CRYPTOLOGY WITH CRYPTOOL v 1.4.30

Practical Introduction to Cryptography and Cryptanalysis

Scope, Technology, and Future of CrypTool

Prof. Bernhard Esslinger and the CrypTool Team, August 2010

www.cryptool.org
www.cryptool.com
www.cryptool.de
www.cryptool.es
www.cryptool.pl

Content (I)

Content (II)

III. Examples

IV. Project / Outlook / Contact

Content

Relevance of Cryptography

Examples of Applied Cryptography

- Phone cards, cell phones, remote controls
- Cash machines, money transfer between banks
- Electronic cash, online banking, secure email
- Satellite TV, pay-per-view TV
- Immobilizer systems in cars
- Digital Rights Management (DRM)
- Cryptography is no longer limited to agents, diplomats, and the military. Cryptography is a modern, mathematically characterized science.
- The breakthrough of cryptography followed the broadening usage of the Internet
- For companies and governments it is important that systems are secure and that

users (i.e., clients and employees) are aware of and understand IT security!

Definition Cryptology and Cryptography

Cryptology *(from the Greek kryptós, "hidden," and lógos, "word") is the science of secure (or, generally speaking, secret) communication. This security requires that legitimate users, a transmitter and a receiver, are able to transform information into a cipher by virtue of a key – that is, a piece of information known only to them. Although the cipher is inscrutable and often unforgeable to anyone without this secret key, the authorized receiver can either decrypt the cipher to recover the hidden information or verify that it was sent in all likelihood by someone possessing the key.*

Cryptography *was concerned initially with providing secrecy for written messages. Its principles apply equally well, however, to securing data flow between computers or to encrypting television signals. Today, the modern (mathematical) science of cryptology is not just a set of encryption mechanisms. It has since been applied to a broad range of aspects of modern life, including data and message integrity, electronic signatures, random numbers, secure key exchange, secure containers, electronic voting, and electronic money.*

Source: Britannica (www.britannica.com)

A similar definition can be found on Wikipedia: http://en.wikipedia.org/wiki/Cryptography

Cryptography – Objectives

- **Confidentiality**

  Information can be made effectively unavailable or unreadable for unauthorized individuals, entities, and processes.

- **Authentication**

  The receiver of a message can verify the identity of the sender.

- **Integrity**

  Integrity ensures that data has not been altered or destroyed in an unauthorized manner.

- **Non-Repudiation**

  The receiver can prove that the message he or she received is precisely what the sender sent; the sender will have no means to deny any part of his or her participation.

The CrypTool Project

- Originated as an awareness program for a large bank (internal training)
  → **Employee education**
- Developed in cooperation with universities (improvement of education)
  → **Media didactic approach and standard oriented**

  1998 **Project start** – over 40 person-years of effort have since been invested
  2000 CrypTool available as **freeware**
  2002 CrypTool available on the **Citizen's CD and website of the BSI** (German Information Security Agency)
  2003 CrypTool becomes **open source** – hosting by University of Darmstadt (Prof. Eckert)
  2007 CrypTool available in German, English, Polish, and Spanish
  2008 .NET and Java versions started – hosted by University of Duisburg (Prof. Weis) and SourceForge
  2010 CT1 available in fifth language, Serbian; .NET and Java versions to be released

- **Awards**

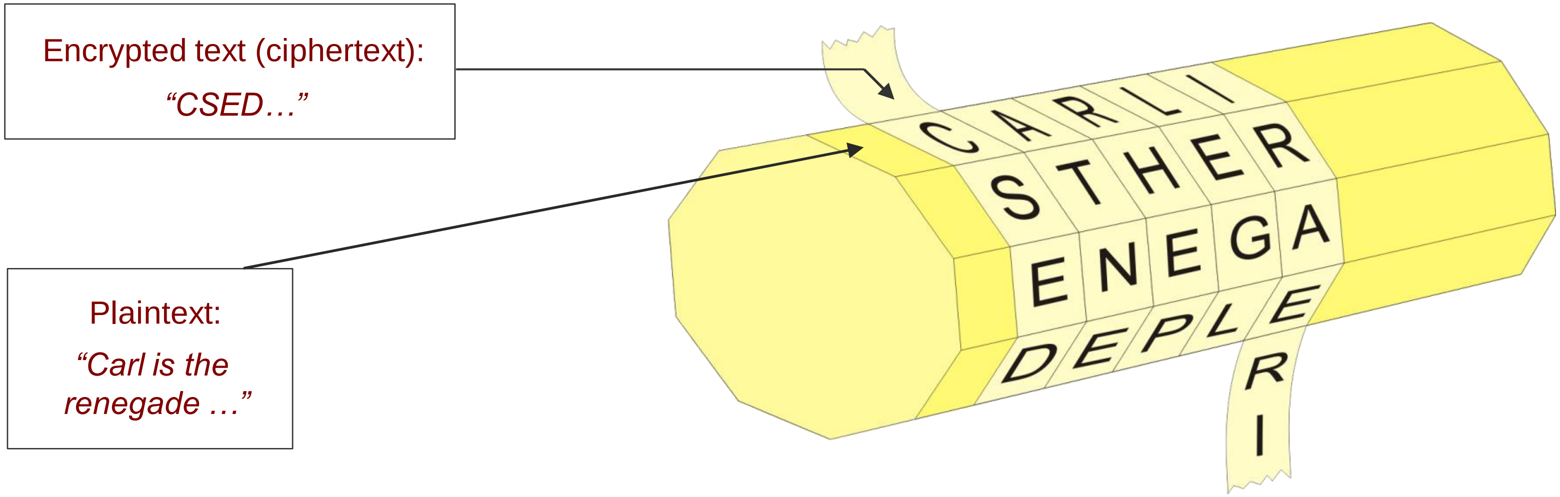
  2004 TeleTrusT (TTT Förderpreis / Sponsorship Award)
  2004 NRW (IT Security Award NRW)
  2004 RSA Europe (Finalist of European Information Security Award 2004)
  2008 "Selected Landmark" in initiative "Germany – Land of Ideas"

- **Developers**
  - Developed by people from companies and universities in many different countries.
  - → Currently there are about 50 people working on CrypTool worldwide.
    Additional project members or applicable resources are always appreciated.

Examples of Early Cryptography (1)

Ancient encryption methods

- **Tattoo on the shaven head of a slave, concealed by regrown hair**
- **Atbash** (circa 600 B.C.)
  - Hebrew secret language, reversed alphabet
- **Scytale from Sparta** (circa 500 B.C.)
  - Described by Greek historian/author Plutarch (45 - 125 B.C.)
  - The sender and receiver each need a cylinder (such as a wooden rod) with the same diameter
  - Transposition (plaintext characters are re-sorted)



Examples of Early Cryptography (2)

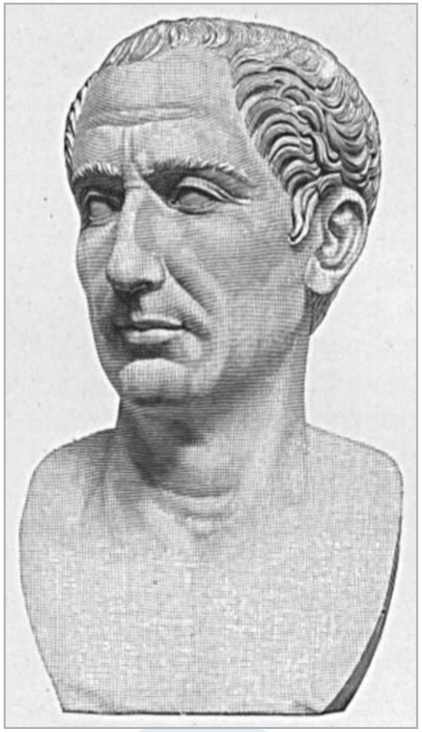
Caesar encryption (mono-alphabetic substitution cipher)

- **Caesar encryption** (Julius Caesar, 100 - 44 B.C.)
- Simple substitution cipher

```
GALLIA  EST  OMNIS  DIVISA ...

Plaintext:
    ABCDEFGHIJKLMNOPQRSTUVWXYZ
Secret alphabet:
    DEFGHIJKLMNOPQRSTUVWXYZABC

JDOOLD  HVW  RPQLV  GLYLVD ...
```



- **Attack:** Frequency analysis (typical character allocation)

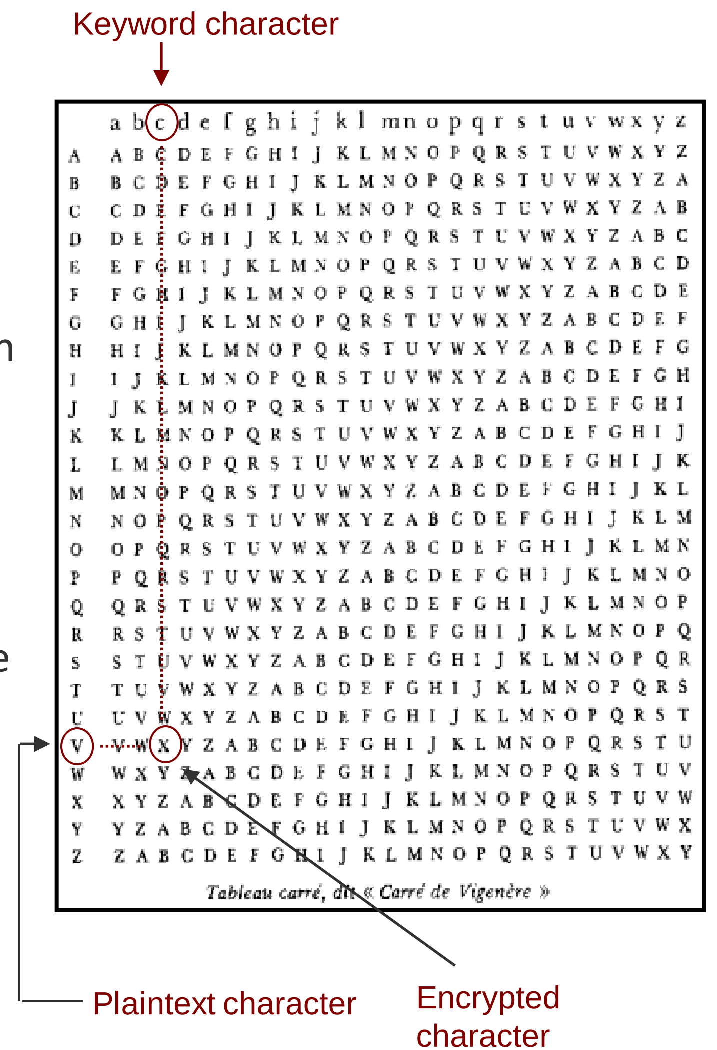
Presentation with CrypTool via the following menus:

- Animation: “Indiv. Procedures” \ “Visualization of algorithms” \ “Caesar”
- Implementation: “Crypt/Decrypt” \ “Symmetric (classic)” \ “Caesar / Rot-13”

Examples of Early Cryptography (3)

Vigenère encryption (poly-alphabetic substitution cipher)

- **Vigenère encryption** (Blaise de Vigenère, 1523-1596)
- Encryption with a keyword using a key table
- Example
  Keyword: **`CHIFFRE`**
  Encrypting: **`VIGENERE`** becomes **`XPOJSVVG`**
- The plaintext character (V) is replaced by the character in the corresponding row and in the column of the first keyword character (c). The next plaintext character (I) is replaced by the character in the corresponding row and in the column of the next keyword character (h), and so on.
- If all characters of the keyword have been used, then the next keyword character is the first key character.
- **Attack** (via Kasiski test; other tests also exist): Plaintext combinations with an identical cipher text combination can occur. The distance of these patterns can be used to determine the length of the keyword. An additional frequency analysis can then be used to determine the key.



	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

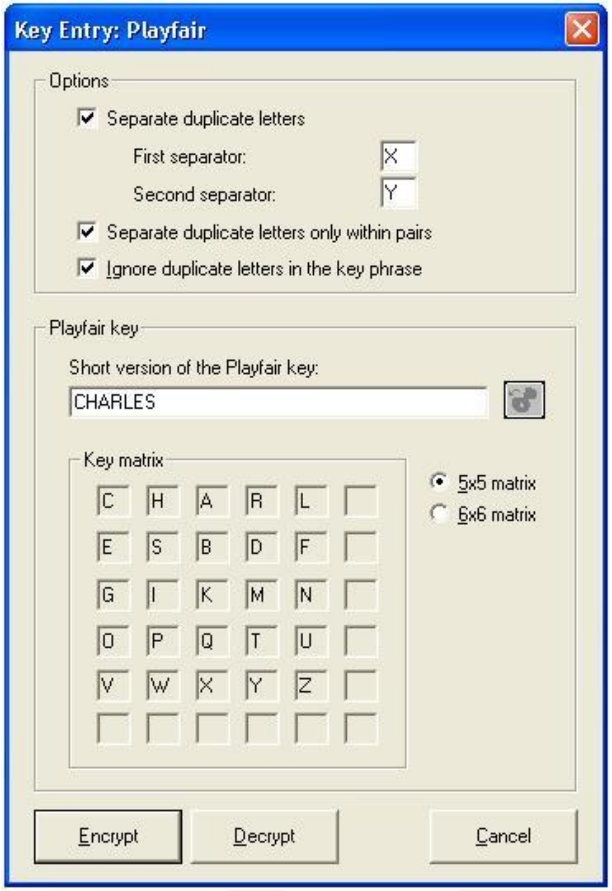
Tableau carré, dit « Carré de Vigenère »

Examples of Early Cryptography (4)

Other classic encryption methods

- **Homophone substitution**
- **Playfair** (invented 1854 by Sir Charles Wheatstone, 1802-1875)
  - Published by Baron Lyon Playfair
  - Substitution of one character pair by another one based on a square-based alphabet array
- **Transfer of book pages**
  - Adaptation of the One-Time Pad (OTP)
- **Turning grille** (Fleissner)
- **Permutation encryption**
  - "Double Dice" (double column transposition)

    (Pure transposition, but very effective)



Cryptography in Modern Times

Developments in cryptography from 1870-1970

Classic methods

- are still in use today

  (since not everything can be done by a computer…)
- and their principles of transposition and substitution the foundation of the design of modern algorithms, which combine simpler operations at a bit level (a type of multiple encryption or cipher cascade), use block ciphers, and/or use repeated uses of an algorithm over multiple rounds.

Encryption becomes

- more **sophisticated**,
- **mechanized** or **computerized**, and
- remains **symmetric**.

Example from the First Half of the 20th Century

Mechanical encryption machines (rotor machines)

Enigma Encryption (Arthur Scherbius, 1878-1929)

- More than 200,000 machines were used in WWII.
- The rotating cylinders encrypt every character of the text with a new permutation.
- The Polish Cipher Bureau broke the pre-war Enigma prototype as early as 1932.
- Based on this work, the later Enigma was broken only with massive effort. About 7000 cryptographers in the UK used decryption machines, captured Enigma prototypes, and intercepted daily status reports (such as weather reports).
- **Consequences of the successful cryptanalysis**
  "The successful cryptanalysis of the Enigma cipher was a strategic advantage that played a significant role in winning the war. Some historians assert that breaking the Enigma code shortened the war by several months or even a year."

  (translated from http://de.wikipedia.org/wiki/Enigma_%28Machine%29 - March 6, 2006)

Cryptography – Important Insights (1)

- **Kerckhoffs' principle** (first stated in 1883)
  - Separation of algorithm (method) and key e.g. Caesar encryption:
    Algorithm: "Shift alphabet by a certain number of positions to the left"
    Key: The "certain number of positions"
  - Kerckhoffs' principle:
    The secret lies within the key and not within the algorithm;
    "security through obscurity" is invalid
- **One-Time Pad – Shannon / Vernam**
  - Theoretically completely unbreakable, but highly impractical (used by the red telephone*)
- **Shannon's concepts: Confusion and Diffusion**
  - Relation between M, C, and K should be as complex as possible
    (M=message, C=cipher, K=key)
  - Every ciphertext character should depend on as many plaintext characters and as many characters of the encryption key as possible
  - "Avalanche effect" (small modification, big impact)
- **Trapdoor function** (one-way function)
  - Fast in one direction, not in the opposite direction (without secret information)
  - Possessing the secret allows the function to work in the opposite direction
    (access to the trapdoor)

* See http://en.wikipedia.org/wiki/Moscow-Washington_hotline

Examples of Breaches of Kerckhoffs' Principle

The secret should lie within the key, not in the algorithm

- **Cell phone encryption penetrated** (December 1999)

  "Israeli researchers discovered design flaws that allow the descrambling of supposedly private conversations carried by hundreds of millions of wireless phones. Alex Biryukov and Adi Shamir describe in a paper to be published this week how a PC with 128 MB RAM and large hard drives can penetrate the security of a phone call or data transmission in less than one second. The flawed algorithm appears in digital GSM phones made by companies such as Motorola, Ericsson, and Siemens, and used by well over 100 million customers in Europe and the United States." […]

  "Previously the GSM encryption algorithms have come under fire ***for being developed in secret away from public scrutiny*** *-- but most experts say high security can only come from published code. Moran [GSM Association] said "it wasn't the attitude at the time to publish algorithms" when the A5 ciphers was developed in 1989, but* ***current ones being created will be published for peer review.****"*
  [http://www.wired.com/politics/law/news/1999/12/32900]

- **Additional example:** In 1999, Netscape Navigator stored email server passwords using a weak proprietary encryption method.

Sample of a One-Time Pad Adaptation

Clothes hanger of a Stasi agent
with a secret one-time pad
(source: *Spiegel Spezial, 1/1990)*

Menu:
“Crypt/Decrypt” \
“Symmetric (classic)” \
“Vernam”

Key Distribution Problem

Key distribution for symmetric encryption methods

If **2 persons** communicate with each other using symmetric encryption, they **need one common secret key**.

If n persons communicate with each other, then they need $\mathbf{S_n = n * (n-1) / 2}$ keys.

That is:

n = 100 persons require

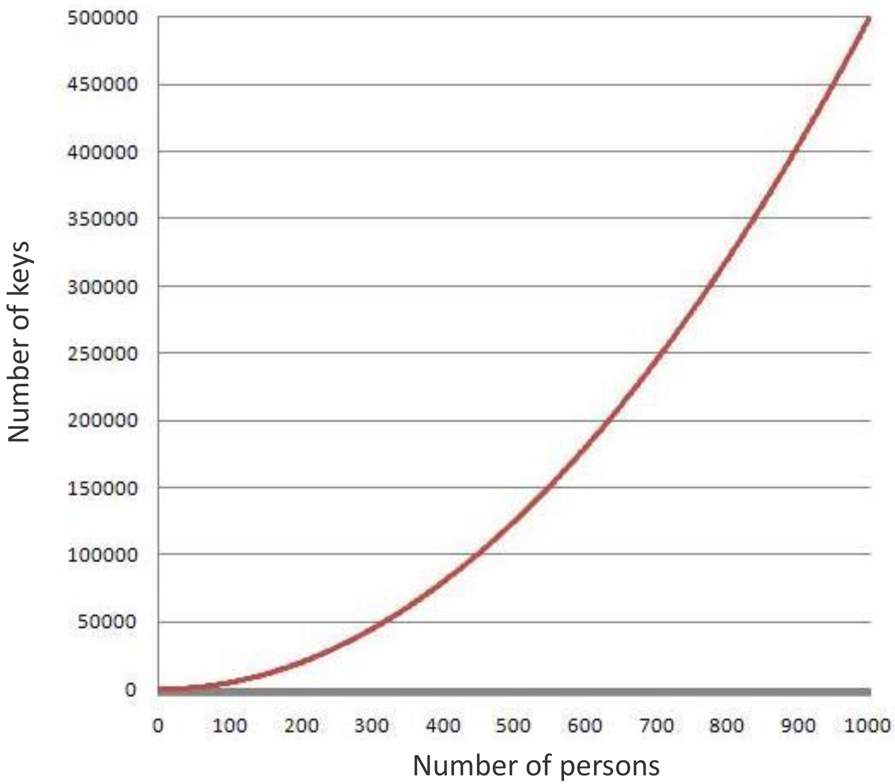
$\mathbf{S_{100} = 4{,}950}$ keys; and

n = 1,000 persons require

$\mathbf{S_{1000} = 499{,}500}$ keys.

⇨ A factor of 10 more persons means a factor of 100 more keys.

Number of required keys



Cryptography – Important Insights (2)

Solving the key distribution problem through asymmetric cryptography

Asymmetric cryptography

- For centuries it was believed that the sender and receiver need to know the same secret.
- New idea: Every person needs a key pair (which also solves the key distribution problem)

Asymmetric encryption

- "Everyone can lock a padlock or drop a letter in a mail box."
- MIT, 1977: Leonard Adleman, Ron Rivest, Adi Shamir (well known as RSA)
- GCHQ Cheltenham, 1973: James Ellis, Clifford Cocks (publicly declassified December 1997)

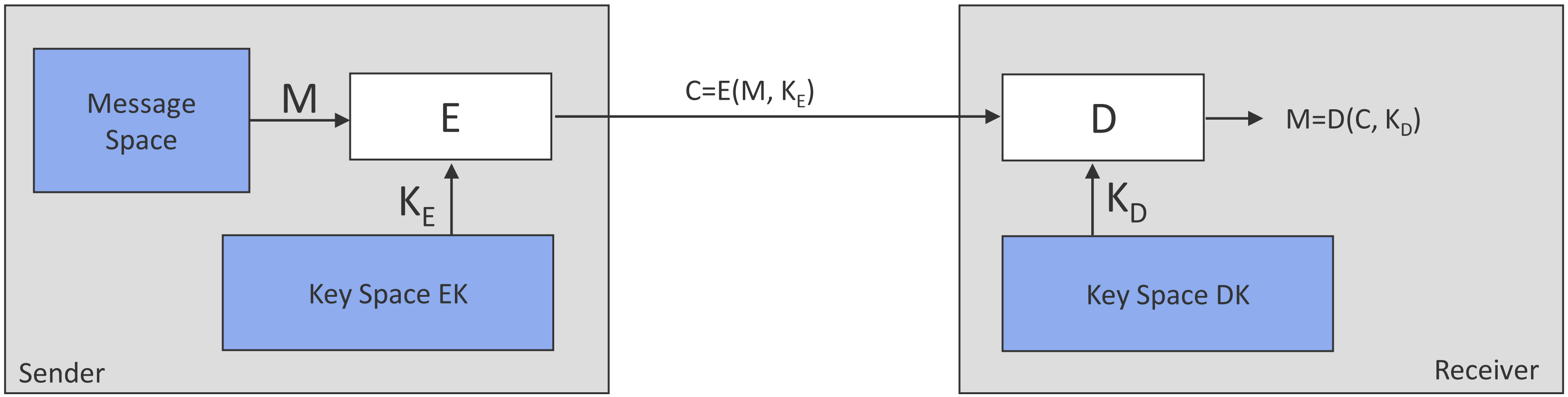
Key distribution

- Stanford, 1976: Whitfield Diffie, Martin Hellman, Ralph Merkle
  (Diffie-Hellman key exchange)
- GCHQ Cheltenham, 1975: Malcolm Williamson

Security in open networks (such as the Internet) would be extremely expensive and complex without asymmetric cryptography!

Performing Encryption and Decryption

Symmetric und asymmetric encryption



secret

a) Symmetric Encryption: $\mathbf{K_E = K_D}$ (e.g. AES)

b) Asymmetric Encryption: $\mathbf{K_E \neq K_D}$ (e.g. RSA)

public private/secret

Cryptography – Important Insights (3)

Increasing relevance of mathematics and information technology

- **Modern cryptography** is increasingly based on **mathematics**
  - There are still new symmetric encryption methods, such as AES; these often feature better performance and shorter key length compared to asymmetric methods that are based purely on mathematical problems.
- The security of encryption methods heavily depends on the current state of **mathematics** and **information technology** (IT)
  - Computation complexity (meaning processing effort in relation to key length, storage demand, and data complexity).
    → see RSA: Bernstein, TWIRL device, RSA-160, RSA-768 (CrypTool script, chapter 4.11.3)
  - Major topics in current research:
    Factorization of very large numbers, non-parallelizable algorithms (to counter quantum computers), protocol weaknesses, random generators, etc.).
- Serious mistake: "Real mathematics has no effects on war."
  (G.H. Hardy, 1940)
- Vendors have realized that **security** is an essential **purchase criterion**.

Demonstration in CrypTool

- **Statistic Analysis**
- **Encrypting twice is not always better:**

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

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

  - "Hund" + "Katze" ="RUGCLENWGYXDATRNHNMH")
- **Vernam (OTP)**
- **AES (output key, brute-force analysis)**

Content

I. CrypTool and Cryptology – Overview

II. CrypTool Features

III. Examples

IV. Project / Outlook / Contact

Appendix

CrypTool Features

eLearning

1. What is CrypTool?

- Freeware program with graphical user interface
- Cryptographic methods can be applied *and* analysed
- Comprehensive online help (understandable without a deep knowledge of cryptography)
- Contains nearly all state-of-the-art cryptography functions
- Easy entry into modern and classical cryptography
- Not a *“hacker tool”*

2. Why CrypTool?

- Originated in an awareness initiative of a financial institute
- Developed in close cooperation with universities
- Improvement of university education and in-firm training

3. Target group

- *Core group*: Students of computer science, business computing, and mathematics
- *But also for*: computer users, application developers, employees, high school students, etc.
- *Prerequisite*: PC knowledge
- *Preferable*: Interest in mathematics and/or programming

Content of the Program Package

English, German, Polish, Spanish, and Serbian

CrypTool program
- All functions integrated in a *single* program with consistent graphical interface
- Runs on Win32
- Includes cryptography libraries from Secude, cryptovision, and OpenSSL
- Long integer arithmetic via Miracl, APFLOAT and GMP/MPIR, lattice-based reduction via NTL (V. Shoup)

AES Tool
- Standalone program for AES encryption (and creation of self extracting files)

Educational game
- “Number Shark” encourages the understanding of factors and prime numbers.

Comprehensive Online Help (HTML Help)
- Context-sensitive help available via F1 for all program functions (including menus)
- Detailed use cases for most program functions (tutorial)

Script (.pdf file) with background information
- Encryption methods • Prime numbers and factorization • Digital signatures
- Elliptic curves • Public-key certification • Basic number theory • Crypto 2020 • Sage

Two short stories related to cryptography by Dr. C. Elsner
- “The Dialogue of the Sisters” (features an RSA variant as key element)
- “The Chinese Labyrinth” (number theory tasks for Marco Polo)

Learning tool for number theory

Features (1)

Cryptography

Classical cryptography

- Caesar (and ROT-13)
- Monoalphabetic substitution (and Atbash)
- Vigenère
- Hill
- Homophone substitution
- Playfair
- ADFGVX
- Byte Addition
- XOR
- Vernam
- Permutation / Transposition (Rail Fence, Scytale, etc.)
- Solitaire

Several options to easily comprehend cryptography samples from literature

- Selectable alphabet
- Options: handling of blanks, etc.

Cryptanalysis

Attack on classical methods

- Ciphertext only
  - Caesar
  - Vigenère (according to Friedman + Schroedel)
  - Addition
  - XOR
  - Substitution
  - Playfair
- Known Plaintext
  - Hill
  - Single-column transposition
- Manual (program supported)
  - Mono alphabetical substitution
  - Playfair, ADFGVX, Solitaire

Supported analysis methods

- Entropy, floating frequency
- Histogram, n-gram analysis
- Autocorrelation
- Periodicity
- Random analysis
- Base64 / UU-Encode

Features (2)

Cryptography

Modern symmetric encryption

- IDEA, RC2, RC4, RC6, DES, 3DES, DESX
- AES candidates of the last selection round (Serpent, Twofish, etc.)
- AES (=Rijndael)
- DESL, DESXL

Asymmetric encryption

- RSA with X.509 certificates
- RSA demonstration
  - For improved understanding of examples from literature
  - Alphabet and block length selectable

Hybrid encryption (RSA + AES)

- Visualized as an interactive data flow diagram

Cryptanalysis

Brute-force attack on symmetric algorithms

- For all algorithms
- Assumptions:
  - Entropy of plaintext is small,
  - Key is partially known, or
  - Plaintext alphabet is known

Attack on RSA encryption

- Factorization of RSA modulus
- Lattice-based attacks

Attack on hybrid encryption

- Attack on RSA, or
- Attack on AES (side-channel attack)

Features (3)

Cryptography

Digital signature

- RSA with X.509 certificates
  - Signature as data flow diagram
- DSA with X.509 certificates
- Elliptic Curve DSA, Nyberg-Rueppel

Hash functions

- MD2, MD4, MD5
- SHA, SHA-1, SHA-2, RIPEMD-160

Random generators

- Secude
- x2 mod n
- Linear congruence generator (LCG)
- Inverse congruence generator (ICG)

Cryptanalysis

Attack on RSA signature

- Factorization of the RSA module
- Feasible up to 250 bits or 75 decimal places (on standard desktop PCs)

Attack on hash functions / digital signature

- Generate hash collisions for ASCII based text (birthday paradox) (up to 40 bits in about five minutes)

Analysis of random data

- FIPS-PUB-140-1 test battery
- Periodicity, Vitányi, entropy
- Floating frequency, histogram
- n-gram analysis, autocorrelation
- ZIP compression test

Features (4)

Visualizations / Demos

- Caesar, Vigenère, Nihilist, DES (all with ANIMAL)
- Enigma (Flash)
- Rijndael/AES (two versions with Flash, one with Java)
- Hybrid encryption and decryption (AES-RSA and AES-ECC)
- Generation and verification of digital signatures
- Diffie-Hellman key exchange
- Secret sharing (with CRT or Shamir)
- Challenge-response method (network authentication)
- Side-channel attack
- Secure email with the S/MIME protocol (with Java and Flash)
- Graphical 3D presentation of (random) data streams
- Sensitivity of hash functions regarding plaintext modifications
- Number theory and RSA cryptosystem (with Authorware)

Features (5)

Additional functions

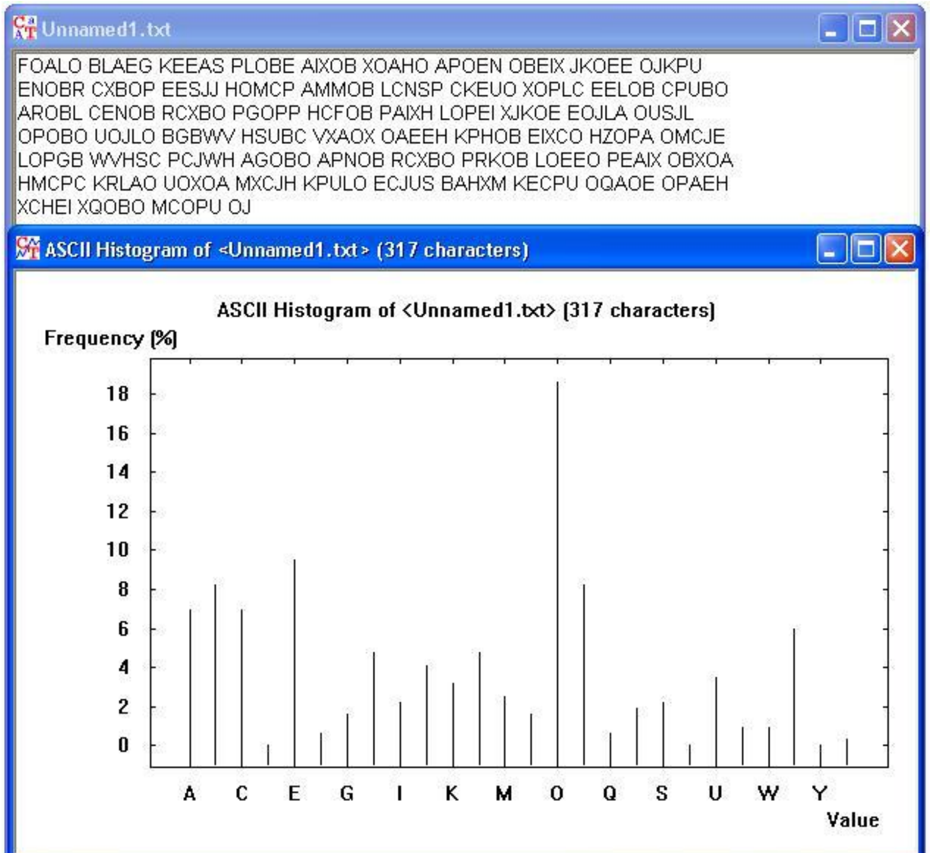
- Different functions for RSA and prime numbers
- Homophone and permutation encryption (Double Column Transposition)
- PKCS #12 import and export for PSEs (Personal Security Environment)
- Hash generation of large files (without loading them)
- Flexible brute force attacks on any modern symmetric algorithm
- ECC demonstration (as Java application)
- Password Quality Meter (PQM) and password entropy
- Manifold text options for the classic ciphers (see example 24)
- And plenty more…

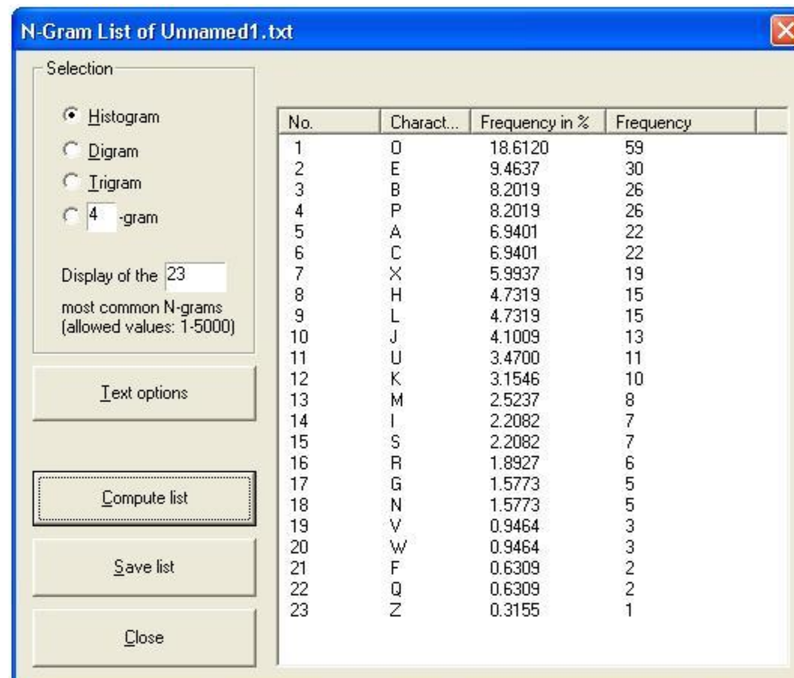
Language Structure Analysis

Language analysis options available in CrypTool

Number of characters, n-gram, entropy

- See menu “Analysis” \ “Tools for Analysis” \ ...





Demonstration of Interactivity (1)

Vigenère analysis

Demonstration in CrypTool

The result of the Vigenère analysis can be manually reworked (changing the key length)

1. Encrypt the sample file with **TESTETE**
   - *"Crypt/Decrypt" \ "Symmetric (classic)" \ "Vigenère"*
   - Enter TESTETE ⇨ *"Encrypt"*

   Analysis of the encryption results:
   - *"Analysis" \ "Symmetric Encryption (classic)" \ "Ciphertext only" \ "Vigenère"*
   - Derived key length 7, derived key TESTETE ✓
2. Encrypt starting sample with **TEST**
   - *"Crypt/Decrypt" \ "Symmetric (classic)" \ "Vigenère"*
   - Enter TEST ⇨ *"Encrypt"*

   Analysis of the encryption results:
   - *"Analysis" \ "Symmetric Encryption (classic)" \ "Ciphertext only" \ "Vigenère"*
   - Derived key length 8 – incorrect ✗
   - Key length automatically set to 4 (can also be adjusted manually)
   - Derived key TEST ✓

Demonstration of Interactivity (2)

Automated factorization

Demonstration in CrypTool

Factorization of a compound number with factorization algorithms

- The algorithms are executed in parallel (multi-threaded)
- Each algorithm has specific advantages and disadvantages; for example, some methods can only determine small factors

Factorization example 1

48-digit decimal number

316775895367314538931177095642205088158145887517
=
3 * 1129 * 6353 * 1159777 * 22383173213963 * 567102977853788110597

Factorization example 2

75-digit decimal number

2^250 - 1
=
3 * 11 * 31 * 251 * 601 * 1801 * 4051 * 229668251 * 269089806001 * 4710883168879506001 * 5519485418336288303251

Menu: "Indiv. Procedure" \ "RSA Cryptosystem" \ "Factorization of a Number"

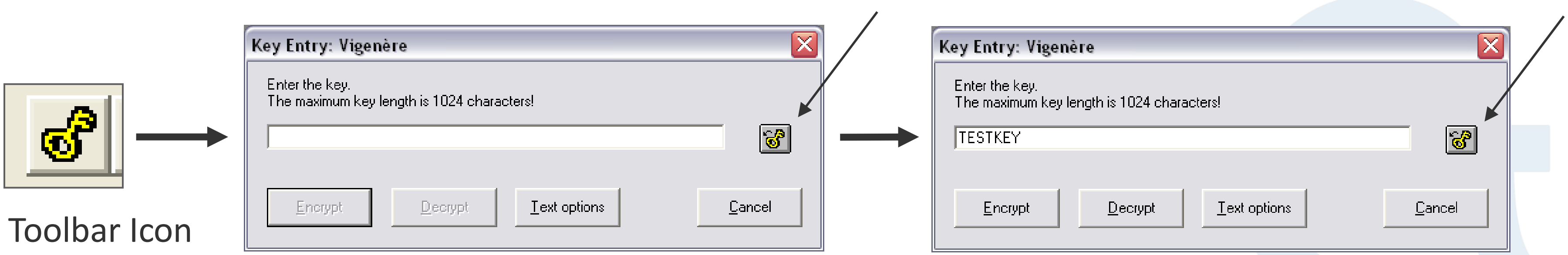
Concepts for a User-Friendly Interface

1. Context sensitive help (F1)

- F1 on a selected menu entry shows information about the algorithm/method.
- F1 in a dialog box explains the usage of the dialog.
- These assistants and the contents of the top menus are cross-linked in the online help.

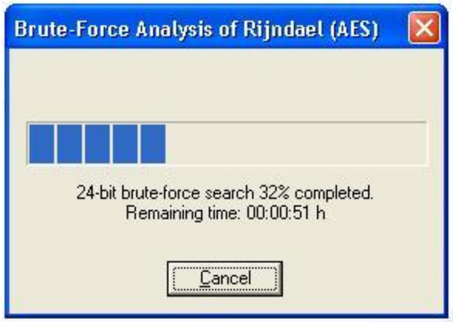
2. Copying keys to the key entry dialog

- CTRL-V can always be used to paste contents from the clipboard.
- Stored keys can be copied from ciphertext windows via an icon in the toolbar. A corresponding icon in the key entry dialog can be used to paste the key into the key field. CrypTool uses an **internal keystore**, which is available for every method of the program. (This is particularly helpful for large “specific” keys, such as in homophone encryption.)



Challenges for Developers (Examples)

1. **Allow additional functions to run in parallel**
   - Factorization already uses multi-threading to run several algorithms at once
2. **High performance**
   - Locate hash collisions (birthday paradox) or perform brute force analysis
3. **Consider memory limits**
   - In particular with regard to the Floyd algorithm (mappings to locate hash collisions) and quadratic sieve factorization
4. **Time measurement and estimation**
   - Display remaining time (e.g. while using brute force)



5. **Reusability / Integration**
   - Forms for prime number generation
   - RSA cryptosystem (switches the view after successful attack from public key user to private key owner)
6. **Partially automate the consistency of functions, GUI, and online help**
   (including different languages and the supported Windows OSes: XP, Vista, Win7)

Content

CrypTool Examples

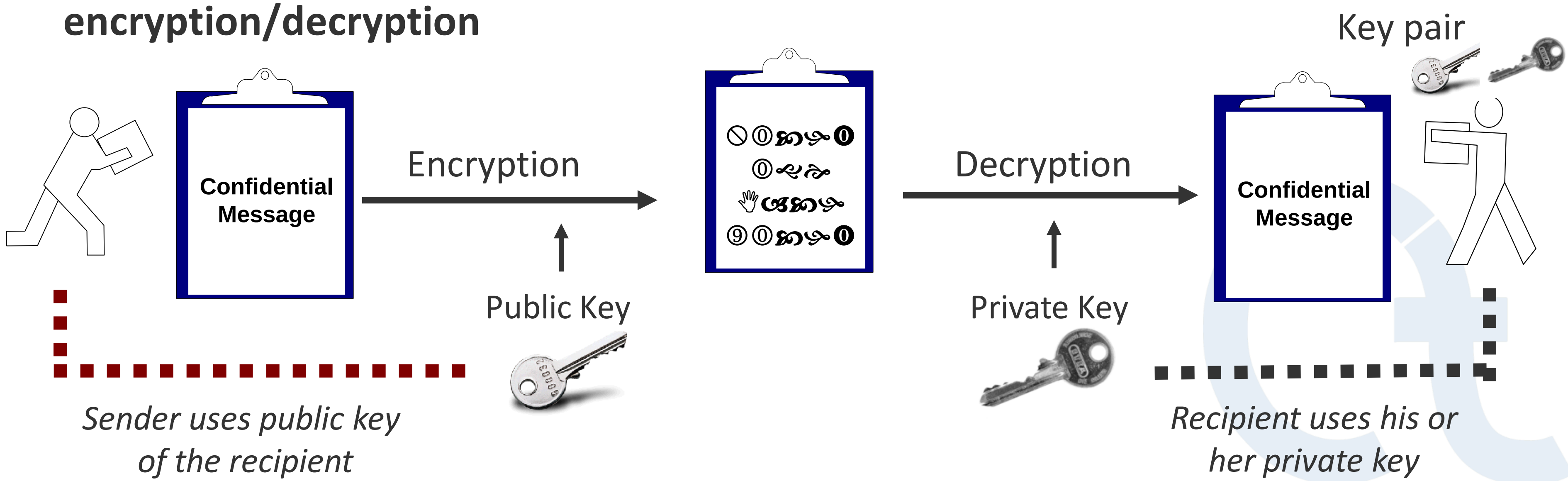
Overview of examples

1. Encryption with RSA / Prime number tests / Hybrid encryption and digital certificates / SSL
2. Digital signature visualized
3. Attack on RSA encryption (small modulus N)
4. Analysis of encryption in PSION 5
5. Weak DES keys
6. Locating key material (“NSA key”)
7. Attack on digital signature through hash collision search
8. Authentication in a client-server environment
9. Demonstration of a side-channel attack (on hybrid encryption protocol)
10. Attack on RSA using lattice reduction
11. Random analysis with 3-D visualization
12. Secret Sharing using the Chinese Remainder Theorem (CRT) and Shamir
13. Implementation of CRT in astronomy (solving systems of linear modular equations)
14. Visualization of symmetric encryption methods using ANIMAL
15. Visualizations of AES
16. Visualization of Enigma encryption
17. Visualization of Secure Email with S/MIME
18. Generation of a message authentication code (HMAC)
19. Hash demonstration
20. Educational tool for number theory and asymmetric encryption
21. Point addition on elliptic curves
22. Password quality meter (PQM) and password entropy
23. Brute force analysis
24. Scytale / Rail Fence
25. Hill encryption / Hill analysis
26. CrypTool online help / Menu tree of the program

Examples (1)

Encryption with RSA

- **Basis of the SSL protocol (access to protected websites), among others**
- **Asymmetric encryption using RSA**
  - Every user has a key pair – one public and one private key.
  - Sender encrypts with public key of the recipient.
  - Recipient decrypts with his or her private key.
- **Usually implemented in combination with symmetric methods (hybrid encryption): the symmetric key is transmitted using RSA asymmetric encryption/decryption**



Examples (1)

Encryption using RSA – Mathematical background / algorithm

- Public key: (n, e) [the modulus N is often capitalized]
- Private key: (d)

where

p, q are large, randomly chosen prime numbers with $n = p*q$;

d is calculated under the constraints $gcd[\varphi(n),e] = 1$; $e*d \equiv 1 \bmod \varphi(n)$.

Encryption and decryption operation: $(m^e)^d \equiv m \bmod n$

- n is the modulus (its length in bits is referred to as the key length of RSA).
- gcd = greatest common divisor.
- $\varphi(n)$ is Euler's totient function.

Procedure

- Transform the message into its binary representation
- Encrypt message blockwise such that $m = m_1,...,m_k$ where for all m_j: $0 \leq m_j < n$;
  The maximum block size r should be chosen such that $2^r \leq n$ (and $2^r-1 < n$)

Hint: Interactive Flash animation about the basics of the RSA cipher:
http://cryptool.com/download/RSA/RSA-Flash-en/player.html

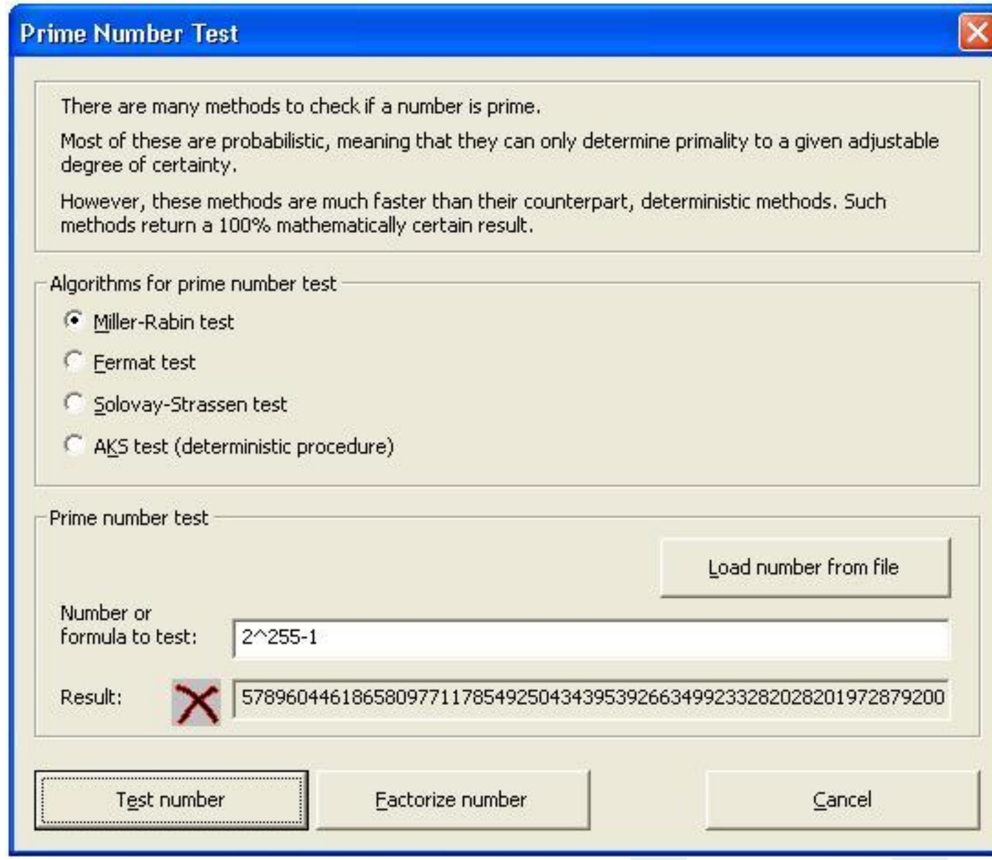
Examples (1)

Prime number tests – RSA requires the use of very large primes

- Fast probabilistic tests
- Deterministic tests

The prime number test methods can test whether a large number is prime much faster than the known factorization methods can divide a number of similar size into its prime factors.

For the AKS test the GMP / MPIR library (**G**NU **M**ultiple **P**recision Arithmetic Library; **M**ultiple **P**recision **I**ntegers and **R**ationals) was integrated into CrypTool.



Menu: "Indiv. Procedures" \ "RSA Cryptosystem" \ "Prime Number Test"

Remark: 2^255 - 1 = 7 * 31 * 103 * 151 * 2143 * 11119 * 106591 * 131071 * 949111 * 9520972806333758431 * 5702451577639775545838643151

Examples (1)

Printing of the current prime number records – Mersenne primes

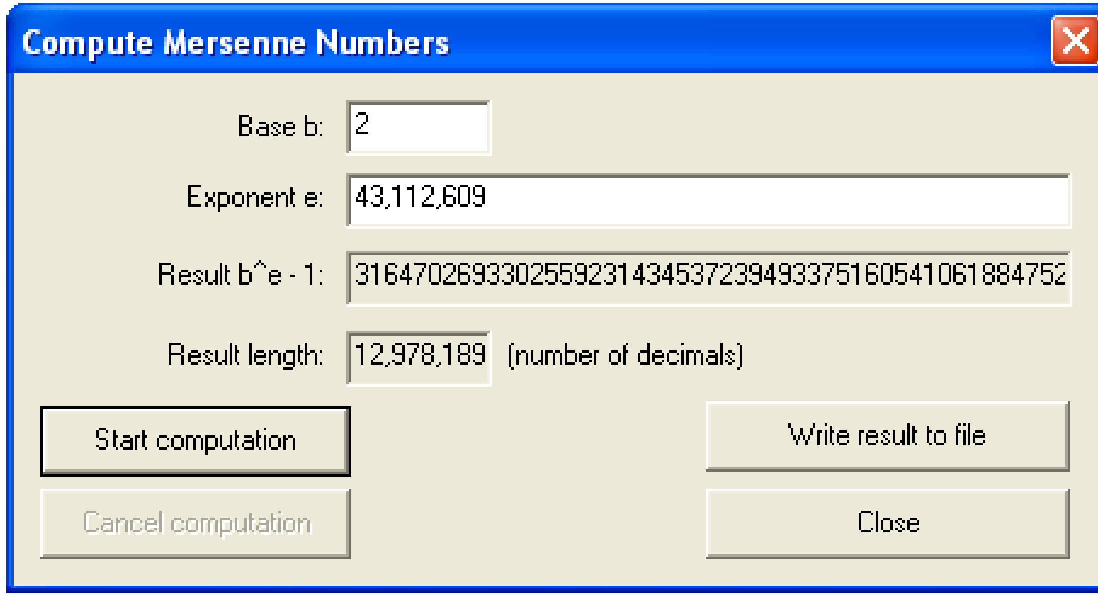
The biggest known primes are so called Mersenne primes.

The currently biggest one has 12,978,189 decimal digits and was discovered in 2008 by the GIMPS project.

The adjoining dialog allows to calculate and write all digits of such numbers very fast.

To do so the APFLOAT library was integrated into CrypTool.

Within the context menu of each input or output field of this dialog you can switch on and off the thousands separator.



Remark: 2^43,112,609 - 1 = 316,470,269 … 697,152,511

Large numbers should not be marked and copied from the "Result" field – because of the performance of the GUI. Please use the button "Write result to file" in order to show the resulting number in its completeness within the CrypTool main window.

Menu: "Indiv. Procedures" \ "Number Theory – Interactive" \ "Compute Mersenne Numbers"

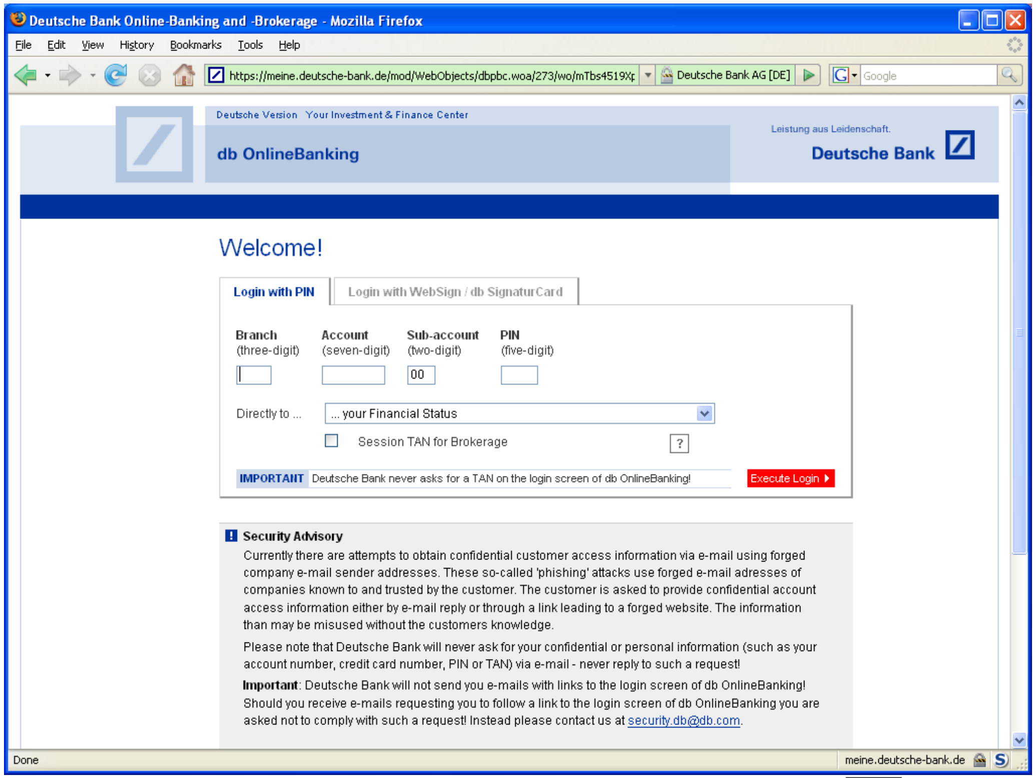
Examples (1)

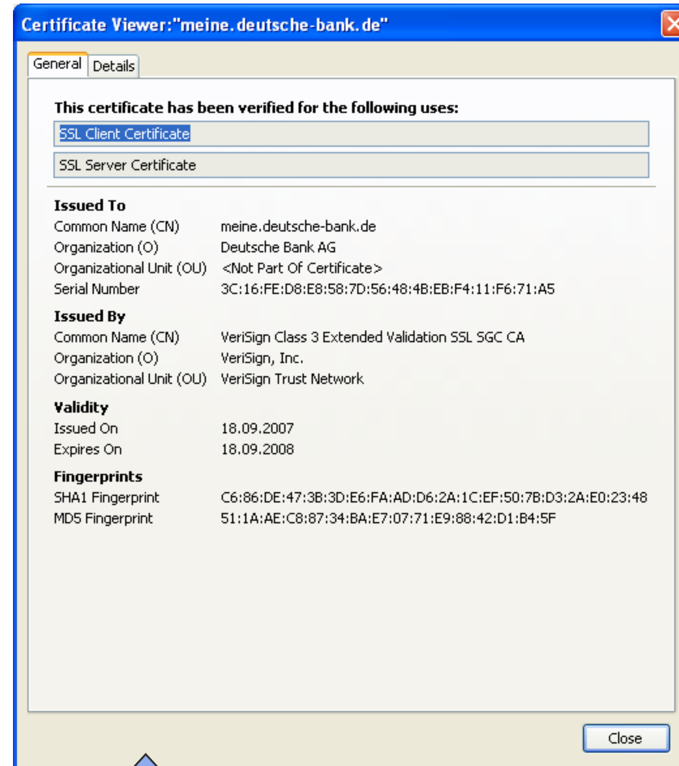
Hybrid encryption and digital certificates

- Hybrid encryption **– combination of asymmetric and symmetric encryption**
  1. Generation of a random symmetric key (session key)
  2. Session key is transferred – protected by asymmetric key
  3. Message is transferred – protected by session key
- Problem**: man-in-the-middle attacks – does the public key of the recipient really belong to the recipient?**
- Solution: digital certificates **– a central instance (e.g., GlobalSign, Telesec, VeriSign, Deutsche Bank PKI), trusted by all users, ensures the authenticity of the certificate and the associated public key (similar to a passport issued by a national government).**
- Hybrid encryption based on digital certificates **is the foundation for all secured electronic communication**
- Internet shopping and online banking
- Secure email

Examples (1)

Secured online connection using SSL and certificates

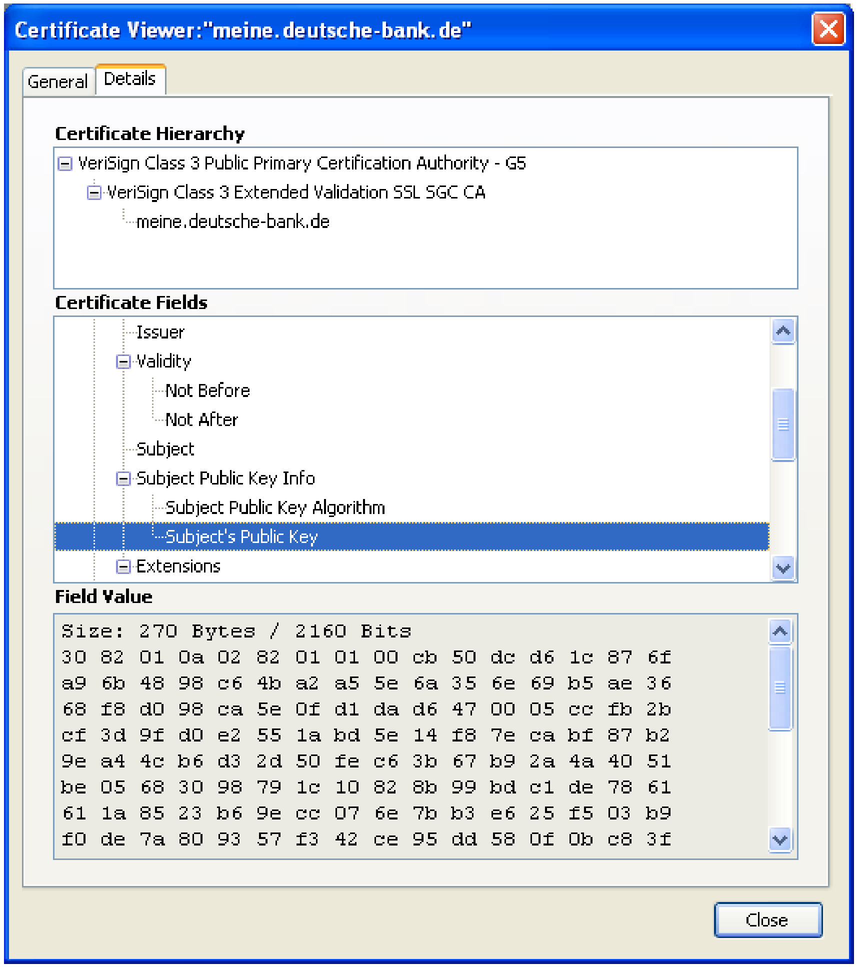




This means that the connection is authenticated (at least on one side) and that the transferred data is strongly encrypted.

Examples (1)

Attributes / fields of a certificate



General attributes / fields

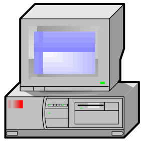
- Issuer (e.g., VeriSign)
- Requestor
- Validity period
- Serial number
- Certificate type / version (X.509v3)
- Signature algorithm
- Public key (and method)

Public Key

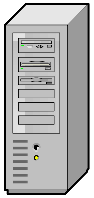
Examples (1)

Establishing a secure SSL connection (server authentication)

Client



Server



1. SSL initiation

2. Send server certificate

3. Validate server certificate (using locally installed root certificates)

4. Retrieve public key of server (from server certificate)

5. Generate a random symmetric key (session key)

6. Send session key (encrypted with public key of server)

7. Receive session key (decrypted by private key of the server)

Encrypted communication based on exchanged session key

Examples (1)

Establishing a secure SSL connection (server authentication)

General

- The example shows the typical SSL connection establishment in order to transfer sensitive data over the internet (e.g. online shopping).
- During SSL connection establishment only the server is authenticated using a digital certificate (authentication of the user usually occurs through user name and password after the SSL connection has been established).
- SSL also offers the option for client authentication based on digital certificates.

Remarks on establishing an SSL connection (see previous slide)

- Step 1: SSL Initiation – the characteristics of the session key (e.g. bit size) as well as the symmetric encryption algorithm (e.g. 3DES, AES) are negotiated.
- Step 2: In a multi-level certificate hierarchy, the required intermediate certificates are also passed to the client.
- Step 3: The root certificates installed in the browser's certificate store are used to validate the server certificate.
- Step 5: The session key is based on the negotiated characteristics (see step 1).

Examples (2)

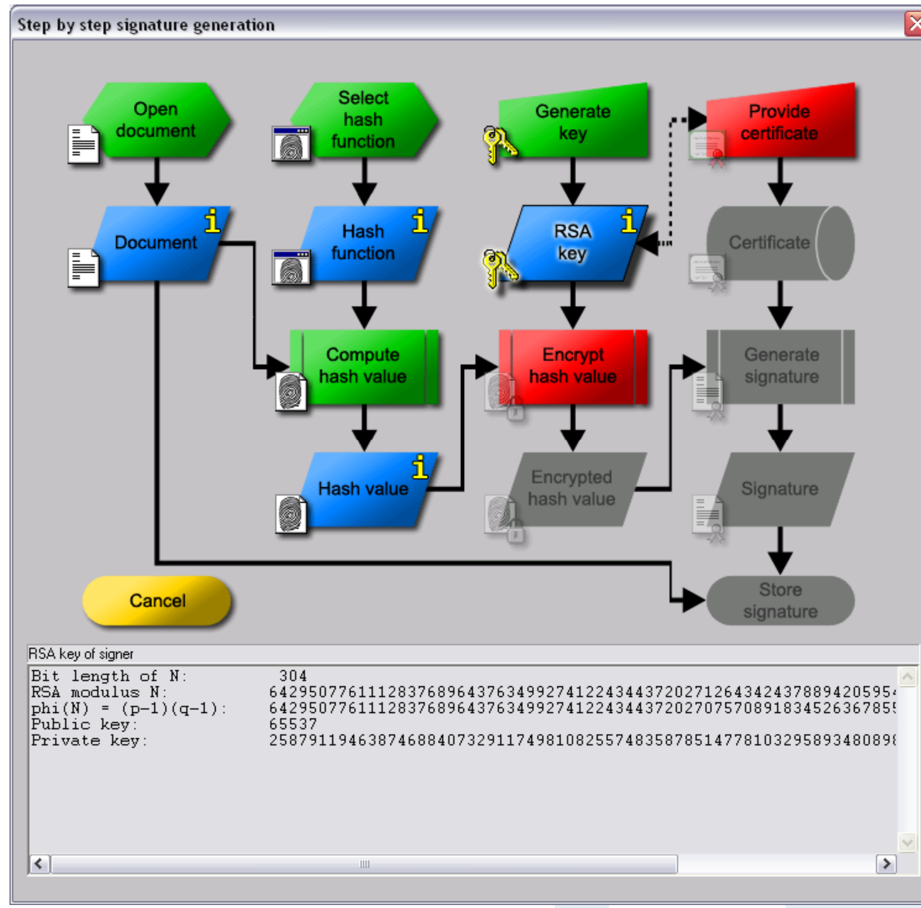
Digital signature visualized

Digital signature

- Increasingly important
  - Equivalent to a handwritten signature (digital signature law)
  - increasingly used by companies, governments, and consumers
- Few actually know how it works

Visualization in CrypTool

- Interactive data flow diagram
- Similar to the visualization of hybrid encryption



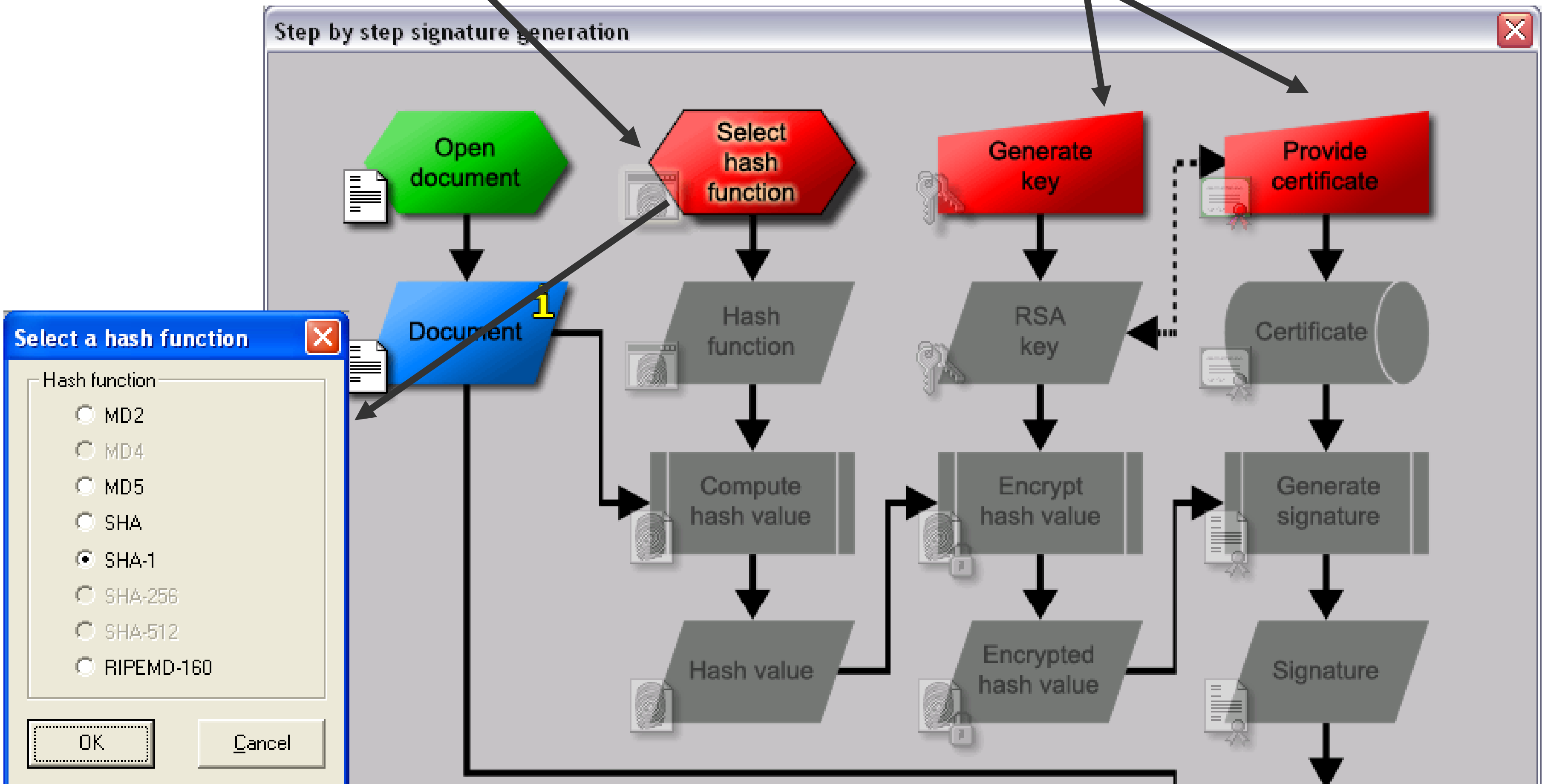
Menu: “Digital Signatures/PKI” \
“Signature Demonstration (Signature Generation)”

Examples (2)

Digital signature visualized: a) Preparation

1. Select hash function

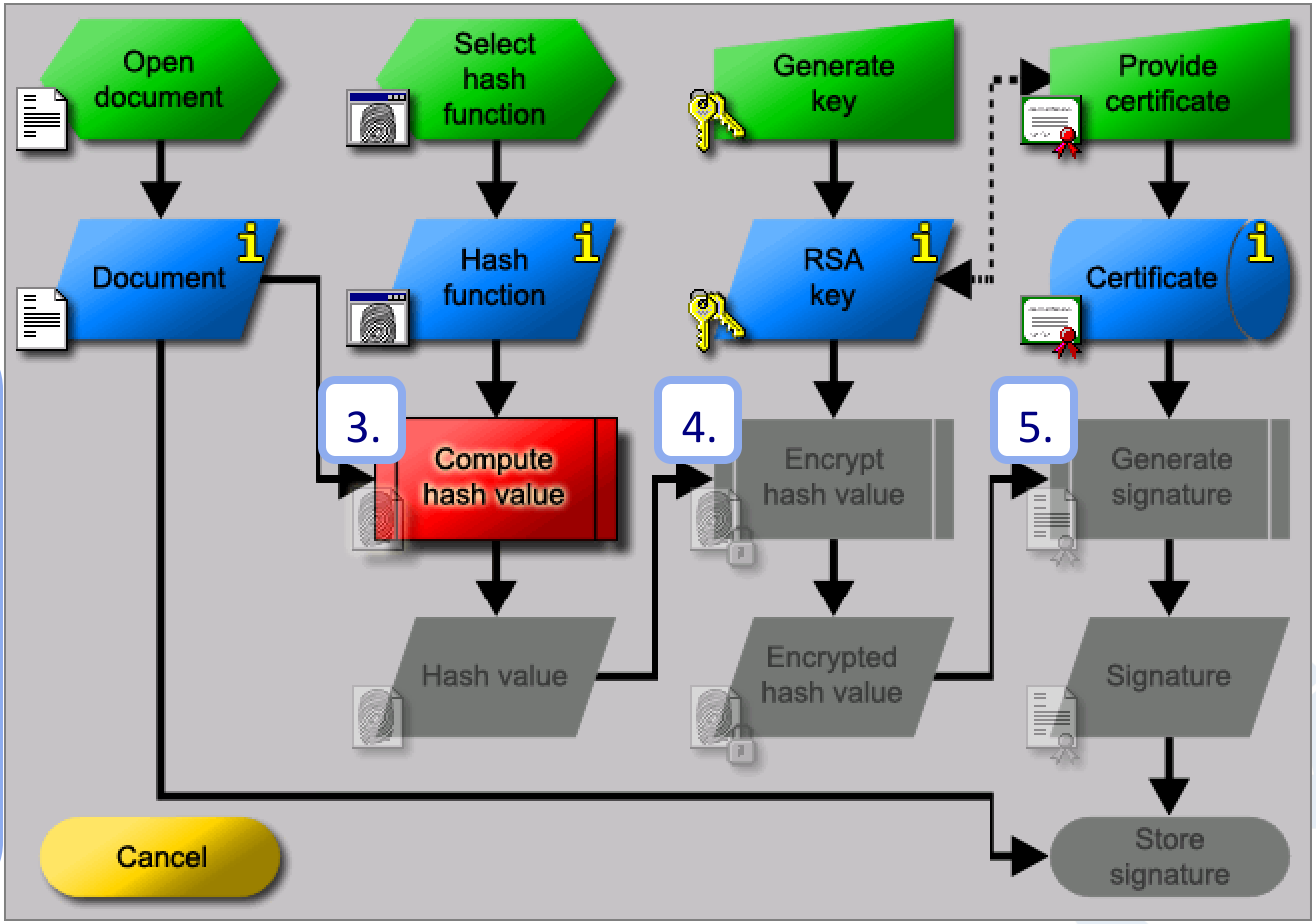
2. Provide key and certificate (not shown here)



Examples (2)

Digital signature visualized: b) Cryptography

3. Calculate hash value
4. Encrypt hash value with private key (sign)
5. Generate signature

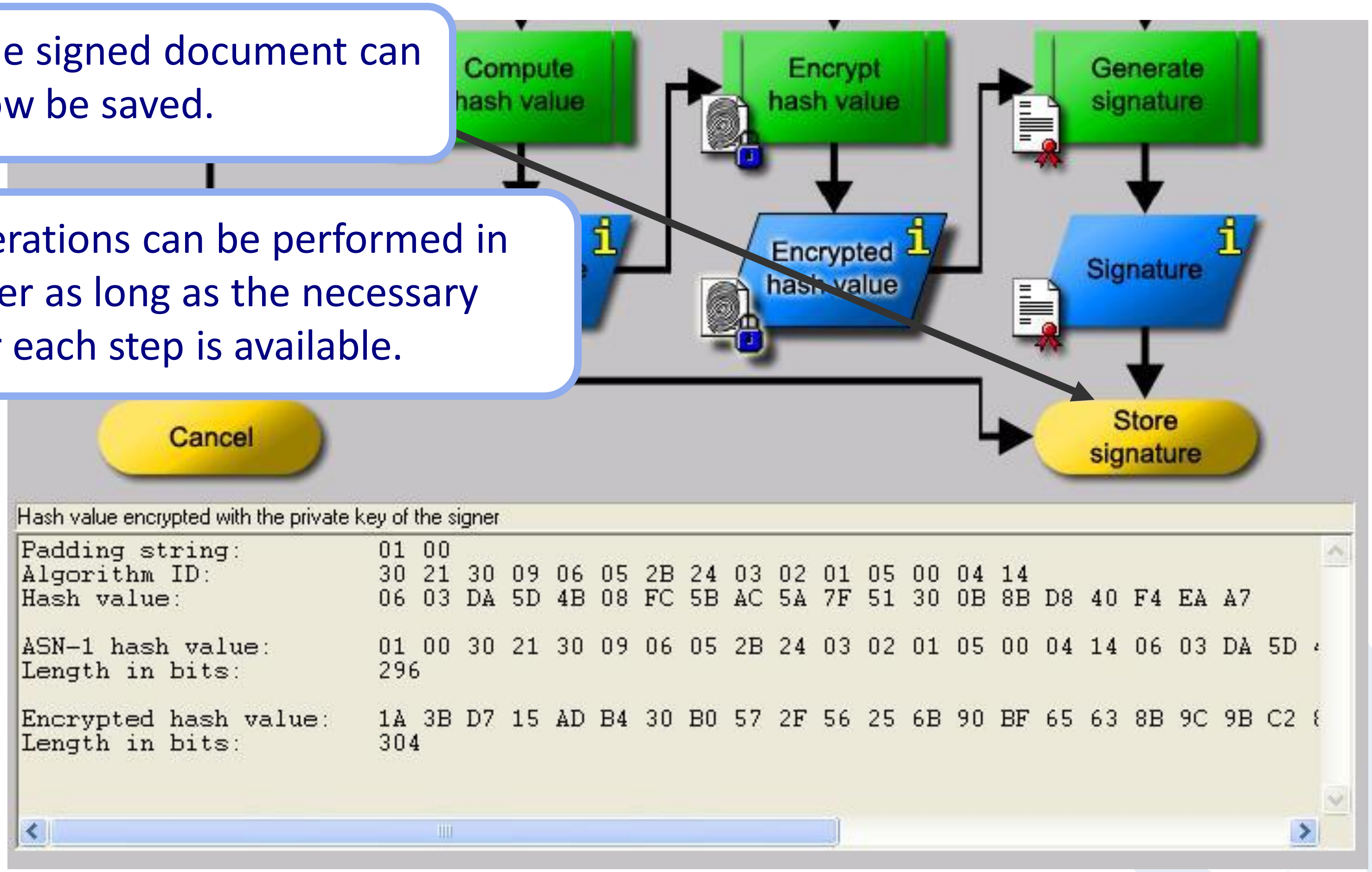


Examples (2)

Digital signature visualized: c) Result

6. The signed document can now be saved.

The operations can be performed in any order as long as the necessary data for each step is available.



Examples (3)

Attack on RSA encryption with short RSA modulus

Example from Song Y. Yan, *Number Theory for Computing*, Springer, 2000

- Public key
  - RSA modulus **N = 63978486879527143858831415041** (95 bits, 29 decimal digits)
  - public exponent **e = 17579**
- Ciphertext (block length = 8):

  C_1 = 45411667895024938209259253423,
  C_2 = 16597091621432020076311552201,
  C_3 = 46468979279750354732637631044,
  C_4 = 32870167545903741339819671379
- This text must be deciphered!

To perform the actual cryptanalysis (revealing the private key), the ciphertext is not actually necessary!

Solution using CrypTool (further details in the examples section of the online help)

- Enter public parameters into “RSA cryptosystem” (menu: “Indiv. Procedures”)
- Clicking the button “Factorize the RSA modulus” yields the two prime factors pq = N
- Based on that information the private exponent $d=e^{-1}$ mod (p-1)(q-1) can be determined
- Decrypt the ciphertext with d: $M_i = C_i^d$ mod N

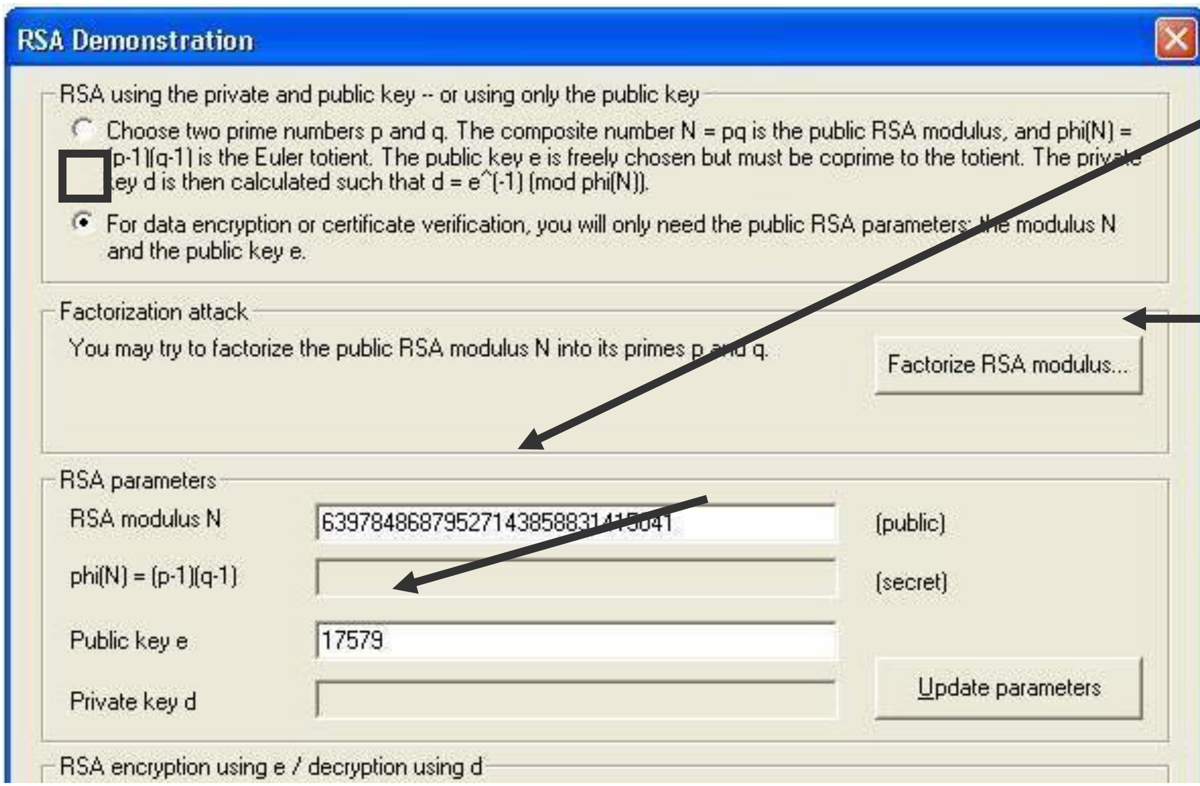
In CrypTool, this attack is only practical for RSA key sizes up to about 250 bits.

A successful attack means you could then digitally sign in someone else’s name!

Examples (3)

Short RSA modulus: Enter public RSA parameters

Menu: “Indiv. Procedures” \ “RSA Cryptosystem” \ “RSA Demonstration …”

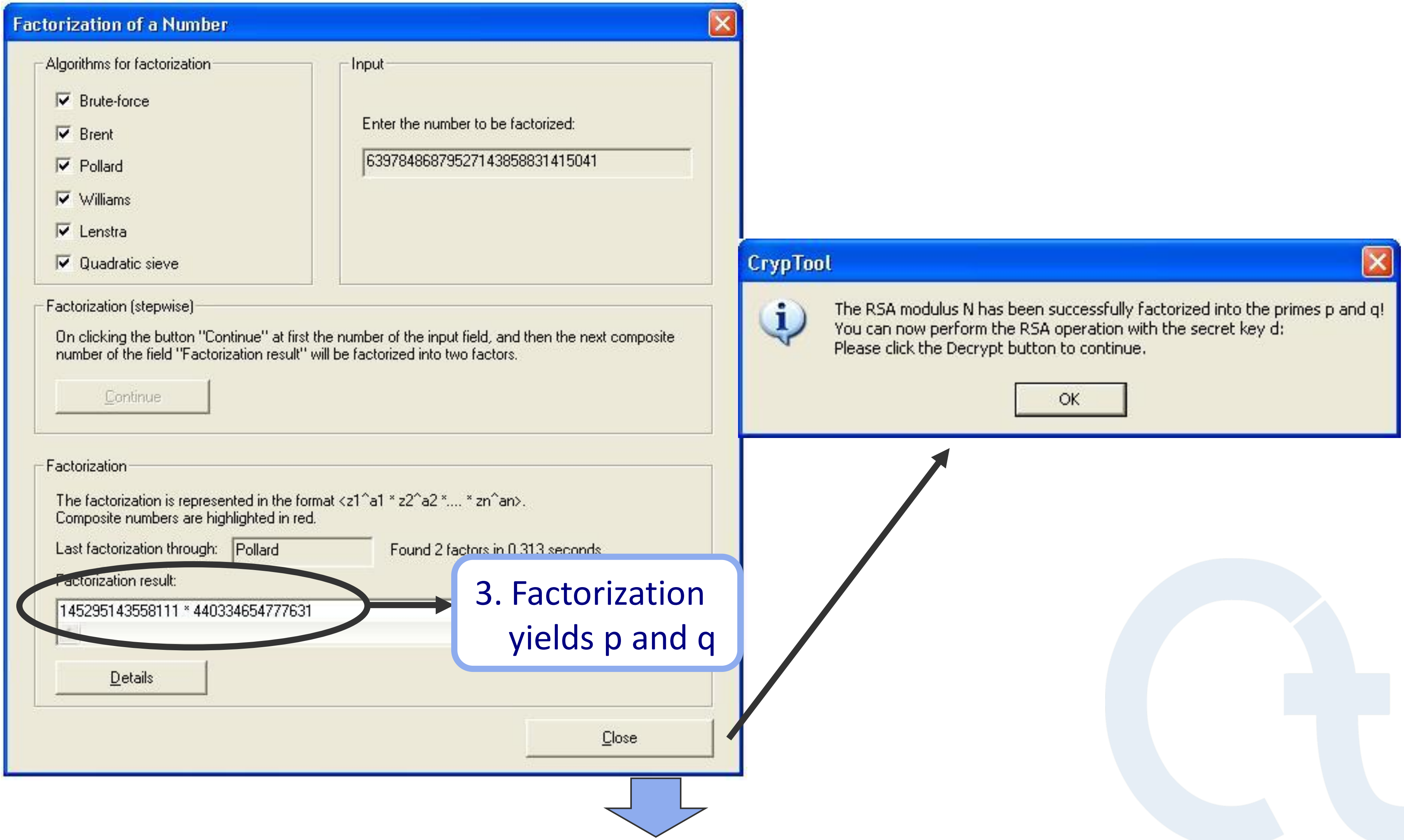


1. Enter RSA parameters N and e

2. Factorize

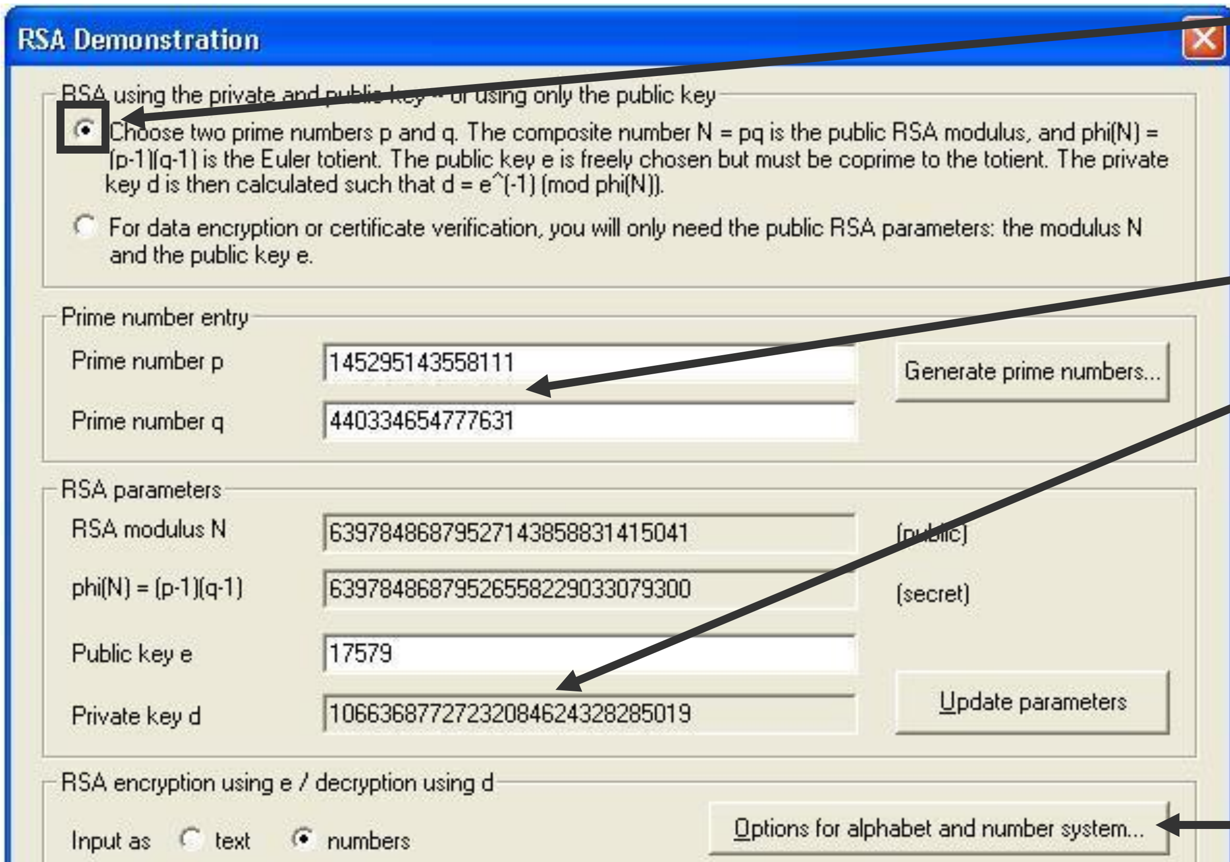
Examples (3)

Short RSA modulus: Factorize RSA modulus



Examples (3)

Short RSA modulus: Determine private key d



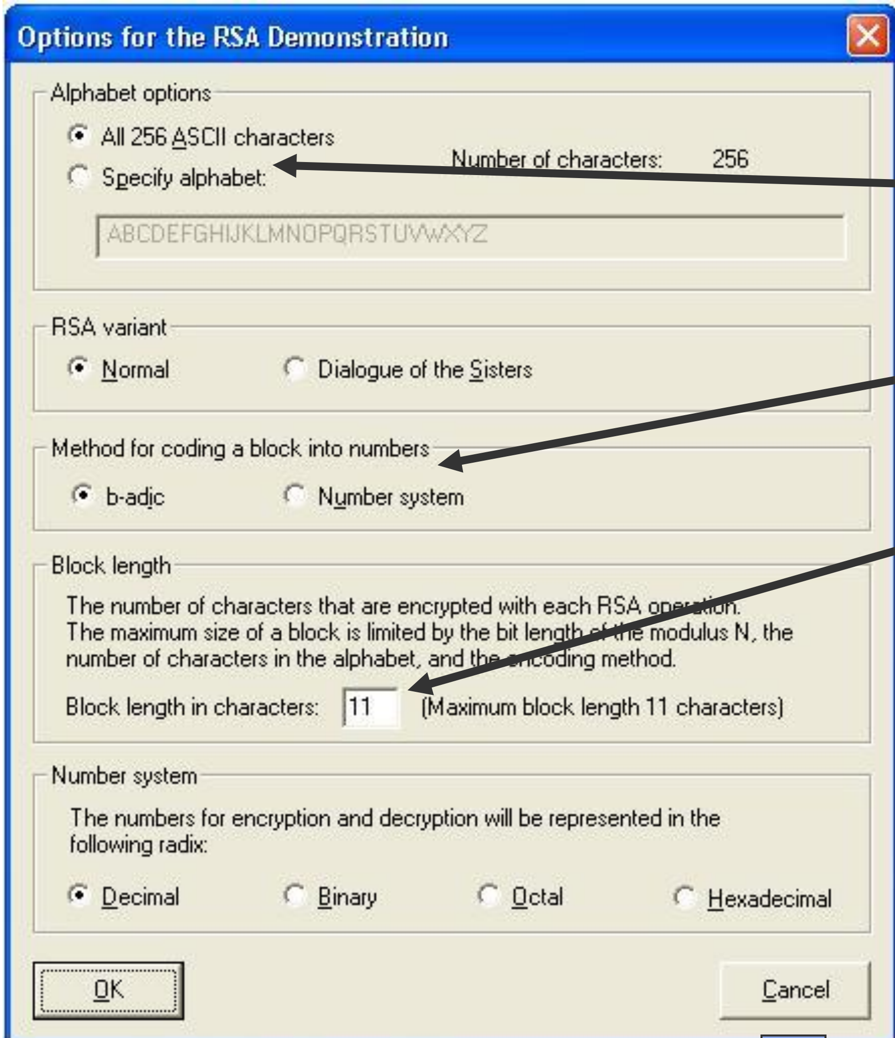
Change the view to the owner of the secret key

4. p and q have been entered automatically, and secret key d has been calculated

5. Change settings

Examples (3)

Short RSA modulus: Change settings



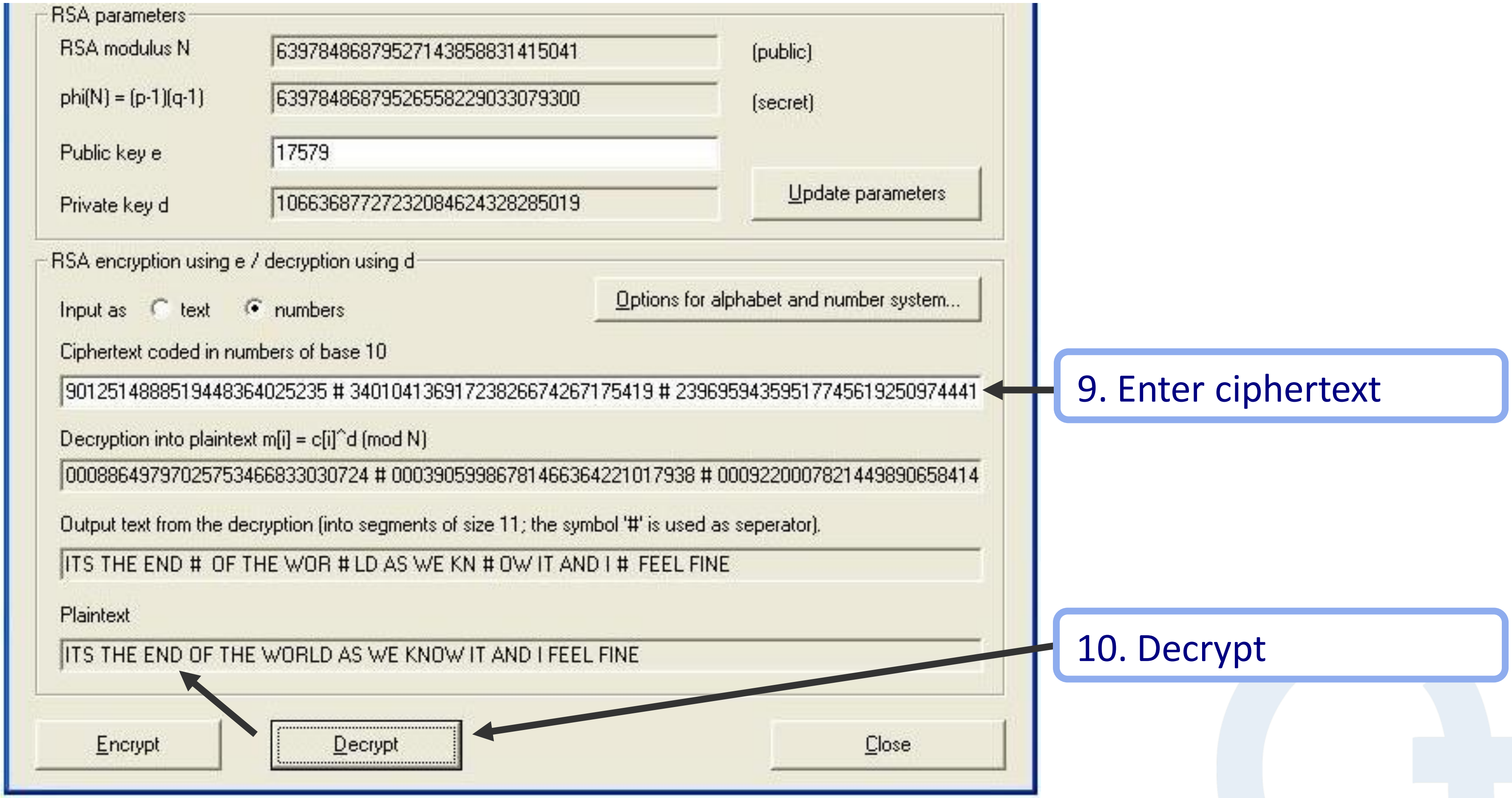
6. Select alphabet

7. Select coding method

8. Select block length

Examples (3)

Short RSA modulus: Decrypt ciphertext



Examples (4)

Analysis of encryption used in the PSION 5

Practical application of cryptanalysis

Attack on the encryption option in the PSION 5 PDA word processing application

Starting point: an encrypted file on the PSION

Requirements

- Encrypted English or German text
- Depending on method and key length, text of at least 100 bytes up to several kB

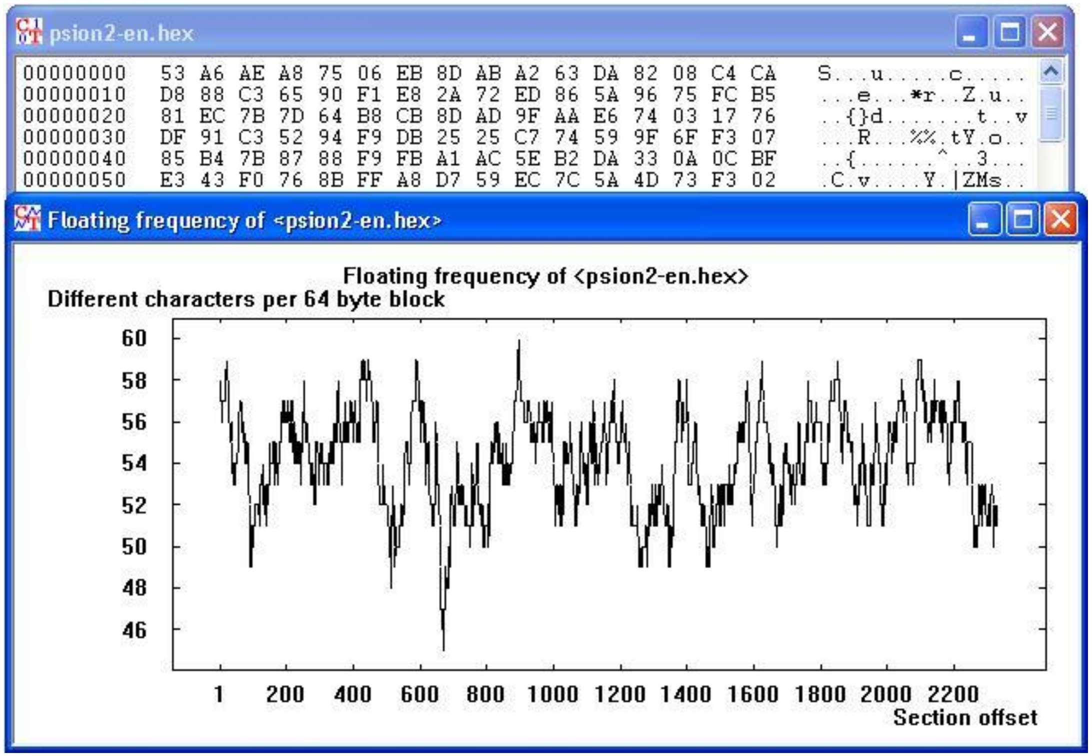
Procedure

- Pre-analysis
  - entropy
  - floating entropy
  - compression test
- Auto-correlation
- Automated analysis with classical methods

probably classical encryption algorithm

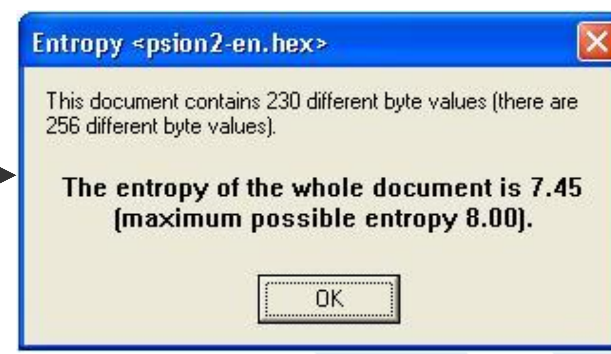
Examples (4)

PSION 5 PDA – determine entropy, compression test



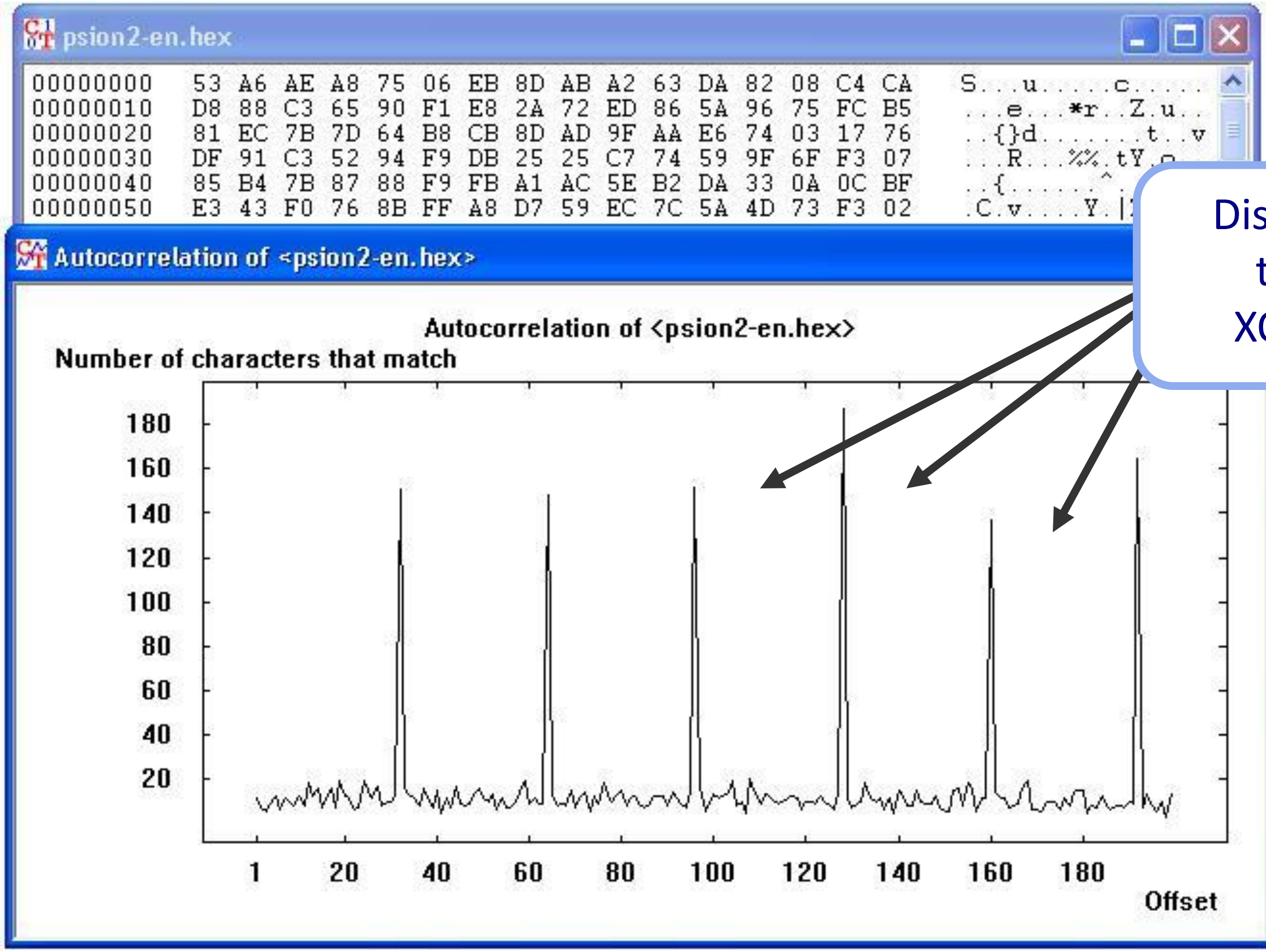
Compressibility: not indicative; a larger value would be a clear indication of weak cryptography

Entropy: not all possible values are present, but does not indicate a specific encryption method.



Examples (4)

PSION 5 PDA – determine auto-correlation



Distinctive comb pattern: typical for Vigenère, XOR, and byte addition

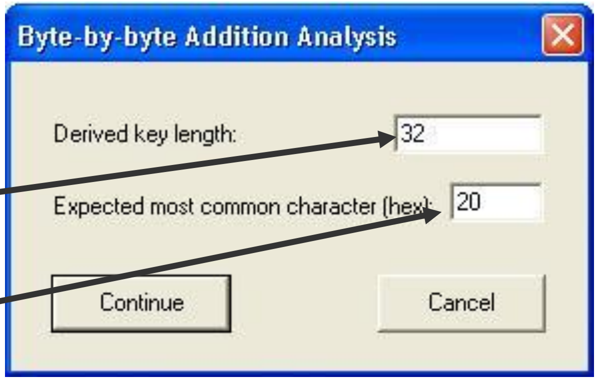
* The encrypted file is available in CrypTool (see CrypTool\examples\psion-en-enc.hex).

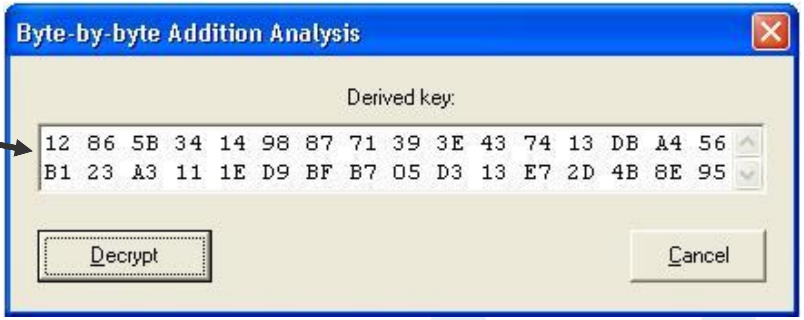
Examples (4)

PSION 5 PDA – automatic analysis

Automatic analysis using

- **Vigenère: no success**
- **XOR: no success**
- **Byte addition**
  - CrypTool calculates the key length using auto-correlation: 32 bytes
  - The user can choose which character is expected to occur most frequently: the empty space = 0x20 (ASCII code)
  - Analysis calculates the most likely key (based on assumptions regarding distribution)
  - Result: good, but not perfect



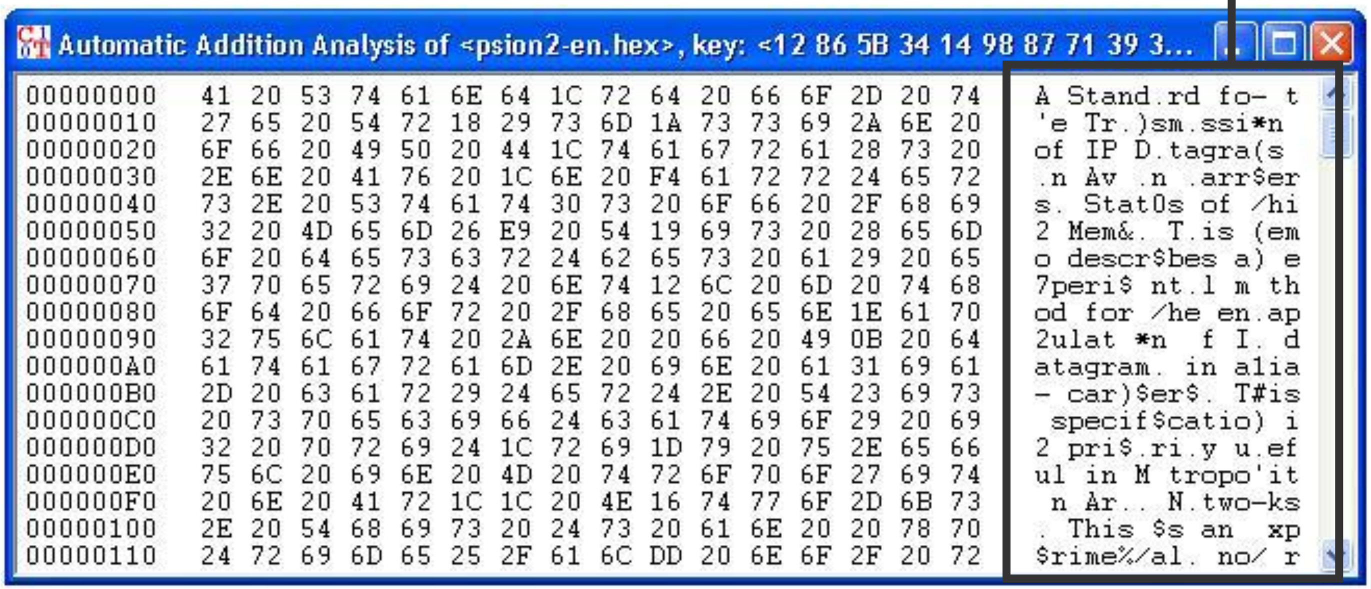


Examples (4)

PSION 5 PDA – results of automatic analysis

Results of automatic analysis under the assumption of “byte addition”

- Result is good, but not perfect: 25 out of 32 key bytes correct.
- The key length 32 was correctly determined.




```
00000000  41 20 53 74 61 6E 64 1C 72 64 20 66 6F 2D 20 74   A Stand.rd fo- t
00000010  27 65 20 54 72 18 29 73 6D 1A 73 73 69 2A 6E 20   'e Tr.)sm.ssi*n
00000020  6F 66 20 49 50 20 44 1C 74 61 67 72 61 28 73 20   of IP D.tagra(s
00000030  2E 6E 20 41 76 20 1C 6E 20 F4 61 72 72 24 65 72   .n Av .n .arr$er
00000040  73 2E 20 53 74 61 74 30 73 20 6F 66 20 2F 68 69   s. Stat0s of /hi
00000050  32 20 4D 65 6D 26 E9 20 54 19 69 73 20 28 65 6D   2 Mem&. T.is (em
00000060  6F 20 64 65 73 63 72 24 62 65 73 20 61 29 20 65   o descr$bes a) e
00000070  37 70 65 72 69 24 20 6E 74 12 6C 20 6D 20 74 68   7peri$ nt.l m th
00000080  6F 64 20 66 6F 72 20 2F 68 65 20 65 6E 1E 61 70   od for /he en.ap
00000090  32 75 6C 61 74 20 2A 6E 20 20 66 20 49 0B 20 64   2ulat *n  f I. d
000000A0  61 74 61 67 72 61 6D 2E 20 69 6E 20 61 31 69 61   atagram. in alia
000000B0  2D 20 63 61 72 29 24 65 72 24 2E 20 54 23 69 73   - car)$er$. T#is
000000C0  20 73 70 65 63 69 66 24 63 61 74 69 6F 29 20 69    specif$catio) i
000000D0  32 20 70 72 69 24 1C 72 69 1D 79 20 75 2E 65 66   2 pri$.ri.y u.ef
000000E0  75 6C 20 69 6E 20 4D 20 74 72 6F 70 6F 27 69 74   ul in M tropo'it
000000F0  20 6E 20 41 72 1C 1C 20 4E 16 74 77 6F 2D 6B 73    n Ar.. N.two-ks
00000100  2E 20 54 68 69 73 20 24 73 20 61 6E 20 20 78 70   . This $s an  xp
00000110  24 72 69 6D 65 25 2F 61 6C DD 20 6E 6F 2F 20 72   $rime%/al. no/ r
```

- The password entered was not 32 bytes long.
  → PSION Word derives the actual key from the password.
- Manual post-processing produces the encrypted text (not shown).

Examples (4)

PSION 5 PDA – determining the remaining key bytes

First, copy the key to the clipboard during automatic analysis.

Then, in the automatic analysis hex dump:

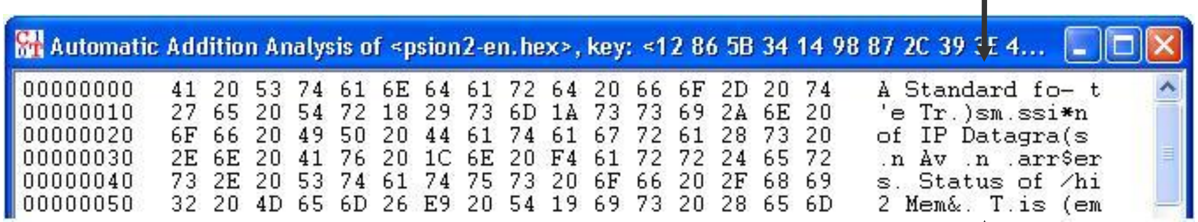
- Determine incorrect byte positions, e.g. 0x1C at position 8
- Guess and write down corresponding correct bytes: “a” = 0x61

Next, in the encrypted initial file hex dump:

- Determine initial bytes from the calculated byte positions: 0x8D
- Calculate correct key bytes with CALC.EXE: 0x8D - 0x61 = 0x2C

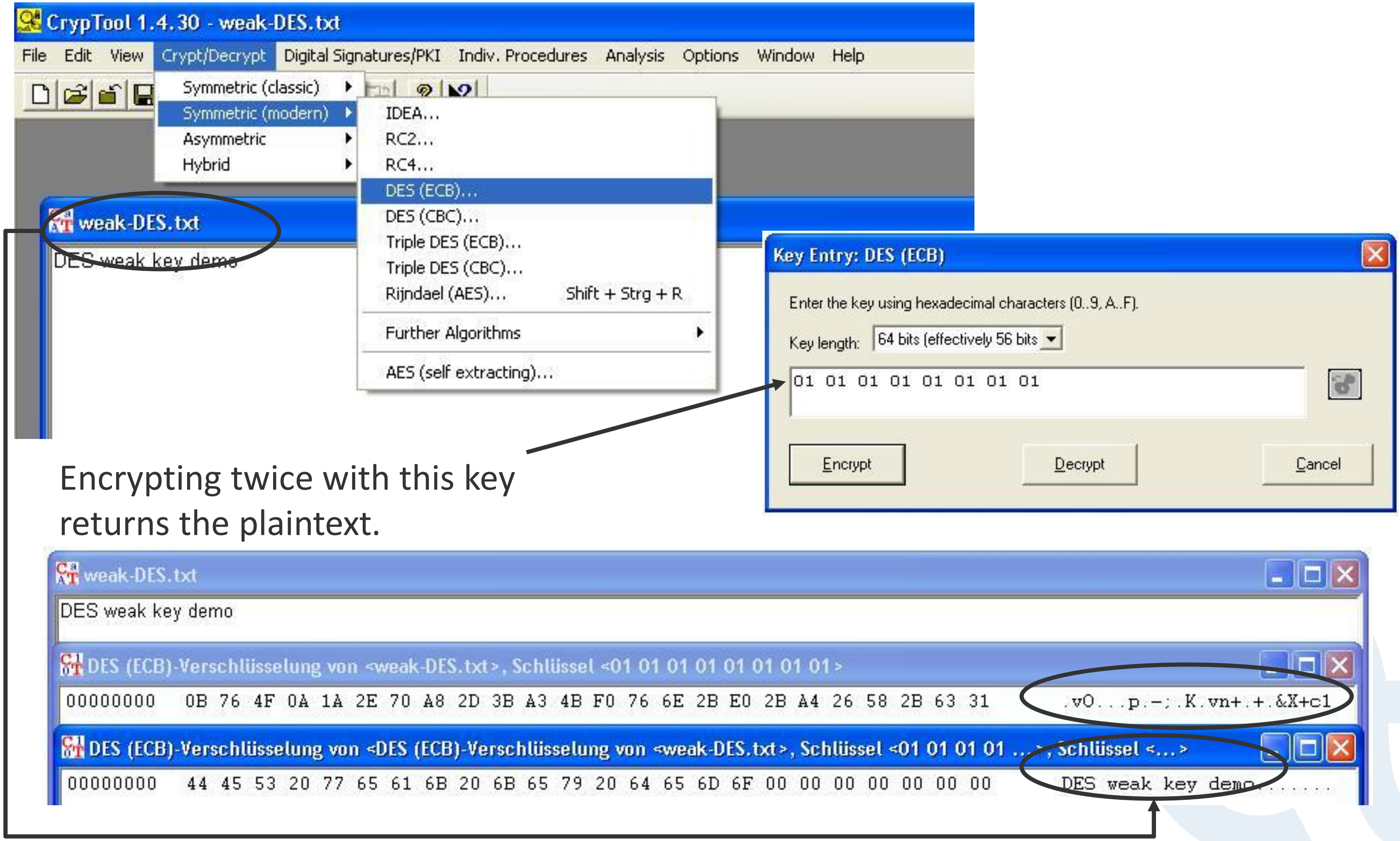
Finally, get the key from the clipboard:

- Correct 1286 5B34149887**2C**393E437413DBA456B123A3111ED9BFB705D313E72D4B8E95
- Decrypt encrypted initial document using byte addition
- Bytes at position 3, 3+32, 3+2*32, etc. are now correct



Examples (5)

Weak DES key



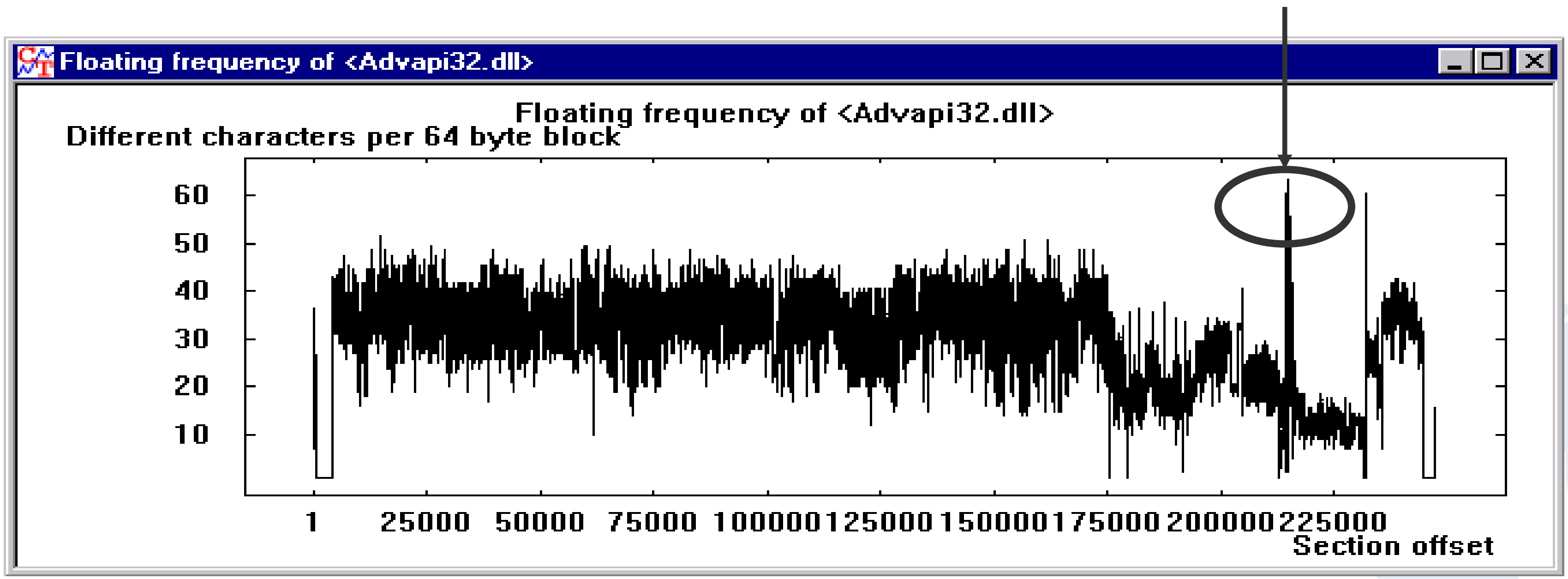
Examples (6)

Locate key material

The function “Floating frequency” is suitable for locating key material and encrypted areas in files.

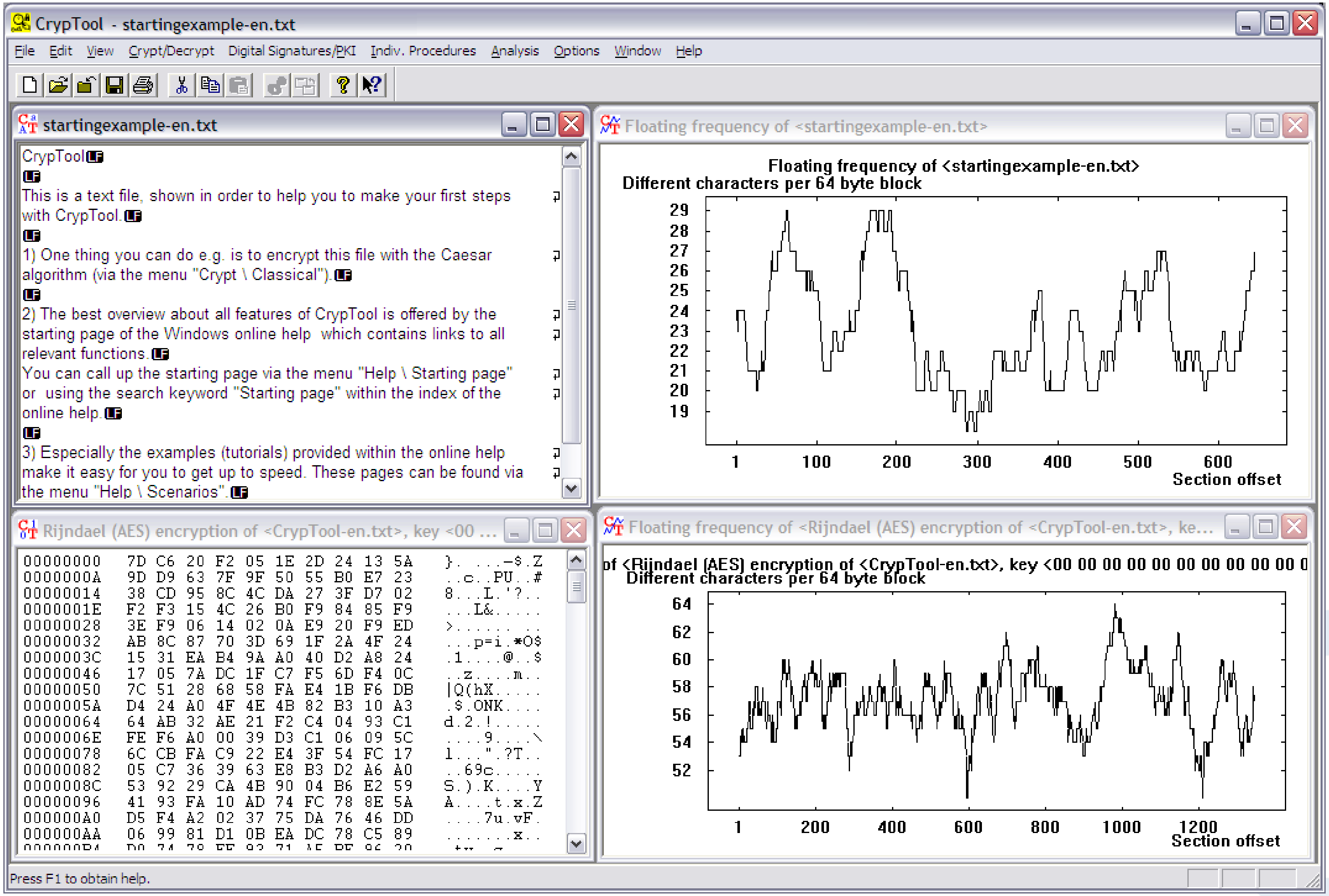
Background

- Key data is “more random” than text or program code
- Can be recognized as peaks in the “floating frequency”
- Example: the “NSA key” in advapi32.dll (Windows NT)



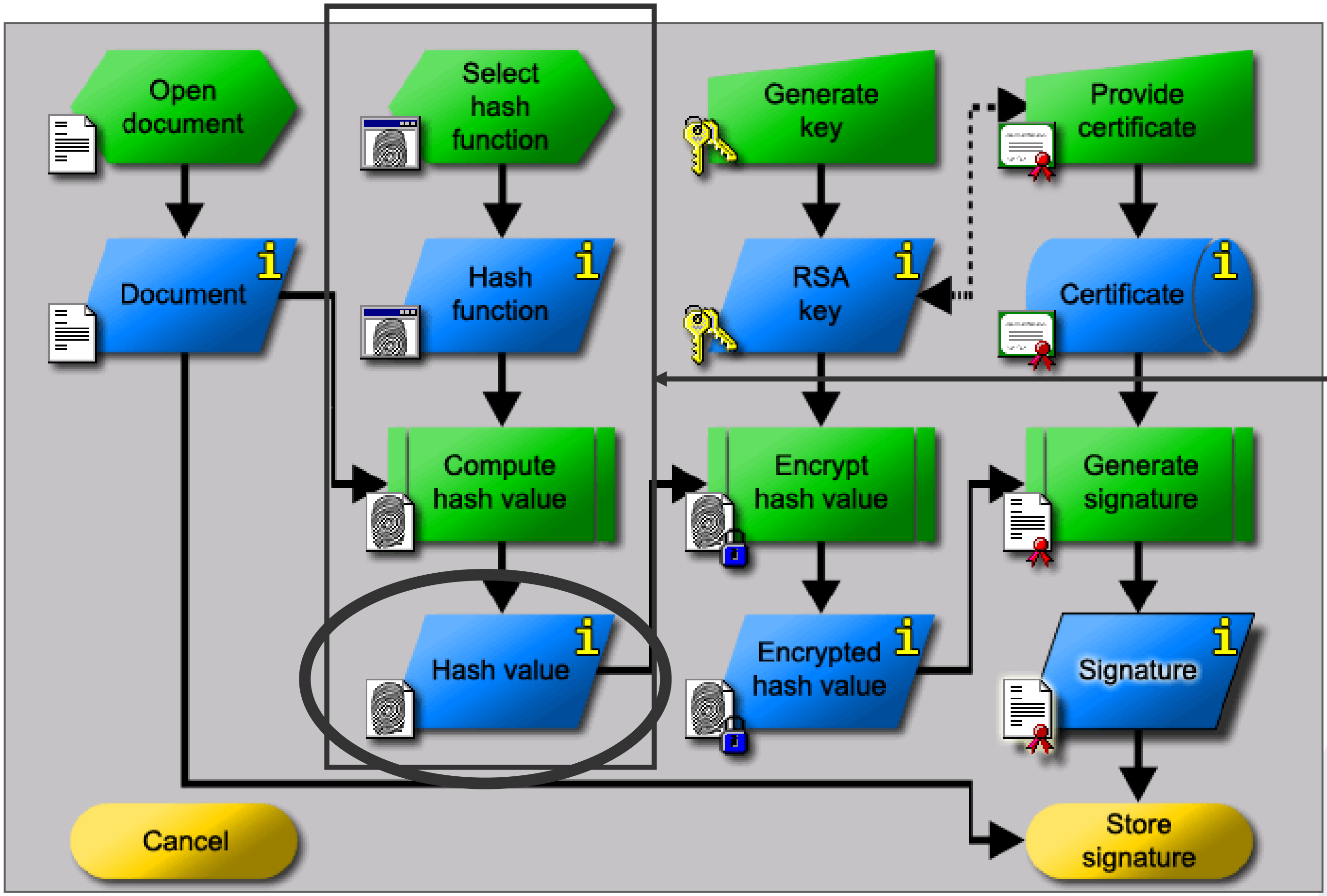
Examples (6)

Floating frequency comparison



Examples (7)

Attack on digital signatures



Attack

Find two messages with the same hash value!

Menu: “Analysis” \ “Hash” \ “Attack on the Hash Value of the Digital Signature”

Examples (7)

Attack on digital signature – idea (I)

Attack on the digital signature of an ASCII text by means of a hash collision search.

Idea:

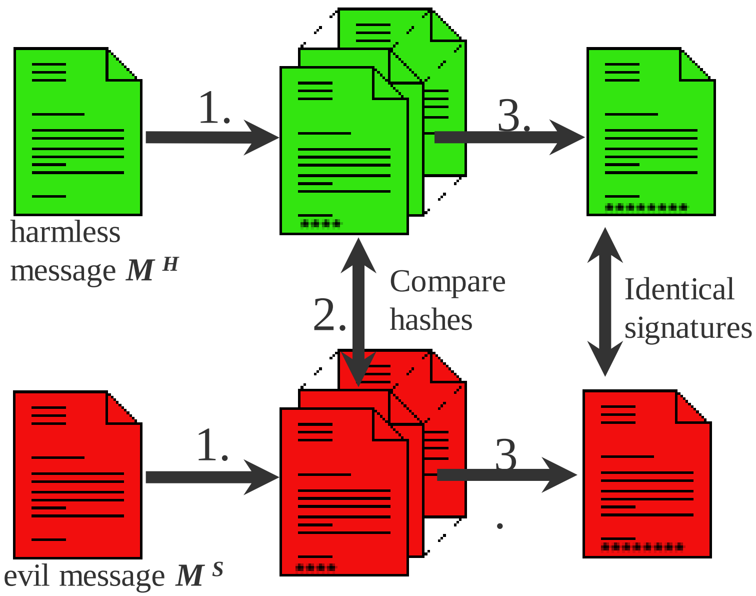
- ASCII texts can be modified by changing/inserting **non-printable** characters without changing the visible content
- Modify two texts in parallel until a hash collision is found
- Exploit the birthday paradox (birthday attack)
- Generic attack applicable to all hash functions
- Can parallelized across many machines (not implemented in CrypTool)
- Implemented in CrypTool as part of the bachelor thesis
  "Methods and Tools for Attacks on Digital Signatures" (German), 2003.

Concepts :

- Mappings
- Modified Floyd algorithm (constant memory consumption)

Examples (7)

Attack on digital signature – idea (II)



1. **Modification**: starting from a message M create N different messages $M_1, \ldots, M_N$ with the same "content" as M.
2. **Search:** find modified messages M_i^H and M_j^S with the same hash value.
3. **Attack:** the signatures of those two documents M_i^H and M_j^S are the same.

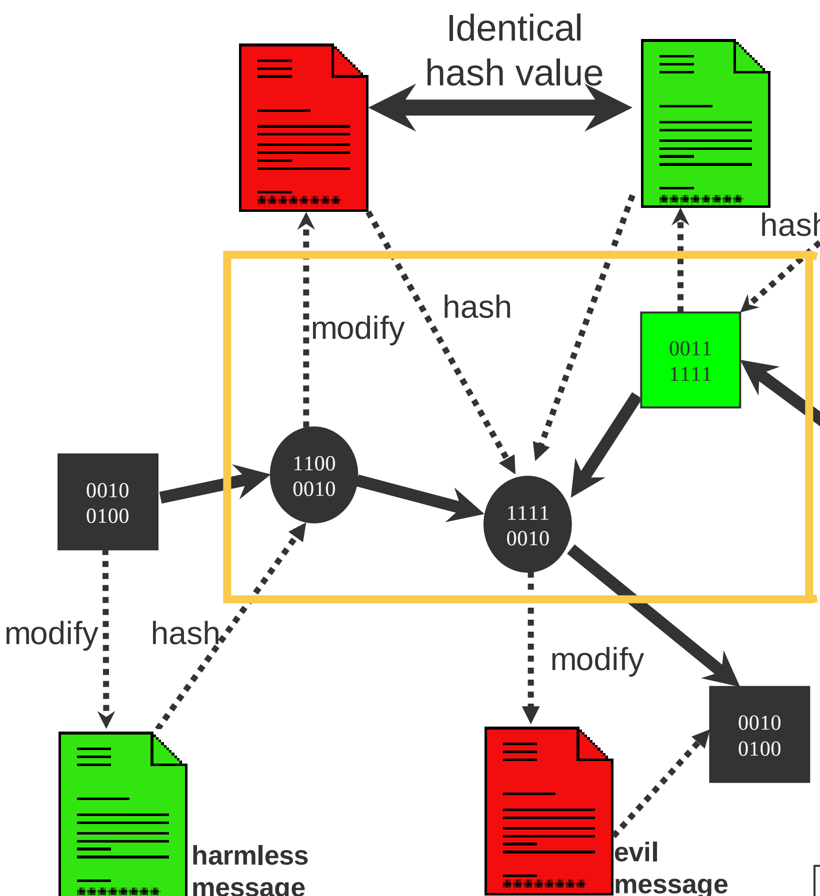
We know from the birthday paradox that for hash values of bit length n:

- search collision between M^H and $M_1^S, \ldots, M_N^S$: $N \approx 2^n$
- search collision between $M_1^H, \ldots, M_N^H$ and $M_1^S, \ldots, M_N^S$: $N \approx 2^{n/2}$

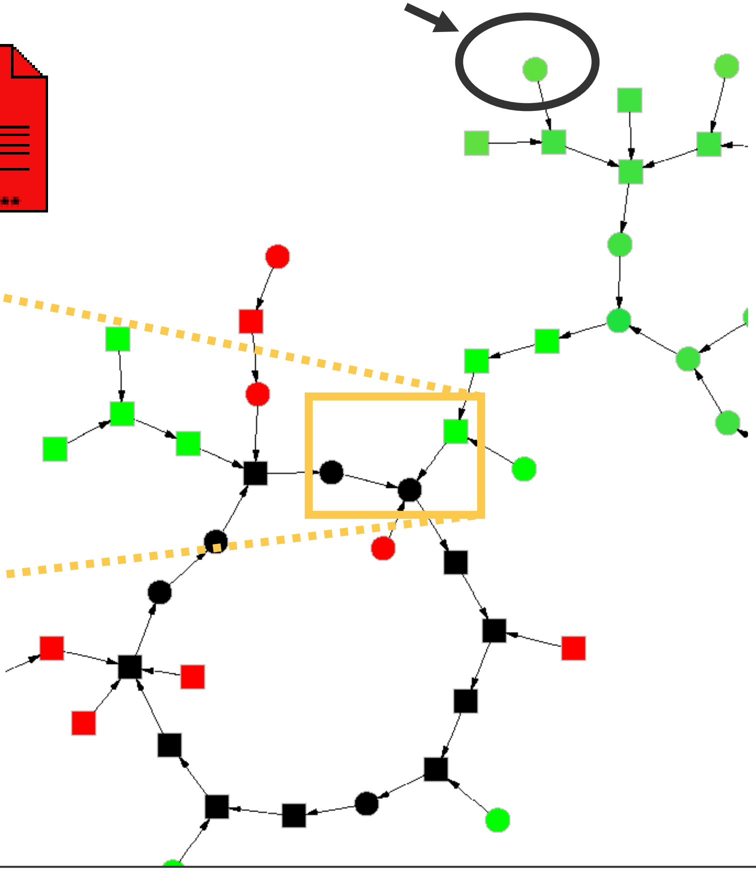
Estimated number of generated messages in order to find a hash collision.

Locate Hash Collisions (1)

Mapping via text modifications



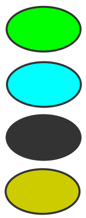
Randomly selected starting point for collisions search

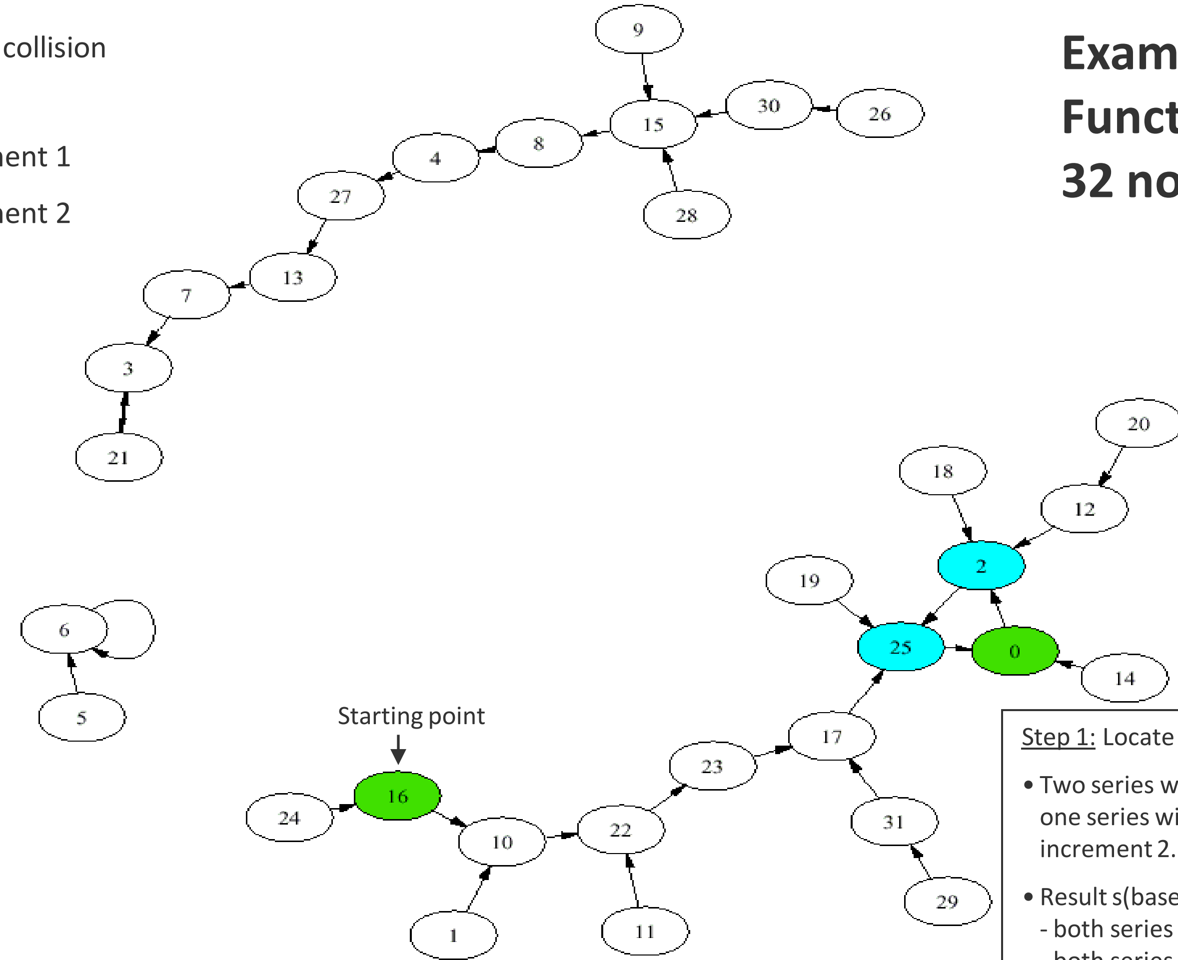


- green / red: path from a tree to the cycle – this can lead to a useful or useless collision, respectively.
- square / round: hash value has even / odd parity, respectively
- black: all nodes within the cycle

Locate Hash Collisions (2)

Floyd Algorithm: Meet within the cycle





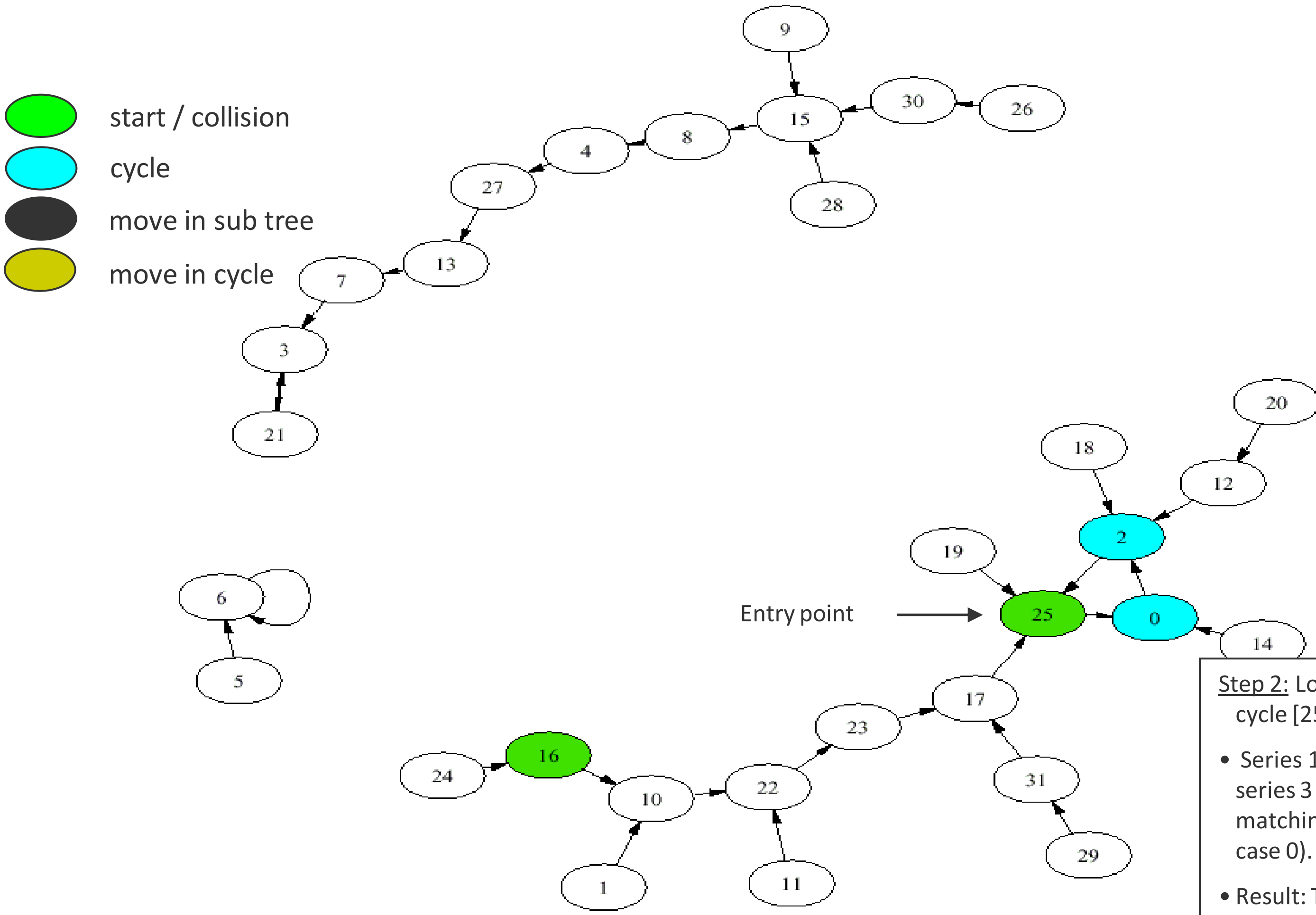
Example:
Function graph with 32 nodes

<u>Step 1:</u> Locate matching point within cycle:

- Two series with identical starting point [16]: one series with increment 1, the other with increment 2.
- Result s(based on graph theory):
  - both series always end up in a cycle.
  - both series match in a node within the cycle (in this case 0).

Locate Hash Collisions (3)

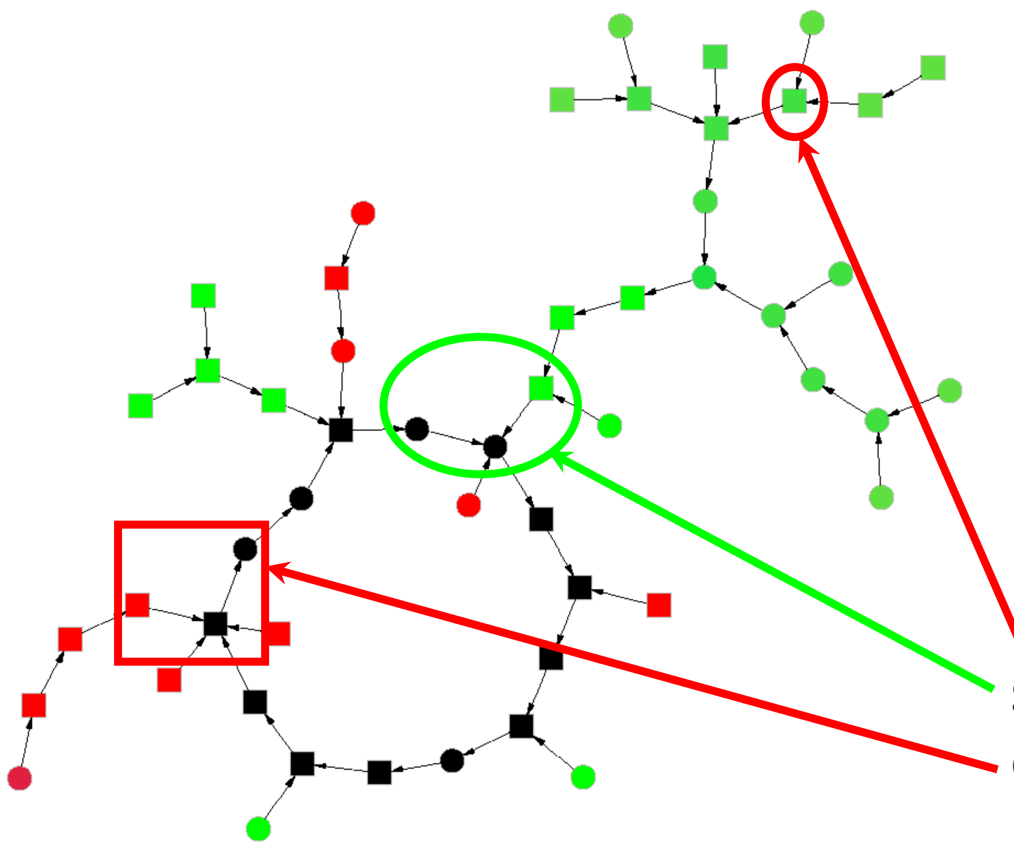
Step into cycle (extension of Floyd): Find entry point



Step 2: Locate entry point of series 1 in the cycle [25]:

- Series 1 starts again from starting point; series 3 with an increment of 1 starts at matching point within the cycle (in this case 0).
- Result: The series (1 and 3) match in cycle entry point of series 1 (in this case 25)
- The predecessors (in this case 17 and 2) result in a hash collision.

Birthday Paradox Attack on Digital Signature



Examination of Floyd algorithm

- Visual and interactive presentation of the Floyd algorithm ("Moving through the mapping" into a cycle).
- Adaptation of the Floyd algorithm for a digital signature attack.

Starting point

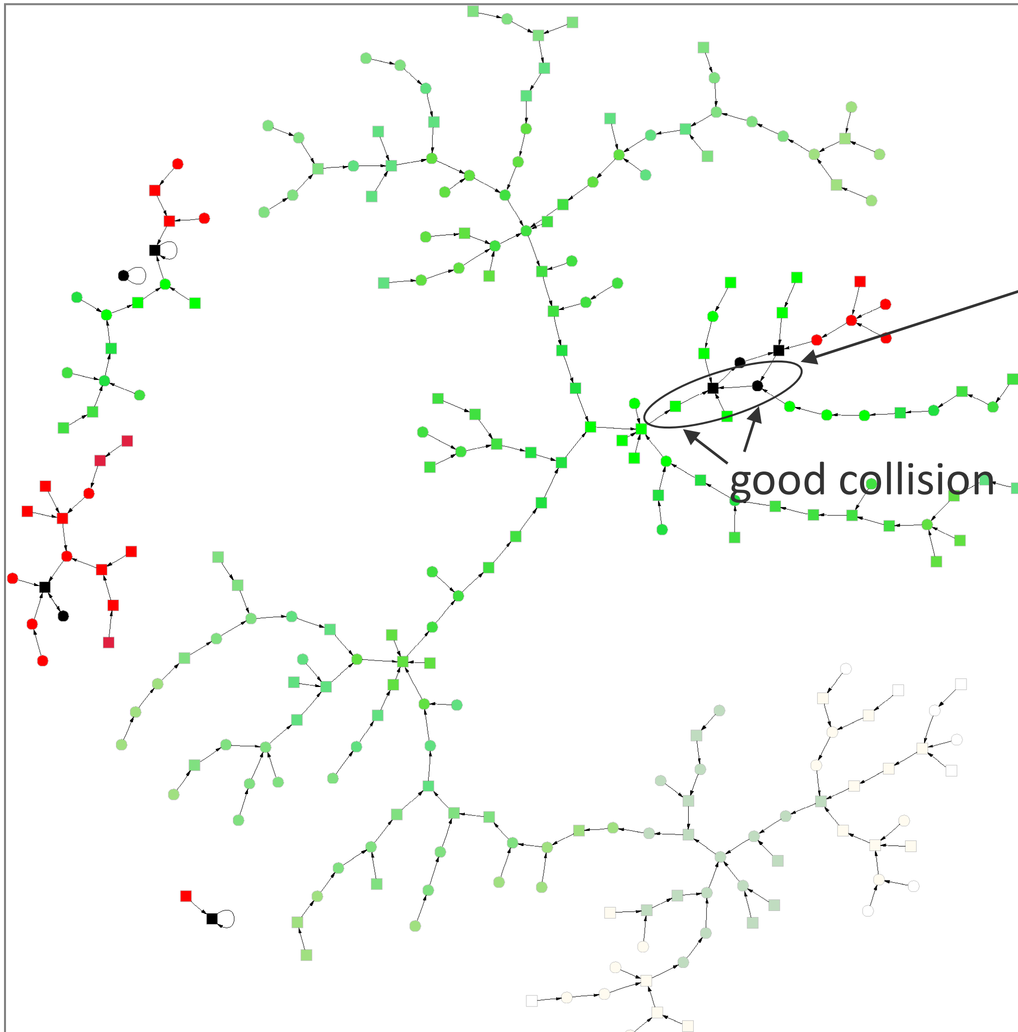
Good collision

Bad collision

*The Floyd algorithm is implemented in CrypTool, but the visualization of the algorithm has not yet been implemented.

Examples (7)

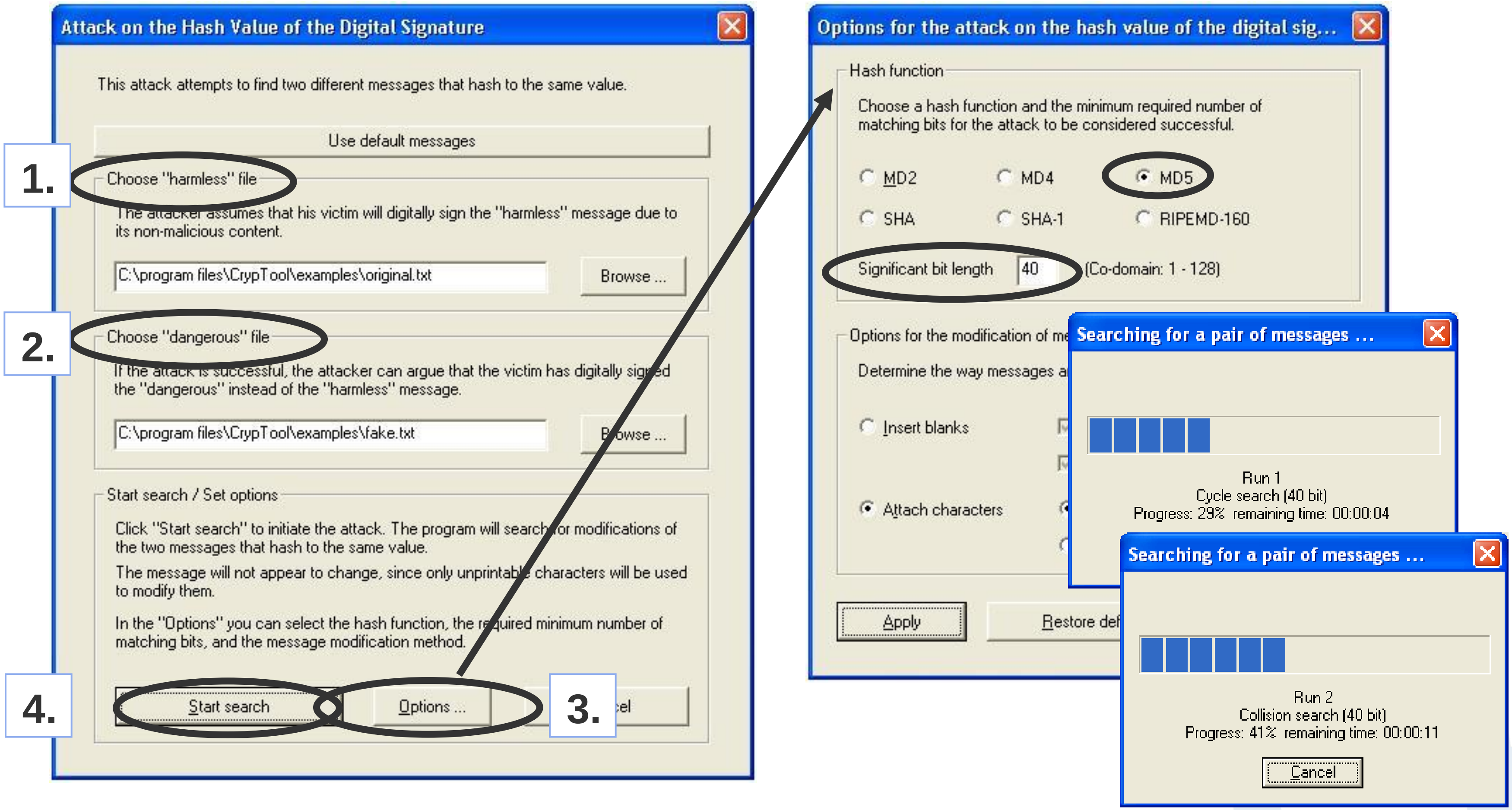
Attack on digital signature



An example of a **"good" mapping** (nearly all nodes are green).
In this graph almost all nodes belong to a big tree, which leads into the cycle with an even hash value and where the entry point predecessor within the cycle is odd. That means that the attacker finds a useful collision for nearly all starting points.

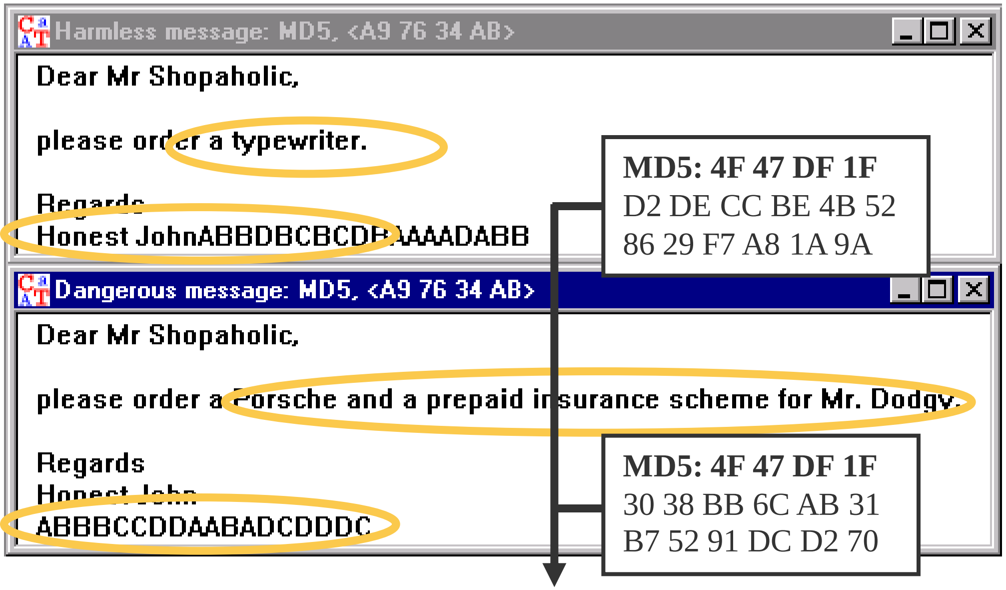
Examples (7)

Attack on digital signature: attack



Menu: “Analysis” \ “Hash” \ “Attack on the Hash Value of the Digital Signature”

Attack on digital signature: results



The first 32 bits of the hash values are identical.

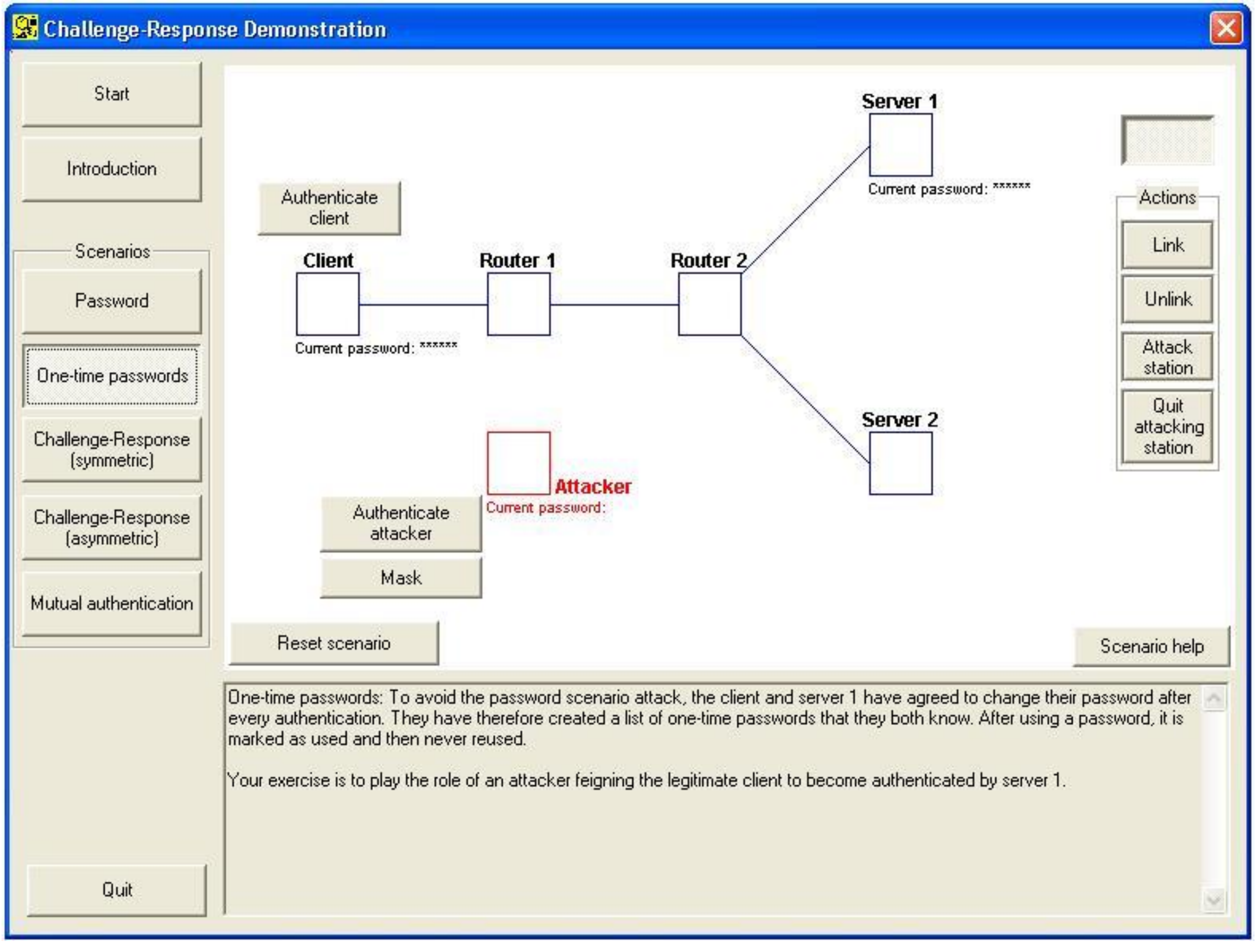
Experimental results

- A 72-bit *partial collision* (i.e., the first 72 hash value bits are identical) was found in a couple of days using a single PC.
- Today, signatures with hash values of 128 bits or less are vulnerable to a massive parallel search!
- It is therefore recommended to use hash values with a length of at least 160 bits.

In addition to the interactive tool, CrypTool also includes a command-line feature to execute and log the results for entire sets of parameter configurations.

Examples (8)

Authentication in a client-server environment

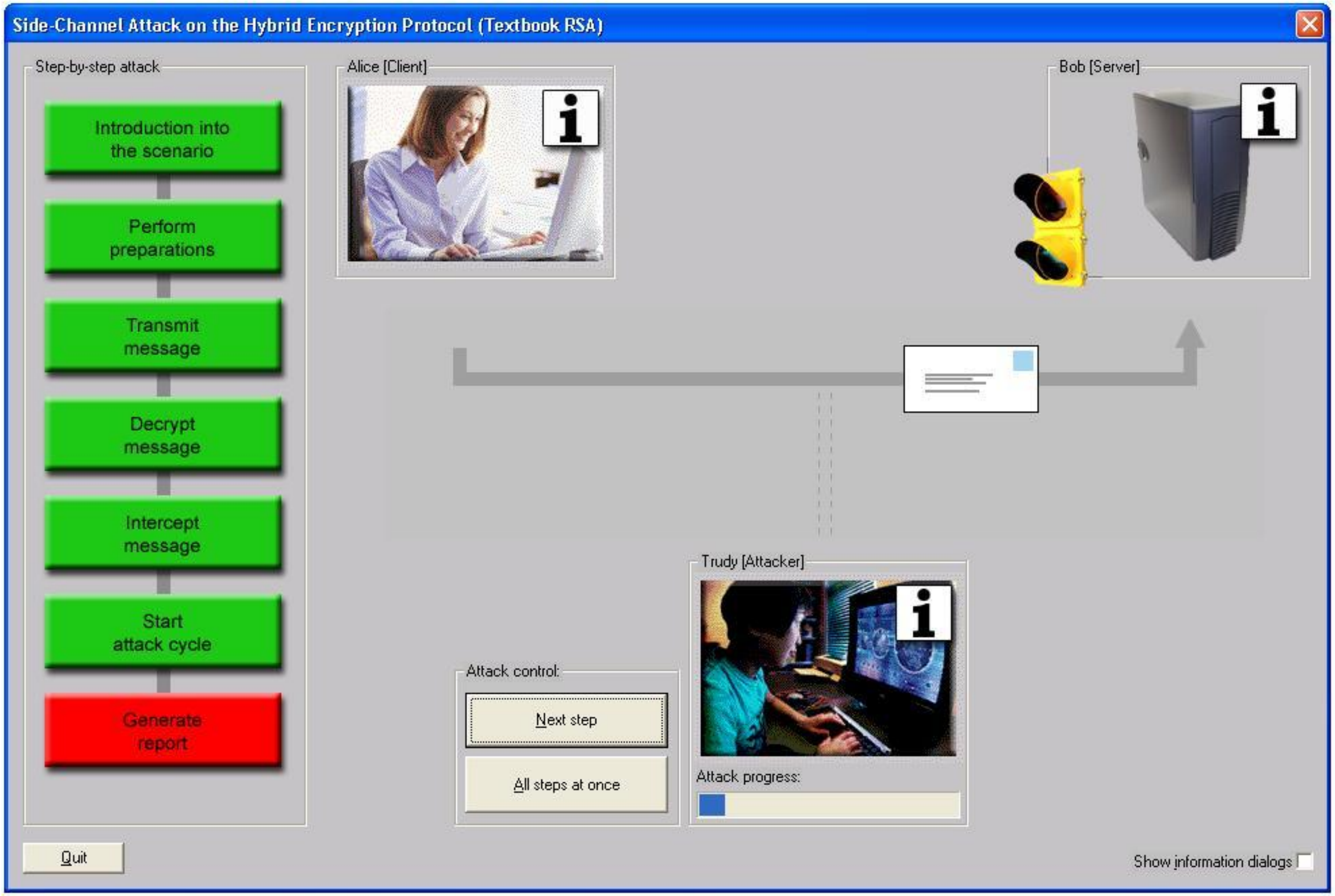


Menu: “Indiv. Procedures” \ “Protocols” \ “Network Authentication”

- Interactive demo for different authentication methods.
- Specifies vulnerabilities that an attacker could take advantage of.
- Allows the user to play the role of an attacker.
- **Learning outcome:** Only mutual authentication is secure.

Examples (9)

Demonstration of a side-channel attack (on a hybrid encryption protocol)



Menu: “Analysis” \ “Asymmetric Encryption” \ “Side-Channel Attack on Textbook RSA”

Examples (9)

Side channel attack concept

Ulrich Kuehn: *"Side-channel attacks on textbook RSA and ElGamal encryption"*, 2003

Prerequisites [CCA (Chosen-ciphertext attack) against deciphering oracle]

- RSA encryption: $C = M^e \pmod N$ and decryption: $M = C^d \bmod N$.
- 128-bit session keys (in M) are encoded according to textbook RSA (null padding).
- The server knows the secret key d and
  - uses after decryption only the least significant 128 bits without validating the null-padded bits, meaning that the server does not recognize if there is something there other than zero.
  - An error message is prompted if the encryption attempt results in an "incorrect" session key (decrypted text cannot be interpreted by the server). In all other cases there will be no message.

Idea for attack: Approximation of Z in 129 bits from the equation $N = M * Z$ per $M = \lfloor N/Z \rfloor$

M = | 000.................................000 | Session Key |

Null-Padding

M

$C = M^e \pmod N$

All bit positions for Z are successively calculated: for each step the attacker gets one additional bit. He or she then modifies C to C' (see below). If a bit overflow occurs while calculating M' on the server (recipient), the server sends an error message. Based on this information, the attacker can determine a single bit of Z.

If and only if the most significant bit of M is equal to 1, then M' is not equal to M mod 2^{128}.

M' = | Session Key | 0..............000 | Session Key |

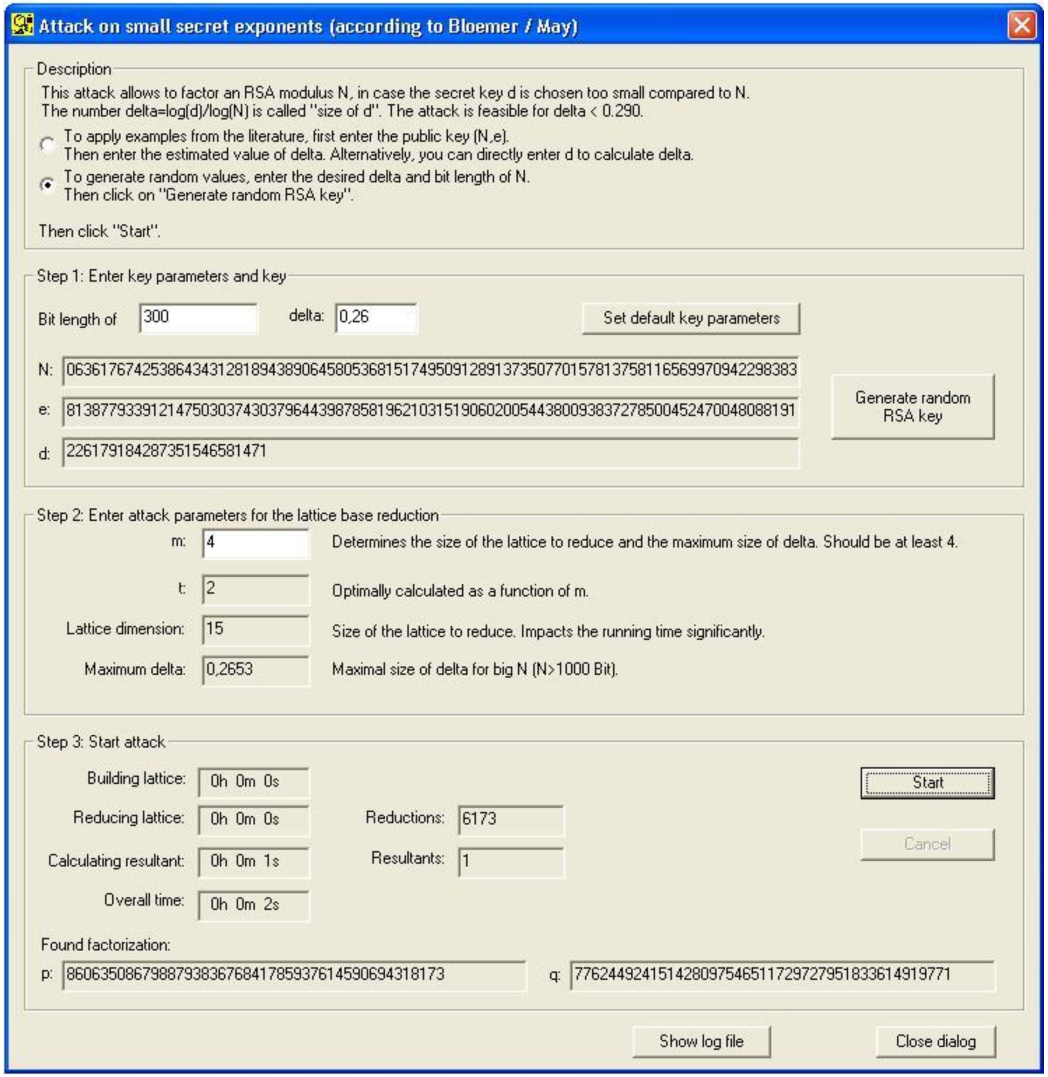
$M \cdot Z \cdot 2^{128}$

M

$C' = M'^e = M^e \cdot (1+Z \cdot 2^{128})^e \pmod N$

Examples (10)

Mathematics: Attacks on RSA using lattice reduction



- Demonstrates that the parameters of RSA should be chosen to withstand the lattice reduction attacks described in current literature.
- **3 variants** which are *not* resistant:
  1. The secret exponent d is too small in comparison to N.
  2. One of the factors of N is partially known.
  3. A part of the plaintext is known.
- These assumptions are realistic.

Menu: "Analysis" \
"Asymmetric Encryption" \
"Lattice Based Attacks on RSA" \ …

Examples (11)

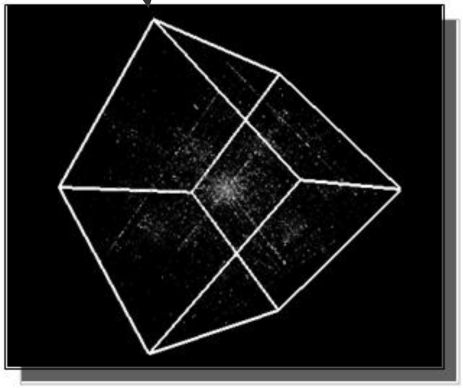
Random analysis with 3-D visualization

You can turn the cube with the mouse to the perspective you wish.

3-D visualization for random analysis

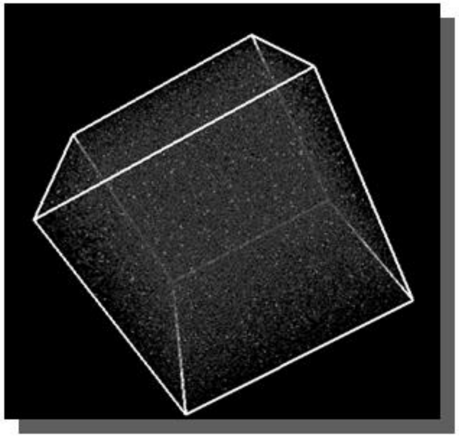
Example 1

- Open an arbitrary file (e.g. report in Word or PowerPoint presentation)
- It is recommended to select a file with at least 100 kB
- 3-D analysis
- Result: **structures are easily recognizable**



Example 2

- Generation of random numbers via menu:
  "Indiv. Procedures" \ "Tools" \ "Generate Random Numbers"
- It is recommended to generate at least 100,000 random bytes
- 3-D analysis
- Result: **uniform distribution (no structures are recognizable)**



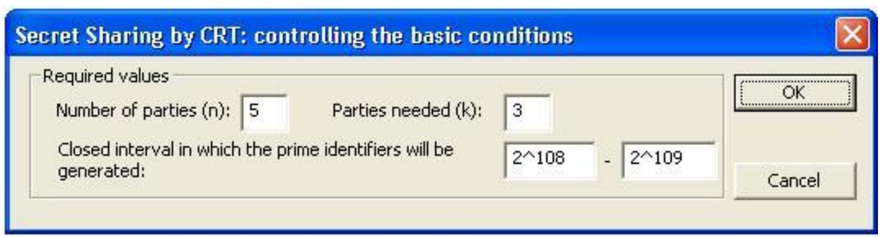
Menu: "Analysis" \ "Analyze Randomness" \ "3-D Visualization"

Examples (12)

Secret sharing with CRT – implementation of the Chinese remainder theorem (CRT)

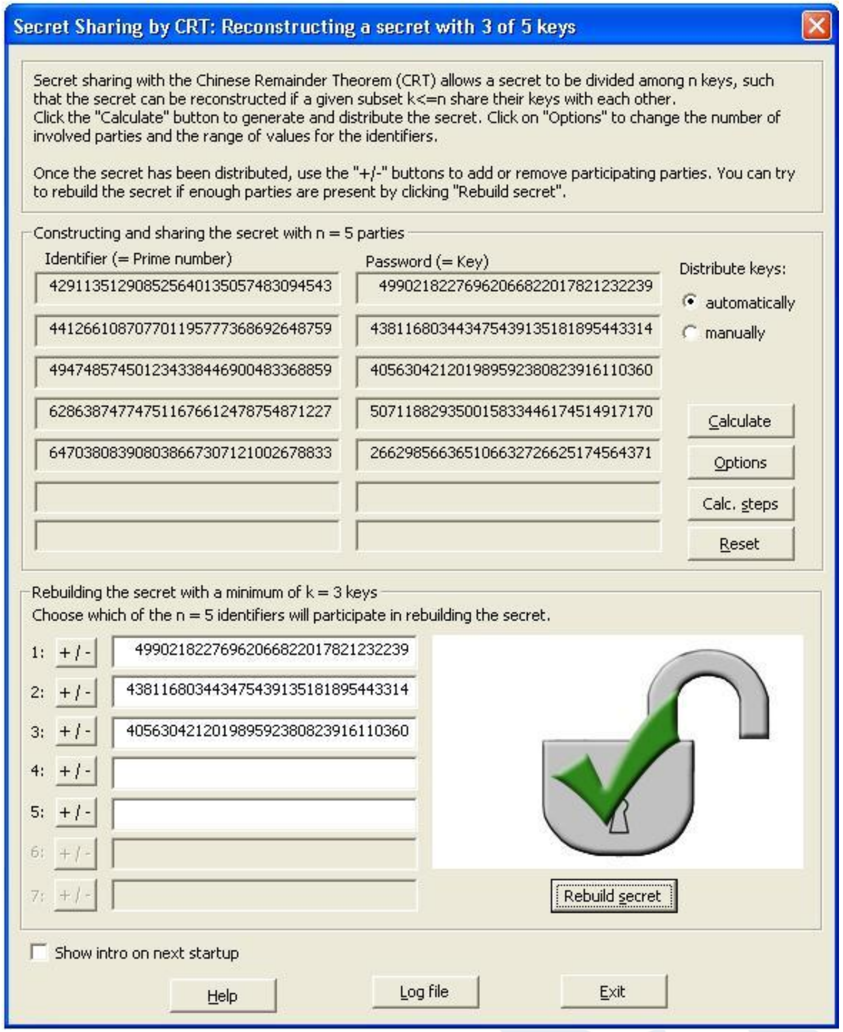
Secret sharing example (1)

- **Problem**
  - 5 people each receive a single key
  - To gain access, at least 3 of the 5 people must be present
- **“Options”** allows the user to configure additional settings.



- **“Calc. steps”** shows all of the steps in key generation.

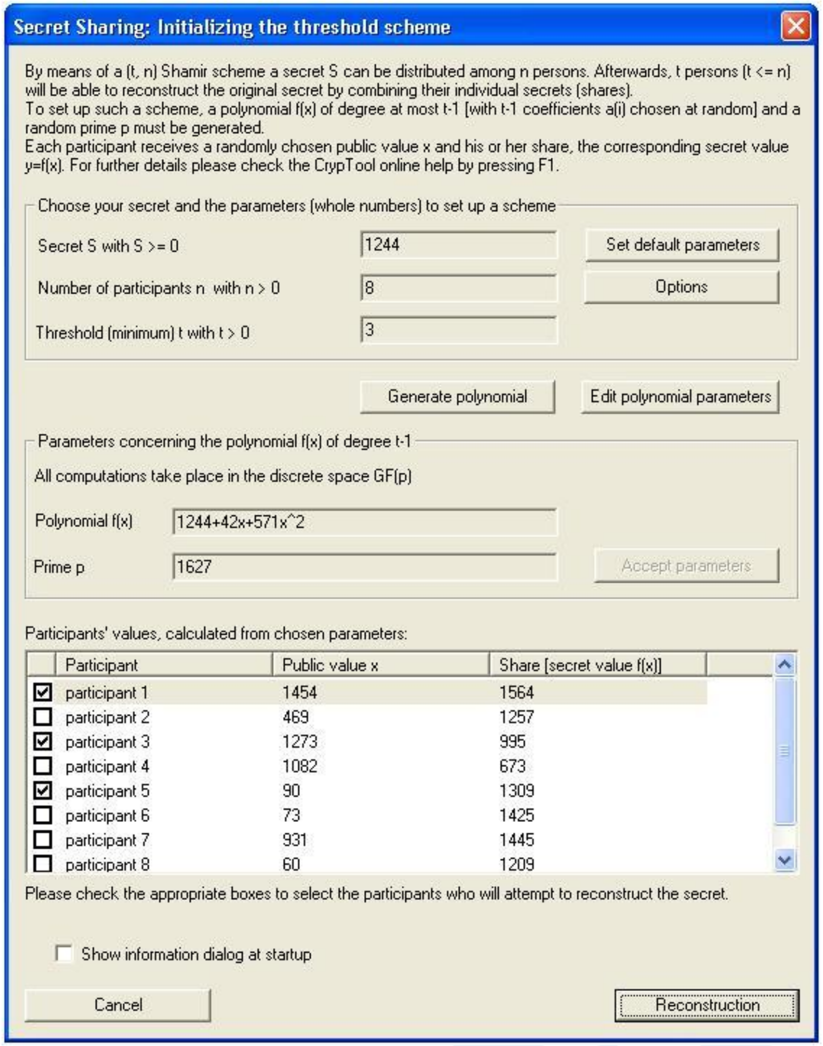
Menu: “Indiv. Procedures” \
“Chinese Remainder Theorem Applications” \
“Secret Sharing by CRT”



Secret sharing example (2)

- **Problem**
  - A secret value is to be divided among n people.
  - t out of n people are required to restore the secret value K.
  - (t, n) threshold scheme
- **Perform it in the dialog:**
  1. Enter the secret K, number of persons n and threshold t
  2. Generate polynomial
  3. Select parameters
  4. Click **“Reconstruction”** to restore the secret.

Menu: “Indiv. Procedures” \
“Secret Sharing Demonstration (Shamir)”

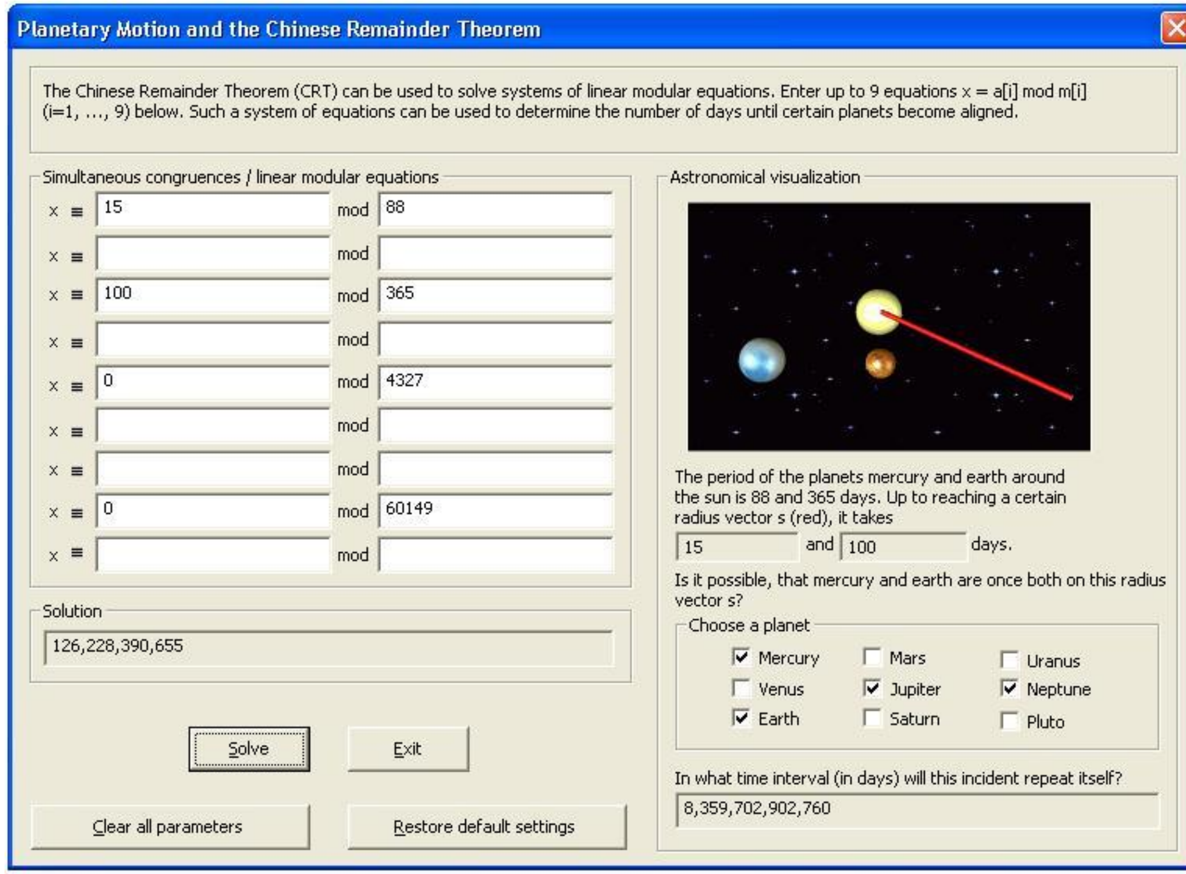


Examples (13)

Implementation of CRT to solve linear modular equation systems

Astronomical scenario

- How long would it take for a given number of planets (with different rotation times) to become aligned?
- The result is a linear modular equation system that can be solved with the Chinese remainder theorem (CRT).
- In this demo you can enter up to 9 equations and compute a solution using the CRT.



Menu: "Indiv. Procedures" \ "Chinese Remainder Theorem Applications" \ "Astronomy and Planetary Motion"

Examples (14)

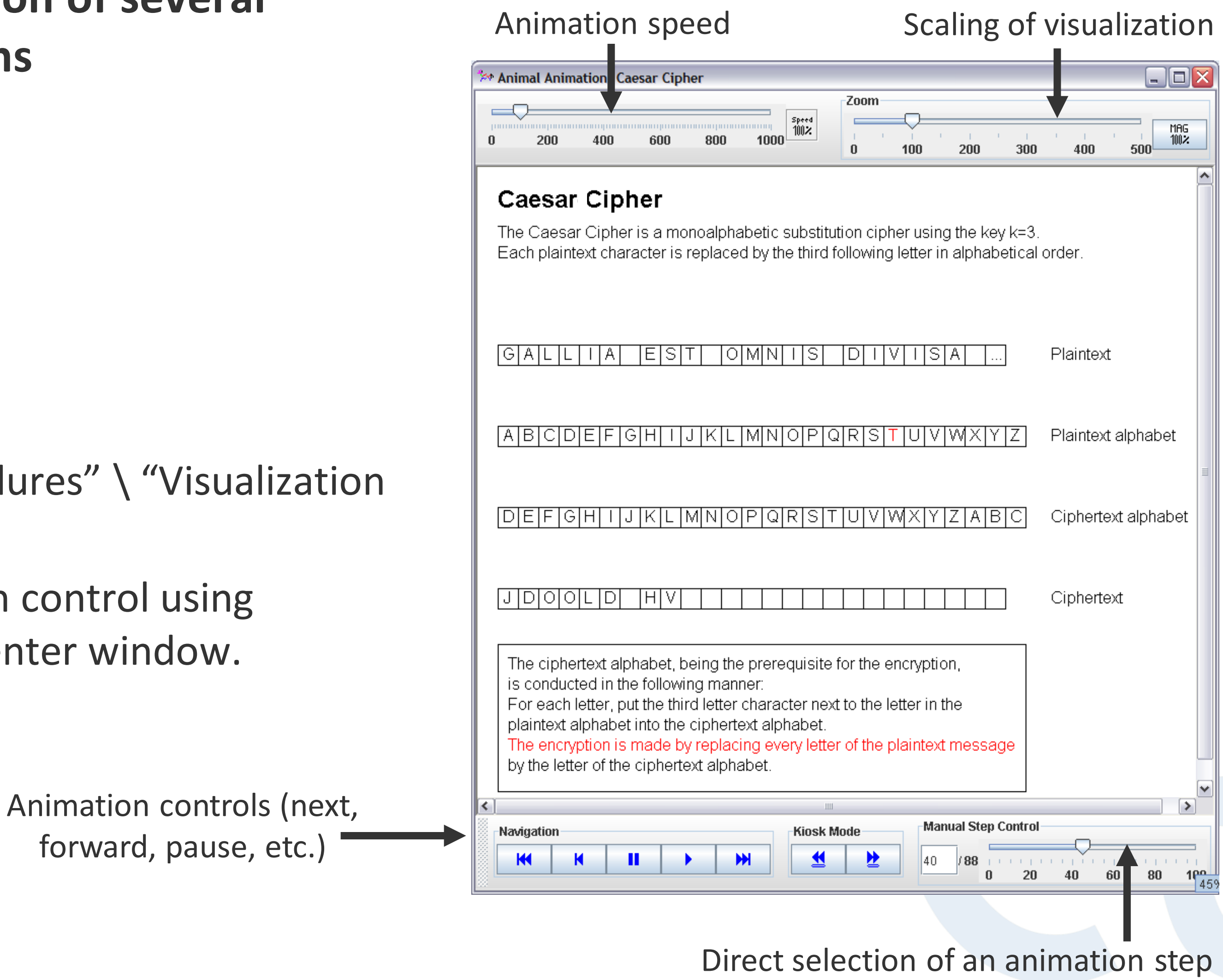
Visualization of symmetric encryption methods using ANIMAL (1)

Animated visualization of several symmetric algorithms

- Caesar
- Vigenère
- Nihilist
- DES

CrypTool

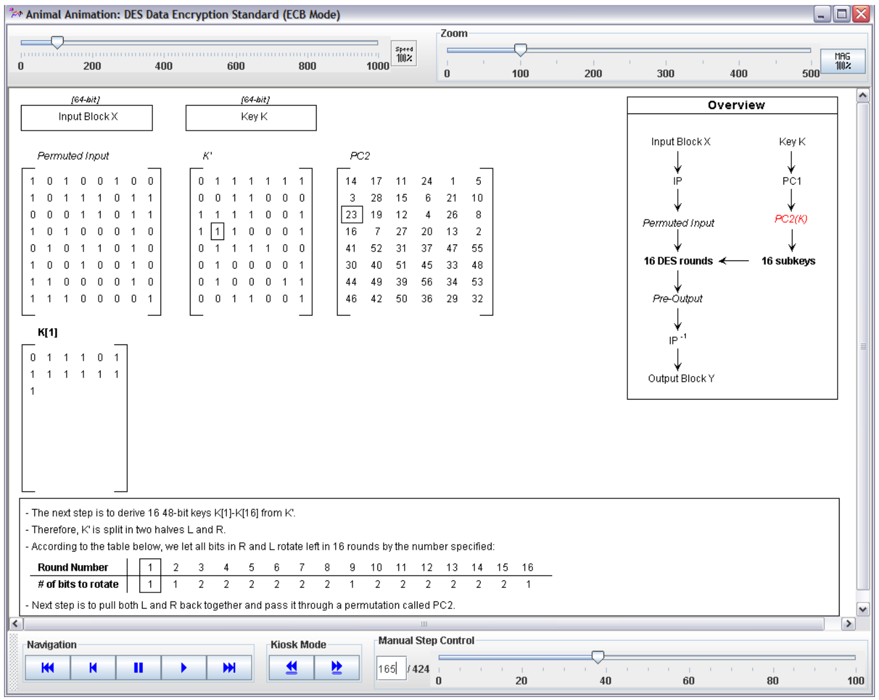
- Menu: “Indiv. Procedures” \ “Visualization of Algorithms” \ …
- Interactive animation control using integrated control center window.



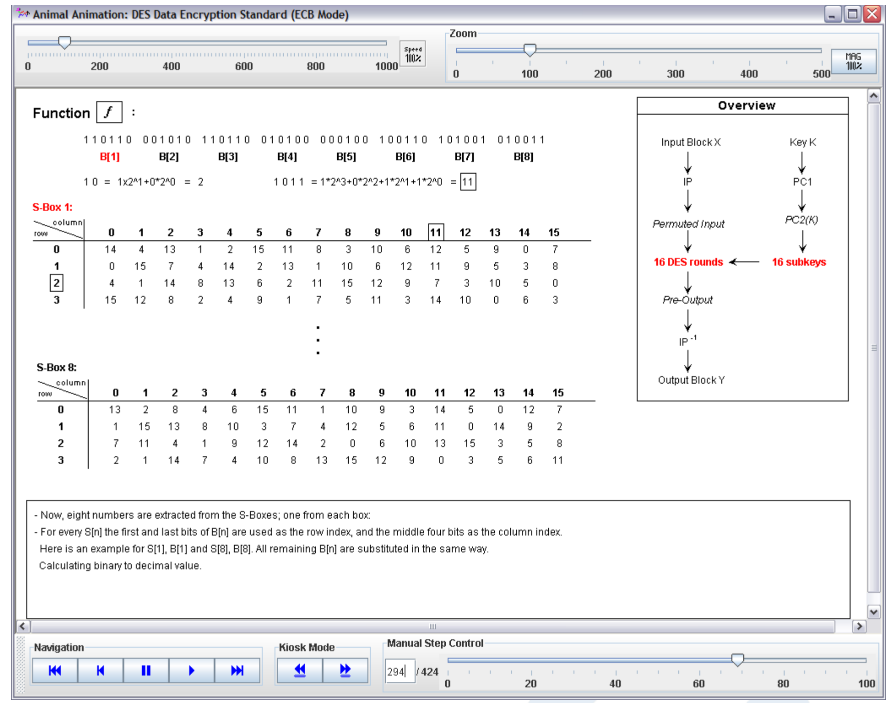
Examples (14)

Visualization of symmetric encryption methods using ANIMAL (2)

Visualization of DES encryption



After the permutation of the input block with the initialization vector (IV), the key K is permuted with PC1 and PC2.



The core function f of DES, which links the right half of the block R_{i-1} with the partial key K_i.

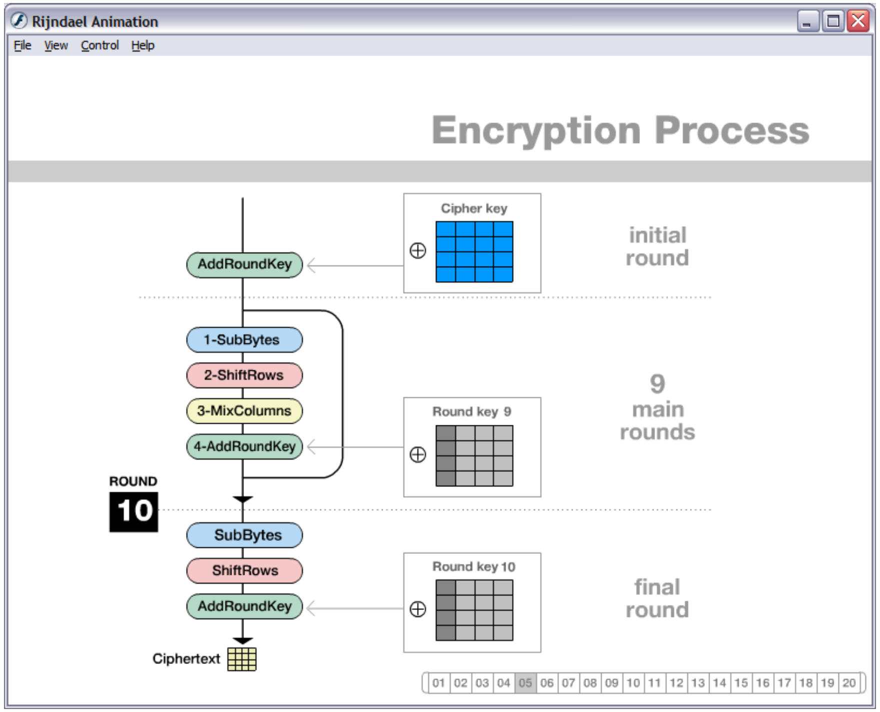
Visualizations of AES (Rijndael cipher) – in Flash

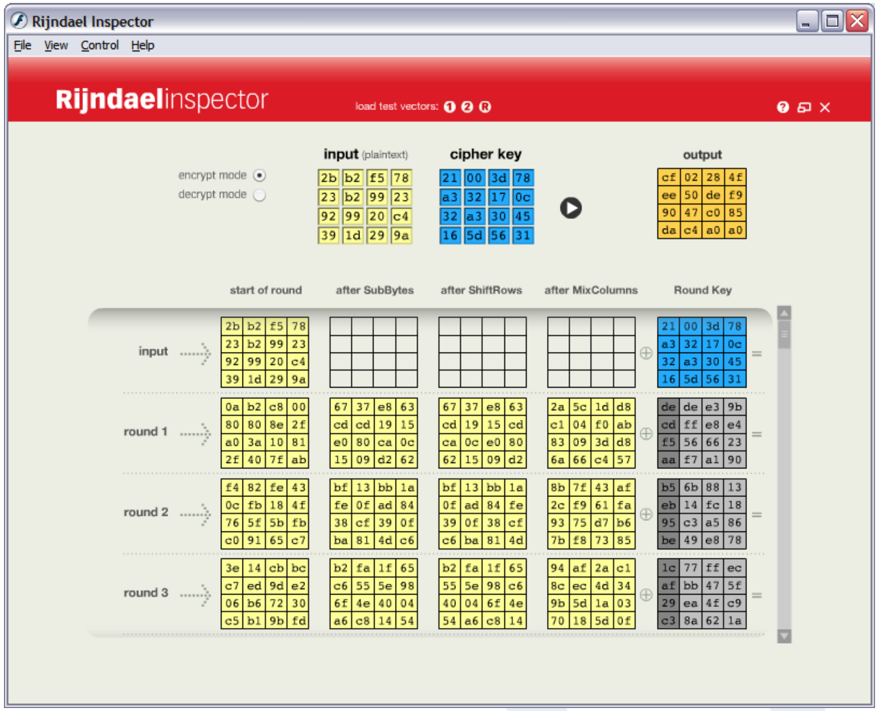
Rijndael Animation (the Rijndael cipher was the winner of the AES selection competition)

- Shows the encryption processes of each round (using fixed initial data)

Rijndael Inspector

- Test with your own data (shows the contents of the matrix after each round)

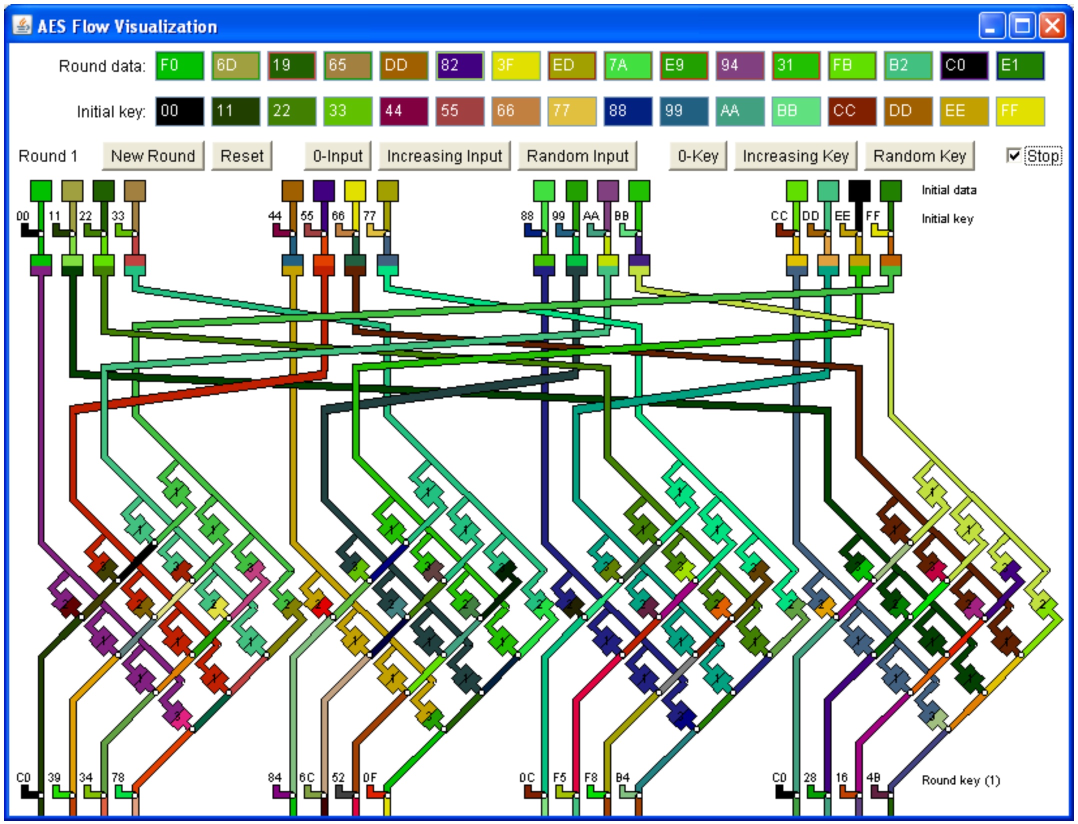




Menu: “Indiv. Procedures” \ “Visualization of Algorithms” \ “AES” \ “Rijndael Animation” or “Rijndael Inspector”

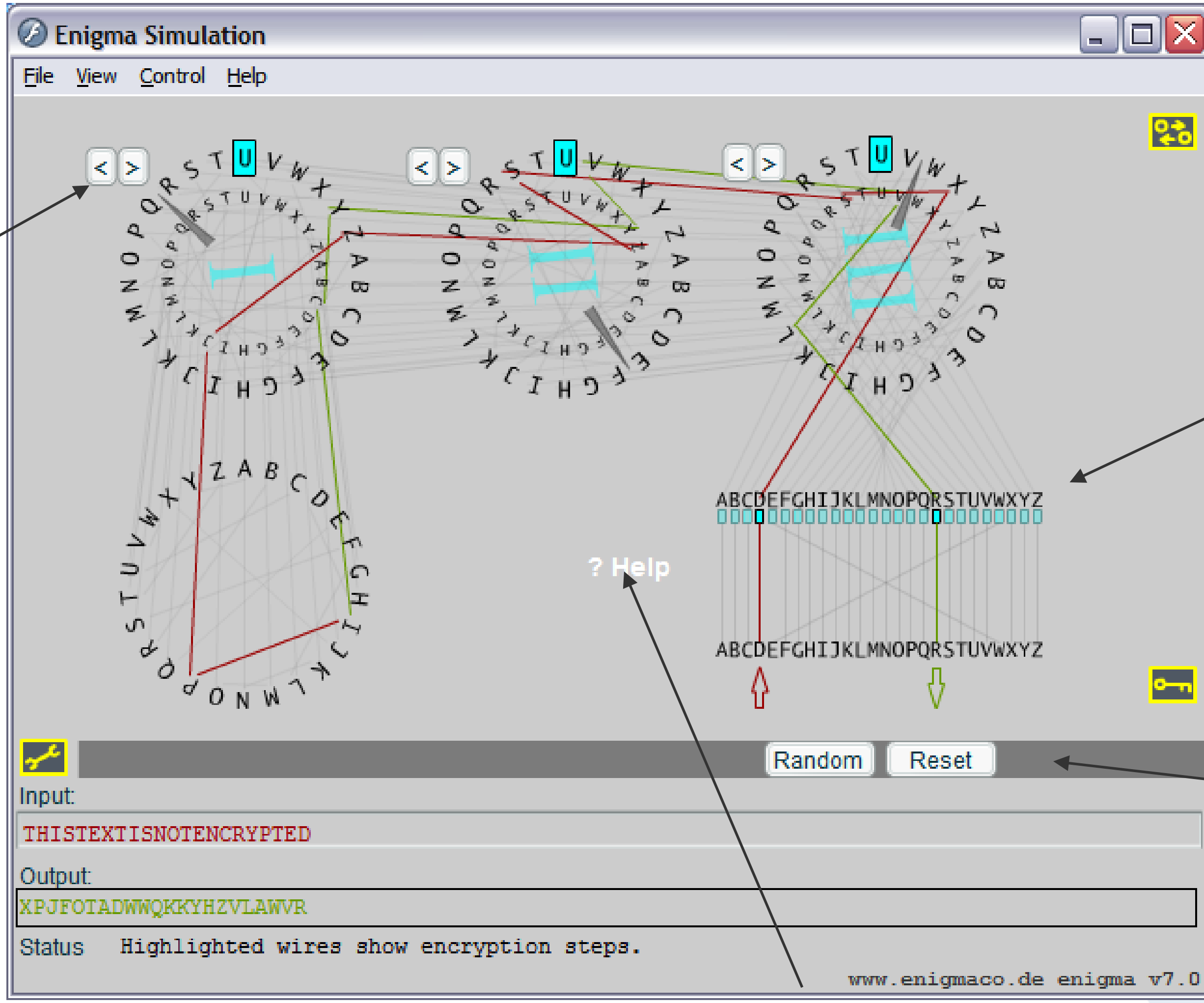
Rijndael flow visualization

- Visualization of data changes per round using color gradient



Menu: “Indiv. Procedures” \ “Visualization of Algorithms” \ “AES” \ “Rijndael Flow Visualization…”

Visualization of the Enigma encryption



Select rotors

Change rotor setting

Change plugs

Show settings

Input of plaintext

Output of encrypted text

Reset Enigma to initial state or random state

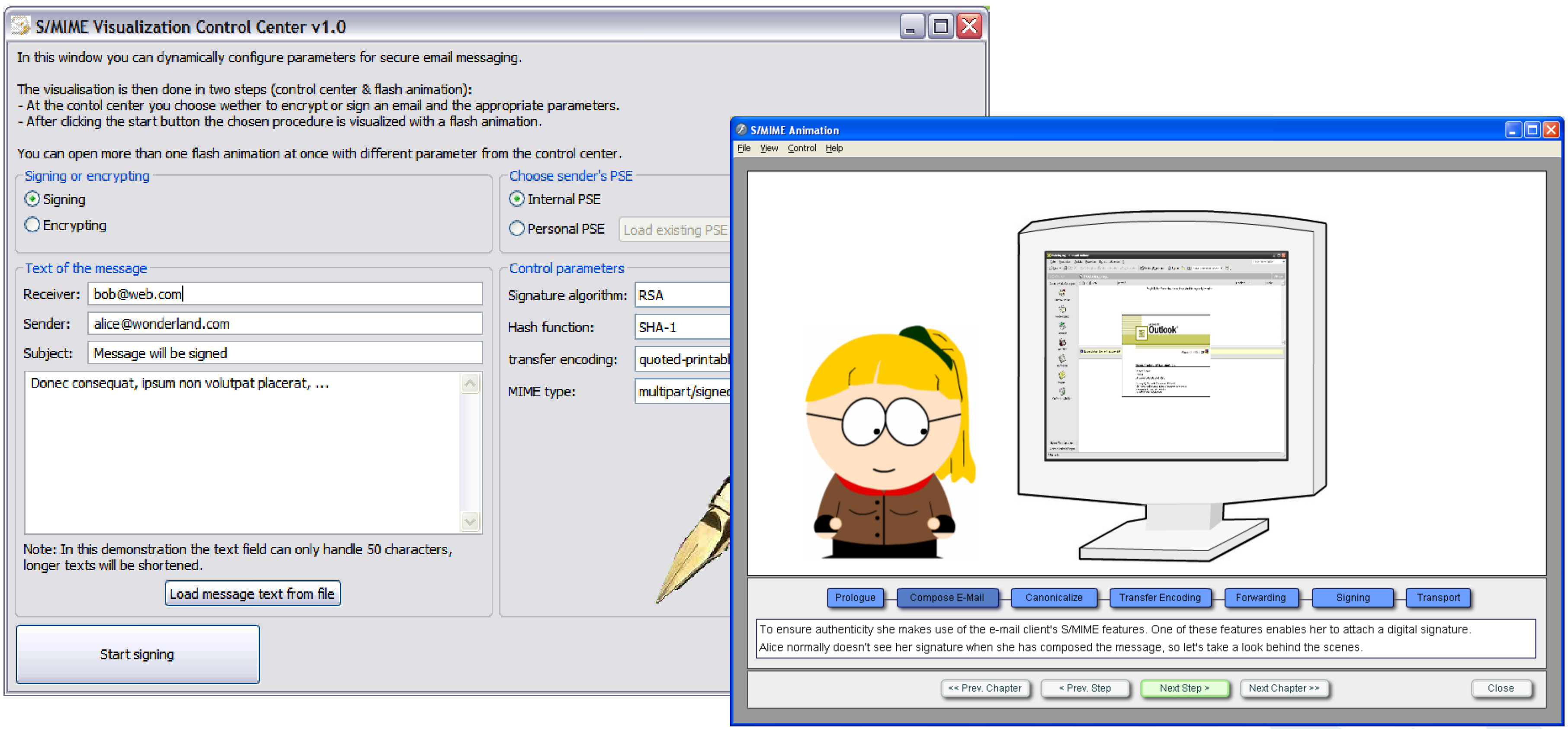
Additional HTML online help

Examples (17)

Visualization of secure Email using S/MIME

S/MIME visualization

- Control Center: Sign/Encrypt messages with different parameters
- Animation: From the sender’s creation of the message until it is read by the receiver



Menu: “Indiv. Procedures” \ “Protocols” \ “Secure E-Mail with S/MIME…”

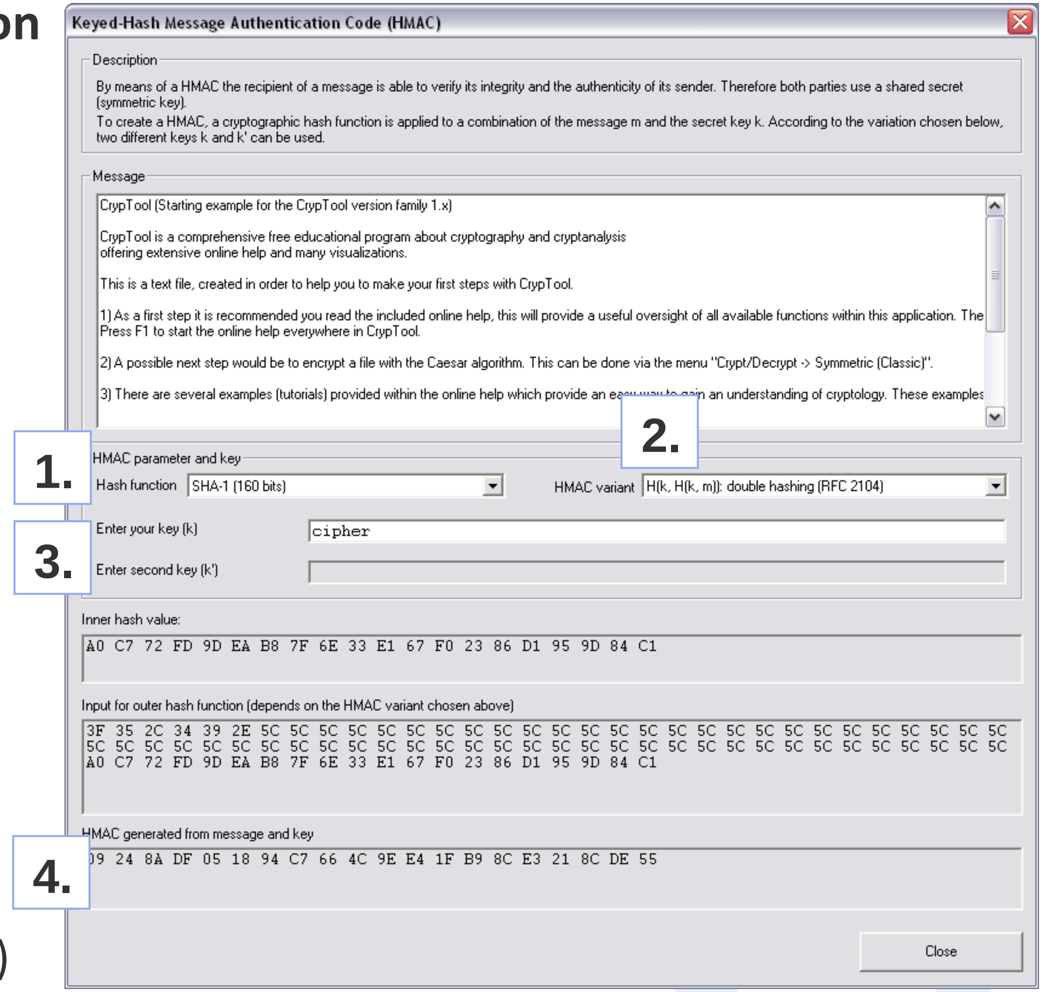
Generation of a keyed-hash message authentication code (HMAC)

Keyed-Hash Message Authentication Code (HMAC)

- Ensures
  - Integrity of a message
  - Authentication of the message
- Basis: a common key for sender and recipient
- Alternative: Digital signature

Generation of a MAC in CrypTool

1. Choose a hash function
2. Select HMAC variant
3. Enter a key (or keys, depending on the HMAC variant)
4. Generation of the HMAC (automatic)



Menu: “Indiv. Procedures” \ “Hash” \ “Generation of HMACs”

Examples (19)

Hash demonstration

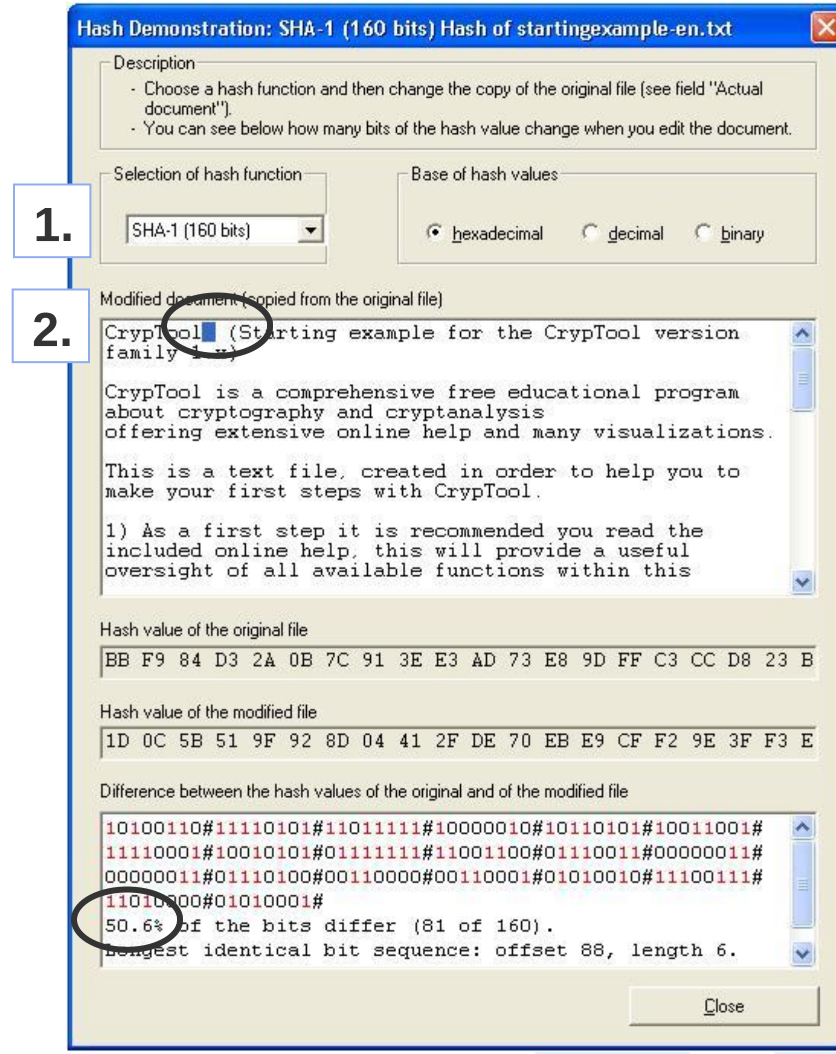
Sensitivity of hash functions to plaintext modifications

1. Select a hash function
2. Modify characters in plaintext

Example:
By adding a space after the word “CrypTool” in the example text, 50.6 % of the bits in the resulting hash value will change.

A good hash function should react highly sensitively to even the smallest change in the plaintext – “Avalanche effect” (small change, big impact).

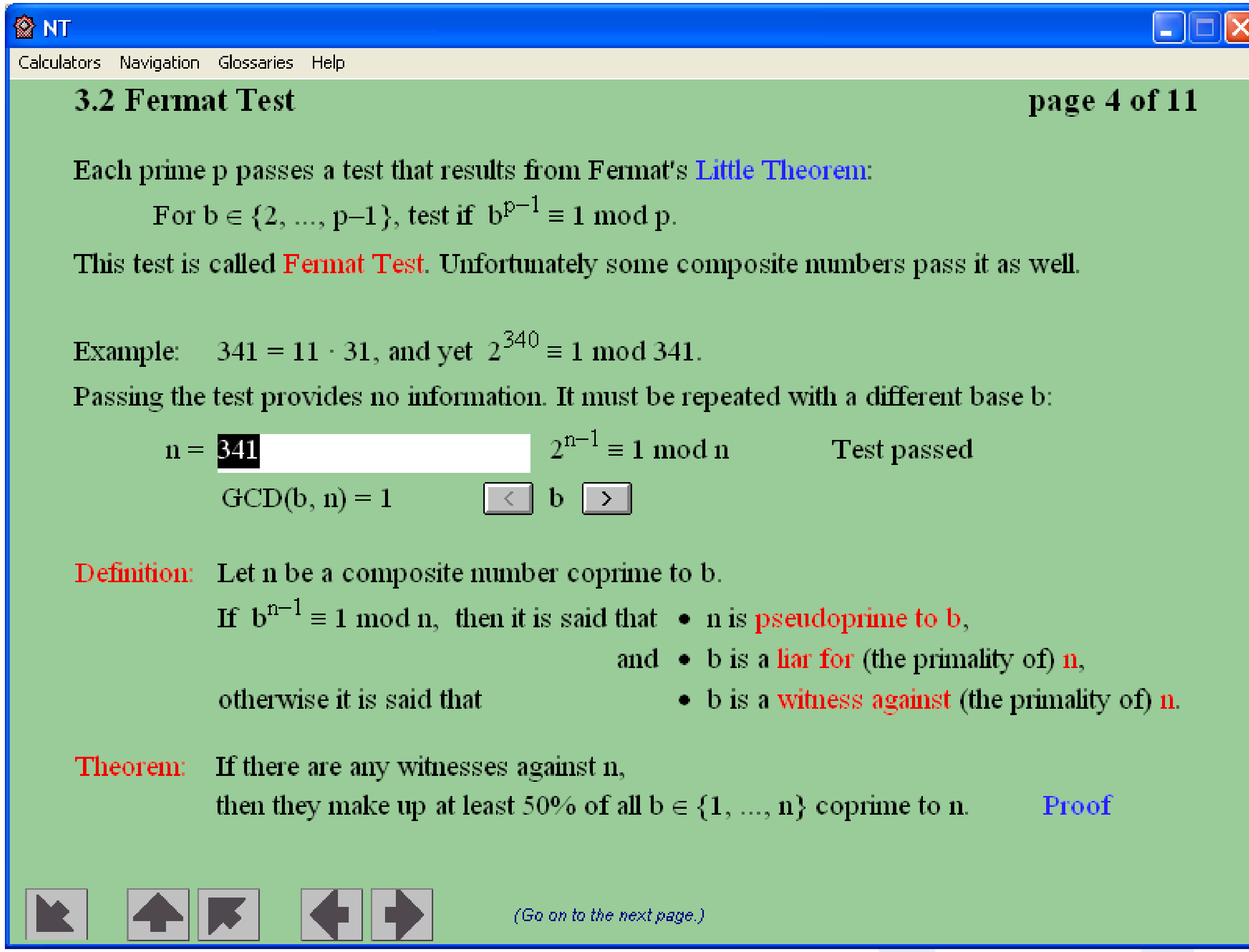
Menu: “Indiv. Procedures” \ “Hash” \ “Hash Demonstration”



Examples (20)

Educational tool for number theory

- **Number theory** supported by graphical elements and interactive tools
- **Topics**
  1. Integers
  2. Residue classes
  3. Prime generation
  4. Public-key cryptography
  5. Factorization
  6. Discrete logarithms



Menu: “Indiv. Procedures” \ “Number Theory – Interactive” \ “Learning tool for number theory”

Examples (21)

Point addition on elliptic curves

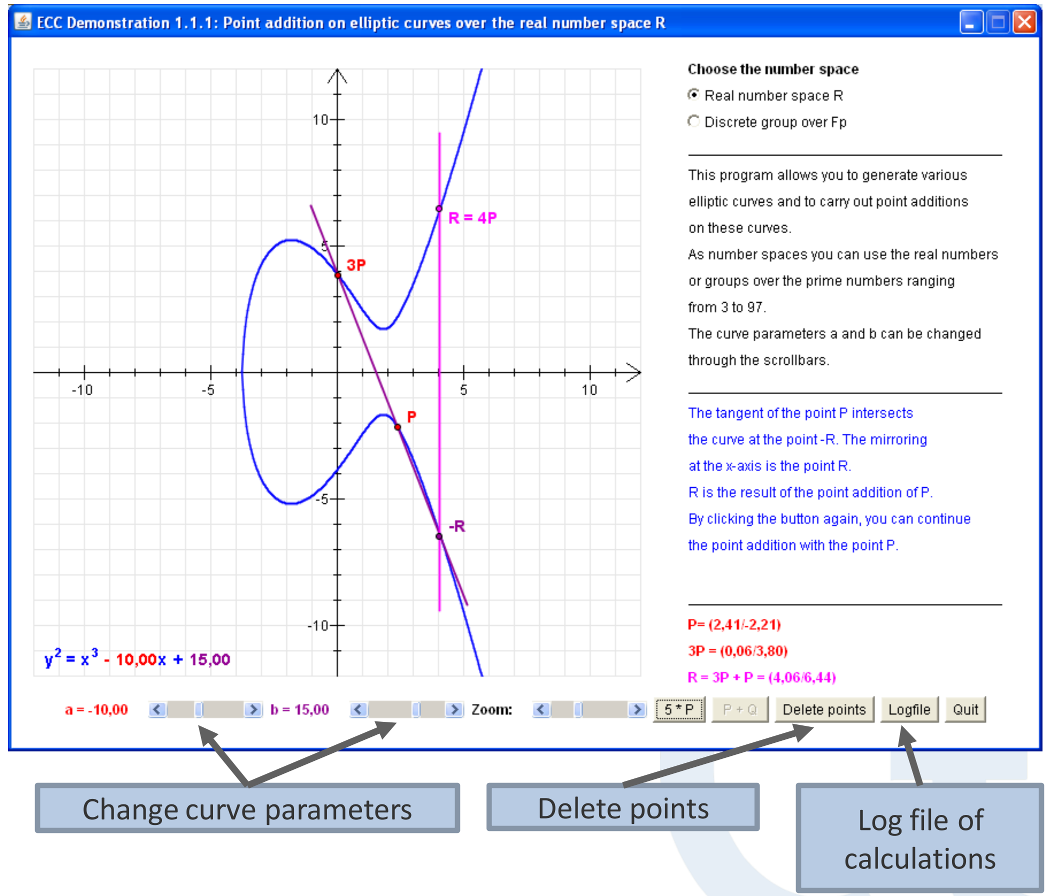
- Visualization of point addition on elliptic curves (both real and discrete)
- Foundation of elliptic curve cryptography (ECC)

Example 1: Add two different points

- Mark point P on the curve
- Mark point Q on the curve
- Pressing button “P+Q” creates point R:
  - The straight line through P and Q intersects the curve at point -R.
  - Mirroring -R over the X-axis produces the point R.

Example 2: Multiply a single point

- Mark point P on the curve
- Pressing button “2*P” creates point R:
  - The tangent of point P intersects the curve at point -R.
  - Mirroring -R over the X-axis produces the point R.



Menu: “Indiv. Procedures” \ “Number Theory – Interactive” \ “Point Addition on Elliptic Curves”

Examples (22)

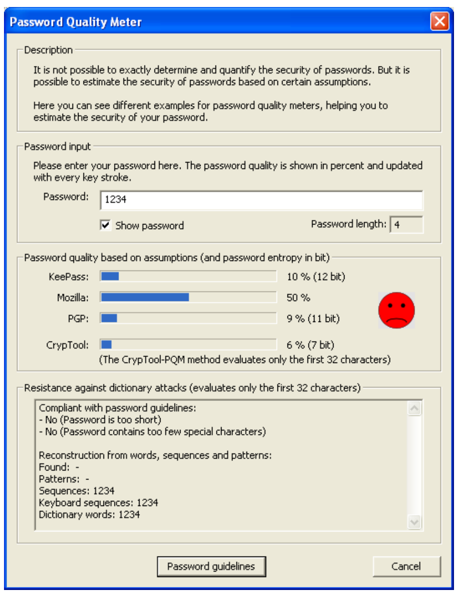
Password quality meter (PQM) (1)

Functions

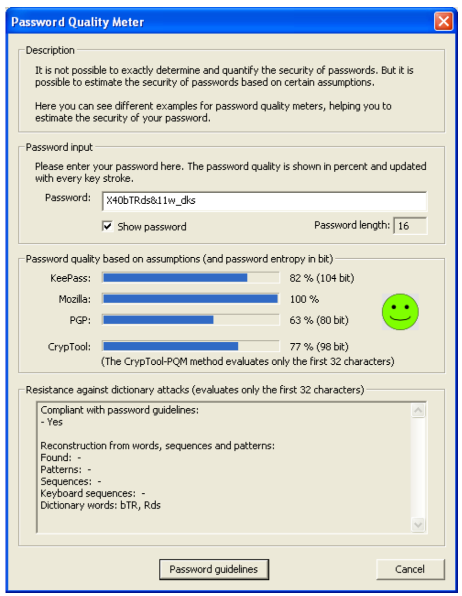
- Measure the quality of passwords
- Compare with PQMs in other applications: KeePass, Mozilla und PGP
- Experimental evaluation with the CrypTool algorithm

- Example: Input of a password in cleartext

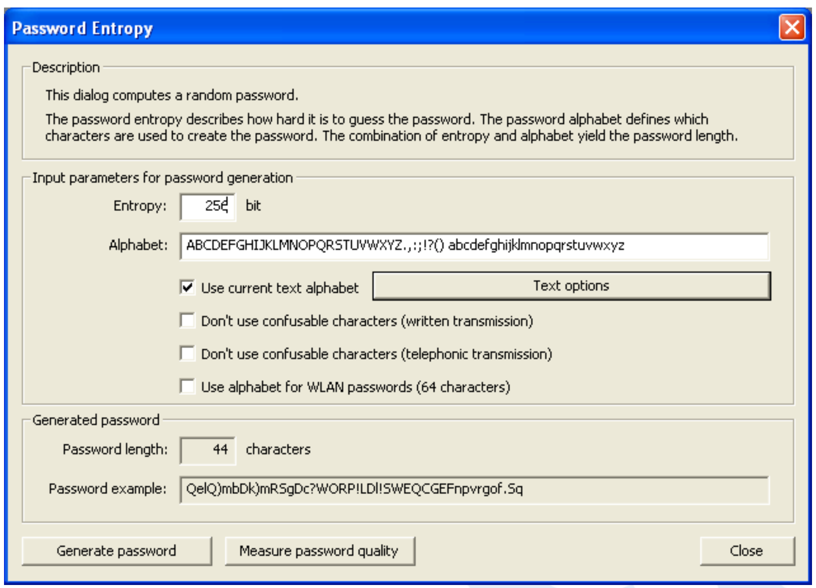
Password: ***1234***



Password: ***X40bTRds&11w_dks***



Menu: “Indiv. Procedures” \ “Tools” \ “Password Quality Meter”



Menu: “Indiv. Procedures” \ “Tools” \ “Password Entropy”

Examples (22)

Password quality meter (PQM) (2)

Insights from the Password Quality Meter

- Password quality depends primarily on the **length of the password**.
- A higher quality of the password can be achieved by using **different types of characters**: upper/lower case, numbers, and special characters **(password space)**
- **Password entropy** is an indicator of the randomness of the password characters within the password space (higher password entropy results in improved password quality)
- Passwords should **not exist in a dictionary** (remark: a dictionary check is not yet implemented in CrypTool).

Quality of a password from an attacker's perspective

- Attack on a password (if any number of attempts are possible):
  1. Classical **dictionary attack**
  2. Dictionary attack **with variants** (e.g., 4-digit number combinations: "Summer2007")
  3. **Brute-force attack** by testing all combinations (with additional parameters such as limitations on the types of character sets)

⇨ A good password should be chosen so that attacks 1 and 2 do not compromise the password. Regarding brute-force attacks, the most important factors are the length of the password (recommended at least 8 characters) and the character set that was used.

Examples (23)

Brute-force analysis (1)

Brute-force analysis

Optimized brute-force analysis with the assumption that the key is partially known.

Example – Analysis with DES (ECB)

Attempt to find the remainder of the key in order to decrypt an encrypted text.
(Assumption: the plaintext is a block of 8 ASCII characters.)

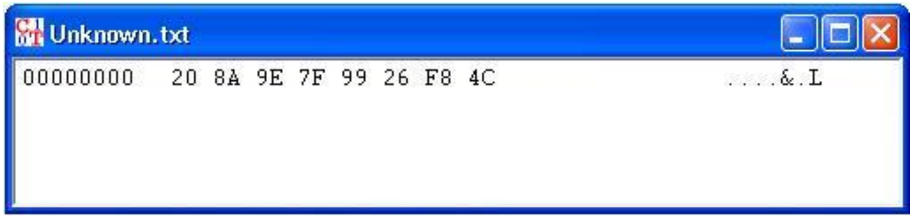
Key (Hex)	Encrypted text (Hex)
68ac78dd40bbefd*	66b9354452d29eb5
0123456789ab****	1f0dd05d8ed51583
98765432106*****	bcf9ebd1979ead6a
0000000000******	8cf42d40e004a1d4
000000000000****	0ed33fed7f46c585
abacadaba*******	d6d8641bc4fb2478
dddddddddd******	a2e66d852e175f5c

Examples (23)

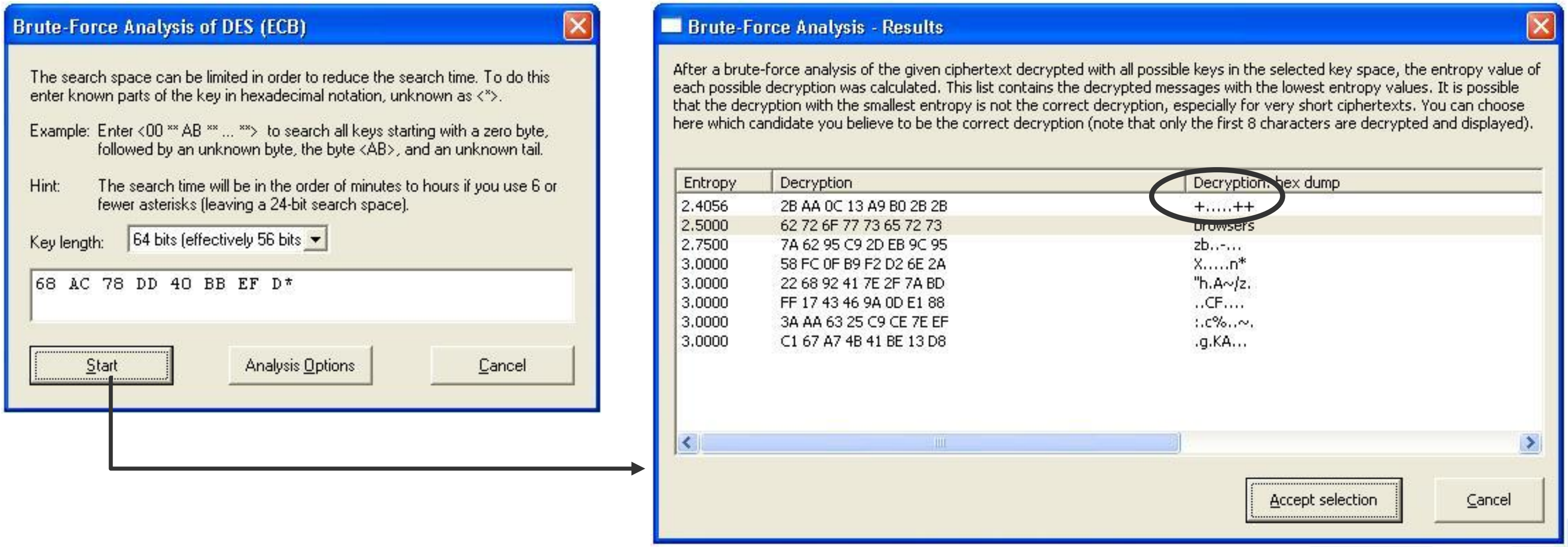
Brute-force analysis (2)

1. Input of encrypted text
2. Use brute-force analysis
3. Input partially known key
4. Start brute-force analysis
5. Analysis of the results: the correct decryption usually has relatively low entropy. However, because a very short plaintext has been used in this example, the correct result does not have the lowest entropy.

Select “View” \ “Show as HexDump”

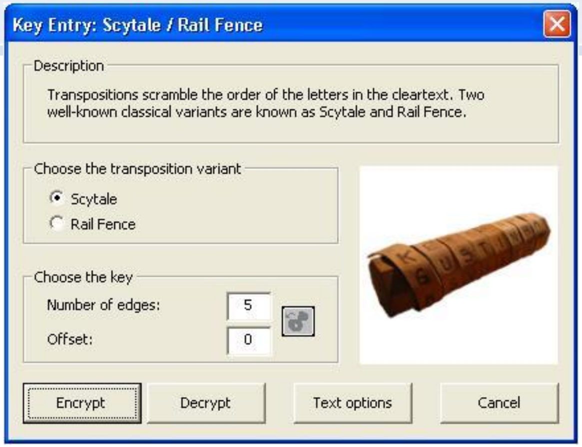


Menu: “Analysis” \ “Symmetric Encryption (modern)” \ “DES (ECB)”



Examples (24)

Scytale / Rail Fence



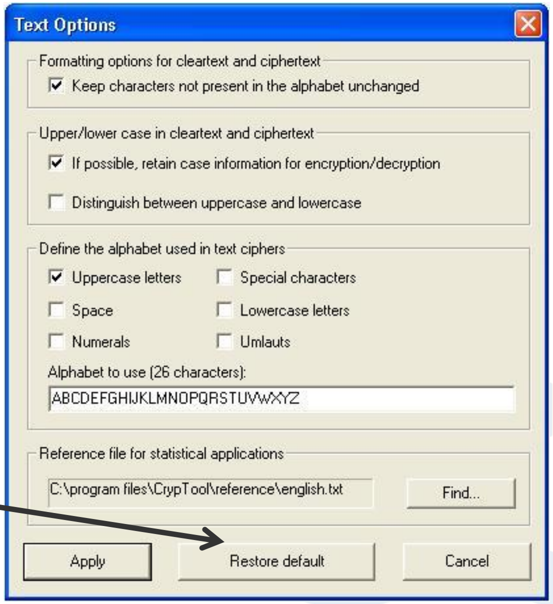
Scytale and Rail Fence

- Transpositions scramble the order of letters in the cleartext
- **Transposition variant**
  - Number of edges (Scytale)
  - Number of rows (Rail Fence)
  - Offset

Menu: “Crypt/Decrypt” \ “Symmetric (classic)” \ “Scytale / Rail Fence…”

Text options

- General text options (Menu: “Options” \ “Text Options…”)
- Formatting options for cleartext and ciphertext
- Processing of upper/lower case
- Alphabet for text processing (i.e., what set of characters should be encrypted/decrypted)
- Return to the default settings by clicking the “Restore Standard” button
- Creates the statistical reference patterns dynamically



Examples (25)

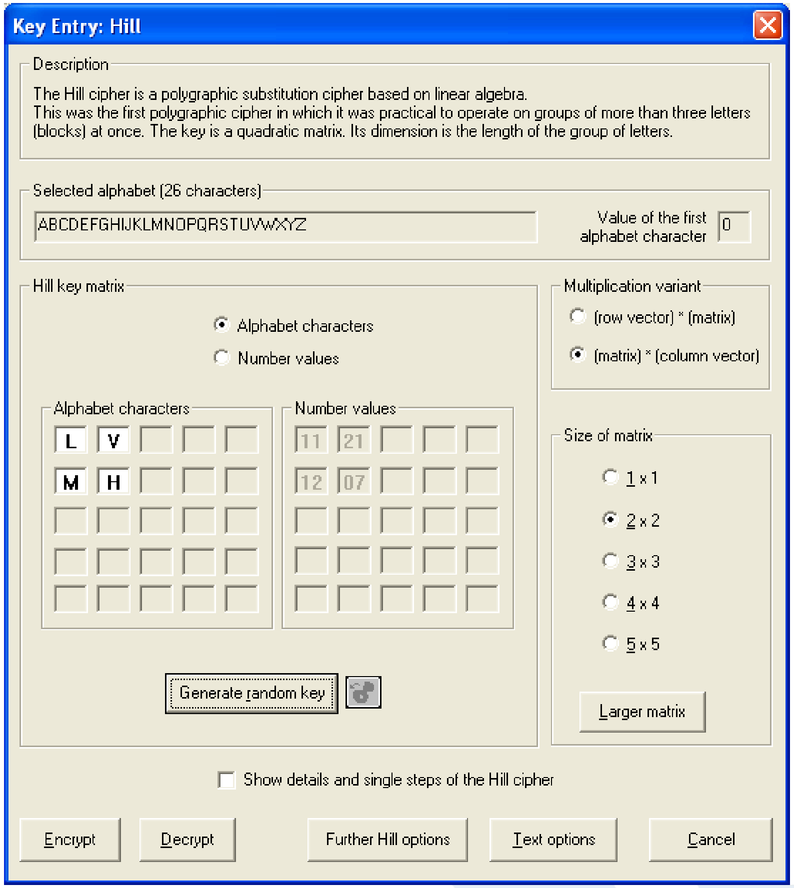
Hill encryption / Hill analysis (1)

Hill encryption

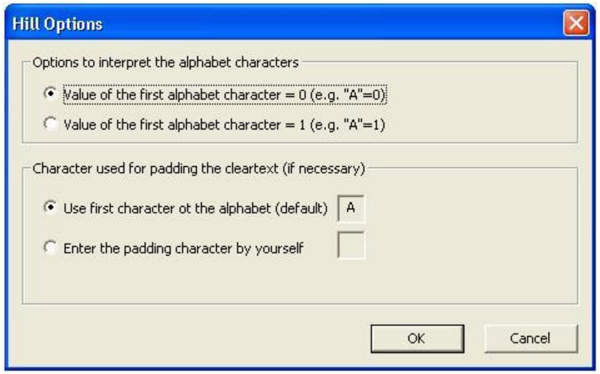
- Polygraphic substitution cipher
- Based on linear algebra

Key

- Alphabet characters (see text options) or number values
- Enter or generate random key
- Select multiplication variant
- Size of matrix
- Hill options



Menu:
“Crypt/Decrypt” \
“Symmetric (classic)” \
“Hill …”



Examples (25)

Hill encryption / Hill analysis (2)

Hill encryption

- Sample text with key `LVMH`

Hill analysis (with known plaintext)

1. **Long plaintext/ciphertext**

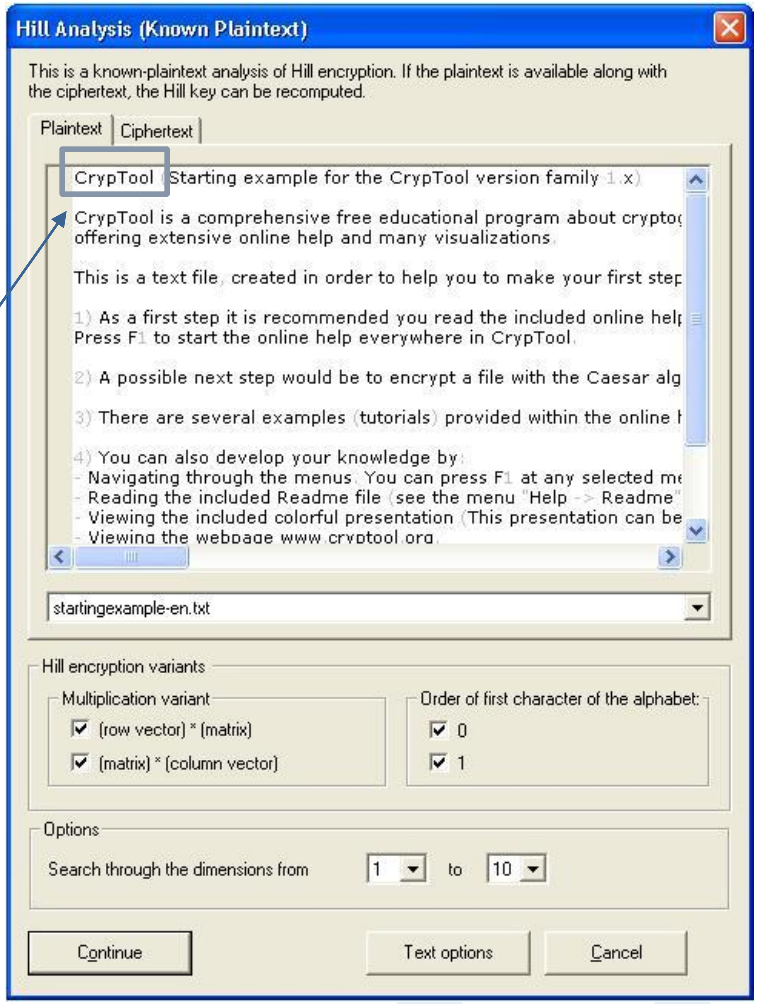
- Select plaintext (startingexample-en.txt)
- Select ciphertext

  (Hill encryption of <startingexample-en.txt>)
- Click “Continue” to search for the key

2. **Reduced plaintext/ciphertext**

- Clear all of the plaintext except the first word (“CrypTool”)
- Clear all of the ciphertext except for the first eight characters (“PnhdJovl”)
- Click “Continue” to reveal the key!

Which length of plaintext/ciphertext is required to find the correct encryption key?

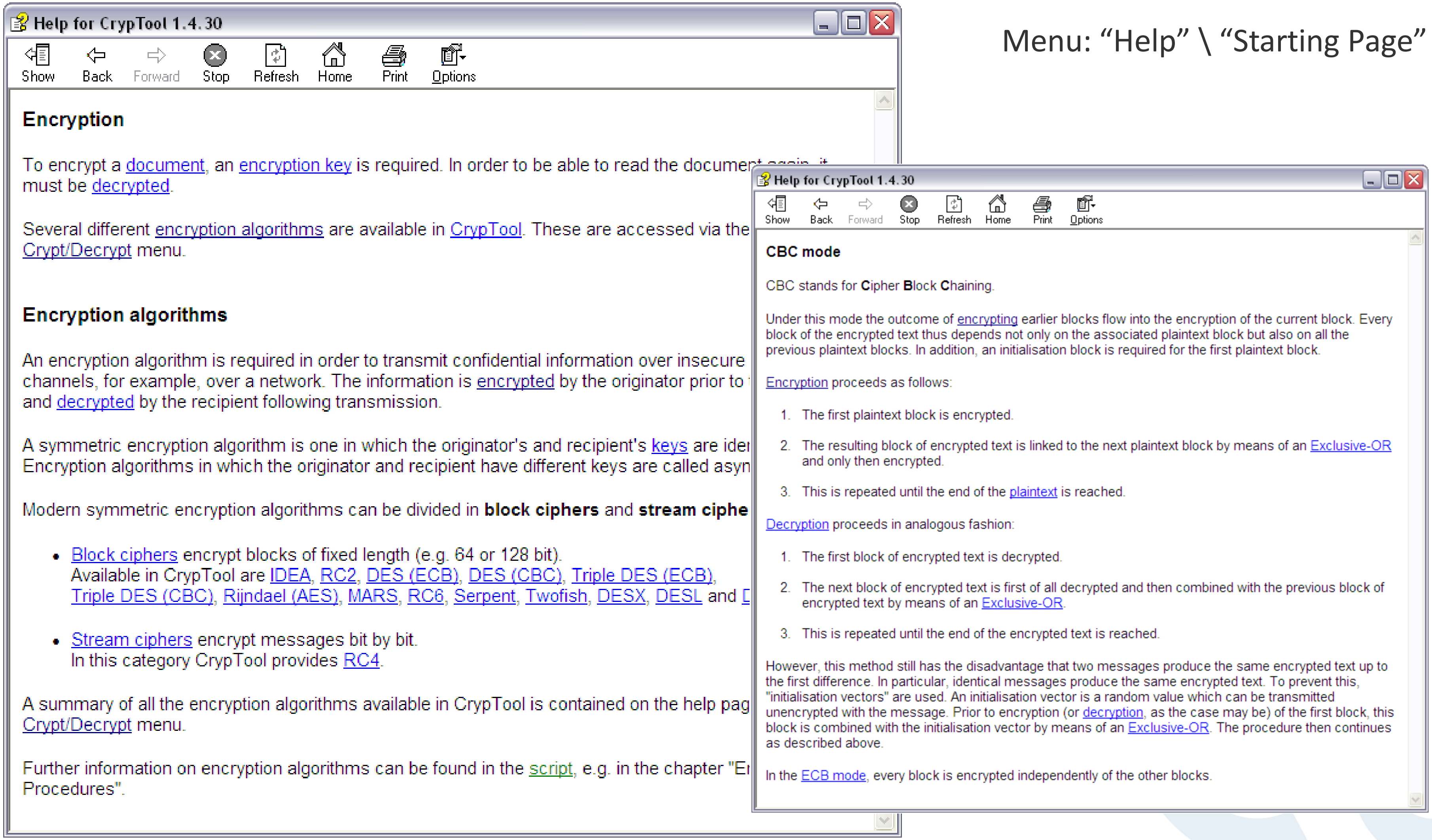


Menu: “Analysis” \ “Symmetric Encryption (classic)” \ “Known Plaintext” \ “Hill…”

Examples (26)

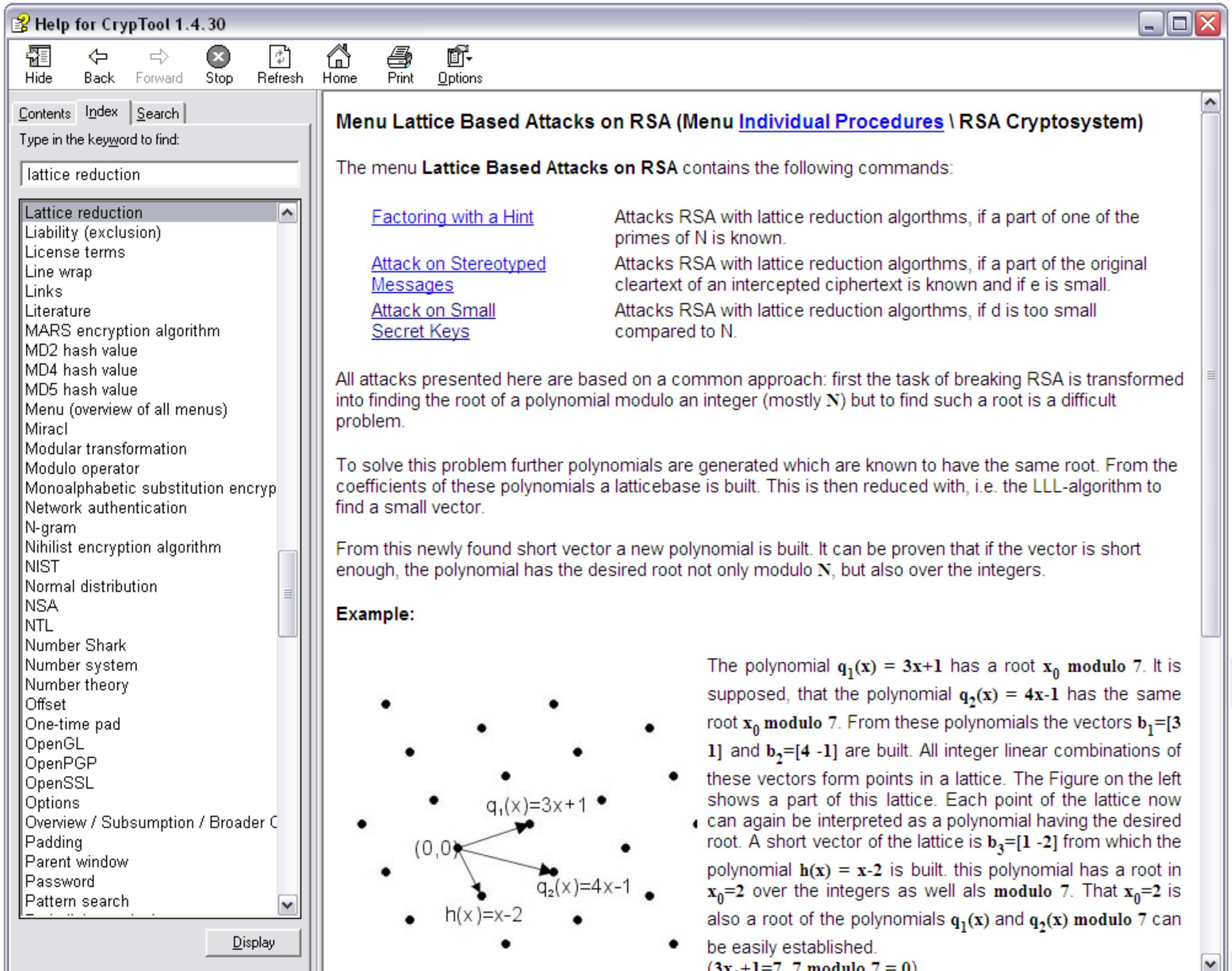
CrypTool online help (1)

Menu: “Help” \ “Starting Page”



Examples (26)

CrypTool online help (2)



Menu Lattice Based Attacks on RSA (Menu Individual Procedures \ RSA Cryptosystem)

The menu **Lattice Based Attacks on RSA** contains the following commands:

Factoring with a Hint	Attacks RSA with lattice reduction algorithms, if a part of one of the primes of N is known.
Attack on Stereotyped Messages	Attacks RSA with lattice reduction algorithms, if a part of the original cleartext of an intercepted ciphertext is known and if e is small.
Attack on Small Secret Keys	Attacks RSA with lattice reduction algorithms, if d is too small compared to N.

All attacks presented here are based on a common approach: first the task of breaking RSA is transformed into finding the root of a polynomial modulo an integer (mostly **N**) but to find such a root is a difficult problem.

To solve this problem further polynomials are generated which are known to have the same root. From the coefficients of these polynomials a latticebase is built. This is then reduced with, i.e. the LLL-algorithm to find a small vector.

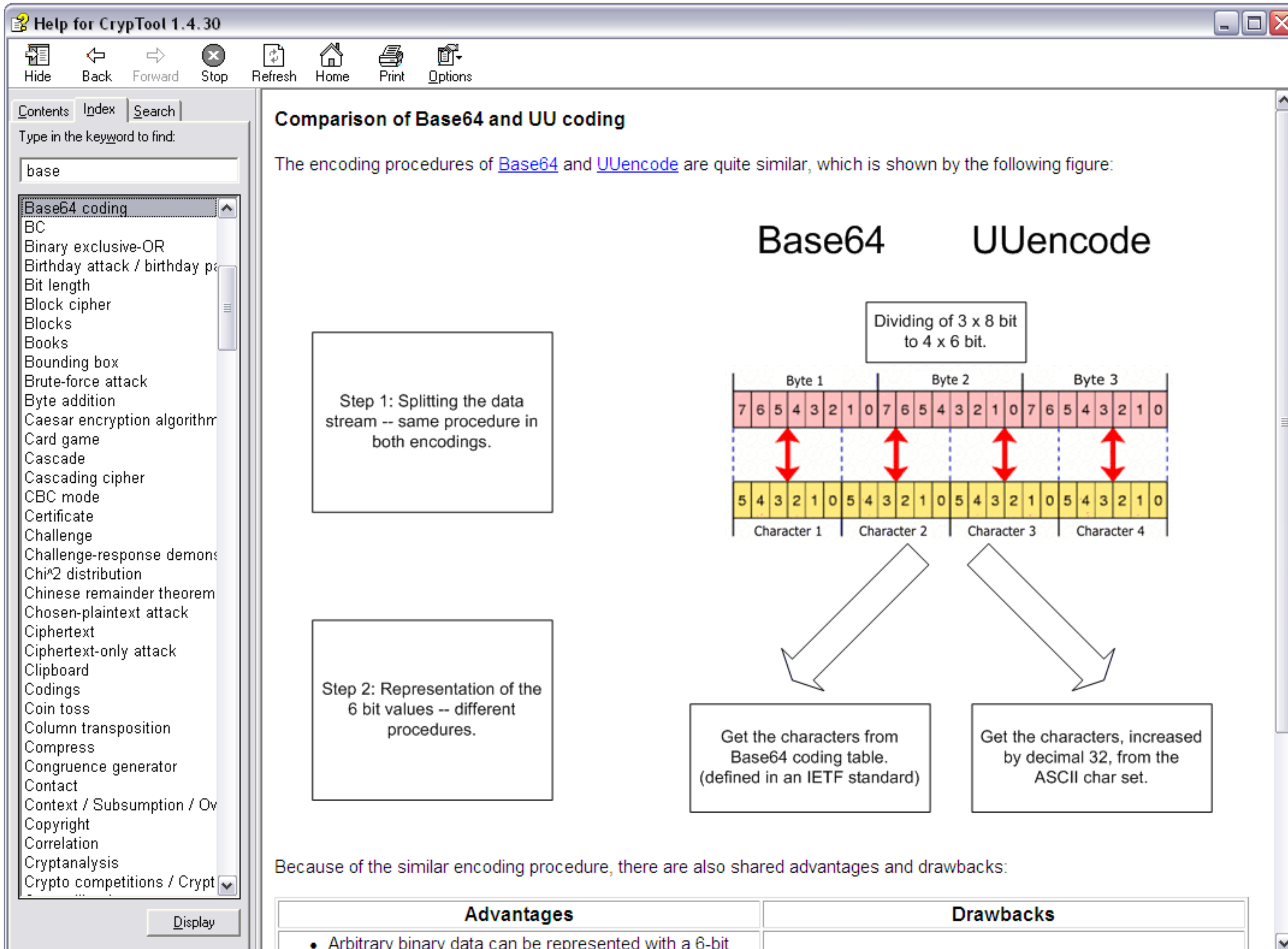
From this newly found short vector a new polynomial is built. It can be proven that if the vector is short enough, the polynomial has the desired root not only modulo **N**, but also over the integers.

Example:

The polynomial $q_1(x) = 3x+1$ has a root x_0 **modulo** 7. It is supposed, that the polynomial $q_2(x) = 4x-1$ has the same root x_0 **modulo** 7. From these polynomials the vectors $b_1=[3\ 1]$ and $b_2=[4\ -1]$ are built. All integer linear combinations of these vectors form points in a lattice. The Figure on the left shows a part of this lattice. Each point of the lattice now can again be interpreted as a polynomial having the desired root. A short vector of the lattice is $b_3=[1\ -2]$ from which the polynomial $h(x) = x-2$ is built. this polynomial has a root in $x_0=2$ over the integers as well als **modulo** 7. That $x_0=2$ is also a root of the polynomials $q_1(x)$ and $q_2(x)$ **modulo** 7 can be easily established.
$(3x_0+1=7,\ 7$ **modulo** $7 = 0)$

Examples (26)

CrypTool online help (3)



Examples (26)

Menu tree of the program CrypTool 1.4.30

- **File**
  - New
  - Open...
  - Close
  - Save
  - Save as...
  - Document Properties...
  - Print...
  - Print Setup...
  - Recent Files
  - Exit
- **Edit**
  - Undo
  - Redo
  - Cut
  - Copy
  - Paste
  - Delete
  - Find/Replace...
  - Find Next
  - Select All
  - Show Key...
  - Parent Window
- **View**
  - Toolbar
  - Status Bar
  - Show as Text
  - Show as HexDump
  - Bar Chart
  - Alphabet
  - End of Line
  - Line Wrap
  - Whitespace
  - Font
    - Arial 8
    - Arial 10
    - Arial 12
    - Arial 14
    - Arial 16
    - Courier 8
    - Courier 10
    - Courier 12
    - Courier 14
    - Courier 16
  - Format Text Document...
  - Show Box (cube's borderlines)
- **Encrypt/Decrypt**
  - Symmetric (classic)
    - Caesar / Rot-13...
    - Vigenère...
    - Hill...
    - Substitution / Atbash...
    - Playfair...
    - ADFGVX...
    - Byte Addition...
    - XOR...
    - Vernam / OTP...
    - Homophone...
    - Permutation / Transposition...
    - Solitaire...
    - Scytale / Rail Fence...
  - Symmetric (modern)
    - IDEA...
    - RC2...
    - RC4...
    - DES (ECB)...
    - DES (CBC)...
    - Triple DES (ECB)...
    - Triple DES (CBC)...
    - Rijndael (AES)...
    - Further Algorithms
      - MARS...
      - RC6...
      - Serpent...
      - Twofish...
      - DESX...
      - DESL...
      - DESXL...
    - AES (self extracting)...
  - Asymmetric
    - RSA Encryption...
    - RSA Decryption...
    - RSA Demonstration...
  - Hybrid
    - RSA-AES Encryption...
    - RSA-AES Decryption...
    - ECC-AES Encryption...
    - ECC-AES Decryption...
- **Digital Signatures/PKI**
  - PKI
    - Generate/Import Keys...
    - Display/Export Keys...
  - Sign Document...
  - Verify Signature...
  - Extract Signature
  - Signature Demonstration (Signature Generation)...
- **Indiv. Procedures**
  - Hash
    - MD2
    - MD4
    - MD5
    - SHA
    - SHA-1
    - SHA-256
    - SHA-512
    - RIPEMD-160
    - Hash Value of a File...
    - Hash Demonstration...
    - Key Generation from Password (PKCS #5)...
    - Generation of HMACs...
  - RSA Cryptosystem
    - Prime Number Test...
    - Generate Prime Numbers...
    - Factorization of a Number...
    - RSA Demonstration...
    - Signature Demonstration (Signature Generation)...
    - Lattice Based Attacks on RSA
      - Factoring with a Hint...
      - Attack on Stereotyped Messages...
      - Attack on Small Secret Keys...
  - Protocols
    - Diffie-Hellman Demonstration...
    - Network Authentication...
    - Secure E-Mail with S/MIME...
  - Chinese Remainder Theorem Applications
    - Astronomy and Planetary Motion...
    - Modular Forward and Backward Transformation...
    - Secret Sharing by CRT...
  - Visualization of Algorithms
    - Caesar...
    - Vigenère...
    - Nihilist...
    - DES...
    - AES
      - Rijndael Animation...
      - Rijndael Inspector...
      - Rijndael Flow Visualization...
    - Enigma...
  - Secret Sharing Demonstration (Shamir)...
  - Tools
    - Codes
      - Base64 Encode/Decode
        - Base64 Encode
        - Base64 Decode
      - UU Encode/Decode
        - UU Encode
        - UU Decode
      - Decode ASN.1 Code of a Document
    - Compress
      - Zip
      - UnZip
    - Generate Random Numbers...
    - Password Quality Meter...
    - Password Entropy...
  - Educational Games
    - Number Shark
  - Number Theory - Interactive
    - Learning Tool for Number Theory...
    - Point Addition on Elliptic Curves...
    - Compute Mersenne Numbers...
- **Analysis**
  - Tools for Analysis
    - Entropy
    - Floating Frequency
    - Histogram
    - N-Gram...
    - Autocorrelation
    - Periodicity
  - Symmetric Encryption (classic)
    - Ciphertext-Only
      - Caesar
      - Vigenère
      - Vigenère (Analysis according to Schroedel)...
      - ADFGVX...
      - Substitution...
      - Solitaire...
      - Byte Addition
      - XOR / Vernam
    - Known Plaintext
      - Hill...
      - Single Column Transposition...
    - Manual Analysis
      - Substitution...
      - Playfair...
      - Solitaire...
  - Symmetric Encryption (modern)
    - IDEA...
    - RC2...
    - RC4...
    - DES (ECB)...
    - DES (CBC)...
    - Triple DES (ECB)...
    - Triple DES (CBC)...
    - Rijndael (AES)...
    - Further Algorithms
      - MARS...
      - RC6...
      - Serpent...
      - Twofish...
      - DESX...
      - DESL...
      - DESXL...
  - Asymmetric Encryption
    - Factorization of a Number...
    - Lattice Based Attacks on RSA
      - Factoring with a Hint...
      - Attack on Stereotyped Messages...
      - Attack on Small Secret Keys...
    - Side-Channel Attack on "Textbook RSA"...
  - Hash
    - Attack on the Hash Value of the Digital Signature...
  - Analyze Randomness
    - Frequency Test...
    - Poker Test...
    - Runs Test...
    - Serial Test...
    - FIPS PUB-140-1 Test Battery
    - Vitányi
    - 3D Visualization...
- **Options**
  - Plot Options...
  - Analysis Options...
  - Text Options...
  - Starting Options...
- **Window**
  - Cascade
  - Tile
  - Arrange Icons
  - Close All
- **Help**
  - Starting Page
  - Index
  - Scenarios (Tutorials)
  - Readme
  - Script
  - Presentation
  - About CrypTool...

Content

Future CrypTool Development (1)

Examples of what is coming after the release of CrypTool 1.4.30 (see readme for details)

CT1	FIPS test with the ability to analyze packets with lengths other than 2500 bytes, etc.
JCT	Tri-partite key agreements
JCT	Visualization of the interoperability between S/MIME and OpenPGP formats
JCT	Entropy analysis
JCT	Fleissner grille, Autokey Vigenère, interactive cryptanalysis of classic ciphers
JCT	Analysis of transposition ciphers using the ACO algorithm
JCT	Visualization of zero-knowledge proofs
JCT	Visualization of Quantum Key Agreement, BB84 protocol
JCT	Visualization of the SETUP attack against RSA key generation (Kleptography)
JCT	Action history with the ability to create and replay any given cipher cascade
CT2	Comprehensive visualization on the topic of prime numbers
CT2	GNFS (General number field sieve)
CT2	Encryption and automated cryptanalysis of the Enigma machine (and possibly of Sigaba as well)
CT2	Cube attack (I. Dinur and A. Shamir: “Cube Attacks on Tweakable Black Box Polynomials”, 2008)
CT2	Demonstration of Bleichenbacher’s and Kuehn’s RSA signature forgery
CT2	Demonstration of virtual credit card numbers (as an educational tool against credit card abuse)
CT2	WEP encryption and WEP analysis
CT2	Mass pattern search
CT2	Framework for distributed cryptanalysis
CT2	Demonstration of SOA security (SOAP messages with WS-Security)
CT2	Framework to create and analyze LFSR stream ciphers
CT2/JCT	Creation of a command-line interface for batch processing
CT2/JCT	Modern pure plugin architecture with plugin reloading capability
All	Expanded parameterization and flexibility of present algorithms
Ideas	Visualization of the SSL protocol // Demonstration of visual cryptography

CT1 = CrypTool 1.x

New versions:

CT2 = CrypTool 2.0

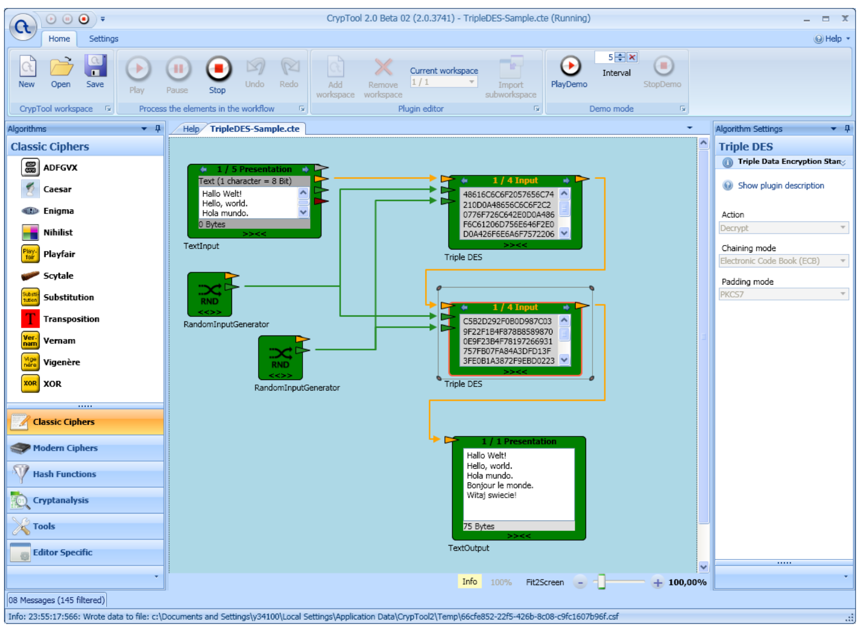
JCT = JCrypTool

(both introduced on the next slides)

Future CrypTool Development (2)

In Progress: the two successor versions of CT v1 (see readme file)

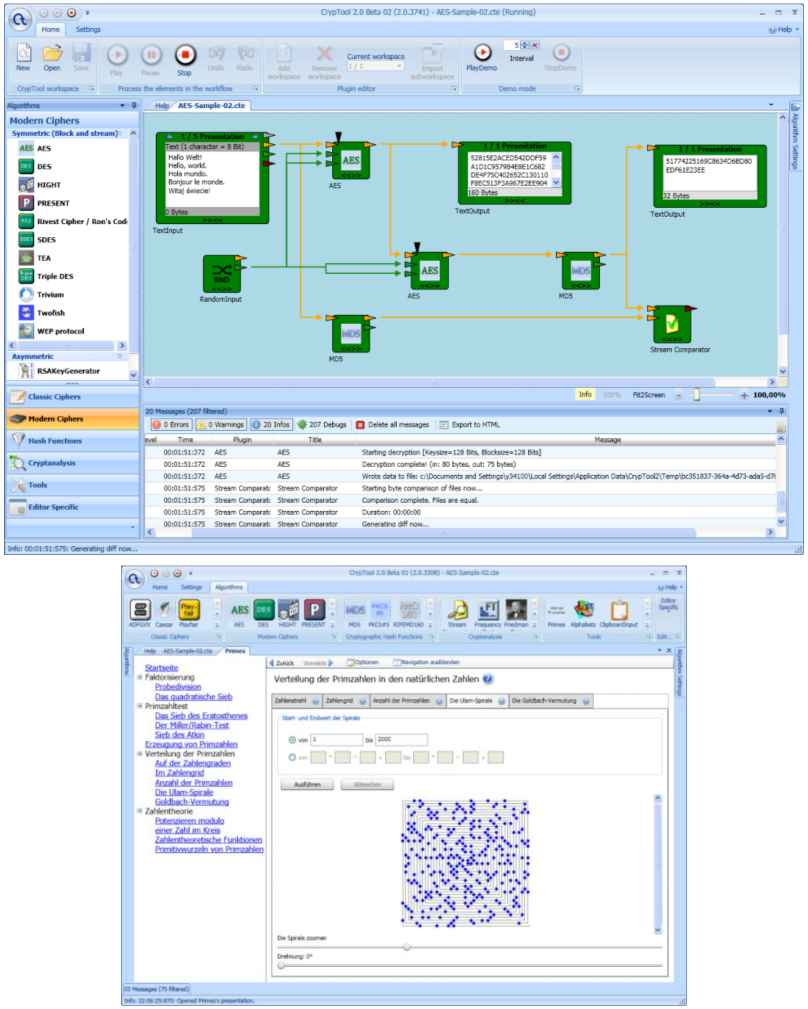
1. JCT: Port and redesign of CrypTool in Java / SWT / Eclipse 3.6 / RCP
   – see: http://jcryptool.sourceforge.net
   – Release Candidate RC3 is available since July 2010.
2. CT2: Port and redesign of the C++ version with C# / WPF / VS2010 / .NET 4.0
   – direct successor of current release: allows visual programming, etc.
   – Download from: http://cryptool2.vs.uni-due.de/index.php?page=14&lm=1&ql=4
   – Beta 3 is available since August 2010 (since July 2008, betas are continuously updated, nightly builds).



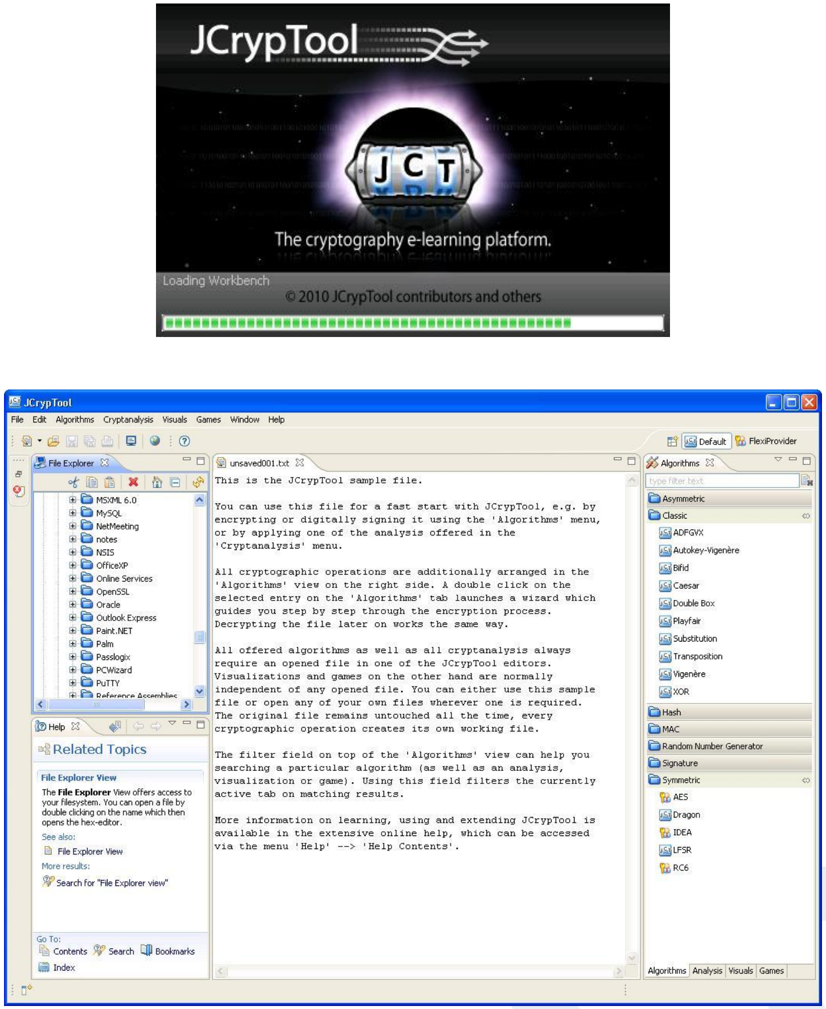
CrypTool 2 (CT2)

JCrypTool (JCT)

Future CrypTool Development (3)



CrypTool 2 (CT2)



JCrypTool (JCT)

CrypTool as a Framework

Proposal

- Reuse the comprehensive set of algorithms, included libraries, and interface elements as a foundation.
- Free training to help get started with CrypTool development.
- Advantage: code written for university theses or other projects will not simply disappear, but rather be further maintained.

Current development environment for *CT1*: Microsoft Visual Studio C++ , Perl, Subversion Source Code Management

- CrypTool 1.4.30: Visual C++ .NET (= VC++ 9.0)(= Visual Studio 2008 Standard)
- Description for developers: see readme-source.txt
- Sources and binaries of release versions are available for download.
  To get sources of current betas, please see the Subversion repository.

Development environments for *CT2* and *JCT*

- CT2 – C# version: .NET with Visual Studio 2010 Express Edition (free) and WPF
- Java – Java version: Eclipse 3.6, RCP, SWT (free)

CrypTool – Request for Contribution

Every contribution to the project is highly appreciated

- Feedback, criticism, suggestions, and ideas
- Integration of additional algorithms, protocols, analysis (consistency and completeness)
- Development assistance (programming, layout, translation, testing)
- For the current C/C++ project
- For the new projects (preferred):
  - C# project: “CrypTool 2.0” = CT2
  - Java project: “JCrypTool” = JCT
- In particular, university faculties that use CrypTool for educational purposes are invited to contribute to the further development of CrypTool.
- Samples of open tasks are on the following developer pages:
  - CT2: See the list http://cryptool2.vs.uni-due.de/, Volunteer, Open Tasks
  - JCT: See the wiki http://sourceforge.net/apps/mediawiki/jcryptool/index.php?title=CurrentDevelopment
- Users that make a significant contributions can request to be referenced by name in the online help, the readme file, the about dialog, and/or on the CrypTool website.
- CrypTool v1 is currently being downloaded over 6000 times per month from the CrypTool website.
  Just over half of these downloads are of the English version.
  The betas of the two successors are already being downloaded over 1000 times a month each.

CrypTool – Summary

THE e-learning program for cryptology

Successfully active as an open-source project for over ten years

Over 400,000 total downloads

Widespread international usage in schools, universities, companies, and government agencies

Extensive online help and documentation

Available for free

Multilingual

Contact

Prof. Bernhard Esslinger

University of Siegen
Faculty 5, Economics and Business Computing

Deutsche Bank AG
Director, IT Security Manager

esslinger@fb5.uni-siegen.de

www.cryptool.org

www.cryptool.com

www.cryptool.de

www.cryptool.es

www.cryptool.pl

Additional contacts: See readme within the CrypTool folder

Content

Additional Literature

As an introduction to cryptology – and more

- **Klaus Schmeh, "*Codeknacker gegen Codemacher. Die faszinierende Geschichte der Verschlüsselung*", 2nd edition, 2007, W3L [German]**
- **Simon Singh, *"The Codebook"*, 1999, Doubleday**
- **Johannes Buchmann, "*Introduction to Cryptography*", 2nd edition, 2004, Springer**
- **Paar / Pelzl: *"Understanding Cryptography – A Textbook for Students and Practitioner"*, 2009, Springer**
- **[HAC] Menezes / van Oorschot / Vanstone, *"Handbook of Applied Cryptography"*, 1996, CRC Press**
- **Van Oorschot / Wiener, *"Parallel Collision Search with Application to Hash Functions and Discrete Logarithms"*, 1994, ACM**
- **Antoine Joux, "Algorithmic Cryptanalysis", 2009, Chapman & Hall/CRC Cryptography and Network Security Series**
- **Additional cryptography literature – see also the links at the CrypTool web page and the literature in the CrypTool online help (e.g. by Wätjen, Salomaa, Brands, Schneier, Shoup, Stamp/Low, etc.)**
- **Importance of cryptography in the broader context of IT security and risk management**
  - See e.g. Kenneth C. Laudon / Jane P. Laudon / Detlef Schoder, "Wirtschaftsinformatik", 2nd edition, 2009, Pearson, chapter 15 about IT Security [German]
  - Wikipedia: http://en.wikipedia.org/wiki/Risk_management
  - CrypTool site: http://cryptool.com/index.php/en/cryptool-for-awareness-aboutmenu-74.html

www.cryptool.org / .com / .de / .es / .pl

About

- CrypTool Introduction
- CrypTool in Education
- CrypTool for Awareness
- Coverage in Print Media
- Awards
- Contributors
- Related Projects
- Contact

Features

- CrypTool Features
- Roadmap

Media

- Screenshots
- Screencast

Documentation

- Presentations
- Script
- Crypto History
- Links / Books

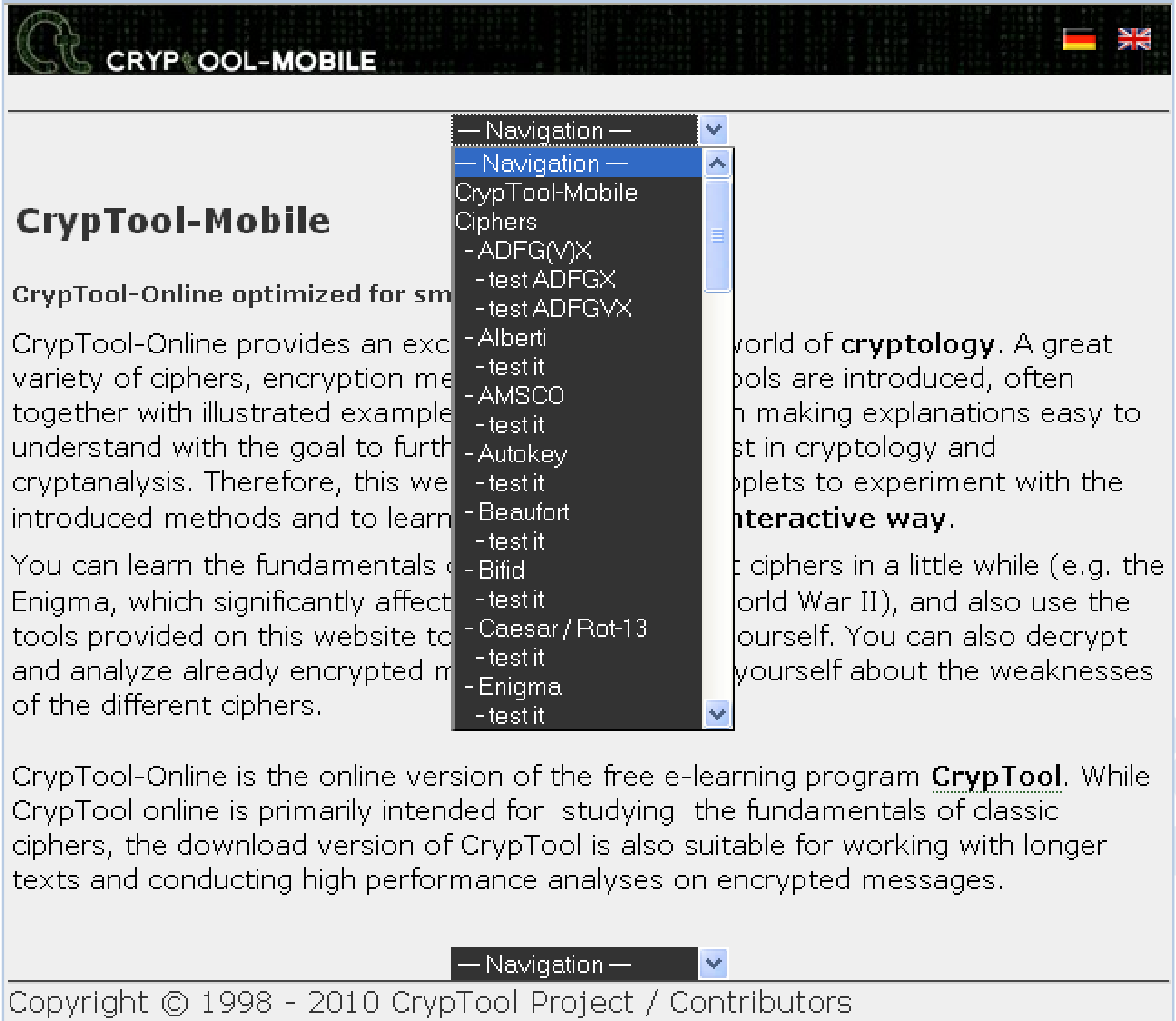
www.cryptool-online.org

The members in the family of CrypTool related websites:

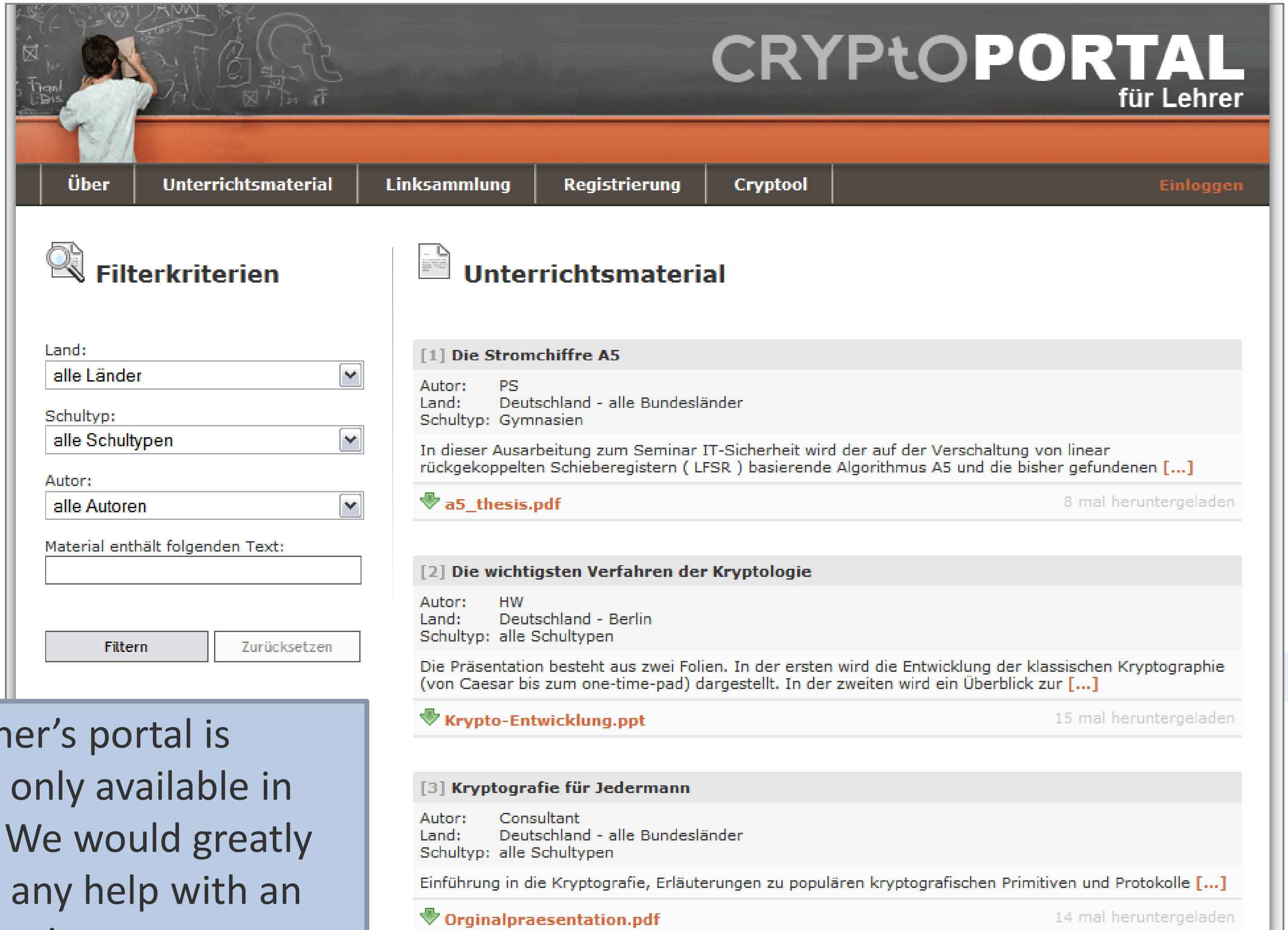
- **CrypTool** site (CT1)
- **CT2 developer** site
- **JCT developer** site
- **CrypTool-Online + CrypTool-Mobil** (experiment with cryptography from within your browser, and with your smart phone)
- **CryptoPortal** for teachers (currently only in German)
- **Mystery Twister C3** (MTC3) is an international crypto challenge contest.

m.cryptool.org

Experiment with cryptography from within smart phone

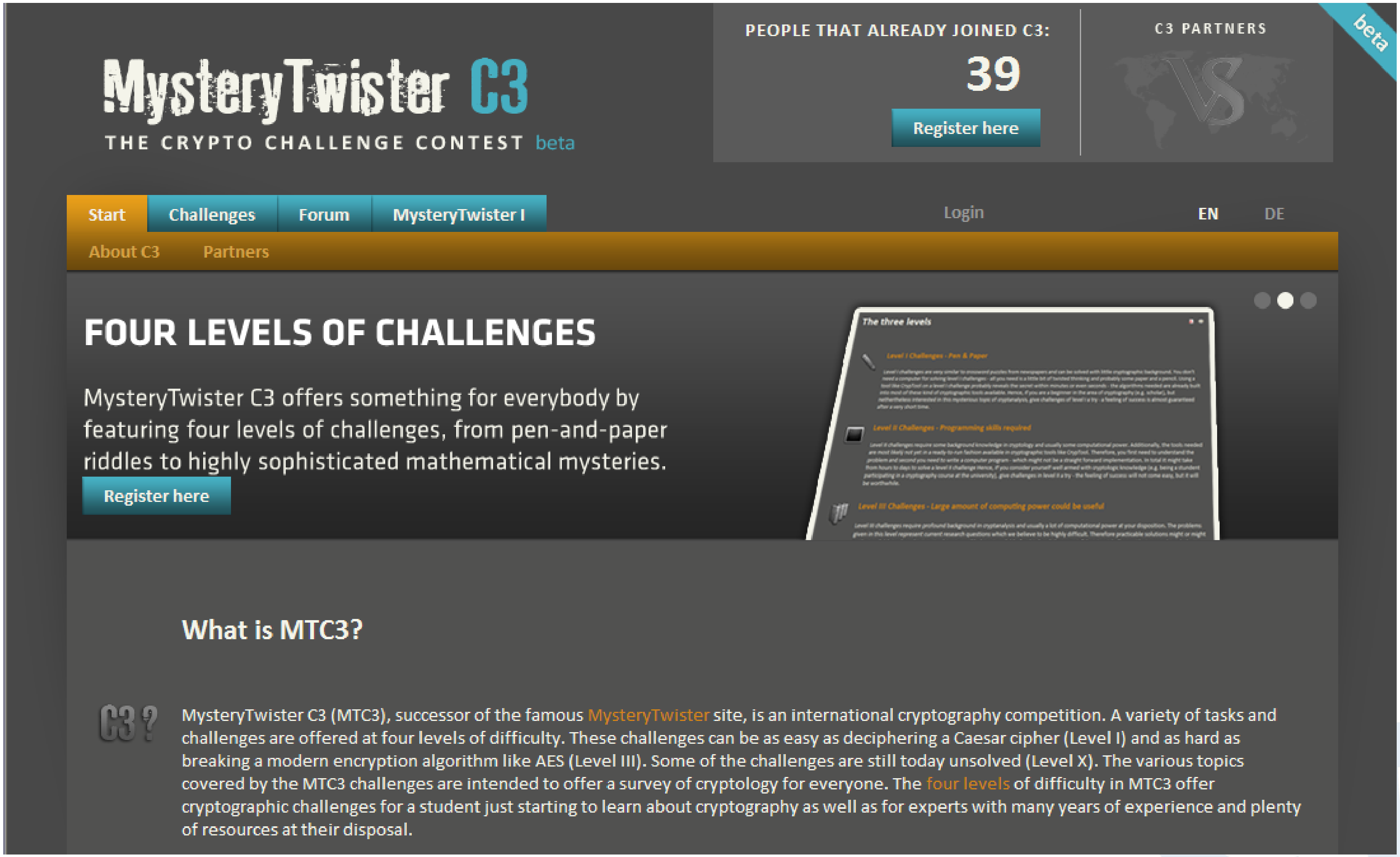


www.cryptoportal.org



The teacher's portal is currently only available in German. We would greatly welcome any help with an English version.

www.mysterytwisterc3.org



Mystery Twister C3 (MTC3) is an international crypto challenge contest.

Downloads: Software and the CrypTool Script