Office de la Propriété Intellectuelle du Canada
Un organisme d'Industrie Canada

Canadian Intellectual Property Office
An agency of Industry Canada

CA 2513035 A1 2006/12/23

(21) 2 513 035

(12) DEMANDE DE BREVET CANADIEN
CANADIAN PATENT APPLICATION

(13) A1

(22) Date de dépôt/Filing Date: 2005/07/22

(41) Mise à la disp. pub./Open to Public Insp.: 2006/12/23

(30) Priorité/Priority: 2005/06/23 (CA2,510,647)

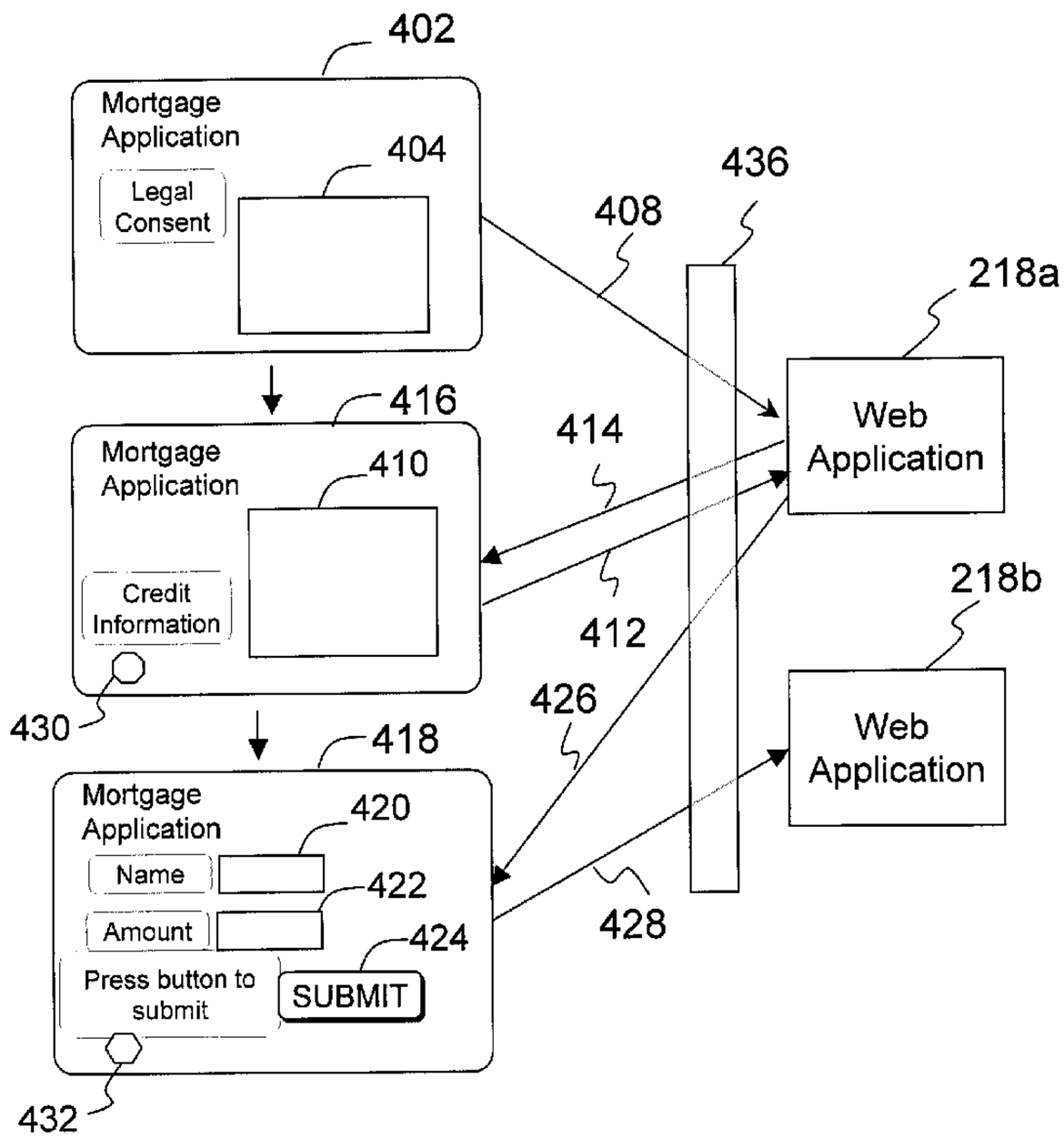
(51) Cl.Int./Int.Cl. *H04L 9/00* (2006.01)

(71) Demandeur/Applicant:
COGNOS INCORPORATED, CA

(72) Inventeurs/Inventors:
GRAVELINE, MARC, CA;
ROY, PATRICK, CA;
VINEY, ULF, CA

(74) Agent: GOWLING LAFLEUR HENDERSON LLP

(54) Titre : PROTECTION CRYPTOGRAPHIQUE OPAQUE DE DONNEES D'APPLICATION WEB
(54) Title: OPAQUE CRYPTOGRAPHIC WEB APPLICATION DATA PROTECTION



(57) Abrégé/Abstract:
A method and a system for external and distributed protection of Web application data against prying, tempering, and impersonation using cryptographic mechanisms. The protection is offered opaquely so as to not expose the cryptographic mechanism to the Web application. Protection against prying prevents users from looking at data the Web application considers private. When protected against prying, protect data may be sent to the client but the userwill not be able to understand it. Protection against tempering, guaranties the Web application that the data it is receiving originated from a trusted source, usually the Web application itself. A user session state stored client-side is a good candidate for tempering protection. Protection against impersonation ensures the Web application that the data it is receiving comes from a specific user.

ABSTRACT

A method and a system for external and distributed protection of Web application data against prying, tempering, and impersonation using cryptographic mechanisms. The protection is offered opaquely so as to not expose the cryptographic mechanism to the Web application. Protection against prying prevents users from looking at data the Web application considers private. When protected against prying, protect data may be sent to the client but the user will not be able to understand it. Protection against tempering, guaranties the Web application that the data it is receiving originated from a trusted source, usually the Web application itself. A user session state stored client-side is a good candidate for tempering protection. Protection against impersonation ensures the Web application that the data it is receiving comes from a specific user.

Opaque Cryptographic Web Application Data Protection

FIELD OF INVENTION

[0001] The present invention relates to Web application. More specifically, the present invention relates to Web application security.

BACKGROUND OF THE INVENTION

[0002] The Internet is by far the largest, most extensive publicly available network of interconnected computer networks that transmit data by packet switching using a standardized Internet Protocol (IP) and many other protocols. The Internet has become an extremely popular source of virtually all kinds of information. Increasingly sophisticated computers, software, and networking technology have made Internet access relatively straightforward for end users. Applications such as electronic mail, online chat and Web client allow the users to access and exchange information almost instantaneously.

[0003] The World Wide Web (WWW) is one of the most popular means used for retrieving information over the Internet. The WWW can cope with many types of data which may be stored on computers, and is used with an Internet connection and a Web client. The WWW is made up of millions of interconnected pages or documents which can be displayed on a computer or other interface. Each page may have connections to other pages which may be stored on any computer connected to the Internet. Uniform Resource Identifiers (URI) is an identifying system in WWW, and typically consists of three parts: the transfer format (also known as the protocol type), the host name of the machine which holds the file (may also be referred to as the Web server name) and the path name to the file. URIs are also referred as Universal Resource Locators (URLs). The transfer format for standard Web pages is Hypertext Transfer Protocol (HTTP). Hyper Text Markup Language (HTML) is a method of encoding the information so it can be displayed on a variety of devices.

[0004] Web applications are engines that create Web pages from application logic, stored data, and user input. Web applications often preserve users session state. Web applications may not require software to be installed in the client environment. Instead, Web applications make use of standard Web browser components to view server-side built pages. Web application can also deliver services through programmatic interface like Software Development Kits (SDKs).

[0005] HTTP is the underlying transactional protocol for transferring files (text, graphic images, sound, video, and other multimedia files) between Web clients and servers. HTTP defines how messages are formatted and transmitted, and what actions Web servers and Web client browsers should take in response to various commands. A Web browser as an HTTP client, typically initiates a request by establishing a TCP/IP connection to a particular port on a remote host. An HTTP server monitoring that port waits for the client to send a request string. Upon receiving the request string (and message, if any), the server may complete the protocol by sending back a response string, and a message of its own, in the form of the requested file, an error message, or any other information. Web pages regularly reference to pages on other servers, whose selection will elicit additional transfer requests. When the browser user enters file requests by either "opening" a Web file by typing in a Uniform Resource Locator (URL), or clicking on a hypertext link, the browser builds an HTTP request.

[0006] Web applications have server and client components. The Web application logic can be distributed between the server and the client. State information about a user session is often stored client-side for performance reasons. The state information is Web application data used to remember previous activities by a user. Because it is stored at the client, the Web application data can be tempered by a malicious user. An attack can occur when tempered Web application data is send as part of a request back to the server. Cryptographic algorithms are available to prevent a user from tempering or prying at restricted data like state information..

[0007] The information being passed from the Web application to the client may be sensitive in nature. In a commercial transaction, this sensitive information may be simply the price of an item to be purchased. In a Web application such as a business report engine, the information may include business intelligence. Although the sensitive information may be hidden from an average user, a malicious user may be able to find a way to access this information due to the nature of the HTML language.

[0008] Because many Web applications are developed without strong security initially, there is often a need to add cryptographic data protection to existing Web applications. A frequent requirement when implementing a data protection solution for an existing Web application is to minimize the amount of changes in the Web application.

[0009] Various methods have been proposed in the prior art to verify the integrity of electronic documents and to protect the sensitive information by using cryptographic functions.

[0010] US Application 20020023220 describes a system wherein a digital document processed with a one-way cryptographic hash function to yield a digital fingerprint value that is associated with the digital document. A document identification number is created, uniquely associated with the digital fingerprint DFP, and with document identification number and digital fingerprint are associated optional credential information. A registration certificate that represents an optional electronic signature associated with the document and that includes the document identification and digital fingerprint is promulgated and archived at a plurality of storage locations. The system can authenticate whether a putative document is the original document by generating a digital fingerprint value for the putative document and comparing it to digital fingerprint retrieved from various storage locations. Authentication can confirm that the electronic signature is unaltered.

[0011] US Application 20010002929 describes a keyed-hashing technique for authentication of messages communicated in a distributed system from an originator to a destination, whereby the data to be authenticated is concatenated with a private (secret) key and then processed to the cryptographic hash function. The data are transmitted together with the digest of the hash function from the originator to the destination. The data comprises temporal validity information representing the temporal validity of the data.

[0012] US Patent 6578144 describes a method and apparatus for constructing secure digital signature schemes using a "hash-and-sign" paradigm, while maintaining security.

[0013] US Patent 6144739 is directed to a method and apparatus for protecting software objects from external modification. A cryptographic seal protects the object at the object level and also supports secure inter-object communication. A software object is packaged in a crypto seal providing a cryptographic code hasher for performing a cryptographic form of hashing on the code of object, a crypto seal communications authenticator which authenticates communications received by object, a crypto seal encryptor which encrypts communications sent by the object , a challenge manager which causes the cryptographic code hasher to perform its hashing function on the code of object periodically and on demand when a challenge message is received, and a communications interface which controls inter-object communication.

[0014] There are cases where cryptographic data protection needs to be shared across multiple Web applications. In this scenario, a centralized cryptographic data

protection service to multiple Web applications is desirable. It simplifies the configuration, reduces the maintenance, and is easier to secure against attacks.

[0015] Therefore, there is a need for an external and centralized cryptographic data protection service for Web applications.

SUMMARY OF THE INVENTION

[0016] The present invention is directed to a method and a system for external and distributed protection of Web application data against prying, tempering, and impersonation using cryptographic mechanisms. The protection is offered opaquely so as to not expose the cryptographic mechanism to the Web application. Protection against prying prevents users from looking at data the Web application considers private. When protected against prying, protect data may be sent to the client but the user will not be able to understand it. Protection against tempering, guaranties the Web application that the data it is receiving originated from a trusted source, usually the Web application itself. A session state stored client-side is a good candidate for tempering protection. Protection against impersonation ensures the Web application that the data it is receiving comes from a specific user.

[0017] According to one aspect of the present invention there is provided a method for protecting Web application data between a server and a client comprising the steps of: building a response for the client; b) invoking a data protection service for the response, the response comprising a first data having a first state; c) modifying the response by replacing the first data with a protected data; d) sending the modified response to the client; e) receiving a request with the protected data from the client; f) passing the received protected data to the data protection service for verification and converting to the first data; g) restoring the request corresponding to the first state of the response; and h) sending the request to a Web application.

[0018] According to another aspect of the present invention there is provided a storage medium readable by a computer encoding a computer program for execution by the computer to carry out a method for protecting Web application data between a server and a client, the computer program comprising: code means for building a response for the client; code means for invoking a data protection service for the response, the response comprising a first data having a first state; code means for modifying the response by replacing the first data with a protected data; code means for sending the modified response to the client; code means for receiving a request with the protected data from the client; code means for passing the received protected data to the data

protection service for verification and converting to the first data; code means for restoring the request corresponding to the first state of the response; and code means for sending the request to a Web application.

[0019] According to another aspect of the present invention there is provided a computer system for protecting Web application data between a server and a client comprising: means for building a response for the client; means for invoking a data protection service for the response, the response comprising a first data having a first state; means for modifying the response by replacing the first data with a protected data; means for sending the modified response to the client; means for receiving a request with the protected data from the client; means for passing the received protected data to the data protection service for verification and converting to the first data; means for restoring the request corresponding to the first state of the response; and means for sending the request to a Web application.

BRIEF DESCRIPTION OF THE DRAWINGS

[0020] These and other features of the invention will become more apparent from the following description in which reference is made to the appended drawings wherein:

[0021] FIGURE 1 shows a generic computing system in which the present invention may be implemented;

[0022] FIGURE 2 shows a generic overview of a Web application environment;

[0023] FIGURE 3 shows examples of firewalls in relation to the OSI model;

[0024] FIGURE 4 (a) shows an example of client browser with a fill-out form showing different Web pages during exchanges of requests and responses;

[0025] FIGURE 4 (b) shows a query originated from the fill-out form in Figure 4 (a);

[0026] FIGURE 5 (a) and (b) show examples of relationships between application firewall and Web applications;

[0027] FIGURE 6 illustrates three data protection functions in an application firewall;

[0028] FIGURE 7 illustrates a flowchart of a data protection process according to one embodiment of the present invention; and

[0029] FIGURES 8 (a), (b) and (c) illustrates structural examples of protected data according to present invention.

DETAILED DESCRIPTION OF A PREFERRED EMBODIMENT

[0030] Reference will now be made in detail to some specific embodiments of the invention including the best modes contemplated by the inventors for carrying out the invention. Examples of these specific embodiments are illustrated in the accompanying drawings. While the invention is described in conjunction with these specific embodiments, it will be understood that it is not intended to limit the invention to the described embodiments. On the contrary, it is intended to cover alternatives, modifications, and equivalents as may be included within the spirit and scope of the invention as defined by the appended claims. In the following description, numerous specific details are set forth in order to provide a thorough understanding of the present invention. The present invention may be practiced without some or all of these specific details. In other instances, well known process operations have not been described in detail in order not to unnecessarily obscure the present invention.

[0031] In this specification and the appended claims, the singular forms "a," "an," and "the" include plural reference unless the context clearly dictates otherwise. Unless defined otherwise, all technical and scientific terms used herein have the same meaning as commonly understood to one of ordinary skill in the art to which this invention belongs.

[0032] Figure 1 and the following discussion are intended to provide a brief general description Figure 1 illustrates a block diagram of a suitable computing system in which a preferred embodiment of the present invention may be implemented.

[0033] Those skilled in the art will appreciate that the invention may be practiced with many computer system configurations, including personal computers, hand-held devices, multi-processor systems, microprocessor-based or programmable consumer electronics, network PCs, minicomputers, mainframe computers and the like. The invention may also be practiced in distributed computing systems where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing system, program modules may be located in both local and remote memory storage devices.

[0034] Although not required, the invention will be described in the general context of computer-executable instructions, such as program modules, being executed by a

personal computer. Generally, program modules include routines, programs, objects, components, data structures and the like that perform particular tasks or implement particular abstract data types.

[0035] With reference to Figure 1 an exemplary system **100** for implementing the invention may be, for example, one of the general purpose computers. The system **100** includes processor **102**, which in the exemplary embodiment are each connected to cache memory **104**, the cache **104** is connected in turn to a system bus **106** that couples various system components.

[0036] Also connected to system bus **106** are a system memory **108** and a host bridge **110**. Host bridge **110** connects I/O bus **112** to system bus **106**, relaying and/or transforming data transactions from one bus to the other. The system bus **106** and the I/O bus **112** may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. The system memory includes read-only memory (ROM) 114 and random access memory (RAM) **116**. A basic input/output system **118** (BIOS), containing the basic routines that help to transfer information between elements within the personal computer **100**, such as during start-up, is stored in ROM **114**.

[0037] In the exemplary embodiment, the system **100** may further include a graphics adapter **120** connected to I/O bus **112**, receiving user interface information for display device **122.** A user may enter commands and information into the system **100** through input devices **130** such as a conventional mouse, a key board **130**, or the like. Other input devices **134** may include a microphone, joystick, game pad, satellite dish, scanner or the like. The devices may be connected via an Industry Standard Architecture (ISA) bridge **126**, or a Universal Serial Bus (USB) bridge **132** to I/O bus **112**, respectively. PCI device such as a modem **138** may be connected to the I/O bus **112** via PCI bridge **136**.

[0038] The exemplary system **100** may further include a hard disk drive **124** for reading from and writing to a hard disk, connected to the I/O bus via a hard disk interface **140**, and an optical disk drive **142** for reading from or writing to a removable optical disk **144** such as a CD-ROM or other optical media. The hard disk drive **124**, magnetic disk drive 28, and optical disk drive **142** may be connected to the I/O bus **112** by a hard disk drive interface **140**, and an optical drive interface **146**, respectively. The drives and their associated computer-readable media provide non-volatile storage of computer readable instructions, data structures, program modules and other data for the system

100. Although the exemplary environment described herein employs a hard disk **124** and a removable optical disk **144**, it should be appreciated by those skilled in the art that other types of computer readable media which can store data that is accessible by a computer, such as magnetic cassettes, flash memory cards, digital video disks, Bernoulli cartridges, random access memories (RAMs), read-only memories (ROMs) and the like may also be used in the exemplary operating environment.

[0039] A number of program modules may be stored on the hard disk **124**, optical disk **144**, ROM **118** or RAM **116**, including an operating system **148**, one or more application programs **150**, other program modules **152** and program data **154**.

[0040] The exemplary system **100** may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer **156**. The remote computer **156** may be another personal computer, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to the exemplary system **100**. The logical connections depicted in Figure 1 include a network **158**, for example, a local area network (LAN) or a wide area network (WAN). Such networking environments are commonplace in offices, enterprise-wide computer networks, Intranets and the Internet.

[0041] When used in a networking environment, the exemplary system **100** is connected to the local network **158** through a network interface or adapter **160**. The exemplary system **100** may use the modem **138** or other means for establishing communications **162** over a wide area network such as the Internet. In a networked environment, program modules depicted relative to the exemplary system **100**, or portions thereof, may be stored in the remote memory storage device. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

[0042] The exemplary embodiment shown in Figure 1 is provided solely for the purposes of explaining the invention and those skilled in the art will recognize that numerous variations are possible, both in form and function. For instance, the exemplary system **100** may also include a magnetic disc drive, and numerous other optional components. All such variations are believed to be within the spirit and scope of the present invention. The exemplary system **100** and the exemplary figures below are provided solely as examples for the purposes of explanation and are not intended to imply architectural limitations. In fact, this method and system can be easily adapted

for use on any programmable computer system, or network of systems, on which software applications can be executed.

[0043] Figure 2 provides an overview of a network **210** with an application firewall **216** separating the Web application **218** with the client browser **240** on a computer **212** over a public network **214** such as Internet. The Web server **217** generally monitors the requests **220** from a Web browser **240** to a Web application **218**. The Web server **217** may include a firewall **216**. One example of the Web applications **218** is a business reporting engine (RE).

[0044] Referring to Figure 3, one type of firewalls is the network layer firewall **324** operating at the TCP/IP protocol stack as IP-packet filters, allowing packets to pass through the firewall only when they match the rules. The rules could be defined by the user; or by default. Network firewalls exist in most computer operating system and network appliances. Network firewalls deal with the numerous possible combinations that are to be allowed and forbidden at the transmission control protocol (TCP) and Internet protocol (IP) level.

[0045] Another type of firewall is an application layer firewall (326) operating at the application layer of a protocol stack. It may be a host using a proxy server or gateway, or a module embedded within an application. The purpose of an application firewall is to proxy traffic instead of routing it. As it works at the application layer, it may be configured to inspect the contents of the traffic, blocking what the firewall administrator views as inappropriate content, such as invalid request or attempts to exploit logical flaws in the application. The application firewall may be configured by a set of predetermined rules that are read at the time of startup.

[0046] A service to a Web application may be delivered through a simple interface hiding the complexity of the logic. This solution is called opaque. For example, an opaque interface to a data protection service offers services to a Web application without exposing the complexity of the cryptographic operations. To achieve higher opaqueness, the data protection service needs to protect the data so that it can reside in its original location and safely travel in requests (220) and responses (222). A data protection service can protect Web application data against prying, tempering, or impersonation.

[0047] The present invention is directed to a protection service in an application firewall (326). The application firewall has the ability to modify requests and responses transmitted through it. The application firewall is therefore able to protect data before

the data leaving or reaching Web application. The application firewall can also offer a programming interface to the Web application to offer data protection services. The protection service used to protect Web application server state will be described in details.

[0048] Web application server state is data used by the server, and is stored in the client environment for performance reasons. When the Web application server receives a request with server state data in it, it uses the data to reconstitute the user session context. If not secured, the server state data may be tempered by a malicious user client-side and then send to the server. The tempered data when used to build the user session context can then crash the server, result in a privilege escalation, or other forms of security exploit.

[0049] Referring to Figures 2 and 4, an example of a figurative mortgage application is shown at different stages 402, 416, 418 with different data protections. The first Web client browser page 402 being displayed is for a legal consent 404. The request 408 from the client browser page 402 to Web application 218a may be protected through integrity protection and origin authentication. The response 414 from the Web application 218a to the client browser page 416 constitutes a server state 430 for this client page. The server state 430 comprises, for example, integrity protected legal consent. Similarly, the Web application 218a and the client browser may exchange information related to credit information 410 through another request 412 and response 426. This information may be protected through confidentiality protection, and forming part of the server state information 432 on client browser page 418. The server state information 432 now has both the integrity protected legal consent and the confidentiality protected credit information. Web client browser page 418 has a fill-out form with two text input fields 420, 422 and two hidden fields with the server states illustrated as {integrity protected legal consent} and {confidentiality protected credit information} in Figure 4 (b). The data will be sent to the Web application 218 through a request 428 after the submit button 424 is pressed. Figure 4 (b) shows the corresponding implementation in HTML. It should be apparent to a person skilled in the art that more than one Web applications, for example Web applications 218a and 218b as illustrated in Figure 4 (a) may be used for responding the requests from the Web client browser. In addition, a firewall such as an application firewall 436 is usually used between the Web applications 218 and the Web client browser.

[0050] To secure its server state data, the Web application may have one or more of these requirements: integrity, authentication, and confidentiality. Integrity is required

most of time. It requires the data to have the exact same value as when inserted in the response to the client. Authentication is required when the Web application enforces that data is received from a specific user. The purpose is to prevent a malicious user to send data it captures from other users. Confidentiality is required if the data contains secret information that cannot be viewed by anyone outside the server.

[0051] Referring to Figure 5 (b), when building a response in accordance to one embodiment of the present invention, the Web application 218 sends its server state data 502 to an application firewall 326 with protection requirements. Figure 5 (b) is an example of a single application firewall 326 providing data protection services to multiple Web applications 218. To minimize the impact on the Web application the application firewall will take a parameter value and return a replacement protected value. As used herein, the term parameter may include but not limited to a GET request query parameter, a form POST body parameter, or an XML attribute or element.

[0052] Figure 6 illustrates the tasks which may be accomplished at the application firewall 326: integrity protection 602; origin authentication 604 and confidentiality protection 606. An example of a protection request may have the requirements to protect response data for integrity, origin authentication, and confidentiality. The application firewall 326 invokes the appropriate cryptographic operations 602, 604, 606, encodes the protected data so as to safely travel between the client and the server, and then wraps the data with the addition of a prefix to recognize its protected state. This wrap protected value is returned 504 to the Web application to replace the original value in the response. There are no other parameters created to store cryptographic information.

[0053] Integrity protection may be implemented in the application firewall using a cryptographic keyed hash message authentication code (HMAC). Standard for the cryptographic keyed HMAC may be found at: http://csrc.nist.gov/publications/fips/fips198/fips-198a.pdf, the entirety of the document is incorporated hereby by reference. The HMAC key is secret to the application firewall. The HMAC value and its length are included in the wrapped protected data.. The hashing algorithm is configurable in the application firewall. The default algorithm is SHA-1 from the Secure Hash Standard. Secure hash standard is described at http://www.itl.nist.gov/fipspubs/fip180-1.htm, the entirety of the document is incorporated hereby by reference.

[0054] Origin authentication protection may be accomplished by concatenating the user session token to the end of the data prior to it being protected for integrity. The

user session token is a unique value associated to the user. Origin authentication protection requires integrity protection.

[0055] Confidentiality protection is accomplished by using a encryption symmetric key algorithm on the data. The encryption key is secret to the application firewall. The encryption algorithm is configurable in the application firewall. One of the examples of the encryptions is RSA Security RC-4 (40-bit key). When both confidentiality and integrity protection are required, encryption is performed before integrity.

[0056] For safe travel in HTTP responses and requests, some of the encoded data need to be encoded. For example, the HMAC is binary data thus could not be sent as a replacement value to any parameter. The application firewall may thus use the Base64 encoding (http://www.faqs.org/rfcs/rfc3548.html) to render the encoded data Web safe. Example of a Base64 encoding maybe found at http://www.faqs.org/rfcs/rfc3548.html, the entirety of the document is incorporated hereby by reference.

[0057] When integrity protection is required the original data is also Base64 encoded to avoid accidental modifications to it. For example, spaces or carriage returns can be inserted by clients for readability, using Base64 encoding creates an un-interrupted string that is not subject to formatting modifications.

[0058] The protection information and the encoded data are wrapped to facilitate their transport. A unique prefix is used at the beginning of the value to identify its protected state. The prefix is terminated by a letter indicating which protection and encoding was applied to the data. For example, "CAFS" may be used as a prefix for an authentication protection that has been Base64 encoded. The prefix is followed by the length of the protected data. When multiple protections are used multiple wrapping occurs; a wrap value becomes the un-protected data for the other protection.

[0059] Figure 7 shows an example of data protection in an application firewall 326. After the data which is to be protected is received 702, it is first decided whether confidentiality needs to be protected 704, if yes, the data is encrypted 706. It the data is to be protected for its integrity 708, authenticated for its origin 710, respectively, before the data is cryptographically encoded 714. The HMAC'ed data is encoded, for example, using Base64 and prefix may be added 716.

[0060] Figure 8 shows examples of protected data structure: (a) HMAC encoded data; (b) encoded encrypted data; and (c) HMAC encoded and encrypted data.

[0061] When a request is received, the application firewall examines it for protected data. Based on parameter rules part in the application firewall, the application firewall verifies the protected data and restores it to its unprotected state if required. A parameter rule declares a given parameter, usually specify by name, to require a protected state. A parameter rule may specify that the restore to the original value will be skipped in the application firewall because it will be handled by a separate call from the Web application. This is convenient when a Web application wants to reuse the same protected data in multiple responses without having it unwrap or decrypt for every request. The verification is performed even if the restore is skipped.

[0062] The verification of protected data occurs when a parameter rules require has an integrity requirement. To verify a value, the application firewall retrieves the original data value from the wrap protected data. It then performs the same HMAC logic as done to protect the value. The resulting HMAC value must match the HMAC value that was stored as part of the wrapped data. The two HMACs match only if the received data is the exact same data as when protection was invoked. A malicious user cannot create fake HMAC protected data bundles because he is not aware of the application firewall secret used to key the HMAC algorithm. When authentication is required, the user session token is also added, thus the two HMACs match only if the same user session token was used when protection was invoked. In the case of a HMAC mismatch, a verification failure, the application firewall blocs the request.

[0063] When a parameter rule specifies that protect data should not be unwrap in the application firewall, the Web application is responsible to call the application firewall to unwrap it and obtain the original data before using it.

[0064] One of the advantages of the present invention is to limit changes to the Web application. By only adding simple calls to the application firewall at the exit and entry point of data in the Web application, the risk of negative impact on the Web application is minimized. Opaqueness has been achieved by having the Web application request one or more of three protection requirements on its data. The Web application is may not be aware that cryptographic operations are performed. The application firewall data protection service can easily be shared between multiple Web applications.

[0065] The invention can be implemented in digital electronic circuitry, or in computer hardware, firmware, software, or in combinations thereof. Apparatus of the invention can be implemented in a computer program product tangibly embodied in a machine-readable storage device for execution by a programmable processor; and method actions can be performed by a programmable processor executing a program

of instructions to perform functions of the invention by operating on input data and generating output. The invention can be implemented advantageously in one or more computer programs that are executable on a programmable system including at least one programmable processor coupled to receive data and instructions from, and to transmit data and instructions to, a data storage system, at least one input device, and at least one output device. Each computer program can be implemented in a high-level procedural or object oriented programming language, or in assembly or machine language if desired; and in any case, the language can be a compiled or interpreted language. Suitable processors include, by way of example, both general and special purpose microprocessors. Generally, a processor will receive instructions and data from a read-only memory and/or a random access memory. Generally, a computer will include one or more mass storage devices for storing data files. Storage devices suitable for tangibly embodying computer program instructions and data include all forms of non-volatile memory, including by way of example semiconductor memory devices, such as EPROM, EEPROM, and flash memory devices; magnetic disks such as internal hard disks and removable disks; magneto-optical disks; and CD-ROM disks. Any of the foregoing can be supplemented by, or incorporated in, ASICs (application-specific integrated circuits). Further, a computer data signal representing the software code which may be embedded in a carrier wave may be transmitted via a communication network. Such a computer readable memory and a computer data signal are also within the scope of the present invention, as well as the hardware, software and the combination thereof.

[0066] The present invention has been described with regard to one or more embodiments. However, it will be apparent to persons skilled in the art that a number of variations and modifications can be made without departing from the scope of the invention as defined in the claims.

WHAT IS CLAIMED IS:

1. A method for protecting Web application data between a server and a client comprising the steps of:

a) building a response for the client;

b) invoking a data protection service for the response, the response comprising a first data having a first state;

c) modifying the response by replacing the first data with a protected data;

d) sending the modified response to the client;

e) receiving a request with the protected data from the client;

f) passing the received protected data to the data protection service for verification;

g) restoring the request corresponding to the first state of the response at the data protection service; and

h) sending the request to a Web application.

2. The method as claimed in claim 1, wherein the first data is protected based on protection rules in the data protection service.

3. The method as claimed in claim 1, further comprising the step of providing a data characteristic to the data protection service by the Web application; whereby the data protection service decides the protection logic to be used.

4. The method as claimed in claim 1, wherein the data protection service is a stand-alone service external to the Web application.

5. The method as claimed in claim 1, wherein the data protection service is provided by an application firewall.

6. The method as claimed in claim 1, wherein the data protection service is shared between multiple Web applications.

7. The method as claimed in claim 1, wherein more than one data protection services are invoked, the data protection services being external to the Web application.

8. The method as claimed in claim 7 wherein more than one data protection services are shared between multiple Web applications.

9. The method as claimed in claim 1, wherein the modifying step further comprising the step of replacing the parameter values at their original location.

10. The method as claimed in claim 1, wherein the data protection service comprises an opaque interface to the Web application providing services such as protection against prying, tempering, or impersonation; thereby hiding cryptographic details to the Web application.

11. The method as claimed in claim 1, wherein the data protection service encodes the data for safe travel between the Web application client and server.

12. The method as claimed in claim 1, further comprising the step of having the data protection service dynamically maintaining a list of parameter names requiring protection in requests based the calls from the Web application.

13. A storage medium readable by a computer encoding a computer program for execution by the computer to carry out a method for protecting Web application data between a server and a client, the computer program comprising:

a) code means for building a response for the client;

b) code means for invoking a data protection service for the response, the response comprising a first data having a first state;

c) code means for modifying the response by replacing the first data with a protected data;

d) code means for sending the modified response to the client;

e) code means for receiving a request with the protected data from the client;

f) code means for passing the received protected data to the data protection service for verification and converting to the first data;

g) code means for restoring the request corresponding to the first state of the response; and

h) code means for sending the request to a Web application.

14. The storage medium according to claim 13, wherein the first data is protected based on protection rules in the data protection service.

15. The storage medium according to claim 13, further comprising code means for providing a data characteristic to the data protection service by the Web application; whereby the data protection service decides the protection logic to be used.

16. The storage medium according to claim 13, wherein the data protection service is a stand-alone service external to the Web application.

17. The storage medium according to claim 13, wherein the data protection service is provided by an application firewall.

18. The storage medium according to claim 13, wherein the data protection service is shared between multiple Web applications.

19. The storage medium according to claim 13, wherein more than one data protection services are invoked, the more than one data protection services being external to the Web application.

20. The storage medium according to claim 13, wherein more than one data protection services are shared between multiple Web applications.

21. The storage medium according to claim 13, wherein the data protection service encodes the data for safe travel between the Web application client and server.

22. The storage medium according to claim 13, further comprising code means for having the data protection service dynamically maintaining a list of parameter names requiring protection in requests based the calls from the Web application.

23. A computer system for protecting Web application data between a server and a client comprising:

a) means for building a response for the client;

b) means for invoking a data protection service for the response, the response comprising a first data having a first state;

c) means for modifying the response by replacing the first data with a protected data;

d) means for sending the modified response to the client;

e) means for receiving a request with the protected data from the client;

f) means for passing the received protected data to the data protection service for verification and converting to the first data;

g) means for restoring the request corresponding to the first state of the response; and

h) means for sending the request to a Web application.

24. The computer system according to claim 23, wherein the first data is protected based on protection rules in the data protection service.

25. The computer system according to claim 23, wherein the data protection service is a stand-alone service external to the Web application.

26. The computer system according to claim 23, wherein the data protection service is provided by an application firewall.

27. The computer system according to claim 23, wherein the data protection service is shared between multiple Web applications.

28. The computer system according to claim 23, wherein more than one data protection services are shared between multiple Web applications.

29. The computer system according to claim 23, wherein the data protection service encodes the data for safe travel between the Web application client and server.

30. The computer system according to claim 23, further comprising means for having the data protection service dynamically maintaining a list of parameter names requiring protection in requests based the calls from the Web application.

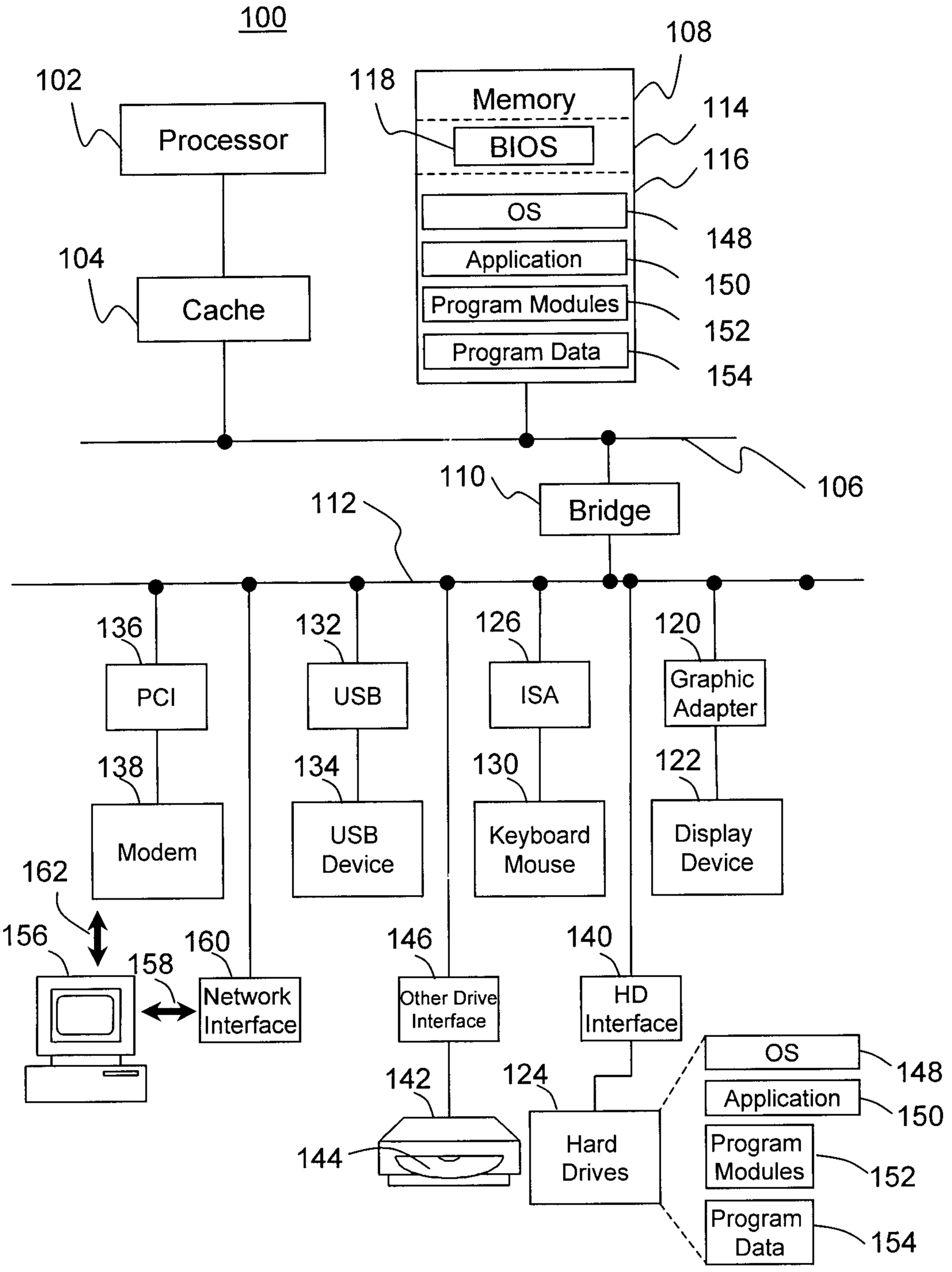

100
102
Processor
104
Cache
118
Memory
BIOS
OS
Application
Program Modules
Program Data
108
114
116
148
150
152
154
106
110
Bridge
112
136
PCI
138
Modem
132
USB
134
USB Device
126
ISA
130
Keyboard Mouse
120
Graphic Adapter
122
Display Device
162
156
158
160
Network Interface
146
Other Drive Interface
140
HD Interface
142
144
124
Hard Drives
OS
Application
Program Modules
Program Data
148
150
152
154

Figure 1

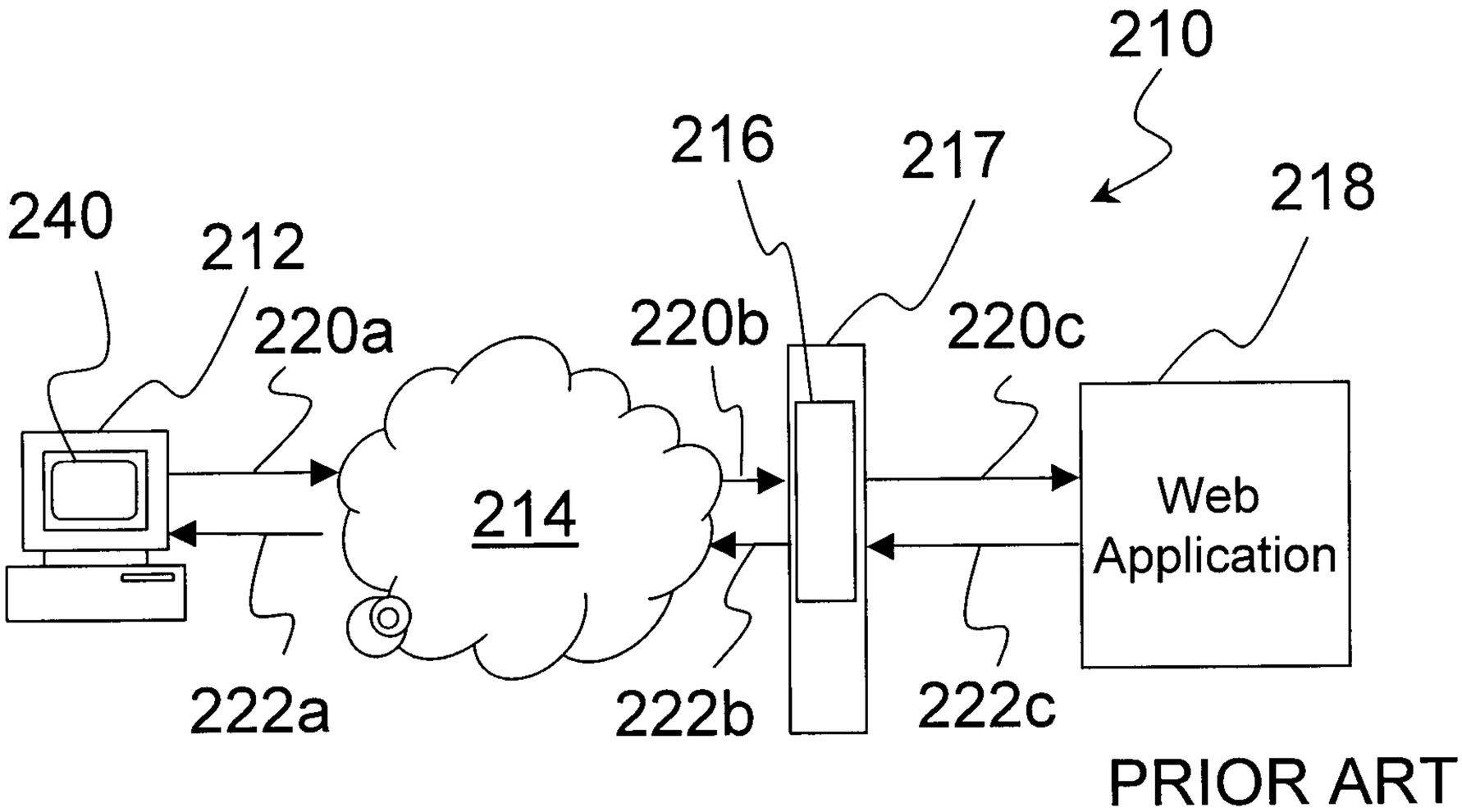


Figure 2

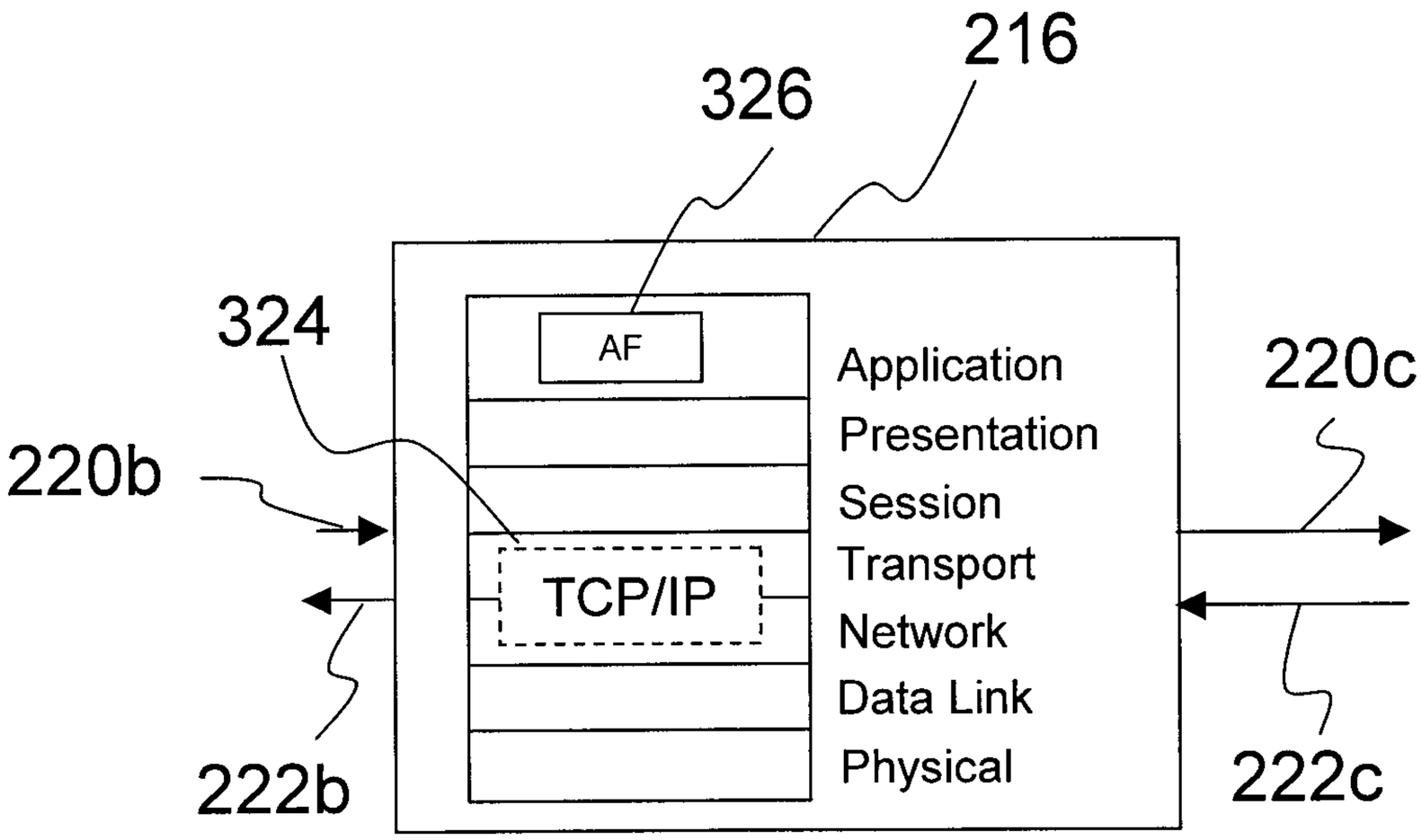


Figure 3

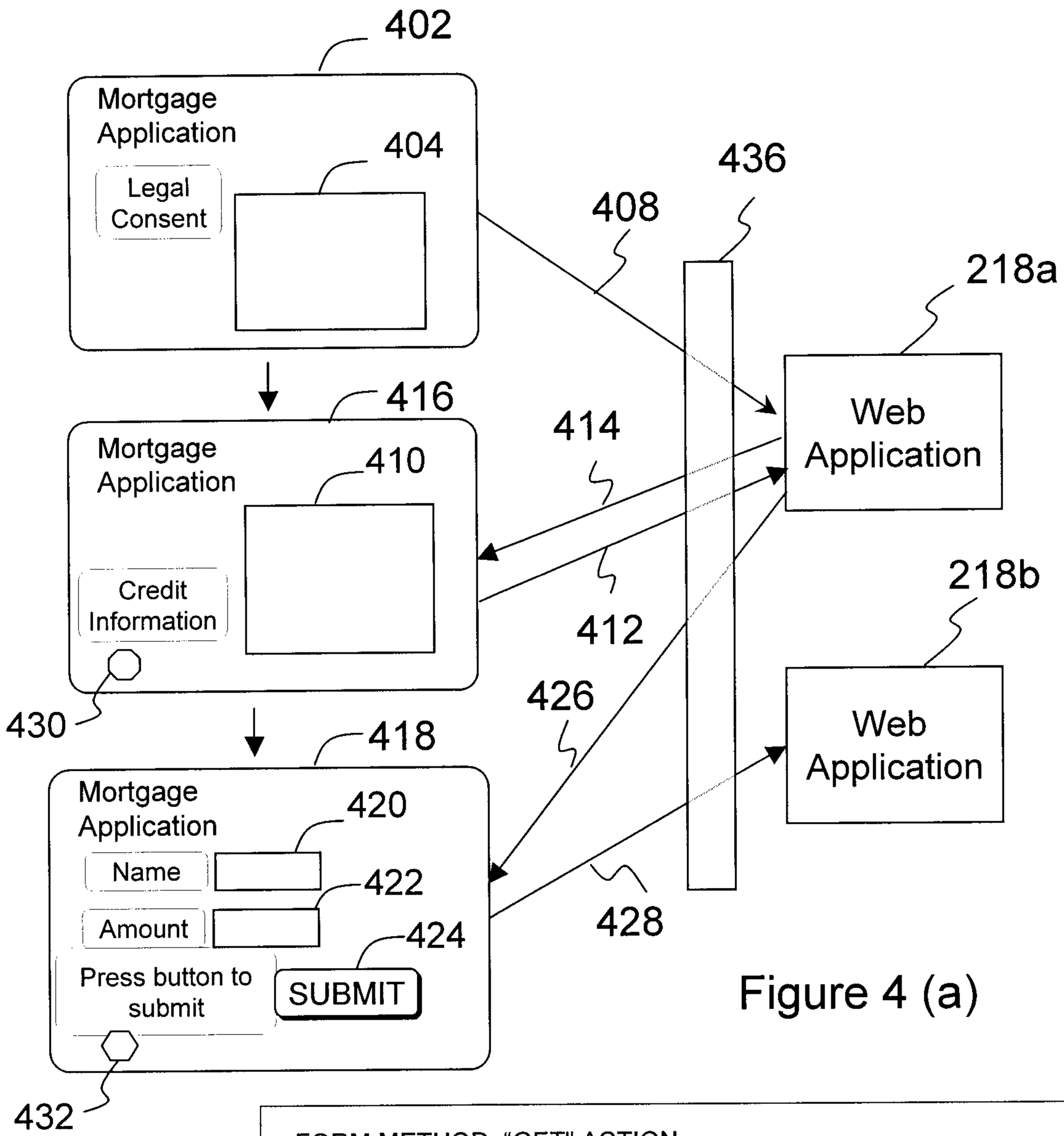


Figure 4 (a)

```
<FORM METHOD="GET" ACTION=
"http://www.example.com/cgi-bin/approve">
Name <INPUT NAME="Bob"> <P>
Amount <INPUT NAME="100,000"> <P>
<INPUT TYPE="hidden" NAME="legal consent" value={integrity protected
legal consent}>
<INPUT TYPE="hidden" NAME="credit information" value={confidentiality
protected credit information}>
Press button to submit: <INPUT TYPE="submit" VALUE="SUBMIT"> <P>
</FORM>
```

Figure 4 (b)

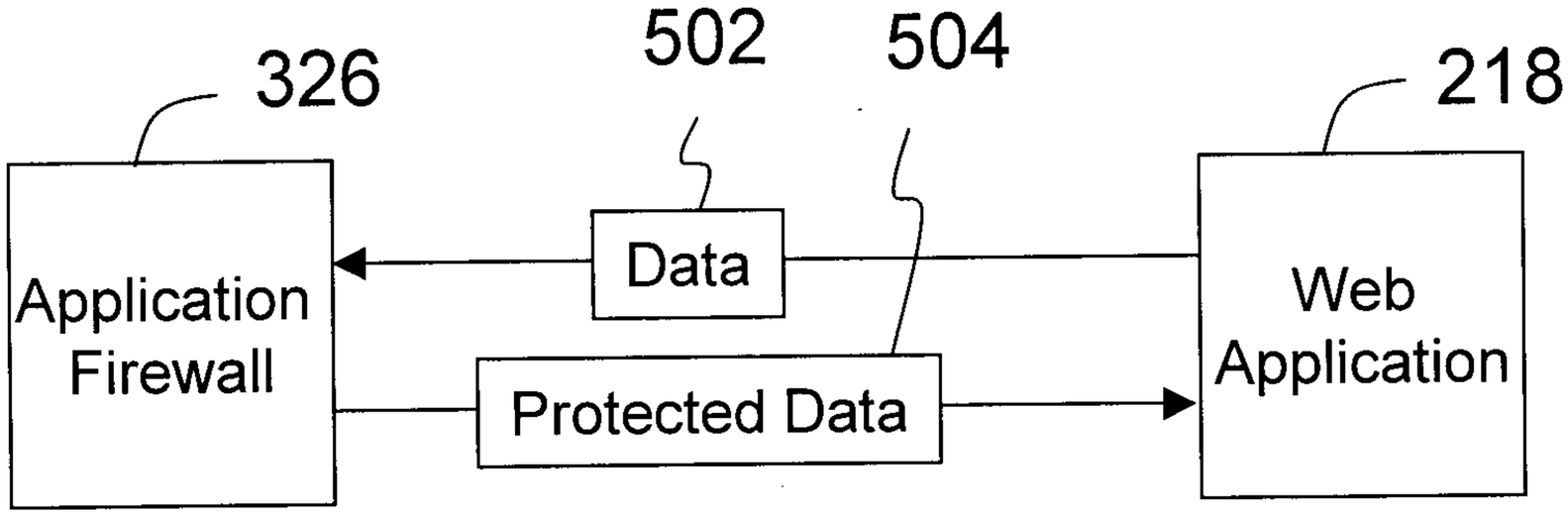


Figure 5 (a)

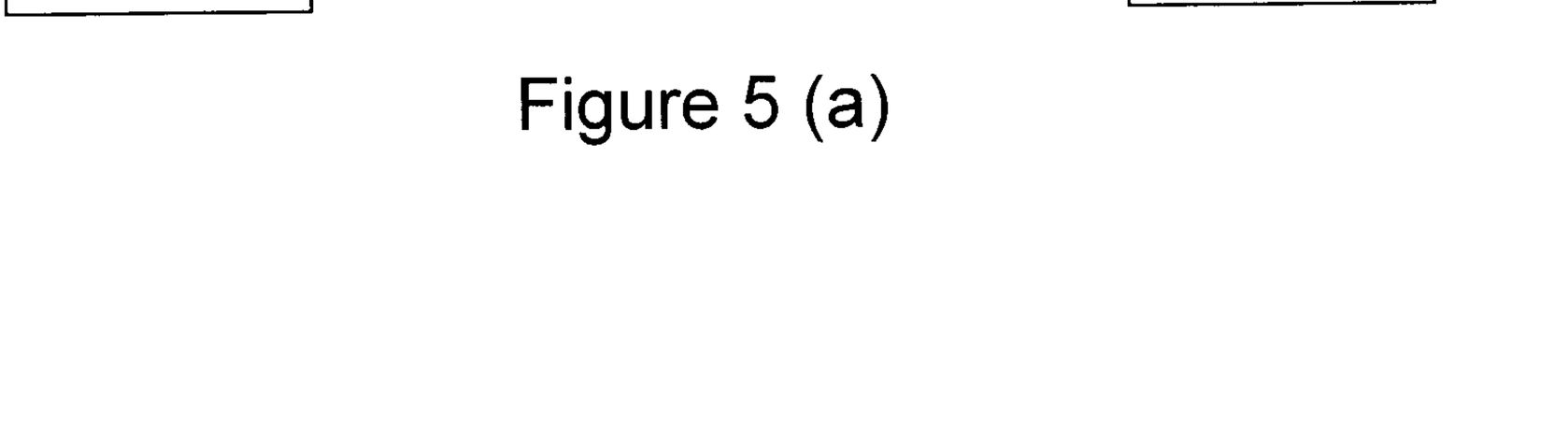


Figure 5 (b)

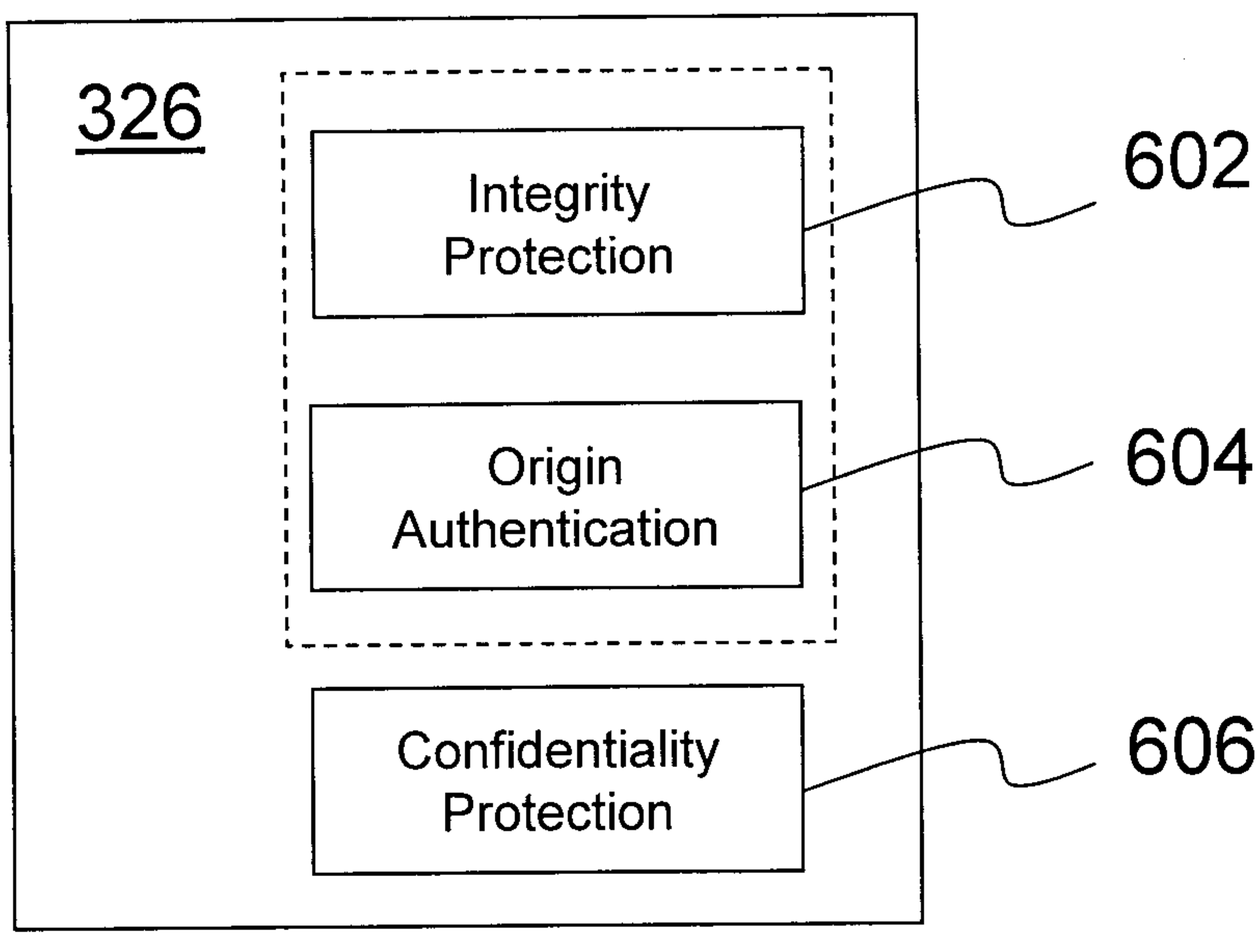


Figure 6

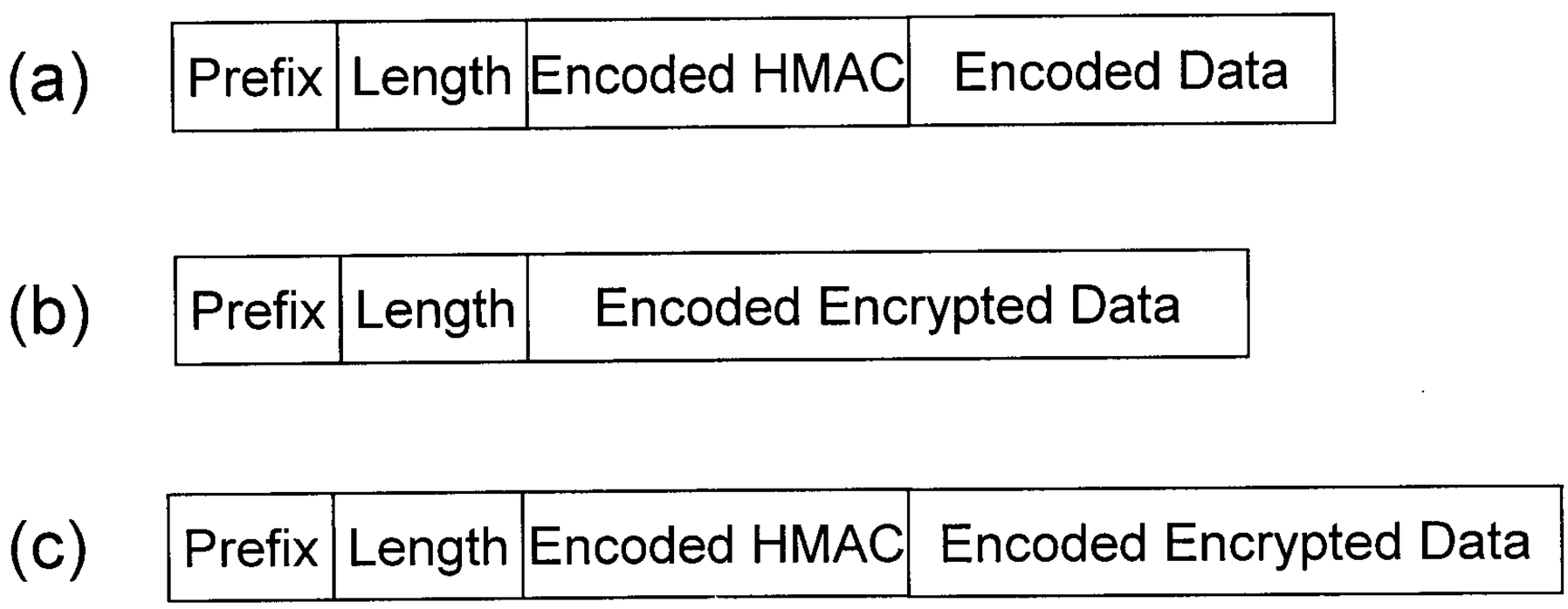


Figure 8

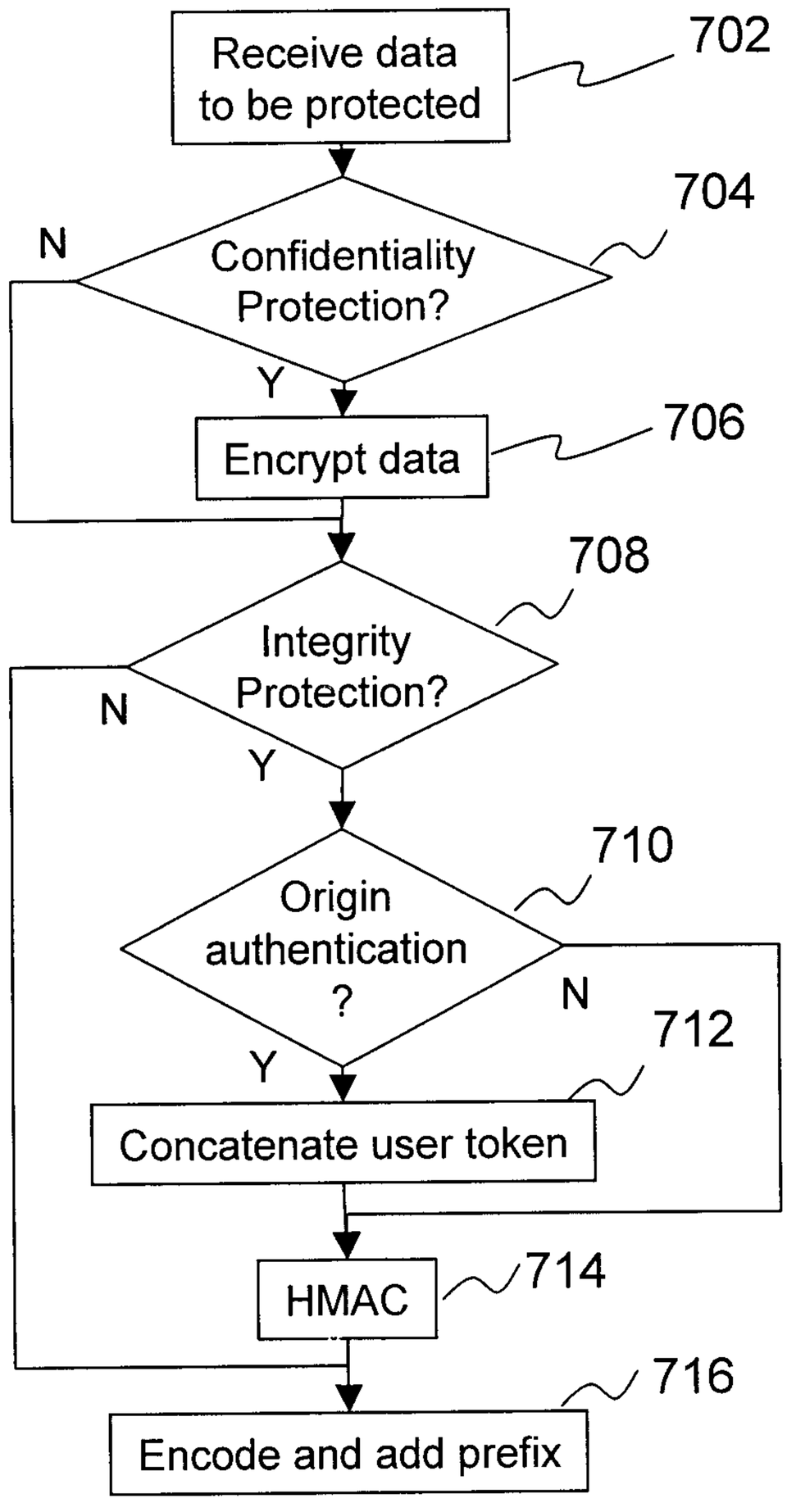
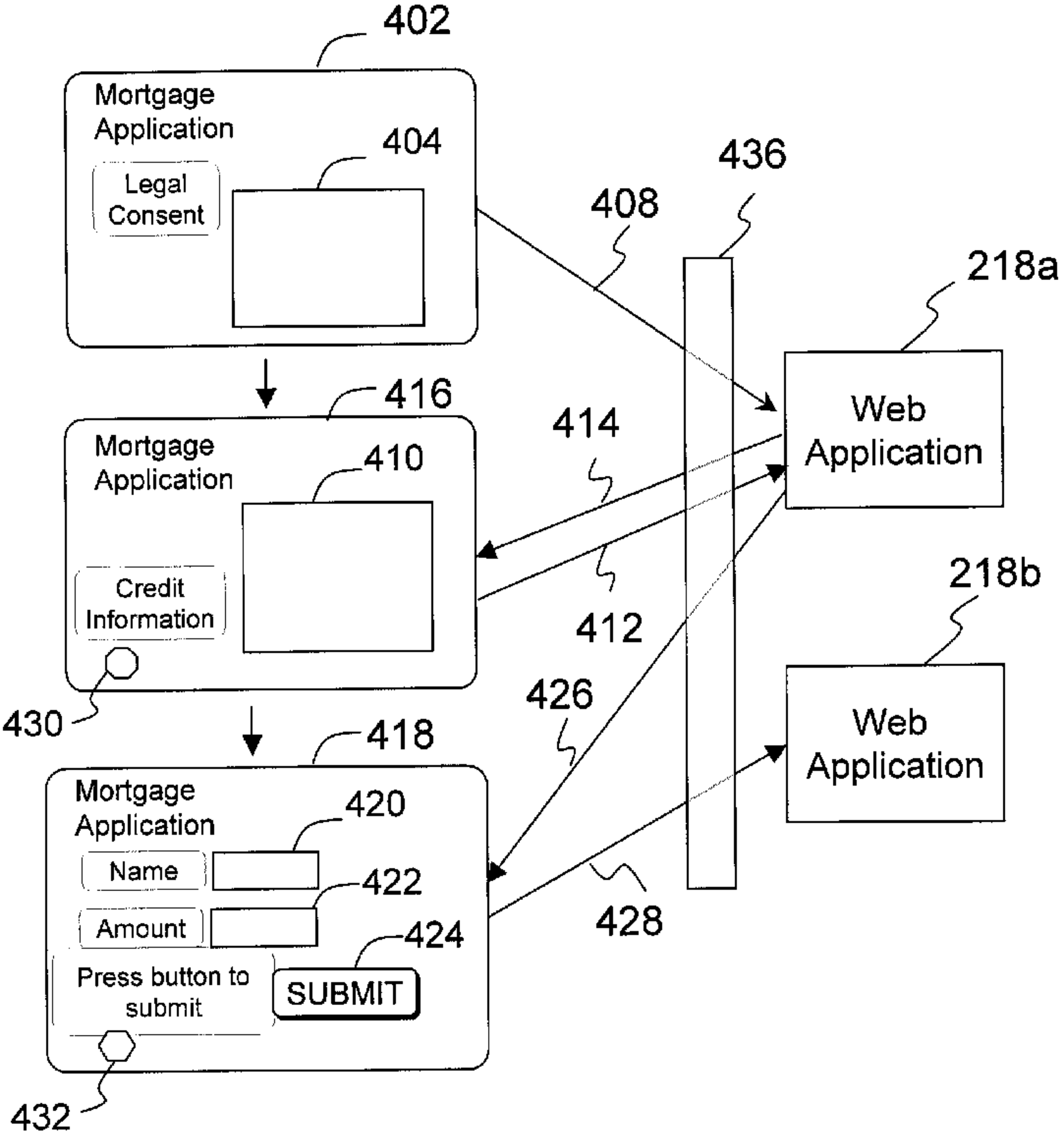

Receive data to be protected
702
Confidentiality Protection?
704
N
Y
Encrypt data
706
Integrity Protection?
708
N
Y
Origin authentication ?
710
N
Y
Concatenate user token
712
HMAC
714
Encode and add prefix
716

Figure 7


402
Mortgage Application
404
Legal Consent
436
408
218a
Web Application
414
416
Mortgage Application
410
Credit Information
412
218b
426
430
418
Web Application
Mortgage Application
420
Name
422
Amount
424
Press button to submit
SUBMIT
428
432