A Novel Privacy and Copyright Protection Enforced Peer-to-Peer Network

Xiaoming Wang[1], Bin Zhu[2], and Shipeng Li[2]

[1] Dept. of Computer Science, Jinan Univ., Guangzhou, 510632, China
twxm@jnu.edu.cn
[2] Microsoft Research Asia, Beijing, 100080, China
{binzhu, spli}@microsoft.com

Abstract. Peer-to-peer (P2P) networks are widely used to share copyrighted contents and software illegally around the world. Development and applications of P2P technologies have faced strong legal and legislative challenges. A P2P network has to have built-in copyright protection to enable P2P technologies to advance on its own without court's and legislature's interference. In this paper, we propose a novel and practical P2P network with a strong privacy protection and reliable tracking mechanism to track the original uploader of any material in the P2P network. When a pirated material is found, its uploader is tracked down, and can be punished by revoking the access to the network, removing his or her uploaded materials, etc. The whole protection system is completely transparent to end users. The proposed scheme would effectively deter users from uploading illegally any copyrighted materials to and dramatically reduce copyrighted materials shared through our P2P network.

1 Introduction

Introduction and proliferation of peer-to-peer (P2P) software have facilitated a large-scale piracy among networked computer users. We have witnessed a dramatic increase in using P2P software such as Kazaa [1] and BitTorrent [2] to share files by hundreds of thousands of users throughout the world. In addition to legitimate usage, an individual can easily use the same P2P software to illegally exchange copyrighted digital commodities such as digital multimedia and software with another he or she has never known or met. This empowerment of large-scale piracy by P2P networks has seriously infringed the interests of copyright holders. Recording Industry Association of America (RIAA) has mounted an anti-piracy war against P2P networks. The first generation of P2P networks such as Napster and Scour that used centralized servers to list contents available among peers was a natural target for RIAA. Those P2P networks were successfully shut down by RIAA as a result of the lawsuits. A direct consequence is accelerated development and adoption of completely decentralized P2P networks such as Gnutella [3] and FreeNet [4], which are much more difficult to shut down by lawsuits since there is no centralized service provider for illegal sharing. Although P2P networks are widely used for legitimate purposes, a recent ruling on June 27, 2005 by the U. S. Supreme Court in MGM v. Grokster held that "one who distributes a device with the object of promoting its use to infringe copyright, as shown by clear expression or other affirmative steps taken to foster infringement, is liable for the resulting acts of infringement by third parties" [5]. According to this ruling, the providers of software that designed to enable "file-sharing" of copyrighted works may be held liable for the copyright infringement that takes place using that software. In addition to lawsuits against P2P software and service providers as well as file-swapping individuals, RIAA has also been lobbying the US legislation to act with tough bills aiming at P2P technologies and service providers.

US Congress responded with several unsuccessful tries to pass bills targeted at P2P networks. In July 2002, US Representative H. Berman introduced a controversial bill H. R. 5211 [6] granting copyright holders immunity for hacking into personal computers (PCs) and P2P networks in thwarting piracy on P2P networks. In June 2004, US Senator O. Hatch introduced a bill S. 2560 [7] that would hold technology companies liable for creating products that could be used to pirate digital content, and therefore would effectively ban all P2P networks. Those bills, while intending to cure the widespread piracy of copyrighted materials, would severely impair advances and applications of P2P technologies that can be used by law-abiding users for many legitimate purposes. The lawsuits against P2P networks and the US legislation anti-piracy bills have shown a clear pattern: technologists are held liable for the activities of their end-users. The brief but dramatic history of Napster and other file-sharing services and the current debating in Congress underscore an important issue that we have largely ignored so far in the development of P2P technologies: how to protect intellectual property (IP) from illegal distribution in a P2P network so that the technology can advance on its own without court's and legislature's interference.

A major effort in the past decade or so has been directed to develop copyright protection technologies such as digital rights management (DRM) and watermarking to fight against piracy of digital assets. These technologies have also been applied to develop law-abiding P2P networks. A typical example is the P2P network to share music proposed in [8] which uses watermarking, fingerprinting, etc. to prevent pirated digital music from entering and sharing through a P2P network, and to ensure that a user can access and play only those music files that he or she is entitled to. These a priori approaches are based on immature technologies and such a system is very complex to implement and operate. A viable alternative approach is the a posteriori technology which relies on a tracking mechanism in a P2P network to find out and punish uploaders of pirated materials. Bakker et al. [9] have used this approach to build a P2P network called globe distribution network (GDN) for efficient free-software distribution. The law-abiding P2P network we are going to propose in this paper also adopts this a posteriori approach. Our system is called the privacy- and copyright-protected peer-to-peer network (PCPN) which is based on proven and widely used cryptographic primitives and technologies, and is therefore a compromised and viable solution to the current debate whether to shut down all P2P networks by passing some harsh laws or to tolerate rampant piracy through P2P networks. PCPN uses a novel and secure hardware-bound tracking mechanism that can reliably trace back to uploaders of pirated materials yet protect uploaders' privacy. This is very different from GDN which sacrifices an uploader's privacy in tracking software uploaders. GDN requests an uploader to provide passport or other sensitive private information before gaining a permission to upload a copy of software to GDN. This scheme is intrusive and impractical. Unlike GDN, there is no need to provide any sensitive private information to upload anything to PCPN. Operations of the protection mechanism in PCPN are completely transparent to an end user and fully decentralized. Once a peer is convicted of uploading pirated, malicious, or illicit materials to PCPN, all the materials the peer has uploaded can be removed from the network, the peer's access to PCPN can be permanently revoked, and legal actions may be taken against the illegal user. We argue that PCPN will meet the anti-piracy requirements sought by RIAA and the legislation yet allow law-abiding users to use it transparently in any legitimate applications. We would like to emphasize that our goal is not to stop illegal sharing of pirated contents through *all* P2P networks, which is a mission impossible. Our goal is to incorporate an anti-piracy mechanism in our P2P network to deter users from using it for illegal applications so that our P2P network can be used for legitimate applications without any legislative or legal interference.

This paper is organized as follows. In Section 2 we first review the existing IP protection technologies and their inadequacy in fighting piracy in P2P networks. PCPN is then described in detail in Section 3. We conclude the paper in Section 4 with future research we plan to do.

2 Related Protection Technologies

In this section, we briefly review existing IP protection technologies and their inadequacies in preventing copyrighted materials from illegal distribution in P2P networks. One protection technology is conditional access used in satellite TV and other applications that imposes restriction on access to protected contents to only the users who have subscribed a premium service. Conditional access cannot stop a subscriber from uploading and distributing protected materials in a P2P network. A more sophisticated protection system is the DRM system that regulates a user's rights for a protected content and ensures that the rights are observed throughout the life of the content. A DRM system is typically based on encryption of the content. Such a DRM protection can be easily bypassed by recording a protected content digitally or through digital and analog conversions. The recorded content which is free of DRM-protection can then be uploaded to and distributed through P2P networks. Current consumer recording devices and coding technologies can deliver a high-fidelity recording and an efficiently compressed multimedia file very easily for Internet distribution. Both protection technologies fall into the "breaking once, breaking all" scenario that only one person with needed privilege, expertise, and equipment is needed to break the whole protection system when P2P networks enter the picture. Although the US law of the Digital Millennium Copyright Act (DMCA) [10] prohibits anybody from getting around a content protection mechanism, and the copyright law prohibits illegal sharing copyrighted materials with others. They are very difficult to enforce, not to mention that one country's laws may not be applicable to computer users in other countries.

Another type of protection technologies is robust watermarking. Robust watermarking is a technology to embed an imperceptible mark in the content that is difficult or impossible to remove or fake. Two types of watermarking can be used for copyright protection: global watermarking and individualized watermarking (also referred to as watermarking and fingerprinting, respectively). Global watermarking embeds a mark to content to indicate its copyright owner and allowed actions such as copy-once or no-copy. Typical efforts to apply this type of watermarking to content protection are the Secure Digital Music Initiative (SDMI) [11] for music protection and the Copy-Protection Technical Working Group (CPTWG) [12] for video protection. Individualized watermarking, on the other hand, embeds a unique ID for each sale or instance of content so any pirated copy of the content can be traced back to its original buyer. Both types of watermarking suffer from various vulnerabilities. The major vulnerability of a watermarking protection is its weakness against intentional attacks. We have yet seen a watermark embedding with reasonable detection complexity that can survive an intentional attack to strip off the embedded mark or to render the mark unreadable while maintaining acceptable quality. Both watermarking technologies have yet shown effectiveness in fighting against piracy in real life. Some researchers even argue that such a technology may remain to be a dream forever. Additional vulnerability for the individualized watermarking is the simple but very effective collusion attack that multiple copies of the same content with different marks are combined together by some methods such as simple averaging to fake a new mark or make the embedded mark undetectable. All proposed individualized-watermarking schemes can survive at most a small-scale collusion attack at the expense of a dramatically reduced payload. With all those known vulnerabilities and ineffectiveness, the entertainment industry and the legislation still advance the idea of mandatory checking of copyright-indicating watermarks on every incoming bit for copyright protection. The bill S. 2048 [13] proposed by US Senator F. Hollings is a recent step towards that direction. In addition to technology unreadiness, such a dramatic measure may also have unintended consequences such as an invasion of users' privacy, and dramatically slowing down data communications. We can similarly argue that the copyright-protected P2P systems such as the one described in [8] based on watermarking technologies may not be practical in quite a few years.

GDN proposed by Bakker et al. [9] uses a completely different approach to get rid of pirated materials from a P2P network. Instead of preventing pirated software from entering a P2P network, GDN uses a cryptographically signed certificate attached to uploaded software to track the original uploader of pirated software. When a GDN user wants to publish software in GDN, he or she has to contact one of the access-granting organizations (AGOs) to apply for a tracking certificate. An AGO verifies an applicant's passport or other means of identification and checks against a blacklist of banned users with all the AGOs. If this checking step is fine, the AGO issues the applicant an AGO-signed certificate linking the applicant's identity to the applicant-supplied public key. This certificate allows the candidate to upload software into GDN. Without a valid certificate issued by an AGO, a user cannot publish anything to GDN. The certificate is attached to the software. When pirated software is detected, the attached certificate is retrieved, and its uploader's publishing right is revoked by placing him or her to the list of banned users maintained by AGOs. All the software published by the uploader is also removed from GDN. The GDN's protection mechanism seems to be a good deterring tool to potential pirates who would like to use GDN to distribute pirated software, but the tracking mechanism is not very practical. It is very cumbersome for a user to publish software to GDN since he or she has to contact a server, i.e., an AGO, and provide his or her passport or other sensitive private information that the applicant may be unwilling to disclose. It is hard to imagine that a user is willing to surrender sensitive private information to a P2P network server in exchange for a right to publish something to the network. Checking authenticity of supplied passport or other means of identification is time-consuming and expensive. It is impractical for an AGO to check if the supplied passport is authentic and really matches the applicant for every GDN user who wants to start publishing software to GDN. In addition, the certificate signed by an AGO may contain sensitive personal information to identify the uploader, which is a severe invasion to the uploader's privacy since everybody can read the information. Even if the certificate does not contain any sensitive private information, a user can still collect statistics to find out who published what, which is also a privacy invasion. To address the privacy issue, AGOs have to store the supplied sensitive private information associated with each issued certificate so that an illegal uploader can be properly identified when a user applies for a tracking certificate, a big burden to AGOs.

3 Our Privacy- and Copyright-Protected P2P Network

Like GDN, we use the a posteriori approach in our PCPN to be described in this section. We believe that the a prior approach that prevents pirated materials from entering and distributing in a P2P network is too complicated and expensive, and is beyond what the current and near-future's technologies can deliver. Our system is based on robust and proven technologies, and is therefore easy to implement and operate. PCPN relies on a novel and secure hardware-bound tracking mechanism to find out uploaders of pirated or illicit materials in PCPN. Each digital asset uploaded to PCPN is attached with persistent metadata which contains an uploader-signed certificate used to track the original uploader. Authenticity and validity of the certificate and the associated material are verified when a digital material is uploaded to PCPN, or replicated from one client to another. A major design principle for PCPN is the assumption that an uploader is liable to whatever he or she uploads to PCPN. Every PCPN end user is entitled to publish anything in PCPN, and to remain anonymous until a pirated, illicit, or malicious material is found. The tracking mechanism is subsequently invoked to track down the original uploader of the material. Once convicted, the materials uploaded by an illegal uploader can be removed from PCPN, and the uploader is punished in several possible ways ranging from permanently revoking the access or publishing privilege to PCPN to legal actions. A revocation list contains a list of convicted illegal peers along with the action specifications taken by PCPN against these peers which ranges from banning access to or publishing in PCPN, removing materials uploaded, etc. The action specifications can be as fine as specific actions for each individual peer in the revocation list. We believe that such a severe punishment would deter any potentially illegal uploaders and dramatically reduce pirated digital assets, pornography, illicit or malicious materials distributed through our P2P network. Unlike GDN, no sensitive private information is needed in basic operations of PCPN. All the operations of the protected mechanism in PCPN are completely transparent to end users. There is no need to provide any sensitive private information or to obtain access permission from a server to publish anything to PCPN. We would like to point out that our proposed system can incorporate any a priori protection technologies such as watermarking, DRM, or access control to make the system even better in fighting against piracy.

In the following description, a copy of digital material such as multimedia content, software, a file, in PCPN is called an object. The metadata associated with an object to provide auxiliary information or to specify behaviors is called attributes of the object. For example, the certificate used for tracking an uploader is a tracking attribute. In PCPN, an object and its tracking attribute are treated as an atomic unit when uploaded to PCPN or transferred from one peer to another.

3.1 PCPN Architecture

PCPN adds a copyright protection part to a conventional decentralized P2P network. The protection part controls access to PCPN and whether an object can be uploaded to, downloaded from, or replicated in PCPN. Each PCPN peer has a tamper-proof security module called signing and verifying module (SVM) which enforces copyright protection and access control. SVM functions like a black box to a user or other P2P client modules. It is very similar to the client side DRM module in a DRM system. Its security plays a key role in the copyright and privacy protection of PCPN. This paper focuses on the protection part of PCPN. The rest of the proposed P2P network is the same as a conventional decentralized P2P network. Details of most popular P2P systems can be found in [14].

Fig. 1 shows the basic architecture of the PCPN's protection part. The system consists of a trustworthy access control server (ACS) which individualizes SVM at each peer during software installation by issuing a (root) certificate that binds the peer's public key to the hardware of the peer. ACS also issues an ACS-signed revocation list to peers from time to time. ACS plays a critical role in PCPN. Its security must be guaranteed to ensure our P2P network operational. Once compromised, all clients' SVM modules need to be updated along with ACS, and all materials in the P2P network can no longer be used and will need to be re-uploaded. SVM at a peer generates a peer-signed certificate which is attached as the tracking attribute to each object the peer uploads to PCPN. SVM also checks the revocation list, verifies authenticity and integrity of an object and its tracking attribute before the peer uploads an object to or downloads an object from PCPN, or replicates an object. Any objects failed in this verification are removed from PCPN.

A revocation list (RL) containing a list of revoked certificates issued by ACS along with specifications of actions is distributed to peers or made available at a central server. A peer caches RL in local storage for later usage so that it does not have to download RL every time its SVM needs to check revoked certificates. When a peer enters PCPN, it optionally checks and updates the local RL from another peer or from the central server. If a threshold of maximum non-updating period has been reached, a peer is forced to update its locally stored RL. Objects signed by a revoked certificate may be removed from PCPN, depending on the action specifications in the revocation list. The action specifications also specify if a user whose ACS-issued certificate is listed in RL is allowed to access, upload to or download from PCPN. The SVM may also inform other modules of the peer's P2P software to refuse any service requests by the peer.

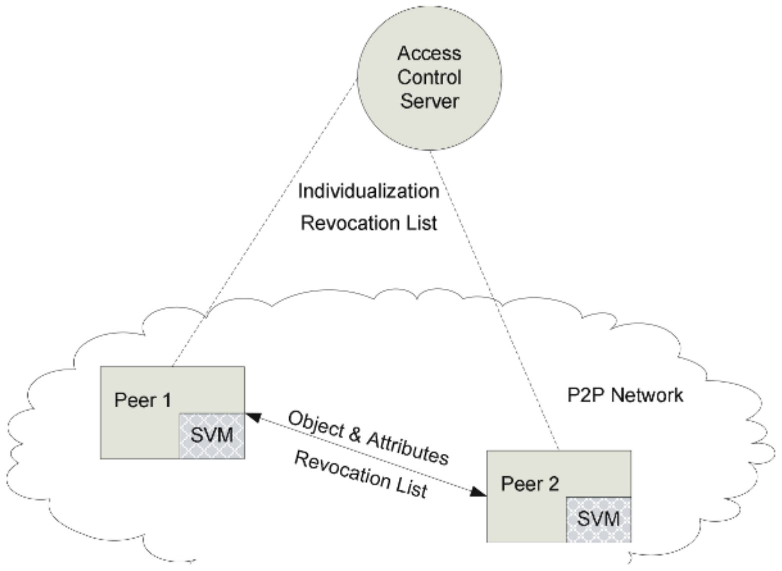

Fig. 1. Architecture of the PCPN's protection part

3.2 Peer Individualization

When the P2P software is first installed to a peer's PC, the peer's SVM is individualized. Individualization consists of several steps. A secure, tamper-proof individualization module (IM) at the client side executes these steps together with ACS. At the first step, IM retrieves the PC's hardware ID ID_p which is a combination of all the unique IDs of the PC's consisting components such as the hard drive(s), the network card, etc. IM also generates a private key k_p and a corresponding public key K_P , $D^a_{K_P}\{E^a_{k_p}\{x\}\} \equiv x, \forall x$, where $E^a_k\{\cdot\}$ and $D^a_k\{\cdot\}$ are respective asymmetric encryption and decryption operations with a key k . Symmetric encryption and decryption with a key k will be denoted by $E^s_k\{\cdot\}$ and $D^s_k\{\cdot\}$, respectively. This pair of private and public keys { k_p , K_P } will be used to sign the objects uploaded by the peer and verify authenticity and integrity of objects in PCPN. IM then sends a request to acquire an ACS-signed certificate together with the peer's hardware ID ID_p and the generated public key K_P to ACS securely.

At the second step, ACS receives ID_p and K_P sent by a peer's IM, and calculates an message authentication code (MAC) or keyed hash of ID_p: $GUID = h_{k_h}(ID_p)$, with the key k_h known only to ACS. The generated *GUID* is compared with those in revoked certificates. If the peer's *GUID* appears in the list, the request is rejected. Otherwise ACS signs *GUID* and K_P , $C_{ACS} = E^a_{k_{ACS}}\{GUID \,//\, K_P \,//\, T \,//\, Others\}$, where “ // ” means concatenation, k_{ACS} is the ACS’s private key used to sign certificates issued by ACS to peers, and T is the current time. In the basic form of the certificate C_{ACS} where the only action against an illegal peer is to remove the objects uploaded by the peer and to revoke the peer’s access to PCPN, there is no other peer’s information included in the certificate C_{ACS}, i.e., “Others” in C_{ACS} is empty. If the setting of PCPN requires the tracking mechanism to provide information for possible legal actions against an illegal user, personal information such as the peer’s IP address, email address, etc. may be obtained from the peer, verified by ACS, encrypted by a symmetric encryption with the key known only to ACS, and inserted into “Others” which is signed together with *GUID* by ACS. We note that the personal information is only known to ACS. A peer or its SVM cannot extract the personal information from the encrypted field in the certificate. Once a peer is convicted and the person is needed to be identified for legal actions, the personal information contained in C_{ACS} is decrypted by ACS and sent to law enforcement agencies to find out the perpetrator. At the end of this step, ACS sends C_{ACS} or rejection back to the peer.

At the third step, C_{ACS} received by the peer’s IM is stored together with the public key K_P in the local secure storage. IM also stores securely the private key k_p. They will be used by the peer’s SVM. IM then sends an acknowledgment to ACS which closes the individualization session. We note that the whole individualization process is completely transparent to an end user (except a user may possibly need to provide some personal information such as the email address in some setting of PCPN).

SVM consists of two components: the signing module (SM) and the verifying module (VM). SM is used to sign objects uploaded by the peer, and VM is used to verify authenticity and integrity of an object before the object is uploaded to, downloaded from, or replicated in PCPN. Both modules share a pair of secret keys k_1 and k_2. VM also contains the ACS’s public key to verify ACS-signed certificates and revocation list.

3.3 Uploading Objects

To upload an object *Obj* to PCPN, the following steps are executed by a peer’s SVM and P2P software. SM inside SVM first generates two random numbers α and β, and calculates the hash values $c = h_{k_1}(Obj // \alpha)$ and $\pi = h_{k_2}(\beta)$, where $h_k(\cdot)$ is a cryptographic keyed hash or MAC function using a key k. Then SM signs c to generate a peer signed certificate $C_p = E^a_{k_p}(c // T)$, where T is the current time, and encrypts its public key K_P and root certificate C_{ACS} with a symmetric cipher and the key π: $u = E^s_{\pi}\{K_P // C_{ACS}\}$. The set { C_P, u, α, β } is then inserted by SM to the object’s tracking attribute field which is treated as an integrated part of the object when moving into or out of PCPN or from one peer to another. SM sends the result to VM before finishing its task.

When VM receives an object, it first extracts the set $\{ C_P , u , \alpha , \beta \}$ from the object tracking attribute field, calculates $\pi = h_{k_2}(\beta)$, and decrypts u to extract K_P and C_{ACS}: $K_P // C_{ACS} = D_{\pi}^{s}\{u\}$. VM checks C_{ACS} against the revocation list. If it is revoked, then VM returns the action specified in the revocation list, such as no uploading is allowed, and the client software executes corresponding action. Otherwise, VM verifies the root certificate C_{ACS} and the just decrypted K_P, the uploader's public key. If it is fine, VM decrypts C_p to extract the hash value c : $c // T = D_{K_P}^{a}\{C_P\}$. This value c is compared with the hash value calculated from the object $h_{k_1}(Obj // \alpha)$. If they agree with each other, the authenticity and integrity of the object are verified and VM returns OK. The authenticity check verifies that the object is indeed signed by the claimed peer. This is done by first checking the client's public key against the ACS signed certificate and then using the public key to check the object integrity against the peer signed certificate C_p. Only the authentic peer who knows the peer's private key can pass these checks. The P2P software then uploads the object to PCPN. If any step in the authenticity and integrity verification fails, VM returns failure and the request to upload the object is rejected. When a peer uploads an object to PCPN, the peer does not need to contact any server. The whole process is completely transparent to an end user.

3.4 Other Operations

When a peer is going to download or replicate an object from another peer, the peer's VM checks authenticity and integrity of the object first. This checking is the same as the checking done by VM when a peer uploads an object, as described in Section 3.3. If the checking is OK, VM returns OK and the request is executed. Otherwise VM returns failure and the request is rejected. In the latter case, if the object fails authenticity and integrity verification or its uploader is in the revocation list and an action is required, the peer which stores the object that fails in the checking is contacted with the information of the problem, and the peer performs the corresponding checking. If the allegation is confirmed, the corresponding action is executed. For example, if the object fails the authenticity and integrity checking, then the alleged object is removed. If the uploader of the alleged object is in the revocation list, then the action specified in the revocation list is taken.

A peer also checks periodically the revocation list and the authenticity and integrity of all the objects stored at its side for PCPN. Any object that fails authenticity and integrity checking is removed from the peer's local storage, and actions on those objects uploaded by the peers in the revocation list are executed as specified by the revocation list. An example of actions is to remove all the objects uploaded by a peer in the revocation list. This checking usually occurs when the local revocation list is updated and new revoked certificates are found. This procedure ensures that PCPN functions as specified.

Depending on the policy set for PCPN and the actions specified in the revocation list, a peer in the revocation list may be denied to upload to or download from PCPN, or to access PCPN; confirmed pirated objects, if listed in the revocation list, are removed from PCPN; or even all the uploaded objects by the peers in the revocation list are removed. This can be implemented by requiring VM to check the revocation list and compare with the peer's public key.

It is also possible that a peer may be allowed to access or upload objects to PCPN when certain conditions are met. This can be realized simply by removing the peer from the revocation list or by modifying the specified actions. When VM rejects an a peer's request to upload or access PCPN due to the fact that the peer is in the revocation list, it usually updates its revocation list and checks if the peer's rights are recovered in the latest revocation list before returning failure.

3.5 Discussion

PCPN provides a strong protection of uploaders' privacy. What an end user of PCPN can see is the set { C_P, u, α, β } associated with an object. Since the security module SVM appears like a black box to end users, a user cannot extract any information about the object's uploader from that set. In other words, PCPN provides anonymous uploading. In addition, there is no way for an end user to find out if two objects are uploaded by the same user or not. This prevents a user from using statistical analysis to find out how many objects a specific user, although the real identity of the user is unknown, has uploaded to PCPN. We note that the client side SVM knows only the uploader's GUID and public key. Any personal information such as the email address is known only to ACS. A user cannot access SVM or ACS by the design. This means that PCPN has a strong protection of the uploader's privacy against PCPN users. This strong privacy protection is very desirable in many applications.

In PCPN, the server ACS is lightly involved in routine operations of the P2P network. ACS is involved only when a peer installs the P2P software to a peer and when the revocation list needs to be updated. If our tracking mechanism can effectively deter most users from uploading pirated contents, then almost all users of PCPN are law-abiding. ACS would issue new revocation lists only occasionally. This light involvement of a server in using P2P networks is exactly what we sought for in designing PCPN: PCPN should be decentralized as much as possible, and the whole protection mechanism should be transparent to end users as much as possible. Otherwise users would not be willing to use the P2P network for legitimate applications.

In PCPN, before ACS issues a root certificate C_{ACS} to a peer, the hash value of the peer's hardware ID is checked against revoked peers. Once revoked, a user cannot regain access to PCPN by reinstalling the P2P software. This guarantees convicted peers are permanently revoked unless their access is recovered by ACS. In other words, ACS has a robust control on which peers cannot access PCPN.

When a user updates his or her PC's hardware containing a unique ID, for example, replacing a network card with a new one, there is no need in a typical setting of PCPN to update the peer's root certificate C_{ACS} to match the new hardware. This mismatch between C_{ACS} and the hardware is allowed in general, which will be corrected when the P2P software is reinstalled or updated. If needed, it is possible to set up PCPN to periodically check if C_{ACS} matches the corresponding hardware by requiring a peer's SVM to periodically report the peer's hardware IDs to ACS, either voluntarily or requested by ACS, and ACS responds with the checking result. If a mismatch is detected, the same steps executed in the individualization stage to acquire the root certificate C_{ACS} are executed between the ACS and the peer's SVM. In this case, IM is put as an integrated part of SVM. This change would dramatically increase the workload and bandwidth requirement for ACS. PCPN should not be set in this way unless an application really requires so.

A major issue we have not touched is how to find out and prove that an object is pirated, illicit, or malicious. This issue is very important in real applications of PCPN yet very tough to find a satisfactory solution for. Watermarking and fingerprinting can facilitate fulfillment of the task with automatic tools, but current technologies are far from delivering such tools. PCPN currently relies on manual or semi-manual methods to achieve the goal. Every peer is encouraged to report abnormal objects to the content owner or ACS. Content owners or law enforcement agencies can also set up listening posts to monitor traffic in PCPN and report to ACS pirated objects with the evidences to prove the allegation, and/or offer incentives to encourage end users to report pirated objects. ACS accepts orders from a court or makes a judgment by itself to determine if an alleged peer's certificate should be revoked.

3.6 Security

Many issues and components may have impact on PCPN's security. Since all the cryptographic primitives used in PCPN are well studied and widely used in real applications, we can assume that those cryptographic primitives are all secure, and it is very difficult for an opponent to break the built-in authenticity and integrity checking schemes. Then the major security issues are the security of SVM and IM, and the interaction between SVM and other components of the P2P software. Most of those issues are typical security engineering issues in designing security modules or systems used in hostile environments. For example, a DRM system faces similar security engineering problems. Many commercial DRM systems have already been widely used and accepted on the market. For example, the Windows Media Rights Manager [15] from Microsoft is widely used by hundreds of thousands of users around the world. Those successful security engineering experiences and skills can be applied to build secure PCPN software.

SVM at each peer contains the secret keys k_1 and k_2. It is possible that some user with necessary expertise can successfully compromise his or her PC's SVM and extract the secret keys k_1 and k_2. This would enable the opponent to read the content in the root certificate C_{ACS} and the public key K_P of the uploader of each object, which weakens the PCPN's strong privacy protection. It would not enable the opponent to get around the PCPN's copyright protection mechanism. Since the information about an uploader obtained by the opponent in such a compromise is very limited, the opponent cannot gain much personal advantage with the hacking activity. We argue that an opponent would not bother to take trouble to hack SVM to extract the secret keys k_1 and k_2, and the current simple design for privacy protection is good enough for most applications.

Another security issue we would like to discuss here is a revoked user may get around our hardware ID checking in peer individualization by changing some hardware components such as the hard drive, the network card, and reinstalling the

P2P software to regain access or uploading privilege. We argue that this is not really an issue. Most users are not willing to spend money to buy new hard drives or network cards and install to PCs simply to distribute pirated materials to people they have never met or known. There exist many more efficient means than P2P networks to distribute pirated materials to relatives and friends. Even some people are willing to spend their own money and take troubles to install new hardware for the sake of other unknown users of PCPN, they have to keeping spending and installing since PCPN will ban their access to PCPN every time their uploaded pirated contents are detected. This "loophole" can be tightened if ACS replaces *GUID* with an encrypted version of ID_p with the decryption known only to ACS when generating a root certificate C_{ACS} for a peer. When a peer sends its hardware ID ID_p which contains all the unique IDs of the consisting hardware components of the PC during the P2P software installation, all these hardware component IDs are compared with those of revoked peers. If any single submitted component ID matches with a hardware component ID of a revoked peer, the request to issue a root certificate is rejected. This forces a user to replace all the hardware components with unique IDs in a PC to regain access to PCPN, which dramatically increases the cost. A negative impact of this modification is that a legitimate user may buy some used hardware components from a convicted user, resulting in being unable to access PCPN if the P2P software is reinstalled to the modified PC.

4 Conclusion

We have described a novel law-abiding P2P network with strong protection of uploaders' privacy. The network relies on a secure, reliable, and user-transparent tracking mechanism to track the uploader of any material in the network for copyright protection. An illegal peer is punished with its access to the network permanently revoked. All the materials uploaded by an illegal peer are also removed from the network. The system may also provide information for law enforcement agencies to track down the actual person who uploaded the illegal materials to the network and take legal actions again him or her. The proposed tracking mechanism should effectively reduce pirated and illicit materials distributed through a P2P network, and would ensure P2P technologies to advance on their own without legal or legislative interference.

We are implementing and testing the proposed law-abiding P2P network. We also continue improving the anti-piracy and privacy protection mechanisms in the network. One of the major research efforts is to develop technologies to automatically or semi-automatically detect pirated or illicit multimedia materials.

References

1. Kazaa. http://www.kazaa.com/
2. BitTorrent. http://bittorrent.com/
3. Gnutella. http://gnutella.wego.com/
4. FreeNet. http://freenet.sourceforge.net/

5. Supreme Court Rules on June 27, 2005 in MGM v. Grokster. http:// www. copyright.gov/docs/mgm/
6. Berman, H.: Peer to Peer Piracy Prevention Act (H.R. 5211). http://www.house.gov/berman/newsroom/piracy_prevention_act.html (July 2002)
7. Hatch, O.: Inducing Infringement of Copyrights Act (S. 2560). http://hatch.senate.gov/index.cfm?FuseAction=PressReleases.Print&PressRelease_id=1083&suppresslayouts=true (June 2004)
8. Kalker, T., Epema, D. H. J., Hartel, P. H., Lagendijk, R. L., van Steen, M.: Music2Share–Copyright-Compliant Music Sharing in P2P Systems. In: Proc. of IEEE. 92(6) (2004), pp. 961–970
9. Bakker, A., van Steen, M., Tanenbaum, A.: A Law-Abiding Peer-to-Peer Network for Free-Software Distribution. In: IEEE Proc. Intl. Symp. Network Computing and Applications (Oct. 2001) pp. 60–67
10. Digital Millennium Copyright Act. United States Public law (Oct. 1998)
11. Secure Digital Music Initiative (SDMI). http://www.sdmi.org/
12. Copy-protection Technical Working Group. http://www.cptwg.org
13. Hollings, F.: Consumer Broadband and Digital Television Promotion Act (CBDTPA) (S. 2048). http://thomas.loc.gov/cgi-bin/query/z?c107:S.2048: (March 2002)
14. Aberer K., Hauswirth, M.: An Overview on Peer-to-Peer Information Systems. In: Proc. Workshop Distributed Data and Structures (WDAS'2002) (2002)
15. Microsoft: Digital Rights Management (DRM) – Windows Media DRM 10. http://www.microsoft.com/windows/windowsmedia/drm/default.aspx