

# Cryptology with JCrypTool (JCT)

A Practical Introduction  
to Cryptography and  
Cryptanalysis

*Prof Bernhard Esslinger  
and the CrypTool team*

Nov 24<sup>th</sup>, 2020





**Introduction to the e-learning software JCrypTool**

2

Applications within JCT – a selection

22

How to participate

87

# Introduction to the software JCrypTool (JCT)



## Overview

JCrypTool – A cryptographic e-learning platform	Page 4
What is cryptology?	Page 5
The Default Perspective of JCT	Page 6
Typical usage of JCT in the Default Perspective	Page 7
The Algorithm Perspective of JCT	Page 9
The Crypto Explorer	Page 10
Algorithms in the Crypto Explorer view	Page 11
The Analysis tools	Page 13
Visuals & Games	Page 14
General operation instructions	Page 15
User settings	Page 20
Command line parameters	Page 21

# JCryptTool – A cryptographic e-learning platform

## The project

### Overview

- JCrypTool – abbreviated as JCT – is a free e-learning software for classical and modern cryptology.
- JCT is platform independent, i.e. it is executable on **Windows, MacOS and Linux**. It has a modern pure-plugin architecture.
- JCT is developed within the open-source project CrypTool ([www.cryptool.org](http://www.cryptool.org)).
- The CrypTool project aims to explain and visualize cryptography and cryptanalysis in an easy and understandable way while still being correct from a scientific point of view.
- The target audience of JCT are mainly:
  - Pupils and students
  - Teachers and lecturers/professors
  - Employees in awareness campaigns
  - People interested in cryptology.
- As JCT is open-source software, everyone is capable of implementing his own plugins. Already developed components can be easily reused.
- JCT was built by more than 100 contributors from different countries.



JCT Splash Screen

# What is cryptology?

## What is JCrypTool all about?



### The meaning of cryptology

- From Greek: „kryptós“ („hidden, secret“) and „lógos“ („writing“, however in this context „lógos“ means „study“).
- Cryptology is about techniques and protocols making information available only for authorized persons. Cryptology consists of two parts (fields).

### The field cryptography

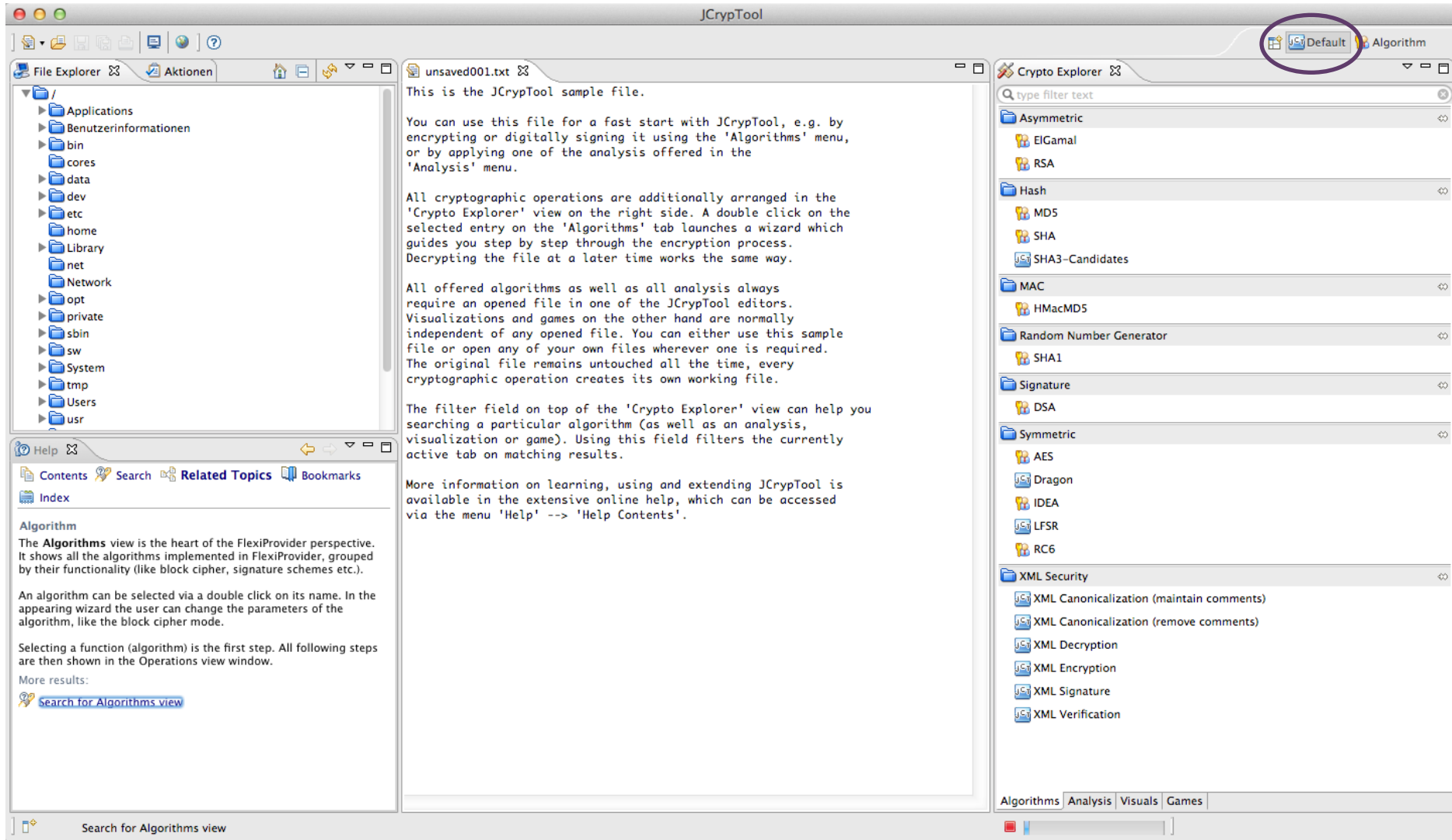
- Science of encryption systems guaranteeing secure and confidential storage and exchange of information (e.g. between computers).
- Nowadays, other important tasks are a secure exchange of the encryption keys and integrity checking, e.g. for online banking, for electronic elections, or for electronic money.
- Most of the methods used in this field are based on (unsolved/difficult) mathematical problems.

### The field cryptanalysis

- Cryptanalysis is the counter part to cryptography and studies theories and techniques for testing and breaking cryptographic methods.
- It tries e.g. to derive information about the original plaintext or the used encryption key by investigating a ciphertext (the result of an encryption process).
- Therefore, maths and computer science are used (e.g. statistical tests, entropy, analysis of frequency and structure, complexity considerations, brute-force algorithms and much more).

# The Default Perspective of JCT

... focuses on documents (document-oriented)



# Typical usage of JCT in the Default Perspective

... selecting a method from the main menu "Algorithms"

The screenshot displays the JCryptTool application interface. The main menu is open, showing the 'Algorithms' category selected. The 'Algorithms' menu is expanded, listing various cryptographic methods: Asymmetric, Classic, Hash, MAC, Random Number Generator, Signature, Symmetric, and Keystore. The 'Symmetric' sub-menu is further expanded, showing AES, ARC4/Spritz, Dragon, IDEA, LFSR, and RC6. The 'Crypto Explorer' view on the right side of the interface is active, displaying a list of available cryptographic operations categorized by type (Asymmetric, Classic, Hash, MAC, Random Number Generator, Signature, Symmetric). The 'Symmetric' category is expanded, showing a list of algorithms: ADFGVX, Autokey-Vigenère, Bifid, Caesar, Double Box, Playfair, Substitution, Transposition, Vigenère, and XOR. The 'Algorithms' tab is selected in the bottom navigation bar.

File Edit Algorithms Analysis Visuals Games Window Help

Asymmetric  
Classic  
Hash  
MAC  
Random Number Generator  
Signature  
Symmetric  
Keystore

unsaved001.txt  
This is the JCryptTool sample file.  
You can use this file for a fast start with JCryptTool, e.g. by encrypting or digitally signing it using the 'Algorithms' menu, or by applying one of the analysis offered in the 'Analysis' menu.  
operations are additionally arranged in the view on the right side. A double click on the the 'Algorithms' tab launches a wizard which step through the encryption process. e at a later time works the same way.  
thms as well as all analysis always require an opened file in one of the JCryptTool editors. Visualizations and games on the other hand are normally independent of any opened file. You can either use this sample file or open any of your own files wherever one is required. The original file remains untouched all the time, every cryptographic operation creates its own working file.  
The filter field on top of the 'Crypto Explorer' view can help you searching a particular algorithm (as well as an analysis, visualization or game). Using this field filters the currently active tab on matching results.  
More information on learning, using and extending JCryptTool is available in the extensive online help, which can be accessed via the menu 'Help' --> 'Help Contents'.

Help Contents Search Related Topics  
Bookmarks Index

**Crypto Explorer View**  
The **Crypto Explorer** view lists all available cryptographic operations in JCryptTool ordered by category. A file must be opened in one of the JCryptTool editors in order to execute an **Algorithm**. This is the case for views in the category **Analysis** too. **Visualizations** and **Games** generally do not require an opened file.  
Use the menu to switch between the tree and palette presentation. The filter field enables you to filter for the name of an

Crypto Explorer  
type filter text

Asymmetric  
Classic  
ADFGVX  
Autokey-Vigenère  
Bifid  
Caesar  
Double Box  
Playfair  
Substitution  
Transposition  
Vigenère  
XOR  
Hash  
MAC  
Random Number Generator  
Signature  
Symmetric  
AES  
ARC4/Spritz  
Dragon  
IDEA  
LFSR  
RC6

Algorithms Analysis Visuals Games



# Typical usage of JCT in the Default Perspective

... selecting a method from the main menu "Visuals"

The screenshot displays the JCryptTool 1.0 interface. The main menu bar includes File, Edit, Algorithms, Analysis, Visuals, Games, Window, and Help. The 'Visuals' menu is open, listing various cryptographic methods. The 'Crypto Explorer' view is active, showing a list of algorithms under the 'Visuals' category. The 'Grille' method is selected. The 'File Explorer' window shows the file system structure, with the 'snuc' folder selected. The 'Crypto Explorer View' section provides a detailed description of the view and its usage.

File Edit Algorithms Analysis Visuals Games Window Help

File Explorer

- bin
- boot
- cdrom
- dev
- etc
- home
  - snuc
    - \_from\_sp3
    - data
    - Desktop
    - Documents
    - Downloads

Help

Contents Search Related

Bookmarks Index

**Crypto Explorer View**

The **Crypto Explorer** view lists all cryptographic operations in JCryptTool, ordered by category. A file must be loaded in one of the JCryptTool editors in order to execute an **Algorithm**. This is the view in the category **Analysis** to **Visualizations** and **Games** generally require an opened file.

Use the menu to switch between the view and palette presentation. The filter enables you to filter for the name of the algorithm.

Visuals

- Android Unlock Pattern (AUP)
- Ant Colony Optimization (ACO)
- ARC4 / Spritz
- Certificate Verification
- Chinese Remainder Theorem (CRT)
- Diffie-Hellman Key Exchange (EC)
- ElGamal Cryptosystem
- Elliptic Curve Calculations
- Extended Euclidean / Reciprocal Subtraction
- Extended RSA Cryptosystem
- Grille
- Hash Sensitivity
- Homomorphic Encryption (HE)
- Huffman Coding
- Inner States of the Data Encryption Standard (DES)
- Kleptography**
- McEliece Cryptosystem
- Merkle Signatures (XMSS<sup>MT</sup>)
- Merkle-Hellman Knapsack Cryptosystem
- Multipartite Key Exchange (BD II)
- Multivariate cryptography
- RSA Cryptosystem
- Shamir's Secret Sharing
- Shanks Babystep-Giantstep
- Signature Demonstration
- Signature Verification
- Simple Power Analysis / Square and Multiply
- SPHINCS Signature
- SPHINCS+ Signature

Crypto Explorer

type filter text

Visuals

- Diffie-Hellman Key Exchange (EC)
- ElGamal Cryptosystem
- Elliptic Curve Calculations
- Extended Euclidean / Reciprocal Subtraction
- Extended RSA Cryptosystem
- Grille**
- Hash Sensitivity
- Homomorphic Encryption (HE)
- Huffman Coding
- Inner States of the Data Encryption Standard (DES)
- Kleptography
- McEliece Cryptosystem
- Merkle Signatures (XMSS<sup>MT</sup>)
- Merkle-Hellman Knapsack Cryptosystem
- Multipartite Key Exchange (BD II)
- Multivariate cryptography
- RSA Cryptosystem
- SPHINCS Signature
- SPHINCS+ Signature
- SSL/TLS Handshake
- Shamir's Secret Sharing

Algorithms Analysis Visuals Games

Insert 7:1



# The Algorithm Perspective of JCT

... focuses on functions (function-oriented)

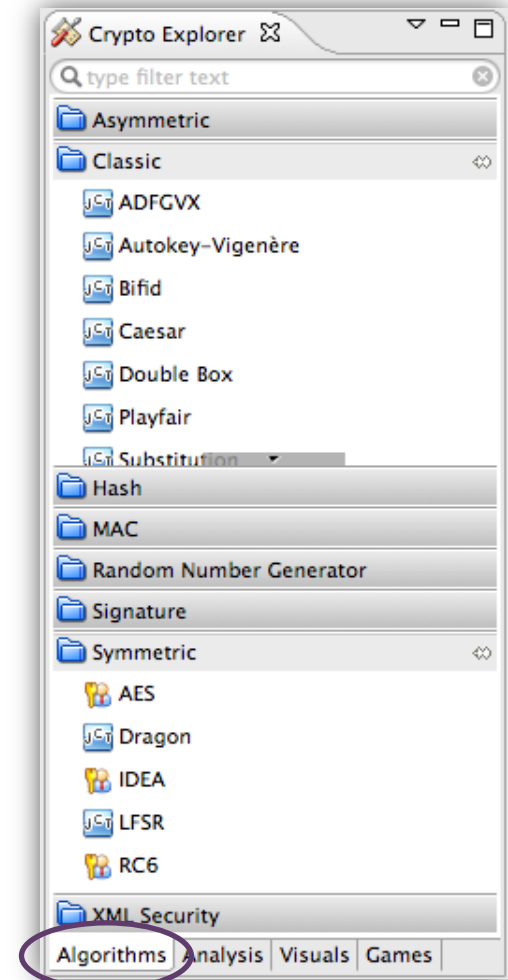


# The Crypto Explorer

## In the Default Perspective of JCT

### Functionality

- On the right side in the Default Perspective of JCT you can find the tab “Crypto Explorer”. In this explorer the functions of JCT are shown.
- All functions shown in the explorer can be found in the menus as well.
- In the same manner as the menus, the explorer is clustered into
  - Algorithms
  - Analysis
  - Visuals
  - Games
- Usually algorithms and analyses are applied to the active document in the editor; the calculated output is shown in a new editor window.
- Visuals and games are independent from the document shown in the editor.



# Algorithms in the Crypto Explorer view

## Clustering 1/2

### Classic methods

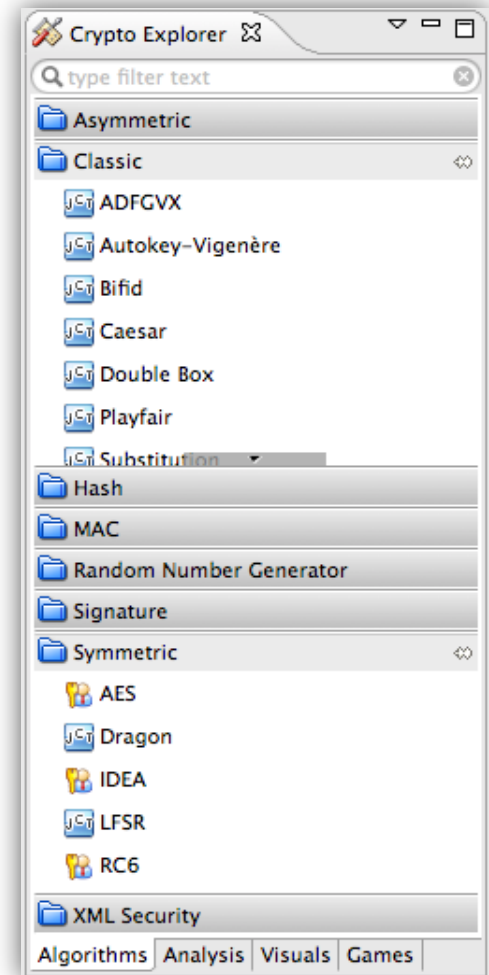
- This category gathers methods, which were used to encrypt messages roughly until World War I. Many of them are breakable by analyzing frequencies. Most of these methods are nowadays insecure.

### Symmetric methods

- Modern methods, where sender and receiver need to have the same key.
- A main problem of symmetric methods is:  
The key must be shared safely between the relevant participants of the communication.

### Asymmetric methods

- Modern methods, where each participant has a pair of keys – a private and a public one.
- The sender **encrypts** his message with the public key of the receiver, while only the receiver can **decrypt** the message with his own private key.



# Algorithms in the Crypto Explorer view

## Clustering 2/2

### Hash & MAC

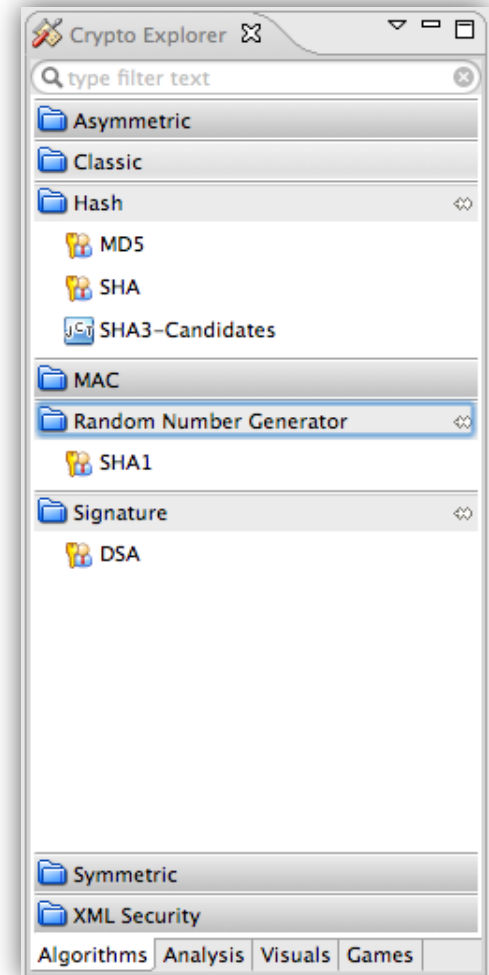
- Hash functions map data of arbitrary length to a hash value. This hash value is associated with the data in a preferably unique way and has a fixed bounded length which is normally much smaller than the length of the referred data (comparable to a fingerprint).
- Hash values are used to check for changes in documents (integrity). A widely used second application is to check passwords. Therefore the hash value (instead of the plain password) is stored in the database.

### Signatures

- Signature algorithms are used to sign messages and documents.
- With a signature one can check the integrity of documents – the property that a document is unchanged.

### Random number generators

- In cryptography random numbers play a major role. Therefore functions for generating (pseudo-random) sequences of numbers are implemented in JCT as well.

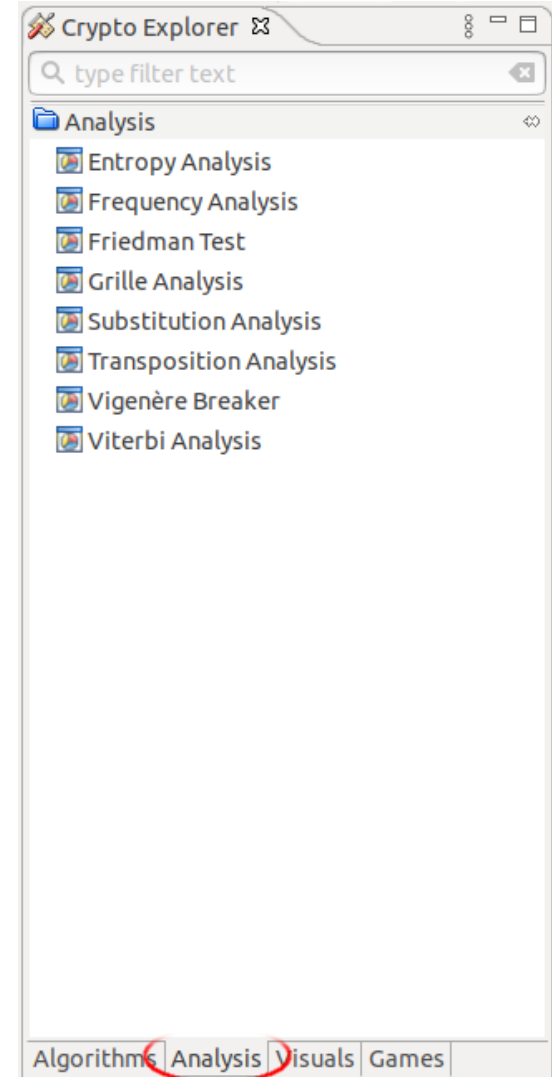


# The Analysis tools

## In the Crypto Explorer

### Analysis algorithms

- In this tab of the Crypto Explorer analysis tools are listed. These tools allow the user to analyze a given cipher text, to find possible regularities (patterns) to derive the plain text or the password (key) of the encryption.
- The algorithms are also applied to the document which is currently opened in the editor.
- Different kinds of analyses are possible. E.g. a *transposition analysis*: a ciphertext which was transposed column-wise or row-wise might be rearranged to its original plaintext.
- With an analysis of frequencies the frequencies of characters or pairs of characters can be determined. As characters appear with variant frequency in each natural language, patterns or recurrences can be found and first ideas of the plain text can be deviated.

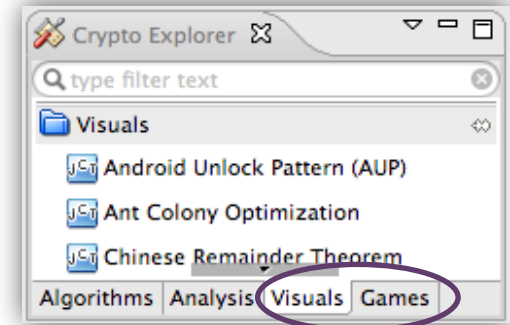


# Visuals & Games

## In the Crypto Explorer

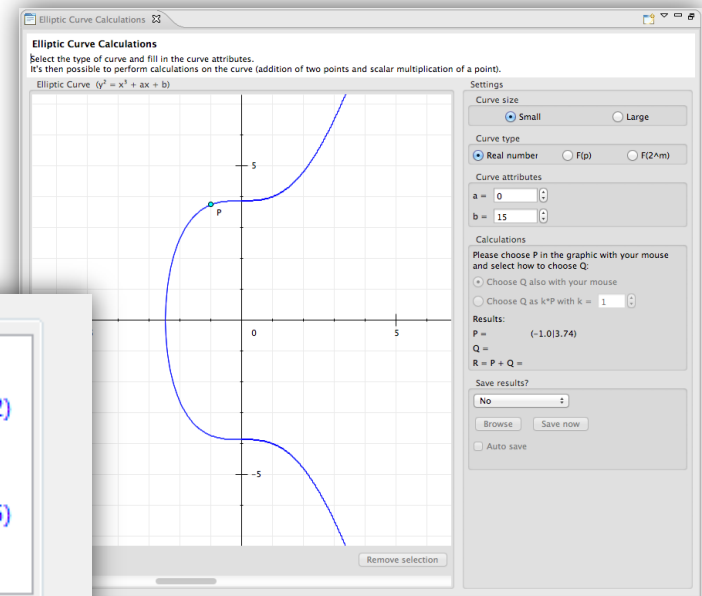
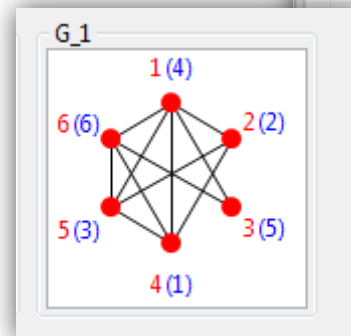
### Visuals

- Visuals can be found in the tab “Visuals” in the Crypto Explorer or in the menu “Visuals”.
- More than 20 visuals of cryptographic problems, circumstances and algorithms shall help the user to understand cryptography in a descriptive and playful way.
- To understand cryptology, a basic knowledge of mathematics and informatics is necessary. Therefore the visuals explain the appropriate knowledge as well.



### Games

- In the section “Games”, games can be played and strategies developed to solve apparently easy problems.
- Some games (e.g. the number shark) provide extensive theories and possible strategies.



# General operation instructions

... 1/5 (Quick Access: Search over all parts in JCT)

## Tips and tricks

- With the key combination "**Ctrl+3**", the quick access window can be opened.
- Algorithms, visualizations and other content from JCrypTool can be found here and directly opened.
- This is the quickest way to search something in the Default Perspective or in the online help, if you don't know where to look for in the menus or in the Crypto Explorer.
- Clicking at a shown entry makes JCT to jump straight there.
- Remark: Elements from the Algorithm Perspective are currently not shown in the quick access window.

The screenshot shows a search window with the text 'ana' in the search bar. The results are organized into three sections: Views, Commands, and Help. The Views section lists various analysis tools. The Commands section lists actions like 'Show In' and 'Show View' for each tool. The Help section shows a search result for 'ana' in the help documentation. At the bottom, a note states 'Results per category are limited. Press 'Ctrl+3' to see all'.

Category	Item
Views	Entropy <b>Analysis</b> (Analysis)
Views	Frequency <b>Analysis</b> (Analysis)
Views	Friedman Test ( <b>Analysis</b> )
Views	Grille <b>Analysis</b> (Analysis)
Views	Simple Power <b>Analysis</b> / Square and Multiply (Visuals)
Views	Substitution <b>Analysis</b> (Analysis)
Views	Transposition <b>Analysis</b> (Analysis)
Views	Vigenère Breaker ( <b>Analysis</b> )
Views	Viterbi <b>Analysis</b> (Analysis)
Commands	Entropy <b>Analysis</b>
Commands	Frequency <b>Analysis</b>
Commands	Grille <b>Analysis</b>
Commands	Show In (Entropy <b>Analysis</b> )
Commands	Show In (Frequency <b>Analysis</b> )
Commands	Show In (Grille <b>Analysis</b> )
Commands	Show In (Simple Power <b>Analysis</b> / Square and Multiply)
Commands	Show In (Substitution <b>Analysis</b> )
Commands	Show In (Transposition <b>Analysis</b> )
Commands	Show In (Viterbi <b>Analysis</b> )
Commands	Show View (Entropy <b>Analysis</b> ) - Shows a particular view
Commands	Show View (Frequency <b>Analysis</b> ) - Shows a particular view
Commands	Show View (Grille <b>Analysis</b> ) - Shows a particular view
Commands	Show View (Simple Power <b>Analysis</b> / Square and Multiply) - Shows a particular view
Commands	Show View (Substitution <b>Analysis</b> ) - Shows a particular view
Commands	Show View (Transposition <b>Analysis</b> ) - Shows a particular view
Help	Search ' <b>ana</b> ' in Help

Results per category are limited. Press 'Ctrl+3' to see all


Quick Access window

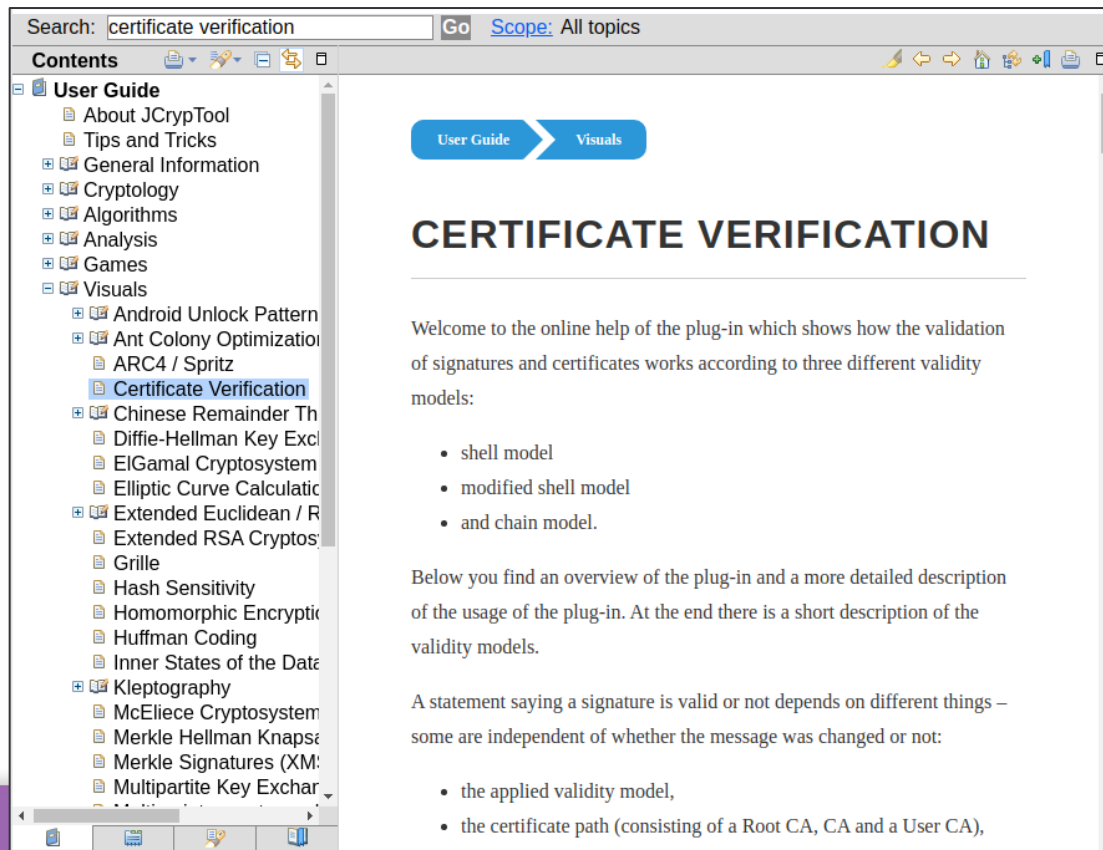
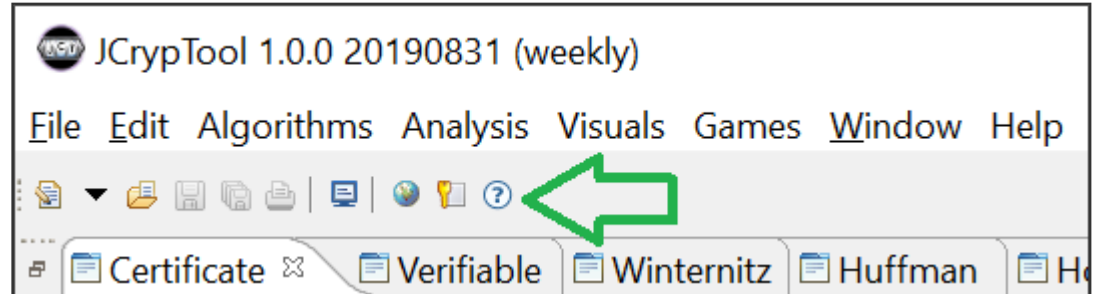


# General operation instructions

... 2/5 (open the exhaustive online help within a new browser tab)

## Tips and tricks

- With the question mark symbol  in the toolbar, an exhaustive **online help** can be opened in a new **browser** tab.



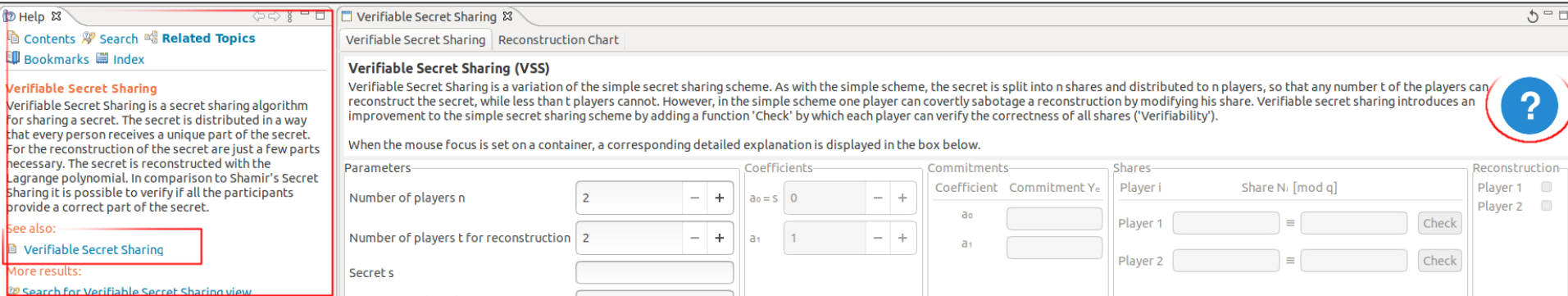
Online help  
within browser

# General operation instructions

... 3/5 (open the docked context help)

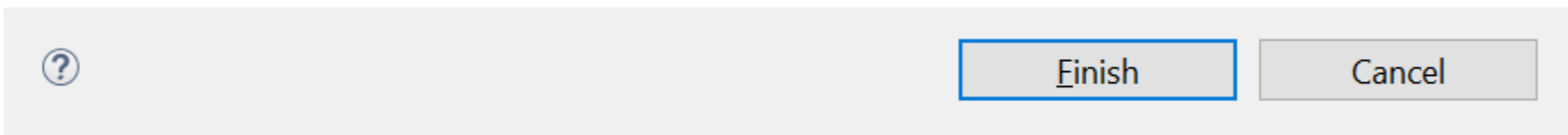
## Tips and tricks

- The function key **F1** can be used (under Linux and Windows) to open the context-sensitive help at any time.  
The context help contains detailed information and references for the current view.
- Alternatively, you can use the big **blue question mark** symbol to show (or hide) the context-help window. This big question mark can be found in all description headers of visual plugins.



Context help for the visuals plugin „Verifiable Secret Sharing“




- In dialog windows: Clicking F1 or pressing the small question mark on the left shows a context-help window docked at the right side of the dialog.

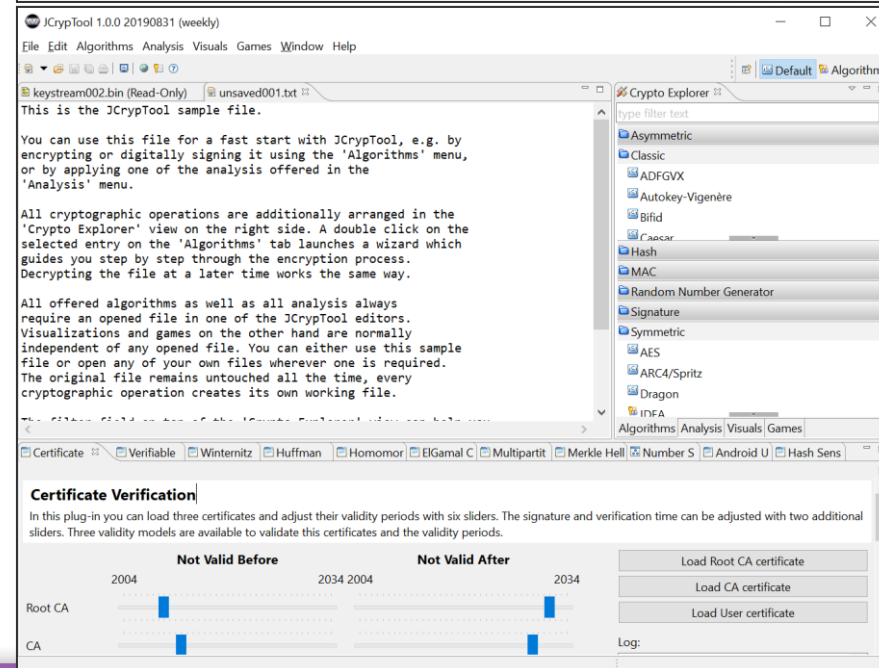
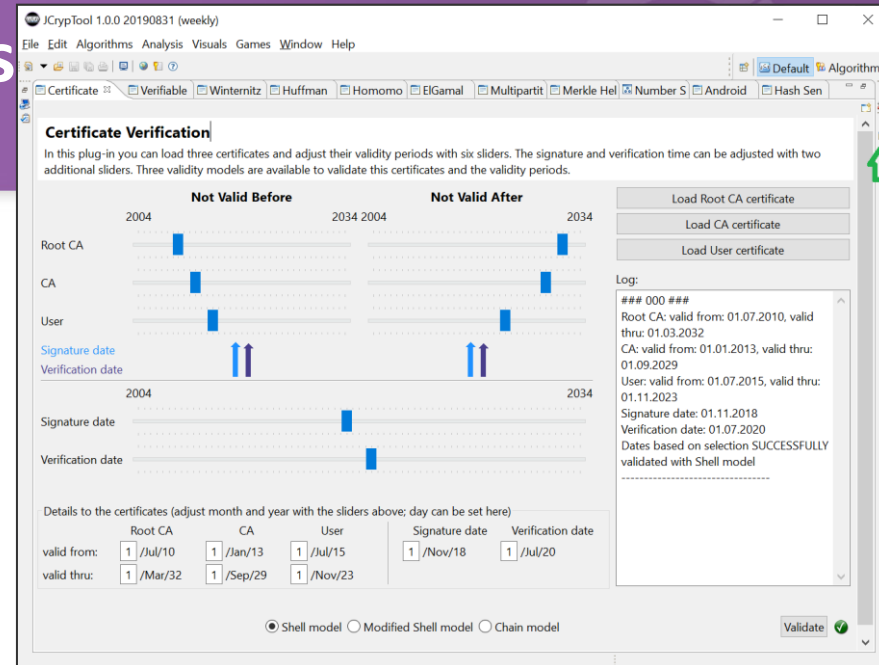


# General operation instructions

... 4/5 (minimize, maximize and resize)

## Tips and tricks

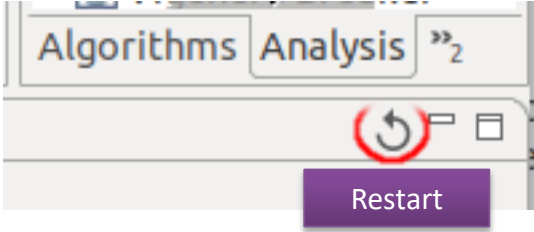
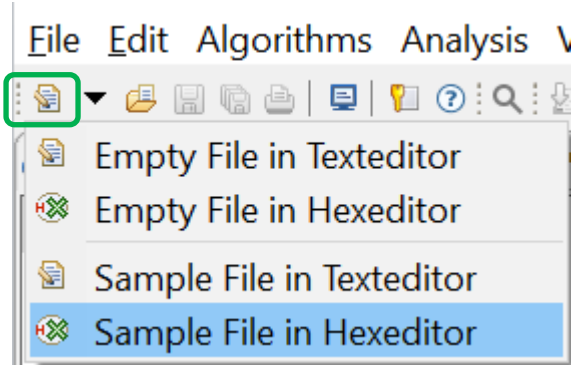
- The size of each area in JCT can be adjusted via the buttons on the upper right corner of the area.
  - Maximize current area 
  - Minimize area 
- Once minimized, an area is represented as a small bar at the left or right border. The tabs contained are displayed as small icons inside the bar.
  - An area can be reset to its old size by clicking on 
  - The other icons represent the tabs inside an area. Clicking on one of these, the appropriate tab will be shown as an overlapping window which will be automatically hidden again.
- Reset of a **view** or of the **perspective**:
  - Menu “Window” \ “Reset Perspective”
  - Double-click on the plugin’s tab changes the view between full and reduced place for the plugin (typical behavior in Eclipse)



# General operation instructions

... 5/5 (restart within an opened visual plugin; open an editor)

## Tips and tricks

- Each visual can be reset to its initial settings by clicking on the button “**Restart**”. The button is located in the upper toolbar of the plugin window.
- Quick creation of an editor window
  - On the far left of the toolbar is the editor icon. If you click on it, a text editor with the sample file is opened.
  - Directly to the right of it is the arrow. With the arrow you can choose which type of editor (and whether empty or filled) should be opened in the middle of the Default Perspective or the middle of the Algorithm Perspective.

# User settings

...the global preferences of JCT

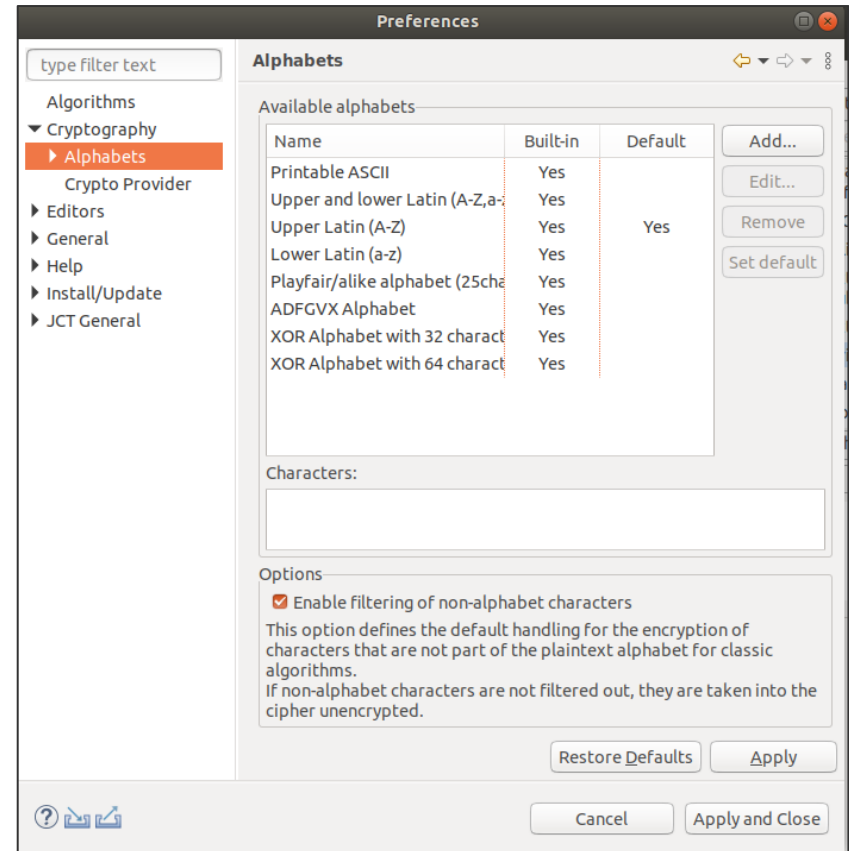
## More settings

- The global settings of JCT can be found in the preferences: See the menu paths:  
on Windows and Linux: „Window \ Preferences”; and under MacOS: „JCrypTool \ Preferences”.

The most important JCT specific settings are:

## Concerning cryptography

- **Alphabets:** Manage alphabets which are used for many of the classic encryption methods.
- **Keystore:** Here you can manage the files in which the keys of the JCT keystore are saved.  
A newly generated keystore can then be used in the perspective “Algorithm”.



# Command line parameters

## ...setting the language and the data directory



### Default

- If JCrypTool is started without parameters its language will be inferred from the operating system, if the operating system is in English or German. Otherwise, JCrypTool defaults to English.
- Settings of the last JCrypTool session as well as JCT-specific files are stored in a directory named 'workstation'. This directory is created when JCrypTool is started for the first time, below the JCrypTool directory.

### Parameters to control the JCT application from the command line when starting

- **Language**  
-nl [de, en]
- **Data directory** (with -data you can change the directory 'workstation')  
-data [directory]

Sample: JCrypTool -nl de -clean -data USERDIRECTORY/jct-de  
JCrypTool -nl en -clean -data USERDIRECTORY/jct-en

With -nl you control, whether the German or English JCT appears.

With -data you control, where JCT stores its data. It's up to you to specify a meaningful value for USERDIRECTORY (under Windows for instance %LOCALAPPDATA%).



Introduction to the e-learning software JCrypTool

2

**Applications within JCT – a selection**

22

How to participate

87



# Applications within JCT – Overview



The ant colony optimization (ACO)	Page 25
Viterbi analysis	Page 30
Verifiable Secret Sharing	Page 35
Signature demonstration	Page 40
Extended RSA cryptosystem	Page 45
SETUP attack on the RSA key generation (Kleptography)	Page 50
Zero-knowledge protocol: Fiat Shamir	Page 55
Android Unlock Pattern (AUP)	Page 60
Cascades in the Actions window	Page 64
Variable alphabets for classic algorithms	Page 70
JCryptTool console for classic methods	Page 74
The perspective “Algorithm”	Page 79

# The ant colony optimization (ACO)

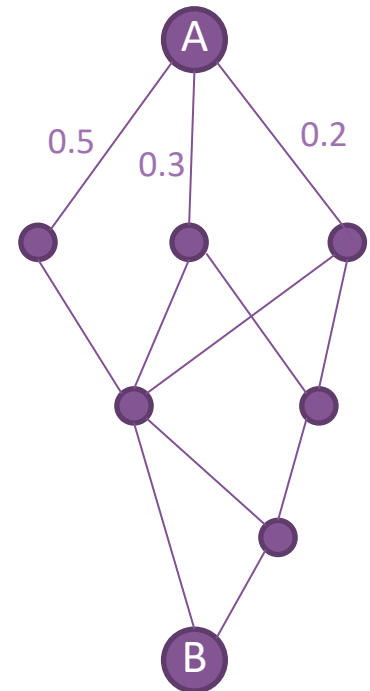
## The idea

### Abstract

- The implementation of the ant colony optimization in JCT is a visualization which allows the user to decrypt a cipher text which was encrypted by a transposition cipher.

### Functionality

- The ant colony algorithm is an efficient algorithm for solving combinatorial problems.
- E.g. it can be used to find the shortest path from A to B in a graph.
- The algorithm appreciates the way of ants quickly finding their path to a desired location.
- In the algorithm an ant chooses its path based on local information (e.g. information stored in the edges of the graph) and depending on decisions of preceding ants.
- The more ants choose a certain way, the more ants follow. This behavior is called swarm intelligence.
- In principle, this algorithm is based on statistical evaluations.



# The ant colony optimization

## The implementation in JCT

### In the menu

„Visuals“ \ „Ant Colony Optimization“

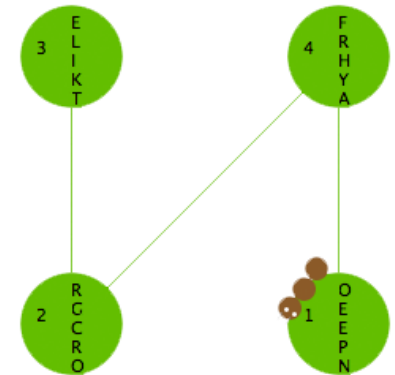
### The algorithm in application

- Ciphertexts encrypted with a simple column transposition cipher can be decrypted with the ant colony optimization.
- To do so, the key length  $n$  is needed and the ciphertext is written row wise in  $n$  columns. These columns will then be arranged as a graph.
- Different pairs of characters arise by concatenating the columns in different orders. In each language, these combinations of characters appear with differing frequencies. Weights on edges in the graph are calculated based on these probabilities, and frequency of an ant following a preceding ant.
- In each iteration a possible plain text is generated from a different ordering of the columns. The resulting text is then compared and rated with a given list of words of a language.
- The rating influences the pheromone matrix. The decisions of following ants is based on this pheromone matrix and hopefully these ants will find the right solution.

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
...	...	...	...



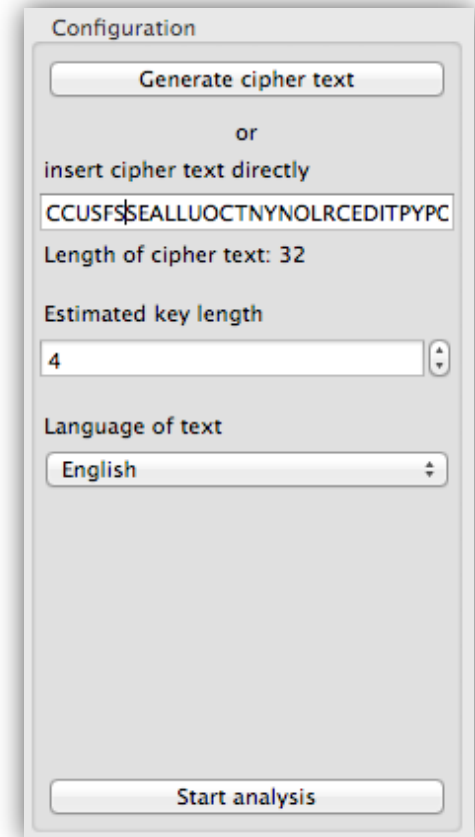
# The ant colony optimization

## Application sample 1/2

Try to decrypt the following text:

**CCUSFSSEALLUOCTNYNOLRCEDITPYPONO**

- Paste this sequence of characters into the text field next to „Insert ciphertext directly“ and choose 4 as the key length<sup>[1]</sup>.
- Press on „Start analysis“.



The screenshot shows a 'Configuration' dialog box with the following elements:

- A button labeled 'Generate cipher text'.
- The text 'or' centered below the button.
- The text 'insert ciphertext directly' below the 'or'.
- A text input field containing the ciphertext: 'CCUSFSSEALLUOCTNYNOLRCEDITPYPC'.
- The text 'Length of cipher text: 32' below the input field.
- The text 'Estimated key length' above a spin box.
- The spin box containing the value '4'.
- The text 'Language of text' above a dropdown menu.
- The dropdown menu showing 'English'.
- A button labeled 'Start analysis' at the bottom.

[1] The length of the key can be estimated with statistic evaluations.

Additionally, here the length of the ciphertext has to be a multiple of the length of the key.

# The ant colony optimization

## Application sample 2/2

Now the two frames “Analysis” and “Visual” are activated.

There you have the following parameters:

### Alpha & Beta:

- These parameters influence the probability of an ant to choose a certain edge. The higher the value of alpha, the more often an ant chooses a path a preceding ant had already chosen. The higher beta, the more important are bigrams of characters.

### Evaporation:

- A high evaporation lets the pheromone - dropped by an ant on its way – evaporated much faster. So following ants will find a less intensive pheromone trace and will be influenced less.
- The pheromone matrix is calculated by these three parameters and indirectly controls the ants. More precise information can be found in the help.

### Ant controller:

With the buttons in this sub-frame the ants can be steered on their path through the graph.

The screenshot displays the JCryptTool 1.0 interface, divided into two main panels: "Analysis" and "Visualization".

**Analysis Panel:**

- Level of detail:** Radio buttons for "Knot-by-knot" (selected) and "Multiple iterations".
- Animation:** A checked checkbox.
- Ant controller:** Three buttons: "To next knot", "Until last knot", and "Place new ant".
- Algorithm settings:** Three sliders: "Alpha: 0.8", "Beta: 0.8", and "Evaporation: 0.9".

**Visualization Panel:**

- Radio buttons for "Visualization as graph" (selected) and "Show pheromone matrix".
- A graph with four nodes: "USLTO" (3), "SEUNL" (4), "CSLGN" (2), and "CFAOY" (1). The nodes are connected by green lines.
- A small ant icon is positioned near the "CFAOY" node.

# The ant colony optimization

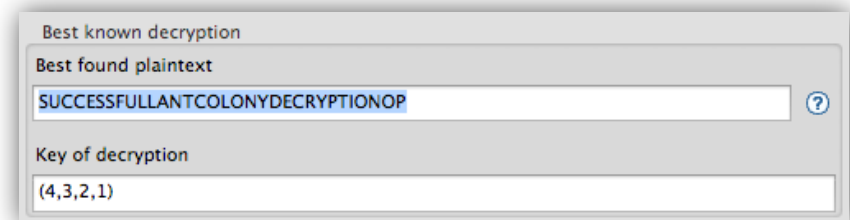
## Educational objective



## Result

- Did you manage to decrypt the text given on page 20?
- As plaintext (after approx. 25 iterations<sup>[1]</sup> with  $\alpha=0.8$ ,  $\beta=0.8$ ,  $\text{evaporation}=0.9$ ) you should get:

SUCCESSFULLANTCOLONYDECRYPTIONOP<sup>[2]</sup>



## Conclusion

- The permutation cipher is not a secure encryption method.
- The ant colony algorithm is an efficient algorithm to solve different combinatorial optimization problems. Not only in the field of cryptanalysis.
- For many problems nature has already found a solution, its just necessary to detect, understand and abstract this solution.

[1] The number of iterations diverges a lot. It can happen, that a solution has not yet been found after 50 or more iterations. Then it pays off to restart the plugin and start from scratch.

[2] Pad character which where appended to the ciphertext such that its length is divisible by 4.

# Viterbi analysis

## The idea



### The problem

- Given is a running key ciphertext resulting of two plaintexts which were combined by either an XOR or a modular addition.
- Is it possible to regain the original two plaintexts?

Indeed, it is possible. The Viterbi algorithm is designed to solve such a problem.

### Functionality

- The Viterbi algorithm is a recursive algorithm which uses the method of dynamic programming.
- The algorithm analyses probabilities of hidden Markov chains in a given input sequence.
- Beside cryptanalysis, the algorithm is also used in a broad range of other fields, e.g. in voice recognition or analysis of DNA structure. It is also used for the reduction of errors in transmissions.
- See [http://en.wikipedia.org/wiki/Viterbi\\_algorithm](http://en.wikipedia.org/wiki/Viterbi_algorithm)



# Viterbi analysis

## The implementation in JCT



### In the menu

„Visuals“ \ „Viterbi“

### The algorithm in the cryptanalytical application

- The basic concept of the algorithm is statistical evaluation of the probability of N-grams combined with the usage of a dictionary of the language in which the ciphertext is written.
- The model of the the analysis is set up with the knowledge that the ciphertext was originally constructed by modular addition or by XOR.
- The ciphertext is iterated letter by letter and possible letters for the plaintext are calculated. Surrounding letters build N-grams and their probabilities in the given language will be included in the reconstruction.
- The different possible letters at each position form different paths for different possible plaintexts. For each path a probability is generated and more unlikely paths won't be considered anymore.

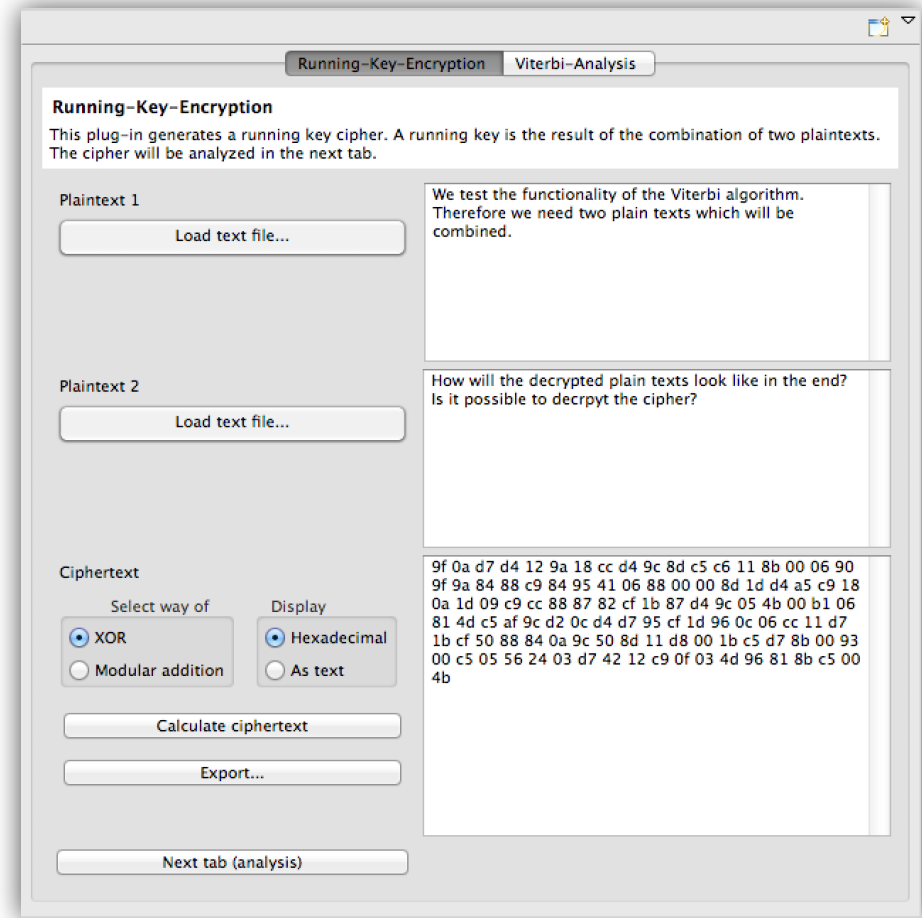
# Viterbi analysis

## Application sample 1/2

First, we have to generate a ciphertext which can be handled by the Viterbi analysis.

The plugin therefore comes with a special generator for texts.

- Type in two plaintexts or load plaintexts from files.
- You can decide in which way (XOR or modular addition) the plaintexts are combined – letter by letter.
- By clicking on “Calculate ciphertext” a ciphertext is calculated from the given plaintexts.
- Press on “Next tab (analysis)”.



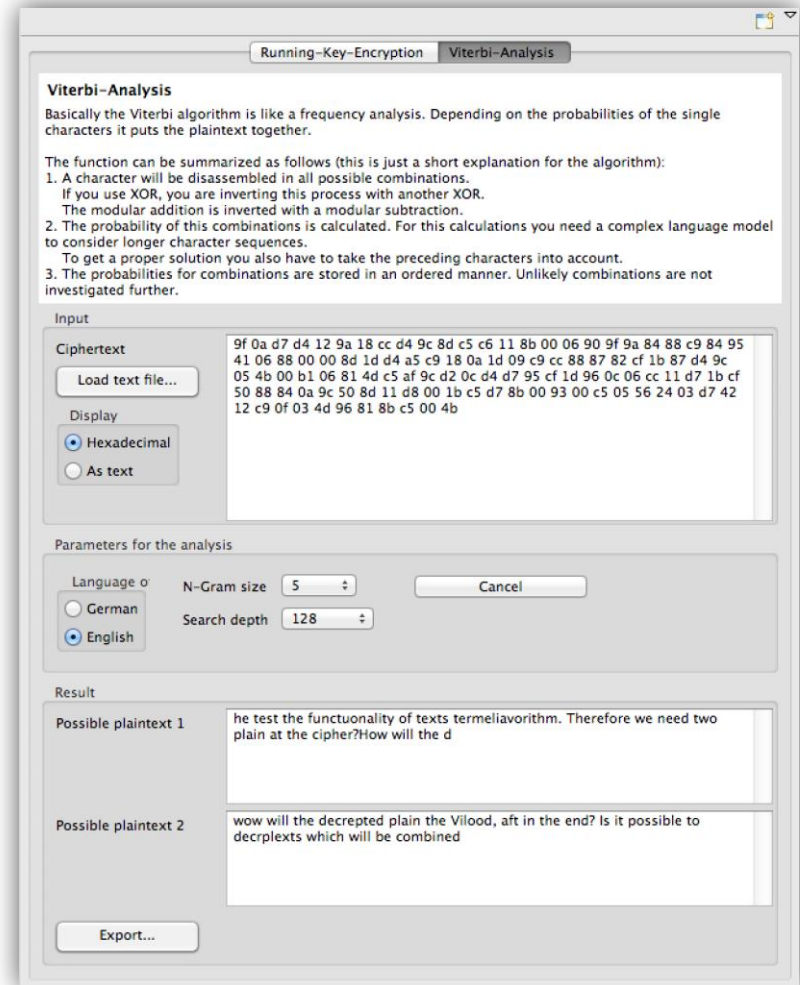
# Viterbi analysis

## Application sample 2/2

In the next step, the „Viterbi analysis“, the Viterbi algorithm is applied to the ciphertext. The algorithm tries to gain information about the two original plaintexts.

- Choose the language you guess the plaintexts are written in.
- Possibly, adjust the size of the N-grams and the depth of search and click on “Start analysis”.
- In the lower two text areas the results are being calculated now. One can observe how the sequences of letters are being generated dynamically.  
This can take some seconds.  
The best way to watch the whole dialog is full screen.

How does the size of the N-grams and the search depth affect the algorithm?



# Viterbi analysis

## Educational objective



### Conclusion

- Plaintexts which were encrypted by modular addition or XOR can be decrypted with the help of the Viterbi algorithm.
- A disadvantage of the algorithm: The beginning of the revealed plaintexts is often not decrypted correctly. Surrounding N-grams are missing and paths of probabilities are not yet calculated.
- Long words are seldom decrypted in the right way.
- The used dictionary plays an important role for the quality of the resulting text, because it is the source of words for the algorithm.
- Only N-grams which are contained in the dictionary are found. The length of N-grams is limited in the plugin by  $N=5$ . As otherwise the dictionary needs to contain all words with length  $N$ . For  $N=7$  there are already a lot of words more.
- The variation of the size of N-grams and the depth of search directly influences the result.
  - The size of N-grams determines which words from the dictionary are used.
  - The parameter depth of search determines how many candidates for plaintext pairs (paths) are used for the analysis of the next character (the algorithm discards after each character unlikely paths).

# Verifiable Secret Sharing

## The idea



### The problem

- Verifiable Secret Sharing (VSS) is an enhanced variant of **Secret Sharing**.
- Secret Sharing is about sharing a common secret between multiple persons or players. Each player receives a so called “share”.
- A small number of players, but not all, is needed to reconstruct the common secret.
- A single share or less than the predefined minimal amount of needed shares shall be useless.

### The enhancement “Verifiable”

- VSS is more secure than normal Secret Sharing. Before sharing the secret, one person, the “dealer” needs to know the secret to share it. Before handing out the shares, he can easily modify the shares and so make them useless.
- To resolve this problem, the dealer hands out “commitments” to each player. With a commitment, each player is able to test whether his share is right or not.

# Verifiable Secret Sharing

## The implementation in JCT



### In the menu

“Visuals” \ “Verifiable Secret Sharing”

### The algorithm applied

- The secret is represented by a number (instead of a secret in form of a text). So a transformation between the text and the number is necessary.
- Each of the  $n$  players receive a share. For reconstruction of the secret any  $t$  shares ( $1 < t \leq n$ ) shall suffice.
- In mathematics, a polynomial of degree  $(t-1)$  can be reconstructed by the knowledge of  $t$  points which lie on the graph. This can be done with the so called Lagrange interpolation.
- This mathematical knowledge is used in a clever way by VSS.
- The secret is stored in the absolute term of a polynomial. Therefore the secret is simply the evaluation of the polynomial at the point 0.

# Verifiable Secret Sharing

## Application sample 1/2

### First step

- Choose the number of players  $n$  and the minimal number of players  $t$  which is needed to reconstruct the secret.
- Determine the secret in form of a number.

The numbers “Safe prime”, “Prime factor” und “Generator” are calculated automatically, if possible.

- Click on “Determine coefficients”.

Parameters

Number of players $n$	6
Number of players $t$ for reconstruction	5
Secret $s$	10
Safe prime $p$ ( $p > 2s$ )	23
Prime factor $q$ ( $2q = p - 1$ )	11
Generator $g$	2

Next step:

### Second step

The polynomial can now be specified. As a dealer, here you can influence the polynomial from which the shares are calculated. The commits are as well generated from the polynomial.

- The initial polynomial gives player 1 too much information. So you should generate random coefficients via the button “Generate”.
- Press “Commit” to calculate the commits.

If you change the polynomial now, and later check the shares with the previously generated commits, then the shares are marked as invalid.

- Click on “Calculate shares”.

Coefficients

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

$P(x)$

Next step:



# Verifiable Secret Sharing

## Application sample 2/2

### Step of reconstruction

The secret is shared between the players.

- The shares can be checked for validity by clicking on “Check”.
- In our example on the right, the polynomial was changed after generating the commits. Therefore the shares are invalid and the dealer should not be seen as trustworthy.

Commitments		Shares		Reconstruction	
Coefficient	Commitment $Y_i$	Player	Share	Valid	Selected
$a_0$	4	Player 1	14	4 (Red)	<input type="checkbox"/>
$a_1$	3	Player 2	70	0 (Red)	<input checked="" type="checkbox"/>
$a_2$	5	Player 3	232	2 (Red)	<input checked="" type="checkbox"/>
$a_3$	4	Player 4	584	4 (Red)	<input checked="" type="checkbox"/>
$a_4$	3	Player 5	1234	4 (Green)	<input checked="" type="checkbox"/>
		Player 6	2314	4 (Red)	<input checked="" type="checkbox"/>

Interesting is the fact, that one share was marked as valid even though the polynomial was changed. So it is necessary to check multiple shares for validity.

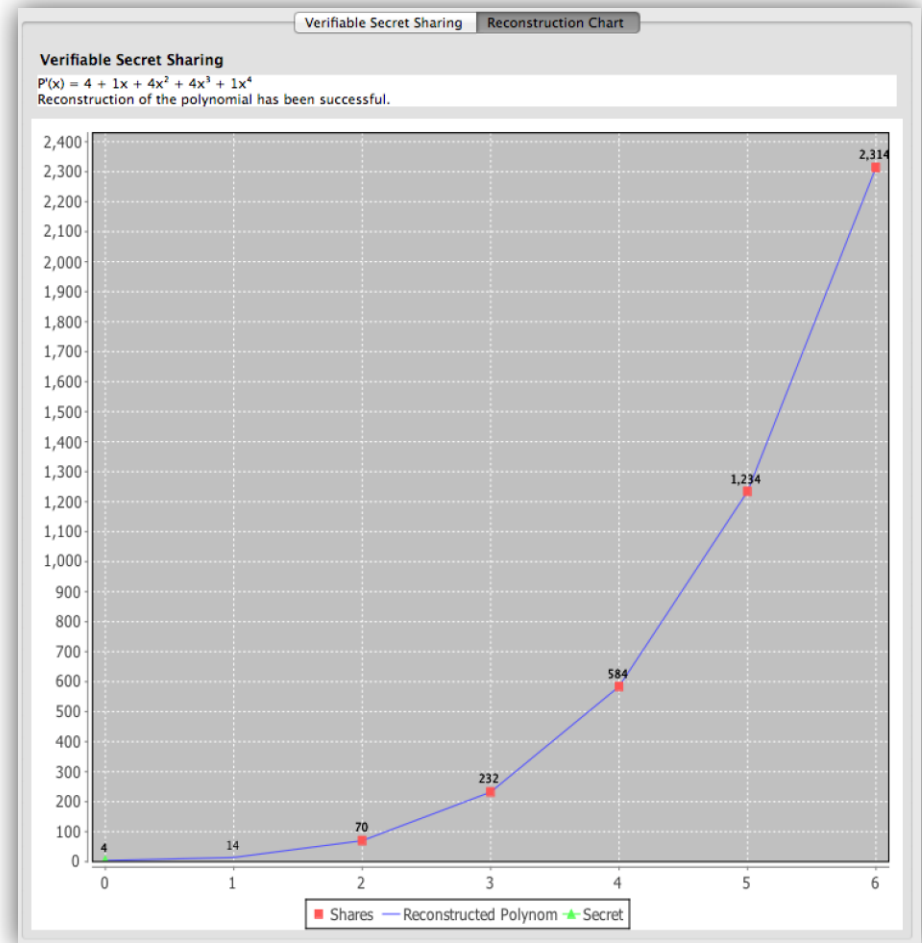
- To reconstruct the secret, the players whose shares shall be used can be selected (see screenshot on the right).
- For our example we selected five players which is as well the minimal number of shares needed. As we have  $t = 5$ .
- Now the secret can be reconstructed by clicking on “Reconstruct” (invalid shares don’t necessarily deliver a wrong secret).

# Verifiable Secret Sharing

## Educational objective

### Conclusion

- A secret can be split between multiple persons such that it can be decrypted only in the group.
- E.g. multiple ambassadors can transmit volatile data without them knowing the important secret.
- A tolerance can be implemented such that not each of the ambassadors is needed for reconstruction.
- In the VSS, again a mathematical model, the Lagrange interpolation, can be used in an important application.



# Signature demonstration

## The idea



### The problem

1. The author of electronic documents cannot be checked à priori. An attribute to verify the author is needed. This can be a signature.
2. Having only an electronic document one hardly can notice a belated change.

To solve this problem, an author can digitally sign his document.

### Functionality

- The author generates a hash value from the document (see slide [41](#)).
- The hash value is encrypted with the private key of the author (if using RSA, signing is equivalent to encrypting with the private key).
- The encrypted hash value and the used hash function are made available to the public or to the receiver next to the document.
- A person who is interested in the integrity of the document, can use the public key of the author to decrypt the hash value of the document.
- By calculating the hash value of the received document and comparing it to the decrypted hash value, it is easy to ensure that the document was finally changed by the named author.

# Signature demonstration

## The implementation in JCT

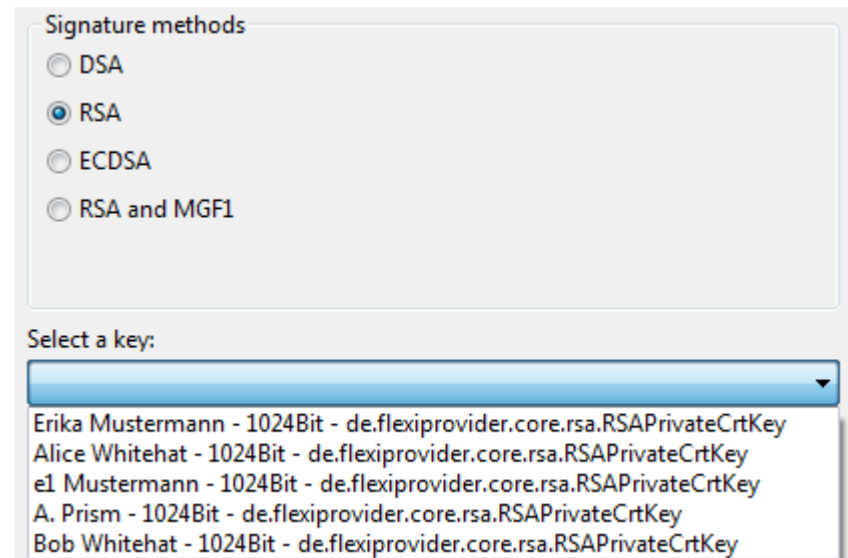


### In the menu

“Visuals” \ “Signature Demonstration”

### The algorithm applied

- The plugin is capable of digitally signing a document, such as a file or an arbitrary text typed in.
- As hash method one can choose between MD5, SHA-1, and SHA-2 (SHA-256, SHA-384, and SHA-512).
- Finally, depending on the chosen hash function it is possible to choose between DSA, RSA, ECDSA, or RSA with MGF1 as signature method.
- Below that you can choose the subject (key owner) who owns an according key for the chosen signature method.\*



- \* There are two ways to create according keys for the subjects (key owners, users):
- a) within the Algorithm Perspective.
  - b) with the visualization plugin “Public-Key Infrastructure” (JCT-PKI).

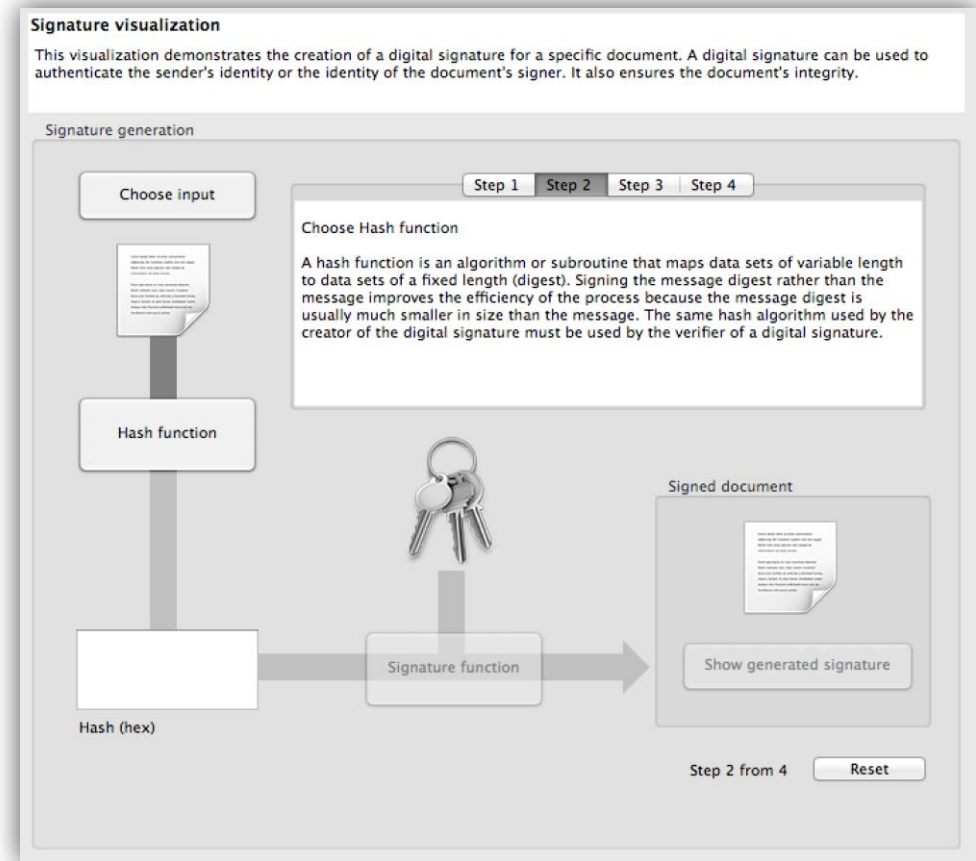
# Signature demonstration

## Application sample 1/2

Signing a document is neither hard nor elaborate. And can be done in two simple steps.

### First step: Create hash value

- Via “Choose input” the document to be signed can be chosen.
- A dialog appears and either a text can be typed in directly or an arbitrary document can be opened with “From file”.
- Next a “Hash function” must be selected.
- The hash value is then generated and is shown in the lower left area. This hash value is an electronic fingerprint of the document.



# Signature demonstration

## Application sample 2/2

### Second step: Create signature

- Clicking “Signature function”, an encryption algorithm for the hash value can be selected.
- We choose “ECDSA” as signature method. Below we choose from the JCT keystore a key of the signing person (here: “Alice Whitehead”).
- By clicking on “Finish”, the signature is generated and can finally be displayed via “Show generated signature”.

Owner of signature: -

Used key/curve: ANSI X9.62 prime256v1 (256 bits)

Signature method: SHA384withECDSA

Signature

Address	Hex	Ascii
0000	30 44 02 20 02 E9 72 99 19 80 A4 A1 4B 5E	0D éππKΛ
0000E	14 2A 6F 8D C8 16 9F 74 71 52 73 39 5A 3D	*oEtqRs9Z=
0001C	B4 C4 CF C0 2D 5F 3B 15 02 20 6D B0 1C 44	'AlA-_: m"D
0002A	63 38 CA 91 F8 C8 0D 40 58 ED 79 A2 0F AD	c8ÈøÈ
00038	73 78 BE 92 12 95 C0 E4 CA 58 58 5F 75 1B	sx³ÀãÈXX_u

Length of signature: 560 Bits

Display options for the signature

Hex dump (hex and ascii)  Octal  Decimal  Hex

Signed message

Address	Hex	Ascii
00000	53 6F 6D 65 20 61 72 62 69 74 72 61 72 79	Some arbitrary
0000E	20 74 65 78 74 20 79 6F 75 20 68 61 76 65	text you have
0001C	20 74 6F 20 74 79 70 65 20 69 6E 20 68 65	to type in he
0002A	72 65 2E 20 49 74 20 77 69 6C 6C 20 62 65	re. It will be
00038	20 74 68 65 6E 20 73 69 67 6E 65 64 2E	then signed.

Length of signed message: 552 Bits

To show the signed file and the signature, click on "Save" and then open the saved file with the hex editor from JCT.

Save Close

# Signature demonstration

## Educational objective



### Conclusion

- The integrity of electronic documents can be checked with the help of electronic signatures.
- Cryptographic algorithms help to verify the author and the integrity of the document.
- If a document was changed in any way, the hash value changes.
- To make sure the document was created by the named author, its author signs it with his private key. Only with the “right” public key (the one from the signing person) it is possible, to validate the original hash value (to verify the integrity of the document). So the hash value can be publicly accessible, without being endangered to be changed.

# Extended RSA cryptosystem

## The idea



### Which encryption ciphers are used nowadays, which guarantee security?

- For data which is transmitted over public channels, an encryption method should be used. One possible cipher for such tasks is the RSA cryptosystem (if used with the correct parameters).
- The RSA cipher is an asymmetric method. To communicate each participant needs two keys, a private and a public key. These two keys have to be generated first.
- Data which was encrypted with the public key of one participant can only be decrypted with the corresponding private key.
- To communicate in an encrypted manner with another person, one has to have his public key. Therefore, the public keys have to be exchanged preliminarily.  
A “Certificate Authority” (PKI) is often used to simplify the process. This “authority” saves, manages and verifies the public keys of the possible communicators and generates certificates.  
→ See the visualization plugin “Public-Key Infrastructure” (JCT-PKI)  
which visualizes the processes within a PKI with its instances User, RA and CA.



# Extended RSA cryptosystem

## The implementation in JCT



### In the menu

“Visuals” \ “Extended RSA Cryptosystem”

### Functionality

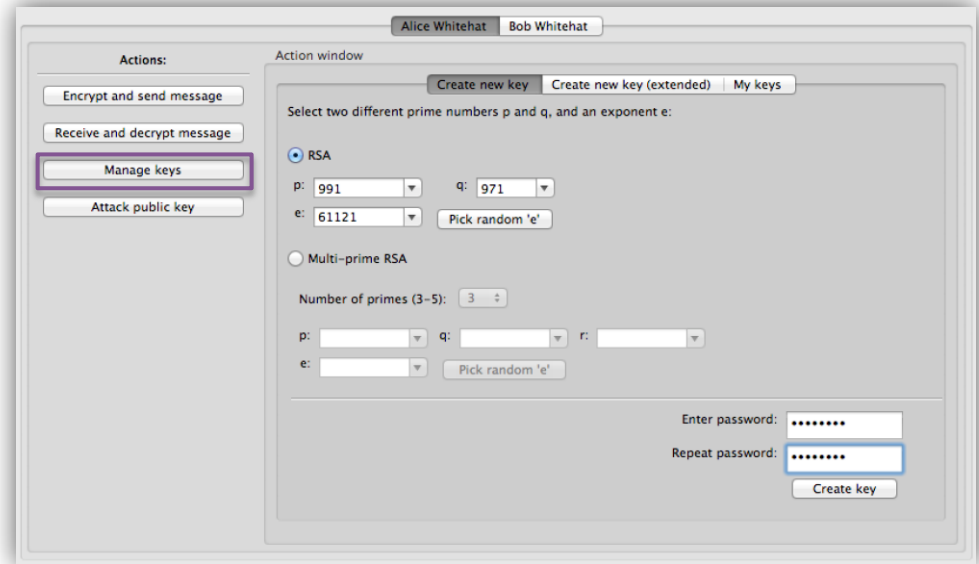
- This plugin implemented in JCT helps managing identities and their associated keys, and it offers a complete independent communication platform to send and receive messages.
- Further on, it is possible to attack the system via attacking the key. Therefore a brute-force method is used to factor the modulus “ $n$ ” into its primes.
- The user can experiment and find security holes of the RSA cryptosystem.

# Extended RSA cryptosystem

## Application sample 1/2

### Generation of primes

- First, we generate a key which can then be attacked.
- Therefore the plugin provides the option “Manage keys”. We choose primes  $p$  and  $q$  and a random  $e$ .
- Finally, the key has to be saved in a keystore using a password, which we enter in the lower right.
- Now we have created a key for the identity “Alice Whitehat”. Next we try to attack the keys. Using the RSA cryptosystem this means solving the factorization of the modulus  $n = p * q$ .
- As Alice knows the keys and will not attack her own keys, we switch the tab to “Bob Whitehat”. (Alice and Bob are default identities in JCT.)



# Extended RSA cryptosystem

## Application sample 2/2

### The attack

- Bob Whitehat is now able to attack the public key of Alice.
- So, you click the button “Attack public key” on behalf of Bob and choose the according key of Alice. The previously generated key can be recognized at its bit length (here 20 bit).
- Due to the short bit length, the key can be attacked via the button “Attack key”. So the key generated by Alice can be factorized without knowing the primes  $p$  and  $q$ .
- Here, the factorizing is done only via a brute-force attack.
- A bit length of only 20 bit for the modulus does by far not offer appropriate security for the RSA cipher.

The screenshot shows the JCryptTool 1.0 interface. On the left, under "Actions:", the "Attack public key" button is highlighted with a purple border. A dropdown menu is open, listing several RSA public keys. The selected key is "Alice Whitehat - 20Bit - RSAPublicKey - KeyID: 5". Below the dropdown, the modulus  $N$  is shown as 962261. A message states: "The factorization was successful. The following values were reconstructed in 0.022 seconds:". Below this, a table displays the reconstructed parameters:

Parameter	Value
$p$	971
$q$	991
$e$	733
$d$	233197

# Extended RSA cryptosystem

## Educational objective



### Conclusion

- The factorization methods allows us to factorize numbers with a short bit length in almost no time. Given a modulus  $n$  with only 64 bit (binary representation of the number has 64 digits, which is around 20 decimal digits, like the number  $2^{64}-15$ ) for instance can be factorized with a current notebook (Intel Core i7 2,4GHz) in less than a second.
- Once an attacker can find a factorization of the modulus  $n$ , the messages which are sent from the associated identity can be decrypted by the attacker.
- Nowadays, bit length of 2048 bit are rated as secure.

### And more ...

- The plugin offers the possibility to send and attack messages encrypted with the RSA cipher.

# SETUP attack on the RSA key generation (Kleptography)

## The idea

### Problem

- There are some “backdoor” attacks, which make the RSA cipher insecure.
- Most of these attacks start by modifying the key generation. The user needs to rely on the random generation of the primes – this is not always possible.
- The SETUP ("secretly embedded trapdoor with universal protection") attack is such an attack where the generation of the key is modified.

A short description of the attack:

### Functionality

- Some extra values and keys are injected into the system.
- The public keys, which are needed by the RSA method, are modified such that information needed for decryption can easily be extracted by the attacker. However, without knowing the implementation of the key generation, one can hardly detect that it is not really random.

# SETUP attack on RSA

## The implementation in JCT



### In the menu

“Visuals” \ “Kleptography”

### Functionality in detail

- Generally the RSA cipher uses two randomly generated private primes  $P$  and  $Q$ . Their product, the modulus  $N = P \cdot Q$  is published.
- For the attack, initially the prime number  $P$  is generated, then this prime is encrypted with the public key of the attacker. Next the prime  $Q$  will be chosen such that the first digits of the modulus  $N$  represent the encrypted value of  $P$ .
- As  $N$  is publicly available, the attacker can easily reveal the prime  $P$  by decrypting the first digits of the modulus  $N$  with his own private key, and the cipher is hacked.
- As only the encrypted prime number  $P$  is part of the modulus  $N$  and  $P$  was randomly chosen, the modulus seems to be random too.  
Moreover, as  $P$  will be regenerated for each new pair of keys the attack is not detectable without reverse engineering the code of the key generator.

# SETUP attack on RSA

## Application sample 1/2

The attack is divided into two main steps: the generation of the keys and the decryption by the attacker.

### Key generation

- In the dropdown menu choose the method “Attack 4: SETUP”.
- First, the two keys of the attacker have to be generated. This is done by “Generate new attacker keys”.
- Next, the primes P and Q which are used in the ordinary key generation can be generated.
- The prime Q will be chosen such that the modulus N contains the encrypted prime P (marked yellow in the figure).
- Finally, N and D can be generated. Then, in the lower third part of the plugin a plaintext can be encrypted.
- By clicking on the button “Save public key and ciphertext”, the user can switch to the tab “SETUP attack” to continue and decrypt the ciphertext.

Key generation

Settings

Method: Attack 4: SETUP  Binary  Decimal  Hexadecimal

Key bit length: 64 (in decimal)

Additional cryptosystem values

Generate new attacker keys

Attacker's N: d7ffe043

Attacker's E: c457ec9d

Encrypted P: 80e4885c

N' (temporary composite): 80e4885c84c43eb7

Standard cryptosystem values

Generate all at once

Generate primes P and Q

P (prime): b9cedb01

Q (prime): b19580e1

Calculate N

N = P \* Q: 80e4885be4e3fbe1

E (public exponent): 10001

Generate random E

Restore default E

Calculate D

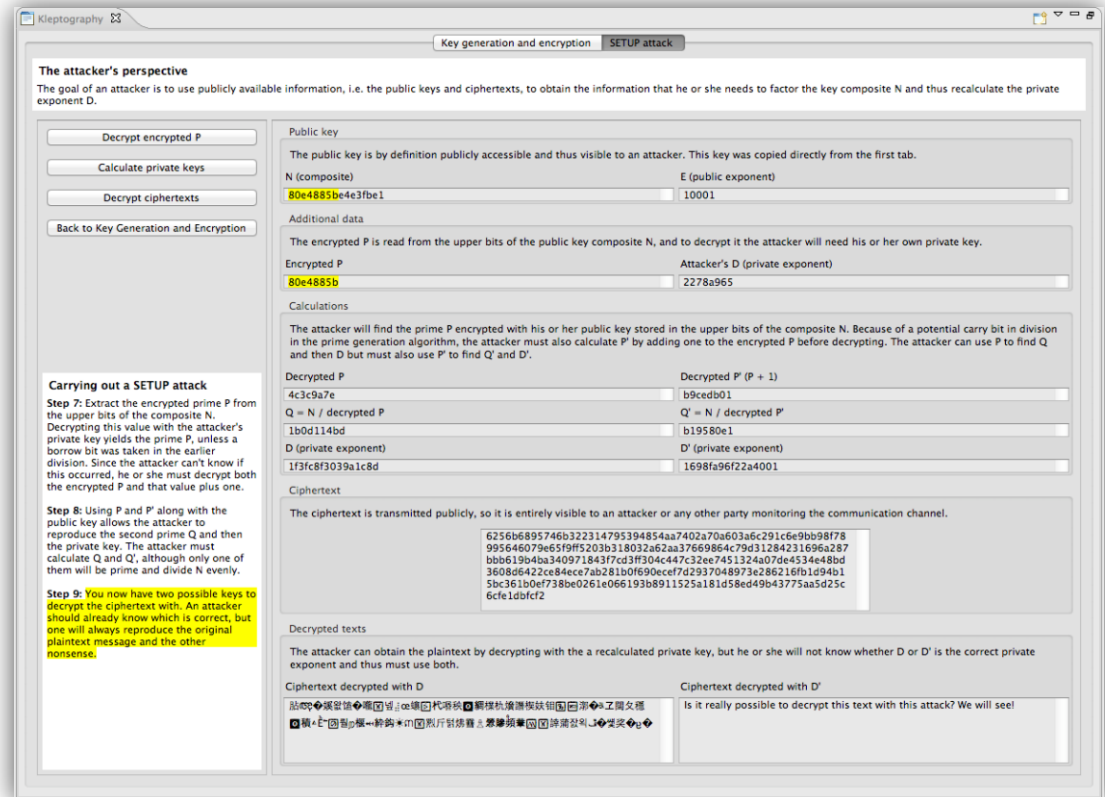
D (private exponent): 1698fa96f22a4001

# SETUP attack on RSA

## Application sample 2/2

### The decryption by the attacker

- Switch to the tab “SETUP attack”
- The data known by the attacker is directly shown in the appropriate fields: These are the keys of the attacker, the modulus N and the exponent E. The last two values are public, as the communication partner needs them to encrypt the text.
- Using the four buttons on the left, the text can be decrypted by the attacker.
- First, the encrypted prime P is extracted from the modulus N, and decrypted with the attacker’s private key.
- Because of a potential carry bit two different cases have to be analyzed.





# SETUP attack on RSA

## Educational objective



### Conclusion

- By cleverly encroaching the key generation, an attacker has the possibility to decrypt the cipher text with the use of his own keys.
- Almost all effective attacks on RSA attack the key generation. Therefore, one has to confide to the key generation, which is often done by a “Certificate Authority” (CA) or within a hardware security module (HSM).
- As the modulus  $N$  still appears to be random, as  $P$  and  $Q$  are chosen differently for each pair of keys, it is hard to detect the attack by just analyzing the output – without applying reverse engineering.
- For this attack, only the public key of the attacker is needed. So, revealing his attack does not cause any insecurity for his communication.

# Zero-knowledge protocol: Fiat Shamir

## The idea



### Problem

- A person A wants to convince a second person B that he knows a secret which person B knows as well.
- It is required to do the verification in public without revealing the whole secret. So a possible attack from a third person will disclose the secret.
- A solution for this problem is called a zero-knowledge protocol.
- An important characteristic for such a protocol is its need for honest players. A third person C shall not be able to convince B of knowing the secret, without really knowing it.

In this application sample we present the zero-knowledge protocol from Fiat Shamir. There exist a couple more zero-knowledge protocols, like Feige Fiat Shamir, or a version using an isomorphism for graphs.

# Zero-knowledge protocol: Fiat Shamir

## The implementation in JCT



### In the menu

“Visuals” \ “Fiat Shamir”

### Functionality

- The Fiat Shamir method relies on the difficulty of the following problem: Given an arbitrary number in the field modulo  $n$ , its square root can only be found by factoring of the number  $n$ .
- If the modulus  $n$  is a product of two unknown primes  $p$  and  $q$  which are chosen large enough, it is hardly possible to find the factorization of  $n$ .
- As the method operates on numbers, the secret  $s$  must be given as a number.
- Person A published the number  $v = s^2 \bmod n$ , generates a random number  $r < n$ , and receives another random number  $b$ .  $b$  is 0 or 1. Person B now receives from person A the number  $x = r^2 \bmod n$ .
- Person A calculates  $y = rs^b \bmod n$  and sends this number to person B. Person B verifies if the equation  $y^2 = xv^b \bmod n$  holds. If it does, the secret is verified, due to the fact:

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

# Zero-knowledge protocol: Fiat Shamir

## Application sample 1/2

### As prover

- Choose the radio button “Prover”.
- First, the two primes  $p$  and  $q$  have to be generated. Their product is the public modulus  $n$ . Additionally, the secret  $s$  has to be generated.
- In the section “Action flow” the steps which are required for the verification can be executed.
- All values – public and private ones – which are calculated during this process are shown in the lower part of the plugin.
- In this example, Alice performs the proof, as she actually knows the secret. Her communication partner will verify that she knows the secret (green hint in the lower left part of the figure).

The screenshot shows the Fiat Shamir application interface. The title bar reads "Fiat Shamir".

**Situation**

- Prover: Choose two prime numbers  $p$  and  $q$ . The digit  $n = pq$  is the public module. Alice knows a digit  $s < n$  with  $\gcd(s, n) = 1$  and  $v = s^2 \bmod n$ .  $v$  is published.
- Attacker

**Input for a prime number**

$p$ : 139       $n$ : 2!     

$q$ : 181     

**Action Flow**

- : Alice generates a random digit  $r < n$  and sends  $x = r^2 \bmod n$  to Bob.
- : Bob generates a random bit  $b$  from  $\{0,1\}$  and sends  $b$  to Alice.
- : Alice computes an answer  $y$  depending on  $b$  and sends  $y$  to Bob.  $y = rs^b$ .
- : Bob verifies Alice's answer. He checks, whether it's right:  $y^2 = xv^b$ .

**Information**

Bob	Alice	Public
$b$ : 0	Secret	$v = s^2 \bmod n$ 24303
$y^2$ : 1406	$r$ : 1664	$y$ : 1664
$x v^b$ : 1406	$s$ : 2903	$x = r^2 \bmod n$ 1406
Has been verified		

# Zero-knowledge protocol: Fiat Shamir

## Application sample 2/2

### As attacker

- On the other hand, the plugin offers the possibility to act as an “attacker” who pretends to know the secret.
- By cleverly choosing the values  $x$  and  $y$  it is possible to convince the other person in 50 percent of the cases that one knows the secret.
- This can be done in this scenario. By repeating the method multiple times, the probability to detect the attacker is  $1-(0,5)^n$ .
- The more often the test is repeated the higher is the probability to detect the attacker.

The screenshot shows the Fiat Shamir application interface. The 'Situation' section has the 'Attacker' radio button selected. The 'Input for a prime number' section shows  $p = 211$  and  $q = 167$ , with  $n = 3!$ . The 'Action Flow' section shows the steps: 'Generate random numb' (Carol generates a random digit  $r < n$  and a bit  $c$  from  $\{0,1\}$ . She sends  $x = r^2 v^A - c \bmod n$  to Bob.), 'Generate b' (Bob generates a random bit  $b$  from  $\{0,1\}$  and sends  $b$  to Carol.), 'Calculate answer' (Carol computes an answer  $y$  and sends  $y$  to Bob:  $y = r$ .), and 'Verify' (Bob verifies Carol's answer. He checks, whether it's right:  $y^2 = x v^A b$ .). The 'Information' section shows Bob's values:  $b = 1$ ,  $y^2 = 5711$ ,  $x v^A b = 20335$ . Carol's values:  $r = 2969$ ,  $s = ?$ ,  $c = 0$ . Public values:  $v = s^2 \bmod n = 13374$ ,  $y = 2969$ ,  $x = r^2 v^A - c = 5711$ . A red message at the bottom says 'Has not been verified'.

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

# Zero-knowledge protocol: Fiat Shamir

## Educational objective



### Conclusion

- Zero-knowledge protocols are methods which are used to convince someone else of owning a secret without handing out the secret.
- The Fiat-Shamir protocol is such a method.
- It is important to know that an attacker can fake the result with a probability of  $(0,5)^n$  .  
Here,  $n$  is the number of repetitions of the test. The more often the method is repeated the better is the quality of the result.
- Hint: If it is possible to factorize large numbers easily, then this method is not be secure anymore (this means, that then the above described probabilities don't hold any more).

# Android Unlock Pattern (AUP)

## The idea



### Problem

- Nowadays, smart phones offer – next to writing messages and calling – a lot more functions, e.g. checking mails, creating notes, or online banking.  
Using such functions implies storing much sensible data on the phone (or in a cloud).
- People who lost their smartphone often ask themselves, whether it is possible for others to access their data. How secure is the lock of the smartphone? What is the difference between the security of a common PIN and the Android Unlock Pattern which is used by Android devices.
- The Android Unlock Pattern is visualized in JCT, and in its online help the security evaluation is documented and compared with other unlock patterns.

# Android Unlock Pattern

## The implementation in JCT



### In the menu

“Visuals” \ “Android Unlock Pattern (AUP)”

### Functionality

- The Android Unlock Pattern can be used on smartphones running on Android to lock the screen. Typically nine points on the screen are arranged as a square. The user can create a pattern by connecting the dots (under certain rules). This pattern has to be entered before using the phone.
- In the visual in JCT the user can check different patterns concerning their security. Therefore, a security indicator is provided. The indicator shows the number of different patterns possible with the used number of points of the pattern.



# Android Unlock Pattern

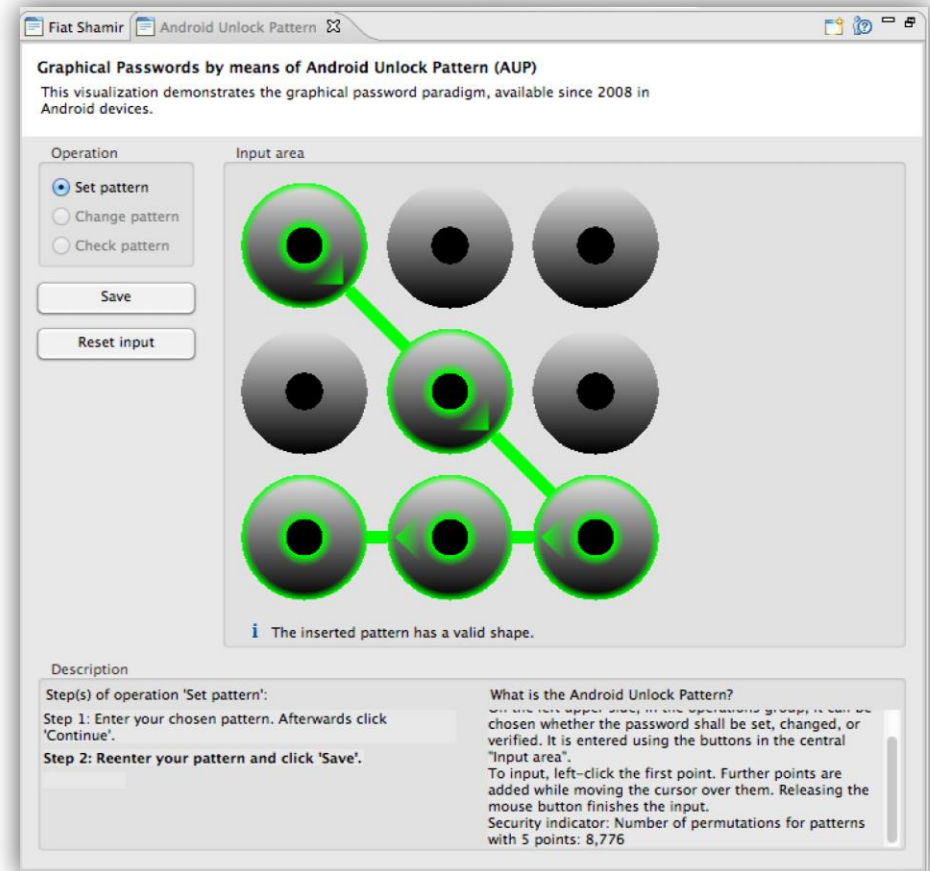
## Application sample

### Set pattern

- The visuals come along with the typical unlock screen of Android.
- First, a pattern can be set by clicking on one of the points and the moving the mouse over the other points. To finish the pattern you click on the last point of the pattern.
- Once created, in the lower right text field the security indicator shows the possible permutations of a pattern with the same amount of points.  
For instance, there are 8776 possible combinations for a pattern with five points.

### Change pattern, check pattern

- The plugin also provides the possibility to save a pattern, and then draw a second pattern to compare it with the saved one.
- The stored pattern can also be changed. Therefore, you either need to know stored one. If you forgot the pattern the visual can simply be reset.



# Android Unlock Pattern

## Educational objective



### Conclusion

- For a Android Unlock Pattern the order of the used points is important.
- A pattern for the Android unlock screen has to fulfill some rules. For example, each point can only be visited once.
- Due to this (and some more) rules the possible number of patterns shrinks. In total there are 389,112 different patterns.
- Comparing this AUP pattern to a 4 to 9 digit PIN of the numbers 1 to 9, where each number can be used only once, there are 985,824 different PIN combinations.

The Android pattern fulfills the following rule: A connection of two points, where the connection line crosses an unused point, is not a valid. If this rules was not applied, there would exist as many combinations as for the PIN, where each number can be used only once.

# Cascades in the Actions window

## The idea

### Functionality

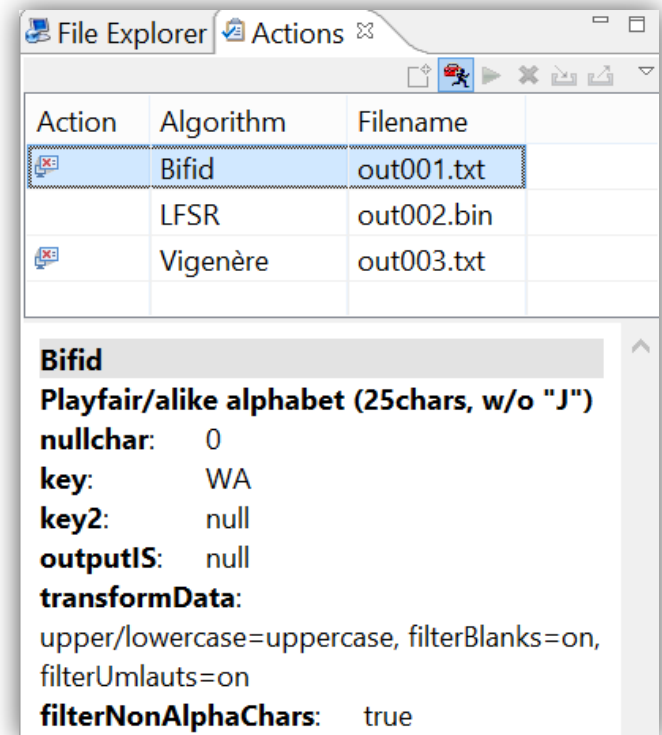
- In the Actions window, sequences of application of crypto methods (cascades) can be recorded and reapplied. Basically, it's a recorder and player for JCT functions.
- Arbitrary many function calls can be recorded and reapplied in the JCT Default Perspective.
- Cascades of classic crypto methods can also be viewed in the crypto console (see slide [73](#)).

### Examples of application

- Multiple files can quickly be encrypted or decrypted with the same algorithms, settings and ordering of the algorithm.
- Commutativity, the exchangeability of the order of different encryption algorithms, can easily be investigated with this cascade functionality (see slides [66](#) ff).

The Actions window allows to automate and re-run procedures – similar as with batch files on the command prompt.

In some cases, recorded cascades may not yield the exact same result after playback as was recorded.





# Cascades in the Actions window

## The implementation in JCT




### In the menu

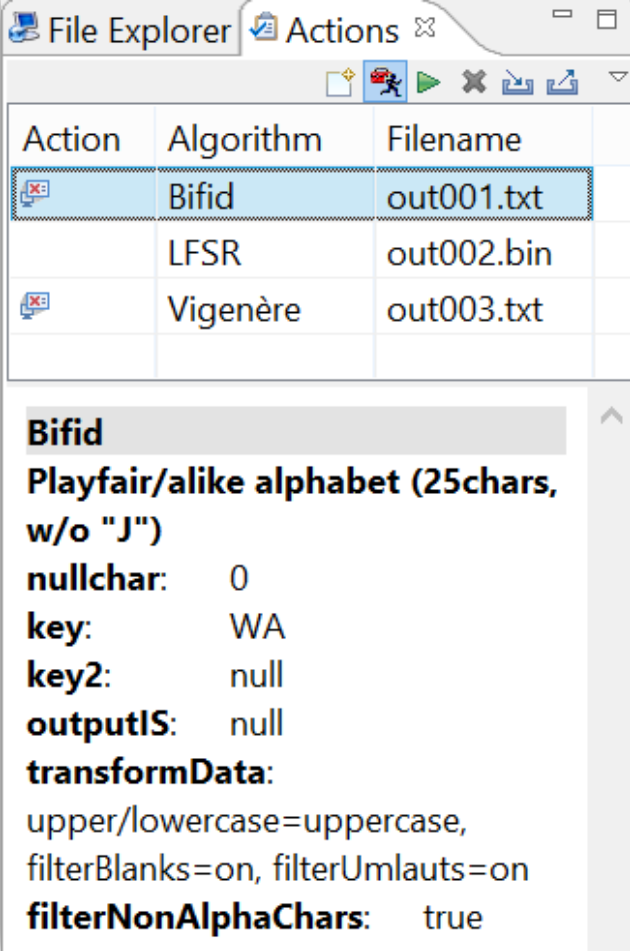
„Window“ \ „Show view“ \ „Actions“

### Create a recording



- To start recording a cascade press .
- All algorithms being executed now are recorded.
- To finish an recording just press  again.

### Edit, store and rerun a recording

- In the list below the toolbar, all algorithms are displayed in the order they have been processed.
- By selecting an algorithm, its execution details (e.g. alphabet, key etc.) are shown in the area below.
- Now the recorded cascade can be applied to an opened file in JCT by pressing .
- Use the buttons  and  to simply export or import a cascade simply (save as / load from a file).



The screenshot shows the 'Actions' window in JCT. It contains a table with three columns: 'Action', 'Algorithm', and 'Filename'. The first row is selected and shows 'Bifid' for the algorithm and 'out001.txt' for the filename. Below the table, the execution details for the selected 'Bifid' action are displayed.

Action	Algorithm	Filename
	Bifid	out001.txt
	LFSR	out002.bin
	Vigenère	out003.txt



**Bifid**  
**Playfair/alike alphabet (25chars, w/o "J")**  
**nullchar:** 0  
**key:** WA  
**key2:** null  
**outputIS:** null  
**transformData:**  
upper/lowercase=uppercase,  
filterBlanks=on, filterUmlauts=on  
**filterNonAlphaChars:** true

# Cascades in the Actions window

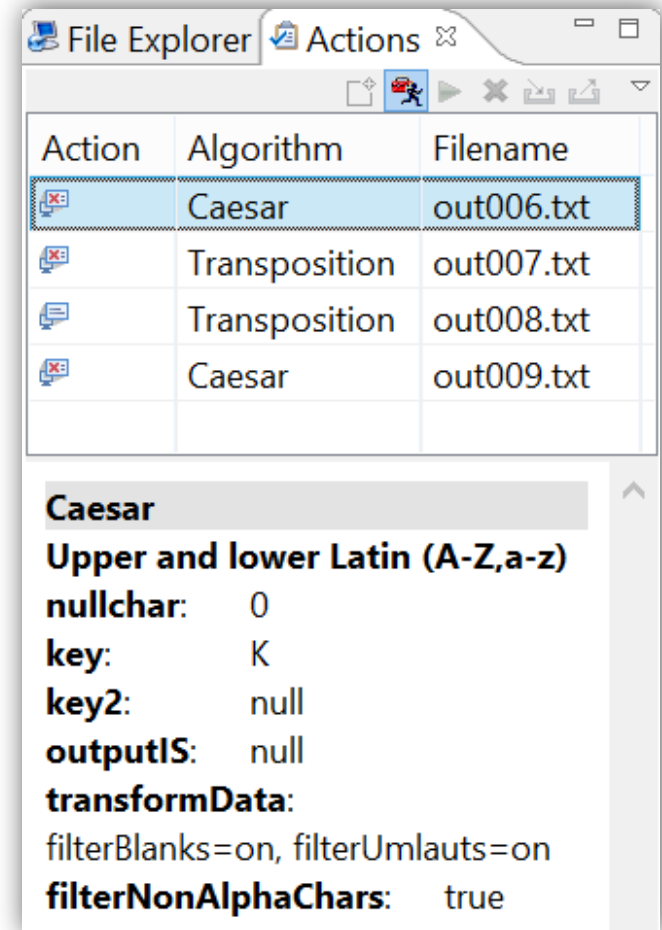
## Application sample 1/3

In this example we show, that the order of a Caesar and a transposition cipher can be exchanged in the decryption process (**commutativity**).

### A first recording

- Start recording the cascade with .
- **Encrypt** an arbitrary text with Caesar:  
„Algorithms“ \ „Classic“ \ „Caesar“
- Add a transposition cipher and **encrypt**:  
„Algorithms“ \ „Classic“ \ „Transposition“
- Apply a transposition **decryption** which reverts the last encryption: Therefore, simply use the same settings as for the encryption, but just use “Decrypt”.
- **Decryption** of the Caesar encryption applied first.
- Stop the recording with .

The action window should now look like the figure on the right.



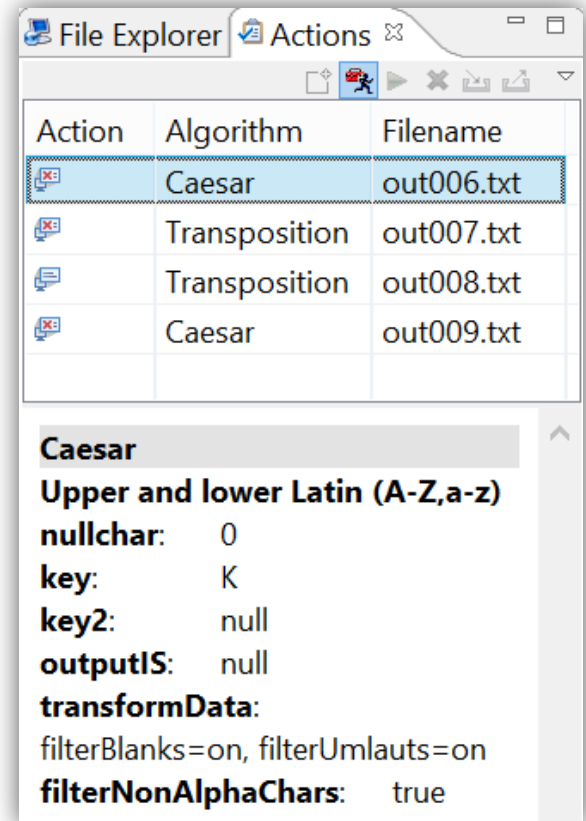
# Cascades in the Actions window

## Application sample 2/3

The cascade we created on the last slide should output a text unchanged, as each ciphertext will directly be decrypted afterwards.

### The current sequence of the algorithms

- Now there should be the following sequence of crypto operations, where E stands for encryption and D for decryption.
  - > E (Caesar)
  - > E (Transposition)
  - > D (Transposition)
  - > D (Caesar)
- Note the different layers of algorithms and its inverse. Such a structure will always output the original plaintext. So all the functions together form a identity transformation.
- So the **question** arises:  
When could we rearrange the order of the calls of the decryption algorithms such that a text will still be “decrypted” to itself?



# Cascades in the Actions window


## Application sample 3/3

Now we want to reorder our decryption algorithm and observe what happens to the output.

### Rearrange a recording

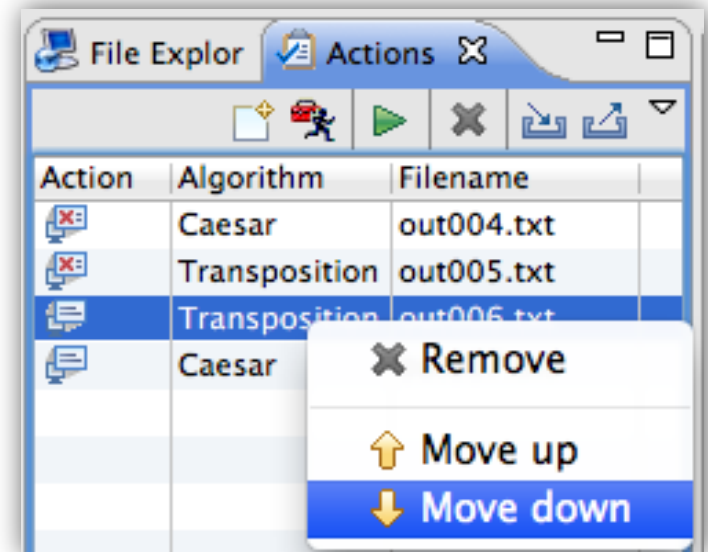
- By right-clicking on a row (e.g. the Caesar decryption), a context menu appears, allowing the user to exchange the position in the call stack (“Move up” / “Move down”).

### A new ordering

- Rearrange the stack to the following:
  - > E (Caesar)
  - > E (Transposition)
  - > D (Caesar)
  - > D (Transposition)
- Open a text file in JCT.
- Apply the new stack to the opened file by clicking on  .

What happens to the plaintext?

Does this also work with other encryption methods?



# Cascades in the Actions window

## Educational objective



### Conclusion

- The cascade function is perfect for saving and automatically applying different sequences of cryptographic operations to multiple files at once.

### Conclusion with a sample

- A text which was encrypted by the Caesar and a transposition cipher can be decrypted in arbitrary order. So these methods are commutative.
- This is possible as the Caesar method shifts each character by a fixed number of characters in the alphabet and the transposition cipher permutes each character in the text. Both methods are applied to the exactly same objects.  
It would be same by taking a monoalphabetic substitution instead of Caesar.
- Many methods (e.g. ADFGVX and Playfair) use a technique so called "fractioning".  
For instance, pairs of characters are substituted but then single characters are permuted. In this way substitution and transposition are not commutative any longer.

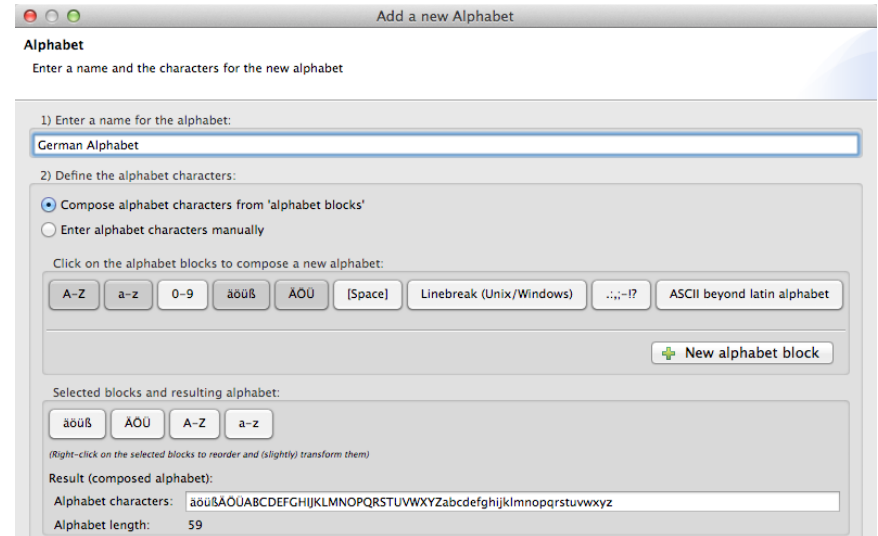


# Variable alphabets for classic algorithms

## The idea

### User-defined alphabets

- For most of the classic encryption algorithms (e.g. Vigenère), the ciphertext depends on the alphabet used in the plaintext.
- Frequently used alphabets are upper- or lowercase alphabets (A-Z, a-z) with or without digits (0-9).
- Many cryptographic tools restrict themselves to a fixed set of alphabets or characters, or an alphabet has to be entered manually.
- In order to improve the usability, a user should be able to easily create and test an encryption method with his own alphabets to get a feeling for the importance of encryption alphabets.
- JCT provides the following solution:
  - A custom alphabet can always be created for classic encryption algorithms in the appropriate encryption wizard.
  - Own alphabets can be built by arranging frequently used building blocks.

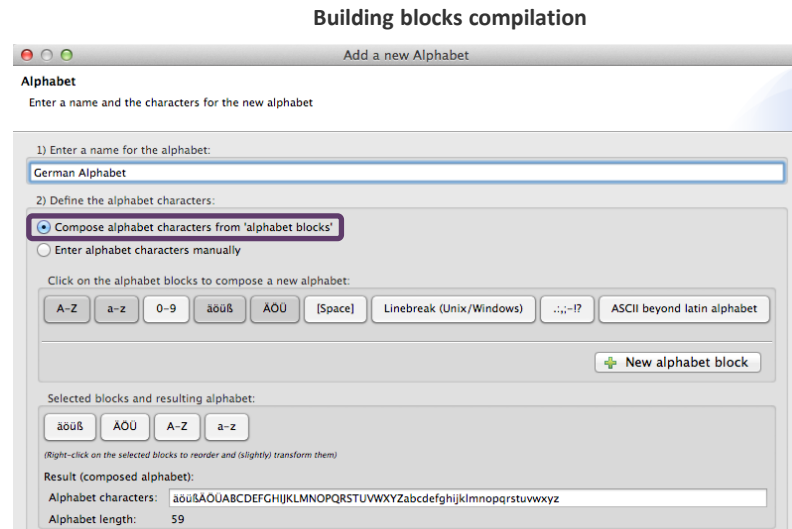
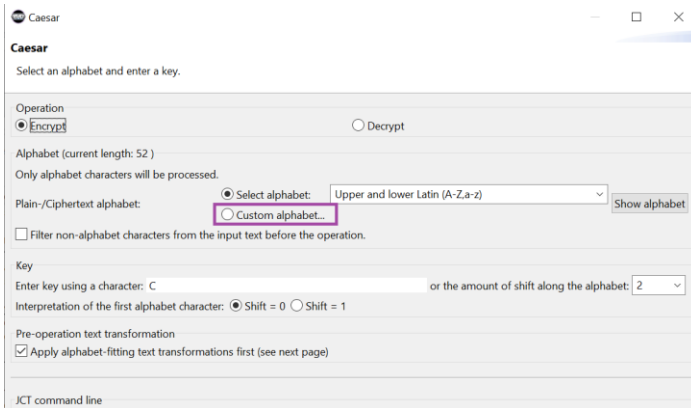


# Variable alphabets for classic algorithms

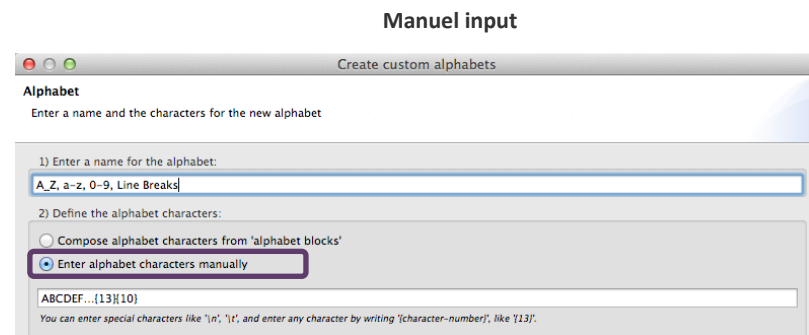
## The implementation in JCT 1/2

### Generate a custom alphabet

- If a method supports custom alphabets, a user can always provide and create an alphabet for de- and encryption on-the-fly.



- Special characters which are not on the keyboard, can be entered in curly brackets via their ASCII value. E.g. {10} represent a line break.



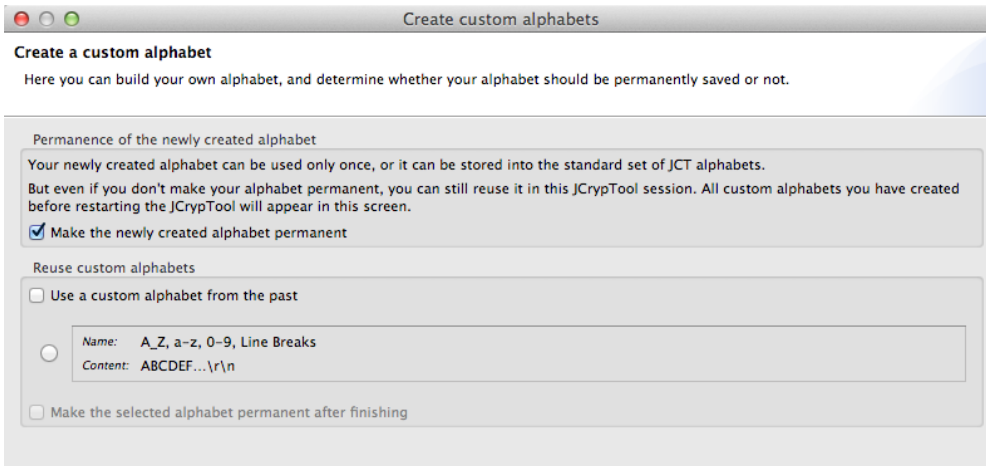
# Variable alphabets for classic algorithms

## The implementation in JCT 2/2



### Further hints

- Custom defined alphabets can be stored permanently.
- In a JCT session predefined alphabets can also be reused without storing them for permanent usage.



- The saved alphabets can be managed and later edited in the global settings in JCT:
  - Windows + Linux: „Window“ \ „Preferences/Settings“
  - MacOS: „JCrypTool“ \ „Preferences“

# Variable alphabets for classic algorithms

## Educational objective



## Conclusion

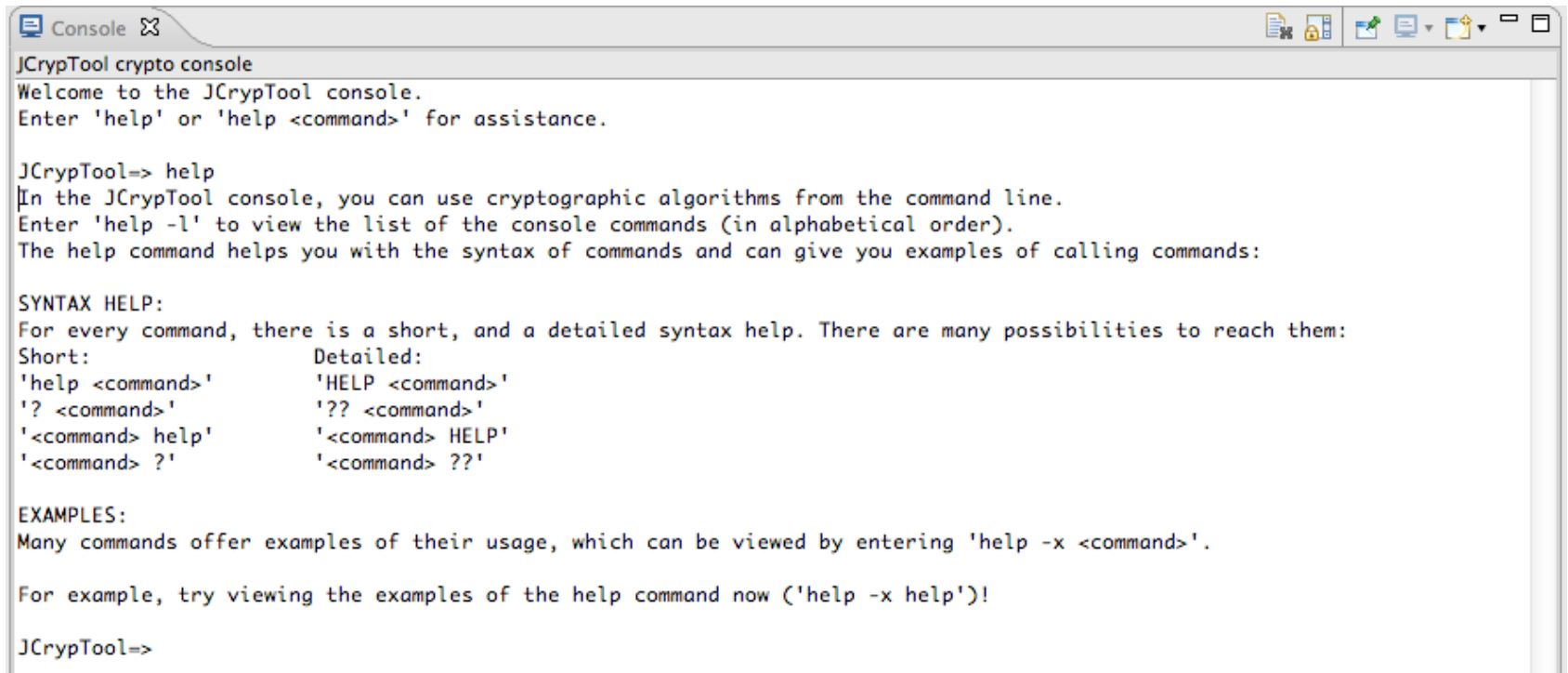
- New alphabets can be created quickly in JCT using existing building blocks.
- As special characters can be included as well, there are no limits for the usage of alphabets.
- As it is easy to understand, and efficient to build a custom alphabet, a user is motivated to try out.
- Most of the common crypto tools use fixed sets of alphabets for classic ciphers. At this juncture, JCT is maximally flexible.

# JCryptTool console for classic methods

## The implementation in JCT

### The console

- The classic cryptographic methods can be started from the console as well.



```
Console [X]
JCryptTool crypto console
Welcome to the JCrypTool console.
Enter 'help' or 'help <command>' for assistance.

JCryptTool=> help
[In the JCrypTool console, you can use cryptographic algorithms from the command line.
Enter 'help -l' to view the list of the console commands (in alphabetical order).
The help command helps you with the syntax of commands and can give you examples of calling commands:

SYNTAX HELP:
For every command, there is a short, and a detailed syntax help. There are many possibilities to reach them:
Short:           Detailed:
'help <command>'  'HELP <command>'
'? <command>'    '?? <command>'
'<command> help'  '<command> HELP'
'<command> ?'    '<command> ??'

EXAMPLES:
Many commands offer examples of their usage, which can be viewed by entering 'help -x <command>'.

For example, try viewing the examples of the help command now ('help -x help')!

JCryptTool=>
```


- To receive some additional information about the console, simply type the command “help”.
- There are help and example pages for each single method.

# JCryptTool console for classic methods

## Application sample 1/2



### Example Autokey Vigenère

- From the console, all classic cryptographic methods can be invoked on the current editor's content, a file on the disk or text as an argument in the console.
- The console can be called via the icon bar (below the main menu) via the following icon: 
- Example with the Autokey-Vigenère method:
  - Invoke help and examples:

```
JCryptTool=> help autovigenere
The Vigenère cipher, but the key is generated partly from the plaintext.
Syntax:autovigenere [-a <ALPHABET>] -D | -E -ed | -f <FILE_PATH> | -t <TEXT> -k <KEY> [-noFi]
Examples for this command are available under 'help -x autovigenere'.
For a more detailed help, enter 'HELP autovigenere'.
More information for this algorithm is available in the JCrypTool online help.

JCrypTool=> help -x autovigenere
'autovigenere -E -ed -k akey' -> Encrypts the active editor's text with the key "akey"
'autovigenere -D -ed -k akey' -> Decrypts the active editor's text with the key "akey"
'autovigenere -E -a A-Z -t "TEST TEXT" -k AKEY' -> Encrypts the text "TEST TEXT" with the key "AKEY", using only the uppercase alphabet
```

- The upper screen shot shows the command line options described in the console help for a special method (here using the example „HELP autovigenere“).

# JCrypTool console for classic methods

## Application sample 2/2



### Encryption and decryption with Autokey Vigenère

- As sample plaintext we use „ACTIONxCODExDAYYTT “, and as key we use „THEKEY“:

```
JCrypTool=> autovigenere -E -a a-zA-Z -t "ACTIONxCODExDAYYTT" -k THEKEY  
TJXSSlxEhLSKACmbXQ
```

- The 2<sup>nd</sup> row shows the generated ciphertext „TJXSSlxEhLSKACmbXQ “, generated by the command „autovigenere -E -a a-zA-Z -t "ACTIONxCODExDAYYTT" -k THEKEY“
- By substituting “-E” with “-D” in the command we can simply revoke the encryption:  
„autovigenere -D -a a-zA-Z -t "TJXSSlxEhLSKACmbXQ" -k THEKEY“

```
JCrypTool=> autovigenere -D -a a-zA-Z -t "TJXSSlxEhLSKACmbXQ" -k THEKEY  
ACTIONxCODExDAYYTT
```

# JCryptTool console for classic methods

## Educational objective 1/2

### Advantages using the console

- The parameters of an operation (such as the alphabet and the key) can be easily inserted and reused via Copy&Paste.
- The more parameter one uses, the more efficient the usage of the console is. There can be much more parameters than the alphabet, the key and the filtering of non-alphabet characters.
- For instance, the transposition encryption method uses a lot of parameters:
  - Each of the following parameters can be configured for the 1<sup>st</sup> and 2<sup>nd</sup> round (at all 6 parameters):
    - Direction of read in
    - Direction of read-out
    - Key
  - Alphabet
  - Filtering of characters not in the alphabet
- Once entered in the dialog window the command line contains all parameters.
- The command line can be copied, pasted and easily modified.

Transposition

Please enter transposition key(s), and in-/out-reading modes.

Operation

Encrypt  Decrypt

Alphabet

Plain-/Ciphertext alphabet:  Select alphabet: Anzeigbares ASCII  Custom alphabet...

Filter non-alphabet characters from the input text before the en-/decryption.

Transposition(s)

First transposition	Second (optional) transposition
1) Read the text into the transposition table... <input type="radio"/> Column-wise <input checked="" type="radio"/> Row-wise	1) Read the text into the transposition table... <input type="radio"/> Column-wise <input checked="" type="radio"/> Row-wise
2) Transposition - enter the key C4D3 3 2 4 1	2) Transposition - enter the key 3143 2 1 4 3
3) Read the ciphertext out of the transposition table... <input type="radio"/> Column-wise <input checked="" type="radio"/> Row-wise	3) Read the ciphertext out of the transposition table... <input checked="" type="radio"/> Column-wise <input type="radio"/> Row-wise

Appropriate command for the console:

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



# JCrypTool console for classic methods

## Educational objective 2/2



### Detailed help from the console

- Help on the console for the transposition method

```
JCrypTool=> HELP transposition
Transposes characters of the plain (columnar transposition with definable read-in / read-out directions).
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'>]
Option explanation:
-a,--currentAlphabet <ALPHABET>          One of ASCII, a-zA-Z, A-Z, a-z, Playfair, ADFGVX,
                                           Xor32, Xor64, default: ASCII
-D,--modeDecrypt                          Decryption
-E,--modeEncrypt                          Encryption (Default, if neither en- nor
                                           decryption is specified)
-ed,--editor                               Use active Editor as Input
-f,--inputFile <FILE_PATH>               File is input
-k,--key <KEY>                             Key (only characters from the selected alphabet
                                           are allowed)
-k2,--key2 <KEY>                          Optional second transposition key, which
                                           signalsizes that a double columnar transposition
                                           has to be executed.
-noFi,--noFilter                          Non-alphabetic characters will not be filtered
-t,--inputText <TEXT>                    Text as input (as string between "")
-t1ReadIn,--transposition1ReadInOrder <ORDER = 'cw'/'rw'> ORDER = 'cw' (column by column) / 'rw' (row by
                                           row). Read-in direction of plaintext into
                                           transposition table (if not defined,row-wise).
                                           (applies for the 1st transposition)
-t1ReadOut,--transposition1ReadOutOrder <ORDER = 'cw'/'rw'> see argument 't1ReadIn's description (if not
                                           defined,column-wise).
-t2ReadIn,--transposition2ReadInOrder <ORDER = 'cw'/'rw'> see argument 't1ReadIn's description (if not
                                           defined,row-wise).
-t2ReadOut,--transposition2ReadOutOrder <ORDER = 'cw'/'rw'> see argument 't1ReadIn's description (if not
                                           defined,column-wise).

Examples for this command are available under 'help -x transposition'.
Aliases for this command are 'transp'.
More information for this algorithm is available in the JCrypTool online help.
```

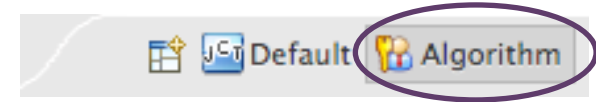
# The perspective “Algorithm”

## The implementation in JCT



### JCT perspectives

- JCT supports two main user interfaces: the Default Perspective and the Algorithm Perspective.
- The Algorithm Perspective is function oriented and comes along with more advanced settings.



The **Algorithm Perspective** is separated – next to the editor and the help – in the following 3 windows:

#### 1. Keystore

- Allows to save keys and key pairs for later usage.

#### 2. Algorithms

- An explorer for algorithms. The algorithms are provided by the crypto libraries FlexiProvider<sup>[1]</sup> and BouncyCastle<sup>[2]</sup>. In contrast to the Crypto Explorer in the Default Perspective, many different variants of the algorithms are directly listed and selectable. Altogether, the Algorithm Explorer is much more extensive than the Crypto Explorer.

#### 3. Operations

- The algorithm, chosen via double-click in the Algorithm Explorer, is listed here. Then additional settings (e.g. the source of the input, the target for the output, the key and the algorithm's parameter) are outlined here.

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

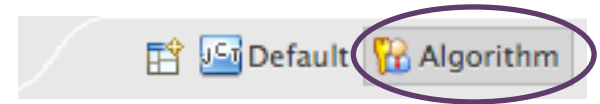
[2] <http://www.bouncycastle.org>

# The perspective “Algorithm”

## The introduction plugin

### Algorithm Perspective explained

- On opening the Algorithm Perspective for the first time, a slide show appears in the editor area which explains the basic functionality of that perspective.





A screenshot of the JCT software interface. The main editor area displays a tutorial slide titled "Tutorial for the Algorithm Perspective using RC6 encryption as an example." The slide is framed by a red border. The interface includes a menu bar (File, Edit, Window, Help), a toolbar, and several panels: "Keystore" (listing Alice Whitehat, Bob Whitehat, and [PW: 1234] Alice Whitehead), "Help" (with sections for Contents, Search, Related Topics, Bookmarks, and Index), "Operations" (showing "Current Entry: &lt;None&gt;"), and "Algorithms" (listing various cryptographic algorithms like Asymmetric Block Ciphers, Block Ciphers, Hybrid Ciphers, etc.). A "Do not show again" checkbox is visible at the bottom right of the tutorial slide.

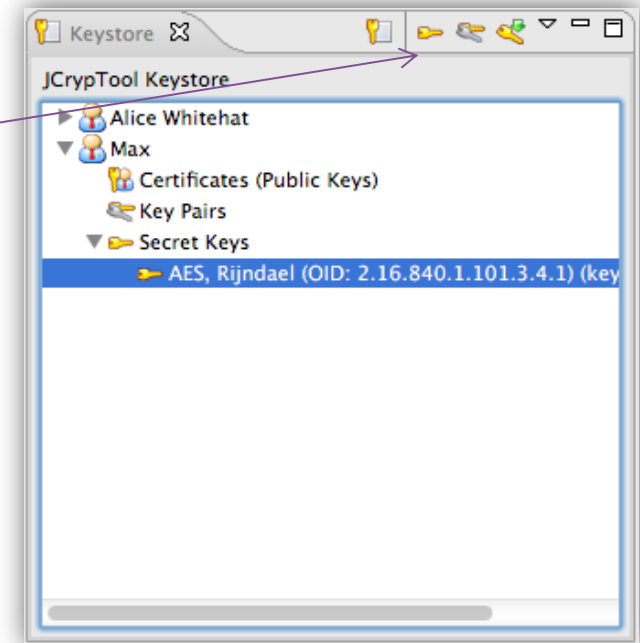
# The perspective “Algorithm”

## Application sample 1/3: select and customize the AES operation

In this example we encrypt a text from the opened editor with AES and export the result to a file.

### Generate a new key and assign it to a contact

- First, we generate a key to be used for encryption.
- With  a new symmetric key can be created. As AES is a symmetric crypto system, it does only need a single secret key (instead of a key pair). Alternatively, for asymmetric crypto systems the appropriate key pairs can be generated with . This is step **1** on slide **83**.
- In the wizard „New Symmetric Key“ we choose „AES, Rijndael (OID 2.16.840.1.101.3.4.1)“<sup>[1]</sup>, select or create our new contact by changing the contacts name and set an arbitrary password.
- Then, the key is stored in the JCT keystore, listed below the chosen contact name (in the example “Max”).



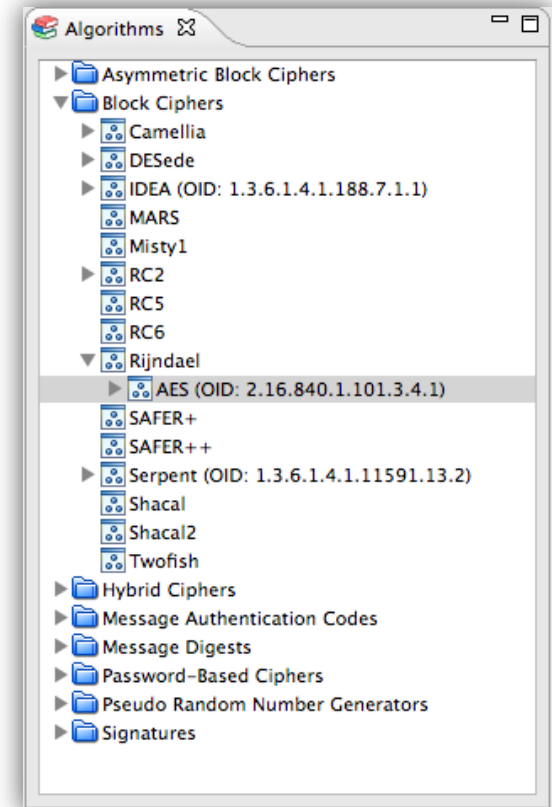
[1] OID: Object Identification, a unique identifier for an algorithm  
Defined by ITU ([http://en.wikipedia.org/wiki/Object\\_identifier](http://en.wikipedia.org/wiki/Object_identifier)).

# The perspective “Algorithm”

## Application sample 2/3: select and customize the AES operation

### Selection of the algorithm for the current operation

- Now, it’s time to choose the appropriate algorithm: In the tab “Algorithm“ below “Block Ciphers” you find the the AES-Rijndael algorithm.
- Select the algorithm with a **double click** (step **2** on slide 83).
- A wizard appears where padding and mode<sup>[1]</sup> of the block cipher can be adjusted. Additionally, further algorithm specific settings can be adjusted here (e.g. for AES the length of each block in bits).
- Like in the Crypto Explorer, the algorithms in this explorer are also grouped by the kind of the cryptographic method.
- Note about the usage:  
Step 2 (selection of the algorithm) can also be performed BEFORE step 1 (generation of a key for its owner in the keystore). So the order of step 1 and step 2 is independent. Especially if the contact (owner) already has a key for the chosen algorithm, you can directly start with step 2.



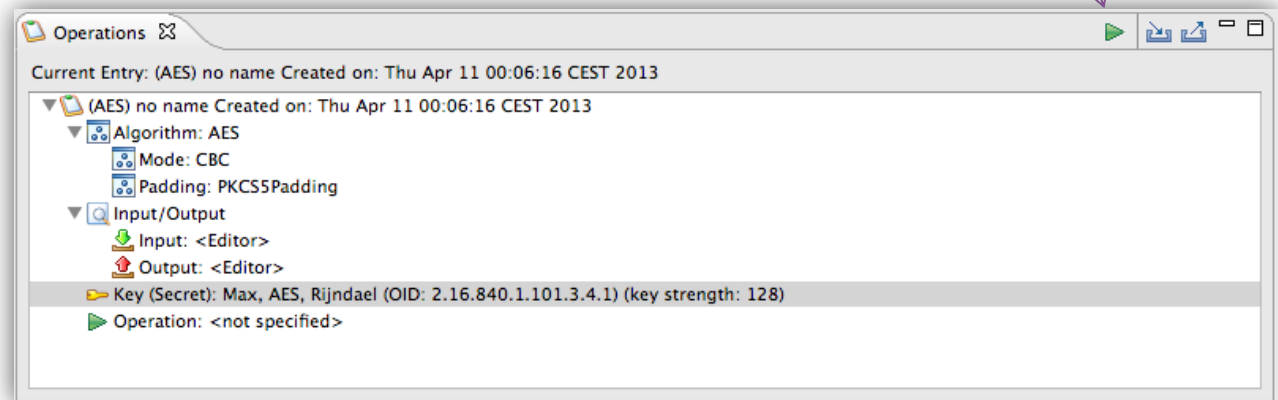
[1] The mode of a block cipher is responsible for the mapping of the plain text to the blocks, which will be eventually encrypted. If in the last block some bits are missing, the padding rules how these bits will be filled.

# The perspective “Algorithm”

## Application sample 3/3: select and customize the AES operation

### Settings for input and output of the current operation

- Via the double click (in the Algorithms tab on the right) Rijndael was added to the Operations tab.
- Via drag’n’drop the generated key (from the JCT keystore on the left) can now be copied on the key field of the algorithm (see slide 83, step 3 ).
- The option Input/Output offers to select via double click the source and target for the algorithm. E.g., performing a double click on Input, you can switch text input from a file or from an active editor window (see slide 83, step 4 ).
- To toggle between encryption or decryption, you can adjust the option “Operation”.
- After setting all the parameters you can start the operation by clicking the green arrow within the title of the Operations window.
- With asymmetric algorithms, the kind of operation (encrypt/decrypt, sign/verify) is dictated by the type of key. A public key encrypts or verifies, a private key decrypts or signs.





# The perspective “Algorithm”

## Overview of the 4 steps to select and customize an operation

The screenshot displays the JCryptTool application interface with four numbered steps overlaid:

- 1**: A callout box labeled "1" points to the "JCryptTool Keystore" tree view in the left sidebar, specifically to the "AES, Rijndael (OID: 2.16.840.1.101.3.4.1) (key)" entry.
- 2**: A callout box labeled "2" with the text "Double click" points to the same "AES, Rijndael" entry in the "Algorithms" tree view on the right.
- 3**: A callout box labeled "3" with the text "Drag'n'Drop" points to the "AES, Rijndael" entry in the "Keystore" view, with an arrow indicating the drag action towards the "Operations" view.
- 4**: A callout box labeled "4" points to the "Key (Secret): Max, AES, Rijndael (OID: 2.16.840.1.101.3.4.1) (key strength: 128)" entry in the "Operations" view at the bottom.

The "Operations" view shows the following configuration:

- Current Entry: (AES) no name Created on: Sun Sep 01 20:25:39 CEST 2013
- Algorithm: AES
- Input/Output
  - Input: <Editor>
  - Output: /Users/johannesspath/secretfile
- Key (Secret): Max, AES, Rijndael (OID: 2.16.840.1.101.3.4.1) (key strength: 128)
- Operation: Encrypt

# The perspective “Algorithm”

Result after executing the operation (here both, input and output, are in the JCT editor)

The screenshot displays the JCryptTool interface with the following components:

- Keystore:** Lists keys and certificates, including Alice Whitehat, Bob Whitehat, Erika Mustermann, JCT-PKI Root Certificates, and e1 Mustermann. A specific key is highlighted: AES, Rijndael (OID: 2.16.840.1.101.3.4.1) (key strength: 128).
- Operations:** Shows the current entry: (Rijndael) no name Created on: Wed Apr 16 15:11:45 CEST 2014. The operation is configured as follows:
  - Algorithm: Rijndael
  - Mode: CBC
  - Padding: PKCS5Padding
  - Input/Output: Input: <Editor>, Output: <Editor>
  - Key (Secret): Alice Whitehat, AES, Rijndael (OID: 2.16.840.1.101.3.4.1) (key strength: 128)
  - Operation: Encrypt
- Algorithms:** A list of cryptographic algorithms, with AES (OID: 2.16.840.1.101.3.4.1) selected.
- Hex Editor:** Displays the output of the encryption operation as a hex string: 59 30 61 E5 87 D7 4A 91 89 C5 38 41 70 2D 1D CD...



# Further functions in JCrypTool



## Further samples what is in JCrypTool

- Tri-partite key agreements (MPKE)
- Visualization of the inner states of DES
- Visualization of calculations on elliptic curves over real and discrete fields
- ElGamal Cryptosystem (for encryption and signing)
- Visualization of the simple power analysis attack against RSA (SPA)
- Quick solver of the number shark game with heuristic methods; solving of Sudoku variants
- Mathematical games: number shark, divider game, zero-knowledge Sudoku (Zudo-Ku)
- Entropy analysis
- Dynamic visualization of Huffman coding trees
- Signature demonstration, signature and certificate verification (shows effect of validity models)
- Visualization of the SSL/TLS handshake (protocol)
- Implementation and visualization of ARC4 and Spritz
- Visualization of post-quantum signature algorithms [ **SPHINCS+**, MerkleTree XMSS-MT, WOTS, McEliece (error-correcting code), multivariate cryptography (rainbow signature scheme) ]
- Fast cryptanalysis of the grille cipher
- ...

# Overview about all functions in JCrypTool

Visible within JCT. Alternatives: Listed within the CrypTool Portal or with the JCT admin tool

The screenshot shows the CrypTool Portal interface. At the top, there's a navigation bar with the logo and 'CrypTool Portal' text. Below it, a 'Did you know?' box provides background information. A central section titled 'Cryptological functions in different CrypTool versions' contains search filters: 'Cryptographic category' (No filter applied), 'Additional search phrase' (Search...), and version checkboxes (CrypTool 1, 2, JCT, Online). Below the filters, a message states '111 of a total 310 function groups match the selection criteria.' A table lists functions with their JCT and JCT Path.

Function	JCT	JCT Path
ADFGX / ADFGVX	D	[D] \ Algorithms \ Classic \ ADFGVX
AES	A/D	[A] \ Block Ciphers \ Rijndael \ AES \ AES128_CBC [A] \ Block Ciphers \ Rijndael \ AES \ AES128_CFB [A] \ Block Ciphers \ Rijndael \ AES \ AES128_ECB [A] \ Block Ciphers \ Rijndael \ AES \ AES128_OFB [A] \ Block Ciphers \ Rijndael \ AES \ AES192_CBC [A] \ Block Ciphers \ Rijndael \ AES \ AES192_CFB [A] \ Block Ciphers \ Rijndael \ AES \ AES192_ECB [A] \ Block Ciphers \ Rijndael \ AES \ AES192_OFB [A] \ Block Ciphers \ Rijndael \ AES \ AES256_CBC [A] \ Block Ciphers \ Rijndael \ AES \ AES256_CFB [A] \ Block Ciphers \ Rijndael \ AES \ AES256_ECB [A] \ Block Ciphers \ Rijndael \ AES \ AES256_OFB [D] \ Algorithms \ Symmetric \ AES
AES MAC	A	[A] \ Message Authentication Codes \ CBCMac \ CBCmacAES128 [A] \ Message Authentication Codes \ CBCMac \ CBCmacAES192 [A] \ Message Authentication Codes \ CBCMac \ CBCmacAES256 [A] \ Message Authentication Codes \ CMac \ CmacAES128 [A] \ Message Authentication Codes \ CMac \ CmacAES192 [A] \ Message Authentication Codes \ CMac \ CmacAES256

<https://www.cryptool.org/en/documentation/functionvolume>

Legend:

[A] in Algorithm Perspective

[D] in Default Perspective



Introduction to the e-learning software JCrypTool

2

Applications within JCT – a selection

22

**How to participate**

87

# How to participate – Overview



---

JCrypTool – Request for participation

Page 89

---

Participation in JCrypTool

Page 90

---

Contacts

Page 92

---



## Arms are wide open for your participation

- Feedback, critique, helpful suggestions and ideas
- Implementation of more algorithms, protocols or techniques for analysis
- Help to ensure consistence and completeness
- Participation in the development (programming, layouting, translation, tests, website development)
  - in the “old” C/C++ project CryptTool 1 and
  - in the new projects (preferred):
    - C# project: „CryptTool 2“ = CT2 (<https://www.cryptool.org/de/ct2/volunteer>)
    - Java project: „JCrypTool“ = JCT (<https://www.cryptool.org/en/jct/volunteer>)
    - Browser project: „CryptTool-Online“ = CTO (<http://www.cryptool-online.org>)
- Especially faculties/chairs who use JCrypTool for educational purposes, are invited to join the development.
- Significant contribution can be mentioned (in the online help, in about dialogs, and on the website).

# Participation in JCrypTool



## Example ideas for more visuals

- Visualization of the interoperability between S/MIME and OpenPGP formats
- Demonstration of visual cryptography
- Protocol validator
- Cryptanalysis of further algorithms
- Visualization of different methods from post-quantum cryptography
- Visualization of current developments like indistinguishability obfuscation

## Further things of high interest

- One place for all the manipulations of frequency tables (creation, exchange, deepness) and of permutations
- Key Storage
- JavaFX support

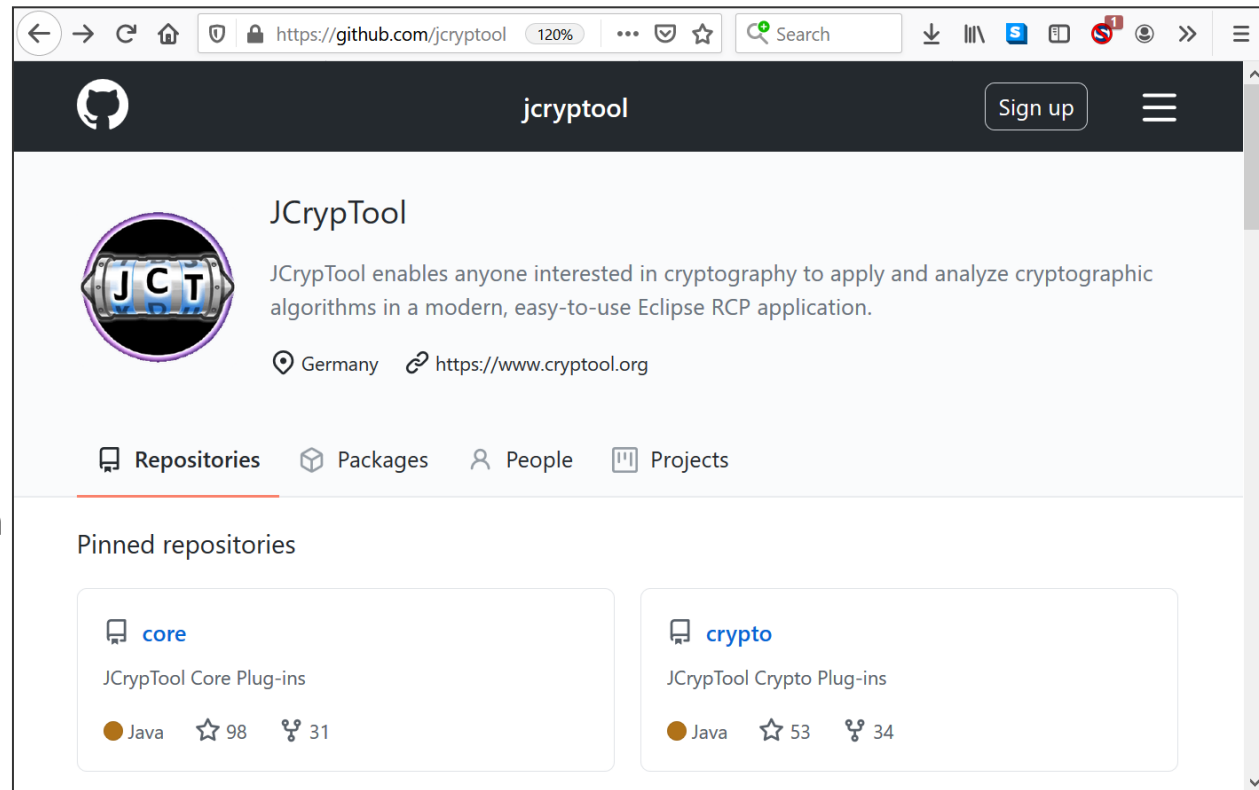
## Open tasks are also mentioned on the developer sites:

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

# Participation in JCrypTool

## More information for developers

- 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>
- Information for developing plugins is provided in the JCT wiki. The wiki in the internet offers links and information for JCT core developers and crypto plugin developers.
- Plugin developers should not need any projects from the JCT repository. They just need to run JCT as a target platform and develop for it.



The screenshot shows the GitHub profile page for the organization 'jcryptool'. The page header includes the GitHub logo, the organization name 'jcryptool', and a 'Sign up' button. The profile picture is a circular logo with 'JCT' and 'CRYPTO' text. The bio states: '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'. Below the bio are navigation tabs for 'Repositories', 'Packages', 'People', and 'Projects'. The 'Pinned repositories' section displays two repositories: '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.



<b>Prof. Bernhard Esslinger</b>	<b>Simon Leischnig</b>	<b>Thorben Groos</b>
Overall CT lead University of Siegen	JCT project lead	JCT project co-lead
<a href="mailto:bernhard.esslinger@uni-siegen.de">bernhard.esslinger@uni-siegen.de</a> <a href="mailto:bernhard.esslinger@gmail.com">bernhard.esslinger@gmail.com</a>	<a href="mailto:simonjena@gmail.com">simonjena@gmail.com</a>	<a href="mailto:thorben.groos@web.de">thorben.groos@web.de</a>

**Dominik Schadow:** former project lead, [info@xml-sicherheit.de](mailto:info@xml-sicherheit.de)

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