## Topic not found

Please contact the author to get the latest version of BFA95.HLP.

## Introduction

Blowfish Advanced 95 (BFA95) is a file encryption program. That means you can input a secret password, only known by you, and BFA95 will encrypt all selected files with that password. After the encryption **nobody** has access to the original data if (s)he doesn't know the **right password**. If you want to restore your original files you only have to type in the same password and BFA95 will do the rest.
So far, so good. But why do I need such a program, you might ask. Well, we're living in a world where **informations becomes more and more important**. And today's informations are mostly stored on computer systems. Ask yourself: which data on my computer's hard drive is **private**? Or **confidential**? There might be some love letters you want nobody to read them. Or, much more important, sensitive buisness data, documents containing informations about your financial status, your next step against the competition and so on. Or just those pictures with the sexy stuff you got from the Internet. Now the second important question: who has access to your computer? Please don't be naive, the answer is: nearly **everybody**. Everybody can turn on your computer and get any information that is stored on it. In a network people might have access to your computer because all users have the same rights. If you have a protected account the system administrator is still the last instance and is able to watch everything you read or write in the network. Not to talk about the real bad guys: crackers breaking into computer systems to steal important data. The list of those possible and really frightening situations is very long and if you've got enough fantasy and/or technical background you'll find even more of them.
Sounds like paranoia? No, it's not. We're in the digital age now and we should get used to accept **digital keys** as well as we do in real life. There we use keys to lock the door to our homes, to avoid someone reading private documents and so on. It's time to protect the privacy in the computer world as well.
The solution for all these problems is **encryption**. If you encrypt a file containing sensitive informations with a proved algorithm and a password complex and long enough to resist a brute force attack (that means to try out all possible passwords) your private data will be safe. Crackers might be able to copy the encrypted files but they are absolutely worthless without the right password to decrypt them. But that's not all encryption can do. File-based programs as BFA95 are able to encrypt your valuable software, too. Just encrypt your new word processor in the office before you leave to home. Nobody can steal the program now. Another important aspect is **authentication**.You're writing a new book or just an interesting article for the local newspaper? Encrypt it and you can be sure that nobody will be able to **modify** even a single bit in your document without your knowledge.
File encryption software should be **easy to use**, should offer the **best encryption technology** available and last but not least it should be **fast**. BFA95 was designed to cope with all of these demands. It's very **comfortable** to use, has **powerful features** for every daily situation and uses the **latest and safest state-of-the-art technologies** to protect your data.

## Registration

This is a shareware version of Blowfish Advanced 95 (BFA95).
After the a trial time of 21 days you should get your own copy of BFA95 for a very small registration fee of **US $10.00**.
For more informations about registration please click here:

**Registration for International Users**

**Registrierung für Benutzer aus der BRD**

If you want to register the program now you can find a nice **registration form here...**

You are encouraged to spread this shareware around the world by publishing it on CD-ROM, BBS, offering it via FTP or the World Wide Web. The only limitation is that the complete program package should not be changed in any way.
The maximum key length of this shareware version has been restricted to 5 characters (40bit), for that there should be no problems for re-exporting BFA95 out of the United States.
Read the accompanying manual before using the program on "live" data. This program contains powerful encryption technology and data wiping features that can destroy data if used improperly. This software is provided as is and without warranty.   The author assumes no liability for damages, either direct or consequential, which may result from the use of this product.
It's forbidden to disassemble, modify or reengineer this software in any way.
All trademarks in this documents are trademarks and owned by their owners.

If you have problems, questions or suggestions you can contact the author at the following address by snail-mail:

**Markus Hahn**
**Schellingstrasse 13**
**72622 Nuertingen**
**Germany**

You can also send an Internet e-mail to the address:

**hahn@rz.fht-esslingen.de**

(you may even try hahn@pcmail.rz.fht-esslingen.de, if the standard address doesn't work).
Latest shareware versions of Blowfish Advanced 95 can be dowloaded at many software servers in the Internet or directly at the author's homepage:

**http://www-hze.rz.fht-esslingen.de/~hahn/software.html**

Also watch out for **Blowfish Advanced 97**, it's already under development.

## Blowfish Advanced 97

**Things that are planned:**

- Easy-to-use symmetric key database (exchange with PGP support)
- Built-in ASCII editor
- Direct clipboard encryption/decryption support
- Faster data compression
- Archive storage
- UCDI compatible (Universal Crypt Driver Interface)
- Improved file format: unicode filenames and 64bit file length support
- Available in US English and Deutsch.
- CryptFile.DLL, offering an encryption interface for other applications

The final release will be available in June 1997.

**Further suggestions?**

Don't hesitate to send them to the author via e-mail: **hahn@rz.fht-esslingen.de**

## How to use BFA95 with an e-mail client?

Blowfish Advanced 95 can be used to send encrypted files via e-mail:

1. Check if your e-mail client has the capability to attach binary files to an e-mail (very most of them can do that). In the Microsoft Internet Mail program (bundled with the Internet Explorer 3.0, available for free at www.microsoft.com) e.g. you should use the menu item "Insert...File attachment" to add files to your e-mail.

2. Encrypt the files which you want to send with BFA95 as usual. It's recommended not to rename the files (so you'll be able to select them later in the file dialog box more easily) but to compress them to make the e-mail as small as possible.
Attach the encrypted files to your e-mail and send it.

3. The receiver must extract the attached (and encrypted) files. In the MS Internet Mail just click on the menu item "File...Save attachments...". Then decrypt them with BFA95 and the right password to get the secret content.

**Please remember:** Blowfish Advanced 95 only uses **symmetric encryption** that means you must exchange the secret password with the receiver via a **secure channel**. Don't use the telephone (it might be tapped)! Don't e-mail it without any additional protection (e.g. with PGP, if you know how to use it) ! The best way might to meet the receiver in the real world and tell her/him the actual password (if possible).

Besides, the outcoming encryption program **Blowfish Advanced 97** will offer you all what you need to exchange files and text messages very comfortable with fast and easy-to-use key exchange and much more.

## Registration for International Users

**Why should you register?**

Blowfish Advanced 95 (BFA95) is shareware, meaning that you can test the program for free and check if it's useful for you. You have to register if it's useful for you (i.e. you pay a small registration fee), then you'll receive the full version of BFA95.

As you can seen, shareware is pretty much like commercial software you can buy in a shop, but you can test it before buying it. If you continue using Blowfish Advanced 95 after a trial period (21 days maximum), you have to register. If you don't register, your copy of Blowfish Advanced 95 becomes an illegal copy.

If you think Blowfish Advanced 95 is worth using, then it's certainly worth registering. And BFA95 is really unexpensive (US $10.00). If you are part of a business and wish to use several copies of the software within your company, greatly reduced site licensing rates are also available.

When you register, you'll get:

- your personal copy of the newest version
- a special user key which enables you to unlock the latest shareware versions of BFA95
- an update right to future versions of BFA95

You should register now.
The registration fee for BFA95 will rise sooner or later.

For serious companys and institutions it's possible to buy the complete sourcecode of the cryptengine of the actual version of Blowfish Advanced 95. For detailed informations please make your cross onto the registration form.

**What to do if you don't want to register**

That's not so good...
In this case you have to delete Blowfish Advanced 95 (including all files belonging to it) after the trial period (21 days) and discontinue using it. If a newer version of Blowfish Advanced 95 is released which may fulfill your needs, you can test it again, of course.

**What to do if you want to register**

**1.** A good decision!

**2.** Please complete and print out the registration form in this help file. The form fits nicely on a single sheet of paper.

Please write clear and legible if you print the form first and then complete it manually. If you don't have a printer, please write the most important items on a piece of paper:

- your name and complete address (use your real name, no alias)
- if existent: your Internet e-mail address
- how you're going to pay the registration fee (see below)
- your signature
- if you want: comments, questions etc.

For a private registration you must use the name of a real person. Please ask for prices of multiple licences (group licences).

Please send the form by snail-mail to the following address:

**Markus Hahn**
**Schellingstrasse 13**
**72622 Nuertingen**
**GERMANY**

Sometimes the processing of registrations can be delayed somewhat (e.g. vacations, illness, examinations etc.). I apologize for any inconveniences if this happens.

**3.** Pay the registration fee (see next question). Because you'll pay cash or per cheque, please make sure that it is not visible through the envelope.

### How to pay the registration fee

These are the following possibilities:

- US Dollar cash
- An Eurocheque. The amount must be in DM. Users outside of Germany should add DM 5.00 to the normal prize.
- Other cheques, for that payment please add US $5.00

It is not possible to pay by cash on delivery or by credit card. I don't accept International Money Transfer, because it is rather slow (depending on the country), and it involves inacceptable costs for you and me.

### The sad part: not for everyone

Some countries (such as France, Iran, Iraq, Russia and China) have laws that prohibit or regulate the use of cryptography. Please check out your country's national laws and governmental policies on cryptography before attempting to use the registered version of BFA95. You should also make sure that you obtain your registered copy of BFA95 from a place outside of USA and Canada. In some countries, the use of cryptography is restricted by law. For example, in the UK and Germany it is illegal to transmit encrypted data by radio communication. In some countries, it is outright illegal to encrypt data at all. In other countries, they're working on it.
Please keep both of us out of any trouble. Thank you.

```
To:  Markus Hahn
     Schellingstr. 13
     72622 Nuertingen
     GERMANY

Single user registration form for Blowfish Advanced 95.
Please print this file and send it to the address above.
Please write clear and legible!

Name___________________________________________ Date____________

Company________________________________________ Phone___________

Street_________________________________________ FAX_____________

City _____________ State _____ Country _________ Zip ___________

E-mail__________________________________________________________

SINGLE USER LICENSE:
[  ] x single private registration: US $10.00

Please tick:

Payment:  [ ] cash in $US (enclosed)
          [ ] Eurocheque (enclosed), add DM 5.00
          [ ] other cheque (enclosed), add US $5.00

Please send me the software as:

[ ] postal mail (3.5" floppy), add US $5.00
[ ] uuencoded e-mail (don't forget your address above!)

Total                                          US$ ______________


[ ] We're interested in buying the complete source code of the
    cryptengine of Blowfish Advanced 95. Please send us
    informations about price and conditions.


I assure that the import of strong cryptography in my
country is legal and that I'm allowed to use BFA95.



Signature: ________________________________________
```

## Registrierung für Benutzer aus der BRD

Für eine Registrierung dieser Software senden Sie bitte den Betrag in bar oder per Verrechnungsscheck an die folgende Adresse:

**Markus Hahn**
**Schellingstrasse 13**
**72622 Nürtingen**

Verfügen Sie über eine E-Mail-Adresse im Internet, so können Sie sich Ihre registrierte Software auch per E-Mail, uu-codiert (schneller, sicherer) senden lassen.
Bitte im Formular dann auch die entsprechende Adresse angeben.

An: Markus Hahn
Schellingstr.13
72622 Nürtingen

*** REGISTRIERUNGS- UND BESTELLFORMULAR ZU BFA95 ***

Bei handschriftlicher Ausfüllung bitte leserlich und in Druckbuchstaben schreiben.
Mehrplatz- und Gruppen-Lizenzen auf Anfrage.

Vor-, Nachname: ____________________________________________

____________________________________________

Straße, Nr.: ____________________________________________

PLZ, Ort: ____________________________________________

E-Mail-Adresse
(falls vorhanden): ________________________________________

Bitte ausfüllen:

[  ] x Einzelregistrierung (Privat)    : DM 10.00

Zahlung: [ ] bar (anbei)
[ ] V-Scheck/Euroscheck (anbei)

Ich möchte die Software per:

[ ] Postsendung (3.5 Zoll Diskette), DM 5.00 Aufpreis
[ ] E-Mail, uuencoded (Adresse nicht vergessen!)

-----------------
Summe: _______ DM

Datum und Unterschrift: _______________________________________

## File Format Specifications and Technical Reference

**File Format**

When you encrypt a file with BFA95 its original data will be put in a kind of digital envelope. This envelope guarantees save data recovering, secure password error and data corruption detection and at last an authentication check. Another design criteria was serialization, that means that the file's data will always be written one byte after the other, thus no seek operation will be necessary at any time. This enables BFA95 to work with backup utilities, especially streamer tapes, and avoids a lot of i/o overhead because the read/write heads of harddisks mustn't be moved too much. But the most important argument was to develop a format that can be extended in future versions easily but also be read by older versions if the new added informations aren't important for an accurate decryption. BFA95's predecessor, BFA7 (for DOS) uses nearly the same file format, so BFA95 can read and create files which are compatible the the old program.
A file encrypted BFA95 will rawly look like that :

```
==============
|  loader    |
==============
|  header    |
==============          -----
| infoblock  |           / \
| (filename) |            |
==============            |
| data chunk |       encrypted (*)
| data chunk |            |
|    ...     |            |
==============            |
|   tailer   |           \ /
==============          -----
```

Before I start to explain this scheme please remember that all data structures are declared in the Delphi programming language (2.0) for Win32 platforms with no alignment between the structure members. And you'll often hear about keys and passwords. Both were the same in BFA7, but in BFA95 a new key setup was implemented. This key setup will convert each password to a binary key fitting for the desired algorithm(s). For details please read the next section.

The first unit in a cryptfile is the so called "loader", in Object Pascal its structure looks like the following:

```
TCryptfileLoader = record
  LatestVersion : Word;
  Signature     : Longint;
  HeaderSize    : Byte;
end;
```

The loader is only 7 bytes long, so it can be read very quickly. This allows BFA95 to detect unencrypted or unsupported files fast and easily (as you can see in the browser).

`LatestVersion` contains the version number of the program necessary to decrypt this file. By this way BFA95 detects files encrypted with BFA7. Future versions may also have new features which won't be handled correctly with BFA95. The minor version number is stored in the low, the major version number in the high byte.

`Signature` is a 32bit identification number which always contains the hexadecimal constant 75191114 (yes, it's a birthday). If BFA95 detects this value the file will be identified as an encrypted file, otherwise you'll see the "not encrypted" description in the browser.

`HeaderSize` contains the size of the following header structure. This allows future version to store additional data in the header. BFA95 will read and ignore them. So you see from where the loader got its name.

After the loader has been interpreted correctly BFA95 will get `TCrypfileLoader.HeaderSize` number of bytes and treat them as defined the following structure:

```
TCryptfileHeader = record
  Version         : Word;
  TailerSize      : Byte;
  FileInfChecksum : Word;
  FileInfSize     : Byte;
  CBC_IV_Left     : Longint;
  CBC_IV_Right    : Longint;
end;
```

The header contains all data necessary to start the decryption process. Please remember that all bytes following the header are already encrypted (*).

`Version` stores the version of the program which has encrypted the actual file. Therefore future versions might have the chance to fix problems or speed up the decryption process.

`TailerSize` contains the length of the cryptfile's tailer. More about the tailer will follow later.

`FileInfChecksum` contains a CRC16 checksum of the following file information block (FIB). As already mentioned, the FIB will be the first part of the file that will be encrypted. This gives BFA95 the possibility to detect wrong passwords in an acceptable time. If a wrong password is given, the FIB won't be decrypted correctly, the CRC16 check will fail and BFA95 will break with the "password/algorithms failed" message. Please watch that the CRC16 is only a kind of "fingerprint" of the FIB. If the calculated CRC16 is equal to the one stored in the header it does NOT mean that the password is the right one for 100%, but only for a propability of 1:65536. Let us assume that you have to try out 2^32 (about 4 billions) passwords because your estimated length of the password is 4 bytes. The CRC16 check will show success every 65536th password on average. With 2^32 possibilities at all you will get 2^32/65536 possible passwords. Secure passwords should always be longer than 4 bytes (at least 8), so the CRC16 check is not a security lack, but a fine method to detect wrong passwords quickly, even if you want to encrypt your whole file system. Even if a wrong key passes the CRC16 check BFA95 will detect it by the global CRC32 check.

`FileInfSize` is the size of the FIB. Future versions will have larger FIB's, so it's necessary to play the same game as we did it with the variable header size.

`CBC_IV_Left` and `CBC_IV_Right` contain the so called Cipher Block Chaining Initialisation Vector (CBC IV), a 64bit random value that is splitted into two 32bit parts. This start value for encryption makes every encrypted file unique, so even if you encrypt the file with the same password you will always get different encrypted files. Because BFA95 encrypts all data in 64bit blocks the CBC IV must have the same size. All the following blocks, which contain the encrypted data are linked together with the CBC IV as the first element.
You may miss an entry for the algorithm(s) used for encryption. Well, it was decided to resign for that option for two reasons:

- Decryption of multiple files can be managed much faster.
- It's an additional secret information.

If someone doesn't know the algorithm(s) you've choosen (provided that you didn't used those stored in BFA95.CFG) (s)he has to try out all algorithms in addition to all possible passwords. BFA95 now offers 5 different algorithms.

The file information block (FIB) is placed direcly after the header and has the following form:

```
TCryptfileInfoBlock = record
  Attribute   : Byte;
  DateTime    : Longint;
  FileNameLen : Byte;
  Dummy       : array[0..1] of Byte;
end;
```

`Attribute` contains the original attribute of the encrypted file, which will be restored after decryption.

`DateTime` is the original date/time stamp of the encrypted file and also be restored after decryption.

`FileNameLen` stores the length of the filename appended to the FIB. If you forced BFA95 not to store the original filename then its value is always zero.

`Dummy` aligns the size of TCryptfileInfoBlock to the next 64bit (= 8 bytes) border, which you can check out easily:

```
  size of Attribute     1 byte
+ size of DateTime      4 bytes
+ size of FileNameLen   1 byte
+ size of Dummy         2 bytes
                     = 8 bytes = 64 bits
```

This 8 byte border wasn't choosen arbitrary, it depends on how the used algorithms (Blowfish(32), GOST, TDES, Cobra and IDEA) work. They always take 8 bytes (64 bits) together in a block (that's why they called block ciphers) and encrypt these blocks. If your data doesn't fit into an area with a size of a multiple of 8 bytes you will have to add some dummy bytes. These bytes can be everything, but it's better to choose them via a random generator, as BFA95 does. The FIB is encrypted, as already mentioned. So nobody who doesn't know the right password/key will be able to get the original file's attribute, date+time stamp and the length of the filename.

If you decide to store the original filename with the encrypted file then the filename's bytes will be adjusted with random bytes to the next 8 byte border before being encrypted. E.g. with the filename "test1.txt" BFA95 has to encrypt 9 bytes. First the program must add (2*8)-9 = 7 random bytes, then it encrypts these two 8 byte blocks and appends them after the FIB. Now it's obvious why we have to store the original filename length into the FIB.

The encrypted file data follows right after the FIB and the optional stored filename. For easier handling and because BFA95 offers (additional) data compression the original data is splitted, encrypted (and compressed) in so called "chunks". A chunk is a block with a small header and an appended data area. A chunk header for BFA95 is defined like this:

```
TChunkHeader = record
  Flags     : Byte;
  ChunkLen  : array[0..2] of Byte;
  OrigBytes : array[0..2] of Byte;
end;
```

It's not necessary for BFA95 to encrypt chunk headers, because they don't contain any sensitive information (this also saves one byte per chunk since the header musn't be aligned from 7 to 8 bytes).

`Flags` defines a bit mask to describe the header's characteristics. The following bits are valid, all others are undefined and always set to zero:

```
0 0 0 0 0 x x x
          | | |
          | | +--- 1 = last, 0 = normal chunk
          | +----- 1 = compressed, 0 = uncompress
          +------- 1 = 24bit, 0 = 16bit chunk
```

The first bit is set if no chunk will follow the actual one, so the decryption routine can prepare itself to read out the tailer. The second bit is set if the chunk contains compressed data, if so the decryption routine has to pass the decrypted data through a decompressor to get the original bytes. The third bit decicdes wether the chunk length has the maximum size of 64 kB (standard chunk) or 16 MB (extended chunk). If you encrypt files in BFA95 chunks will always created as extended ones. If you want to beware compatibility to BFA7 standard chunks will be created instead.

`ChunkLen` contains the number of bytes of the chunk (without being aligned to the next 8 byte border), and is a 24bit value if the chunk is extended (for BFA95) and a 16bit value if the chunk is compatible to BFA7. The `TChunkHeader` structure above shows a header for an extended chunk, where the 24 bits are stored in an array of three bytes. In a header for 16bit chunks the length-storeing members will look like that:

```
ChunkLen  : Word;
OrigBytes : Word;
```

BFA7 uses an internal I/O buffer of about 64 kB, BFA95 (due to its 32bit nature) a buffer of 256 kB, which well speed up the en-/decryption processes and slightly increase the data compression ratio. Of course it will be possible to use I/O buffers up to 16 MB with BFA95 but that'll be overkill.

`OrigBytes` contains the number of the original (uncompressed) bytes that were stored in the chunk. If the chunk is not compressed it's easy to see that the number in `ChunkLen` will be equal to that in `OrigBytes`.

To make the whole chunking process more clearly let's try an example (assuming that you're encrypting files compatible to BFA7): Your original file is 80001 bytes large. BFA95 gets the maximum number of possible bytes (65536-4096=61440). Let's assume these bytes are compressed to about 45223 bytes, then the chunk put out l will ook like this:

```
-----------------------------------
header: Flags     = 0 0 0 0 0 0 1 0
        ChunkLen  = 45223
        OrigBytes = 61440
-----------------------------------
data: 45223+1 bytes (45224/8 = 0)
-----------------------------------
```

Now you've got 80001-61440=18561 bytes left. Let's assume that this rest isn't compressible. Of course BFA95 tries to compress them, but the compressed result will be larger that the original. It's the last chunk, so the it has to look like this now:

```
-----------------------------------
header: Flags     = 0 0 0 0 0 0 0 1
        ChunkLen  = 18561
        OrigBytes = 18561
-----------------------------------
data: 18561+7 bytes (18568/8 = 0)
-----------------------------------
```

You see that you can't say exactly that a file encrypted with BFA95 is compressed or not. It <u>can</u> contain compressed chunks. There's only one exception: if you use the "Force Compression" option the program will even stored compressed data that is larger than the original. But BFA95 knows nothing about you and the actual settting of this option until it has read out the chunk header and interpreted it.

After the final chunk has been added the encrypted file is closed with the tailer structure:

```
TCryptfileTailer = record
  OrigFileLen  : Longint;
  FileChecksum : Longint;
end;
```

The tailer is also encrypted, but doesn't need any additional bytes to align, because it already fits into a single 8 byte block.

`OrigFileLen` contains the number of original bytes. Please remember that encrypted files are read and written in a linear way. Because of that the original file length can only be stored after all bytes have been read (, compressed) and encrypted.

`FileChecksum` is a CRC32 checksum of the file's original bytes. It's mainly there for detecting data corruption, e.g. bad sectors on floppy disks or undetected transmission errors, but it's also good for the authentication of a file. Because the CRC32 is encrypted nobody can add or remove data from your file without the knowledge of your password.

**Key Setup**

Different encryption algorithms require different key lengths. E.g. TDES needs a key of 21 bytes while IDEA uses a key 128 bits (16 bytes) long. It's very uncomfortable to find passwords that have exactly the needed length every time, so the program has to convert the password into a key for the individual algorithm. In BFA7 the passhrase was repeated by the so called "Simple Repetition Method" (SRM) to get the right key length. If your password was "helloworld" then BFA7 extended it to "helloworldhelloworldh" to be a key for TDES.
BFA95 uses an improved key setup: your password will be hashed by the SHA (secure hash algorithm). The advantage is that the resulting key is in binary form and looks like random numbers. Additionally the length of the password isn't be restricted to the maximum key length of the selected algorithm (as it was in BFA7). BFA95 allows you to use passwords up to 255 characters long. To understand the key setup of BFA95 better let's make two examples...
Our passhrase is "helloworld", the same as above. We want to create a key for IDEA, so we have to convert the password into 16 bytes. The SHA allows us to input as much data bytes as we want and puts out a hash of 160 bits (20 bytes). A hash is nearly the same as a CRC checksum, just like a digital fingerprint. The only difference is that a hash is cryptographically more secure (and needs more time to be calculated). So far, so good. To resize the 20 bytes of the hash to the required 16 bytes for the key we take the first 16 bytes of the hash and XOR the rest (4 bytes) over the beginning of these 16 bytes. So we don't have to throw away any bytes of the hash:

```
password:                        "helloworld"
                                       |
                                      SHA
                                       |
                 a3d4ff09e22710946702eab2cc382596a8e3197322
                 a3d4ff09e22710946702eab2cc382596a8
                 ||||||||
             XOR e3197322
                 ||||||||
IDEA key  :      40cd8c2be22710946702eab2cc382596a8
```

In the second example we assume that our password is still "helloworld" but we need a key for Blowfish

which has the required length of 448 bits (56 bytes). As already mentioned SHA only returns a 20 byte hash. So we have to create 36 more bytes from the passhrase in the following way: we hash the password with the SHA and get 20 bytes. Then we add these 20 bytes to the original password and hash this modified passhrase again. The result is a new hash which means 20 new bytes for our key. Due to the modified password this new hash is completely different from the first one. Now we append this second hash to the modified passhrase again and rehash it to get the last 20 bytes. Of course now we have 4 bytes too much, so we XOR them over the first hash as we do in the first example above. Now we have the needed 56 bytes for the Blowfish encryption algorithm.
SHA is (besides MD5) the most secure hash algorithm available today.

## Algorithms

BFA95 offers the user 6 different algorithms. All algorithms are 64 bit block ciphers and used in CBC (Cipher Block Chaining) mode.

### *Blowfish*

The algorithm was designed by Bruce Schneier (schneier@counterpane.com) and gave BFA95 its name. Blowfish is a very fast algorithm, performing excellent on modern 32bit processors. Another advantage is its variable key size up to 448 bits (56 bytes). It was first published in Doctor Dobb's Journal, issue 4/94. After a year of intensive cryptanalysis it is still unbroken (as reported in DDJ 10/95).

### *Blowfish32*

Standard Blowfish encryption works with 16 rounds, that means that every 64bit block will pass the encryption function 16 times. Blowfish is easy to extend to make more or less rounds. Blowfish32 now is just Blowfish with 32 instead of 16 rounds. That means that the data will be twice more scrambled as with Blowfish16, but the encryption and decryption will take twice more time, too. Although Blowfish32 might be "overkill" it's a good additional feature.

### *GOST*

An algorithm descending from the former Sovjet Union. It's like the russian counterpart to the western DES algorithm. Although it has been used for a long time there are no known weaknesses. The only strange part is a short table of fixed data (the so called substitution boxes, s-boxes). These data is exchangable and might have influence on the algorithms security. Now you can speculate that some USSR institutions might have had "better" s-boxes and some "minor" institutions s-boxes easier to crack, but nobody knows. The s-boxes used in BFA95 are choosen randomly so there's no built-in weakness. GOST isn't as fast as Blowfish, but it encrypts data in 32 rounds as standard (however the encryption function is simpler than the one of Blowfish). GOST's key length is 32 bytes.

### *TDES*

DES is the standard encryption algorithm, designed by IBM in the middle 70es and very often used all over the world. Although it has been cryptanalyzed for more than 20 years now nobody has found any weakness. The only problem of DES is its short key length of 7 bytes. If you have some very fast computers you can try out all possible keys within a few hours ("very fast" means machines which will cost about 250.000 dollars). There are some DES variants, extending the original algorithm to a new one with a larger key. The most popular one is triple-DES, where a 64bit data block will be encrypted 3 times with DES, using 3 different keys. Originally I didn't want to implement any DES variant, because DES is really slow (not to talk about triple-DES), but a lot of people trust this proved algorithm. To speed things up triple-DES was modified in an uncritical way. Before and after the original encryption a 64bit block should be permutated (refering to the DES reference papers). Originally DES was implemented in hardware, not in software, and these permutations were good for loading the block's bytes into hardware registers. But they don't affect the security of DES - so it's the programmers decision to implement the permutation stage or not. BFA95's implementation doesn't use these permutations to obtain more speed. I called this "reduced" algorithm TDES instead of triple-DES, because it's not the original algorithm. The key length is 21 bytes. For the experts: TDES gets a block with 8 bytes, encrypts it with the byte 0..6 of the key, decrypts it with byte 7..13 of the key and encrypts it with byte 17..20 of the key.

*Cobra*

This is a new algorithm, designed by Christian Schneider (100542.2132@compuserve.com). It was published in the newsgroup sci.crypt.research in the middle of April 1996. Normally it's silly to implement an unproved algorithm. But Cobra wasn't designed from the scratch. You can describe it rather as a mutation of Blowfish using some interesting and already proved extending techniques. A real difference is its key size: 32 bytes (Cobra-64) instead of 56 bytes (Blowfish). Cobra was originally designed to be a 128bit block cipher with 24 encryption rounds. Due to it's open architecture it can be reduced or extended for larger or smaller block handling. BFA95 uses Cobra-64 with 16 rounds. An advanced Cobra variant is already available and will find its way into future versions of BFA95 after basic cryptanalysis has been done. For more information about Cobra please contact Chris via e-mail.

*IDEA*

This algorithm was removed in version 8.2 because of license problems (it was not clear if the algorithm can be ised for free in BFA95). There were also (unproved) rumours in the Usenet that this algorithm may have weaknesses.

**Mixed Encryption**

This algorithm-doubling technique was removed in version 8.2 because it was (more or less) furtile. It could be reduced by a so called "meet-in-the-middle attack" to the strength of one algorithm (although this attack seems to be more a mathematical formular than an implementable attack, because one will need trillions (or more) of bytes of memory and much more computer power, too).
Thanks to John Savard to show these circumstances.

**Weak Keys**

Weak keys are such passwords for which the encryption algorithm doesn't produce a secure encrypted output. Only Blowfish and TDES have known weak keys, but they will be correctly identified by BFA95. The program will then pop up a warning message that a weak key was detected and give you the chance to choose another one.

**Random generator**

The random generator used by BFA95 is a pseudo-random-sequence generator using linear feedback shift registers. Although the program needs random bytes only to fill unaligned 8byte blocks and for generating the CBC initialisation vectors this method is much safer than the conventional built-in linear congruential generators of the programming languages' runtime libraries.

**Data compression**

BFA95 uses the LZH (LZ77 with adaptive huffman coding) algorithm to compress data. The original LZ77 algorithm was coded by Haruhiko Okumura, the adaptive Huffman coding by Haruyasu Yoshizaki (developer of the LHA compression utility). The translation into a Turbo Pascal unit was done by Douglas Webb, the conversion into a Delphi 32bit DCU by Markus Hahn.

## Programming

Blowfish Advanced 95 was programmed in Borland Delphi 2.0 (running under Windows 95) on an amd486DX4/100 with 40 MB RAM, 256 kB L2-cache, 500MB and 2 GB HDDs (IDE), S3 Vision 864-PCI graphics adapter (2 MB), 15"-screen, Mitsumi 6x CD-ROM (IDE), Microsoft Mouse, cherry keyboard and iomega Easy 800 streamer (external).

## Security aspects

If you choose a password there's one important thing you must think about first: every password/password barrier can be broken using the so-called brute force attack. This means an attacker will try out every possible password If your password is three letters long and the bad girl/guy estimates that you've only used characters from 'a' to 'z' then are 26 * 26 * 26 = 17,576 possibilities for choosing a password! Yes, "only"! In the worst case (s)he'll have to try out 17,575 combinations to find your password. With a well optimized key search program run on fast computer system the attacker cracks you password within a second! But if you're using only 2 letters more, (s)he must try out 26 * 26*26*26*26 = 11,881,376 combinations, which takes much more time, even with a fast computer. The rule is simple: the more letters (or in binary view: the more bytes for the key) you use, the more possibilites exist, the harder is the encrypted file to break. With a 6 bytes long binary key created from a password (e.g. by a hash function) there are 256*256*256*256*256*256 = 281,474,976,710,700,000,000 possibilites. Let's assume a fast computer can try out 1 million keys per second, them it'll take about 9 years to test all possible combinations. Using just one byte more and the brute force attack will be senseless.
How should you choose your password? Don't use personal common words, such as the name of your husband, wife, daughter, son, dog, cat, lover, your insurance-, house- or telephone-number, the numbers of your birthday or -year (not even in reversed order or another funny combination), etc. You can be sure an attacker will try these ones first! If you don't want to use passwords with extra characters like "&%$*" then use passwords at least 8 letters long. Don't use only small letters! Use numbers. For example if you use "Hollywoody1995" as your password, there are over 15,515,568,480,000,000,000,000,000 combinations, too many for any kind of brute force attack! But watch out: with the growing length of a password the chance to forget it grows as well.
A good password is a sentence with no sense, but which can be remembered, e.g. "The dog is too green?".

Please always remember the golden rule:

**DON'T FORGET YOUR PASSPHRASE!**
**REMEMBER IT!**
**IF YOU CAN'T REMEMBER IT: WRITE IT DOWN!**

The author of BFA95 has no utilities to restore files encrypted with an unknown password of a secure length. If the password is lost, you will never be able to decrypt the file. The program doesn't store the password, neither in the encrypted file nor anywhere else. The password is even deleted in memory before the program is terminated. For additional informations please read the file specifications.

## Credits

Thank you to all people who have supported me in the development of this software.

Special thanks go to:

**Peter C. Gutmann,** for his SFS wiping method

**Jean-Paul Kroepfli,** betatesting and the nice icon

**Herbert Pohlai,** betatesting and good suggestions

**Christian Schneider,** for his Cobra encryption algorithm

**Bruce Schneier,** for his Blowfish encryption algorithm and his famous book

**Applied Cryptography**
**2nd Edition, John Wiley & Sons Inc., ISBN 0-471-11709-9**

**Tobias Überschär,** betatesting

**Borland Inc. (http://www.borland.com),** for their 1st class Delphi 2.0!

**Steve,** for his neverending suggestions and wishes :)

## Encrypt selected files

BFA95 will start to encrypt all the files/folders/drives you have selected after a valid password is entered.
If you select a directory/drive every file in it will be encrypted (even if it's placed in another subdirectory).
Please remember that encrypted documents can't be loaded into the relayed applications anymore and that encrypted executables can't be run, too. They might contain such silly byte combinations that the system will be totally confused and might crash!
And don't encrypt anything you might need for running your system properly, e.g. initialisation files or systems files (like WIN.COM).
Remember also that encrypted files are not compressible anymore. If you want to save disk space compress them before, with programs like WINZIP or   ARJ, or use the compression option.
While BFA95 is working with the selected files you can see a progressbar and a terminal-like screen where the actions are logged. All informations (file names, error messages, compression ratio, new/old filenames, ...) are buffered so you can scroll back and have a look if something went wrong. You can even save all informations shown in the display by clicking on the **Save messages** button in the resume dialog.

## Password Input Dialog

Here you input your **secret** password.
The password can be everything, a single word or a complete sentence. If you encrypt files to be compatible to BFA7 the passwords are restricted to the maximum length of the algorithm's key. Otherwise the maximum length of a password is 255 characters.
Enter the passwords in the boxes, confirm them (for encryption only) to avoid typing mistakes, especially when the characters are not visible. If the password was entered click on the "OK" button to continue.
To get help on choosing the right password please read the security aspects.

## TEMPEST

This switch enables a kind of camouflage against screen interception. With special aerials and receivers it's possible to read someone's screen content over a short distance or to detect keystrokes (sounds like dialing). So a spy can get your password (or just the length if the show password switch is off) while you're typing it in. The flicker (red/blue or green/blue) makes it much more difficult to read out any informations about your password (it's even hard enough to read it on the original screen). And the "noise" on the keyboard wire (you can see the scroll lock LED flicker) will make keyboard interception harder, too.

## Numerical Input

This option allows you to input all possible characters, even the non-printable ones like the carriage return. You must enter the decimal values (in the range from 0 to 255) of the characters seperated by commas.

Here's an example:

```
13,0,255,10,12
```

Here an **illegal** example:

```
233,228,99,270, a
```

(`270` is out of range, `a` is not valid anyway)

Another example:

```
104,101,108,108,111,32,119,111,114,108,100
```

These values represent the ASCII characters of the equal password "`hello world`".

Using numerical input enlarges the key space and guarantees the maximum security with the lowest amount of characters. But a password long enough will do that too, of course.

## Show password

If your **100% sure** that nobody (not even a hidden camera) has the possibility to look over your shoulder you can turn on this switch and see your password instead of the "***". If this option is used your input will be automatically mirrored into the confirmation input box(es).
It's recommended to use the option in combination with the TEMPEST switch to avoid screen interception.

## Decrypt selected files

BFA95 will start to decrypt all the files/folders/drives you have selected after a valid password is entered. If you select a directory/drive every file in it will be decrypted (even if it's placed in another subdirectory). While BFA95 is working with the selected files you can see a progressbar and a terminal-like screen where the actions are logged. All informations (file names, error messages, recovered filenames, ...) are buffered so you can scroll back and have a look if something went wrong. You can even save all informations shown in the display by clicking on the **Save messages** button in the resume dialog.

## Wipe selected files

BFA95 will start to wipe all the files/folders/drives you have selected. If you select a directory/drive every file in it will be wiped (even if it's placed in another subdirectory).
The settings for wiping after encryption will also have affect to the pure wiping operaions (number of overwrites, SFS wiping method).
The number of overwrites or the special SFS wpiping method can be configured in the options dialog.

**Please be careful! If a file has been wiped its original data will be lost forever and can not be restored even with low-level disk editing utilities!**

## Options

Options are useful to switch the functionality of Blowfish Advanced 95 in a kind of way to fulfill your personal wishes as best as possible. With options you can turn special features of the program on or off, declare an algorithm, other ways to input the password and so on.
Although BFA95 works fine with the default configuration, please take the time and try to learn the single options' characteristics to use all the powerful features offered to you.

## Algorithm(s)

Here you can selected which algorithm you want to use to encrypt your files. BFA95 offers today 5 encryption algorithms: Blowfish, Blowfish32, GOST, TDES and Cobra.
For details about the algorithms' characteristics please read the technical reference.

## BFA7 compatibility

This switch allows you to read and write files compatible to its predecessor Blowfish Advanced 7 (BFA7). Due to the fact that BFA7 is a program for DOS there are some differences in the programs' file formats (BFA7 uses a different key setup and has a 16bit architecture). Please remember that BFA7 also offers a mixed encryption method which isn't supported by BFA95 since version 8.2 anymore (because it's more or less furtile).
If you turn on the BFA7 compatibility the browser will show all filenames in the 8.3 format. Long filenames will be concated and can't be restored when they're saved in the old format.
It's strongly recommended to turn off this switch (if you really don't need it) to enable all the advantages of the new BFA95 file format.

## Compress files

BFA95 can compress the data before it's going to encrypt it. By setting this switch you can turn on the data compression. If a file can't be compressed (e.g. multimedia documents like JPG, GIF, MPG or just the archives of compression programs like ZIP, ARJ,...) it'll be stored uncompressed. Depending on what the original file contains it can be compressed up to 98%. Data compression takes a lot of performance, so please keep in mind that the whole process while take much more time than without compression.

## Apply

A click on this button will fix the actual settings, so they'll be stored in the configuration file when the program is terminated and be reloaded the next time. Otherwise all modified options will vanish. This button is useful if you want to change some settings only for one time and don't want your standard configuration to be touched.

## Confirmations

If you turn on this switch BFA95 will ask you to start encrypting/decrypting/wiping after it has finished the file searching process.
This option is useful to speed up the whole work, especially when you've selected a large number of directories and don't wish to confirm every single one.
If you try to wipe complete directories or files you must confirm the processing every time, even if the switch is turned off.

## Wipe original files after encryption

This option forces the program not only to delete the original file but also overwrites its data before it's going to be removed. The reason is that if you delete a file conventionally only an entry in the file allocation table is changed, but the original data stays onto your hard disk until it's overwritten by a new file's bytes. Until. It's hard to say when this situation will occur. But it's easy to undelete a file or to read out the data with a disk editing utility. Very easy, just anybody can do this.
Using wiping this danger will vanish. The original data will be physically deleted by overwriting it according to the NTSC-TG-025 regulations (Version 2, Sep 1991), except if you choose the SFS wiping method. Even the slack will be cleared. You can repeat the overwrite process up to 99 times by setting the Number of overwrites counter.
The wiping itself is done by writing through all buffers and caches of the operating system, so the data will really be **physically deleted**, not only in cache memory (as other programs do so). The file will be (invisibily) renamed before the wiping starts (although it doesn't work well under Windows 95/NT due to caching techniques for the FAT area) and be set to zero length when the wiping has finished.

## Examine file states

If you want the browser to show you the states of the files every time you should select "Always". After changing to a new drive/directory the program will try to scan every file about its state.

These are the possible states:

***???***
(white icon), the file's state is unknown (mostly because its used by another application at the same time)

*Not encrypted*
(green icon), the file could be successfully scanned at is not encrypted with BFA95 or BFA7

*Encrypted with BFA95*
(red icon), the file was encrypted with Blowfish Advanced 95

*Encrypted with BFA7*
(red icon), the file was encrypted with Blowfish Advanced 7, the predecessor of BFA95 for DOS

( If a file was encrypted with BFA7 and the BFA7 compatibility is turned off its icon will be slightly red to show you it'll be necessary to switch to decrypt it (and vice versa). )

The selection allows you to use three scan modes:

**Never**
The program will never scan the files found, every file's state is *???*.

**Only on fixed disks**
The program will only scan drives which were declared as built-in-devices by the operation system. Floppy or CD-ROM drives won't be scanned.

**Always**
The program will scan every drive, even when it takes a long time to examine every single file (especially on large CD-ROM archives with lots of files in a slow drive).

## Exclude

Due to the fact that a file's original attribute is stored in its encrypted counterpart and will be restored after decryption BFA95 is able to encrypt any file on your hard disk, system and hidden files included. With these switches you can set the so called excluding attribute mask. The mask defines a number of attributes. If a file has one of the attributes defined with the (turned on) switches it will not be handled by the program.
The following attributes can be masked out:

- files with the "archive" attribute set
- hidden files
- write-protected files
- system files

If you want to have access to everything just turn off all switches. The default setting will only allow you to access files with the "archive" attribute set. Please remember that encrypted files will always have the "archive" attribute, even if they were originally hidden, system and/or readonly.

## Font

Here you can select the actual font used by the browser and the progress display in the same way you do in all other applications for Windows 95. You can select every font in every size and style that is available on your system.
The default font is "MS Sans Serif" (8 pts).

## Force compression

If BFA95 compresses a file and the compressed data is larger than the uncompressed it'll store the original data to make decryption faster. If you want the program to store the compressed data even when it's larger (in the worst case about 10% plus) you should turn on this switch. This might be good to make a cryptanalysis harder because compressed data looks totally different from it's original counterpart.
This switch is only be available if the compress switch is turned on, too.

## Ignore already encrypted files

If this switch is turned off the program checks if a file is already encrypted. If it is it won't be encrypted for a second time. By turning the switch on you can deactivate this checking. The program won't check (and skip) now for already encrypted files. Please remember that if you encrypt a file multiple times you have to decrypt it multiple times in reversed order, too.

## Ignore CRC32 errors

To check the integrity of a file's original data after it has been decrypted BFA95 makes a CRC32 checksum comparison. If the checksums (the calculated and the stored one) aren't equal the file won't be restored and the encrypted filed is recovered to give the user the chance to solve the problem (e.g. fix disk errors, choose another password). With this switch you can turn off the integrity check. A file will always be decrypted, even if some data are damaged or a (wrong) password passes the CRC16 check as described in the technical reference.

**Please use this option VERY CAREFULLY!**
**Use it only if you're 100% sure that the password is the right one. And don't forget to backup the encrypted file to have another chance for decryption. A file decrypted with the wrong password is just trash!**

## Keep date+time stamps

The original date and time information of a file is stored in the encrypted file and restored after decryption. You can force the program to keep the original date+timp stamp even in the encrypted state by turning this switch on. Otherwise the encrypted file will get the standard "archive" attribute.

## Keep file extensions

This option is only accessable if the rename encrypted files switch is turned on. Then you can avoid the renaming of the original file extensions by turning this switch on.

## Number of overwrites

Here you can set the number of wiping cycles if you use standard wiping. Although a single cycle does three individual overwrites you can extend the data destruction up to 99 times.
This option will be deactivated if the SFS wiping is turned on.

## Warn before overwriting existing files

If you turn off this switch the program won't warn you before it's going to overwrite an existing file.It's not recommended not to use this warning system. Even if files have the same name they might contain totally different data.

## Query for destination

Normally encrypted files will be written at the same place were the original files were placed. If you want to create the en-/decrypted files in an other path, e.g. for backup purposes, you can declare a destination if this switch is turned on. This enables the destination selection dialog which will appear every time before an encryption/decryption process is started.

## Remove original files

If you have defined a special destination for the files the result will look like a copy process with additional encryption. Use this option and the original files will be deleted, too. So you'll have a file movement.
It's strongly recommended to use this option in combination with the wipe original files after encryption option to destroy any original data.
This option is only available if the switch to query for destination is turned on.

## Rename encrypted files

With this option all your encrypted files will be renamed with 12 random letters (8 name, 3 ext.) from the set "A..Z, 0..9". This is useful because **a)** nobody can predict anymore what was in your file because the original name and extension have vanished **b)** you can easily see which files are encrypted and which are not (so there isn't the danger to start encrypted executables and risk a system crash).
Please don't use this option in combination with the store original filenames option turned off, otherwise your original filenames will be lost.
If you want to keep the file extensions, e.g. to have the possibility to delete all textfiles from within an encrypted set of files, then turn on the keep file extensions switch.

## Restore original attributes

Original attributes are stored in the encrypted files. If you don't want them to be restored turn this switch off. This might be useful e.g. if you don't want to recreate write-protected files (e.g. if the original files were located on a CD-ROM).

## Restore original filenames

The program will store every file's original name in the header of the corresponding encrypted file if the store original filenames switch is turned on. If you decrypt the file the original name will be restored. By turning off this switch you can tell the program not to do this, but to keep the actual name of the encrypted file. This might be useful if you have changed the file's name while it was encrypted and want to keep these changes now.

## Save sensitive settings to BFA95.CFG

The program stores the actual options and paths settings in the configuration file BFA95.CFG at the same directory where BFA95.EXE is located.
If you turn off this switch the following informations will **not** be stored in BFA95.CFG (instead of them the default values will be used):

- actual path of the browser
- actual destination path
- the used algorithms
- state of the "Zero CBC..." switch

This is useful if you don't want third persons to know at which directories you have worked or which algorithm(s) you prefer to encrypt your files.

## SFS Wiping

If this switch is turned on the program will use a special wiping method instead of the standard one. The SFS wiping was developed by Peter C. Gutmann and has been implemented in his Secure File System (SFS) first. This method uses 35 special overwrites which will kill every information on a carrier using magnetic storage.
It's recommended to use this option if you want to use 20 or more overwrites (instead of the standard one, although there are 99 overwrites possible).
Please remember that wiping files with this method will take much time, even on fast SCSI drives.

## Show details

With this switch you can turn on the browser's extended columns, showing you the size, date, time, attribute and state of every file. If you turn it off you'll only see the files' names, but the search for new files will also be faster (because less informations must be stored into the list).

## Store original filenames

Blowfish Advanced 95 stores the original filenames in the encrypted files (of course there the filenames are encrypted, too) if this switch is turned on. If you don't want the program to do this turn it off. This will make the encrypted file some bytes smaller.
Please disable this option when you rename encrypted files. Otherwise the original filenames will be lost.

## File wiping requires confirmation

If this switch is turned on every wiping request must be confirmed. This only takes effect if you wipe files by clicking on the Wipe button.
If you want to wipe single files without any warnings turn this witch off.

## Zero CBC and random number generation

Blowfish Advanced 95 uses random numbers to align data and to create the CBC IV (read the technical reference for further informations).
If you turn this switch on the program will create equal encrypted counterparts from the same original file because every random number will be replaced by zeros. This option was implemented for debugging purposes (more or less). It's not recommended to use it.

## About

Tells you the version number and copyright informations about the program. If the program isn't registered yet you can enter your name and the registration code in the dialog which appears if you click on the "Register" button.

## Exit

Click on this button to quit the application.
All actual settings (options, window measurements and position, ...) will be stored in the configuration file BFA95.CFG.

## Help

Opens the help file. Guess you already know that :)

## Browser

The browser of Blowfish Advanced 95 allows you to navigate easily through your whole file system on your computer...

You can **enter directories** just by clicking on them, you can **leave** them by clicking on the "one level up" entry (".."), which can always found at the top of the list.

You can **start applications** by clicking on the COM, EXE, BAT, ... files.

You can **start documents** (e.g. DOC, TXT, HTML, ...) with their relayed applications just by clicking on them.

Click on the **drives** just to change to another data carrier.

By holding down the right mouse button a popup menu with further **file managment** possibilities will appear.

**Select files** by using the mouse or the cursor keys in combination with the CTRL key.

**Sort columns** by name, size, etc. just by clicking on the column headers.

## Popup menu

The popup menu can be activated by pressing the right mouse button and holding it down until a menu item (most are for a more comfortable file managment) is selected.
The following items are available:

### Refresh

Forces the browser to re-read the current directory. Use this method if files/directories have been changed since the last refreshment.

### Set new path

This allows you to change to a new drive/directory by typing it in directly. You can also define a wildcards,like "*.txt" or "manual???.doc", to get certain files. Please remember that the wildcards will ke kept until you change them again. The wildcards are also used if you let the program search for files in a directory recursively to en-/decrypt/wipe them.

### Options...

You can call the option dialog with the right sheet for configuring the browser directly.

### Select...All files

If you want to select all single files in the current path.

### Select...All files

If you want to select all folders in the current path.

### Select...All files and folders

If you want to select all single files and folders in the current path.

### Select...All decrypted files

If you want to select all single files which were *not* encrypted in the current path.

### Select...All encrypted files

If you want to select all single files which *were* encrypted in the current path.

### Select...All files containing string

If you only want to select files with a certain number of defined characters you can do this by using this option. Enter the search string (just like in a word processor) to select all files which names contain this string.

### Copy

Copies all selected files to the destination path declared in a special window which will popup after you've clicked on this item. A copy process can't be interrupted by the user. You can follow the copy progress in the status bar.

### Move

Moves all selected files to the destination path declared in a special window which will popup after you've clicked on this item. A move process can't be interrupted by the user. You can follow the move progress in the status bar.

### Delete

Deletes all selected files after you've clicked on this item. A delete process can't be interrupted by the user. You can follow the delete progress in the status bar. Please remember that files delete with Blowfish Advanced 95 will not appear in the "Recycle Bin" of Windows 95/NT! So you can't (simply) undelete the files. If the files contain sensitive data you should always wipe them!

### Create directory

Create a new directory/folder in the actual path. You must enter the new name in a separate input box. After the directory has been created the browser will refresh the actual path.

### Clear empty disk space

With this option you can clear the whole empty space on the actual drive. The program will create a temporary file and write random data into it until the disk is full. This will kill every sensitive information that might have been deleted without wiping but is still stored on the disk (and can be recovered by low level disk editing utilities, of course).
A popup dialog will show you how much random data have already been written to the disk. You can break the process by clicking on the "Cancel" button.
It's recommended to use this disk clearing as often as you defrag your drives and always after a system crash. Tests have also shown that if Windows 95 accesses a drive in the "MS-DOS compatibility mode" the clearing isn't flowing due to a reduced multi-tasking. On all other systems the clearing task isn't much time-consuming (but also a good method to show how fast your hard drive is) so you can run it in the background.

## Destination Dialog

In this dialog you can declare a destination path for the files you want to copy, move, en- or decrypt. If you copy or move files via the browser the buttons "OK" (ok-button) and "Cancel" (cancel-button) will appear, otherwise "Accept" (ok) and "Leave the files were they are" (cancel).
Press the ok-button if you want the files to be copied/moved or to create the en-/decrypted files at the selected destination. Press the cancel-button to abort the copy/move process or to create the en-/decrypted files at the same places of their de-/encrypted counterparts.
You can enter the destination path either manually or use a special dialog which you can activate via the "Browse" button
For en-/decryption processes this dialog will only appear if the switch for query for destination has been turned on.