(57) Procédé selon lequel une classe qui indique le degré de liberté d'accès est attribuée à un paquet (10, 13) de données, dans le cadre d'une liste de contrôle d'accès (ACL), pour éviter l'accès non désiré à ce paquet. En outre, certains privilèges sont assignés à ce paquet (10, 13) de données, privilèges qu'un appareil de traitement (30) de données doit remplir pour obtenir l'accès au paquet (10, 13) de données. Ces privilèges peuvent être mis en mémoire dans l'appareil de traitement (31, 41) de données qui dispose du paquet (10, 13) de données. Une demande de paquet (10, 13) de données peut avoir lieu de manière protégée dans le cadre d'une procédure de sécurité.

(57) A class indicating the degree of open access is allocated to a data packet within the framework of an access control list (ACL) in order to avoid undesired access to said data packet (10, 13). Specific privileges which must be met by a data processing unit (30) in order to gain access to said data packet (10, 13) are also assigned to the data packet (10, 13). These privileges can be stored in the data processing device (31, 34) which is provided with said data packet (10, 13). A request for a data packet (10, 13) can occur in a protected form within the framework of a security procedure.
Abstract

System and method for granting accesses to information in a distributed computer system

In order to avoid undesirable access to a data packet (10, 13) the latter is assigned, within the context of an access control list (ACL), a class which indicates the degree of freedom of access. The data packet (10, 13) is additionally assigned to specific privileges which a data processing device (30) has to satisfy in order to obtain access to the data packet (10, 13). These privileges can be stored in the data processing device (31, 41) which has the data packet (10, 13) available. A request for a data packet (10, 13) can be made in a protected way within the context of a security procedure.

Figure 10
System and method for granting accesses to information in a distributed computer system

The invention relates to a method for granting accesses to information in a distributed computer system. The most important distributed computer systems at present are the Internet and, for business purposes, the Intranet. In these systems, network security is a critical factor for the success of these systems.

Computer networks are terminals, controllers, peripheral devices and processors and their connections to one another. Such configurations are defined as logical units, which have a relation with physical units in a network and specify the rules for interactions between these logical units. The components of a network are described, for example, in EP 0 362 105 B1, columns 1 and 2. Many other network systems which, depending on their configuration and their use, are generally operated as local or super-local networks, so-called Local Area Networks or Wide Area Networks, these being described corresponding to the network in EP 0 362 105 B1. All these networks can be connected to one another by using standardized protocols. When a number of such systems are connected together, this is referred to as a distributed computer system. A prominent representative of such a computer system is the so-called World Wide Web (WWW).

For the purpose of communication among the computer systems, and also within these computer systems, some transfer protocols have become established. One important transfer protocol is TCP/IP (Transmission Control Protocol/Internet Protocol).

Within the networks, some computers operate as so-called servers, which can be addressed by the actual users, the so-
called clients. The servers can provide different computing resources, documents, programs and so on. In particular, the servers provide the clients with so-called hypertext documents. This is the description for documents which are produced by means of a so-called "Hypertext Markup Language" (HTML). HTML is a language which produces documents that can be displayed at the client, irrespective of the platform. HTML is an application of the ISO Standard 8879: 1986 Information Processing Text and Office Systems; Standard Generalized Markup Language (SGML). SGML documents are a sequence of symbols which are organized physically as a set of entities and logically in a hierarchy of elements. A SGML document contains data in the form of characters and so-called "markups". These markups describe the structure of the information and reproduce an example of this structure.

In addition to text, graphics, video and the like, the hypertext documents often contain references, so-called links, to other hypertext documents. These links are commands in which instructions that make it possible for the system of the client to search for the target document are embedded. The target document can be assigned to a different server, which may be located in a different network that is a component of the distributed computer system. If the HTML standard is used for the hypertext document, the link then contains a so-called Uniform Resource Locator (URL), which indicates the actual name of a document and the server system from which access can be made to the document.

The HTML pages are transmitted between server and client by means of a specific transfer protocol. One example of such a protocol is the Hypertext Transfer Protocol HTTP, which is specified in the Internet Request for Comments 1945 (RFC1945). Such a protocol contains messages that are to be interpreted either as a request for a document
or as a reply to such a request. Requests and replies can be classified as a function of their content. For example, the HTTP protocol defines two request types: a so-called GET request, which contains only the requested URL and hence the sought document and the server system of the latter, and a POST request, which in addition to the URL also contains data which is added by the client.

The client computers, with which hypertext documents can be displayed and processed, have an appropriate application program, a so-called browser. The browser may be available on the client computer on its own or embedded in another application program. This browser reproduces the contents of the hypertext document, if necessary using further application programs, at the client in the form of texts, images, sounds and video sequences. In addition, a browser allows the links contained in the hypertext document to be followed. This branching to the links is carried out, from the point of view of the user, by a simple click with a computer mouse or another pointer instrument at an appropriately marked point in the displayed hypertext document.

The hypertext documents are provided by servers. For this purpose, the servers access a document directly or generate it dynamically. In the case of dynamic generation, the servers use application programs which, in a manner similar to a client, carry out a data exchange with a database by means of network protocols. The data read from the database are prepared as a hypertext document and sent to the client.

Hypertext documents can contain fields which have to be filled out by the client. The filled-out fields supply the necessary information to the server. For example, a hypertext document looks like a form that is to be filled out by the user.
This form is sent from the client to the server, where the information received is processed in an application program. The application program may in this case run directly on the server or in a further data processing device outside the server. If this functionality is used, it becomes possible to make only the browser available on a client computer. The hypertext document then serves as the user interface for an application program. This application program no longer needs to be installed at the client, instead it is sufficient to provide this application program at the server and to access it by means of a hypertext document and standard browser.

A special group of servers are the so-called proxy servers. These proxy servers only pass on the requests and replies which reach them to their intended destination. In addition, the proxy servers are capable of storing some replies temporarily in a cache memory. This is expedient, in particular, in the case of such replies which contain hypertext documents which are often requested. This functionality reduces the network bandwidth needed, since the amount of data to be transmitted is reduced. A further important task of the proxy server is to carry out security routines. Proxy servers are used to treat incoming messages from a security point of view, by encoding these messages, for example.

When hypertext documents and hypertext user interfaces of applications were made available over distributed networks, a few security requirements arose. These requirements relate, on the one hand, to security during data transmission and, on the other hand, to access to individual documents or groups of documents. With regard to data transmission, use is made of various encoding methods, with which the transmitted data are made difficult to read for third parties or are made completely illegible. In the case of access
protection for documents, access is permitted or rejected in a user-specific fashion.

General access protection is described, for example, in Internet Request for Comments 1508 (RFC1508) for the so-called Generic Security Services Application Program Interfaces (GSS-API). A user/client transmits data that prove its identity to the server. For example, this is carried out by exchanging passwords or by using specific cryptographic methods.

Once the identity of the client has been determined, the result can be stored in a list and used for subsequent access decisions.

One method for exchanging passwords in accordance with the Hypertext Transfer Protocol HTTP is described in Internet Request for Comments 1945 (RFC1945). A cryptographic method for data security is described in the Secure Socket Layer Protocol (The SSL Protocol, Version 3.0, Internet Draft, March 1996). An asymmetric method, a so-called Public Key Method (RSA), is proposed for mutual authentication between server and client.

The known methods have the disadvantages listed below:

1. Problems with the management of situations in which a plurality of clients, whose access wishes must be controlled, wish to access a plurality of servers. One reason for this is the outlay on administration which has to be made when the decision about granting access is based on the specific rights of a client. This type of allocation of rights means that each data source (server) must have an access control list, in which all the clients with access rights must be contained. Such lists are very comprehensive and are difficult to maintain in such a way that they always exhibit a current status. It therefore often occurs that clients whose access rights have already
lapsed still have access to protected data because some
time passes until the access control list is updated.
2. The allocation of rights is typically carried out by
the administrator of the server. However, it would
frequently be better to transfer this right to a
security administrator or to leave this right with the
owner of the data.
3. The servers always confirm the existence of a
hypertext document, even when access to this document
is subject to access control and is not granted. This
confirmation may inherently already denote the issuing
of security-relevant information.
4. The access decision is made exclusively on the basis
of the identity of the client.
5. The identity of the client is always disclosed to
the server. In some cases, this approach harbors an
infraction of the regulations relating to data
protection, since the administrator of the server can
determine which client has access to which document.
6. As described above, many hypertext documents are
generated dynamically at the server. For this purpose,
the server makes access, via a network protocol, to
another system, where the actual content of the
document is stored. In these cases, security methods
which are used at present cannot provide a secure
connection between the server and the database in the
background in such a way that these security methods
are based on the security attributes of the actual
client browser.
7. It is frequently necessary for the actual content of
documents to be changed in order to ensure secure
access to them. For example, the SSL method requires
that the URL of a document have a specific type.

In the prior art, solutions relating to some of
these problems in conjunction with specific security-
application programs have been shown. For example, the
European Computer Manufacturers Association (ECMA), in
its technical report
ECMA TR/46, has defined a security architecture whose access control is based on attributes which are assigned to the client.

A particular embodiment of this method was developed under the description SESAME (Secure Europeans System for Applications in a Multivendor Environment). SESAME is defined in the corresponding Standard ECMA 235.

A common method for integrating security functionality in an application program was defined in RFC 1508 in conjunction with RFC 2048. These contain the definition of a so-called Application Program Interface (API), this being known as the so-called Generic Security Services API (GSS-API).

The present invention is based on the object of indicating a system for granting accesses to information in a distributed computer system that ensures simple access control which is always up-to-date in a largely anonymous form.

This object is achieved by the features specified in the claims.

The protection of data packets, in particular of hypertext documents, is carried out not by determining the identity of the person making access but by using the properties of a person making access. These properties have to be divulged only for the purpose of access to a protected document. The outlay on administration for the access is extremely low, since it is not necessary to maintain lists specific to persons making access in which all the data packets to which access is legal are listed. The currency of the access authorization is always ensured.

As a result of the introduction of at least three security classes, of which a first security class (open) grants unrestricted access to
a data packet, and a second security class (entry) and a third security class (hidden) grant conditional access to a data packet, the ability to find data packets can be made so difficult that even the existence of data packets can be veiled or completely concealed. Thus, any information about the existence of a data packet in the third security class is suppressed, while in the case of data packets in the second security class, reference to their existence is made. By means of the temporary storage of the properties specific to a person making access, the outlay for data exchange is minimized and therefore the response time (performance) is improved.

In the case of the World Wide Web, data packets are exchanged in the form of hypertext documents. These are divided into at least three categories. A distinction is thus drawn between hypertext documents with open access with the classification "open", hypertext documents with access control, so-called "entry documents", and hidden hypertext documents, so-called "hidden documents". Following an appropriate request, which can be made by any data processing device having the properties of a browser, a hypertext document with the "open" classification is transmitted to said browser. Hypertext documents with the "entry" and "hidden" classifications can only be made available to those browsers whose clients have been authenticated before access. Within the context of this authentication, privileges and thus access rights must have been defined which are sufficient to permit access to the desired hypertext document. If a request for access to a hypertext document with the "entry" or "hidden" classification comes from the browser of a client or the privileges which have been defined during the authentication of the client are not sufficient for access to the hypertext document, then the server will deny the existence of the hypertext document. This denial is communicated to the client by
transmitting an error message in accordance with the transfer protocol used.

For the case in which the privileges of the client have not yet been defined within the context of an authentication, and the requested hypertext document is one from the "entry" category, the error message may also contain a reference to the fact that the server administers some protected hypertext documents. This reference may be used by the browser of the client as a trigger for the start of the authentication procedure.

Access to the hypertext documents is controlled by the "open", "entry" or "hidden" classifications. In addition, properties, so-called privileges, must be stored at the server, possession of which privileges a browser client must demonstrate in order to obtain access to hypertext documents classified as "entry" or "hidden". In order to be able to demonstrate these privileges to the server as required, these must be stored at the browser client. If this is a set of privileges which is generally valid, these access rights must be stored, at least temporarily, at the browser client. The present invention therefore meets the requirements of the Secure Socket Layer SSL, according to which it must also be possible for the access rights to be stored over a relatively long period of time. As an alternative to this, in the case of the present invention the conditions of the ECMA SESAME project can also meet, and generate the access rights dynamically with the aid of a so-called Network Security Service.

If the access rights of a browser client have been registered in a secure way at the server, these access rights are intermediated stored at the server. This has the advantage that the access times to hypertext documents are reduced. However, this intermediate storage makes it necessary for subsequent requests and replies to be authenticated in such a way that the server knows with certainty
which browser client is involved. Within the time period during which the access rights of the browser client are intermediately stored at the server, it is accordingly sufficient for the browser client to be authenticated at the server. Repeated transmission of the access rights can be dispensed with. This is therefore of particular advantage, since connections to specific servers only ever exist for a short time, during the access to a specific hypertext document, and are subsequently interrupted again. For these constantly repeating access wishes from the browser client and the replies of the server to them, it is possible for an encoding method to be used in order to ensure data protection.

According to an embodiment of the invention, the person making access, in particular the browser client, has the option of transmitting only a reference to his access rights to the server. The server can then make access to the location referred to, and obtain the access rights of the current client. The location referred to may be another network service or a local memory in the server.

According to a development of the invention, in order to communicate the access rights or a reference to the access rights, a security protocol is used. The transmission to the server takes place via so-called Security Protocol Data Units PDU, which are embedded within the used for the access to the hypertext document of the transfer protocol.

According to a further development and refinement of the invention, the server has a method available to it for forwarding access rights to following data processing devices and therefore to server systems of the databases. As a result, it becomes possible to adapt the content of a requested hypertext document to the access rights of a browser client. This functionality is particularly advantageous
in those cases in which the hypertext documents are used as an interface for applications which run outside the browser client
and which further process the data available to said client. The result of this further processing is presented to the browser client in the same way as in an application program which runs locally. The scope of the result is determined here by the stored access rights. Repeated authentication between browser client and server and for each following server to which access has to be made within the context of the arrangement is not required for this purpose.

A server can be used to convert the access rights of a browser client into a specific transfer protocol which can be processed by following servers which do not contain the subject of the present invention. The interposed server which performs this conversion does not need to contain the requested hypertext document and thus serves as a so-called proxy server. Using this functionality, a multiplicity of servers which do not satisfy the security mechanisms of the present invention can be inserted into the security system via the proxy server. The servers following the proxy can continue to be operated in unmodified form. Even the hypertext documents or the programs used to produce a document which are present on such an existing server can remain unmodified, and access is nevertheless made possible via the security mechanism according to the invention.

In a corresponding way, the browser client can also be preceded by a proxy which is used to store the access rights and to transmit them in a secure manner.

This makes it possible to use the present invention in the browser client without having to modify a browser program previously used. In such a case, the present invention provides access control, encoding and authentication between the proxy on the browser-client side and the server which has the features of the present invention. If the server does not have these
features, it can also be preceded by a corresponding proxy server having these features, so that previously existing browsers and servers both on the client and on the server side can continue to be used.

The introduction of the present invention can thus be carried out without modifications having to be made on the sides of the existing browser, server, hypertext document and corresponding programs.

The invention will be explained in more detail below with reference to the drawing, in which:

Figure 1 shows a schematic illustration of hardware components of a data processing network,

Figure 2 shows the essential software components of the data processing network according to Figure 1,

Figure 3 shows a schematic illustration of a link from one hypertext document to another,

Figure 4 shows a schematic illustration of the conversion of a link according to Figure 3,

Figure 5 shows an example of an access control list,

Figure 6 shows an outline of a server module with a security mechanism,

Figure 7 shows an outline of a browser module with a security mechanism,

Figure 8 shows an outline of a server without a security module, and an additional proxy module with a security mechanism,
Figure 9 shows an outline of a browser without a security module, and also a proxy with a security module,

Figure 10 shows a visualization of a data exchange for access to a hypertext document in the entry class,

Figure 11 shows a tabular representation of different address forms and address components,

Figure 12 shows a flow chart for the access mechanism on the browser side,

Figure 13 shows a flow chart of the access mechanism on the server side and

Figure 14 shows a detail of the flow chart of the access mechanism on the server side according to Figure 12.

Figure 1 shows a schematic illustration of a data processing network. This data processing network contains a personal computer PC or workstation PC, which is connected to a host computer 21 via a network line 22. This host computer 21 is in turn connected via a further network line 23 to at least one further host computer 24. The host computers 21, 24 and the personal computer PC are a component part of a known network architecture.

The exemplary embodiment uses this network architecture for the needs of a distributed computer system, in particular for the so-called World Wide Web WWW. The software components of this WWW are illustrated in Figure 2. Software which is referred to as a browser 30 and enables access to hypertext documents 10 (see Figure 3) is loaded at the personal computer PC, the so-called client. This browser 30 is
also able to support the security mechanisms according to the invention. If it does not do this, a proxy which satisfies these security requirements may optionally be available.
at the personal computer PC. This proxy 35 then satisfies the security requirements according to the invention. The network line 22 transmits data from the browser 30 to the host computer 21 via a transfer protocol 32. One example of such a transfer protocol 32 is the hypertext transfer protocol HTTP. At the host computer 21, the transmitted data is accepted by a web server program 31, referred to below as server 31. If this server 31 does not contain the necessary security mechanisms, a proxy program, referred to below as proxy 31, which supports these security mechanisms according to the invention, can be inserted between the network line 22 and the server 31. Via the network lines 23, the server 31 has access to network components running in the background. For example, the server 31 can access a database 34 via a specific transfer protocol 33. For access to this database, the network protocol 33 may differ fundamentally from the network protocol 32. The network protocol 33 can be adapted specifically to the requirements of the database 34.

Using the distributed computer system according to Figure 1 and the software distributed thereon according to Figure 2, a data exchange is carried out in the form of so-called hypertext documents 10. Such hypertext documents 10 are illustrated in Figure 3. A hypertext document 10 contains data in the form of symbols 11, images 12, videos 12 or audio sequences and in the form of references 14, so-called links, to further hypertext documents 13.

These hypertext documents 10, 13 are assigned to different servers 31, 41, as shown in Figure 4. The hypertext document 10 is assigned to the server 31, and the hypertext document 13 is assigned to the server 41. If the hypertext document 10 is called up by the browser 30, this call is transmitted to the server 31 via the transfer protocol 32. The call contains the name of the server 31, the name of the hypertext document 10 and the location at which this hypertext document 10
is stored at the server 31. This information is known as the so-called Unique Resource Layer URL. The
transfer protocol used, HTTP in this example, is also specified in the URL. A complete URL reads, for
example, HTTP://Host.sse.ie/documents/doc1. Once this document has been found on the basis of the URL, it is
packaged in accordance with the rules of the transfer protocol HTTP used and is communicated to the browser
30. The browser 30 is capable of displaying this
hypertext document 10 on the screen of the personal
computer PC or of making it audible. The hypertext
document 10 contains a link to the hypertext document
13. This link is a complete URL. On the basis of this
URL, the browser 30 sends a request, packaged in a
transfer protocol 32, to the server 41 which is named
in the URL. Using the information from the URL, the
server 41 is able to find the hypertext document 13 and
transmit it to the browser 30. The browser 30 can also
display the hypertext document 13.

Access to hypertext documents 10, 13 can also
be suppressed. In this case, three categories of
hypertext documents 10, 13 are to be distinguished:
generally accessible documents, restricted-access
documents and hidden documents. These categories are
also referred to as "open", "entry" and "hidden". Each
server 31, 41 which administers documents classified in
this way has an access control list ACL. Such an access
control list ACL is illustrated in Figure 5. The access
control list ACL contains document-specific statements
about classification of the document, necessary
privileges for access to the document and supplementary
instructions for carrying out the access to the
hypertext document 10, 13. The document specification
is carried out by entering the complete URL in the
first column of a row. The classification of the
document is carried out in the three categories "open",
"entry" and "hidden". It is also possible for such a
classification to be omitted entirely.
Nevertheless, role-dependent access mechanisms (privileges) can be used in such a system.

The privileges which are necessary for access to the hypertext document are entered in the third column of a row. Access rights are subdivided into roles and graduations of the roles. Such roles are known from ECMA 235 and ECMA TR/46, where they are compulsorily defined. Examples of roles are administrators, role: admin., openness of the document, clearance: top secret or clearance: restricted, and clients. These individual roles may have different degrees of authorization, for example degrees 1 to 6. It is also possible to define only an authentication as an access prerequisite. In this case, for example, the right ANY is entered in the third column.

The fourth column contains instructions which the server has to carry out if the desired document cannot simply be read from a memory and sent. Figure 5 shows two types of instructions: the "program" instruction contains the name of a program, for example "AdminP" which the server 31 has to carry out. Such a program then produces a hypertext document 10, 13 and makes it available to the server 31. The server 31 can then send it to the browser client 30. The instruction "ad on" is used to instruct the server to append a command sequence, a so-called "string", to the current request in the HTTP format, this containing the access rights cited in the table in relation to the requested document 10, 13. The request expanded in this way is passed on to another server. A condition for this type of command execution is that the hypertext document is produced by a program or is to be obtained from another server via the HTTP protocol.

A decision as to whether access to a document referred to in the first column of Figure 5 is permitted
is based exclusively on the document name specified in this column. If the requested document name is not contained in this column, but this document is available on the server, then this fundamentally concerns a document in the "open" class.

Figure 6 shows the functional component parts of a server 31 which contains the present invention. On the network side, the server contains a transfer protocol converter 61. This receives requests coming in via the network 25 and the network lines 22. These requests meet the condition of the TCP/IP network protocol and the HTTP format. By means of its "parser functionality" the protocol converter 61 is able to convert incoming requests into a form such that they can be further processed in the server. It is likewise capable of converting replies going out from the server and corresponding to the requirements of the network 25 in a form which can be transmitted there, in accordance with the transfer protocol used. A further component of the server is the access control unit ACU. This unit ACU has an algorithm which, using the designation of a hypertext document 10, 13, makes access to the access control list ACL according to Figure 5. Using the decision criteria contained in the algorithm, the access control unit ACU is placed into the position of making a decision relating to a right of access to this hypertext document 10, 13. The access control unit ACU is also able to output further information contained in the decision table ACL, in particular that from columns 3 and 4.

A further server component is the privilege generator 63. This unit 63 is able to determine the access rights of a person making a request. Furthermore, it is responsible for protecting the requests and replies. To do this, it uses a security context. A privilege generator of this type is, for example, the GSS-API, as described in RFC 1508. The access rights are generated temporarily
by the privilege generator 63 for a current access from
a browser client 30 to a server 31, 41.

A further unit of the server 31, 41 is the
intermediate memory 64. The access rights generated in
the privilege generator 63 are temporarily stored in
said memory. Likewise, the security context which may
be used is temporarily stored in this intermediate
memory 64. The duration of storage in the intermediate
memory 64 can be restricted to a current access, but
may also last for a number of hours or days. The
duration of storage can be defined freely.

The three components "access control unit ACU", 
"privilege generator 63" and "intermediate memory 64"
are controlled by the security control unit 65. This
unit 65 contains a status machine that enables or
inhibits actual access to a hypertext document 10, 13.
In order to be able to implement this decision, it
corresponds with the abovementioned units ACU, 63, 64.

The server 31, 41 contains a further component
66. This document access unit 66 is able to make access
to the desired hypertext documents 10, 13 in a document
memory 67. If this is a hypertext document 10, 13 which
first has to be produced by a program sequence, the
document access unit 66 is able to start external
programs 68 which produce the desired hypertext
document 10, 13. For this purpose, the document access
unit 66 corresponds with the program execution unit 68
via a generally used interface, such as the Common
Gateway Interface (CGI) or a specific interface. When
the desired hypertext document 10, 13 is available, the
document access unit 66 transfers said document to the
security control unit 65.

The components that are available on the
browser-client side 30 are illustrated in Figure 7. The
browser 30 contains
a user interface 72 that transfers the hypertext documents 10, 13 for display on the personal computer PC. To this end, the user interface 72 converts the format of the hypertext document 10, 13 in such a way that it can be reproduced by the monitor or sound card functional units. In terms of their functionality, the remaining components of the browser 30 correspond to the corresponding units 63, 64, 65, 61 of the server. The privilege generator 73 generates the available access rights and provides the corresponding security context mechanisms needed for the protection of the requests to be sent and the replies received. Just as in the case of the server 31, 41, these mechanisms are defined, for example, by the GSS-API in accordance with RFC 1508.

The access rights and security contexts received are temporarily stored in the intermediate memory 74. These two units 73, 74 are coordinated by the security control unit 75 on the browser side, which passes on the corresponding information in the direction of the network. A protocol converter 71 is arranged between the network 25 as security control unit 75 and, with its browser functionality, converts the information in accordance with the network protocol used and the display form of the information used in the browser 30.

Servers 31, 41 which do not contain the components according to Figure 6 are already in use. In order to be able to continue to use such servers 31, 41, a proxy server 36 according to Figure 8 can be inserted between the network 25 and server 31, 41. Such a proxy server 36 is largely identical to a server as illustrated in Figure 6. However, hypertext documents 10, 13 are not permanently stored in the proxy server 36. Storage is only exceptionally carried out for frequently requested documents 10, 13. On the network side, the proxy server 36 contains the protocol converter 61 via which requests and replies are
converted. A security control unit 85 on the proxy-server side corresponds
to the access control unit ACU, the privilege generator
63 and the intermediate memory 64, in order to decide
whether it should pass on a request for a hypertext
document 10, 13, to the server 31, 41 or not. If the
hypertext document 10, 13 is not stored at the server
31 but must be generated dynamically, this procedure
then has to be initiated, starting from the proxy
server 36. The security control unit 85 contains a
corresponding interface 86 for this. This ensures that
the program execution unit 68 coupled via the interface
86 can receive the necessary access information.

The server 31 is coupled to the proxy server 36
via a second protocol converter 87 of the proxy server
36. This protocol converter 87 converts the data in
accordance with the in the network protocol of a
security network 89. The security network 89 connects
the server 31 to the proxy server 36. On the security
network 89, the security-relevant data is transmitted
in a protected form, as is known, for example, from
GSS_API.

Browsers 38 already in use can also continue to
be used on the browser side. The browsers 38 already in
use comprise the user interface 72 and a protocol
converter 94. From the protocol converter 94, requests
pass, via a network protocol 92 of a security network
93, to a proxy browser 35. The latter contains a
further protocol converter 91, which converts the
requests from the security network 93 for the proxy
browser 35, and converts the replies from the proxy
browser 35 for the security network 93. The other
components 73, 74, 71, 75 of the proxy browser 35 are
identical to those of the browser 30 from Figure 7.

The data exchange between browser 30 and server
31 is illustrated in Figure 10 using the example of an
access to a hypertext document 10, 13 in the "entry"
class. The browser
30 sends a request GET for a hypertext document 10, 13 to the server 31. The browser 30 does not know at this time that this is a protected hypertext document 10, 13 in the "entry" class. It therefore sends a hypertext transfer protocol HTTP-defined GET request. On the side of the server 31, the security control unit 65, 85 determines, by using the access control unit ACU, that this is a protected hypertext document 10, 13. An error message 101 is then generated and sent to the browser 30. This error message 101 contains an indication of the fact that the desired document exists but that access to it is not free. This error message 101 can be interpreted by a conventional browser 38 only as an actual error message which does not provide any further information about the existence and availability of the desired hypertext document. In a corresponding way, an error message is output on the personal computer PC with the aid of the conventional browser 38. In the case of a browser 30 according to the invention, or in the case of a conventional browser 38 which is preceded by a proxy browser 35, the additional information which refers to the existence of the document is recognized and interpreted in such a way that a further request is sent to the server 31. This further request is packaged by the browser 30 into a particular form and is enhanced with additional information.

The original Unique Resource Locator URL begins, as shown in Figure 11, with the statement of the network protocol to which this request relates. In the exemplary case, this is the HTTP protocol. This is followed by the domain of the server, which contains the name and the address of the server 31 addressed and, if appropriate, the TCP port number, e.g. 8080. This TCP port number is treated in the same way as the domain of the server. Finally, the path and the document name of the sought hypertext document 10, 13 are specified in the last part of the URL. From this URL, the browser 30 extracts the domain, as is
indicated in the DNS line in Figure 11. DNS is a designation of an external address service provider, in this
The case of the "Internet Domain Name Services". The DNS obtained, "Host.sse.ie" is abbreviated by the components at the left-hand edge. The remainder yields the so-called SESAME domain, "SSE.IE". Within this SESAME domain, the principal name reads "websec.Host.sse.ie". The SESAME domain can be interpreted correspondingly by the server 31. The protocol statement "websec" is used for this. A complete principal name, as is shown in Figure 11, last line, is formed from this principal name. To this end, the path designation "/SSE.IE@SSE.IE", for example, is appended to the principal name. The access rights of the browser client 30, and a protected copy of the actual HTTP Get request for the desired hypertext document 10, 13 are contained in this path name. This complete principal name is transmitted as a request 102 from the browser 30 to the server 31.

The security control unit 65, 85 of the server 31 extracts both the name and location of the desired hypertext document 10, 13 and the access rights of the browser client 30, also supplied by the browser 30, from the data transmitted with the request 102. There are cases in which, in order to transmit the entire request 102, a multiple data exchange between the browser 30 and server 31 is necessary. An example of this is described in the GSS_API mechanism.

If the access rights that were transmitted with the request 102 to the server 31 are sufficient, then access to this hypertext document 10, 13 or to a corresponding program in the program execution unit 68, which in turn supplies the desired hypertext document 10, 13 to the server 31, is carried out via the document access unit 66. Access is correspondingly made from a proxy server 36 to the server 37, which returns the hypertext document 10, 13, or via the security control unit 85 to the program execution unit 68, which returns the appropriate result.
The hypertext document 10, 13 which is thus available to the server 31 or to the proxy server 36 is packaged into a protocol-conformant reply, possibly encoded and sent back to the browser 30. The latter processes the hypertext document 10, 13 in such a way that, in turn, it is displayed at the client on its personal computer PC. The method sequences in the browser 30 will be explained below using the flow chart according to Figure 12.

In a first step 201, the browser 30 outputs a request in the form of a Get request to the security control unit via the user interface 72. The requested hypertext document 10, 13 belongs to a security class "open", "entry" or "hidden", which is not known at the browser side. Further security classes can be defined, in virtually any desired number. The only restriction for the number of security classes is the amount of classes to be processed with a sensible outlay on administration.

If a browser 38 of conventional type is used, the document request passes via the user interface 72, the protocol converter 94, the security network 93 and the protocol converter 91 on the proxy browser side to the security control unit 75. In any case, the security control unit 65, 75 will perform, in step 202, a comparison between the desired server domain and server domains which are known on the browser side and, with step 203, chooses to make a decision.

If the decision is positive, a context is stored in the intermediate memory 74. In this case, this context is used for packaging the request. This packaging and unpackaging is known under the terms "wrapping" and "unwrapping" within the scope of the GSS_API, RFC 1508. The GSS-WRAP and GSS-UNWRAP functions are used as follows in the invention: the security control unit 75 receives a Get message, for example "Get URL1" in clear text. A corresponding
context is taken from the intermediate memory 74. This context is used for packaging the information "Get URL1" from which the packaged message is derived. The packaging function GSS.WRAP is given the abovementioned information, whereupon this function supplies the packaged message as a result. The packaged message is a so-called paqueoctet string whose internal structure is not known to the caller of the GSS.WRAP function. The packaged message is formatted as a HTTP post message having the form "POST/Special Place <...>". This completes the fourth step 204.

In the fifth step 205, the HTTP post message is transmitted to the server 30. The server 30 then determines from whom this message originates and searches its intermediate memory 64 for a suitable context for this browser client. If such a context is available, the server 30 unpackages the HTTP post message with the aid of the GSS.Unwrap function, to which it transfers the determined context and the packaged message. This function supplies the clear text of the HTTP message "GET URL1", with which the desired hypertext document 10, 13 is adequately identified. This unpackaging operation can be carried out only by the addressed server 31. The processing of the request of the designated hypertext document can be seen in the context of the description relating to Figures 13 and 14.

If the processing on the server side has supplied a positive result, the requested hypertext document 10, 13 is transmitted to the security control unit on the browser side via the network 25 and the protocol converters 61, 71 on the server and browser sides. In step 206, the hypertext document supplied is unpackaged in the manner described above. The unpackaged hypertext document 10, 13 is either a warning message, which informs the browser client 31 about the failed access wish or the requested hypertext document 10, 13. The warning message is displayed on the personal computer PC in step 207,
and the desired hypertext document 10, 13 is displayed on the personal computer PC of the browser client 31 in step 208.

For the case in which it is established, in step 203, that no context for the server 31 has been stored in the intermediate memory, the browser client 30 sends an unprotected request to this server 31 in step 210 and waits for a reply. In step 211, this reply is then checked as to whether it is an error message. If this is not the case, in step 213 the reply is displayed in the personal computer PC of the browser client 30. If, on the other hand, it is an error message, a check is made, in a further step 212, as to whether the error message has had a security-relevant indication, according to which it is a document in the "entry" class, appended to it. If this is not the case, the error message is displayed on the personal computer PC in step 213. If, however, there is such an indication, then a method for defining a context in relation to the desired server 31 is initiated by a step 214.

The method according to step 214 for creating a so-called Initial Context Token ICT is a GSS-API concept, which designates such data of a protocol which are exchanged between units with the intention that a security context between the two points is agreed. More on this point is stated in the introduction to Internet Requestor Comments 1508 (RFC 1508). This concerns a multi-stage process, at the end of which the complete context for the desired server 31 is present and is stored in the intermediate memory 74. A first stage of the ICT is generated. In step 215, an interrogation is made as to whether the ICT has now been formed completely. If this is not the case, the previously formed ICT is sent to the desired server 31. This server 31 generates a reply ICT and sends this to the security control unit 75 on the browser side (see also
Figure 14). The security control unit 75 on the browser side waits for a reply from the server 31 in step 217. When said reply is present, this reply is processed and, in a step 218, it is then determined whether the Initial Context Token ICT is completely present. If this is the case, steps 214, 215, 216, 217 and 218 are repeated. If, finally, the complete ICT is present, the corresponding context is stored in the intermediate memory 74 in step 219. The completely present ICT is reported to the step 204 and transmitted. As a result, the renewed sending of the Get message is initiated, with which access to a specific hypertext document 10, 13 of the desired server 31 is to be obtained. To this end, steps 204, 205, 206 and 207 or 208 are executed.

The security strategy on the server side is explained using Figure 13. In general, the sequences described below take place in the server 31. If the hypertext documents 10, 13 are stored in a conventional server 37, however, then the method sequences described are carried out in the proxy server 36.

The server 31 receives a Get request via the network 25 and its protocol converter 61. When this Get request is received, the security control unit 75 conditions the Get request in such a way that, using the access control unit ACU and the access control list ACL contained therein, it is able to decide whether a hypertext document 10, 13 of the "open" class is concerned. This question is answered in step 302. If the reply is positive, the security control unit 65 requests the desired hypertext document 10, 13 via the document access unit in the document memory 67 or the program execution unit 68. If the documents are administered by another server 37, a transmission of this document from the server 37 to the proxy server 36 is correspondingly initiated. The hypertext document 10, 13 obtained is transmitted to the browser client 30 and reproduced there.
If a document of the "open" class is not concerned, a check is made in step 304 as to whether a protected request is concerned, and whether this is a request which contains an Initial Context Token ICT. If neither of the two conditions is met, in a step 305, a check is made by the access control unit ACU, on the basis of the access control list ACL, as to whether a hypertext document 10, 13 of the "entry" or "hidden" class is concerned. If a document 10, 13 of the "entry" class is concerned, in a step 306 a corresponding error message is sent to the browser 30, with an indication that an "entry" document is concerned. The browser 30 processes it according to the way described in Figure 12. If a hypertext document 10, 13 of the "hidden" class is concerned, an error message is sent to the browser in a step 307. It is not possible to derive, from this error message, whether this document exists or not.

If it is determined in step 304 that a protected request and/or a request that contains an ICT is concerned, then in a step 308 such requests for further processing of the ICT are branched to a step 309.

Steps 309 will now be explained in more detail using Figure 14. The ICT is processed in a first step 401. If it is established during the processing that a faulty ICT is concerned, a corresponding error message is sent to the browser 30 in a step 402, and the procedure is terminated. If the processing of the ICT can be concluded in a regular fashion, a check is made in a step 403 as to whether the ICT is already completely present. If this is not the case, in a step 404 a corresponding request is sent to the browser 30 which, as described in more detail in connection with Figure 12, answers this question after it has been processed. If the complete context of the ICT does not emerge from this reply,
steps 401 to 404 are repeated. If, however, the completeness of the context is determined with this step 405, or if its completeness is determined in the following process step 401 and 403, then the context and the determined access rights of the browser client 30 are stored in the intermediate memory 64 with the step 406. After this procedure has been concluded, a branch is made in step 407 back to the step 309 of Figure 13 again.

Once this step 309 has been completed, or if it is determined in step 308 that no ICT is contained in the request, then it is determined, in a step 310, whether a context for the browser client 30 is contained in the intermediate memory 74. If this is not the case, at step 311 an error message is sent to the browser client 30. If a context for the browser client 30 is present, in a step 312 the request is unpackaged. If the request is invalid, in a step 3, a corresponding error message is sent to the browser client 30. If the request is valid, a check is made by the access control unit ACU, with the aid of the access control list ACL, as to what access privileges must be present for the purpose of access to the hypertext document 10, 13. These requests are compared by the access control unit ACU with the access rights of the browser client 30 that are stored in the intermediate memory 64. If the stored access rights are inadequate, an error message is generated in a step and, using the context, is packaged and sent in protected form to the browser client 30. However, if the access control unit ACU determines in step 314 that the access rights are adequate, the access control unit ACU checks, in step 316, as to whether further instructions for access to the desired hypertext document 10, 13 are contained in the fourth column in the access control list ACL. If this is not the case, the access control unit reports this to the access control unit 85, whereupon the latter requests the hypertext document 10, 13 in
step 317, packages it and sends it to the browser client 30.

If, however, further instructions are present in the access control list ACL, in a step 318 the corresponding instruction is read from the access control list ACL and output to the security control unit 85, whereupon the latter initiates the implementation of the instructions. The security control unit 85 waits until the instructions have been carried out and it has the result, as a rule a hypertext document 10, 13. In a step 319, the hypertext document 10, 13 obtained is packaged and thus returned to the browser client 30 on a secure path.
Patent Claims

1. A system for granting accesses to information in a distributed computer system, in which a plurality of data processing devices, in particular personal computers (PC) and servers (31, 41), are connected to one another in a network-overlapping fashion by means of transfer protocols (TCP/IP), and random access to data packets (10, 13) is possible, irrespective of the location at which they are stored, data packets (10, 13) being contained which are protected against random access by access rights which, if they are present, enable access to the data packet (10, 13), characterized in that access rights are defined for each data packet, independently of other data packets, by means of allocation to a security class, for which specific properties of a person making access (30) can be defined.

2. The system as claimed in claim 2, having an allocation of the data packets (10, 13) to at least three security classes, of which a first security class (open) grants unrestricted access to a data packet (10, 13), and a second security class (entry) and a third security class (hidden) grant conditional access to a data packet (10, 13), it being possible to define a plurality of properties which can be allocated in a manner specific to data packets and, before access to the data packet (10, 13) is enabled, a check being made for the presence of the properties allocated to a data packet (10, 13), and, for data packets (10, 13) of the second security class (entry), for the case in which the properties of the person making access (30) are unknown, requests the latter to communicate his properties.

3. The system as claimed in either of claims 1 and 2, in which the properties of a person making access
(30) can be stored temporarily in the data processing device (31, 41).
4. The system as claimed in either of claims 2 and 3, in which a person making access (30) communicates to the data processing device (31, 41) a reference to a location at which the properties are stored, instead of his properties.

5. The system as claimed in one of claims 2 to 4, in which, in order to communicate the access rights or a reference to the access rights, a security protocol is used.

6. The system as claimed in one of claims 1 to 5, in which a data processing device (31, 41) can pass on existing properties of a person making access (30) to a following data processing device (31, 41) or to a program for generating a data packet (10, 13).

7. The system as claimed in one of claims 1 to 6, in which the necessary properties for access to a data packet (10, 13) are defined using an access control list (ACL), the list containing:
   - a document field (501) in which the name and location of a data packet (10, 13) are stored,
   - a classification field (502) in which the security class of the data packet (10, 13) is entered,
   - a privileges field (503) in which the necessary properties for access to the data packet (10, 13) are entered, and
   - an instruction field (504) in which additional information, in particular for generating the data packet (10, 13), can be entered.

8. A method for granting accesses to information in a distributed computer system in which a plurality of data processing devices, in particular personal computers (PC) and servers (31, 41) are connected to one another in a network-overlapping fashion by means of transfer protocols (TCP/IP) and random access to data packets (10, 13) is possible, irrespective of the location at which they are stored, data packets (10,
13) being contained which are protected against random access by access rights which, if they are present, enable access to the data packet (10, 13), characterized in that access rights are defined for each data packet, independently of other data packets, by means of allocation to a security class, for which specific properties of a person making access (30) can be defined, in which

- an access wish to a data packet (10, 13) is communicated from a first data processing device (30) to a second data processing device (31, 41),
- the second data processing device (31, 41) checks whether access stipulations are defined and, if not, makes the data packet (10, 13) available and, if there are access stipulations, checks that they have been complied with and, if the relevant properties of the first data processing device (30) are unknown, sends an error message in which disclosure of the properties is requested,

- the first data processing device (30) communicates its properties to the second data processing device (31, 41) which, using an access list (ACL), compares the received properties with the required properties and, if the latter are met, communicates the data packet (10, 13) to the first data processing device (30).

9. The method as claimed in claim 8, in which the identity of the data packet (10, 13) to which access is to be made is veiled, in that the name of the protocol used, (http), is changed to that of a security protocol (websec), this is followed by the indication of the data processing device (31, 41) on which the desired data packet (10, 13) is available, and then the name and location of the data packet (10, 13) are indicated in encoded form at the data processing device (31, 41), and the properties are disclosed by using a security
protocol (GSS-API) in the data processing devices (30, 31, 41).
10. The use of the system and of the method according to claims 1 to 9 in the World Wide Web with data packets (10, 13) in the HTML format, which can be transmitted in accordance with an HTTP protocol.
FIG 3

FIG 4

Web Server

Browser