

# CrypTool

**A free software program**

- **for creating awareness of IT security issues**
- **for learning about and obtaining experience of cryptography**
- **for demonstrating encryption algorithms and analysis procedures**

[www.cryptool.de](http://www.cryptool.de)  
[www.cryptool.com](http://www.cryptool.com)  
[www.cryptool.org](http://www.cryptool.org)

Claudia Eckert / Thorsten Clausius  
Bernd Esslinger / Jörg Schneider / Henrik Koy



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

# Contents

## Introduction

1. What is CrypTool?
2. Why CrypTool?
3. Audience

## CrypTool Overview

1. Features
2. Software package contents
3. New in release 1.3.xx

## Examples of use

1. Hybrid encryption visualised
2. Digital signature visualised
3. Attack on RSA encryption with short RSA modulus
4. Analysis of encryption used in the PSION 5 PDA
5. Demonstration of weak DES keys
6. Locating key material (keyword: NSAKEY)
7. Attack on digital signature

## Contact addresses

# Introduction

## 1. What is CrypTool?

- a freeware Program with graphical user interface
- a tool for applying and analysing cryptographic algorithms
- with extensible online help, understandable without deep crypto knowledge
- contains nearly all state of the art crypto algorithms
- “playful” introduction to modern and classical cryptography
- not a “hacker tool”

## 2. Why CrypTool?

- origin in Deutsche Bank’s IT security awareness program
- developed in co-operation with universities
- improve IT security related courses in universities and companies

## 3. Audience

- target group: students of computer science, commercial IT and mathematics
- also for: interested computer users and application developers
- prerequisites: secondary school mathematics or programming skills

# CrypTool Overview

## 1. Features

### Cryptography

#### Classical algorithms

- Caesar
- Vigenère
- Hill
- Monoalphabetic substitution
- Homophonic substitution
- Playfair
- Permutation
- Addition
- XOR
- Vernam

#### To facilitate performing text book examples with CrypTool

- alphabet can be configured
- treatment of white space etc. configurable

### Cryptoanalysis

#### Attacks on classical algorithms

- ciphertext only
  - Caesar
  - Vigenère
  - Addition
  - XOR
- known plaintext
  - Hill
  - Playfair
- manual
  - mono-alphabetic substitution

#### Supporting analysis procedures

- entropy, floating frequency
- histogram, n-gram analysis
- auto-correlation
- ZIP compression test

# CrypTool Overview

## 1. Features

### Cryptography

#### Modern symmetric algorithms

- IDEA, RC2, RC4, DES, 3DES
- last round AES candidates
- AES (=Rijndael)

#### Asymmetric algorithms

- RSA with X.509 certificates
- RSA demonstration
  - to facilitate performing text book examples with CrypTool
  - alphabet and block length configurable

#### Hybrid encryption

- RSA combined with AES encryption
- visualised by an interactive data flow diagram

### Cryptoanalysis

#### Brute-force attack on symmetric algorithms

- implemented for all algorithms
- assumption:
  - entropy of plain text small

#### Attack on RSA encryption

- factor RSA modulus
- workable for bit lengths  $\leq 250$

#### Attack on hybrid encryption

- attack on RSA (see below) or
- attack on AES (see above)

# CrypTool Overview

## 1. Features

### Cryptography

#### Digital Signature

- RSA with X.509 certificates
  - signature procedure visualised by an interactive data flow diagram
- DSA with X.509 certificates
- Elliptic curve DSA, Nyberg-Rueppel

#### Hash functions

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

#### Random generators

- SECUDE
- $X^2$  modulo  $N$
- Linear Congruence Generator (LCG)
- Inverse Congruence Generator (ICG)

### Cryptoanalysis

#### Attack on RSA Signature

- RSA modulus factorisation
- workable up to approx. 250 bit

#### Attack on hash function/digital signature

- Generation of hash collisions to ASCII texts

#### Random data analysis

- FIPS-PUB-140-1 test battery
- periodicity, Vitany, entropy
- histogram, n-gram analysis
- auto-correlation
- ZIP compression test

# CrypTool Overview

## 2. Software package contents

### CrypTool program

- all functions integrated in *one* program with uniform graphical user interface
- platforms: Win32 and Linux with WINE emulator
- cryptography based on Secude library ([www.secude.com](http://www.secude.com))
- arbitrary precision arithmetic: Miracl library (<http://indigo.ie/~mscott/>)

### AES-Tool

- standalone program for AES encryption (self extracting)

### Extensive online help (Winhelp)

- context sensitive online help for *all* program functions and all menu items
- detailed usage examples for many program features

### Script (PDF) with background information on

- encryption algorithms • prime numbers • digital signature
- elliptic curves • public key certification • elementary number theory

### Short story “Dialogue of the Sisters” by Dr. C. Elsner

completely bilingual  
English  
German

# CrypTool Overview

## 3. New in release 1.3.xx

**Most important changes (details: see ReadMe-en.txt):**

### **Release 1.3.00 published January 2002**

- completely bilingual English/German
- dialog box consistency and comprehensibility improved
- Windows 9x file size limit removed
- homophonic and permutation encryption
- random generators, random data analysis (FIPS-140-1, periodicity, n-gram)
- AES-Tool: create self-decrypting files (AES)
- demonstration: number theory and RSA crypto system (further improved in 1.3.02)
- PKCS#12 export/import for PSEs

### **Release 1.3.02 published June 2002**

- visualisation of hybrid encryption and decryption
- visualisation of signature creation and verification
- hash value calculation of large files (without loading them into memory)
- visualisation of the sensitivity of hash functions to changes in the hashed data
- short story “Dialogue of the Sisters” by Dr. C. Elsner included



# CrypTool Overview

## 3. New in release 1.3.xx

### Release 1.3.04 published June 2003

- visualisation of Diffie-Hellman key exchange
- attack on digital signature using hash-collisions (birthday paradox)
- brute-force attack on symmetric ciphers improved
- script updated (primes, factorization) and extended (hash functions, ECC, CrypTool menu tree)
- many small improvements (especially online help) and bug fixes

# Examples of use

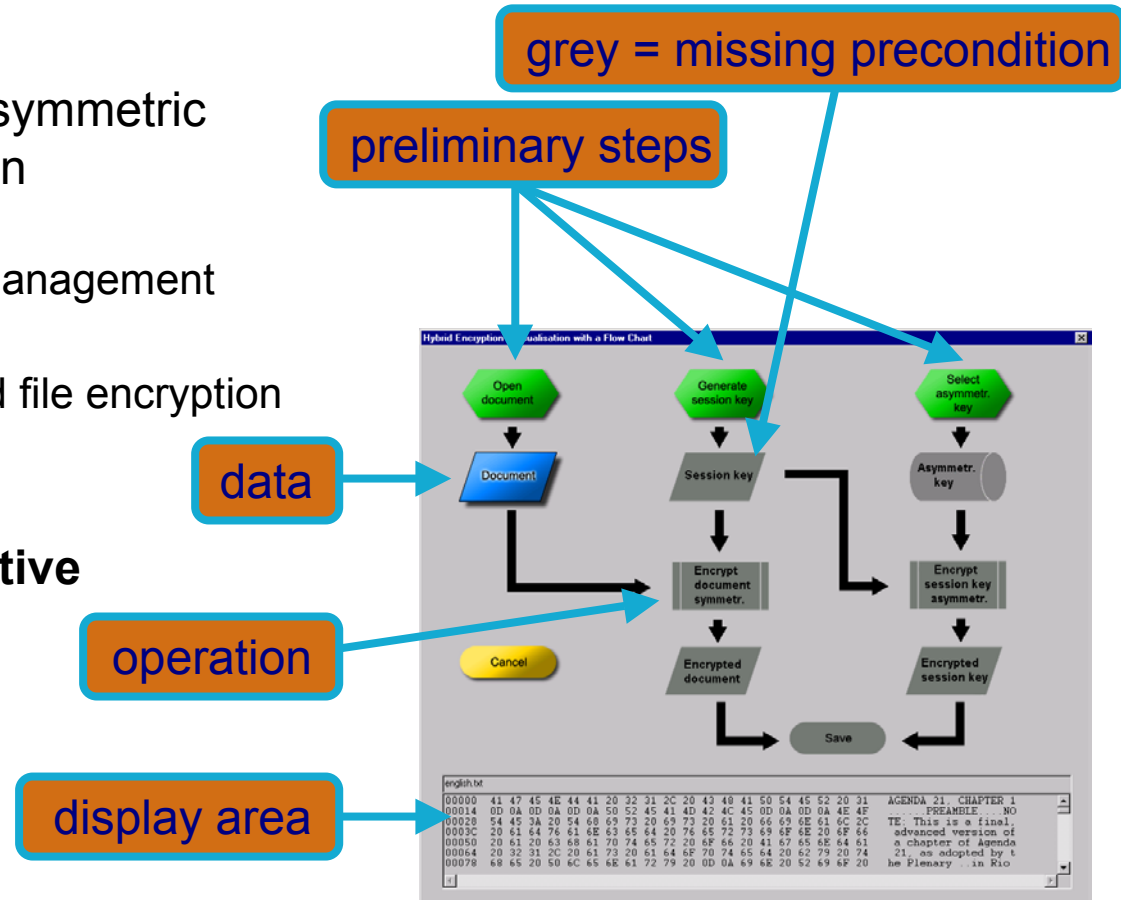
## 1. Hybrid encryption visualised

### Hybrid encryption

- combines advantages of symmetric and asymmetric encryption
  - speed
  - simple and scalable key management
- widely used in practice
  - e-mail (S/MIME, PGP) and file encryption
  - SSL (https)

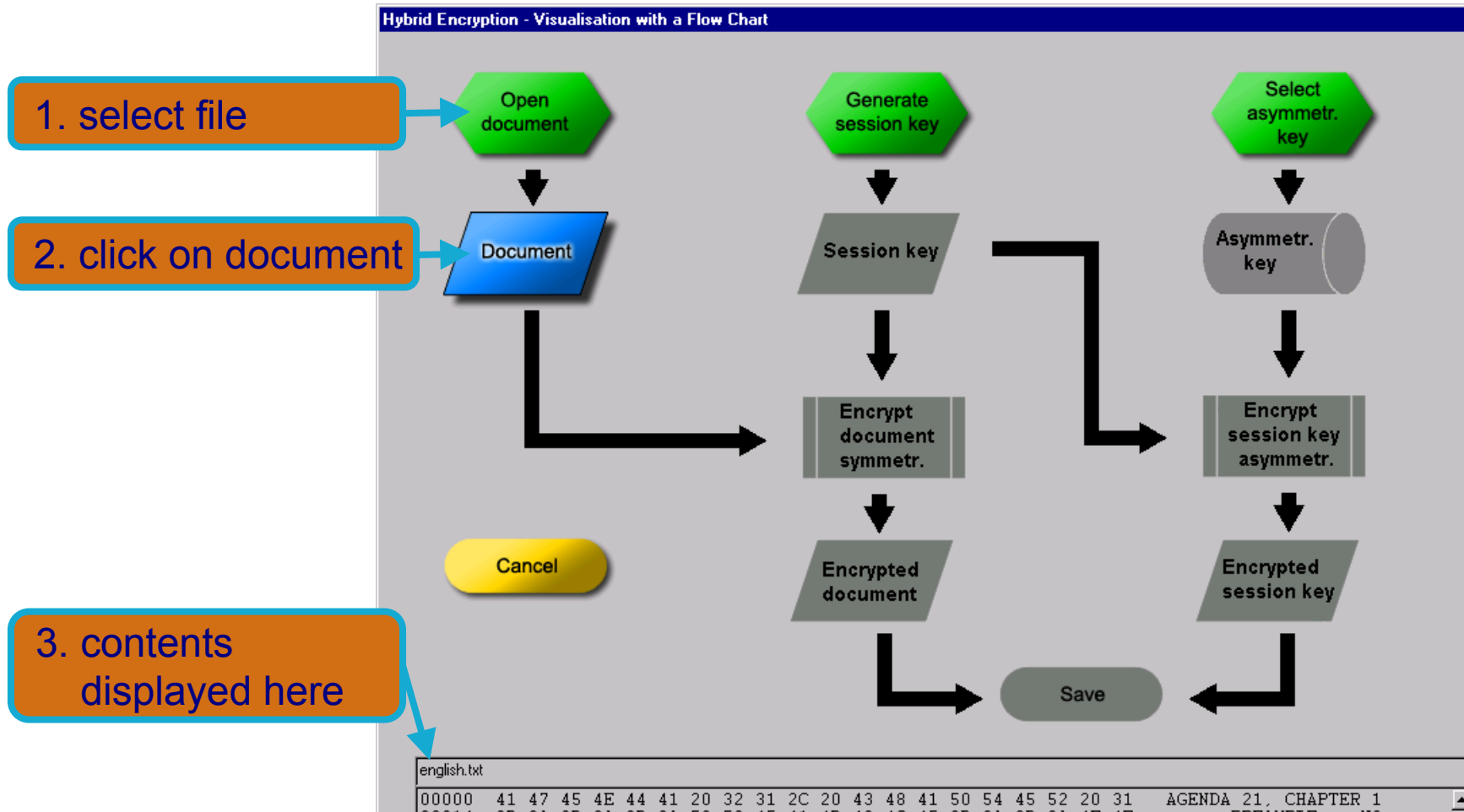
### Visualisation by an interactive data flow diagram

- playful learning leads to deeper understanding



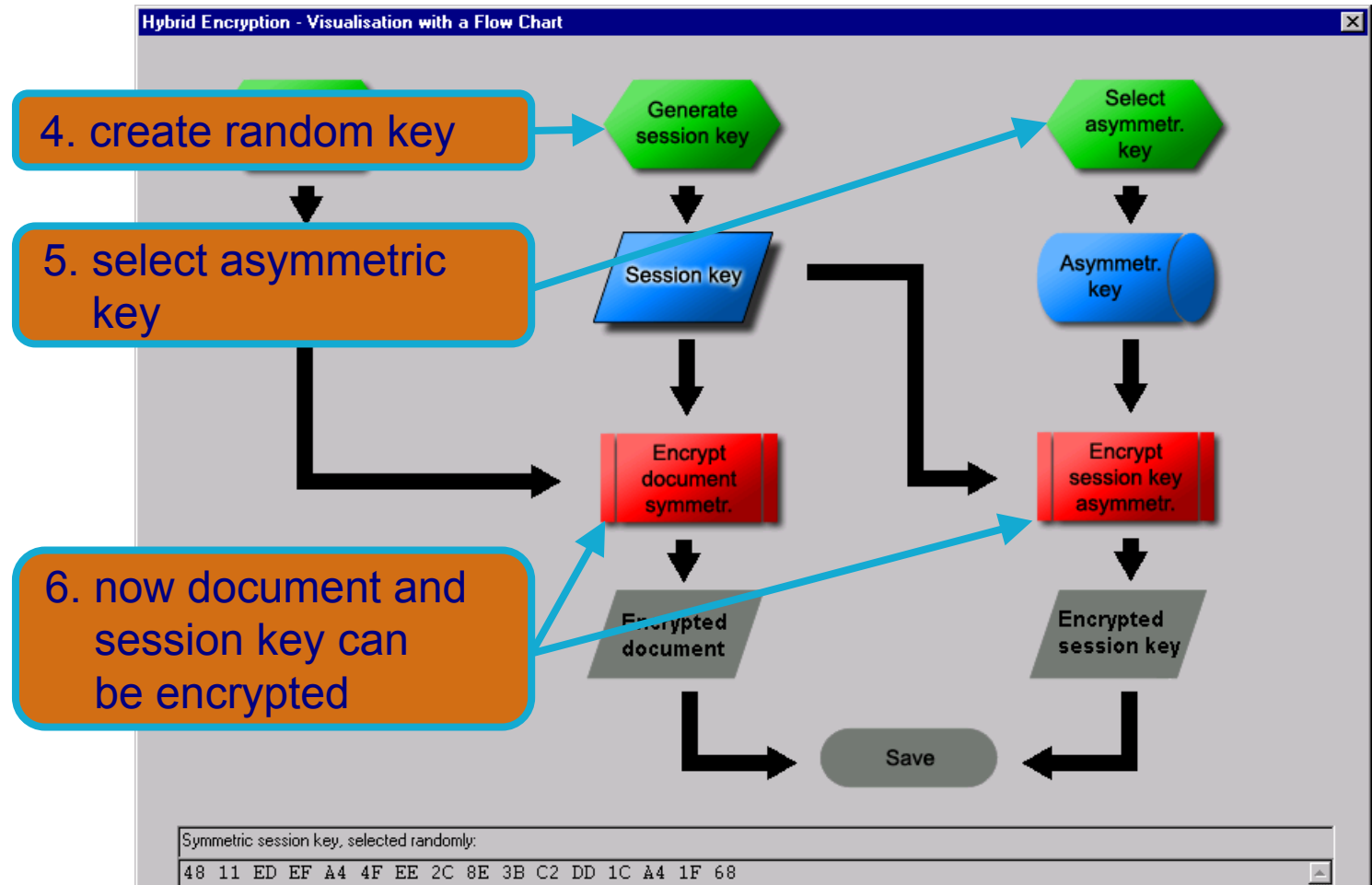
# Examples of use

## 1. Hybrid encryption visualised: Preparation



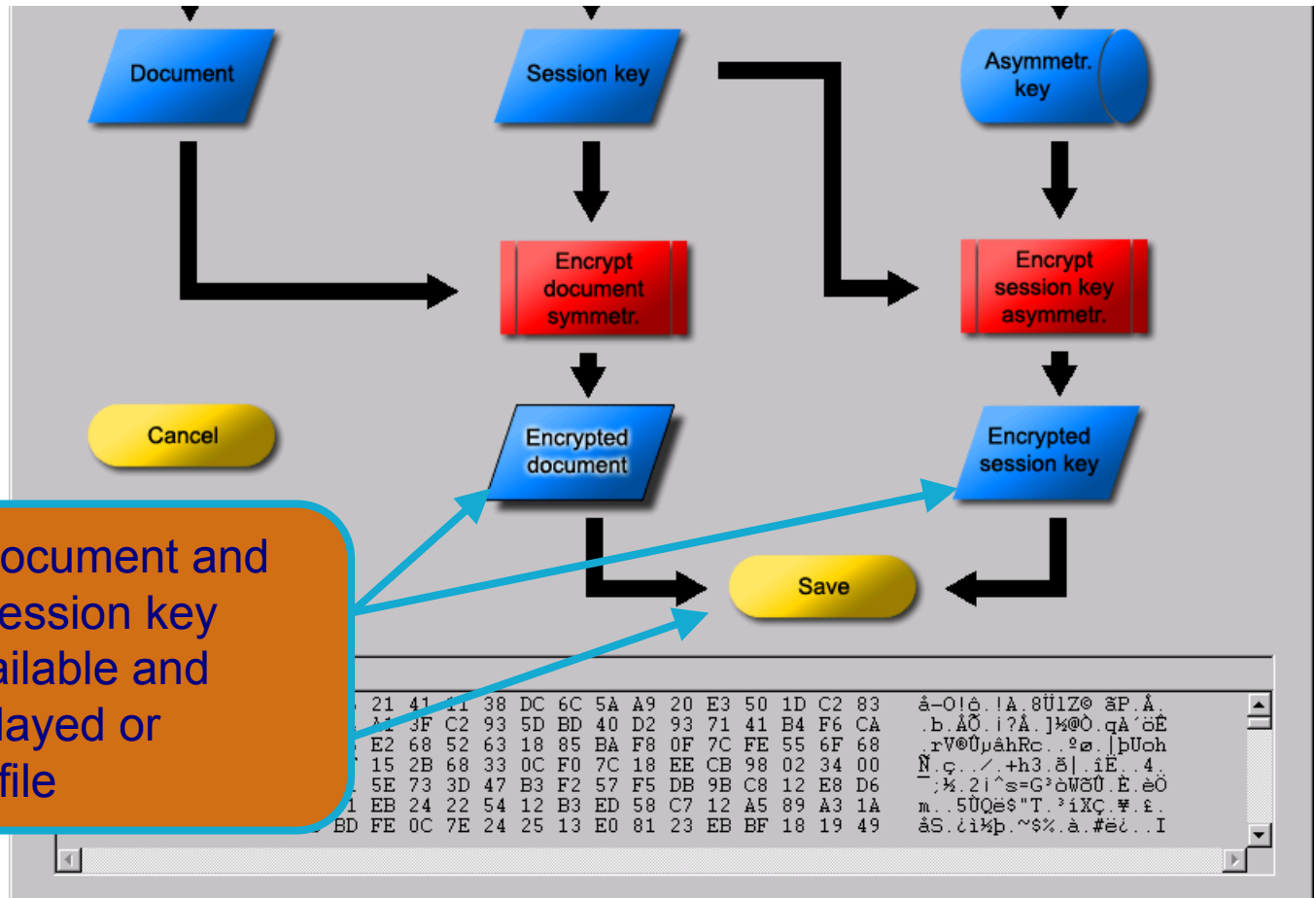
# Examples of use

## 1. Hybrid encryption visualised: Cryptography



# Examples of use

## 1. Hybrid encryption visualised: Result



# Examples of use

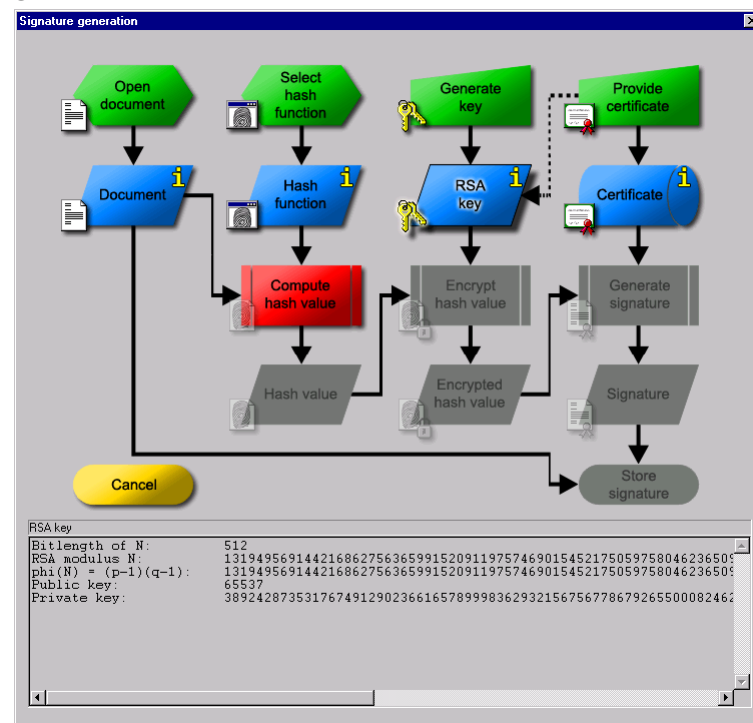
## 2. Digital signature visualised

### Digital signature

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

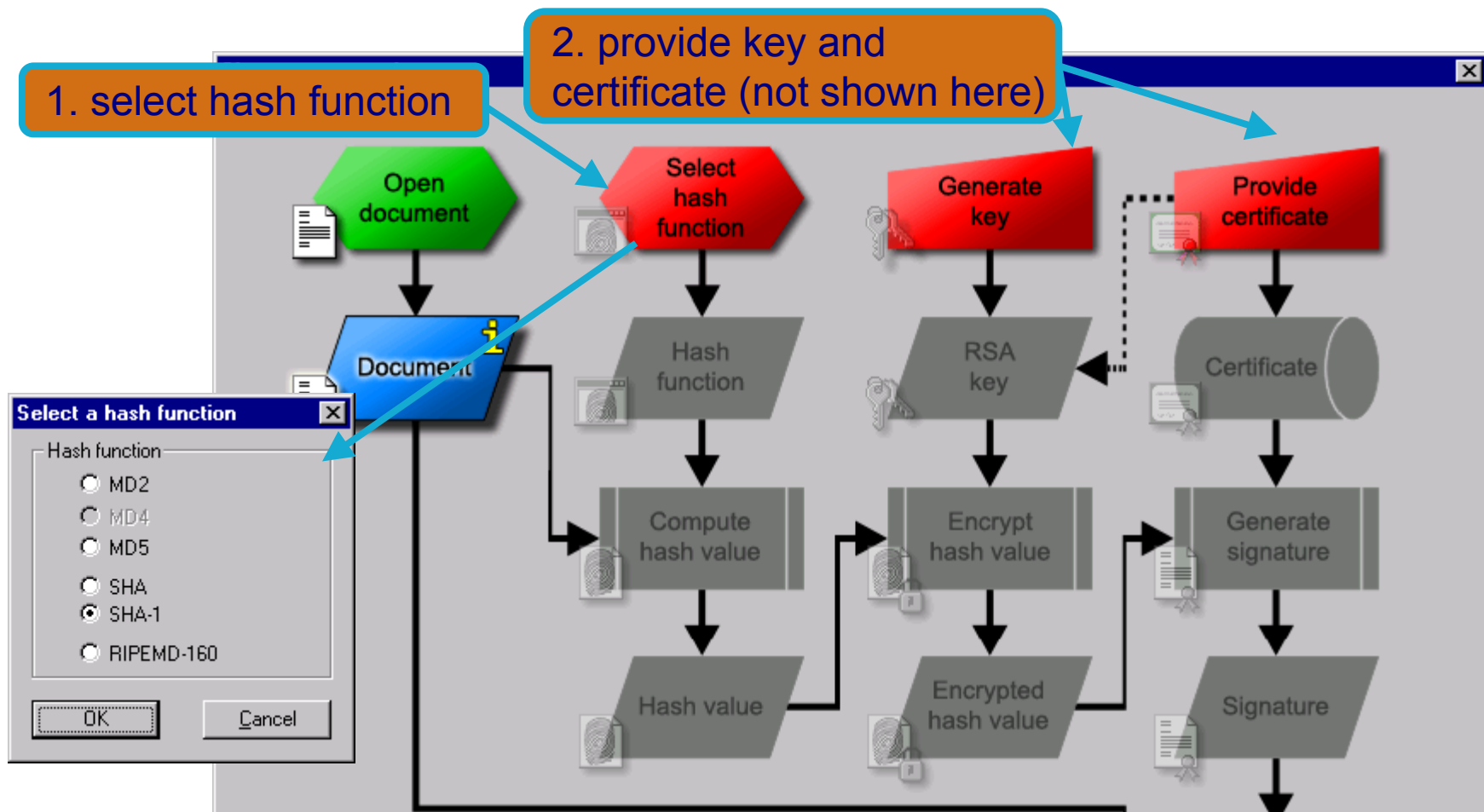
### Visualisation in CrypTool

- interactive data flow diagram
- similar to the visualisation of hybrid encryption



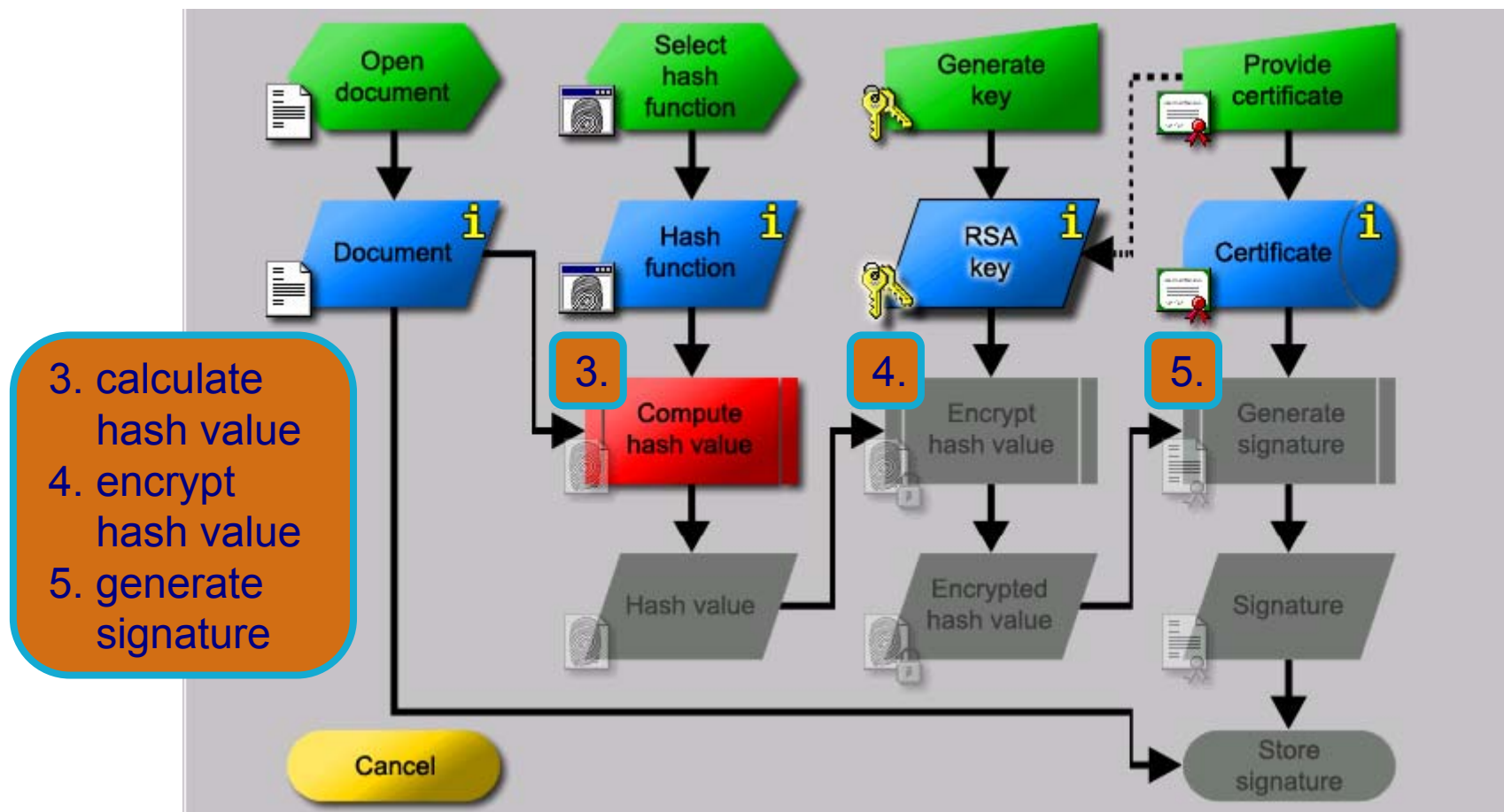
# Examples of use

## 2. Digital signature visualised: Preparation



# Examples of use

## 2. Digital signature visualised: Cryptography



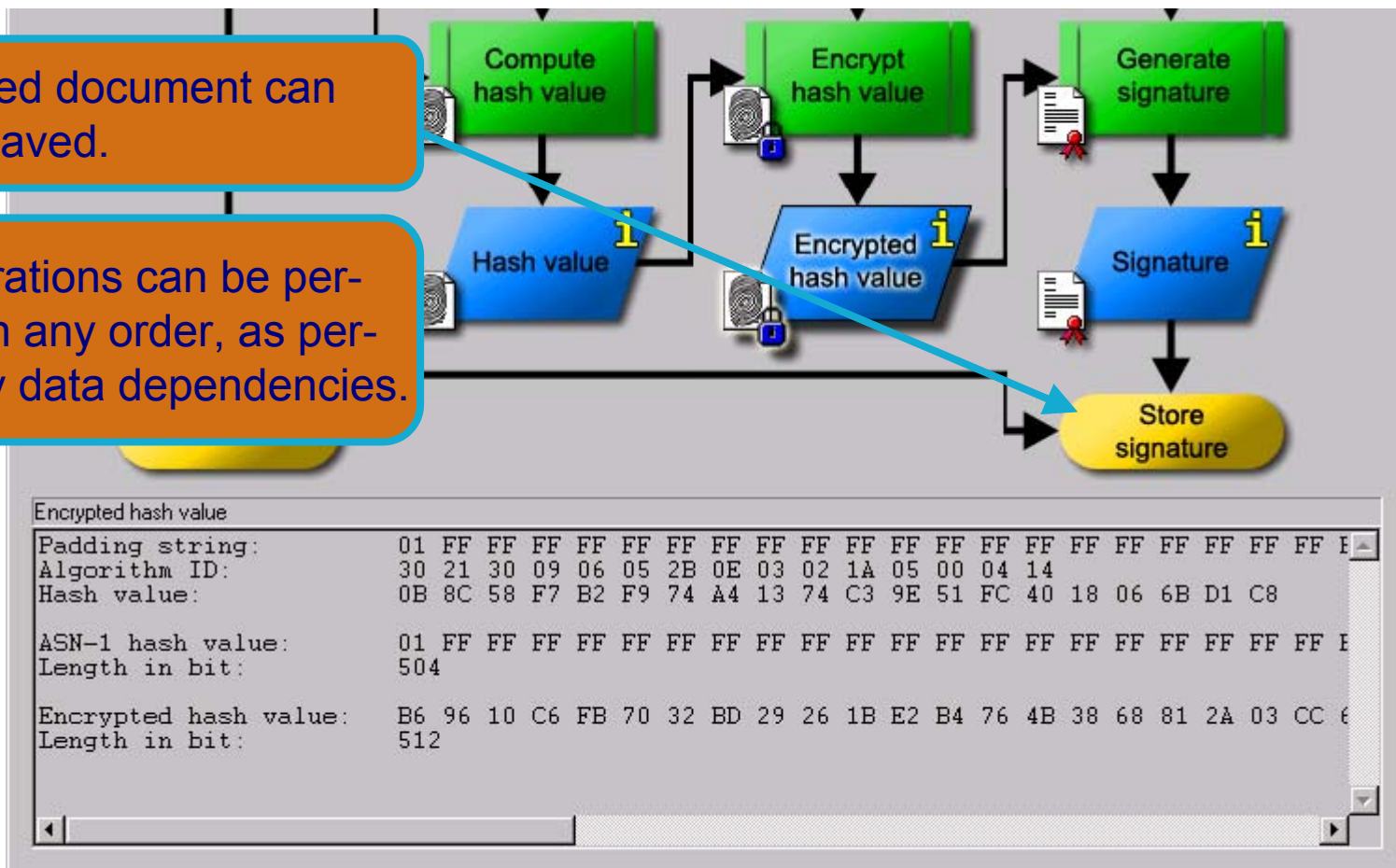


# Examples of use

## 2. Digital signature visualised: Result

The signed document can now be saved.

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



## Examples of use

### 3. Attack on RSA encryption with short RSA modulus

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

- public key
  - RSA modulus  $N = 63978486879527143858831415041$  (95bit, 29 decimal digits)
  - public exponent  $e = 17579$
- cipher text (block length = 14):
  - $C_1 = 45411667895024938209259253423,$   
 $C_2 = 16597091621432020076311552201,$   
 $C_3 = 46468979279750354732637631044,$   
 $C_4 = 32870167545903741339819671379$
- the text shall be deciphered!

**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 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**

## Examples of use:

### 3. Short RSA modulus: enter public RSA parameters

**The RSA Cryptosystem**

☐ 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 $N$	<input type="text" value="63978486879527143858831415041"/>	(public)
$\phi(N) = (p-1)(q-1)$	<input type="text"/>	(secret)
Public key $e$	<input type="text" value="17579"/>	
Private key $d$	<input type="text"/>	

Update parameters

☐ RSA encryption using  $e$  / decryption using  $d$

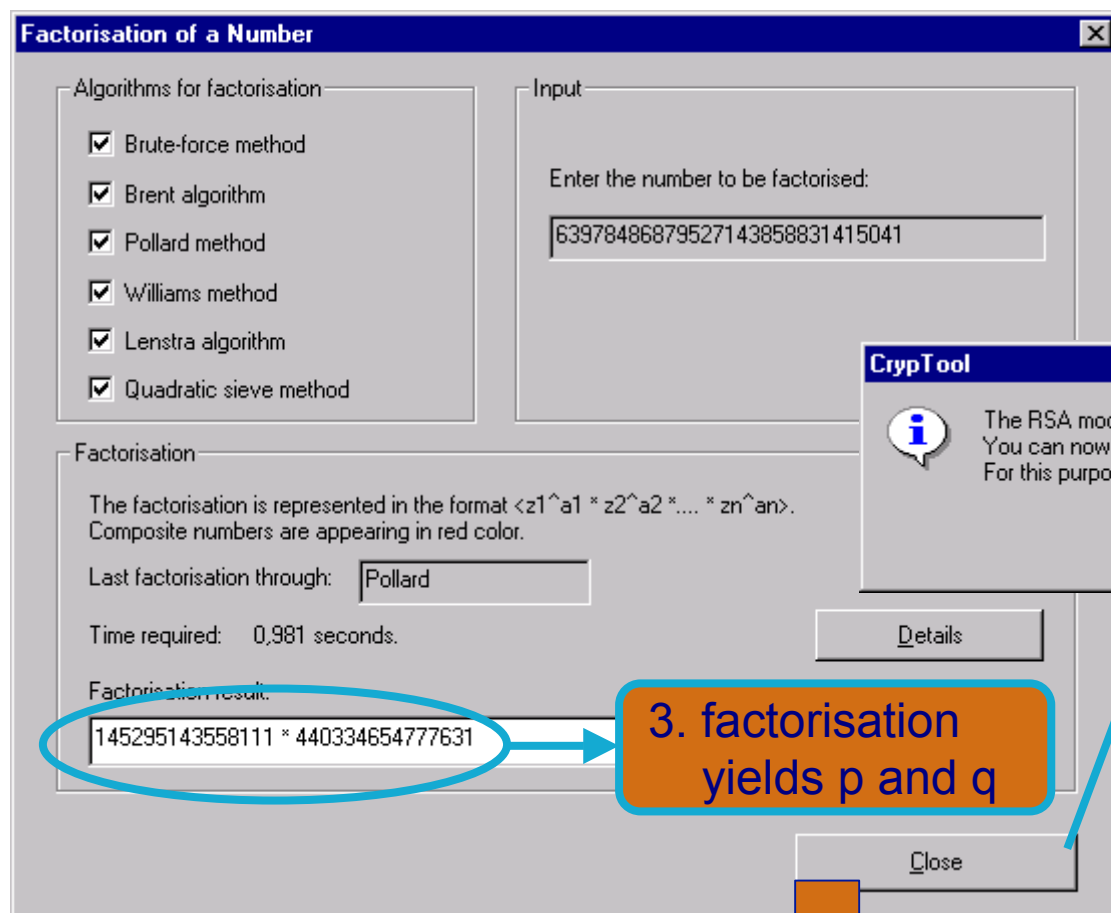
2. factorise

1. enter RSA parameters  $N$  and  $e$



## Examples of use:

### 3. Short RSA modulus: factorise RSA modulus



## Examples of use:

### 3. Short RSA modulus: determine private key d

**The RSA Cryptosystem**

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 will do with the published RSA parameter: 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

Options for alphabet and number system...

Enter either as text or as numbers in the format number[1] # ... # number[n] (numbers of base 1240752)

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

5. adjust options

## Examples of use:

### 3. Short RSA modulus: adjust options

**Options for RSA Encryption**

**Text options**

☐ All 256 ASCII characters

☒ Specify alphabet: Number of characters: 27

ABCDEFGHIJKLMNOPQRSTUVWXYZ

**Block length**

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

Block length in characters:  (Maximum block length 2 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 of use:

## 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

7091621432020076311552201 # 46468979279750354732637631044 # 32870167545903741339819671279

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

0000000000001401202118011200 # 0000000000001421130205181900 # 000000000000011805001301

Output text from the decryption (into segments of size 8; the symbol '#' is used as separator).

NATURAL # NUMBERS # ARE MADE # BY GOD

Plaintext

NATURAL NUMBERS ARE MADE BY GOD

Encrypt Decrypt Close

9. enter cipher text

10. decrypt

# Examples of use

## 4. Analysis of encryption used in the PSION 5 PDA

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



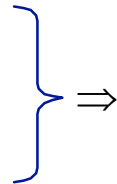
**Starting point: an encrypted file on the PSION**

### Requirements

- encrypted English or German text
- depending on method and key length, 100 bytes up to several kB of text

### Procedure

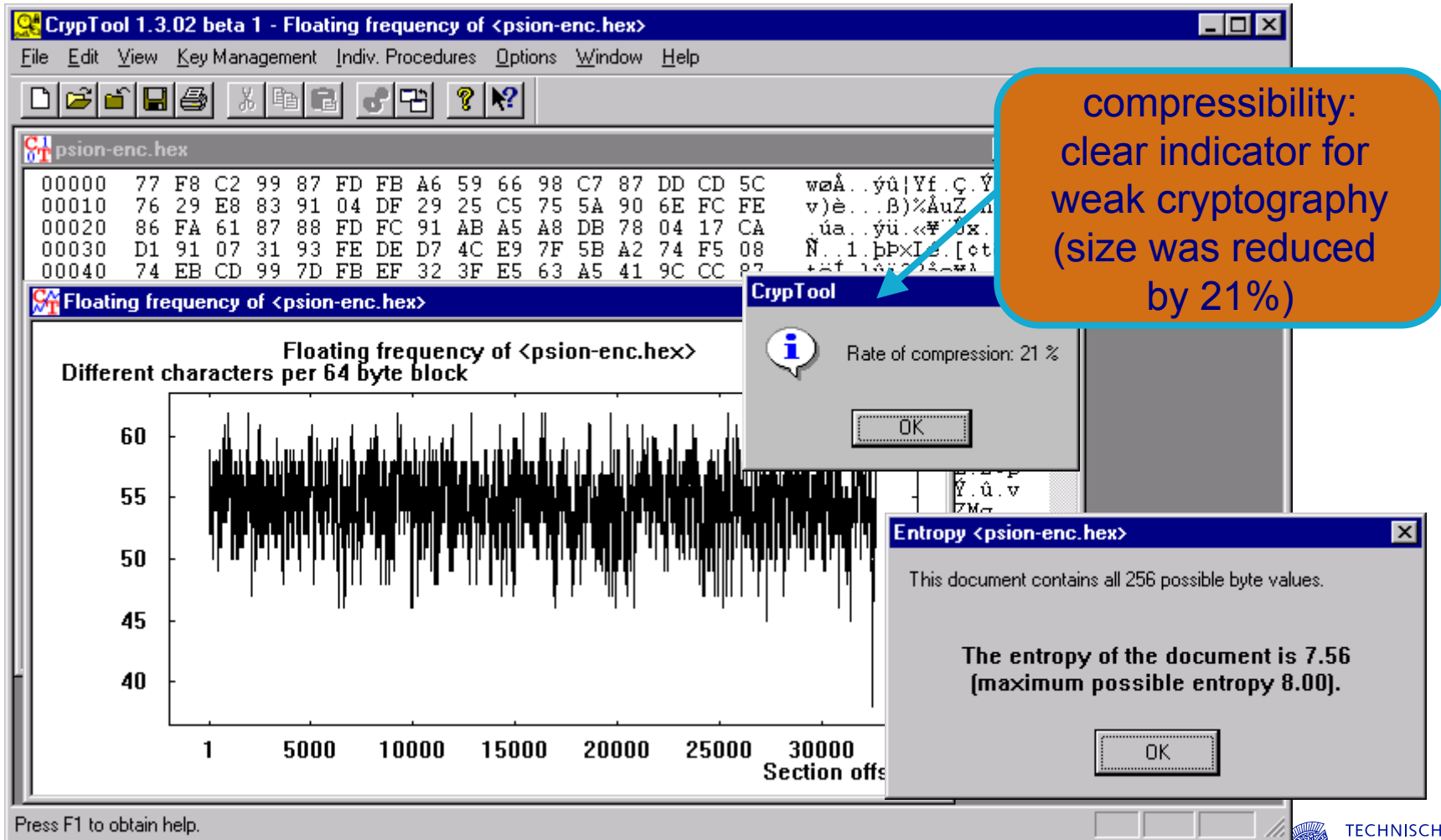
- pre-analysis
  - entropy
  - floating entropy
  - compression test
- auto-correlation
- try out automatic analysis with classical methods



probably classical  
encryption algorithm

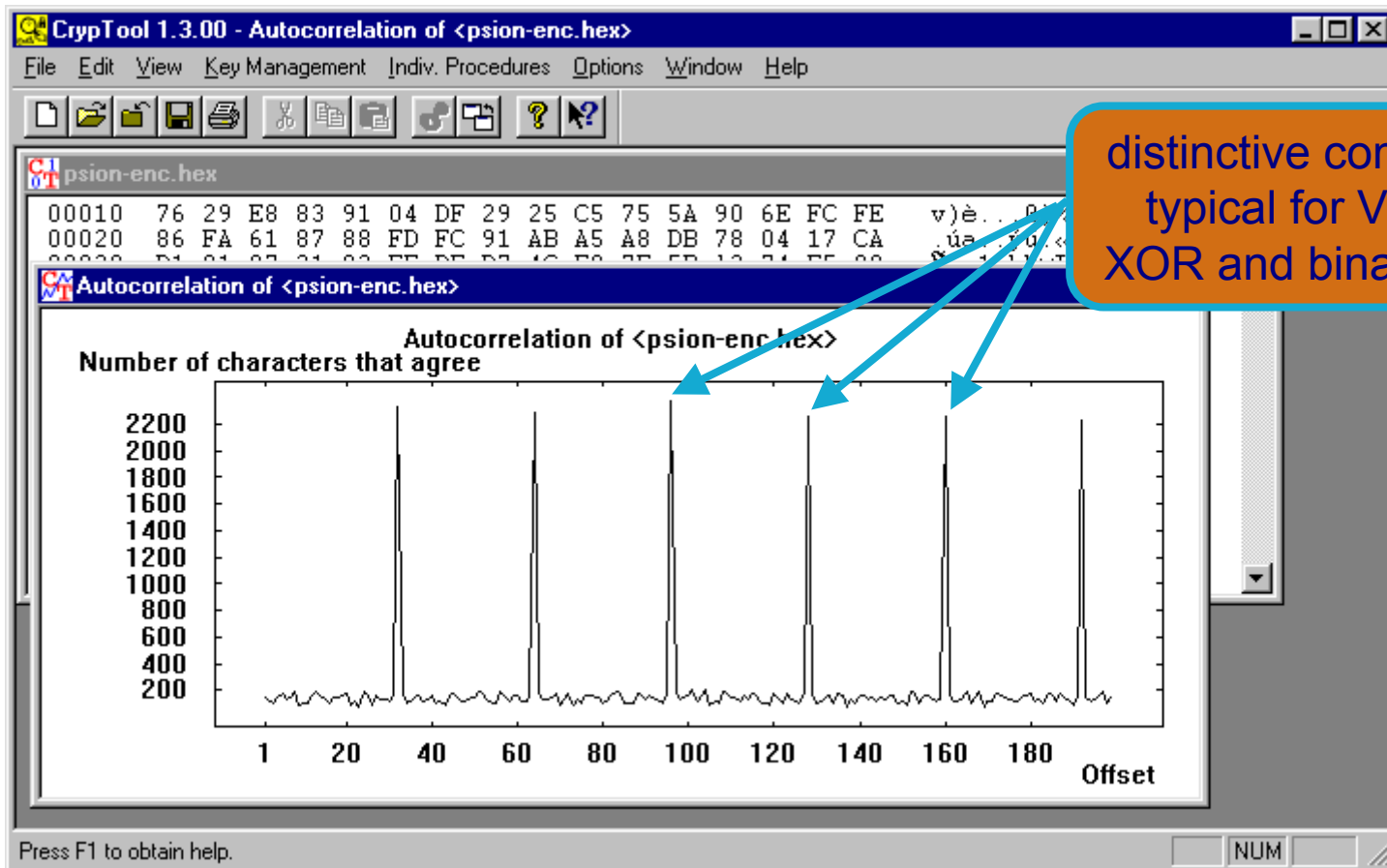


#### 4. PSION PDA: determine entropy, compression test



# Examples of use

## 4. PSION PDA: determine auto-correlation



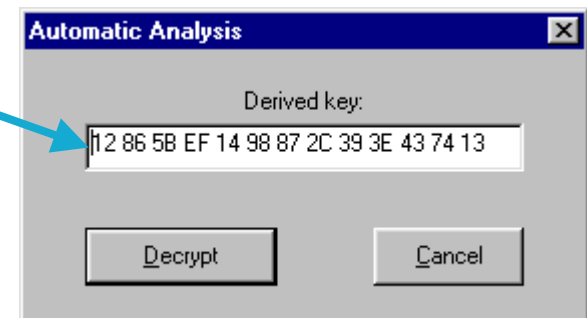
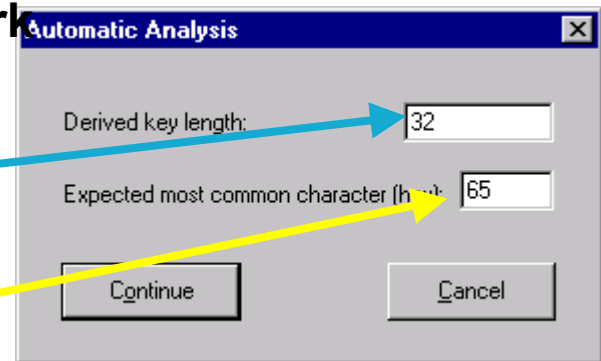
# Examples of use

## 4. PSION PDA: automatic analysis

**Automatic analysis using XOR: does not work**

**Automatic analysis using Binary Addition:**

- CrypTool calculates the key length using auto-correlation: 32 bytes
- The user can choose which character is expected to occur most frequently: “e” = 0x65 (ASCII code)
- Analysis calculates the most likely key (based on the assumptions about distribution)
- Results: good, but not perfect

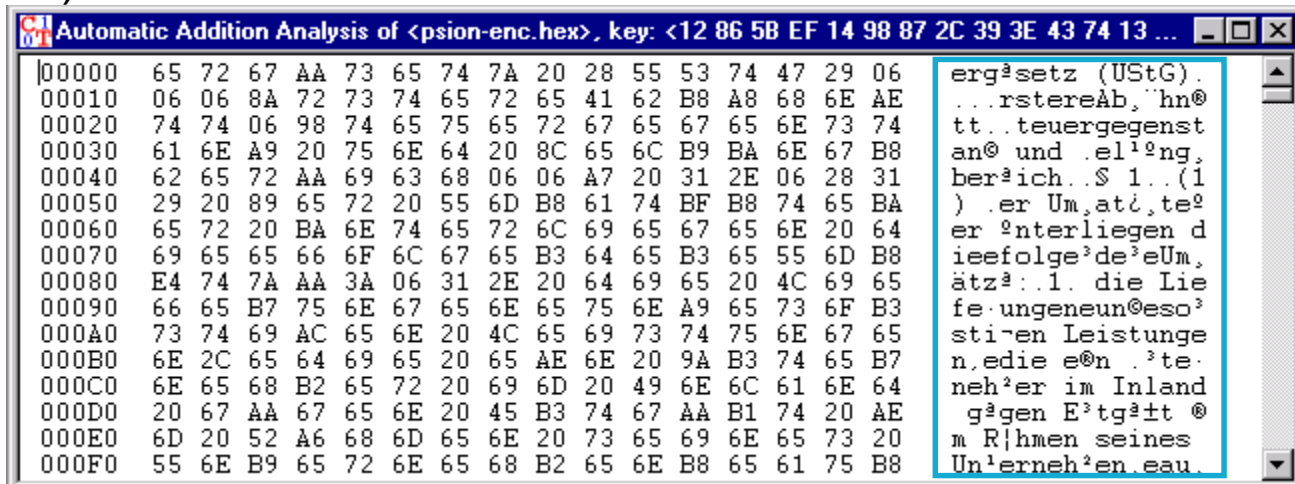


# Examples of use

## 4. PSION PDA: results of automatic analysis

### Results of automatic analysis with assumption “binary addition”:

- results good, but not perfect: 24 out of 32 key bytes correct.
- the key length was correctly determined.
- the password entered was not 32 bytes long.  
⇒ PSION Word derives the actual key from the password.
- manual post-processing produces the encrypted text  
(not shown)



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

Offset	Hex	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 .el¹ng,
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,at³,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: ungeneun@eso³
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 R³hmen seines
000F0	55 6E B9 65 72 6E 65 68 B2 65 6E B8 65 61 75 B8	Un¹erneh²en.eau.

# Examples of use

## 4. PSION PDA: determining the remaining key bytes

**Copy key to clipboard during automatic analysis**

**In automatic analysis hexdump,**

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

**In encrypted initial file hexdump,**

- 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...

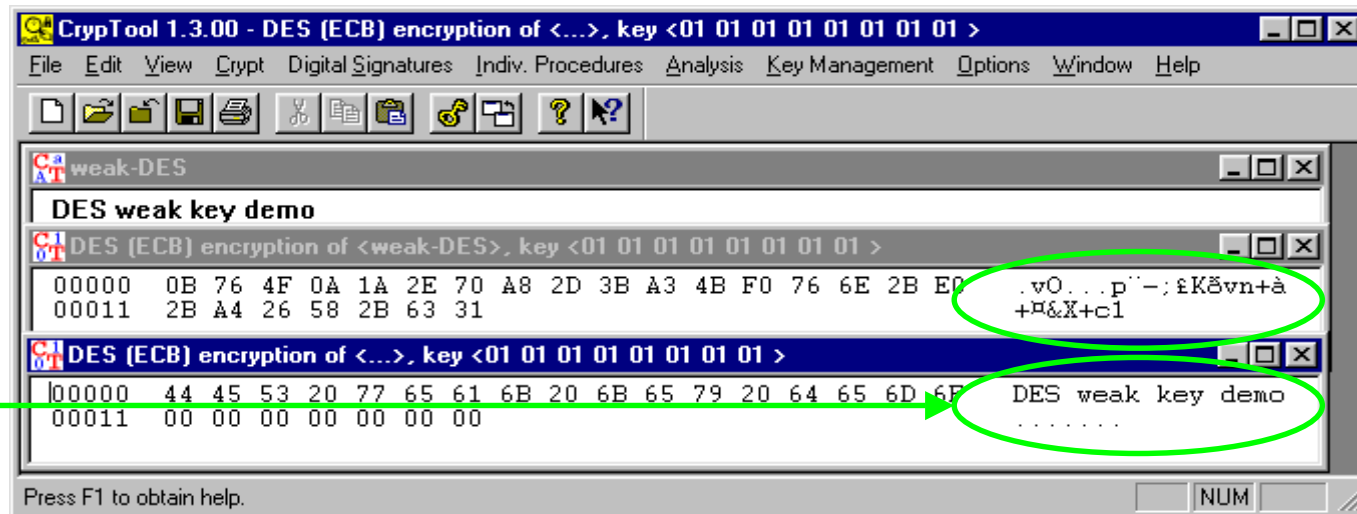
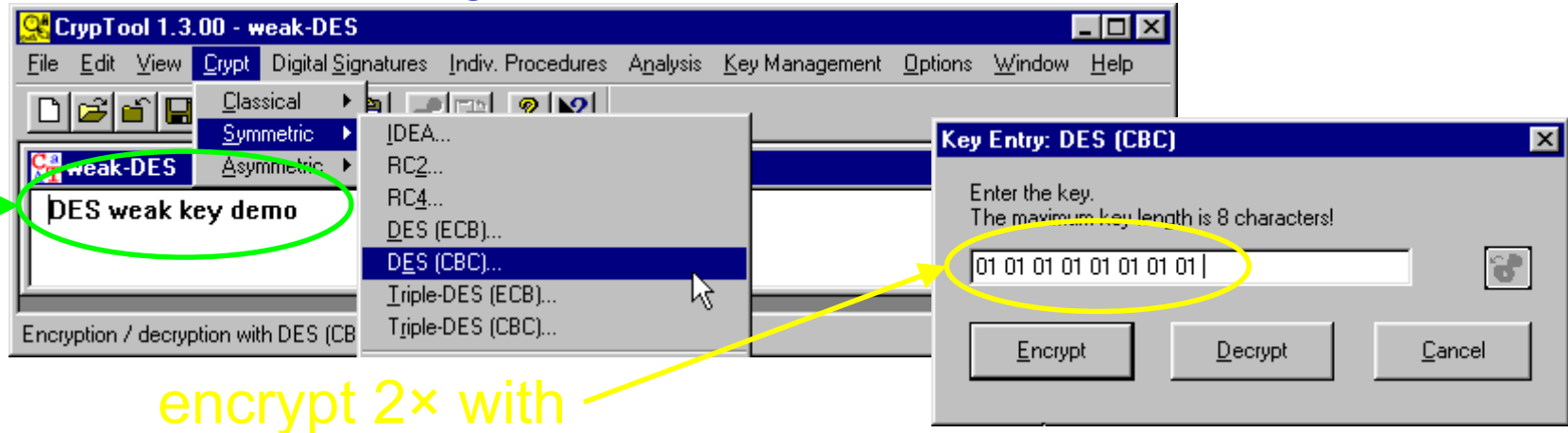
**Decrypt encrypted initial document using binary addition**

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

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

# Examples of use

## 5. Weak DES keys



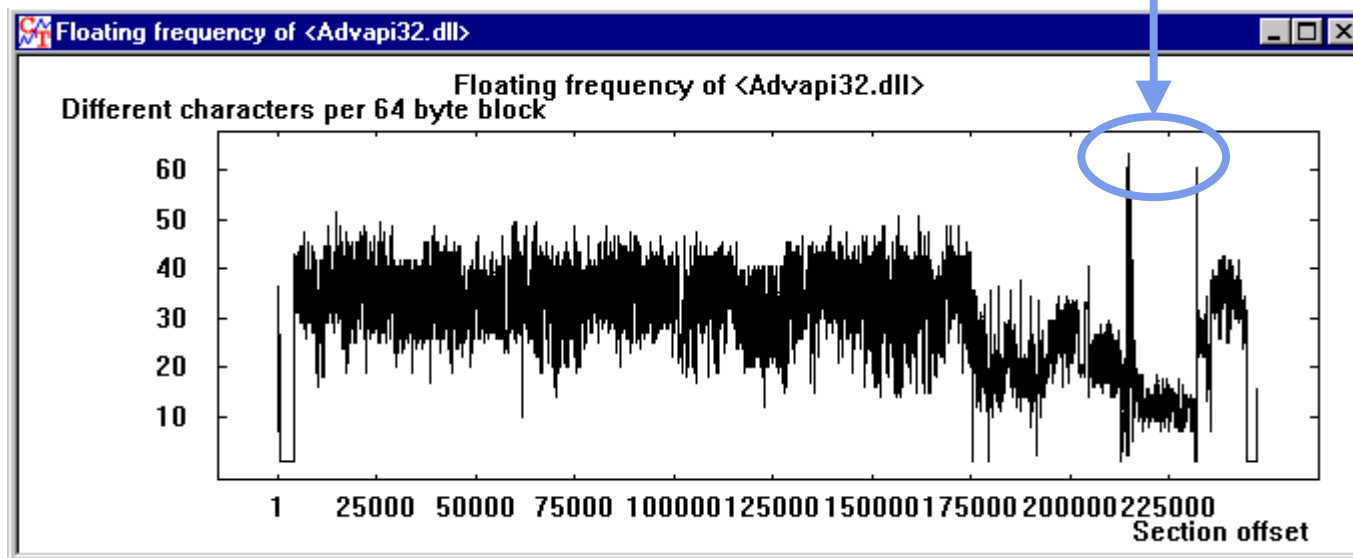
# Examples of use

## 6. Locate key material

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

### Background:

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



# Examples of use

## 7. Attack on digital signature: idea

### Attack on digital signatures of printable ASCII-text based on hash collision

#### Idea:

- An ASCII-text can be extended by an indefinite number of **non-printable** characters without changing the readable content
- Insert **non-printable** characters in an “evil” and a “harmless” text until you get a hash collision
- Attack uses the Birthday Paradoxon (birthday attack)
- Generic attack on any hash function
- Attack can be parallelized
- Implementation based on a work from Jan Blumenstein: *Methoden und Werkzeuge für Angriffe auf die digitale Signatur*, 2003



# Examples of use

## 7. Attack on digital signature: idea

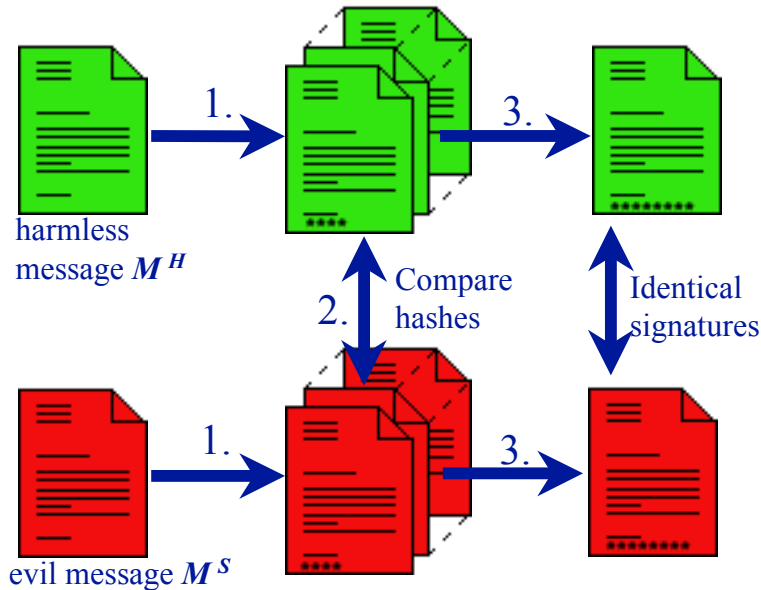
### Attack on the digital signature of an ASCII text based on hash collision search

#### 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 by Jan Blumenstein as part of his bachelor thesis "*Methods and tools for attacks on digital signatures*" (German), 2003.

# Examples of use

## 7. Attack on digital signature: idea (2)



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

# Examples of use

## 7. Attack on digital signature: attack

**Angriff auf den Hashwert der digitalen Signatur**

Der hier implementierte Angriff auf die digitale Signatur beruht auf dem Versuch, zwei verschiedene Nachrichten mit gleichem Hashwert zu finden.

1. "Harmlose" Datei auswählen

Die "harmlose" Datei ist eine Nachricht, von der der Angreifer vermutet, dass der Unterzeichner sie digital signieren wird.

C:/Programme/CrypTool-medida/Original.txt Suchen ...

2. "Gefährliche" Datei auswählen

Die "gefährliche" Datei ist eine Nachricht, von der der Angreifer nach erfolgreichem Angriff behaupten wird: "Diese Nachricht wurde vom Unterzeichner digital signiert."

C:/Programme/CrypTool-medida/Faelschung.txt Suchen ...

Suche starten / Optionen festlegen

Mit "Nachrichtenpaar suchen" starten Sie den Versuch, zu den oben eingestellten Nachrichten zwei Modifikationen zu finden, die denselben Hashwert haben. Der Sinn (Semantik) der Nachrichten ändert sich während der Suche nicht, da zum Modifizieren nicht darstellbare bzw. Formatierungszeichen verwendet werden.

Unter "Optionen" können Sie das Hashverfahren, die Anzahl der für den Vergleich der Hashwerte herangezogenen Bits sowie verschiedene Modifikationsverfahren auswählen.

4. Nachrichtenpaar suchen

Optionen ... Dialog beenden

**Optionen für den Angriff auf den Hashwert der digitalen Signatur**

Hashfunktion

Wählen Sie eines der sechs Hashverfahren sowie die Anzahl der Bits, die für den Vergleich der Hashwerte herangezogen werden sollen.

☐ MD2 ☐ MD4 ☒ MD5 ☐ SHA ☐ SHA-1 ☐ RIPEMD-160

Signifikante Bitlänge 32

Optionen für die Nachrichtenmodifikation

Entscheiden Sie, nach welchem Vorgehen die Nachrichten modifiziert werden sollen.

☐ Leerzeichen einfügen ☒ Zeichen anhängen ☐ Zeichen entfernen

Übernehmen Standard wiederherstellen

**Ein Nachrichtenpaar wird gesucht ...**

Run 1  
Zyklussuche  
Fortschritt: 23% Restzeit: 00:00:35

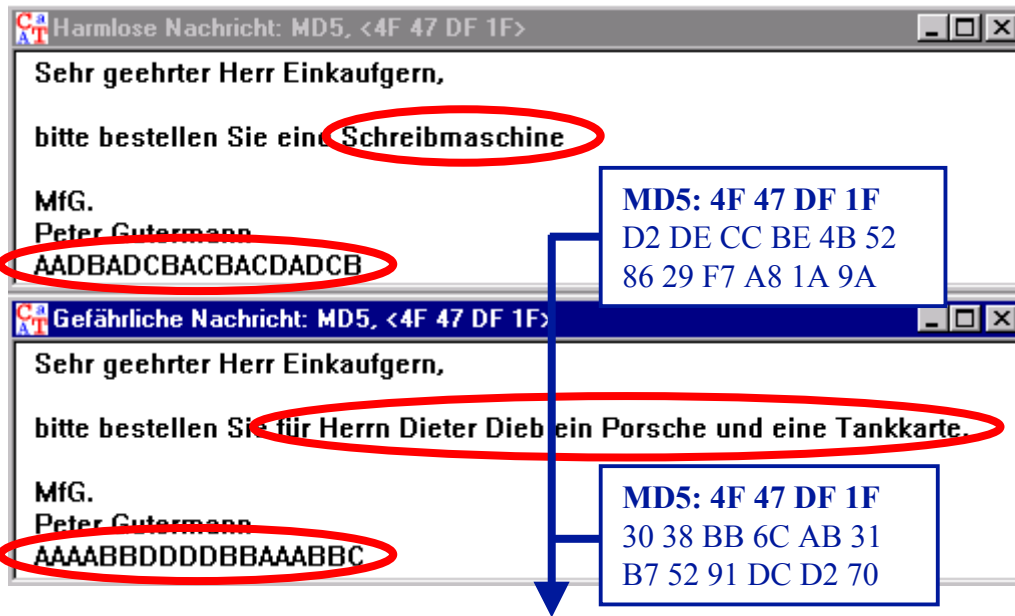
**Ein Nachrichtenpaar wird gesucht ...**

Run 1  
Kollisionssuche  
Fortschritt: 27% Restzeit: 00:01:08

Abbrechen

# Examples of use

## 7. Attack on digital signature: results



### 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 massively parallel search!

# Further development

## Work in process

- visualisation of challenge-response authentication
- attack on single-sided authentication with CR and weak encryption
- mass pattern search

## Planned for near future

- visualisation of SSL protocol
- visualisation of Man-in-the-Middle attack
- demonstration of a side-channel attack

## Planned for remote future

- visualisation of different security protocols (e.g. Kerberos)
- visualisation of attacks on these different security protocols
- port to Linux or Java
- many more ideas can be found in the readme file, chapter 6

## Contact addresses

**Prof. Dr. Claudia Eckert**  
Technical University Darmstadt  
Faculty of Computer Science  
IT Security  
Wilhelminenstr. 7

64283 Darmstadt, Germany  
[claudia.eckert@sec.informatik.tu-darmstadt.de](mailto:claudia.eckert@sec.informatik.tu-darmstadt.de)

**Thorsten Clausius**  
Technical University Darmstadt  
[thorsten.clausius@sec.informatik.tu-darmstadt.de](mailto:thorsten.clausius@sec.informatik.tu-darmstadt.de)

**Bernhard Esslinger**  
- University Siegen  
Faculty of Economics  
- Deutsche Bank AG  
Head of Information Security  
[bernhard.esslinger@db.com](mailto:bernhard.esslinger@db.com)  
[besslinger@web.de](mailto:besslinger@web.de)

**Jörg Cornelius Schneider**  
Deutsche Bank AG  
[joerg-cornelius.schneider@db.com](mailto:joerg-cornelius.schneider@db.com)  
[js@joergschneider.com](mailto:js@joergschneider.com)

[www.cryptool.de](http://www.cryptool.de)  
[www.cryptool.org](http://www.cryptool.org)  
[www.cryptool.com](http://www.cryptool.com)

**Mailing list: [cryptool-list@sec.informatik.tu-darmstadt.de](mailto:cryptool-list@sec.informatik.tu-darmstadt.de)**