Description

# SYSTEM FOR PROTECTING OBJECTS DISTRIBUTED OVER A NETWORK

## TECHNICAL FIELD

This invention is related to protecting objects such as code, documents, and images that are distributed over a network, particularly those exchanges that take place according to the Hypertext Transfer Protocol.

## BACKGROUND OF THE INVENTION

The Internet is now commonly used in the course of business to search for information and exchange code, documents, images, etc. among collaborators, prospective business partners, and customers. The increase in business conducted on the Internet has been accompanied by an increasing concern about protecting information stored or communicated on the Internet from "hackers" who can gain unauthorized access to this information and either use it for their own financial benefit or compromise the information or the system on which it is stored. Given the enormous volume of business conducted on the Internet and the corresponding value of that business, it is imperative that the objects (including code, documents and images - anything represented in digital form) that are stored and exchanged and the intellectual property contained within those objects are secure - i.e., they cannot be accessed by individuals or companies who have no right to them, they cannot be printed unless there is permission to do so, they cannot be edited except where that right has been conferred by the owner.

Protection of objects and object exchanges may have many components. One of these, authentication, is the process of verifying the identity of a party requesting or sending information. This is generally accom-

plished through the use of passwords. A drawback to this approach is that passwords can be lost, revealed, or stolen.

A stricter authentication process uses digital certificates authorized by a certificate authority. A digital certificate contains the owner's name, serial number, expiration dates, and the digital signature (data appended to a message identifying and authenticating sender and message data using public key encryption (see below)) of the issuing authority. The certificate also contains the certificate owner's public key. In public key cryptography, which is widely used in authentication procedures, individuals have public keys and private keys which are created simultaneously by the certificate authority using an algorithm such as RSA. The public key is published in one or more directories containing the certificates; the private key remains secret. Messages are encrypted using the recipient's public key, which the sender captures in a directory, and decrypted using the recipient's private key. To authenticate a message, a sender can encrypt a message using the sender's private key; the recipient can verify the sender's identity by decrypting the signature with the sender's public key.

Authorization determines whether a user has any privileges (viewing, modifying, etc.) with regard to a resource. For instance, a system administrator can determine which users have access to a system and what privileges each user has within the system (i.e., access to certain files, amount of storage space, etc.). Authorization is usually performed after authentication. In other words, if a user requests access to an object, the system will first verify or authenticate the identity of the user and then determine whether that user has the right to access the object and how that user may use the object.

Encryption may also be used to protect objects. Encryption converts a message's plaintext into ciphertext. In order to render an encrypted object, the

recipient must also obtain the correct decryption key (see, for instance, the discussion of the public key infrastructure and public key cryptography above). Although it is sometimes possible to "break" the cipher used to encrypt an object, in general, the more complex the encryption, the harder it is to break the cipher without the decryption key. A "strong" cryptosystem has a large range of possible keys which makes it almost impossible to break the cipher by trying all possible keys. A strong cryptosystem is also immune from previously known methods of code breaking and will appear random to all standard statistical tests.

Other types of security to protect the entire computer system may also be employed at the computer locations. For instance, many businesses set up firewalls in an attempt to prevent unauthorized users from accessing the business' data or programs. However, firewalls can be compromised and do not guarantee that a computer system will be safe from attack. Another problem is that firewalls do not protect the system or the system's resources from being compromised by a hostile user located behind the firewall.

Transmission of messages can also be secured. Transport Layer Security (TLS) and Secure Sockets Layer (SSL) protocols are commonly used to provide encrypted communications between servers and clients. Both these protocols are incorporated into most Web browsers and servers.

The above-mentioned security devices may be used separately, or more commonly, in some combination. In addition to these general devices, there are other approaches to security in the prior art.

U.S. Patent No. 6,041,411 "Method for Defining and Verifying User Access Rights to Computer Information" discloses a method for authenticating and authorizing access rights to electronically transmitted information. A user requests information which the provider wraps with digital information, or instructions, which must success-

fully be answered before the rest of the information can be accessed. The "answer" to these instructions takes the form of a digital token which is associated with validly requested data and indicates whether access to the information is authorized. The information may be accessed upon "unwrapping" only if the token is present and indicates the user may access the information. This patent is primarily concerned with ensuring the authorized use of software programs purchased on-line and electronically transmitted to a user.

InterTrust Technologies Corporation has received several patents related to their digital rights management technology. InterTrust's Digibox container technology enables the encryption and storage of information, including content and rules regarding access to that content, in a Digibox container, essentially a software container. Once the information is stored in a Digibox container, that information may be viewed only by Intertrust software. Keys are passed with the encrypted data.

There is a need for an invention that will protect objects (basically, anything which may be represented in digital form), including code, documents, images, and software programs, that are available on the Internet without requiring authorized requestors to run special software on their computers in order to access protected information. (For instance, students are often on a limited budget and, even if they have their own computers, cannot reasonably be expected to buy extra software which would enable them to download information like course notes, schedules, etc. that schools are increasingly making available to authorized users over the Internet.) Additional desirable features for a digital rights management system include passing most of the protection "duties" to a third party in order to relieve the object server of the processing burden of providing security and providing one-time encryption keys that are securely passed between the requestor and the "security

server" rather than passing the encryption keys with the encrypted data. It is also desirable for a digital rights management system to offer protection to an object even after the object has been sent to the requestor.

SUMMARY OF THE INVENTION

This invention provides a method and system for protection of objects (anything represented in digital form, i.e., code, documents, images, software programs, etc.) distributed over a network. Protection denotes restricting certain operations (i.e., viewing, printing, editing, copying) on the objects by certain recipients.

An object server containing objects, both protected and unprotected, is equipped with software that designates whether an object should be protected and, if so, what the security policy (type and degree of protection the object should receive) is. The security policy may include restrictions on who may view the object, the lifetime of the object, the number of times the object may be viewed, as well as actions policies relating to actions such as whether the object may be printed, edited, etc. Object controls are mechanisms which implement the security policy.

When the object server receives a request for an object, the software checks whether the requested object is protected. If the object is unprotected, the server will send the object to the requester. If the object is protected, the software creates a new object which includes authentication and time of the original request as well as serialization, nonce, security policy, and description of the requested object; all of these are encrypted. The new object is sent back to the requesting browser in a reply, along with a redirect command that points the requesting browser to a "security server."

After the security server, which is equipped with software for providing protection services, receives and authenticates the redirected request, it obtains the requested object either from its own cache or from the

server containing the object via a secure transmission. The security server then encrypts the requested object (using strong and non-malleable encryption) and combines it with mobile code (software sent from remote systems, transferred across a network, and downloaded and executed on a local system without explicit installation or execution by the recipient), the security policy, and object controls. This resulting package is sent back to the requesting computer as a reply to the redirected request.

The requesting computer then tries to execute the mobile code in order to render the requested object. The mobile code will execute tests to ensure proper instantiation of the object controls; when these controls are properly instantiated, the requestor may request a decryption key which is sent via secure transmission to the requestor upon satisfactory authentication of the request. The decryption keys are one-time keys which may be used only for decrypting the specific object in question. If the mobile code executes successfully and a decryption key is obtained, the requested object is rendered subject to the constraints of the security policy and object controls.

The security server is used to execute most of the activities associated with protecting and delivering the requested object. Therefore, the object server is not spending processing resources on security issues and instead is dedicated to handling requests for information. In addition, all set-up time and maintenance for the security server is handled by that server's system administrators, resulting in further savings to the owners of the object servers.

This method and system differ from other object protection methods and systems in that common software does not need to be installed on all computers involved in the request and provision of a requested object. In addition, the keys used to encrypt/decrypt the object are one-time keys and are not passed with the encrypted object.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a block diagram of the components of an object protection system in accordance with the invention.

Fig. 2a is a flow chart showing how an object is protected in accordance with the invention.

Fig. 2b is a flow chart showing how an object is protected in accordance with the invention.

BEST MODE FOR CARRYING OUT THE INVENTION

With reference to Fig. 1, a requestor device 10 (in this embodiment, the device is a computer; however, the device includes anything that can act as a client in a client/server relationship), an object server 12, containing objects 16 and protection software 14 which designates whether objects are to be protected, and a security server 18 containing software 94 for providing protection services are all connected to a network, in this embodiment, the Internet 20. An object 16 includes anything which may be represented in digital form, such as code, a document, an image, a software program, etc. An adversary 22, a person or device such as a computer or recorder which may be used to gain unauthorized access to a protected object, may also be present. Although a single requestor device 10, object server 12, and security server 18 are discussed here, it is envisioned that this method and system will accommodate a plurality of requestor devices 10, object servers 12, and security servers 18.

In this embodiment, the object server 12 and the security server 18 are Hypertext Transfer Protocol (http) servers. The requestor device 10 should be running a software program acting as a World Wide Web browser 24. Requests for objects 16 from the requestor device 10 are relayed by the browser 24 to the object server 12 via http requests. Similarly, replies to requests conform to the http protocol.

As noted above, the object server 12 is running protection software 14, which in this embodiment is an extension of http server software. This protection software 14 is used by an authorized system administrator to designate which objects 16 are unprotected and which are to be protected. If an object 16 is designated as protected, the protection software 14 also allows the administrator to specify the type and degree of protection (i.e., the security policy) for the object 16. The security policy may include restrictions on who may view the object, the lifetime of the object (i.e., temporal restrictions), the number of times the object may be viewed (i.e., cardinal restrictions), as well as actions policies relating to whether the object may be printed, edited, etc. The actions that the requestor may perform on an object may vary depending on the identity of the requestor. Object controls are mechanisms which implement the security policy.

The security server 18 is also running software 94 which is an extension of http server software. This software 94 provides the protection services for objects.

In Fig. 2, a requestor requests an object (step 26). The object server storing the requested object receives the request (step 28). If the object server has an independent authentication policy, the object server will execute that policy and authenticate the request upon receipt. The protection software examines the http request to determine whether the request is for a protected object (step 30). If the requested object is not protected, the requested object is sent to the requestor (step 32).

However, if the object is protected (step 30), the protection software creates an enhanced request (step 34) that is included in a reply to the request and is subsequently redirected to the security server (step 36). The enhanced request is an object comprising encrypted data including authentication and time of the original request as well as serialization (ensuring only one ap-

proved version of an object is available), nonce, security policy, and a description of the requested object. (Information about authentication depends on whether the object server has an independent authentication policy. If there is an authentication policy, the enhanced request includes the result of the authentication. If there is no authentication policy, that information is also included in the enhanced request.)

Encryption provides a variety of services. It can protect the integrity of a file (i.e., prevent unauthorized alterations) as well as assisting with the authentication and authorization of a request. The use of encryption here can also protect the privacy of the requestor. Other uses for encryption include non-repudiation and detecting alterations. Protocols supporting both strong and non-malleable encryption are used. (Protocols determine the type of encryption used and whether any exchanges between the requestor and security server are necessary before encryption takes place (for example, a key many need to be exchanged so the recipient can decrypt an object encrypted at the server).)

The enhanced request is included in the reply to the requestor along with a command to redirect the request to the security server. This redirection should be transparent to the requestor.

The security server software decrypts the enhanced request (step 38). A shared key for encrypting/decrypting the enhanced request is present at the object server and the security server. The key is generated when the software is installed on the object server.

The security server software then checks whether the enhanced request meets the requirements for a well-formed request (step 40). If the requirements for a well-formed request are not met, the security server sends a message back to the object server indicating an invalid request (step 42). (The object server may then send a message to the requestor about the invalid re-

quest. The system administrator for the object server determines whether these messages will be sent.)

If the request is valid, the security server software next authenticates the request (step 44). The security server software will compare the time and authentication in the redirected request heading with those contained in the enhanced request. If the security server software cannot authenticate the request (for instance, the two request times differ such that a replay attack is indicated or the identity of the requestor in the redirected request differs from the identity of the requestor in the enhanced request), a message is sent back to the object server indicating unsatisfactory authentication (step 46). If the request is authenticated, the security server software decrypts the request and obtains the requested object either from the security server's cache or the object server (step 48). (The protection software will pass the object on to the security server upon request.) If the security server has to obtain the object from the object server, the object is passed via a secure transmission.

Once the security server has the requested object, the security server software encrypts it using protocols for strong encryption and non-malleable encryption and combines the object with mobile code (software sent from remote systems, transferred across a network, and downloaded and executed on a local system without explicit installation or execution by the recipient), a security policy with authentication contained in the enhanced request, and object controls (step 50). Encryption of the requested protected object serves to protect the object, its requestor, and the provider by ensuring integrity, privacy, authentication (where appropriate), and authorization as well as being a tool for non-repudiation (i.e., a party to a transaction cannot falsely deny involvement in that transaction) and detecting alterations. The resulting package is then sent to the requestor (step 52; see step B, Fig. 2b).

In Fig. 2b, the requestor receives the reply and attempts to execute the mobile code (step 54). Upon execution of the mobile code, the security policy and object controls for the requested object are instantiated on the requestor's computer (step 54). The mobile code executes tests to determine whether the object controls were correctly instantiated. If so, if the requestor needs a decryption key (step 56), the requestor may request it from the security server (step 58). The security server software authenticates the request (step 60). If it cannot authenticate the request, a message to that effect is sent to the object server (step 62). However, if the message is authenticated, the security server software sends the requested key back to the requestor (step 64) via a secure transmission, and the requested object is decrypted (step 66). The key used by the security server to encrypt/decrypt the object is a one-time key. The "seed" for randomly generating the one-time key is determined at the installation of security server software.

Once the mobile code is executed, the requestor may view the object subject to any constraints imposed on the object by the security policy or object controls (step 68).

04 MAR 2005



## Claims

1. In a communications network, a system for protecting objects, said system comprising:

a) an object server running a software program which designates:

i) what objects among a set of objects on the object server are to be protected; and

ii) a security policy for protected objects, said object server connected to a network;

b) a requestor device requesting a protected object from the object server, said device connected to the network; and

c) a security server running another software program providing protection services for objects designated by the software program as protected, said security server connected to the network, said software providing protection services including:

i) means for receiving a redirected, enhanced request for the requested protected object from the requestor device, said enhanced request corresponding to the requestor device's original request for the requested protected object and created by the object server;

ii) means for obtaining said requested protected object from a cache or from the object server on which the requested protected object is stored;

iii) means for encrypting said requested protected object;

iv) means for combining the requested protected object with mobile code, a security policy, and object controls; and

v) means for sending the resulting file to the requestor device, said requestor device having to execute the mobile code to render the requested object to

the requestor device, a user of the requestor device to use and view the object subject to the security policy and object controls that are put in place on the requesting device upon execution of the mobile code; and

vi) means for verifying proper instantiation of the object controls; and

vii) means for providing a decryption key to the requestor device upon verification of proper instantiation of the object controls and satisfactory authentication of a request for said key.

2. The system of claim 1 wherein the encrypted data of the enhanced request includes authentication, time of original request, serialization, nonce, security policy, and description of the requested protected object.

3. The system of claim 1 wherein the object server and the security server share an encryption key for encrypting and decrypting enhanced requests.

4. The system of claim 1 wherein the enhanced request is an object containing encrypted authentication of the original request.

5. The system of claim 1 wherein the enhanced request is an object containing encrypted time of the original request.

6. The system of claim 1 wherein the enhanced request is an object containing encrypted serialization of the original request.

7. The system of claim 1 wherein the enhanced request is an object containing encrypted nonce of the original request.

8. The system of claim 1 wherein the enhanced request is an object containing encrypted security policy of the original request.

9. The system of claim 1 further including means for strong encryption.

10. The system of claim 1 further including means for non-malleable encryption.