



## **Aufgabenstellung SN-Labor Versuch 4: CrypTool**

Das Programm CrypTool kann man unter <http://www.cryptool.de/> herunter laden. Auf der Download Seite befindet sich auch das Skript zum Programm.

1. Führen Sie die Demo zum Diffie/Hellman Schlüsseltausch durch (-> Einzelverfahren). Benutzen Sie die Zufallszahlengeneratoren. Dokumentieren Sie alle Einzelschritte und – Ergebnisse. Begründen Sie, warum Alice und Bob am Ende über einen gemeinsamen Schlüssel verfügen. Was ist der Sinn der Operation?
2. Erzeugen Sie mit einer anderen Gruppe gemeinsam einen Sitzungsschlüssel. Verschlüsseln Sie mit diesem Schlüssel unter Verwendung von DES einen kurzen Text und entschlüsseln den Text der anderen Gruppe. Wählen Sie beliebig eine der beiden DES Methoden und erläutern Sie die Unterschiede der beiden.
3. Führen Sie die Demo für RSA aus (-> Einzelverfahren -> RSA-Demo -> RSA Kryptosystem). Wählen Sie geeignete Primzahlen: Miller Rabin Test für Primzahlen, Wertebereich zwischen  $2^4$  und  $2^5$  für  $p$  und  $q$  wählen. Dokumentieren Sie alle Einzelschritte und – Ergebnisse. Warum ist der angezeigte Wert für  $e$  geeignet? Wie wird der geheime Schlüssel errechnet? Geben sie nun zum Verschlüsseln einen mindestens zwanzig Buchstaben langen Text ein, der weder Satzzeichen noch Umlaute enthält.
4. Benutzen Sie nun das obige Tool zur RSA- verschlüsselten Kommunikation mit einer anderen Gruppe, Bedingungen wie oben. Welche Informationen müssen Sie an die andere Gruppe geben? Dokumentieren Sie alle Einzelschritte und – Ergebnisse.
5. Führen Sie nun einen RSA-Faktorisierungsangriff auf die erhaltene RSAverschlüsselte Nachricht aus (> CrypTool > Hilfe). Erläutern Sie die Vorgehensweise!
6. Führen Sie die Signatur-Demo durch. Als Text benutzen Sie wieder den im Anhang stehenden Txt „Rechtsfragen“ oder den „Brian-Text“ . Alle Zwischenschritte dokumentieren! Hash: SHA-1, Primzahlen zwischen  $2^{150}$  und  $2^{151}$  für  $p$  und  $q$  wählen, Fermat-Test. Erzeugen Sie ein Zertifikat.
  - a) Verifizieren Sie die Signatur.
  - b) Ist ein Faktorisierungs-Angriff sinnvoll? Begründung? Wenn ja, bitte durchführen.
7. Greifen Sie den in Anhang 2 gegebenen Text (Hinweis: Vigenère, englisch, Analyse > allgemein > N-Gramm) an. Berechnen Sie die Schlüssellänge mit der Friedman Formel und beschreiben Sie, wie man vorgeht, wenn man nur das N-Gramm als Hilfe hat. (Tipp: Häufigkeitsverteilung der Buchstaben – englisch)



**1. Führen Sie die Demo zum Diffie/Hellman Schlüsseltausch durch (-> Einzelverfahren). Benutzen Sie die Zufallszahlengeneratoren. Dokumentieren Sie alle Einzelschritte und – Ergebnisse. Begründen Sie, warum Alice und Bob am Ende über einen gemeinsamen Schlüssel verfügen. Was ist der Sinn der Operation?**

Um die Aufgabe durchführen zu können, wurde zunächst das CrypTool installiert (siehe Abbildung 1).



Abbildung 1 – Installation CrypTool

Anschließend wählten wir über die Registerkarte „Einzelverfahren“ die „Diffie-Hellman-Demo“ aus (siehe Abbildung 2).

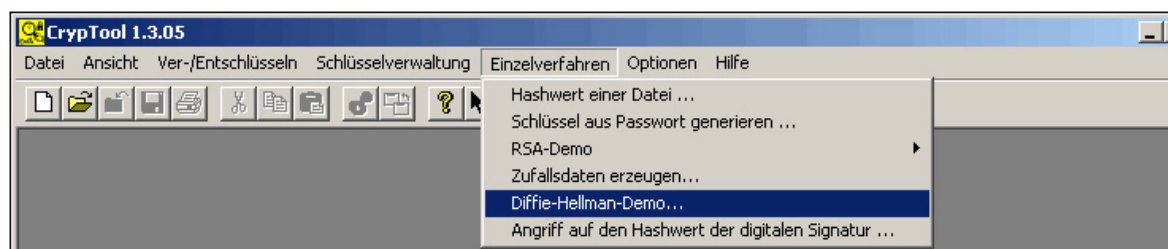


Abbildung 2 – Auswahl „Diffie-Hellman-Demo“

Nun kommen wir zum Eingabefenster, wo wir insgesamt 5 Schritte ausführen müssen, um einen geheimen Schlüssel auszuhandeln (siehe Abbildung 3).

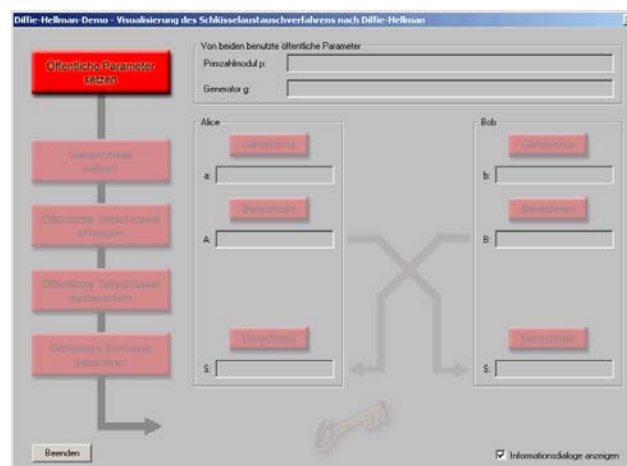


Abbildung 3 – Fenster „geheimen Schlüssel aushandeln“



Im 1. Schritt müssen wir die „Öffentlichen Parameter“ für den Schlüsseltausch eingeben. Das „Primzahlmodul (p)“ und den „Generator (g)“ (siehe Abbildung 4).

Abbildung 4 – Öffentliche Parameter setzen

Das Primzahlmodul soll eine Primzahl sein. Der Einfachheit halber, lassen wir uns eine Primzahl von dem Programm mit einem Klick auf „Primzahl erzeugen“ generieren. Die vorgegeben Werte übernehmen wir und erzeugen eine Primzahl (siehe Abbildung 5).

Abbildung 5 – Primzahl generieren

Anschließend erzeugen wir den Generator mit einem Klick auf „Generator erzeugen“ (siehe Abbildung 6).



**Öffentliche Parameter setzen**

Beide Parameter - sowohl der Generator (g) als auch der Primzahlmodul (p) - dürfen öffentlich bekannt sein.

Der Primzahlmodul (p) sollte eine Primzahl sein. Falls Sie keine große Primzahl kennen, so erzeugen Sie eine durch Klicken des Buttons 'Primzahl erzeugen'.

Primzahlmodul:

Als Generator (g) bezeichnet man eine natürliche Zahl, die nach Möglichkeit nicht Null oder Eins und kein Vielfaches des Primzahlmoduls (p) sein sollte.

Generator:

Abbildung 6 – Generator erzeugen.

Im 2. Schritt werden jeweils nacheinander die Geheimnisse für Alice und Bob erstellt (siehe Abbildung 7).

**Diffie-Hellman-Demo - Visualisierung des Schlüsselaustauschverfahrens nach Diffie-Hellman**

Von beiden benutzte öffentliche Parameter:

Primzahlmodul p:

Generator g:

**Öffentliche Parameter setzen**

**Geheimnisse wählen**

Öffentliche Teilschlüssel erzeugen

Öffentliche Teilschlüssel austauschen

Geheimen Schlüssel generieren

☒ Informationsdialoge anzeigen

**Alice**

a:

A:

S:

**Bob**

b:

B:

S:

Abbildung 7 – Geheimnisse festlegen

Der jeweilige Klick auf „Geheimnis“ genügt um die jeweiligen Geheimnisse zu erzeugen (siehe Abbildung 8 und 9).

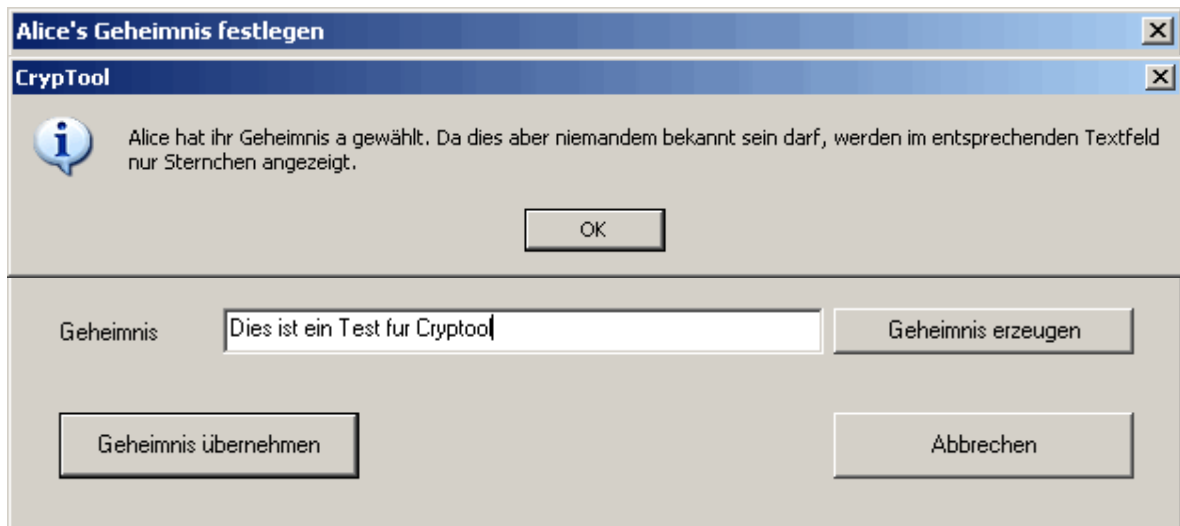


Abbildung 8 – Geheimnis für Alice festgelegt

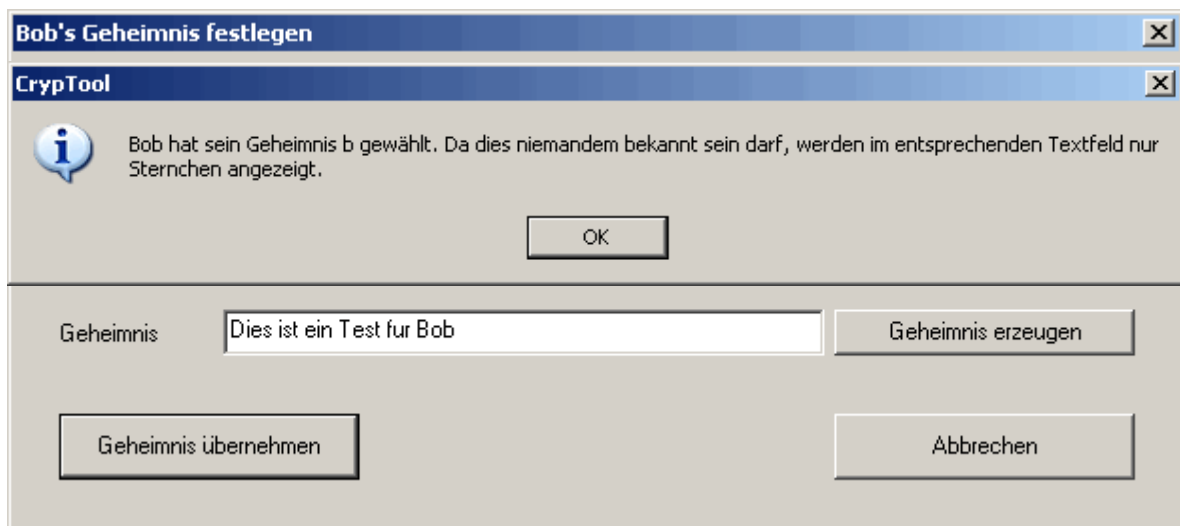


Abbildung 9 – Geheimnis für Bob festgelegt

Im 3. Schritt werden die jeweiligen „öffentlichen Schlüssel“ für die Geheimnisse von Alice und Bob festgelegt und durch einen Klick auf „Berechnen“ berechnet (siehe Abbildung 10).

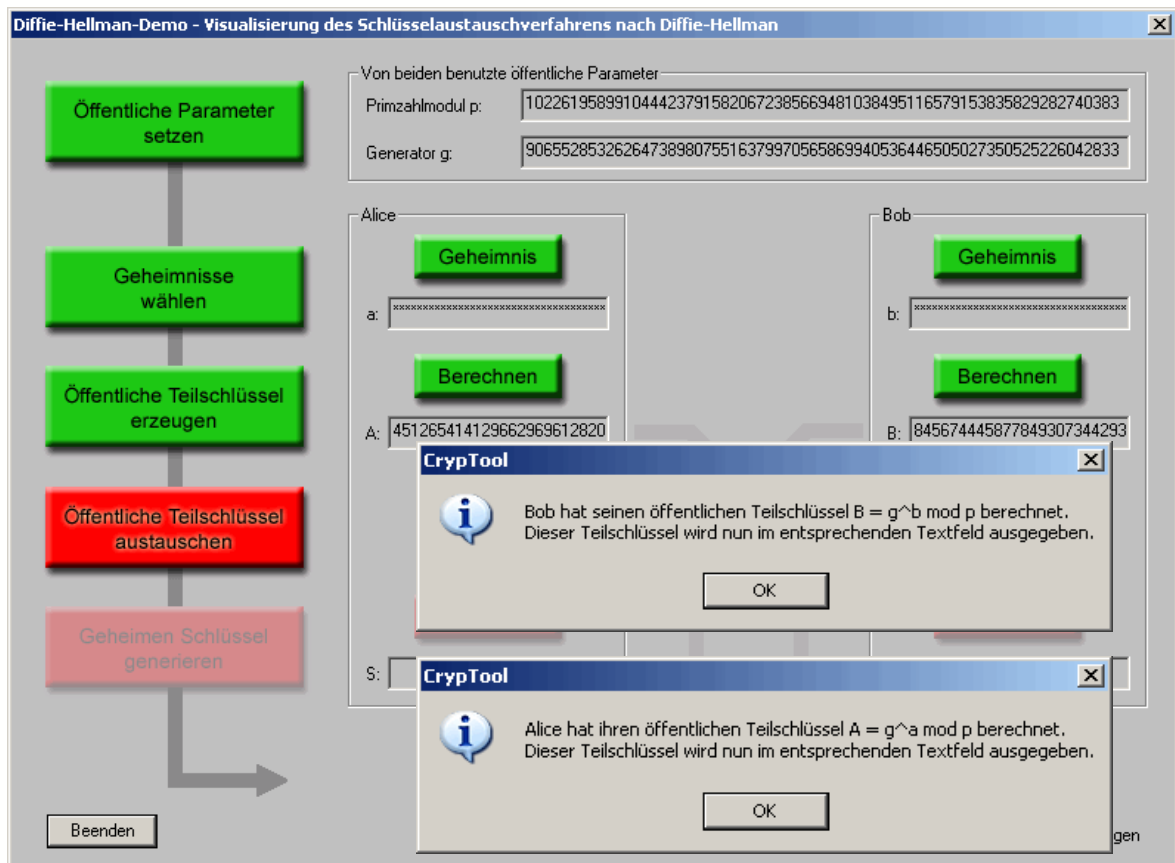


Abbildung 10 – Öffentliche Teilschlüssel erzeugen

Im 4. Schritt werden die beiden öffentlichen Schlüssel ausgetauscht, damit Alice und Bob ihren gemeinsamen Schlüssel S berechnen können. Dies geschieht durch Klicken auf „Öffentliche Teilschlüssel austauschen“ (siehe Abbildung 11).

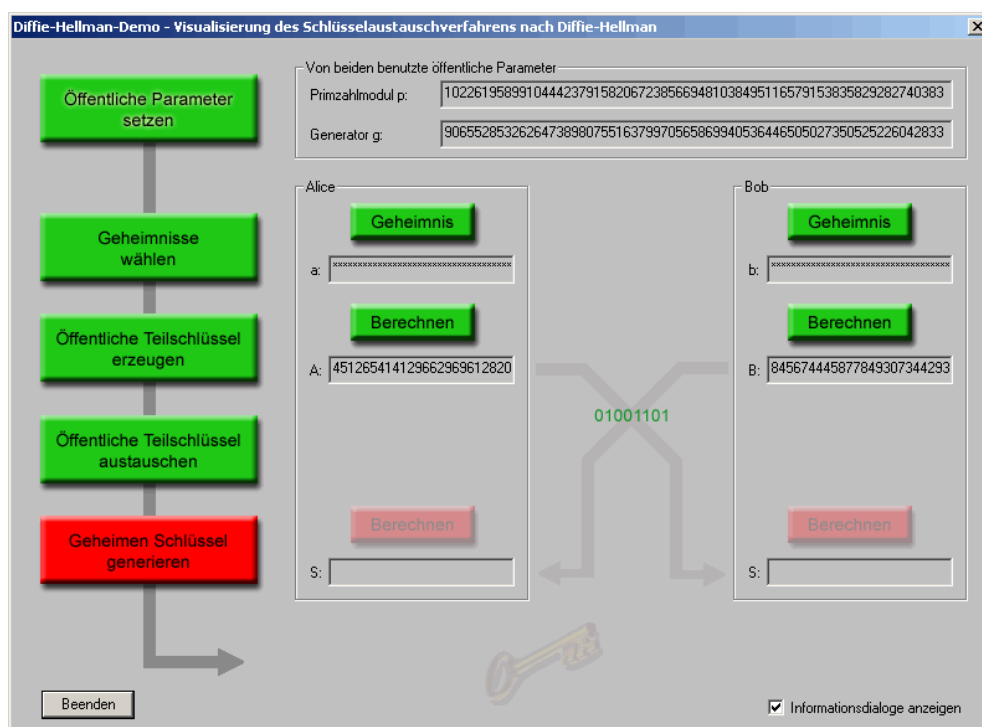


Abbildung 11 – Öffentliche Teilschlüssel austauschen



Im 5. Schritt wird der geheime und gemeinsame Schlüssel  $S$  für Alice und Bob berechnet (siehe Abbildung 12).

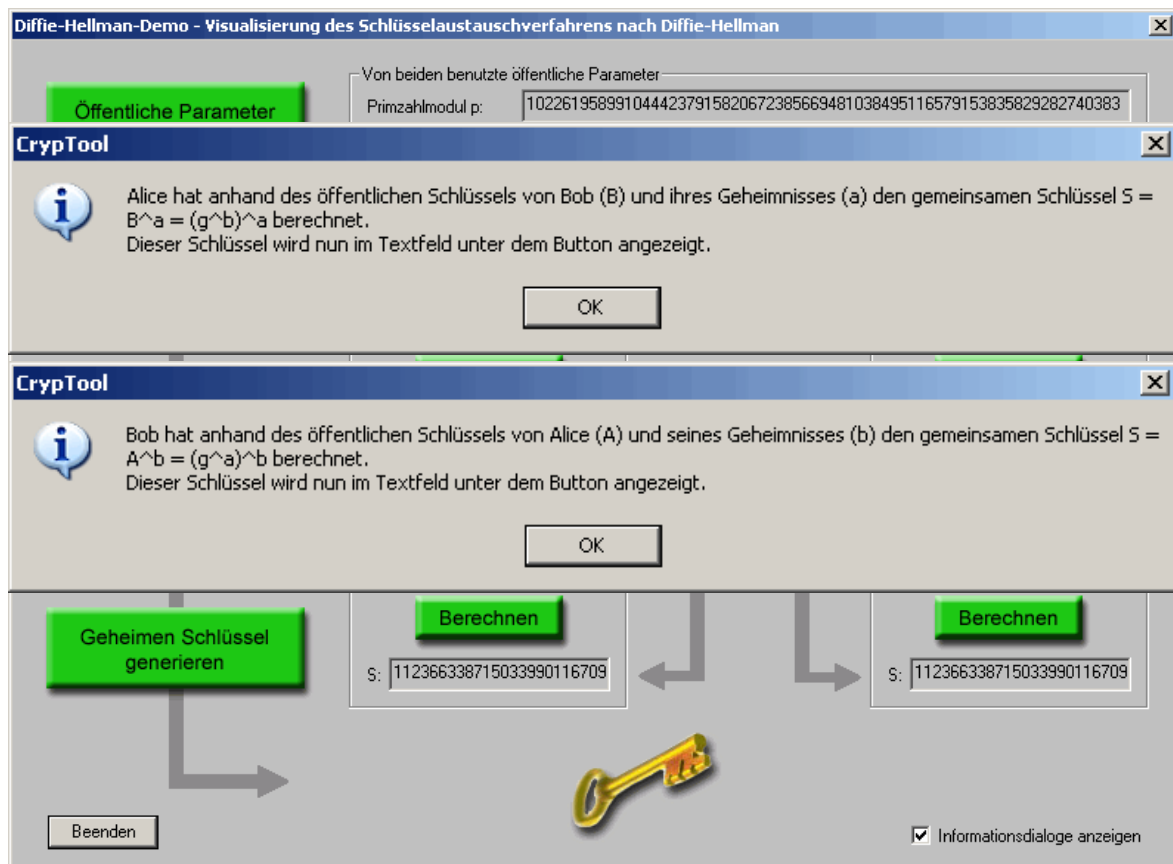


Abbildung 12 – Geheimen Schlüssel generieren

Anschließend haben wir die erfolgreiche Statusmeldung „Diffie-Hellman-Schlüsselaustausch erfolgreich“ erhalten (siehe Abbildung 13).

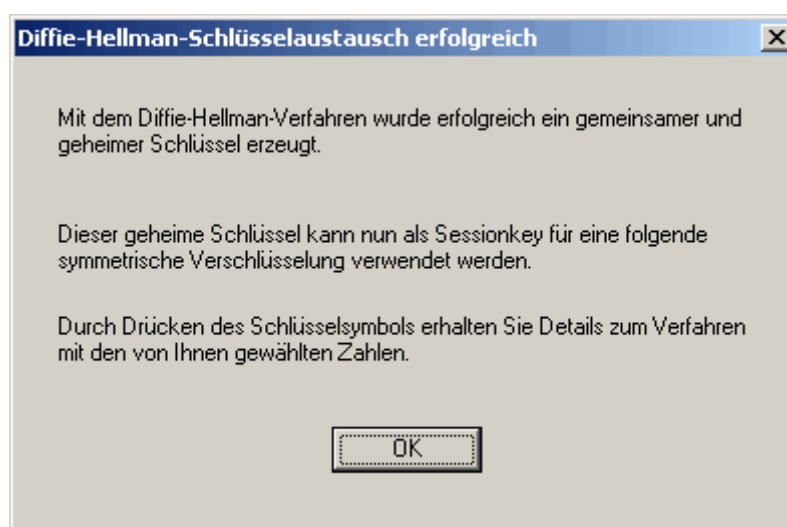


Abbildung 13 – Schlüsselaustausch erfolgreich

Nach dem erfolgreichen Schlüsselaustausch, kann man sich noch die Schlüssel-details durch Klicken auf den gelben Schlüssel anzeigen lassen (s. Abbildung 14).



Schlüssel-Details - Geheimer und gemeinsamer Schlüssel von Alice und Bob

Unabhängig voneinander haben Alice und Bob jeweils für sich den Session Key berechnet:

Alice's Session Key S: 95068710278421734003116591551077178736158833702035681112366338715033990116709  
Schlüssellänge [bit]: 256

Bob's Session Key S: 95068710278421734003116591551077178736158833702035681112366338715033990116709  
Schlüssellänge [bit]: 256

Verifikation  
Verifikation erfolgreich! - Die Schlüssel von Alice und Bob sind identisch.

Damit ist das Ziel erreicht

Jede Partei (Alice und Bob) berechnete für sich aus den eigenen geheimen Informationen und aus den übertragenen Informationen das gemeinsame Geheimnis.

Dieses Geheimnis bzw. den Session Key kann ein "Lauscher" nicht in Erfahrung bringen, wenn die gewählten Parameter groß genug sind. Der Session Key kann nun z.B. für eine symmetrische Verschlüsselung zwischen Alice und Bob verwendet werden.

OK Log-Text anzeigen Abbrechen

Abbildung 14 – Ansicht der Schlüsseldetails

Um noch mal eine Übersicht über alle zuvor getätigten Schritte zu bekommen, lassen wir uns den „Log-Text anzeigen“ (siehe Abbildung 15).

CrypTool

Der Protokollverlauf mit Ihren Zahlen wurde erfolgreich in ein Fenster im CrypTool-Hauptprogramm geschrieben.

OK

Abbildung 15 – Protokollverlauf in Fenster schreiben

### Logdatei:

Zunächst einigten sich Alice und Bob auf die öffentlichen Parameter des Schlüsselaustauschs. Hierzu wählten sie eine Primzahl  $p$  und einen Generator  $g$ :

$p$ :  
10226195899104442379158206723856694810384951165791538358292827403  
8327147593447

$g$ :  
90655285326264738980755163799705658699405364465050273505252260428  
330000571845





Nun wählten sowohl Alice als auch Bob jeweils für sich ihre Geheimzahlen. Alice und Bob legten ihre Geheimnisse a und b wie folgt fest:

a:

77228843981382115201469655675616326379263576503389846551224211600  
855174536515

b:

36373855713447201821558088679848752381080826509104518692245350692  
75334286946

Anhand ihrer zuvor gewählten Geheimnisse erzeugte Alice ihren öffentlichen Teilschlüssel A. Bob erzeugte parallel dazu seinen öffentlichen Teilschlüssel B:

A:

75051155382429275217347870502369532137713471622301845451265414129  
662969612820

B:

68264136174893079749683152606310151648053661024264293845674445877  
849307344293

Als nächstes tauschten Alice und Bob ihre öffentlichen Teilschlüssel aus, d.h. Alice sendete ihren Schlüssel A an Bob und Bob sendete seinen Schlüssel B an Alice.

Nun konnten Alice und Bob jeweils den geheimen und gemeinsamen Session Key erzeugen. Alice berechnete ihren Schlüssel SA, Bob berechnete den Schlüssel SB:

SA:

95068710278421734003116591551077178736158833702035681112366338715  
033990116709

SB:

95068710278421734003116591551077178736158833702035681112366338715  
033990116709

Theoretisch wäre es Alice und Bob nun möglich, mit Hilfe des jeweils berechneten Session Keys SA und SB untereinander auszutauschende Dokumente zum Schutz vor Dritten zu verschlüsseln bzw. zu entschlüsseln.

### **Sinn und Zweck:**

Zwei Kommunikationspartner, z.B. Alice und Bob, wollen einen geheimen Schlüssel über einen unsicheren Kanal aushandeln. Dabei soll jedoch kein Angreifer die Möglichkeit haben, den ausgehandelten Schlüssel zu erlangen, auch wenn er den Kanal komplett abhören kann. Dieses Problem wird mit dem Schlüsselaustauschverfahren nach Diffie-Hellman gelöst.



**2. Erzeugen Sie mit einer anderen Gruppe gemeinsam einen Sitzungsschlüssel. Verschlüsseln Sie mit diesem Schlüssel unter Verwendung von DES einen kurzen Text und entschlüsseln den Text der anderen Gruppe. Wählen Sie beliebig eine der beiden DES Methoden und erläutern Sie die Unterschiede der beiden.**

Zunächst müssen wir für die 2. Aufgabe einen „Schlüssel aus Passwort generieren“ (s. Abbildung 16), damit wir den erstellen Text verschlüsseln können.

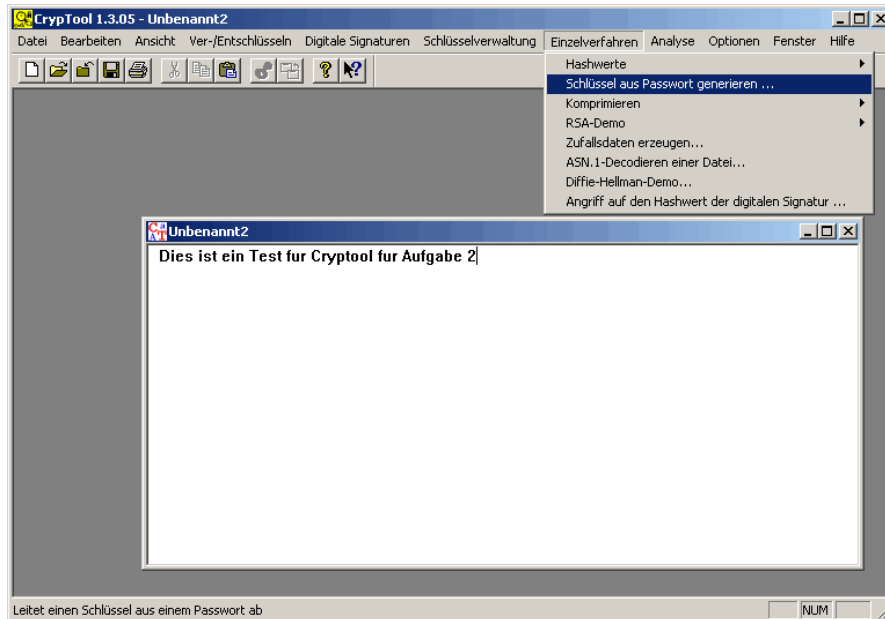


Abbildung 16 – Schlüssel aus Passwort generieren I

Um ein Passwort generieren zu können, sind folgenden Eingaben notwendig: Passwort „0815“, Länge des Ausgabeschlüssels „8 Bytes“  
Anschließend wird der Schlüssel aus dem Passwort mit einem Klick auf „Schlüssel erzeugen“ generiert (siehe Abbildung 17).

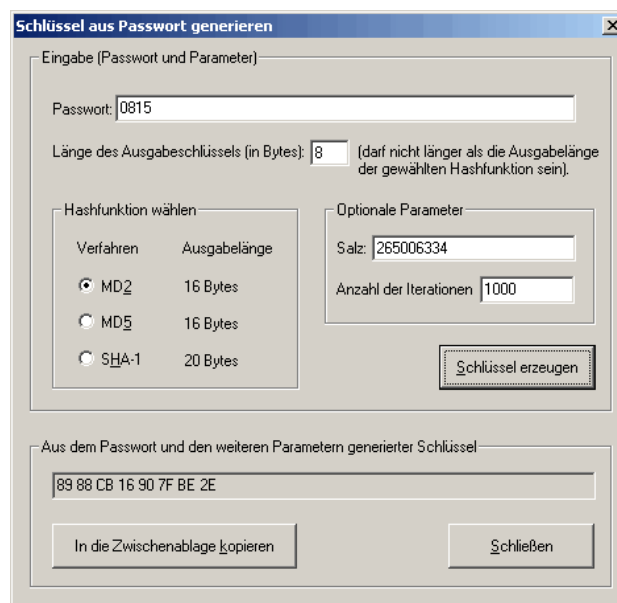


Abbildung 17 – Schlüssel aus Passwort generieren II



Nun wird der erstellte Text verschlüsselt. Dies geschieht über die Registerkarte „Ver-/Entschlüsseln > Symmetrisch“. Wir werden aufgefordert, den zuvor erstellen 8 Byte Schlüssel einzugeben, um den Text verschlüsseln zu können (siehe Abbildung 18).

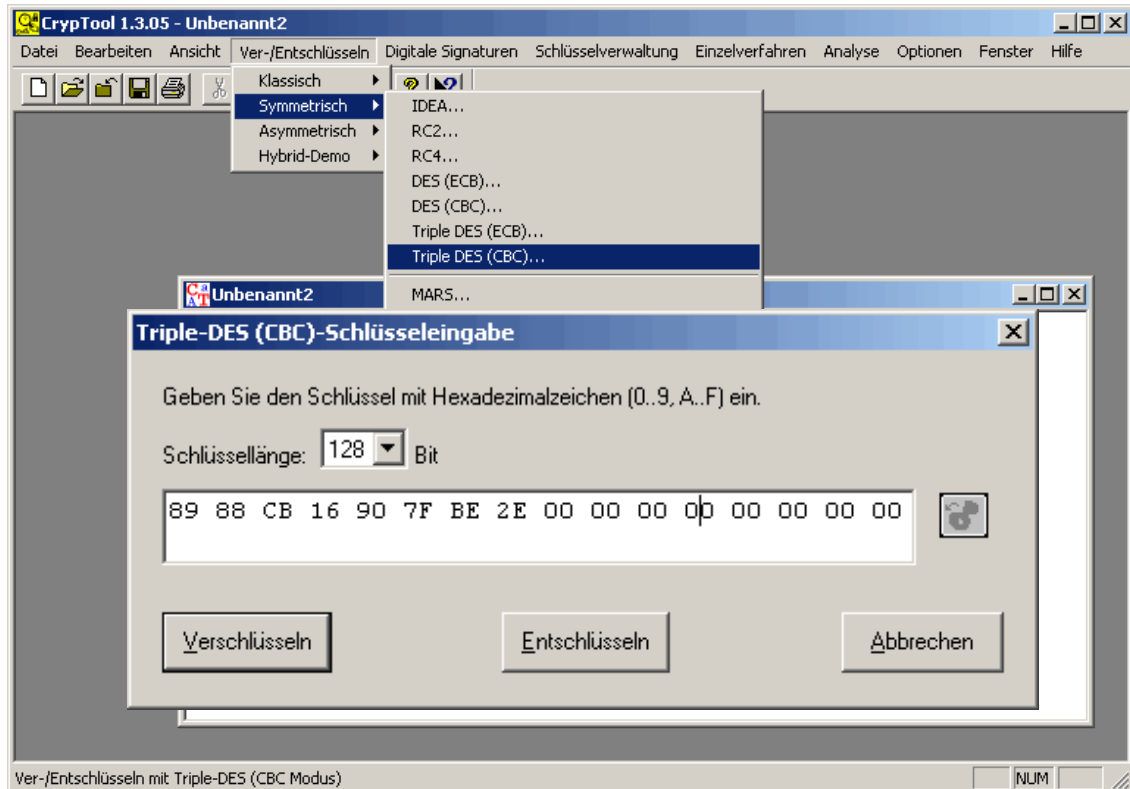


Abbildung 18 – Text verschlüsseln

Hier sieht man den verschlüsselten Text (siehe Abbildung 19):

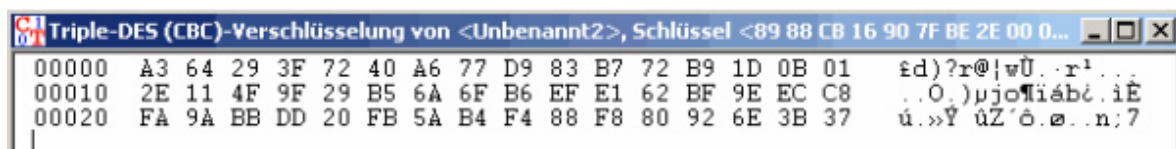


Abbildung 19 – Verschlüsselter Text

Nun wird der verschlüsselte Text für unsere Nachbargruppe gespeichert (siehe Abbildung 20).

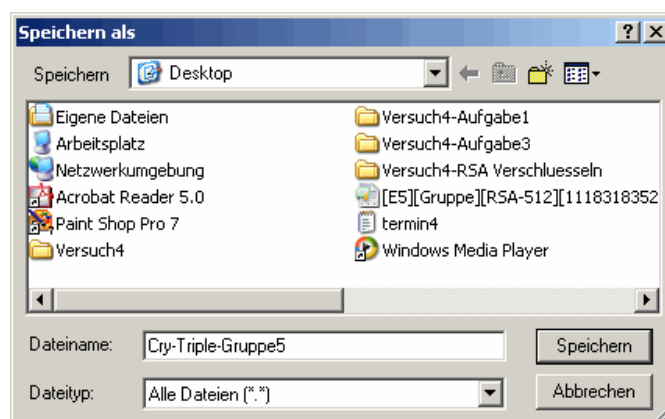


Abbildung 20 – Verschlüsselter Text speichern



Den von der Nachbargruppe bekommenen Schlüssel öffnen wir mit dem CrypTool und entschlüsseln ihn über die Registerkarte „Ver-/Entschlüsseln“. Hier ist dabei darauf zu achten, dass wir auch die Methode auswählen, mit der unsere Nachbargruppe den Text verschlüsselt hat (DES CBC).

Nun wird noch der von der Nachbargruppe erstellte Schlüssel zum Entschlüsseln eingegeben (siehe Abbildung 21).

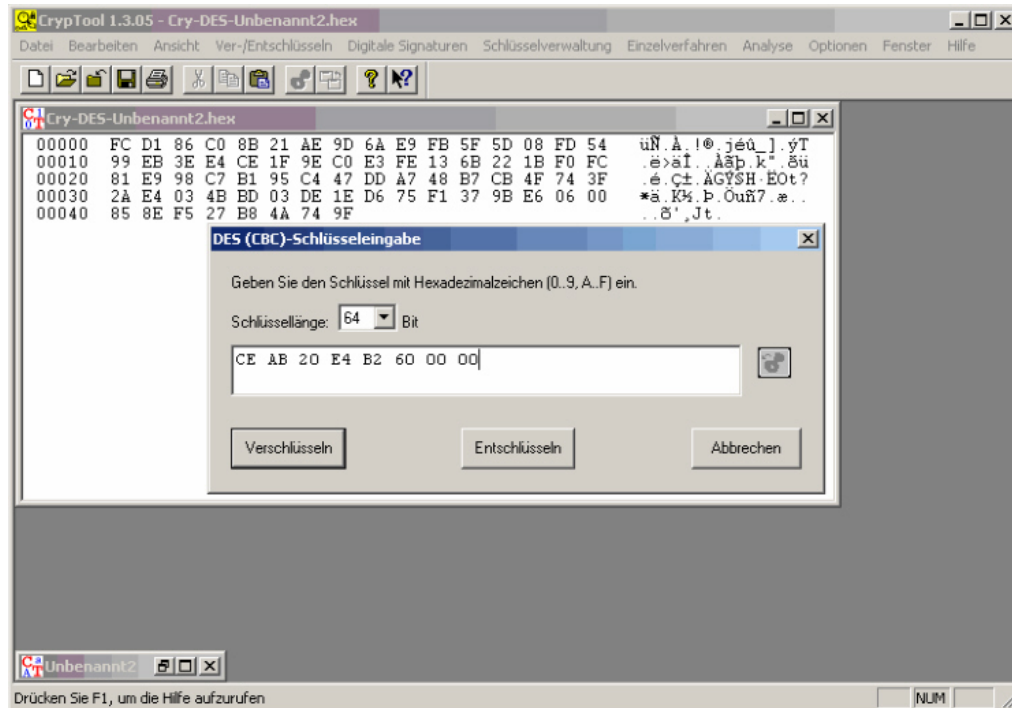


Abbildung 21 – Text entschlüsseln

Jetzt erhalten wir den entschlüsselten Text von unserer Nachbargruppe (siehe Abbildung 22)

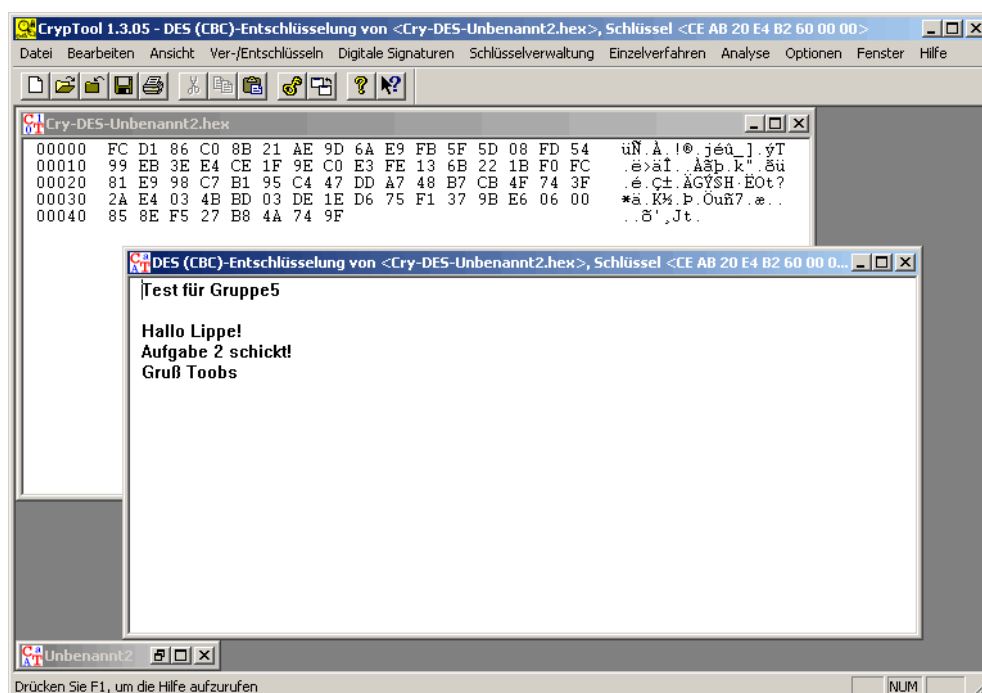


Abbildung 22 – Text entschlüsselt



## Unterschiede zwischen den beiden DES Methoden:

### ECB:

- ECB → Electronic Code Book
- 64-Bit Klartextblöcke werden nacheinander und getrennt voneinander sowie kontextfrei verschlüsselt
- Einfachste Anwendungsmöglichkeit eines Blockchiffre

### CBC:

- CBC → Cipher Block Chaining Mode
- Vor der Verschlüsselung eines Klartextblocks wird der bereits verschlüsselte vorhergehende Block hinzuaddiert
- Auf den ersten Klartextblock, der ja keinen Vorgänger besitzt, wird statt dessen ein Initialisierungsvektor, der geheim gehalten werden muss, addiert
- Dadurch entsteht eine Verkettung (engl. chain) der Blöcke
- Trotz dieser Verkettung ist die CBC-Verschlüsselung „selbstkorrigierend“: Fehler/Manipulation in einem Block wirken sich bei der Entschlüsselung nur auf den fehlerhaften/manipulierten und den nächsten Block aus. Der Nachteil ist, dass man nicht auf einen beliebigen Block des Geheimtextes zugreifen kann sondern immer den gesamten Text entschlüsseln muss

**3. Führen Sie die Demo für RSA aus (-> Einzelverfahren -> RSA-Demo -> RSA Kryptosystem). Wählen Sie geeignete Primzahlen: Miller Rabin Test für Primzahlen, Wertebereich zwischen  $2^4$  und  $2^5$  für p und q wählen. Dokumentieren Sie alle Einzelschritte und – Ergebnisse. Warum ist der angezeigte Wert für e geeignet? Wie wird der geheime Schlüssel errechnet? Geben sie nun zum Verschlüsseln einen mindestens zwanzig Buchstaben langen Text ein, der weder Satzzeichen noch Umlaute enthält.**

Zunächst starten wir die RSA-Demo über den Menüpunkt „Einzelverfahren > RSA-Demo > RSA-Kryptosystem...“ (siehe Abbildung 23 und Abbildung 24).

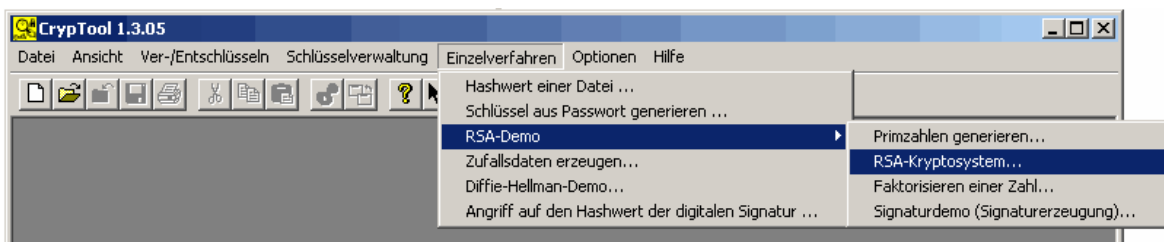


Abbildung 23 – Auswahl RSA-Kryptosystem

Danach müssen wir 2 Primzahlen, welche im Bereich zwischen  $2^4$  und  $2^5$  liegen sollten, nach dem Miller-Rabin-Test erzeugen (siehe Abbildung 25). Das Ergebnis ( $p = 19$ ,  $q = 17$ ) wird in das vorherige Fenster übernommen, aus den Primzahlen werden nun öffentlicher und geheimer Schlüssel nach den danach folgenden Formeln berechnet (siehe Abbildung 26).



Abbildung 24 –  
Übersicht  
RSA-Krypto-  
system

Abbildung 25 – Eingabe der Unter- und Obergrenze zur Primzahlerzeugung



The image shows the CrypTool application interface. A dialog box titled 'Primzahlen generieren' (Generate Primes) is open, showing the process of generating two prime numbers, p and q. The dialog includes instructions, algorithm selection (Miller-Rabin-Test, Solovay-Strassen-Test, Fermat-Test), and range selection (independent or equal). Below the dialog, the main interface shows the generated primes (p=19, q=17) and the calculated RSA parameters: N=323, phi(N)=288, public key e=2^16+1, and private key d=161. The interface also includes fields for inputting a message and buttons for encryption and decryption.

**Primzahlen generieren**

Primzahlen spielen in der modernen Kryptographie eine wichtige Rolle. Hier erzeugen Sie sich zwei zufällige Primzahlen p und q aus dem Wertebereich [Untergrenze, ..., Obergrenze].

Algorithmen zur Generierung

- ☒ Miller-Rabin-Test
- ☐ Solovay-Strassen-Test
- ☐ Fermat-Test

Wertebereich der beiden Primzahlen

- ☒ unabhängig voneinander eingeben
- ☐ beide gleich (nur einen eingeben)

Primzahl p

Untergrenze:

Obergrenze:

Ergebnis:

Primzahl q

Untergrenze:

Obergrenze:

Ergebnis:

Primzahlen generieren Primzahlen übernehmen Abbrechen

Primzahleingabe

Primzahl p:

Primzahl q:

Primzahlen generieren...

RSA-Parameter

RSA-Modul N:  (öffentlich)

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

Öffentlicher Schlüssel e:

Geheimer Schlüssel d:

Parameter aktualisieren

RSA-Verschlüsselung mit e / Entschlüsselung mit d

Eingabe als ☒ Text ☐ Zahlen

Eingabe der zu ver- oder entschlüsselnden Nachricht als Text oder als Hex-Dump.

Verschlüsseln Entschlüsseln Schließen

Abbildung 26 – Erzeugung der RSA-Parameter



Das Programm hat nun automatisch die übrigen Parameter-Felder ergänzt.  
Wir erhalten folgende Werte:

RSA-Modul  $N = p \cdot q = 19 \cdot 17 = 323$

Öffentlicher Schlüssel  $e = 2^{16} + 1$

Geheimes Element  $\phi(N) = (p - 1) \cdot (q - 1) = 18 \cdot 16 = 288$

Geheimer Schlüssel  $d = e^{(-1)} \cdot (\text{mod } \phi(N)) = 161$

Der Wert des öffentlichen Schlüssels  $e$  ist dabei geeignet, da  $e$  teilerfremd zu  $\phi(N)$  sein muss.

**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 öffentlichen RSA-Parameter angeben: den RSA-Modul  $N$  und den öffentlichen Schlüssel  $e$ .

Primzahleingabe

Primzahl  $p$ : 19

Primzahl  $q$ : 17

Primzahlen generieren...

RSA-Parameter

RSA-Modul  $N$ : 323 (öffentlich)

$\phi(N) = (p-1)(q-1)$ : 288 (geheim)

Öffentlicher Schlüssel  $e$ :  $2^{16}+1$

Geheimer Schlüssel  $d$ : 161

Parameter aktualisieren

RSA-Verschlüsselung mit  $e$  / Entschlüsselung mit  $d$

Eingabe als ☒ Text ☐ Zahlen

Optionen für Alphabet und Zahlensystem...

Eingabe der zu ver- oder entschlüsselnden Nachricht als Text oder als Hex-Dump.

Dies ist ein Test für RSA Kryptosystem

Verschlüsseln Entschlüsseln Schließen

Abbildung 27 – eingegebener Text zum Verschlüsseln





Nun benötigen wir einen kurzen Text ohne Satz- und Sonderzeichen (siehe Abbildung 27), welchen wir verschlüsseln lassen. Hierzu zerlegt das Programm den Text in 1-Zeichen große Blöcke und trennt jedes Zeichen durch eine Raute (#), welche als Trennzeichen dient. Danach wird der Text in ASCII-Werte umgewandelt und diese dann mittels  $c[i] = m[i]^e \cdot (\text{mod } N)$  verschlüsselt (siehe Abbildung 28).

**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 öffentlichen RSA-Parameter angeben: den RSA-Modul N und den öffentlichen Schlüssel e.

Primzahleingabe

Primzahl p: 19

Primzahl q: 17

Primzahlen generieren...

RSA-Parameter

RSA-Modul N: 323 (öffentlich)

$\phi(N) = (p-1)(q-1)$ : 288 (geheim)

Öffentlicher Schlüssel e:  $2^{16}+1$

Geheimer Schlüssel: 161

Parameter aktualisieren

RSA-Verschlüsselung mit e / Entschlüsselung mit d

Eingabe als: ☒ Text ☐ Zahlen

Optionen für Alphabet und Zahlensystem...

Eingabetext

Dies ist ein Test für RSA Kryptosystem

Der Eingabetext wird in Blöcke der Länge 1 aufgeteilt (das Symbol '#' dient als Trennzeichen).

D # i # e # s # t # f # u # r # R # S # A # K # r # y # p

Zahlendarstellung der Eingabe zur Basis 10.

068 # 105 # 101 # 115 # 032 # 105 # 115 # 116 # 032 # 101 # 105 # 110 # 032 # 084 # 101 # 115 # 116 #

Verschlüsselung in den Chiffretext  $c[i] = m[i]^e \pmod{N}$ .

102 # 173 # 016 # 115 # 117 # 173 # 115 # 048 # 117 # 016 # 173 # 280 # 117 # 050 # 016 # 115 # 048 #

Verschlüsseln Entschlüsseln Schließen

Abbildung 28 – Text wird in 3 Schritten verschlüsselt



**4. Benutzen Sie nun das obige Tool zur RSA- verschlüsselten Kommunikation mit einer anderen Gruppe, Bedingungen wie oben. Welche Informationen müssen Sie an die andere Gruppe geben? Dokumentieren Sie alle Einzelschritte und – Ergebnisse.**

Ebenso wie schon in Aufgabe 3 starten wir die RSA-Demo wieder über den Menüpunkt „Einzelverfahren > RSA-Demo > RSA-Kryptosystem...“ (siehe Abbildung 29).

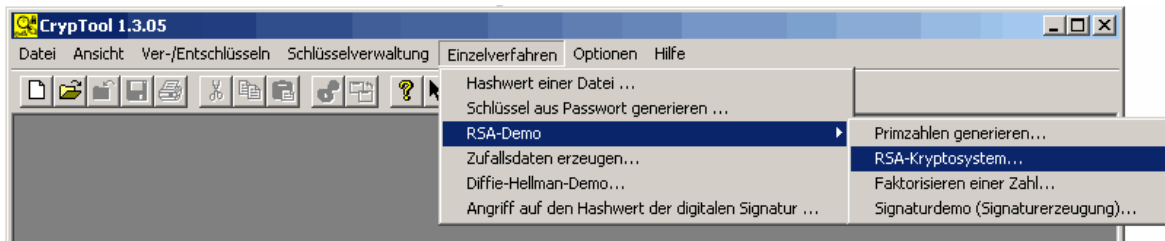


Abbildung 29 – Erneutes Starten des RSA-Kryptosystems

Diesmal benötigen wir aber nur den Schlüssel N unserer Nachbargruppe und deren öffentlichen Schlüsseln e. Danach geben wir den zu verschlüsselnden Text ein und verschlüsseln über den Button „Verschlüsseln“ (siehe Abbildung 30). Durch den Ausfall des Screenshotprogramms haben wir die folgenden zwei Screenshots von einer anderen Gruppe erhalten.



**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 öffentlichen 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 $N$	<input type="text" value="437"/>	(öffentlich)
$\phi(N) = (p-1)(q-1)$	<input type="text"/>	(geheim)
Öffentlicher Schlüssel $e$	<input type="text" value="2^16+1"/>	
Geheimer Schlüssel	<input type="text"/>	

**RSA-Verschlüsselung mit  $e$  / Entschlüsselung mit  $d$**

Eingabe als ☒ Text ☐ Zahlen 

Eingabe der zu ver- oder entschlüsselnden Nachricht als Text oder als Hex-Dump.

Abbildung 30 – Verschlüsseln des Textes für die Nachbargruppe



**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 öffentlichen 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-Modul faktorisieren...

**RSA-Parameter**

RSA-Modul N:  (öffentlich)

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

Öffentlicher Schlüssel e:

Geheimer Schlüssel:  Parameter aktualisieren

**RSA-Verschlüsselung mit e / Entschlüsselung mit d**

Eingabe als: ☒ Text ☐ Zahlen Optionen für Alphabet und Zahlensystem...

Eingabetext:

Der Eingabetext wird in Blöcke der Länge 1 aufgeteilt (das Symbol '#' dient als Trennzeichen).

Zahlendarstellung der Eingabe zur Basis 10.

Verschlüsselung in den Chiffretext  $c[i] = m[i]^e \pmod{N}$ .

Verschlüsseln Entschlüsseln Schließen

Abbildung 31 – Text für Nachbargruppe ist fertig verschlüsselt

Im Labor hatten wir als Text „Dies ist ein Test für Aufgabe 4“ eingetragen und mit dem RSA-Modul N mit dem Wert 323 (öffentlicher Schlüssel e:  $2^{16} + 1$ ) verschlüsselt. Wir erhielten dabei folgendes Chifftrat:

102 # 173 # 016 # 115 # 117 # 173 # 115 # 048 # 117 # 016 # 173 # 280 # 117 #  
050 # 016 # 115 # 048 # 117 # 068 # 032 # 114 # 117 # 031 # 032 # 068 # 069 #  
029 # 013 # 016 # 117 # 205

Auch wir haben einen Text von unserer Nachbargruppe erhalten, welchen wir mit unseren Primzahlen und dem öffentlichen Schlüssel e wieder entschlüsselt haben (siehe Abbildung 32).



**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 öffentlichen RSA-Parameter angeben: den RSA-Modul N und den öffentlichen Schlüssel e.

Primzahleingabe

Primzahl p: 31

Primzahl q: 17

Primzahlen generieren...

RSA-Parameter

RSA-Modul N: 527 (öffentlich)

$\phi(N) = (p-1)(q-1)$ : 480 (geheim)

Öffentlicher Schlüssel e:  $2^{16}+1$

Geheimer Schlüssel: 353

Parameter aktualisieren

RSA-Verschlüsselung mit e / Entschlüsselung mit d

Eingabe als ☐ Text ☒ Zahlen

Optionen für Alphabet und Zahlensystem...

Chiffretext in Zahlendarstellung zur Basis 10:

# 033 # 445 # 033 # 148 # 032 # 202 # 033 # 445 # 353 # 299 # 445 # 033 # 023 # 184 # 033 # 511 # 184

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

068 # 105 # 101 # 115 # 032 # 105 # 115 # 116 # 032 # 101 # 105 # 110 # 032 # 118 # 246 # 108 # 108 #

Ausgabertext aus der Entschlüsselung (in Blöcken der Länge 1; das Symbol '#' dient nur als Trennzeichen).

D # i # e # s # # # i # s # t # # e # i # n # # v # ö # l # l # i # g # # s # i # n # n # f # r # e # i # e # r #

Klartext

Dies ist ein völlig sinnfreier Beispieltext

Verschlüsseln Entschlüsseln Schließen

Abbildung 32 – Entschlüsselter Text der Nachbargruppe

##### 5. Führen Sie nun einen RSA-Faktorisierungsangriff auf die erhaltene RSA-verschlüsselte Nachricht aus (CrypTool > Hilfe). Erläutern Sie die Vorgehensweise!

Durch einen Klick auf „RSA-Modul faktorisieren“ bei vorheriger Auswahl der Vorgehensweise nur mit dem öffentlichen Schlüssel gelangen wir zu einem Menü, in welchem wir das öffentliche RSA-Modul N der Nachbargruppe (in unserem Falle 41449) eingeben müssen (siehe Abbildung 33).



Abbildung 33 – Eingabe der RSA-Parameter und des verschlüsselten Textes

Da wir zunächst nicht wissen können, mit welchem Algorithmus die Zahl faktorisiert werden kann, wählen wir alle Algorithmen aus und klicken im Feld der schrittweisen Faktorisierung auf „Weiter“. Der Computer versucht nun der Reihe nach mit den unterschiedlichen Faktorisierungsalgorithmen die eingegebene Zahl in ihre Primzahlen zu zerlegen.

Schon nach kurzer Zeit erhalten wir eine Erfolgsmeldung und die entsprechende Produktdarstellung der Faktorisierung der eingegebenen Zahl. In unserem Fall hat der PC die Zahl durch die „Brute Force“-Methode geknackt (siehe Abbildung 34).

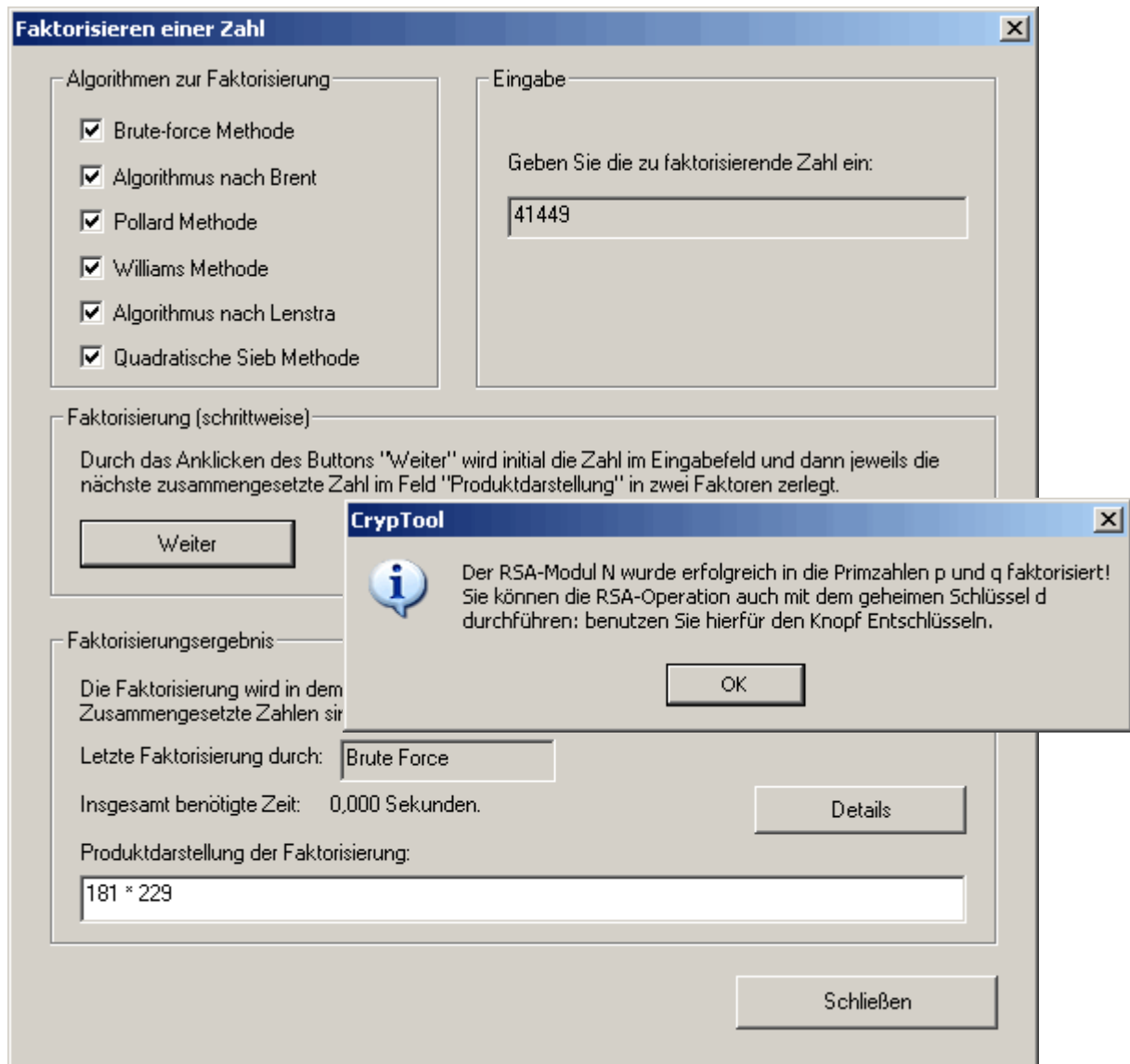


Abbildung 34 – Knacken der zu faktorisierenden Zahl durch Brute-Force

Dadurch hatten wir die Möglichkeit einen anderen verschlüsselten Text zu knacken (siehe Abbildung 35).



**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 öffentlichen RSA-Parameter angeben: den RSA-Modul N und den öffentlichen Schlüssel e.

Primzahleingabe

Primzahl p

Primzahl q

RSA-Parameter

RSA-Modul N  (öffentlich)

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

Öffentlicher Schlüssel e

Geheimer Schlüssel

RSA-Verschlüsselung mit e / Entschlüsselung mit d

Eingabe als ☐ Text ☒ Zahlen 

Chiffretext in Zahlendarstellung zur Basis 10 .

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

Ausgabertext aus der Entschlüsselung (in Blöcken der Länge 1; das Symbol '#' dient nur als Trennzeichen).

Klartext

Abbildung 35 – Entschlüsseln des Textes





**6. Führen Sie die Signatur-Demo durch. Als Text benutzen Sie wieder den im Anhang stehenden Text „Rechtsfragen“ oder den „Brian-Text“. Alle Zwischenschritte dokumentieren! Hash: SHA-1, Primzahlen zwischen  $2^{150}$  und  $2^{151}$  für p und q wählen, Fermat-Test. Erzeugen Sie ein Zertifikat.**

Zunächst starten wir die Signatordemo über den Menüpunkt „Einzelverfahren > RSA-Demo > Signatordemo (Signaturerzeugung)...“ (siehe Abbildung 36).

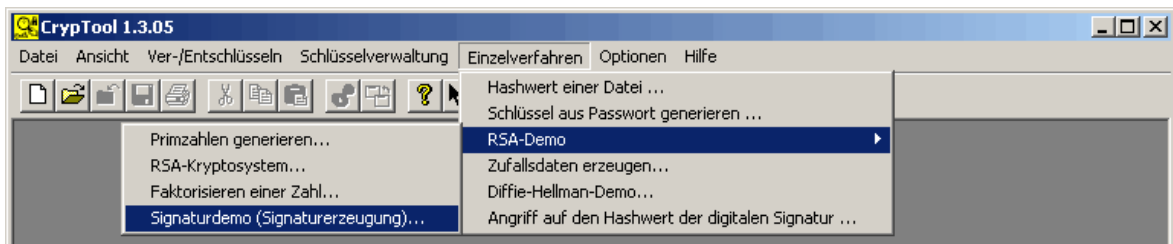


Abbildung 36 – Starten der Signatordemo

Danach öffnet sich ein Programmfenster, welches schrittweise die Signaturerzeugung erklärt und gleichzeitig die Erstellung einer Signatur für eine beliebige Datei bietet (siehe Abbildung 37).

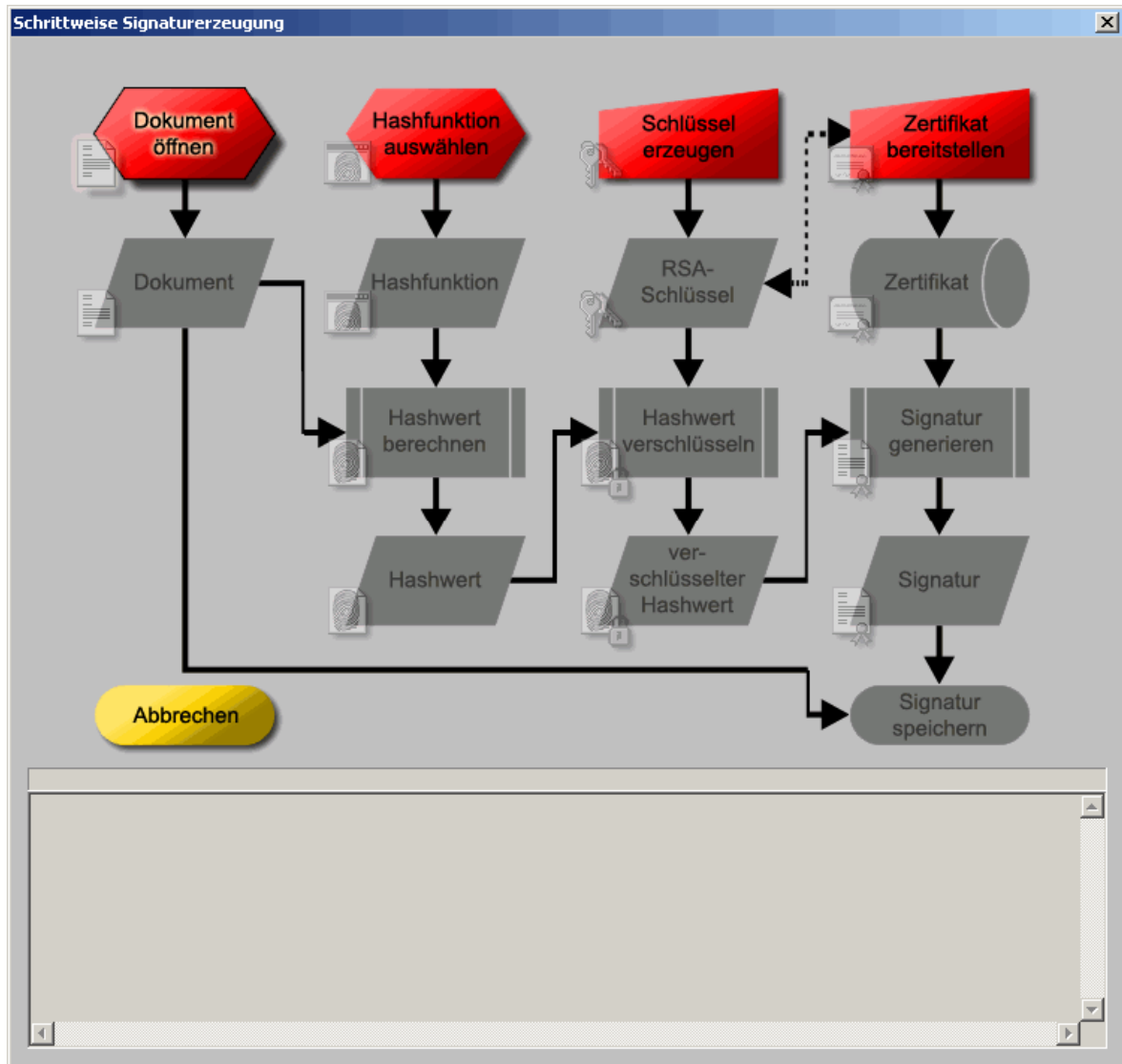


Abbildung 37 – Überblick über die Vorgehensweise

Nun öffnen wir im ersten Schritt eine Datei, in unserem Fall „brian.txt“ (siehe Abbildung 38).

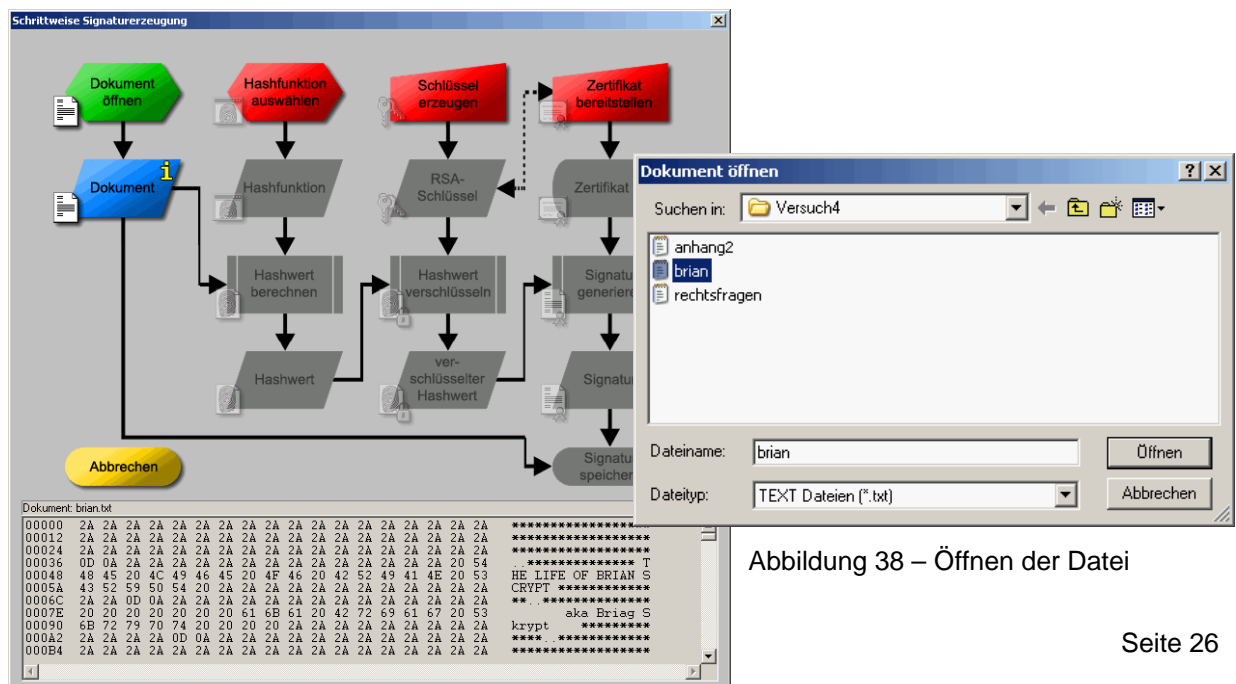


Abbildung 38 – Öffnen der Datei



Im zweiten Schritt müssen wir nun eine Hashfunktion auswählen. Wir entscheiden uns aufgrund der Aufgabenstellung für SHA-1 (siehe Abbildung 39).



Abbildung 39 – Auswahl der Hashfunktion

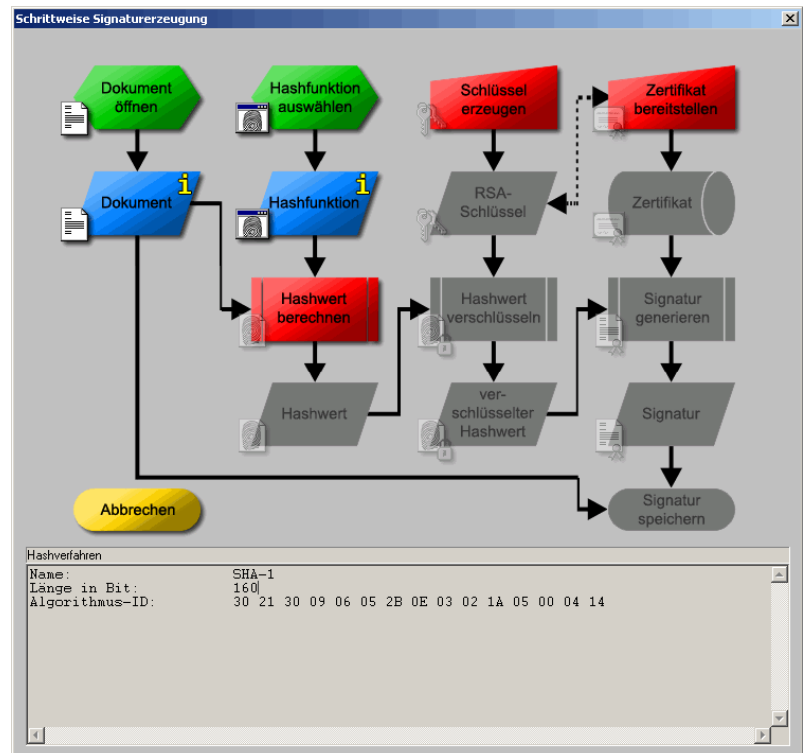


Abbildung 40 – Beendigung des zweiten Schritts

Dies wird uns auch im unteren Teil des Programmfensters angezeigt:

Hash Funktion:

Name: SHA-1

Länge in Bit: 160

Algorithmus-ID: 30 21 30 09 06 05 2B 0E 03 02 1A 05 00 04 14

Nun berechnen wir im nächsten Schritt durch einen Klick den Hash-Wert (siehe Abbildung 41).

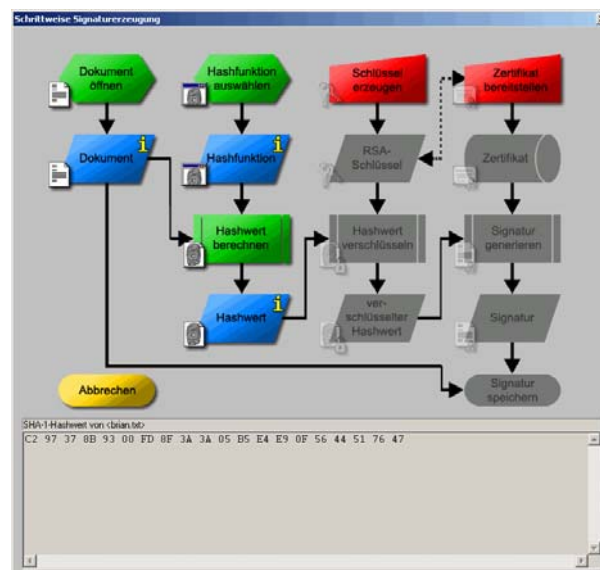


Abbildung 41 – Berechnung des Hash-Werts



Das Ergebnis lautet:

Hash Wert: C2 97 37 8B 93 00 FD 8F 3A 3A 05 B5 E4 E9 0F 56 44 51 76 47

Im vierten Schritt erzeugt das Programm den RSA-Schlüssel. Dabei wählen wir unter Benutzung des Miller-Rabin-Test-Algorithmus Primzahlen zwischen  $2^{150}$  und  $2^{151}$  für p und q (siehe Abbildung 42).

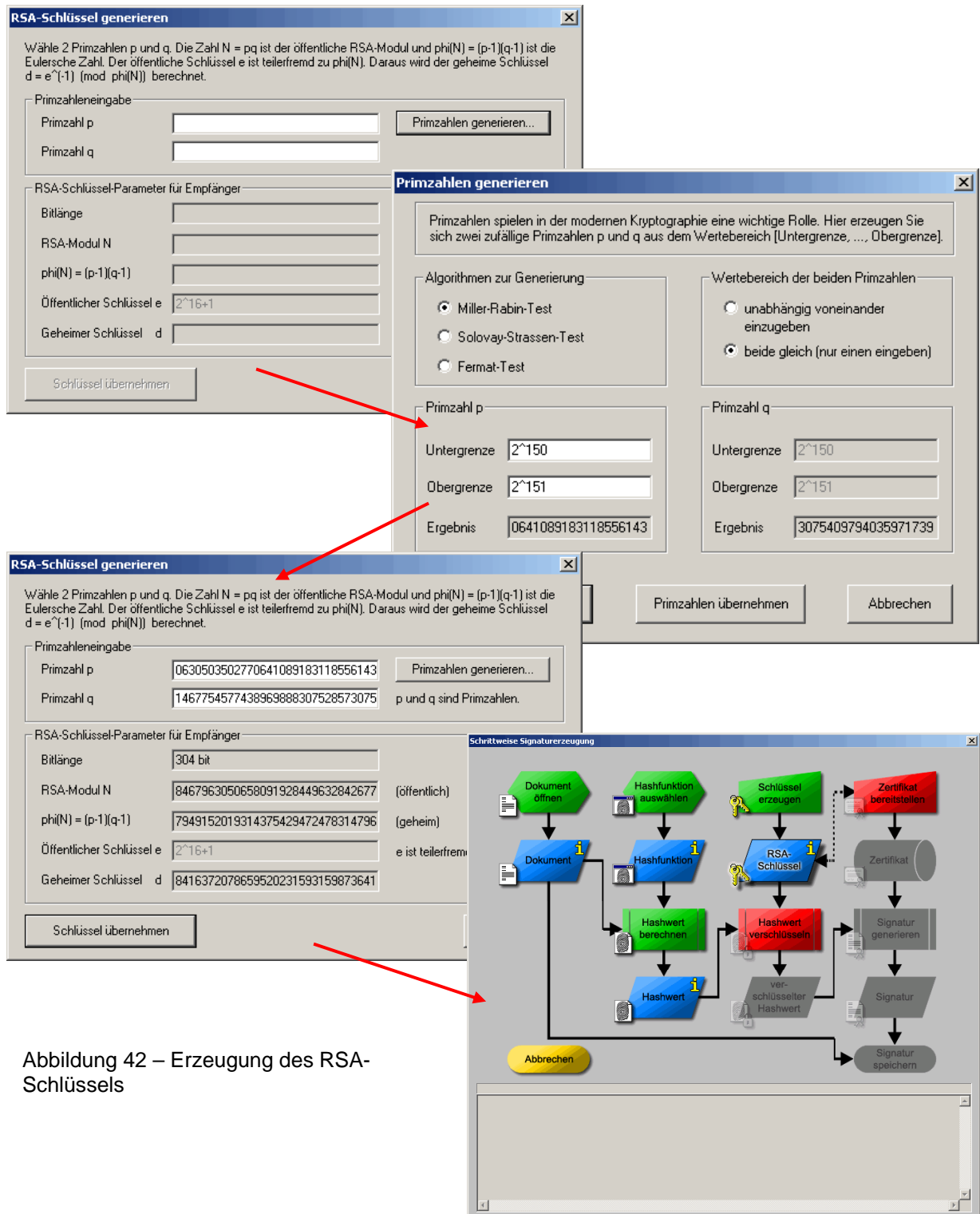


Abbildung 42 – Erzeugung des RSA-Schlüssels



Das Programm liefert uns folgendes Ergebnis:

Primzahl p: 1622110048067680630503502770641089183118556143  
Primzahl q: 1467754577438969888307528573075409794035971739

RSA Modul N (öffentlich):

23808594481610857178185515773860700839704070116705083656529084679  
63050658091928449632842677

RSA Modul (geheim):

23808594481610857178185515773860700839704070085806437401462579491  
52019314375429472478314796

geheimer Schlüssel d:

13676218744445463315836427160577382910592480926045884651654184163  
72078659520231593159873641

Im fünften Schritt verschlüsseln wir mit dem vorher berechneten RSA-Schlüssel unseren davor berechneten Hash-Wert (siehe Abbildung 43).

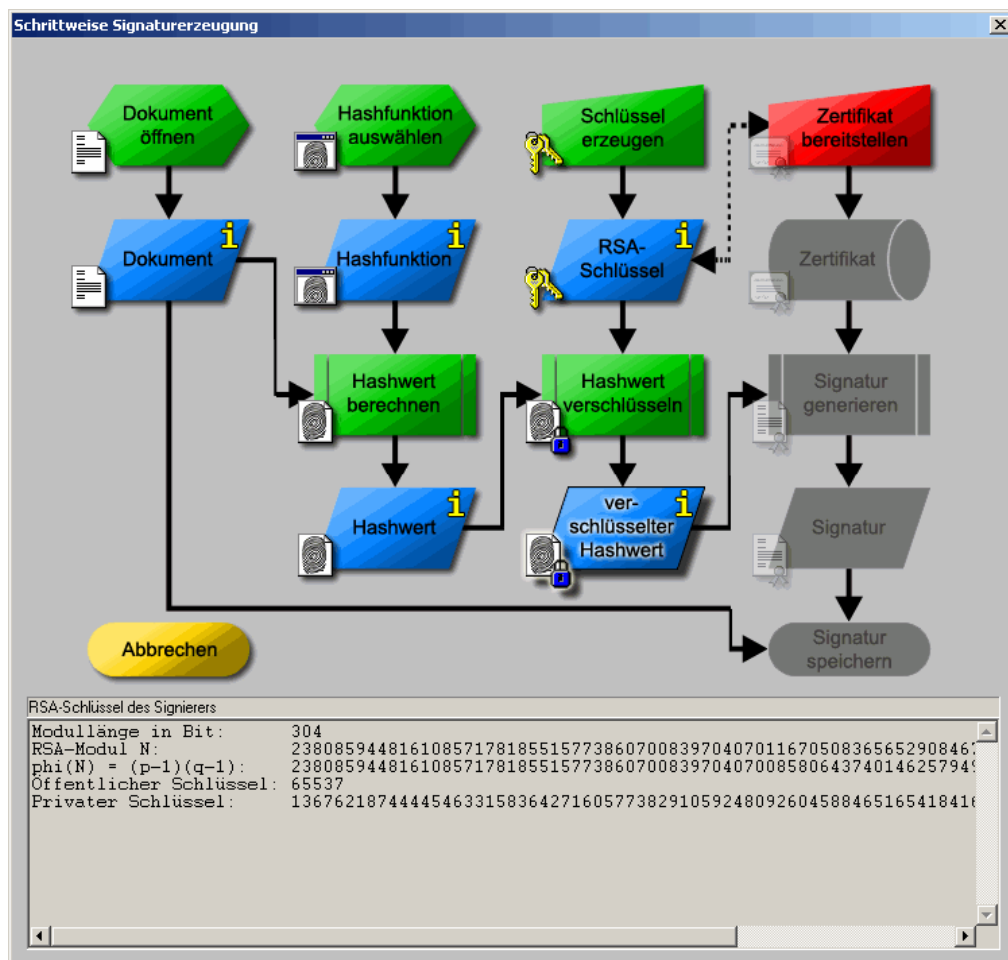


Abbildung 43 – Verschlüsseln des Hash-Werts



RSA Schlüssel:

Modullänge in Bit: 304

RSA-Modul N:

23808594481610857178185515773860700839704070116705083656529084679  
63050658091928449632842677

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

23808594481610857178185515773860700839704070085806437401462579491  
52019314375429472478314796

Öffentlicher Schlüssel: 65537

Privater Schlüssel:

13676218744445463315836427160577382910592480926045884651654184163  
72078659520231593159873641

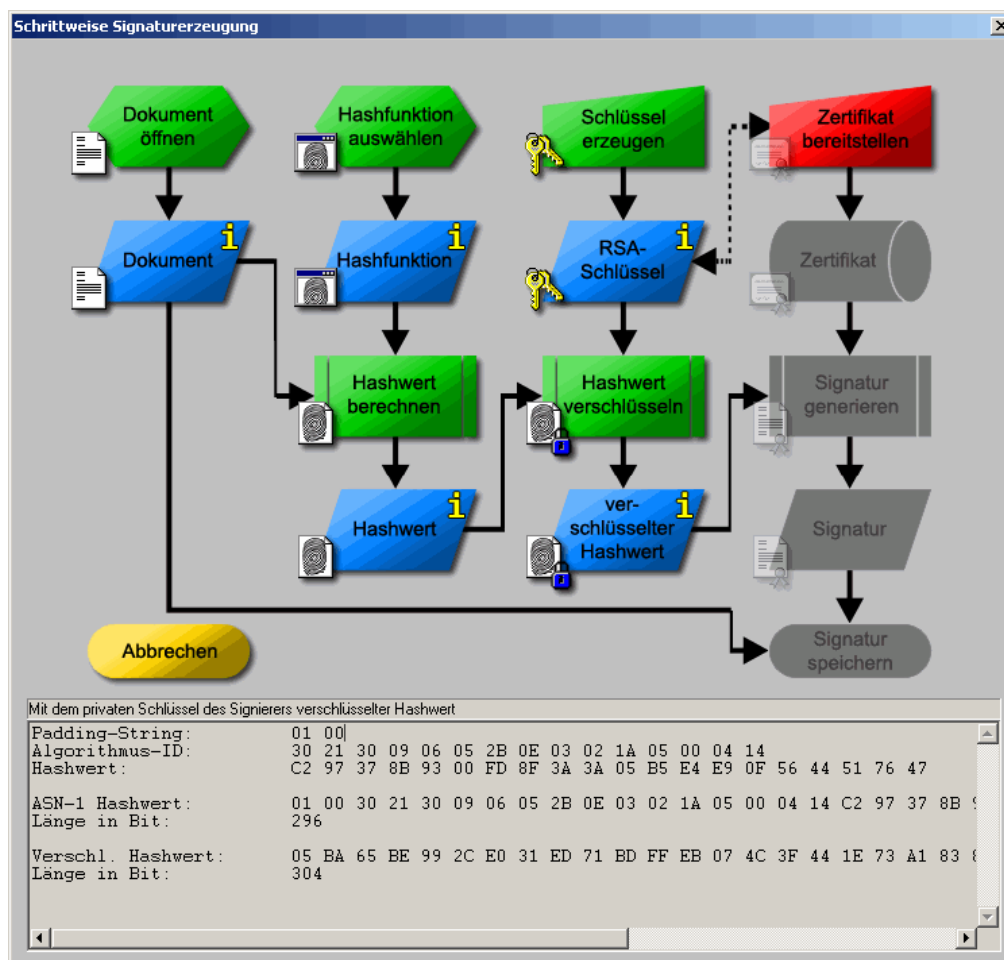


Abbildung 44 – Verschlüsselter Hash-Wert



Dies bringt uns zum verschlüsselten Hash-Wert (siehe Abbildung 44).

Padding-String: 01 00  
Algorithmus-ID: 30 21 30 09 06 05 2B 0E 03 02 1A 05 00 04 14  
Hashwert: C2 97 37 8B 93 00 FD 8F 3A 3A 05 B5 E4 E9 0F 56 44 51 76  
47

ASN-1 Hashwert: 01 00 30 21 30 09 06 05 2B 0E 03 02 1A 05 00 04 14 C2 97  
37 8B 93 00 FD 8F 3A 3A 05 B5 E4 E9 0F 56 44 51 76 47  
Länge in Bit: 296

Verschl. Hashwert: 05 BA 65 BE 99 2C E0 31 ED 71 BD FF EB 07 4C 3F 44 1E  
73 A1 83 80 73 42 F3 3E 16 02 CD 38 16 97 4F F3 17 57 FF 1A  
Länge in Bit: 304

Im sechsten und vorletzten Schritt müssen wir noch das Zertifikat erzeugen.  
Unter Eingabe von unserem Namen (E5), Vornamen (Gruppe) und einem  
beliebigen PIN-Code (4711) gelangen wir zum letzten Schritt (siehe Abbildung 45).

**Zertifikat und PSE erzeugen**

Öffentliche RSA-Parameter

Bitlänge: 304 bit

RSA-Modul N: 23808594481610857178185515773860700839704070116705083656529

Öffentlicher Schlüssel e: 65537

Persönliche Daten für das Zertifikat

Name: E5

Vorname: Gruppe

Schlüsselbezeichner: (optional)

PIN-Code: 4711

PIN-Verifikation: 4711

Generierte Namen für PSE und Zertifikat

User Key ID: [E5][Gruppe][RSA-304][1118320324]

Distinguished Name: [CN=Gruppe E5 [1118320324], L=Frankfurt, C]

Zertifikat und PSE erzeugen    Zertifikat / Schlüssel importieren

Abbildung 45 – Eingabe der Daten zur Erzeugung des Zertifikats

Im siebten Schritt lassen wir durch einen Klick die Signatur generieren und damit das Dokument signieren (siehe Abbildung 46). Sobald dies geschehen ist, bekommen wir eine Erfolgsmeldung und die fertige Signatur von CrypTool geliefert (siehe Abbildung 48).

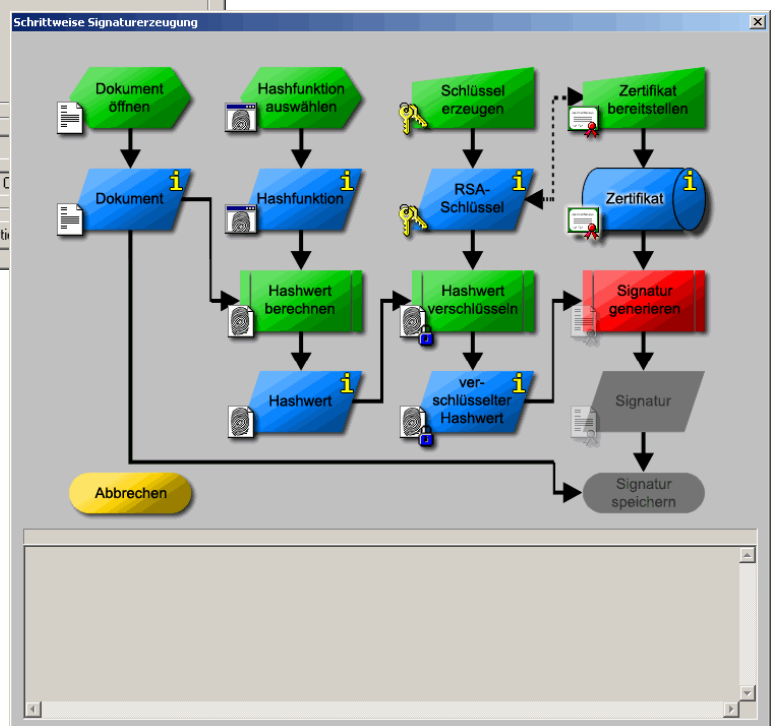


Abbildung 46 – Erzeugung der Signatur





Signatur generieren liefert folgendes Ergebnis:

```
00000 53 69 67 6E 61 74 75 72 3A 20 20 20 20 20 20 05 BA Signatur: .°
00012 65 BE 99 2C E0 31 ED 71 BD FF EB 07 4C 3F 44 1E 73 A1 e%.,à1íq½ÿë.L?D.sj
00024 83 80 73 42 F3 3E 16 02 CD 38 16 97 4F F3 17 57 FF 1A ..sBó>..Í8..Oó.Wÿ.
00036 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 53 69 Si
00048 67 6E 61 74 75 72 6C E4 6E 67 65 3A 20 20 33 30 34 20 gnaturlänge: 304
0005A 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 56 65 72 Ver
0006C 66 61 68 72 65 6E 3A 20 20 20 20 20 20 52 53 41 20 20 fahren: RSA
0007E 20 20 20 20 20 20 20 20 20 20 20 20 20 20 48 61 73 68 Hash
00090 66 75 6E 6B 74 69 6F 6E 3A 20 20 20 53 48 41 2D 31 20 funktion: SHA-1
000A2 20 20 20 20 20 20 20 20 20 20 20 20 20 20 53 63 68 Sch
000B4 6C FC 73 73 65 6C 3A 20 20 20 20 20 20 5B 45 35 5D 5B lüssel: [E5][
000C6 47 72 75 70 70 65 5D 5B 52 53 41 2D 33 30 34 5D 5B 31 Gruppe][RSA-304][1
000D8 31 31 38 33 32 30 33 32 34 5D 20 20 20 20 20 20 20 20 118320324]
000EA 20 20 20 20 20 20 20 20 4E 61 63 68 72 69 63 68 74 3A Nachricht:
000FC 20 20 20 20 20 20
```

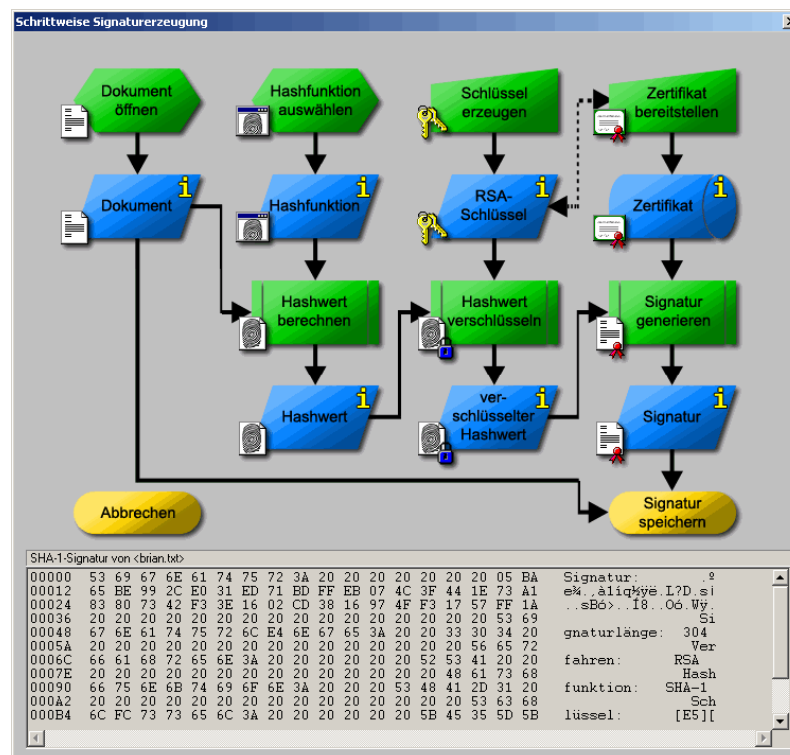


Abbildung 47 – Die Signatur wurde erzeugt

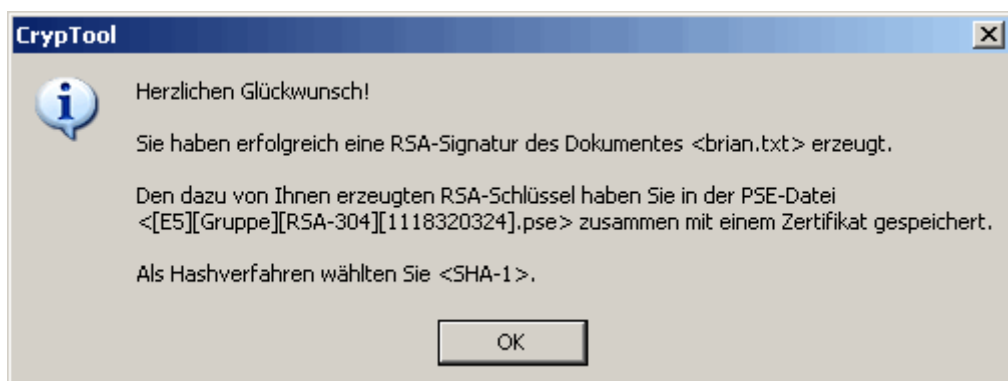
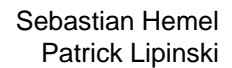


Abbildung 48 – Erfolgsmeldung nach Beendigung der Signaturerzeugung



[illegible]

The screenshot shows the CryptTool 1.3.05 interface. The main window is titled "RSA-Signatur mit SHA-1 von <brian.txt>". It displays a hex dump of the signed data. The "Verifizieren einer Signatur" (Verify a Signature) window is open, showing a list of available signers (Mustermann Max) and the verification settings (RSA-Schlüssel, SHA-1 Hashfunktion). The "Angegebene Daten" (Specified Data) section shows "Signatur-Verfahren: RSA" and "Hashfunktion: SHA-1". The "Aufgelistete Schlüsseltypen" (Listed Key Types) section shows "RSA-Schlüssel", "DSA-Schlüssel", and "EC-Schlüssel". The "Verifikation mit Verfahren:" (Verification with Method:) section shows "ECDSA" and "ECDSA-R". The "Verifikation mit Hashfunktion:" (Verification with Hash Function:) section shows "SHA-1" and "RIPEMD-160". The "Repräsentation der EC-Punkte in:" (Representation of EC Points in:) section shows "Affine Koord." and "Projektive Koord.". The "Suche Schlüssel" (Search Key) button is visible. The "Abbrechen" (Cancel) button is visible.

Seite 33



Wie uns das Programm uns mitteilt, ist die Signatur korrekt (siehe Abbildung 52).

Abbildung 52 – Erfolgsmeldung nach Überprüfung der Signatur

**b) Ist ein Faktorisierungs-Angriff sinnvoll? Begründung? Wenn ja, bitte durchführen.**

Durch den sehr grossen Primzahlenbereich von Minimum  $2^{150}$ , welchen wir für diesen Versuch verwendet haben, ist ein Faktorisierungsangriff nicht sinnvoll. Es würde schlichtweg eine zu grosse Zeit in Anspruch nehmen aufgrund der sehr vielen Kombinationsmöglichkeiten.

**7. Greifen Sie den in Anhang 2 gegebenen Text (Hinweis: Vigenère, englisch, Analyse > allgemein > N-Gramm) an. Berechnen Sie die Schlüssellänge mit der Friedman Formel und beschreiben Sie, wie man vorgeht, wenn man nur das N-Gramm als Hilfe hat. (Tipp: Häufigkeitsverteilung der Buchstaben – englisch)**

Der Text wird mit dem CrypTool geöffnet und der Text wird „angegriffen“. Das „Angreifen“ wird über die Registerkarte „Analyse > Allgemein > N-Gramm“ durchgeführt (siehe Abbildung 53).

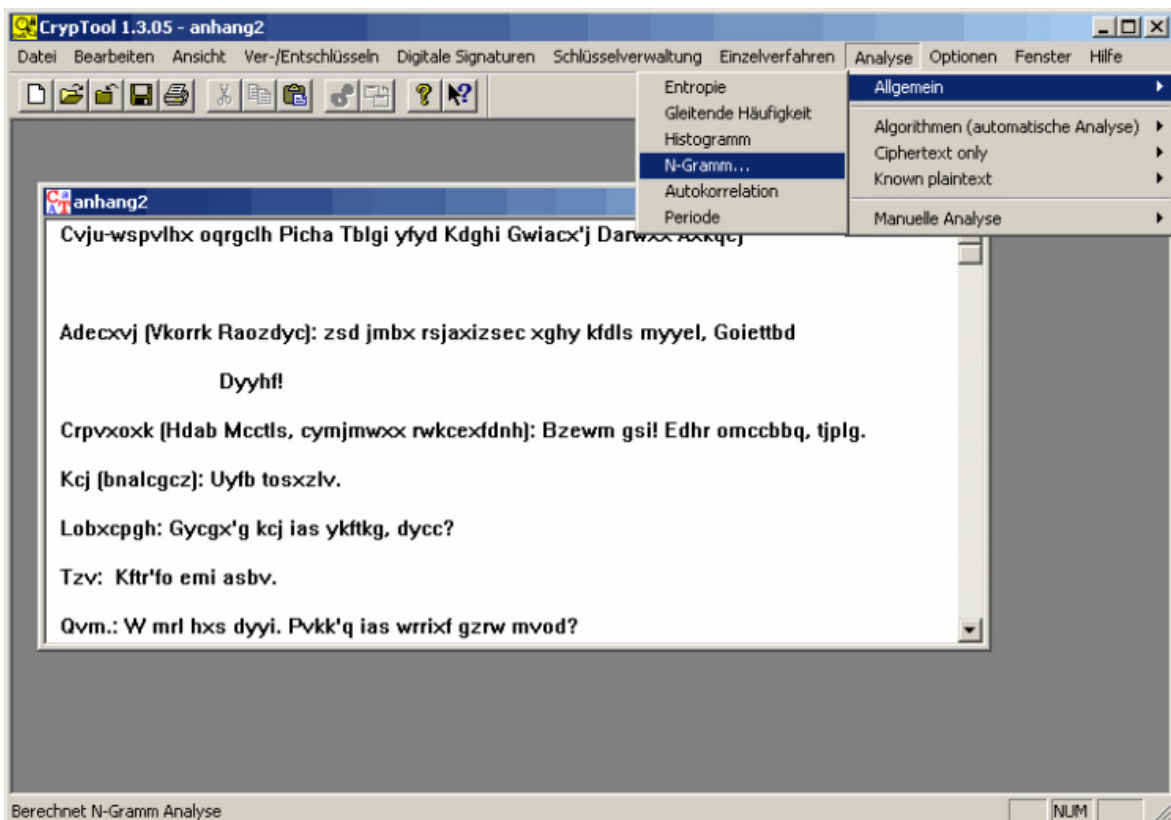


Abbildung 53 – Text angreifen



Anschließend erhalten wir eine Tabelle mit der Anzahl der Häufigkeiten, welche wir für die Berechnung in Excel importieren (siehe Abbildung 54).

Nr.	Zeichen...	Häufigkeit in %	Häufigkeit
1	C	6.6421	306
2	G	5.2312	241
3	R	4.9273	227
4	X	4.9273	227
5	D	4.8839	225
6	K	4.6234	213
7	B	4.6017	212
8	H	4.5366	209
9	V	4.4932	207
10	Y	4.1459	191
11	W	3.9722	183
12	S	3.9505	182
13	M	3.9071	180
14	I	3.8420	177
15	T	3.6900	170
16	F	3.6032	166
17	L	3.5598	164
18	P	3.2125	148
19	Z	3.1691	146
20	O	3.1474	145
21	E	2.7784	128
22	J	2.7784	128
23	A	2.7567	127
24	N	2.4311	112

Abbildung 54 – Anzahl der Häufigkeiten

Für die Berechnung benutzen wir die Friedman Formel:

$$l = \frac{(I_e - I_r) \cdot m}{(m-1) \cdot I - I_r \cdot m + I_e} \quad I = \frac{\sum_{i=1}^{26} n_i(n_i - 1)}{n(n-1)}$$

mit  $l$  = Länge des Schlüsselwortes,  
 $I_e$  = Koinzidenzindex für englisch (= 0,0661)  
 $I_r$  = Normalverteilungskoeffizient (= 0,0385)  
 $I$  = Koinzidenzindex des verschlüsselten Textes  
 $m$  = Menge der gültigen Buchstaben  
 $n_i$  = Häufigkeit des Buchstaben

Die benötigten Häufigkeiten entnehmen wir der N-Gramm Statistik.

Ansicht der Excel Tabelle, wobei in unserer Berechnung wohl ein Fehler ist. Leider können wir ihn nicht finden, deswegen anbei die komplette Tabelle mit Formeln:



	A	B	C	D	E	F	G	H
3		Nr.	Teilstring	Häufigkeit in %	Häufigkeit	$mi^*(mi-1)$		
4		1	C	6,6421	306	93330		
5		2	G	5,2312	241	57840		
6		3	R	4,9273	227	51302		
7		4	X	4,9273	227	51302		
8		5	D	4,8839	225	50400		
9		6	K	4,6234	213	45156		
10		7	B	4,6017	212	44732		
11		8	H	4,5366	209	43472		
12		9	V	4,4932	207	42642		
13		10	Y	4,1459	191	36290		
14		11	W	3,9722	183	33306		
15		12	S	3,9505	182	32942		
16		13	M	3,9071	180	32220		
17		14	I	3,8420	177	31152		
18		15	T	3,6900	170	28730		
19		16	F	3,6032	166	27390		
20		17	L	3,5598	164	26732		
21		18	P	3,2125	148	21756		
22		19	Z	3,1691	146	21170		
23		20	O	3,1474	145	20880		
24		21	E	2,7784	128	16256		
25		22	J	2,7784	128	16256		
26		23	A	2,7567	127	16002		
27		24	N	2,4311	112	12432		
28		25	U	2,1489	99	9702		
29		26	Q	2,0404	94	8742		
30								
31			Summen:	100 %	4607	872134		
32								
33			$l/(m^*(m-1))$ :			0,041099929		
34								
35			Schlüsselwort:			10,5935636689198000000		
36								
37			Formel Schlüsselwort :	$((0,0661-0,0385)*4607)/((4607-1)*F33-0,0385*4607+0,0661)$				
38								
39			Formel $l/(m^*(m-1))$ :	$F31/(4607*(4607-1))$				
40								

Die korrekte Lösung wäre eine Länge von 6 Zeichen.

Wenn wir bei der Rechnung auf das richtige Ergebnis gekommen wären, wäre der nächste Schritt, alle Buchstaben des verschlüsselten Textes in 6er Blöcken untereinander aufzuschreiben und durch versuchen probiert den Text zu entschlüsseln. Wobei man versucht hätte den im verschlüsselten Text am häufigsten vorkommenden Buchstaben mit den in der Sprache häufigsten Buchstaben zu ersetzen.

Durch Rumprobieren wäre man dann auf das Passwort „Krypto“ gekommen, welches wir aus der automatischen Vigenère Entschlüsselung des CrypTools entnommen haben.