokumente zu signieren.
- Mit einer Signatur kann die Integrität – also die Eigenschaft, ob ein Dokument unverändert ist – überprüft werden.

Zufallszahlengeneratoren

- Zufallszahlen spielen eine große Rolle in der Kryptografie, weshalb hier auch Funktionen zum Generieren von (pseudo-zufälligen) Zahlenfolgen implementiert sind.

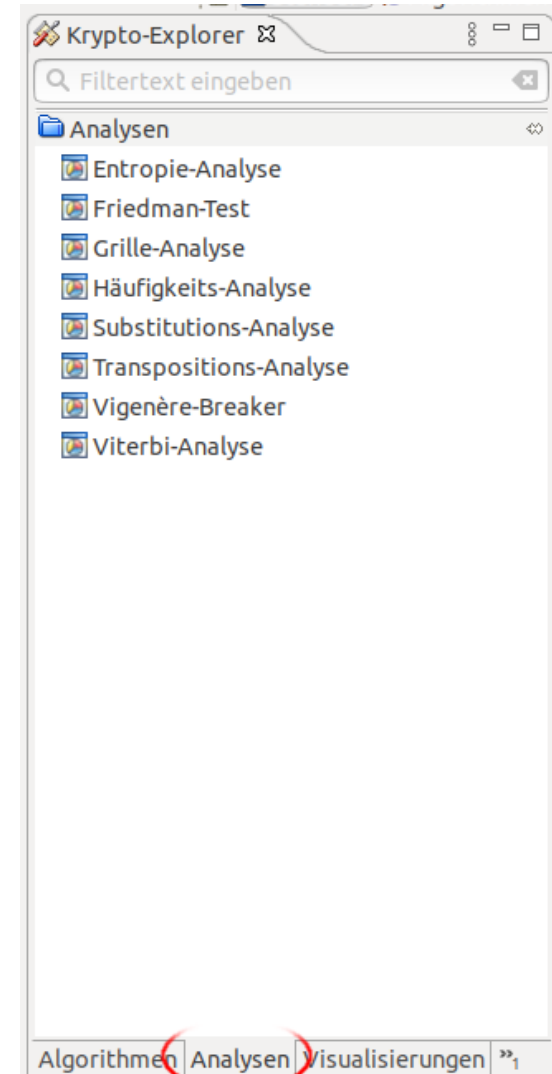


Die Analysetools

Im Krypto-Explorer

Analyse-Algorithmen

- Die in diesem Reiter des Krypto-Explorers aufgelisteten Analyse-Tools eignen sich, um den Geheimtext zu analysieren, um eventuelle Regelmäßigkeiten (Muster) festzustellen und damit auf den Klartext oder das Passwort (Schlüssel) zu schließen.
- Diese Algorithmen werden ebenfalls auf das aktuell geöffnete Dokument im Editor angewandt.
- Es sind unterschiedliche Analysen möglich, z.B. eine *Transpositions-Analyse*, mit der ein Geheimtext, der zeilen- oder spaltenweise transponiert wurde, wieder in den Klartext überführt werden kann.
- Mit einer *Häufigkeitsanalyse* lässt sich die Häufigkeit einzelner Buchstaben oder Buchstabenpaare in einem Text bestimmen. Da die Buchstaben in einer natürlichen Sprache verschieden häufig vorkommen, lassen sich so Muster und Wiederholungen erkennen und erste Ideen für den Klartext ableiten.



Visualisierungen & Spiele

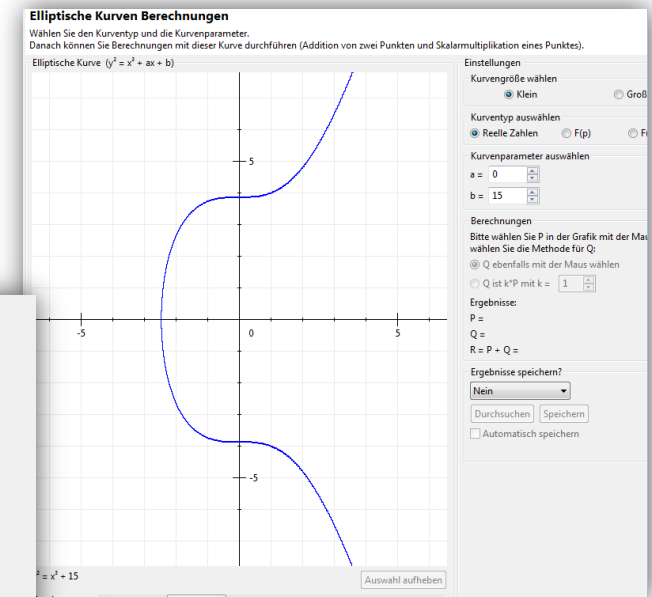
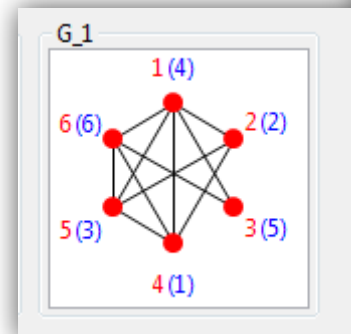
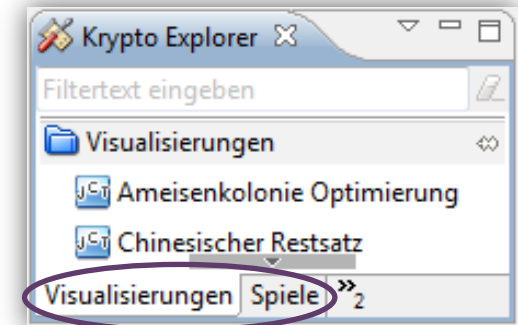
Im Krypto-Explorer

Visualisierungen

- Die Visualisierungen sind im Krypto-Explorer unter dem Reiter "Visualisierungen", oder direkt im Menü "Visualisierungen" zu finden.
- Über 20 Visualisierungen lassen den Benutzer kryptografische Probleme, Sachverhalte und Algorithmen auf spielerische und anschauliche Weise erkunden und verstehen.
- Die Kryptologie bedient sich aus den verschiedensten wissenschaftlichen Feldern der Mathematik und Informatik. In den Visualisierungen werden daher auch die benötigten Grundlagen beider Fachgebiete erklärt.

Spiele

- In der Sektion "Spiele" kann man Spiele spielen, Strategien probieren, und nachdenken, wie die scheinbar einfachen Probleme gelöst werden können.
- Bei einzelnen (wie dem Zahlenhai) werden auch die Theorie dahinter und umfangreiche Lösungshinweise geliefert.



Allgemeine Bedienungshinweise

... 1/5 (Quick-Access: Suche über alle Teile von JCT)

Tipps und Tricks

- Mit der Tastenkombination **Strg-3** kann man das Quick-Access-Fenster öffnen.
- Hier können Verfahren und andere Inhalte aus JCrypTool gesucht und direkt geöffnet werden.
- Dies ist der schnellste Weg, etwas in der Standard-Perspektive oder in der Onlinehilfe zu finden, wenn man nicht weiß, wo man in den Menüs oder im Krypto-Explorer suchen soll.
- Klickt man auf einen angezeigten Eintrag, springt JCT auch direkt dahin.
- Anmerkung: Elemente in der Algorithmen-Perspektive werden vom Quick-Access-Fenster momentan noch nicht durchsucht.

ana	
Vorherige Wahl	Grille- Analyse (Analysen)
Ansichten	Entropie- Analyse (Analysen)
	Friedman-Test (Analysen)
	Häufigkeits- Analyse (Analysen)
	<input type="checkbox"/> Simple Power Analysis / Square and Multiply (Visualisierungen)
	Substitutions- Analyse (Analysen)
	Transpositions- Analyse (Analysen)
	Vigenère-Breaker (Analysen)
	Viterbi- Analyse (Analysen)
Befehle	Anzeigen in (Entropie- Analyse)
	Anzeigen in (Grille- Analyse)
	Anzeigen in (Häufigkeits- Analyse)
	Anzeigen in (Simple Power Analysis / Square and Multiply)
	Anzeigen in (Substitutions- Analyse)
	Anzeigen in (Transpositions- Analyse)
	Anzeigen in (Viterbi- Analyse)
	Entropie- Analyse
	Häufigkeits- Analyse
	Sicht anzeigen (Entropie- Analyse) - Zeigt eine bestimmte Sicht an
	Sicht anzeigen (Grille- Analyse) - Zeigt eine bestimmte Sicht an
	Sicht anzeigen (Häufigkeits- Analyse) - Zeigt eine bestimmte Sicht an
	Sicht anzeigen (Simple Power Analysis / Square and Multiply) - Zeigt eine bestimmte Sicht an
	Sicht anzeigen (Substitutions- Analyse) - Zeigt eine bestimmte Sicht an
Sicht anzeigen (Transpositions- Analyse) - Zeigt eine bestimmte Sicht an	
Sicht anzeigen (Viterbi- Analyse) - Zeigt eine bestimmte Sicht an	
Hilfe	Suche " ana " in der Hilfe


"Strg+3" drücken, um alle Übereinstimmungen anzuzeigen

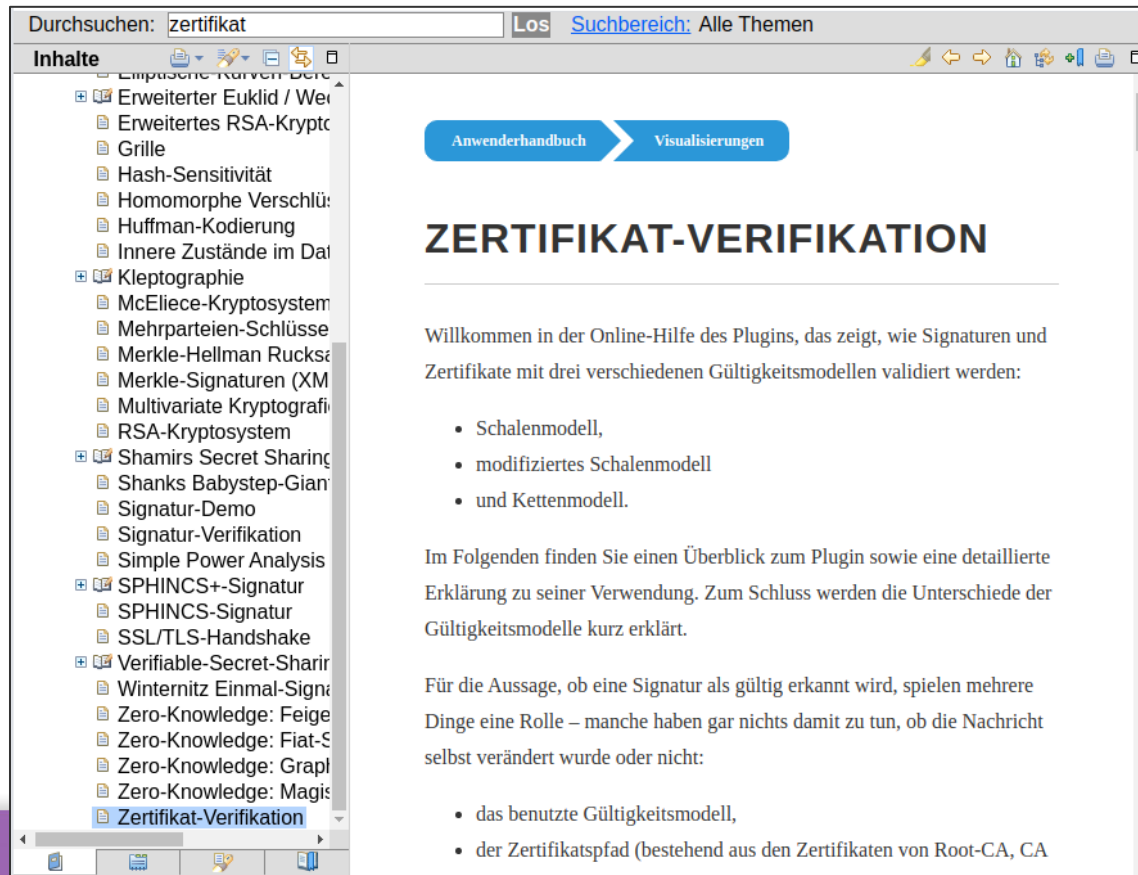
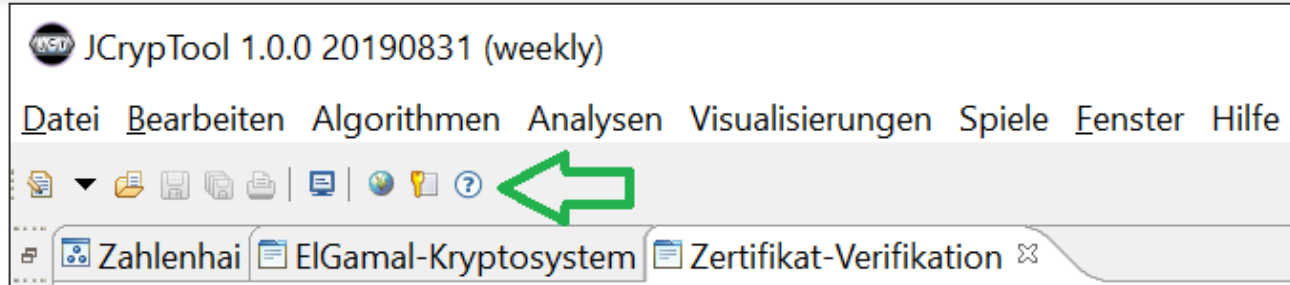
Quick-Access-Fenster

Allgemeine Bedienungshinweise

... 2/5 (Öffnen der umfangreichen Onlinehilfe in einem neuen Browser-Reiter)

Tipps und Tricks

- Die Fragezeichen-Ikone  in der Toolbar öffnet eine ausführliche **Onlinehilfe** in einem **Browser-Reiter**.



Onlinehilfe
im Browser

Allgemeine Bedienungshinweise

... 3/5 (Aufruf der angedockten Kontext-Hilfe)

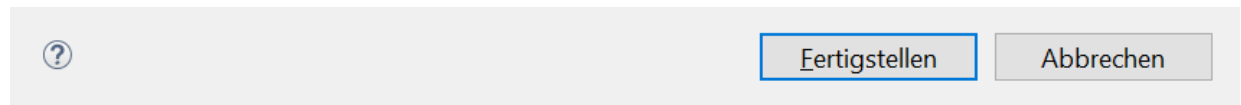
Tipps und Tricks

- Durch Drücken der Funktionstaste **F1** in JCT kann (unter Linux und Windows) die Kontext-Hilfe zu jedem Zeitpunkt geöffnet werden. Die Kontext-Hilfe liefert detailliertere Informationen und Verweise zur aktuellen Ansicht.
- Alternativ kann man mit dem großen **blauen Fragezeichen** das Kontexthilfe-Fenster ein- oder ausblenden. Dieses Fragezeichen befindet sich in allen Beschreibungs-Headern von Visualisierungen.

The screenshot displays the 'Verifiable-Secret-Sharing (VSS)' dialog box in the JCT application. The dialog is titled 'Verifiable-Secret-Sharing (VSS)' and contains a detailed description of the algorithm. Below the description, there are several sections for input and calculation: 'Parameter' (Anzahl der Spieler n, Anzahl Spieler t zur Rekonstruktion, Geheimnis s, Safe Prime p, Primfaktor q), 'Koeffizienten' (a0, a1), 'Commitments' (Koeffizient, Commitment Y), 'Shares' (Spieler i, Share Ni), and 'Rekonstruktion' (Spieler 1, Spieler 2). A 'Hilfe' window is docked on the left, showing the same content as the main dialog, with a red box around the 'Verifiable Secret Sharing' link and a search bar. A blue question mark icon is circled in red in the top right corner of the dialog.

Kontext-Hilfe zum Visualisierungs-Plugin „Verifiable Secret Sharing“





- Drückt man in Dialogfenstern F1 oder das Fragezeichen links unten, wird ein Kontext-Hilfefenster an den Dialog „angedockt“.

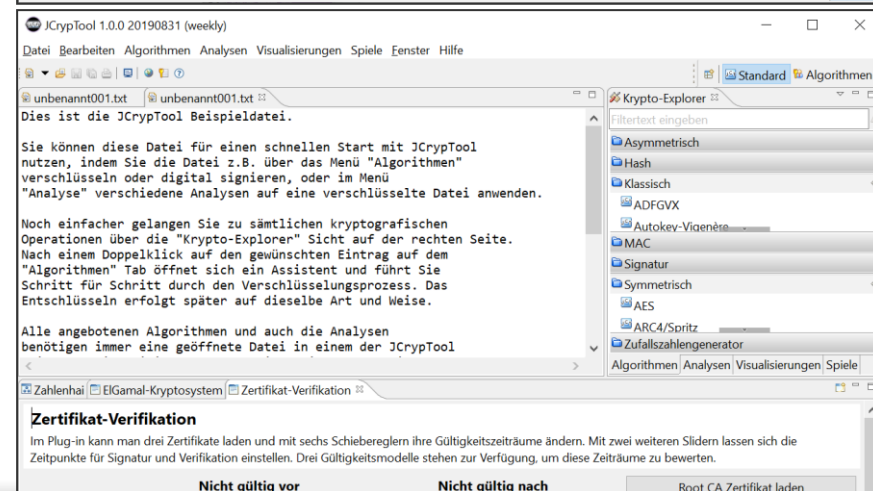
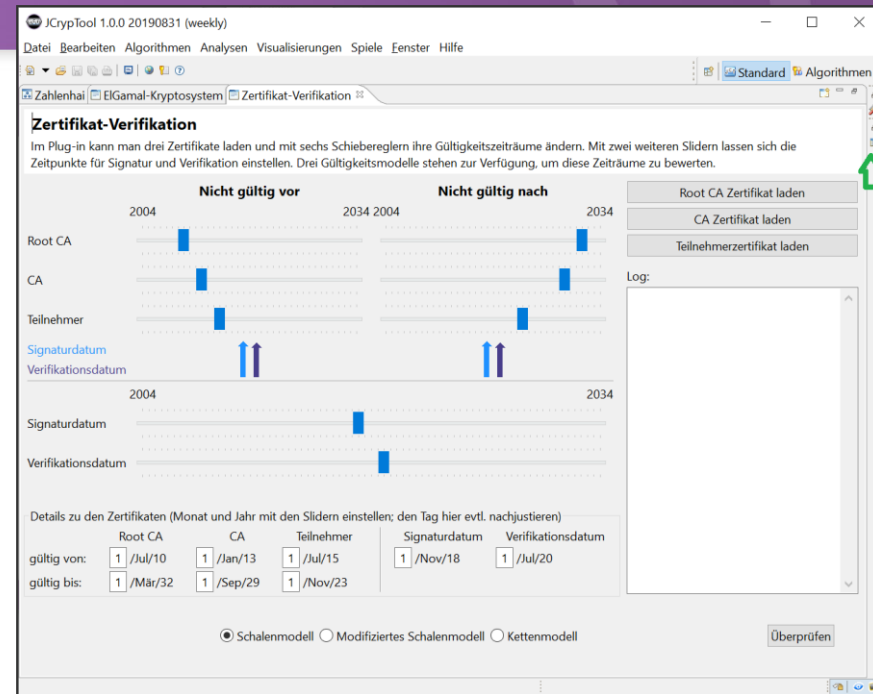


Allgemeine Bedienungshinweise

... 4/5 (kleiner, größer und wiederherstellen)

Tipps und Tricks

- Die Größe eines jeden Bereiches lässt sich in JCT über die Buttons oben rechts im Bereich steuern:
 - Aktuellen Bereich maximieren 
 - Bereich minimieren 
- Ist ein Bereich minimiert, erscheint er als kleine am Rand andockte Leiste. Die vorher in dem Bereich enthaltenen Reiter sind durch kleine Icons repräsentiert.
 - Mit Klick auf  kann man die letzte Größe eines Bereiches wiederherstellen.
 - Mit den Ikonen darunter kann man die einzelnen Reiter kurzzeitig als überlagertes Fenster einblenden. 
- Wiederherstellen von **View + Perspektive**:
 - Menü „Fenster“ \ „Perspektive zurücksetzen“
 - Doppelklick auf den Reiter wechselt zwischen Vollbild und verkleinertem Platz für das Plugin (typisches Eclipse-Verhalten)

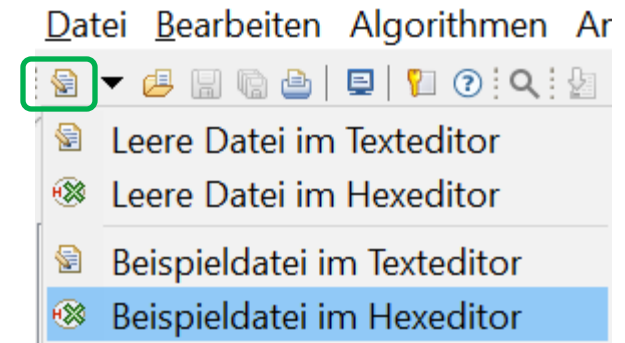
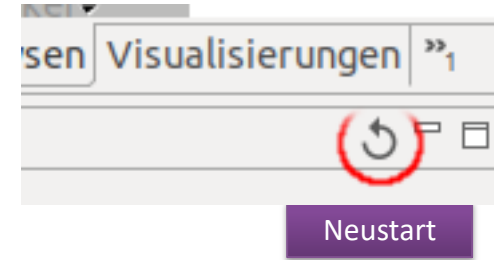


Allgemeine Bedienungshinweise

... 5/5 (Neustart innerhalb einer geöffneten Visualisierung; Öffnen eines Editors)

Tipps und Tricks

- Jede **Visualisierung** lässt sich über den Button „**Neustart**“ auf die Start-Einstellungen zurücksetzen. Der Button befindet sich in der oberen Toolbar des Plugin-Fensters.
- Schnelles Öffnen eines Editors
 - Ganz links in der Toolbar ist die Editor-Ikone. Klickt man darauf, wird ein Texteditor mit der Beispieldatei geöffnet.
 - Direkt rechts daneben ist der Pfeil. Mit dem Pfeil hat man die Auswahl, welcher Editortyp (und ob leer oder gefüllt) in der Mitte der Standard-Perspektive oder der Mitte der Algorithmen-Perspektive geöffnet werden soll.



Benutzervorgaben

...die globalen Einstellungen von JCT

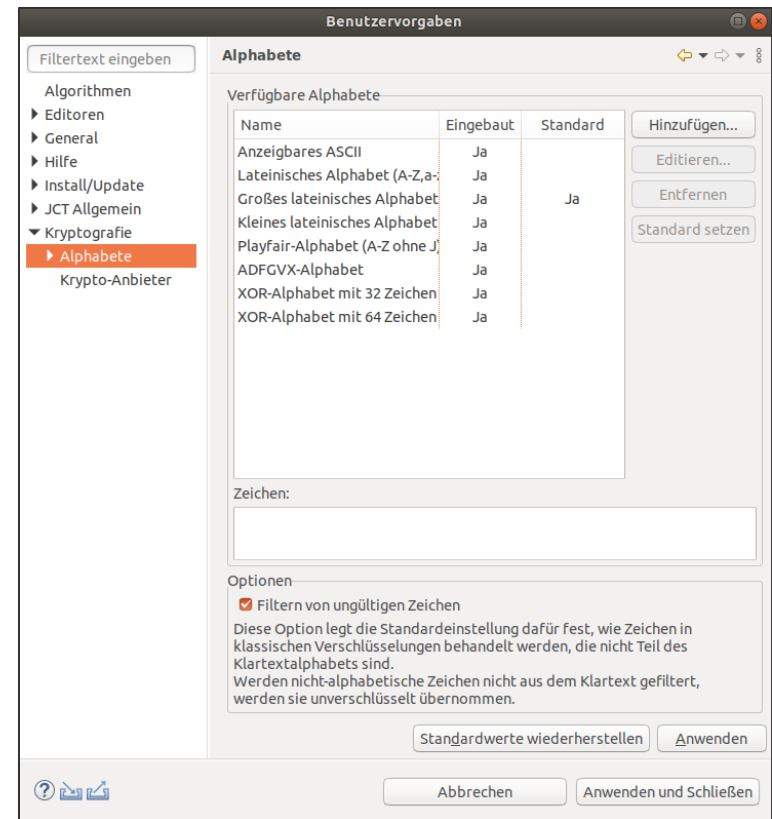
Weitere Einstellmöglichkeiten

- In den Benutzervorgaben befinden sich die globalen Einstellungen von JCT. Siehe die ff. Menüpfade: unter Windows + Linux: „Fenster \ Benutzervorgaben“ und auf MacOS: „JCrypTool \ Einstellungen“.

Die wichtigen JCT-spezifischen Einträge sind:

Bzgl. Kryptografie

- **Alphabete:** Ermöglicht die Verwaltung der Alphabete, die bei vielen klassischen Verfahren benutzt werden.
- **Schlüsselspeicher:** Hier lassen sich die Dateien verwalten, in denen die Schlüssel des JCT-Keystores gespeichert werden. Ein neu angelegter Schlüsselspeicher kann anschließend in der Schlüsselspeicher-Auswahl in der Perspektive „Algorithmen“ verwendet werden.



Kommandozeilen-Parameter

... Sprache und Datenverzeichnis vorgeben



Standardeinstellungen

- Startet man JCrypTool ohne Parameter, erscheint es in der Sprache des Betriebssystems, sofern diese Deutsch oder Englisch ist; ansonsten in Englisch.
- Die Einstellungen der letzten Session und JCT-spezifische, permanente Daten werden in einem Verzeichnis namens 'workstation' abgelegt. Dieses Verzeichnis wird beim ersten Start von JCT angelegt, unterhalb des JCrypTool-Verzeichnisses.

Parameter zur Steuerung von JCT gleich beim Aufruf – über die Kommandozeile

- **Sprache**
-nl [de, en]
- **Datenverzeichnis** (mit -data kann man das Verzeichnis 'workstation' zu ändern)
-data [Verzeichnis]

Beispiel: JCrypTool -nl de -clean -data USERVERZEICHNIS/jct-de
JCrypTool -nl en -clean -data USERVERZEICHNIS/jct-en

Mit -nl kann man festlegen, ob das deutsche oder das englische JCT gestartet wird.

Mit -data kann man festlegen, wohin JCT schreibt. Den Wert für USERVERZEICHNIS müssen Sie für sich passend angeben (unter Windows z.B. %LOCALAPPDATA%).



Einführung in das JCrypTool-Programm

2

Anwendungsbeispiele

22

Möglichkeiten zur Mitwirkung

87

Anwendungsbeispiele – Überblick (1)



Die Ameisenkolonie-Optimierung (ACO)	Seite 25
Viterbi-Analyse	Seite 30
Verifiable-Secret-Sharing	Seite 35
Signatur-Demo	Seite 40
Erweitertes RSA-Kryptosystem	Seite 45
SETUP-Angriff auf die RSA-Schlüsselgenerierung (Kleptographie)	Seite 50
Zero-Knowledge-Protokoll: Fiat Shamir	Seite 55
Android-Mustersperre (AUP)	Seite 60
Kaskaden mit dem Aktionen-Fenster	Seite 64
Variable Alphabete für klassische Algorithmen	Seite 70
JCrypTool-Konsole für klassische Verfahren	Seite 74
Die Perspektive „Algorithmen“	Seite 79

Die Ameisenkolonie-Optimierung (ACO)

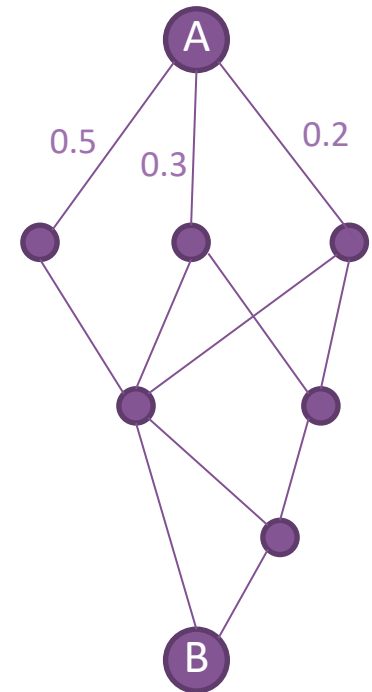
Die Idee

Problematik

- Die Visualisierung der Ameisenkolonie-Optimierung^[1] ermöglicht es, einen Text, der mit einem Transpositionsverfahren verschlüsselt wurde, wieder zu entschlüsseln.

Funktionsweise

- Der Ameisenkolonie-Algorithmus ist ein effizienter Algorithmus zum Lösen von kombinatorischen Problemen.
- Ziel des Algorithmus kann es z.B. sein, in einem Graphen den kürzesten Weg von A nach B zu finden.
- Der Algorithmus ist den Ameisen nachempfunden, die schnell einen kurzen Weg vom Ameisenhaufen zu einer Futterstelle finden können.
- Im Algorithmus wählt eine Ameise ihren Weg anhand lokaler Informationen (z.B. den Kantenbewertungen) und danach, welcher Weg von den Ameisen vorher häufiger gewählt wurde.
- Je mehr Ameisen einen bestimmten Weg laufen, desto mehr Ameisen werden folgen. Dieses Verhalten wird als Schwarmintelligenz bezeichnet.
- Prinzipiell beruht dieser Algorithmus auf statistischen Auswertungen.



[1] ACO = Ant Colony Optimization

Die Ameisenkolonie-Optimierung

Die Implementierung in JCT

Im Menü

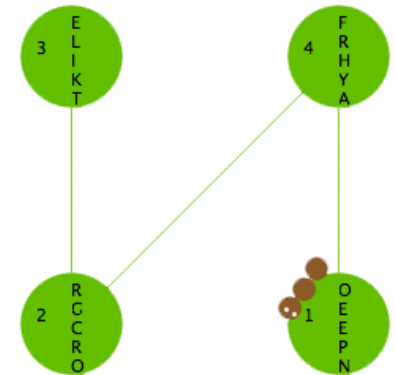
„Visualisierungen“ \ „Ameisenkolonie-Optimierung“

Der Algorithmus in der Anwendung

- Mit der Ameisenkolonie-Optimierung in JCT kann ein mit einem einfachen Spalten-Transpositionsverfahren verschlüsselter Geheimtext entschlüsselt werden.
- Dazu wird die Schlüssellänge n benötigt, der Geheimtext wird zeilenweise in n -viele Spalten geschrieben, diese Spalten dienen nun als Knoten für den Graphen.
- Beim Zusammenfügen der Spalten in unterschiedlichen Reihenfolgen entstehen verschiedene Buchstabenpaare. Diese Buchstabenkombinationen kommen in jeder Sprache unterschiedlich häufig vor. Aus diesen Häufigkeiten und der besuchten Häufigkeit eines Weges von vorherigen Ameisen werden nun Kantengewichtungen des Graphen berechnet.
- In jeder Iteration wird nun ein möglicher Klartext aus einer Spaltenreihenfolge generiert. Dieser Text wird dann mit einer einzugebenden Liste häufig vorkommender Wörter bewertet.
- Die Bewertung fließt in die Pheromon-Matrix ein. Diese Matrix dient nachfolgenden Ameisen dazu, einen neuen, besseren Klartext zu finden.

Pheromonmatrix			
-	0.2	0.1	0.1
0.1	-	0.2	5.5
0.1	5.5	-	0.1
5.5	0.1	0.2	-

Zeichen in den Knoten			
O	R	E	F
E	G	L	R
E	C	I	H
...



Die Ameisenkolonie-Optimierung

Ein Anwendungsbeispiel 1/2

Versuchen Sie, den Geheimtext

OREFEGFRECIHPRKYNOTASLAYNLEG

zu entschlüsseln.

- Fügen Sie diese Zeichenfolge im Visualisierungsfenster unterhalb von „Geheimtext direkt eingeben“ in das Textfeld ein, und wählen Sie als Schlüssellänge 4 ^[1].
- Drücken Sie auf „Analyse starten“.

Konfiguration

Geheimtext erzeugen

oder

Geheimtext direkt eingeben

OREFEGFRECIHPRKYNOTASLAYNLEG

Geheimtextlänge: 28

Vermutete Schlüssellänge

4

Sprache des Textes

Deutsch

Analyse starten

[1] Die Schlüssellänge kann aus statistischen Auswertungen gewonnen werden.
Außerdem muss hier die Länge des Geheimtextes ein Vielfaches der Schlüssellänge sein.

Die Ameisenkolonie-Optimierung

Ein Anwendungsbeispiel 2/2

Die Gruppierungen „Analyse“ und „Visualisierung“ sind jetzt aktiviert.

Dort haben Sie folgende Parameter:

Alpha & Beta:

- Diese Parameter regeln die Wahrscheinlichkeiten, mit der die Ameise eine Kante (einen Verbindungsweg) auswählt. Je höher Alpha, desto öfters folgt eine Ameise einem Pfad, den eine Ameise zuvor bereits eingeschlagen hat. Je größer Beta, desto wichtiger werden Buchstaben-Bigramme gewertet.

Verdunstung:

- Eine hohe Verdunstung lässt das Pheromon einer Ameise schneller verdunsten. Die nachfolgenden Ameisen finden so nur eine dünnere Pheromon-Spur vor.
- Die Pheromon-Matrix berechnet sich aus diesen drei Parametern und steuert so das Verhalten der Ameise. Genauere Informationen dazu finden Sie auch in der Onlinehilfe.

Ameisen-Steuerung:

Mit den Buttons in dieser Unter-Gruppierung können die Ameisen im Graphen gesteuert werden.

The screenshot shows the software interface for Ant Colony Optimization. It is divided into two main sections: 'Analyse' and 'Visualisierung'.
The 'Analyse' section includes:
- 'Detailstufe' with radio buttons for 'Knoten für Knoten' (selected) and 'Mehrere Durchläufe'.
- An 'Animation' checkbox which is currently unchecked.
- 'Ameisen-Steuerung' with three buttons: 'Zum nächsten Knoten', 'Bis zum letzten Knoten', and 'Neue Ameise platzieren'.
- 'Algorithmus-Einstellungen' with three sliders: 'Alpha: 0.8', 'Beta: 0.8', and 'Verdunstung: 0.9'.
The 'Visualisierung' section includes:
- Radio buttons for 'Visualisierung als Graph' (selected) and 'Pheromonmatrix anzeigen'.
- A graph visualization with four green circular nodes. Node 1 (bottom right) contains 'O E E P N' and has three small brown dots representing ants. Node 2 (bottom left) contains 'R C C R O'. Node 3 (top left) contains 'E L I K T'. Node 4 (top right) contains 'F R H Y A'. Edges connect nodes 1-2, 1-4, and 3-4.
A small question mark icon is visible in the bottom right corner of the visualization area.

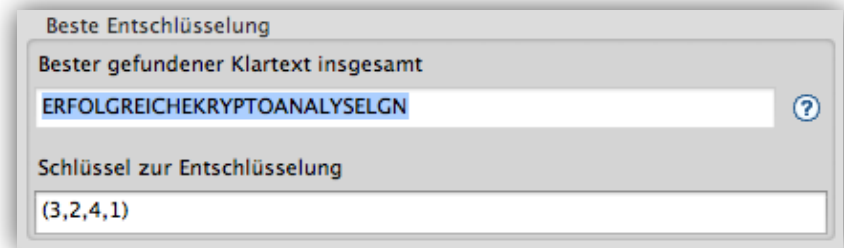
Die Ameisenkolonie-Optimierung

Lernziel

Ergebnis

- Haben Sie es geschafft, den Geheimtext von Seite 20 zu entschlüsseln?
- Als Klartext sollte sich (durchschnittlich nach 25 Durchläufen^[1] mit Alpha= 0,8, Beta= 0,8, Verdunstung= 0,9) ergeben:

ERFOLGREICHEKRYPTOANALYSELGN^[2]



Fazit

- Die Permutations-Verschlüsselung ist keine sichere Verschlüsselung.
- Mit dem Ameisenalgorithmus lassen sich unterschiedliche kombinatorische Optimierungsprobleme lösen – nicht nur aus dem Bereich der Kryptoanalyse.
- Für viele Probleme hat die Natur bereits eine Lösung, es kommt darauf an, diese zu finden, zu verstehen und zu abstrahieren.

[1] Die Anzahl der Durchläufe variiert stark und es kann vorkommen, dass die Lösung nach 50 Durchläufen noch nicht gefunden wurde. Es kann sich dann lohnen, das Plugin zurückzusetzen und von Neuem zu beginnen.

[2] Füllzeichen, die angehängt wurden, damit die Textlänge durch 4 teilbar ist.

Viterbi-Analyse

Die Idee



Problematik

- Gegeben sei ein Running-Key-Chiffre – also ein Geheimtext, der dadurch entstand, dass zwei Klartexte durch XOR oder durch modulare Addition verknüpft wurden.
- Lassen sich die beiden Klartexte wieder aus dem Geheimtext zurückgewinnen?

Tatsächlich ist dies möglich – der Viterbi-Algorithmus löst eine solche Aufgabe.

Funktionsweise

- Der Viterbi-Algorithmus ist ein rekursiver Algorithmus und verwendet die Methode der dynamischen Programmierung.
- Der Algorithmus analysiert Häufigkeiten von versteckten Zustands-/Markov-Ketten in einer Eingabe-Sequenz.
- Neben der Kryptografie findet der Algorithmus ebenfalls Anwendung bei der Spracherkennung, der Analyse von DNS-Strukturen, und bei der Reduktion von Übertragungsfehlern.
- Siehe <http://de.wikipedia.org/wiki/Viterbi-Algorithmus>



Im Menü

„Visualisierungen“ \ „Viterbi“

Der Algorithmus in der kryptoanalytischen Anwendung

- Die statistische Auswertung von Wahrscheinlichkeiten von N-Grammen und die Zuhilfenahme von Wörterbüchern der Verschlüsselungssprache bilden die Grundlage.
- Das Wissen, dass der Text aus zwei Klartexten per modularer Addition bzw. per XOR entstanden ist, fließt in das Analysemodell mit ein.
- Der Geheimtext wird Buchstabe für Buchstabe in mögliche Klartextbuchstaben zerlegt, dabei werden die umgebenen Zeichen und die damit entstehenden N-Gramme und deren Wahrscheinlichkeiten (in der gewählten Sprache) mit in Betracht gezogen.
- Aus den Möglichkeiten der vorkommenden Buchstaben im Klartext werden Pfade generiert, denen Wahrscheinlichkeiten zugeordnet sind. Unwahrscheinliche Pfade werden nicht weiter verfolgt.

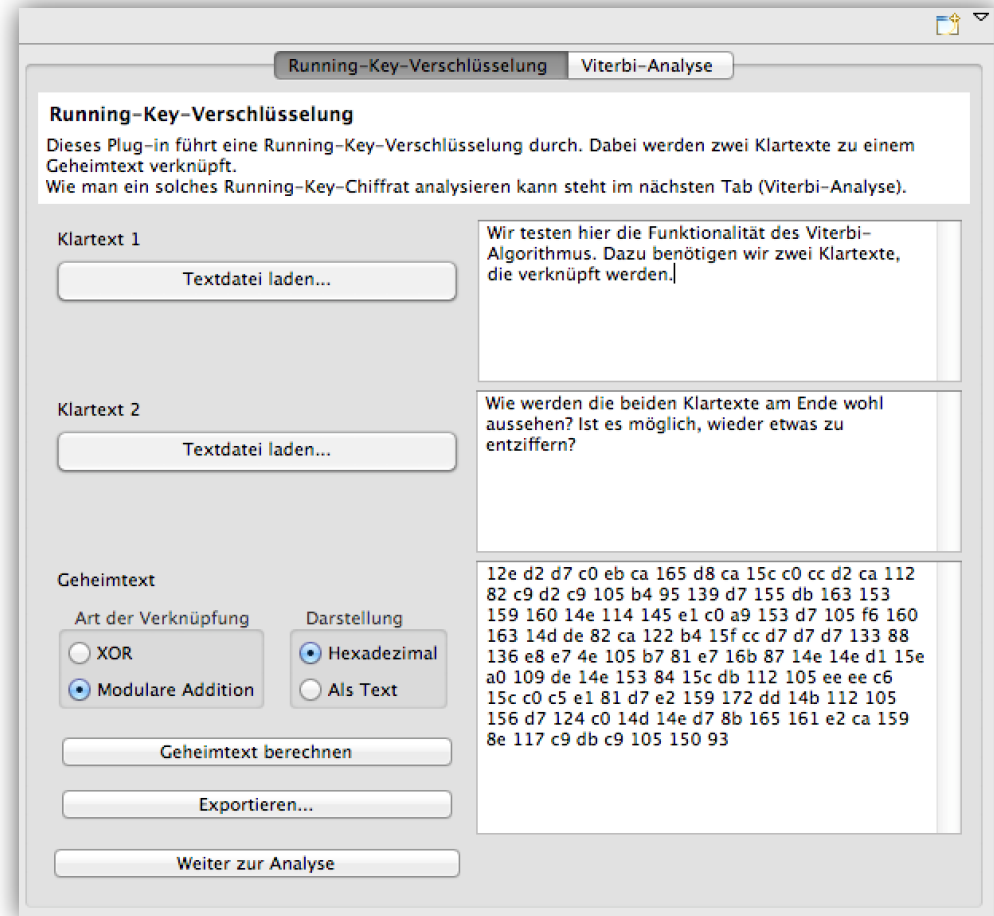
Viterbi-Analyse

Ein Anwendungsbeispiel 1/2

Im ersten Schritt ist ein entsprechender Geheimtext für die Viterbi-Analyse zu erstellen.

Dazu bietet das Plugin einen geeigneten Textgenerator.

- Erstellen Sie zwei Klartexte oder laden Sie die Texte aus Textdateien.
- Bei der Verknüpfung der Einzelbuchstaben der Klartexte kann zwischen XOR und modularer Addition unterschieden werden.
- Durch Klicken auf „Geheimtext berechnen“ wird das Chiffre erstellt.
- Drücken Sie auf „Weiter zur Analyse“.



Viterbi-Analyse

Ein Anwendungsbeispiel 2/2

Im nächsten Schritt, der „Viterbi-Analyse“, wird der Viterbi-Algorithmus auf den Geheimtext angewandt, um möglichst viel Information über die beiden Klartexte zurück zu gewinnen.

- Wählen Sie die vermutete Sprache des Eingabetextes.
- Stellen Sie eventuell die gewünschte Größe der N-Gramme und der Suchtiefe ein und starten Sie die Analyse.
- In den unteren beiden Textfeldern sehen Sie, wie das Ergebnis der Entschlüsselung erzeugt wird. Dort können Sie beobachten, wie sich stets ein großer Teil der Zeichenkette dynamisch verändert. Dies kann einige Sekunden dauern. Am besten im Fullscreen-Modus ansehen.

Was passiert bei der Variation der N-Gramm-Größe und der Suchtiefe?

The screenshot shows a software window titled "Running-Key-Verschlüsselung" with a sub-tab "Viterbi-Analyse". The main content area contains the following text:

Viterbi-Analyse
Der Viterbi-Algorithmus ist eine komplexe Form der Häufigkeitsanalyse. Mithilfe von Wahrscheinlichkeiten des Auftretens von Zeichenketten wird versucht, die Klartexte zusammenzubauen.

Die Arbeitsweise des Viterbi-Algorithmus lässt sich grob in die folgenden drei Schritte einteilen:

1. Ein Buchstabe wird in alle möglichen Kombinationen der Verknüpfung zerlegt. Als Umkehrfunktion verwendet man beim klassischen XOR-Verfahren ein weiteres XOR und bei der modularen Addition eine modulare Subtraktion.
2. Es wird die Wahrscheinlichkeit dieser Kombinationen berechnet. Diese Berechnungen erledigt ein komplexes Sprachmodell, welches längere Zeichenfolgen berücksichtigt. Es fließt also nicht nur das aktuelle Zeichen, sondern alle Vorgängerbuchstaben in die Berechnung mit ein.
3. Die wahrscheinlichsten Pfade werden geordnet gespeichert. Unwahrscheinliche Pfade werden nicht weiter verfolgt.

Eingabe

Geheimtext
[Textdatei laden...]
Darstellung:
 Hexadezimal
 Als Text

12e d2 d7 c0 eb ca 165 d8 ca 15c c0 cc d2 ca 112 82 c9 d2 c9 105 b4 95 139 d7 155 db 163 153 159 160 14e 114 145 e1 c0 a9 153 d7 105 f6 160 163 14d de 82 ca 122 b4 15f cc d7 d7 133 88 136 e8 e7 4e 105 b7 81 e7 16b 87 14e 14e d1 15e a0 109 de 14e 153 84 15c db 112 105 ee ee c6 15c c0 c5 e1 81 d7 e2 159 172 dd 14b 112 105 156 d7 124 c0 14d 14e d7 8b 165 161 e2 ca 159 8e 117 c9 db c9 105 150 93

Parameter für die Analyse

Sprache des Deutsch Englisch
n-Gramm Größe: 4
Suchtiefe: 256
[Analyse starten]

Ergebnis

Möglicher 1. Klartext: Ich testen hier die Buchchonglicht des Viter e4zchen? Aus. Daseinenö, wieder etwas Klarteine, die wie yoben die b

Möglicher 2. Klartext: eoo werden die beider Vorsterte1ÿm Ende wohlben eiorithust es tC`ichtigen wir zwei zu entoffern? Verklicht werden

[Exportieren...]



Fazit

- Mit dem Viterbi-Algorithmus lassen sich zwei per XOR oder modularer Addition zu einem Geheimtext verknüpfte Klartexte wieder entschlüsseln..
- Ein Defizit bei der Entschlüsselung hat der Algorithmus zu Beginn der Klartexte. Hier fehlen die umgebenen N-Gramme und es existieren noch keine Wahrscheinlichkeitspfade, auf die der Algorithmus zurückgreifen kann.
- Lange Wörter werden vom Algorithmus seltener entschlüsselt.
- Das zugrundeliegende Wörterbuch spielt eine wesentliche Rolle, da der Algorithmus die Wörter daraus zusammen sucht.
- Es werden nur N-Gramme gefunden, die auch im Wörterbuch enthalten sind. Daher ist die Länge der N-Gramme im Plugin auf $N=5$ beschränkt. Für größere N müssten sonst alle Wörter der Länge N im Wörterbuch vorkommen, was bei z.B. $N=7$ bereits sehr viele Wörter mehr wären.
- Die Variation der N-Gramm-Größe und der Suchtiefe hat direkten Einfluss auf das Ergebnis.
 - Die Größe der N-Gramme gibt vor, auf welche Wörter im Wörterbuch zurückgegriffen wird.
 - Der Parameter Suchtiefe bestimmt wie viele Kandidaten für Klartextpaare (Pfade) zur Analyse des nächsten Zeichens verwendet werden (der Algorithmus verwirft ja nach jedem Zeichen unwahrscheinliche Pfade). Somit regelt die Suchtiefe den Einfluss der aktuellen Entschlüsselungsposition auf vorangehende Zeichen der Zeichenkette.

Verifiable-Secret-Sharing

Die Idee



Problematik

- Das Verifiable-Secret-Sharing (VSS) ist eine erweiterte Variante des **Secret-Sharing** (zu deutsch „Geheimnis-Teilung“).
- Beim Secret-Sharing geht es darum, ein gemeinsames Geheimnis unter einer gewissen Anzahl an Mitwissenden, den Spielern, aufzuteilen. Die Spieler bekommen die sogenannten „Shares“.
- Es soll außerdem eine Mindestanzahl an Spielern (nicht unbedingt alle) benötigt werden, damit das gemeinsame Geheimnis wieder entschlüsselt werden kann.
- Ein einzelner Share oder weniger Shares als die definierte Mindestanzahl soll hingegen nutzlos sein.

Die Erweiterung „Verifiable“

- Das VSS ist sicherer als das normale Secret-Sharing. Beim Aufteilen des Geheimnis muss demjenigen, der das Geheimnis aufteilt (dem „Dealer“), vertraut werden. Dieser könnte die Shares beim Austeilen abändern, und das Verfahren würde nutzlos.
- Um dieses Problem zu lösen, erzeugt der Dealer beim VSS zusätzliche „Commitments“. Damit können die Shares von den Spielern auf ihre Richtigkeit getestet werden.

Verifiable-Secret-Sharing

Die Implementierung in JCT



Im Menü

„Visualisierungen“ \ „Verifiable-Secret-Sharing“

Der Algorithmus in der Anwendung

- Statt mit einem Geheimnis in Textform wird das Geheimnis durch eine Zahl ausgedrückt. Vorher muss man sich eine Transformation von Text in Zahlen überlegen.
- Jeder der n Spieler erhält einen Share. Zur Rekonstruktion des Geheimnisses soll aber die Kenntnis von beliebigen t Shares genügen ($1 < t \leq n$).
- Ein Polynom vom Grad $(t-1)$ kann durch Kenntnis von t Punkten auf dem Polynomgraphen eindeutig rekonstruiert werden. Dazu kann man die Lagrange-Interpolation benutzen.
- Diese mathematische Erkenntnis wird beim VSS geschickt verwendet.
- Das Geheimnis wird als absoluter Term des Polynoms benutzt. Dadurch erhält man das Ergebnis ganz einfach als Funktionsauswertung des Polynoms an der Stelle 0.

Verifiable-Secret-Sharing

Ein Anwendungsbeispiel 1/2

Erster Schritt

- Wählen Sie die Gesamtanzahl n der Spieler und die benötigte Anzahl t der Spieler zur Rekonstruktion.
- Legen Sie das Geheimnis fest.

Die Zahlen „Safe Prime“, „Primfaktor“ und „Generator“ werden dann, falls möglich, automatisch ausgefüllt.

- Klicken Sie auf „Koeffizienten bestimmen“.

Parameter

Anzahl der Spieler n	6
Anzahl Spieler t zur Rekonstruktion	5
Geheimnis s	10
Safe Prime p ($p > 2s$)	23
Primfaktor q ($2q = p - 1$)	11
Generator g	2

Nächster Schritt:

Zweiter Schritt

Das Polynom wird jetzt festgelegt. Als Dealer haben Sie hier Einfluss auf das Polynom, aus denen die Shares berechnet werden. Aus dem Polynom werden außerdem die Commits berechnet.

- Das initiale Polynom gibt Spieler 1 zu viel Information. Deshalb sollte man den Button „Generieren“ drücken, um zufällige Koeffizienten zu erzeugen.
- Sie können nun über „Commit“ die Commits berechnen.

Ändern Sie das Polynom jetzt nochmal, werden die Shares, falls sie mit den vorherigen Commits überprüft, nun als ungültig erkannt.

- Lassen Sie sich die „Shares berechnen“.

Koeffizienten

$a_0 = s$	10
a_1	3
a_2	8
a_3	3

$P(x)$

Nächster Schritt:

Verifiable-Secret-Sharing

Ein Anwendungsbeispiel 2/2



Rekonstruktions-Schritt

Das Geheimnis ist auf die Spieler aufgeteilt.

- Die Shares können per „Check“ überprüft werden.
- In dem Beispiel rechts wurde das Polynom nachträglich noch einmal verändert. Die Shares sind somit ungültig. Die Vertrauenswürdigkeit des Dealers sollte also angezweifelt werden.

Commitments	
Koeffizient	Commitment Y_i
a_0	12
a_1	2
a_2	16
a_3	16
a_4	16

Shares			
Spieler 1	24	= 2	Check
Spieler 2	92	= 4	Check
Spieler 3	274	= 10	Check
Spieler 4	654	= 5	Check
Spieler 5	1340	= 9	Check
Spieler 6	2464	= 0	Check

Rekonstruktion	
Spieler 1	<input type="checkbox"/>
Spieler 2	<input checked="" type="checkbox"/>
Spieler 3	<input checked="" type="checkbox"/>
Spieler 4	<input checked="" type="checkbox"/>
Spieler 5	<input checked="" type="checkbox"/>
Spieler 6	<input checked="" type="checkbox"/>

Rekonstruieren

Ein Share wurde in dem Beispiel als gültig verifiziert, obwohl das Polynom verändert wurde. Es genügt also nicht, sich auf die Gültigkeit eines einzigen Shares zu verlassen.

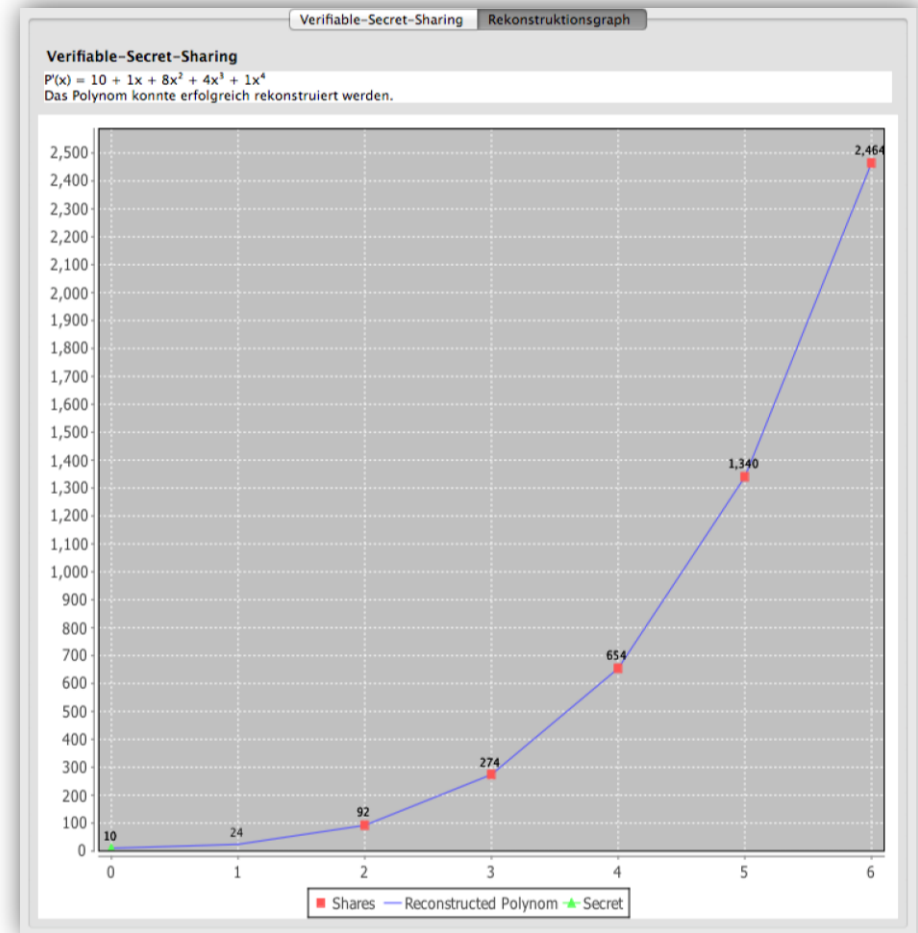
- Für die Rekonstruktion können rechts unter „Rekonstruktion“ die Spieler ausgewählt werden, deren Shares benutzt werden sollen.
- Für unser Beispiel müssen wir mindestens fünf Spieler auswählen (da $t=5$ war).
- Anschließend kann durch „Rekonstruieren“ das Geheimnis wieder erzeugt werden (ungültige Shares müssen nicht notwendigerweise ein falsches Geheimnis liefern).

Verifiable-Secret-Sharing

Lernziel

Fazit

- Ein Geheimnis lässt sich unter mehreren Spielern so aufteilen, dass es nur gemeinsam wieder entschlüsselt werden kann.
- Mehrere Botschafter können so z.B. brisante Informationen aufgeteilt übermitteln, ohne dass sie dabei selbst das ganze Geheimnis kennen.
- Es kann eine Toleranz geschaffen werden, dass später nicht alle Botschafter zur Rekonstruktion benötigt werden.
- Ein weiteres mathematisches Modell, die Lagrange-Interpolation, trifft auf eine interessante Anwendung.





Problematik

1. Elektronische Dokumente können à priori nicht auf den Autor überprüft werden. Dazu braucht man ein Verifizierungsmerkmal des Autors, dies kann z.B. eine Unterschrift sein.
2. Hat man nur das elektronische Dokument, kann man eine nachträgliche Veränderung kaum erkennen.

Um diese Probleme zu umgehen, kann der Autor sein elektronisches Dokument digital signieren.

Funktionsweise

- Der Autor generiert aus dem Dokument einen Hashwert (siehe Folie [41](#)).
- Der Hashwert wird mit dem privaten Schlüssel des Autors verschlüsselt (bei Verwendung von RSA).
- Den verschlüsselten Hashwert und die benutzte Hashfunktion stellt der Autor mit dem Dokument öffentlich bzw. dem Empfänger zur Verfügung.
- Ein Interessent, der die Integrität des Dokuments überprüfen möchte, kann nun mit dem öffentlichen Schlüssel des Autors den Hashwert des Dokuments entschlüsseln.
- Den Hashwert kann der Interessent selbst gegenprüfen, indem er die vom Autor benutzte Hashfunktion erneut auf das Dokument anwendet. Sind der entschlüsselte Hashwert und der neu berechnete Hashwert identisch, kann er sich sicher sein, dass das Dokument nicht verfälscht wurde.

Signatur-Demo

Die Implementierung in JCT

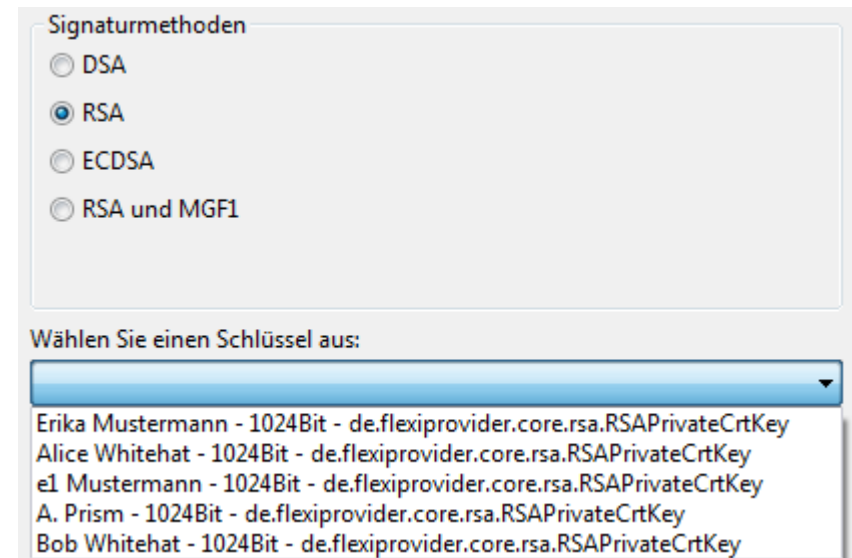


Im Menü

„Visualisierungen“ \ „Signatur-Demo“

Der Algorithmus in der Anwendung

- Das Plugin bietet die Möglichkeit, ein Dokument aus einer Datei oder einen selbst eingegebenen Text zu signieren.
- Als Hashmethoden stehen die Funktionen MD5, SHA-1 und SHA-2 (SHA-256, SHA-384 und SHA-512) zur Verfügung.
- Anschließend kann, je nach gewählter Hashfunktion, DSA, RSA, ECDSA oder RSA mit MFG1 als Signaturmethode benutzt werden.
- Darunter werden die Subjekte (Schlüsselinhaber) angeboten, die einen Schlüssel zur gewählten Signaturmethode haben.*



* Es gibt zwei Wege, um passende Schlüssel für die Subjekte (Schlüsselinhaber, Benutzer) zu generieren:

- a) in der Algorithmen-Perspektive.
- b) mit dem Visualisierungs-Plugin „Public-Key-Infrastruktur“ (JCT-PKI).

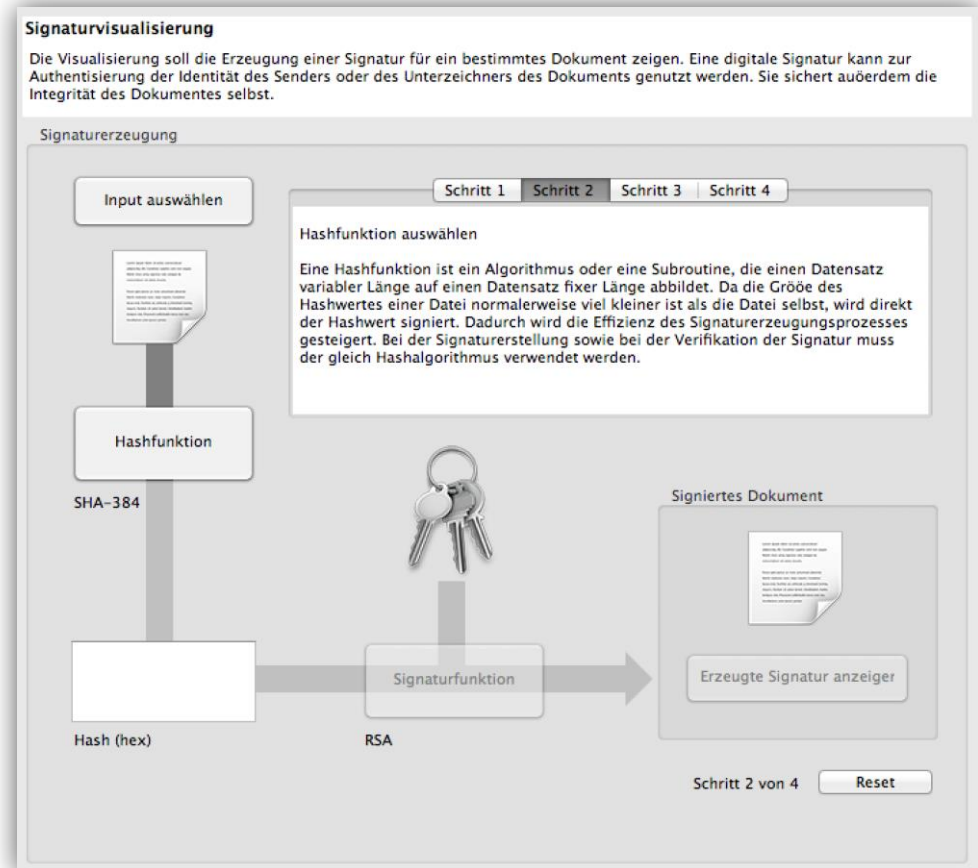
Signatur-Demo

Ein Anwendungsbeispiel 1/2

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

Erster Schritt: Hashwert erzeugen

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



Signatur-Demo

Ein Anwendungsbeispiel 2/2

Zweiter Schritt: Signatur erzeugen

- Durch Klick auf „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.

Erzeugte Signatur anzeigen

Besitzer der Signatur: -

Verwendeter Schlüssel/Kurve: ANSI X9.62 prime256v1 (256 bits)

Signaturmethode: SHA384withECDSA

Signatur

Adresse	Hex	Ascii
00000	30 44 02 20 75 E0 76 4C 20 EB 02 A0 E6 2F	0D uävL ë æ/
0000E	94 5C 74 73 AC 8D 5F F9 58 9C B2 91 1F 34	\ts~_ù]²4
0001C	00 6E 62 D6 D7 69 89 E9 02 20 47 19 8E 2C	nbÖxi¹é G,
0002A	D0 12 32 B2 C4 CA EA 67 94 95 F1 96 39 4B	D²²ÄËgñ9K
00038	DE F9 88 83 16 C4 25 57 C9 0A EF FC D3 8F	püA%WÉ

Länge der Signatur: 70 Bits

Darstellungsmöglichkeiten der Signatur

Hex-Dump (Hex und Ascii) Oktal Dezimal Hex

Signierte Nachricht

Adresse	Hex	Ascii
00000	55 6E 64 20 77 65 6E 6E 20 73 69 65 20 6E	Und wenn sie n
0000E	69 63 68 74 20 67 65 73 74 6F 72 62 65 6E	icht gestorben
0001C	20 73 69 6E 64 2C 20 73 6F 20 6C 65 62 65	sind, so lebe
0002A	6E 20 73 69 65 20 6E 6F 63 68 20 68 65 75	n sie noch heu
00038	74 65 2E 20 0A 55 6E 64 20 77 65 6E 6E 20	te.
00046	69 68 72 20 50 72 6F 66 69 6C 20 76 65 72	ihr Profil ver

Länge der signierte Nachricht: 143 Bits

Um das signierte Dokument und die erzeugte Signatur anzuzeigen, klicken Sie auf "Hex Editor öffnen". Dort können Sie die Signatur speichern, falls Sie diese verifizieren ...

Hex Editor öffnen Schließen



Fazit

- Die Integrität von elektronischen Dokumenten kann mit Hilfe einer Signatur überprüft werden.
- Krypto-Algorithmen helfen, um den Autor und die Integrität des Dokuments zu verifizieren.
- Wird ein Dokument verfälscht, so ändert sich damit auch dessen Hashwert.
- Um sicher zu stellen, dass das Dokument vom angegebenen Autor stammt, signiert es der Autor mit seinem privaten Schlüssel.
Nur mit dem „richtigen“ öffentlichen Schlüssel (also dem des angegebenen Signierers) kann man den originalen Hashwert verifizieren (und damit die Integrität des Dokuments).
Somit kann der Hashwert, obwohl er öffentlich verfügbar ist, nicht geändert werden.

Erweitertes RSA-Kryptosystem

Die Idee



Wie funktioniert heutige Verschlüsselung, die Sicherheit garantiert?

- Für Daten, die auf öffentlichen Kanälen übertragen werden, sollten Verschlüsselungsverfahren verwendet werden. Ein solches ist das RSA-Verfahren (sofern es mit den richtigen Parametern benutzt wird).
- Das RSA-Verfahren ist ein asymmetrisches Verfahren, es benötigt zwei Schlüssel: einen privaten und einen öffentlichen Schlüssel. Jeder Teilnehmer benötigt ein eigenes Schlüsselpaar, das er zunächst generieren (lassen) muss.
- Inhalte, die mit dem öffentlichen Schlüssel eines Teilnehmer verschlüsselt wurden, können lediglich mit dem entsprechenden privaten Schlüssel wieder entschlüsselt werden.
- Für die verschlüsselte Kommunikation mit einer anderen Person muss man im Besitz von deren öffentlichen Schlüssel sein. Zunächst muss daher ein Schlüsselaustausch für die öffentlichen Schlüssel stattfinden.

Um diesen Prozess zu vereinfachen wird oftmals eine „Certificate Authority“ (CA, Trustcenter, PKI) benutzt, die öffentliche Schlüssel speichert, verwaltet und verifiziert und Zertifikate ausstellt.

→ Siehe auch das Visualisierungs-Plugin „Public-Key-Infrastruktur“(JCT-PKI), das die Vorgänge in einer PKI mit ihren Instanzen Benutzer, RA und CA visualisiert.

Erweitertes RSA-Kryptosystem

Die Implementierung in JCT



Im Menü

„Visualisierungen“ \ „Erweitertes RSA-Kryptosystem“

Funktionsweise

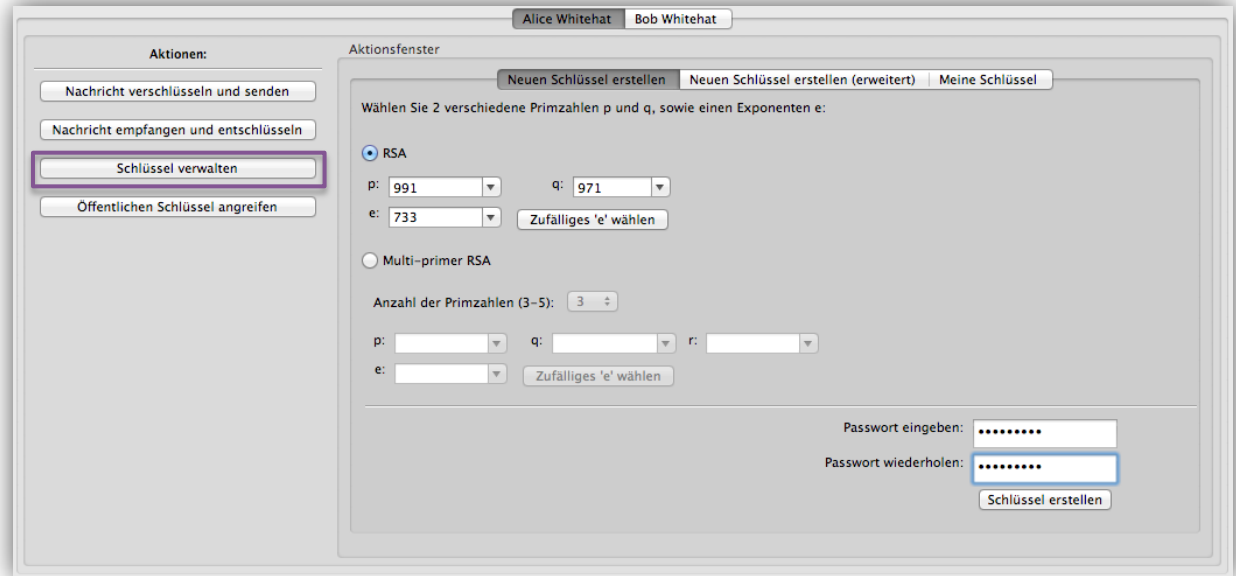
- Dieses Plugin in JCT ermöglicht es, Identitäten sowie Schlüssel zu verwalten und Nachrichten zu verschicken und zu empfangen.
- Zusätzlich ist es möglich, einen Angriff auf die Schlüssel auszuführen. Dabei wird mit Brute-Force-Methoden versucht, den Modul „n“ in seine beiden Primfaktoren zu zerlegen.
- Das Plugin bietet also eine vollständige, unabhängige Kommunikationsplattform. Der Benutzer kann außerdem experimentieren und Sicherheitslücken des RSA-Verfahrens herausfinden.

Erweitertes RSA-Kryptosystem

Ein Anwendungsbeispiel 1/2

Primzahlerzeugung

- Zunächst einmal generieren wir uns einen Schlüssel, den es zu knacken gilt.
- Dazu bietet das Plugin die Option „Schlüssel verwalten“.
- Dort wählen wir die Primzahlen p und q sowie ein zufälliges e .
- Schließlich wird der Schlüssel in einem Keystore gespeichert, wozu man unten rechts ein Passwort eingeben muss.
- Nachdem wir einen Schlüssel für die Identität „Alice Whitehat“ erstellt haben, möchten wir den Schlüssel angreifen (versuchen zu knacken). Beim RSA-Verfahren bedeutet dies, die Primfaktorzerlegung der Zahl $n = p \cdot q$ zu finden.
- Da Alice ihre Schlüssel kennt, wechseln wir die Sichtweise, indem wir zum Reiter „Bob Whitehat“ wechseln. (Die Identitäten Bob und Alice sind standardmäßig in JCT vorhanden.)



Erweitertes RSA-Kryptosystem

Ein Anwendungsbeispiel 2/2

Der Angriff

- Bob White kann jetzt den öffentlichen Schlüssel von Alice angreifen.
- Dazu klickt man als Bob den Button „Öffentlichen Schlüssel angreifen“ und wählt den passenden Schlüssel von Alice. Den soeben generierten Schlüssel erkennt man an der Bitlänge (hier 20 bit).
- Aufgrund der geringen Bitlänge kann anschließend über den Button „Schlüssel attackieren“ der von Alice generierte Schlüssel ohne Kenntnis der Primzahlen p und q wieder faktorisiert werden.
- Hier wird dazu nur ein Brute-Force-Angriff auf den Schlüssel angewendet.
- Eine Bitlänge von nur 20 bit ist bei weitem nicht ausreichend für die Sicherheit des RSA-Verfahrens.

Alice Whitehat - 1024Bit - RSAPublicKey - KeyID: 9
Alice Whitehat - 20Bit - RSAPublicKey - KeyID: 1
Erika Mustermann - 1024Bit - RSAPublicKey - KeyID: 5
Erika Mustermann - 1024Bit - RSAPublicKey - KeyID: 7
Alice Whitehat - 1024Bit - RSAPublicKey - KeyID: 6
Alice Whitehat - 1024Bit - RSAPublicKey - KeyID: 8
✓ Alice Whitehat - 20Bit - RSAPublicKey - KeyID: 10
Erika Mustermann - 1024Bit - RSAPublicKey - KeyID: 2

Mit Klick auf "Schlüssel attackieren" wird versucht, den Modulus N von Alice Whitehat zu faktorisieren:
Bitlänge von N: 20bit
N: 962261

Parameter	Wert
p	971
q	991
e	733
d	233197

Erweitertes RSA-Kryptosystem

Lernziel



Fazit

- Faktorisierungs-Angriffe können Schlüssel mit kurzer Bitlänge sehr schnell faktorisieren. Beispielsweise kann ein Modulus n mit 64 bit (die Binärdarstellung hat 64 Stellen, was circa 20 Dezimalstellen entspricht, wie z.B. die Zahl $2^{64}-15$) in weniger als 1 Sekunde mit einem aktuellen Notebook (Intel Core i7 2,4GHz) faktorisiert werden.
- Falls ein Angreifer eine Faktorisierung des Modul n finden kann, kann er die gesendeten Nachrichten auch entschlüsseln.
- Erst Bitlängen von 2048 bit werden heutzutage als sicher bewertet.

Und mehr ...

- Das Plugin ermöglicht es, Nachrichten mit dem RSA-Verfahren verschlüsselt an einen Gesprächspartner zu versenden.

SETUP-Angriff auf die RSA-Schlüsselgenerierung (Kleptographie)

Die Idee

Problematik

- Es existieren einige „Backdoor“-Angriffe, welche das RSA-Verfahren unsicher machen können.
- Einstiegspunkt der meisten dieser Angriffe auf das RSA-Verfahren ist die Schlüsselgenerierung, denn dabei müssen zufällige Primzahlen erzeugt werden. Auf das „zufällige“ Ergebnis dieser Generatoren muss vertraut werden können, dies ist jedoch nicht immer möglich.
- Dies ist auch bei dem sogenannten SETUP-Angriff (engl. "secretly embedded trapdoor with universal protection", d.h. ein geheim eingebauter, universal geschützter Falltürangriff) der Fall.

Hierzu eine kurze Zusammenfassung des Angriffs:

Funktionsweise

- Es werden zusätzliche Werte und Schlüssel in das System injiziert.
- Die öffentlichen Schlüssel des Verfahrens werden so beeinflusst, dass aus diesen direkt Informationen für die Entschlüsselung gewonnen werden können.
Ohne die konkrete Implementierung der Schlüsselgenerierung zu kennen, erscheinen diese Werte hingegen weiterhin als zufällig.

SETUP-Angriff auf die RSA-Schlüsselgenerierung

Die Implementierung in JCT



Im Menü

„Visualisierungen“ \ „Kleptographie“

Funktionsweise im Detail

- Allgemein werden beim RSA-Verfahren zwei zufällige, geheime Primzahlen P und Q benötigt, deren Produkt das Modul $N = P \cdot Q$ bildet. Der Wert N ist öffentlich.
- Bei dem Angriff wird zunächst die Primzahl P erzeugt und diese mit dem öffentlichen Schlüssel des Angreifers verschlüsselt. Die Primzahl Q wird anschließend so gewählt, dass die ersten Zahlen des Moduls N der verschlüsselten Primzahl entsprechen.
- Durch dieses Wissen kann der Angreifer mit seinem privaten Schlüssel die Primzahl P zurückgewinnen, und das Verfahren ist geknackt.
- Da nur die verschlüsselte Primzahl P im Modul N enthalten ist und P zufällig gewählt wurde, erscheint das Modul N als Produkt von P und Q zufällig.
Da P für jedes neue Schlüsselpaar neu erzeugt wird, ist der Angriff nicht erkennbar ohne ein Reverse-Engineering des Codes zur Schlüsselgenerierung.

SETUP-Angriff auf die RSA-Schlüsselgenerierung

Ein Anwendungsbeispiel 1/2

Der Angriff teilt sich in zwei Schritte auf: die Schlüsselgenerierung und das Entschlüsseln des Angreifers.

Die Schlüsselgenerierung

- Wähle im Dropdown-Menü das Verfahren „Angriff 4: SETUP“.
- Zunächst müssen die beiden Schlüssel des Angreifers generiert werden. Dies geschieht über den Button „Neue Schlüssel für den Angreifer erzeugen“.
- Anschließend können die üblichen, für das RSA-Verfahren benötigten Primzahlen P und Q erzeugt werden.
- Die Primzahl Q wird dabei so gewählt, dass der Modul N die verschlüsselte Primzahl P enthält (in der Abbildung gelb hervorgehoben).
- Nach dem Berechnen von N und D kann im Textfeld im unteren Drittel des Plugins ein Klartext verschlüsselt werden.
- Durch den Button „Öffentlichen Schlüssel und Geheimtext speichern“ ist es anschließend möglich, im Reiter „SETUP-Angriff“ den Geheimtext wieder zu entschlüsseln.

Screenshot of the 'Schlüsselerzeugung' (Key Generation) interface in JCryptTool 1.0. The interface is divided into 'Einstellungen' (Settings) and 'Standard-Kryptosystemwerte' (Standard Cryptosystem Values). Under 'Einstellungen', 'Verfahren' is set to 'Angriff 4: SETUP', 'Bitlänge der Schlüssel' is 64 (decimal), and 'Hexadezimal' is selected. Under 'Zusätzliche Kryptosystemwerte', the 'Neue Schlüssel für den Angreifer erzeugen' button is visible. Below that, the values for 'N des Angreifers' (df9b5945), 'E des Angreifers' (8138b321), 'Verschlüsselte P' (ba69fea8), and 'N' (vorläufiges Modul) (ba69fea80b64e2af) are shown. Under 'Standard-Kryptosystemwerte', the 'Alles auf einmal erzeugen' button is visible. Below that, the values for 'Primzahlen P und Q erzeugen' (P: da7f23db, Q: da6920c7), 'N berechnen' (N = P * Q: ba69fea7f0233f3d), 'E (öffentlicher Exponent)' (10001), and 'D berechnen' (D (privater Exponent): 4037dd9b049297dd) are shown. There are also buttons for 'Neues E erzeugen' and 'Standard-E wiederherstellen'.

SETUP-Angriff auf die RSA-Schlüsselgenerierung

Lernziel

Fazit

- Durch geschicktes Eingreifen in die Schlüssel-Erzeugung ist es für einen Angreifer möglich, den verschlüsselten Geheimtext mit der Hilfe seines eigenen privaten Schlüssel zu entschlüsseln.
- Fast alle effektiven Angriffe auf RSA setzen bereits bei der Schlüsselgenerierung an. Daher müsste das Vertrauen in die Schlüsselgenerierung, die oft von einer Zertifizierungsstelle (engl. „Certificate Authority“) oder in einem Hardware-Security-Module (HSM) durchgeführt wird, gewährleistet sein.
- Das Modul N scheint bei der Erzeugung zweier unterschiedlicher Schlüsselpaare zufällig, da die Primzahlen P und Q jeweils neu gewählt werden. Daher ist es allgemein schwierig, den Angriff anhand des Output aufzudecken -- ohne Reverse Engineering zu betreiben.
- Für den Angriff wird nur der öffentliche Schlüssel des Angreifers benötigt, so dass ein Aufdecken des Angriffes keine weitere Unsicherheit für die Kommunikation des Angreifers darstellt.

Zero-Knowledge-Protokoll: Fiat Shamir



Die Idee

Problematik

- Eine Person A möchte eine Person B davon überzeugen, dass sie ein Geheimnis kennt, welches auch Person B kennt.
- Die Überprüfung soll dabei so stattfinden, dass das eigentliche komplette Geheimnis nicht preisgegeben werden muss. Dadurch ist es möglich, den Abgleich öffentlich durchzuführen und einem Mithörer ist es nicht möglich, das Geheimnis zu lüften.
- Eine Lösung für diese Problemstellung wird Zero-Knowledge-Protokoll genannt.
- Wichtig ist, dass die Beteiligten eines Zero-Knowledge-Protokolls ehrlich sind. Dies bedeutet, dass eine dritte Person C die Person B nicht davon überzeugen kann, vorzugeben das Geheimnis zu kennen, obwohl sie es nicht kennt.

In diesem Anwendungsbeispiel zeigen wir das Zero-Knowledge-Protokoll Fiat Shamir. Es gibt jedoch noch weitere, z.B. Feige Fiat Shamir, oder eine Variante über einen Graphenisomorphismus.

Zero-Knowledge-Protokoll: Fiat Shamir

Die Implementierung in JCT



Im Menü

„Visualisierungen“ \ „Fiat Shamir“

Funktionsweise

- Beim Verfahren Fiat Shamir wird ausgenutzt, dass im Restklassenring modulo n die Quadratwurzel einer Zahl nur über die Primfaktorenzerlegung der Zahl n gefunden werden kann.
- Ist n zusätzlich das Produkt zweier geheimer Primfaktoren p und q , und sind p und q genügend groß, dann ist eine Primfaktorzerlegung von n nicht effizient möglich.
- Da das Verfahren auf Zahlen operiert, muss das Geheimnis s als Zahl gegeben sein.
- Person A veröffentlicht die Zahl $v = s^2 \bmod n$, generiert eine Zufallszahl $r < n$ und bekommt von Person B eine Zufallszahl b übersendet, die 0 oder 1 ist. Person B erhält die Zahl $x = r^2 \bmod n$.
- Person A berechnet $y = rs^b \bmod n$ und schickt diese Zahl an Person B. Person B verifiziert, ob die Gleichung Zahl $y^2 = xv^b \bmod n$ gilt. Ist diese erfüllt, so gilt das Geheimnis als verifiziert.

$$y^2 = (rs^b)^2 = r^2s^{2b} = xv^b \bmod n$$

Zero-Knowledge-Protokoll: Fiat Shamir

Ein Anwendungsbeispiel 1/2

Als Prüfer

- Wähle oben den Radio-Button „Prüfer“.
- Zunächst müssen die beiden Primzahlen p und q generiert werden, deren Produkt ist der öffentliche Modul n . Zusätzlich muss das Geheimnis s erstellt werden.
- Unter „Ablauf“ können dann die Schritte, die zur Verifizierung benötigt werden, durchgeführt werden.
- Alle Werte – öffentliche sowie private – die bei dem Verfahren berechnet werden, sind im unteren Teil des Plugins aufgeführt.
- Da die geprüfte Person Alice hier tatsächlich das Geheimnis kennt, wird ihr Kommunikationspartner dies positiv verifizieren (grüner Hinweis im Screenshot unten links).

Situation

Prüfer Wählen Sie zwei Primzahlen p und q . Die Zahl $n = pq$ ist der öffentliche Modul. Alice kennt eine Zahl $s < n$ und $v = s^2 \bmod n$. v wird öffentlich bekannt gegeben.

Angreifer

Primzahleingabe

p : 197 q : 151 n : 29747

Buttons: Primzahlen generieren, Geheimnis erstellen, Primzahlen übernehmen

Ablauf

Zufallszahl erstellen: Alice generiert eine Zufallszahl $r < n$ und schickt $x = r^2 \bmod n$ an Bob.

b generieren: Bob generiert ein Zufallsbit b aus $\{0,1\}$ und sendet b an Alice.

Antwort berechnen: Alice berechnet eine Antwort y und sendet diese an Bob. $y = rs^b$.

Verifizieren: Bob verifiziert Alices Antwort. Er überprüft, ob gilt: $y^2 = xv^b$.

Buttons: Zurücksetzen, Neuer Durchlauf, Mehrmals ausführen

Information

Bob	Alice	Öffentlich
b : 1	Geheim	$v = s^2 \bmod n$ 5286
y^2 : 7490	r : 6138	y : 18520
$x v^b$: 7490	s : 1360	$x = r^2 \bmod n$ 15342

Wurde verifiziert

Zero-Knowledge-Protokoll: Fiat Shamir

Ein Anwendungsbeispiel 2/2

Als Angreifer

- Das Verfahren bietet jedoch auch Angreifern die Möglichkeit vorzutäuschen, sie besäßen das Geheimnis. Dies lässt sich durch die Auswahl „Angreifer“ nachspielen und verstehen.
- Durch geschickte Wahl der Werte x und y , mit deren Hilfe die Glaubwürdigkeit überprüft wird, lässt sich in 50% der Fälle vortäuschen, dass der Angreifer das Geheimnis kennt.
- Dies lässt sich in diesem Szenario durchspielen. Durch n -faches Ausführen wird letztlich mit einer Wahrscheinlichkeit $1-(0,5)^n$ aufgedeckt, dass es sich nicht um den wahren Geheimnisträger handelt.
- Je häufiger der Test wiederholt wird, umso sicherer wird ein Angreifer entdeckt.

The screenshot shows the Fiat Shamir simulator interface. The 'Situation' section has 'Angreifer' selected. The 'Primzahleingabe' section shows $p = 131$ and $q = 229$, with $n = 29999$. The 'Ablauf' section shows the following steps:

- Zufallszahl erstellen: Carol generiert eine Zufallszahl $r < n$ und ein Bit c aus $\{0,1\}$. Sie sendet $x = r^2 v^a - c$ mod n an Bob.
- b generieren: Bob generiert ein Zufallsbit b aus $\{0,1\}$ und sendet b an Carol.
- Antwort berechnen: Carol berechnet eine Antwort y und sendet diese an Bob: $y = r$.
- Verifizieren: Bob verifiziert Carols Antwort. Er überprüft, ob gilt: $y^2 = x v^a b$.

The 'Information' section shows the following values:

Bob		Carol		Öffentlich		
b :	0	Geheim	r :	6711	$v = s^2 \text{ mod } n$:	6955
y^2 :	9022	s :	?	y :	6711	
$x v^a b$:	15926	c :	1	$x = r^2 v^a - c$:	15926	

The status 'Wurde nicht verifiziert' is shown in red.

n	1	2	3	4	5	6	7	8	9	10
P(n)	0,5	0,75	0,875	0,9375	0,96875	0,984375	0,9921875	0,99609375	0,998046875	0,999023438



Fazit

- Zero-Knowledge-Protokolle sind Verfahren, die eingesetzt werden, um jemand anderen zu überzeugen, dass man ein Geheimnis kennt. Dabei soll das Geheimnis jedoch nicht offen gelegt werden.
- Das Fiat-Shamir-Protokoll ist ein solches Verfahren.
- Es ist wichtig zu wissen, dass ein Angreifer das Verfahren mit Wahrscheinlichkeit $(0,5)^n$ hintergehen kann (n ist die Anzahl der Wiederholungen des Tests). Je öfter das Verfahren wiederholt wird, desto besser ist die Qualität der Aussage.
- Hinweis: Wenn sehr große Zahlen effizient und schnell in ihre Primfaktoren zerlegt werden können, dann ist das Verfahren nicht mehr sicher (d.h. dann gelten die oben genannten Wahrscheinlichkeiten nicht).

Android-Mustersperre (AUP)

Die Idee



Problematik

- Smartphones bieten heutzutage – neben dem Telefonieren und Nachrichten verschicken – viele weitere Funktionen, wie z.B. Mails bearbeiten, Notizen erstellen oder Online-Banking machen. Das Nutzen solcher Funktionen führt dazu, dass viele vertrauliche Daten auf dem Smartphone (oder in einer Cloud) gespeichert werden.
- Wer sein Smartphone einmal verloren hat, fragt sich häufig, ob jemand an seine sensiblen Daten herankommt. Wie sicher ist die Sperre des Smartphones? Was unterscheidet eine einfache PIN-Eingabe von der bei Android zum Einsatz kommenden Mustersperre ^[1].
- Die Mustersperre des Betriebssystems Android ist in JCT visualisiert, und in der Onlinehilfe werden die Sicherheitsbewertungen dazu dargestellt und mit anderen Mustersperren verglichen.

[1] AUP = Android Unlock Pattern

Android-Mustersperre

Die Implementierung in JCT



Im Menü

„Visualisierungen“ \ „Android-Mustersperre (AUP)“

Funktionsweise

- Die Android-Mustersperre kann als Bildschirmsperre bei Smartphones mit dem Betriebssystem Android benutzt werden. Typischerweise sind dazu neun Punkte auf dem Bildschirm im Quadrat angeordnet, und der Benutzer kann durch Verbinden der Punkte (nach bestimmten Regeln) ein Muster anlegen, das er dann zum Entsperren des Smartphones eingeben muss.
- Die Visualisierung bietet die Möglichkeit, verschiedene Muster auf ihre Sicherheit zu prüfen. Dabei wird ein Sicherheitsindikator angezeigt, der anzeigt, wie viele verschiedenen Kombinationen mit der benutzen Anzahl an Punkten auf dem Muster möglich ist.

Android-Mustersperre

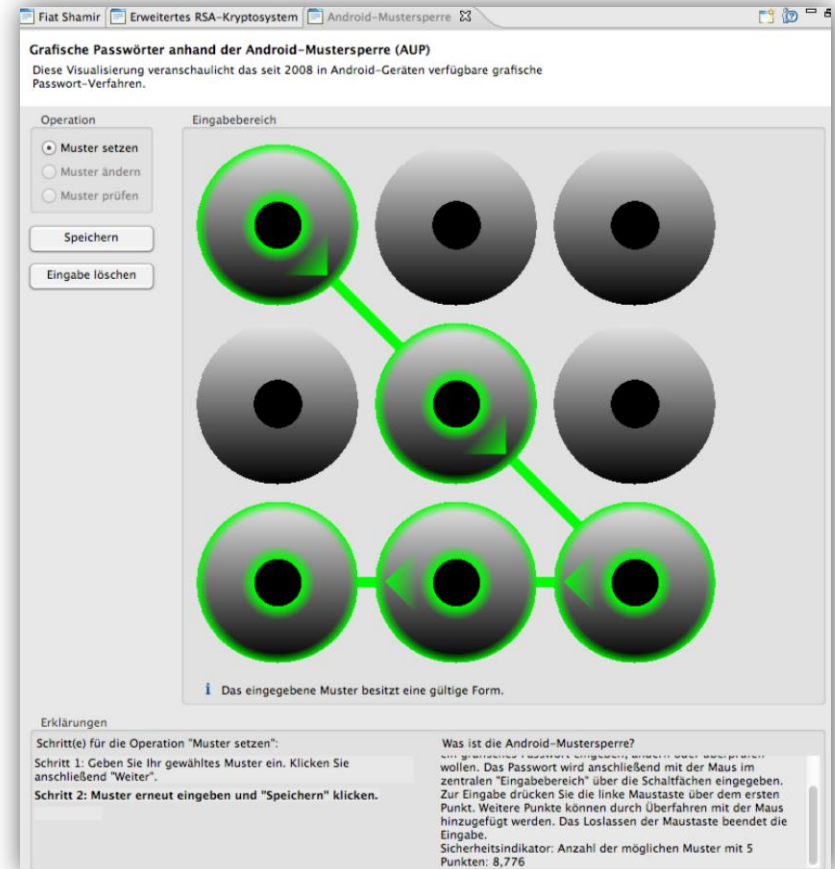
Ein Anwendungsbeispiel

Muster setzen

- Startet man die Visualisierung, sieht man den typischen Sperrbildschirm von Android.
- Nun kann man ein Muster festlegen, indem man auf einen der Punkte klickt und anschließend die Maus über die andere Punkte bewegt.
- Um das Muster abzuschließen, muss man auf den Endpunkt des Musters klicken.
- Ist das Muster angelegt, wird im rechten Textfeld der Sicherheitsindikator angezeigt. Dieser gibt einen Hinweis darauf, wie sicher das Passwort ist. Z.B gibt es für ein Muster mit 5 Punkten 8776 verschiedene Kombinationsmöglichkeiten.

Muster ändern, Muster prüfen

- Das Plugin bietet zusätzlich noch die Möglichkeit, das Muster zu speichern und danach ein anderes Muster mit dem gespeicherten zu vergleichen.
- Das gespeicherte Muster kann auch geändert werden. Hierzu wird das zuletzt gespeicherte Muster benötigt. Kennt man das Muster nicht mehr, kann man das Plugin einfach komplett zurücksetzen.





Fazit

- Für das AUP-Muster ist die Reihenfolge der besuchten Felder entscheidend.
- Ein Muster der Android-Mustersperre muss gewisse Regeln erfüllen, beispielsweise darf jedes Feld nur einmal besucht werden.
- Durch diese (und weitere) Einschränkungen reduziert sich die Anzahl der möglichen Muster. Insgesamt gibt es 389.112 verschiedene Muster.
- Vergleicht man das AUP-Muster mit einer vier- bis neunstelligen PIN aus den Ziffern 1 bis 9, bei der jede Ziffer höchstens einmal vorkommt, so gibt es hierfür 985.824 PIN-Kombinationen. Bei der Android-Mustersperre gilt die Regel, dass Verbindungen zwischen zwei Feldern, deren Verbindungslinie ein ungenutztes Feld schneidet, unzulässig sind. Würde man diese Regel vernachlässigen, so hätte die Mustersperre genauso viele Möglichkeiten wie die PIN, bei der jede Zahl höchstens einmal vorkommt.

Kaskaden mit dem Aktionen-Fenster

Die Idee

Funktionalität

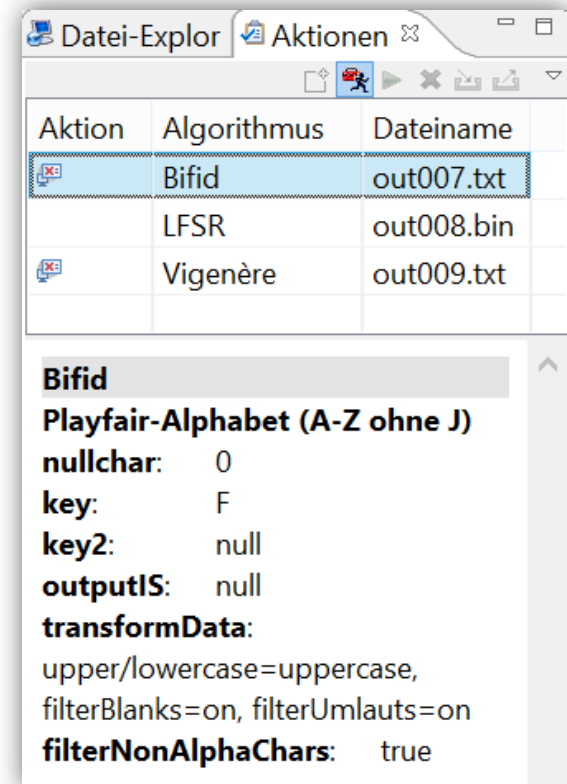
- Mit dem Aktionen-Fenster lassen sich Abfolgen (Kaskaden) von Aufrufen von Krypto-Verfahren aufnehmen und erneut ausführen (dies ist also ein Rekorder und Player für die JCT-Funktionen).
- Es können beliebig viele Algorithmen aus der Standard-Perspektive aufgenommen und wieder abgespielt werden.
- Kaskaden klassischer Krypto-Verfahren lassen sich auch mit der Krypto-Konsole abbilden (siehe Folie [73](#)).

Anwendungsbeispiele

- Dateien lassen sich schnell mit den denselben Algorithmen und Einstellungen und gleicher Reihenfolge ver- bzw. entschlüsseln.
- Mit der Kaskadenfunktion lässt sich die Kommutativität, also die Vertauschbarkeit der Reihenfolge zweier Verschlüsselungsverfahren, leicht ausprobieren (siehe Folien [66](#) ff).

Mit dem Aktionen-Fenster kann man Prozeduren automatisieren und abspielen – ähnlich wie mit Stapeldateien auf der Kommandozeile.

Es kann vorkommen, dass aufgenommene Kaskaden nicht dasselbe Ergebnis liefern wie bei der Aufnahme.





Kaskaden mit dem Aktionen-Fenster

Die Implementierung in JCT




Im Menü

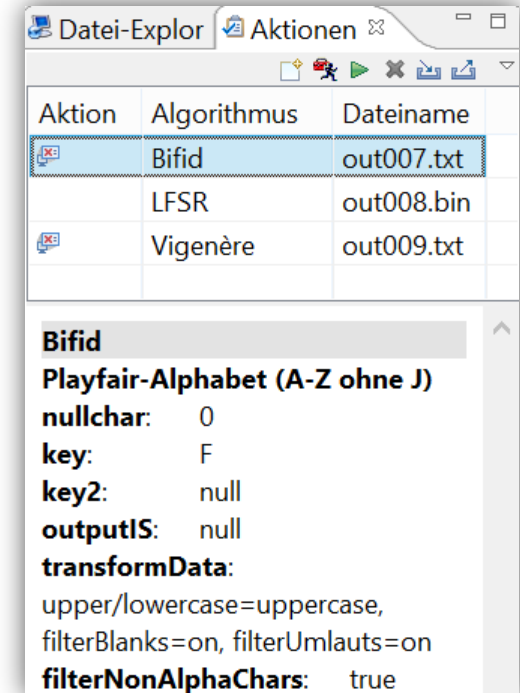
„Fenster“ \ „Sicht anzeigen“ \ „Aktionen“

Erstellen einer Aufnahme

- Um eine Kaskaden-Aufnahme zu starten, den Button  drücken.
- Alle Algorithmen, die nun ausgeführt werden, werden gespeichert.
- Zum Beenden der Aufnahme erneut den Button  drücken.

Aufnahme bearbeiten, speichern und ausführen

- In der Liste unterhalb der Icon-Leiste sind alle Algorithmen in der Reihenfolge aufgelistet, wie sie ausgeführt wurden.
- Wählt man eine Zeile in der Liste aus, werden im unteren Bereich des Fensters Details zu den Einstellungen der Ver-/Entschlüsselung angezeigt.
- Nun lässt sich die erstellte Kaskade auf eine in JCT geöffnete Datei durch Drücken von  anwenden.
- Mit den beiden Buttons  und  lassen sich die Kaskaden importieren bzw. exportieren (laden und abspeichern).





Kaskaden mit dem Aktionen-Fenster

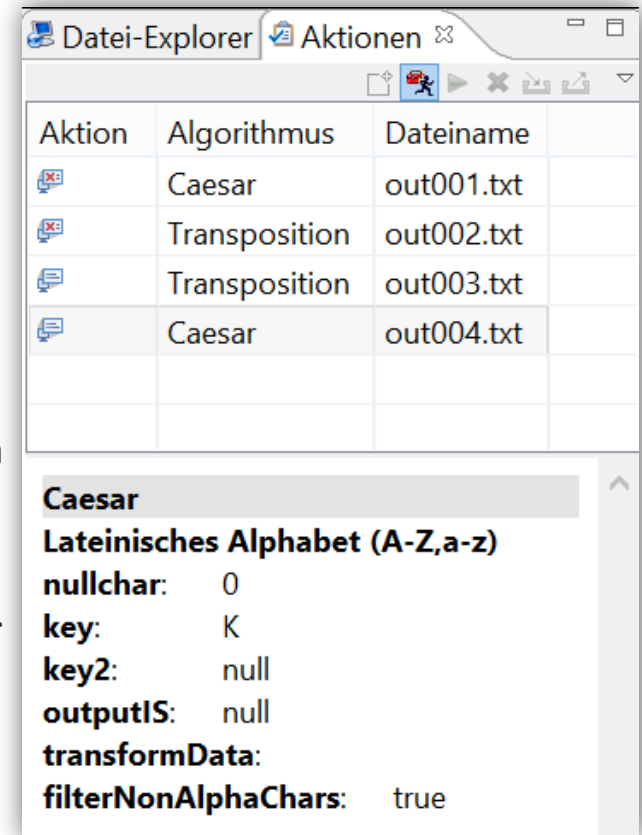
Ein Anwendungsbeispiel 1/3

An einem Beispiel wird gezeigt, dass man die Reihenfolge des Caesar- und des Transpositionsverfahrens beim Entschlüsseln vertauschen kann (**Kommutativität**).

Eine erste Aufnahme

- Start der Kaskaden-Aufnahme mit .
- **Verschlüsselung** eines beliebigen Textes mit Caesar:
„Algorithmen“ \ „Klassisch“ \ „Caesar“
- Hinzufügen einer Transpositions-**Verschlüsselung**:
„Algorithmen“ \ „Klassisch“ \ „Transposition“
- Anwenden einer Transpositions-**Entschlüsselung**, die die letzte Verschlüsselung wieder rückgängig macht: Dazu sind die gleichen Einstellungen wie bei der entsprechenden Verschlüsselung zuvor zu wählen, nur diesmal mit der Richtung „Entschlüsseln“.
- **Entschlüsselung** der zuerst ausgeführten Caesar-Verschlüsselung.
- Stoppen der Aufnahme mit .

Das Aktionen-Fenster sollte nun wie rechts abgebildet aussehen.



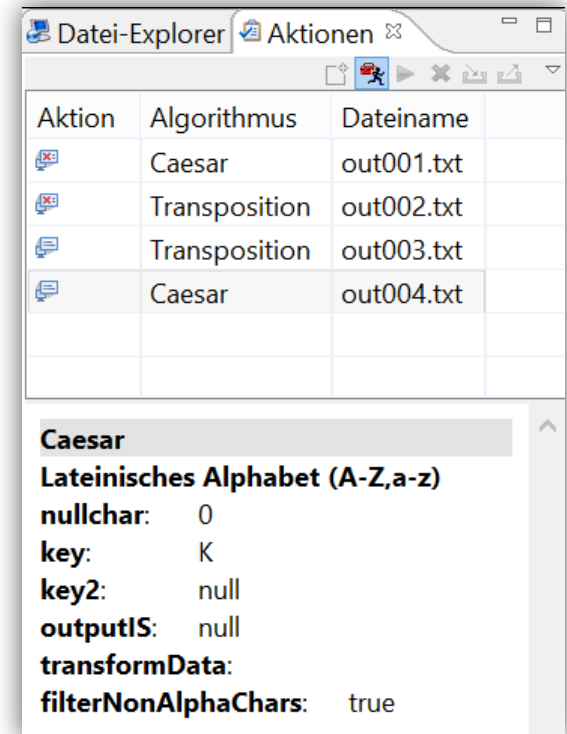
Kaskaden mit dem Aktionen-Fenster

Ein Anwendungsbeispiel 2/3

Die Kaskade, die wir auf der letzten Folie erstellt haben, sollte uns jetzt jeden Text wieder unverschlüsselt anzeigen, denn sie entschlüsselt gleich wieder, was sie zuvor verschlüsselte.

Die aktuelle Abfolge der Algorithmen

- Es sollte folgende Abfolge von Krypto-Operationen vorliegen, wobei hier E für die Encryption (Verschlüsselung) und D für Decryption (Entschlüsselung) steht.
 - > E (Caesar)
 - > E (Transposition)
 - > D (Transposition)
 - > D (Caesar)
- Erkennbar ist die Zwiebel-artige Reihenfolge der Verschlüsselungsalgorithmen und die Anwendung des jeweils inversen Algorithmus. Eine solche Struktur liefert mit allen Algorithmen die Identitätsabbildung – der Klartext bleibt also invariant.
- Damit kommt die **Frage** auf:
Inwieweit lassen sich die Entschlüsselungsalgorithmen in ihrer Reihenfolge vertauschen, so dass weiterhin in den Klartext "entschlüsselt" wird?



Kaskaden mit dem Aktionen-Fenster


Ein Anwendungsbeispiel 3/3

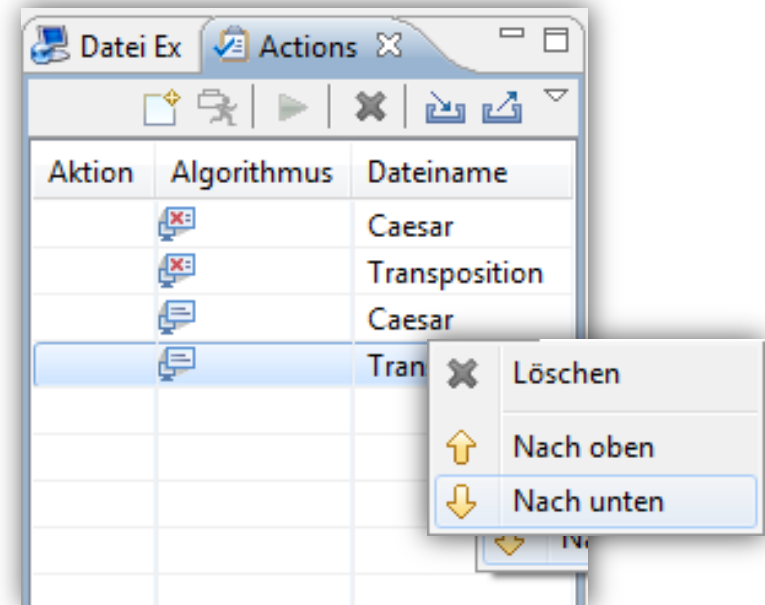
Wir wollen nun die Reihenfolge der Entschlüsselungs-Algorithmen ändern und beobachten, was passiert.

Aufnahmen umsortieren

- Ein Rechtsklick auf eine Zeile (z.B. die Caesar-Entschlüsselung) und die Auswahl „Nach oben“ und „Nach unten“ erlauben es, die Reihenfolge der Operationen zu ändern.

Eine neue Reihenfolge

- Damit ergibt sich die geänderte Reihenfolge der Algorithmen:
 - > E (Caesar)
 - > E (Transposition)
 - > D (Caesar)
 - > D (Transposition)
- Öffne eine beliebige Textdatei in JCT.
- Wende mit  unsere neue Kaskade auf die Datei an.



Was passiert nun mit dem Klartext?

Funktioniert dies auch mit anderen Verschlüsselungsverfahren?

Kaskaden mit dem Aktionen-Fenster

Lernziel



Fazit

- Die Kaskadenfunktion eignet sich hervorragend, um Abfolgen kryptografischer Operationen zu speichern und auf unterschiedliche Dateien automatisch anzuwenden.

Fazit anhand eines Beispiels

- Wurde auf einen Klartext die Caesar- und die Transpositionsverschlüsselung angewandt, dann ist die Reihenfolge bei der Entschlüsselung beliebig. Diese Verfahren sind bei der Entschlüsselung kommutativ.
- Dies ist möglich, da das Caesar-Verfahren jeden einzelnen Buchstaben um eine feste Anzahl an Buchstaben im Alphabet verschiebt. Das Transpositionsverfahren permutiert jeden Buchstaben im Text. Beide arbeiten exakt auf denselben (Teil-)Objekten.
Genauso wäre es auch, wenn man statt Caesar eine monoalphabetische Substitution nimmt.
- Viele Verfahren (z.B. ADFGVX und Playfair) arbeiten mit einer sogenannten "Fraktionierung". Sie substituieren beispielsweise Buchstabenpaare, transponieren aber Einzelbuchstaben. Dann sind die Substitution und die Transposition nicht mehr kommutierbar.

Variable Alphabete für klassische Algorithmen

Die Idee

Benutzer-definierte Alphabete

- Der verschlüsselte Text hängt bei den meisten klassischen Verschlüsselungs-Algorithmen (beispielsweise Vigenère) vom verwendeten Klartext-Alphabet ab.
- Häufig verwendete Alphabete sind Groß- und Kleinbuchstaben (A-Z, a-z), mit oder ohne Ziffern (0-9).
- Viele Kryptografie-Tools beschränken sich auf einen festen Satz von Alphabeten oder Zeichen, bei dem Rest müssen die Buchstaben des Alphabets einzeln angegeben werden.
- Es ist wünschenswert, dass der Benutzer die einzelnen Verfahren mit eigenen Alphabeten schnell und unkompliziert ausprobieren kann, um deren Bedeutung besser zu verstehen.
- Lösung mit JCrypTool:
 - Bei den klassischen Verfahren wird immer ein Wizard für die Erstellung von eigenen Alphabeten angeboten.
 - Eigene Alphabete können durch einfaches Zusammenfügen häufig benutzter Bausteine erstellt werden.

Alphabeterstellung

Alphabet
Geben Sie Namen und Zeichen des neuen Zeichensatzes ein

1) Geben Sie einen Namen für das Alphabet an:
Deutsches Alphabet

2) Geben Sie die Zeichen des Alphabets an:
 Ein Alphabet aus 'Alphabet-Bausteinen' zusammenstellen
 Alphabet manuell eingeben

Klicken Sie auf die Alphabetbausteine, um ein neues Alphabet zusammenzustellen:

A-Z a-z 0-9 äöüß ÄÖÜ (Leerzeichen) Zeilenumbruch (Unix/Windows) .,:;-!? ASCII-Zeichen außer dem lateinischen Alphabet

Neuer Alphabetbaustein

Ausgewählte Bausteine und das daraus resultierende Alphabet:

A-Z a-z äöüß ÄÖÜ

(Klicken Sie mit der rechten Maustaste auf die Bausteine für weitere Optionen)

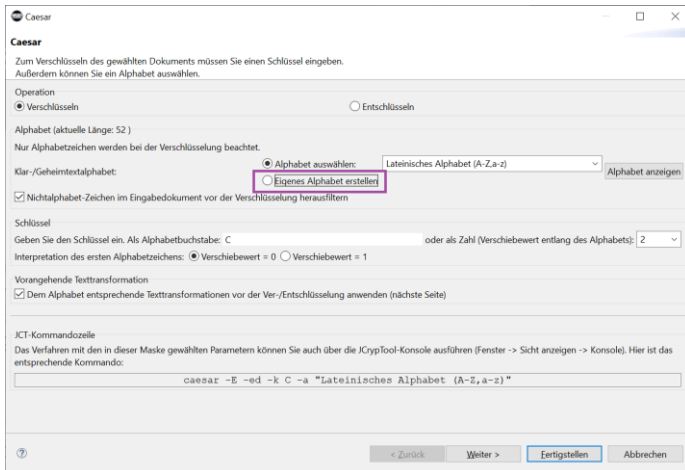
Ergebnis (zusammengestelltes Alphabet):
Alphabetzeichen: ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyzäöüßÄÖÜ
Alphabetlänge: 59

Variable Alphabete für klassische Algorithmen

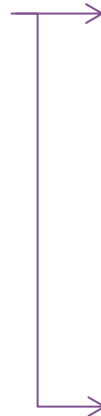
Die Implementierung in JCT 1/2

Ein eigenes Alphabet anlegen

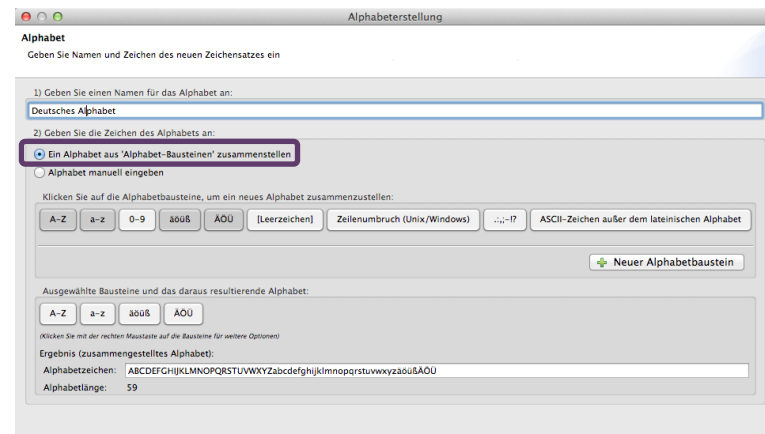
- Sofern das Krypto-Verfahren dies unterstützt, kann der Benutzer eigene Alphabete für die Ver- und Entschlüsselung definieren.



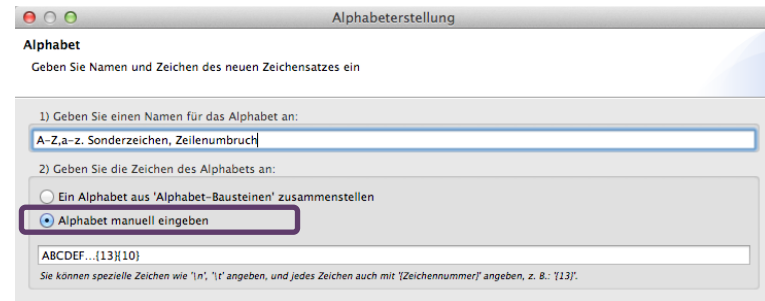
- Sonderzeichen, die nicht auf der Tastatur erscheinen, können über ihren ASCII-Wert innerhalb von geschweiften Klammern ebenfalls eingegeben werden, z.B. {10} für den Zeilenumbruch.



Baustein-Zusammenstellung



Manuelle Eingabe



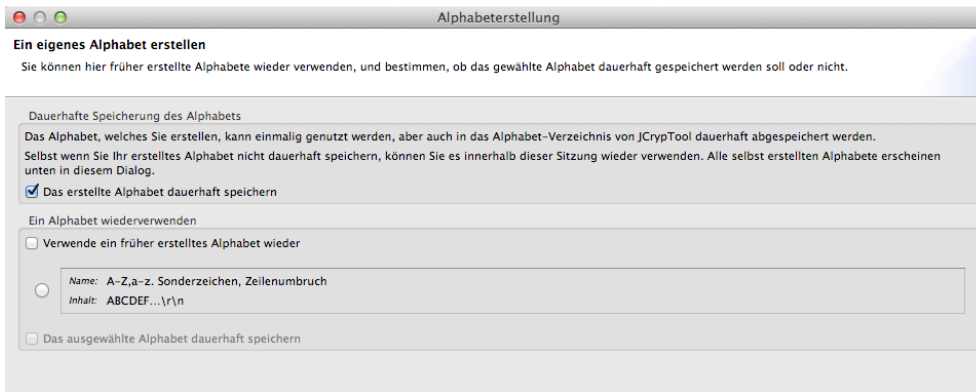
Variable Alphabete für klassische Algorithmen

Die Implementierung in JCT 2/2



Weitere Hinweise

- Selbst definierte Alphabete können zur permanenten Nutzung abgespeichert werden.
- Solange man JCT nicht beendet, können in einer JCT-Sitzung auch selbstdefinierte und noch nicht abgespeicherte Alphabete wieder aufgerufen werden (ohne Neueingabe).



- Die gespeicherten Alphabete lassen sich in den globalen Einstellungen von JCT verwalten und nachträglich bearbeiten:
 - Windows + Linux: „Fenster“ \ „Benutzervorgaben“
 - MacOS: „JCrypTool“ \ „Einstellungen“

Variable Alphabete für klassische Algorithmen

Lernziel



Fazit

- Neue Alphabete können in JCrypTool spielerisch aus Bausteinen zusammen gefügt werden.
- Die Verwendung der Alphabete ist mächtig, da auch alle Sonderzeichen verwendet werden können.
- Dadurch, dass das Erstellen, Verwenden und Wiederverwenden von eigenen Alphabeten für den Nutzer effizient und leicht verständlich ist, wird zum Ausprobieren angeregt.
- Klassische Verfahren sind in den meisten herkömmlichen Krypto-Tools auf ein festes Set von verwendbaren Alphabeten eingeschränkt. JCrypTool ist hier maximal flexibel.

JCrypTool-Konsole für klassische Verfahren

Die Implementierung in JCT



Die Konsole

- Die klassischen Verfahren in JCrypTool können auch über eine Kommandozeile gestartet werden:

```
Konsole X
JCrypTool Krypto Konsole
Willkommen bei der JCrypTool-Konsole.
Durch Eingabe von 'help' oder 'help <Befehl>' erhalten Sie Hilfe.

JCrypTool=>help
In der JCrypTool-Konsole können Sie die kryptografischen Algorithmen von der Kommandozeile aufrufen.
Geben Sie 'help -l' ein, um eine vollständige (alphabetisch geordnete) Liste der Algorithmen zu erhalten.
Zu jedem Befehl liefert die Hilfe sowohl die Befehlssyntax als auch Aufrufbeispiele:

SYNTAX-HILFE:
Zu jedem Befehl gibt es eine kurze und eine ausführliche Syntaxausgabe, die man auf folgende Weise aufrufen kann:
Kurz:           Ausführlich:
'help <command>'   'HELP <command>'
'? <command>'     '?? <command>'
'<command> help'  '<command> HELP'
'<command> ?'     '<command> ??'

BEISPIELE:
Viele Befehle bieten Beispiele ihrer Benutzung an. Sie können sie mit 'help -x <command>' aufrufen.


Versuchen Sie doch einmal, die Beispiele dieses 'help'-Befehls (mit 'help -x help') aufzurufen!

JCrypTool=>
```

- Das Kommando „help“ gibt Auskunft über die Konsole im Allgemeinen (siehe oben).
- Zusätzlich kann man auch für jedes einzelne Verfahren Hilfe und Beispiele erhalten.



Beispiel Autokey-Vigenère

- Von der Kommandozeile lassen sich alle klassischen Verfahren sowohl auf Textdateien und Editor-Inhalte als auch auf als Kommandozeilenargument übergebenen Text anwenden.
- Aufruf der Konsole in der Icon-Leiste (unter dem Hauptmenü) über das ff. Icon: 
- Beispiel mit dem Autokey-Vigenère-Verfahren:
 - Hilfe und Beispiele aufrufen:

```
JCrypTool=>help autovigenere
Vigenère-Verschlüsselung, bei der mit Hilfe des Klartextes ein langer Schlüssel generiert wird.
Syntax:autovigenere [-a <ALPHABET>] -D | -E -ed | -f <FILE_PATH> | -t <TEXT> -k <KEY> [-noFi]
Beispiele zur Benutzung des Befehls erhalten Sie durch 'help -x autovigenere'.
Ausführlichere Hilfe erreichen Sie mit 'HELP autovigenere'.
Weitere Informationen zu dem Verfahren finden Sie in der JCrypTool Onlinehilfe.

JCrypTool=>help -x autovigenere
'autovigenere -E -ed -k akey'           -> Verschlüsselt den Text im aktiven Editor mit dem Schlüssel "akey"
'autovigenere -D -ed -k akey'           -> Entschlüsselt den Text im aktiven Editor mit dem Schlüssel "akey"
'autovigenere -E -a A-Z -t "TEST TEXT" -k AKEY' -> Verschlüsselt den Text "TEST TEXT" mit dem Schlüssel "AKEY", über dem Großbuchstaben-Alphabet
```

- Der obige Screenshot zeigt, wie die Kommandozeilenoptionen in der Konsolen-Hilfe erklärt werden (hier am Beispiel „HELP autovigenere“).



Ver- und Entschlüsselung mit Autokey-Vigenère

- Als Klartext nehmen wir „ZUGRIFFxCODExTAGJTT“, als Schlüssel nehmen wir „THEKEY“:

```
JCrypTool=>autovigenere -E -a a-zA-Z -t "ZUGRIFFxCODExTAGJTT" -k THEKEY  
sbKbMdeRifLJCQCUMXQ
```

- Die zweite Zeile zeigt den erstellten Geheimtext „sbKbMdeRifLJCQCUMXQ“, erzeugt mit dem Befehl „autovigenere -E -a a-zA-Z -t \"ZUGRIFFxCODExTAGJTT\" -k THEKEY“
- Durch den Austausch von „-E“ durch „-D“ im Befehl macht man die Verschlüsselung rückgängig:
„autovigenere -D -a a-zA-Z -t \"sbKbMdeRifLJCQCUMXQ\" -k THEKEY“

```
JCrypTool=>autovigenere -D -a a-zA-Z -t "sbKbMdeRifLJCQCUMXQ" -k THEKEY  
ZUGRIFFxCODExTAGJTT
```

JCrypTool-Konsole für klassische Verfahren

Lernziel 1/2

Vorteile der Verwendung der Konsole

- Parameter einer Operation (wie eben das Alphabet und der Schlüssel) können in der Kommandozeile bequem durch Copy&Paste noch einmal verwendet werden.
- Je mehr Parameter es gibt (nicht nur Alphabet, Schlüssel und das Filtern von Nichtalphabetzeichen), desto größer ist der Nutzen.
- Das Transpositionsverfahren z.B. hat eine große Anzahl von Parametern:
 - Für die erste und zweite Runde kann man jeden der ff. Parameter setzen (insgesamt also 6):
 - Einleserichtung
 - Ausleserichtung
 - Schlüssel
 - Alphabet
 - Filtern von Nichtalphabetzeichen
- Eine per Maske erstellte Kommandozeile enthält alle gewünschten Parameter.
- Lädt ein zum Wiederverwenden bzw. leichten Abändern durch Copy&Paste.

Entsprechender Konsolenbefehl:

```
transposition -E --editor -a „Printable ASCII“ --key CAD4  
-t1ReadIn rw -t1ReadOut cw --key2 RT334 -t2ReadIn rw -t2ReadOut cw
```



Detaillierte Hilfe über die Konsole

- Konsolen-Hilfe zum Transpositionsverfahren:

```
JCrypTool=>HELP transposition
Vertauscht die Buchstaben im Klartext untereinander (spaltenweise Transpositionsverschlüsselung mit einstellbarer Einlese-/Ausleserichtung).
Syntax:transposition [-a <ALPHABET>] -D | -E -ed | -f <FILE_PATH> | -t <TEXT> -k <KEY> [-k2 <KEY>] [-noFi] [-t1ReadIn
    <ORDER = 'cw'/'rw'>] [-t1ReadOut <ORDER = 'cw'/'rw'>] [-t2ReadIn <ORDER = 'cw'/'rw'>] [-t2ReadOut <ORDER =
    'cw'/'rw'>]
Erläuterung der Optionen:
-a,--currentAlphabet <ALPHABET>
    Eines von ASCII, a-zA-Z, A-Z, a-z, Playfair,
    ADFGVX, Xor32, Xor64, 0123456789abcdefx,
    Standard: ASCII
-D,--modeDecrypt
    Entschlüsselung durchführen
-E,--modeEncrypt
    Verschlüsselung durchführen (Standard, wenn weder
    Ver- noch Entschlüsselungsoption angegeben)
-ed,--editor
    Aktiven Editor als Eingabe verwenden
-f,--inputFile <FILE_PATH>
    datei als Eingabe verwenden
-k,--key <KEY>
    Schlüssel (es sind nur Buchstaben aus dem
    ausgewählten Alphabet erlaubt)
-k2,--key2 <KEY>
    Zweiter Transpositionsschlüssel, dessen Angabe
    eine zweite Transposition bewirkt (Doppelte
    Spaltentransposition)
-noFi,--noFilter
    Filterung von Nichtalphabet-Zeichen weglassen
-t,--inputText <TEXT>
    Text als Eingabe verwenden (als String zwischen
    "")
-t1ReadIn,--transposition1ReadInOrder <ORDER = 'cw'/'rw'>
    RICHTUNG = 'cw' (zeilenweise) / 'rw'
    (spaltenweise). Einleserichtung des Klartextes in
    die Transpositionstabelle (wenn nicht angeben,
    zeilenweise). (wird auf die 1. Transposition
    angewendet)
-t1ReadOut,--transposition1ReadOutOrder <ORDER = 'cw'/'rw'>
    siehe Beschreibung v. 't1ReadIn' (Wenn nicht
    angegeben, spaltenweise).
-t2ReadIn,--transposition2ReadInOrder <ORDER = 'cw'/'rw'>
    siehe Beschreibung v. 't1ReadIn' (Wenn nicht
    angegeben, zeilenweise).
-t2ReadOut,--transposition2ReadOutOrder <ORDER = 'cw'/'rw'>
    siehe Beschreibung v. 't1ReadIn' (Wenn nicht
    angegeben, spaltenweise).

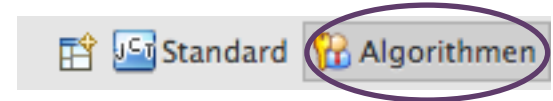
Beispiele zur Benutzung des Befehls erhalten Sie durch 'help -x transposition'.
Andere Namen für diesen Befehl sind 'transp'.
Weitere Informationen zu dem Verfahren finden Sie in der JCrypTool Onlinehilfe.
```

Die Perspektive „Algorithmen“

Die Implementierung in JCT

JCT-Perspektiven

- In JCT gibt es zwei Hauptansichten: die Standard-Perspektive und die Algorithmen-Perspektive.
- Die Algorithmen-Perspektive ist Funktions-orientiert und bietet erweiterte Einstellmöglichkeiten.



Die **Algorithmen-Perspektive** gliedert sich – neben dem Editor und der Hilfe – in die 3 Fensterbereiche:

1. Schlüsselspeicher (Keystore)

- Ermöglicht das Speichern von Schlüsseln und Schlüsselpaaren zur späteren Wiederverwendung.

2. Algorithmen

- Ein Explorer für die Algorithmen. Die Algorithmen stammen aus den Krypto-Bibliotheken FlexiProvider^[1] und BouncyCastle^[2]. Im Gegensatz zum Krypto-Explorer der Standard-Perspektive sind hier auch einzelne Varianten der Algorithmen direkt aufgelistet und auswählbar. Insgesamt ist der Algorithmen-Explorer umfangreicher als der Krypto-Explorer.

3. Operationen

- Hier wird der Algorithmus aufgelistet, der per Doppelklick im Algorithmen-Explorer ausgewählt wurde. Zusätzliche Einstellmöglichkeiten (z.B. die Quelle der Eingabe und Ziel der Ausgabe, Schlüssel und Parameter) sind unter jedem Algorithmus aufgeführt.

[1] <http://www.flexiprovider.de>

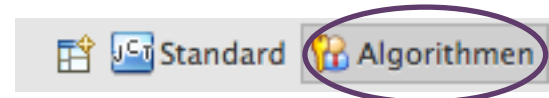
[2] <http://www.bouncycastle.org>

Die Perspektive „Algorithmen“

Die Implementierung in JCT

Algorithmen-Perspektive erklärt

- Wenn die Algorithmen-Perspektive zum ersten Mal geöffnet wird, erscheint eine Slide-Show, die die grundlegende Funktionsweise dieser Perspektive erklärt.





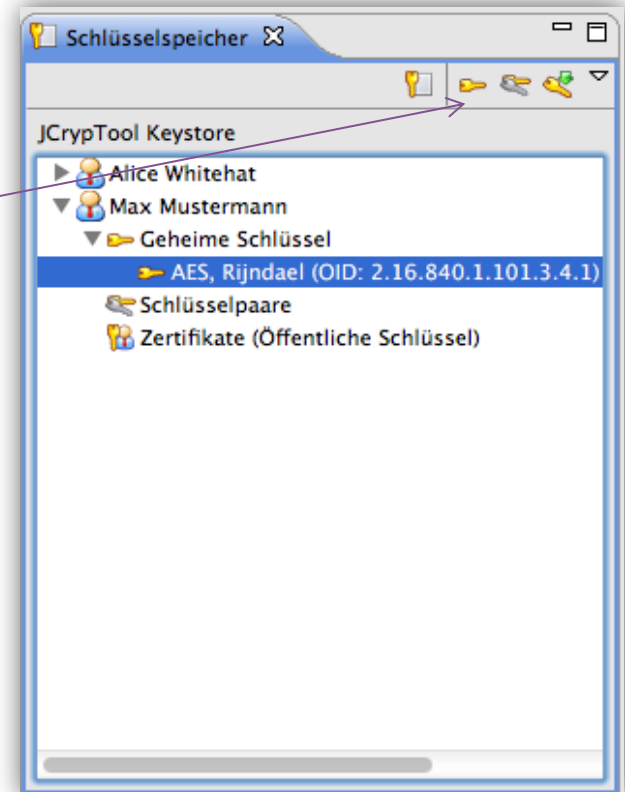
Die Perspektive „Algorithmen“

Ein Anwendungsbeispiel 1/3: Auswählen und Parametrisieren der AES-Operation

In diesem Beispiel wird ein Text aus dem geöffneten Editor mit AES verschlüsselt und das Chifftrat in eine Datei ausgegeben.

Neuen Schlüssel erzeugen und einem Kontakt zuordnen

- Als Erstes erzeugen wir einen Schlüssel, den wir zur Verschlüsselung benutzen.
- Mit  lässt sich ein neuer symmetrischer Schlüssel anlegen. Als symmetrisches Kryptosystem benötigt AES genau einen geheimen Schlüssel (statt eines Schlüsselpaars). Alternativ kann man für asymmetrische Kryptosysteme Schlüsselpaare mit  anlegen. Das ist Schritt **1** in der Grafik auf Folie **83**.
- Im Wizard „Neuer geheimer Schlüssel“ wählen wir unter Algorithmus den „AES, Rijndael (OID 2.16.840.1.101.3.4.1)“^[1], geben einen gewünschten Kontaktnamen an und setzen ein beliebiges Passwort.
- Der Schlüssel ist damit im JCT-Keystore (Schlüsselspeicher) unter dem gewählten Kontaktnamen (im Beispiel-Screenshot „Max Mustermann“) gespeichert.



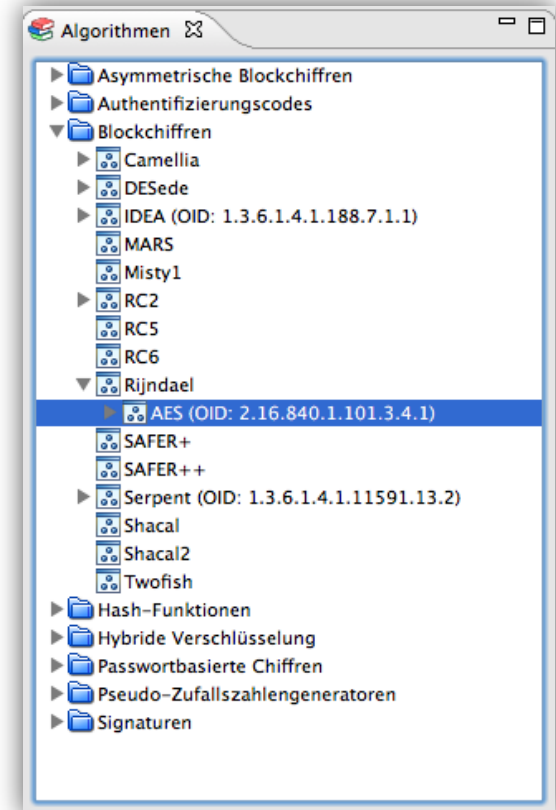
[1] OID: Object Identification, ein eindeutiger Bezeichner eines Algorithmus. Festgelegt von der ITU (http://en.wikipedia.org/wiki/Object_identifier).

Die Perspektive „Algorithmen“

Ein Anwendungsbeispiel 2/3: Auswählen und Parametrisieren der AES-Operation

Auswahl des Algorithmus für die aktuelle Operation

- Jetzt kann der entsprechende Algorithmus ausgewählt werden: Im Reiter „Algorithmen“ unter Blockchiffren befindet sich der AES-Rijndael-Algorithmus.
- Mit einem **Doppelklick** kann der Algorithmus ausgewählt werden (Schritt 2 auf Folie 83).
- Es erscheint ein Wizard, in dem man Padding und Modus^[1] der Blockchiffre einstellen kann. Außerdem können weitere spezifische Einstellungen des Algorithmus vorgenommen werden. (z.B. bei AES die Blocklänge in Bits).
- Analog zum Krypto-Explorer sind die Algorithmen nach Art des kryptografischen Verfahrens gruppiert.
- Anmerkung zur Bedienung:
Schritt 2 (Auswahl des Algorithmus) kann auch VOR Schritt 1 (Erzeugen eines Schlüssels für den Besitzer im Keystore) durchgeführt werden. Die Reihenfolge von Schritt 1 und 2 ist also egal. Und wenn der Kontakt (Besitzer) schon einen Schlüssel für diesen Alg. hat, kann man auch gleich mit Schritt 2 beginnen.



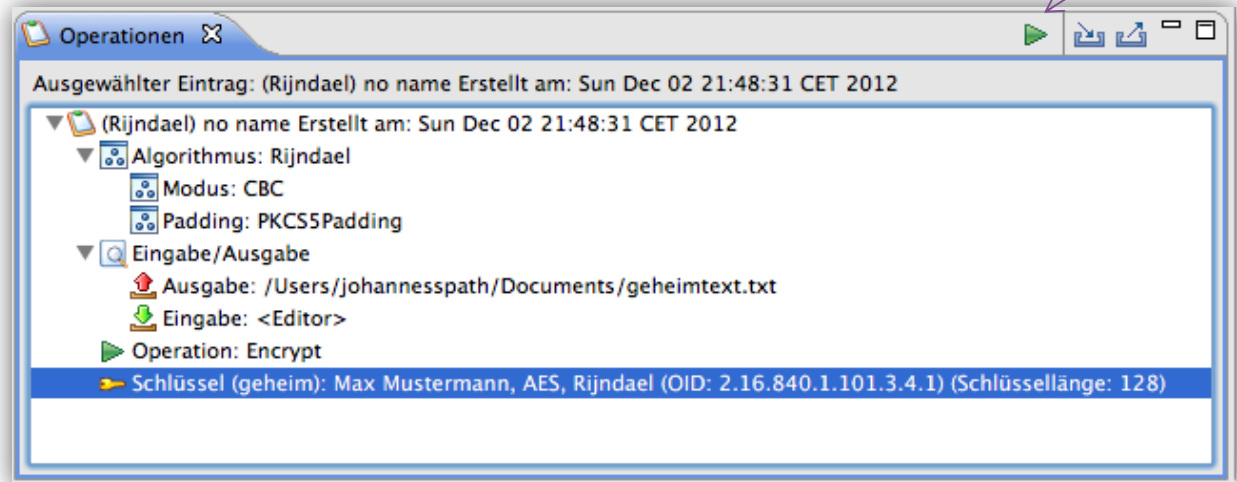
[1] Der Modus einer Blockchiffre steuert das Mapping des Klartextes auf die Blöcke, die verschlüsselt werden. Sollten im letzten Block Bits fehlen, regelt das Padding das Auffüllen der fehlenden Bits.

Die Perspektive „Algorithmen“

Ein Anwendungsbeispiel 3/3: Auswählen und Parametrisieren der AES-Operation

Einstellungen zu Ein- und Ausgabe für die aktuelle Operation

- Durch den Doppelklick (rechts im Algorithmen-Reiter) kam im Operationen-Reiter Rijndael hinzu.
- Der erstellte Schlüssel (aus dem JCT-Keystore links) kann nun per Drag'n'Drop auf das Schlüssel-Feld des Algorithmus gezogen werden (siehe Folie 83, Schritt 3).
- Unter Ein-/Ausgabe kann per Doppelklick die Quelle und das Ziel der Verschlüsselung gewählt werden. Z.B. kann man beim Input den Text aus einer Datei oder aus einem aktiven Editorfenster laden. Gleiches gilt für die Ausgabe (siehe Folie 83, Schritt 4).
- Per Rechtsklick auf „Operation“ kann man zwischen Verschlüsseln und Entschlüsseln wechseln.
- Nachdem nun alle Parameter gefüllt sind, startet man die Operation per Klick auf den grünen Pfeil rechts in der Titelleiste des Operationen-Fensters.
- Bei asymmetrischen Verfahren wird die Operation (ver-/entschlüsseln, signieren/verifizieren) durch die Art des verwendeten Schlüssels festgelegt. Ein öffentlicher Schlüssel verschlüsselt oder verifiziert, ein privater Schlüssel entschlüsselt oder signiert.



Die Perspektive „Algorithmen“

Überblick: 4 Schritte, um eine Operation auszuwählen und zu parametrisieren

The screenshot displays the JCryptTool application window with the following components and annotations:

- 1:** Points to the 'Schlüsselspeicher' (Keystore) pane on the left, where a key is selected.
- 2:** Points to the 'Algorithmen' (Algorithms) pane on the right, where the 'AES (OID: 2.16.840.1.101.3.4.1)' algorithm is double-clicked.
- 3:** Points to the 'Operationen' (Operations) pane at the bottom, where the selected algorithm is dragged.
- 4:** Points to the configuration details of the operation in the 'Operationen' pane, including 'Ausgabe' (Output) and 'Operation' (Encrypt).

The 'Algorithmen' pane shows a tree view of cryptographic algorithms:

- Asymmetrische Blockchiffren
- Authentifizierungscodes
- Blockchiffren
 - Camellia
 - DESede
 - IDEA (OID: 1.3.6.1.4.1.188.7.1.1)
 - MARS
 - Misty1
 - RC2
 - RC5
 - RC6
 - Rijndael
 - AES (OID: 2.16.840.1.101.3.4.1)**
 - SAFER+
 - SAFER++
 - Serpent (OID: 1.3.6.1.4.1.11591.13.2)
 - Shacal
 - Shacal2
 - Twofish
- Hash-Funktionen
 - DHA256
 - FORK256
 - MD4 (OID: 1.2.840.113549.2.4)
 - MD5 (OID: 1.2.840.113549.2.5)
 - RIPMD128 (OID: 1.3.36.3.2.2)
 - RIPMD160 (OID: 1.3.36.3.2.1)
 - RIPMD256 (OID: 1.3.36.3.2.3)
 - RIPMD320
 - SHA (OID: 1.3.14.3.2.26)
 - SHA224 (OID: 2.16.840.1.101.3.4.2.4)
 - SHA256 (OID: 2.16.840.1.101.3.4.2.1)
 - SHA384 (OID: 2.16.840.1.101.3.4.2.2)
 - SHA512 (OID: 2.16.840.1.101.3.4.2.3)
 - Tiger (OID: 1.3.6.1.4.1.11591.12.2)
 - VSH
- Hybride Verschlüsselung
- Passwortbasierte Chiffren
 - PBES2 (OID: 1.2.840.113549.1.5.13)
 - PbeWithMD5AndDES_CBC (OID: 1.2.840.113549.1.5.13)
 - PbeWithSHA1AndDES_CBC (OID: 1.2.840.113549.1.5.13)
 - PbeWithSHAAnd3_KeyTripleDES_CBC (OID: 1.2.840.113549.1.5.13)
 - PbeWithSHAAnd40BitRC2_CBC (OID: 1.2.840.113549.1.5.13)

Die Perspektive „Algorithmen“

Ergebnis nach Ausführung der Operation (Eingabe und Ausgabe stehen in 2 Reitern im Editor)

The screenshot shows the JCryptTool interface with the following components:

- Schlüsselspeicher:** A tree view of the JCryptTool Keystore containing keys for Alice Whitehat, Bob Whitehat, Erika Mustermann, and JCT-PKI Root Certificates.
- Hex Editor:** Displays the input file 'unbenannt001.txt' and the output file 'out001.bin'. The hex data is as follows:

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E
00000...	A5	A9	71	E3	B0	95	1D	9B	DF	72	33	13	05	C7	CA
00000...	58	77	CC	47	23	06	44	DC	08	BC	49	D2	B2	7D	18
00000...	3B	B5	11	44	D9	32	DC	BA	9E	1B	B7	59	F3	30	A7
00000...	06	C4	00	87	3B	FF	8F	8A	39	BD	C4	1D	66	A9	CA
00000...	CD	75	98	47	25	41	73	38	8D	23	D6	A2	DD	D9	1D
- Algorithmen:** A tree view of cryptographic algorithms. The selected algorithm is 'AES (OID: 2.16.840.1.101.3.4.1)'.
- Operationen:** A tree view of the selected operation. The configuration is:
 - Algorithmus: AES
 - Modus: CBC
 - Padding: PKCS5Padding
 - Eingabe/Ausgabe
 - Ausgabe: <Editor>
 - Eingabe: <Editor>
 - Operation: Encrypt
 - Schlüssel (geheim): Alice Whitehat, AES, Rijndael (OID: 2.16.840.1.101.3.4.1) (Schlüssellänge: 128)



Weitere Beispiele, was JCrypTool bietet

- Tri-partite Schlüsselvereinbarung
- Visualisierung der inneren Zustände bei DES
- Visualisierung von Elliptische Kurven-Berechnungen über reellen und diskreten Körpern
- ElGamal-Kryptosystem (zum Verschlüsseln und Signieren)
- Visualisierung der Simple Power Analysis-Angriffe gegen RSA (SPA)
- Schnelle Lösung des Zahlenhai-Spiels mit heuristischen Methoden; Lösung von Sudoku-Varianten
- Mathematische Spiele: Zahlenhai, Teilerspiel, Zero-Knowledge-Sudoku (Zudo-Ku)
- Entropie-Untersuchungen
- Dynamische Visualisierung von Huffman-Bäumen
- Signatur-Demo, Signatur- und Zertifikats-Verifikation (zeigt Auswirkung versch. Gültigkeitsmodelle)
- Visualisierung des SSL- / TLS-Handshake (Protokoll)
- Implementierung und Visualisierung von ARC4 und Spritz
- Visualisierung von Post-Quantum Signierverfahren [**SPHINCS+**, MerkleTree XMSS-MT, WOTS , McEliece (error-correcting code), multivariate Kryptografie (rainbow signature scheme)]
- Schnelle Kryptoanalyse der Fleißner-Schablone (Grille)
- ...

Übersicht über alle Funktionen in JCrypTool

Sichtbar in JCT selbst. Alternativen: CrypTool-Portal (siehe unten) / mit dem JCT-Admintool

CrypTool Portal
Cryptography for everybody

Schon gewusst?
Ursprünglich für IT-Sicherheits-Schulungen im Unternehmen entwickelt, hat sich CrypTool inzwischen zu einem bedeutenden Open-Source-Projekt im Bereich Kryptologie entwickelt, an dem über 50 Entwickler weltweit ehrenamtlich mitarbeiten.

Über CrypTool | Dokumentation | Lehre | Mitwirkende | Links / Bücher

Kryptologische Funktionen in verschiedenen CrypTool-Versionen

Auswahl

Kryptographische Kategorie: Kein Filter angewendet

Zusätzlicher Suchfilter: Suchen...

CrypTool 1 (CT1) CrypTool 2 (CT2) JCrypTool (JCT) CrypTool Online (CTO)

111 von insgesamt 310 Funktionsgruppen entsprechen dem Auswahlkriterium.

Funktionen	JCT	Pfad in JCT
ADFGX / ADFGVX	D	[D] \ Algorithmen \ Klassisch \ ADFGVX
AES	A/D	[A] \ Blockchiffren \ Rijndael \ AES \ AES128_CBC [A] \ Blockchiffren \ Rijndael \ AES \ AES128_CFB [A] \ Blockchiffren \ Rijndael \ AES \ AES128_ECB [A] \ Blockchiffren \ Rijndael \ AES \ AES128_OFB [A] \ Blockchiffren \ Rijndael \ AES \ AES192_CBC [A] \ Blockchiffren \ Rijndael \ AES \ AES192_CFB [A] \ Blockchiffren \ Rijndael \ AES \ AES192_ECB [A] \ Blockchiffren \ Rijndael \ AES \ AES192_OFB [A] \ Blockchiffren \ Rijndael \ AES \ AES256_CBC [A] \ Blockchiffren \ Rijndael \ AES \ AES256_CFB [A] \ Blockchiffren \ Rijndael \ AES \ AES256_ECB [A] \ Blockchiffren \ Rijndael \ AES \ AES256_OFB [D] \ Algorithmen \ Symmetrisch \ AES
AES MAC	A	[A] \ Authentifizierungscodes \ CBCMac \ CBCmacAES128 [A] \ Authentifizierungscodes \ CBCMac \ CBCmacAES192 [A] \ Authentifizierungscodes \ CBCMac \ CBCmacAES256 [A] \ Authentifizierungscodes \ CMac \ CmacAES128 [A] \ Authentifizierungscodes \ CMac \ CmacAES192

<https://www.cryptool.org/de/documentation/functionvolum>

Legende:

[A] in Algorithmen-Perspektive

[D] in Standard-Perspektive
(D für default)



Einführung in das JCrypTool-Programm

2

Anwendungsbeispiele

22

Möglichkeiten zur Mitwirkung

87

Möglichkeiten zur Mitwirkung

Überblick



JCrypTool – Bitte um Mitwirkung

Seite 89

Mitwirkung bei JCrypTool

Seite 90

Kontaktadressen

Seite 92



Wir freuen uns über jede weitere Mitarbeit

- Feedback, Kritik, Anregungen und Ideen
- Einbau weiterer Algorithmen, Protokolle, Analysen
- Hilfe, um Konsistenz und Vollständigkeit zu sichern
- Mithilfe bei der Entwicklung (Programmierung, Layout, Übersetzung, Test, Webseiten-Erweiterung)
 - im bisherigen C/C++-Projekt CryptTool 1 und
 - in den neuen Projekten (bevorzugt):
 - C#-Projekt: „CryptTool 2“ = CT2 (<https://www.cryptool.org/de/ct2/volunteer>)
 - Java-Projekt: „JCrypTool“ = JCT (<https://www.cryptool.org/de/jct/volunteer>)
 - Browser-Projekt: „CryptTool-Online“ = CTO (<http://www.cryptool-online.org>)
- Insbesondere Lehrstühle, die JCrypTool zur Ausbildung verwenden, sind herzlich eingeladen, zur Weiterentwicklung beizutragen.
- Signifikante Beiträge können namentlich erwähnt werden (in der Online-Hilfe, in Info-Dialogen und auf der Webseite).



Beispiel-Ideen für weitere Visualisierungen

- Visualisierung der Interoperabilität von S/MIME- und OpenPGP-Formaten
- Demo zur visuellen Kryptografie
- Protokoll-Validierer
- Kryptoanalyse weiterer Algorithmen
- Visualisierung verschiedener Verfahren der Post-Quantum-Kryptografie
- Visualisierung moderner Entwicklungen (bspw. indistinguishability obfuscation)

Weitere wünschenswerte Implementierungen

- Einheitliche Manipulation (Erstellung, Austausch, Tiefe) aller Häufigkeitstabellen und aller Permutationen
- Schlüsselspeicher
- Support für JavaFX

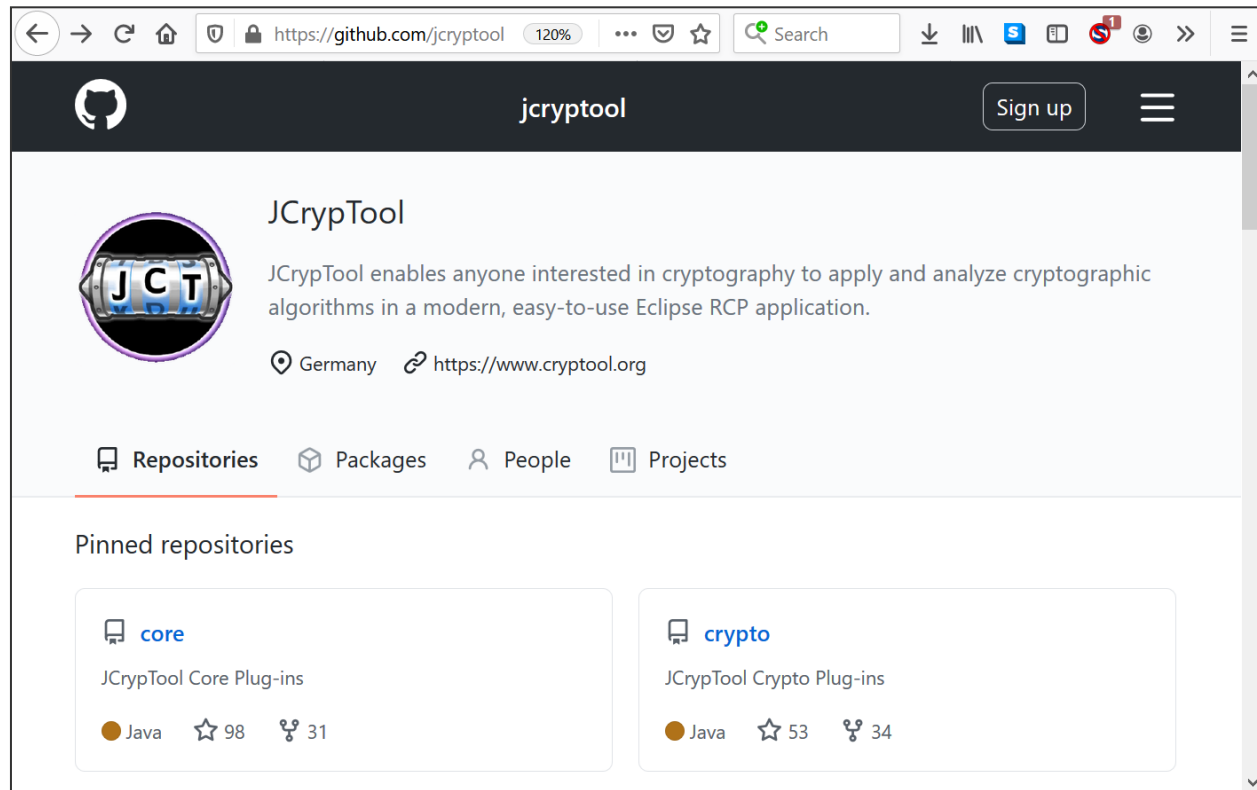
Beispiele offener Aufgaben finden sich auch auf den entsprechenden Entwickler-Seiten:

- JCrypTool: <https://github.com/jcryptool/core/wiki/project-Ideas>

Mitwirkung bei JCrypTool

Weitere Informationen für Entwickler

- Wiki: <https://github.com/jcryptool/core/wiki>
- Style-Guide: <https://github.com/jcryptool/doc/raw/master/Guidelines/JCrypTool-GUI-Guidelines.pdf>
- Tutorial: <https://github.com/jcryptool/core/wiki/Getting-started-as-a-JCrypTool-Developer>
- Alles rund um die Plugin-Entwicklung steht in dem JCT-Wiki. Das Wiki im Internet enthält Verweise und Infos für die Core-Entwickler und für Krypto-Plugin-Entwickler.
- Plugin-Entwickler benötigen keine Projekte aus dem JCT-Repository, sondern richten JCT als Target-Plattform ein und entwickeln damit.



The screenshot shows the GitHub profile page for the organization 'jcryptool'. The page features the GitHub logo, the organization name 'jcryptool', and a 'Sign up' button. Below this is the organization's profile picture, a circular logo with 'JCT' and 'CRYPTO' text. The description reads: 'JCrypTool enables anyone interested in cryptography to apply and analyze cryptographic algorithms in a modern, easy-to-use Eclipse RCP application.' The location is listed as 'Germany' and the website as 'https://www.cryptool.org'. Navigation tabs for 'Repositories', 'Packages', 'People', and 'Projects' are visible. Under 'Pinned repositories', two repositories are shown: 'core' (JCrypTool Core Plug-ins) with 98 stars and 31 forks, and 'crypto' (JCrypTool Crypto Plug-ins) with 53 stars and 34 forks. Both are marked as Java projects.



Prof. Bernhard Esslinger	Simon Leischnig	Thorben Groos
CT Gesamt-Projektleiter Universität Siegen	JCT-Projektleiter	JCT Co-Projektleiter
<u>bernhard.esslinger@uni-siegen.de</u> <u>bernhard.esslinger@gmail.com</u>	<u>simonjena@gmail.com</u>	<u>thorben.groos@web.de</u>

Dominik Schadow: früherer Projektleiter, info@xml-sicherheit.de

www.cryptool.org