

CrypTool

Ein freies Programmpaket

- **zur Sensibilisierung für IT-Sicherheit**
- **zum Erlernen und Erfahren von Kryptographie**
- **zur Demonstration von Algorithmen und Analyse-Verfahren der Kryptographie**

www.cryptool.de
www.cryptool.com
www.cryptool.org

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



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Inhalt

Einführung

1. Was ist CrypTool?
2. Warum CrypTool?
3. Zielgruppe

Programmüberblick

1. Funktionsumfang
2. Inhalt des Programmpakets
3. Neu in Release 1.3.xx

Anwendungsbeispiele

1. Hybridverschlüsselung visualisiert
2. Elektronische Signatur visualisiert
3. Angriff auf RSA-Verschlüsselung mit zu kurzem RSA-Modul
4. Analyse der Verschlüsselung im PSION 5 PDA
5. Demonstration schwacher DES-Schlüssel
6. Auffinden von Schlüsselmaterial (Stichwort: NSAKEY)
7. Angriff auf digitale Signatur

Kontaktadressen

Einführung

1. Was ist CrypTool?

- Freeware-Programm mit graphischer Oberfläche
- kryptographische Verfahren anwenden und analysieren
- sehr umfangreiche Online-Hilfe, verstehbar ohne tiefes Krypto-Wissen
- enthält fast alle State-of-the-art-Kryptofunktionen
- „spielerischer“ Einstieg in moderne und klassische Kryptographie
- kein „Hackertool“

2. Warum CrypTool?

- Ursprung im Deutsche Bank End-User Awareness-Programm
- Entwickelt in Kooperation mit Hochschulen
- Verbesserung der Lehre an Hochschulen und der betrieblichen Ausbildung

3. Zielgruppe

- Kernzielgruppe: Studierende der Informatik, Wirtschaftsinformatik, Mathematik
- Aber auch: Computernutzer und Anwendungsentwickler
- Voraussetzung: Abitur-Mathematik oder Programmierkenntnisse

Programmüberblick

1. Funktionsumfang

Kryptographie

Verschlüsselungsklassiker

- Caesar
- Vigenère
- Hill
- Monoalphabetische Substitution
- Homophone Substitution
- Playfair
- Permutation
- Addition
- XOR
- Vernam

Zum besseren Nachvollziehen von Literaturbeispielen ist

- Alphabet wählbar
- Behandlung von Leerzeichen etc. einstellbar

Kryptoanalyse

Angriffe auf klassische Verfahren

- ciphertext only
 - Caesar
 - Vigenère
 - Addition
 - XOR
- known plaintext
 - Hill
 - Playfair
- manuell
 - Monoalphabetische Substitution

Unterstützende Analyseverfahren

- Entropie, Gleitende Häufigkeit
- Histogramm, N-Gramm-Analyse
- Autokorrelation
- ZIP-Kompressionstest

Programmüberblick

1. Funktionsumfang

Kryptographie

Moderne symmetrische Verschlüsselung

- IDEA, RC2, RC4, DES, 3DES
- AES-Kandidaten der letzten Auswahlrunden
- AES (=Rijndael)

Asymmetrische Verschlüsselung

- RSA mit X.509-Zertifikaten
- RSA-Demonstration
 - zum Nachvollziehen von Literaturbeispielen
 - Alphabet und Blocklänge einstellbar

Hybridverschlüsselung (RSA + AES)

- visualisiert als interaktives Datenflussdiagramm

Kryptoanalyse

Brute-force Angriff auf symmetrische Algorithmen

- für alle Algorithmen
- Annahme: Entropie des Plaintextes klein

Angriff auf RSA-Verschlüsselung

- Faktorisierung des RSA-Moduls
- praktikabel bis ca. 250 bit bzw. 75 Dezimalstellen

Angriff auf Hybridverschlüsselung

- Angriff auf RSA oder
- Angriff auf AES

Programmüberblick

1. Funktionsumfang

Kryptographie

Digitale Signatur

- RSA mit X.509-Zertifikaten
 - Signatur zusätzlich visualisiert als interaktives Datenflussdiagramm
- DSA mit X.509-Zertifikaten
- Elliptic Curve DSA, Nyberg-Rueppel

Hashfunktionen

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

Zufallsgeneratoren

- Secude
- X^2 modulo N
- Lineare Kongruenz Generator (LCG)
- Inverse Kongruenz Generator (ICG)

Kryptoanalyse

Angriff auf RSA-Signatur

- Faktorisierung des RSA-Moduls
 - praktikabel bis ca. 250 bit bzw. 75 Dezimalstellen

Angriff auf Hashfunktion/digitale Signatur

- Generieren von Kollisionen für ASCII-Texte

Analyse von Zufallsdaten

- FIPS-PUB-140-1 Test-Batterie
- Periode, Vitany, Entropie
- Gleitende Häufigkeit, Histogramm
- N-Gramm-Analyse, Autokorrelation
- ZIP-Kompressionstest

Programmüberblick

2. Inhalt des Programmpakets

CrypTool-Programm

- alle Funktionen integriert in *einem* Programm mit einheitlicher graphischer Oberfläche
- läuft unter Win32 und unter Linux mit WINE-Emulator
- Kryptographie von Secude-Bibliothek (www.secude.com)
- Langzahlarithmetik: Miracl-Bibliothek (<http://indigo.ie/~mscott/>)

AES-Tool

- Standalone-Programm zur AES-Verschlüsselung (selbst extrahierend)

Umfangreiche Online-Hilfe (Winhelp)

- kontextsensitive Hilfe für *alle* Programmfunktionen und zu jedem Menüpunkt
- ausführliche Benutzungs-Szenarien für viele Programmfunktionen

Skript (PDF) mit Hintergrundinformationen zu

- Verschlüsselungsverfahren • Primzahlen • Digitale Signatur
- Elliptische Kurven • Public Key-Zertifizierung • Elementare Zahlentheorie

Kurzgeschichte „Dialog der Schwestern“ von Dr. C. Elsner

komplett zweisprachig
Deutsch
Englisch

Programmüberblick

3. Neu in Release 1.3.xx

Wichtigste Neuerungen (Details: siehe ReadMe-de.txt):

Release 1.3.00 veröffentlicht Januar 2002

- komplett zweisprachig in Deutsch und Englisch
- Konsistenz und Verständlichkeit der Dialoge verbessert
- Dateigrößenbeschränkung unter Win9x aufgehoben
- Homophone und Permutationsverschlüsselung
- Zufallsgeneratoren, Analyse von Zufallsdaten (FIPS, Perioden, N-Gramm)
- AES-Tool: Erzeugung selbstentschlüsselnder Dateien (AES)
- Demo: Zahlentheorie und RSA-Kryptosystem (weiter verbessert in 1.3.02)
- PKCS#12-Export/Import für PSEs

Release 1.3.03 veröffentlicht August 2002

- Visualisierung der Hybrid-Ver- und Entschlüsselung
- Visualisierung der Erzeugung und Verifikation von Signaturen
- Hashwerte großer Dateien berechnen, ohne sie zu laden
- Visualisierung der Sensibilität von Hashfunktionen bzgl. Änderungen der Daten
- Kurzgeschichte „Dialog der Schwestern“ (Dr. C. Elsner) beigelegt

Programmüberblick

3. Neu in Release 1.3.xx

Release 1.3.04 veröffentlicht Juni 2003

- Visualisierung des Diffie-Hellman Schlüsseltausches
- Angriff auf digitale Signatur über Suche nach Hash-Kollisionen (Geburtstagsparadoxon)
- Verbesserung der Brute-Force-Attacke auf symmetrische Algorithmen
- Skript: aktualisiert (Primzahlen, Faktorisierung) und erweitert (Hash, ECC, CrypTool-Menübaum)
- Viele Detailverbesserungen (insbesondere der Online-Hilfe) und Bug-Fixes

Anwendungsbeispiele

1. Hybridverschlüsselung visualisiert

Hybridverschlüsselung

- kombiniert die Vorteile symmetrischer und asymmetrischer Verschlüsselung
 - Geschwindigkeit
 - einfache und skalierbare Schlüsselverteilung
- weit verbreiteter praktischer Einsatz
 - E-Mail (S/MIME, PGP) und Dateiverschlüsselung
 - SSL (https)

Visualisiert durch ein interaktives Datenflussdiagramm

- Zusammenhänge verstehen durch spielerisches Lernen

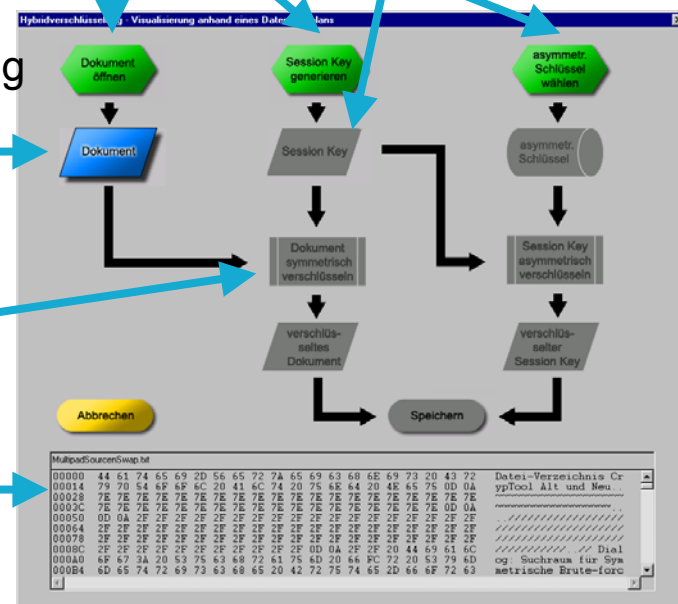
grau = Voraussetzung fehlt

vorbereitende Schritte

Datenelement

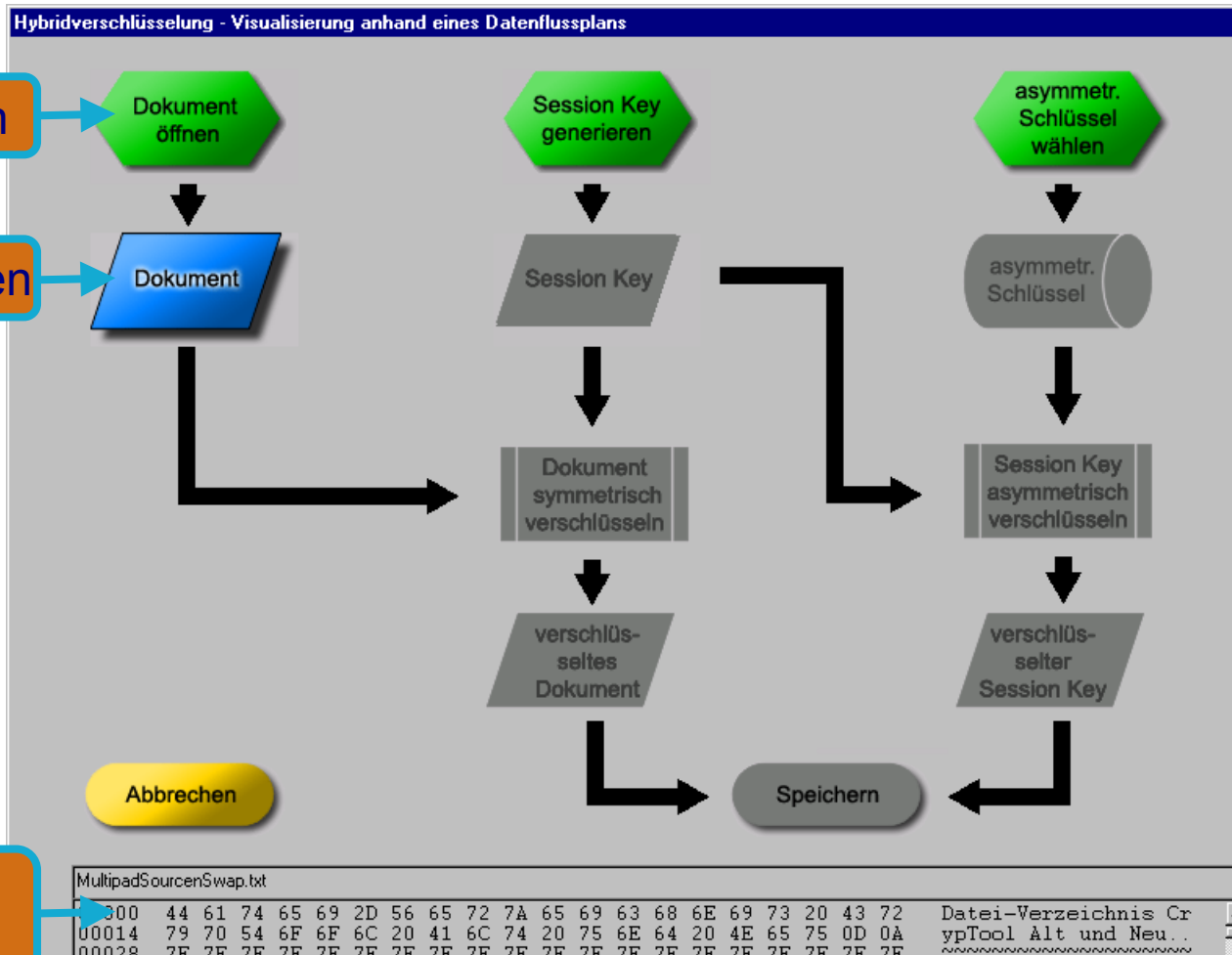
Operation

Anzeigebereich



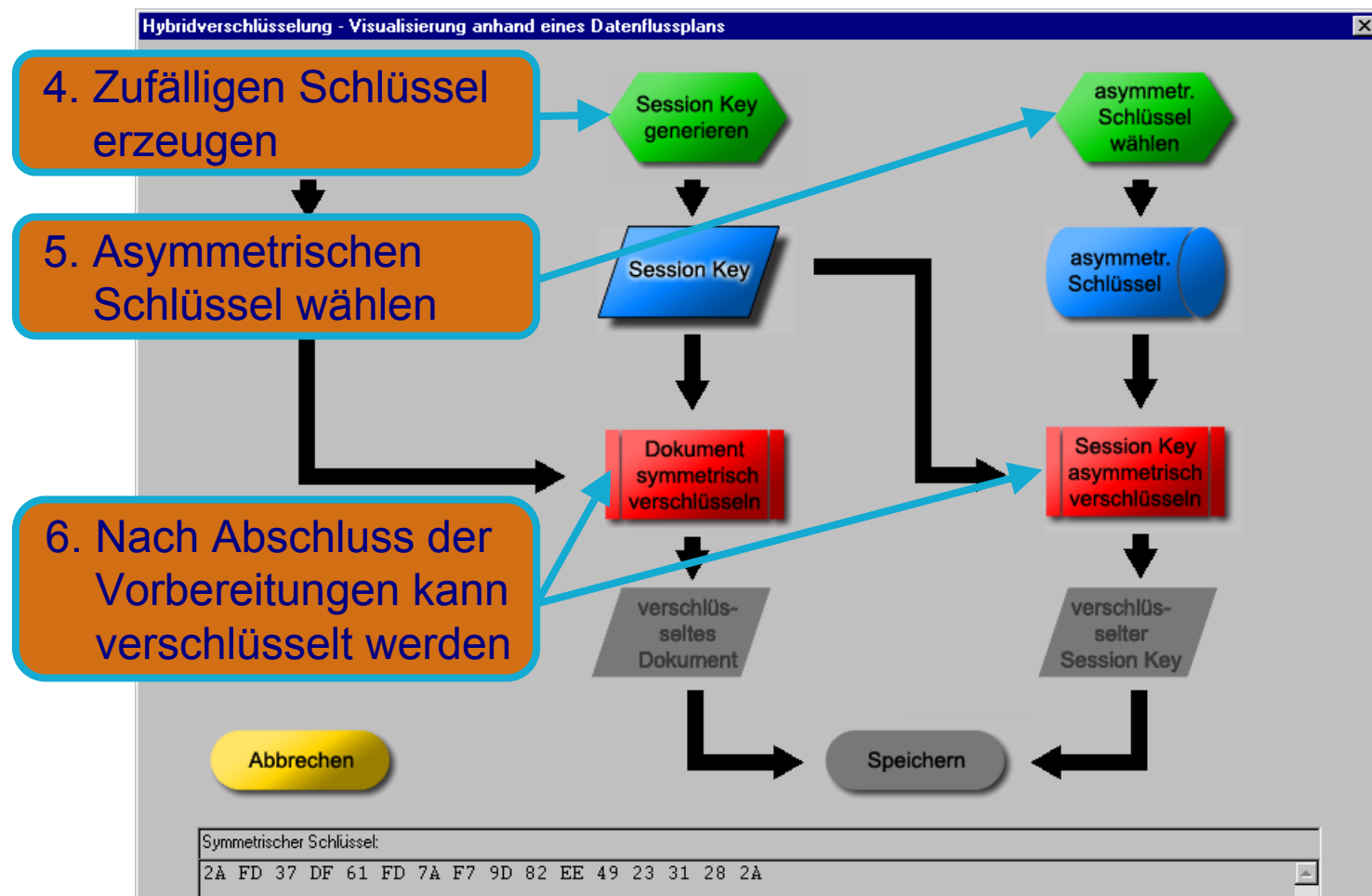
Anwendungsbeispiele

1. Hybridverschlüsselung visualisiert: Vorbereitung



Anwendungsbeispiele

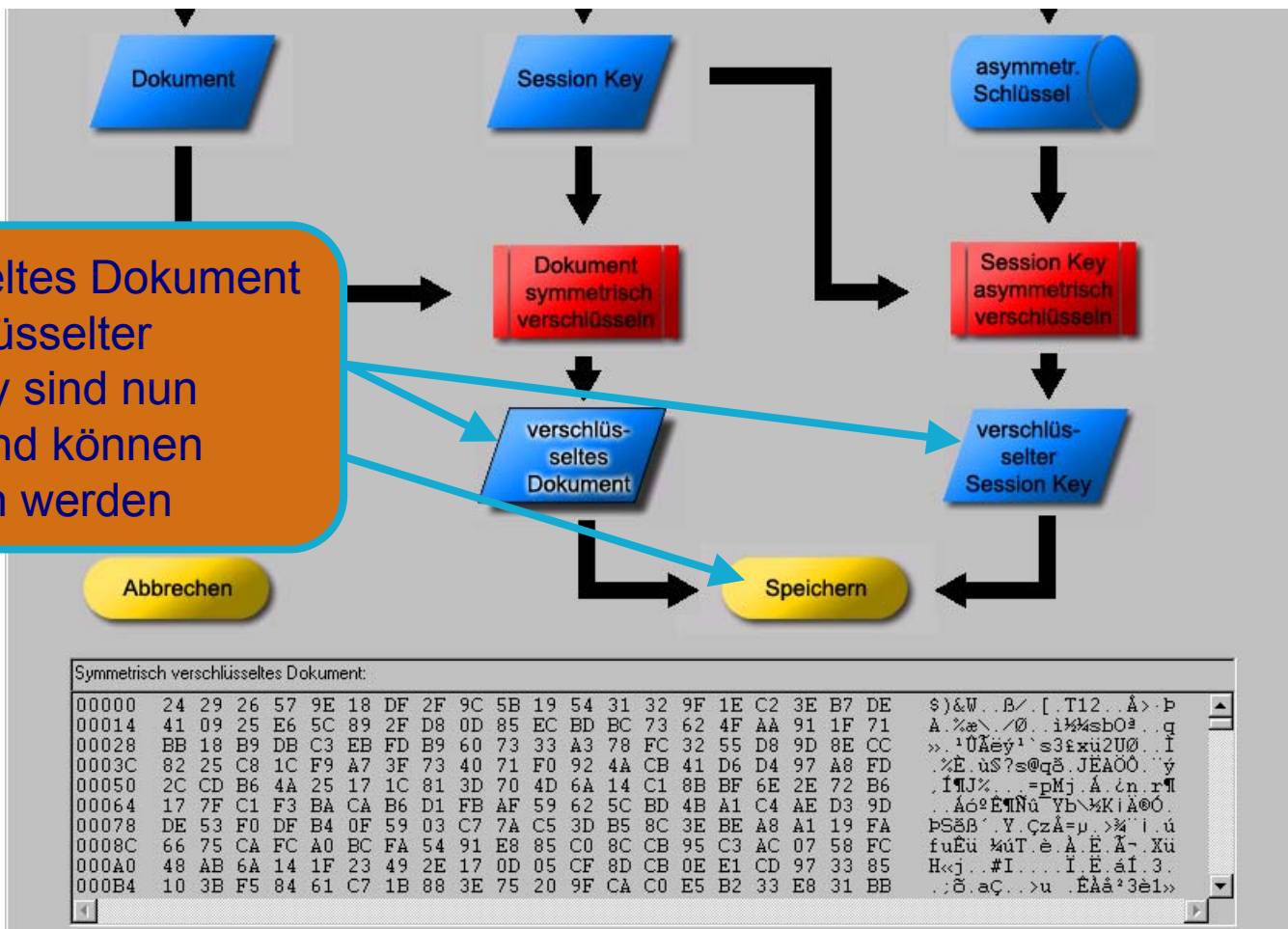
1. Hybridverschlüsselung visualisiert: Kryptographie



Anwendungsbeispiele

1. Hybridverschlüsselung visualisiert: Ergebnis

7. Verschlüsseltes Dokument und verschlüsselter Session Key sind nun verfügbar und können ausgegeben werden



Anwendungsbeispiele

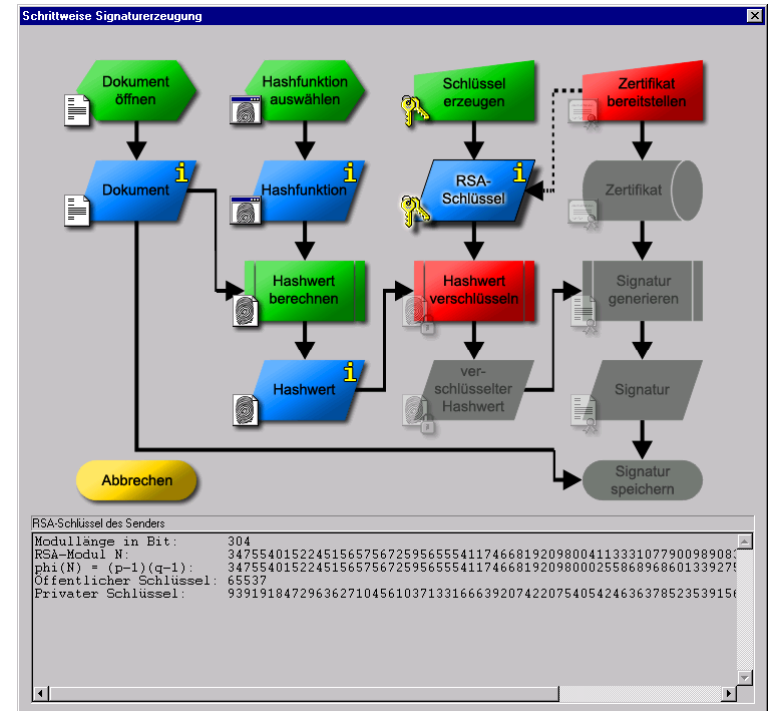
2. Elektronische Signatur visualisiert

Elektronische Signatur

- Wird immer wichtiger durch
 - Gleichstellung mit manueller Unterschrift (Signaturgesetz)
 - Zunehmenden Einsatz in der Wirtschaft, durch den Staat und privat
- Wer weiß, wie sie funktioniert?

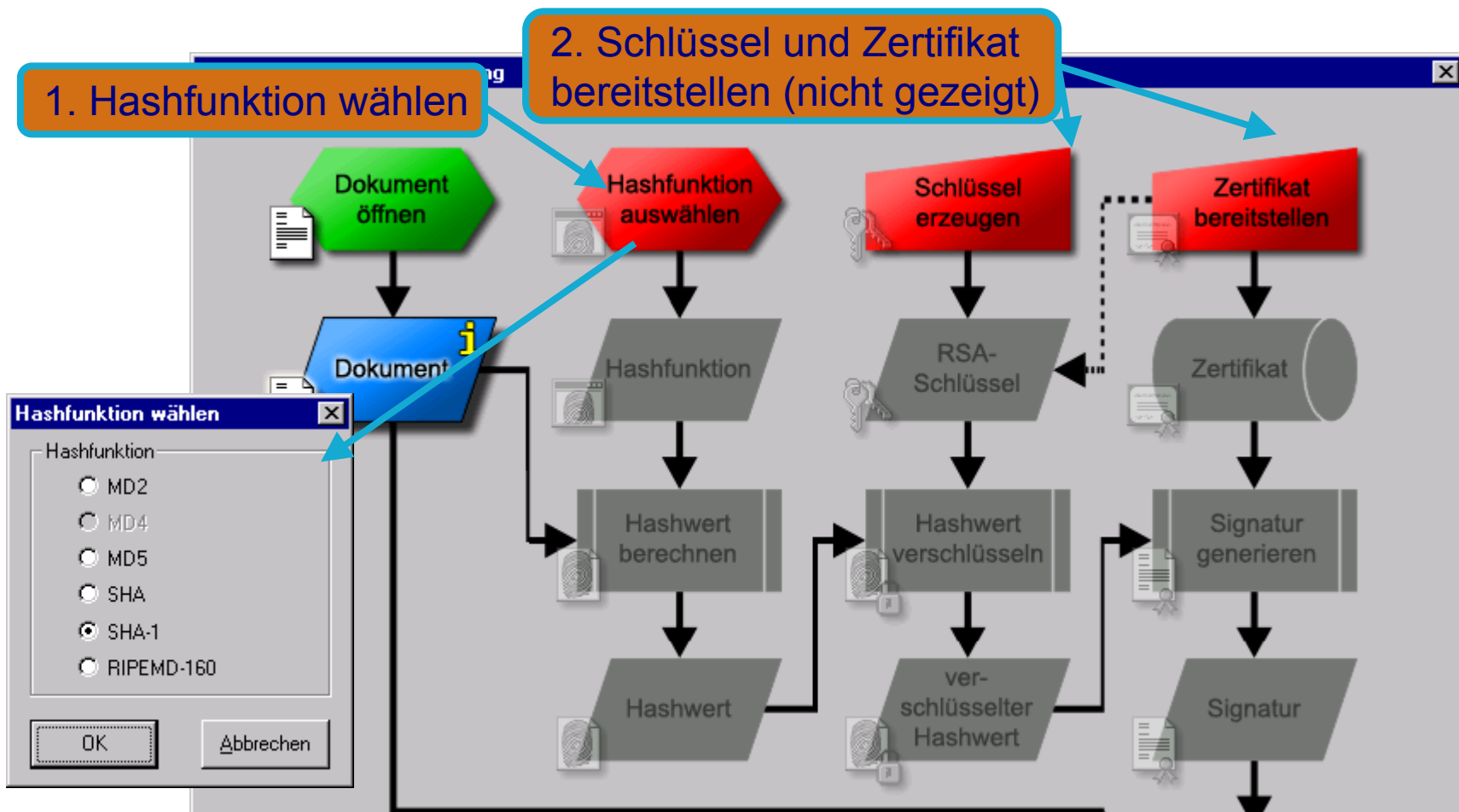
Visualisierung in CrypTool

- Interaktives Datenflussdiagramm
- Ähnlich wie die Visualisierung der Hybridverschlüsselung



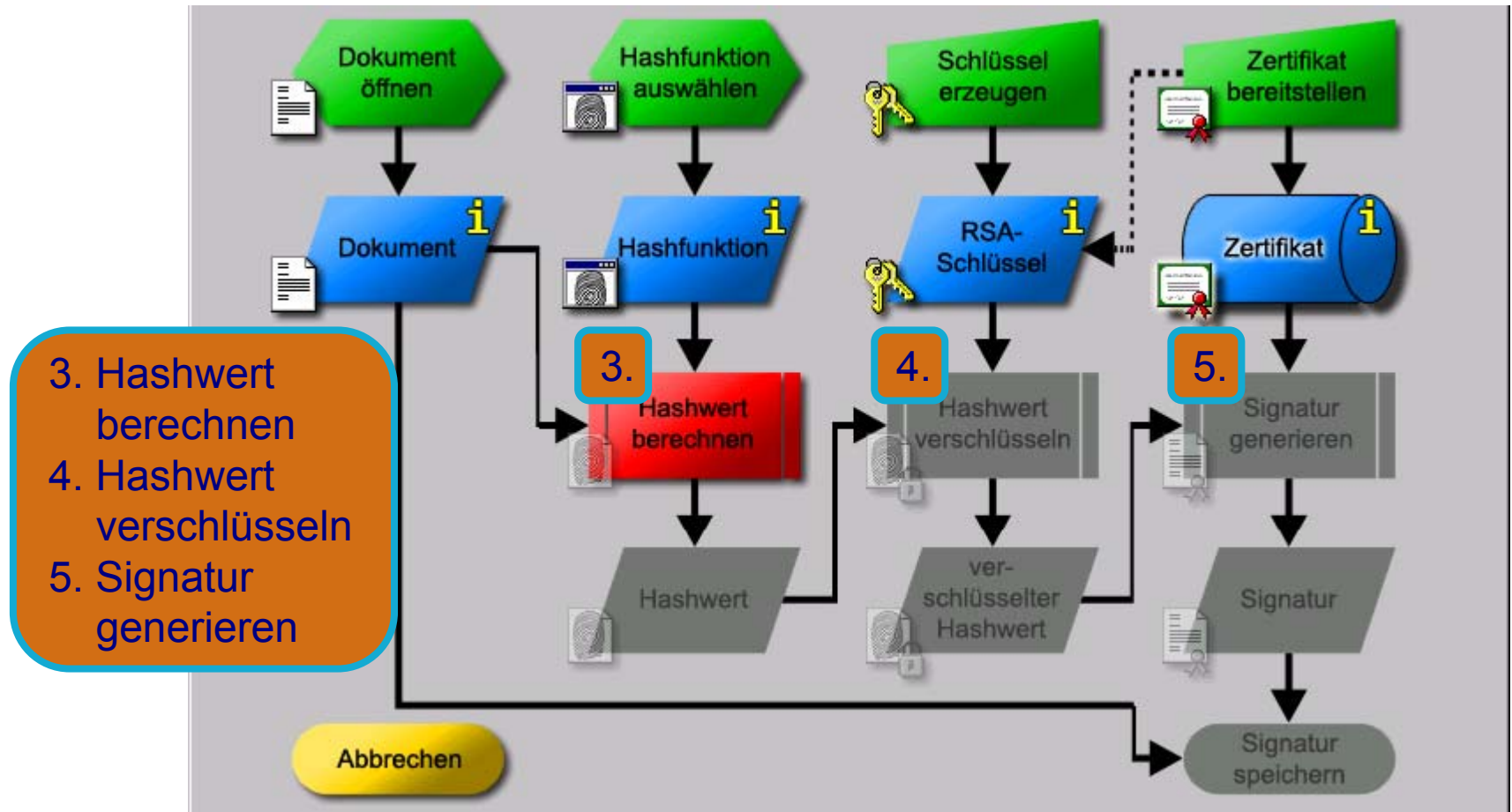
Anwendungsbeispiele

2. Elektronische Signatur visualisiert: Vorbereitung



Anwendungsbeispiele

2. Elektronische Signatur visualisiert: Kryptographie

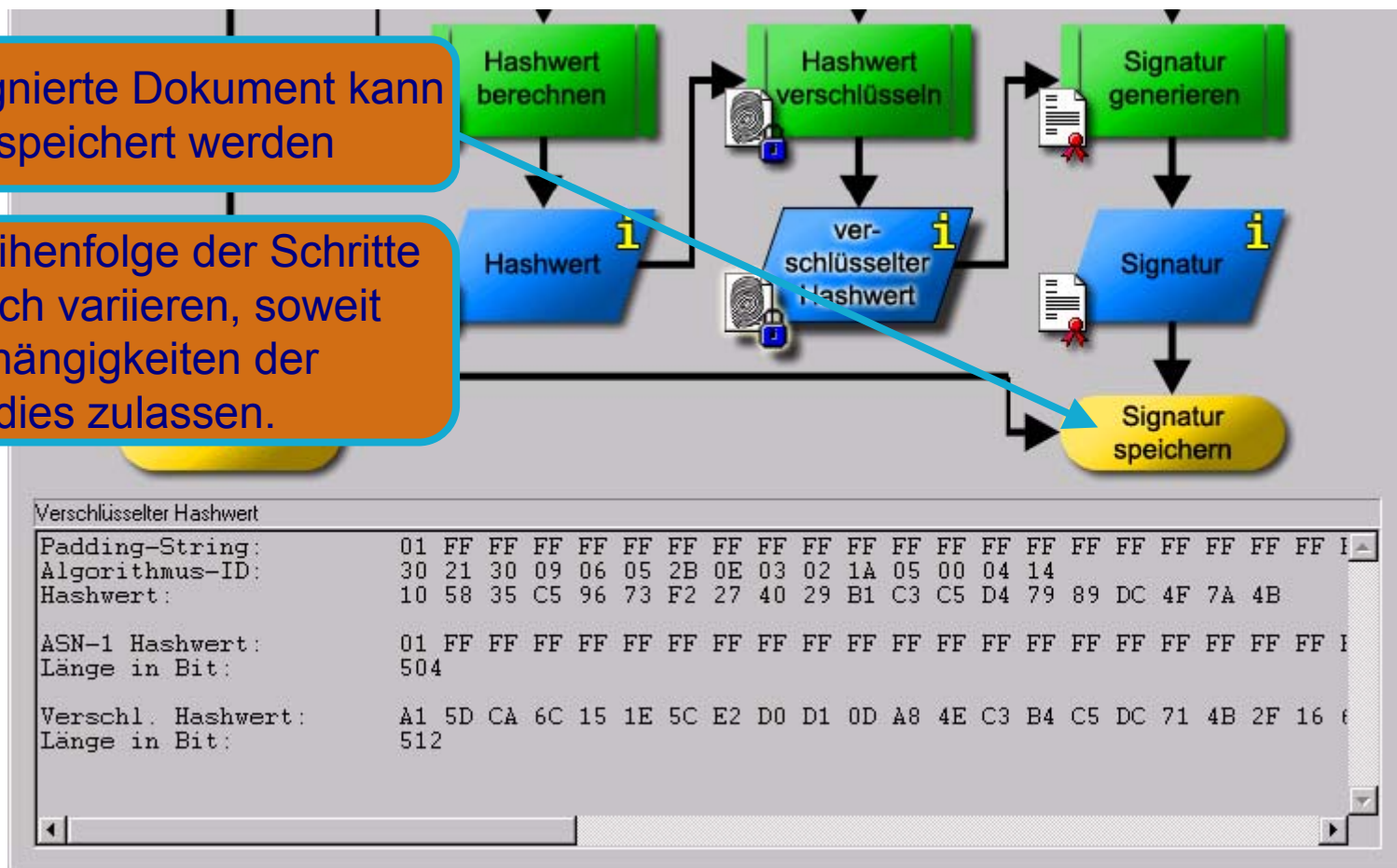


Anwendungsbeispiele

2. Elektronische Signatur visualisiert: Ergebnis

Das signierte Dokument kann nun gespeichert werden

Die Reihenfolge der Schritte lässt sich variieren, soweit die Abhängigkeiten der Daten dies zulassen.



Anwendungsbeispiele

3. Angriff auf zu kurzen RSA-Modul

Aufgabe aus Song Y. Yan, Number Theory for Computing, Springer, 2000

■ Öffentlicher Schlüssel

- RSA-Modul $N = 63978486879527143858831415041$ (95bit, 29 Dezimastellen)
- Öffentlicher Exponent $e = 17579$

■ Verschlüsselter Text (Blocklänge = 8):

- $C_1 = 45411667895024938209259253423$,
 $C_2 = 16597091621432020076311552201$,
 $C_3 = 46468979279750354732637631044$,
 $C_4 = 32870167545903741339819671379$

■ Der Text soll entschlüsselt werden

Lösung mit CrypTool (ausführlich in Szenarien der Online-Hilfe)

- Öffentliche Parameter in RSA-Kryptosystem (Menü Einzelverfahren) eintragen
- Funktion „RSA-Modul faktorisieren“ liefert Primfaktoren $pq = N$
- Daraus wird der geheime Schlüssel $d = e^{-1} \bmod (p-1)(q-1)$ abgeleitet
- Entschlüsseln des Textes mit Hilfe von d : $M_i = C_i^d \bmod N$

Angriff mit CrypTool ist für RSA-Module bis ca. 250 bit praktikabel

Anwendungsbeispiele:

3. Kurzer RSA-Modul: öffentliche Parameter eingeben

Das RSA-Kryptosystem

☐ RSA mit privatem und öffentlichem Schlüssel -- oder nur mit öffentlichem Schlüssel

☐ Wählen Sie 2 Primzahlen p und q . Die Zahl $N = pq$ ist der öffentliche RSA-Modul und $\phi(N) = (p-1)(q-1)$ ist die Eulersche Zahl. Der öffentliche Schlüssel e ist teilerfremd zu $\phi(N)$. Daraus wird der geheime Schlüssel $d = e^{-1} \pmod{\phi(N)}$ berechnet.

☒ Zur Verschlüsselung von Daten oder zur Verifikation einer Signatur genügt es, dass Sie die veröffentlichten RSA-Parameter angeben: den RSA-Modul N und den öffentlichen Schlüssel e .

Faktorisierungsangriff

Sie können mit Hilfe der Faktorisierung versuchen, den öffentlichen RSA-Modul N in seine Primfaktoren p und q zu faktorisieren.

RSA-Parameter

RSA-Modul N (öffentlich)

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

öffentlicher Schlüssel e

geheimer Schlüssel d

☐ RSA-Verschlüsselung mit e / Entschlüsselung mit d

2. Faktorisieren

1. RSA-Parameter N und e eingeben



Anwendungsbeispiele:

3. Kurzer RSA-Modul: RSA-Modul faktorisieren

Faktorisieren einer Zahl

Algorithmen zur Faktorisierung

- ☒ Brute-force Methode
- ☒ Algorithmus nach Brent
- ☒ Pollard Methode
- ☒ Williams Methode
- ☒ Algorithmus nach Lenstra
- ☒ Quadratische Sieb Methode

Eingabe

Geben Sie die zu faktorisierende Zahl ein:

63978486879527143858831415041

Faktorisierungsergebnis

Die Faktorisierung wird in dem Format $\langle z1^{a1} * z2^{a2} * \dots * zn^{an} \rangle$ dargestellt. Zusammengesetzte Zahlen sind rot markiert.

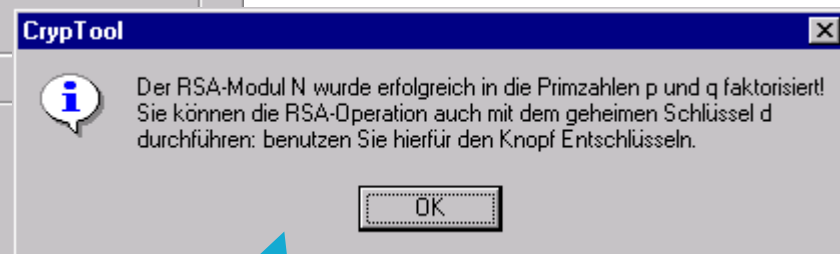
Letzte Faktorisierung durch: Pollard

Benötigte Zeit: 01,191 Sekunden.

Produktdarstellung der Faktorisierung:

145295143558111 * 440334654777631

Schließen



3. Faktorisierung
ergibt p und q

Anwendungsbeispiele:

3. Kurzer RSA-Modul: geheimen Schlüssel d bestimmen

Das RSA-Kryptosystem

☒ RSA mit privatem und öffentlichem Schlüssel -- oder nur mit öffentlichem Schlüssel

☒ Wählen Sie 2 Primzahlen p und q. Die Zahl $N = pq$ ist der öffentliche RSA-Modul und $\phi(N) = (p-1)(q-1)$ ist die Eulersche Zahl. Der öffentliche Schlüssel e ist teilerfremd zu $\phi(N)$. Daraus wird der geheime Schlüssel $d = e^{-1} \pmod{\phi(N)}$ berechnet.

☐ Zur Verschlüsselung von Daten oder zur Verifikation einer Signatur genügt es, dass Sie die veröffentlichten RSA-Parameter angeben: den RSA-Modul N und den öffentlichen Schlüssel e.

Primzahleingabe

Primzahl p: 145295143558111

Primzahl q: 440334654777631

RSA-Parameter

RSA-Modul N: 63978486879527143858831415041 (öffentlich)

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

öffentlicher Schlüssel e: 17579

geheimer Schlüssel d: 10663687727232084624328285019

RSA-Verschlüsselung mit e / Entschlüsselung mit d

Eingabe als ☐ Text ☒ Zahlen

Eingabe der Nachricht als Zahlen im Format: Zahl(1) # Zahl(2) # ... # Zahl(n) (Zahlen zur Basis 10).

4. p und q wurden automatisch eingetragen und geheimer Schlüssel d berechnet

5. Optionen einstellen

Anwendungsbeispiele:

3. Kurzer RSA-Modul: Optionen einstellen

Optionen für die RSA-Verschlüsselung

Textoptionen

☐ Alle 256 Zeichen Anzahl Zeichen: 27

☒ Alphabet vorgeben:

Verfahren bei der Kodierung der Nachricht in Zahlen

Methode: ☐ b-adisch ☒ Basissystem

Blocklänge

Die Anzahl der Zeichen, die pro RSA-Operation verschlüsselt werden.
Die maximale Anzahl ist abhängig von der Bitlänge des RSA-Moduls N, der Anzahl der Zeichen im Alphabet und der Kodierungsmethode der Nachricht.

Blocklänge in Zeichen: (maximale Blocklänge 14 Zeichen)

Zahlensystem

Die Zahlen der RSA-Ver-/Entschlüsselung werden in dem folgenden Zahlensystem dargestellt.

☒ Dezimal ☐ Binär ☐ Oktal ☐ Hexadezimal

OK Abbrechen

6. Alphabet wählen

7. Kodierung wählen

8. Blocklänge wählen

Anwendungsbeispiele:

3. Kurzer RSA-Modul: Text entschlüsseln

RSA-Parameter

RSA-Modul N	63978486879527143858831415041	(öffentlich)
$\phi(N) = (p-1)(q-1)$	63978486879526558229033079300	(geheim)
öffentlicher Schlüssel e	17579	
geheimer Schlüssel d	10663687727232084624328285019	

Parameter aktualisieren

RSA-Verschlüsselung mit e / Entschlüsselung mit d

Eingabe als ☐ Text ☒ Zahlen [Optionen für Alphabet und Zahlensystem...](#)

Chiffretext in Zahlendarstellung zur Basis 10:

7091621432020076311552201 # 46468979279750354732637631044 # 3287016754590374133981967129

Entschlüsselung in den Klartext $m[i] = c[i]^d \pmod{N}$

0000000000001401202118011200 # 00000000000001421130205181900 # 0000000000000011805001301

Ausgabertext aus der Entschlüsselung (in Blöcken der Länge 8; das Symbol '#' dient nur als Trennzeichen).

NATURAL # NUMBERS # ARE MADE # BY GOD

Klartext

NATURAL NUMBERS ARE MADE BY GOD

Verschlüsseln Entschlüsseln Schließen

9. Ciphertext eingeben

10. Entschlüsseln

Anwendungsbeispiele

4. Analyse der Verschlüsselung im PSION 5 PDA

Angriff auf die Verschlüsselungsoption der Textverarbeitungsapplikation im PSION 5 PDA

Gegeben: eine auf dem PSION verschlüsselte Datei

Voraussetzung

- verschlüsselter deutscher oder englischer Text
- je nach Verfahren und Schlüssellänge 100 Byte bis einige kB Text

Vorgehen

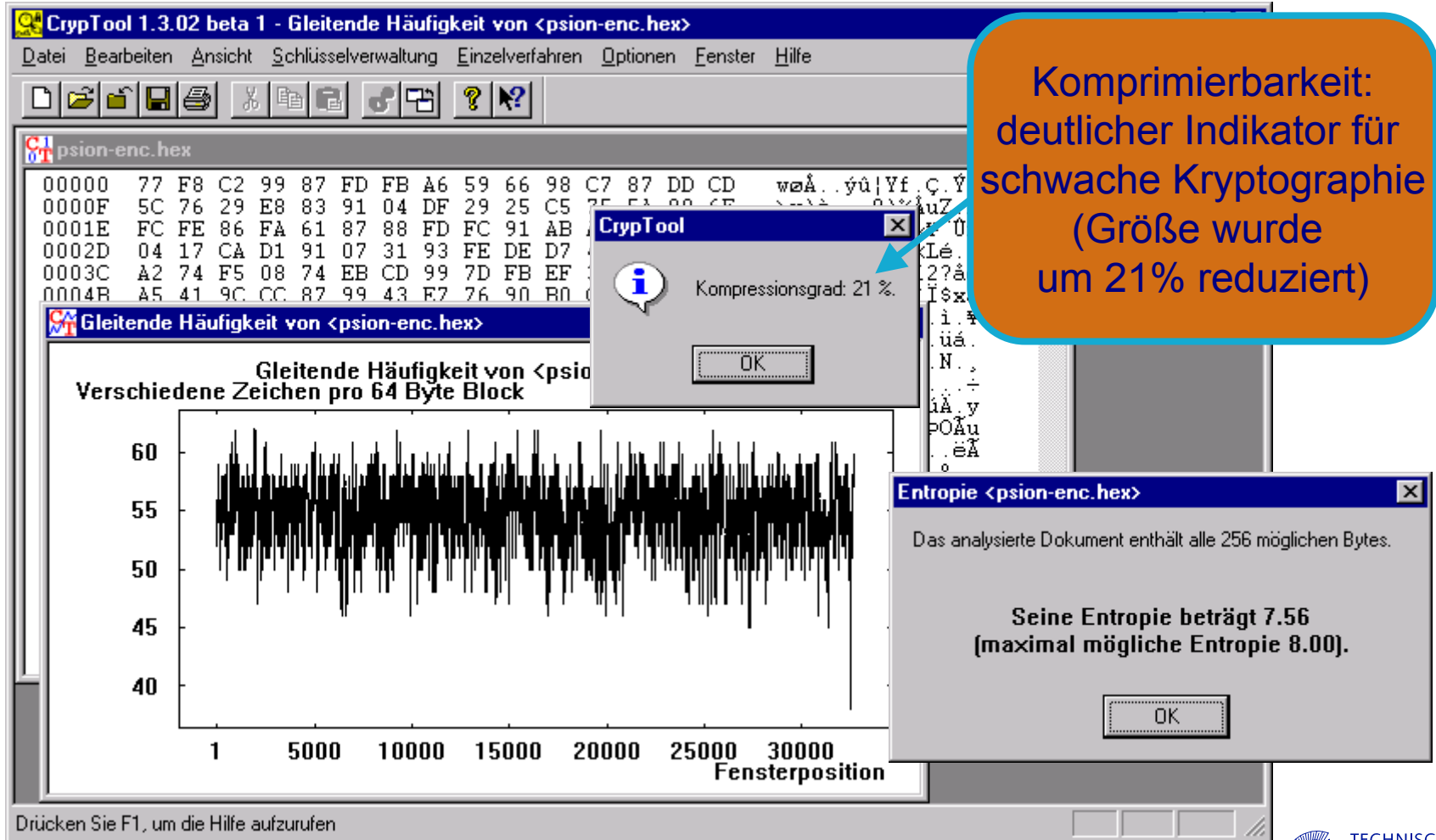
- Voranalyse
 - Entropie
 - gleitende Häufigkeit
 - Kompressionstest
- Autokorrelation
- automatische Analyse klassischer Verfahren durchprobieren

} ⇒ wahrscheinlich klassische Verschlüsselung



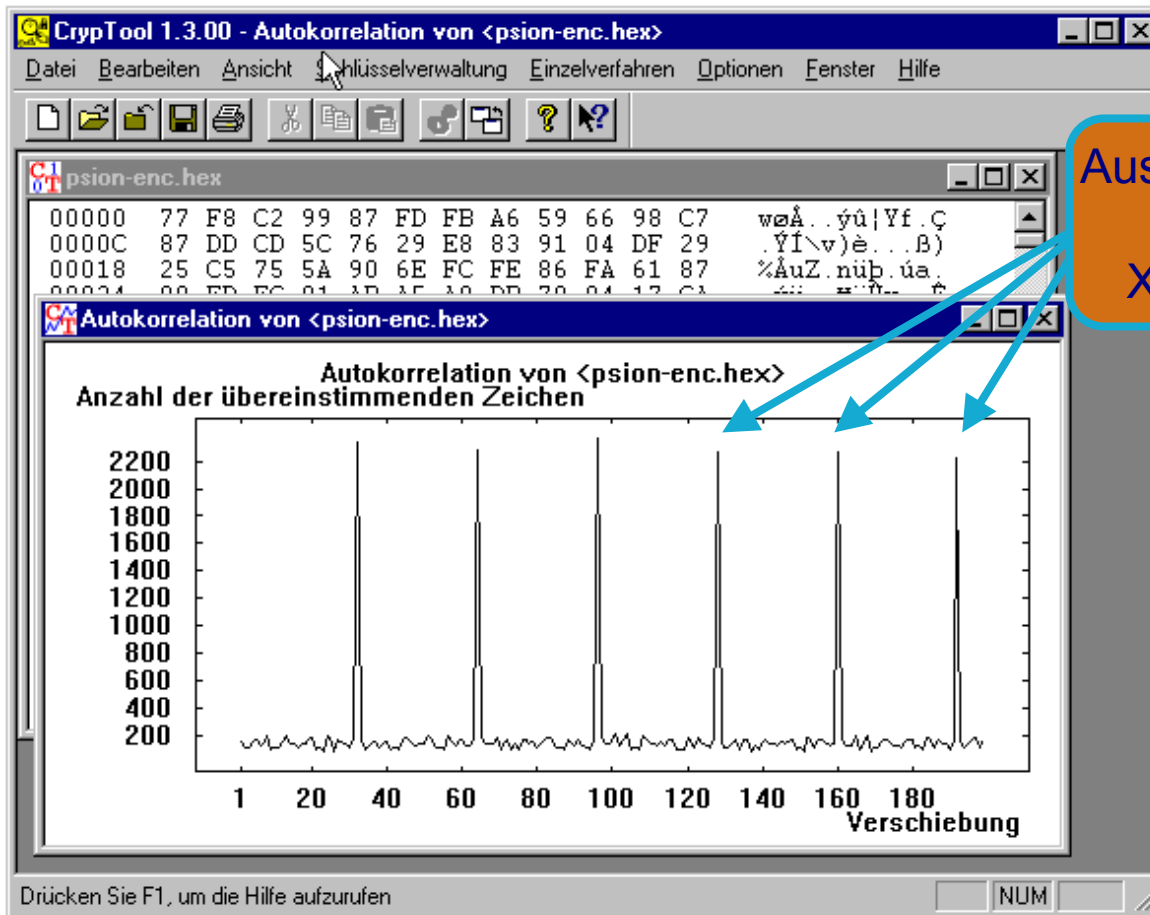
Anwendungsbeispiele

4. PSION-PDA: Entropie bestimmen, Kompressionstest



Anwendungsbeispiele

4. PSION-PDA: Autokorrelation bestimmen



Ausgeprägtes Kamm-Muster:
typisch für Vigenère,
XOR und binäre Addition

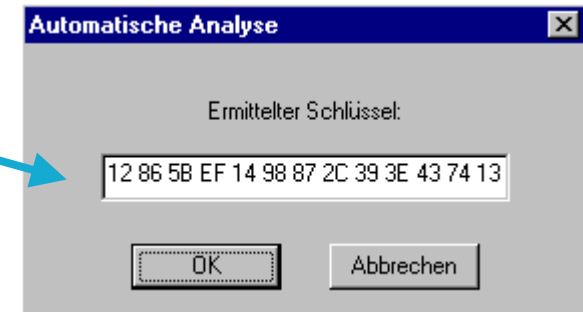
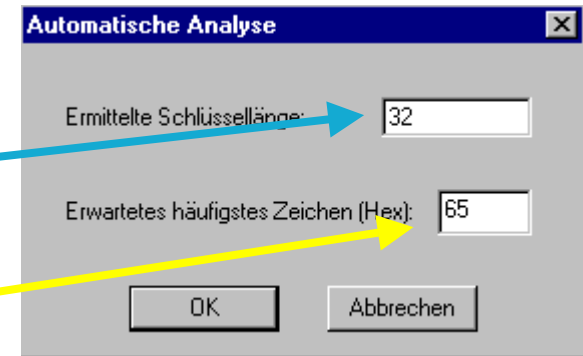
Anwendungsbeispiele

4. PSION-PDA: Automatische Analyse

Automatische Analyse XOR: kein Erfolg

Automatische Analyse binäre Addition:

- CrypTool ermittelt die Schlüssellänge mittels Autokorrelation: 32 Byte
- Das erwartete häufigste Zeichen kann der Benutzer wählen: „e“ = 0x65 (ASCII-Code)
- Analyse ermittelt den (unter der Verteilungsannahme) wahrscheinlichsten Schlüssel
- Ergebnis: gut, aber nicht perfekt

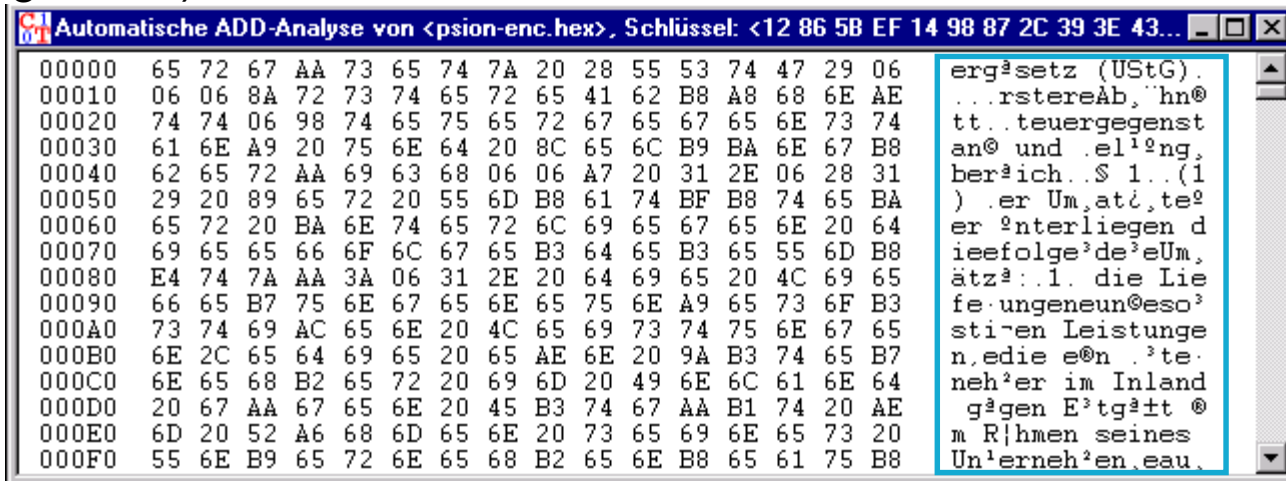


Anwendungsbeispiele

4. PSION-PDA: Ergebnis der automatischen Analyse

Ergebnis der automatischen Analyse mit Annahme „binäre Addition“:

- Ergebnis gut, aber nicht perfekt: 24 von 32 Schlüsselbytes richtig.
- Die Schlüssellänge wurde korrekt bestimmt.
- Das eingegebene Passwort war aber nicht 32 Byte lang.
⇒ PSION Word leitet aus dem Passwort den eigentlichen Schlüssel ab.
- Nacharbeiten von Hand liefert den entschlüsselten Text
(nicht abgebildet)



```
Automatische ADD-Analyse von <psion-enc.hex>, Schlüssel: <12 86 5B EF 14 98 87 2C 39 3E 43...
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.
```

Anwendungsbeispiele

4. PSION-PDA: Bestimmung des kompletten Schlüssels

Schlüssel während der automatischen Analyse in die Zwischenablage kopieren

im Hexdump der automatischen Analyse

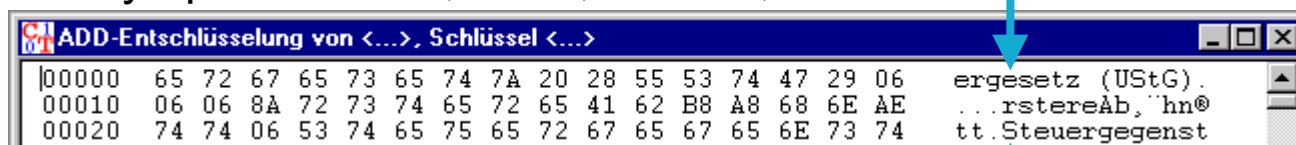
- Falsche Bytepositionen bestimmen, z.B. 0xAA an Position 3
- Korrespondierende korrekte Bytes erraten und notieren: „e“ = 0x65

im Hexdump der verschlüsselten Ausgangsdatei

- Ausgangsbyte an der ermittelten Byteposition bestimmen: 0x99
- Mit CALC.EXE korrekte Schlüsselbytes errechnen: $0x99 - 0x65 = 0x34$

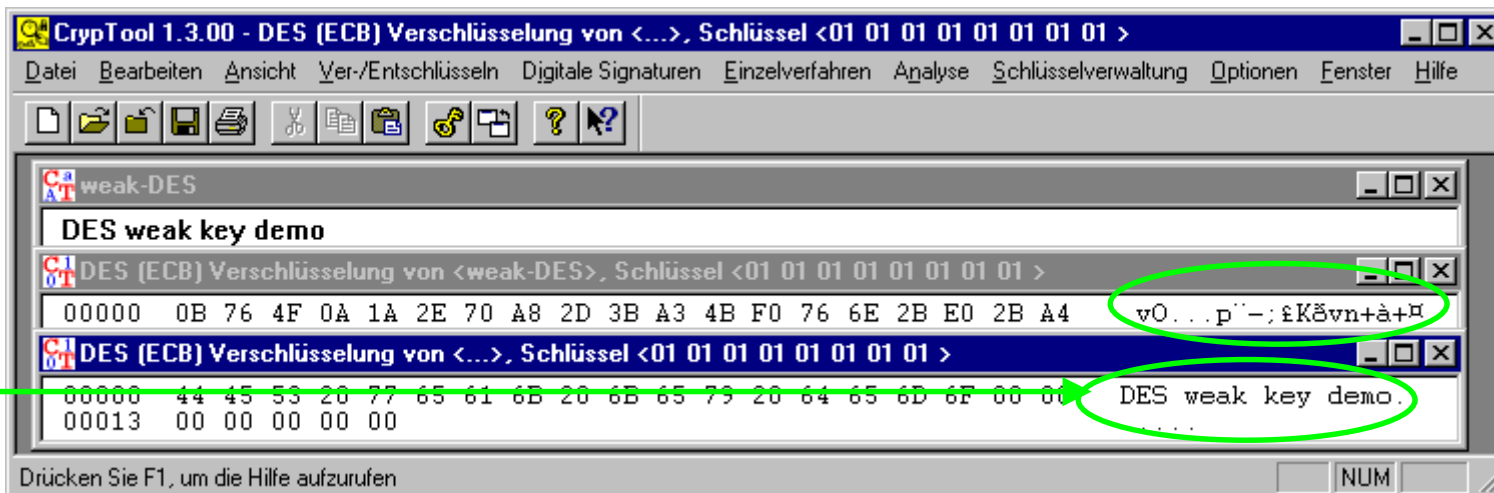
Schlüssel aus der Zwischenablage

- korrigieren
12865B341498872C393E43741396A45670235E111E907AB7C0841...
- verschlüsseltes Ausgangsdokument mittels binärer Addition entschlüsseln
- Nun sind Bytepositionen 3, 3+32, 3+2*32, ... ok



Anwendungsbeispiele

5. Schwache DES-Schlüssel



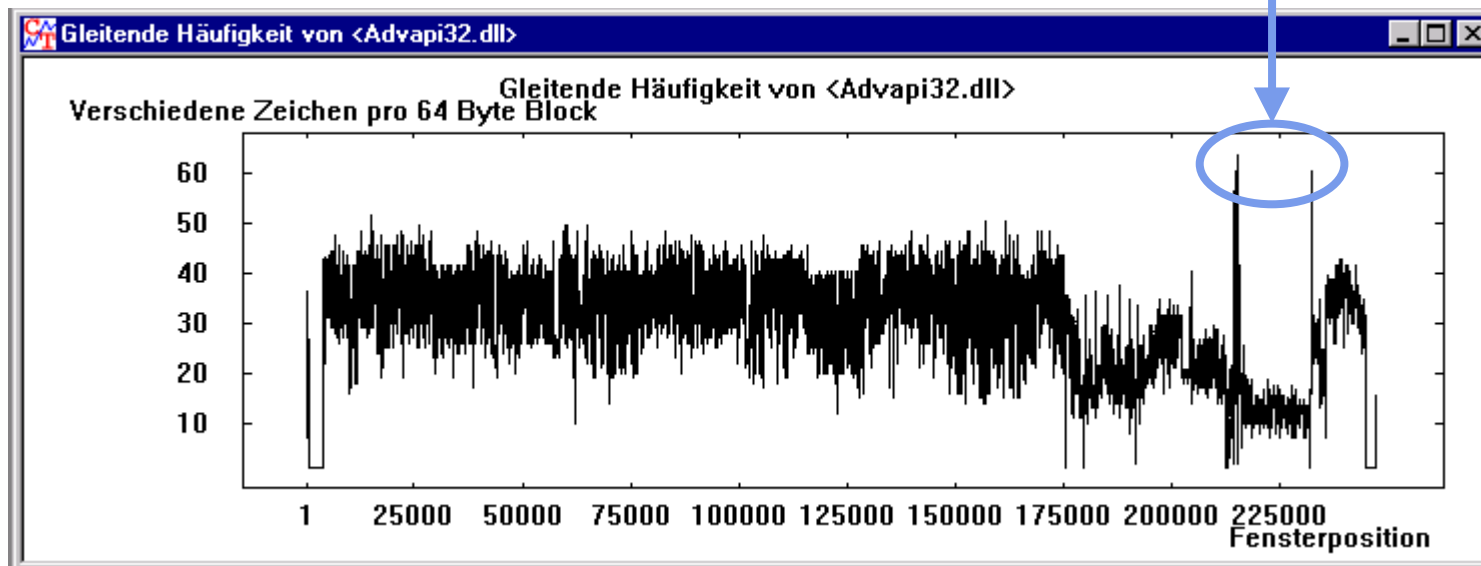
Anwendungsbeispiele

6. Auffinden von Schlüsselmateral

Die Funktion „Gleitende Häufigkeit“ eignet sich zum Auffinden von Schlüsselmateral und verschlüsselten Bereichen in Dateien.

Hintergrund:

- diese Daten sind „zufälliger“ als Text oder Programmcode
- sie sind als Peak in der „gleitenden Häufigkeit“ zu erkennen
- Beispiel: der „NSAKEY“ in advapi32.dll



Anwendungsbeispiele

7. Angriff auf digitale Signatur: Idee

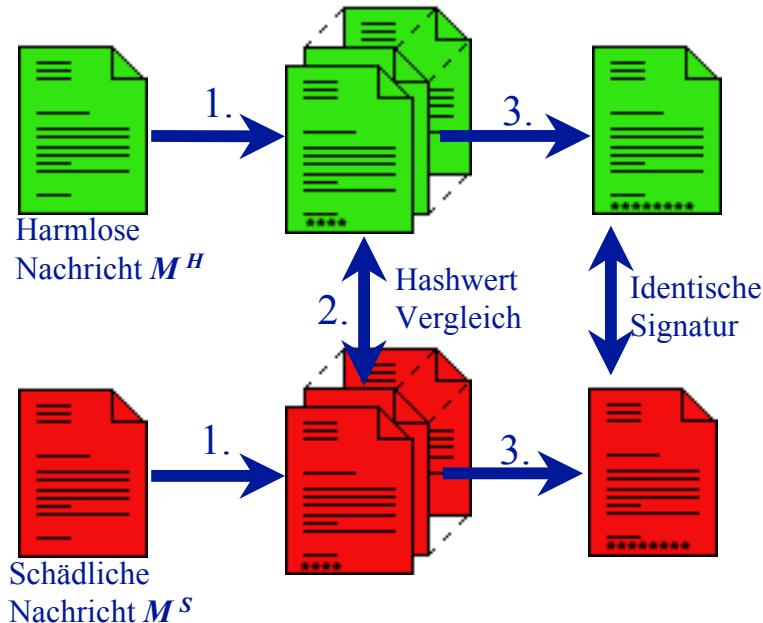
Angriff auf die digitale Signatur eines ASCII-Textes durch Suche nach Hashkollisionen

Idee:

- ASCII-Text kann mittels **nicht-druckbarer** Zeichen modifiziert werden, ohne den lesbaren Inhalt zu verändern
- Modifiziere parallel zwei Texte, bis eine Hashkollision erreicht wird
- Ausnutzung des Geburtstagsparadoxons (Geburtstagsangriff)
- Generischer Angriff auf beliebige Hashfunktion
- Angriff ist gut parallelisierbar (nicht implementiert)
- In CrypTool implementiert von Jan Blumenstein im Rahmen der Bachelor-Arbeit „*Methoden und Werkzeuge für Angriffe auf die digitale Signatur*“, 2003.

Anwendungsbeispiele

7. Angriff auf digitale Signatur: Idee (2)



- 1. Modifikation:** Ausgehend von der Nachricht M werden N verschiedene Nachrichten M_1, \dots, M_N – „inhaltlich“ gleich mit der Ausgangsnachricht – erzeugt.
- 2. Suche:** Gesucht werden *modifizierte* Nachrichten M_i^H und M_j^S mit gleichem Hashwert.
- 3. Angriff:** Die Signaturen zweier solcher Dokumente M_i^H und M_j^S sind identisch.

Für Hashwerte der Bitlänge n sagt das Geburtstagsparadoxon:

- Kollisionssuche zwischen M^H und M_1^S, \dots, M_N^S : $N \approx 2^n$
- Kollisionssuche zwischen M_1^H, \dots, M_N^H und M_1^S, \dots, M_N^S : $N \approx 2^{n/2}$

Anwendungsbeispiele

7. Angriff auf digitale Signatur: Angriff

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 Verfahren 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

Anwendungsbeispiel

7. Angriff auf digitale Signatur: Ergebnisse

Harmlose Nachricht: MD5, <4F 47 DF 1F>

Sehr geehrter Herr Einkaufsgern,
bitte bestellen Sie eine Schreibmaschine

MfG.
Peter Gutermann
AADBADCBACBACDADCB

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

Gefährliche Nachricht: MD5, <4F 47 DF 1F>

Sehr geehrter Herr Einkaufsgern,
bitte bestellen Sie für Herrn Dieter Dieb ein Porsche und eine Tankkarte.

MfG.
Peter Gutermann
AAAABBDDBBAAABBC

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

Die ersten 32 Bit des Hashwertes sind gleich.

Praktische Resultate

- 72 Bit *Teilkollisionen* (Übereinstimmung der ersten 72 Bit-Stellen der Hashwerte) konnten im Zeitraum von wenigen Tagen auf einen einzigen PC gefunden werden.
- Signaturverfahren mit Hashverfahren bis zu 128 Bit Länge sind heute gegenüber massiv parallelen Verfahren angreifbar!

Weiterentwicklung

In Arbeit

- Visualisierung des Challenge-Response-Verfahrens
- Angriff auf einseitige Authentifikation mit CR bei schwacher Verschlüsselung
- Massenmustersuche

Geplant

- Visualisierung des SSL-Protokolls
- Visualisierung von Man-in-the-Middle-Angriffen
- Darstellung eines Seitenkanal-Angriffs

Angedacht

- Visualisierung von Protokollabläufen (z.B. Kerberos)
- Visualisierung von Angriffen auf diese Protokollabläufe
- Portierung nach Linux oder Java
- Viele weitere Ideen stehen im Readme, Kapitel 6

Kontaktadressen

Prof. Dr. Claudia Eckert

TU Darmstadt

Fachbereich Informatik

Fachgebiet Sicherheit in der IT

Wilhelminenstr. 7

64283 Darmstadt

claudia.eckert@

sec.informatik.tu-darmstadt.de

Bernhard Esslinger

- Universität Siegen

Fachbereich 5 Wirtschaftswissenschaften

- Deutsche Bank AG

Leiter Information Security

bernhard.esslinger@db.com

besslinger@web.de

Thorsten Clausius

TU Darmstadt

thorsten.clausius@

sec.informatik.tu-darmstadt.de

Jörg Cornelius Schneider

Deutsche Bank AG

joerg-cornelius.schneider@db.com

js@joergschneider.com

www.cryptool.de

www.cryptool.org

www.cryptool.com

Mailing list: cryptool-list@sec.informatik.tu-darmstadt.de