Hymn Manual

Anonymous (Not Given)

Hymn Manual

by Anonymous (Not Given)

I am, on prinipal, opposed to any DRM technology; I will fight it at every level. If you are so inclined, please read Appendix C, *Law vs. Ethics* for my full opinion on DRM. The hymn program is just one part of a larger fight against DRM; it frees my music so that it can be played anywhere I want to play it.

This document will show you how to use hymn to free your iTunes Music Store purchases from Apple's "FairPlay" DRM scheme.

INFORMATION WANTS TO BE FREE.

Table of Contents

Chapter 1. Getting Started

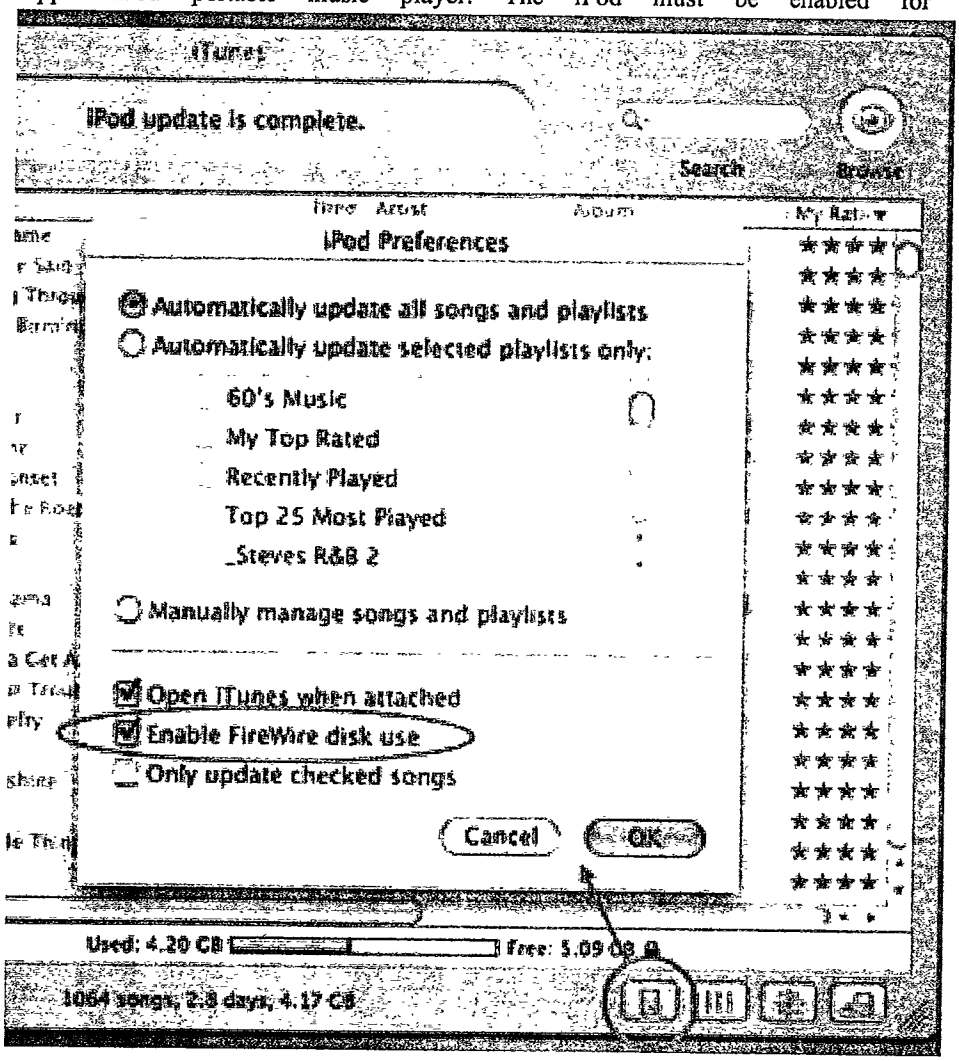
This chapter will tell you what you need to do to get your system set up to use hymn.

1.1. Setup for Microsoft Windows Systems

At the time of this writing, using hymn on Windows systems is easier than non-Windows systems. To use hymn on Windows systems you only need to have iTunes installed and have the computer authorized through iTunes to play all of the tracks you wish to decode. No other setup should be necessary.

1.2. Setup for Other Systems

At the time of this writing, using hymn on non-Windows systems is slightly more difficult than on Windows systems. In order to decrypt music tracks on a non-Windows sytem, you must have access to an Apple iPod portable music player. The iPod must be enabled for disk use:



All of the songs you wish to decode must have been copied to the iPod through iTunes using a computer authorized through iTunes.

On some systems, it may be necessary to set an environment variable that tells hymn your "iPod id". The environment variable is named IPODID and is the GUID of the connected iPod once it is mounted.

How to determine your iPod id:

GNU/Linux

```
bash$ dmesg | grep ieee1394 | grep GUID
```

*BSD (If you know, email me!)

Chapter 2. Hymn Interfaces

There are, as of this writing, two different user interfaces for hymn. There is a command-line interface (CLI) that works on most Unix-like platforms (Linux, OS X, *BSD, etc.) and Windows. There is also a graphical user interface (GUI) written using the Cocoa platform that works on Mac OS X.

2.1. Command Line Interface

In order to run hymn using the CLI, you must be somewhat familiar with command prompts in general. On Unix-like systems, there are generally programs called "terminal emulators" or "shell windows" that will allow you to type commands at a prompt using the keybaord. On Windows systems, there is a command processor called `CMD.EXE`. The hymn command syntax is as follows:
hymn [-l *n*] [-x *ext*] [-v] {*file1*} [[*file2 ... fileN*]] [*destdir*]

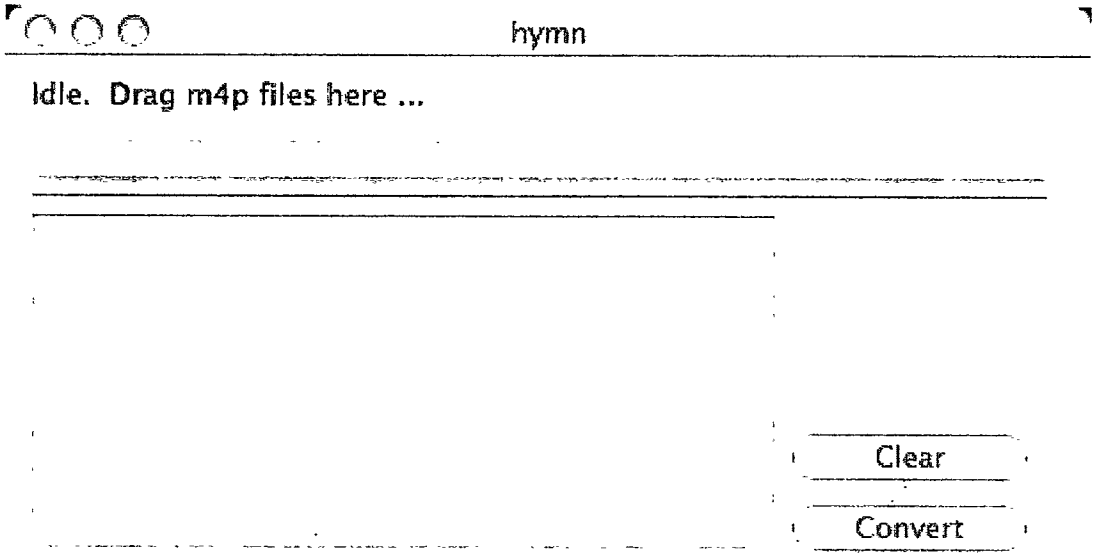
-l *n* The hymn program has some built-in logging facilities that allow you to monitor the progress of what is going on. These logging messages may aid in debugging why hymn may not be working in your environment. The logging level can be set to a number between 0 (no logging) and 5 (maximum logging) and defaults to level 1.

-x *ext* By default, the output file(s) will be written with a `.m4a` file extension. This option allows you to override that default and specify any extension you want (such as mp4, aac, etc.)

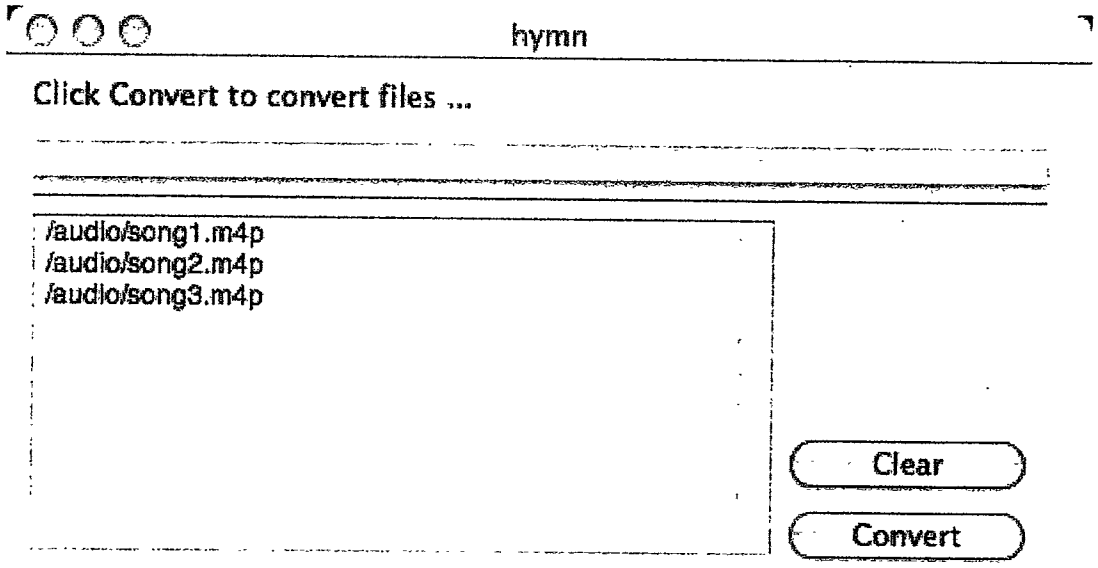
-v Display version and copyright information about the program.

2.2. Mac OS X (Cocoa) GUI

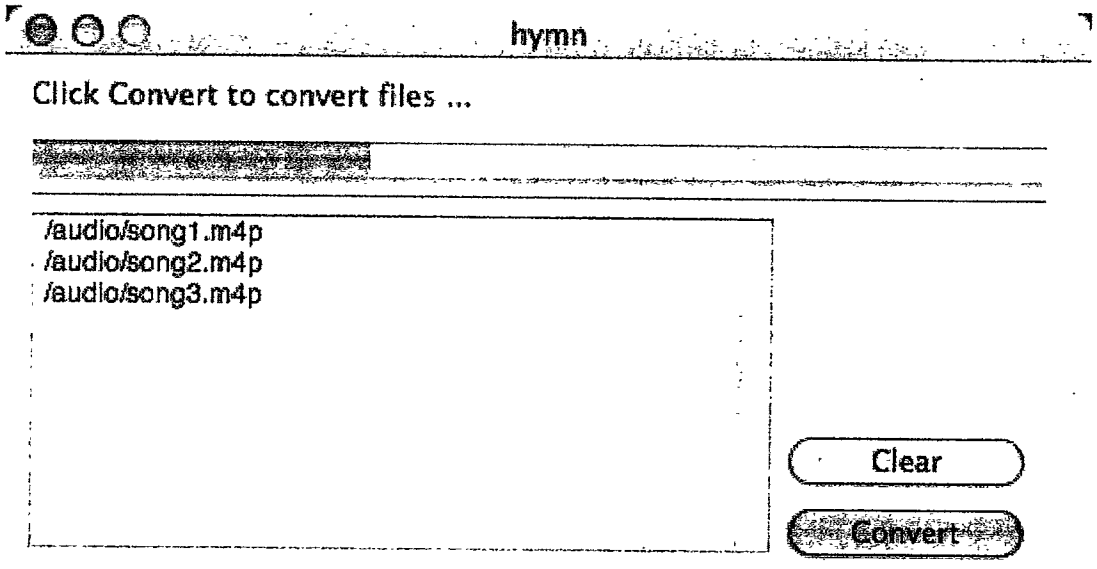
To use the Mac OS X GUI, simply double-click the application icon to load hymn. You will be presented with a window with a status bar at the top, a progress meter, an empty text pane and a few action buttons.



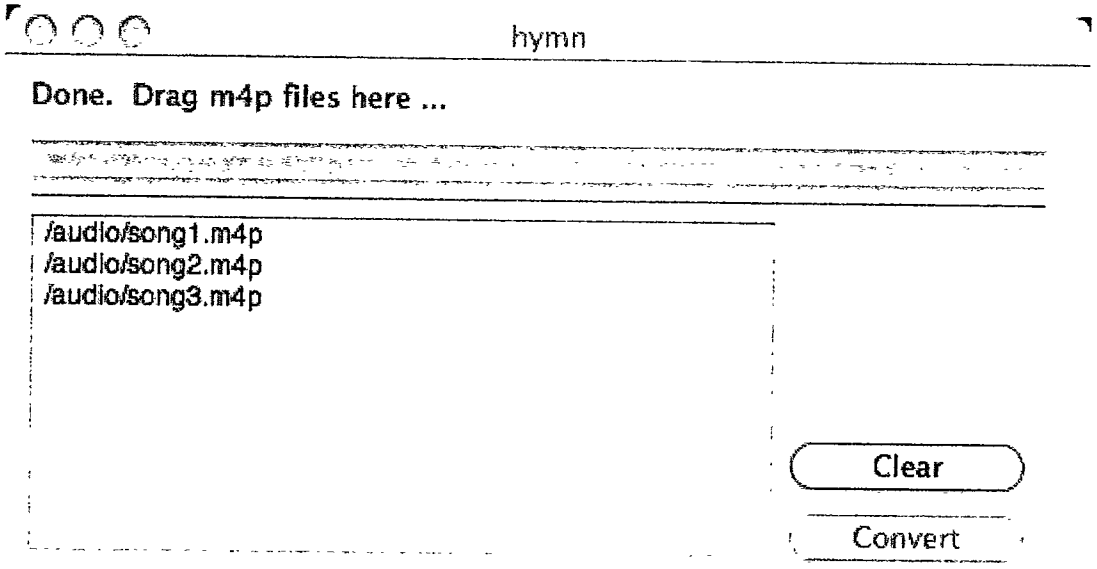
Use of the GUI is fairly straightforward. Find the .m4p files on your disk that you'd like to convert. When you've found them, drag them onto the hymn window. When you do so, the songs you dragged will be listed in the text pane and the Convert and Clear buttons will become enabled.



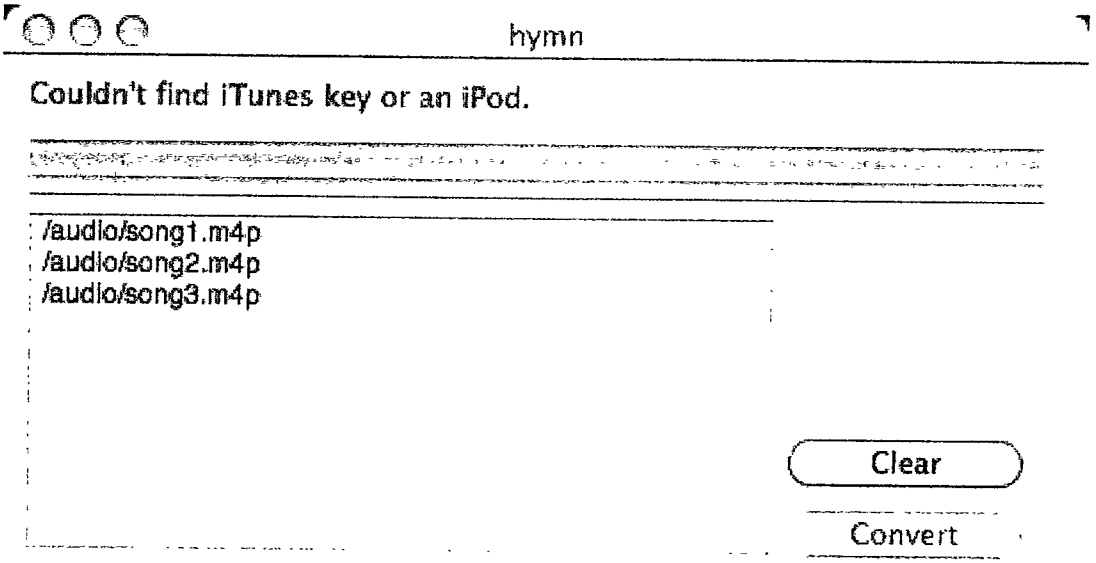
At this point, you can either drag more files onto the window, convert the listed files, or clear the listed files and start over. Once you have the files listed that you wish to convert, click the Convert button. After you do so, the conversion process will begin, giving you status updates along the way.



After the songs have finished converting, the Convert button will become disabled and hymn will tell you it has finished converting the files. At this point, you can either drag new files to be converted or quit the program.



If errors occurred during the conversion, they will be displayed in the status bar in red. See Chapter 3, *Troubleshooting* for help with correcting these errors.



Chapter 3. Troubleshooting

This chapter is meant to give more in-depth technical info when things don't go as expected. It lists all of the known error messages, what they mean, and how to attempt to resolve them.

3.1.

Couldn't open file: <filename>

The specified file could not be opened. Ensure that the file exists and that your user account has access to read the file.

3.2.

Couldn't determine file size of <filename>.

The size of the specified file could not be determined. Ensure that the file exists and that your user account has access to read the file.

3.3.

Couldn't allocate <size> bytes.

Not enough memory is available to process the file. Close some other programs and then try again.

3.4.

Tried to read <size> bytes, read only <size>.

The full file could not be read into memory. Make sure the file did not move or get deleted during processing.

3.5.

Couldn't write file: <filename>

The specified file could not be created. Ensure that your user account has write access to the destination diretory and that there is sufficient room on the disk to write the file.

3.6.

Couldn't find iTunes key or an iPod.

(non-Windows only error.) This error occurs when there is no .drms folder in your home directory and no iPod can be detected in /Volumes or /mnt. Make sure your iPod is connected and is the only device on the FireWire bus before running hymn.

3.7.

Couldn't get DRM key for user.

This happens when one of the user keys could not be decrypted with the system key and could not be found in decrypted form on a connected iPod. Ensure that the computer in question is authorized to play the input file(s) and (if on a non-Windows system) that the iPod contains the song, is connected and is the only device on the FireWire bus.

Appendix A. Frequently Asked Questions

A.1. Technical Questions

A.1.1
. On which platforms does hymn work?

It has been tested on Mac OS X 10.3.3, Gentoo Linux and Windows XP Professional. It should build and work on any automake-compliant unix-ish environment, including MingW for Windows.

A.1.2
. Can I use hymn on a non-Windows platform without an iPod?

This is a complex question that, unfortunately, requires a long and complex answer. First of all, the FairPlay encryption scheme uses multiple different keys. The first key is the system key. On Windows systems, how the system key is generated has been reverse-engineered; on non-Windows systems, it has not. This system key is then used to decrypt each of the user keys that are stored on a given computer. There may be one *or more* user keys per iTunes Music Store account. That is, there is no guarantee that two songs purchased by the same iTunes user will have been encrypted with the same user key.

When iTunes copies the protected files to the iPod, it copies *decrypted* versions of the user keys for those songs to the iPod at the same time. Thus, using the user keys that are stored on the iPod, we can decrypt the songs just as if we had the system key. At this time, this is the only way to decrypt songs on non-Windows platforms.

When hymn runs, it stores (for future use) the keys for future use in a folder inside of the home directory of the user who runs it. Thus, any songs that were encrypted with one of the keys that is stored in this folder should be able to be decrypted with no problem. However, if the songs were encrypted with a different user key, the songs will need to be copied to the iPod and the iPod will need to be plugged in in order to decrypt them.

There is probably a way to copy the information off of the iPod and onto the local machine, but for me, it's just as easy to leave the iPod hooked up, so I haven't tested whether or not that actually works, though I presume it would.

So the answer is both yes and no. :-)

A.1.3
. hymn runs fine, but the files it outputs crash iTunes / Winamp / Quicktime, etc. What's wrong?

This bug should have been fixed in version 0.5.0. If this is happening to you in a version later than 0.5.0, please file a bug. In 0.5.0+ versions of hymn, the program should fail gracefully if the appropriate decryption keys can not be found / generated and not output garbage files.

A.1.4
. My iPod is not detected properly. What's wrong?

Make sure your iPod is enabled for disk use. Also, hymn has only been tested with the FireWire version of the iPod, not the USB2 version.

A.1.5

. The kbps seems to have dropped from 128kbps to somewhere between 125kbps and 127kbps. Is this okay?

This should not happen as of version 0.6.0. If this is happening to you in a version later than 0.6.0, please file a bug. In 0.6.0+ versions of hymn, all of the meta data should be copied exactly as it was in the original song.

A.1.6

. Why aren't the iTunes copyright and "explicit -- parental advisory" tags copied?

This should not happen as of version 0.6.0. If this is happening to you in a version later than 0.6.0, please file a bug. In 0.6.0+ versions of hymn, all of the meta data should be copied exactly as it was in the original song.

A.2. Legal / Ethical Questions

A.2.1

. Why are you trying to promote music "piracy"? Shouldn't musicians make money, too?

First of all, I buy all of my music. In fact, most of the music I buy these days comes from the iTunes Music Store. However, I want to be able to play the music I buy wherever I want to play it without quality loss, since *I PAID FOR* that quality. I want musicians to make money. I want Apple to make money. I don't condone sharing music through P2P networks with the masses, though I believe making a mix CD or playlist for a friend is okay. I also think the RIAA are a bunch of crooks, but that's another story.

Secondly, hymn leaves the apple ID embedded in the output file, so anyone who shares the decoded files on P2P networks is bound to be prosecuted under copyright law.

A.2.2

. But if you don't promote "piracy", why release the program to the public and not just use it for yourself? After all, don't you *know* that people will misuse it?

I believe there are other people who want to use my program legitimately, just as I use it. I don't believe the majority of the people who use my program will use it so that they can share their files on Kazaa, especially since their apple ID is embedded in the files. Anyway, in order to use my program, you had to pay for music on the iTunes Music Store to begin with. These are the people who are *willing* to pay for their music. Besides, should a baseball bat manufacturer stop manufacturing baseball bats just because someone *may* use the baseball bat to beat another person's head in?

A.2.3

. Why is the apple ID left intact in the output files generated by hymn? Is it intentional?

It *is intentional*. Hymn is not meant to be a tool to enable copyright infringement. Having the apple ID in the output files allows anyone sharing such files on P2P networks to be tracked down. I can't think of a legitimate reason to remove the apple ID, so long as the songs are playable

everywhere. As long as it is technically feasible to do so, hymn will leave the apple ID intact.

A.2.4
. Aren't you afraid of being prosecuted under the DMCA?

Yes. That's why this is an anonymously developed project. However, I didn't actually write the code that cracks the DRM. Other people did. I'm just using their code in my program. So, technically, I'm not violating the DMCA.

A.3. Miscellaneous Questions

A.3.1
. I'm not a developer. Will there be a binary release soon so I don't have to try to compile this program to use it?

There are binary releases for the Mac OS X GUI version on OS X and for the command-line version on Windows right now. Most unix users don't mind the extra compilation step.

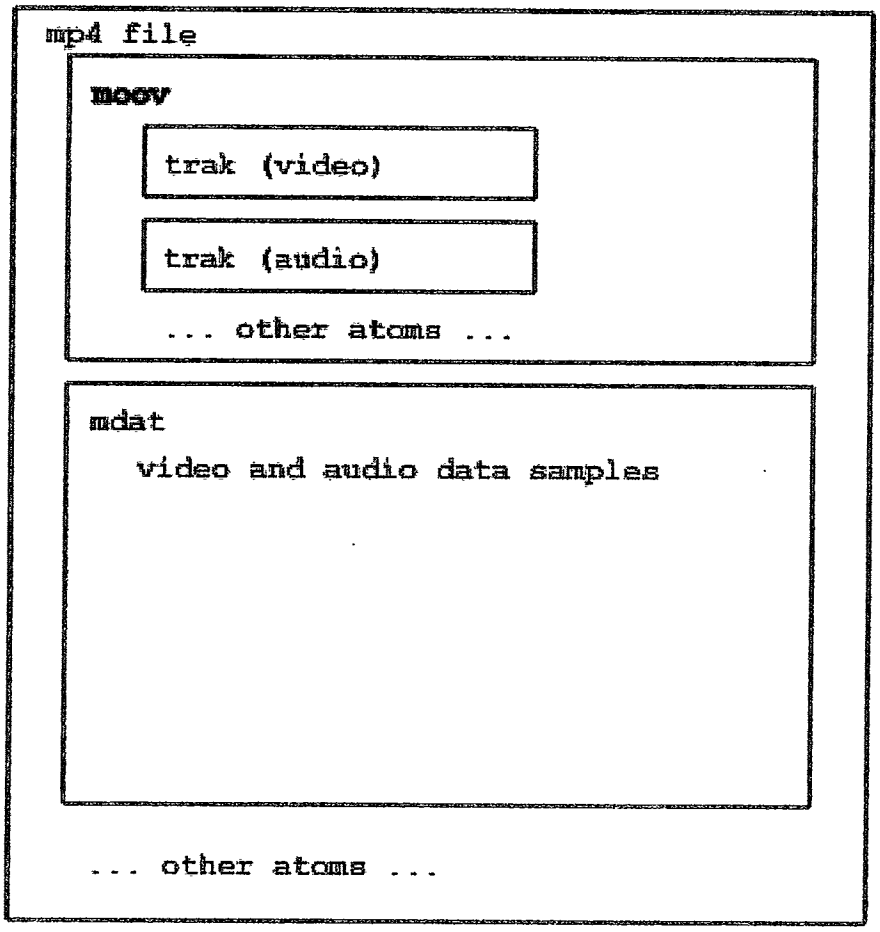
A.3.2
. Are there plans for a GUI frontend for my platform?

There is a Cocoa-based OS X GUI right now. I have no plans on developing a GUI for Windows, Linux, FreeBSD or any other OS, but am willing to accept patches and / or developers to the project if someone has interest in developing them.

Appendix B. How Hymn Works

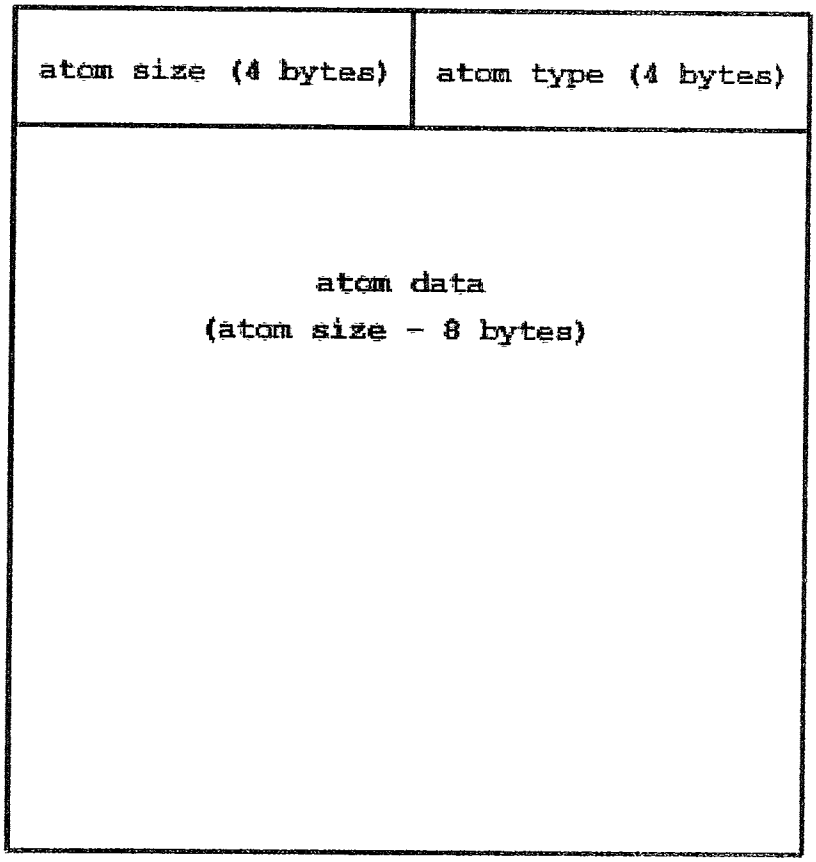
iTunes Music Store files are encoded in the AAC file format. AAC is the audio layer in MPEG-4 files. Apple uses a DRM scheme called "FairPlay" to encrypt the audio data inside of the AAC file. Such files are often referred to as "Protected AAC Files". A Protected AAC file is a viable MPEG-4 file in terms of how the data is arranged within the file. The only difference is that the music data itself is encrypted.

An MPEG-4 file is structured like this:



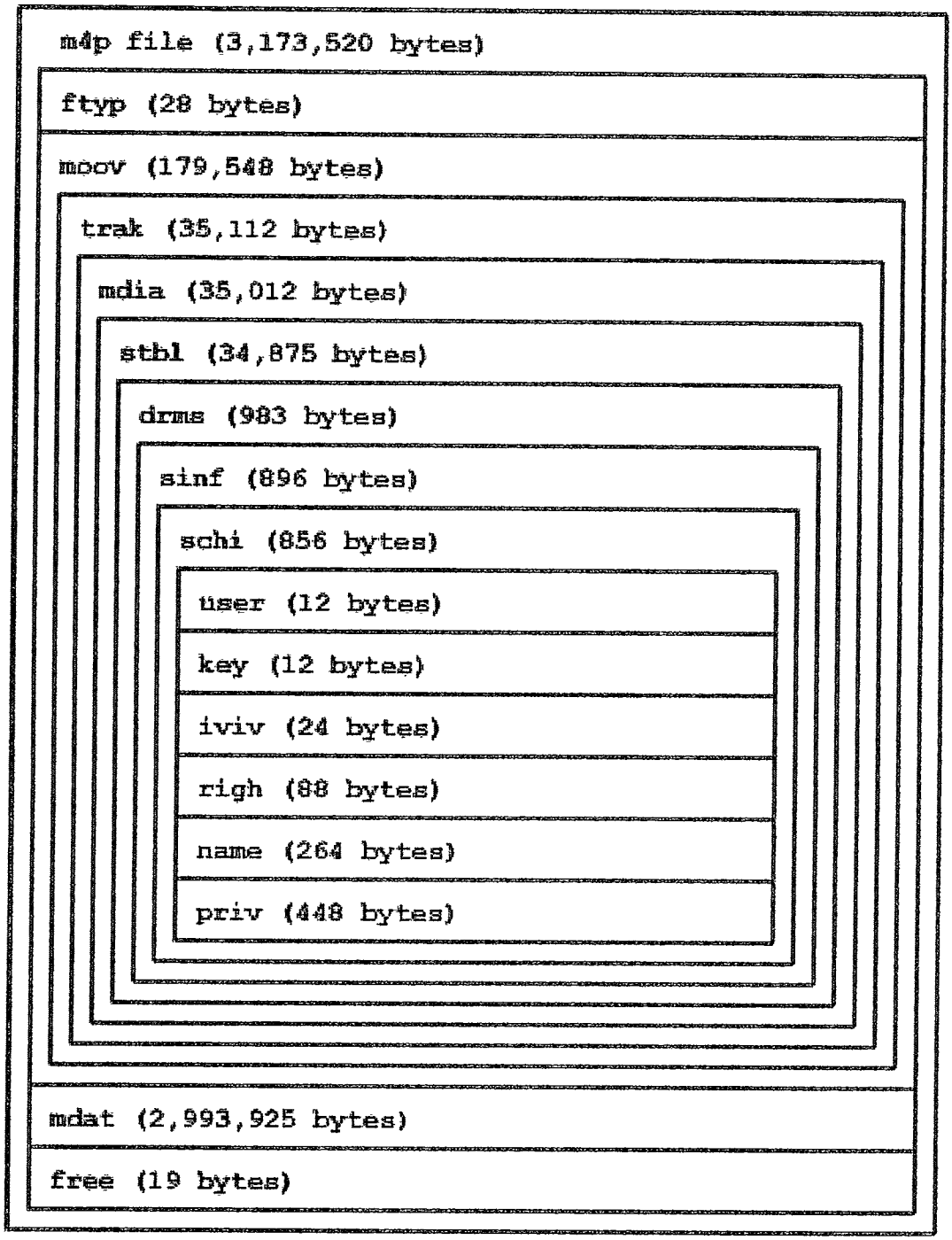
Each MPEG-4 file is a set of nested "atoms". Each atom consists of an 8-byte header followed by the actual atom data. The first four bytes of the header is an integer representing the size of the atom to follow. The next four bytes is the atom type:

atom



The atom type is what is used to determine how to process the atom data itself. Different atom types have differently structured data.

An iTunes Music Store Protected AAC file is structured as shown here. Some atoms have been ommitted for clarity:



Buried deep inside the 'moov' (metadata) atom, there is a drms atom. It is this atom that holds the key to understanding the iTunes Music Store "FairPlay" DRM scheme. FairPlay uses AES (Rijndael) encryption.

- user -- iTunes user ID
- key -- iTunes user key #

- iviv -- AES initialization vector
- name -- iTunes user name
- priv -- AES private key

When a user authorizes a computer to play a song through the iTunes Music Store, iTunes adds the key for that song to the key database stored on the hard drive. Each key in the key database is encrypted using a system key. On Windows, the system key is a hash of items from the registry: Bios Version, Processor Name and Windows Version. The system key hash for Macintosh machines has not yet been cracked. The system key for the key database stored on the iPod (when songs are transferred from iTunes to an iPod) is the iPod hardware ID.

The encryption is three-levels deep. The audio data in a protected AAC file is encrypted with a key and initialization vector that is contained within the 'priv' atom. The 'priv' atom, however, is encrypted using the user key from the user's iTunes key database. The user's key database is also encrypted, using the system key described above. Hymn does the following to try to decrypt a song:

1. Read the iTunes user ID and iTunes user key # from the protected AAC file.
2. Check the key database for the given key (based on user ID / key #).
3. Hash the 'name' and 'iviv' atoms from the protected AAC file to obtain the initialization vector for the 'priv' block.
4. Use the obtained key and hashed initialization vector to decrypt the 'priv' atom within the protected AAC file.
5. Read the key and initialization vector for the audio data from the decrypted 'priv' atom.
6. Use the obtained key and initialization vector from the 'priv' atom to decrypt each sample of the audio data.

If any of these steps fail, hymn will fail to decrypt the file properly.

The AES (Rijndael) encryption algorithm is a published standard. The MPEG-4 file format is also a published standard. The tricky part of figuring out Apple's FairPlay DRM scheme comes with figuring out how to decrypt the user's key database. Jon Lech Johansen (of DeCSS / VideoLAN fame) was the first to figure out how FairPlay works.

Hopefully this brief explanation will help you in some way, either for academic purposes, to better understand the hymn source code or to write your own iTunes Music Store file decryptor.

Appendix C. Law vs. Ethics

Many people want to buy music in a pure digital format. They would rather buy an album or a song online to save themselves the trouble of "ripping" a CD to a compressed digital format. Every time a new technology comes along that will make copying music easier, the people who depend on the revenue stream of music sales accept the technology in three phases:

1. Attempt to fight it through legislation.
2. Attempt to fight it with technology.
3. After (1) and (2) have failed, figure out a way to make money from it

This time around, the phases all got blended together. When the idea of internet music distribution was first popularized with the MP3 format, the recording industry (RIAA) panicked. They lobbied congress to outlaw MP3 players. At the same time, they tried to create technology that would limit the ability to copy such songs (DRM technology). Then, at the same time, they crafted a law called the DMCA that would make it illegal to try to circumvent copy-protection technology. Shortly after that, they began trying to capitalize on the new technology (internet music distribution) in order to create new revenue streams. And at the same time as that, they have started a public relations war calling friends who share music "pirates" as if people are raping and pillaging.

This time around it is different. The DMCA *passed* in the United States. There is a treaty called the WIPO treaty that enforces copyrights across national borders. At the same time, they have managed to convince most consumers that people who share music with their friends are "pirates" and that DRM is in their best interests. It is *not*

I wrote hymn because I, on principal, disagree with DRM. Although I clicked "accept" when presented with Apple Computer's license agreement for the iTunes Music Store, I did not do so because I accepted it. I did it because I wanted to buy music in a high-quality format without the inconvenience of plastic, paper and a bunch of songs I didn't want. Although according to the DMCA I am in the wrong, I do believe that I am *ethically in the right.* I will never "accept" DRM. I will fight it through technology and activism. It is just plain wrong. We, the consumers of music are *not* pirates, criminals, enemies or terrorists, despite what the public relations campaign of the RIAA would have people believe.

Appendix D. Acknowledgements and References

AudioCoding.com	(http://audiocoding.com/) This project is responsible for the FAAD2 library the source code of which I used as a reference while trying to figure out the Protected AAC file format.
MPEG4IP	(http://www.mpeg4ip.net/) This project is responsible for the MP4v2 library that was used by early versions of hymn. It is no longer used.
m4p2mp4	(http://www.techfreaks.org/utilities/m4p2mp4.zip) The source code for the first version of hymn was derived from a Windows-only program called m4p2mp4.
VideoLAN	(http://videolan.org/) This project is responsible for the VLC Media Player that was the first end-user software to include support for decrypting protected AAC files. The code in hymn that reverse-engineers the DRM is kept in sync with the VLC project.
Jon Lech Johansen	(http://nanocrew.net/) Jon is the person who first reverse-engineered the FairPlay DRM scheme. He is more widely known for his work on DeCSS, which was a program that allowed DVDs to be played on unsupported devices and operating systems, such as Linux. He is also a VideoLan developer.
Apple Computer	(http://www.apple.com/) Apple creates excellent products. OS X is the most usable commercial operating system I've ever experienced. The iPod is great little device. The iTunes Music Store is a great way to buy music. Technologically speaking, I am enamored with almost everything they do. Ethically / legally speaking, however, they are (unfortunately) just as bad as most other big corporations.