bmpPacker

User Manual

Author : Jens Gödeke
Program Version : 1.2

Email:
Security@goedeke.net

Homepage :
http://www.goedeke.net/bmppacker.html

Contents :

1 Disclaimer: 3

2 What and why bmpPacker 4

3 How to use bmpPacker 5

3.1 Encode/Decode Interactively 5

3.1.1 Encode a file 5

3.1.2 Decode a file 7

3.2 Encode/Decode automaticaly 9

3.3 Options 10

3.3.1 Startup Encryption Method 11

3.3.2 Automatic Conversion 11

3.3.3 Passwords & Keys 11

3.3.4 Use compressed file 11

3.3.5 Compression Level 12

3.3.6 Misc. Options 12

3.3.7 Target Filename is Source Filename with... 12

3.4 Configuration fileformat 13

3.5 Commandline options 13

4 Features 14

4.1 Encrypting Methods 14

4.1.1 BlowFish 14

4.1.2 TwoFish 14

4.1.3 Rijndael 14

4.1.4 NIST National Institute of Standards & Technology 15

4.2 Checksum Methods 15

4.3 Compression Methods 15

4.4 History 15

4.5 What is a „hex“ key 16

1 Disclaimer:

**bmpPacker uses strong cryptography,
so even if it is created, maintained and distributed from
liberal countries in Europe (where it is legal to do this),
it falls under certain export/import and/or use restrictions
in some other parts of the world.**

**Please remember that export/import and/or use of strong
cryptography software is illegal in some parts of the world.**

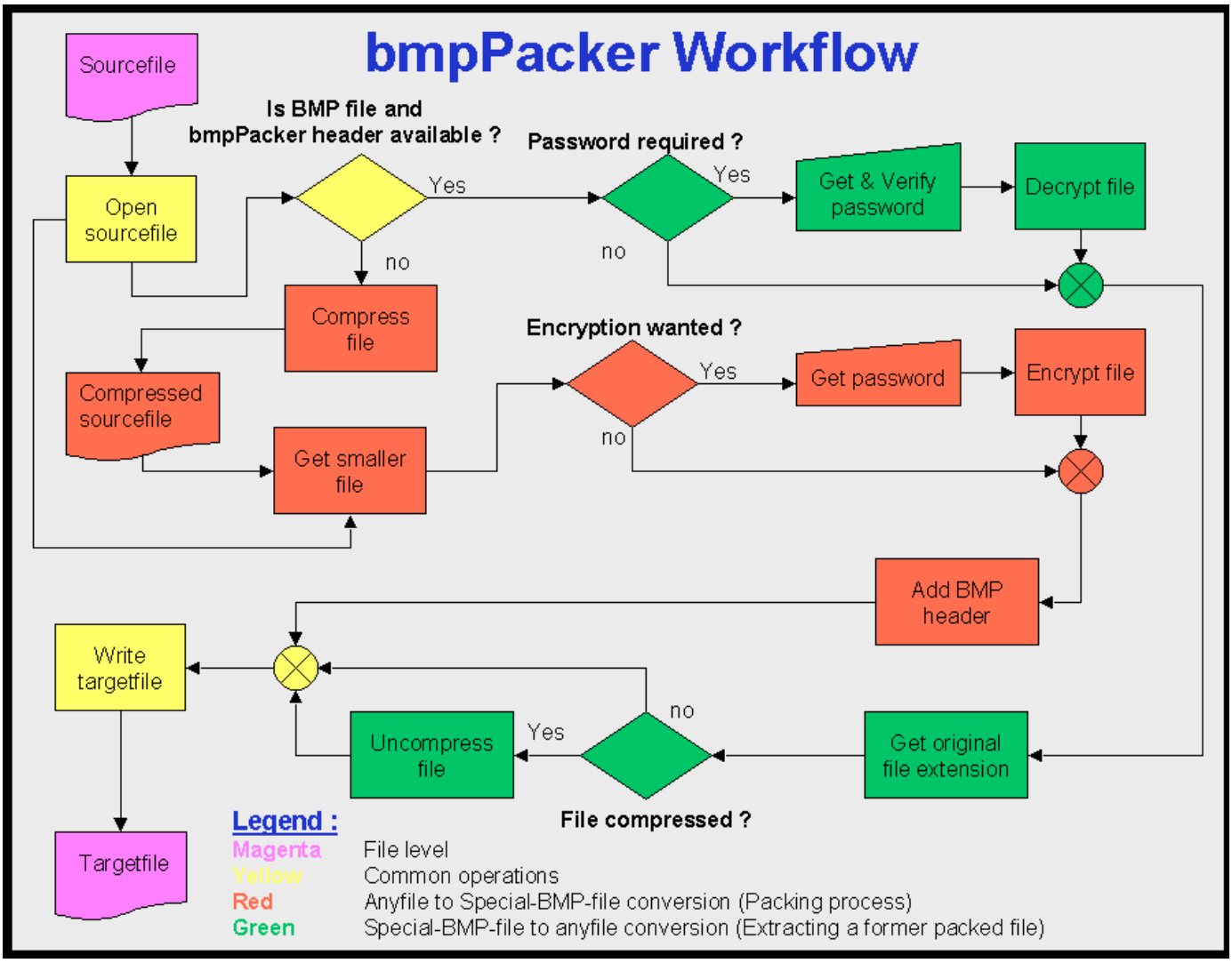
**So when you import bmpPacker to your country or re-distribute
it from there you are strongly advised to pay close attention to any
export/import and/or use laws that apply to you.**

**Further bmpPacker provides a kind of „backdoor“ by hiding
any kind of file in BMP-containers. You have to ensure that you
have the system adminstrators permission if you want to use
bmpPacker in your company or in any other non-private environment.**

**The author of bmpPacker is not liable for any illegal acts
you may perform by exporting/importing or using bmpPacker.**

2 What and why bmpPacker

bmpPacker is a small tool which takes a single file as input and returns a target file as a bitmap graphic with optional compression & encryption.

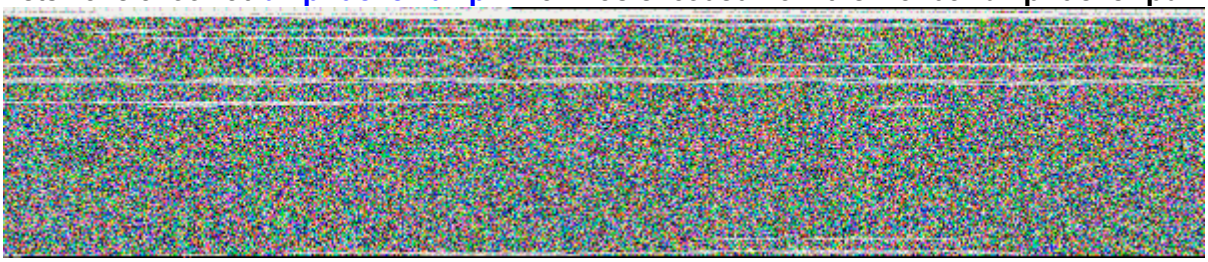


The resulting file is a bitmap file which is 100% compatible to the Windows BMP format.

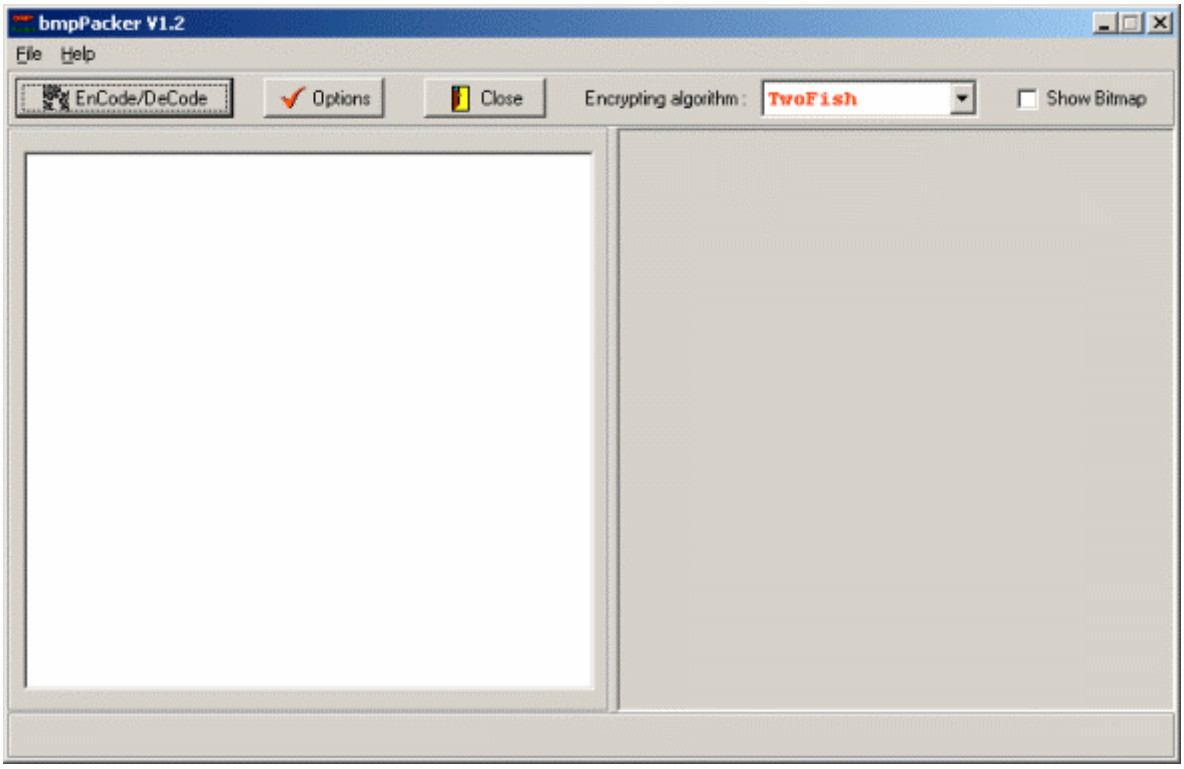
You may consider this like a zip file optionaly encrypted with a password
but the extension of the file is "**bmp**" and not "**zip**".

These resulting files are normaly viewable as Windows BMP files.
But the only chance to decode (& optional decrypt) these files is if you use **bmpPacker**.

A little example :
Lets have a look at **bmpPacker.bmp** which was encoded from the manual **bmpPacker.pdf**



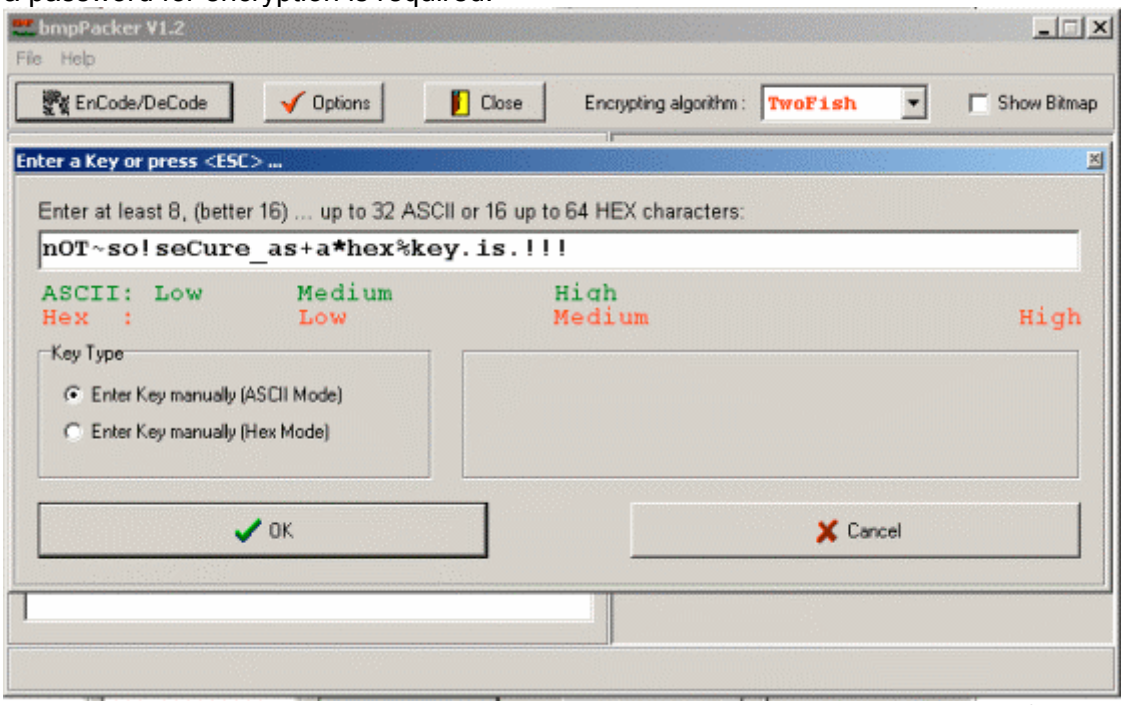
3 How to use bmpPacker



3.1 Encode/Decode Interactively

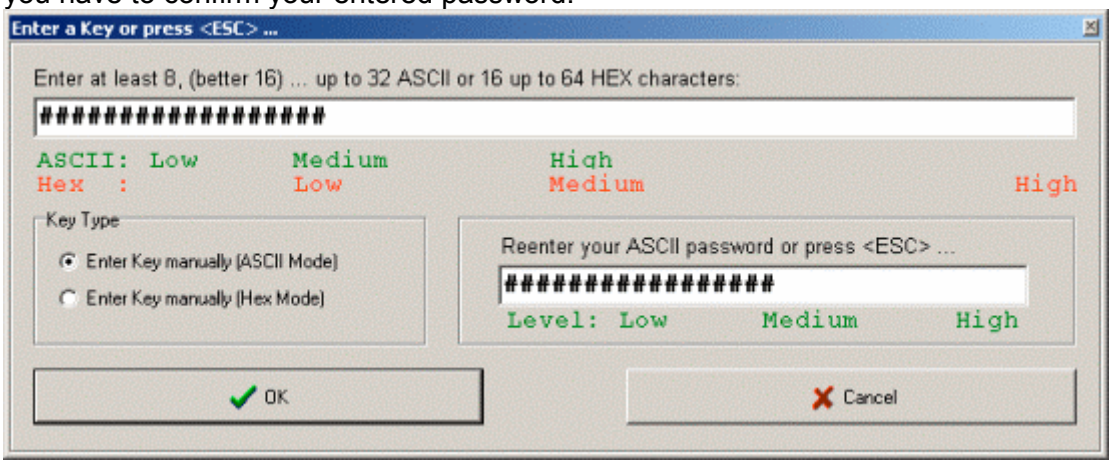
3.1.1 Encode a file

Press the "EnCode/Decode" button and select a file.
Depending on whether you have chosen an Encryption Mode
a password for encryption is required:

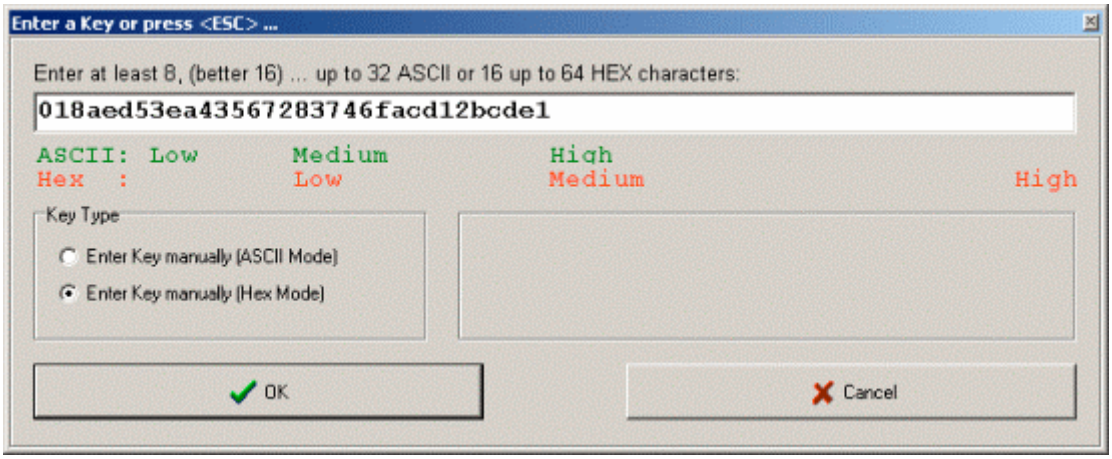


Tip: You may use lots of non-standard-alphanumerical characters like „~_#+.,?$%&“ to ensure that your precious data won’t be decrypted by a brute force attack.

If you have choosen a hidden password input (see 3.3.3),
you have to confirm your entered password.



Much more efficient is the new feature with version 1.2:
You can enter the en/decryption key in hex mode.
With this mode you are able to use all 256 different 8 bit combinations.
(see appendix if you don't know what „hex" is...)



If you want to Escape you may press "Cancel" or press the <ESC> Key.
The whole Encoding/Encrypting process aborts here.

If you don't have to enter a password because of no encryption or you have entered a correct password the encoding process starts and the result is shown in a listbox: (see below)

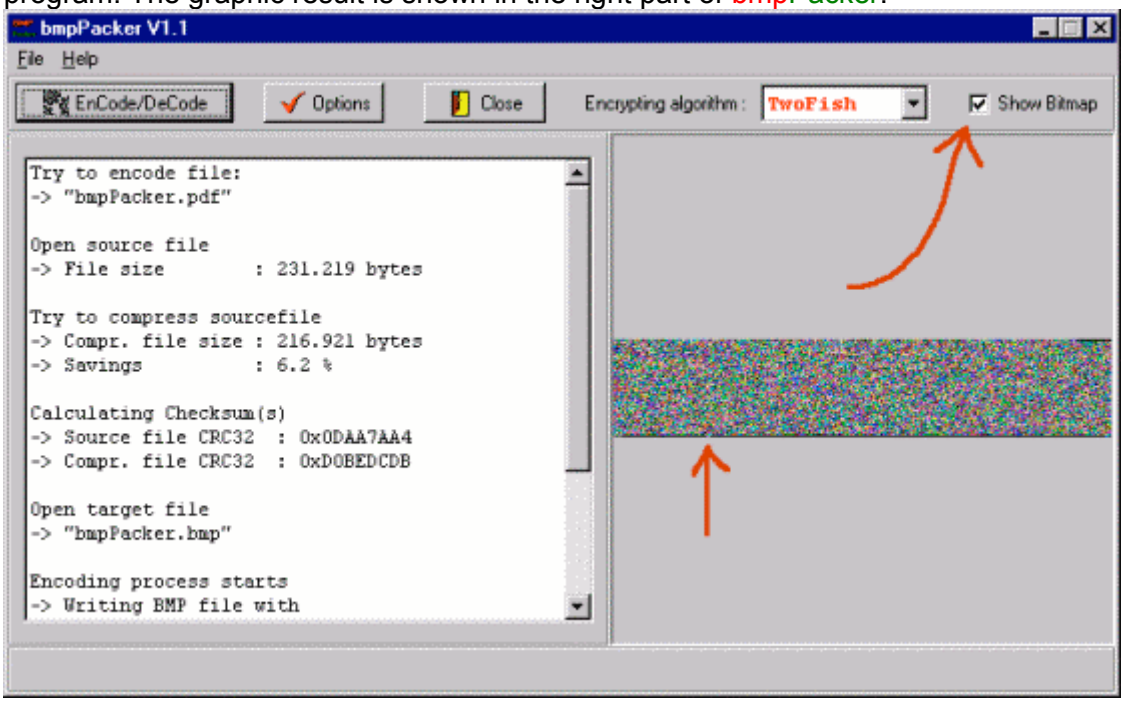
```
Try to encode file:
-> "bmpPacker.pdf"

Open source file
-> File size         : 231.219 bytes

Try to compress sourcefile
-> Compr. file size : 216.921 bytes
-> Savings          : 6.2 %

Calculating Checksum(s)
-> Source file CRC32  : 0x0DAA7AA4
-> Compr. file CRC32  : 0xD0BEDCDB

Open target file
-> "bmpPacker.bmp"
```

```
Encoding process starts
-> Writing BMP file with
-> 608 x 120 Pixel

Enabling password encryption
-> CRC Key : 0xCBCC
-> TwoFish encryption method
-> Block cipher mode = 192 bit

Ready in (hh:mm:ss) = 00:00:00
```

Ready

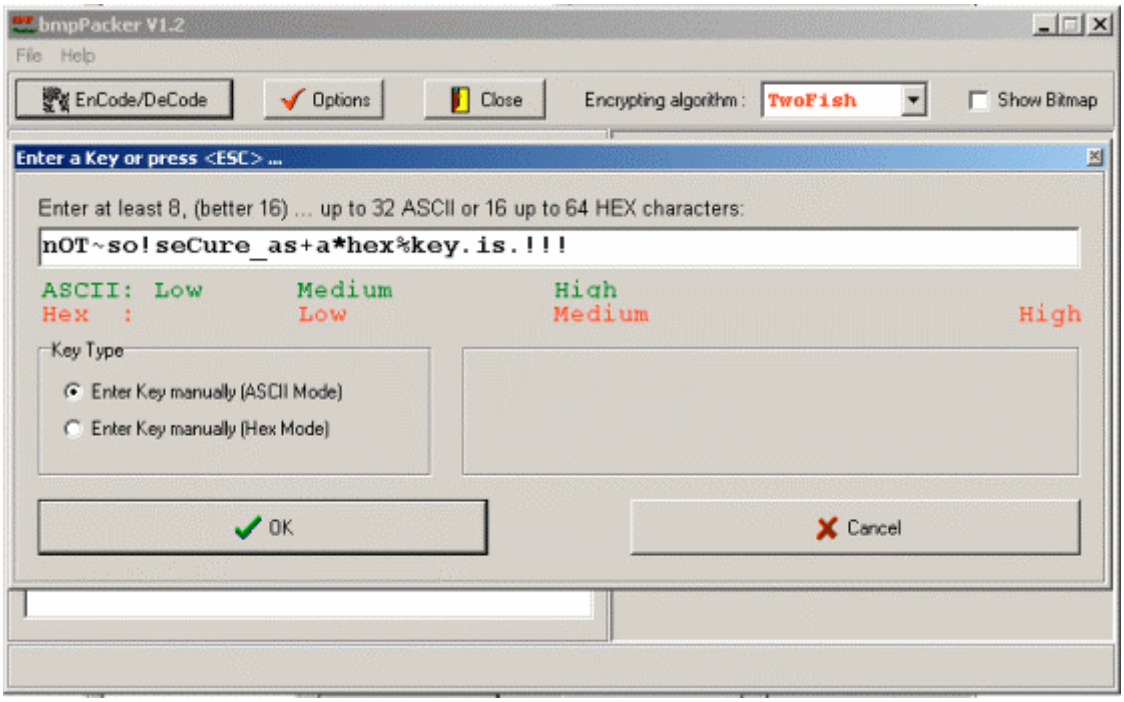
Tip: You may double-click onto the Listbox which shows the results of the conversion. With that double click, you copy the information of the listbox to the Windows clipboard.

You may watch the result by pressing the "Show Bitmap" Button BEFORE you encode a program. The graphic result is shown in the right part of bmpPacker:

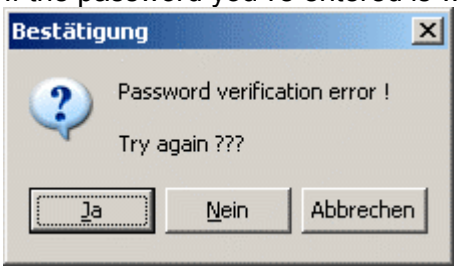


3.1.2 Decode a file

The same way how to encode:
Press the "EnCode/Decode" button and select a BMP file.
If the BMP file is encryted by a password you have to enter the correct password



If the password you've entered is wrong you get a warning message:



After the correct password is given or no password is required...

The Decoding process starts and the result is shown:

```
Try to decode file:
-> "NVIDIA_nforce-1.bmp"

Open source file
-> File size        : 98.550 bytes
-> Searching for bmpPacker header

Decoding process starts
-> Password is required...

Writing target file :
-> "NVIDIA_nforce-1.0-0241.suse80.i386.rpm"
-> Original   file size : 97.646 bytes
-> Compressed file size : 95.863 bytes

Enabling password decryption
-> CRC Key : 0xE910
-> TwoFish decryption method
-> Block cipher mode = 192 bit (medium)
```

```
-> Calculation CRC 32 checksum ...

Checking compressed file consistency...
-> Stored CRC32 checksum     : 0xF23D1148
-> Calculated CRC32 checksum : 0xF23D1148
-> Compressed File should be OK :-)

Decompressing file ...

-> Calculating CRC 32 checksum ...

Checking file consistency...
-> Stored CRC32 checksum     : 0x1E9B3D38
-> Calculated CRC32 checksum : 0x1E9B3D38
-> File should be OK :-)

Ready in (hh:mm:ss) = 00:00:00
```

3.2 Encode/Decode automaticaly

bmpPacker can receive a filename given as a parameter from the commandline or if you place a link on bmpPacker.exe in the "SendTo" folder of your windows profile:

For XP Users :

```
c:\Documents & Settings\<your user profile>\SendTo
```

For NT & W2k Users :

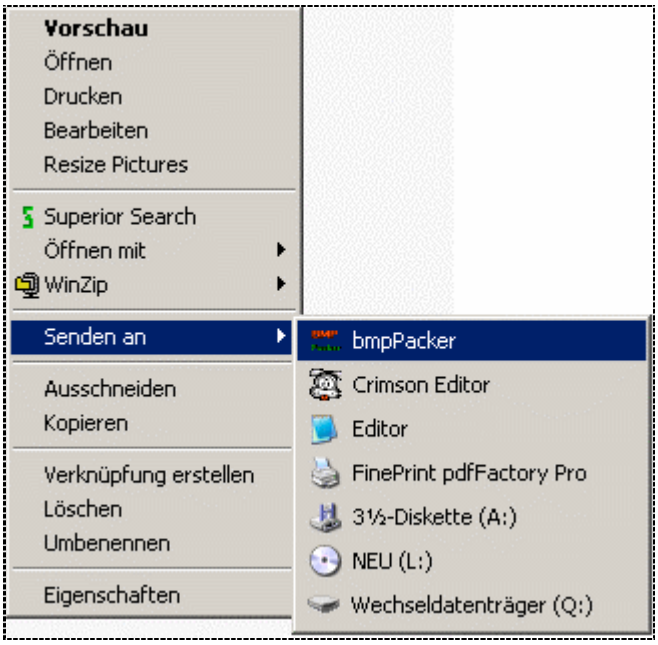
```
c:\<Windows-installation-folder>\profiles\<your user profile>\SendTo
```

For Win95/98/Me Users:

```
c:\<Windows-installation-folder>\SendTo
```

For using bmpPacker in automatic mode you may select your file with the windows file explorer and press the right mouse button when the mouse is placed on it.

After that you choose "SentTo" or "Senden an" and press the menu item "bmpPacker" with the left mouse button.



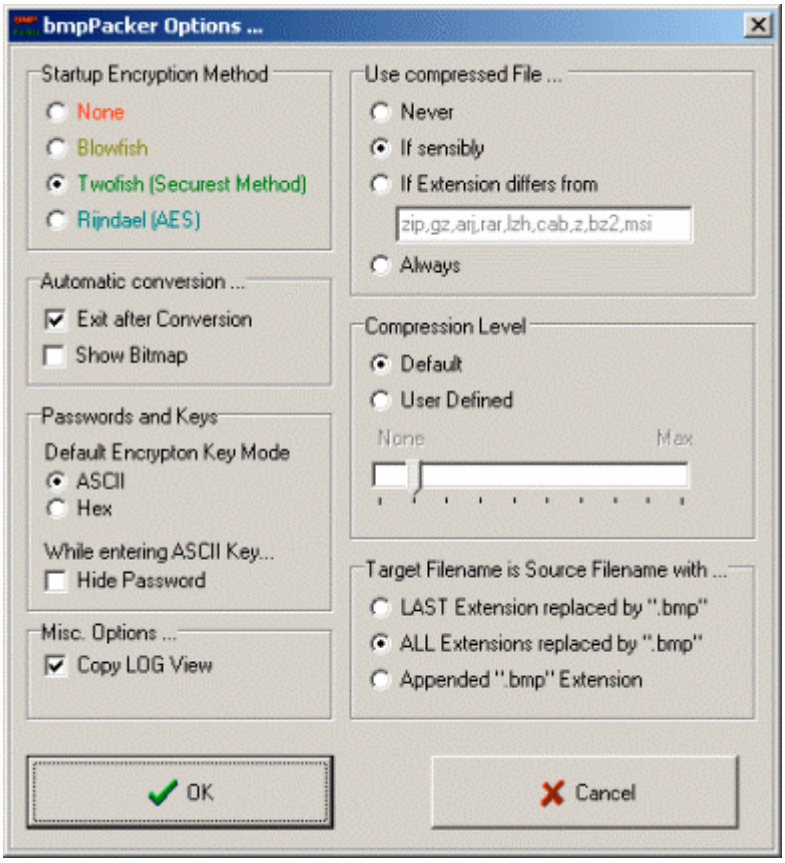
bmpPacker starts and en- or decodes the selected program.

3.3 Options

Since V1.1 you can select a lot of options.
To get to the Options window you may press the "Options" Button or select the "Options" Menu item of the main menu "File".

Some changes to the options take effect immediatly - others take effect only after your next start of bmpPacker.

The changes are saved before the program exits.



3.3.1 Startup Encryption Method

This is the Encryption method that is used by default.

3.3.2 Automatic Conversion

- **Exit after Conversion**
  If you give bmpPacker a filename by the command line (see 3.2) this option forces bmpPacker to end after encoding/decoding the given file.

- **Show Bitmap**
  This option determines whether the "Show Bitmap" button on the main application is checked or not after a new start.

3.3.3 Passwords & Keys

- **Default Encryption Key Method**
  ASCII : Give keys as ASCII string by default.
  Hex : Give keys as Hex string by default.

- **Hide Password**
  This option determines if your password is shown as plain text or in hidden mode while you enter it for a encrypting or decrypting process. This option works only in ASCII mode.

3.3.4 Use compressed file ...

Since V1.1 bmpPacker is able to compress files before encryption. These option group determines how a sourcefile has to be treated.

- **Never**
  Compressing function is disabled.

- **If sensibly (Default)**
  bmpPacker compresses the sourcefile automaticaly and decides after that which of the two files should be used.
  => If the compressed file is smaller than the sourcefile the compressed file is encoded.
  => If the compressed file is equal or bigger than the sourcefile the sourcefile is encoded.

- **If Extension differs from...**
  The file is compressed if the extension of the sourcefile doesn't match one of the given extensions in the edit-field below. The extensions have to be given separated by a comma or a semicolon. There is no case sensitive checking so "ZIP" or "Zip" or "zip" is the same. The default extensions are:

zip	Compressed file, managed by **PKZIP** or **WINZIP**
gz	Compressed file, managed by **GNU ZIP** (gzip)
arj	Compressed file, managed by **ARJ** file archiver
rar	Compressed file, managed by **RAR** file archiver
lzh	Compressed file, managed by **Lharc** file archiver
cab	Compressed file, managed by Windows
z	Compressed file, managed by the UNIX/LINUX/BSD **compress**
bz2	Compressed file, used by a lot of UNIX/LINUX/BSD systems
msi	Microsofts own compression format

- **Always**
  Compress and encode the compressed file any case.

3.3.5 Compression Level

If a file is compressed, this option group determines the compression settings.

- **Default**
  Use default compression level.

- **User Defined**
  Use the compression level of the trackbar below.
  The trackbar is divided into 10 steps:
  Full-Left = No compression
  Full-Right = Maximum compression

3.3.6 Misc. Options

- Copy LOG View
  After encoding or decoding the LOG view content (left side of the program) is copied to the Windows clipboard (as Text).

3.3.7 Target Filename is Source Filename with...

- Last Extension replaced by „.bmp“
  The target filename is the same as the source filename, except that the last extension is replaced by „.bmp“. This is the setting of the former versions of bmpPacker.
  Example:
  Source filename = xyz.txt.tar.**gz** => Target filename = xyz.txt.tar.**bmp**

- All Extensions replaced by „.bmp“
  The target filename is the same as the source filename, except that all extensions are replaced by „.bmp“.

Example:
Source filename = xyz.**txt.tar.gz** => Target filename = xyz.**bmp**

- Appended „.bmp“ Extension
  The target filename is the same as the source filename, except that „.bmp“ is appended.
  Example:
  Source filename = xyz.txt.tar.gz => Target filename = xyz.txt.tar.gz.**bmp**

3.4 Configuration fileformat

The Options are stored in a file which is called "**bmpPacker.cfg**".
You have to ensure that the bmpPackers directory is not write protected so that this file can be written. The size of the file is dependent on the extensions you've given in the edit-field of the compression options.

3.5 Commandline options

The commandline syntax:

```
bmpPacker [ -K:Key | -k:Key | -B | -b | -T | -t | -R | -r | -N | -n | -S |
-s ] [filename]
```

Meaning:

`-K: oder -k:`	Hex-„Key“ which should be used for command line compression. (To use this option effectively you may activate the option „**Exit after Conversion**“)
`-B or -b`	Use Blowfish for encryption
`-T or -t`	Use Twofish for encryption
`-R or -r`	Use Rijndael for encryption
`-N or -n`	Use no encryption
`-S or -s`	Show Bitmap
`filename`	A file that has to be encoded/encrypted or decoded/decrypted

These commandline options override the options you've chosen in the Option form but they do not change them.

For example:
If you have chosen Rijndael as your favorite encrypting algorithm and you call:

```
bmpPacker -T myfile.doc
```

...your file is encoded using the Twofish algorithm. After you start bmpPacker anew (without commandline options) your selected algorithm Rijndael is used.

4 Features

The program bmpPacker was written in C++.
I've used the Borland C++ Builder 5 to compile the project.

http://www.borland.com/cbuilder/index.html

The manual was written with MS-Word 97,
the conversion to PDF was done by Adobe Acrobat 4.05.

4.1 Encrypting Methods

All of the chosen encryption algorithms in this program are much more secure
than the two best known synchronous encryption algorithms DES & triple DES.

4.1.1 BlowFish

A highly secure 64bit block cipher encrypting algorithm with variable-length key.
Blowfish was designed in 1993 by Bruce Schneier
Many applications use BlowFish.
Open BSD uses this algorithm to crypt the systems passwords etc.

The base sourcecode was written in C by Bruce Schneier.
You may refer these informative link to Bruce Schneiers security company:

http://www.counterpane.com/blowfish.html

The C++ code that I've implemented was written by Jim Conger.
(The Code is also available at counterpanes download page)

4.1.2 TwoFish

TwoFish is also developed by Bruce Schneier.
It's not so wide spreaded but more secure than BlowFish.
Its block cipher size is dependent on the password length from 128 up to 256 bit.
Until today nobody could crack TwoFish.

You may refer these informative link to Bruce Schneiers security company:

http://www.counterpane.com/twofish.html

The C/C++ code that I've implemented was written by Dr B. R. Gladman

You may refer the following link for more information:

http://fp.gladman.plus.com/cryptography_technology/aes2/index.htm

4.1.3 Rijndael

Rijndael is the AES winning encryption mode of the year 2002.
Rijndael was developed by Vincent Rijmen and Joan Daemen.
Its block cipher size is dependent on the password length from 128 up to 256 bit.
Until today nobody could hack Rijndael.

You may refer these informative link:

http://www.esat.kuleuven.ac.be/~rijmen/rijndael/

The C/C++ code that I've implemented was written by Dr B. R. Gladman

You may refer the following link for more information:

http://fp.gladman.plus.com/cryptography_technology/aes2/index.htm

4.1.4 NIST National Institute of Standards & Technology

This is a link to the computer security division (CSD) of the united states of america.

http://csrc.nist.gov

Link to the Advanced Encryption Standard

http://csrc.nist.gov/encryption/aes/

4.2 Checksum Methods

I used a **CRC 16** Routine to calculate the password checksums and
a **CRC 32** Routine to calculate the different file checksums.
Link to the CRC32 Sourcecode:

http://www.createwindow.com/programming/crc32/crcfile.htm

4.3 Compression Methods

I've implemented the *gnu* **zlib compression library** from Jean-loup Gailly and Mark Adler which is also used by the ***gnu zip*** (gzip & gunzip etc.) programms.
You may refer the following link for more information:

http://www.gzip.org/zlib

4.4 History

Version	Build – ID	Contents
0.1	2002-10-14	First Version
0.2	2002-10-18	• BMP filechooser filter now working • A "Encryption" & "Decryption" hint is shown in the listbox • Source file name can be given by the command line • The target file is written to the same folder as the source file • The complete target file name is shown in the bottom panel
0.3	2002-10-28	• If source file name is given by command line (maybe the FAT16 short name), the full long file name is read from the file system • A CRC16 function builds a checksum for the passwort in order to check the correct password
0.4	2002-10-30	• The listbox now shows the CRC16 password checksum value as an unsigned short (no negative values are shown anymore)
0.5	2002-11-07	• A new standard encrypting method named **BLOWFISH** has been integrated.
1.0	2002-11-12	• Two high secure encrypting methods have been implemented: **TwoFish** & **Rijndael**
1.1	2002-11-27	• Automatic file compression implemented • An Option formular has been implemented • Commandline Options have been implemented • A new password input field has been implemented
1.1a	2003-06-13	• Some corrections after the document was reviewed by A. Miller
1.2	2003-11-30	• Listbox shows now current information (you don't need to scroll

		anymore) • Keys can be given as hex strings • Option -n/-N works now. • If a bmpPacker configfile is not found in the current directory, the searchpath is used to find a configfile. • The option -K enables the possibility to give a password by command line hex string. • New target filename options have been implemented

4.5 What is a „hex“ key

Definition:
hex - aka **hexadecimal**

The base 16 numbering system, sometimes used as a short way of representing binary numbers. The digits 0-9 are used, plus the letters A-F which stand for numbers 10 to 15. The farthest-right digit is the ones place; the digit next to the left is the 16s place; the next place to the left is 16^2 = 256, etc. Each place is 16 times the place immediately to the right of it.

Samples:
12 hex => 1*16+2 => 16+2 => 18 decimal
3F hex => 3*16+15 => 48+15 => 63 decimal
FA hex => 15*16+10 => 240+10 => 250 decimal

So you are able to identify any possible alphanumerical character but also the characters that are not reachable in a normal edit field of Windows. You only have to consider that you have to give in hex mode two characters to define one ASCII character.
(Your key gets two times longer than a corresponding ASCII key)

Hint:
bmpPacker does not distinguish between big or a small characters in hex mode
(„3A“ is the same as „3a“)