CRYPTOLOGY WITH CRYPTOOL v 1.4.21

Introduction to Cryptography und Cryptanalysis

Scope, Technology and Future of CrypTool

Prof. Bernhard Esslinger and CrypTool-Team, 2008

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

Content (I)

I. CrypTool and Cryptology – Overview

1. Definition and relevance of cryptology
2. The CrypTool project
3. Examples of classical encryption methods
4. Insights from cryptography development

II. CrypTool Features

1. Overview
2. Interaction examples
3. Challenges for developers

III. Examples

1. Encryption with RSA / Prime number test / Hybrid encryption and digital certificates / SSL
2. Digital signature visualised
3. Attack on RSA encryption (modul N too short)
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)

(…)

Content (II)

III. Examples

10. RSA attack using lattice reduction
11. Random analysis with 3-D visualisation
12. Secret Sharing using the Chinese Remainder Theorem (CRT) and Shamir
13. Implementation of CRT in astronomy (solving linear modular equation systems)
14. Visualisation of symmetric encryption methods using ANIMAL
15. Visualisation of AES
16. Visualisation of Enigma encryption
17. Generation of a message authentication code (MAC)
18. Hash demonstration
19. Learning tool for number theory and asymmetric encryption
20. Point addition on elliptic curves
21. Password quality mete (PQM) and password entropy
22. Brute-force analysis
23. CrypTool online help

IV. Project / Outlook / Contact

Content

I. **CrypTool and Cryptology – Overview**

II. CrypTool Features

III. Examples

IV. Project / Outlook / Contact

Definition Cryptology and Cryptography

Cryptology *(from the Greek kryptós, "hidden," and lógos, "word") is the science of secure (generally secret) communications. This security obtains from legitimate users, the transmitter and the receiver, being able to transform information into a cipher by virtue of a key -- i.e., 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 contains not only mechanisms for encryption but also for integrity, electronic signatures, random numbers, secure key exchange, secure containers, electronic voting and electronic money, and has achieved to render a broad range of applications in modern life.*

Source: Britannica (www.britannica.com)

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

Relevance of Cryptography

Examples for Cryptography Usage

- Phone cards, cell phones, remote controls
- Cash machines, money transfer between banks
- Electronic cash, online banking, secure eMail
- Satellite TV, Pay TV
- Immobiliser systems in cars
- Digital Rights Management (DRM)

- Cryptography is no longer limited to agents, diplomats or the military. Cryptography is a modern, mathematically characterised science.
- Breakthrough for cryptography started with the broad use of the Internet
- For companies and governments it is important that systems are secure and

… users (clients, employees) have a certain understanding and awareness for IT security!

Cryptography – Objectives

- **Confidentiality**
  - Information can practically not be made available or disclosed to unauthorized individuals, entities or processes.
- **Authentication**
  - Authentication ensures that users are identified and those identities are appropriately verified.
- **Integrity**
  - Integrity ensures that data has not been altered or destroyed in an unauthorized manner.
- **Non-Repudiation**
  - The principle that, afterwards, it can be proven that the participants of a transaction did really authorize the transaction and that they have no means to deny their participation.

The CrypTool Project

- Origin in awareness program of a bank (in-firm training)
  → **Awareness for employees**
- Developed in co-operation with universities (improving education)
  → **Media didactic approach and standard oriented**

1998 **Project start – effort more than 17 man-years since then**

2000 CrypTool available as **freeware**

2002 CrypTool on **Citizen-CD-ROM from BSI** (German Information Security Agency)

2003 CrypTool becomes **Open-Source** – Hosting by University of Darmstadt (Prof. Eckert)

2007 CrypTool available in German, English, Polish und Spanish

2008 .NET version and Java version – Hosted by University of Duisburg (Prof. Weis)

- **Awards**

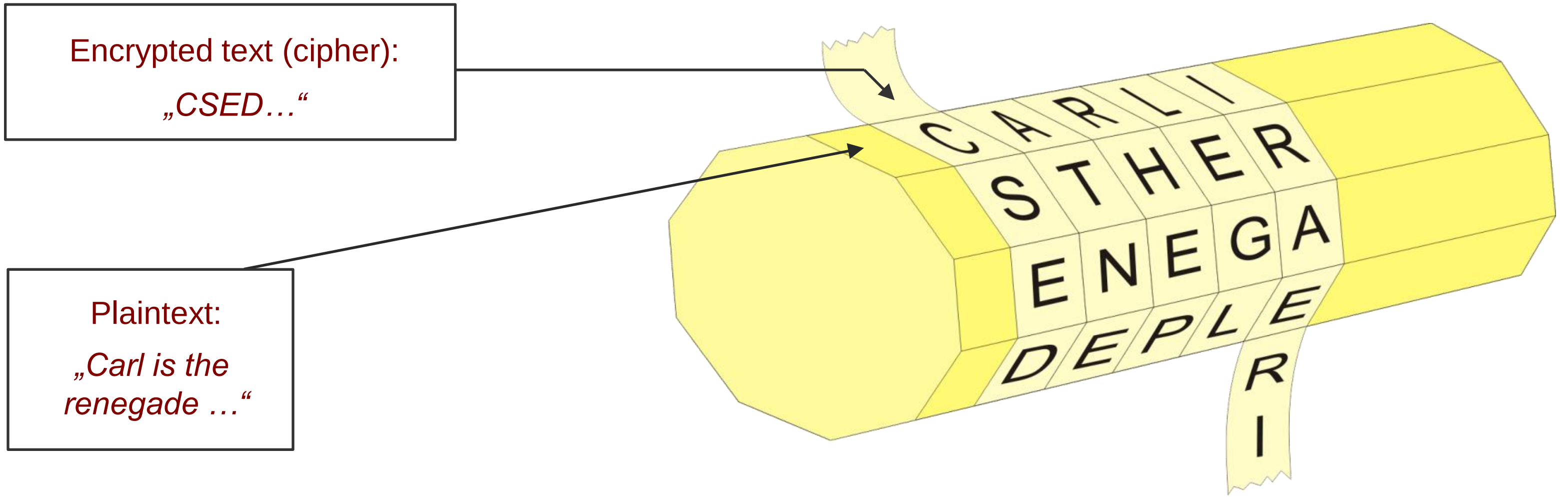
2004	TeleTrusT	(TTT Förderpreis)
2004	NRW	(IT Security Award NRW)
2004	RSA Europe	(Finalist of European Information Security Award 2004)
2008	“Selected Landmark” in initiative “Germany – Land of Ideas"	

- **Developers**
  – Developed by people from companies and universities in different countries
  → Additional project members or usable sources are always appreciated (currently there are around 30 people working on CrypTool world wide).

Examples of Early Cryptography (1)

Ancient encryption methods

- **Tattoo on a slave's head concealed by re-grown hair**
- **Atbash** (around 600 B.C.)
    - Hebrew secret language, reversed alphabet
- **Scytale from Sparta** (500 B.C.)
    - Described by Greek historian/author Plutarch (45 - 125 B.C.)
    - Two cylinders (wooden rod) with identical diameter
    - Transposition (plaintext characters are re-sorted)



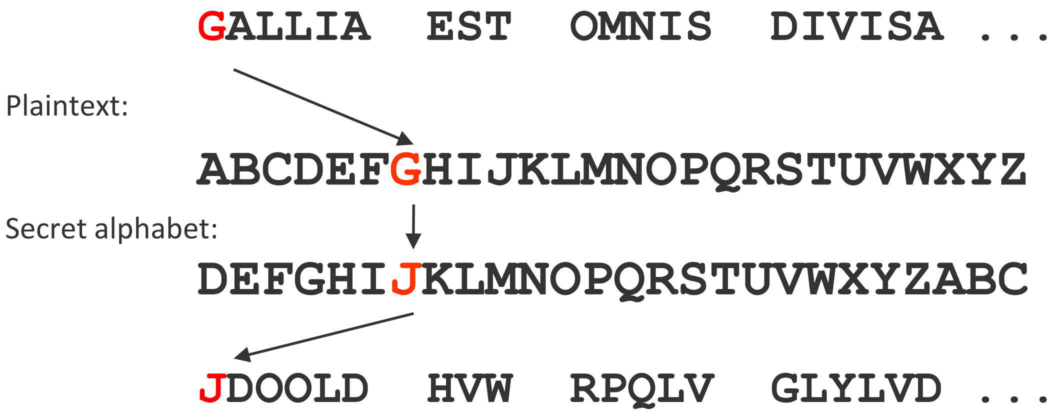
Encrypted text (cipher):
„CSED…“

Plaintext:
„Carl is the renegade …“

Examples of Early Cryptography (2)

Symmetric Caesar encryption

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



GALLIA EST OMNIS DIVISA . . .

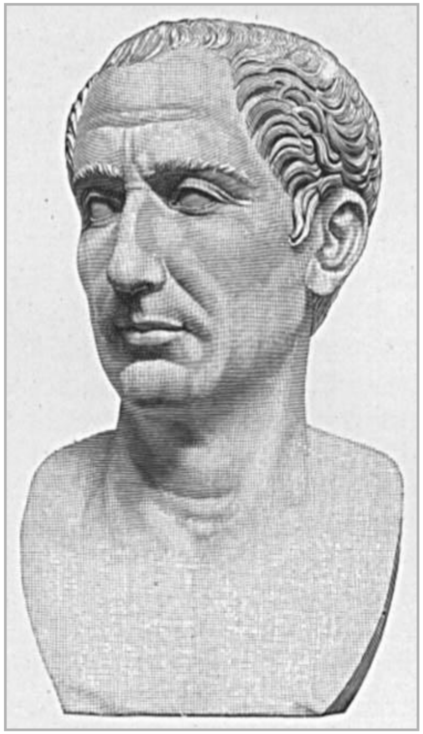
Plaintext:

ABCDEFGHIJKLMNOPQRSTUVWXYZ

Secret alphabet:

DEFGHIJKLMNOPQRSTUVWXYZABC

JDOOLD HVW RPQLV GLYLVD . . .



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

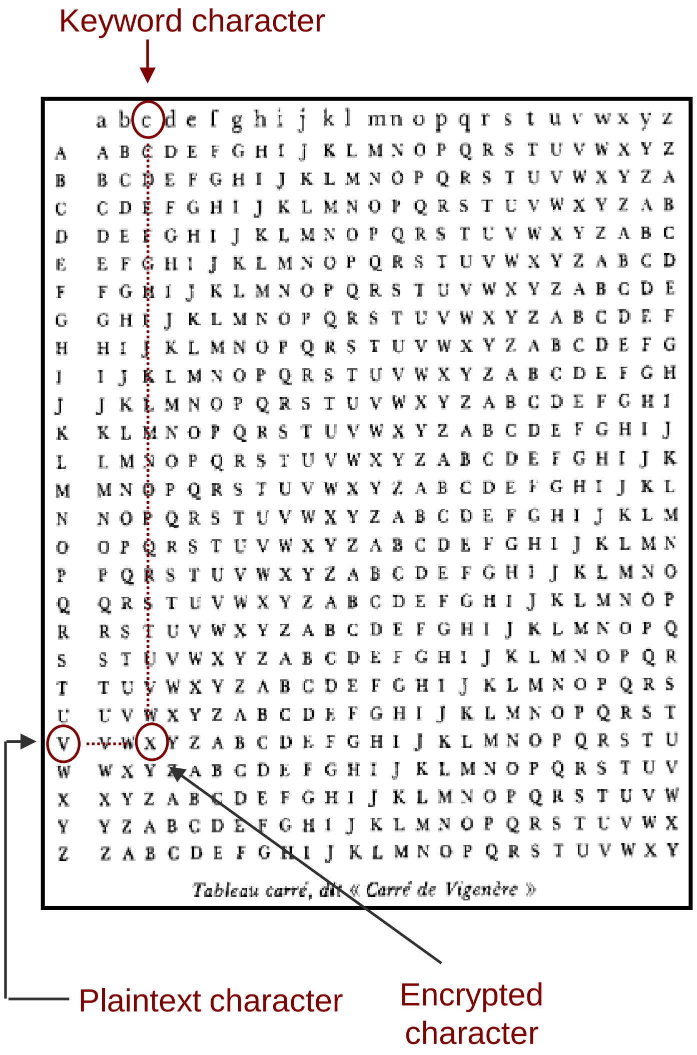
Presentation with CrypTool via the following menus:

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

Examples of Early Cryptography (3)

Symmetric Vigenère encryption

- **Vigenère-Encryption** (Blaise de Vigenère, 1523-1596)
- Encryption with a key word 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 key word character (c). The next plaintext character (I) is replaced by the character in the corresponding row and in the column of the next key word character (h), and so on.
- If all characters of the key word have been used, then the next key word character is the first key character.
- **Attack** (via Kasiski test): 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.


Keyword character
Plaintext character
Encrypted character

	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

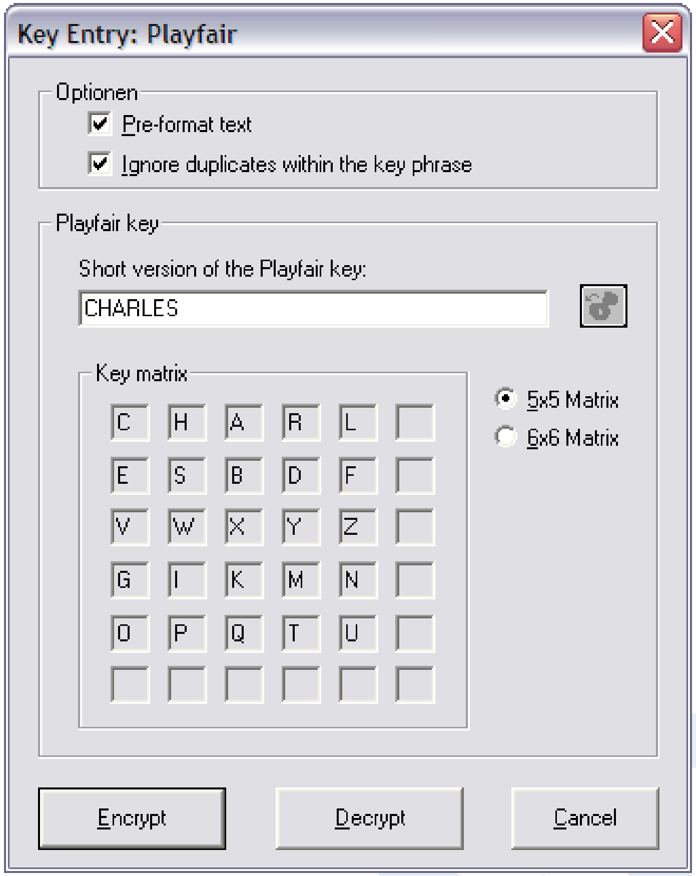
Tableau carré, dit « Carré de Vigenère »

Examples of Early Cryptography (4)

Other symmetric 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)

    (Transposition / very effective)



Key Entry: Playfair

Optionen

☑ Pre-format text

☑ Ignore duplicates within the key phrase

Playfair key

Short version of the Playfair key: CHARLES

Key matrix

C	H	A	R	L	
E	S	B	D	F	
V	W	X	Y	Z	
G	I	K	M	N	
O	P	Q	T	U	

● 5x5 Matrix

○ 6x6 Matrix

Cryptography in Modern Times

Cryptography developments in the last 100 years till 1970

Classic methods

- are still in use today.
  (since, not everything can be done by a computer…)
- and their principals of transposition and substitution are inputs for the design of modern algorithms: combination of simple operation (a type of multiple encryption, a so called cascades of ciphers), on bit level, block cipher, rounds.

Encryption becomes

- more **sophisticated**,
- **mechanised** or **computerised** and
- remains **symmetric**.

Examples of the First Half of the 20th Century

Mechanical encryption machines (rotor machines)

Enigma Encryption (Arthur Scherbius, 1878-1929)

- More than 200000 machines have been used in WW2
- The rotating cylinder set causes, that every character of the text becomes encrypted with a new permutation.
- Broken by massive effort of cryptography experts (around 7000 persons in UK) with decryption machines, captured original Enigmas and by intercepting daily status reports (e.g. weather reports).
- **Consequences of this successful crypto analysis:**
  "In general the successful crypto analysis of the engima encryption has been a strategic advantage, that has played a significant role in winning the war. Some historians assume that the break of the enigma code has 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** (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” (Caesar for example)
  - Kerckhoffs principle:
    The secret lies within the key and not within the algorithm or „No security through obscurity“
- **One-Time Pad – Shannon / Vernam**
  - Demonstrably theoretically secure, but not usable in reality (only red phone)
- **Shannons concepts: Confusion and Diffusion**
  - Relation between M, C and K has to be as complex as possible (M=message, C=cipher, K=key)
  - Every cipher text character should depend on as many plaintext characters and as many character of encryption key
  - „Avalanche effect“(small modification, big impact)
- **Trapdoor function** (one-way function)
  - Fast in one direction, not in the opposite direction (without secret information)
  - Having the secret the opposite direction works (access to the trapdoor)

Examples for a Breach of the Kerckhoffs Principle

Secret lies within the key and not within 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 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://wired.lycos.com/news/politics/0,1283,32900,00.html]

Sample of a One-Time Pad Adaptation

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

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 **S_n = n * (n-1) / 2** keys.

This means

n = 100 persons require

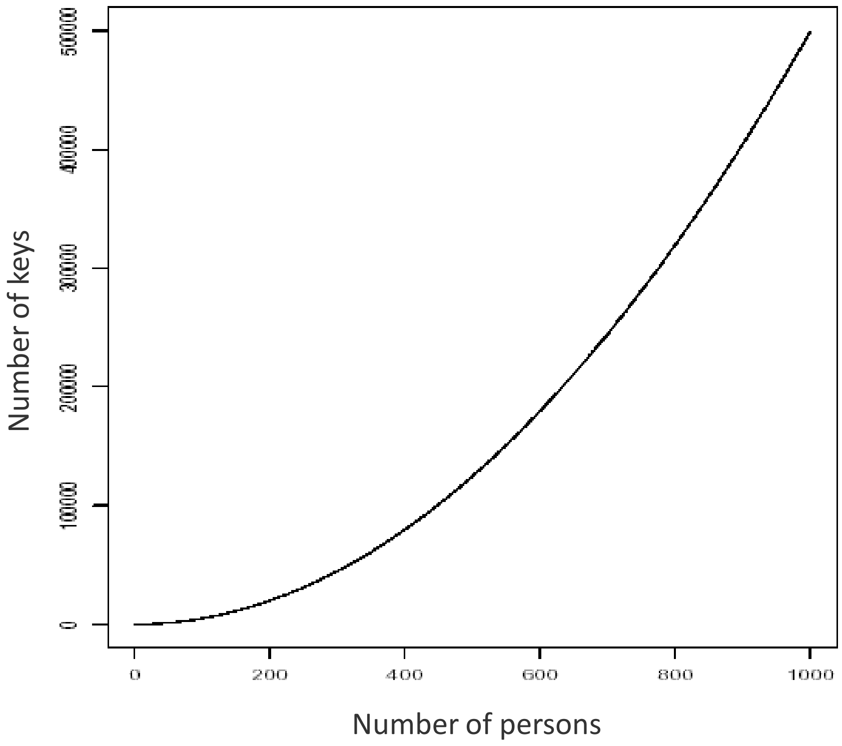
S_{100} = 4.950 keys; and

n = 1.000 persons require

S_{1000} = 499.500 keys.

⇨ factor 10 more persons,
factor 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: Sender and receiver need same secret.
- New: Every member needs a key pair (Solution of the key distribution problem)

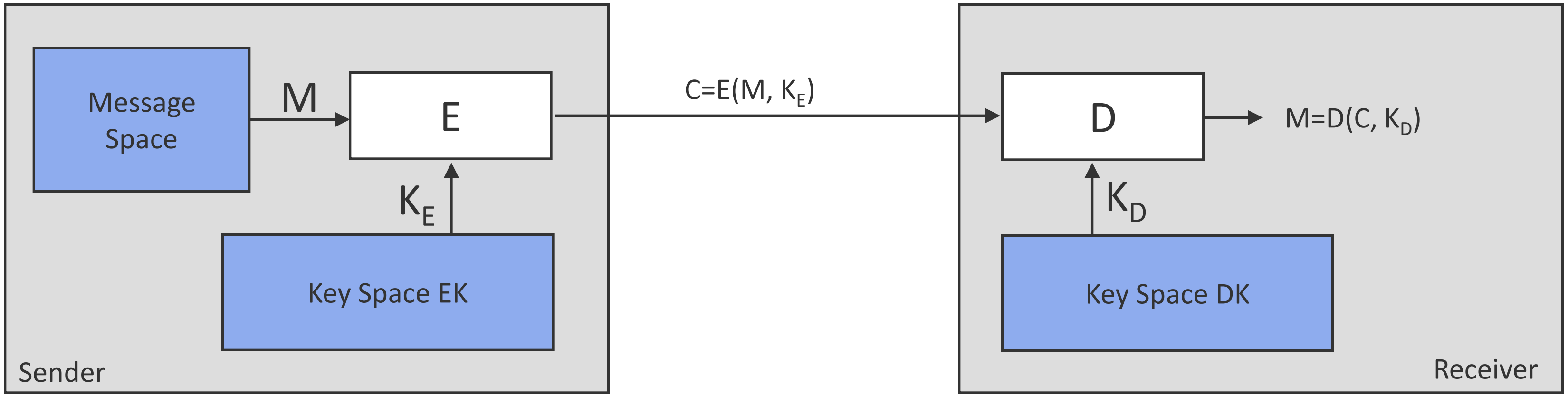
- **Asymmetric encryption**
  - „Everyone can lock a padlock or can drop a letter in a mail box.“
  - MIT, 1977: Leonard Adleman, Ron Rivest, Adi Shamir (well known as RSA)
  - GCHQ Cheltenham, 1973: James Ellis, Clifford Cocks (admitted in public December 1997)

- **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!

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 based on **mathematics**
  - Still new symmetric encryption methods such as AES (better performance and shorter key length compared to the asymmetric methods purely based on mathematical problems).
- The security of encryption methods heavily depends on the current status 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-200 (CrypTool script, chapter 4.11.3)
  - Very high activity in current research:

    Factorisation, non-parallelizable algorithm (because of quantum computing), better understanding of protocol weaknesses and random generators, ...).
- Serious mistake: "Real mathematics has no effects on the war."
  (G.H. Hardy, 1940)
- Vendors discover security as 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

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 deeper cryptography knowledge)
- Contains nearly all state-of-the-art cryptography functions
- Easy entry into modern and classical cryptography
- Not a *"hacker tool"*

2. Why CrypTool?

- Origin in 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
- *Prerequisite*: PC knowledge
- *Preferable*: Interest in mathematics and/or programming

Content of the Program Package

German, English, Polish and Spanish

CrypTool program

- All functions integrated in a *single* program with consistent graphical interface
- Runs on Win32
- Cryptography libraries from Secude and OpenSSL
- Long integer arithmetic from Miracl and GMP, Lattice base reduction via NTL (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 a lot of program functions (tutorial)

Script (.pdf file) with background information

- Encryption methods • Prime factorisation • Digital signature
- Elliptic curves • Public-key certification • Basic number theory • Crypto 2020

Two short stories related to cryptography by Dr. C. Elsner

- „The Dialogue of the Sisters“ (a RSA variant as key element)
- „The Chinese Labyrinth“ (Numbers 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
- Solitaire

Several options to easily understand the cryptography methods

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

Cryptanalysis

Attack on classical methods

- Cipher text only
  - Caesar
  - Vigenère
  - Addition
  - XOR
  - Substitution
  - Playfair
- Known-plaintext
  - Hill
- Manually (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, …)
- AES (=Rijndael)
- DESL, DESXL

Asymmetric encryption

- RSA with X.509 certificates
- RSA demonstration
  - Understanding of examples
  - Alphabet and block length selectable

Hybrid encryption (RSA + AES)

- Interactive data flow diagram

Cryptanalysis

Brute-force attack on symmetric algorithm

- For all algorithms
- Assumptions:
  - Entropy of plaintext is small or key is partly known or plaintext alphabet is known

Attack on RSA encryption

- Factorisation of RSA module
- 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
- x^2 mod n
- Linear congruence generator (LCG)
- Inverse congruence generator (ICG)

Cryptanalysis

Attack on RSA signature

- Factorisation 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 bit in around 5 min)

Analysis of random data

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

Features (4)

Animation / Demos

- Caesar, Vigenère, Nihilist, DES (all with ANIMAL)
- Enigma (Flash)
- Rijdael/AES (Flash)
- 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 (authentication)
- Side-channel attack
- Graphical 3D presentation of (random) data streams
- Sensitivity of hash functions regarding plaintext modifications
- Number theory and RSA crypto system

Features (5)

Additional functions

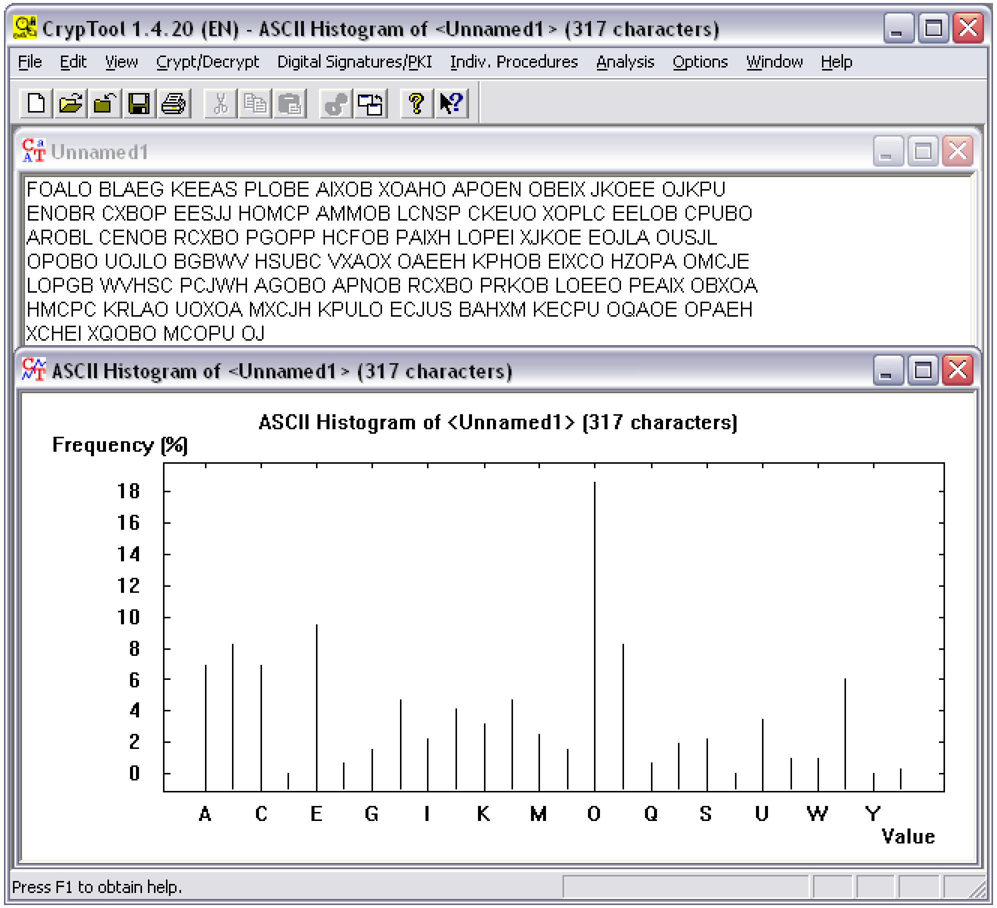
- Homophone and permutation encryption (Double Column Transposition)
- PKCS #12 import and export for PSEs (Personal Security Environment)
- Generate hashes of large files, without loading them
- Flexible brute-force attacks on any modern symmetric algorithm
- ECC demonstration (as Java application)
- Password Quality Meter (PQM) and password entropy
- And a lot more …

Language Structure Analysis

Language analysis options available in CrypTool

Number of characters, n-gram, entropy

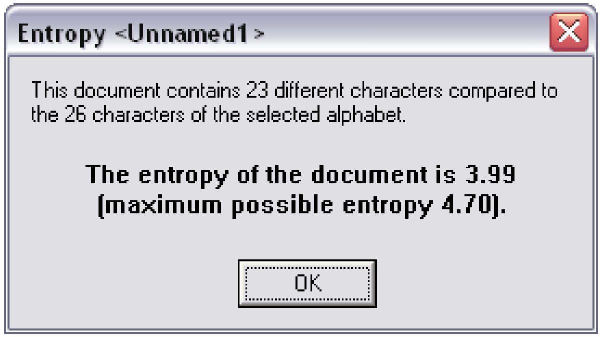
- See menu „Analysis“ \ „Tools for Analysis“ \ …



CrypTool 1.4.20 (EN) - ASCII Histogram of <Unnamed1> (317 characters)
File Edit View Crypt/Decrypt Digital Signatures/PKI Indiv. Procedures Analysis Options Window Help
Unnamed1

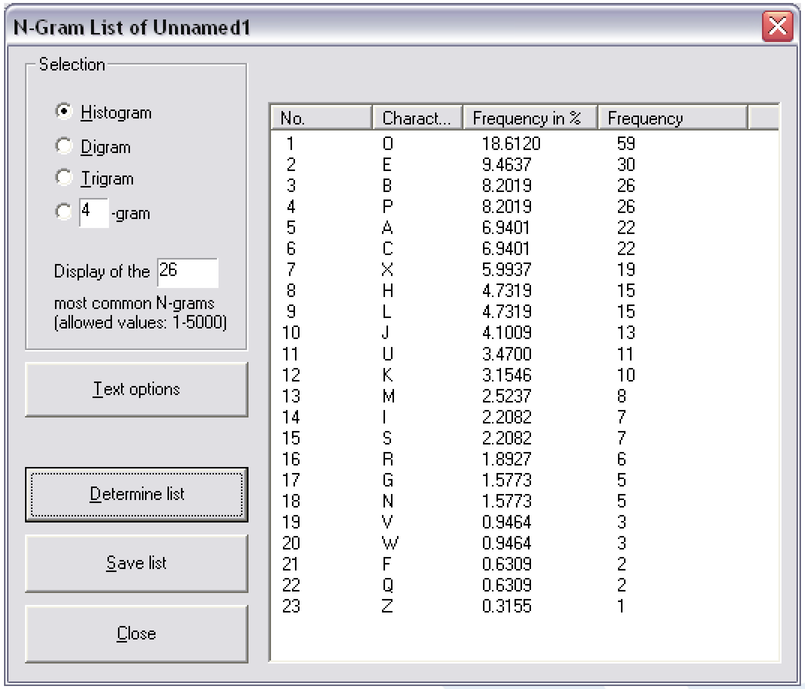
```
FOALO BLAEG KEEAS PLOBE AIXOB XOAHO APOEN OBEIX JKOEE OJKPU
ENOBR CXBOP EESJJ HOMCP AMMOB LCNSP CKEUO XOPLC EELOB CPUBO
AROBL CENOB RCXBO PGOPP HCFOB PAIXH LOPEI XJKOE EOJLA OUSJL
OPOBO UOJLO BGBWV HSUBC VXAOX OAEEH KPHOB EIXCO HZOPA OMCJE
LOPGB WVHSC PCJWH AGOBO APNOB RCXBO PRKOB LOEEO PEAIX OBXOA
HMCPC KRLAO UOXOA MXCJH KPULO ECJUS BAHXM KECPU OQAOE OPAEH
XCHEI XQOBO MCOPU OJ
```

ASCII Histogram of <Unnamed1> (317 characters)
ASCII Histogram of <Unnamed1> (317 characters)
Frequency (%)
18
16
14
12
10
8
6
4
2
0
A
C
E
G
I
K
M
O
Q
S
U
W
Y
Value
Press F1 to obtain help.



This document contains 23 different characters compared to the 26 characters of the selected alphabet.

The entropy of the document is 3.99 (maximum possible entropy 4.70).



● Histogram
○ Digram
○ Trigram
○ 4 -gram

Display of the 26 most common N-grams (allowed values: 1-5000)

No.	Charact...	Frequency in %	Frequency
1	O	18.6120	59
2	E	9.4637	30
3	B	8.2019	26
4	P	8.2019	26
5	A	6.9401	22
6	C	6.9401	22
7	X	5.9937	19
8	H	4.7319	15
9	L	4.7319	15
10	J	4.1009	13
11	U	3.4700	11
12	K	3.1546	10
13	M	2.5237	8
14	I	2.2082	7
15	S	2.2082	7
16	R	1.8927	6
17	G	1.5773	5
18	N	1.5773	5
19	V	0.9464	3
20	W	0.9464	3
21	F	0.6309	2
22	Q	0.6309	2
23	Z	0.3155	1

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 starting example 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 example 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 – not correct ✗
   - Key length automatically set to 4 (can also be adjusted manually)
   - Derived key TEST ✓

Demonstration of Interactivity (2)

Automated factorisation

Demonstration in CrypTool

Factorisation of a compound number with factorisation algorithms

- Menu: „Indiv. Procedures“ \ „RSA Cryptosystem“ \ „Factorisation of a Number“
- Some methods are executed in parallel (multi-threaded)
- Methods have specific advantages and disadvantages (e.g. some methods can only determine small factors)

Factorisation example 1 :

316775895367314538931177095642205088158145887517

=

3 * 1129 * 6353 * 1159777 * 22383173213963 * 567102977853788110597

48-digit decimal number

Factorisation example 2:

2^250 - 1

=

3 * 11 * 31 * 251 * 601 * 1801 * 4051 * 229668251 * 269089806001 * 4710883168879506001 * 5519485418336288303251

75-digit decimal number

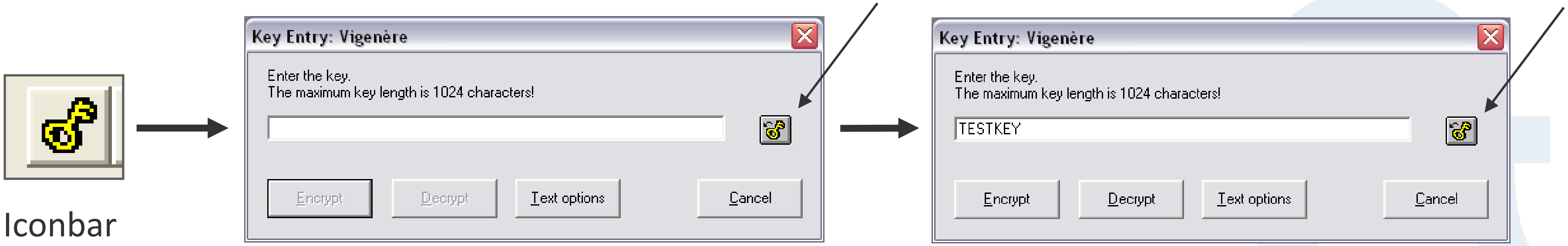
Concepts for a User-Friendly Interface

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 assistances and the contents of the super ordinate menus are cross linked in the online help.

2. Paste of keys in key-input dialog

- CTRL-V can be used to paste contents from the clipboard.
- Used keys can be taken out of cipher text windows via an icon in the icon bar. A corresponding icon in the key-input dialog can be used to paste the key into the key field. A CrypTool-internal memory which is available for every method is used (helpful for large „specific“ keys – e.g. homophone encryption).



Key Entry: Vigenère

Enter the key.
The maximum key length is 1024 characters!

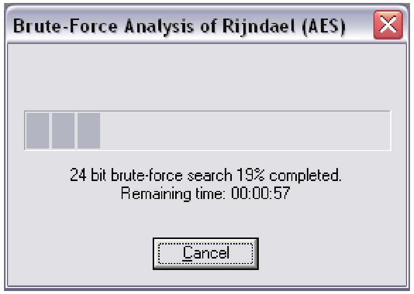
Key Entry: Vigenère

Enter the key.
The maximum key length is 1024 characters!

TESTKEY

Challenges for Developers (Examples)

1. **Many functions running in parallel**
   - Factorisation runs with multi-threaded algorithms
2. **High performance**
   - Locate hash collisions (birthday paradox) or perform brute-force analysis
3. **Consider memory limits**
   - Floyd algorithm (mappings to locate hash collisions) or factorisation with quadratic sieve
4. **Time measurement and estimates**
   - Display of elapsed time 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. **Partly automate the consistency of functions, GUI and online help** (including different languages)



24 bit brute-force search 19% completed.
Remaining time: 00:00:57

Content

I. CrypTool and Cryptology – Overview

II. CrypTool Features

III. Examples

IV. Project / Outlook / Contact

CrypTool Examples

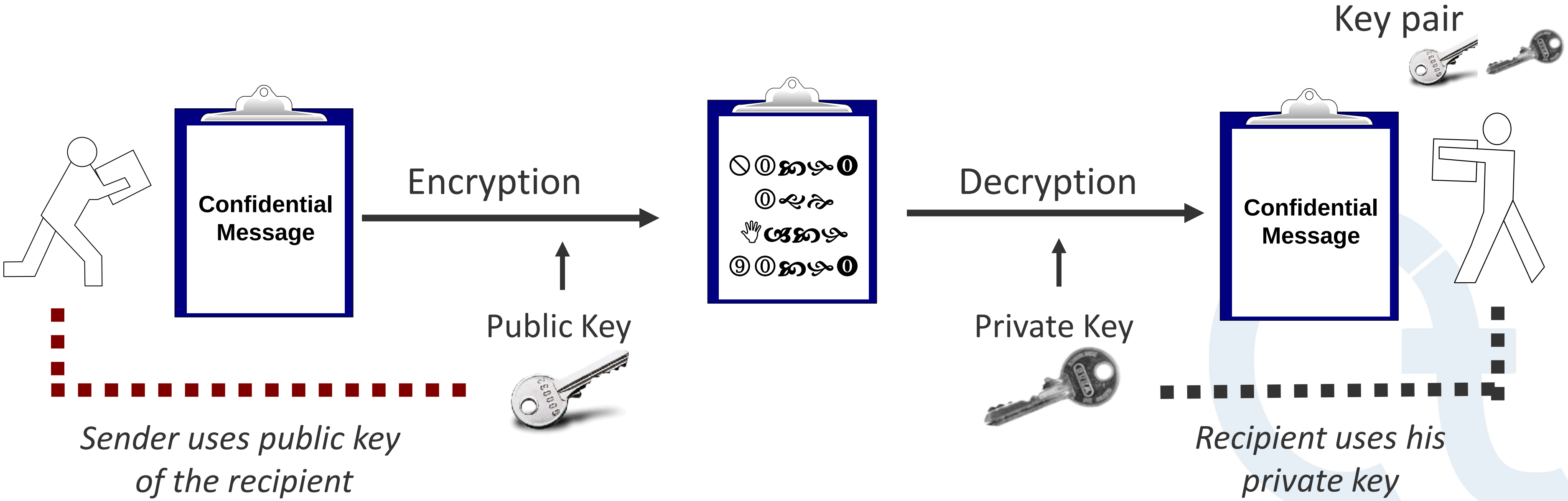
Overview of examples

1. Encryption with RSA / Prime number tests / Hybrid encryption and digital certificates / SSL
2. Digital signature visualised
3. Attack on RSA encryption (modul N too short)
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 visualisation
12. Secret Sharing using the Chinese Remainder Theorem (CRT) and Shamir
13. Implementation of CRT in astronomy (solving linear modular equation systems)
14. Visualisation of symmetric encryption methods using ANIMAL
15. Visualisation of AES
16. Visualisation of Enigma encryption
17. Generation of a message authentication code (MAC)
18. Hash demonstration
19. Learning tool for number theory and asymmetric encryption
20. Point addition on elliptic curves
21. Password quality meter (PQM) and password entropy
22. Brute-force analysis
23. CrypTool online help

Examples (1)

Encryption with RSA (in reality mostly hybrid encryption)

- **Basis for e.g. SSL protocol (access to protected web sites)**
- **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 private key
- **Implemented usually in a combination with symmetric methods (transfer of the symmetric key through RSA asymmetric encryption/decryption)**



Examples (1)

Encryption using RSA – Mathematical background / algorithm

- Public key: (n, e)
- Private key: (d)

where:

p, q 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 module, which length in bits is referred to as RSA key length.
- gcd = greatest common divisor.
- $\varphi(n)$ is the Euler phi function.

Procedure :

- Transformation of message in binary representation
- Encrypt message $m = m_1,...,m_k$ block wise, with for all m_j:
  $0 \leq m_j < n$; maximum block size r, so that: $2^r \leq n$ $(2^r-1 < n)$

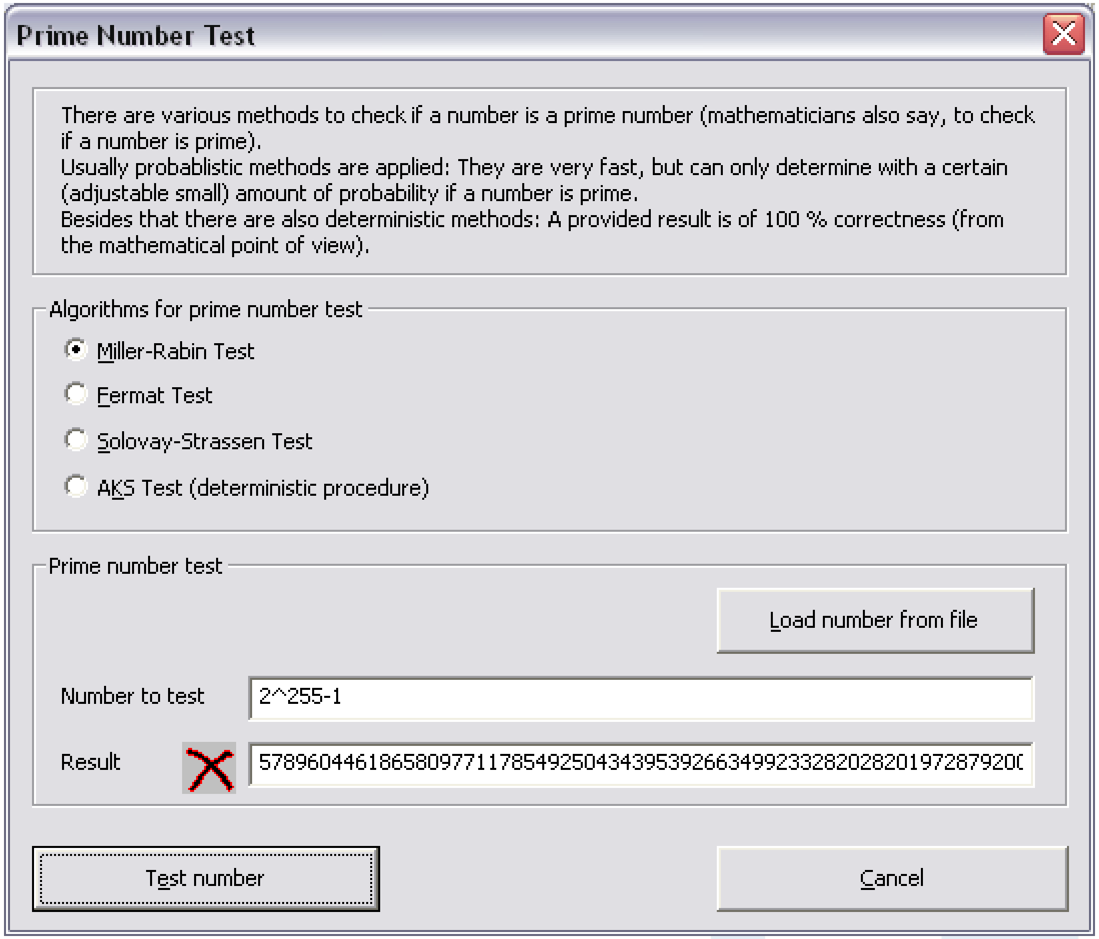
Examples (1)

Prime number tests – For RSA huge primes are needed

- Fast probabilistic tests
- Deterministic tests

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

For the AKS test the GMP library (**G**NU **M**ultiple **P**recision Arithmetic Library) was integrated into CrypTool.



There are various methods to check if a number is a prime number (mathematicians also say, to check if a number is prime).
Usually probablistic methods are applied: They are very fast, but can only determine with a certain (adjustable small) amount of probability if a number is prime.
Besides that there are also deterministic methods: A provided result is of 100 % correctness (from the mathematical point of view).

Algorithms for prime number test

● Miller-Rabin Test
○ Fermat Test
○ Solovay-Strassen Test
○ AKS Test (deterministic procedure)

Prime number test

Number to test: 2^255-1

Result: ✗ 578960446186580977117854925043439539266349923328202820197287920(

Menu: „Indiv. Procedure“ \ „RSA Cryptosystem“ \ „Prime Number Test “

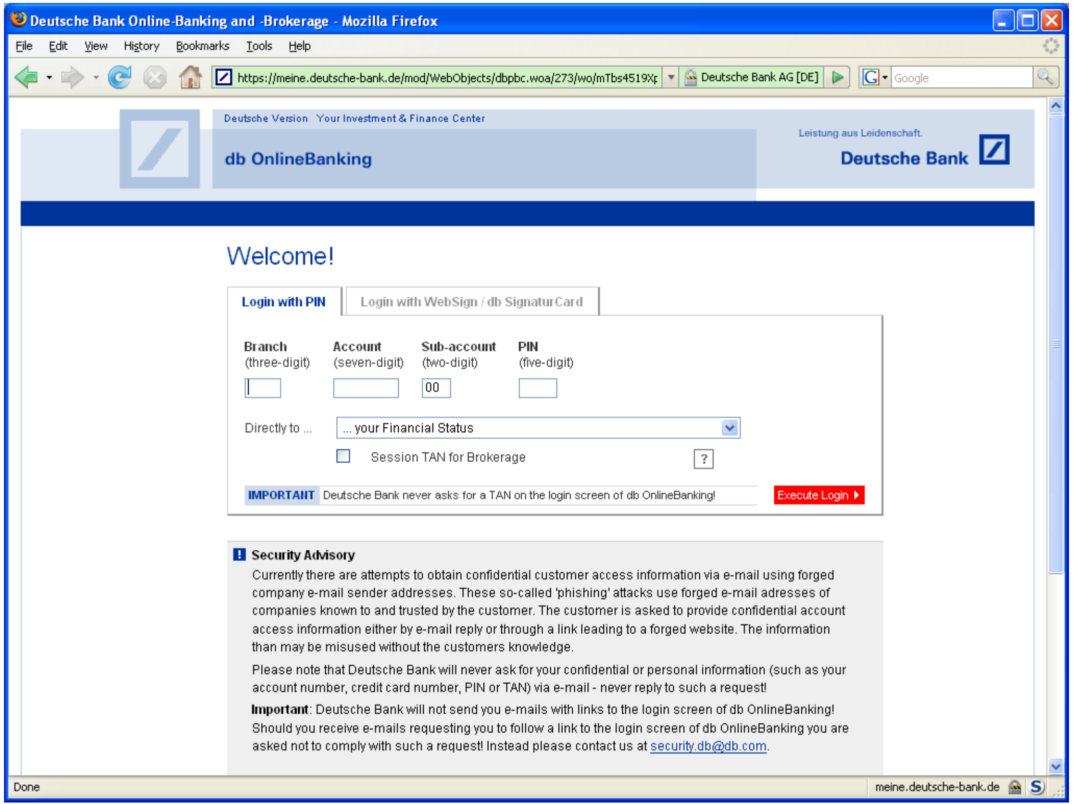
Examples (1)

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. Telesec, VeriSign, Deutsche Bank PKI), that is being trusted by all users, ensures the authenticity of the certificate and the contained public key (similar to a passport issued by the state).
- **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


Deutsche Bank Online-Banking and -Brokerage - Mozilla Firefox
File Edit View History Bookmarks Tools Help
https://meine.deutsche-bank.de/mod/WebObjects/dbpbc.woa/273/wo/mTbs4519Xp
Deutsche Bank AG [DE]
Google
Deutsche Version Your Investment & Finance Center
db OnlineBanking
Leistung aus Leidenschaft.
Deutsche Bank

Welcome!

Login with PIN | Login with WebSign / db SignaturCard

Branch (three-digit)	Account (seven-digit)	Sub-account (two-digit)	PIN (five-digit)
		00	

Directly to ... ... your Financial Status

□ Session TAN for Brokerage ?

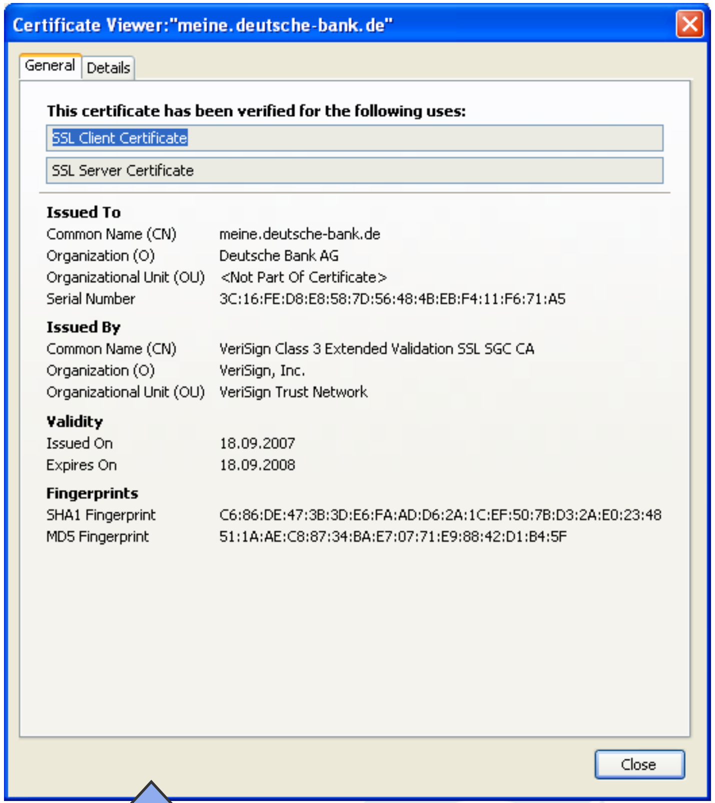
IMPORTANT Deutsche Bank never asks for a TAN on the login screen of db OnlineBanking! Execute Login

Security Advisory

Currently there are attempts to obtain confidential customer access information via e-mail using forged company e-mail sender addresses. These so-called 'phishing' attacks use forged e-mail adresses of companies known to and trusted by the customer. The customer is asked to provide confidential account access information either by e-mail reply or through a link leading to a forged website. The information than may be misused without the customers knowledge.

Please note that Deutsche Bank will never ask for your confidential or personal information (such as your account number, credit card number, PIN or TAN) via e-mail - never reply to such a request!

Important: Deutsche Bank will not send you e-mails with links to the login screen of db OnlineBanking! Should you receive e-mails requesting you to follow a link to the login screen of db OnlineBanking you are asked not to comply with such a request! Instead please contact us at security.db@db.com.



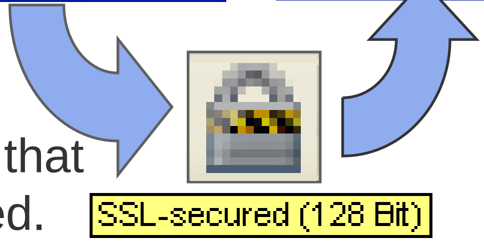
This certificate has been verified for the following uses:

SSL Client Certificate

SSL Server Certificate

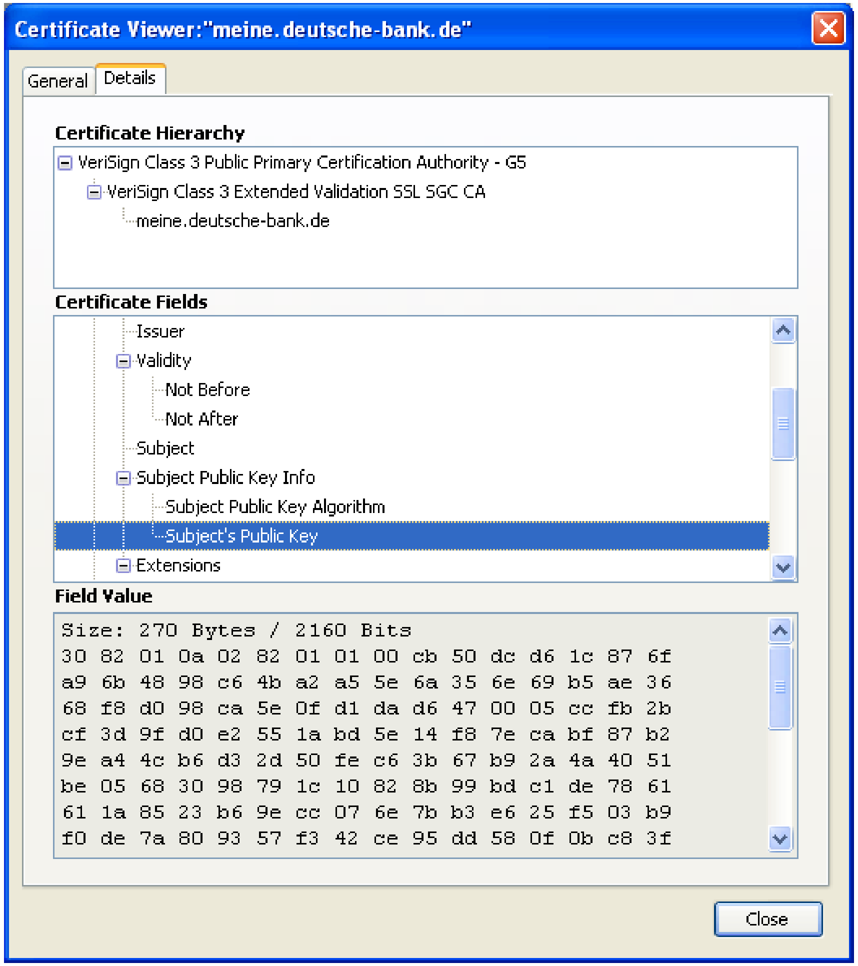
Issued To	
Common Name (CN)	meine.deutsche-bank.de
Organization (O)	Deutsche Bank AG
Organizational Unit (OU)	<Not Part Of Certificate>
Serial Number	3C:16:FE:D8:E8:58:7D:56:48:4B:EB:F4:11:F6:71:A5
Issued By	
Common Name (CN)	VeriSign Class 3 Extended Validation SSL SGC CA
Organization (O)	VeriSign, Inc.
Organizational Unit (OU)	VeriSign Trust Network
Validity	
Issued On	18.09.2007
Expires On	18.09.2008
Fingerprints	
SHA1 Fingerprint	C6:86:DE:47:3B:3D:E6:FA:AD:D6:2A:1C:EF:50:7B:D3:2A:E0:23:48
MD5 Fingerprint	51:1A:AE:C8:87:34:BA:E7:07:71:E9:88:42:D1:B4:5F

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



Examples (1)

Attributes or fields of a certificate



Certificate Hierarchy

- VeriSign Class 3 Public Primary Certification Authority - G5
  - VeriSign Class 3 Extended Validation SSL SGC CA
    - meine.deutsche-bank.de

Certificate Fields

- Issuer
- Validity
  - Not Before
  - Not After
- Subject
- Subject Public Key Info
  - Subject Public Key Algorithm
  - Subject's Public Key
- Extensions

Field Value

```
Size: 270 Bytes / 2160 Bits
30 82 01 0a 02 82 01 01 00 cb 50 dc d6 1c 87 6f
a9 6b 48 98 c6 4b a2 a5 5e 6a 35 6e 69 b5 ae 36
68 f8 d0 98 ca 5e 0f d1 da d6 47 00 05 cc fb 2b
cf 3d 9f d0 e2 55 1a bd 5e 14 f8 7e ca bf 87 b2
9e a4 4c b6 d3 2d 50 fe c6 3b 67 b9 2a 4a 40 51
be 05 68 30 98 79 1c 10 82 8b 99 bd c1 de 78 61
61 1a 85 23 b6 9e cc 07 6e 7b b3 e6 25 f5 03 b9
f0 de 7a 80 93 57 f3 42 ce 95 dd 58 0f 0b c8 3f
```

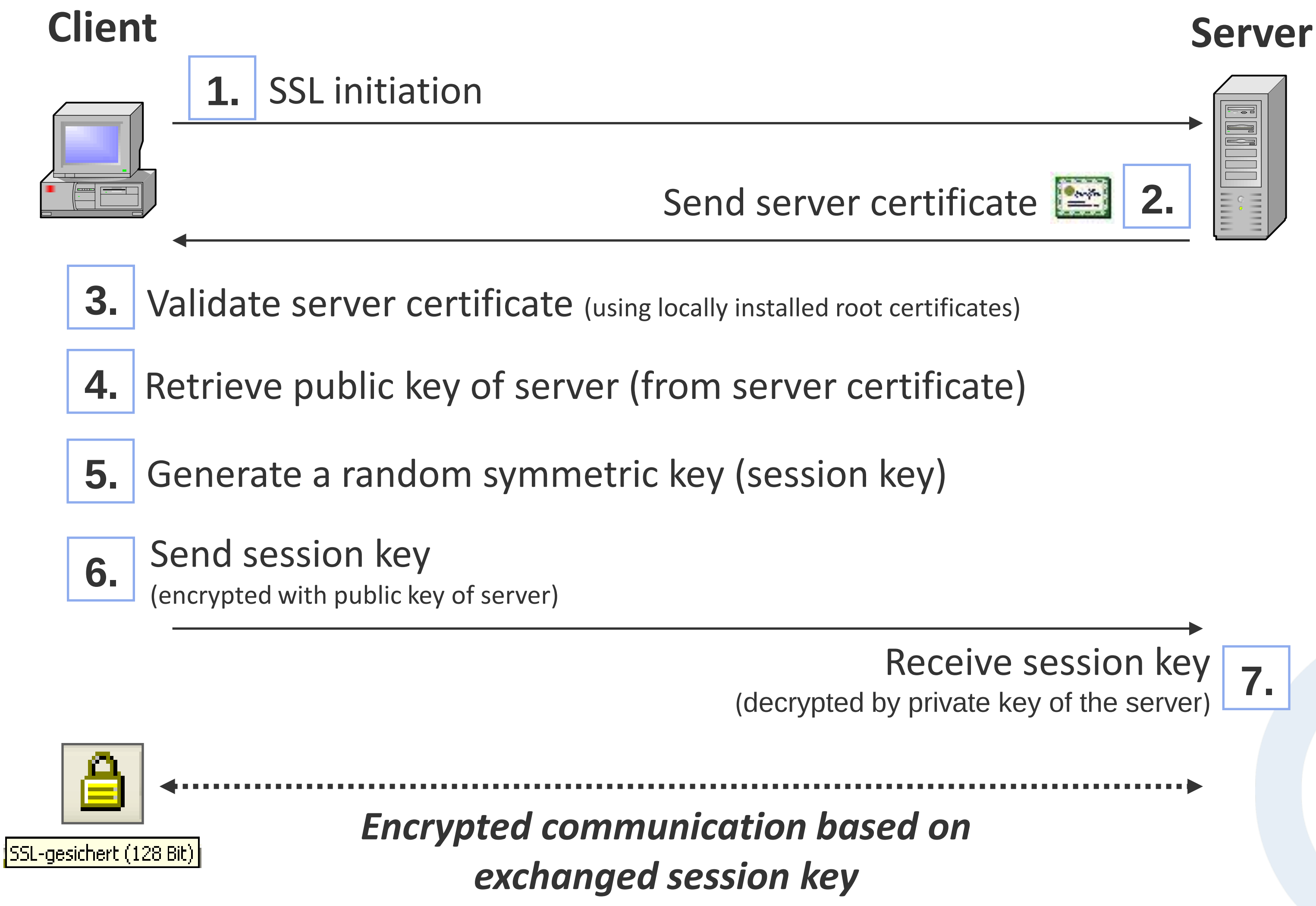
General attributes / fields

- 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)

SSL-gesichert (128 Bit)

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 the 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.

Comments to the SSL connection establishment

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

Examples (2)

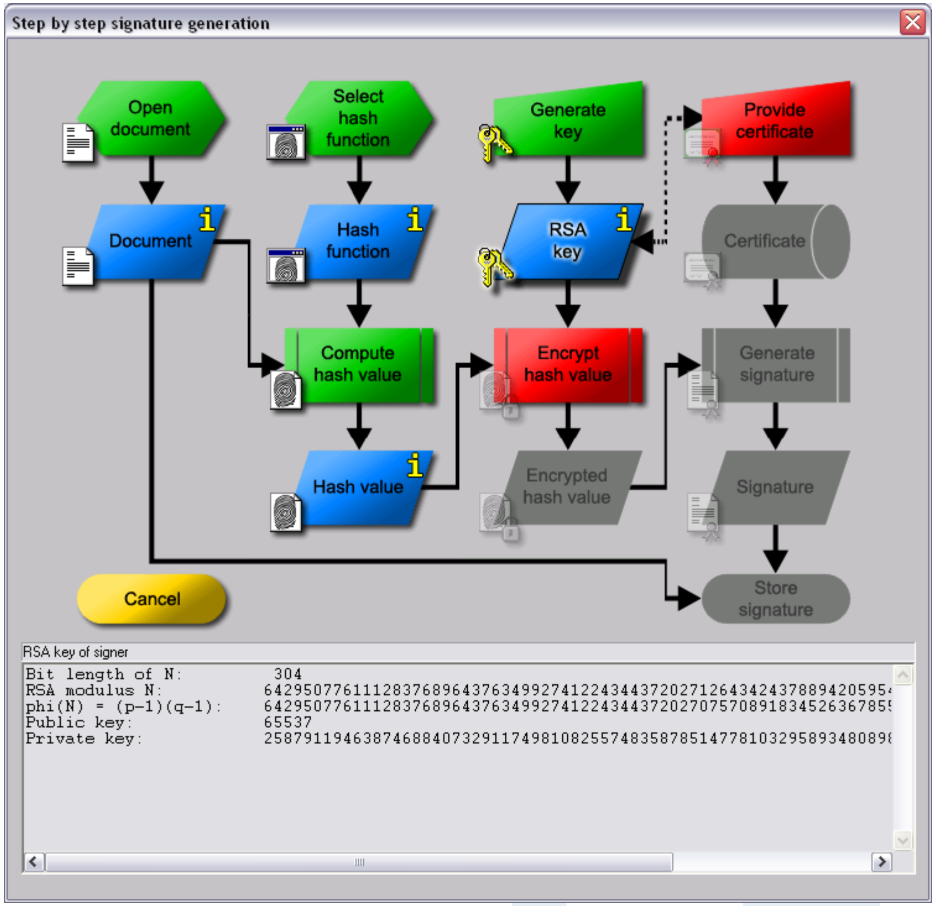
Digital signature visualised

Digital signature

- Increasingly important
  - equivalence with manual signature (digital signature law)
  - increasingly used by industry,
  - government and consumers
- Few people know how it works exactly

Visualisation in CrypTool

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



```
Bit length of N:          304
RSA modulus N:           642950776111283768964376349927412243443720271264342437889420595
phi(N) = (p-1)(q-1):     642950776111283768964376349927412243443720270757089183452636785
Public key:              65537
Private key:             258791194638746884073291174981082557483587851477810329589348089
```

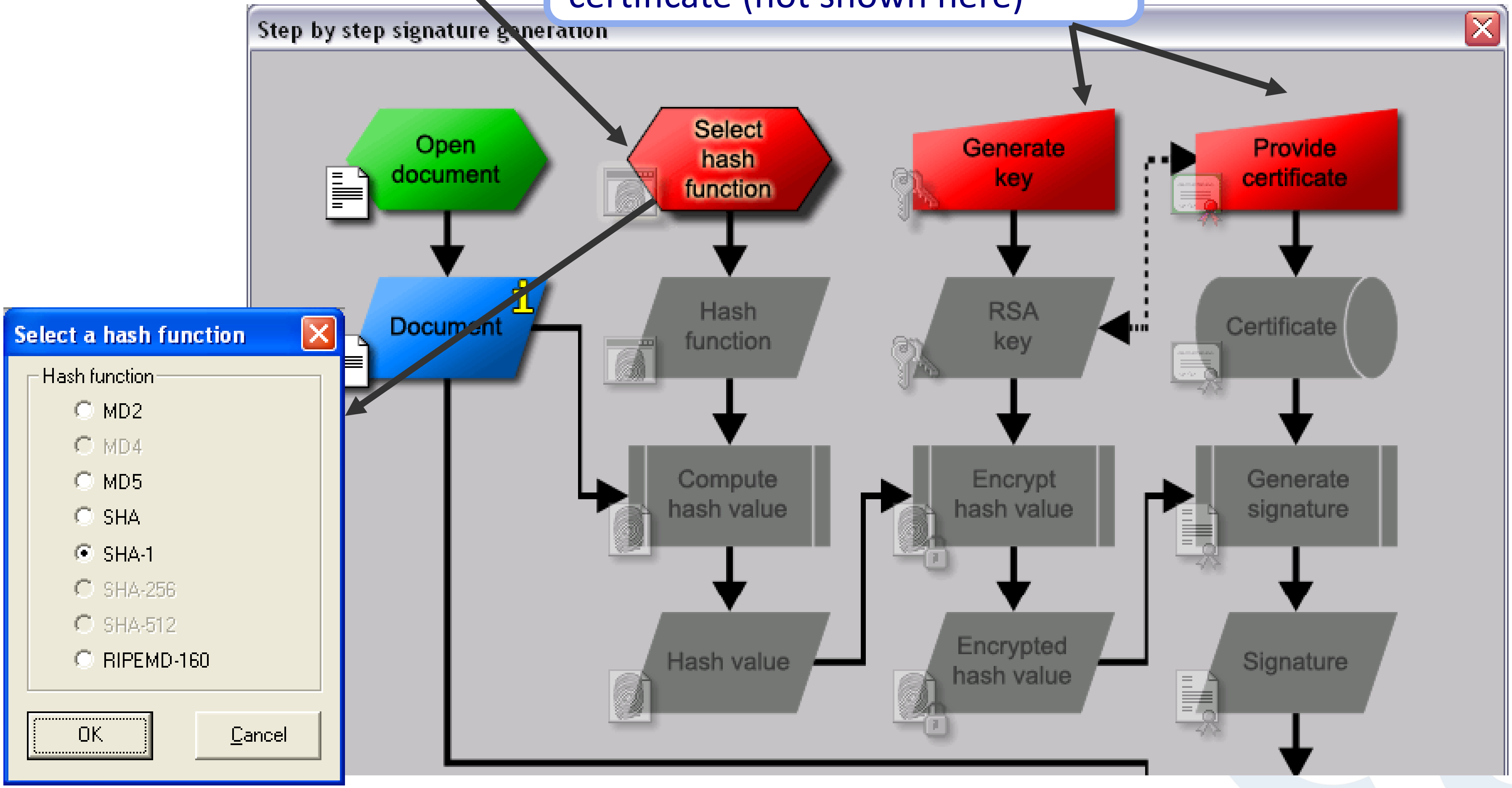
Menu: „Digital Signatures/PKI“ \
„Signature Demonstration (Signature Generation)“

Examples (2)

Digital signature visualised: a) Preparation

1. Select hash function

2. Provide key and certificate (not shown here)



Select a hash function

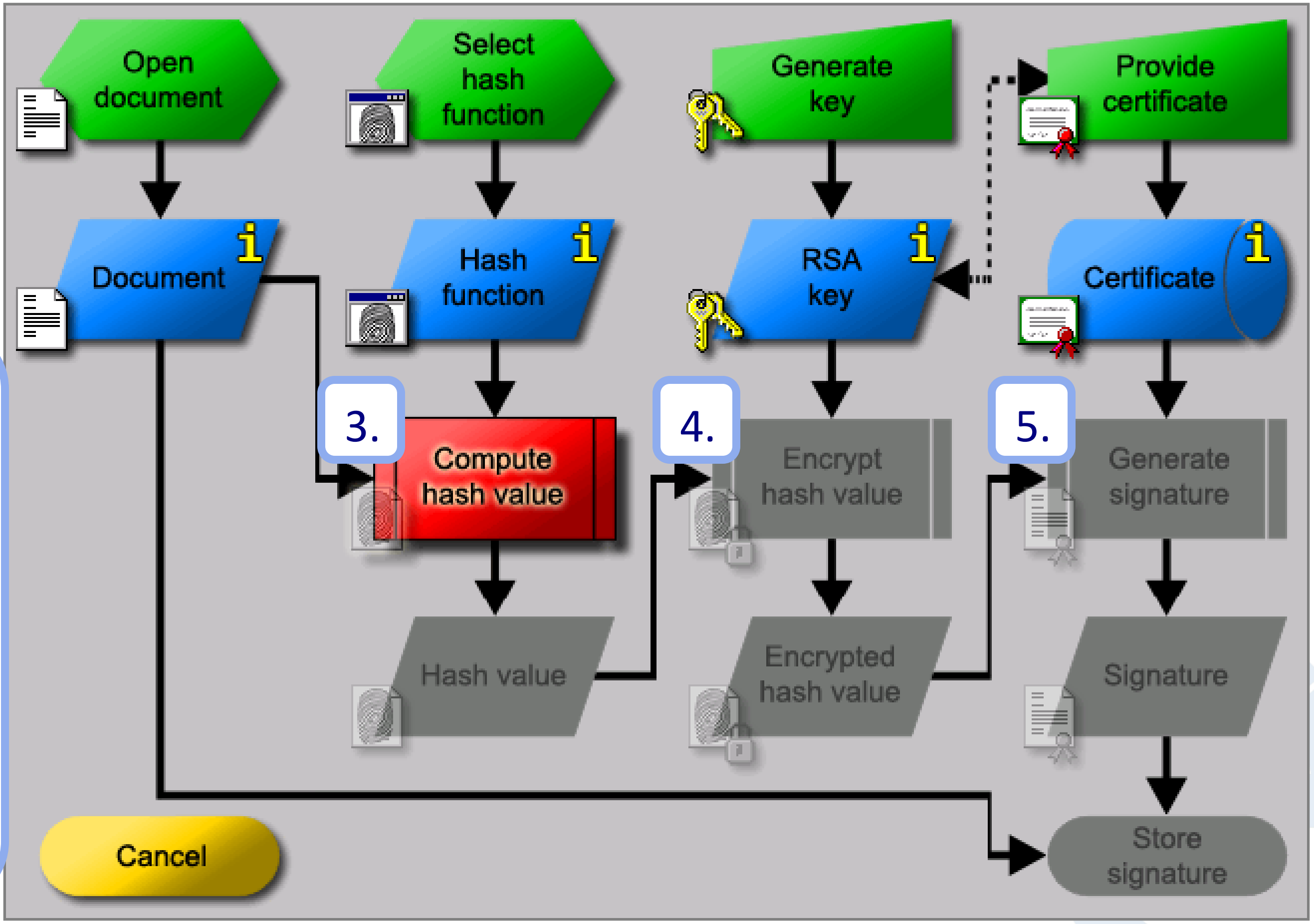
Hash function

- ○ MD2
- ○ MD4
- ○ MD5
- ○ SHA
- ● SHA-1
- ○ SHA-256
- ○ SHA-512
- ○ RIPEMD-160

Examples (2)

Digital signature visualised: b) Cryptography

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

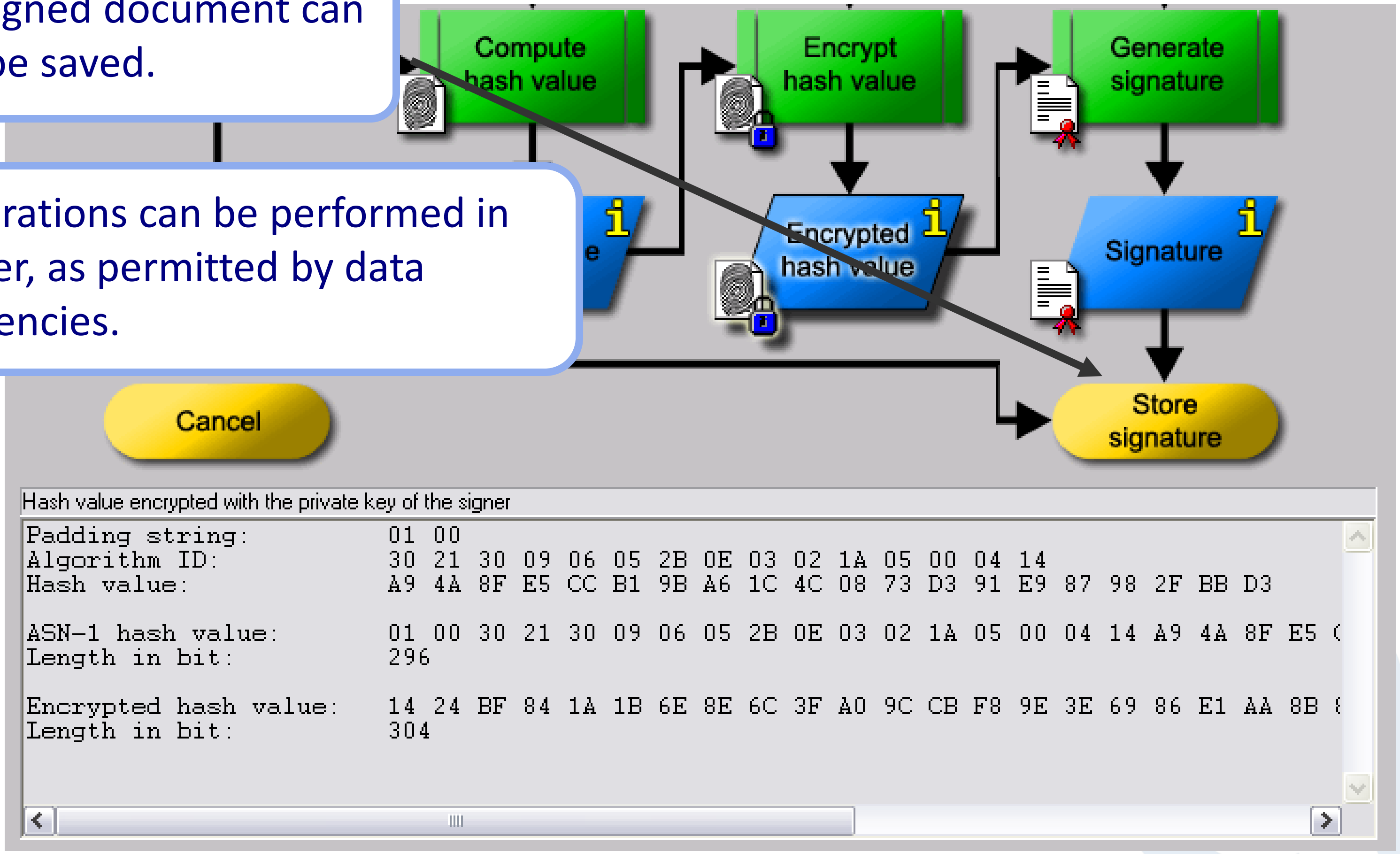


Examples (2)

Digital signature visualised: c) Result

6. The signed document can now be saved.

The operations can be performed in any order, as permitted by data dependencies.



Hash value encrypted with the private key of the signer

```
Padding string:         01 00
Algorithm ID:           30 21 30 09 06 05 2B 0E 03 02 1A 05 00 04 14
Hash value:             A9 4A 8F E5 CC B1 9B A6 1C 4C 08 73 D3 91 E9 87 98 2F BB D3

ASN-1 hash value:       01 00 30 21 30 09 06 05 2B 0E 03 02 1A 05 00 04 14 A9 4A 8F E5 (
Length in bit:          296

Encrypted hash value:   14 24 BF 84 1A 1B 6E 8E 6C 3F A0 9C CB F8 9E 3E 69 86 E1 AA 8B 8
Length in bit:          304
```

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 bit, 29 decimal digits)
  - public exponent **e = 17579**
- Ciphertext (block length = 8):

  C_1 = 45411667895024938209259253423,
  C_2 = 16597091621432020076311552201,
  C_3 = 46468979279750354732637631044,
  C_4 = 32870167545903741339819671379
- The text shall be deciphered!

The ciphertext is not necessary for the actual cryptanalysis (locating the private key) !

Solution using CrypTool **(more detailed in online help examples section)**

- Enter public parameters into “RSA cryptosystem” (menu: „Indiv. Procedures”)
- Button “Factorise 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) is determined
- Decrypt the cipher text with d: $M_i = C_i^d$ mod N

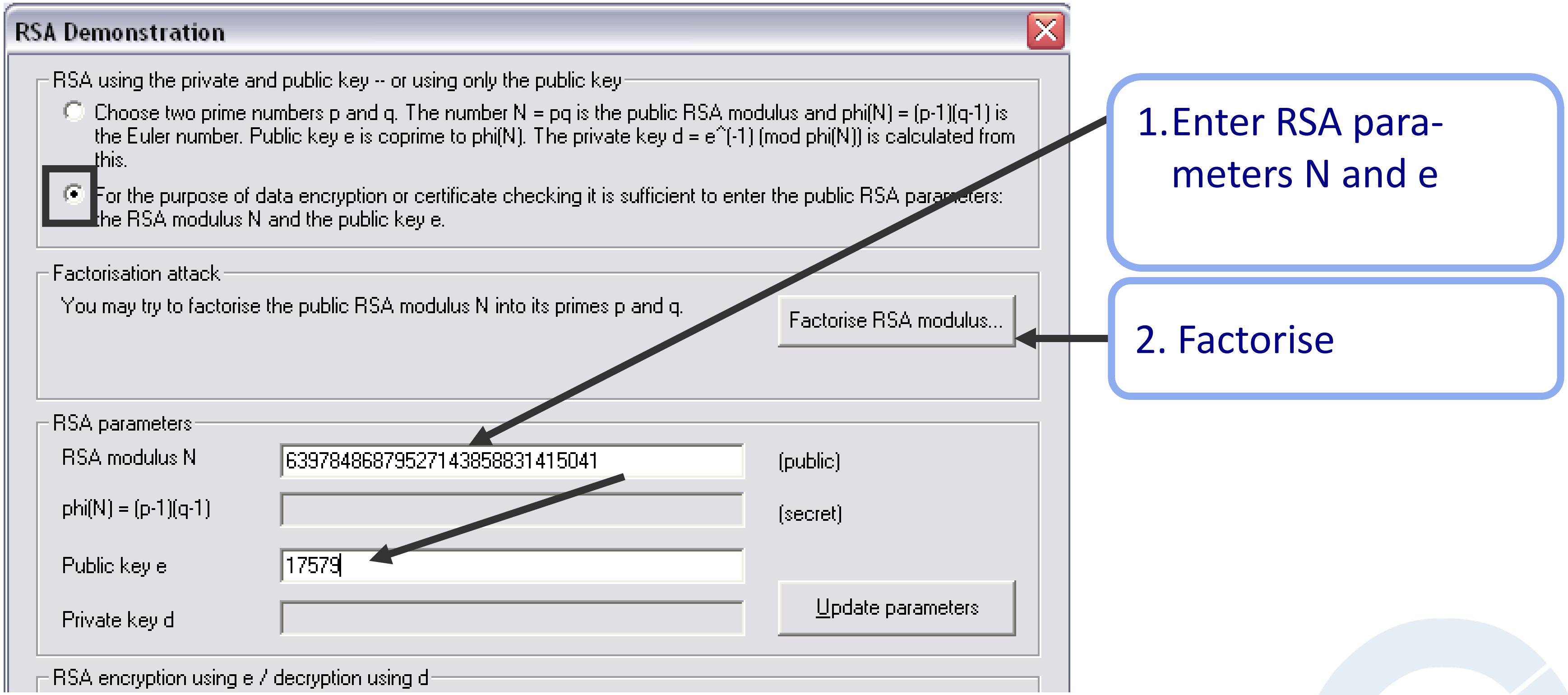
The attack with CrypTool works for RSA moduli up to 250 bit.

Then you could digitally sign for someone else !

Examples (3)

Short RSA modulus: enter public RSA parameters

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



RSA Demonstration

RSA using the private and public key -- or using only the public key

○ Choose two prime numbers p and q. The number N = pq is the public RSA modulus and phi(N) = (p-1)(q-1) is the Euler number. Public key e is coprime to phi(N). The private key d = e^(-1) (mod phi(N)) is calculated from this.

● For the purpose of data encryption or certificate checking it is sufficient to enter the public RSA parameters: the RSA modulus N and the public key e.

Factorisation attack

You may try to factorise the public RSA modulus N into its primes p and q.

RSA parameters

RSA modulus N	63978486879527143858831415041	(public)
phi(N) = (p-1)(q-1)		(secret)
Public key e	17579	
Private key d		

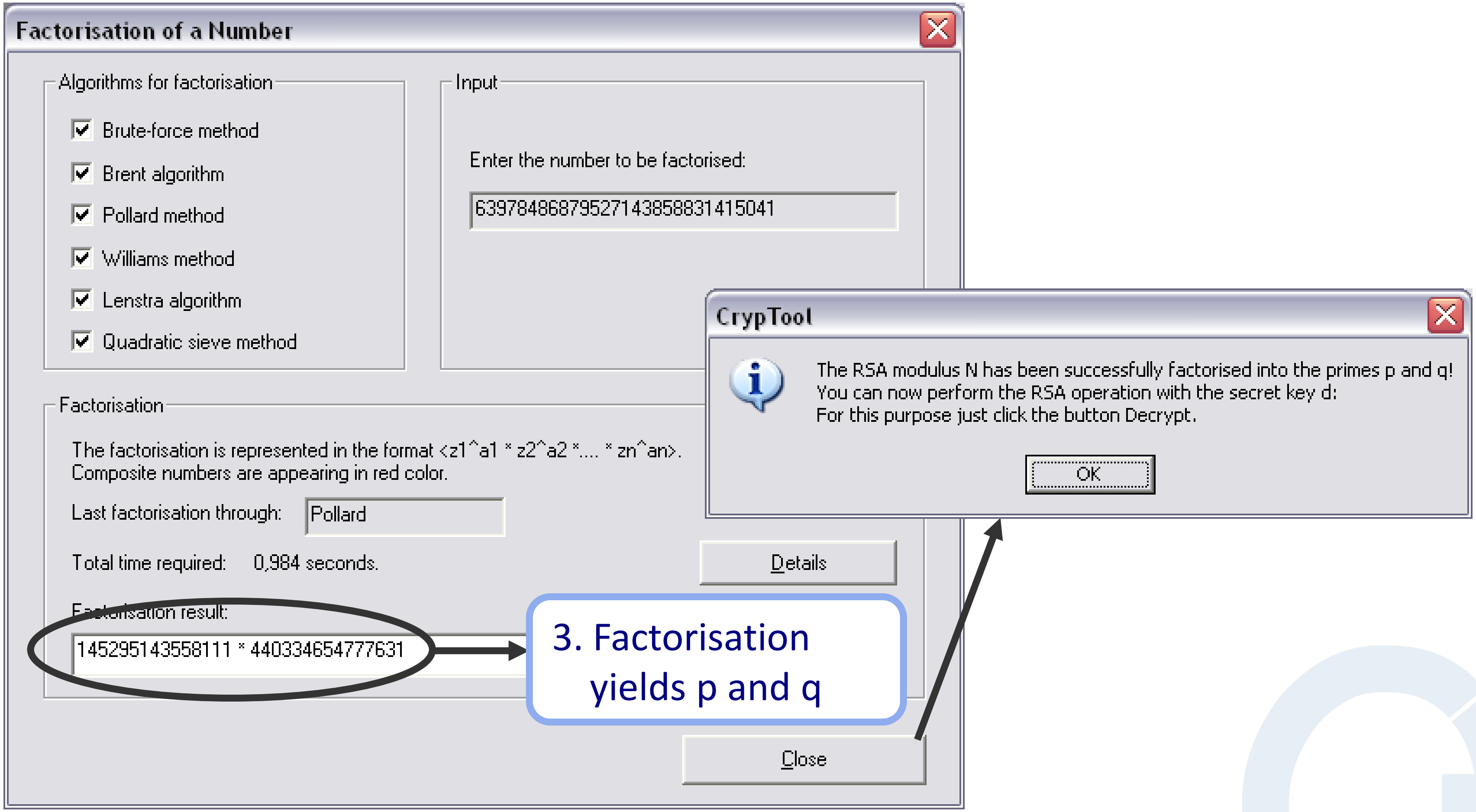
RSA encryption using e / decryption using d

1. Enter RSA parameters N and e

2. Factorise

Examples (3)

Short RSA modulus: factorise RSA modulus



Factorisation of a Number

Algorithms for factorisation

- ☑ Brute-force method
- ☑ Brent algorithm
- ☑ Pollard method
- ☑ Williams method
- ☑ Lenstra algorithm
- ☑ Quadratic sieve method

Input

Enter the number to be factorised: 63978486879527143858831415041

Factorisation

The factorisation is represented in the format <z1^a1 * z2^a2 *.... * zn^an>.
Composite numbers are appearing in red color.

Last factorisation through: Pollard

Total time required: 0,984 seconds.

Factorisation result: 145295143558111 * 440334654777631

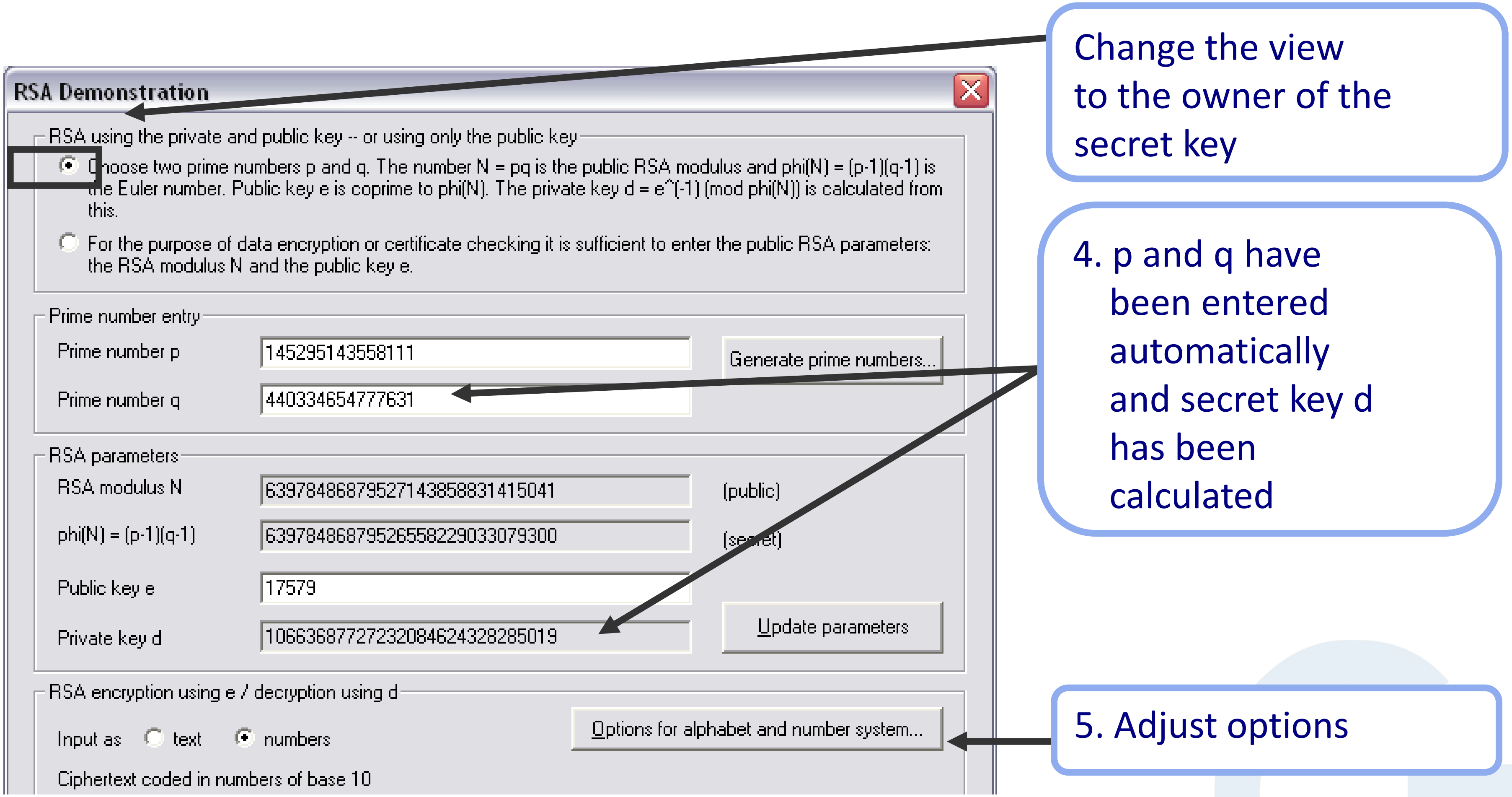
CrypTool

The RSA modulus N has been successfully factorised into the primes p and q!
You can now perform the RSA operation with the secret key d:
For this purpose just click the button Decrypt.

3. Factorisation yields p and q

Examples (3)

Short RSA modulus: determine private key d



RSA Demonstration

RSA using the private and public key -- or using only the public key

● Choose two prime numbers p and q. The number N = pq is the public RSA modulus and phi(N) = (p-1)(q-1) is the Euler number. Public key e is coprime to phi(N). The private key d = e^(-1) (mod phi(N)) is calculated from this.

○ For the purpose of data encryption or certificate checking it is sufficient to enter the public RSA parameters: the RSA modulus N and the public key e.

Prime number entry

Prime number p	145295143558111
Prime number q	440334654777631

RSA parameters

RSA modulus N	63978486879527143858831415041	(public)
phi(N) = (p-1)(q-1)	63978486879526558229033079300	(secret)
Public key e	17579	
Private key d	10663687727232084624328285019	

RSA encryption using e / decryption using d

Input as ○ text ● numbers

Ciphertext coded in numbers of base 10

Generate prime numbers...
Update parameters
Options for alphabet and number system...

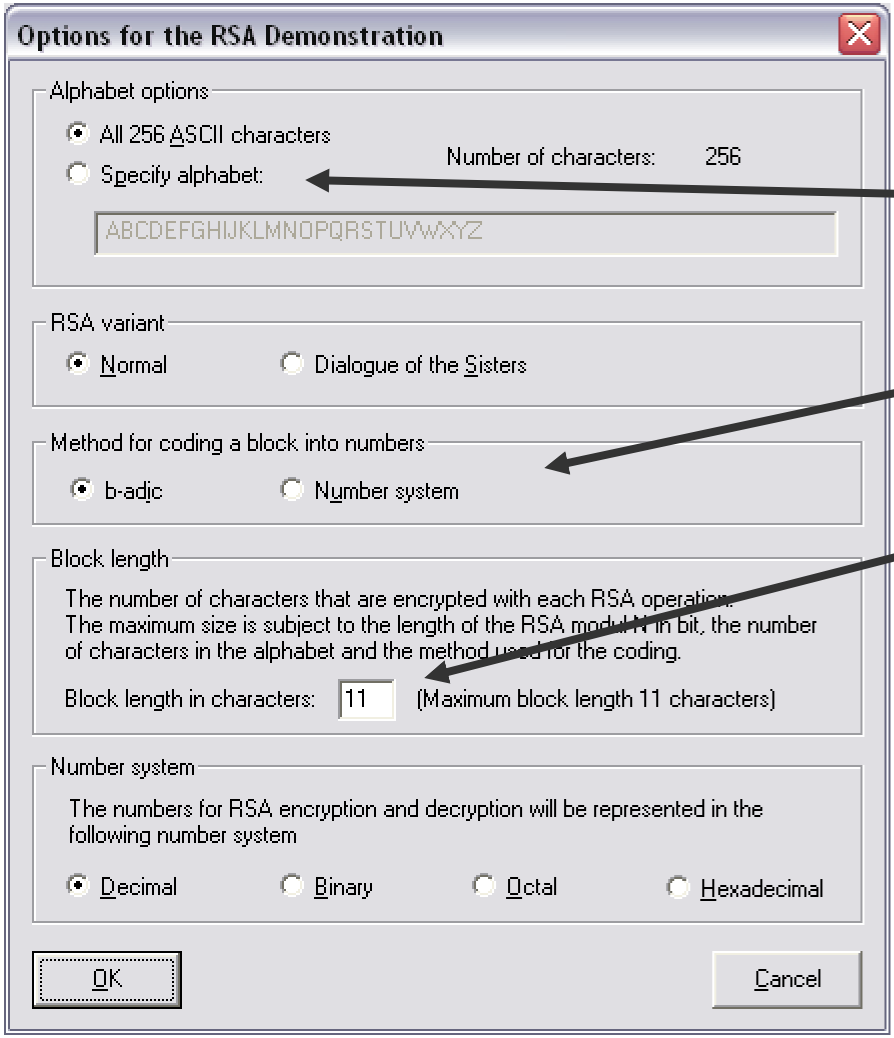
Change the view to the owner of the secret key

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

5. Adjust options

Examples (3)

Short RSA modulus: adjust options



Options for the RSA Demonstration

Alphabet options

● All 256 ASCII characters

○ Specify alphabet:

Number of characters: 256

ABCDEFGHIJKLMNOPQRSTUVWXYZ

RSA variant

● Normal ○ Dialogue of the Sisters

Method for coding a block into numbers

● b-adic ○ Number system

Block length

The number of characters that are encrypted with each RSA operation. The maximum size is subject to the length of the RSA modul N in bit, the number of characters in the alphabet and the method used for the coding.

Block length in characters: 11 (Maximum block length 11 characters)

Number system

The numbers for RSA encryption and decryption will be represented in the following number system

● Decimal ○ Binary ○ Octal ○ Hexadecimal

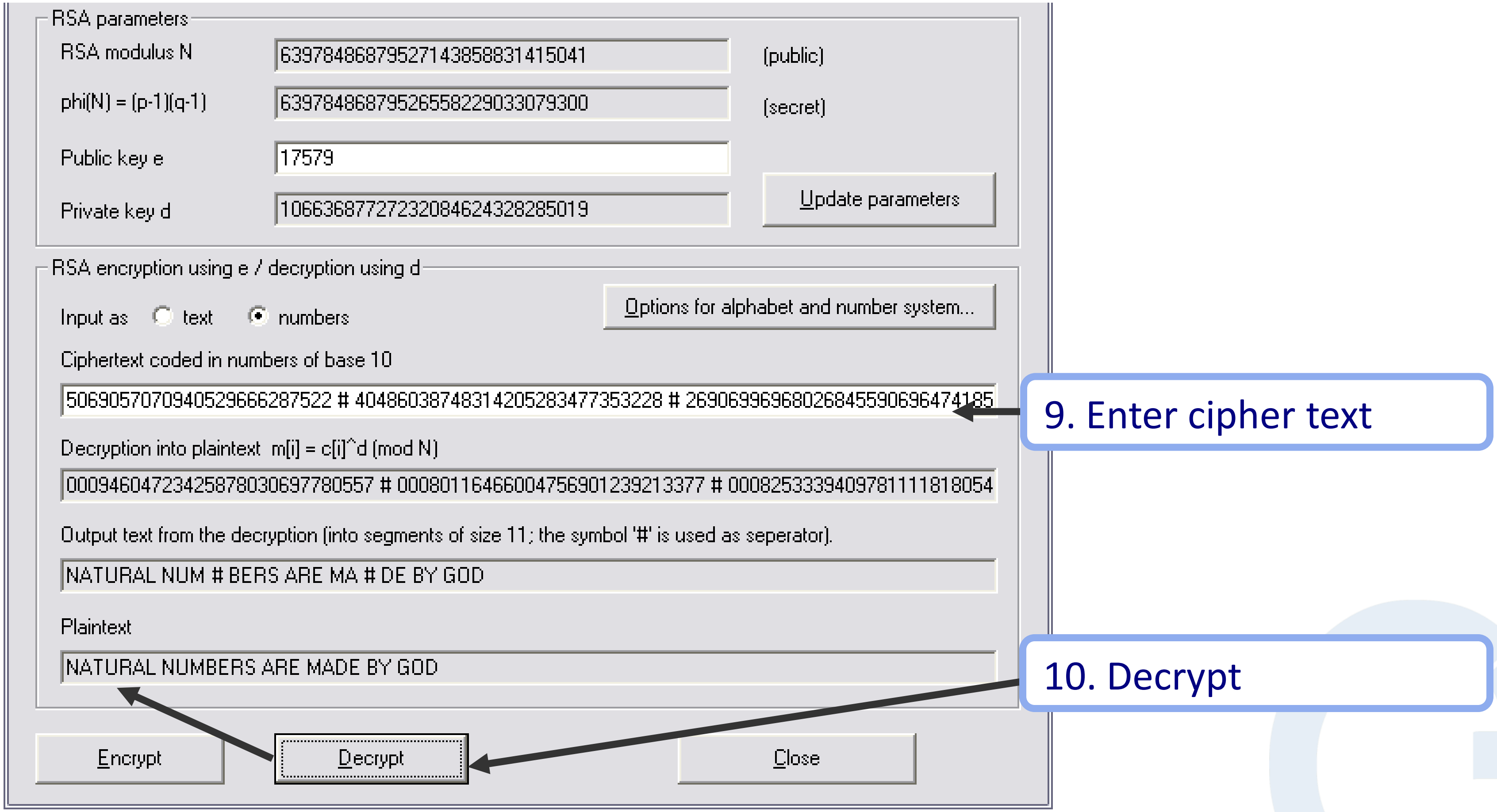
6. Select alphabet

7. Select coding method

8. Select block length

Examples (3)

Short RSA modulus: decrypt cipher text



RSA parameters

RSA modulus N: 63978486879527143858831415041 (public)

phi(N) = (p-1)(q-1): 63978486879526558229033079300 (secret)

Public key e: 17579

Private key d: 10663687727232084624328285019

RSA encryption using e / decryption using d

Input as ○ text ● numbers

Ciphertext coded in numbers of base 10

5069057070940529666287522 # 40486038748314205283477353228 # 26906996968026845590696474185

Decryption into plaintext m[i] = c[i]^d (mod N)

00094604723425878030697780557 # 00080116466004756901239213377 # 00082533394097811111818054

Output text from the decryption (into segments of size 11; the symbol '#' is used as seperator).

NATURAL NUM # BERS ARE MA # DE BY GOD

Plaintext

NATURAL NUMBERS ARE MADE BY GOD

9. Enter cipher text

10. Decrypt

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, 100 bytes up to several kB of text

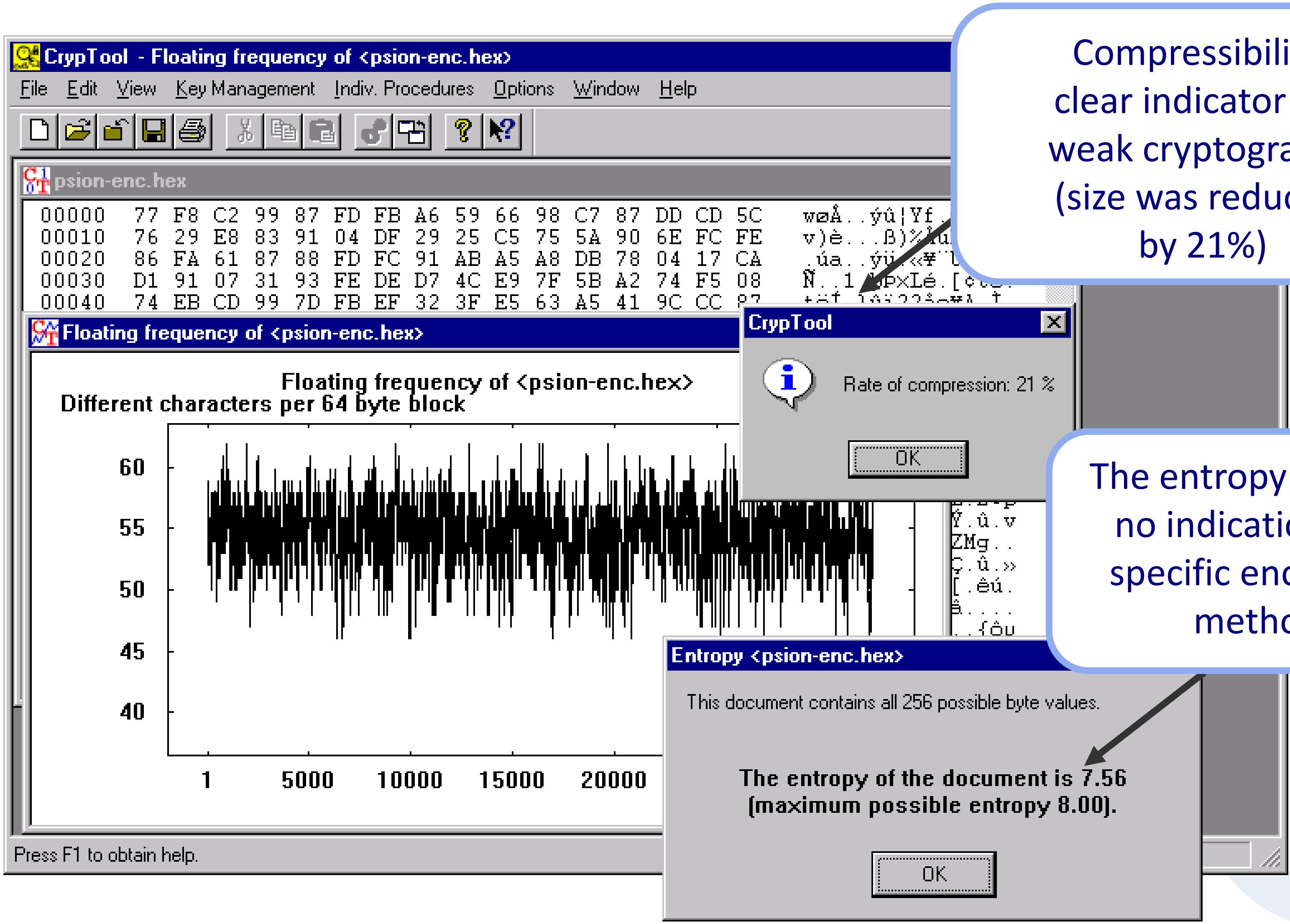
Procedure

- Pre-analysis
  - entropy
  - floating entropy
  - compression test

  } *probably classical encryption algorithm*
- Auto-correlation
- Try out automatic analysis with classical methods

Examples (4)

PSION 5 PDA – determine entropy, compression test


CrypTool - Floating frequency of <psion-enc.hex>
File Edit View Key Management Indiv. Procedures Options Window Help
psion-enc.hex

```
00000  77 F8 C2 99 87 FD FB A6 59 66 98 C7 87 DD CD 5C   wøÅ..ýû¦Yf
00010  76 29 E8 83 91 04 DF 29 25 C5 75 5A 90 6E FC FE   v)è...ß)%
00020  86 FA 61 87 88 FD FC 91 AB A5 A8 DB 78 04 17 CA   .úa..ýü
00030  D1 91 07 31 93 FE DE D7 4C E9 7F 5B A2 74 F5 08   Ñ..1
00040  74 EB CD 99 7D FB EF 32 3F E5 63 A5 41 9C CC 87
```

Floating frequency of <psion-enc.hex>
Floating frequency of <psion-enc.hex>
Different characters per 64 byte block
60
55
50
45
40
1
5000
10000
15000
20000
CrypTool

Rate of compression: 21 %

This document contains all 256 possible byte values.

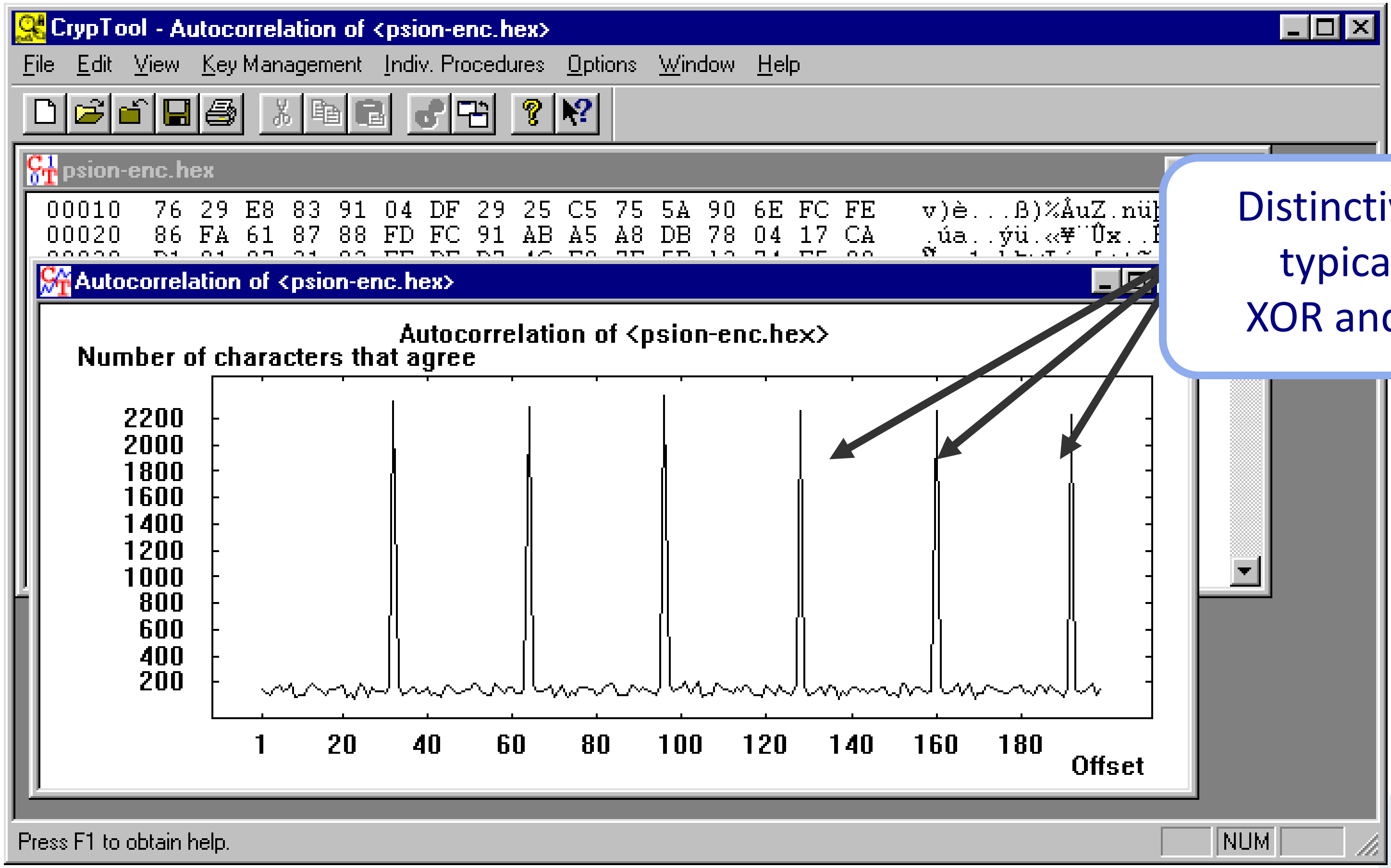
The entropy of the document is 7.56 (maximum possible entropy 8.00).

Compressibility: clear indicator for weak cryptography (size was reduced by 21%)

The entropy provides no indication for a specific encryption method.

Examples (4)

PSION 5 PDA – determine auto-correlation


CrypTool - Autocorrelation of <psion-enc.hex>
File Edit View Key Management Indiv. Procedures Options Window Help
psion-enc.hex
Autocorrelation of <psion-enc.hex>
Autocorrelation of <psion-enc.hex>
Number of characters that agree
2200
2000
1800
1600
1400
1200
1000
800
600
400
200
1
20
40
60
80
100
120
140
160
180
Offset
Press F1 to obtain help.
NUM

```
00010   76 29 E8 83 91 04 DF 29 25 C5 75 5A 90 6E FC FE    v)è...ß)%ÅuZ.nü
00020   86 FA 61 87 88 FD FC 91 AB A5 A8 DB 78 04 17 CA    .úa..ýü.«¥¨Ûx..
```

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

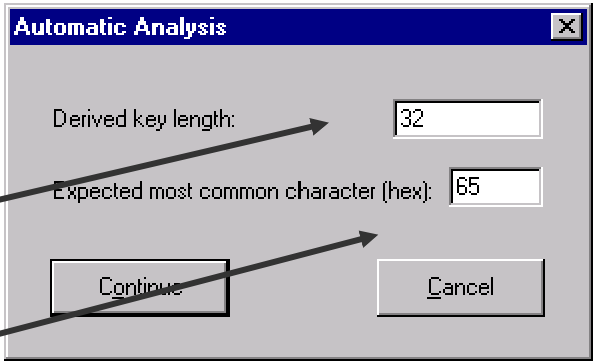
* The encrypted file is available with CrypTool (see CrypTool\examples\psion-enc.hex)

Examples (4)

PSION 5 PDA – automatic analysis

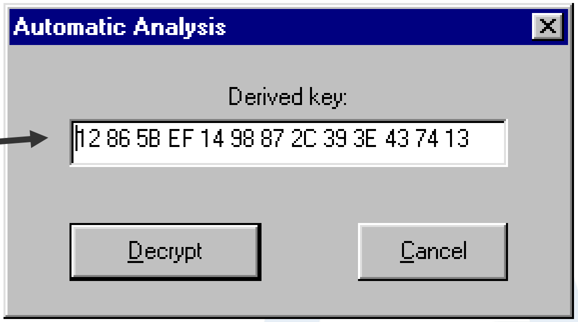
Automatic analysis using

- **Vigenère: no success**
- **XOR: no success**
- **binary addition**
  - CrypTool calculates the key length using auto-correlation: 32 bytes
  - The user can choose which character is expected to occur most frequently: “e” = 0x65 (ASCII code)
  - Analysis calculates the most likely key (based on the assumptions about distribution)
  - Result: good, but not perfect


Automatic Analysis

Derived key length: 32

Expected most common character (hex): 65


Automatic Analysis

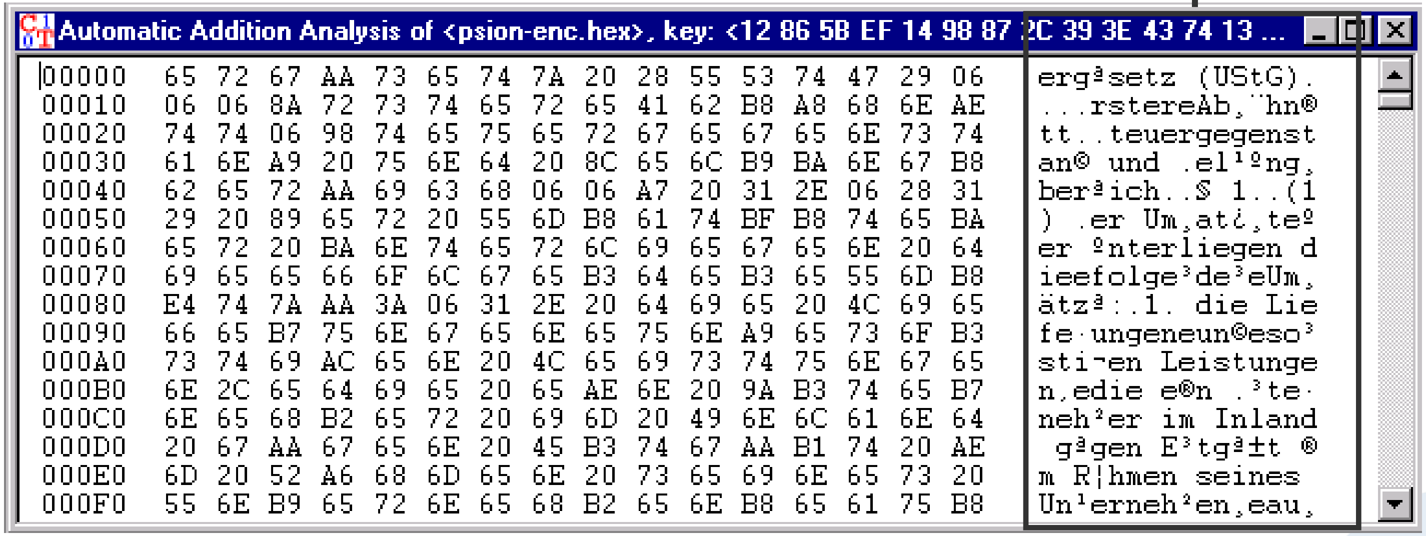
Derived key: 12 86 5B EF 14 98 87 2C 39 3E 43 74 13

Examples (4)

PSION 5 PDA – results of automatic analysis

Results of automatic analysis with assumption “binary addition”:

- Result is good, but not perfect: 24 out of 32 key bytes correct.
- The key length 32 was correctly determined.



Automatic Addition Analysis of <psion-enc.hex>, key: <12 86 5B EF 14 98 87 2C 39 3E 43 74 13 ...

```
00000   65 72 67 AA 73 65 74 7A 20 28 55 53 74 47 29 06   ergªsetz (UStG).
00010   06 06 8A 72 73 74 65 72 65 41 62 B8 A8 68 6E AE   ...rstereAb¸¨hn®
00020   74 74 06 98 74 65 75 65 72 67 65 67 65 6E 73 74   tt..teuergegenst
00030   61 6E A9 20 75 6E 64 20 8C 65 6C B9 BA 6E 67 B8   an© und .el¹ºng¸
00040   62 65 72 AA 69 63 68 06 06 A7 20 31 2E 06 28 31   berªich..§ 1..(1
00050   29 20 89 65 72 20 55 6D B8 61 74 BF B8 74 65 BA   ) .er Um¸at¿¸teº
00060   65 72 20 BA 6E 74 65 72 6C 69 65 67 65 6E 20 64   er ºnterliegen d
00070   69 65 65 66 6F 6C 67 65 B3 64 65 B3 65 55 6D B8   ieefolge³de³eUm¸
00080   E4 74 7A AA 3A 06 31 2E 20 64 69 65 20 4C 69 65   ätzª:.1. die Lie
00090   66 65 B7 75 6E 67 65 6E 65 75 6E A9 65 73 6F B3   fe·ungeneun©eso³
000A0   73 74 69 AC 65 6E 20 4C 65 69 73 74 75 6E 67 65   sti¬en Leistunge
000B0   6E 2C 65 64 69 65 20 65 AE 6E 20 9A B3 74 65 B7   n,edie e®n .³te·
000C0   6E 65 68 B2 65 72 20 69 6D 20 49 6E 6C 61 6E 64   neh²er im Inland
000D0   20 67 AA 67 65 6E 20 45 B3 74 67 AA B1 74 20 AE    gªgen E³tgª±t ®
000E0   6D 20 52 A6 68 6D 65 6E 20 73 65 69 6E 65 73 20   m R¦hmen seines
000F0   55 6E B9 65 72 6E 65 68 B2 65 6E B8 65 61 75 B8   Un¹erneh²en¸eau¸
```

- 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

Copy key to clipboard during automatic analysis

In automatic analysis hex dump,

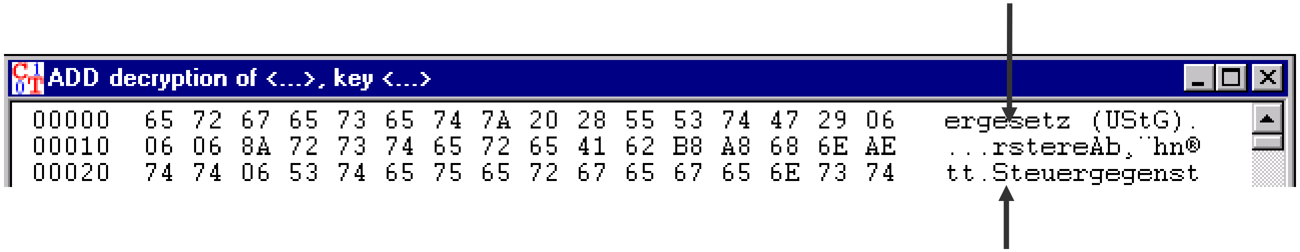
- Determine incorrect byte positions, e.g. 0xAA at position 3
- Guess and write down corresponding correct bytes: „e“ = 0x65

In encrypted initial file hex dump,

- Determine initial bytes from the calculated byte positions: 0x99
- Calculate correct key bytes with CALC.EXE: 0x99 - 0x65 = 0x34

Key from the clipboard

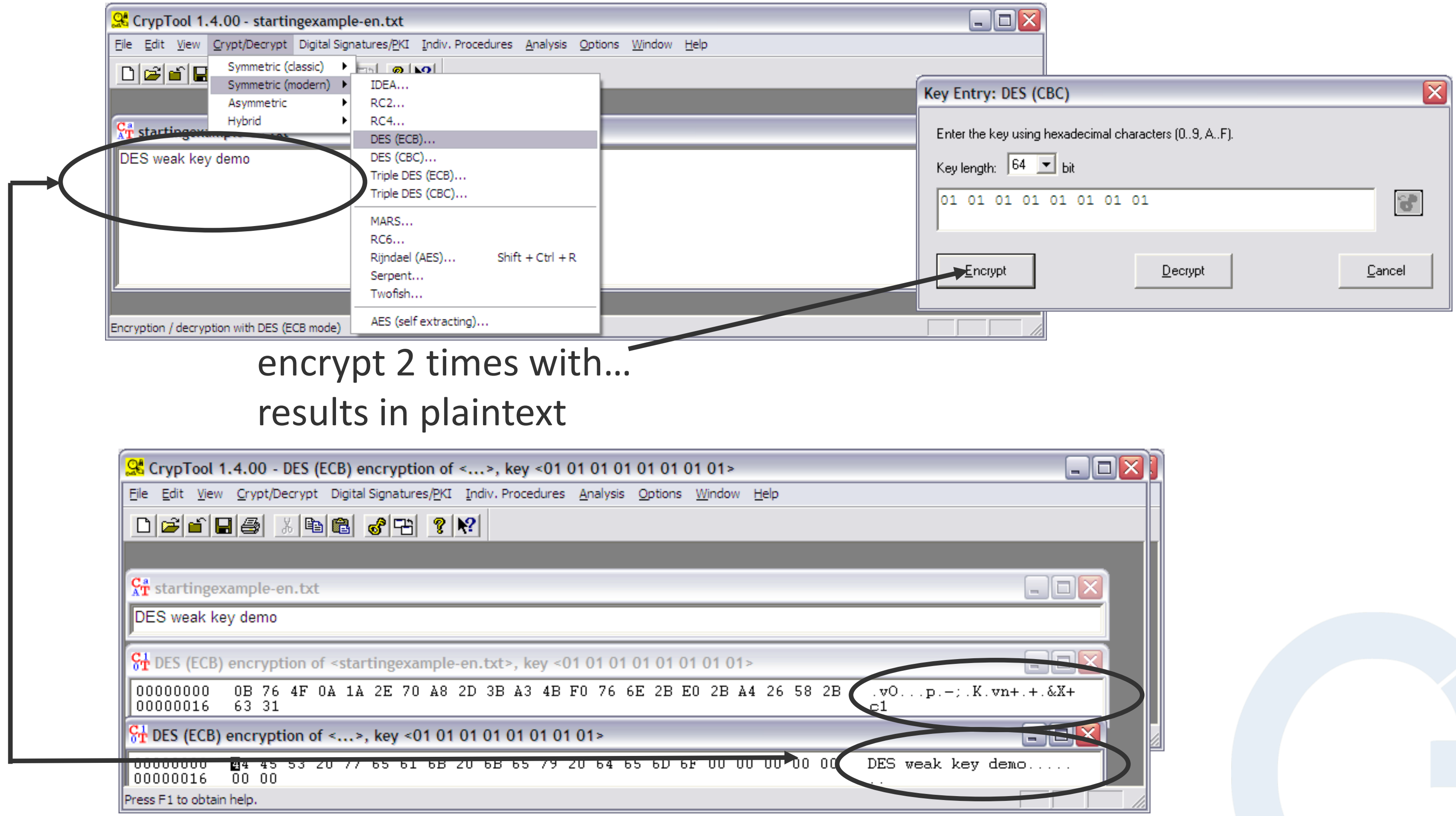
- Correct 12865B**34**1498872C393E43741396A45670235E111E907AB7C0841...
- Decrypt encrypted initial document using binary addition
- Bytes at position 3, 3+32, 3+2*32, ... are now correct



```
00000   65 72 67 65 73 65 74 7A 20 28 55 53 74 47 29 06    ergesetz (UStG).
00010   06 06 8A 72 73 74 65 72 65 41 62 B8 A8 68 6E AE    ...rstereAb,¨hn®
00020   74 74 06 53 74 65 75 65 72 67 65 67 65 6E 73 74    tt.Steuergegenst
```

Examples (5)

Weak DES key



CrypTool 1.4.00 - startingexample-en.txt
File Edit View Crypt/Decrypt Digital Signatures/PKI Indiv. Procedures Analysis Options Window Help
Symmetric (classic)
Symmetric (modern)
Asymmetric
Hybrid
IDEA...
RC2...
RC4...
DES (ECB)...
DES (CBC)...
Triple DES (ECB)...
Triple DES (CBC)...
MARS...
RC6...
Rijndael (AES)... Shift + Ctrl + R
Serpent...
Twofish...
AES (self extracting)...
Encryption / decryption with DES (ECB mode)
Key Entry: DES (CBC)
Encrypt
Decrypt
Cancel
CrypTool 1.4.00 - DES (ECB) encryption of <...>, key <01 01 01 01 01 01 01 01>
File Edit View Crypt/Decrypt Digital Signatures/PKI Indiv. Procedures Analysis Options Window Help
Press F1 to obtain help.

startingexample-en.txt

DES weak key demo

Enter the key using hexadecimal characters (0..9, A..F).

Key length: 64 bit

01 01 01 01 01 01 01 01

encrypt 2 times with…

results in plaintext

DES (ECB) encryption of <startingexample-en.txt>, key <01 01 01 01 01 01 01 01>

```
00000000   0B 76 4F 0A 1A 2E 70 A8 2D 3B A3 4B F0 76 6E 2B E0 2B A4 26 58 2B   .vO...p.-;.K.vn+.+.&X+
00000016   63 31                                                                c1
```

DES (ECB) encryption of <...>, key <01 01 01 01 01 01 01 01>

```
00000000   44 45 53 20 77 65 61 6B 20 6B 65 79 20 64 65 6D 6F 00 00 00 00 00   DES weak key demo.....
00000016   00 00                                                                ..
```

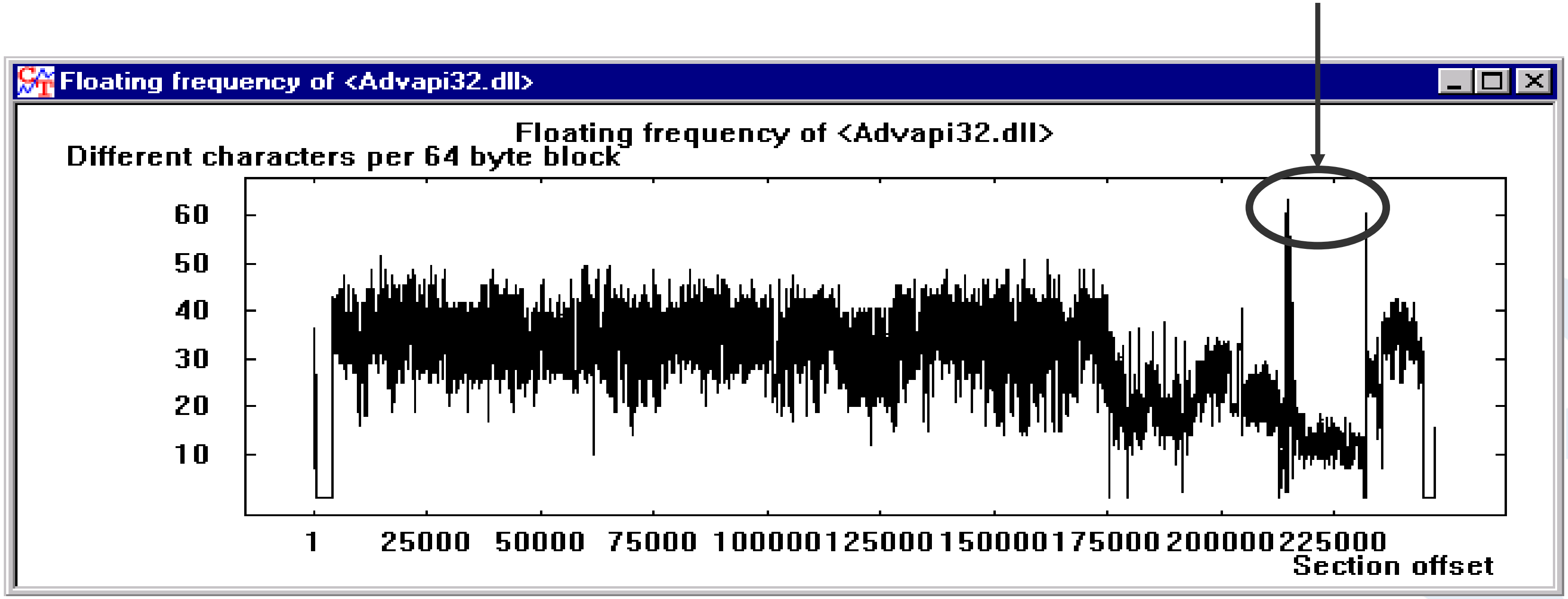
Examples (6)

Locate key material

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

Background:

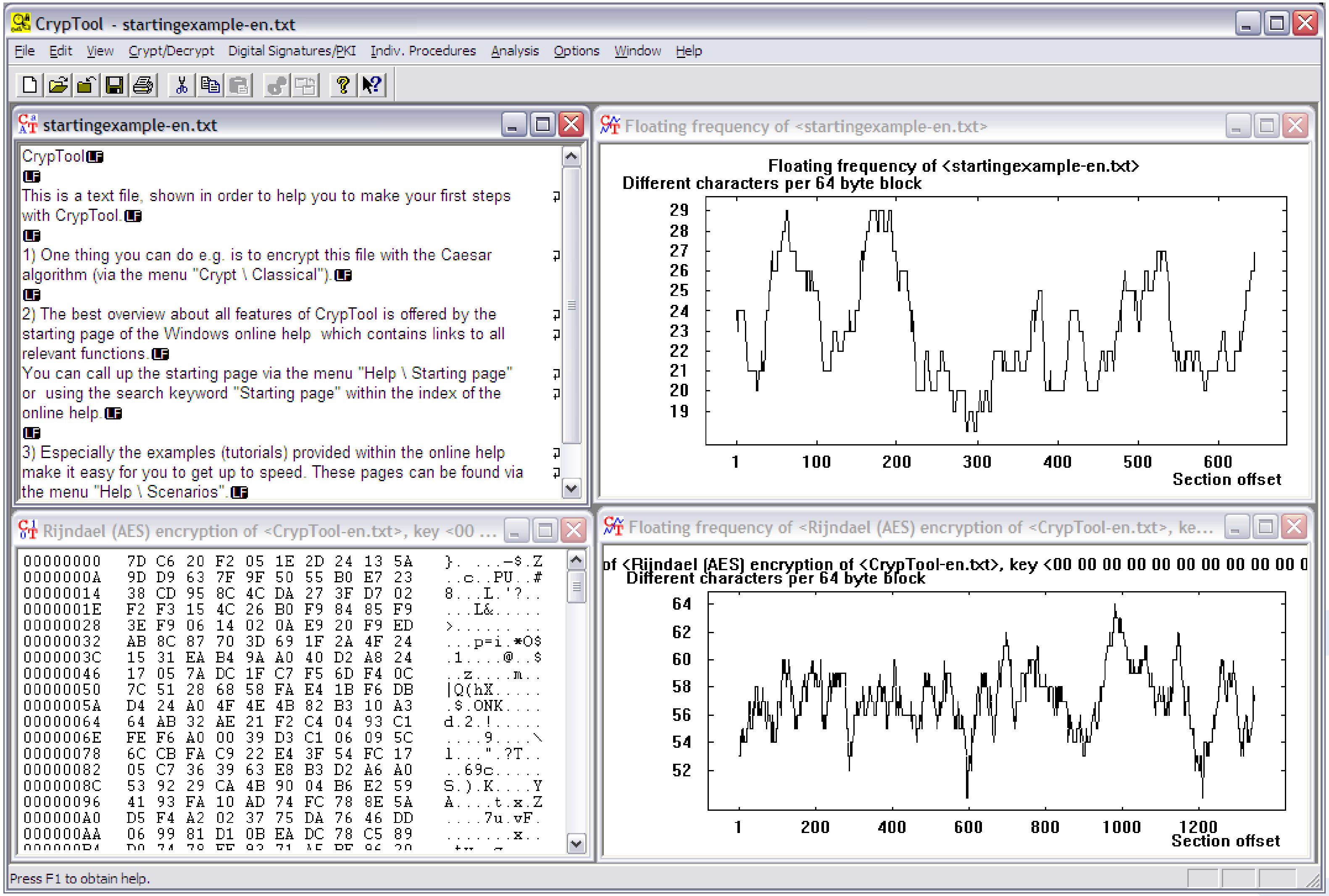
- Key data is "more random" than text or program code
- Can be recognized as peaks in the "floating frequency"
- Example: the "NSA key" in advapi32.dll (Windows NT)



Floating frequency of <Advapi32.dll>
Floating frequency of <Advapi32.dll>
Different characters per 64 byte block
60
50
40
30
20
10
1
25000
50000
75000
100000
125000
150000
175000
200000
225000
Section offset

Examples (6)

Comparison on floating frequency with other files

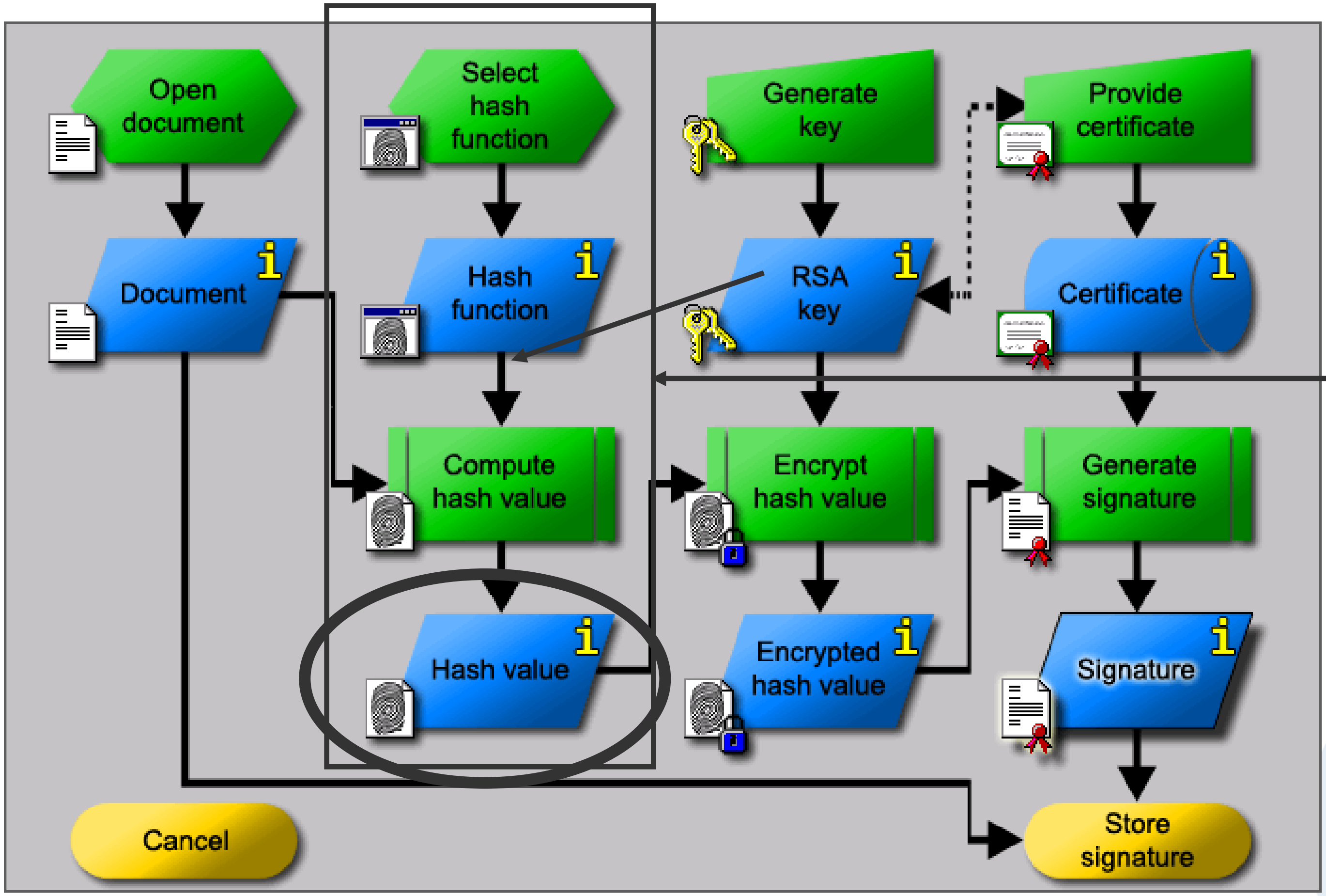

CrypTool - startingexample-en.txt
File Edit View Crypt/Decrypt Digital Signatures/PKI Indiv. Procedures Analysis Options Window Help
startingexample-en.txt
Floating frequency of <startingexample-en.txt>
Floating frequency of <startingexample-en.txt>
Different characters per 64 byte block
29
28
27
26
25
24
23
22
21
20
19
1
100
200
300
400
500
600
Section offset
Rijndael (AES) encryption of <CrypTool-en.txt>, key <00 ...
Floating frequency of <Rijndael (AES) encryption of <CrypTool-en.txt>, ke...
of <Rijndael (AES) encryption of <CrypTool-en.txt>, key <00 00 00 00 00 00 00 00 00 00 0
Different characters per 64 byte block
64
62
60
58
56
54
52
1
200
400
600
800
1000
1200
Section offset
Press F1 to obtain help.

CrypTool LF
LF
This is a text file, shown in order to help you to make your first steps with CrypTool. LF
LF
1) One thing you can do e.g. is to encrypt this file with the Caesar algorithm (via the menu "Crypt \ Classical"). LF
LF
2) The best overview about all features of CrypTool is offered by the starting page of the Windows online help which contains links to all relevant functions. LF
You can call up the starting page via the menu "Help \ Starting page" or using the search keyword "Starting page" within the index of the online help. LF
LF
3) Especially the examples (tutorials) provided within the online help make it easy for you to get up to speed. These pages can be found via the menu "Help \ Scenarios". LF

```
00000000  7D C6 20 F2 05 1E 2D 24 13 5A   }. ...-$.Z
0000000A  9D D9 63 7F 9F 50 55 B0 E7 23   ..c..PU..#
00000014  38 CD 95 8C 4C DA 27 3F D7 02   8...L.'?..
0000001E  F2 F3 15 4C 26 B0 F9 84 85 F9   ...L&.....
00000028  3E F9 06 14 02 0A E9 20 F9 ED   >.........
00000032  AB 8C 87 70 3D 69 1F 2A 4F 24   ...p=i.*O$
0000003C  15 31 EA B4 9A A0 40 D2 A8 24   .1....@..$
00000046  17 05 7A DC 1F C7 F5 6D F4 0C   ..z....m..
00000050  7C 51 28 68 58 FA E4 1B F6 DB   |Q(hX.....
0000005A  D4 24 A0 4F 4E 4B 82 B3 10 A3   .$.ONK....
00000064  64 AB 32 AE 21 F2 C4 04 93 C1   d.2.!.....
0000006E  FE F6 A0 00 39 D3 C1 06 09 5C   ....9....\
00000078  6C CB FA C9 22 E4 3F 54 FC 17   l...".?T..
00000082  05 C7 36 39 63 E8 B3 D2 A6 A0   ..69c.....
0000008C  53 92 29 CA 4B 90 04 B6 E2 59   S.).K....Y
00000096  41 93 FA 10 AD 74 FC 78 8E 5A   A....t.x.Z
000000A0  D5 F4 A2 02 37 75 DA 76 46 DD   ....7u.vF.
000000AA  06 99 81 D1 0B EA DC 78 C5 89   .......x..
```

Examples (7)

Attack on digital signature



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 based on hash collision search.

Idea:

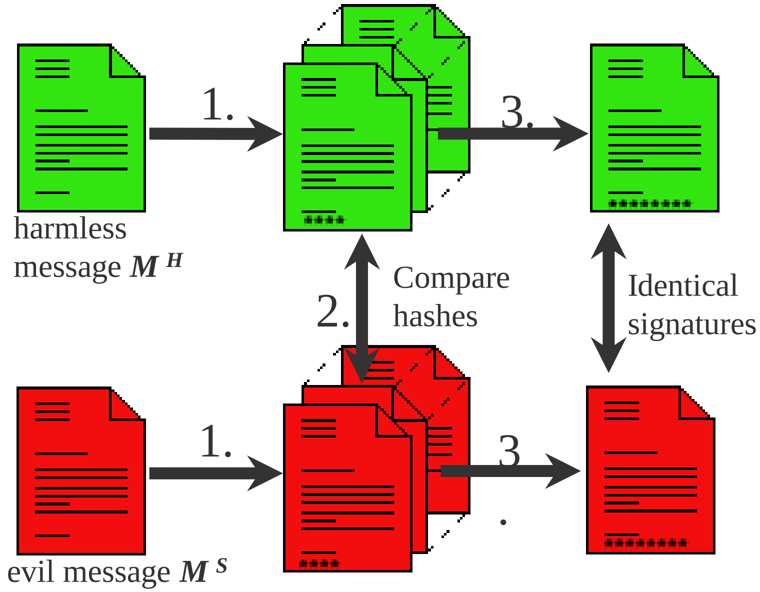
- ASCII texts can be modified by changing/inserting non-printable characters, without changing the visible content
- Modify two texts in parallel until a hash collision is found
- Exploit the birthday paradox (birthday attack)
- Generic attack applicable to all hash functions
- Can be run in parallel on many machines (not implemented)
- 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 $M_1, ..., M_N$ 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:

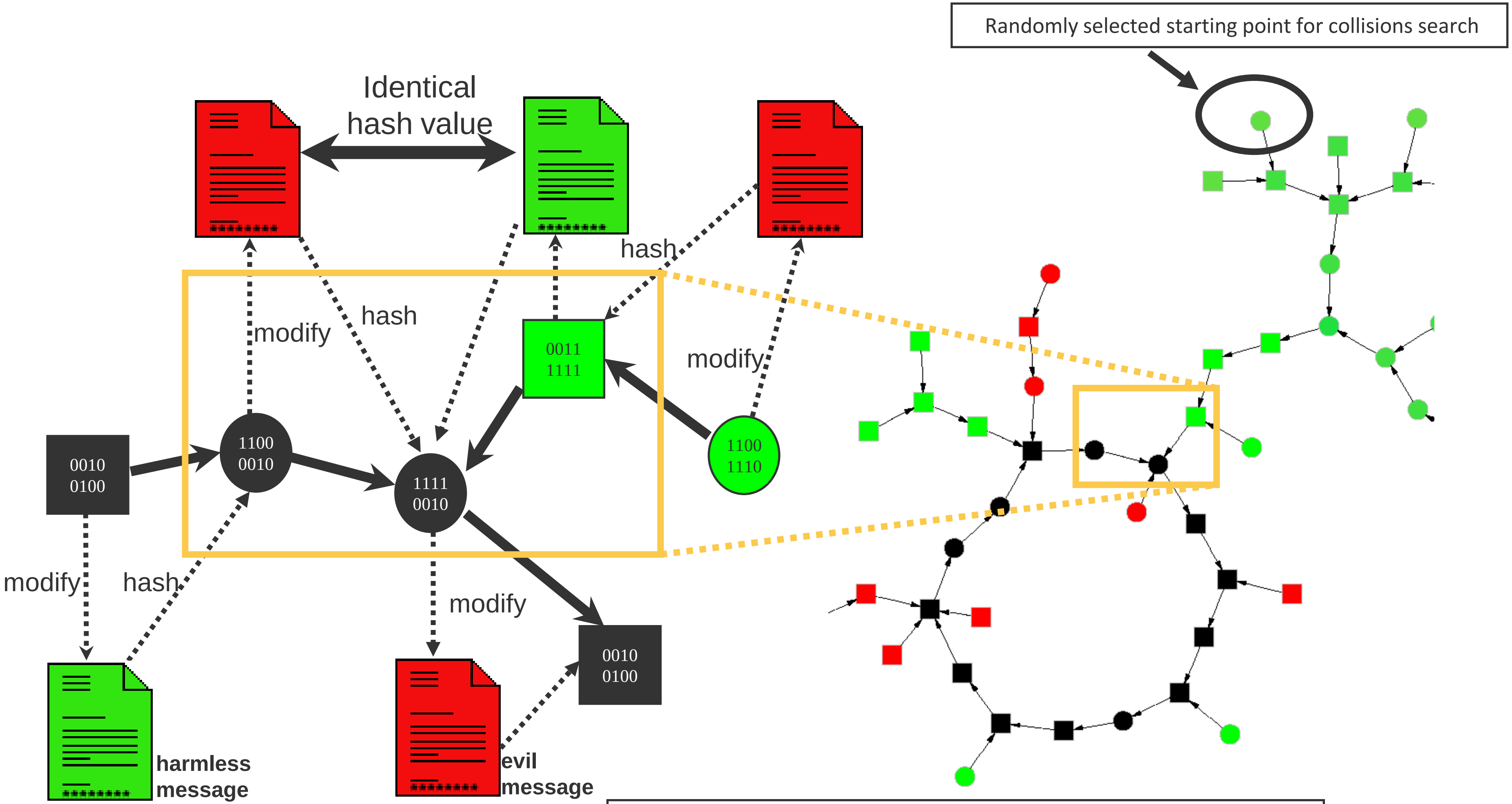
- search collision between M^H and $M_1^S, ..., M_N^S$: $N \approx 2^n$
- search collision between $M_1^H, ..., M_N^H$ and $M_1^S, ..., M_N^S$: $N \approx 2^{n/2}$

↑

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

Locate Hash Collisions (1)

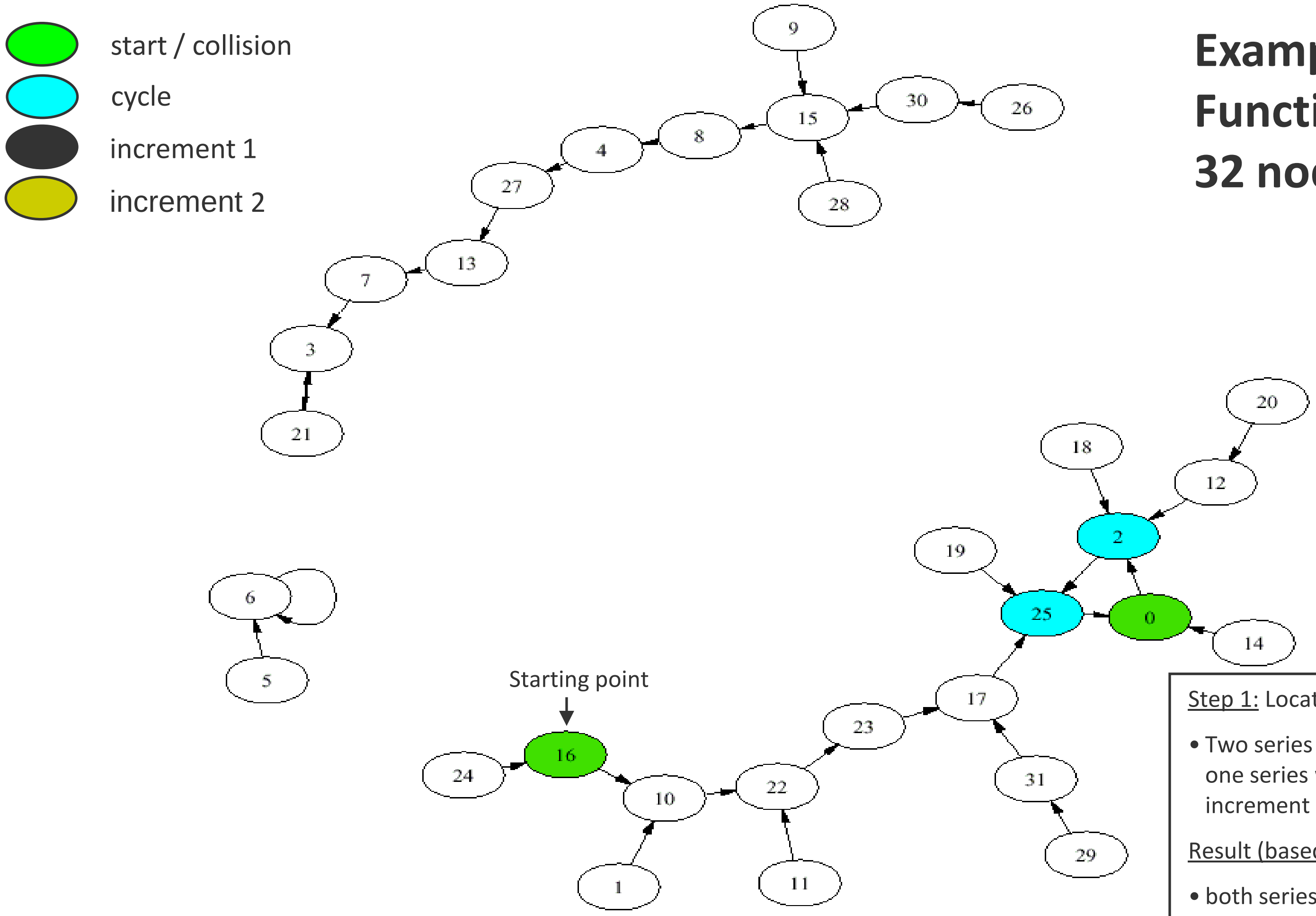
Mapping via text modifications



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

Locate Hash Collisions (2)

Floyd Algorithm: meet within the cycle



Example:
Function graph with 32 nodes

Step 1: Locate matching point within cycle:

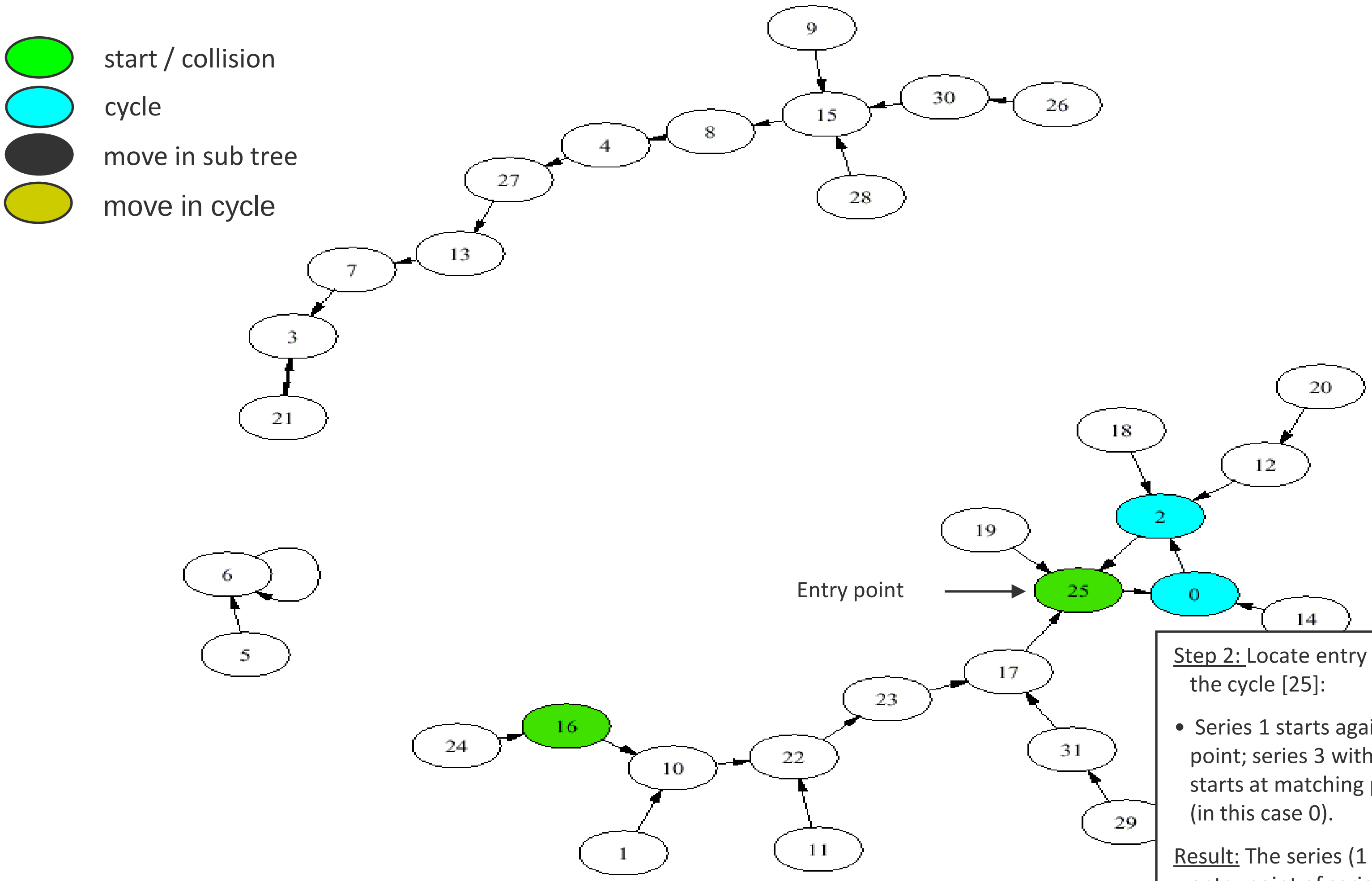
- Two series with identical starting point [16]: one series with increment 1, the other with increment 2.

Result (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



Entry point

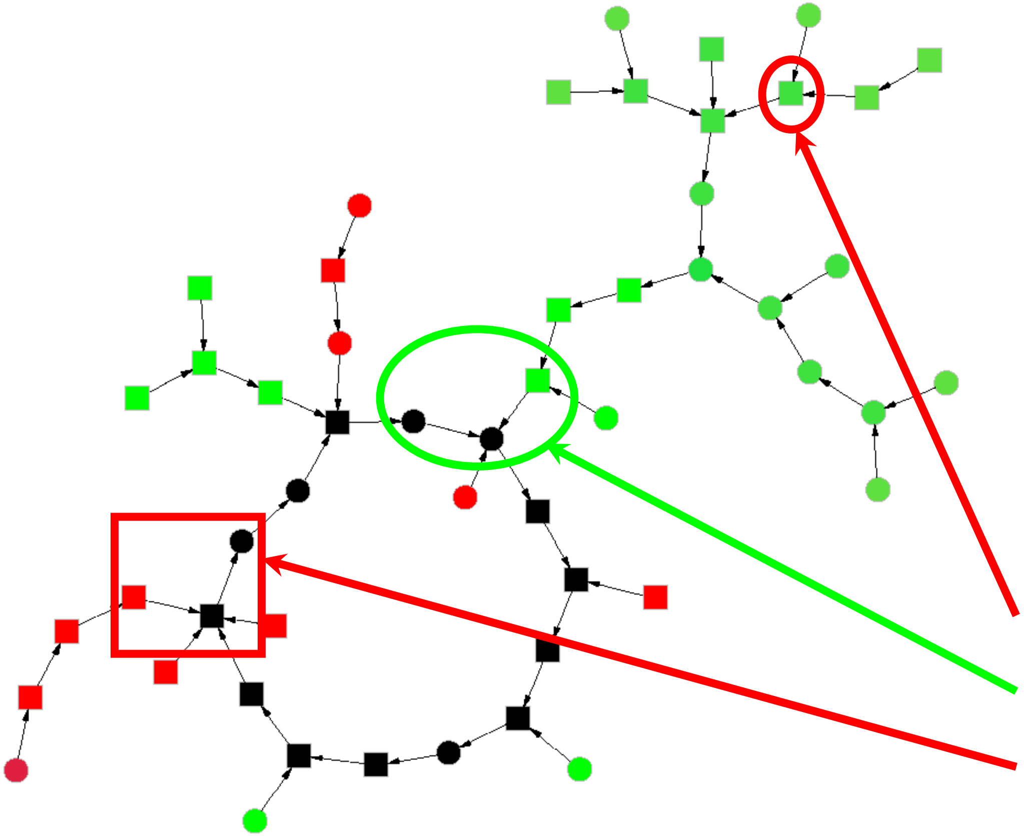
Step 2: Locate entry point of series 1 in the cycle [25]:

- 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



Starting point

Good collision

Bad collision

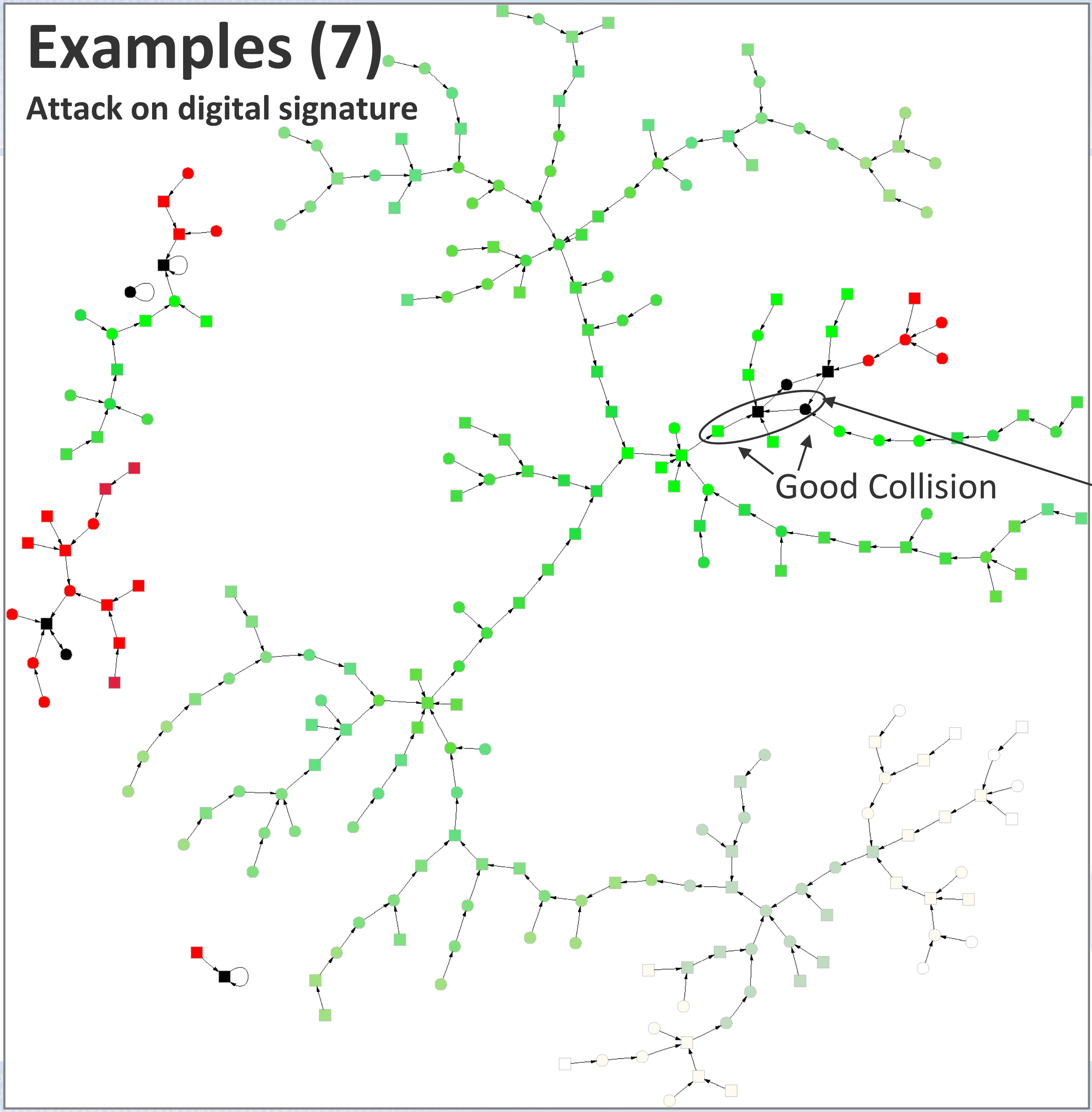
Examination of Floyd algorithm

- 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.

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

Examples (7)

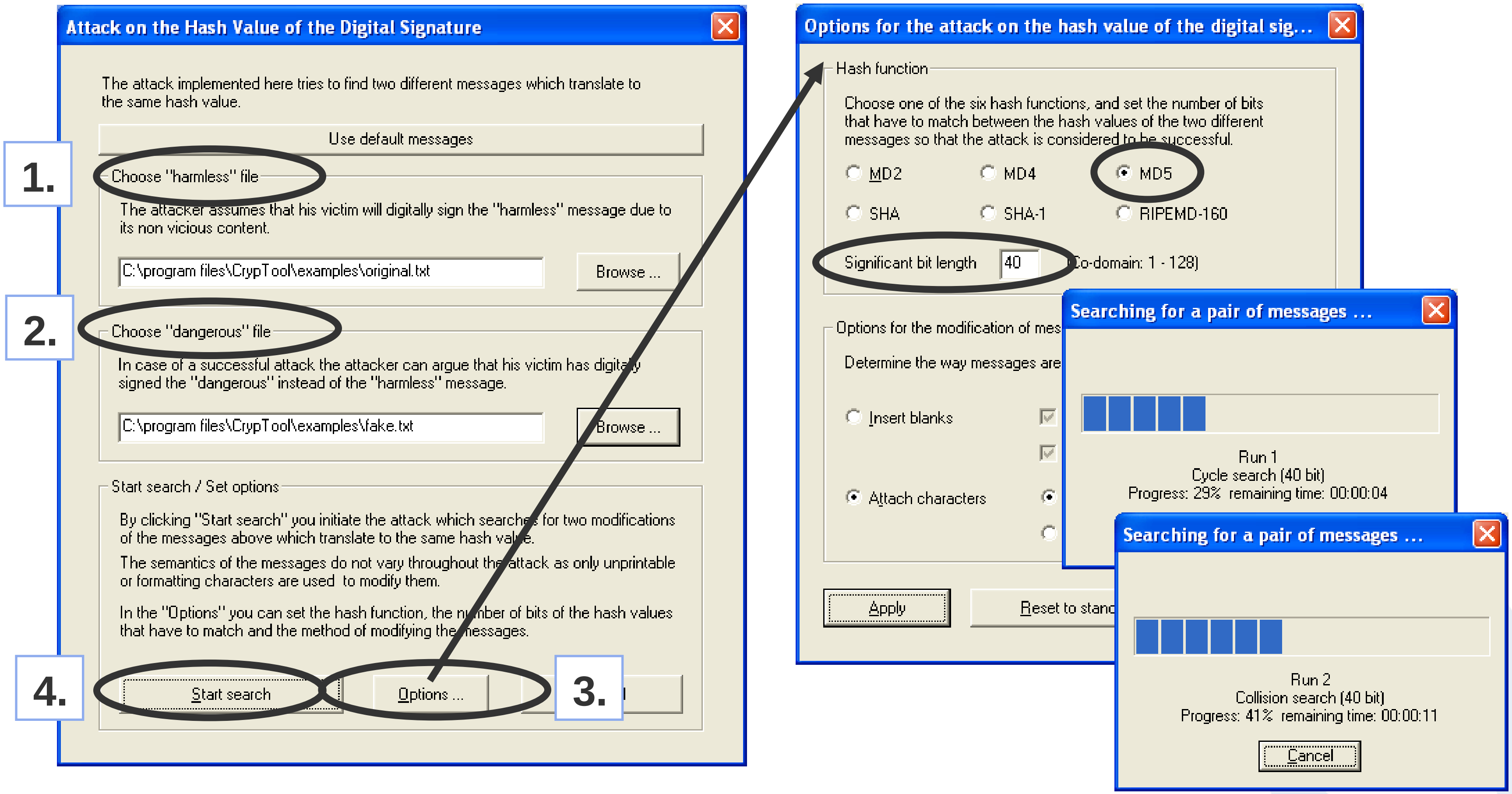
Attack on digital signature



An example for 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



Attack on the Hash Value of the Digital Signature

The attack implemented here tries to find two different messages which translate to the same hash value.

Use default messages

1. Choose "harmless" file

The attacker assumes that his victim will digitally sign the "harmless" message due to its non vicious content.

C:\program files\CrypTool\examples\original.txt — Browse ...

2. Choose "dangerous" file

In case of a successful attack the attacker can argue that his victim has digitally signed the "dangerous" instead of the "harmless" message.

C:\program files\CrypTool\examples\fake.txt — Browse ...

Start search / Set options

By clicking "Start search" you initiate the attack which searches for two modifications of the messages above which translate to the same hash value.

The semantics of the messages do not vary throughout the attack as only unprintable or formatting characters are used to modify them.

In the "Options" you can set the hash function, the number of bits of the hash values that have to match and the method of modifying the messages.

4. Start search

3. Options ...

Options for the attack on the hash value of the digital sig...

Hash function

Choose one of the six hash functions, and set the number of bits that have to match between the hash values of the two different messages so that the attack is considered to be successful.

○ MD2 ○ MD4 ● MD5

○ SHA ○ SHA-1 ○ RIPEMD-160

Significant bit length: 40 (Co-domain: 1 - 128)

Options for the modification of mes

Determine the way messages are

○ Insert blanks ☑

☑

● Attach characters ●

○

Apply — Reset to stand

Searching for a pair of messages ...

Run 1
Cycle search (40 bit)
Progress: 29% remaining time: 00:00:04

Searching for a pair of messages ...

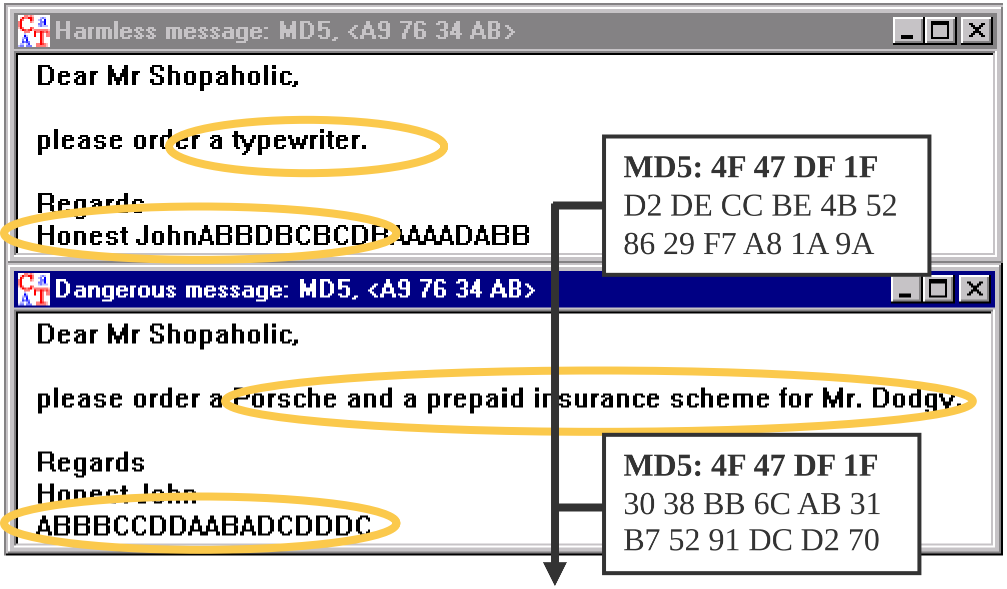
Run 2
Collision search (40 bit)
Progress: 41% remaining time: 00:00:11

Cancel

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

Examples (7)

Attack on digital signature: results



Harmless message: MD5, <A9 76 34 AB>

Dear Mr Shopaholic,

please order a typewriter.

Regards
Honest JohnABBDBCBCDBAAAADABB

MD5: 4F 47 DF 1F
D2 DE CC BE 4B 52
86 29 F7 A8 1A 9A

Dangerous message: MD5, <A9 76 34 AB>

Dear Mr Shopaholic,

please order a Porsche and a prepaid insurance scheme for Mr. Dodgy.

Regards
Honest John
ABBBCCDDAABADCDDDC

MD5: 4F 47 DF 1F
30 38 BB 6C AB 31
B7 52 91 DC D2 70

The first 32 bits of the hash values are identical.

Experimental results

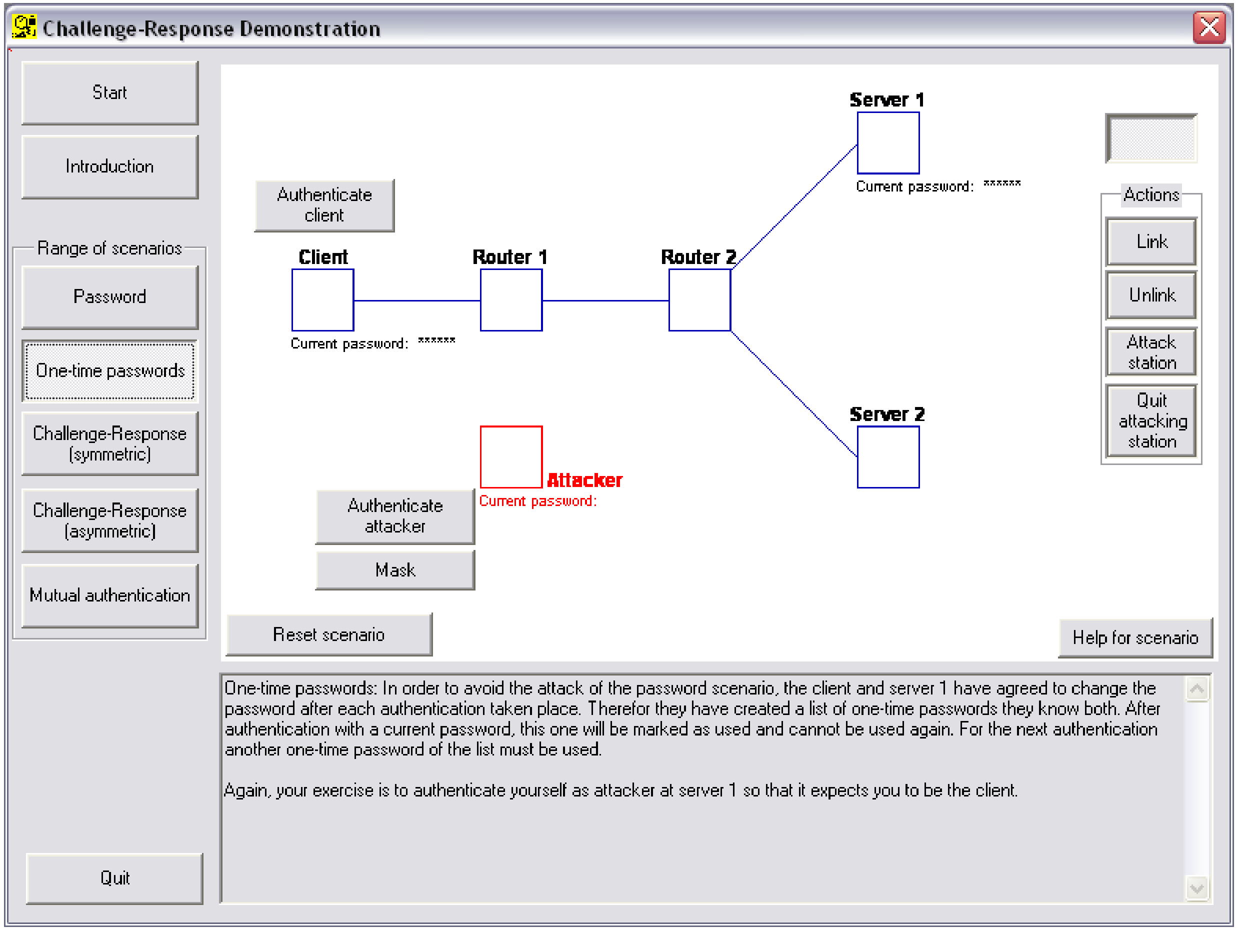
- 72 Bit *partial collision* (equality of the first 72 hash value bits) were found in a couple of days on a single PC.
- Signatures using hash values of up to 128 bit can be attacked today using massive parallel search!
- Use hash values of at least 160 bit length.

In addition to the interactive handling:

Automated offline feature in CrypTool: Execute and log the results for entire sets of parameter configurations. Available through command line execution of CrypTool.

Examples (8)

Authentication in a client-server environment



One-time passwords: In order to avoid the attack of the password scenario, the client and server 1 have agreed to change the password after each authentication taken place. Therefor they have created a list of one-time passwords they know both. After authentication with a current password, this one will be marked as used and cannot be used again. For the next authentication another one-time password of the list must be used.

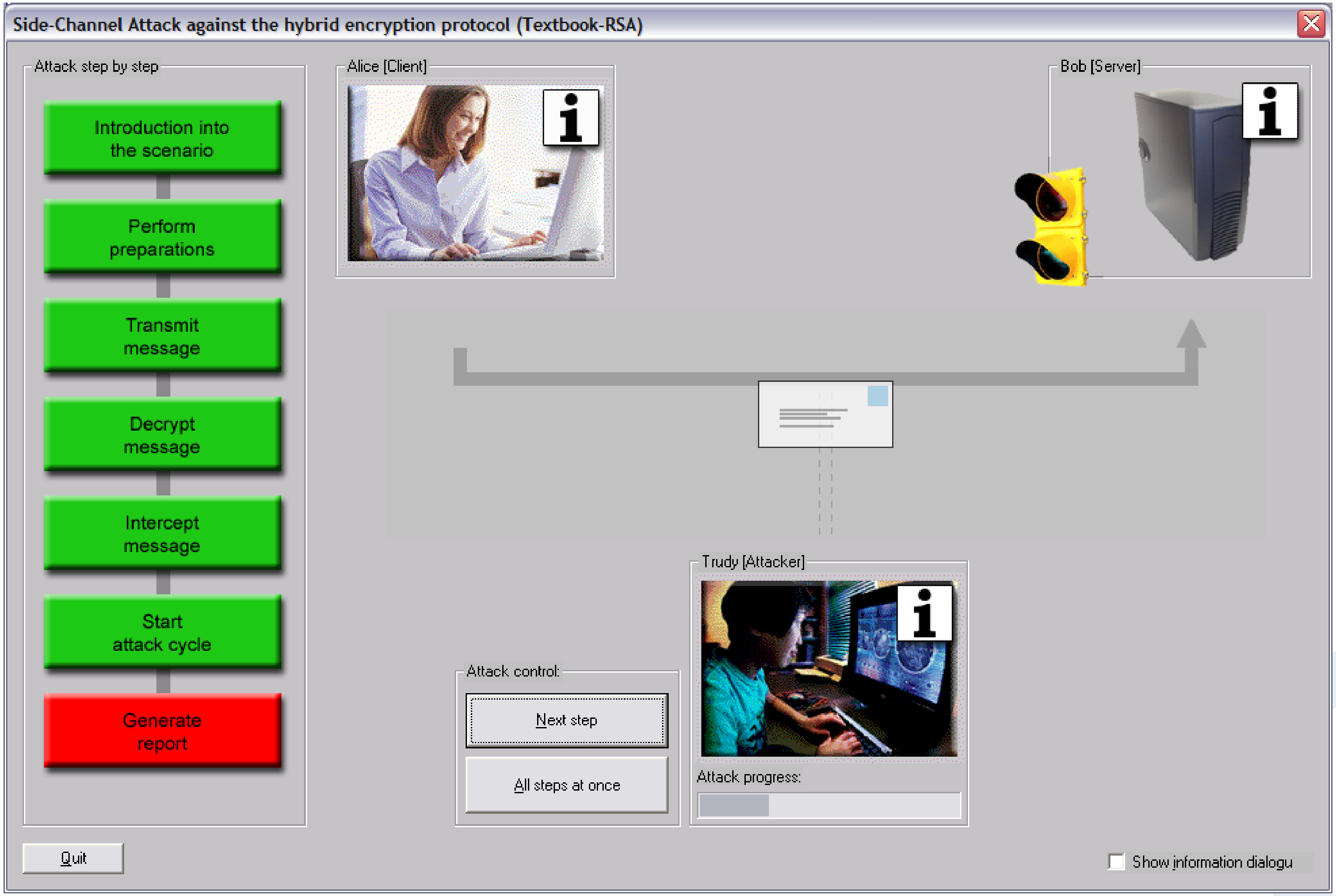
Again, your exercise is to authenticate yourself as attacker at server 1 so that it expects you to be the client.

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

- Interactive demo for different authentication methods.
- Defined opportunities of the attacker.
- You can play the role of an attacker.
- **Learning effect:** 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)

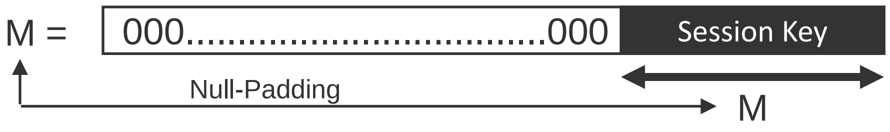
Idea for this side channel attack

Ulrich Kühn *“Side-channel attacks on textbook RSA and ElGamal encryption”,* 2003

Prerequisites:

- RSA encryption: $C = M^e$ (mod N) and decryption: $M = C^d$ mod N.
- 128-Bit session keys (in M) are „word book encoded“ (null padding).
- The server knows the secret key d and
  - uses after decryption the 128 least significant bits only (no validation of zero padding bits) (that means the server does not recognize if there is something other than zero).
  - Prompts an error message, if the encryption attempt results in a wrong session key (decrypted text can not be interpreted by the server). In all other cases there will be no message.

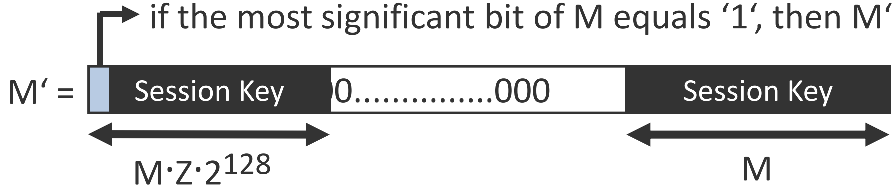
Idea for attack: Approximation for Z out of the equation N = M * Z per M = $\lfloor|N/Z|\rfloor$



$C = M^e$ (mod N)

All bit positions for Z are successively calculated: For every step one gets 1 further bit. The attacker 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 gets a bit for Z.

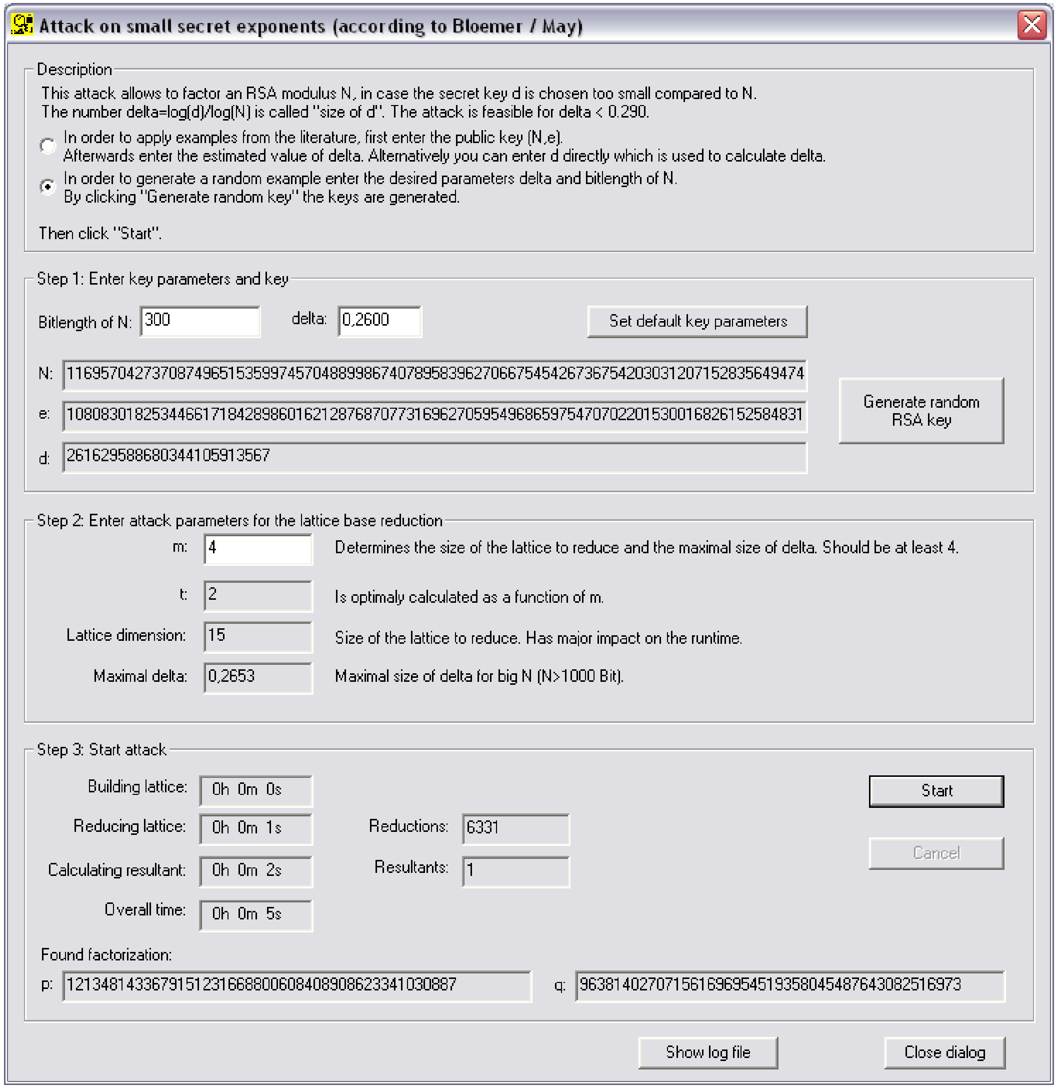
if the most significant bit of M equals ‘1‘, then M‘ unequal M mod 2^{128} ,



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

Examples (10)

Mathematics: Attacks on RSA using lattice reduction



Attack on small secret exponents (according to Bloemer / May)

Description

This attack allows to factor an RSA modulus N, in case the secret key d is chosen too small compared to N.
The number delta=log(d)/log(N) is called "size of d". The attack is feasible for delta < 0.290.

○ In order to apply examples from the literature, first enter the public key (N,e).
Afterwards enter the estimated value of delta. Alternatively you can enter d directly which is used to calculate delta.

● In order to generate a random example enter the desired parameters delta and bitlength of N.
By clicking "Generate random key" the keys are generated.

Then click "Start".

Step 1: Enter key parameters and key

Bitlength of N: 300 delta: 0,2600

N: 116957042737087496515359974570488998674078958396270667545426736754203031207152835649474

e: 108083018253446617184289860162128768707731696270595496865975470702201530016826152584831

d: 2616295886803441059135 67

Step 2: Enter attack parameters for the lattice base reduction

m: 4 — Determines the size of the lattice to reduce and the maximal size of delta. Should be at least 4.

t: 2 — Is optimaly calculated as a function of m.

Lattice dimension: 15 — Size of the lattice to reduce. Has major impact on the runtime.

Maximal delta: 0,2653 — Maximal size of delta for big N (N>1000 Bit).

Step 3: Start attack

Building lattice: 0h 0m 0s

Reducing lattice: 0h 0m 1s Reductions: 6331

Calculating resultant: 0h 0m 2s Resultants: 1

Overall time: 0h 0m 5s

Found factorization:

p: 121348143367915123166880060840890862334103088 7

q: 963814027071561696954519358045487643082516973

- Shows how the parameters of the RSA method have to be chosen, so that the algorithm resists the lattice reduction attacks described in current literature.
- **3 variants**
  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

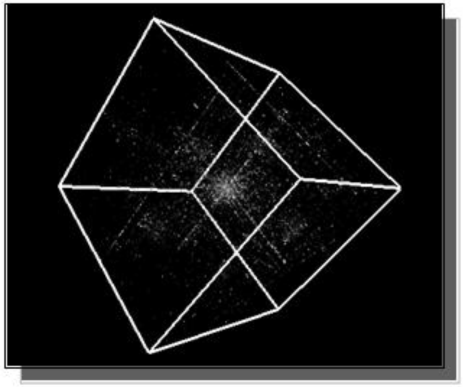
Menu: „Analysis“ \ „Asymmetric Encryption” \ „Lattice Based Attacks on RSA“ \ …

Examples (11)

Random analysis with 3-D visualisation

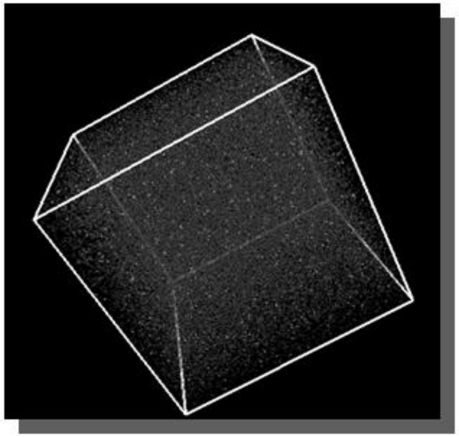
3-D visualisation for random analysis

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 using menu: „Analysis“ \ „Analyse Randomness“ \ „3-D Visualization“
- Result: **structures are easily recognisable**

Example 2



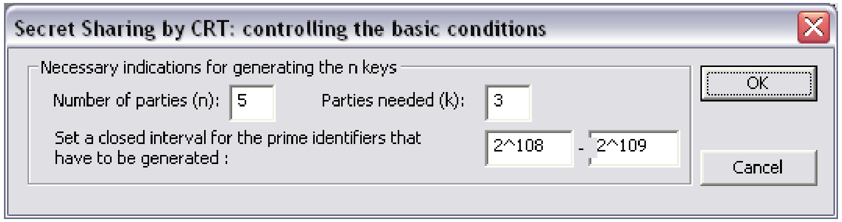
- Generation of random numbers: „Indiv. Procedures“ \ “Tools” \ „Generate Random Numbers“
- It is recommended to generate at least 100.000 random bytes
- 3-D analysis using menu: „Analysis“ \ „Analyse Randomness“ \ „3-D Visualization”
- Result: **uniform distribution (no structures are recognisable)**

Examples (12)

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

Secret sharing example (1):

- **Problem:**
  - 5 people get a single key
  - To gain access at least 3 of the 5 people have to be present
- **Menu:** „Indiv. Procedures“ \ „Chinese Remainder Theorem Applications“ \ „Secret Sharing by CRT“
- **„Options“** allows to configure more details of the method.



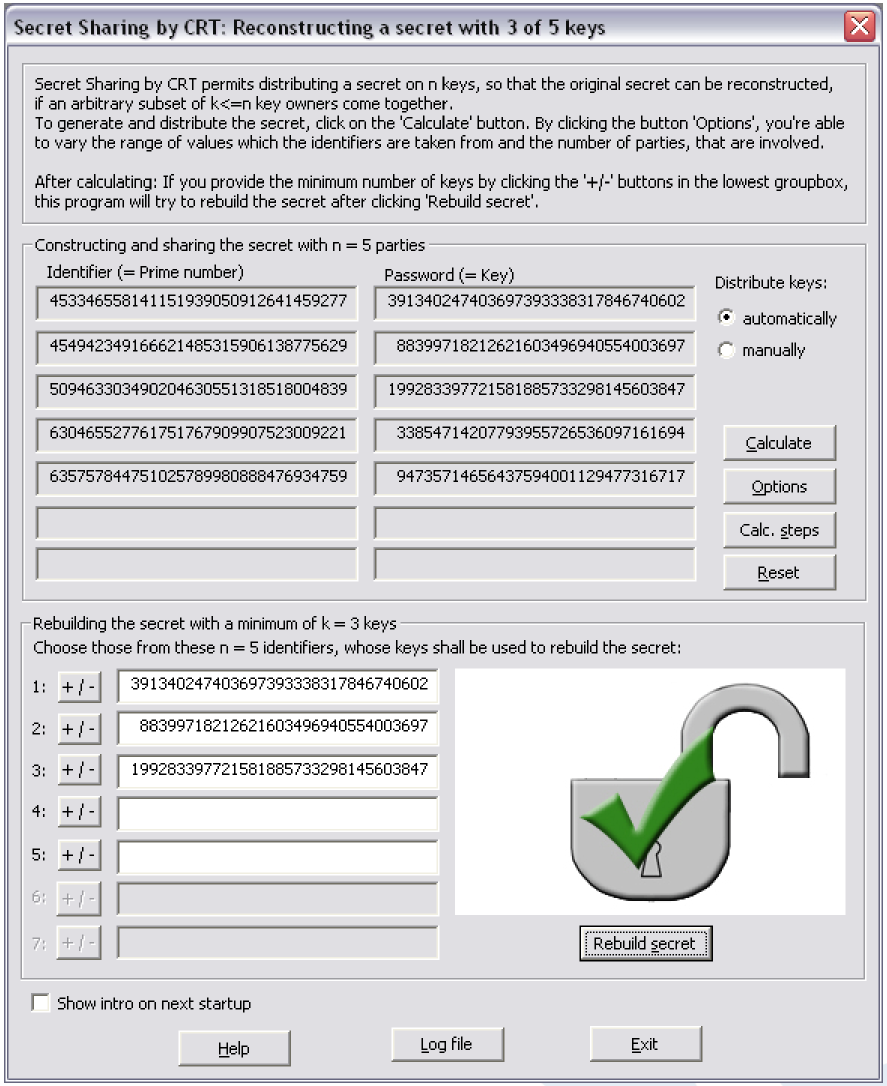
Secret Sharing by CRT: controlling the basic conditions

Necessary indications for generating the n keys

Number of parties (n): 5   Parties needed (k): 3

Set a closed interval for the prime identifiers that have to be generated : 2^108 - 2^109

- **„Calc. steps“** shows all steps to generate the key.



Secret Sharing by CRT: Reconstructing a secret with 3 of 5 keys

Secret Sharing by CRT permits distributing a secret on n keys, so that the original secret can be reconstructed, if an arbitrary subset of k<=n key owners come together.
To generate and distribute the secret, click on the 'Calculate' button. By clicking the button 'Options', you're able to vary the range of values which the identifiers are taken from and the number of parties, that are involved.

After calculating: If you provide the minimum number of keys by clicking the '+/-' buttons in the lowest groupbox, this program will try to rebuild the secret after clicking 'Rebuild secret'.

Constructing and sharing the secret with n = 5 parties

Identifier (= Prime number)	Password (= Key)
45334655814115193905091264145927 7	3913402474036973933383178467406 02
45494234916662148531590613877562 9	883997182126216034969405540036 97
50946330349020463055131851800483 9	1992833977215818857332981456038 47
63046552776175176790990752300922 1	338547142077939557265360971616 94
63575784475102578998088847693475 9	947357146564375940011294773167 17

Distribute keys: ● automatically ○ manually

Rebuilding the secret with a minimum of k = 3 keys
Choose those from these n = 5 identifiers, whose keys shall be used to rebuild the secret:

1: +/- 391340247403697393338317846740602
2: +/- 88399718212621603496940554003697
3: +/- 199283397721581885733298145603847
4: +/-
5: +/-
6: +/-
7: +/-

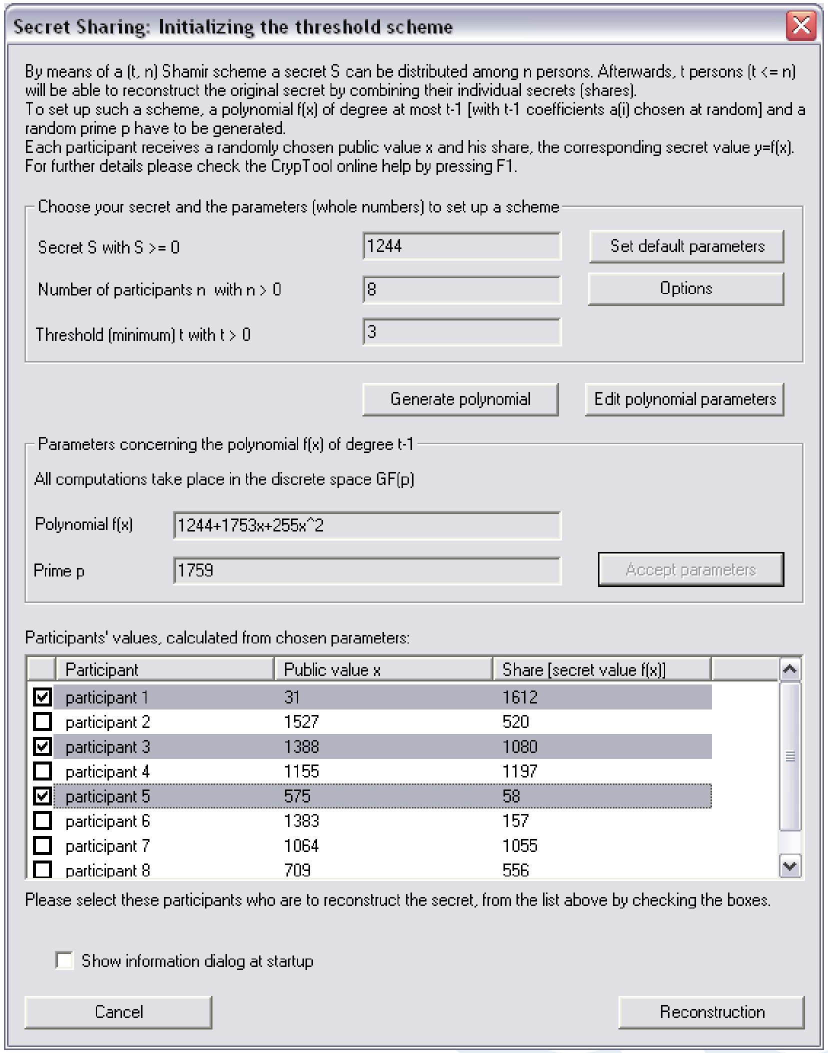
☐ Show intro on next startup

Examples (12)

Shamir secret sharing

Secret sharing example (2)

- **Problem**
  - A secret value should be split for n people.
  - t out of n people are required to restore the secret value K.
  - (t, n) threshold scheme
- **Menu:** „Indiv. Procedures“ \ „Secret Sharing Demonstration (Shamir)“
  1. Enter the secret K, number of persons n and threshold t
  2. Generate polynom
  3. Use parameters
- Using **„Reconstruction“** the secret can be restored



Secret Sharing: Initializing the threshold scheme

By means of a (t, n) Shamir scheme a secret S can be distributed among n persons. Afterwards, t persons (t <= n) will be able to reconstruct the original secret by combining their individual secrets (shares).
To set up such a scheme, a polynomial f(x) of degree at most t-1 [with t-1 coefficients a(i) chosen at random] and a random prime p have to be generated.
Each participant receives a randomly chosen public value x and his share, the corresponding secret value y=f(x).
For further details please check the CrypTool online help by pressing F1.

Choose your secret and the parameters (whole numbers) to set up a scheme

Secret S with S >= 0: 1244

Number of participants n with n > 0: 8

Threshold (minimum) t with t > 0: 3

Parameters concerning the polynomial f(x) of degree t-1

All computations take place in the discrete space GF(p)

Polynomial f(x): 1244+1753x+255x^2

Prime p: 1759

Participants' values, calculated from chosen parameters:

	Participant	Public value x	Share (secret value f(x))
☑	participant 1	31	1612
☐	participant 2	1527	520
☑	participant 3	1388	1080
☐	participant 4	1155	1197
☑	participant 5	575	58
☐	participant 6	1383	157
☐	participant 7	1064	1055
☐	participant 8	709	556

Please select these participants who are to reconstruct the secret, from the list above by checking the boxes.

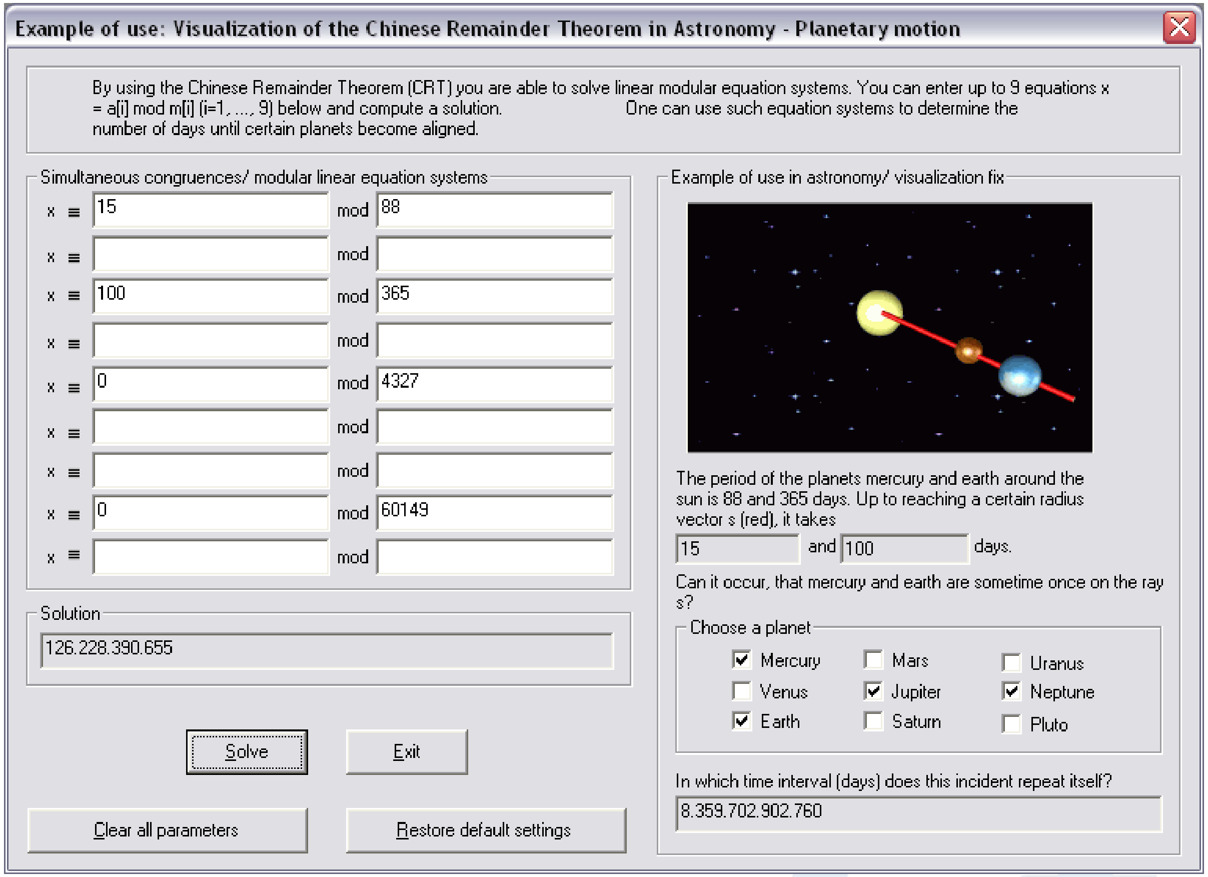
☐ Show information dialog at startup

Examples (13)

Implementation of CRT to solve linear modular equation systems

Scenario in astronomy

- How long does it take until 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.



Example of use: Visualization of the Chinese Remainder Theorem in Astronomy - Planetary motion

By using the Chinese Remainder Theorem (CRT) you are able to solve linear modular equation systems. You can enter up to 9 equations x = a[i] mod m[i] (i=1, ..., 9) below and compute a solution. One can use such equation systems to determine the number of days until certain planets become aligned.

Simultaneous congruences/ modular linear equation systems

x ≡ 15 mod 88
x ≡ mod
x ≡ 100 mod 365
x ≡ mod
x ≡ 0 mod 4327
x ≡ mod
x ≡ mod
x ≡ 0 mod 60149
x ≡ mod

Solution

126.228.390.655

Example of use in astronomy/ visualization fix

The period of the planets mercury and earth around the sun is 88 and 365 days. Up to reaching a certain radius vector s (red), it takes

15 and 100 days.

Can it occur, that mercury and earth are sometime once on the ray s?

Choose a planet

☑ Mercury ☐ Mars ☐ Uranus
☐ Venus ☑ Jupiter ☑ Neptune
☑ Earth ☐ Saturn ☐ Pluto

In which time interval (days) does this incident repeat itself?

8.359.702.902.760

Examples (14)

Visualisation of symmetric encryption methods using ANIMAL (1)

Animated visualisation of several symmetric algorithms

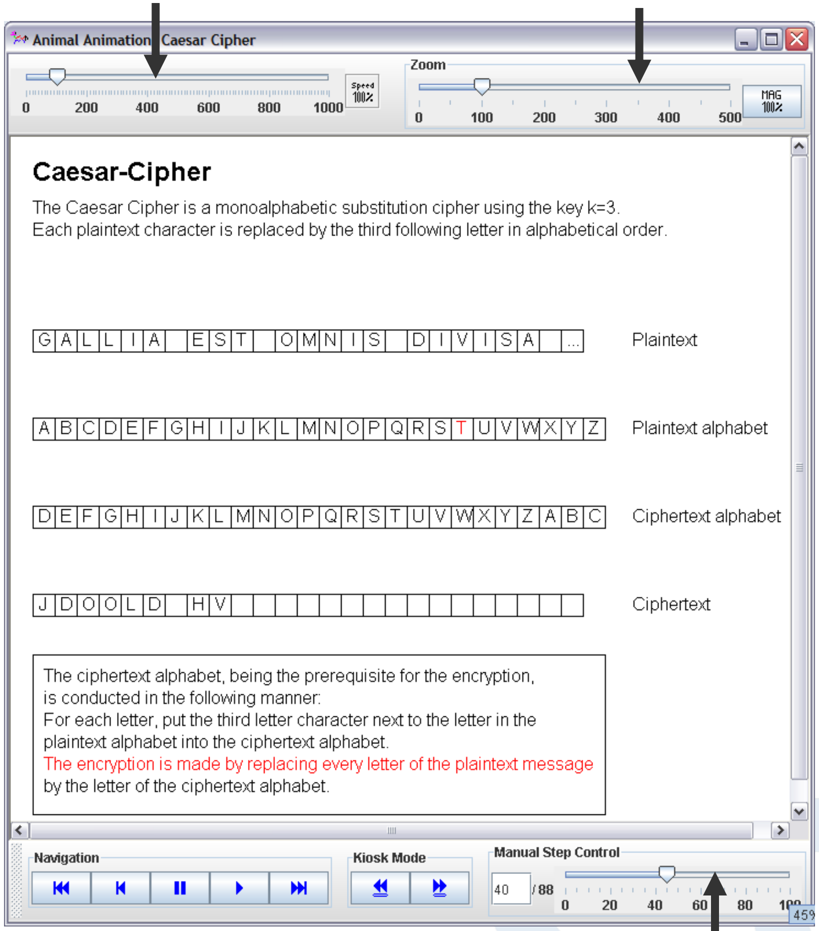
- Caesar
- Vigenère
- Nihilist
- DES

CrypTool

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

Animation speed

Scaling of visualisation



Caesar-Cipher

The Caesar Cipher is a monoalphabetic substitution cipher using the key k=3.
Each plaintext character is replaced by the third following letter in alphabetical order.

G A L L I A   E S T   O M N I S   D I V I S A   ... — Plaintext

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 — Plaintext alphabet

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 — Ciphertext alphabet

J D O O L D   H V — Ciphertext

The ciphertext alphabet, being the prerequisite for the encryption, is conducted in the following manner:
For each letter, put the third letter character next to the letter in the plaintext alphabet into the ciphertext alphabet.
The encryption is made by replacing every letter of the plaintext message by the letter of the ciphertext alphabet.

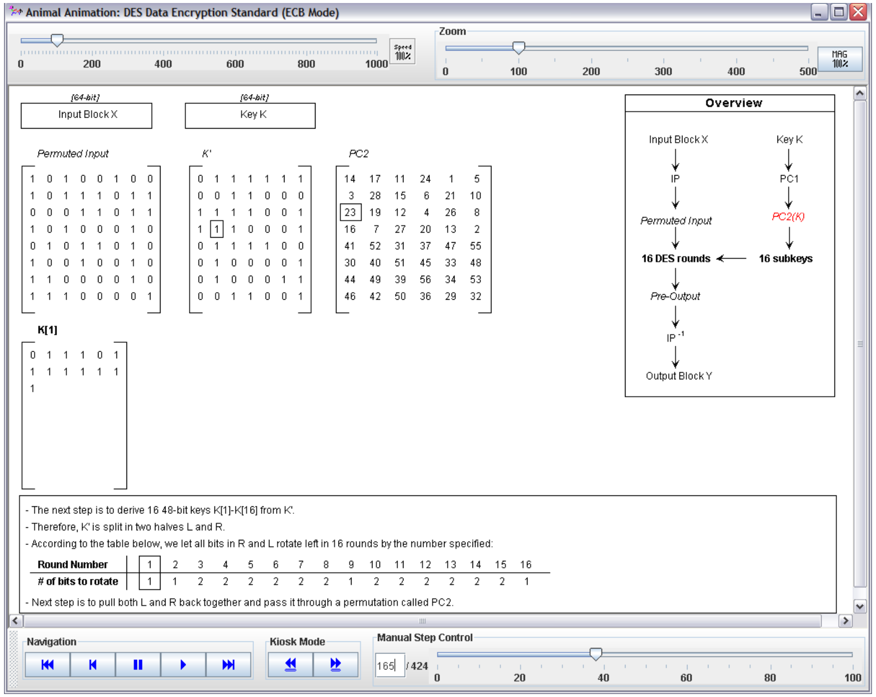
Animation controls (next, forward, pause, etc.)

Direct selection of an animation step

Examples (14)

Visualisation of symmetric encryption methods using ANIMAL (2)

Visualization of DES encryption



Permuted Input

1	0	1	0	0	1	0	0
1	0	1	1	1	0	1	1
0	0	0	1	1	0	1	1
1	0	1	0	0	0	1	0
0	1	0	1	1	0	1	0
1	0	0	1	0	0	1	0
1	1	0	0	0	0	1	0
1	1	1	0	0	0	0	1

K'

0	1	1	1	1	1	1
0	0	1	1	0	0	0
1	1	1	1	0	0	1
1	1	1	0	0	0	1
0	1	1	1	1	0	0
0	1	0	0	0	0	1
0	1	0	0	0	1	1
0	0	1	1	0	0	1

PC2

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

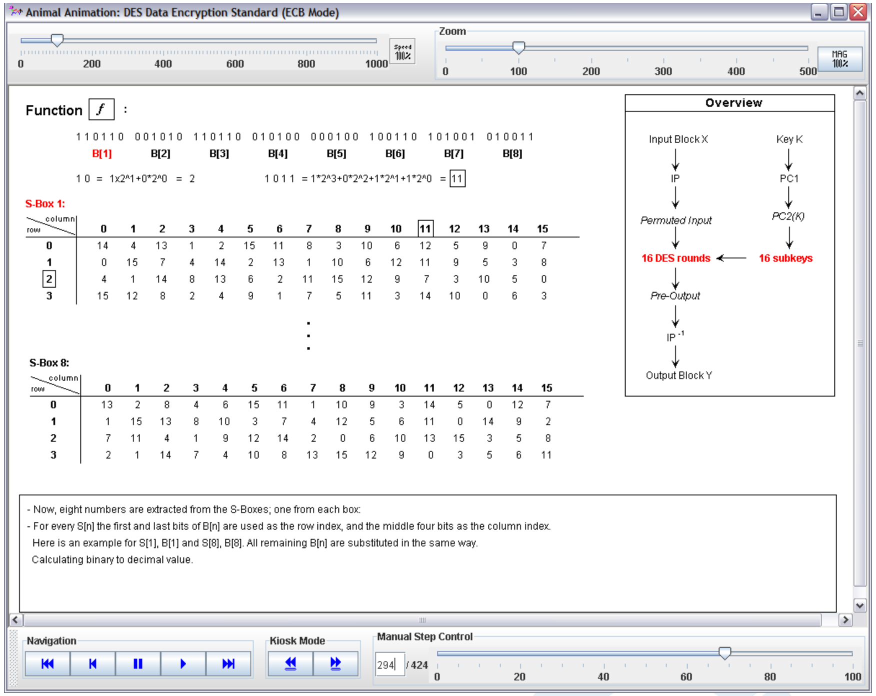
K[1]

0	1	1	1	0	1
1	1	1	1	1	1
1					

- The next step is to derive 16 48-bit keys K[1]-K[16] from K'.
- Therefore, K' is split in two halves L and R.
- According to the table below, we let all bits in R and L rotate left in 16 rounds by the number specified:

Round Number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
# of bits to rotate	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

- Next step is to pull both L and R back together and pass it through a permutation called PC2.



Function f :

110110	001010	110110	010100	000100	100110	101001	010011
B[1]	B[2]	B[3]	B[4]	B[5]	B[6]	B[7]	B[8]

1 0 = 1x2^1+0*2^0 = 2 1 0 1 1 = 1*2^3+0*2^2+1*2^1+1*2^0 = 11

S-Box 1:

row \ column	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	3

⋮

S-Box 8:

row \ column	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

- Now, eight numbers are extracted from the S-Boxes; one from each box:
- For every S[n] the first and last bits of B[n] are used as the row index, and the middle four bits as the column index.
Here is an example for S[1], B[1] and S[8], B[8]. All remaining B[n] are substituted in the same way.
Calculating binary to decimal value.

After the permutation of the input block using the initialisation vector IV the key K is being 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.

Examples (15)

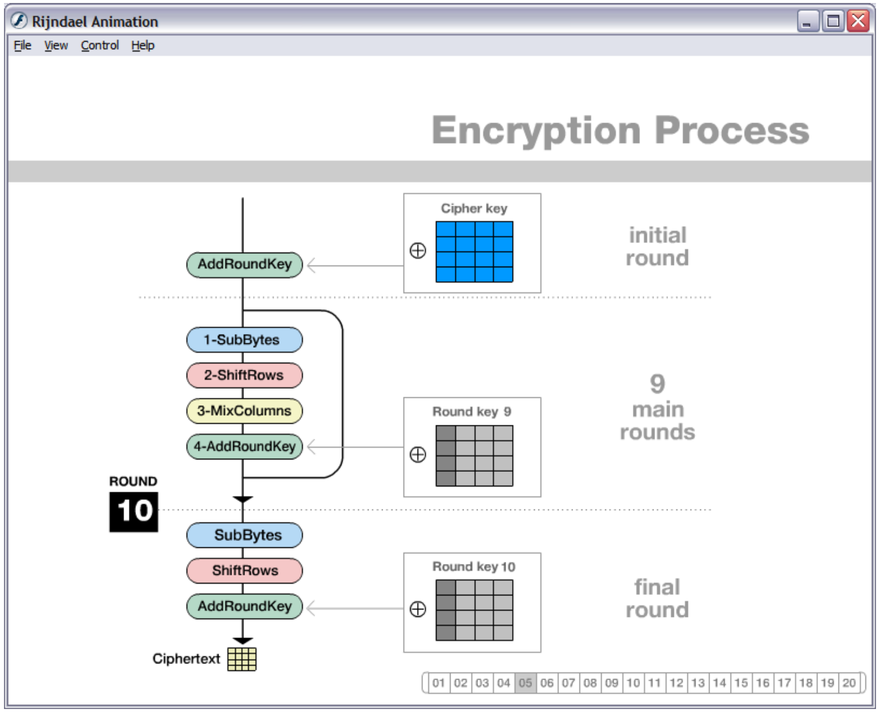
Visualisation of AES (Rijndael cipher)

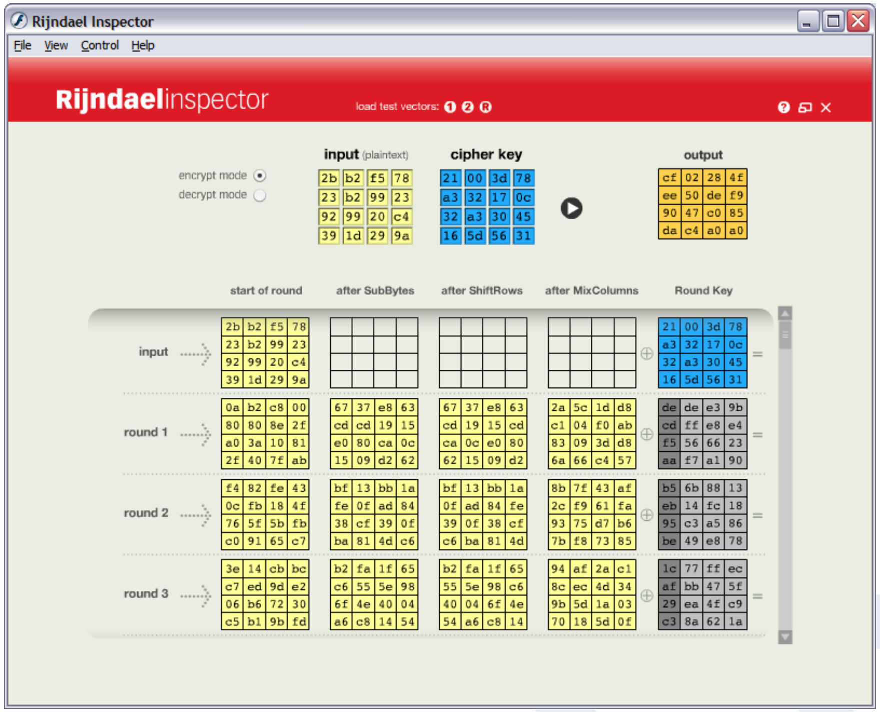
Rijndael Animation (the Rijndael cipher was the winner of the AES submission)

- Visualisation shows animation of the round-based encryption process (using fixed data)

Rijndael Inspector

- Encryption process for testing (using your own data)





● encrypt mode
○ decrypt mode

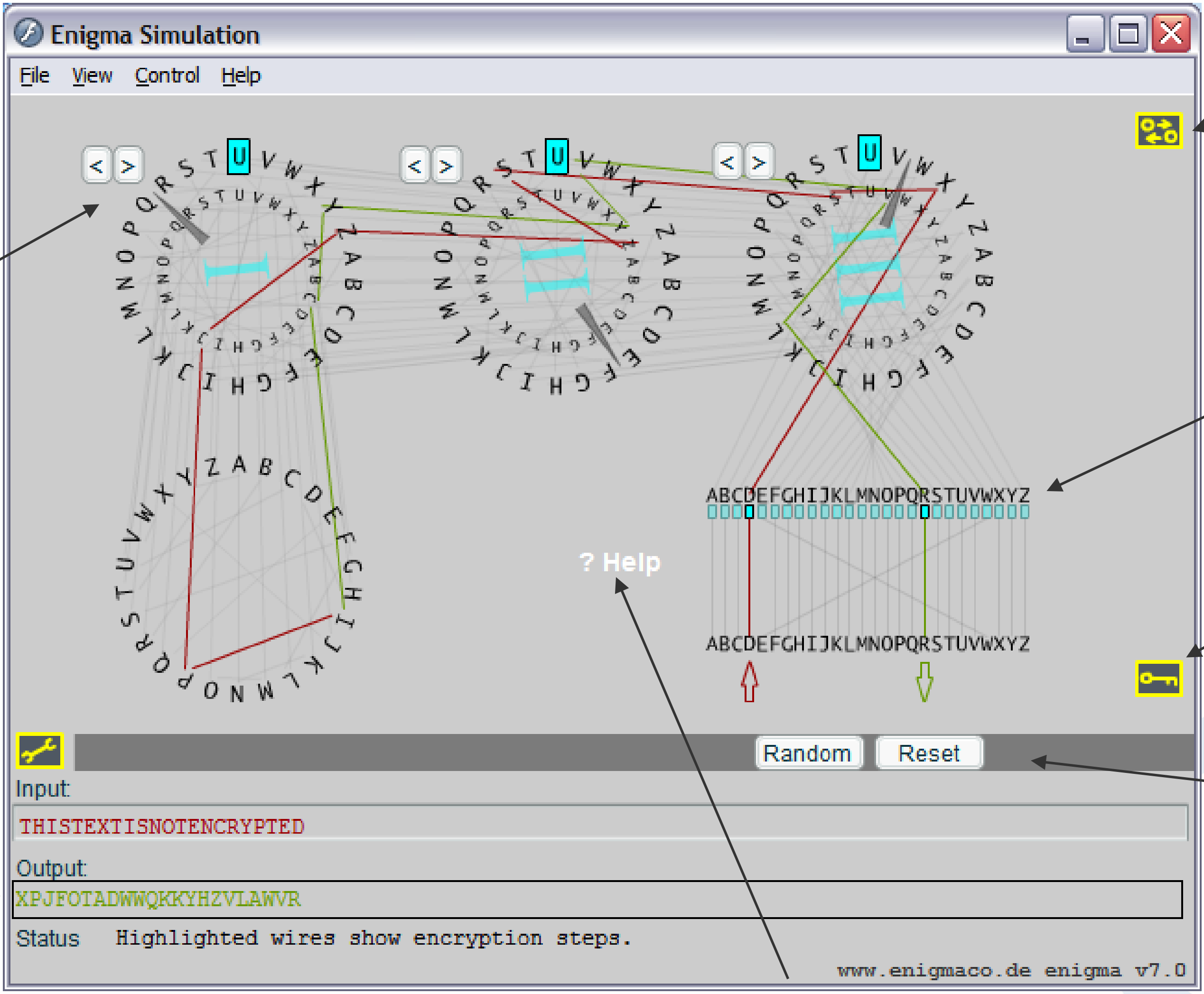
input (plaintext)	cipher key	output
2b b2 f5 78 23 b2 99 23 92 99 20 c4 39 1d 29 9a	21 00 3d 78 a3 32 17 0c 32 a3 30 45 16 5d 56 31	cf 02 28 4f ee 50 de f9 90 47 c0 85 da c4 a0 a0

	start of round	after SubBytes	after ShiftRows	after MixColumns		Round Key	
input	2b b2 f5 78 23 b2 99 23 92 99 20 c4 39 1d 29 9a				⊕	21 00 3d 78 a3 32 17 0c 32 a3 30 45 16 5d 56 31	=
round 1	0a b2 c8 00 80 80 8e 2f a0 3a 10 81 2f 40 7f ab	67 37 e8 63 cd cd 19 15 e0 80 ca 0c 15 09 d2 62	67 37 e8 63 cd 19 15 cd ca 0c e0 80 62 15 09 d2	2a 5c 1d d8 c1 04 f0 ab 83 09 3d d8 6a 66 c4 57	⊕	de de e3 9b cd ff e8 e4 f5 56 66 23 aa f7 a1 90	=
round 2	f4 82 fe 43 0c fb 18 4f 76 5f 5b fb c0 91 65 c7	bf 13 bb 1a fe 0f ad 84 38 cf 39 0f ba 81 4d c6	bf 13 bb 1a 0f ad 84 fe 39 0f 38 cf c6 ba 81 4d	8b 7f 43 af 2c f9 61 fa 93 75 d7 b6 7b f8 73 85	⊕	b5 6b 88 13 eb 14 fc 18 95 c3 a5 86 be 49 e8 78	=
round 3	3e 14 cb bc c7 ed 9d e2 06 b6 72 30 c5 b1 9b fd	b2 fa 1f 65 c6 55 5e 98 6f 4e 40 04 a6 c8 14 54	b2 fa 1f 65 55 5e 98 c6 40 04 6f 4e 54 a6 c8 14	94 af 2a c1 8c ec 4d 34 9b 5d 1a 03 70 18 5d 0f	⊕	1c 77 ff ec af bb 47 5f 29 ea 4f c9 c3 8a 62 1a	=

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

Examples (16)

Visualisation of the Enigma encryption



Input:
THISTEXTISNOTENCRYPTED

Output:
XPJFOTADWWQKKYHZVLAWVR

Status Highlighted wires show encryption steps.

www.enigmaco.de enigma v7.0

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)

Generation of a message authentication code (MAC)

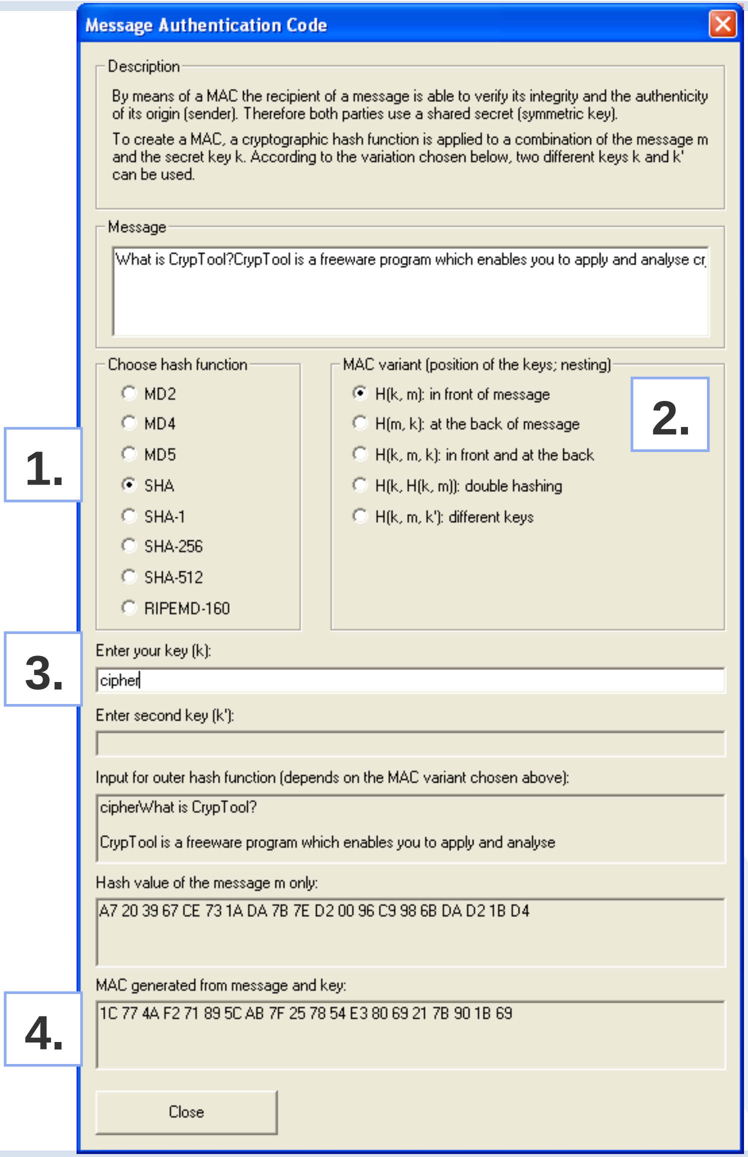
Message Authentication Code (MAC)

- Ensures integrity of a message
- Authentication of the message
- Basis: a common key

Generation of a MAC in CrypTool

1. Choose a hash function
2. Select MAC variant
3. Enter a key (depending on MAC variant also two keys)
4. Generation of the MAC (automatic)

Menu: „Indiv. Procedures” \ „Hash“ \ „Generation of MACs”



Message Authentication Code

Description

By means of a MAC the recipient of a message is able to verify its integrity and the authenticity of its origin (sender). Therefore both parties use a shared secret (symmetric key).

To create a MAC, a cryptographic hash function is applied to a combination of the message m and the secret key k. According to the variation chosen below, two different keys k and k' can be used.

Message

What is CrypTool?CrypTool is a freeware program which enables you to apply and analyse cr

Choose hash function

- ○ MD2
- ○ MD4
- ○ MD5
- ● SHA
- ○ SHA-1
- ○ SHA-256
- ○ SHA-512
- ○ RIPEMD-160

MAC variant (position of the keys; nesting)

- ● H(k, m): in front of message
- ○ H(m, k): at the back of message
- ○ H(k, m, k): in front and at the back
- ○ H(k, H(k, m)): double hashing
- ○ H(k, m, k'): different keys

Enter your key (k):

cipher

Enter second key (k'):

Input for outer hash function (depends on the MAC variant chosen above):

cipherWhat is CrypTool?

CrypTool is a freeware program which enables you to apply and analyse

Hash value of the message m only:

```
A7 20 39 67 CE 73 1A DA 7B 7E D2 00 96 C9 98 6B DA D2 1B D4
```

MAC generated from message and key:

```
1C 77 4A F2 71 89 5C AB 7F 25 78 54 E3 80 69 21 7B 90 1B 69
```

Examples (18)

Hash demonstration

Sensitivity of hash functions to plaintext modifications

1. Select a hash function

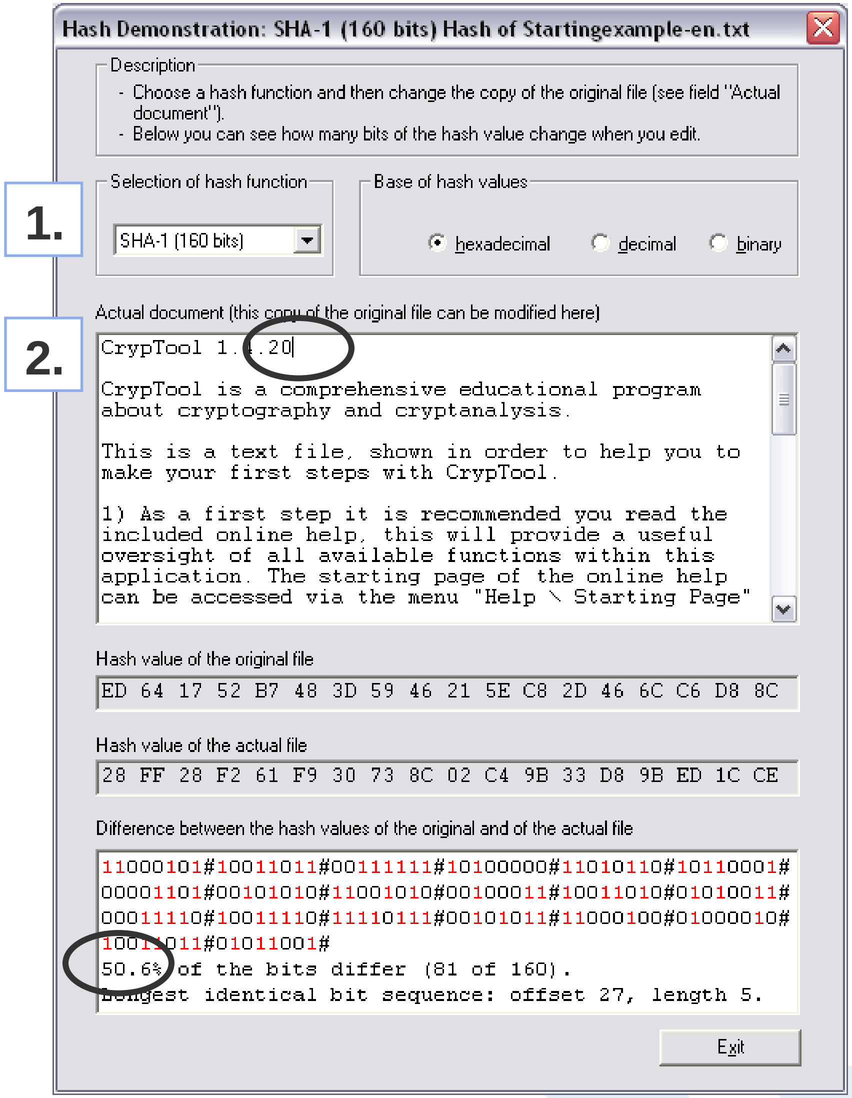
2. Modification of characters in plaintext

Example:

Entering a blank after „CrypTool“ in the example text results in a 50.6 % change of the bits of the generated hash value.

A good hash function should react sensitive to even the smallest change within the plaintext – „Avalanche effect“ (small change, big impact).

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



Hash Demonstration: SHA-1 (160 bits) Hash of Startingexample-en.txt

Description
- Choose a hash function and then change the copy of the original file (see field "Actual document").
- Below you can see how many bits of the hash value change when you edit.

Selection of hash function: SHA-1 (160 bits)

Base of hash values: ● hexadecimal ○ decimal ○ binary

Actual document (this copy of the original file can be modified here)

```
CrypTool 1.4.20

CrypTool is a comprehensive educational program
about cryptography and cryptanalysis.

This is a text file, shown in order to help you to
make your first steps with CrypTool.

1) As a first step it is recommended you read the
included online help, this will provide a useful
oversight of all available functions within this
application. The starting page of the online help
can be accessed via the menu "Help \ Starting Page"
```

Hash value of the original file

```
ED 64 17 52 B7 48 3D 59 46 21 5E C8 2D 46 6C C6 D8 8C
```

Hash value of the actual file

```
28 FF 28 F2 61 F9 30 73 8C 02 C4 9B 33 D8 9B ED 1C CE
```

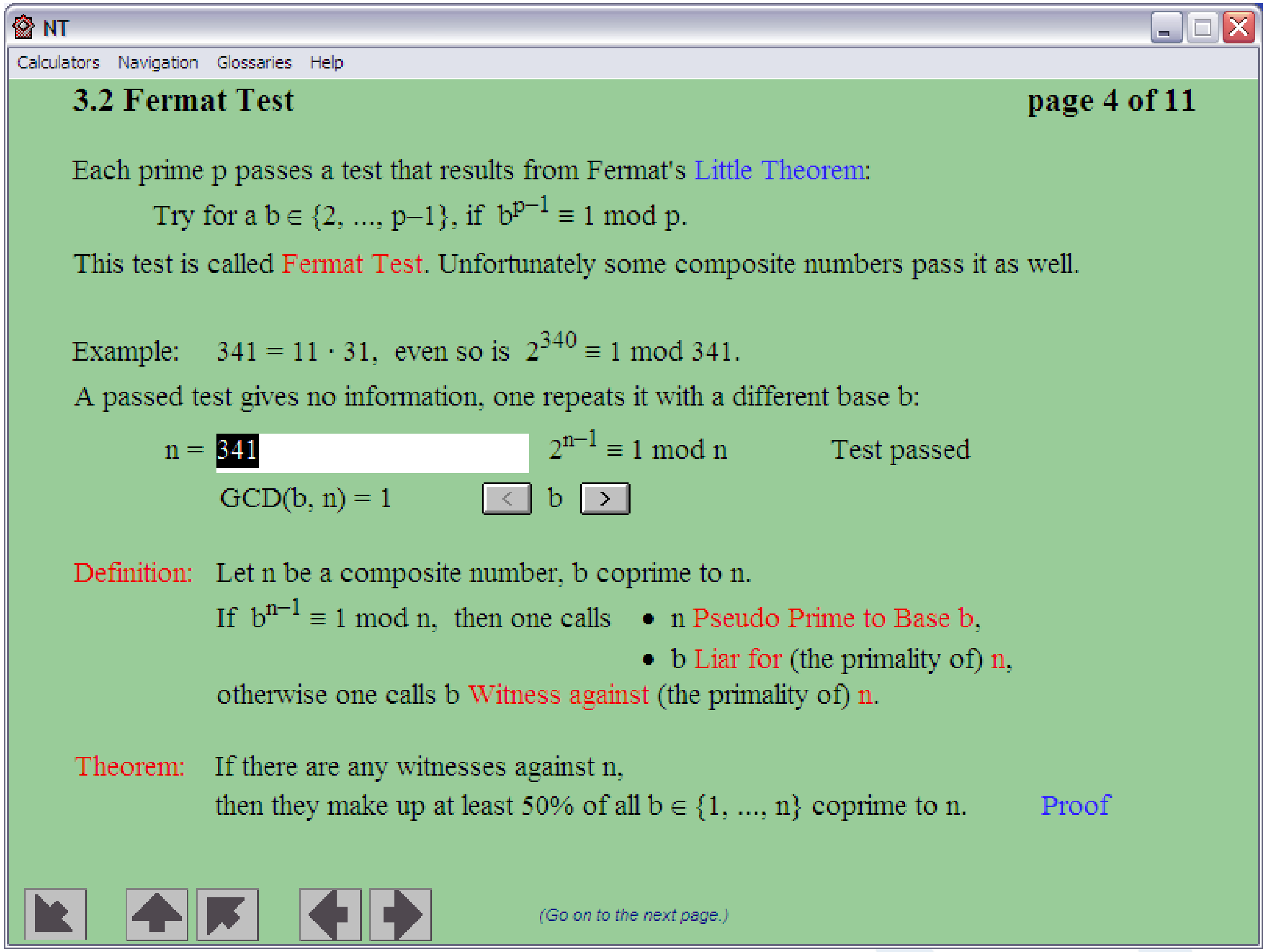
Difference between the hash values of the original and of the actual file

```
11000101#10011011#00111111#10100000#11010110#10110001#
00001101#00101010#11001010#00100011#10011010#01010011#
00011110#10011110#11110111#00101011#11000100#01000010#
10011011#01011001#
50.6% of the bits differ (81 of 160).
Longest identical bit sequence: offset 27, length 5.
```

Examples (19)

Learning tool for number theory

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



3.2 Fermat Test **page 4 of 11**

Each prime p passes a test that results from Fermat's Little Theorem:

Try for a $b \in \{2, \ldots, p-1\}$, if $b^{p-1} \equiv 1 \bmod p$.

This test is called Fermat Test. Unfortunately some composite numbers pass it as well.

Example: $341 = 11 \cdot 31$, even so is $2^{340} \equiv 1 \bmod 341$.

A passed test gives no information, one repeats it with a different base b:

n = 341 $2^{n-1} \equiv 1 \bmod n$ Test passed

GCD(b, n) = 1 < b >

Definition: Let n be a composite number, b coprime to n.

If $b^{n-1} \equiv 1 \bmod n$, then one calls
- n Pseudo Prime to Base b,
- b Liar for (the primality of) n,

otherwise one calls b Witness against (the primality of) n.

Theorem: If there are any witnesses against n, then they make up at least 50% of all $b \in \{1, \ldots, n\}$ coprime to n. Proof

(Go on to the next page.)

Menu: „Indiv. Procedures" \ „Number Theory - Interactive" \ „Learning tool for number theory"

Examples (20)

Point addition on elliptic curves

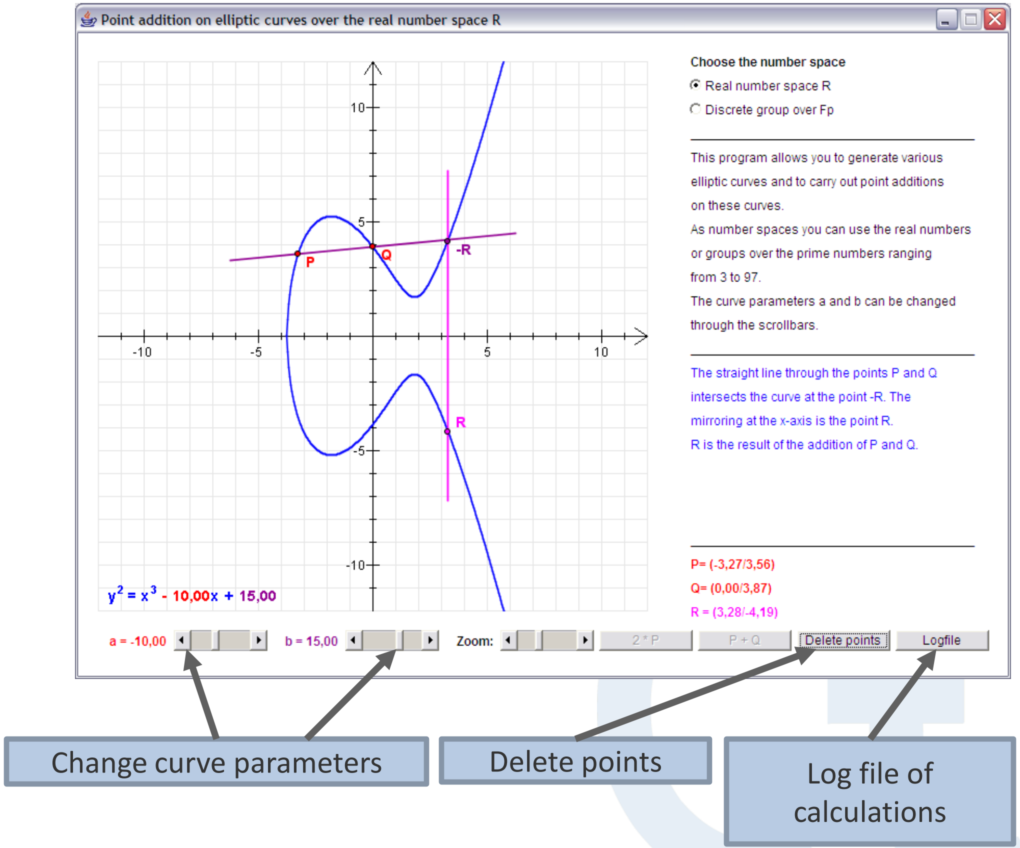
- Visualisation of point addition on elliptic curves
- Foundation of elliptic curve cryptography (ECC)

Example 1

- Mark point P on the curve
- Mark point Q on the curve
- Press button „P+Q“: The straight line through P and Q intersects the curve in point -R
- Mirroring on the X-axis results in point R

Example 2

- Mark point P on the curve
- Press button „2*P“: The tangent of point P intersects the curve in point -R
- Mirroring on the X-axis results in point R



Point addition on elliptic curves over the real number space R

Choose the number space

● Real number space R

○ Discrete group over Fp

This program allows you to generate various elliptic curves and to carry out point additions on these curves.

As number spaces you can use the real numbers or groups over the prime numbers ranging from 3 to 97.

The curve parameters a and b can be changed through the scrollbars.

The straight line through the points P and Q intersects the curve at the point -R. The mirroring at the x-axis is the point R.

R is the result of the addition of P and Q.

$y^2 = x^3 - 10{,}00x + 15{,}00$

P= (-3,27/3,56)

Q= (0,00/3,87)

R = (3,28/-4,19)

a = -10,00

b = 15,00

Change curve parameters

Delete points

Log file of calculations

Menu: „Indiv. Procedures“ \ „Number Theory - Interactive“ \ „Point Addition on Elliptic Curves“

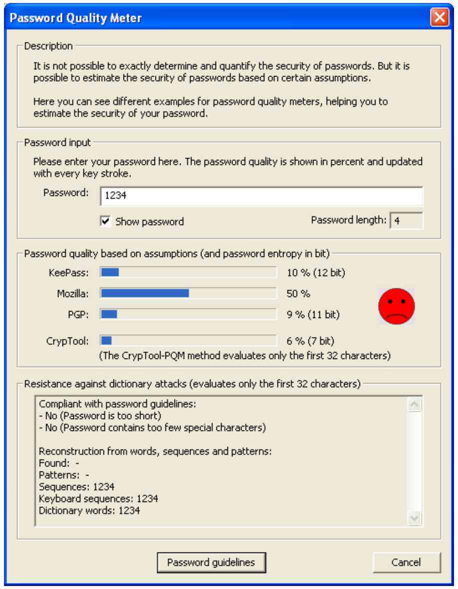
Examples (21)

Password Quality Meter (PQM) 1

Functions

- Measuring the quality of passwords
- Compare with PQMs in other applications: KeePass, Mozilla und PGP
- Experimental measuring through CrypTool algorithm
- Example: Input of a password (while showing the password)

Password: ***1234***



Description

It is not possible to exactly determine and quantify the security of passwords. But it is possible to estimate the security of passwords based on certain assumptions.

Here you can see different examples for password quality meters, helping you to estimate the security of your password.

Password input

Please enter your password here. The password quality is shown in percent and updated with every key stroke.

Password: 1234

☑ Show password   Password length: 4

Password quality based on assumptions (and password entropy in bit)

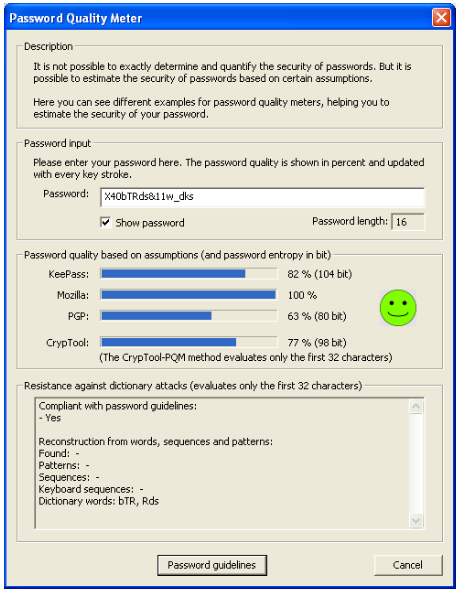
KeePass: 10 % (12 bit)
Mozilla: 50 %
PGP: 9 % (11 bit)
CrypTool: 6 % (7 bit)
(The CrypTool-PQM method evaluates only the first 32 characters)

Resistance against dictionary attacks (evaluates only the first 32 characters)

```
Compliant with password guidelines:
- No (Password is too short)
- No (Password contains too few special characters)

Reconstruction from words, sequences and patterns:
Found:  -
Patterns:  -
Sequences: 1234
Keyboard sequences: 1234
Dictionary words: 1234
```

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



Description

It is not possible to exactly determine and quantify the security of passwords. But it is possible to estimate the security of passwords based on certain assumptions.

Here you can see different examples for password quality meters, helping you to estimate the security of your password.

Password input

Please enter your password here. The password quality is shown in percent and updated with every key stroke.

Password: X40bTRds&11w_dks

☑ Show password   Password length: 16

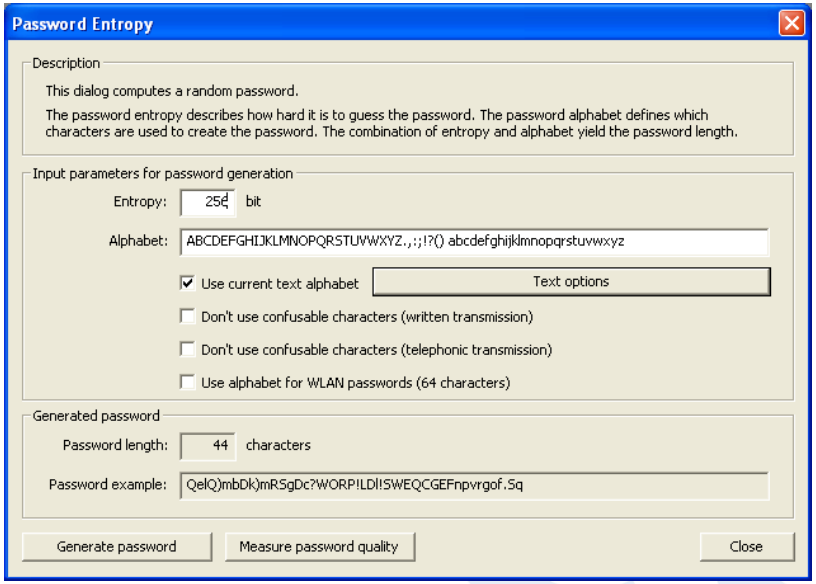
Password quality based on assumptions (and password entropy in bit)

KeePass: 82 % (104 bit)
Mozilla: 100 %
PGP: 63 % (80 bit)
CrypTool: 77 % (98 bit)
(The CrypTool-PQM method evaluates only the first 32 characters)

Resistance against dictionary attacks (evaluates only the first 32 characters)

```
Compliant with password guidelines:
- Yes

Reconstruction from words, sequences and patterns:
Found:  -
Patterns:  -
Sequences:  -
Keyboard sequences:  -
Dictionary words: bTR, Rds
```



Description

This dialog computes a random password.

The password entropy describes how hard it is to guess the password. The password alphabet defines which characters are used to create the password. The combination of entropy and alphabet yield the password length.

Input parameters for password generation

Entropy: 256 bit

Alphabet: ABCDEFGHIJKLMNOPQRSTUVWXYZ.,:;!?() abcdefghijklmnopqrstuvwxyz

☑ Use current text alphabet

□ Don't use confusable characters (written transmission)

□ Don't use confusable characters (telephonic transmission)

□ Use alphabet for WLAN passwords (64 characters)

Generated password

Password length: 44 characters

Password example: QelQ)mbDk)mRSgDc?WORP!LDl!SWEQCGEFnpvrgof.Sq

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

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

Examples (21)

Password Quality Meter (PQM) 2

Findings of 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** as indicator of the randomness of password characters of 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 choosen so that attack 1. and 2. do not compromise the password. Regarding brute-force attacks the length of the password (at least 8 characters) as well as the used character sets are important.

Examples (22)

Brute-force analysis 1

Brute-force analysis

Optimised brute-force analysis under the assumption that the key is partly 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)

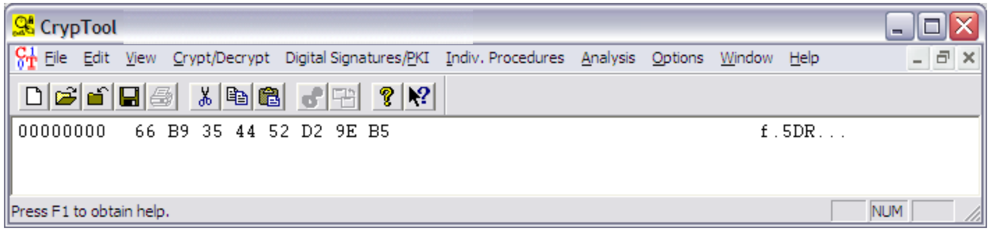
Key (Hex)	Encrypted text (Hex)
68ac78dd40bbefd*	66b9354452d29eb5
0123456789ab****	1f0dd05d8ed51583
98765432106*****	bcf9ebd1979ead6a
0000000000******	8cf42d40e004a1d4
000000000000****	0ed33fed7f46c585
abacadaba*******	d6d8641bc4fb2478
dddddddddd******	a2e66d852e175f5c

Examples (22)

Brute-force analysis 2

1. Input of encrypted text
2. Use brute-force analysis
3. Input partly known key
4. Start brute-force analysis

Use „View“ \ „Show as HexDump“



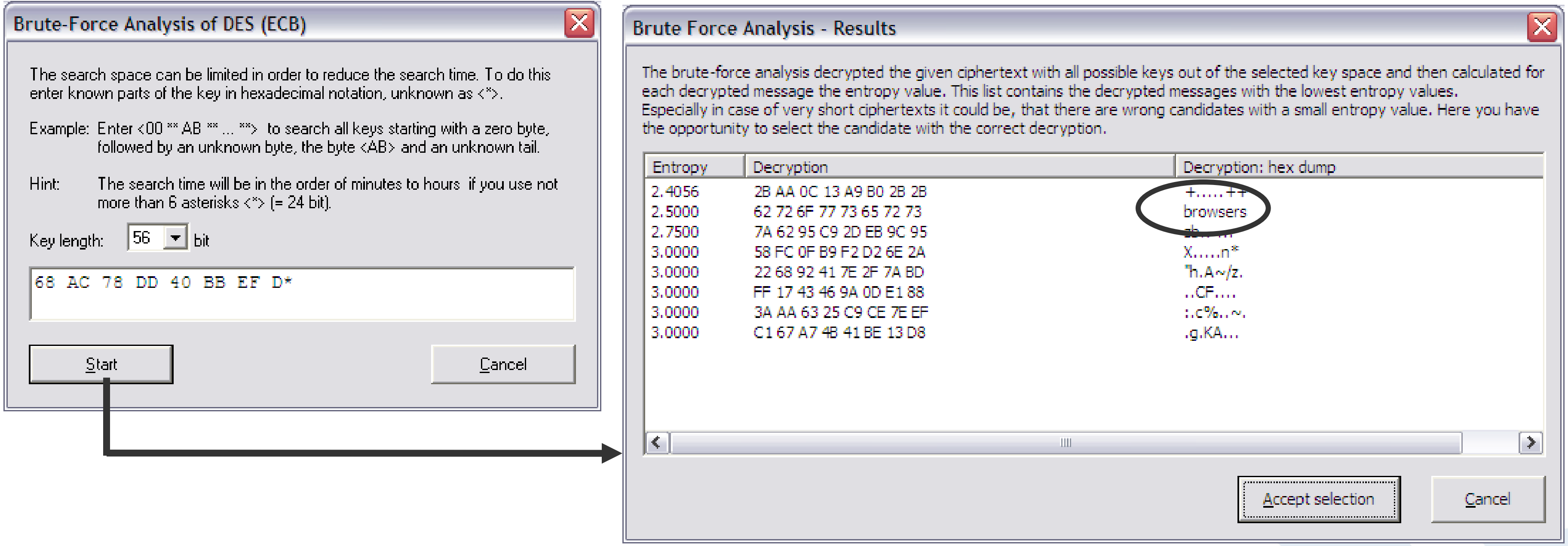
CrypTool
File Edit View Crypt/Decrypt Digital Signatures/PKI Indiv. Procedures Analysis Options Window Help

```
00000000    66 B9 35 44 52 D2 9E B5                            f.5DR...
```

Press F1 to obtain help.

5. Analysis of the results: Low entropy as evidence of a possible decryption. However, because a very short plaintext has been used in this example, the correct result does not have the lowest entropy.

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



Brute-Force Analysis of DES (ECB)

The search space can be limited in order to reduce the search time. To do this enter known parts of the key in hexadecimal notation, unknown as <*>.

Example: Enter <00 ** AB ** ... **> to search all keys starting with a zero byte, followed by an unknown byte, the byte <AB> and an unknown tail.

Hint: The search time will be in the order of minutes to hours if you use not more than 6 asterisks <*> (= 24 bit).

Key length: 56 bit

```
68 AC 78 DD 40 BB EF D*
```

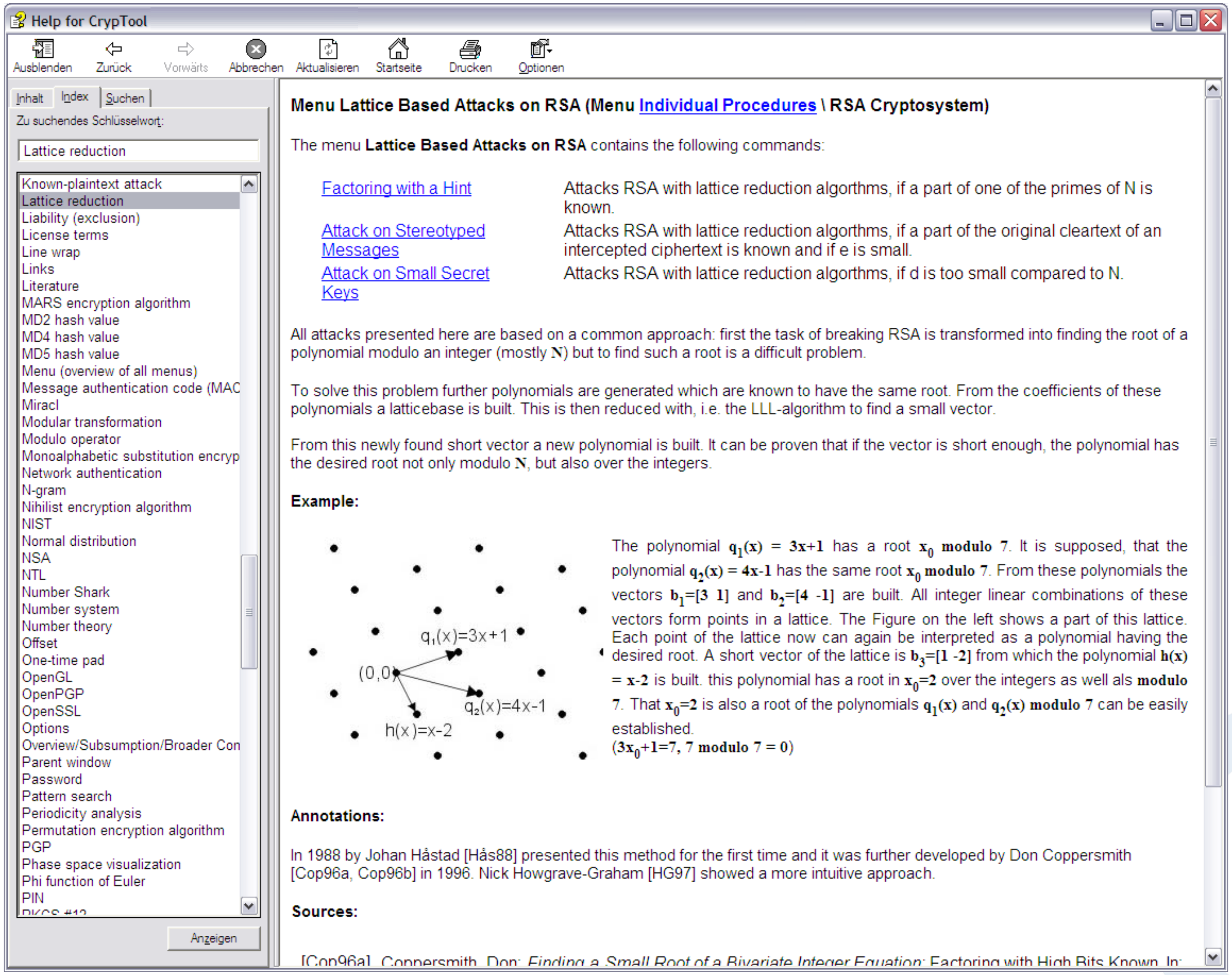
Brute Force Analysis - Results

The brute-force analysis decrypted the given ciphertext with all possible keys out of the selected key space and then calculated for each decrypted message the entropy value. This list contains the decrypted messages with the lowest entropy values. Especially in case of very short ciphertexts it could be, that there are wrong candidates with a small entropy value. Here you have the opportunity to select the candidate with the correct decryption.

Entropy	Decryption	Decryption: hex dump
2.4056	2B AA 0C 13 A9 B0 2B 2B	+.....++
2.5000	62 72 6F 77 73 65 72 73	browsers
2.7500	7A 62 95 C9 2D EB 9C 95	zb..-...
3.0000	58 FC 0F B9 F2 D2 6E 2A	X.....n*
3.0000	22 68 92 41 7E 2F 7A BD	"h.A~/z.
3.0000	FF 17 43 46 9A 0D E1 88	..CF....
3.0000	3A AA 63 25 C9 CE 7E EF	:.c%..~.
3.0000	C1 67 A7 4B 41 BE 13 D8	.g.KA...

Examples (23)

CrypTool online help 1



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 $\mathbf{q_1(x) = 3x+1}$ has a root $\mathbf{x_0}$ **modulo** 7. It is supposed, that the polynomial $\mathbf{q_2(x) = 4x-1}$ has the same root $\mathbf{x_0}$ **modulo** 7. From these polynomials the vectors $\mathbf{b_1=[3\ \ 1]}$ and $\mathbf{b_2=[4\ \ -1]}$ 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 $\mathbf{b_3=[1\ -2]}$ from which the polynomial $\mathbf{h(x)}$ = $\mathbf{x-2}$ is built. this polynomial has a root in $\mathbf{x_0=2}$ over the integers as well als **modulo** 7. That $\mathbf{x_0=2}$ is also a root of the polynomials $\mathbf{q_1(x)}$ and $\mathbf{q_2(x)}$ **modulo** 7 can be easily established.
$(\mathbf{3x_0+1=7,}$ 7 **modulo** 7 = 0)

Annotations:

In 1988 by Johan Håstad [Hås88] presented this method for the first time and it was further developed by Don Coppersmith [Cop96a, Cop96b] in 1996. Nick Howgrave-Graham [HG97] showed a more intuitive approach.

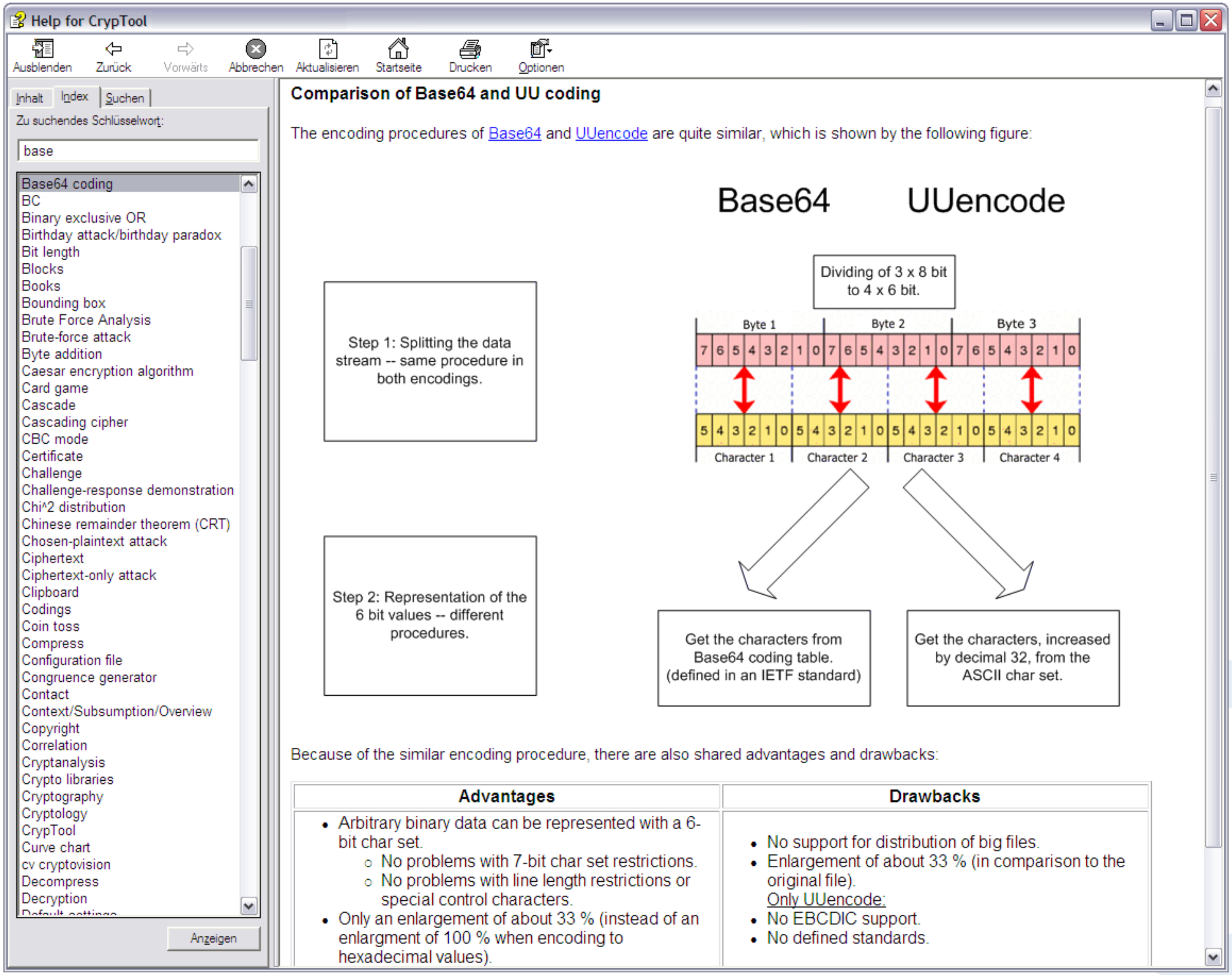
Sources:

[Cop96a] Coppersmith, Don: *Finding a Small Root of a Bivariate Integer Equation;* Factoring with High Bits Known. In:

Menu: „Help“ \ „Starting Page“

Examples (23)

CrypTool online help 2



Zu suchendes Schlüsselwort:

base

Base64 coding
BC
Binary exclusive OR
Birthday attack/birthday paradox
Bit length
Blocks
Books
Bounding box
Brute Force Analysis
Brute-force attack
Byte addition
Caesar encryption algorithm
Card game
Cascade
Cascading cipher
CBC mode
Certificate
Challenge
Challenge-response demonstration
Chi^2 distribution
Chinese remainder theorem (CRT)
Chosen-plaintext attack
Ciphertext
Ciphertext-only attack
Clipboard
Codings
Coin toss
Compress
Configuration file
Congruence generator
Contact
Context/Subsumption/Overview
Copyright
Correlation
Cryptanalysis
Crypto libraries
Cryptography
Cryptology
CrypTool
Curve chart
cv cryptovision
Decompress
Decryption
Default settings

Comparison of Base64 and UU coding

The encoding procedures of Base64 and UUencode are quite similar, which is shown by the following figure:

Base64 UUencode

Dividing of 3 x 8 bit to 4 x 6 bit.

Step 1: Splitting the data stream -- same procedure in both encodings.

Byte 1	Byte 2	Byte 3
7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0

Character 1	Character 2	Character 3	Character 4
5 4 3 2 1 0	5 4 3 2 1 0	5 4 3 2 1 0	5 4 3 2 1 0

Step 2: Representation of the 6 bit values -- different procedures.

Get the characters from Base64 coding table. (defined in an IETF standard)

Get the characters, increased by decimal 32, from the ASCII char set.

Because of the similar encoding procedure, there are also shared advantages and drawbacks:

Advantages	Drawbacks
• Arbitrary binary data can be represented with a 6-bit char set. o No problems with 7-bit char set restrictions. o No problems with line length restrictions or special control characters. • Only an enlargement of about 33 % (instead of an enlargment of 100 % when encoding to hexadecimal values).	• No support for distribution of big files. • Enlargement of about 33 % (in comparison to the original file). Only UUencode: • No EBCDIC support. • No defined standards.

Content

I. CrypTool and Cryptology – Overview

II. CrypTool Features

III. Examples

IV. Project / Outlook / Contact

Future CrypTool Development (1)

CT = CrypTool
CT2 = CrypTool 2.0
JCT = JCrypTool

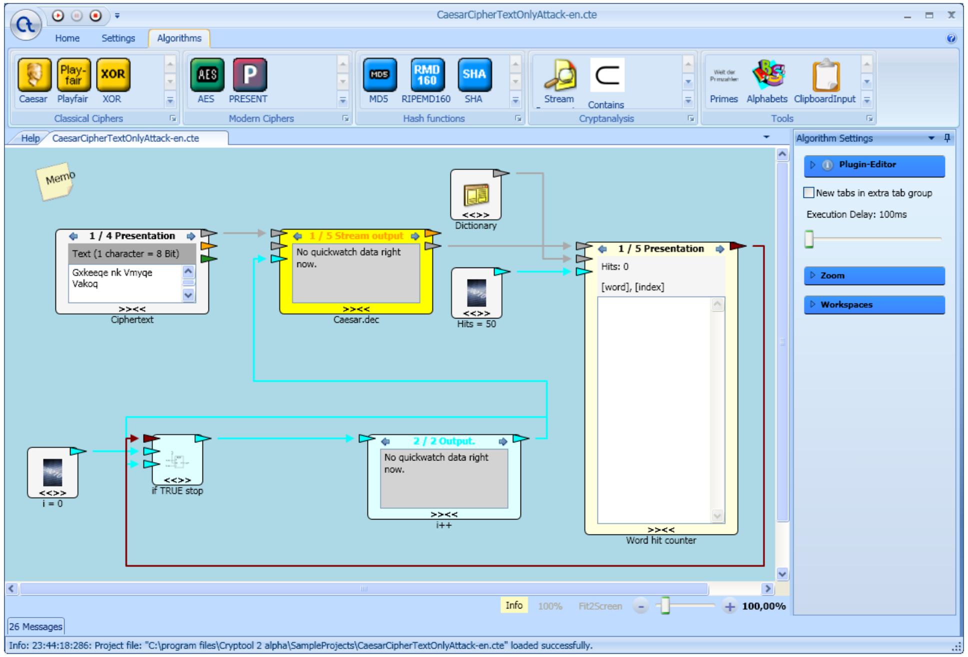
Planned after release 1.4.21 (see readme file)

CT 1.x	Mass pattern search
JCT	Visualisation of interoperability of S/MIME and OpenPGP formats
JCT	Tri-partite key agreements
JCT	Analysis of entropy
JCT	Statistical analysis of block ciphers
CT2	Comprehensive visualisation on the topic of prime numbers
CT2	Demonstration of Bleichenbacher‘s RSA signature forgery
CT2	Demonstration of virtual credit card numbers (approach against credit card abuse)
CT2	WEP encryption and WEP analysis
CT2	Graphical design oriented mode for beginners plus expert mode
CT2/JCT	Creation of a command line version for batch processing
CT2/JCT	Modern pure plugin architecture with loading of plugins
All	Further parameterization / Increasing the flexibility of present algorithms
CT2/JCT	Visualisation of the SSL protocol
CT2/JCT	Demonstration of visual cryptography
CT2/JCT	Integration of crypto library crypto++ from Wei Dai

Future CrypTool Development (2)

In Progress (see readme file)

1. JCT: Port and redesign of CrypTool in Java / SWT / Eclipse 3.4 / RPC
   - see: http://jcryptool.sourceforge.net
   - Milestone 2 available for users and developers from August 2008
2. CT2: Port and redesign of the C++ version with C# / WPF / VS2008 / .NET 3.5
   - direct successor of current releases: allows visual programming, …
   - Beta 1 available for users and developers from July 2008
3. C2L: Direct port of the C++ version to Linux with Qt4
   - see: http://www.cryptoolinux.net

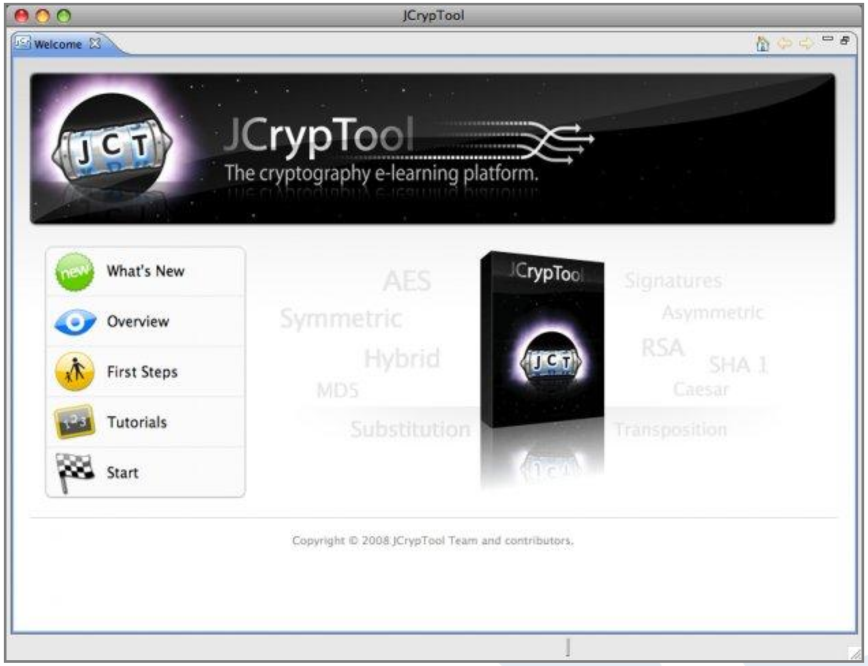


☐ New tabs in extra tab group

Execution Delay: 100ms

Info: 23:44:18:286: Project file: "C:\program files\Cryptool 2 alpha\SampleProjects\CaesarCipherTextOnlyAttack-en.cte" loaded successfully.

CrypTool 2 (CT2)



Copyright © 2008 JCrypTool Team and contributors.

JCrypTool (JCT)

CrypTool as a Framework

Proposal

- Re-use the comprehensive set of algorithms, included libraries and interface elements as foundation
- Free of charge training in Frankfurt, how to start with CrypTool development
- Advantage: Your own code does not „disappear“, but will be maintained

Current development environment: Microsoft Visual Studio C++ , Perl, Subversion Source-Code Management

- Until CrypTool 1.3.05: Visual C++ 6.0 only (was available with books for free)
- Until CrypTool 1.4.21: Visual C++ .net (= VC++ 7.1)(= Visual Studio 2003)
- Description for developers: see readme-source.txt
- Download: Sources and binaries of releases.
  To get sources of current betas, please see subversion repository.

Future development environments

- For versions after 1.4.2x:
  - CT2 – C# version: .NET with Visual Studio 2008 Express Edition (free), WPF und Perl
  - Java – Java version: Eclipse 3.4, RCP, SWT (free)
  - C2L – C++ version for Linux with Qt 4.x, GCC 4.x and Perl

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, test)
  - For the current C/C++ project
  - For the new projects
    - C# project: „CrypTool 2.0“
    - Java project: „JCrypTool“
  - Especially University faculties using CrypTool for educational purposes are invited to contribute to the further development of CrypTool.
- Significant contributions can be referenced by name (in help, readme, about dialog and on the CrypTool web site).
- Currently CrypTool is being downloaded more than 3000 times a month (with 1/3 for the English version).

CrypTool – Summary

- *THE* eLearning program for cryptology
- For 10 years a successful open source project
- More than 150.000 downloads
- International utilisation in schools, universities as well as companies and government agencies
- Extensive online help and documentation
- Available for free and multi-language support

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

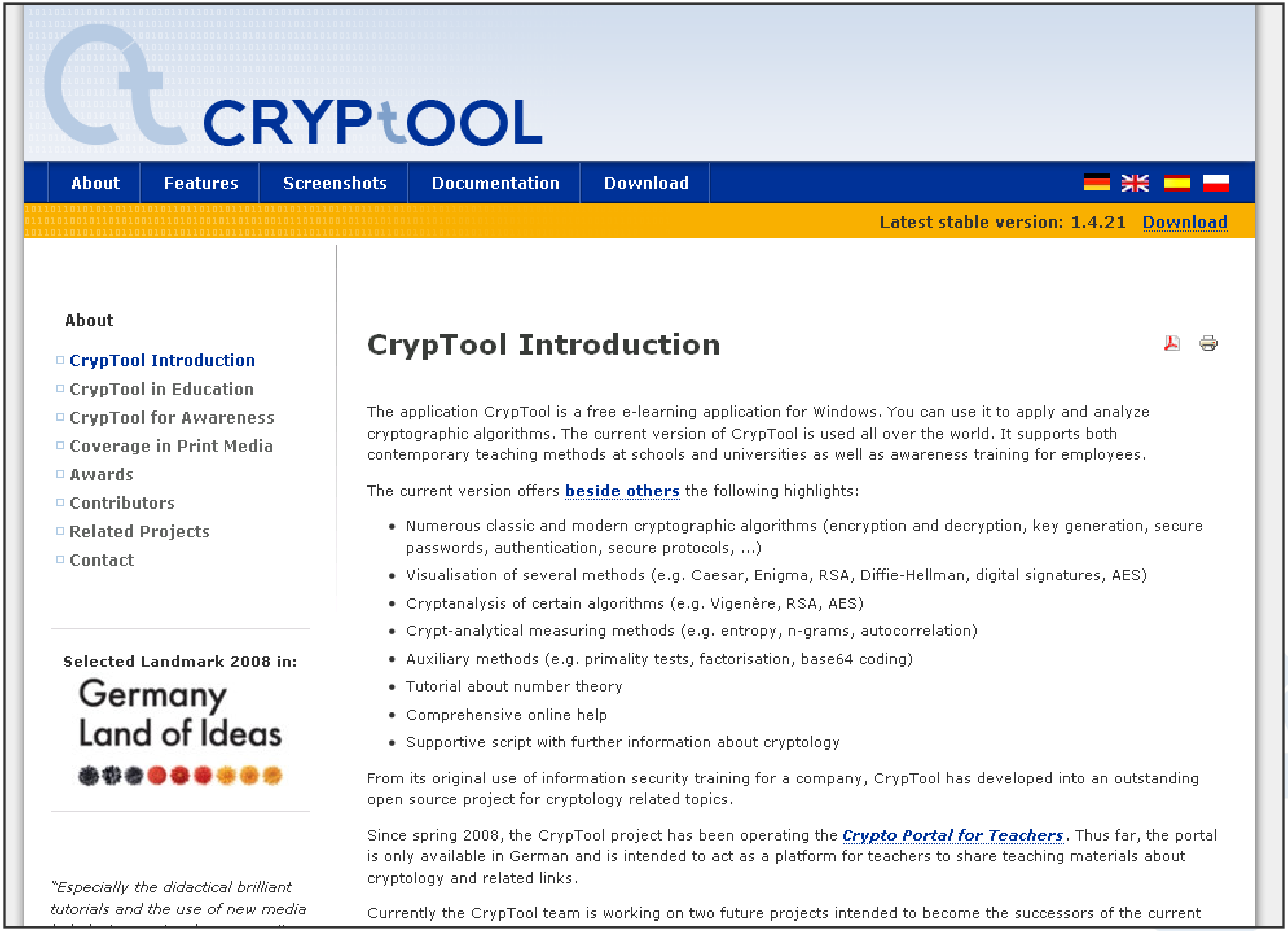
additional contacts: See readme within the CrypTool folder

Additional Literature

as introduction to cryptology

- Simon Singh, *“The Codebook”*, 1999, Doubleday
- Klaus Schmeh, “*Codeknacker gegen Codemacher. Die faszinierende Geschichte der Verschlüsselung*“, 2nd edition, 2007, W3L [German]
- Udo Ulfkotte, “Wirtschaftsspionage“, 2001, Goldmann [German]
- Johannes Buchmann, “*Introduction to Cryptography*“, 2nd edition, 2004, Springer
- Claudia Eckert, *“IT-Sicherheit”,* 3rd edition, 2004, Oldenbourg [German]
- A. Beutelspacher / J. Schwenk / K.-D. Wolfenstetter, *“Moderne Verfahren der Kryptographie”*, 5th edition, 2004, Vieweg [German]
- [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
- 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, …)
- 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”, 2005, Pearson, chapter 14 [German]
  - See Wikipedia (http://en.wikipedia.org/wiki/Risk_management)

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



CrypTool Introduction

The application CrypTool is a free e-learning application for Windows. You can use it to apply and analyze cryptographic algorithms. The current version of CrypTool is used all over the world. It supports both contemporary teaching methods at schools and universities as well as awareness training for employees.

The current version offers **beside others** the following highlights:

- Numerous classic and modern cryptographic algorithms (encryption and decryption, key generation, secure passwords, authentication, secure protocols, ...)
- Visualisation of several methods (e.g. Caesar, Enigma, RSA, Diffie-Hellman, digital signatures, AES)
- Cryptanalysis of certain algorithms (e.g. Vigenère, RSA, AES)
- Crypt-analytical measuring methods (e.g. entropy, n-grams, autocorrelation)
- Auxiliary methods (e.g. primality tests, factorisation, base64 coding)
- Tutorial about number theory
- Comprehensive online help
- Supportive script with further information about cryptology

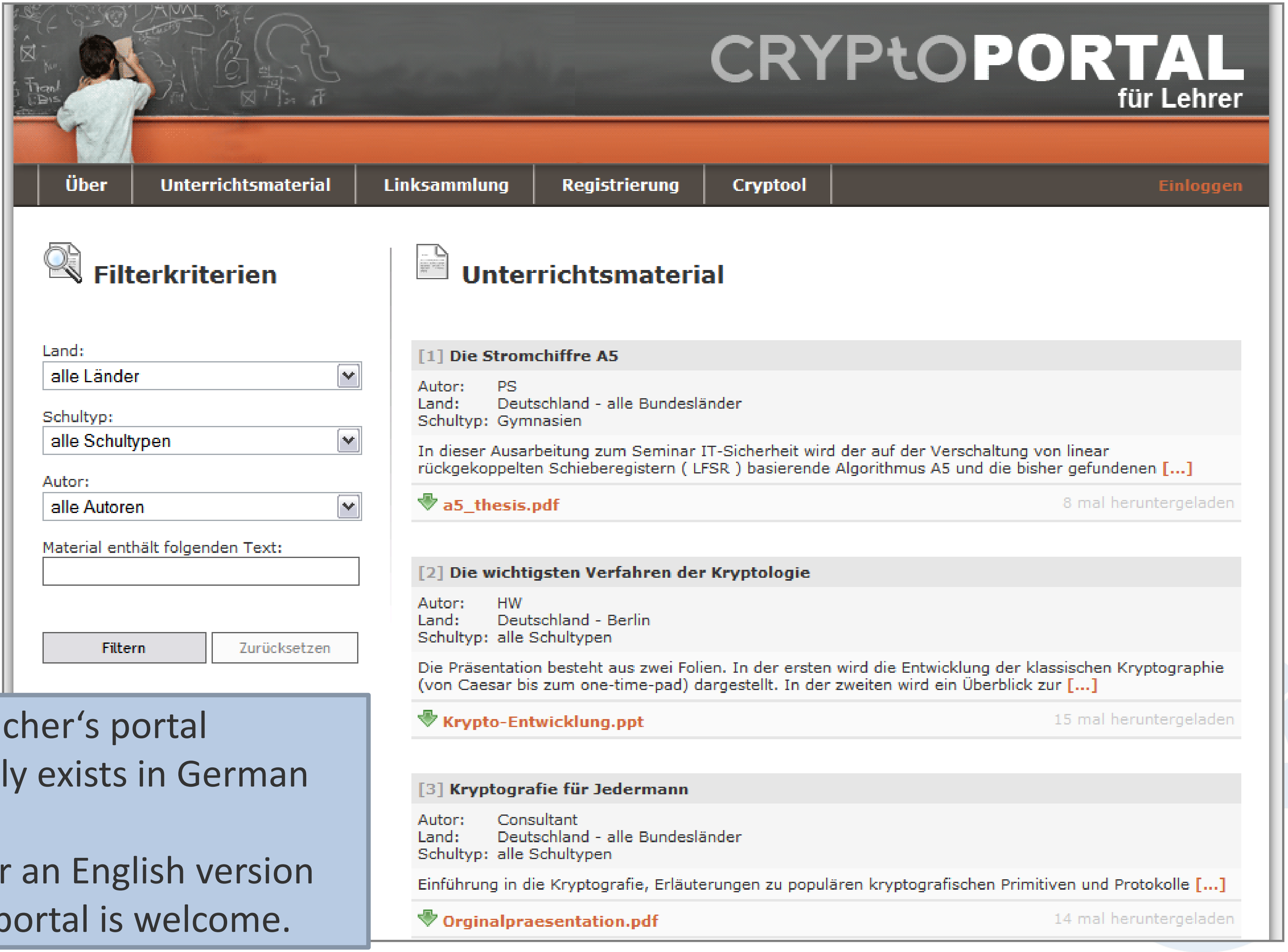
From its original use of information security training for a company, CrypTool has developed into an outstanding open source project for cryptology related topics.

Since spring 2008, the CrypTool project has been operating the ***Crypto Portal for Teachers***. Thus far, the portal is only available in German and is intended to act as a platform for teachers to share teaching materials about cryptology and related links.

Currently the CrypTool team is working on two future projects intended to become the successors of the current

"Especially the didactical brilliant tutorials and the use of new media

www.cryptoportal.org



Filterkriterien

Land: alle Länder

Schultyp: alle Schultypen

Autor: alle Autoren

Material enthält folgenden Text:

Unterrichtsmaterial

[1] **Die Stromchiffre A5**

Autor: PS
Land: Deutschland - alle Bundesländer
Schultyp: Gymnasien

In dieser Ausarbeitung zum Seminar IT-Sicherheit wird der auf der Verschaltung von linear rückgekoppelten Schieberegistern (LFSR) basierende Algorithmus A5 und die bisher gefundenen [...]

a5_thesis.pdf — 8 mal heruntergeladen

[2] **Die wichtigsten Verfahren der Kryptologie**

Autor: HW
Land: Deutschland - Berlin
Schultyp: alle Schultypen

Die Präsentation besteht aus zwei Folien. In der ersten wird die Entwicklung der klassischen Kryptographie (von Caesar bis zum one-time-pad) dargestellt. In der zweiten wird ein Überblick zur [...]

Krypto-Entwicklung.ppt — 15 mal heruntergeladen

[3] **Kryptografie für Jedermann**

Autor: Consultant
Land: Deutschland - alle Bundesländer
Schultyp: alle Schultypen

Einführung in die Kryptografie, Erläuterungen zu populären kryptografischen Primitiven und Protokolle [...]

Orginalpraesentation.pdf — 14 mal heruntergeladen

The teacher‘s portal currently exists in German only.
Help for an English version of this portal is welcome.