

CRYPTOLOGY WITH CRYPTOOOL v 1.4.20

Introduction to Cryptography und Cryptanalysis

Scope, Technology and Future of CryptTool

www.cryptool.com

www.cryptool.de

www.cryptool.org

www.cryptool.pl

www.iec.csic.es/cryptool

Prof. Bernhard Esslinger and CryptTool-Team, 2008

Content (I)

I. CrypTool and Cryptology – Overview

1. The CrypTool-Project
2. Relevance of cryptography and examples of classical encryption methods
3. 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
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)



Content (II)

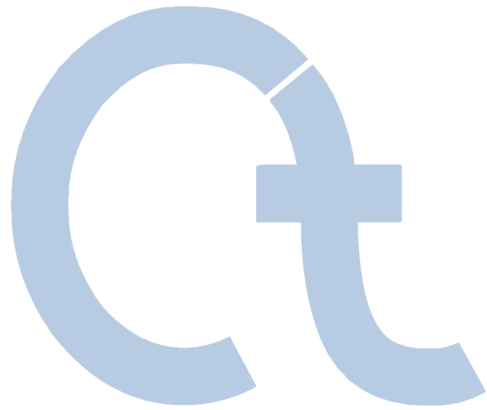
III. Examples

10. [RSA attack 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 encryption 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](#)

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!



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”



Ministerium für Innovation,
Wissenschaft, Forschung
und Technologie des Landes
Nordrhein-Westfalen



- **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).

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.

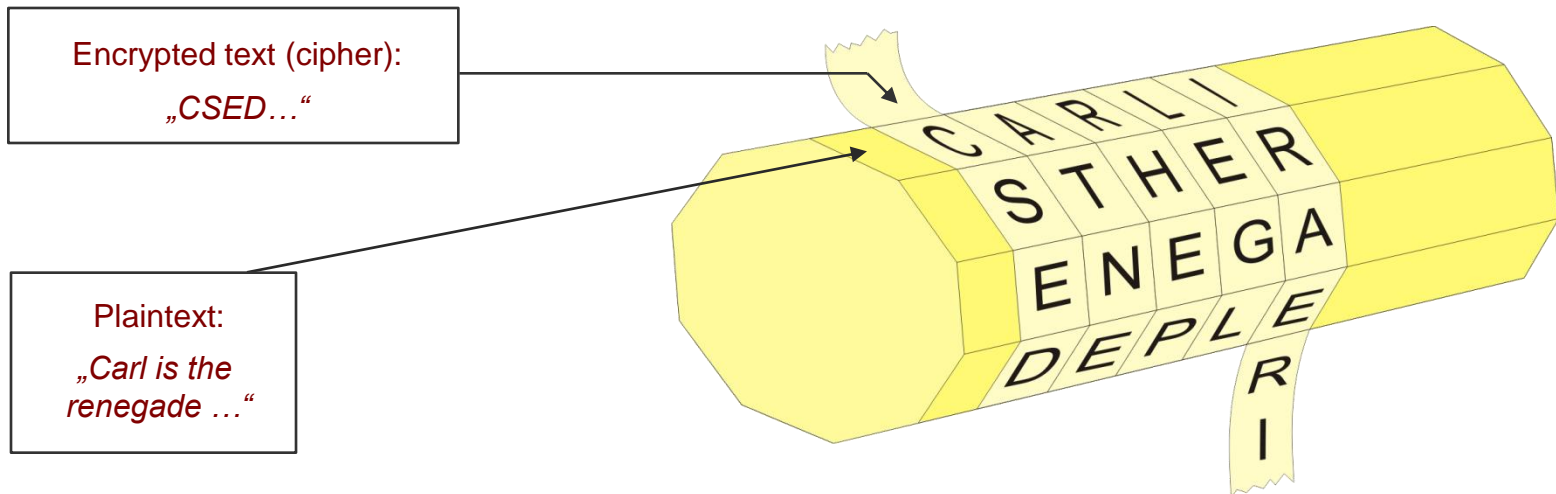
■ Non-Repudiation

- 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.

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)



Examples of Early Cryptography (2)

Symmetric Caesar encryption

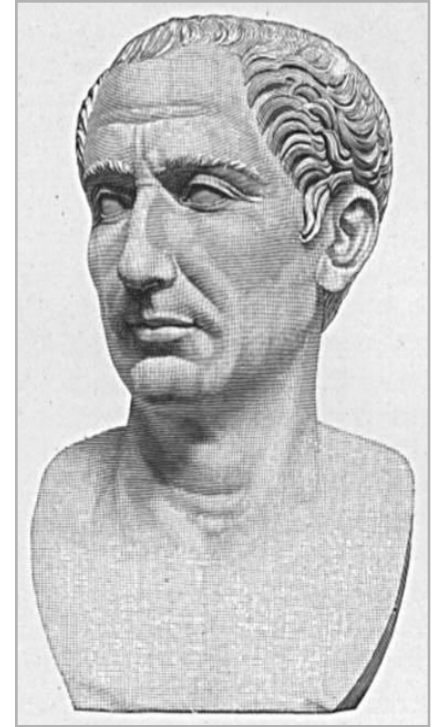
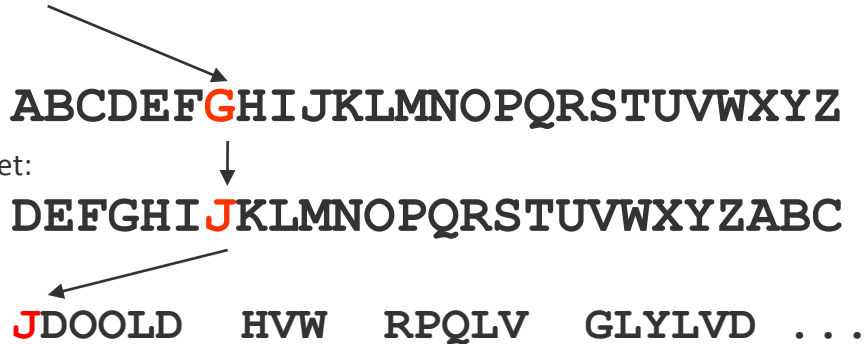
- **Caesar encryption** (Julius Caesar, 100 - 44 B.C.)
- Simple substitution cipher

Plaintext: **G**ALLIA EST OMNIS DIVISA ...

Secret alphabet: **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

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

JDOOLD HVW RPQLV GLYLVD ...



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

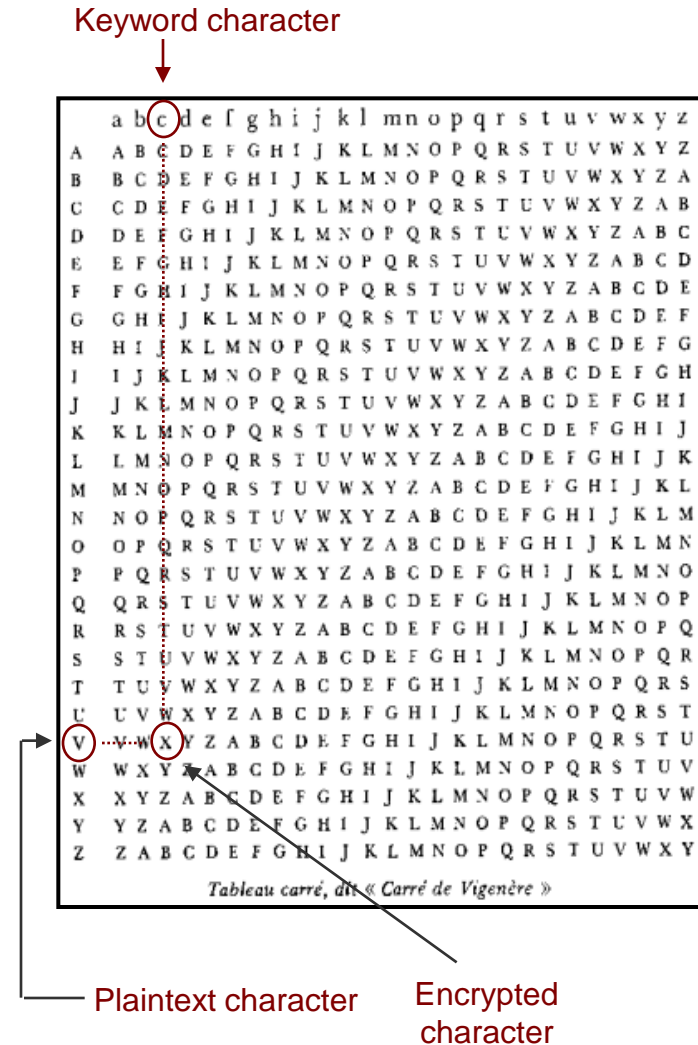
Presentation with CrypTool via the following menus:

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

Examples of Early Cryptography (3)

Symmetric Vigenère encryption

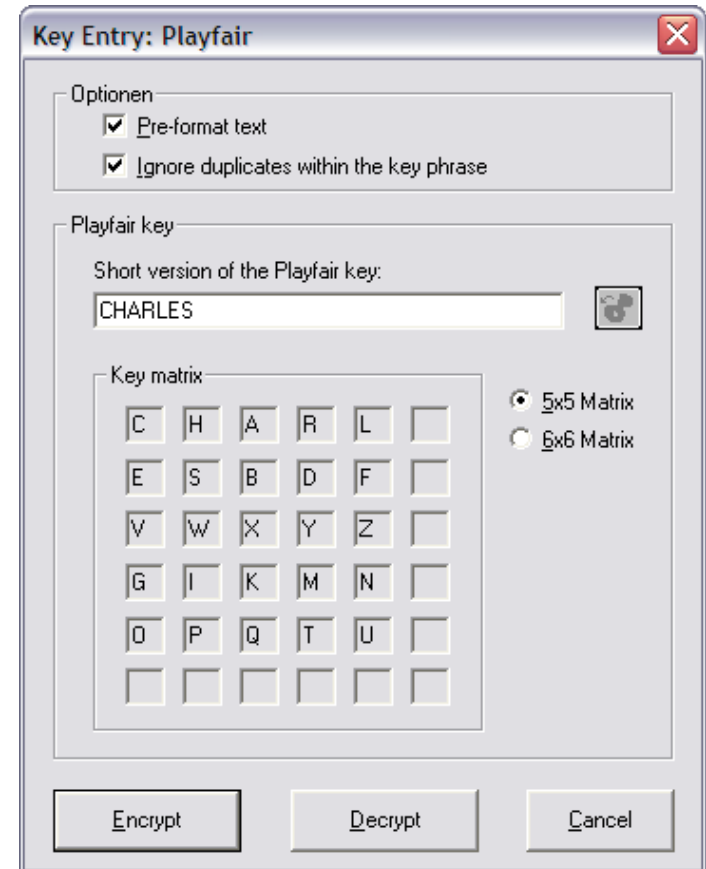
- **Vigenère-Encryption** (Blaise de Vigenère, 1523-1596)
- Encryption with a key word using a key table
- Example:
Keyword: **CHIFFRE**
Encrypting: **VIGENERE** becomes **XPOJSVVG**
- The plaintext character (V) is replaced by the character in the corresponding row and in the column of the first key 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.



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)



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

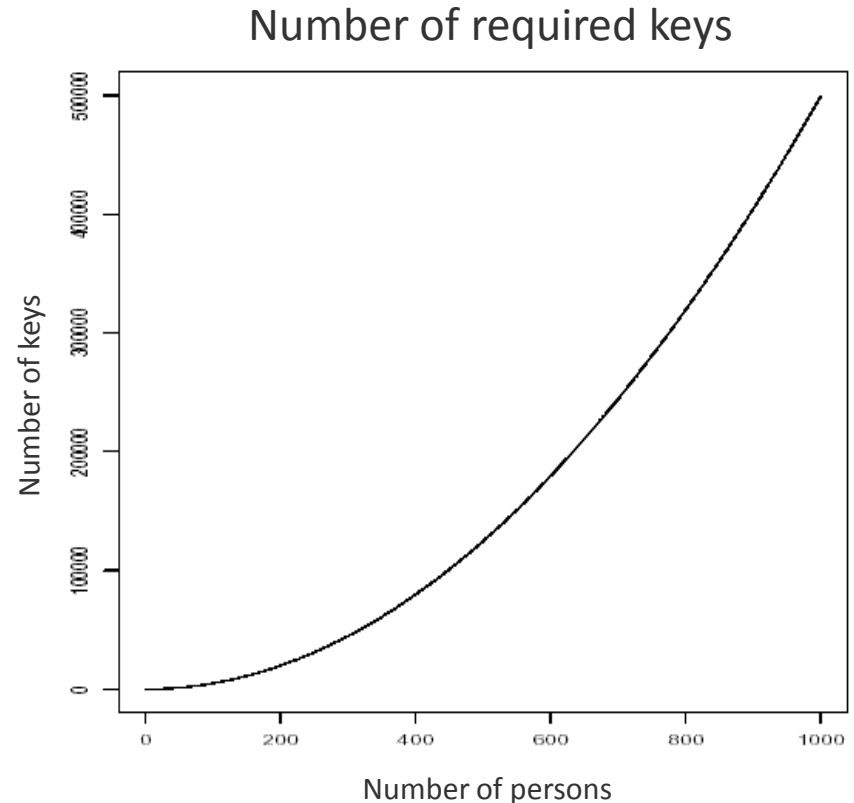
$n = 100$ persons require

$S_{100} = 4.950$ keys; and

$n = 1.000$ persons require

$S_{1000} = 499.500$ keys.

⇒ factor 10 more persons, factor 100 more keys



Cryptography – Important Insights (2)

Solving the key distribution problem through asymmetric cryptography

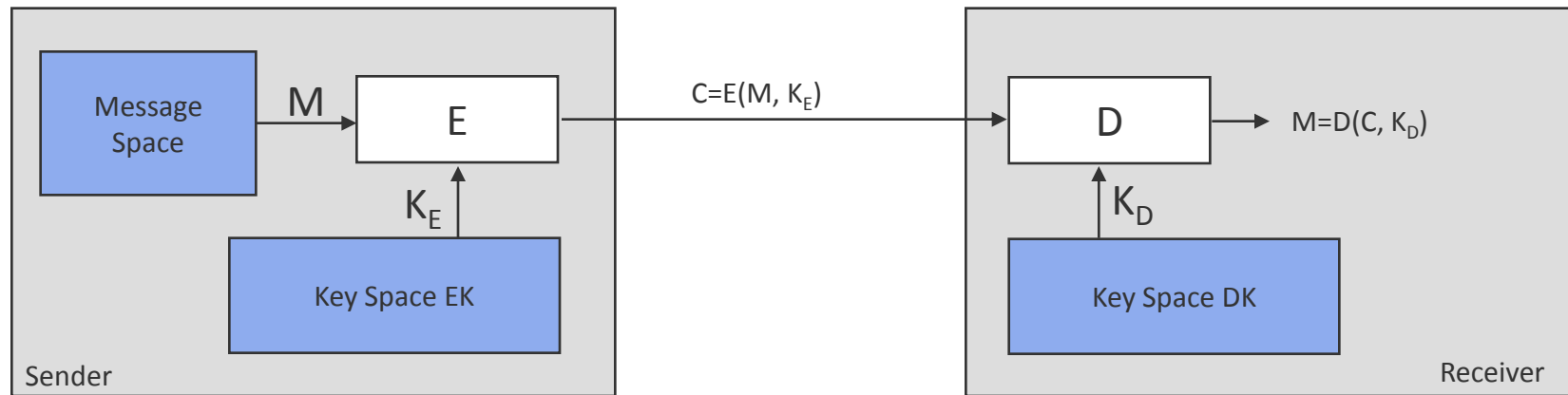
Asymmetric cryptography

- 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!

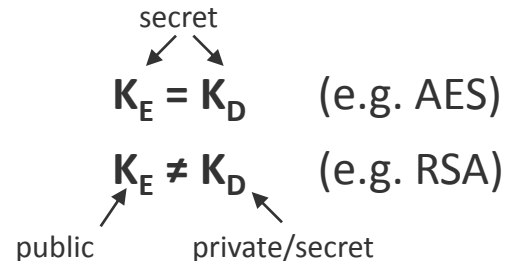
Encryption and Decryption

Symmetric und asymmetric encryption



a) Symmetric Encryption:

b) Asymmetric Encryption:



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
 - 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: $C + D = G$ ($3 + 4 = 7$)

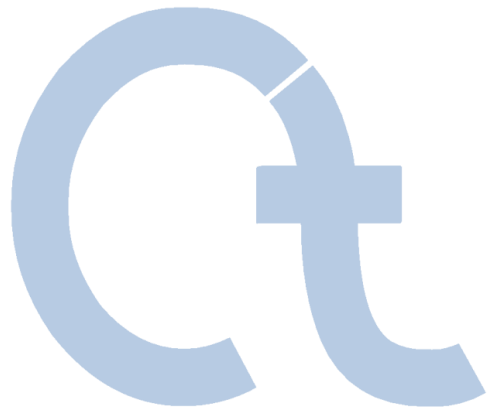
Vigenère: - $CAT + DOG = FOZ$ [$(2,0,19)+(3,14,6)=(5,14,25)$]

- $"Hund" + "Katze" = "RUGCLENWGYXDATRNNHNMH"$

- *Vernam (OTP)*

- *AES (output key, brute-force analysis)*

Content



I. CrypTool and Cryptology –
Overview

II. CrypTool Features?

III. Examples

IV. Project / Outlook / Contact

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
- Manual (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
- 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 (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
- $x^2 \bmod n$
- 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)
- a lot more ...

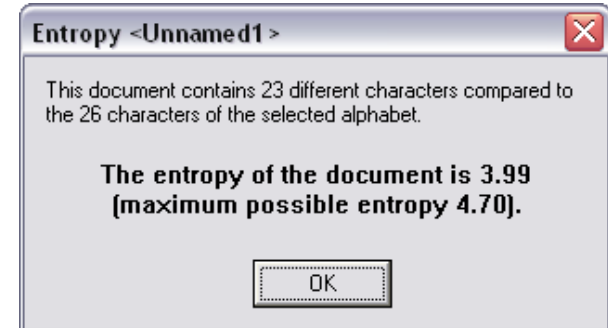
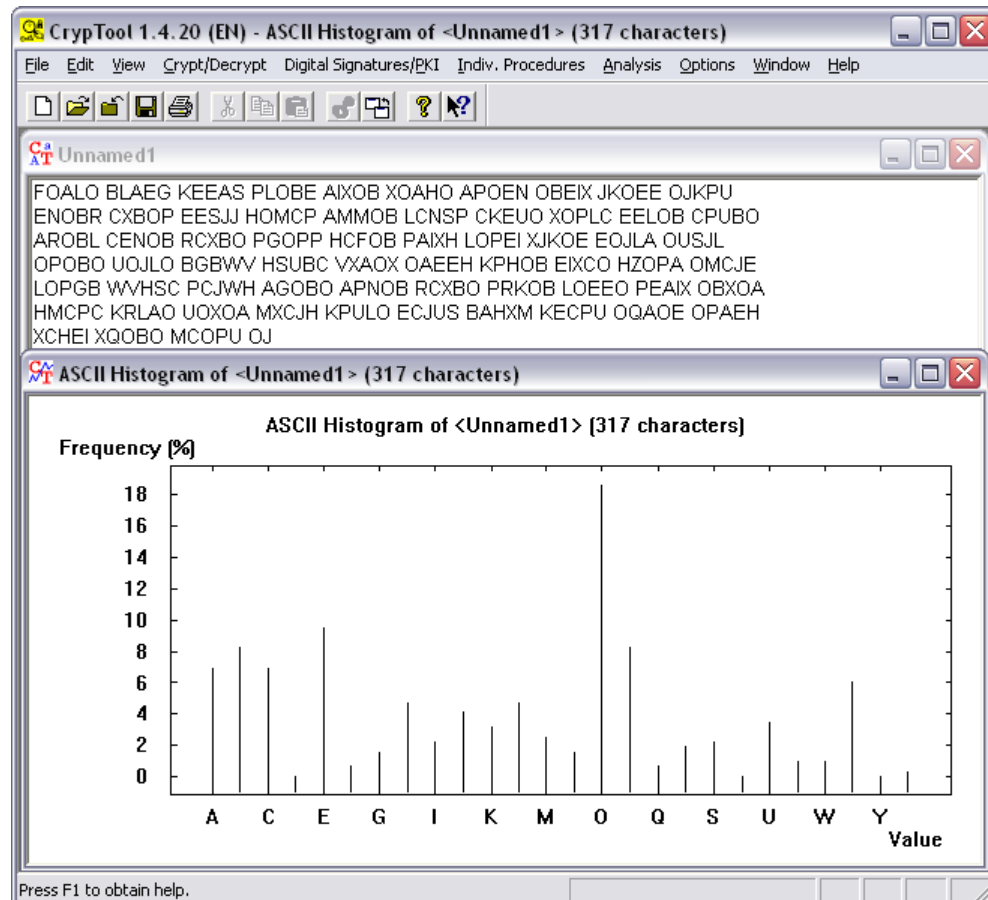


Language Structure Analysis

Language analysis options available in CrypTool

Number of characters, n-gram, entropy

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



N-Gram List of Unnamed1

Selection

☒ Histogram
☐ Digram
☐ Trigram
☐ 4 -gram

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

Text options

Determine list

Save list

Close

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)

*Demonstration in
CrypTool*


Vigenère analysis

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

48-digit decimal number

=

3 * 1129 * 6353 * 1159777 * 22383173213963 * 567102977853788110597

Factorisation example 2:

$2^{250} - 1$

75-digit decimal number

=

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

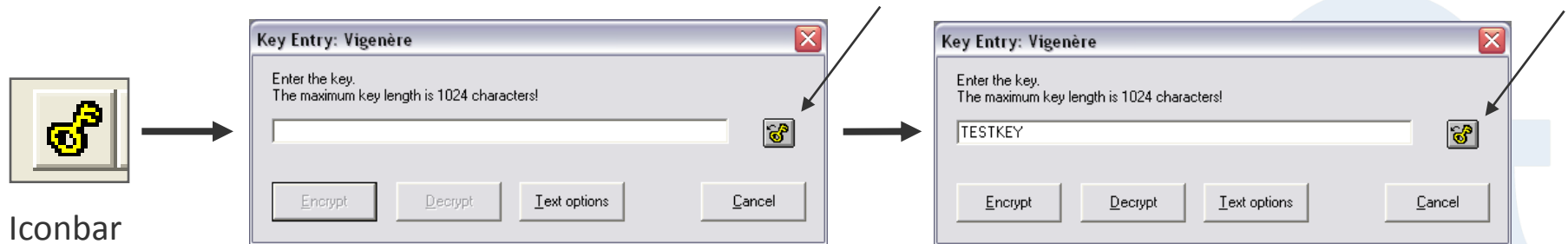
Concepts for a User-Friendly Interface

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 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).



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

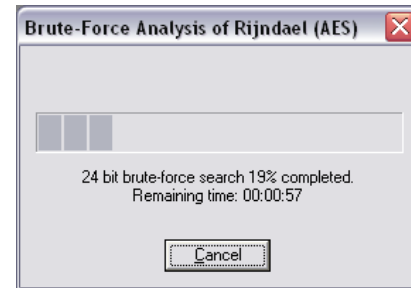
- Floyd algorithm (mappings to locate hash collisions) or factorisation with quadratic sieve

4. Time measurement and estimates

- Display of elapsed time while using brute-force

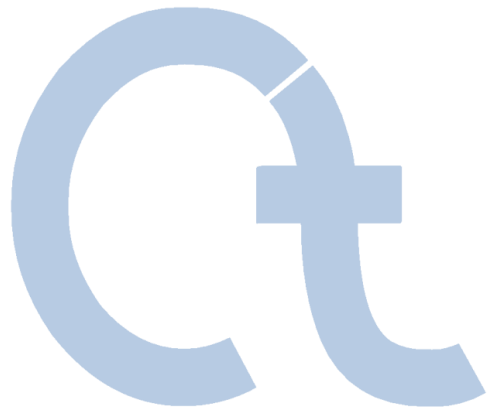
5. Reusability / Integration

- 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)

Content



I. CrypTool and Cryptology – Overview

II. CrypTool Features

III. Examples

IV. Project / Outlook / Contact

CrypTool Examples

Overview of examples

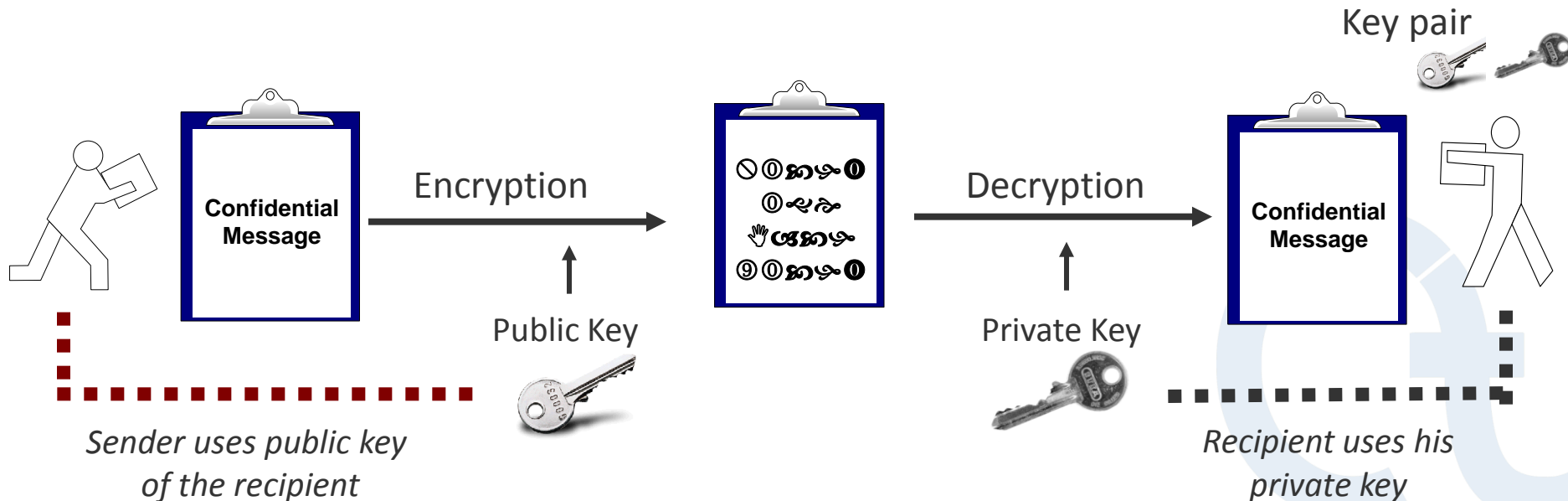
1. [Encryption with RSA / Prime number test / Hybrid encryption and digital certificates](#)
2. [Digital signature visualised](#)
3. [Attack on RSA encryption \(modul N too short\)](#) Attack on RSA encryption (modul N too short)
4. [Analysis of encryption in PSION 5](#)
5. [Weak DES keys](#)
6. [Locating 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 $n = p \cdot q$;

d is calculated under the constraints $\gcd(\phi(n), e) = 1$; $e \cdot d \equiv 1 \pmod{\phi(n)}$.

Encryption and decryption operation: $(m^e)^d \equiv m \pmod{n}$

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

Procedure :

- Transformation of message in binary representation
- Encrypt message $m = m_1, \dots, m_k$ 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.

The screenshot shows a window titled "Prime Number Test" with a close button (X) in the top right corner. Inside the window, there is a text box explaining that various methods exist to check if a number is prime, including probabilistic methods (fast but not 100% correct) and deterministic methods (100% correct). Below this, a section titled "Algorithms for prime number test" contains four radio buttons: "Miller-Rabin Test" (selected), "Fermat Test", "Solovay-Strassen Test", and "AKS Test (deterministic procedure)". Underneath, a section titled "Prime number test" contains a "Load number from file" button, a "Number to test" input field with the value "2^255-1", and a "Result" field. The "Result" field has a red "X" icon to its left and contains a long string of digits: "5789604461865809771178549250434395392663499233282028201972879200". At the bottom of the window, there are two buttons: "Test number" and "Cancel".

Examples (1)

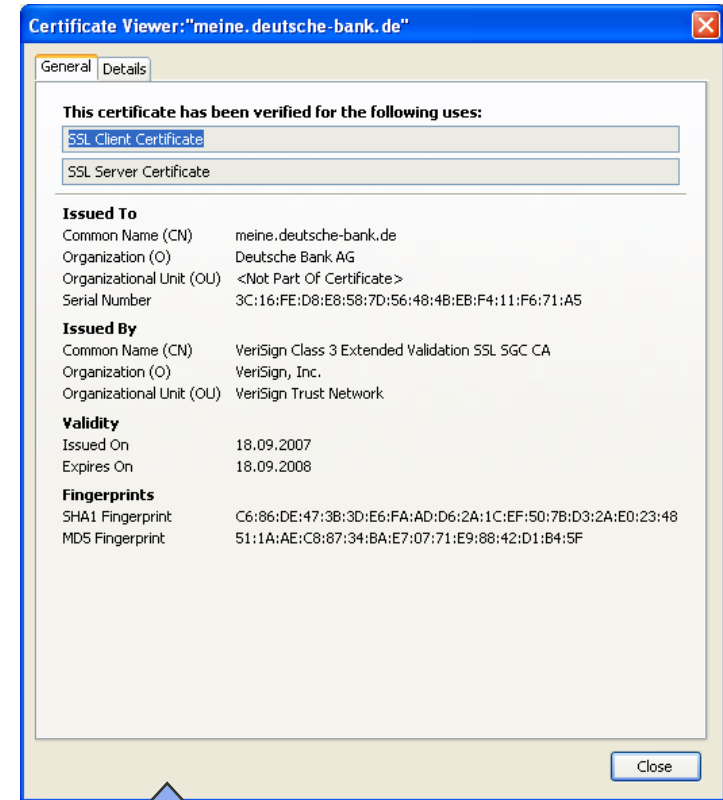
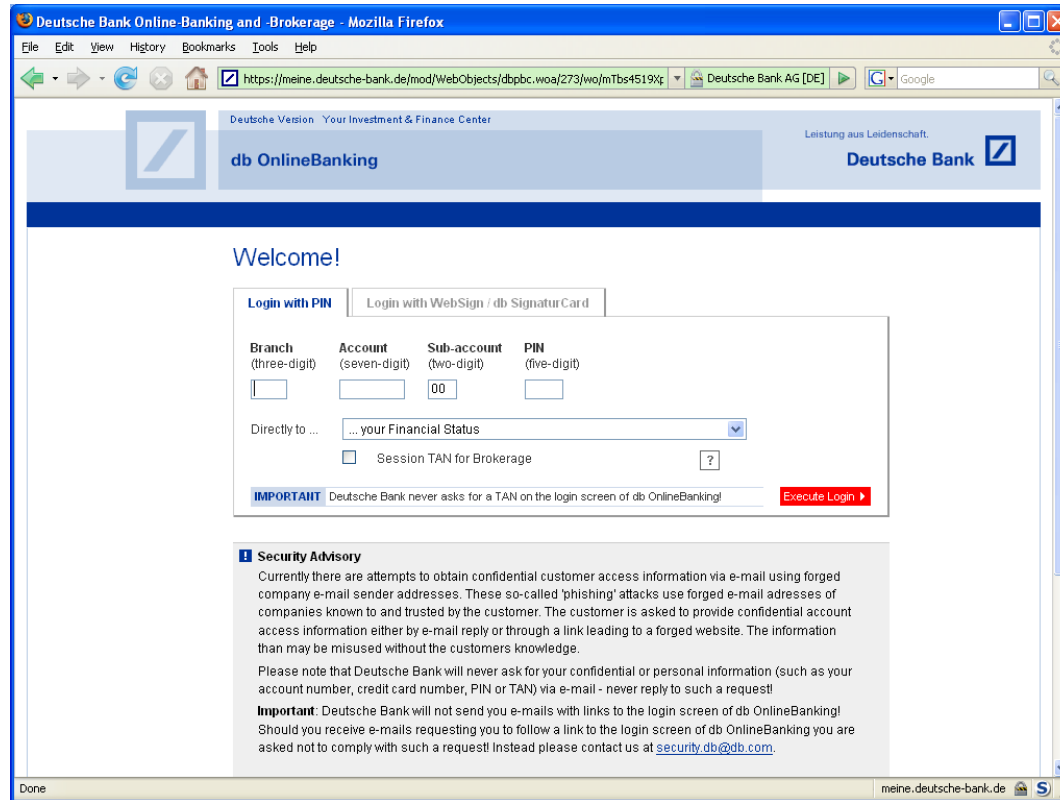
Hybrid encryption and digital certificates

- **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

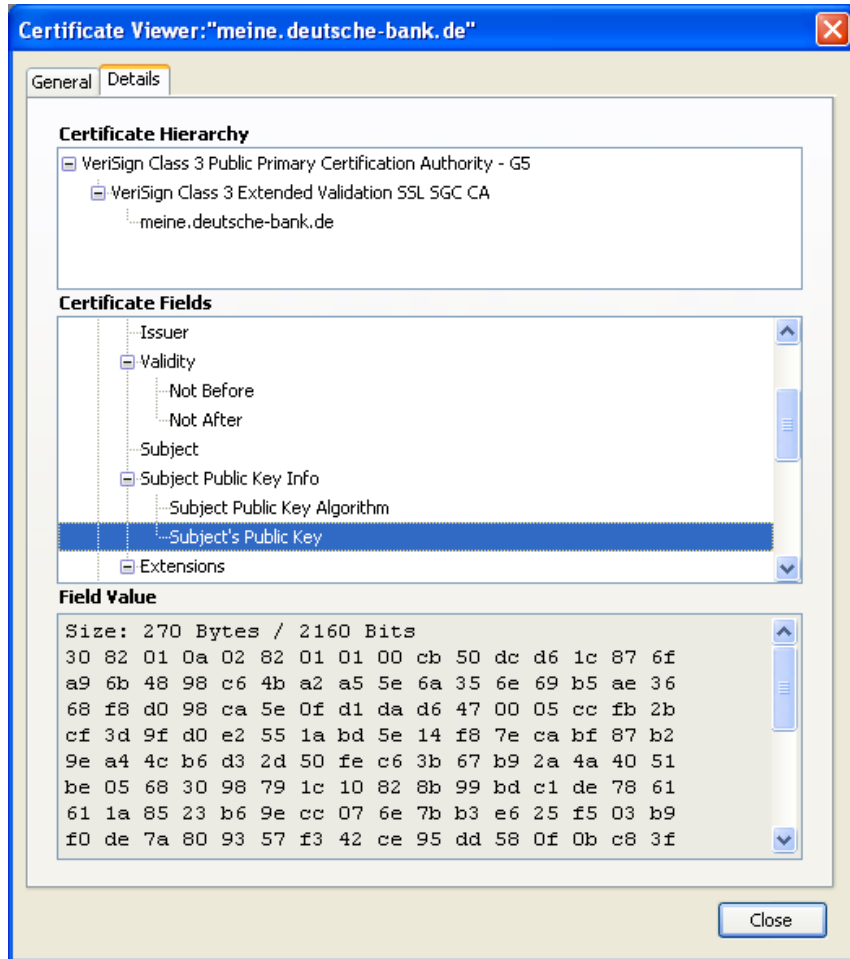


This means, that the connection is authenticated (at least at one side) and that the transferred data is strongly encrypted.

SSL-secured (128 Bit)

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)

Client



Server



1. SSL initiation

Send server certificate



2.

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)

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

7.



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)

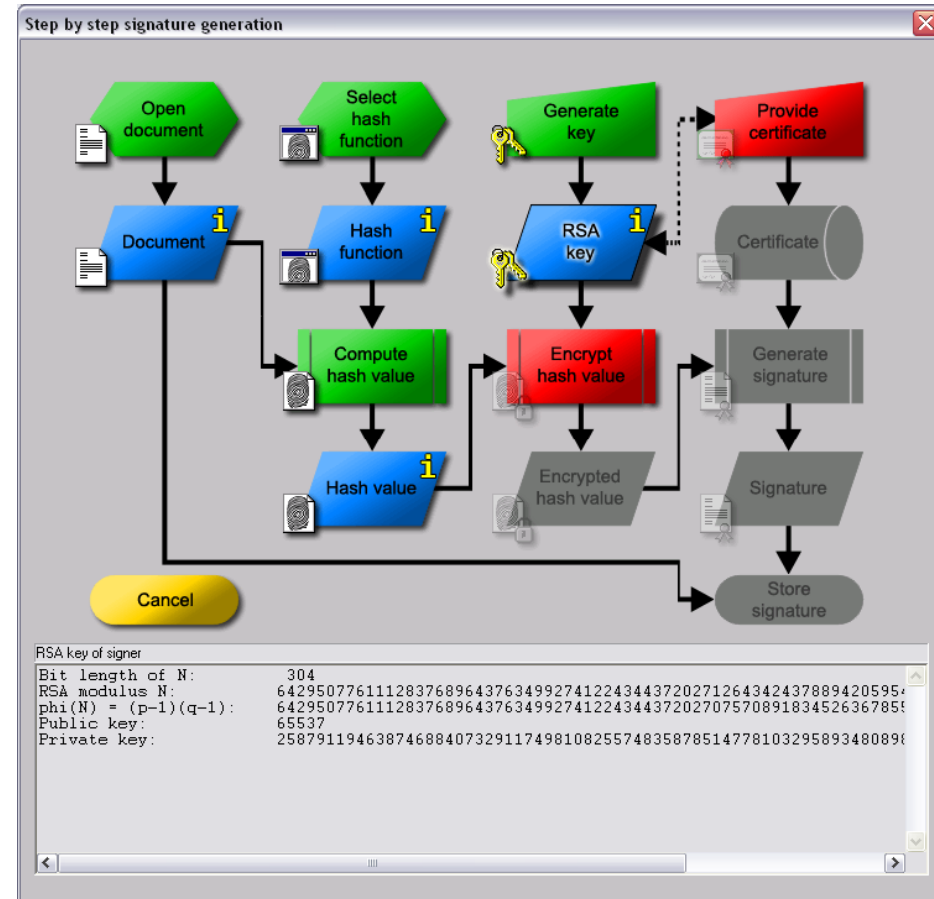
Digital signature visualised

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



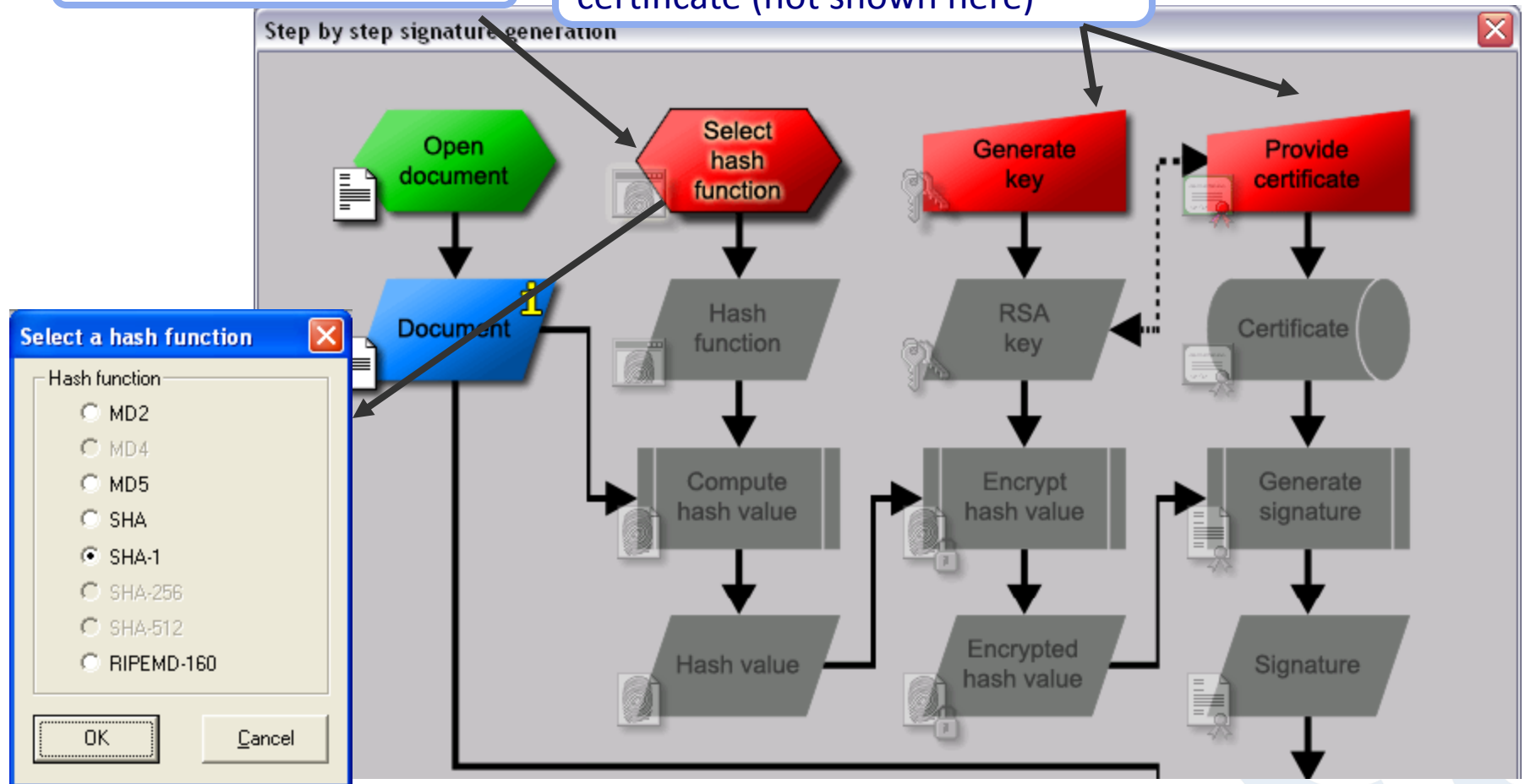
Menu: „Digitale Signatures/PKI“ \
„Signature Demonstration (Signature Generation)“

Examples (2)

Digital signature visualised: a) Preparation

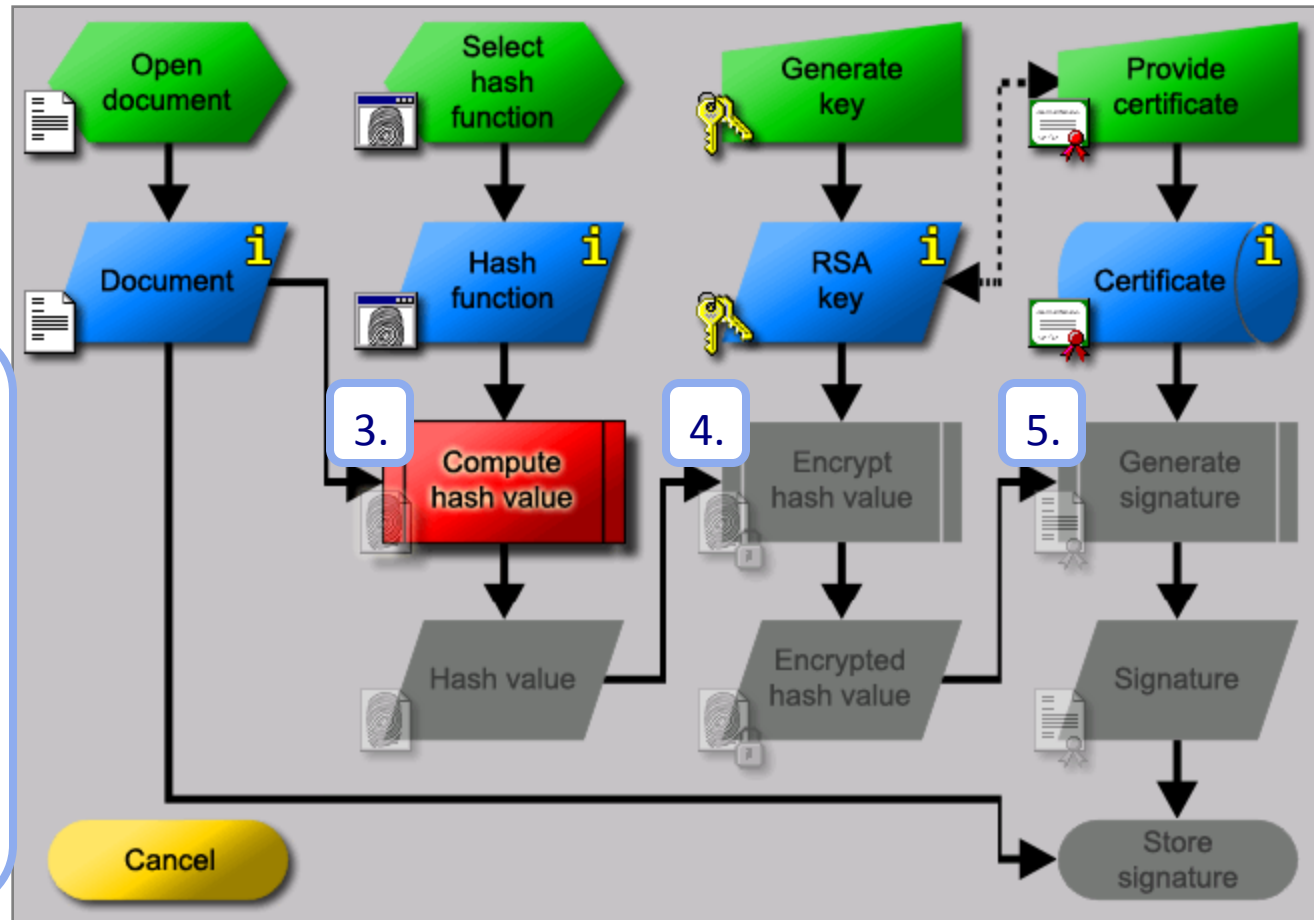
1. Select hash function

2. Provide key and certificate (not shown here)



Examples (2)

Digital signature visualised: b) Cryptography



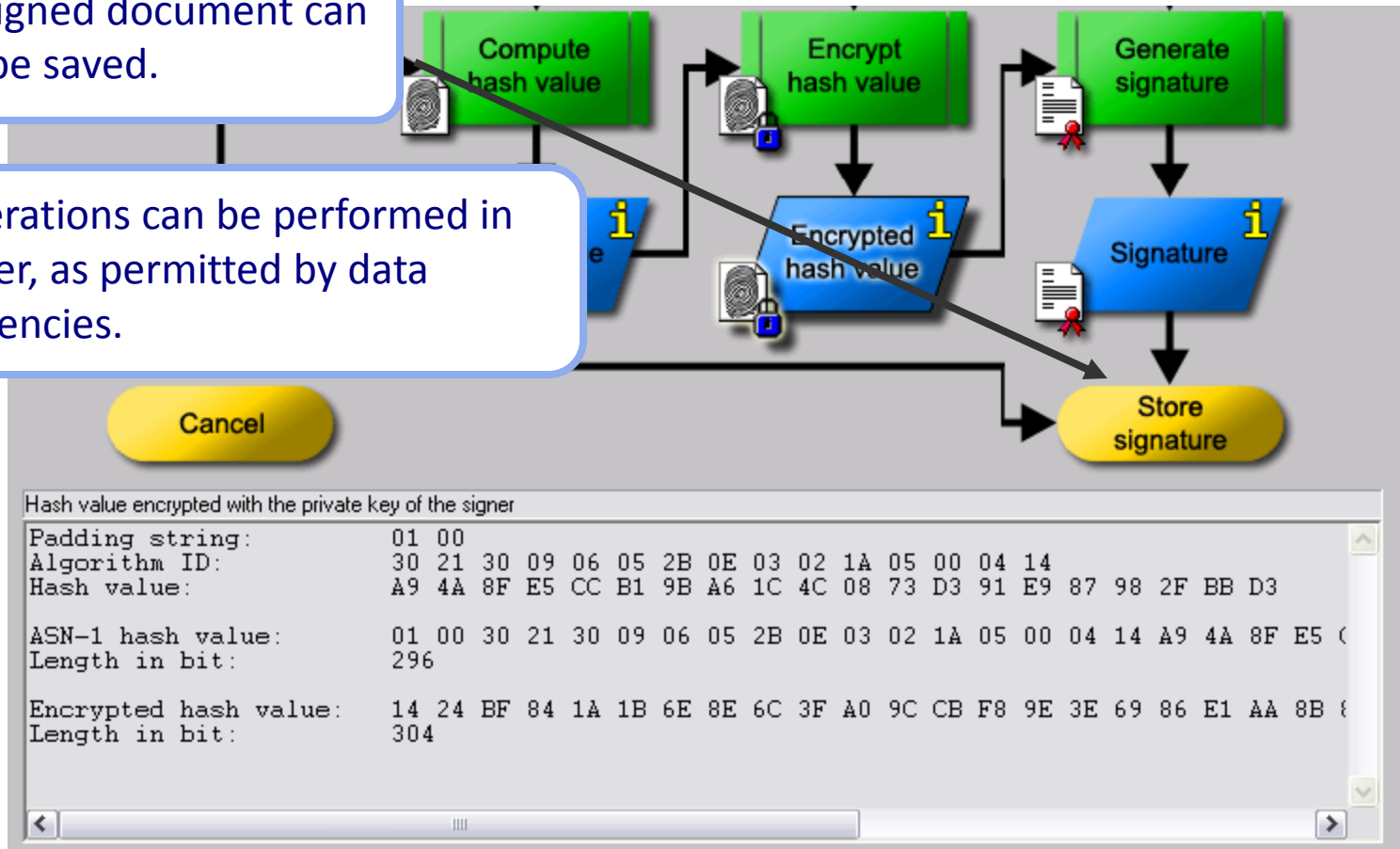
3. Calculate hash value
4. Encrypt hash value with private key (sign)
5. Generate signature

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

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 $d = e^{-1} \pmod{\phi(N)}$ 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 .

Factorise RSA modulus...

RSA parameters

RSA modulus N (public)

$\phi(N) = (p-1)(q-1)$ (secret)

Public key e (secret)

Private key d

Update parameters

☐ RSA encryption using e / decryption using d

1. Enter RSA parameters* N and e

2. Factorise

Examples (3)

Short RSA modulus: factorise RSA modulus

The screenshot displays the 'Factorisation of a Number' window in CrypTool 1.4.20. The 'Algorithms for factorisation' section on the left has all methods checked: Brute-force method, Brent algorithm, Pollard method, Williams method, Lenstra algorithm, and Quadratic sieve method. The 'Input' section on the right contains the text 'Enter the number to be factorised:' followed by a text box with the value '63978486879527143858831415041'. Below this, the 'Factorisation' section shows 'The factorisation is represented in the format $\langle z_1^{a_1} * z_2^{a_2} * \dots * z_n^{a_n} \rangle$. Composite numbers are appearing in red color.' It also indicates 'Last factorisation through: Pollard' and 'Total time required: 0,984 seconds.' The 'Factorisation result:' section shows the output '145295143558111 * 440334654777631', which is circled in black. A blue box with the text '3. Factorisation yields p and q' has an arrow pointing to this result. Another arrow points from the result to the 'OK' button in a 'CrypTool' dialog box that appears in the foreground. The dialog box contains an information icon and the text: '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.' The dialog box has an 'OK' button. The main window also has 'Details' and 'Close' buttons at the bottom.

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 $\langle z_1^{a_1} * z_2^{a_2} * \dots * z_n^{a_n} \rangle$. Composite numbers are appearing in red color.

Last factorisation through: Pollard

Total time required: 0,984 seconds.

Factorisation result:

145295143558111 * 440334654777631

3. Factorisation yields p and q

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.

OK

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 $d = e^{-1} \pmod{\phi(N)}$ 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

Generate prime numbers...

RSA parameters

RSA modulus N: 63978486879527143858831415041 (public)

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

Public key e: 17579

Private key d: 10663687727232084624328285019

Update parameters

RSA encryption using e / decryption using d

Input as: ☐ text ☒ numbers

Ciphertext coded in numbers of base 10

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 Number of characters: 256

☐ Specify alphabet:

RSA variant

☒ Normal ☐ Dialogue of the Sisters

Method for coding a block into numbers

☒ b-adjc ☐ 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 modulus N in bit, the number of characters in the alphabet and the method used for the coding.

Block length in characters: (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

OK Cancel

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	

[Update parameters](#)

RSA encryption using e / decryption using d

Input as ☐ text ☒ numbers [Options for alphabet and number system...](#)

Ciphertext coded in numbers of base 10

5069057070940529666287522 # 40486038748314205283477353228 # 26906996968026845590696474185

Decryption into plaintext $m[i] = c[i]^d \pmod{N}$

00094604723425878030697780557 # 00080116466004756901239213377 # 0008253339409781111818054

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

NATURAL NUM # BERS ARE MA # DE BY GOD

Plaintext

NATURAL NUMBERS ARE MADE BY GOD

[Encrypt](#) [Decrypt](#) [Close](#)

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 v
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

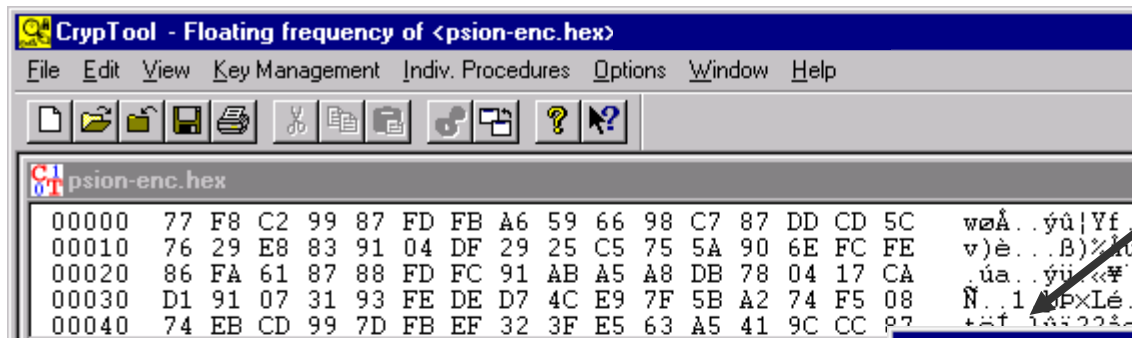
- pre-analysis
 - entropy
 - floating entropy
 - compression test
- auto-correlation
- try out automatic analysis with classical methods

} *probably classical
encryption algorithm*

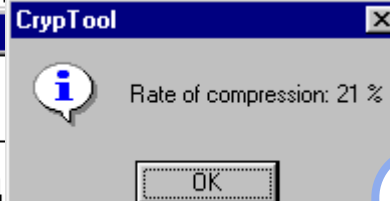
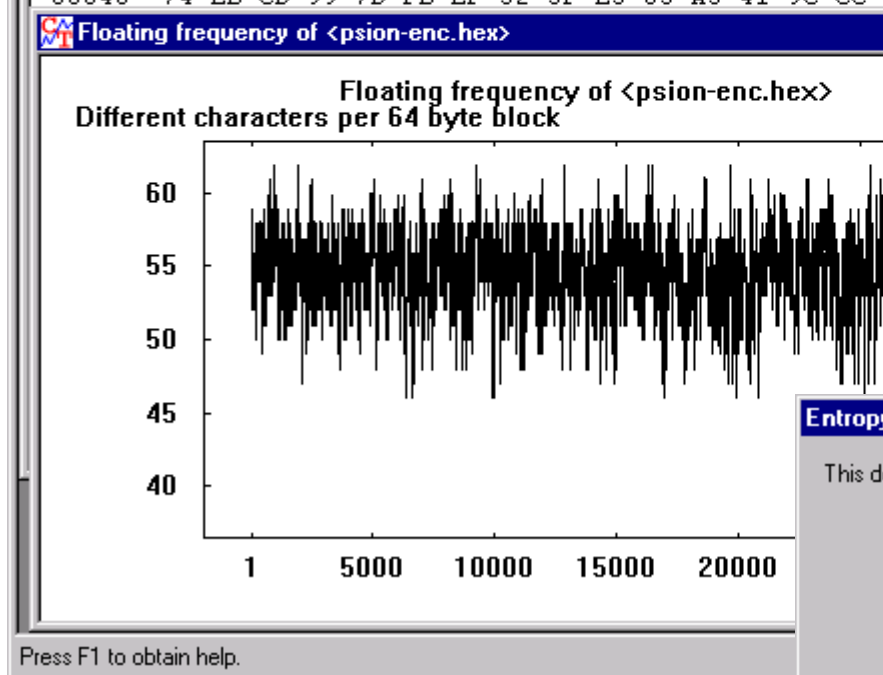


Examples (4)

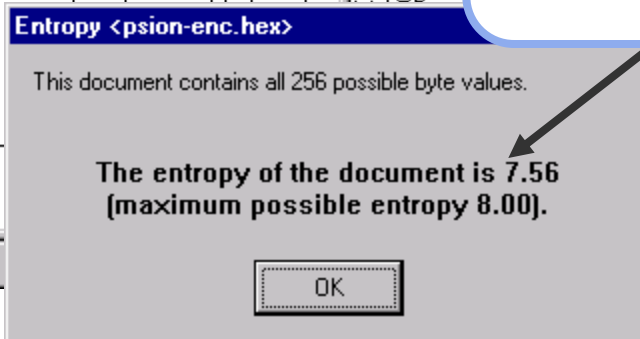
PSION 5 PDA – determine entropy, compression test



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

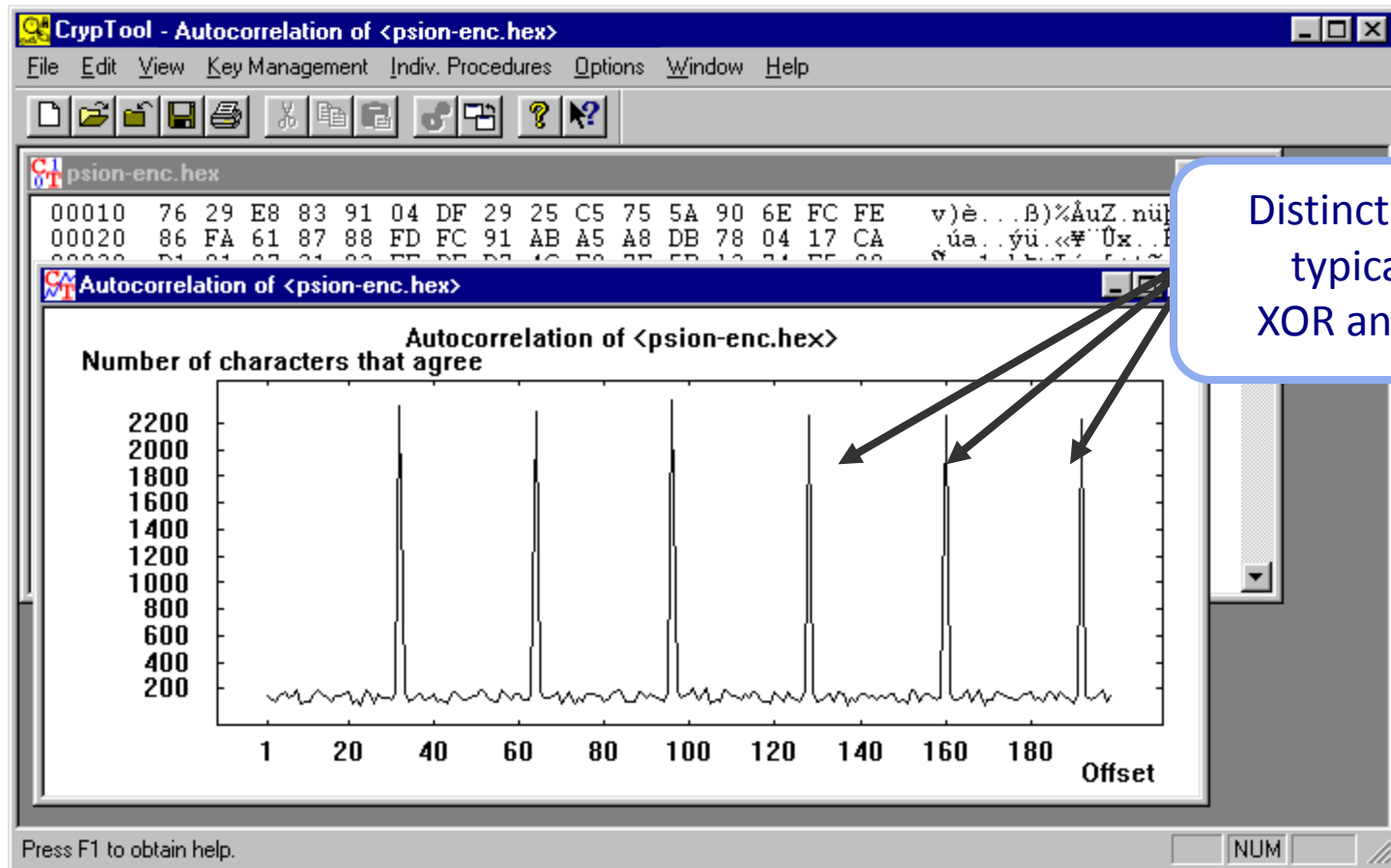


The entropy provides
no indication for a
specific encryption
method.



Examples (4)

PSION 5 PDA – determine auto-correlation



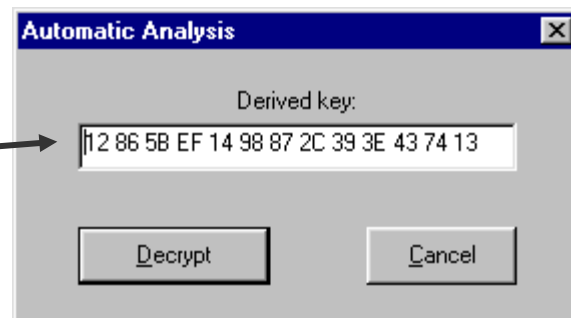
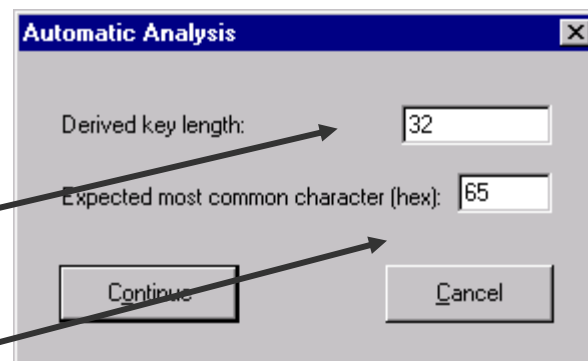
* The encrypted file is available with CrypTool (see CrypTool\examples\psion-enc.hex)

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



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.

Address	Key 1	Key 2	Key 3	Key 4	Key 5	Key 6	Key 7	Key 8	Key 9	Key 10	Key 11	Key 12	Key 13	Key 14	Key 15	Key 16	Decrypted Text
00000	65	72	67	AA	73	65	74	7A	20	28	55	53	74	47	29	06	erg³setz (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 .el¹²ng,
00040	62	65	72	AA	69	63	68	06	06	A7	20	31	2E	06	28	31	ber³ich..\$ 1..(1
00050	29	20	89	65	72	20	55	6D	B8	61	74	BF	B8	74	65	BA) .er Um,at¿,te²
00060	65	72	20	BA	6E	74	65	72	6C	69	65	67	65	6E	20	64	er ²nterliegen 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	ätz³: 1. die Lie
00090	66	65	B7	75	6E	67	65	6E	65	75	6E	A9	65	73	6F	B3	fe .ungeneun³eso³
000A0	73	74	69	AC	65	6E	20	4C	65	69	73	74	75	6E	67	65	stiren 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	g³gen E³tg³tt @
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,

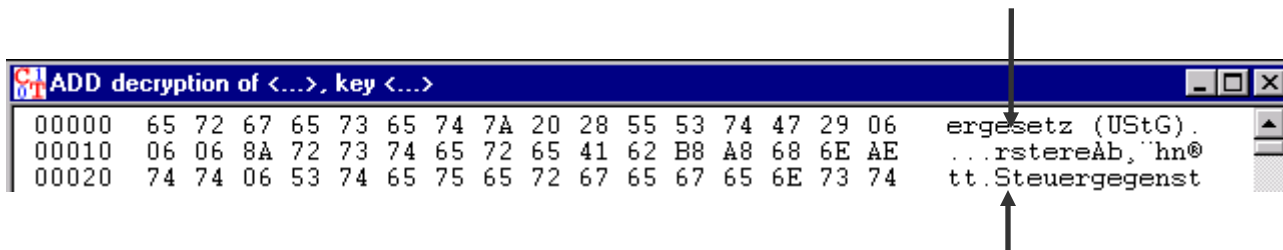
- determine incorrect byte positions, e.g. 0xAA at position 3
- guess and write down corresponding correct bytes: „e“ = 0x65

In encrypted initial file hex dump,

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

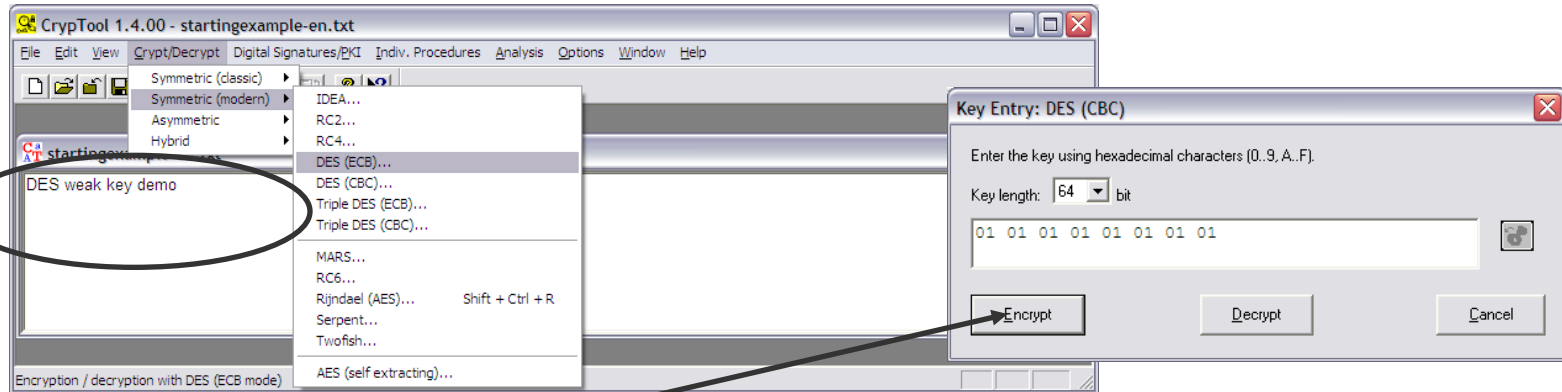
Correct key from the clipboard

- 12865B341498872C393E43741396A45670235E111E907AB7C0841...
- Decrypt encrypted initial document using binary addition
- bytes at position 3, 3+32, 3+2*32, ... are now correct

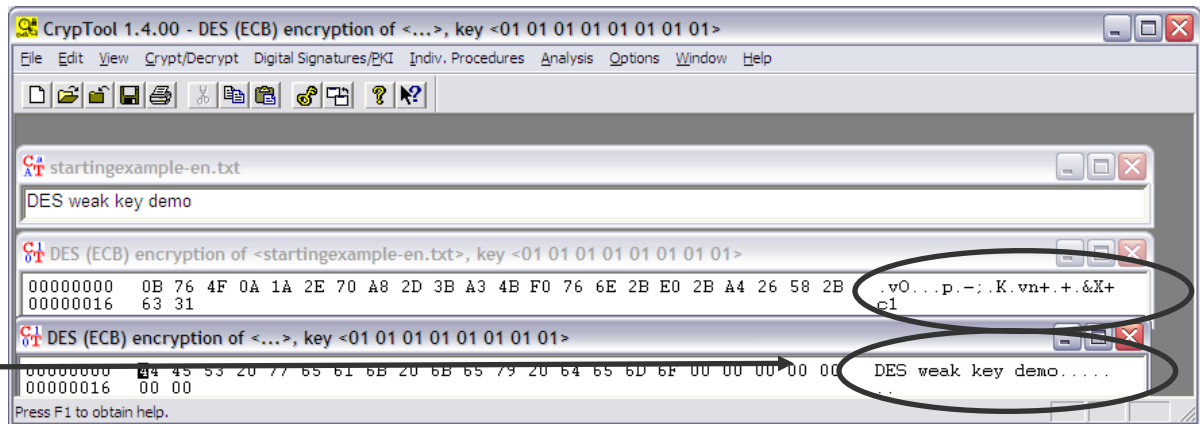


Examples (5)

Weak DES key



encrypt 2 times with...
results in plaintext



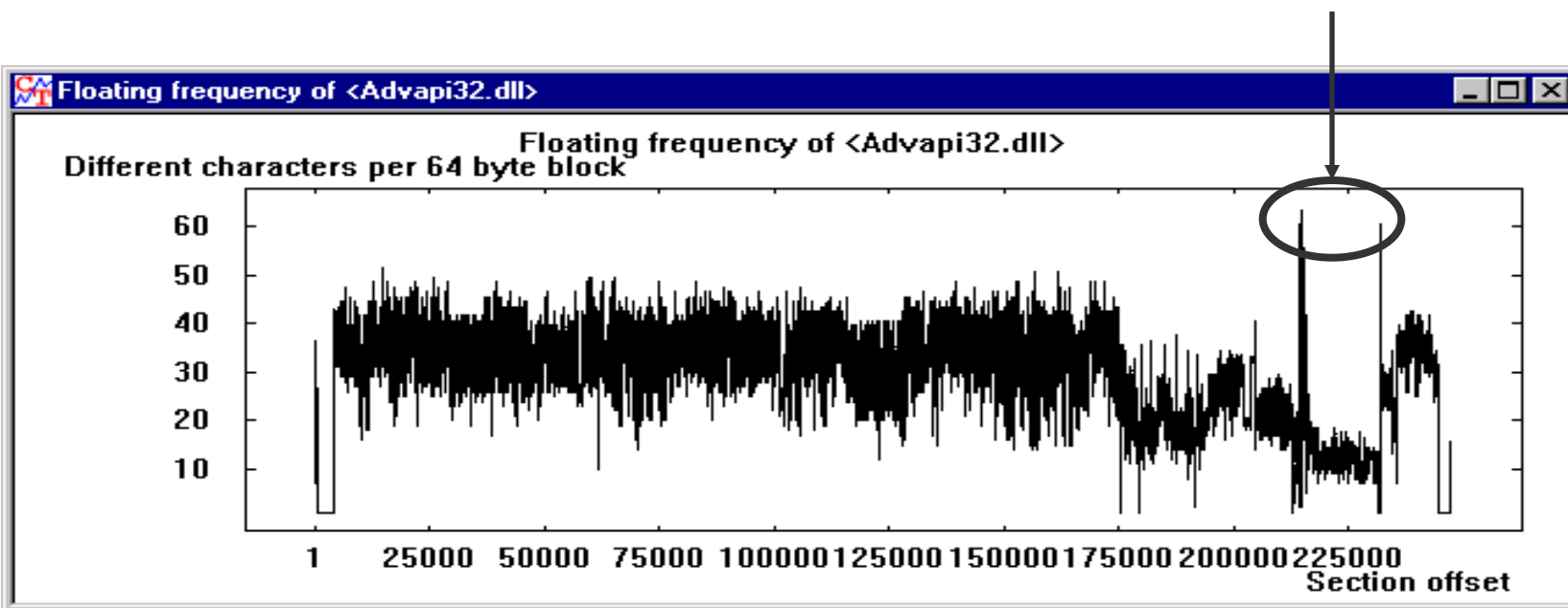
Examples (6)

Locate key material

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

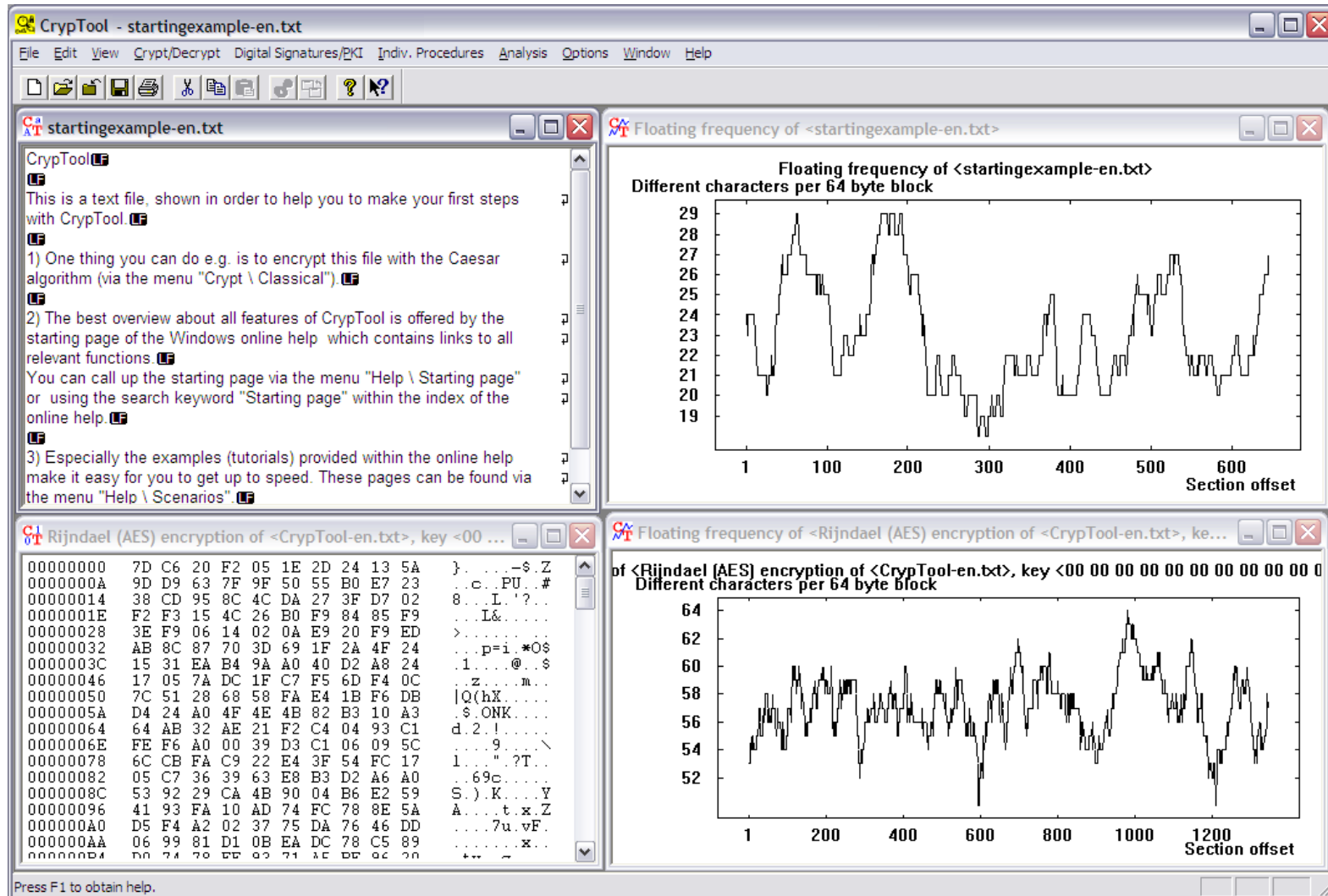
Background:

- 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 (Windows NT)



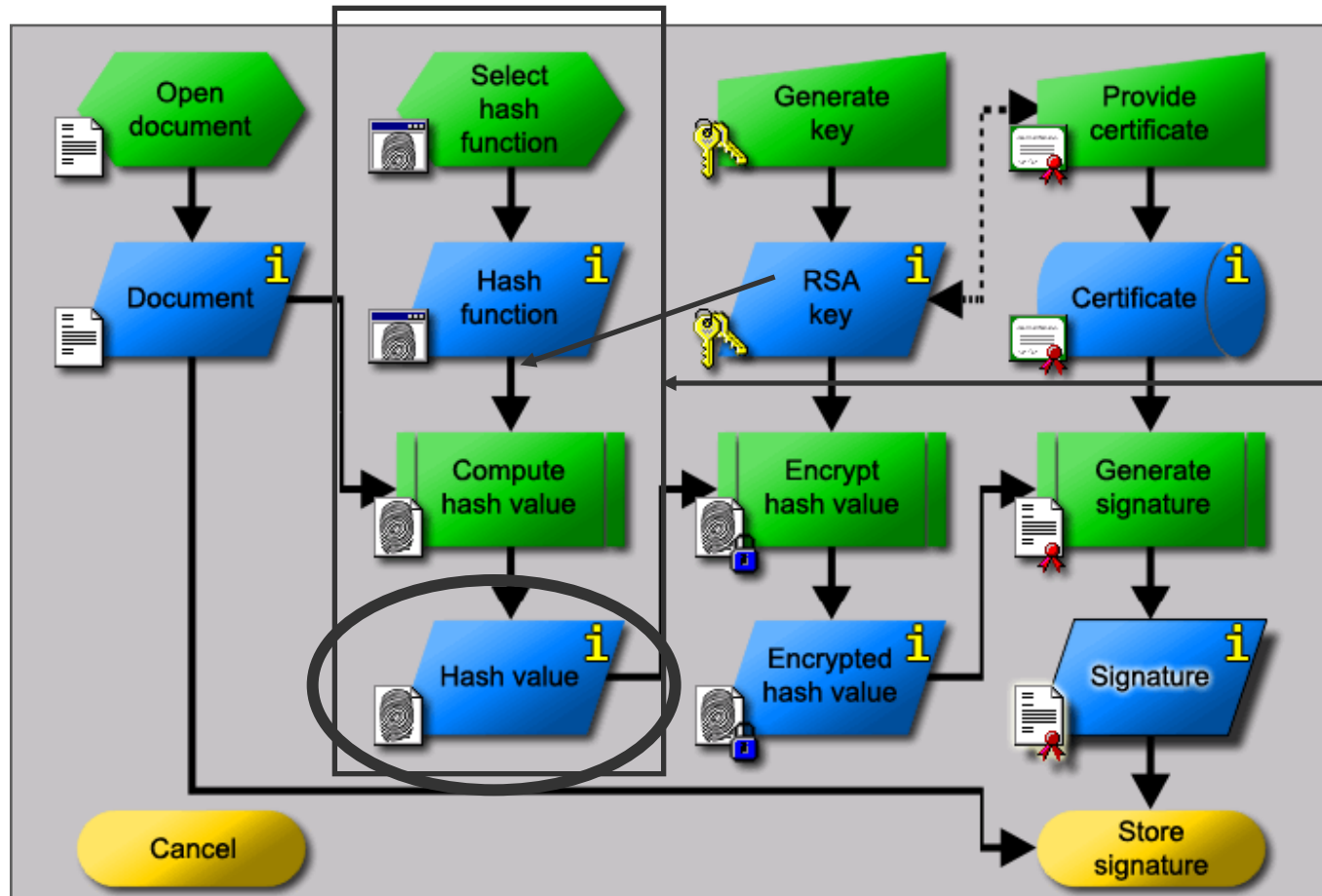
Examples (6)

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:

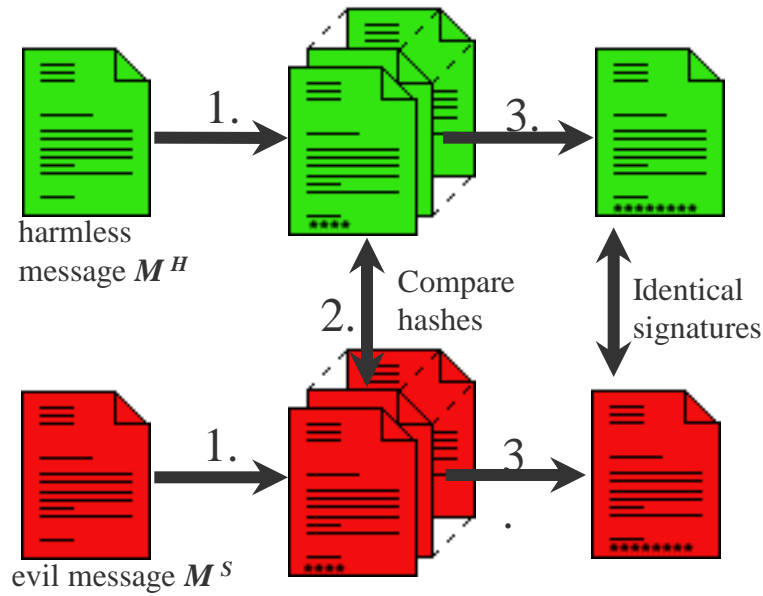
- ASCII texts can be modified by changing/inserting non-printable characters, without changing the visible content
- modify two texts in parallel until a hash collision is found
- exploit the birthday paradox (birthday attack)
- generic attack applicable to all hash functions
- can be run in parallel 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, \dots, 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, \dots, M_N^S :
- search collision between M_1^H, \dots, M_N^H and M_1^S, \dots, M_N^S :

$$N \approx 2^n$$

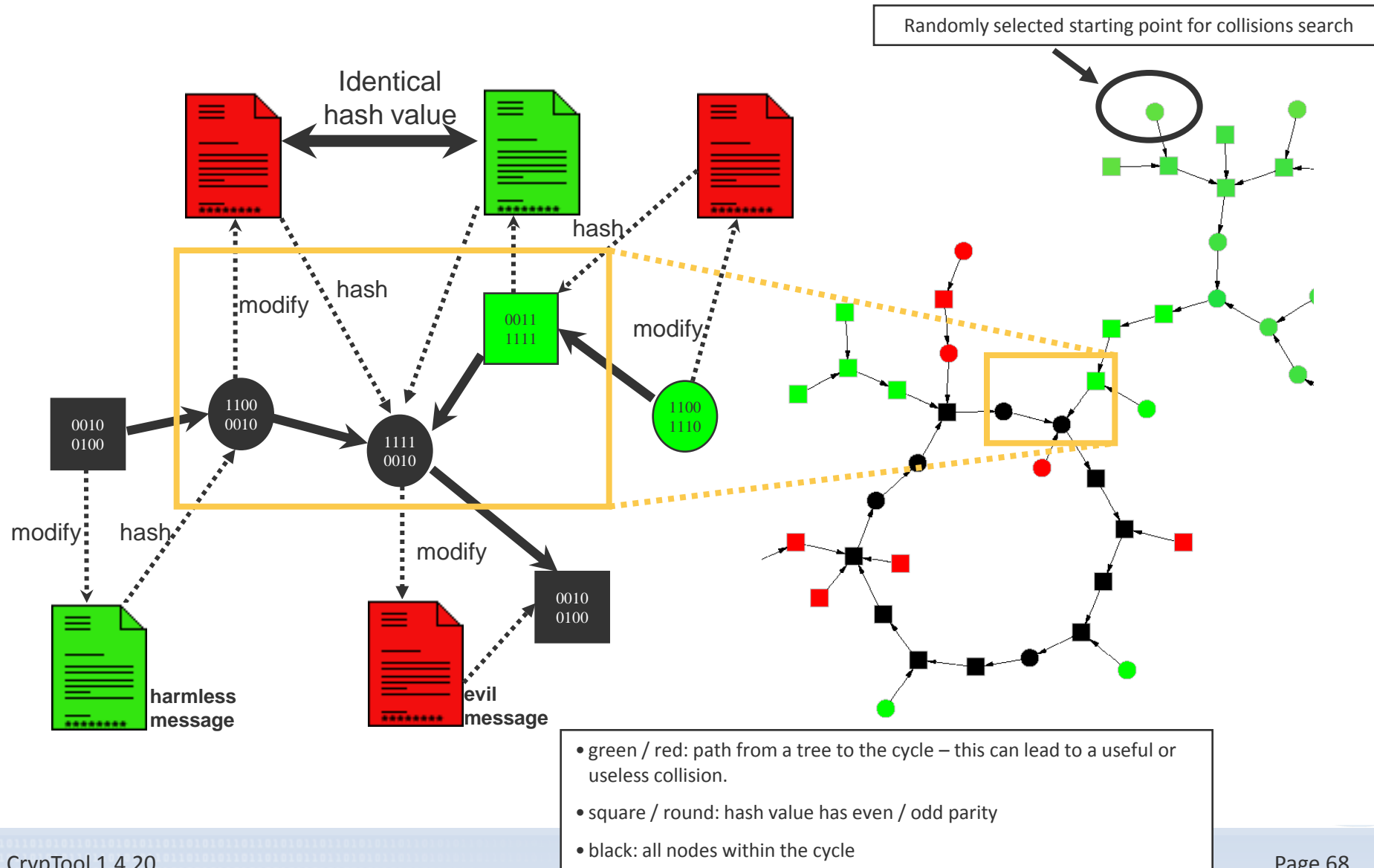
$$N \approx 2^{n/2}$$



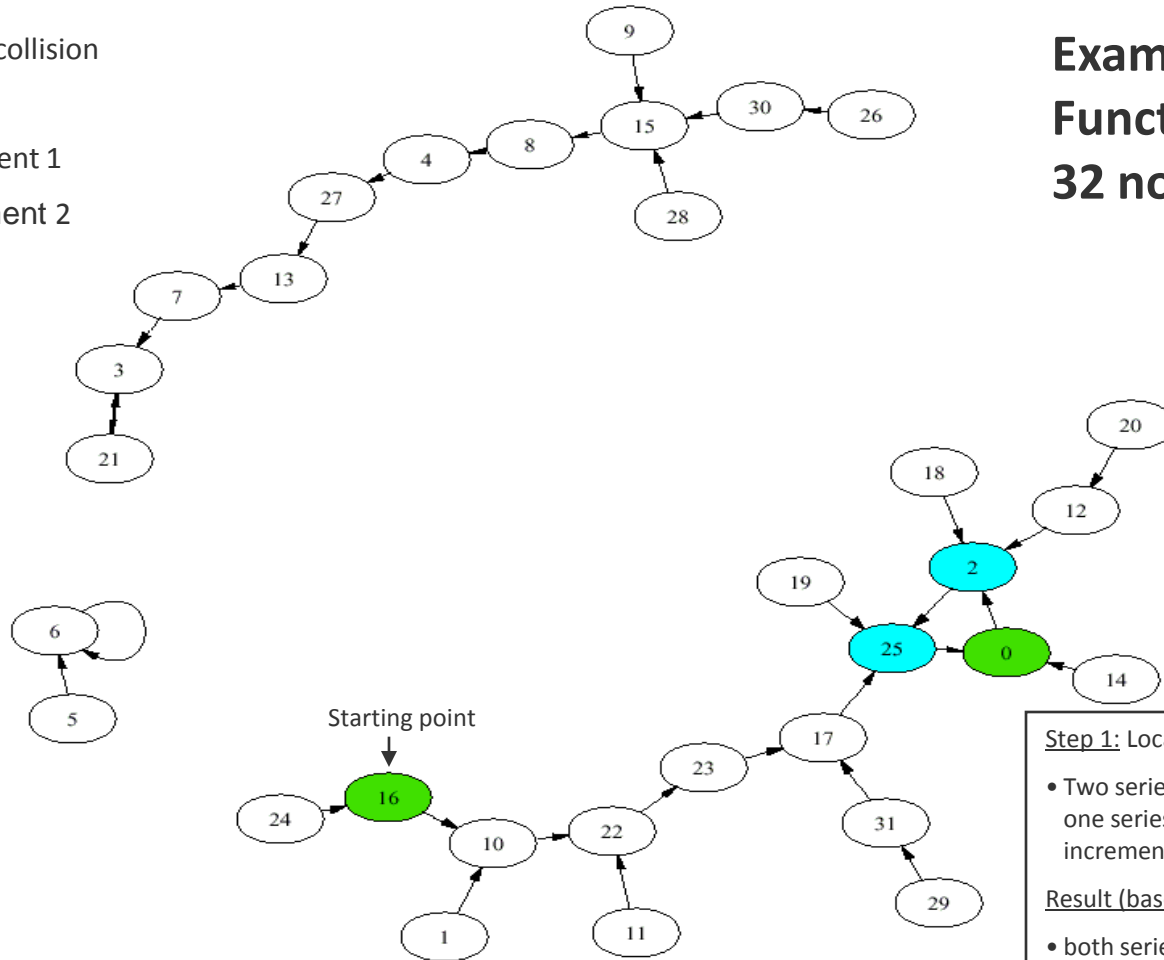
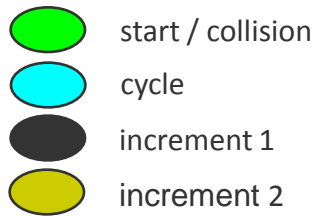
Estimated number of generated messages in order to find a hash collision.

Locate Hash Collisions (1)

Mapping via text modifications



Floyd Algorithm: meet within the cycle



Example: Function graph with 32 nodes

Step 1: Locate matching point within cycle:

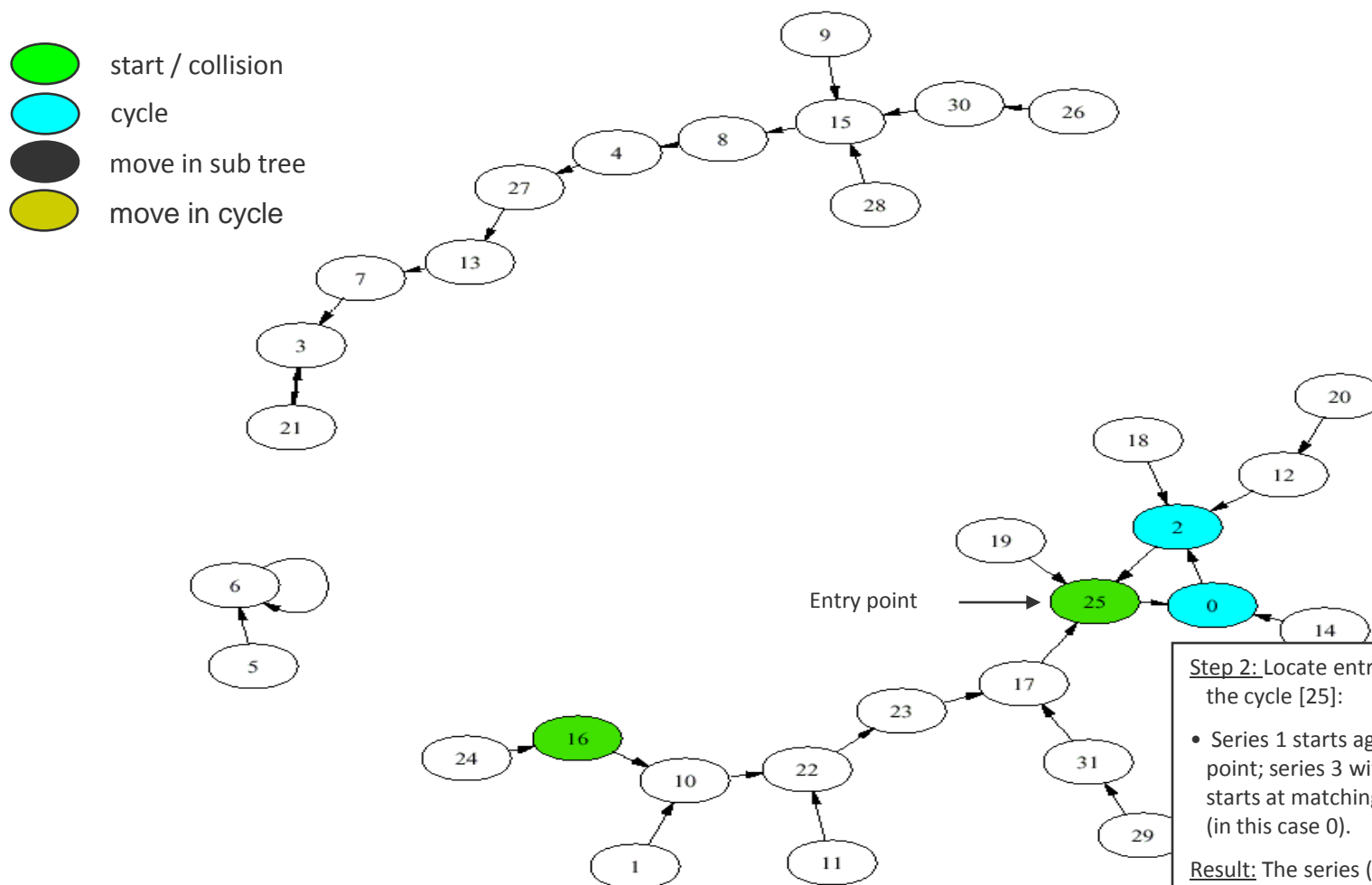
- Two series with identical starting point [16]: one series with increment 1, the other with increment 2.

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



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).

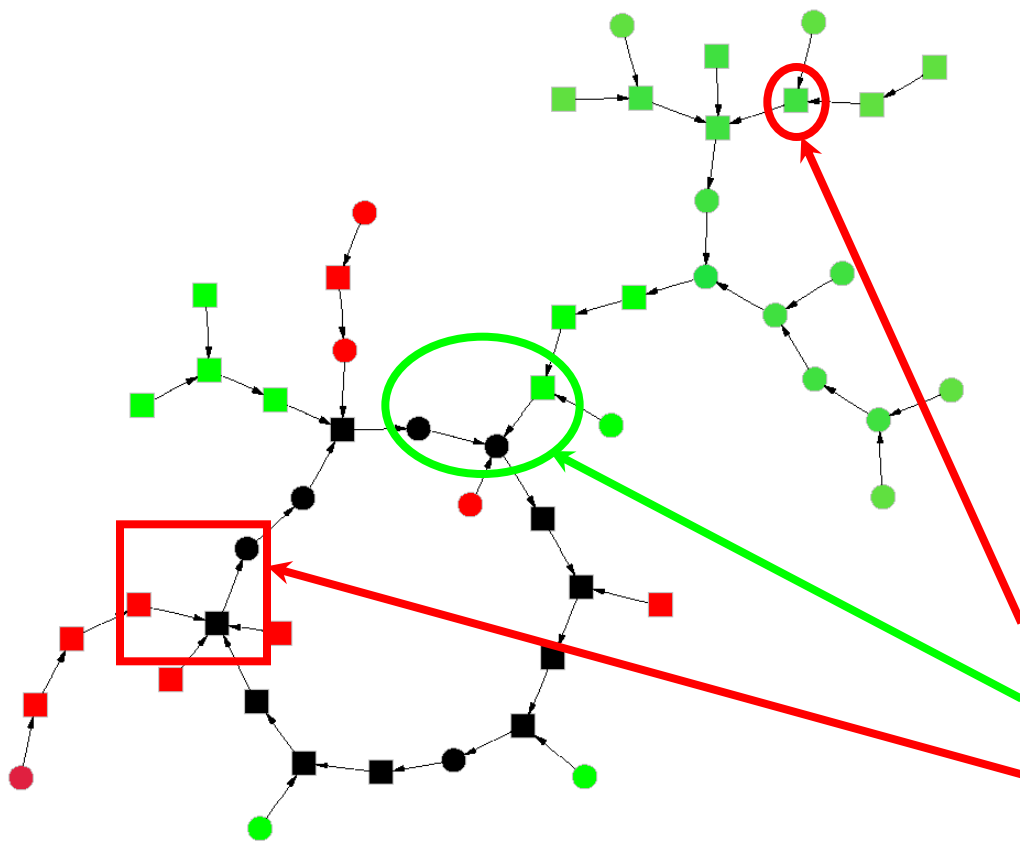
Result: The series (1 and 3) match in cycle entry point of series 1 (in this case 25)

- 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.

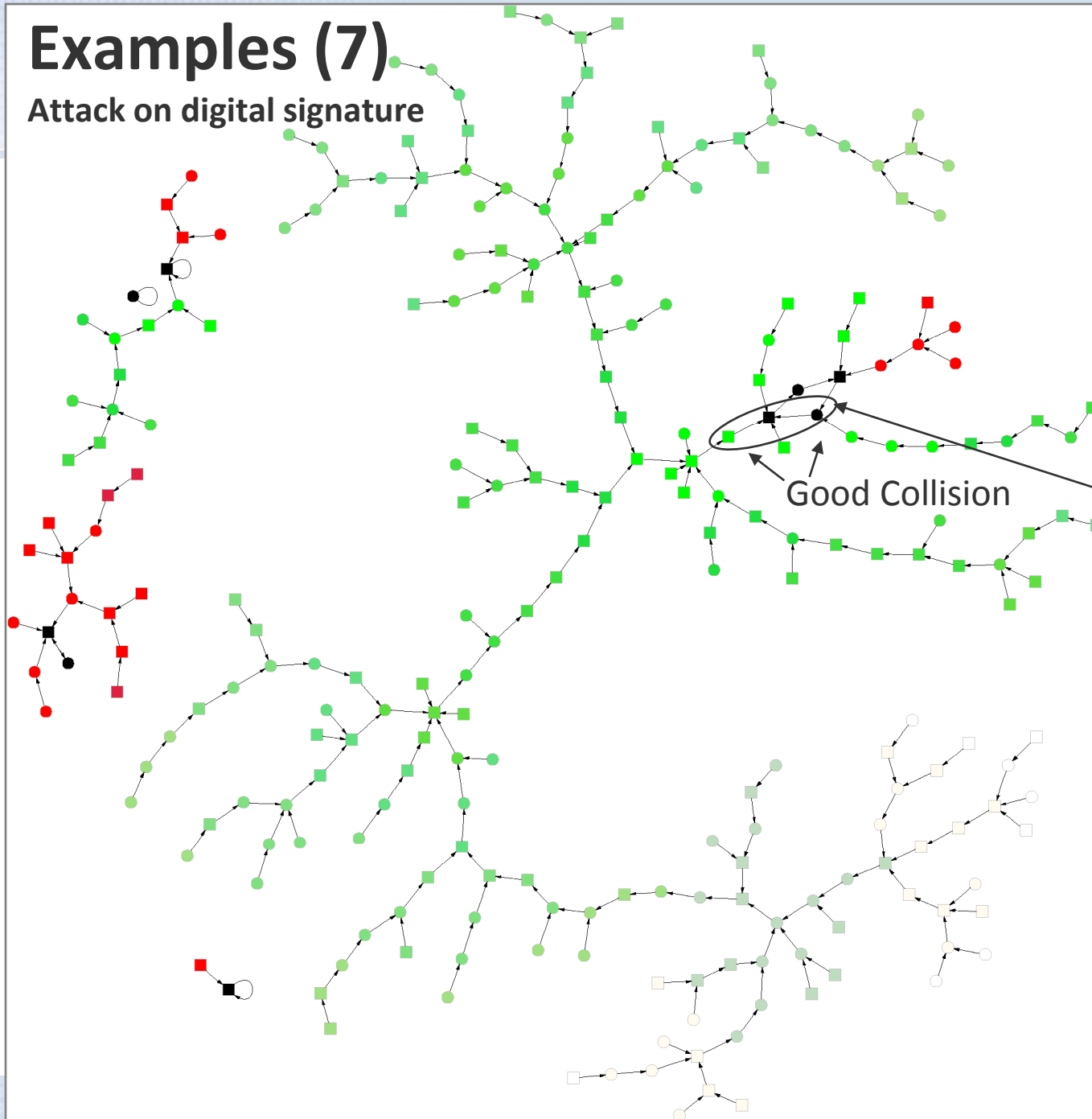


Starting point
Good collision
Bad collision

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

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

The image shows the 'Attack on the Hash Value of the Digital Signature' window in CryptTool 1.4.20. The window is divided into several sections. A large arrow points from the 'Options...' button in the main window to the 'Options for the attack on the hash value of the digital sig...' window.

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

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

4. 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.

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 messages
Determine the way messages are modified:

☐ Insert blanks ☒
☒
☒ Attach characters ☐

Apply Reset to standard

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

Examples (7)

Attack on digital signature: Results

The image shows two overlapping email window screenshots. The top window, titled 'Harmless message: MD5, <A9 76 34 AB>', contains the text: 'Dear Mr Shopaholic, please order a typewriter. Regards Honest John ABBDBCBCDF AAAADABB'. The bottom window, titled 'Dangerous message: MD5, <A9 76 34 AB>', contains the text: 'Dear Mr Shopaholic, please order a Porsche and a prepaid insurance scheme for Mr. Dodgy. Regards Honest John ABBBCCDDAABADCDDDC'. Both messages have the same MD5 hash prefix 'MD5: 4F 47 DF 1F'. A yellow oval highlights the subject line of the harmless message, and another yellow oval highlights the signature 'Honest John' in both messages. A yellow oval also highlights the body text of the dangerous message. A black arrow points from the 'Honest John' signature in the harmless message to the 'Honest John' signature in the dangerous message. A text box at the bottom states: 'The first 32 bits of the hash values are identical.'

Harmless message: MD5, <A9 76 34 AB>

Dear Mr Shopaholic,
please order a typewriter.
Regards
Honest John ABBDBCBCDF AAAADABB

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

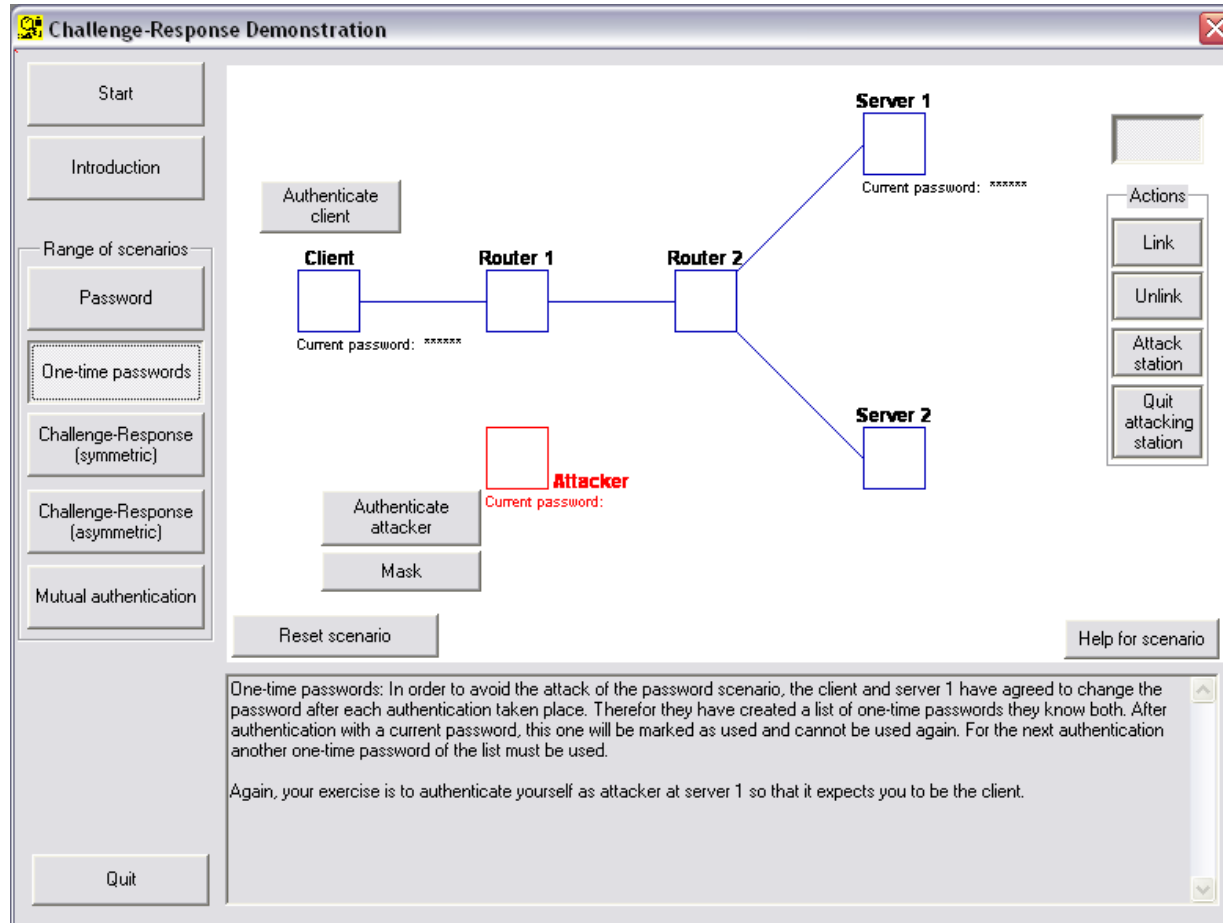
- 72 Bit *partial collision* (equality of the first 72 hash value bits) were found in a couple of days on a single PC.
- Signatures using hash values of up to 128 bit can be attacked today using massive parallel search!
- Use hash values of at least 160 bit length.

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

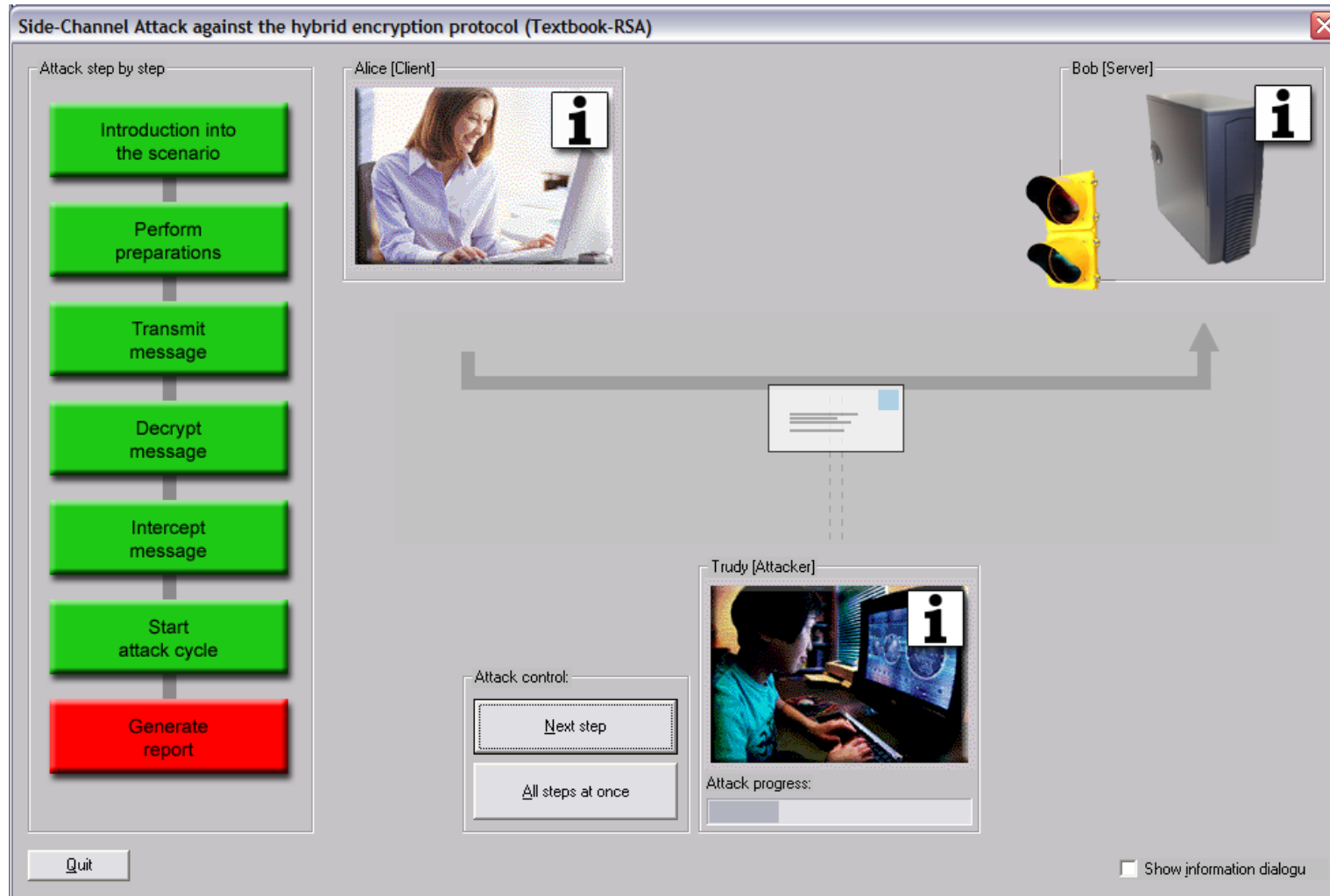


- Interactive demo for different authentication methods.
- Defined opportunities of the attacker.
- You can play the role of an attacker.
- **Learning effect:** Only mutual authentication is secure.

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

Examples (9)

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



Menu: „Analysis“ / „Asymmetric Encryption“ / „Side-Channel Attack on Textbook-RSA“

Examples (9)

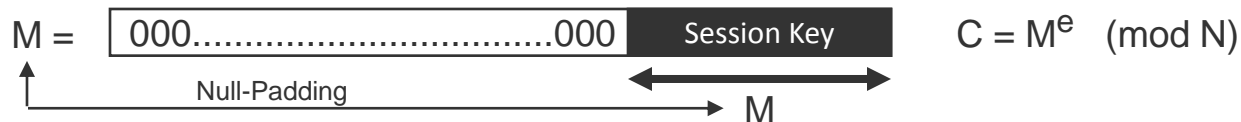
Idea for this side channel attack

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

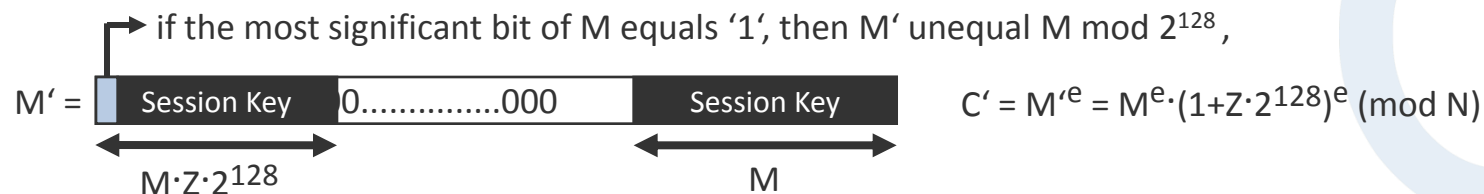
Prerequisites:

- RSA encryption: $C = M^e \pmod{N}$ and decryption: $M = C^d \pmod{N}$.
- 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 $M = \lfloor N/Z \rfloor$




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 .



Examples (10)

Mathematics: Attacks on RSA using lattice reduction

The image shows a software window titled "Attack on small secret exponents (according to Bleumer / May)". It contains three main sections: "Description", "Step 1: Enter key parameters and key", and "Step 2: Enter attack parameters for the lattice base reduction". The "Description" section explains the attack's purpose and provides instructions. "Step 1" includes input fields for "Bitlength of N" (300), "delta" (0.2600), and a "Generate random RSA key" button. "Step 2" includes input fields for "m" (4), "t" (2), "Lattice dimension" (15), and "Maximal delta" (0.2653), with explanatory text for each. A "Step 3: Start attack" section shows progress indicators for "Building lattice", "Reducing lattice", "Calculating resultant", and "Overall time", along with "Start" and "Cancel" buttons. At the bottom, there are fields for "Found factorization" (p and q) and "Show log file" / "Close dialog" buttons.

Attack on small secret exponents (according to Bleumer / 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: delta:

N:

e:

d:

Step 2: Enter attack parameters for the lattice base reduction

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

t: Is optimally calculated as a function of m.

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

Maximal delta: Maximal size of delta for big N (N>1000 Bit).

Step 3: Start attack

Building lattice:

Reducing lattice: Reductions:

Calculating resultant: Resultants:

Overall time:

Found factorization:

p: q:

- 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“ / ...

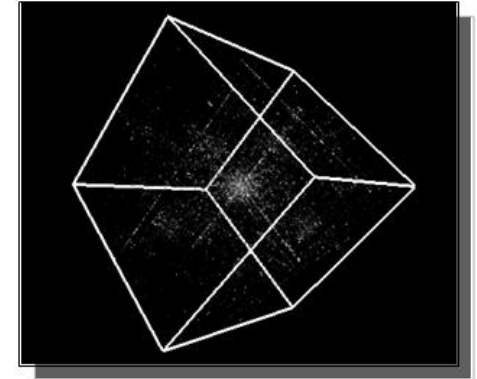
Examples (11)

Random analysis with 3-D visualisation

3-D visualisation for random analysis

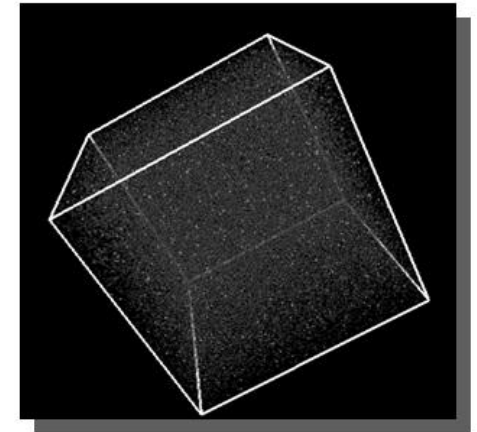
Example 1

- 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)**

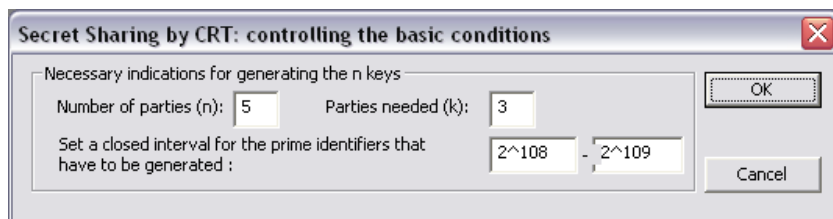


Examples (12)

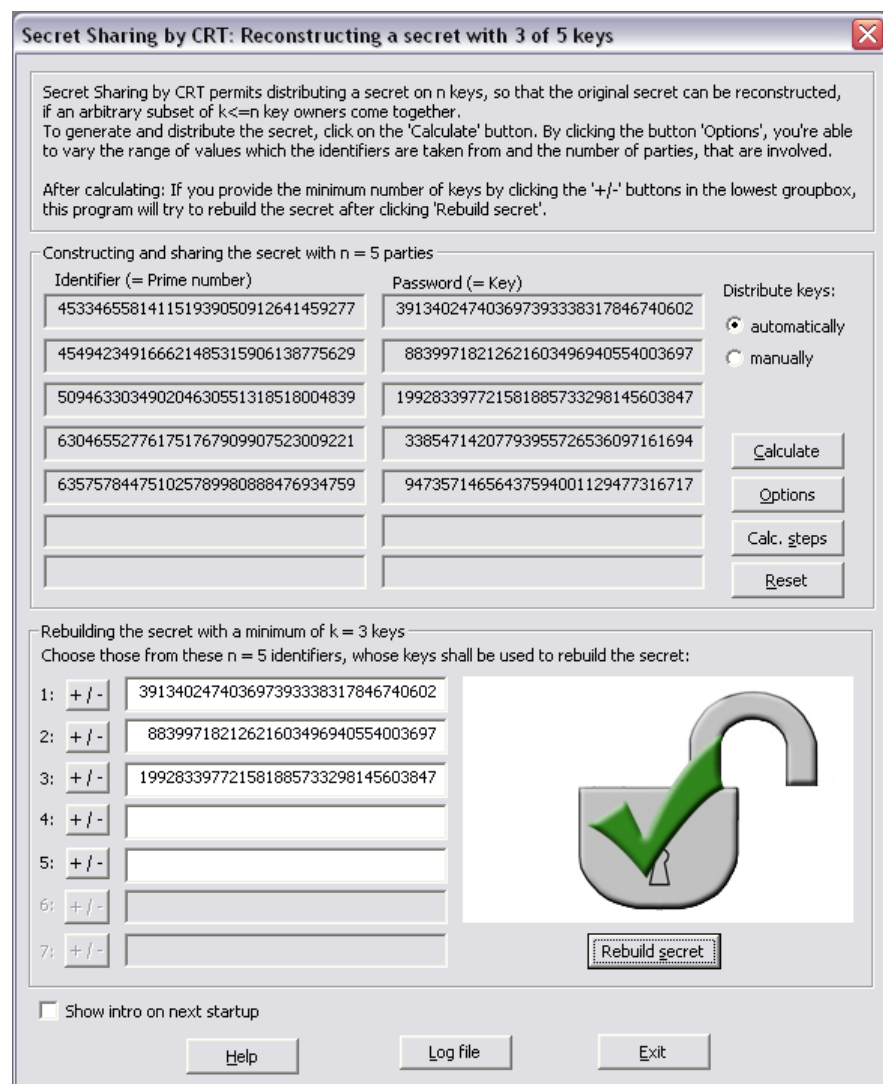
Secret sharing with CRT – Implementation of the Chinese remainder theorem (CRT)

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.



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



Examples (12)

Shamir secret sharing

Secret Sharing Beispiel (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 polynomial
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 ($t \leq n$) 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 $S \geq 0$

Number of participants n with $n > 0$

Threshold (minimum) t with $t > 0$

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

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

Polynomial $f(x)$

Prime p

Participants' values, calculated from chosen parameters:

	Participant	Public value x	Share [secret value $f(x)$]
<input checked="" type="checkbox"/>	participant 1	31	1612
<input type="checkbox"/>	participant 2	1527	520
<input checked="" type="checkbox"/>	participant 3	1388	1080
<input type="checkbox"/>	participant 4	1155	1197
<input checked="" type="checkbox"/>	participant 5	575	58
<input type="checkbox"/>	participant 6	1383	157
<input type="checkbox"/>	participant 7	1064	1055
<input type="checkbox"/>	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 $x = a[i] \bmod m[i]$ ($i=1, \dots, 9$) below and compute a solution. One can use such equation systems to determine the number of days until certain planets become aligned.

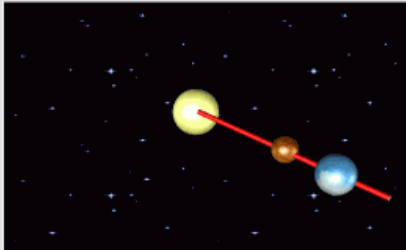
Simultaneous congruences/ modular linear equation systems

$x \equiv$	15	mod	88
$x \equiv$		mod	
$x \equiv$	100	mod	365
$x \equiv$		mod	
$x \equiv$	0	mod	4327
$x \equiv$		mod	
$x \equiv$		mod	
$x \equiv$	0	mod	60149
$x \equiv$		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

<input checked="" type="checkbox"/> Mercury	<input type="checkbox"/> Mars	<input type="checkbox"/> Uranus
<input type="checkbox"/> Venus	<input checked="" type="checkbox"/> Jupiter	<input checked="" type="checkbox"/> Neptune
<input checked="" type="checkbox"/> Earth	<input type="checkbox"/> Saturn	<input type="checkbox"/> 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)

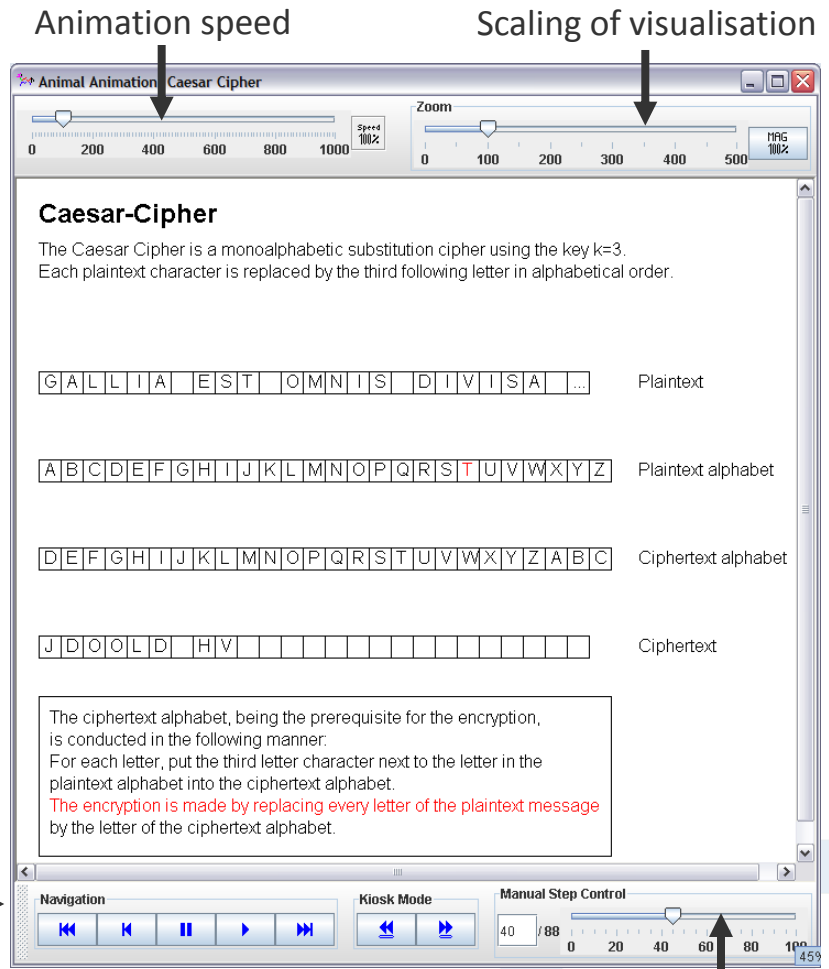
Animated visualisation of several symmetric algorithms

- Caesar
- Vigenère
- Nihilist
- DES

CrypTool

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

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

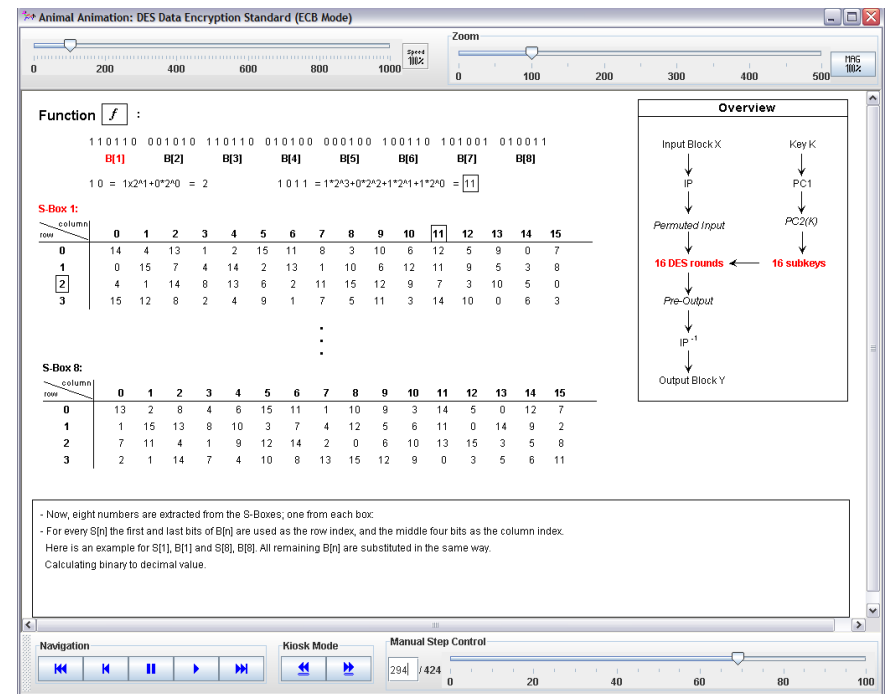
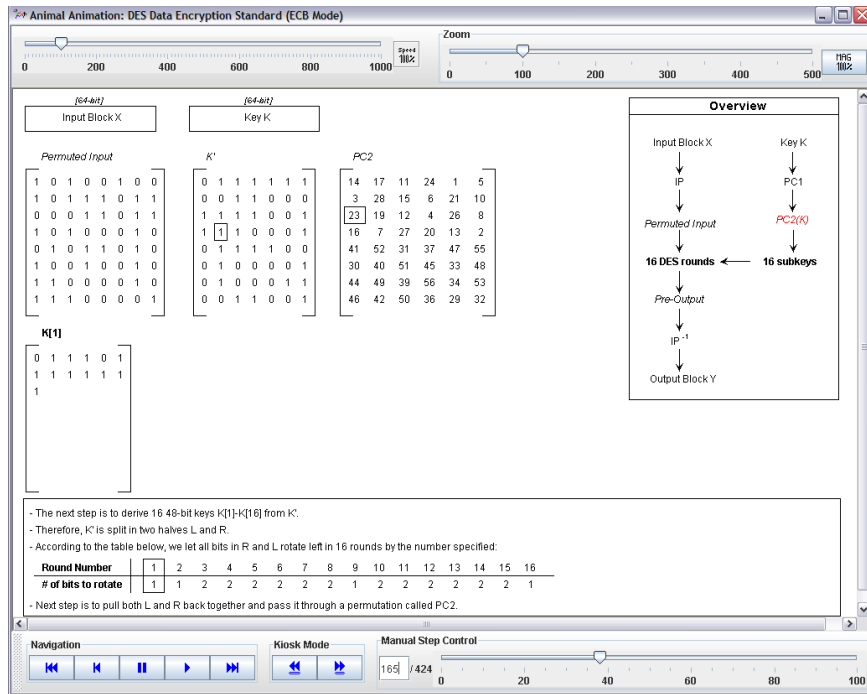


Direct selection of an animation step

Examples (14)

Visualisation of symmetric encryption methods using ANIMAL (2)

Visualization of DES encryption



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)

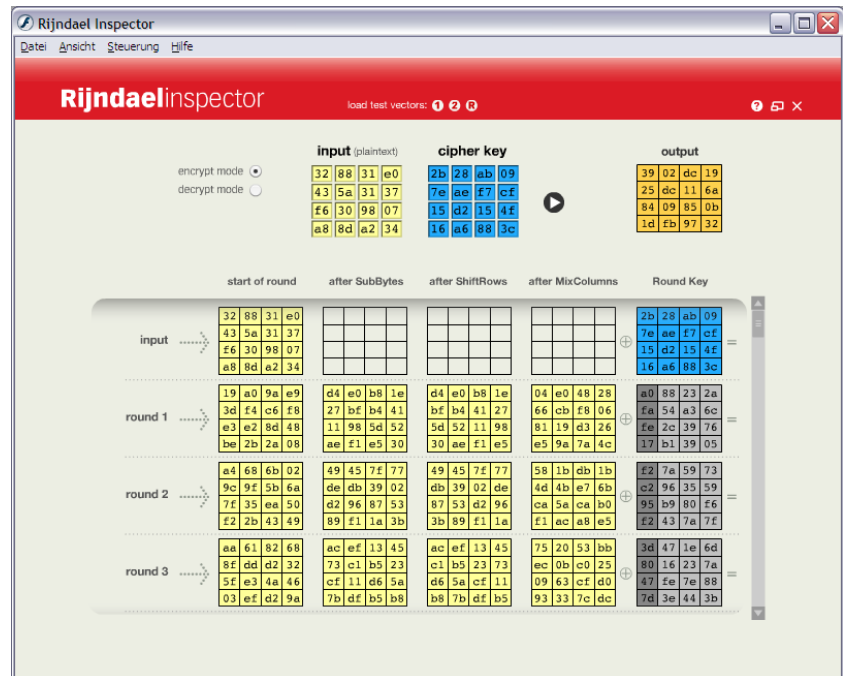
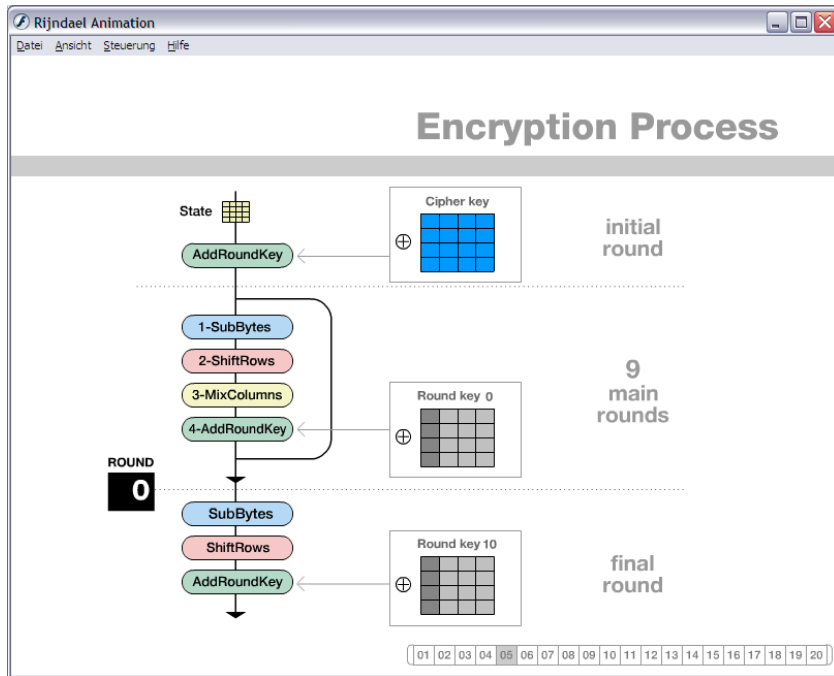
Visualisation of AES (Rijndael cipher)

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" \ "Visualisation of Algorithms" \ "AES" \ "Rijndael Animation" or "Rijndael Inspector"

Examples (16)

Visualisation of the Enigma encryption

The screenshot shows the 'Enigma Simulation' application window. At the top, there are three circular rotor displays, each with a letter highlighted in a colored box (B, F, P). Below these are three keyboard layouts. On the right side, there are two rows of letters (A-Z) with colored squares below them, and a red and a green arrow pointing to them. At the bottom, there are input and output text fields, a status bar, and a URL. Various icons are present in the top right and bottom right corners.

Labels and arrows pointing to specific features:

- Select rotors**: Points to a yellow icon with four rotors in the top right corner.
- Change plugs**: Points to a yellow icon with a plug in the bottom right corner.
- Show settings**: Points to a yellow icon with a key in the bottom right corner.
- Reset Enigma to initial state or random state**: Points to a yellow icon with a key and a plus sign in the bottom right corner.
- Change rotor setting**: Points to a '< >' button above the first rotor.
- Input of plaintext**: Points to the 'Input:' text field containing 'THIS TEXT IS NOT ENCRYPTED'.
- Output of encrypted text**: Points to the 'Output:' text field containing 'LIUGFGYLWXSERZRZDZTRBK'.
- Additional HTML online help**: Points to a '? Help' button in the center of the interface.

Text in the interface:

Enigma Simulation
Datei Ansicht Steuerung Hilfe

Input: THIS TEXT IS NOT ENCRYPTED

Output: LIUGFGYLWXSERZRZDZTRBK

Status Highlighted wires show encryption steps.

www.enigmaco.de enigma v6.0

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

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

1.

3.

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

4.

Close

Examples (18)

Hash demonstration

Sensitivity of hash functions to plaintext modifications

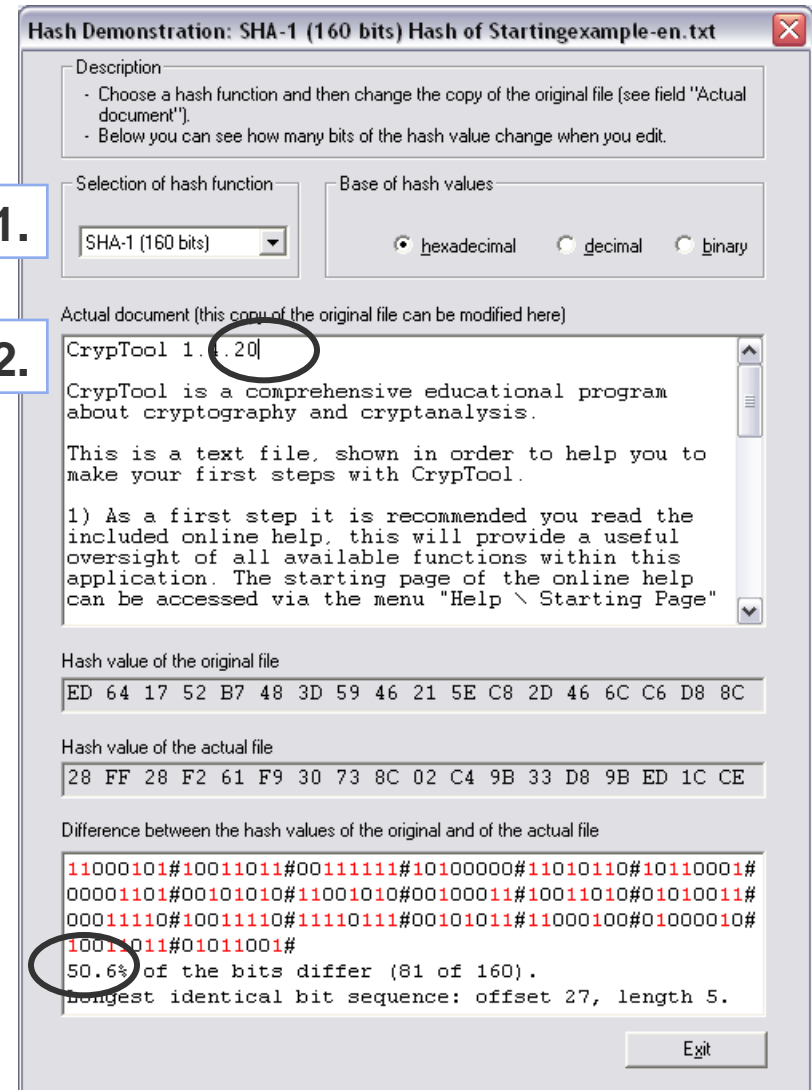
1. Select a hash function
2. Modification of characters in plaintext

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-Demo“



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

The screenshot shows a window titled "NT" with a menu bar containing "Calculators", "Navigation", "Glossaries", and "Help". The main content area has a green background and is titled "3.2 Fermat Test" in bold. In the top right corner of the content area, it says "page 4 of 11".

Each prime p passes a test that results from Fermat's [Little Theorem](#):
Try for a $b \in \{2, \dots, p-1\}$, if $b^{p-1} \equiv 1 \pmod{p}$.

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

Example: $341 = 11 \cdot 31$, even so is $2^{340} \equiv 1 \pmod{341}$.
A passed test gives no information, one repeats it with a different base b :

$n =$ $2^{n-1} \equiv 1 \pmod{n}$ Test passed
GCD(b, n) = 1 b

Definition: Let n be a composite number, b coprime to n .
If $b^{n-1} \equiv 1 \pmod{n}$, 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 $b \in \{1, \dots, n\}$ coprime to n . [Proof](#)

At the bottom of the window, there are navigation buttons: a back arrow, a home arrow, a forward arrow, and a double forward arrow. To the right of these buttons is the text "(Go on to the next page.)".

Menu: „Indiv. Procedures“ / „Number Theory - Interactive“ /
„Learning tool for number theory“

Examples (20)

Point addition on elliptic curves

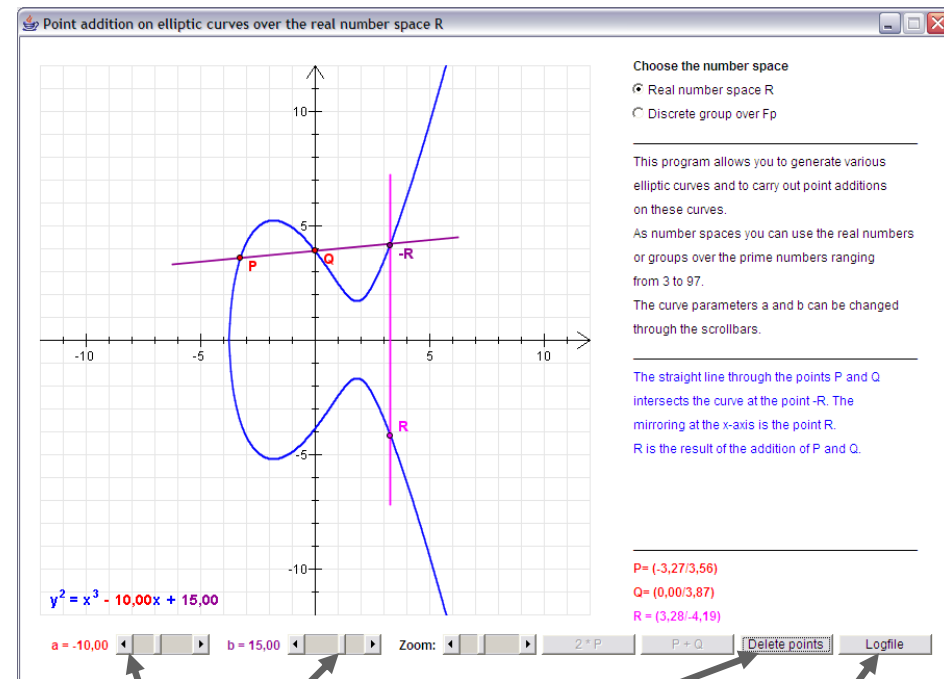
- Visualisation of point addition on elliptic curves
- Foundation of elliptic curve cryptography (ECC)

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



Change curve parameters

Delete points

Log file of calculations

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

Examples (21)

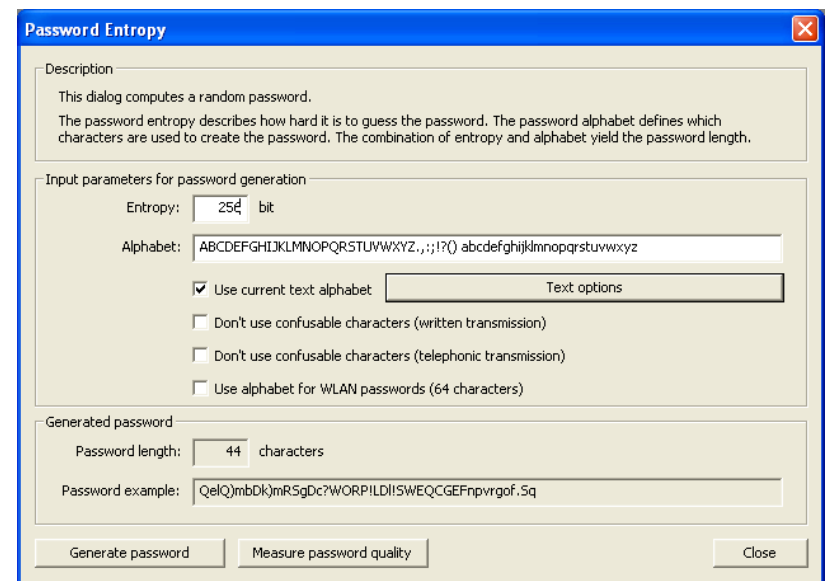
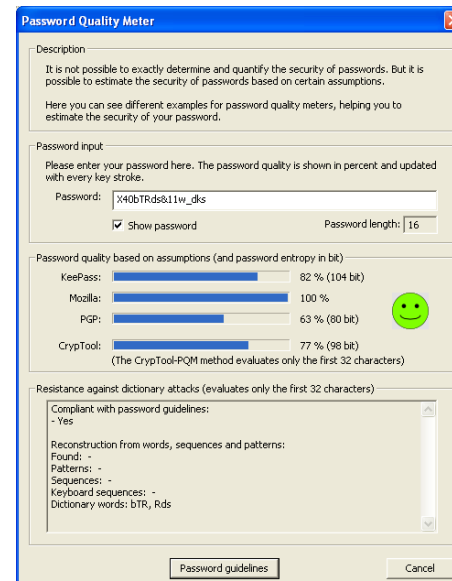
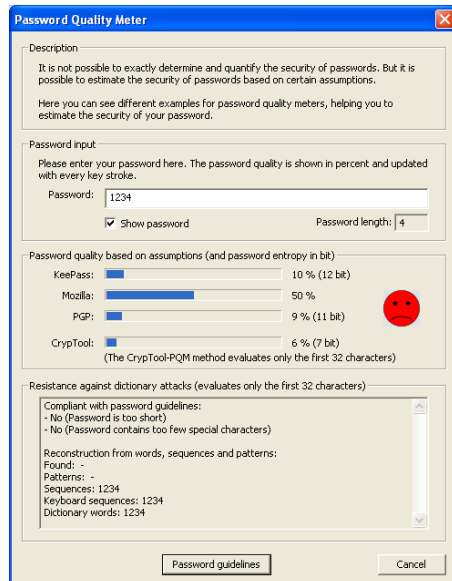
Password Quality Meter (PQM) 1

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**

Password: **X40bTRds&11w_dks**



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 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 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)

68ac78dd40bbefd*
0123456789ab****
98765432106*****
0000000000*****
000000000000****
abacadaba*****
dddddddddd*****

Encrypted text (Hex)

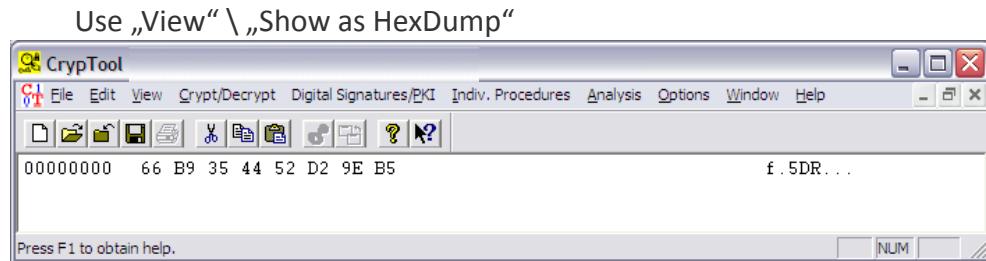
66b9354452d29eb5
1f0dd05d8ed51583
bcf9ebd1979ead6a
8cf42d40e004a1d4
0ed33fed7f46c585
d6d8641bc4fb2478
a2e66d852e175f5c



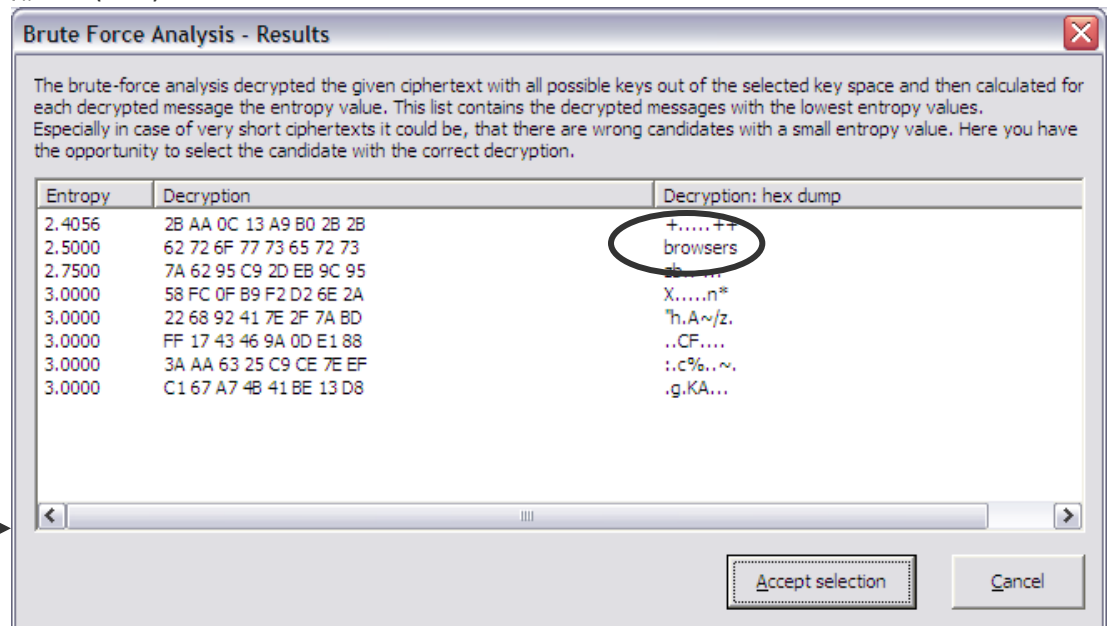
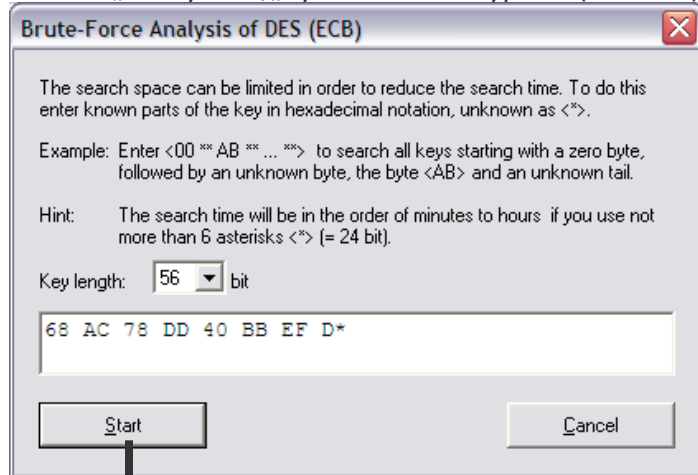
Examples (22)

Brute-force analysis 2

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



Examples (23)

CrypTool Online Help 1

Help for CrypTool

Ausblenden Zurück Vorwärts Abbrechen Aktualisieren Startseite Drucken Optionen

Inhalt Index Suchen

Zu suchendes Schlüsselwort:

Lattice reduction

- Known-plaintext attack
- Lattice reduction
- Liability (exclusion)
- License terms
- Line wrap
- Links
- Literature
- MARS encryption algorithm
- MD2 hash value
- MD4 hash value
- MD5 hash value
- Menu (overview of all menus)
- Message authentication code (MAC)
- Miracel
- Modular transformation
- Modulo operator
- Monoalphabetic substitution encryp
- Network authentication
- N-gram
- Nihilist encryption algorithm
- NIST
- Normal distribution
- NSA
- NTL
- Number Shark
- Number system
- Number theory
- Offset
- One-time pad
- OpenGL
- OpenPGP
- OpenSSL
- Options
- Overview/Subsumption/Broader Con
- Parent window
- Password
- Pattern search
- Periodicity analysis
- Permutation encryption algorithm
- PGP
- Phase space visualization
- Phi function of Euler
- PIN
- PKCS #412

Anzeigen

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

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

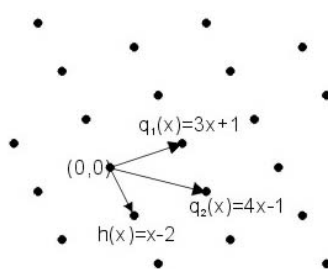
Factoring with a Hint	Attacks RSA with lattice reduction algorithms, if a part of one of the primes of N is known.
Attack on Stereotyped Messages	Attacks RSA with lattice reduction algorithms, if a part of the original cleartext of an intercepted ciphertext is known and if e is small.
Attack on Small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

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

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

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

Example:



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

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:

Examples (23)

CrypTool Online Help 2

Help for CrypTool

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

Chi² 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

Anzeigen

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.

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

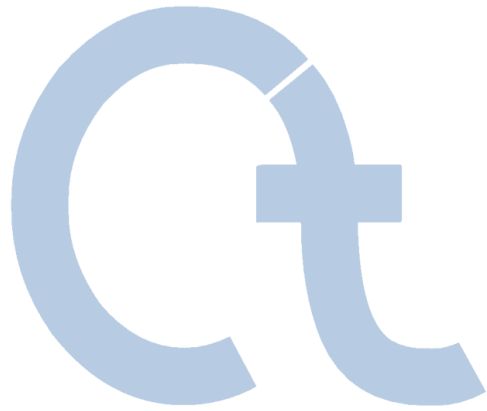
Get the characters from Base64 coding table. (defined in an IETF standard)

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
<ul style="list-style-type: none">Arbitrary binary data can be represented with a 6-bit char set.<ul style="list-style-type: none">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 enlargement of 100 % when encoding to hexadecimal values).	<ul style="list-style-type: none">No support for distribution of big files.Enlargement of about 33 % (in comparison to the original file). <u>Only UUencode:</u>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 1.4.20 (see readme file)

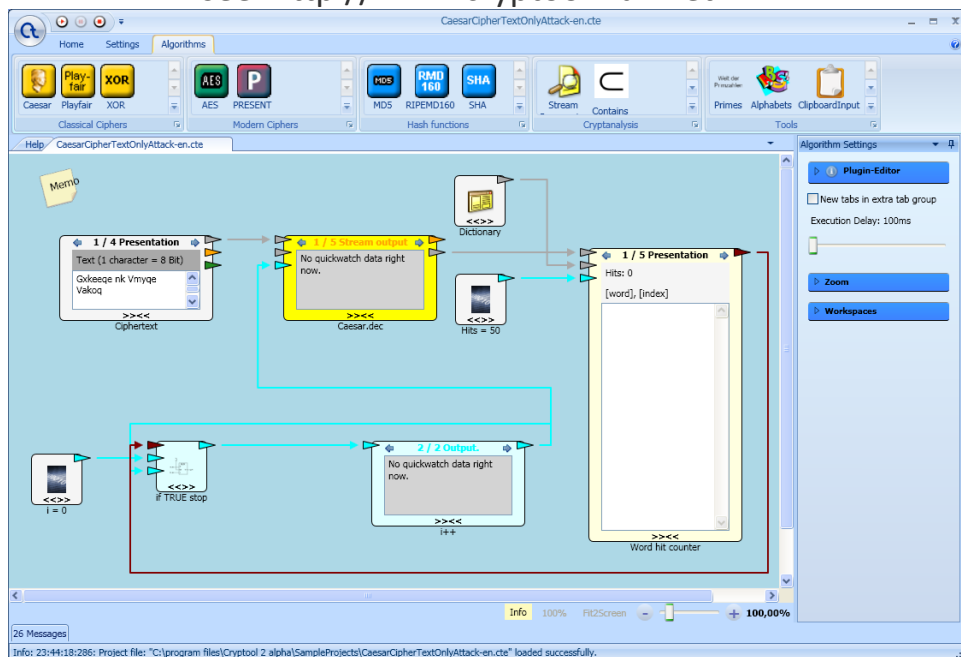
CT 1.x	Mass pattern search
JCT	Visualisation of interoperability of S/MIME and OpenPGP formats
JCT	Tri-partite key agreements
JCT	Analysis of entropy
JCT	Statistical analysis of block ciphers
CT2	Comprehensive visualisation on the topic of prime numbers
CT2	Demonstration of Bleichenbacher's RSA signature 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 end of July 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, ...
 - Alpha 2 available for users and developers from end of July 2008
3. C2L: Direct port of the C++ version to Linux with Qt4
 - see: <http://www.cryptoolinux.net>



CrypTool 2 (CT2)



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 1.4.20: 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 1.4.20:
 - 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 as well as for the new projects for
 - 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

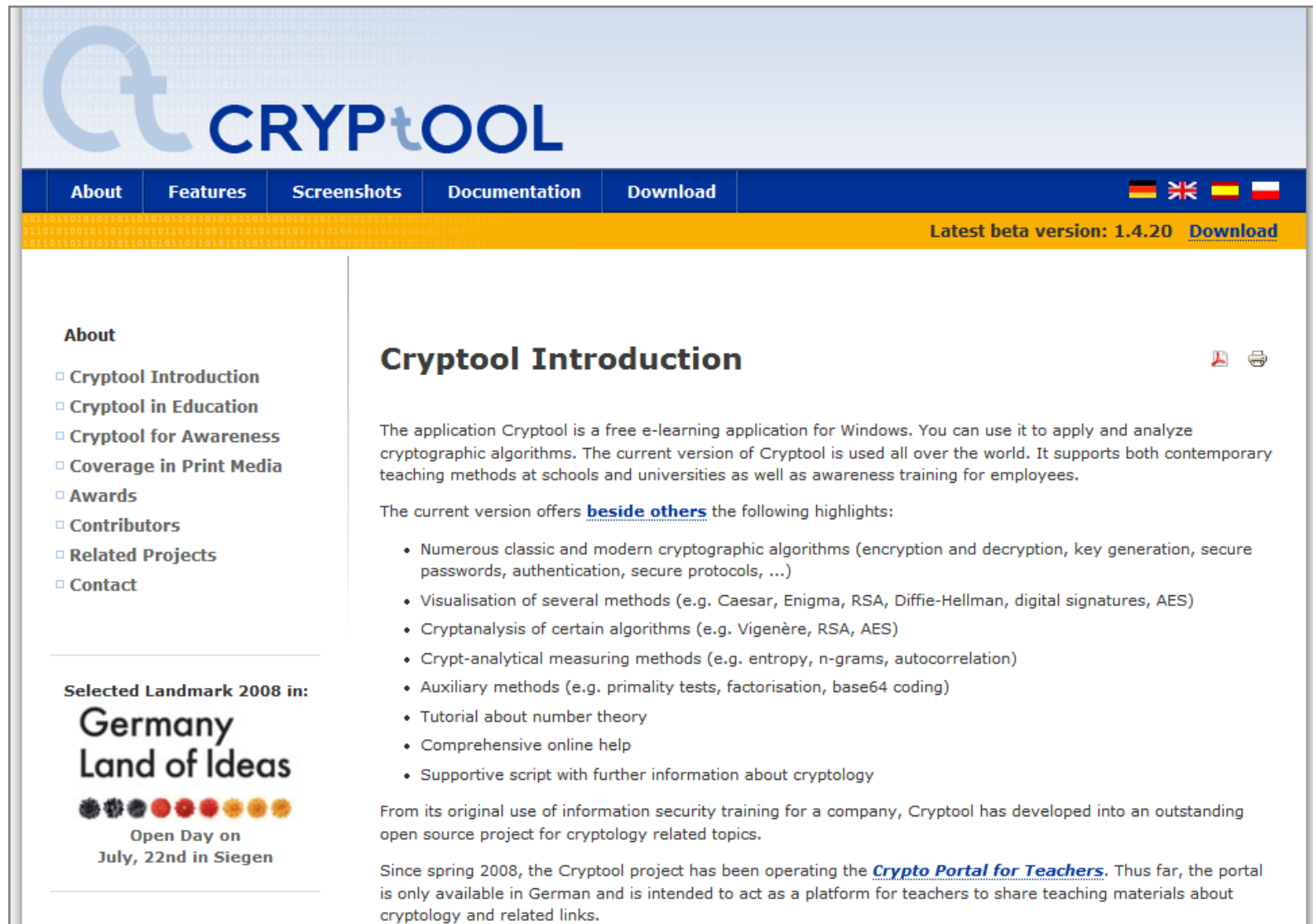
additional contacts: See readme within the CrypTool folder



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
- 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)



The screenshot shows the homepage of the Cryptool website. At the top, there is a navigation bar with links for 'About', 'Features', 'Screenshots', 'Documentation', and 'Download'. To the right of these links are flags for Germany, the United Kingdom, France, and Poland. Below the navigation bar, a yellow banner announces the 'Latest beta version: 1.4.20' with a 'Download' link. The main content area is divided into two columns. The left column contains an 'About' section with a list of links: 'Cryptool Introduction', 'Cryptool in Education', 'Cryptool for Awareness', 'Coverage in Print Media', 'Awards', 'Contributors', 'Related Projects', and 'Contact'. Below this list is a section titled 'Selected Landmark 2008 in: Germany Land of Ideas' featuring a row of colored circles and the text 'Open Day on July, 22nd in Siegen'. The right column features a 'Cryptool Introduction' section with a paragraph describing the application as a free e-learning tool for Windows. It lists several highlights: numerous classic and modern cryptographic algorithms, visualisation of methods like Caesar, Enigma, RSA, and Diffie-Hellman, cryptanalysis of certain algorithms, crypt-analytical measuring methods, auxiliary methods like primality tests and factorisation, a tutorial about number theory, comprehensive online help, and a supportive script with further information about cryptology. At the bottom of the right column, it mentions that since spring 2008, the project has been operating the 'Crypto Portal for Teachers' and is intended to act as a platform for teachers to share teaching materials.

CRYPTtOOL

About Features Screenshots Documentation Download

Latest beta version: 1.4.20 [Download](#)

About

- [Cryptool Introduction](#)
- [Cryptool in Education](#)
- [Cryptool for Awareness](#)
- [Coverage in Print Media](#)
- [Awards](#)
- [Contributors](#)
- [Related Projects](#)
- [Contact](#)

Selected Landmark 2008 in:

Germany
Land of Ideas

Open Day on
July, 22nd in Siegen

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.

CRYPTOportal
für Lehrer

Über Unterrichtsmaterial Linksammlung Registrierung Cryptool Einloggen

Filterkriterien

Land:
alle Länder

Schultyp:
alle Schultypen

Autor:
alle Autoren

Material enthält folgenden Text:

Filtern Zurücksetzen

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 [...]

Originalpraesentation.pdf 14 mal heruntergeladen

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