US00RE47296E

(19) **United States**

(12) **Reissued Patent**

Chen et al.

(10) **Patent Number: US RE47,296 E**

(45) **Date of Reissued Patent: *Mar. 12, 2019**

(54) **SYSTEM AND METHOD FOR AN ADAPTIVE TCP SYN COOKIE WITH TIME VALIDATION**

(71) Applicant: **A10 NETWORKS, INC.**, San Jose, CA (US)

(72) Inventors: **Lee Chen**, Saratoga, CA (US); **Ronald Wai Lun Szeto**, San Francisco, CA (US); **Shih-Tsung Hwang**, San Jose, CA (US)

(73) Assignee: **A10 NETWORKS, INC.**, San Jose, CA (US)

(*) Notice: This patent is subject to a terminal disclaimer.

(21) Appl. No.: **14/151,803**

(22) Filed: **Jan. 9, 2014**

Related U.S. Patent Documents

Reissue of:

(64) Patent No.: **7,675,854**
Issued: **Mar. 9, 2010**
Appl. No.: **11/358,245**
Filed: **Feb. 21, 2006**

U.S. Applications:

(63) Continuation of application No. 13/413,191, filed on Mar. 6, 2012, now Pat. No. Re. 44,701, which is an application for the reissue of Pat. No. 7,675,854.

(51) **Int. Cl.**

G01R 31/08	(2006.01)
H04L 12/801	(2013.01)
H04L 29/06	(2006.01)

(52) **U.S. Cl.**

CPC ***H04L 47/10*** (2013.01); ***H04L 63/1458*** (2013.01)

(58) **Field of Classification Search**

CPC H04L 47/10; H04L 63/1458

(Continued)

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,218,602 A 6/1993 Grant et al.
5,774,660 A 6/1998 Brendel et al.

(Continued)

FOREIGN PATENT DOCUMENTS

CN 1372662 A 10/2002
CN 1449618 10/2003

(Continued)

OTHER PUBLICATIONS

Cardellini et al., "Dynamic Load Balancing on Web-server Systems", IEEE Internet Computing, vol. 3, No. 3, pp. 28-39, May-Jun. 1999.

(Continued)

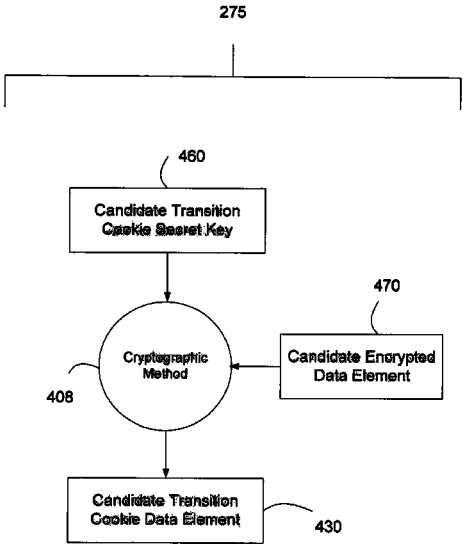
Primary Examiner — David E England

(74) *Attorney, Agent, or Firm* — Keith Kline; The Kline Law Firm PC

(57) **ABSTRACT**

Provided is a method and system for TCP SYN cookie validation. The method includes receiving a session SYN packet by a TCP session setup module of a host server, generating a transition cookie including a time value representing the actual time, sending a session SYN/ACK packet, including the transition cookie, in response to the received session SYN packet, receiving a session ACK packet, and determining whether a candidate transition cookie in the received session ACK packet comprises a time value representing a time within a predetermined time interval from the time the session ACK packet is received.

18 Claims, 10 Drawing Sheets



(58) **Field of Classification Search**
USPC 370/230.1; 709/227, 228; 726/22, 3, 6
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,862,339 A 1/1999 Bonnaure et al.
5,875,185 A 2/1999 Wang et al.
5,935,207 A 8/1999 Logue et al.
5,958,053 A * 9/1999 Denker 726/1
5,995,981 A 11/1999 Wikstrom
6,003,069 A 12/1999 Cavill
6,047,268 A * 4/2000 Bartoli et al. 713/153
6,075,783 A 6/2000 Voit
6,131,163 A 10/2000 Wiegel
6,219,706 B1 4/2001 Fan et al.
6,259,705 B1 7/2001 Takahashi et al.
6,321,338 B1 * 11/2001 Porras et al. 709/224
6,374,300 B2 4/2002 Masters
6,456,617 B1 9/2002 Oda et al.
6,459,682 B1 10/2002 Ellesson et al.
6,483,600 B1 11/2002 Schuster et al.
6,535,516 B1 3/2003 Leu et al.
6,578,066 B1 6/2003 Logan et al.
6,587,866 B1 7/2003 Modi et al.
6,600,738 B1 7/2003 Alperovich et al.
6,658,114 B1 12/2003 Farn et al.
6,748,414 B1 6/2004 Bournas
6,772,205 B1 8/2004 Lavian et al.
6,772,334 B1 * 8/2004 Glawitsch 713/153
6,779,017 B1 8/2004 Lamberton et al.
6,779,033 B1 * 8/2004 Watson et al. 709/227
6,804,224 B1 10/2004 Schuster et al.
6,952,728 B1 10/2005 Alles et al.
7,010,605 B1 3/2006 Dharmarajan
7,013,482 B1 3/2006 Krumel
7,058,718 B2 * 6/2006 Fontes et al. 709/228
7,069,438 B2 * 6/2006 Balabine et al. 713/168
7,076,555 B1 7/2006 Orman et al.
7,143,087 B2 11/2006 Fairweather
7,167,927 B2 1/2007 Philbrick et al.
7,181,524 B1 2/2007 Lele
7,218,722 B1 5/2007 Turner et al.
7,228,359 B1 6/2007 Monteiro
7,234,161 B1 6/2007 Maufer et al.
7,236,457 B2 6/2007 Joe
7,254,133 B2 * 8/2007 Govindarajan et al. 370/394
7,269,850 B2 9/2007 Govindarajan et al.
7,277,963 B2 10/2007 Dolson et al.
7,301,899 B2 * 11/2007 Goldstone 370/230
7,308,499 B2 12/2007 Chavez
7,310,686 B2 12/2007 Uysal
7,328,267 B1 2/2008 Bashyam et al.
7,334,232 B2 2/2008 Jacobs et al.
7,337,241 B2 2/2008 Boucher et al.
7,343,399 B2 3/2008 Hayball et al.
7,349,970 B2 3/2008 Clement et al.
7,370,353 B2 * 5/2008 Yang 726/11
7,373,500 B2 5/2008 Ramelson et al.
7,391,725 B2 * 6/2008 Huitema et al. 370/230.1
7,398,317 B2 * 7/2008 Chen et al. 709/229
7,423,977 B1 9/2008 Joshi
7,430,755 B1 * 9/2008 Hughes et al. 726/3
7,463,648 B1 12/2008 Eppstein et al.
7,467,202 B2 12/2008 Savchuk
7,472,190 B2 12/2008 Robinson
7,492,766 B2 2/2009 Cabeca et al.
7,506,360 B1 * 3/2009 Wilkinson et al. 709/223
7,509,369 B1 3/2009 Tormasov
7,512,980 B2 * 3/2009 Copeland et al. 726/22
7,533,409 B2 5/2009 Keane et al.
7,552,323 B2 * 6/2009 Shay H04L 63/02 713/153
7,584,262 B1 9/2009 Wang et al.
7,584,301 B1 9/2009 Joshi
7,590,736 B2 9/2009 Hydrie et al.
7,610,622 B2 * 10/2009 Touitou H04L 63/168 726/22
7,613,193 B2 * 11/2009 Swami et al. 370/395.52
7,613,822 B2 11/2009 Joy et al.
7,673,072 B2 3/2010 Boucher et al.
7,675,854 B2 * 3/2010 Chen et al. 370/230.1
7,703,102 B1 4/2010 Eppstein et al.
7,707,295 B1 4/2010 Szeto et al.
7,711,790 B1 * 5/2010 Barrett et al. 709/217
7,733,866 B2 * 6/2010 Mishra H04L 12/2854 370/349
7,747,748 B2 6/2010 Allen
7,765,328 B2 7/2010 Bryers et al.
7,792,113 B1 9/2010 Foschiano et al.
7,808,994 B1 10/2010 Vinokour et al.
7,826,487 B1 * 11/2010 Mukerji et al. 370/477
7,881,215 B1 2/2011 Daigle et al.
7,948,952 B2 5/2011 Hurtta et al.
7,965,727 B2 6/2011 Sakata et al.
7,970,934 B1 6/2011 Patel
7,979,694 B2 * 7/2011 Touitou H04L 63/1458 709/229
7,983,258 B1 7/2011 Ruben et al.
7,990,847 B1 8/2011 Leroy et al.
7,991,859 B1 8/2011 Miller et al.
7,992,201 B2 8/2011 Aldridge et al.
8,019,870 B1 9/2011 Eppstein et al.
8,032,634 B1 10/2011 Eppstein et al.
8,081,640 B2 12/2011 Ozawa et al.
8,090,866 B1 1/2012 Bashyam et al.
8,099,492 B2 1/2012 Dahlin et al.
8,116,312 B2 2/2012 Riddoch et al.
8,122,116 B2 2/2012 Matsunaga et al.
8,151,019 B1 4/2012 Le et al.
8,179,809 B1 5/2012 Eppstein et al.
8,185,651 B2 5/2012 Moran et al.
8,191,106 B2 5/2012 Choyi et al.
8,224,971 B1 7/2012 Miller et al.
8,261,339 B2 9/2012 Aldridge et al.
8,266,235 B2 9/2012 Jalan et al.
8,296,434 B1 10/2012 Miller et al.
8,312,507 B2 11/2012 Chen et al.
8,379,515 B1 2/2013 Mukerji
8,499,093 B2 7/2013 Grosser et al.
8,539,075 B2 9/2013 Bali et al.
8,554,929 B1 10/2013 Szeto et al.
8,559,437 B2 * 10/2013 Mishra H04L 12/2854 370/349
8,560,693 B1 10/2013 Wang et al.
8,584,199 B1 11/2013 Chen et al.
8,595,791 B1 11/2013 Chen et al.
RE44,701 E * 1/2014 Chen et al. 370/230.1
8,675,488 B1 3/2014 Sidebottom et al.
8,681,610 B1 3/2014 Mukerji
8,750,164 B2 6/2014 Casado et al.
8,782,221 B2 7/2014 Han
8,813,180 B1 8/2014 Chen et al.
8,826,372 B1 9/2014 Chen et al.
8,879,427 B2 11/2014 Krumel
8,885,463 B1 11/2014 Medved et al.
8,897,154 B2 11/2014 Jalan et al.
8,965,957 B2 2/2015 Barros
8,977,749 B1 3/2015 Han
8,990,262 B2 3/2015 Chen et al.
9,094,364 B2 7/2015 Jalan et al.
9,106,561 B2 8/2015 Jalan et al.
9,137,301 B1 9/2015 Dunlap et al.
9,154,577 B2 10/2015 Jalan et al.
9,154,584 B1 10/2015 Han
9,215,275 B2 12/2015 Kannan et al.
9,219,751 B1 12/2015 Chen et al.
9,253,152 B1 2/2016 Chen et al.
9,270,705 B1 2/2016 Chen et al.
9,270,774 B2 2/2016 Jalan et al.
9,338,225 B2 5/2016 Jalan et al.
9,350,744 B2 5/2016 Chen et al.
9,356,910 B2 5/2016 Chen et al.
9,386,088 B2 7/2016 Zheng et al.
9,531,846 B2 12/2016 Han et al.

(56) **References Cited**

U.S. PATENT DOCUMENTS

2001/0042200 A1* 11/2001 Lamberton H04L 63/126 713/151
2001/0049741 A1 12/2001 Skene et al.
2002/0026515 A1 2/2002 Michielsens et al.
2002/0032777 A1 3/2002 Kawata et al.
2002/0032799 A1 3/2002 Wiedeman et al.
2002/0078164 A1 6/2002 Reinschmidt
2002/0091844 A1 7/2002 Craft et al.
2002/0103916 A1* 8/2002 Chen et al. 709/229
2002/0133491 A1 9/2002 Sim et al.
2002/0138618 A1 9/2002 Szabo
2002/0141386 A1 10/2002 Minert et al.
2002/0143991 A1 10/2002 Chow et al.
2002/0178259 A1 11/2002 Doyle et al.
2002/0188678 A1 12/2002 Edecker et al.
2002/0191575 A1 12/2002 Kalavade et al.
2002/0194335 A1 12/2002 Maynard
2002/0194350 A1 12/2002 Lu et al.
2003/0009591 A1 1/2003 Hayball et al.
2003/0014544 A1 1/2003 Pettey
2003/0023711 A1 1/2003 Parmar et al.
2003/0023873 A1 1/2003 Ben-Itzhak
2003/0035409 A1 2/2003 Wang et al.
2003/0035420 A1 2/2003 Niu
2003/0061506 A1 3/2003 Cooper et al.
2003/0091028 A1 5/2003 Chang et al.
2003/0131245 A1 7/2003 Linderman
2003/0135625 A1* 7/2003 Fontes et al. 709/228
2003/0195962 A1 10/2003 Kikuchi et al.
2004/0010545 A1 1/2004 Pandya
2004/0062246 A1 4/2004 Boucher et al.
2004/0073703 A1 4/2004 Boucher et al.
2004/0078419 A1 4/2004 Ferrari et al.
2004/0078480 A1 4/2004 Boucher et al.
2004/0103315 A1 5/2004 Cooper et al.
2004/0111516 A1 6/2004 Cain
2004/0128312 A1 7/2004 Shalabi et al.
2004/0139057 A1 7/2004 Hirata et al.
2004/0139108 A1 7/2004 Tang et al.
2004/0141005 A1 7/2004 Banatwala et al.
2004/0143599 A1 7/2004 Shalabi et al.
2004/0187032 A1 9/2004 Gels et al.
2004/0199616 A1 10/2004 Karhu
2004/0199646 A1 10/2004 Susai et al.
2004/0202182 A1 10/2004 Lund et al.
2004/0210623 A1 10/2004 Hydrie et al.
2004/0210663 A1 10/2004 Phillips et al.
2004/0213158 A1 10/2004 Collett et al.
2004/0250059 A1 12/2004 Ramelson et al.
2004/0268358 A1 12/2004 Darling et al.
2005/0005207 A1 1/2005 Herneque
2005/0009520 A1 1/2005 Herrero et al.
2005/0021848 A1 1/2005 Jorgenson
2005/0027862 A1 2/2005 Nguyen et al.
2005/0036501 A1 2/2005 Chung et al.
2005/0036511 A1 2/2005 Baratakke et al.
2005/0039033 A1 2/2005 Meyers et al.
2005/0044270 A1 2/2005 Grove et al.
2005/0074013 A1 4/2005 Hershey et al.
2005/0080890 A1 4/2005 Yang et al.
2005/0102400 A1 5/2005 Nakahara et al.
2005/0125276 A1 6/2005 Rusu
2005/0163073 A1 7/2005 Heller et al.
2005/0198335 A1 9/2005 Brown et al.
2005/0213586 A1 9/2005 Cyganski et al.
2005/0240989 A1* 10/2005 Kim et al. 726/11
2005/0249225 A1 11/2005 Singhal
2005/0259586 A1 11/2005 Hafid et al.
2005/0281190 A1 12/2005 McGee et al.
2006/0023721 A1* 2/2006 Miyake et al. 370/395.2
2006/0036610 A1 2/2006 Wang
2006/0036733 A1 2/2006 Fujimoto et al.
2006/0041745 A1 2/2006 Parnes
2006/0064478 A1 3/2006 Sirkin
2006/0069774 A1 3/2006 Chen et al.
2006/0069804 A1* 3/2006 Miyake et al. 709/237
2006/0077926 A1 4/2006 Rune
2006/0092950 A1 5/2006 Arregoces et al.
2006/0098645 A1 5/2006 Walkin
2006/0112170 A1 5/2006 Sirkin
2006/0164978 A1 7/2006 Werner et al.
2006/0168319 A1 7/2006 Trossen
2006/0187901 A1 8/2006 Cortes et al.
2006/0190997 A1 8/2006 Mahajani et al.
2006/0209789 A1 9/2006 Gupta et al.
2006/0230129 A1* 10/2006 Swami H04W 12/12 709/223
2006/0233100 A1 10/2006 Luft et al.
2006/0251057 A1 11/2006 Kwon et al.
2006/0277303 A1 12/2006 Hegde et al.
2006/0280121 A1* 12/2006 Matoba 370/235
2007/0019543 A1* 1/2007 Wei et al. 370/229
2007/0022479 A1 1/2007 Sikdar et al.
2007/0076653 A1 4/2007 Park et al.
2007/0086382 A1 4/2007 Narayanan et al.
2007/0094396 A1 4/2007 Takano et al.
2007/0118881 A1 5/2007 Mitchell et al.
2007/0124502 A1 5/2007 Li
2007/0156919 A1 7/2007 Potti et al.
2007/0165622 A1 7/2007 O'Rourke et al.
2007/0180119 A1 8/2007 Khivesara et al.
2007/0185998 A1 8/2007 Touitou et al.
2007/0195792 A1* 8/2007 Chen et al. 370/395.52
2007/0230337 A1 10/2007 Igarashi et al.
2007/0242738 A1 10/2007 Park et al.
2007/0243879 A1 10/2007 Park et al.
2007/0245090 A1 10/2007 King et al.
2007/0248009 A1 10/2007 Petersen
2007/0259673 A1 11/2007 Willars et al.
2007/0283429 A1 12/2007 Chen et al.
2007/0286077 A1 12/2007 Wu
2007/0288247 A1 12/2007 Mackay
2007/0294209 A1 12/2007 Strub et al.
2008/0016161 A1 1/2008 Tsirtsis et al.
2008/0031263 A1 2/2008 Ervin et al.
2008/0076432 A1 3/2008 Senarath et al.
2008/0101396 A1 5/2008 Miyata
2008/0109452 A1 5/2008 Patterson
2008/0109870 A1 5/2008 Sherlock et al.
2008/0120129 A1 5/2008 Seubert et al.
2008/0134332 A1 6/2008 Keohane et al.
2008/0162679 A1 7/2008 Maher et al.
2008/0225722 A1 9/2008 Khemani et al.
2008/0228781 A1 9/2008 Chen et al.
2008/0250099 A1 10/2008 Shen et al.
2008/0253390 A1 10/2008 Das et al.
2008/0263209 A1 10/2008 Pisharody et al.
2008/0271130 A1 10/2008 Ramamoorthy
2008/0282254 A1 11/2008 Blander et al.
2008/0291911 A1 11/2008 Lee et al.
2008/0298303 A1 12/2008 Tsirtsis
2009/0024722 A1 1/2009 Sethuraman et al.
2009/0031415 A1 1/2009 Aldridge et al.
2009/0049198 A1 2/2009 Blinn et al.
2009/0070470 A1 3/2009 Bauman et al.
2009/0077651 A1 3/2009 Poeluev
2009/0092124 A1 4/2009 Singhal et al.
2009/0106830 A1 4/2009 Maher
2009/0138606 A1 5/2009 Moran et al.
2009/0138945 A1 5/2009 Savchuk
2009/0141634 A1 6/2009 Rothstein et al.
2009/0164614 A1 6/2009 Christian et al.
2009/0172093 A1 7/2009 Matsubara
2009/0213858 A1 8/2009 Dolganow et al.
2009/0222583 A1 9/2009 Josefsberg et al.
2009/0227228 A1 9/2009 Hu et al.
2009/0228547 A1 9/2009 Miyaoka et al.
2009/0262741 A1 10/2009 Jungck et al.
2009/0271472 A1 10/2009 Scheifler et al.
2009/0285196 A1 11/2009 Lee et al.
2009/0313379 A1 12/2009 Rydnell et al.
2010/0008229 A1 1/2010 Bi et al.
2010/0023621 A1 1/2010 Ezolt et al.
2010/0036952 A1 2/2010 Hazlewood et al.

(56) **References Cited**

U.S. PATENT DOCUMENTS

2010/0042869 A1 2/2010 Szabo et al.
2010/0054139 A1 3/2010 Chun et al.
2010/0061319 A1 3/2010 Aso et al.
2010/0064008 A1 3/2010 Yan et al.
2010/0082787 A1 4/2010 Kommula et al.
2010/0083076 A1 4/2010 Ushiyama
2010/0094985 A1 4/2010 Abu-Samaha et al.
2010/0095018 A1 4/2010 Khemani et al.
2010/0098417 A1 4/2010 Tse-Au
2010/0106833 A1 4/2010 Banerjee et al.
2010/0106854 A1 4/2010 Kim et al.
2010/0128606 A1 5/2010 Patel et al.
2010/0162378 A1 6/2010 Jayawardena et al.
2010/0205310 A1 8/2010 Altshuler et al.
2010/0210265 A1 8/2010 Borzsei et al.
2010/0217793 A1 8/2010 Preiss
2010/0217819 A1 8/2010 Chen et al.
2010/0223630 A1 9/2010 Degenkolb et al.
2010/0228819 A1 9/2010 Wei
2010/0235507 A1 9/2010 Szeto et al.
2010/0235522 A1 9/2010 Chen et al.
2010/0235880 A1 9/2010 Chen et al.
2010/0238828 A1 9/2010 Russell
2010/0265824 A1 10/2010 Chao et al.
2010/0268814 A1 10/2010 Cross et al.
2010/0293296 A1 11/2010 Hsu et al.
2010/0312740 A1 12/2010 Clemm et al.
2010/0318631 A1 12/2010 Shukla
2010/0322252 A1 12/2010 Suganthi et al.
2010/0330971 A1 12/2010 Selitser et al.
2010/0333101 A1 12/2010 Pope et al.
2011/0007652 A1 1/2011 Bai
2011/0019550 A1 1/2011 Bryers et al.
2011/0023071 A1 1/2011 Li et al.
2011/0029599 A1 2/2011 Pulleyn et al.
2011/0032941 A1 2/2011 Quach et al.
2011/0040826 A1 2/2011 Chadzelek et al.
2011/0047294 A1 2/2011 Singh et al.
2011/0060831 A1 3/2011 Ishii et al.
2011/0083174 A1 4/2011 Aldridge et al.
2011/0093522 A1 4/2011 Chen et al.
2011/0099403 A1 4/2011 Miyata et al.
2011/0099623 A1 4/2011 Garrard et al.
2011/0110294 A1 5/2011 Valluri et al.
2011/0145324 A1 6/2011 Reinart et al.
2011/0149879 A1 6/2011 Noriega et al.
2011/0153834 A1 6/2011 Bharrat
2011/0178985 A1 7/2011 San Martin Arribas et al.
2011/0185073 A1 7/2011 Jagadeeswaran et al.
2011/0191773 A1 8/2011 Pavel et al.
2011/0196971 A1 8/2011 Reguraman et al.
2011/0276695 A1 11/2011 Maldaner
2011/0276982 A1 11/2011 Nakayama et al.
2011/0289496 A1 11/2011 Steer
2011/0292939 A1 12/2011 Subramaian et al.
2011/0302256 A1 12/2011 Sureshehandra et al.
2011/0307541 A1 12/2011 Walsh et al.
2012/0008495 A1 1/2012 Shen et al.
2012/0023231 A1 1/2012 Ueno
2012/0026897 A1 2/2012 Guichard et al.
2012/0030341 A1 2/2012 Jensen et al.
2012/0066371 A1 3/2012 Patel et al.
2012/0084419 A1 4/2012 Kannan et al.
2012/0084460 A1 4/2012 McGinnity et al.
2012/0106355 A1 5/2012 Ludwig
2012/0117382 A1 5/2012 Larson et al.
2012/0117571 A1 5/2012 Davis et al.
2012/0144014 A1 6/2012 Natham et al.
2012/0144015 A1 6/2012 Jalan et al.
2012/0151353 A1 6/2012 Joanny
2012/0170548 A1 7/2012 Rajagopalan et al.
2012/0173759 A1 7/2012 Agarwal et al.
2012/0179770 A1 7/2012 Jalan et al.
2012/0191839 A1 7/2012 Maynard
2012/0215910 A1 8/2012 Wada
2012/0239792 A1 9/2012 Banerjee et al.
2012/0240185 A1 9/2012 Kapoor et al.
2012/0290727 A1 11/2012 Tivig
2012/0297046 A1 11/2012 Raja et al.
2012/0311116 A1 12/2012 Jalan et al.
2013/0046876 A1 2/2013 Narayana et al.
2013/0058335 A1 3/2013 Koponen et al.
2013/0074177 A1 3/2013 Varadhan et al.
2013/0083725 A1 4/2013 Mallya et al.
2013/0100958 A1 4/2013 Jalan et al.
2013/0124713 A1 5/2013 Feinberg et al.
2013/0135996 A1 5/2013 Torres et al.
2013/0136139 A1 5/2013 Zheng et al.
2013/0148500 A1 6/2013 Sonoda et al.
2013/0166762 A1 6/2013 Jalan et al.
2013/0173795 A1 7/2013 McPherson
2013/0176854 A1 7/2013 Chisu et al.
2013/0191486 A1 7/2013 Someya et al.
2013/0198385 A1 8/2013 Han et al.
2013/0250765 A1 9/2013 Ehsan et al.
2013/0258846 A1 10/2013 Damola
2013/0282791 A1 10/2013 Kruglick
2014/0012972 A1 1/2014 Han
2014/0089500 A1 3/2014 Sankar et al.
2014/0164617 A1 6/2014 Jalan et al.
2014/0169168 A1 6/2014 Jalan et al.
2014/0207845 A1 7/2014 Han et al.
2014/0258465 A1 9/2014 Li
2014/0258536 A1 9/2014 Chiong
2014/0269728 A1 9/2014 Jalan et al.
2014/0286313 A1 9/2014 Fu et al.
2014/0298091 A1 10/2014 Carlen et al.
2014/0330982 A1 11/2014 Jalan et al.
2014/0334485 A1 11/2014 Jain et al.
2014/0359052 A1 12/2014 Joachimpillai et al.
2015/0026794 A1 1/2015 Zuk et al.
2015/0039671 A1 2/2015 Jalan et al.
2015/0156223 A1 6/2015 Xu et al.
2015/0215436 A1 7/2015 Kancherla
2015/0237173 A1 8/2015 Virkki et al.
2015/0244566 A1 8/2015 Puimedon
2015/0281087 A1 10/2015 Jalan et al.
2015/0281104 A1 10/2015 Golshan et al.
2015/0296058 A1 10/2015 Jalan et al.
2015/0312092 A1 10/2015 Golshan et al.
2015/0312268 A1 10/2015 Ray
2015/0333988 A1 11/2015 Jalan et al.
2015/0350048 A1 12/2015 Sampat et al.
2015/0350379 A1 12/2015 Jalan et al.
2016/0014052 A1 1/2016 Han
2016/0014126 A1 1/2016 Jalan et al.
2016/0036778 A1 2/2016 Chen et al.
2016/0042014 A1 2/2016 Jalan et al.
2016/0043901 A1 2/2016 Sankar et al.
2016/0044095 A1 2/2016 Sankar et al.
2016/0050233 A1 2/2016 Chen et al.
2016/0088074 A1 3/2016 Kannan et al.
2016/0105395 A1 4/2016 Chen et al.
2016/0105446 A1 4/2016 Chen et al.
2016/0119382 A1 4/2016 Chen et al.
2016/0156708 A1 6/2016 Jalan et al.
2016/0173579 A1 6/2016 Jalan et al.
2017/0048107 A1 2/2017 Dosovitsky et al.
2017/0048356 A1 2/2017 Thompson et al.

FOREIGN PATENT DOCUMENTS

CN 1473300 A 2/2004
CN 1529460 9/2004
CN 1575582 2/2005
CN 1714545 A 12/2005
CN 1725702 1/2006
CN 1910869 A 2/2007
CN 101004740 A 7/2007
CN 101094225 12/2007
CN 101163336 A 4/2008
CN 101169785 A 4/2008
CN 101189598 5/2008
CN 101193089 A 6/2008

(56) **References Cited**

FOREIGN PATENT DOCUMENTS

CN	101247349	A	8/2008
CN	101261644	A	9/2008
CN	101442425	A	5/2009
CN	101495993	A	7/2009
CN	101682532	A	3/2010
CN	101878663	A	11/2010
CN	102123156	A	7/2011
CN	102143075	A	8/2011
CN	102546590		7/2012
CN	102571742		7/2012
CN	102577252		7/2012
CN	102918801		2/2013
CN	103533018		1/2014
CN	103944954		7/2014
CN	104040990		9/2014
CN	104067569		9/2014
CN	104106241		10/2014
CN	104137491		11/2014
CN	104796396	A	7/2015
CN	102577252	B	3/2016
CN	102918801	B	5/2016
EP	1209876		5/2002
EP	1770915		4/2007
EP	1885096		2/2008
EP	02296313		3/2011
EP	2577910		4/2013
EP	2622795		8/2013
EP	2647174		10/2013
EP	2760170		7/2014
EP	2772026		9/2014
EP	2901308	A2	8/2015
EP	2760170	B1	12/2015
HK	1182560		11/2013
HK	1183569		12/2013
HK	1183996		1/2014
HK	1189438		6/2014
HK	1198565	A1	5/2015
HK	1198848	A1	6/2015
HK	1199153	A1	6/2015
HK	1199779	A1	7/2015
HK	1200617	A1	8/2015
IN	39/2015		9/2015
IN	CHE2014	A	7/2016
JP	H09-097233		4/1997
JP	1999096128		4/1999
JP	H11-338836		10/1999
JP	2000276432	A	10/2000
JP	2000307634	A	11/2000
JP	2001051859	A	2/2001
JP	2001298449	A	10/2001
JP	2002091936	A	3/2002
JP	2003141068	A	5/2003
JP	2003186776	A	7/2003
JP	2005141441	A	6/2005
JP	2006332825	A	12/2006
JP	2008040718	A	2/2008
JP	2009500731	A	1/2009
JP	2013528330		7/2013
JP	2014504484	A	2/2014
JP	2014143686		8/2014
JP	2015507380	A	3/2015
JP	5855663	B2	12/2015
JP	5906263	B	3/2016
JP	5913609	B2	4/2016
KR	10-0830413		5/2008
KR	1020120117461		8/2013
KR	101576585	B1	12/2015
TW	269763	B	2/1996
TW	425821	B	3/2001
TW	444478	B	7/2001
WO	WO2001013228		2/2001
WO	WO2001014990		3/2001
WO	WO2001045349		6/2001
WO	WO2003103237		12/2003
WO	WO2004084085	A1	9/2004
WO	WO2006098033	A1	9/2006
WO	WO2008053954		5/2008
WO	WO2008078593	A1	7/2008
WO	WO2011049770		4/2011
WO	WO2011079381	A1	7/2011
WO	WO2011149796		12/2011
WO	WO2012050747		4/2012
WO	WO2012075237		6/2012
WO	WO2012083264	A2	6/2012
WO	WO2012097015	A2	7/2012
WO	WO2013070391		5/2013
WO	WO2013081952		6/2013
WO	WO2013096019		6/2013
WO	WO2013112492		8/2013
WO	WO2014031046	A1	2/2014
WO	WO2014052099		4/2014
WO	WO2014088741		6/2014
WO	WO2014093829		6/2014
WO	WO2014138483		9/2014
WO	WO2014144837		9/2014
WO	WO2014179753		11/2014
WO	WO2015153020	A1	10/2015
WO	WO2015164026	A1	10/2015

OTHER PUBLICATIONS

Spatscheck et al., "Optimizing TCP Forwarder Performance", IEEE/ACM Transactions on Networking, vol. 8, No. 2, Apr. 2000.

Kjaer et al. "Resource allocation and disturbance rejection in web servers using SLAs and virtualized servers", IEEE Transactions on Network and Service Management, IEEE, US, vol. 6, No. 4, Dec. 1, 2009.

Sharifian et al. "An approximation-based load-balancing algorithm with admission control for cluster web servers with dynamic workloads", The Journal of Supercomputing, Kluwer Academic Publishers, BO, vol. 53, No. 3, Jul. 3, 2009.

Hunt et al. NetDispatcher: A TCP Connection Router, IBM Research Report RC 20853 May 19, 1997.

Noguchi, "Realizing the Highest Level "Layer 7" Switch"= Totally Managing Network Resources, Applications, and Users =, Computer & Network LAN, Jan. 1, 2000, vol. 18, No. 1, p. 109-112.

Takahashi, "The Fundamentals of the Windows Network: Understanding the Mystery of the Windows Network from the Basics", Network Magazine, Jul. 1, 2006, vol. 11, No. 7, p. 32-35.

Ohnuma, "AppSwitch: 7th Layer Switch Provided with Full Setup and Report Tools", Interop Magazine, Jun. 1, 2000, vol. 10, No. 6, p. 148-150.

Koike et al., "Transport Middleware for Network-Based Control," IEICE Technical Report, Jun. 22, 2000, vol. 100, No. 53, pp. 13-18.

Yamamoto et al., "Performance Evaluation of Window Size in Proxy-based TCP for Multi-hop Wireless Networks," IPSJ SIG Technical Reports, May 15, 2008, vol. 2008, No. 44, pp. 109-114.

Abe et al., "Adaptive Split Connection Schemes in Advanced Relay Nodes," IEICE Technical Report, Feb. 22, 2010, vol. 109, No. 438, pp. 25-30.

Gite, Vivek, "Linux Tune Network Stack (Buffers Size) To Increase Networking Performance," accessed Apr. 13, 2016 at URL: «http://www.cyberciti.biz/faq/linux-tcp-tuning/», Jul. 8, 2009, 24 pages.

"Tcp—TCP Protocol", Linux Programmer's Manual, accessed Apr. 13, 2016 at URL: «https://www.freebsd.org/cgi/man.cgi?query=tcp&apropos=0&sektion=7&manpath=SuSE+Linux%2Fi386+11.0&format=asci», Nov. 25, 2007, 11 pages.

"Enhanced Interior Gateway Routing Protocol", Cisco, Document ID 16406, Sep. 9, 2005 update, 43 pages.

Crotti, Manuel et al., "Detecting HTTP Tunnels with Statistical Mechanisms", IEEE International Conference on communications, Jun. 24-28, 2007, pp. 6162-6168.

Haruyama, Takahiro et al., "Dial-to-Connect VPN System for Remote DLNA Communication", IEEE Consumer Communications and Networking Conference, CCNC 2008. 5th IEEE, Jan. 10-12, 2008, pp. 1224-1225.

(56) **References Cited**

OTHER PUBLICATIONS

Chen, Jianhua et al., "SSL/TLS-based Secure Tunnel Gateway System Design and Implementation", IEEE International Workshop on Anti-counterfeiting, Security, Identification, Apr. 16-18, 2007, pp. 258-261.
"EIGRP MPLS VPN PE-CE Site of Origin (SoO)", Cisco Systems, Feb. 28, 2006, 14 pages.

* cited by examiner

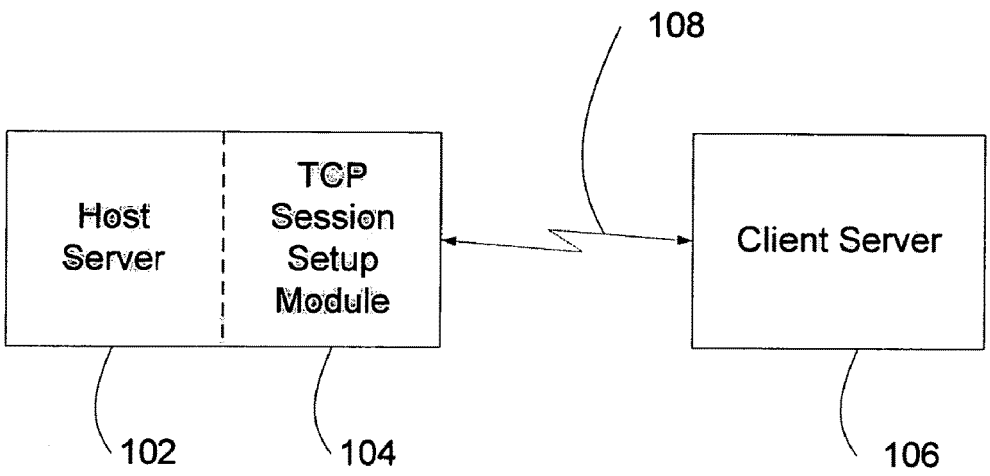

108
Host Server
TCP Session Setup Module
Client Server
102
104
106

Figure 1

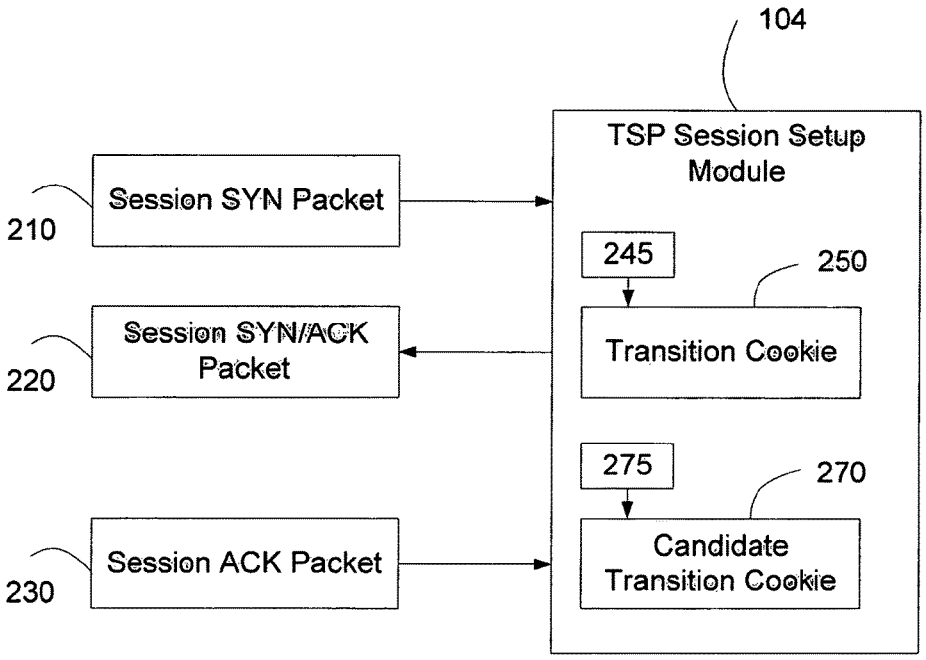

104
TSP Session Setup Module
Session SYN Packet
210
245
250
Session SYN/ACK Packet
220
Transition Cookie
275
270
Session ACK Packet
230
Candidate Transition Cookie

Figure 2

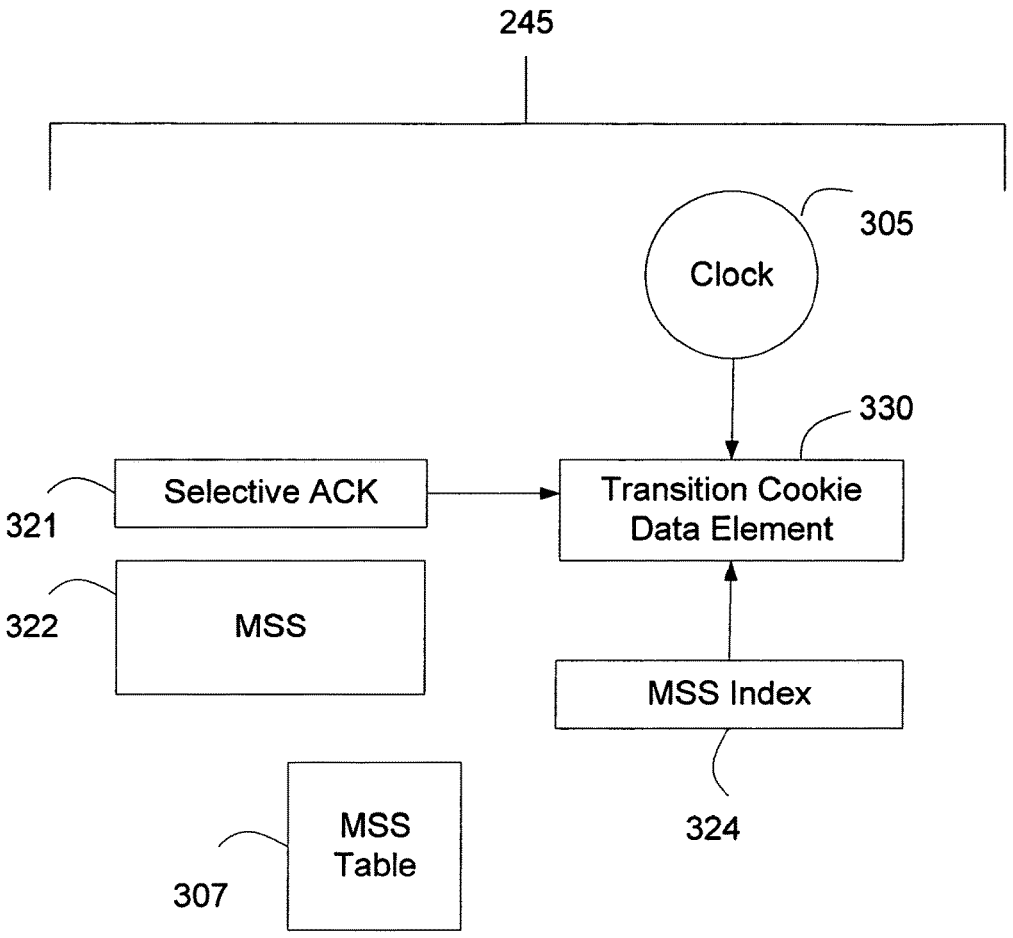

245
Clock
305
330
Selective ACK
321
Transition Cookie Data Element
MSS
322
MSS Index
324
MSS Table
307

Figure 3a

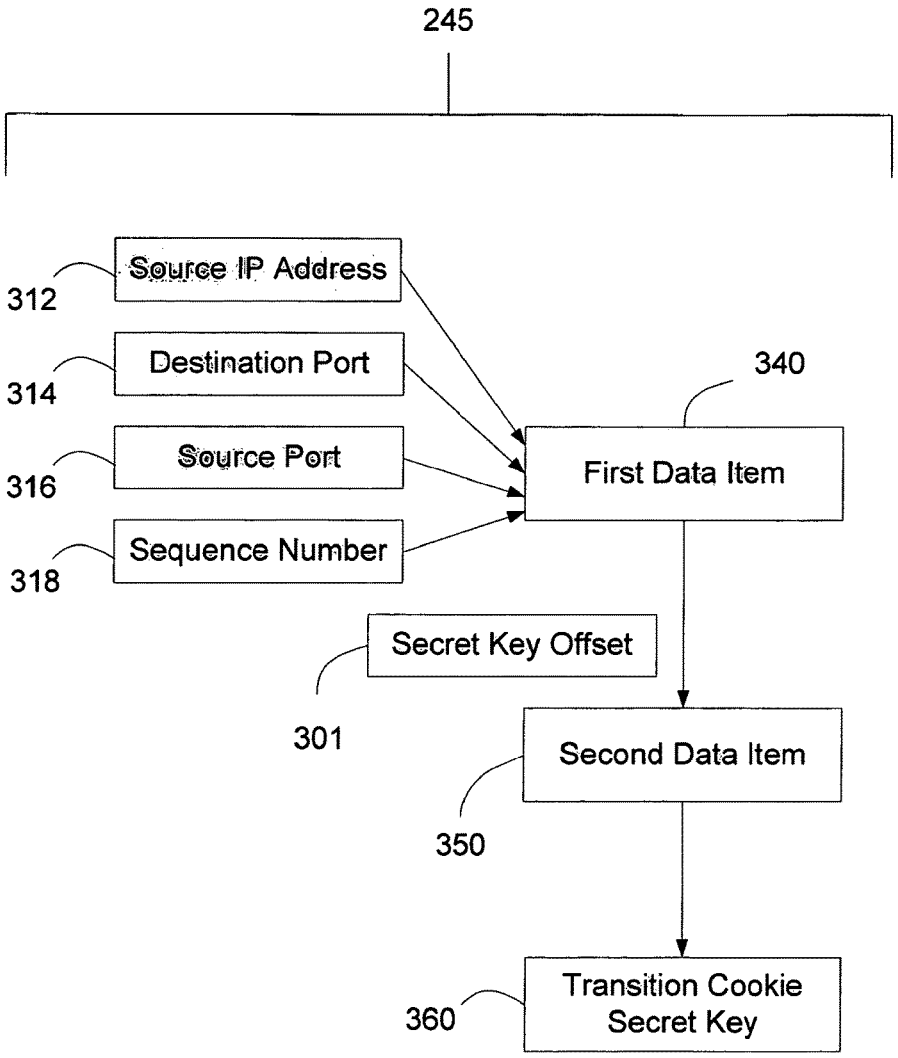

245
Source IP Address
312
Destination Port
314
Source Port
316
Sequence Number
318
340
First Data Item
Secret Key Offset
301
Second Data Item
350
Transition Cookie Secret Key
360

Figure 3b

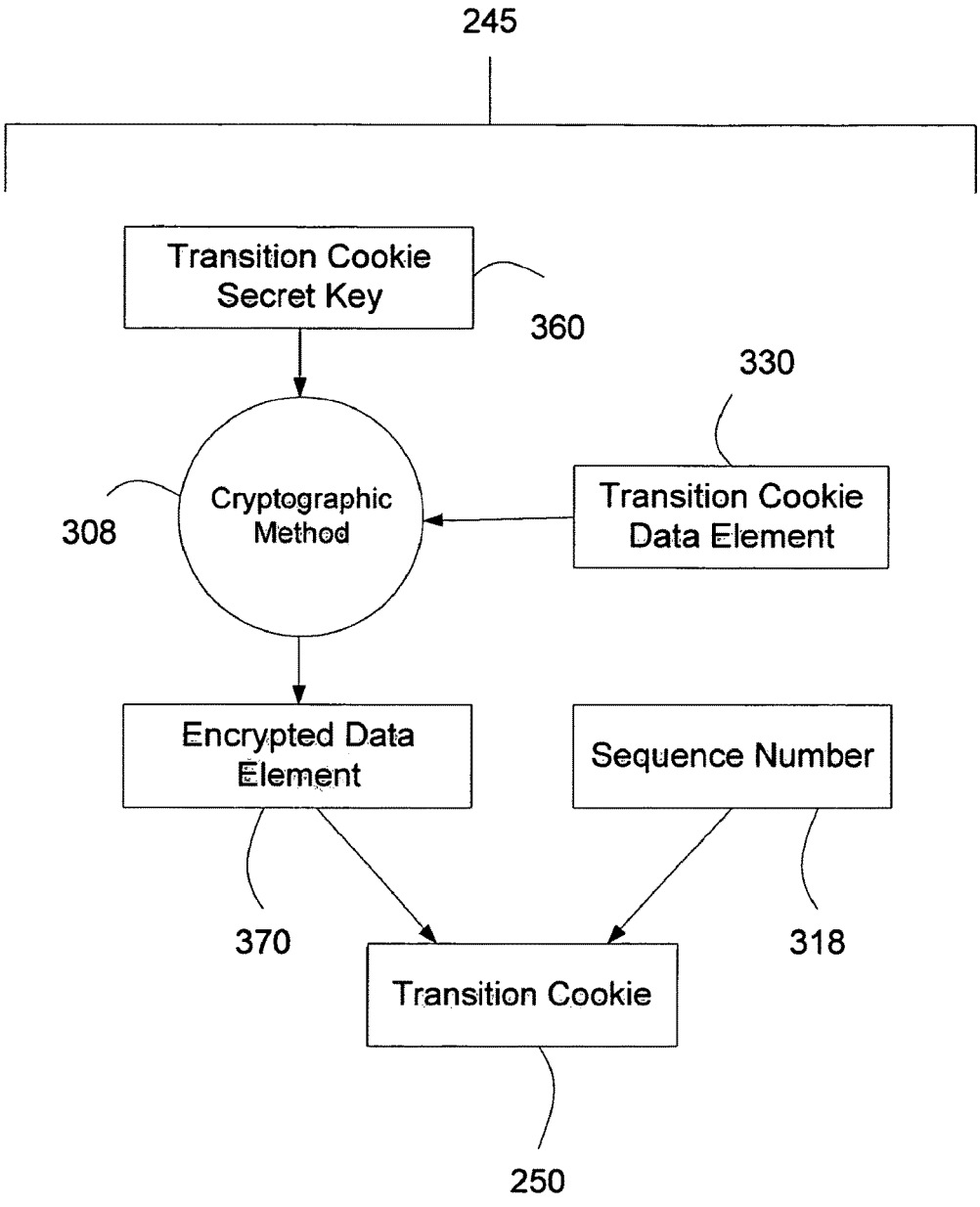

245
Transition Cookie Secret Key
360
330
Cryptographic Method
308
Transition Cookie Data Element
Encrypted Data Element
Sequence Number
370
318
Transition Cookie
250

Figure 3c

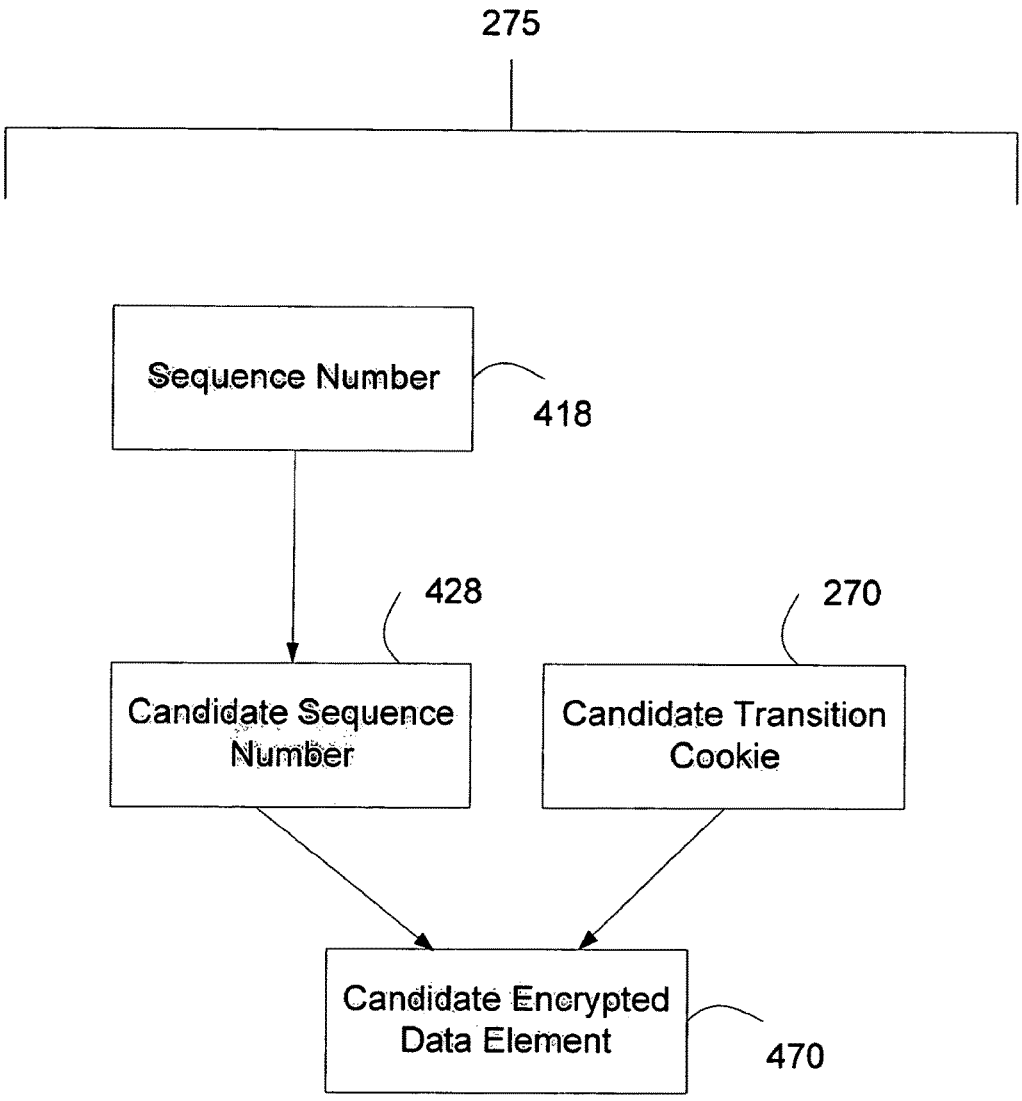

275
Sequence Number
418
428
270
Candidate Sequence Number
Candidate Transition Cookie
Candidate Encrypted Data Element
470

Figure 4a

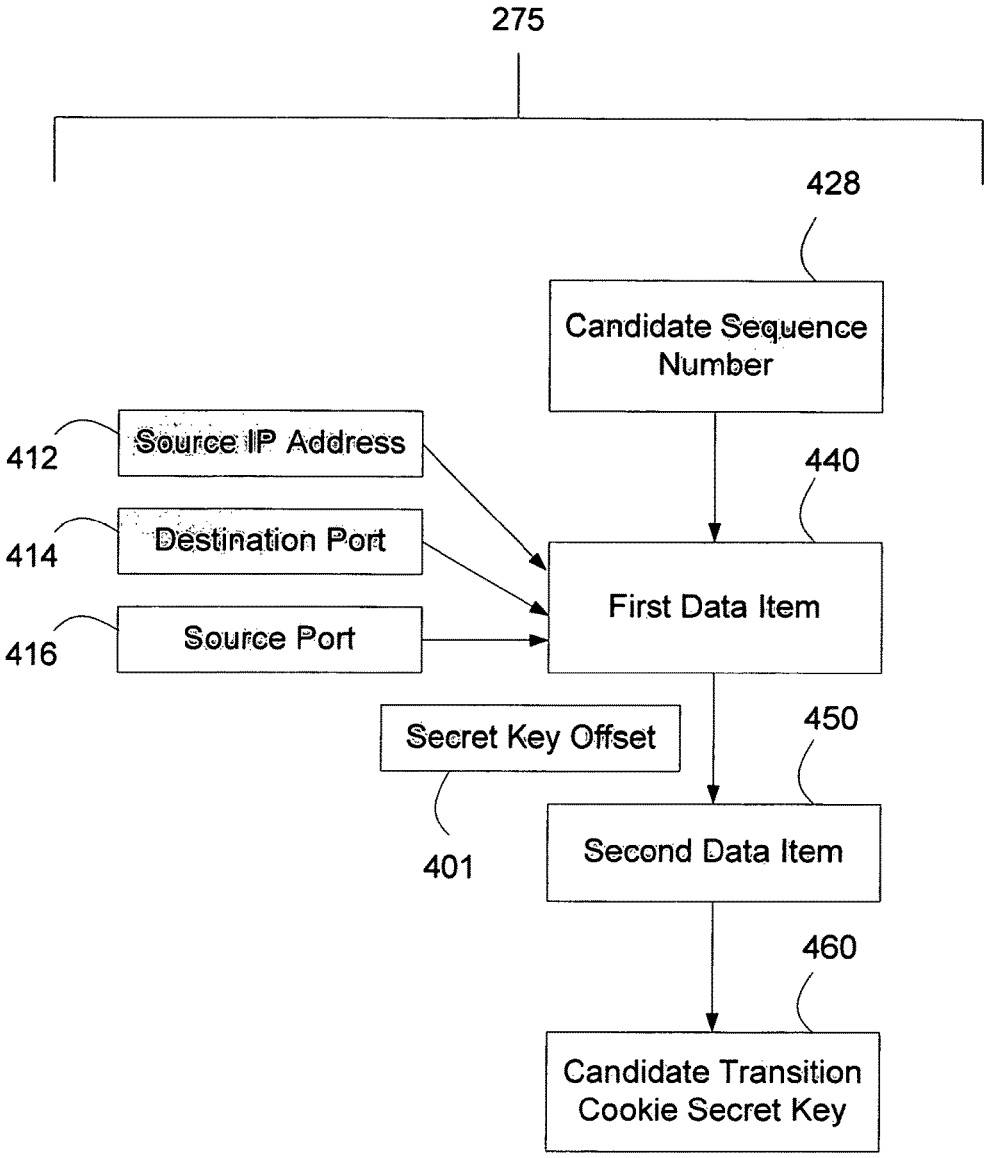

275
428
Candidate Sequence Number
412
Source IP Address
440
414
Destination Port
First Data Item
416
Source Port
Secret Key Offset
450
401
Second Data Item
460
Candidate Transition Cookie Secret Key

Figure 4b

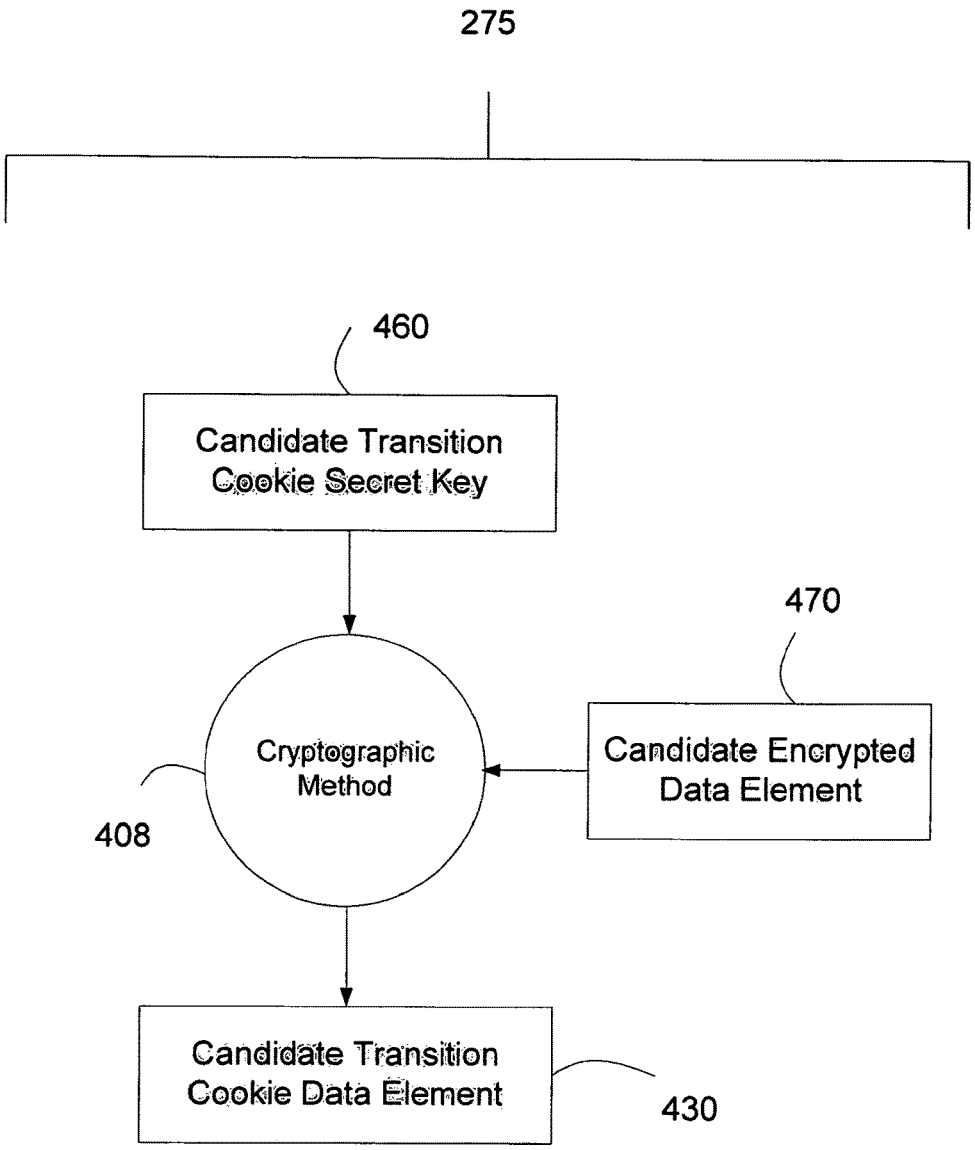

275
460
Candidate Transition Cookie Secret Key
470
Cryptographic Method
Candidate Encrypted Data Element
408
Candidate Transition Cookie Data Element
430

Figure 4c

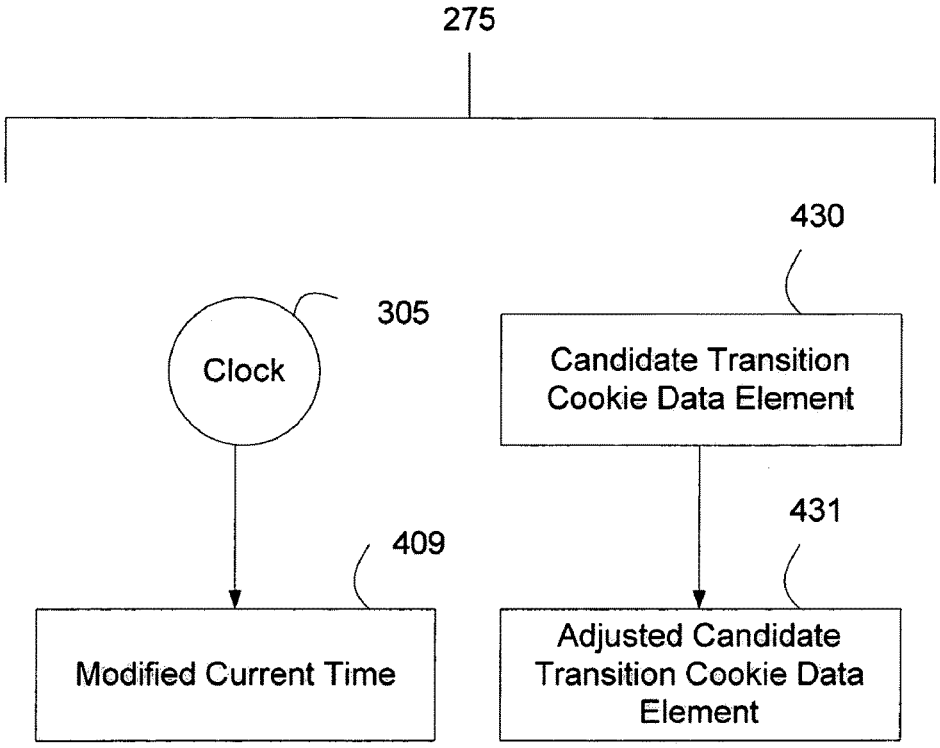

275
430
305
Clock
Candidate Transition Cookie Data Element
431
409
Modified Current Time
Adjusted Candidate Transition Cookie Data Element

Figure 4d

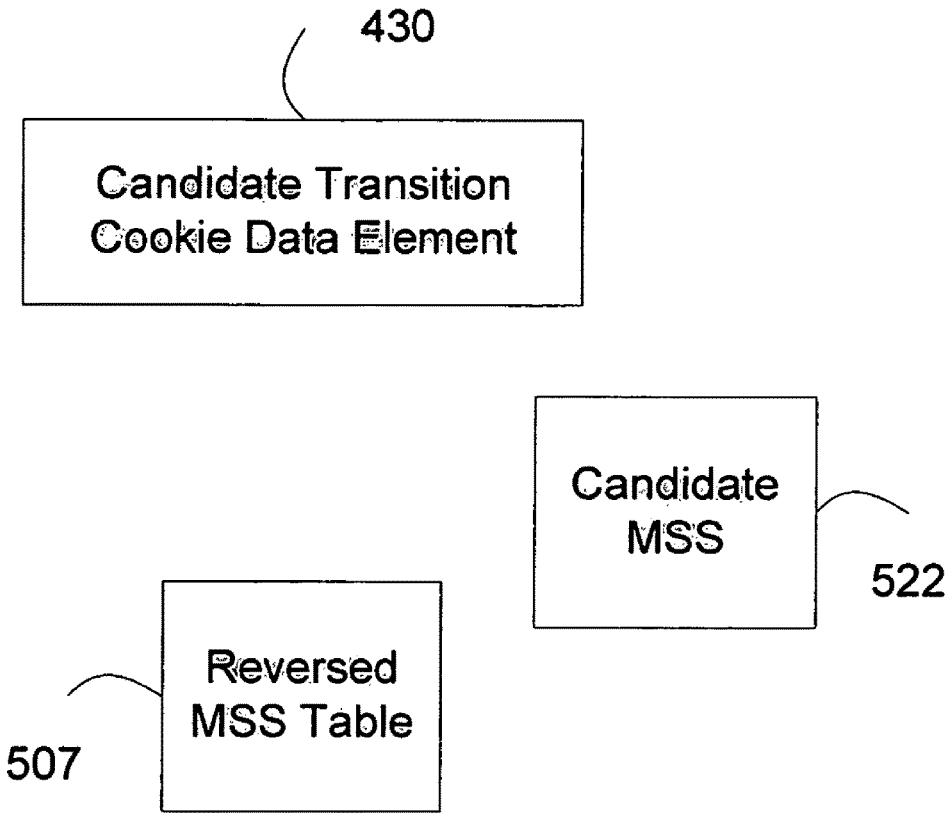

430
Candidate Transition Cookie Data Element
Candidate MSS
522
Reversed MSS Table
507

Figure 5

SYSTEM AND METHOD FOR AN ADAPTIVE TCP SYN COOKIE WITH TIME VALIDATION

Matter enclosed in heavy brackets [ ] appears in the original patent but forms no part of this reissue specification; matter printed in italics indicates the additions made by reissue; a claim printed with strikethrough indicates that the claim was canceled, disclaimed, or held invalid by a prior post-patent action or proceeding.

CROSS-REFERENCE TO RELATED APPLICATIONS

This application is a continuation reissue application of U.S. Pat. No. 7,675,854 and claims benefit under 35 U.S.C. 120 as a continuation of application Ser. No. 13/413,191 filed on Mar. 6, 2012, which is an application for reissue of U.S. Pat. No. 7,675,854, originally issued on Mar. 9, 2010.

BACKGROUND OF THE INVENTION

When a TCP (Transmission Control Protocol) connection starts, a destination host receives a SYN (synchronize/start) packet from a source host and sends back a SYN ACK (synchronize acknowledge). The destination host normally then waits to receiver an ACK (acknowledge) of the SYN ACK before the connection is established. This is referred to as the TCP "three-way handshake."

While waiting for the ACK to the SYN ACK, a connection queue of finite size on the destination host keeps track of connections waiting to be completed. This queue typically empties quickly since the ACK is expected to arrive a few milliseconds after the SYN ACK is sent.

A TCP SYN flood attack is a well known denial of service attack that exploits the TCP three-way handshake design by having an attacking source host generate TCP SYN packets with random source addresses toward a victim host. The victim destination host sends a SYN ACK back to the random source address and adds an entry to the connection queue, or otherwise allocates server resources. Since the SYN ACK is destined for an incorrect or non-existent host, the last part of the "three-way handshake" is never completed and the entry remains in the connection queue until a timer expires, typically, for example, for about one minute. By generating phony TCP SYN packets from random IP addresses at a rapid rate, it is possible to fill up the connection queue and deny TCP services (such as e-mail, file transfer, or WWW) to legitimate users. In most instances, there is no easy way to trace the originator of the attack because the IP address of the source is forged. The external manifestations of the problem may include inability to get e-mail, inability to accept connections to WWW or FTP services, or a large number of TCP connections on your host in the state SYN_RCVD.

A malicious client sending high volume of TCP SYN packets without sending the subsequent ACK packets can deplete server resources and severely impact the server's ability to serve its legitimate clients.

Newer operating systems or platforms implement various solutions to minimize the impact of TCP SYN flood attacks. The solutions include better resource management, and the use of a "SYN cookie".

In an exemplary solution, instead of allocating server resource at the time of receiving a TCP SYN packet, the server sends back a SYN/ACK packet with a specially constructed sequence number known as a SYN cookie. When the server then receives an ACK packet in response to the SYN/ACK packet, the server recovers a SYN cookie from the ACK packet, and validates the recovered SYN cookie before further allocating server resources.

The effectiveness of a solution using a SYN cookie depends on the method with which the SYN cookie is constructed. However, existing solutions using a SYN cookie typically employ a hash function to construct the SYN cookie, which can lead to a high percentage of false validations of the SYN cookie, resulting in less than satisfactory protection again TCP SYN flood attack.

Therefore, there is a need for a better system and method for constructing and validating SYN cookies.

SUMMARY OF THE INVENTION

An aspect of the present invention provides a system for TCP SYN cookie validation. The system includes a host server including a processor and memory. The processor is configured for receiving a session SYN packet, generating a transition cookie, the transition cookie comprising a time value representing the actual time, sending a session SYN/ACK packet, including the transition cookie, in response to the received session SYN packet, receiving a session ACK packet, and determining whether a candidate transition cookie in the received session ACK packet comprises a time value representing a time within a predetermined time interval from the time the session ACK packet is received.

One aspect of the invention includes the system above in which the processor is further configured for regarding the received session ACK packet as valid if the candidate transition cookie in the received session ACK packet comprises a time value representing a time within a predetermined time interval from the time the session ACK packet is received.

In another aspect of the invention, the predetermined time interval is in the range of one to six seconds.

In one aspect of the invention, the predetermined time interval is three seconds.

In another aspect of the invention, the step of generating the transition cookie includes the use of data obtained from the session SYN packet.

In one aspect of the invention, the data obtained from the session SYN packet comprises the source IP address of an IP header associated with the session SYN packet.

In another aspect of the invention, the data obtained from the session SYN packet comprises the sequence number of a TCP header associated with the session SYN packet.

In another aspect of the invention, the data obtained from the session SYN packet comprises a source port associated with the session SYN packet.

In another aspect of the invention, the data obtained from the session SYN packet comprises a destination port associated with the session SYN packet.

Another aspect of the present invention provides a method for TCP SYN cookie validation. The method includes receiving a session SYN packet by a TCP session setup module, generating a transition cookie by the TCP session setup module, the transition cookie comprising a time value representing the actual time, sending a session SYN/ACK packet, including the transition cookie, in response to the received session SYN packet, receiving a session ACK packet, and determining whether a candidate transition cookie in the received session ACK packet comprises a time value representing a time within a predetermined time interval from the time the session ACK packet is received.

In an aspect of the invention, the method further includes indicating the received session ACK packet comprises a valid candidate transition cookie if the time value of the candidate transition cookie is within a predetermined time interval of the time the session ACK packet is received.

In another aspect of the invention, the step of generating the transition cookie includes the use of data obtained from the session SYN packet.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a schematic diagram illustrating a host server including a TCP session setup module and a client server, in accordance with an embodiment of the present invention;

FIG. **2** is a schematic diagram of a TCP/IP handshake in accordance with an embodiment of the present invention;

FIG. **3**a illustrates a method including steps for generating a transition cookie data element by a transition cookie generator **245**, in accordance with an embodiment of the present invention;

FIG. **3**b illustrates a method including steps for generating a transition cookie secret key by a transition cookie generator **245** based on data obtained from the received session SYN packet, in accordance with an embodiment of the present invention;

FIG. **3**c illustrates a method including steps for generating a transition cookie based on a transition cookie data element, a transition cookie secret key, and data obtained from a received session SYN packet in accordance with an embodiment of the present invention;

FIG. **4**a illustrates steps for generating a candidate encrypted data element by a transition cookie validator **275** based on data obtained from a received session ACK packet, in accordance with an embodiment of the present invention;

FIG. **4**b illustrates a method including steps for generating a candidate transition cookie secret key by a transition cookie validator **275** based on data obtained from a received session ACK packet and a candidate sequence number, in accordance with an embodiment of the present invention;

FIG. **4**c illustrates a method including steps for generating a candidate transition cookie data element by a transition cookie validator **275** based on a candidate encrypted data element and a candidate transition cookie secret key, in accordance with an embodiment of the present invention;

FIG. **4**d illustrates a method including the steps for validating a candidate transition cookie data element, in accordance with an embodiment of the present invention; and

FIG. **5** illustrates a method including steps for generating information based on a validated candidate transition cookie data element, in accordance with an embodiment of the present invention.

DETAILED DESCRIPTION

In the following description, for purposes of explanation, specific numbers, materials and configurations are set forth in order to provide a thorough understanding of the invention. It will be apparent, however, to one having ordinary skill in the art, that the invention may be practiced without these specific details. In some instances, well-known features may be omitted or simplified so as not to obscure the present invention. Furthermore, reference in the specification to "one embodiment" or "an embodiment" means that a particular feature, structure or characteristic described in connection with the embodiment is included in at least one embodiment of the invention. The appearances of the phrase "in an embodiment" in various places in the specification are not necessarily all referring to the same embodiment.

Transmission Control Protocol ("TCP") is one of the main protocols in TCP/IP networks. Whereas the Internet Protocol ("IP") deals only with packets, TCP enables two hosts to establish a connection and exchange streams of data. TCP guarantees delivery of data and also guarantees that packets will be delivered in the same order in which they were sent.

The terms "host server" and "client server" referred to in the descriptions of various embodiments of the invention herein described are intended to generally describe a typical system arrangement in which the embodiments operate. The "host server" generally refers to any computer system interconnected to a TCP/IP network, including but not limited to the Internet, the computer system comprising at a minimum a processor, computer memory, and computer software. The computer system is configured to allow the host server to participate in TCP protocol communications over its connected TCP/IP network. Although the "host server" may be a single personal computer having its own IP address and in communication with the TCP/IP network, it may also be a multi-processor server or server bank. The "client server" is similar to the "host server", although it is understood that the "client server" may, in fact, be a single personal computer attached to the TCP/IP network. The only difference between the client and the host server for the purposes of the present invention is that the host server receives the SYN from the client server, sends a SYN ACK to the client server, and waits for the ACK from the client server.

FIG. **1** is a schematic diagram illustrating an embodiment of the present invention. A host server **102** may include a TCP session module **104**. The TCP session setup module **104** can engage in a TCP handshake **108**, such as described above, with a client server **106**. In an embodiment, the TCP session setup module **104** is a software component of the host server **102**. In one embodiment, the TCP session setup module **104** is implemented in an Application Specific Integrated Circuit ("ASIC") or a Field Programmable Gate Array ("FPGA"). It is the TCP session setup module that handles the "3-way handshake" **108** between the host server **102** and the client server **106**. The TCP session setup module may itself also incorporate modules for sending and receiving TCP session packets. These modules may include but are not limited to a session SYN packet receiver, a session SYN/ACK packet sender, and a session ACK packet receiver, which are all known to those of ordinary skill in the computer arts.

The TCP sessions setup module **104** may itself be embedded in one or more other host server modules (not shown). The TCP session setup module may alternatively comprise a hardware or firmware component. For example, the software which handles the TCP handshake **108** on behalf of the host server **102** may be programmed onto a externally programmable read-only memory ("EPROM") (not shown), and the EPROM may then be integrated into the host server. In another example, the ASIC or FPGA is integrated into the host server.

FIG. **2** illustrates a TCP session setup module **104** processing TCP/IP segments (not shown), such as session SYN packet **210**, session SYN/ACK packet **220**, and session ACK packet **230**.

A TCP/IP segment includes a TCP header and an IP header as described in IETF RFC 793 "Transmission Control Protocol" section 3.1 "Header Format", incorporated herein by reference. A TCP header optionally includes a sack-permitted option as described in IETF RFC 2018 "TCP Selective Acknowledgement Options" section 2 "Sack-Per-

mitted Option", incorporated herein by reference. A session SYN packet **210** is a TCP/IP segment with the SYN control bit in the TCP Header set to "1". A session SYN/ACK packet **220** is a TCP/IP segment with the SYN control bit and the ACK control bit in the TCP header set to "1". A Session ACK Packet **230** is a TCP/IP segment with the ACK control bit in the TCP header set to "1".

Referring to FIG. **2**, in an embodiment, the TCP session setup module **104** receives a session SYN packet **210**, obtains data from a session SYN packet **210**, such as but not limited to the source IP address of the IP header, or the sequence number of the TCP header, and uses the data to generate a transition cookie **250**. The transition cookie **250** is preferably a 32-bit data element. In response to the session SYN packet **210**, the TCP session setup module **104** creates and sends out a session SYN/ACK packet **220** in accordance with IETF RFC 793 "Transmission Control Protocol" section 3.4 "Establishing a connection", incorporated herein by reference. The TCP session setup module **104** preferably includes the transition cookie **250** as the sequence number of the TCP header in the session SYN/ACK packet **220**.

After the TCP session setup module **104** has sent out the session SYN/ACK packet **220**, it waits for receipt of a responding session ACK packet **230**. In an embodiment, when a session SYN/ACK packet **230** is received, the TCP session setup module **104** generates a 32-bit candidate transition cookie **270** such that the sum of candidate transition cookie **270** and a value of "1" equal the acknowledgement number of the TCP header in the session ACK packet **230**. For example, if the acknowledgement number is "41B4362A" in hexadecimal format the candidate transition cookie **270** is "41B43629" in hexadecimal format; the sum of "41B43629" and a value of "1" equals "41B4362A". In another example, if the acknowledgement number is "00A30000" in hexadecimal format the candidate transition cookie **270** is "00A2FFFF" in hexadecimal format; the sum of "00A2FFFF" and a value of "1" equals "00A30000". In another example, if the acknowledgement number is "00000000" in hexadecimal format the Candidate Transition Cookie **270** is "FFFFFFFF" in hexadecimal format; the sum of "FFFFFFFF" and a value of "1" equals "00000000", with the most significant bit carried beyond the 32-bit boundary. The TCP session setup module **104** may thus validate the candidate transition cookie **270** in this manner. If the TCP session setup module **104** determines that the candidate transition cookie **270** is thus valid, the session ACK packet **230** is also valid. In this case, the TCP session setup module **104** obtains data from the validated session ACK packet **230** and sends the data and information generated during the validation of candidate transition cookie **270** to a computing module (not shown) for further processing.

In order to generate and validate transition cookies **250**, **270**, the TCP session setup module **104** may include a transition cookie generator **245** and a transition cookie validator **275**, respectively. Alternatively, the generation and validation may be performed directly by the TCP session setup module **104**. In the descriptions herein, references to the TCP and transition cookie validator **275** are understood to include any of the alternative embodiments of these components.

A transition cookie generator **245** includes the functionality of generating a transition cookie based on the data obtained from a session SYN **210** packet received by the TCP session setup module **104**.

A transition cookie validator **275** includes the functionality of validating a candidate transition cookie **270** generated based on data obtained from a session ACK packet **230** received by the TCP session setup module **104**.

In exemplary operation, a transition cookie generator **245** is software or firmware that generates a transition cookie **250** based on data obtained from a session SYN packet **210** received by the TCP session setup module **104**. An exemplary method for generating a transition cookie **250** by a transition cookie generator **245** includes multiple steps as illustrated in FIGS. **3**a-**3**c.

FIG. **3**a illustrates exemplary steps for generating a transition cookie data element **330** by a transition cookie generator **245**. A transition cookie generator **245** includes a clock **305** indicating the current time of day in microseconds in a 32-bit format.

The transition cookie data element **330** is preferably a 32-bit data element, generated by the transition cookie generator **245** based on the selective ACK **321**, the MSS index **324** and the 32-bit current time of day indicated by clock **305**. Selective ACK **321** is a 1-bit data element which is set to a value of "1" by transition cookie generator **245** if a TCP header in a received session SYN packet **210** includes an optional sack-permitted option, or to "**0**" if a TCP header in a received session SYN packet **210** does not include an optional sack-permitted option.

Maximum Segment Size ("MSS") **322** is the maximum number of bytes that TCP will allow in an TCP/IP packet, such as session SYN packet **210**, session SYN/ACK packet **220**, and session ACK packet **230**, and is normally represented by an integer value in a TCP packet header. If a TCP header in a received session SYN packet **210** includes a maximum segment size option, the transition cookie generator **245** sets the MSS **322** to equal the maximum segment size option data of the maximum segment size option. Otherwise, if the TCP header in a received session SYN packet **210** does not include a maximum segment size option, the transition cookie generator **245** sets the MSS **322** to a default value, for example, such as integer "536". The MSS index **324** is a 4-bit data element set by the transition cookie generator **245** based on the MSS **322**. The transition cookie generator **245** preferably includes an MSS table **307**, which maps an MSS **322** to an MSS index **324**. The transition cookie generator **245** maps a MSS **322** with the MSS table **307** to set the value of MSS index **324**. For example, MSS **322** has an integer value of "1460". After the mapping, MSS index **324** has a value of "4" as represented in hexadecimal format. In an alternative embodiment, means other than an MSS table **307** may be employed to determine the MSS index **324** value, such as the use of a mapping algorithm.

In generating a transition cookie data element **330**, the transition cookie generator **245** sets a transition cookie data element **330** to equal the 32-bit current time of day indicated by clock **305**. For example, the 32-bit current time of day may be "A68079E8" as represented in hexadecimal format, so the transition cookie data element **330** has a value of "A68079E8".

Next, the transition cookie generator **245** replaces the least significant 4 bits (bit **0**-**3**) of transition cookie data element **330** with the MSS index **324**, and replaces bit **4** of a transition cookie data element **330** with selective ACK **321**. For example, if a transition cookie data element **330** has been set to a value of "A68079E8", selective ACK **321** has a value of "1", and MSS index **324** has a value of "4" as represented in hexadecimal format, after the replacements, transition cookie data element **330** has a value of "A68079F4" in hexadecimal format.

FIG. **3**b illustrates exemplary steps for generating a transition cookie secret key **360**, such as by a transition cookie generator **245** based on data obtained from a received session SYN packet **210**. The data used in generating the transition cookie secret key **360** may include at least the source IP address **312** of an IP header, a destination port **314**, a source port **316** and a sequence number **318** of a TCP header in a received session SYN packet **210**. In generating a transition cookie secret key **360**, a transition cookie generator **245** forms a 96-bit data element, a first data item **340**, by concatenating a source IP address **312**, a sequence number **318**, a source port **316**, and a destination port **314**. For example, if the source IP address **312** is 192.168.1.134, the hexadecimal representation being "C0A80186", the sequence number **318** is "9A275B84", the source port **316** is 4761, the hexadecimal representation being "1299", and the destination port **314** is 240, the hexadecimal representation being "00F0", then, after the concatenation, the first data item **340** has a hexadecimal value of "C0A801869A275B84129900F0".

Next, since the transition cookie secret key **360** is a 128-bit data element, the transition cookie generator **245** may use a hash function to generate the transition cookie secret key **360** from the first data item **340**. Further, the transition cookie generator **245** may use a secret key offset **301**, which may be a 6-bit integer value, to select a 6-bit non-negative integer from first data item **340** starting at the bit indicated by secret key offset **301**. For example, if the secret key offset **301** has a value of "12" and the first data item **340** has a hexadecimal value of "C0A801869A275B84129900F0", the transition cookie generator **245** selects a 6-bit non-negative integer from the first data item **340** starting at bit **12** (bit **12-17**). The selected non-negative integer is of this example is thus "16". The transition cookie generator **245** then uses the selected non-negative integer to select 64 bits of data from the first data item **340**, starting at the bit indicated by the selected non-negative integer, to generate the second data item **350**, which has 64 bits.

For example, if the selected non-negative integer is "8" and the first data item **340** has a hexadecimal value of "C0A801869A275B84129900F0", the transition cookie generator **245** selects 64 bits (bit **8-71**) of the first data item **340** to generate a second data item **350**, having a hexadecimal value of "869A275B84129900". In another example, if the selected non-negative integer is "52", and the transition cookie generator **245** selects 64 bits (bit **52-95** and bit **0-19**) of the first data item **340** in a wrap-around fashion, bits **52-95** have a hexadecimal value of "C0A801869A2", and bit **0-19** have a hexadecimal value of "900F0", so the generated second data item **350** has a hexadecimal value of "900F0C0A801869A2". The transition cookie generator **245** then generates a transition cookie secret key **360** by storing the second data item **350** in the least significant 64 bits (bit **0-63**) of the transition cookie secret key **360** and setting the most significant 64 bits (bit **64-127**) to "0". For example, if the second data item **350** has a hexadecimal value of "869A275B84129900", the transition cookie secret key **360** has a hexadecimal value of "0000000000000000869A275B84129900".

FIG. **3**c illustrates exemplary steps for generating a transition cookie **250** based on a transition cookie data element **330**, a transition cookie secret key **360**, and data obtained from a received session SYN packet **210**, including a sequence number **318** of a TCP header in a received session SYN packet **210**. To generate a transition cookie **250**, a transition cookie generator **245** applies a cryptographic method **308** on the transition cookie secret key **360** and the transition cookie data element **330**, such as an RC5 algorithm described in IETF RFC 2040 "The RC5, RC5-CBC, RC5-CBC-Pad, and RC5-CTS Algorithms" section 1 "Overview", and sections 2-8 with detailed explanations, incorporated herein by reference. The RC5 algorithm takes a 32-bit plaintext input and a 128-bit encryption key to generate a 32-bit ciphertext output. The transition cookie generator **245** uses the transition cookie data element **330** as the plaintext input to the RC5 algorithm, and the transition cookie secret key **360** as the encryption key input to the RC5 algorithm. The transition cookie generator **245** stores the resulting 32-bit ciphertext output of the RC5 algorithm in the encrypted data element **370**.

Next, the transition cookie generator **245** performs an unsigned binary addition on an encrypted data element **370** and the sequence number **318**, and stores the result in the transition cookie **250**. For example, if the encrypted data element **370** has a value of "0025BC83" in hexadecimal format, and the sequence number **318** has a value of "0743BD55" in hexadecimal format, the result of the addition is hexadecimal "076979D8". After the addition, the transition cookie **250** has a value of "076979D8" in hexadecimal. In another example, if the encrypted data element **370** has a value of "BE43D096" in hexadecimal format, and the sequence number **318** has a value of "9A275B84" in hexadecimal format, the result of the addition, and the value of transition cookie **250** is hexadecimal "1586B2C1A", with the most significant bit carried beyond the 32-bit boundary.

In another embodiment, a transition cookie generator **245** may use different steps to generate a transition cookie secret key **360**. For example, a secret key offset **301** may be an integer of a different bit length, such as a 4-bit integer value, a 3-bit integer value, or a 5-bit integer value. Also, a transition cookie generator **245** may use a secret key offset **301** to select a non-negative integer value of a different bit length from a first data item **340**. For example, a transition cookie generator **245** may select a 4-bit non-negative integer value, a 7-bit non-negative integer value, or a 5-bit non-negative value from a first data item **340**.

In other embodiments, a transition cookie generator **245** may store a second data item **350** in the least significant 64 bits (bit **0-63**) of a transition cookie secret key **360** or store second data item **350** in the most significant 64 bits (bit **64-127**) of a transition cookie secret key **360**.

A transition cookie generator **245** may also perform an exclusive-or operation on the most significant 48 bits (bit **0-47**) of a first data item **340** and the least significant 48 bits (bit **48-95**) of a first data element **340** to form a 48-bit temporary data element (not shown). Similarly, in another embodiment, a transition cookie generator **245** may perform an exclusive-or operation on the 48 even bits (bit **0**, **2**, **4**, . . . **90**, **92**, **94**) and the 48 odd bits (bit **1**, **3**, **5**, . . . **93**, **95**, **97**) to form a 48 bit temporary data element. In yet another embodiment, a transition cookie generator **245** may store a 48-bit temporary data element in the least significant 48 bits (bit **0-47**) and the most significant 48 bits (bit **80-127**) of a transition cookie secret key **360**, and set bit **48-79** to "0", or store a 48-bit temporary data element in the least significant 48 bits (bit **0-47**) of a transition cookie secret key **360**, and set the most significant 80 bits (bit **48-127**) of a transition cookie secret key **360** to "0".

In other embodiments of the invention, a transition cookie generator **245** may use an encryption algorithm to generate a transition cookie secret key **360** from the first data item **340**.

In another embodiment, a transition cookie generator **245** includes a secret key and an encryption algorithm, and uses a first data element **340** as a plaintext input, and a secret key as an encryption key input to the encryption algorithm to generate a 128-bit ciphertext output. Next, a transition cookie generator **245** generates a transition cookie secret key **360** as a 128-bit ciphertext output. Alternatively, the ciphertext output may be a 96-bit data element, and a transition cookie generator **245** stores a 96-bit ciphertext output in the least significant 96 bits (bit **0-95**) of a transition cookie secret key **360**, and sets the most significant 32 bits (bit **96-127**) to "0". In another alternative, a transition cookie generator **245** stores the least significant 32 bits (bit **0-31**) of a 96-bit ciphertext output in the most significant 32 bits (bit **96-127**) of a transition cookie secret key **360**.

As seen in FIG. **2**, a transition cookie validator **275** validates a candidate transition cookie **270** generated from a session ACK packet **230** received by the TCP session setup module **104**. An exemplary method for validating a candidate transition cookie **270** by a transition cookie validator **275** may include multiple steps as illustrated in FIGS. **4**a-**4**d.

FIG. **4**a illustrates exemplary steps for generating a candidate encrypted data element **470** by a transition cookie validator **275** based on data obtained from a received session ACK packet **230**. The candidate encrypted data element **470** may be a 32-bit data element generated based on the sequence number **418** of the TCP header in the received session ACK packet **230**, and the candidate transition cookie **270** generated from the received session ACK packet **230** as illustrated in FIG. **2**.

The candidate sequence number **428** may be a 32-bit data element generated by a transition cookie validator **275** such that the sum of candidate sequence number **428** and a value of "1" equals the sequence number **418**.

The candidate encrypted data element **470** is generated by the transition cookie validator **275** such that the result of performing an unsigned binary addition of the candidate encrypted data element **470** and the candidate sequence number **428** equals the candidate transition cookie **270**.

FIG. **4**b illustrates exemplary steps for generating a candidate transition cookie secret key **460** by the transition cookie validator **275** based on data obtained from the received session ACK packet **230** and a candidate sequence number **428**. The data used for generating the candidate transition cookie secret key **460** may include at least a source IP address **412** of the IP header in a received session ACK packet **230**, a destination port **414** and a source port **416** of the TCP header in a received session ACK packet **230**. In the process, a 96-bit first data item **440** is formed by a transition cookie validator **275** by concatenating a source IP address **412**, a candidate sequence number **428**, a source port **416**, and a destination port **414**. For example, if the source IP address **412** is 192.168.1.134, having a hexadecimal representation of "C0A80186", the candidate sequence number **428** is hexadecimal "9A275B84", the source port **416** is 4761, having a hexadecimal representation of "1299", and the destination port **414** is 240, having a hexadecimal representation of "00F0", after the concatenation, the first data item **440** has a hexadecimal value of "C0A801869A275B84129900F0".

Next, the 128-bit candidate transition cookie secret key **460** is generated from a first data item **440** by a transition cookie validator **275** using a hash function. In an embodiment, a transition cookie validator **275** uses a 6-bit secret key offset **401** to select a 6-bit non-negative integer from a first data item **440** starting at a bit indicated by secret key offset **401**. For example, if the secret key offset **401** has a value of "12" and the first data item **440** is "C0A801869A275B84129900F0", the transition cookie validator **275** selects a 6-bit non-negative integer from the first data item **440** starting at bit **12** (bits **12-17**), selecting the non-negative integer "16". The transition cookie validator **275** then generates a 64-bit second data item **350** by using the selected non-negative integer to select 64 bits of data from the first data item **440**, starting at the bit indicated by the selected non-negative integer.

For example, if the selected non-negative integer is "8" and the first data item **440** has a hexadecimal value of "C0A801869A275B84129900F0", the transition cookie validator **275** selects 64 bits (bit **8-71**) of the first data item **440** to generate a second data item **450** having a hexadecimal value of "869A275B84129900". In another example, if the first data item **440** has a hexadecimal value of "C0A801869A275B84129900F0", and the selected non-negative integer is "52", the transition cookie validator **275** selects 64 bits (bit **52-95** and bit **0-19**) in a wrap-around fashion. Bits **52-95** have a hexadecimal value of "C0A801869A2", and bits **0-19** have a hexadecimal value of "900F0", so the generated second data item **450** has a hexadecimal value of "900F0C0A801869A2".

Next, the transition cookie validator **275** generates a candidate transition cookie secret key **460** by storing the second data item **450** in the least significant 64 bits (bit **0-63**) of the candidate transition cookie secret key **460** and setting the most significant 64 bits (bit **64-127**) to "0". For example, if the second data item **450** has a hexadecinmal value of "869A275B84129900", the candidate transition cookie secret key **460** has a hexadecimal value of "0000000000000000869A275B84129900".

FIG. **4**C illustrates exemplary steps for generating a candidate transition cookie data element **430** by a transition cookie validator **275** based on a candidate encrypted data element **470** and a candidate transition cookie secret key **460**.

In an embodiment, a transition cookie validator **275** applies a cryptographic method **408** on a candidate transition cookie secret key **460** and a candidate encrypted data element **470**. An exemplary cryptographic method **408** is an RC5 algorithm described in IETF RFC 2040 "The RC5, RC5-CBC, RC5-CBC-Pad, and RC5-CTS Algorithms" section 1 "Overview", and sections 2-8 with detailed explanations, incorporated herein by reference. The RC5 algorithm takes a 32-bit ciphertext input and a 128-bit decryption key to generate a 32-bit plaintext output. A transition cookie validator **275** uses a candidate encrypted data element **470** as a ciphertext input to the RC5 algorithm, and a candidate transition cookie secret key **460** as a decryption key input to the RC5 algorithm, to generate a 32-bit candidate transition cookie data element **430** as the plaintext output of the RC5 decryption algorithm.

FIG. **4**d illustrates exemplary steps by a transition cookie validator **275** of validating a candidate transition cookie data element **430**. In an embodiment, a transition cookie validator **275** includes a clock **305**. The clock **305** indicates the current time of day, preferably in microseconds in a 32-bit format. The modified current time **409** is a 32-bit data element set by a transition cookie validator **275** sets to the current time indicated by clock **305**. A transition cookie validator **275** then sets the least significant 5 bits (bit **0-4**) of the modified current time **409** to "0". For example, if the modified current time **409** has a value of "89AE03F6" in hexadecimal format, after setting the least significant 5 bits to "0", the modified current time **409** has a hexadecimal value of "89AE03E0".

Next, a transition cookie validator **275** sets a 32-bit adjusted candidate transition cookie data element **431** to equal the candidate transition cookie data element **430**, and then sets the least significant 5 bits (bit **0-4**) of the adjusted candidate transition cookie data element **431** to "0". For example, if the adjusted candidate transition cookie data element **431** has a hexadecimal value of "89DB468F", after setting the least significant 5 bits to "0", the adjusted candidate transition cookie data element **431** has a hexadecimal value of "89DB4680".

The transition cookie validator **275** may then determine if the candidate transition cookie data element **430** is valid by determining if the adjusted candidate transition cookie data element **431** is within a time margin of 3 seconds of the modified current time **409**. In an embodiment, in order to determine if the adjusted candidate transition cookie data element **431** is within a time margin of 3 seconds of the modified current time **409**, the transition cookie stores the modified current time **409** in the least significant 32 bits (bit **0-31**) of a first 33-bit time data element, sets the most significant bit (bit **32**) to "0", and adds 6 seconds to the first 33-bit time data element. Adding 6 seconds is to add 6,000,000 micro seconds as represented by "5B8D80" in hexadecimal format. For example, if before the addition, the first 33-bit time data element has a hexadecimal value of "0FFFFFAE2", After the addition of "5B8D80", the first 33-bit time data element has a hexadecimal value of "1005B8862". The transition cookie validator **275** stores the adjusted candidate transition cookie data element **431** in the least significant 32 bits (bit **0-31**) of a second 33-bit time data element, sets the most significant bit (bit **32**) to "0", and adds 3 seconds to the second 33-bit time data element. Adding 3 seconds is to add 3,000,000 micro seconds as represented by hexadecimal "2DC6C0". The transition cookie validator **275** stores the modified current time **409** in the least significant 32 bits (bit **0-31**) of a third 33-bit time data element, and sets the most significant bit (bit **32**) to "0". If the second 33-bit time data element is smaller than the first 33-bit time data element and the second 33-bit time data element is larger than the third 33-bit time data element, the transition cookie validator **275** determines that the adjusted candidate transition cookie data element **431** is within 3 seconds of the modified current time **409**, and thus that the candidate transition cookie data element **430** is valid.

FIG. **5** illustrates exemplary steps of generating information based on a validated candidate transition cookie data element **430**. In an embodiment, candidate MSS **522** is an integer. A transition cookie validator **275** includes a reversed MSS table **507**, which includes information that maps a 4-bit data element to a candidate MSS **522**. A transition cookie validator **275** extracts the least significant 4-bit (bit **0-3**) data from candidate transition cookie data element **430**, maps the extracted 4-bit data to a reversed MSS table **507**, and stores the result in a candidate MSS **522**. A transition cookie validator **275** may then generate a maximum segment size option as described in IETF RFC 793 "Transmission Control Protocol" section 3.1 "Header Format", incorporated herein by reference, and sets a maximum segment size option data of the maximum segment size option to equal a candidate MSS **522**. A transition cookie validator **275** may further examine bit **4** of a candidate transition cookie data element **430**. If bit **4** of candidate transition cookie data element **430** has a value of "1", a transition cookie validator **275** may generate a sack-permitted option as described in IETF RFC 2018 "TCP Selective Acknowledgement Options" section 2, incorporated herein by reference. A TCP session setup module **104** may then send a sack-permitted option, a maximum segment size option, and data obtained from a received session ACK packet **230** to a computing module (not shown) for further processing.

There are many different encryption algorithms that use encryption keys of different bit lengths, such as, for example, 56-bit, 64-bit, 96-bit, 128-bit. These may generate ciphertext outputs of different bit lengths, for example, 96-bit, 64-bit, 128-bit, or 32-bit. Persons of ordinary skill in the cipher arts will be able to apply different methods, for example a hash function, to generate the transition cookie secret key **360** from the ciphertext output.

A transition cookie validator **275** may also use different steps to generate a candidate transition cookie secret key **460**. The steps used by a transition cookie validator **275** to generate a candidate transition cookie secret key **460** are similar to the steps used by a transition cookie generator **245** to generate a transition cookie secret key **360**.

Alternative embodiments of the invention may employ a different algorithm for the cryptographic methods **308**, **408**. In one example, the different algorithm is an RC2 algorithm described in IETF RFC 2268 "A Description of the RC2(r) Encryption Algorithm" section 1 "Introduction" and section 2-4 with detailed explanation, incorporated herein by reference. In another example, the different algorithm is a Blowfish algorithm. In one other example, the different algorithm is a Data Encryption Standards ("DES") algorithm based on Federal Information Processing Standards Publication "Data Encryption Standard (DES) FIPS PUB 46-3", which is incorporated herein by reference in its entirety. Other algorithms are also usable.

Also, a transition cookie validator **275** may use different time margins of modified current time **409** to determine if the candidate transition cookie data element is valid. Different time margins include but are not limited to 1 second, 4 seconds, 6 seconds, 2 seconds, or 11 seconds.

In an embodiment, the method of generating a transition cookie includes MD5 signature option information in the TCP options field. When this method is used, the method of validating a candidate transition cookie **270** correspondingly includes the MD5 signature option information in the TCP options field.

In another embodiment, transition cookie generator **245** may include a plurality of transition cookie generation methods for generating transition cookie **250**. For example, the secret key offset **301** may have a different value, such as an integer value of different bit length, such as 4-bit, or 8-bit. In other examples, the selected non-negative integer from first data item **340** may be of different bit length, such as 8-bit, or 10-bit, the cryptographic method **308** may be a different algorithm than RC5, or the generating of transition cookie data element **330** may include MD5 signature option information in the TCP options field of session SYN packet **210**. A transition cookie generation method may include steps different from the steps in the exemplary method illustrated in FIGS. **3a-3c**.

In an embodiment, the transition cookie generator **245** may selects method to generate transition cookie **250** based on random data.

The random data may include time. In one embodiment, transition cookie generator **245** selects a method based on the time of day. Alternatively, the transition cookie generator **245** may select a method after a time period, such as 10 seconds, 30 seconds, 2 minutes or 3 hours.

In another embodiment, the random data may include a source IP address in session SYN packet **210**, or a destination IP address in session SYN packet **210**.

The random data may include the network interface at which a TCP session setup module **104** receives a session SYN packet **210**, or a Virtual Local Area Network (VLAN) information associated with a session SYN packet **210**.

In one embodiment, transition cookie validator **275** includes a plurality of transition cookie validation methods for validating candidate transition cookie **270**. A transition cookie validation method may include steps different from the steps in the exemplary method illustrated in FIGS. **4**a-**4**d. A transition cookie validator **275** may select a method to validate candidate transition cookie **270** based on random data.

In these embodiments it is understood to be preferred that the transition cookie validator **275** selects a complementary method to the method selected by transition cookie generator **245**.

Although the invention herein has been described with reference to particular embodiments, it is to be understood that these embodiments are merely illustrative of the principles and applications of the present invention. It is therefore to be understood that numerous modifications may be made to the illustrative embodiments and that other arrangements may be devised without departing from the spirit and scope of the present invention as defined by the appended claims.

The invention claimed is:

[1. A system for TCP SYN cookie validation at a host server comprising:

a session SYN packet receiver for receiving a session SYN packet;

a transition cookie generator operating to generate a transition cookie with the use of a transition cookie secret key, the transition cookie comprising a time value representing the actual time, wherein the transition cookie generator generates the transition cookie secret key based on data obtained from the received session SYN packet, the data obtained from the SYN packet including at least one of a source IP address of an IP header, a destination port, a source port, and a sequence number of a TCP header in the received session SYN packet, wherein the transition cookie generator concatenates the obtained data from the session SYN packet to generate a first data item of the generator and the transition cookie generator uses a first hash function to generate the transition cookie secret key from the first data item of the generator;

a session SYN/ACK packet sender for sending the transition cookie in response to the received session SYN packet;

a session ACK packet receiver for receiving a session ACK packet, the session ACK packet including a candidate transition cookie; and

a transition cookie validator, for determining whether the candidate transition cookie in the received session ACK packet comprises a time value representing a time within a predetermined time interval from the time the session ACK packet is received, wherein the transition cookie validator generates a candidate transition cookie secret key based on data obtained from the received session ACK packet, the data obtained from the ACK packet including at least one of a source IP address of the IP header, a destination port, and a source port, wherein the transition cookie validator concatenates the obtained data from the session ACK packet to generate a first data item of the validator and the transition cookie validator uses the first or another hash function to generate the candidate transition cookie secret key from the first data item of the validator,

wherein at least one of:

the transition cookie generator uses a secret key offset to select one or more bits of data from the first data item of the generator in order to generate a second data item of the generator, and

the transition cookie validator uses a candidate secret key offset to select one or more bits of data from the first data item of the validator in order to generate a second data item of the validator.**]**

[2. The system according to claim **1**, in which the transition cookie validator determines that the received session ACK packet is valid if the candidate transition cookie in the received session ACK packet comprises a time value representing a time within a predetermined time interval from the time the session ACK packet is received.**]**

[3. The system according to claim **1**, in which the predetermined time interval is in the range of one to six seconds.**]**

[4. The system according to claim **1**, in which the predetermined time interval is three seconds.**]**

[5. The system according to claim **1**, in which the generating of the transition cookie includes the use of random data.**]**

[6. The system according to claim **1**, in which the generating of the transition cookie includes the use of data obtained from the session SYN packet.**]**

[7. A system for TCP SYN cookie validation at a host server comprising:

a session SYN packet receiver for receiving a session SYN packet;

a transition cookie generator operating to generate a transition cookie with the use of a transition cookie secret key, the transition cookie comprising a time value representing the actual time, wherein the transition cookie generator generates the transition cookie by (i) generating an encrypted data element of the generator by applying a cryptographic method on the transition cookie secret key and a transition cookie data element, (ii) performing an unsigned binary addition on the encrypted data element of the generator and a sequence number of a TCP header in the received session SYN packet, and (iii) storing the result in the transition cookie;

a session SYN/ACK packet sender for sending the transition cookie in response to the received session SYN packet;

a session ACK packet receiver for receiving a session ACK packet, the session ACK packet including a candidate transition cookie; and

a transition cookie validator, for determining whether the candidate transition cookie in the received session ACK packet comprises a time value representing a time within a predetermined time interval from the time the session ACK packet is received.**]**

[8. The system according to claim **7**, wherein the transition cookie data element comprises data based on at least one of: a selective ACK, an MSS index, and a 32-bit current time of day indicated by a clock.**]**

[9. A system for TCP SYN cookie validation at a host server comprising:

a session SYN packet receiver for receiving a session SYN packet;

a transition cookie generator operating to generate a transition cookie with the use of a transition cookie secret key, the transition cookie comprising a time value representing the actual time;

a session SYN/ACK packet sender for sending the transition cookie in response to the received session SYN packet;

a session ACK packet receiver for receiving a session ACK packet, the session ACK packet including a candidate transition cookie; and

a transition cookie validator, for determining whether the candidate transition cookie in the received session ACK packet comprises a time value representing a time within a predetermined time interval from the time the session ACK packet is received, wherein the transition cookie validator generates:

a candidate sequence number such that a sequence number of a TCP header in the received session ACK packet equals the sum of the candidate sequence number and a value of 1,

a candidate encrypted data element such that the result of performing an unsigned binary addition of the candidate encrypted data element and a candidate sequence number equals the candidate transition cookie, and

a candidate transition cookie data element by applying a cryptographic method on a candidate transition cookie secret key and the candidate encrypted data element.]

[10. The system according to claim 9, wherein the transition cookie validator validates the candidate transition cookie data element by adjusting the candidate transition cookie data element to generate, and determining if the adjusted candidate transition cookie data element is within a predetermined time margin of a modified current time.]

11. A system for TCP SYN cookie validation at a host server, the system comprising:

at least one processor and a memory storing:

a session SYN packet receiver, wherein when the session SYN packet receiver is executed by the at least one processor, the session SYN packet receiver causing the at least one processor to receive a session SYN packet;

a transition cookie generator, the transition cookie generator being executed by the at least one processor to generate a transition cookie with the use of a transition cookie secret key, the transition cookie comprising a time value representing the actual time;

a session SYN/ACK packet sender, the session SYN/ACK packet sender being executed by the at least one processor to send the transition cookie in response to the received session SYN packet;

a session ACK packet receiver, the session ACK packet receiver being executed by the at least one processor to receive a session ACK packet, the session ACK packet including a candidate transition cookie; and

a transition cookie validator, the transition cookie validator being executed by the at least one processor to determine whether the candidate transition cookie in the received session ACK packet comprises a time value representing a time within a predetermined time interval from the time the session ACK packet is received; and

wherein:

the transition cookie generator is executed by the at least one processor to generate the transition cookie secret key based on data obtained from the received session SYN packet;

the transition cookie validator is executed by the at least one processor to generate a candidate transition cookie secret key based on data obtained from the received session ACK packet;

the transition cookie generator is executed by the at least one processor to concatenate the obtained data from the session SYN packet to generate a first data item of the generator;

the transition cookie validator is executed by the at least one processor to concatenate the obtained data from the session ACK packet to generate a first data item of the validator;

the transition cookie generator is executed by the at least one processor to use a secret key offset to select one or more bits of data from the first data item of the generator in order to generate a second data item of the generator, and

the transition cookie validator is executed by the at least one processor to use a candidate secret key offset to select one or more bits of data from the first data item of the validator in order to generate a second data item of the validator.

12. The system according to claim 11, wherein:

when the transition cookie secret key is generated based on data obtained from the received session SYN packet, the obtained data includes at least one of: a source IP address of an IP header, a destination port, a source port, and a sequence number of a TCP header in the received session SYN packet, and

when the candidate transition cookie secret key is generated based on data obtained from the received session ACK packet, the obtained data includes at least one of:

a source IP address of the IP header, a destination port, and a source port.

13. The system according to claim 11, wherein at least one of:

the transition cookie generator is executed by the at least one processor to use a first hash function to generate the transition cookie secret key from the first data item of the generator, and

when the transition cookie validator is executed by the at least one processor to use the first or another hash function to generate the candidate transition cookie secret key from the first data item of the validator.

14. The system according to claim 11, in which the transition cookie validator is executed by the at least one processor to determine that the received session ACK packet is valid if the candidate transition cookie in the received session ACK packet comprises a time value representing a time within a predetermined time interval from the time the session ACK packet is received.

15. The system according to claim 11, in which the predetermined time interval is in the range of one to six seconds.

16. The system according to claim 11, in which the predetermined time interval is three seconds.

17. The system according to claim 11, in which the generating of the transition cookie includes the use of random data.

18. The system according to claim 11, in which the generating of the transition cookie includes the use of data obtained from the session SYN packet.

19. A system for TCP SYN cookie validation at a host server, the system comprising:

at least one processor and a memory storing:

a session SYN packet receiver, wherein the session SYN packet receiver is executed by the at least one processor to receive a session SYN packet;

a transition cookie generator, wherein the transition cookie generator is executed by the at least one processor to generate a transition cookie with the use of a

transition cookie secret key, the transition cookie comprising a time value representing the actual time;

a session SYN/ACK packet sender, wherein the session SYN/ACK packet sender is executed by the at least one processor to send the transition cookie in response to the received session SYN packet;

a session ACK packet receiver, wherein when the session ACK packet receiver is executed by the at least one processor to receive a session ACK packet, the session ACK packet including a candidate transition cookie; and

a transition cookie validator, wherein the transition cookie validator is executed by the at least one processor to determine whether the candidate transition cookie in the received session ACK packet comprises a time value representing a time within a predetermined time interval from the time the session ACK packet is received; and wherein:

the transition cookie generator is executed by the at least one processor to generate the transition cookie by (i) generating an encrypted data element of the generator by applying a cryptographic method on the transition cookie secret key and a transition cookie data element, (ii) performing an unsigned binary addition on the encrypted data element of the generator and a sequence number of a TCP header in the received session SYN packet, and (iii) storing the result in the transition cookie.

20. The system according to claim 19, wherein the transition cookie data element comprises data based on at least one of: a selective ACK, an MSS index, and a 32-bit current time of day indicated by a clock.

21. The system according to claim 19, in which the transition cookie validator is executed by the at least one processor to determine that the received session ACK packet is valid if the candidate transition cookie in the received session ACK packet comprises a time value representing a time within a predetermined time interval from the time the session ACK packet is received.

22. The system according to claim 19, in which the predetermined time interval is in the range of one to six seconds.

23. The system according to claim 19, in which the predetermined time interval is three seconds.

24. The system according to claim 19, in which the generating of the transition cookie includes the use of random data.

25. The system according to claim 19, in which the generating of the transition cookie includes the use of data obtained from the session SYN packet.

26. A system for TCP SYN cookie validation at a host server, the system comprising:

at least one processor and a memory storing:

a session SYN packet receiver, wherein the session SYN packet receiver is executed by the at least one processor to receive a session SYN packet;

a transition cookie generator, wherein the transition cookie generator is executed by the at least one processor to generate a transition cookie with the use of a transition cookie secret key, the transition cookie comprising a time value representing the actual time;

a session SYN/ACK packet sender, wherein the session SYN/ACK packet sender is executed by the at least one processor to send the transition cookie in response to the received session SYN packet;

a session ACK packet receiver, wherein the session ACK packet receiver is executed by the at least one processor to receive a session ACK packet, the session ACK packet including a candidate transition cookie; and

a transition cookie validator, wherein the transition cookie validator is executed by the at least one processor to determine whether the candidate transition cookie in the received session ACK packet comprises a time value representing a time within a predetermined time interval from the time the session ACK packet is received; and to generate:

a candidate sequence number such that a sequence number of a TCP header in the received session ACK packet equals the sum of the candidate sequence number and a value of 1,

a candidate encrypted data element such that the result of performing an unsigned binary addition of the candidate encrypted data element and a candidate sequence number equals the candidate transition cookie, and

a candidate transition cookie data element by (i) applying a cryptographic method on a candidate transition cookie secret key and the candidate encrypted data element.

27. The system according to claim 26, wherein the transition cookie validator is executed by the at least one processor to validate the candidate transition cookie data element by adjusting the candidate transition cookie data element to generate, and determining if the adjusted candidate transition cookie data element is within a predetermined time margin of a modified current time.

28. The system according to claim 26, in which when the transition cookie validator is executed by the at least one processor to determine that the received session ACK packet is valid if the candidate transition cookie in the received session ACK packet comprises a time value representing a time within a predetermined time interval from the time the session ACK packet is received.

* * * * *