US 20030196109A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2003/0196109 A1

Raley et al. (43) Pub. Date: Oct. 16, 2003

(54) METHOD AND APPARATUS FOR CONTENT TRANSACTION AGGREGATION

(75) Inventors: **Michael Raley**, Downey, CA (US); **Daniel Chen**, Torrance, CA (US); **Hsi-Cheng Wu**, Alhambra, CA (US); **Thanh Ta**, Huntington Beach, CA (US)

Correspondence Address:
NIXON PEABODY, LLP
8180 GREENSBORO DRIVE
SUITE 800
MCLEAN, VA 22102 (US)

(73) Assignee: **CONTENTGUARD HOLDINGS, INC.**, Wilmington, DE

(21) Appl. No.: **10/425,933**

(22) Filed: **Apr. 30, 2003**

Related U.S. Application Data

(63) Continuation of application No. 10/046,670, filed on Jan. 16, 2002, which is a continuation-in-part of application No. 09/649,841, filed on Aug. 28, 2000.

(60) Provisional application No. 60/261,803, filed on Jan. 17, 2001.

Publication Classification

(51) **Int. Cl.**[7] **H04L 9/00**

(52) **U.S. Cl.** **713/200**; 705/52

(57) **ABSTRACT**

A method and apparatus for processing transactions related to content distribution. A content request is made through a user device and, in response to the request, the user device receives a direction to request a security component required to render the content. The user device then requests the security component from a computer associated with a transaction aggregator and billing information of the user is collected. The requested security component is returned to the user device and a protected instance of a rendering engine is created on the user device with the security component. The content is then rendered within the protected instance and the rendering operation is reported to the transaction aggregator.

A.

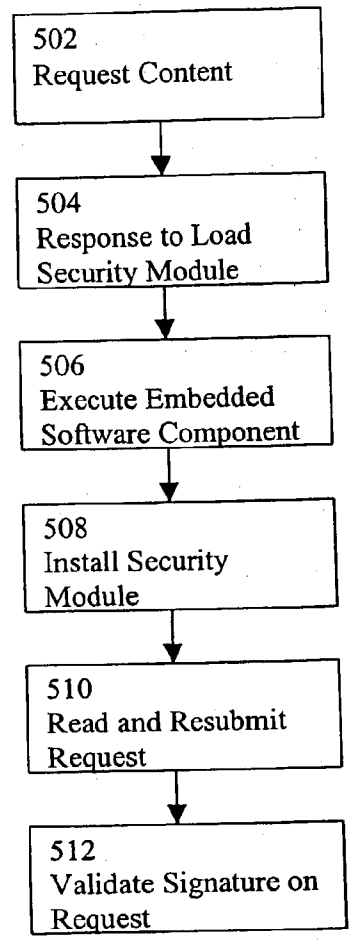


Fig. 1

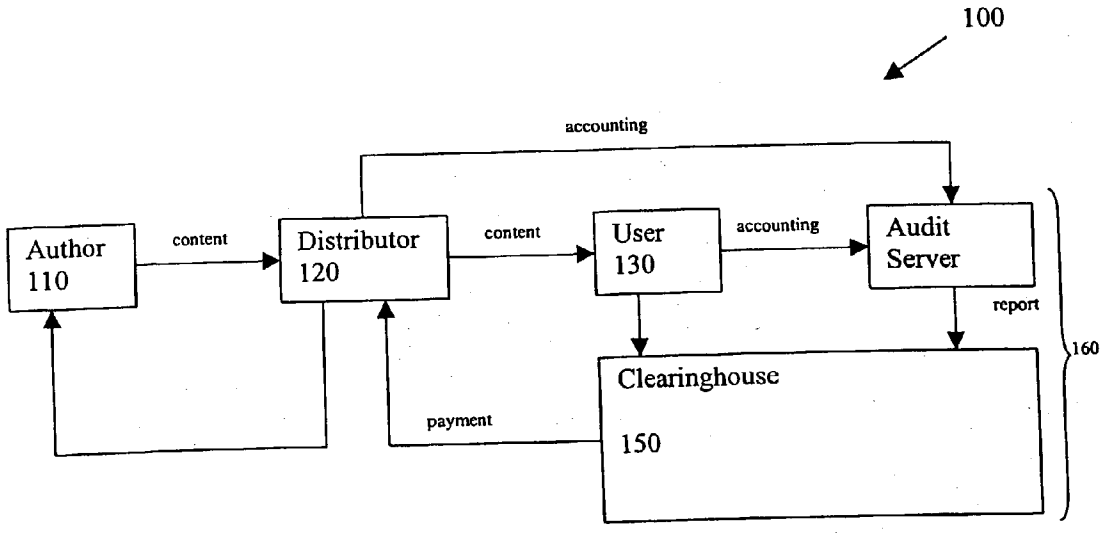

100
accounting
Author
110
content
Distributor
120
content
User
130
accounting
Audit
Server
report
160
Clearinghouse
150
payment

Fig. 2

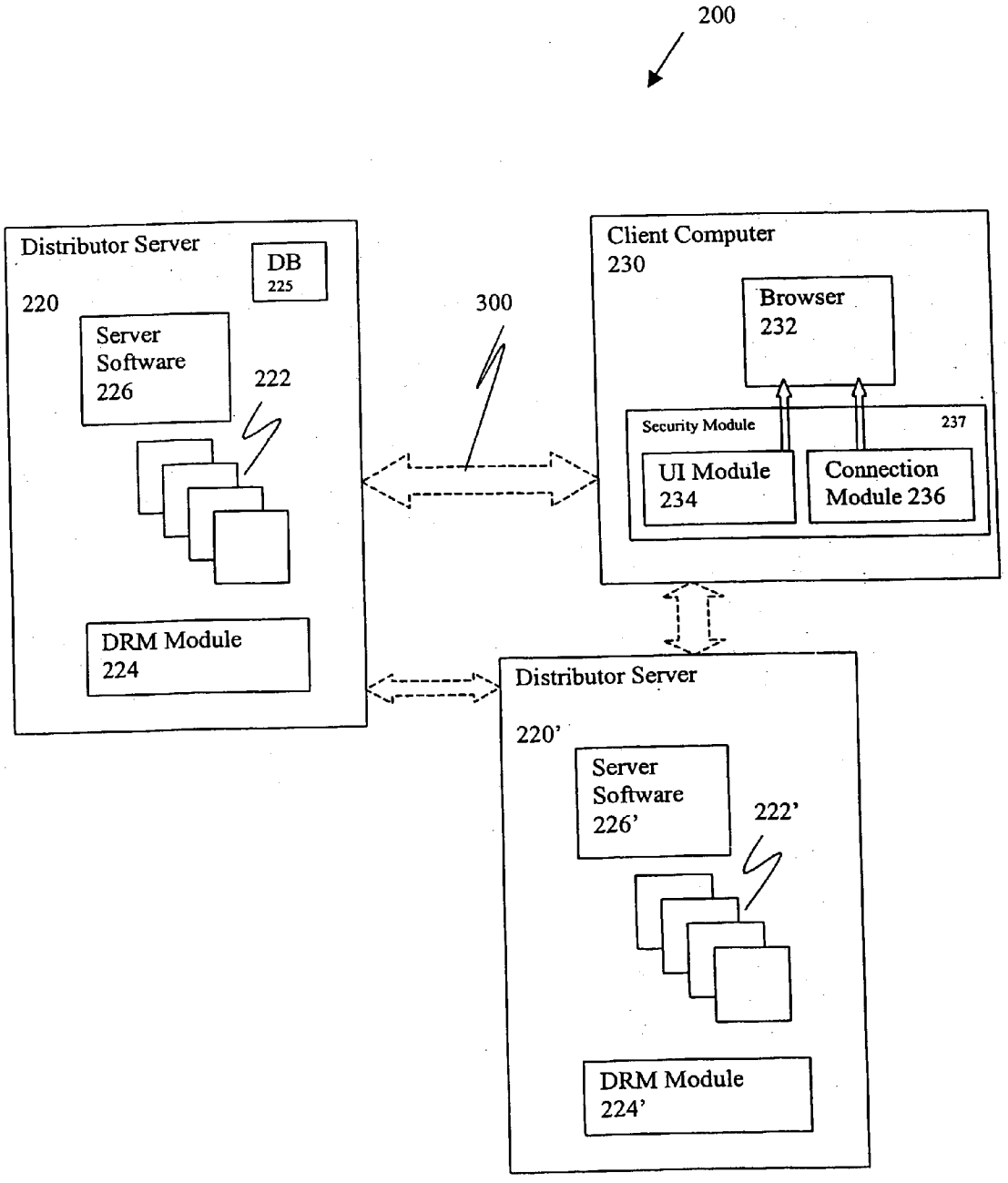

200
Distributor Server
220
DB
225
Server
Software
226
222
DRM Module
224
300
Client Computer
230
Browser
232
Security Module
237
UI Module
234
Connection
Module 236
Distributor Server
220'
Server
Software
226'
222'
DRM Module
224'

Fig. 3

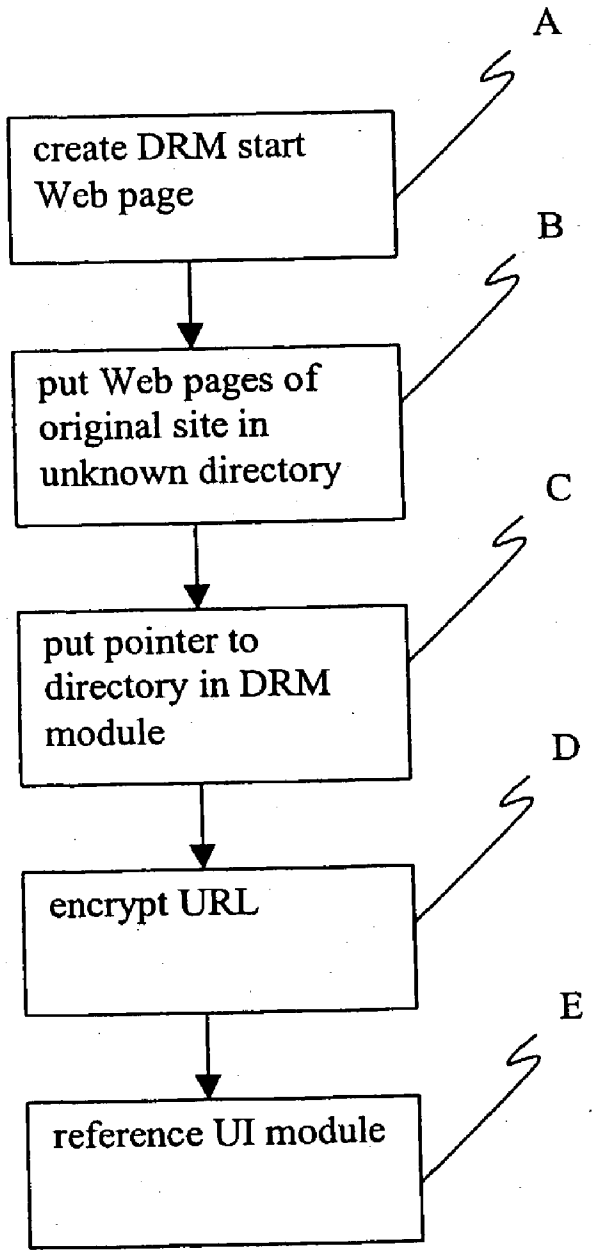

A
create DRM start Web page
B
put Web pages of original site in unknown directory
C
put pointer to directory in DRM module
D
encrypt URL
E
reference UI module

Fig. 4

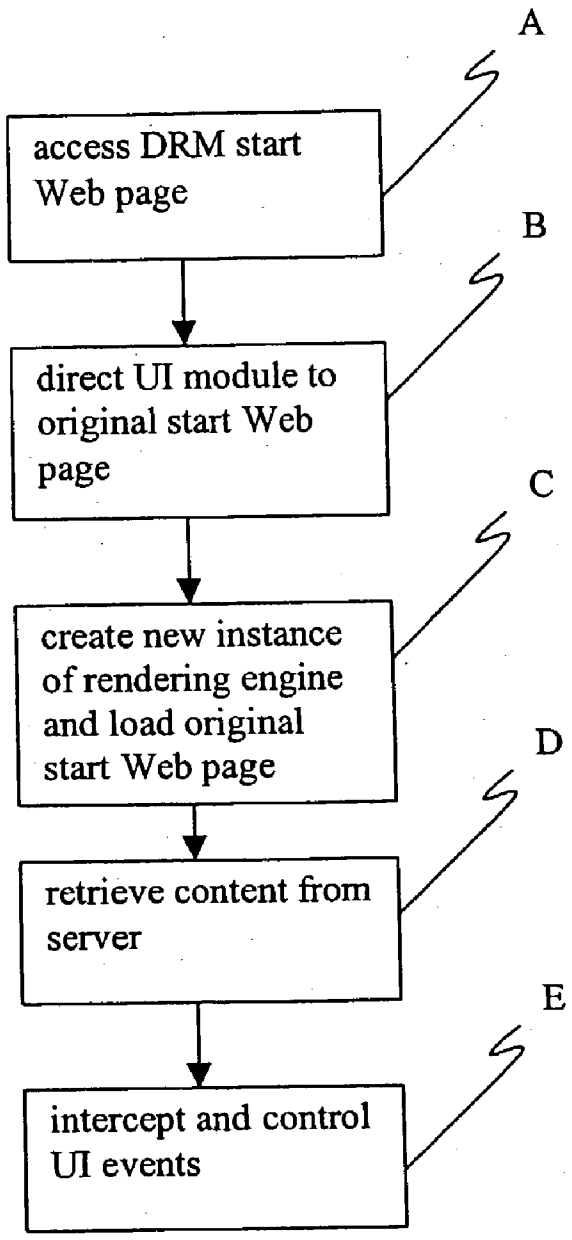

A
access DRM start Web page
B
direct UI module to original start Web page
C
create new instance of rendering engine and load original start Web page
D
retrieve content from server
E
intercept and control UI events

Fig. 5

A.

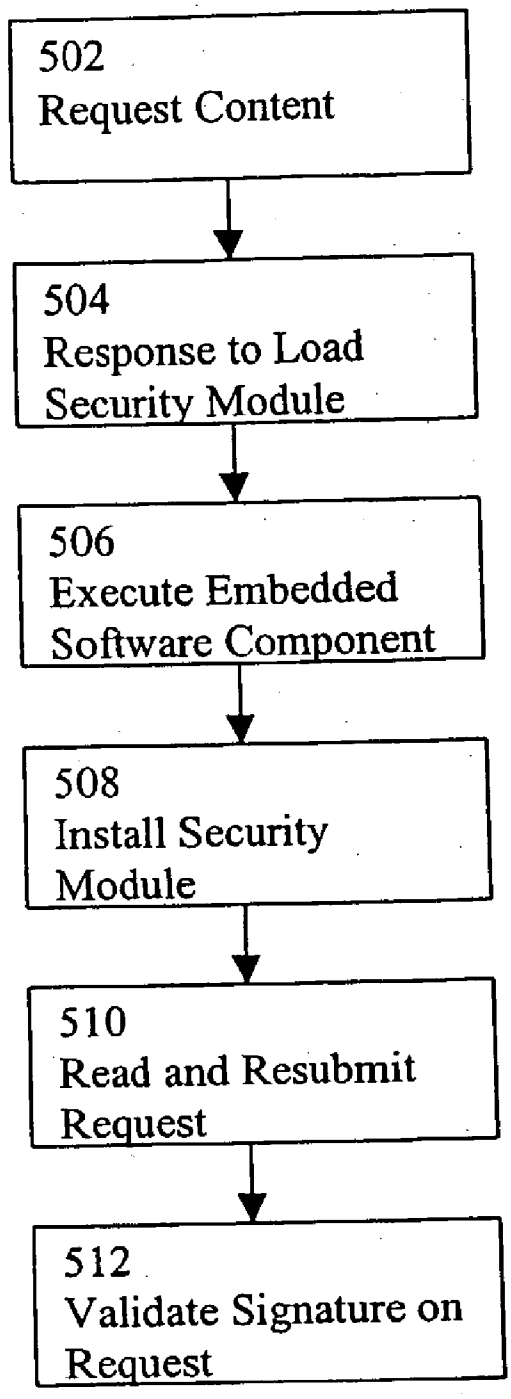

502
Request Content
504
Response to Load Security Module
506
Execute Embedded Software Component
508
Install Security Module
510
Read and Resubmit Request
512
Validate Signature on Request

Fig. 6

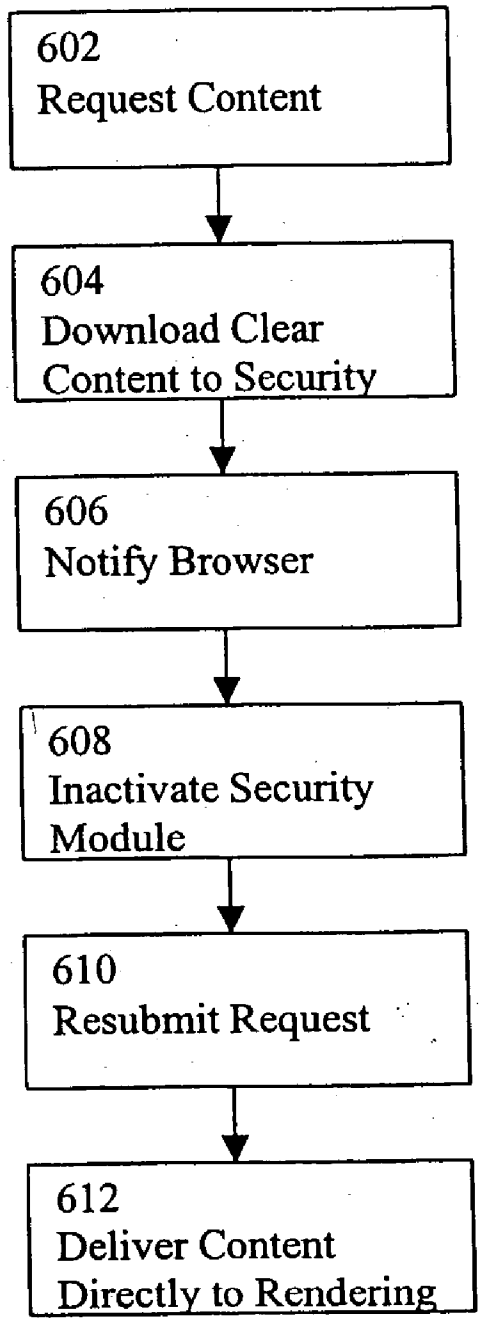

602
Request Content
604
Download Clear Content to Security
606
Notify Browser
608
Inactivate Security Module
610
Resubmit Request
612
Deliver Content Directly to Rendering

Fig. 7

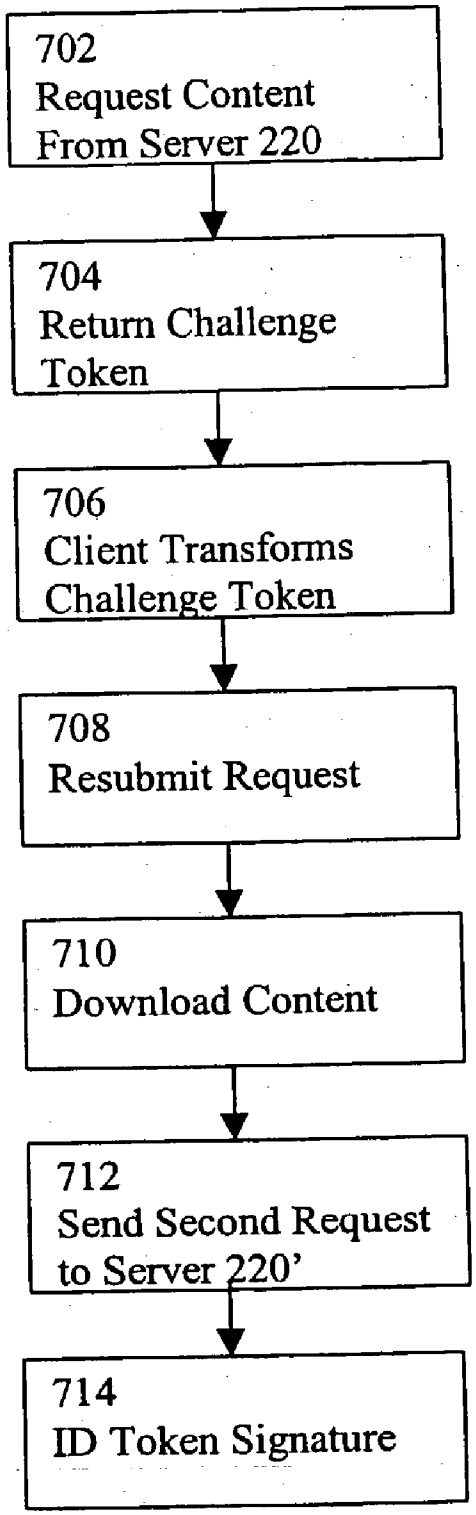

702
Request Content From Server 220
704
Return Challenge Token
706
Client Transforms Challenge Token
708
Resubmit Request
710
Download Content
712
Send Second Request to Server 220'
714
ID Token Signature

Fig. 8

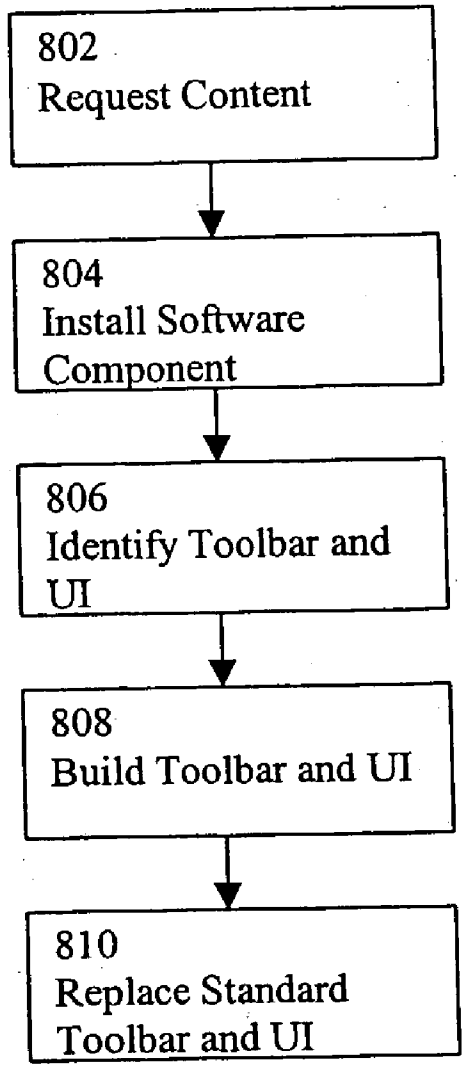

802
Request Content
804
Install Software Component
806
Identify Toolbar and UI
808
Build Toolbar and UI
810
Replace Standard Toolbar and UI

Fig. 9

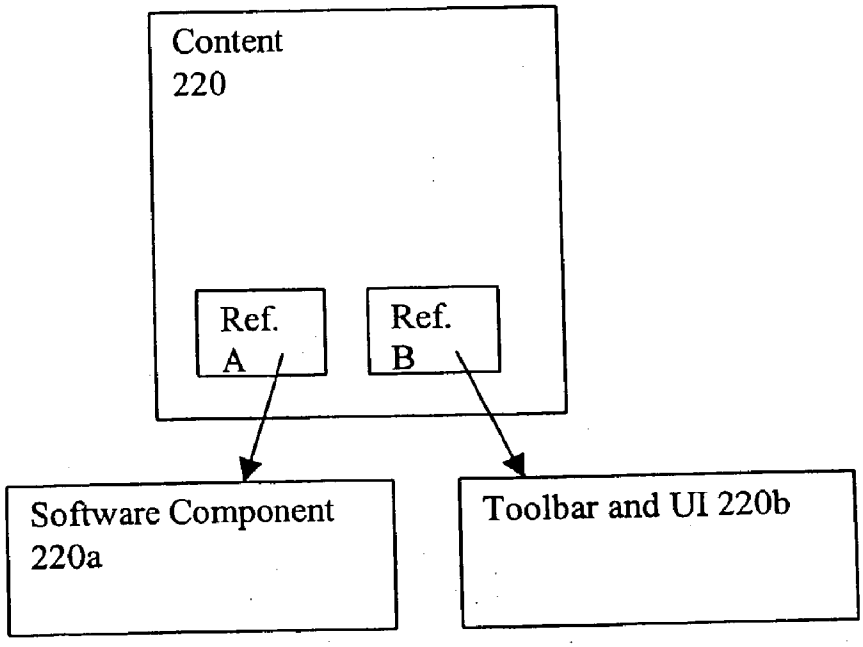

Content
220
Ref.
A
Ref.
B
Software Component
220a
Toolbar and UI 220b

Fig. 10

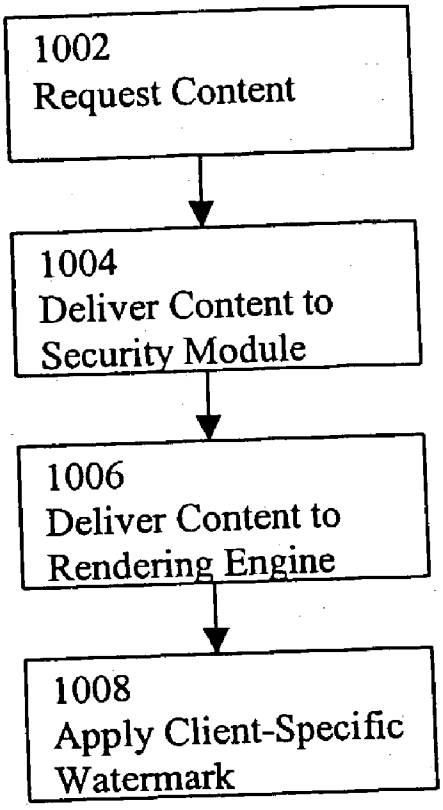

1002
Request Content
1004
Deliver Content to
Security Module
1006
Deliver Content to
Rendering Engine
1008
Apply Client-Specific
Watermark

Fig. 11

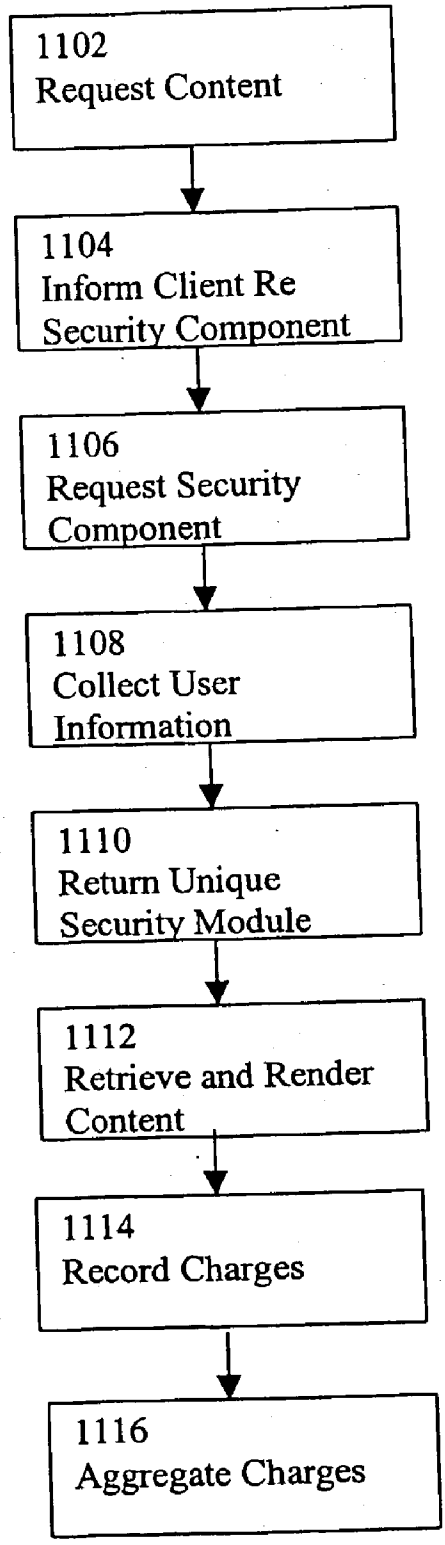

1102
Request Content
1104
Inform Client Re
Security Component
1106
Request Security
Component
1108
Collect User
Information
1110
Return Unique
Security Module
1112
Retrieve and Render
Content
1114
Record Charges
1116
Aggregate Charges

Fig. 12

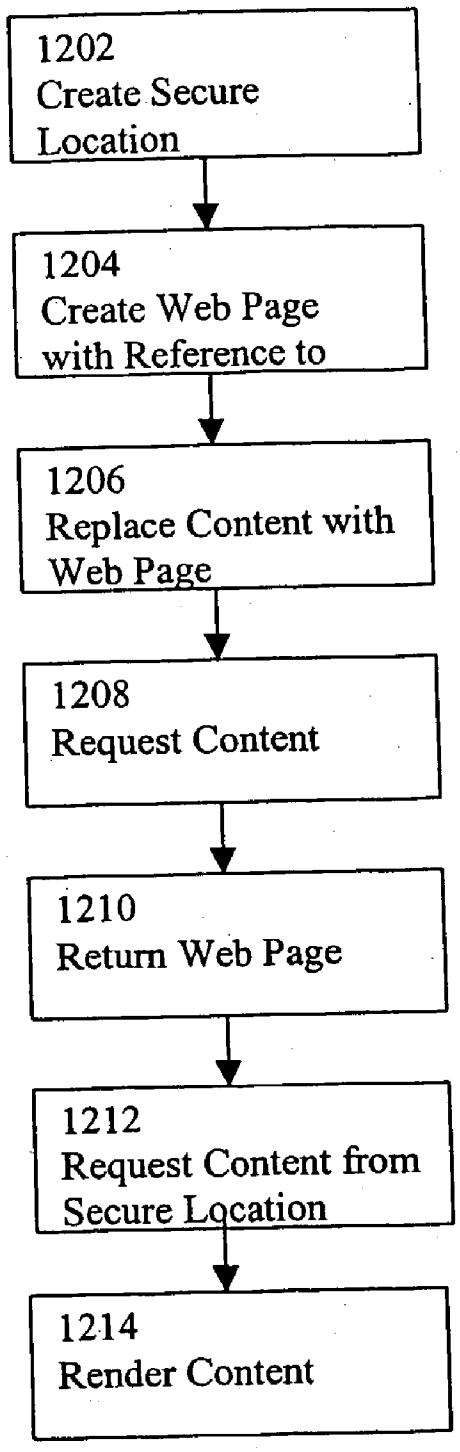

1202
Create Secure Location
1204
Create Web Page with Reference to
1206
Replace Content with Web Page
1208
Request Content
1210
Return Web Page
1212
Request Content from Secure Location
1214
Render Content

Fig. 13

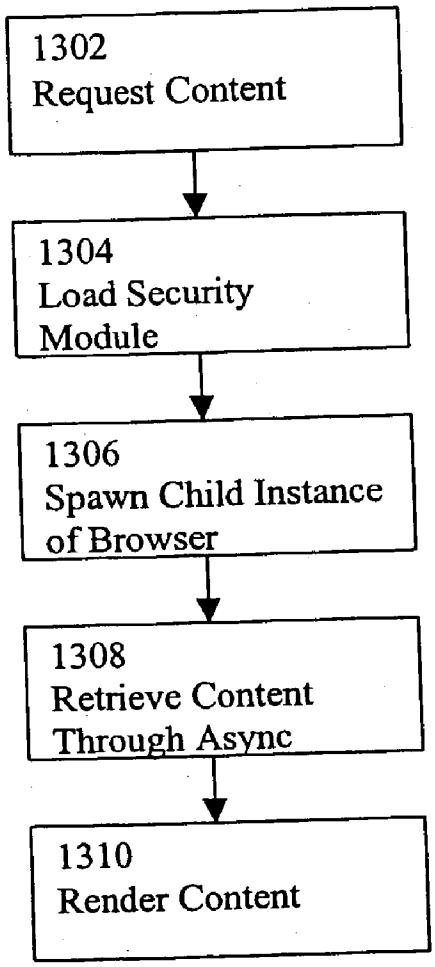

1302
Request Content
1304
Load Security Module
1306
Spawn Child Instance of Browser
1308
Retrieve Content Through Async
1310
Render Content

Fig. 14

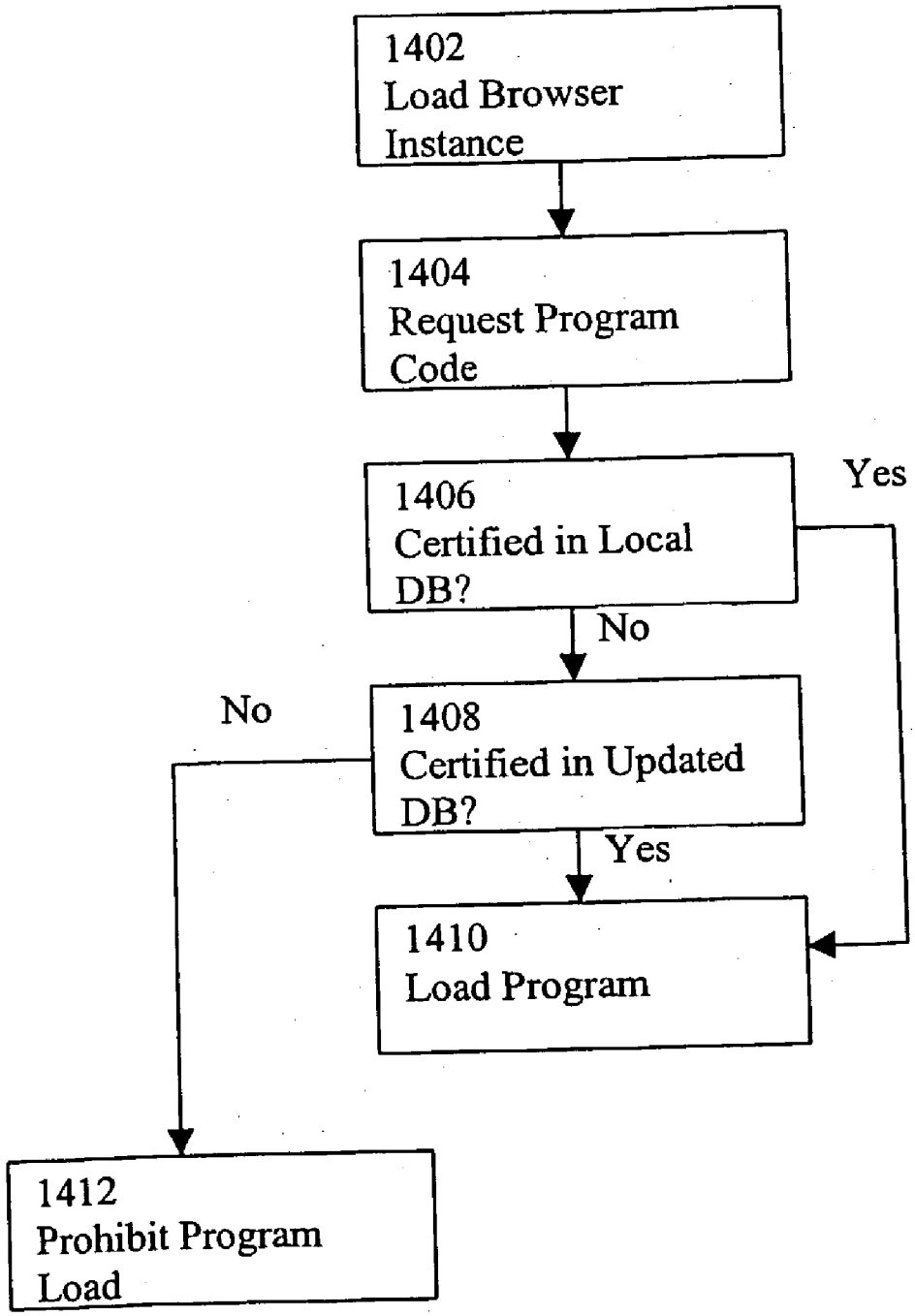

1402
Load Browser Instance
1404
Request Program Code
1406
Certified in Local DB?
Yes
No
1408
Certified in Updated DB?
No
Yes
1410
Load Program
1412
Prohibit Program Load

Fig. 15

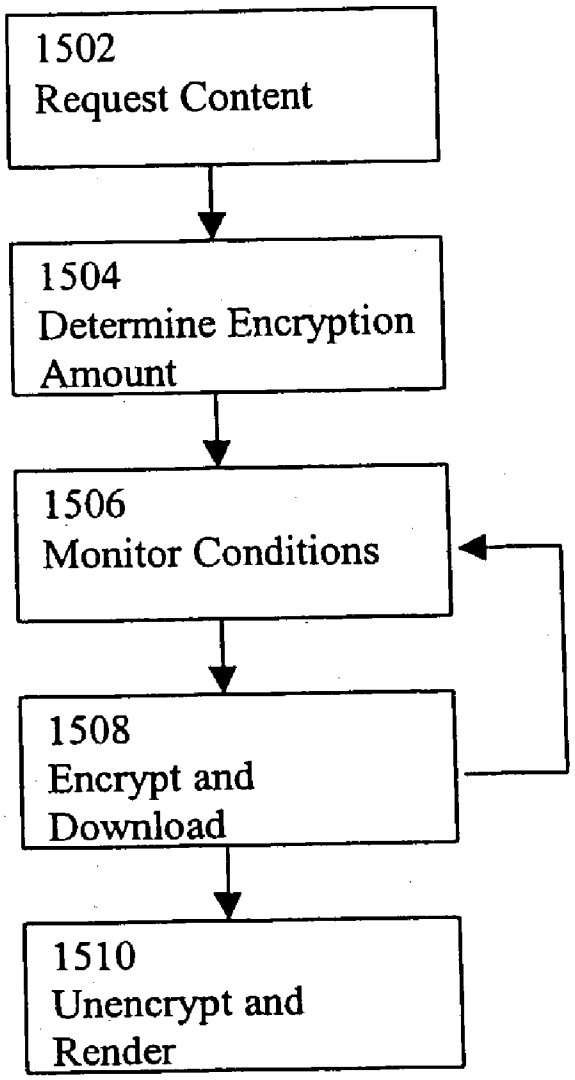

1502
Request Content
1504
Determine Encryption Amount
1506
Monitor Conditions
1508
Encrypt and Download
1510
Unencrypt and Render

Fig. 16

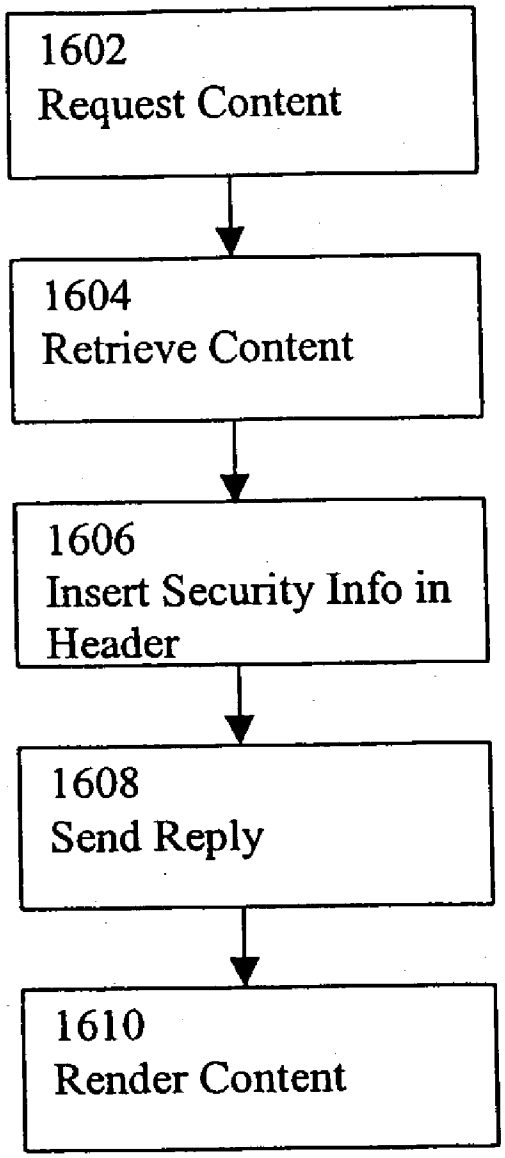

1602
Request Content
1604
Retrieve Content
1606
Insert Security Info in Header
1608
Send Reply
1610
Render Content

Fig. 17

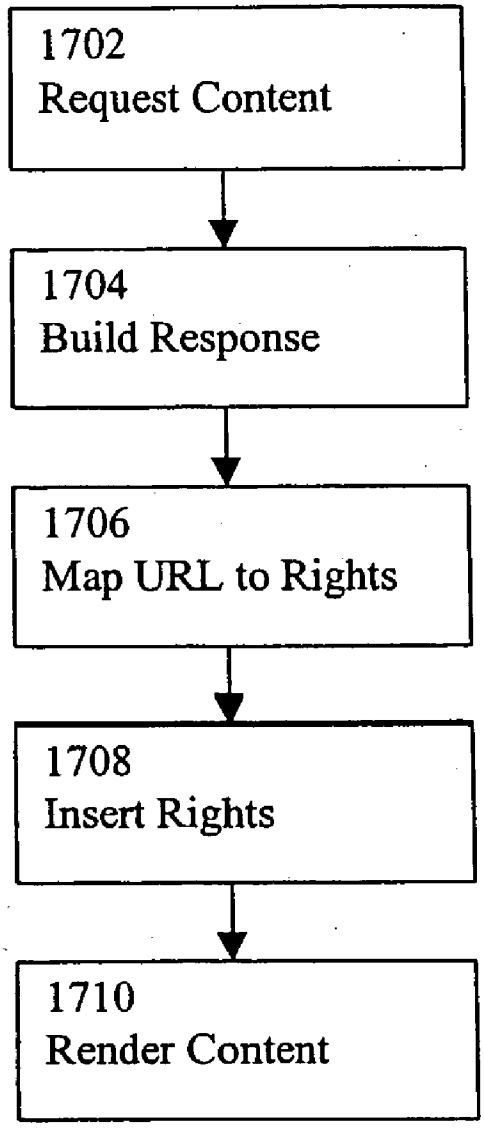

1702
Request Content
1704
Build Response
1706
Map URL to Rights
1708
Insert Rights
1710
Render Content

Fig. 18

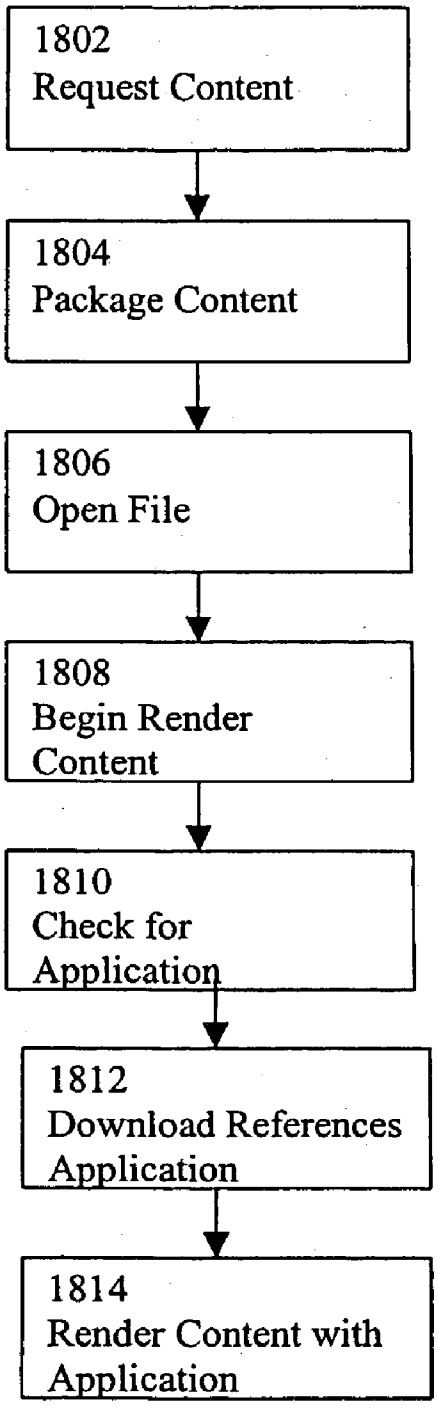
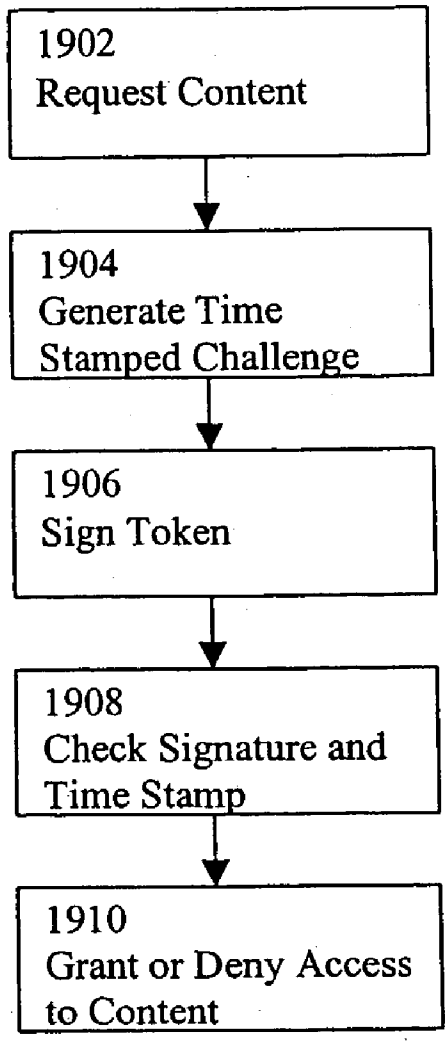

1802
Request Content
1804
Package Content
1806
Open File
1808
Begin Render Content
1810
Check for Application
1812
Download References Application
1814
Render Content with Application

Fig. 19


1902
Request Content
1904
Generate Time
Stamped Challenge
1906
Sign Token
1908
Check Signature and
Time Stamp
1910
Grant or Deny Access
to Content

METHOD AND APPARATUS FOR CONTENT TRANSACTION AGGREGATION

RELATED APPLICATION DATA

[0001] This application is a Continuation of application Ser. No. 10/046,670 filed Jan. 16, 2002, which claims benefit of priority to Provisional Application Serial No. 60/261,803 filed on Jan. 17, 2001 and is a Continuation-In-Part of application Ser. No. 09/649,841 filed Aug. 28, 2000, the entire disclosures of all of which are incorporated herein by reference.

FIELD OF THE INVENTION

[0002] The invention relates to distribution of digital content, and more particularly, to a method and apparatus for facilitating distribution of protected documents displayed with the rendering engine of a standard application program, such as an Internet Web Browser.

BACKGROUND OF THE INVENTION

[0003] The Internet is a worldwide network of computers linked together by various hardware communication links all running a standard suite of protocols known as TCP/IP (transmission control protocol/Internet protocol). The growth of the Internet over the last several years has been explosive, fueled in the most part by the widespread use of software tools (known as "browsers") which allow both HTML (hypertext markup language) viewing and HTTP (hypertext transfer protocol) navigation. Browsers allow a simple GUI (graphical user interface) to be used to communicate over the Internet. Browsers generally reside on the computer used to access content on the Internet, i.e. the client computer. HTTP is a component on top of TCP/IP and provides users access to documents of various formats using the standard page description language known as HTML and more recently XML (extensible markup language) and XHTML (extensible hypertext markup language), a reformulation of HTML into XML. The collection of servers on the Internet using HTML/HTTP has become known as the "World Wide Web" or simply the "Web."

[0004] Through HTML, XHTML, and interactive programming protocols, the author of content is able to make the content available to others by placing the content, in the form of a Web page, on an Internet Web server. The network path to the server is identified by a URL (Uniform Resource Locator) and, generally, any client running a Web browser can access the Web server by using the URL. A client computer running a browser can request a display of a Web page stored on a Web server by issuing a URL request through the Internet to the Web in a known manner.

[0005] Since the Web utilizes standard protocols and a standard rendering engine, i.e. the rendering engine of the browser, the Web has become ubiquitous. One of the primary applications of the Web has been distribution of content in the form of documents. A "document", as the term is used herein, is any unit of information subject to distribution or transfer, including but not limited to correspondence, books, magazines, journals, newspapers, other papers, software, photographs and other images, audio and video clips, and other multimedia presentations. A document may be embodied in printed form on paper, as digital data on a storage medium, or in any other known manner on a variety of media.

[0006] However, one of the most important issues impeding the widespread distribution of digital documents, i.e. documents in forms readable by computers, via electronic means, and the Internet in particular, is the current lack of protection of the intellectual property rights of content owners during the distribution and use of those digital documents. Efforts to resolve this problem have been termed "Intellectual Property Rights Management" ("IPRM"), "Digital Property Rights Management" ("DPRM"), "Intellectual Property Management" ("IPM"), "Rights Management" ("RM"), and "Electronic Copyright Management" ("ECM"), collectively referred to as "Digital rights management (DRM)" herein.

[0007] In the world of printed documents, a work created by an author is usually provided to a publisher, which formats and prints numerous copies of the work. The copies are then sent by a distributor to bookstores or other retail outlets, from which the copies are purchased by end users. While the low quality of copying and the high cost of distributing printed material have served as deterrents to unauthorized copying of most printed documents, it is far too easy to copy, modify, and redistribute unprotected digital documents. Accordingly, some method of protecting digital documents is necessary to make it more difficult to copy and distribute them without authorization.

[0008] Unfortunately, it has been widely recognized that it is difficult to prevent, or even deter people from making unauthorized distributions of electronic documents within current general-purpose computing and communications systems such as personal computers, workstations, and other devices connected over communications networks, such as local area networks (LANs), intranets, and the Internet. Many attempts to provide hardware-based solutions to prevent unauthorized copying have proven to be unsuccessful. The proliferation of "broadband" communications technologies (NII) will render it even more convenient to distribute large documents electronically, including video files such as full length motion pictures, and thus will remove any remaining deterrents to unauthorized distribution of documents. Accordingly, DRM technologies are becoming very useful.

[0009] Two basic schemes have been employed to attempt to solve the document protection problem: secure containers and trusted systems. A "secure container" (or simply an encrypted document) offers a way to keep document contents encrypted until a set of authorization conditions are met and some copyright terms are honored (e.g., payment for use). After the various conditions and terms are verified with the document provider, the document is released to the user in clear form. Commercial products such as Cryptolopes by IBM™ and by InterTrust's™ Digiboxes fall into this category. Clearly, the secure container approach provides a solution to protecting the document during delivery over insecure channels, but does not provide any mechanism to prevent legitimate users from obtaining the clear document and then using and redistributing it in violation of content owners' intellectual property.

[0010] Cryptographic mechanisms are typically used to encrypt (or "encipher") documents that are then distributed and stored publicly, and ultimately privately deciphered, i.e. unencrypted, by authorized users. This provides a basic form of protection during document delivery from a document

distributor to an authorized user over a public network, as well as during document storage on an insecure medium.

[0011] In the "trusted system" approach, the entire system is responsible for preventing unauthorized use and distribution of the document. Building a trusted system usually entails introducing new hardware such as a secure processor, secure storage and secure rendering devices. This also requires that all software applications that run on trusted systems be certified to be trusted. While building tamper-proof trusted systems is still a real challenge to existing technologies, current market trends suggest that open and untrusted systems such as PC's and workstations using browsers to access the Web, will be the dominant systems used to access copyrighted documents. In this sense, existing computing environments such as PC's and workstations equipped with popular operating systems (e.g., Windows™, Linux™, and UNIX) and rendering applications such as browsers are not trusted systems and cannot be made trusted without significantly altering their architectures. Of course, alteration of the architecture defeats a primary purpose of the Web, i.e. flexibility and compatibility.

[0012] U.S. Pat. No. 5,715,403, the disclosure of which is incorporated herein by reference, discloses a system for controlling the distribution of digital documents. Each rendering device has a repository associated therewith. Usage rights labels are associated with digital content. The labels include usage rights that specify a manner of use of the content and any conditions precedent for exercising the manner of use. U.S. Pat. No. 5,052,040 discloses the use of a label prefixed to digital files so that different users can have specific encryption capability and rights with respect to the same file.

[0013] Two basic approaches have been taken to control the distribution of documents over the Web. The first approach is the use of subscription based services in which the user is only granted access to content after paying a subscription fee. However, once the subscription fee is paid and the document is rendered by the browser, the user can copy, print, and modify the document, i.e. all control of the document by the publisher is lost.

[0014] The second approach is to utilize proprietary formats wherein the document can only be rendered by a select rendering engine that is obligated to enforce the publisher's rights. Of course, this approach requires the use of a single proprietary format and loses the ability to combine plural popular formats and the richness of content associated therewith. Further, this approach requires the user to use a proprietary rendering application that must be obtained and installed on the user's computer and requires development of the rendering application for each format to be rendered in a secure manner. Further, the documents must be generated or converted using non-standard tools.

[0015] Further, there are various known mechanisms by which functionality can be added to a standard rendering engine, such as a Web browser. For example, an ActiveX control can be automatically downloaded and executed by a Web browser. ActiveX is a set of rules for how applications should share information and ActiveX controls can be developed in a variety of programming languages, including C, C++, Visual Basic, and Java.

[0016] An ActiveX control is similar to a Java applet. Unlike Java applets, however, ActiveX controls have full access to the Windows™ operating system. Microsoft™ has developed a registration system so that browsers can identify and authenticate an ActiveX control before downloading it. Java applets can run on all platforms, whereas ActiveX controls are currently limited to Windows environments.

[0017] A scripting language called VBScript enables Web authors to embed interactive elements in HTML documents to initiate a download and installation of ActiveX controls and other functions. Currently, Microsoft's Web browser, Internet Explorer™, supports Java, JavaScript, and ActiveX, whereas Netscape's Navigator™ browser supports only Java and JavaScript, though its plug-ins can enable support of VBScript and ActiveX. However, the availability of various plug-in and add-on software for browsers further complicates the user experience and presents a variety of problems in implementing a reliable DRM system over the Web or other open networks.

[0018] VYOU.COM has developed a system for protecting intellectual property in documents distributed over the Web. The system includes a software plug-in, to the user's Web browser. The plug-in includes a proprietary rendering engine for the proprietary format in which documents are represented and transmitted. Accordingly, documents must be reformatted into the proprietary format and the plug-in rendering engine for the appropriate final viewing format is used in place of the standard browser rendering engine. This arrangement requires the rendering engine for each format must be developed. Therefore, this system is difficult to implement and loses the advantages of the Web as an open architecture.

[0019] The proliferation of the Web, and its usefulness in document distribution, makes it desirable to apply DRM features to Web browsers and other standard rendering engines without requiring the rendering engines to be rewritten. However, conventional DRM technologies are not easily adapted to use with Web browsers and other standard rendering engines because they require proprietary formats and rendering engines which contradict the open architecture of the Web. The inability to control application programs, such as Web browsers, independently from their rendering engines has made it difficult to apply DRM features over distribution networks.

[0020] Another roadblock to implementing DRM systems over the Web is the fact that often the fees paid for proprietary documents, particularly for limited rights in proprietary documents are relatively small. For example, in many cases, the fees for limited rights in proprietary documents may be less that one dollar ($1.00) U.S. In such cases, the expense associated with processing a credit card charge, including access fees, transaction fees, and the like are relatively large as compared to the entire document fee. For such relatively small transactions, often referred to as "micro-transactions," the use of a credit card for the corresponding "micropayment", i.e. relatively small payment, is not practical. Further, since each credit card transaction is processed as an individual charge, a customer purchasing a large volume of documents in various transactions, will generate a large number of small transactions which is not efficient for credit card transactions.

[0021] Various proprietary solutions have been developed for handling micropayments, and other payments, over the Internet. For example, CyberCash™, Inc. and ePayment

Systems™, Inc. each provide such solutions. Also, Intellicent™ provides a specific solution for micropayments. However, these solutions are not integrated in a DRM environment.

SUMMARY OF THE INVENTION

[0022] An aspect of the invention is a method of processing transactions related to content distribution. The method comprises requesting content with a user device, receiving a direction to request a security component required to render the content, the user device requesting the security component from a computer associated with a transaction aggregator, collecting billing information of a user of the user device, returning the requested security component to the user device, creating a protected instance of a rendering engine on the user device with the security component, rendering the content within the protected instance, and reporting the rendering operation to the transaction aggregator.

BRIEF DESCRIPTION OF THE DRAWING

[0023] The invention is described through a preferred embodiment and the attached drawing in which:

[0024] **FIG. 1** is a block diagram of a document distribution system utilizing DRM technology;

[0025] **FIG. 2** is a schematic representation of a DRM system of the preferred embodiment;

[0026] **FIG. 3** is a flowchart of a method of operation of the preferred embodiment for causing the server to respond only to a protected client;

[0027] **FIG. 4** is a flowchart of a method of operation of the preferred embodiment for accessing protected content;

[0028] **FIG. 5** is a flowchart of a method of operation of the preferred embodiment for installing a security module;

[0029] **FIG. 6** is a flowchart of a method of operation of the preferred embodiment for inactivating a security module;

[0030] **FIG. 7** is a flowchart of a method of operation of the preferred embodiment for facilitating authentication of a client for multiple servers;

[0031] **FIG. 8** is a flowchart of a method of operation of the preferred embodiment for permitting a service to control the user interface of a client;

[0032] **FIG. 9** is a block diagram of content having references to user interface components;

[0033] **FIG. 10** is a flowchart of a method of operation of the preferred embodiment for applying client-specific watermarks;

[0034] **FIG. 11** is a flowchart of a method of operation of the preferred embodiment for aggegating transaction information;

[0035] **FIG. 12** is a flowchart of a method of operation of the preferred embodiment for address obfuscation;

[0036] **FIG. 13** is a flowchart of a method of operation of the preferred embodiment for using an asynchronous protocol for HTTP document transfer;

[0037] **FIG. 14** is a flowchart of a method of operation of the preferred embodiment for dynamic certification of software;

[0038] **FIG. 15** is a flowchart of a method of operation of the preferred embodiment for dynamic variable encryption;

[0039] **FIG. 16** is a flowchart of a method of operation of the preferred embodiment for embedding security information in a document;

[0040] **FIG. 17** is a flowchart of a method of operation of the preferred embodiment for determining usage rights based on a requesting URL;

[0041] **FIG. 18** is a flowchart of a method of operation of the preferred embodiment for downloading necessary rendering applications; and

[0042] **FIG. 19** is a flowchart of a method of operation of the preferred embodiment for time stamping validation tokens.

DETAILED DESCRIPTION

[0043] The invention is described below with reference to a preferred embodiment. It will be apparent that the invention can be embodied in a wide variety of forms, some of which may be quite different from those of the disclosed embodiment. Consequently, the specific structural and functional details disclosed herein are merely representative and do not limit the scope of the invention.

[0044] **FIG. 1** is a block diagram of a model for a system for the electronic distribution of digital documents. Author **110** creates original content **112** and passes it to distributor **120** for distribution. Ordinarily, author **110** is the creator of the content. However, the term "author" as used herein can be the creator, owner, editor, or other entity controlling the content or an agent (e.g. a publisher) of one of those entities. Also author **110** may distribute documents directly, without involving another party such as distributor **120**, and thus the author and distributor may be the same entity. However, the division of functions set forth in **FIG. 1** is more efficient, as it allows author **110** to concentrate on content creation and not the administrative functions of distribution. Moreover, such a breakdown facilitates economies of scale by permitting distributor **120** to associate with a number of authors **110**. The term "document", as used herein, generally refers to any type of content, such as text, audio, or other data, including any encryption, formatting, or the like. The term "content", as used herein, generally refers to the underlying information of a document. However, these terms overlap and thus are used interchangeably herein.

[0045] Distributor **120** distributes documents to user **130** upon request. In a typical electronic distribution model, the content is distributed as a document in encrypted form. Distributor **120** encrypts the content with a random key and then encrypts the random key with a public key corresponding to user **130**. Thus the encrypted document is customized solely for the particular user **130**. User **130** is then able to use their private key to unencrypt the random key and use it to unencrypt and view the document.

[0046] Payment for the document is passed from user **130** to distributor **120** by way of clearinghouse **150** which collects requests from user **130** and from other users who wish to view a particular document. Clearinghouse **150** also

collects payment information, such as debit transactions, credit card transactions, or other known electronic payment schemes, and forwards the collected payments as a payment batch to distributor **120**. Of course, clearinghouse **150** may retain a share of the payment as a fee for the above-noted services. Distributor **120** may retain a portion of the batch payment from clearinghouse **150** for distribution services and forward a payment (for example royalties) to author **110**. Distributor **120** may await a bundle of user requests for a single document before distributing the document. In such a case, a single encrypted document can be generated for unencryption by all of the requesting users **130**.

[**0047**] Each time user **130** requests (or uses) a document, an accounting message is sent to audit server **140** which ensures that each request by user **130** matches with a document sent by distributor **120**. Accounting information is received by audit server **140** directly from distributor **120**. Any inconsistencies are transmitted via a report to clearinghouse **150**, which can then adjust the payment batches made to distributor **120** accordingly. This accounting scheme is present to reduce the possibility of fraud in electronic document distribution and to handle any time-dependent usage permissions that may result in charges that vary, depending on the duration or other extent of use. Audit server **140** and clearinghouse **150**, in combination, can serve as transaction aggregator **160** which functions to aggregate plural transactions over a period of time, and charge distributor **120** in an appropriate manner to reduce the accounting overhead of distributor **120**. The model for electronic document distribution illustrated in **FIG. 1** can be applied to the electronic document distribution system of the preferred embodiment disclosed herein. Further, content can include usage rights as described above.

[**0048**] **FIG. 2** is a schematic representation of a computer architecture of a document distribution system in accordance with a preferred embodiment of the invention. As noted above, the invention can be used in connection with known models for effecting accounting and payment of fees, such as use of a clearinghouse and an audit server. Further, the invention can be used in connection with various commerce models. Accordingly, the apparatus for auditing distribution, effecting payment, and authoring a document is not described in detail herein and is omitted from the discussion of the preferred embodiment to simplify description thereof.

[**0049**] As illustrated in **FIG. 2**, digital content distribution system **200** comprises distributor server **220**, corresponding to distributor **120** described above, and client computer **230**, corresponding to user **130** described above. Server **220** and client computer **230** can be general purpose computers programmed to accomplish the desired functions. For example, server **220** can be a standard server or workstation running the Windows NT™ operating system and including HTTP server software **226** such as Apache or another HTTP server. Client **230** can be a personal computer running the Windows™ operating system. In the preferred embodiment, server **220** and client **230** are each coupled to communications network **300**, such as the Internet, or more specifically, the Web. Accordingly, client **230** includes browser **232** as a standard application program having a rendering engine. Browser **232** can be any HTTP compliant browser, such as Microsoft Internet Explorer™ or Netscape Navigator™. The phrase "standard application program", as used herein, refers to any application program designed to accomplish a task, such as document creation, viewing and editing, and having a rendering engine. Examples of standard application programs include word processors, Web browsers, editors, viewers, spreadsheet programs, database programs, and the like.

[**0050**] Server **220** has a plurality of documents **222** stored thereon, in the form of Web pages, for distribution. Documents **222** can be stored in an encrypted format. The term "encrypted", as used herein, refers to any mechanism by which accessibility of content is partially or completely prohibited, such as by use of asymmetric or symmetric encryption algorithms, scrambling algorithms, or the like. Server **220** also includes digital rights management module **224**, in the form of software, for storing and managing usage rights associated with particular ones of documents **222**, users, and/or payment amounts or other conditions. Other functions of rights management module **224** are described in greater detail below. Distributor server **220** can be part of a server farm or other group of computers, which can also include distributor server **220'** as illustrated in **FIG. 2**.

[**0051**] Client **230** also has user interface (UI) module **234** and connection module **236** each in the form of software and each adapted to attach to browser **232** without the need for modification of browser **232**. For example, UI module **234** and connection module **236** can be in the form of plug-ins, ActiveX controls, or in any form that allows attachment to the rendering engine of browser **232** without the need for modifying the code of browser **232**. Such attachment is described in greater detail below. In combination, UI module **234** and connection module **236** constitute a security module which is described in detail below. While security module **237** is illustrated as residing in client computer **230**, it will become clear that security module **237** can include client side components and server side components. For example, DRM module **224** described below can be a server side component of security module **237**.

[**0052**] Rights management module **224** is a server side component that can store labels of usage rights and identify which rights are associated with each document **222**. The rights also can vary based on the identity of the user requesting access to document **222**, any payment made by the user through a clearinghouse or the like, and any other conditions. For example, the user may have the option of paying one fee to view document **222** or a higher fee for viewing and printing the same document **222**, as is well known. Rights management module **224** is also operative to deliver the appropriate list of rights along with the document, via communications network **300**, to connection module **236** of client **230** as described below.

[**0053**] Connection module **236** can be a client side software component which verifies the integrity of the environment of client **230** by verifying that UI module **234** is attached to browser **232**, identifies the user of client **230**, i.e. the person requesting content, retrieves the document and the appropriate list of rights sent by rights management module **224**, and in appropriate circumstances, unencrypts any retrieved documents that are encrypted and generates any necessary signatures and/or keys. UI module **234** can be a client side component that monitors requests from the user to access content of documents **222** and either grants or denies the request based on the list of rights retrieved by connection module **236**. Further, UI module **234** can disable

specified functions of browser **232** and the operating system of client **230** based on the list of rights in the manner described below, by interfacing with the operating system API and intercepting and redirecting commands for example. Connection module **236** verifies that the industry standard rendering engine running in the environment of client **230** has not been tampered with or otherwise compromised in a way that may allow the user to access protected content in a way that bypasses UI module **234**.

[**0054**] The invention can be implemented in connection with known client/server networking architectures, such as the Web, without modifying obviating, or bypassing the standard client software, server software, and rendering engines. Rights management module **224** is installed in server **220** along side the existing server software **226**. As noted above, rights management module **224** identifies which rights are associated with documents **222** existing on server **220** or later stored on server **222**. For example. Rights management module **224** can have a programmable database, lookup table or the like including the various rights associated with each document **222** and other variables, such as the identity of the user and the payment made by the user, in a well known manner. Rights management module **224** further interfaces with the operating system API of server **220** to cause server software **226** to only respond to connections from client(s) **230** having the proper components of security module **237**, such as connection module **236** and UI module **234**. Also, rights management module **224** serves as in interface with database **225** described below.

[**0055**] For example, once rights management module **234** is installed the procedure illustrated in **FIG. 3** is accomplished. In step **302**, a new DRM start Web page, or other secure interface display, is created which references UI module **234** and the existing server start Web page. In step **304**, the various Web pages of a Web site on server **220** can be placed in a directory having a random label or any unknown directory. In step **306**, rights management module **224** is programmed to include a pointer to this directory and, in step **308**, rights management module **224** encrypts the URL of this directory. In step **310**, the start DRM Web page is modified to reference UI module **235** which can instruct connection module **236** to unencrypt the encrypted URL to permit access to original start page and the rest of the Web site. If client **230** does not have UI module **234** and connection module **236**, the URL cannot be unecrypted and thus the Web site on server **220** cannot be accessed.

[**0056**] Alternatively, connection module **236** can generate a signature and send the signature to server **220** with any URL request to server **220**. Access to the Web site on server **220** will only be granted if the signature is present and valid. In this alternative, rights management module **224** can include code to validate the signature.

[**0057**] When a user of client computer **230** attempts to access server **220** having rights management module **224**, rights management module **224** verifies if all required components of security module **237** UI module **234**, such as are installed on client **230** as described above. If not, instructions in the DRM start Web page, in the form of a java applet, ActiveX control, or the like, instruct browser **232** to download and install UI module **234** in the manner described in greater detail below. Download can be accomplished from server **220** or another server coupled to communications network **300**. Such download and installation can be accomplished in a known manner using conventional mechanisms, and the user can be prompted to authorize installation and to enter other necessary information, such as where to store the installation files. Connection module **236** can be imbedded in UI module **234** and downloaded and installed simultaneously or through a separate download and installation process. Of course, if UI module **234** is detected as installed on server **230**, the installation step can be skipped. If UI module **234** is not installed on client **230**, and the user does not authorize such installation, access to documents on server **222** is prohibited, or limited only to documents specified as being freely distributable.

[**0058**] As noted above, UI module **234** and connection module **236** are in a form in which they can be attached to browser **232** without the need to modify the code of browser **232**. The term "attached" as used herein with respect to the modules, refers to software modules that can be combined or coupled with browser without modifying the code of browser **232**. For example, UI module **234** and connection module **236** are in the form of plug-ins, in the case of Netscape Navigator™ or ActiveX Controls in the case of Internet Explorer™. The mechanisms for developing and installing such components are well known.

[**0059**] A procedure for accessing protected content, in the form of documents **222**, stored on server **220** is illustrated in **FIG. 4**. In step **402**, the DRM start Web page is accessed through its URL in a known manner. In step **404**, the DRM start Web page directs UI module **234** to the original start page or pages referenced by the DRM start Web page using one of the methods described above. In step **406**, UI module **234** creates another instance of the rendering engine of browser **232**, loads the original start Web page, and instructs the operating system to display the new instance in a browser window, using known techniques. The new instance is directed, by UI module **234**, to retrieve content from server **220** through connection module **236** in step **408**. In other words, in the preferred embodiment, UI module **234** intercepts commands from browser **232** and redirects them through connection module **236**. UI module **234** can instruct the new instance to utilize a secure asynchronous protocol through connection module **236** as describe in greater detail below. Therefore, UI protection is validated and all user interface events, can be intercepted and controlled in step **410**. For example, when the user initiates a "print" or "copy" command through the standard user interface of browser **232**, UI module **234** intercepts the request and only permits response if the set of rights received by connection module **236** permits the requested function to be carried out.

[**0060**] More specifically, when connection module **236** receives a request from the rendering engine of browser **232**, connection module **236** validates that the rendering engine is protected by UI module **234**, i.e. UI module **234** is attached, and that the rendering engine has not been tampered with or otherwise compromised. If so, connection module **236** permits connection to rights management module **224** of server **220** and negotiates permission to retrieve the original start Web page on server **220** and the set of rights for the user for the Web page. Rights management module **224** then initiates a connection between server software **226** of server **220** and connection module **236** of client **230**. The connection can be established using any protocol, such as HTTP or HTTPS or any other standard or proprietary connection protocol.

[0061] The requested document **222** is then retrieved and delivered to connection module **236** which unencrypts document **222**, if encrypted on server **220**, and delivers the document in unencrypted form to the new instance of the rendering engine of browser **232** along with the set of rights associated with the document. Once again, the contents of the set of rights may be determined based on the document, the user's identity, a payment made by the user, or any other appropriate parameter. Connection module **236** then transmits the set of rights to UI module **234** which limits the functions available to the user based on the set of rights by controlling the new instance of the rendering engine of browser **236** as described above.

[0062] The content of the document is now viewable in a window of browser **232** as any other Web page would be. However, browser **232** does not have direct access to the Web page of the document because browser **232** is "wrapped" by UI module **234** or other components of security module **237** as will be described below. UI modules **234** prevents browser **232** from performing any prohibited functions outside of the scope of the set of rights for the document.

[0063] The preferred embodiment utilizes a standard rendering engine of an application program, such as a browser, a word processor, or any other application or display program. The preferred embodiment achieves this by interfacing with the application and standing between the application and the document to control access to the document. Accordingly, a separate proprietary rendering engine for each document format is not required. Further, any data format supported by the application will be supposed by the invention without modification. It can be seen that the preferred embodiment permits DRM systems to be adapted to standards, such as TCP/IP and the use of browsers to render HTML. Further, the preferred embodiment facilitates various functionality that permits DRM to be applied to systems in a manner that is transparent to the user. Several examples of methods of operation of document distribution system **200** are described below.

[0064] In the first example, client computer **230** initially does not have all required components of security module **237** installed therein. As illustrated in **FIG. 5**, client computer **230**, makes a request of distributor server **220** for one or more documents **222** in step **502**. Distributor server **220** analyzes the request and based on a lack of signature information within the request (indicating that components of security module **237** are not loaded in client computer **230**), sends a response to client computer **230** to load the requisite components of security module **237** in step **504**. As noted above, security module **237** functions to enforce usage rights in client computer **230**. The response sent in step **504** is specific to the type of client requesting the content. If the client software on client computer **230** is a Web browser, for example, distributor server **220** will send a response that is a Web page including an executable software component. For example, the software component can be in a standard form such as Java Script or Active Server Pages. In addition the response, a Web page in the preferred embodiment, can include a copy of the unsigned request for content sent in step **502**.

[0065] Client computer **230** receives the Web page and executes the software component that includes information about where to get the components of security module **237**, in step **506**, to request a copy of the components. Client computer **230** receives and installs the components of security module **237** in step **508**. Security module **237** is configured to automatically begin to run in browser **232**, using the mechanism described above for example. In step **510**, security module **237** then reads the copy of the original request for content **222** contained in the Web page which invoked security component **237** and resubmit the request to distributor server **220** with a digital security signature. In step **512**, distributor server **220** receives the signed request and validates the signature on the request. Because this request is properly signed by security module **237**, distributor server **220** delivers the document **222** to security module **237** installed on client computer **230** for rendering by browser **232** in accordance with usage rights and conditions associated with the content of document **222**. The method illustrated in **FIG. 5** provides for auto-engaging security control to seamlessly and transparently provide the client with the software that the user needs to render content **222** in a secure manner. Security module **237** can include a software agent that is operative to analyze any rendering engine or other components of client computer **230**, in step **508**, to validate that the client environment is secure. Security module **237** can resubmit the request, in step **510**, after such validation.

[0066] **FIG. 6** illustrates another example of a method of operation of the preferred embodiment. In step **602**, security module **237** is directed to retrieve a document **222** from distributor server **220** server. In this example, document **222** is "clear content," i.e., is not encrypted or otherwise obscured or limited and does not have any use restrictions. Document **222** is returned by server **220** to security module **237** in step **604**. Because document **222** is not signed, or encrypted, or otherwise marked as content that needs to be handled by security module **237**, security module **237** recognizes that it is no longer required. In step **606**, security module **237** notifies browser **232** that browser **232** should request document **222** directly by sending the original request for content to server **220**. Security module **237** then removes itself as a running component, i.e., inactivates, in step **608** to preserve resources of client computer **230**. In step **610**, browser **232** then resubmits the request for document **222** that was originally sent by the security module **237**. Distributor server **220** then delivers documents **222** directly to browser **232** in step **612**.

[0067] In order to maintain security and enforce usage rights, all requests for content are initially made through security module **237**. However, when the request returns content that does not require security, security module **237** becomes a potential liability because it utilizes computer resources. In this example, if security component **237** is not needed, it is removed from a running state.

[0068] System **200** can use PKI encryption technology or any other encryption, ciphering, or watermarking technology. Of course, each technology requires that a client making a request for content be identified as an authorized user. A digital certificate or signature can be used for identification purposes. In other words, an attachment to an electronic message used for identification purposes is sent with a message. The most common use of a digital certificate or signature is to verify that a user sending a message is who he or she claims to be, and to provide the receiver with the

means to encode a reply, e.g., an encryption key. An individual wishing to send an encrypted message can apply for a digital certificate from a Certificate Authority (CA). The CA issues an encrypted digital certificate containing the applicant's public key and a variety of other identification information. The CA makes its own public key readily available, through the Internet for example. The recipient of an encrypted message uses the CA's public key to decode the digital certificate attached to the message, verifies it as issued by the CA and then obtains the sender's public key and identification information held within the certificate. With this information, the recipient can send an encrypted reply. The most widely used standard for digital certificates is X.509.

[0069] This identification process can be cumbersome when repeated at each server from which content is requested. **FIG. 7** illustrates another example of a method of operation of the preferred embodiment in which the identification procedure at plural servers is expedited.

[0070] As illustrated in **FIG. 7**, client computer **230** requests document **222** from distributor server **220** in step **702**. Assuming that PKI encryption schemes are used, the request can be in the form of "PrivateClient[request]" in which the request is encrypted with a private key of client computer **230**. Distributor server **220** looks at the signature of the request and recognizes that the request is not from an authenticated client and generates a unique "challenge" token which is returned to client computer **230** in step **704**. In the preferred embodiment, the challenge token can be in the form of "PrivateServer[PrivateClient[request]]" in which the original encrypted request is encrypted again using the private key of distributor server **220**.

[0071] Client computer **230** answers the challenge taken by transforming the challenge in a unique way, i.e., signing the challenge token, in step **706**. For example the transformation can be in the form of encryption of the challenge token with the public key of distributor server **220**. In such a case, the transformation will be in the form of [PublicClient[PrivateServer[PrivateClient[request]]]." Client computer **230** then resubmits the request to the distributor server **220** with the transformed challenge token in step **708**. Distributor server **220** establishes authentication with the client by recognizing the transformation. i.e. recognizing the challenge token as its own, and returns the requested document **222** in step **710**.

[0072] In many cases, distributor server **220** is part of a server farm or other set of related computers as noted above. Therefore, there is no guarantee that the server that gets the next request for this session will in fact be the same server that generated the "challenge" token. For example, the next session request may be received by distributor server **220'**. When client computer **230** sends another request reusing the same challenge token and the request is received by distributor server **220'**, in step **712**, distributor server **220'** looks for the signature of the challenge token, and finds that the signature belongs to distributor server **220**, in step **714**. Since distributor server **220** has a trusted relationship with distributor server **220'**, distributor server **220'** will honor the challenge token of distributor server **220**. In particular, distributor server **220'** evaluates the transformation of the challenge token performed by the client and authenticates the client by identifying server **220** as the creator of the challenge token. In step **716**, content **222'** is delivered from distributor server **220'** to client computer **230**.

[0073] In this method of operation a token supported by other related servers is honored by a server receiving a request. In order to simplify the process, keys are not exchanged again. The approval of a previous key exchange with a related server is used for any other server in a group of related servers to speed up the process of authentication.

[0074] As noted above, the use of a standard rendering engine presents significant complications for DRM systems. For example, when using a browser as the rendering engine, the standard user interface includes copy and print commands and other commands not necessarily compatible with DRM systems. It is known to disable such commands in which case most GUIs, such as the Windows GUI, will shadow the menu selections corresponding to disabled commands. However, this is often confusing and not ascetically pleasing. Further, it may be desirable to provide content specific menu selections, such as choices of usage rights and conditions for exercise thereof, such as fees to be paid. Further, a content vendor may want to present a proprietary branded user interface or a user interface that is consistent with other vendors. Also, it may be desirable to highlight menu selections, such as a print button, under certain circumstances.

[0075] **FIG. 8** illustrates a method of operation of the preferred embodiment which permits content specific toolbars to be displayed as the user interface of browser **232**. Documents **222** are stored on distributor server **220**, as described above, in a form compatible with the rendering application, a Web page in the preferred embodiment. Documents **222** are illustrated in detail in **FIG. 9** and include reference A to software component **220*a*** and reference B to description of a browser toolbar and UI **220*b***. Software component **220*a*** can be in the form of a Java applet, an Active X control, or the like. As the content is rendered, the reference to the software component is identified by the browser i.e. ActiveX Control, Java applet.

[0076] Referring to **FIG. 8**, browser **232** requests document **222** in step **802**. Browser **232** attempts to render document **222** and follows reference A to thereby execute software component **220*a*** in step **804**. Software component **220*a*** then looks at content **222** that invoked it and identifies reference B to description of toolbar and UI **220*b*** in step **806**. Software component **220*a*** then is operative to build a platform/browser specific toolbar and UI based on description **220*b*** in step **808**. In step **810**, software component **220*a*** removes or hides the standard browser UI and tool bar and replaces them with those built in step **808**. This method of operation permits the Web site (distributor for server **220** in this case) to dictate the navigation's motif, look, and appearance and thus tailor the user's browser to the site, customizing buttons, colors, patterns, animations, menus, and tool bars.

[0077] **FIG. 10** illustrates another manner of operation of the preferred embodiment in which a client-specific, or even instance specific, watermark can be applied to content for security and tracking purposes. The concept of digital "watermarking" is well known generally and allows content owners to imbed information within graphics and audio files that can be used to identify the owner's rights to these works. The term "digital watermark" is derived from the traditional

watermarks that exist in high-quality letterhead and certain currency. Traditional watermarks typically are not apparent to the reader, but, when held to the light, reveal the name or logo of the paper's manufacturer or the entity using the letterhead. Similarly, digital watermarks also serve the purposes of identifying quality and assuring authenticity. A graphic or audio file bearing a digital watermark can contain information relating to the content owner or other information. Digital watermarks may be only perceptible under certain conditions, such as when content is printed in an unauthorized manner. In graphic images, for example, digital watermarks alter the image to provide digital information supplied by the party who imbedded the watermark. The watermarks may be viewed with stand-alone or plug-in software and can reveal, for example, a unique identification code that can be traced to the copyright owner or more complete copyright ownership information.

[0078] As illustrated in **FIG. 10**, client computer **230** requests document **222** from distributor server **220** over communications channel **300** in step **1002**. Distributor server **220** delivers document **222** to security module **237** in step **1004**. Note that document **222**, as delivered to security module **237**, may or may not have a watermark embedded therein. In step **1006**, security module **237** delivers document **222** to an instance of the rendering engine, browser **232** in the preferred embodiment, for rendering in the manner described above. Security module **237** then uses instance-specific information relating to the instance of the rendering engine used to render content **222** to apply a client-specific watermark to the window that the instance of the rendering engine uses in step **1008**. The client-specific watermark can be applied using any techniques or algorithms for watermarking. Also, the client-specific watermark can be applied in addition to an existing watermark. The client-specific watermark data can be stored or generated in any manner.

[0079] In either case, because the watermark is applied on the client, it can be unique to that client, and thus be traceable. Distributor server **220** can deliver identical document **222** to all clients (thus minimizing server side performance impact). The client then applies a unique watermark using, for example, translucent windows to the image. If the consumer of the content then uses screen capture or another unauthorized mechanism to capture the content, the captured content is then watermarked with an ID that is unique to that user for tracking and enforcement purposes.

[0080] **FIG. 11** illustrates a manner of use of the preferred embodiment in which transaction payments are easily aggregated. In step **1102** client computer **230**, prior to installation of the requisite components of security module **237**, requests document **222** from distributor server **220**. In step **1104**, distributor server **220** then informs client computer **230** that document **222** is protected and client computer **230** needs to have security module **237** to render document **222**, and where security module **237** can be acquired. In this case, security module **237** can be acquired from a computer associated with transaction aggregator **160** (See **FIG. 1**). In step **1106**, client computer **230** requests the requisite components of security module security module **237** from transaction aggregator **160** by opening a session with the computer associated with transaction aggregator **160**. In step **1108**, transaction aggregator **160** requests and collects various user information including billing information and the like for example in a conventional manner.

[0081] Transaction aggregator **160** generates a unique security module **237** with a hidden unique public private pair or other indicia of the identity of client computer **230**, and returns security module **237** to client computer **230** in step **1110**. Security module **237** creates a protected instance of browser **232**, or other 3rd party rendering application, enforces access protection around it, and instructs it to retrieve and render protected content **222** from distributor server **220** in step **1112**. Distributor server **220** recognizes document **222** is being requested by a rendering engine that has been protected with security module **237**. And returns document **222**.

[0082] The protected rendering application, e.g. browser **232** with security module **237** attached, informs security module **237** that it is about to render document **222**. In step **1114**, security module **237** analyzes what the digital rights are associated with document **222**, and records an appropriate charge back to transaction aggregator **160**. Transaction aggregator **160** tracks many small transactions performed against many forms of content accessed by this instance of a public private key, then aggregates them into a single charge periodically to a financial institution or other party associated with the indicia in security module **237**.

[0083] The method of **FIG. 11** permits a user to log on to a new Web site to initiate a transaction. If the user's information is already on file at a trusted site (such as transaction aggregator **160**) the new Web site verifies the user through the trusted site. The trusted site reconciles all of the transactions and sends the results to the appropriate entity periodically, monthly for example. Accordingly, the burden of handling the transactions (often of small denominations) is shifted from the distributor or credit agency to the aggregator, which reduces the overall cost of transactions.

[0084] The new Web site does not have to obtain the detailed information of the user, which reduces concerns over privacy issues. In addition, the log-in process for the new Web site is simplified. Because the Website uses only an anonymous ID sent to the trusted site, and only the trusted site has the user's personal and credit information, the user's information is protected from the new Website that the user is transacting with.

[0085] Another method of operation of the preferred embodiment utilizes directory obfuscation for security without the need for a server side executable component. As illustrated in **FIG. 12**, the content owner, or other party having an interest in documents **222** creates a subdirectory on distributor server **220** with a random name, or other difficult name to discover, to serve as a secure location of documents **222** in step **1202**. The interested party then creates a Webpage which has a reference to security module **237** and an encrypted form of the new secure location of the documents **222** in step **1204**. The protected documents **222** are then replaced with the Web page, in step **1206**, and protected documents **222** are moved into the secure directory.

[0086] In step **1208**, client computer **230** issues a request to retrieve a document **222** from the original directory in the manner described above. The security Webpage which has the secret location of the content encrypted in it is returned

instead of the requested document in step **1210**. Security module **237** decrypts the location of the content referenced by the Web page and requests document **222** from the secure location in step **1212**. The content is delivered to security module **237** which creates an instance of a protected rendering engine, e.g. browser **232** in the preferred embodiment, and renders document **222** in step **1214**.

[**0087**] The method of operation described above does not require a server side executable and thus is an inexpensive way to provide adequate, while not necessarily maximum, security. In this example, the content is stored in a location having and address determined by a random number (or a pseudo-random number), for security purposes. Then, when the user initiates a transaction, the user is presented with an HTML page having the location in an encrypted form. The security component decrypts the location and the client never discovers the location.

[**0088**] **FIG. 13** illustrates another method of operation of the preferred embodiment which utilizes relative addressing for security. In step **1302** web browser **232** of client computer **230** requests content from distributor server **220**. Distributor server **220** recognizes that the content request is not coming from an appropriate security module **237** (by the lack of a proper signature for example), so it instructs client computer **230** to load the requisite components of security module **237** in step **1304**. In step **1306**, security module **237** spawns a child HTML rendering engine, i.e. instance of browser **232**, inside the existing instance of browser **232** so that it can have full control of the child HTML rendering engine. Security module **237** then instructs the child instance to retrieve content through an asynchronous protocol installed in security module **237** instead of through ordinary HTTP protocol and addressing in step **1308**. For example, the asynchronous protocol can be HTML with Active X controls embedded therein to establish a secure authenticated communication channel. The asynchronous protocol can direct browser **232** to retrieve content through another Web site that includes filtering technology to prevent access from unwanted or unauthorized users. Document **222** is rendered in step **1310**.

[**0089**] For example, the asynchronous protocol can send the address of the user to a third party for verification that the user is wanted and authorized. The asynchronous protocol can cause the child instance of browser **232** can request the top level HTML page via a designated secure Web site. After the top level page loads, the child instance can use an address prefix to retrieve all of the component parts of the page. Unprotected content can be retrieved via standard HTTP.

[**0090**] Generally, security is handed to HTML for rendering. However, in this example, content is retrieved using a proprietary asynchronous protocol. Therefore, a single instance of an HTML rendering engine can be used to "pull" compound pieces of content. As an example, standard HTML rendering can be used to access a start Web page containing an Active X control in it. The control spawns a child rendering engine which retrieves content through a specified server, which in turn accesses the server side, which includes the filtering technology or the like. Note that a Web page (a compound document) has references to other files and images. Conventionally, only the top level (HTML page) is protected, and the references are not protected. However, in this example, since requests are handled through a secured server, both the top level and the references are secure.

[**0091**] **FIG. 14** illustrates another method of operation of the preferred embodiment. This method provides security by prohibiting the loading of code, such as plug-ins and Dynamic Link Libraries (DLLs), into the rendering engine unless the code is certified as not compromising security. This method recognizes that certification can be a dynamic process that should permit users to use certified software immediately upon certification.

[**0092**] In step **1402** security module **237** of client computer **230** loads an instance of a rendering application, browser **232** in the preferred embodiment. Browser **232** requests to load a third party add in program, such as DLL, in step **1404**. Security module **237** intercepts the request and querys local database **225** including a list of trusted certified third party add in programs in step **1406**. If security module **237** does not find the third party program that is attempting to load, security module **237** contacts a trusted server to update its database of trusted third party programs that are certified in step **1408**. If the third party program is found in the updated list, security module **237** permits the loading of the third party program into the rendering engine in step **1410**. If the determination in step **1406** is that the program is certified by being listed in database **225**, the method goes directly to step **1410**. If the determination in step **1408** is that the program is not in the updated database as being certified, loading is prohibited in step **1412**.

[**0093**] Whenever the rendering application wants to load any executable code, it should be approved, i.e., certified, to avoid compromising security. If at the time of shipment of the security component a third party product is not ready for certification, it cannot be included in the approved list in the security module. If the program is approved later, the signature of the program will be compared to a list updated by logging onto to a server having and updated certification database, data base **225** for example.

[**0094**] **FIG. 15** illustrates a method of operation of the preferred embodiment which is well suited for the transfer of content in the form of video or other large files. It is known to encrypt only portions of data to reduce overhead and data transfer speed while still providing a level of security. However, in the method illustrated in **FIG. 15**, the percentage of encryption of a data stream is adaptive based on network latency, connection speed and other factors. In step **1502**, document **222** is requested by client computer **230**. Distributor server **220** determines what percentage of encryption to use by examining a database, usage rights, or other indication of encryption associated with document **222** in step **1504**. Such information can be stored in digital rights managers module **2**. For example, the indication of encryption can specify that encryption be greater than a specified percentage or in a range of specified percentages.

[**0095**] In step **1506**, distributor server **220** monitors various conditions related to data transfer, such as the files size of document **222**, network latency, communication speed, and the like in a known manner. In step **1508**, distributor server **220** encrypts portions of document **222** based on the conditions monitored in step **1506** and the encryption amount determined in step **1504**. Steps **1506** and **1508** are conducted continuously or iteratively until all of document

222 has been transferred. In step **1510**, security module **237** decrypts the content and delivers it to the rendering application, i.e. browser **232** in the preferred embodiment.

[**0096**] A variable portion (percentage) of a data stream can be encrypted. Content data can be divided by time intervals or based on the byte size. For example, 10 bytes encrypted, and 90 bytes not-encrypted. The percentage of encryption can be adaptive. That is, depending on the data file size and other conditions, at different times, the percentage of encryption can vary between values specified to speed up the process.

[**0097**] It is known to embed signatures and other security information in the body of an HTTP document. However, such a practice requires special security tags and is difficult to manage since the security information must be parsed out of the document. However, the method of operation of the preferred embodiment illustrated in **FIG. 16** simplifies this operation by using only the header of an HTML document for conveying security information.

[**0098**] In step **1602**, browser **232** that is regulated by security module **237** requests document **222** from distributor server **220** and security module **237** opens up a standard HTTP or HTTPS connection to distributor server **220**. In step **1604**, distributor server **220**, functioning as a standard HTTP server, retrieves or builds document **222** for downloading. In particular, digital rights management module **224** or another security module component of distributor server **220** authenticates the requesting client by analyzing security information embedded in the headers of the HTTP request and builds a standard HTTP reply.

[**0099**] In step **1606**, distributor server **220** inserts security information into the headers of the HTTP reply. For example, the security information, such as a signature can be an attribute of the <Header>tag in an HTML document as set forth below:

[**0100**] <Header>signature=13490680486724869
MY BOOK <Header/>

[**0101**] In the example above, the title of the HTML page is "MY BOOK" which will be rendered in accordance with standard HTML rules. The signature is a number as an attribute of the header and will not be rendered but can be culled for security purposes. In step **1608**, the reply is sent to security module **237** of client computer **230**. In step **1610**, security module **237** analyzes the security information in the reply header and passes content of document **222** to browser **232** for rendering in accordance with the usage rights described by or associated with the security information.

[**0102**] Since all of the security information is contained in the header, the resulting DRM system is less intrusive and easier to manage. Also, a new security tag schema or other specification is not necessary. The security component need only know to look in the header to get security information.

[**0103**] Often content and usage rights are dynamic. For example, the content may change over time and the usage rights may change over time and may be dependent on where the request for content comes from. For example, a company may want to let an employee print or save a document if the document is requested from an on-site, or otherwise secure, computer. However, the same employee requesting the same document from home may only be permitted to view the document. **FIG. 17** illustrates a method of use of the preferred embodiment that provides both address and URL filtering to address these issues.

[**0104**] In step **1702**, client computer **230** having security module **237**, requests secure document **222**. In step **1704**, distributor server **220** gathers information from either static or dynamic sources of content **222** and builds the response in a known manner. After the response has been built, a server side component of security module **237** accesses a database **225** that maps regular expressions of URL's to usage rights in step **1706**. Server side component of security module **237** inserts the rights associated with the reply based on the URL of the request by selecting the rights corresponding to the URL in database **225** in step **1708**. The reply is then sent to client computer **230** for rendering of the requested content **222** in accordance with the inserted usage rights under control of client side component of security module **237**.

[**0105**] Since both URL addressing and directory addressing are used, dynamic content and content that is best identified by incoming request URL's can be handled appropriately. Directories are filtered in order to provide a high level of confidence that content stored on the distributor server **220** as a file cannot be delivered to an unauthorized user no matter what URL is used to reach the file stored on the server. By using both types of filters, the content owner has flexibility in determining what content should be protected and to what degree. Further, putting security content in the header of an HTML document permits dynamic content to be handled easily because the body of the content does not need to be modified for security and thus permits dynamic content to be used to build a document on the fly.

[**0106**] Another problem often encountered in rendering content, particularly when distributing content over the Internet or other networks, is that the user may not always have the proper rendering application. For example, when downloading a PDF file, content providers will often warn the user that Adobe Acrobat Reader™ is required and may even provide a link to download the software. If the user does not download the software, they cannot render the content. However, downloading the software requires significant action on the part of the viewer, such as clicking on the link, choosing a directory for download, executing the installation software, and the like. In many cases, the user will choose not to download the content to avoid the cumbersome process of installing the proper rendering application. In DRM systems, the need for multiple rendering applications raises security issues if the security component is not attached to a newly installed rendering application.

[**0107**] **FIG. 18** illustrates a manner of use for providing the proper rendering applications in a manner that is transparent to the user. In step **1802**, client computer **230** requests document **222** that is of a file format that cannot be rendered by browser **232**. In step **1804**, document **222** is packaged as a file of the same format but is "disguised" as an HTML file. For example, the Windows™ operating system identifies file types by file extension. In such a case, the file, for example a PDF file, can be named with an "HTM" extension to be identified as an HTML file by client computer **230**.

[**0108**] In step **1806**, the file is downloaded to client computer **230** and the user "opens" the file which client

computer **230** recognizes as an HTML file. Accordingly, browser **232** is launched as the default HTML viewer. In step **1808**, browser **232** begins to render the HTML and finds a reference to an embedded application, like an ActiveX control or Java applet. The browser embedded application causes **232** to check and finds that the referenced application is not installed on client computer **230** in step **1810**. The browser follows the reference in the file to download the application in step **1812** and the application is installed on client computer **230** and attached to security module **237** as described above. The application, now used as the rendering application is directed by security module **237** to retrieve the content from within the HTML file and render the content in step **1814**.

[**0109**] The drawback of distributing a new file type extension is that if the user receives one of your data files and there is no registered application to handle the request, then the user cannot continue working with the content or must manually install a new application. However, if the new file type is packaged inside an HTML file the Web browser then loads the HTML file and automatically finds the code (JavaScripts, etc.). If the code sees a registered application on the client platform it passes the contained data to that client application. If it does not find an application to handle the data type, it calls upon the browser to navigate to a site that downloads the appropriate rendering application.

[**0110**] Another security issue when distributing content over a network, such as the Internet, is the possibility that hackers will intercept messages and "crack" encryption routines to obtain access to protected content. However, circumventing encryption often requires a relatively great deal of time (several seconds for example) because of the need to execute complex software algorithms or generate random numbers. **FIG. 19** illustrates a method of operation of the preferred embodiment in which the risk of encryption circumvention is reduced by creating tokens that expire after short period so time.

[**0111**] In step **1902**, client computer requests secure content **222**. Assuming that a server side security component of security module **237** has not authenticated client computer **230**, distributor server **120** generates a challenge token, that is time stamped, in step **1904**. Client computer **230** receives the token and uses its non-unique public key private key pair to add a request and sign it in a known manner in step **1906** and returns the signed token to distributor server **220**. Upon receipt of the signed token, distributor server **220** verifies the signature of client computer **230** and checks to see when the token was generated by examining the time stamp in step **1908**. If the token was generated more than a predetermined time period, 0.5 seconds for example, before being received in a signed fashion, the token is no longer valid and access to content **222** will be denied, in step **1910**, even if the signature is otherwise correct.

[**0112**] The time stamp can indicate how long the signature is valid (usually a very short time that permits proper signature but does not permit encryption circumvention) or the time that the signature was created. If an unauthorized party intercepts the message, and tries to imitate the message at a later time, then the signature will have expired and will not be valid.

[**0113**] The invention can be implemented over any type of communications Network, such as the Internet, a local area network (LAN), a wide area network (WAN), direct computer connections, or the like, using any type of communication hardware and protocols. Any type of hardware or combination of hardware can be used for the various clients and servers. Accordingly, the terms "client" and "server" as used herein, can refer to any type of computing device or data terminal, such as a personal computer, a portable computer, a dumb terminal, a thin client, a hand held device, a wireless phone, or any combination of such devices. The various clients and servers can be a single computer at a single location or multiple computers at a single or multiple locations. For example a server may be comprised of a plurality of redundant computers disposed in co-location facilities at various locations to facilitate scalability. There can be any number of clients and any number of servers. The client can physically be located on the same hardware as the server.

[**0114**] Any appropriate server or client software can be used and any communication protocols can be used. Communication can be accomplished over electric cable, fiber optic cable, or any other cable, or in a wireless manner using radio frequency, infrared, or other technologies. The various information can be stored in any format and thus the term "database" as used herein refers to any collection of information such as a database file, a lookup table, or the like. The documents can be of any type and can contain any type of content, such as text, audio information, video information, or combinations of plural types of content. The portions of the invention described above that are described as software components could be implemented as hardware. Moreover, while certain functional blocks are described herein as separate and independent from each other, these functional blocks can be consolidated and performed on a single general-purpose computer, or further broken down into sub-functions as recognized in the art. The set of rights can be one or more rights or rules governing use of the document, can be in any appropriate form, and can be based on various parameters such as the document type, the user's identity, a payment by the user, and the like. The various software modules can be located on the client or the server. For example, the security module can include one or plural components on the server side and/or on the client side as appropriate to accomplish the various functions disclosed above.

[**0115**] While a preferred embodiment of the invention has been described in detail above, it should be recognized that other forms, alternatives, modifications, versions and variations of the invention are equally operative and would be apparent to those skilled in the art. The disclosure is not intended to limit the invention to any particular embodiment, and is intended to embrace all such forms, alternatives, modifications, versions and variations. Accordingly, the true scope of the invention is defined by the appended claims and legal equivalents.

What is claimed:

1. A method of processing transactions related to content distribution comprising:

a user device requesting content;

the user device receiving a direction to request a security component required to render the content;

the user device requesting the security component from a computer associated with a transaction aggregator;

collecting billing information of a user of the user device;

returning the requested security component to the user device;

creating a protected instance of a rendering engine on the user device with the security component;

rendering the content within the protected instance; and

reporting said rendering to the transaction aggregator.

2. The method of claim 1, wherein the security component is unique to each user device.

3. The method of claim 1, wherein the security component uniquely identifies the user device to a document distribution system.

4. The method of claim 1, wherein said step of receiving a direction to request a security component is generated in response to said requesting step.

5. The method of claim 4, wherein said requesting step comprises requesting the purchase of the one or more documents and the security component controls the rendering of the one or more documents.

6. The method of claim 1, further comprising grouping together one or more similar transactions that are reported in said reporting step.

7. The method of claim 6, further comprising processing the one or more similar transactions in a batch-like manner.

8. A system for processing transactions related to content use comprising:

means for requesting content with a user device;

means for the user device to receive a direction to request a security component required to render the content;

means for requesting the security component from a computer associated with a transaction aggregator;

means for collecting user billing information;

means for returning the requested security component to the device;

means for creating a protected instance of a rendering engine on the user device with the security component;

means for rendering the content in the protected instance; and

means for reporting the rendering to the transaction aggregator.

9. The system of claim 8, wherein the security component is unique to each user computer environment.

10. The system of claim 8, wherein the security component uniquely identifies the user device to a document distribution system.

11. The system of claim 1, wherein said means for receiving a direction to request a security component is generated in response to the request for the security component.

12. The system of claim 11, wherein said means for requesting comprises means for requesting the purchase of the one or more documents and wherein the security component controls the rendering of the one or more documents.

13. The system of claim 8, further comprising means for grouping together one or more similar transactions that are reported in said reporting step.

14. The system of claim 13, further comprising means for processing the one or more similar transactions in a batch-like manner.

15. The system of claim 8, wherein said means for requesting content, said means for receiving, said means for requesting the security component, said means for collecting, said means for returning, said means for creating, said means for rendering, and said means for reporting comprise devices of a computer system.

16. The system of claim 8, wherein said means for requesting content, said means for receiving, said means for requesting the security component, said means for collecting, said means for returning, said means for creating, said means for rendering, and said means for reporting comprise computer readable instructions recorded on a medium.

* * * * *