

```
1: ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
2: //XorCryptx - Programm zur Ver- und Entschlüsselung von Texten und Worten //
3: //Copyright © 2007 Benedikt Welp //
4: // //
5: //Projekt1 zur Besonderen Lernleistung im Fachbereich Mathematik über das //
6: //Thema Kryptologie - RSA-Verfahren vorgelegt von Benedikt Welp //
7: ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
8: ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
9: // Datei : Unit1.pas ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
10: // Erstellt : 11.02.2007 ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
11: // Copyright: © 2007 Benedikt Welp ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
12: // eMail : benedikt.welp@web.de ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
13: // Version : 2.4.1.XX ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
14: // ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
15: // Das Programm wurde unter Windows XP mit dem Borland Developeer Studio 2006 //
16: // Professional (SSL-Lizenz) entwickelt und getestet. //
17: // ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
18: ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
19: ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
20: //Abkürzungen: ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
21: // Btxxxx - Button ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
22: // Lxxxxx - Label ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
23: // Mxxxxx - Memo ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
24: // Rbxxxx - RadioButton ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
25: // Exxxxx - Editfeld ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
26: ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
27: ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
28:
29:
30: unit Unit1;
31:
32: interface
33:
34: uses
35: Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
36: Dialogs, StdCtrls, ComCtrls, Menus, ExtCtrls, Gauges;
37:
38: type
39: TForm1 = class(TForm)
40: MainMenu1: TMainMenu; //MainMenu Leiste.
41: Dateil: TMenuItem;
42: Exit1: TMenuItem;
43: Infol: TMenuItem;
44: Info2: TMenuItem;
45: OpenDialog1: TOpenDialog; //OpenDialog für Wortlisten Auswahl.
46: ffneWordlistel: TMenuItem; //Aufruf des OpenDialog1.
47: GroupBox1: TGroupBox;
48: GroupBox2: TGroupBox;
49: GroupBox3: TGroupBox;
50: Label17: TLabel;
51: LWordlistStatus: TLabel; //Status der Wortliste.
52: BtLadeStdWordlist: TButton; //Laden der standard Wortliste.
53: BtClearWordlist: TButton; //Löschen der Wortliste aus der TStringlist.
54: MKlartext: TMemo; //Memo für die Klartexteingabe.
55: ESchluessel: TEdit; //Editfeld für den Schlüssel (0-255).
56: BtEncrypt: TButton; //verschlüsseln.
57: BtDecrypt: TButton; //entschlüsseln.
58: ComboBoxEx1: TComboBoxEx;
59: Label14: TLabel;
60: LEncrypt1: TLabel; // immer Caption := "X".
61: Encrypt: TLabel; // immer Caption := "Encrypt".
62: Decrypt: TLabel; // immer Caption := "Decrypt".
63: Label1: TLabel;
64: Label9: TLabel;
65: Label4: TLabel;
66: Label7: TLabel;
67: LZeit: TLabel; // Anzeige der benötigten Zeit (nur Decrypt).
68: LEncrypt2: TLabel; // Tabelle: benötigte Iterationen für Encrypt.
69: LDecrypt1: TLabel; // Tabelle: benötigte Versuche für Decrypt.
70: LDecrypt2: TLabel; // Tabelle: benötigte Iterationen für Decrypt.
71: GroupBox4: TGroupBox;
```

```
72:     Label5: TLabel;
73:     Label6: TLabel;
74:     MASCI: TMemo;           // Memo Konsole (ASCII-Zeichensatz).
75:     MHEX: TMemo;           // Memo Konsole (Hexadezimal).
76:     Label3: TLabel;
77:     LDecryptFortschritt: TLabel; // Label über der Fortschrittsanzeige TGauge1.
78:     Ansicht1: TMenuItem;
79:     bersicht1: TMenuItem;    // Übersichtsfenster aufrufen/verstecken.
80:     Gauge1: TGauge;         // Fortschrittsanzeige.
81:     RbSchnelltest: TRadioButton; // Schnelltestverfahren für Decrypt.
82:     RbSinW: TRadioButton;    // weiteres Verfahren für Decrypt.
83:     RbWinS: TRadioButton;    // weiteres Verfahren für Decrypt.
84:     procedure bersicht1Click(Sender: TObject);
85:     procedure FormClose(Sender: TObject; var Action: TCloseAction);
86:     procedure ffneWordlistClick(Sender: TObject);
87:     procedure BtClearWordlistClick(Sender: TObject);
88:     procedure BtLadeStdWordlistClick(Sender: TObject);
89:     procedure Exit1Click(Sender: TObject);
90:     procedure Info2Click(Sender: TObject);
91:     procedure BtDecryptClick(Sender: TObject);
92:     procedure BtEncryptClick(Sender: TObject);
93:     procedure FormCreate(Sender: TObject);
94:
95:
96: private
97:     { Private-Deklarationen }
98: public
99:     x: String;              // ÜbergabevARIABLE; Verschlüsselung -> Entschlüsselung.
100:    w: TStringList;         // Speicherort der Wortliste.
101:    Function getTime:String; // Funktion zum Auslesen der Zeit.
102:    Function StrToHex(s: String): String; //String nach Hexadezimal wandeln.
103: { Public-Deklarationen }
104: end;
105:
106: var
107:     Form1: TForm1;
108:
109: implementation
110:
111: uses Unit2; //Übersichtsfenster einbinden.
112:
113: {$R *.dfm}
114:
115:
116: //=====Functions=====
117: Function TForm1.StrToHex(s: String): String;
118: var i: Integer;
119: begin
120:     result:='';
121:     if length(s) > 0 then
122:         for i := 1 to length(s) do
123:             result := result + IntToHex(Ord(s[i]),2);
124: end;
125:
126: Function TForm1.GetTime: String;
127: var a: TDateTime;
128: begin
129:     a := Time(); // Systemzeit auslesen.
130:     result := FormatDateTime('hh:nn:ss:zzz',a); // Zeit formatieren und ausgeben.
131: end;
132:
133: Function GetFileVersion(Path: string): string; // vorgefertigte Funktion
134: var // zum Auslesen der Programm-
135:     lpVerInfo: pointer; // version.
136:     rVerValue: PVSFixedFileInfo;
137:     dwInfoSize: cardinal;
138:     dwValueSize: cardinal;
139:     dwDummy: cardinal;
140:     lpstrPath: pchar;
141: begin
142:     if Trim(Path) = EmptyStr then
```

```

143:     lpstrPath := pchar(ParamStr(0))
144: else lpstrPath := pchar(Path);
145: dwInfoSize := GetFileVersionInfoSize(lpstrPath, dwDummy);
146:     if dwInfoSize = 0 then
147:     begin
148:         Result := 'No version specification';
149:         Exit;
150:     end;
151: GetMem(lpVerInfo, dwInfoSize);
152: GetFileVersionInfo(lpstrPath, 0, dwInfoSize, lpVerInfo);
153: VerQueryValue(lpVerInfo, '\\', pointer(rVerValue), dwValueSize);
154:
155: with rVerValue^ do
156: begin
157:     Result := IntToStr(dwFileVersionMS shr 16);
158:     Result := Result + '.' + IntToStr(dwFileVersionMS and $FFFF);
159:     Result := Result + '.' + IntToStr(dwFileVersionLS shr 16);
160:     Result := Result + '.' + IntToStr(dwFileVersionLS and $FFFF);
161: end;
162: FreeMem(lpVerInfo, dwInfoSize);
163: end;
164:
165: //=====
166: //=====Hauptprogramm=====
167: //=====
168:
169: //=====Verschlüsselung=====
170: Procedure TForm1.BtEncryptClick(Sender: TObject);
171: var s: String; // Lokale Variablen: s - String der verschlüsselt werden soll.
172:     i: Integer; // i - Zählvariable für Schleife.
173:     key: Integer; // Schlüssel.
174: begin
175:     Gauge1.Progress := 0;
176:     s := MKLartext.text;
177:
178:     if MKLartext.text = '' then
179:         exit
180:     else if (length(s) > 0) then
181:     begin
182:         Form2.Label15.Caption := s;
183:         Form2.Label6.Caption := ESchluesel.Text;
184:         Form2.Image1.Visible := true;
185:         Form2.Image2.Visible := true;
186:         LDecrypt2.caption := '0';
187:         LDecrypt1.caption := '0';
188:         x := s;
189:         LEncrypt2.Caption := '0';
190:         key := StrToInt(ESchluesel.Text);
191:         for i := 1 to length(s) do
192:         begin
193:             LEncrypt2.Caption := inttostr(strtoint(LEncrypt2.caption) + 1);
194:             s[i] := char(key xor Ord(s[i]));
195:         end;
196:         MASCI.Text := '';
197:         MHEX.Text := '';
198:         MASCI.Lines.Add('<' + gettime + '> ' + 'Klartext: ' + x);
199:         MHEX.Lines.Add('<' + gettime + '> ' + 'Klartext: ' + strtohex(x));
200:         MASCI.Lines.Add('<' + gettime + '> ' + 'Verschlüsselter Text: ' + s);
201:         MHEX.Lines.Add('<' + gettime + '> ' + 'Verschlüsselter Text: ' + strtohex(s));
202:         LDecryptFortschritt.Caption := 'Klartext verschlüsselt.';
203:         Form2.Label8.Caption := s;
204:         x := s; // Übergabe des Endprodukts.
205:     end
206: end;
207:
208: //=====Entschlüsselung=====
209: Procedure TForm1.BtDecryptClick(Sender: TObject);
210: var i, j, l, m: Integer; // Lokale Zählvariablen für Schleifen.
211:     s, t: String;
212:     d: Integer; // Zählvariable für Stringlist.
213:     g, h: Boolean; // g: Fund (ja/nein); h: Teilfund (ja/nein).

```

```
214:     q, p: Extended;           // Gleitkommavariable für die benötigte Zeit.
215:     zaehl: Integer;           // Zählvariable: benötigten Schleifendurchläufe.
216: begin
217:     Gauge1.Progress := 0;
218:     zaehl := 0;
219:     g := false;
220:     h := false;
221:     q := GetTickCount;
222:     p := 0;
223:     if (ComboBoxEx1.Text = 'Brute-Force') and (BtLadeStdWordlist.Enabled = false) and (length(x)
> 0) then //ComboBoxEx und Wortliste prüfen.
224:     begin
225:         LDecryptFortschritt.Caption := 'Processing...';
226:         LDecryptFortschritt.Update; // da Labels erst am Ende der Hauptprozedur
227:                                     // (hier: "BtDecryptClick()")
228:                                     // mit neuem Inhalt beschrieben werden, muss
229:                                     // bei Bedarf "Labelx.Update;" eingefügt
230:                                     // werden, um ein Label in Echtzeit zu
231:                                     // beschreiben.
232:
233:         Form2.Image3.Visible := true; // Einstellungen im Übersichtsfenster.
234:         Form2.Image4.Visible := true;
235:         Form2.Label10.Visible := true;
236:         Form2.Label9.Visible := false;
237:         LDecrypt1.caption := '0';
238:         LDecrypt2.caption := '0';
239:         if (length(x) > 0) then // s ungleich 0 Zeichen.
240:         begin
241:             zaehl := strtoint(LDecrypt2.Caption);
242:             for i := 0 to 255 do // Testen aller möglichen Schlüssel (0-255).
243:             begin
244:                 inc(zaehl); // Erhöhen der internen Zählvariable für LDecrypt2.
245:                 s := x; // Übergebenen String in Lokale Variable übergeben.
246:                 LDecrypt1.caption := inttostr(i); // Anzeige für die Anzahl getesteter
247:                                                         // Schlüssel.
248:
249:                 for l := 1 to length(s) do //Entschlüsseln mit aktuellem Key-i.
250:                 begin
251:                     s[l] := char(i Xor ord(s[l]));
252:                     inc(zaehl);
253:                 end;
254:
255:                 // In Konsole ausgeben, welche Schlüsselmöglichkeit getestet wird.
256:                 MASCI.Lines.Add('<' + gettime + '> ' + 'Try(' + inttostr(i) + '): ' + s);
257:                 MHEX.Lines.Add('<' + gettime + '> ' + 'Try(' + inttostr(i) + '): ' + strophe(s));
258:
259:                 // Drei Methoden, um Entschlüsselungsprodukt auf Fund zu überprüfen.
260:                 if RbSchnelltest.Checked = true then //Standard Methode.
261:                 begin
262:                     for d := 0 to w.count-1 do // Wort 0 - letztes Wort.
263:                     begin
264:                         if s = w.Strings[d] then // Wort d mit s Vergleichen.
265:                         begin
266:                             inc(zaehl);
267:                             g := true; // Booleanvariable für Fund -> break;.
268:                             p := GetTickCount; //Zeitnahme.
269:                         end;
270:                     end;
271:                 end;
272:
273:                 if RbSinW.Checked = true then //Methode: Wort in der Wordlist suchen.
274:                 begin
275:                     if w.IndexOf(s) > 0 then
276:                     begin
277:                         g := true;
278:                         p := GetTickCount;
279:                     end
280:                     else
281:                     for j := length(s)-1 downto 3 do
282:                     begin
283:                         inc(zaehl);
```

```
284:         t := Copy(s, 1, j);
285:         if w.IndexOf(t) > 0 then
286:         begin
287:             h := true;
288:             p := GetTickCount;
289:         end;
290:     end;
291: end;
292:
293: if RbWinS.Checked = true then // Methode: Wort der Wortliste in
294: begin // entschlüsseltem Wort suchen.
295:     if w.IndexOf(s) > 0 then
296:     begin
297:         g := true;
298:         p := GetTickCount;
299:     end
300:     else
301:     begin
302:         for m := 0 to w.Count - 1 do
303:         begin
304:             inc(zaehl);
305:             if AnsiPos(w[m],s) > 0 then
306:             begin
307:                 h := true;
308:                 p := GetTickCount;
309:             end;
310:         end;
311:     end;
312: end;
313:
314: //Überprüfung auf Funde.
315: if g = true then //Fund.
316: begin
317:     MASCI.Lines.Add('<' + gettime + '> ' + 'Final: ' + s + #13#10);
318:     MHEX.Lines.Add('<' + gettime + '> ' + 'Final: ' + strtohex(s) + #13#10);
319:     LDecryptFortschritt.Caption := 'Entschlüsselung erfolgreich.';
320:     Form2.Label14.Caption := 'Entschlüsselung mit Brute-Force erfolgreich';
321:     Form2.Label12.Caption := s;
322:     LZeit.caption := floattostr((p-q)/1000) + ' Sekunden.'; //benötigte Zeit errechnen.
323:     break;
324: end
325: else if h = true then //Teilfund.
326: begin
327:     MASCI.Lines.Add('<' + gettime + '> ' + 'Teilfund in: ' + s + #13#10);
328:     MHEX.Lines.Add('<' + gettime + '> ' + 'Teilfund in: ' + strtohex(s) + #13#10);
329:     LDecryptFortschritt.Caption := 'Entschlüsselung erfolgreich. Teilfund !';
330:     Form2.Label14.Caption := 'Entschlüsselung mit Brute-Force erfolgreich. Teilfund !';
331:     Form2.Label12.Caption := s;
332:     LZeit.caption := floattostr((p-q)/1000) + ' Sekunden.'; //benötigte Zeit errechnen.
333:     break;
334: end;
335: Gaugel.Progress := Gaugel.Progress + 1;
336: end;
337: if (i = 256) and (g = false) then //Fehlschlag.
338: begin
339:     p := GetTickCount;
340:     MASCI.Lines.Add('<' + gettime + '> ' + 'Entschlüsselung fehlgeschlagen. Halte bei: ' +
s);
341:     MHEX.Lines.Add('<' + gettime + '> ' + 'Entschlüsselung fehlgeschlagen. Halte bei: ' + s
+ strtohex(s));
342:     LDecryptFortschritt.Caption := 'Entschlüsselung fehlgeschlagen.';
343:     Form2.Label14.Caption := 'Entschlüsselung mit Brute-Force fehlgeschlagen.';
344:     Form2.Label12.Caption := s;
345:     LZeit.caption := floattostr((p-q)/1000) + ' Sekunden.'; //benötigte Zeit errechnen.
346: end;
347: end;
348: end;
349:
350: //Entschlüsseln mit Schlüsseleingabe.
351: if ComboBoxEx1.Text = 'Schlüssel' then //Direkte Entschlüsselung.
352: begin
```

```
353: LDecrypt1.caption := '0';
354: LDecrypt2.caption := '0';
355: Form2.Image3.Visible := true;
356: Form2.Image4.Visible := true;
357: Form2.Label10.Visible := false;
358: Form2.Label9.Visible := true;
359: s := x;
360:   for l := 1 to length(s) do
361:     begin
362:       s[l] := char(strtoint(ESchluessel.Text) Xor ord(s[l]));
363:       inc(zaehl);
364:       LDecrypt2.caption := inttostr(zaehl);
365:     end;
366:   MASCI.Lines.Add('<' + gettime + '> ' + 'Mit Schlüssel entschlüsselter Text: ' + s);
367:   MHEX.Lines.Add('<' + gettime + '> ' + 'Mit Schlüssel entschlüsselter Text: ' + strtohex(s)
);
368: LDecryptFortschritt.Caption := 'Geheimtext entschlüsselt.';
369: Form2.Label12.Caption := s;
370: Form2.Label14.Caption := 'Geheimtext mit Schlüssel erfolgreich entschlüsselt.';
371: end;
372: LDecrypt2.caption := inttostr(zaehl);
373: end;
374:
375:
376: //=====
377: //=====Diverses=====
378: Procedure TForm1.FormCreate(Sender: TObject); //Initialisierung des Programms.
379: begin
380:   Form1.Caption := 'XorCryptx' + ' v.' + GetFileVersion(Application.Exename);
381:   Form1.Height := 739; //Maße bestimmen.
382:   Form1.Width := 688;
383:   RbSchnelltest.Checked := true;
384:   RbWinS.Checked := false;
385:   RbSinW.Checked := false;
386:   MASCI.Text := ''; //Inhalt Memofenster bei Anwendungsstart löschen.
387:   Mklartext.Text := '';
388:   MHEX.Text := '';
389:   LDecrypt1.Caption := '';
390: end;
391:
392: Procedure TForm1.Info2Click(Sender: TObject);
393: begin
394:   ShowMessage(' Xor Ver- und Entschlüssler' + #13#10 + '
(c) Benedikt Welp' + #13#10 + 'erstellt mit BORLAND® DELPHI® 2006 PROFESSIONAL ');
395: end;
396:
397: Procedure TForm1.Exit1Click(Sender: TObject);
398: begin
399:   Form1.close; //Anwendung beenden; Speicher der Stringlist
400: end; //wird automatisch in "OnClose" freigeräumt.
401:
402: Procedure TForm1.BtLadeStdWordlistClick(Sender: TObject);
403: begin
404:   w := TStringList.create; //Stringlist erzeugen.
405:   w.LoadFromFile('germanwordlist.txt'); //Wordlist aus Programmverzeichnis laden.
406:   LWordlistStatus.Caption := inttostr(w.count) + ' Worte Geladen!'; //angeben ob Liste geladen
.
407:   BtLadeStdWordlist.enabled := false; //neuladen verhindern.
408:   ffneWordlist1.Enabled := false;
409:   Gaugel.Progress := 0;
410:   Form2.Label2.Caption := inttostr(w.count) + ' Worte in germanwordlist.txt';
411: end;
412:
413: Procedure TForm1.BtClearWordlistClick(Sender: TObject);
414: begin
415:   BtLadeStdWordlist.enabled := true; //neuladen aktivieren.
416:   ffneWordlist1.Enabled := true;
417:   w.Free; //Speicher der alten Liste freigeben.
418:   w := nil; //w "entknüpfen".
419:   LWordlistStatus.Caption := ''; //angeben, dass nichts geladen ist.
420:   Gaugel.Progress := 0;
```

```
421:   Form2.Label2.Caption := '';
422: end;
423:
424: Procedure TForm1.ffneWordlist1Click(Sender: TObject);
425: begin
426:   if opendirlog1.execute then           //ermöglicht laden beliebiger Wordlists.
427:   begin
428:     if Assigned(w) then w.Free;
429:     w := TStringList.create;
430:     w.LoadFromFile(opendirlog1.FileName);
431:     LWordlistStatus.Caption := inttostr(w.count) + ' Worte Geladen!';
432:     BtLadeStdWordlist.enabled := false;           //neuladen verhindern.
433:     ffneWordlist1.Enabled := false;
434:     Gaugel.Progress := 0;
435:     Form2.Label2.Caption := inttostr(w.count) + ' Worte in ' + ''' + opendirlog1.FileName + '''
;
436:   end;
437: end;
438:
439: procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
440: begin
441:   if assigned(w) then w.free;
442: end;
443:
444: procedure TForm1.bersicht1Click(Sender: TObject);
445: begin
446:   if bersicht1.Checked = false then // Prüfen ob Übersichtsfenster aktiviert
447:   begin // ist, und es dann aktivieren oder
448:     Form2.Visible := true;           // deaktivieren.
449:     bersicht1.Checked := true;
450:   end
451:   else begin
452:     bersicht1.Checked := false;
453:     Form2.Visible := false;
454:   end;
455: end;
456:
457: end.
```