METHOD AND APPARATUS FOR DYNAMIC AUTHENTICATION IN A DISTRIBUTED NETWORK

Methods and apparatus for providing dynamic authentication in a distributed network (200) are disclosed. As a method, an authentication agent server (212), as part of a host computer (204), provides a merchant lock key (214) and a merchant open key (216) where the merchant open key (216) is stored locally. The merchant lock key (214) in combination with a customer lock key (220) provided by a client computer (202) are used to lock secret data (224) related to a particular customer in a secure data vault (226). The secure data vault (226) is then coupled to transaction data (222) where the host computer (204) unlocks the secure data vault (226) using the stored merchant open key (226) in combination with the customer lock key (220) based upon a decryption algorithm (230) associated with the merchant lock key (214).
METHOD AND APPARATUS FOR DYNAMIC AUTHENTICATION IN A DISTRIBUTED NETWORK

BACKGROUND OF THE INVENTION

1. Field of Invention

The invention relates generally to computer systems. More particularly, methods and apparatus for providing dynamic authentication in a distributed network are disclosed. Specifically, in the realm of e-commerce, an authentication agent provides transaction security between an e-customer and an e-merchant.

2. Description of Relevant Art

Generally speaking, a browser is an application program that provides a way to look at and interact with information on distributed computer networks such as the Internet. In particular, a Web browser is a client program that uses the Hypertext Transfer Protocol (HTTP) to make requests of Web servers throughout the Internet on behalf of the browser user. One of the most recent uses of browsers is in the realm of electronic (e-) commerce in which any number of client side users (i.e. "e-customers") interact in a real time basis with any number of server side processors (i.e., "e-sellers") over the Internet. In order to transact business in a Web-based environment, browsers typically execute Web commerce applications specifically designed to facilitate e-commerce transactions such as requesting quotes, selecting options and assembling components into complex bundles, and placing orders.

In this regard, successful Web commerce applications must be capable of providing both the e-customer and the e-merchant with the confidence that any
transactions are being carried out in a highly secure manner since both parties will be
transferring highly sensitive data “across the wire”. A conventional approach to
providing such transaction security in, for example, a distributed network of
computers is referred to as a Kerberos type security system. Fig. 1 illustrates an
exemplary implementation of a distributed network 100 that utilizes a Kerberos type
security system in order to provide transaction security. The network of computers
100 includes a client computer 102 used by an e-customer 104 to conduct an e-
commerce transaction with an e-merchant 106 coupled to a merchant server 108.
Typically, the e-customer 104 will access a particular e-merchant’s web page by first
issuing a web page request in the form of a URL. In response to the URL, the
merchant server 108 provides an HTTP response in the form of an HTML page
generally consisting of the “expected” interface page used by the e-customer 104 to
enter a particular e-commerce transaction, such as, for example, placing a purchase
order.

However, as is the case with most transactions, the e-customer 104 must
provide secure data such as a credit card number, social security number, and the like
that must be protected from unauthorized access. In order to provide the necessary
protection whenever such sensitive data are transmitted between the client computer
102 and the server computer 108, a Kerberos security system that resides in a
Kerberos server 110 is typically used. It should be noted that in the example shown
the Kerberos server 110 resides in a single Kerberos server but can nonetheless be a
distributed type server computer.

In order, therefore, to access the merchant server 108, a Kerberos "ticket" is
required. Typically, to get this ticket, the e-customer 104 must first request
authentication from an Authentication Server (AS) 114. In response to the
authentication request, if verified, the AS 114 creates what is referred to as a "session key" 116 (which is also an encryption key) that is typically based upon a requestor supplied password and a random value that represents the requested service (i.e., the particular e-merchant 106). It should be noted that the session key 116 is effectively a "ticket-granting ticket".

Once the session key has been generated by the AS 114, the client computer 102 sends the session key 116 to a ticket-granting server (TGS) 118. The TGS 118 then returns the encrypted 120 ticket to the client computer 102 which can now be provided with a particular transaction to the server computer 108 in a secure manner.

At this point, the merchant server either rejects the ticket 120 or accepts it and performs the completed the transaction 120. In most cases, the ticket 120 received from the TGS 116 is time-stamped providing the e-customer 104 the capability of making additional requests using the same ticket within a certain time period (typically, eight hours) without having to be re-authenticated.

Unfortunately, however, the use of a third party (even a trusted third party) such as the Kerberos server introduces additional points where security can be breached. In addition, if, for example, the Kerberos server 110 (or any components therein) are malfunctioning or are otherwise not available, the transaction between the e-merchant and the e-customer can not take place.

Therefore, what is required is a method and apparatus for providing efficient, secure, and fault tolerant dynamic authentication in a distributed network environment.

SUMMARY OF THE INVENTION

In one embodiment of the invention, a computer implemented method of
providing a secure transaction between a client computer and a host computer in a
distributed network is disclosed. A host lock key and a host open key are generated
where the host open key is stored in the host computer. The host lock key is sent to
the client computer where a client lock key is generated. Secret data associated with
the secure transaction is secured using the host lock key and the client lock key
which is then retrieved at the host computer.

In another embodiment, a system for dynamically authenticating a secure
transaction between a client computer and a host computer in a distributed network is
disclosed. The system includes, an authentication agent server coupled to the host
computer provides a host lock key, a host open key, and an authentication agent,
wherein the authentication agent server stores the host open key in the host computer
and sends the authentication agent and the host lock key to the client computer in
response to a client request. A client lock key generator included in the client
computer that provides a client lock key and a secure data vault generator coupled to
the client lock key generator arranged to securely store secret client data based upon
the customer lock key and the host lock. The system also includes a decryptor block
included in the host computer arranged to open the secure data vault using the
merchant open key, wherein the merchant open key only resides in the host
computer.
BRIEF DESCRIPTION OF THE DRAWINGS

The invention, together with further advantages thereof, may best be understood by reference to the following description taken in conjunction with the accompanying drawings.

Fig. 1 shows a conventional Kerberos-type security system implemented in a distributed network of computers.

Fig. 2 illustrates a browser/server system in accordance with an embodiment of the invention is shown.

Fig. 3 illustrates a secured data vault in accordance with an embodiment of the invention.

Fig. 4 is a flowchart detailing a process for conducting a secure transaction over a distributed network of computers in accordance with an embodiment of the invention.

Fig. 5 is a flowchart detailing a process as one implementation of generating an authentication agent of the process detailed in Fig. 4.

Fig. 6 is a flowchart detailing a process as one implementation of locking customer secure data of the process detailed in Fig. 4.

Fig. 7 is a flowchart detailing a process as one implementation of the merchant unlocking the secure data of the process detailed in Fig. 4.

Fig. 8 illustrates a computer system that can be employed to implement the present invention.
DETAILED DESCRIPTION OF THE EMBODIMENTS

In the following description, frameworks and methods of providing dynamic authentication services in a distributed network environment are described. In addition, an apparatus embodied in a computer system that provides dynamic authentication in a distributed network is also described.

It should be noted that although the invention is described in terms of the Internet, any distributed network can be suitably employed to implement any desired embodiment of the invention. It is one of the advantages of the invention that it is well suited for low bandwidth systems capable of executing client side applications. Such low bandwidth systems include, but are not limited to virtual private networks, direct serial connections across telephone lines ("BBS systems"), and LANs and WANs regardless of network protocol.

When implemented in a network using the HTTP protocol, such as the Internet, when an end user (i.e., client) desires to run an application within a browser environment, the end user generates an HTTP request for a resource identified by a URL (universal resource locator). This request is transmitted by way of a distributed network, such as the Internet, to a server computer.

Referring now to Fig. 2, a browser/server system 200 in accordance with an embodiment of the invention is shown. The system 200 includes a client (customer) computer 202 coupled to a server (merchant) computer 204. Typically, the merchant computer 204 is part of a distributed interconnected computer network such as the Internet, but can also be part of a private wide or local area network (WAN/LAN).
utilizing HTTP protocols, sometimes referred to as an intranet. It is one of the advantages of the invention that the interconnected computer network can be any low bandwidth system.

In order to facilitate communication between the various users and/or computers that form the network, the client computer 202 utilizes the graphical user interface resources presented by a Web page (sometimes referred to as an HTML page) 206 resident in a browser 208, most of which are obtained by various HTTP requests. When a user desires to download a particular HTML page from the server 204, the browser 208 generates an HTTP request 210. The URL for the requested page includes information related both to the location of the server computer 204, and to the location within the server computer 204 where the requested page is located. In response to the URL, an authentication agent server 212 residing in, or coupled to, the merchant computer 204 generates a merchant lock key 214 and a merchant open key 216. In the described embodiment, the merchant open key 216 is private since it is only stored in the server 204 and is not at any time “on the wire” (i.e., transmitted over the network). On the other hand, the merchant lock key 214 is attached to an authentication agent server 216 by the authentication agent server 212 as an HTTP response. Typically, the HTTP response takes the form of an HTML page generally consisting of the “expected” interface page and the authentication agent 216, which, preferably, is transparent to the e-customer 104. However, in some situations, the authentication agent 216 can form a visual interface so as to be provide the customer 104 with the capability of providing secure data or other information related to the transaction.
In the described embodiment, the authentication agent 216 takes the form of an embedded client-side application such as for example a Java applet. As a class, Java applets are generally small programs that can be sent along with a Web page to a browser to execute interactive animations, immediate calculations, or other simple tasks using the computing resources of the client without having to send a request back for processing on the server. In Java based systems, therefore, the authentication agent 216 takes the form of a Java based authentication applet 218. Once all required user load time components are available, the authentication agent applet 218 can then proceed in processing user supplied inputs in the particularized context of the received data and any API extension code during what is referred to as a sub-application. It should be noted that in almost all cases, a user session will consist of a series of different sub-applications as, for example, a user navigates the application interface and interacts with the different sections each with its own particularized UI and data.

Once the authentication agent applet 218 is downloaded to the client server 202, the authentication agent applet 218 verifies the customer 104 and generates what is referred to as a customer lock key 220.

Once all transaction data 222 and secret data 224 are made available, the authentication agent applet 218, instantiates what is referred to as a secure data vault 226 an example of which is shown in Fig. 3. The secure data vault 226 includes a secure data field 300 (usually a social security number, a credit card number, etc.) secured by both a customer lock key 302 and a merchant lock key 304. In this way, any unauthorized entity (such as a hacker) must be able to decrypt both the customer lock key and the merchant lock key. However, since the merchant open key is also
required and is never made public, the probability of successfully obtaining the 
secured (or secret) data is very low.

Returning back to Fig. 2, once the authentication agent applet 218 has
instantiated the secure data vault 226, it attaches the appropriate transaction data field
222 (such as color, size, weight), etc to the secure data vault 226 to form a 
transaction request 228. The client computer 2202 then transmits the transaction 
request 228 to the merchant server 204 for further processing.

Once the merchant server 204 receives the transaction request 228, the 
merchant server 204 retrieves the stored merchant open key 216. By using the 
merchant lock key 214 as a pointer to select a particular decryption algorithm 230,
the merchant server 204 opens the secure data vault 226 by using the combination of 
the merchant open key 216 and the customer open key 220 based upon the selected 
algorithm 230. Once the secure data vault 226 is open, the merchant server 204 
confirms the identity of the e-customer 104 and proceeds to retrieve the secret data
224 that is then used to complete the transaction.

As can be readily appreciated, since there is no third party (trusted or not),
potential downtime is reduced. In addition, since the merchant open key is not “on 
the wire” at any time, the possibility of it getting into the hands of an unauthorized 
party is substantially eliminated. In addition, security is further enhanced since it is 
the merchant lock key that points to any one of a number of potential decryption 
algorithms that can be used to open the secured data vault. Since these decryption 
algorithms are known only to the merchant, the likelihood that a hacker can divine 
the correct decryption algorithm is vanishingly small.
Fig. 4 is a flowchart detailing a process 400 for conducting a secure transaction over a distributed network of computers in accordance with an embodiment of the invention. The process 400 begins at 402 by an e-customer downloading a particular e-merchant’s webpage, part of that is an interface that the e-customer uses to initiate a transaction at 404. Once the transaction is received by the merchant at the merchant server, an authentication agent server instantiates an authentication agent at 406 and sends the authentication agent to the customer’s client computer at 408. The authentication agent then verifies the customer at 410, and if verified at 412, the authentication agent locks the customer’s secure data at 414 into a data vault. If, however, the customer is not verified at 412, then processing stops. Once the customer’s secure data is locked at 414, the client computer attaches the appropriate transaction data fields to the secured data vault at 416 and sends the transaction data and secured data vault to the merchant at 418. The merchant server then retrieves the secured data from the secured data vault at 420 and then proceeds to complete the transaction at 422.

Fig. 5 is a flowchart detailing a process 500 as one implementation of generating an authentication agent 406 of the process 400 detailed in Fig. 4. The process 500 begins at 502 by the authentication agent server loading merchant attribute data. Typically, merchant attribute data is used to identify the particular merchant associated with the authentication agent server. Such attribute data can include specific information such as a merchant ID number, location, etc. Once the merchant attribute data has been loaded, the authentication agent server creates a merchant open/lock keyset at 504 based, in part, upon the particular merchant attribute data previously loaded. The merchant open key is then stored at the
merchant site 1t 506 and is not put on the wire therefore substantially eliminating the possibility of unauthorized acquisition of the merchant open key. At 508, the merchant lock key is then attached to the authentication agent in preparation for being sent to the client computer at 408.

Fig. 6 is a flowchart detailing a process 600 as one implementation of locking customer secure data 414 of the process 400 detailed in Fig. 4. The process 600 begins at 602 by the authentication agent generating a customer open key. The secure customer data is then retrieved at 604 while at 606 a secure data vault is instantiated. At 608, the secure data is locked in the secure data vault with both the customer lock key and the merchant lock key in preparation for being attached to particular transaction datafields at 416.

Fig. 7 is a flowchart detailing a process 700 as one implementation of the merchant unlocking the secure data 418 of the process 400 detailed in Fig. 4. The process 700 begins at 702 by the merchant server retrieving the stored (and private) merchant open key. The authentication agent then uses the merchant lock key to select an appropriate decryption algorithm at 704. Once an appropriate decryption algorithm is selected, the authentication agent opens the locked secured data vault at 706 using both the customer open key and the merchant open key based upon the selected decryption algorithm. At 708, the merchant then retrieves the unlocked secured data.

Fig. 8 illustrates a computer system 800 that can be employed to implement the present invention. The computer system 800 or, more specifically, CPUs 802, may be arranged to support a virtual machine, as will be appreciated by those skilled in the art. As is well known in the art, ROM acts to transfer data and instructions
uni-directionally to the CPUs 802, while RAM is used typically to transfer data and instructions in a bi-directional manner. CPUs 802 may generally include any number of processors. Both primary storage devices 804, 806 may include any suitable computer-readable media. A secondary storage medium 808, which is typically a mass memory device, is also coupled bi-directionally to CPUs 802 and provides additional data storage capacity. The mass memory device 808 is a computer-readable medium that may be used to store programs including computer code, data, and the like. Typically, mass memory device 808 is a storage medium such as a hard disk or a tape which generally slower than primary storage devices 804, 806. Mass memory storage device 808 may take the form of a magnetic or paper tape reader or some other well-known device. It will be appreciated that the information retained within the mass memory device 808, may, in appropriate cases, be incorporated in standard fashion as part of RAM 806 as virtual memory. A specific primary storage device 804 such as a CD-ROM may also pass data uni-directionally to the CPUs 802.

CPUs 802 are also coupled to one or more input/output devices 810 that may include, but are not limited to, devices such as video monitors, track balls, mice, keyboards, microphones, touch-sensitive displays, transducer card readers, magnetic or paper tape readers, tablets, styluses, voice or handwriting recognizers, or other well-known input devices such as, of course, other computers. Finally, CPUs 802 optionally may be coupled to a computer or telecommunications network, e.g., an Internet network or an intranet network, using a network connection as shown generally at 812. With such a network connection, it is contemplated that the CPUs 802 might receive information from the network, or might output information to the network in the course of performing the above-described method steps. Such
information, which is often represented as a sequence of instructions to be executed using CPUs 802, may be received from and outputted to the network, for example, in the form of a computer data signal embodied in a carrier wave. The above-described devices and materials will be familiar to those of skill in the computer hardware and software arts.

Although only a few embodiments of the present invention have been described, it should be understood that the present invention may be embodied in many other specific forms without departing from the spirit or the scope of the present invention.

Although the inventive methods of representing complex products and selling processes purely in data models are particularly suitable for implementation in browser environments, the methods may generally be applied in any suitable low bandwidth or high bandwidth system. Such low bandwidth systems include, but are not limited to virtual private networks, direct serial connections across telephone lines ("BBS systems"), and LANs and WANs regardless of network protocol.

While the present invention has been described as being used with a computer system that has an associated web browser and web server, it should be appreciated that the present invention may generally be implemented on any suitable computer system. Therefore, the present examples are to be considered as illustrative and not restrictive, and the invention is not to be limited to the details given herein, but may be modified within the scope of the appended claims along with their full scope of equivalents.

*What is claimed is:*
In the claims:

1. A method for dynamically authenticating a secure transaction between a client computer and a host computer in a distributed network, comprising:
   generating a host lock key and a host open key;
   storing the host open key in the host computer;
   sending the host lock key to the client computer;
   generating a client lock key;
   securing secret data associated with the secure transaction using the host lock key and the client lock key; and
   retrieving the secret data.

2. A method as recited in claim 1, wherein the securing further comprises:
   instantiating a secure data vault; and
   encapsulating the secret data, the host lock key, and the client lock key within the secure data vault.

3. A method as recited in claim 2, wherein the retrieving further comprises:
   retrieving the stored host open key;
   selecting a decryption algorithm based upon the host lock key; and
   decrypting the secured secret data using the selected decryption algorithm
based upon the customer lock key and the host open key.

4. A method as recited in claim 3, wherein the distributed network is an HTTP protocol type network.

5. A system for dynamically authenticating a secure transaction between a client computer and a host computer in a distributed network, comprising:

   in response to a client request,

   an authentication agent server coupled to the host computer provides a host lock key, a host open key, and an authentication agent, wherein the authentication agent server stores the host open key in the host computer and sends the authentication agent and the host lock key to the client computer;

   a client lock key generator included in the client computer that provides a client lock key;

   a secure data vault generator coupled to the client lock key generator arranged to securely store secret client data based upon the customer lock key and the host lock; and

   a decryptor block included in the host computer arranged to open the secure data vault using the merchant open key, wherein the merchant open key only resides in the host computer.

6. A system as recited in claim 5, wherein the host computer is a merchant
computer and wherein the client computer is a customer computer.

7. A system as recited in claim 6, wherein the host computer is one of a plurality of interconnected computers.
Fig. 1

Prior Art

1. Client computer
   102

2. Merchant server
   108

3. Ticket granting server
   116

4. Authentication server
   114

5. Completed transaction

6. Transaction
   112

7. Encrypted ticket
   110

8. Session key
   120

9. Authentication request
   100

10. Completion of transaction

11. Transaction request
   120

12. Prior Art

13. Diagramatic representation of a transaction flow with authentication and ticket granting steps.
vendor lock key 304

customer lock key 302

secure data 300

secure data vault 226

Fig. 3
start

customer downloads merchant's webpage

customer initiates transaction

instantiate authentication agent

merchant sends authentication agent to customer

authentication agent verifies customer

customer verified?

no

yes

authentication agent locks customer secure data

attach data vault to associated transaction data

send to merchant

merchant unlocks secure data

merchant completes transaction

stop

Fig. 4
500

load merchant attribute data

generate merchant open/lock keyset

store merchant open key at merchant site

attach merchant lock key to authentication agent

408

Fig. 5
generate customer open key

get customer secure data

instantiate secure data vault

lock secure data vault with customer lock key and merchant lock key

stop

Fig. 6
start

merchant retrieves merchant open key

authentication agent uses merchant lock key to select appropriate decryption algorithm

authentication agent appropriately opens the locked secure data vault using the customer lock key in combination with the merchant open key based upon the selected decryption algorithm

merchant retrieves secure data

stop

Fig. 7
Fig. 8