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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
13 August 2009 (13.08.2009)

PCT

(10) International Publication Number
WO 2009/100259 A2

(51) International Patent Classification:
H04L 9/32 (2006.01) *H04L 9/00* (2006.01)

(21) International Application Number:
PCT/US2009/033281

(22) International Filing Date:
5 February 2009 (05.02.2009)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/026,465 5 February 2008 (05.02.2008) US
61/026,728 6 February 2008 (06.02.2008) US

(71) Applicant *(for all designated States except US)*: **ICONTROL, INC.** [US/US]; 3235 Kifer Road, Suite 260, Santa Clara, CA 95051 (US).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: **ERICKSON, David, Lee** [US/US]; 660 South Fifteenth Street, San Jose, CA 95112 (US).

(74) Agent: **PENILLA, Albert, S.**; Martine Penilla & Gencarella, LLP, 701 Lakeway Drive, Suite 200, Sunnyvale, CA 94085 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, SE, SI, SK, TR),

[Continued on next page]

(54) Title: METHODS AND SYSTEMS FOR SHORTENED HASH AUTHENTICATION AND IMPLICIT SESSION KEY AGREEMENT

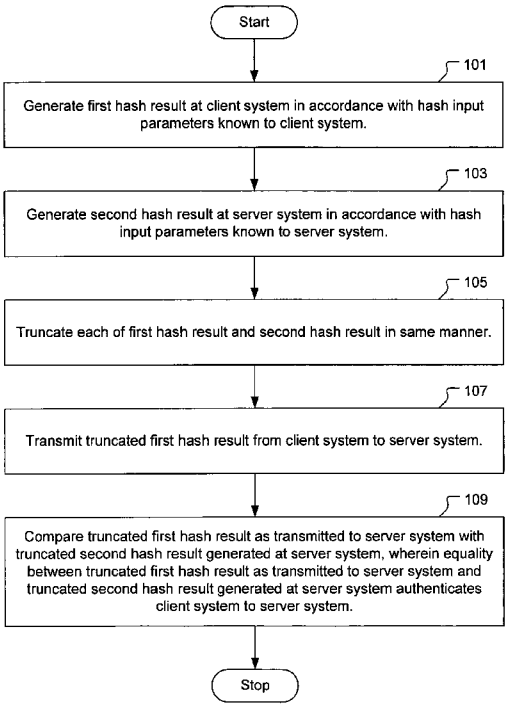

Fig. 1

(57) Abstract: Secure communication between a client and a server is often required in modern telecommunication systems. Communication security involves identifying and authentication of a client to a server. In general networking systems, complex identification and authentication methods may be deployed. However, such complex security methods typically require substantial computing and power resources on both the client side and server side, as well as substantial communication bandwidth to convey identification and authentication credentials, which may be lengthy. In situations where one or both of the client and server systems are limited on computing and/or power resources, or where a limited communication bandwidth exists between the client and server systems, it is desirable to have a strong identification and authentication security capability that does not compromise system or network operability.

WO 2009/100259 A2

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

Published:

— *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*

Methods and Systems for Shortened Hash Authentication and Implicit Session Key Agreement

by Inventor

David Lee Erickson

BACKGROUND

[0001] Secure communication between a client and a server is often required in modern telecommunication systems. Communication security involves identifying and authentication of a client to a server. In general networking systems, complex identification and authentication methods may be deployed. However, such complex security methods typically require substantial computing and power resources on both the client side and server side, as well as substantial communication bandwidth to convey identification and authentication credentials, which may be lengthy. In situations where one or both of the client and server systems are limited on computing and/or power resources, or where a limited communication bandwidth exists between the client and server systems, it is desirable to have a strong identification and authentication security capability that does not compromise system or network operability.

SUMMARY

[0002] In one embodiment, a method is disclosed for performing a shortened hash authentication. The method includes an operation for generating a first hash result at a client system in accordance with hash input parameters known to the client system. A second hash result is also generated at a server system in accordance with hash input parameters known to the server system. Each of the first hash result and the second hash result is truncated in a same manner. The method further includes transmitting the truncated first hash result from the client system to the server system. The truncated first hash result as transmitted to the server system is compared with the truncated second hash result generated at the server system. Equality between the truncated first hash result as transmitted to the server system and the truncated second hash result generated at the server system authenticates the client system to the server system.

[0003] In another embodiment, a method is disclosed for performing a shortened hash authentication. In the method, a secret code is stored on each of a client system and a server system. The secret code is identical on each of the client and server systems. The method also includes an operation for generating a nonce code specific to a given authentication process. The nonce code is provided to both the client system and the server system. The secret code and the nonce code are combined to generate a local code word at each of the client system and server system. Each generated local code word is processed through a hashing algorithm to generate a local hash result at each of the client system and the server system. Also, at each of the client system and server system, each local hash result is truncated to obtain a client-generated authentication code and a server-generated authentication code. The client-generated authentication code is transmitted to the server system. The method further includes an operation for comparing the client-generated

authentication code to the server-generated authentication code at the server system. Equality between the client-generated authentication code and the server-generated authentication code authenticates the client system to the server system.

[0004] In another embodiment, a system for performing a shortened hash authentication is disclosed. The system includes a client defined to generate a first hash result in accordance with hash input parameters known to the client. The client is also defined to truncate the first hash result to obtain a client-generated authentication code. The system also includes a server defined to generate a second hash result in accordance with hash input parameters known to the server. The server is also defined to truncate the second hash result to obtain a server-generated authentication code. The client is defined to transmit the client-generated authentication code to the server. The server is defined to receive the client-generated authentication code and compare the client-generated authentication code to the server-generated authentication code. Equality between the client-generated and server-generated authentication codes authenticates the client to the server.

[0005] In another embodiment, a method is disclosed for implicit session key agreement. The method includes an operation for storing a secret code on each of a client system and a server system. The secret code is identical on each of the client and server systems. The method also includes an operation for generating a nonce code specific to a given authentication process. The nonce code is provided to both the client system and the server system. The method further includes an operation for combining the secret code and the nonce code to generate a local code word at each of the client system and server system. Each generated local code word is processed through a hashing algorithm to generate a local hash result at each of the client system and the server system. The method further includes an operation for using the local hash result at each of the client system and the

server system as an implicit session key for data communication between the client and server systems.

[0006] Other aspects and advantages of the invention will become more apparent from the following detailed description, taken in conjunction with the accompanying drawings, illustrating by way of example the present invention.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 shows a flowchart of a method for performing a shortened hash authentication, in accordance with one embodiment of the present invention;

Figure 2 shows a shortened hash authentication process over a simplex channel, in accordance with one embodiment of the present invention;

Figure 3 shows a comparison between the shortened hash authentication method over simplex and duplex channels, in accordance with one embodiment of the present invention;

Figure 4 shows an example shortened hash authentication using concatenation of the secret code and nonce code, in accordance with one embodiment of the present invention;

Figure 5 shows an example shortened hash authentication using algebraic addition of the secret code and nonce code, in accordance with one embodiment of the present invention;

Figure 6 shows a shortened hash authentication process over a simplex channel with dynamic process parameters, in accordance with one embodiment of the present invention;

Figure 7 shows an example shortened hash authentication using algebraic addition of the secret code and nonce code with dynamic parameter settings, in accordance with one embodiment of the present invention;

Figure 8 shows a flowchart of a method for performing a shortened hash authentication, in accordance with one embodiment of the present invention;

Figure 9 shows a method for performing implicit session key agreement with a simplex channel, in accordance with one embodiment of the present invention;

Figure 10 shows a comparison of the simplex and duplex embodiments in the implicit session key agreement methods, in accordance with one embodiment of the present invention;

Figure 11 shows an example implicit session key agreement implementation, in accordance with one embodiment of the present invention;

Figure 12 shows another example implicit session key agreement implementation, in accordance with one embodiment of the present invention;

Figure 13 shows an implicit session key agreement process with dynamic process parameters, in accordance with one embodiment of the present invention;

Figure 14 shows an example in which the method of combination and hash algorithm are dynamically specified, in accordance with one embodiment of the present invention; and

Figure 15 shows a flowchart of a method for implicit session key agreement, in accordance with one embodiment of the present invention.

DETAILED DESCRIPTION

[0007] In the following description, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, to one skilled in the art that the present invention may be practiced without some or all of these specific details. In other instances, well known process operations have not been described in detail in order not to unnecessarily obscure the present invention.

[0008] Shortened hash authentication methods are disclosed herein for strong authentication security while reducing the communication burden between two entities, such as between a client and a server. The shortened hash authentication methods are well-suited for application in high-security, low-bandwidth communication channels in distributed networks. Moreover, the shortened hash authentication methods disclosed herein satisfy identification and authentication requirements of low-bandwidth networks, which are utilized in low-cost and/or low-power applications.

[0009] Shortened Hash Authentication

[0010] A shortened hash authentication method is disclosed herein for strong authentication of a client to a server at a reduced communication channel bandwidth. The shortened hash authentication method strongly secures distributed communication networks while minimizing an impact to overall communication channel bandwidth. Figure 1 shows a flowchart of a method for performing a shortened hash authentication, in accordance with one embodiment of the present invention. The method includes an operation 101 for generating a first hash result at a client system in accordance with hash input parameters known to the client system. An operation 103 is also performed to generate a second hash result at a server system in accordance with hash input parameters

known to the server system. In an operation 105, each of the first hash result and the second hash result is truncated in a same manner.

[0011] The method also includes an operation 107 for transmitting the truncated first hash result from the client system to the server system. An operation 109 is then performed to compare the truncated first hash result as transmitted to the server system with the truncated second hash result generated at the server system. If the comparison yields equality between the truncated first hash result as transmitted to the server system and the truncated second hash result generated at the server system, the client system is authenticated to the server system. Otherwise, the client system is not authenticated to the server system.

[0012] In one embodiment, the hash input parameters known to the client system include a client code word representing a combination of a secret code and a nonce code as known to the client system. Also, in this embodiment, the hash input parameters known to the server system include a server code word representing a combination of a secret code and a nonce code known to the server system. In one embodiment, the secret code known to the client system and the secret code known to the server system are identical and are stored on the client and server systems prior to authentication. In various embodiments, the secret code can take different forms, so long as the secret code is identical at both the client system and server system. In one exemplary embodiment, the secret code is an ASCII text phrase.

[0013] The nonce code is uniquely generated for each authentication process. It should be understood that because the nonce code varies from one authentication process to another, combination of the secret code and nonce code to generate the code word serves to prevent a successful malevolent decoding of the code word in one authentication process from being used in subsequent authentication processes. In one embodiment, the nonce code is derived from a monotonic data source, such that the nonce code will not repeat itself. In

one embodiment, the nonce code is a current time stamp. Because the code word includes a private portion, i.e., the shared secret code, the nonce code may be transmitted in public view between the client and server systems. The same nonce code needs to be used by each of the client and server systems in generating their respective code words.

[0014] In one embodiment, the nonce code is generated at the client system and is transmitted from the client system to the server system. This embodiment may be utilized when a simplex communication channel exists between the client system and the server system. In such a simplex communication channel, uni-directional communication is possible from the client system to the server system. In another embodiment, the nonce code can be generated at the server system and can be transmitted from the server system to the client system. This embodiment may be utilized when a duplex communication channel exists between the client system and the server system. In such a duplex communication channel, bi-directional communication is possible between the client and server systems. It should be appreciated that greater computing resources of the server system, as compared to the client system, can be advantageously utilized to generate the nonce code when the duplex communication channel is present. In another embodiment, the nonce code is generated independently at each of the client system and server system. In this embodiment, the nonce code is generated based on identical information that is independently known to both the client system and server system, such that the nonce code known to the client system is identical to the nonce code known to the server system.

[0015] As previously mentioned, the client and server code words represent a combination of the secret code and the nonce code as known to the client and server systems, respectively. In various embodiments, the secret code and the nonce code can be combined in different ways to generate the client and server code words, so long as the client and server system use the identical method for combining the secret code and nonce code. For

example, in one embodiment, at each of the client and server systems, the secret code and nonce code can be combined by concatenation to form the code word. In this embodiment, the concatenation may be in any order, so long as the same concatenation order is utilized at both the client and server systems.

[0016] In another embodiment, at each of the client and server systems, the secret code and nonce code can be combined in an algebraic manner to form the code word. For example, a bit-wise addition of the binary forms of the secret code and nonce code may be performed to generate the code word. In other embodiments, the algebraic combination of the secret code and nonce code may include bit-wise subtraction, bit-wise multiplication, bit-wise division, or essentially any other bit-wise computation, so long as the same algebraic combination method is utilized at both the client and server systems. The method for combining the secret code and nonce code may be known *a priori* and encoded in the design in various embodiments, or may be dynamically selected by information shared in common between the client and server systems. This shared information may be encoded within the message, implicit in the connection address or port number or in some other data, or may be implicitly or explicitly available in some other manner to the client and server systems.

[0017] It should be understood that the identical hashing algorithm is used to generate the first hash result at the client system and the second hash result at the server system. In one embodiment, both the client and server systems are pre-loaded with information specifying the required hashing algorithm and its associated parameters, such as the hash seed to be used. In another embodiment, the hashing algorithm and its associated parameters may be dynamically determined by each of the client and server systems based on information shared in common between the client and server systems. For example, the hashing algorithm and its associated parameters may be encoded within the message, implicit in the

connection address or port number or in some other data, or may be implicitly or explicitly available in some other manner to the client and server systems.

[0018] The method of Figure 1 can further include an operation for specifying a truncation size for both the first hash result and the second hash result. The truncation size is specified based on a bandwidth available for transmission of the truncated first hash result and a probability of malevolent decoding of the truncated first hash result. In one embodiment, the truncation size is pre-defined in both the client and server systems. In another embodiment, the truncation size may be known *a priori* and encoded in the design, or may be dynamically selected by information shared in common between the client and server systems. In yet another embodiment, the truncation size may be encoded within the message, implicit in the connection address or port number or in some other data, or may be implicitly or explicitly available in some other manner to the client and server systems. It should be understood that the truncation size is identical for both the client and server systems.

[0019] Figure 2 shows a shortened hash authentication process over a simplex channel, in accordance with one embodiment of the present invention. The client 201 combines the client's secret code (S_C) with a monotonic nonce code (T_C) to generate a client code word ($\sum_C$), which is submitted to a selected hash algorithm ($Hash_C$) to produce a long authentication code (H_C). This long authentication code (H_C) is truncated to a shorter client-generated authentication code (H_C') for efficient transmission from the client to the server. Truncation of the long authentication code (H_C) is mathematically equivalent to the operation $H_C \bmod 2^{L_C}$, where L_C is the specified truncation size. The client 201 transmits the nonce code (T_C) and the client-generated authentication code (H_C') to the server 203.

[0020] The server 203 combines the server's secret code (S_S), where ($S_S = S_C$), with the shared monotonic nonce code (T_C) to generate a server code word ($\sum_S$), which is submitted

to a selected hash algorithm ($Hash_S$), where ($Hash_S = Hash_C$), to produce a long authentication code (H_S). This long authentication code (H_S) is truncated to a shorter server-generated authentication code (H_S'). Truncation of the long authentication code (H_S) is mathematically equivalent to the operation $H_S \bmod 2^{L_S}$, where ($L_C = L_S$). The server 203 compares the client-generated authentication code (H_C') to the server-generated authentication code (H_S') and declares a successful authentication of the client 201 to the server 203 if $H_S' = H_C'$.

[0021] The hashing algorithm is chosen such that when subjected to a brute force attack method, the expected number of attempts required to discover the mutual secret is $2^{|H_C|-1}$, where $|H_C|$ is the length of the long authentication code H_C. Some hashing algorithms have less cryptographic strength, and research on a published hashing algorithm generally reduces the cryptographic strength over time as new methods of cryptographic attack are discovered. The shortened hash authentication method is defined such that the probability of successfully guessing the truncated authentication code is at most $2^{1-|H_C'|}$, where $|H_C'|$ is the length of the truncated authentication code word H_C'. Since the probability of falsely authenticating two messages is limited to $2^{1-2|H_C'|}$, the probability of falsely authenticating (n) messages is limited to $2^{1-n|H_C'|}$ or $2^{1-|H_C|}$, whichever is less. Therefore, the shortened hash authentication method provides conventional cryptographic strength in the long term while offering a significant reduction in required authentication code transmission bandwidth.

[0022] Based on the foregoing, it should be understood that the authentication code (H) originates from a strong hash algorithm and is truncated (H') to minimize the bandwidth burden on low-bandwidth channels. Although the probability of anticipating the correct authentication code is approximately $2^{|H'|}$ (where $|H'|$ is the length of H'), the ability to determine the secret code remains approximately $2^{|H|}$ (assuming brute force attack, and

hashing algorithm strength variability), where |H| >> |H'|. Unless the mutual secret can be discovered, the probability of reliably compromising authentication is $2^{|H|}$, and the average expected compromise rate is $2^{|H'|}$, which can be adjusted to the specific security requirements of the application.

[0023] Figure 3 shows a comparison between the shortened hash authentication method over simplex and duplex channels, in accordance with one embodiment of the present invention. In the simplex embodiment, the client produces the nonce code (T) and provides it to the server. In the duplex embodiment, the server produces and shares the nonce code (T) as needed when the client is ready to transmit. The simplex embodiment may be used where simplex channels or excessive channel latency is present. The duplex embodiment may be used where the client cannot reliably produce a monotonic nonce code (T) or when the server can produce the monotonic nonce code (T) more conveniently or for less cost.

[0024] Figure 4 shows an example shortened hash authentication using concatenation of the secret code (S) and nonce code (T), in accordance with one embodiment of the present invention. Each symbol is drawn from the set of {0..255} and may be ASCII or other coding for the convenience of the implementation. In this illustration, the code word (ST) is appended with 0's until the total length is 128 bits long, then the code word (ST) is submitted to the SHA-1 hashing algorithm to produce a 16-byte value for the hash code (H), of which 4 bytes are retained for the truncated authentication code (H').

[0025] Figure 5 shows an example shortened hash authentication using algebraic addition of the secret code (S) and nonce code (T), in accordance with one embodiment of the present invention. More specifically, the nonce code (T) is added to the secret code (S), byte-by-byte (modulus 256). Each symbol is drawn from the set of {0..255} and may be ASCII or other coding for the convenience of the implementation. In this example embodiment, the code word (S+T) is appended with 0's until the total length is 128 bits

long, then the code word (S+T) is submitted to the SHA-1 hashing algorithm to produce a 16-byte value for the hash code (H), of which 4 bytes are retained for the truncated authentication code (H').

[0026] Figure 6 shows a shortened hash authentication process over a simplex channel with dynamic process parameters, in accordance with one embodiment of the present invention. The method ($\sum$) for combining the secret code (S) and the nonce code (T) is a dynamic parameter that can be set for both the client 201 and server 203. The hashing algorithm and associated parameters (Hash) is a dynamic parameter that can be set for both the client 201 and server 203. The truncation size (2^L) and offset is also a dynamic parameter that can be set for both the client 201 and server 203.

[0027] Figure 7 shows an example shortened hash authentication using algebraic addition of the secret code (S) and nonce code (T) with dynamic parameter settings, in accordance with one embodiment of the present invention. Each symbol is drawn from the set of {0..255} and may be ASCII or other coding for the convenience of the implementation. In this example embodiment, the code word (S+T) is appended with 0's until the total length is 128 bits long, then the code word (S+T) is submitted to an MD5 hashing algorithm to produce a 16-byte value for the hash code (H), of which 6 bytes are retained for the truncated authentication code (H'). Thus, in this example embodiment, the combination method ($\sum$) is set as algebraic addition, the hashing algorithm (Hash) is set as MD5, and the truncation size (2^L) is set as 6 bytes.

[0028] Figure 8 shows a flowchart of a method for performing a shortened hash authentication, in accordance with one embodiment of the present invention. The method includes an operation 801 for storing a secret code on each of a client system and a server system. The secret code is identical on each of the client and server systems. The method also include an operation 803 for generating a nonce code specific to a given authentication

process. The nonce code is uniquely generated for each authentication process based on a monotonically changing source. In one embodiment, the nonce code is a time stamp. An operation 805 is performed to provide the nonce code to both the client system and the server system. In one embodiment, the nonce code is generated at the client system and is transmitted in public view from the client system to the server system. In another embodiment, the nonce code is generated at the server system and is transmitted in public view from the server system to the client system.

[0029] The method continues with an operation 807 to combine the secret code and the nonce code to generate a local code word at each of the client system and server system. In one embodiment, combining the secret code and the nonce code is performed by either a concatenation or an algebraic combination and is performed in an identical manner at each of the client system and server system. An operation 809 is then performed to process each generated local code word through a hashing algorithm to generate a local hash result at each of the client system and the server system. An operation 811 is then performed to truncate each local hash result to obtain a client-generated authentication code and a server-generated authentication code, at each of the client system and server system, respectively. In one embodiment, an operation is performed to specify a common set of hashing parameters and truncation parameters to be used at each of the client system and server system. The hashing parameters may include a hashing algorithm identification and a hash seed, among other parameters. The truncation parameters may include a truncation length and a truncation offset, among other parameters.

[0030] The method further includes an operation 813 for transmitting the client-generated authentication code to the server system. Then, an operation 815 is performed to compare the client-generated authentication code to the server-generated authentication code at the

server system. Equality between the client-generated authentication code and the server-generated authentication code authenticates the client system to the server system.

[0031] As discussed above, the shortened hash authentication methods disclosed herein can be implemented in many different ways. A number of exemplary embodiments are briefly identified below. It should be understood, however, that the generalized shortened hash authentication method as disclosed herein can be implemented in various ways that may not be explicitly identified in the exemplary embodiments below.

[0032] Shortened Hash Authentication – Simplex Channel

[0033] In this embodiment, the client combines a shared secret and a monotonic nonce to generate a client code word, submits the client code word to a hashing algorithm, and truncates the result to obtain the client authentication code. The client transmits the monotonic nonce to the server. The server similarly combines the privately shared secret and the public monotonic nonce to generate a server code word, submits the server code word to the same hashing algorithm, and truncates the result to obtain a server authentication code. If the client and sever authentication codes exactly match, the client identity is authenticated by the server. Otherwise, the server should reject the client credentials.

[0034] Shortened Hash Authentication – Duplex Channel

[0035] In this embodiment, the monotonic nonce is produced by the server and is transmitted to the client. The client combines a shared secret and a monotonic nonce to generate a client code word, submits the client code word to a hashing algorithm, and truncates the result to obtain the client authentication code. The server similarly combines the privately shared secret and the public monotonic nonce to generate a server code word, submits the server code word to the same hashing algorithm, and truncates the result to obtain a server authentication code. If the client and sever authentication codes exactly

match, the client identity is authenticated by the server. Otherwise, the server should reject the client credentials.

[0036] Shortened Hash Authentication – Synchronized Channel

[0037] In this embodiment, the monotonic nonce is inherently known to both the client and server and is not explicitly transmitted. The client combines a shared secret and the monotonic nonce to generate a client code word, submits the client code word to a hashing algorithm, and truncates the result to obtain the client authentication code. The server similarly combines the privately shared secret and the public monotonic nonce to generate a server code word, submits the server code word to the same hashing algorithm, and truncates the result to obtain a server authentication code. If the client and sever authentication codes exactly match, the client identity is authenticated by the server. Otherwise, the server should reject the client credentials.

[0038] Shortened Hash Authentication – Concatenated Codes, Simplex Channel

[0039] This embodiment is a particular variant of the above-described Simplex Channel embodiment in which the shared secret and monotonic nonce are concatenated, in any order, and with optional padding codes to obtain a canonical code length. The monotonic nonce is produced by the client and transmitted to the server.

[0040] Shortened Hash Authentication – Concatenated Codes, Duplex Channel

[0041] This embodiment is a particular variant of the above-described Duplex Channel embodiment in which the shared secret and monotonic nonce are concatenated, in any order, and with optional padding codes to obtain a canonical code length. The monotonic nonce is produced by the server and transmitted to the client.

[0042] Shortened Hash Authentication – Concatenated Codes, Synchronized Channel

[0043] This embodiment is a particular variant of the above-described Synchronized Channel embodiment in which the shared secret and monotonic nonce are concatenated, in any order, and with optional padding codes to obtain a canonical code length. The monotonic nonce is inherently known to both the client and server and is not explicitly transmitted.

[0044] Shortened Hash Authentication – Algebraic Codes, Simplex Channel

[0045] This embodiment is a particular variant of the above-described Simplex Channel embodiment in which the shared secret and monotonic nonce are combined by algebraic operations including but not limited to addition, multiplication or division, in any order, and with optional padding codes to obtain a canonical code length. The monotonic nonce is produced by the client and transmitted to the server.

[0046] Shortened Hash Authentication – Algebraic Codes, Duplex Channel

[0047] This embodiment is a particular variant of the above-described Duplex Channel embodiment in which the shared secret and monotonic nonce are combined by algebraic operations including but not limited to addition, multiplication or division, in any order, and with optional padding codes to obtain a canonical code length. The monotonic nonce is produced by the server and transmitted to the client.

[0048] Shortened Hash Authentication – Algebraic Codes, Synchronized Channel

[0049] This embodiment is a particular variant of the above-described Synchronized Channel embodiment in which the shared secret and monotonic nonce are combined by algebraic operations including but not limited to addition, multiplication or division, in any order, and with optional padding codes to obtain a canonical code length. The monotonic nonce is inherently known to both the client and server and is not explicitly transmitted.

[0050] Shortened Hash Authentication – General Codes, Simplex Channel

[0051] This embodiment is a particular variant of the above-described Simplex Channel embodiment in which the shared secret and monotonic nonce are combined by any operation with output dependent upon both the secret and nonce, in any order, and with optional padding codes to obtain a canonical code length. The monotonic nonce is produced by the client and transmitted to the server.

[0052] Shortened Hash Authentication – General Codes, Duplex Channel

[0053] This embodiment is a particular variant of the above-described Duplex Channel embodiment in which the shared secret and monotonic nonce are combined by any operation with output dependent upon both the secret and nonce, in any order, and with optional padding codes to obtain a canonical code length. The monotonic nonce is produced by the server and transmitted to the client.

[0054] Shortened Hash Authentication – General Codes, Synchronized Channel

[0055] This embodiment is a particular variant of the above-described Synchronized Channel embodiment in which the shared secret and monotonic nonce are combined by any operation with output dependent upon both the secret and nonce, in any order, and with optional padding codes to obtain a canonical code length. The monotonic nonce is inherently known to both the client and server and is not explicitly transmitted.

[0056] Shortened Hash Authentication – Dynamic Parameters, Concatenated Codes, Simplex Channel

[0057] This embodiment is a particular variant of the above-described Simplex Channel embodiment in which the shared secret and monotonic nonce are concatenated, in any order, and with optional padding codes to obtain a canonical code length. The monotonic nonce is produced by the client and transmitted to the server. One or more hashing and reduction parameters including but not limited to the hashing algorithm, hash seed,

truncated code length and offset, and combination method are transmitted so that the server and client implementations dynamically match.

[0058] Shortened Hash Authentication – Dynamic Parameters, Concatenated Codes, Duplex Channel

[0059] This embodiment is a particular variant of the above-described Duplex Channel embodiment in which the shared secret and monotonic nonce are concatenated, in any order, and with optional padding codes to obtain a canonical code length. The monotonic nonce is produced by the server and transmitted to the client. One or more hashing and reduction parameters including but not limited to the hashing algorithm, hash seed, truncated code length and offset, and combination method are transmitted so that the server and client implementations dynamically match.

[0060] Shortened Hash Authentication – Dynamic Parameters, Concatenated Codes, Synchronized Channel

[0061] This embodiment is a particular variant of the above-described Synchronized Channel embodiment in which the shared secret and monotonic nonce are concatenated, in any order, and with optional padding codes to obtain a canonical code length. The monotonic nonce is inherently known to both the client and server and is not explicitly transmitted. One or more hashing and reduction parameters including but not limited to the hashing algorithm, hash seed, truncated code length and offset, and combination method are transmitted so that the server and client implementations dynamically match.

[0062] Shortened Hash Authentication – Dynamic Parameters, Algebraic Codes, Simplex Channel

[0063] This embodiment is a particular variant of the above-described Simplex Channel embodiment in which the shared secret and monotonic nonce are combined by algebraic operations including but not limited to addition, multiplication or division, in any order,

and with optional padding codes to obtain a canonical code length. The monotonic nonce is produced by the client and transmitted to the server. One or more hashing and reduction parameters including but not limited to the hashing algorithm, hash seed, truncated code length and offset, and combination method are transmitted so that the server and client implementations dynamically match.

[0064] Shortened Hash Authentication – Dynamic Parameters, Algebraic Codes, Duplex Channel

[0065] This embodiment is a particular variant of the above-described Duplex Channel embodiment in which the shared secret and monotonic nonce are combined by algebraic operations including but not limited to addition, multiplication or division, in any order, and with optional padding codes to obtain a canonical code length. The monotonic nonce is produced by the server and transmitted to the client. One or more hashing and reduction parameters including but not limited to the hashing algorithm, hash seed, truncated code length and offset, and combination method are transmitted so that the server and client implementations dynamically match.

[0066] Shortened Hash Authentication – Dynamic Parameters, Algebraic Codes, Synchronized Channel

[0067] This embodiment is a particular variant of the above-described Synchronized Channel embodiment in which the shared secret and monotonic nonce are combined by algebraic operations including but not limited to addition, multiplication or division, in any order, and with optional padding codes to obtain a canonical code length. The monotonic nonce is inherently known to both the client and server and is not explicitly transmitted. One or more hashing and reduction parameters including but not limited to the hashing algorithm, hash seed, truncated code length and offset, and combination method are transmitted so that the server and client implementations dynamically match.

[0068] Shortened Hash Authentication – Dynamic Parameters, General Codes, Simplex Channel

[0069] This embodiment is a particular variant of the above-described Simplex Channel embodiment in which the shared secret and monotonic nonce are combined by any operation with output dependent upon both the secret and nonce, in any order, and with optional padding codes to obtain a canonical code length. The monotonic nonce is produced by the client and transmitted to the server. One or more hashing and reduction parameters including but not limited to the hashing algorithm, hash seed, truncated code length and offset, and combination method are transmitted so that the server and client implementations dynamically match.

[0070] Shortened Hash Authentication – Dynamic Parameters, General Codes, Duplex Channel

[0071] This embodiment is a particular variant of the above-described Duplex Channel embodiment in which the shared secret and monotonic nonce are combined by any operation with output dependent upon both the secret and nonce, in any order, and with optional padding codes to obtain a canonical code length. The monotonic nonce is produced by the server and transmitted to the client. One or more hashing and reduction parameters including but not limited to the hashing algorithm, hash seed, truncated code length and offset, and combination method are transmitted so that the server and client implementations dynamically match.

[0072] Shortened Hash Authentication – Dynamic Parameters, General Codes, Synchronized Channel

[0073] This embodiment is a particular variant of the above-described Synchronized Channel embodiment in which the shared secret and monotonic nonce are combined by any operation with output dependent upon both the secret and nonce, in any order, and with

optional padding codes to obtain a canonical code length. The monotonic nonce is inherently known to both the client and server and is not explicitly transmitted. One or more hashing and reduction parameters including but not limited to the hashing algorithm, hash seed, truncated code length and offset, and combination method are transmitted so that the server and client implementations dynamically match.

[0074] Implicit Session Key Agreement

[0075] Principles of the shortened hash authentication methods described above can also be applied in a method for implicit session key agreement, as described below. The method for implicit session key agreement provides for strong session key agreement security while reducing the communication burden between two entities, i.e., between a client and a server. The implicit session key agreement method is well-suited for application in high-security, low-bandwidth communication channels in distributed networks where low cost and low power consumption are required. Also, session keys generated through the implicit session key agreement method may be shared with third parties for the duration of a session or when securing other sessions, and without compromising the secret code used to generate the session key.

[0076] Figure 9 shows a method for performing implicit session key agreement with a simplex channel, in accordance with one embodiment of the present invention. A client 901 combines the client's secret code (S_C) with a monotonic nonce (T_C) to generate a code word ($\sum_C$), and submits the code word ($\sum_C$) to a selected hash algorithm ($Hash_C$) to produce a session key (H_C). Similarly, a server 903 combines the server's secret code (S_S), where ($S_S = S_C$), with the synchronized monotonic nonce (T_S), where ($T_S = T_C$), to generate a code word ($\sum_S$), and submits the code word ($\sum_S$) to a selected hash algorithm ($Hash_S$), where ($Hash_S = Hash_C$), to produce a session key (H_S). Because session keys are used in communication protocols, the client session key (H_C) and the server session key (H_S)

should match for successful client-server communication. The implicit session key agreement method disclosed herein is useful for low-bandwidth networks of well-known peers because the shared secret is not transmitted in whole or part, and because the monotonic nonce is known in common among synchronized peers. Additionally, in one embodiment, a session key may be shared with one or more third-party network peers while retaining the privacy of the secret code word and all other session keys.

[0077] Figure 10 shows a comparison of the simplex and duplex embodiments in the implicit session key agreement methods, in accordance with one embodiment of the present invention. In the simplex embodiment, the client explicitly synchronizes by producing the nonce (T) and providing it to the server. In the duplex embodiment, the server explicitly synchronizes by producing and sharing the nonce (T) as needed when the client is ready to transmit. The simplex embodiment may be used where simplex channels and/or excessive channel latency are present. The duplex embodiment may be used where the client cannot reliably produce a monotonic nonce and/or when the server can produce the nonce more conveniently and/or for less cost.

[0078] Figure 11 shows an example implicit session key agreement implementation, in accordance with one embodiment of the present invention. In this embodiment, the secret code (S) is concatenated with the nonce code (T) to generate the code word (ST). Each symbol, i.e., code, is drawn from the set of {0..255} and may be ASCII or other coding for the convenience of the implementation. In this example, the code word (ST) is appended with 0's until the total length is 128 bits long. Then, the code word (ST) is submitted to the SHA-1 hashing algorithm to produce a 16-byte value for the hash code (H), where the hash code (H) represents the session key.

[0079] Figure 12 shows another example implicit session key agreement implementation, in accordance with one embodiment of the present invention. In this embodiment, the

nonce code (T) is added to the secret code (S), byte-by-byte (modulus 256). Each symbol, i.e., code, is drawn from the set of {0..255} and may be ASCII or other coding for the convenience of the implementation. In this example, the code word (S+T) is appended with 0's until the total length is 128 bits long. Then, the code word (S+T) is submitted to the SHA-1 hashing algorithm to produce a 16-byte value for the hash code (H), where the hash code (H) represents the session key.

[0080] Figure 13 shows an implicit session key agreement process with dynamic process parameters, in accordance with one embodiment of the present invention. The method ($\sum$) for combining the secret code (S) and the nonce code (T) is a dynamic parameter that can be set for both the client 901 and server 903. The hashing algorithm and associated parameters (Hash) is a dynamic parameter that can be set for both the client 901 and server 903.

[0081] The method for combining the secret code and the nonce code may be known *a priori* and encoded in the design of alternate embodiments, or it may be dynamically selected by information shared in common between the client and server. This information may be encoded within the message, implicit in the connection address or port number, or be conveyed by some other method, implicit or explicit. The method for combining the secret code and the nonce code should be identical between the client and server.

[0082] In one embodiment, the hashing algorithm is pre-defined. In other embodiments, the hashing algorithm may be known *a priori* and encoded in the design of alternate embodiments, or it may be dynamically selected by information shared in common between the client and server. This information may be encoded within the message, implicit in the connection address or port number, or be conveyed by some other method, implicit or explicit. The hashing algorithm should be identical between the client and server.

[0083] Figure 14 shows an example in which the method of combination and hash algorithm are dynamically specified, in accordance with one embodiment of the present invention. In this embodiment, the secret code (S) and nonce code (T) are combined by bit-wise addition to generated the code word (S+T). Each symbol, i.e., code, is drawn from the set of {0..255} and may be ASCII or other coding for the convenience of the implementation. In this example, the code word (S+T) is appended with 0's until the total length is 128 bits long. Then, the code word (S+T) is submitted to the MD5 hashing algorithm to produce a 16-byte value for the hash code (H), where the hash code (H) represents the session key.

[0084] Based on the foregoing, it should be appreciated that the implicit session key agreement method provides for strong key agreement between a client and a server, where each possess a common secret code, and at a reduced communication channel bandwidth. Moreover, the implicit session key agreement method strongly secures distributed communication networks while minimizing the impact to overall communication channel bandwidth. Also, it should be appreciated that the session key (H) originates from a strong hash algorithm. Therefore, because the session key is generated from a mutual secret code and public nonce code, the mutual secret code cannot be practically deduced even if the session key is known. Additionally, the monotonic nonce ensures that the session key is not reused.

[0085] The implicit session key agreement methods discussed above can be implemented in many different ways. A number of exemplary embodiments are briefly identified below. It should be understood, however, that the generalized implicit session key agreement method as disclosed herein can be implemented in various ways that may not be explicitly identified in the exemplary embodiments below.

[0086] Implicit Session Key Agreement – Synchronized Channel

[0087] In this embodiment, both the client and server independently combine a shared secret and a monotonic nonce and submit the resulting code word to a hashing algorithm to obtain the session key. The monotonic nonce is inherently known to both the client and server and is not explicitly transmitted.

[0088] Implicit Session Key Agreement – Simplex Channel

[0089] In this embodiment, the monotonic nonce is produced by the client and is transmitted to the server. Both the client and server independently combine a shared secret and the monotonic nonce and submit the resulting code word to a hashing algorithm to obtain the session key.

[0090] Implicit Session Key Agreement – Duplex Channel

[0091] In this embodiment, the monotonic nonce is produced by the server and is transmitted to the client. Both the client and server independently combine a shared secret and the monotonic nonce and submit the resulting code word to a hashing algorithm to obtain the session key.

[0092] Implicit Session Key Agreement – Concatenated Codes, Synchronized Channel

[0093] This embodiment is like the synchronized channel embodiment above with the shared secret and monotonic nonce combined by concatenation, in any order, and with optional padding codes to obtain a canonical code length.

[0094] Implicit Session Key Agreement – Concatenated Codes, Simplex Channel

[0095] This embodiment is like the simplex channel embodiment above with the shared secret and monotonic nonce combined by concatenation, in any order, and with optional padding codes to obtain a canonical code length.

[0096] Implicit Session Key Agreement – Concatenated Codes, Duplex Channel

[0097] This embodiment is like the duplex channel embodiment above with the shared secret and monotonic nonce combined by concatenation, in any order, and with optional padding codes to obtain a canonical code length.

[0098] Implicit Session Key Agreement – Algebraic Codes, Synchronized Channel

[0099] This embodiment is like the synchronized channel embodiment above with the shared secret and monotonic nonce combined by algebraic operations including but not limited to addition, multiplication or division, in any order, and with optional padding codes to obtain a canonical code length.

[00100] Implicit Session Key Agreement – Algebraic Codes, Simplex Channel

[00101] This embodiment is like the simplex channel embodiment above with the shared secret and monotonic nonce combined by algebraic operations including but not limited to addition, multiplication or division, in any order, and with optional padding codes to obtain a canonical code length.

[00102] Implicit Session Key Agreement – Algebraic Codes, Duplex Channel

[00103] This embodiment is like the duplex channel embodiment above with the shared secret and monotonic nonce combined by algebraic operations including but not limited to addition, multiplication or division, in any order, and with optional padding codes to obtain a canonical code length.

[00104] Implicit Session Key Agreement – General Codes, Synchronized Channel

[00105] This embodiment is like the synchronized channel embodiment above with the shared secret and monotonic nonce combined by any operation with output dependent upon both the secret code and nonce, in any order, and with optional padding codes to obtain a canonical code length.

[00106] Implicit Session Key Agreement – General Codes, Simplex Channel

[00107] This embodiment is like the simplex channel embodiment above with the shared secret and monotonic nonce combined by any operation with output dependent upon both the secret code and nonce, in any order, and with optional padding codes to obtain a canonical code length.

[00108] Implicit Session Key Agreement – General Codes, Duplex Channel

[00109] This embodiment is like the duplex channel embodiment above with the shared secret and monotonic nonce combined by any operation with output dependent upon both the secret code and nonce, in any order, and with optional padding codes to obtain a canonical code length.

[00110] Implicit Session Key Agreement – Dynamic Parameters

[00111] Any of the above-described embodiments can be implemented in a dynamic manner through specification/recognition of dynamic parameter settings including hashing parameters (including but not limited to the hashing algorithm and hash seed) and/or a particular secret code and nonce combination method. The dynamic parameter settings are transmitted or otherwise recognized by the client and server so that the client and server implementations dynamically match.

[00112] Figure 15 shows a flowchart of a method for implicit session key agreement, in accordance with one embodiment of the present invention. The method includes an operation 1501 for storing a secret code on each of a client system and a server system. The secret code is identical on each of the client and server systems. The method also includes an operation 1503 for generating a nonce code specific to a given authentication process. In one embodiment, the nonce code is uniquely generated for each session based on a monotonically changing source. For example, in one embodiment, the nonce code is a time stamp. An operation 1505 is performed to provide the nonce code to both the client system and the server system. In one embodiment, providing the nonce code

in operation 1505 includes generating the nonce code at the client system and transmitting the nonce code in public view from the client system to the server system. In another embodiment, providing the nonce code in operation 1505 includes generating the nonce code at the server system and transmitting the nonce code in public view from the server system to the client system.

[00113] The method further includes an operation 1507 for combining the secret code and the nonce code to generate a local code word at each of the client system and server system. In one embodiment, combination of the secret code and the nonce code is performed by either a concatenation or an algebraic combination and is performed in an identical manner at each of the client system and server system. The method also includes an operation 1509 for processing each generated local code word through a hashing algorithm to generate a local hash result at each of the client system and the server system. Additionally, an operation 1511 is performed to use the local hash result at each of the client system and the server system as an implicit session key for data communication between the client and server systems. In one embodiment, the method also includes an operation for specifying a common set of combination parameters and hashing parameters to be used at each of the client system and server system. The combination parameters can include identification of a method by which the secret code and the nonce code are to be combined to generate the local code word. The hashing parameters may include a hashing algorithm identification and a hash seed.

[00114] It should be understood that the invention described herein can be embodied as computer readable code on a computer readable medium. The computer readable medium is any data storage device that can store data which can thereafter be read by a computer system. Examples of the computer readable medium include hard drives, network attached storage (NAS), read-only memory, random-access memory, CD-ROMs,

CD-Rs, CD-RWs, magnetic tapes, and other optical and non-optical data storage devices. The computer readable medium can also be distributed over a network of coupled computer systems so that the computer readable code is stored and executed in a distributed fashion.

[00115] Any of the operations described herein that form part of the invention are useful machine operations. The invention also relates to a device or an apparatus for performing these operations. The apparatus may be specially constructed for the required purpose, such as a special purpose computer. When defined as a special purpose computer, the computer can also perform other processing, program execution or routines that are not part of the special purpose, while still being capable of operating for the special purpose. Alternatively, the operations may be processed by a general purpose computer selectively activated or configured by one or more computer programs stored in the computer memory, cache, or obtained over a network. When data is obtained over a network the data maybe processed by other computers on the network, e.g., a cloud of computing resources.

[00116] The embodiments of the present invention can also be defined as a machine that transforms data from one state to another state. The data may represent an article, that can be represented as an electronic signal and electronically manipulate data. The transformed data can, in some cases, be visually depicted on a display, representing the physical object that results from the transformation of data. The transformed data can be saved to storage generally, or in particular formats that enable the construction or depiction of a physical and tangible object. In some embodiments, the manipulation can be performed by a processor. In such an example, the processor thus transforms the data from one thing to another. Still further, the methods can be processed by one or more machines or processors that can be connected over a network. Each machine can transform data

from one state or thing to another, and can also process data, save data to storage, transmit data over a network, display the result, or communicate the result to another machine.

[00117] While this invention has been described in terms of several embodiments, it will be appreciated that those skilled in the art upon reading the preceding specifications and studying the drawings will realize various alterations, additions, permutations and equivalents thereof. Therefore, it is intended that the present invention includes all such alterations, additions, permutations, and equivalents as fall within the true spirit and scope of the invention.

What is claimed is:

CLAIMS

1. A method for performing a shortened hash authentication, comprising:

generating a first hash result at a client system in accordance with hash input parameters known to the client system;

generating a second hash result at a server system in accordance with hash input parameters known to the server system;

truncating each of the first hash result and the second hash result in a same manner;

transmitting the truncated first hash result from the client system to the server system; and

comparing the truncated first hash result as transmitted to the server system with the truncated second hash result generated at the server system, wherein equality between the truncated first hash result as transmitted to the server system and the truncated second hash result generated at the server system authenticates the client system to the server system.

2. The method of claim 1, wherein the hash input parameters known to the client system include a client code word representing a combination of a secret code and a nonce code as known to the client system, and wherein the hash input parameters known to the server system include a server code word representing a combination of a secret code and a nonce code known to the server system.

3. The method of claim 2, wherein the secret code known to the client system and the secret code known to the server system are identical and are stored on the client and server systems prior to authentication.

4. The method of claim 2, wherein the nonce code is uniquely generated for each authentication process.

5. The method of claim 4, wherein the nonce code is a time stamp.

6. The method of claim 4, further comprising:

generating the nonce code at the client system for use in the client code word; and

transmitting the nonce code from the client system to the server system for use in the server code word, wherein the nonce code is transmitted in public view.

7. The method of claim 4, further comprising:

generating the nonce code at the server system for use in the server code word; and

transmitting the nonce code from the server system to the client system for use in the client code word, wherein the nonce code is transmitted in public view.

8. The method of claim 4, further comprising:

generating the nonce code independently at each of the client system and server system, wherein the nonce code is generated based on identical information known to both the client system and server system such that the nonce code known to the client system is identical to the nonce code known to the server system.

9. The method of claim 2, wherein the secret code and the nonce code known to the client system are combined in either a concatenated manner or an algebraic manner to form the client code word, and wherein the secret code and the nonce code known to the

server system are combined in same manner to form the server code word as used to form the client code word.

10. The method of claim 1, wherein both the first hash result and second hash result are generated using an identical hashing algorithm.

11. The method of claim 1, further comprising:

specifying a truncation size for both the first hash result and the second hash result, wherein the truncation size is specified based on a bandwidth available for transmission of the truncated first hash result and a probability of malevolent decoding of the truncated first hash result.

12. A method for performing a shortened hash authentication, comprising:

storing a secret code on each of a client system and a server system, wherein the secret code is identical on each of the client and server systems;

generating a nonce code specific to a given authentication process;

providing the nonce code to both the client system and the server system;

combining the secret code and the nonce code to generate a local code word at each of the client system and server system;

processing each generated local code word through a hashing algorithm to generate a local hash result at each of the client system and the server system;

at each of the client system and server system, truncating each local hash result to obtain a client-generated authentication code and a server-generated authentication code;

transmitting the client-generated authentication code to the server system; and

comparing the client-generated authentication code to the server-generated authentication code at the server system, wherein equality between the client-generated authentication code and the server-generated authentication code authenticates the client system to the server system.

13. The method of claim 12, wherein the nonce code is uniquely generated for each authentication process based on a monotonically changing source.

14. The method of claim 13, wherein the nonce code is a time stamp.

15. The method of claim 12, wherein providing the nonce code includes generating the nonce code at the client system and transmitting the nonce code in public view from the client system to the server system.

16. The method of claim 12, wherein providing the nonce code includes generating the nonce code at the server system and transmitting the nonce code in public view from the server system to the client system.

17. The method of claim 12, wherein combining the secret code and the nonce code is performed by either a concatenation or an algebraic combination and is performed in an identical manner at each of the client system and server system.

18. The method of claim 12, further comprising:

specifying a common set of hashing parameters and truncation parameters to be used at each of the client system and server system, wherein the hashing parameters

include a hashing algorithm identification and a hash seed, and wherein the truncation parameters include a truncation length and a truncation offset.

19. A system for performing a shortened hash authentication, comprising:

a client defined to generate a first hash result in accordance with hash input parameters known to the client, and further defined to truncate the first hash result to obtain a client-generated authentication code;

a server defined to generate a second hash result in accordance with hash input parameters known to the server, and further defined to truncate the second hash result to obtain a server-generated authentication code;

wherein the client is defined to transmit the client-generated authentication code to the server, and

wherein the server is defined to receive the client-generated authentication code and compare the client-generated authentication code to the server-generated authentication code, wherein equality between the client-generated and server-generated authentication codes authenticates the client to the server.

20. The method of claim 19, further comprising:

a simplex communication connection between the client and the server, wherein the client is defined to generate and transmit a nonce code to the server, wherein the nonce code is a component of the hash input parameters for generating each of the first and second hash results.

21. The method of claim 19, further comprising:

a duplex communication connection between the client and the server, wherein the server is defined to generate and transmit a nonce code to the client, wherein the nonce code is a component of the hash input parameters for generating each of the first and second hash results.

22. The method of claim 19, a synchronized communication connection between the client and the server, wherein both the client and the server have access to a commonly known nonce code, wherein the nonce code is a component of the hash input parameters for generating each of the first and second hash results

23. A method for implicit session key agreement, comprising:

storing a secret code on each of a client system and a server system, wherein the secret code is identical on each of the client and server systems;

generating a nonce code specific to a given authentication process;

providing the nonce code to both the client system and the server system;

combining the secret code and the nonce code to generate a local code word at each of the client system and server system;

processing each generated local code word through a hashing algorithm to generate a local hash result at each of the client system and the server system; and

using the local hash result at each of the client system and the server system as an implicit session key for data communication between the client and server systems.

24. The method of claim 23, wherein the nonce code is uniquely generated for each session based on a monotonically changing source.

25. The method of claim 24, wherein the nonce code is a time stamp.

26. The method of claim 23, wherein providing the nonce code includes generating the nonce code at the client system and transmitting the nonce code in public view from the client system to the server system.

27. The method of claim 23, wherein providing the nonce code includes generating the nonce code at the server system and transmitting the nonce code in public view from the server system to the client system.

28. The method of claim 23, wherein combining the secret code and the nonce code is performed by either a concatenation or an algebraic combination and is performed in an identical manner at each of the client system and server system.

29. The method of claim 23, further comprising:

specifying a common set of combination parameters and hashing parameters to be used at each of the client system and server system, wherein the combination parameters include identification of a method by which the secret code and the nonce code are to be combined to generate the local code word, and wherein the hashing parameters include a hashing algorithm identification and a hash seed.

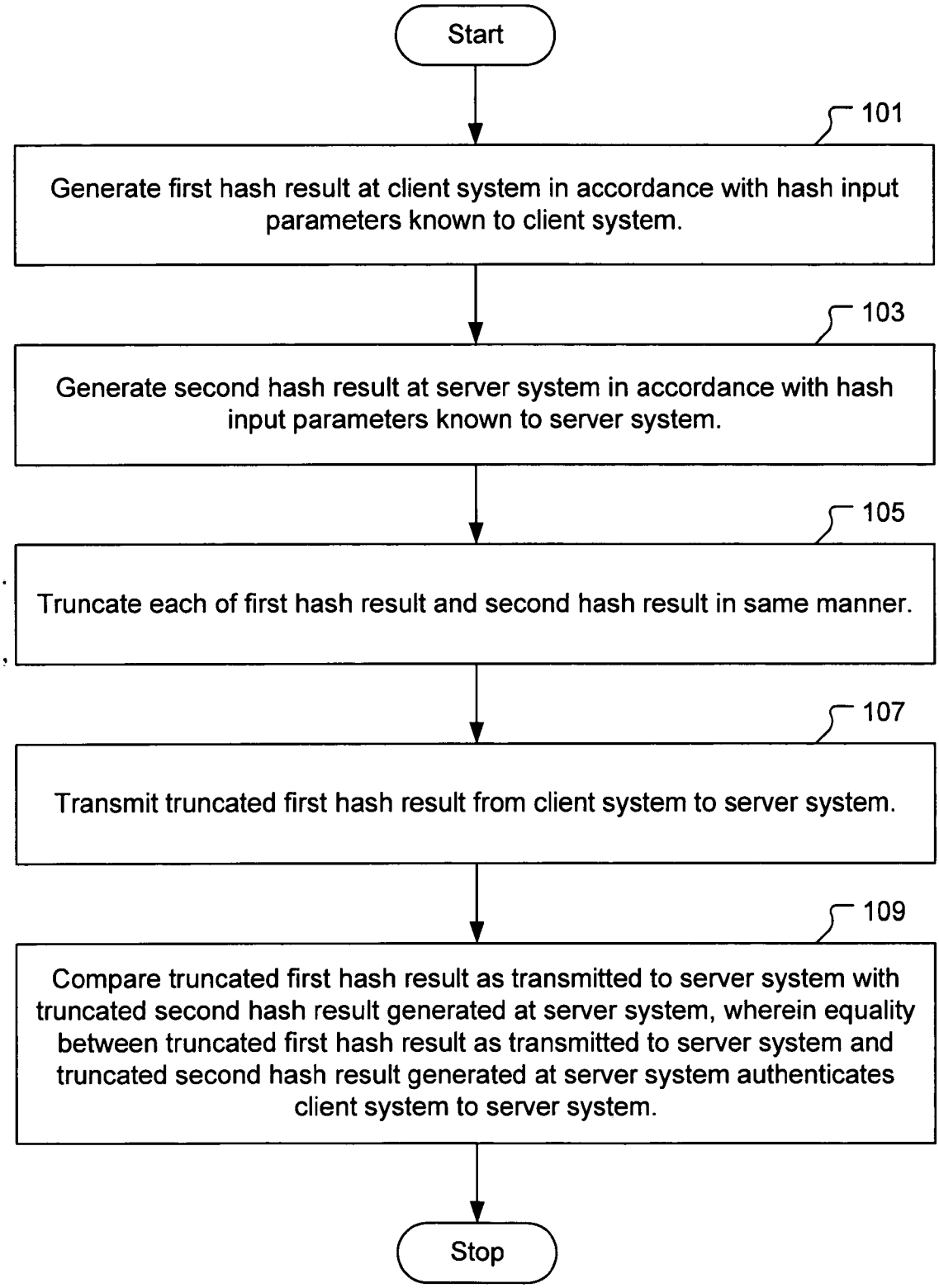
Start
101
Generate first hash result at client system in accordance with hash input parameters known to client system.
103
Generate second hash result at server system in accordance with hash input parameters known to server system.
105
Truncate each of first hash result and second hash result in same manner.
107
Transmit truncated first hash result from client system to server system.
109
Compare truncated first hash result as transmitted to server system with truncated second hash result generated at server system, wherein equality between truncated first hash result as transmitted to server system and truncated second hash result generated at server system authenticates client system to server system.
Stop

Fig. 1

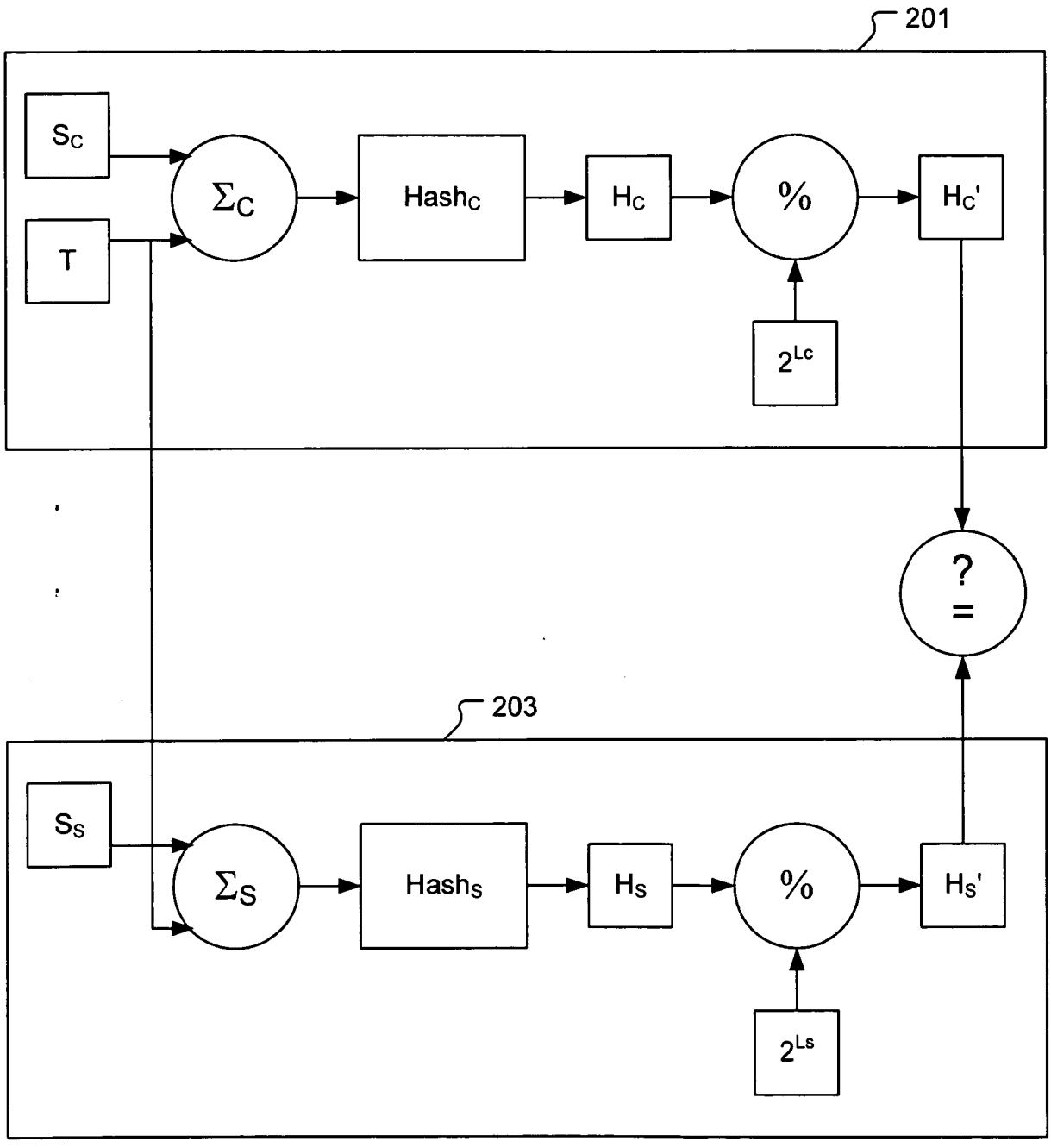

Fig. 2

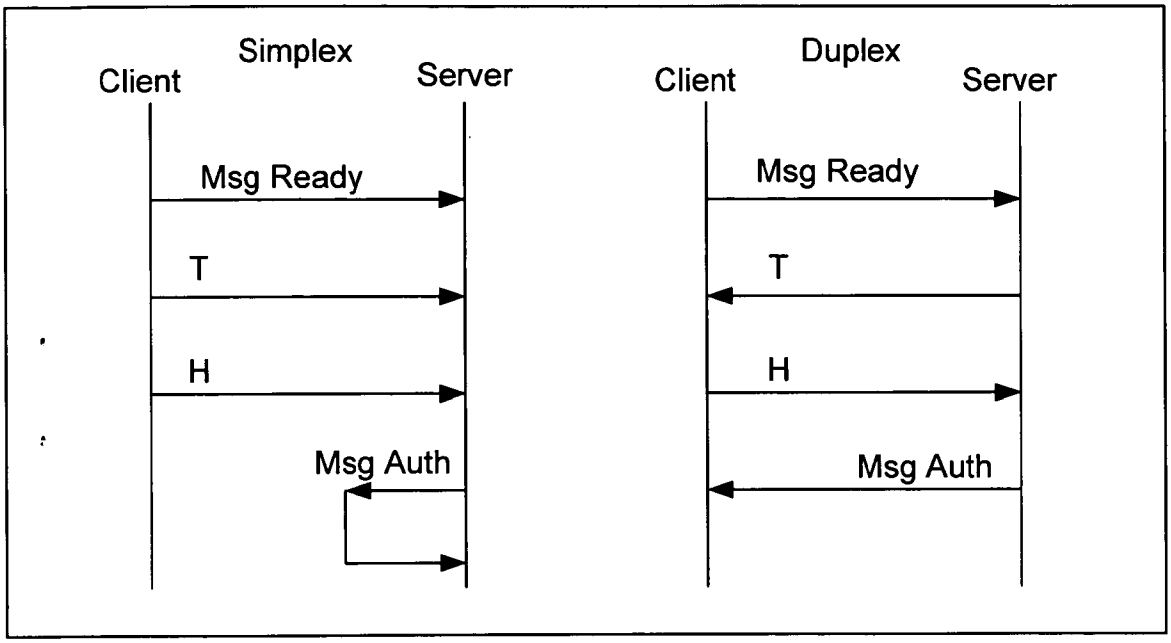
Simplex
Client
Server
Msg Ready
T
H
Msg Auth
Duplex
Client
Server
Msg Ready
T
H
Msg Auth

Fig. 3

Shortened Hash Authentication
Secret (S): secret_password
Nonce (T): Wed Feb 21 22:42:22 2007
Concatenation (ST): secret_passwordWed Feb 21 22:42:22 2007
Hash Method: SHA1
Hash Code (H): 7dae97e19a067e3a886c00aa14c1653e
Auth Length (|H'|): 4 bytes
Auth Code (H'): 14c1653e

Fig. 4

Shortened Hash Authentication
Secret (S): secret_password
Nonce (T): Wed Feb 21 22:42:22 2007
Addition (S+T): [illegible]
Hash Method: SHA1
Hash Code (H): 2857e7a061a15231a71e7abf5b08da7b
Auth Length (|H'|): 4 bytes
Auth Code (H'): 5b08da7b

Fig. 5

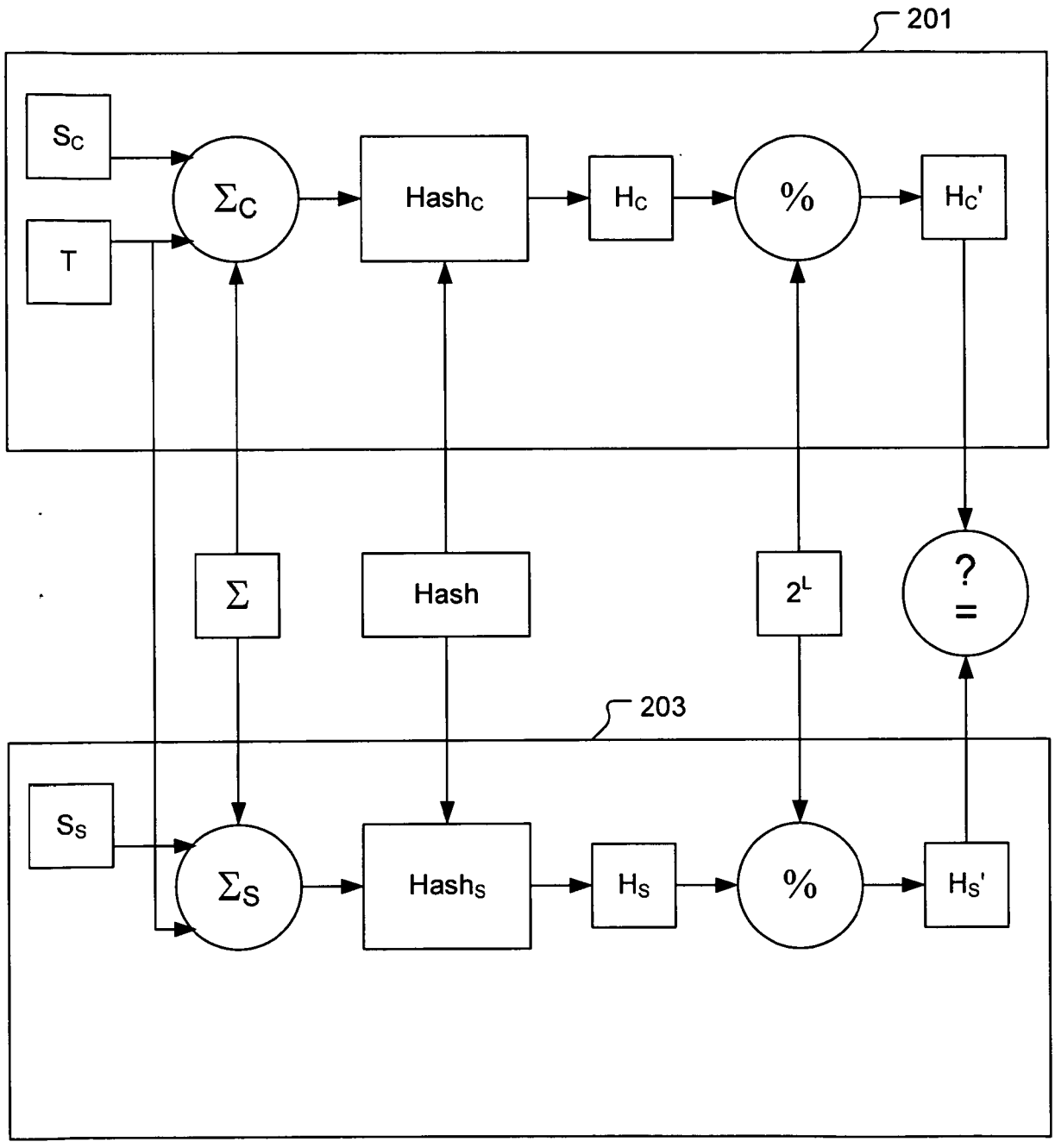

Fig. 6

Shortened Hash Authentication
Secret (S): secret_password
Nonce (T): Wed Feb 21 22:42:22 2007
Addition (S+T): [illegible]
Hash Method: MD5
Hash Code (H): 91fef0520068419bb7e1af3119fae17f
Auth Length (|H'|): 6 bytes
Auth Code (H'): af3119fae17f

Fig. 7

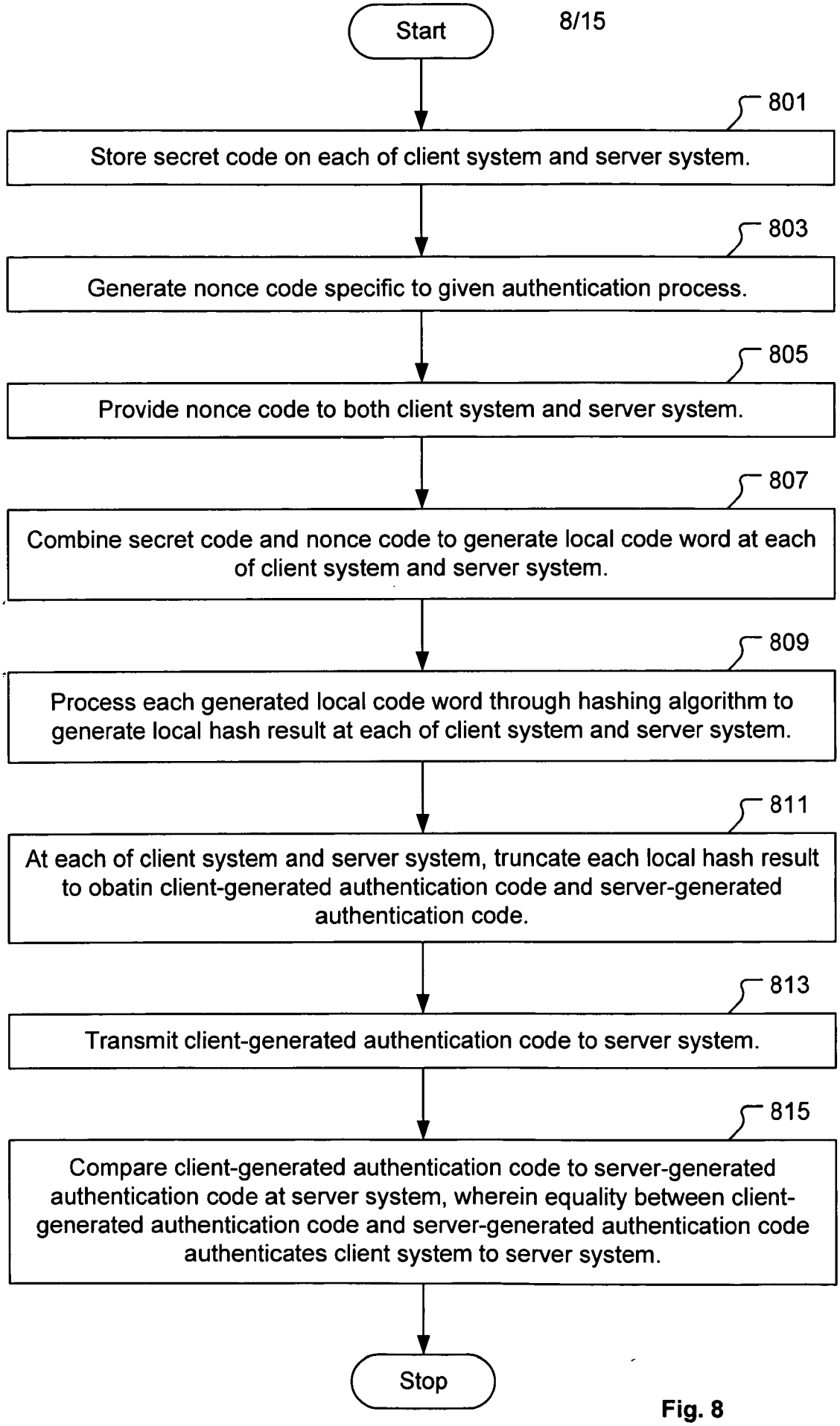
Start
801
Store secret code on each of client system and server system.
803
Generate nonce code specific to given authentication process.
805
Provide nonce code to both client system and server system.
807
Combine secret code and nonce code to generate local code word at each of client system and server system.
809
Process each generated local code word through hashing algorithm to generate local hash result at each of client system and server system.
811
At each of client system and server system, truncate each local hash result to obatin client-generated authentication code and server-generated authentication code.
813
Transmit client-generated authentication code to server system.
815
Compare client-generated authentication code to server-generated authentication code at server system, wherein equality between client-generated authentication code and server-generated authentication code authenticates client system to server system.
Stop

Fig. 8

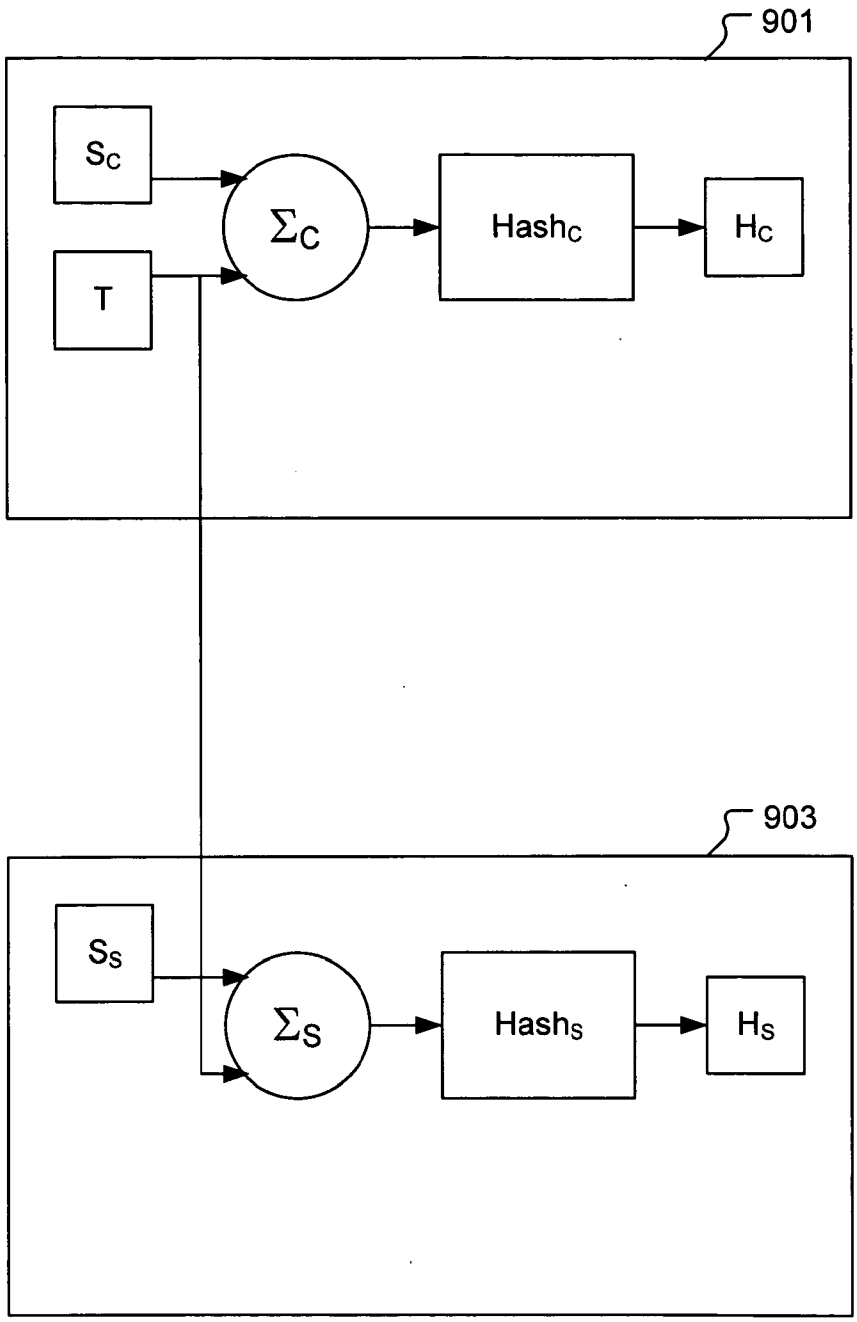

Fig. 9

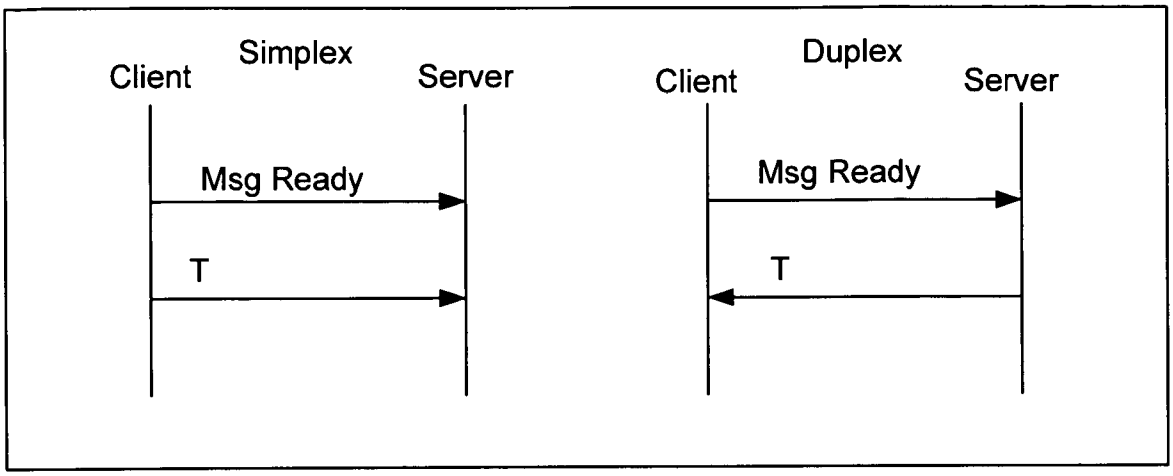
Simplex
Client
Server
Msg Ready
T
Duplex
Client
Server
Msg Ready
T

Fig. 10

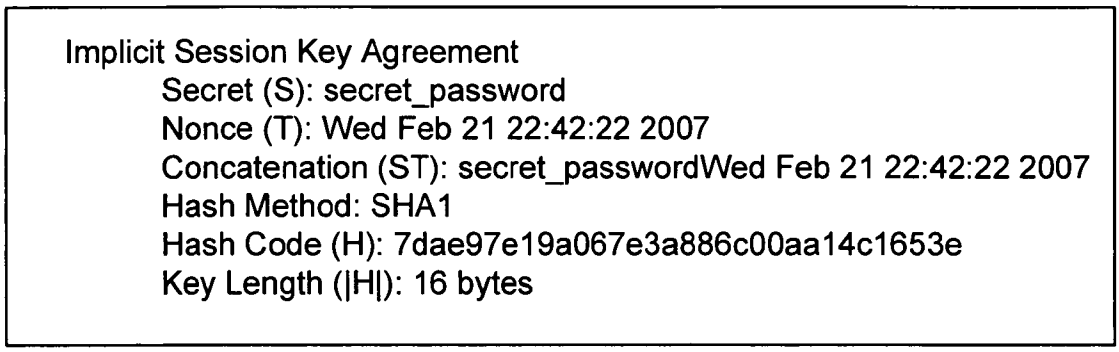
Implicit Session Key Agreement
Secret (S): secret_password
Nonce (T): Wed Feb 21 22:42:22 2007
Concatenation (ST): secret_passwordWed Feb 21 22:42:22 2007
Hash Method: SHA1
Hash Code (H): 7dae97e19a067e3a886c00aa14c1653e
Key Length (|H|): 16 bytes

Fig. 11

Implicit Session Key Agreement
Secret (S): secret_password
Nonce (T): Wed Feb 21 22:42:22 2007
Addition (S+T): [illegible]
Hash Method: SHA1
Hash Code (H): 2857e7a061a15231a71e7abf5b08da7b
Key Length (|H|): 16 bytes

Fig. 12

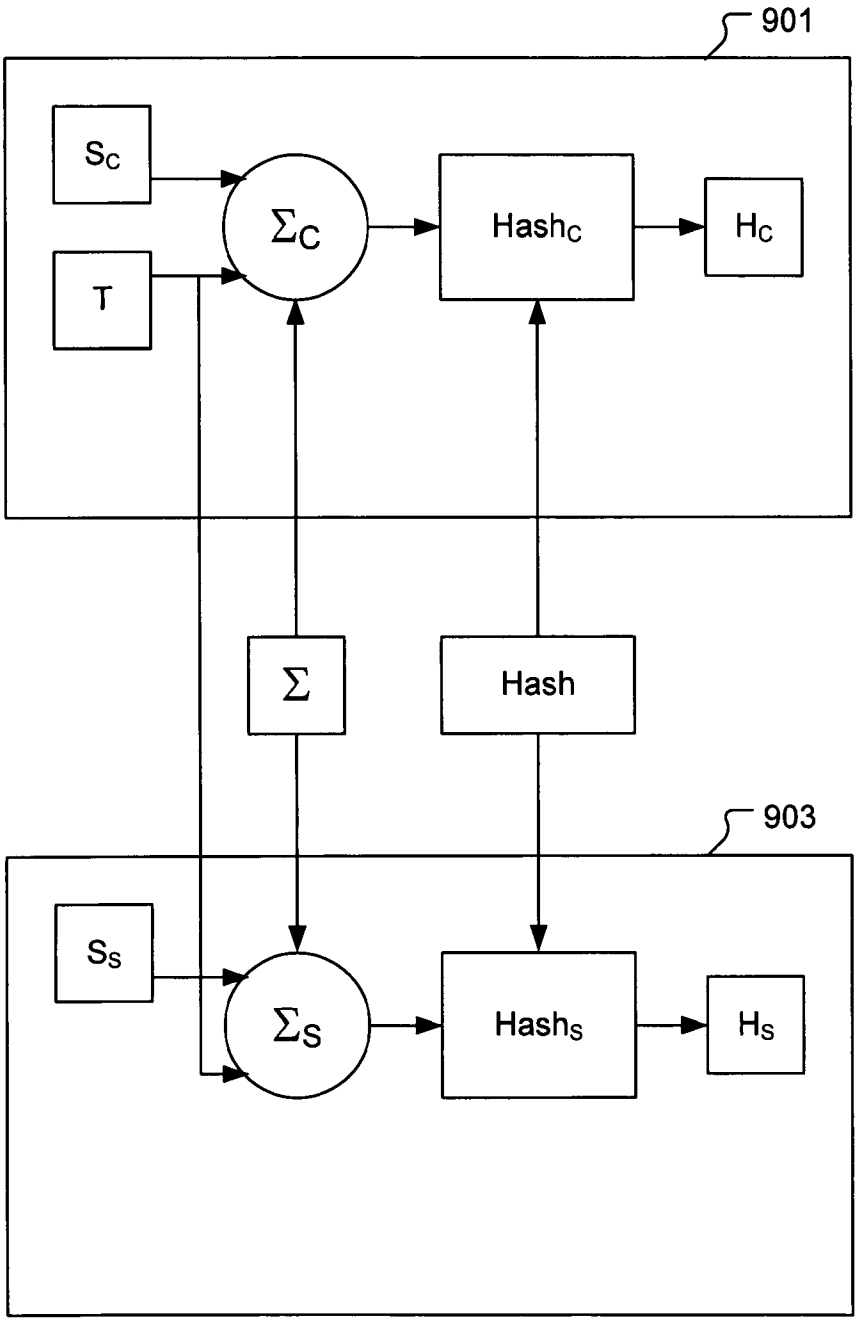

Fig. 13

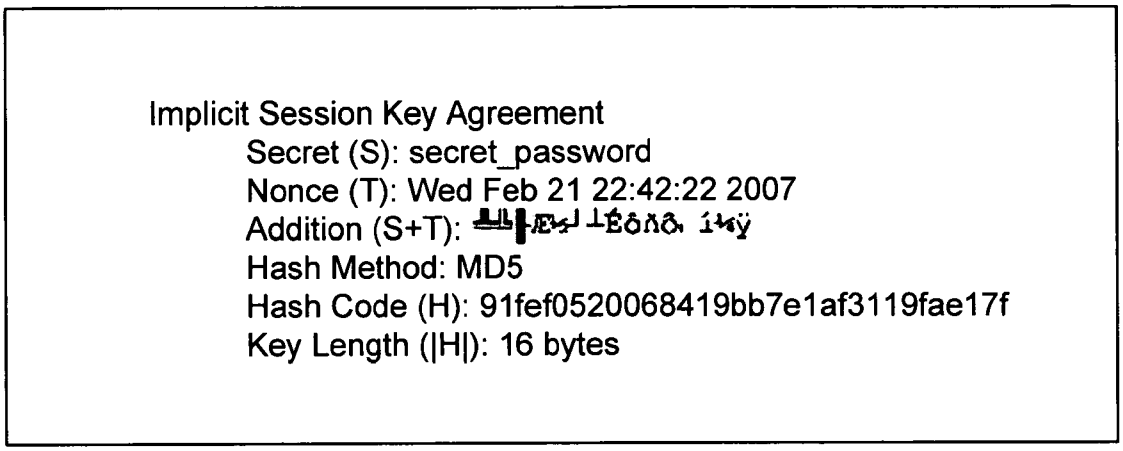
Implicit Session Key Agreement
Secret (S): secret_password
Nonce (T): Wed Feb 21 22:42:22 2007
Addition (S+T):
Hash Method: MD5
Hash Code (H): 91fef0520068419bb7e1af3119fae17f
Key Length (|H|): 16 bytes

Fig. 14

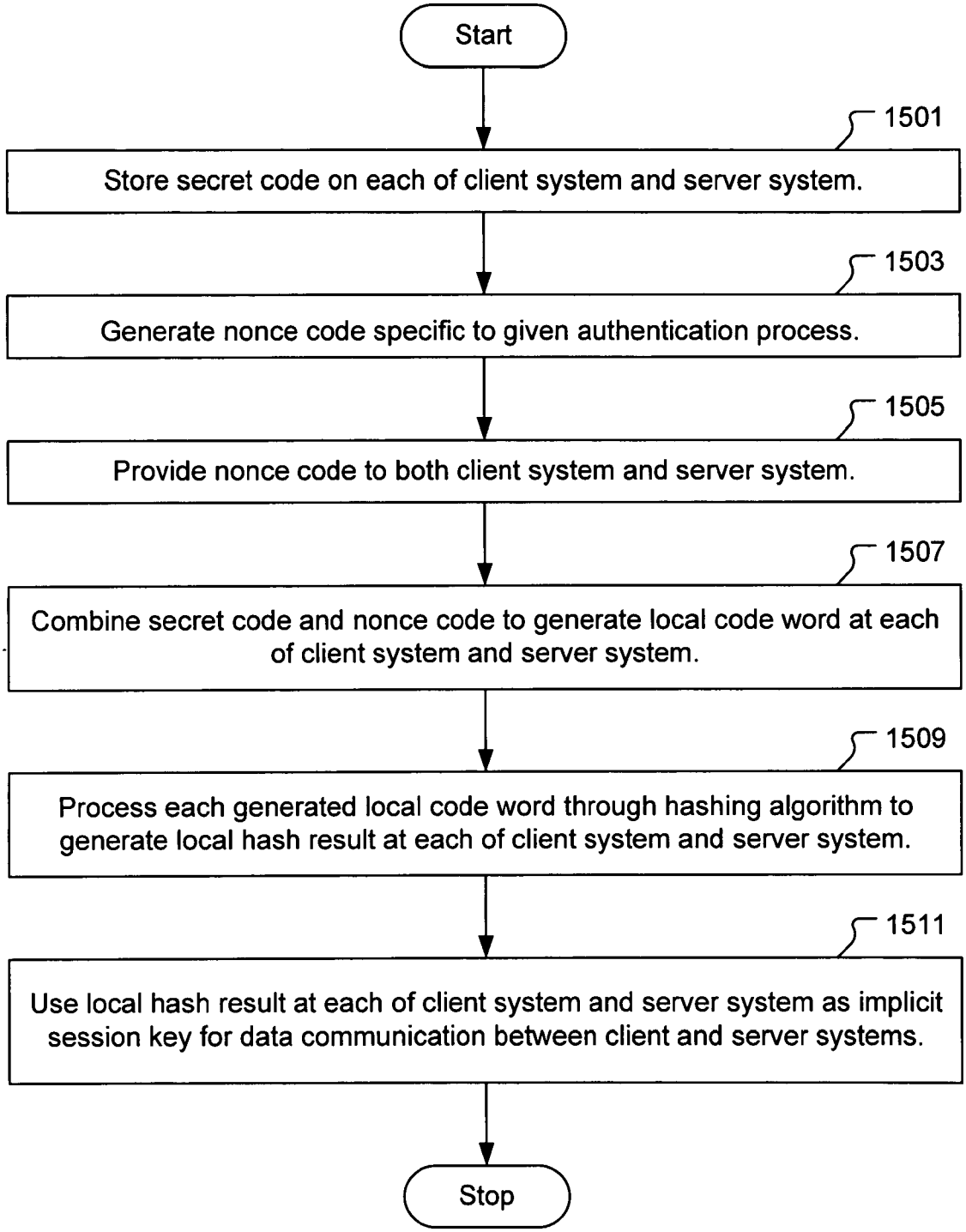
Start
1501
Store secret code on each of client system and server system.
1503
Generate nonce code specific to given authentication process.
1505
Provide nonce code to both client system and server system.
1507
Combine secret code and nonce code to generate local code word at each of client system and server system.
1509
Process each generated local code word through hashing algorithm to generate local hash result at each of client system and server system.
1511
Use local hash result at each of client system and server system as implicit session key for data communication between client and server systems.
Stop

Fig. 15