Cryptology with JCrypTool (JCT)

Practical introduction to cryptography and cryptanalysis

Prof Bernhard Esslinger
and the CrypTool team

February 16th, 2015

Agenda

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 machines. Its modern pure-plugin architecture offers dynamically reloading of plugins.
- JCT is developed within the open-source project CrypTool (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
  - Students
  - Teachers and professors
  - People with interests in cryptology
- As being Open Source and developed in Java, everyone is capable of implementing his own plugins and tools. Already developed components can be easily reused.

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

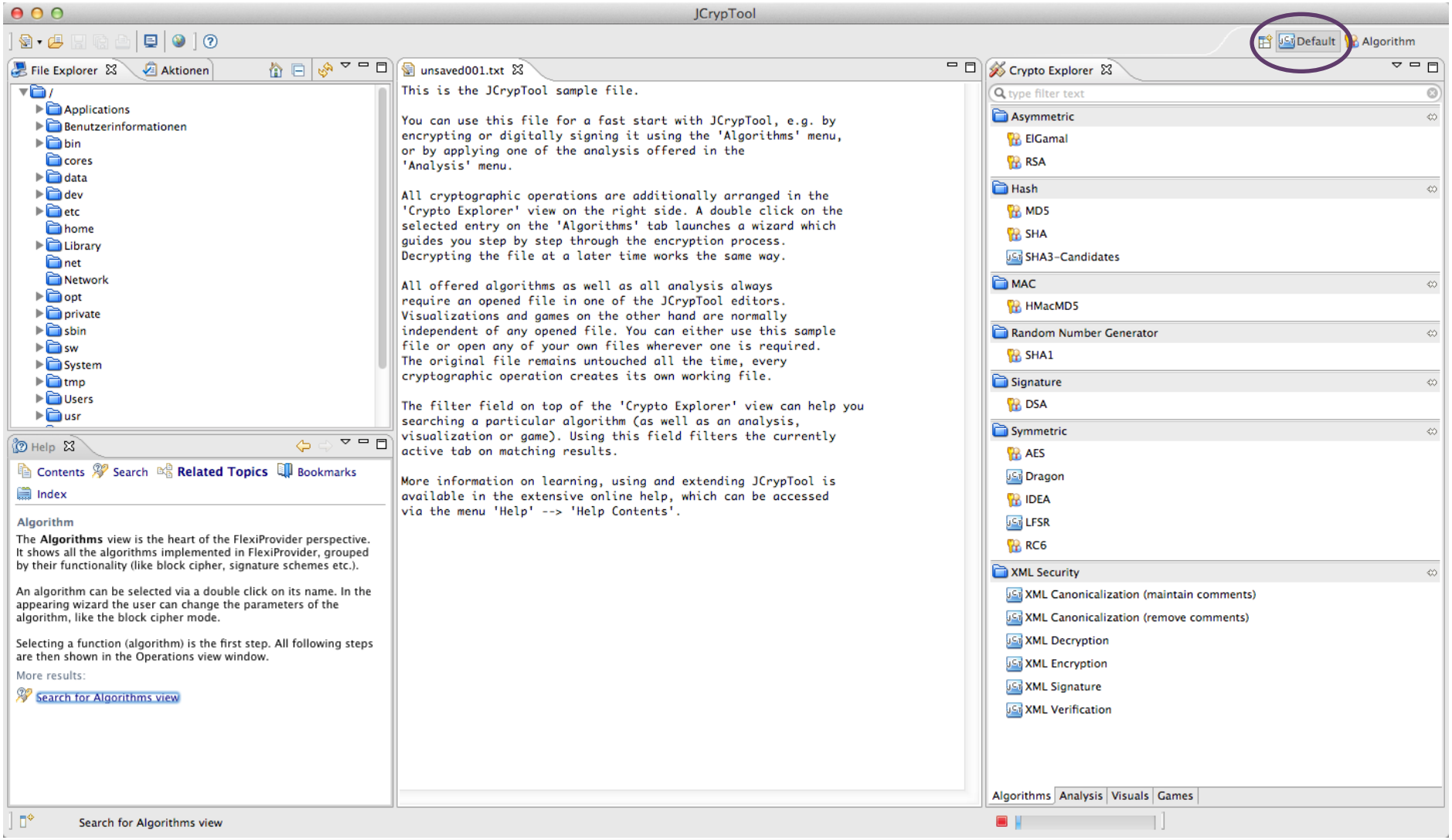
- Science about encryption systems guaranteeing secure and confidential storage and exchange of information (e.g. between computers).
- Nowadays, next to encryption important tasks are a secure exchange of the encryption keys and the integrity check, e.g. for online banking, for electronic elections, 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).

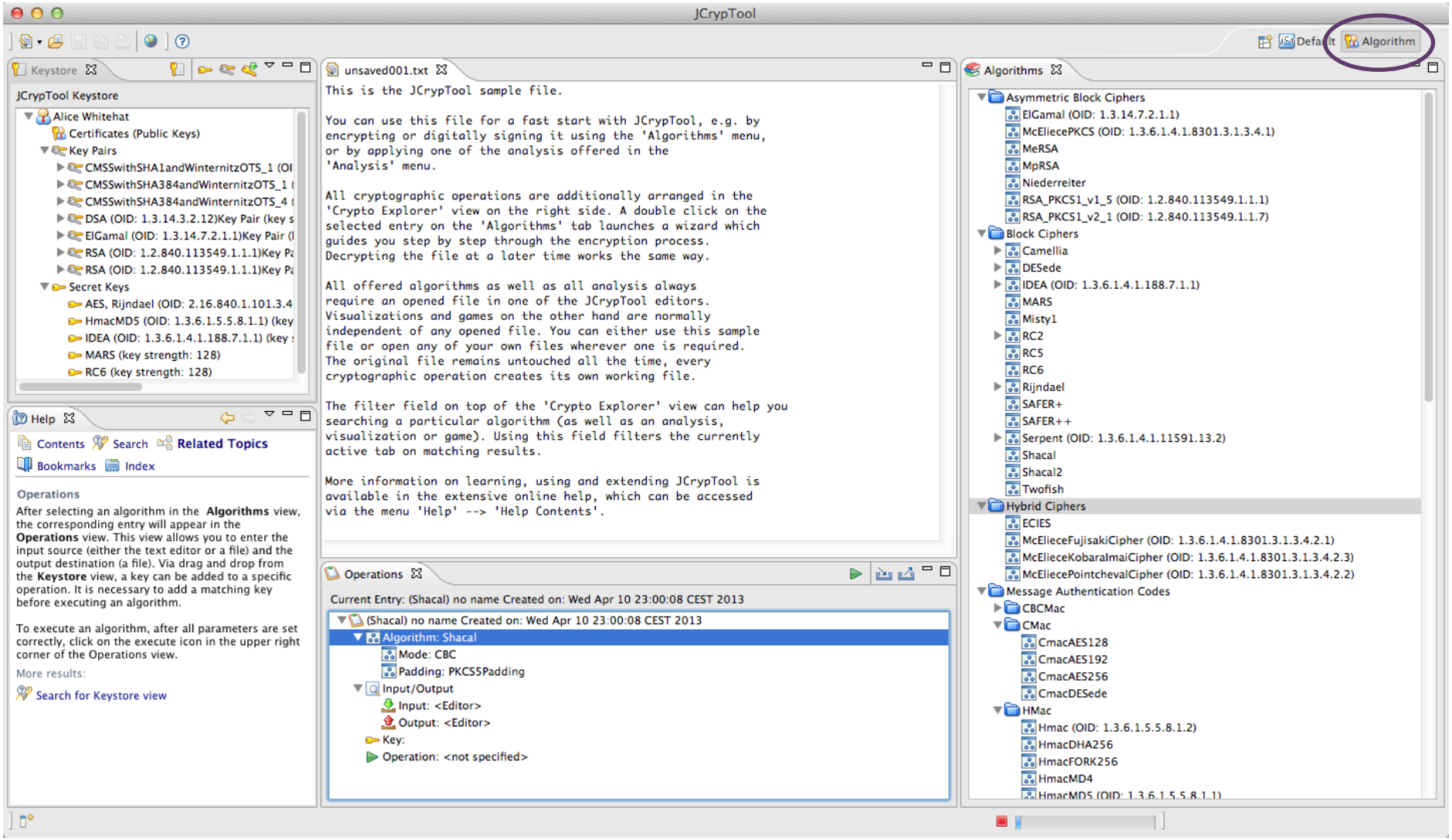
The Default perspective of JCT

... focuses on documents (document-oriented)



The Algorithm perspective of JCT

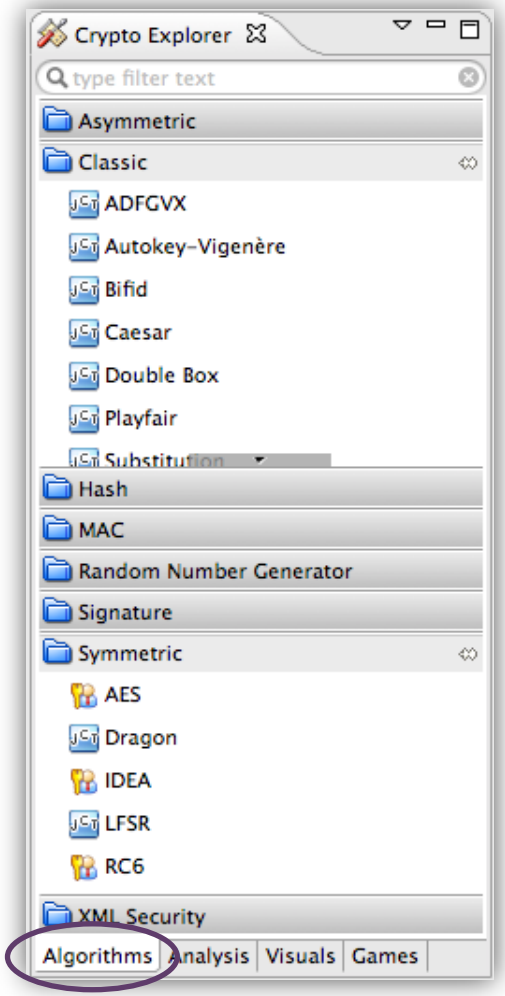
... focuses on functions (function-oriented)



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

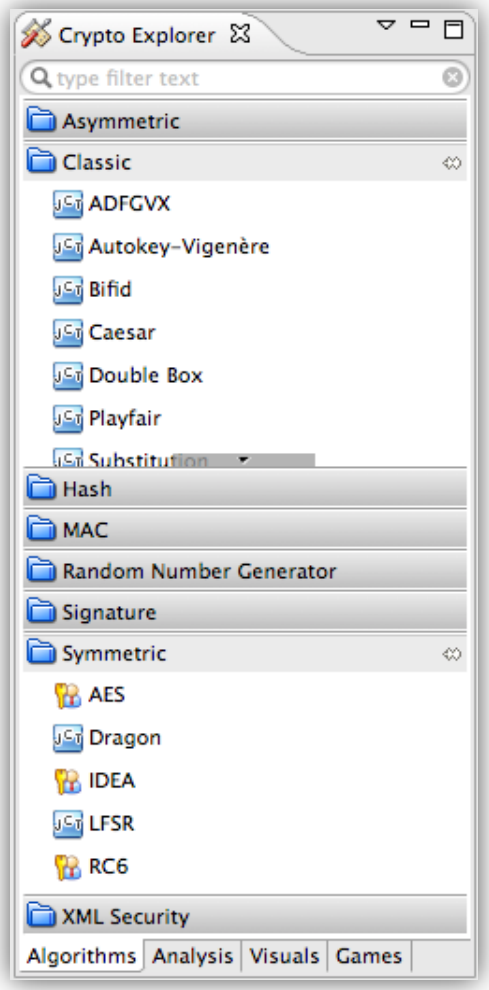
- This category gathers methods, which were used to encrypt messages roughly until World War I. Many of them are breakable by analyzing frequencies. Most of these methods are nowadays insecure.

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 **en**crypt his message with the public key of the receiver, while only the receiver can **de**crypt the message with his own private key.



Algorithms in the Crypto Explorer view

Clustering 2/2

Hash & MAC

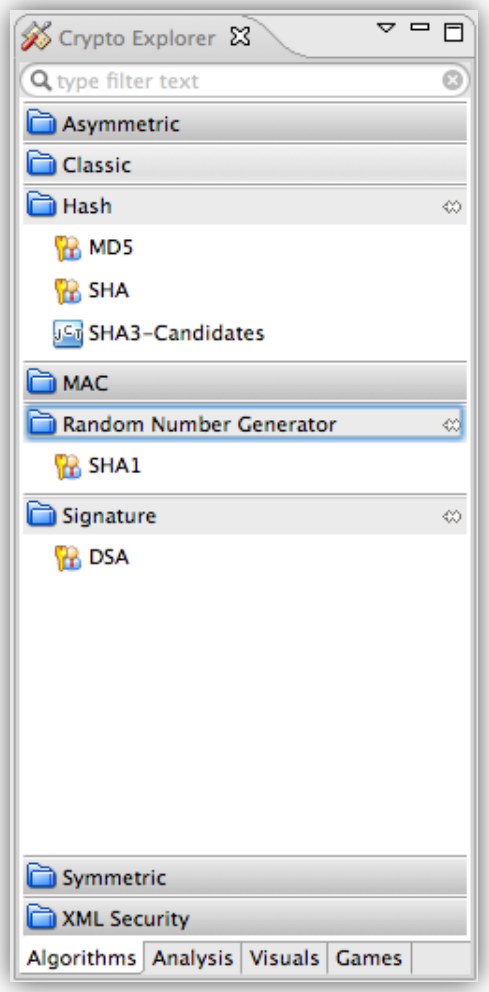
- Hash functions map data of arbitrary length to a hash value. This hash value is associated to the data in a preferably unique way and has a fixed bounded length which is normally much smaller than the length of the referred data (comparable to a fingerprint).
- Hash values are used to check for changes 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

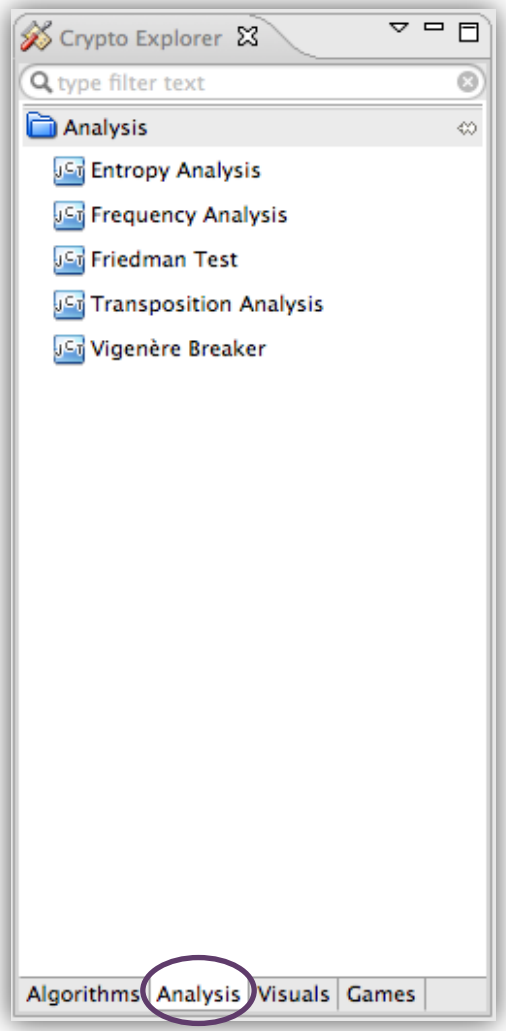
- In cryptography random numbers play a major role. Therefore functions for generating (pseudo-random) sequences of numbers are implemented in JCT as well.



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.

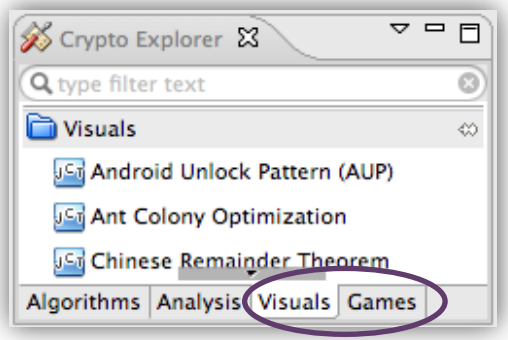


Visuals & Games

In the Crypto Explorer

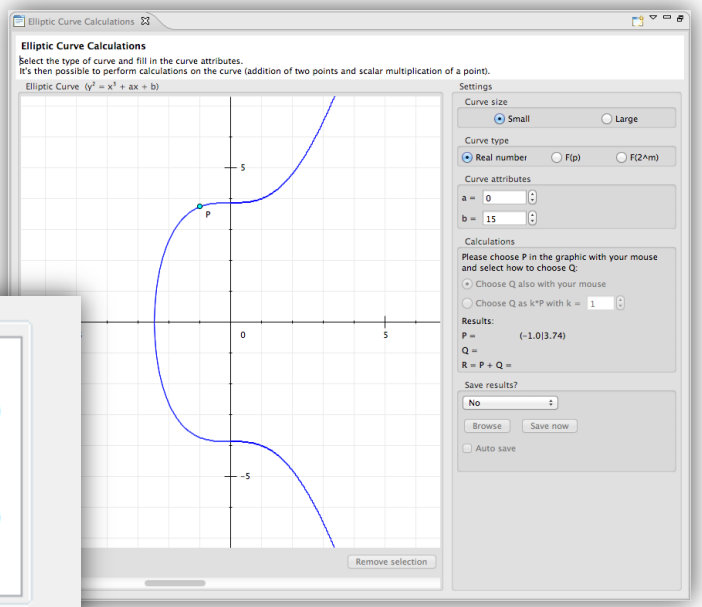
Visuals

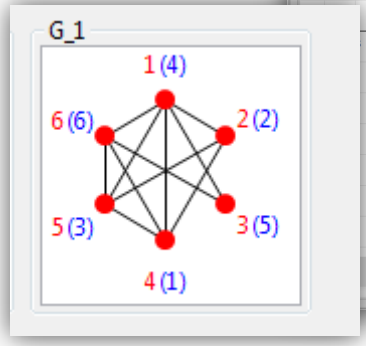
- Visuals can be found in the tab “Visuals” in the Crypto Explorer or in the menu “Visuals”.
- More than 20 visuals of cryptographic problems, circumstances and algorithms shall help the user to understand cryptography in a descriptive and playful way.
- To understand cryptology, a basic knowledge of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

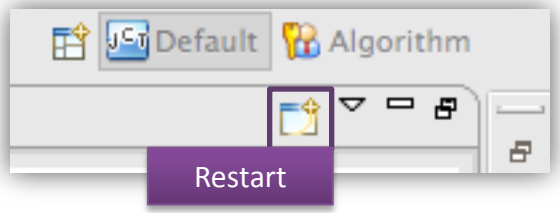




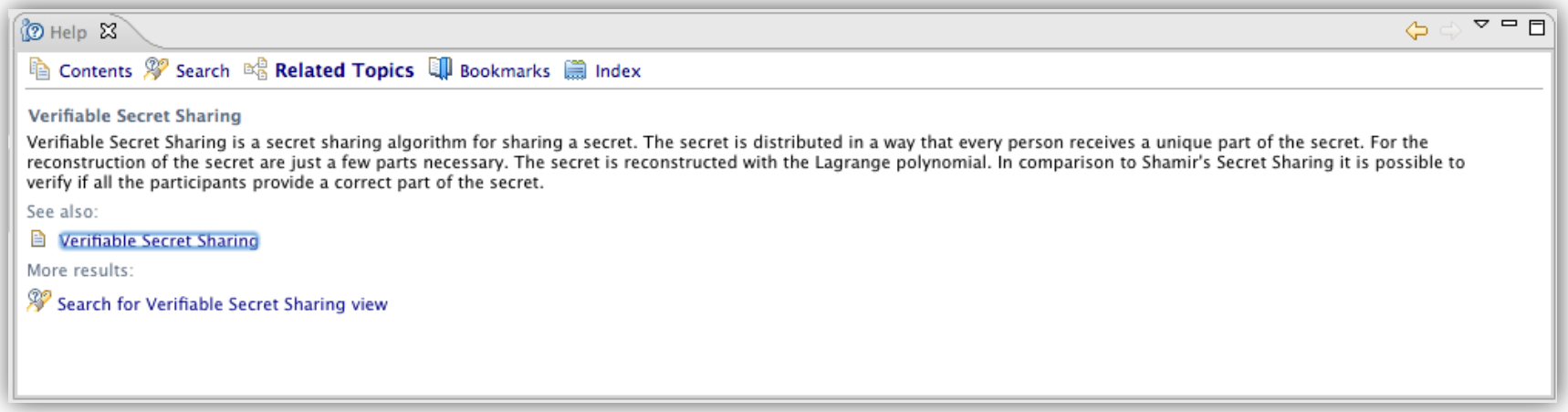
General operation instructions
...and more 1/2

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.



- The function key **F1** can be used to open the context-sensitive help at any time. The appropriate help page which belongs to the current view will be loaded automatically. In the help detailed information can be quickly found.

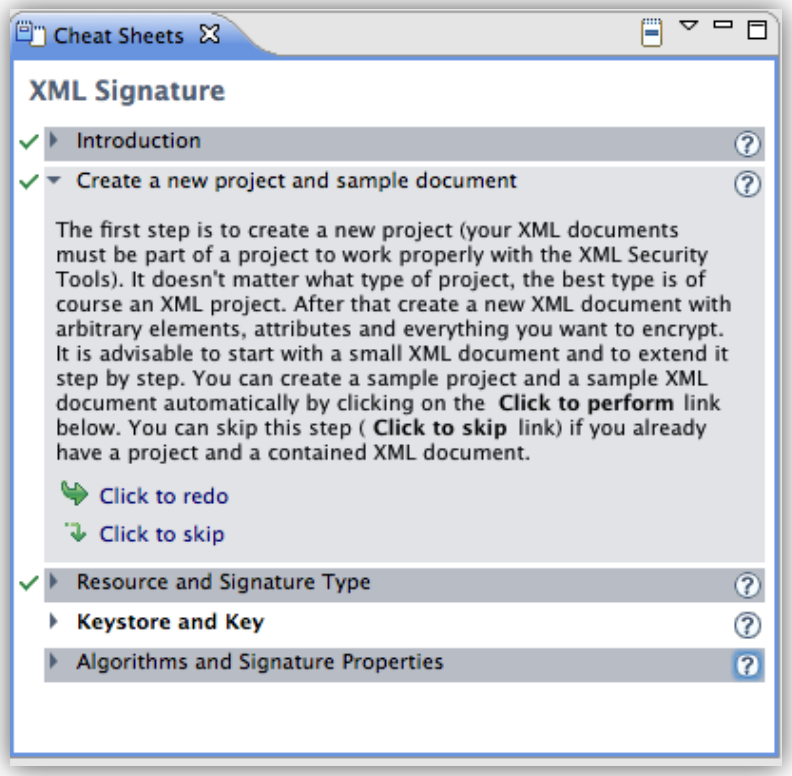


Help for the plugin „Verifiable Secret Sharing“

General operation instructions
...and more 2/2

Tips and tricks

- In the menu „Help“ > „Cheat Sheets“ **cheat sheets** can be activated. They are short introductions and tutorials which are integrated into the JCT view. So the steps and hints can be easily redone in JCT.
- 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. The tabs which are contained in an area are then 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 hid again.

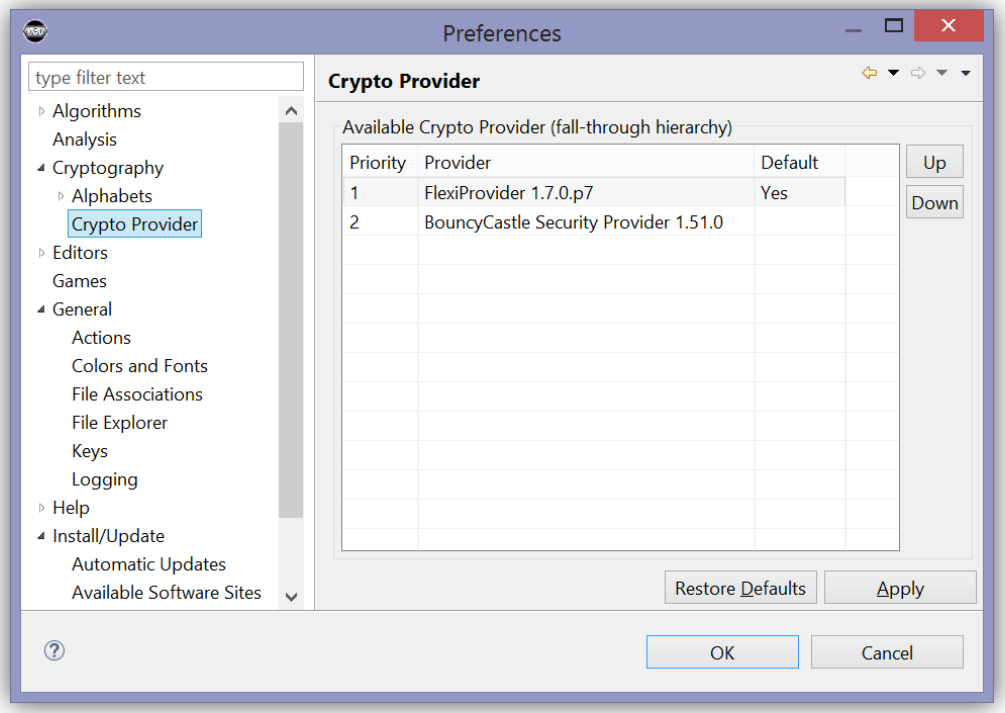


Sample of a cheat sheet

User settings
...the global preferences of JCT

More settings

- The global settings of JCT can be found in the preferences: See the menu paths: under Windows: „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.
- **Crypto Provider**: A list of the crypto libraries included in JCT. The priority defines the order in which an algorithm is chosen from the libraries, if multiple implementations exist.
- **Keystore**: Manage the files in which keys of the JCT keystore are saved.
  A newly generated keystore can then be used in the perspective “Algorithm”.

Agenda

JCT in application
Overview

The Ant Colony optimization (ACO)

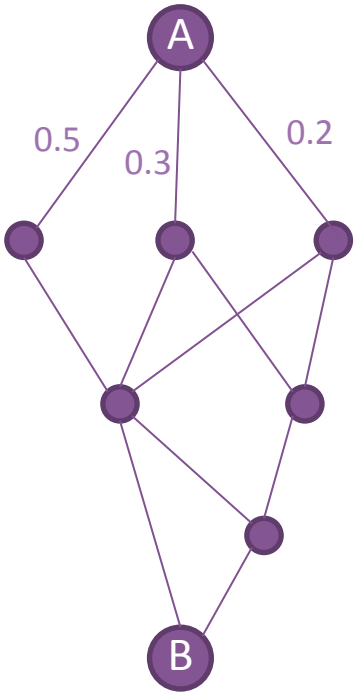
The idea

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

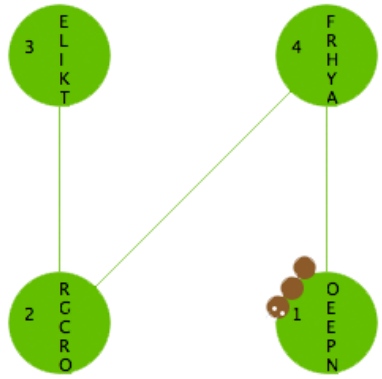
- Ciphertexts encrypted with a simple column transposition cipher can be decrypted with the ant colony optimization.
- To do so, the key length *n* is needed and the ciphertext is written row wise in *n* columns. These columns will then be arranged as a graph.
- Different pairs of characters arise by concatenating the columns in different orders. In each language, theses combination of characters appear with differing frequencies. Weights on edges in the graph are calculated based on theses 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 theses 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
...	...	...	...

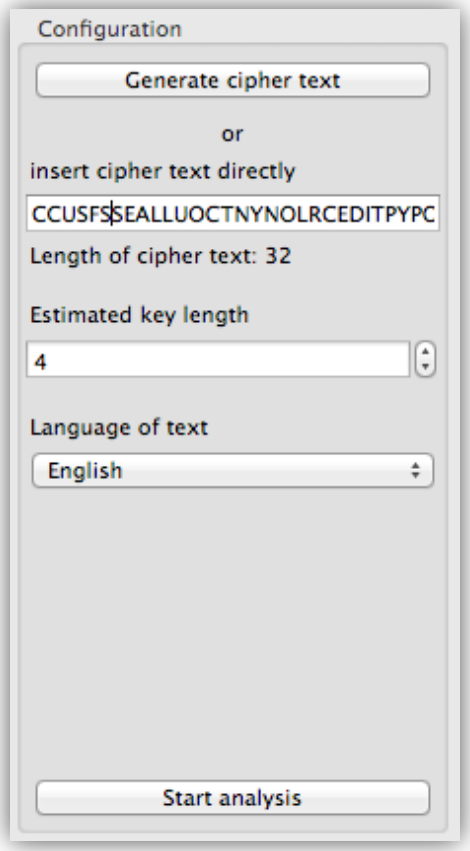


The Ant Colony optimization
Application sample 1/2

Try to decrypt the following text:

CCUSFSSEALLUOCTNYNOLRCEDITPYPONO

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



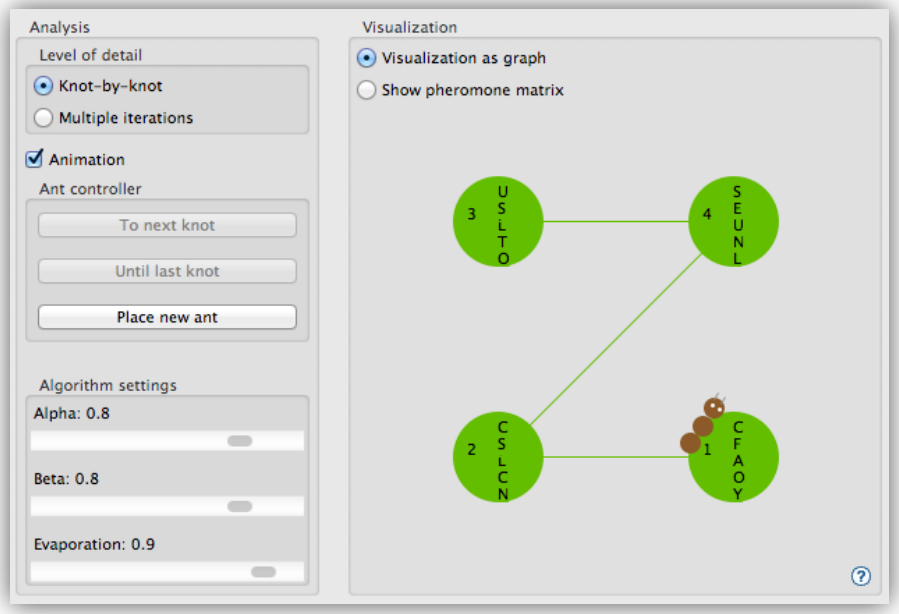
[1] The length of key can be estimated with statistic evaluations.
Additionally, here the length of the ciphertext has to be a multiple of the length of the key.

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:

- Theses 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.

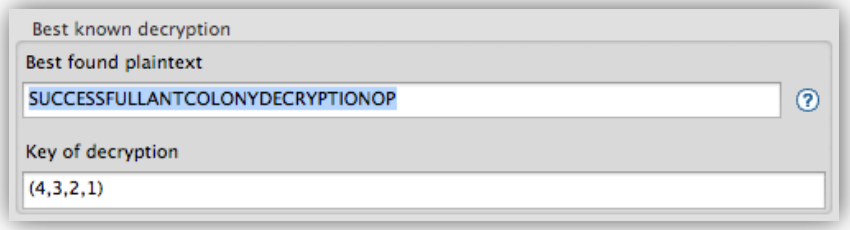
The Ant Colony optimization

Educational objective

Result

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

SUCCESSFULLANTCOLONYDECRYPTIONOP[2]



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 of 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

Viterbi analysis
The implementation in JCT

In the menu

„Visuals“ \ „Viterbi“

The algorithm in the cryptanalytical application

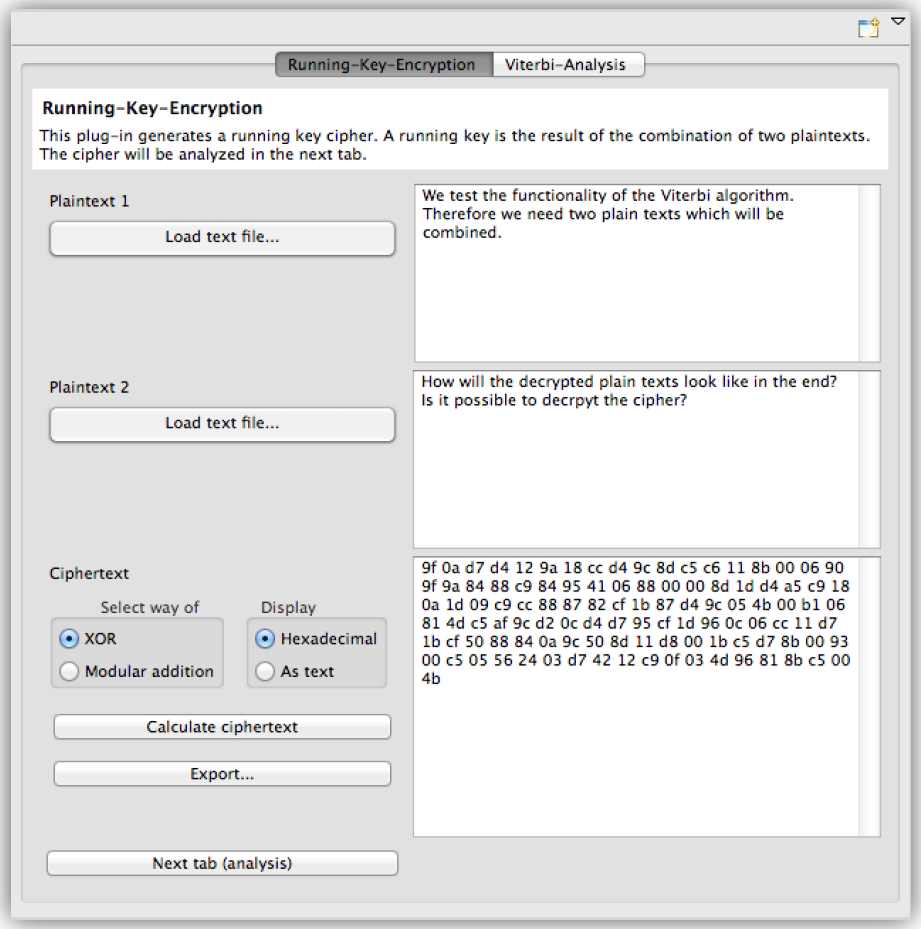
- The basic concept of the algorithm is statistical evaluation of the probability of N-grams combined with the usage of a dictionary of the language in which the ciphertext is written.
- The model of the the analysis is set up with the knowledge that the ciphertext was originally constructed by modular 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 path 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)”.

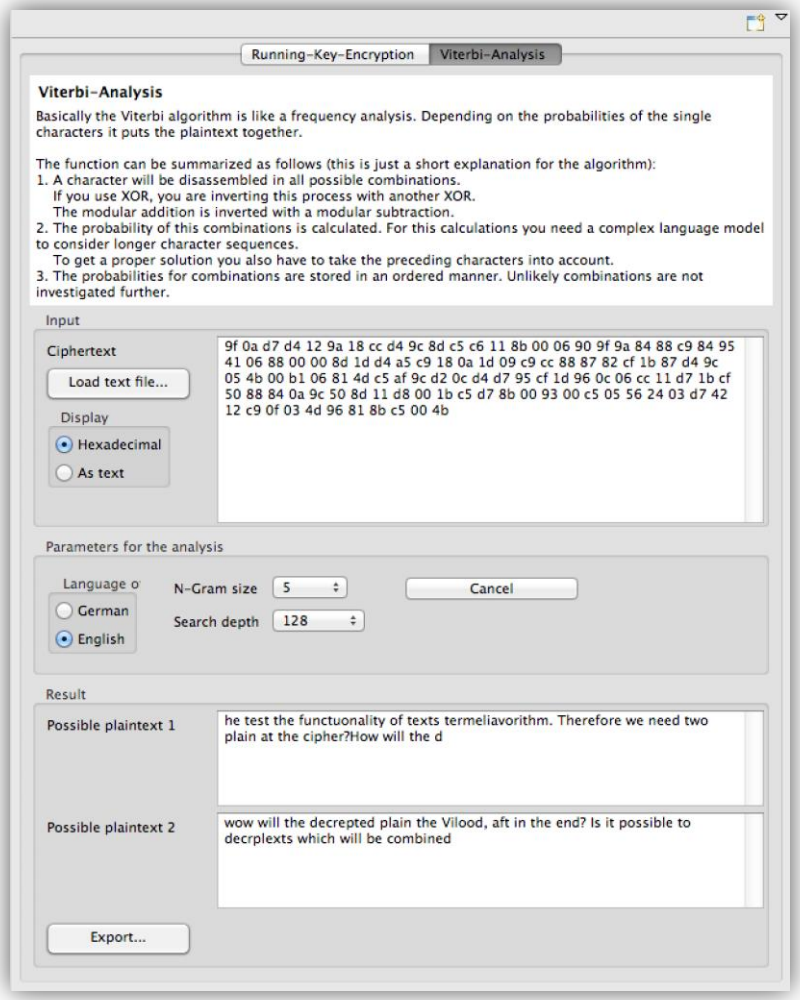


Viterbi analysis
Application sample 2/2

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?



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

- The 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“.
- Moreover it is important that a minimal 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 could easily modify the shares and make the shares be useless.
- To resolve this problem, the dealer hands out “commitments“ to each player. With a commitment, each player is able to test his share whether it 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 <= 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

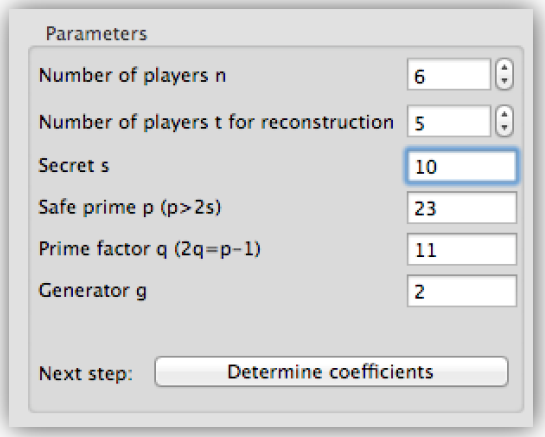
Application sample 1/2

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“.



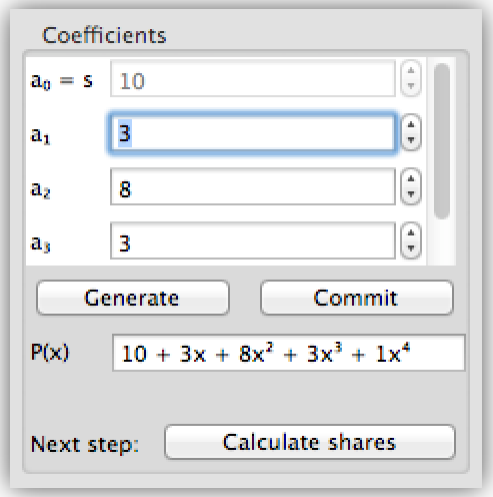
Second step

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”.



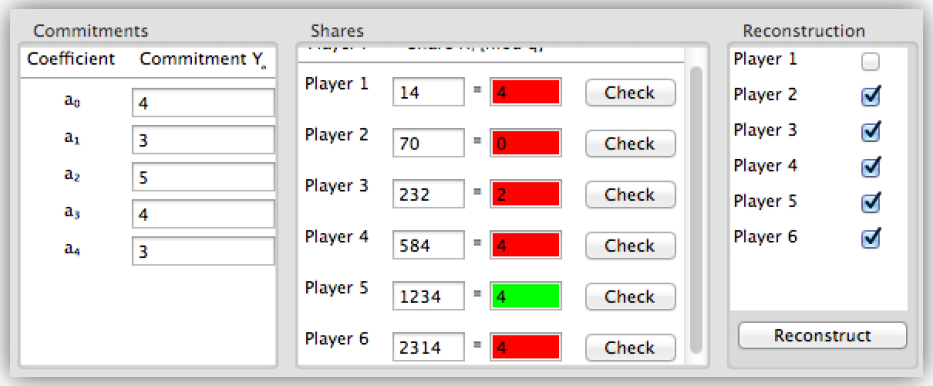
Verifiable Secret Sharing

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.



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.

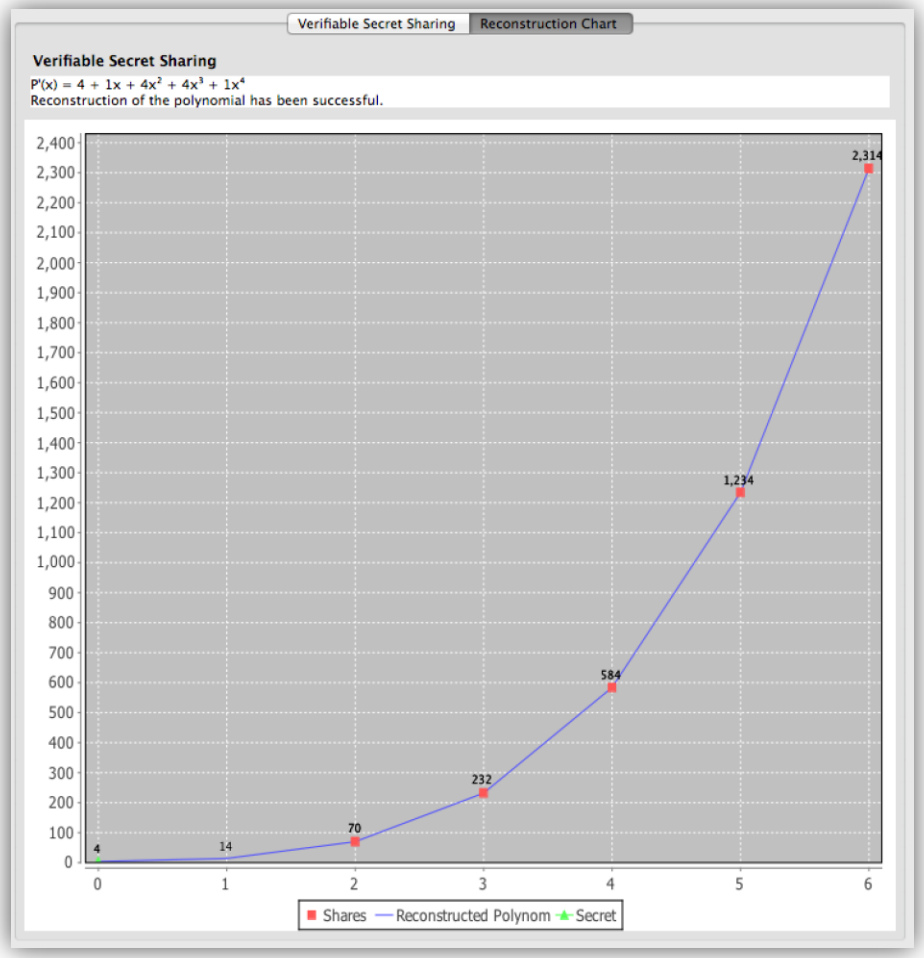
- To reconstruct the secret, the players whose shares shall be used can be selected (see screenshot on the right).
- For our example we selected five players which is as well the minimal number of shares needed. As we have t = 5.
- Now the secret can be reconstructed by clicking on “Reconstruct” (invalid 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 35).
- 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.

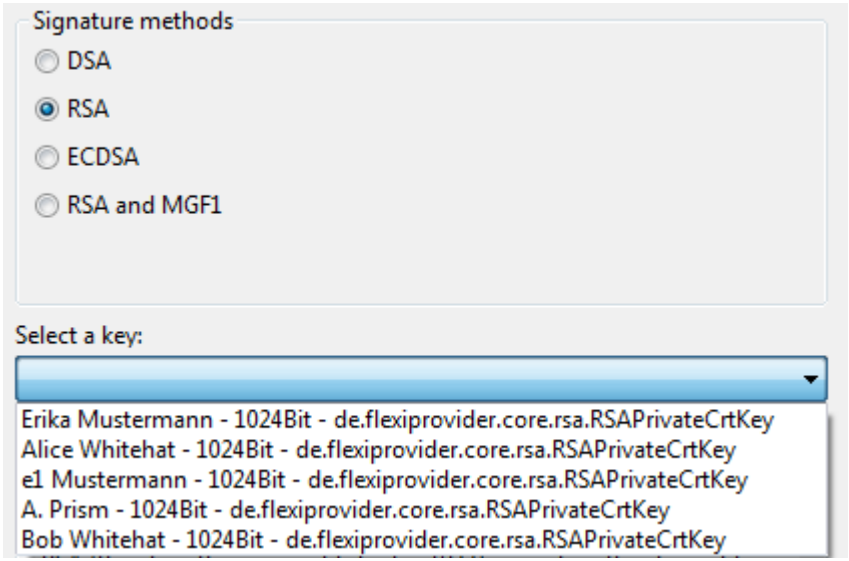
Signature demonstration
The implementation in JCT

In the menu

“Visuals“ \ “Signature Demonstration“

The algorithm applied

- The plugin is capable of digitally sign a document from 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 MFG1 as signature method.
- Below that you can choose the subject (key owner) who owns an according key for the chosen signature method.*



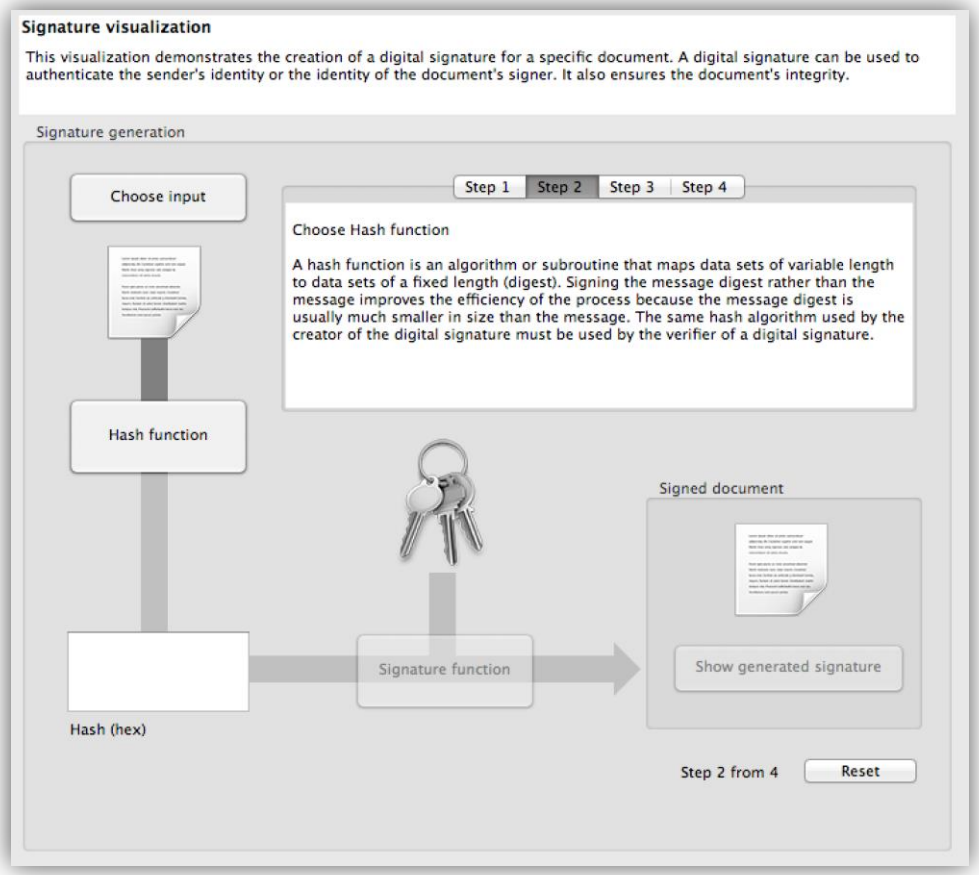
* There are two ways to create according keys for the subjects (key owners, users):
  a) Within the algorithm perspective.
  b) With the visualization plugin “Public-Key Infrastructure” (JCT-PKI).

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

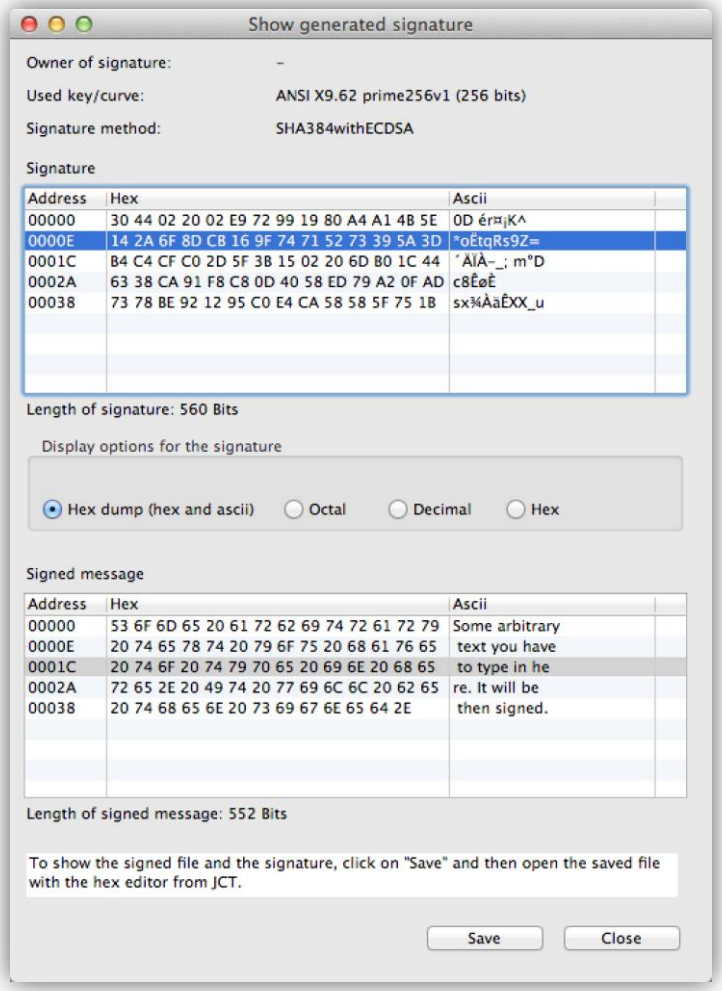
- Via “Choose input” the document to be signed can be chosen.
- A dialog appears and either a text can be typed in directly or an arbitrary document can be opened with “From file”.
- Next a “Hash function” must be selected.
- The hash value is then be generated and is shown in the lower left area.
 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



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“

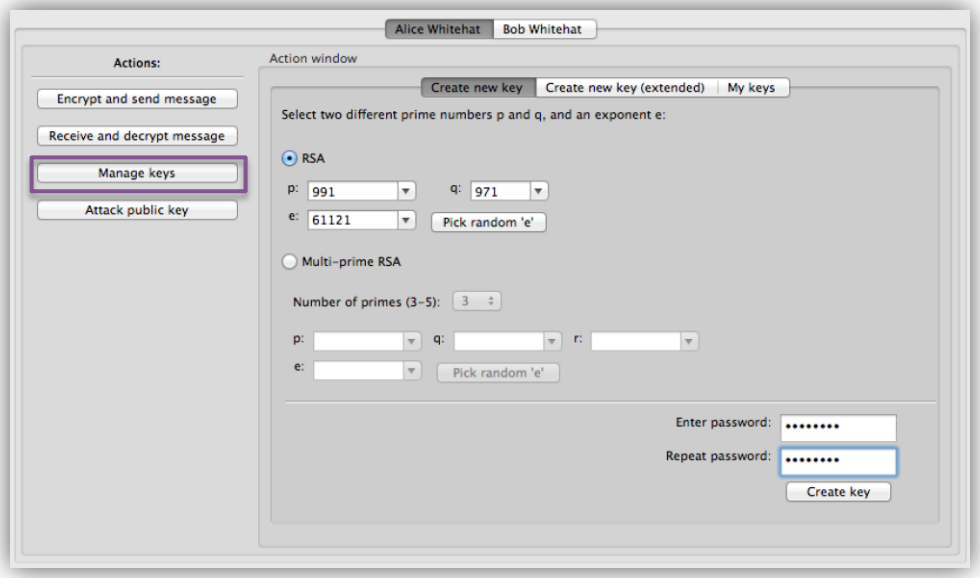
Functionality

- 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



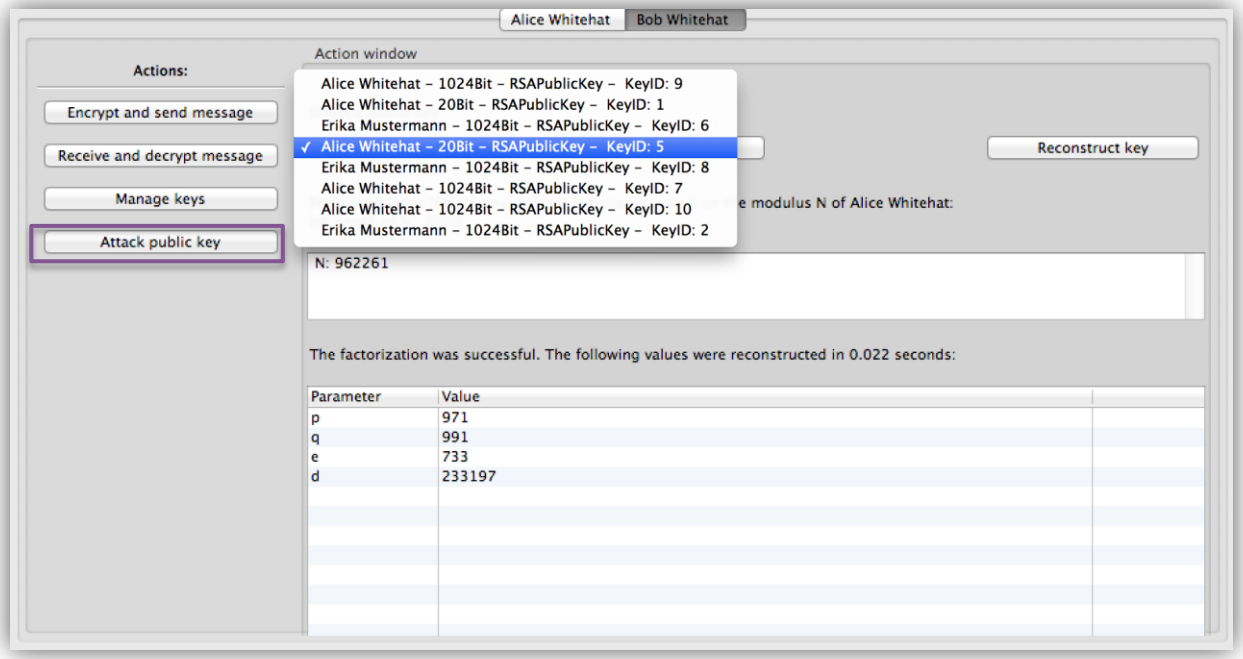
- First, we generate a key which can then be attacked.
- Therefore the plugin provides the option “Manage keys”. We choose primes p and q and a random e.
- Finally, the key have to be saved in a keystore using a password, which we enter in the lower right.
- Now we have created a key for the identity “Alice Whitehat”. Next we try to attack the keys. Using the RSA crypto-system 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.



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” attack, 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 be easily extracted by the attacker. But without knowing the implementation of the key generation, one can hardly detect that 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*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 public 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.
  More over, as P will be generated newly for each new pair of keys the attack is not detectable without reverse engineering the code of the key generator.

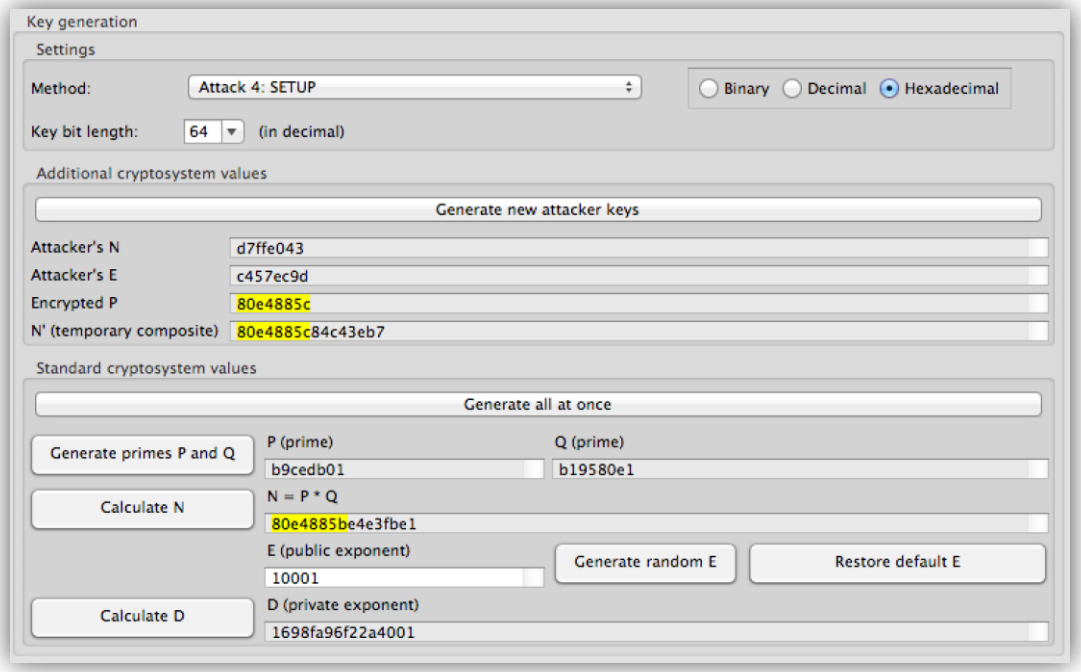
SETUP attack on RSA

Application sample 1/2

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



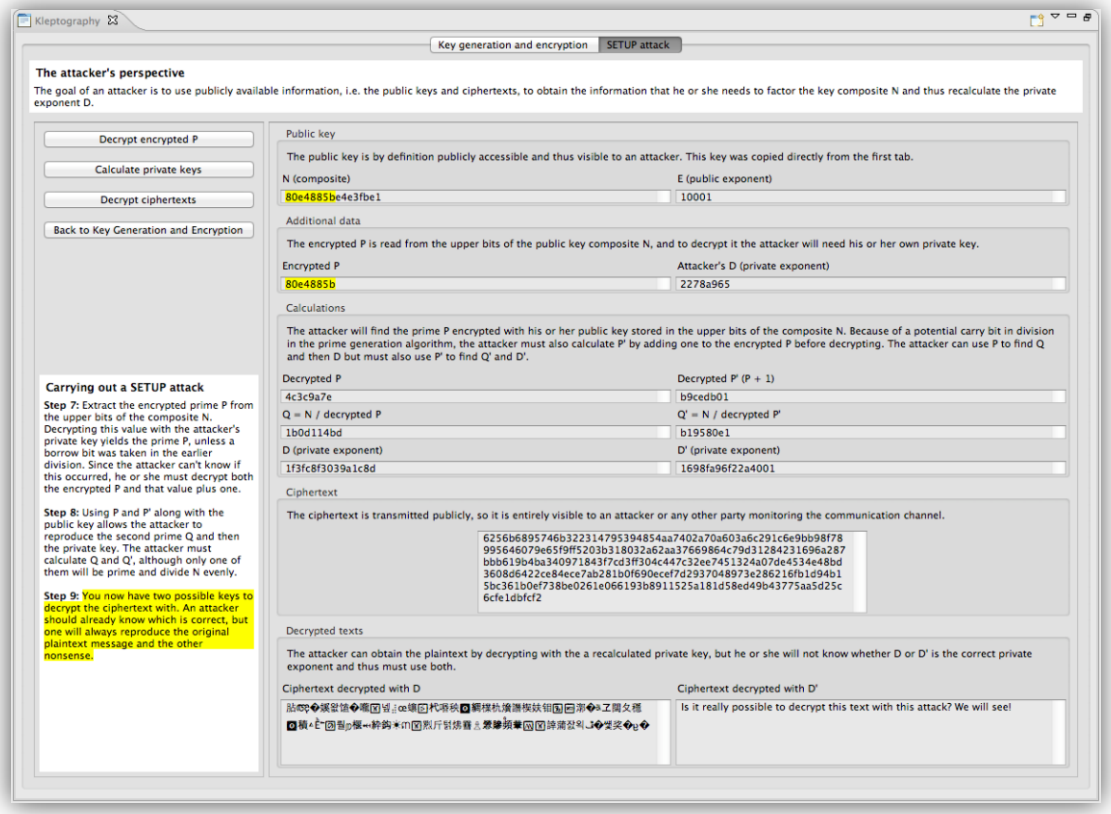
- Finally, N and D can be generated. Then, in the lower third part of the plugin a plaintext can be encrypted.
- By clicking on the button “Save public key and ciphertext”, the user can switch to the tab “SETUP attack” to continue and decrypt the ciphertext.

SETUP attack on RSA

Application sample 2/2

The decryption of 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 button 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 seems still randomly, 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 claiming to know 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:

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

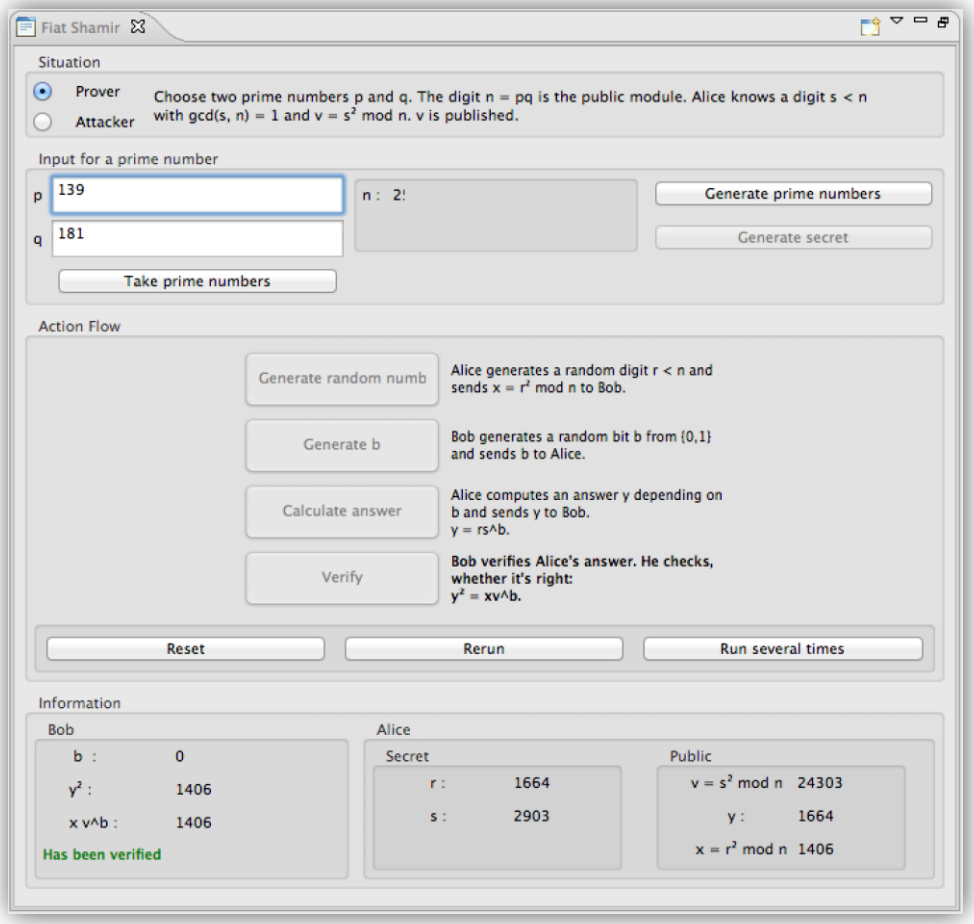
Zero-knowledge protocol: Fiat Shamir

Application sample 1/2

As prover

- Choose the radio button “Prover”.
- First, the two primes p und 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 the person to prove is Alice. As she really knows the secret, so her communication partner will verify that she knows the secret (green hint in the lower left part of the figure).

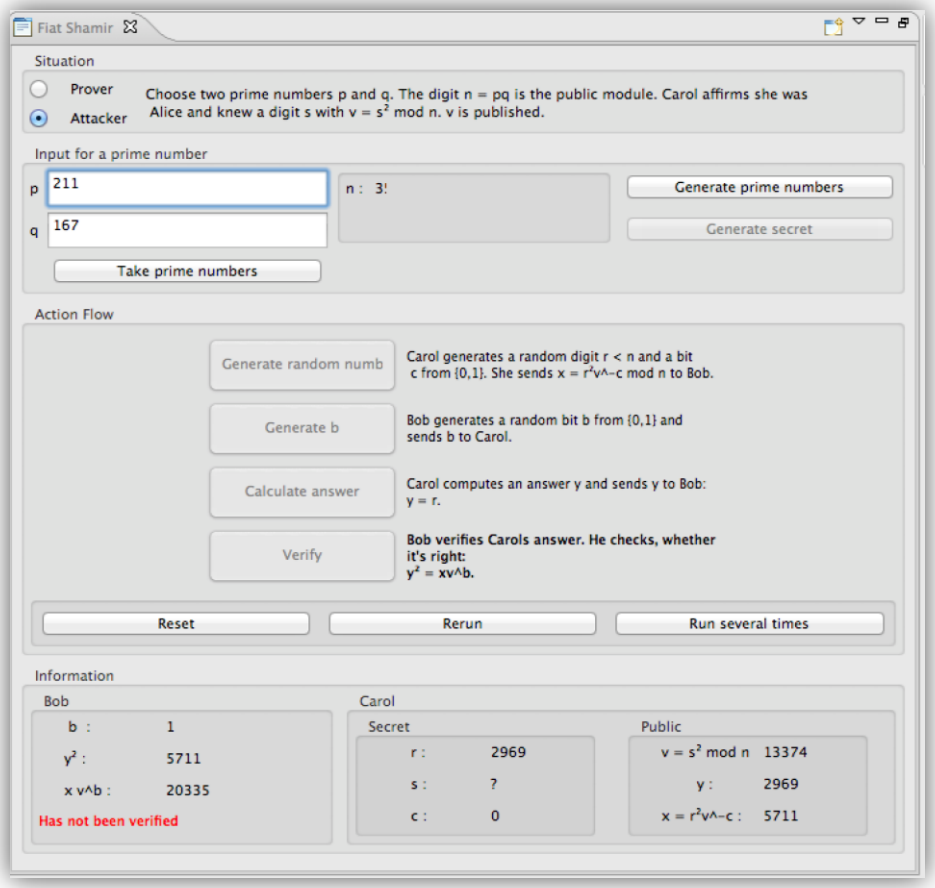


Zero-knowledge protocol: Fiat Shamir

Application sample 2/2

As attacker

- On the other hand, the plugin offers the possibility to act as an “attacker” who pretend to know the secret.
- By 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.



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)”

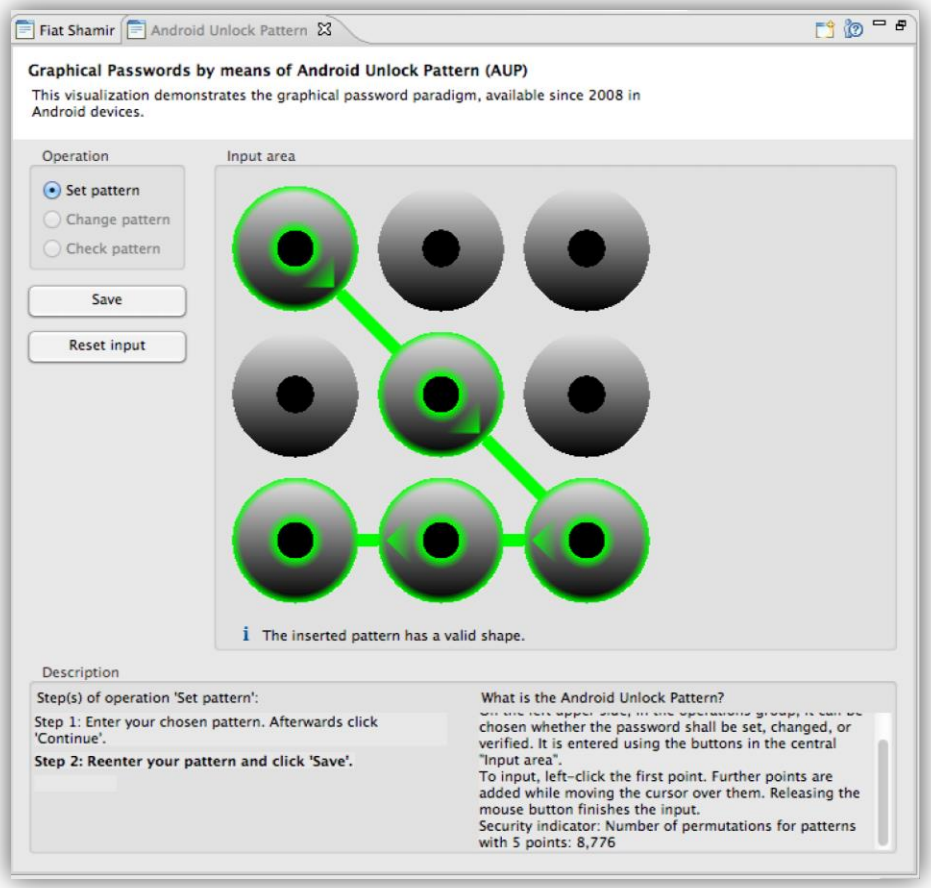
Functionality

- 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 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.

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 to be visited only once.
- Due to this (and some more) rules the possible number of patterns shrinks. In total there are 389,112 different patterns.
- If you compare the 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 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 would not be applied there would exist as many combinations as for the PIN, where each number can be used only once.

Cascades in the Actions window

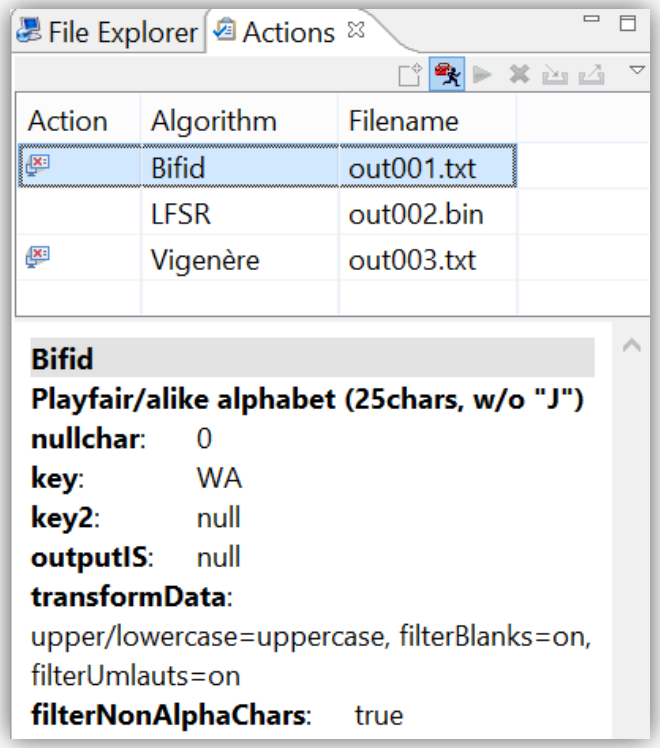
The idea

Functionality

- In the Actions window sequences of application of crypto methods (cascades) can be recorded and reapplied. Basically, it's a recorded 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 page 67).

Examples of application

- Multiple files can quickly be encrypted or decrypted with the same algorithms, settings and ordering of the algorithm.
- With this cascade functionality commutativity, the exchangeability of different encryption algorithms, can easily be investigated (see slides 60 ff).



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

Cascades in the Actions window

The implementation in JCT

In the menu

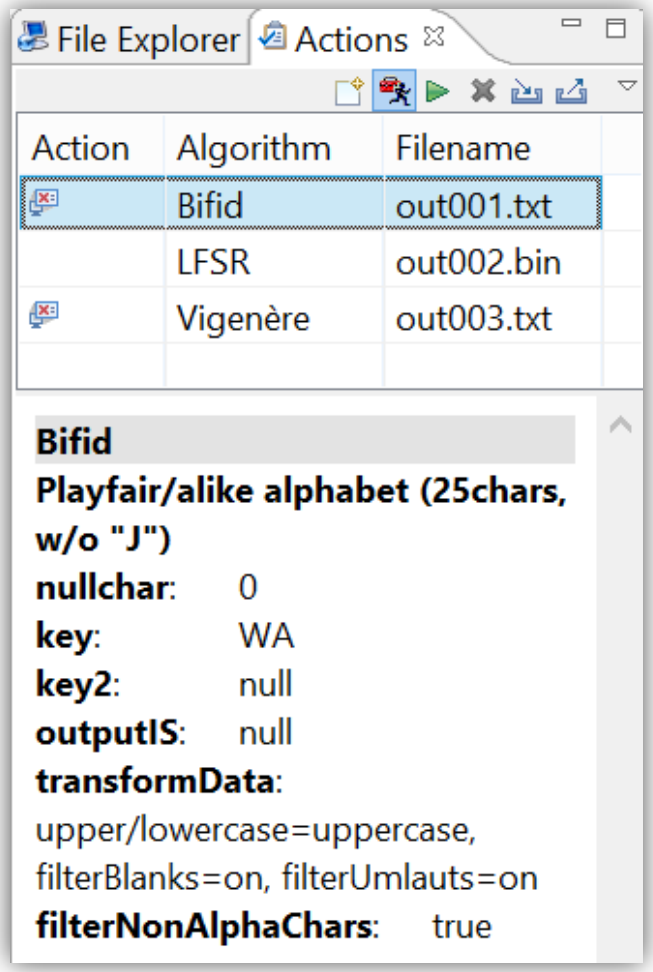
„Window“ \ „Show view“\ „Actions“

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).



Cascades in the Actions window

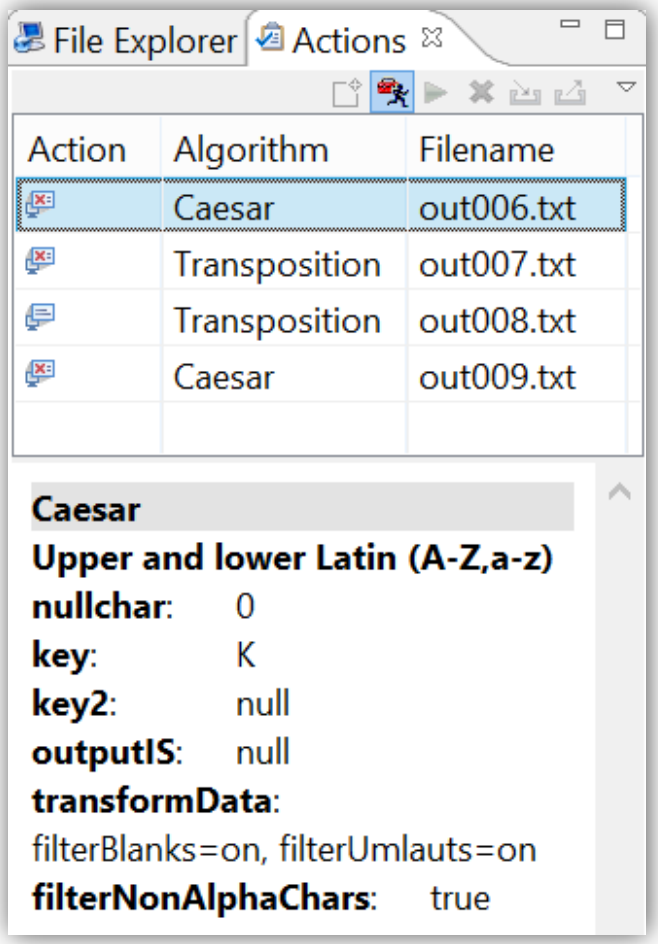
Application sample 1/3

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  .
- **En**crypt an arbitrary text with Caesar:
  “Algorithms“ \ „Classic“ \ „Caesar“
- Add a transposition cipher and **en**crypt:
  „Algorithms“ \ „Classic “ \ „Transposition“
- Apply a transposition **de**cryption which reverts the last encryption: Therefore, simply use the same settings as for the encryption, but just use “Decrypt”.
- **De**cryption of the Caesar encryption applied first.
- Stop the recording with  .

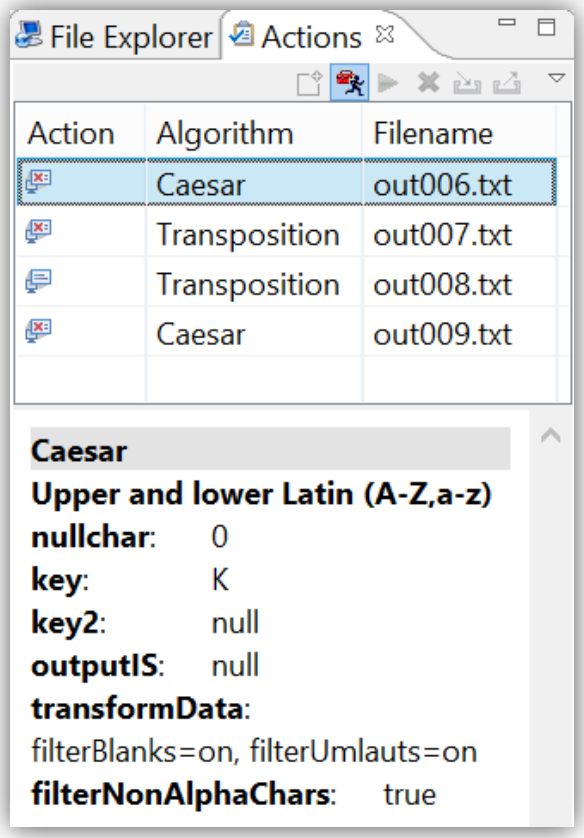
The action window should now look like the figure on the right.



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:

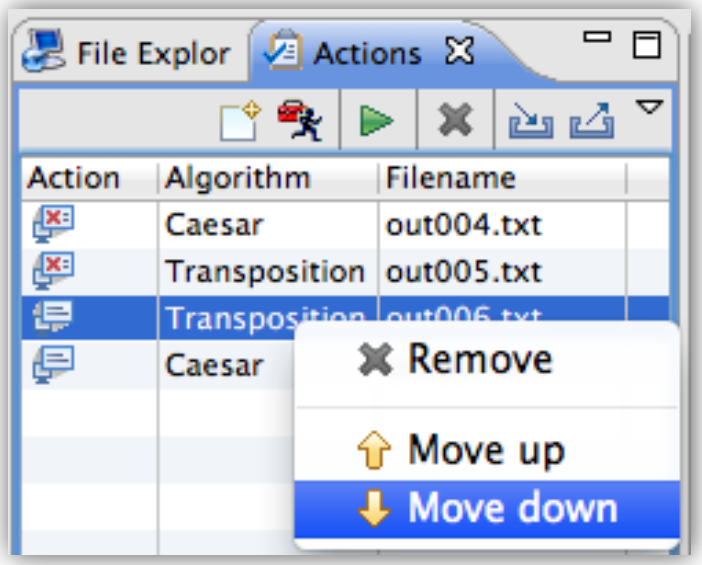
  --> E (Caesar)

  --> 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?

Conclusion

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

- 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.
- Many methods (e.g. ADFGVX and Playfair) use a technique so called ”Fractionation”:
  For instance, pairs of characters are substituted but then single characters are permuted. In this way substitution and transposition are not commutative any longer.

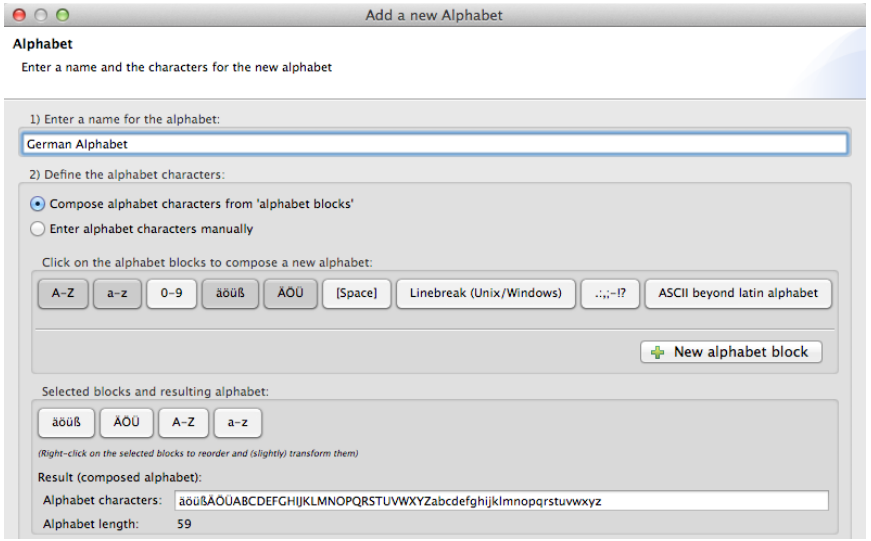
Variable alphabets for classic algorithms

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.

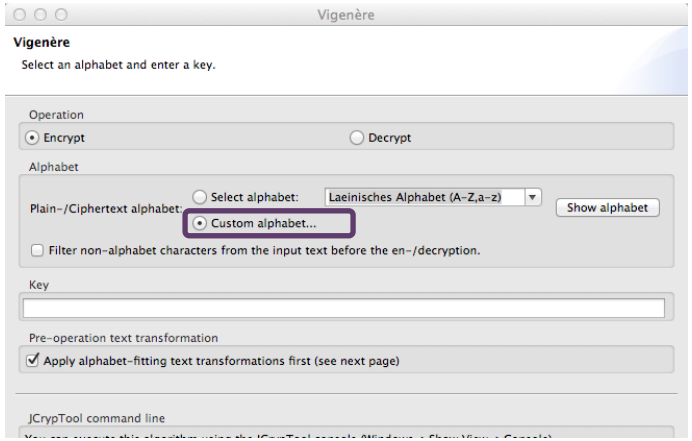


Variable alphabets for classic algorithms

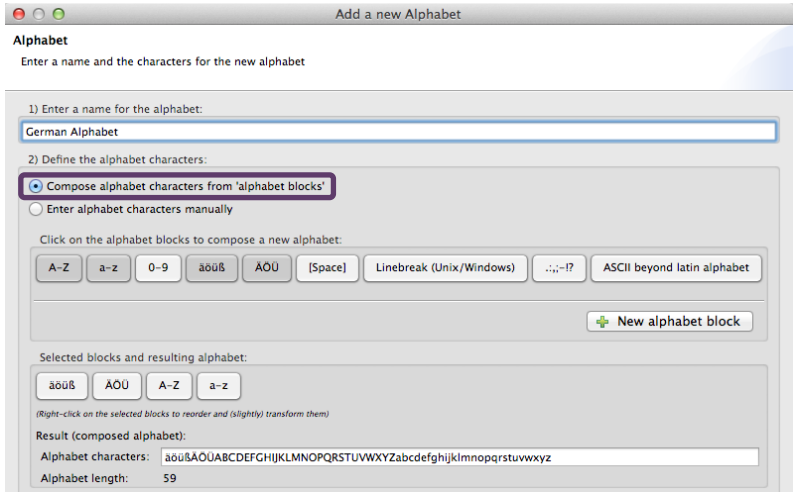
The implementation in JCT 1/2

Generate a custom alphabet

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

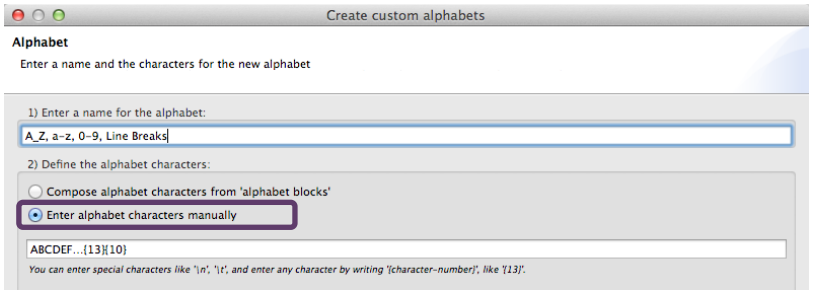


Building blocks compilation



- 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.

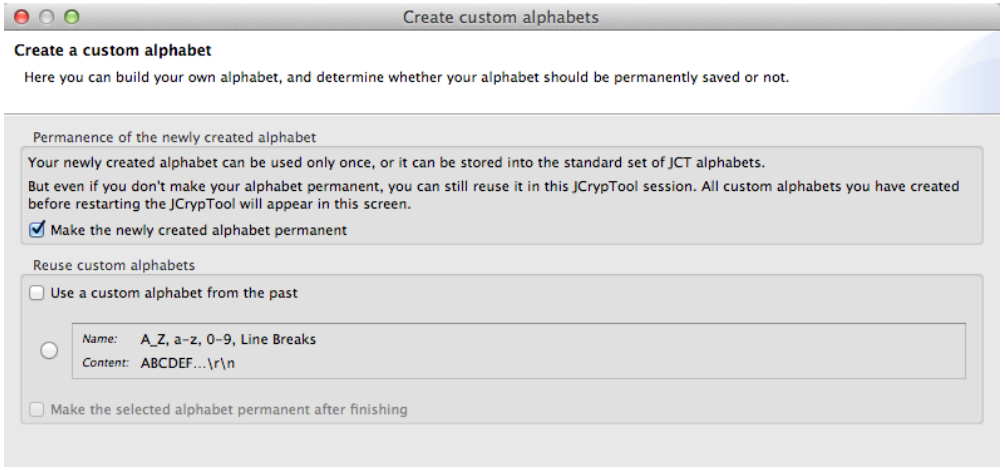
Manuel input



Variable alphabets for classic algorithms
The implementation in JCT 2/2

Further hints

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



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

Variable alphabets for classic algorithms

Educational objective

Conclusion

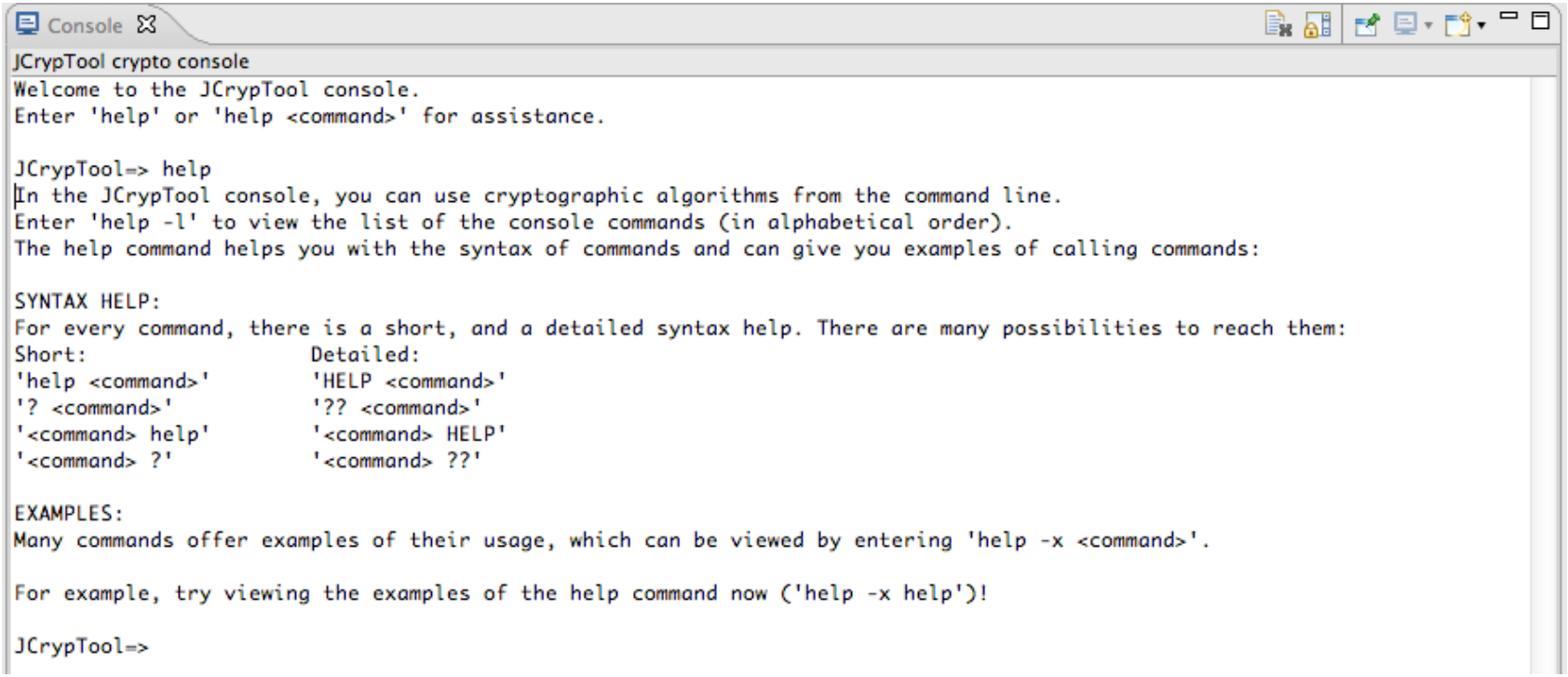
- New alphabets can be created quickly in JCT using existing buildings blocks.
- As special characters can be included as well, there are no limits for the usage of alphabets.
- As it is easy to understand, and efficient to build a custom alphabet, a user is motivated to try out.
- Most of the common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

JCrypTool console for classic methods

The implementation in JCT

The console

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



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

JCrypTool 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:

```
JCrypTool=> 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 2nd 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`“

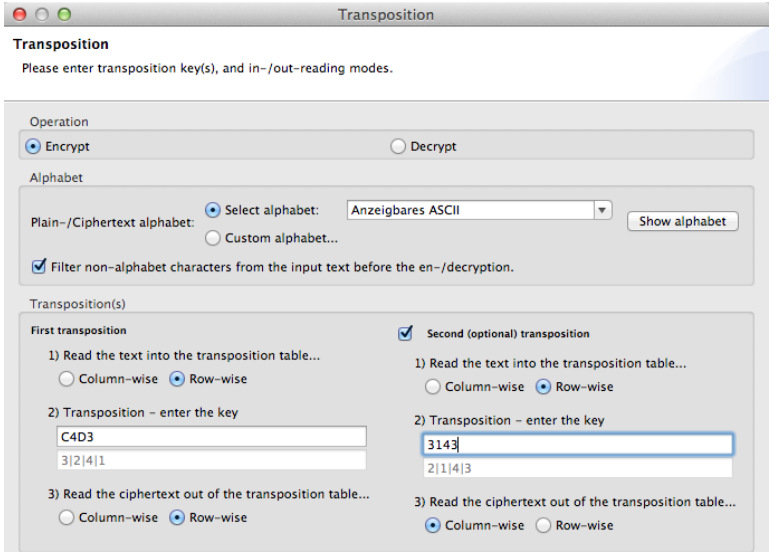
```
JCrypTool=> autovigenere -D -a a-zA-Z -t "TJXSSlxEhLSKACmbXQ" -k THEKEY
ACTIONxCODExDAYYTT
```

JCrypTool 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 1st and 2nd 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.

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

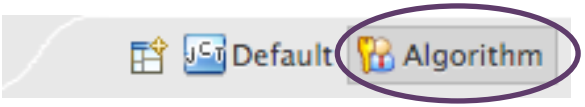
```
JCrypTool=> HELP transposition
Transposes characters of the plain (columnar transposition with definable read-in / read-out directions).
Syntax:transposition [-a <ALPHABET>] -D | -E  -ed | -f <FILE_PATH> | -t <TEXT>  -k <KEY> [-k2 <KEY>] [-noFi]  [-t1ReadIn
       <ORDER = 'cw'/'rw'>] [-t1ReadOut <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
                                                             signalizes 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 windows:

Keystore

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

Algorithms

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

Operations

- An algorithm, chosen via double-click in the Algorithm Explorer, is listed here. Then additional settings (e.g. the source of 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

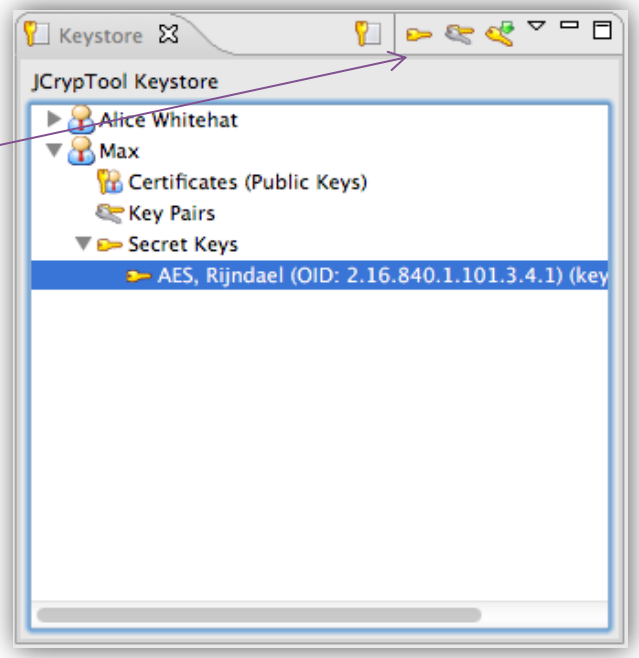
The perspective “Algorithm“

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 contact

- First, we generate a key to be used for our encryption.
- With  a new key can be created.
  As AES is a symmetric crypto system, it does only need a single secret key and not a key pair. For asymmetric systems appropriate key pairs can be generated with 
  (step 1 on slide 76).
- In the wizard „New Symmetric Key“ we choose „AES, Rijndael (OID 2.16.840.1.101.3.4.1)“[1], 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”).



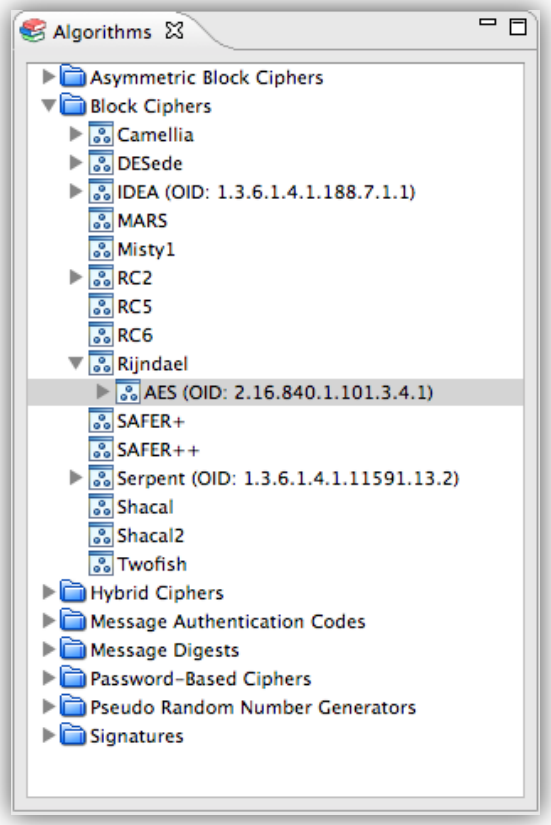
[1] OID: Object Identification, a unique identifier for an algorithm
Defined by ITU (http://en.wikipedia.org/wiki/Object_identifier).

The perspective “Algorithm“

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

Selection of the algorithm

- 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 76).
- A wizard appears where padding and mode[1] of the block cipher can be adjusted.
  Additionally, more algorithm specific settings can be adjusted here (e.g. for AES the length of each block in bits).
- Note: The algorithms in this explorer are as well grouped by the kind of the cryptographic method.



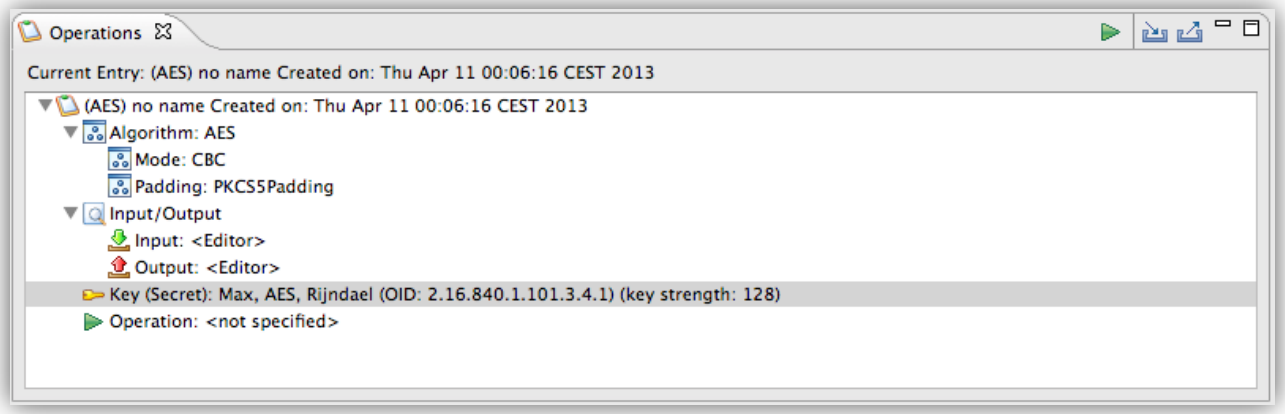
[1] The mode of a block cipher is responsible for the mapping of the plain text to the blocks, which will be eventually encrypted. If in the last block some bits are missing, the padding rules how these bits will be filled.

The perspective “Algorithm“

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

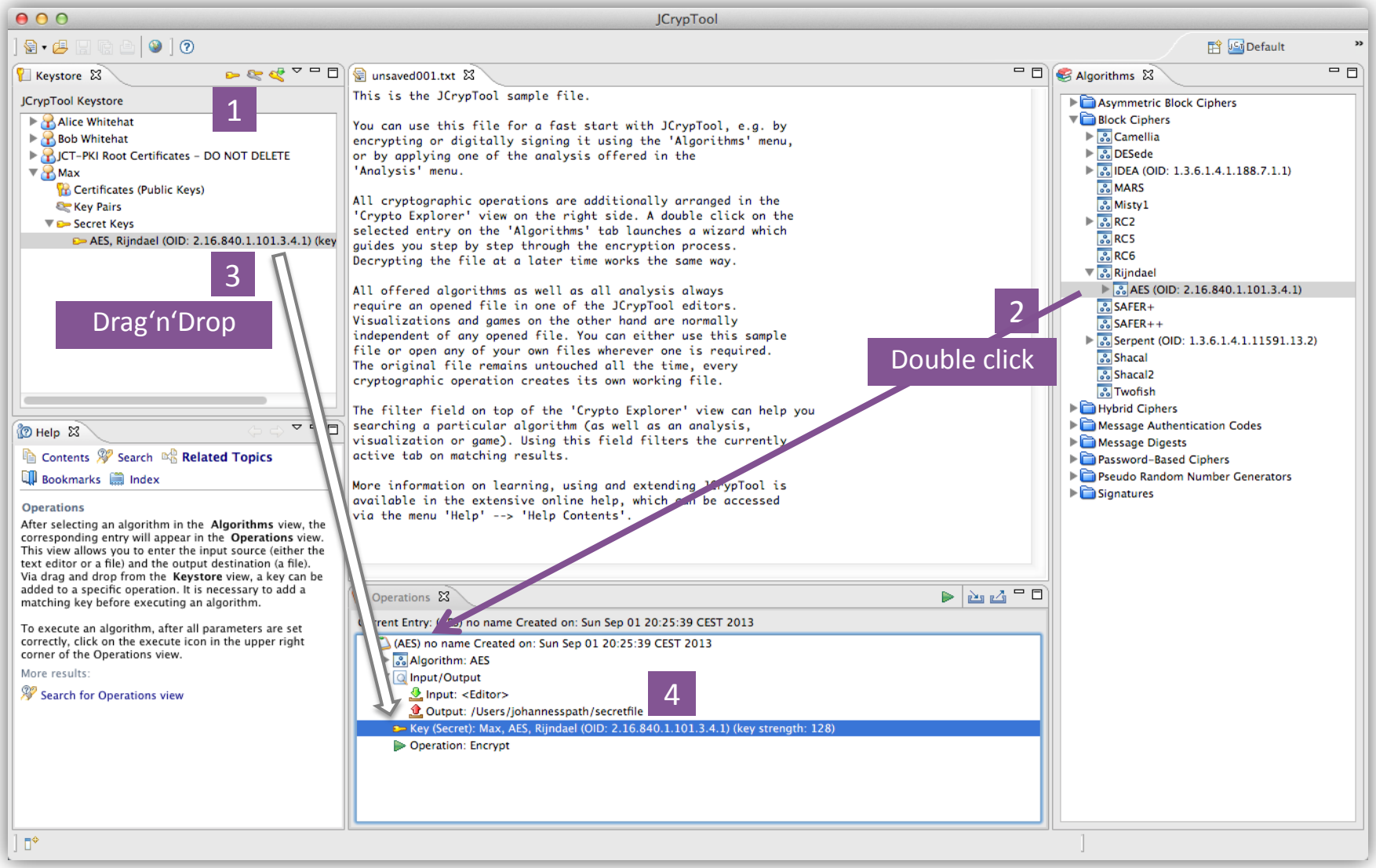
Settings for the input and output

- Via the double click the algorithm Rijndael was added to the Operations tab.
- Via drag’n’drop the generated key listed in the JCT keystore can be copied on the key field of the algorithm (see slide 76, step 3).
- The option Input/Output offers to select via double click 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 76, 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.



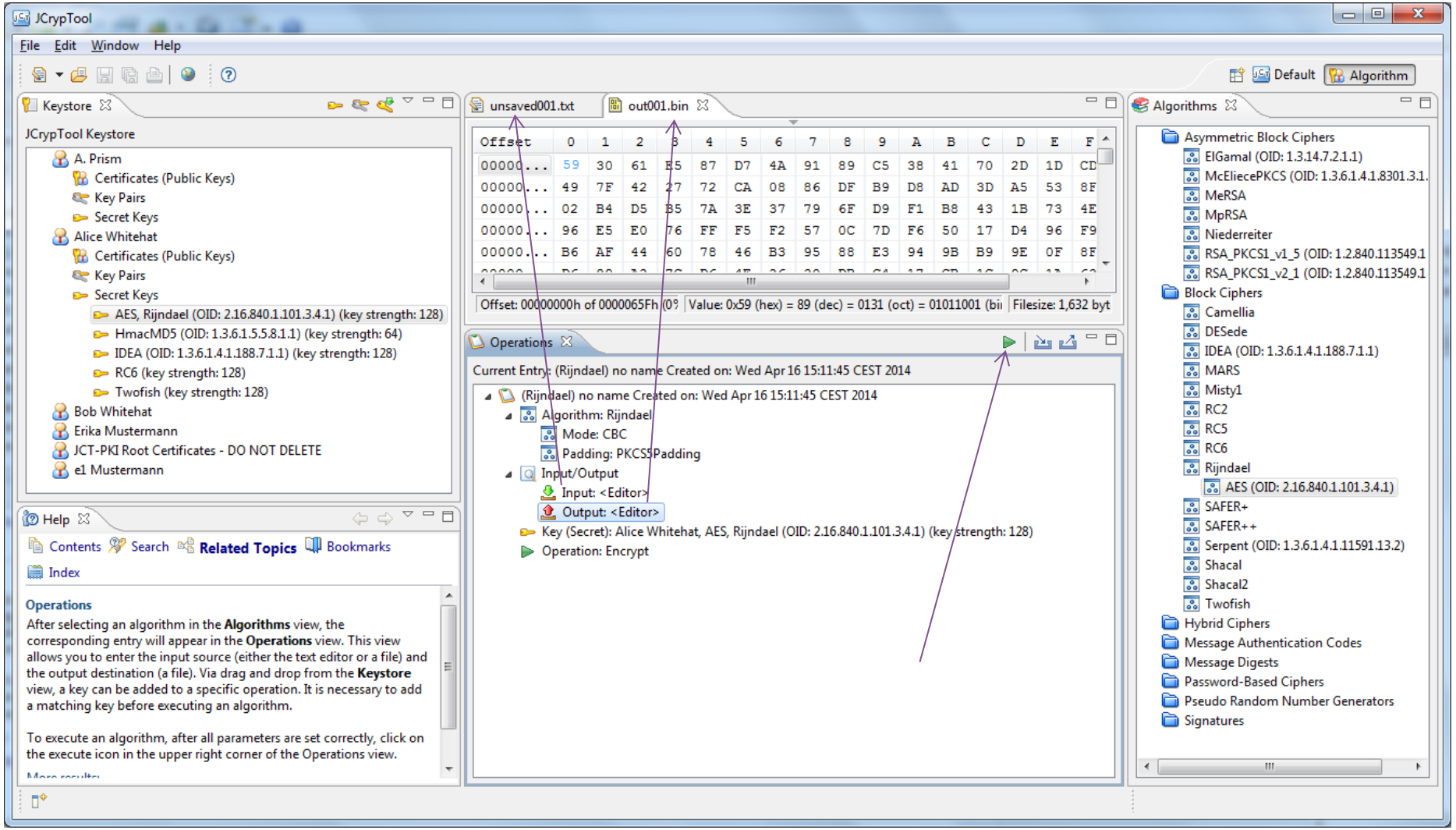
The perspective “Algorithm“

Overview of the 4 steps to select and customize an operation



The perspective “Algorithm“

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



Further functions in JCrypTool

Further samples what's in JCrypTool

- Tri-partite key agreements
- Visualization of the inner states of DES
- Visualization of calculations on elliptic curves over real and discrete fields
- Visualization of Quantum Key Agreement, BB84 protocol
- Visualization of the simple (SPA) and differential (DPA) power analysis attack against RSA
- Quick solver of the number shark game with heuristic methods; solving of Sudoku variants
- Mathematical games: number shark, divider game, zero-knowledge Sudoku
- Entropy analysis
- Dynamic visualization of Huffman coding trees
- Signature demonstration, signature and certificate verification
- How does a PKI work?
- Visualization of the SSL/TLS handshake
- …
- Implementation and visualization of ARC4 and Spritz (almost done)
- Brute-force attack against modern symmetric ciphers (under construction)
- Dynamic linking of BouncyCastle as Algorithm perspective (already started)

Agenda

How to participate

Overview

JCrypTool – Request for participation

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 CrypTool 1 and
  - in the new projects (preferred):
    - C# project: „CrypTool 2.0“ = CT2 (http://www.cryptool.org/en/ct2-volunteer-en)
    - Java project: „JCrypTool“ = JCT (http://www.cryptool.org/en/jct-volunteer-en)
    - Browser project: „CrypTool-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
- [C030] Framework for analyzing modern symmetric ciphers

Further things of high interest

- One place for all the manipulations of frequency tables (creation, exchange, deepness) and of permutations
- x[C010] Key Storage, [N005] Hudson Build Process, [N010] Automatic Tests

Open tasks and more ideas are mentioned on the developer sites:

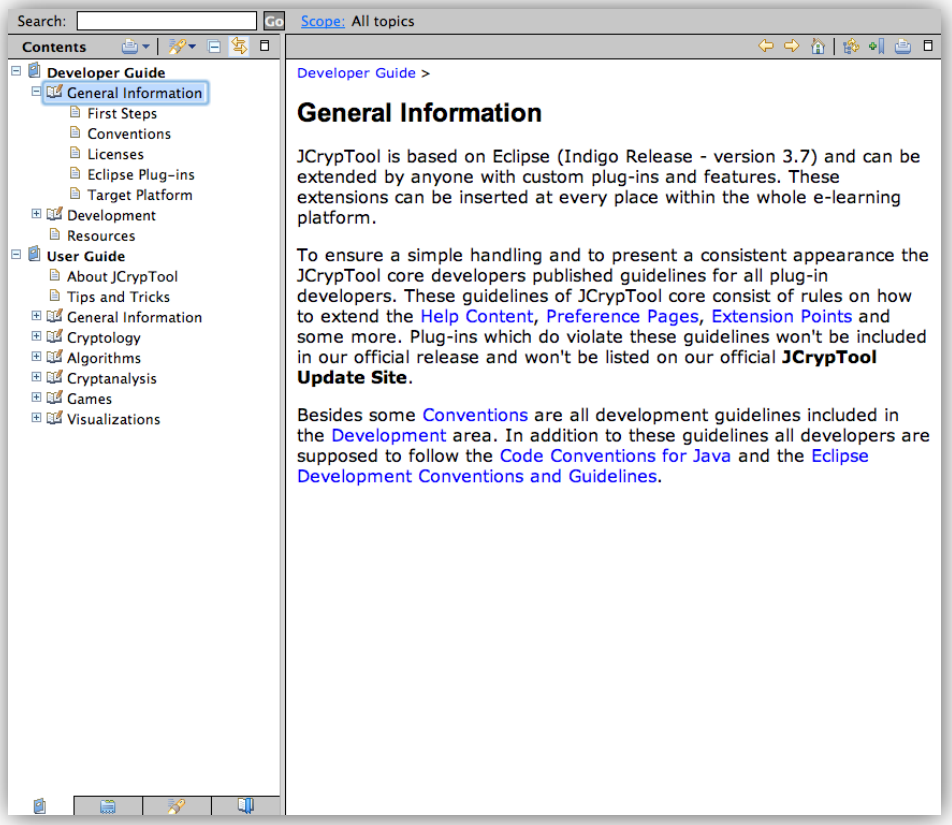
- JCrypTool: http://www.cryptool.org/en/jct-volunteer-en
  https://github.com/jcryptool/core/wiki/project-Ideas
  https://github.com/jcryptool/core/wiki/Google-Summer-of-Code-2015

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

- Information for developing plugins is provided in the JCT online help (same as with Eclipse). The wiki in the internet offers links and information for JCT core developers, or current issues, which after a release could not be included in the online help.
- 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.



Contacts

Prof. Bernhard Esslinger

University of Siegen

bernhard.esslinger@uni-siegen.de
bernhard.esslinger@gmail.com

Dominik Schadow

JCT project lead

info@xml-sicherheit.de

www.cryptool.org