APPARATUS AND METHOD FOR MANAGING DIGITAL RIGHTS THROUGH HOOKING A KERNEL NATIVE API

CROSS-REFERENCE TO RELATED APPLICATION

[0001] This application claims priority to Korean Patent Application No. 10-2010-0112146 filed on November 11, 2010 and all the benefits accruing therefrom under 35 U.S.C. §119, the disclosure of which is incorporated herein by reference in its entirety.

BACKGROUND

[0002] The present invention relates to an apparatus and method for managing digital rights and, more particularly, to an apparatus and method for managing digital rights through hooking a kernel native API.

[0003] Digital Rights Management (DRM) is a technology for protecting various digital contents, such as e-book, music, video, game, software, stock information, image, and so on, from illegal duplication and for setting access rights to electronic documents for preventing leaks of information by insiders and outsiders. In past, most information leakage incidents are performed by hackers or viruses. However, in recent, as information leakage incidents have been caused for money by authorized insiders or related-parties, the voices of supplementing weak points inherent in existing security solutions are getting louder. Almost all documents of business entities are digitalized, and systems for sharing the digitalized documents are increasingly introduced. Thus, Enterprise DRM (E-DRM) is perceived as the most representative document security technology. The Enterprise DRM solution growing quickly domestically and abroad represents evolution to

various technologies for meeting requirements of the current security market. Especially, the needs for new products applying these technologies increase in the markets requiring more enhanced information security solution. DRM solutions for directly encrypting contents which are containing core information and controlling the use of contents on the basis of the predefined right have a good position through core security infrastructures satisfying this market needs.

[0004] In typical DRM processing techniques, the DRM processing is implemented by a kernel driver or filter driver. Accordingly, the typical DRM processing techniques support only specific application programs in DRM processing of electronic documents or digitalized files, and thus additional DRM modules should be developed additionally, thereby additional developing time and cost are required. Also, the typical DRM processing techniques analyze specific application programs by using Reverse Engineering, and control and monitor specific messages, instructions and user interfaces (UIs) in the application programs by using the DLL (Dynamic Linking Library) injection method. In this case, the typical DRM processing techniques control DRM processing and manage digital rights through a kernel driver or filer driver.

[0005] However, there are following limitations in the typical DRM processing technique. First, since the DLL for the control of the corresponding application program should be developed on the basis of the analysis results of the specific application programs using reverse engineering, it takes long time in developing a new DLL. Thus, security issues may occur during the developing time. Second, modules which are injected to operate at the user level may conflict with other programs to make a stability problem. Third, in DRM processing by the kernel driver, conflicts with other kernel drivers or programs may arise and cause a system

crash. Fourth, the flexibility is reduced because of developing the new kernel driver in changing the kernel or the operation system.

SUMMARY

[0006] The present invention is directed to providing an apparatus and method for managing digital rights which can perform DRM processing on new products or versions of all application programs and thus immediately respond to the leakage of electronic documents.

[0007] The present invention is also directed to providing a non-transitory computer readable medium recording a program for executing in a computer a method for managing digital rights which can perform DRM processing on new products or versions of all application programs and thus immediately cope with the leakage of electronic documents.

[0008] According to an aspect of the present invention, there is provided an apparatus for managing digital rights includes: an agent unit managing application programs to which DRM is to be applied and a right to contents processed by the application programs to which DRM is to be applied and encrypting or decrypting the content processed by the application programs to which DRM is to be applied; a rights management unit authenticating a user and managing a user right to the contents processed by the application programs to which DRM is to be applied; and a kernel API hooking unit monitoring input/output of a file through hooking of kernel native APIs, requesting the rights management unit to verify a user right to contents to be processed when the contents are detected as being processed by the application programs to which DRM is to be applied, and requesting the agent unit to encrypt or decrypt the contents to be processed when the user right to the contents to

be processed is verified, wherein the agent unit, the rights management unit, and the kernel API hooking unit are driven for each application program to which DRM is to be applied.

[0009] According to another aspect of the present invention, there is provided a method for managing digital rights includes: (a) injecting a DRM engine module into an application program to which DRM is to be applied when the application program to which DRM is to be applied is driven; (b) monitoring input/output of a file through hooking a kernel native API, and when a content is detected as being processed by the application program to which DRM is to be applied, verifying a user right to the content to be processed in the application program to which DRM is to be applied; (c) selectively encrypting or decrypting the content to be processed when the user right to the content to be processed is verified; and (d) returning the encrypted or decrypted content to the kernel native API, wherein step (a) and step (c) are performed at user level, and step (b) and step (d) are performed at kernel level.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] The above and other objects, features and advantages of the present invention will become more apparent to those of ordinary skill in the art by describing in detail exemplary embodiments thereof with reference to the accompanying drawings, in which:

[0011] FIG. 1 is a block diagram showing a configuration of an apparatus for managing digital rights according to an exemplary embodiment of the present invention;

[0012] FIG. 2 is a flowchart illustrating a procedure of performing a method for managing digital rights according to an exemplary embodiment of the present

invention;

[0013] FIG. 3 is a flowchart illustrating a procedure of reading a document in a method for managing digital rights according to an exemplary embodiment of the present invention;

[0014] FIG. 4 is a flowchart illustrating a procedure of writing a document in a method for managing digital rights according to an exemplary embodiment of the present invention; and

[0015] FIG. 5 is a flowchart illustrating a procedure of performing document print, clip board copy, and screen capture in a method for managing digital rights according to an exemplary embodiment of the present invention.

DETAILED DESCRIPTION OF EMBODIMENTS

[0016] Exemplary embodiments of the present invention will be described in detail below with reference to the accompanying drawings. While the present invention is shown and described in connection with exemplary embodiments thereof, it will be apparent to those skilled in the art that various modifications can be made without departing from the spirit and scope of the invention.

[0017] An apparatus and method for managing digital rights according to an exemplary embodiment of the present invention should satisfy following requirements. First, the apparatus and method should include a DRM processing structure which can automatically apply DRM on the electronic files generated by all application programs. Second, the apparatus and method should immediately respond to new versions or products of the application program. Third, the apparatus and method should minimize the possibility of conflicts with other application programs or modules. Fourth, the apparatus and method should have enhanced

availability and flexibility to be applied even when the operating program or the kernel is changed. Fifth, the apparatus and method should facilitate the addition or improvement of functions through simple module change.

[0018] FIG. 1 is a block diagram showing a configuration of an apparatus for managing digital rights according to an exemplary embodiment of the present invention.

[0019] Referring to FIG. 1, an apparatus for managing digital rights 100 includes an agent unit 110, a rights management unit 120, and a kernel API hooking unit 130.

[0020] The agent unit 110 injects a DRM processing module into an application program for the control of the application program, manages rights to contents applied DRM (hereinafter, 'DRM content'), and encrypting/decrypting the DRM content. For this, the agent unit 110 includes a communication unit 112, a user policy management unit 114, an injection unit 116, a document rights management unit 118, and an encryption/decryption unit 119.

[0021] The communication unit 112 is an element for performing communication between the agent unit 110 and a management server (not shown) managing DRM policies, and transmits/receives the policies related with DRM to/from the management server. Then, the DRM policies received from the management server may include information for each application program or content, information on the application programs to which DRM is to be applied, and information on allowance/limitation/change of a user right. Accordingly, the specific application programs (for example, MS EXEL, MS WORD, etc.) are set to perform DRM processing through the DRM policies provided from the management server to the agent unit 110, and change or restrict access to specific documents by a specific

user or user group. Furthermore, access rights of a user to a specific application program or content may be temporarily changed or restricted during a certain period. Like this, the DRM policies received from the management server is delivered to a document rights management unit 118 and rights management unit 120 depending on their attribute. In the following description, the content will be described using a document as an example, but includes a file with an electronic form, which is written for delivering information such as a video, picture, and web document.

[0022] The user policy management unit 114 performs a function of changing (that is, adding or deleting) user rights which are set to a user or a document. Accordingly, a user can change an access rights to a document within his/her rights. Furthermore, information on the user rights which is received from the management server may be reflected in DRM processing.

[0023] The injection unit 116 drives a DRM engine for each application program for the control of the application program. For this, the injection unit 116 is driven together when a power is supplied to a system to perform an initial operation (for example, booting). The injection unit 116 monitors all application programs executed on an operating system (OS). And, in executing an application program, to which DRM is to be applied, the injection unit 116 drives the DRM engine to monitor input/output of a file. In this case, the injection unit 116 may include a list of application programs to which DRM is to be applied. However, for more flexible operation, the document rights management unit 118 may manage a list of application programs to which DRM is to be applied. And also, when an application program is driven, the injection unit 116 inquires of the document rights management unit 118 to confirm whether the corresponding application program is the application program to which DRM is to be applied. The apparatus for managing digital rights

100 according to the present invention may perform the management of digital rights regardless of kinds of application programs through this configuration, unlike existing apparatuses for managing digital rights which is only for each application program (for example, in the case of a Hangul program of Hancom, Inc., managing digital rights is performed using internal functions of the corresponding program).

[0024] The document rights management unit 118 manages a list of application programs to which DRM are to be applied, and manages rights information for each application program or document. The rights information corresponding to a certain application program may include temporary restriction information on rights of printing, storing, and modifying any document which is processed by the application program. The rights information corresponding to a document may include information on transmission, copy, and edit rights to the document. Various rights suitable for enterprise environments may be controlled through rights management by the document rights management unit 118.

[0025] The encryption/decryption unit 119 performs encryption and decryption on a document. The encryption/decryption unit 119 can be controlled by an encryption/decryption control unit 134 included in the kernel API hooking unit 130, and outputs the encryption and decryption results of the document to the encryption/decryption control unit 134.

[0026] The rights management unit 120 is an element for operating at the user level and performs access history management for DRM documents and user rights management for processing of documents, such as user authentication, modification, print, copy, and so on. For this, the rights management unit 120 includes a user authentication unit 122, a user rights management unit 124, and a use history management unit 126.

[0027] The user authentication unit 122 verifies a user right to a document or an application program on the basis of identification information on a user or a user terminal and outputs the result of user authentication. Also, the user rights management unit 124 processes matters associated with read, write, modification, capture, clipboard use, and print rights to a document on the basis of a result of user authentication. Accordingly, if the user right does not include the right of printing a document, the user rights management unit 124 does not perform instructions for printing the document by blocking a printing function of the application program. The use history management unit 126 manages a use history related to opening, modification, and storage of a DRM document. The use history management unit 126 is an element selectively included as necessary. Access to documents may be supported in other's computer using the user authentication described above, and logging information on document utilization (that is, open, storage, deletion, print, and capture) history may be correctly managed through use history management.

[0028] The kernel API hooking unit 130 hooks kernel native APIs to monitor input/output of a file, and on the basis of this, to control operations of the agent unit 110 and the rights management unit 120. The kernel native APIs are internally used on a windows operating system. The kernel native APIs of the windows operating system are classified with APIs starting as 'Nt' which can be used only in a kernel mode and APIs starting as 'Zw' which can be used in a user mode. The kernel native APIs are common regardless of application programs, and thus can perform DRM processing independently in application programs through the hooking. However, typical DRM management techniques using a kernel API hooking have a limitation due to the dependency on application programs in that API hooking is performed through reverse analysis of each application program and various APIs are used for

input/output of a file.

[0029] For the DRM processing through the kernel native API hooking, the kernel API hooking unit 130 includes a rights control unit 132, an encryption/decryption control unit 134, and a cache processing unit 136. In this case, the cache processing unit 136 may be included as necessary.

[0030] The rights control unit 132 inquires of the document rights management unit 118 of the agent unit 110 and the user rights management unit 124 of the rights management 120 to confirm a right to the document corresponding to a file input/output event notified from the encryption/decryption control unit 134, and delivers the confirmation result to the encryption/decryption control unit 134 to control storage and decryption rights. At this point, the right control unit 132 inquires of the document rights management unit 118 whether temporary restriction of user rights is set according to rights policy received from the management server.

[0031] The encryption/decryption control unit 134 hooks the kernel native APIs to detect file input/output events. As described above, the apparatus for managing digital rights according to the present invention is driven for each application program to which DRM will be applied. Accordingly, the encryption/decryption control unit 134 may recognize the file input/output events, which are caused by an application program managed by the encryption/decryption control unit 134, among the kernel native API events. And also, the encryption/decryption control unit 134 inquires of the rights control unit 132 about rights to a document which is an object of the file input/output events, and according to the inquiry result, processes decryption, encryption, storage, and modification of the document. Furthermore, when the document is necessary to be decrypted (that is, the reading of the document is requested), the encryption/decryption control unit 134

inquires of the cache processing unit 136 whether the document has been previously decrypted. If the read-requested document is confirmed as having been already decrypted, the encryption/decryption control unit 134 receives the decrypted document from the cache processing unit 136 and provides the received document to the file input/output kernel native API. Unlike this, if the read-requested document is confirmed as having been not decrypted, the encryption/decryption control unit 134 requests the encryption/decryption unit 119 to decrypt the encrypted document, receives the decrypted document from the encryption/decryption control unit 134, and provides the received document to the file input/output kernel native API. When a kernel native API event occurs, such as clipboard copy, screen capture, or document print, the encryption/decryption control unit 134 provides the result, according to rights of the user or rights set to the document which are confirmed through the rights control unit 132, to the file input/output kernel native API

[0032] It is described above that the encryption/decryption control unit 134 detects the file input/output kernel native API and returning the detection result as an example. Alternatively, the rights control unit 132 may detect the file input/output kernel native API, and the each corresponding element may return the result. As an example, in the case of document reading, if a read right to the read-requested document is confirmed, the rights control unit 132 requests the encryption/decryption control unit 134 to decrypt the document, and the encryption/decryption control unit 134 inquires of cache control unit 136 whether the document has been previously decrypted. If the document has been previously decrypted, the cache control unit 136 provides the decrypted document to the corresponding file input/output kernel native API directly or through the encryption/decryption control unit 134. Unlike this, if the document is confirmed as having been not decrypted, the

encryption/decryption control unit 134 requests the encryption/decryption unit 119 to decrypt the encrypted document, receives the decrypted document from the encryption/decryption control unit 134, and provides the received document to the file input/output kernel native API.

[0033] If the file input/output kernel native API event corresponding to document writing, the encryption/decryption control unit 134 requests the encryption/decryption unit 119 to encrypt the document to be written. And then, the encryption/decryption control unit 134 receives the encrypted document received from the encryption/decryption unit 119, and provides the received document to the cache processing unit 136 such that the cache processing unit 136 can store the document. And also, the encryption/decryption control unit 134 provides the received document to the file input/output kernel native API. Where a file input/output kernel native API event corresponding to document modification is detected, the procedure related to file encryption is the same as the case of the document writing, but the procedure of confirming a document modifying right is add.

[0034] The cache processing unit 136 is an element for preventing an opened document from being encrypted or decrypted repeatedly. Accordingly, the cache processing unit 136 receives first encrypted or decrypted document from the encryption/decryption control unit 134 and stores the document in a built-in cache memory. If an inquiry whether the document to be processed has been decrypted is input from the encryption/decryption control unit 134, the cache processing unit 136 searches the cache memory, and then returns the document or outputs a signal of notifying that the document has been not decrypted to the encryption/decryption control unit 134.

[0035] FIG. 2 is a flowchart illustrating a procedure of performing a method for managing digital rights according to an exemplary embodiment of the present invention

[0036] Referring to FIG. 2, in booting a system, an injection module is executed and monitors whether an application program is running (S200). At this time, the injection module monitors all application programs running on an OS (S210), and checks whether the running application program is a predetermined application program to which DRM is to be applied (S220). If the running application program is the predetermined application program to which DRM is to be applied, the injection module injects a DRM engine module, where a method for managing digital rights according to the present invention is implemented, into the application program (S230). The DRM engine module injected into the application program hooks kernel native APIs to perform DRM processing on documents (S240).

[0037] FIG. 3 is a flowchart illustrating a procedure of reading a document in a method for managing digital rights according to an exemplary embodiment of the present invention.

[0038] Referring to FIG. 3, a document reading command is output from an application program to which DRM is to be applied (S300). The DRM engine module monitors document reading events in the kernel native APIs (S310). Then, the DRM engine module checks whether a user has a right to the document (S320). If the user is confirmed as not having a right to read (S330), the DRM engine module blocks the document from being opened (S340). Unlike this, if the user is confirmed as having the right to read (S330), the DRM engine checks whether the document has been previously decrypted (S350). If the decrypted document is not confirmed as existing (S360), the DRM engine module decrypts the document (S370) and stores

the decrypted document in the cache memory (S380). And then, the DRM engine module calls an original kernel native API and delivers the decrypted document (S395). Unlike this, if the decrypted document is confirmed as existing (S360), the DRM engine module does not perform a new decryption process, but directly call the original kernel native API and delivers the stored document.

[0039] FIG. 4 is a flowchart illustrating a procedure of writing a document in a method for managing digital rights according to an exemplary embodiment of the present invention.

[0040] Referring to FIG. 4, a document opening command is output from an application program to which DRM is to be applied (S400). The DRM engine module monitors document opening events in the kernel native APIs (S410). Then, the DRM engine module checks whether a user has a right to the document (S420). If the user is not confirmed as having a modification right to the document (S430), the DRM engine module blocks the documents from being stored (S440). Unlike this, if the user is confirmed as having the modification right (S430), the DRM engine checks whether the document has been changed (S450). If the document is confirmed as having been changed, the DRM engine module encrypts the document (S460). In this process, the DRM engine module may store the encrypted document in the cache memory according to settings. Finally, The DRM engine module calls an original kernel native API and delivers the encrypted document (S470).

[0041] FIG. 5 is a flowchart illustrating a procedure of performing document print, clip board copy, and screen capture in a method for managing digital rights according to an exemplary embodiment of the present invention.

[0042] Referring to FIG. 5, when an execution command such as document print, clipboard copy, screen capture is output from an application program to which

DRM is to be applied (S500), the DRM engine module checks which rights are set for the document (S510). In this case, the print right and clipboard copy right is included in user rights, and the screen capture right is included in document rights. If the right to the execution command (for example, clipboard copy) is confirmed as being set for the document (S520), the DRM engine module permits requested processing (S530). Unlike this, if the right to the execution instruction is confirmed as not being set for the document (S520), the DRM engine module refuses requested processing (S540).

[0043] The method for managing digital rights according to an exemplary embodiment of the present invention with reference to FIGS. 2 to 5 may be performed on the apparatus for managing digital rights according to an exemplary embodiment of the present invention with reference to FIG. 1. In this case, the apparatus for managing digital rights according to the present invention functions as the DRM engine module. And also, the method for managing digital rights according to an exemplary embodiment of the present invention with reference to FIGS. 2 to 5 may be implemented in software. In this case, the DRM engine module is installed in a target system as a program, and injected into an application program to which DRM is to be applied, by the injection module included in the DRM engine module.

[0044] Using the apparatus and method for managing digital rights according to the present invention as described above, DRM processing can be performed on all application programs, thus immediately responding to electronic file leakage and applying DRM to electronic files of various formats. And also, since DLLs required for the injection of a specific application program do not exist separately, the size of the engine can be reduced, thus using a storage medium efficiently. Furthermore, at

the kernel level, control and DRM processing are performed, but the control and request of the only DRM processing are performed, thereby reducing processing processes to minimize unnecessary resource occupancy. Accordingly, although applying DRM, stability may be enhanced without a computer being slow. Compared with existing DRM techniques, a comfortable computer environment can be provided due to speed enhancement by 40% or more and kernel level operation stability. And also, a company introducing DRM can enlarge a range of application programs to which DRM is to be applied without additional module development cost, thus effectively reducing the cost.

[0045] The apparatus and method for managing digital rights according to the present invention can prevent decryption in information technology leakage and allow only authorized people to use the corresponding document, using DRM processing, such as user rights and encryption, on electronic documents or assets with electronic file format corresponding to the electronic document, in response to increasing private information, confidential information asset, and industrial technology leakages and developing hacking and information leakage techniques. The present invention is not dependent on a specific application program, thus performing DRM processing on new products or versions of all application programs and immediately responding to the leakage of electronic documents. Furthermore, limitation in application programs to which DRM is to be applied is minimized, thereby widely expanding application range of DRM, compared with existing methods.

[0046] The invention can also be embodied as computer readable codes on a computer readable recording medium. The computer readable recording medium is any data storage device that can store data which can be thereafter read by a

computer system. Examples of the computer readable recording medium include read-only memory (ROM), random-access memory (RAM), CD-ROMs, magnetic tapes, floppy disks, optical data storage devices, and carrier waves (such as data transmission through the Internet). The computer readable recording medium can also be distributed over network coupled computer systems so that the computer readable code is stored and executed in a distributed fashion.

[0047] The present invention has been particularly shown and described with reference to preferred embodiments thereof. The invention should not be construed as being limited to the embodiments set forth herein. It will be understood by a person skilled in the art that various changes in form and details may be made therein without departing from the spirit and scope of the present invention as defined by the following claims.

WHAT IS CLAIMED IS:

1. An apparatus for managing digital rights, the apparatus comprising:

an DRM agent unit managing application programs to which DRM (Digital Rights Management) is to be applied and a right to contents processed by the application programs to which DRM is to be applied and encrypting or decrypting the content processed by the application programs to which DRM is to be applied;

a DRM rights management unit authenticating a user and managing a user right to the contents processed by the application programs to which DRM is to be applied; and

a kernel API (Application Program Interface) hooking unit monitoring input/output of a file through hooking of kernel native APIs, requesting the DRM rights management unit to verify a user right to contents to be processed when the contents are detected as being processed by the application programs to which DRM is to be applied, and requesting the DRM agent unit to encrypt or decrypt the contents to be processed when the user right to the contents to be processed is verified,

wherein the DRM agent unit and the DRM rights management unit operate at user level, and the kernel API hooking unit operates at kernel level,

wherein the DRM agent unit, the DRM rights management unit, and the kernel API hooking unit are driven for each application program to which DRM is to be applied.

2. The apparatus of claim 1, wherein the DRM agent unit comprises:

a user policy management unit changing rights of a user or rights to contents on the basis of rights information received from the user or a management server managing DRM policies;

a document rights management unit managing a list of application programs to which DRM is to be applied and managing rights information for each document and each application program to which DRM is to be applied;

an injection unit monitoring application programs running on an operating system and driving the DRM agent unit, the DRM rights management unit, and the kernel API hooking unit for each application program when the application program to which DRM is to be applied is running; and

an encryption/decryption unit encrypting or decrypting the contents and returning

the encrypted or decrypted contents to the kernel API hooking unit in response to the request for encryption or decryption of the contents from the kernel API hooking unit.

3. The apparatus of claim 2, wherein the rights information received from the management server comprises rights information for each content or application program to which DRM is to be applied, information on the application programs to which DRM is to be applied, and information on allowance/limitation/change of the user right.

4. The apparatus of claim 2, wherein, when an application program is detected as being running, the injection unit inquires of the document rights management unit whether the running application program is the application program to which DRM is to be applied.

5. The apparatus of claim 1, wherein the DRM rights management unit comprises:

a user authentication unit verifying the user right to the contents or the application programs to which DRM is to be applied and outputting an authentication result on the user; and

a user rights management unit managing the user rights to read, write, modification, capture, clipboard use, and print on the basis of the authentication result on the user and returning a verification result in response to a request for verifying of the right to the contents to be processed from the kernel API hooking unit.

6. The apparatus of claim 5, wherein the DRM rights management unit further comprises a use history management unit managing a use history including opening, modification, and storage of the contents to which DRM is applied.

7. The apparatus of claim 1, wherein the kernel API hooking unit comprises:

a rights control unit requesting the DRM agent unit and the DRM rights management unit to verify the user right to the contents corresponding to file input/output events and outputting the verification result; and

an encryption/decryption control unit hooking the kernel native APIs to detect the file input/output events, requesting the DRM rights management unit to verify the user

right to the contents to be processed corresponding to the file input/output events, and delivering the contents to be processed to the DRM agent unit to direct encryption or decryption on the basis of the verification result.

8. The apparatus of claim 1, wherein the kernel API hooking unit comprises:

a rights control unit hooking the kernel native APIs to detect file input/output events, requesting the DRM agent unit and the DRM rights management unit to verify the user right to the contents to be processed corresponding to the detected file input/output events, and outputting the verification result; and

an encryption/decryption control unit delivering the contents to be processed to the DRM agent unit on the basis of the verification result to direct encryption or decryption.

9. The apparatus of claim 7 or 8, wherein the kernel API hooking unit further comprises a cache memory and stores previously encrypted or decrypted contents delivered from the encryption/decryption control unit in the cache memory in order to prevent an opened document from being encrypted or decrypted repeatedly, and when an inquiry whether the contents to be processed have been encrypted or decrypted is input from the encryption/decryption control unit, searches the cache memory and returning the contents or outputting a notification that the contents has been not decrypted.

10. A method for managing digital rights performed in an apparatus for managing digital rights, the method comprising:

(a) injecting a DRM (Digital Rights Management) engine module into an application program to which DRM is to be applied when the application program to which DRM is to be applied is driven;

(b) monitoring input/output of a file through hooking a kernel native API (Application Program Interface), and when a content is detected as being processed by the application program to which DRM is to be applied, requesting for verifying a user right to the content to be processed in the application program to which DRM is to be applied;

(c) selectively encrypting or decrypting the content to be processed when the user right to the content to be processed is verified; and

(d) returning the encrypted or decrypted content to the kernel native API,

wherein step (a) and step (c) are performed at user level, and step (b) and step (d)

are performed at kernel level.

11. The method of claim 10, wherein step (a) comprises injecting the DRM engine module into the detected application program, when the detected application program is in a predetermined list of application programs to which DRM is to be applied.

12. The method of claim 10, wherein the user right set to the user or the content is changeable on the basis of rights information received from the user or a management server managing DRM policies.

13. The method of claim 12, wherein the rights information received from the management server comprises at least one information out of rights information for each content, rights information for each application program to which DRM is to be applied, information on the application program to which DRM is to be applied, and information on allowance/limitation/change of the user right.

14. The method of claim 10, further comprising (e) updating a use history which includes opening, modification, and storage of the content to which DRM is applied.

15. The method of claim 10, wherein step (c) comprises:

(c1) hooking the kernel native API to detect a file input/output event;

(c2) verifying the user right to the content to be processed corresponding to the file input/output event, and requesting a rights management module operated at user level to verify the user and manage the user right to the content to be processed in the application program to which DRM is to be applied; and

(c3) delivering the content to be processed to an encryption/decryption module operated at user level on the basis of the verification result by the rights management module and instructing to perform encryption or decryption.

16. The method of claim 15, wherein step (c) further comprises (c4) storing previously encrypted or decrypted contents in a cache memory in order to prevent an opened document from being encrypted or decrypted repeatedly, searching the cache memory when encryption or decryption is performed to the contents to be processed, and

reading out the contents corresponding to the contents to be processed from the cache memory instead of performing step (c3).

17. A program for executing the contents transmitting method according to any one of claims 10 to 16 in a computer.

18. A computer-readable recording medium storing the program according to claim 17.