Cryptology with CrypTool

Version 1.4.10

Introduction to Cryptography and Cryptanalysis
Scope, Technology and Future of CrypTool

Bernhard Esslinger and CrypTool Team, 2007

Content (I)

I. CrypTool and Cryptography – Overview

II. CrypTool Features

III. Examples

Content (II)

Content

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)

2004 – Awards

TeleTrusT	(TTT Förderpreis 2004)
NRW	(IT Security Award NRW)
RSA Europe	(Finalist of European Information Security Award 2004)

2007 – CrypTool in German, English and Polish

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

Relevance of Cryptography

Typical scenarios for using cryptography in the daily life

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

Protection goals related to cryptography

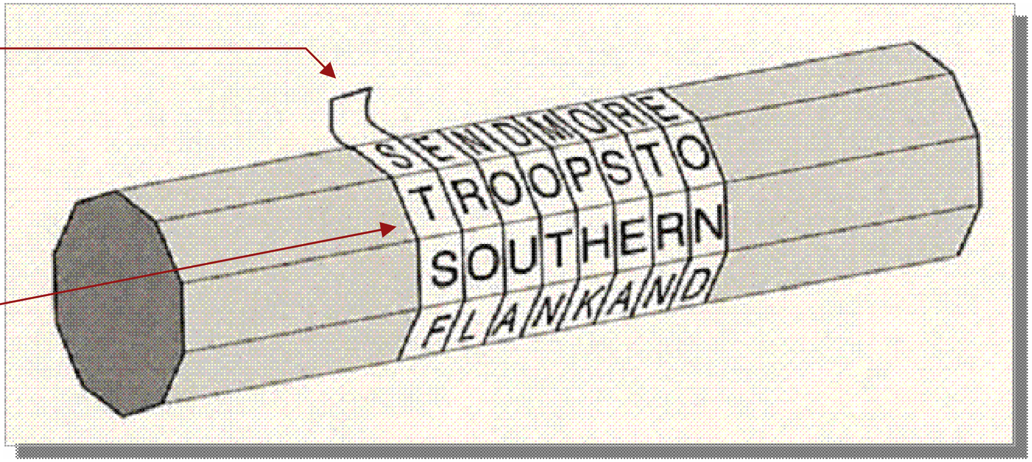
- **Confidentiality**
  - *Information can practically not be made available or disclosed to unauthorized individuals, entities or processes.*
- **Authentication**
  - *Authentication ensures that users are identified and those identities are appropriately verified.*
- **Integrity**
  - *Integrity ensures 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.*

Examples of Early Cryptography (I)

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)



Plaintext

„Send more troops to southern flank and …“

Examples of Early Cryptography (II)

Symmetric Caesar encryption

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

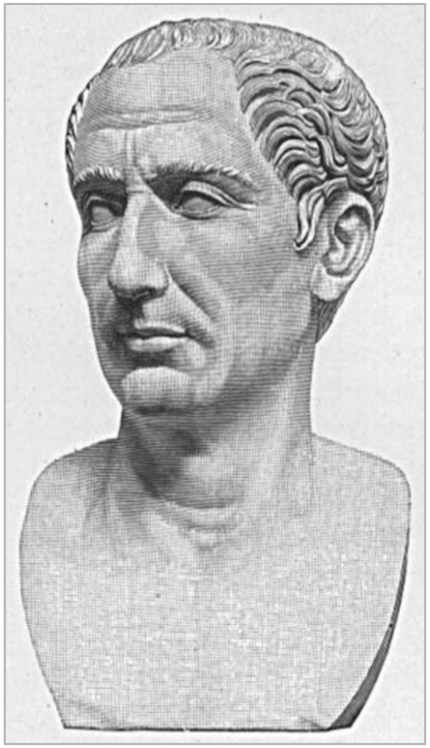
```
GALLIA  EST  OMNIS  DIVISA ...
```

Plaintext: `ABCDEFGHIJKLMNOPQRSTUVWXYZ`

Secret alphabet: `DEFGHIJKLMNOPQRSTUVWXYZABC`

```
JDOOLD  HVW  RPQLV  GLYLVD ...
```

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



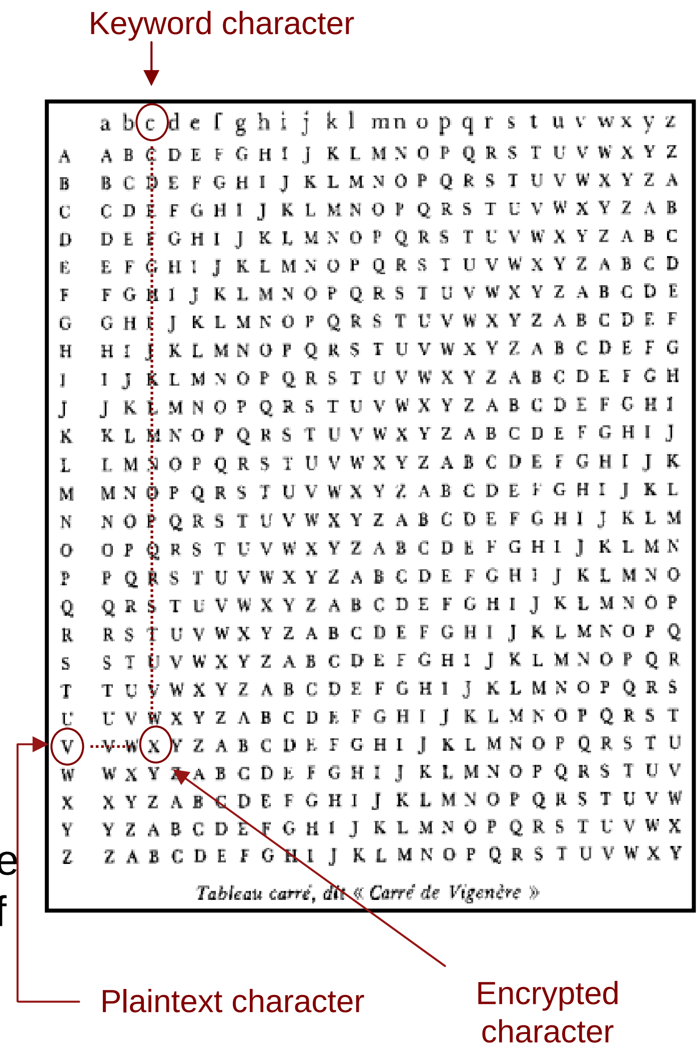
Presentation with CrypTool via the following menus:

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

Examples of Early Cryptography (II)

Symmetric Vigenère encryption

- **Vigenère Encryption** (Blaise de Vigenère, 1523-1596)
- Encryption with a key word using a key table
- Key word: **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.



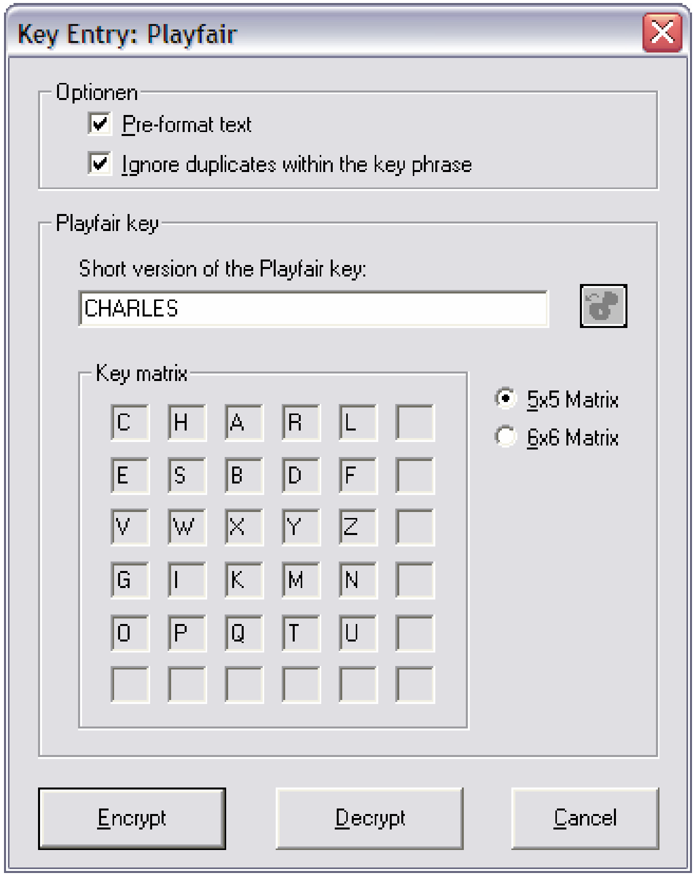
	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T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

Other symmetric encryption methods

- **Homophone substitution**
- **Playfair** (1854 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)



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:
  combinations of simple operations (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 shorten 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 (I)

- **Kerckhoffs principle** (1883)
  - Separation of algorithm (method) and key

    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 as possible
  - „Avalanche effect“ (small modification, big impact)
- **Trapdoor function** (one-way function)
  - Fast in one direction, very slow in the opposite direction
  - Only the secret key grants access to 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 $\mathbf{S_n = n(n-1) / 2}$ keys.

This means

n = 100 persons require

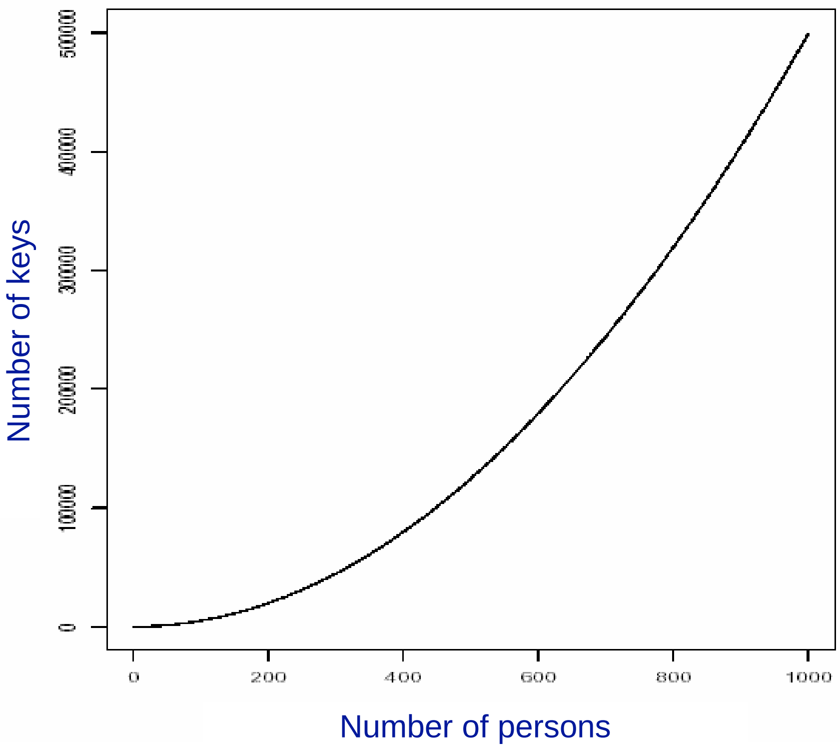
$\mathbf{S_{100} = 4.950}$ keys, and

n = 1.000 persons require

$\mathbf{S_{1000} = 499.500}$ keys.

⇨ factor 10 more persons, factor 100 more keys.

Number of required keys



Cryptography – Important Insights (II)

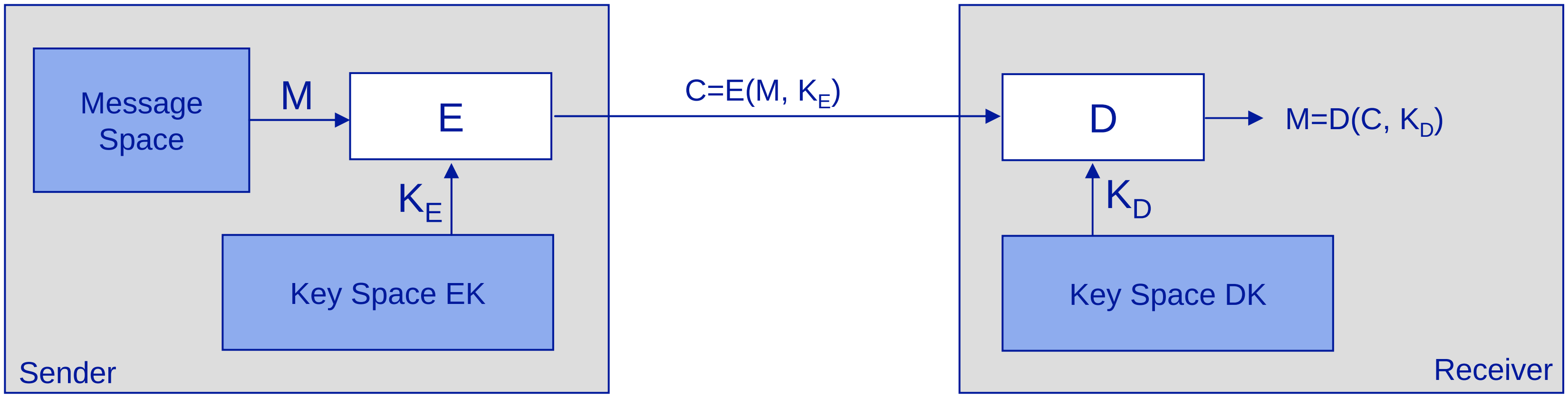
Solving the key distribution problem through asymmetric cryptography

- **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. DES)

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

public private/secret

Cryptography – Important Insights (III)

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
  - 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

- *Statistical 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

CrypTool Features

E-Learning

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 and Polish

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 (I)

Cryptography

Classical cryptography

- Caesar
- Vigenère
- Hill
- Homophone Substitution
- Playfair
- ADFGVX
- Addition
- XOR
- Vernam
- Permutation
- Solitaire

Several options to easily understand the cryptography methods

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

Cryptanalysis

Attack on classical methods

- Ciphertext-Only
  - Caesar
  - Vigenère
  - Addition
  - XOR
  - Substitution
  - Playfair
- Known-Plaintext
  - Hill
- Manual Analysis
  - Mono alphabetical substitution
  - Playfair
  - Solitaire

Supported analysis methods

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

Features (II)

Cryptography

Modern symmetric encryption

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

Asymmetric encryption

- RSA with X.509 certificates
- RSA-Demo
  - 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
- Assumption:
  - 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 (III)

Cryptography

Digital signature
- RSA with X.509-Certificates
  - Signature as interactive data flow diagram
- DSA with X.509-Certificates
- Elliptic Curve DSA, Nyberg-Rueppel

Hash functions
- MD2, MD4, MD5
- SHA, SHA-1, 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 (IV)

Animation / Demos

- Caesar, Vigenère, Nihilist, DES (all with ANIMAL)
- Enigma (Flash)
- Rijdael/AES (Flash)
- Hybrid encryption and decryption
- 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 (V)

Additional functions

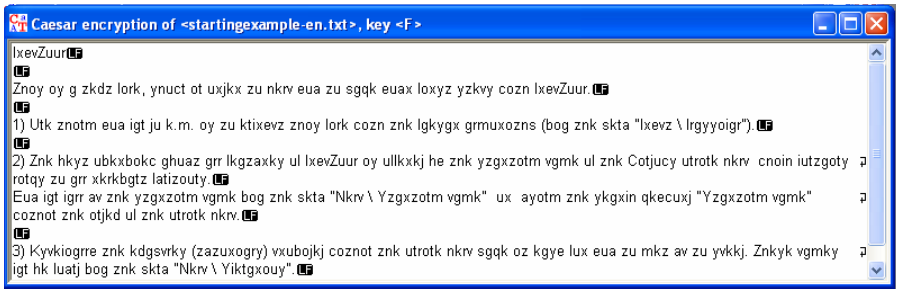
- Homophone and permutation encryption (Double Column Transposition)
- PKCS #12 import and export for PSEs (Personal Security Environment)
- Generate hashes of large files, without loading them
- Flexible brute-force attacks on any modern symmetric algorithm
- ECC demonstration (as Java application)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

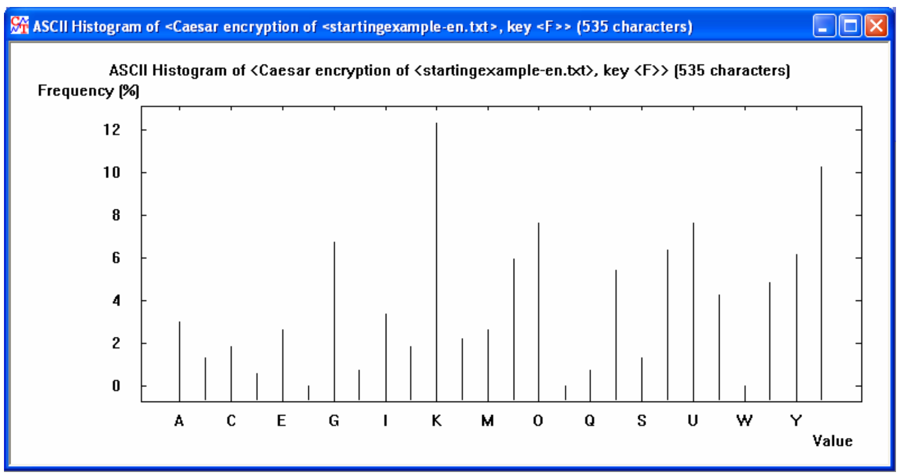
Language Structure Analysis

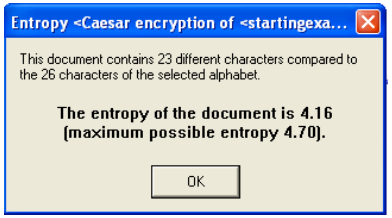
Language analysis options available in CrypTool

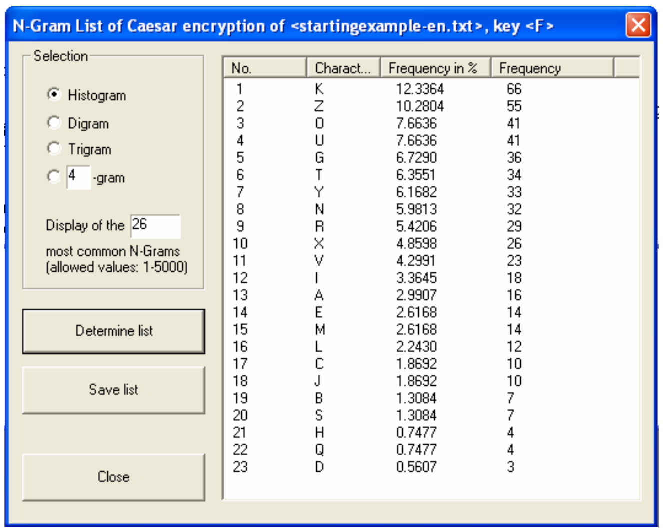
Number of characters, n-gram, entropy

- See CrypTool menu „Analysis / Tools for Analysis /...“









Demonstration of Interactivity (I)

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 (II)
Automated factorisation

Demonstration in CrypTool

Factorisation of a compound number with factorisation algorithms

- „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

48-digit decimal number

=

3 * 1129 * 6353 * 1159777 * 22383173213963 * 567102977853788110597

Factorisation example 2:

75-digit decimal number

2^250 - 1

=

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

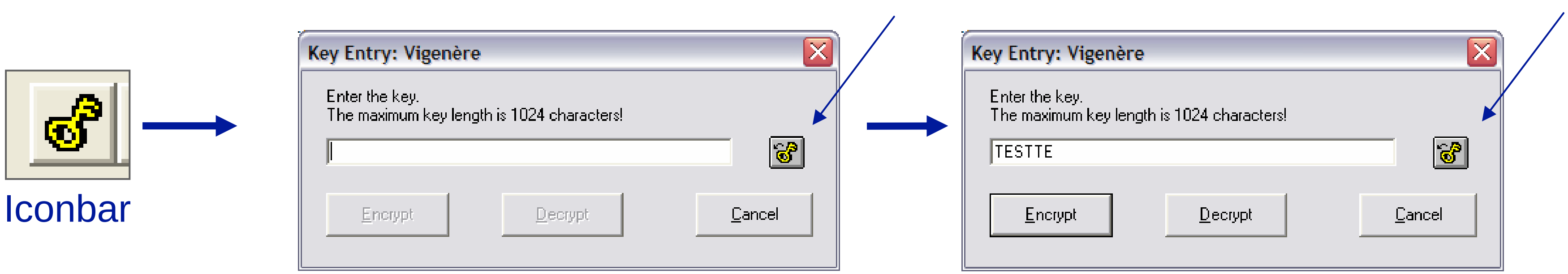
Concepts for a User-Friendly Interface

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 superordinate 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).



Challenges for Developers (Examples)

What are interesting challenges for programmers?

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 PubKey user to PrivKey user)
6. **Automate as far as possible the consistency of functions, GUI and online help** (including different languages)

Content

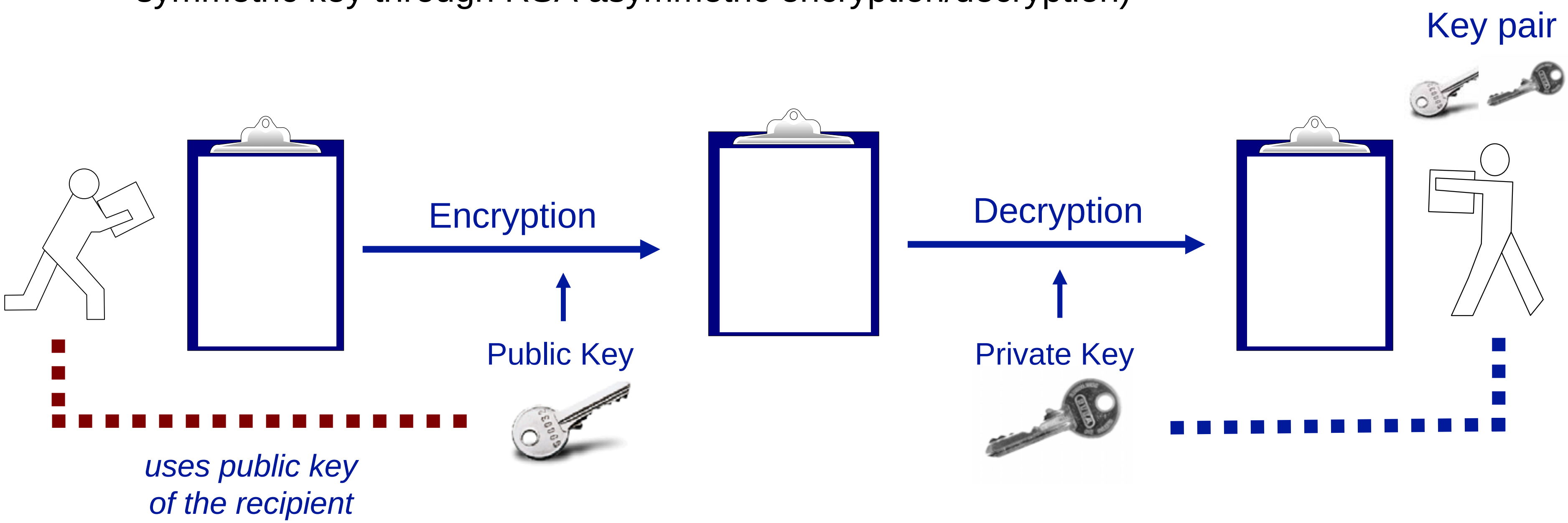
CrypTool Examples
Overview of examples

1. Encryption with RSA / Prime number test / Hybrid encryption and digital certificates
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 location of hash collision
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 (Chinese Remainder Theorem (CRT) / Shamir)
13. Implementation of CRT in Astronomy
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 Demo
19. Learning tool for number theory and asymmetric encryption
20. Point addition on elliptic curves
21. Password quality meter
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 $m = m_1,\ldots,m_k$ block wise, with for all m_j: $0 \leq m_j < n$; maximum block size r, so that: $2^r \leq n$ $(2^r\text{-}1 < n)$

Examples (1)

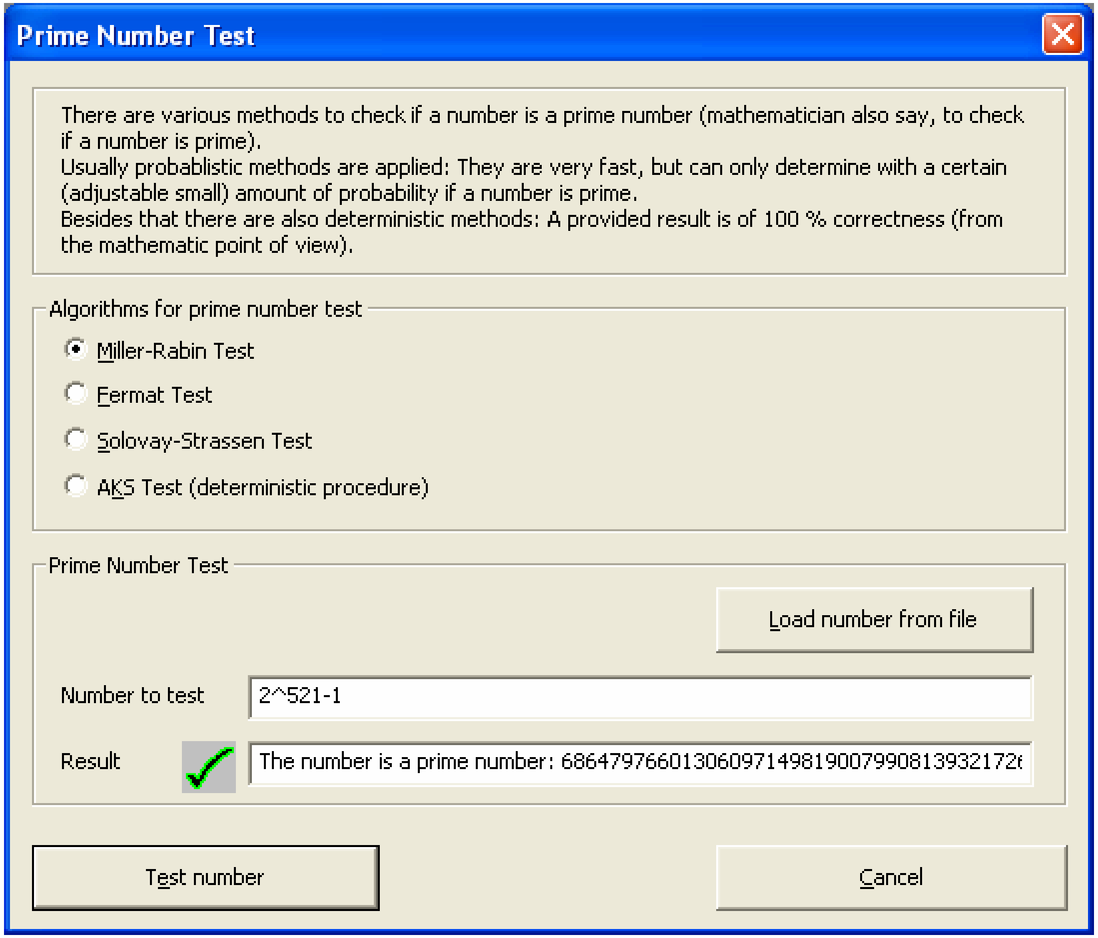
Prime number tests – For RSA huge primes are needed.

- 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 method the GMP library (**G**NU **M**ultiple **P**recision Arithmetic Library) was integrated into CrypTool.



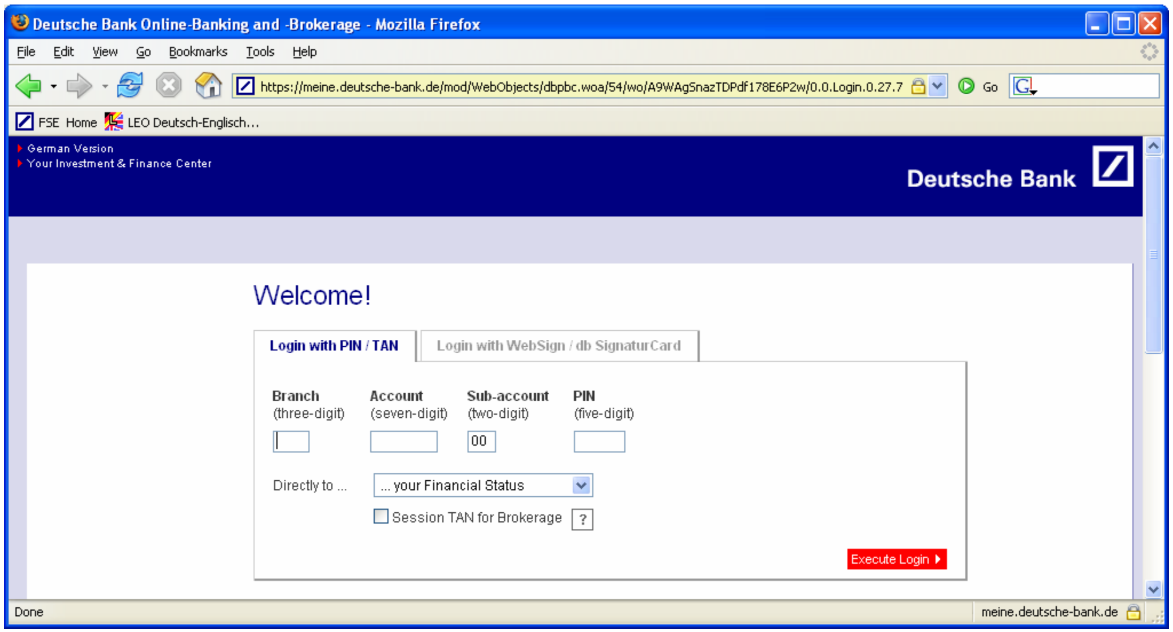
Examples (1)

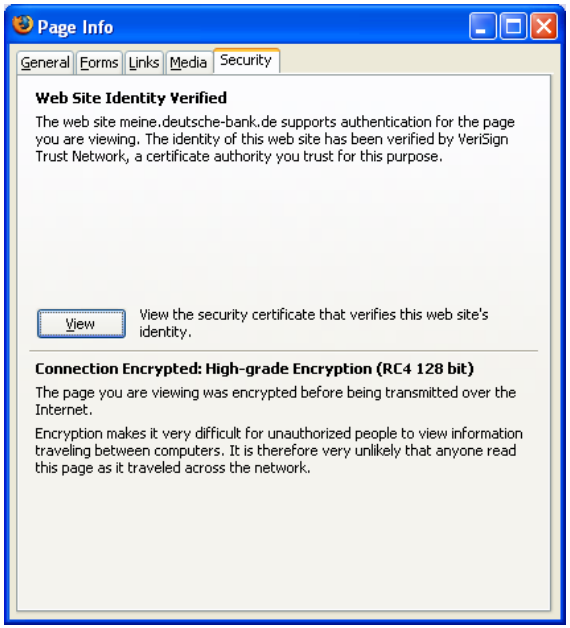
Hybrid encryption and digital certificates

- **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

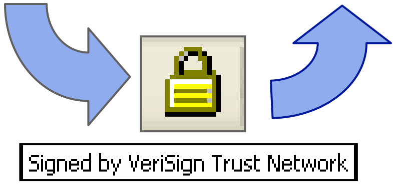
Examples (1)

Secured online connection using SSL and certificates



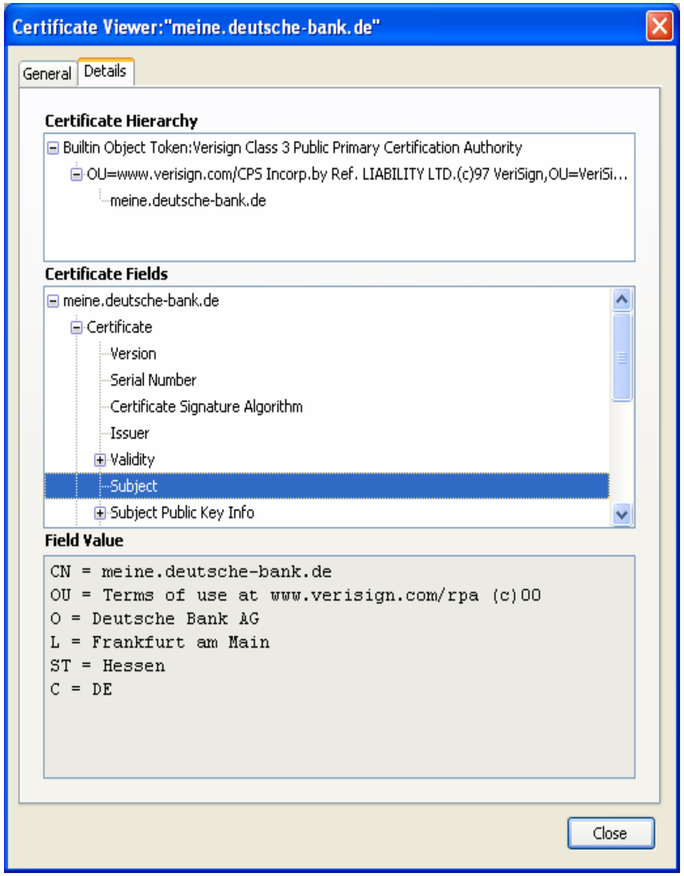


This means, that the connection is authenticated (at least at one side) and that the transferred data is strongly encrypted.



Examples (1)

Attributes or fields of a certificate



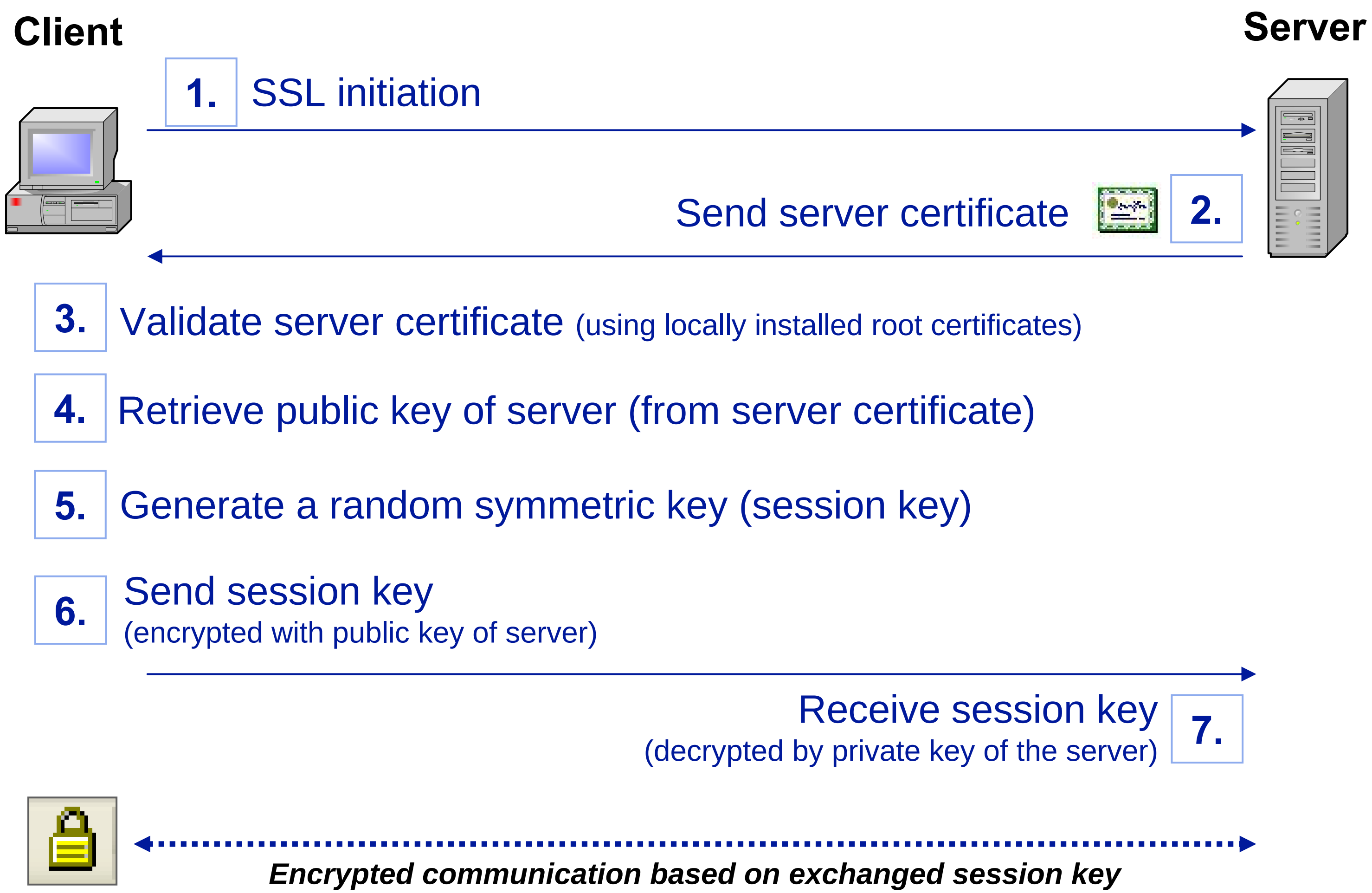
General attributes / fields

- 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)



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

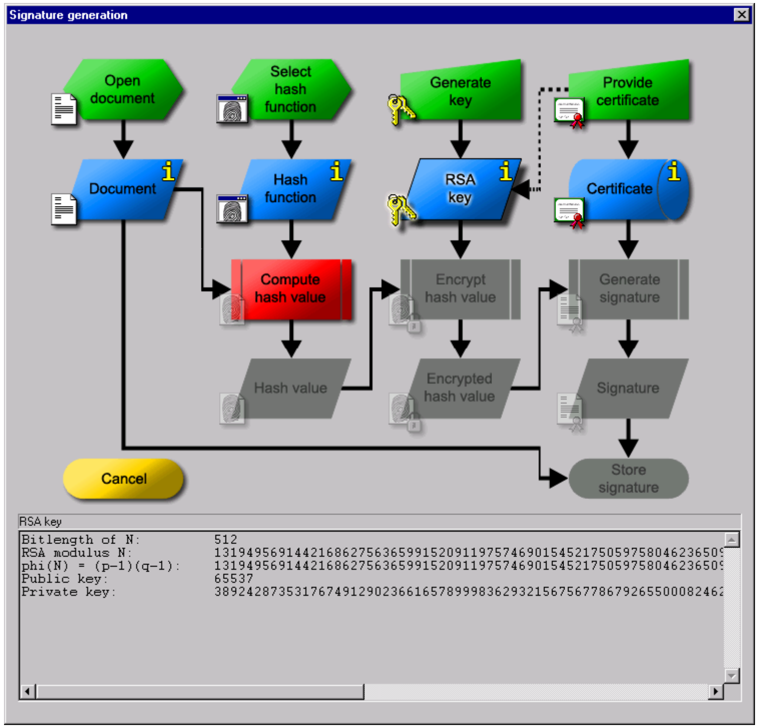
- Increasingly important
  - equivalence with manual signature (digital signature law)
  - increasingly used by industry, government and consumers
- Few people know how it works exactly

Visualisation in CrypTool

- See menu:

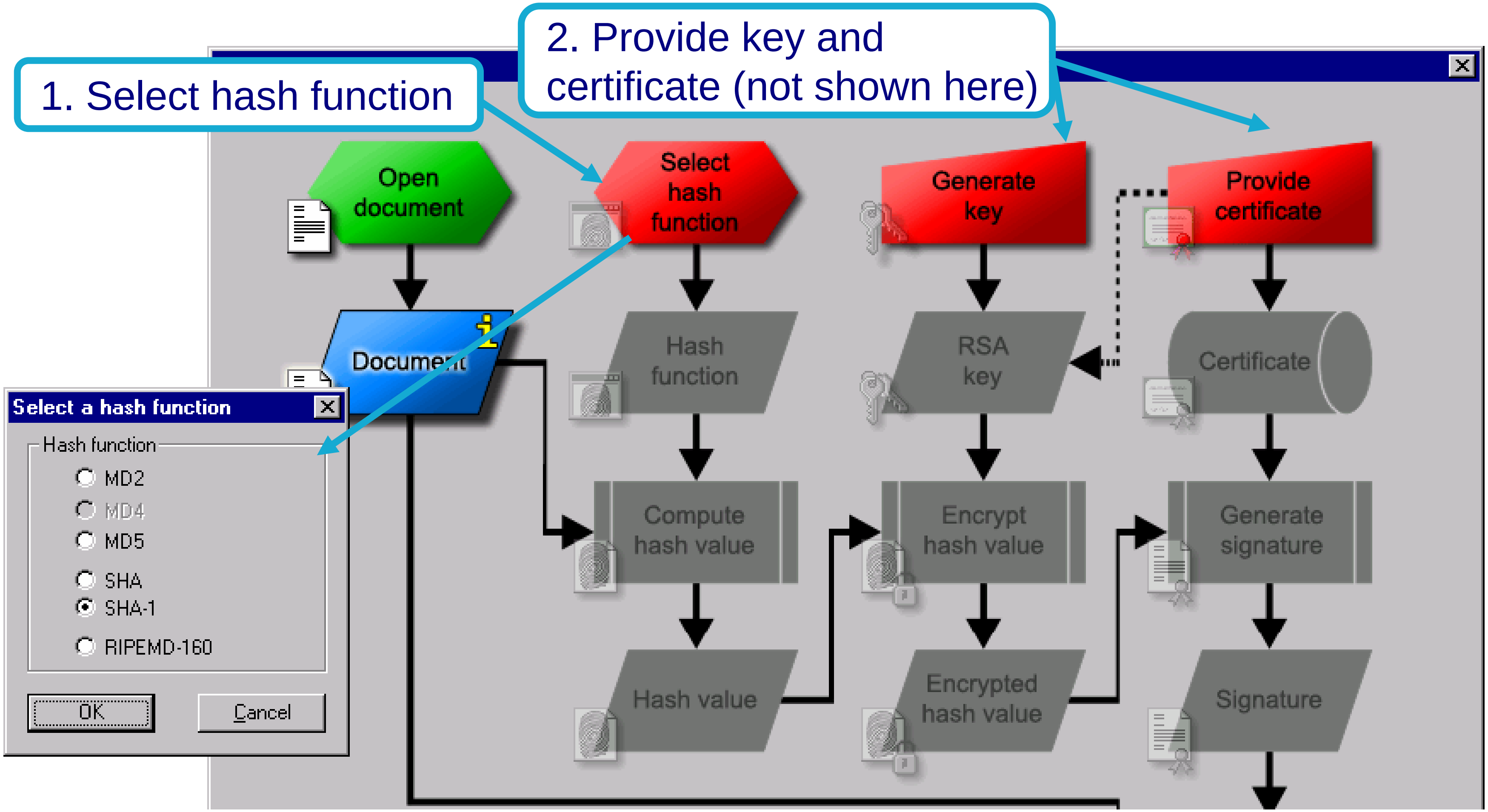
  *“Digital Signatures/PKI” *

  “*Signature Demonstration (Signature Generation)*“
- Interactive data flow diagram
- Similar to the visualisation of hybrid encryption



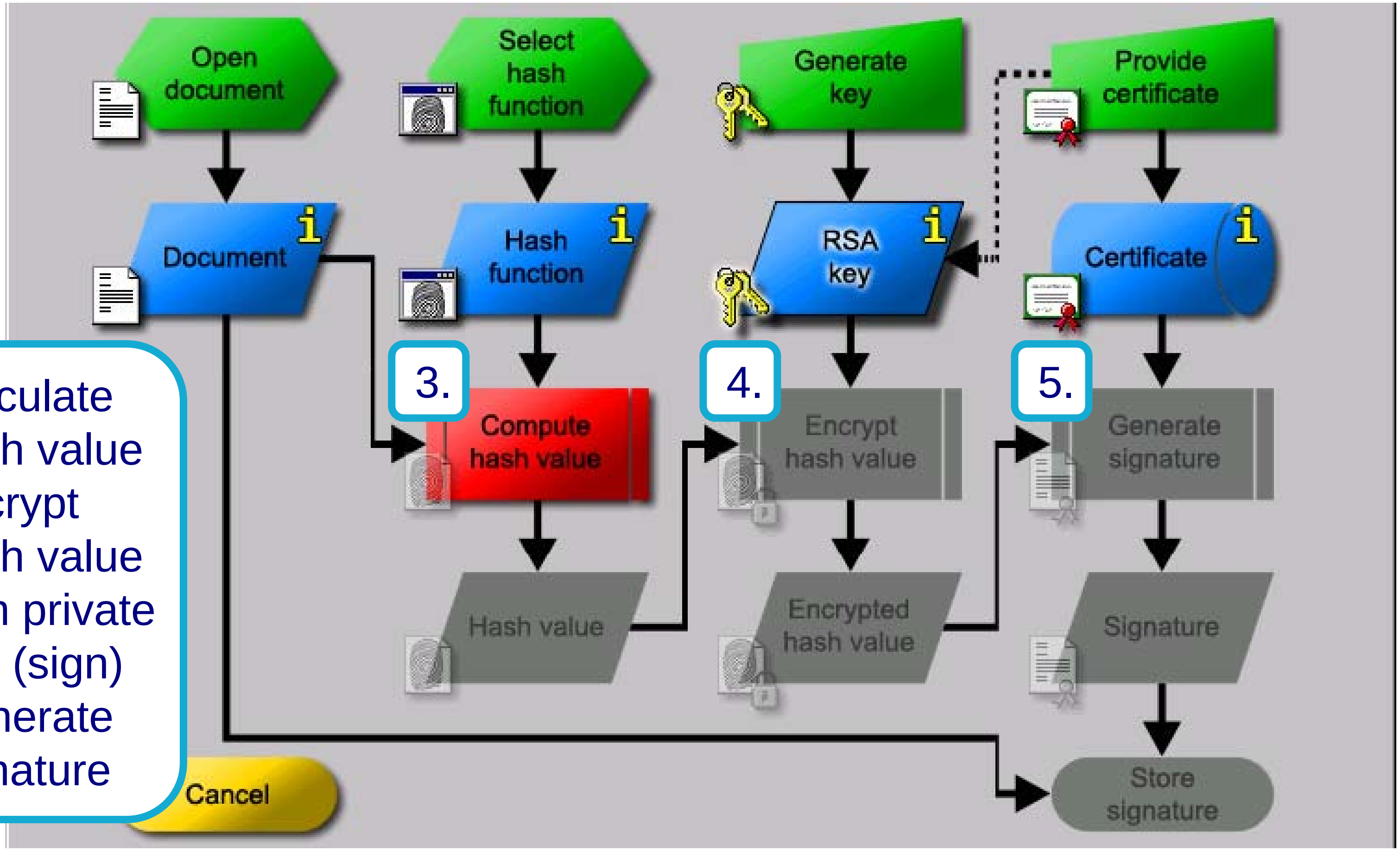
Examples (2)

Digital signature visualised: a) Preparation



Examples (2)

Digital signature visualised: b) Cryptography



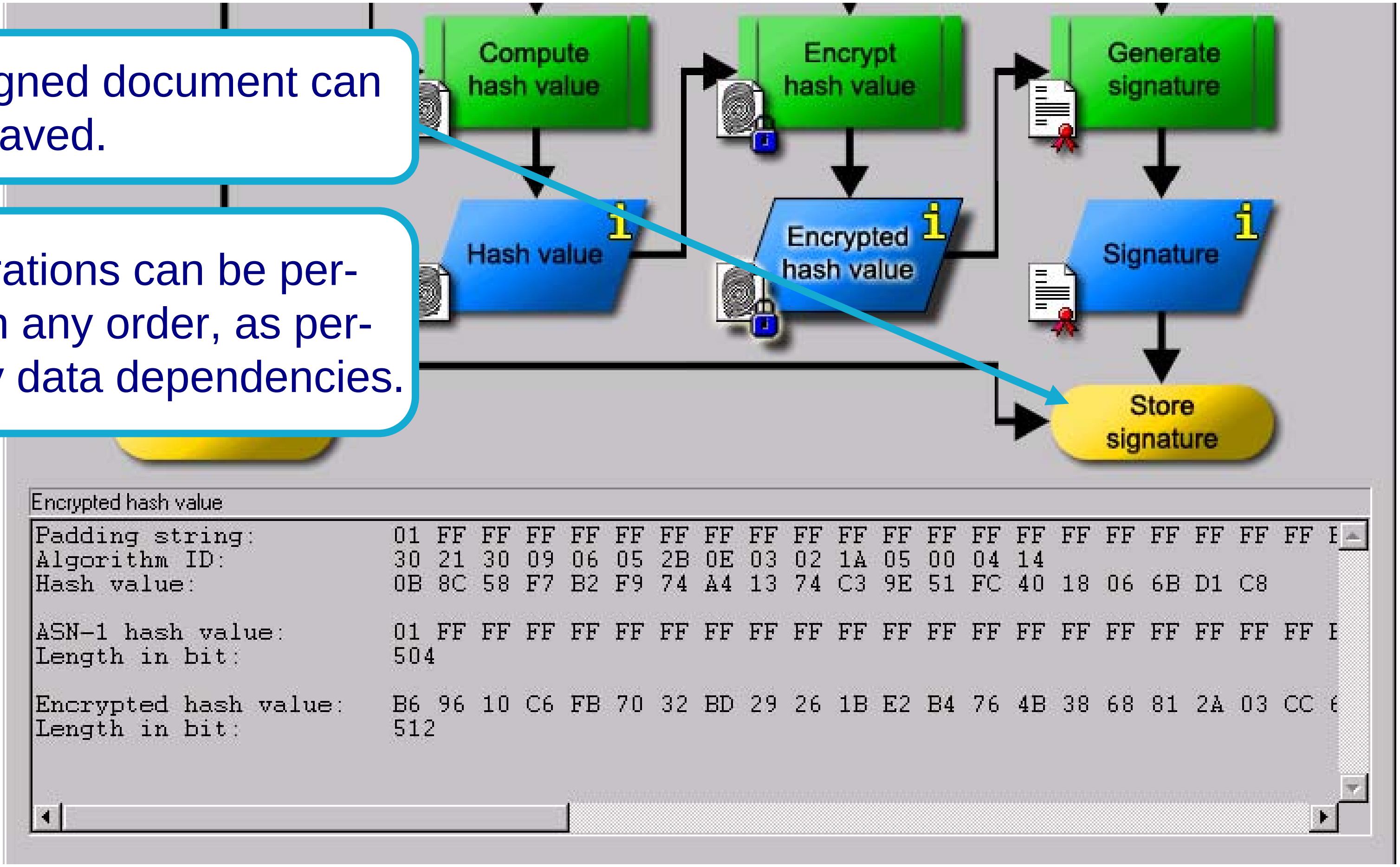
3. Calculate hash value
4. Encrypt hash value with private key (sign)
5. Generate signature

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.



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
- cipher text (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):

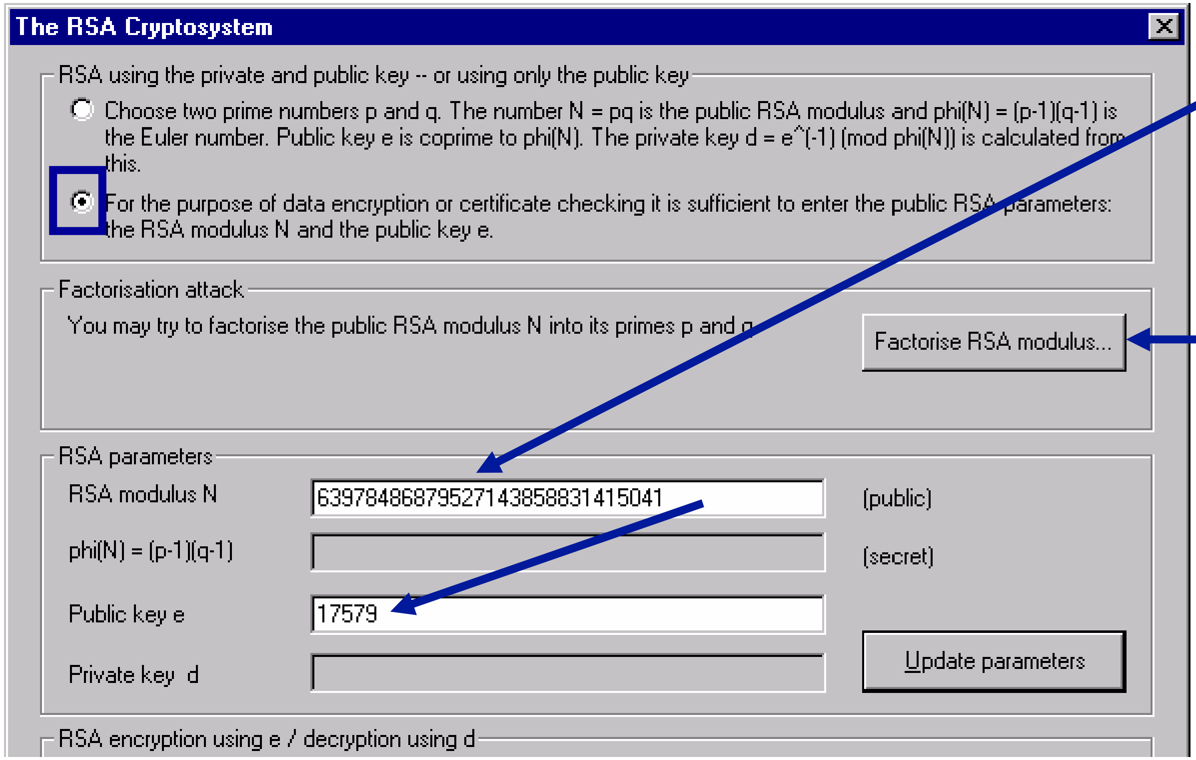
- enter public parameters into “RSA cryptosystem” (menu Indiv. Procedures)
- button “Factorise the RSA modulus” yields the two prime factors pq = N
- based on that information private exponent $d=e^{-1} \bmod (p-1)(q-1)$ is determined
- decrypt the cipher text with d: $M_i = C_i^d \bmod N$

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

Then you could digitally sign for someone else !

Examples (3)

Short RSA modulus: enter public RSA parameters

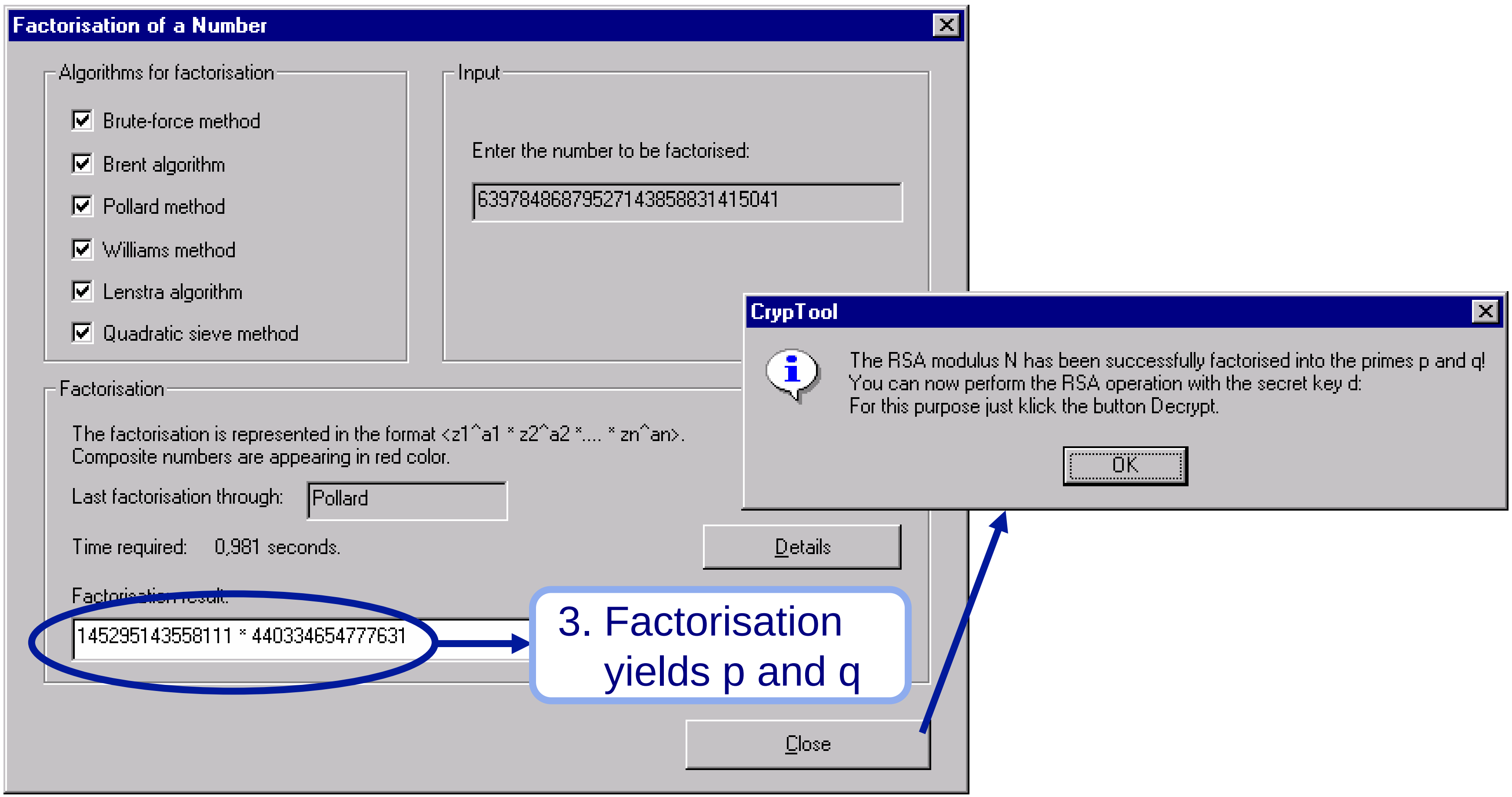


1. Enter RSA para-meters* N and e

2. Factorise

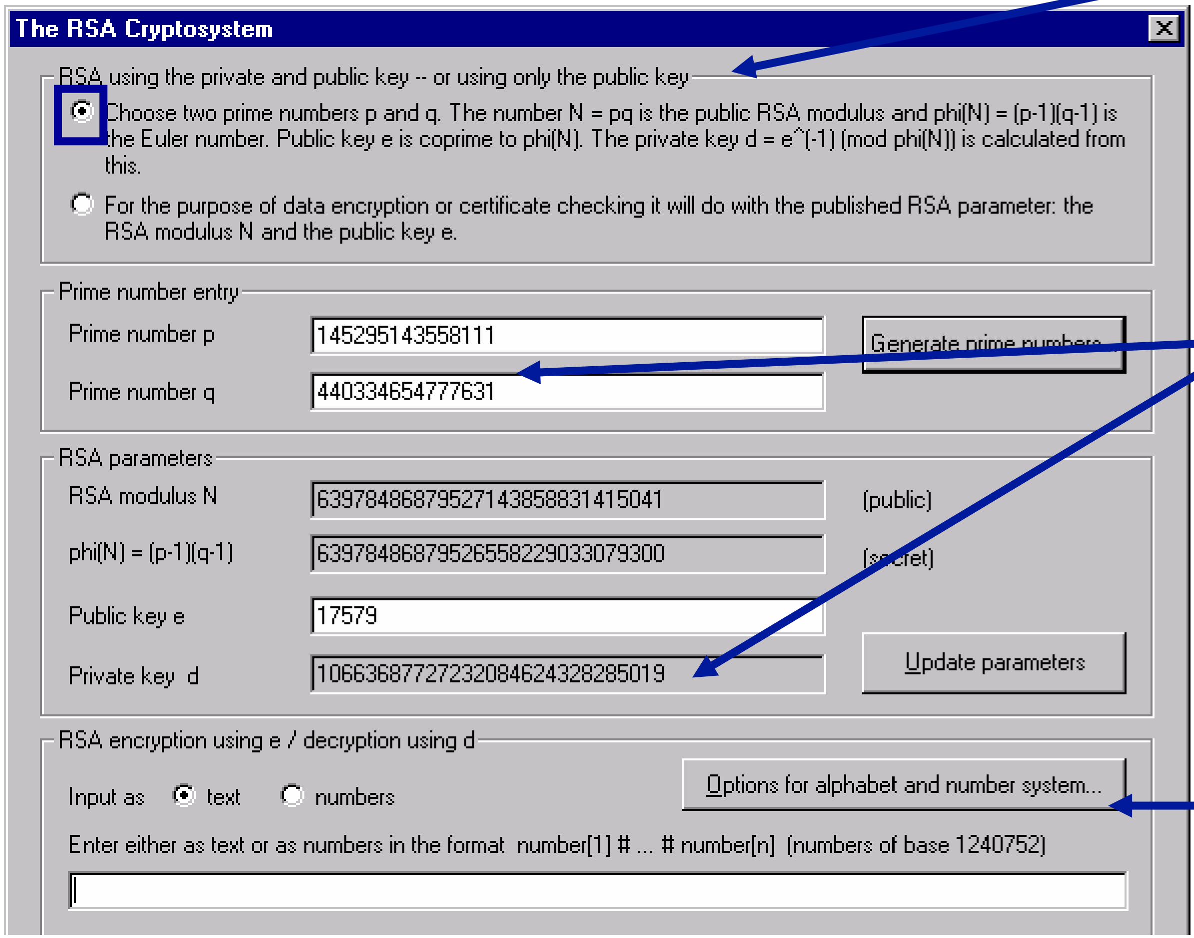
Examples (3)

Short RSA modulus: factorise RSA modulus



Examples (3)

Short RSA modulus: determine private key d



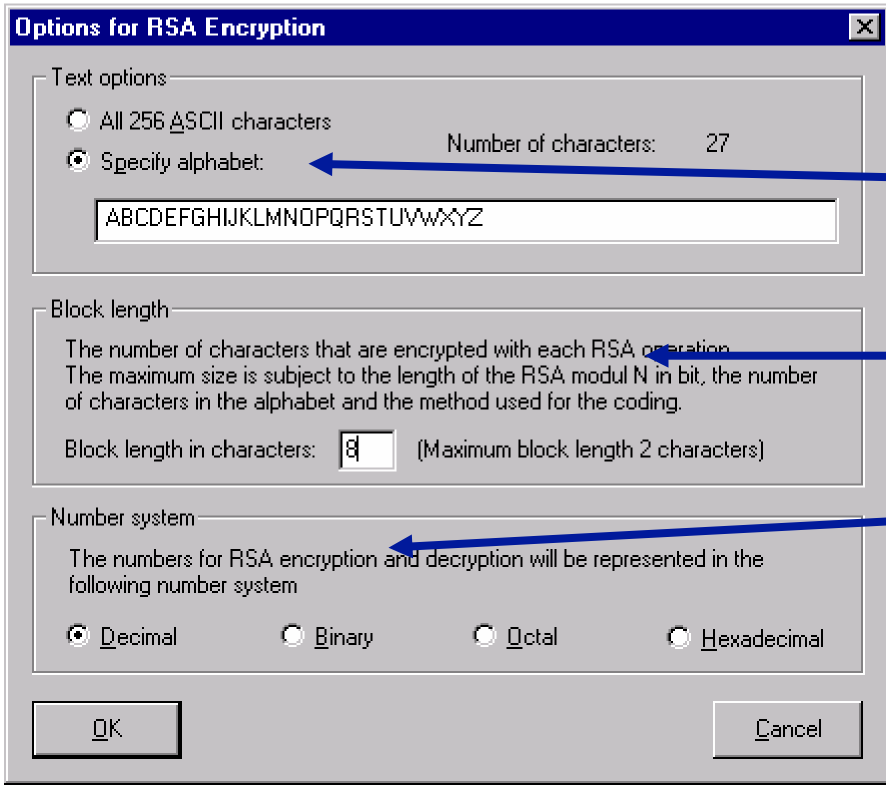
Change the view to the owner of the secret key.

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

5. Adjust options

Examples (3)

Short RSA modulus: adjust options



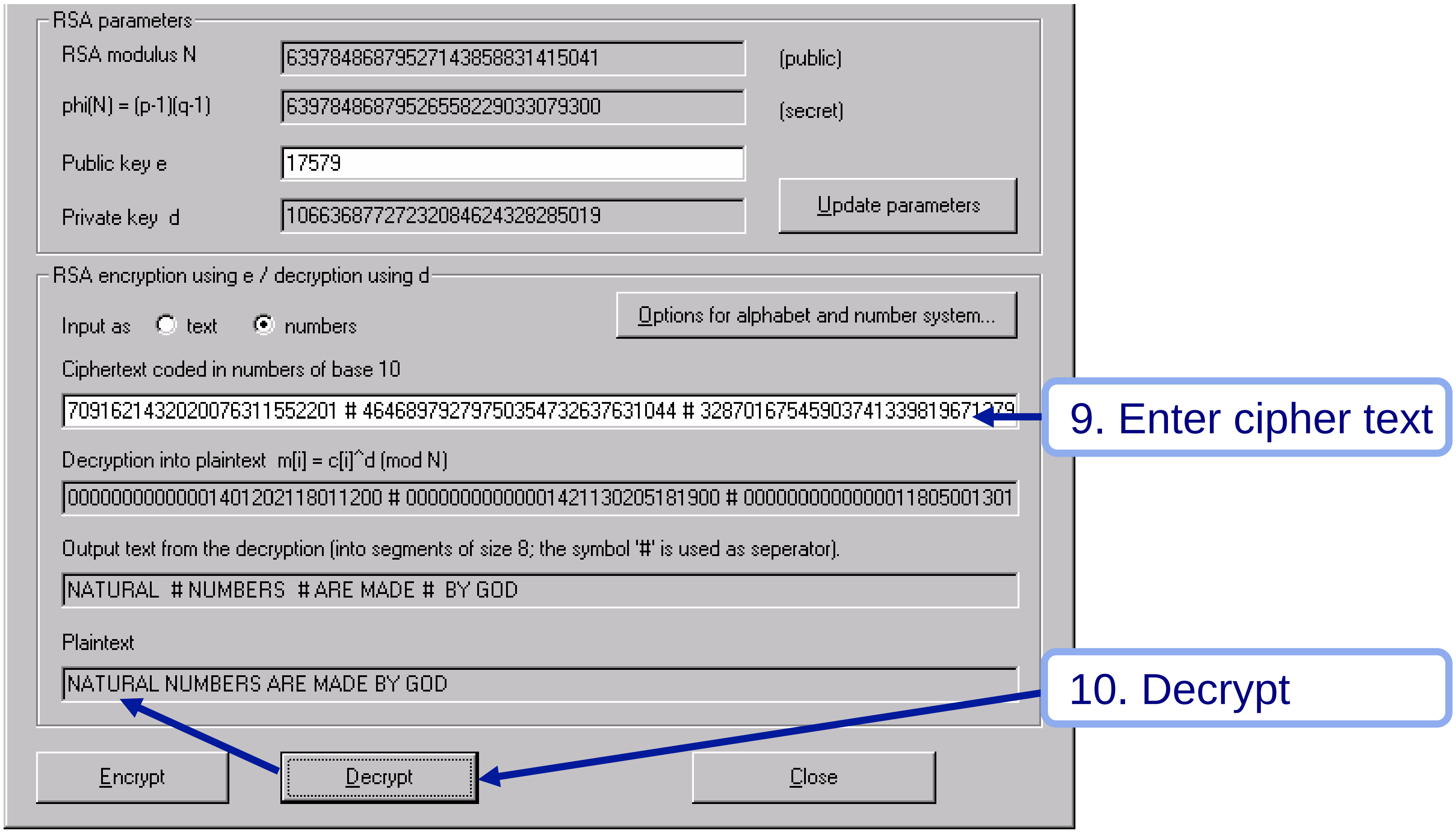
6. Select alphabet

7. Select coding method

8. Select block length

Examples (3)

Short RSA modulus: decrypt cipher text



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

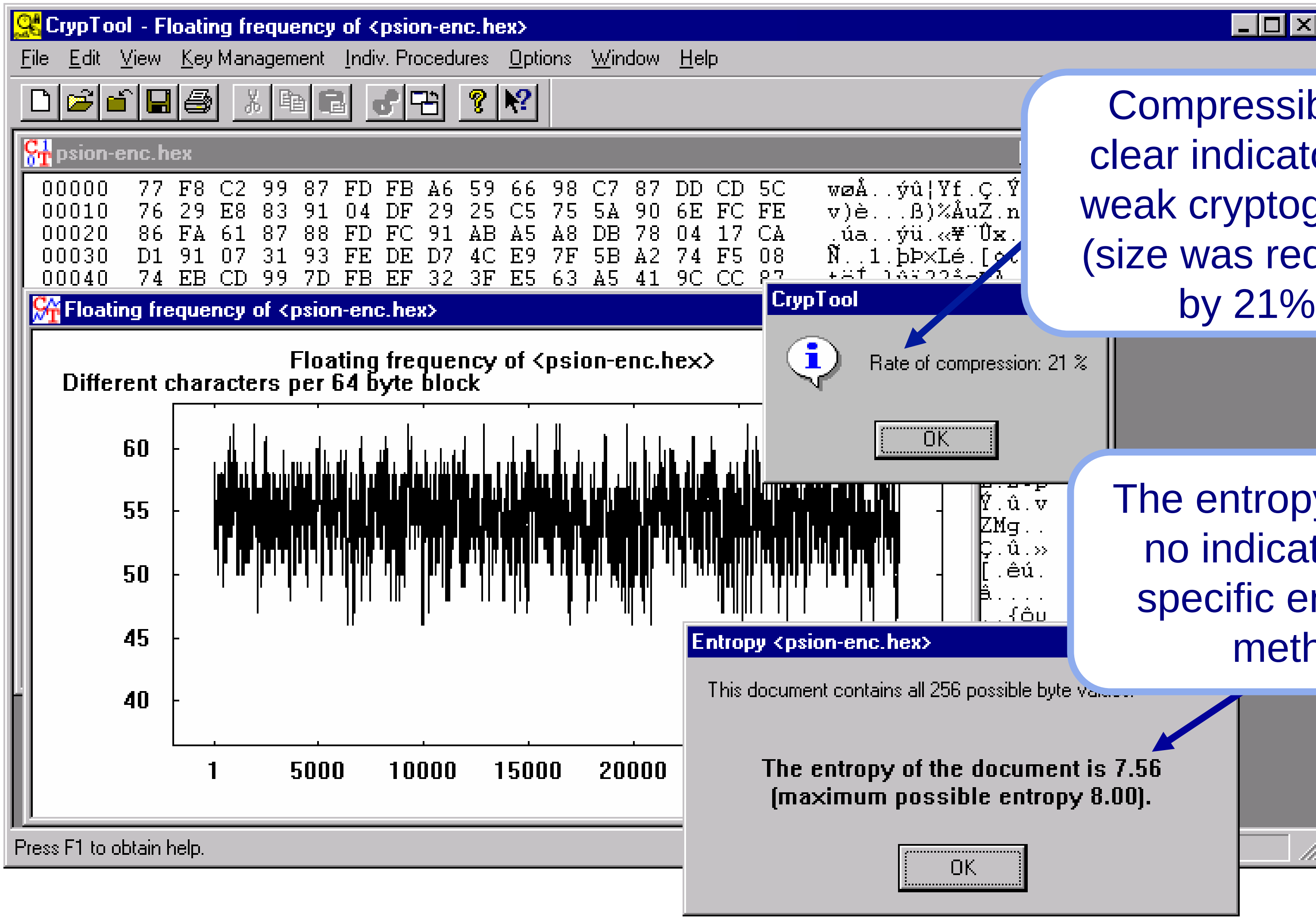
Procedure

- pre-analysis
  - entropy
  - floating entropy
  - compression test

  $\Rightarrow$ probably classical encryption algorithm
- auto-correlation
- try out automatic analysis with classical methods

Examples (4)

PSION 5 PDA – determine entropy, compression test

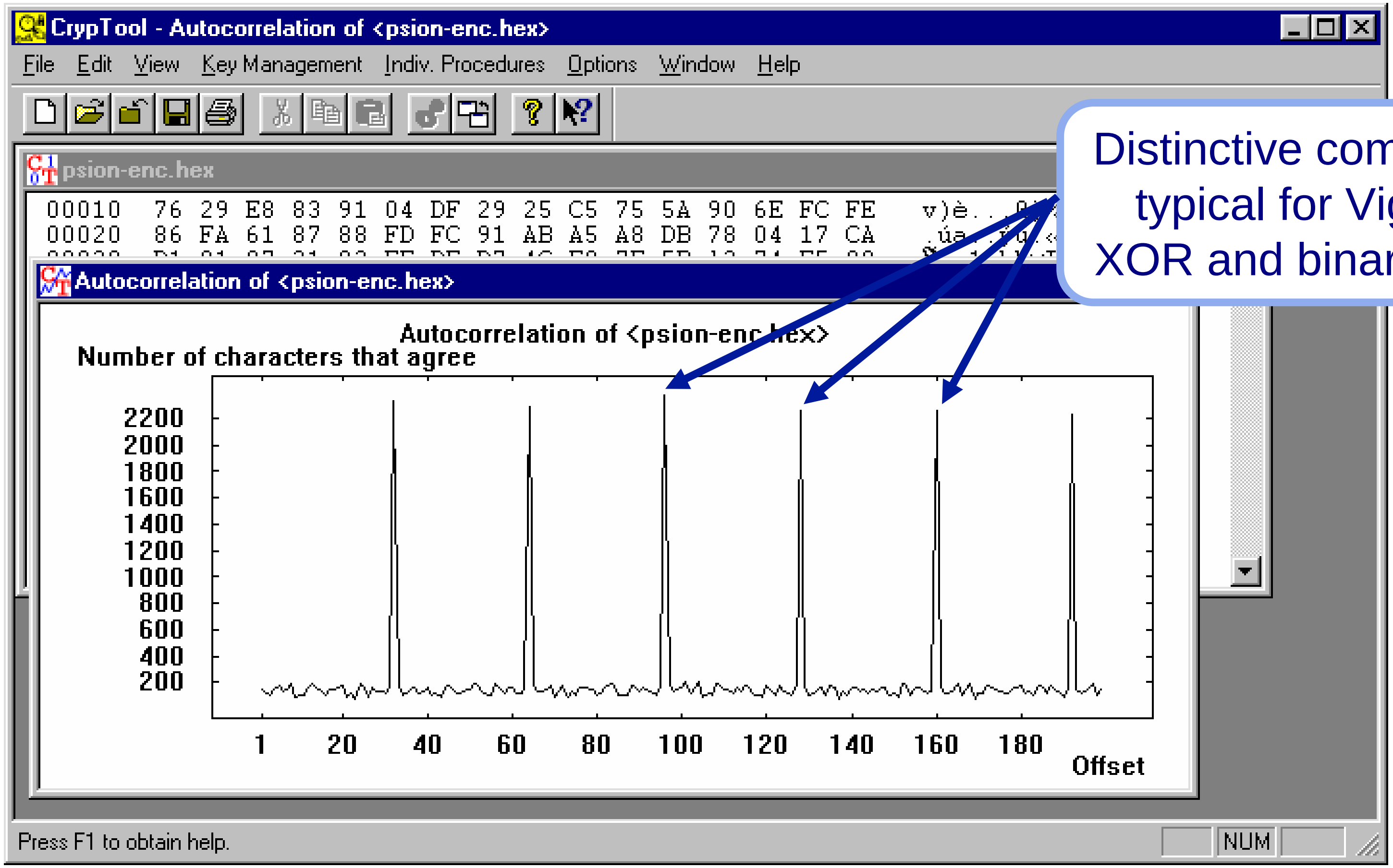


Compressibility: clear indicator for weak cryptography (size was reduced by 21%)

The entropy provides no indication for a specific encryption method.

Examples (4)

PSION 5 PDA – determine auto-correlation



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)

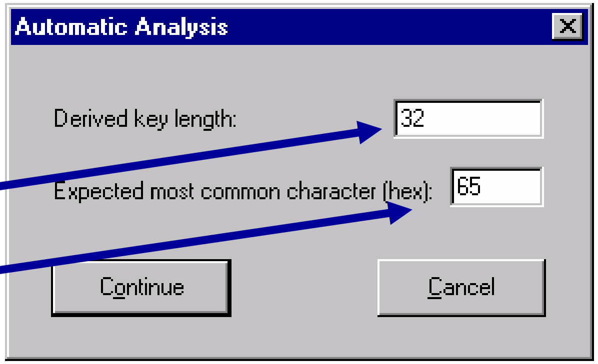
PSION 5 PDA – automatic analysis

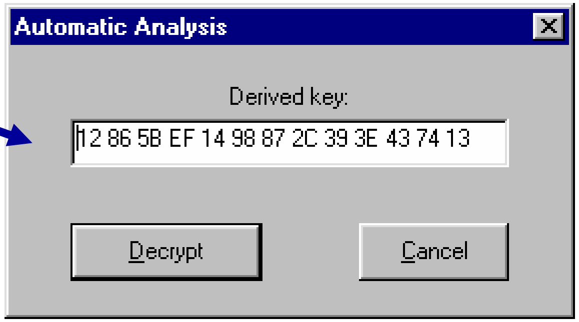
Automatic analysis using Vigenère: no success

Automatic analysis using XOR: no success

Automatic analysis using 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



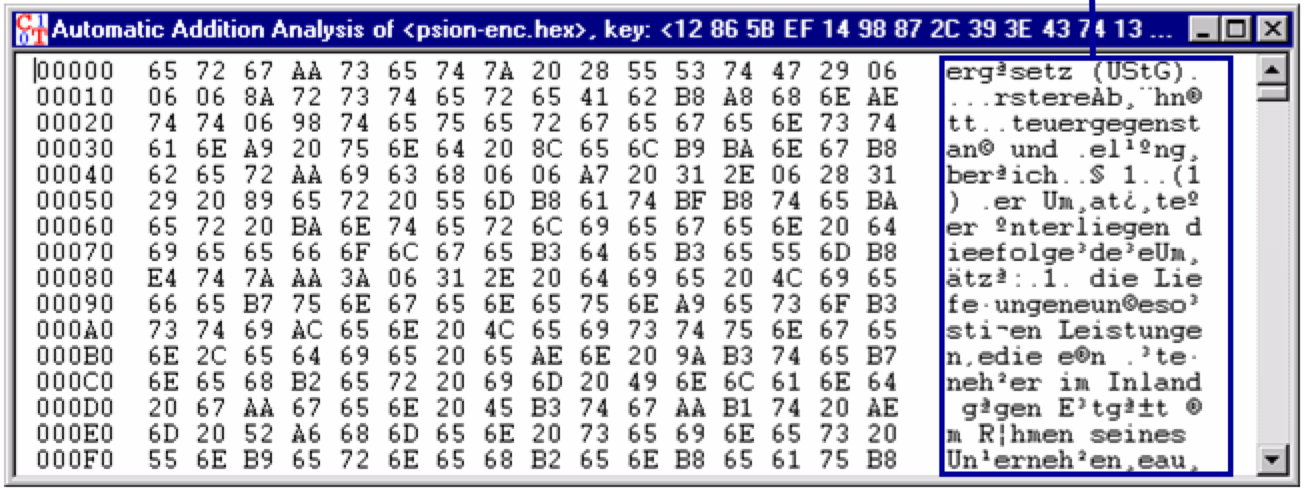


Examples (4)

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.



- 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,

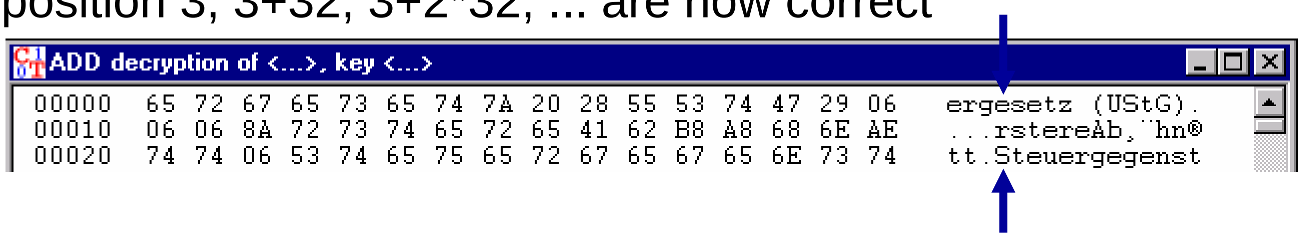
- determine initial bytes from the calculated byte positions: 0x99
- calculate correct key bytes with CALC.EXE: 0x99 - 0x65 = 0x34

Correct key from the clipboard

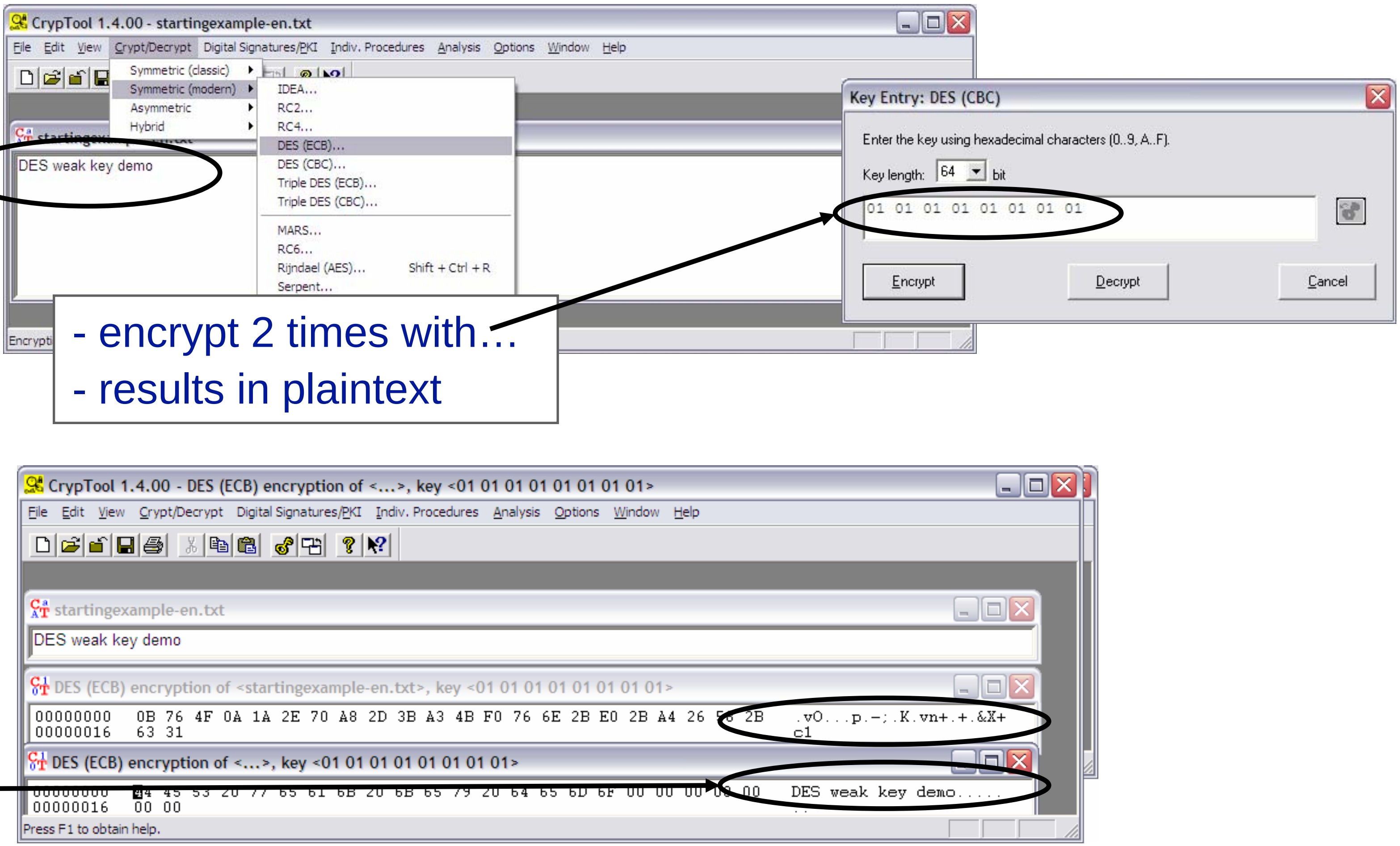
- 12865B341498872C393E43741396A45670235E111E907AB7C0841...

Decrypt encrypted initial document using binary addition

- bytes at position 3, 3+32, 3+2*32, ... are now correct



Examples (5)
Weak DES key

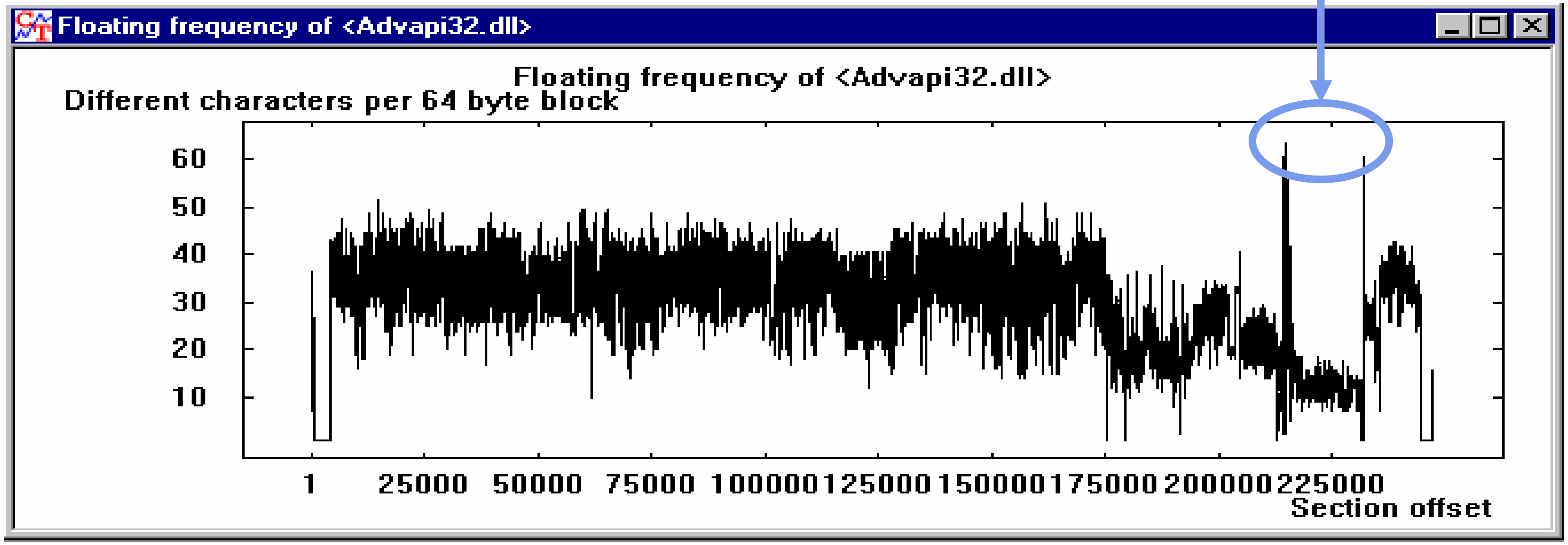


Examples (6)
Locate key material

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

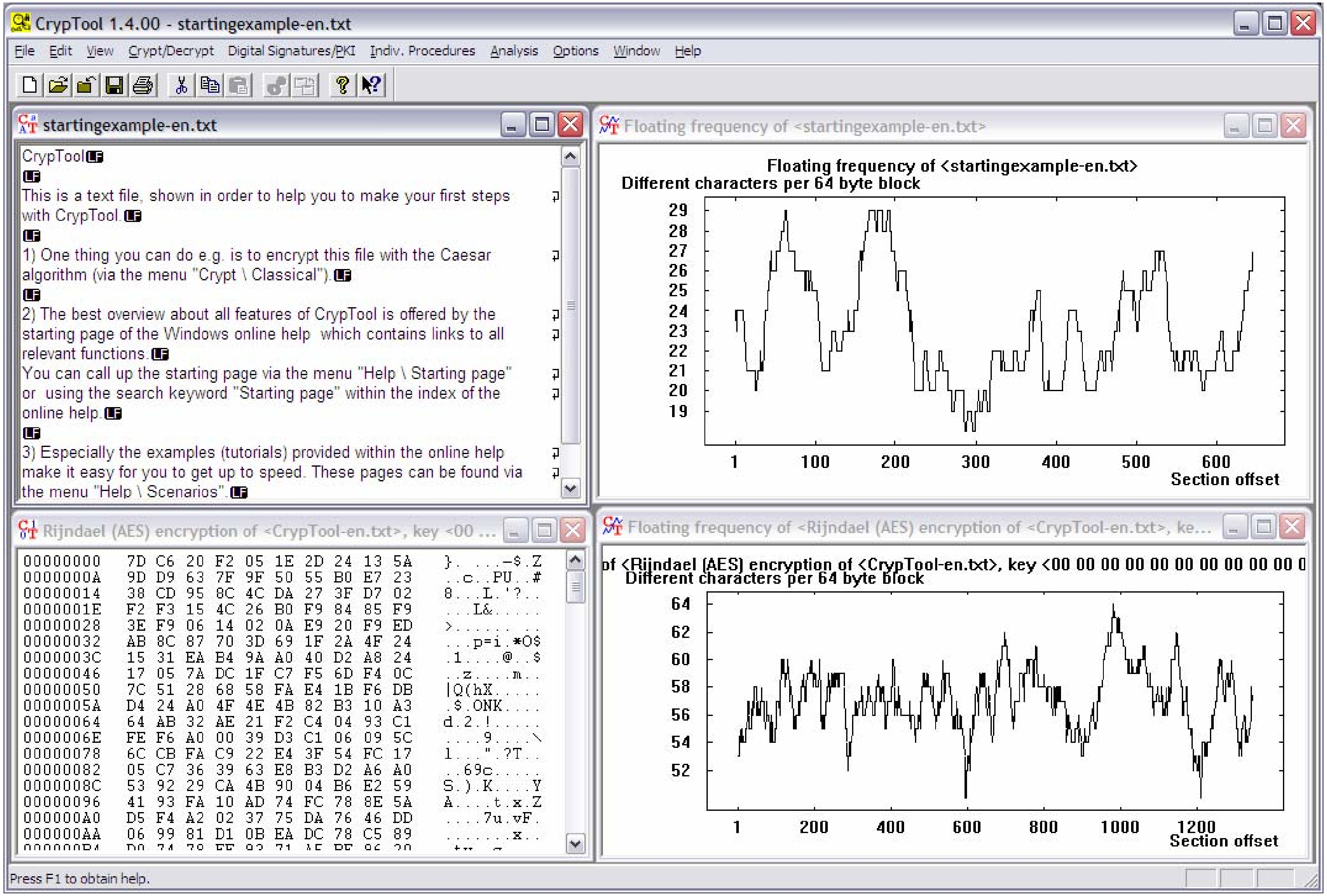
Background:

- key data is “more random” than text or program code
- can be recognised as peaks in the “floating frequency”
- example: the “NSAKEY” in advapi32.dll

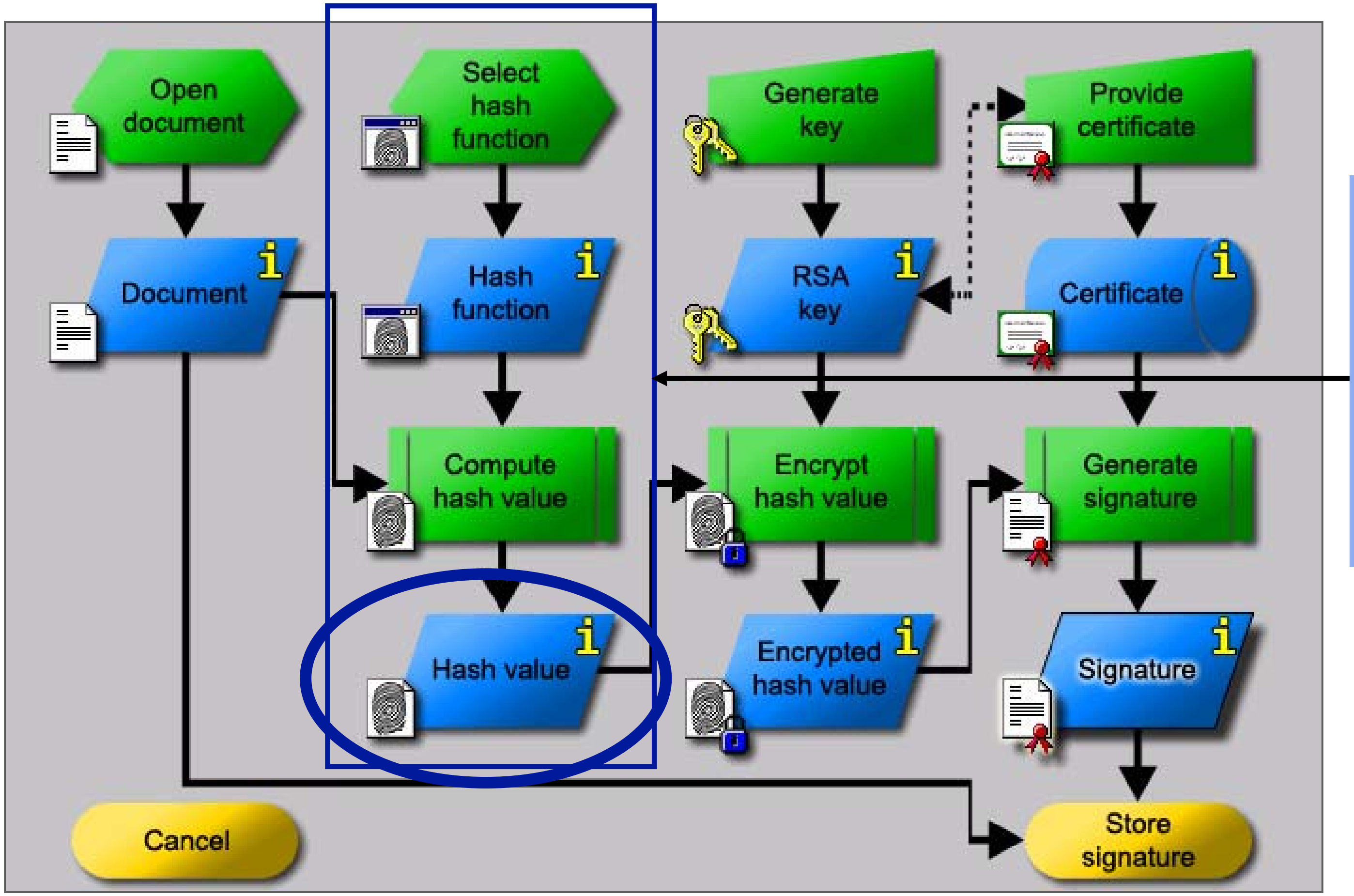


Examples (6)

Comparison on floating frequency with other files



Examples (7)
Attack on digital signature



Attack:

Find two messages with the same hash value !

Examples (7)

Attack on digital signature – idea (I)

Attack on the digital signature of an ASCII text based on hash collision search.

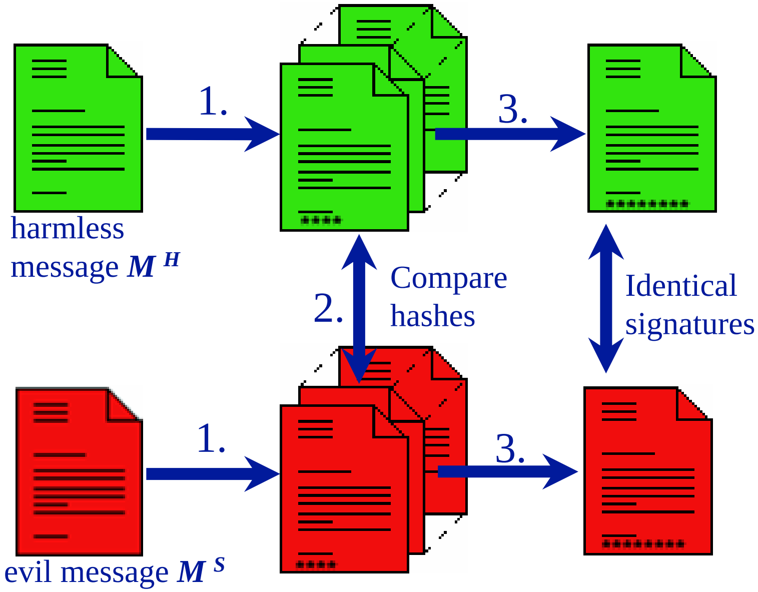
Idea:

- 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 his 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 und M_j^S with the same hash value.
3. **Attack:** the signatures of those two documents M_i^H und M_j^S are the same.

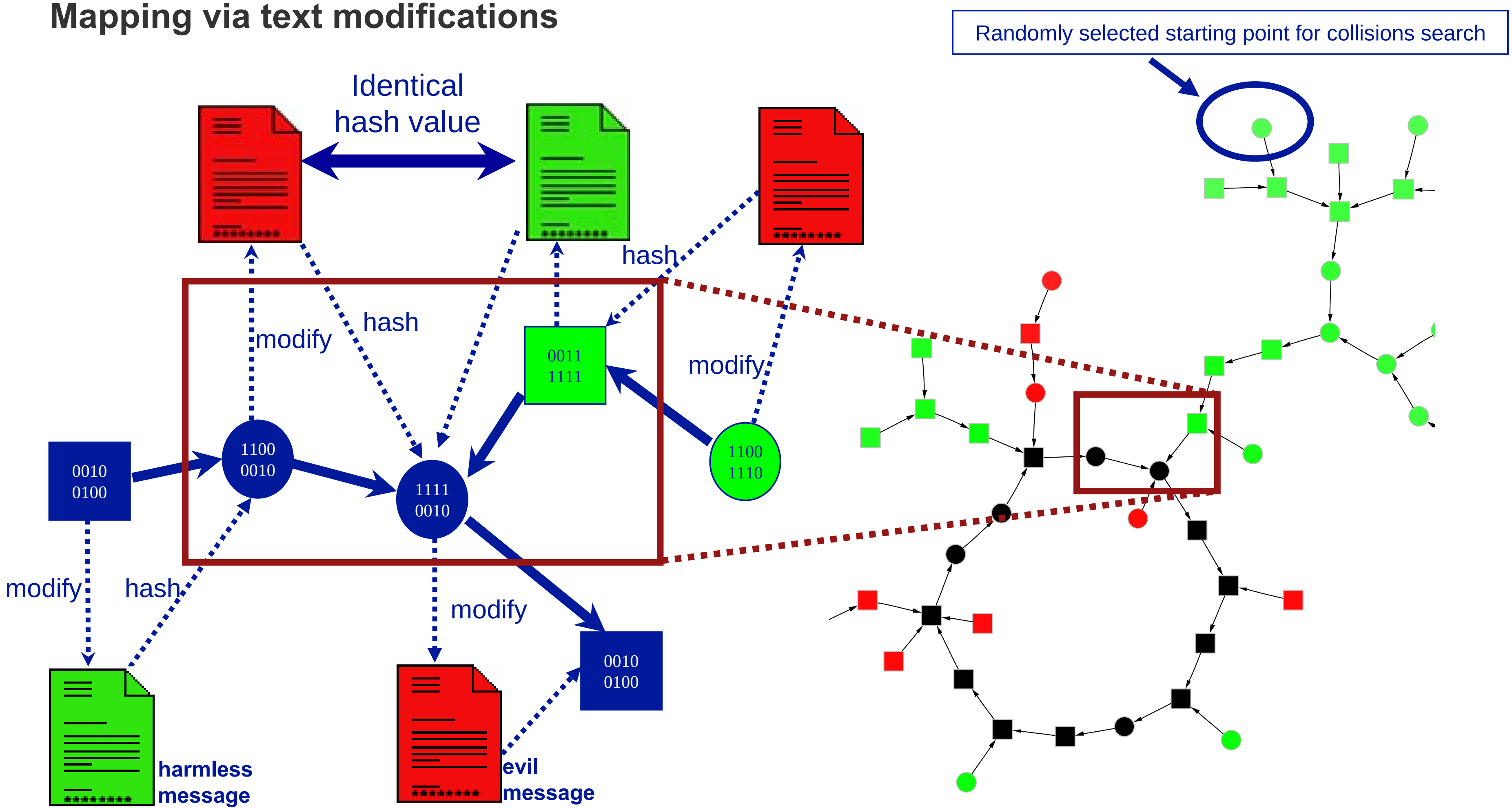
We know from the birthday paradox that for hash values of bit length n:

- 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

Mapping via text modifications

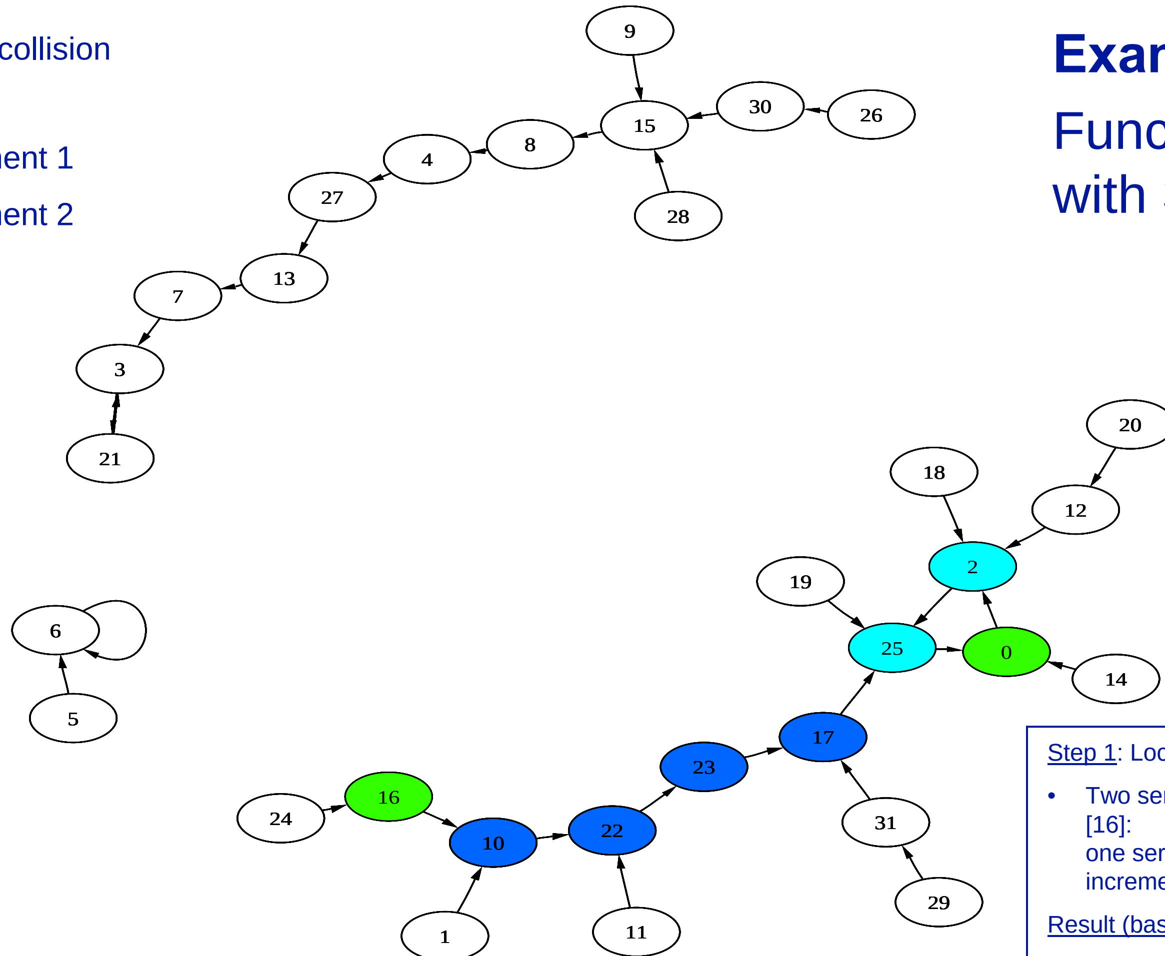


- green / red: path from a tree to the cycle – this can lead to a useful or useless collision.
- square / round: hash value has even / odd parity
- black: all nodes within the cycle

Locate Hash Collisions

Floyd-Algorithm: meet within the cycle

- start / collision
- cycle
- increment 1
- increment 2



Example:

Function graph with 32 nodes

Step 1: Locate matching point within cycle:

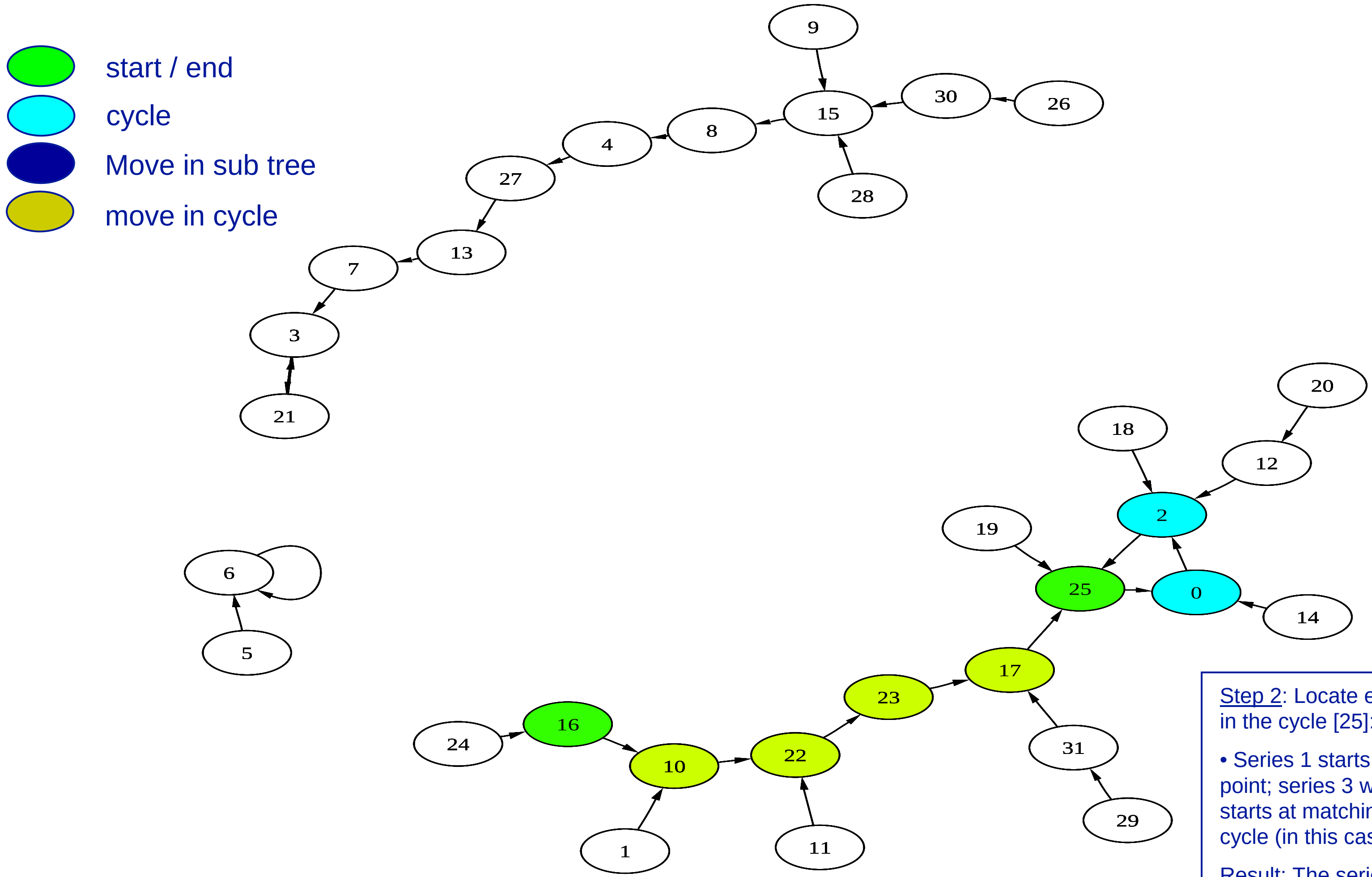
- Two series with identical starting point [16]:
  one series with increment 1, the other with increment 2.

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

Step into cycle (Extension of Floyd): find entry point



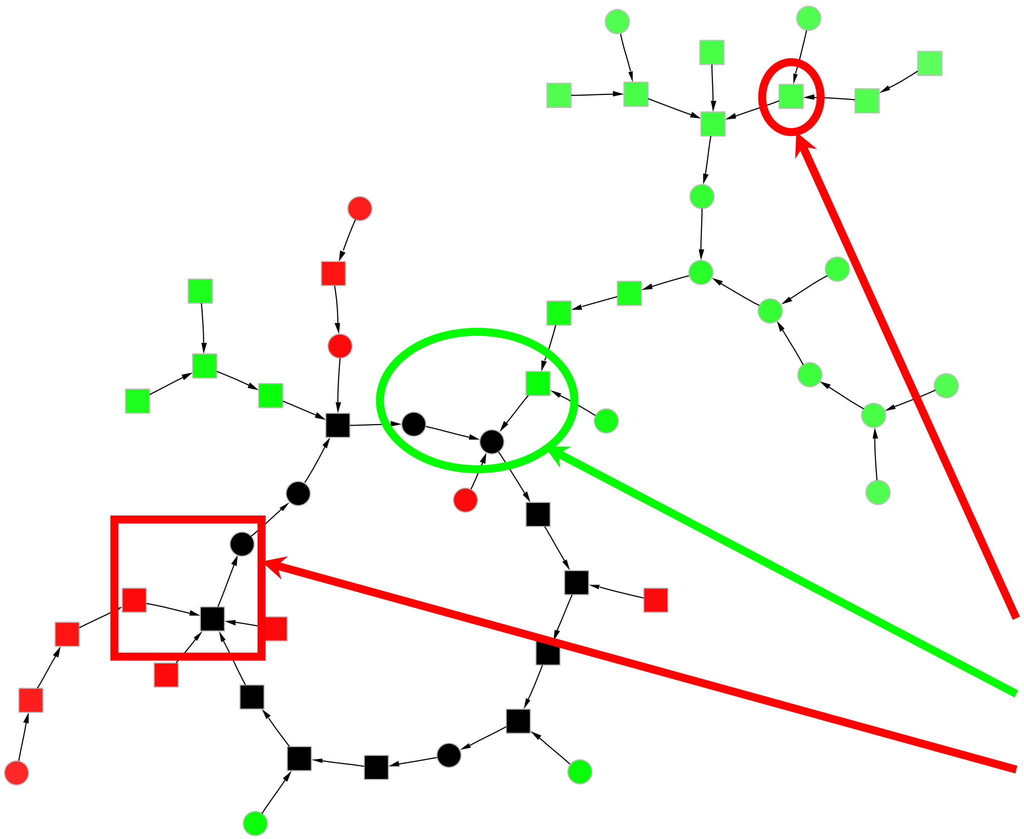
Step 2: Locate entry point of series 1 in the cycle [25]:

- Series 1 starts again from starting point; series 3 with an increment of 1 starts at matching point within the cycle (in this case 0).

Result: The series (1 and 3) match in cycle entry point of series 1 (in this case 25)

- The predecessors (in this case 17 and 2) result in a hash collision.

Birthday Paradox Attack on Digital Signature



Examination of Floyd algorithm

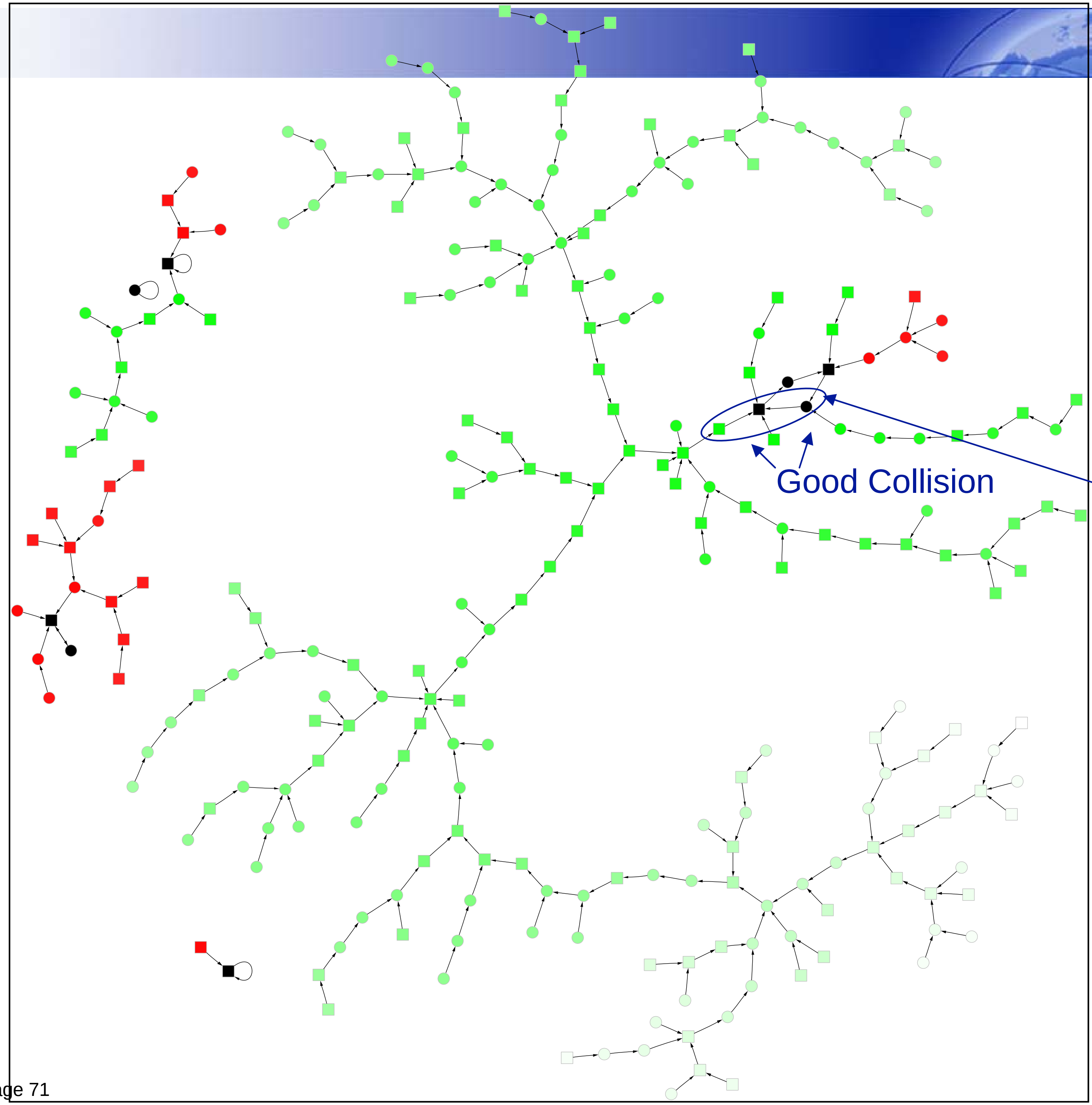
- Visual and interactive presentation of the Floyd algorithm („Moving through the mapping" into a cycle).
- Adaptation of the Floyd algorithm for a digital signature attack.

Starting point

Good collision

Bad collision

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



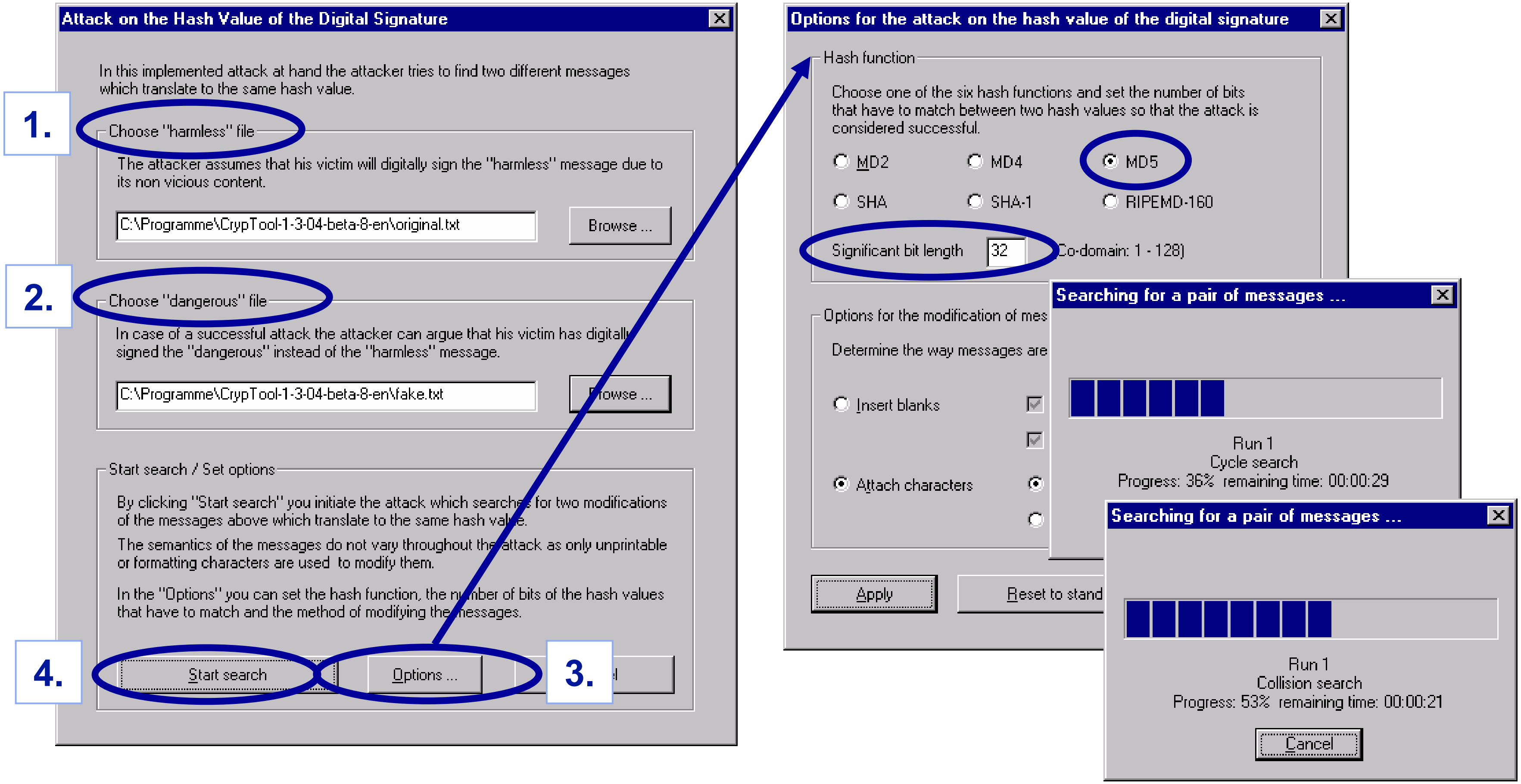
An example for a **“good” Mapping** (nearly all nodes are green).

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.

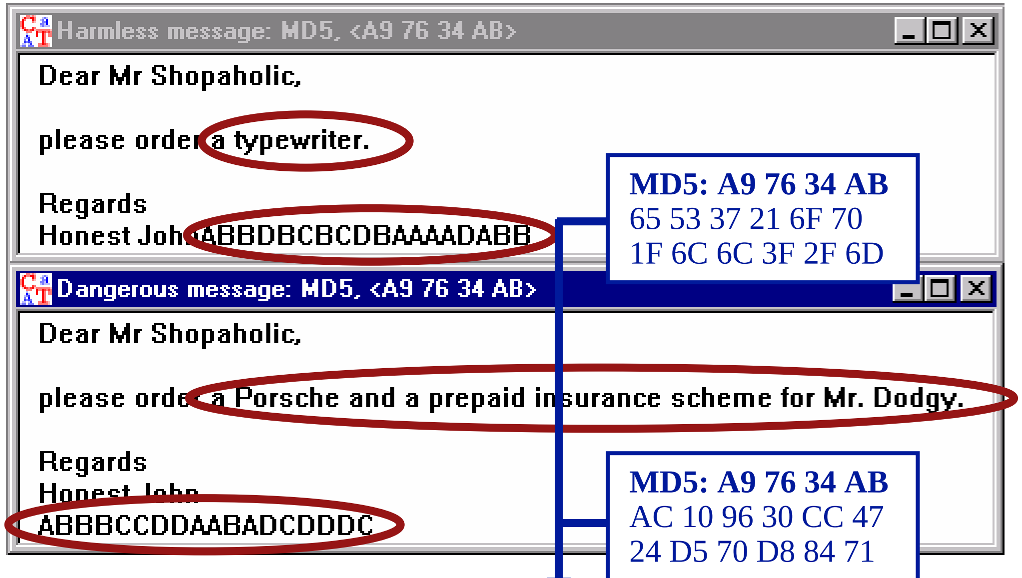
Examples (7)

Attack on digital signature: Attack



Examples (7)

Attack on digital signature: Results



The first 32 bits of the hash values are identical.

Experimental results

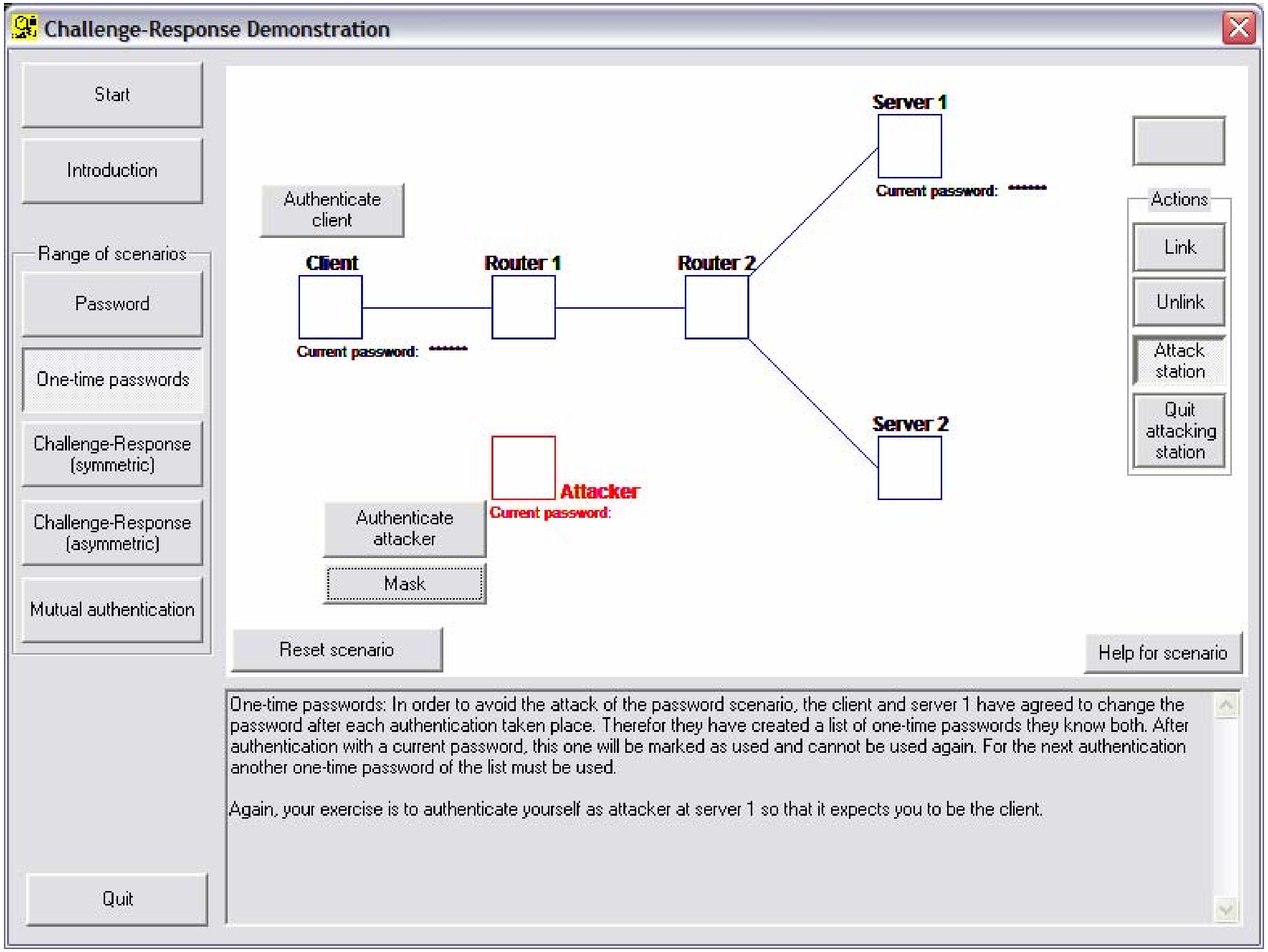
- 72 Bit *partial collision* (equality of the first 72 hash value bits) were found in a couple of days on a single PC.
- Signatures using hash values of up to 128 bit can be attacked today using massive parallel search!
- Use hash values of at least 160 bit length.

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



- 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 (at a hybrid encryption protocol)

Examples (9)

Idea for this side channel attack

- Ulrich Kühn, *Side-channel attacks on textbook RSA and ElGamal encryption* (2003)

Prerequisites:

- RSA encryption: $C = M^e \pmod N$ and decryption: $M = C^d \bmod N$.
- 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 $N = M * Z$ via $M = \lfloor |N/Z| \rfloor$

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

Null padding

M

$C = M^e \pmod N$

All bit positions for Z are successively calculated: For 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},

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

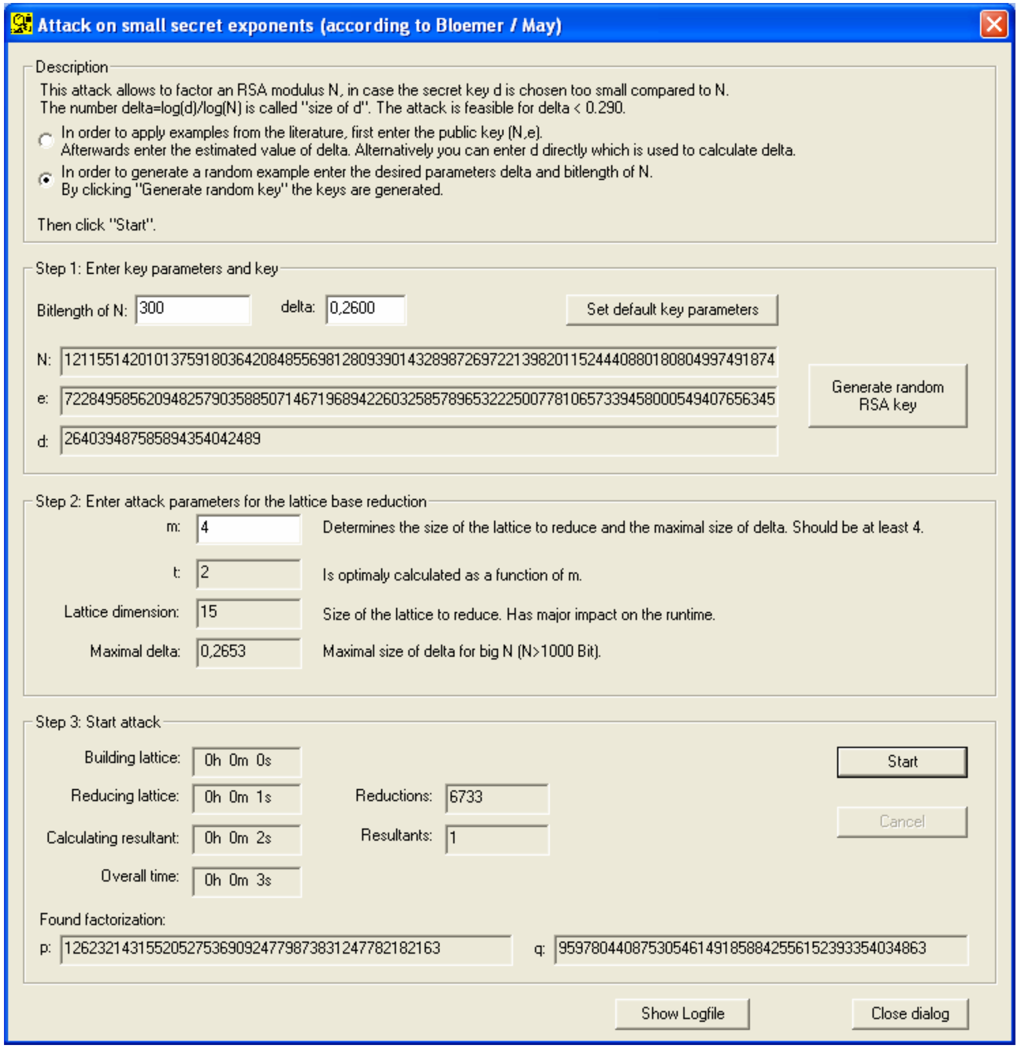
$M \cdot Z \cdot 2^{128}$

M

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

Examples (10)

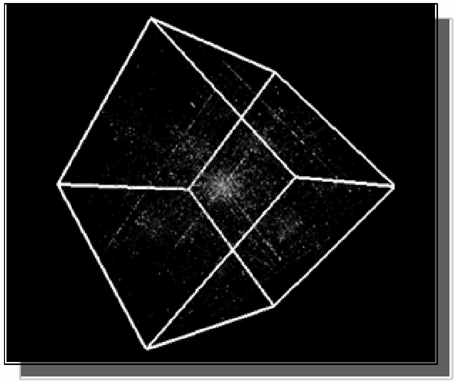
Mathematics: Attacks on RSA using lattice reduction



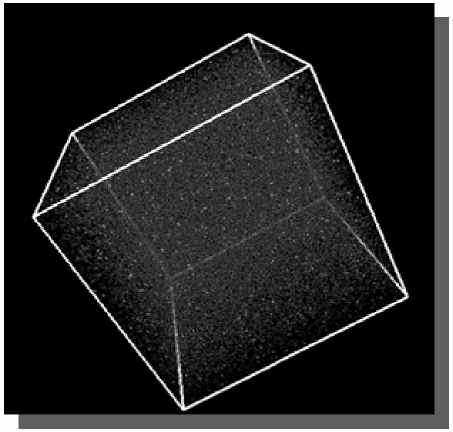
- Shows how the parameters of the RSA method have to be chosen, so that the algorithm resists the lattice reduction attacks described in current literature.
- **3 variants**
  1. The secret exponent d is too small in comparison to N.
  2. One of the factors of N is partially known.
  3. A part of the plaintext is known.
  - These assumptions are realistic.

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



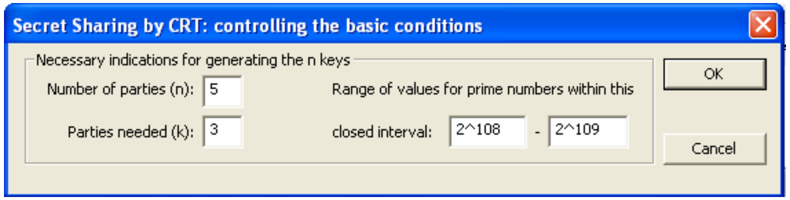
Example 2
- Generation of random numbers (menu „Indiv. Procedures“ \ “Tools” \ „Generate Random Numbers…“)
- It is recommended to generate at least 100.000 random bytes
- 3-D analysis using menu „Analysis“ \ „Analyse Randomness“ / „3-D Visualization…“
- Result: uniform distribution (no structures are recognisable)

Examples (12)

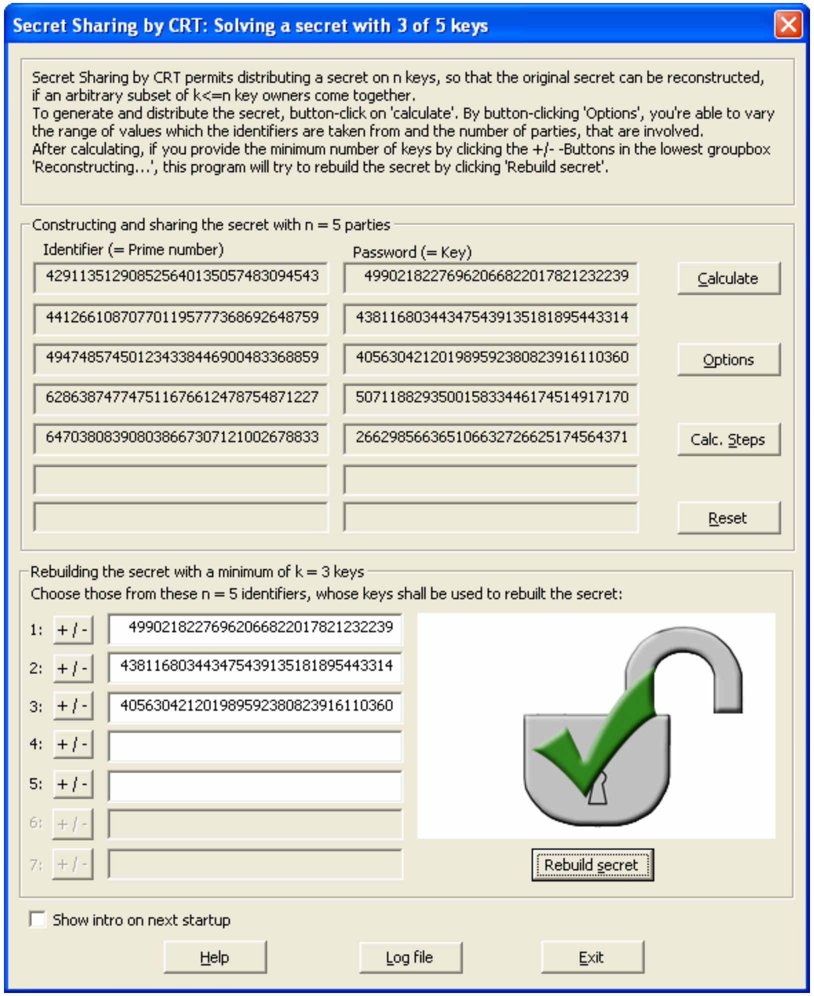
Secret sharing with CRT – Implementation of the chinese remainder theorem (CRT)

Secret sharing example (I):

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



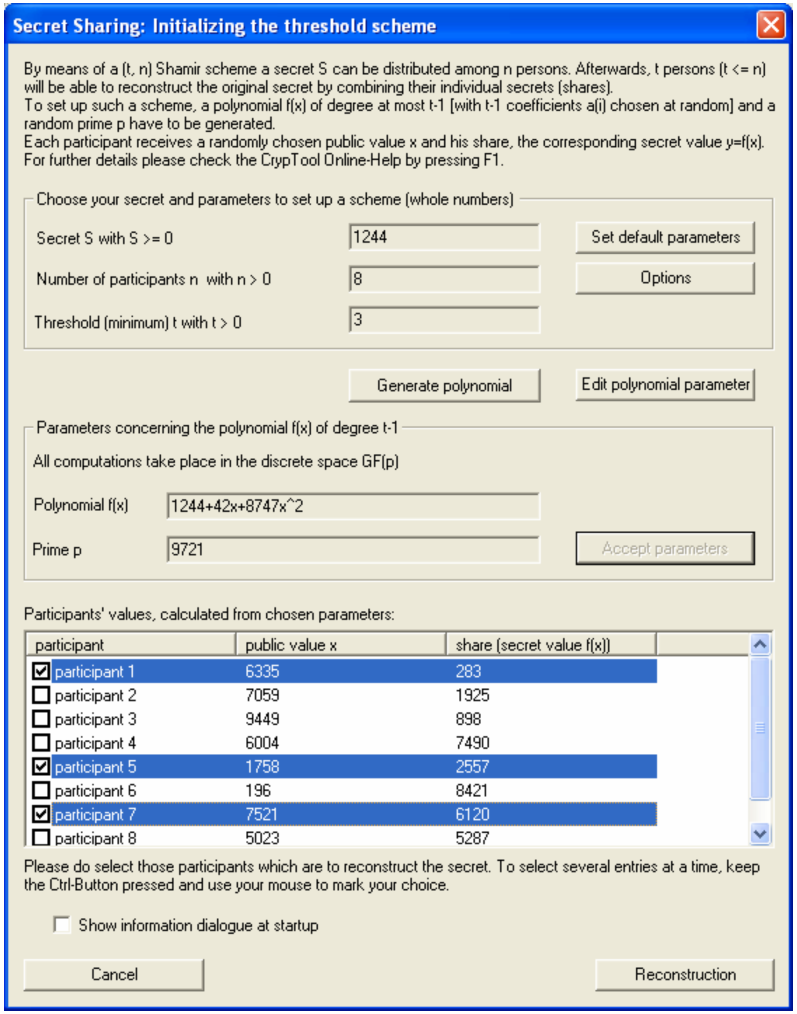
- „Calc. steps“ shows all steps to generate the key.



Examples (12)
Shamir secret sharing

Secret sharing example (II):

- 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
- CrypTool: Menu „Indiv. Procedures“ \ „Secret Sharing Demonstration (Shamir)…“
  1. Enter the secret K, number of persons n and threshold t
  2. Generate polynomial
  3. Use parameters
- Using „Reconstruction“ the secret can be restored

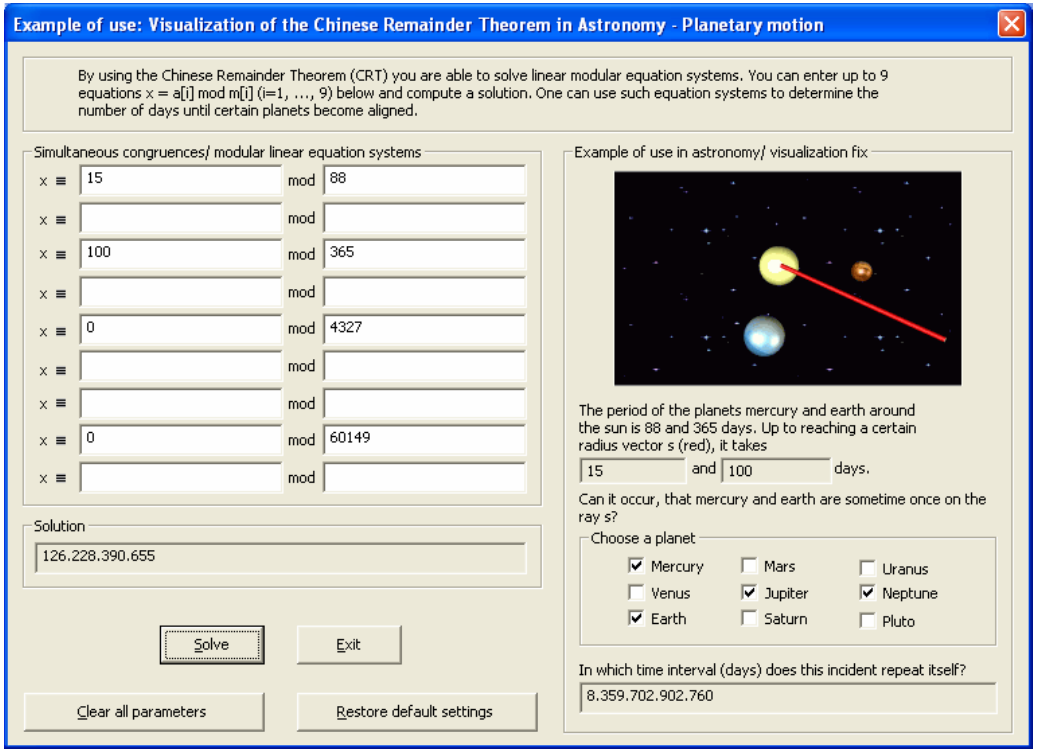


Examples (13)

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.



Examples (14)

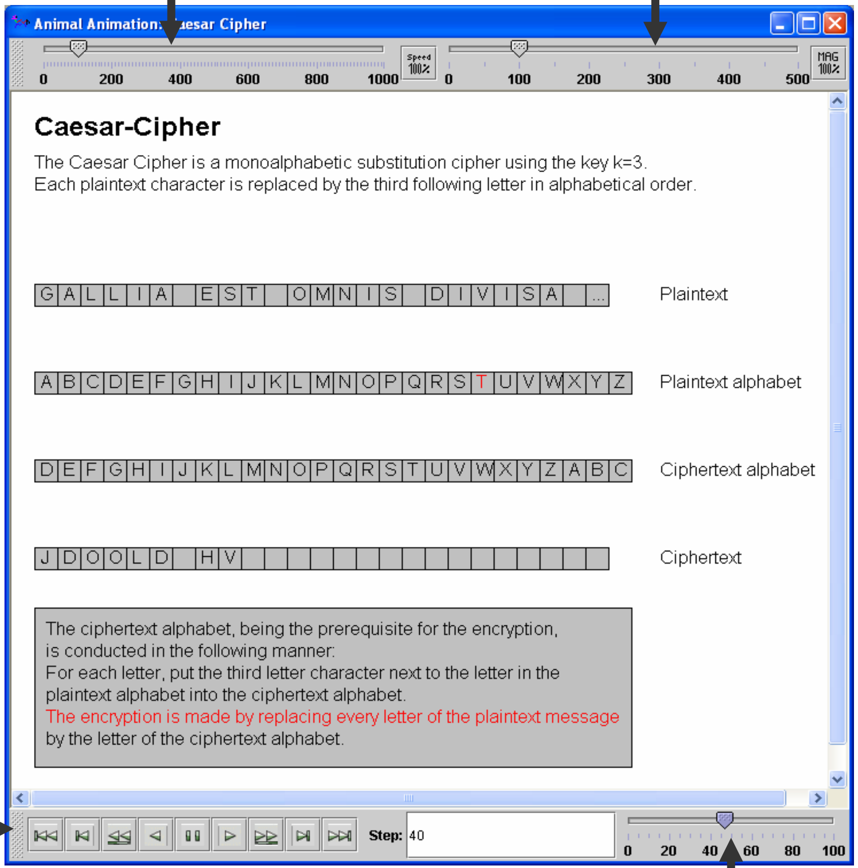
Visualisation of symmetric encryption methods using ANIMAL (1)

Animated visualisation of several symmetric algorithms

- Caesar
- Vigenère
- Nihilist
- DES

CryptTool

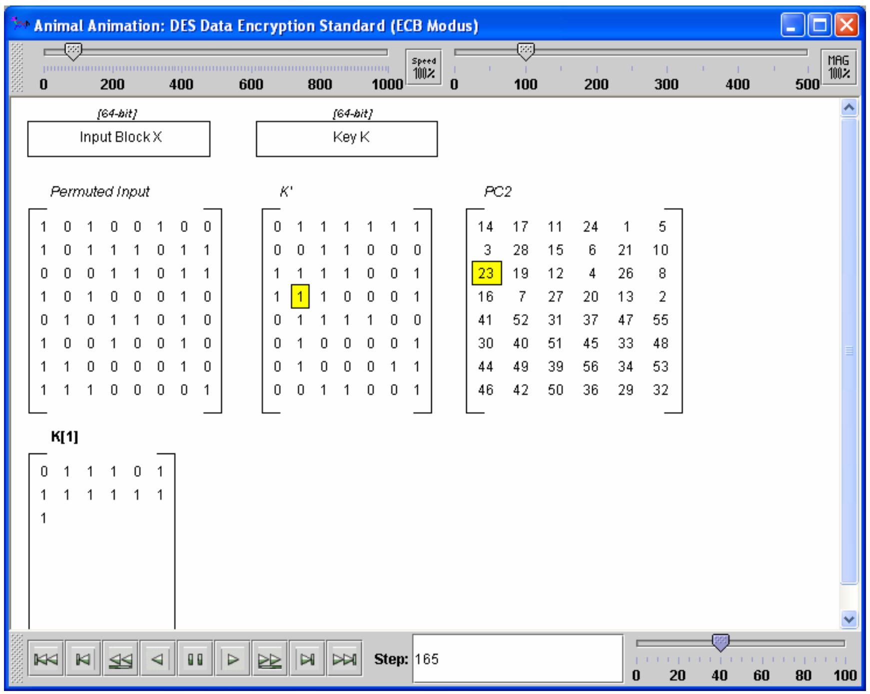
- Menu „Indiv. Procedures“ \ „Visualization of algorithms“ \ …
- Interactive animation control using integrated control center window.



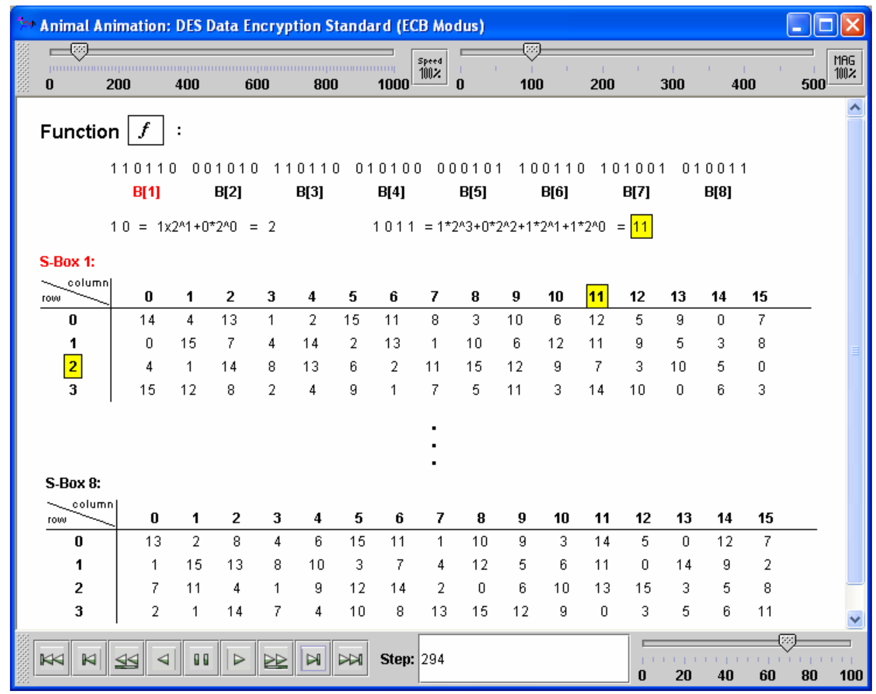
Examples (14)

Visualisation of symmetric encryption methods using ANIMAL (2)

- **Visualization of DES encryption**



After the permutation of the input block using the initialisation vector IV the key K is being permuted with PC1 and PC2.

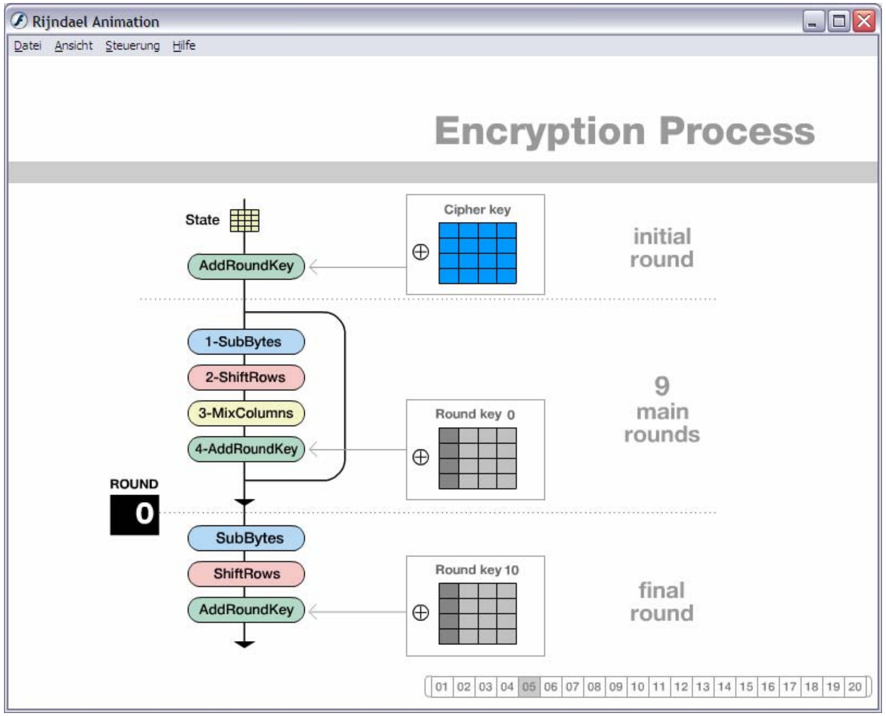


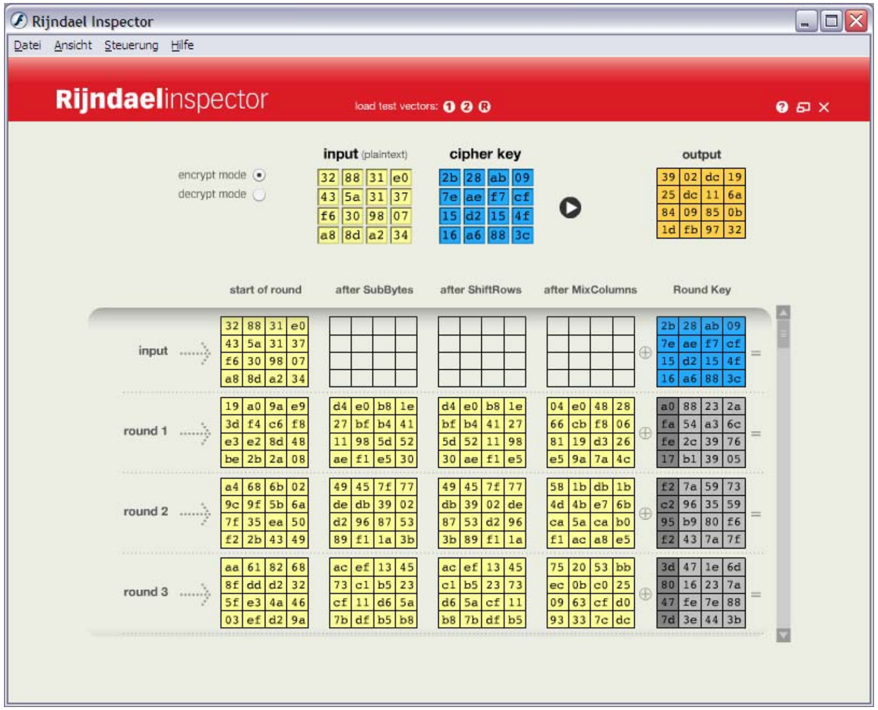
The core function *f* of DES, which links the right half of the block R_{i-1} with the partial key K_i.

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)

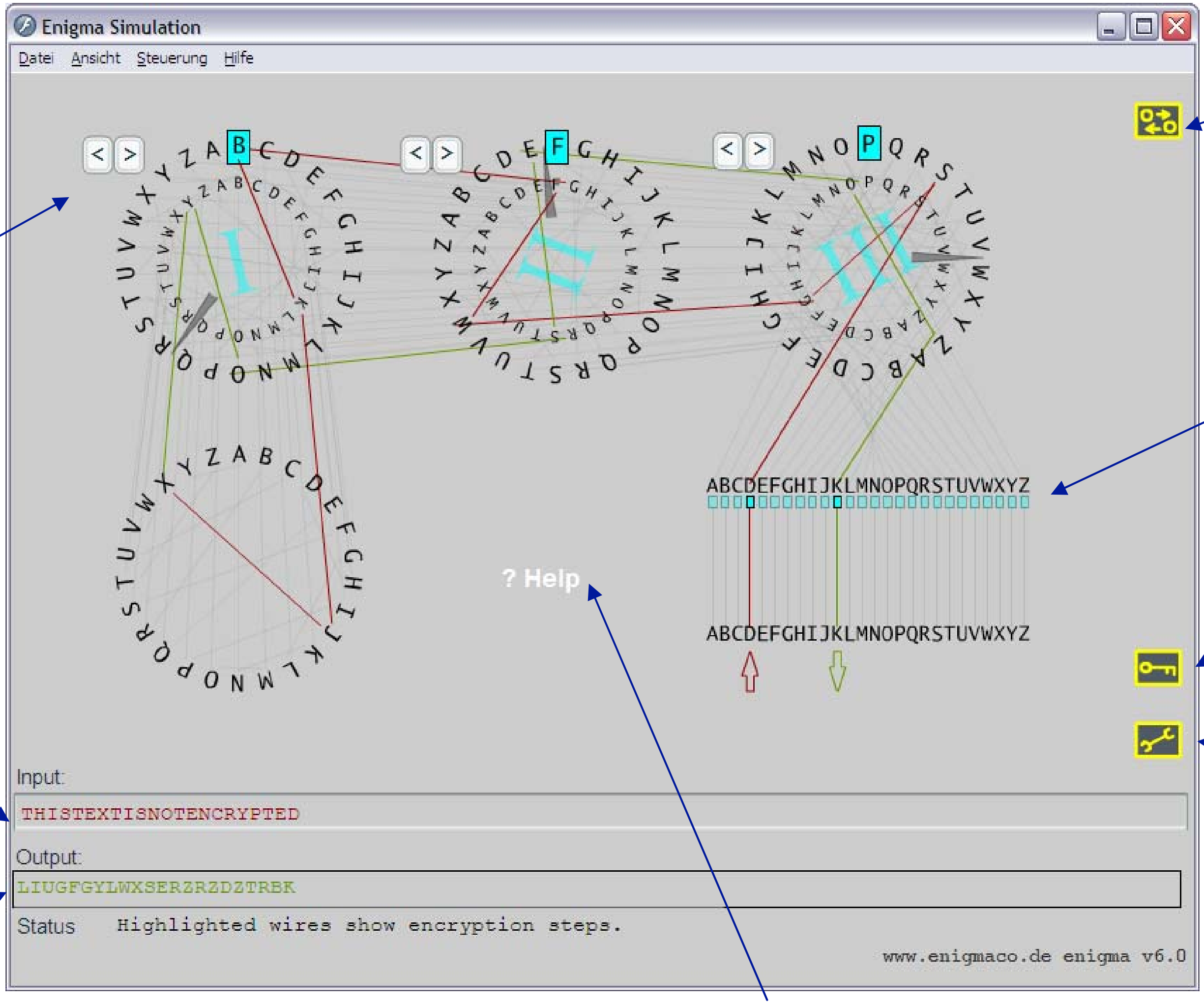




Menu “Indiv. Procedures” / “Visualization of Algorithms” / “AES” / “Rijndael Animation …” or “Rijndael Inspector …”

Examples (16)

Visualisation of the Enigma encryption



Select rotors

Change rotor setting

Change plugs

Show settings

Input of plaintext

Reset Enigma to initial state or random state

Output of encrypted text

Additional HTML online help

Examples (17)

Generation of a message authentication code

Message Authentication Code (MAC)

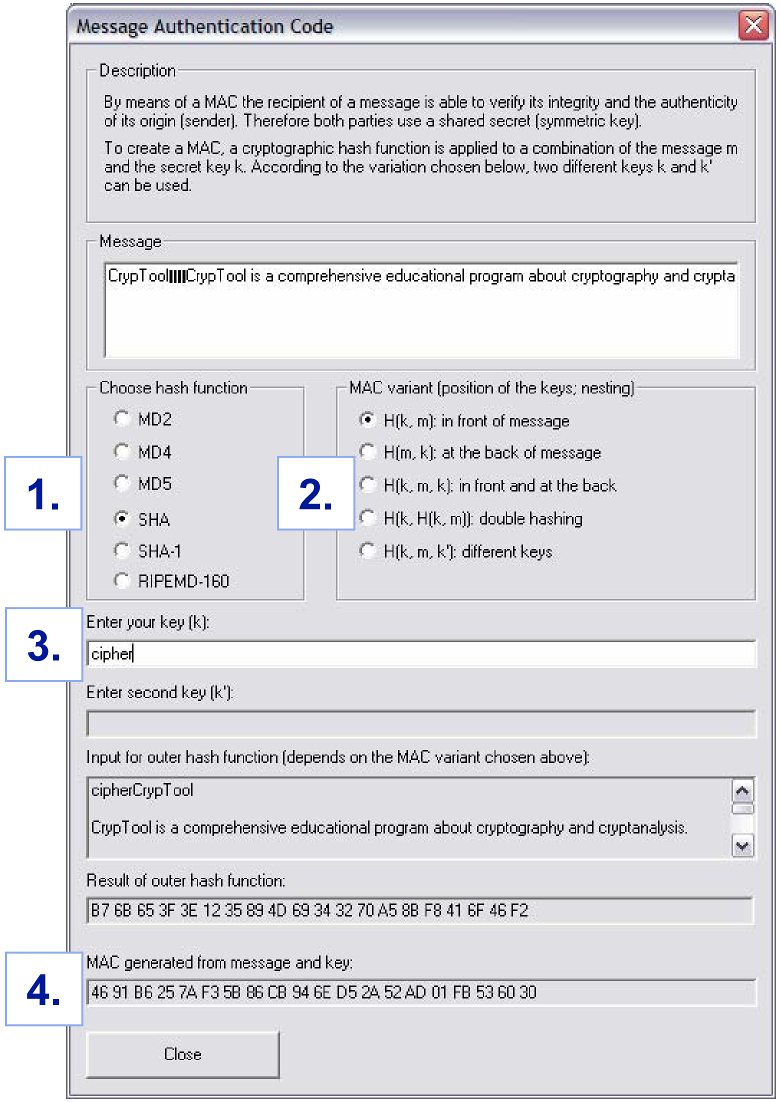
- Ensures integrity of a message
- Authentication of the message
- Basis: a common key

CrypTool

- Menu „Indiv. Procedures“ / „Hash“ / „Generation of MACs…“

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)



Examples (18)

Hash demonstration

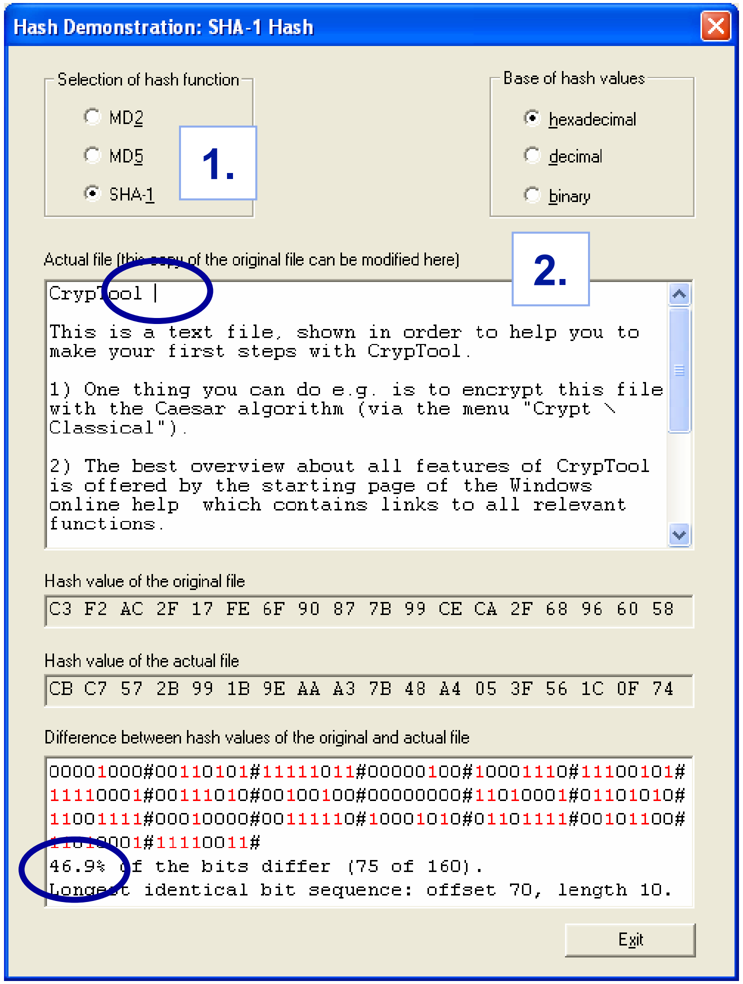
Sensitivity of hash functions to plaintext modifications

1. Select a hash function
2. Modification of characters in plaintext

Example:

Entering a blank after „CrypTool“ in the example text results in a 46,9% change of the bits of the generated hash value.

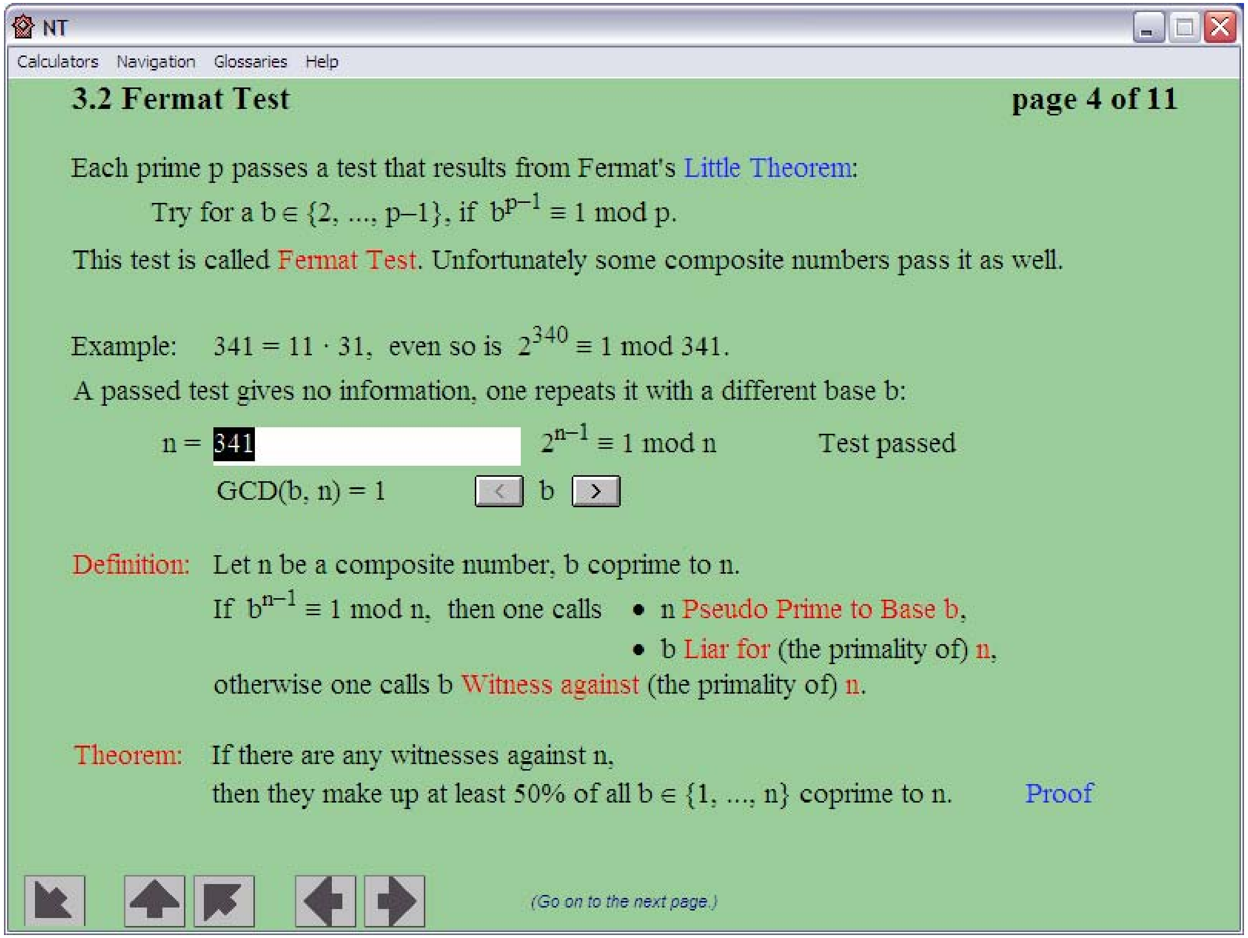
A good hash function should react sensitive to even the smallest change within the plaintext – *„Avalanche effect“* (small change, big impact).



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



Examples (20)

Point addition on elliptic curves

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

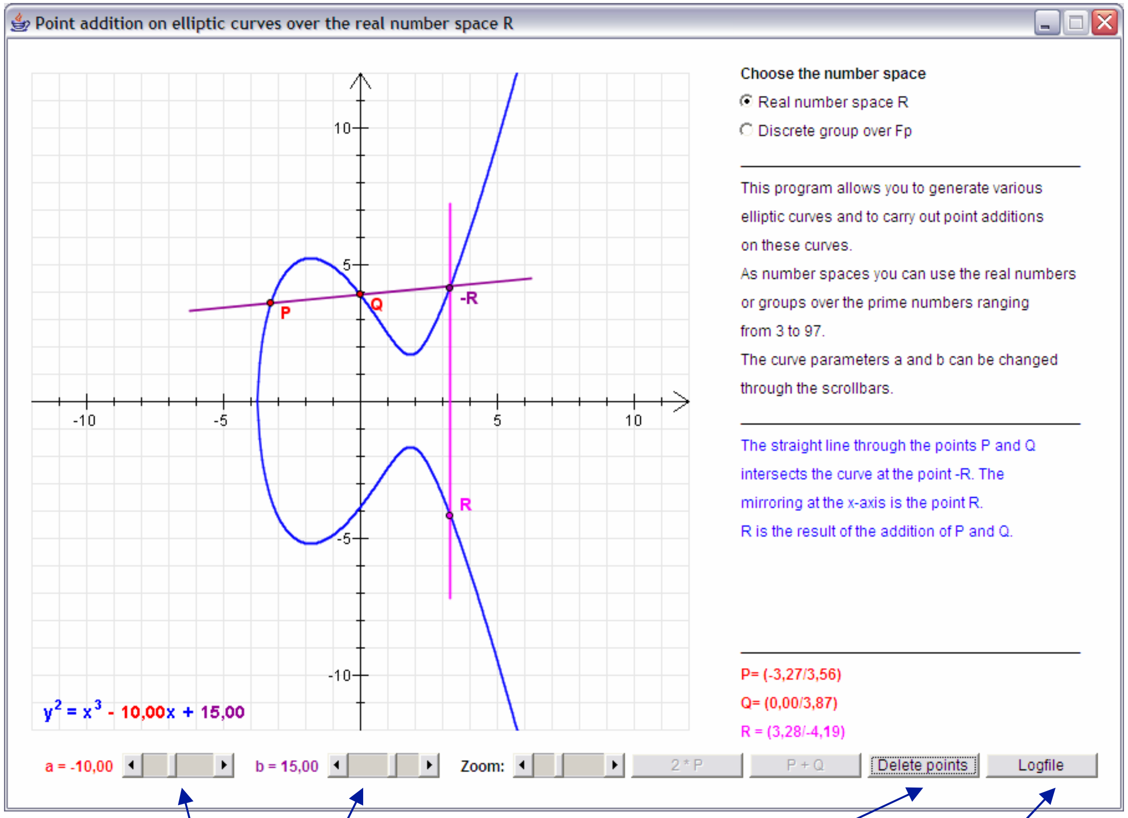
Menu „Individ. Procedures“ / „Number Theory - Interactive“ / „Point Addition on Elliptic Curves…“

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



Change curve parameters

Delete points

Logfile of calculations

Examples (21)

Password Quality Meter (PQM) I

- 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

Password: X40bTRds&11w_dks

Menu „Individ. Procedures“ / „Tools“ / „Password Quality Meter…“

Examples (21)
Password Quality Meter (PQM) II

- 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 chosen 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 I

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)

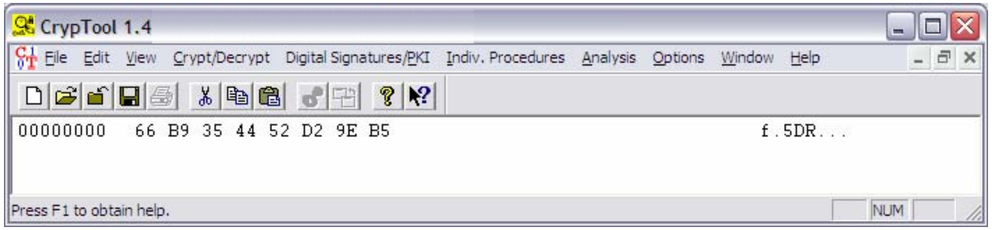
Key (hex)	Encrypted text (hex)
68ac78dd40bbefd*	66b9354452d29eb5
0123456789ab****	1f0dd05d8ed51583
98765432106*****	bcf9ebd1979ead6a
0000000000******	8cf42d40e004a1d4
000000000000****	0ed33fed7f46c585
abacadaba*******	d6d8641bc4fb2478
dddddddddd******	a2e66d852e175f5c

Examples (22)

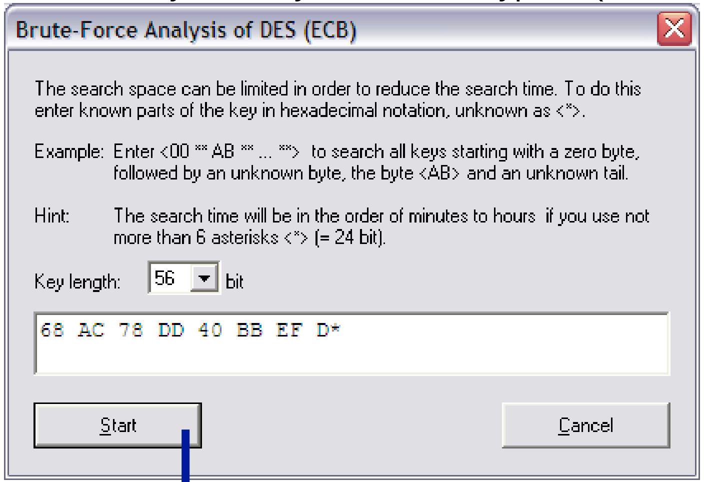
Brute-force analysis II

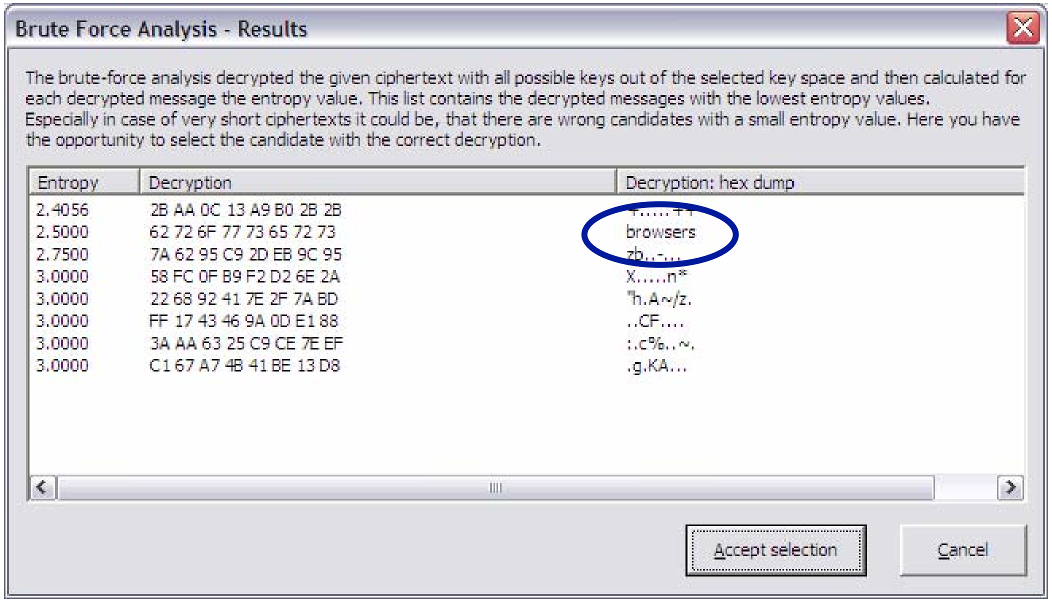
1. Input of encrypted text (as hex)
2. Use brute-force analysis
3. Input partly known key
4. Start brute-force analysis
5. Analysis of the results: Low entropy as evidence of a possible decryption. However, because a very short plaintext has been used in this example, the correct result does not have the lowest entropy.

Menu „View“ / „Show as HexDump“



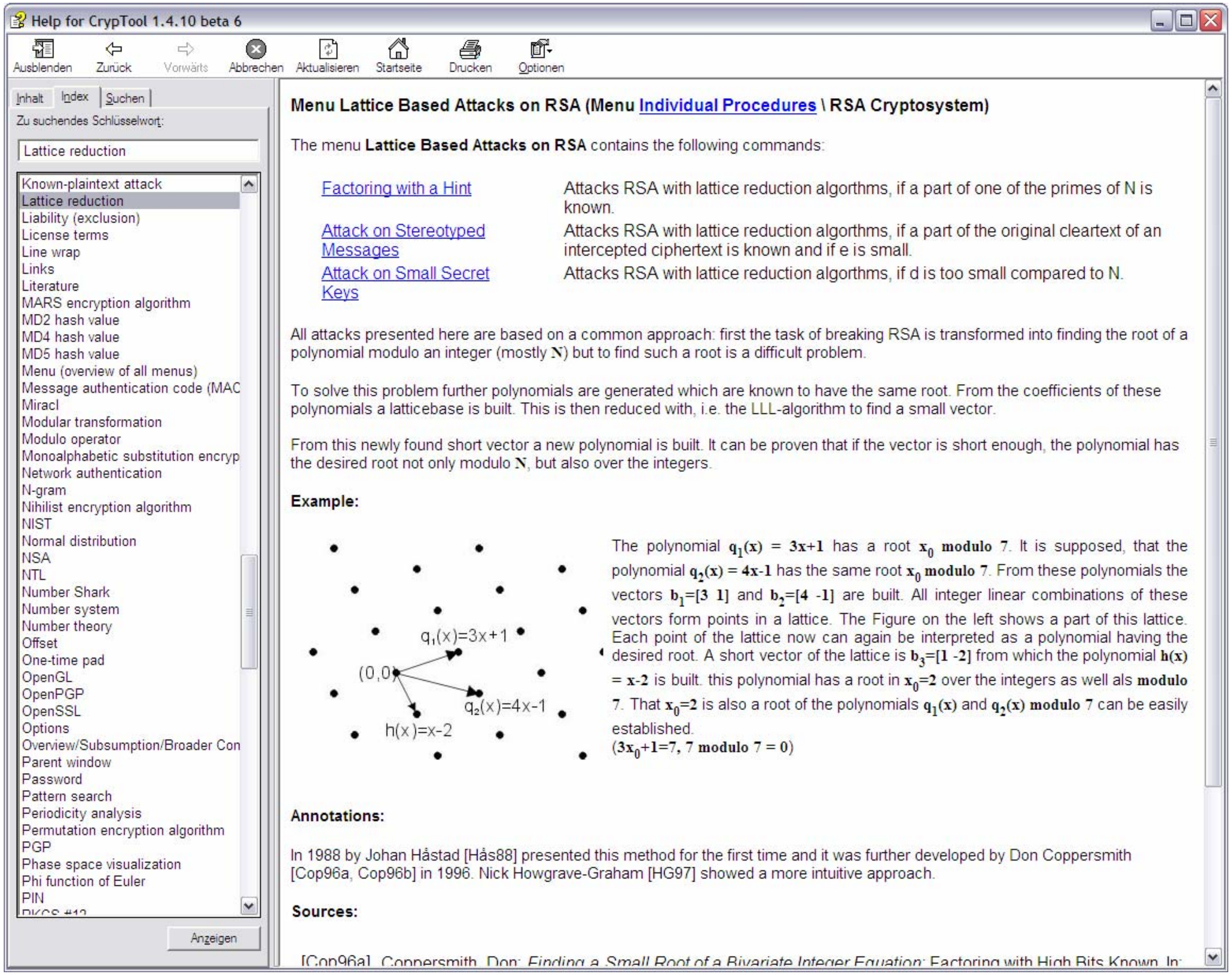
Menu “Analysis” \ “Symmetric Encryption (modern)” \ “DES (ECB)…”





Examples (23)

CrypTool Online Help I



Examples (23)

CrypTool Online Help II

Help for CrypTool 1.4.10 beta 6

Ausblenden | Zurück | Vorwärts | Abbrechen | Aktualisieren | Startseite | Drucken | Optionen

Inhalt | Index | Suchen

Zu suchendes Schlüsselwort:

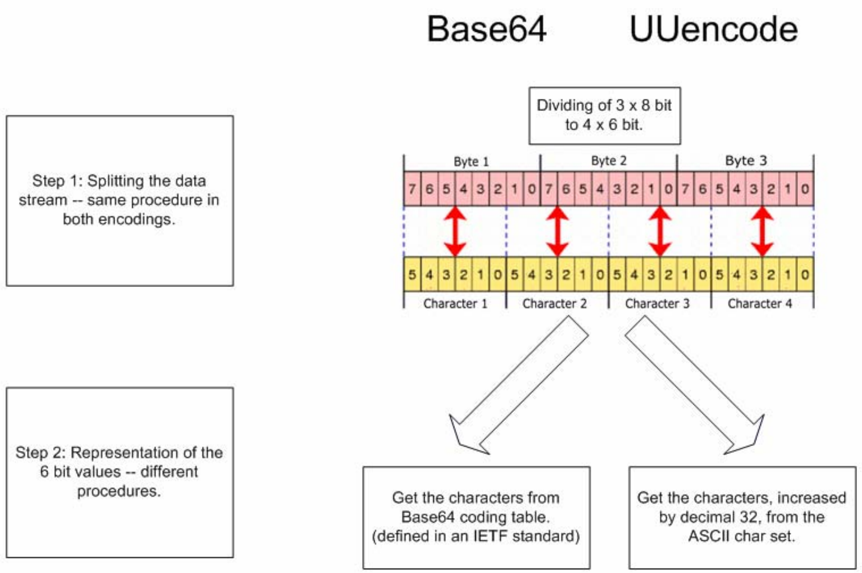
base

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

Anzeigen

Comparison of Base64 and UU coding

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



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.
  ○ No problems with 7-bit char set restrictions.
  ○ 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

Future CrypTool Development

Planned after release 1.4.10 (see readme file)

- Mass pattern search
- Visualisation of interoperability of S/MIME and OpenPGP formats
- Visualisation of SSL protocol
- Demonstration of visual cryptography
- Integration of crypto library crypto++ from Wei Dai
- Demonstration of Bleichenbacher's RSA signature forgery
- Demonstration of virtual credit card numbers as an approach against credit card abuse

In Progress (see readme file)

- Port of existing C++ version to Linux using Qt4
(see: http://www.cryptoolinux.net/)
- Port and redesign of CrypTool in Java / SWT / Eclipse / RPC
(see: http://jcryptool.sourceforge.net/)
- Port and redesign of C++ version with C# / WPF/Vista / VS2005 / .NET
(successor of current release: tree view instead of menus, …)

Future plans (see readme file)

- Visualisation of protocols (e.g. Kerberos)
- Visualisation of attacks on these protocols
- Command line interface for batch processing
- Additional parameters for existing methods / algorithms
- Graphical design oriented mode for beginners plus expert mode

CrypTool as a Framework for Own Developments

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 VC++ , Perl,
Subversion source code management

- Until CrypTool 1.3.05: Visual C++ 6.0 only (was available within books for free)
- CrypTool 1.4.10: 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 environment

- For versions after 1.4.10:
  - C# version: .NET with Visual Studio 2005 Express Edition (free), WPF (no more MFC) and Perl
  - Java version: with Eclipse 3.2, SWT, RCP (free)
  - C++ version for Linux with Qt 4.x, GCC 4.0 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 as well as for the new projects for CrypTool 2.0: Java project and C# project!
    - 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).

Contact

Bernhard Esslinger

University of Siegen
University lecturer, 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

additional contacts: See readme within the CrypTool folder
mailing list: cryptool-list@sec.informatik.tu-darmstadt.de

Additional Literature

- Simon Singh, *“The Codebook”*, 1999, Doubleday [English]
- Simon Singh, *“Geheime Botschaften”*, 2000, Hanser [German]
- U. Ulfkotte, *“Wirtschaftsspionage“*, 2001, Goldmann [German]
- 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
- Additional cryptography literature
  (e.g. by Wätjen, Buchmann, 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)