Communications and Network, **2010, 2, 230-234**
doi:10.4236/cn.2010.24033 Published Online November 2010 (http://www.SciRP.org/journal/cn)

A Novel Digital Rights Management Scheme in P2P Networks

Jing Feng, Ruoshan Kong, Yulin Wang
International School of Software, Wuhan University, Wuhan, China
E-mail: *gfeng@whu.edu.cn*
Received November 12, 2010; *revised November* 15, 2010; *accepted November* 16, 2010

Abstract

P2P networking is a distributed application architecture that partitions tasks or workloads between peers. How to integrate P2P networks and DRM to offer a novel content distribution mode for digital media resources is a significant research project. In this paper, a novel DRM architecture in P2P Networks is proposed, three phases include content provide phase, content purchase phase and content access phase, are modeled, and key technologies are introduced. Finally analysis indicates that the proposed scheme has the characteristics of security, controllability and scalability.

Keywords: Peer-to-peer (P2P), Digital Rights Management (DRM), Intellectual Property (IP)

1. Introduction

Peer-to-peer (P2P) networking is a distributed application architecture that partitions tasks or workloads between peers [1]. The rise of P2P networks has been an inevitable outgrowth of the rise of the Internet. Unfortunately, P2P networks have grown from useful tools in information sharing to havens for trafficking in unauthorized copies of Intellectual Property (IP) [2]. The widespread application of P2P file sharing brings about some critical problems, such as security and piracy. How to protect IP in such P2P networks has become an urgent problem.

Digital rights management (DRM) is a term for access control technologies that can be used by hardware manufacturers, publishers, copyright holders and individuals to limit the usage of digital content and devices [3]. There are a number of DRM solutions on the market, such as Microsoft's Windows Media Rights Manager (WMRM), IBM's Electronic Media Management System (EMMS), InterTrust's Rights System, and RealNetworks's RealSystems Media Commerce Suite (RMCS) [4]. However, there are few DRM solutions in P2P networks. How to integrate P2P networks and DRM to offer a novel content distribution mode for digital media resources is a significant research project [5]. By now some research results have been achieved. In [6], a manageable overlay network architecture with DRM is proposed for live streaming. In [7], an integrated copyright protection scheme for large-scale content delivery over the Internet is proposed. And [8] describes a solution for the problem of copyright infringement in P2P networks for music sharing, and a P2P protocol that integrates the functions of identification, tracking, and sharing of music with those of licensing, monitoring, and payment is proposed.

In this paper, a novel digital rights management scheme in P2P networks is proposed. The scheme combines the advantages of P2P networks and DRM. The rest of this paper is organized as follows. The proposed architecture integrating DRM and P2P networks is introduced in Section 2. In Section 3, key technologies include file packaging, kernel control and file hiding in the scheme are described. Finally, the characteristics of the proposed system are concluded in Section 4.

2. Proposed Architecture

The network architecture of P2P is shown in **Figure 1.** There are four kinds of key nodes: (1) Certificate server, offers two kind of certificate, one for digital content providers and the other for digital content buyer; (2) Authentic server, stores the user authorizing certificates and keys, and is responsible to verify user's identity; (3) Index server, maintain the status and buffer information of peers. The status information consists of the peers' identifying information such as user identification, password, MAC address, and so on; (4) Peers, digital content

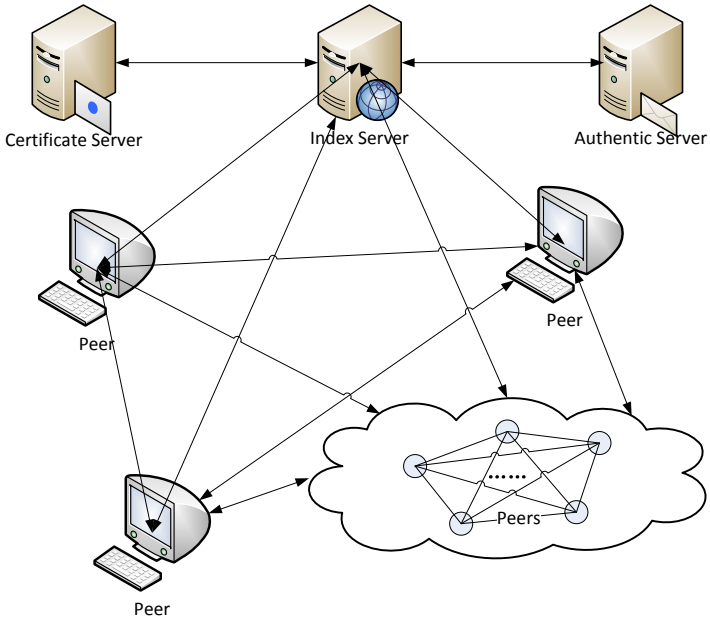

Figure 1. The DRM architecture in P2P network.

providers or buyers, storing content package, keep connecting to index server during the whole alive time.

- *CS*: certificate server
- *AS*: authentic server
- *IS*: index server
- p_i: peer *i*
- HFP_i: hardware fingerprint information of peer *i*
- UI_i: user information of user *i*
- $C_{i,j}$: the content *j* on peer *i*
- $\overline{C_{ij}}$: the encrypted content *j* on peer *i*
- $Mi_{,j:}$ the piece of content *j* offered by peer *i*
- $\overline{M_{i,j}}$: the piece of encryption content *j* offered by peer *i*
- K_i: content encryption key for peer *i*
- $\overline{K_i}$: content decryption key for peer *i*

There are mainly three phases in the scheme: content provide phase, content purchase phase and content access phase.

2.1. Content Provide Phase

In this system, any content provider needs to register on *CS* and then the corresponding certificate of provided content can be obtained. The certificate includes user information (such as user identification, password), hardware fingerprint information (such as CPU identification, MAC address, disk serial number, etc.), content information (such as content size, content type, content price and etc.) and encryption key. After encrypting and packaging, the content package can be shared on the peer of content provider. The basic work flow of content provide phase is as follows:

Step 1: content provider, suppose *Pi*, sends a request to *CS* for sharing the digital content. The user information and hardware fingerprint information of this provider and content information will be send to *CS*. *CS* offers digital content provider certificate to the peer.

$\boldsymbol{K_i}$, as (1) show, composed of UI_i and HFP_i is the major part of this certificate.

$$K_i = UIi_i \text{ ※ } HFP \tag{1}$$

Step 2: the content will be encrypted using K_i, as (2) shown, and then packaged by the P2P client application to add prefix to the package. The details of package prefix will be described in the Section 3. Then the self-extracting package can be shared by the P2P client application in p_i.

$$C_{i,i} \xrightarrow{\text{encrypt with } K_i} C_{ii} \tag{2}$$

2.2. Content Purchase Phase

Any content buyer, suppose p_i, want to buy content j needs to register on certificate server and downloads content using certificate by paying with a purchase order. Then the list of the peers sharing the content package can be obtain from IS. The P2P client application on p_i can download complete content package pieces from these listed peers. The basic work flow of content purchase phase is as follows:

Step 1: p_i register on certificate server using user

information and hardware fingerprint information.

Step 2: search the content in the P2P client application in p_i. After paying with a purchase order, the list of peers sharing the content can be obtained from IS.

Step 3: download content pieces from these peers and compose them to $\overline{C_{i,j}}$, as (3) described, where P is set of peers offer content pieces to p_i.

$$\overline{C_{i,j}} = \sum_{p_m \in P} \left(M_{m,j} encryptwithK_i \right)$$

where $$\overline{M_{m,f}} \xrightarrow{decryptwith\ \overline{K_m}} M_{m,f} \quad (3)$$

Then, $\overline{C_{i,j}}$ and some control programs are packaged into self-extracting package. At this time p_i is not only a buyer, but also a content sharing peer until the package is deleted in p_i.

2.3. Content Access Phase

Step 1: double click the self-extracting package, the programs stored in prefix of package will first be run. p_i's certificate will be verified by AS. If p_i is authorized, $\overline{K_i}$ can be get from AS and the $\overline{C_{i,j}}$ can be decrypted using (4). Otherwise, unauthorized accessing message will pop-up to p_i.

$$\overline{C_{i,j}} \xrightarrow{decrypt\ with\ \overline{K_i}} C_{i,j} \quad (4)$$

Step 2: $C_{i,j}$, can be opened by the universal application associated with file type of $C_{i,j}$.

Step 3: Once finish accessing $C_{i,j}$, temporary files generated by self-extracting package will be cleared.

3. Key Technologies

3.1. File Packaging

The main function of the package module (shows in **Figure 2**) is to package the encrypted ciphertext, monitoring procedures, packing procedures, files and processed hiding drive, monitoring module DLL and API interception drive together.

3.2. Kernel Control

Client monitoring module is the most important and complicated module in this system. It takes charge of the running state of the third-party applications. And it can prevent illegal operations such as "copy", "paste" or "save as" which may destruct copyrights. Monitoring module includes three files: monitoring procedure (decipher.exe), mouse/keyboard hook(mousehook.dll), and API interception drive(driver_hook_ssdt.sys). The basic flowing is as follows:

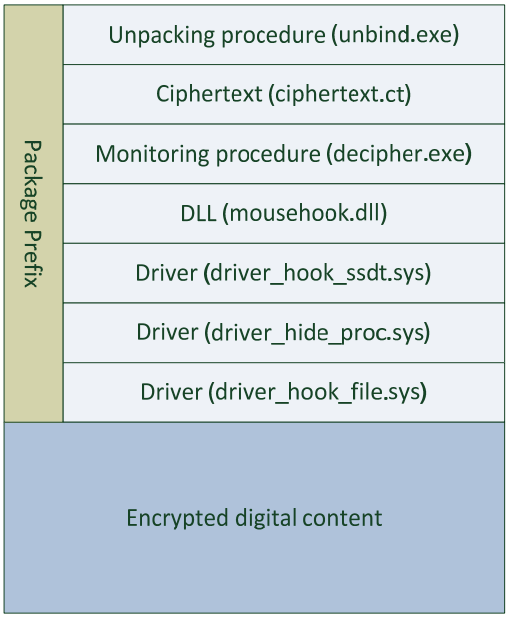

Figure 2. Internal structure of the packaged file.

1) After unpacking, monitoring procedure and API interception drive will run automatically. First, monitoring procedure must obtain the absolute path which leads to object file, and then pass the path to the API interception drive. The system uses the "ShellExecute" function to open the file. This function would ask for the file's path, and it can choose the third-party application according to the file format.

2) Since the API interception drive has the absolute path, system will monitor all processes that are opening files at the moment, and check whether their absolute paths are the same path as the one which monitoring procedure has transferred already. If they are the same, the system will return the process ID to the monitoring procedure.

3) When the monitoring procedure obtains the process ID of a third-party application, it will use the SetWindowsHookEx API function to hang the object process to the mouse/keyboard hook. This action will prevent unauthorized "copy" or "save as". The detail principle will be expressed in the next section.

4) After obtaining the process ID of the third-party application, API interception drive will change the ZwWriteFile API function to prevent all the "write" operation that the function may operate. This will protect the content at the drives level. More detail can be found in the next section.

5) The monitoring procedure will wait the third-party process until the process stop. Once the process stops, system will unload the DLL and drives, and delete the two file folders and all their contents that are generate when unpacking occurs.

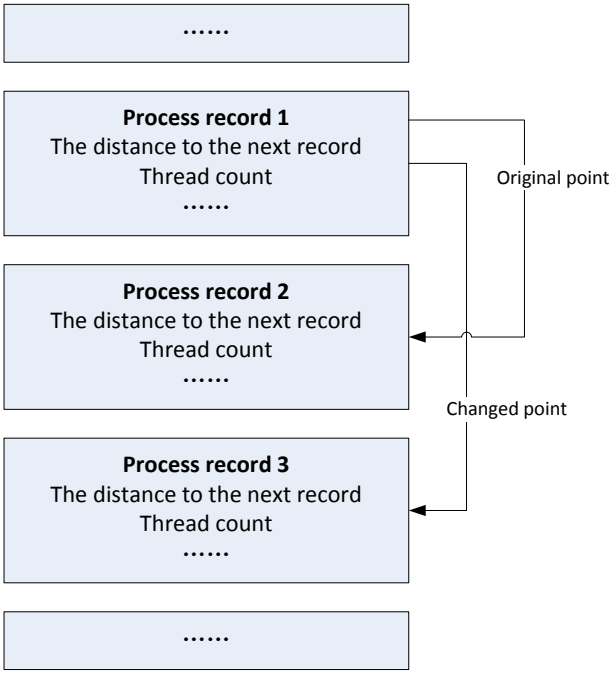

Figure 3. Modification of processes list.

3.3. Processes Hiding

Processes hiding can make use of the Rootkit technology. When the files are unpacked, system will load a process hiding drive (driver_hide_proc.sys) to hide the running monitoring procedure. This can prevent users to check or end the monitoring process with some tools such as explorer. Processes hiding module does its job at the drives level. The details of principle are as follows:

1) When the system obtains the current system processes list, it will call a kernel function named ZwQuerySystemInformation. But if the address of ZwQuerySystemInformation in SSDT is changed, the system can call the NewZwQuerySystemInformation (Detour Function) driver_hide_proc.sys.

2) The driver_hide_proc.sys stores the original address of ZwQuerySystemInformation. NewZwQuerySystem Information can call the original ZwQuerySystemInformation function with the original address.

3) Call the original function ZwQuerySystemInformation.

4) Because NewZwQuerySystemInformation calls ZwQuerySystemInformation when the system is running, ZwQuerySystemInformation will return the result to NewZwQuerySystemInformation, not directly to the system.

5) **Figure 3** shows that NewZwQuerySystemInformation function can modify the returned processes list, delete the records of packing process and monitoring process, then return the changes list to the system.

3.4. Files Hiding

The principle of files hiding module is basically the same as processes hiding module. To hide files, we can replace the kernel function of ZwQueryDirectoryFile, or modify the FileInformationBuffer list.

4. Conclusions

According to the scheme, we can conclude the characteristics of the proposed system as follows:

1) Security: the system security can be carried out from three aspects including content, user and right. Specific certificates are defined for specific users.

2) Controllability: it can process transmission control, post control and access control etc. Certificates are combined with hardware information. The system can take control of the personal computers that consumers use.

3) Simplicity: consumers can use digital content without installing any other custom software.

Scalability: it can be realized from many aspects such as the functions, modularization, interfaces and rights expression language.

5. Acknowledgements

This work was supported in part by the National High-Tech Program “863” of P. R. China under Grant No. 2009AA01Z412.

6. References

[1] En.wikipedia.org/wiki/Peer-to-peer.

[2] B. Rosenblatt, "Integrating DRM with P2P Networks: Enabling the Future of Online Content Business Models," *DRM Watch*, Vol. 18, 2003.

[3] En.wikipedia.org/wiki/Digital_rights_management

[4] Q. Liu, R. Safavi-Naini and N. P. Sheppard, "Digital Rights Management for Content Distribution," *Proceedings of the First Australasian Information Security Workshop*, Vol. 2003, 2003, pp. 49-58.

[5] Y. B. Wang, R. Lv and Z. G. Hong, "A P2P Application Demonstration System for Resilient Overlay Networks with Intelligent Nodes Supporting DRM," *Proceedings of the* 2009 *International Joint Conference on Computational Sciences and Optimization*, 2009, pp. 336-339.

[6] X. G. Lan, J. R. Xue, L. H. Tian, *et al*., "A Peer-to-Peer Architecture for Live Streaming with DRM," *Proceedings of the* 6*th IEEE Consumer Communications and Networking Conference*, 2009, pp. 1-5.

[7] X. S. Lou, K. H. wang and R. F. Zhou, "Integrated Copyright Protection in Peer-to-Peer Networks," *Proceedings of the* 27*th International Conference on Distributed Computing System Workshops*, 2007, pp. 28-35.

[8] T. Kalker, D. H. J. Epema, P. H. Hartel, *et al.*, "Music2Share-Copyright-Compliant Music Sharing in P2P Systems," *Proceedings of the IEEE*, Vol. 92, No. 6, 2004, pp. 961-970.

