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

Canadian Intellectual Property Office
An agency of Industry Canada

CA 2446304 C 2012/03/20

(11)(21) 2 446 304

(12) BREVET CANADIEN
CANADIAN PATENT

(13) C

(86) Date de dépôt PCT/PCT Filing Date: 2002/04/30
(87) Date publication PCT/PCT Publication Date: 2002/11/14
(45) Date de délivrance/Issue Date: 2012/03/20
(85) Entrée phase nationale/National Entry: 2003/11/03
(86) N° demande PCT/PCT Application No.: US 2002/013521
(87) N° publication PCT/PCT Publication No.: 2002/091662
(30) Priorité/Priority: 2001/05/01 (US60/287,858)

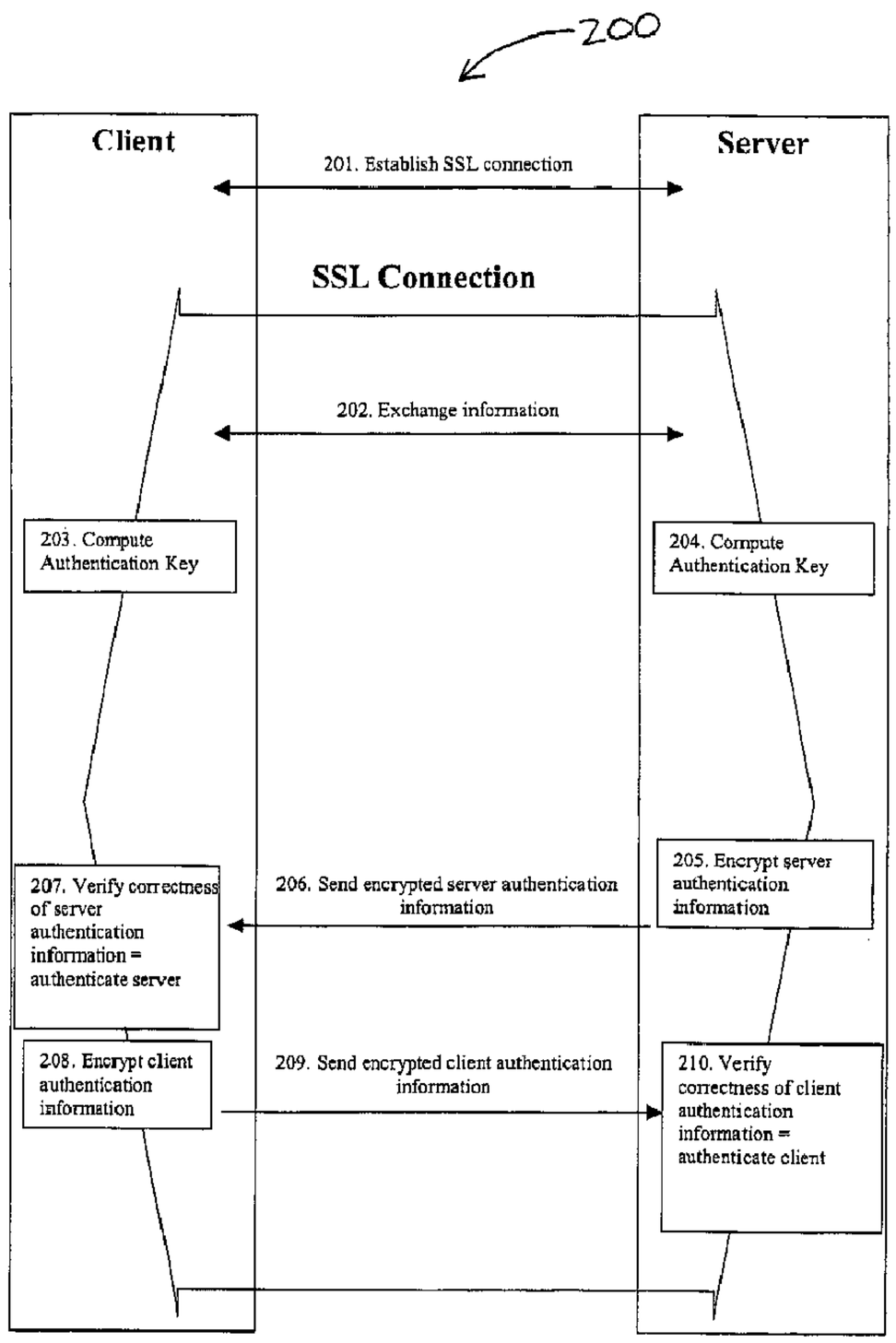
(51) Cl.Int./Int.Cl. *H04L 9/00* (2006.01),
H04L 29/06 (2006.01)

(72) Inventeur/Inventor:
COULIER, FRANK, BE

(73) Propriétaire/Owner:
VASCO DATA SECURITY, INC., US

(74) Agent: GOWLING LAFLEUR HENDERSON LLP

(54) Titre : UTILISATION ET PRODUCTION D'UNE CLE DE SESSION DANS UNE CONNEXION SSL
(54) Title: USE AND GENERATION OF A SESSION KEY IN A SECURE SOCKET LAYER CONNECTION



(57) Abrégé/Abstract:
The invention describes a method (200) and system for verifying the link between a public key and a server's identity as claimed in the server's certificate without relying on the trustworthiness of the root certificate of the server's certificate chain. The system

CA 2446304 C 2012/03/20

(11)(21) 2 446 304

(13) C

(57) Abrégé(suite)/Abstract(continued):
establishes a secure socket layer type connection (201) between a client and a server, wherein the server transmits information including the server's public key to the client while establishing the connection. Next, a first information is sent from the client to the server (202). The client and the server create an identical authentication key using a shared secret known to the server and the client (203 and 204). Next, the server transmits a first encrypted message to the client (206), wherein the first encrypted message includes the server's public key encrypted with the authentication key. Then, the client decrypts the first encrypted message and verifies the correctness (207) of that message including comparing the public key included in the decrypted first encrypted message to the public key transmitted during the set-up of the secure socket layer type connection to authenticate the client and to establish the trustworthiness of the server's public key and thereby the entire SSL connection. The client then transmits a second encrypted message to the server (209), wherein the second encrypted message is the first information encrypted with the authentication key. Finally, the server then decrypts the second encrypted message and verifies the correctness of the decrypted second encrypted message to authenticate the client (210).

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

CORRECTED VERSION

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
14 November 2002 (14.11.2002)

PCT

(10) International Publication Number
WO 02/091662 A1

(51) International Patent Classification[7]: H04L 9/00

(21) International Application Number: PCT/US02/13521

(22) International Filing Date: 30 April 2002 (30.04.2002)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/287,858 1 May 2001 (01.05.2001) US

(71) Applicant *(for all designated States except US)*: **VASCO DATA SECURITY, INC.** [US/US]; Suite 210, 1901 S. Meyers Road, Oakbrook Terrace, IL 60181 (US).

(72) Inventor: **COULIER, Frank**; Sint-Hubertusstraat 12, 1850 Grimbergen (BE).

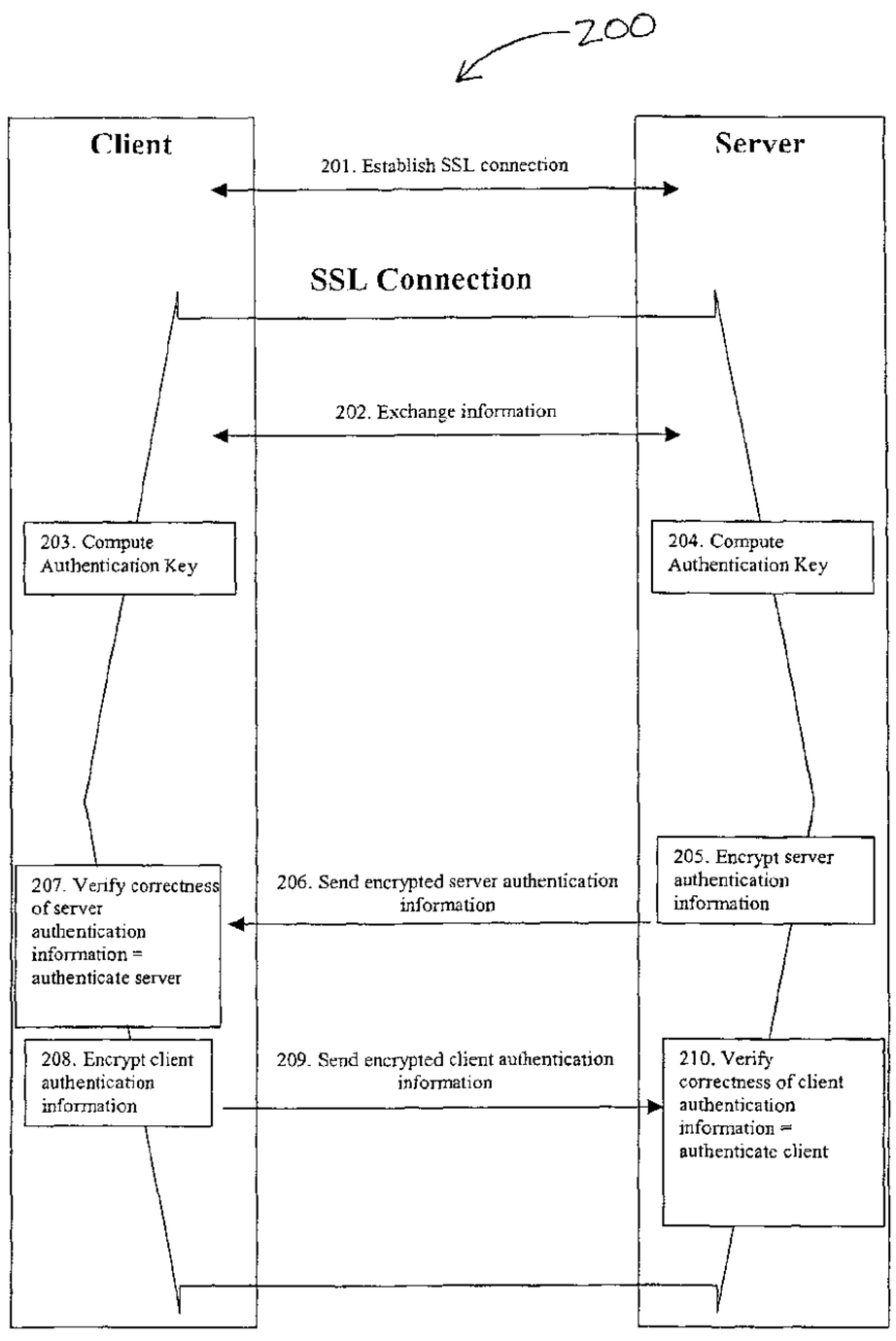
(74) Agents: **MEYER, Sheldon, R.** et al.; Fliesler Dubb Meyer and Lovejoy LLP, Fourth Embarcadero Center, Suite 400, San Francisco, CA 94111-4156 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZM, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent

[Continued on next page]

(54) Title: USE AND GENERATION OF A SESSION KEY IN A SECURE SOCKET LAYER CONNECTION



(57) Abstract: The invention describes a method (200) and system for verifying the link between a public key and a server's identity as claimed in the server's certificate without relying on the trustworthiness of the root certificate of the server's certificate chain. The system establishes a secure socket layer type connection (201) between a client and a server, wherein the server transmits information including the server's public key to the client while establishing the connection. Next, a first information is sent from the client to the server (202). The client and the server create an identical authentication key using a shared secret known to the server and the client (203 and 204). Next, the server transmits a first encrypted message to the client (206), wherein the first encrypted message includes the server's public key encrypted with the authentication key. Then, the client decrypts the first encrypted message and verifies the correctness (207) of that message including comparing the public key included in the decrypted first encrypted message to the public key transmitted during the set-up of the secure socket layer type connection to authenticate the client and to establish the trustworthiness of the server's public key and thereby the entire SSL connection. The client then transmits a second encrypted message to the server (209), wherein the second encrypted message is the first information encrypted with the authentication key. Finally, the server then decrypts the second encrypted message and verifies the correctness of the decrypted second encrypted message to authenticate the client (210).

WO 02/091662 A1

WO 02/091662 A1

(BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report*

(48) Date of publication of this corrected version:

14 August 2003

(15) Information about Correction:

see PCT Gazette No. 33/2003 of 14 August 2003, Section II

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

USE AND GENERATION OF A SESSION KEY IN A SECURE SOCKET LAYER CONNECTION

Cross Reference to Related Applications

The present application is related to the following United States Patents and Patent Applications, which patent/applications are assigned to the owner of the present invention:

United States Patent Application publishing under Publication No. 2001/0054148, entitled "Field Programmable Smart Card Terminal and Token Device", filed on February 20, 2001;

United States Patent No. 4,599,489, entitled "Solid State Key for Controlling Access to Computer Software", filed on February 22, 1984, issued on 15 July 8, 1986;

United States Patent No. 4,609,777, entitled "Solid State Key for Controlling Access to Computer Software", filed on December 23, 1985, issued on September 2, 1986; and

United States Patent No. 4,819,267, entitled "Solid State Key for Controlling Access to Computer Systems and to Computer Software and/or for Secure Communications, filed on June 9, 1987, issued on April 4, 1989.

Field of the Invention

The invention relates generally to establishing secure connections over a network, and more particularly to establishing a secure socket layer type connection over a digital public network.

Background

Extended development and public acceptance have made electronic commerce and distributed transactions over public networks widespread. As shown in FIG. 1, many of these transactions involve a client device 110, such as a personal computer, accessing and communicating with a server 120. The connection between the client 110 and the server 120 may be used to exchange confidential information or enable the server to provide restricted or secured access. As a result, the need for security in transactions between a client and a server occurring over a digital connection on a network has become widespread as well. Therefore, in many cases, the connection between the client 110 and the server 120 is an encrypted and mutually authenticated connection 130.

The prior art includes several methods that attempt to resolve this need for security that requires mutual authentication and encrypted communication between the server and client. One method utilizes a symmetrical encryption algorithm based on a shared secret. In general, a shared secret is known to both the client and the server. However, a shared secret may sometimes also be possessed by a trusted third party. In general, a shared secret is not known to or easily determined by the public at large. The shared secret is used to derive an encryption key. The encryption key is then used to encrypt communication between the server and the client using a symmetrical encryption algorithm. The symmetrical encryption

algorithm achieves confidentiality because the encrypted messages can't be read without knowing the shared secret. The method achieves authentication in that only a participant to the connection who possesses the shared secret may properly encrypt and decrypt messages with another participant. Thus, if a participant can read and generate a message or connection request that is encrypted, the party must possess the secret and is deemed authenticated. For practical reasons, the shared secret often originates from the client side by a physical person operating the client side (e.g. by typing in a password). For ergonomic reasons, the size of the shared secret (e.g. the number of characters in a password) is therefore quite limited. As a result, the cryptographic strength of the symmetric encryption key derived from that shared secret is also rather limited.

Another method used by previous systems involves an asymmetric cryptographic algorithm and public key infrastructure (PKI) certificates for the server and the client. This method does not utilize a shared secret known to both ends of a connection before setting up the connection. Rather, a client and a server exchange certificates when they establish a connection. The client and server then authenticate one another by validating each other's certificates. Next, dynamically generated random data is exchanged by the client and server using the public keys certified by each other's certificates. Both the client and the server use this dynamically generated data to compute separate but identical symmetric encryption key. This symmetric encryption key is then used to encrypt further communication between the client and the server and thereby provide confidentiality. In practice, it may be impractical to provide clients with certificates. As a result, sometimes only the server has a certificate to be validated. In this case, the server certificate is sufficient to authenticate the server and to establish the symmetric encryption

key that provides confidentiality. Another method must be used to authenticate the client. For example, the client may provide proof to the server that it possesses a shared secret known only to the server and the client (e.g. the client might send the server a password). As mentioned above, the server end in a secure socket layer connection is authenticated by validating or verifying the server's certificate. The server certificate may be considered a digital signature generated by a certification authority (CA) linking the server's public key to the server's identity. The public key of the CA may then be certified by another higher-level CA. In theory, an entire hierarchy or chain of certificates may require verification. Regardless of the number of levels in a server's certificate chain, the client must have the public key of the highest level or root CA to be able to validate the entire chain. Thus, the security of this method depends on the trustworthiness of the root's public key.

In practice, many client systems and network browsers have an extensive list of certificate roots that the client "trusts." It is usually not difficult to convince a user to add additional certificate roots to their list of trusted roots. Thus, a user may unknowingly add a tainted or false certificate root by an illegitimate CA. This false root may have been used by a dishonest entity to generate a certificate for an illegitimate server that poses as a legitimate server. In this way, a dishonest entity may lead a client to make a connection with the illegitimate server and unknowingly provide sensitive information such as the user's password to the legitimate server.

For purposes of illustration, one may consider a bank operating a website that allows bank customers to consult their accounts and perform financial transactions online. Only the rightful owner of an account should have access to their account. Therefore, the bank might require its customers to authenticate

themselves by entering a password when accessing the site. To protect the confidentiality of information exchanged between the client and the bank's website such as a user password, clients connect to the bank's website using the SSL protocol. A entity who wants to compromise the account of a legitimate bank customer could mount a man-in-the-middle. To do so, the entity could set up a website that mimics the legitimate bank website. The entity generates a certificate for the fraudulent website with a bogus certification authority. The entity tricks the legitimate customer into adding the root certificate of the bogus CA to his list of trusted roots. If the legitimate customer now connects to the fraudulent website, the legitimate customer will think it's the legitimate bank website and enter a password. The entity has now obtained a valid password and can access the real bank website posing as the legitimate user to e.g. transfer money from the legitimate user's account to the entity's account. Thus, relying on a technique that requires a client to trust the public key of the root CA of a server's SSL certificate to validate the authenticity of that server may reduce the security level of a secure socket layer connection and jeopardize server security.

What is therefore needed is a way to increase the security of a secure socket layer connection. A client system should be able to verify the validity of a server's certificate or validate the link between the server's public key and its claimed identity without the client system having to trust the root CA or some intermediate CA of the server's certificate chain.

Summary of the Invention

The invention satisfies the shortcomings of the prior art by providing a method and system for verifying the link between a public key and a server's identity as claimed in the server's certificate without relying on the trustworthiness

of the root certificate of the server's certificate chain. A secure socket layer type connection is established between a client and a server. While establishing the connection, the server transmits the server's certificate information to the client. Next, information is sent from the client to the server. Then, the client and the server independently create identical authentication keys by utilizing a shared secret known to the server and the client. Next, the server transmits a first encrypted message to the client over the secure socket layer connection, wherein the first encrypted message is encrypted with the server authentication key and wherein the correctness of the contents of the first message can be verified by the client. Then, the client decrypts the first encrypted message and validates the decrypted first message thus authenticating the server. The client then transmits a second encrypted message to the server, wherein the second encrypted message is encrypted with the client authentication key and wherein the correctness of the contents of the second message can be verified by the server. Finally, the server decrypts the second encrypted message and validates the decrypted second message thus authenticating the client. In one embodiment, the first encrypted message sent by the server to the client contains the server's certificate or public key that was used during the set-up of the secure socket layer type connection. In one embodiment, the shared secret used to create the authentication key may be the response of a strong authentication token. In this embodiment, the strong authentication token may be a challenge-response, event, internal counter-based or time-based token, or any combination of these three strong authentication token variants.

Brief Description of the Drawings

Figure 1 is a illustration of a server client connection requiring a secured mutually authenticating connection in accordance with the prior art;

Figure 2 is an illustration of a method for mutual authentication of an SSL connection in accordance with one embodiment of the present invention;

Figure 3 is an illustration of a method for mutual authentication of an SSL connection in accordance with one embodiment of the present invention;

Figure 4 is an illustration of a method for mutual authentication of an SSL connection, in accordance with one embodiment of the present invention; and

Figure 5 is an illustration of a method for an authentication key generation process in accordance with one embodiment of the present invention.

Detailed Description

The present invention solves the problems of prior systems by using a secret to verify the link between a public key and a server's identity as claimed in the server's certificate without relying on the trustworthiness of the root certificate of the server's certificate chain. The invention takes the advantages of an SSL type connection without relying on the root certificate needed to verify the server's identity. This is achieved using a symmetric encryption authentication key that the client and server generate independently from each other. The authentication key is used by the server to encrypt the server's public key or certificate that was used to set-up the SSL connection. Thus, the present invention achieves mutual authentication and encryption without depending on the trustworthiness of the server certificate.

One method for establishing a secure connection between a client and server involves using public key infrastructure (PKI) certificates for the server and the client. The first step in this method includes a client and a server exchanging certificates as they establish a connection. The client and server may validate each other's certificates to authenticate one another. However, in some instances, the client may not provide a certificate. Next, dynamically generated random data is exchanged by the client and server using an asymmetric cryptographic algorithm and the public keys certified by the client and server (each by their own cert) certificates. Both the client and the server use this dynamically generated data to compute identical symmetric encryption keys. This symmetric encryption key uses a symmetric encryption algorithm to encrypt all further communication during the connection between the client and the server. A secure socket layer (SSL) connection as used herein shall be meant to include all types of connections that are

established using the certificate-based method discussed above. Examples of such connections include transaction layer security (TLS) connections, a wireless TLS (WTLS) connection, and IP secure (IPSEC) connections.

In one embodiment of the present invention, two types of encryption are occurring. The first level of encryption involves the encryption provided by the SSL connection. Messages sent over an SSL connection are encrypted. The second level of encryption involves the encryption performed by an authentication key. The authentication key is used to encrypt and decrypt messages to ensure that a client or server is who it represents itself to be. Unless otherwise specified, references to encryption herein are intended to pertain to the encryption performed by an authentication key.

A method 200 for establishing a server connection between the server and the client in accordance with one embodiment of the present invention is shown in FIG. 2. In method 200, a client and server establish an SSL connection in step 201. This SSL connection may be established with or without using a client certificate. While the SSL connection is being established, the client receives the server's certificate. All further communication between client and server occurs through this SSL connection. In one embodiment, the result of the verification of the server' certificate chain may be ignored.

Next, the authentication key to be used for mutually authenticating the server and the client is created. First, the client and server exchange some information in step 202. Then, the client and server each use the same shared secret to independently compute a symmetric encryption key in steps 203 and 204. This symmetric encryption key is the authentication key. A shared secret as used herein is generally defined as a secret possessed by the client and the server. In one

embodiment, a third party such as a certificate authority may possess the secret. However, a secret known to a third party is still considered secret from the public at large. In one embodiment, the shared secret is not transferred between the server and the client.

Next, the client and server are both authenticated. First, the server encrypts server authentication information in step 205. The server constructs this server authentication information so the client may verify its correctness. The server then encrypts the server authentication information with the authentication key generated in the previous step and transmits the encrypted server authentication information to the client in step 206. By successfully decrypting this encrypted server authentication information and verifying its correctness in step 207, the client authenticates the server end of the SSL connection and establishes the trustworthiness of the SSL connection. Next, the client encrypts client authentication information in step 208 with the same authentication key generated in step 203 and sends the encrypted client authentication information to the server in step 209. The server then decrypts the encrypted client authentication information received from the client and verifies its correctness in step 210. By successfully decrypting and verifying the encrypted client authentication information, the server authenticates the client. The client authentication information can include the username and/or part of the information exchanged between client and server earlier on. The authentication key can be used to encrypt any further communication between the server and the client in addition to the encryption already provided by the SSL connection.

FIG. 3 illustrates a method for establishing a secure connection between a client and a server in another embodiment of the present invention. In FIG. 3, the

server authentication information includes the server's public key that is certified previously by the SSL server certificate. Thus, verifying the correctness of the server authentication information by the client includes verifying that the public key included in the server authentication information is the same as the public key certified by the SSL server certificate. The advantage of this embodiment is that the server's SSL certificate has been authenticated in a way which does not rely on using the public key of the certification authority that signed the SSL server certificate. The SSL connection can now be considered to be a secure, mutually authenticated connection that provides integrity and confidentiality, without the client side having to rely on the trustworthiness of the root certificate that holds the key of the certification authority that signed the SSL server certificate and has been stored on the client.

Method 300 of FIG. 3 begins when the server and client establish a secured socket layer type connection in step 301. This step is similar to the step of 201 in FIG. 2 except that the server sends its certificate to the client. The client then stores the certificate in step 302. Once the connection is established, all communication between the client and server will occur through this connection.

Next, the server and client compute authentication keys. First, the client and server exchange information in step 303. Then, the client and server each use the same shared secret to independently compute a symmetric encryption key in steps 304 and 305. This symmetric encryption key is the authentication key. Next, the server encrypts the server certificate using the symmetric encryption key in step 306 and transmits the encrypted server certificate to the client in step 307. The client compares the stored server certificate from step 302 to the encrypted server certificate sent in step 308 to authenticate the server. The client then encrypts

client authentication information in step 309. The encrypted client authentication information is then sent to the server from the client in step 310. The server then decrypts the client authentication information to authenticate the client in step 311.

In another embodiment of the present invention, the entire SSL server certificate may be included in the server authentication information in step 301. Verifying the correctness of the server authentication information by the client in step 308 then includes verifying that the certificate included in the server authentication information is the same as the SSL server certificate received during the set-up of the SSL connection in step 301.

In one embodiment of the present invention, the shared secret used by the client and the server to generate the authentication key is the output of a strong authentication token (SAT). In a preferred embodiment of the invention, the output of the token is the dynamic password generated by the token. In this embodiment, the output of a SAT is provided to both the client and the server. The client and server generate an authentication key using the SAT output. The SAT is not transmitted between the client and the server before generating an authentication key.

In one embodiment of the present invention, the SAT may be a challenge-response token. In this embodiment, the output of the SAT may rely on a challenge to generate a response. If a strong authentication token requires a challenge from the server to generate a dynamic password, the server side of the SSL connection sends this challenge to the client side of the SSL connection. The client and server will both independently generate a separate but identical response to the challenge. The SAT response is then considered the shared secret between the server and client and is used to generate the authentication key.

In another embodiment, the token may be a time-based SAT. This embodiment of a SAT may introduce synchronization issues to generating an authentication key. In one embodiment where strong authentication tokens are based on an internal real-time clock and/or an event such as an internal counter, the server may allow a limited set of possible SAT responses rather than one single SAT response. One reason for allowing multiple SAT responses is to overcome synchronization issues. If a strong authentication token is time based, the server may not be capable of knowing exactly which value of the time the SAT has used to generate its response. This is because the internal clock of the SAT might slightly drift with respect to the server's clock and because the user interaction with the SAT always takes some varying unpredictable time. Similarly, if a strong authentication token is event based, the server is often not capable of knowing exactly which value of the event counter the SAT has used to generate its response. This is because it may be possible that the SAT has generated a response that incremented a counter but didn't reach the server. Thus, the server would not be aware that the event counter has been increased.

To overcome the synchronization problems in case of a time or event based SAT, the server may allow a certain number of possible responses rather than a single response. In one embodiment of the present invention, the number of possible responses is relatively small. In the case of a time based SAT, the server will use a window of time around the current time of its real-time clock and will allow multiple responses that can be generated on the basis of time values in that window. Similarly, in the case of an event based token the server will allow multiple responses that can be generated on the basis of the current value of the server's counter and a limited number of consecutive counter values.

In one embodiment, to account for more than one valid response, the server should allow more than one authentication key. However, since the server first sends data to the client encrypted with the authentication key, the server should already know which of the possible authentication keys to use because the client only has one authentication key. To enable the server to figure out which key of the set of possible authentication keys is the one being used by the client, a synchronization phase may be implemented.

FIG. 4 illustrates a method 400 for using a SAT to generate an authentication key. The method 400 begins when the client and server set up an SSL connection in operation 401, with or without client authentication. While the connection is established, the server sends the server certificate to the client. The client obtains and stores the server's certificate in operation 402. Once the connection is established, all further communication between client and server happen through this SSL connection.

Next, the authentication key is established as represented by steps 403 and 404. In operation 403, the client sends information to the server. In one embodiment of the invention, the information is the username or other information client authentication information identifying the user at the client end. Next, the server and the client exchange some data that will be used in calculating the authentication key in step 404. In one embodiment of the present invention, the data exchanged are random seeds created independently by the client and server. Next, if the SAT is a challenge-response SAT, the server sends a challenge to the client in operation 405. Then, in operation 406 and 407 the client and server independently generate an identical symmetric encryption key using the response of the SAT and the data exchanged in the operation 404. This symmetric

encryption key is the authentication key. Next, if the SAT is a time-based or event-based SAT, the server sends a synchronization challenge to the client in step 408. Then, the client encrypts this synchronization challenge with the authentication key it has computed at step 409. The client then sends the encrypted synchronization challenge back to the server in step 410.

The server then determines what authentication key to use in step 411. In one embodiment, the server computes several acceptable SAT responses using information that may include the SAT challenge, the value of its clock and the current value of the event counter. In one embodiment, the server computes all the acceptable SAT responses. Then, based on the acceptable SAT responses computed and the data exchanged, the server computes a set of candidate authentication keys in operation.

In one embodiment, the server may decrypt the encrypted synchronization challenge it received from the client with each of the authentication keys that are allowed on the basis of its clock or event counter in step 410. The key from the set of allowed keys that successfully decrypts the encrypted synchronization challenge is the key that will be used by the server for authentication purposes.

Next, the client and server are authenticated. First, in operation 412 the server encrypts the certificate that it has sent to the client during the set-up of the SSL connection with its authentication key. The server sends this encrypted certificate to the client in operation 413. Then, in operation 414, the client decrypts the certificate received from the server and compares it to the certificate received from the client in the set-up of the SSL connection. Upon successfully decrypting of the encrypted server certificate and verifying it is the same certificate that the client has obtained during the set-up of the SSL connection, the client authenticates

the server end of the SSL connection and establishes the trustworthiness of the SSL connection. Next, in step 415 the client encrypts the username or other client authentication information sent to the server in step 403 with the same authentication key. The client sends this encrypted information to the server in operations 416. The server then decrypts the information and compares it to the information received from the client in step 417. Upon successfully decrypting the encrypted username or other information previously received, the server authenticates the client. The SSL connection can now be considered to be a secure, mutually authenticated connection that provides integrity and confidentiality, without relying on the trustworthiness of the root certificate that has been stored on the client. The authentication key can be used to encrypt any further communication between the server and the client on top of the encryption already offered by the SSL connection.

Figure 5 is a flow chart of a process 500 showing a key generation algorithm based on the dynamic password of a challenge-response token in accordance with one embodiment of the present invention. First, in operations 501 and 502, the client and server exchange random seeds. In one embodiment, the seeds are 256 bits in length are processed by both the client and the server in a similar fashion. The seeds are then combined in an operation 503. In one embodiment, the random seeds are combined using an XOR function. If the token is a challenge-response token the server will also provide the client with a challenge for the token. Next, the initial generation vector is created. First, the user name which is known to the both the server and the client at the time of the session key generation is repeated to yield a string of 128 bits in operation 504. A challenge issued by a server and known to both the server and the client is then

repeated to yield a string of 128 bits in operation 506. Next, the first 128 bits of the combined random seeds and the expanded user name and the expanded challenge are then combined to yield the Initial Generation Vector in operation 507. In one embodiment of the present invention, the elements are all combined using an XOR function.

Next, the generation key is created. First, the response to the challenge is repeated to yield a string of 128 bits in operation 509. The response is independently generated by both the server and the client. Then, the last 128 bits of the combined seeds and the expanded response are combined to yield the Generating Key. In one embodiment, the elements are combined using an XOR function.

Then, in operation 511, using the generating key as a symmetrical block cypher key, the symmetrical block cipher algorithm is applied X number of times on the Initial Generation Vector with the output of round N serving as the input to round N+1, where X is a fixed number known in advance by both client and server such that the described calculation takes something between 10 and 100 milliseconds on a typical client. Finally, the resulting 128 bits of this calculation are the Authentication Key in operation 512. In one embodiment the symmetrical block cipher algorithm is the 3DES algorithm. In another embodiment the symmetrical block cipher algorithm is the AES algorithm.

The present invention solves the problems of prior systems by using a secret to verify the link between a public key and a server's identity as claimed in the server's certificate without relying on the trustworthiness of the root certificate of the server's certificate chain. The invention takes the advantages of an SSL type connection without relying on the root certificate needed to verify the server's identity. This is achieved using a symmetric encryption authentication key that the

client and server generate independently from each other. The authentication key is used by the server to encrypt the server's public key or certificate that was used to set-up the SSL connection. Thus, the present invention achieves mutual authentication and encryption without depending on the trustworthiness of the server certificate.

Other features, aspects and objects of the invention can be obtained from a review of the figures and the claims. It is to be understood that other embodiments of the invention can be developed and fall within the spirit and scope of the invention and claims.

The foregoing description of preferred embodiments of the present invention has been provided for the purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise forms disclosed. Obviously, many modifications and variations will be apparent to the practitioner skilled in the art. The embodiments were chosen and described in order to best explain the principles of the invention and its practical application, thereby enabling others skilled in the art to understand the invention for various embodiments and with various modifications that are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the following claims and their equivalence.

What is claimed is:

1. A method for establishing a secure connection and authenticating a server in connections formed with PKI procedures, wherein a server public key, obtained from the server by a client, is used with asymmetric cryptography to establish a symmetric session key for encryption of communications with symmetric cryptography during the connection, said method offering an alternative for authenticating the server public key, and comprising:

generating a server authentication key by the server,

transmitting said server public key by the server to the client in clear text form;

generating a client authentication key by the client, the server authentication key and the client authentication key being identical to each other as both are generated using a common secret;

generating server authentication information from data derived from the server public key and processed with a symmetric cryptographic algorithm and the server authentication key,

sending said server authentication information to the client,

verifying the server authentication information at the client in order to authenticate the server public key, said verifying using the client authentication key to determine that the server authentication information is based on said server authentication key and the server public key used in establishing the secure connection and received from the server.

2. A method as recited in claim 1 wherein:

the server authentication key is used for encrypting the server authentication information with a symmetric encryption algorithm; and which further includes:

decrypting, at the client, the received server authentication information with the client authentication key and a symmetric decryption algorithm, and

wherein said verifying implements verifying the correctness of the server authentication information at the client by comparing the decrypted server

authentication information with the server public key used in establishing the secure connection and received from the server.

3. The method of claim 2 wherein the server authentication information includes a server certificate.

4. The method of claim 2 wherein the secure connection includes an SSL connection.

5. The method of claim 2 wherein the secure connection includes an WTLS connection.

6. The method of claim 2 wherein the secure connection includes an IPSEC connection.

7. The method of claim 2 wherein the secure connection includes a TLS connection.

8. The method of claim 2 wherein the secret is generated by a strong authentication token.

9. The method as recited in claim 2 which further includes:

sending client information to the server to authenticate the client, the client information encrypted by the client using the client authentication key and decrypted by the server using the server authentication key, the correctness of the client information verified by the server.

10. The method of claim 2 wherein the server authentication information includes the server public key.

11. The method of claim 2 wherein the server authentication information includes data derived from the server public key.

12. The method of claim 8 wherein the strong authentication token is a challenge response token, wherein generating an authentication key by both the server and the client includes:

sending a challenge from a server to the strong authentication token;

generating a first strong authentication token response to the challenge at the client;

generating a second strong authentication token response to the challenge at the server, the first response identical to the second response when the secret is common to the client and server;

deriving a client authentication key by the client from the first strong authentication token response;

deriving a server authentication key by the server from the second strong authentication token response.

13. The method of claim 8 wherein the strong authentication token is a time-based token, wherein generating an authentication key includes:

generating a strong authentication token time-based response by the strong authentication token at the client;

deriving a client authentication key from the response by the client;

sending a synchronization challenge from the server to the client;

encrypting the synchronization challenge with the client authentication key by the client;

sending the encrypted synchronization challenge from the client to the server; and

generating a server authentication key by the server that corresponds to the client authentication key used by the client.

14. A method for authenticating a server public key and establishing a secure connection between a client and a server, the connection formed with PKI procedures and including a symmetric key established using the server public key with asymmetric cryptography, said symmetric key used to encrypt

communications during the connection with symmetric cryptography, the method offering an alternative for authenticating the server public key, and comprising:

transmitting a server certificate from the server to the client, the server certificate including server public key information;

generating separate authentication keys by the server and the client, the keys being identical as generated using a common secret, said generating separate authentication keys including:

sending user authentication information from the client to the server;

exchanging dynamic information between the client and the server;

generating a secret by the client and the server from the response of a strong

authentication token; and

generating authentication keys at client and server using the user authentication information, the dynamic information, and the secret; thereafter

generating server authentication information at the server from data derived from the server public key and processed with a symmetric cryptographic algorithm and the server authentication key;

sending said server authentication information to the client;

receiving the server authentication information at the client, and

verifying the server authentication information at the client in order to authenticate the server public key, said verifying using the client authentication key to determine that the server authentication information is based on said server authentication key and the server public key information from the server certificate.

15. A method as recited in claim 14 wherein:

said generating server authentication information comprises encrypting data related to the server public key using the server authentication key and a symmetric encryption algorithm;

said verifying includes decrypting the received server authentication information at the client using a symmetric decryption algorithm and the client authentication key; and

determining the correctness of the server authentication information by comparing the decrypted server authentication information with the server public key information from the server certificate.

16. The method of claim 15 wherein the user authentication information includes a user identification information.

17. The method of claim 15 wherein the secure connection is an SSL connection.

18. The method of claim 15 wherein the dynamic information includes random information.

19. The method of claim 15 wherein the strong authentication token includes a challenge-response strong authentication token, wherein the secret is derived from the response of the challenge response token.

20. The method of claim 15 wherein the strong authentication token includes a time based token.

21. The method of claim 15 wherein the strong authentication token includes an event based token.

22. The method as recited in claim 15 further including:

sending encrypted user authentication information to the server, the user authentication information encrypted by the client using the authentication key generated by the client; and

receiving and decrypting the user authentication information by the server, the server decrypting the user authentication information using the authentication key created by the server, the correctness of the user authentication information verified by the server.

23. The method of claim 15 wherein the server certificate transmitted to the client includes the server public key.

24. The method of claim 15 wherein the server authentication information includes the server public key.

25. The method of claim 15 wherein the server authentication information includes data derived from the server public key.

26. The method of claim 11 wherein the data derived from the server public key includes a hash of the server public key.

27. The method of claim 25 wherein the data derived from the server public key includes a hash of the server public key.

28. A method for establishing a secure connection using PKI procedures and authenticating a server public key, wherein the server public key, obtained from the server by a client, is used with asymmetric cryptography to establish a symmetric session key for encryption of communications with symmetric cryptography during the connection and offering an alternative for authenticating the server public key where a server authentication key is generated by the server and used to create server authentication information for transmission to the client, said method comprising

generating a client authentication key by the client, the server authentication key and the client authentication key being identical to each other as both are generated using a common secret;

receiving the server public key in clear text form from the server;

receiving the server authentication information at the client to authenticate the server public key, the server authentication information including data derived from the server's public key and processed with a server authentication key and with a symmetric cryptographic algorithm; and

verifying the server authentication information at the client in order to authenticate the server public key, said verifying using the client authentication key to determine that the server authentication information is based on the server authentication key and the server public key used in establishing the secure connection and received from the server.

29. A method as recited in claim 28 wherein the server authentication key is used to encrypt the server authentication information for transmission to the client, wherein said method further includes:

decrypting, at the client, the received server authentication information with the client authentication key and a symmetric decryption algorithm to obtain decrypted data, and

wherein said verifying includes verifying the correctness of the server authentication information at the client in order to authenticate the server public key by comparing the decrypted data with the server public key used in establishing the secure connection and received from the server.

30. The method of claim 29 wherein the server authentication information includes a server certificate.

31. The method of claim 29 wherein the secure connection includes an SSL connection.

32. The method of claim 29 wherein the secure connection includes an WTLS connection.

33. The method of claim 29 wherein the secure connection includes an IPSEC connection.

34. The method of claim 29 wherein the secure connection includes a TLS connection.

35. The method of claim 29 wherein the secret is generated by a client strong authentication token.

36. The method as recited in claim 29 which further includes:

sending client information to the server to authenticate the client, the client information encrypted by the client using the client authentication key.

37. The method of claim 29 wherein the server authentication information includes the server public key.

38. The method of claim 35 wherein the strong authentication token is a challenge response token, wherein generating an authentication key by the client includes:

receiving a challenge from the server for the strong authentication token;

generating a strong authentication token response to the challenge at the client; and

deriving a client authentication key by the client from the strong authentication token response.

39. The method of claim 35 wherein the strong authentication token is a time-based token, wherein generating an authentication key includes:

generating a strong authentication token time-based response by the strong authentication token at the client;

deriving a client authentication key from the response by the client;

receiving, at the client, a synchronization challenge from the server;

encrypting the synchronization challenge with the client authentication key by the client; and

sending the encrypted synchronization challenge from the client to the server for generating a server authentication key by the server that corresponds to the client authentication key used by the client.

40. A method for authenticating a server public key and establishing a secure connection between a client and a server, the connection formed with PKI procedures and including a symmetric key, established using the server public key with asymmetric cryptography, to encrypt communications during the connection with symmetric cryptography, the method offering an alternative for authenticating the server public key, and comprising:

receiving a server certificate at the client, the server certificate including server public key information in clear text form;

generating an authentication key by the client corresponding to an authentication key generated at the server, the keys being identical as generated using a common secret, said generating an authentication key by the client including:

sending user authentication information from the client to the server;

exchanging dynamic information between the client and server,

generating a secret by the client from a response of a client strong authentication token corresponding to a secret generated by the server; and

the client generating said authentication key, corresponding to an authentication key generated at the server, using the user authentication information, the dynamic information, and the secret; thereafter

receiving server authentication information at the client, the server authentication information including data derived from the server public key, processed using the authentication key generated by the server and a symmetric cryptographic algorithm; and

verifying the server authentication information at the client by using the client authentication key to determine that the server authentication information is based on the server authentication key and the server public key information received in clear text form.

41. A method as recited in claim 40 wherein;

said receiving server authentication information comprises receiving server authentication information encrypted using the authentication key generated by the server and a symmetric encryption algorithm, and wherein

said verifying further comprises;

decrypting the server authentication information by the client, the client decrypting the server authentication information using a symmetric decryption algorithm and the authentication key created by the client, and

comparing the decrypted server authentication information at the client with server public key information received in clear text form.

42. The method of claim 40 wherein the user authentication information includes a user identification information.

43. The method of claim 40 wherein the secure connection is an SSL connection.

44. The method of claim 40 wherein the dynamic information includes random information.

45. The method of claim 40 wherein the strong authentication token includes a challenge-response strong authentication token, wherein the secret is derived from the response of the challenge response token.

46. The method of claim 40 wherein the strong authentication token includes a time based token.

47. The method of claim 40 wherein the strong authentication token includes an event based token.

48. The method as recited in claim 40 further including:

sending encrypted user authentication information to the server, the user authentication information encrypted by the client using the authentication key generated by the client.

49. The method of claim 40 wherein the server certificate received by the client includes the server public key.

50. The method of claim 40 wherein the server authentication information includes the server public key.

51. The method of claim 29 wherein the data derived from the server public key includes a hash of the server public key.

52. The method of claim 41 wherein the data derived from the server public key includes a hash of the server public key.

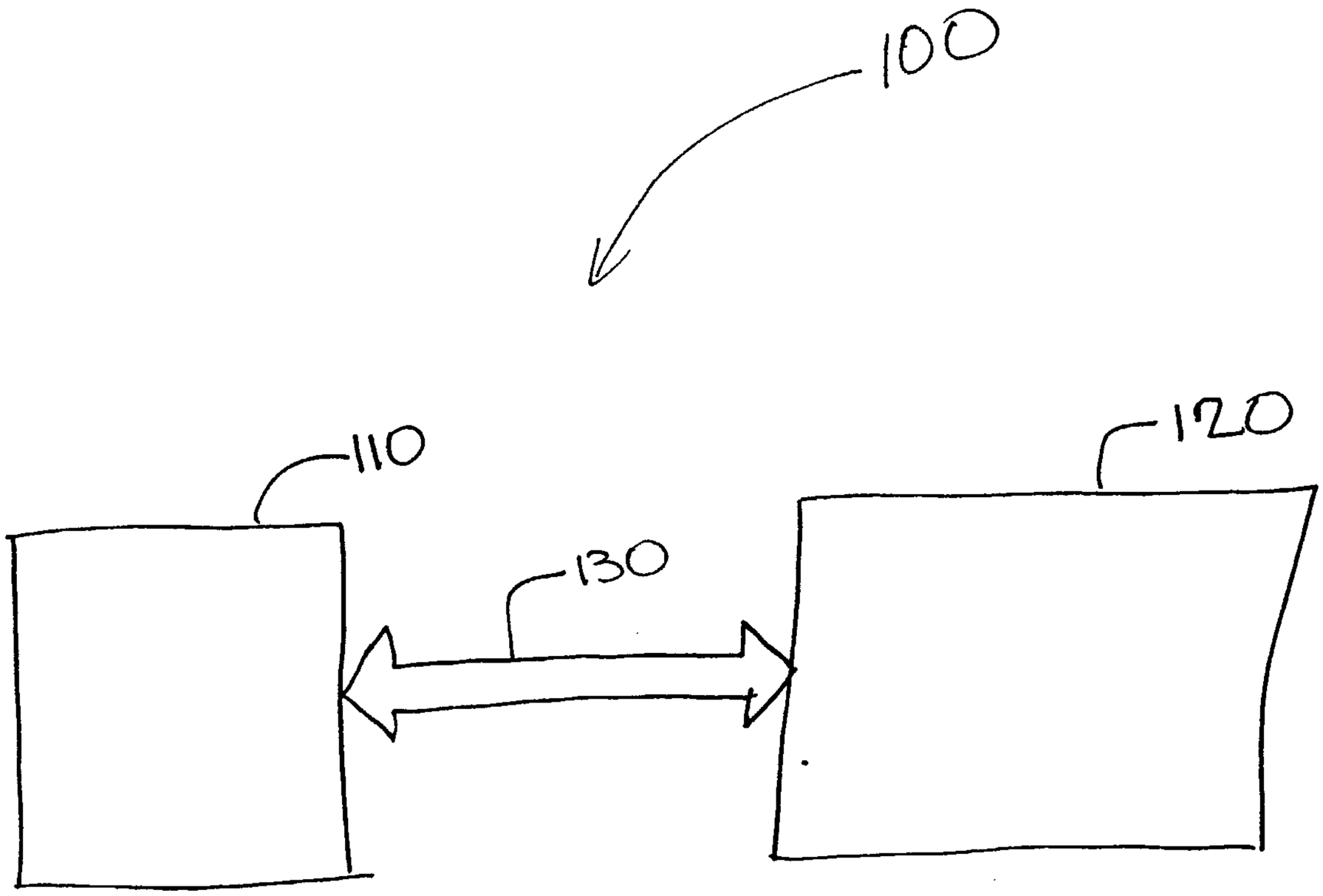


FIG. 1

(PRIOR ART)

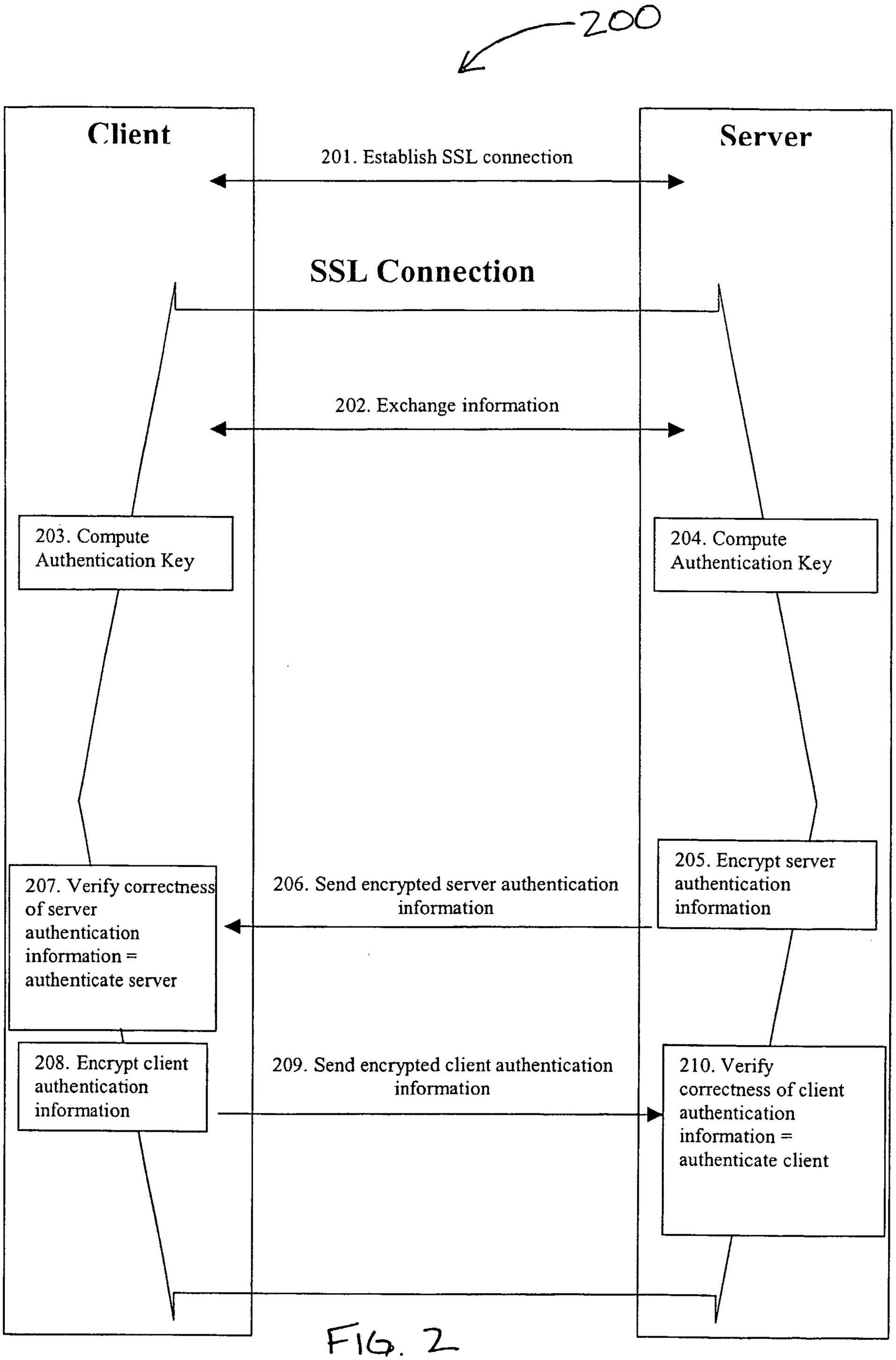

200
Client
Server
201. Establish SSL connection
SSL Connection
202. Exchange information
203. Compute Authentication Key
204. Compute Authentication Key
205. Encrypt server authentication information
206. Send encrypted server authentication information
207. Verify correctness of server authentication information = authenticate server
208. Encrypt client authentication information
209. Send encrypted client authentication information
210. Verify correctness of client authentication information = authenticate client

FIG. 2

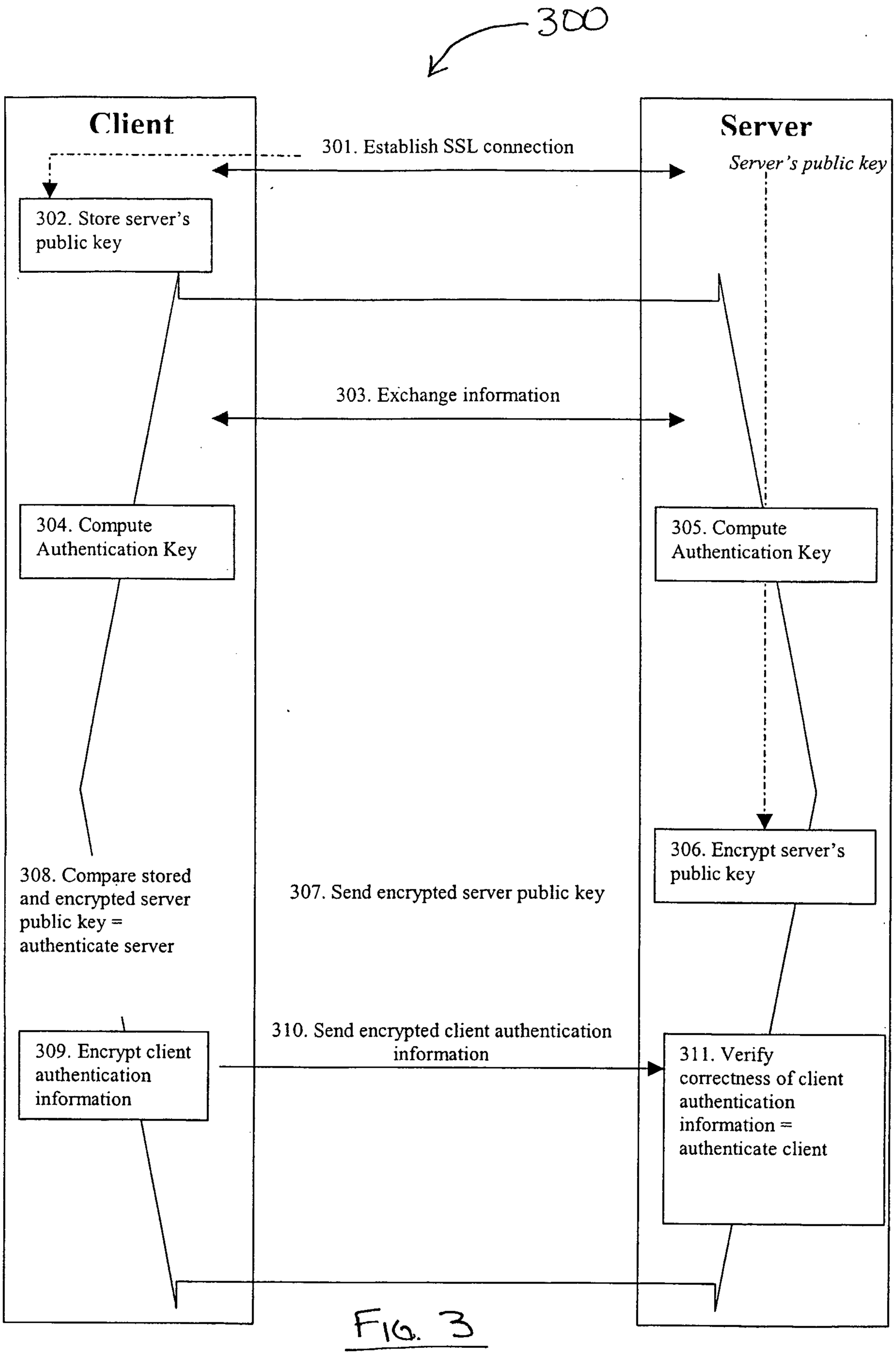

300
Client
Server
301. Establish SSL connection
Server's public key
302. Store server's public key
303. Exchange information
304. Compute Authentication Key
305. Compute Authentication Key
306. Encrypt server's public key
307. Send encrypted server public key
308. Compare stored and encrypted server public key = authenticate server
309. Encrypt client authentication information
310. Send encrypted client authentication information
311. Verify correctness of client authentication information = authenticate client

FIG. 3

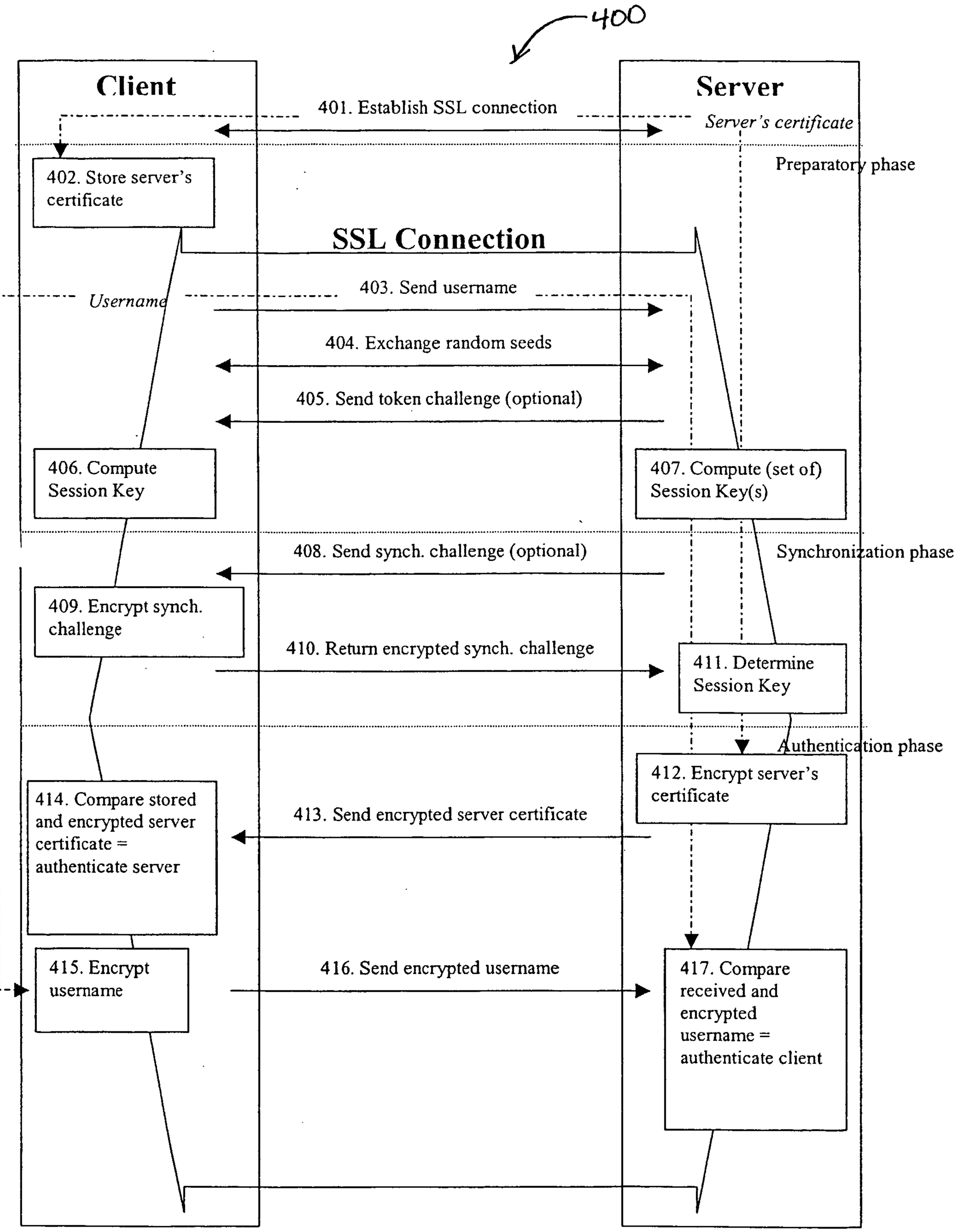

400
Client
Server
401. Establish SSL connection
Server's certificate
Preparatory phase
402. Store server's certificate
SSL Connection
403. Send username
Username
404. Exchange random seeds
405. Send token challenge (optional)
406. Compute Session Key
407. Compute (set of) Session Key(s)
408. Send synch. challenge (optional)
Synchronization phase
409. Encrypt synch. challenge
410. Return encrypted synch. challenge
411. Determine Session Key
Authentication phase
412. Encrypt server's certificate
413. Send encrypted server certificate
414. Compare stored and encrypted server certificate = authenticate server
415. Encrypt username
416. Send encrypted username
417. Compare received and encrypted username = authenticate client

FIG. 4

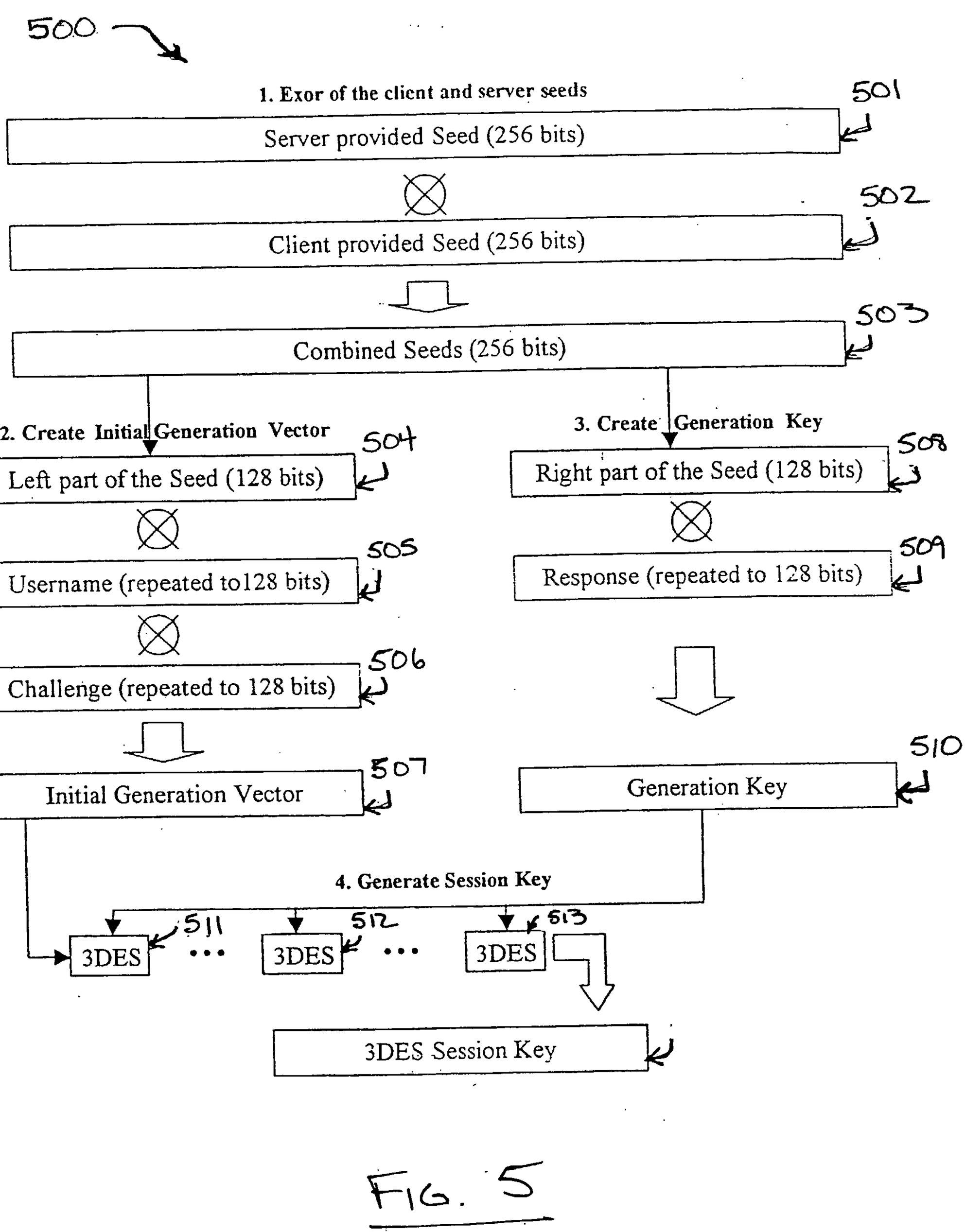
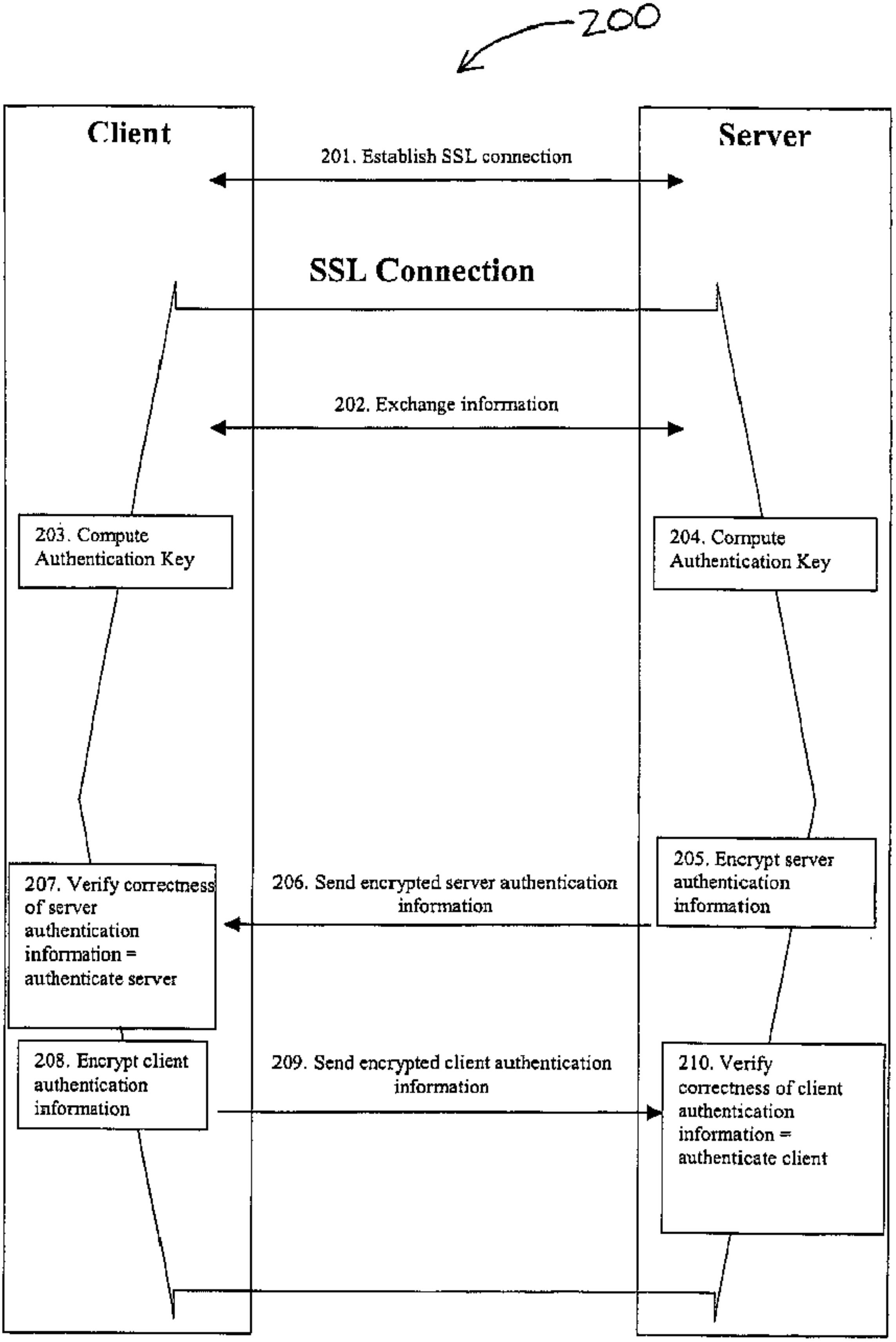

500
1. Exor of the client and server seeds
Server provided Seed (256 bits)
501
Client provided Seed (256 bits)
502
Combined Seeds (256 bits)
503
2. Create Initial Generation Vector
Left part of the Seed (128 bits)
504
Username (repeated to128 bits)
505
Challenge (repeated to 128 bits)
506
3. Create Generation Key
Right part of the Seed (128 bits)
508
Response (repeated to 128 bits)
509
Initial Generation Vector
507
Generation Key
510
4. Generate Session Key
3DES
511
3DES
512
3DES
513
3DES Session Key

FIG. 5


200
Client
Server
201. Establish SSL connection
SSL Connection
202. Exchange information
203. Compute Authentication Key
204. Compute Authentication Key
205. Encrypt server authentication information
206. Send encrypted server authentication information
207. Verify correctness of server authentication information = authenticate server
208. Encrypt client authentication information
209. Send encrypted client authentication information
210. Verify correctness of client authentication information = authenticate client