iDRM: An Analysis of Digital Rights Management for the iTunes Music Store

Aaron Franks, Stephen LaRoy, Mike Wood, and Mike Worth

***Abstract*— Digital rights management (DRM) is necessary to protect the interests of multimedia copyright holders who sell digital copies of their works online. In this report we analyze the design and implementation of FairPlay - Apple's approach to DRM. A survey of this system's weaknesses is presented, along with the most prevalent attacks that exploit them. We discuss simple yet effective measures that improve key management and key distribution protocols to provide much greater protection against such attacks. We also introduce a watermarking scheme that acts as a deterrent against large scale piracy - a feature not present in the current system. The system improvements are evaluated for efficiency and resilience against attack.**

***Index Terms*— digital rights management, watermarking, obfuscation, iTMS, Apple**

I. Introduction

THE need for secure digital rights management (DRM) is more urgent today than ever before. With the rapid increase in broadband availability, Internet file sharing has become a threat to content providers' bottom line. The Motion Picture Association of America (MPAA) estimated in 2004 that eDonkey and FastTrack serve a combined five million users each day [1]. If the average cost of an album is $12.99 with approximately 12 songs per album and if each user only transfers 1 song a day, the estimated cost to the record companies is over five million dollars per day.

To combat this trend, content providers have tried a multi-tiered approach which includes legislation, prosecution and DRM. Legislation successfully brought down Napster and Grokster servers. However, the decentralized peer-to-peer Gnutella system has a rapidly growing user base and a large database of illegally shared content [24]. The direct prosecution of individual users has produced mixed results. A small fraction of existing users abandoned illegal file-sharing for fear of prosecution, however the total number of users continues to grow unabated [24].

The inability to legislate the problem away pushed content providers to consider an alternate approach - to provide a more convenient online experience for consumers. Purchasing digital content with a few mouse clicks and offering it at a lower price than store-bought CDs drastically simplifies the consumer's role. Furthermore, the flexibility to purchase individual tracks from an artist rather than an entire album is a feature unique to online distribution. However, this content must be protected by DRM to prevent users from abusing the system and illegally sharing the purchased media with others.

The most successful of these online content providers is Apple's iTunes Music Store (iTMS) which is expected to sell 170 million files this year [2]. Apple uses a digital rights protection system titled FairPlay to protect content purchased from iTMS on a per user basis [2]. However, hackers were able to reverse engineer the FairPlay system and successfully remove the DRM protection from the audio files. Apple prosecuted and shut down a handful of web sites hosting these exploits, but a few remain publicly available.

The rest of this paper is structured as follows. Section II discusses the design and implementation of the FairPlay DRM system used in iTunes version 4.x and the weaknesses exploited by attackers. Section III discusses our improved design for the DRM system. Section IV discusses social and legal issues related to DRM, both in general and specifically with respect to iTMS. Section V discusses related work and in section VI we conclude.

II. FairPlay

FairPlay-protected content is stored in compressed Advanced Audio Codec (AAC) files that are encrypted using the Advanced Encryption Standard (AES). The AES key and initialization vector are stored in the file's meta data, but the key is encrypted with a *user key* that is randomly generated when the audio file is purchased [4]. The user keys are encrypted with a *system key* and stored in a repository on the computer or iPod. The iPod is the only portable player supported by iTMS.

A central policy is to allow a user's purchased content to be playable on five computers (Windows or Macintosh only) at any given time. Apple's iTMS servers keep a record of the list of computers that are registered to the account as well as the user's keys. When a new computer is registered to a user account, a machine identifier is uploaded to the iTMS servers, and all of the user keys are encrypted and stored on the newly registered machine. Likewise, a computer can be deregistered by having iTunes delete its key repository and remove its machine identifier from the iTMS servers [11]. Communication between iTunes processes and iTMS servers is accomplished using the HTTP protocol.

A. Attacks

There have been several successful attacks on the FairPlay system. Collaborative reverse engineering efforts produced PlayFair, which used the definitions for the system key for the Windows platform and iPod devices. The system key on Windows was determined to be a hash of registry entries for the BIOS version, processor name, and Windows version; for iPods, the system key is simply a hash of the iPod's hardware identification code. At the time of writing, the system key

definition for Macintosh systems has yet to be determined [4], [7].

Jon Johansen, widely known for his work in breaking the content protection scheme used in DVDs, later released FairKeys and DeDRMS tools. The former masquerades as an iTunes process to retrieve the user key database directly from the iTunes server. The latter uses the retrieved key database to decrypt the protected iTMS audio files [11]. Hymn and JHymn are related open-source projects that use the same mechanisms found in the FairKeys and DeDRMS source, that wrap the DRM removal processes with a front-end GUI [13]. This enabled users unfamiliar with command line applications to strip DRM protection from their content.

Two other programs, PyMusique and SharpMusique, allow users to log in to the iTMS and purchase songs without using iTunes. By subverting iTunes, these programs discovered that the iTMS server transmits unprotected song files to the client machine, where DRM protection is added by the iTunes process. Thus, these programs enable the purchase of music from the iTMS that is completely free of DRM protection.

B. Analysis of FairPlay Weaknesses

The most obvious weakness in the implementation of any audio DRM protection scheme is that the user can simply record the analog playback of the protected content legally back to his or her computer (dubbed the “analog hole”). The user is then free to do whatever he or she wishes with the re-recorded content, while having to accept some audio quality loss in the re-encoding. A second exploitation of the analog hole is possible by burning CDs, as FairPlay allows users to burn protected files to a limited number of playlist CDs. These can be re-encoded to unprotected audio files through iTunes or another media player application.

The survey of successful attacks presented above illustrate several technical weaknesses specific to the FairPlay system as well. Despite its three layers of encryption, FairPlay was completely defeated with PlayFair’s discovery of the system key definitions for Windows systems and iPods. This attack demonstrated the central importance of the system key in FairPlay’s design, and showed that Apple’s attempt at defense in depth (by layering encryption) did not provide an appreciable increase in security. The hash used for the Windows platform is also insecure, as many users will end up with the same key. Since the hash only incorporates the system BIOS, OS version, and CPU name, all users with the same combination of these three system parameters will have the same system key.

FairKeys exploited weaknesses in the authentication of iTunes processes. The program uses the device ID of an iPod connected to the user’s machine, but if no iPod is available, FairKeys can simply substitute a random value [9]. Furthermore, the only encryption in the authentication of an iTunes process involves a single HTTP header field. This field consists of an MD5 hash of the request URL, a static user agent string, a static base 64 string, and a nonce that is shipped along with the HTTP request. The use of a standard hash function with a handful of trivial parameters allows malicious processes to authenticate as valid iTunes processes and download an unencrypted user key database. With the key database, the user is then able to decrypt any file purchased from the iTMS.

PyMusique and SharpMusique also take advantage of the weak authentication mechanisms. They further exploit that DRM protection is added on the client machine and not the iTMS servers, enabling the direct purchase of unprotected music.

C. Controversy

The FairPlay DRM system has raised many non-technical concerns as well, mostly related to the concept of fair use. Restrictions on user rights, and not piracy, have been cited as the driving force behind the attacks discussed above [4]. Circumventing the DRM has allowed users to play their legally purchased content on operating systems, media players, and mobile devices that are not supported by FairPlay. Fair use has also continued to be a point of contention for other potential users, who refuse to purchase any music that contains DRM because they view it as being too restrictive.

Vendor lock-in is another major issue with FairPlay, since protected files can only be purchased from Apple’s store and played from Apple’s media player (iTunes) and portable devices (iPods). Apple has declined requests from other companies to license FairPlay, and this position has been defended successfully in court [14]. Furthermore, when RealNetworks enabled their own protected content to be played on iPods with their Harmony technology, Apple responded by issuing firmware updates that changed the way the iPod plays back protected content, effectively disabling Harmony [11]. The fact that only Apple products can distribute and play FairPlay content has continued to be a major problem for many other companies that have had difficulty competing with the hugely popular iPod and iTMS without being able to offer interoperability.

While these issues result in friction between content providers and consumers, their solution is beyond the scope of this paper.

III. iDRM

In this section, we present two orthogonal mechanisms that improve the security of FairPlay and strengthen the deterrence of large scale piracy on legally purchased material, respectively.

A. Key Management

In this section we propose a key generation algorithm to replace the existing definition of the per-user-device system key. Our proposed algorithm seeks to increase the effort required to retrieve this key via brute-force and reverse engineering attacks. We also discuss how authentication processes and user key distribution improvements better defend against masquerading attacks such as that of FairKeys.

We note that the requirement to provide offline access to DRM protected content forces the security of the system to be

dependent on the secrecy of the mechanism itself. For offline access, the client machine must possess all of the information (code and data) necessary to decrypt and access protected content. Regardless of how this information is stored, whether embedded in application(s), encrypted, obfuscated, or even divided across different physical media, it must all still be accessible by the client machine to ensure the offline access requirement. The security of the system then depends on the client's ignorance of the mechanism that uses this information to provide access to the DRM protected content. Thus, the system cannot exercise the Principle of Open Design [5] and must rely on the secrecy of the mechanism.

1) System Key Algorithm: The key generation algorithm requires a larger input space from which to derive the system key. It must also provide greater variability between the system keys of different users. These goals are accomplished with the inclusion of a user-specific random value and the user's Apple ID as key generation material. Both the random value and Apple ID must be stored in plaintext on the client machine, but this is no different from all other system properties which must be available in the clear. The guaranteed uniqueness of an Apple ID across the entire user space combined with the probabilistic uniqueness of a random value provide sufficient variability between the key generation input for different users.

Permuting the bits of the key generation material further strengthens the key's definition. This requires a random-looking permutation of the bits that can be computed deterministically. This type of pseudo-randomness can be accomplished using a universal hash function [6]. Such hash functions take the form

$$h(x, n) = ((ax + b) \bmod p) \bmod n$$

where p is a prime, $a \in \mathbb{Z}_p^*, b \in \mathbb{Z}_p$ and $p > n$. This class of hash functions has the property that for any x, y in the key space and any randomly chosen function $h \in \mathbb{H}$, the probability of $h(x) = h(y)$ is $1/n$.

We defined the following permutation function to operate on bytes, where h is a universal hash function. An important property of this permutation function is that the original distribution of bits (the number of zeros and ones) is the same in the permuted stream as in the input.

```
BYTE-PERM( bytes, len, h )
  n = len * 8
  for ( int i=0; i < n ; i+=8 ) do
    j = h( i, n )
    tmp = bytes[i/8]
    bytes[i/8] = CSHIFT( bytes[j/8], j%8 )
    bytes[j/8] = CSHIFT( tmp, j%8 )
```

Three experiments were performed to evaluate the randomness introduced with this permutation. Table I shows the distribution of bits of the input, the number of trials for each experiment, and the expected maximum randomness (`chars` are single ASCII characters, `ids` are hand crafted examples of key generation input, and `dict` is the /usr/share/dict/words file). Each input trial (or file line) was separately permuted with `BYTE-PERM` using two universal hash functions, with $n = 512$ and $n = 1024$. Table II shows the permutation nearly matches the expected randomness for the `chars` and `ids` experiments and achieves close to the expected value for the `dict` experiment.

TABLE I

INPUT USED TO EVALUATE RANDOMNESS

name	*% zeros* $(p_z * 100)$	*% ones* $(p_o * 100)$	*trials*	*max % randomness* $(2 * p_z * p_o * 100)$
random	50.00	50.00	-	50.00
`chars`	52.27	47.73	88	49.90
`ids`	50.37	49.63	5	50.00
`dict`	47.26	52.74	234937	49.85

TABLE II

`BYTE-PERM` RANDOMNESS.

	`chars`	`ids`	`dict`
perm (512)	49.72	50.07	46.75
perm (1024)	49.72	49.05	43.58

The deterministic sequence of swapped indices in the permutation algorithm is dependent on the bit length (n). Thus, key generation input of different lengths will have different bytes swapped. This introduces further benefit to increasing the variability of key generation material between users.

The permutation of the input bits serves as the input to a standard hash function - one of the various SHA flavours. These functions are part of the Secure Hash Standard as defined by NIST and provide up to 256 bits of security [8]. Collision resistance and no input inference properties ensure the permuted input is well disguised from the output, and that it is nearly impossible for an attacker to find some other input to produce the same hash.

Code obfuscation is required to mitigate the threat of reverse engineering techniques that may expose the mechanisms in place. Since we are particularly concerned with masking the key generation algorithm, it is most crucial to disrupt the disassembly of the iTunes binaries, rather than the de-compilation of assembly to a higher level language. Techniques to thwart static disassembly are proposed in [10], in which inactive "junk bytes" are strategically placed to introduce errors in the derivation of assembly instructions from machine code.

2) Performance Analysis: The overhead to include an additional permutation step in the computation of the system key is negligible with respect to the total system load. Using the implementation of SHA-1 in openssl for Max OS X version 10.3.9, we compared the cost of key generation with and without the byte permutation described above. The code was written in C and compiled using gcc version 3.3. Each line in `/usr/share/dict/words` was used as the key generation material, which totals to 234,937 trials. Profiling with gprof indicated that for both runs the SHA-1 routines were the bottleneck and that the permutation function is less than half the cost of SHA-1 (see table III for results).

The code obfuscation process was evaluated using the SPECint95 benchmark and was found to add an average of

TABLE III

EVALUATION OF KEY GENERATION PROCEDURE

function	*exec time*	*normalized*
SHA-1	.895	1.00
`BYTE-PERM`	.447	0.499

thirteen percent to the execution time of unobfuscated code [10]. SPECint95 is a well recognized standard to measure the performance of CPU bound workloads. As encryption, decryption and hash function computations are CPU bound as well, this suggests obfuscated versions of our permutation are likely to experience an equally minor slowdown.

Lastly, the per-byte cost of SHA-1 and AES are nearly equal [25]. Thus, the additional overhead incurred by the key generation algorithm is insignificant compared to the AES algorithm, which processes megabytes of data in the decryption of protected content.

3) Authentication and Key Distribution: A challenge-response protocol will strengthen the authentication of iTunes clients with the iTMS server. We propose the shared secret in this protocol be a similar algorithm to the key generation algorithm described above, requiring a proprietary permutation followed by a standard hash function computation. Assuming the code obfuscation is effective in maintaining the secrecy of this process, this better defends against masquerading attacks such as FairKeys and PyMusique, as the iTMS server can be more certain it is communicating with a true iTunes process.

We presume Apple has performed their own internal cost-benefit analysis with regards to the application of DRM on the client machine. This analysis likely indicates the cost of applying DRM at the iTMS servers outweighs the risk of users illegitimately purchasing unprotected content. Although our solution does not directly mitigate this risk, stronger authentication prevents illegitimate processes from gaining access to the iTMS server.

B. Watermarking

As long as a user has possession of a media file, it will be nearly impossible to enforce copy protection. Therefore, since we cannot prevent iTMS content from being copied, we wish to provide a deterrence from doing so. We propose embedding an identifying mark into the song to track (and possibly prosecute) those who violate the iTMS usage agreement by removing the encryption and distributing copyrighted media purchased from iTunes. Obviously, this data cannot be stored in a header or separate data stream or it may easily be removed [15]. This method emphasizes detection and recovery over prevention.

Therefore, we propose embedding a watermark into the audio stream. This watermark consists of a secure modern hash of the following: iTunes user name, song name, artist name, download details (time, price, etc.), and a random number to ensure security. This hash is encrypted using a key belonging to the iTMS. Once properly decrypted, a court can be certain that the iTMS music store embedded the watermark and that the data corresponds to the purchase details of the song. In this way, a copy of a song can be uniquely identified as belonging to a particular user. Thus, in the event of large scale piracy, Apple and only Apple can prove in court that a particular song was widely distributed by a particular individual. This enables easier prosecution of file sharers, thereby deterring unauthorized distribution through anonymous P2P file sharing networks. The user's information is never compromised since the watermark consists of an encrypted hash, and it is only recoverable by Apple, who already has all their details.

To make this system work, it must satisfy a number of difficult and sometimes conflicting criteria [20][16][19] [18][17]. The watermark must be

1) embedded in the host media.
2) statistically undetectable. This will prevent unauthorized detection and removal of the watermark.
3) perceptually inaudible within the host media. Watermarked and unwatermarked media must be indistinguishable to the listener.
4) robust against manipulation and signal processing such as noise, time scaling, random and fixed length cropping, compression and decompression, filtering, re-sampling, D-to-A conversion and format conversion. This will prevent an attacker from destroying the watermark or making it unreliable (and therefore legally useless for prosecution). The watermark must be impossible to defeat without destroying the audio.
5) readily extracted to completely identify the media purchaser.

There are many schemes that provide protection against most of the attacks listed above. However, most are not self-synchronizing, and are not robust against random cropping of samples in the middle of the song. We identified one method proposed by IBM researchers that has been demonstrably resilient against these attacks [18], and is especially suited for iTMS since it is able to embed the watermark into AAC compressed media [17], saving much computational resources.

To ensure that the signal is imperceptible, its amplitude must vary with the input media signal. This imperceptibility is possible because of masking: the human ear filters sounds it hears, and certain sounds are masked by other sounds that are close in frequency and time to the masked sound. Psycho-acoustic models are used [16][20] to determine the imperceptibility levels of the watermark signal. The Power Spectral Density of an imperceptible watermark can be seen in [20].

We have taken Ryuki et al's algorithm to embed the watermark and modified it to meet the needs of the iTMS watermarking scheme:

- Divide the signals DFT representation (Frequency components at different time samples) up into message blocks.
- Subdivide message blocks into tiles. Each tile is four AAC frames long. Each time slice has multiple tiles representing multiple frequency sub-bands. Each tile corresponds to a data bit or a synchronization bit for the watermark.
- Map a pseudo-random array of +1/-1 onto the tiles. The pseudo-random array is unique to the song, but the same for every copy of the song sold. This array serves as one secret key for the watermark.
- Decompose each tile into four frames for each bit, where the first two frames have one polarity, and the next two frames have the opposite polarity.
- Calculate the watermark message, and encode it using an

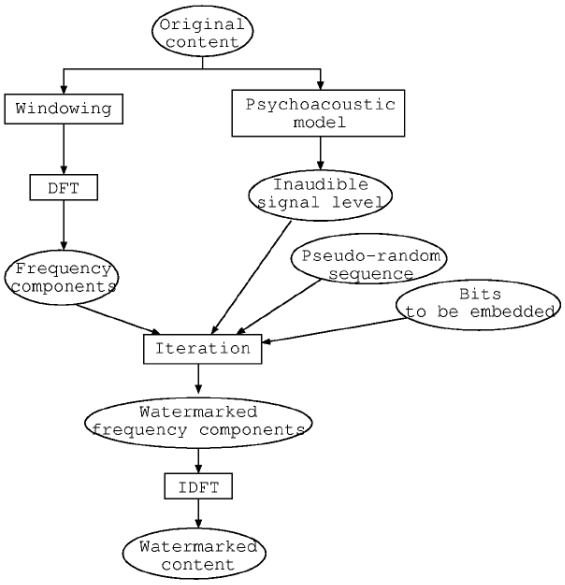

Fig. 1. Flow Chart of AAC Watermark Encoding Scheme [21]

error-correcting turbo code.

- Modulate the pseudo-random array bits with sync and watermark bits, multiplied by the appropriate amplitude as determined by the psycho-acoustic model of sound masking. This determines the degree of modification for the DCT coefficients. [1]To avoid clicking sounds at the borders of frames, the watermark signal is multiplied by a windowing function and overlapped with neighbouring frames.
- Change the Modified Discrete Cosine Transform coefficients by the amount determined in the previous step.
- The watermarking process is repeated for multiple blocks in the file for redundancy.

This is shown in Figure 1. Note that in the AAC file, the DFT and windowing has already been performed.

In detection, the pseudo-random array used for the embedding is multiplied with the normalized frequency components from each AAC frame, and the watermark vector of bits is detected. Bit decisions are made by comparing the vector with thresholds [21].

1) Analysis of Watermarking Scheme: Obviously, a great deal of detail in the chosen watermarking scheme has been omitted. Much more can be found here [18][17][21] [22].

Although the proposed watermarking scheme meets our requirements for undetectibility and robustness against attack, it may be replaced at any time by a superior algorithm. This is useful if a successful attack against this watermarking technique is found. However, since the playability of a media file is never noticeably affected by the watermark, the watermark scheme can be updated, since its use is to detect piracy, not thwart it.

The watermarking scheme is better than a copy-protection scheme in that the keys (the pseudo-random array and the key used to sign the hash) used to encrypt the data are never in the hands of the end users. Even if the users did manage to find out the pseudo-random array, the signed watermark remains secure. However, all watermarking schemes are vulnerable to arbitrary large collusion attacks [19], i.e. many users share their copies of the song, average out the differences and remove the watermark.

This scheme is robust against a variety of signal transformations: echo, pitch shifting, conversions between formats (eg. MiniDisc, MP3), D/A and A/D conversion, stretching and cropping, and random noise [18]. There are also other detector methods used to improve robustness against stretching [22].

Even in the presence of malicious tampering with the file causing bit errors in the watermark signal, the turbo codes and redundancy ensure that the message will almost always be reconstructed in the presence of attacks that do not render the file unbearable to listen to. Most attacks that remove the watermark will distort the audio too much and not be successful. However, there is still a finite probability that even an uninformed attack can remove the watermark successfully [19]. Moreover, the attacker has no way of determining success since he has no access to the original watermark message or the pseudo-random array.

False positives (detecting a watermark that isn't there) are extremely unlikely in our system because the detector relies on both the original content and the pseudo-random array.

In the language of secure design principles, watermarking does not address Least Privilege, Fail-Safe Defaults, Complete Mediation, Defense in Depth or Separation of Privilege, since watermarking does not attempt to enforce usage rights. However, it is psychologically acceptable because it is imperceptible and does not expose user's personal information. It is an open design in that the watermarking method may be published, and security depends on keys and random numbers unknown to users. There is no common mechanism since the decoders are not included in the iTunes program, so they cannot be compromised. This system allows us to continually question our assumptions about what kinds of watermarking may be broken, and it is easy to upgrade the iTMS to introduce new, more robust watermarks as they are developed.

IV. Legal Issues Surrounding iTMS

The laws regarding digital media lack specificity, leaving open the interpretation of legal versus illegal actions with respect to the control and/or use of digital content. The Digital Millennium Copyright Act (DMCA) in the United States (US) seeks to protect the interests of content distributors, rendering illegal even the attempt to compromise protection mechanisms [3]. Unfortunately, this legislation is so broad that it is argued that it impedes research and stifles new innovations [3]. Moreover, it infringes upon consumer rights as set out in the Fair Use doctrine. Fair Use in the US dictates that legally purchased media may be used for research, teaching (with the exception of distance education), criticism, review or news reporting [3]. It also permits the resale of purchased media and the creation of backups for personal use. However, Fair Use also suffers from an imprecise definition, forcing the balance

[1]Computing the psycho-acoustic model is costly and may need to be done in the time domain, which negates the advantage of this method that saved computation by doing the watermarking directly in the compressed AAC file.

of power between DMCA and Fair Use to be determined on a case-by-case basis in court.

The limitations imposed by FairPlay fail to meet the requirements of the Fair Use doctrine in US copyright law. For this reason, many users feel justified in circumventing the copy restrictions imposed by the iTMS [4]. Apple's legal team successfully forced the removal of PlayFair and associated DRM removal software from their host web sites [11], through cease and desist orders referencing DMCA legislation. By making the system less susceptible to attacks, our improvement to strengthen key management only maintains the tension in the debate: making it more difficult to remove the DRM protection simultaneously improves copyright protection and reduces the ability of the user to use his purchased media according to his Fair Use rights.

Another consumer issue regarding DRM is that of privacy and anonymity. By requiring the user to sign into a service before listening to audio, it is possible to monitor a user's consumption preferences and build a profile of his or her activities [3]. This is a common point made by civil rights groups against the implementation of DRM technology.

Our implementation of watermarking does not affect privacy since the watermarks are only read by Apple from files that are shared on P2P networks. Moreover, since Apple is the only one with the pseudo-random array, encryption key, hash function and input information, no one can extract user details from the watermark. Also, play counts and other user information are not tracked using the watermark, since a watermark decoder is not included in the iTunes player.

V. Related Work

TiVo, a producer of digital television recording systems and equipment, are incorporating a similar watermarking technique for programs destined for iPod or PSP devices [23]. This feature is currently in a trial stage with a select group of TiVo subscribers, with production implementation expected early next year.

The release of iTunes version 6 in October 2005 includes an improved (or modified) implementation for FairPlay. This has rendered many of the DRM circumvention programs useless for more recent versions of iTunes. The JHymn web site notes that the program will not work with iTunes versions 6.0 or later, since the program has yet to "learn how to perform the iTunes 6.0 protocol." [12] We speculate the plethora of attacks on FairPlay motivated Apple to strengthen the system in ways similar to those presented in this paper.

VI. Conclusion

The protection of digital content to prevent and deter large scale piracy is a necessity, especially with the growing ubiquity of digital media players and devices. Apple's initial protection mechanisms in FairPlay suffered from considerable weaknesses, leaving the content vulnerable to the removal of protection information.

Providing users with offline access to DRM protected content necessitates the secrecy of the mechanisms used to uncover the protected media. Several attacks took advantage of blatant security flaws in the design of Apple's FairPlay DRM system. Key generation and key management protocols can easily and effectively be hardened using proprietary permutation routines in conjunction with standard cryptographic algorithms to thwart such attacks. Watermarking songs with the purchaser's identity using a tamper-resistant scheme provides a psychological barrier against widespread piracy.

References

[1] MPAA. "Illegal Downloading". *http://www.mpaa.org/CurrentReleases/2004_11_04_Statistics.pdf*

[2] Gibson, Brad. "TMO Reports - Analysts See 3.3 Million iTMS Sales As Download Explosion". *http://www.macobserver.com/article/2004/05/05.13.shtml*

[3] Liu, Qiong. Safavi-Naini, Reihaneh. Sheppard, Nicholas Paul. "Digital Rights Management for Content Distribution". Univeristy of Wollongong. *page 8.*

[4] Anonymous. "Hymn Manual". *http://hymn-project.org/docs/hymn-manual.pdf*

[5] Bishop, Matt. "Introduction to Computer Security". Pearson Education, Inc. December 2004. *pages 204.*

[6] Cormen, Thomas H., Leiserson, Charles E., Rivest, Ronald L., Stein, Clifford. "Introduction to Algorithms". MIT Press 2002.

[7] Fisher, Ken. "Apple's FairPlay DRM cracked". *http://arstechnica.com/news/posts/1081206124.html*

[8] NIST. "FIPS 180-2: Secure Hash Standard". August 1 2002. *http://csrc.nist.gov/publications/fips/index.html*

[9] Johanson, Jon L. "FairKeys". C-Sharp program. *http://www.nanocrew.net/* Accessed on: Oct 4th, 2005.

[10] Linn, Cullen. Debray, Saumya. "Obfuscation of Executable Code to Improve Resistance to Static Disassembly". CCS'03, October 27-31, 2003.

[11] Anonymous. "FairPlay". *http://en.wikipedia.org/wiki/FairPlay*

[12] Anonymous. "JHymn Info and Help". *http://hymn-project.org/jhymndoc/*

[13] Anonymous. "JHymn - Frequently Asked Questions". *http://hymn-project.org/jhymndoc/jhymn_faq.php*

[14] Betteridge, Ian. "Court Won't Force Apple to License DRM". *http://www.eweek.com/article2/0,1759,1725747,00.asp?kc=EWRSS03119TX1K0000594* November 12, 2004

[15] L. Boney, A. Tewfik, K HJamdy. "Digital Watermarks for Audio Signals". Proceedings of the Third IEEE International Conference on Multimedia Computing and Systems, 1996.

[16] M. Swanson, B. Zhu, A. Tewfik, L. Boney. "Robust audio watermarking using perceptual masking". Signal Processing 66(1998) 337-355

[17] Ryukia Tachibana. "Two-Dimensional Audio Watermark for MPEG AAC Audio". SPIE-IS&T, Vol. 5306, 2004

[18] R. Tachibana, S. Shimizu, S. Kobayashi, T. Nakamura. "An audio watermarking method using a two-dimensional pseudo-random array". Signal Processing 82(2002), 1455-1569

[19] R. Sion, M. Atallah. "Attacking Digital Watermarks". SPIE-IS&T, Vol 5306, 2004

[20] P. Bassia, I. Pitas, N. Nikolaidis. "Robust Audio Watermarking in the Time Domain". IEEE Transactions on Multimedia, Vol 3, No. 2, 2001

[21] T. Nakamura, R. Tachibana, S. Kobayashi. "Statistical Model and Experiment of Reliability in Detecting Multi-bit Watermark". IBM Research Report, *http://www.trl.ibm.com/people/taiga/pdf/RT0367.pdf*

[22] R. Tachibana. "Improving Audiowatermark Robustness Using Stretched Patterns Against Geometric Distortion". Proc. of the 3rd IEEE Pacific-Rim Conference on Multimedia (PCM2002), pp. 647-654

[23] "TiVo To Bring TV Programming To Apple Video iPod™ and PSP™". *http://www.tivo.com/cms_static/press_66.html*

[24] Mennecke, Thomas. "Media Metrix Depicts Rapid Kazaa Decline". *http://www.slyck.com/news.php?story=729*

[25] Heinlein, Paul. "OpenSSL Command-Line HOWTO". *http://madboa.com* November 23, 2005.