# ENCRYPTED MP3 TRACKS

The **MP3Crypt** utility encrypts MP3 tracks, for playback on a H0420 MP3 player or a H0440 audio controller. The H04x0 series use a strong encryption scheme with a 128-bit key. This key is hard-coded in the device; when ordering a series of MP3 players, a customer may request a unique device key for that series.

When encrypting a track with the MP3Crypt utility, you must use the same key as the one in the respective H04x0 series. The key is therefore also provided as a computer file, which must be stored alongside the encryption utility. It is the responsibility of the customer to ensure that this computer file does not drop into the wrong hands. Recommended practices are:

- The key file are provided on a CD-ROM, along with the MP3Crypt utility. Run the MP3Crypt utility directly from CD-ROM (do not copy it to a hard disk). After encrypting a set of files, remove the CD-ROM from the computer and keep it at a safe place.
- Alternatively: implement an encrypted, password-protected disk and store the MP3Crypt utility along with the key file on this disk.

See the "Encrypting File System" (EFS) in Windows XP for more information; various alternatives for EFS are available for other operating systems.

- Running MP3Crypt from a USB stick is also an option (this may be more convenient than using a CD-ROM). For increased security, use an encrypted USB stick. See the TrueCrypt web site for a suitable encryption strategy.

In addition to the device-specific key, you may optionally specify a "user password" when encrypting (and decrypting) the file. To decrypt an MP3 track, both the device key and the password must match with those used for encryption. In summary, this means that an encrypted MP3 track can only be played back:

- on a H04x0 device that has the correct device key, *and*
- when the script passes in the correct user password

ID3 tag information (version 2) is preserved and non-encrypted. However, **the ID3 tag should not be modified after encrypting the file**. An ID3 tag editor may change the tag size and/or the padding, and the MP3Crypt utility depends on proper alignment of the compressed audio data that follows the ID3 tag.

Because of the alignment requirements, an encrypted MP3 track is usually slightly larger than the original MP3 track. Note that if you decrypt the MP3 track using this utility, the utility keeps the alignment that was set during encryption.

Decrypting and encrypted file does therefore not result in a track that is byte-for-byte identical to the original file. The audio data, though, is identical to the original file.

### *Usage*

MP3Crypt is a console mode application. It runs from the command line in a command shell or "DOS box". The utility takes the names of the input files as parameters on the command line. In addition, the utility accepts the following options on the command line:

| | |
|---|---|
| -d | decrypt the file (instead of encrypting it) |
| -k<filename> | set the key file to use |
| -o<filename> | set output directory or output filename (by default, encrypted files get the extension .mpx) |
| -p<password> | Set an optional "user password" for encryption |
| -replace | delete the source (input) files after encryption or decryption |

The key file is always read from a file. Usually there is only one key file, and it is called "**default.key**". Unless you have obtained multiple hardware keys from CompuPhase (and therefore have several key files), there is no need to use the **-k** option.

You can encrypt (or decrypt) multiple files with a single command. The meaning of the **-o** option depends on whether there is only one input file on the command line, or whether there are several files. When encrypting multiple files, the **-o** *must* specify a directory. When encrypting a single file, the **-o** option may specify either a directory or a filename.

With the **-p** option, you set an additional user password for encryption and decryption. This is strictly optional, but it allows you to limit the use of encrypted tracks to specific installations or customers. The user password does not make the encryption stronger or weaker; the item that should be a closely guarded secret is the key file.

Examples of use are:

| | |
|---|---|
| **mp3crypt *.mp3** | Encrypt all MP3 files in the current directory. The output files (with the extension ".MPX") go to the current directory too. MP3Crypt uses default settings for all options. |
| **mp3crypt *.mp3 -oC:\Outgoing** | Encrypt all MP3 files in the current directory. The output files go to **C:\Outgoing**. |
| **mp3crypt track1.mpx -d -pSECRET** | Decrypt **track1.mpx** to **track1.mp3** in the same directory, using the password **SECRET**. Note that the |

| | |
|---|---|
| | track should have been encrypted with the same password to get a correctly decrypted output file. |

To play encrypted files, you need an MP3 player that has the correct device key. If you set a password at the encryption phase, you must also set that password in the MP3 player, See the function **mp3password()** for details. Once the password is set, you can just play the files using the **play()** function. The **mp3password()** function is documented in the "Reference Guide" for the H0420 or Starling audio player.

When the H04x0 MP3 player executes a **play()** command and either the device key or the password is wrong, the MP3 player will read through the encrypted MP3 file at top speed without producing any sound. In rare occasions, an encrypted file may produce a short burst garbled sound. Other than remaining silent and taking only a short time to "play" the file, there is no indication for decryption failure.

### *Additional information regarding the encryption*

In the design of the H04x0 series, the 128-bit key is hard-coded in the Flash ROM. The key is not stored on the Compact Flash card, nor is it transmitted to the MP3 player in any other way. This avoids that the key transmission forms the "weak link" in the schema.

The H04x0 devices implement "ROM protection", which blocks read/write access to Flash ROM. The key can therefore not be extracted or "sniffed" from the device, not even after desoldering the Flash ROM chip from the device.

For obvious reasons, neither the device key, nor the user password are stored in the encrypted track. If they were, an eavesdropper could extract the key/password from the encrypted track itself. It would be like locking the door and hiding the key below the mat (or in a flower pot). Contrariwise: if you loose the device key or forget the user password, there is no way to recover the original MP3 file from the MPX file —except by trying all possible keys, which will take a lifetime. The reason that the H04x0 MP3 player remains silent when "playing" a track where the device key or password are wrong, is that the device has only one way to determine whether the decryption succeeded or failed: check whether the decrypted data is a valid MP3 stream. This is not straightforward, though, because the MP3 file format lacks a file header with a discernible "signature". The device therefore sends the stream to a (hardware) MP3 decoder chip, which ignores invalid data.

The encryption algorithm is "Block TEA", a secure encryption algorithm suitable for variable-sized blocks. The Tiny Encryption Algorithm (TEA) is a high-performance cryptographic algorithm, designed by David Wheeler and Roger Needham at the Computer Laboratory of Cambridge University.

TEA is a Feistel block-cipher which encrypts 64 data bits at a time using a 128-bit key. The "Block TEA" algorithm that the H04x0 MP3 controllers use is a variant (by the same designers) that is faster and more secure on large blocks.

Source: http://www.compuphase.com/mp3/mp3crypt.htm